Practical Compressed Sensing: modern data acquisition and signal processing

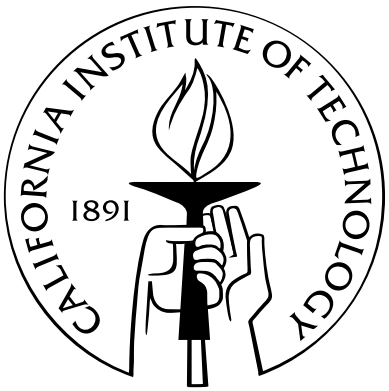
Thesis by

Stephen R. Becker

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2011

(Defended April 7, 2011)

Acknowledgements

It has been my privilege to work with highly talented and motivated researchers over the past few years. Much of the thesis work presented here was inspired or due to direct collaboration with these people and there would be no thesis were it not for them.

The most significant influence has been from my adviser, Professor Emmanuel Candès, who has supported me in many ways since I first came to campus. I owe him not just for scientific ideas, but for guiding me in the world of academics. I'm especially impressed by his integrity, his enthusiasm for the beauty of math, and his desire to use math to improve technology such as MRI. Like many great teachers, he leads by example.

It was a pleasure to work on the RMPI project with Dr. Emilio Sovero and Dr. Eric Nakamura from the Northrop Grumman Corporation; Dr. Michael Grant; and Professors Michael Wakin, Justin Romberg, Azita Emami, and Emmanuel Candès. Most importantly, thanks to my fellow student Juhwan Yoo from the MICS lab at Caltech. He is primarily responsible for the implementation of the RMPI chip. I'm grateful to his dedication and friendship over the past three years. Matthew Loh of the MICS lab also deserves recognition for his significant contributions to the project.

On my optimization endeavors, I've benefited greatly from interactions with many people. A great source of knowledge and friendship has been my co-authors, Jérôme Bobin, Michael Grant, and Emmanuel. Thanks as well to scholars in the field who have at some point contributed to my projects either explicitly or through informative discussions. I'm indebted to Patrick Combettes, Marc Teboulle, Joel Tropp, and Emmanuel for writing letters on my behalf, and I thank the Fondation Sciences Mathématiques de Paris for their support. I was fortunate to attend an IPAM long-workshop "Modern Trends in Optimization and Its Application" in Fall 2010 at UCLA; thanks to Russ Caflisch, Amber Puha, and the NSF for making this happen, and to Lieven Vandenberghe and Emmanuel in particular for the scientific organization. This was a fantastic opportunity for a graduate student. Similarly, I am very grateful to Alexandre d'Aspremont, Francis Bach, and Martin Wainwright for organizing and then inviting me to the "Sparse Statistics, Optimization and Machine Learning" conference in Banff, January 2011.

In the Applied & Computational Math (ACM) department at Caltech, I'm always impressed by the level of thought in the graduate students. Thanks to all of them for friendship, discussions, and

the occasional “ACM-TEA” seminar. The faculty in ACM has always been excellent. Sheila Shull and Sydney Garstang have been incredibly helpful to me and all the other students and faculty. Special thanks to Alex Gittens and Michael McCoy for academic discussions, as well as the non-academic discussions during the commute to UCLA.

Many thanks to my committee members: Lieven Vandenberghe, Babak Hassibi, Azita Emami, Joel Tropp, and Emmanuel Candès. Joel especially has been instrumental to my academic development over the past three years, and it's clear he cares deeply about the students at Caltech. The Candès group, past and present, has been a great source of role models for a young graduate student. Thanks to Ewout van den Berg, Laurent Demanet, Hannes Helgason, Ben Recht, Justin Romberg, Mike Wakin, Lexing Ying, Paige Randall, Peter Stobbe, Jérôme Bobin, and Xiaodong Li. I'm thankful for the years of camaraderie and friendship with Yaniv Plan. I'm also grateful to my undergraduate adviser, Francis Starr, who started me on my academic path.

My friends at Caltech have made my stay here very enjoyable. In particular, Sean Tulin and Justus Brevik were willing to be in close quarters with me for 3 weeks, so that says something about their friendship. My parents, Mary and Charlie, and brother, Andy, have always been supportive, and they have helped incredibly by shaping me as a person and always encouraging my academic pursuits.

Finally, I owe my happiness to my wife Kimberly. Thank you, Kim.

Abstract

Since 2004, the field of compressed sensing has grown quickly and seen tremendous interest because it provides a theoretically sound and computationally tractable method to stably recover signals by sampling *at the information rate*. This thesis presents in detail the design of one of the world's first compressed sensing hardware devices, the random modulation pre-integrator (RMPI). The RMPI is an analog-to-digital converter (ADC) that bypasses a current limitation in ADC technology and achieves an unprecedented 8 effective number of bits over a bandwidth of 2.5 GHz. Subtle but important design considerations are discussed, and state-of-the-art reconstruction techniques are presented.

Inspired by the need for a fast method to solve reconstruction problems for the RMPI, we develop two efficient large-scale optimization methods, NESTA and TFOCS, that are applicable to a wide range of other problems, such as image denoising and deblurring, MRI reconstruction, and matrix completion (including the famous Netflix problem). While many algorithms solve unconstrained ℓ_1 problems, NESTA and TFOCS can solve the *constrained* form of ℓ_1 minimization, and allow weighted norms. In addition to ℓ_1 minimization problems such as the LASSO, both NESTA and TFOCS solve total-variation minimization problem. TFOCS also solves the Dantzig selector and most variants of the nuclear norm minimization problem. A common theme in both NESTA and TFOCS is the use of smoothing techniques, which make the problem tractable, and the use of optimal first-order methods that have an accelerated convergence rate yet have the same cost per iteration as gradient descent. The conic dual methodology is introduced in TFOCS and proves to be extremely flexible, covering such generic problems as linear programming, quadratic programming, and semi-definite programming. A novel continuation scheme is presented, and it is shown that the Dantzig selector benefits from an exact-penalty property. Both NESTA and TFOCS are released as software packages available freely for academic use.

Contents

Chapter 1

Introduction

> "I'm disappointed in *Wired*, and perhaps in the good professor as well, for not displaying the self-control to decline publication of what reads as a classic red herring. Where are discussion and references to support this claim of getting something for nothing?"
>
> "Can you believe that someone would have such a fundamental misunderstanding of basic mathemetics [sic] and information theory that they would base a medical diagnosis on features produced by data interpolation? I hope it's not my doctor doing it. Prettying up pictures is great. Looking for tumors, etc. is insane. By definition, you're looking for an aberration, which, by definition, this algorithm would not produce."
>
> "Would you be willing to gamble your life on interpolated data where the shortcoming would be missing clinical pathology? A MRI image contains many subtle shades of gray in abstract shapes. For an artistic image clear, sharp edges and vivid colors may enhance, but to render a line on a diagnostic image that appears to abruptly end and restart may draw a complete artery where there is really an occlusion or to smooth out faint variations representing a brain tumor can be deadly."
>
> "Sounds great, but extraordinary claims demand extraordinary proof to distinguish them from hype."
>
> "You can't make something from nothing. You can create something by inference, but there's no certainty what you're making is right."
>
> "You won't be able, anyway, to save the data nobody ever entered... ."

The above quotes are from readers' comments on the online version of a 2010 *Wired Magazine* article [Ell10] about compressed sensing (CS). To sheltered academics in certain fields, compressed sensing is just another buzzword; these quotes serve to remind that there really has been great progress. This thesis can be viewed as an answer to these skeptical readers. The power of compressed

sensing techniques is highlighted, while maintaining that the theory really is intuitive (with the benefit of hindsight), since the basic tenet is that it is necessary to *sample at the information rate.*

To the several thousand academics just mentioned, the results of this thesis can be read and digested without causing much commotion. The same content, but in the year 2000, would have been met with astonishment; in the year 1985, incredulity. To be sure, beginning in the late 1980s, a small group of researchers had premonitions of what was possible, but the main notions behind compressed sensing were not widespread. What has changed since the 80s? Not just one major advance, but advances in many fields of science, math, and engineering. Namely:

- Signal processing and statistical advances, mainly in the ability to exploit sparsity. The results of compressed sensing in 2004 were fundamental [CRT06, Don06], although maybe not a complete surprise to a handful of statisticians since around 1988. The real significance of compressed sensing was a change in the very manner of thinking of many engineers and mathematicians. Instead of viewing ℓ_1 minimization as a post-processing technique to achieve better signals, CS has inspired devices, such as the RMPI system described in this thesis, that acquire signals in a fundamentally novel fashion, regardless of whether ℓ_1 minimization is involved.

- Computing power advances. For believers in Moore's law, this is no surprise, but nonetheless it is still impressive. The experiments in this thesis all require massive computation; so much, in fact, that before the development of the algorithms presented in Chapters 3 and 4, the simulations were performed on the massive Caltech SHC cluster.

- Circuits advances (also relating to the computing power). The design of the integrated circuit (IC) discussed in this thesis began in 2008, and the IC has now been fabricated in 90 nm CMOS technology; in 2011, there are even 32 nm technologies. Beyond just fabrication improvements, the design knowledge and design toolkits (such as powerful SPICE simulators) have increased. This allowed precise design of the IC. We note that we are still on the border of the feasible, since a full SPICE calibration of the system would take roughly a month of computational time on a multi-core workstation.

- Algorithmic advances. By 1997, with the publications of [Wri97,NN94], interior point methods (IPM) were a mature technology, and the powerful models of semi-definite programming [VB96] were being increasingly used. However, this recent decade has seen tremendous growth in data collection; as just one example, the Compact Muon Solenoid detector at the LHC at CERN produces 40 terabytes *per second* [The08]. Thus problems that were once "solved" have now been re-opened because these modern scientific problems are of such large scale. On the extreme side are tools like MapReduce and algorithms like Google's PageRank. For large—but not gigantic—datasets, a new theme is first-order methods. And, as always, parallel methods

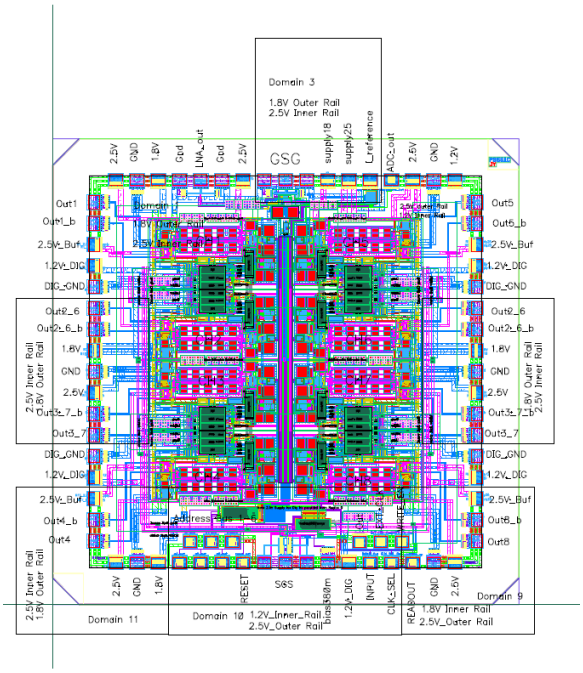


Figure 1.1: Diagram of the version 1 RMPI chip

are still crucial, with increasing attention on the GPGPU and on minimal-communication distributed computing.

This thesis brings together all of these advances and presents a state-of-the art system that bypasses a current barrier in ADC technology. The actual physical system is a receiver/ADC called the "Random Modulator Pre-Integrator" (RMPI); Figure 1.1 shows a layout of the actual integrated circuit. The general concept of the RMPI is not original to this thesis (see [SOS+05, KLW+06, LKD+07, TLD+10]), but it is one of the world's first working hardware devices to be built based on the compressed sensing paradigm. In designing the system, numerous engineering and mathematical problems were overcome, and we hope that the results presented in this thesis will be of use to others in the field.

The second component of the system is the signal-processing back-end, which involves many steps, but the heart of the recovery is solving an optimization program. This key computation is "just" a linear program (LP), or a second-order cone program (SOCP) for fancier variants, but we will show why existing algorithms are inadequate to deal with the large quantities of data, and how our proposed algorithms make the computation tractable.

Chapter 2 discusses the design of the RMPI receiver which was part of a larger "analog-to-information" (A2I) project. We take a basic CS idea, the RMPI, and from a high-level block diagram, turn it into a working radar receiver integrated circuit that has now been fabricated in 90 nm CMOS. The chapter explores both hardware decisions as well as signal processing tricks, but

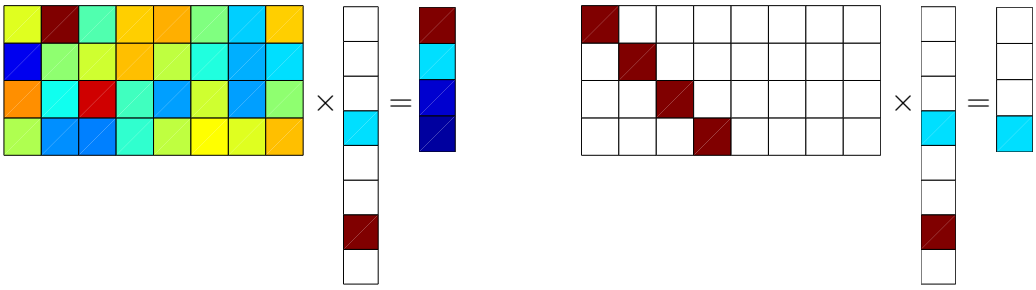


Figure 1.2: Democracy/incoherence in measurements. Pictoral representation of linear measurements $Ax = b$. Historically, good operators A were square and assumed to be easily invertible, such as the identity I or Fourier matrix $\mathcal{F}$. For *undersampling*, choosing A to be a partial identity matrix is bad, and recovery depends on luck. For sparse signals, it turns out that we can recover all of them if measurements are sufficiently global and incoherent. We informally refer to this idea as "democracy": each measurement provides the same amount of information as every other measurement, so there is no one special measurement without which reconstruction is impossible.

without treating the optimization problems in detail.

Chapter 3 presents an optimization algorithm "NESTA" that efficiently solves the signal-processing optimization problem for the RMPI. Chapter 4 presents an alternative optimization algorithm "TFOCS" that has some additional attractive features and is applicable to diverse problems.

The emphasis of this thesis is on the RMPI section, since this has not yet been published. The solvers (NESTA and TFOCS) have been previously described in publications [BBC11, BCG10a]. Chapters 3 and 4 are based on these publications, but contain some new material (see the chapter introductions for an overview of what is new). These chapters were written with co-authors, named in the chapter introductions.

The rest of this introduction presents some background material. The theory of compressed sensing is not discussed in much detail since there are already several excellent review articles and monographs [Can06, CW08, Bar07, FR11]; see also the resources at [RU]. Instead, we exploit the unconstrained nature of this thesis to elaborate a little on the history of some ideas that contributed to our current understanding of ℓ_1 processing, since it is rare that current literature has the luxury of going into background.

What compressed sensing is. A brief overview of the theoretical results is reserved for §1.2.4; we first mention the main ideas in an intuitive style. Compressed sensing (CS) addresses the situation in which there is an unknown mathematical object of interest that lies in a large ambient space, but because of prior information, the information content of the signal is less than the dimension of the space. The prior information need not be interpreted in the Bayesian sense; a more appropriate interpretation is that the signal is "compressible".

An implication of the premise that the signal is compressible is that sampling can be performed at the "information rate", or only marginally faster. This generally means that under-sampling is possible. As always, more samples improve the performance in the presence of noise.

The other key aspect of compressed sensing is the idea of incoherent measurements. Each mea-

surement of a signal should give some global information; see Figure 1.2 for a depiction. This comes at a price: when all the measurements are combined, the reconstruction algorithm is non-linear. In fact, CS theory says that the most efficient measurements look like a random sum, regardless of the reconstruction algorithm or specific signal. Think of the Voyager 1 spacecraft taking compressed measurements and sending them back to Earth; such measurements would allow modern techniques to extract as much information as possible, even using techniques such as wavelet analysis which did not exist at the time of Voyager's launch.

The *Wired Magazine* article failed to convey to its readers the importance of incoherence, hence the skepticism. To the readers, it seems as though subsampling drops information. But with incoherent measurements, every measurement gives a little bit of information, so the aberration tumor *is* measured, just not directly. However, compressed sensing does not give "something for nothing". In the case of noisy measurements, it is always helpful to take more measurements since this reduces the effect of the noise, and so under-sampling will always perform worse than exact- or over-sampling (assuming the same post-processing techniques).

What compressed sensing is not. Because compressed sensing deals with sparsity and compressibility, it is related to many other fields, such as sparse approximation, classic problems in image and signal processing such as denoising and deconvolution, dictionary learning, computational harmonic analysis, etc. In brief, these fields are related to the first ingredients of compressed sensing: sparsity and compressibility. But since they do not fundamentally involve incoherent measurements, they are distinct fields.

The architecture proposed in Chapter 2 is a pure compressed sensing architecture, because fundamental to its operation is the fact that measurements are incoherent. The optimization algorithms proposed in Chapters 3 and 4 are not limited to compressed sensing; they solve a wide class of problems which include CS problems, but also include problems from sparse approximation, machine learning, etc. In particular, they solve so-called ℓ_1 minimization problems, which have a broad applicability.

Perhaps one of the biggest impacts of CS is that it has spurred research in related fields, with the idea of exploiting prior knowledge. Yet the impact on hardware devices is much more limited: even though compressed sensing theory is about 7 years old and is quite well understood, there are very few pure compressed sensing applications. A few novel schemes are discussed in §1.1.

1.1 Applications

Before describing compressed sensing in detail, we motivate it with a few applications. Of course this thesis presents one such application, but the techniques of this thesis extend beyond this specific

case. As mentioned earlier, only true CS devices are mentioned, and thus we consider only hardware that takes incoherent measurements.

- **Optical and near-optical imaging.** The iconic "single pixel camera" developed at Rice University [WLD+06] is the best known compressed sensing device; it was singled out in 2007 as one of the top 10 emerging technologies by MIT Technology Review [Mag07]. The idea behind the camera is to trade spatial resolution for temporal resolution; for frequencies such as infrared, where each pixel is extremely expensive, this is a smart trade off. The camera uses a single pixel, but it takes many measurements over time. Each measurement needs to be different and also encode the entire scene; this is achieved by placing a micro-mirror array in front of a conventional optical lens system. The array is a grid of pixel-like mirrors called a digital micro-mirror device (DMD); each micro-mirror can be oriented so as to reflect light toward the single receiving pixel, or in some other direction. The effect is that of a binary mask.

  This is similar to the ideas used in coded aperture imaging in astronomy since the 1960s, except that due to sub-sampling, direct inversion is no longer possible. Coded aperture imaging uses a mask that is spatially moved in order to acquire a full set of data. Similar to radio interferometry, coded aperture is used as a means to increase spatial resolution, since radio telescopes' antenna have limited spatial resolution.

  Overall, these CS-based optical devices seem promising, and InViewCorp has recently mobilized to commercialize such a system. Similar ideas can also be applied to microscopy [WCW+09], where again DMDs are used to reduce the number of raster scans needed in confocal imaging.

  However, the subject is not yet closed, for there remain a few key difficulties. The first challenge is calibration. If the system can be modeled sufficiently accurately, then calibration is not an issue, but for high dynamic range acquisition, it is likely that the system will need to be characterized. Calibration at optical frequencies is challenging, since the phase is not as easy to control as in radio frequency (RF) systems.

  Another difficulty is related to the binary sensing matrix and the type of noise. The receiver sees a superposition of light from many sources for every measurement, and the photon count is greater than it would be in a multi-pixel setting. The DMD acts as a mask, and is equivalent to a matrix of zeros and ones. Because there are no negative numbers, the average of each row is necessarily greater than zero. It is this effect that causes the problem. Photons arrive according to a Poisson process, so there is "shot" noise. The distinguishing feature of this Poissonian shot noise is that the variance is equal to the mean. Because the single-pixel sensor sees many pixels worth of light, it consequently sees greater fluctuations about its mean. This appears to be a fundamental problem with no clever workaround. Bright point-like sources

can be resolved, but sources that have an even intensity are completely lost in the shot-noise and impossible to recover. This is currently the limiting factor in these systems.

- Medical Resonance Imaging (MRI). One of the "hot-topic" CS applications is MRI, which has seen much research, and it is also one of the first, with conference proceedings of the authors in [LDP07] going back to 2005. The work of Lustig [LDP07] has even been incorporated into the Stanford research hospital [Ell10], and current MRI manufactures are paying close attention to the field; the Siemens Corporate Research office has stayed active in the field of compressed sensing. MRI works by acquiring points in 2D or 3D k-space (i.e., Fourier space), and conventional MRI acquires specific grids of points so that the image may be reconstructed by the inverse Radon transform. Using sparse-approximation ideas, it is possible to sub-sample this grid and solve a linear inverse problem to recover the signal. CS applies to MRI by allowing k-space samples that are not on standard grids, and fewer samples than conventionally needed. Fewer samples leads to faster scans, which is significant since scans are performed on living, moving, people. However, the true potential of CS for MRI is that it might allow for non-standard types of measurements or certain types of systematic errors, such as non-linearities in the magnetic field, or weaker magnetic fields. To our knowledge, these types of breakthroughs have not yet happened.

- Microarray sequencing. Microarrays are used commonly in biology to identify specimens, using fluorescent tags to identify where samples bind; most samples have only a few active parts, thus using ideas from CS and group testing suggest that it is possible to take many fewer measurements and still accurately infer which specimens are present. Recent work in industry has suggested that this approach is useful in practice [MSS$^+$10].

- Seismic imaging. Because the Earth is made of discrete layers and so separated by sparse boundaries, seismic imaging has benefited from sparse recovery techniques since the 1970s. However, a true compressed-sensing-based seismic imaging system goes further and acquires images in a different fashion; for example, by changing the type of excitation signal sent by a ship or controlled explosion. Another method to apply compressed sensing is by controlling the rate and location of samples, much like the non-uniform sampler (NUS) introduced in §1.4.1. Undersampling is a fact of life for many seismic imaging systems; the system in [HH08] proposes that undersampling should be nearly at random, like the NUS, in order to turn coherent aliases into incoherent noise. They also suggest that the undersampling should not be completely at random; instead, the maximum separation between measurements should be controlled.

- Hyperspectral imaging. Hyperspectral imaging is the practice of acquiring images at many wavelengths (though usually not a continuum of wavelengths). It can be useful because it may

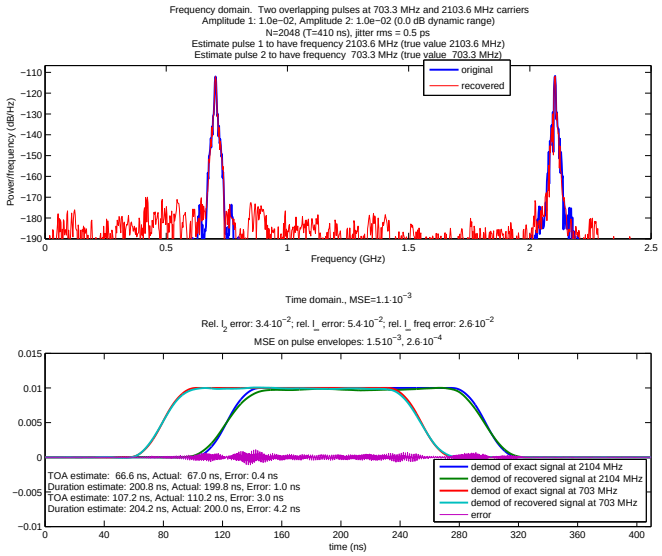


Figure 1.3: Preview: beating Nyquist with the RMPI. Reconstructing two pulses from 2.5 GHz of bandwidth using only 400 MHz sampling rate in a realistic simulation with noise and non-idealities.

ease identification; for example, it may aid a satellite imaging the ground, trying to determine the type of foliage on the surface. Images are typically compressible, and hyperspectral images even more so. See the work in [WGB07] for a proposed hyperspectral imaging system that uses a binary aperture mask (similar to the single pixel camera) with a multi-pixel array.

- Radar. Much like seismic imaging, it is possible to design special radar pulses to exploit CS; for example, [HS09] proposes sending a type of chirp called an Alltop sequence to probe targets, and shows that when there are only a few targets, this allows greater resolution than traditional radar. This is distinct from the radar problems discussed in this thesis. Typical CS radar applications design an emitter and a receiver; in this thesis, we are concerned with learning pulse characteristics from a foreign object's radar emitter, so not only is there no control over the emitted pulses, but the pulse parameters are unknown *a priori*.

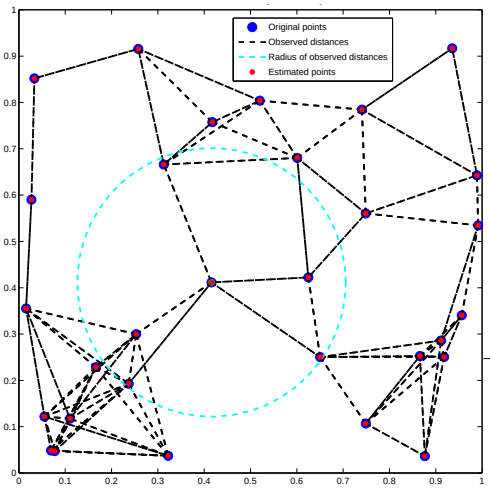
- CS has much potential for astronomy by extending what is possible with various types of coded aperture and, recently, DMD arrays. The benefits and problems are similar to those already discussed in the optical imaging section. It does not appear that a true CS-based astronomy device exists, but a CS-based coding scheme has been applied in astronomy [BSO08]. The Herschel satellite, launched on May 14, 2009, takes images with a photometer and needs to compress the images by a factor of 16 (in real-time) because of the limited rate of the downlink to Earth. Using lossless entropy coding, the data can be compressed by a factor of 2.5, but this is still insufficient. Because of the unique situation (low-powered sensor, but unlimited time and power for the receiver), [BSO08] proposed using a particular rectangular pseudo-

random matrix (a noiselet transform [CGM01]) to multiply groups of 6 images; importantly, this operation is easy to carry out in the satellite. The compressed data is sent to Earth and decompressed using ℓ_1 techniques or similar. The Herschel satellite adopted this scheme, so it is one of the first compressed sensing hardware devices, even if the image acquisition is still conventional. For more details of CS in astronomy, see [SB10].

- Quantum physics. In quantum computing and related fields, quantum states are prepared for specific purposes, and it is important to verify that an experiment has produced the expected state. The process of verifying a quantum state is known as quantum tomography. In an n qubit system, there are n particles but the spin of the system cannot be represented by a n-dimensional state vector due to entanglement. Instead, the system can be represented by a $n \times n$ Hermitian matrix. In a generic high entropy system, this matrix is rank n, but in specially prepared states, the matrix has low rank, or even rank 1. In a collaboration with quantum information physicists, this author worked on the problem of estimating the quantum state via subsampled measurements [GLF$^+$10]. Measurements are taken in an incoherent fashion; unfortunately, the best type of measurements are difficult to enact experimentally. Feasible measurements that are slightly less optimal are also proposed. The situation is interesting for several reasons. The first reason is that there is much experimental control over which measurements are taken, and no calibration is needed. It even allows the possibility of adaptive measurements. Secondly, measurements are inherently noisy because they are samples of a quantum wave function. Each measurement is really a Bernoulli random variable, so repeated measurements are binomial random variables. Thus the noise can be made arbitrarily small at the expense of requiring more time. Furthermore, in the experimental setup, there is a time penalty for switching to a new measurement, so it is advantageous to take repeated measurements of the same quantity. These tradeoffs make for interesting future study.

- ADC (Analog-to-Digital Converters). A compressed sensing ADC device is presented in Chapter 2 and also in §1.4, where history and background will be presented in detail, so we defer discussion. Figure 1.3 shows the recovery of two radar pulses from a realistic simulation that included non-idealities of the circuit.

Applications of sparse recovery. As pointed out, the above applications are true compressed sensing devices. The algorithms developed in Chapters 3 and 4 are designed for compressed sensing, but also for generic sparse recovery problems. Sparse recovery problems include topics mentioned in the CS applications section, as well as image and signal processing applications (interpolation, deconvolution, denoising, etc.; see Figures 1.5 and 1.6), and various applications in dictionary learning, blind source separation and morphological separation, and blind deconvolution. They also have ap-



	Denver	Chicago	Atlanta	Dallas	Seattle
Denver	0	?	591	398	?
Chicago	?	0	218	?	982
Atlanta	591	218	0	634	?
Dallas	398	?	634	0	521
Seattle	?	982	?	521	0

Figure 1.4: Preview: distance completion. In general, this is an NP-hard problem, but sometimes the convex relaxation is exact. The algorithm proposed in Chapter 4 can solve these minimization problems that involve matrix variables. This problem is like filling in the unknown "?" entries in the sample mileage chart on the right. The plot on the left shows a set of points for which about 29% of the pairwise distances are known (line segments represent known distances); from this, their true positions are recovered.

plications in many communications and network problems, and can be used to solve general statistics estimation problems. Thus our algorithms apply to almost all fields in the sciences. The algorithms can also solve certain matrix-variable problems in low-rank (and/or sparse) recovery. Such problems include matrix completion [CR09, Gro11] and various variants of robust PCA [CLMW09, MT10] (Figure 1.4). TFOCS can solve general semi-definite programs (SDP) [VB96]. Over the past 15 years, an increasing number of problems from engineering and control theory, as well as the fields mentioned above, have been cast in this framework. SDP relaxations of difficult problems such as MAX-CUT and matrix completion are also of high interest.

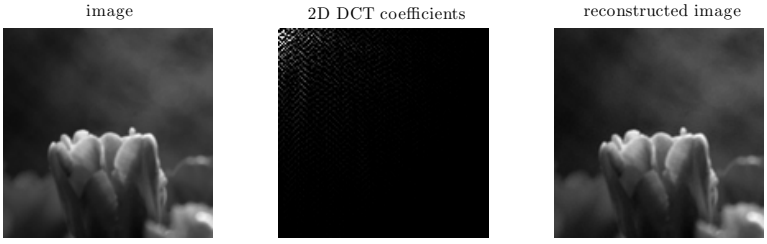


Figure 1.5: Compression. The image on the right is reconstructed from less than half the DCT coefficients of the original image on the left. It is an empirical fact that real images are usually compressible in the 2D frequency domain. JPEG uses an 8×8 2D DCT (in HSV color space) and quantizes coefficients, and has revolutionized the internet. The algorithms in Chapters 3 and 4 can be used for much more advanced image processing problems; see Figures 1.6 and 4.8.

1.1.1 Problems of interest

This section covers the formulation of some linear inverse problems. The meaning of the variables may change slightly in subsequent sections, but we aim for consistency of notation. In a linear

Figure 1.6: Preview: image denoising with TFOCS. For full details, see Figure 4.8. The image on the right is a denoised version of the noisy image on the left.

inverse problem, we seek an estimate $\hat{x}$ of a true signal x_0 given (possibly noisy) measurements of the formulation

$$b = Ax_0 + z, \quad x \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n} \tag{1.1.1}$$

where z is a noise vector (usually stochastic). On occasion we allow complex numbers, and this will be explicitly stated. The model $b = A(x + \tilde{z})$ also arises, which is equivalent to (1.1.1) if we set $z = A\tilde{z}$. The *linearity* of the problem means that A is a linear operator, and since this thesis only considers finite dimensional problems, A is a matrix. Estimates for $\hat{x}$ are not necessarily linear functions of A and b.

This thesis is concerned with the under-determined case, $m < n$, but the algorithms are also useful in the ill-conditioned case where $m = n$ but A is either singular or has large condition number. The focus is not on the $m = n$ case, and it is assumed that A has full column rank unless otherwise stated. Recovering an arbitrary vector x from a given A and b is then ill-posed because there are infinitely many solutions to this under-determined equation.

Consider temporarily a problem in which $m > n$, so the situation is over-determined. A standard approach (if z is Gaussian, this is the maximum likelihood estimator) is to set $\hat{x} = \operatorname{argmin}_x \|Ax - b\|_2^2$, which is the famous "least-squares" estimator made popular by Gauss. This can be written in closed form as $\hat{x} = A^\dagger b$ where $A^\dagger = (A^T A)^{-1} A$ is the Moore-Penrose pseudo-inverse of A. In the case that A is ill-conditioned, a classical technique is to regularize the problem (known as Tikhonov regularization or ridge regression):

$$\hat{x}_\gamma = \operatorname*{argmin}_x \|Ax + b\|_2^2 + \gamma \|x\|_2^2 = (A^T A + \gamma^2 I)^{-1} A^T b. \tag{1.1.2}$$

The second equality holds when γ is large enough that $A^T A + \gamma^2 I$ is invertible. The answer depends

on the choice of γ. If it is assumed that the signal x_0 comes from a normal distribution with mean 0 and variance $\sigma^2 I$ (which we will write as $x_0 \sim \mathcal{N}(0, \sigma^2 I)$) and $z \sim \mathcal{N}(0, I)$, then this is the maximum *a posteriori* (MAP) estimator if we choose $\gamma = \sigma$. In some contexts it may be possible to choose γ via cross-validation.

The ℓ_2 norm used in (1.1.2) is part of the class of ℓ_p norms, defined on a vector x as $\|x\|_p = (\sum_i |x_i|^p)^{1/p}$; so $\|x\|_1 = \sum_i |x|$ is just the sum of the absolute values. The ℓ_0 quasi-norm is not a norm: it just measures the number of non-zero entries of a vector.

We use Tikhonov regularization to motivate our first problem of interest, called the LASSO [Tib96] (Least Absolute Shrinkage and Selection Operator):

$$\hat{x} = \underset{x}{\operatorname{argmin}} \ \frac{1}{2}\|Ax - b\|_2^2 + \lambda \|x\|_1. \tag{1.1.3}$$

This is quite similar in form to Tikhonov regularization. From a Bayesian perspective, (1.1.3) is the MAP estimator if x has a Laplacian prior and z is i.i.d. Gaussian. At the solution $\hat{x}$, the first-order stationary condition [BV04] gives

$$A^T(b - A\hat{x}) \in \lambda \partial \|\hat{x}\|_1$$

where ∂f is the sub-differential of f [Roc70] (that is, the set of all sub-gradients). In particular, the absolute value of every entry of $\partial \|x\|_1$ is bounded by 1. If the original signal x_0 were to be a solution, then this gives the necessary condition that $\|A^T z\|_\infty \leq \lambda$. When z is a general stochastic vector, it is not expected that $\hat{x}$ is exactly x_0, but it turns out that $\lambda \approx \|A^T z\|_\infty$ is a reasonable value for the parameter. If $z \sim \mathcal{N}(0, \sigma^2 I)$, then with high probability $\|A^T z\|_\infty \leq 2\sqrt{\log(n)}$.

The key difference between the LASSO and Tikhonov regularization is that the ℓ_1 norm has a sharp discontinuity at 0; see Figure 1.7.

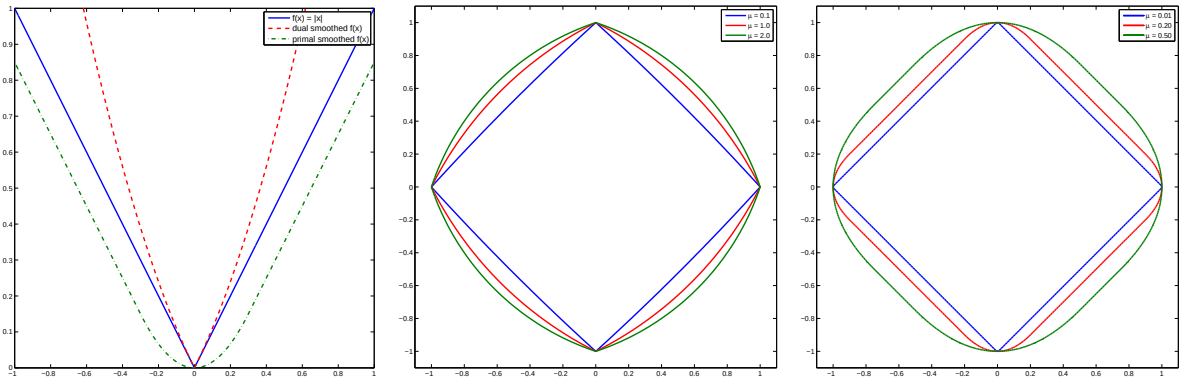
A closely related problem is the following:

$$\underset{x}{\text{minimize}} \ \|Ax - b\|_2^2 \quad \text{such that} \quad \|x\|_1 \leq \tau. \tag{1.1.4}$$

Confusingly, this is the version suggested by Tibshirani in his original LASSO paper [Tib96] so it also goes by the name "LASSO". To keep the variants clear, we will stick with the λ or τ parameters, and refer to the first problem as the "unconstrained LASSO". The unconstrained LASSO has seen much more attention in the literature than the ℓ_1 constrained LASSO.

Another variant is known as basis pursuit (BP) [CDS98], which is the linear program (LP):

$$\underset{x}{\text{minimize}} \ \|x\|_1 \quad \text{such that} \quad Ax = b \tag{1.1.5}$$



(a) The absolute value function (blue), a primal smoothed version known as the Huber function which will be used in Chapter 3, and a dual smoothed version which will be used in Chapter 4

(b) Contours of the dual smoothed absolute value function, for various levels of smoothing

(c) Contours of the Huber function for various levels of smoothing

Figure 1.7: The ℓ_1 norm promotes sparsity because of the kink at 0. Any function which is quadratic near zero (e.g., least-squares norm, or the Huber function [Hub64]) does not promote sparsity because of the "diminishing returns" in pushing a coefficient to zero. The dual-smoothed version keeps this feature. Unfortunately, the non-smoothness of ℓ_1 also makes it difficult to work with; only the Huber function has a continuous derivative. See also Figure 4.1

which always has the same value as its dual problem

$$\underset{\nu}{\text{maximize}}\ b^T\nu \quad \text{such that} \quad \|A^T\nu\|_\infty \leq 1.$$

This is mainly used when $z = 0$. For noisy cases, we turn to basis pursuit denoising (denoted BPDN or BP_ε), which is a second-order cone program (SOCP) [BV04]:

$$\underset{x}{\text{minimize}}\ \|x\|_1 \quad \text{such that} \quad \|Ax - b\|_2 \leq \varepsilon \tag{1.1.6}$$

and its dual

$$\underset{\nu}{\text{minimize}}\ \|\nu\|_2 - b^T\nu \text{ such that } \|A^T\nu\|_\infty \leq \varepsilon.$$

This dual problem has the same value as the primal problem when A has full row-rank and $\varepsilon > 0$ (see the Slater conditions [BV04]). BPDN is sometimes referred to as the LASSO as well.

BPDN will be the focus of the optimization algorithms introduced in later chapters. A basic result of convex optimization [Roc70, BNO03] is that unconstrained LASSO, the ℓ_1 constrained LASSO, and BPDN are all equivalent for some values of ε, τ, and λ (as long as these are all non-zero and finite). That is, given ε, there is some $\lambda(\varepsilon)$ such that the unconstrained LASSO estimator is the same as the BPDN estimator. In general, calculating $\lambda(\varepsilon)$ is not easy, and requires at least as much computation as solving the estimation problem; for example, to calculate $\varepsilon(\lambda)$, we set $\varepsilon = \|A\hat{x}_\lambda - b\|_2$ which requires knowing $\hat{x}_\lambda$.

The value of ε is typically chosen by $\varepsilon \simeq \mathbb{E}\,\|z\|_2$. If the entries of z are independent and have a finite second moment (and zero first-moment), this is just the square root of the sum of the variances. Methods for estimating variances, using the sample variance or a robust statistic like the median absolute deviation, are well-understood, so it is quite reasonable to assume that a reasonable value of ε is known. In practice, some researchers may prefer to estimate ε rather than λ; this was the motivation behind NESTA's development.

The final canonical problem we discuss is the Dantzig selector [CT07a], which motivated the development of the TFOCS algorithm. The Dantzig selector is the solution to

$$\underset{x}{\text{minimize}} \ \|x\|_1 \quad \text{such that} \quad \|A^T(Ax-b)\|_\infty \leq \delta. \tag{1.1.7}$$

The parameter δ should be about the same as the parameter λ from the unconstrained LASSO. The Dantzig selector is used for the same types of problems as BPDN, and enjoys similar theoretical properties. Indeed, the differences between the two was the subject of much discussion in the Annals of Statistics (see [EHT07, CT07b] and related papers in the same issue).

Lastly, we present two variants of BPDN that are useful when a signal x_0 is compressible in some basis or dictionary Ψ; these variants can also be applied to the LASSO and Dantzig formulations. Suppose $x_0 = \Psi\alpha_0$ (this is useful when α_0 is sparse or has decaying coefficients), and take measurements $b = \Phi x_0 + z = \Phi\Psi\alpha_0 + z$. There are two ways to use BPDN. The first is known as "synthesis":

$$\underset{\alpha}{\text{minimize}} \ \|\alpha\|_1 \quad \text{such that} \quad \|A\alpha - b\|_2 \leq \varepsilon, \quad A \triangleq \Phi\Psi \tag{1.1.8}$$

and the second is "analysis":

$$\underset{x}{\text{minimize}} \ \|\Psi^T x\|_1 \quad \text{such that} \quad \|\Phi x - b\|_2 \leq \varepsilon. \tag{1.1.9}$$

Since we are often interested in solving the analysis problem, it is convenient to allow BPDN (and the other variants) to use weighted ℓ_1 norms; we often write W for the weighting matrix, e.g., $W = \Psi^T$. The differences between synthesis and analysis are discussed in §2.7.2.1, §3.6.1, and §4.7.5.

1.2 Historical development

Here, we present a selective history of some key developments leading to compressed sensing, as well as some classic signal processing results.

1.2.1 Classical signal processing

We start with the following fundamental theorem.

Theorem 1.2.1 (Shannon-Nyquist-Whittaker, this form modified from [Mal08]). *Let $x(t)$ be a band-limited signal with frequencies inside the band $[-B, B]$ (in Hz). Then*

$$x(t) = \sum_{n=-\infty}^{\infty} x(n\Delta T)\phi_s(t - n\Delta T)$$

where $f_s \triangleq 2B$ is the "Nyquist rate" or "Nyquist frequency", $\Delta T \triangleq 1/f_s$, and $\phi_s = \frac{\sin(2\pi f_s t)}{2\pi f_s}$ is a scaled sinc *function.*

The theorem is obvious using the machinery of the inverse Fourier transform and convolutions, and noting that the Fourier transform of a *sinc* is a boxcar function; this may explain why it was discovered independently by several people in the first half of the 20th century. In [ME09a], it is referred to as the WKS theorem, where the K stands for Kotelńikov.

In plain language, Theorem 1.2.1 states that if a signal has bandwidth B, then it is completely specified by sampling at rate $f_s = 2B$. For an arbitrary bandlimited signal, this sampling rate is also necessary, otherwise aliasing occurs and the original signal cannot be reconstructed from the samples. Sampling theorems such as these are important since most algorithms manipulate digitized signals, and digitizing an analog signal requires sampling. The bandlimited assumption is quite natural since most receivers and emitters have finite bandwidths (or more accurately, nearly finite bandwidths: at very high frequencies, signals are attenuated so much that they can be neglected).

Suppose now that a bandlimited signal only has small spectral occupancy. If these are in a few continuous regions, and are known, then down-conversion tricks (see §1.3) show that the sampling rate only needs to be at twice the amount of occupied bandwidth. A more precise, general and robust result is due to Landau [Lan67] from the 1960s. In the following, $\mathcal{F}$ denotes the Fourier transform.

Theorem 1.2.2 (Landau sampling theorem; this form modified from [ME09a]). *Define*

$$\mathcal{B}_\Omega = \{x(t) \in L^2(\mathbb{R}) \mid \operatorname{supp}(\mathcal{F}x(f)) \subset \Omega\}.$$

A set of time samples $R = \{r_n\}$ is a sampling set *for $\mathcal{B}_\Omega$ if the sequence of samples $x_R[n] = x(r_n)$ is stable; stability means $\exists \alpha > 0, \beta < \infty$ such that*

$$\alpha\|x - y\|^2 \leq \|x_R - y_R\|^2 \leq \beta\|x - y\|^2 \quad \forall x, y \in \mathcal{B}_\Omega.$$

Then a necessary condition for R to be a sampling set for $\mathcal{B}_\Omega$ is

$$D^-(R) \triangleq \lim_{t\to\infty} \inf_{y\in\mathbb{R}} \frac{|R \cap [y, y+t]|}{t} > |\Omega|$$

where $|\cdot|$ is the Lebesgue measure. The term D^- is known as the lower Beurling density.

For example, if R takes uniform samples with spacing ΔT, then $D^- = 1/\Delta T$ which is the average sampling density. If in addition $\Omega = [-B, B]$, then the theorem requires $1/\Delta T > 2B$, which is just the Shannon-Nyquist theorem.

Note an interesting consequence of the theorem: if $x(t) = \sin(2\pi f t)$ is a pure tone, then $|\Omega| = 0$ so the theorem does not provide a minimal necessary sampling rate to recover this signal. In fact, any periodic signal has a zero measure frequency support, and hence the theorem provides no bound.

For practical recovery from such a model, and sampling at the minimal rate, it is necessary to know Ω. An example of a related compressed sensing result is due to [ME09a], which, to paraphrase, says that if the frequency support Ω of X is *unknown* but $|\Omega|$ is bounded and it is known that $\Omega \subset [-B, B]$, then a necessary condition for a sampling set that stably recovers all signals in this model is that its lower Beurling density must be twice the value it would have been had Ω been a fixed, known set.

1.2.2 Estimation and the rise of alternatives to least-squares

Least-squares is the quintessential linear estimation technique; the ordinary least-squares (OLS) solution to an overdetermined system $\min_x \|Ax - b\|_2$ is $\hat{x}_{OLS} = A^\dagger b$ which is linear in b. Starting with at least Gauss, the nice properties of least-squares estimation (smoothness, convexity, closed-form expressions) have been much touted.

Particularly because of the rise of computers in the 20th century, there have been an increasing number of alternatives to linear methods. These alternatives have accumulated gradually but steadily: Dantzig's simplex algorithm in the '40s, message passing and belief propagation used for decoding (Viterbi's algorithm in the '60s), and graphical models used for inference in the past two decades, to name just a few. Linear estimation is certainly still a cornerstone technique and has produced successes like the Kalman filter. Below is a brief and partial history of some ideas that form a foundation for modern compressed sensing. Most of these are related to processing sparse signals and using ℓ_1 techniques; the idea of using incoherent measurements has much less precedent, with perhaps the exception of coded aperture receivers.

In 1965, Logan's dissertation [Log65] proved a result now known as "Logan's phenomenon": if a continuous-time band-limited signal $x(t)$ is corrupted by noise z and the noise has bounded L_1 norm and has a very sparse time support (but the support is unknown), then finding the minimal L_1 norm projection of the corrupted data can *perfectly* recover the original signal x. This is remarkable,

because it is independent of the magnitude of the error.

The discrete version of this result, using the ℓ_1 norm, was empirically demonstrated by Santosa and Symes [SS86] in 1986, and proved in [DS89]. The work of [SS86] proved that recovery of a k-sparse vector from just m Fourier measurements was possible if $m \geq (1 - 1/(2k))n$, using the unconstrained LASSO formulation (they also note the pessimistic nature of this bound). Even by 1986, the fact that "it is possible to construct a sparse spike train from part of its spectrum using the minimum ℓ_1 criterion" was "well-known (and believed)" [SS86].

The 1989 article by Donoho and Stark [DS89] extended Santosa and Symes' results by proving a more general new type of uncertainty principle (see also [DL92]). The classical time-frequency uncertainty principle says that a signal cannot have small support in both time and frequency. The [DS89] result extended this result by relaxing the restriction that a signal's support be on intervals. To be concrete, let x be a discrete signal of length n with discrete Fourier transform X. Donoho and Stark prove

$$\|x\|_0\|X\|_0 \geq n \quad \Big(\text{and hence,} \quad \|x\|_0 + \|X\|_0 \geq 2\sqrt{n}\Big). \tag{1.2.1}$$

Intuitively, the result says that there are not many discrete signals that are sparse in both time and frequency. Thus, if a signal happens to be sparse in both time and frequency, it has no "neighboring" signals, and for this reason it is easy to recover. The multiplicative inequality is sharp, because if $n = d^2$, then a Dirac comb of spacing d has $\|x\|_0 = \|X\|_0 = d$; this example is also sharp for the additive identity (the additive identity follows from the multiplicative one by the arithmetic-geometric mean inequality). Another example is when x is a single Fourier element, in which case $\|x\|_0 = n$ and $\|X\|_0 = 1$, so the multiplicative inequality is sharp but the additive inequality is not. It turns out the additive inequality is only sharp in cases such as $n = d^2$; the situation when n is prime is quite different [Tao05].

[DS89] also reports numerical experiments suggesting that much stronger results are possible if the sparsity is "scattered" in a random way, so it is clear that the authors had some insight into what was achievable; their official bound on measurements was the same as Santosa and Symes. It is worth mentioning that their numerical experiments went only as large as $n = 256$, and it is not easy to distinguish $\mathcal{O}(n)$, $\mathcal{O}(\sqrt{n})$, and $\mathcal{O}(\log n)$ growth from experiments with just $n = \{64, 128, 256\}$.

This theory in the late 1980s built on empirical work from the previous 15 years, as different scientific communities were using ℓ_1 minimization and/or sparsity. The work in [CM73] in 1973 argues that the geophysics community should consider ℓ_1 regularized problems for the sake of robustness to outliers, using the mean and median estimators as examples. They also argue that ℓ_1 generates sparsity, but only to the extent that there is always a vertex solution to a linear program (so sparsity of a solution can be chosen less than m). In addition to sparsity in space, they give

examples of using sparsity in first- and second-differences; this was further explored in the highly influential total-variation (TV) minimization paper in 1992 [ROF92].

Similar works in geophysics, such as [TBM79], cite ℓ_1 studies going back to 1964. Incremental results continued in geophysics, exemplified by [OSK94] in 1994 which cites numerous ℓ_1 results from the 1980s (but appears to be unaware of [SS86, DS89]). Many of these works proposed special algorithms to avoid recasting the problem as a linear program (LP).

In radio astronomy, the CLEAN algorithm [Hög74], introduced around 1971, exploits sparsity of radio sources and is similar to a matching pursuit algorithm [MZ93] as shown by [WJP+09], though it is not equivalent to ℓ_1 minimization (indeed, the first analysis of CLEAN [Sch78] compares it to ℓ_2 minimization). Its idea of exploiting sparsity and post-processing has been extremely useful in astronomy; to quote from [Cor09], "The impact of CLEAN on radio astronomy has been immense. First, there is the accumulated science from the telescopes that have used CLEAN—GBI, MERLIN, WSRT, VLA, VLBI, etc. ... Second, by showing what could be achieved with some postprocessing, CLEAN has encouraged a wave of innovation in synthesis processing that continues to this day."

The article [PC79] introduces the authors' algorithms from 1975 to the application of estimating the transfer function of an unknown medium using ultrasound pulses, and exploits sparsity in the number of layers in most mediums. The algorithm they propose is similar to alternating projections between the time domain (which is sparse; this set is not convex) and the frequency domain (where it is assumed that reliable data exists in some band). The 1982 article [ME82] discusses an application of ℓ_1 minimization for improving diffraction-limited images, and also suggests its use in video compression.

A noticeable similarity in all these works is that they almost exclusively work with deconvolution, and hence operate in the Fourier and time/spatial domains.

1.2.3 Leading up to compressed sensing

Another key milestone in the work preceding compressed sensing was a paper by Donoho and Huo in 1999 [DH01], and was followed by several papers in the next five years, including [DE03, EB02, GN03, Tro04, DET06, Tro06]. These results concern sparse approximation: suppose a signal $x \in \mathbb{C}^n$ can be represented as $x = \Psi\alpha$ where $\Psi = (I, \mathcal{F})$ contains "spikes and sines" ($\mathcal{F}$ is the DFT). Since Ψ is rectangular, there are infinitely many α which satisfy this. The result of [DH01] is that if there is some $\|\alpha_0\|_0 < \frac{1}{2}\sqrt{n}$ and $x = \Phi\alpha_0$, then ℓ_1 minimization

$$\min_{\alpha} \|\alpha\|_1 \quad \text{such that} \quad x = \Phi\alpha$$

will return α_0. Similar results hold for solving by Orthogonal Matching Pursuit (OMP; see §1.5) [Tro04]. These results were generalized to other bases using a convenient tool called the coherence (1.2.2).

