Convex optimization for generalized sparse recovery

by

Ewout van den Berg

M.Sc., Delft University of Technology, The Netherlands, 2003

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

Doctor of Philosophy

in

The Faculty of Graduate Studies

(Computer Science)

The University of British Columbia

(Vancouver)

December 2009

Abstract

The past decade has witnessed the emergence of compressed sensing as a way of acquiring sparsely representable signals in a compressed form. These developments have greatly motivated research in sparse signal recovery, which lies at the heart of compressed sensing, and which has recently found its use in altogether new applications.

In the first part of this thesis we study the theoretical aspects of joint-sparse recovery by means of sum-of-norms minimization, and the ReMBo-ℓ_1 algorithm, which combines boosting techniques with ℓ_1-minimization. For the sum-of-norms approach we derive necessary and sufficient conditions for recovery, by extending existing results to the joint-sparse setting. We focus in particular on minimization of the sum of ℓ_1, and ℓ_2 norms, and give concrete examples where recovery succeeds with one formulation but not with the other. We base our analysis of ReMBo-ℓ_1 on its geometrical interpretation, which leads to a study of orthant intersections with randomly oriented subspaces. This work establishes a clear picture of the mechanics behind the method, and explains the different aspects of its performance.

The second part and main contribution of this thesis is the development of a framework for solving a wide class of convex optimization problems for sparse recovery. We provide a detailed account of the application of the framework on several problems, but also consider its limitations. The framework has been implemented in the SPGL1 algorithm, which is already well established as an effective solver. Numerical results show that our algorithm is state-of-the-art, and compares favorably even with solvers for the easier—but less natural—Lagrangian formulations.

The last part of this thesis discusses two supporting software packages: SPARCO, which provides a suite of test problems for sparse recovery, and SPOT, a Matlab toolbox for the creation and manipulation of linear operators. SPOT greatly facilitates rapid prototyping in sparse recovery and compressed sensing, where linear operators form the elementary building blocks.

Following the practice of reproducible research, all code used for the experiments and generation of figures is available online at

`http://www.cs.ubc.ca/labs/scl/thesis/09vandenBerg/`

Table of Contents

List of Tables

List of Figures

Acknowledgments

I have been very fortunate to have spent the past four years working with my supervisor Michael Friedlander. His unwaivering support, enthusiasm, and encouragement to work on topics that are interesting to me, have made working on my Ph.D. a thoroughly enjoyable experience; I couldn't have asked for a better supervisor.

There has also been plenty of fruitful interdisciplinary collaboration with Özgur Yılmaz, Felix Herrmann, and fellow students Rayan Saab, Gilles Hennenfent, and Mark Schmidt. The numerous discussions, with reasoning from different perspectives, have certainly broadened my view. It has been a privilege to work with such a fine group of people.

I would further like to thank Will Evans and Eldad Haber for being my university examiners, Richard Baraniuk for agreeing to be my external examiner, and Lara Hall for taking care of all administrative matters.

The internet has made academic life more convenient in terms of easy access to information, but has also hastened its pace due to the continuous and relentless flow of new results. Thanks to Igor Carron for writing his blog and to the digital signal processing group at Rice for maintaining the compressive sensing repository. Both resources have been very helpful at centralizing information and creating some order in the chaos.

Lab life was enlivened by the many labmates over the past several years. Many thanks to Hui, Dan, Tim, Tracy, Jelena, Liz, Kathrin, Maria, Shi-dong, Mitch, Essex, and Landon for their humor and for reminding me to get out of the lab from time to time. A special thanks also to all my friends here at UBC, who have made it a memorable stay. In particular I would like to express my gratitude to Xin-Yi, Hong-Rae, Andy, Suh, Apple, Hamed, Abhijit, Jie, and Rong-Rong for their warm friendship.

Finally, I am indebted to my parents, brother, and sister for their unconditional love and support throughout my entire study.

I would like to dedicate this thesis to my late aunt Dorret Vermeij who has always been very supportive and who would have been so proud of my thesis completion.

Ewout van den Berg

Vancouver
December 12, 2009

Chapter 1

Introduction

The main topic of this thesis is sparse recovery. Because this research area is still relatively young, we use this introductory chapter to provide a brief overview of its origins, along with recent developments. We start by noting the importance of sparse signal representations in transform-based coding. From there we move to the basis pursuit method, which aims to find sparse representations for classes of signals that cannot be represented as such in an orthonormal basis. This naturally leads to sparse recovery and ultimately to the newly established field of compressed sensing in which sparse recovery is firmly ingrained.

1.1 Signal compression and sparsity

In their uncompressed form, digital signals such as audio, video, and still images often require vast amounts of data to be stored. Take for example the 256×256 grayscale image shown in Figure 1.1(a). Without compression, such an image would require all 65,536 individual grayscale values to be stored. Far more compact (though approximate) representations of the image can be obtained by using techniques such as transform coding, which, for example, forms the basis of the ubiquitous JPEG compression scheme [138]. The storage reduction achieved by these methods is based on the fact that much of the data can be discarded without greatly degrading the appearance of the image or signal. This type of compression, which typically provides an inexact signal representation, is called lossy compression, as opposed to lossless compression. So, how exactly does it work?

Let $s \in \mathbb{R}^m$ be a vectorized representation of our signal; in this case the Euler image. If s were sparse it would suffice to store the location and value of the nonzero entries, giving a lossless compression. Because most practical signals are not exactly sparse we can choose a small positive threshold and treat all entries in s with magnitude below the threshold as zero. Alternatively we can retain a fixed number of entries that are largest in magnitude, thus giving a sparse approximate representation of the signal. The quality of the resulting signal approximation depends on the number entries stored, and how fast the magnitude of the signal entries decreases; the faster the decay, the less the loss of signal energy, and the more accurate the representation.

In transform-based coding we do not work with the signal s directly, but rather with a representation in terms of an orthonormal transformation. That is, given an orthonormal basis Φ, we represent s as $\Phi^T x$, where $x := \Phi s$. The

reason for doing so is that by judicious choice of Φ, we can affect the decay rate or compressibility of the signal. More formally, denote by $v_{(k)}$ the best k-term approximation of a vector v, zeroing out all but the k largest entries in magnitude, and with ties resolved based on a lexicographical ordering. We can then represent the signal energy retained by keeping the largest k entries of a vector v as

$$\gamma_k(v) := \|v_{(k)}\|_2 / \|v\|_2.$$

Recall that the ℓ_p-norm $\|\cdot\|_p$ for $p \geq 1$ is given by

$$\|v\|_p = \Big(\sum |v_i|^p \Big)^{1/p}. \tag{1.1}$$

Now, based on the orthogonality of Φ we can express the k-term approximation error in $s_{(k)} := \Phi^T x_{(k)}$ as

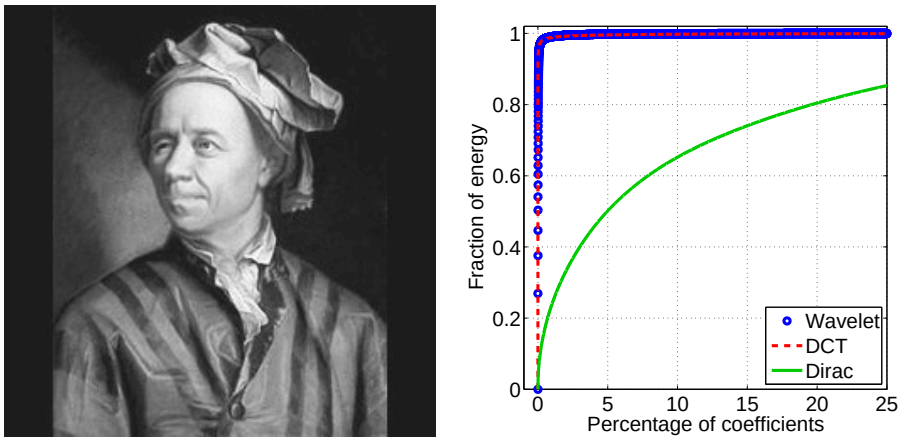
$$\|s_{(k)} - s\|_2 = \|\Phi(x_{(k)} - x)\|_2 = \|x_{(k)} - x\|_2 = \sqrt{1 - \gamma_k(x)^2} \cdot \|s\|_2,$$

where we use the fact that $\|x\|_2 = \|\Phi^T x\|_2 = \|s\|_2$, and that $x_{(k)}$ coincides with x on its support. An appropriate choice of Φ causes $\gamma_k(\Phi s)$ to increase faster, leading to a smaller approximation error for a given k, or, more interestingly, allowing a sparser representation for an equivalent approximation.

To illustrate this, we consider the compression of the Euler image using three different bases Φ: the two-dimensional Daubechies wavelet [49], the two-dimensional discrete cosine transform (DCT), and the Dirac basis (i.e., the identity basis). Figure 1.1(b) plots the value of $\gamma_k(\Phi s)$ as a function of k, with the horizontal axis representing k as a percentage of the total number of entries in s. It can be seen that the signal representation in both the wavelet and DCT transform has most of the energy concentrated in a small number of coefficients. This results in much more accurate signal representations $\Phi^T x_{(k)}$ compared to $s_{(k)} = I^T x_{(k)}$ with the same number of coefficients, as illustrated in Figure 1.2.

1.2 Basis pursuit

Given a class of signals $\mathcal{S}$, such as audio signals or natural images. For compression we would like to find a basis Φ such that $\gamma_k(\Phi s)$ is as large as possible for typical signals $s \in \mathcal{S}$. Now, suppose that the class of signals is simply too large to be accurately represented in any basis. As an example, let $\mathcal{S}_1$ and $\mathcal{S}_2$ be all signals that have a k-sparse representation in respectively the Dirac basis $\Phi_1 \equiv I$ and DCT basis Φ_2. These bases are such that signals with a sparse representation in the one are necessarily dense in the other (see for example [62]). This is illustrated in Figure 1.3, which shows on the left a signal, and on the right the corresponding DCT coefficients. The left side also shows the Dirac coefficients themselves since $\Phi_1 = I$. It is clear that the one-sparse Dirac signal in plot (a) has a fully dense representation in the DCT basis, shown in plot (b). On the other hand, the sparse DCT representation in plot (d) is fully dense in the Dirac basis (c). The combined signal, shown in plot (e) clearly has no sparse



(a) (b)

▶ Figure 1.1: Picture of Euler (a), and (b) the cumulative energy contributed by sorted coefficients in different bases.

representation in either basis, even though it only differs from (c) at the one location indicated by the arrow.

This lack of sparse representation is not so much due to the choice of basis, but a consequence of the class of signals: there simply does not exist any basis that can sparsely represent all signals in the set $\mathcal{S} := \mathcal{S}_1 + \mathcal{S}_2$. This inherent limitation of bases led people in the signal processing community to consider alternative ways of representing signals. A natural extension of bases are overcomplete dictionaries. These can be motivated by considering the representation of a signal s in basis A:

$$s = Ax = \sum_{i=1}^{n} x_i A^{\downarrow i} \tag{1.2}$$

where $A^{\downarrow i}$ denotes the ith column of A. The freedom of choosing x_i is clearly limited by the number n of columns or atoms in A; increasing n gives much more flexibility. This leads to the notion of overcomplete dictionaries, represented by $m \times n$ matrices A, with $m \leq n$.

When expressing a signal s in terms of an overcomplete dictionary, its coefficients x are no longer unique. In fact, provided that s is in the range of A, there will be infinitely many representations. This is unlike the case of orthogonal bases, or invertible matrices in general, where there is exactly one such representation. Given that we are trying to obtain a compressed representation of s, it makes sense to find the sparsest x that satisfies $Ax = s$. This can be done by solving the following problem with $b = s$:

$$\underset{x}{\text{minimize}} \quad \|x\|_0 \quad \text{subject to} \quad Ax = b, \tag{1.3}$$

where $\|x\|_0$ denotes the number of nonzero entries in x. The solution to this

▶ Figure 1.2: Reconstruction of the Euler image based on (from top to bottom) the largest 20, 5, and 2 percent of the coefficients of (from left to right) the Daubechies-8 wavelet, discrete cosine transform, and Dirac basis. The label below each figure gives the value of $\sqrt{1-\gamma_k(\Phi s)^2}$.

problem may still contain m nonzero entries, which is equal to the number of entries in s. The same typically holds true in transform-based coding, where $x := \Phi s$ is compressible, but not sparse. In the latter setting we sparsify x by discarding the smaller entries as a post-processing step. In sparsity-minimizing formulations the same can be achieved directly by either fixing the maximum sparsity level and finding x that minimizes the misfit, or by finding the sparsest x satisfying a given misfit. The second approach yields to following generalization to (1.3):

$$\underset{x}{\text{minimize}} \quad \|x\|_0 \quad \text{subject to} \quad \|Ax - b\|_2 \leq \sigma, \tag{1.4}$$

where σ is the maximum permissible misfit. Unfortunately, both (1.3) and (1.4) are combinatorial and were shown to be NP-hard by Natarajan [115].

In their seminal paper, Chen et al. [39] propose a convex relaxation of (1.3), called basis pursuit (BP):

$$\underset{x}{\text{minimize}} \quad \|x\|_1 \quad \text{subject to} \quad Ax = b. \tag{BP}$$

In contrast to (1.3), this formulation can be practically solved using techniques from convex optimization (see Chapter 4 for more details), and is therefore tractable. Another important property is that the non-differentiability of the ℓ_1-norm along the coordinate axes (where one or more entries of x are zero) induces sparsity in the solution. This property has long been known empirically in geophysics, and was already used in the 1970s by Claerbout and Muir [40], and Taylor et al. [139].

The analogous ℓ_1 relaxation of (1.4) is given by

$$\underset{x}{\text{minimize}} \quad \|x\|_1 \quad \text{subject to} \quad \|Ax - b\|_2 \leq \sigma, \tag{BP$_\sigma$}$$

which we, somewhat against convention, refer to as the basis pursuit denoise (BPDN) problem. Commonly this label is applied to the regularized basis pursuit problem introduced originally by Chen et al. [39]:

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1, \quad \text{with} \quad \lambda > 0. \tag{QP$_\lambda$}$$

This latter formulation can be interpreted as the Lagrangian form of (BP$_\sigma$), and the two formulations are equivalent in the sense that for any $\sigma > 0$ there exists a λ such that (BP$_\sigma$) and (QP$_\lambda$) have the same solution (or set of solutions if the columns in A are not in general position).

Going back to our earlier DCT-Dirac example, we construct a dictionary $A = [\Phi_1, \Phi_2]$, consisting of the atoms of both the Dirac basis Φ_1 and DCT basis Φ_2. We then solve (BP), with A and b as above, to obtain the least ℓ_1-norm coefficient vector x satisfying $Ax = b$. The resulting solution x, plotted in Figure 1.3(g), is exactly what we hoped it would be: is has a single nonzero Dirac coefficient in the first half and a single DCT coefficient in the second half. In this case we conclude that ℓ_1 minimization has in fact found the sparsest possible solution, and therefore the solution to the ℓ_0 minimization problem (1.3). A natural question then, is to ask whether this was simply a fortuitous

example or whether there are conditions under which the two formulations have an identical unique solution. It turns out that, under certain conditions, this indeed holds and we discuss this topic at length in Chapter 2. As a comparison, we also computed the least ℓ_2-norm solution satisfying $Ax = b$. This solution is entirely nonzero and given by the vector $[b; \Phi_2^T b]/2$.

1.3 Compressed sensing

From the illustration on the Euler image we see that a mere 5 to 20 percent of the wavelet coefficients suffices to accurately represent this signal, and the same applies to a large class of natural images. The field of compressed sensing (CS) [25, 58] stems from the realization that the conventional process of acquiring signals such as images is inherently wasteful. That is, for a $p \times q$ image we first record all $p \cdot q$ pixels, and then discard roughly 90 percent of the underlying coefficients during compression. With B an $n \times n$ basis in which the signals of interest have a sparse or compressible representation, the full encoding-decoding cycle is summarized as follows:

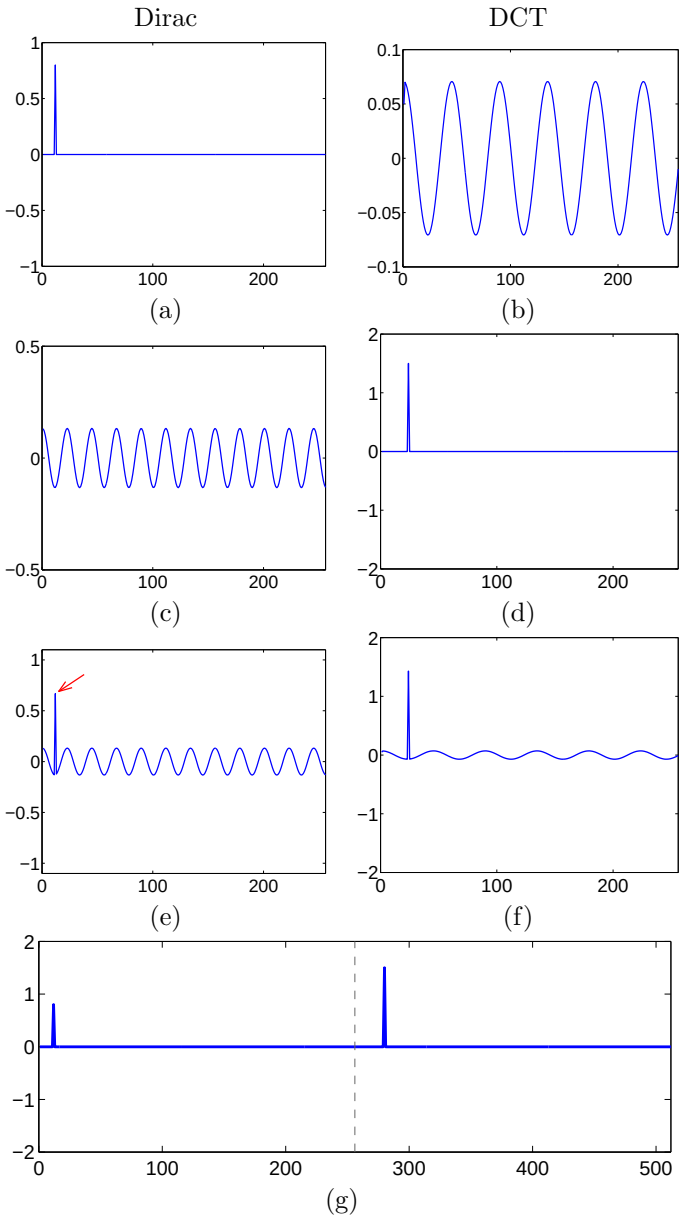
1. Measure s
2. Compute $x = \Phi s$
3. Set approximate coefficient vector $x_{(k)}$
4. Construct signal $\hat{s} = \Phi^T x_{(k)}$.

The intention behind compressed sensing is to sample the data in such a way that we only record the information necessary to reconstruct the signal [58]. Instead of measuring signal s directly, and processing $x = \Phi s$, compressed sensing proceeds by recording $b = Ms$, where M is a suitable $m \times n$ measurement matrix. By choosing $m < n$, this immediately gives a compressed measurement vector b of length m instead of n. Assuming that Φ permits a sparse (approximate) representation of s, we can decode b by finding a sparse representation x by solving (1.4) or (BP_σ), with $A = M\Phi^T$. From the resulting coefficients we can obtain the decoded signal by computing $\hat{s} := \Phi^T x$. Summarizing the above, we have the following procedure:

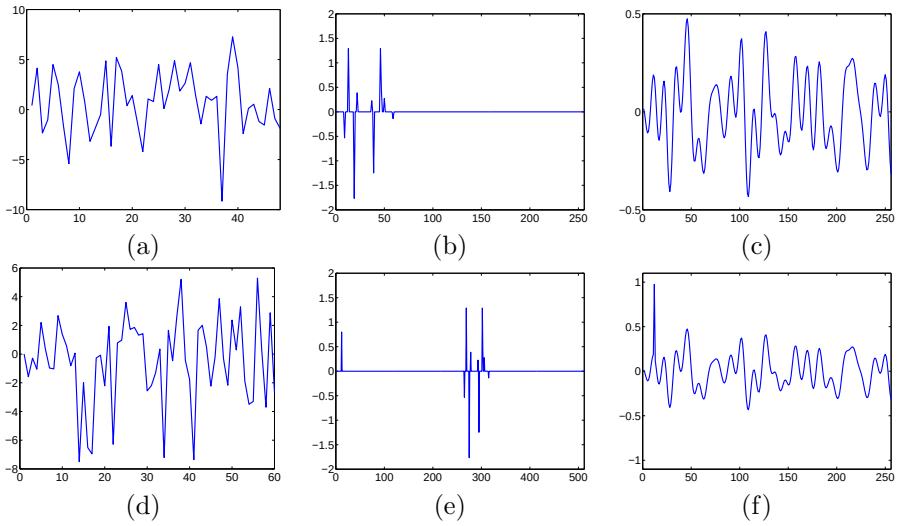
1. Measure $b = Ms$
2. Determine x by solving (BP_σ) with $A := M\Phi^T$
3. Construct signal $\hat{s} := \Phi^T x$.

In the compressed sensing approach, in contrast to the conventional scheme, most of the effort is spent in the decoding phases where we need to find a sparse solution to the system $Ax = b$. On the other hand, the encoding phase is non-adaptive and does not need to analyze the signal in order to determine the final encoding.

Interestingly, the absence of the sparsity basis Φ in the encoding phase of compressed sensing allows us to acquire a signal and choose the most suitable



▶ Figure 1.3: Plot of (a) a signal that is sparse in the Dirac basis, along with (b) its coefficients in the DCT domain. Likewise, (c) a signal that (d) has a sparse coefficients in the DCT domain, followed by (e) the summation of the signals and (f) its coefficients. Finally, (g) the minimum ℓ_1-norm representation in a dictionary containing the atoms of both bases.



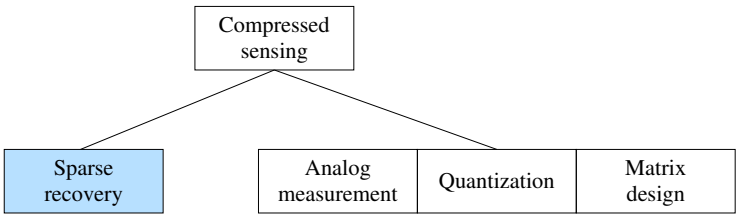
▶ Figure 1.4: Compressed sensing applied to a sinusoidal signal s using an 48×256 measurement matrix M, with (a) measurement $b = Ms$, (b) coefficients x obtained by solving (BP), and (c) recovered signal Bx. Likewise (d–f) for the same signal with a single impulse, using an overcomplete sparsity basis B and a 60×256 measurement matrix M.

basis Φ, or even overdetermined dictionary B, during the decoding phase. The measurement matrix, on the other hand, is used and therefore needs to be fixed.

To illustrate the compressed sensing methodology, we apply it to the sparse DCT signal of Figure 1.3(c). This is done by first generating a 48×256 measurement matrix M with its entries randomly drawn from the normal distribution. We then compute the observation vector $b = Ms$, which is plotted in Figure 1.4(a). Because we know that the original signal s has an exact sparse representation we then solve the basis pursuit formulation with $A := MB$, and $B^T = \Phi_2$ the DCT basis. This gives the coefficient vector x shown in Figure 1.4(b). The decoded signal obtained by computing Bx is shown in plot (c), and in this case corresponds exactly to the original signal.

We now consider taking compressed measurements of the mixed DCT-Dirac signal of Figure 1.3(e). We choose $B = [\Phi_1^T, \Phi_2^T]$ as the sparsity basis[1], and draw a random 64×256 measurement matrix M. Proceeding as above, this gives the measurement, coefficient, and decoded signal vectors given in Figure 1.4(d–f). Due to the existence of a exact sparse coefficient vector and the fact that m is sufficiently large, we again recover original signal s.

[1] In this context we prefer to use the term 'basis' rather than 'dictionary', despite the fact that the underlying matrix is nonsquare.



▶ Figure 1.5: Graphical representation of the different aspects of compressed sensing. In this thesis we will predominantly be concerned with the highlighted (shaded) topic of sparse recovery.

1.3.1 Practical issues

Besides improved theoretical recovery conditions, and faster sparsity-promoting solvers, there are a number of important practical issues in compressed sensing that we have so far ignored. These are summarized in Figure 1.5 and discussed below.

Taking measurements

Recall that the idea behind compressed sensing is to sample only a limited number of indirect measurements, rather than the entire signal s. This is done by taking inner products of x with a set of vectors or function $\{f_1, \ldots, f_m\}$, i.e.,

$$b_i = \langle f_i, s \rangle. \tag{1.5}$$

For decoding we construct a measurement matrix M, whose rows represent discretized versions of f_i. That is, $M^{i\rightarrow} := \hat{f}_i$, where $M^{i\rightarrow}$ denotes the (column) vector corresponding to the ith row of M. The inner product in (1.5) cannot be digitally evaluated as $\sum_j M_{i,j} s_j$, because this requires access to all individually sampled entries of s_j, which defeats the purpose of CS. Therefore, for compressed sensing to work in practice, these inner products must somehow be evaluated by means of an analog process. At present there has not been done a lot of work on how to solve this issue. There are, however, a number of specific settings in which this can be done efficiently. We will discuss some of these in Section 1.3.2.

Quantization

Once analog measurements have been made, they need to be quantized before they are suitable for digital processing. This can be done, for example, by taking a codebook $\mathcal{C}$ of discrete values and mapping each analog value v_i to the nearest element in $\mathcal{C}$:

$$b_i \in \operatorname*{argmin}_{\kappa \in \mathcal{C}} \quad \|v_i - \kappa\|,$$

breaking ties in some predetermined manner. This step in the encoding of the compressed signal obviously affects the decoding process and reduces the accuracy with which the original signal can be reconstructed. Recent work by

Dai et al. [44], Goyal et al. [82], and Boufounos and Baraniuk [19] considers the theoretical implications of quantization in compressed sensing. Meanwhile, Zymnis et al. [156] and Jacques et al. [97] have proposed ways of explicitly incorporating quantization into numerical solvers. One alternative way of dealing with quantization in solvers is to model the observation as $b = v + e$, where e represents a vector of additive noise due to quantization. However, this excludes prior information about the measurement process and is unlikely to achieve the best possible results.

Measurement matrix

Because of limitations on analog hardware [20], only certain measurement 'matrices' can be used. In the larger context of sparse recovery using basis pursuit or equivalent formulations, however, there is much more freedom. Properties of A that are important include the amount of data required to store the matrix, whether or not it facilitates an implicit representation through a fast routine that evaluates its matrix-vector products, and most importantly, whether or not it satisfies conditions that guarantee near or exact recovery for certain types of signals.

In the course of Chapter 2, which discusses theoretical results in sparse recovery, we mention known matrix classes that satisfy given conditions. Typically these matrices have probabilistic constructions and satisfy conditions only with high probability. Recently a number of explicit matrix constructions that are guaranteed to satisfy certain equivalence conditions have been proposed by DeVore [53]. For a survey of known constructions and theoretical guarantees, see Berinde et al. [14].

Sparse recovery

A crucial component of compressed sensing is an algorithm for the recovery, or decoding, of coefficient vector x from the compressed measurement b. When x is assumed to be sparse, this can be done using sparse recovery techniques, such as basis pursuit denoise. In this thesis we are predominantly concerned with sparse recovery using convex optimization. Various approaches are reviewed in Chapter 4. In Chapter 5 we develop a new convex optimization framework for sparse recovery. Besides these methods, a number of sparse recovery algorithms have been proposed that work only in combination with particular measurement matrices. We forego a discussion of these types of algorithms and refer instead to Indyk [96], Jafarpour et al. [98], and Howard et al. [95].

1.3.2 Examples

There are a number of settings where the compressed acquisition of signals corresponding to (1.5) can be done in analog. We discuss some of these applications in the following three sections.

MRI and Angiography

The most natural setting for compressed sensing to date is in magnetic resonance imaging (MRI). This application was first suggested by Candès et al. [32] and was further developed by Lustig et al. [105]. The latter article also gives an excellent introduction to the topic.

The key property of MRI that makes it fit the CS framework is that the signal of interest is measured directly in the frequency domain, i.e., the k-space. This means that instead of sampling the image directly, we instead sample points in k-space. As an illustration of this we consider the modified Shepp-Logan phantom [141, p.199], depicted in Figure 1.6(a). The k-space representation of this image is shown in plot (b) and it is from this space that we obtain our measurements. Time restrictions on the scans prevent us from sampling the entire space and we can only sample along trajectories though k-space. These trajectories are limited by physiological and hardware constraints [105], and typical trajectories have the form of straight lines, radial lines, or spirals. For our example we ignore such restrictions and use the random pattern given in Figure 1.6(c). This pattern takes into account the fact that most of the image energy in k-space is concentrated close to the center.

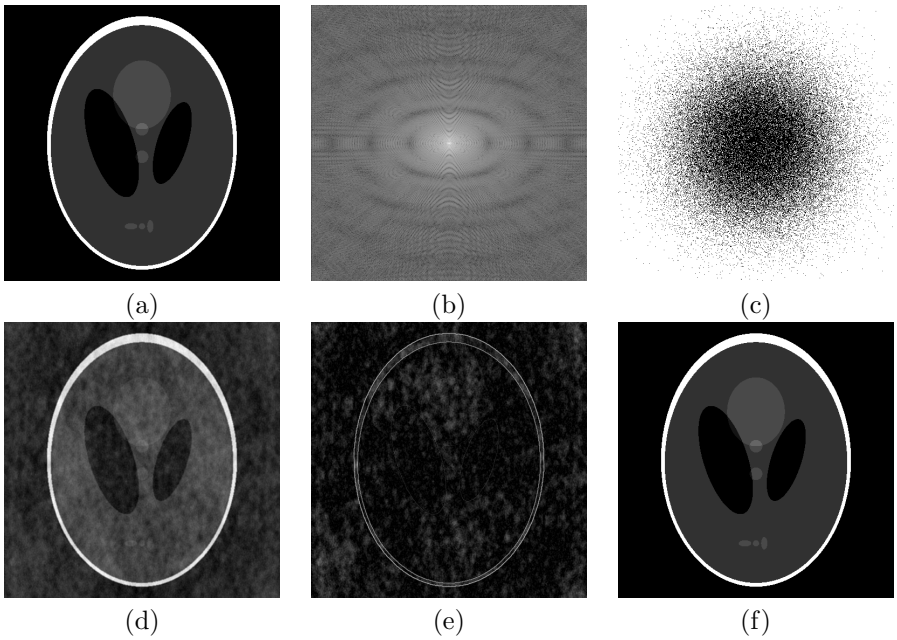
Mathematically, let s represent the image, F the two-dimensional Fourier transform, and R a restriction matrix corresponding to the frequencies measured. Then the observed signal is given by

$$b = RFs.$$

Setting $M = RF$ gives us our measurement matrix which represents the actual sampling process used in practice; conversion to the digital domain does not happen until after inner-products with the rows of the matrix have been evaluated.

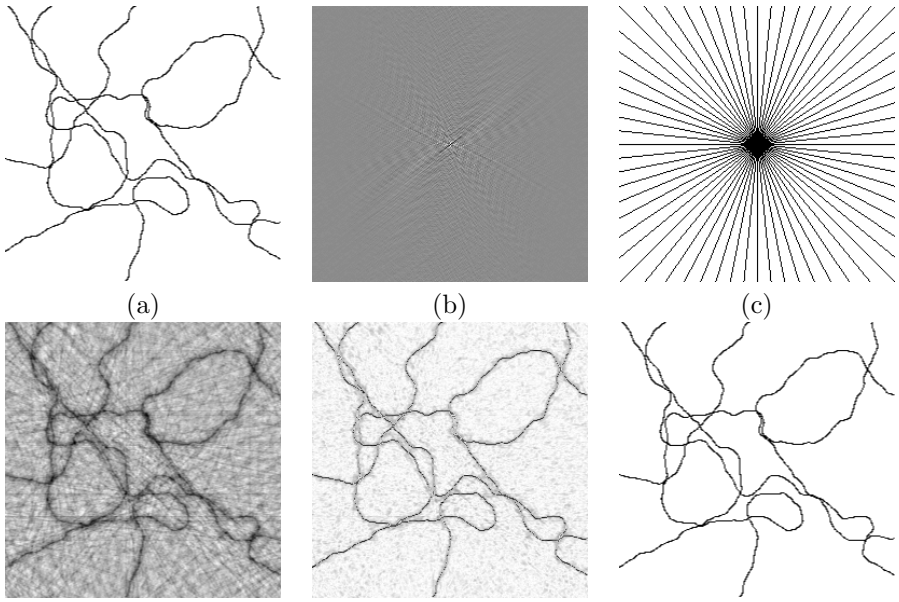
Given the partial-Fourier coefficients, the easiest way to obtain an image is using filtered backprojection [32]. In this approach we simply compute $\hat{s} = F^* R^T b$, and essentially assume that all missing coefficients are zero. The result of this operation is shown in Figure 1.6(d). While fast, this method often introduces severe undersampling artifacts, as shown by plot(e), which gives the error in the reconstruction.

Before we can apply sparse recovery for the reconstruction of the signal, we first need a sparsity basis B. Images obtained through magnetic resonance (MR) often have sparse representations in a wavelet basis. For our example we use the sparsifying transform similar to that used in the JPEG-2000 standard [138]: we divide the image into 32×32 blocks and apply a two-dimensional Haar wavelet transformation to each block. Letting B represent this operator, we can apply basis pursuit with $A := MB = RFB$ to find coefficients x and signal $\hat{s} = Bx$. When applied to the MRI example this gives an exact recovery of the phantom image, as shown in Figure 1.6(f). An alternative way of recovering MR images is to use of total-variation minimization [30, 104], which takes advantage of the fact that the gradient of these images is often sparse.



▶ Figure 1.6: Plots of (a) 512×512 modified Shepp-Logan phantom, (b) logarithmic k-space representation, (c) sampling mask covering 20% of all coefficients, (d) reconstruction using filtered backprojection, (e) error in reconstruction, and (f) exact reconstruction using basis pursuit.

Another example of the use of CS in medical imaging is the visualization of blood vessels and organs using angiograms. The use of contrast agents injected into the body prior to the imaging process causes angiograms to be inherently sparse in the pixel representation [105]. Figures 1.7(a,b) respectively show a synthetic angiogram and its representation in k-space. Sampling along the set of radial lines in plot (c) followed by backprojection gives the result shown in (d). Using compressed sensing, in this case with the sparsity basis set to the identity matrix, we recover the sparse image shown in plot (f).



▶ Figure 1.7: Plot of (a) 256×256 synthetic angiogram in reverse color, (b) k-space representation, (c) sampling pattern along 32 radial lines giving 12.1% of all coefficients, (d) reconstruction using filtered backprojection, (e) error in reconstruction, and (f) sparse reconstruction using basis pursuit.

Single-pixel camera

A second example of the application of compressed sensing is the single-pixel camera architecture suggested by Takhar et al. [137]. Before looking at this new architecture, let us look at the imaging steps of conventional digital cameras. The key component of these cameras is a sensor array that converts photons at visual wavelengths into electrons [66]. An image can be formed by opening the shutter for a brief period of time and measuring the amount of light that hits each sensor. For more advanced imaging systems, requiring observations beyond the visual wavelengths, sensors become more complicated and much

larger. This typically means that fewer sensors are available, thus leading to a reduced imaging resolution.

The single-pixel architecture overcomes this problem by using a digital micromirror device to reflect part of the incident light beams onto a single sensor, which records the total reflected light intensity. A number of such measurements are taken with different mirror orientations. The micromirror device consists of an array of tiny mirrors that can each be oriented at two different angles. At one angle it reflects light away from the sensor and at the other angle it reflects light onto the sensor. Let s be the light intensities impeding upon the mirrors and m be a vector that contains 0 or 1 depending on whether the corresponding mirror is positioned away or towards the sensor respectively. Then the amount of light recorded by the sensor is given by $m^T s$. This is exactly of the form (1.5), and by changing the mirror orientations, we effectively create the rows of a measurement matrix M. Note that by changing the position of the mirror during a measurement we can effectively create arbitrary weights between 0 and 1. Negative values can be obtained by subtracting one measurement vector from another.

DNA mircoarrays

DNA microarrays are used for microbe detection and classification, and are made up of a number of genetic sensors or spots containing short DNA oligonucleotides called probes [132]. These probes are chosen to complement sequences in the target DNA which is tagged with fluorescent labels. The relative abundance of the sequences in the target is determined by flushing the target sample against the array and letting the complementary strands bind or hybridize. The remaining solvent is then washed away and the array read out by measuring the illumination pattern at each spot [45].

In conventional DNA microarrays it is important that each sensor is carefully designed to respond to a single target and to avoid cross-hybridization with similar but non-complementary sequences. For the compressed sensing DNA microarray introduced by Sheikh et al. [132], this requirement is relaxed and each spot is designed to respond to a larger group of targets. The measurement matrix M consists of entries m_{ij}, which represent the probability that target j hybridizes with spot i. The inner product of each row with the sample corresponds to the illumination intensity measured at the associated spot. Because we can expect each sample to contain only a limited number of target agents, no sparsity basis is required.

1.4 Generalized sparse recovery

The use of sparse recovery is not limited to compressed sensing. Instead there are a number of applications that can be formulated as (1.3), or, if some level of misfit between Ax and b is allowed, as (1.4).

By choosing an appropriate A we can, for example, apply the framework

to data interpolation or inpainting [71]. In these problems we are given an incomplete data set, and the goal is to complete the data set in such a way that it is compatible with the known data. That is, letting s be the original complete signal, and R be a restriction matrix, we observe $b = Rs$. By choosing a sparsity basis B we can set $A = RB$, and use basis pursuit to recover a sparse x satisfying $b = Ax$. This may be seen as another application of compressed sensing by interpreting R as a measurement matrix, but it seems to have a wider applicability than compressed signal acquisition.

For our first example of interpolation we go back to the sinusoidal signal of Figure 1.3(c). In that example we created a 48×256 restriction matrix consisting of rows with a single nonzero value at the sampled location. The resulting measurements, plotted at their original location, are shown in Figure 1.8(b). Basis-pursuit recovery of the entire signal using $A = R\Phi_2^T$, with Φ_2 the DCT transform, gives an exact reconstruction of the original signal, shown in (c).

Interpolation of seismic data with missing traces using sparse recovery was proposed by Hennenfent and Herrmann [91, 92]. Such missing data can be due to the breakdown of geophones (sensors), or the physical difficulty of placing them at desired locations. A full synthetic data set, courtesy of Eric Verschuur, is shown in Figure 1.8(d). The restriction operator in this example is such that it retains only a small number of columns for the vectorized image, as illustrated in Figure 1.8(e). Accurate recovery is possible due to the fact that seismic images are highly compressible in the curvelet domain [27, 92]. The signal recovered using basis pursuit is shown in Figure 1.8(f).

Applications that are less related to compressed sensing often require different formulations. Below we will introduce convex relaxations of three different sparse-recovery formulations that are discussed later in this thesis.

1.4.1 Nonnegative basis pursuit

The reason for the success of basis pursuit in recovering sparse vectors is due to the fact that ℓ_1-norm minimization quite capably captures the prior information that the solution be sparse.

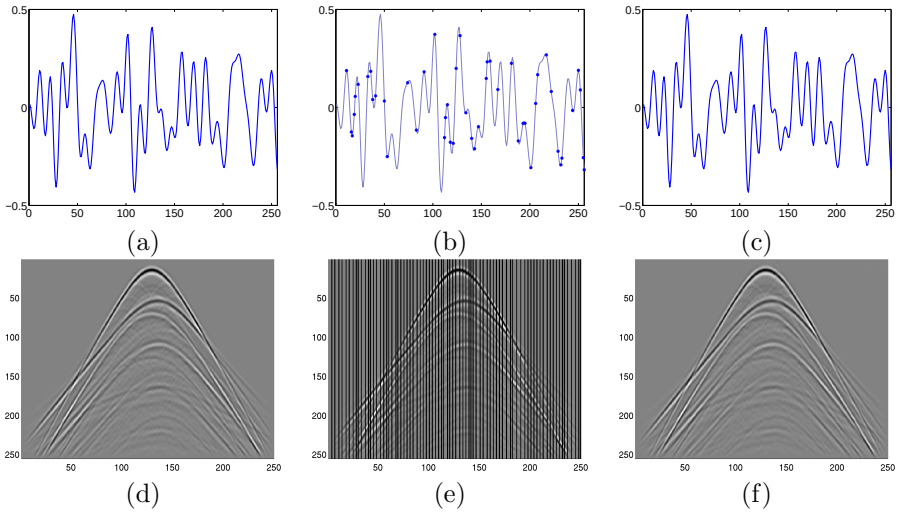
By adding more prior information on the coefficients we can reduce the feasible set of solutions, thus making recovery more likely. One such prior is to require x to be nonnegative; this is the case, for example, in angiogram images, where it is physically impossibility to have negative values. Modifying the basis pursuit formulation accordingly gives

$$\underset{x}{\text{minimize}} \quad e^T x \quad \text{subject to} \quad Ax = b,\ x \geq 0, \tag{1.6}$$

and likewise for basis pursuit denoise:

$$\underset{x}{\text{minimize}} \quad e^T x \quad \text{subject to} \quad \|Ax - b\|_2 \leq \sigma,\ x \geq 0.$$

The same formulation can be used to incorporate specific sign restrictions on each coefficient, by appropriately redefining the columns of A.



▶ Figure 1.8: Reconstruction of missing data, showing (a) the original signal, which has a 10-sparse DCT representation, (b) observed data, obtained by applying a 48×256 restriction matrix (81.25% missing), and (c) exact recovery. For (d) a seismic data set with (e) 32% of traces (columns) missing, along with (f) the interpolated result. Seismic data set courtesy of Eric Verschuur.

1.4.2 Group and joint-sparse recovery

In areas such as neuroimaging using functional MRI or magnetoencephalography [80], sparsity arises predominantly in fixed patterns: certain parts of the brain that are associated with different functions are mostly active simultaneously [42]. In this type of sparsity we are not so much interested in sparsity within functional groups but more in sparsity in terms of entire groups. This can be achieved by grouping the variables into disjoint sets Λ_i, and solving

$$\underset{x,s}{\text{minimize}} \quad \|s\|_0 \quad \text{subject to} \quad \|Ax - b\|_2 \leq \sigma, \ s_i = \|x_{\Lambda_i}\|,$$

for any norm on x_{Λ_i}. This can be relaxed to a convex formulation by writing

$$\underset{x}{\text{minimize}} \quad \sum_i \|x_{\Lambda_i}\|_2 \quad \text{subject to} \quad \|Ax - b\| \leq \sigma. \tag{1.7}$$

A special case of group sparsity arises in the situation where we are given a number of measurement vectors $B^{\downarrow i}$, $i = 1, \ldots, k$, such that the coefficients $X^{\downarrow i}$ are known to be jointly sparse, i.e., have nonzero entries clustered in a small number of rows. This joint-sparse problem can be rewritten as a group-sparse problem by vectorizing the coefficients, grouping the entries in each row into the same set, and setting A to $I \otimes A$, where $\otimes$ denotes the Kronecker product. A more convenient formulation for the joint-sparse problem, however, is given

by

$$\underset{X \in \mathbb{R}^{n \times k}}{\text{minimize}} \quad \|X\|_{1,2} \quad \text{subject to} \quad \|AX - B\|_F \leq \sigma, \tag{1.8}$$

where $\|X\|_{1,2}$ is a specific instance of the $\ell_{p,q}$-norm, defined by

$$\|X\|_{p,q} = \Big(\sum_i \|X^{i\rightarrow}\|_q^p \Big)^{1/p}. \tag{1.9}$$

Formulations other than (1.8) are possible: Tropp [145] suggests the use of $\ell_{1,\infty}$, whereas Cotter et al. [42] adopt $\ell_{p,2}$, with $p \leq 1$. In Section 6.2 we propose an algorithm for solving the $\ell_{1,2}$ formulation, and in Section 3.4 study the conditions for sparse recovery using $\ell_{1,2}$-minimization:

$$\underset{X}{\text{minimize}} \quad \|X\|_{1,2} \quad \text{subject to} \quad AX = B, \tag{1.10}$$

which is a special case of the $\ell_{p,q}$-minimization problem

$$\underset{X}{\text{minimize}} \quad \|X\|_{p,q} \quad \text{subject to} \quad AX = B. \tag{1.11}$$

1.4.3 Low-rank matrix reconstruction

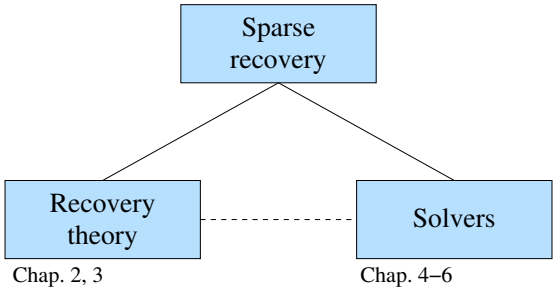
An altogether different area in which convex relaxation can sometimes be used to solve otherwise intractable problems is in low-rank matrix completion. In this problem we are given an incomplete $m \times n$ matrix B, of which only the entries $(i, j) \in \Omega$ are known. Based on this information we want to reconstruct the missing entries. Assuming that B is low-rank, the ideal—but unfortunately NP-hard [29]—formulation for this is

$$\underset{X \in \mathbb{R}^{m \times n}}{\text{minimize}} \quad \text{rank}(X) \quad \text{subject to} \quad X_\Omega = B_\Omega. \tag{1.12}$$

Fazel [75] proves that the nearest convex relaxation of the rank is given by the nuclear norm $\|X\|_*$ of X. This norm is a specific instance of the Schatten p-norm, and is defined as the sum of the singular values of X. Applying this convex relaxation to (1.12) leads to the nuclear norm minimization problem [29]

$$\underset{X}{\text{minimize}} \quad \|X\|_* \quad \text{subject to} \quad X_\Omega = B_\Omega. \tag{1.13}$$

An interesting analogy between nuclear-norm minimization and basis pursuit, pointed out by Recht et al. [126], is that both formulations replace minimization of cardinality by minimization based on the ℓ_1-norm. That is, basis pursuit relaxes $\|x\|_0$ to $\|x\|_1$, whereas nuclear-norm minimization relaxes $\|\sigma\|_0$ to $\|\sigma\|_1$, where σ denotes the vector of singular values of X. We further discuss nuclear norm minimization in Section 6.4.



▶ Figure 1.9: Graphical representation of the two main components of sparse recovery: recovery theory and numerical solvers. The dotted line indicates the dependency between theory and solvers in the case of, for example, greedy recovery algorithms.

1.5 Thesis overview and contributions

As illustrated in Figure 1.9, there are two major aspects to sparse recovery. On the one hand, there is the theoretical aspect, which studies the precise conditions under which recovery is successful, and the inherent limitations of different approaches. On the other hand, there is the practical aspect, in which algorithms are developed to solve sparse recovery problems or their convex relaxation.

The first half of this thesis is concerned with the theoretical aspects of sparse recovery. Chapter 2 provides a survey of current results and places a particular emphasis on the different techniques used to establish these results. We illustrate the meaning of various theoretical results with empirical simulations, and connect different results where possible. Chapter 3, whose contents have been submitted for publication [8], studies joint-sparse recovery using two classes of algorithms. For this we extend the results in Chapter 2, and leverage the geometrical interpretation of basis pursuit.

In the second half of this thesis we focus on sparse recovery algorithms. We give a comprehensive overview of existing approaches in Chapter 4. The main contribution of this thesis is contained in Chapters 5 and 6, which provide a new algorithmic framework for solving a wide class of convex formulations for sparse recovery. Parts of these chapters have been published in [10, 131], and the widely-used software implementation, SPGL1, is freely available online [9]. The development of SPGL1 was motivated by the lack of feasible alternatives, and was the first algorithm specifically designed to solve the basis pursuit denoise formulation (BP_σ). This is in contrast to most other algorithms, which apply only to the simpler penalized formulation (QP_λ). SPGL1 can now solve many more problems, and we compare its performance against other state-of-the-art algorithms in Chapter 7. We conclude with Chapter 8, in which we present a test problem suite for sparse recovery, published in [12], as well as the linear-operator toolbox, SPOT [11], that was used for its implementation.

Chapter 2

Theory behind sparse recovery

In this chapter we review theoretical results on the sparse recovery properties of basis pursuit and basis pursuit denoise: the conditions under which the solution of (BP) coincides with the sparsest solution obtained using (1.3); necessary and sufficient conditions for (BP) to recover x_0 from $b = Ax_0$; probabilities with which arbitrary vectors x_0 are recovered; and bounds on the approximation error when applying basis pursuit or basis pursuit denoise on noisy observations. Results regarding the exact recovery of vectors x_0 using basis pursuit can be classified according to the class of vectors they apply to: individual vectors, vectors with nonzero entries at fixed locations, or uniform recovery, concerning the recovery of all vectors x_0 with a given maximum number of nonzero entries.

Since the publication of the seminal paper by Donoho and Huo [62] there has been a deluge of theoretical results and it would be far beyond the scope of this thesis to discuss them all. So instead, we focus here on the tools and techniques used to establish recovery conditions and, at the risk of being incomplete, only mention some typical results as an illustration. Figure 2.1 summarizes the techniques discussed in this chapter and highlights the techniques (see shaded boxes) that will be used in Chapter 3 to derive equivalence conditions for joint-sparse recovery using (1.10).

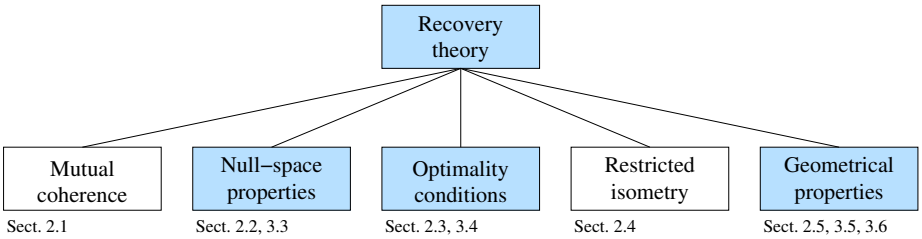
It is worth emphasizing that the strength of the results surveyed in this chapter lies in the fact that they hold for a specific problem formulations (predominantly basis pursuit). As such, they continue to hold irrespective of the algorithm used to solve the problem. This is in stark contrast to greedy approaches and other heuristics, where formulations lack and results are intricately connected to the algorithm.

2.1 Mutual coherence

The very first equivalence results derived by Donoho and Huo [62] were based on the mutual coherence, or similarity of the atoms in dictionary A. Assuming A has unit-norm columns, the mutual coherence is defined as

$$\mu(A) = \max_{i \neq j} |\langle A^{\downarrow i}, A^{\downarrow j} \rangle|,$$

and is one of the few sparse-recovery metrics that can be computed for a given matrix A in reasonable time. For the special case where A consists of a pair



▶ Figure 2.1: Overview of the most common tools and techniques used to derive and express recovery results for basis pursuit.

of orthonormal bases, Elad and Bruckstein [69] show that a sufficient condition for ℓ_1 to uniquely recover all r-sparse vectors x from $b = Ax$ is that

$$r < \frac{\sqrt{2} - 1/2}{\mu(A)}.$$

This was later shown to be both necessary and sufficient by Feuer and Nemirovski [76]. For general dictionaries, Donoho and Elad [60], and Malioutov et al. [108] derive the sufficient condition

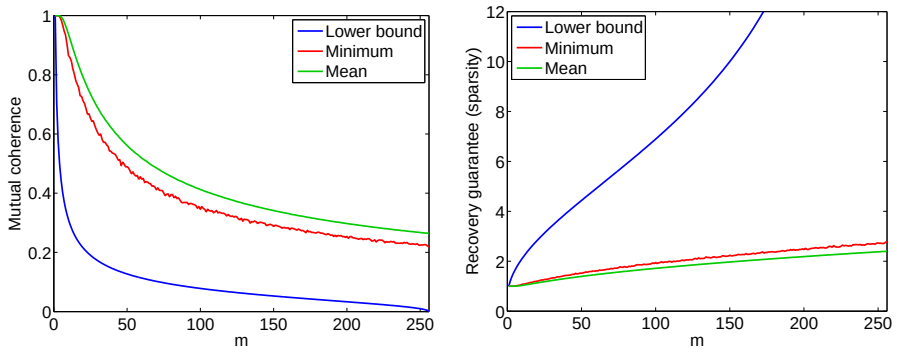
$$r < \frac{1 + 1/\mu(A)}{2}. \tag{2.1}$$

Unfortunately, the use of mutual coherence typically gives very pessimistic recovery guarantees. As an illustration, we determine the average and minimal mutual coherence based on ten thousand $m \times 256$ matrices with random Gaussian entries and columns normalized to one. The resulting values, as a function of m, are plotted in Figure 2.2(a). By applying (2.1) to these mutual coherence values we obtain the recovery guarantees shown in Figure 2.2(b). Note that based on the lower mutual coherence of near-square matrices, recovery is guaranteed for vectors that are at most 2-sparse.

Observe that the mutual coherence of A is equal to the largest absolute off-diagonal entry in the gram matrix $A^T A$. This allows us to directly apply the results by Rosenfeld [128] as a lower bound for the mutual coherence of $m \times n$ matrices A with unit norm columns (see also Strohmer and Heath [136]):

$$\mu(A) \geq \sqrt{\frac{n - m}{m(n - 1)}}.$$

Substituting this lower bound into (2.1) gives the best possible uniform recovery guarantee based on mutual coherence, as shown in Figure 2.2(b). However, even under these ideal conditions, the sparsity level for which recovery is guaranteed is still very low, at least for $m \ll n$. To weaken the conditions, Tropp [144, 146] and Donoho and Elad [60] have proposed extensions and generalizations on mutual coherence, but they are seldom used.



▶ Figure 2.2: (a) Mutual coherence for random Gaussian $m \times 256$ matrices with columns normalized to unit norm, and (b) the resulting recovery guarantees.

The use of mutual coherence is not limited to the analysis of basis pursuit. For example, Tropp [146] uses mutual coherence to characterize the recovery error obtained using (QP_λ) for a given λ. Donoho et al. [61] consider the recovery of r-sparse x_0 from $b = Ax_0 + v$ using (BP_σ). Assuming $\|v\|_2 \leq \epsilon \leq \sigma$, they show that the solution x^* of (BP_σ) satisfies

$$\|x^* - x_0\|_2^2 \leq \frac{(\epsilon + \delta)^2}{1 - \mu(A) \cdot (4r - 1)},$$

provided that $r < (1/\mu(A) + 1)/4$. Mutual coherence is also useful in deriving bounds on other metrics that are used to characterize sparse recovery.

2.2 Null-space properties

Recall that the kernel, or null-space, of a matrix A is defined as

$$\text{Ker}(A) = \{z \mid Az = 0\}.$$

Theoretical results based on null-space properties generally characterize recovery for the set of all vectors x_0 with a fixed support, which is defined as

$$\text{Supp}(x) = \{j \mid x_j \neq 0\}.$$

We say that x_0 can be uniformly recovered on $\mathcal{I} \subseteq \{1, \ldots, n\}$ if all x_0 with $\text{Supp}(x_0) \subseteq \mathcal{I}$ can be recovered. The following theorem illustrates typical null-space conditions for uniform recovery via ℓ_1 on an index set.

Theorem 2.1 (Donoho and Elad [60], Gribonval and Nielsen [85])**.** *Let A be an $m \times n$ matrix and $\mathcal{I} \subseteq \{1, \ldots, n\}$ be a fixed index set. Then all $x_0 \in \mathbb{R}^n$ with $\mathrm{Supp}(x_0) \subseteq \mathcal{I}$ can be uniquely recovered from $b = Ax_0$ using basis pursuit* (BP) *if and only if for all $z \in \mathrm{Ker}(A) \setminus \{0\}$,*

$$\sum_{j \in \mathcal{I}} |z_j| < \sum_{j \notin \mathcal{I}} |z_j|. \tag{2.2}$$

That is, the ℓ_1-norm of z on $\mathcal{I}$ is strictly less than the ℓ_1-norm of z on the complement $\mathcal{I}^c$.

The essential observation behind this result is that any feasible x satisfying $Ax = b$ can be written as $x = x_0 + z$, where z is a vector in the null-space of A. For equivalence and uniqueness we then require that x_0 has an ℓ_1-norm that is strictly less than all other vectors x. This then leads to the sufficiency of the condition. The necessity follows from the construction of a counter example x_0 from any $z \in \mathrm{Ker}(A)$ not satisfying (2.2). Gribonval and Nielsen [85, 86] show a more general relation for recovery using ℓ_p minimization

$$\underset{x}{\text{minimize}} \quad \|x\|_p \quad \text{subject to} \quad Ax = b. \tag{2.3}$$

with $0 \leq p \leq 1$. For this they define the ℓ_p-concentration of all non-trivial vectors in the null-space of A on a given index set $\mathcal{I}$ as

$$C_p(A, \mathcal{I}) := \underset{z \in \mathrm{Ker}(A) \setminus \{0\}}{\text{maximize}} \quad \frac{\sum_{i \in \mathcal{I}} |z_i|^p}{\sum_{i \notin \mathcal{I}} |z_i|^p}.$$

They then use this quantity to characterize recovery of x_0, with $\mathrm{Supp}(x_0) \subseteq \mathcal{I}$, from $b = Ax_0$ by means of (2.3) as follows:

$$C_p(A, \mathcal{I}) \begin{cases} < 1/2: & \text{all } x_0 \text{ can be uniquely recovered;} \\ = 1/2: & \text{all } x_0 \text{ are solutions of (2.3) but may not be unique;} \\ > 1/2: & \text{there are vectors } x_0 \text{ that can not be recovered.} \end{cases}$$

Uniform sparse recovery. Donoho and Elad [60] derive conditions for uniform recovery using ℓ_0 minimization (1.3) based on the spark of A. This quantity is defined as the smallest number of columns in A that are linearly dependent. When a set of columns $\mathcal{I}$ is linearly dependent, there exists a vector z such that $A_{\mathcal{I}} z = 0$, and therefore the spark coincides with the sparsest possible nonzero vector in the null-space of A:

$$\mathrm{Spark}(A) = \min \{\|z\|_0 : z \in \mathrm{Ker}(A) \setminus \{0\}\}.$$

It then follows from the ℓ_0-concentration that a necessary and sufficient condition for the uniform recovery of all r-sparse vectors is that

$$r < \mathrm{Spark}(A)/2. \tag{2.4}$$

There are currently no algorithms that can evaluate the spark of a matrix in a tractable way; checking if $\text{Spark}(A) > s$ would require taking, for example, the QR-factorization of $A_{\mathcal{I}}$ for all possible subsets $\mathcal{I} \subseteq \{1, \ldots, n\}$ of cardinality s. A lower bound on the spark, in terms of mutual coherence, albeit very pessimistic, is given by

$$\text{Spark}(A) > 1/\mu(A).$$

This follows from the diagonal dominance, and therefore invertibility, of the Gram matrix $A_{\mathcal{I}}^T A_{\mathcal{I}}$ [60]. On the other hand, note that an $m \times n$ matrix A with columns drawn independently and uniformly at random from the Euclidean sphere, due to their expected general position, gives $\text{Spark}(A) = \min\{m, n\} + 1$ with probability one. (A set of vectors in $\mathbb{R}^m$ are said to be in general position if no $k+1$ of them support a $(k-1)$-dimensional hyperplane, for any $k \in [0, m]$.)

2.3 Optimality conditions

For convex optimization problems, the Karush-Kuhn-Tucker (KKT) optimality conditions give both necessary and sufficient conditions for a point to be optimal (see, e.g., [121, Ch.12]). From these conditions, some of the most precise recovery results can be derived.

Applied to the basis-pursuit formulation, the KKT conditions state that x is a solution of (BP) if and only if x is feasible and there exists a Lagrange multiplier y satisfying

$$(A^T y)_i \in \begin{cases} \text{sign}(x_i) & \text{if } i \in \text{Supp}(x); \\ [-1, 1] & \text{otherwise.} \end{cases} \tag{2.5}$$

A necessary and sufficient condition for x to be the unique minimizer is that there exists a y that, in addition, satisfies

$$|(A^T y)_i| < 1, \text{ for all } i \notin \text{Supp}(x). \tag{2.6}$$

Interestingly, this condition reveals a very important property of ℓ_1 minimization, namely that recoverability depends only on the support of x and the sign of these nonzero values; if the above relation holds for one such x it will hold for all others as well. This granularity is much finer than the r-sparse uniform recovery results obtained using mutual coherence, or the uniform recovery on a support by null-space properties of A.

For sufficiently sparse x we can use the Moore-Penrose pseudoinverse to construct vectors y that, by construction, satisfy the first condition in (2.5). Recalling that the pseudoinverse for full-rank matrices M is defined as

$$M^\dagger = \begin{cases} (M^T M)^{-1} M^T & \text{if } M \text{ has independent columns;} \\ M^T (M M^T)^{-1} & \text{if } M \text{ has independent rows,} \end{cases} \tag{2.7}$$

we proceed as follows. Denote by $\mathcal{I}$ the support of x, let $s = \text{sign}(x_{\mathcal{I}})$ be the sign of the on-support entries, and define $M = A_{\mathcal{I}}^T$. Then the choice $y = M^\dagger s$

satisfies $A_{\mathcal{I}}^T y = MM^\dagger s = s$. A sufficient condition for recovery is then that (2.6) holds for all i not in the support of x. This is formalized in the following theorem.

Theorem 2.2. *Let $\mathcal{I} \subseteq \{1, \dots, n\}$ be an index subset with $|\mathcal{I}| \leq \operatorname{rank}(A)$ and denote $M = A_{\mathcal{I}}^T$. Then all x supported on $\mathcal{I}$ with sign pattern $\operatorname{sign}(x)_{\mathcal{I}} = s$ can be recovered from $b = Ax$ using* (BP) *if* (2.6) *holds for $y = M^\dagger s$.*

The pseudoinverse is also used by Tropp [144] in the definition of the exact recovery constant (ERC). In a slightly modified form, this constant is defined in terms of A and support $\mathcal{I}$ as

$$\kappa(A, \mathcal{I}) := \max_{i \notin \mathcal{I}} \|A_{\mathcal{I}}^\dagger A^{\downarrow i}\|_1. \tag{2.8}$$

The following theorem, due to Tropp [144], uses ERC to guarantee the uniform recovery of all vectors on the given support. We provide a simplified proof.

Theorem 2.3 (Tropp [144])**.** *A sufficient condition for* (BP) *to recover all x_0 supported on $\mathcal{I}$ with $|\mathcal{I}| \leq \operatorname{rank}(A)$ is that*

$$\kappa(A, \mathcal{I}) < 1. \tag{2.9}$$

Proof. Choose any $x \neq x_0$ satisfying $Ax = b$, and denote its support by $\mathcal{J}$. By the assumption on $\mathcal{I}$ there is at least on $j \in \mathcal{J}$ such that $j \notin \mathcal{I}$. Further noting that $A_{\mathcal{I}}^\dagger A_{\mathcal{I}} = I$, and $\|A_{\mathcal{I}}^\dagger A^{\downarrow i}\|_1 = 1$ for $i \in \mathcal{I}$ we have

$$\begin{aligned} \|x_0\|_1 = \|A_{\mathcal{I}}^\dagger A_{\mathcal{I}} x_0\|_1 = \|A_{\mathcal{I}}^\dagger b\|_1 = \|A_{\mathcal{I}}^\dagger A x\|_1 = \|\sum_{j \in \mathcal{J}} x_j (A_{\mathcal{I}}^\dagger A^{\downarrow j})\|_1 \\ \leq \sum_{j \in \mathcal{J}} |x_j| \cdot \|A_{\mathcal{I}}^\dagger A^{\downarrow j}\|_1 < \sum_{j \in \mathcal{J}} |x_j| = \|x\|_1. \end{aligned}$$

□

We connect the two theorems using the following result.

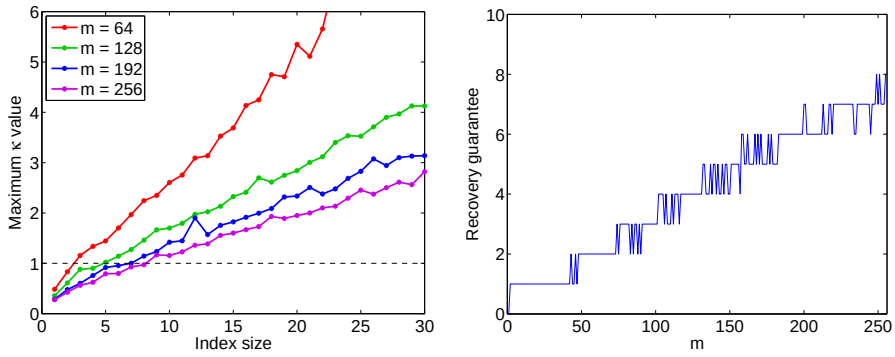
Theorem 2.4. *Theorem 2.3 implies Theorem 2.2 for all sign patterns s on $\mathcal{I}$ and vice versa.*

Proof. Let s be an arbitrary sign pattern and set $y = M^\dagger s$ with $M = A_{\mathcal{I}}^T$. We need to show that $|(A^{\downarrow j})^T y| < 1$ for all $j \notin \mathcal{I}$. Denoting $a_j = A^{\downarrow j}$ for convenience, we have

$$\begin{aligned} |(A^{\downarrow j})^T y| = |a_j^T y| = |a_j^T M^\dagger s| = |(A_{\mathcal{I}}^\dagger a_j)^T s| \\ \leq \sum_i |s_i| \cdot |(A_{\mathcal{I}}^\dagger a_j)_i| = \sum_i |(A_{\mathcal{I}}^\dagger a_j)_i| = \|A_{\mathcal{I}}^\dagger a_j\|_1 < 1. \end{aligned}$$

where we used assumption (2.9) and the fact that $(M^\dagger)^T = A_{\mathcal{I}}^\dagger$. For the reverse result note that, by assumption, (2.6) holds with $y = M^\dagger s$ for any sign pattern s. In particular choosing $s = \operatorname{sign}(A_{\mathcal{I}} a_j)$ gives

$$1 > |(A^T y)_j| = |a_j^T y| = |a_j^T M^\dagger s| = |s^T (A_{\mathcal{I}} a_j)| = \|A_{\mathcal{I}} a_j\|_1.$$



(a) (b)

▶ Figure 2.3: (a) Maximum value of $\kappa(A, \mathcal{I})$, sampled over 1,000 random index sets $\mathcal{I}$ for normalized randomly drawn Gaussian $m \times 256$ matrices A; (b) upper bound on uniform recovery guarantees for given matrices A.

Because $j \notin \mathcal{I}$ was arbitrary, this implies (2.9). □

Note that when A has unit-norm columns and the cardinality of $\mathcal{I}$ is one, condition (2.9) reduces to $\mu(A) < 1$. This in turn is equivalent to (2.1) for $r = 1$. Finally, as an example of the guarantees given based on the ERC, we create random $m \times 256$ matrices with unit norm columns for different values of m. For each matrix we generate 1,000 random index size for each cardinality up to 30 and compute the value of κ in (2.8). The maximum quantities are plotted in Figure 2.3(a) for a number of different values of m. Based on these values we determine the largest support size for which (2.9) holds. This gives an upper bound on the uniform recovery guarantees based on the ERC, which is shown in Figure 2.3(b). Like the mutual coherence, the ERC for a fixed support $\mathcal{I}$ can be practically computed.

2.4 Restricted isometry

The restricted isometry condition introduced by Candès and Tao [33] is one of the most widely used conditions for guaranteeing exact recovery, or establishing bounds on the recovery error. These sufficient conditions for recovery, as well as the error bounds, are expressed in terms of the r-restricted isometry constant $\delta_r(A)$, which they define as the smallest δ satisfying

$$(1 - \delta)\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1 + \delta)\|x\|_2^2 \tag{2.10}$$

for every r-sparse vector x. This condition equivalently requires that for all sets Λ of cardinality r, the squared singular values of A_Λ are bounded by $1 \pm \delta_r$. Another way to see this is that the eigenvalues of all $r \times r$ principal submatrices of the Gram matrix $A^T A$ are bounded between $(1 - \delta_r)$ and $(1 + \delta_r)$.

It is beyond the scope of this thesis to try and give an exhaustive summary of all results that have been expressed in terms of the restricted isometry constant. We therefore limit ourselves to illustrating the flavor of these results by citing two recent theorems by Candès.

Theorem 2.5 (Noiseless recovery, Candès [26]). *Let $b = Ax$ with any $x \in \mathbb{R}^n$, and denote by $x_{(r)}$ a vector whose nonzero entries correspond to the r entries of x largest in magnitude. Assume that $\delta_{2r} < \sqrt{2} - 1$. Then the solution x^* to* (BP) *obeys*

$$\|x^* - x\|_1 \leq c_1 \|x - x_{(r)}\|_1,$$

and

$$\|x^* - x\|_2 \leq c_1 r^{-1/2} \|x - x_{(r)}\|_1,$$

with constant c_1 depending only on δ_{2r}. Recovery is exact if x is r-sparse.

With slight modifications to the proof given in [26], the sufficient condition for exact uniform r-sparse recovery can be generalized to

$$\delta_{r+s}(A) < \frac{1}{1 + \sqrt{2r/s}}, \tag{2.11}$$

with $s \in [1, r]$. Choosing $s = r$ reduces to $\delta_{2r} < \sqrt{2} - 1$, as above. For the noisy case where $b = Ax + z$, the following result holds.

Theorem 2.6 (Noisy recovery, Candès [26]). *Assume that $\delta_{2r} < \sqrt{2} - 1$ and $\|Ax - b\|_2 \leq \epsilon$. Then the solution of* (BP_σ) *obeys*

$$\|x^* - x\|_2 \leq c_1 r^{-1/2} \|x - x_{(r)}\|_1 + c_2 \epsilon,$$

with constants c_1 and c_2 depending only on δ_{2r}.

In a recent paper, Cai et al. [24] sharpen these results and require $\delta_{1.75r}(A) < \sqrt{2} - 1$. Interestingly, they also show that earlier conditions by Candès and Tao [35] are actually implied by those in Theorem 2.5.

Restricted isometry constants for matrices. The restricted isometry constant of a matrix A, like its spark, is exceedingly hard to evaluate and essentially requires the computation of the extreme singular values of all A_Λ, with $|\Lambda| = r$. Given that we can may never be able to verify if condition (2.11) is satisfied, there are three ways to proceed: derive bounds on δ_r, design matrices with inherent restricted isometry properties, or consider families of random matrices. Recent work by d'Aspremont [48] uses semidefinite relaxation to obtain upper bounds on $(1 + \delta_r)$. By randomly sampling column supports Λ we can derive lower bounds, but these can only be used to show that (2.11) is not satisfied. The second approach is to design matrices in such a way that the restricted isometry constant is known by construction. Unfortunately though, the only known results in this direction, established by DeVore [53], satisfy the

restricted isometry conditions only for relatively small values of r. The most fruitful approach by far has been to consider families of random $m \times n$ matrices A and determine bounds on r such that, for a matrix randomly drawn from the family, conditions like those in Theorem 2.5 are satisfied with high probability. The first results in this direction were obtained by Candès and Tao [34]. They showed that an $m \times n$ matrix A consisting of m randomly selected rows of an $n \times n$ Fourier matrix, satisfies $\delta_{2r} + \delta_{3r} < 1$ with very high probability, provided that $r \leq C \cdot m/(\log n)^6$. These results were improved by Rudelson and Vershynin [129], who lowered the exponent from six to four. Another important class is the Gaussian ensemble where $m \times n$ matrices A are randomly generated by independently sampling its entries from the normal distribution with zero mean and variance $1/m$. Work by Candès and Tao [34], and Baraniuk et al. [4] shows that such matrices satisfy certain restricted isometry conditions with probability $1-\mathcal{O}(e^{-\gamma n})$, for some $\gamma > 0$, provided that $m \geq C \cdot r \log(n/r)$. Other examples include: Bernoulli ensembles (Baraniuk et al. [4]), randomly restricted Hadamard matrices (Rudelson and Vershynin [129]), and random Toeplitz matrices (Bajwa et al. [2]).

Experiments. For a more concrete illustration of the restricted isometry property, we generate a random 128×256 matrix A with normally distributed entries and scale its columns to have unit norm. We then sample the extreme singular values for 50,000 random submatrices A_Λ with $|\Lambda| = r$, for each $r = 1, \ldots, 128$, as shown in Figure 2.4(a). Using these singular values we can then determine a lower bound on δ_r by noting that

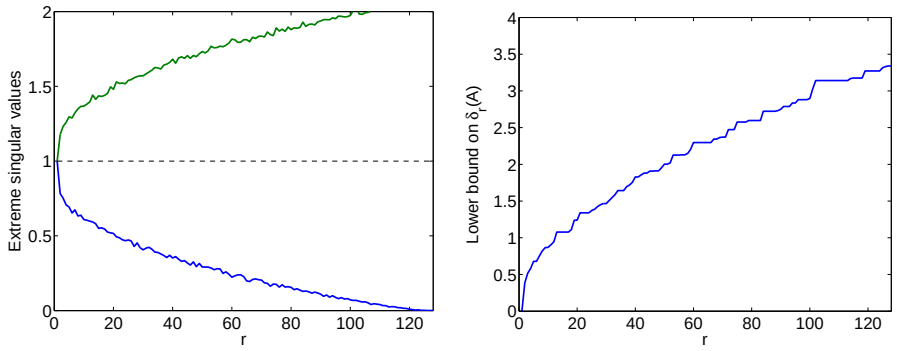
$$\delta_r(A) \geq \max\{[\sigma_1(A_\Lambda)]^2 - 1, 1 - [\sigma_r(A_\Lambda)]^2\}.$$

To ensure δ_r is non-decreasing in r, we enforce the fact that $\delta_r(A) \geq \delta_{r-1}(A)$, where needed, and obtain the lower bound on $\delta_r(A)$ shown in Figure 2.4(b). From these values it follows that condition (2.11) is satisfied only for $s = r = 1$, thus guaranteeing uniform recovery of at most one-sparse signals. This is much lower than the empirical result in which we considered the recovery of 5,000 random x_0 with different sparsity levels. In these experiments, as shown in Figure 2.5, all vectors with a sparsity of up to 32 were successfully recovered.

2.5 Geometry

We now look at the geometrical interpretation of basis pursuit recovery, which was first described by Donoho [57]. For basis pursuit, and to a lesser extent basis pursuit denoise, this interpretation provides an intuitive way of thinking about ℓ_1-based sparse recovery and has led to asymptotically sharp recovery results. We discuss this interpretation in detail here because of its relevance to the results we derive in Chapter 3.

Starting with the geometry of the ℓ_1-norm, note that the set of all points of the unit ℓ_1-ball, $\{x \in \mathbb{R}^n \mid \|x\|_1 \leq 1\}$, can be formed by taking convex



(a) extreme singular values (b) lower bound on $\delta_r(A)$

▶ Figure 2.4: Experiments with restricted isometry for a 128×256 random matrix A, showing (a) the extreme singular values attained by random submatrices of A with s columns, over 50,000 trials, and (b) the resulting lower bound on $\delta_s(A)$.

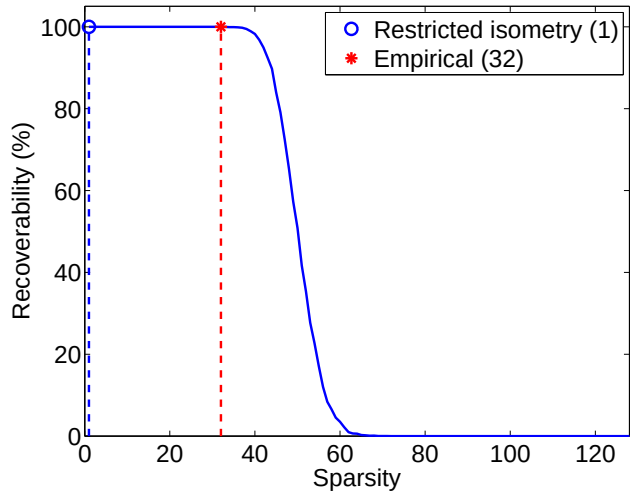
combinations of $\pm e_j$, the signed columns of the identity matrix. Geometrically this is equivalent to taking the convex hull of these vectors, giving the cross-polytope, or n-octahedron $\mathcal{C}_n = \text{conv}\{\pm e_1, \pm e_2, \ldots, \pm e_n\}$. As an example, we illustrate $\mathcal{C}_2$ and $\mathcal{C}_3$ in Figures 2.6(a,b) respectively; more information about these polytopes can be found in Grünbaum [89] and Ziegler [155].

For the satisfaction of the basis pursuit constraint $Ax = b$ we need to consider the image of x under A. By applying the linear mapping $x \mapsto Ax$ to all points $x \in \mathcal{C}$ we obtain another polytope $\mathcal{P} = \{Ax \mid x \in \mathcal{C}\} = A\mathcal{C}$. This is illustrated in Figure 2.6(c), from which it can be seen that $\mathcal{P}$ coincides with $\text{conv}\{\pm A^{\downarrow j}\}_j$, the convex hull of all points in $\mathbb{R}^m$ corresponding to the columns of A and their negation.

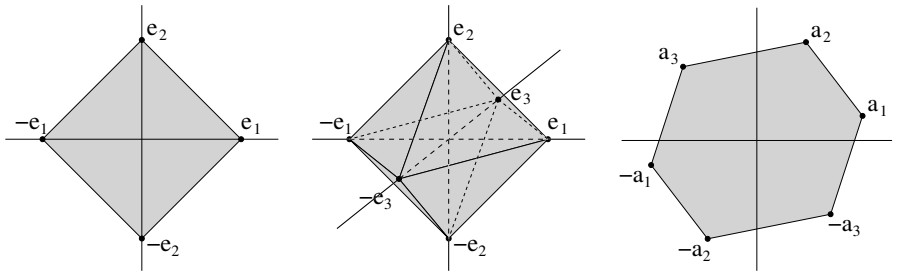
Now, recall that we want to minimize $\tau = \|x\|_1$ such that $Ax = b$ is satisfied. In other words, we want to find the smallest value of τ such that there exists an $x \in \tau\mathcal{C}$ satisfying $Ax = b$, that is $b \in A(\tau\mathcal{C}) = \tau\mathcal{P}$. When τ is too small, it is apparent from Figure 2.7(a) that $b \notin \tau\mathcal{P}$. On the other hand, when b is in the relative interior of $\tau\mathcal{P}$ we can reduce the size of the polytope and still satisfy $b = Ax$. We therefore conclude that basis pursuit finds the value of τ such that $\tau\mathcal{P}$ just meets b, as shown in Figure 2.7(b). The same principle applies to the basis pursuit denoise formulation (BP_σ). Defining the unit ℓ_2-norm ball as $\mathcal{B} = \{x \in \mathbb{R}^n \mid \|x\|_2 \leq 2\}$, this formulation looks for the smallest τ for which $\tau\mathcal{P} \cap (b + \sigma\mathcal{B}) \neq \emptyset$, as illustrated in Figure 2.7(c).

2.5.1 Facial structure and recovery

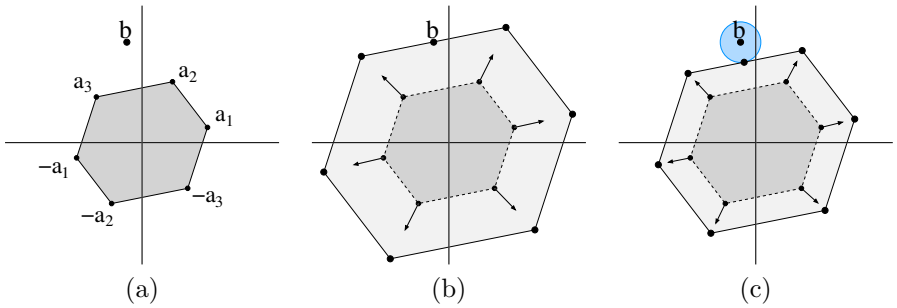
So far, we have only looked at the ℓ_1-norm of the basis pursuit (denoise) solution x^*. Fix τ, and let y denote the point where $b + \sigma\mathcal{B}$, with $\sigma = \|Ax - b\|$, meets $\mathcal{P}$. By definition, y must lie in the relative interior of exactly one face $\mathcal{F}$ of



▶ Figure 2.5: Percentage of r-sparse vectors x_0 recovered from $b = Ax_0$ for random 128×256 matrix A, based on 5,000 experiments for each r.



▶ Figure 2.6: Illustration of (a) cross-polytope $\mathcal{C}_2$, (b) cross-polytope $\mathcal{C}_3$, and (c) the image of $\mathcal{C}_3$ under A. For convenience we here denote $A^{\downarrow j}$ by a_j.



▶ Figure 2.7: (a) Initial situation with polytope $A\mathcal{C}$ and point b, and the geometry for (b) basis pursuit, and (c) basis pursuit denoise.

$\mathcal{P} := \tau A\mathcal{C}$. The location and sign of the non-zero entries of x^* are determined by the points $\pm\tau A^{\downarrow j}$ that lie in $\mathcal{F}$. In particular, $x_j^* > 0$ only if $\tau A^{\downarrow j} \in \mathcal{F}$, and $x_j^* < 0$ only if $-\tau A^{\downarrow j} \in \mathcal{F}$. By the definition of the ℓ_1-norm we have that the sum of the coefficient magnitudes is given by τ. The relative magnitudes correspond to the convex weights on each of the $\pm\tau A^{\downarrow j} \in \mathcal{F}$ to give y. As an example, consider the situation in Figure 2.7(b). Here, $y = b$ lies in the relative interior of the face given by the convex hull of $\tau A^{\downarrow 2}$ and $\tau A^{\downarrow 3}$, thus giving $x_2^*, x_3^* > 0$, and $x_2^* + x_3^* = \tau$. Based on this view it is immediate that x^* is unique if and only if $\mathcal{F}$ is a non-degenerate face of $\mathcal{P}$—that is, whenever the number of points $\pm\tau A^{\downarrow j}$ in $\mathcal{F}$ exceeds the dimension of the affine hull of $\mathcal{F}$ by at least one. When this is not the case we can write y as infinitely many different convex combinations of the vertices of $\mathcal{F}$, thus implying non-uniqueness of x^*.

Going back to the cross-polytope $\mathcal{C}$, we note that each face $\mathcal{F} \in \mathcal{C}$ can be expressed as the convex hull of a subset of vertices $\{\pm e_i\}_{i\in\mathcal{I}}$ not including any pair of vertices that are reflections with respect to the origin. Each face in $\mathcal{C}$ thus corresponds to a support and sign pattern of x with $\|x\|_1 = 1$, and $x_i \neq 0$ only if $i \in \mathcal{I}$, with the sign corresponding to the sign of $\pm e_i$. Now, note that A is a linear mapping from $\mathbb{R}^n$ to $\mathbb{R}^m$. It is well known (see for example Ziegler [155, Lemma 7.10]) that the image of a polytope $\mathcal{Q}$ in $\mathbb{R}^n$ under the linear map A, $\mathcal{R} = A\mathcal{Q}$, is also a polytope, and that the preimage of each face of $\mathcal{R}$ is a face of $\mathcal{Q}$. Applying this result to $\mathcal{P} = A\mathcal{C}$ we see that each face in $\mathcal{P}$ is the image of some face in $\mathcal{C}$, and therefore corresponds to a certain support and sign pattern of coefficient vectors $x \in \mathbb{R}^n$. The reverse is not always true, and some of the faces of $\mathcal{C}$ may not map to a face of $\mathcal{P}$.

Whenever some x_0 is on a face $\mathcal{F}$ of $\mathcal{C}$ that does not map to a face on $\mathcal{P}$, it means that $b = Ax_0$ lies in the interior of $\mathcal{P}$. This in turn means that there is another x with smaller ℓ_1-norm (i.e., we can reduce the size of $\mathcal{P}$ while maintaining $b \in \mathcal{P}$) that also satisfies $Ax = b$. In other words, x_0 is not a solution of the basis-pursuit problem and can therefore not be recovered. We conclude that a vector x_0 with $\|x_0\|_1 = \tau$ can be uniquely recovered if and only if the smallest face $\mathcal{F}$ in $\tau\mathcal{C}$ that contains x_0 maps to a nondegenerate face in $\mathcal{P} = \tau A\mathcal{C}$. From this we also conclude, as before, that recovery of x_0 depends only on its support and sign pattern.

2.5.2 Connection to optimality conditions

It is interesting to point out the close connection between the geometrical interpretation of basis pursuit, and the optimality conditions in Section 2.3. Let x be a vector supported on $\mathcal{I}$, and assume, without loss of generality, that $\|x\|_1 = 1$, and $x_i > 0$ for all $i \in \mathcal{I}$. In order for x to be the unique solution of (BP), $\mathcal{S} = \text{conv}\{A^{\downarrow i}\}_{i\in\mathcal{I}}$ must be a nondegenerate face of $\mathcal{P} = A\mathcal{C}$. By definition, each face of $\mathcal{P}$ can be supported by a hyperplane in $\mathbb{R}^m$. For $\mathcal{S}$ to be a face, there

must exist a normal vector z such that for each $j \in \mathcal{I}$,

$$\begin{aligned}(A^{\downarrow i} - A^{\downarrow j})^T z = 0 \quad & \forall i \in \mathcal{I}, \\ (-A^{\downarrow i} - A^{\downarrow j})^T z < 0 \quad & \forall i \in \mathcal{I}, \\ (\pm A^{\downarrow i} - A^{\downarrow j})^T z < 0 \quad & \forall i \notin \mathcal{I}.\end{aligned}$$

By scaling z such that $(A^{\downarrow j})^T z = 1$, this reduces to

$$\begin{aligned}(A^T z)_i = (A^T z)_j = 1 \quad & \forall i \in \mathcal{I} \\ |(A^T z)_i| < 1 \quad & \forall i \notin \mathcal{I},\end{aligned}$$

for each $j \in \mathcal{I}$, which exactly gives the conditions in (2.5) and (2.6). In fact, all vectors z, or more generally all vectors in the normal cone of $\mathcal{P}$ at b, can be seen to be Lagrange multipliers for (BP). This also explains why the strict inequality in (2.6) is a necessary and sufficient condition for x to be the unique solution. It also shows that whenever there exists a y that satisfies (2.6), then there also exists a y that satisfies only (2.5). For z, this corresponds to the relative interior and the boundary of the normal cone, respectively. However, if there only exists a z that satisfies $(A^T z)_i = 1$ for $i \in \mathcal{I}$, and $|(A^T z)_i| \leq 1$ for $i \notin \mathcal{I}$, with at least one equality, then $\mathcal{S}$ is (part of) a degenerate face, and x is not the unique solution.

2.5.3 Recovery bounds

We now consider the recovery bounds for basis pursuit as derived from the geometrical perspective. For simplicity of exposition, we assume that the faces of $\mathcal{P}$ are all non-degenerate, or equivalently, that $\mathcal{P}$ has $2n$ vertices.

A first observation is that the probability of recovering a random vector with s nonzero entries whose signs are ± 1 with equal probability, is equal to the ratio of $(s-1)$-faces in $\mathcal{P}$ to the number of $(s-1)$-faces in $\mathcal{C}$. That is, letting $\mathcal{F}_d(\mathcal{P})$ denote the collection of all d-faces [89] in $\mathcal{P}$, the probability of recovering an arbitrary exactly s-sparse x_0 using basis pursuit is given by

$$P_{\ell_1}(A, s) = \frac{|\mathcal{F}_{s-1}(A\mathcal{C})|}{|\mathcal{F}_{s-1}(\mathcal{C})|} = \frac{|\mathcal{F}_{s-1}(A\mathcal{C})|}{2^s}.$$

This coincides with the probability that is estimated (as a percentage) by the curve in Figure 2.5. For the probability of recovering vectors with a given support $\mathcal{I}$ we can write

$$P_{\ell_1}(A, \mathcal{I}) = \frac{|\mathcal{F}_{\mathcal{I}}(A\mathcal{C})|}{|\mathcal{F}_{\mathcal{I}}(\mathcal{C})|} = \frac{|\mathcal{F}_{\mathcal{I}}(A\mathcal{C})|}{2^{|\mathcal{I}|}}, \tag{2.12}$$

where $F_{\mathcal{I}}(\mathcal{C})$ denotes the number of faces in $\mathcal{C}$ formed by the convex hulls of $\{\pm e_i\}_{i \in \mathcal{I}}$, and $\mathcal{F}_{\mathcal{I}}(A\mathcal{C}) = F_{|\mathcal{I}|}(\mathcal{P}) \cap AF_{\mathcal{I}}(\mathcal{C})$ is the number of faces on $A\mathcal{C}$ generated by $\{\pm A^{\downarrow j}\}_{j \in \mathcal{I}}$.

Uniform recovery. Donoho [57] showed that the uniform recovery of all r-sparse vectors using basis pursuit corresponds to the notion of r-neighborliness of $\mathcal{P}$. The neighborliness of a polytope is defined as the largest number r such that the convex hull of any r of its vertices corresponds to a face of that polytope [89]. In the case of centrally symmetric polytopes (i.e., $\mathcal{P} = -\mathcal{P}$) this gives a neighborliness of one because the convex hull of any two vertices that are reflected through the origin will not be a face of $\mathcal{P}$. Such pairs are therefore excluded in the definition of neighborliness for centrally symmetric polytopes.

Bounds on central r-neighborliness were known as early as 1968, when McMullen and Shephard [113] showed that $r \leq \lfloor (m+1)/3 \rfloor$ whenever $2 < m \leq n-2$. More recently, Donoho [59] studied central neighborliness for random $m \times n$ orthogonal projectors A with $m \sim \delta n$. He derives, amongst other things, a function $\rho_N(\delta) > 0$ such that $\mathcal{P} = A\mathcal{C}$ is centrally $\lfloor \rho d \rfloor$-neighborly with high probability for large d, whenever $\rho < \rho_N(\delta)$. In other work, Linial and Novik [101] show the existence of m-dimensional centrally symmetric r-neighborly polytopes with $2(n+m)$ vertices, with

$$r(m,n) = \Theta\left(\frac{m}{1+\log((m+n)/m)}\right),$$

and show that this bound on r is tight.

Nonnegative basis pursuit and neighborliness. At first sight, formulation (1.6) seems very similar to basis pursuit with only an additional constraint requiring nonnegativity of x. This seemingly small change has major implications on the bounds for uniform recovery, essentially because the cross-polytope $\mathcal{C}$ is replaced by a simplex, thus giving rise to non-centrally symmetric polytope images under projection. Donoho and Tanner [55, 54] study recovery using nonnegative basis pursuit and mention the family of cyclic polytopes. These m-dimensional polytopes exist with any number $n > m$ of vertices and are $\lfloor m/2 \rfloor$-neighborly (recall that this guarantees the uniform recovery with (1.6) of all nonnegative vectors x_0 with at most $\lfloor m/2 \rfloor$ nonzero entries). Moreover, simple explicit constructions for these polytopes are available [79].

Chapter 3

Joint-sparse recovery

In this chapter we consider sparse recovery in the case where, instead of a single measurement vector (SMV) $b = Ax$, we have multiple measurement vectors (MMV) $B = AX$ (see Section 1.4.2 for notation). In the MMV setting it is typically assumed that the $n \times k$ matrix X is jointly sparse, meaning that the nonzero entries in each column are located at identical positions. As mentioned in Section 1.4.2, the MMV problem is a special case of group-sparse SMV where nonzero entries in x are clustered in predefined non-overlapping groups of indices Λ_i. (Some authors use the term 'blocks' instead of 'groups'. In the text, we use whichever term is used in the original context.) We can write MMV as a group-sparse problem by vectorizing B and X, setting $\Lambda_i = \{i, i+n, \dots, i+(k-1)n\}$ for $i = 1, \dots, n$, and redefining $A := I \otimes A$. This transformation to a group-sparse SMV is characterized by the special block-diagonal structure of A with identical blocks.

There is considerable freedom in choosing a convex formulation for the MMV problem. In this chapter we will concentrate on the $\ell_{1,2}$ relaxation (1.10) given in Section 1.4.2, and the ReMBo algorithm proposed by Mishali and Eldar [114]. Before doing so, we briefly survey some results on ℓ_0-based minimization and conditions for group-sparse recovery and their connection to the MMV problem. For related results on greedy approaches for the MMV problem, we refer to Chen and Huo [38], Cotter et al. [42], Eldar and Rauhut [74], Gribonval et al. [87], Leviatan and Temlyakov [100], and Tropp et al. [147, 148].

Throughout this chapter we make the following assumptions. The matrix $A \in \mathbb{R}^{m \times n}$ is full-rank. The unknown matrix to be recovered $X_0 \in \mathbb{R}^{n \times k}$, is r-row-sparse, and, except for our study of ReMBo, the columns of X_0 have identical support with exactly r nonzero entries.

3.1 Uniqueness of sparsest MMV solution

We define the sparsest solution to the MMV problem as the matrix X with the least number of rows containing nonzero entries that satisfies $AX = B$. This can be formulated as the non-convex optimization problem

$$\underset{X,v}{\text{minimize}} \quad \|v\|_0 \quad \text{subject to} \quad AX = B, \quad \|X^{i\rightarrow}\| \le v_i, \tag{3.1}$$

for any norm $\|\cdot\|$, or even any other nonnegative function $f(x)$ that is 0 if and only if $x = 0$. Denoting for convenience $\|X\|_{0,*} := \|[f(X^{i\rightarrow})]_i\|_0$, the following result gives conditions under which X is the unique solution of (3.1).

Theorem 3.1 (Cotter et al. [42], Chen and Huo [38]). *If $B = AX$ and*

$$\|X\|_{0,*} < \frac{\operatorname{Spark}(A) + \operatorname{rank}(B) - 1}{2}$$

then X is the unique solution to (3.1).

This result generalizes (2.4) for the SMV problem, where $\operatorname{rank}(B) = 1$. Formulation (3.1), just like (1.3), is intractable and we therefore proceed by studying recovery using convex relaxations.

3.2 Recovery of block-sparse signals

As noted earlier, MMV problems can be reformulated as block-sparse SMV problems, which have recently been studied by Eldar et al. [72] and Eldar and Mishali [73], who provide recovery conditions in terms of generalizations of the mutual coherence and the restricted isometry property (RIP). In particular, Eldar et al. define the block-coherence of a matrix A with blocks Λ_i of size k as

$$\mu_\Lambda(A) = \max_{i \neq j} \frac{1}{k} \sigma_1(A_{\Lambda_i}^T A_{\Lambda_j}),$$

where $\sigma_1(M)$ denotes the largest singular value of M. The coherence within blocks is described by the sub-coherence

$$\nu_\Lambda(A) = \max_l \max_{i,j \in \Lambda, i \neq j} |(A^{\downarrow i})^T A^{\downarrow j}|.$$

Using these two quantities they derive the following result.