The result means that a signal $\alpha \in \mathbb{C}^{2n}$ of length $N = 2n$ which is k sparse, with $k < \frac{1}{2}\sqrt{n}$, can be recovered from $m = n$ entries. In terms of the dimension N and measurements m, the sparsity must obey $k < \frac{1}{\sqrt{2}}\frac{m}{\sqrt{N}}$. This $\sqrt{N}$ sparsity may not seem very sparse to readers with knowledge of recent results, who might expect that sparsity can grow with rate $k \simeq \frac{m}{\log N}$, which only weakly depends on N. But as proved in [DH01], these results are sharp. Other than special cases (such as n prime; see [Tao05]), there are counter examples showing that better bounds cannot be obtained, such as with the Dirac comb. So how does compressed sensing improve these bounds? The key difference is that compressed sensing bounds typically do not hold for *all* measurement matrices, but rather hold with overwhelming probability[1] or with high probability[2].

1.2.4 Compressed Sensing

This section gives a brief review of basic CS theory; for more details, the short review article [CW08] is recommended.

Let a signal $x(t) = \sum_{i=1}^{n} \alpha_i \psi_i(t)$ where Ψ is the matrix with columns ψ_i. We deal mainly with discrete, finite-length samples, so $x \in \mathbb{R}^n$, and Ψ is a $n \times n$ matrix, typically chosen to be orthogonal. For example, the signal x might just be a sparse vector, so then Ψ can be the identity matrix I.

Let Φ be another orthogonal $n \times n$ matrix; we will eventually subsample this to create the $m \times n$ "sampling matrix"; outside of the introduction, it is always assumed that Φ is $m \times n$ with $m < n$, and except for the NESTA algorithm in Chapter 3, the rows of Φ will not need to be orthogonal. Since Φ and Ψ are both orthogonal, and therefore invertible (and well-conditioned), it is possible to recover x (or equivalently, α) from samples

$$b = \Phi x = \Phi\Psi\alpha, \quad \text{or} \quad b = A\alpha, A \triangleq \Phi\Psi.$$

So far, this is not remarkable, and there is no benefit to doing so (even in the noisy case, since Gaussian white noise is unchanged under orthogonal transformations).

The remarkable results from compressed sensing say that if α is sparse (we typically use k to denote the sparsity, and say that α is "k-sparse") and Φ and Ψ are incoherent, then it suffices to only take a little more than k rows of the Φ matrix in order to stably reconstruct x.

Define the *coherence* of Ψ and Φ to be

$$1 \leq \mu(\Phi, \Psi) \triangleq \sqrt{n} \max_{1 \leq k,j \leq n} |\langle \psi_k, \phi_j \rangle| \leq \sqrt{n}. \tag{1.2.2}$$

The two ortho-bases are "incoherent" when $\mu \simeq 1$ or has a weak-dependence on the dimension n.

[1] Used here, this is a technical term (see [Tao11]); if a statement or event E_n depends on n, then E_n holds *with overwhelming probability* if $\forall \alpha > 0$ we have $\forall n, \mathbb{P}(E_n) \geq 1 - c_\alpha n^{-\alpha}$ where c_α is a constant independent of n; in compressed sensing, n is the dimension of the signal.

[2] Similarly, we say E_n holds *with high probability* if $\exists \alpha > 0$ such that $\mathbb{P}(E_n) \geq 1 - cn^{-\alpha}$.

Complete coherence, $\mu = \sqrt{n}$, is obtained if, for example, $\Psi = \Phi$. Maximal incoherence, $\mu = 1$, is obtained for $\Psi = F$, the Fourier basis, and $\Phi = I$, the identity. One remarkable fact is that for *any* fixed basis Ψ, if Φ is chosen from the general orthogonal ensemble (GOE), the coherence of Φ and Ψ is $\sqrt{2\log n}$ with high probability. Thus random matrices Φ are universally good measurement matrices. Intuitively, we want low coherence so that each row of Φ takes an equal amount of information about the signal. If Φ and Ψ are very coherent, then each row of Φ only tells us how much of one particular basis element ψ_i is present in the signal, and thus we would need all n measurements, since leaving out the ith measurements is taking the very risky gamble that $\alpha_i = 0$. See Figure 1.2 on page 4 for a graphical depiction.

An early result [CR07a] is that if

$$m \geq C\mu^2(\Phi, \Psi)k \log n$$

then if α is k-sparse and Φ_m is Φ restricted to m rows chosen uniformly at random, then with overwhelming probability, the measurements $b = \Phi_m x$ are sufficient to *tractably* recover x. The recovery is tractable because it can be done by solving basis pursuit (1.1.5), which can be solved in polynomial time. Here, C is a constant that is independent of k and n.

Very recent results [CP10a] have extended this, which we briefly paraphrase. For simplicity, let A incorporate both Φ and Ψ, $A = \Phi\Psi$. Write the rows of A as a_i^T, and assume each row is drawn iid from some probability distribution P, and that each row has zero mean and identity covariance matrix (so in particular, all the entries $a_{i,j}$ of A are uncorrelated, and furthermore entries from different rows are independent). Define the self-coherence to be the smallest number μ_s such that for all rows a_i,[3]

$$\max_{1 \leq j \leq n} |a_i(j)|^2 \leq \mu_s. \tag{1.2.3}$$

If P is a sub-Gaussian distribution, then $\mu = c \log n$ for some c independent of n, and if P is sub-exponential, $\mu = c \log^2 n$.

From now on we'll assume x is the sparse object, e.g., $x = \alpha$. The following theorem differs from other compressed sensing results in that we assume x is a *fixed* k-sparse vector. This is a stronger assumption than before because for any given realization of a sampling matrix A, we can no longer pick an adversarial signal (and one might argue that since adversarial signals rarely arise in practice, this stronger assumption is palatable).

Theorem 1.2.3 (due to [CP10a])**.** *Let x be a fixed k-sparse vector, and take m measurements $b = Ax$. If $m \geq (1+\beta)C\mu_s k \log n$ then with probability $1 - 5/n - e^{-\beta}$, basis pursuit* (1.1.5) *will recover x.*

[3] This need not hold deterministically; see [CP10a] for details.

This can be extended to cover compressible signals and noisy measurements. Let x_k denote the best k-term approximation to x; in loose terms, we say x is compressible if there is a small k such that $\|x - x_k\|_2$ is small. In particular, if x is k-sparse, then $x = x_k$. We present a simplified theorem below:

Theorem 1.2.4 (due to [CP10a]). *Let $x \in \mathbb{R}^n$ be arbitrary; in particular, it need not be sparse. Let $y = Ax + z$ where $z \sim \mathcal{N}(0, \sigma^2)$, and define $\lambda = 10\sigma\sqrt{\log n}$. Then if*

$$m \geq (1 + \beta)C\mu_s k \log n$$

then with probability at least $1 - 6/n - 6e^{-\beta}$, the LASSO (1.1.3) will recover an estimate $\hat{x}$ satisfying

$$\|\hat{x} - x\|_2 \leq \tilde{C}(1 + \log^2 n)\left(\frac{\|x - x_k\|_1}{\sqrt{k}} + \sigma\sqrt{\frac{k \log n}{m}}\right).$$

Identical results hold for the Dantzig selector (1.1.7).

To make these results meaningful, we compare to an "oracle". Similarly to how an unbiased estimator's performance can be compared to the optimal performance given by the Cramér–Rao bound, comparing with an oracle gives an idea of how close this performance is to the optimal performance. Consider recovery of a k-sparse vector x given noisy observations $y = x + \sigma z$, where z is any noise vector with independent entries and unit variances. Suppose an oracle whispers in your ear that the nonzero entries of x are precisely the first k. Then a quite reasonable strategy (though not optimal in terms of mean-square error, as Willard James and Charles Stein famously showed) is to take $\hat{x}_i = y_i$ for $i \leq k$ and $\hat{x}_i = 0$ otherwise; in fact, with this information, we only need to take k measurements. The expected error is $\mathbb{E}\,\|\hat{x} - x\|^2 = k\sigma^2$, and this is referred to as the oracle error.

The oracle takes k measurements and the error is $k\sigma^2$. The above theorem says that without the benefit of knowing *a priori* which entries are nonzero, then if we take slightly more measurements (by a factor of about $\log n$), we can do almost as well, with an error that is worse by a factor of about $\log^5 n$. Theorem 1.2.6 will improve this to a constant factor, and in Chapter 2 we show empirically that this factor is about 3.

For completeness, we cover what is now the *de facto* "standard model" of compressed sensing, which relies on a tool called the RIP:

Definition 1.2.5 (RIP (Restricted Isometry Property)). *The restricted isometry constant δ_k of order k for a matrix A is the smallest number such that*

$$(1 - \delta_k)\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1 + \delta_k)\|x\|_2^2$$

holds for all k-sparse vectors x.

For example, if A is an orthogonal matrix, then A is an isometry, and $\delta_k = 0$ for any k, which is the best possible constant. If A is $m \times n$, then it has a non-trivial nullspace, so $\delta_k \geq 1$ for $k \geq m$, but it is still possible for $\delta_k < 1$ for $k < m$. In general, adding rows to a matrix A will improve (i.e., decrease) its RIP constant. The literature often refers informally to a "matrix having the RIP" whenever that class of matrices satisfies $\delta_{\gamma k} \ll 1$ with high probability whenever $m \simeq k \log(n)$, where γ is typically 2 or 3. There are many variations of results using the RIP, but we show one of the most compact results.

Theorem 1.2.6 (due to [Can08]). *Let $\delta_{2k} < \sqrt{2}-1$ and $b = Ax + z$ for any (possibly deterministic) z, with $\|z\|_2 \leq \varepsilon$. Then basis pursuit denoising* (1.1.6) *with this ε gives an estimator $\hat{x}$ satisfying*

$$\|\hat{x} - x\| \leq C_0 \frac{\|x - x_k\|_1}{\sqrt{k}} + C_1 \varepsilon$$

for some constants C_0 and C_1 (which are not unreasonably large).

The theorem is deterministic, but unfortunately, there are few deterministic matrices which have small RIP constants. For a fixed number of rows m, random matrices have the best known RIP constants (although these bounds are not deterministic, and hence there is a nonzero chance of failure). The bound $\delta_{2k} < \sqrt{2}-1$ can be slightly improved, but this has little qualitative significance to our study here.

There are different bounds for random matrices, but we present here a paraphrase of an elegant result that sums up the general idea:

Theorem 1.2.7 (due to [BDDW08]). *Let the entries of the $m \times n$ matrix A be drawn i.i.d. according to either a zero-mean Gaussian or $\{+1, -1\}$ Bernoulli distribution, and scaled so the expected value of each column is 1. For any k and δ, with overwhelming probability (which depends on δ) the matrix A will satisfy $\delta_k(A) \leq \delta$ as long as*

$$m \geq Ck \log(n/k)$$

for some constant C independent of n (but dependent on δ).

Note that there is a qualitative difference between the $\{+1, -1\}$ Bernoulli matrices (also known as signed Bernoulli, or Rademacher) and the $\{0, 1\}$ Bernoulli matrices in their compressed sensing performance; the simple shift makes a big difference. In fact, $\{0, 1\}$ matrices do not obey the RIP unless m scales with either k^2 or n (as opposed to $k \log n$) [Cha08]. This does not mean that the matrices are useless for CS, rather just that RIP-based techniques fail; for example, types of expander graphs [JXHC09] fail the RIP but still have probable recovery results. However, in some applications, non-negative measurement matrices induce problems; for example, the micro-mirror optical systems that work in either on- or off-states are affected by this non-negativity due to an increase in noise; see the discussion in §1.1.

Figure 1.8: Group testing. Suppose N soldiers are tested for a disease using a sensitive but expensive blood test; if blood from different soldiers is grouped together, the test is sensitive enough to declare a "positive" if any of the contributed blood samples were positive. If k soldiers are infected, how many measurements M are necessary? If k is large, then $M = N$ is the best we can do. If $k = 1$ (and k is known *a priori*), then $M = \log_2(N)$ is possible (with or without adaptive measurements). The key idea is to take global measurements and then use reasoning. In this pictorial example, soldier # 45 tests positive, and 45 is 101101 in binary, so using the binary tests depicted by the gray shading, and assuming $k = 1$, we can deduce that this is the infected soldier. Each row represents one test, which involves the columns that are gray. See [GS06a] for more info. This was first proposed in the 1940s by two economists helping the military screen draftees for syphilis; their proposal was to group together the blood of five men and apply the test. Since few men were likely to have syphilis, this saves tests on average. The test was never put into practice, in part because the test was barely sensitive enough to detect a positive hit from diluted blood. The subject was revived again in the late 1950s.

Another useful result of CS is that random rows of a discrete Fourier matrix also work well for sampling when the signal is sparse. This is significant since the fast Fourier transform allows one to compute measurements faster than simply taking a matrix-vector product. The number of required measurements is similar to that of the Gaussian ensemble case, except the $\log(n)$ factor is $\log^5(n)$; this bound may be lowered in the future, so it is not necessarily indicative that Fourier matrices are inferior to Gaussian matrices in practice.

To summarize the intuition behind compressed sensing: if a signal has a sparse representation, then typically only $\mathcal{O}(k \log n)$ measurements are necessary (this is an intuitive number: for each of the k nonzeros, we need some constant number of measurements to encode the amplitude, and $\log n$ number of measurements to encode the location of the entry).

The other intuition is that for a noisy signal, the undersampling hurts us, but only by an amount proportional to m/n. This is a fundamental limitation: to accurately estimate a quantity given noisy measurements (say, estimate the population mean given the sample mean), it is always beneficial to take more measurements.

Our final remark is that because random sensing matrices have good properties, CS tells us they are universal encoders. Furthermore, the measurements are not adaptive. Clearly, well-chosen adaptive measurements are not detrimental, but perhaps they are not as beneficial as one might expect. For example, consider the case of group testing [GS06a, DH93] depicted in Figure 1.8, with n samples and k samples that test "positive". We wish to find the k positives. If $k = 1$ and k is known, then one obvious solution is an adaptive binary tree scheme (aka bisection method) which will take $\log_2 n$ measurements. But a non-adaptive scheme that groups the measurements according to their binary representation also takes $\log_2 n$ measurements, so there is no advantage to the adaptive scheme. If k is larger and unknown, then the adaptive scheme is better, but not by too much.

Alternatives to RIP. The RIP is a convenient tool, but so far it has not proven to be the perfect tool to analyze deterministic matrices. Most undesirably, it is generally NP-hard to check if the RIP holds for a specific matrix. Furthermore, the RIP is not a necessary condition. It is clearly a useful tool, but because of the importance of CS, there has been work on alternatives. We mention just a few of these alternatives here, in addition to the coherence property which was already discussed.

The work in [BGI+08] introduces a simple extension of the RIP to the so-called RIP-p, and in particular the RIP-1. The RIP-p is stated as the RIP is, except using ℓ_p norms instead of the ℓ_2 norm; thus RIP-2 is the standard RIP. The RIP-1 is particularly useful for expander graphs [JXHC09] and other sparse encoding matrices. Another extension of the RIP is model-based RIP [BCDH10] which concerns all x restricted to a specific model; if the model is the set of k-sparse signals, then this is just the usual RIP. A variant on this is the restricted-amplification property [BCDH10].

Another approach is that taken by Donoho in his original paper [Don06]; see [DT10] for an overview. Results are proven using combinatorial geometry, and using results on Gel'fand n-widths. The approach in [XH11] is a simple nullspace condition that is both necessary and sufficient, and uses Grassmannian angles to prove that matrices satisfy the condition. A complicated hierarchy of conditions, some of them implying others, is collected and discussed at Terence Tao's website [Tao] for those seeking further information.

1.3 The need for the RMPI

The random modulation pre-integrator (RMPI) is an analog-to-digital converter (ADC). Technically, the RMPI also combines a RF front-end so it is also a receiver, but its novelty resides in its approach to ADC. The RMPI is part of a new class of so-called "analog-to-information converters" (which is abbreviated variously as A2I or AIC). The purpose of an ADC is to digitize analog information so that it may analyzed and/or stored on a computer. For example, when recording music, an ADC is used to sample the input of a microphone and convert to a digital bit-stream. This is actually an example of an application that could benefit from CS, since CD-quality recordings require sampling at 44.1 kHz in order to capture frequencies up to about 20 kHz. The bit rate is 1411.2 kbit/s (e.g., two stereo 16-bit channels), yet on a computer, this is typically encoded by MP3 (or improvements, such as AAC) to 128 kbit/s, throwing away data.

Sampling audio-frequency sounds in the kHz range is not difficult, but sampling wide-band radio in the GHz range is. As a rule, ADC performance degrades as a function of the sampling speed. The most fundamental measure of an ADC performance is the number of bits $\tilde{B}$ of its output, meaning that the digitized output takes on $2^{\tilde{B}}$ possible values.

To describe both the bits and distortion, the effective number of bits (ENOB) term is used. This comes from the error (peak SNR) that an *ideal* $\tilde{B}$-bit ADC incurs due to quantization. Let the

desired sampling rate be f_s (i.e., the Nyquist rate), and suppose the ADC samples at γf_s, where $\gamma \geq 1$. The peak SNR of the ideal ADC is $\text{SNR}_p = \frac{3}{2}2^{2\tilde{B}}\gamma$ for a sinusoidal signal. Converting this to dB gives $1.76 + 6.02\tilde{B} + 10\log_{10}(\gamma)$ [GSS02].

For a non-ideal ADC, the SNR (signal-to-noise-ratio) or SINAD (signal-to-noise-and-distortion) may be different. The ENOB is thus an expression of the equivalent number of bits in an ideal ADC that would achieve the same SNR. In the following, we use SNDR (signal-to-noise-and-distortion-ratio) that accounts for all noise and distortion. Hence for $\gamma = 1$

$$B \triangleq \text{ENOB} = \frac{\text{SNDR}_{dB} - 1.76}{6.02}. \tag{1.3.1}$$

This is convenient because it incorporates distortion and the number of bits.

In the 2005 review [LRRB05], which supersedes the widely cited but now outdated review by Walden [Wal99], two useful figures-of-merit are discussed:

$$P = 2^B f_s \quad \text{and} \quad F = \frac{2^B f_s}{P_{\text{diss}}} \tag{1.3.2}$$

where P_{diss} is the power dissipation of the ADC. Walden suggested that the F improves over time due to more efficient power management, but that P is rather flat. Reviewing newer technology, [LRRB05] finds that P is not flat, but has slightly improved when f_s is slow.

There are many types of ADC: sigma-delta, folding, half-flashed, pipelined, SAR, and flash. The fastest ADC are of flash type, and reach about 1 GHz sampling rate, with about 7 ENOB; they are limited to a maximum of about 8 ENOB due to non-linearities. The ENOB for an ADC of 1 MHz is much better, going up to about 15 or 16. Fast ADC with high ENOB and accuracy are greatly desired. The cell phone revolution has spurred receivers to routinely deal with 100 dB dynamic range, and hence they need high accuracy, while the growing trend toward spread-spectrum RF systems also requires systems with high bandwidth.

Among current applications of high-rate ADC is the sensing of wide-band signals, such as several GHz of radio frequency bandwidth. This is of particular interest to the military, who seek to monitor these frequencies for signals of interest; this is referred to as SIGINT (signal intelligence) in military lingo. One application is detection of foreign radar pulses. This is quite different than usual radar, since it involves detecting radar pulses from *other* emitters, so the characteristics of the signal are not known. To prevent jamming, radar pulses may "frequency hop", meaning that the carrier frequency is changed over time. The pattern of carrier frequencies is known to the foreign radar emitter, but unknown to our third-party receiver.

Basics of heterodyne receivers. One of the most fundamental tools for designing a receiver is the principle of heterodyning. In heterodyning, a low-bandwidth signal (e.g., human voice, or

a radar pulse envelope) is mixed (i.e., multiplied in the time domain) with a carrier sinusoid for the purposes of transmission. The receiver then mixes with the same carrier and filters in order to recover the low-bandwidth signal.

To see how this works mathematically, let $a(t)$ be the low-bandwidth signal, and ω_c be the frequency of the high-rate carrier. Then the transmitted signal is

$$x(t) = a(t)\sin(\omega_c t).$$

For simplicity, let $a(t) = \sin(\omega_s t)$ where $\omega_s \ll \omega_c$ is a slow frequency. Using trigonometric identities, we express $x(t)$ as

$$x(t) = \sin(\omega_s t)\sin(\omega_c t) = \frac{1}{2}(\cos((\omega_s - \omega_c)t - \cos((\omega_s + \omega_c)t)).$$

To recover a, one multiplies the received signal $x(t)$ by another carrier of $\sin(\omega_c t)$. Then using the same trigonometric identities,

$$\begin{aligned}\sin(\omega_c t)x(t) = \frac{1}{4}(&\sin((\omega_c - (\omega_s - \omega_c))t) + \sin((\omega_c + (\omega_s - \omega_c))t) + \\ &- \sin((\omega_c - (\omega_s + \omega_c))t) - \sin((\omega_c + (\omega_s + \omega_c))t) \\ = -\frac{1}{2}&\sin(\omega_s t) + \frac{1}{2}\sin(\omega_s t)\cos(\omega_c t).\end{aligned}$$

When this new signal is low-pass filtered to frequencies below $\omega_c - \omega_s$, all that remains is $\frac{-1}{2}a(t)$ because $\omega_c \gg \omega_s$ and so only $a(t)$ is in the passband of the filter.

This means that if a signal has a known carrier frequency $f_c = 2\pi\omega_c$, then it can be "down-coverted" by mixing with $\sin(\omega_c t)$ at the receiver. Thus the bandwidth constraint depends only on $a(t)$.

For SIGINT applications, f_c is not known in advance, which complicates the system. The receiver needs to cover the entire range of possible carrier frequencies. As mentioned earlier, ADCs can only go up to one GHz, and do so at considerable expense of ENOB, but a frequency hopping signal might easily range over two or three GHz. The conventional approach is to channelize the receiver: a received signal is split into different paths or channels, then each channel k mixes the signal with a carrier frequency ω_k and then low-pass filters the result. The effect is that each channel views a small section of the spectrum. The clear downside of this approach is that it requires many channels, which requires more power and complexity. Consider a system to monitor about 2.5 GHz of bandwidth. Typical hardware uses 50 MHz channels, so it requires 50 channels! Since each channel is 50 MHz, the ADCs must sample at 100 MHz; a typical high ENOB 100 MHz ADC optimistically requires about 1 Watt of power, so the whole receiver requires about 50 Watts. In contrast, the design we

propose in Chapter 2 consumes about 1 Watt and covers the same bandwidth.

Because of the huge power requirements of current high-bandwidth receivers, in 2007 the DARPA called for proposals of innovative ADC designs [Hea07]. A joint Caltech and Northrop Grumman Corporation (NG, or NGC) team answered this proposal and in 2008 began the "A2I" project after completing a preliminary "Phase 0" proof-of-concept study. The on-site Caltech team consists of Professors Emmanuel Candès and Azita Emami, and graduate student Juhwan Yoo and this author. Further signal processing help is provided by the extended Caltech team, consisting of Dr. Michael Grant, and Professors Michael Wakin and Justin Romberg and their graduate students. The Nothrop Grumman team is based at the Northrop Grumman Space Technologies division, and led by Dr. Emilio Sovero, with chief designer Dr. Eric Nakamura.

The A2I project consists of three devices:

1. Caltech Random Modulator Pre-Integrator (RMPI). This is an eight channel RMPI device built in complementary metal–oxide–semiconductor (CMOS) by the Caltech team. It is designed to capture radar pulses over a 2.5 GHz bandwidth.

2. NG RMPI. This is a four channel RMPI device built in indium-phosphide (InP) by the Northrop Grumman team, and designed as a backup to the Caltech design. The design goals are the same as the Caltech design.

3. NG Non-Uniform Sampler (NUS). This is built by Northrop Grumman, and is a specialized ADC that samples at irregular time intervals. It is designed to capture signals that are longer in duration than radar pulses, and specifically, the GSM cell phone band. It has a 1.2 GHz bandwidth.

Signal processing techniques are shared between the Caltech and NG teams, as are some hardware plans.

At various stages, the team met with DARPA and the program manager (first, Professor Dennis Healy, and later, Dr. Daniel Purdy) to review progress. Several other teams, such as a joint Rice University and Applied Signal Technology team headed by Richard Baraniuk, answered the same proposal and are working on similar projects. Other groups include a team at HRL led by Peter Petre, a team at L-3 lead by Jerry Fudge, and a team at Texas A&M led by Sebastian Hoyos. Most of these projects focus on frequency sparsity, whereas the RMPI attempts to exploit time-frequency sparsity, which is more complicated.

The review meetings allowed teams to interact, and consequently some ideas have spread throughout all the teams. For example, the Hoyos team has a planned design that is quite similar to the RMPI presented here: 8 channels, seeking to cover 1.5 GHz of bandwidth.

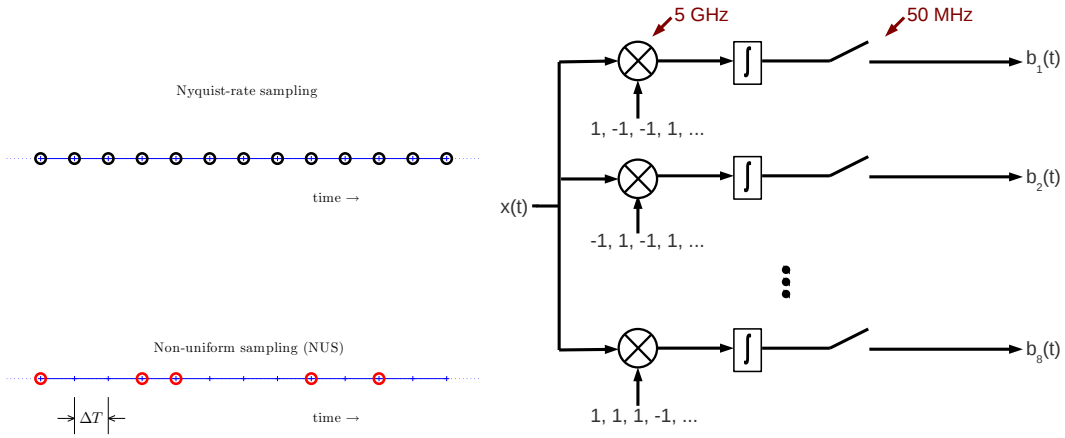


Figure 1.9: Pictoral representations of NUS (left) and RMPI (right). In the NUS, the large circles represent time-domain samples. The NUS samples at irregularly spaced intervals, though always in multiples of ΔT. In the RMPI, each channel mixes with a unique pseudo-random bit sequence (PRBS), and then integrates and samples.

1.4 Principles of the RMPI

In this section, the high-level operating principles of the RMPI are discussed. First, we introduce the NUS, which is conceptually simpler, and also simpler to build.

1.4.1 The NUS

The non-uniform sampler (NUS) acts as an ADC that samples at irregularly spaced time intervals. If the Nyquist rate is f_s, then a traditional ADC samples a signal $x(t)$ at regularly spaced time intervals $t_1, t_2, \ldots$ where $t_{i+1} - t_i = \Delta T = 1/f_s$. Limiting our discussion to a finite time window T, there are then $N = T/\Delta T$ Nyquist samples. Let x be the vector of samples $x[k] = x(k\Delta T)$, so the vector of conventional measurements can be written

$$b_{\text{conventional}} = Ix + z$$

where z is a noise vector and I is the $N \times N$ identity matrix. See the upper left of Figure 1.9 for a cartoon representation.

The NUS takes only a subset of these samples. Let Ω be a subset of $\{1, \ldots, N\}$, and let the cardinality of Ω be written as M; sometimes we may abuse notation and also refer to Ω when we really mean the time samples indexed by Ω. Write S for the matrix formed by keeping only the rows of I that have an index in Ω, so S is a $M \times N$ matrix. Then

$$b_{\text{NUS}} = Sx + z.$$

For example, if $\Omega = \{2, 4, 6, 8, \ldots\}$, then the NUS would be sampling at $2\Delta T$. This is *not* a

desirable choice for Ω, because there is no way to distinguish between a tone at frequency f and a tone at frequency $f + f_s/4$: both tones look identical if sampled on Ω. This can be remedied by replacing the 2 entry in Ω with either a 1 or 3. However, this change does not guarantee that arbitrary signals can be reconstructed, it merely addresses this particular ambiguity.

One of the first compressed sensing results [CRT06] is that the Fourier and time domains are usually extremely incoherent. Even though there are some bad sampling schemes (such as sampling at $2\Delta T$), *random* sampling in one domain means that coherence is likely small. If the signal x is sparse in frequency, then the time samples Ω should be chosen randomly. Hence the NUS chooses Ω via a pseudo-random number generator (note that Ω obviously must be known when trying to reconstruct the signal). The authors in [HH08] nicely state the intuition: "random undersampling renders coherent aliases into harmless incoherent random noise, effectively turning the interpolation problem into a much simpler denoising problem."

This is almost the entire story: the final complication is that hardware limitations impose some restrictions on Ω so it cannot be chosen completely randomly. Note also that the samples are not chosen uniformly over the time period $[0, T]$, but chosen uniformly from the discrete time grid $t_1, t_2, \ldots, t_N$. This is because it is more practical for hardware implementation and calibration, and because the theory is well understood (in terms of coherence between the Fourier and impulse discrete bases). Continuous-time compressed sensing theory has been developed to a limited extent but is certainly not as mature as discrete-space (finite dimensional) CS. See Figure 1.9 (left) for a depiction of the NUS.

What is the advantage of a design like NUS over a conventional ADC? The NUS might take samples spaced only ΔT apart, so at its fastest, it still needs to sample at f_s. However, the *average sampling rate* is M/N. For applications of interest, the spectrum is typically sparse, so it is possible to severely undersample with $M < N$. By using hardware tricks (beyond the scope of this thesis) it is possible to exploit this lower average sampling rate. This has the net effect of obtaining higher resolution samples than would be possible using an ADC sampling directly at f_s.

1.4.2 The RMPI

The RMPI system is designed to capture radar pulses; see Figure 1.9 (right) for a block diagram. A radar pulse consists of a pulse envelope, which is nearly trapezoidal except smoothed at the corners, which is modulated by a high-frequency carrier signal. The relevant features of a radar pulse is that it is sparse in frequency, since the bandwidth of the pulse envelope is typically not great (e.g., 10 MHz), and it is also sparse in time, since the pulse window is of finite duration. Radar signals typically repeat (on the order of 10 kHz), but the gaps between the pulses is greater (typically much greater) than the length of the pulses themselves, so the sparsity assumption is still valid.

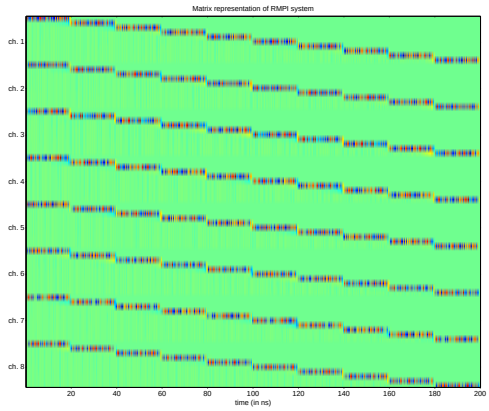


Figure 1.10: Sample measurement matrix. The green entries are zero or nearly zero. This particular matrix has rows ordered such that the measurements from each channel are grouped together.

Why not use NUS? The NUS operates on the principle that sparsity in the time domain and sparsity in the frequency domain are mutually exclusive. It takes non-uniform time samples, which are almost always incoherent with pure tones. On the other hand, the RMPI is designed to detect pulses that are also sparse in the time domain. If a short-duration pulse is sampled non-uniformly in time, it will only appear in some of the measurements, which is clearly inefficient. Viewed another way, the pulses are sparse in a Gabor time-frequency dictionary (see §2.7.2.2) which is not maximally incoherent with the impulse domain.

Operating principles of the RMPI. What *is* incoherent with the Gabor time-frequency dictionary? Another early result of CS [CT06] is that an instance of a *random* sensing matrix with iid entries from a Gaussian or signed Bernoulli distribution is overwhelmingly likely to be incoherent with a sparse signal in *any* fixed basis. The Gabor dictionary is not a basis, so these results do not directly apply; for example, the theory in [TLD+10] assumes sparsity in the Fourier domain, though recent work [CENR11] may help extend this to work with the Gabor dictionary. However, the intuition is that a Gaussian or Bernoulli matrix is a good sensing matrix. We note that these are not the only random matrices with good properties, but as they are particularly simple we focus on them.

The goal of the RMPI design is to approximate a random signed Bernoulli matrix as closely as possible. Due to practical considerations, the RMPI ends up approximating a block Bernoulli matrix. We motivate the RMPI with this approach for now, though in Chapter 2 it will also be motivated in a frequency-domain viewpoint.

Consider a Bernoulli matrix Φ that has r_{ch} rows and N_{int} columns. For example, if $r_{\text{ch}} = 3$ and

$N_{\text{int}} = 8$, it might look like

$$\Phi = \begin{pmatrix} 1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 & -1 & -1 & 1 & -1 \\ -1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 \end{pmatrix}.$$

Suppose $x(t), t \in [0, T_{\text{int}}]$, is represented by a vector $x \in \mathbb{R}^{N_{\text{int}}}$. For now, let this representation be obtained by averaging:

$$x[n] = \frac{1}{\Delta T} \int_{(n-1)\Delta T}^{n\Delta T} x(t)dt.$$

In future sections, digital representations are usually point samples $x[n] = x(n\Delta T)$. For low-frequency samples, there is not much difference in the two representations.

With this representation, the action of the first row of Φ applied to x is just

$$b_1 = \sum_{n=1}^{N_{\text{int}}} p_n x[n] = \sum_{n=1}^{N_{\text{int}}} p_n \frac{1}{\Delta T} \int_{(n-1)\Delta T}^{n\Delta T} x(t)dt = T_{\text{int}} \int_0^{T_{\text{int}}} x_p(t)$$

where p_n is the n^{th} column of the first row of Φ, and x_p is the signal x after mixing with p:

$$t \in [(n-1)\Delta T, n\Delta T) \implies x_p(t) = p_n x(t).$$

The two key components of the RMPI are thus mixing with a sequence p_n, followed by integration. The different rows of the Φ matrix correspond to different channels. Each channel is identical except that it uses a different sequence p_n so that it collects unique information.

The integrator must be sampled at some point in time T_{int}. Each channel sends its sample to an ADC which operates at frequency $f_{\text{ADC}} = 1/T_{\text{int}}$; the benefit of the RMPI is that $f_{\text{ADC}} \ll f_s$ so the ADCs do not need to be high-rate, and therefore they can be very accurate (e.g., 15 ENOB). To describe the system behavior after a long time T, multiple Φ matrices are concatenated. For example, if $T = 2T_{\text{int}}$,

$$\Phi = \begin{pmatrix} \Phi_1 & 0 \\ 0 & \Phi_2 \end{pmatrix} = \begin{pmatrix} 1 & -1 & -1 & -1 & 1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & 1 \end{pmatrix}.$$

Note that Φ_1 and Φ_2 do not need to be the same.

For a given time period T, assuming T_{int} divides T evenly, the number of rows of Φ is $M = r_{\text{ch}} \frac{T}{T_{\text{int}}}$. Since each row is a measurement, the number of measurements can be increased by either sampling more often (decreasing T_{int}, which means increasing sampling rate f_{ADC}), or increasing the number of channels r_{ch}. For a given number of measurements per time M/T (i.e., the information rate), a decrease in channels can be compensated for by an increase in sampling rate.

For example, the Caltech design uses 8 channels and a sampling rate of $f_{\text{ADC}} = 50$ MHz. The Northrop Grumman design uses 4 channels and a sampling rate of $f_{\text{ADC}} = 100$ MHz.

For long periods of time $T \gg T_{\text{int}}$, the Φ matrix is not Bernoulli, but rather block Bernoulli. For a fixed information rate, the number of channels can be increased and T_{int} increased, so that the matrix is exactly a Bernoulli. This is expected to have better performance, but it is impractical to have arbitrarily many channels, so in practice we deal with a finite value of r_{ch}. This trade off is explored in Chapter 2. The other extreme is a single channel operating with a short T_{int}; this case is dangerously close to a sub-sampled identity matrix, which it is coherent with signals that are sparse in the time domain, so it is not a useful CS matrix.

In a real system, the matrix is only an approximation of a block diagonal Bernoulli matrix. For reasonable perturbations, this does not affect the recovery, as long as the recovery algorithm knows Φ. Estimating the true value of Φ is the subject of the calibration section in Chapter 2. The effect of perturbations will also be discussed.

Finally, we note that because the RMPI is designed to mimic a Bernoulli matrix, is is nearly optimal for any kind of compressible signal, not just time-frequency sparse radar signals! In this sense, it is much more general than NUS. The tradeoff is extra complexity in the design and recovery.

1.5 Optimization background

The main post-processing step for recovering a signal from compressed measurements is solving one of the optimization problems described in §1.1.1. The optimization chapters of this thesis will cover state-of-the-art optimization algorithms, but do not discuss in depth the alternatives to optimization-based approaches. Below is a brief survey of the various methods researchers use to attack linear inverse problems; for a readable recent review, see [TW10]. We eschew specifics here; some state-of-the-art ℓ_1 solvers, as of 2009, are compared in Chapter 3. See also the masters thesis [Pop09] and the paper [Lor09] which both review ℓ_1 solvers as of 2009.

Below is a list of categories of algorithms that are used for recovering signals from undersampled or ill-posed data; these categories are not necessarily mutually exclusive. For example, soft-thresholding algorithms fit into several of these categories, and some greedy algorithms can be viewed as convex optimization algorithms on dual problems.

- Convex relaxation methods. These are true ℓ_1 solvers.
    - — Interior-point methods (IPM). These are second-order methods that require solving a linear system at every iteration; in return for this expense, they typically need few iterations (20 to 50) to converge. The theory behind them is well understood [NN94]. General purpose solvers are SeDuMi, SDPT3, and CVXOPT. Specialized solvers are l1ls [KKB07]

and l1Magic [CR07b]. The downside of IPM is the poor scaling with dimension, and recent years has seen trends away from IPM when solving very large problems. For medium sized problems, IPM are the gold standard. We also note the useful modeling framework CVX [GB10], which uses IPM solvers.

— First-order methods. These rely only on gradient calculations. Each iteration is cheaper than those in an IPM method, but typically more iterations are required to reach convergence. However, this trade off may be worth it. The algorithms presented in Chapters 3 and 4 are of this type.

— Splitting methods. Some splitting methods, such as the forward-backward method (used in [HYZ07]), can be motivated also as gradient methods. Others, such as the Douglas-Rachford, are distinct from gradient-based approaches, and may work quite well in practice. For an example, see Figure 4.15. These methods were originally inspired by PDE splitting methods. Convergence of the Douglas-Rachford splitting is proved in [Com04].

— Homotopy/pivoting methods. For a problem like the unconstrained LASSO with parameter λ_0, these methods find all solutions for values of λ between $[\lambda_0, \infty]$. This is useful for model selection when λ is unknown. Unfortunately, these methods are typically too slow for signal processing applications. Examples of homotopy methods are LARS [EHJT04] and HOMOTOPY [OPT00] which solve the LASSO (see also [DT08]), and DASSO [JRL09] which solves the Dantzig selector.

— Coordinate descent. If done properly, this can exploit prior computation and be quite efficient. By itself, it is an extremely old optimization method; see [FHT10] for recent versions. It is related to greedy methods.

— Variants. Noiseless basis pursuit (BP) is a linear program, and so can be solved via the simplex method. The other variants, such as BPDN, are not linear programs, but can be solved by various standard optimization techniques such as sequential quadratic programming (SQP), augmented Lagrangian methods, subgradient methods, etc. These are really classes of techniques, not specific algorithms.

  In terms of compressed sensing, two algorithms deserve special mention. The first is non-linear conjugate gradient; whenever this can be used, it is likely to be fast. It is used to solve the non-convex rank minimization problem in [BM03] and recently in [MT10]. The other algorithm is the alternating direction method of multipliers (ADMM), aka alternating direction augmented Lagrangian. This method "splits" a variable into two distinct copies of itself, and then imposes a penalty function that encourages the two copies to be equal. The two copies of the variable are updated in an alternating fashion (much like the Jacobi method). Very recent theoretical work has extended the ADMM

method to updates in a Gauss-Seidel-like fashion, which crucially allows for the first ever convergence rate estimates [GM10] and proves convergence for the case when a variable has been split into more than two copies (this had been used in practice but not analyzed). An extension by the same authors proposes an accelerated variant, using Nesterov-style ideas, which converges more rapidly [GMS10]. One reason for the recent interest in ADMM is that the subproblems are tractable for nuclear-norm minimization problems, and especially one type of robust PCA problem [CLMW09]. Another interest in ADMM is that multiple splittings potentially allow for very low-communication parallel algorithms [BPC+10].

- Thresholding. Soft-thresholding can be cast in other frameworks (e.g., gradient-based, splitting). Hard thresholding is sometimes possible to cast as a solution to a convex functional using a Landweber iteration. The iterative hard thresholding algorithm itself can be analyzed on its own, and recent work in [BD09] has established noiseless recovery guarantees similar to ℓ_1 minimization under the assumption of the RIP; the main issue seems to be dealing with noisy signals.

- Iterative ℓ_1 algorithms. The canonical example is reweighted ℓ_1 [CWB08], which is a more advanced version of the adaptive LASSO [Zou06]. Other variants include iterative-support detection (ISD) [WY10] and active-set methods, such as FPC-AS [WYGZ10]. The ISD method is quite similar in principle to CoSaMP, a greedy algorithm, but can also be viewed as a reweighting algorithm that allows weights with zero value. These methods solve convex sub-problems, but the overall method itself need not be convex. Reweighting is one of the techniques used extensively in Chapter 2. So far, there is no theoretical result showing that reweighted ℓ_1 converges to a *global* minimum. Other algorithms, such as ISD, are shown to converge (locally) in the noiseless case. The algorithms considered in [Zou06, KXAH10] are both two-step methods, so convergence is not an issue; both also offer quantitative theoretical performance improvements over the LASSO. Another two-step method in [Chr09] appears to work as well. See further discussion in the reweighting §2.7.2.3 and debiasing §2.7.2.4 sections in Chapter 2.

- Greedy pursuits. These methods seek to update the signal with the best possible update, without taking into account global structure. Early methods (OMP) update one entry at a time, while more recent methods update many entries at once (CoSaMP). In statistics, this is a type of forward stepwise regression; in signal processing these are often called "pursuits" (such as matching pursuit [MZ93]); and in approximation theory and optimization they are called "greedy methods". These do not exist independently of other optimization techniques, and are closely related to variants of coordinate descent.

  The prototypical greedy pursuit is orthogonal matching pursuit (OMP) [TG07] which im-

proves on matching pursuit. OMP has recovery guarantees almost as good as ℓ_1 recovery [TG07, DW10] for the case of Gaussian matrices, and was one of the first methods to be analyzed [Tro04] after the seminal work [DH01] in the early 2000s. OMP is important and simple enough that we describe it briefly. If b are the observations and Φ is the measurement matrix, start with $x = 0$ and the residual $r(x) = \Phi x - b$. Every step consists of two parts. First, select the atom ϕ_i that is most correlated with the residual $r(x)$; second, update x by solving the least-squares problem using only the atoms that have been previously selected.

Improvements on OMP are stagewise OMP (StOMP) [DTDS06] and regularized OMP (ROMP) [NV10, NV09]. So far, the best results are from the CoSaMP method, which adds up to k entries at every step, solves the least-squares problem, and then prunes the support set; it is not unlike the iterative support detection algorithm in [WY10]. In [NT09], it is proved that the RIP implies CoSaMP has the same recovery guarantees as ℓ_1 minimization. Furthermore, the running time is fast: it has linear convergence. One downside is that you must specify the desired sparsity level (though this is not to say that the algorithm fails on compressible signals; in that case, it returns an estimate close to the best k-term approximation, just as ℓ_1 recovery does).

- Bayesian framework. This puts a prior on the signal x and attempts to solve for the maximum *a posteriori* (MAP) estimate [WR04]. The functional may not be known in closed form, which makes rigorous results difficult. The recent work on correcting for jitter (discussed in Chapter 2) [WG11] is of this type.

- Nonconvex optimization. This set of problems seeks to minimize a ℓ_p norm for $0 < p < 1$, which is non-smooth and non-convex. Other than special non-convex problems such as rank-minimization for low-rank problems [BM05], there are no known results on efficiently finding global minima (of course, non-convex techniques such as simulated annealing can be used, but these are much slower). Work by Chartrand has shown that the global solution of ℓ_p problems does enjoy recovery guarantees [Cha07].

  We also note the very interesting work [WXT10]. It shows that for strong recovery (which is what we have discussed in earlier theorems), ℓ_p recovery outperforms ℓ_1 recovery. Yet for a type of fixed-design case, called "weak recovery," which considers recovery for a fixed sign pattern of the signal, it is shown that ℓ_1 minimization is actually superior to ℓ_p minimization. They also demonstrate that ℓ_p solutions can be denser than ℓ_1 solutions.

- Brute force. The field of subset selection [Mil02] has long been concerned with minimizing the ℓ_0 quasi-norm, since common information criteria such as AIC and BIC are based in terms of ℓ_0. For small problems, brute force combinatorial search is possible, and dynamic programming

techniques may be of some use. In general, this approach is completely infeasible for large signal recovery problems.

- Belief-propagation and message-passing. These methods are well-known in coding theory and graphical modeling, and can be remarkably quick, but often depend upon special properties of the measurement matrix. The work of [BSB10] proposes a belief propagation algorithm for a class of low-density parity check (LDPC) matrices specialized for compressed sensing; similarly, the work in [SBB06] presents a fast algorithm, but relies on specialized measurement matrices. In [DMM09], an approximate message-passing algorithm is explored, but it lacks the theory of ℓ_1 minimization.

  Other ideas related to coding theory are discussed in [XH07, JXHC09] where types of expander graphs are used to encode signals. Decoding is similar to belief propagation of LDPC codes, and the running time of the algorithm is $\mathcal{O}(n \log(n/k))$ which is extremely fast. Futhermore, [JXHC09] has shown that only $k \log(n/k)$ measurements are necessary to recover a k-sparse signal, which is the same order as found for ℓ_1 minimization. The main disadvantage is that the encoding matrix must be chosen to have special properties. It is possible to deterministically construct such matrices, but, for the case of RMPI, it is not yet apparent how to construct the physical realization of an arbitrary matrix. This topic is worth pursuing, and is especially tantalizing because it may allow on-chip real-time recovery without the need for a computer backend.

- Combinatorial algorithms. Similar to the coding-theory based ideas, these use special decoding algorithms that are similar to group testing, and rely on highly structured encoding matrices. So far, they have been shown to recover sparse signals but with more measurements than for ℓ_1 minimization. Examples include heavy hitters on steroids (HHS) [GSTV07], which improves on the related earlier algorithm [GSTV06] by the same authors. A similar algorithm is the Fourier sampling algorithm [GST08].

Discussion. Of the categories discussed, the two most commonly used are the convex optimization methods (perhaps with tweaks), and the greedy methods. Most other methods are too specialized, or not yet thoroughly understood.

One of the benefits of greedy methods is that they can be modified to work with models other than sparsity; for example, [BCDH10] has modified CoSaMP to enforce tree-based structure. See the discussion in §2.7.2.3. However, new works show that optimization-based methods may be able to solve model-based problems, using variants of reweighting techniques (as suggested in this thesis, and by [DWB08]), and also since the proximity operators may be computable [JMOB10].

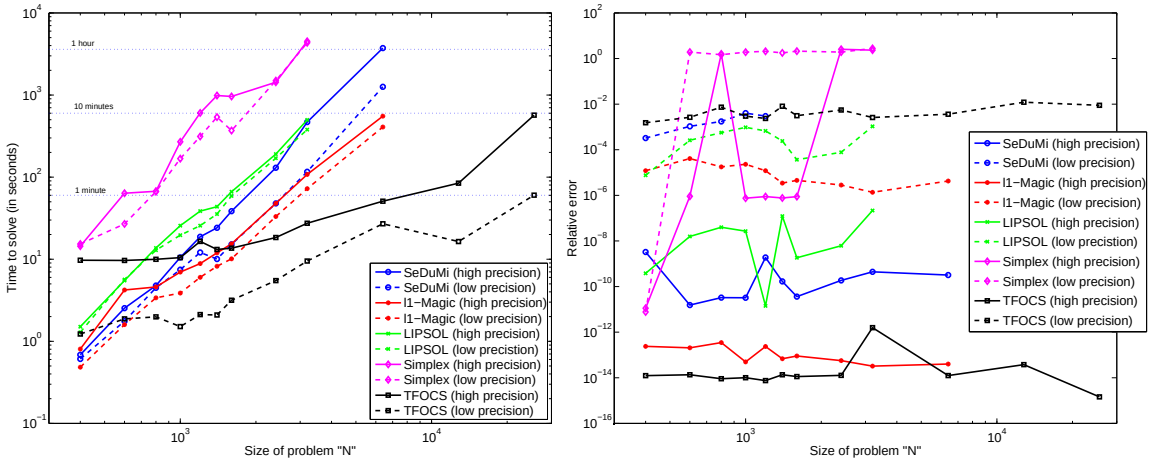


Figure 1.11: Using noiseless basis pursuit (1.1.5) with $m = n/2$ measurements and a sparse solution with $k = m/5$ nonzeros. The TFOCS solver used $\mu = .1$ (see equation (4.1.14)) which was selected on the first try and not "tweaked". In order to be fair to IPM, this example used a dense measurement matrix A. If A is a partial DCT or FFT, then TFOCS can solve problems with 2^{20} variables in about a minute; see Figure 1.12.

The conventional wisdom is that ℓ_1 minimization methods are the best, but at the expense of being slower. Like most conventional wisdom, there is truth behind this yet it is not the whole story. We focus on ℓ_1 minimization methods in this thesis, but remain open to other methods in the future. As research progresses, the disadvantages of each method will be either exposed or fixed, so it is possible that variants of greedy methods will come to be considered as robust as ℓ_1 minimization, and that ℓ_1 minimization will be as fast as greedy methods. In short, the disparate methods may start looking more and more alike.

Further discussion of IPM. The initial approach we took to solve reconstructions for the RMPI was via interior-point methods. A version of l1Magic [CR07b] was modified to allow for analysis (1.1.9); another variant using a null-space formulation was also used, but only covered the noiseless case. The l1Magic software is specially designed to solve the Newton step via iterative methods. In the noiseless synthesis formulation (1.1.8), with $A = \Phi\Psi$, the update step requires solving

$$\begin{pmatrix} D_x & A^t \\ A & 0 \end{pmatrix} \cdot \begin{pmatrix} \Delta x \\ \Delta \nu \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \tag{1.5.1}$$

where D_x is a positive diagonal matrix that changes every iteration. From here, we could solve this "saddle-point" system (e.g., with MINRES, since the system is symmetric and non-singular, but not positive definite unless A is), especially if we have a preconditioner. Or, the standard technique is to reduce further, and get

$$AD_x^{-1}A^t \cdot (\Delta\nu) = \tilde{w} = AD_x^{-1}w_1 - w_2. \tag{1.5.2}$$

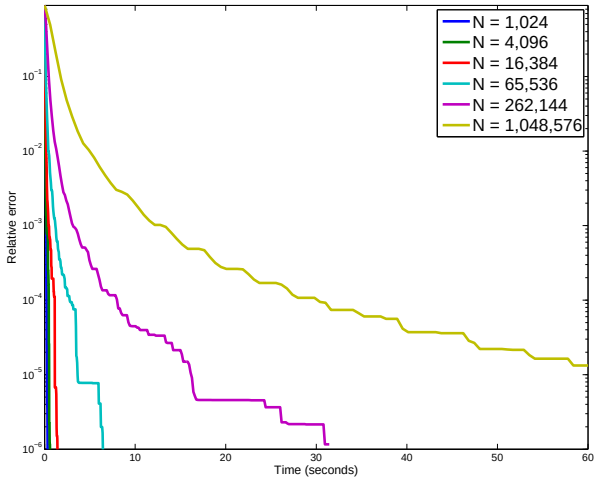


Figure 1.12: Using noiseless basis pursuit (1.1.5) as in Figure 1.11, but now with a DCT measurement matrix. Using $m = .7n$ measurements and a sparse solution with $k = n/16$ nonzeros. Solved on a modern laptop with a quadcore Intel i7 processor.

In this case, since D_x is a positive-definite diagonal matrix, the matrix $AD_x^{-1}A^t$ is positive definite. Recall $A = \Phi\Psi$. The matrix Φ is of reasonable size, but Ψ is an overcomplete Gabor dictionary which is quite large. The key benefit of the Gabor dictionary is that it exploits the FFT to make matrix-vectors multiplies in $\mathcal{O}(n \log n)$ time. Thus l1Magic solves (1.5.2) using the Conjugate Gradient (CG) method; any direct factorization method is impractical. Unfortunately, a well known issue with IPM is that D_x splits into widely different scales near the solution, and hence AD_xA^T has widely separated eigenvalues. This causes CG to take many iterations.