Theorem 3.2 (Eldar et al. [72]). *Let x be an r-block-sparse vector with index sets Λ_i of cardinality k, and let A have unit norm columns. Then x is the unique solution of* (1.7) *with $b = Ax$, if*

$$kr < \frac{1}{2}\left(\frac{1}{\mu_\Lambda(A)} + k - (k-1)\frac{\nu(A)}{\mu_\Lambda(A)}\right). \tag{3.2}$$

With $A = I \otimes \bar{A}$ (as in MMV), it is easy to see that $\mu_\Lambda = \mu\bar{A}/k$ and $\nu(A) = 0$. Substituting these into (3.2) then gives $r < (1 + 1/\mu(A))/2$, which is exactly (2.1). In other words, based on these mutual-coherence conditions, the recovery guarantees for block-sparse signals using (1.7) are identical to those for independent recovery of each $X^{\downarrow j}$ from $B^{\downarrow j}$ using (BP). As an extension of the restricted isometry, Eldar and Mishali [73] define the block-RIP $\delta_k(A, \Lambda)$ as the smallest δ satisfying

$$(1 - \delta)\|c\|_2^2 \leq \|Ac\|_2^2 \leq (1 + \delta)\|c\|_2^2 \tag{3.3}$$

for all c supported on $\cup_{i \in \mathcal{I}} \Lambda_i$, with $|\mathcal{I}| \leq k$. They then show that any r-block-sparse vector x can be uniquely recovered using (1.7) if $\delta_{2r}(A, \Lambda) < \sqrt{2} - 1$. Note

that $\delta_{2r}(A, \Lambda) \leq \delta_{2kr}(A)$ because the vectors c considered for $\delta_{2r}(A, \Lambda)$ are a subset of those considered for $\delta_{2kr}(A)$. Indeed, Eldar and Mishali show that the block-restricted isometry condition (3.3) has a larger probability of being satisfied for random Gaussian ensembles than the standard non-block version. However, as with the block-coherence, this result does not give any additional guarantees for MMV since $\delta_{2r}(A, \Lambda) = \delta_{2r}(\bar{A})$. In both cases, the reason for this is the special block-diagonal structure of A that follows from the reformulation of the MMV problem as an SMV problem.

3.3 MMV recovery using row-norm sums

Our analysis of sparse recovery for the MMV problem of recovering X_0 from $B = AX_0$ begins with an extension of Theorem 2.1 to recovery using the convex relaxation

$$\underset{X}{\text{minimize}} \quad \sum_{j=1}^{n} \|X^{i\rightarrow}\| \quad \text{subject to} \quad AX = B; \tag{3.4}$$

note that the norm within the summation is arbitrary. Define the row support of a matrix as

$$\underset{\text{row}}{\text{Supp}}(X) = \{i \mid \|X^{i\rightarrow}\| \neq 0\}.$$

With these definitions we have the following result. (A related result is given by Stojnic et al. [135].)

Theorem 3.3. *Let A be an $m \times n$ matrix, $\mathcal{I} \subseteq \{1, \ldots, n\}$ be a fixed index set, and let $\|\cdot\|$ denote any vector norm. Then all $X_0 \in \mathbb{R}^{n\times k}$ with $\text{Supp}_{\text{row}}(X_0) \subseteq \mathcal{I}$ can be uniquely recovered from $B = AX_0$ using (3.4) if and only if for all $Z \neq 0$ with columns $Z^{\downarrow j} \in \text{Ker}(A)$,*

$$\sum_{i\in\mathcal{I}} \|Z^{i\rightarrow}\| < \sum_{i\notin\mathcal{I}} \|Z^{i\rightarrow}\|. \tag{3.5}$$

Proof. For the "only if" part, suppose that there is a $Z \neq 0$ with columns $Z^{\downarrow j} \in \text{Ker}(A)$ such that (3.5) does not hold. Now, choose $X^{i\rightarrow} = Z^{i\rightarrow}$ for all $i \in \mathcal{I}$ and with all remaining rows zero. Set $B = AX$. Next, define $V = X - Z$, and note that $AV = AX - AZ = AX = B$. The construction of V implies that $\sum_i \|X^{i\rightarrow}\| \geq \sum_i \|V^{i\rightarrow}\|$, and consequently X cannot be the unique solution of (3.4).

Conversely, let X be an arbitrary matrix with $\text{Supp}_{\text{row}}(X) \subseteq \mathcal{I}$, and let $B = AX$. To show that X is the unique solution of (3.4) it suffices to show that for any Z with columns $Z^{\downarrow j} \in \text{Ker}(A) \setminus \{0\}$,

$$\sum_i \|(X + Z)^{i\rightarrow}\| > \sum_i \|X^{i\rightarrow}\|.$$

This is equivalent to

$$\sum_{i \notin \mathcal{I}} \|Z^{i\rightarrow}\| + \sum_{i \in \mathcal{I}} \|(X+Z)^{i\rightarrow}\| - \sum_{i \in \mathcal{I}} \|X^{i\rightarrow}\| > 0.$$

Applying the reverse triangle inequality, $\|a+b\| - \|b\| \geq -\|a\|$, to the summation over $i \in \mathcal{I}$ and reordering exactly gives condition (3.5). □

In the special case of the sum of ℓ_1-norms, i.e., the $\ell_{1,1}$ norm defined in (1.9), summing the norms of the columns is equivalent to summing the norms of the rows. As a result, (3.4) can be written as

$$\underset{X}{\text{minimize}} \quad \sum_{j=1}^{k} \|X^{\downarrow j}\|_1 \quad \text{subject to} \quad AX^{\downarrow j} = B^{\downarrow j}, \quad i = 1, \ldots, k. \tag{3.6}$$

Because this objective is separable, the problem can be decoupled and solved as a series of independent basis pursuit problems, giving one $X^{\downarrow j}$ for each column $B^{\downarrow j}$ of B. The following result relates recovery using the sum-of-norms formulation (3.4) to recovery using $\ell_{1,1}$-minimization (see (1.11)).

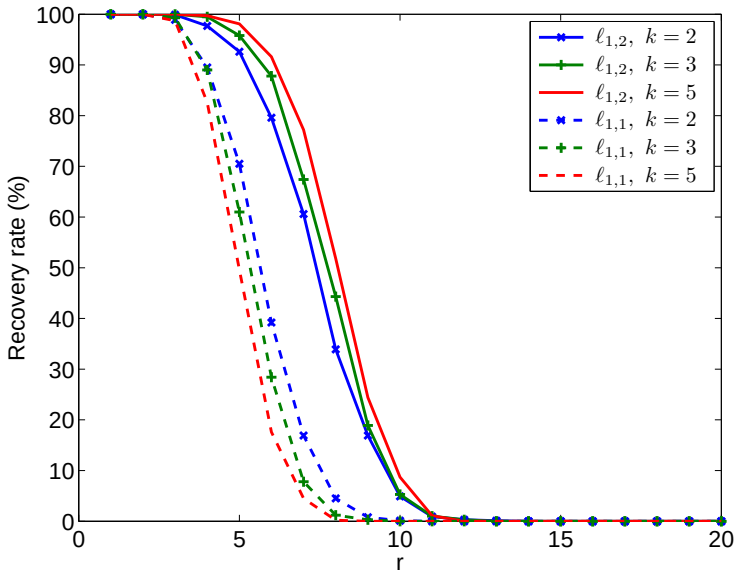
Theorem 3.4. *Let A be an $m \times n$ matrix, $\mathcal{I} \subseteq \{1, \ldots, n\}$ be a fixed index set, and $\|\cdot\|$ denote any vector norm. Then uniform recovery of all $X \in \mathbb{R}^{n \times k}$ with $\mathrm{Supp}_{\mathrm{row}}(X) \subseteq \mathcal{I}$ using sums of norms (3.4) implies uniform recovery on $\mathcal{I}$ using $\ell_{1,1}$.*

Proof. For uniform recovery on support $\mathcal{I}$ to hold it follows from Theorem 3.3 that for any matrix Z with columns $Z^{\downarrow j} \in \mathrm{Ker}(A) \setminus \{0\}$, property (3.5) holds. In particular it holds for Z with $Z^{\downarrow j} = \bar{z}$ for all j, with $\bar{z} \in \mathrm{Ker}(A) \setminus \{0\}$. Note that for these matrices there exist a norm-dependent constant γ such that

$$|\bar{z}_i| = \gamma \|Z^{i\rightarrow}\|.$$

Since the choice of $\bar{z}$ was arbitrary, it follows from (3.5) that the NS-condition (2.2) for independent recovery of vectors $B^{\downarrow j}$ using ℓ_1 in Theorem 2.1 is satisfied. Moreover, because $\ell_{1,1}$ is equivalent to independent recovery, we also have uniform recovery on $\mathcal{I}$ using $\ell_{1,1}$. □

An implication of Theorem 3.4 is that the use of restricted isometry conditions (or any technique for that matter) to analyze uniform joint r-sparse recovery conditions for the sum-of-norms approach necessarily lead to results that are no stronger than uniform ℓ_1 recovery for each vector independently. Eldar and Rauhut [74, Prop. 4.1] make a similar observation with regard to recovery using the $\ell_{1,2}$ norm. Their result can easily be extended to the general sum-of-norms formulation.



▶ Figure 3.1: Recovery rates for fixed, randomly drawn 20×60 matrices A, averaged over 1,000 trials at each row-sparsity level r. The nonzero entries in the $60 \times k$ matrix X_0 are sampled i.i.d. from the normal distribution. The solid and dashed lines represent $\ell_{1,2}$ and $\ell_{1,1}$ recovery, respectively.

3.4 MMV recovery using $\ell_{1,2}$

In this section we take a closer look at the $\ell_{1,2}$-minimization problem (1.10), which is a special case of the sum-of-norms problem. Although Theorem 3.4 establishes that uniform recovery via $\ell_{1,2}$ is no better than uniform recovery via $\ell_{1,1}$, there are many situations in which it recovers signals that $\ell_{1,1}$ cannot. Indeed, it is evident from Figure 3.1 that the probability of recovering individual signals with random signs and support is much higher for $\ell_{1,2}$. It is also clear from this figure that the probability of recovery via $\ell_{1,1}$ even reduces with increasing k. (We explain this phenomenon in Section 3.5.) The limitation of analysis for uniform recovery was also observed by Eldar and Rauhut [74], and they proceed by replacing of this worst-case analysis by an appropriate average-case analysis. Based on this model, they show that joint-sparse MMV recovery using $\ell_{1,2}$ is, on average, superior to independent basis pursuit SMV recovery.

We next derive the optimality conditions for $\ell_{1,2}$ similar to those for ℓ_1 given in Section 2.3. Building on these conditions we then construct examples for which $\ell_{1,2}$ recovery works and $\ell_{1,1}$ fails, and vice versa. We conclude the section with a set of experiments that illustrate the aforementioned construction and show the difference in recovery rates.

3.4.1 Sufficient conditions for recovery via $\ell_{1,2}$

The optimality conditions of the $\ell_{1,2}$ problem (1.10) play a vital role in deriving a set of sufficient conditions for joint-sparse recovery. In this section we derive the dual of (1.10) and the corresponding necessary and sufficient optimality conditions. These allow us to derive sufficient conditions for recovery via $\ell_{1,2}$.

We start with the Lagrangian for (1.10), which is defined as

$$\mathcal{L}(X, Y) = \|X\|_{1,2} - \langle Y, AX - B\rangle, \tag{3.7}$$

with $\langle V, W\rangle := \operatorname{trace}(V^T W)$ an inner-product defined over real matrices. The dual is then given by maximizing

$$\begin{aligned}\inf_X \mathcal{L}(X, Y) &= \inf_X \{\|X\|_{1,2} - \langle Y, AX - B\rangle\} \\ &= \langle B, Y\rangle - \sup_X \left\{\langle A^T Y, X\rangle - \|X\|_{1,2}\right\}\end{aligned} \tag{3.8}$$

over Y. (Because the primal problem has only linear constraints, there necessarily exists a dual solution Y^* that maximizes this expression [127, Theorem 28.2].) To simplify the supremum term, we note that for any convex, positively homogeneous function f,

$$\sup_v \{\langle w, v\rangle - f(v)\} = \begin{cases} 0 & \text{if } w \in \partial f(0), \\ \infty & \text{otherwise.}\end{cases}$$

To derive these conditions, note that positive homogeneity of f implies that $f(0) = 0$, and thus $w \in \partial f(0)$ implies that $\langle w, v\rangle \leq f(v)$ for all v. Hence, the supremum is achieved with $v = 0$. If on the other hand $w \notin \partial f(0)$, then there exists some v such that $\langle w, v\rangle > f(v)$, and by the positive homogeneity of f, $\langle w, \alpha v\rangle - f(\alpha v) \to \infty$ as $\alpha \to \infty$. Applying this expression for the supremum to (3.8), we arrive at the necessary condition

$$A^T Y \in \partial \|0\|_{1,2}, \tag{3.9}$$

which is required for dual feasibility.

We now derive an expression for the subdifferential $\partial\|X\|_{1,2}$. For rows i where $\|X^{i\rightarrow}\|_2 > 0$, the gradient is given by $\nabla\|X^{i\rightarrow}\|_2 = X^{i\rightarrow}/\|X^{i\rightarrow}\|_2$. For the remaining rows, the gradient is not defined, but $\partial\|X^{i\rightarrow}\|_2$ coincides with the set of unit ℓ_2-norm vectors $\mathcal{B}^k_{\ell_2} = \{v \in \mathbb{R}^k \mid \|v\|_2 \leq 1\}$. Thus, for each $i = 1, \ldots, n$,

$$\partial_{X^{i\rightarrow}} \|X\|_{1,2} \in \begin{cases} X^{i\rightarrow}/\|X^{i\rightarrow}\|_2 & \text{if } \|X^{i\rightarrow}\|_2 > 0, \\ \mathcal{B}^k_{\ell_2} & \text{otherwise.}\end{cases} \tag{3.10}$$

Combining this expression with (3.9), we arrive at the dual of (1.10):

$$\underset{Y}{\text{maximize}} \quad \operatorname{trace}(B^T Y) \quad \text{subject to} \quad \|A^T Y\|_{\infty,2} \leq 1. \tag{3.11}$$

Note that this dual formulation could have been obtained directly by noting that the $\ell_{\infty,2}$-norm is the dual of the $\ell_{1,2}$-norm. However, the above derivation does give additional information. Indeed, it shows that the following conditions are necessary and sufficient for a primal-dual pair (X^*, Y^*) to be optimal for (1.10) and its dual (3.11):

$$AX^* = B \qquad \text{(primal feasibility);} \tag{3.12a}$$

$$\|A^T Y^*\|_{\infty,2} \leq 1 \qquad \text{(dual feasibility);} \tag{3.12b}$$

$$\|X^*\|_{1,2} = \operatorname{trace}(B^T Y^*) \qquad \text{(zero duality gap).} \tag{3.12c}$$

The existence of a matrix Y^* that satisfies (3.12) provides a certificate that the feasible matrix X^* is an optimal solution of (1.10). However, it does not guarantee that X^* is also the unique solution. The following theorem gives sufficient conditions, similar to those in Section 2.3, that also guarantee uniqueness of the solution.

Theorem 3.5. *Let A be an $m \times n$ matrix, and B be an $m \times k$ matrix. Then a set of sufficient conditions for X to be the unique minimizer of (1.10) with Lagrange multiplier $Y \in \mathbb{R}^{m \times k}$ and row support $\mathcal{I} = \operatorname{Supp}_{\text{row}}(X)$, is that*

$$AX = B, \tag{3.13a}$$

$$(A^T Y)^{\downarrow i} = (X^*)^{i\rightarrow} / \|(X^*)^{i\rightarrow}\|_2, \qquad i \in \mathcal{I} \tag{3.13b}$$

$$\|(A^T Y)^{\downarrow i}\|_2 < 1, \qquad i \notin \mathcal{I} \tag{3.13c}$$

$$\operatorname{rank}(A_{\mathcal{I}}) = |\mathcal{I}|. \tag{3.13d}$$

Proof. The first three conditions clearly imply that (X, Y) primal and dual feasible, and thus satisfy (3.12a) and (3.12b). Conditions (3.13b) and (3.13c) together imply that

$$\operatorname{trace}(B^T Y) \equiv \sum_{i=1}^{n} [(A^T Y)^{\downarrow i}]^T X^{i\rightarrow} = \sum_{i=1}^{n} X^{i\rightarrow} \equiv \|X\|_{1,2}.$$

The first and last identities above follow directly from the definitions of the matrix trace and of the norm $\|\cdot\|_{1,2}$, respectively; the middle equality follows from the standard Cauchy inequality. Thus, the zero-gap requirement (3.12c) is satisfied. The conditions (3.13a)–(3.13c) are therefore sufficient for (X, Y) to be an optimal primal-dual solution of (1.10). Because Y determines the support and is a Lagrange multiplier for every solution X, this support must be unique. It then follows from condition (3.13d) that X must be unique. □

3.4.2 Counter examples

Using the sufficient and necessary conditions developed in the previous section we now construct examples of problems for which $\ell_{1,2}$ succeeds while $\ell_{1,1}$ fails, and vice versa. Because of its simplicity, we begin with the latter.

Recovery using $\ell_{1,1}$ where $\ell_{1,2}$ fails. Consider the matrices

$$A = \begin{bmatrix} 1 & 0.5 & 1 \\ 0 & 0.5 & 0.8 \end{bmatrix}, \quad \text{and} \quad X_0 = \begin{bmatrix} 2 & 1 \\ 2 & 10 \\ 0 & 0 \end{bmatrix}.$$

By drawing $A\mathcal{C}$, the convex hull of the columns of $\pm A$, it is easily seen that convex combinations of the first two columns give points on a face of the polytope. Because the weights in the columns of X_0 are scalar multiples of such points they can be uniquely recovered using ℓ_1 minimization, and consequently X_0 itself can be recovery using $\ell_{1,1}$.

On the other hand, for $\ell_{1,2}$ minimization to recover X_0, there must exist a $Y \in \mathbb{R}^{2\times 2}$ satisfying both (3.12b) and (3.13b). However, the unique Y satisfying the latter condition does not satisfy the former, thereby showing that $\ell_{1,2}$ fails to recover X_0.

Recovery using $\ell_{1,2}$ where $\ell_{1,1}$ fails. For the construction of a problem where $\ell_{1,2}$ succeeds and $\ell_{1,1}$ fails, we consider two vectors, f and s, with the same support $\mathcal{I}$, in such a way that individual ℓ_1 recovery fails for f, while it succeeds for s. In addition we assume that there exists a vector y that satisfies

$$y^T A^{\downarrow j} = \operatorname{sign}(s_j) \quad \text{for all } j \in \mathcal{I}, \qquad \text{and} \qquad |y^T A^{\downarrow j}| < 1 \quad \text{for all } j \notin \mathcal{I};$$

i.e., y satisfies conditions (3.13b) and (3.13c). Using the vectors f and s, we construct the 2-column matrix $X_0 = [(1-\gamma)s,\ \gamma f]$, and claim that for sufficiently small $\gamma > 0$, this gives the desired reconstruction problem. Clearly, for any $\gamma \neq 0$, $\ell_{1,1}$ recovery fails because the second column can never be recovered, and we only need to show that $\ell_{1,2}$ does succeed.

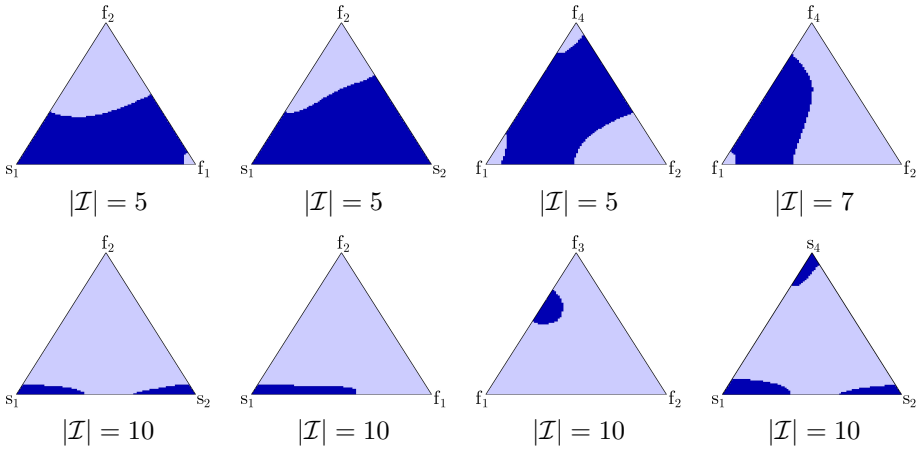
For $\gamma = 0$, the matrix $Y = [y, 0]$ satisfies conditions (3.13b) and (3.13c) and, assuming (3.13d) is also satisfied, X_0 is the unique solution of $\ell_{1,2}$ with $B = AX_0$. For sufficiently small $\gamma > 0$, the conditions that Y need to satisfy change slightly due to the division by $\|X_0^{i\rightarrow}\|_2$ for those rows in $\operatorname{Supp}_{\text{row}}(X)$. By adding corrections to the columns of Y those new conditions can be satisfied. In particular, these corrections can be done by adding weighted combinations of the columns in $\bar{Y}$, which are constructed in such a way that it satisfies $A_{\mathcal{I}}^T \bar{Y} = I$, and minimizes $\|A_{\mathcal{I}^c}^T \bar{Y}\|_{\infty,\infty}$ on the complement $\mathcal{I}^c$ of $\mathcal{I}$.

Note that the above argument can also be used to show that $\ell_{1,2}$ fails for γ sufficiently close to one. Because the support and signs of X remain the same for all $0 < \gamma < 1$, we can conclude the following: recovery using $\ell_{1,2}$ can be influenced by the magnitude of the nonzero entries of X_0. This is unlike $\ell_{1,1}$, where recovery depends only on the support and sign pattern of the nonzero entries. A consequence of this conclusion is that the notion of faces used in the geometrical interpretation of ℓ_1 is not applicable to the $\ell_{1,2}$ problem.

3.4.3 Experiments

To get an idea of just how much more $\ell_{1,2}$ can recover in the above case where $\ell_{1,1}$ fails, we generate a 20×60 matrix A with entries i.i.d. normally distributed,

and determine a set of vectors s_i and f_i with identical support for which ℓ_1 recovery succeeds and fails, respectively. Using triples of vectors s_i and f_j we construct row-sparse matrices such as $X_0 = [s_1, f_1, f_2]$ or $X_0 = [s_1, s_2, f_2]$, and attempt to recover from $B = AX_0W$, where $W = \mathrm{diag}(\omega_1, \omega_2, \omega_3)$ is a diagonal weighting matrix with nonnegative entries and unit trace, by solving (1.10). For problems of this size, interior-point methods are very efficient and we use SDPT3 [142, 150] through the CVX toolbox [84, 83]. We consider X_0 to be recovered when the maximum absolute difference between X_0 and the $\ell_{1,2}$ solution X^* is less than 10^{-5}. The results of the experiment are shown in Figure 3.2. In addition to the expected regions of recovery around individual columns s_i and failure around f_i, we see that certain combinations of vectors s_i still fail, while other combinations of vectors f_i may be recoverable. By contrast, when using $\ell_{1,1}$ to solve the problem, any combination of s_i vectors can be recovered while no combination including an f_i can be recovered.



▶ Figure 3.2: Generation of problems where $\ell_{1,2}$ succeeds, while $\ell_{1,1}$ fails. For a 20×60 matrix A and fixed support of size $|\mathcal{I}| = 5, 7, 10$, we create vectors f_i that cannot be recovered using ℓ_1, and vectors s_i than can be recovered. Each triangle represents an X_0 constructed from the vectors denoted in the corners. The location in the triangle determines the weight on each vector, ranging from zero to one, and summing up to one. The dark areas indicates the weights for which $\ell_{1,2}$ successfully recovered X_0.

3.5 Bridging the gap from $\ell_{1,1}$ to ReMBo

We begin this section with a discussion showing that the performance of $\ell_{1,1}$ can only get worse with increasing number of observations, thus explaining empirical observations made earlier by Chen and Huo [38] and Mishali and Eldar [114]. We then show how the recovery rate can be improved by using the boosting technique introduced by Mishali and Eldar [114]. The resulting

boosted-ℓ_1 approach is a simplified version of the ReMBo-ℓ_1 algorithm, which we discuss in Section 3.6. Although boosted-ℓ_1 has a lower performance, we include it because its simplicity makes it easy to analyze and allows us to show more intuitively what it is that makes ReMBo-ℓ_1 work so well. Also, the recovery rate of boosted-ℓ_1 motives a performance model for ReMBo-ℓ_1 recovery. Thus, while boosted-ℓ_1 may not be a viable algorithm in practice, it does help bridge the gap between $\ell_{1,1}$ and ReMBo-ℓ_1.

As described in Section 3.3, recovery using $\ell_{1,1}$ is equivalent to individual ℓ_1 recovery of each column $x^{(j)} := X_0^{\downarrow j}$ based on solving (BP) with $b := B^{\downarrow j}$, for $j = 1, \ldots, k$. Assume that the signs of nonzero entries in the support of each $x^{(j)}$ are uniformly distributed. Then we can express the probability of recovering X_0 with row support $\mathcal{I}$ using $\ell_{1,1}$ in terms of the probability of recovering individual vectors on that support using ℓ_1. By the separability of (3.6), $\ell_{1,1}$ recovers X_0 if and only if (BP) successfully recovers each $x^{(j)}$. Recalling (2.12), denote the recovery rate of an $x^{(j)}$ supported on $\mathcal{I}$ by $P_{\ell_1}(A, \mathcal{I})$. Then the expected $\ell_{1,1}$ recovery rate is

$$P_{\ell_{1,1}}(A, \mathcal{I}, k) = \left[P_{\ell_1}(A, \mathcal{I})\right]^k.$$

This expression shows that the probability of recovery using $\ell_{1,1}$ can only decrease as k increases, which clearly defeats the purpose of gathering multiple observations; see Figure 3.1.

There are many problem instances where $\ell_{1,1}$ fails to recover X_0 as a whole but does correctly recover a subset of columns $x^{(j)}$. The following boosting procedure [114] exploits this fact and uses it to help generate the entire solution. Given such a vector $x^{(j)}$ with support $\mathcal{J}$ of sufficiently small cardinality (e.g., less than $m/2$), solve the following system for $\bar{X}$:

$$\text{minimize}\, \bar{X} \quad \|A_{\mathcal{J}}\bar{X} - B\|_F. \tag{3.14}$$

If the residual in (3.14) is zero, conclude that the support $\mathcal{J}$ coincides with $\mathcal{I}$ and assume that the nonzero entries of X_0 are given by $\bar{X}$. If the residual is nonzero, the support $\mathcal{J}$ is necessarily incorrect, and the next sufficiently-sparse vector is checked. This approach is outlined in Algorithm 3.1.

The recovery properties of the boosted ℓ_1 approach are opposite from those of $\ell_{1,1}$: it fails only if all individual columns with support $\mathcal{I}$ fail to be recovered using ℓ_1. Hence, given an unknown $n \times k$ matrix X_0 supported on $\mathcal{I}$ with its sign pattern uniformly random, the boosted ℓ_1 algorithm enjoys an expected recovery rate of

$$P(A, \mathcal{I}, k) = 1 - \left[1 - P_{\ell_1}(A, \mathcal{I})\right]^k. \tag{3.15}$$

Note that this result hinges on our blanket assumption that the columns of X_0 have identical support.

To experimentally verify the recovery rate, we generate a 20×80 matrix A with entries independently sampled from the normal distribution and fix a randomly chosen support set $\mathcal{I}_r$ for three levels of sparsity, $r = 8, 9, 10$. On each of these three supports we generate vectors with all possible sign patterns and solve (BP) to verify if they can be recovered (see Section 3.4.3). This gives

Algorithm 3.1: The boosted ℓ_1 algorithm

given A, B
for $j = 1, \ldots, k$ **do**
  solve (BP) with $b := B^{\downarrow j}$ to get x
  $\mathcal{J} \leftarrow \mathrm{Supp}(x)$
  if $|\mathcal{J}| < m/2$ **then**
    solve (3.14) to get $\bar{X}$
    if $A_{\mathcal{J}}\bar{X} = B$ **then**
      $X^* = 0$
      $[(X^*)^{i\rightarrow}]_{i\in\mathcal{J}} \leftarrow \bar{X}$
      return solution X^*
return failure

exactly the face counts required to compute the ℓ_1 recovery probability in (2.12), and the expected boosted ℓ_1 recovery rate in (3.15).

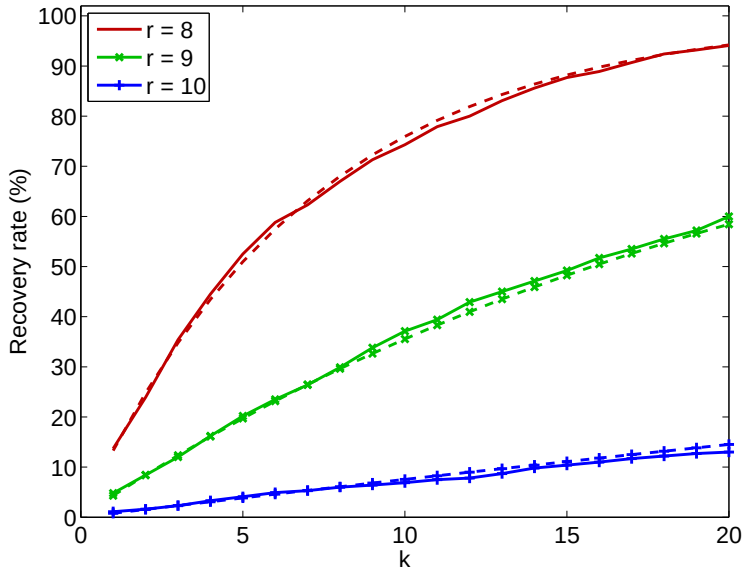
For the empirical success rate we take the average over 1,000 trials with random matrices X_0 supported on $\mathcal{I}_r$, and its nonzero entries independently drawn from the normal distribution. Because recovery of individual vectors using ℓ_1 minimization depends only on their sign pattern, we reduce the computational time by comparing the sign patterns against precomputed recovery tables (this is possible because both A and $\mathcal{I}_r$ remain fixed), rather than invoking an ℓ_1 solver for each vector. The theoretical and empirical recovery rates using boosted ℓ_1 are plotted in Figure 3.3.

3.6 Recovery using ReMBo

The ReMBo algorithm by Mishali and Eldar [114] proceeds by taking a random k-vector w and combining the individual observations in B into a single weighted observation $b := Bw$. It then solves an SMV problem $Ax = b$ for this b and checks if the computed solution x^* is sufficiently sparse. If not, the above steps are repeated with different weight vectors w until a given maximum number of trials is reached. If the support $\mathcal{J}$ of x^* is small, we form $A_{\mathcal{J}} = [A^{\downarrow j}]_{j\in\mathcal{J}}$, and check if (3.14) has a solution $\bar{X}$ with zero residual. If this is the case we have the nonzero rows of the solution X^* in $\bar{X}$ and are done. Otherwise, we simply proceed with the next weight vector w.

The ReMBo algorithm does not prescribe a particular SMV solver. However, throughout this section we use ReMBo in conjunction with ℓ_1 minimization, thus giving the ReMBo-ℓ_1 algorithm summarized in Figure 3.2. It can be seen that the ReMBo-ℓ_1 algorithm reduces to boosted ℓ_1 when setting the maximum number of iterations to k and choosing $w := e_i$ in the ith iteration.

The formulation given in [114] requires a user-defined threshold on the cardinality of the support $\mathcal{J}$ instead of the fixed threshold $m/2$ in Algorithm 3.2. Ideally, based on (2.4), this threshold should be half the spark of A, but this



▶ Figure 3.3: Theoretical (dashed) and experimental (solid) performance of boosted ℓ_1 on three problem instances with different row-support sizes r.

is too expensive to compute. If, however, we assume that A is in general position, we can take $\text{Spark}(A) = m + 1$. Choosing an even larger threshold can help recover signals with row sparsity exceeding $m/2$, although, in this case, the solution can no longer be guaranteed to be the sparsest possible solution.

In our study of ReMBo-ℓ_1, we fix an unknown matrix X_0 with row support $\mathcal{I}$ of cardinality r. We deviate from the blanket assumption made in the introduction of this chapter and allow for individual columns to be supported on a subset of $\mathcal{I}$. Each time we multiply B by a random weight vector $w^{(k)}$, we in fact create a new problem which, with probability one, has an exact r-sparse solution $x_0 := X_0 w^{(k)}$.

Recall that the recovery of sparse vectors from $b = Ax_0$ using ℓ_1 depends only on the support and sign-pattern of x_0. Clearly, the probability of recovery improves as the number of distinct sign patterns encountered by ReMBo-ℓ_1 increases. The maximum number of sign patterns encountered with boosted ℓ_1 is the number of observations k. The question thus becomes, how many different sign patterns ReMBo-ℓ_1 can encounter by taking linear combinations of the columns in X_0? (We disregard the situation where elimination occurs and $|\text{Supp}(X_0 w)| < r$.) Equivalently, we can ask how many orthants in $\mathbb{R}^s$ (each corresponding to a different sign pattern) can be properly intersected by the subspace given by the range of the submatrix $\bar{X}$ consisting of the nonzero rows of X_0 (with proper we mean intersection of the interior). In Section 3.6.1 we derive an exact expression for the maximum number of proper orthant intersections in $\mathbb{R}^n$ by a subspace generated by d vectors, denoted by $C(n, d)$.

Algorithm 3.2: The ReMBo-ℓ_1 algorithm

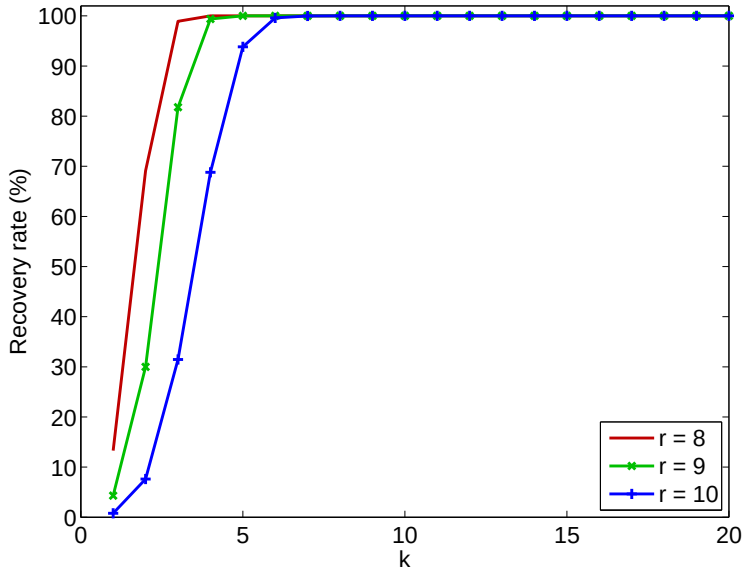
given A, B. Set Iteration $\leftarrow 0$
while Iteration $<$ MaxIteration **do**
  $w \leftarrow \text{Random}(k, 1)$
  solve (BP) with $b = Bw$ to get x
  $\mathcal{J} \leftarrow \text{Supp}(x)$
  if $|\mathcal{J}| < m/2$ **then**
    solve (3.14) to get $\bar{X}$
    if $A_{\mathcal{J}}\bar{X} = B$ **then**
      $X^* = 0$
      $[(X^*)^{i\rightarrow} \leftarrow X^{i\rightarrow}$ for $i \in \mathcal{J}$
      return solution X^*
  Iteration $\leftarrow$ Iteration $+ 1$
return failure

Based on the above reasoning, a good model for a bound on the recovery rate for $n \times k$ matrices X_0 with $\text{Supp}_{\text{row}}(X_0) = \mathcal{I} < m/2$ using ReMBo-ℓ_1 is given by

$$P_R(A, \mathcal{I}, k) = 1 - \prod_{i=1}^{S} \left[1 - \frac{\mathcal{F}_{\mathcal{I}}(A\mathcal{C})}{\mathcal{F}_{\mathcal{I}}(\mathcal{C}) - 2(i-1)} \right]. \tag{3.16}$$

where S denotes the number of unique sign patterns tried. The maximum possible value of S is $C(|\mathcal{I}|, k)/2$. This maximum number of orthant intersections is attained with probability one for subspaces $\text{Range}(X_0)$, where the entries in the support of X_0 are normally distributed (cf. Corollary 3.8). The term within brackets denotes the probability of failure and the fraction represents the success rate, which is given by the ratio of the number of faces $\mathcal{F}_{\mathcal{I}}(A\mathcal{C})$ that survived the mapping to the total number of faces to consider. The total number reduces by two at each trial because we can exclude the face f we just tried, as well as $-f$. The factor of two in $C(|\mathcal{I}|, k)/2$ is also due to this symmetry. (Henceforth we use the convention that the uniqueness of a sign pattern is invariant under negation.)

This model would be a bound for the average performance of ReMBo-ℓ_1 if the sign patterns generated would be randomly sampled from the space of all sign patterns on the given support. However, because they are generated from the orthant intersections with a subspace, the actual set of patterns is highly structured. Indeed, it is possible to imagine a situation where the $(r-1)$-faces in $\mathcal{C}$ that perish in the mapping to $A\mathcal{C}$ all have sign patterns that are contained in the set generated by a single subspace. Any other set of sign patterns would then necessarily include some faces that survive the mapping and by trying all patterns in that set we would recover X_0. In this case, the average recovery over all X_0 on that support could be much higher than that given by (3.16). An interesting questions is how the surviving faces of $\mathcal{C}$ are distributed. Due to the simplicial structure of the facets of $\mathcal{C}$, we can expect the faces that perish to be



▶ Figure 3.4: Theoretical performance model for ReMBo on three problem instances with different sparsity levels r.

partially clustered and partially unclustered. (If a $(d-2)$-face perishes, then so will the two $(d-1)$-faces whose intersection gives this face, thus giving a cluster of faces that are lost. On the other hand, there will be faces that perish, while all their sub-faces survive.) Note that, regardless of these patterns, recovery is guaranteed in the limit whenever the number of sign patterns tried exceeds, $|\mathcal{F}_{\mathcal{I}}(\mathcal{C})| - |\mathcal{F}_{\mathcal{I}}(\mathcal{AC})|$, the number of faces lost.

Figure 3.4 illustrates the theoretical performance model based on $C(n, d)$, for which we derive the exact expression in Section 3.6.1. In Section 3.6.2 we discuss practical limitations, and in Section 3.6.3 we empirically look at how the number of sign patterns generated grows with the number of normally distributed vectors w, and how this affects the recovery rates. To allow comparison between ReMBo and boosted ℓ_1, we used the same matrix A and support $\mathcal{I}_r$ used to generate Figure 3.3.

3.6.1 Maximum orthant intersections with subspace

In this section we give an exact characterization of the number of orthants intersected by a subspace. The maximum number of intersections is given by the following theorem.

Theorem 3.6. *Let $C(n,d)$ denote the maximum attainable number of orthant interiors intersected by a subspace in $\mathbb{R}^n$ generated by d vectors. Then $C(n,1) = 2$, $C(n,d) = 2^n$ for $d \geq n$. In general, $C(n,d)$ is given by*

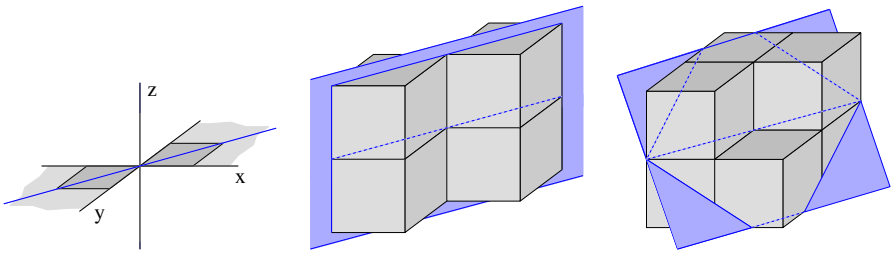
$$C(n,d) = C(n-1,d-1) + C(n-1,d) = 2\sum_{i=0}^{d-1} \binom{n-1}{i}. \tag{3.17}$$

Proof. The number of intersected orthants is exactly equal to the number of proper sign patterns (excluding zero values) that can be generated by linear combinations of those d vectors. When $d = 1$, there can only be two such sign patterns corresponding to positive and negative multiples of that vector, thus giving $C(n,1) = 2$. Whenever $d \geq n$, we can choose a basis for $\mathbb{R}^n$ and add additional vectors as needed, and we can reach all points, and therefore all $2^n = C(n,d)$ sign patterns.

For the general case (3.17), let $v_1, \ldots, v_d$ be vectors in $\mathbb{R}^n$ such that the affine hull with the origin, $S = \operatorname{aff}\{0, v_1, \ldots, v_d\}$, gives a subspace in $\mathbb{R}^n$ that properly intersects the maximum number of orthants, $C(n,d)$. Without loss of generality, assume that vectors v_i, $i = 1, \ldots, d-1$, all have their nth component equal to zero. Now, let $T = \operatorname{aff}\{0, v_1, \ldots, v_{d-1}\} \subseteq \mathbb{R}^{n-1}$ be the intersection of S with the $(n-1)$-dimensional subspace of all points $\mathcal{X} = \{x \in \mathbb{R}^n \mid x_n = 0\}$, and let C_T denote the number of $(n-1)$-orthants intersected by T. Note that T itself, as embedded in $\mathbb{R}^n$, does not properly intersect any orthant. However, by adding or subtracting an arbitrarily small amount of v_d, we intersect $2C_T$ orthants; taking v_d to be the nth column of the identity matrix would suffice for that matter; see Figure 3.5(a,b). Any other orthants that are added have either $x_n > 0$ or $x_n < 0$, and their number does not depend on the magnitude of the nth entry of v_d, provided it remains nonzero. Because only the first $n-1$ entries of v_d determine the maximum number of additional orthants, the problem reduces to $\mathbb{R}^{n-1}$. In fact, we ask how many new orthants can be added to C_T taking the affine hull of T with v, the orthogonal projection v_d onto $\mathcal{X}$. Since the maximum orthants for this d-dimensional subspace in $\mathbb{R}^{n-1}$ is given by $C(n-1,d)$, this number is clearly bounded by $C(n-1,d) - C_T$. Adding this to $2C_T$, we have

$$\begin{aligned} C(n,d) &\leq 2C_T + [C(n-1,d) - C_T] = C_T + C(n-1,d) \\ &\leq C(n-1,d-1) + C(n-1,d) \\ &\leq 2\sum_{i=0}^{d-1} \binom{n-1}{i}. \end{aligned} \tag{3.18}$$

The final expression follows by expanding the recurrence relations, which generates (a part of) Pascal's triangle, and combining this with $C(1,j) = 2$ for $j \geq 1$. In the above, whenever there are free orthants in $\mathbb{R}^{n-1}$, that is, when $d < n$, we can always choose the corresponding part of v_d in that orthant. As a consequence, no subspace generated by a set of vectors can intersect the maximum number of orthants when the range of those vectors includes some e_i.



(a) (b) (c)

▶ Figure 3.5: Illustration of orthant intersection, (a) one-dimensional subspace in the two-dimensional xy-plane, intersecting two orthants, (b) trivial extension to three-dimensions, doubling the number of orthants intersected, (c) optimal number of six intersections in three dimensions.

We now show that this expression holds with equality. Let U denote an $(n-d)$-subspace in $\mathbb{R}^n$ that intersects the maximum $C(n, n-d)$ orthants. We claim that in the interior of each orthant not intersected by U there exists a vector that is orthogonal to U. If this were not the case then T must be aligned with some e_i and can therefore not be optimal. The span of these orthogonal vectors generates a d-subspace V that intersects $C_V = 2^n - C(n, n-d)$ orthants, and it follows that

$$\begin{aligned} C(n,d) &\geq C_V = 2^n - C(n, n-d) \\ &\geq 2^n - 2\sum_{i=0}^{n-d-1}\binom{n-1}{i} = 2\sum_{i=0}^{n-1}\binom{n-1}{i} - 2\sum_{i=0}^{n-d-1}\binom{n-1}{i} \\ &= 2\sum_{n-d}^{n-1}\binom{n-1}{i} = 2\sum_{i=0}^{d-1}\binom{n-1}{i} \geq C(n,d), \end{aligned}$$

where the last inequality follows from (3.18). Consequently, all inequalities hold with equality. □

Corollary 3.7. *Given $d \leq n$, then $C(n,d) = 2^n - C(n, n-d)$, and $C(2d,d) = 2^{2d-1}$.*

The exact characterization of the number of orthant intersections follows directly from the proof of Theorem 3.6 and is summarized by the following corollary.

Corollary 3.8. *A subspace $\mathcal{H}$ in $\mathbb{R}^n$, defined as the range of $V = [v_1, v_2, \ldots, v_d]$, intersects the maximum number of orthants $C(n,d)$ whenever $\operatorname{rank}(V) = n$, or when $e_i \notin \operatorname{Range}(V)$ for $i = 1, \ldots, n$.*

3.6.2 Practical considerations

In practice, it is generally not computationally feasible to generate all of the $C(|\mathcal{I}|, k)/2$ unique sign patterns. This means that we would have to set S in (3.16) to the number of unique patterns actually tried. For a given X_0 the actual probability of recovery is determined by a number of factors. First of all, the linear combinations of the columns of the nonzero part of $\bar{X}$ prescribe a subspace and therefore a set of possible sign patterns. With each sign pattern is associated a face in $\mathcal{C}$ that may or may not map to a face in $A\mathcal{C}$. Second, depending on the probability distribution from which the weight vectors w are drawn, there is a certain probability for reaching each sign pattern. Summing the probability of reaching those patterns that can be recovered gives the probability $P(A, \mathcal{I}, X_0)$ of recovering with an individual random sample w. The probability of recovery after t trials is then of the form

$$1 - [1 - P(A, \mathcal{I}, X_0)]^t.$$

To attain a certain sign pattern $\bar{e}$, we need to draw a k-vector w such that $\operatorname{sign}(\bar{X}w) = \bar{e}$. For a positive sign on the ith position of the support, we can take any vector w in the open halfspace $\{w \mid \bar{X}^{i\rightarrow}w > 0\}$, and likewise for negative signs. The region of vectors w in $\mathbb{R}^k$ that generates a desired sign pattern thus corresponds to the intersection of $|\mathcal{I}|$ open halfspaces. The measure of this intersection as a fraction of $\mathbb{R}^k$ determines the probability of sampling such a w. To formalize, define $\mathcal{K}$ as the cone generated by the rows of $-\operatorname{diag}(\bar{e})\bar{X}$, and the unit Euclidean $(k-1)$-sphere $\mathcal{S}^{k-1} = \{x \in \mathbb{R}^k \mid \|x\|_2 = 1\}$. The intersection of halfspaces then corresponds to the interior of the polar cone of $\mathcal{K}$: $\mathcal{K}^\circ = \{x \in \mathbb{R}^k \mid x^T y \leq 0,\ \forall y \in \mathcal{K}\}$. The fraction of $\mathbb{R}^k$ taken up by $\mathcal{K}^\circ$ is given by the $(k-1)$-content of $\mathcal{S}^{k-1} \cap \mathcal{K}^\circ$ to the $(k-1)$-content of $\mathcal{S}^{k-1}$ [89]. This quantity coincides precisely with the definition of the external angle of $\mathcal{K}$ at the origin.

3.6.3 Experiments

In this section we illustrate the results from Section 3.6 and examine some practical considerations that affect the performance of ReMBo. For all experiments that require the matrix A, we use the same 20×80 matrix that was used in Section 3.5, and likewise for the supports $\mathcal{I}_r$. To solve (BP), we again use CVX in conjunction with SDPT3. We consider x_0 to be recovered from $b = Ax_0 = AX_0w$ if $\|x^* - x_0\|_\infty \leq 10^{-5}$, where x^* is the computed solution.

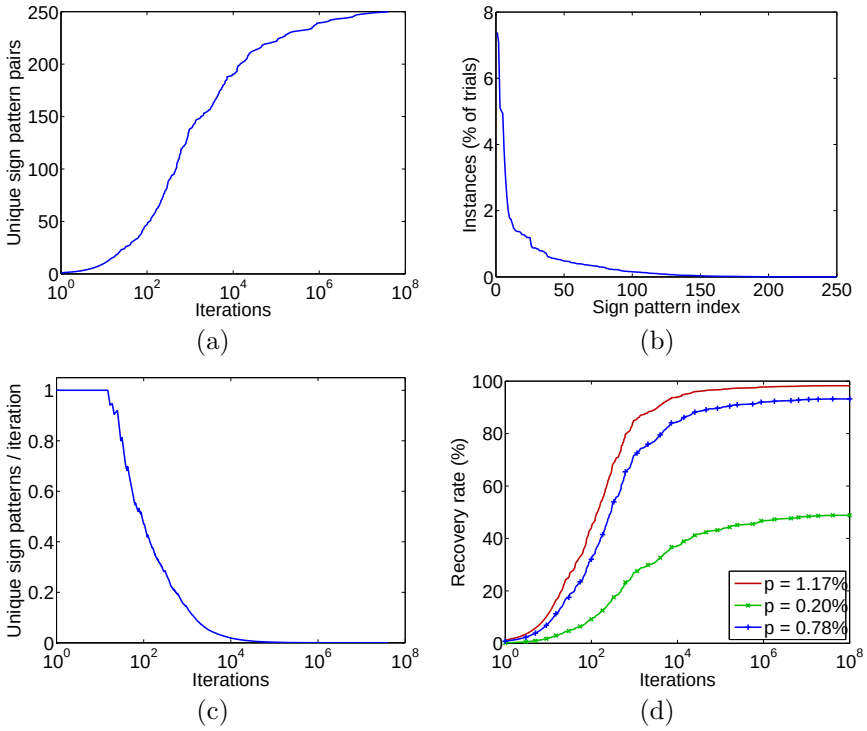
The experiments that are concerned with the number of unique sign patterns generated depend only on the $r \times k$ matrix $\bar{X}$ representing the nonzero entries of X_0. Because an initial reordering of the rows does not affect the number of patterns, those experiments depend only on $\bar{X}$, $r = |\mathcal{I}|$, and the number of observations k; the exact indices in the support set $\mathcal{I}$ are irrelevant for those tests.

Generation of unique sign patterns. The practical performance of ReMBo-ℓ_1 depends on its ability to generate as many different sign patterns as possible using the columns in X_0. A natural question to ask then is how the number of such patterns grows with the number of randomly drawn samples w. Although this ultimately depends on the distribution used for generating the entries in w, we shall, for sake of simplicity, consider only samples drawn from the normal distribution. As an experiment, we take a 10×5 matrix $\bar{X}$ with normally-distributed entries, and over 10^8 trials record how often each sign-pattern (or negation) was reached, and in which trial they were first encountered. The results of this experiment are summarized in Figure 3.6. From the distribution in Figure 3.6(b) it is clear that the occurrence levels of different orthants exhibits a strong bias. The most frequently visited orthant pairs were reached up to 7.3×10^6 times, while others, those hard to reach using weights from the normal distribution, were observed only four times over all trials.

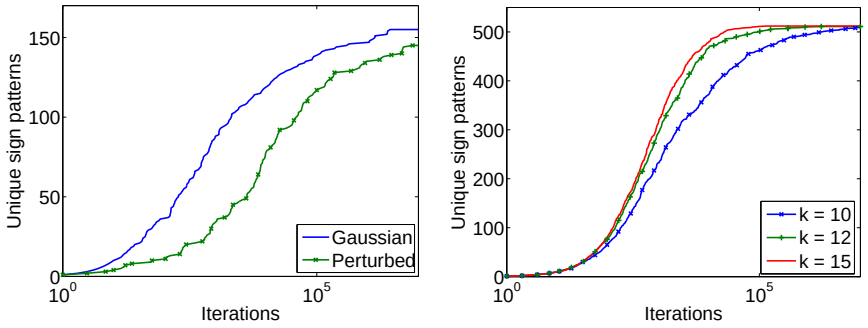
Likewise, we can look at the rate of encountering new sign patterns. This is done in Figure 3.6(c), which shows how the average rate changes over the number of trials. The curves in Figure 3.6(d) illustrate the probability given in (3.16), with S set to the number of orthant pairs at a given iteration, and with face counts determined as in Section 3.5, for three instances with support cardinality $r = 10$, and observations $k = 5$.

Influence of $\bar{X}$ on the performance of ReMBo. The number of orthants that a subspace can intersect does not depend on the basis with which it was generated. However, the basis does greatly influence the ability to sample those orthants. Figure 3.7 shows two ways in which this can happen. In part (a) we sample the number of unique sign patterns for two different 9×5 matrices $\bar{X}$, each with columns scaled to unit ℓ_2-norm. The entries of the first matrix are independently drawn from the normal distribution, while those in the second are generated by repeating a single column drawn likewise and adding small random perturbations to each entry. This difference causes the average angle between any pair of columns to decrease from 65 degrees in the random matrix to a mere 8 in the perturbed matrix, and greatly reduces the probability of reaching certain orthants. The same idea applies to the case where $d \geq n$, as shown in part (b) of the same figure. Although choosing d greater than n does not increase the number of orthants that can be reached, it does make reaching them easier, thus allowing ReMBo-ℓ_1 to work more efficiently. Hence, we can expect ReMBo-ℓ_1 to have higher recovery on average when the number of columns in X_0 increases and when they have a lower mutual coherence $\mu(X) = \max_{i \neq j} |x_i^T x_j| / (\|x_i\|_2 \cdot \|x_j\|_2)$.

Limiting the number of iterations. The number of iterations used in the previous experiments greatly exceeds that what is practically feasible: we cannot afford to run ReMBo-ℓ_1 until all possible sign patterns have been tried, even if there was to way detect that the limit had been reached. Realistically, we should set the number of iterations to a fixed maximum that depends on the computational resources available, as well as on the problem setting itself.

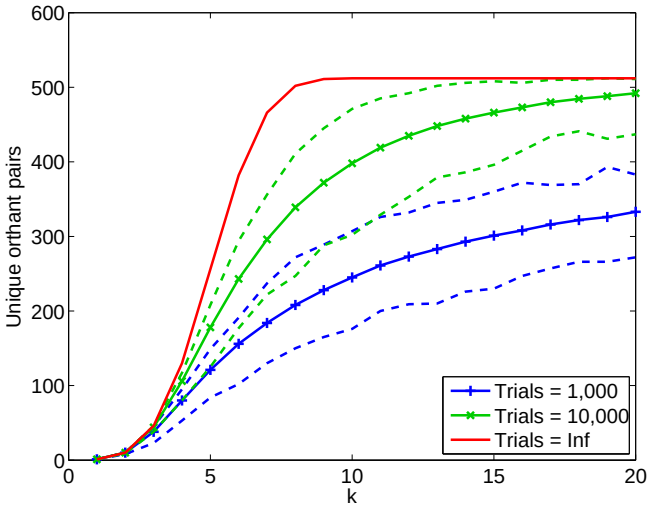


▶ Figure 3.6: Sampling the sign patterns for a 10×5 matrix $\bar{X}$, with (a) number of unique sign patterns versus number of trials, (b) relative frequency with which each orthant is sampled, (c) average number of new sign patterns per iteration as a function of iterations, and (d) modeled probability of recovery using ReMBo-ℓ_1 for three instances of X_0 with row sparsity $r = 10$, and $k = 5$ observations.

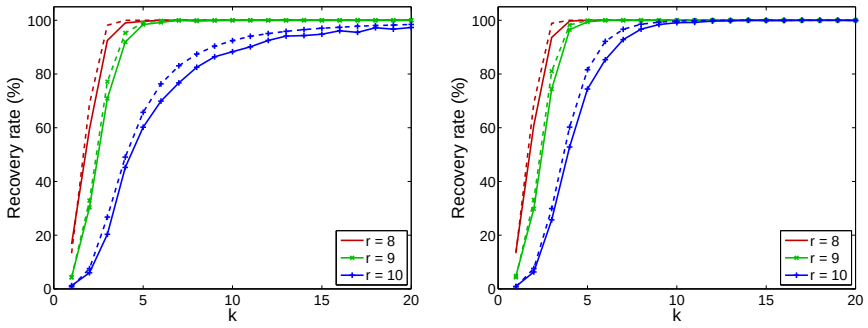


▶ Figure 3.7: Number of unique sign patterns for (a) two 9×5 matrices $\bar{X}$ with columns scaled to unit ℓ_2-norm; one with entries drawn independently from the normal distribution, and one with a single random column repeated and random perturbations added, and (b) $10 \times k$ matrices with $k = 10, 12, 15$.

Figure 3.6(a) shows the empirical number of unique orthants S as a function of iterations for a fixed $\bar{X}$. For the performance analysis it would be useful to know what the distribution of unique orthant counts looks like on the average $\bar{X}$ for a fixed number of trials. To find out, we draw 1,000 random $r \times k$ matrices $\bar{X}$ with $r = 10$ nonzero rows fixed and the number of columns ranging from $k = 1, \ldots, 20$. For each $\bar{X}$ we count the number of unique sign patterns attained after 1,000 and 10,000 iterations. The resulting minimum, maximum, and median values are plotted in Figure 3.8, along with the theoretical maximum. More interestingly, of course, is the average recovery rate of ReMBo-ℓ_1 based those number of iterations. For this test we again use the 20×80 matrix A with fixed support $\mathcal{I}$. For each value of $k = 1, \ldots, 20$, we generate random matrices X on $\mathcal{I}$ and run ReMBo-ℓ_1 with the maximum number of iterations set to 1,000 and 10,000, respectively. To save on computation time, we again compare the on-support sign pattern of each coefficient vector Xw to the precomputed results instead of solving ℓ_1. The average recovery rate thus obtained is plotted in Figures 3.9(a,b), along with the average of the modeled performance using (3.16) with S set to the orthant counts found in the previous experiment.



▶ Figure 3.8: Effect of limiting the number of weight vectors w on the distribution of unique orthant counts for $10 \times k$ random matrices $\bar{X}$, solid lines give the median number and the dashed lines indicate the minimum and maximum values, the top solid line is the theoretical maximum.



(a) 1,000 trials (b) 10,000 trials

▶ Figure 3.9: Effect of limiting the number of weight vectors w on the average performance of the ReMBo-ℓ_1 algorithm. The recovery with (a) 1,000, and (b) 10,000 trials is plotted (solid line), along with the average predicted performance (dashed line). These results are based on a fixed 20×80 matrix A and three different support sizes $r = 8, 9, 10$. The support patterns used are the same as those used for Figure 3.3.

Chapter 4

Solvers

Along with the establishment of the theoretical foundations of sparse recovery, there has been a natural interest in algorithms that can efficiently solve the underlying optimization problems. So far, we have predominantly considered convex relaxations of ideal—but impractical—formulations for sparse recovery. Besides these relaxations, there is a large class of greedy approaches that do not solve any fixed problem formulation, but instead prescribe an algorithm for obtaining sparse (and possibly approximate) solutions. Theoretical recovery results have also been developed for these greedy approaches, but because they do not solve an optimization problem, the results only apply to a particular algorithm. By contrast, the theoretical results for the convex formulations studied in Chapters 2 and 3 apply to any algorithm for solving the given problem, regardless of the implementation. The ReMBo-ℓ_1 results in Chapter 3 are algorithm-specific as a whole, but are nevertheless independent of the algorithm used to solve the basis-pursuit subproblems. More information on greedy algorithms can be found in [43] and [117].

A third class of algorithms uses mostly heuristic approaches to solve nonconvex formulations. Despite the fact that these algorithms are typically guaranteed only to converge to a local minimizer, they do seem to work quite well in practice. Figure 4.1 summarizes these different approaches and highlights our primary focus on solvers for convex formulations. In this classification we can further distinguish between solvers that fit the data exactly, and those that allows some misfit.

In the first three sections of this chapter we discuss solvers for basis pursuit, basis pursuit denoise, and regularized basis pursuit. We do not go into the details of nonconvex heuristics. Although we do not extensively cover solvers for other problems, such as nonnegative basis pursuit or MMV, we mention extensions of techniques to these problems whenever possible.

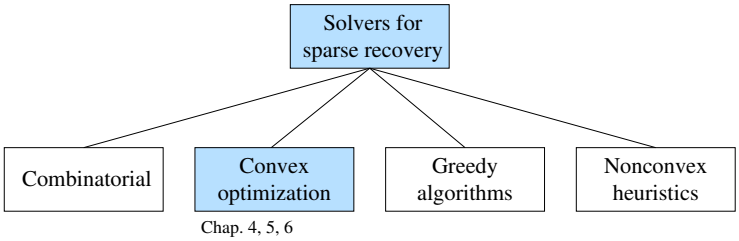
4.1 Basis pursuit (BP)

When Chen et al. [39] proposed basis pursuit, they emphasized that it can be conveniently reformulated as a standard linear program:

$$\underset{x}{\text{minimize}} \quad c^T x \quad \text{subject to} \quad Ax = b,\ x \geq 0. \tag{4.1}$$

Given the basis pursuit formulation

$$\underset{\bar{x}}{\text{minimize}} \quad \|\bar{x}\|_1 \quad \text{subject to} \quad \bar{A}\bar{x} = b, \tag{4.2}$$



▶ Figure 4.1: Classification of solvers for sparse recovery.

this can be done using the well-established technique of variable splitting, in which we write $\bar{x}$ as $x^+ - x^-$ with $x^+, x^- \geq 0$. This allows us to rewrite the ℓ_1-norm of $\bar{x}$ as the sum $e^T(x^+ + x^-)$ of entries in its positive and negative parts, with e denoting a vector of all ones. The term $\bar{A}\bar{x}$ is conveniently rewritten as $\bar{A}x^+ - \bar{A}x^-$. With $A = [\bar{A},\ -\bar{A}]$ and $x = [x^+;\ x^-]$ it follows that (4.2) can equivalently be expressed as (4.1). Once the latter formulation is solved, we can recovery $\bar{x}$ simply by subtracting the second half of x (corresponding to x^-) from the first. It is easily seen that at most one entry j of x_j^+ or x_j^- is nonzero, for otherwise we could lower the objective by subtracting the smallest of the two from both entries without changing Ax. In the special case of nonnegative basis pursuit (1.6), we can forego splitting, and immediately replace $\|x\|_1$ by $e^T x$.

4.1.1 Simplex method

The advantage of rewriting the basis pursuit formulation as a linear program is that solvers for the latter are at a very mature stage. Perhaps the most famous amongst these, and certainly the most established, is the simplex method [47]. Let the polyhedron $\mathcal{S} = \{x \geq 0 \mid Ax = b\}$ denote the feasible set of (4.1). Assuming $\mathcal{S}$ is nonempty, the simplex method systematically traverses neighboring vertices of $\mathcal{S}$ until either an optimal solution or a direction of unbounded descent has been found. Associated with each vertex is a support $\mathcal{B}$ of m indices such that $x_i \geq 0$ for $i \in \mathcal{B}$ and $x_i = 0$ for $i \in \mathcal{N}$, with $\mathcal{N} = \{1, \ldots, n\} \setminus \mathcal{B}$ the complement of $\mathcal{B}$. The index sets $\mathcal{B}$ of neighboring vertices are such that all but one entry is the same. The key part of each iterations of the simplex algorithm is to determine which entry from $\mathcal{B}$ to remove and which entry from $\mathcal{N}$ to add. This choice ideally lowers the objective value at the new vertex, but this is sometimes impossible and we may have to settle for one at which the objective remains the same. In this latter situation, which arises when (4.1) is degenerate, special care has to be taken not to traverse a cyclic path of vertices, causing the algorithm to loop forever.

The major cost of the simplex method consists of solving a linear system of equations $A_{\mathcal{B}} x_{\mathcal{B}} = b$ or $A_{\mathcal{B}}^T y = c_{\mathcal{B}}$. Efficient implementations of the simplex methods maintain a factorization of $A_{\mathcal{B}}$, which is updated after each iteration, and occasionally regenerated for numerical stability. This implies that, when

applied to linear programs where A is dense, the simplex method does not scale very well with the problem size. In addition, it cannot take advantage of fast operators for products with A.

4.1.2 Interior-point algorithms

Feasible interior-point methods for linear programming approach the optimal solution not by tracing a path on the boundary of $\mathcal{S}$, but instead by generating iterates on the interior of the cone $x \geq 0$.

We here briefly discuss the primal-dual interior-point method described by Nocedal and Wright [122]. Let y and s denote the Lagrange multipliers associated respectively with the constraints $Ax = b$ and $x \geq 0$. The KKT conditions for (4.1) are then given by

$$F(x, y, s) = \begin{bmatrix} A^T y + s - c \\ Ax - b \\ XSe \end{bmatrix} = 0, \qquad (x, s) \geq 0, \tag{4.3}$$

where $X = \text{diag}(x)$ and $S = \text{diag}(s)$. This system can be solved by applying Newton's method, requiring at each iteration the solution of the linear system

$$J(x, y, s) \cdot d = -F(x, y, s),$$

where $J(x, y, s)$ denotes the Jacobian of F evaluated at (x, y, s).

The interior-point method modifies the right-hand side of this system. In particular, at each iteration it requires that $x_i s_i \geq \tau$ for an iteration-specific value of τ that positive and gradually decreases towards zero, based on a duality measure and a centering parameter. This leads to the following system of equations that needs to be solved at each iteration, and which constitutes the main computational part of the algorithm:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -XSe + \tau e \end{bmatrix}. \tag{4.4}$$

Once the search direction is found a line search ensures that x and s remain strictly within the feasible region.

Needless to say, the above algorithm description is rather elementary and more advanced versions exist. An important modification is the infeasible approach, in which the first two criteria in (4.3) need not be satisfied at intermediate iterates. For a more in-depth discussion, see Nesterov and Nemirovskii [120], and the very accessible book by Wright [152].

All of the interior-point methods mentioned above require the solution of a system of equations to determine a search direction. Rather than solving the equivalent of (4.4) directly, they typically work with the so-called normal-equations which follow by eliminating Δs and Δx, giving $AD^2A^T\Delta y = v$ for $D = S^{-1/2}X^{1/2}$ and an appropriate vector v.

Solving this system becomes a real bottleneck with increasing problem size. When A is known implicitly through routines for fast multiplication it is necessary to solve either the normal equations, or the augmented system (4.4), using an iterative linear solver. However, these linear systems necessarily become arbitrarily ill-conditioned near the solution, and iterative solvers typically struggle to obtain the required accuracies.

4.1.3 Null-space approach

In compressed sensing applications, A is often constructed by choosing arbitrary rows from an orthonormal basis B. We can write this as $A = RB$, where R is a restriction operator, i.e., a matrix composed of a subset of rows from an identity matrix. It is easily seen that, due to orthonormality of the rows of B, an orthonormal basis for the null-space of A is given by $N = B^T(R^c)^T$, where R^c is composed of the columns of the identity matrix that do not appear in R.

Any feasible point $x \in \mathcal{F} = \{x \mid Ax = b\}$ can then be written as $x_b + Nc$, where x_b satisfies $Ax_b = b$ and c is any vector in $\mathbb{R}^{n-m}$. We can find x_b by applying the pseudo-inverse of A to b, giving $x_b = A^\dagger b = (A^TA)^{-1}A^Tb = A^Tb$, since $A^TA = I$. This allows us to reformulate (4.1) as

$$\underset{c}{\text{minimize}} \quad \|x_b + Nc\|_1.$$

While convex and unconstrained, this problem is still hard to solve because it is non-differentiable. We do not further elaborate on this approach.

4.2 Basis pursuit denoise (BP_σ)

The basis pursuit denoise formulation (BP_σ) minimizes the ℓ_1-norm of the coefficients subject to a bound on the ℓ_2-norm misfit $b - Ax$. This formulation is very natural for practical signal recovery where noise is ubiquitous, but often well understood. Despite this appealing property, very few efficient solvers for this problem exist. The remainder of this section reviews various approaches.

4.2.1 Second-order cone programming

Just like (BP) can be reformulated as a standard linear program, we can reformulate (BP_σ) as a standard second-order cone program

$$\begin{array}{ll} \underset{x}{\text{minimize}} & c^Tx \\ \text{subject to} & \|A_{(i)}x + b_{(i)}\|_2 \le c_{(i)}^Tx + d_{(i)}, \quad i = 1, \dots, k, \\ & Mx = y, \end{array} \tag{4.5}$$

The inequality constraint in (BP_σ) requires no conversion and easily fits in into the above formulation. The objective $\|x\|_1$ can be rewritten as the minimum

sum of u_i such that $-u_i \leq x_i \leq u_i$. This gives

$$\begin{array}{ll} \underset{x,u}{\text{minimize}} & e^T u \\ \text{subject to} & |x_i| \leq u_i, \quad i = 1, \ldots, n, \\ & \|Ax + b\|_2 \leq \sigma, \end{array}$$

which can be rewritten as (4.5) by combining x and u into a single vector and appropriately defining the matrices $A_{(i)}$ and the remaining vectors.

Second-order cone programs can be solved using primal-dual interior-point methods (see Nesterov and Nemirovskii [120] and Alizadeh and Goldfarb [1]). An alternative approach, used by Candès and Romberg in their pioneering sparse recovery toolbox ℓ_1-magic [31], is to use a primal log-barrier method. Although these methods are very robust and capable of generating highly accurate solutions, they both suffer the same problem as the interior-point method described in Section 4.1.2, namely, they scale extremely poorly with increasing problem size.

Second-order cone programs can also be used to solve other problems such as basis pursuit with complex variables, as well as the group-sparse (1.7) and $\ell_{1,2}$ minimization (1.10) problems. See Malioutov et al. [110], Eldar and Mishali [73], and Stojnic et al. [135] for the latter two formulations.

4.2.2 Homotopy

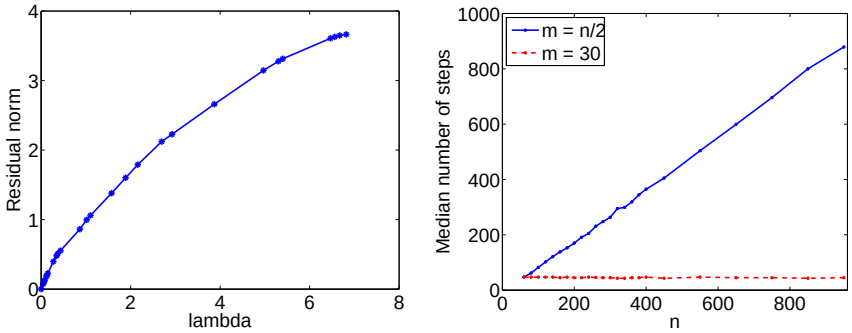
The homotopy method was introduced by Osborne et al. [123] as an approach for solving variable-selection problems. The method takes advantage of the fact that the entries of the optimal solution x^*_λ of (QP_λ) are piecewise linear in λ. This allows the algorithm to trace the objective

$$f_\lambda(x) = \tfrac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1, \tag{4.6}$$

as a function of λ, starting with $\lambda = \|A^T b\|_\infty$, and reducing it in discrete steps until no more progress can be made. With some minor modifications, the algorithm can find the solution to (BP_σ), provided, of course, that $\|Ax - b\| \leq \sigma$ can be satisfied.

As an illustration, we compute the homotopy trajectory for a random 20×40 matrix A and random b. The resulting curve, plotted in terms of the ℓ_2 residual norm against λ, is shown in Figure 4.2(a). The dots on the curve indicate the values of λ at which the active set changes.

While the homotopy algorithm can be extremely fast, it also has a number of problems. First, the number of steps required to reach the solution can become quite large, especially if the final solution is not very sparse. Second, the subproblems, which need to be solved at each iteration, can become expensive to solve whenever the number of active variables is large. This again happens when the basis pursuit denoise solution is not sparse. Finally, we cannot warm-start the algorithm and need to start from $x = 0$ even when the problem changes only slightly.



(a) homotopy trajectory (b) number of iterations

▶ Figure 4.2: (a) Complete homotopy trajectory of a random 20×40 problem, and (b) median number of steps required to solve 50 random $m \times n$ problems with $m = n/2$ (solid) and $m = 30$ (dashed), for different values of n.

From a geometrical perspective (see Section 2.5) it is intuitive that the number of steps required to solve basis pursuit problem by homotopy should predominantly depend on the dimensionality of the problem and the distribution of the vectors in A. To test the former idea, we apply the homotopy algorithm to two sets of random $m \times n$ matrices A for 50 random vectors b and determined the median number of steps taken until completion. For the first set we fixed $m = 30$ and varied n, while in the second we chose $m = n/2$. The results are plotted in Figure 4.2 and confirms (for such matrices) the observation by Malioutov et al. [111] that the number of steps grows proportional to m.

4.2.3 NESTA

The NESTA algorithm by Becker et al. [7] is based on a framework developed by Nesterov [118, 119] for minimization of convex nonsmooth functions $f(x)$ over a convex set $\mathcal{Q}_p$. We assume that $f(x)$ can be written as

$$f(x) = \max_{u \in \mathcal{Q}_d} \langle u, Wx \rangle, \tag{4.7}$$

where $W \in \mathbb{R}^{p \times n}$ and $\mathcal{Q}_d$ is a closed and bounded convex set. This definition covers all induced norms, and holds in particular for $f(x) = \|x\|_1$ by setting $W = I$ and $\mathcal{Q}_d = \{u \mid \|u\|_\infty \leq 1\}$. Because functions of the form (4.7) are convex but not generally smooth, Nesterov [118] proposed to augment $f(x)$ with a smoothing prox-function $p_d(u)$ that is continuous and strongly convex on $\mathcal{Q}_d$. For some smoothing parameter $\mu > 0$ this gives a continuously differentiable function

$$f_\mu(x) = \max_{u \in \mathcal{Q}_d} \langle u, Wx \rangle - \mu p_d(u). \tag{4.8}$$

This function can then be minimized over $x \in \mathcal{Q}_p$ to give an approximate solution to the original problem. The algorithm proposed by Nesterov solves

this type of problem by generating a sequence of iterates $x^{(i)}$ based on two other sequences $y^{(i)}$ and $z^{(i)}$ and a primal proxy function $p_p(x)$ (see Becker et al. [7] for more details).

By assuming that the rows in A are orthonormal, and choosing a convenient proxy function, Becker et al. find closed form solutions to the subproblems that determine the iterates $y^{(i)}$ and $z^{(i)}$. Consequently, they obtain an efficient algorithm for minimizing of $f_\mu(x)$ over the convex set $\mathcal{Q}_p$. Applied to the basis pursuit denoise formulation this gives

$$\underset{x}{\text{minimize}} \quad f_\mu(x) \quad \text{subject to} \quad \|Ax - b\|_2 \leq \sigma. \tag{4.9}$$

They also show that the accuracy of this approximate solution to (BP_σ) is proportional to the smoothing parameter μ. It thus follows that for an accurate approximation we have to solve (4.9) for a value of μ close to zero. Because doing so directly is inefficient, they propose a continuation method that gradually reduces μ to the desired level, and warm-starting the solver with the previous solution x of (4.9) as an initial point.

The major advantage of NESTA over other approaches is its versatility, making it applicable to many different problems that few other solvers can handle.

4.3 Regularized basis pursuit (QP_λ)

The regularized basis pursuit formulation (QP_λ), introduced by Chen et al. [39] under the name basis pursuit denoise, is the Lagrange formulation of (BP_σ). In this formulation, the value of λ, which is related to the Lagrange multiplier corresponding to the constraint $\|Ax - b\|_2 \leq \sigma$, is assumed to be known. This valuable information allows us to use the unconstrained formulation (QP_λ), which is much easier to solve than (BP_σ). This is perhaps the reason why most solvers proposed so far work with this regularized formulation.

4.3.1 Quadratic programming

In Section 4.1 we show how the basis pursuit problem can be rewritten as a linear program by splitting x into positive and negative parts. This same technique can be applied to convert (QP_λ) into a convex quadratic program (QP). Using the same notation, and expanding $\|z\|_2^2$ as $z^T z$, this gives

$$\underset{\bar{x}}{\text{minimize}} \quad \tfrac{1}{2}(\bar{x}^T \bar{A}^T \bar{A} \bar{x}) + (A^T b + \lambda e)^T \bar{x} \quad \text{subject to} \quad \bar{x} \geq 0 \tag{4.10}$$

There are several ways to solve quadratic programs, including-active set methods, interior-point methods, and gradient projection; see e.g, Nocedal and Wright [122]. We concentrate on the latter two, which have been successfully applied to sparse recovery.

Interior-point methods

Kim et al. [99] propose a primal log-barrier interior-point method for the quadratic programming formulation of regularized basis pursuit. Instead of using (4.10), they reformulate (QP_λ) as

$$\underset{x,u}{\text{minimize}} \quad \tfrac{1}{2}Ax - b_2^2 + e^T u, \quad st \quad -u \le x \le u.$$

The constraints are then incorporated into the objective using a log-barrier function:

$$b(u,x) = -\sum_{i=1}^{n} \log(u_i - x_i) - \sum_{i=1}^{n} \log(u_i - x_i) = \sum_{i=1}^{n} \log(u_i^2 + x_i^2),$$

giving a new objective $f(x,u) = \frac{1}{2}Ax - b_2^2 + e^T u + \mu \cdot b(u,x)$, which is optimized over the interior of the constraint $-u \le x \le u$.

At each iteration, a search direction d is computed based on the Newton system $Hd = -g$, where H denotes the symmetric positive define Hessian $\nabla^2 f$ and g denotes the gradient ∇f. For large-scale problems, it is prohibitively expensive to compute the exact solution to this system, and Kim et al. therefore suggest a truncated Newton method based on preconditioned conjugate gradients. This requires a truncation rule determining the accuracy of the solves, and a preconditioner. Because a dual-feasible point is cheaply computed, the truncation rule is based on the duality gap, and the Newton system is solved increasingly accurately as the iterates approach the solution. Kim et al. also describe a preconditioner consisting of a 2-by-2 block matrix with diagonal blocks. In the actual implementation of the algorithm, a slightly different preconditioner is used to avoid the potentially expensive computation of $\mathrm{diag}(A^T A)$, which is needed by their preconditioner. Once the search direction has been determined, a backtracking line search is done in combination with an Armijo condition for sufficient descent. At the end of each iteration, the parameter μ is updated to reduce the influence of the barrier term.

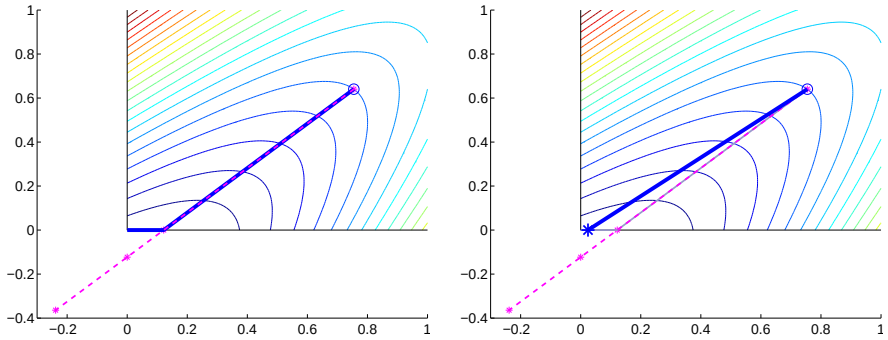
Besides being crucial for obtaining search directions for large-systems, the conjugate gradient method has the additional advantage of not requiring an explicit representation of A. This means that the algorithm can make use of fast routines for computing matrix-vector products, if these are available, thus facilitating the use of large dense implicit matrices.

As an alternative, Chen et al. [39] (see also Sardy et al. [130]) derive a primal-dual log-barrier formulation for the QP formulation given in (4.10).

Gradient projection

The GPSR solver by Figueiredo et al. [78] uses gradient projection to solve (4.10). Gradient projection methods proceed by projecting a step from the current iterate $x^{(k)}$ along the negative gradient $-\nabla f(x^{(k)}) = -A^T(Ax^{(k)} - b) + \lambda e$ onto the feasible set, giving a new point

$$x(\alpha) = \left(x^{(k)} - \alpha\beta\nabla f(x^{(k)})\right)_+,$$



▶ Figure 4.3: Contour plot of objective value in two variables in the nonnegative orthant, along with negative gradient (dashed) at current iterate (o). Plot (a) shows the projection arc (solid), while plot (b) shows the backtracking path from a single point on the projection arc, indicated by (*).

where $(x)_+ := \max\{0, x\}$, and β is an initial scaling parameter. The way this new point is used gives rise to two variants of the algorithm. In the first variant, a backtracking line search along the projection arc $x(\alpha)$ is done with $\alpha \in (0, 1]$. The second variant takes a fixed α and determines search direction $d^{(k)} = x(\alpha) - x^{(k)}$, and then performs a line search along $x = x^{(k)} + \gamma d^{(k)}$, for $\gamma \in [0, 1]$. Due to the quadratic nature of the objective function in (4.10), this second line search can be done exactly, with a closed-form optimal step length γ^*. We illustrate the two line-search methods in Figure 4.3.

Both of the above approaches are implemented in GPSR. To speed up the convergence for the latter method, GPSR uses a nonmonotonic approach where β is determined using a method by Barzilai and Borwein [5]. We discuss this method in more detail in Section 5.4.1.

4.3.2 Fixed-point iterations

Iterative soft-thresholding

In iterative soft-thresholding (IST) the fixed-point iterations are given by

$$x^{(k+1)} = s_{\gamma\lambda}(x^{(k)} - \gamma A^T(Ax^{(k)} - b)), \quad \gamma > 0, \tag{4.11}$$

with the soft-thresholding function $S_\lambda(x)$ defined as

$$[S_\lambda(x)]_i = S_\lambda(x_i) := \operatorname{sgn}(x_i) \cdot \max\{0, |x_i| - \lambda\}, \quad i = 1, \ldots, n. \tag{4.12}$$

This iteration has been derived from many different perspectives, including expectation maximization by Figueiredo and Nowak [77], and through the use of surrogate functionals by Daubechies et al. [50]; see Elad et al. [70] for an

overview. Perhaps the easiest way to obtain this iteration however, is by considering the optimality conditions for (QP_λ). The following derivation is based on work by Hale et al. [90].

Each minimizer x of (QP_λ) is characterized by

$$0 \in \nabla_x(\tfrac{1}{2}\|Ax-b\|_2^2) + \partial_x \lambda \|x\|_1 = A^T(Ax-b) + \lambda \partial_x \|x\|_1.$$

Moving the first term to the left-hand side, multiplying by $\gamma > 0$, and adding x to either side gives

$$x - \gamma A^T(Ax-b) \in x - \gamma\lambda \cdot \partial_x \|x\|_1 = F_{\gamma\lambda}(x), \tag{4.13}$$

where F_α denotes the operator $(I(\cdot) + \alpha \partial_x \|\cdot\|_1)$. This operator is separable, meaning that $[F_\alpha(x)]_i = f_\alpha(x_i)$, where

$$f_\alpha(x_i) = x_i + \alpha \cdot \partial |x_i|.$$

By inspection, and as illustrated in Figure 4.4, it is easily shown that the inverse $f_\alpha^{-1}(x_i)$ is given by the soft-thresholding operator $s_\alpha(x_i)$, and it follows from separability that the inverse of F is given by $[F_\alpha^{-1}(x)]_i = f_\alpha^{-1}(x_i)$. Applying this inverse to both sides of (4.13) yields

$$x = F_{\gamma\lambda}^{-1}(x - \gamma A^T(Ax-b)) = S_{\gamma\lambda}(x - \gamma A^T(Ax-b)),$$

which forms the basis for the fixed-point iteration (4.11). This iteration can also be formulated as a proximal forward-backward splitting process (see Combettes and Wajs [41]) for the minimization of $f_1(x) + f_2(x)$ with $f_1(x) = \frac{1}{2}\|Ax-b\|_2^2$ and $f_2(x) = \lambda\|x\|_1$. The forward step consists of setting

$$t^{i+1} = x^i - \gamma \nabla_x f_1(x^i) = x^i - \gamma A^T(Ax-b),$$

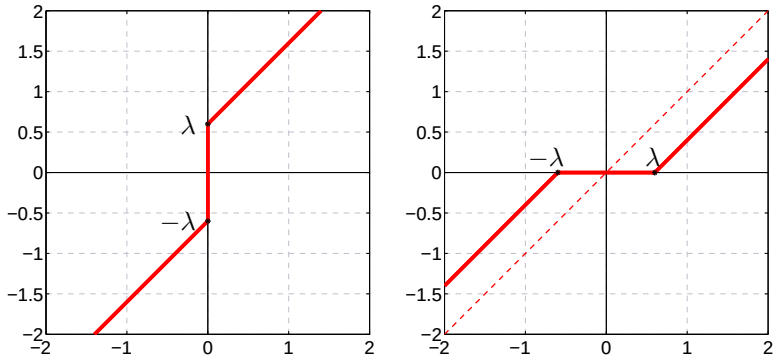
and can be seen as a gradient descend step with step length γ. The backward step applies the proximity operator to the intermediate result giving

$$x^{i+1} = \operatorname{prox}_{\gamma f_2} t^{i+1} = S_{\gamma\lambda}(t^{i+1}),$$

where the last equality follows from [41, Example 2.20].

The soft-thresholding iteration (4.13) is implemented in the FPC code by Hale et al. [90]. They note that IST can be very slow when applied directly to (QP_λ) with small regularization parameter λ. To overcome this, they propose a continuation approach in which a series of problems with decreasing λ is solved, each warm-started from the previous solution. This leads to a considerable speed up of the algorithm.

Importantly, Daubechies et al. [50] prove convergence of (4.11) with $\gamma = 1$ to the minimizer of (QP_λ), under the condition that the norm of A is strictly less than one. More general results can be found in Combettes and Wajs [41]. Convergence rates for (4.11) were derived by Hale et al. [90] and Bredies and Lorenz [22]. The latter also discuss an extension of the soft-thresholding iteration for group sparse minimization (1.7).



▶ Figure 4.4: Plots of (a) $f(x) = x+\lambda\cdot\partial_x|x|$, and (b) its inverse $f^{-1}(x) = S_\lambda(x)$.

Chapter 5

Solver framework

The common thread tying together many of the sparse-recovery formulations discussed so far, is that they minimize a convex objective function, subject to bounds on some measure of misfit. In this chapter we present a framework for solving an optimization problem that encompasses all of the formulations that we discuss in this thesis, and thus consider the general problem

$$\underset{x}{\text{minimize}} \quad \kappa(x) \quad \text{subject to} \quad \rho(Ax - b) \leq \sigma, \tag{5.1}$$

where κ and ρ are convex functions that are general enough to cover a wide range of problems in sparse recovery. This framework was first described in van den Berg and Friedlander [10], and we further extend and analyze it here. Chapter 6 discusses the application of the framework for several specific instances of the functions κ and ρ. An implementation of the framework, called SPGL1, is available on the web [9].

5.1 Assumptions

For the derivation of the solver framework for (5.1) we assume that κ and ρ are gauge functions. A function f is a gauge function [127], if it satisfies the following properties:

(1) f is nonnegative, $f(x) \geq 0$;

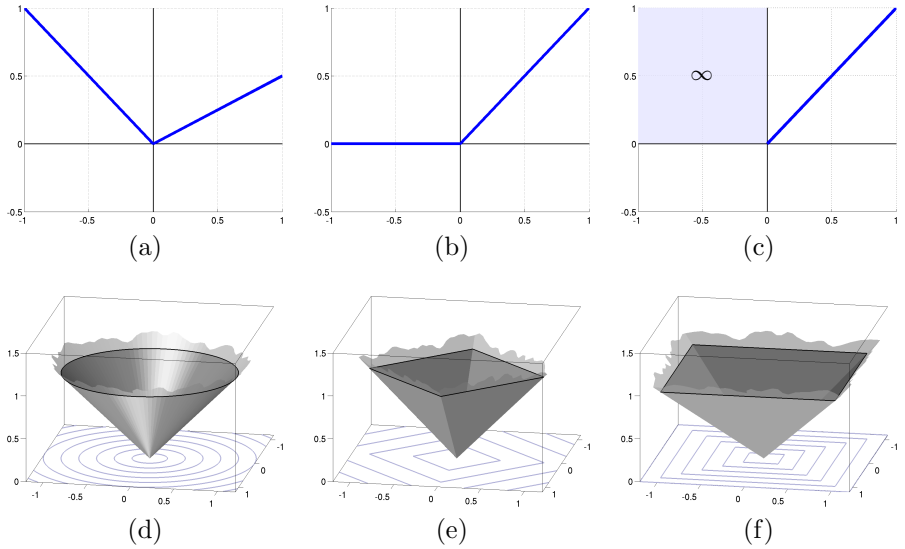
(2) f is positive homogeneous, $f(\alpha x) = \alpha f(x)$ for all $\alpha \geq 0$;

(3) f vanishes at the origin, $f(0) = 0$;

(4) f is convex.

To give an idea what such functions look like we plotted a number of one and two-dimensional gauge functions in Figure 5.1. Norms are special cases of gauge functions that additionally satisfy $f(x) = f(-x)$. Examples of this special class of gauge functions are depicted in Figure 5.1(d–f) and correspond to the ℓ_2, ℓ_1, and ℓ_∞ norms respectively.

As a second criterion, we require ρ to be continuously differentiable away from the origin. As an example, this is satisfied by the gauge function in Figure 5.1(d), but not by the ones in plots (e) and (f). Alternative criteria are possible, for example, κ is continuously differentiable away from the origin and A is full row-rank.



▶ Figure 5.1: One and two-dimensional gauge functions.

Finally, we assume that b is in the range of A, and without loss of generality that $b \neq 0$; otherwise the unique optimal solution to (5.1) for all $\sigma \geq 0$ would be given by $x^* = 0$. In section 5.2.2 we explain the rationale behind these conditions and describe the implications of not meeting them.

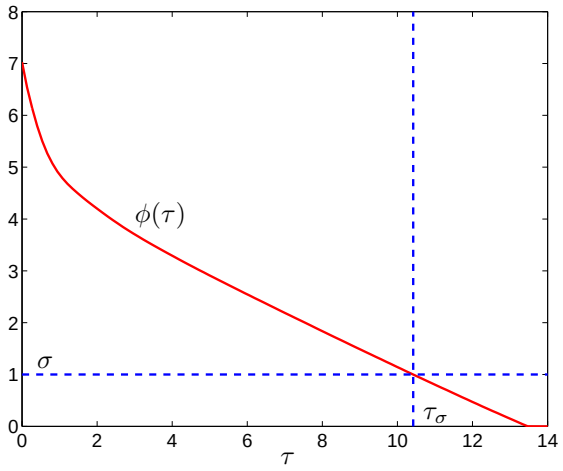
5.2 Approach

We begin the derivation of the solver framework by making the following observation. Let x_σ be any solution to (5.1). Then, by choosing $\tau = \tau_\sigma := \kappa(x_\sigma)$, it is also a solution of

$$\phi(\tau) := \underset{x}{\text{minimize}} \quad \rho(Ax - b) \quad \text{subject to} \quad \kappa(x) \leq \tau. \tag{5.2}$$

This is easily shown by assuming to the contrary that x_σ is not an optimal point for (5.2). Since x_σ is feasible by the choice of τ, there must be some point, x_τ, such that $\kappa(x_\tau) \leq \tau$, and $\rho(Ax_\tau - b) < \rho(Ax_\sigma - b)$. But this means that x_τ is strictly feasible for (5.1) and implies the existence of a positive scalar $\gamma < 1$ such that $\rho(\gamma Ax_\tau - b) \leq \sigma$ and $\kappa(\gamma x_\tau) = \gamma\kappa(x_\tau) < \tau$. In other words, if x_σ is not a solution of (5.2) then it could not have been an optimal solution for (5.1) either.

With the equivalence between these formulations established, the question becomes: how to find the value of τ corresponding to a given σ? For this we turn to the Pareto curve, which, when applied to (5.2), the optimal trade-off between τ and the resulting misfit $\rho(Ax^* - b)$. We express this curve as the function $\phi(\tau)$ for $\tau \in [0, \tau_0]$, where τ_0 is the smallest τ for which $\phi(\tau) = 0$.



▶ Figure 5.2: Pareto curve $\phi(\tau)$ with desired level of misfit σ and corresponding value of τ.

For the equivalence between (5.1) and (5.2), we need to find the smallest τ for which $\phi(\tau) \leq \sigma$, which amounts to solving the nonlinear equation $\phi(\tau) = \sigma$, as illustrated in Figure 5.2.

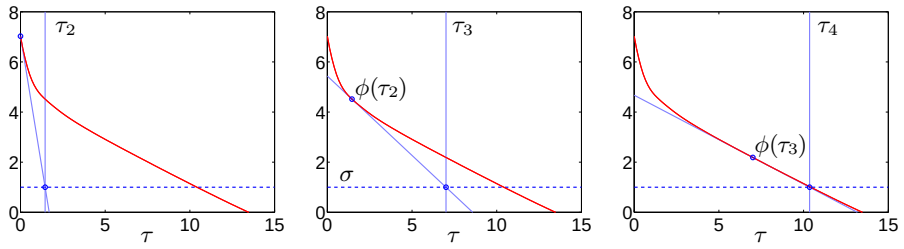
One way of solving the nonlinear equation $\phi(\tau) = \sigma$ is by applying Newton's method for nonlinear equations. This iterative method works by generating a linear approximation to the function at the current iterate and using it to find the next iterate. That is, given the current iterate τ_k, we create the model $f(\Delta\tau_k) = \phi(\tau_k) + \Delta\tau_k\phi'(\tau_i)$ and solve $f(\Delta\tau_k) = \sigma$ for step size $\Delta\tau_k$. With this step size, we have the following update:

$$\tau_{k+1} = \tau_k + \Delta\tau_k, \quad \text{with} \quad \Delta\tau_k = (\sigma - \phi(\tau_k))/\phi'(\tau_k). \tag{5.3}$$

We continue this process, illustrated in Figure 5.3, until some stopping criterion is satisfied. Summarizing the above gives the following algorithm.

Algorithm 5.1: Newton root-finding framework

Given A, b, σ, and tolerance $\epsilon \geq 0$
Set $k = 0$, $\tau_1 = 0$
repeat
    Increment iterate $k \leftarrow k + 1$
    Evaluate $\phi(\tau_k)$ by solving (5.2) and obtain x_k^*
    Set $r_k = b - Ax_k^*$
    Compute $\phi'(\tau_k)$
    Update $\tau_{k+1} = \tau_k + (\sigma - \phi(\tau_k))/\phi'(\tau_k)$
until $|\rho(r_k) - \sigma| \leq \epsilon$
Return $x^* = x_k^*$



▶ Figure 5.3: Three iterations of Newton's root-finding procedure for solving $\phi(\tau) = \sigma$.

For this approach to work the Pareto curve $\phi(\tau)$ must be differentiable, and both $\phi(\tau)$, and $\phi'(\tau)$ must be practically computable. In addition, we must have some guarantee that the iterations τ_k converge. We proceed by demonstrating differentiability, deriving the gradient, and analyzing the local convergence of Newton's method under inexact evaluations of $\phi(\tau)$ and $\phi'(\tau)$. We discuss solving the subproblems (5.2) in Section 5.4.

5.2.1 Differentiability of the Pareto curve

We prove differentiability of the Pareto curve using two results. The first result [127, Theorem 25.1] states that a convex function is differentiable at a point x if and only if the subgradient at that point is unique. Naturally, the gradient at x is then given by the unique subgradient. The second result [16, Propositions 6.1.2b, 6.5.8a] establishes that for convex problems of the form

$$p(c) := \underset{x}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad g(x) \leq c,$$

a vector y is a Lagrange multiplier if and only if $-y \in \partial p(c)$, provided that $p(c)$ is finite. By combining the two results we conclude that $p(c)$ is differentiable if and only if the Lagrange multiplier y is unique. To apply this result to $\phi(\tau)$ we first need to show that it is convex and bounded. Boundedness follows from the fact that $x = 0$ is feasible for all $\tau \geq 0$, ensuring that the objective $\rho(Ax - b)$ is bounded above by $\rho(b)$. For convexity we have the following result.