To overcome this, the author and Emmanuel Candès tried preconditioning the saddle-point system. One possible preconditioner for the saddle-point system, motivated by [RG07, GS06b], is

$$P = \begin{pmatrix} D_x^{-1} + \gamma^{-1}A^tA & 0 \\ 0 & \gamma I \end{pmatrix}.$$

The saddle-point system (1.5.1), which is not positive definite, is solved via MINRES, and the preconditioner P is inverted via CG. Hence the overall algorithm requires three levels of iteration: the primal-dual iterations, the MINRES iterations, and CG iterations. This was implemented using both the l1Magic [CR07b] and CVXOPT [DV10] packages. Unfortunately, the three levels of iterations proved to be too slow. After trying other first-order methods, the authors developed the algorithms presented in Chapters 3 and 4. These significantly improve over IPM in terms of running time, and do not suffer in accuracy. See Figure 1.11.

One of the issues with IPM is that the most effective methods are predictor-corrector methods that take two steps every iteration. The equation (1.5.2) is solved via a Cholesky factorization, and this factorization can be re-used for the second step, so the "correction" step is very cheap. When

the Newton step is solved with an iterative method like CG, this is no longer as attractive since the two steps are both full cost. The ill-conditioning issue is another problem. But a basic fundamental problem is that for extremely large numbers of variables, any type of method that solves a linear system of equations at every step is going to be too slow.

There is work on improving the use of warm-starts for IPM [GG03, BS07, YW02], which may lead to better performance. Special versions of conjugate gradient that exploit information from previous solves could also make IPM more attractive. Finally, we mention randomized linear algebra (see [HMT11] and the references therein) which may lead to efficient methods to approximately solve the Newton step.

1.6 Reading guide

Different readers may find certain parts of the thesis more interesting than others. The PDF version of the thesis is equipped with internal links to facilitate quick jumps from section to section.

RMPI design. The reader interested in the RMPI will find Chapter 2 contains the details, and Chapters 3 and 4 may be skipped. The conclusion in Chapter 5 is also relevant to the RMPI.

Signal processing and optimization. Chapters 3 and 4 discuss the NESTA and TFOCS optimization algorithm. From the RMPI chapter, the section on calibration §2.3.4, phase-blind calibration §2.3.5, and recovery §2.7 discuss signal processing techniques. The concluding chapter also contains some discussion.

Novelty. All of Chapter 2 is new and has not appeared in print before. Most of Chapters 3 and 4 are verbatim from published or submitted work, and so may be skipped if the reader is already familiar with the work. However, these chapters contain a few new sections which we mention here. For NESTA, the new section §3.7 discusses an extension to deal with the case when $\Phi\Phi^* \neq I$.

In TFOCS, the new section §4.6 recasts the original derivation of TFOCS in a dual function framework, as opposed to the dual conic framework. A section on convergence §4.6.4 has been added, and the appendix §4.12 on test problems has been rewritten slightly and comments on new literature that appeared in March 2011. The section §4.8 discusses applying TFOCS to conic programs in standard form (e.g., LP, SOCP, SDP) and specialized algorithms for this case; it also discusses some results regarding the special problem of matrix completion and using splitting algorithms such as Douglas-Rachford, as well as a novel automatic restart scheme. Future directions are discussed in the concluding chapter.

Chapter 2

RMPI

This chapter introduces the Random Modulation Pre-Integrator (RMPI) and details the status of an integrated circuit implementing the architecture. To our knowledge, this is the first full-speed hardware RMPI in existence. The goal of this chapter is to describe some key parameters of the RMPI that are essential to its performance. This is not a hardware chapter and does not go into detail of the designs; the reader does not need any knowledge of Op-amps, though standard mathematical and signal processing background is helpful. The idea behind the parameter selection is that of trade offs. We presuppose that a competent engineer is able to make several designs, and the task is to decide which design is better.

This introduction describes the setting of the RMPI and its basic principles, and tries to make clear the intuition behind the receiver, as well as outline when the RMPI is an appropriate architecture and when it is not. Further sections of the chapter describe the system in more detail, and report a some of the many results from 3 years of study and simulation.

The hardware implementation of the RMPI has been returned from the manufacturer, and is in the process of being calibrated, which has required significant work (to be discussed in §2.3.4), but results are encouraging. An earlier chip, which we refer to as "version 1," was manufactured and some reconstructed pulses obtained, but otherwise all plots in this chapter are from simulated data. The simulations took place at various levels of realism, which will are described in §2.3. Most computational work was performed in Matlab and/or Simulink.

The RMPI is due to the work of a large number of people, but we especially want to mention the efforts of Juhwan Yoo. He has done most of the hardware design, collaborated on the system design, and contributed figures and occasional text to this chapter. Funding for work was graciously provided by DARPA grant FA8650-08-C-7853.

2.1 Introduction

2.1.1 Signal class

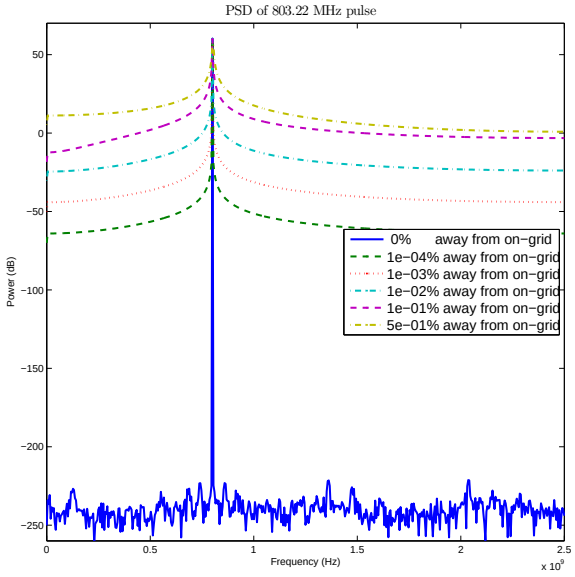


Figure 2.1: The problem of spectral leakage. Shown are FFT values. The FFT is a bad way to do spectral estimation [Tho82].

The data acquisition of the RMPI works well for several signal models, but each signal model has certain issues, so we restrict ourselves to a specific model. The goal of this RMPI is recovery of radar pulses, so our signal model is tailored to this. Let $x(t)$ be a continuous time signal, and $x[n] = x(n\Delta T)$ be a sampled version, where $\Delta T \triangleq 1/f_s$ is the Nyquist time; samples are not actually acquired at this resolution, but this Nyquist rate version of $x[n]$ is used in the digital post-processing. See Table 2.1 for a reference of the notation that is used.

The first assumption is that if the signal $x(t)$ can be characterized if we limit time to an interval $[0, T]$. This is true if $x(t)$ is periodic with period T, or if $x(t)$ is zero outside a narrow time window, as it is for a short radar pulse. An implicit requirement is that T is not too great. The postprocessing stage can handle up to about $N = 8192 = 2^{13}$ length vectors in the case of radar signals, and up to perhaps $262144 = 2^{18}$ for pure tones; the radar pulses are limited because of the size of the Gabor dictionary used in analysis, while pure tones use a simpler over-sampled FFT dictionary. Typical simulations in this chapter work with $N = 1024$ to $N = 8192$, so the corresponding time periods are 204.8 ns to 1.638 μs.

The next assumption is actually at odds with the the first: it is assumed that $x(t)$ is bandlimited to the range $[-f_s/2, f_s/2]$. This appears to rule out radar pulses, since it is impossible for signal to have compact support in both time and frequency. However, we simply consider the periodic extension of a radar pulse. By forcing the signal to be periodic, it means the spectrum is discrete

and given by the discrete Fourier transform (DFT).

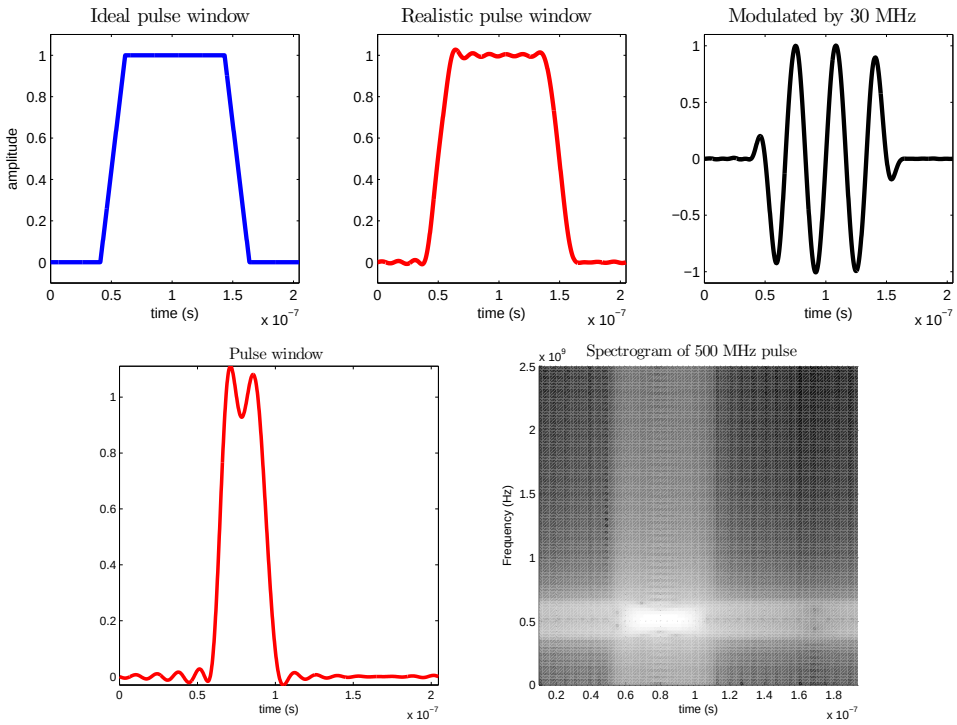


Figure 2.2: Radar pulses. Top row: on the left and middle, two sample pulse windows. On the right, a radar pulse in the time domain. Bottom row: sample pulse window on left, and on right, same pulse, modulated at 500 MHz, in the time-frequency plane

The pragmatic definition of the signal class is any signal that is well-approximated by N of its Nyquist-rate samples. One type of signal which does *not* belong to this class is the pure tone that has frequencies which are not on the periodic grid that has been imposed by the finite number of samples. Such a tone is called "off-grid," and Figure 2.1 shows the DFT of such tones. The frequency grid runs from 0 to $f_s/2$ in spacings of $\Delta f = 1/T$, which makes it clear that as $T \to \infty$, the spacing approaches 0 and the signal class approaches all band-limited signals. Viewed another way, the sampling from $[0, T]$ is like windowing a signal with the boxcar filter, so the spectrum is convolved with the Fourier transform of the boxcar filter, which is the sinc function. The sinc function has slow decay, so the off-grid tones cannot be well-approximated by just N Nyquist-rate samples. In §2.7.2.6 we discuss the problem further and propose solutions using windows. For now, these signals are not part of the signal model and do not concern us.

Signals that do fit this category are radar pulses, as shown in Figure 2.2. A radar pulse consists of an envelope, modulated by a carrier frequency. This modulation is just time-domain multiplication. The carrier is in the RF band between $[0, f_s/2]$, and is necessary for transmission. Other than the carrier frequency, the information content of a pulse is its time-of-arrival and duration, since these are used in radar systems to estimate range, angle, and speed. In order for time-of-arrival estimation

to be consistent, it is desirable for a pulse to have a well-defined leading edge that rises sharply, and hence pulses are approximately just trapezoids or even rectangles. However, a trapezoid is similar to the boxcar function, and so it has high spectral content due to the sharp corners. Radar emitters have only so much frequency bandwidth, so the radar pulse envelope is bandpass filtered either intentionally or unintentionally. This is beneficial for the RMPI system, since it makes the pulse sparser in frequency. This fact has been taken into account in pulse simulations. Another deviation from the trapezoidal model is that receivers often see multiple reflections of a signal, which means that the trailing edge of the pulse is generally degraded. This is not accounted for in our simulations, but it is not expected to significantly affect performance.

Looking at Figure 2.2, it is apparent that the radar signals are slightly sparse in time, and slightly sparse in frequency, but very sparse in the time-frequency plane. Using a time-frequency dictionary is essential to recovery, and is discussed in §2.7.2.2. The pulses considered in the model have duration varying between 100 and 1000 ns, and typically 200 ns.

Having stated the signal model, what is the goal of the system? The goal is an ambitious one: full reconstruction of the signal up to the accuracy permitted by the finite model. This means not just frequency domain estimation, but also time-domain estimation. Of all the works discussed in the related literature §2.2.3, none of them attempt time-domain reconstruction. We don't preclude the possibility of recovering multiple pulses simultaneously. Simulation results indicate that recovering two pulses of the same size, even if they overlap in time, is not much different than recovering a single pulse. The biggest challenge lies with recovering two pulses of extremely disparate size; this is discussed in §2.8.

In addition to this primary goal, a secondary goal is more practical: we wish to estimate the pulse parameters, termed "pulse descriptor words" (PDW), which are significant for radar processing. These include carrier frequency, phase, time-of-arrival, duration, and pulse repetition rate. For this kind of estimation, we propose a "compressive matched filter" (see [ERW11] for very recent results), discussed further in §2.7.1.

When is the RMPI a useful system?

- For high bandwidth designs, requiring 1 GHz bandwidth or more.
- When the input is sparse in the time-frequency plane; for example, it might consist of a few narrowband communication channels and a few radar pulses.
- For applications that need high precision, due to high dynamic range. For example, one might be interested in recovering very faint radar pulses that occur concurrently with some very large signals from a cell phone tower or airport.
- For applications that have power restrictions, so that massive filter banks are not an option. An example would be an airborne systems with a tight power budgets.

- For systems that transmit signal information to a central processing location. With the RMPI, the compressed measurements, or even the PDW, are transmitted, so less channel capacity is needed. The Herschel satellite telescope is an example: it has limited capability for processing, and a low-rate down-link channel to the Earth, so it sends compressed measurements and processing is done on the ground.

- For identifying unconventional signals, such as ultra-wideband spread spectrum. A channelized receiver is limited to analyzing 50 MHz (or similar) of spectrum at a time, while the RMPI analyzes the entire bandwidth simultaneously, and can make use of prior knowledge about the signal.

- For noisy environments that contain signals with relatively low-information content (such as so-called "finite rate of innovation" signals). The RMPI recovery process applies digital denoising, so for some signals, the RMPI reconstruction is much more accurate than even high ENOB Nyquist-rate measurements, since a conventional receiver does not denoise.

- Similarly, spectral estimation [Tho82] with the RMPI can be more accurate then conventional spectral estimation, since compressed spectral estimation [DB10] is done in digital and takes into account all the measurements at once.

- Other estimation, such as angle-of-incidence, can be done in the compressed framework, and like spectral estimation, they may benefit from using all measurements at once.

The RMPI is not appropriate for all applications. One of the current drawbacks is that it requires a computer for the recovery and does not yet process data in real-time (meaning that it takes the computer longer than time T to process the data generated from T worth of samples). These limitations are not fundamental but rather just a limit due to the current state of technology. The compressive matched filter is a promising technique for accelerating computation, and it is conceivable that this could be performed in real-time in hardware or via an FPGA.

Representation of the system. Characterizing the system will be the subject of §2.3.4, but one aspect of this is conceptually very important, so we mention it now. Because of the sampling, our system is not time-invariant, so we cannot analyze it in a pure linear-time-invariant (LTI) framework. However, it is linear (or at least approximately so; this is discussed in §2.7.2.5). This is fundamental for many reasons. First, it gives us a way to calibrate our system and determine how it is really functioning. It also allows us to write down the measurements in the linear inverse framework described in the introduction:

$$b = \Phi x, \quad \text{or, in the noisy case,} \quad b = \Phi(x + \sigma_1 z_1) + \sigma_2 z_2.$$

Notation	Typical values	Fixed or variable	Description
f_s	5 GHz	fixed	Nyquist rate
$\Delta T \triangleq 1/f_s$	200 ps	fixed	Nyquist period
N	1024, 4096	variable	length of digital signal x_n
$T \triangleq N\Delta T$	204 ns, 820 ns	variable	period of digital signal
$c(t)$	NA	NA	chipping sequence with PRBS $\{c_n\}$
N_{chip}	128	variable	periodicity of chipping sequence $\{c_n\}$
$T_{\text{chip}} \triangleq N_{\text{chip}}\Delta T$	25.6 ns	variable	period of $c(t)$, assuming $f_{\text{chip}} = f_s$
$1/T_{\text{chip}}$	39 MHz	variable	frequency of repetition of $c(t)$
f_{chip}	f_s	mainly fixed	rate of modulation with c_n
N_{int}	100	mainly fixed	length of integration in units of ΔT
$f_{\text{ADC}} \triangleq f_s/N_{\text{int}}$	50 MHz	mainly fixed	sampling rate of ADCs
$T_{\text{ADC}} \triangleq 1/f_{\text{ADC}}$	20 ns	mainly fixed	period of ADC samples
$T_{\text{int}} \triangleq T_{\text{ADC}}$	20 ns	mainly fixed	synonym for T_{ADC}
r_{ch}	8	mainly fixed	number of channels

Table 2.1: Notation used with the RMPI. We will use $i = \sqrt{-1}$ (rather than j). "Fixed or variable" refers to whether we often change the value of the parameter. For example, determining an appropriate value of r_{ch} is the topic of §2.4.3, but is usually fixed at $r_{\text{ch}} = 8$ in the rest of the sections.

Variant	Time	Frequency
Fourier Transform, $\hat{x} = \mathcal{F}x$	$x(t) = \int_{-\infty}^{\infty} \hat{x}(f) e^{i2\pi ft} df$	$\hat{x}(f) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi ft} dt$
Fourier Series	$x_T(t) = \sum_{k=-\infty}^{\infty} \hat{x}[k] e^{i2\pi kt/T}$	$\hat{x}[k] = \frac{1}{T}\int_0^T x_T(t) e^{-i2\pi kt/T} dt$
DFT, $\hat{x} = Fx$	$x_n = \frac{1}{N}\sum_{k=0}^{N-1} \hat{x}_k e^{i2\pi kn/N}$	$\hat{x}_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N}$
Unitary DFT, $\hat{x} = F_u x$	$x_n = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} \hat{x}_k e^{i2\pi kn/N}$	$\hat{x}_k = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N}$

Table 2.2: Variants of the Fourier transform. Listed to show normalization and notation conventions. The notation x_T means x is periodic with period T. Frequency variables written as f are in Hz, and written as ω are angular frequencies in radian/s. The notation $\hat{x}$ is used for both $\mathcal{F}x$, and occasionally to denote an estimate of x; context should distinguish the two uses. DFT stands for discrete Fourier transform, and is synonymous with the fast Fourier transform (FFT).

The system itself is represented by the Φ matrix, and we make much use of this fact. The linearity is also important since all our reconstruction algorithms assume linear measurements.

We also make use of many symbols, which have been collected in Table 2.1, and also speak of various types of Fourier transforms. Table 2.2 shows the normalization conventions used.

2.2 The design

2.2.1 Basic design

The design of the RMPI can be motivated by considering our goals, and then reasoning about what is necessary. At first, we need only consider a single channel, since extra channels simply give more information of the same type.

Overall, we aim to digitally sample a wideband signal at a low-rate. The incoming bandwidth to the system is $f_s/2 = 2.5$ GHz, and the ADC samples at 50 MHz, which is $100\times$ slower than the Nyquist rate f_s. We allow a few channels, but 100 channels is impractical, since this would have the same giant power consumption as traditional channelized receivers, so a filter bank approach is not valid, and thus it is fair to assume that the input to a single channel has the full bandwidth of 2.5 GHz. Sampling this at 50 MHz causes aliasing, so that an input of frequency f_{in} and an input of $f_{in} + 25$ MHz are indistinguishable.

In order to make these inputs distinguishable, the signals must be modified at some point before the ADC. A reasonable strategy is to mix the signals with another known signal; this is equivalent to multiplication in the time domain (note that *adding* another signal would be no use, since the ADC sampling is linear). Let this known signal be called the "chipping sequence," to borrow the term from spread-spectrum design [PSM82], and denote it by $c(t)$. Given two distinct signals x and x', we desire that $x(t)c(t)$ and $x'(t)c(t)$ are distinguishable even after being aliased by 50 MHz sampling.

Unfortunately, this is not possible just via mixing. Suppose the signals x and x' are distinct, but become identical after aliasing, which means $x(kT_{\text{int}}) = x'(kT_{\text{int}})$ for any k. Then clearly multiplying x and x' by the same signal c will not resolve the situation. In order to distinguish the two signals, the samples must involve information from in between $(k-1)T_{\text{int}}$ and kT_{int}. Let the samples of our function be $y = y(x)$ and $y' = y(x')$. Linear samples y_k can be represented by convolving with some kernel h known as the transfer function:

$$y_k = \int_{(k-1)T_{\text{int}}}^{kT_{\text{int}}} c(t)x(t)h(T_{\text{int}} - t)dt.$$

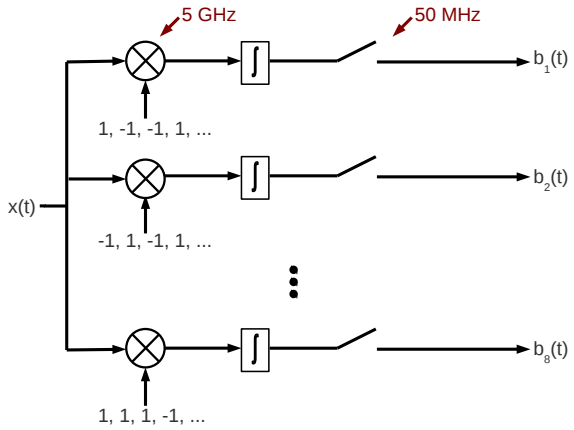


Figure 2.3: Principles of the RMPI. Each horizontal section represents a channel, and each channel has a unique PRBS sequence of ± 1's. After integration, each channel is sampled at 50 MHz.

For now, let $h = 1$ and consider

$$y_k - y_k' = \int_{(k-1)T_{\text{int}}}^{kT_{\text{int}}} (x(t) - x'(t))c(t)dt = \langle (x - x'), c \rangle$$

as a linear operator $\langle (x - x'), \cdot \rangle$. Thus $y_k = y_k'$ if c is in the null space of this operator. If $x \neq x'$ (by which we mean they differ on a set of nonzero measure), then this is not the zero operator, so there is a non-trivial complement of the null space (assume $x, x' \in L_2[(k-1)T_{\text{int}}, kT_{\text{int}}]$). Though we do not make the notion precise, a "random" function $c(t)$ drawn from L_2 is almost surely not entirely in the null space of this operator, and so $y_k \neq y_k'$.

The RMPI is born from this idea of making all signals have a unique signature after the mixing and integration. It is also motivated by the approximation of a signed Bernoulli matrix, as was mentioned in the introduction. Before going into further details of the system, we pause to codify the two guiding principles of the RMPI, which will be essential to bear in mind as we depart from the idealized RMPI to make a realistic system.

Principle 2.2.1 (Uniqueness)**.** *All distinct signals in the signal class should produce distinct measurements.*

Restriction to the signal class is required, since for undersampling, it is not possible to produce distinct measurements for all signals.

The next principle means that noise will effect all measurements about the same.

Principle 2.2.2 (Democracy)**.** *All equi-energy inputs should produce measurements of similar energy.*

These ideas can be interpreted as the RMPI analog of the "minimum distance" concept used in coding theory [McE02].

Practical version. The first design question for the RMPI is choosing a suitable chipping sequence $c(t)$. Since c must be known to the reconstruction algorithm, it must be predictable, and furthermore, it must be implementable. A reasonable choice is choosing c to be a pseudo-random bit sequence (PRBS), which flips pseudo-randomly between $+1$ and -1. The flips always occur at a multiple of the Nyquist grid ΔT, so that they may be controlled by a digital clock running at f_s. Technology for generating a chip sequence is available since this is the same principle behind spread-spectrum systems [PSM82]. In fact, the RMPI may be thought of as a spread spectrum device, using spreading in order to increase robustness to jamming and interference from foreign signals. Figure 2.4 shows a frequency domain representation of what this mixing looks like. The PRBS is discussed in detail in §2.5. Since our PRBS operates at the rate f_s, the RMPI system is a high-rate system, and it may appear to have just shifted the burden from high-rate sampling to high-rate mixing. This is true, but high-rate mixing is actually much easier than high-rate sampling, so the trade is worth it. The clock jitter introduced in the mixing problem is an issue, so our design is careful to keep the standard deviation of jitter to less than 0.5 ps; see §2.7.2.7.

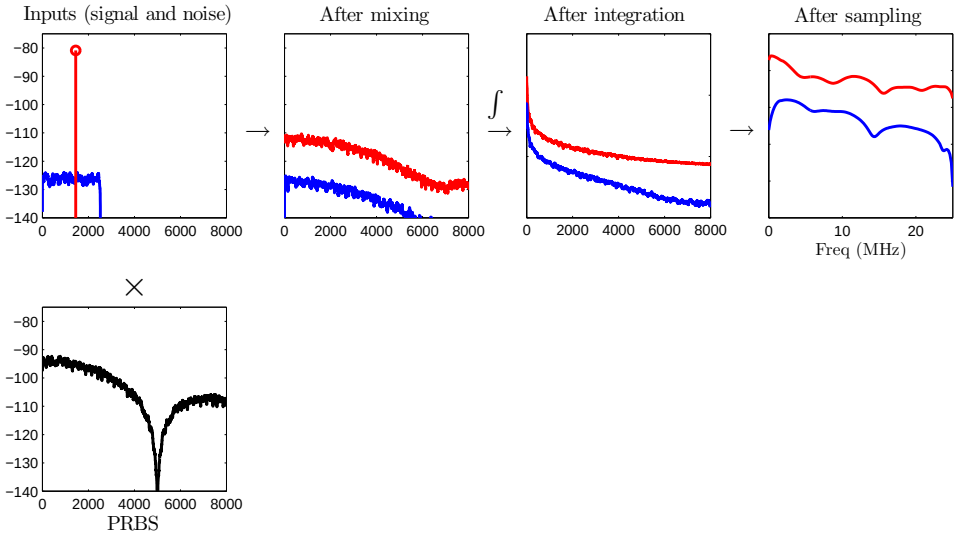


Figure 2.4: Frequency domain picture of one channel of the RMPI system. Red represents a signal, blue represents band-limited noise. The inputs are shown in the upper left, and are then convolved with the spectrum of the PRBS (shown below in black). Continuing to the right, the plots represent the signal after this mixing, then after integration (which attenuates high frequencies; the y-axis scale is shifted here), and then after the low-rate sampling which aliases high frequencies down to low frequencies. The x-axis of every plot is in MHz.

The next stage of the device is integration. This relies on capacitors, which actually charge and discharge at exponential rates. If this rate is too fast, then the integration becomes similar to a point sample, and as we argued above, point samples do not distinguish between pulses that have carrier frequencies separated by multiples of f_{ADC}. Another reason point samples are bad is that our signals include pulses, so they are sparse in time, and any sparse sampling mechanism will be coherent with these signals. For example, a pulse with duration less than T_{ADC} could possibly be

missed altogether if it was directly sampled (even after the mixing stage). To think about this in the Fourier domain, an ideal integrator has transfer function

$$H_{int}(s) = \frac{1}{s}$$

which means it attenuates high frequencies. A capacitor is similar to single pole filter

$$H(s) = \frac{1}{s+a}$$

for a pole frequency a. At high frequencies, this has little effect, but at low frequencies $f \ll a$, it means the response is nearly flat, like a band-pass filter. This can cause problems: in the extreme case, if the response is flat in the whole band, it means that $h(t)$ is a dirac, and this is just point sampling, which performs poorly. The location of this pole is a major factor of determining the performance of the system, and is described in detail in §2.6.

We briefly mention one more component of a practical RMPI, the low-noise amplifier (LNA). This amplifies the input signal from an antenna to make it suitable for processing. Though not part of the actual compressed sensing of the chip, it is still a crucial component in a useful hardware device, and it is what makes the RMPI a true receiver and not just an ADC.

2.2.2 Theoretical performance

The results from compressed sensing theory give an idea of what performance to expect from the RMPI. This analysis assumes the signal is perfectly sparse in the Fourier dictionary, which is not true, but the assumption simplifies the error estimates and gives meaningful qualitative relationships. Let Φ be the measurement matrix of the system, and suppose measurements are of the forming

$$b = \Phi x + \sigma_{\text{RMPI}} z$$

where $z \sim \mathcal{N}(0,1)$. Assume $x \in \mathbb{R}^n$ has exactly $k/2$ on-grid pure tones, each of rms amplitude $V_{\text{rms}} = A/\sqrt{2}$, and we observe for time T. The frequencies and phases are unknown, so there are K degrees-of-freedom. In this section, $\hat{x}$ refers to an estimate, not the Fourier transform.

Then if k is small enough, the ℓ_1 recovery gives an estimator $\hat{x}_{\text{RMPI}}$ such that

$$\text{MSE}(\hat{x}_{\text{RMPI}}) \leq c\frac{k}{nr_{\text{ch}}}\sigma^2_{\text{RMPI}} \tag{2.2.1}$$

for some constant $c \simeq 1$, where we define $\text{MSE}(\hat{x}) = \mathbb{E}\|\hat{x} - x\|_2^2/n$. Later sections of the chapter will define MSE in a different normalized sense, but we stick with the above definition in this subsection.

This bound (2.2.1) can be found in [CT07a] for the case of using the Dantzig selector for recovery,

or using [CP07a] (which gives a bound on $\|\Phi(x-\hat{x}_{\text{RMPI}})\|_2$) in conjunction with assuming the RIP on Φ. We show below a heuristic argument that derives this bound, based on the canonical compressed sensing results.

Let $A_i \in \mathbb{R}^{m_i \times n}$ be a Bernoulli ± 1 matrix scaled to have unit-normed columns, and let $b_i = A_i x + \sigma z_i$ with $z_i \sim \mathcal{N}(0,1)$ and x is k-sparse. Then

Theorem 2.2.3 ([Can08] plus [BDDW08]). *If $m_i \geq \bar{c}k\log(n/k)$, then with high probability*

$$MSE(\hat{x}_i) \equiv \|x - \hat{x}_i\|_2^2/n \leq c\|z_i\|_2^2/n = c\frac{m_i}{n}\sigma^2.$$

Suppose we really have a matrix $\Phi = [A_1; A_2; \cdots; A_L] \in \mathbb{R}^{m \times n}$ with dimensions $m = Lm_1$ where $m_1 = m_2 = \ldots = m_L = \bar{c}k\log(n/k)$, and $b = [b_1; \cdots; b_L]$ where $b = \Phi x + \sigma z$. If we form an estimator $\hat{x}$ by averaging the $\hat{x}_i$, then we improve the MSE by $1/L \leq 1$ since we are averaging the errors z_i:

$$\text{MSE}(\hat{x}) \leq c\frac{m_1}{nL}\sigma^2.$$

Now, we wish to allow arbitrary scalings of Φ. For RMPI, the average of the squared column norms in Φ_{RMPI} is r_{ch} (this is just $\|\Phi_{\text{RMPI}}\|_F^2/n = \text{trace}(\Phi_{\text{RMPI}}^T\Phi_{\text{RMPI}})/n$).

The average of the squared columns norms of the Φ constructed above is L. Thus by change-of-variables, using a scaling of $\sqrt{L/r_{\text{ch}}}$ for the noise z, we recover (2.2.1):

$$MSE(\hat{x}_{\text{RMPI}}) \leq c\frac{m_1}{nr_{\text{ch}}}\sigma^2 = \tilde{c}\frac{k\log(n/k)}{nr_{\text{ch}}}\sigma^2 \simeq \tilde{c}\frac{k}{nr_{\text{ch}}}\sigma^2. \tag{2.2.2}$$

Note that Φ *gains* the signal in power by r_{ch}, which improves the MSE.

According to numerical simulations (see Figure 2.5), the constant $\tilde{c}$ for the 8 channel RMPI is about .39.

$$\text{MSE}(\hat{x}) \simeq .389\frac{k}{n}\sigma_{\text{RMPI}}^2 = \frac{3.11}{r_{\text{ch}}n}k\sigma_{\text{RMPI}}^2 \tag{2.2.3}$$

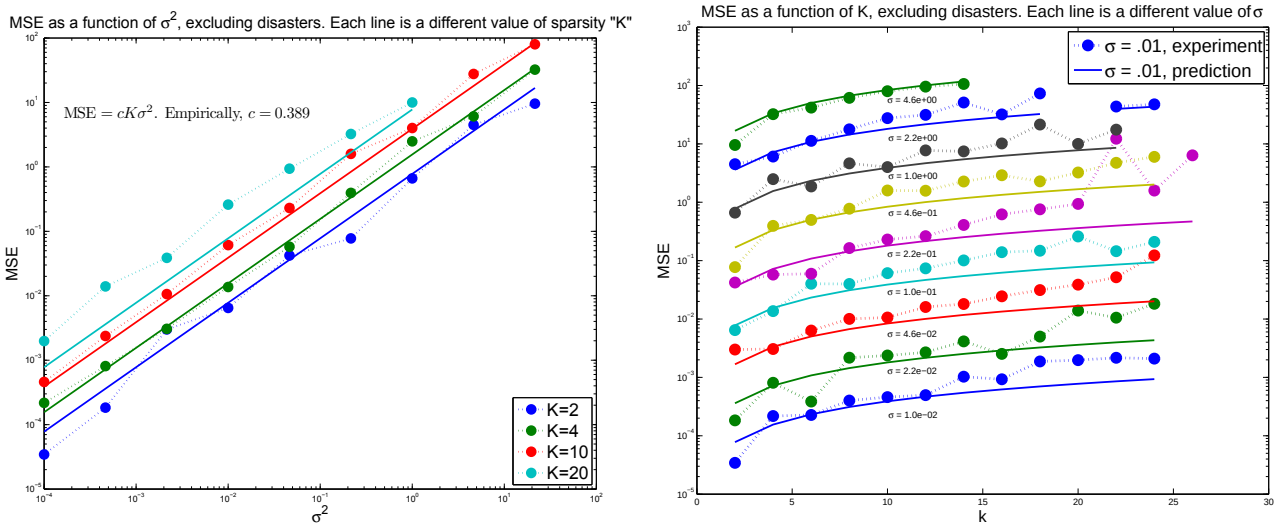
2.2.2.1 Input noise and channelized receivers

In a well-designed system, error should be dominated by input noise:

$$b = \Phi(x + \sigma_{\text{input}}z), \quad z \sim \mathcal{N}(0,1).$$

This is the same as modeling $y = \Phi x + \tilde{z}$ for a modified noise vector $\tilde{z}$ where $\tilde{z} \sim \mathcal{N}(0, \sigma_{\text{input}}^2\Phi\Phi^T)$. Then

$$\sigma_{\text{RMPI}}^2 \equiv \mathbb{E}\|\tilde{z}\|_2^2/m = \sigma_{\text{input}}^2\frac{n}{m}r_{\text{ch}}.$$



(a) MSE as function of noise variance

(b) MSE as function of sparsity k

Figure 2.5: Simulations to find an estimate of the constant in (2.2.1). Shown are averages over 10 trials of non-catastrophic recoveries, using k pure tones (on-grid, but random phase), $n = 1000$, and $m = 100$ measurements. Matrix Φ modeled with ± 1 matrix. Used re-weighted ℓ_1 using [GB10].

Let $\rho \equiv \frac{m}{n} = \frac{1}{12.5}$ be the under-sampling ratio. Thus the error of the RMPI in terms of *input* noise is

$$\mathrm{MSE}(\hat{x}_{\mathrm{RMPI}}) = 3.11 \frac{k}{\rho n} \sigma^2_{\mathrm{input}}. \tag{2.2.4}$$

Is this error good? We compare it to three other systems: a Nyquist-rate system with infinite precision ADCs and no post-processing; the same Nyquist-rate system but in a so-called "channelized" architecture; and a smart "oracle" system that does post-processing.

Nyquist-rate ADC. Neglecting quantization error, if we could make a high-bit ADC at this rate f_s (which is *not* possible), then with no post-processing, the error is

$$\mathrm{MSE}(\hat{x}_{\mathrm{ADC}}) = \sigma^2_{\mathrm{input}} = \frac{\rho n}{3.11 k} \mathrm{MSE}(\hat{x}_{\mathrm{RMPI}})$$

which is terrible compared to the RMPI, since this grows with n

Channelized receiver. Channelized receivers are the current state-of-the-art for high bandwidth systems; see Figure 2.6 for a frequency domain depiction. The input signal x is split into many channels, and each channel processes a small portion of the spectrum after downconverting to baseband via the heterodyne principle discussed in the introduction. For concreteness, let a channelized receiver split the 2.5 GHz of bandwidth into 50 chunks of 50 MHz each. There are two advantages to this: first, each channel processes only 50 MHz of bandwidth, so it can use a 100 MHz ADC which is lower power and more accurate than a higher rate ADC; second, each channel sees only 50 MHz

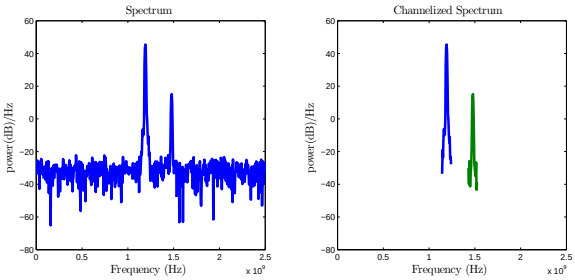


Figure 2.6: Left: two pulses with noise in 2.5 GHz of bandwidth. Right: channelizing the input into channels of width 50 MHz. Only channels with power above a threshold are recovered. The out-of-band noise around the pulses is removed, which helps the SNR.

of noise. Assuming the noise is white, the noise power is directly proportional to the bandwidth B:

$$\sigma^2_{\text{input}} = B\sigma_0^2.$$

If there is only one input tone, then only the channel that contains this tone is used to collect data, so the noise power is proportional to the 50 MHz bandwidth. In contrast, the RMPI is inherently wideband and sees the entire $B = f_s/2$ band, so it has 50 times more noise-power, and hence 17 dB ($=10\log_{10} 50$) less SNR. The MSE for a 50 channel ADC, where σ_{input} is defined over the entire bandwidth $f_s/2$, is

$$\text{MSE}(\hat{x}_{\text{ADC}}) = \sigma^2_{\text{input}}/50 = \frac{\rho n}{50 \cdot 3.11k}\text{MSE}(\hat{x}_{\text{RMPI}})$$

which still grows with n compared to the RMPI MSE.

In practice, the bands should overlap slightly, which requires extra channels. Since each channel requires an ADC, and there are 50 or more channels, this design is extremely power hungry, and uses 50 to 200 watts, depending on the specifics of the ADCs. For comparison, the RMPI uses on the order of 1 to 4 watts.

Oracle. Consider the (un-channelized) infinite precision Nyquist rate samples again. If we knew in advance which tones were non-zero (such information is called *oracle* information), then with post-processing denoising techniques, the error is

$$\text{MSE}(\hat{x}_{\text{ideal}}) = \frac{k}{n}\sigma^2_{\text{input}}.$$

This does not change if the input is from a channelized receiver, since the denoising process reduces out-of-band noise already.

Comparison. Comparing the RMPI to the non-post-processing techniques is tricky. The RMPI error is a function of the information content in the signal. For a fixed information content, the

RMPI error improves as more samples are collected. With the Nyquist-rate receiver, there is no estimation occurring, and more samples do not improve the estimate.

The RMPI penalty compared to an *oracle* pays only a penalty of $3.11\rho^{-1} \simeq 15.9$ dB. This is quite reasonable, since the oracle result requires knowing the frequency of the signals and having an ADC that samples at f_s with very high ENOB. This comparison is only valid when k is sufficiently small.

2.2.3 Related literature

Since 2006, there have been several analog-to-information (AIC or A2I) systems discussed in the literature. A few such systems are reviewed below. Many of these papers contain major ideas but offer few details. All prior work has assumed exact spectral sparsity, and used the Fourier basis for reconstructions; this assumption is relaxed in this thesis. Systems that have been implemented in hardware will be specifically mentioned. Our RMPI design has been influenced by conversations with many people, but among published papers, [TLD+10] was most influential.

2.2.3.1 Other RMPI systems

The RMPI was proposed in 2005 [SOS+05], and has been discussed in the literature by a team at Rice University [KLW+06] in 2006, and further discussed by the same group in [LKD+07, KRL+06]. The first main theoretical work was the "Beyond Nyquist" paper [TLD+10] discussed below. Another DARPA group led by Hoyos has also proposed multi-channel RMPI designs [YHS08, CYH+11].

The initial work [KLW+06] proposed the basic RMPI design, including the concepts of a PRBS (and declared the rate must be at f_s or higher), mixing, integration, and sampling. The signal model assumes that signals are on-grid sinusoids or sparse in some ortho-basis. Notably, they propose a high-level calibration scheme. The design was further expanded in [LKD+07], which assumes the same on-grid signal model, although it also discusses the Gabor dictionary. The authors implement a transistor level SPICE simulation of the basic system, though without any front-end amplifier, and only at schematic level since there was no layout. The implementation includes non-idealities at a few stages, although their estimates of the limiting non-idealities differ than what we consider the most important; according to their design, the non-linearity of the mixer is the most significant non-ideality, since they do not model the LNA. The integrator is allowed to be a single pole, though the pole is set to 100 Hz which may be quite optimistic. It is not clear if thermal noise is included. Using the end-to-end simulation, a superposition of on-grid tones is reconstructed and the spectrum of the reconstruction looks reasonably accurate.

The work [KRL+06] was a precursor to [LKD+07] and had simulation results from SPICE, and built simpler non-SPICE behavioral models to mimic the effects of non-idealities. They tested the effect of a nonzero pole in the integrator, and found that performance degraded as the pole moved

farther from DC, although this may have been using a non-calibrated model, in which case this is due to model mismatch and not any intrinsic effect of the pole. Their dominant non-ideality was the non-linearity in the mixer, which was more significant than clock jitter in the mixer. Our RMPI has been designed carefully to keep the mixer very linear, and we find that the LNA non-linearity is just as significant. The work of [KLW+06,KRL+06,LKD+07] is due to a Rice University team. Rice University has a contract with the same DARPA BAA as the Caltech team, and they are actively working to implement a type of compressed sensing system in hardware, though their device has very different bandwidth and recovery goals than the RMPI [RLN+08].

The first published work on a multi-channel design appears in a report by a team at Texas A & M University [YHS08]; this application is tailored for spectrum sensing in cognitive radio. These authors also have a contract with the DARPA BAA, so it is not surprising that their design, the Rice University design, and the Caltech design are all related. Their methodology is called the "Parallel Segmented Compressed Sensing" (PSCS) structure, and at a high level, it is identical to the RMPI. They cite worries over the "10th order" integrating filter in [LKD+07] (though this may be due to a misinterpretation), and suggest that the single-channel design's samples are redundant, which can be fixed with the multi-channel design. Regarding the number of channels, they suggest that an upper limit is necessary, due to practical implementation issues. Notably, they introduce a stagger to the channels, suspecting that samples would have higher fidelity in the interior of a sampling period and thus the edges of the channels should not align. Using orthogonal matching pursuit (OMP), they demonstrate basic feasibility with Matlab simulations using on-grid tones and a signed Bernoulli matrix.

The same team recently published [CYH+11], detailing a 90 nm CMOS chip they plan to fabricate. The layout was fully simulated in Cadence Spectre. The chip consumes 120 mW per channel, which includes specs for ADCs, although they assume that a LNA is off-chip and that clock distribution, which typically accounts for 30% or so of power, is also off-chip. Their signal model assumes on-grid tones, of which 4% are occupied. They predict a 44 dB SNR. The effective bandwidth is 1.5 GHz, using 8 channels, each sampling at 110 MHz. This is very similar to the Caltech design, though less ambitious in terms of the undersampling ratio (they undersample by $\times.587$, our design undersamples by $\times.08$). Similar to what we will find in this chapter, they conclude that mixing jitter is a limiting non-ideality, and assume their jitter will have 0.5 ps rms (same as our design). Non-DC poles in the integrator are not discussed, but they mention using a dual-integrator design which samples on one branch while resetting the other branch, then swapping, so that railing/clipping of the integrator is not an issue. They briefly discuss calibration. Like in the earlier paper, signal recovery is done straightforwardly via OMP and a Fourier dictionary, since off-grid tones are not considered. A thorough analysis of noise at different blocks is presented. Due to the similarities with the Caltech RMPI, it will be interesting to see the results once their chip is fabricated, since

both teams face similar problems.

We note recent work by [TV11] that considers parallel branches of mixers and integrators, which is similar to already proposed designs except for an extra step at the sampling which introduces extra samples; the extra samples are correlated to earlier samples, but they claim it gives processing gains. Another new A2I device has been mentioned in the dissertation of Michael Trakimas at Tufts, but we are unaware of any papers on the subject and so do not know the details. The recent work of [MPAL11] also proposes a RMPI design, and they have constructed a prototype in 500 nm CMOS with 200 MHz of bandwidth, and appear to be planning construction of a 130 nm device. They demonstrate on-grid recovery with Matlab simulated data.

Beyond Nyquist. The paper [TLD+10] called "Beyond Nyquist" is probably the most influential A2I paper, so we give it an extended discussion. Though authors of the paper are associated with both the Rice and Caltech hardware projects, the paper itself is mainly theoretical. Their setup is identical to the model considered in this chapter, except they only consider a single channel, and with different assumptions on non-idealities and practical considerations. Using a particular signal model, they prove the first rigorous theorems, which state, in effect, that the RMPI mixing operations are not much worse than using a standard compressed sensing matrix like a random signed Bernoulli matrix. This signal model is not overly realistic, but it is important because the result suggests that the operations of the system adhere to the compressed sensing principle, and follow a phase-transition similar to the Donoho-Tanner phase transition for Gaussian matrices.

Specifically, assume x is a periodic signal with K on-grid tones. For example, amplitude-modulation of on-grid signals falls into this class, while frequency-modulation does not. Their model uses a single channel which undersamples the Nyquist rate by the amount M/N. Empirically, they suggest that recovery happens when

$$M = 1.7K \log(N/K + 1).$$

This is complemented by two theoretical results, which we state here. Fix a time T and let $N = T/\Delta T$. Write X for the unitary DFT of the discrete signal x.

Theorem 2.2.4 (Noiseless recovery of a random on-grid random signals, [TLD+10])**.** *Suppose x has K on-grid frequencies chosen uniformly at random, so X has up to $2K$ non-zeros, including the negative frequencies. Let the non-zero entries of X have arbitrary amplitudes, and random complex phases. Pick a PRBS sequence at random. Then a single-channel ideal RMPI with M measurements (and assume M divides N)*

$$M \geq C \left(K \log(N) + \log^3(N) \right)$$

will exactly recover the signal from the noiseless measurements

$$b = \Phi x, \quad \Phi \in \mathbb{R}^{M \times N}$$

by solving basis pursuit (1.1.5)*, except with probability* $\mathcal{O}(N^{-1})$*. The constant* C *is independent of* N *and* K*.*

This theorem might be interpreted as saying that most PRBS sequences will work to reconstruct most inputs, as long as M is large enough. However, since the PRBS sequence is typically fixed and not changed, we might wonder if there is a big risk of choosing a "bad" PRBS. It turns out as long as the PRBS is initially selected at random, then most likely, recovery will still happen. Furthermore, this is robust to additive noise, and adapts to compressible signals. To be specific:

Theorem 2.2.5 (Noisy recovery of arbitrary signals, [TLD+10])**.** *Fix any PRBS sequence. Let*

$$M \geq CK \log^6(N)$$

and assume M *divides* N*. Take measurements*

$$b = \Phi x + \eta$$

and estimate $\hat{x}$ *be the solution to the basis pursuit denoising problem* (1.1.6) *with parameter* $\varepsilon = \|\eta\|_2$*. Then except with probability* $\mathcal{O}(N^{-1})$*,*

$$\|\hat{x} - x\|_2 \leq C \max\left\{\varepsilon, \frac{1}{\sqrt{K}}\|X - X_K\|_1\right\}$$

where X_K *is any* K*-sparse vector in* $\mathbb{C}^N$*, i.e., the best approximation of* X*.*

Another improvement of this theorem is that it doesn't require x to be exactly K sparse in frequency; good reconstruction occurs as long as its DFT is compressible. However, a major issue still happens for off-grid tones. These tones are not periodic, so instead of the DFT, it is appropriate to analyze their continuous Fourier transform. Because only a finite time sample T is observed, this has the effect of a boxcar filter in the time domain, which means the Fourier transform of the time-limited function will show a sinc decay away from the active tone. Since the sinc function decays like $1/f$ in frequency, it is not well-approximated by a sparse DFT.

This is a common problem in signal processing, and the usual solution is windowing, as will be discussed in detail in §2.7.2.6. The "Beyond Nyquist" paper proposes implementing a window in hardware via a time-varying amplifier; furthermore, because two shifted windows are needed, the number of channels must double. Whether such a scheme is practical remains to be seen.

Though calibration is not discussed in detail, the paper suggests that non-ideal integrators should be acceptable as long as they can be characterized well, which agrees with the findings presented in this chapter. Some Matlab simulations are presented using IRLS.

2.2.3.2 Related systems

NUS. The non-uniform sampler (NUS) was described in the introduction. It is ideal for sensing signals composed of only a few frequencies or frequency bands, such as in typical narrowband modulation schemes. In [ME10], a multi-co-set sampling strategy is discussed, which is effectively the same as NUS except with periodicity of the samples made more explicit, and uses a bank of ADCs (each sampling at the same rate, but offset from each other in time). It is suggested that a drawback of the system is that it needs specialized implementation, since even though the ADCs only need to sample at a low-rate, they still sample a high-bandwidth signal. Since ADCs are designed to sample signals that have already been low-pass filtered to the ADC sampling rate, this would introduce errors due to parasitic response in the ADC. The Northrop-Grumman NUS overcomes this by a specialized implementation which includes their own sample-and-hold block.

Random convolution. The random convolution has been proposed in 2006 by [TWD$^+$06] and analyzed in detail in [Rom09]. In spirit, it is quite similar to the RMPI: the idea is that a signal is convoluted with a random complex point-spread function that changes the phase of the spectrum while keeping magnitudes the same. Following this, two schemes are analyzed: either sampling the output directly, or sampling a Rademacher sum (similar to the RMPI). The latter technique is shown to be a universally good system, meaning that the input can be sparse in any fixed ortho-basis.

The random convolution and the random sums make the system more complicated than the RMPI, and it has not yet been implemented. Applications are suggested in radar sensing (as opposed to passive radar observation like the RMPI), and Fourier optics. In the latter setting, it is assumed the receiver can detect the phase of the light, which may make it less enticing in practice.

Multiband sampling theory. The "Xampling" methodology [ME10, MEE09, Eld09, MEDS11], which follows in the steps of the authors' blind multiband sampling theorem [ME09a], is similar to the RMPI in many respects. Unlike most other proposals, a prototype has been developed in the lab [MEDS11]. The signal model is that x has only N bands of frequency, and each band has size at most B, so the overall bandwidth is NB, and it is assumed $NB \ll f_s$. The approach is implemented in a "modulated wide-band converter" (MWC) which splits a signal into channels, mixes each channel with a unique periodic sequence, then low-pass filters the output before sampling at a low rate. The periodicity of the mixing sequence, the rate of the sampling, and the number of channels are discussed in detail in [ME10]. The design works for both known and unknown

(a.k.a. "blind") bands.

If the band locations are unknown, then after collecting a reasonable number of samples (for example, 1 to 10 μs worth), an ℓ_1 block sparsity problem is solved to find locations of non-zero blocks of spectrum; this is referred to as the "continuous-to-finite" (CTF) block. The computational problem may be reasonably sized, though perhaps not possible for real-time implementation. This processing is cheapest when the bands are large; if there are many bands that are very small and evenly spread out, the computational problem may be expensive. In the follow-up paper [MEE09], the authors describe a digital processing step that recovers the base-band versions of the occupied frequency bands.

The methodology behind Xampling is motivated by working directly with analog signals, and doing all processing at base-band, rather than Nyquist rate. The work seems promising, though it may be best adapted for spectrum sensing, such as in cognitive radio, rather than full recovery; or suited for recovering signals with known types of modulation (such as QPSK). Another key feature of the model is that the system is relatively simple to implement and does not rely on delicate hardware; to this extent, they seem to have achieved the goal admirably.

The hardware of both the MWC and the RMPI are not too dissimilar, the main difference being the chipping sequence rate, which is much higher for the RMPI case. The essential difference of the system is the processing. The RMPI makes full digital processing, while the MWC uses digital processing only to identify the support, and then works with base-band digital processing. Our view is that this methodology is exciting and worth further investigation. However, below we detail a few minor criticisms of the MWC and address some comments from [MEE09] about the RMPI.

We first note the difference in signal models, which is simply a matter of application. Both the MWC and RMPI do poorly when analyzed using the wrong signal model. The MWC assumes a static spectral occupancy. Whenever the spectrum changes significantly, the CTF block must recompute the spectral support. Furthermore, this spectral estimation requires many time samples, and would give inaccurate estimates for time-dependent inputs like a radar pulse. Similarly, the RMPI adds unnecessary complexity and processing if applied to the problem of finding signals with infinite support. It doesn't exploit continuity in the spectrum from one reconstruction window to the next, and it does not automatically offer full analog resolution since it works with finite discrete samples.

It is mentioned that RMPI systems have a high computational load, and this is true. Real-time processing of the RMPI is not yet achievable, though as this thesis shows, the processing time is reasonable. The standard RMPI reconstruction works at the Nyquist rate, although the compressive matched filter doesn't rely on this, so is closer to the baseband processing suggested in Xampling. Other valid criticisms are the lack of flexibility for changing the bandwidth f_s on-the-fly, and sensitivity to windowing issues. In particular, the on-grid tone model is viewed as too academic.

For the radar signals studied in this thesis, windowing and on-grid frequencies are not a major issue since the radar envelope is itself a window with finite bandwidth. More about windowing is discussed in §2.7.2.6. It is true that if our RMPI were restricted to measuring on-grid pure tones, it would perform much better, but we find adequate performance without this assumption.

It is suggested that in the RMPI, any deviation of the integrator filter from being a true integrator will introduce signal-dependent errors, and the system really must be calibrated *continuously*! This is quite misleading. The RMPI system certainly needs to be calibrated, but this can be done once. In particular, the post-processing does not rely in any way on the ideal integrator model. The calibration will not be exact, and the error due to this will be signal dependent, but this is not necessarily dangerous, and is inherent in many hardware devices, including the MWC. For example, the system may be mis-calibrated because of an absolute timing shift, but this is harmless because the reconstruction will just be a time-shifted version of the input.

Cognitive radio. Cognitive radio systems are designed to sense a broad range of RF spectrum to detect unoccupied channels, and as such, this technique is not tied to a particular hardware implementation. For example, both the MWC and the RMPI can be used for this. Many papers on this area have appeared recently; see [ME10] for a discussion.

Nyquist Folding Receiver (NYFR). Proposed in [FBC+08] and also part of the same DARPA project as the RMPI, this design samples at a time-varying rate. For example, the sampling rate averages $f_{avg} = 2$ GHz, but varies sinuisoidally in time between 1950 and 2050 MHz. For input tones of high frequency, say $f_{in} = 10$ GHz, this induces aliasing. The aliased signal will change frequency over time since the sampling rate is changing over time. The amount this aliased signal changes over time is determined by what "Nyquist zone" the carrier signal was in. The Nyquist zone L is the smallest integer such that $f_{avg}L > f_{in}$. Thus even from aliased signals, it is possible to determine what the original absolute frequency was, since the time dependence introduces a splitting.

This design is examined in [FBC+08] for a few narrowband tones, and it is shown experimentally that their original frequencies can be successfully recovered; this is compared to a standard time-independent sampler, which cannot recover the original frequency. However, the implications are not clear, since many applications are not actually concerned with recovery of absolute frequency information and only need the narrowband content. It has not been shown that this design is robust to recovering arbitrary sparse tones, e.g., tones that are close in frequency, or tones with significant bandwidth that overlap two Nyquist zones. The method is claimed not to rely on digital Nyquist-rate ℓ_1 recovery, though the actually recovery algorithm is not discussed.

2.3 Modeling the system

This section describes how to simulate the system numerically. The process of describing the modeling should also make it clear what the RMPI does. The results in this chapter present many different types of simulations and models. Some models were improved over time, so preliminary simulations were necessarily not as accurate. The best models, that of the Simulink and SPICE simulations, are orders of magnitude slower than the simplest models, so in some cases we had to choose between a single accurate simulation and many less accurate ones. We discuss the modeling inaccuracies and when we think they may affect results.

There are two purposes behind modeling the system. The first is to generate realistic measurements for testing purposes. The second purpose is to create a model matrix Φ that the reconstruction solvers will use. In a single computational experiment, we may use two different models: for example, a highly accurate model incorporating non-idealities is used for the measurements, and a simpler model is used to generate Φ.

This section concludes with a discussion of calibrating the hardware device. This is extremely important, so some time is spent discussing the issues and various strategies.

First, we review the mathematical representation of the system. Unless otherwise specified, we will work with only a single channel, since including the representation from other channels is trivial (except perhaps for alignment issues during hardware calibration). Let $x(t)$ be in the input to the integrator, and $y(t)$ be the output. Then if $h(t)$ is the system's *impulse response* (or "Green's function" in PDE terminology), we have $y(t) = h(t) * x(t)$ where "$*$" is the convolution operator:

$$y(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau.$$

Our interest is in causal systems, which means $h(t) = 0$ for $t < 0$, so the convolution integral can be taken from $[0, \infty]$. Let $\mathcal{L}$ be the Laplace transform operator (in the subsequent chapters, this symbol will be used instead for the Lagrangian), defined by $\mathcal{L}(x)(s) = \int_{-\infty}^{\infty} x(t)e^{-st}dt$. The transfer function $H(s)$ of a system is defined as $H(s) = Y(s)/X(s)$, or equivalently as $H = \mathcal{L}(h)$. If the output is an integral, $y(t) = \int_{-\infty}^{t} x(\tau)d\tau$, and if $x(t)$ has finite support, then using integration-by-parts gives

$$h(t) = u(t) = \begin{cases} 1 & t \geq 0 \\ 0 & \text{else} \end{cases}, \qquad H(s) = 1/s.$$

Another important system is the single-pole system:

$$h(t) = \begin{cases} e^{-at} & t \geq 0 \\ 0 & \text{else} \end{cases}, \qquad H(s) = \mathcal{L}(h) = \frac{1}{s+a}.$$

The frequency response of the system to a frequency f is $H(if)$, since this is just the Fourier transform of h. A fundamental fact of linear time-invariant (LTI) systems is that they can be represented exactly by an impulse response. The RMPI system is linear but not time-invariant since it is periodically sampled. Restricted to within a sampling period, the system is LTI. When using the $H(s)$ terminology, we are implicitly restricting our attention to a single sampling period and assuming that the input signal is 0 before the period of interest.

Let $c(t)$ be the continuous time version of the chipping sequence (a.k.a. pseudo-random bit sequence PRBS). For now, H and h may be arbitrary, but are still called the "integrating" filter since the system should be similar to an integrator. Then the 1^{st} output is

$$y_1 = \int_0^{T_{\text{int}}} c(t)x(t)h(T_{\text{int}} - t)dt = \sum_{n=0}^{N_{\text{int}}-1} c_n \int_{n\Delta T}^{(n+1)\Delta T} x(t)h(T_{\text{int}} - t)dt \tag{2.3.1}$$

and the m^{th} output is defined similarly, with integration limits $[(m-1)T_{\text{int}}, mT_{\text{int}}]$ and the discrete chip sequence shifted appropriately. In particular, h will be the same regardless of m; for this reason, the RMPI system is not an LTI system on large timescales, but for the period $[(m-1)T_{\text{int}}, mT_{\text{int}}]$ it is.

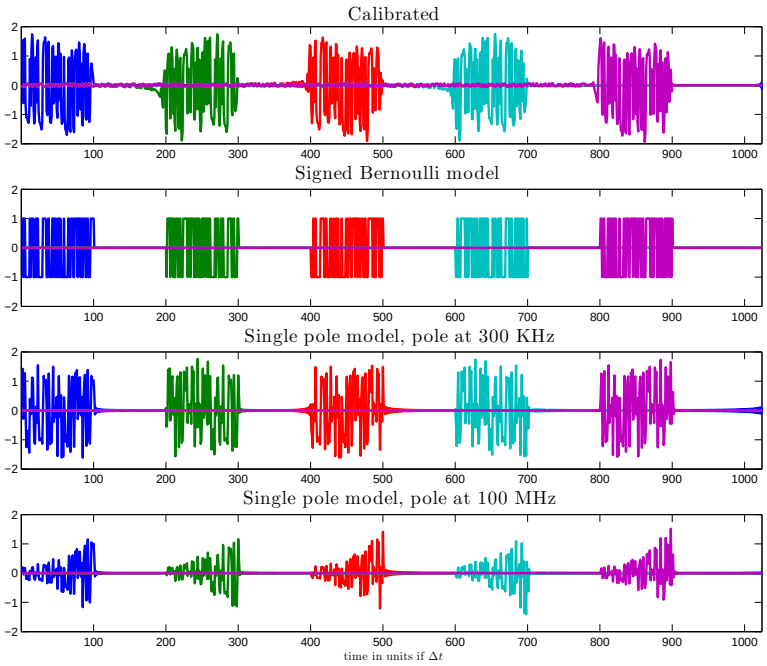


Figure 2.7: Showing samples for one channel of various models (showing every other sample for visual clarity), all with the same chip sequence. The Fourier-based models are not guaranteed to be causal. The entries of the single-pole model are not the same as windowing the Bernoulli model by e^{-at}.

2.3.1 Simple models

The simplest model of the system is as a block-diagonal matrix with ± 1 entries on the nonzero portions. Recall that the discrete vector x represents $x(t_n)$ for $t_n = n\Delta T$, and $n = 0, \ldots, N-1$; we may occasionally change between 0-based and 1-based representation when one representation is cleaner. Thus multiplication with a row c of this matrix is

$$b = \sum_{i=0}^{N_{\text{int}}-1} c_n x_n$$

which can be thought of as a quadrature approximation of the integral $\int_0^{T_{\text{ADC}}} c(t)x(t)dt$. For high frequencies, this is quite inaccurate. If instead we decide that x_n represents $\int_{n\Delta T}^{(n+1)\Delta T} x(t)$, then now this is an exact representation of the integral. However, this discrete representation is problematic, since it is not clear how to extract frequency information, and this representation does not represent the system if $h \neq 1$.

With the discrete representation of x, it is very easy to consider the discrete Fourier transform coefficients $\hat{x} = Fx$ since this is fast to compute and involves no loss of information. Using the conventions of Table 2.2 on page 45,

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} \hat{x}_k e^{i2\pi kn/N}. \tag{2.3.2}$$

Note that we can equivalently write

$$x_n = \frac{1}{N} \hat{x}_{N/2} \cos(2\pi(N/2)(n/N)) + \frac{1}{N} \sum_{k=-(N/2-1)}^{N/2-1} \hat{x}_k e^{i2\pi kn/N} \tag{2.3.3}$$

using the convention that $\hat{x}_{-k} = \hat{x}_{N-k}$. For convenience, we have chosen N to be even. Since x is real, $\hat{x}$ is conjugate-symmetric: $\hat{x}_k = \bar{\hat{x}}_{-k}$.

To approximate the integral in (2.3.1), the plan is to approximate $x(t)$ using $\hat{x}$. Recall that our pragmatic definition of the signal class was any signal that can be well approximated by its DFT, so now we put this assumption to use. If $x(t)$ is evaluated at an arbitrary point in time, rather than a grid point, the two inversion formulas (2.3.2) and (2.3.3) are not equivalent. Both of them interpolate x at the nodes t_n, but are different in between. One benefit of using the second formula (2.3.3) is that by the conjugate symmetry of $\hat{x}$, this guarantees that $x(t) \in \mathbb{R}$; using (2.3.2), there is no reason to expect $x(t)$ to be real-valued. The approximation is

$$x(t) = \frac{1}{N} \hat{x}_{N/2} \cos(\pi t N/T) + \frac{1}{N} \sum_{k=-(N/2-1)}^{N/2-1} \hat{x}_k e^{i2\pi kt/T} \tag{2.3.4}$$

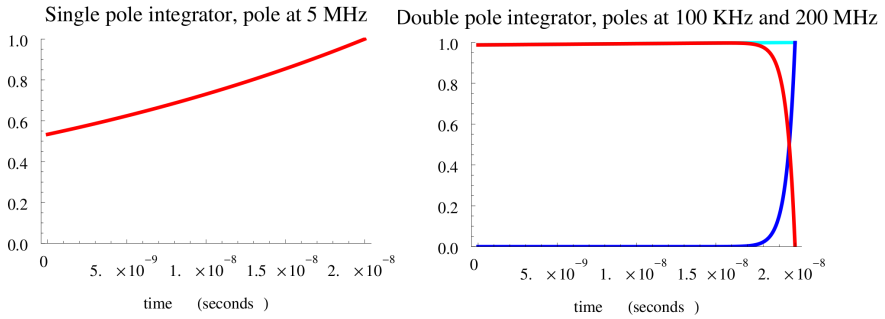


Figure 2.8: Time domain representation of integration. $h(T_{\text{ADC}} - t)$ for the integrator. Left: single-pole system. Right: double-pole system, showing the effects of each pole, and their combined effect (light blue)

for $T = N\Delta T$.

We can substitute this expression in (2.3.1) and then exchange the order of summation and integration. Thus

$$y_1 = \sum_{k=0}^{N-1} \hat{x}_k q_k \tag{2.3.5}$$

for some q which depends only on the discrete sequence c and the impulse response h. This is the form of an inner product, so we can build a representation of the system by stacking and shifting the q vectors for different time periods. This builds a matrix Φ_{FFT} which operates on $\hat{x}$, and hence a time-domain matrix representation of the system is $\Phi = \Phi_{\text{FFT}} F$ where F is the DFT matrix. The computation is easy to perform since $\Phi_{\text{FFT}} F = (F^* \Phi^*_{\text{FFT}})^*$, and applying F^* is equivalent to applying the inverse FFT (and scaling by N).

The computation of q, which needs only be done once, requires evaluating definite integrals of the form $\int e^{i2\pi kt/T} h(T_{\text{int}} - t)$. If h represents an ideal integrator, then $h = 1$ and the integral can be done in closed form; this is the system analyzed in [TLD+10]. If the poles and zeros of H are known, then h can be written concisely in terms of exponentials, and this can also be integrated easily.

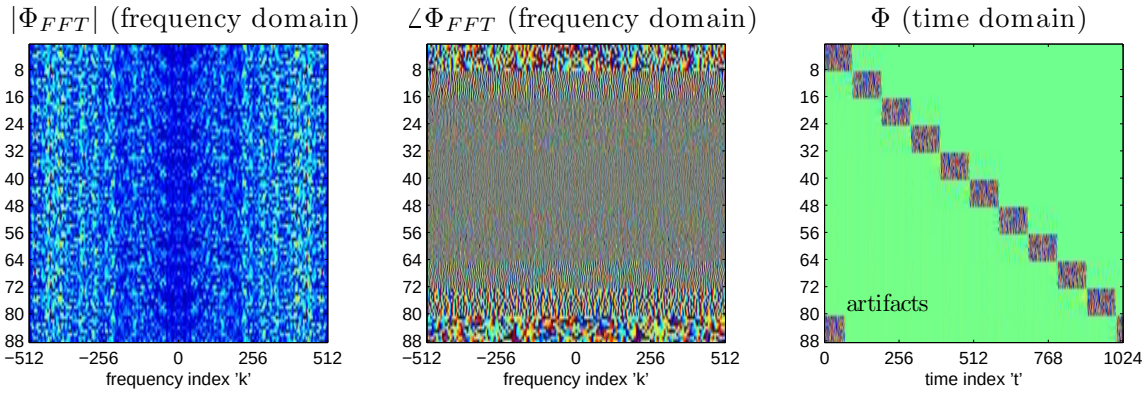


Figure 2.9: The left and center matrices show the magnitude and phase of the complex entries of Φ_{FFT}, which is designed to act on Fourier coefficients $\hat{x}$. The time-domain version Φ is shown on the right, and is generated by taking the inverse FFT of Φ_{FFT}.

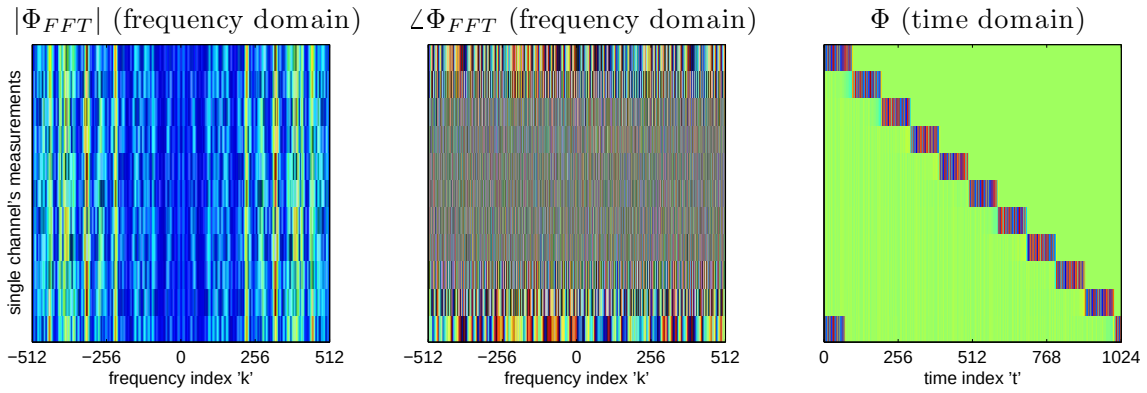


Figure 2.10: Showing the same matrix as Figure 2.9, but now only showing a single channel

The advantage of the previous model is that it easily extends to other versions of h and H. As we discuss in the section on integration §2.6, it is not possible to implement $H(s) = 1/s$, but rather we consider the *single-pole model* $H(s) = \frac{1}{s+a}$ for a pole frequency a. The corresponding impulse response is $h(t) = e^{-at}$ (where a is in Hz); this is sometimes characterized by the *time constant*, which is the time it takes h to decay to e^{-1}, so clearly the time constant is $1/a$. See Figure 2.8

There is a bizarre effect of using the frequency domain representation. In the simple Bernoulli model, a row of Φ consists of N_{int} nonzero entries. For example, the first row has nonzero entries in locations $1, \ldots, N_{\text{int}}$, and zero entries from $N_{\text{int}} + 1, \ldots, N$. This makes sense, because the system is causal, and the measurement y_1 taken at time T_{ADC} should only depend on $x(t), t \leq T_{\text{ADC}}$. However, when $\Phi = \Phi_{\text{FFT}} F$, there is no guarantee that the entries $N_{\text{int}} + 1, \ldots, N$ in the first row are zero. Typically, these entries are small but nonzero. This seems to violate causality. Really, there is no paradox, because the discrete vector x is no longer the time samples of $x(t)$ but rather just the inverse DFT of $\hat{x}$. Or, viewed differently, we know that the continuous signal $x(t)$ can be represented exactly by its DFT only if x is band-limited and periodic with period T, and if x is periodic there is no violation of causality. The matrix should not be "cleaned" by removing the non-causal portions, as this reduces the modeling power. An example of this effect can be seen by closely examining the gaps between samples in the top row of Figure 2.7.

A related phenomenon is shown in Figures 2.9 and 2.10. The last sample is incomplete, since it goes from $n = 1001$ to 1100, but $N = 1024$. Using the frequency approach, it appears as if the sample is still there, but has wrapped back to $n = 1, \ldots, 76$. This sample is inaccurate and we remove these rows from the matrix.

2.3.2 SPICE

The schematics and layout of the chip were done with the Cadence design suite. Cadence includes a number of circuit and transient simulators which allow high-accuracy simulations. In particular,

verilog-ams was used for behavioral models, and spectre-RF was used for AC analysis and transient simulations. These simulators include various forms of SPICE (Simulation Program with Integrated Circuit Emphasis), and rely on the device models from BSIM4 [BSI09]. For simplicity, we refer to these tools collectively under the generic label "SPICE".

Because a single license for Cadence is extremely expensive, these simulations cannot be run on a cluster and instead are run on a single high-performance server. Simulations with these models take on the order of a few minutes up to a few hours, depending on how many blocks are being simulated and which effects are included (e.g., just schematics, or considering the physical layout as well). Simulations can be used to determine parameters such as the frequency response of a specific block, or they can perform and end-to-end simulation of an arbitrary input.

2.3.3 Simulink

Because the SPICE simulation is slow, it is complemented with a behavioral model created in Simulink. The Simulink model also had the advantage of interfacing easily with Matlab, which is the language used for most of the experiments. The Simulink model consists of the following key blocks:

1. A low-noise amplifier (LNA) to gain the signal and remove out-of-band noise. After this point, the signal is split into 8 channels, and so there are 8 copies of each of the next components.
2. A transconductor (Gm) to convert the voltage signal into a current signal.
3. A mixer to multiply the signal with the chip sequence.
4. A common gate (CG) amplifier to provide current gain, which improves the noise figure.
5. An integrator.
6. A sample-and-hold block, and samplers.

Schematics of the entire system are shown in Figure 2.11, which includes various input sources, the LNA, the clock, channel blocks, and sampling. The schematics of a single channel block, which includes the mixing, CG, and integrator, are shown in Figure 2.12. The parameters for each component, such as the poles and zeros of the filters, were extracted from simulations with the SPICE model.

Additionally, the model included non-idealities, which could be turned on or off. White noise was injected into the LNA (this was the input referred noise, and takes into account the noise sources from the whole block), using an estimated value from SPICE. We also tested adding in noise directly to the individual components. The LNA, Gm, and CG were distorted by polynomials, with the non-linearity fitted by extrapolating to IIP2/IIP3 values taken from SPICE. Similarly, the frequency response of the LNA, Gm, CG,

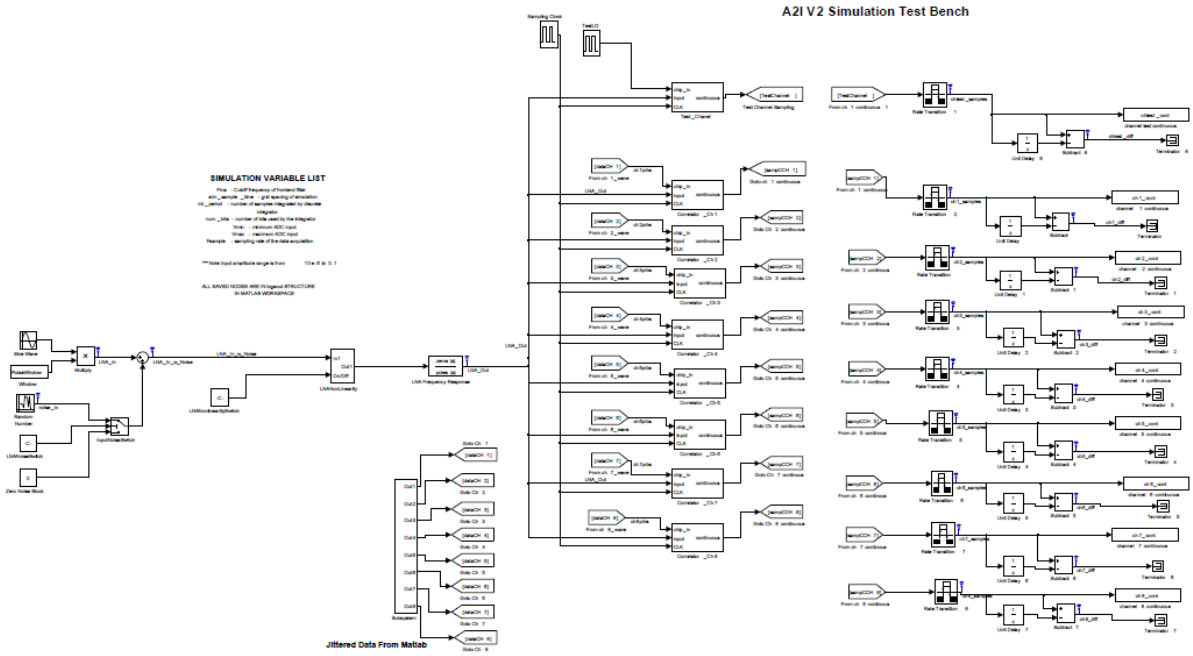


Figure 2.11: Simulink model. This was constantly updated as the chip design was updated.

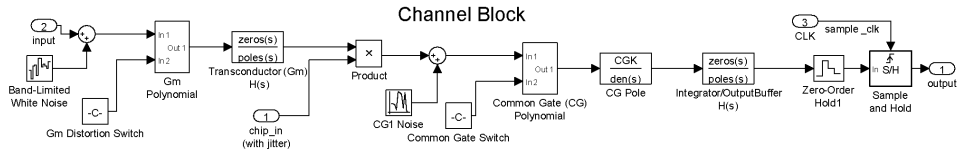


Figure 2.12: Details of a single channel in the Simulink model. The various sources of non-idealities are present, and were able to be switched on and off.

To solve the system, we set Simulink to use Matlab's `ode3` solver, with a fixed time step that we varied from 0.1 to 0.5 ps (recall the Nyquist time ΔT is 200 ps). Depending on the total period T and the size of the fixed time step, simulations took between 10 seconds and a few minutes, if the model was pre-compiled into binaries (compilation takes several minutes).

The Simulink model was used for two purposes:

1. With various non-idealities turned on, it generated test samples, which were fed into the reconstruction algorithms to test performance of the system.

2. With non-idealities turned off, it was used to generate a frequency domain model Φ_{FFT}, which in turn generates Φ. This process is referred to as "calibration" and is described in the next section.

The purpose of the first mode of use was to generate noisy samples. Specifically, this goes beyond the additive white Gaussian noise (AWGN) model, and includes non-linear effects such as the jitter

noise and non-linearities. The Simulink model is only an approximation, but it is useful as long as its noisy measurements test the robustness of the measurement and recovery process.

2.3.4 Calibration

The Simulink model, despite its flaws, is much more accurate than modeling the entire system as mixing followed by a single- or double-pole filter, as considered in in the simple models. To get the Simulink model to provide a matrix representation of the system Φ, we build up an "impulse response" by sending in a series of known inputs. Let S represent the system as a "black-box", which can be probed indirectly by sending in signals and recording the output. Furthermore, S is linear and approximately finite-dimensional, so we seek to represent it with a matrix Φ. If Φ is a $M \times N$ matrix and $S(x) = \Phi x$, then sending in any set of N linearly independent vectors into S and recording the output is sufficient to characterize Φ. This was initially tried with the unit impulse functions e_i, in order to provide the traditional impulse response. This approach does not work well since e_i have sharp rise and fall times, and thus slowly decaying Fourier coefficients, so they are not in the signal class. A much better approach is via the Fourier domain. Using the representation (2.3.4), we see that we need only the response of S to sines and cosines on the frequency grid. Thus we build a frequency-domain impulse response to Φ_{FFT}, and then invert via the IFFT to get Φ. The basis functions are $e^{i2\pi kt/T}$ for $k = -N/2+1, \ldots, N/2$. To measure $S(e^{i2\pi kt/T})$ we expand the complex exponential and use linearity to get $S(e^{i2\pi kt/T}) = S(\cos 2\pi kt/T) + iS(\sin 2\pi kt/T)$. Thus a single basis function requires two measurements, but since the negative frequencies can be inferred from the positive frequencies only N overall measurements are needed.

This procedure defines the calibration of the system. The response of the system is measured for each input from a set of $N/2$ cosines and $N/2-1$ sines (and also a DC signal); see Figure 2.13 for a diagram. A Simulink model represents the RMPI system, since for $N = 1024$, a SPICE simulation would take about a month of computation time. This approach is more accurate than the simpler single pole models, and has the added benefit that it does not require estimation of the location of poles or other parameters. Because this only requires a black-box system, it can use the actual hardware system to generate measurements; this is useful when the noise in the hardware is small and has less effect than the modeling errors in the simulators. Since the inputs can have arbitrary amplitude, they are chosen in the "sweet-spot" amplitudes which is small enough that non-linearities can be ignored yet large enough that thermal noise plays a small role.

To validate the calibration approach, there are two considerations.

1. Given a known input x, do the actual measurements $S(x)$ agree with the predicted measurements Φx?

2. Given a known input x with measurements $b = S(x)$, does the reconstruction $\tilde{x}$ (a function of

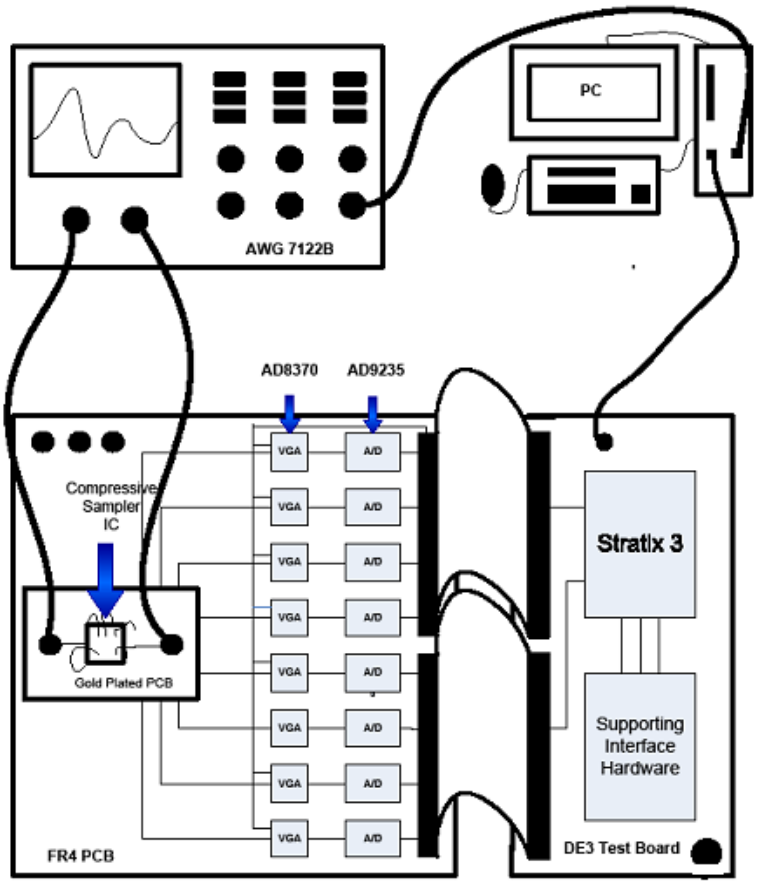


Figure 2.13: Cartoon of the calibration procedure. The arbitrary waveform generator (AWG) generates basis functions $x(t) = e^{i2\pi kt/T}$, the RMPI chip generates measurements which are recorded on the Stratix 3 FPGA, which then uploads to a computer. The computer controls the AWG and tells it which grid frequency k to use.

Φ and b) closely match x?

Note that the first item is more strict than the second item. If the second condition holds, and $\tilde{x} \simeq x$, this implies that $\Phi\tilde{x} \simeq S(x)$ but does not imply that $\Phi x \simeq S(x)$. For example, suppose Φ is an incorrect model of S because it uses a chip sequence which is shifted forward or backward in time by an amount $n\Delta T$. This has a strong effect on the measurements of any particular signal, and $\Phi(x) \neq S(x)$. However, this is not actually significant, since the recovery algorithm will just recover the estimate $\tilde{x}(t) = x(t + n\Delta T)$, which is a systematic error that is of little consequence.

Validation of the first type is considered in Figure 2.14, which plots the values of $S(x)$ and Φx for a signal x of medium (left) and small (right) amplitudes. The measurements S are performed with the Simulink model, and include a constant level of noise z. Also shown is $\Phi(x+z)$, which should be a better match to the Simulink model than just Φx; it does provide a better match, but not a perfect match, since white noise is not band-limited and is not well represented in the finite-dimensional signal model. Without the effect of noise, the agreement between $S(x)$ and Φx is excellent, but this

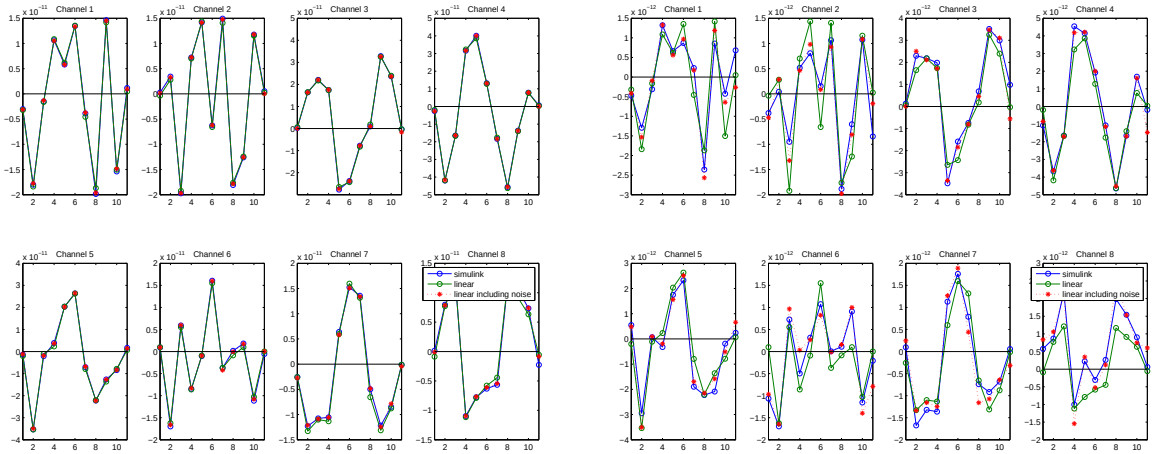


Figure 2.14: Testing if $S(x) = \Phi x$. Left: medium amplitude input. Right: small amplitude input, so noise has a significant effect. A large amplitude input is not shown because agreement is nearly perfect.

is because Φ was calibrated using the same Simulink model. In real hardware calibration, we would not expect such agreement.

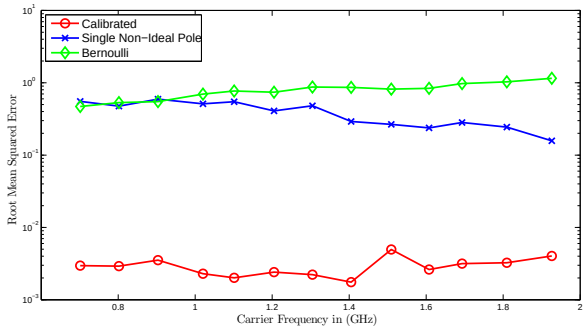


Figure 2.15: Reconstruction error for different types of calibration, as a function of frequency. Calibration is highly beneficial.

Figure 2.15 shows results from validation of the second type, plotting RMSE of the recovered signal, as a function of carrier frequency. Noisy samples $S(x)$ are generated with a Simulink model using non-idealities. Reconstructions are performed using three models of Φ: the simple ± 1 signed Bernoulli model, the single-pole model, and the calibrated model. As expected, the calibrated model performs much better. Also note that the single-pole model performs better than the Bernoulli model at high frequencies, which is reasonable since the Bernoulli model (which is equivalent to quadrature evaluation of an integral) is most accurate at low frequencies.

The value of N has so far been considered arbitrary. As N increases, the DFT approximation of the signal $x(t)$ becomes increasingly accurate, but typically a value $N = 1024$ provides sufficient approximation to the band-limited signal and it is not necessary to go larger.

However, there is a value $N_{\text{min}} = \text{LCM}(N_{\text{int}}, N_{\text{chip}})$ (LCM is the least common multiple) which is the smallest N that is needed to fully characterize the system. Both the chipping sequence and the

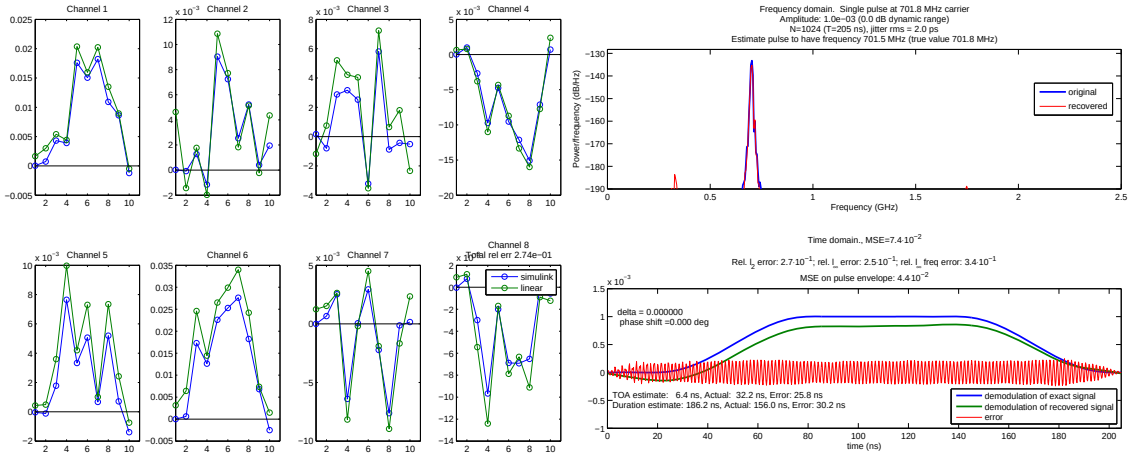


Figure 2.16: Phase blind calibration, using Simulink to generate "real" data, and using a simple Bernoulli model for Φ_{crude}. The Bernoulli model is inaccurate and does not capture phase response of the system, but it is sufficiently accurate to allow estimation of the phase of input signals since this is a massively overdetermined problem. From this phase estimate, a linear model Φ is created. The plot on the left shows how well Φx predicts measurements $S(x)$ from a test pulse; and the right plot shows reconstruction of this test pulse using. In both plots, the errors appear to be systematic in a harmless manner, affecting mainly the amplitude.

ADC samples are periodic, but the actual rows of the Φ matrix will not repeat until both periodic sequences line up. For $N_{\text{int}} = 100$ and $N_{\text{chip}} = 128$, this is $N_{\min} = 3200$. The system is fully characterized by a $\Phi_{\min}$ matrix with 3200 columns. To simulate a larger system, a new Φ matrix can be created by using copies of $\Phi_{\min}$ on the diagonals.

2.3.5 Phase blind calibration

The calibration procedure builds up the N rows of the frequency-domain matrix using

$$\Phi_{\text{FFT}}(:, k) = S(e^{i2\pi kt/T}) = S(\cos 2\pi kt/T) + iS(\sin 2\pi kt/T)$$

where the last equation follows from linearity of S and Euler's formula. The signals sin and cos are said to be "in quadrature" because they have 90° phase difference. This is convenient but not necessary: the measurement $S(e^{i2\pi kt/T})$ can be determined by measuring any two pure tones of frequency kt/T as long as they have distinct phases modulo 180°, since then the two input tones are linearly independent. The calibration is more robust if the phases are close to quadrature.

For the moment, we will consider the problem of just measuring a single-tone $\cos(2\pi kt/T + \theta)$. For calibration using hardware, an arbitrary waveform generator (AWG) can very accurately produce a sinusoid at the frequency kt/T; see Figure 2.13 for a diagram of the experimental setup. The issue with this procedure is that the phase θ of the incoming signal is unknown. The AWG cannot control phase, nor is it possible to accurately estimate the phase delay during travel from the AWG to the chip.

We propose the following method to estimate the phase. This requires a previous estimate of the

system, Φ_{crude}. This can be created from any of the methods described above, such as a Bernoulli model, or a Simulink calibrated model. The better this estimate is, the more accurate the phase estimation will be. Let $b = S(\cos 2\pi kt/T + \theta)$ be the measurements. Then we solve

$$\hat{\theta} = \measuredangle \operatorname*{argmin}_{A \in \mathbb{C}} \|b - \Phi_{\text{crude}} A e^{i2\pi kt/T}\|_2 \tag{2.3.6}$$

where t is a Nyquist time grid. This estimation is a 2-dimensional least squares problem which can be efficiently solved, similar to the method mentioned in §2.4.2.

It is likely that the crude estimate Φ_{crude} has different alignment than the true hardware, so the estimate $\hat{\theta}$ will be off. The biggest source of error is a shift in the bit sequence. Thus the above procedure can be repeated for a shifted version of Φ_{crude}. For every shift, we generate a new estimate of the phase; using this phase estimate, we compute the sample error, and pick the shift that has the least error; see Figure 2.17. If Φ_{crude} is a Bernoulli matrix, then shifting it is trivial. For a Simulink calibrated matrix of size $N_{\min}$, exactly $N_{\min}/N_{\text{chip}}$ of the possible N_{int} shifts can be easily generated by shifting the matrix. If N_{chip} and N_{int} are relatively prime, this is all of the shifts. In our system, $N_{\text{chip}} = 128$ and $N_{\text{int}} = 100$, so only 32 of the possible 128 shifts are generated by shifting the rows and columns of Φ_{crude}. To generate the other shifts, we need to build up the impulse response of 3 other Φ_{crude} matrices, each with a shift from a different co-set.

The estimation problem is parametric, with θ being determined by 2 parameters (or 1 complex parameter), and the shifts being determined by N_{int} parameters. If T is short, then b only has a few entries, so this problem is ill-posed. However, hardware measurements can easily generate long T samples. Furthermore, the shift of the chipping sequence is fixed regardless of the signal measured, so all the data can be combined to estimate the best shift of the chipping sequence. Hence the problem is estimating about 256 parameters from a dataset of size on the order of 10^4 to 10^6, and hence should be extremely accurate. The shift of the PRBS is estimated separately for each channel since each channel may see slightly different delays, but the shifts are expected to be similar, so this helps the estimation.

The crude model Φ_{crude} may not account for some complicated properties of the system, such as the phase response of the various filters, and this will lead to systematic errors in the phase estimation which are frequency dependent. However, preliminary tests (using Simulink as the "real" device, and a Bernoulli Φ_{crude}) suggest that the error is a smooth function of phase and also concentrated at low frequency. Thus it may be possible to correct for this; even if it is not corrected, the effect on reconstruction may be small; see Figure 2.16.

We now return to the issue of finding the response of the system to two signals of the same frequency but of different phase. Let θ_1 and θ_2 be the phases of the two input tones, each of grid frequency kt/T for some integer $|k| \leq N$. One possibility is to separately estimate θ_1 and θ_2 in

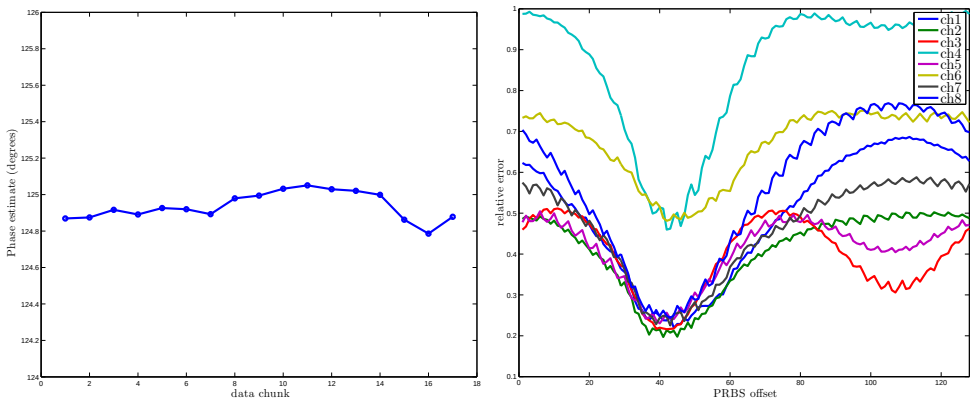


Figure 2.17: Phase blind calibration. Showing preliminary results from late March, 2011, using a single frequency input. This is real hardware data, using a Bernoulli Φ_{crude}. Left: the estimated phase for different data sets (single channel shown). Right: since there is no absolute time reference, the PRBS sequence needs to be aligned. The estimated shift of the sequence is the location of the minimum. Each channel shows a similar minimum, which indicates the method works. This is calculated from a single data set; by including more data sets, the data will improve.

the manner just described. A more accurate method is to record $2N_{\text{min}}$ measurements of a single input tone; let $T_{\text{min}} = N_{\text{min}}\Delta T$. Recall that $T = N\Delta T$. If the input is $x(t) = \cos(2\pi kt/T + \theta_1)$, then we consider $x_1 = x(t)$ as the first signal, and $x_2 = x(t + T_{\text{min}})$ as the second signal. This is equivalent to $x_2 = x(2\pi kt/T + \theta_2)$ where $\theta_2 = \theta_1 + 2\pi kT_{\text{min}}/T$. Thus x_2 is linearly dependent on x_1 whenever $kT_{\text{min}}/(2T) = 2kN_{\text{min}}/(N) \in \mathbb{Z}$. If $N > N_{\text{min}}$, then there are some frequencies k for which this is not an integer, and hence we simply use the second set of measurements to complete the characterization of this frequency (and θ_1 and θ_2 are now jointly estimated). If the quantity is not an integer but lies close to an integer, then we can improve robustness by skipping ahead by a few more blocks of N_{min}.

For frequencies k where this is an integer (if $N = N_{\text{min}}$, this is all frequencies) then $\theta_2 = \theta_1 \mod 180$ so the samples are linearly dependent. In this situation, we generate several sets of measurements from the AWG, and most likely each waveform will have a different phase. These phases are estimated separately and then combined. Note that if N is prime, then this situation will never occur, though it may not be robust. A large prime value of N, with $N \gg N_{\text{min}}$, will be a more robust choice, though it will require more measurements.

When $N > N_{\text{min}}$, then the Φ and Φ_{FFT} matrices have unnecessary columns. If necessary, the time-domain version Φ can just remove the extra columns *after* being created by an inverse DFT of the larger Φ_{FFT}. This will induce a few errors (since the time vectors should really be thought of as the inverse DFT of the frequency vectors, not as a inherently meaningful themselves), but should not be significant.

This calibration system is in the process of being implemented on the RMPI chip. A computer controls the entire process, but it is far from trivial since several thousand sets of precise measurements must be taken. Preliminary results are shown in Figure 2.17 which indicate the

method consistently estimates phase and shift parameters. These results show the the phase and shift estimates of a 1 GHz signal collected from hardware measurements, and using Φ_{crude} for the estimation.

We note that changing the ADC rate f_{ADC}, and hence T_{int}, is one of the easiest system parameters to modify. Changing T_{int} from 100 to 96 makes a big change, since now $N_{\min} = \text{LCM}(N_{\text{chip}}, N_{\text{int}})$ is 384 instead of 3200, so it would be possible to build a calibration model using a smaller N since the only requirement is that $N \geq 384$.

An alternative to the phase blind calibration is to use the accurate SPICE model for calibration, since then the input signal is known exactly. However, since the Cadence software is run on a single computer, it would take about a month of computational time to record $N = 1024$ signals, so we favor the hardware-based approach.

2.4 General design considerations

This section and the following two sections consider some important design choices. Below are some of the major questions we aim to answer.

- At what rate does the chipping sequence need to operate?
- What periodicity does the chipping sequence need?
- What characteristics should the PRBS have?
- Given a fixed information rate (400 MHz), how many channels are necessary?
- How closely must the integrating filter approximate an integrator?
- What bandwidth does the integrating filter need need?
- Which blocks of the system must be highly linear?
- Which blocks of the system are affected by clock jitter?

We are also concerned about whether the system is robust to non-idealities, such as thermal noise, over which we have little control. This section discusses a few of the design decisions and rational behind them. The chipping sequence and integrator are discussed in detail in sections §2.5 and §2.6, respectively. By necessity, components are introduced one-at-a-time, but of course it is impossible to simulate changing one part of the system without first having a model for the entire system, so occasionally we reference a system block out of order.

The design parameters are coupled to each other. For example, a PRBS with short repetition rate can be partially compensated for by using a nearly ideal integrator, so these two systems

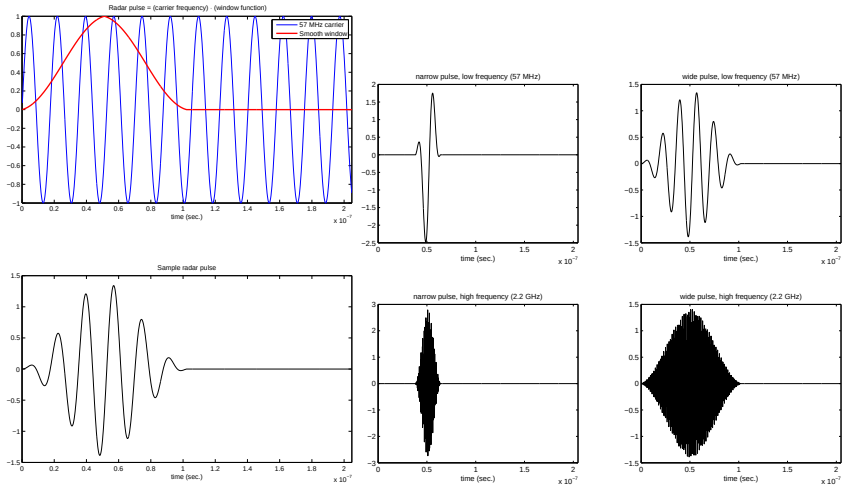


Figure 2.18: (Left) A radar pulse consists of a pulse envelope and a carrier frequency. (Right) For initial tests beginning in 2008, four types of pulses were used in "corner" sims. The four test pulses cover low frequency (57 MHz) to high frequency (2.2 GHz), and narrow (20 ns) to wide (100 ns), all with smooth pulse envelopes.

cannot be considered in isolation. Unfortunately, due to time limitations, it is not practical to optimize all the parameters at once, and instead we perform one round of "coordinate-descent": fix all parameters but one, and then find the best value for this parameter; then repeat for the other parameters. Furthermore, as our simulation model gains complexity over time, certain previous design decisions become fixed, so it is not simple to revisit old choices. Another complication is that recovery, discussed in §2.7, involves many parameters as well, so the overall parameter space is high dimensional. However, results indicate that the approach has worked remarkably well.

2.4.1 Test signals

The performance of the system was primarily evaluated on its accuracy for reconstructing single pulses, although multiple pulses were tested on occasion. To capture various types of behavior, it is important that the system tests a wide range of carrier frequencies and pulse widths. The simulations consisted of either "corner simulations" using extreme types of pulses (see Figure 2.18) or a combination of systematic parameters, such as pulse length, and random parameters, such as phase and frequency of the carrier.

The window of the pulse was occasionally changed, since this affects the spectral sparsity of reconstruction. Window functions varied from smooth windows, like a Gaussian pulse, to trapezoidal windows. Most of the recent simulations used filtered trapezoidal windows. For the compressive matched filter recovery described in §2.7.1, the pulse window has little effect, since spectral leakage is not an issue. In fact, a matched filter works best when the window is a pure box-car with infinitely fast initial rise time.

Simulations were careful to either use frequencies chosen uniformly at random in $[0, f_s/2]$, or

fixed frequencies that are off-grid. The system can perform much better with on-grid signals, but this is an unrealistic assumption.

Testing dynamic range was done in two manners. The first test was with a single signal, and varying the amplitude. This gives an absolute limit to the dynamic range. The other test was with two signals, and varying the relative amplitudes. This limit is always less than the single-pulse dynamic range since the large pulse introduces extra difficulties for the small pulse, such as creating extra error due to jitter, as well as creating non-linear effects.

2.4.2 Error metrics

For synthetic simulations, the true signal x_0 is known, so this can be used as a reference. The most common metric is the root-mean-squared error (RMSE), which we define as

$$\mathrm{RMSE}(x) \triangleq \frac{\|x - x_0\|_2}{\|x_0\|_2}. \tag{2.4.1}$$

This metric is perhaps more correctly referred to as *relative* RMSE. On occasion, we reference MSE, which is just the square of this quantity. One benefit of RMSE and MSE is that they are the same regardless of whether we consider time-domain samples or frequency components. Other reconstruction-based metrics we consider are $\|x - x_0\|_\infty$ and $\|\hat{x} - \hat{x}_0\|_\infty$, where $\hat{x}$ is the (non-unitary) DFT of x.

For pulses, the parameters of interest are the pulse descriptor words (PDW), such as time-of-arrival, so these can be the basis of error metrics, though we do not use them much. The envelope of the function itself is also of much interest, so the error can be cast in terms of this. To find the envelope of a pulse x, aka demodulation, we perform the following, which requires an estimate of the carrier frequency f (found via the Welch method or similar):

1. Using f, estimate the phase ϕ of the carrier. This is done by matched filter in the following manner:

   $$(a, b) \leftarrow \operatorname*{argmin}_{a,b} \|x - a\sin(2\pi t f) - b\cos(2\pi t f)\|$$

   where t is a time grid. This is a 2D least-squares problem that can be solved efficiently. The phase is estimated $\phi = \tan^{-1}(b/a)$. The operation is similar to phase estimation of the calibration scheme, although in this case the measurements are from full Nyquist-rate samples.