Lemma 5.1. *Let κ and ρ be gauge functions. Then the function $\phi(\tau)$ defined in (5.2) is convex and nonincreasing for all $\tau \geq 0$.*

Proof. The fact that $\phi(\tau)$ is nonincreasing is follows directly from the observation that the feasible set enlarges as τ increases. Next, consider any nonnegative scalars τ_1 and τ_2, and let x_1 and x_2 be the corresponding minimizers of (5.2). For any $\beta \in [0, 1]$ define $x_\beta = \beta x_1 + (1 - \beta)x_2$, and note that by convexity of κ,

$$\kappa(x_\beta) = \kappa(\beta x_1 + (1 - \beta)x_2) \leq \beta\kappa(x_1) + (1 - \beta)\kappa(x_2) = \beta\tau_1 + (1 - \beta)\tau_2,$$

where the last equality follows from the positive homogeneity of κ and ρ. With $\tau_\beta := \beta\tau_1 + (1-\beta)\tau_2$ this gives $\kappa(x_\beta) \leq \tau_\beta$, thus showing that x_β is a feasible point for (5.2) with $\tau = \tau_\beta$. For the objective we then have

$$\begin{aligned}\phi(\tau_\beta) &\leq \rho(Ax_\beta - b) = \rho(\beta Ax_1 - \beta b + (1-\beta)Ax_2 - (1-\beta)b) \\ &\leq \beta\rho(Ax_1 - b) + (1-\beta)\rho(Ax_2 - b) = \beta\phi(\tau_1) + (1-\beta)\phi(\tau_2),\end{aligned}$$

as required for convexity of ϕ. □

It then remains to show that the Lagrange multiplier for (5.2) it unique. For this we shift our attention to the dual problem.

Derivation of the dual

As a first step in the derivation of the dual, we rewrite (5.2) in terms of x and an explicit residual term r:

$$\underset{x,r}{\text{minimize}} \quad \rho(r) \quad \text{subject to} \quad Ax + r = b, \quad \kappa(x) \leq \tau. \tag{5.4}$$

The dual to this equivalent problem is given by

$$\underset{y,\lambda}{\text{maximize}} \quad \mathcal{L}(y,\lambda) \quad \text{subject to} \quad \lambda \geq 0, \tag{5.5}$$

where $y \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}$ are dual variables, and $\mathcal{L}$ is the Lagrange dual function, given by

$$\mathcal{L}(y,\lambda) := \inf_{x,r}\{\rho(r) - y^T(Ax + r - b) + \lambda(\kappa(x) - \tau)\}.$$

By separability of the infemum over x and r we can rewrite $\mathcal{L}$ in terms of two separate suprema, giving

$$\mathcal{L}(y,\lambda) = b^T y - \tau\lambda - \sup_r \{y^T r - \rho(r)\} - \sup_x \{y^T Ax - \lambda\kappa(x)\}. \tag{5.6}$$

We recognize the first supremum as the conjugate function of ρ; and the second supremum as the conjugate function of $\lambda\kappa(x)$. For a gauge function f, the conjugate function f^* can be conveniently expressed as

$$f^*(u) := \sup_w \; w^T u - f(w) = \begin{cases} 0 & \text{if } f^\circ(u) \leq 1 \\ \infty & \text{otherwise,} \end{cases} \tag{5.7}$$

where the polar of f is defined by

$$f^\circ(u) = \sup_w\{w^T u|\; f(w) \leq 1\}. \tag{5.8}$$

In case f is a norm, the polar reduces to the dual norm (for more details see Rockafellar [127] and Boyd and Vandenberghe [21, Section 3.3.1]). It follows from substitution of (5.7) in (5.6) that the dual of (5.2) is

$$\underset{y,\lambda}{\text{maximize}} \quad b^T y - \tau\lambda \quad \text{subject to} \quad \rho^\circ(y) \leq 1, \quad \kappa^\circ(A^T y) \leq \lambda. \tag{5.9}$$

Note that the constraint $\lambda \geq 0$ in (5.5) is implied by $\kappa^\circ(A^T y) \leq \lambda$, because κ° is nonnegative.

Importantly, in the case $\rho(r) = \|r\|_2$, we have $\rho^\circ(r) = \|r\|_2$, and the dual variables y and λ can easily be computed from the optimal primal solutions. To derive y, first note from (5.7) that

$$\sup_r \; y^T r - \|r\|_2 = 0 \quad \text{if} \quad \|y\|_2 \leq 1.$$

Therefore, $y = r/\|r\|_2$, and we can without loss of generality take $\|y\|_2 = 1$ in (5.9). To derive the optimal λ, note that as long as $\tau > 0$, λ must be at its lower bound $\kappa^\circ(A^T y)$, for otherwise we can increase the objective. Consequently, we take $\lambda = \kappa^\circ(A^T y)$. The dual variable y can be eliminated, and we arrive at the following necessary and sufficient optimality conditions for the primal-dual solution $(r_\tau, x_\tau, \lambda_\tau)$ of (5.4) with $\rho(r) = \|r\|_2$:

$$Ax_\tau + r_\tau = b, \quad \kappa(x_\tau) \leq \tau \qquad \text{(primal feasibility);} \tag{5.10a}$$

$$\kappa^\circ(A^T r_\tau) \leq \lambda_\tau \|r_\tau\|_2 \qquad \text{(dual feasibility);} \tag{5.10b}$$

$$\lambda_\tau(\kappa(x_\tau) - \tau) = 0 \qquad \text{(complementarity).} \tag{5.10c}$$

Uniqueness of the multiplier

Differentiability of the Pareto curve ultimately depends on the uniqueness of the Lagrange multiplier λ. This leads us to the following theorem.

Theorem 5.2. *Let ρ and κ be gauge functions, satisfying*

1. *ρ is differentiable away from the origin, or*
2. *κ is differentiable away from the origin and A has full row-rank.*

Then, on the open interval $\tau \in (0, \tau_0)$, the Pareto curve

$$\phi(\tau) := \underset{x}{\text{minimize}} \quad \rho(Ax - b) \quad \textit{subject to} \quad \kappa(x) \leq \tau$$

is strictly decreasing, and continuously differentiable with $\phi'(\tau) = -\lambda_\tau$.

Proof. For the first condition, note that differentiability of ρ implies strict convexity of the level sets of ρ° (see Lemma 5.3, below). For $\phi(\tau)$ to be differentiable it suffices to show that y^*, and hence λ, in (5.9) is unique. Suppose that this is not so, and that (y_1, λ_1) and (y_2, λ_2) are two distinct solutions to the dual formulation with $\lambda_i = \kappa^\circ(A^T y_i)$. Over the given range of τ, the primal objective is strictly greater than zero. By the Slater constraint qualification (which is trivially satisfied for $\tau > 0$) the primal-dual gap is zero and therefore the dual objective is strictly greater than zero. Moreover, we must have $\rho^\circ(y_1) = \rho^\circ(y_2) = 1$, for otherwise we could multiply each y by a scalar $\beta > 1$ which, given positivity of the dual objective and positive homogeneity of κ°, increases the objective by the same factor. Taking any convex combination $y = \alpha y_1 + (1 - \alpha) y_2$ with $0 < \alpha < 1$, gives $\rho^\circ(y) < 1$, by strict convexity. In

addition, $\kappa^\circ(A^T y) \leq \lambda$, with $\lambda = \alpha\lambda_1 + (1-\alpha)\lambda_2$. It can then be seen that the dual objective at (y, λ) either remains the same or increases (depending on κ). However, by the argument made above we can always multiply y by a scalar strictly larger than one, thus giving the desired result.

For the second condition we rewrite (5.9) as

$$\underset{y}{\text{maximize}} \quad b^T y - \tau\kappa^\circ(A^T y), \qquad \rho^\circ(y) \leq 1.$$

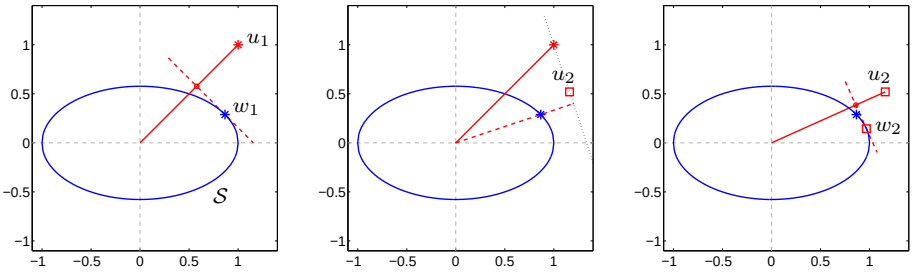
From differentiability of κ and the assumption on A, $-\kappa^\circ(A^T y)$ is strictly concave except along rays emanating from the origin. This guarantees a unique solution unless there exists an $\alpha > 0$ such that αy^* is both feasible and optimal. However, this is possible only when the objective is zero, which we exclude by assumption. □

Lemma 5.3. *Let f be a gauge function that is differentiable away from the origin. Then the level sets of f° are strictly convex.*

Proof. We start with the geometrical determination of the argument w for which (5.8) attains the supremum. From the differentiability of f it follows that $f(u) < \infty$ for all $u \in \mathbb{R}$. Consequently, there always exists a w for which $u^T w > 0$, and, by positive homogeneity, the supremum of (5.8) is always attained for a point w satisfying $f(w) = 1$. Now, let $\mathcal{S} := \{x \mid f(x) = 1\}$ denote the unit sphere induced by f. For a fixed u_1, we can write any $w_1 \in \mathcal{S}$ as $\alpha u_1 + v$, such that $u_1^T v = 0$. In order to determine $f^\circ(u_1)$ we therefore need to maximize α. As illustrated in Figure 5.4(a), this amounts to finding a supporting hyperplane of $\mathcal{S}$ with normal $u_1/\|u_1\|_2$. The point where the hyperplane touches $\mathcal{S}$ then gives the desired w_1, which is unique due to differentiability of f.

For the determination of strict convexity of the level sets of f°, we need to consider only the unit level set $\mathcal{S}^\circ := \{x \mid f^\circ(x) = 1\}$. Assume, by contradiction, that there exist two disjoint points $u_1, u_2 \in \mathcal{S}^\circ$ for which strict convexity is violated. That, is $f^\circ(u_\alpha) = 1$ for all $\alpha \in [0, 1]$, with $u_\alpha := \alpha u_1 + (1-\alpha)u_2$. Let $w_1 \in W_1$ and $w_2 \in W_2$ represent vectors in $\mathcal{S}$ such that $u_1^T w_1 = 1$, and $u_2^T w_2 = 1$. For $w \in \mathcal{S}$ to satisfy $w^T u_\alpha = 1$, we must have $w^T u_1 = w^T u_2$, which shows that $W_1 = W_2$, and that $w \in W_1$. However, we now claim that w_2 cannot be equal to w_1, unless $u_1 = u_2$. For $w_1 = w_2$ to hold, the supporting hyperplanes at that point need to be the same, due to differentiability of f. This also implies that the normals $u_1/\|u_1\|_2$ and $u_2/\|u_2\|$ of these hyperplanes are identical, but that means that $u_1 = u_2$, for otherwise $f^\circ(u_1) \neq f^\circ(u_2)$. We thus conclude that w_1 must be different from w_2 (see Figure 5.4(c)), and therefore that $\mathcal{S}^\circ$ is strictly convex. □

For an efficient implementation of the framework it is crucial that the Lagrange multiplier λ can be easily obtained. Indeed, in many situations solving the dual formulation (5.9) may be as hard as solving the original problem (5.1). The following theorem is therefore extremely convenient.



(a) (b) (c)

▶ Figure 5.4: Illustration for proof of Lemma 5.3, with (a) unit level set $\mathcal{S} := \{x \mid f(x) = 1\}$, vector u_1, and $w_1 \in \mathcal{S}$ maximizing $u_1^T w_1$, (b) another point u_2 satisfying $u_2^T w_1 = u_1^T w_1$, and (c) disjoint $w_2 \in \mathcal{S}$ maximizing $u_2^T w_2$.

Theorem 5.4. *Let κ be a gauge function and let $\rho(r) = \|r\|_2$. Then, on the interval $\tau \in (0, \tau_0)$, the Pareto curve is continuously differentiable with*

$$\phi'(\tau) = -\kappa^\circ(A^T r^*)/\|r^*\|_2, \tag{5.11}$$

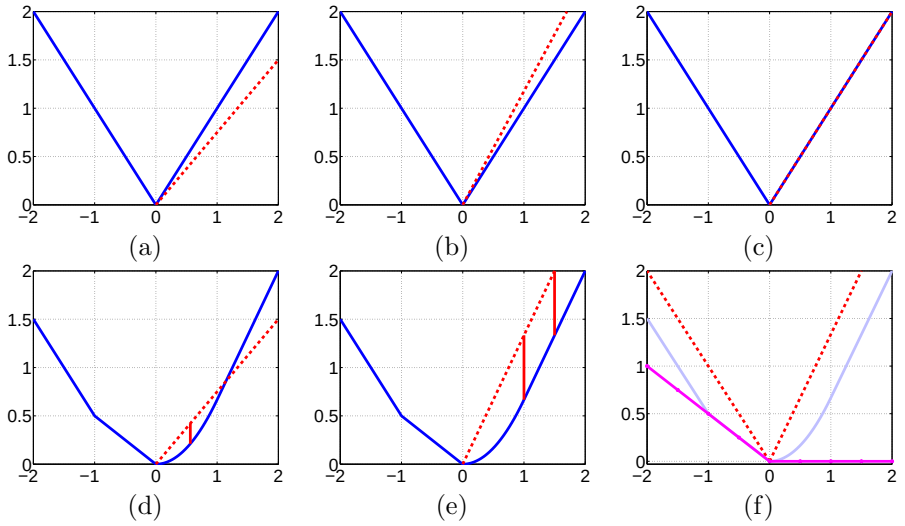
where $r^* = b - Ax^*$ with primal solution x^*.

Proof. By strict convexity of the level sets of ρ there must be a unique solution r^* to problem (5.4). Given this r^*, the second supremum in (5.6) must evaluate to zero with $r = r^*$ for any dual optimal y^*. With $\rho(r) = \|r\|_2$, it can be verified that the only y^* satisfying this criterion is given by $y^* = r^*/\|r^*\|_2$. Combining Theorem 5.2 with the fact that $\lambda = \kappa^\circ(A^T y^*)$ and κ° is positive homogeneous gives (5.11). □

5.2.2 Rationale for the gauge restriction

In Section 5.2.1 and Theorem 5.2 we limited κ and ρ to be gauge functions. We next consider what happens if this restriction is lifted.

Recall that in the derivation of the gradient of the Pareto curve, the dual formulation plays a key role. For the derivation of this dual we conveniently used the conjugate function given in (5.7). For more general nonnegative convex functions $f(x)$ that vanish at the origin, the supremum of $v^T x - f(x)$ over x has a similar characterization but with an important difference. Before discussing this, let us take a look at the idea behind the conjugate function for gauge functions. In the top half of Figure 5.5 we plot the function $f(x) = |x|$. The dotted line indicates the inner-product between v and x for different values of v. In plot (a) this product lies below $f(x)$ and the supremum of $v^T x - f(x)$ is easily seen to be zero. For the value of v in plot (b), the inner-product $v^T x$ exceeds $f(x)$, and by positive homogeneity of $f(x)$, we can make the gap arbitrarily large by increasing x. Plot (c) shows the critical value of v where $f(x)$ coincides with $v^T x$.



▶ Figure 5.5: Graphical illustration of the conjugate function for (a–c) a gauge function and (d–f) a more general function. The dotted line in (f) represents the recession function.

From the above we can see that the supremum is zero as long as $v^T x$ does not exceed $f(x)$ for any x. Because of positive homogeneity it suffices to check this for $f(x) = 1$, and therefore, without loss of generality for $f(x) \leq 1$. Doing so gives the requirement

$$\sup_x \{v^T x \mid f(x) \leq 1\} \leq 1,$$

which by (5.8) coincides with $f^\circ(v) \leq 1$. Combining both cases yields (5.7).

Omitting the condition of positive homogeneity changes the above characterization of the conjugate. As an example, consider the function $f(x)$ (solid line) in Figure 5.5(d), along with $v^T x$ (dotted line). Here, the supremum of $v^T x - f(x)$ is strictly positive without being unbounded. Function $f(x)$ is chosen such that it is quadratic on the interval $[0, 1]$, and linear on the intervals $(-\infty, -1]$, $[-1, 0]$, and $[1, \infty)$. It can be shown that the dotted line $v^T x$ in plot (e) corresponds to the critical value of v; increasing v by any amount gives an unbounded conjugate. This critical value is easily seen to coincide with the slope of $f(x)$ on the rightmost linear segment. In higher dimensions a similar relation holds and instead of looking directly at the function $f(x)$ to determine boundedness of the conjugate we need to look at the smallest gauge function $\gamma(x)$ such that $f(x) \leq \gamma(x)$ for all x. This function turns out to be the recession function $f0^+(x)$ of $f(x)$, which is defined as the function whose epigraph gives the largest cone in $\mathbb{R}^{n+1}$ that fits into the epigraph of $f(x)$ [127]. Figure 5.5(f) shows this recession function as a dotted line. Analogously we can define a function $f0_+(x)$ whose epigraph is the smallest cone containing $f(x)$, indicated by the dark solid line. With these two gauge functions we can characterize the

conjugate of nonnegative convex functions that vanish at the origin as

$$f^*(y) := \sup_w w^T u - f(w) = \begin{cases} 0 & \text{if } [f0_+]^\circ(x) \leq 1 \\ \in (0, \infty) & \text{if } [f0_+]^\circ(x) > 1,\ [f0^+]^\circ(x) \leq 1 \\ \infty & \text{otherwise.} \end{cases}$$

Unlike conjugate functions of gauges, this is no longer an indicator function. As a consequence, the expression for the dual problem is no longer as convenient as that for gauge functions. Moreover, obtaining the dual solution may require solving the dual problem, which of course greatly reduces the efficiency of our proposed method.

5.3 Practical aspects of the framework

The Newton root-finding framework in Algorithm 5.1 requires the computation of $\phi(\tau_k)$ and $\phi'(\tau_k)$. In practice these quantities will never be exact and it may even be too expensive to compute them to a high level of accuracy. In this section we derive a bound on the primal-dual gap to evaluate the optimality of a feasible point and study local convergence of Newton's method based on inaccurate function and gradient values. Throughout this section we assume that $\rho(r) := \|r\|_2$.

5.3.1 Primal-dual gap

The algorithms for solving (5.2), which we outline in Section 5.4, maintain feasibility of the iterates at all iterations. As a result, an approximate solution $\bar{x}_\tau$ and its corresponding residual $\bar{r}_\tau := b - A\bar{x}_\tau$ satisfy

$$\kappa(\bar{x}_\tau) \leq \tau, \quad \text{and} \quad \|\bar{r}_\tau\|_2 \geq \|r_\tau\|_2 > 0, \tag{5.12}$$

where the second set of inequalities holds because $\bar{x}_\tau$ is suboptimal and $\tau < \tau_0$. We can use these to construct dual variables

$$\bar{y}_\tau := \bar{r}_\tau / \|\bar{r}_\tau\|_2 \quad \text{and} \quad \bar{\lambda}_\tau := -\kappa^\circ(A^T \bar{y}_\tau),$$

which are dual feasible, i.e., they satisfy (5.10b). The objective of the dual problem (5.9), evaluated at any feasible point, gives a lower bound on the optimal value $\|r_\tau\|_2$. Therefore,

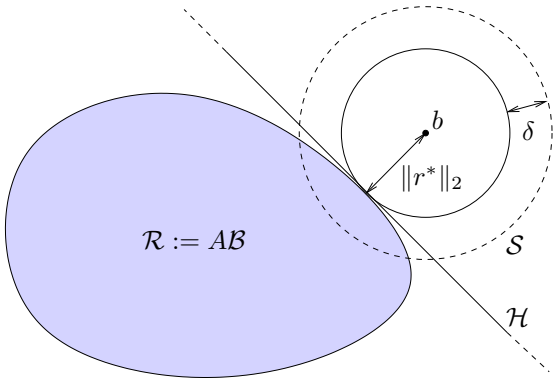
$$b^T \bar{y}_\tau - \tau \bar{\lambda}_\tau \leq \|r_\tau\|_2 \leq \|\bar{r}_\tau\|_2.$$

With the duality gap defined as

$$\delta_\tau := \|\bar{r}_\tau\|_2 - (b^T \bar{y}_\tau - \tau \bar{\lambda}_\tau), \tag{5.13}$$

we can bound the difference in objective as

$$0 \leq \|\bar{r}_\tau\|_2 - \|r_\tau\|_2 \leq \delta_\tau.$$



▶ Figure 5.6: Illustration for proof of Lemma 5.5.

5.3.2 Accuracy of the gradient

Let $\bar{\phi}(\tau) := \|\bar{r}_\tau\|_2$ be the objective value of (5.2) at the approximate solution $\bar{x}_\tau$. The duality gap δ_τ at $\bar{x}_\tau$ provides a bound on the difference between $\phi(\tau)$ and $\bar{\phi}(\tau)$. Assuming that A is full rank, we can use the relative duality gap $\eta_\tau := \delta_\tau / \|\bar{r}_\tau\|_2$ to obtain a bound on the difference between the derivatives $\phi'(\tau)$ and $\bar{\phi}'(\tau)$. In order to do so, we first bound the quantity $\|\bar{r}_\tau - r_\tau\|$.

Lemma 5.5. *Let (x^*, r^*) be the optimal solution for (5.4) with $\rho(r) := \|r\|_2$, and let δ be the duality gap for a feasible point $(\bar{r}, \bar{x})$. Then*

$$\|\bar{r} - r^*\|_2 \leq \sqrt{\delta^2 + 2\delta\|r^*\|_2}.$$

Proof. Formulation (5.4) with the given ρ is equivalent to

$$\underset{x}{\text{minimize}} \quad \|Ax - b\|_2 \quad \text{subject to} \quad \kappa(x) \leq \tau.$$

With the feasible set denoted by $\mathcal{B} := \{x \mid \kappa(x) \leq \tau\}$, the range of Ax over all feasible x is given by $\mathcal{R} := A\mathcal{B}$. Because convexity of sets is preserved under linear maps, $\mathcal{R}$ is convex. It follows from the use of the two-norm misfit that a unique separating hyperplane $\mathcal{H}$ exists between $\mathcal{R}$ and the Euclidean ball of radius $\|r^*\|_2$ around b (see Figure 5.6). Let $\mathcal{S}$ denote the $(n-1)$-sphere around b of radius $\|\bar{r}\|_2 \leq \|r^*\| + \delta$. The distance between $\bar{r}$ and r^* can be seen to be bounded by the distance between Ax^* and the points in $\mathcal{S} \cap \mathcal{R}$. This distance itself is bounded by distance from Ax^* to any point on the intersection between $\mathcal{S}$ and the separating hyperplane $\mathcal{H}$. For the latter distance d we have $(\|r^*\|_2 + \delta)^2 = \|r^*\|_2^2 + d^2$ and the stated result follows. □

We next derive a bound on the difference of the gradients based on $\bar{r} - r^*$.

Lemma 5.6. *Let $\|\bar{r} - r^*\|_2 \leq \gamma$, then*

$$|\bar{\phi}'(\tau) - \phi'(\tau)| \leq c \cdot \gamma / \|\bar{r}\|_2,$$

where c is a positive constant independent of τ.

Proof. Define $v := \bar{r} - r^*$. We consider two cases: first assume that $\bar{\phi}'(\tau) \leq \phi'(\tau)$. With the definition of $-\phi'(\tau)$ this gives the following

$$\begin{aligned}\phi'(\tau) - \bar{\phi}'(\tau) &= \frac{\kappa^\circ(A^T\bar{r})}{\|\bar{r}\|_2} - \frac{\kappa^\circ(A^T r^*)}{\|r^*\|_2} \leq \frac{\kappa^\circ(A^T r^*)}{\|\bar{r}\|_2} + \frac{\kappa^\circ(A^T v)}{\|\bar{r}\|_2} - \frac{\kappa^\circ(A^T r^*)}{\|r^*\|_2} \\ &\leq \frac{\kappa^\circ(A^T v)}{\|\bar{r}\|_2} \leq c \cdot \|v\|_2 / \|\bar{r}\|_2 \leq c \cdot \gamma / \|\bar{r}\|_2.\end{aligned}$$

The first inequality second line follows from the fact that $1/\|\bar{r}\|_2 \leq 1/\|r^*\|_2$. For the second case we consider $\phi'(\tau) \leq \bar{\phi}'(\tau)$. This gives

$$\begin{aligned}\bar{\phi}'(\tau) - \phi'(\tau) &= \frac{\kappa^\circ(A^T r^*)}{\|r^*\|_2} - \frac{\kappa^\circ(A^T\bar{r})}{\|\bar{r}\|_2} \leq \frac{\kappa^\circ(A^T r^*)}{\|r^*\|_2} - \frac{\kappa^\circ(A^T r^*)}{\|\bar{r}\|_2} + \frac{\kappa^\circ(A^T v)}{\|\bar{r}\|_2} \\ &\leq \frac{(\|\bar{r}\|_2 - \|r^*\|_2) \cdot \kappa^\circ(A^T r^*)}{\|r^*\|_2 \cdot \|\bar{r}\|_2} + \frac{\kappa^\circ(A^T v)}{\|\bar{r}\|_2} \\ &\leq \frac{\gamma}{\|\bar{r}\|_2} \cdot \frac{\kappa^\circ(A^T r^*)}{\|r^*\|_2} + \frac{\kappa^\circ(A^T v)}{\|\bar{r}\|_2} \leq c_1 \cdot \frac{\gamma}{\|\bar{r}\|_2} + c_2 \cdot \frac{\|v\|_2}{\|\bar{r}\|_2} \\ &\leq c \cdot \gamma / \|\bar{r}\|_2.\end{aligned}$$

For the first inequality we used the reverse triangle inequality. We then used the above inverse norm inequality, along with bounds on $\kappa^\circ(A^T r^*)\|r^*\|_2$, and likewise for $A^T v$. □

Combining these two results, we arrive at a bound on $|\bar{\phi}'(\tau) - \phi'(\tau)|$, in terms of the relative duality gap η.

Corollary 5.7. *Let η be the relative duality gap $\delta/\|\bar{r}\|_2$, then*

$$|\bar{\phi}'(\tau) - \phi'(\tau)| \leq c \cdot \sqrt{\eta^2 + 2\eta}.$$

Proof. From Lemma 5.5 and the fact that $\|r^*\|_2 \leq \|\bar{r}\|_2$, we have

$$\|\bar{r} - r^*\|_2 \leq \sqrt{\delta^2 + 2\delta\|\bar{r}\|_2} =: \gamma.$$

Applying Lemma 5.6 then gives the result. □

5.3.3 Local convergence rate

We start our analysis of the local convergence properties by defining

$$\gamma_\tau := \min\{\bar{\phi}(\tau) - \phi(\tau), |\bar{\phi}'(\tau) - \phi'(\tau)|\}. \tag{5.14}$$

Based on this quantity we bound the difference between the exact and inexact root-finding step size.

Lemma 5.8. *Let γ_τ be as defined in* (5.14)*. Then*

$$\epsilon(\tau) := \left| \frac{\phi(\tau) - \sigma}{\phi'(\tau)} - \frac{\bar{\phi}(\tau) - \sigma}{\bar{\phi}'(\tau)} \right| \leq c\gamma_\tau,$$

where c is a constant independent of τ.

Proof. Let α and β be such that $\bar{\phi}'(\tau) = \phi'(\tau) + \alpha$, and $\bar{\phi}(\tau) = \phi(\tau) + \beta$. It follows from the assumptions, that $|\alpha| \leq \gamma_\tau$, and $|\beta| \leq \gamma_\tau$. This allows us to write

$$\begin{aligned}
\epsilon(\tau) &= \left| \frac{\bar{\phi}'(\tau)\phi(\tau)}{\bar{\phi}'(\tau)\phi'(\tau)} - \frac{\phi'(\tau)\bar{\phi}(\tau)}{\bar{\phi}'(\tau)\phi'(\tau)} - \sigma \frac{(\bar{\phi}'(\tau) - \phi'(\tau))}{\bar{\phi}'(\tau)\phi'(\tau)} \right| \\
&\leq \sigma \left| \frac{(\bar{\phi}'(\tau) - \phi'(\tau))}{\bar{\phi}'(\tau)\phi'(\tau)} \right| + \left| \frac{(\phi'(\tau) + \alpha)\phi(\tau) - \phi'(\tau)(\phi(\tau) + \beta)}{\bar{\phi}'(\tau)\phi'(\tau)} \right| \\
&\leq \frac{\sigma\gamma_\tau}{\bar{\phi}'(\tau)\phi'(\tau)} + \left| \frac{\alpha\phi(\tau)}{\bar{\phi}'(\tau)\phi'(\tau)} - \frac{\beta}{\bar{\phi}'(\tau)} \right| \\
&\leq \frac{\sigma\gamma_\tau}{\bar{\phi}'(\tau)\phi'(\tau)} + \frac{\gamma_\tau\phi(\tau)}{\bar{\phi}'(\tau)\phi'(\tau)} + \frac{\gamma_\tau}{|\bar{\phi}'(\tau)|} \leq c\gamma_\tau.
\end{aligned}$$

The last step follows from the fact that there exists a constant $c_1 > 0$ such that $c_1 \leq \phi'(\tau)$ and $c_1 \leq \phi(\tau)$, and that $\phi(\tau) \leq \|b\|_2$. □

The following theorem establishes the local convergence rate of an inexact Newton method for $\phi(\tau) = \sigma$, where ϕ and ϕ' are known only approximately.

Theorem 5.9. *Suppose that A has full rank, $\sigma \in (0, \|b\|_2)$, and $\gamma_k := \gamma_{\tau_k} \to 0$. Then, if τ_0 is close enough to τ_σ, and τ_k remains in the interval $(0, \tau_0)$, the iteration* (5.3)*—with ϕ and ϕ' replaced by $\bar{\phi}$ and $\bar{\phi}'$—generates a sequence $\tau_k \to \tau_\sigma$ that satisfies*

$$|\tau_{k+1} - \tau_\sigma| = c\gamma_k + \nu_k|\tau_k - \tau_\sigma|, \tag{5.15}$$

where $\nu_k \to 0$ and c is a positive constant.

Proof. Because $\phi(\tau_\sigma) = \sigma \in (0, \|b\|_2)$, it follows from the definition of τ_0, that $\tau_\sigma \in (0, \tau_0)$. By Theorem 5.2 we have that $\phi(\tau)$ is continuously differentiable for all τ close enough to τ_σ, and so by Taylor's theorem,

$$\begin{aligned}
\phi(\tau_k) - \sigma &= \int_0^1 \phi'(\tau_\sigma + \alpha[\tau_k - \tau_\sigma])\, d\alpha \cdot (\tau_k - \tau_\sigma) \\
&= \phi'(\tau_k)(\tau_k - \tau_\sigma) + \int_0^1 \left[\phi'(\tau_\sigma + \alpha[\tau_k - \tau_\sigma]) - \phi'(\tau_k)\right] \cdot d\alpha\, (\tau_k - \tau_\sigma) \\
&= \phi'(\tau_k)(\tau_k - \tau_\sigma) + \omega(\tau_k, \tau_\sigma),
\end{aligned}$$

where the remainder ω satisfies

$$\omega(\tau_k, \tau_\sigma)/|\tau_k - \tau_\sigma| \to 0 \quad \text{as} \quad |\tau_k - \tau_\sigma| \to 0. \tag{5.16}$$

It follows from the definition of $\phi'(\tau)$ and from standard properties of matrix norms that $c_l\sigma_m(A) \leq -\phi'(\tau) \leq c_u\sigma_1(A)$. We use this property, along with Lemma 5.8, and the definition $\Delta\tau_k = \big(\sigma - \bar{\phi}(\tau_k)\big)/\bar{\phi}'(\tau_k)$, to establish that

$$\begin{aligned}
|\tau_{k+1} - \tau_\sigma| &= |\tau_k - \tau_\sigma + \Delta\tau_k| \\
&= \left| -\frac{\bar{\phi}(\tau_k) - \sigma}{\bar{\phi}'(\tau_k)} + \frac{1}{\phi'(\tau_k)}\big[\phi(\tau_k) - \sigma - \omega(\tau_k, \tau_\sigma)\big] \right| \\
&\leq \left| \frac{\phi(\tau_k) - \sigma}{\phi'(\tau_k)} - \frac{\bar{\phi}(\tau_k) - \sigma}{\bar{\phi}'(\tau_k)} \right| + \left| \frac{\omega(\tau_k, \tau_\sigma)}{\phi'(\tau_k)} \right| \\
&= c_1\gamma_k + c_2|\omega(\tau_k, \tau_\sigma)| \\
&= c_1\gamma_k + \nu_k|\tau_k - \tau_\sigma|,
\end{aligned}$$

with positive constants c_1 and c_2, and $\nu_k := c_2|\omega(\tau_k, \tau_\sigma)|/|\tau_k - \tau_\sigma|$. When τ_k is sufficiently close to τ_σ, (5.16) implies that $\nu_k < 1$. Apply the above inequality recursively $\ell \geq 1$ times to obtain

$$|\tau_{k+\ell} - \tau_\sigma| \leq c_1 \sum_{i=1}^{\ell} (\nu_k)^{\ell-i}\gamma_{k+i-1} + |\tau_k - \tau_\sigma| \cdot \prod_{i=1}^{\ell} \nu_{k+1-i},$$

and because $\gamma_k \to 0$ and $\nu_k < 1$, it follows that $\tau_{k+\ell} \to \tau_\sigma$ as $\ell \to \infty$. Thus $\tau_k \to \tau_\sigma$, as required. By again applying (5.16), we have that $\nu_k \to 0$. □

Note that if (5.2) is solved exactly at each iteration, such that $\gamma_k = 0$, then Theorem 5.9 shows that the convergence rate is superlinear, as we expect of a standard Newton iteration. In effect, the convergence rate of the algorithm depends on the rate at which $\delta_k \to 0$. If A is rank deficient, then the constant c in (5.15) is infinite; we thus expect that ill-conditioning in A leads to slow convergence unless $\gamma_k = 0$, i.e., ϕ is evaluated accurately at every iteration. For related results see also Dembo et al. [52, Section 4].

5.3.4 First root-finding step

A natural choice for the initial value of τ is zero. Besides guaranteeing that τ is not too large, it also has the advantage that the solution of $\phi(0)$ and $\phi'(0)$ are often available in closed form. That is, whenever $\kappa(x) = 0$ implies $x = 0$, the only feasible point, and therefore the solution to (5.2) at $\tau = 0$ is $x^* = 0$. It follows from the definition of $\phi(\tau)$ that $\phi(0) = \|Ax^* - b\|_2 = \|b\|_2$. Moreover, by application of (5.11) in Theorem 5.4 we have $\phi'(0) = -\kappa^\circ(A^T b)/\|b\|_2$. This means that under the above condition on κ the first root-finding step is essentially free, requiring only a single matrix-vector product with A^T and one evaluation of κ°.

5.4 Solving the subproblems

Ultimately, the root-finding framework depends on an efficient method to solve subproblems of the form (5.2). In this section we discuss two such methods. Due to the need to obtain efficient gradients of the Pareto curve, we restrict ourselves to the case where $\rho(r) = \|r\|_2$, giving the generalized Lasso-formulation

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|Ax - b\|_2^2 \quad \text{subject to} \quad \kappa(x) \leq \tau. \tag{5.17}$$

As we will see in Chapter 6, many problems of interest have a Euclidean or Frobenius norm misfit. This means that the restriction on ρ does not greatly limit the practical application of the framework.

Note however that both subproblem solvers are easily adapted to deal with more general ρ (or ρ^2), provided that the gradient exists everywhere on the feasible set and is available to the solver. Indeed, both methods are capable of solving

$$\underset{x}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad x \in \Omega, \tag{5.18}$$

where Ω is some non-empty convex set. For this they require the evaluation of objective $f(x)$, the gradient $g(x) := \nabla f(x)$, and the orthogonal projection of arbitrary points v onto the feasible set

$$P(v) = P_\Omega(v) := \underset{u \in \Omega}{\text{argmin}} \ \|u - v\|_2. \tag{5.19}$$

For (5.17) we have $\Omega := \{x \mid \kappa(x) \leq \tau\}$, and we discuss efficient projection algorithms for various functions $\kappa(x)$ in Chapter 6. For (5.17), the evaluation of both the objective and the gradient essentially reduces to matrix-vector products with A and A^T. This allows matrix A to be an implicit operator, provided that routines for matrix-vector products are available. In Chapter 8 we present a Matlab toolbox aimed at the construction of large implicit matrices with associated operations, including matrix-vector multiplication.

As an aside, we remark that (5.17) reduces to the well-known Lasso problem [140] when $\kappa(x) = \|x\|_1$. The Lasso problem can be solved using a variety of methods, including homotopy [123], least angle regression (LARS) [68], and the active-set method by [124, 149]. In our context, the use of the homotopy and LARS algorithms makes no sense because, as discussed in Section 4.2, they can be more efficiently used to solve the basis pursuit denoise problem directly.

5.4.1 Spectral projected gradients

The first method for solving (5.17) is the spectral projected-gradient (SPG) algorithm introduced by Birgin et al. [17]. Their method generates a sequence of feasible iterates $\{x_k\}$, which is shown to converge to a constrained stationary point, which corresponds to the global minimizer for (5.17). Given the current iterate x_k, the algorithm proceeds as follows. First we compute search direction $d_k = -g(x_k)$ and determine an initial step length β. This gives the first point

$x_k + \beta d_k$, which is then projected onto the feasible region to obtain the first feasible trial point $P(x_k + \beta d_k)$. To ensure sufficient descent we then perform a backtracking line-search either directly from the first trial point or by reducing β. Once an acceptable trial point is found we set x_{k+1} and proceed with the next iteration until a stopping criterion is met. A detailed version of this algorithm is given in Algorithm 5.2, and we now proceed with a more elaborate discussion of each of the main steps.

Algorithm 5.2: Scaled projected gradient algorithm with (left) backtracking line search, and (right) curvilinear line search.

Given A, b, $x_0 \in \Omega$, τ
while *not converged* **do**
  $g = A^T(Ax - b)$
  $\beta =$ initial steplength
  $d = \mathcal{P}(x - \beta g) - x$
  $\alpha = 1$
  repeat
    $\bar{x} = x + \alpha d$
    Reduce α
  until *sufficient descent*
  $x = \bar{x}$

Given A, b, $x_0 \in \Omega$, τ
while *not converged* **do**
  $g = A^T(Ax - b)$
  $\beta =$ initial steplength
  $\alpha = 1$
  repeat
    $\bar{x} = \mathcal{P}(x - \alpha\beta g)$
    Reduce α
  until *sufficient descent*
  $x = \bar{x}$

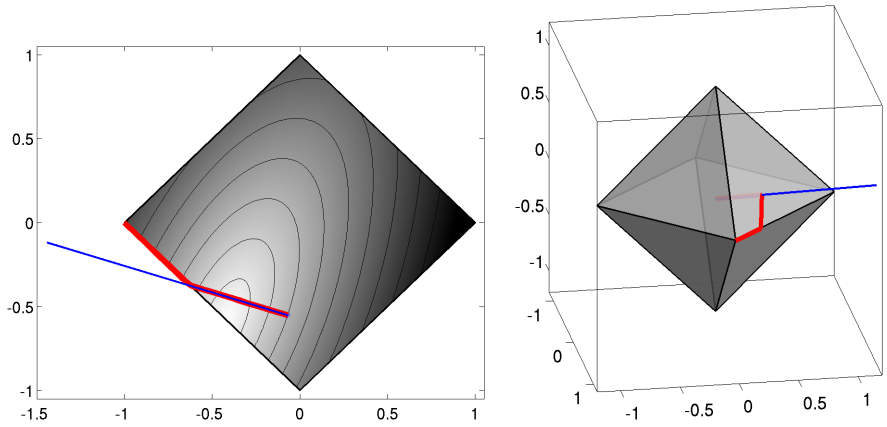
Initial step length. In the spectral projected gradient method, the initial step length is determined by

$$\beta_k := \frac{\langle \Delta x, \Delta x \rangle}{\langle \Delta x, \Delta g \rangle}, \tag{5.20}$$

where $\Delta x := x_k - x_{k-1}$ is the difference between the current and previous iterate and $\Delta g := g(x_k) - g(x_{k-1})$ the corresponding difference in gradients. This step length was originally introduced by Barzilai and Borwein [5] for unconstrained minimization and motivated as follows. Given the current iterate we like to take a step $x_{k+1} = x_k + S_k g(x_k)$, where $S_k = \beta_k I$. The value of β_k is chosen such that it minimizes $\|\Delta x - \beta_k \Delta g\|$ or $\|\Delta x/\beta_k - \Delta g\|$. Solving the latter problem gives (5.20).

Birgin et al. [17] introduced the Barzilai-Borwein step length to constrained optimization and relate the step length to the quadratic approximation $q(x)$ of $f(x)$, with approximate Hessian $B_k = \beta_k^{-1} I$. Alternative choices for step length have been studied as well, for example by Dai and Fletcher [46].

Backtracking line search. The two components that characterize a line search are the way in which trial points are generated, and the conditions under which a trial point is deemed acceptable. For the scaled projected gradient method we adopt two different ways of generating trial points. The first method generates a sequence of point along the projection arc $x(\alpha) = P(x_k + \alpha\beta_k d_k)$,



▶ Figure 5.7: Illustration of projection arcs on the two- and three-dimensional cross-polytope.

with $\alpha \geq 0$, and d_k the (scaled) search direction. Examples of projection arcs for the nonnegative orthant and the ℓ_1-ball are illustrated in Figures 4.3(a) and 5.7 respectively. In practice we typically sample the projection arc along the set $\alpha = \mu^i$, for trial points $i \geq 0$ and with $\mu \in (0, 1)$. As noted by Birgin et al. [17], the distance between two consecutively projected points can be small or even zero for corner points. In such situations, which can arise when $x_k + \beta_k d_k$ is far from the feasible region, it may be necessary to use extremely small values of α in order to reach new points and make progress with the line search. To safeguard against this we follow Birgin et al. [17] and use backtracking from a single projection when projection along the arc fails. In this approach we generate a new search direction $\hat{d}_k := P(x_k + \beta_k d_k) - x_k$, and perform regular backtracking line search along this direction, as shown in Figure 4.3(b). This approach requires only a single projection onto the feasible set and guarantees that all points $x_k + \alpha\hat{d}_k$ are feasible for $\alpha \in [0, 1]$.

For the acceptance of trial points we use the non-monotonic generalization of the Armijo condition by Grippo et al. [88] for projected line search by Bertsekas [15]. Under the typical Armijo condition sufficient descent is required relative to the most recent iterate, leading to monotonically decreasing objective values. While this property is convenient from a theoretical perspective it was recognized that enforcing monotonicity can considerably slow the rate of convergence [88]. In order to overcome this problem, while maintaining the theoretical properties of Armijo, Grippo et al. [88] suggest to define sufficient descent relative to the maximum objective value over a fixed number M of the most recent iterates. With sufficient-descent parameter γ, this amounts to the condition

$$f(x(\alpha)) \leq \max_{i \in [\max\{1, k-M\}, k]} f(x_i) + \gamma g(x_k)^T (x(\alpha) - x_k).$$

Stopping criteria. The most natural stopping criterion for any optimization algorithm is sufficient optimality of the current iterate. In SPG this can be done based on the projected gradient or on the primal-dual gap derived in Section 5.3.1. Regarding the gradient, a sufficient condition for a point x_k to be optimal for (5.17) is that $P(x_k - g(x_k)) = x_k$. This is satisfied either when $g(x_k)$ is zero, or when the negative gradient is outward orthogonal to the boundary of the feasible set at x_k.

In our SPGL1 implementation [9] it was found that the number of iterations required to reach a sufficiently small duality gap can be prohibitively large. At the same time it was observed that the primal objective may be near optimal, meaning that the large gap was predominantly due to the sensitivity of the dual feasible point based on r_k. Therefore, although not ideal, the stopping criterion also takes into account the progress of the objective. Together with the duality gap, this approach seems to work well in practice, although it does sometimes lead to undesirable behavior in the root-finding process when the gradient is insufficiently accurate. Finally, to ensure finite termination of the algorithm, we include additional constraints on the number of iterations and matrix-vector products taken.

5.4.2 Projected quasi-Newton

As an alternative to SPG as a solver for subproblem (5.17), we developed the projected quasi-Newton (PQN) approach described in Schmidt et al. [131]. The algorithm is aimed at solving the more general problem (5.18), in particular when the objective $f(x)$ and the corresponding gradient $g(x)$ are expensive to compute while projection onto the feasible set Ω is relatively cheap.

The PQN algorithm consist of an outer and an inner level. In the outer level, information from subsequent iterates is used to construct a quadratic model of the objective function around the current x_k:

$$q_k(x) := f_k + (x - x_k)^T g_k + \tfrac{1}{2}(x - x_k)^T B_k (x - x_k).$$

Here, the positive definite matrix B_k is an approximation to the Hessian $\nabla^2 f(x)$ is constructed implicitly based on L-BFGS updates [102]. The inner level uses the quadratic model to find a search direction by approximately solving:

$$\underset{x}{\text{minimize}} \quad q_k(x) \quad \text{subject to} \quad x \in \Omega. \tag{5.21}$$

Once determined, the outer level does a line search along $x_k + \alpha d_k$ with $\alpha \in [0, 1]$ to determine the next feasible iterate x_{k+1}. A summary of the algorithm is given in Algorithm 5.3.

Solving the PQN subproblem. Problem (5.21) is a special case of (5.18) and can be solved using the SPG algorithm described in the previous section. Because the objective function depends only on B_k and the fixed vector $g(x_k)$, we can solve the subproblem without evaluating $f(x)$ or $g(x)$. Another important property, following from positive definiteness of B_k, is that any x satisfying

Algorithm 5.3: Projected quasi-Newton algorithm.

```
Given A, b, x_0 ∈ Ω, τ
Initialize Hessian approximation Ĥ
while not converged do
    Solve (5.21) for search direction d
    Initialize α = 1
    repeat
        x̄ = x + αd
        Update α
    until sufficient descent
    x = x̄
    Update Hessian approximation Ĥ
return x
```

$q(x) < q(0)$ can be used to construct a valid search direction $d_k := x - x_k$ that is guaranteed to be descent direction for the original problem. An illustration of the quadratic model on the two-dimensional cross-polytope is given in Figure 5.7(b).

Chapter 6

Application of the solver framework

In Chapter 5 we derived a framework for problems of the form

$$\underset{x}{\text{minimize}} \quad \kappa(x) \quad \text{subject to} \quad \|Ax - b\|_2 \leq \sigma, \tag{6.1}$$

and proposed two algorithms for solving the subproblems arising in the framework. Both algorithms require the evaluation of $\kappa(x)$ and $\kappa^\circ(x)$, and projection onto the feasible set. In this chapter we establish these components for a number of problem formulations that commonly arise in sparse recovery and satisfy the conditions required by the framework.

For each problem formulation we give a motivation for its importance and provide potential applications. We then describe the two essential components needed by the solvers: orthogonal projection onto the feasible set, and the evaluation of the polar of $\kappa(x)$. For most problems considered, $\kappa(x)$ is a norm, in which case the polar is usually referred to as the dual norm. For sake of uniformity in section names, etc., we often use the term 'polar', even in the context of norms.

6.1 Basis pursuit denoise

The most widely used and studied convex sparse recovery formulation to date is the basis pursuit denoise problem (BP_σ), described and motivated in Chapter 1. In this section we consider a generalization of this problem, which allows positive weights to be associated to each coefficient, giving the weighted basis pursuit denoise formulation:

$$\underset{x}{\text{minimize}} \quad \|Wx\|_1 \quad \text{subject to} \quad \|Ax - b\|_2 \leq \sigma, \tag{6.2}$$

where W is a diagonal weighting matrix. The most prominent use of this formulation (with $\sigma = 0$) is the reweighted ℓ_1 algorithm by Candès et al. [37], which is somewhat akin to iteratively reweighted least-squares. Empirically it is shown [37] that the probability of recovering a vector x_0 from observation $b = Ax_0$ benefits from this reweighting, compared to a single iteration of unweighted basis pursuit.

The weighted formulation fits (6.1) with $\kappa(x) := \|Wx\|_1$ and we next discuss the derivation of $\kappa^\circ(x)$, and the Euclidean projection onto the (weighted) one-norm ball.

6.1.1 Polar function

For the polar of the weighted one-norm $\|Wx\|_1$ we use the following, more general, result.

Theorem 6.1. *Let κ be a gauge function, and let Φ be an invertible matrix. Given $\kappa_\Phi(x) := \kappa(\Phi x)$, then*

$$\kappa_\Phi^\circ(x) = \kappa^\circ(\Phi^{-T}x).$$

Proof. Applying the definition of gauge polars (5.8) to κ_Φ gives

$$\kappa_\Phi^\circ(x) = \sup_w\{\langle w, x\rangle \mid \kappa_\Phi(w) \le 1\} = \sup_w\{\langle w, x\rangle \mid \kappa(\Phi w) \le 1\}.$$

Now define $u = \Phi w$. Invertibility of Φ then allows us to write

$$\kappa_\Phi^\circ(x) = \sup_u\{\langle \Phi^{-1}u, x\rangle \mid \kappa(u) \le 1\} = \sup_u\{\langle u, \Phi^{-T}x\rangle \mid \kappa(u) \le 1\} = \kappa^\circ(\Phi^{-T}x),$$

as desired. □

Applied to the weighted one-norm, we obtain the following result.

Corollary 6.2. *Let W be any non-singular diagonal matrix. Then the dual norm of $\|Wx\|_1$ is given by $\|W^{-1}x\|_\infty$.*

6.1.2 Projection

Good performance of the SPGL1 and PQN algorithms depends crucially on being able to efficiently compute projections onto the feasible set. We now show that this is indeed possible for projection onto the one-norm ball. We also discuss the modifications necessary to incorporate weights.

Unweighted one-norm projection

The Euclidean projection of a point $c \in \mathbb{R}^n$ onto the one-norm ball of radius τ is given by the solution of

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|c - x\|_2^2 \quad \text{subject to} \quad \|x\|_1 \le \tau. \tag{6.3}$$

This problem can be solved with a specialized $\mathcal{O}(n \log n)$ algorithm, which we derive next. The algorithm was independently developed by Candès and Romberg [30], Daubechies et al. [51], and van den Berg and Friedlander [10]. Despite similarities in the approach, our implementation is nevertheless quite different from that of the others.

In case $\|c\|_1 \le \tau$, the projection is trivially found by setting $x = c$. For all other cases there exists for each τ a scalar λ such that

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|c - x\|_2^2 + \lambda\|x\|_1 \tag{6.4}$$

has the same solution as (6.3). The solution of this penalized formulation is obtained by applying (componentwise) the soft-thresholding operator (4.12).

The problems (6.3) and (6.4) are equivalent when we choose λ such that $\|S_\lambda(c)\|_1 = \tau$. Once this value is know we can solve (6.3) by applying S_λ on c. To simplify the discussion, let a_i, $i = 1, \ldots, n$, be the absolute values of c in decreasing order. It is convenient to add $a_{n+1} := 0$ and to define the function

$$\theta(\lambda) := \|S_\lambda(c)\|_1. \tag{6.5}$$

It can be verified from (4.12) that $\theta(\lambda)$ is strictly decreasing in λ from $\theta(a_{n+1}) = \theta(0) = \|c\|_1$ to $\theta(a_1) = 0$. Therefore, there exists an integer k such that

$$\theta(a_k) \leq \tau < \theta(a_{k+1}). \tag{6.6}$$

Suppose that k is given. Then it remains to find a correction $\delta \geq 0$ such that $\theta(a_k - \delta) = \tau$. It follows from the definition of θ that

$$\theta(a_i) = \sum_{j=1}^{n} \max\{0, a_j - a_i\} = \sum_{j=1}^{i} (a_j - a_i) = \Big(\sum_{j=1}^{i} a_j\Big) - i \cdot a_i. \tag{6.7}$$

For a δ such that $0 \leq \delta \leq a_i - a_{i+1}$, it similarly holds that

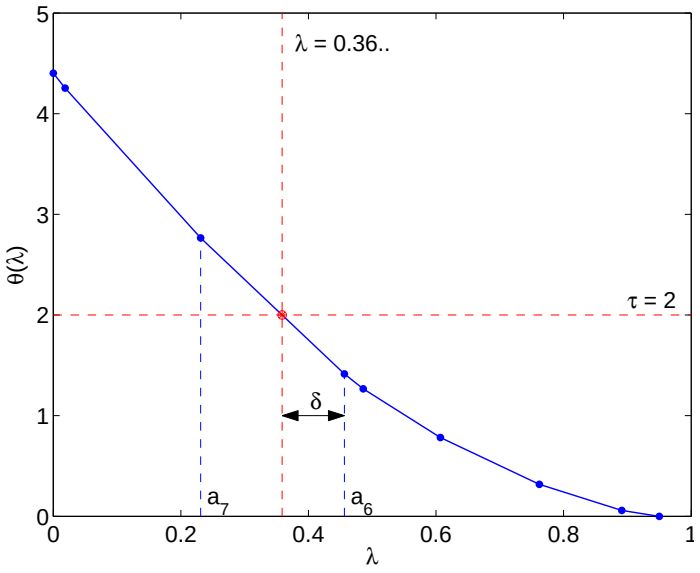
$$\theta(a_i - \delta) = \sum_{j=1}^{n} \max\{0, a_j - (a_i - \delta)\} = \sum_{j=1}^{i} (a_j - a_i + \delta) = \delta i + \theta(a_i), \tag{6.8}$$

which is linear in δ on the specified interval, as illustrated in Figure 6.1. These results suggest the following projection algorithm:

1. sort the absolute values of c to get vector a;
2. find k satisfying (6.6);
3. solve $\theta(a_k - \delta) = \tau$ for δ; based on (6.8), this gives $\delta = (\tau - \theta(a_k))/k$;
4. compute $x := S_\lambda(c)$ with $\lambda = a_k - \delta$.

Because of the sorting step, the overall time complexity of these algorithms is $\mathcal{O}(n \log n)$. The third step takes constant time, and soft-thresholding in step four requires $\mathcal{O}(n)$ time. For the second step we need to be a little careful because direct evaluation of $\theta(a_k)$ using (6.7) for all k could take $\mathcal{O}(n^2)$ operations. Fortunately, it is not hard to see that $\theta(a_i) = \theta(a_{i+1}) - i(a_i - a_{i+1})$, thus yielding an $\mathcal{O}(1)$ update to compute $\theta(a_k)$ for successive k.

By replacing the first two steps of the algorithm with more advanced techniques it is possible to bring the complexity down to an expected run time linear in n; see Duchi et al. [67] and van den Berg et al. [13].



▶ Figure 6.1: Plots of $\theta(\lambda)$ for a random c of length eight, showing the relationship between τ, λ, and δ. The points on the curve indicate the value of $\theta(\lambda)$ for values of λ coinciding with some a_i.

Weighted one-norm projection

The Euclidean projection onto a weighted one-norm ball is given by

$$P_\tau(x) = \underset{z}{\operatorname{argmin}} \quad \tfrac{1}{2}\|x - z\|_2^2 \quad \text{subject to} \quad \|Wz\|_1 \leq \tau. \tag{6.9}$$

It is easily seen that for all i with $w_i = 0$, the solution is given by $P_\tau(x)_i = x_i$. Therefore, without loss of generality, we can assume that $w_i \neq 0$. In addition, since $P_0(x) = 0$, we only need to consider the case where $\tau > 0$.

An efficient projection algorithm can again be derived starting with the Lagrangian formulation of (6.9),

$$\underset{x}{\text{minimize}} \;\; \tfrac{1}{2}\|x - z\|_2^2 + \lambda\|Wx\|_1.$$

For x to be a solution of this problem it suffices that the subgradient of the objective, evaluated at x, contains zero. Since the objective is separable, this reduces to

$$0 \in x_i - c_i + \lambda|w_i| \cdot \operatorname{sgn}(x_i).$$

This condition is satisfied by setting x_i to the soft-thresholded value of z_i with threshold $\lambda|w_i|$, i.e.,

$$x^*(\lambda) := \left[\operatorname{sgn}(z_i) \cdot \max\{0, |z_i| - \lambda|w_i|\}\right]_i. \tag{6.10}$$

The function $\theta_w(\lambda) := \|Wx(\lambda)\|_1$ is again non-increasing and piecewise linear with breaks occurring at $\lambda = |z_i/w_i|$. With only minor changes, this allows us to use the same algorithm as for the regular one-norm projection. With careful implementation it is also possible to apply the linear time algorithm in [67, 13].

6.2 Joint and group sparsity

In this section we develop the theory and methods required to apply the root-finding framework to the group sparse recovery problem (1.7). We also consider the MMV problem (1.8) as a special case of group recovery. To start, let us look at a practical application of the latter problem in source localization.

6.2.1 Application – Source localization

In astronomy, radio telescopes are used to image distant objects that emit radio waves. Due to the distance of the source, these waves arrive at the telescope as plane waves, and the design of the parabolic dish (see illustration in Figure 6.2(a)) is such that all reflected signals travel the same distance to the central receiver. This causes the signals to arrive at the receiver in phase, thus leading to an amplified signal. By changing the direction of the telescope, it can focus on different parts of the sky.

An alternative approach is to use an array of stationary omnidirectional sensors, as shown in Figure 6.2(b). Focusing the array at a particular direction can then be achieved by applying an appropriate delay or phase shift to the signal of each sensor prior to summation.

Let $S_{i,j}$ represent a set of narrowband signals arriving from angles θ_i, $i = 1, \dots, n$, at time t_j, $j = 1, \dots, k$. Under the narrowband assumption, and provided the spacing between sensors is sufficiently small, the sensor output can be formulated as

$$B = A(\theta)S + N$$

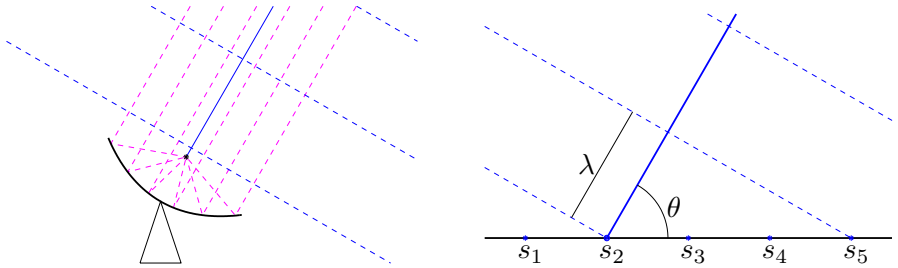
where N is a matrix of random additive noise, and $A(\theta)$ represents the phase shift and gain matrix. Each column in B contains the measurement values taken by the sensors at a given point in time. Each column in A matrix corresponds to a single arrival angle θ_i, and each row corresponds to one of m sensors. In the two-dimensional case, with sensor positions given by $(p_i, 0)$, $i = 1, \dots, m$, the complex phase shift for sensor i, relative to the origin at $(0, 0)$, for angle θ_j, is given by

$$A_{i,j} = \exp\{2\imath\pi \cos(\theta_j)p_i/\lambda\},$$

with $\imath = \sqrt{-1}$, and wavelength λ.

In source localization the angles θ, and consequently $A(\theta)$, are unknown. When the number of sources is small, we can discretize the space into a set of discrete angles ψ and find a sparse approximate solution to

$$A(\psi)X = B.$$



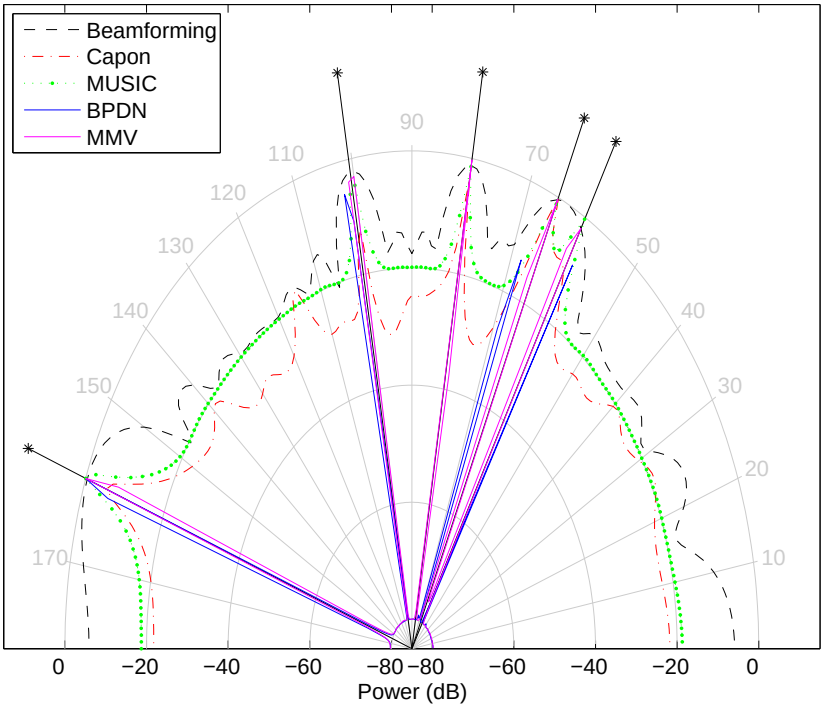
▶ Figure 6.2: Focusing planar waves from a given direction using (a) a radio telescope, (b) an array of omnidirectional sensors.

Assuming sources are stationary or moving slowly with respect to the observation time, we would like the nonzero entries in X (corresponding to different angles of arrival) to be restricted to a small number of rows. This motivates the approach taken by Malioutov et al. [109], which amounts exactly to the MMV problem (1.8). Note that the misfit between $A(\psi)X$ and B in this case is due not only to the signal noise N, but also to the angular discretization ψ.

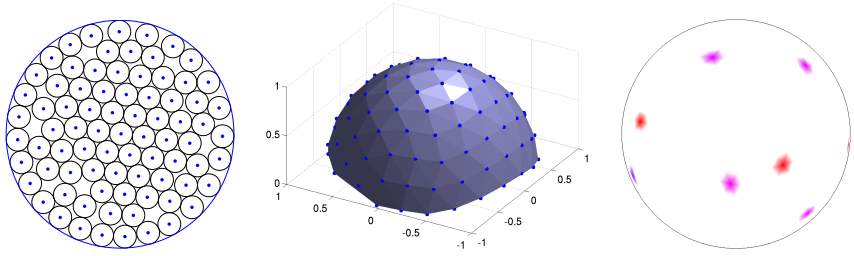
As a two-dimensional example, consider an array of twenty omnidirectional sensors spaced at half the wavelength of interest. Impinging on this array are five far-field sources, located at angles 60°, 65°, 80°, 100.5°, and 160° relative to the horizon. We are given twenty observations measured by the array at a signal-to-noise ratio of 10dB. To recover the direction of arrival we discretize the (two-dimensional) space at an angular resolution of 1° and compare MMV and BPDN to beamforming, Capon, and MUSIC (see [107] for more information). The resulting powers from each direction of arrival are shown in Figure 6.3. Both MMV and BPDN can be seen to give very good results.

For a more realistic example we consider a three-dimensional source localization problem. Because of the added dimension, discretizing the space of all possible directions and positioning of sensors becomes somewhat harder. For the placement of the sensors we choose a near-uniform distribution within a unit norm circle, which is conveniently done using existing circle-packing results [134]. For the discretization of arrival directions we choose a set of points $P := \{p_i\}$ on the unit sphere, with their potential energy, $\sum_{i \neq j} 1/\|p_i - p_j\|_2$, approximately minimized. The final result is then obtained by discarding the points in the halfspace below the surface. The discretizations obtained for 80 sensors and 100 directions are shown in Figure 6.4, along with a signal coming from eight directions.

Given a set of such signals we run MMV and obtain an approximate solution on the coarse grid; see Figure 6.5(a,e). Because the actual direction of arrival are unlikely to be captured by the grid we can expect there to be some misfit. This misfit can be reduced by locally refining the grid based on the approximate signal directions by adding new directions of arrival and repeating reconstruction until the desired results it reached. This is illustrated in Figure 6.5(b–d,f–h).

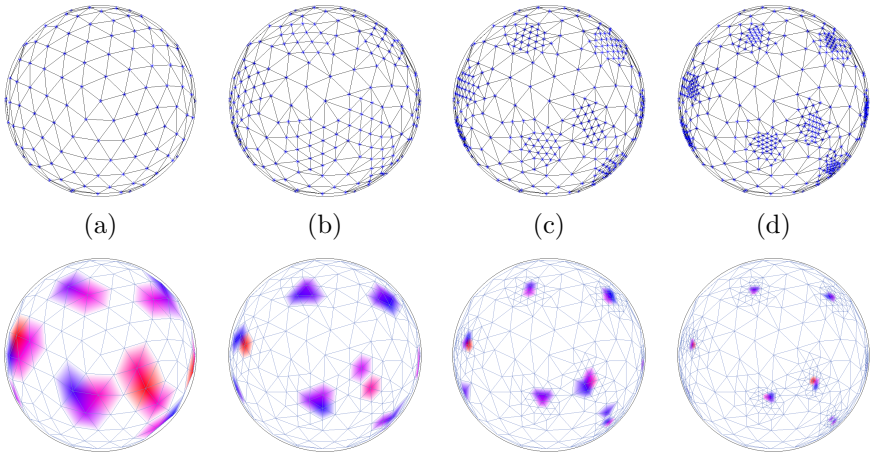


▶ Figure 6.3: Angular spectra obtained using beamforming, Capon, MUSIC, BPDN, and MMV for five uncorrelated far-field sources at angles 60°, 65°, 80°, 100.5°, and 160°. The directions of arrival are discretized at 1° resolution.



(a) sensor positions (b) coarse direction grid (c) actual directions

▶ Figure 6.4: Configuration of (a) 80 sensors on the plane, (b) coarse grid of 100 arrival directions on the half-sphere, and (c) top view of actual arrival directions on a fine grid.



▶ Figure 6.5: Grid of (a) initial arrival directions, and (b–d) after up to three refinement steps, along with corresponding solutions (e–f).

6.2.2 Polar functions of mixed norms

The $\|\cdot\|_{p,q}$ norm defined in (1.9), and the sum of norms in (1.7), are special cases of more general mixed norms. The dual or polar function for such norms can be obtained using the following result.

Theorem 6.3. *Let σ_i, $i = 1, \ldots, k$ represent disjoint index sets such that $\bigcup_i \sigma_i = \{1, \ldots, n\}$. Associate with each group i a primal norm $\|\cdot\|_{p_i}$ with dual norm $\|\cdot\|_{d_i}$. Denote $v_i(x) = \|x_{\sigma_i}\|_{p_i}$ and $w_i(x) = \|x_{\sigma_i}\|_{d_i}$. Let $\|\cdot\|_p$ be a norm such that for all vectors $q, r \geq 0$, $\|q\|_p \leq \|q + r\|_p$, and let $\|\cdot\|_d$ denote its dual norm. Then the dual norm of $|||\cdot|||_p := \|v(\cdot)\|_p$ is given by $|||\cdot|||_d := \|w(\cdot)\|_d$.*

Proof. First we need to show that $|||x|||_p$ is indeed a norm. It is easily seen that the requirements $|||x|||_p \geq 0$, $|||\alpha x|||_p = |\alpha| \cdot |||x|||_p$, and $|||x|||_p = 0$ if and only if $x = 0$ hold. For the triangle inequality we need to show that $|||q + r|||_p \leq |||q|||_p + |||r|||_p$. Using the triangle inequality of the group norms, we have that $0 \leq v(q + r) \leq v(q) + v(r)$, componentwise. The assumption on the outer norm $\|\cdot\|_p$ then allows us to write

$$|||q + r|||_p = \|v(q + r)\|_p \leq \|v(q) + v(r)\|_p \leq \|v(q)\|_p + \|v(r)\|_p = |||q|||_p + |||r|||_p,$$

as desired. Next, to derive the dual norm we note that the dual of any norm is defined implicitly by the following equation:

$$\|x\|_d := \{\sup_z x^T z \mid \|z\|_p \leq 1\}.$$

For each given subvector x_{σ_i}, the supremum of $x_{\sigma_i}^T z_{\sigma_i}$ with $\|x_{\sigma_i}\|_{p_i} \leq 1$ is given by $w_i(x) := \|x_{\sigma_i}\|_d$. This quantity scales linearly with the bound we impose on the primal norm, i.e., under the condition $\|x_{\sigma_i}\|_{p_i} \leq t_i$, the supremum becomes $t_i w_i(x)$. Writing $w = \{w_i(x)\}_i$, we can write the supremum over the entire vectors as

$$\begin{aligned}\{\sup_z\ x^T z \mid \|z\|_p \leq 1\} &= \{\sup_{t,z} \sum_i t_i x_{\sigma_i}^T z_{\sigma_i} \mid \|t\|_p \leq 1,\ \|z_{\sigma_i}\|_{p_i} \leq 1\} \\ &= \{\sup_t\ t^T w \mid \|t\|_p \leq 1\}.\end{aligned}$$

But this is exactly the definition of $\|w\|_d$. □

Note that the requirement on the outer primal norm $\|\cdot\|_p$ is essential in deriving the triangle inequality, but does not hold for all norms. For example, the norm $\|x\| := \|\Phi x\|_2$, with any non-diagonal, invertible matrix Φ does not satisfy the requirement. Importantly though, the requirement is satisfied for the common norms $\|x\|_\gamma$, $1 \leq \gamma \leq \infty$;

By repeated application of the above theorem we can derive the dual of arbitrarily nested norms. For example, the function $\|x_{(1)}\|_2 + \max\{\|x_{(2)}\|_1, \|x_{(3)}\|_2\}$ applied to a vector consisting of $x_{(1)}, x_{(2)}, x_{(3)}$ is a norm whose dual is given by $\max\{\|x_{(1)}\|_2, \|x_{(2)}\|_\infty + \|x_{(3)}\|_2\}$. Likewise, by vectorizing X and imposing appropriate groups, we can use Theorem 6.3 to obtain the dual of $\|X\|_{p,q}$:

Corollary 6.4. *Given $p, q \geq 1$. Let p' and q' be such that $1/p + 1/p' = 1$ and $1/q + 1/q' = 1$. Then*

$$(\|X\|_{p,q})^\circ = \|X\|_{p',q'}.$$

6.2.3 Projection

Euclidean projection onto the norm-balls induced by $\|X\|_{1,2}$ and $\sum_i \|x_{\sigma_i}\|_2$ can be reduced to projection onto the one-norm ball, using the following theorem.

Theorem 6.5. *Let $c_{(i)}$, $i = 1, \ldots, n$, be a set of vectors, possibly of different length. Then the solution $x^* = (x^*_{(1)}, \ldots, x^*_{(n)})$ of*

$$\underset{x}{\text{minimize}} \quad \sum_i \tfrac{1}{2}\|c_{(i)} - x_{(i)}\|_2^2 \quad \text{subject to} \quad \textstyle\sum_i \|x_{(i)}\|_2 \leq \tau, \tag{6.11}$$

can be obtained by solving the one-norm projection problem

$$\underset{u}{\text{minimize}} \quad \tfrac{1}{2}\|v - u\|_2^2 \quad \text{subject to} \quad \|u\|_1 \leq \tau, \tag{6.12}$$

with $v_i = \|c_{(i)}\|_2$, and setting

$$x^*_{(i)} = \begin{cases} (u^*_i / v_i) \cdot c_{(i)} & \text{if } v_i \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Proof. We first treat the special case where $v_i = 0$. The projection for these groups is trivially given by $x_{(i)} = c_{(i)} = 0$, thus allowing us to exclude these groups. Next, rewrite (6.11) as

$$\underset{x,u}{\text{minimize}} \quad \sum_i \tfrac{1}{2}\|c_{(i)} - x_{(i)}\|_2^2 \quad \text{subject to} \quad \|x_{(i)}\|_2 \leq u_i, \ \|u\|_1 \leq \tau. \quad (6.13)$$

Fixing $u = u^*$ makes the problem separable, reducing the problem for each i to

$$\underset{x_{(i)}}{\text{minimize}} \quad \tfrac{1}{2}\|c_{(i)} - x_{(i)}\|_2^2 \quad \text{subject to} \quad \|x_{(i)}\|_2^2 \leq u_i^2.$$

For $u_i = 0$ this immediately gives $x_{(i)} = 0$. Otherwise the first-order optimality conditions on x require that the gradient of the Lagrangian,

$$\mathcal{L}(\lambda_i) = \tfrac{1}{2}\|c_{(i)} - x_{(i)}\|_2^2 + \lambda(\|x_{(i)}\|_2^2 - u_i^2),$$

with $\lambda \geq 0$, be equal to zero; that is, $\nabla\mathcal{L}(x_{(i)}) = x_{(i)} - c_{(i)} + 2\lambda_i x_{(i)} = 0$. It follows that $x_{(i)} = c_{(i)}/(1 + 2\lambda_i) = \gamma_i c_{(i)}$, such that $\|x_{(i)}\|_2 = \gamma\|c_{(i)}\|_2 = u_i$ (which also holds for $u_i = 0$). Using the definition $v_i = \|c_{(i)}\|_2$, and the fact that $x_{(i)} = \gamma_i c_{(i)}$, we can rewrite each term of the objective of (6.11) as

$$\begin{aligned} \|c_{(i)} - x_{(i)}\|_2^2 &= c_{(i)}^T c_{(i)} - 2\gamma_i c_{(i)}^T c_{(i)} + \gamma_i^2 c_{(i)}^T c_{(i)} \\ &= \|c_{(i)}\|_2^2 - 2\gamma_i\|c_{(i)}\|_2^2 + \gamma_i^2\|c_{(i)}\|_2^2 \\ &= v_i^2 - 2\gamma_i v_i^2 + \gamma_i^2 v_i^2 = (v_i - \gamma_i v_i)^2 = (v_i - u_i)^2. \end{aligned}$$

Finally, substituting this expression into (6.13) yields (6.12), because the constraint $\|x_{(i)}\|_2 = u_i$ is automatically satisfied by setting $x_{(i)} = \gamma_i c_{(i)} = (u_i/v_i) \cdot c_{(i)}$. □

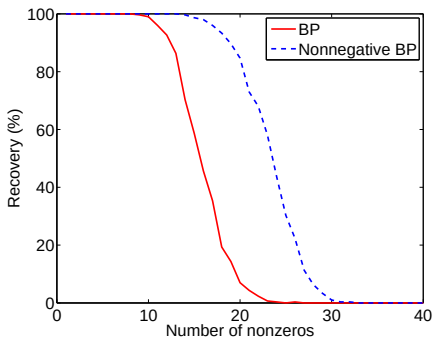
This proof can be extended to deal with weighted group projection. In that case the problem reduces to projection onto the weighted one-norm ball.

6.3 Sign-restricted formulations

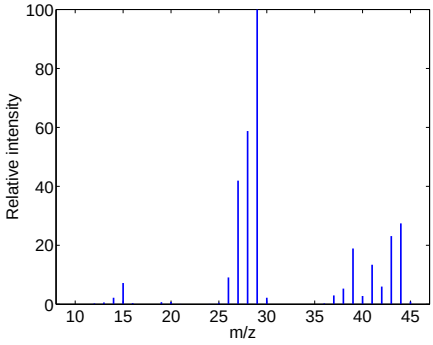
In this section we consider the generalized sign-restricted basis pursuit denoise formulation

$$\underset{x}{\text{minimize}} \quad \kappa(x) \quad \text{subject to} \quad \|Ax - b\|_2 \leq \sigma, \quad x_i \begin{cases} \geq 0 & i \in \mathcal{I}_p \\ \leq 0 & i \in \mathcal{I}_n, \end{cases}$$

where $\mathcal{I}_p$, $\mathcal{I}_n \subseteq \{1, \ldots, n\}$ are two (possibly empty) disjoint sets of indices, and κ is a norm. The use of sign information as a prior can greatly help with the recovery of sparse signals. For nonnegative basis pursuit (NNBP) this advantage was theoretically shown by Donoho and Tanner [55, 54], as pointed out in Section 2.5.3. Indeed, Figure 6.6 shows that NNBP clearly outperforms general basis pursuit in the fraction of randomly chosen sparse x_0 that can be recovered from $b = Ax_0$. We next describe a problem in analytical chemistry that can conveniently be expressed as an NNBP.



▶ Figure 6.6: Equivalence breakdown curve for 40×80 random Gaussian matrix averaged over 300 random nonnegative x_0.



▶ Figure 6.7: Mass spectrum of propane using electron ionization [116].

6.3.1 Application – Mass spectrometry

Mass spectrometry is a powerful method used to identify the chemical composition of a sample. There exist several different approaches but we restrict ourselves here to electron ionization (EI) mass spectrometry in which analyte molecules are ionized using high-energy electrons. Such ionization is often followed by fragmentation of the molecule with existing bonds breaking and possibly new bonds forming in a manner that is characteristic of the original molecule. Mass spectrometers register the relative abundance of ions for a range of mass-to-charge (m/z) ratios, which can be used to deduce the chemical make-up of the compound [112, 133]. Once analyzed, the mass spectrum can subsequently be used as a signature to recognize the corresponding compound.

As an example, consider the mass spectrum of propane (C_3H_8), illustrated in Figure 6.7. The molecular ion generally has a mass of 44 u (unified atomic mass units) consisting of three ^{12}C atoms and eight ^{1}H atoms (the small peak at 45 m/z is due to presence of ^{13}C isotopes). The peaks directly preceding 44 m/z are ions with increasingly many hydrogen atoms missing. The most intense peak at 29 m/z corresponds to ethyl ($C_2H_5^+$) ions, which is again preceded by ions with fewer hydrogen atoms. Finally, there are the peaks around the methyl (CH_3^+) ion at 15 m/z.

When analyzing mixtures[2] the components contribute independently to the measured spectrum. In case of electron ionization, this superposition is linear [112] and the spectrum can thus be written as a nonnegative combination of the individual spectra. Thus, given a mixed mass spectrum b we can identify the components by forming a dictionary of spectra A from possible substances and finding a sparse nonnegative solution x satisfying $Ax \approx b$. This can be

[2] In practice mixtures are generally separated using one of several types of chromatograph before introduction into the mass spectrometer.

formulated as

$$\underset{x}{\text{minimize}} \quad \|x\|_1 \quad \text{subject to} \quad \|Ax - b\|_2 \leq \sigma, \quad x \geq 0. \tag{6.14}$$

A similar formulation was recently proposed by Du and Angeletti [65].

To evaluate the approach we created a dictionary A containing the mass spectra of 438 compounds obtained from the NIST Chemistry WebBook [116]. Each spectrum was normalized and expanded to contain the intensities for 82 m/z values. The spectrum b of a synthetic mixture was created by adding the spectra of twelve compounds with randomly selected ratios; see Figures 6.8. We then solved (6.14) with appropriate σ for b, and likewise for a measurement contaminated with additive noise. The results of this simulation are shown in Figure 6.8(f).

6.3.2 Polar function

There are two ways to restrict x to have a certain sign pattern: by adding explicit constraints, or by extending κ to be an extended real function that is infinity at all x violating the desired sign pattern. In this section we consider the second approach. The following result gives the polar of the desired function.

Theorem 6.6. *Let $\mathcal{C}$ be the intersection of (half)spaces $\mathcal{S}_i$ with*

$$\mathcal{S}_i = \{x \in \mathbb{R}^n \mid x_i \geq 0\}, \quad \text{or} \quad \mathcal{S}_i = \{x \in \mathbb{R}^n \mid x_i \leq 0\}, \quad \text{or} \quad \mathcal{S}_i = \mathbb{R}^n.$$

Further, let $\|x\|$ be any norm that is invariant under sign changes in x, and let $\kappa(x)$ be given by

$$\kappa(x) = \begin{cases} \|x\| & \text{if } x \in \mathcal{C}, \\ \infty & \text{otherwise.} \end{cases}$$

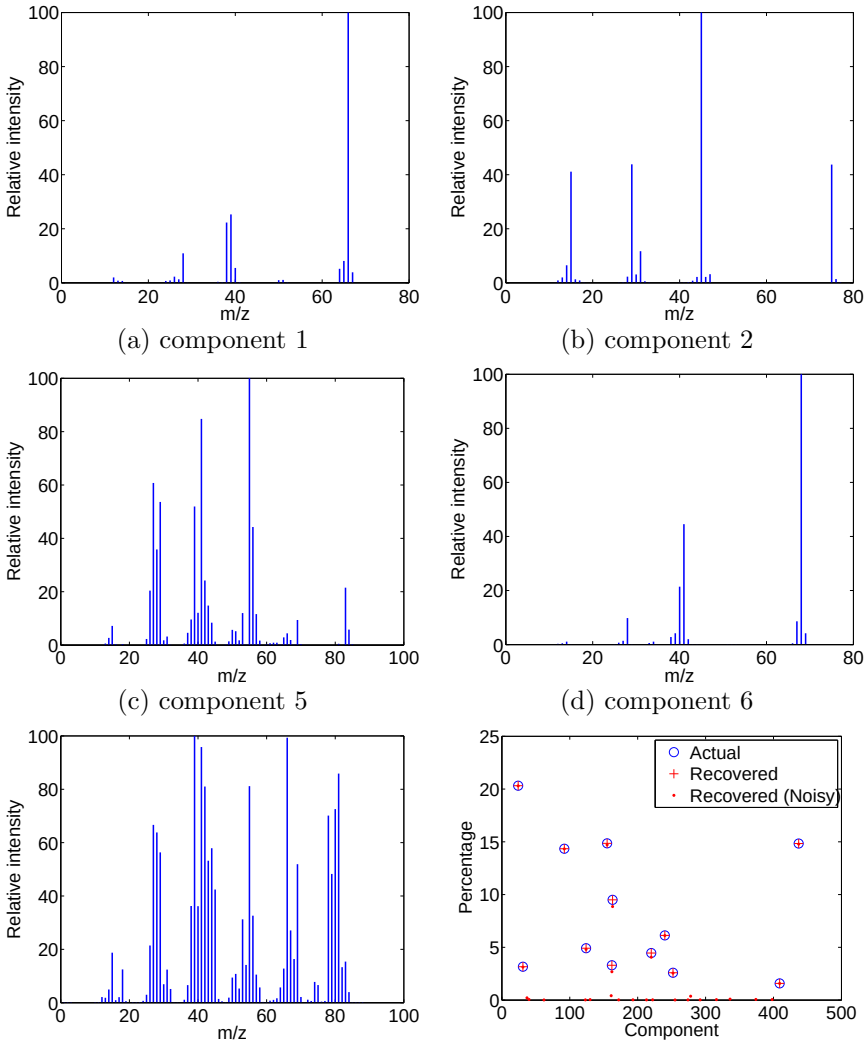
Then, with $\|\cdot\|_*$ the dual norm and $\mathcal{P}(x)$ the Euclidean projection onto $\mathcal{C}$,

$$\kappa^\circ(x) = \|\mathcal{P}(x)\|_*.$$

Proof. We consider three different cases: (i) $x \in \mathcal{C}$, (ii) $x \in \mathcal{C}^\circ$, and (iii) $x \notin \mathcal{C} \cup \mathcal{C}^\circ$, which cover all $x \in \mathbb{R}^n$. In the first case, we have $\mathcal{P}(x) = x$ and therefore only need to show that the polar $\kappa^\circ(x)$ in (5.8) is attained by some $w \in \mathcal{C}$. This implies that for those x it does not matter whether we use $\kappa(x)$ or $\|x\|$, hence giving $\kappa^\circ(x) = \|x\|_*$. It suffices to show that w lies in the same orthant as x. Assuming the contrary, it is easily seen that $x^T w$ is increased by flipping the sign of the violating component while $\kappa(w)$ remains the same, giving a contradiction.

For the second case, $x \in \mathcal{C}^\circ$, it can be seen that $w^T x \leq 0$ for all $w \in \mathcal{C}$, and we therefore have $\kappa^\circ(x) = 0$. The results then follows from the fact that $\mathcal{P}(x) = 0$, and therefore that $\kappa^\circ(x) = \|0\|_* = 0$.

For third case we define $u = \mathcal{P}(x)$, and $v = x - u$, where $u^T v = 0$ due to projection. Now let w give the supremum in $\kappa^\circ(u)$. We know from the first case



(a) component 1 (b) component 2

(c) component 5 (d) component 6

(e) mixture (f) recovery results

▶ Figure 6.8: Mass spectra of (a)–(d) four of the twelve components in the mixture (e), along with (f) the actual and recovered relative abundance. The dictionary contains the mass spectra of 438 different molecules.

that w must be in the orthant as u, i.e., share the same sign pattern, possibly with more zeroes. As a consequence we then have $w^T v = 0$, and

$$\kappa^\circ(x) = x^T w/\kappa(w) = (u^T w + v^T w)/\kappa(w) = u^T w/\kappa(w) = \kappa^\circ(u) = \|u\|_*.$$

To see that the first equality holds, note that for all feasible w such that $w^T \neq 0$, we have $w^T v < 0$. This can only lower the value of the polar and such w can therefore not be optimal. □

The sign invariance of the norm is clearly satisfied for all ℓ_p norms. Interestingly, it also applies to all nested norms using such norms. In this case the outer norms do not need to satisfy the condition of invariance under sign changes. Since there are no restrictions on index sets $\mathcal{I}_n$ and $\mathcal{I}_p$, aside from their disjointness, it is possible, for example, to impose independent sign restrictions on the real or imaginary parts of complex numbers. With only minor modifications to the proof it can be shown that Theorem 6.6 holds for all closed convex cones $\mathcal{C}$ when $\kappa(x) = \|x\|_2$ for all $x \in \mathcal{C}$.

6.3.3 Projection

The projection for the sign-restricted formulation consists of setting to zero all components of x that violate the restriction and projecting the remainder onto the ball induced by the underlying norm.

6.4 Low-rank matrix recovery and completion

The matrix completion problem described in Section 1.4.3 is a special case of the low-rank matrix recovery problem:

$$\underset{X \in \mathbb{R}^{m \times n}}{\text{minimize}} \quad \text{rank}(X) \quad \text{subject to} \quad \|\mathcal{A}(X) - b\|_2 \leq \sigma,$$

where $\mathcal{A}(X)$ is a linear operator mapping X to a vector. This problem is again intractable, and a convex relaxation of this formulation, based on the nuclear norm, was suggested by Fazel [75] and Recht et al. [126]:

$$\underset{X \in \mathbb{R}^{m \times n}}{\text{minimize}} \quad \|X\|_* \quad \text{subject to} \quad \|\mathcal{A}(X) - b\|_2 \leq \sigma. \tag{6.15}$$

Conditions for exact recovery of X_0 from $b := \mathcal{A}(X_0)$ using (6.15) with $\sigma = 0$ were recently studied by Recht et al. [126], who leveraged the restricted isometry technique developed for exact vector recovery using ℓ_1. They derived necessary and sufficient conditions for recovery, and subsequently used them to compute recovery bounds for linear operators $\mathcal{A}$ whose matrix representation has independent random Gaussian entries.

Choosing $\mathcal{A}$ in (6.15) to be an operator that restricts elements of a matrix to the set Ω gives the nuclear-norm formulation for noisy matrix completion:

$$\underset{X}{\text{minimize}} \quad \|X\|_* \quad \text{subject to} \quad \|X_\Omega - B_\Omega\|_2 \leq \sigma.$$

With $\sigma = 0$ this reduces to the exact matrix completion formulation (1.13). Conditions for exact recovery using the latter formulation were studied by Candès and Recht [29] and Candès and Tao [36]. An extension of this work to the noisy case was given by Candès and Plan [28].

6.4.1 Application – Distance matrix completion

As an illustration of nuclear norm minimization for matrix completion consider the following scenario (see also [29, 126]). Let $X = [x_1, \ldots, x_n] \in \mathbb{R}^{d\times n}$ denote the coordinates of n sensors in $\mathbb{R}^d$. Given the squared pairwise distance $D_{i,j} := (x_i - x_j)^T(x_i - x_j)$ for a limited number of pairs $(i, j) \in \Omega$, we want to determine the distance between any pair of sensors. That is, we want to find the Euclidean (squared) distance matrix D given by

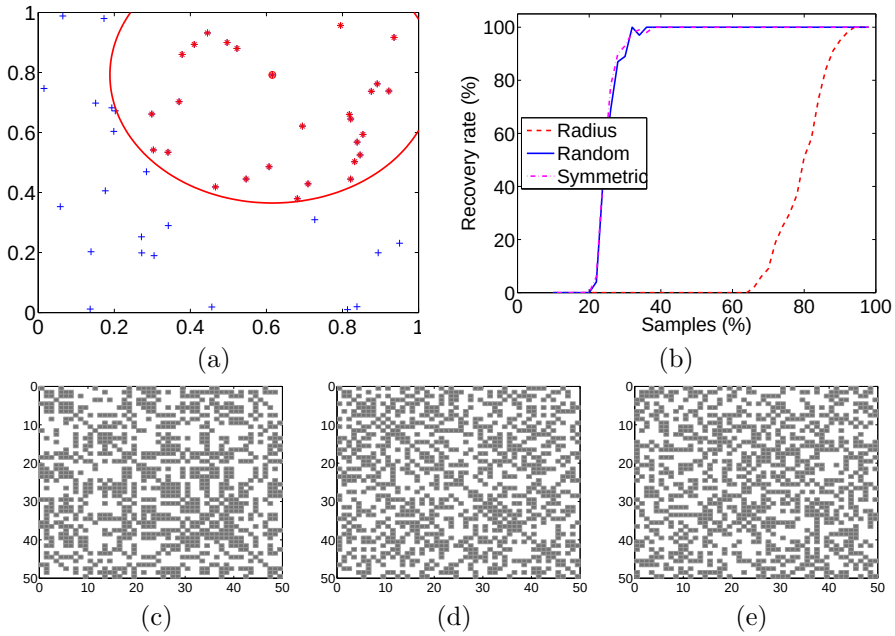
$$D = es^T + se^T - 2X^TX, \tag{6.16}$$

where e denotes the vector of all ones, and each $s_i := \|x_i\|_2$. Because D is a low-rank matrix (it can be seen from (6.16) that D has rank at most $d + 2$), we can try to apply (1.13) to recover D from D_Ω.

We consider three ways of choosing Ω. In the first one we restrict the known distances to those that are below a certain threshold: $\Omega_1 := \{(i, j) \mid D_{i,j} \leq r^2\}$ (see Figure 6.9(b)). In the second method we fix the cardinality of Ω_2 and choose its elements uniformly at random. Finally, for Ω_3 we also fix the cardinality, but make sure that all diagonal entries are in the set, and that all other elements occur in symmetric pairs. These assumptions reflect the fact that the distance from a sensor to itself is zero and that the distance between two sensors is independent of their ordering ($D_{i,j} = D_{j,i}$).

For the empirical recovery rate of (1.13) as a function of the cardinality of the three different Ω sets, we proceed as follows. First we generate 100 random 3×50 instances of X with entries uniformly sampled from $[0, 1]$. For each of these matrices we compute D using (6.16) and generate index sets Ω_1, Ω_2, and Ω_3 of varying cardinality. We then solve (1.13) for each of these combinations and record the number of exact reconstructions for different cardinalities and index set type. The resulting recovery rates are plotted in Figure 6.9(b).

The recovery curves reveal that recovery from random samples is far superior to recovery from distance-based samples; the latter essentially requires most distance pairs to be known in order to successfully complete the matrix. This is very likely due to the fact that the latter is much more structured and may provide little distance information for isolated sensors such as sensors located in a corner. In Figure 6.9(c–e) we plot instances of the sampling pattern obtained from the three methods. For the distance-based pattern in plot (c) it can be seen that little information is known for sensors 21, 43, and 46. Random sampling alleviates this problem and consequently gives much better recovery results. In a sense, the non-symmetric random sampling gives the most information because we could infer the distance for any pair of sensors whenever information is known for its symmetric counterpart. However, we did not perform any such



▶ Figure 6.9: Euclidean distance matrix completion: (a) example of nodes observed from central node in circle; (b) recovery probabilities for random and distance-based sampling; and entries observed with (c) distance-based sampling, (d) symmetric random sampling, and (e) uniformly random sampling.

preprocessing step, nor did we alter formulation (1.13) to take into account this symmetry. Nevertheless, for random sampling it can be seen that the use of (1.13) does successfully solve the matrix completion problem in many instances.

6.4.2 Polar function

The polar or dual of the nuclear norm $\|X\|_*$ is well-known to be given by the operator norm $\|X\|_2$, which corresponds to the largest singular value of X.

6.4.3 Projection

From existing results it follows that projection onto the nuclear norm ball reduces to computing the singular value decomposition (SVD) followed by projection of the singular values onto the ℓ_1-norm ball, which we discussed in Section 6.1.2:

Theorem 6.7. *Let C be an $m \times n$ matrix with singular value decomposition $C = U_c D_c V_c^T$, and U and V orthonormal and D_c an $m \times n$ matrix with diagonal d_c. Then the solution X^* to the nuclear-norm restricted problem*

$$\underset{X}{\text{minimize}} \quad \|C - X\|_F^2 \quad \text{subject to} \quad \|X\|_* \leq \tau, \tag{6.17}$$

is given by $U D_x V$ where D_x is an $m \times n$ matrix with diagonal entries d_x:

$$d_x := \arg\min_d \quad \|d_c - d\|_2 \quad \text{subject to} \quad \|d\|_1 \leq \tau. \tag{6.18}$$

Proof. This follows directly from [23, Theorem 2.1] or [106, Theorem 3]. □

Chapter 7

Experiments

Having established both the theoretical foundation of the solver framework, and the necessary ingredients for its application to a number of specific formulations, we can now consider its performance. For our empirical experiments we consider three implementations of the framework: the main spectrally projected-gradient implementation SPGL1, as well as two implementations with tighter tolerances for each subproblem solve, based on the spectral projected gradient (SPG-TIGHT) and projected quasi-Newton (PQN) methods respectively. Each solver accepts functions for the evaluation of $\kappa(x)$, $\kappa^\circ(x)$ and the projection onto the feasible set, thus simplifying the addition of new formulations that fit the framework.

The majority of solvers available for the problems under consideration work with a penalized version of the formulation. To include them in our comparison, we follow the approach suggested by Becker et al. [7]. For brevity we only discuss their approach in the context of basis pursuit denoise. Nevertheless, with only minor changes it applies equally to all other formulations discussed in this chapter. Becker et al. generate equivalent formulation in σ and λ (for (BP_σ) and (QP_λ) respectively) by first solving (BP_σ) approximately, for some $\bar{\sigma}$. Based on this they define $\lambda := \|A^T r\|_\infty$, with misfit $r := b - Ax$, and use FISTA [6] to solve (QP_λ) to high accuracy. This gives a corresponding $\sigma := \|b - Ax^*\|_2$, as well as an accurate solution x^* to both formulations.

Because at the time of writing there was no publicly available code for FISTA, we implemented our own version. This is relatively straightforward and we here discuss only the stopping criterion, which again applies to all other formulations with minor changes. Let r denote the misfit $b - Ax$ for the current iterate x. We define the relative accuracy as

$$\mu = \max\left\{ \frac{|\lambda - \|A^T r\|_\infty|}{\max\{1, \lambda\}},\ \frac{|\|x\|_1 + (r^T r - r^T b)/\|A^T r\|_\infty|}{\max\{1, \|x\|_1\}} \right\}.$$

The first term is the relative difference between λ and the estimation $\|A^T r\|_\infty$ obtained for (BP_σ) with the current misfit $\hat{\sigma} := \|r\|_2$. The second part is the relative primal-dual gap of the (BP_σ) formulation, again with misfit $\hat{\sigma}$. This gap can be derived using the techniques described in Section 5.2.1. Throughout this chapter, we pre-solve the test problems to relative accuracy $\mu \leq 1e\text{–}10$, unless otherwise stated.

All run times reported in this chapter are as obtained on a single-core 2GHz AMD machine with 2Gb of RAM, running Matlab 7.8.