2. The signal is mixed with $\sin(2\pi t f + \phi)$ and then digitally low-pass filtered by truncating its DFT coefficients.

Another demodulation method is taking the absolute value of the Hilbert transform of a signal, since the Hilbert transform puts the phase information into the complex phase of the analytic signal.

Number of channels, r_{ch}	1	2	4	8	16	32
f_{ADC} (in MHz)	400	200	100	50	25	12.5
T_{ADC} (in ns)	2.5	5	10	20	40	80
N_{int}	12.5	25	50	100	200	400

Table 2.3: Various system values as a function of r_{ch}, using a fixed overall sampling rate of 400 MHz. The current RMPI uses 8 channels.

However, this approach works poorly when there are several signals present. Our demodulation method works well in multi-signal environments, since the filtering discards the background signals. The main disadvantage is that it requires an accurate estimate of the frequency, but it is not overly sensitive to this, so we find that it works well in practice.

For some of the tests on hardware, the original signal x_0 is unknown, or only approximately known, in which case "recovery" is more subjective, and usually based on visual inspection of time and frequency plots.

Some reconstruction methods, such as the matched filter, are more appropriately analyzed only in the frequency domain. Error is then recorded as a function of how far the estimated frequency is from the true frequency, $|f - f_0|$. Often this type of error is thresholded and the reconstruction is declared a "success" if $|f - f_0| < f_{\text{cutoff}}$, and a "failure" otherwise, with f_{cutoff} typically 1 MHz. These reconstructions can be aggregated into a sample recovery percentage statistic.

2.4.3 Number of channels

For a fixed ADC sampling rate, increasing the number of channels will always improve performance. However, each channel requires extra power; in the extreme case, we could have 100 channels each sampling at $f_{\text{ADC}} = 50$ MHz, but this is just a conventional channelized receiver. But on the other hand, a single channel receiver at $f_{\text{ADC}} = 50$ MHz would undersample too much.

In order to analyze the appropriate number of channels, consider a fixed information rate system that undersamples the Nyquist rate by 12.5×. This means samples are taken at an average rate of 400 MHz, since $f_s = 5$ GHz. The final Caltech RMPI design uses 8 channels at 50 MHz each, whereas the Northrop Grumman design uses 4 channels at 100 MHz each. This section discusses this trade off.

From a theoretical point-of-view, more channels are better. The matrix representation Φ of such a system closely approximates a signed Bernoulli matrix, which is known to be incoherent with any basis. In contrast, a single channel operating at a fast rate becomes closer to a simple underclocked regular ADC, which is not a good measurement system.

However, multi-channel systems cause quite a few problems. Consider the situation of recovering a large pulse and a small pulse. If these pulses are separated in time, then we don't expect the

large pulse to have any effect on the recoverability of the small pulse. If the pulses overlap in time, then because clock jitter induces error proportional to the amplitude of the largest signal, the small pulse will see a relatively large amount of additive noise which will make recovery difficult. If the RMPI has many channels, then each channel only takes measurements every T_{ADC}, where T_{ADC} can be quite large. This means that even if a big pulse and small pulse do not overlap in time but are within T_{ADC} of each other, then they will affect each other. As the number of channels increases, T_{ADC} increases, so this is more likely to occur.

The longer integration window means that more noise is added to each measurement. By itself, this does not affect the signal-to-noise ratio, since more signal is included in each measurement, but it also means that the noise included is more correlated (i.e., covariance matrix $\Phi\Phi^T$), so it is more important to account for this correlation in recovery algorithms. A longer T_{ADC} also increases the chance of clipping the signal by reaching the maximum amplitude; see §2.8.1.5.

Another effect is caused by the non-ideal integration. Consider a single-pole integrator, and assume the time constant (which is the inverse of the pole location) is constant, regardless of the number of channels. The time constant measures how long it takes an equivalent capacitor to discharge or charge e^{-1} of its max capacity. For the initial RMPI design, the time constant was 22 ns (this has been subsequently improved). In a 32 channel system, $T_{\mathrm{ADC}} = 80$ ns, which means that samples arriving early in the time interval will be attenuated by about $e^{-4} \approx 0.018$. The measurements are heavily weighted toward the end of the interval. To a crude approximation, we are only taking about 20 ns worth of information, so this may cause problems when sampling very short pulses. Decreasing T_{ADC} helps the problem.

For reconstruction, having many channels causes more difficult calibration and more difficult reconstructions. To see how reconstructions are affected, consider taking m measurements per channel, so the system matrix Φ has $M = mr_{\mathrm{ch}}$ rows. It is convenient to choose the number of columns, N, to be a multiple of m. However, this is not always possible for two reasons: the first is that analysis with the Gabor dictionary is much more efficient when N is a power of 2, and in fact the code only handles this case. The second reason is that we would like to consider the possibility of *staggering* each channel (to be discussed in §2.6), which means that the ADC samples of each successive channel are offset by a small amount Δm from the previous channel. Thus we want all of $N, N + \Delta m, \ldots, N + (r_{\mathrm{ch}} - 1)\Delta m$ to all be divisible by m, and the offset is most naturally $\Delta m = N_{\mathrm{int}}/r_{\mathrm{ch}}$. This is impossible to satisfy in general. Whenever there is mismatch, it leads to zeros in the Φ matrix, since we cannot make use of a partial sample. For a small number of channels, this is not an issue, but with a larger number of channels, it happens with increasing frequency. For example, with 8 channels, N_{int} is 100, and if $N = 1024$ and no stagger is used, this leaves 24 columns of zeros in the Φ matrix and each channel takes $\lfloor N/N_{\mathrm{int}} \rfloor = 10$ measurements. Now consider staggering each channel by an offset of about 12.5 Nyquist samples on average, so

that channel two is shifted by 25 Nyquist samples. The last sample of channel two runs from 926 to 1025, which exceeds the length of N, so this sample is discarded, and this channel only keeps 9/10 of its data. For a fixed length of N, this loss of data is exacerbated as the N_{int} increases, so it gets worse for a many-channel system.

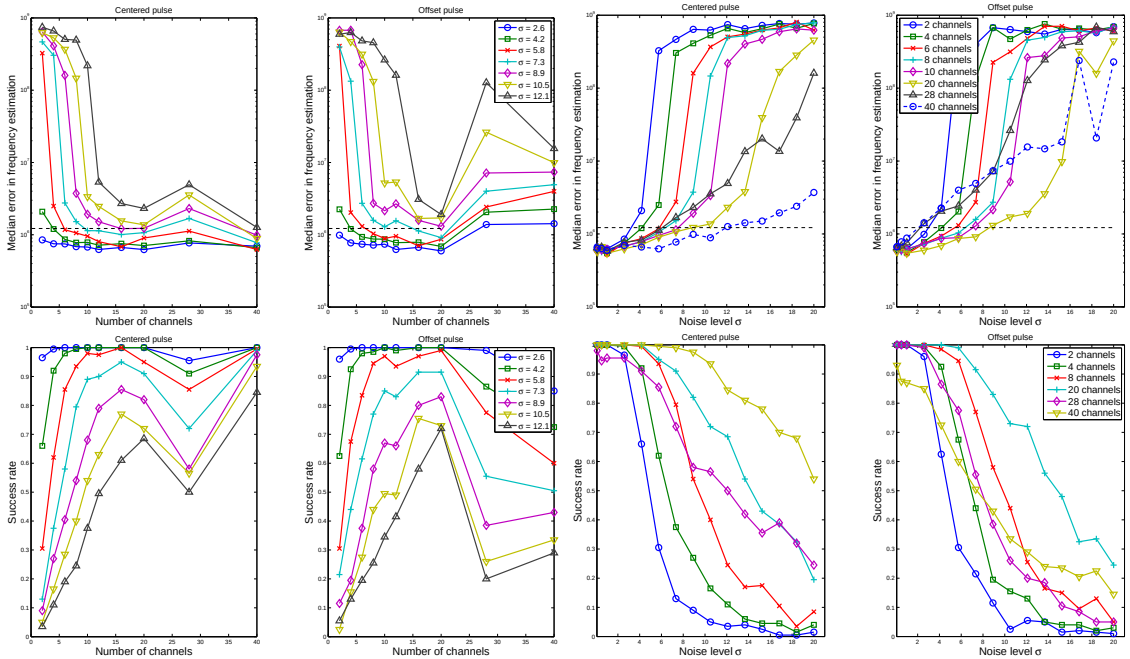


Figure 2.19: Matched filter simulations. Top row plots the error of frequency estimation, bottom row uses a 1 MHz frequency estimation cutoff to determine "successes" and "failures", and plots the success rate (using 200 independent trials). Plots either show results as a function of noise standard deviation σ, and each line represents a fixed r_{ch}; or plots as a function of r_{ch}, and each line represents a fixed σ. The title of the plot is "centered" or "offset", which refers to the location of the pulse. Offset pulses are near the boundary, and are more affected by missing measurements whenever N_{int} does not exactly divide N. In all plots, the dashed black line represents the resolution of an FFT of length N.

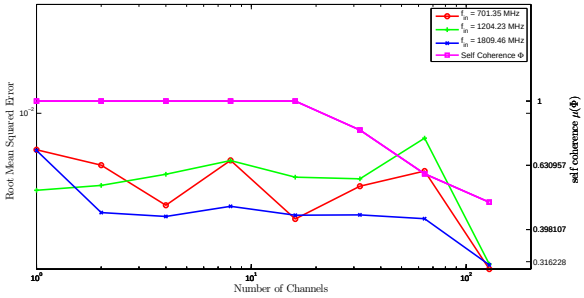


Figure 2.20: The RMSE error as a function of the number of channels (but fixed overall data rate at 400 MHz) for a signal with SNR = 0 dB, using Φ from the Bernoulli model

To study the effect of these considerations, simulations were run using Φ matrices representing systems of various channels. Since we do not have a working Simulink model of an arbitrary channel system, the Φ matrices were chosen with signed Bernoulli entries, but in an otherwise realistic fashion (the PRBS repeated, only whole measurements were kept, and stagger used for some tests).

Figure 2.19 shows the median results of 200 random simulations per parameter. In order to collect statistics, we used the ideal matched filter reconstruction because it is extremely fast and has no parameters. The signal was a pulse of length 100 ns and the phase and frequency were chosen uniformly at random, with $N = 1024$. The matched filter must calculate normalizing factors for every Φ matrix, which is relatively slow, so for every parameter, two Φ matrices were generated and then 100 random signals were used with every Φ matrix. This version of the matched filter searches only over frequency, phase, and amplitude, and used *a priori* knowledge of the location and size of the pulse. Noise was added to the measurements *after* the measurement operation: $b = \Phi x + z$.

The results are shown in Figure 2.19. We consider several levels of noise to get a full picture of how the system performs. We tested both centered pulses and pulses offset to the side of the interval, since our hypothesis is that zero columns in the Φ matrix, which always occur at the edges, will affect these offset pulses. Indeed, this is what we observe. Consider the four plots on the right. The 40 channel system is amazingly robust to noise when we measure a centered pulse, since for a success rate of 50%, it can withstand noise up to $\sigma = 20$, whereas the 8 channel receiver can only withstand up to $\sigma = 10$. Yet on the *offset* pulse, the 8 channel receiver outperforms the 40 channel receiver since the 40 channel Φ matrix has more incomplete measurements that must be discarded. Looking at the left four plots shows that, depending slightly on the noise level, the best design is 16 or 20 channels. The plots are not monotonic, which may seem surprising, but this is an artifact due to using any fixed N, since some large numbers divide into this with smaller remainder than do smaller numbers. For $N = 2048$ or $N = 4096$, the maximum would be in a slightly different location. As computing power increases and it is easier to perform reconstructions with extremely large N, it would become increasingly beneficial to use large channels system.

Figure 2.20 show results using a similar Bernoulli matrix and using full ℓ_1 reconstruction (with reweighting), so this is a more realistic scenario. Due to the increased computational complexity, only three signals (low, medium, and high frequency) are used, and only one trial is performed per channel number. Because of the full reconstruction, we are able to examine the root-mean-squared error (RMSE). Also plotted is the self-coherence $\mu(\Phi)$ (that is, the maximum absolute value of a non-diagonal entry of $\Phi^T\Phi$), which decreases as the number of channels increases. A small self-coherence is a sign of a useful compressed sensing matrix, so this corroborates the intuition that a large number of channels is good for theoretical purpose. From these limited trials, which did not test an offset pulse like in the earlier figure, we only see benefits of an increased number of channels when the channels approach 100, which is impractical.

From a system design point-of-view, more channels are more difficult to design, since it becomes important to isolate each channel to prevent cross-talk, and it also requires considerable power for the clock distribution system.

Because of this hardware complexity, 8 or 12 channels is a practical limit. The simulations

suggest that increasing channels up to a point, probably around 20, may significantly improve the performance of the system. However, we choose 8 channels for the final design because of prohibitive design complexity for 12 or more channels. The Northrop Grumman design uses 4 channels, and the simulations suggest it will be more sensitive to noise than the Caltech version, but it will benefit from simpler design.

2.5 Chip sequence

Perhaps the most important design choice is the chip sequence, which we refer to synonymously as the PRBS (pseudo-random bit sequence). The RMPI chip sequence is periodic, but for the sake of analysis, we will consider both periodic and infinite chip sequences. We may abuse notation and write $c(t)$ for the time domain chip sequence, but also use $\{c_n\} = \{c_0, c_1, \ldots, c_{N_{\text{chip}}-1}\}, c_n \in \{-1, 1\}$ to be the pseudo-random bit sequence (PRBS) that defines $c(t)$. Specifically,

$$c(t) = \sum_{n=-\infty}^{\infty} c_n g(t - n\Delta T) \tag{2.5.1}$$

where g is the rectangular window that is 1 inside $[-\Delta T/2, \Delta T/2]$ and 0 elsewhere. When $N_{\text{chip}} < \infty$, we implicitly use $c_n = c_m$ whenever $n \equiv m \mod N_{\text{chip}}$. There are two time periods associated with the chip sequence: ΔT is the Nyquist period, and is the duration of the bit. The RMPI operates at 5 GHz and so ΔT is 200 ps. Throughout this thesis, this rate if fixed. The other time period is $T_{\text{chip}} \triangleq \Delta T N_{\text{chip}}$, which is the periodicity of the chip sequence. In the final RMPI design, $N_{\text{chip}} = 128$, but in this chapter will will consider varying this. For a summary of notation, see Table 2.1.

In the Caltech RMPI, the chipping bits $\{c_k\}$ are loaded into a memory unit, and thus can be arbitrarily chosen. We will assume they are randomly and independently chosen, with equal probability of $+1$ and -1. The Northrop Grumman RMPI does not allow arbitrary chip sequences and instead uses a Gold code, which is a mixture of simple linear feedback shift registers (LFSR).

For the rectangular window g, an elementary calculation gives

$$\mathcal{F}g(f) \triangleq G(f) = \frac{\sin \pi f \Delta T}{\pi f} = \Delta T \operatorname{sinc}(f \Delta T) \tag{2.5.2}$$

where $\operatorname{sinc}(x) \triangleq \sin(\pi x)/(\pi x)$ at $x \neq 0$ and $\operatorname{sinc}(0) = 1$. It would be interesting to consider alternatives, such as smoother windows, so that the spectrum is more concentrated in the $[-f_s, f_s]$ band, but this may be difficult to implement in hardware and was not considered in our studies.

2.5.1 Spectral properties of the chip sequence

2.5.1.1 Infinite period

We wish to find the power-spectral density of a random chip sequence of infinite length. A chip sequence is not a stationary process, since if times t_1 and t_2 both lie inside the same ΔT interval, then the correlation is exactly 1, and if they live in different ΔT intervals, the correlation is 0, and hence there is some absolute time dependence. Signals with this discrete structure are a type of cyclostationary process.

The autocorrelation function of a random signal is defined

$$r_{tt}(t,\tau) = \mathbb{E}\, x(t)\bar{x}(t-\tau) \tag{2.5.3}$$

and if the signal is wide-sense stationary, then so is the autocorrelation and we have $r_{tt}(t,\tau) = r_{tt}(\tau)$. In this case, the Wiener-Khinchin theorem shows how to calculate the power spectral density (PSD) Ψ:

$$\Psi(f) = \mathcal{F} r_{tt}(\tau). \tag{2.5.4}$$

The theorem is useful because stationary signals are not necessary square-integrable, and so care must be taken when using the Fourier transform. When x is square-integrable, then the direct formula for the PSD is:

$$\Psi(f) = \lim_{T\to\infty} \frac{\mathbb{E}\,|\hat{x}(f)|^2}{T} \tag{2.5.5}$$

where $\hat{x}$ is understood to be the Fourier transform of x restricted to the domain $[-T/2, T/2]$.

Let g be a boxcar signal; that is, it is zero everywhere except it is 1 on the interval $[0, \Delta T]$. Then a chip sequence $c(t)$ with infinite period can be viewed as a superposition of shifted versions of $\pm g$. Neither the shifting nor the ± 1 affects $|G(f)|$, so intuitively the PSD of an infinite-period chip sequence is just $|G(f)|^2/T$ using (2.5.5). It turns out this is correct, and we can make it rigorous as follows, using a subtle trick (see, e.g., [BM90]) to make $c(t)$ stationary. This is also known as the time-average autocorrelation function [Pro01].

The autocorrelation of $c(t)$ is

$$r_{tt}(t,\tau) = \mathbb{E}\, c(t)c(t+\tau) = \sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} \mathbb{E}(c_j c_k) g(t-k\Delta T) g(t+\tau-j\Delta T) \tag{2.5.6}$$

$$= \sum_{k=-\infty}^{\infty} g(t-k\Delta T) g(t+\tau-k\Delta T) \tag{2.5.7}$$

due to the independence of c_k and c_j. This is not stationary because it depends on τ *and* t.

The insight is that a time shift of $c(t)$ will have the same PSD (since it only changes the phase of the Fourier transform), so we'll consider a random shift of the chip sequence by an amount

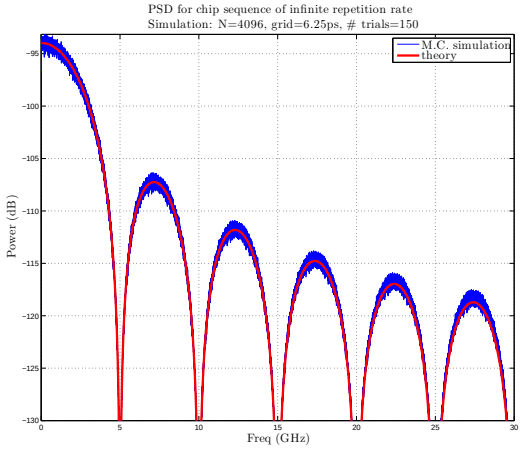


Figure 2.21: PSD Ψ of a chip sequence with infinite repetition rate; see (2.5.9). This plot was generated by averaging many sample realizations of very long chip sequences. The chip sequence is modulated at $f_s = 5$ GHz.

$\phi \sim U[0, \Delta T]$. The expectation is now taken over c_i and over ϕ.

Taking the expectation over ϕ is just the integral from $[0, \Delta T]$ divided by ΔT. Each term k in the sum is an integral from $[k\Delta T, (k+1)\Delta T]$, and thus the t dependence integrates out:

$$r_{tt}(t, \tau) = r_{tt}(\tau) = \frac{1}{\Delta T} \int_{-\infty}^{\infty} g(t) g(t+\tau) dt.$$

We are now in position to use the Wiener-Khinchin theorem (2.5.4), and take the Fourier transform with respect to τ (we can change the order of integration using Fubini's theorem since the integrands are well-behaved). Using $\mathcal{F} g(t+\tau)(f) = e^{i2\pi ft} G(f)$, we have

$$\begin{aligned} \Psi(f) &= \frac{1}{\Delta T} \int_{-\infty}^{\infty} g(t) e^{i2\pi ft} G(f) dt \\ &= \frac{G(f)}{\Delta T} \int_{-\infty}^{\infty} g(t) e^{i2\pi ft} dt \\ &= \frac{G(f)}{\Delta T} \bar{G}(f) = |G(f)|^2 / \Delta T \end{aligned} \tag{2.5.8}$$

which agrees with the intuition. In our system, g is a rectangular window, so using (2.5.2),

$$\Psi(f) = \frac{1}{f_s} \left(\frac{\sin \pi f / f_s}{\pi f / f_s} \right)^2. \tag{2.5.9}$$

See Figure 2.21 for a plot. Most of the power falls inside the first lobe from $[-f_s, +f_s]$.

2.5.1.2 Finite period

The actual RMPI design repeats the PRBS chip sequence every N_{chip}. Since $c(t)$ is then periodic, it has a Fourier series representation, which we now derive.

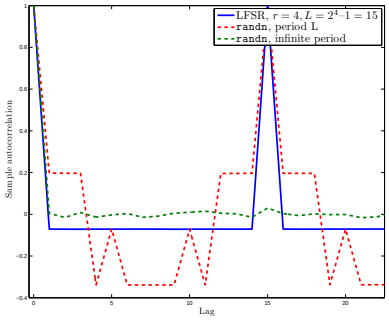


Figure 2.22: Sample autocorrelations of some PRBS sequences. The LFSR is of maximal length $L = 2^r - 1$. For lags greater than 1 and less than L, the autocorrelation of the LFSR is $-\frac{1}{L}$, whereas it is 0 in this region for a random sequence. The `randn` sequences are the signs of a pseudo-random sequence generated with Matlab's default `randn` routine which approximates a Bernoulli random variable very well.

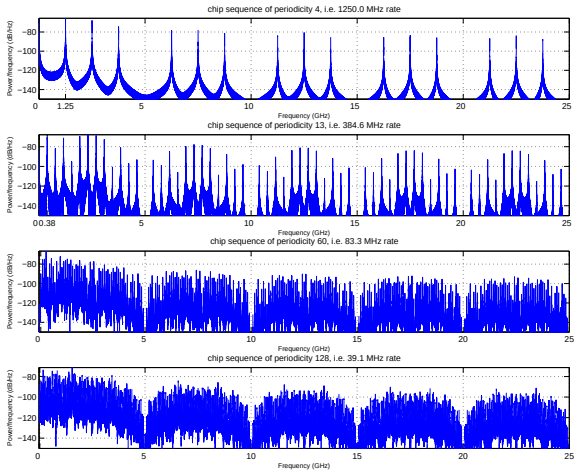


Figure 2.23: Spectrum of finite length PRBS. The longer the periodicity in the PRBS, the finer the grid spacing, which in general is advantageous. Each row is generated by a different N_{chip}. Contrast this to the smooth spectrum of a chip sequence of infinite length in Figure 2.21.

Let N_{chip} be the repetition length of the chipping sequence ($T_{\text{chip}} = N_{\text{chip}}\Delta T$; see Table 2.1 on page 45). Then the Fourier series is

$$\begin{aligned}
\hat{c}[k] &\triangleq \frac{1}{T_{\text{chip}}} \int_0^{T_{\text{chip}}} c(t) e^{-i2\pi kt/T} dt \\
&= \frac{1}{T} \sum_{n=0}^{N_{\text{chip}}-1} \int_{n\Delta T}^{(n+1)\Delta T} c_n e^{-i2\pi kt/T} dt \\
&= \frac{1}{T} \sum_{n=0}^{N_{\text{chip}}-1} c_n \frac{T}{-i2\pi k} \left(e^{-i2\pi k(n+1)/N_{\text{chip}}} - e^{-i2\pi k(n)/N_{\text{chip}}} \right) \\
&= \frac{\sin(\pi k/N_{\text{chip}})}{\pi k} \sum_{n=0}^{N_{\text{chip}}-1} c_n e^{-i2\pi k(n+\frac{1}{2})/N_{\text{chip}}}.
\end{aligned}$$

With the convention that $\sin(0)/0 = 1$, this holds for any integer k. Note that this would be periodic

in N_{chip} if it were not for the sinc term. For nonzero k,

$$\hat{c}[k+\tau N_{\text{chip}}] = \frac{\sin(\pi k/N_{\text{chip}}+\pi\tau)}{\pi(k+\tau N)} \sum_{n=0}^{N_{\text{chip}}-1} c_n e^{-i2\pi k(n+\frac{1}{2})/N_{\text{chip}}}(-1)^{\tau} = \frac{k}{k+\tau N_{\text{chip}}}\hat{c}[k] \quad (2.5.10)$$

and for zero k, $\hat{c}[\tau N_{\text{chip}}] = 0$ if $\tau \neq 0$.

To get the Fourier transform from the Fourier series, we haven

$$\hat{c}(f) = \sum_{k=-\infty}^{\infty} \hat{c}[k]\delta(f - \frac{k}{T_{\text{chip}}}) \quad (2.5.11)$$

which has discrete spacing T_{chip}. It is almost periodic in $N_{\text{chip}}/T_{\text{chip}} = f_s$, except for the decay of $k/(k+\tau N_{\text{chip}})$ from equation (2.5.10).

To find the power-spectral density of a finite-length sequence, we revisit (2.5.6). The double sum no longer collapses to $j = k$ but rather $j = k + j'N_{\text{chip}}$ for any integer j'. Otherwise, the same reasoning holds, and the equivalent of (2.5.8) is

$$\begin{aligned}\Psi(f) &= \frac{|G(f)|^2}{\Delta T} \sum_{j'=\infty}^{\infty} e^{-i2\pi f j' T_{\text{chip}}} \\ &= \frac{|G(f)|^2}{\Delta T}\frac{1}{T_{\text{chip}}} \sum_{k=-\infty}^{\infty} \delta(f - \frac{k}{T_{\text{chip}}}) = \frac{1}{N_{\text{chip}}}\left(\frac{\sin \pi f/f_s}{\pi f/f_s}\right)^2 \sum_{k=-\infty}^{\infty} \delta(f - \frac{k}{T_{\text{chip}}})\end{aligned} \quad (2.5.12)$$

where we used the Poisson summation formula [Mal08]:

$$\sum_{k=-\infty}^{\infty} e^{-i2\pi kt/T} = T \sum_{k'=-\infty}^{\infty} \delta(t - k'T).$$

If the PRBS is generated via a linear-feedback shift register (LFSR), the statistics change slightly, due to the extra correlation. In particular, outputs c_n of a maximal LFSR with r taps, where $L = N_{\text{chip}} = 2^r - 1$, have exactly $(L+1)/2$ values of $+1$ and $(L-1)/2$ values of -1. Furthermore, half the runs of consecutive $+1$ or -1 are of length one, one-fourth are of length two, one-eighth are of length three, etc. [PSM82]. The effect of this is seen in Figure 2.22; in contrast to a random sequence, the autocorrelation is not 0 when $\tau > \Delta T$. The properties of LFSR are relevant since the Northrop Grumman design uses Gold codes, which are based on LFSR output. Properties of LFSR sequences have been much studied since they are used in spread spectrum communication and can also be converted into 2D masks for use in coded aperture interferometry. For reference we give the one-sided PSD of a LFSR sequence with length N_{chip} [PSM82]:

$$\Psi(f) = \frac{N_{\text{chip}}+1}{{N_{\text{chip}}}^2}\Big(\frac{\sin \pi f/f_s}{\pi f/f_s}\Big)^2 \sum_{k=-\infty}^{\infty} \delta(f - \frac{k}{T_{\text{chip}}}), \quad f > 0 \quad (2.5.13)$$

and $\Psi(0) = \frac{1}{{N_{\text{chip}}}^2}\delta(0)$. This is quite similar to (2.5.12).

2.5.2 Chip design considerations

For this section, we will suppose that the PRBS coefficients are chosen randomly from ± 1, and that we use a square window g. A recent paper [HBC11] suggests using run-length-limited codes as an alternative to random coefficients, but this is beyond the scope of our work. Given our decisions, there are two major questions left unresolved:

1. At what *rate* should the chipping sequence be modulated? Until now, we have implicitly assumed that the width of the window g was $\Delta T \triangleq 1/f_s$, since this is what the final design uses. In this section, we will use the variable f_{chip} to represent the rate of the chipping sequence, so that g has width $1/f_{\text{chip}}$.

2. How often can the PRBS *repeat*? That is, how small can the periodicity N_{chip} be?

Recalling the guiding principle of the RMPI design, we want all signals in the model to generate measurements of approximately the same energy. For a probabilistic analysis, we might look for a uniform bound over all input tones using the chip Fourier series, similar to the analysis in [ERW11] for the NUS. A simpler approach is to consider just the chip PSD, which gives an idea of "average case" behavior.

Regarding the finite-period chip sequence PSD (2.5.12), there are two salient issues. The first is the sinc window, and this will be the subject of §2.5.2.1 which discusses f_{chip}. The second feature is the discrete spacing, which is caused by $N_{\text{chip}} < \infty$, and this is the subject of §2.5.2.2.

Both sections will consider the power output when faced with an input signal of the form $x(t) = Ae^{2\pi i f_{\text{in}} t}$. If $H_{\text{sys}}(f)$ is the transfer function of the system (discussed in §2.6), then the output power is

$$P_{\text{out}} = \int_{-\infty}^{\infty} |A|^2 |H_{\text{sys}}(f)|^2 \Psi(f - f_{\text{in}}) df.$$

If η_0 is the noise output power (also a function of H), then a reasonable prerequisite for the signal post-processing to recover x is that

$$P_{\text{out}} \geq \eta_0 \text{SNR}_{\text{min}}$$

for some value of SNR_{min}.

Before discussing f_{chip} and N_{chip} in detail, we note a few complications which arise. First, the power output depends on both Ψ and H_{sys}, so the chip sequence and the integrator cannot be considered independently. In future work we hope to make a combined analysis, but for now we treat them independently. The second issue is that the length of N_{chip} is not the only consideration, but that the relation of N_{chip} to N_{int} is significant. For example, if N_{chip} and N_{int} are relatively prime, the system matrix Φ needs $N_{\text{min}} = N_{\text{chip}} N_{\text{int}}$ columns to characterize it, which means the

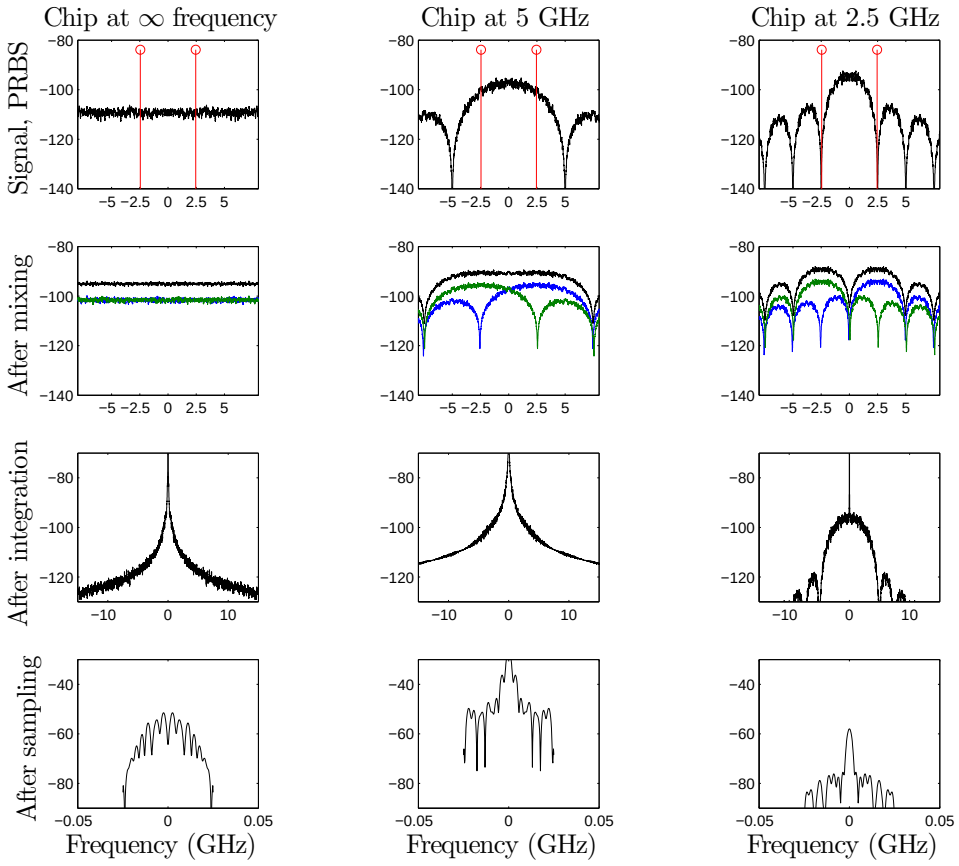


Figure 2.24: Effect of chipping rate. See also the companion Figure 2.25. Note that chipping at an infinite frequency is not actually desirable.

calibration procedure is much longer. In general, Φ has $N_{\text{min}} = \text{LCM}(N_{\text{chip}}, N_{\text{int}})$ columns. If $N_{\text{int}} = 100$, there is a very significant difference between $N_{\text{chip}} = 1000$ and $N_{\text{chip}} = 1001$. For the calibration reason, it is convenient to keep $\text{LCM}(N_{\text{chip}}, N_{\text{int}}) \lesssim 4000$. But as we will see below, it can also be quite bad for reconstruction behavior if N_{chip} is a low multiple of N_{int} or vice-versa.

2.5.2.1 Chip sequence rate

The PSD Ψ of the chip sequence is proportional to $\frac{\sin(\pi f/f_{\text{chip}})}{\pi f/f_{\text{chip}}}$, which is zero when $f = f_{\text{chip}}$. If an input has frequency f_{in}, the signal, after mixing with the PRBS, is just a convolution of the sinc with f_{in} and $-f_{\text{in}}$. Because the integration step will act as a low-pass filter, this means the output power is determined by the integral of Ψ in the regions around $-f_{\text{in}}$ and f_{in}. Thus, if $f_{\text{in}} = f_{\text{chip}}$, the output power will be very small.

Figure 2.24 shows this effect. Each column shows a different PRBS: the left column uses a PRBS that operates at such an extremely high frequency $f_{\text{chip}} \gg f_s$ that it has a flat spectrum, the middle column shows $f_{\text{chip}} = f_s$, and the right column shows $f_{\text{chip}} = f_s/2$. The red line in the top row represents an input tone at frequency $f_{\text{in}} \sim f_s/2$. The second row shows the result after

mixing with the PRBS (the blue and green lines show the convolution of the PRBS with f_{in} and $-f_{\text{in}}$, respectively). The third and fourth lines show the system after integration, which attenuates the high frequencies, and sampling, which folds everything into $[-f_{\text{ADC}}/2, f_{\text{ADC}}/2]$ (using $f_{\text{ADC}} = 50$ MHz). By examination of the bottom row (which has the same y-axis scale for all columns), it's clear that the power in the $f_{\text{chip}} = f_s$ variant is much greater than in the other cases.

Figure 2.25 shows the output power as a function of the input tone frequency f_{in}, averaging over many random trials. For $f_{\text{chip}} = f_s$, low frequency tones have only slightly higher energy than high frequency tones. For $f_{\text{chip}} = f_s/2$, the output power vanishes as f_{in} approaches $f_s/2$ since f_{in} falls into a zero of the sinc function. With the infinite frequency PRBS, the power is constant as a function of f_{in}. However, the absolute value of power is lower than in the $f_{\text{chip}} = f_s$ case, since the energy of the PRBS was spread out among all frequencies, whereas for $f_{\text{chip}} = f_s$ the energy is concentrated in the first lobe of the sinc function. Thus $f_{\text{chip}} \gg f_s$ is actually not desirable. The ideal PRBS would have a PSD that is flat inside $[-f_s, +f_s]$ and 0 elsewhere.

If we restrict ourselves to using a pseudo-random ± 1 sequence, then $f_{\text{chip}} = f_s$ is close to the optimal rate. If $f_{\text{chip}} \ll f_s$, then a low-frequency signal will generate large measurements and a high-frequency signal will generate tiny measurements, thus failing the principle of democracy. If $f_{\text{chip}} \gg f_s$, then too much PRBS power is out-of-band and wasted, and measurements of all signals will be too small and thus affected greatly by noise.

2.5.2.2 Chip sequence period

For hardware calibration and practical design reasons, the chip sequence repeats every N_{chip} terms, which is T_{chip} seconds where $T_{\text{chip}} = N_{\text{chip}}/f_{\text{chip}}$ (and from henceforth, we will fix $f_{\text{chip}} = f_s$). The PSD of the chip sequence has a discrete spectrum that is only nonzero at multiples of $1/T_{\text{chip}}$. The considerations here are exactly the same as in the previous section. If an input signal has carrier frequency f_{in}, then because the integration acts as a low-pass filter, the dominant contribution to the

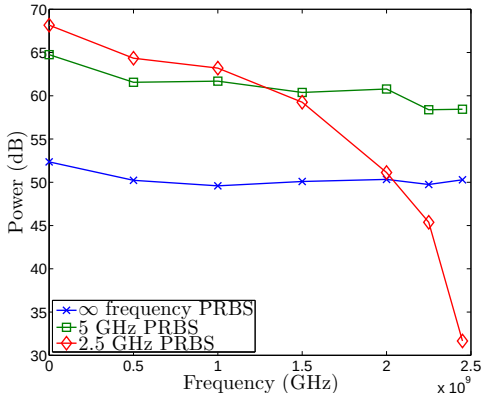


Figure 2.25: Using the same setup as Figure 2.24, and plotting power as a function of f_{in}

output comes from integrating Ψ in the regions near $-f_{\text{in}}$ and $+f_{\text{in}}$. If f_{in} is a multiple of $1/T_{\text{chip}}$, then $\Psi(f_{\text{in}})$ is nonzero. If f_{in} is not a multiple, then the attenuation effect of the the integrator H will be more significant.

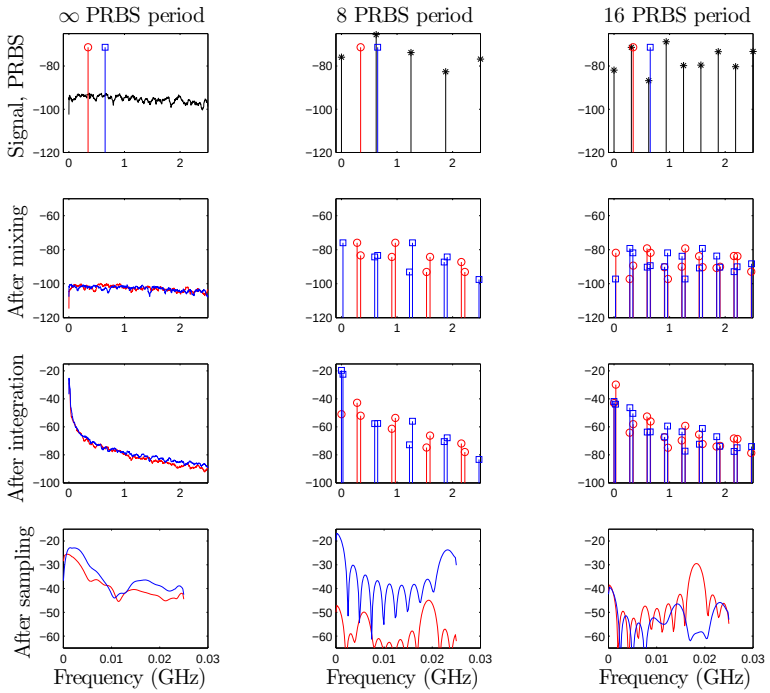


Figure 2.26: Depiction of problem associated with short chip periods N_{chip}. Using $N = 2048$. Shown are two possible inputs, one in red and one in blue. The left column shows an infinite-period chip sequence, while the middle and right columns show finite length chip sequences. The chip sequence with length $N_{\text{chip}} = 8$ works very well for signals (like the blue signal) with carrier frequencies that are lucky enough to be near one of its harmonics (harmonics spaced every $1/T_{\text{chip}} = 625$ MHz). For signals that are not near the carrier frequency, like the red signal, the power of the output is much less. For the left column, the energy of the ADC samples in the blue tone were .69 that of the red tone. For the rightmost column, the blue samples were 2.8 times larger in energy than the red samples. The center column shows the real disaster: the energy of the samples of the blue tone are 91 times larger that of the red tone. See Figure 2.29 for validation that this hurts reconstruction.

Figure 2.26 demonstrates this problem. Shown are three versions of the PRBS, with $N_{\text{chip}} = \{\infty, 8, 16\}$. In practice, we will consider $N_{\text{chip}} \simeq 100$, but we use small values of N_{chip} in the plot since they demonstrate the effect more clearly. The red and blue lines represent two input frequencies f_{in} and f'_{in}, chosen so that both of them lie on the $1/T_{\text{chip}}$ grid when $N_{\text{chip}} = 16$, but so that only f'_{in} lies on the $1/T_{\text{chip}}$ grid when $N_{\text{chip}} = 8$. After mixing, the power of both signals is the same. However, the integration severely reduces the power of the red signal, since its components are farther from DC.

This problem cannot be eliminated by changing the frequency response of the integrator, because we argued previously that we need the integrator to show decay: if it has a flat response, then it is really just sampling from a single point in time (i.e., $h(t) = \delta(t)$). However, the exact shape of H determines the how bad the problem is. The design of the integrator is considered in §2.6 independently of N_{chip}, but we hope to simultaneously analyze the two parameters in future work.

Recall that when choosing f_{chip}, we argued that $f_{\text{chip}} = \infty$ is not actually desirable (nor is it implementable). However, $N_{\text{chip}} = \infty$ *is* desirable from a theoretical perspective, as Figure 2.26 suggests. Unfortunately this would make calibration impossible, so we do not consider this option. The Caltech design uses a shift register which consumes power proportional to N_{chip}. The Northrop Grumman design is based on LFSR which generates $N_{\text{chip}} = 2^r - 1$ where r is the number of taps, and consumes power approximately proportional to r. Thus the LFSR variant might be better for future work since it allows large N_{chip}, although this would require more work for calibration.

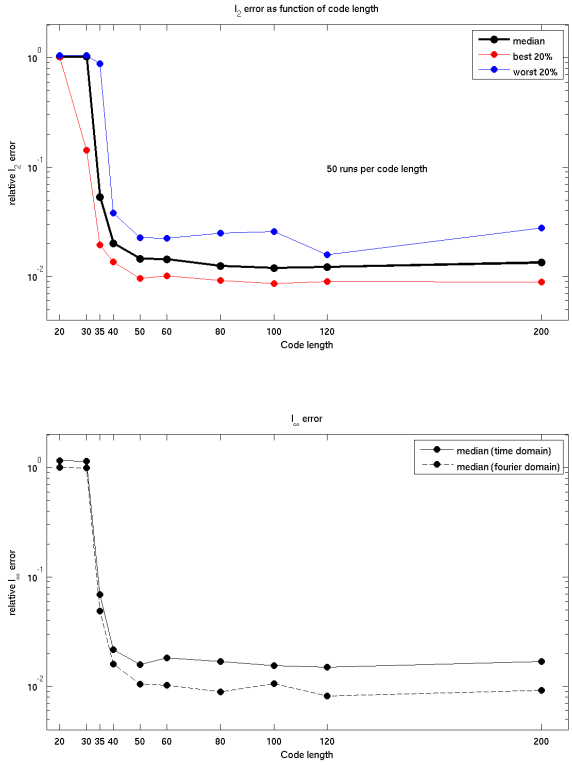


Figure 2.27: Initial codelength N_{chip} tests from 2008. We chose 128 to be safe.

Other than just the length of N_{chip}, the relation of N_{chip} to the integration period N_{int} is important. This does not affect the power, but rather the identifiability. If N_{chip} is a multiple of N_{int}, then after sampling, the result will be a discrete spectrum, and it may be possible for two distinct signals with carrier frequencies f_{in} and $f'_{\text{in}} = f_{\text{in}} + f_{\text{ADC}}$ to generate the same measurements. In general, if N_{chip} and N_{int} have a large GCD, then the outputs of the RMPI are more similar to each other, so the system is less robust.

To verify all of the above reasoning, we present results from several numerical tests. Figure 2.27 shows initial tests from August 2008 that were used to inform the initial version 1 design. This test and all other tests were using $N_{\text{int}} = 100$ (i.e., $f_{\text{ADC}} = 50$ MHz) unless otherwise specified. The

results from this first test suggested that performance was catastrophically bad if $N_{\text{chip}} < 40$, and inspired the choice $N_{\text{chip}} = 128$ in order to be a safe margin away from this failure point.

Figure 2.28 shows the spectrum of a single multi-tone input, and how it aligns with the PRBS grid. Using a realistic model of the system and full ℓ_1 reconstruction, we found that the tones near $1/T_{\text{chip}}$ gridpoints were recoverable, while those far from the gridpoints were not recoverable.

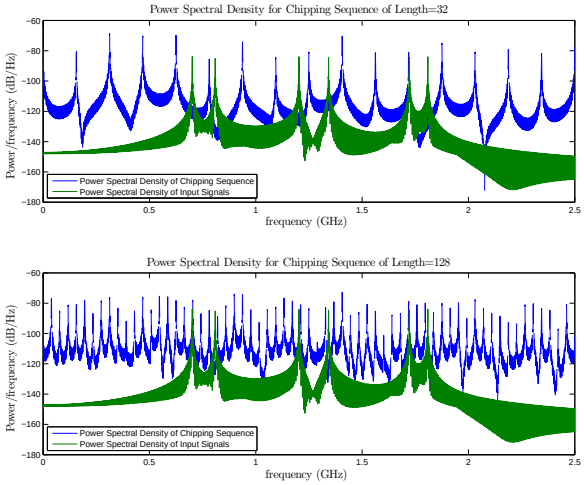


Figure 2.28: The PSD of chipping sequences (blue) and pure tone inputs (green). For a fixed integrator bandwidth B, with PRBS of length $N_{\text{chip}} = 32$ (top plot), the tones at 700, 1200, and 1800 MHz did not reconstruct successfully because they have large frequency separation from the nearest PRBS frequency $1/T_{\text{chip}} = 156.25$ MHz. Tones at 800 and 1750 MHz, close to a PRBS frequency, were recoverable. In the bottom plot with a 128 bit sequence, the maximum frequency separation was far less ($1/T_{\text{chip}} = 39.06$ MHz), and reconstruction was successful for all tones.

Next, we present evidence from some matched filter reconstructions, using a similar testing procedure as that described in the chip rate section. Because of the speed of reconstruction, we are able to generate meaningful statistics, and also have the benefit that the matched filter reconstruction relies on no parameters. The data were collected using an ideal Bernoulli matrix Φ, so integration was approximated by a discrete sum; for the sake of PRBS length, this approximation is most likely not significant.

Figure 2.29 shows the error of the frequency estimate as a function of $(f_{\text{in}} - 1/T_{\text{chip}}) \mod 1/T_{\text{chip}}$. A value of 0 on the x-axis indicates that the input frequency f_{in} was exactly on a grid point. The different plots show various values of N_{chip}. Each point represents the result of one independent test. Noise was included in the measurements; these data are a subset of the same data plotted differently in Figure 2.30. In Figure 2.29, we see a clear dependence between low error and $f_{\text{in}} \simeq 1/T_{\text{chip}} \mod 1/T_{\text{chip}}$, which confirms the reasoning above.

Figure 2.30 plots frequency error (and empirical frequency recovery "success" rates) as a function of of N_{chip} for various amounts of additive noise. As expected, recovery is more likely for large N_{chip}, but the effect is small after about $N_{\text{chip}} = 100$, so the choice of $N_{\text{chip}} = 128$ is about optimal. We also include results of the same simulations but using a matrix with staggered rows, and this improves recovery, especially for small N_{chip}. The simulations show periods N_{chip} of various lengths, including

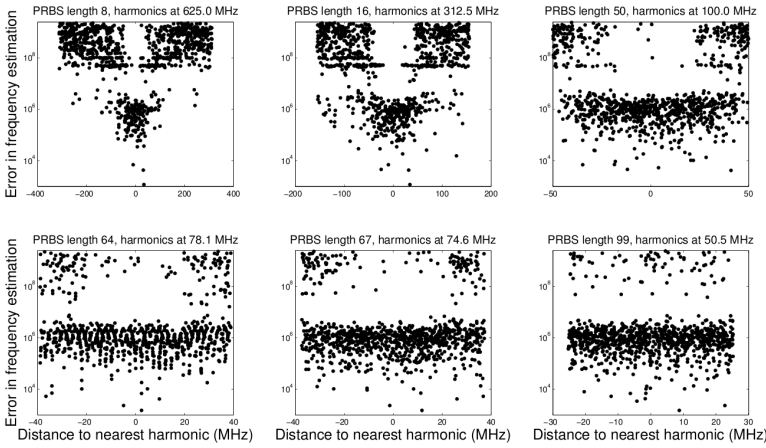


Figure 2.29: Explains why we want long N_{chip}. Error increases the farther we are from a harmonic. See Figure 2.26 for a cartoon depiction. See also Figure 2.30.

some chosen to be very close to each other but with different properties related to $N_{\text{int}} = 100$; for example, we included $N_{\text{chip}} = \{99, 100, 101\}$. The plots show very clearly that the behavior is different for even these small changes, but this behavior is not consistent over all noise levels and we find the results inconclusive and in need of further study.

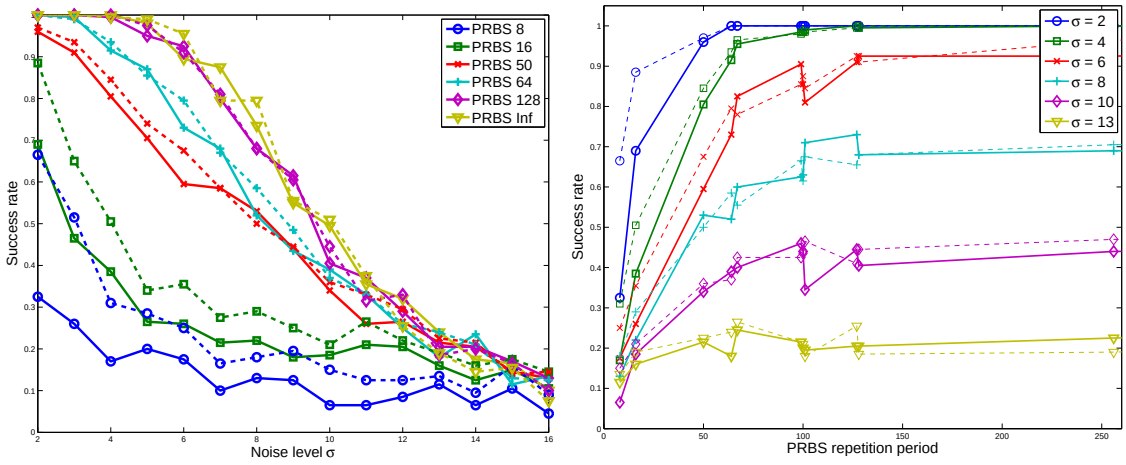


Figure 2.30: Tested PRBS sequences of period $N_{\text{chip}} = \{8, 16, 50, 64, 67, 99, 100, 101, 127, 128, 256, 512, \infty\}$ (∞ refers to length N, which was $N = 1024$ for this test). Other simulations parameters were the same as in Figure 2.19. Dashed lines for a matrix with staggered channels; this typically improves results, but not by too much. The seemingly bizarre choices of PRBS lengths were chosen because it is not only relevant what their length is, but also the least common multiple of the PRBS length N_{chip} and the ADC rate N_{int}. The ADC samples every $N_{\text{chip}} = 100$ Nyquist samples.

2.5.2.3 Case study: test of NG chip sequence

In the Caltech RMPI, the PRBS for each channel is loaded from a 128-bit shift register, so this can be programmed to an arbitrary sequence. However, this adds significant power cost to the design. Since this is the first working RMPI chip, it was considered worth the sacrifice in order to allow the extra flexibility: for example, it is possible to implement the run-length-limited ideas.

The Northrop Grumman chip saves power by avoiding shift registers, and instead generates the PRBS sequence from a LFSR. Initially, the direct outputs of a LFSRs were used, but after this author's analysis in January 2009, it was identified that this caused significant performance degradation, partially due to the occasional very long runs of $+1$ or -1. For a system with many channels, this is not an issue, but the NG chip only has 4 channels. In addition, the sequences sent to each channel were correlated, which added to the problems. Offsetting (aka staggering) the channels helps the situation, but not enough to match the performance of the Caltech RMPI.

Following the analysis, the PRBS was modified so that it was generated by a Gold code, which can be created by mixing the outputs of LFSR, so the additional hardware complication was minimal. The output of each Gold code repeated every 63 periods, but only the first 52 periods were used. Thus each of the four channels used a PRBS with length $N_{\text{chip}} = 52$. Because the NG design uses four channels, the ADC rate is greater than in the Caltech design, and operates just below 100 MHz, so that $N_{\text{int}} = 52$. Thus $N_{\text{chip}} = N_{\text{int}}$.

This design underperformed, so we analyzed the system. An empirical test was conducted as follows, which consisted of 100 independent trials. For each trial, a carrier frequency was picked uniformly at random from between 300 MHz and 2.5 GHz, which was modulated by a simple trapezoidal pulse that spanned almost all the signal; the Nyquist length was $N = 1024$ so this was about 200 ns. To be realistic, synthetic noise was added both to the signal and to the measurements, calculated to give 60 dB SNR separately.

Reconstruction was done via two methods. The first method used the over-complete Gabor dictionary with about 8 reweightings using the NESTA algorithm. The second method was more complicated, and consisted of first reconstructing the signal with a DCT orthobasis and reweighting a few times, followed by using the over-complete Gabor dictionary with reweightings. The second method almost always worked better. These techniques will be discussed in §2.7.

Reconstructions were declared either a success or a failure. To fail, two criteria had to be met (in practice, criterion 1 was almost never met without criterion 2 also being met). Both criteria were based on the frequency only, using Welch's method to estimate the PSD. The criteria were

1. Failure Criterion 1: the peak of the spectrum was not close to the actual carrier frequency. The allowable tolerance was set to the the width of the main lobe at -20 dB from the peak.

2. Failure Criterion 2: the side-lobes were too large. Side-lobes were required to be 20 dB lower

Design	Failure rates per 100 tests Method 1	Method 2
$N_{\text{chip}} = 52$, stagger 1	46	33
$N_{\text{chip}} = 52$, stagger 2	44	31
$N_{\text{chip}} = 52$, stagger 10	48	35
$N_{\text{chip}} = 52$, seed 1	57	43
$N_{\text{chip}} = 52$, seed 2	54	39
$N_{\text{chip}} = 52$, seed 3	50	35
$N_{\text{chip}} = 51$, seed 1	25	13
$N_{\text{chip}} = 104$, seed 1	13	7
$N_{\text{chip}} = 104$, seed 2	20	7
$N_{\text{chip}} = 104$, seed 3	20	9
$N_{\text{chip}} = 3276$	0	0
$N_{\text{chip}} = 312$, random signed Bernoulli	0	0
Entire channel a Gaussian matrix	0	0
Caltech design, $N_{\text{chip}} = 128$	0	0

Figure 2.31: June 2009 tests of the Northrop Grumman chipping sequence. The first half of the table show various NG designs; $N_{\text{int}} = 52$. When $N_{\text{chip}} = N_{\text{int}}$ or N_{chip} is short, the performance suffers. The NG design was modified to use $N_{\text{chip}} = 3276$ which results in much better performance. The second half of the table show some other models for comparison; the Caltech design seems to work well.

than the main peak.

By inspection, most signals that passed these frequency criteria also looked good in the time domain. The results are presented in Table 2.31. The LFSR depends on an initial state, referred to as a "seed," and the test was run using various seeds in order to determine if the poor performance was due to an unlucky choice of the seed.

This alignment of $N_{\text{chip}} = N_{\text{int}}$ proved to be disastrous, and shifting the channels relative to one-another did nothing to improve the results, nor did changing the seeds of the Gold codes. These results are reported in the first 6 rows of the table.

Initially attempts tried two fixes that yielded only modest improvement. The first fix is to take 51 samples of the Gold code to use for the periodic PRBS, with the idea that the mis-alignment $N_{\text{chip}} \neq N_{\text{int}}$ will improve recovery. This appears to have helped (see Table 2.31). A similar modification is to take 104 samples from the Gold code (which means the last 41 sample are duplicates of the first 41), and this also helps. However, neither modification matches the performance of the Caltech design or other baseline tests shown at the bottom of the table.

In light of these results that show poor performance compared to the Caltech Φ matrix, the NG team found a way to devise a PRBS of length $N_{\text{chip}} = 52 \times 63 = 3276$ by adjusting the LFSR inputs to the Gold code. According to the above criteria, this adjustment is extremely beneficial and results in no errors per 100 tests.

Note that in order to fully characterize the system, Φ must have 3276 rows. For the Caltech system, $N_{\text{chip}} = 128$ and $N_{\text{int}} = 100$, so their LCM is 3200; thus Φ has 3200 rows, so calibration of the Caltech and NG systems are of about the same difficulty.

2.6 Integration

Integration can be thought of in the time domain, or as a filter in the frequency domain, with a $H(s) = 1/s$ transfer function. If we think of integration as a filter, then we quickly realize that $H(s) = 1/s$ is not realizable, and so we will use approximate-integrators. The common approximations are that the integrator is either a single-pole system or a double-pole system. For any of these cases, we can write down the exact formula that represents (non-ideal) integration, as described in the modeling section. This section concerns the constraints on the poles of the integrator.

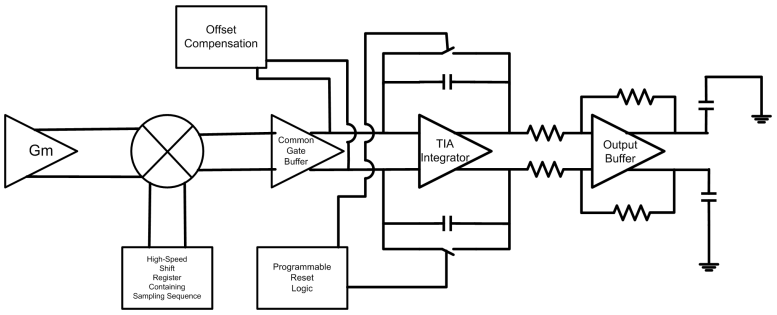


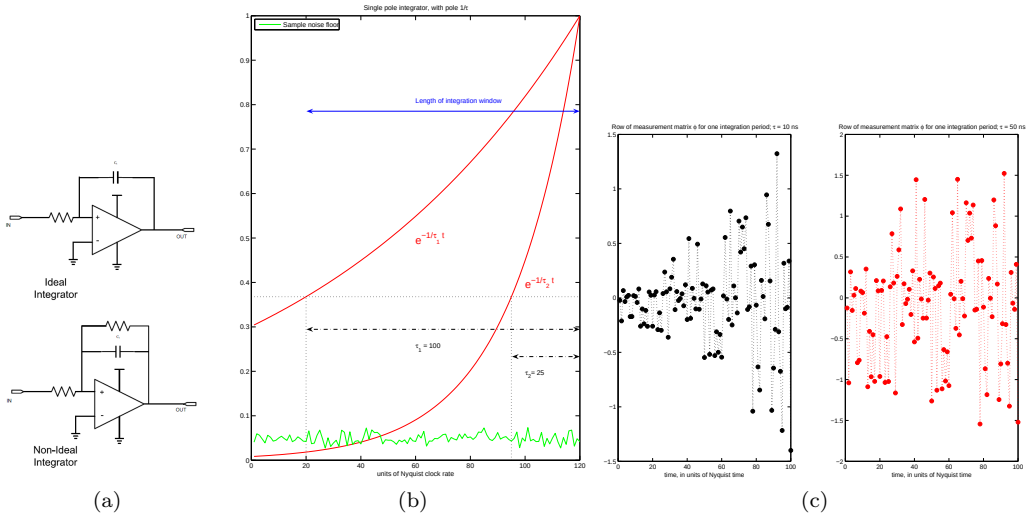
Figure 2.32: A more detailed diagram of the blocks in a single channel (after the LNA). See also Figures 2.11 and 2.12.

At low frequencies, the system is dominated by the smallest pole, also called the "inner pole". The transfer function of a single-pole system, with pole a, is $H(s) = 1/(s + a)$, and the impulse response is $h(t) = e^{-at}$. If the pole is large, then h decays quickly in time. See Figure 2.33 for a visualization.

2.6.1 General constraints

The ideal RMPI uses an exact integrator, so $H(s) = 1/s$. A first-order low-pass filter has attenuation of 20 dB per decade, which means exactly that the attenuation at high frequencies obeys $1/s$, so an integrator is quite similar to a low-pass filter at high frequencies. In papers such as [TLD+10], it is mentioned that an exact integrator is not necessary, and any similar low-pass filter works: "It suffices to perform low-pass filtering before the samples are taken." The actual story is a bit complicated, since the performance depends heavily on the bandwidth of the low-pass filter. We present results from a preliminary Northrop Grumman design that tried to approximate a band-pass filter (see Figure 2.34), and see that it has bad performance. Typical results are shown in Figure 2.35.

Consider an ideal band-pass filter with large bandwidth, such as from $[-f_s/2, f_s/2]$. That is, $H(s) = 1, |s| \leq f_s/2$ (or any constant) and $H(s) = 0$ otherwise. We can show how this system will perform badly. Consider two inputs x_1 and x_2 with frequencies f_1 and f_2. After mixing with the chip sequence $c(t)$, the spectrum of the signal is just a convolution of $\hat{c}(f)$ with f_1 or f_2. Since H has a flat response, the Fourier transform of the first signal $\mathcal{F}(x_1 c)$ and the second signal $\mathcal{F}(x_2 c)$



(a) (b) (c)

Figure 2.33: Plot (a) a block diagram of a realistic integrator, showing the parasitic resistance. It is impossible to make an integrator without a pole a. Plot (b) shows $h(T_{\text{int}} - t)$ where h is the impulse response $h(t) = e^{t/\tau}$ and $\tau = 1/a$ is the time constant. If the time constant is too small, then the correlation with the signal only occurs for the time right before the sampling, which is bad for sampling radar pulses. See also Figure 2.8. The right plot (c) shows the rows of a Φ matrix from the single pole model, with $\tau = 10$ ns (left) and $\tau = 50$ ns (right). The version 1 RMPI design has $\tau = 22$ ns, and the version 2 RMPI design has $\tau = 3333$ ns.

only differ in the region $[f_s/2 - \Delta f, f_s/2 + \Delta f]$ (and the negative frequency counterpart) where $\Delta f = |f_1 - f_2|$. If $f_1 \simeq f_2$, then the difference in the measurements of the system are determined by integrating the integral of the chip sequence Fourier transform over $[f_s/2 - \Delta f, f_s/2 + \Delta f]$. So for similar frequencies, the measurements will not be significantly distinct, and therefore reconstruction by any method is not stable.

This agrees with the time-domain intuition. The larger the flat response of the filter $H(s)$, the more similar the system is to being a delta function in time. As we've argued, a delta function in time is not good for compressed measurements since it is reasonably coherent with radar pulses.

2.6.1.1 Northrop Grumman integrator design

To summarize the above argument, if $H(s) = 1$, then $H(if_1)$ and $H(if_2)$ are indistinguishable. If $H(s)$ is a bandpass filter with large band, then $H(if_1)$ and $H(if_2)$ are distinguishable, but not robustly so. A much better frequency response is $H(s) = 1/s$ since then $H(if_1)$ and $H(if_2)$ are distinct for even small shifts $|f_2 - f_1|$.

Shifting. If the integrator has a large pole, then $h(t)$ is narrowly concentrated in time. See Figure 2.33 for a depiction. The first measurement from the system is $\int_0^{T_{\text{int}}} x(t)c(t)h(T_{\text{int}} - t)dt$, so this heavily weights $x(T_{\text{int}})$ and earlier, and ignores $x(0)$. Subsequent measurements will weight regions just before $x(nT_{\text{int}})$. Thus parts of the signal x are almost completely ignored by the measurements. An obvious fix is to shift/stagger each channel by a unique amount. For example,

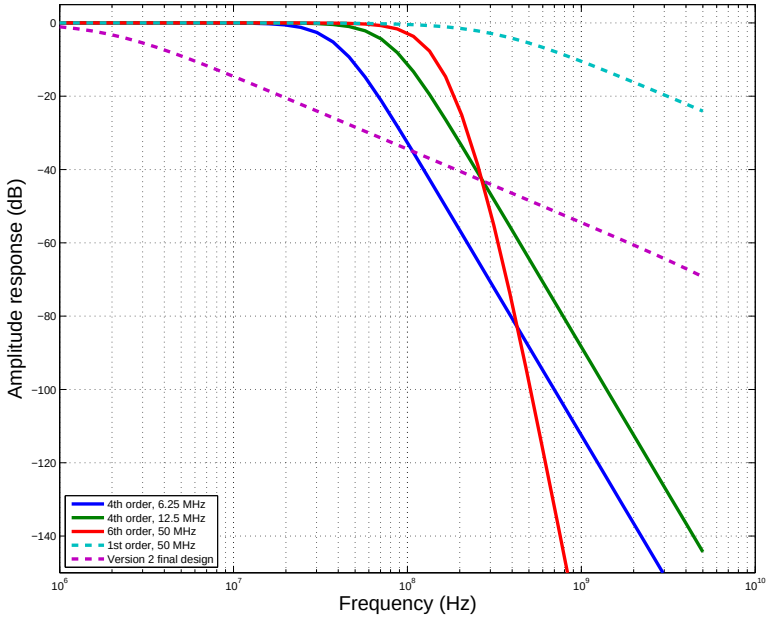


Figure 2.34: Bode plots of preliminary NG integrator designs (solid lines), proposed modification (50 MHz single pole), and the Caltech ver. 2 final design (302 KHz pole; see Table 2.4). The integration filter should have its first pole as close to DC as possible so there is a "memory" effect. These NG designs performed poorly (see Figure 2.35) because they are too similar to an ideal low-pass filter.

channel one integrates from $[0, T_{\text{int}}]$, channel two integrates from $[T_{\text{int}}/8, 9/8T_{\text{int}}]$, etc.

Consider the preliminary Northrop Grumman designs shown in Figure 2.34. We also include a slightly improved version with its lowest pole at 50 MHz pole. The top of Figure 2.35 shows that even for the improved 50 MHz version, we fail to reconstruct a pulse; using a Bernoulli matrix, or using the Caltech design, the same pulse was reconstructed, so this is an indication that the integrator is underperforming. The bottom of the plot shows the reconstruction when using the same design but with shifted channels. Performance is much better. To systematically test the effect of shifting, we use the single-pole model, and vary the time constant $\tau = 1/a$ and vary the amount of shifting. We use $N_{\text{int}} = 100$, and consider offsetting each channel from one another by $0, 12, 25$, or 50. Since there are 8 channels, the offset of 12 per channel makes the most sense, since then no two channels align. Figure 2.37 shows the results. For large time constants, the error is small regardless of shifting. For a time constant of $12.5\Delta T$, the error is very large (relative error is about 1) without using shifts. Using any of the nonzero shifts, the relative error for this time constant goes down to between .02 and .1. Interestingly, the exact amount of shift does not seem to make a big difference. Because of these results, we highly recommend staggering the channels. The only reason not to stagger channels is for ease of calibration. The current version 2 design can be programmed to shift by an arbitrary amount.

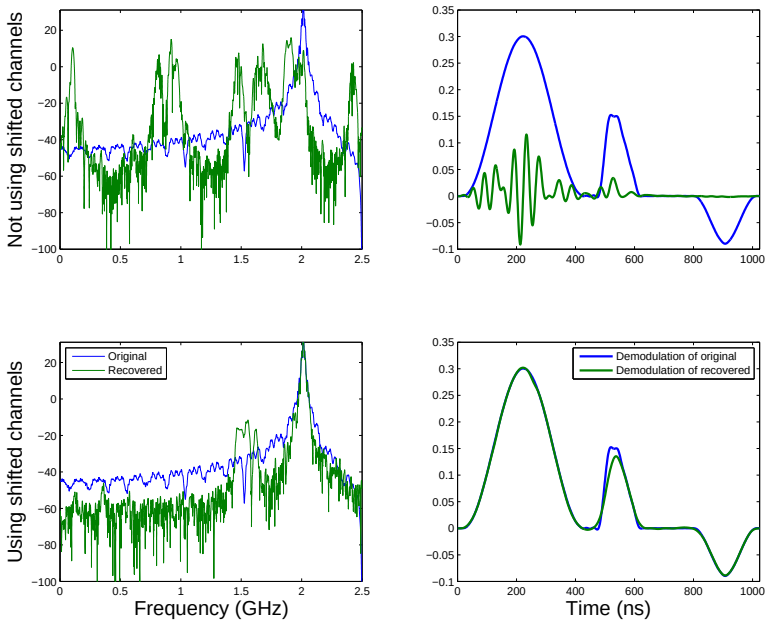


Figure 2.35: Simulation using a preliminary Northrop Grumman design, using first-order filter 50 MHz pole and 4 channel at 100 MHz each (see Figure 2.34). Some of the initial NG designs were too similar to band-pass filters, rather than integrators, and lacked the memory necessary to encode pulses. This was the best model tested (February 2009). The top row is with all 4 channels aligned; the bottom row has shifted/offset each channel by a quarter of the integration period. This shifting is extremely helpful.

2.6.1.2 Multipole systems

From a circuit design point-of-view, creating a single-pole integrator with a very small pole a is difficult. In version 1 of the RMPI chip, the lowest achievable pole was about 44 MHz. If we allow a multi-pole system, then it is much easier to push the first pole closer to 0. By doing this, the version 2 RMPI chip has its first pole at 302 kHz, and a second pole at about 300 MHz. With such a small first pole, the impulse response $h(t)$ is very flat over the integration time $[0, T_{\text{int}}]$. Figure 2.38 shows matched filter tests of an exact 300 kHz integrator, and compares recovery to that of a Bernoulli ± 1 matrix. The results are averaged over 400 samples, and the matched filter requires no parameters, so this test is quite reliable. The single-pole model without stagger suffers slightly at high noise level, but not by a significant amount, since the time constant $1/(300\text{ kHz})$ is very long. This test adds noise of the form $\Phi(x) + z$, instead of $\Phi(x + z)$, since for the latter model there is no discernible difference in performance. The variance of z was adjusted to account for the gain of the system. With the former noise model, the additive noise z will make it slightly more difficult to recover x at the beginning of time periods. We conclude that the 300 kHz pole has little effect on the system, and is much better than the 44 MHz pole in the version 1 design.

But does the second pole near 300 MHz have an effect? Figure 2.39 shows several tests on different types of integrators. The left column shows a version of Φ generated from a Simulink model that incorporated a realistic version of the integrator, with several zeros and poles. It performs similarly

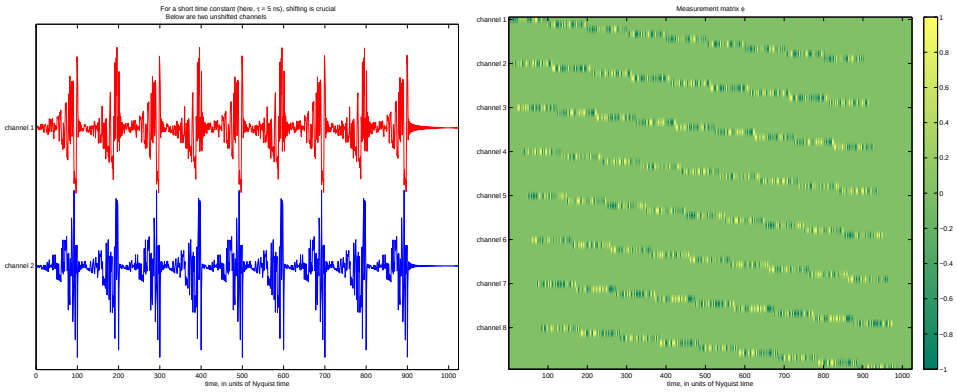


Figure 2.36: Figure (a) shows copies of $h(nT_{\text{int}} - t)$ for $n = 1, \ldots, 9$ overlaid, for $h(t) = e^{-at}$ where a is large. The top data (in red) are for channel one, and the bottom data (in blue) are for channel two. We see that without shifts, these align. For times $t = [0, 20\Delta T]$, $x(t)$ is barely measured. This can be fixed by staggering the channels. Figures (b) shows the matrix Φ of a staggered system.

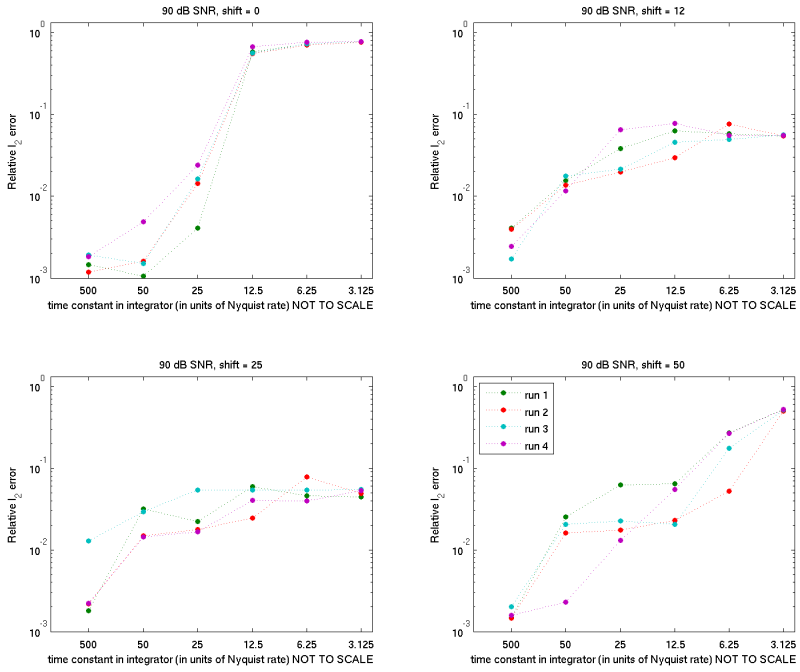


Figure 2.37: The results of four independent reconstructions using a single-pole model. The four plots show four different values of the channel stagger. In the 0 offset model, all channels aligned; in the 12 offset model, none of the channels aligned; in the 25 and 50 offset models, 2 and 4 channels aligned, respectively. For small values of the time constant, shifting by any amount is helpful.

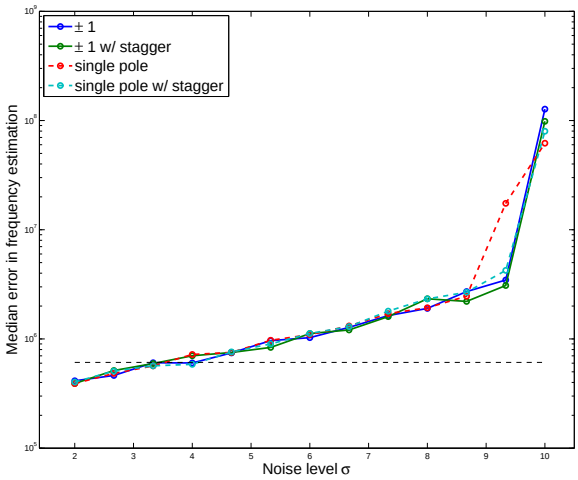


Figure 2.38: The ±1 model compared to the single-pole model (with 300 KHz pole). This compares performance, not modeling error. The single-pole model suffers a bit, due to the decay of the time domain transfer function $h(t)$. Shifting the channels helps this. $N = 2048$, and simulations used 2 realizations of each Φ matrix and then 200 sample noise vectors and carrier frequencies; x was a smooth pulse of length 100 ns. The dashed horizontal line shows the frequency resolution of the grid search.

to the other Φ models, so it seems that it has little effect.

Let the second pole be at location a_2. The transfer function of a two-pole system is

$$H(s) = \frac{1}{(s+a)(s+a_2)}$$

so that $H(s)$ decays like $1/s^2$ for frequencies much larger than a_2. To reason about the effects of the second pole, we simplify and consider

$$H(s) = \begin{cases} \frac{1}{a_2(s+a)} & s < a_2 \\ 0 & s \geq a_2 \end{cases}.$$

If a_2 is in the range 100 MHz to 1 GHz, this is quite similar to the scaled single-pole integrator $H(s) = \frac{1}{a_2(s+a)}$. The main limit is that if the PRBS has a short repetition period T_{chip}, then we must have $a_2 \gg 1/T_{\text{chip}}$, otherwise signals with frequency that is far from a harmonic of $1/T_{\text{chip}}$ will generate very small measurements. For the current chip design, $T_{\text{chip}} = 128\Delta T$, so $1/T_{\text{chip}} =$ 39 MHz, and hence a practical lower-limit of a_2 is about 100 MHz, since otherwise all but one or two of the spikes in the spectrum of $x(t)c(t)$ will extremely attenuated. With only one or two of Fourier series of c in the passband, it is much more likely for two inputs to generate similar measurements, and hence the system is less robust. The small bandwidth would also exacerbate the power difference between inputs that are exact harmonics of $1/T_{\text{chip}}$ and those that aren't.

To summarize the findings, we can view the integrator as a low-pass filter in the sense that we do not need frequencies above about 100 MHz, but the passband behavior of the filter must not be

Model	Zero	First pole	Second pole	Third pole
NG InP	NA	12 MHz	NA	NA
Caltech ver. 1	NA	44 MHz	NA	NA
Caltech ver. 2	360 MHz	302 KHz	361 MHz	1.696 GHz

Table 2.4: Poles and zeros of the integrating filter for various design. Poles are calculated at 3 dB cutoff from magnitude response. See also Figure 2.40.

flat! The exact poles and zeros of the RMPI are shown in Table 2.4, and the extracted frequency response of the integrator, taken from SPICE simulations, is shown in Figure 2.40.

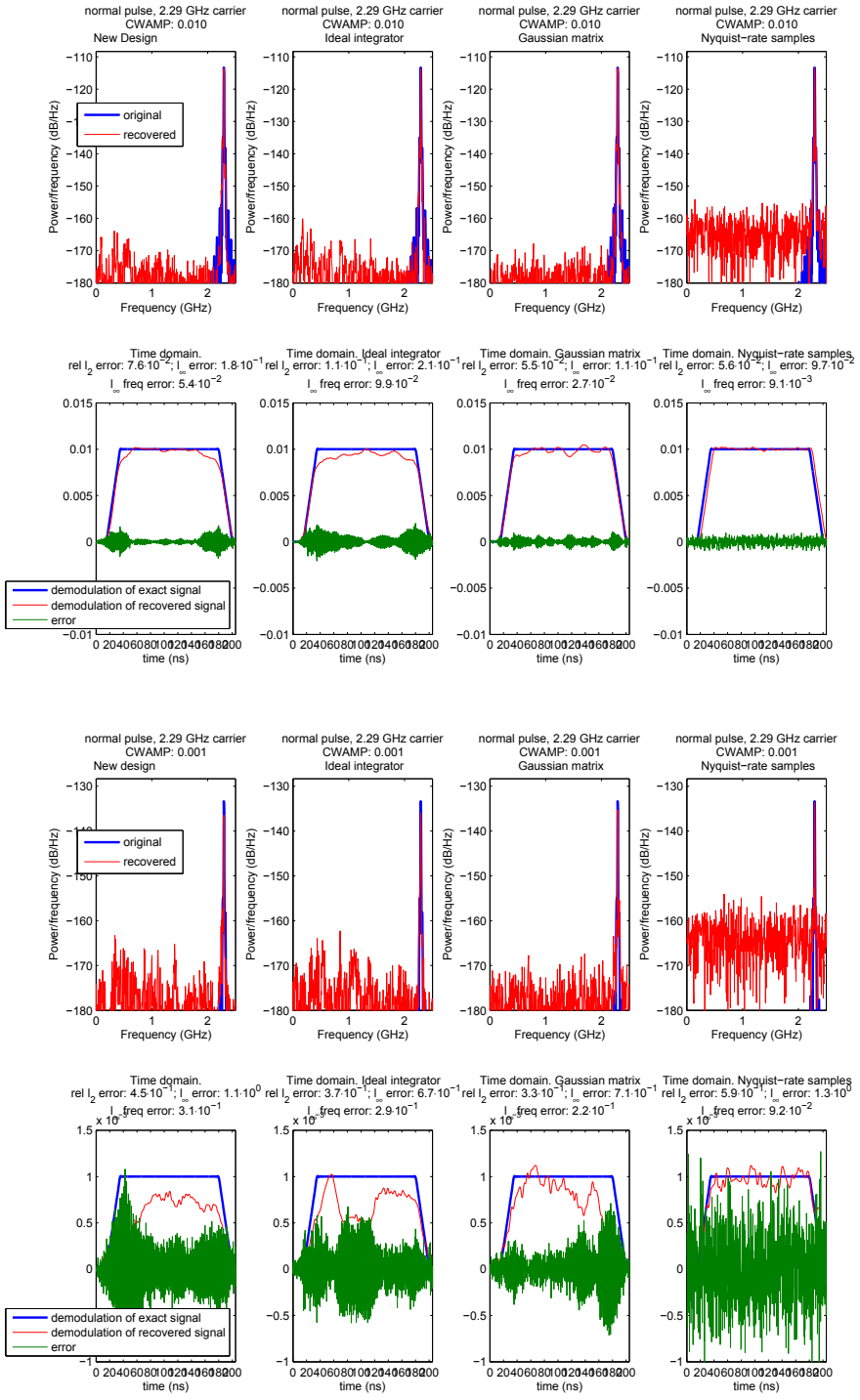


Figure 2.39: Does the new integrator suffer compared to, say, a Gaussian matrix? Not much. This test used a pessimistic noise level, and then ran three different amplitudes. For each test, 4 types of Φ matrix were used: a realistic version from the calibration, a single-pole model, a matrix with iid $\mathcal{N}(0,1)$ entries, and a Nyquist sampled version ($\Phi = I$). There is no significant difference. Only the two smallest signals are shown, since the large amplitude signal had visually perfect reconstruction.

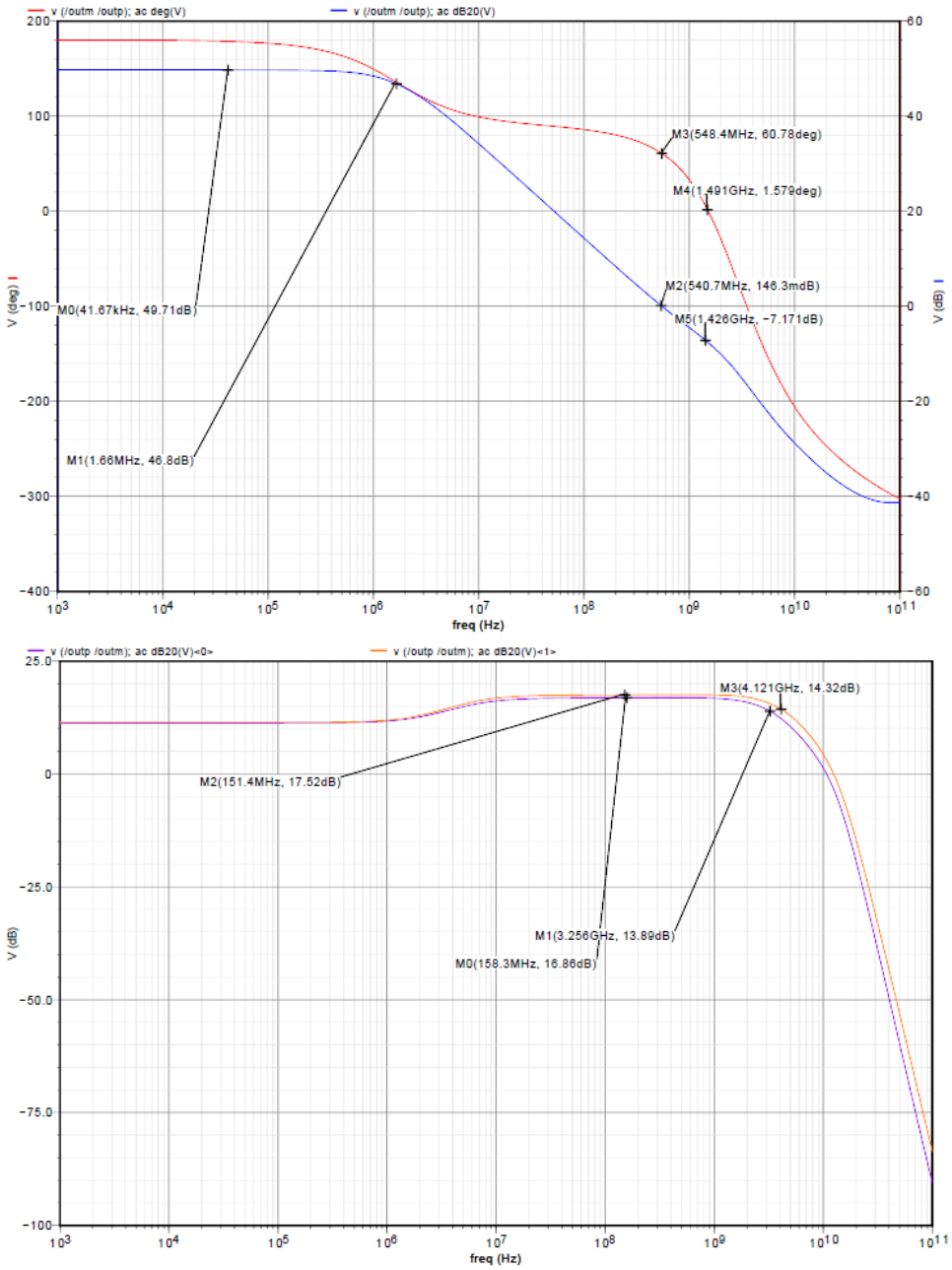


Figure 2.40: Top: integrator frequency response. Bottom: LNA frequency response. Blue curves are gain (scale on right axis), red curves are frequency response (scale on left axis). Results are from SPICE-extracted simulations. See Table 2.4 for values of poles and zeros. Other major blocks were modeled as well, and the resulting design parameters used to inform the Simulink model.

2.7 Recovery

This section covers the reconstruction of a pulse from its samples, assuming that a linear model Φ of the system is available. One of the key computations is solving an ℓ_1 minimization problem. This was originally solved using a version of l1Magic [CR07b] modified to allow the analysis formulation. Subsequently, we developed the algorithm NESTA (see Chapter 3) which is much faster. Detailed considerations of ℓ_1 solvers are discussed in Chapter 3 and 4. This discussions below assumes the existence of a ℓ_1 solver.

To give an idea of the computational time involved, a very high accuracy solve for $N = 1024$ with NESTA as the ℓ_1 solver, using several reweighting steps, takes 39 seconds on an i7 quad-core computer at 1.6 GHz. For lower accuracy computations, recovery time is closer to 10 seconds. Of the 39 seconds, roughly 65% of the time is spent computing the Gabor frame code. This code spends about half its time with $\mathcal{O}(N)$ re-arrangements, and the other half with $\mathcal{O}(N \log N)$ FFT calls. Most of this code is multi-threaded, and a reasonable estimate is that it would take 150 seconds to run the entire recovery on a single core (when desiring extremely high accuracy). This same i7 machine reaches about 1.8 Gflop/s per core on the FFT using the LINPACK Benchmark [DBMS84], so the computation requires roughly 270 Gflop. A medium-precision reconstruction requires about 100 Gflop. As of 2011, good GPU cards run about 500 to 1000 Gflop/s, so even with a special GPU implementation, and assuming no major overhead losses, the computation takes .1 to .2 seconds. Since $N = 1024$ represents $T = 204$ ns, this is a factor of at least $5 \cdot 10^5$ away from real-time recovery. More work is needed to bring this down to a reasonable level. Matched filtering will likely be key. One might imagine a hybrid system that uses crude methods for pulse detection and high-accuracy ℓ_1 solves only when a pulse is detected.

Practical aspects of signal recovery for RMPI have not been discussed in a single paper, but the ideas presented here draw upon many sources. Some ideas were discovered independently of known results. For the matched filter, see related ideas in [DDW$^+$07, DB10], as well as the recent paper [ERW11]. There is also recent unpublished work by Michael Wakin on using a notched filter to project out interfering signals; this relies on a result from [DW10].

2.7.1 Matched filter

An alternative to ℓ_1 minimization is the compressive matched filter, which is discussed in this section. If there is much prior knowledge about the signal—for example, the exact shape of the pulse envelope is known—then the matched filter is a very powerful technique, for in this case it is robust, fast, and optimal. In general, the pulse envelope is not known well, and in this case ℓ_1-based recovery methods are preferred. For the purposes of testing the system, the matched filter is used since it represents the best recovery possible, and so exposes the limits of the system. Because it is fast, it

allows hundreds of simulations in a short amount of time.

Consider our discrete model $b = \Phi x_0 + z$. Assume x belongs to some parameterized set $\mathcal{X}$; for example, $\mathcal{X}$ could be the set of Fourier modes on a grid of length N. We will actually allow the set $\{ax : x \in \mathcal{X}, a \in \mathbb{C}\}$ where c contains amplitude and phase information. Both x and c may be complex, but we assume all other variables are real.

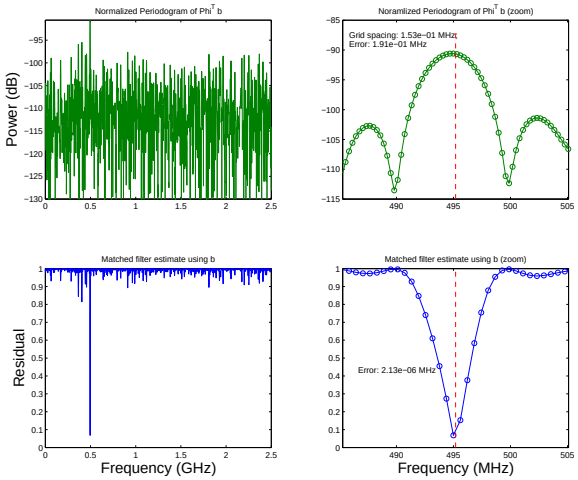


Figure 2.41: Matched filter estimation of an off-grid pure tone. The periodogram estimates (top row) are biased, even after accounting for the norms of the FFT of Φ.

Theorem 2.7.1 (Matched filter in "compressed domain")**.** *If $z \sim \mathcal{N}(0, \Sigma)$ and $a_0 x_0 \in \mathcal{X}$, then the maximum likelihood estimator x_{MLE} is given by the matched filter estimate*

$$x_{MF} = \underset{x \in \mathcal{X}, a \in \mathbb{C}}{\operatorname{argmin}} \|b - \Phi(ax)\|^2_{\Sigma^{-1}}.$$

Proof. Trivial, since the log-likelihood function is

$$\ell(\hat{x}) \equiv \log \mathbb{P}(b | x = \hat{x}) = \log \mathbb{P}(z = b - \Phi\hat{x}) = \text{const} - \langle b - \Phi\hat{x}, \Sigma^{-1}(b - \Phi\hat{x}) \rangle.$$

□

We'll specialize this to the following parametric model. Let x be a real-valued tone which we observe for $T = N/f_s$ seconds, with arbitrary phase and frequency (the frequency can be off-grid, but still within $[0, f_s/2]$), and windowed by a *known* window w (with $W = \text{diag}(w)$). Let $f_\gamma(n) = e^{i\omega_\gamma n/f_s}$ be a complex exponential and $F_\gamma = [f_\gamma, \bar{f}_\gamma]$ be a $N \times 2$ matrix. For the moment, let γ be fixed. The reason we use f_γ and its conjugate (i.e., the negative frequency) $\bar{f}_\gamma$ is that the measurements b are real valued. This approach is equivalent to minimizing $\|b - \text{Re}(\Phi W a f_\gamma)\|$ over $a \in \mathbb{C}$.

We can solve for the coefficient $a \in \mathbb{C}^2$ easily (c has two entries, but it will turn out to be

conjugate symmetric, so it only has one complex-valued degree-of-freedom):

$$a_\gamma \equiv \underset{a\in\mathbb{C}^2}{\operatorname{argmin}} \|b - \Phi W F_\gamma a\|^2_{\Sigma^{-1}} = (F_\gamma^* W \Phi^T \Sigma^{-1} \Phi W F_\gamma)^{-1} F_\gamma^* W \Phi^T \Sigma^{-1} b, \tag{2.7.1}$$

$$\min_{a\in\mathbb{C}^2} \|b - \Phi W F_\gamma a\|^2_{\Sigma^{-1}} = \|b\|^2_{\Sigma^{-1}} - c^*(F_\gamma^* W \Phi^T \Sigma^{-1} b). \tag{2.7.2}$$

In practice, we choose f_γ so that ω_γ is on a (possibly over-sampled) grid. To minimize over all γ on this grid, we compute $\tilde{b} = W\Phi^T\Sigma^{-1}b$ once, and then compute $F_\gamma^*\tilde{b}$ for all frequencies simultaneously using a (possibly over-sampled) FFT. For every γ, the 2×2 matrix $(F_\gamma^* W \Phi^T \Sigma^{-1} \Phi W F_\gamma)^{-1}$ must be computed, but this is independent of b, so it can be computed once and stored. So amortizing this cost, we can find the best frequency ω_γ up to a fixed frequency resolution ϵ by computing over-sampled FFTs with complexity proportional to $\mathcal{O}((N2^{-\epsilon})\log(N)/\epsilon)$. Σ^{-1} is computed via a pseudo-inverse or regularized inverse, in order to make the process more robust.

For even higher accuracy in the frequency, a one-dimensional optimization using Matlab's `fminbnd` is performed in the region near the peak frequency found by the FFT method. This function does not use the FFT, but is efficient since it typically only requires a handful of calls; it is also possible to use the Goertzel algorithm, but this would probably not offer an advantage. See Figure 2.41 for a plot of the minimal values of the functional in a region near a minima. In special cases, this function is convex [ERW11] within small regions near the minimum. For our case, empirical results suggest that it is also convex in a reasonably wide interval around its minimum. For very noisy signals, the high accuracy search is not performed, since this level of precision is generally beyond the accuracy level of the estimation.

Note that with the matched filter, there is less difference between input signals x that are "on-grid" and "off-grid". The spectral leakage which hurts ℓ_1-based recovery is no longer an issue, but the sacrifice is the simpler signal model.

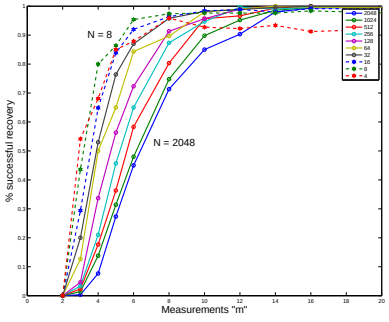


Figure 2.42: Rate of successful recovery via matched filter as function of M and N. "Success" is defined as frequency error less than .1 KHz. Input is tone of random phase and frequency. Matrix Φ is chosen iid signed Bernoulli ± 1. Recovery shows a strong dependence on M, and a weak dependence on N, as expected.

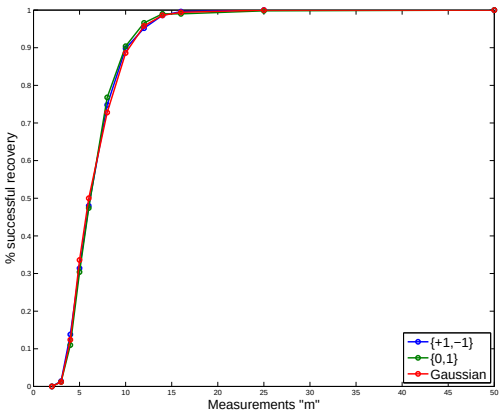
Figure 2.42 shows recovery of a randomly chosen sinusoid from M noiseless compressed measurements, using a $M \times N$ matrix Φ with each entry iid signed Bernoulli. The signal has three

parameters: amplitude, frequency, and phase, so typically $M \geq 3$ measurements are needed to have 100% recovery. Note that recovery rates are lower when N is large, since this enlarges the size of the ambient vector space. Because the frequency is chosen uniformly at random from $[0, f_s/2]$, this is an infinite-dimensional problem and more closely resembles Logan's phenomena [Log65, DS89] than compressed sensing. To see how we can recover from so few measurements, consider the case when the amplitude is known and we have two time samples, one at t_1 and one at $t_1 + \Delta T$, so $M = N = 2$. Let the signal be $\sin(\omega t + \theta)$, and the measurements are $b_1 = \sin(\omega t_1 + \theta)$ and $b_2 = \sin(\omega t_1 + \omega \Delta T + \theta)$. Then ω and θ can be determined by solving a 2×2 system of equations using the data $\sin^{-1} b_1$ and $\sin^{-1} b_2$; this is not always solvable since $|\sin^{-1} b_1 - \sin^{-1} b_2| < \pi$, but if the signal is band-limited then $\omega \Delta T < \pi$ and there is a unique solution.

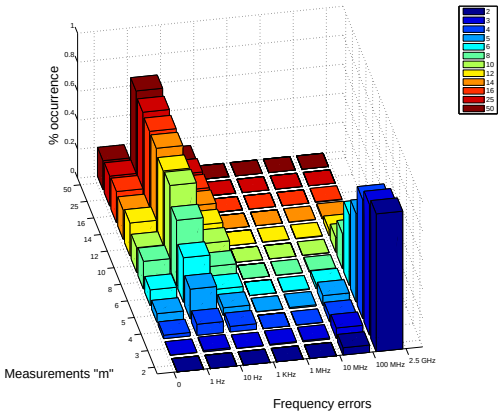
For the special case when Φ is a subset of the identity, and the frequency is on-grid, then we can be more specific. This is now a finite-dimensional problem. If N is prime, then Theorem 1.1 in [CRT06] guarantees that the the matched filter will recover all signals composed of $K \leq M/4$ "on-grid" tones (the extra factor of 2 is because in the change-of-variables from time to frequency, a signal with K tones has $2K$ nonzeros in the Fourier transform; think of this as the price to pay for unknown phase). Furthermore, if $K > M/4$, then one can construct an un-recoverable signal.

In general, now let N be arbitrary, and again let the signal x have K "on-grid" tones so that it is exactly $2K$ sparse in the Fourier basis F, $x = F\alpha$ with $\|\alpha\|_0 = 2K$. If Φ is a $M \times N$ matrix with iid Gaussian entries, then ℓ_0 recovery will successfully recover x whenever all $M \times 4K$ submatrices of ΦF have only trivial nullspace. Thus typically $M \geq 4K$ is a necessary condition to guarantee recovery. The results in Figure 2.43, which have $K = 1$, show that recovery with $M < 4$ is very rare. When $K = 1$, the matched filter recovery is equivalent to ℓ_0 minimization.

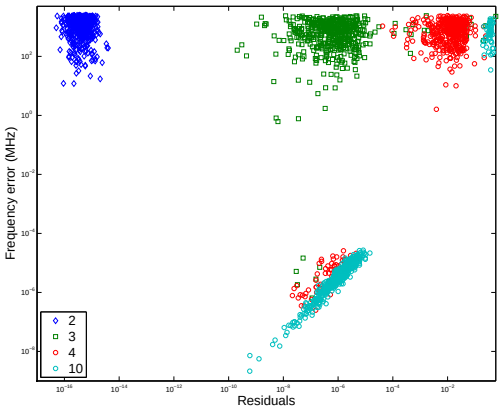
Figure 2.43 shows the results of some matched filter experiments which give an idea of the limits of recovery. The matrix is $\Phi \in \mathbb{R}^{M \times N}$ with $N = 1024$, and the results are a function of M; we do not yet impose a block structure on Φ or use a windowed signal. For comparison, the RMPI has $M = 80$ when $N = 1024$, it is well-above the threshold of recovery. The input signal is a pure sinusoid with random phase and (off-grid) frequency, and no noise. The matched filter is extremely computationally efficient when multiple trials use the same Φ matrix, so for each setting, 5 different Φ matrices were generated, and for each matrix 100 trials were carried out.



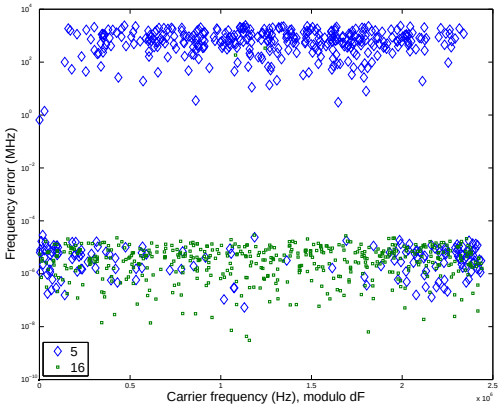
(a) Rate of successful recovery, N fixed. "Success" is defined as frequency error less than .1 KHz.



(b) Histogram of size of frequency errors



(c) Residual of matched filter



(d) Effect of off-grid frequencies

Figure 2.43: (a) uses data from three different Φ ensembles: either each entry $\Phi_{i,j}$ is chosen from $\{+1, -1\}$ uniformly, or from $\{0, 1\}$ uniformly, or from $(-\infty, +\infty)$ with a normal distribution. In all subsequent subfigures, we only show data from the $\{+1, -1\}$ model since the results are not significantly different among the ensembles. In (a), we confirm our intuition that more samples (larger M) improves the chance of recovery. In (b), a histogram of the type of frequency errors is shown for each value of M. For large M, most errors are small, while for small M (e.g., $M = 5$), errors are either small or catastrophic. Subfigure (c) shows the relationship of the matched filter residual (2.7.2) to the frequency error. For large M, there is a clear relationship, likely due to how well the convex optimization solver worked. For $M = 2$, the error was always catastrophic, but the residuals were small since the matched filter is an exactly-determined problem. For $M = 4$, only the non-catastrophic errors show a relationship between error and residual. Subfigure (d) plots the error as a function of how far the carrier signal frequency is from a "grid" frequency $dF = f_s/N_{\text{chip}} = 4.88/2$ MHz ($N_{\text{chip}} = N$ in this example; that is, the PRBS did not repeat). There is no discernible relationship for large M, which suggests that the matched filter recovery works equally well for tones that are on- and off-grid. For small M, there is a non-linear correlation: if the frequency was close to a grid-frequency, then a disaster was less likely to happen. But conditioned on whether a disaster occurs or not, the nearness to a grid frequency does not affect accuracy.

The matched filter can be used on real data as well, but in this case the exact pulse window is not known. The pulse window may be approximated by a rectangle, but a parametric search over the start and end of the rectangle must be performed; for speed considerations, this search may be done by a power analysis in the compressed domain. If more than one pulse is present, the matched filter proceeds in a greedy fashion by identifying the dominant pulse, then subtracting off

the pulse and repeating the process. This subtraction may not be accurate if the pulse window is not exactly rectangular, so we expect l1 recovery to perform better than matched filtering in this case. In general, if the window is unknown, matched filtering does not return an accurate estimate of the signal, but it can often return accurate pulse descriptor words (PDW) such as the frequency, phase, and time-of-arrival.

2.7.2 ℓ_1 recovery

Once data b have been taken, there are many options to recover an estimate $\hat{x}$ of the original signal x_0. This section reviews the basic techniques, plus many tricks that are essential in practice.

2.7.2.1 Analysis versus synthesis

Let $\mathcal{D}^*$ be the transform that induces sparsity in x, which is discussed in the subsequent section. For now, assume it is fixed. The two variants of ℓ_1 minimization are *synthesis*:

$$\underset{\alpha}{\text{minimize}} \ \|\alpha\|_1 \quad \text{such that} \quad \|\Phi\mathcal{D}\alpha - b\|_2 \leq \varepsilon, \tag{2.7.3}$$

and *analysis*:

$$\underset{x}{\text{minimize}} \ \|\mathcal{D}^* x\|_1 \quad \text{such that} \quad \|\Phi x - b\|_2 \leq \varepsilon. \tag{2.7.4}$$

The dictionary $\mathcal{D}$ is assumed to be a tight frame, so that $\mathcal{D}\mathcal{D}^* x = x$. If $\mathcal{D}$ is a basis, then the two programs are equivalent by a change-of-variables, since $\mathcal{D}^*\mathcal{D}\alpha = \alpha$. In general, $\mathcal{D}$ is an overcomplete dictionary, and the two programs are not equivalent. That is, if $x = \mathcal{D}\alpha$, it is not necessarily true that $\alpha = \mathcal{D}^* x$. The differences will be further explored in §3.6.1 and §4.7.5. Since $\mathcal{D}$ is overcomplete (say, $N \times D$ and $N < D$), x is N-dimensional and α is D-dimensional. The analysis problem has fewer variables, but the same number of constraints, so it is effectively more constrained.

As a simple example of the differences, let $\mathcal{D} = (I, I)$, and $x = e_1$ where e_1 is the first unit vector. By analyzing x, we have $\mathcal{D}^* x = [e_1; e_1]$ (in Matlab notation) which is 2-sparse. Via synthesis, we have $x = \mathcal{D}\alpha_1 = \mathcal{D}\alpha_2$ for $\alpha_1 = [e_1; 0]$ and $\alpha_2 = [0; e_1]$, and even $x = \mathcal{D}(t\alpha_1 + (1-t)\alpha_2)$ for any t. The synthesis coefficients *can* be sparser, but not they are unique and less stable

Some preliminary results about the differences appear in Chapter 5 of a recent PhD thesis [Ran09], where it is suggested that analysis often recovers signals more effectively. The Phase 0 report on the DARPA project reaches the same conclusion. The work in [EMR07] reports that analysis performs better than synthesis when using standard test images, and [CWB08] reports similar results on pulse signals. That work also notes that when reweighting, the differences are exacerbated. More experiments appear in [SED04, SED05]. Recent theoretical work [CENR11] has shown that analysis enjoys similar theoretical properties as the synthesis formulation does.