7.1 Basis pursuit

In Section 6.1 we derived the polar function for both the weighted and unweighted ℓ_1-norm, and presented an efficient routine for projection onto the feasible set. For our experiments we focus on the unweighted formulations, due to the lack of solvers for the weighted case.

We start with the application of the framework to the basis pursuit formulation ($\sigma = 0$), which precludes the use of solvers for the regularized version (QP_λ). Solvers that do apply are homotopy, implementation courtesy of Michael Friedlander and Michael Saunders, and the Bregman algorithm [154], as implemented by Wotao Yin. NESTA [7] can be used whenever the rows in matrix A are orthonormal. All solvers are called with their default parameters, or, when missing, with the parameters suggested in the accompanying paper. We slightly modified the Bregman algorithm to allow up to five updates of its penalty parameter μ, whenever the subproblem solver failed.

For the first set of experiments we apply the different solvers to a selection of SPARCO test problems (for details, see Table 8.2). The results of the experiments are summarized in Table 7.1. We evaluate the accuracy in terms of the ℓ_1-norm of the solution, the ℓ_2-norm of the residual (which should be zero), and the ℓ_∞-norm of the difference between the solution and the original x_0 used in generating the problem. In general, comparing against x_0 may be meaningless, as it need not be the solution to the basis pursuit problem. However, based on the ℓ_1-norm of the solution and x_0, we conclude that they do coincide for this set of problems.

Starting with the `sgnspike` problem we see that all solvers successfully recover the original x_0. The fastest solvers are SPGL1 and homotopy, closely followed by SPG-TIGHT and PQN. The most accurate, but also the slowest is CVX/SDPT3. This is even more pronounced on `zsgnspike`, the complex version of the problem, where the root-finding approach is substantially faster.

The `dcthdr` test problem seems to be solved inaccurate, but this is mostly due to the fact that we report absolute deviations from x_0, which for `dcthdr` has an ℓ_1-norm of the order 4.3e+4. NESTA has a very small residual but solves the problem somewhat less accurate than the other methods. Homotopy and Bregman are the most accurate here, but at the expense of run time. The root-finding algorithms do very well in terms of time. Finally, we omit CVX/SDPT3, because it took too long to solve.

There is little we can say about the large complex `angiogram` problem because the root-finding implementations are the only ones that can be practically applied. However, the number of matrix-vector products and run time for SPGL1 are quite low, considering the number of variables. SPGL1 is also the fastest solver for `jitter`, but trails in accuracy to CVX/SDPT3 and Bregman. The relative inaccuracy with which SPGL1 solves it subproblems takes its toll for the difficult `blkheavi` problem. All solvers but CVX/SDPT3 have considerable trouble solving this problem, despite the fact that its solution is unique (in fact, the matrix A used in this problem has a simple closed-form inverse). SPGL1 is seen to slightly overshoot the target τ^*, leading to an inaccurate solution. This

	Solver	Time (s)	Aprod	$\|x\|_1$	$\|r\|_2$	$\|x - x_0\|_\infty$
sgnspike	CVX/SDPT3	4.1e+2	n.a.	2.0000e+1	7.5e−10	1.5e−9
	SPGL1	5.7e−1	61	2.0000e+1	7.3e−5	7.2e−5
	SPG-TIGHT	9.6e−1	114	2.0000e+1	5.4e−6	4.4e−6
	PQN	1.1e+0	82	2.0000e+1	1.3e−5	2.1e−5
	Homotopy	4.6e−1	49	2.0000e+1	9.4e−6	5.6e−6
	Bregman	3.9e+0	301	2.0000e+1	6.6e−6	6.9e−6
zsgnspike	CVX/SDPT3	1.2e+3	n.a.	2.8284e+1	1.9e−10	1.3e−9
	SPGL1	1.0e+0	59	2.8284e+1	9.5e−5	1.1e−4
	SPG-TIGHT	1.4e+0	110	2.8284e+1	1.1e−5	8.4e−6
	PQN	1.9e+1	324	2.8284e+1	4.7e−5	5.8e−5
dcthdr	NESTA	9.6e+0	824	4.3765e+4	8.5e−13	2.6e−1
	SPGL1	4.1e+0	304	4.3744e+4	9.8e−5	1.0e−4
	SPG-TIGHT	3.4e+0	276	4.3749e+4	7.4e−13	3.9e−2
	PQN	6.2e+0	188	4.3748e+4	3.3e−2	7.9e−2
	Homotopy	1.9e+1	735	4.3744e+4	3.7e−5	9.2e−6
	Bregman	2.9e+1	1334	4.3744e+4	2.2e−6	1.0e−6
angiogr.	SPGL1	1.6e+0	101	3.8433e+2	6.1e−5	2.0e−5
	SPG-TIGHT	2.4e+0	202	3.8433e+2	7.5e−5	9.9e−6
	PQN	5.8e+1	328	3.8437e+2	3.5e−15	1.6e−4
jitter	CVX/SDPT3	1.7e+1	n.a.	1.7412e+0	1.6e−12	2.6e−10
	NESTA	2.5e+0	620	1.7413e+0	1.3e−16	5.0e−6
	SPGL1	2.0e−1	39	1.7409e+0	8.0e−5	1.6e−4
	SPG-TIGHT	3.5e−1	74	1.7411e+0	3.7e−5	6.1e−5
	PQN	4.5e−1	64	1.7412e+0	3.1e−5	4.7e−5
	Bregman	1.4e+0	163	1.7412e+0	2.2e−8	3.6e−8
blkheavi	CVX/SDPT3	4.7e−1	n.a.	4.1000e+1	1.6e−9	1.0e−9
	SPGL1	1.3e+1	3789	4.2102e+1	2.9e−1	1.5e−1
	SPG-TIGHT	1.4e+2	41037	4.1000e+1	3.0e−5	1.9e−5
	PQN	1.2e+2	10161	4.1000e+1	5.1e−5	3.3e−5
	Bregman	1.6e+1	2054	1.7691e+1	8.2e+0	3.7e+0

▶ Table 7.1: Solver performance on a selection of SPARCO test problems.

holds even more so for Heaviside with normalized columns (not shown).

The poor performance on the `blkheavi` problem led us to consider problems where the columns of A are increasingly similar. This was achieved by taking convex combinations $A := (1 - \alpha)G + \alpha E$, where G is a fixed 600×1800 Gaussian matrix, and E is a matrix of all ones. The 80-sparse coefficient vector x_0 was randomly chosen and fixed for all A. The results obtained for different convex combinations of G and E are shown in Table 7.2. Despite the increasing similarity of the columns, basis pursuit can recovery x_0 in all cases considered, as indicated by the ℓ_1-norm of the solutions. Among the different methods, homotopy is by far the best, except when $A = G$, in which case it matches SPGL1. Both SPGL1 and the Bregman algorithm fail on the problems in which the E component in A dominates; the former by overestimating τ, the latter by underestimating it. The tighter versions SPG-TIGHT and PQN do not have this problem, but do require many more matrix-vector products to reach a solution. Even the run time of CVX/SDPT3, commonly less sensitive to A, goes up as the presence of E in A becomes more prominent.

To conclude the basis pursuit experiments, we look at the scalability of the different methods, and their sensitivity to the sparsity level of x_0. For this we let A be an implicit representation of a randomly restricted DCT matrix of size $2^{n-2} \times 2^n$, and choose x_0 to be 2^{n-5} respectively 2^{n-4} sparse, for different scaling parameters n. We summarize the number of matrix-vector products and run time in Table 7.3. Because CVX does not work with implicit matrices we form an explicit representation prior to calling the solver. In order to save time, we limit the problem size solved with CVX/SDPT3, and likewise for the homotopy method. The remaining methods can be seen to scale very well to large problem sizes. For the sparsity level of 2^{n-5}, the number of matrix-vector products remains nearly constant throughout for NESTA and SPGL1. For the Bregman algorithm, this number fluctuates heavily, while it steadily increases for homotopy. The accuracy of the different methods is comparable throughout, while in terms of run time, SPGL1 and SPG-TIGHT clearly excel.

Decreasing the sparsity to 2^{n-4} affects the performance of the solvers, as shown by the increased number of matrix-vector products required. For NESTA the number of matrix-vector products remain nearly the same. Closer inspection of the data shows that the solutions obtained by NESTA, while closely fitting $Ax = b$, typically have an overly large ℓ_1-norm. The Bregman algorithm, by contrast, tends to give solutions with an ℓ_1-norm that is too small, which necessarily leads to larger misfits. Interestingly, the more accurate subproblem solves in SPG-TIGHT benefit its overall performance, making it even faster than SPGL1. Finally, note that the sparsity level strongly affects the performance of homotopy, which now requires substantially more time to reach the solution.

7.2 Basis pursuit denoise

We start the performance evaluation for basis pursuit denoise on four of the most challenging problems in the SPARCO test set: `srcsep`, `seismic`, `phantom2`, and

	Solver	Time (s)	Aprod	$\|x\|_1$	$\|r\|_2$	$\|x - x_0\|_\infty$
0%	CVX/SDPT3	3.8e+2	n.a.	6.417e+1	1.5e−5	3.7e−7
	SPGL1	1.6e+0	307	6.417e+1	9.4e−5	2.4e−6
	SPG-TIGHT	3.0e+0	680	6.417e+1	2.7e−6	9.0e−8
	PQN	4.4e+0	304	6.417e+1	3.6e−6	6.8e−8
	Homotopy	1.5e+0	224	6.417e+1	3.9e−7	3.6e−9
	Bregman	7.1e+0	1267	6.417e+1	2.0e−4	2.5e−6
20%	CVX/SDPT3	3.8e+2	n.a.	6.417e+1	1.8e−5	2.8e−7
	SPGL1	7.0e+0	1535	6.417e+1	7.5e−5	1.6e−6
	SPG-TIGHT	1.2e+1	2896	6.417e+1	3.4e−6	1.4e−7
	PQN	2.9e+1	2408	6.417e+1	5.7e−6	1.3e−7
	Homotopy	1.3e+0	229	6.417e+1	4.9e−7	5.5e−9
	Bregman	8.8e+0	1545	6.417e+1	3.9e−4	7.3e−6
40%	CVX/SDPT3	4.7e+2	n.a.	6.417e+1	3.0e−6	5.9e−8
	SPGL1	2.9e+1	6385	6.439e+1	9.7e−5	7.8e−3
	SPG-TIGHT	4.6e+1	10529	6.417e+1	4.9e−6	2.6e−7
	PQN	1.6e+2	11224	6.417e+1	1.8e−5	8.0e−7
	Homotopy	1.5e+0	253	6.417e+1	6.5e−7	1.0e−8
	Bregman	2.3e+1	3949	6.400e+1	2.6e−1	4.1e−3
60%	CVX/SDPT3	4.9e+2	n.a.	6.417e+1	2.4e−6	4.9e−8
	SPGL1	3.0e+1	6810	6.762e+1	9.6e−5	1.1e−1
	SPG-TIGHT	1.9e+2	44236	6.417e+1	1.3e−6	3.0e−8
	PQN	4.8e+2	20162	6.417e+1	4.1e−5	2.8e−6
	Homotopy	1.5e+0	269	6.417e+1	9.7e−7	2.3e−8
	Bregman	1.2e+1	2052	5.388e+1	1.2e+1	3.8e−1
80%	CVX/SDPT3	5.5e+2	n.a.	6.417e+1	1.4e−6	8.1e−8
	SPGL1	7.1e+0	1597	1.184e+2	6.3e−5	1.2e+0
	SPG-TIGHT	2.8e+2	64201	6.417e+1	1.5e−3	6.9e−5
	PQN	1.5e+3	46065	6.417e+1	1.5e−3	7.1e−5
	Homotopy	2.0e+0	333	6.417e+1	1.9e−6	9.1e−8
	Bregman	1.2e+1	2052	1.025e+1	3.9e+1	2.3e+0

▶ Table 7.2: Convex combination of random 600×1800 Gaussian matrix G, and a matrix $E = ee^T$ of all ones. Percentage indicates relative weight of E.

	Sparsity $= 2^{n-5}$					
Solver	$n = 10$	$n = 11$	$n = 12$	$n = 13$	$n = 16$	$n = 19$
CVX/SDPT3	2.7e+1 n.a.	4.0e+2 n.a.	— —	— —	— —	— —
NESTA	1.8e+0 777	2.2e+0 813	3.1e+0 837	4.3e+0 801	3.1e+1 825	2.7e+2 855
SPGL1	4.7e–1 923	6.8e–1 997	7.8e–1 997	1.3e+0 977	9.3e+0 994	9.6e+1 1069
SPG-TIGHT	7.8e–1 1350	1.1e+0 1515	1.2e+0 1460	1.9e+0 1453	1.4e+1 1455	1.2e+2 1529
PQN	1.2e+0 1087	1.9e+0 1187	2.7e+0 1179	4.7e+0 1155	4.1e+1 1162	3.0e+2 1219
Homotopy	2.5e–1 1425	6.7e–1 1672	2.1e+0 1755	9.4e+0 2037	— —	— —
Bregman	2.6e+0 529	7.1e+0 1287	5.1e+0 773	7.1e+0 825	9.3e+1 1844	2.7e+2 702

	Sparsity $= 2^{n-4}$					
Solver	$n = 10$	$n = 11$	$n = 12$	$n = 13$	$n = 16$	$n = 19$
CVX/SDPT3	5.5e+1 n.a.	4.7e+2 n.a.	— —	— —	— —	— —
NESTA	1.6e+0 681	2.1e+0 753	3.0e+0 825	4.0e+0 741	3.1e+1 801	2.6e+2 843
SPGL1	8.2e+0 3469	8.2e+0 3096	9.0e+0 2729	1.8e+1 3248	1.1e+2 2961	1.5e+3 4092
SPG-TIGHT	5.9e+0 6096	6.0e+0 5425	8.3e+0 5110	1.1e+1 5460	9.6e+1 5295	7.7e+2 6251
PQN	6.2e+0 3935	8.3e+0 3556	1.2e+1 3149	2.4e+1 3658	2.0e+2 3299	1.5e+3 4374
Homotopy	1.7e+0 6471	6.0e+0 6188	3.6e+1 6721	2.4e+2 8461	— —	— —
Bregman	9.5e+0 1899	4.9e+1 8979	2.6e+1 3253	3.6e+1 3523	1.9e+2 2750	1.7e+3 9212

▶ Table 7.3: Basis pursuit solver performance on randomly restricted $2^{n-2} \times 2^n$ DCT matrices. The two rows for each solver respectively denote run time in seconds, and number of matrix-vector products.

Problem	μ ($\|x^*\|_1$)		
	$\sigma = 0.1\|b\|_2$	$\sigma = 0.01\|b\|_2$	$\sigma = 0.001\|b\|_2$
srcsep1	1e–10 (8.3347e+2)	1e–10 (1.0535e+3)	1e–10 (1.0873e+3)
seismic	1e–10 (1.5789e+4)	1e–10 (2.0164e+4)	4.8e–4 (2.0771e+4)
phantom2	1e–10 (7.0455e+3)	1e–10 (8.0027e+3)	1e–10 (8.1185e+3)
finger	1e–10 (4.3956e+3)	1e–10 (5.3703e+3)	1e–10 (5.4726e+3)

▶ Table 7.4: Relative accuracy of FISTA solutions for a subset of the SPARCO test problems. The number in parentheses gives the ℓ_1-norm of the solution.

`finger`. These problems are generated based on signals, rather than on sparse coefficients, and no strictly sparse representation may exist. We therefore allow a solution to satisfy $\|Ax - b\| \leq \sigma$, with a misfit σ proportional to a fixed fraction of the norm of b. Table 7.4 summarizes the pre-solve accuracy μ, along with the ℓ_1-norm of the solution for different values of σ. For the `seismic` problem we could not obtain a very accurate solution within a reasonable amount of time, and had to settle for $\mu = 4.8\text{e–}4$.

We compare our implementation of the root-finding algorithm against NESTA and homotopy. For the penalized formulation we use GPSR [78], FPC [90], and our implementation of FISTA. All solvers were called with default parameters, except for FISTA, which was called with $\mu = 1\text{e–}6$. The large size of the problems rendered the use of CVX/SDPT3 impracticable.

The first thing to note about the test results with $\sigma = 0.1\|b\|_2$, given in Table 7.5, is that all solvers had considerable difficulty in reaching even modestly accurate solutions. FISTA did very well, obtaining higher accuracy with competitive run time. Recall, however, that FISTA solves the penalized version of basis pursuit denoise, and therefore required the Lagrange multiplier λ to be specified. SPGL1 is the fastest solver in all cases, but tends to over shoot the target τ value on these problems (compare with Table 7.4). This affects the accuracy of the solution, which, nevertheless, is comparable to the solution obtained by FPC. The quality of the solutions improves with increasing accuracy of the subproblem solves, as evident from the results obtained using SPG-TIGHT, and PQN. The latter two solvers do require substantially more time than SPGL1, but still compare favorably against GPSR.

Decreasing the value of σ generally increases the amount of time spent on solving the problem. This is not so much due to the number of root finding iterations, but more as a consequence of the subproblems getting increasingly hard to solve accurately; the same holds true for the penalized formulation with decreasing λ.

The results obtained for $\sigma = 0.01\|b\|_2$ give a similar picture as those for $\sigma = 0.1\|b\|_2$, except that the run time is quite a bit higher. As evident from Table 7.4, the `seismic` problem could not be solved to the desired level of $\mu = 1\text{e–}6$, even after running FISTA for over a day. Therefore, instead of reporting exceedingly large run times, with otherwise similar solver behavior, we decided to limit the run time of each solver to half an hour, and see how much could

	Solver	Time (s)	Aprod	$\|x\|_1$	$\|x - x^*\|_\infty$
srcsep1	SPGL1	5.2e+1	152	8.3415e+2	4.8e−1
	SPG-TIGHT	1.2e+3	5341	8.3346e+2	1.9e−3
	PQN	7.4e+2	1158	8.3346e+2	5.1e−3
	GPSR	1.3e+3	9705	8.3347e+2	1.2e−2
	FPC	1.2e+2	10154	8.3335e+2	3.4e−1
	FISTA	3.7e+3	20916	8.3347e+2	9.7e−6
seismic	SPGL1	1.8e+2	325	1.5831e+4	3.0e+1
	SPG-TIGHT	8.4e+3	23381	1.5789e+4	2.1e+1
	PQN	1.8e+4	6401	1.5789e+4	2.2e+1
	GPSR	5.5e+3	34307	1.5789e+4	1.7e+1
	FPC	4.1e+2	35216	1.5783e+4	1.9e+1
	FISTA	9.3e+3	55230	1.5789e+4	4.7e−2
phantom2	SPGL1	5.3e+0	87233	7.0482e+3	2.4e−1
	SPG-TIGHT	2.4e+2	94345	7.0455e+3	3.8e−3
	PQN	5.5e+3	90241	7.0456e+3	7.8e−2
	FISTA	4.4e+2	102834	7.0455e+3	8.2e−5
finger	SPGL1	2.6e+1	191	4.4114e+3	4.3e+0
	SPG-TIGHT	2.1e+3	26794	4.3956e+3	3.7e−1
	PQN	4.4e+3	9375	4.3956e+3	6.3e−1
	GPSR	2.5e+3	46810	4.3957e+3	1.9e+0
	FPC	8.5e+1	47823	4.3953e+3	4.1e+0
	FISTA	2.2e+3	67837	4.3956e+3	4.2e−3

▶ Table 7.5: Solver performance for the basis pursuit formulation on a selection of SPARCO test problems, with σ set to $0.1\|b\|_2$.

	Solver	Time (s)	Aprod	$\|x\|_1$	$\|x - x^*\|_\infty$
srcsep1	SPGL1	4.8e+2	1501	1.0891e+3	5.7e−1
	SPG-TIGHT	1.8e+3	9660	1.0560e+3	2.0e−1
	PQN	1.8e+3	3369	1.0838e+3	1.4e−1
	GPSR	1.8e+3	15838	1.0895e+3	5.5e−1
	FPC	2.7e+2	16777	1.0899e+3	5.7e−1
	FISTA	1.8e+3	22087	1.0873e+3	6.7e−2
seismic	SPGL1	5.6e+2	1016	2.0877e+4	2.8e+1
	SPG-TIGHT	1.8e+3	5171	1.3890e+4	3.0e+1
	PQN	1.8e+3	1540	5.2461e+3	3.8e+1
	GPSR	5.6e+2	6341	2.0912e+4	2.8e+1
	FPC	2.7e+2	6942	2.0970e+4	2.8e+1
	FISTA	1.8e+3	10808	2.0799e+4	2.8e+1
phantom2	SPGL1	2.1e+1	59683	8.1232e+3	2.3e−1
	SPG-TIGHT	8.4e+2	76207	8.1186e+3	7.4e−2
	PQN	1.8e+3	60823	7.9624e+3	2.3e−1
	FISTA	9.9e+2	96222	8.1185e+3	2.9e−3
finger	SPGL1	1.0e+2	793	5.4877e+3	3.8e+0
	SPG-TIGHT	1.8e+3	16955	5.0560e+3	1.9e+0
	PQN	1.8e+3	1997	2.7825e+3	3.3e+0
	GPSR	1.3e+2	18047	5.5010e+3	3.9e+0
	FPC	4.1e+1	18528	5.5212e+3	3.9e+0
	FISTA	1.8e+3	34836	5.4726e+3	2.1e−1

▶ Table 7.6: Solver performance for basis pursuit denoise on a selection of SPARCO test problems, with σ set to $0.001\|b\|_2$, and run time limited to 30 minutes.

be done within this time. The results for this experiment, with the smallest of the three misfits ($\sigma = 0.001\|b\|_2$), are shown in Table 7.6. Here, the tighter tolerance on subproblem solves of SPG-TIGHT and PQN indirectly limits the number of root finding steps that can be taken. This is apparent, for example, in `srcsep1` and `seismic`, where the target ℓ_1-norm is not quite reached. The less stringent stopping criteria for SPGL1, on the other hand, still cause the target to be exceeded. It is clear that for limited-time runs some tuning is needed to balance between these two options.

As a final set of test problems, we apply basis pursuit denoise to the compressed sensing scenario where a 1400×4096 partial DCT matrix is used to sample a compressible signal. We created four different problems, all using the same A, but with different compressibility rates and misfit values. For `compress1a` we set x_0 to a random permutation of a vector with entries decaying exponentially from 10 to 10e−3, and chose $\sigma = 0.01\|x\|_1$. For the remaining three problems `compress1b`–`compress1d` we fixed x_0 with entries ranging from 10 to 10e−4, and chose σ as $0.01\|x_0\|_1$, $0.001\|x_0\|_1$, and $0.0001\|x_0\|_1$ respectively. In this setting, in addition to the other solvers, both homotopy and NESTA can

be applied, giving the results are shown in Table 7.7.

The first thing to notice for this problem is the rapid deterioration of the performance of homotopy for smaller values of σ. Unlike in previous experiments with exact sparse solutions, we here encounter a problem that requires a large number of homotopy steps. In all fairness, however, it should be noted that homotopy does give extremely accurate solutions. The others solvers behave mostly as before with SPGL1 giving a fast approximation, and SPG-TIGHT giving accurate solutions in very competitive time.

7.3 Joint sparsity

As mentioned in the introduction, we use FISTA to obtain for each test problem an accurate pair of (σ, λ) values, as well as an accurate solution for both formulations. In order for FISTA to work, we need to provide it with the proximal function for $g(x) := \sum_i \|x_{\sigma_i}\|_2$:

$$\text{prox}_\gamma(g)(x) := \underset{u}{\text{argmin}} \left\{ g(u) - \frac{1}{2\gamma}\|u - x\|_2^2 \right\} \tag{7.1}$$

We can conveniently express this in terms of a general signum function on vectors, defined as

$$\text{gsgn}(x) := \begin{cases} x/\|x\| & \|x\| > 0 \\ \{z \mid \|z\| \leq 1\} & \text{otherwise.} \end{cases}$$

With this definition it can be shown that the proxy of $g(x)$, with $x_{\sigma_i} \in \mathbb{C}^{|\sigma_i|}$, is given by

$$[\text{prox}_\gamma(x)]_{\sigma_i} = \text{gsgn}(x_{\sigma_i}) \cdot \max\{\|x_{\sigma i}\|_2 - \gamma, 0\}.$$

When the cardinality of each group is one, i.e., $|\sigma_i| = 1$, this expression reduces to the proxy function of $g(x) = \|x\|_1$, given by the standard soft-thresholding operator.

The pre-solve procedure was applied to the five test problems given in Table 7.8. For the first four problems we drew A from the Gaussian ensemble, and created an X_0 with random support containing randomly normally distributed entries. The observed matrix $B := AX_0 + R$ was formed by adding a normalized noise term R with $\|R\|_F = \nu\|AX_0\|_F$, to the ideal observation AX_0. Because the norm of the noise term is typically not known exactly, we used an over, respectively under estimation for two of the problems. The fifth problem is the first iteration of the radio telescope example shown in Figure 6.5(a,e).

We compare our solver against FISTA, CVX/SDPT3, as well as SPARSA, developed by Wright et al. [153]. The performance of SPGL1, as seen from the results in Table 7.9, is excellent on all `mmv1` test problems. In particular, even with the more relaxed subproblem stopping criteria, it is able to pin-point the right value of τ, and obtain an accurate final solution. It does have some trouble with the `telescope` problem, however, which is likely due to the stronger coherence

	Solver	Time (s)	Aprod	$\|x\|_1$	$\frac{\|\|r\|_2-\sigma\|}{\sigma}$	$\|x-x_0\|_\infty$
compress1a	NESTA	2.5e+0	681	9.6343e+2	3.3e−13	1.3e+0
	SPGL1	1.1e−1	20	9.5683e+2	9.2e−6	4.5e−1
	SPG-TIGHT	7.4e−1	341	9.5652e+2	1.5e−9	1.0e−4
	PQN	2.1e+0	166	9.5652e+2	4.2e−9	4.5e−4
	Homotopy	1.7e+1	1380	9.5652e+2	3.6e−13	1.2e−9
	GPSR	3.0e−1	1450	9.5653e+2	1.5e−6	8.4e−4
	FPC	2.0e−1	1539	9.5652e+2	5.6e−7	7.3e−4
	FISTA	1.4e+0	1865	9.5652e+2	1.8e−8	1.5e−5
compress1b	NESTA	2.5e+0	657	1.0334e+3	4.8e−13	1.0e+0
	SPGL1	1.1e−1	20	1.0271e+3	8.1e−5	4.5e−1
	SPG-TIGHT	9.4e−1	387	1.0265e+3	4.3e−9	5.0e−5
	PQN	2.4e+0	176	1.0265e+3	1.2e−8	4.4e−4
	Homotopy	2.2e+1	1544	1.0265e+3	5.9e−13	1.3e−9
	GPSR	3.8e−1	1632	1.0265e+3	6.2e−6	7.8e−4
	FPC	3.2e−1	1749	1.0265e+3	6.4e−6	7.1e−4
	FISTA	1.7e+0	2153	1.0265e+3	1.4e−8	1.4e−5
compress1c	NESTA	2.7e+0	753	2.3953e+3	3.2e−11	2.4e+0
	SPGL1	6.4e−1	879	2.3674e+3	6.6e−5	7.6e−1
	SPG-TIGHT	1.0e+1	3779	2.3662e+3	8.1e−7	1.6e−2
	PQN	2.4e+1	1463	2.3662e+3	1.1e−5	4.6e−2
	Homotopy	1.8e+2	6781	2.3662e+3	5.1e−11	2.0e−8
	GPSR	7.5e+0	8448	2.3661e+3	2.2e−4	1.2e−1
	FPC	1.2e+0	8765	2.3660e+3	1.5e−3	3.4e−1
	FISTA	3.7e+1	17459	2.3662e+3	3.4e−7	1.2e−4
compress1d	NESTA	2.7e+0	729	2.5495e+3	2.3e−9	2.9e+0
	SPGL1	2.5e+0	1229	2.5161e+3	1.8e−4	1.1e+0
	SPG-TIGHT	4.1e+1	11946	2.5147e+3	7.4e−5	1.9e−1
	PQN	6.0e+1	2293	2.5148e+3	1.1e−3	3.7e−1
	Homotopy	2.3e+2	15337	2.5147e+3	1.4e−10	4.5e−8
	GPSR	2.2e+1	20196	2.5150e+3	3.0e−3	8.3e−1
	FPC	1.4e+0	20545	2.5164e+3	2.7e−2	1.3e+0
	FISTA	8.5e+1	40559	2.5147e+3	3.1e−7	1.1e−3

▶ Table 7.7: Solver performance for basis pursuit denoise on a set of problems with compressible x_0 and different levels of misfit.

Problem	A	Observations	Sparsity	ν	σ
mmv1a	60×100	5	12	0.01	$1.0\|R\|_F$
mmv1d	200×400	5	20	0.01	$1.2\|R\|_F$
mmv1e	200×400	10	20	0.02	$1.0\|R\|_F$
mmv1f	300×800	3	50	0.01	$0.9\|R\|_F$
telescope	80×109	2	8	—	3

▶ Table 7.8: Test problem settings for MMV experiments.

between the columns in A. CVX does very well on this problem, but does not overall scale very well with the number of measurement vectors (see for example `mmv1f`). SPARSA has a run time similar to SPG-TIGHT, but yields solutions that are not nearly as accurate. The most accurate results for these tests are obtained using FISTA. In terms of run time, FISTA is about three to four times slower than SPG-TIGHT, except on `telescope`, where the two are similar both in run time and accuracy.

7.4 Nonnegative basis pursuit

We apply the nonnegative basis pursuit denoise framework on a set of randomly restricted DCT operators with noise scaled to $\nu\|Ax_0\|_2$, and σ set close to $\|r\|_2$. In most cases we underestimate σ, which makes the problem harder to solve for SPGL1 and variants. The parameters for the different problems in the test set, including the noisy mass spectrometry setting described in Section 6.3.1, are given in Table 7.10.

For the experiments we again follow the pre-solve procedure explained in the introduction. The proximal function (7.1) required by FISTA, corresponding to nonnegative BPDN is a one-sided soft-thresholding operator, given by

$$\text{prox}_\gamma(x) = [x - \gamma]_+,$$

where $[v]_+ := \max\{v, 0\}$ denotes projection onto the nonnegative orthant.

Unfortunately, there are not many solvers that are specific for the nonnegative basis pursuit problem. Although it would have been possible to modify GPSR to forego variable splitting and work directly with the nonnegative part only, no such attempt was made. As a result, this leaves us with CVX for smaller problems, and FISTA for the penalized formulation.

From the run time and number of matrix-vector products reported in Table 7.11, it is apparent that both SPGL1 and FISTA require more effort to solve problems where the number of nonzero entries in x_0 is large compared to the length of b. The hardest amongst these problems is `nonnegn02`, which has very few measurements per nonzero entry in x_0. Comparing `nonnegn04` to `nonnegn06` shows, as expected, that an increased σ makes the problem easier to solve. Likewise, `nonnegn03` is solved much faster and more accurate than `nonnegn02` due to the larger number of measurements. The behavior of the different solvers is similar to earlier experiments, so we will not discuss them further. Finally, we note that CVX/SDPT3 did very well on the mass spectrometry problem.

7.5 Nuclear-norm minimization

A number of solvers for matrix completion based on nuclear norm minimization have recently been introduced: Ma et al. [106] propose FPCA, which combines approximate SVD with fixed-point iterations to solve the penalized formulation of (6.15), while Cai et al. [23] derive a singular value soft-thresholding algorithm,

	Solver	Time (s)	Aprod	$\|X\|_{1,2}$	$\frac{\lvert\|\|r\|_2-\sigma\rvert}{\sigma}$	$\|x-x_0\|_\infty$
mmv1a	CVX/SDPT3	2.9e+0	n.a.	1.4653e+1	3.9e−9	1.4e−6
	SPGL1	2.4e−1	345	1.4653e+1	3.3e−6	5.9e−5
	SPG-TIGHT	6.1e−1	1055	1.4653e+1	1.5e−5	5.2e−7
	PQN	1.3e+0	900	1.4653e+1	1.5e−5	4.8e−7
	SPARSA	5.5e−1	580	1.4693e+1	4.6e−2	1.2e−2
	FISTA	1.8e+0	3221	1.4653e+1	6.9e−8	4.0e−8
mmv1d	CVX/SDPT3	8.8e+1	n.a.	4.8226e+1	1.6e−9	8.9e−8
	SPGL1	3.1e−1	305	4.8226e+1	1.5e−6	7.6e−5
	SPG-TIGHT	1.0e+0	1185	4.8226e+1	2.3e−6	1.7e−7
	PQN	2.5e+0	995	4.8226e+1	2.3e−6	1.8e−7
	SPARSA	3.6e+0	950	4.8241e+1	9.8e−3	3.9e−3
	FISTA	3.2e+0	3891	4.8226e+1	3.7e−8	7.1e−8
mmv1e	CVX/SDPT3	3.5e+2	n.a.	5.8488e+1	8.6e−10	9.2e−6
	SPGL1	4.3e−1	530	5.8488e+1	2.4e−5	4.7e−4
	SPG-TIGHT	1.8e+0	2670	5.8488e+1	6.7e−5	3.5e−6
	PQN	4.6e+0	2270	5.8488e+1	6.7e−5	3.5e−6
	SPARSA	5.1e+0	1580	5.8480e+1	3.8e−3	4.1e−3
	FISTA	6.0e+0	8372	5.8488e+1	6.8e−9	4.7e−8
mmv1f	CVX/SDPT3	1.7e+2	n.a.	8.1691e+1	8.6e−10	1.6e−6
	SPGL1	6.0e−1	312	8.1691e+1	7.6e−5	7.9e−4
	SPG-TIGHT	2.4e+0	1392	8.1690e+1	7.2e−5	1.3e−6
	PQN	6.4e+0	1002	8.1690e+1	7.3e−5	1.3e−6
	SPARSA	1.2e+1	990	8.1690e+1	7.2e−5	2.7e−4
	FISTA	8.9e+0	5515	8.1691e+1	4.9e−8	5.0e−8
telescope	CVX/SDPT3	4.1e+0	n.a.	1.6909e+1	3.9e−10	2.8e−5
	SPGL1	7.1e−1	604	1.6917e+1	5.8e−5	2.4e−1
	SPG-TIGHT	1.6e+1	16418	1.6909e+1	3.5e−6	2.4e−6
	PQN	2.4e+2	106842	1.6909e+1	3.5e−6	8.1e−6
	SPARSA	8.7e−1	340	1.7146e+1	1.7e−1	6.8e−1
	FISTA	1.7e+1	19740	1.6909e+1	6.9e−7	2.3e−5

▶ Table 7.9: Solver performance on a set of noisy or approximately sparse MMV problems.

Problem	A	Sparsity	ν	$\sigma/\|r\|_2$	$\|x\|_1$
nonnegn01	100×256	10	0.01	1.02	6.1484e+0
nonnegn02	500×8192	200	0.001	0.9	1.5039e+2
nonnegn03	1500×8192	200	0.001	0.9	1.7499e+2
nonnegn04	500×8192	100	0.001	0.9	8.6661e+1
nonnegn05	500×8192	10	0.001	0.9	8.6205e+0
nonnegn06	500×8192	100	0.05	0.9	8.2346e+1
massspec	82×438	12	0.012	1.0	9.9505e−1

▶ Table 7.10: Test problem settings for nonnegative basis pursuit denoise experiments.

	Solver	Time (s)	Aprod	$\|x\|_1$	$\frac{\|\|r\|_2-\sigma\|}{\sigma}$	$\|x-x_0\|_\infty$
nonnegn01	CVX/SDPT3	1.3e+0	n.a.	6.1484e+0	2.1e−10	5.6e−8
	SPGL1	3.4e−1	71	6.1476e+0	5.6e−3	1.2e−4
	SPG-TIGHT	6.4e−1	153	6.1483e+0	4.2e−4	7.1e−6
	PQN	7.8e−1	120	6.1484e+0	1.3e−4	4.5e−6
	FISTA	1.4e+0	335	6.1484e+0	9.1e−9	3.8e−9
nonnegn02	SPGL1	8.9e+0	1052	1.5044e+2	2.9e−2	1.3e−1
	SPG-TIGHT	3.1e+1	4158	1.5041e+2	9.9e−3	1.0e−1
	PQN	1.1e+2	1350	1.5044e+2	7.8e−4	1.1e−1
	FISTA	1.5e+2	20011	1.5039e+2	9.0e−5	1.4e−3
nonnegn03	SPGL1	1.7e+0	190	1.7499e+2	1.0e−2	2.6e−5
	SPG-TIGHT	1.9e+0	246	1.7499e+2	1.2e−3	2.1e−5
	PQN	5.5e+0	190	1.7499e+2	1.1e−2	2.9e−4
	FISTA	1.7e+1	2283	1.7499e+2	2.9e−9	3.4e−9
nonnegn04	SPGL1	1.1e+1	1273	8.6662e+1	4.1e−2	9.7e−3
	SPG-TIGHT	1.7e+1	2287	8.6665e+1	1.7e−3	7.9e−3
	PQN	7.0e+1	900	8.6674e+1	5.2e−3	1.2e−2
	FISTA	1.5e+2	20011	8.6661e+1	5.2e−5	1.2e−5
nonnegn05	SPGL1	4.6e−1	51	8.6202e+0	1.7e−2	1.2e−4
	SPG-TIGHT	8.0e−1	101	8.6163e+0	2.2e−1	5.1e−4
	PQN	6.7e+0	470	8.6205e+0	1.3e−3	4.0e−5
	FISTA	1.3e+1	1789	8.6205e+0	1.0e−7	1.3e−9
nonnegn06	SPGL1	2.1e+0	241	8.2343e+1	7.9e−4	8.4e−3
	SPG-TIGHT	4.7e+0	624	8.2346e+1	7.0e−5	6.0e−4
	PQN	2.9e+1	676	8.2345e+1	1.9e−4	2.5e−3
	FISTA	8.3e+1	10963	8.2346e+1	1.8e−7	1.8e−6
massspec	CVX/SDPT3	1.1e+0	n.a.	9.9505e−1	2.2e−9	1.1e−8
	SPGL1	3.2e+0	2172	9.7853e−1	4.4e+0	3.8e−2
	SPG-TIGHT	3.8e+1	30070	9.9504e−1	2.4e−3	1.2e−5
	PQN	2.3e+2	30975	9.9581e−1	2.4e−4	2.4e−3
	FISTA	2.5e+1	20023	9.9506e−1	2.9e−3	4.2e−5

▶ Table 7.11: Solver performance on the set nonnegative basis pursuit denoise test problems.

Problem	M	Rank	Observed entries	(γ)
nucnorm(n)01	10×10	2	80%	0.05
nucnorm(n)02	50×50	4	60%	0.01
nucnorm(n)03	100×100	7	50%	0.02
nucnorm(n)04	100×100	7	30%	0.03
nucnorm(n)05	200×200	12	20%	0.01
nucnorm(n)06	200×200	2	20%	0.01

▶ Table 7.12: Test problem settings for matrix completion experiments.

called SVT, for solving a slight relaxation to the exact matrix completion problem (1.13). Two more solvers were proposed by Toh and Yun [143], and Liu et al. [103], but their implementation is not yet publicly available.

We start our experiments for the exact matrix completion problem (1.13). Both FPCA and SVT approximate the solution to this problem by choosing a sufficiently small penalty parameter in their respective formulations. The same, in a way, holds true for SPGL1, which finds a τ that is sufficiently close to the ℓ_1-norm of the true solution. From the results in Table 7.13 it can be seen that SVT reaches its default maximum number of 500 iterations on problems `nucnorm02`, `nucnorm04`, and `nucnorm05`. Despite the fact that SVT takes fewer singular value decompositions, it is still slower and less accurate than FPCA. The reason for this difference seems to lie predominantly in the way the SVDs are computed. We should note here that SVT is designed for large-scale problems, and that the results reported here may not show its true potential. FPCA does really well on the first two problems, but less so on `nucnorm05`. This is most likely due to the fixed value of the penalty parameter, which also leads to a higher misfit and objective. CVX/SDPT3 does not scale very well with the problem size and, as noted in [103], can hardly be used for matrices with more than 100 rows and columns. Having said this, we acknowledge that, as a consequence of the way projection is currently implemented, SPG-TIGHT and PQN do not scale very well either. This can be improved by replacing full singular value decomposition by a partial one, preferably of minimal size. Interestingly, SPG-TIGHT is more accurate, and nearly twice as fast as SPGL1 on `nucnorm02` and `nucnorm05`. For the noisy problem instances, SPG-TIGHT takes about twice as long as SPGL1, but does obtain substantially more accurate solutions. FISTA also has a good performance, but does not yet scale well because of the way we implemented the singular value thresholding proxy function. Finally, PQN is not very well suited to the nuclear norm minimization problem, as a consequence of its strong dependence on projection, which is exactly the bottleneck.

	Solver	Time (s)	#SVD	$\|X\|_*$	$\frac{\lvert\|\|r\|_2-\sigma\rvert}{\max(1,\sigma)}$	$\lvert X-X^*\rvert$
nucnorm02	CVX/SDPT3	1.0e+1	n.a.	2.0179e+2	0.0e+0	—
	SPGL1	9.1e+0	540	2.0180e+2	9.7e−5	3.5e−1
	SPG-TIGHT	4.4e+0	524	2.0179e+2	5.9e−4	3.2e−1
	PQN	1.5e+1	2554	2.0180e+2	9.5e−4	3.3e−1
	FPCA	5.9e+0	4517	2.0179e+2	5.8e−5	2.5e−4
	SVT	3.2e+0	500	2.0184e+2	4.6e−2	9.4e−1
nucnorm04	CVX/SDPT3	6.6e+1	n.a.	7.1411e+2	0.0e+0	—
	SPGL1	7.8e+1	1074	7.1470e+2	8.3e−5	1.8e+1
	SPG-TIGHT	9.7e+1	2778	7.1415e+2	8.4e−4	2.8e+0
	PQN	4.8e+2	20033	7.1435e+2	2.5e−13	7.9e+0
	FPCA	1.1e+1	4546	7.1433e+2	2.4e−4	8.6e+0
	SVT	1.6e+2	500	8.7067e+2	4.4e+1	1.1e+2
nucnorm05	CVX/SDPT3	1.0e+3	n.a.	2.2578e+3	0.0e+0	—
	SPGL1	5.2e+2	1259	2.2583e+3	1.9e−5	1.9e+1
	SPG-TIGHT	3.9e+2	2106	2.2579e+3	4.1e−3	5.5e+0
	PQN	1.8e+3	13132	2.2590e+3	8.0e−3	8.8e+0
	FPCA	3.3e+1	4666	2.3208e+3	7.8e−4	2.8e+2
	SVT	6.4e+2	500	3.6454e+3	2.1e+2	5.0e+2
nucnormn02	CVX/SDPT3	1.3e+1	n.a.	2.0061e+2	3.1e−9	4.3e−5
	SPGL1	2.1e+0	110	2.0063e+2	3.8e−5	3.0e−1
	SPG-TIGHT	4.6e+0	519	2.0061e+2	5.2e−7	5.5e−3
	PQN	1.2e+1	2058	2.0061e+2	1.3e−6	1.1e−2
	FPCA	3.4e+0	2500	2.0045e+2	7.7e−2	5.0e−1
	FISTA	6.3e+0	538	2.0061e+2	3.7e−1	8.9e−6
nucnormn04	CVX/SDPT3	8.9e+1	n.a.	6.9528e+2	2.1e−11	4.2e−4
	SPGL1	1.2e+1	138	6.9531e+2	9.2e−5	3.8e+0
	SPG-TIGHT	2.6e+1	679	6.9528e+2	3.7e−5	6.1e−2
	PQN	1.5e+2	5490	6.9528e+2	6.5e−5	3.2e−1
	FPCA	5.2e+0	2000	6.9383e+2	3.2e−1	1.7e+1
	FISTA	7.4e+1	1319	6.9528e+2	8.5e−1	1.8e−4
nucnormn06	CVX/SDPT3	9.8e+2	n.a.	3.9998e+2	1.9e−8	1.3e−4
	SPGL1	2.7e+1	51	3.9999e+2	6.9e−5	1.3e+0
	SPG-TIGHT	3.0e+1	134	3.9998e+2	3.1e−7	4.7e−3
	PQN	1.4e+2	833	3.9998e+2	4.1e−7	6.1e−2
	FPCA	1.2e+1	2000	3.9974e+2	5.0e−2	2.9e+0
	FISTA	7.6e+1	235	3.9998e+2	7.3e−1	1.9e−3

▶ Table 7.13: Solver performance on a set of matrix completion problems.

Chapter 8

Sparco and Spot

In this chapter we describe two Matlab software packages that were originally intended to support the development of SPGL1 but have evolved to become significant codes in their own right. The first of the two packages, called SPARCO, provides a suite of problems for testing and benchmarking algorithms for sparse recovery. The second package, called SPOT, is a Matlab toolbox for the construction, application, and manipulation of linear operators [11].

8.1 Sparco

Collections of test problem sets, such as those provided by Matrix Market [18], COPS [56], and CUTEr [81], are routinely used in the numerical linear algebra and optimization communities as a means of testing and benchmarking algorithms. These collections are invaluable because they provide a common and easily identifiable reference point, which is especially important when objectively comparing the performance of different algorithms.

When we developed SPGL1 [10], there did not exist any coherent set of test problems for sparse recovery. At that time, all published codes were evaluated on either random or paper-specific test problems. Moreover, with the noticeable exceptions of SparseLab [64], ℓ_1-magic [31], and GPSR [78], no code was generally available for reproducing the given problems.

This situation motivated us to compile a standard set of test problems and make them available through a consistent interface, which was realized with SPARCO [12]. We next discuss the design criteria of SPARCO, and then move on to discuss the particulars of the test problems.

8.1.1 Design objectives

The design of SPARCO was guided by a number of criteria that help promote the wide use of the ensuing test suite. We designed SPARCO to be

Open, freely available, and extensible. A test suite should be considered the collective property of the research community as a whole, rather than of a small group of individuals. Making the software open, freely available, and extensible creates a situation in which research groups can share or contribute to a common set of test problems. This is conducive to a collaborative and transparent research process, as it establishes a widely accepted benchmark free of potential bias.

General and coherent. When providing a set of test problems it is important that the problems are sufficiently general; the choice of problem formulation should not severely restrict the class of applicable algorithms. On the other hand, the test set should provide a coherent class of problems; it would not make sense to mix MMV problems with matrix completion problems.

Comprehensive. The variety of problems in a test suite is extremely important for a number of reasons. First, given an established set of test problems, it is natural for researchers to try and make their code perform as good as possible on these problems. This may inadvertently lead to codes that are highly fine tuned for these problems, but show a moderate or even poor performance on other important problems. Second, the inclusion of problems with different characteristic properties may expose situations not anticipated by the solvers, thus helping improve robustness of codes. Finally, partially related to the first point, it is important to provide problems with different gradations of size and difficulty. In particular, the problem set should include challenging test problems that motivate the development of more advanced approaches.

8.1.2 Test problems

The current version of SPARCO focuses on sparse recovery and compressed sensing problems that can be expressed in the form

$$b = Ax_0 + r,$$

where $A := MB$ is the combination of a measurement matrix M and a sparsity basis B, and r is an additive noise vector. For sparse recovery problems we typically have no additive noise ($r = 0$), and use the identity matrix for the sparsity basis ($B = I$). For compressed sensing, as well as inpainting, b is typically obtained by directly applying the measurement matrix to the original signal s, giving $b = Ms + r$. In these applications we typically do not know the value of x_0 and have to make the assumption that we can write s as Bx_0 with a sparse or compressible x_0. Although this may appear to be a disadvantage, it does reflect practical problems. Moreover, unlike with sparse recovery, these two applications are not so much concerned with finding a sparse representation x_0, but rather with recovering the original signal s from partial or compressed data.