We propose an additional explanation, based on the observation that reweighting exacerbates the difference between synthesis and analysis. Consider the case of a dictionary that is heavily redundant. To simplify, we take this to the extreme, and assume that there are two atoms ϕ_1 and ϕ_2 which are identical; write α_1 and α_2 for their respective coefficients.

In synthesis, since α is the object of interest, it is intended to be sparse, so even if there is energy in the ϕ_1 component, it is likely that either α_1 is zero or α_2 is zero. Without loss of generality, let $\alpha_1 > 0$ and $\alpha_2 = 0$. If ℓ_1 minimization were performed on a slightly perturbed dataset $\tilde{b} \simeq b$, perhaps $\tilde{\alpha}_1 = 0$ and $\tilde{\alpha}_2 > 0$. But once reweighting is applied, the symmetry is broken. If $\alpha_1 > 0$ and $\alpha_2 = 0$, then the weights obey $w_1 < w_2$, so at all future reweighting steps, it is nearly guaranteed to keep $\alpha_1 > 0$ and $\alpha_2 = 0$. In a sense, the support has been locked-in. In a realistic case, ϕ_1 and ϕ_2 are not identical, so their coefficients are not interchangeable. If α_1 and α_2 are incorrectly identified after the first round, then reweighting will not fix this, and α becomes trapped in a local minimum. The opposite problem might happen too: reweighting is used to force some coefficients to zero, but if there are nearly interchangeable coefficients, forcing one coefficient to zero will just cause another similar coefficient to grow larger.

In contrast, analysis is much more uniform. If $\phi_1 \simeq \phi_2$, then α_1 is guaranteed to be close to α_2. Analysis can better exploit the reweighting technique. The downside of analysis is that extreme sparsity is not as possible, but this may be worth it for the trade offs.

To determine which formulation to use for the RMPI recovery, realistic numerical simulations were performed. A few of the results are shown in Figure 2.44. The results were overwhelmingly in favor of analysis, so all subsequent experiments have used analysis. However, these simulations were in the early phase of the project, and it would be useful to re-run the comparison tests using the better calibrated Φ matrices.

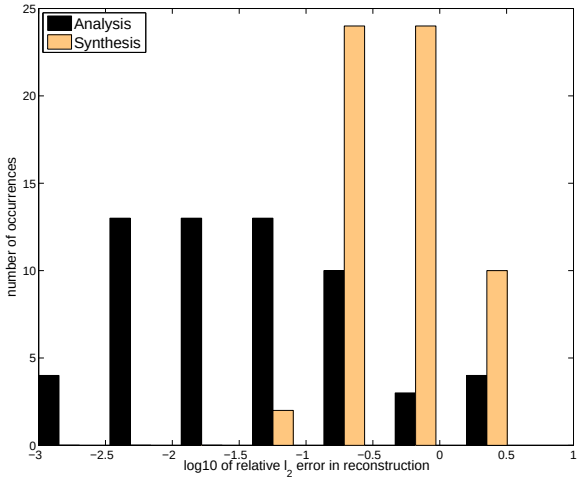


Figure 2.44: Performance of analysis formulation and synthesis formulation. Original tests from 2008 using l1Magic. Repeated over independent runs (noise varied) and 4 signal models: narrow and wide pulses, and low and high frequencies.

2.7.2.2 Dictionary choice

A general purpose reference for this section is [Mal08], but we don't cite specifics since most of this material is well known. Gabor frames go back to at least Gabor in the 1940s, who was inspired by musical scores. This type of harmonic analysis has been well-studied in many communities due to its widespread usefulness and the interest in wavelets beginning in the 1980s.

Dictionaries. A dictionary $\mathcal{D} \in \mathbb{R}^{N \times D}$ of D unit-norm atoms ϕ_d is chosen to represent the radar signal x: $x = \mathcal{D}\alpha = \sum_{d=1}^{D} \alpha_d \phi_d$ for some coefficients α. It is also possible to consider complex-valued dictionaries, but since x is real-valued, it is always possible to convert a length D complex coefficient vector α to a length $2D$ real-valued vector, so we assume wlog that α is real. The "analysis" operation is $\mathcal{D}^* x$, and the "synthesis" operation is $\mathcal{D}\alpha$. The statements below are tailored to finite dimensions, but we remark that most of these hold in any separable Hilbert space. In Hilbert space, a dictionary must have a dense span; in finite dimensional spaces, all we require is that $\mathcal{D}$ has full row-rank. If it doesn't have full column-rank, we say $\mathcal{D}$ is redundant or over-complete.

We are typically interested in over-complete dictionaries, so $D > N$, and thus it is not possible for $\mathcal{D}^*\mathcal{D} = I_D$, although it is possible for $\mathcal{D}\mathcal{D}^* = I_N$. The dictionary is a *frame* if $\mathcal{D}\mathcal{D}^*$ is non-singular, and a *tight frame* if $\mathcal{D}\mathcal{D}^*$ is a scaled identity aI_N, where we call a the frame bound. Thus a frame has a pseudo-inverse $\mathcal{D}^\dagger = \mathcal{D}^*(\mathcal{D}\mathcal{D}^*)^{-1}$, and for a tight frame, the pseudo-inverse is just a scaled version of the adjoint. The dual analysis operation is $\mathcal{D}^\dagger x$, and the dual synthesis operation is $(\mathcal{D}^\dagger)^* \alpha$. For tight frames, the analysis and dual analysis, and synthesis and dual synthesis, are the same, up to constants.

If $\mathcal{D}_1$ and $\mathcal{D}_2$ are both tight frames, then via block matrix multiplication, it is clear that $\mathcal{D} = [\mathcal{D}_1, \mathcal{D}_2]$ is also a tight frame.

There are three considerations when choosing a dictionary $\mathcal{D}$:

1. $\mathcal{D}$ should approximate the signal well, meaning that for every signal x, it can be represented $x = \mathcal{D}\alpha$ for some α that only has a few large coefficients.

2. The atoms ϕ_i of $\mathcal{D}$ should be as incoherent as possible; in other words, D should not be unnecessarily large.

3. All analysis and synthesis operations should be computationally efficient.

The signal model is that x is a radar pulse, which means it is localized in both time and frequency, so we seek a dictionary that approximates these signals well. For this reason, the Fourier basis is a poor choice, since all elements have full time support. However, due to the computation requirements, most realistic choices of dictionaries are based on the Fourier basis so that they can take advantage of FFT routines. Another class of dictionaries is based on wavelets, since there is a fast wavelet transform.

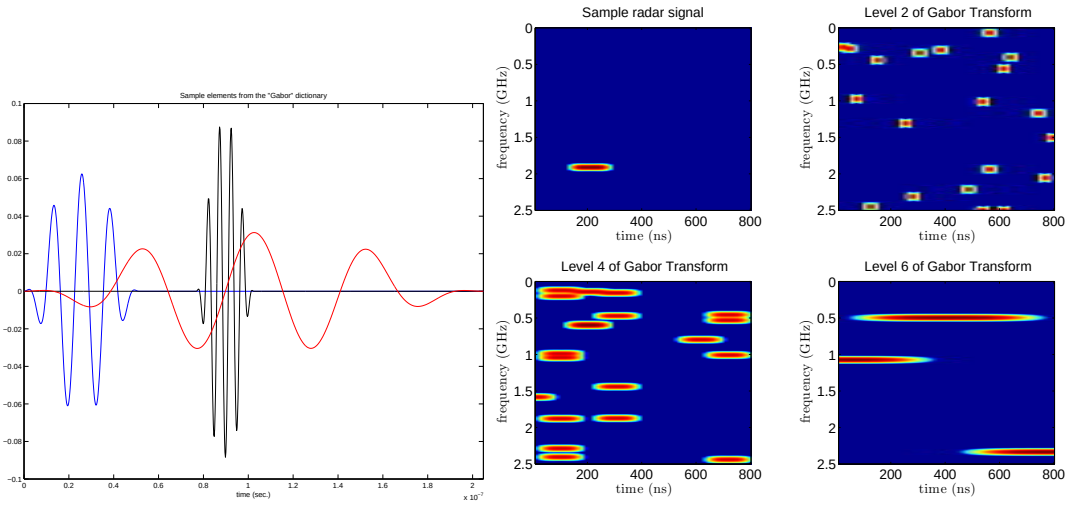


Figure 2.45: Sample elements from a multiscale Gabor frame. Left: time domain. Right: time-frequency plane

Gabor Frames. We consider Fourier-based dictionaries. Two major classes are the Gabor dictionaries and the lapped Fourier/DCT dictionaries (such as the modified discrete cosine transform used in MP3 compression). The two classes share similarities, and while we choose to present the dictionary in a Gabor framework, we have not lost much generality.

Let F be the $N \times N$ finite Fourier basis, with N columns f_d, where

$$f_d(n) = e^{i2\pi dn/N}.$$

Let g be a window function (e.g., Gaussian, sine), and write $G = \text{diag}(g)$. Define $g_m[n] = g[n - mM]$ to be a shifted version for some constant M, and similarly for G_m. Then a Gabor dictionary is

$$\mathcal{D} = [(FG_0), (FG_1), (FG_2), \ldots, (FG_{N/M-1})]$$

where we assume $M|N$.

The most useful Gabor dictionaries are those that are tight frames. To be a tight frame, we need

$$I_N = \mathcal{D}\mathcal{D}^* = F^*(G_1^2 + \ldots G_L^2)F.$$

In other words, g^2 must be a partition of unity when shifted by M. We consider only windows g such that g^2 is periodic in N.

The dictionary used for the RMPI relies on an iterated sine window:

$$g(x) = \sin(\pi/4(1 + cos(\pi x))), \quad x \in (-1, 1]$$

(implicitly, define $g(x) = 0$ outside $(-1, 1]$), and the discrete version is

$$g[n] = g(-1 + 2n/N), \quad n = 1, \ldots, N.$$

Using simple properties of sine and cosines, we see that $g(x)^2 + g(x+1)^2 = 1$, so this is a tight frame.

The trivial window $g_{\mathrm{F}} \equiv 1$ is valid ($M = N$), as is the window g_{T} which is only nonzero for a single discrete point. These can be viewed in the time-frequency plane as extreme types of tiling. Viewing the time-frequency (T-F) plane with time on the horizontal access and frequency on the vertical access (as in Figure 2.45), where as g_{T} corresponds to atoms that are very skinny (and also short). We would like a Gabor frame that tiles the T-F plane with atoms of compact support (so they approximate radar pulses well) and minimal overlap (to minimize coherence among the atoms). What can we expect? In the best case, we want exactly no overlap, which means that $\mathcal{D}$ is not only a dictionary but also a basis.

Unfortunately, the uncertainty principle prevents this, as quantified in Balian-Low theorem which states that no windowed Fourier *basis* exists if the window g is smooth and of compact support (clearly, if we allow g_{F}, then the $\mathcal{D} = F$ which is a basis). If we want well-localized T-F atoms, we must accept overlap, which is why $\mathcal{D}$ is usually over-complete.

We note that Mallat (§12.3.3 [Mal08]) describes a Gabor dictionary based on a discrete Gaussian window which is *nearly* a tight frame; the advantage of that particular construction is that it is translation-invariant and multi-scale. It may merit further consideration for the RMPI, but so far we stick to tight frames.

Multiscale Gabor Frames. Exploiting the property that concatenating two tight frames $\mathcal{D} = [\mathcal{D}_1 \mathcal{D}_2]$ gives a new tight frame, it is easy to build up a multi-scale Gabor frame.

Consider the window function g described previously, where $g(x)^2 + g(x+1)^2 = 1$ for $x \in (-1, 0]$. Let $g^{(2)}(x) \triangleq g(2x \mod (-1, 1])$, then $g^{(2)}(x)^2 + g^{(2)}(x + 1/4)^2 + g^{(2)}(x + 1/2) + g^{(2)}(x + 3/4) = 1$, so the dictionary defined with F and $G^{(2)}$ is also a tight Gabor frame. We can add this to the Gabor frame that used the original window, and the result is a new tight Gabor frame that has window functions at two levels. Clearly this can be repeated, so that windows of various widths are included in the dictionary.

A final twist on the windows is to introduce shifted versions. Since $g(x)^2 + g(x+1)^2 = 1$ and g^2 is periodic with period 2; it is also true that $g(x-1/2)^2 + g(x+1/2)^2 = 1$. So in addition to windows G of different widths, we can also include windows that are shifted. In practice, we typically include about 2 levels of narrow windows, and one shifted variant of each narrow window.

Frequency scalings. Previously, attention was restricted to the $N \times N$ Fourier basis F, which was windowed by some diagonal window matrix G, where the Fourier atoms are $f_d(n) = e^{i2\pi dn/N}$ for $d = 1, N$. It's also possible to use a zero-padded or sub-sampled Fourier frame. The $2\times$ zero-padded Fourier frame consists of atoms

$$f_d(n) = e^{i2\pi dn/(2N)} \quad \text{for} \quad d = 1, \ldots, 2N$$

so it has half the spacing in frequency. Likewise, a $2\times$ sub-sampled Fourier set consists of

$$f_d(n) = e^{i2\pi 2dn/N} \quad \text{for} \quad d = 1, \ldots, N/2$$

which, by itself, does not span the space.

The multiscale Gabor frame that we use in practice will adjust F to be oversampled when using a wide scaling of the window, and a sub-sampled Fourier set when using a narrow scaling of the window. A basic theorem by Daubechies (see [Mal08]) shows that the frame bounds (of a single-scale Gabor frame), and hence the redundancy, are directly related to the product of the frequency scaling and the time scaling.

In general, the parameters of the Gabor frame are picked via trial-and-error by testing which frames give the best reconstruction results. This also depends on the length N used. Because test signals are generated with a known window, it would be easy to overfit the parameters of the Gabor frame to work very well for certain test signals, and hence we do not try to fine tune the parameters. Real radar signals are generally "supposed" to be trapezoidal, but they are generated by waveform generators with limited bandwidth, which has the effect of smoothing the trapezoidal window. This is beneficial to recovery since the signal will have fewer sharp edges and hence a sparser frequency representation.

For a typical reconstruction, the Gabor frame is between $8\times$ and $64\times$ overcomplete, and usually between $12\times$ and $20\times$ overcomplete. The signal length N is between 2^{10} and 2^{14} depending on what calibration matrix is available and how much computational time can be spent. Hence for $N = 2^{14}$, the dimension D can be as large as $2^{20} \approx 10^6$. This was one of the main motivating factors for developing the rapid algorithms described in subsequent chapters. For example, NESTA solves a problem with size $N = 2^{12}$ in about 1 minute.

Given a dictionary $\mathcal{D}$, we solve either the ℓ_1 analysis problem which uses weighted norm $\|\mathcal{D}^* x\|_1$ and measurement matrix $A = \Phi$, or the ℓ_1 synthesis problem with norm $\|\alpha\|_1$ and measurement matrix $A = \Phi\mathcal{D}$. The conclusion of this thesis discusses possible improvements over using a Gabor frame by itself; model-based ideas will probably be useful in future versions.

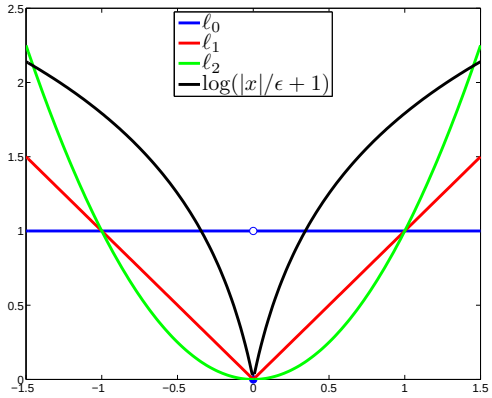


Figure 2.46: The ℓ_0 quasi-norm and various approximations to it. There are two features we wish to capture: a sharp cusp at 0, which will force small values to exactly 0, and a slow growth at infinity, so that large values are not excessively penalized. The ℓ_1 function is in some senses the best possible convex choice.

2.7.2.3 Reweighting

Consider the generic ℓ_1 synthesis problem

$$\min_x \|x\|_1 \quad \text{such that} \quad \|Ax - b\|_2 \leq \varepsilon.$$

The following discussion will also apply to ℓ_1 analysis, but we use the synthesis form for simplicity of exposition. In both cases, the reweighting vector will apply to an object in the coefficient domain (e.g. of size D) and not in the signal domain, but we use x to represent the object to be general.

The assumption is that x is sparse, and thus what we really would like is the solution to the problem

$$\min_x \|x\|_0 \quad \text{such that} \quad \|Ax - b\|_2 \leq \varepsilon$$

where $\|x\|_0$ is the quasi-norm which is the number of nonzero entries in x.

The reweighting approach is to solve a series of weighted ℓ_1 problems

$$\min_x \|Wx\|_1 \quad \text{such that} \quad \|Ax - b\|_2 \leq \varepsilon$$

where $W = \text{diag}(w)$ for a weight vector

$$w_i = \frac{1}{|x_i^{\text{old}}| + \delta} \tag{2.7.5}$$

and x^{old} is a guess of the solution which is updated after every ℓ_1 solve. If $x^{\text{old}} = x_0^\star$ where $x_0^\star$ is the solution to the ℓ_0 problem, then clearly in the neighborhood of $x_0^\star$ the weighted norm approaches the value of the ℓ_0 quasi-norm as $\delta \to 0$. This scaled problem can always be solved by existing ℓ_1

algorithms since even if the algorithm doesn't explicitly handle scalings, a change of variables can be made.

The idea of reweighting is to better approximate the $\|x\|_0$ norm. We follow the approach of [CWB08], which motivates this approach using a log-based approximation mentioned in Fazel's thesis in 2002 [Faz02] if not earlier. The idea is also very similar to the idea of iteratively re-weighted least-squares (IRLS); see [DDFG10] for a modern perspective on this old algorithm. Indeed, IRLS has moved beyond its original use as just a robust replacement to least-squares, and is now used to approximate ℓ_1 norms or even ℓ_p ($p < 1$) norms.

There are variants of this idea in the literature; see [CWB08] and [Cev09] for further references. We briefly note FOCUSS [GGB95] which is a special form of IRLS, and the work of Chartrand and collaborators on both algorithms and theoretical signal recovery guarantees for ℓ_p minimization for $p < 1$ (see, e.g., [Cha07, CY08]). In general, the weight vectors can be chosen in many ways and regularization parameters δ can be chose in many fashions; e.g.,

$$w_i = \frac{1}{|x_i^{\text{old}}|^2 + \delta^2}$$

and the reweighted problems can be of ℓ_1 or of ℓ_2 type. It is our opinion that the differences among these schemes is probably not very great, and so we have restricted attention to weights of the form (2.7.5). Other variants, such as iterative-support detection [WY10], set the weights of all nonzero coefficients to zero. This approach would be interesting to try in future work.

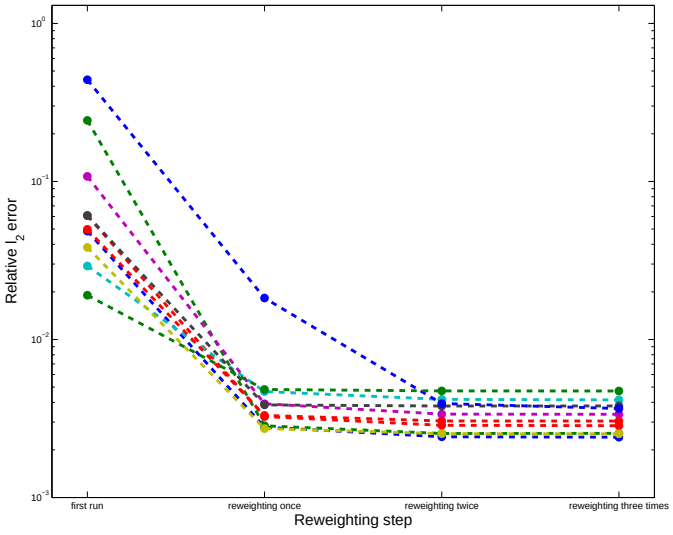


Figure 2.47: Early reweighting results from 2008, using l1-Magic [CR07b]. Shown are 10 independent runs (each color is an independent run). Signal is a pulse with 2.21 GHz carrier frequency. Reweighting is clearly beneficial.

Log approximation. The approach we take to motivate the weights is to approximate the $\|x\|_0$ by $\log(|x|+\delta)$ for some small δ. See Figure 2.46 for the graph of this log function, and the comparison

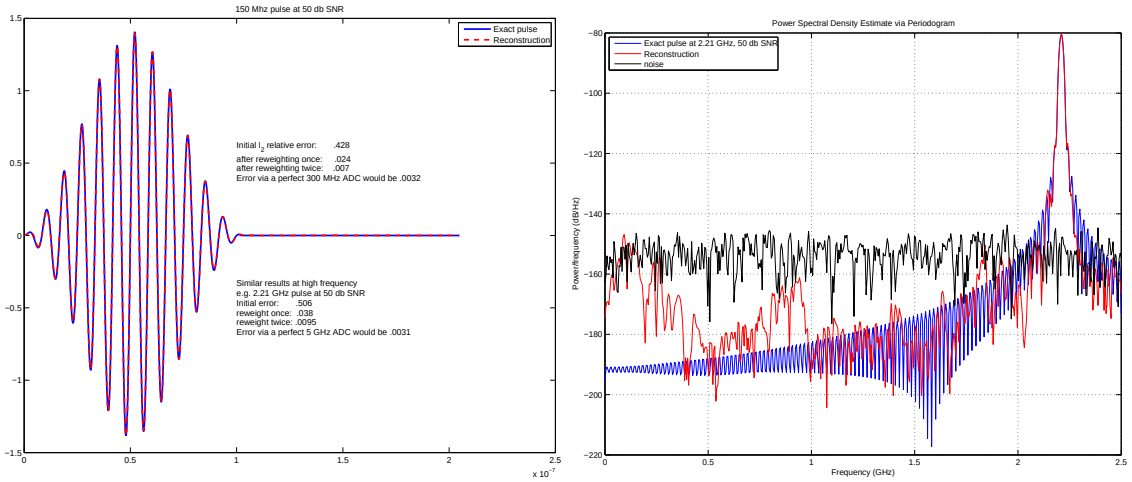


Figure 2.48: Example of a reconstruction that used two rounds of reweighting, which were very beneficial. In the frequency domain, the input noise is shown in black (but not shown in the time domain picture, for clarity). The recovered solution is less noisy than equivalent Nyquist-rate samples.

to the standard ℓ_1 norm (in this 1D graph, just the absolute value function). The log function is not convex, and in fact it is concave if restricted to the positive orthant. The reweighted ℓ_1 approach is a gradient descent algorithm to minimize this concave function; so far, there are no known results about the convergence (to a global minimizer) of this algorithm. However, [Nee09] has basically shown that in terms of recovering sparse vectors, the reweighting procedure will not hurt.

Consider $f(u) = \log(u + \delta)$ for $u > 0$ (for example, introduce the extra variable u and constraint $|x| \leq u$). This is concave, and hence it is below its tangent; we say that the tangent line "majorizes" f. Thus taking a gradient descent step will result in a new point with smaller value f. This is part of the MM framework [HL04] (which in this case, stands for "majorize minimize"). The gradient of f a point u_0 is $\nabla f = 1/(u_0 + \delta)$, so the linearization at u_0 is

$$f(u_0) + u/(u_0 + \delta)$$

and minimizing this (ignoring the constant terms) is equivalent to solving a reweighted ℓ_1 problem with weights $w = 1/(u_0 + \delta)$ with the constraints $|x| \leq u$.

Because this is non-convex, the starting point is very crucial, and we have observed in practice (see §2.5.2.3) that using a DCT basis, prior to using a Gabor frame, to get a suitable starting value generally works better. Furthermore, unlike the log model above, we adaptively choose δ every round instead of fixing it. This works better in practice but makes convergence results even trickier.

The regularization parameter $\delta > 0$ is necessary since otherwise the weights become infinite for any zero coefficients. If $\delta \approx 0$, then the weights might be huge for nearly zero coefficients, and the method will be less stable and take a long time to converge. As δ becomes larger, fewer iterations are needed. If c is the smallest magnitude of the true solution's nonzero entries, then the idea is to

pick $\delta \approx c$, though of course c is unknown.

The RMPI experiments have used two rules for picking δ; both rely on a new percentage parameter ρ which is typically set to .95 or in the range $[.8, .99]$. Let $u = |x|$ be the current estimate of $|x|$ from the previous iteration, and assume wlog that u is sorted from large to small. Let D be the dimension of x. The rules are

1. Let K be the nearest integer to ρD. Pick δ such that $u_K \leq \delta < u_{K+1}$. The idea is that we want a sparse approximation, and we use ρD so that this scales automatically if the size of the dictionary is changed.

2. For any δ, let T_δ be the set of entries of u that have magnitude greater than δ. Pick δ to be the smallest number such that
$$\sum_{i \in T_\delta} |x_i|^2 \geq \rho \sum_{i=1}^{D} |x_i|^2.$$
The idea is that we choose the threshold so that ρ percent of the signal's energy is explained by coefficients of this size or greater. This is computationally efficient to carry out by using a cumulative sum of sorted coefficients.

Both rules depend on ρ, and both perform about the same. Recent experiments have been using rule 2. There are some signals that are well recovered only when ρ is tweaked, but this value of ρ is signal dependent, so we keep ρ relatively constant to avoid overfitting to specific signals.

The reweighting steps are performed for a fixed number of iterations (usually between 2 and 5) or until the new iterates are sufficiently close to previous iterates.

Improvements. The basic premise of recent model-based CS theory [BCDH10] is that realistic signals have much more structure than just sparsity in a basis. For example, the coefficients of a multi-level transform of a signal will not be independent. If energy is present in a small scale coefficient, then some energy is likely present in a "parent" coefficient.

The works of [BCDH10, Cev09, JMOB10] have proposed using special thresholding or proximal method algorithms to enforce special structure. The structure is usually of a hierarchical form. For example, in a multi-scale transform, a tree can be drawn of parent-child windows. The structure assumption is that if a child coefficient is nonzero, then so is the parent node. Adding this additional assumption usually improves recovery, since it is restricting the feasible set to be smaller. The RIP bound can be changed to an RIP over the model set, which should produce better RIP constants.

To stay within the ℓ_1 framework, we suggest using a tree structure of multi-level transforms and then choosing weights to reflect this. For example, instead of reweighting each entry by

$$w_i = \frac{1}{u_i + \delta}$$

we reweight by

$$w_i = \frac{1}{\tilde{u}_i + \delta}, \quad \tilde{u}_i \triangleq \max_{(j \text{ is a child of } i)} u_j$$

(with the convention that a node is a child of itself). A similar idea has been independently suggested in [DWB08]. Unfortunately we have not yet tested these ideas, since it adds an extra layer of complexity and parameter choice on top of the already numerous parameters for selecting a dictionary.

Even if not using a tree structure to determine weights, the multiscale nature of the Gabor dictionary should be taken into account. For example, the coefficients at each level of the transform should have a specific regularization parameter δ_{level}. This has not been implemented yet and is expected to help. In general, we note that there is much existing knowledge about *denoising* via wavelet thresholding, and it is our opinion that *reweighting* techniques can benefit. For example, cycle spinning (averaging over various shifts to make the transform translation invariant) and group thresholding (using local neighborhood information to inform thresholding/reweighting) are well-known in the denoising literature (e.g., [DJ94, Nas08]), and it seems likely that these techniques would improve reweighting performance.

2.7.2.4 Debiasing

Solving an ℓ_1 minimization with inequality constraints problem generally introduces bias into the solution. Consider the generic problem

$$\min_x \|x\|_1 \quad \text{such that} \quad \|Ax - b\|_2 \leq \varepsilon.$$

By duality, if A has full row-rank, there is some scalar $\lambda > 0$ such that this is equivalent to solving

$$\min_x \lambda\|x\|_1 + \frac{1}{2}\|Ax - b\|_2.$$

To analyze this, take a very easy case: $A = I$. Let x_0 be the true signal we wish to estimate, and $b = x_0 + z$ be the measurements where z is any zero-mean random variable. Since A is the identity, the solution is known in closed-form (this is just the proximity operator of ℓ_1, discussed in detail in Chapter 4):

$$\hat{x} = \text{sgn}(b) \cdot [\,|b| - \lambda]_+$$

where $[\cdot]_+$ means the positive part. For the positive entries $i \in \hat{T_+}$ of $\hat{x}$, this is

$$\hat{x}_i = b_i - \lambda = (x_0 + z)_i - \lambda.$$

Regardless of the dimensions of the problem, these entries are biased by the amount λ.

For general matrices A, the estimates are biased as well. For a sparse recovered vector $\hat{x}$, the fix is quite simple. Set $\hat{T}$ to be the support of $\hat{x}$, and then solve the over-determined least-squares problem

$$\min_{x:x|_{\hat{T}^c}=0} \|Ax - b\|_2$$

to find the best vector, with support fixed to $\hat{T}$, that explains the data. If the support $\hat{T}$ is equal to (or an unbiased estimator of) the true support T of the original signal x_0, then since the least squares problem is unbiased, this procedure removes the bias from the solution.

Unfortunately, when x_0 is not exactly sparse but rather just compressible, the support $\hat{T}$ is generally large, and even if it is smaller than N so that the least-squares problem has a unique solution, the debiased version fails to estimate the original signal well. For recovering radar pulses in the Gabor frame, debiasing was tried many times, but it never helped, and thus it is not used. As suggested by the $A = I$ example, coefficients are biased by an amount unrelated to their magnitude, so that small coefficients have more bias relative to their size, yet clearly small coefficients are very sensitive to errors in determining $\hat{T}$. We leave this issue for further study.

Fortunately, empirical evidence is that reweighted techniques remove bias, and since reweighting is used heavily in the reconstructions, we do not worry much about bias. One simple reweighting technique, called the *adaptive LASSO* [Zou06] in the statistics community, has been analyzed. In particular, [Zou06] proved a conjecture from 2001 that the LASSO can be inconsistent, and proved a celebrated result that the adaptive LASSO is consistent. The adaptive LASSO is basically one step of reweighted ℓ_1 minimization. The weights are chosen inversely proportional to some power of the absolute value of coefficients of any consistent estimator (such as the least-squares estimator); it is closely related to the non-negative garotte introduced in 1995 [Bre95], as well as more recent two-step procedures such as [KXAH10]. Intuitively, reweighted ℓ_1 should perform even better. The fact that the adaptive LASSO is consistent does not prove it is unbiased, but suggests that reweighting is useful.

2.7.2.5 Non-linearity correction

Although there is current work such as [Blu11] aimed at extending compressed sensing to work with non-linear observations, the theory and algorithms for *directly* handling non-linearities is in its infancy. In this section we review an indirect method of partially compensating for non-linearities in the data collection.

So far, the working assumption has been that data are collected

$$b = \Phi x$$

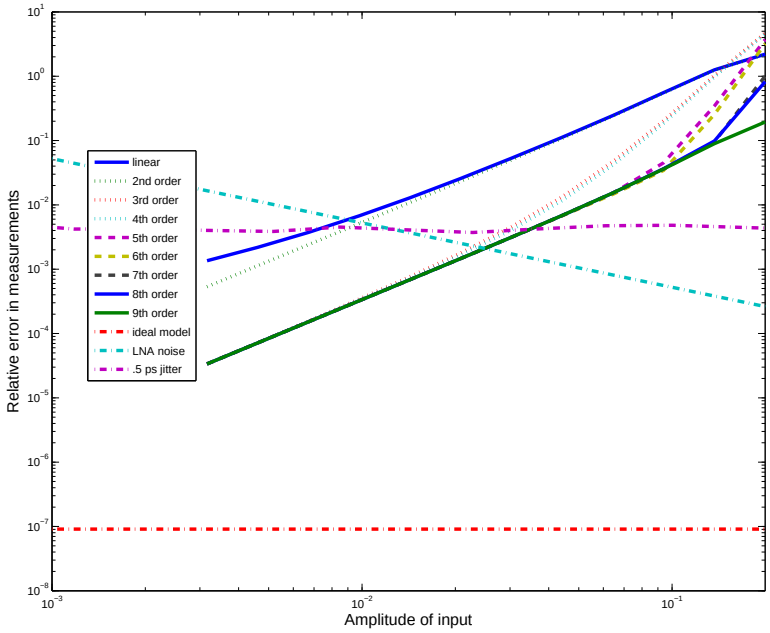


Figure 2.49: Using a non-linear Simulink model, trying to compensate for known non-linearity with a polynomial model by adjusting the predictions from just Φx to $\Phi p(x)$; note that we should not expect perfect agreement since $p(x)$ is calculated on the Nyquist grid, whereas non-linearities will induce harmonics outside this band. Input was 700 MHz tone. Also shown are errors due to jitter and noise (and the ideal case), for comparison. The higher terms of the polynomial only affect large amplitudes, as expected.

which we call the linear model (for conciseness, we don't write down the noise term). In reality, the circuit is non-linear, so a more appropriate model is

$$b = b_0 + \Phi_1 x + \Phi_2 x^2 + \dots.$$

For example, the non-linearities in the mixer and integrator will make such a model. This model is difficult to deal with since the matrices Φ_i are unknown and would be difficult to measure via calibration.

Fortunately, the dominant non-linearity is in the LNA front-end; since this occurs before the mixing, the LNA portion of the system is approximately separable over time in the incoming signal, like a diagonal matrix. This leads to the following model:

$$b = \Phi p(x), \quad p(x) = a_0 + a_1 x + a_2 x^2 + a_2 x^3 \dots \tag{2.7.6}$$

which is much more tractable since the polynomial now has scalar coefficients. Like many RF designs, the dominant terms in the polynomial are a_1 and a_3. For small amplitudes, $p(x) \approx a_1 x$ so the effects of higher-order terms are not significant. But the growth of x^3 outpaces that of x so at some point the third-order term takes effect, and this is the effective upper limit on the system. The point at which this happens can be measured via several standard conventions, such as IIP3. See

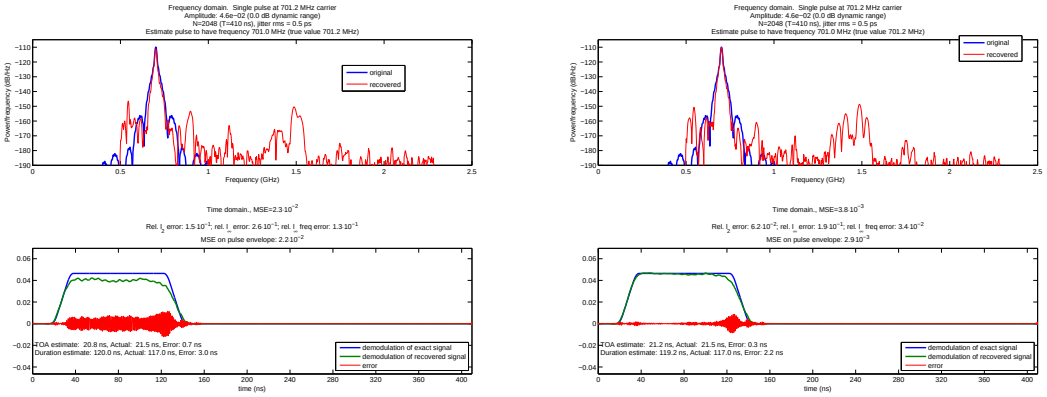


Figure 2.50: (Left) Before non-linearity correction. Notice the gain compression. (Right) After non-linearity correction. See also Figure 2.51.

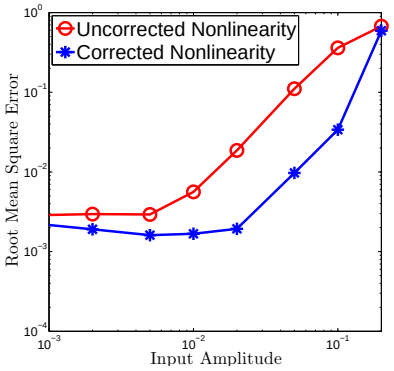


Figure 2.51: Error of reconstruction as a function of input amplitude A, with and without non-linearity correction. Input was a single signal. At very large amplitudes, no post-processing technique can help.

Figure 2.49 for effects of various terms of the polynomial as a function of amplitude.

For now, assume p is known (that is, its coefficients and degree). Is it possible to correct for this effect in post-processing? The answer is yes; see Figure 2.50. However, the technique we present works poorly on high frequency signals. We hypothesize that this is due to the discrete representation of the signal in post-processing. Dealing with a vector x of length n (where Φ has n columns) implicitly assumes that x is band-limited to maximum frequency $f_s/2$. However, non-linearities introduce harmonics, so a high frequency signal with large amplitude will have a component past $f_s/2$. When working with the discrete representation, this is aliased into lower frequencies, and is harder to correct for. It is possible that this problem can be fixed by allowing longer discrete vectors, so that frequencies can go up to f_s or even higher.

We introduce the basic technique. This is a classic example of what applied mathematicians and physicists do best: linearization.

Start with the assumption that $b = \Phi x$ and solve the standard ℓ_1 minimization problem to find

a guess x_1. Now, linearize p about x_1:

$$p(x_1 + h) = p(x_1) + p'(x_1)h + \mathcal{O}(h^2)$$

and update the model to be

$$b = \Phi p(x) + (\Phi p'(x))h, \quad \text{i.e.} \quad \tilde{b} = b - \Phi p(x),\ \tilde{\Phi} = \Phi p'(x).$$

This model is linear in h, so the ℓ_1 minimization with $\tilde{b}$ and $\tilde{\Phi}$ is solved for h. Then the variable is updated

$$x_2 \leftarrow x_1 + h$$

and the process is repeated by linearizing p around x_2, etc.

This technique was tested on synthetic data from Simulink simulations, where we know exactly the non-linearity that Simulink uses (these values taken from SPICE simulations), and it works well on low and medium frequencies. The entire polynomial, up to 9th order, is known; to test robustness, the correction was tried using different number of terms. In reality, it would only be reasonable to correct for the 2nd- and 3rd-order terms, since high order terms would be impossible to estimate reliably. Reconstructions used either 3 or 6 terms in the polynomial. These Simulink tests do show some robustness, since even though the 9th-order polynomial is fully known, the measurements are not exactly in the form of (2.7.6) since the transconductor and common-gate polynomials are applied *after* mixing, so the polynomial is not separable.

A final note: if the harmonics fall out-of-band and are caught by a low-pass filter, this appears to solve the non-linearity problem. However, the effect of the non-linearity is still there, and it is now more appropriate to look at the intermodulation effect: when two tones are passed in the receiver, the non-linearity causes harmonics at the sum and difference of the two frequencies. Furthermore, in addition to distortions of large signals, non-linearities cause "desensitization" and blocking of small input signals that are concurrent. If there is significant noise after this point, there is not much that post-processing can do to recover the small signal. Hence, ultimately non-linearities remain a circuit design issue, and can only be partially corrected in software.

Unknown coefficients. The case when p is not known is difficult, and was not tested numerically. The SPICE model can help give estimates of p, but these will never match the true values of the circuit. We did not yet test the robustness of the non-linearity correction to inaccurately estimated coefficients.

It may be possible to use the EM or MM algorithm [DLR77, HL04] and iterate between non-linearity correction for a fixed p, and then updating the estimate of p. This would be similar to the

dictionary learning techniques that have been used quite successfully in image processing. We leave this issue for further study. Another variation left for further study is the possibility of allowing for a polynomial with *matrix* coefficients Φ_i, since this would model non-linearities other than just the LNA non-linearities, but would require a novel calibration scheme.

2.7.2.6 Windowing

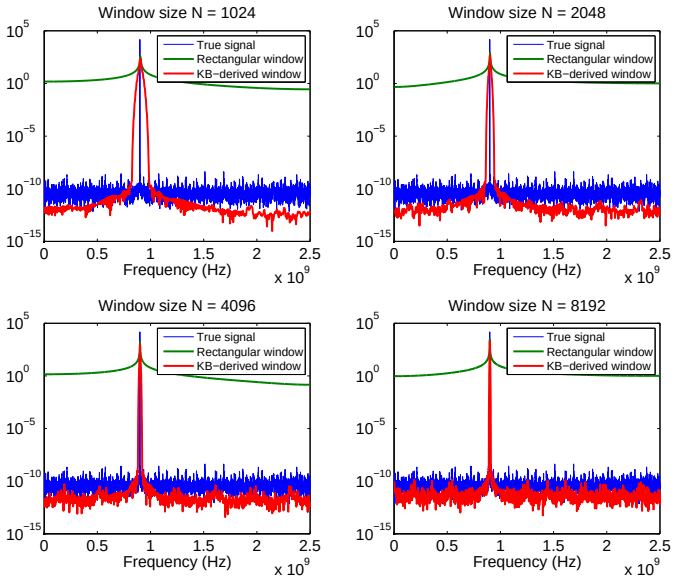


Figure 2.52: The importance of windowing. For an arbitrary off-grid tone, a finite set of samples will experience spectral spreading; this is due to the Fourier transform of the boxcar function, which has very slow decay due to the sharp corners. Using a smooth window introduces some loss of time resolution but offers much greater frequency resolution, and is usually worth the loss of time resolution. For any type of window, the larger the number of samples N, the less spectral spreading; this is a reason why fast methods for large-scale optimization, such as those developed in Chapters 3 and 4, are essential in signal processing. This figure shows the same signal but with different lengths N chosen for processing, with small N in the upper left and large N in the lower right.

The simulations and analysis in this chapter assume that a pulse x is located in the middle of a known time interval, so boundary effects are not significant. In a deployed system, obviously this assumption is invalid. The simplest solution is to increase the chunk of time under analysis. This increases the chances that a signal is in the middle of the interval, and it also reduces the frequency spreading that would be induced by windowing a pure tone by a rectangular window. However, due to processing limitations (since reconstruction algorithms are so far more than linear complexity), there is an upper limit on the size N of the discrete signal. We also would like to seamlessly combine signal estimates from different blocks.

The standard solution to this problem is to introduce a smooth window, and then overlap adjacent chunks of size N, e.g., a 50% overlap. There are many choices of windows, each with a unique time-frequency resolution trade off, and this has been well-studied for many years. It is also convenient to choose windows that form a partition-of-unity (also known as the Princen-Bradley condition), such

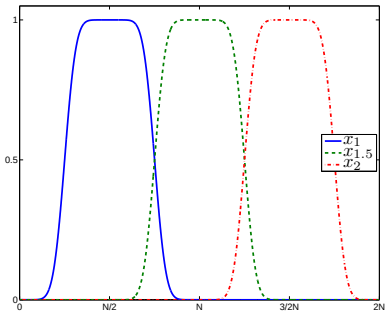


Figure 2.53: Overlapping windows which form a partition of unity, for blocks of length N. Using a Kaiser-Bessel derived window, with parameter $\alpha = 8$

as the windows previously mentioned in the Gabor dictionary section; see Figure 2.53. This is the technique of a modified discrete cosine transform (MDCT) used for MP3 compression, to give just one example. For a compressed sensing device that takes non-uniform time-samples, like the NUS, windowing can be applied "retroactively". For NUS, samples are taken

$$b = Sx$$

where S is a $M \times N$ matrix that is effectively diagonal, since it is just a $N \times N$ identity matrix with $N - M$ of the rows removed. Suppose w is a window and let $W = \mathrm{diag}(w)$ be the matrix version. We desire

$$\tilde{b} = SWx.$$

Because S is so simple and $SS^T = I_M$, it effectively commutes with W; specifically,

$$\tilde{b} = (SWS^T)Sx = \tilde{W}b, \quad \tilde{W} = (SWS^T)$$

so it can be recovered *ex post facto*.

The case of RMPI is unfortunately more difficult, since the measurements Φ have a much more complicated structure than that of S. Again, we seek

$$\tilde{b} = \Phi W x.$$

This time, it is not possible to write $\tilde{b}$ in terms of $b = \Phi x$.

One possibility to overcome this is an iterative scheme, which we suggest here and may be novel. Start with an estimate x_1 of x, obtained via ℓ_1 minimization without windowing. Then update the observations

$$b_1 \leftarrow b - \Phi((W - I)x_1).$$

If the estimate x_1 is very accurate, then $b_1 \simeq \tilde{b}$. Form a new estimate x_2 by re-solving the ℓ_1

minimization using b_1. This process is then repeated until convergence. For best results, each stage would use many consecutive windows to produce estimates of x in a particular (probably overlapped) time block, and then combine them to make a global estimate of x before proceeding with the next iteration. This idea has not yet been tested numerically.

The authors of [TLD+10] suggest adding a time-varying channel and doubling the number of channels in order to window the signal in hardware, before the integration step. This is a serious design change, and reflects the difficulty of the windowing problem. It seems likely that a software solution can obviate such a step; as shown by matched filter, it does not appear to be a fundamental problem with the manner in which the data were collected.

It is also possible that when restricted to radar pulse, and hence using the Gabor dictionary, that windowing of overlapping blocks is not necessary, due to the the localized time support of the Gabor atoms themselves. More study is clearly needed.

2.7.2.7 Further improvements

These improvements have not yet been tried due to time constraints, but we expect that some of them may have beneficial effect.

Modeling noise. The experiments presented have always used the constraints

$$\|\Phi x - b\|_2 \leq \varepsilon.$$

However, most thermal noise is injected before measurements. A complete noise model (ignoring jitter) is

$$b = \Phi(x + z_1) + z_2$$

where z_1 likely dominates z_2, and both are iid. This can be modeled using a single z_2 term which incorporates Φz_1. If both z_1 and z_2 are i.i.d. Gaussian with variances σ_1^2 and σ_2^2, respectively, then modeling both together is equivalent to a new z_2 term with covariance matrix

$$\Sigma = \sigma_1^2 \Phi\Phi^T + \sigma_2^2 I. \tag{2.7.7}$$

With no other priors on the signal, the maximum likelihood estimate of x is

$$\hat{x} = \underset{x}{\operatorname{argmin}} \langle \Phi x - b, \Sigma^{-1}(\Phi x - b)\rangle.$$

This inner product can be concisely written as $\|\Phi x - b\|^2_{\Sigma^{-1}}$.

Of course we have *a priori* knowledge that the signal x is sparse, so we use ℓ_1 minimization. It is then only natural to solve the ℓ_1 problem with a weighted ℓ_2 norm for the constraints. For the

analysis problem, this is

$$\underset{x}{\text{minimize}} \ \|\mathcal{D}^* x\|_1 \quad \text{such that} \quad \|\Phi x - b\|_{\Sigma^{-1}} \leq \varepsilon.$$

This can be solved by existing ℓ_1 algorithms without modification by updating $\Phi \leftarrow \Sigma^{-1/2}\Phi$ and $b \leftarrow \Sigma^{-1/2} b$. The estimate of Σ^{-1} should be performed by pseudo-inverse, or σ^2 should be artificially enlarged, in order to keep it stable.

A final remark is that the parameter ε is an issue. In theory, it can be estimated from the chip during a calibration phase when it is known that there is no input. The variance of iid white noise can be estimated with the sample variance; if it is suspected that other factors are involved, then a more robust measure is mean-absolute-deviation. To estimate the covariance matrix of correlated white noise, the situation is more involved, but there are existing techniques. The maximum likelihood estimate is just the sample covariance, but this could be improved by imposing structure on the covariance matrix, e.g., of the form (2.7.7). This can be solved with software such as TFOCS, discussed in Chapter 4. In the simulations in this thesis, ε was either estimated manually or based off of $\|b\|_\infty$. Performance does depend on ε but it is not overly sensitive. The only time an accurate value of ε is crucial is when one is trying to recover two concurrent tones that are widely different in dynamic range.

Jitter. Jitter has been treated like additive white noise, but this is not true at all since it is signal dependent. Preliminary work by Jerry Fudge's team at L-3 attempts to take this into account, but details are not public and it may not be applicable to our design.

Here, we mention another recent technique. Since jitter is part of the sampling, it has a non-linear effect on the measurements. For this reason, the likelihood function is not known in closed form. The authors in [WG11] propose numerical evaluation of the likelihood function via quadrature rules, and then use the EM algorithm ([DLR77]; see also [HL04]) to maximize the likelihood function, and report improvements. Their technique deals with *sampling* jitter. As mentioned earlier, sampling jitter is not a major concern for the system, but the *mixer* jitter is, and perhaps a similar EM algorithm could improve the system with respect to the mixer jitter.

Block norms. The reconstruction algorithm processes one chunk of length N (or time T) at a time. If the input signal x possesses structure that lasts longer than T, then this should be exploited. For example, if x is a pure tone, it will look nearly identical in every window, and hence the multiple observations should be combined in order to improve its estimate. Radar pulses typically repeat many times a second, so they also posses this structure. Of course there is a limit to how much batch processing can be done (imposed by computational resources; and by perhaps requiring a finite time-lag for real-time processing, where future observations are not available).

A simple way to take advantage of this structure is to use a block norm. We write $\|X\|_{(1,p)}$ to denote the sum of the p-norms of the rows of a $N \times L$ matrix:

$$\|X\|_{(1,p)} \triangleq \sum_{\text{rows } i=1,\ldots,N} \Big(\sum_{\text{columns } j=1,\ldots,L} |X_{i,j}|^p \Big)^{1/p}. \tag{2.7.8}$$

This is in contrast to the norm $\|A\|_{q\to p} \triangleq \sup_{z\neq 0} \|Az\|_p / \|z\|_q$. The $\|\cdot\|_{(1,2)}$ norm is also know as the *row-norm* of a matrix.

Instead of minimizing the ℓ_1 norm of a vector x from time period 1, we minimize the $\|\cdot\|_{(1,2)}$ or $\|\cdot\|_{(1,\infty)}$ norm of the matrix X where each column of X is from a different time period; this technique could also be combined with the windowing technique suggested earlier, in which case the columns of X actually come from time periods that might overlap. This is sometimes referred to as "multi-channel" processing ("channels" here are not the same notion as channels of the RMPI). Modern software such as TFOCS (Chapter 4) can handle these norms. This is discussed in the literature; for example, in [BKR09], which refers to the signal model as a "fusion frame".

2.8 Results

Using the Simulink model with non-idealities, it is possible to get an accurate idea of the limits of the system. This section discusses the robustness of the system to non-idealities, and the expected performance in the presence of a realistic level of the non-idealities.

To measure error against a known input x_0, we use the (relative) RMSE criteria already mentioned in (2.4.1)

$$\text{RMSE}(x) \triangleq \frac{\|x - x_0\|_2}{\|x_0\|_2}$$

and the signal-to-noise-and-distortion ratio SNDR (often referred to as just SNR)

$$\text{SNDR}_{\text{dB}} = 10^{-20\,\text{RMSE}(x)} \tag{2.8.1}$$

as well as the effective number of bits

$$\text{ENOB} = \frac{\text{SNDR}_{\text{dB}} - 1.76}{6.02}. \tag{2.8.2}$$

The maximum allowed input amplitude of the RMPI is 0.1 V, and this is referred to as "full-scale". The terminology dBFS reflects size relative to full-scale. For example, if a pulse is -20 dBFS, this means the amplitude is .01 V.

2.8.1 Non-idealities

The RMPI is affected by many non-idealities. A non-ideality is anything that our model does not account for. We do not discuss errors in creating the model, since those have been discussed in the calibration section. The non-idealities we consider are

- Thermal noise.
- Clock jitter at the mixers and ADCs.
- Quantization.
- Cross-talk between signal paths.
- Clipping.
- Non-linearities. (This was also previously discussed in §2.7.2.5.)

Of these, the thermal noise, jitter, and non-linearities have the biggest effect. Quantization, cross-talk, and clipping are generally not a dominant source of error. The section concludes with §2.8.1.6, which discusses all the non-idealities together.

2.8.1.1 Noise

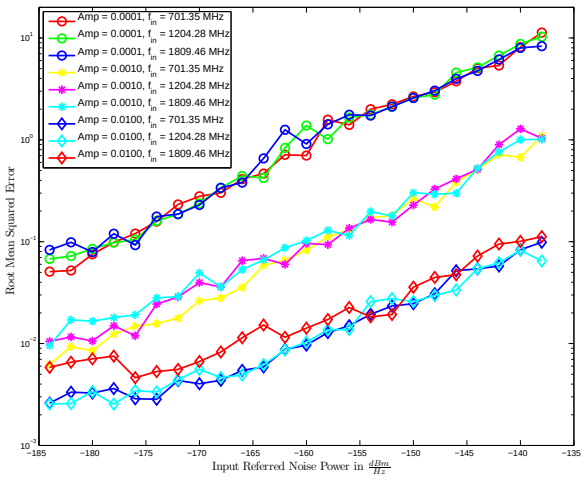


Figure 2.54: Error of reconstruction as a function of input-referred noise power η_{in}, shown for various input amplitudes A (circles, plus, and diamonds) and carrier frequencies f_{in}

Thermal noise is a major issue. Because it has a constant value regardless of the input, it affects small inputs and limits the lower range of signals that the system can reliably distinguish. In the Simulink model, three of the blocks have thermal noise, but by far the dominant source is at the LNA. For the LNA block, we treat input referred noise, which takes into account noise from different

stages of the block. In Simulink, this is modeled as iid Gaussian noise using an appropriately fine time-scale.