This brings us to the aspect of qualitative versus quantitative test problems. In qualitative experiments we are especially interested in algorithms that solve the intended problem, i.e., recover x_0 or reconstruct s. The focus here lies on the quality of the solution. In quantitative tests we are merely interested in solving a problem (formulation) as fast and accurate as possible, and are not so much interested in the solution itself. As an example, consider sparse recovery from $b = Ax_0$. In qualitative experiments the main focus would be on the recovery rate of x_0 with different algorithms. For quantitative experiments we fix the formulation, such as basis pursuit, and compare algorithms that solve

Seismic imaging	Hennenfent and Herrmann [94, 93]
Blind source separation	Wang et al. [151]
MRI imaging	Lustig et al. [104], Candès et al. [32]
Basis pursuit	Chen et al. [39], Donoho and Johnstone [63]
Compressed sensing	Candès and Romberg [31], Takhar et al. [137]
Image processing	Takhar et al. [137], Figueiredo et al. [78]

▶ Table 8.1: List of sources for SPARCO test problems.

Problem	ID	m	n	$\|b\|_2$	operator
angiogram	502	3321	10000	1.6e+1	2D FFT
cosspike	3	1024	2048	1.0e+2	DCT
dcthdr	12	2000	8192	2.3e+3	restricted DCT
finger	703	11013	125385	5.5e+1	2D curvelet
gcosspike	5	300	2048	8.1e+1	Gaussian ensemble, DCT
jitter	902	200	1000	4.7e–1	DCT
p3poly	6	600	2048	5.4e+3	Gaussian ensemble, wavelet
phantom1	501	629	4096	5.1e+0	2D FFT, wavelet
phantom2	503	21679	65536	6.0e+1	2D FFT
seismic	901	41472	480617	1.1e+2	2D curvelet
sgnspike	7	600	2560	2.2e+0	Gaussian ensemble
soccer1	601	3200	4096	5.5e+4	binary ensemble, wavelet
spiketrn	903	1024	1024	5.7e+1	1D convolution
srcsep1	401	29166	57344	2.2e+1	windowed DCT
srcsep2	402	29166	86016	2.3e+1	windowed DCT
yinyang	603	1024	4096	2.5e+1	wavelet
zsgnspike	8	600	2560	3.1e+0	Gaussian ensemble

▶ Table 8.2: Selection of SPARCO test problems.

that problem; in this case we only criterion is that x^* solves (BP), regardless of whether it coincides with x_0 or not.

The limited availability of well-established test problems meant that we had to construct new problems. In order to do so, we performed a comprehensive literature survey and implemented some 25 problems arising in a variety of contexts (see Table 8.1). While the original intention behind SPARCO was to provide test problems for quantitative benchmarking, most of the test scenarios found in the literature were motivated by practical applications, and are therefore mostly qualitative in nature. Nevertheless, they are still suitable for quantitative testing, and in fact provide some of the more challenging problems when compared, for example, to randomly generated test problems. A selection of the problems, along with their name, identifier, and other important properties is listed in Table 8.2. A complete list of SPARCO problems and their implementation is available online at `http://www.cs.ubc.ca/labs/scl/sparco/index.php`.

Field	Description
M	Measurement matrix
B	Sparsity basis
A	Operator ($A := MB$)
b	Observed vector ($b := Ms$, or $b := Ax_0$)
r	Additive noise vector
x_0	Sparse coefficient vector (optional)
signal	Measured signal s ($s := Bx_0$, when x_0 is available)
reconstruct	function handle to reconstruct signal s from coefficients x.
info	problem information structure (name, title, sparcoID, etc.)

▶ Table 8.3: Fields in the data structure representing SPARCO problems.

8.1.3 Implementation

SPARCO is implemented in Matlab and is set up in such a way that it facilitates the addition of test problems contributed by the community. This is achieved by implementing each test problem as a stand-alone Matlab script and storing them in a dedicated directory where SPARCO can automatically find them. A new problem is thus added to the suite simply by copying the script.

Problems can be instantiated through the user interface. This interface provides consistent access to the problems by means of their name or numerical identifier. As an example, consider the seismic interpolation problem discussed in Section 1.4. This problem is included in SPARCO as problem 901 (`seismic`). To instantiate it, a call is made to the `generateProblem` function with the desired problem identifier:

```
P = generateProblem(901);      % or
P = generateProblem('seismic');
```

The result of this call is a structure that contains relevant problem information, which includes the linear operators M, B, and B, as well as vectors b, r, s, and, when available, x_0. A complete list of fields is given in Table 8.3 For other details concerning the implementation of SPARCO please see [12].

Problem scripts. The information for each test problem is generated by a corresponding script located in the problem directory. Each script implements a predefined interface to enable incorporation into the SPARCO framework. Most of the code in these problem scripts is concerned with setting up the problem structure. This involves loading or generating the sparse coefficient vector or signal data, as well as the construction measurement matrix M, and sparsity basis B. Both M and B are represented implicitly using the linear operators provided by SPOT. We will discuss this software package in Section 8.2. For self-documentation, each problem script can include code to generate figures illustrating the problem, as well as a problem description and citations regarding its origin.

8.2 Spot

Spot [11], as mentioned in the introduction, is a Matlab toolbox for the construction, application, and manipulation of linear operators. One of the main goals of the package is to present operators in such a way that they are as easy to work with as explicit matrices, with the difference that they are represented implicitly.

The most elementary operation involving operators is the equivalent of matrix-vector multiplication. As an example, consider the application of a one-dimensional discrete cosine transform on a vector x of length 16, using one of three approaches:

```
>                 > D = opDCT(16);   > D = dct(eye(16));
> y = dct(x);     > y = D  * x;      > y = D  * x;
> z = idct(y);    > z = D' * y;      > z = D' * y;
```

In the left-most column we directly use the `dct` and `idct`. This approach takes advantage of fast routines available for evaluating the discrete cosine transform and its inverse, but lacks in flexibility. At the other extreme, in the right-most column, we explicitly instantiate the corresponding matrix representation by applying DCT to the identity matrix. This simplifies subsequent application and manipulation of the transformation, but at the expense of scalability and computational efficiency. SPOT, in the middle provides a balance between these two approaches, with the scalability and efficiency of fast multiplication routines, and ease of representation of a matrix. Note that, just like the matrix version, we need to specify the size of the operator when instantiating it.

SPOT arose from the need to construct overcomplete dictionaries consisting of the union of bases. This arises, for example, the DCT-Dirac dictionary $A = [\Phi_1, \Phi_2]$ discussed in Section 1.2. While it is not hard to write a function that multiplies a vector by A or A^T, one does easily get weary from writing code for yet another combination of bases. With the help of SPOT we can simply write what we want to do:

```
I = opEye(256);
D = opDCT(256);
A = [I, D];
```

While the above code, as intended, makes A look like an explicit matrix, it is stored fully implicit, and for multiplication still takes advantage of the `dct` and `idct` routines.

The precursor to SPOT, implemented as part of SPARCO, was written based on function handles and anonymous functions. For easier maintenance and increased flexibility, SPOT was written based on the object oriented and operator overloading features provided in the newer versions of Matlab. In the following sections we describe how the basic operators, such as multiplication and addition, are overloaded in SPOT to combine and manipulate more elementary operators. For a list of some of the most important elementary operators, see Table 8.4.

`opEye`	Identity matrix
`opDiag`	Diagonal matrix
`opOnes`	Matrix with all ones
`opZero`	Matrix with all zeros
`opRestriction`	Restriction operator
`opBlockDiag`	Block-diagonal operator
`opMatrix`	Wrapper for explicit matrices
`opDCT`	One-dimensional fast discrete cosine transform
`opDCT2`	Two-dimensional fast discrete cosine transform
`opDFT`	One-dimensional fast Fourier transform
`opDFT2`	Two-dimensional fast Fourier transform
`opHadamard`	Hadamard basis
`opHeaviside`	Heaviside matrix
`opToeplitz`	Toeplitz or circular matrices
`opWavelet`	Discrete wavelet transformation based on the Rice wavelet toolbox [3]
`opCurvelet`	Discrete curvelet transform based on CurveLab 2.0 [27]
`opGaussian`	Explicit or implicit Gaussian ensemble; i.e., matrices with i.i.d. normally distributed entries
`opConvolve`	One- or two-dimensional convolution.

▶ Table 8.4: List of several elementary operators provided by SPOT.

As a final remark, we note that SPOT can be applied to images by working with the vectorized form; SPOT operators for operations on two-dimensional data internally reshape vectors to matrices, when needed.

8.2.1 Multiplication

Multiplication by operators is the most elementary operation in SPOT. In its simplest form we have operator-vector products such as follows. (Here, and throughout this section, we denote A–D for arbitrary SPOT operators, lowercase variables for vectors, and M an explicit matrix.)

```
y = A * x;
z = B * y;
n = m'* C;  % Evaluated as n = (C^H m)^H
```

This code evaluates the application of A on x, and then B to y. In both cases the result will be an explicit vector of real or complex numbers. An equivalent approach is to first create a new operator C representing the product $B \cdot A$, and then apply this to x to directly get vector z:

```
C = B * A;  % Construct compound operator
z = C * x;  % Evaluate z = BAx
```

Although SPOT operators are expected to support only multiplication of vectors, it is possible to write operator-matrix multiplications. Internally, this is

implemented as a series of operator-vector multiplications.

```
N = C * M;
N = M * C;  % Evaluated as N = (C^H M^H)^H
```

A special case of multiplication is multiplication by scalars. Generally, this will give a new operator scaled by that quantity;

```
C = 3 * A;  % Construct compound operator C = 3A
C = A * 3;
```

One notable exception is when the corresponding dimension of the operator is one. In that case we have a valid matrix-vector product which results in a numeric solution. In other words, matrix-vector products take precedence over scalar multiplication.

8.2.2 Transposition and conjugation

As mentioned in the introduction, each SPOT operator implements multiplication by itself and its conjugate. Using Matlab's transpose operator `'` returns a new operator in which these two modes are switched.

```
B = A';
x = A'* y;
x = B * y; % Identical result as previous line
```

When using transposes within calculations these new operators are discarded as soon as the multiplication has been done. The transpose of a complex operator, rather than its conjugate transpose, can be formed using the `.'` operator. This is implemented as the elementwise conjugate of the conjugate transpose:

```
A = C.'; % Transpose of complex operator, C^T = conj(C^H)
```

8.2.3 Addition and subtraction

The next elementary operation is the addition and subtraction of operators. In its simplest form we add two or more operators;

```
x = (B + C + D) * y;
A = B + C + D;
x = A * y;           % Equivalent to first statement
```

When SPOT encounters the sum of an operator with a matrix or some class object that implements the multiplication and size operators, it will first wrap that entity to a SPOT operator. This allows us to write

```
C = A + M;  % Addition of operator and matrix
C = M + A;  % Addition of matrix and operator
```

Addition of scalars to an operator is interpreted as an elementwise addition, just like it is done for matrices. In order to make this work we first create a new operator of appropriate size, consisting of only ones, and premultiply that with the scalar. The following two statements are therefore equivalent

```
A = B + 3;
A = B + 3*opOnes(size(B));
```

Subtraction is implemented by scalar multiplication combined with addition.

```
A = B - C;
A = B + (-C);
```

8.2.4 Dictionaries and arrays

The original motivation for developing SPOT was the concatenation of operators to form a dictionary. This can now be achieved simply by writing

```
A = [B,C,M]; % or
A = [B C M];
```

Note that explicit matrices and classes can be mixed with SPOT operators, provided they are all compatible in size. Like in addition, these are automatically converted to SPOT operators. Vertical concatenation is done by typing

```
A = [B;C;M]; % or
A = [B
     C
     M];
```

With these two operations specified, Matlab automatically converts arrays of operators into a vertical concatenation of dictionaries:

```
A = [B C; C' M]; % or
A = [B   C
     C'  M];     % both represent A = [[B,C];[C',M]];
```

8.2.5 Block diagonal operators

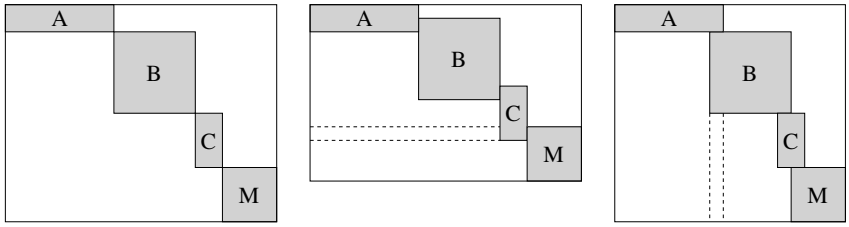
Analogous to matrices of operators it is possible to construct block diagonal operators with operator blocks. This is done using the `blkdiag` command, which takes a list of operators and matrices to create the desired operator:

```
D = blkdiag(A,B,C,M);
```

There is no restriction on the dimension of each operator, and the resulting operator need not necessarily be square. By calling the underlying operator `opBlockDiag` directly, it is possible to specify horizontal or vertical overlap; see Figure 8.1.

8.2.6 Kronecker products

The `kron` operator allows the creation of Kronecker products between operators. Unlike Matlab's built-in `kron` function, the `kron` product in SPOT can take an arbitrary number of arguments. Hence to construct $D = A \otimes B \otimes C$ we can type



No overlap Vertical overlap Horizontal overlap

▶ Figure 8.1: Overlapping block-diagonal operators.

```
D = kron(A,B,C);
```

Needless to say, products with Kronecker operators can quickly increase in computational complexity. Given a set of operators of size $m_i \times n_i$, $i = 1, \ldots, k$, let $m = \prod m_i$, and $n = \prod n_i$. Then the application of the Kronecker product will require n/n_i multiplications with each operator i, and likewise m/m_i products in transpose mode.

Kronecker products can be useful when operating on high dimensional data represented in vectorized form. For example, let x represent a data volume of dimensions $l \times m \times n$. To create an operator that applies a one-dimensional Fourier transformation on vectors along the second dimension, we can simply write

```
op = kron(opEye(n),opDFT(m),opEye(l));
```

8.2.7 Subset assignment and reference

For certain applications, such as the restricted Fourier observations in the MRI example in Section 1.3.2, we may be interested in applying only a part of an operator. This can be done by creating new operators that are restrictions of existing operators. Restriction can be done in terms of rows, columns, a combination of rows and columns, and in terms of individual elements of the underlying matrix. Like in matrices, this is done by indexing

```
A = B(3:5,:);      % Extract rows 3-5
A = C(:,4:6);      % Extract columns 4-6
A = D(3:5,4:6);    % Extract rows 3-5 of columns 4-6
```

The single colon ':' indicates that all elements in the given dimension are selected, and is equivalent to writing the range `1:end`.

```
A = B(3:5,1:end); % Extract rows 3-5
```

There is no restriction on the ordering of the selected columns and rows, and it is possible to repeat entries.

```
A = B([1,2,2,5,3],end:-1:1); % Extract rows in reverse
```

Because row and column indexing is implemented by pre- and post multiplication by selection matrices, the underlying operator is only applied once for each use of the restricted operator.

As an illustration of the use of the subset selection we create an operator that consists of randomly selected rows of the discrete Fourier transform. The first step is to construct a Fourier transform of the desired dimension using the `opDFT` command. After that we choose which rows we want to keep and use subset selection to construct the restricted operator:

```
F = opDFT(128);
R = F(rand(128,1)<=0.8,:);
```

This generates a restricted Fourier operator with approximately 80% of all rows selected.

Assignment. Existing operators can be modified by assigning new values to a subset of the entries of the underlying matrix representation. For example we may want to zero out part of some operator A:

```
A(2,:) = 0;      % Zero out row two
A(:,[3,5]) = 0;  % Zero out columns three and five
```

These operations are not restricted only to zero values; other constants can be used as well. In fact, with the same ease we can assign matrices and other operators to rectangular subsets of existing operators.

A special situation arises when assigning the empty set `[]` to a subset of rows or columns. Doing so causes the given rows or columns to be deleted from the operator. This is implemented by respectively pre- and post multiplying by a restriction matrix. This means that multiplication by the resulting operator still requires application of the full original operator whether or not the relevant rows or columns are needed.

8.2.8 Elementwise operations

There are a number of operations that, just like multiplication by scalars, affect all individual entries of matrix underlying the operator. For complex operators SPOT provides the commands `real`, `imag`, and `conj`. The first command discards the imaginary parts of the operator and results in a real operator, while the second command keeps only the imaginary part of the operator (this results in a real number operator). The `conj` command replaces each element by its complex conjugate. When applied to a real operator, `real` and `conj` do nothing, while `imag` returns a new zero operator (all underlying operators are discarded).

```
F = opMatrix(2 + 3i);
a = real(F);          % a = opMatrix(2)
b = imag(F);          % b = opMatrix(3)
c = conj(F);          % c = opMatrix(2 - 3i)
```

Application of `real`, `imag`, or `conj` leads to the creation of a new operator. Provided the new operator is not the zero operator, multiplication is implemented by distinguishing between real, imaginary, and complex vectors. In the first two cases a single application of the underlying operator suffices. For complex vectors we need to apply the underlying operator to both the real and imaginary parts of the vector and combine the results according to the desired operation, thus requiring two operations per multiplication.

8.2.9 Solving systems of linear equations

The backslash operator \ can be used to solve linear least-squares problems involving SPOT operators.

```
x = A \ b;
```

This problem is solved by calling the LSQR code by Paige and Saunders [125]. The same applies to the pseudo-inverse, which solves a least-squares problem whenever it is applied to a vector:

```
P = pinv(A);
x = P * b;     % Equivalent to x = A \ b;
```

Matlab provides a number of routines for solving systems of linear equations, such as `pcg` and `gmres`, that either take a matrix or a function handle. To make them compatible with operators, SPOT provides a set of thin function wrappers.

8.2.10 Application to non-linear operators

In some situations it may be desirable to create nonlinear operators. For example, when the output of a certain series of operations on complex input is known to be real we may want to have an operator that discards the imaginary part of its input. While SPOT permits the construction of such operators, utmost care has to be taken when using them in combination with other operations such as restriction. Because there are no sanity checks in the implementation of the meta operators, application to nonlinear operators is not guaranteed to be meaningful; this is something the user should decide.

Chapter 9

Conclusions

Sparse recovery is concerned with the reconstruction of vectors or matrices, which are sparse in appropriate sparsity measures, from compressed or incomplete, and possibly noisy representations. In particular, it is concerned with various reconstruction algorithms, and the conditions under which these algorithms recover the desired quantity either exactly or approximately. A large class of reconstruction methods is based on the solution of convex optimization problems. Besides the theoretical properties associated with each formulation, there is the practical aspect of solving the underlying optimization problem. In this thesis we present an algorithm, or framework, which can solve a variety of constrained convex formulations that frequently arise in sparse recovery. In addition, we study the theoretical properties of two methods for joint-sparse recovery. The introduction of this thesis describes the developments in signal processing that led to the emergence of sparse recovery, and gives an overview of several applications, as well as current topics in sparse recovery.

Theoretical results. In the first half of this thesis we obtain theoretical results for the recovery from a matrix B containing multiple measurement vectors whose underlying coefficients X_0 have identical support. A popular convex formulation for solving the MMV problem is minimization the sum of row-norms of X subject to $AX = B$. We show that for uniform recovery of signals on a fixed support, sum-of-norm minimization cannot outperform $\ell_{1,1}$, regardless of the norm used. This is despite the fact that $\ell_{1,1}$ reduces to a series of basis pursuit problems for each column separately, and does not take advantage of the prior information that the columns in X_0 have the same support (or are subsets of the joint support). However, empirical results show that $\ell_{1,2}$ easily outperforms $\ell_{1,1}$ on average, i.e., when applied to randomly generated problems. We give concrete examples where $\ell_{1,2}$ succeeds and $\ell_{1,1}$ fails. This results also serves to highlight that the magnitude of the entries of X_0 can affect recovery. This is unlike recovery using $\ell_{1,1}$, in which only the sign and support matters.

We observe that, beyond the uniform-recovery regime, the recovery rate of $\ell_{1,1}$ degrades rapidly as the number of observations increases. This phenomenon is explained by the subsequent analysis of $\ell_{1,1}$ recovery, which also explains the poor recovery rates reported by others.

We use the geometrical interpretation of basis pursuit to analyze recovery using ReMBo-ℓ_1. We show how recovery is intricately related to the orientation of the subspace spanned by the columns of X_0, and the number of orthants this subspace intersects. We derive an exact expression for the maximum number of

orthants any d-dimensional subspace in $\mathbb{R}^n$ can intersect. The dimension d of the subspace depends on the rank of X_0, and it follows from our results, that adding more observations without increasing d does not increase the number of intersected orthants. Empirical results show, however, that additional observations do increase the probability of reaching certain orthants, which, in turn, increases the efficiency of ReMBo-ℓ_1.

This work has led to a number of questions that are interesting as future work. First, given a set $\{v_1, \ldots, v_n\}$ of randomly distributed vectors, what is the distribution of the external angle of the conic hull of $\{\pm v_1, \pm v_2, \ldots, \pm v_n\}$. Knowing this distribution would help establish the expected number of unique sign-patterns reached by random combinations of vectors v_i, after a fixed number of trials. Second, it would be interesting to study the distribution of faces of cross-polytope $\mathcal{C}$ that vanish under linear mappings, especially the distribution over the faces of $\mathcal{C}$ that are intersected by random d-dimensional subspaces.

Solver for generalized sparse recovery. The main contribution of this thesis is a solver framework for solving sparse recovery problem formulations that are subject to constraints on a measure of misfit. Our implementation of this solver, SPGL1, is available on-line, and has been successfully used by many practitioners. SPGL1 is still one of the very few solvers that can deal with explicit misfit constraints, as opposed penalized formulations, which are the easier to solve, but less natural in their use, because they require the user to provide the Lagrange multiplier corresponding to the constraint.

Our numerical experiments show that SPGL1 does quite well on a range of problems. The main problem with the current implementation is the tendency to slightly overshoot the target objective τ_σ, due to the inaccuracy of the sub-problem solves. This is easily avoided by tightening the tolerance values, but this comes at the cost of an increase of run time. In future work we aim to provide the user with a number of options ranging from fast and dangerous to slow and accurate. The emergence of new test problems helps tune the solver, which is a continuing process until the solver reaches a certain level of maturity.

We have considered two types of algorithms for solving the generalized Lasso sub-problems: the spectrally projected gradient (SPG) method, and a projected quasi-Newton (PQN) method. The latter method is designed for problems where evaluating the function and gradient values of the sub-problem is expensive. To reduce the cost, it forms quadratic models of the objective function and optimizes this model on the feasible set to obtain a search direction for the original problem. Applied to our test problems, PQN does indeed reduce the number of function and gradient evaluations substantially. However, due to the relatively low cost of these evaluations, it does not outperform SPG in terms of run time. As future work, we will study the use of alternative solvers for the generalized Lasso problem. In particular, we plan to incorporate the optimal first-order method by Nesterov [118], and use his complexity results to derive a corresponding complexity of the entire root-finding approach.

Finally, two major modifications to SPGL1 are needed for the matrix-com-

pletion problem. First of all, we need to avoid the evaluation of the full singular value decomposition used in projection, and reuse as much information as possible. This means that projection can no longer be treated as a separate black-box routine, but needs to be integrated, so that information gathered at previous iterations can be used to ensure that the size of the partial SVD is no larger than needed. Second, for scalability of the algorithm, we need to modify the code to work with the partial SVD representation of X, instead of the full matrix.

Sparco and Spot. SPARCO provides a set of test problems for sparse recovery based on single measurement vectors. As such, it is mostly suited for the basis pursuit, and basis pursuit denoise formulations. The numerical experiments done in this thesis suggest that SPARCO needs a number of modifications. First, it would be beneficial to include more synthetic and noisy problems. This requires some careful choice of problem instances to avoid cluttering the suite (once a problem is added and used in benchmarks, it may be hard to remove later). Second, given the fact that the majority of the existing solvers apply to regularized basis pursuit, it may be desirable to include with each problem, the corresponding (σ, λ) pairs, as well as an accurate solution. Third, following developments in sparse recovery, it would be useful to include problems for other formulations as separate test sets.

The SPOT linear operator toolbox is extremely convenient for the construction of linear operators. With the foundation of the package rigorously set up, it only remains to keep adding useful operators, as they arise. Hence, the current version should be regarded as the first version of a continuously evolving toolbox.

Bibliography

[1] Farid Alizadeh and Donald Goldfarb. Second-order cone programming. *Mathematical Programming, Serie B*, 95(1):3–51, January 2003.

[2] Waheed U. Bajwa, Jarvis D. Haupt, Gil M. Raz, Stephen J. Wright, and Robert D. Nowak. Toeplitz-structured compressed sensing matrices. In *IEEE/SP 14th Workshop on Statistical Signal Processing, 2007*, pages 294–298, August 2007.

[3] Richard Baraniuk, Hyeokho Choi, Felix Fernandes, Brent Hendricks, Ramesh Neelamani, Vinay Ribeiro, Justin Romberg, Ramesh Gopinath, Haitao Guo, Markus Lang, Jan Erik Odegard, and Dong Wei. Rice Wavelet Toolbox, 1993. http://www.dsp.rice.edu/software/rwt.shtml.

[4] Richard Baraniuk, Mark Davenport, Ronald DeVore, and Michael Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3):253–263, December 2008.

[5] Jonathan Barzilai and Jonathan M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

[6] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[7] Stephen Becker, Jérôme Bobin, and Emmanuel J. Candès. NESTA: A fast and accurate first-order method for sparse recovery, 2009. Submitted.

[8] Ewout van den Berg and Michael Friedlander. Joint-sparse recovery from multiple measurements. arXiv:CS/0904.2051, April 2009.

[9] Ewout van den Berg and Michael P. Friedlander. SPGL1, 2007. http://www.cs.ubc.ca/labs/scl/index.php/Main/Software.

[10] Ewout van den Berg and Michael P. Friedlander. Probing the Pareto frontier for basis-pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008.

[11] Ewout van den Berg and Michael P. Friedlander. Spot, 2009. http://www.cs.ubc.ca/labs/scl/spot/.

[12] Ewout van den Berg, Michael P. Friedlander, Gilles Hennenfent, Felix J. Herrmann, Rayan Saab, and Özgür Yılmaz. Algorithm 890: Sparco: A testing framework for sparse reconstruction. *ACM Transactions on Mathematical Software*, 35(4):1–16, February 2009.

[13] Ewout van den Berg, Mark Schmidt, Michael P. Friedlander, and Kevin Murphy. Group sparsity via linear-time projection. Technical Report TR-2008-09, Dept. of Computer Science, UBC, June 2008.

[14] Radu Berinde, Anna C. Gilbert, Piotr Indyk, Howard J. Karloff, and Martin J. Strauss. Combining geometry and combinatorics: A unified approach to sparse signal recovery. In *Forty-sixth annual Allerton conference on Communication, Control, and Computing*, September 2008.

[15] Dimitri P. Bertsekas. On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Transactions on Automatic Control*, AC-21(2):174–184, April 1976.

[16] Dimitri P. Bertsekas, Angelia Nedić, and Asuman E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, 2003.

[17] Ernesto G. Birgin, José Mario Martínez, and Marcos Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10(4):1196–1211, 2000.

[18] Ronald F. Boisvert, Roldan Pozo, Karin Remington, Richard Barrett, and Jack J. Dongarra. Matrix Market: A web resource for test matrix collections. In Ronald F. Boisvert, editor, *The quality of numerical software: assessment and enhancement*, pages 125–137. Chapman & Hall, London, 1997.

[19] Petros T. Boufounos and Richard G. Baraniuk. Quantization of sparse representations. Technical Report 0701, ECE Department, Rice University, March 2007.

[20] Petros T. Boufounos and Richard G. Baraniuk. Sigma delta quantization for compressive sensing. In *Proceedings of SPIE*, volume 6701, pages 1–13, August 2007.

[21] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[22] Kristian Bredies and Dirk A. Lorenz. Linear convergence of iterative soft-thresholding. *Journal of Fourier Analysis and Applications*, 14:813–837, 2008.

[23] Jian-Feng Cai, Emmanuel J. Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. arXiv:0810.3286v1, October 2008.

[24] T. Tony Cai, Guangwu Xu, and Jung Zhang. On recovery of sparse signals via ℓ_1 minimization. *IEEE Transactions on Information Theory*, 55(7):3388–3397, July 2009.

[25] Emmanuel J. Candès. Compressive sampling. In *Proceedings of the International Congress of Mathematicians*, Madrid, Spain, 2006.

[26] Emmanuel J. Candès. The restricted isometry property and its implications for compressed sensing. *Compte Rendus de l'Academie des Sciences Paris, ser. I*, 346:589–592, 2008.

[27] Emmanuel J. Candès, Laurent Demanet, David Donoho, and Lexing Ying. Fast discrete curvelet transforms. *Multiscale Modeling and Simulation*, 5(3):861–899, January 2006.

[28] Emmanuel J. Candès and Yaniv Plan. Matrix completion with noise. arXiv:0903.3131v1, March 2009.

[29] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):1–49, December 2009.

[30] Emmanuel J. Candès and Justin Romberg. Practical signal recovery from random projections. In *Proceedings of SPIE – Volume 5674*, March 2005.

[31] Emmanuel J. Candès and Justin Romberg. ℓ_1-magic, 2007. http://www.l1-magic.org.

[32] Emmanuel J. Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, February 2006.

[33] Emmanuel J. Candès and Terence Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(2):4203–4215, December 2005.

[34] Emmanuel J. Candès and Terence Tao. Near-optimal signal recovery from random projections – universal encoding strategies. *IEEE Transactions on Information Theory*, 52(2), 2006.

[35] Emmanuel J. Candès and Terence Tao. The Dantzig selector: Statistical estimation when p is much larger than n. *Annals of Statistics*, 35(6):2313–2351, 2007.

[36] Emmanuel J. Candès and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. arXiv:0903.1476v1, March 2009.

[37] Emmanuel J. Candès, Michael B. Wakin, and Stephen P. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, 14(5–6):877–905, December 2008.

[38] Jie Chen and Xiaoming Huo. Theoretical results on sparse representations of multiple-measurement vectors. *IEEE Transactions on Signal Processing*, 54:4634–4643, December 2006.

[39] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[40] Jon F. Claerbout and Francis Muir. Robust modeling with erratic data. *Geophysics*, 38(5):826–644, October 1973.

[41] Patrick L. Combettes and Valérie R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling and Simulation*, 4(4):1168–1200, 2005.

[42] Shane F. Cotter, Bhaskar D. Rao, Kjersti Engang, and Kenneth Kreutz-Delgado. Sparse solutions to linear inverse problems with multiple measurement vectors. *IEEE Transactions on Signal Processing*, 53:2477–2488, July 2005.

[43] Wei Dai and Olgica Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Transactions on Information Theory*, 55(5):2230–2249, May 2009.

[44] Wei Dai, Hoa Vinh Pham, and Olica Milenkovic. Quantized compressive sensing. arXiv:0901.0749v2, March 2009.

[45] Wei Dai, Mona A Sheikh, Olgica Milenkovic, and Richard G. Baraniuk. Compressive sensing DNA microarrays. *EURASIP Journal on Bioinformatics and Systems Biology*, 2009:1–12, 2009.

[46] Yu-Hong Dai and Roger Fletcher. Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming. *Numerische Mathematik*, 100:21–47, 2005.

[47] George Bernard Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963.

[48] Alexandre d'Aspremont, Francis R. Bach, and Laurent El Ghaoui. Optimal solutions for sparse principal component analysis. *Journal of Machine Learning Research*, 9:1269–1294, July 2008.

[49] Ingrid Daubechies. *Ten Lectures on Wavelets*. SIAM, 1992.

[50] Ingrid Daubechies, Michel Defrise, and Christine de Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57:1413–1457, November 2004.

[51] Ingrid Daubechies, Massimo Fornasier, and Ignace Loris. Accelerated projected gradient method for linear inverse problems with sparsity constraints. *Journal of Fourier Analysis and Applications*, 14(5–6), December 2008.

[52] Ron S. Dembo, Stanley C. Eisenstat, and Trond Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, April 1982.

[53] Ronald A. DeVore. Deterministic constructions of compressed sensing matrices. *Journal of Complexity*, 23(4–6):918–925, August–December 2007.

[54] David L. Dohono and Jared Tanner. Neighborliness of randomly-projected simplices in high dimensions. *Proceedings of the National Academy of Sciences*, 102(27):9452–9457, 2005.

[55] David L. Dohono and Jared Tanner. Sparse nonnegative solution of underdetermined linear equations by linear programming. *Proceedings of the National Academy of Sciences*, 102(27):9446–9451, 2005.

[56] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with COPS. Technical Report ANL/MCS-246, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 2000. Revised January 2001.

[57] David L. Donoho. Neighborly polytopes and sparse solution of underdetermined linear equations. Technical Report 2005-4, Department of Statistics, Stanford University, Stanford, CA, 2005.

[58] David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, April 2006.

[59] David L. Donoho. High-dimensional centrosymmetric polytopes with neighborliness proportional to dimension. *Discrete and Computational Geometry*, 35(4):617–652, May 2006.

[60] David L. Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ^1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, March 2003.

[61] David L. Donoho, Michael Elad, and Vladimir N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on information theory*, 52(1):6–18, January 2006.

[62] David L. Donoho and Xiaoming Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Transactions on Information Theory*, 47(7):2845–2862, November 2001.

[63] David L. Donoho and Iain M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994.

[64] David L. Donoho, Victoria C. Stodden, and Yaakov Tsaig. Sparselab, 2007. http://sparselab.stanford.edu/.

[65] Pei-Cheng Du and Ruth H. Angeletti. Automatic deconvolution of isotope-resolved mass spectra using variable selection and quantized peptide mass distribution. *Analytical Chemistry*, 78(10):3385–3392, May 2006.

[66] Marco F. Duarte, Mark A. Davenport, Dharmpal Takhar, Jason N. Laska, Ting Sun, Kevin F. Kelly, and Richard G. Baraniuk. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 25(2):83–91, 2008.

[67] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, pages 272–279, 2008.

[68] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

[69] Michael Elad and Alfred M. Bruckstein. A generalized uncertainty principle and sparse representation in pairs of bases. *IEEE Transactions on Information Theory*, 48(9):2558–2567, September 2002.

[70] Michael Elad, Boaz Matalon, Joseph Shtok, and Michael Zibulevsky. A wide-angle view at iterated shrinkage algorithms. In *Proceedings of SPIE*, volume 6701, pages 670102:1–670102:19, September 2007.

[71] Michael Elad, Jean-Luc Starck, Philippe Querre, and David L. Donoho. Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA). *Applied and Computational Harmonic Analysis*, 19(3):340–358, November 2005.

[72] Yonina C. Eldar, Patrick Kuppinger, and Helmut Bölcskei. Compressed sensing of block-sparse signals: Uncertainty relations and efficient recovery. arXiv:0906.3173v2, June 2009.

[73] Yonina C. Eldar and Moshe Mishali. Robust recovery of signals from a union of subspaces. arXiv 0807.4581, July 2008.

[74] Yonina C. Eldar and Holger Rauhut. Average case analysis of multichannel sparse recovery using convex relaxation. arXiv:0904.0494, April 2009.

[75] Maryam Fazel. *Matrix Rank Minimization with Applications*. PhD thesis, Stanford University, 2002.

[76] Arie Feuer and Arkadi Nemirovski. On sparse representation in pairs of bases. *IEEE Transactions on Information Theory*, 49(6):1579–1581, June 2003.

[77] Mário A. T. Figueiredo and Robert D. Nowak. An EM algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12(8):906–916, August 2003.

[78] Mário A. T. Figueiredo, Robert D. Nowak, and Stephen J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, December 2007.

[79] David Gale. Neighborly and cyclic polytopes. *Proceedings of Symposia in Pure Mathematics*, VII:225–232, 1963.

[80] Irina F. Gorodnitsky, John S. George, and Bhaskar D. Rao. Neuromagnetic source imaging with FOCUSS: A recursive weighted minimum norm algorithm. *Electroencephalography and Clinical Neurophysiology*, 95(4):231–251, October 1995.

[81] Nicholas I. M. Gould, Dominique Orban, and Philippe L. Toint. CUTEr and SifDec: A constrained and unconstrained testing environment, revisited. *ACM Transactions on Mathematical Software*, 29(4):373–394, December 2003.

[82] Vivek K. Goyal, Alyson K. Fletcher, and Sundeep Rangan. Compressive sampling and lossy compression. *IEEE Signal Processing Magazine*, 25(2), March 2008.

[83] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Lecture Notes in Control and Information Sciences*, pages 95–110. Springer, 2008.

[84] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming (web page and software), February 2009. http://stanford.edu/~boyd/cvx.

[85] Rémi Gribonval and Morten Nielsen. Sparse representations in unions of bases. *IEEE Transactions on Information Theory*, 49(12):3320–3325, December 2003.

[86] Rémi Gribonval and Morten Nielsen. Highly sparse representations from dictionaries are unique and independents of the sparseness measure. *Applied and Computational Harmonic Analysis*, 22(3):335–355, May 2007.

[87] Rémi Gribonval, Holger Rauhut, Karin Schnass, and Pierre Vandergheynst. Atoms of all channels, unite! Average case analysis of multichannel sparse recovery using greedy algorithms. *Journal of Fourier Analysis and Applications*, 14(5–6):655–687, December 2008.

[88] Luigi Grippo, Francesco Lampariello, and Stephano Lucidi. A nonmonotone line search technique for Newton's method. *SIAM Journal on Numerical Analysis*, 23(4):707–716, August 1986.

[89] Branko Grünbaum. *Convex Polytopes*, volume 221 of *Graduate Texts in Mathematics*. Springer-Verlag, second edition, 2003.

[90] Elaine Hale, Wotao Yin, and Yin Zhang. Fixed-point continuation for ℓ_1-minimization: Methodology and convergence. *SIAM Journal on Optimization*, 19(3):1107–1130, 2008.

[91] Gilles Hennenfent and Felix J. Herrmann. Sparseness-constrained data continuation with frames: Applications to missing traces and aliased signals in 2/3-D. In *SEG International Exposition and 75th Annual Meeting*, 2005.

[92] Gilles Hennenfent and Felix J. Herrmann. Application of stable signal recovery to seismic interpolation. In *SEG International Exposition and 76th Annual Meeting*, 2006.

[93] Gilles Hennenfent and Felix J. Herrmann. Random sampling: New insights into the reconstruction of coarsely-sampled wavefields. In *SEG International Exposition and 77th Annual Meeting*, 2007.

[94] Gilles Hennenfent and Felix J. Herrmann. Simply denoise: Wavefield reconstruction via jittered undersampling. *Geophysics*, 73(3):V19–V28, May–June 2008.

[95] Stephen D. Howard, A. Robert Calderbank, and Stephen J. Searle. A fast reconstruction algorithm for deterministic compressive sensing using second order reed-muller codes. In *Proc. Conference on Information Sciences and Systems (CISS2008)*, pages 11–15, March 2008.

[96] Piotr Indyk. Explicit constructions for compressed sensing of sparse signals. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, January 2008.

[97] Laurent Jacques, David K. Hammond, and Mohamed Jalal Fadili. Dequantizing compressed sensing: When oversampling and non-Gaussian constraints combine. arXiv:0902.2367v2, February 2009.

[98] Sina Jafarpour, Weiyu Xu, Babak Hassibi, and A. Robert Calderbank. Efficient and robust compressed sensing using high-quality expander graphs. arXiv:0806.3802, June 2008.

[99] Seung-Jean Kim, Kwangmoo Koh, Michael Lustig, Stephen Boyd, and Dimitry Gorinevsky. An interior-point method for large-scale ℓ_1-regularized least squares. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):606–617, December 2007.

[100] Dany Leviatan and Vladimir N. Temlyakov. Simultaneous greedy approximation in Banach spaces. *Journal of Complexity*, 21(3), June 2005.

[101] Nathan Linial and Isabella Novik. How neighborly can a centrally symmetric polytope be? *Discrete and Computational Geometry*, 36(2):273–281, 2006.

[102] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming, Serie B*, 45(1–3):503–528, August 1989.

[103] Yong-Jin Liu, Defeng Sun, and Kim-Chuan Toh. An implementable proximal point algorithmic framework for nuclear norm minimization. Preprint, July 2009.

[104] Michael Lustig, David L. Donoho, and John M. Pauly. Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magnetic Resonance in Medicine*, 58(6):1182–1195, December 2007.

[105] Michael Lustig, David L. Donoho, Juan M. Santos, and John M. Pauly. Compressed sensing MRI. *IEEE Signal Processing Magazine*, 25(2):72–82, 2008.

[106] Shiqian Ma, Donald Goldfarb, and Lifeng Chen. Fixed point and Bregman iterative methods for matrix rank minimization. arXiv:0905.1643v2, May 2009.

[107] Dmitry M. Malioutov. A sparse signal reconstruction perspective for source localization with sensor arrays. Master's thesis, Dept. Electrical Engineering, Massachusetts Institute of Technology, Cambridge, MA, July 2003.

[108] Dmitry M. Malioutov, Müjad Çetin, and Alan S. Willsky. Optimal sparse representations in general overcomplete bases. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 793–796, May 2004.

[109] Dmitry M. Malioutov, Müjad Çetin, and Alan S. Willsky. A sparse signal reconstruction perspective for source localization with sensor arrays. *IEEE Transactions on Signal Processing*, 53(8):3010–3022, August 2005.

[110] Dmitry M. Malioutov, Müjdat Çetin, and Alan S. Willsky. Source localization by enforcing sparsity through a Laplacian prior: An SVD-based approach. In *IEEE Workshop on Statistical Signal Processing*, 2003.

[111] Dmitry M. Malioutov, Müjdat Çetin, and Alan S. Willsky. Homotopy continuation for sparse signal representation. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 733–736, March 2005.

[112] Fred W. McLafferty and František Tureček. *Interpretation of Mass Spectra*. University Science Books, Mill Valley, CA, fourth edition, 1993.

[113] Peter McMullen and Geoffrey C. Shephard. Diagrams for centrally symmetric polytopes. *Mathematika*, 15:123–138, 1968.

[114] Moshe Mishali and Yonina C. Eldar. Reduce and boost: Recovering arbitrary sets of jointly sparse vectors. *IEEE Transactions on Signal Processing*, 56(10):4692–4702, October 2008.

[115] Balas K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, April 1995.

[116] National Institute of Standards and Technology. NIST Chemistry WebBook, 2009. http://webbook.nist.gov/chemistry/.

[117] Deanna Needell and Joel A. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 29(3):301–321, May 2009.

[118] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming, Serie A*, 103(1):127–152, May 2005.

[119] Yurii Nesterov. Gradient methods for minimizing composite objective function. Technical Report 2007/76, Center for Operations Research and Econometrics, Catholic University of Louvain, 2007.

[120] Yurii Nesterov and Arkadii Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM, 1994.

[121] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999.

[122] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, second edition, 2006.

[123] Michael R. Osborne, Brett Presnell, and Berwin A. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20(3):389–403, 2000.

[124] Michael R. Osborne, Brett Presnell, and Berwin A. Turlach. On the LASSO and its dual. *Journal of Computational and Graphical Statistics*, 9(2):319–337, 2000.

[125] Christopher C. Paige and Michael A. Saunders. LSQR an algorithm for sparse linear equations and sparse least squares. *ACM Transactions om Mathematical Software*, 8(1):43–71, 1982.

[126] Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. arXiv:0706.4138v1, June 2007.

[127] R. Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, 1970.

[128] Moshe Rosenfeld. *In Praise of the Gram Matrix*, pages 318–323. The Mathematics of Paul Erdös, II. Springer, Berlin, 1997.

[129] Mark Rudelson and Roman Vershynin. Sparse reconstruction by convex relaxation: Fourier and Gaussian measurements. In *40th Annual Conference on Information Sciences and Systems*, pages 207–212, Princeton, March 2006.

[130] Sylvain Sardy, Andrew G. Bruce, and Paul Tseng. Block coordinate relaxation methods for nonparametric wavelet denoising. *Journal of Computational and Graphical Statistics*, 9(2):361–379, 2000.

[131] Mark Schmidt, Ewout van den Berg, Michael P. Friedlander, and Kevin Murphy. Optimizing costly functions with simple constraints: a limited-memory projected quasi-Newton algorithm. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, pages 448–455, April 16–18 2009.

[132] Mona A. Sheikh, Olgica Milenkovic, Shri Sarvotham, and Richard G. Baraniuk. Compressed sensing DNA microarrays. Technical Report TREE0706, ECE, Rice University, May 2007.

[133] R. Martin Smith. *Understanding mass spectra: A basic approach*. John Wiley and Sons, Hoboken, NJ, second edition, 2004.

[134] Eckard Specht. Packing of circles in the unit circle, 2009. http://hydra.nat.uni-magdeburg.de/packing/cci/cci.html.

[135] Mihailo Stojnic, Farzad Parvaresh, and Babak Hassibi. On the reconstruction of block-sparse signals with an optimal number of measurements. *IEEE Transactions on Signal Processing*, 57(8):3075–3085, August 2009.

[136] Thomas Strohmer and Robert W. Heath Jr. Grassmannian frames with applications to coding and communication. *Applied and Computational Harmonic Analysis*, 14(3):257–275, 2003.

[137] Dharmpal Takhar, Jason N. Laska, Michael B. Wakin, Marco F. Duarte, Dror Baron, Shriram Sarvotham, Kevin F. Kelly, and Richard G. Baraniuk. A new camera architecture based on optical-domain compression. In *Proceedings of the IS&T/SPIE Symposium on Electronic Imaging: Computational Imaging*, volume 6065, January 2006.

[138] David S. Taubman and Michael W. Marcellin. *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Kluwer Academic Publishers, Boston, 2001.

[139] Howard L. Taylor, Stephen C. Banks, and John F. McCoy. Deconvolution with the ℓ_1 norm. *Geophysics*, 44(1):39–52, January 1979.

[140] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[141] Peter A. Toft. *The Radon Transform — Theory and Implementation.* PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, June 1996.

[142] Kim-Chuan Toh, Michael J. Todd, and Reha H. Tütüncü. SDPT3 – a Matlab software package for semidefinite programming. *Optimization Methods and Software*, 11(1–4):545–581, 1999.

[143] Kim-Chuan Toh and Sangwoon Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. Preprint, April 2009.

[144] Joel A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10), October 2004.

[145] Joel A. Tropp. Algorithms for simultaneous sparse approximation: Part II: Convex relaxation. *Signal Processing*, 86:589–602, 2006.

[146] Joel A. Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52(3):1030–1051, March 2006.

[147] Joel A. Tropp, Anna C. Gilbert, and Martin J. Strauss. Simultaneous sparse approximation via greedy pursuit. In *Proc. 2005 IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5, pages 721–724, Philadelphia, March 2005.

[148] Joel A. Tropp, Anna C. Gilbert, and Martin J. Strauss. Algorithms for simultaneous sparse approximation: Part I: Greedy pursuit. *Signal Processing*, 86:572–588, 2006.

[149] Berwin A. Turlach. On algorithms for solving least squares problems under an l_1 penalty or an l_1 constraint. In *Proceedings of the American Statistical Association*, pages 2572–2577. American Statistical Association, 2004.

[150] Reha H. Tütüncü, Kim-Chuan Toh, and Michael J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical Programming, Serie B*, 95:189–217, 2003.

[151] Deli Wang, Rayan Saab, Özgür Yılmaz, and Felix J. Herrmann. Bayesian wavefield separation by transform-domain sparsity promotion. *Geophysics*, 73:A33–A38, 2008.

[152] Stephen J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, 1997.

[153] Stephen J. Wright, Robert D. Nowak, and Mário A. T. Figueiredo. Sparse reconstruction by separable approximation. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3373–3376, March 2008.

[154] Wotao Yin, Stanley Osher, Donald Goldfarb, and Jerome Darbon. Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing. *SIAM Journal on Imaging Sciences*, 1(1):143–168, 2008.

[155] Günter M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer-Verlag, first edition, 2006.

[156] Argyrios Zymnis, Stephen Boyd, and Emmanuel J. Candès. Compressed sensing with quantized measurements. Submitted for publication, April 2009.