Figure 2.54 shows the results of different levels of LNA noise. Noise power η_{in} is measured in dBm (dB per relative to milliwatts), for 1 Hz bandwidth. The total noise power across a different bandwidth B is $\eta_{\text{in}} + 10\log_{10}(B)$.

The figure plots signals for 3 levels of amplitude and 3 frequencies (low, medium, and high). The frequency dependence is quite small. As a function of noise power, the trend in RMSE is linear. A pessimistic estimate of the noise in the CMOS chip is -164 dBm. At this level, we have just barely acceptable performance for a single input (RMSE$\simeq 0.4$) at $A = 10^{-4}$ V; the max amplitude possible is about 10^{-1} V. For an exact RMSE cutoff of .33, the single-pulse dynamic range of the system is 55 dB or equivalently 9 ENOB.

2.8.1.2 Jitter

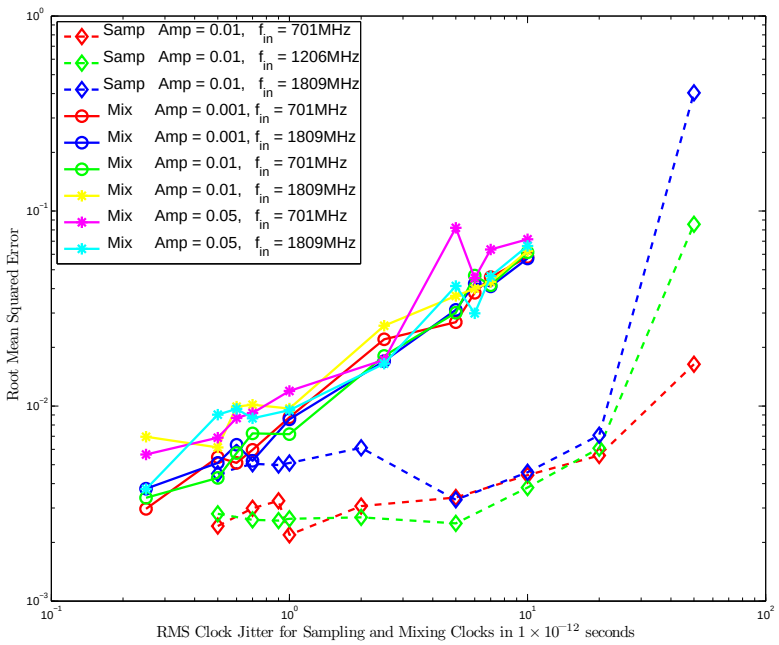


Figure 2.55: Error of reconstruction as a function of rms jitter σ. The diamonds (dashed lines) show error for *sampling* jitter at the ADC, while the circles and pluses show error for jitter at the *mixer*.

The RMPI runs off a clock which operates at f_s, and this clock determines the output of the shift registers that define the PRBS. The actual rise and fall time of the clock signal do not exactly lie on the f_s grid but are jittered slightly forward or backward in time. Let e_n be the error in time between $n\Delta T$ (when the clock *should* change) and when the clock actually changes. Similarly to thermal noise, this can be viewed as a stochastic quantity. The most commonly used figure-of-merit is "rms jitter" σ which is the standard deviation of e_n, since typically $\mathbb{E}\, e_n = 0$. In simulation, e_n is treated as an iid Gaussian variable.

The distinguishing feature of jitter is that it induces an error proportional to both the amplitude and frequency of the signal. If an input is $x(t) = A\sin(\omega t)$, then the difference between the ideal sample at $x(t)$ and a jittered sample at $x(t+e)$ is roughly $A\omega e$. Hence jitter is a non-linear effect, and is more significant at high frequencies.

Jitter also plays a role in the sampling of the ADCs. This sampling jitter is less important because it is after the integration and the signal is dominated by low frequency components; in contrast, mixing jitter is more significant because (a) it affects the PRBS which is very high frequency, so a small time error corresponds to a larger amplitude error, and (b) due to the high rate, the jitter affects every Nyquist sample, whereas sampling jitter only affects every $N_{\text{int}} = 100$ Nyquist samples. In the version 2 design, the CMOS mixer has 0.5 ps *mixing* jitter. The 12-bit ADC samplers have 2 ps sampling jitter but their bandwidth is only 300 MHz.

Figure 2.55 plots the RMSE of the reconstruction for pulses of low, medium, and high frequencies, as a function of σ. The ADC sampling jitter only has an effect for $\sigma > 20$ ps, so the this is not expected to be a limiting factor. For any $\sigma > .1$ ps, the mixing jitter has a negative effect on the reconstruction. Because of this, the RMPI design was thoroughly engineered to keep jitter as minimal as possible, and the rms jitter of 0.5 ps is state-of-the-art. At this value, jitter has a negative effect, but thermal noise and non-linearities have a larger negative effect, as will be discussed in §2.8.1.6.

2.8.1.3 Quantization

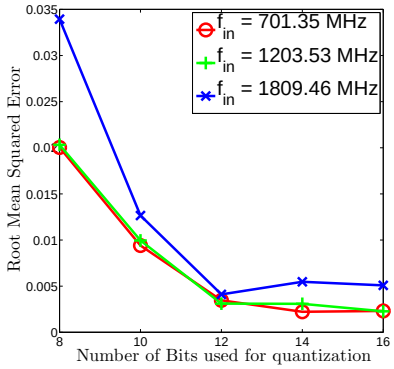


Figure 2.56: Error of reconstruction versus number of bits of quantization for amplitude $A = 0.001V$ (-40 dBFS)

The ADC samplers have finite resolution and quantize the output to only B bits. An ADC can be configured to an expected range of input which should closely match the maximum output of the system. Having set the scale of the ADC to match the largest expected values of our system, Figure 2.56 shows the effect of this quantization on a small signal input. For $B \geq 12$ bits, there is no appreciable effect of the quantization. For even smaller signals, it may be necessary to use 14 bits, assuming that thermal noise doesn't dominate the signal.

If the system is upgraded to handle even more dynamic range, than 16 or more bits may be

necessary, which should not be an issue since there are commercial 50 MHz ADC with large ENOB. An alternative would be to implement a dynamic system that changes the scaling to the ADC, but this may be difficult. The version 2 RMPI uses 12 bit ADCs, but these are off-chip and so could be upgraded easily.

2.8.1.4 Cross-talk

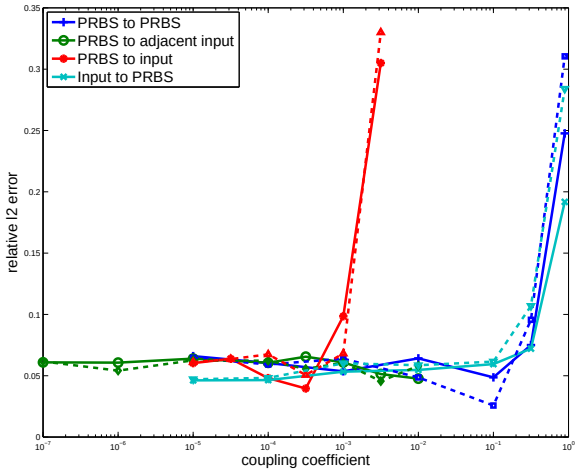


Figure 2.57: Results of the cross-talk simulations. Solid line is a 700 MHz input; dashed line is a 2.1 GHz input. Both inputs were pulses 100 ns long and amplitude .01 V (-20 dBFS).

Cross-talk is the interference caused by signals when electronic components are in close proximity. The dominant sources of cross talk are shown in Figure 2.58. To minimize cross-talk, the chip was designed with several safeguards: each channel's analog and digital supplies are not connected to one another on chip; clock distribution supplies and references are completely independent; the analog and digital grounds are isolated from one another; extensive use is made of guard rings around all blocks, subblocks, and channels; and each block is triple-welled. The worst-case coupling levels reported in the literature for mixers of this class is 10^{-2} [CMA09]. According to the simulations in Figure 2.57, this is right at the borderline level for inducing failure due to PRBS-to-input cross-talk. However, this worst-case result was in a design that didn't use triple wells or substantial guard ring structures, and we expect our coupling levels to be smaller. Since this was a worst-case value, it is not expected that cross-talk is a significant limitation in the design.

2.8.1.5 Clipping

Large amplitude signals run the risk of railing the integrator, since the integration is only designed for a limited voltage range, $\pm v_{max}$. To analyze how likely clipping is to occur, we assume Φ is the Bernoulli ± 1 model, and work in discrete space. What is the worst-type of input signal? It is a signal x_n that exactly mimics the discrete chipping sequence $c_n \subset \{-1, 1\}^N$. However this is extremely

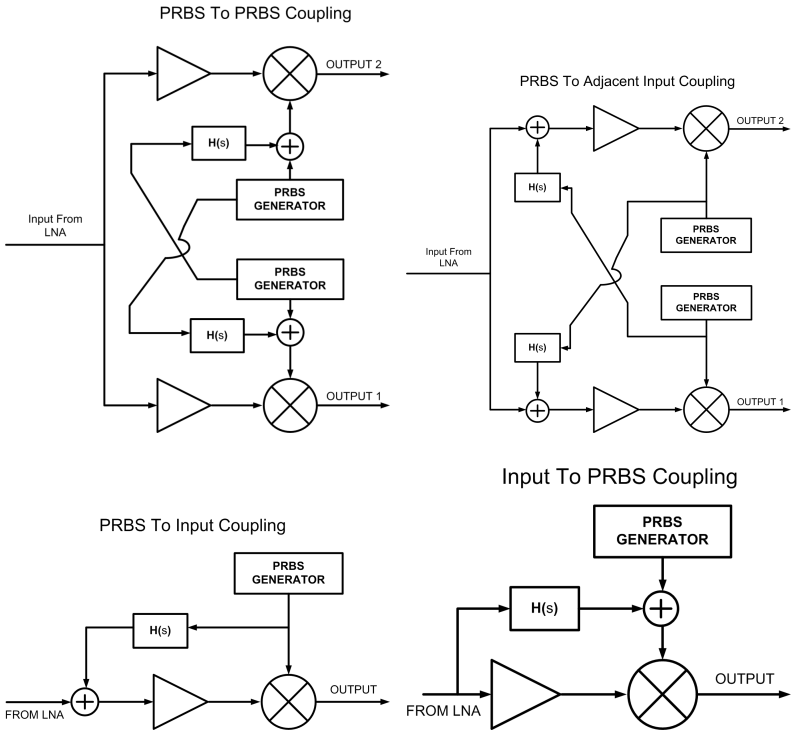


Figure 2.58: The types of cross-talk mechanisms considered in the testing show in Figure 2.57. The $H(s)$ block represents the unwanted cross-talk that might occur.

unlikely. If the chipping sequence c_n is random, then a "typical worst-case" input is actually any arbitrary signal with amplitude A_{max} and a reasonably slow frequency($f \lesssim 1$ GHz, so that the signal is approximately constant over a Nyquist sample), so without loss of generality we assume x_n is a DC signal with amplitude A_{max}. Let $S_k = \sum_{n=1}^{k} c_n$. Thus the question of clipping is: over a fixed interval $T = N\Delta T$, what is the probability that at some (possibly intermediate) time $k \leq N$, $|S_k| \geq \frac{v_{max} f_s}{A_{max}} \equiv v$? We'll break this question into three parts.

First, consider the sum at the *end* ($k = N$) of the time in question, $p \equiv \mathbb{P}(S_N \geq v)$. The sum S_N is just a shifted and scaled sum of Bernoulli random variables, so S_N is a shifted and scaled binomial random variable, which has a known CDF. For large N, this is excellently approximated by the CDF of a Gaussian variable with $\mu = N/2$ and $\sigma = \sqrt{N/4}$.

At first glance, finding $\mathbb{P}(S_k \geq v,\, k \leq N)$ appears difficult. Fortunately there is a well-known clever trick (thanks to Yaniv Plan for pointing this out). Consider an intermediate sum $S_k = v$, and look at the set of all possible random walks of length $N - k$ that start at this point. Then the expected value at time N of these extensions is still v since a random walk has mean 0. In fact, it has median 0 as well, and half of the extensions will end up with value greater than v, and half will end up with value less than v (note: this reasoning requires that $N - k$ is odd, so that none of the extensions can end up with value exactly v).

Thus if we record the number of sums S_N which exceed v, this accounts for half of the sums

which had reached v at an earlier time. Thus $\mathbb{P}(S_k \geq v,\, k \leq N) = 2p$. This reasoning is a little imprecise since it actually depends on whether v (assumed to be an integer for the moment) has the same parity as N, but for large N and values of v that are not too close to n, the error is small. See Figure 2.59 for experimental evidence.

As the final step we account for negative amplitudes, so $p_{clip} = \mathbb{P}(|S_k| \geq v,\, k \leq n) = 4p$.

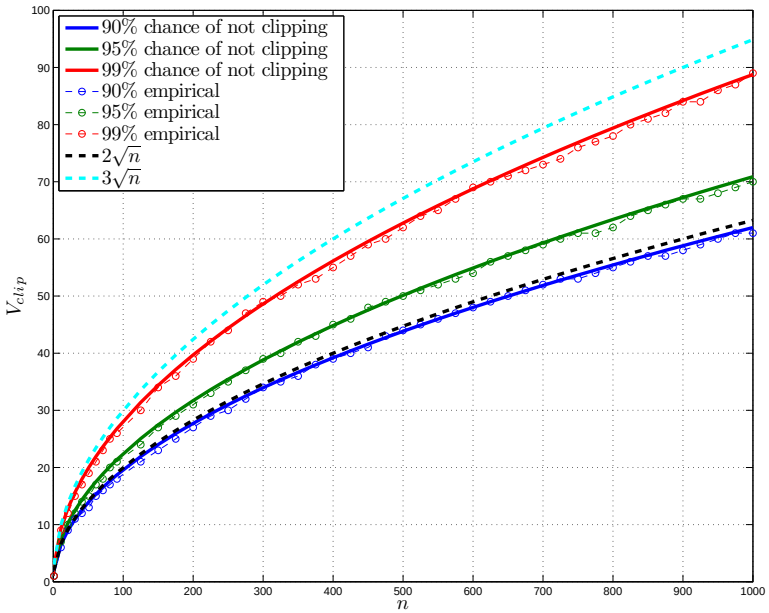


Figure 2.59: Simulation to verify the chance of a random walk clipping. The solid lines are the predictions from using the inverse CDF of a normal distribution, and the circles are the empirical values from 10,000 Monte Carlo simulations, which show excellent agreement, thus validation the $2p$ argument. Furthermore, the lines are bounded between $2\sqrt{n}$ and $3\sqrt{n}$, which provides a useful heuristic. To use the chart: suppose $v_{clip} = 70$ and we want to be 99% sure that a maximum amplitude signal will not clip. Then we must reset the integrator every $n = 600$ Nyquist units. (Note: for a practical calculation, v_{clip} needs to be properly scaled; see (2.8.3).)

Let us calculate the scaling for the version 2 RMPI chip. Suppose the input is a low-frequency sine wave $x(t)$ of amplitude A. If the mixing sequence c is random then almost all inputs look the same, and what really matters is the absolute integral $\int |x(t)|$, so this sine wave input can be modeled as a DC input with amplitude $2A/\pi$. The max acceptable amplitude input is 0.1 V_{pp}, and the LNA gain is 18 dB (this is actually variable; it can be set from 12 to 18 dB), so the input voltage is $v_{in} = 10^{18/20} 0.1 \times 2/\pi \simeq .8 \times (2/\pi)$ V. A transconductor of 20 mS (mS $= 10^{-3}$ Amps/volt) converts the voltage signal into a current signal. The current is attenuated by a factor of 2 in the mixing stage, and the path is differential so we (temporarily) attenuate by another factor of 2 to account for this. Thus the input current to the integrator is $I_{in} = (2/\pi)(.8 \text{ V})(20 \cdot 10^{-3} \text{ Amps/V})/4 = (2/\pi)4 \cdot 10^{-3}$ Amps. To find the output voltage of the integrator (which is approximately just a capacitor), we use the capacitor equation:

$$v_{out}(\tau) = \frac{1}{C} \int_0^\tau I_{in}(t) dt$$

where the capacitance is $C = 40$ pF. For a DC input and discrete chipping sequence, the integral becomes a sum, with time step $\Delta T = \frac{1}{5\text{ GHz}}$. The clipping voltage is $v_{clip} = 4v_{out}$ because we now combine the two differential paths and also consider peak-to-peak voltage. The supply voltage to the chip is 1.5 V, so the peak-to-peak clipping limit is $v_{clip} = 3$ V. Combining all these scalings, we can write down an equation:

$$\begin{aligned} 3\text{V} = v_{clip} = 4v_{out} &= \frac{4}{C}\sum_{k=0}^{n-1} c_k I_{in} \Delta T \\ &= \left(\sum_{k=0}^{n-1} c_k\right) \frac{4}{40 \cdot 10^{-12}} \frac{1}{5 \cdot 10^9} (2/\pi) 4 \cdot 10^{-3} \\ &= \left(\sum_{k=0}^{n-1} c_k\right) \frac{4}{25\pi} \end{aligned} \tag{2.8.3}$$

which means the system will clip with

$$\left|\sum_{k=0}^{n-1} c_k\right| = \frac{75\pi}{4} = 58.9. \tag{2.8.4}$$

The integration period T_{ADC} is 100 Nyquist samples, so $n = 100$ per sample, and thus it is *possible* to clip in a single integration period, which is catastrophic. However, it is extremely *unlikely* to clip in a single sample (if the assumptions hold; to be more rigorous, we would need to impose a prior on the input signals). Using Figure 2.59, then there's a 99% chance the signal will not clip until $n \gtrsim 420$, and a 90% chance the signal will not clip until $n \gtrsim 850$. Thus with a max amplitude signal, and the highest gain setting of the LNA, clipping should not be an issue if we restrict to 4 to 9 integration windows T_{ADC}. If a signal has half the max amplitude, then we need the random walk to reach 118, which is extremely unlikely to occur if $n \leq 1000$.

If the overall experiment is significantly longer than $1000\Delta T$ (which it usually is, even though we typically analyze chunks of size 1024 or so), then for a pure tone input, the system will eventually rail (in fact, if we run the system forever, this occurs with probability 1), so we would periodically reset the integrators; furthermore, the fact that the PRBS sequence c is in periodic would be significant. Such considerations are beyond the scope of this work. For inputs that are pulses at some repetition rate, the system will periodically have no input, and the natural discharge of the capacitors will prevent railing.

The system can be told to reset at will, but this involves the loss (or at least corruption) of a sample. Since we do not expect railing to be a problem for short signals, we have not simulated the effects of resetting when running worst-case simulations, but this issue would be important for future chips that operate robustly. Possible alternative designs may be beneficial, such as having two samplers that alternate, so each capacitor can be reset every integration period. The Northrop

Grumman InP chip uses a system like this. The downside of this design is slightly more complexity and power consumption.

2.8.1.6 Combining non-idealities

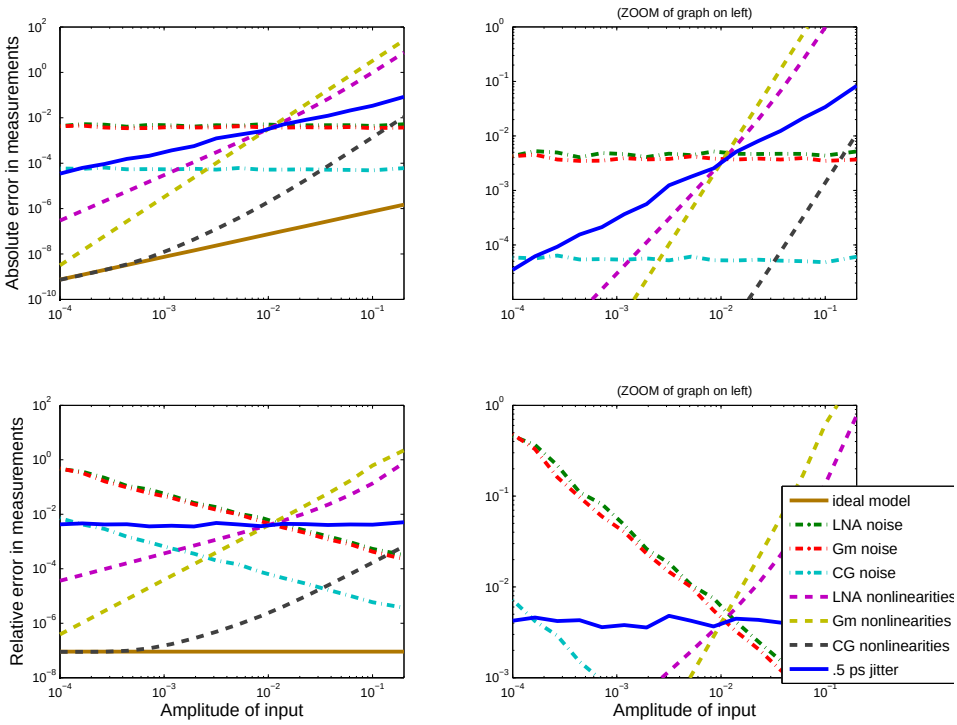


Figure 2.60: Various non-idealities. The four plots show the same data but in different ways: the top row shows error in absolute terms, while the bottom row shows error in relative term. The right column is a zoom of the left column. The "ideal" model uses a Φ matrix generated from a (noiseless) Simulink calibration. The other measurements are from the Simulink model with various non-idealities turned on. This shows at what amplitudes each non-linearity has an effect. The thermal noise has an absolute effect (constant on the top row), while jitter has a constant relative effect (constant on bottom row) since it induces error proportional to the amplitude of the input. See also Figure 2.61.

The major non-idealities affect the system in different ways.

- Thermal noise is constant, so it affects *small* signals.
- Jitter error is proportional to the amplitude of a signal, so it affects *all* signals equally.
- Non-linear effects are only an issue when the amplitude is *large*.

Jitter and non-linearities make two-signal recovery difficult. If one signal is much smaller than the other signal, the smaller signal still sees the large jitter errors induced by the large signal. The small signal is also desensitized or even blocked due to the non-linear effects of the large signal [Raz97]. For this reason, low-jitter and a high linearity range are extremely crucial for good performance.

For the simpler case of just one signal recovery, which of these non-idealities is limiting? To test this, we calibrate an accurate model Φ using Simulink to provide the black-box system S. The

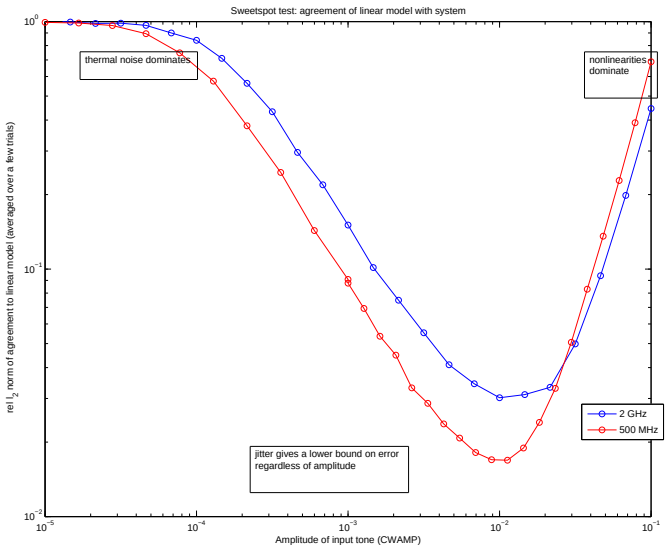


Figure 2.61: "Sweetspot test." The amplitude 10^{-2} (-20 dBFS) has the least measurement error when we include the non-linearities. See also Figure 2.60.

agreement $\|\Phi x - S(x)\|_2 / \|S(x)\|_2$ is very good, typically on the order of 10^{-7} (see the brown line in Figure 2.60). It is not exactly 0 because S is based on ODE solvers, whereas Φx assumes that x is band-limited and is computed as a finite-precision matrix-vector multiply, so both methods have a small amount of error. We now consider $S_{\text{non-ideal}}$, which is the Simulink model using a non-ideality. Figure 2.60 plots

$$\|\Phi x - S_{\text{non-ideal}}(x)\|_2 \quad \text{(top plots)}, \quad \text{and} \quad \|\Phi x - S_{\text{non-ideal}}(x)\|_2 / \|S(x)\|_2 \quad \text{(bottom plots)}.$$

The $S_{\text{non-ideal}}$ model used exactly one type of non-ideality for each line shown in the figure, which gives an idea of the relative importance of each non-ideality.

The top row shows the absolute error, as a function of input amplitude (recall that 0.1 is full-scale input). Thermal noise is independent of amplitude, so it shows up as a flat line on this plot. The thermal noise at the LNA and transconductor (Gm) are about equal; the noise at the common gate filter is not significant. Non-linear effects from the LNA and Gm play a role at inputs of -20 dBFS and higher.

The bottom row shows relative error, and hence the error due to jitter is a flat line. Jitter was tested at several RMS values, and we only show the $\sigma = 0.5$ ps. For larger values of RMS jitter, the jitter will become a dominant source of error, but at this value it is not a limiting factor (though with non-linearity correction, it becomes a slightly limiting factor for a small range of amplitudes). However, since jitter affects two-pulse recovery, there is no "safe" value of σ. The value $\sigma = 0.5$ ps is chosen because it is the lowest practical value that we can design.

If we allow $S_{\text{non-ideal}}$ to use *all* non-idealities at once, then we get the plot in Figure 2.61. The figure shows relative measurement agreement. Two different signals are shown, one at low frequency and one at high frequency, since jitter affects the high frequency signal more. The plot shows that the amplitude 0.01 (which is -20 dBFS) has the least error, since this is large enough to ignore thermal noise but also small enough not to have significant non-linearity problems. We refer to this amplitude as the "sweet spot" of the system. This also indicates that the system has a maximum dynamic range of about 60 dB, since for signals greater than 0 dBFS or smaller than -60 dBFS the measurements have almost 100% relative error, and thus it is unreasonable to expect any recovery algorithm to recover the input. This is not an exact rule, since the reconstruction is non-linear and can denoise in some cases, but it is a reasonable rule-of-thumb.

2.8.2 Simulation results

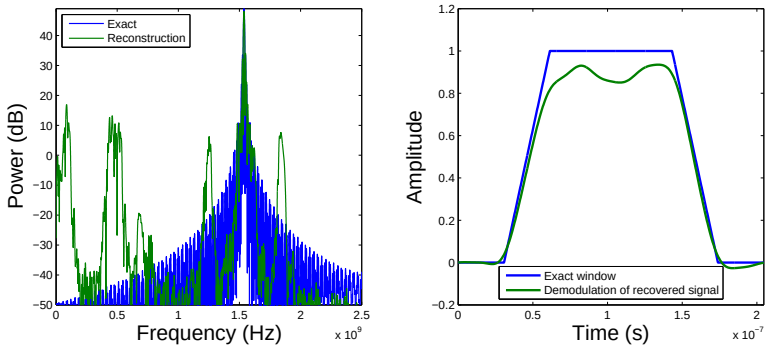


Figure 2.62: Reconstruction using data from SPICE simulation for version 1 design. The matrix Φ was modeled as an ideal single-pole integrator with the pole at 44.05 MHz. Agreement is crude, but not bad considering the non-calibrated Φ matrix. Error in carrier frequency estimation is 0.1 MHz.

The results here are mainly from detailed Simulink simulations, using accurate levels of non-idealities. Less systematic results show that we can recover from SPICE simulations as well.

Figure 2.62 shows a reconstruction from SPICE data from the version 1 model in December 2008. The Φ matrix uses ± 1 entries and is not calibrated, so the performance is quite remarkable given

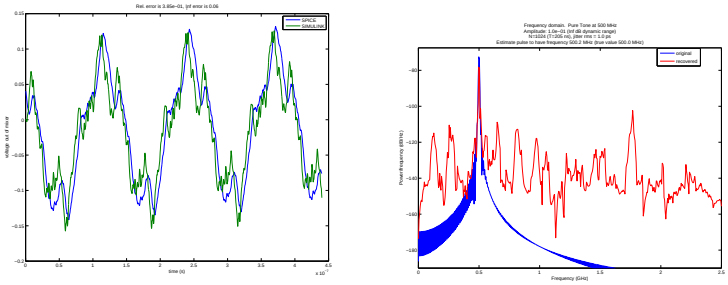


Figure 2.63: May 2010, SPICE simulation, using a Simulink calibrated Φ matrix to recover. Left is the integrator output (before ADC sampling); right is a reconstruction. Input was a pure tone at 500 MHz.

the lack of calibration. Figure 2.63 shows results from the version 2 model. The Φ matrix is from a calibrated Simulink model. Results would be even better if Φ were calibrated from a SPICE model, but this is time-prohibitive.

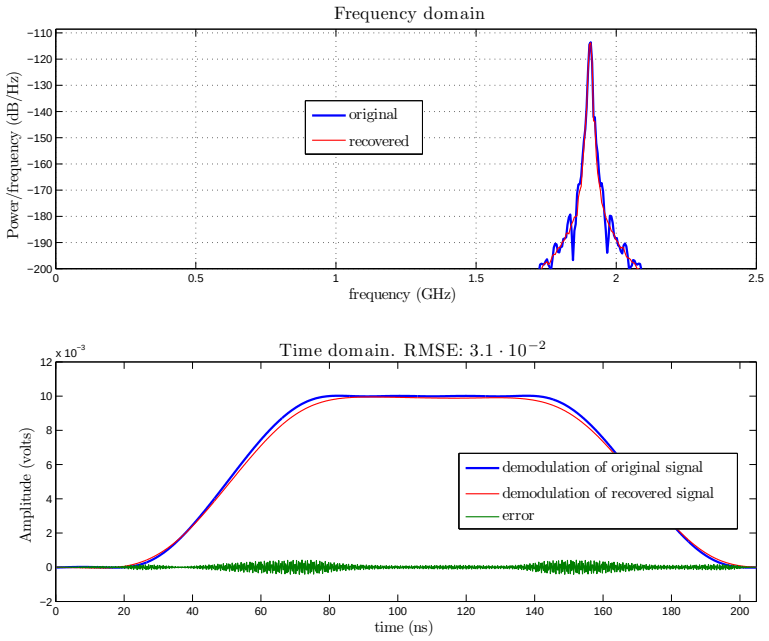


Figure 2.64: An input test signal and its reconstruction. Carrier frequency is $f_{in} = 1907.2$ MHz. Simulation (via Simulink) includes all non-idealities, e.g., jitter (rms .5 ps), noise, non-linearities. Amplitude is 10^{-2} (-20 dBFS), and observation time is $T = 205$ ns ($N = 1024$). The recovered signal estimates the correct frequency to within 0.1 MHz. The relative error $\|\hat{x} - x\|_2/\|x\|_2$ is $3.1 \cdot 10^{-2}$. The time-of-arrival estimate (defined as when the signal first reaches 10% of its max amplitude) was accurate to 1.2 ns, and the duration estimate was accurate to 2.6 ns.

Figure 2.64 shows a typical recovery from noisy Simulink data, using a calibrated Φ. Because of the calibration, the recovery is excellent.

2.8.2.1 Single pulse

To test the recovery of a single pulse, there are three dominant parameters: the amplitude, the frequency, and the pulse duration.

Frequency. The effective instantaneous band-width (EIBW) of the system is a measure of the range of frequencies that can be accurately recovered. Figure 2.65 shows the ENOB of recovery on a 200 ns pulse, as a function of carrier frequency. The EIBW is almost the full 2.5 GHz, though not quite since the pulse window induces spectral spreading, so carriers at 2.45 GHz will cause much of the narrowband content to appear at greater than 2.5 GHz. The hardware is not designed for more then 2.5 GHz, and the recovery algorithms use a Nyquist grid so the post-processing is not suitable for signals past 2.5 GHz either.

The hardware version 2 RMPI has been designed with less bandwidth than in the simulated

versions. In particular, the frequency response is very small below 300 MHz, and a more accurate estimate of the EIBW is 2 to 2.2 GHz.

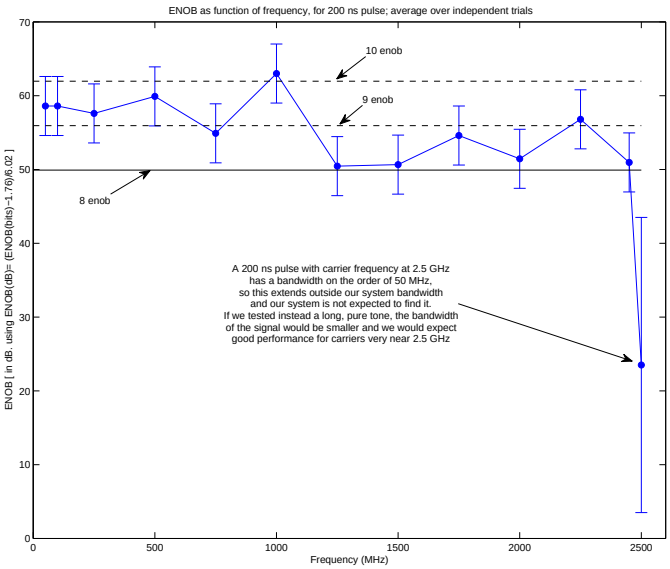


Figure 2.65: ENOB as a function of frequency for 200 ns pulses. For all frequencies, the amplitude is at the sweet-spot amplitude.

Single-pulse dynamic range. To consider the limits of the system, we test the largest and smallest signals it can handle. Recovery is better if the pulses are longer. Figures 2.66 and 2.67 show relative MSE as a function of amplitude and pulse duration, at carrier frequencies 700 MHz and 2100 MHz, respectively. The pulse has a 25 ns rise and fall time, and measurements are taken from a Simulink model with realistic values for all the non-idealities. The recovery software was set to assume a bandwidth from 500 MHz to 2500 MHz (this slightly helps recovery; the restriction to this bandwidth is imposed by adding extra rows to Φ that correspond to entries from the DFT matrix, and adding corresponding 0 samples).

For durations of less than 100 ns, recovery is poor since only a few samples are taken of the system. We focus on 200 ns pulses since this is the shortest reasonable radar pulse our system is designed to recover. At 700 MHz, Figure 2.66 shows that the MSE is less than 10% for amplitudes between 0 and -55 dBFS. Thus using MSE < 0.1 as the cutoff, the dynamic range is 55 dB. Converting from dB to ENOB, this is 8.8 ENOB. For 100 ns pulses, the dynamic range is 50 dB, or about 8.0 ENOB.

At higher frequencies, such as shown in Figure 2.67, results are slightly worse and ENOB is only 8.0 for 200 ns pulses. The frequency- and duration-dependent results may depend slightly on the parameters used for reconstruction, e.g., the parameters of the Gabor dictionary.

Fixing the duration at 200 ns, Figure 2.68 shows the effect of frequency. For each frequency point, the actual frequency is chosen randomly in a small interval, so that on-grid effects are not

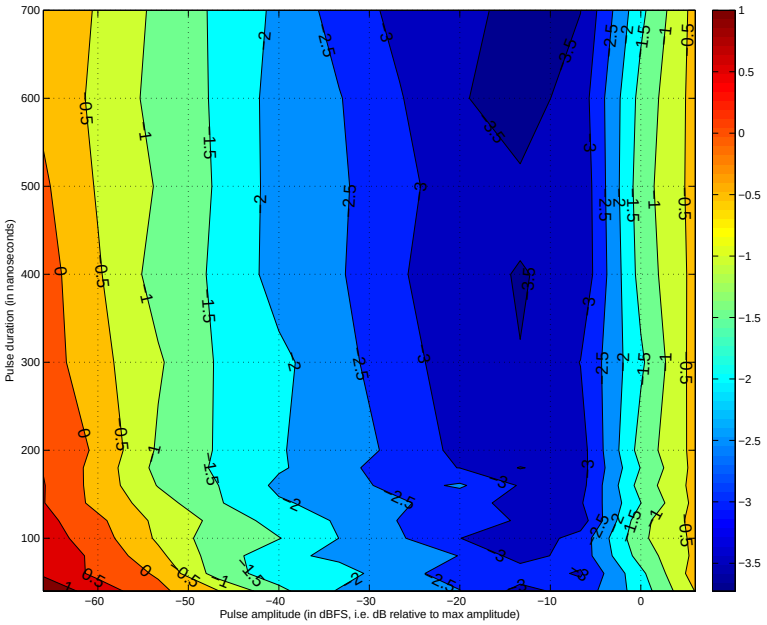


Figure 2.66: Plotting relative MSE (elevation) as function of pulse duration (vertical axis) and amplitude (horizontal axis). Contours shown $\log_{10}$ scale. 701.4 MHz carrier. Results improve for longer duration, as expected.

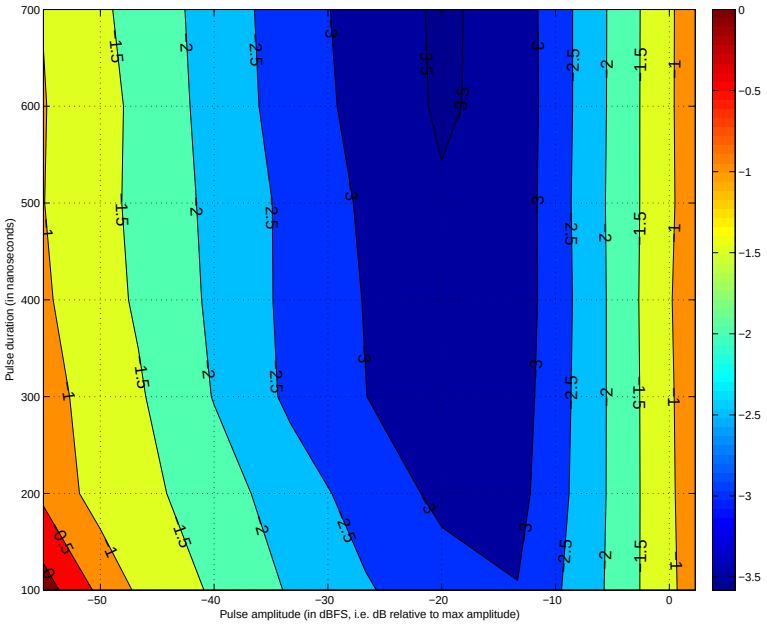


Figure 2.67: Same plot as Figure 2.66 except with 2105.1 MHz carrier instead of 700 MHz carrier (chosen so that it is not on-grid), and showing the median results over 10 independent runs. Results are slightly poorer at this high frequency.

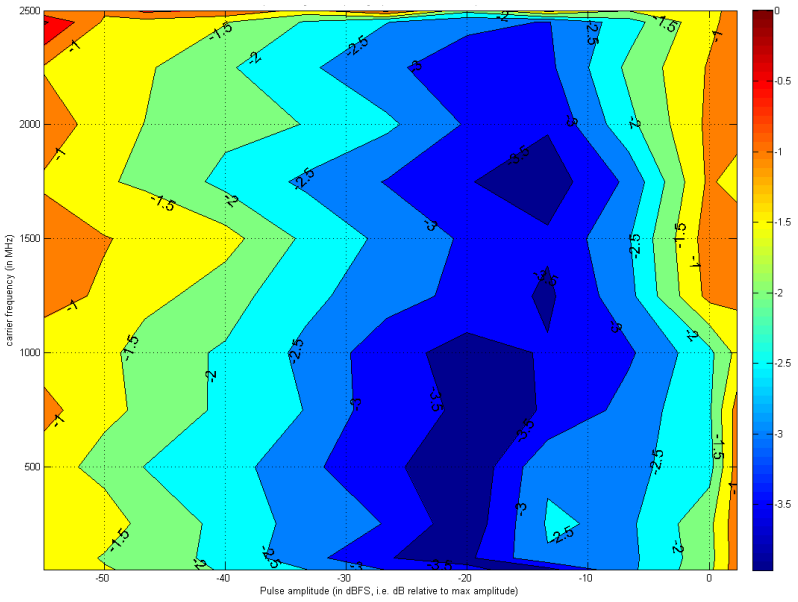


Figure 2.68: From realistic Simulink simulations in spring 2010. Error in relative MSE as a function of frequency. Recovery as a function of frequency and amplitude, for a 200 ns pulse.

significant. Phase is random as well, and pulse rise and fall time are 25 ns. Recovery is generally better at lower frequencies, which is not surprising since the PRBS spectrum is slightly larger at lower frequencies, so low frequency signals are affected less by noise; furthermore, clock jitter at the mixer is more significant for high frequency tones.

2.8.2.2 Two pulses

We consider two types of two-pulse tests: keeping both pulses at the same amplitude, or fixing one amplitude and varying the other pulse amplitude. For both types of tests, the pulses overlap 80% or more.

Equal amplitude. For both pulses with equal amplitudes, the results are quite similar to single-pulse tests. With 200 ns pulses, the relative MSE is less than 10% for amplitudes in the range -6 dBFS to -54 dBFS. The upper limit is slightly worse than the single-pulse case since there is more energy in two pulses, so non-linearity is a larger effect. Using non-linearity correction, the upper limit can be pushed to -3 dBFS, so there is 51 dB dynamic range.

The basic conclusion is that two pulses of the *same* amplitude are not more difficult to recover than a single pulse. In this case, sparsity is not a limiting factor, rather non-idealities are.

The Figure 2.69 shows a typical recovery. This is an extreme case using two very short pulses. The amplitude of both is -20 dBFS which is optimal according to the "sweet spot" calculations in Figure 2.61

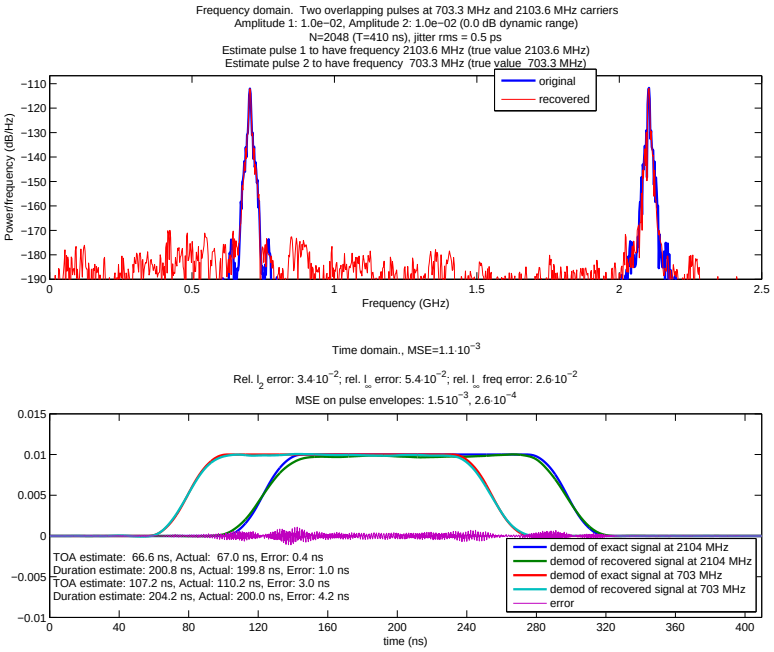


Figure 2.69: Two pulses in realistic simulation, each 80 ns long with 80% overlap. Note that the pulse codewords (carrier frequency, time of arrival, and duration) are estimated very accurately.

Different amplitudes. If two signals overlap in time and have very different amplitudes, recovery of the small tone is quite difficult. Using noisy Simulink simulations, we estimate that the system can recovery long pulses with up to 30 dB difference in amplitude. Figure 2.70 shows an example reconstruction of two pulses with 10 dB difference in amplitude. Clearly, one limiting factor is that large pulses induce much jitter error, which will swamp small signals.

Figure 2.71 shows two concurrent 200 ns pulses that have been recovered from *noiseless* measurements, using a model of the 300 kHz single-pole integrator. The pulses are 40 dB different in amplitude; any greater difference and the small pulse is not recoverable (for this particular instance; with a different input, recovery beyond 40 dB is sometimes possible). Because this is noiseless, it suggest a limit on the recovery procedures of the system. It is possible that better choices of a dictionary or new recovery techniques may improve the dynamic range considerably; for example, slight changes in the Gabor dictionary increase the maximum dynamic range from 30 to 40 dB, so it is reasonable that further improvements may push this to 50 or 60 db. The dynamic range is also affected by variables such as how close the carrier frequency of the two signals are and the chip repetition rate N_{chip}.

Hopefully future results will improve this pulse-on-pulse dynamic range. We note that this problem is quite difficult, and modern radar systems generally ignore any pulses that arrive concurrently. Other issues, such as jitter caused by large signals, are applicable to any ADC system that views two pulses.

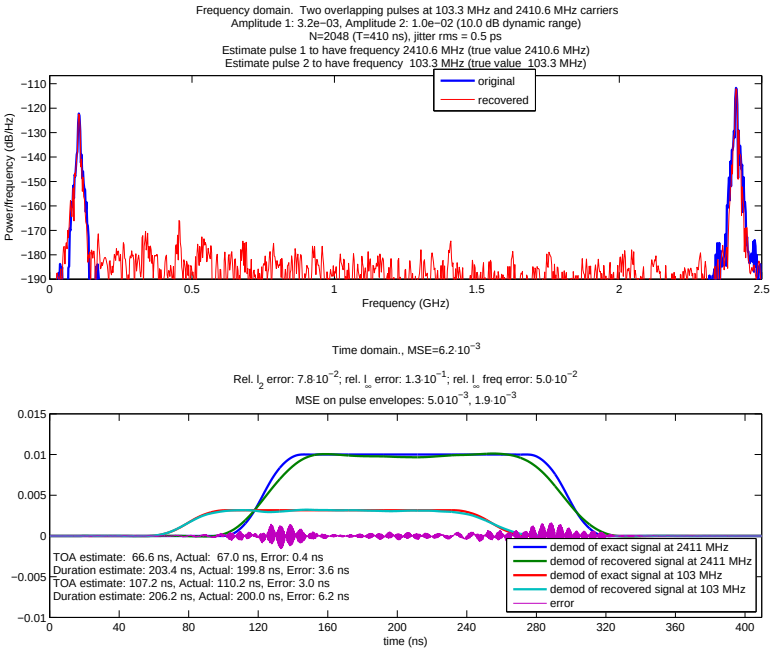


Figure 2.70: Like Figure 2.69 but with pulses of different amplitudes, which is much more difficult

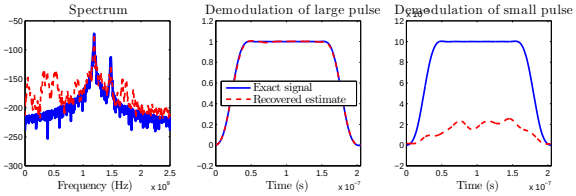


Figure 2.71: Two overlapping 200 ns pulses with 40 dB difference in magnitude. Noiseless setting. With any more dynamic range, recovery is poor. The range is better if we consider longer pulses.

2.8.2.3 Comparison

The single pulse ENOB for the RMPI varies between 8 and 8.8 ENOB. The version in simulation has 2.5 GHz EIBW, and consumes $p = 3.6$ watts including the ADCs (but not including the FPGA or computer). Using Walden's figure-of-merit (FOM) [Wal99], and with 8 ENOB,

$$F = \frac{2^{\text{ENOB}}\text{EIBW}}{p} = 1.76 \cdot 10^{11}.$$

The best ADCs with 1 GHz EIBW have only 7 ENOB and consume about 3 watts, according to [LRRB05]. Thus the RMPI has improved bandwidth, improved ENOB, and about the same power consumption. Many aspects of the RMPI are not optimized for maximizing the FOM, since we placed extra value on reconfigurability. For example, the shift-registers for the PRBS are programmable, but require high amounts of power. In short, the RMPI is not a tweaked system but rather a prototype, and therefore the fact that it already exceeds the state-of-the-art is encouraging.

A channelized conventional receiver can have a higher FOM than the RMPI, but the FOM is

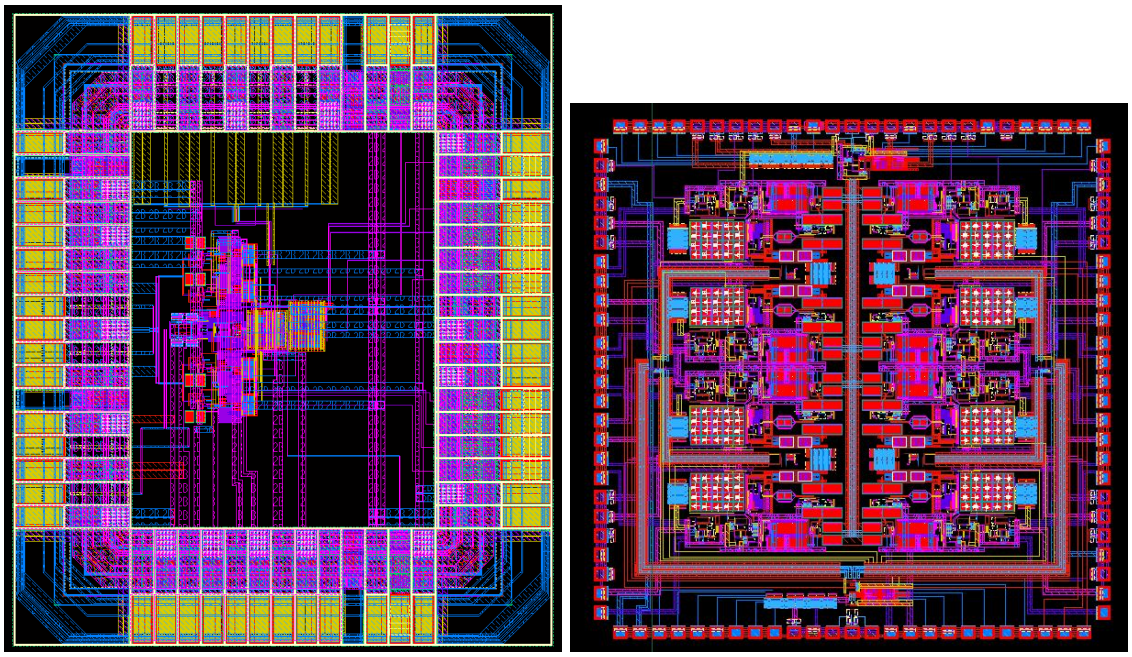


Figure 2.72: Layouts of the version 1 (left) and version 2 (right) Caltech CMOS RMPI chips

not the only meaningful measure. For example, a channelized receiver covering 2.5 GHz EIBW will consume 50 to 200 watts, in contrast to the RMPI which consumes 3.6 watts for the same EIBW. For a low-power environment, if computations can be done off-line, then the RMPI system is quite advantageous.

2.8.3 Hardware

2.8.3.1 NG InP version

The Northrop Grumman indium-phosphide chip has been returned from fabrication and, as of March 2011, is in the process of being tested and calibrated. This design uses 4 channels, each sampling at 100 MHz, as opposed to the 8 channel 50 MHz design of the Caltech CMOS design. The chief designer was Dr. Eric Nakamura from Northrop Grumman Space Technologies. The Caltech team served to consult on their design, and shared hardware ideas as well as higher-level ideas. In particular, we ran an analysis of their initial integrator filters and made improvements. We also found that their LFSR PRBS sequences were under-performing, which motivated using Gold codes to get better statistical properties in the PRBS. Another test (see §2.5.2.3) found that the Gold codes needed to be of a minimal length in order to guarantee good reconstructions.

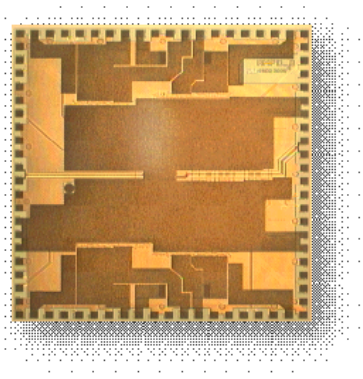


Figure 2.73: Die photo of the NG InP 4-channel RMPI

2.8.3.2 Version 1

The Caltech MICS lab, run by Professor Azita Emami and with head designer Juhwan Yoo, designed the Caltech CMOS chip. From 2008 to 2009, a "version 1" chip was designed and fabricated (see Figures 2.72 and 1.1), arriving back for testing in Spring 2009. Unfortunately, the IBM design toolkit failed to catch a short circuit in the bypass capacitors, and the chip was manufactured with a short. To get around this, each chip was manually cut to stop the short, but this prevented the circuit from working exactly as intended (the bias points were affected) and required large amounts of power to run the circuit (5 watts); overheating was a major issue, and this changed the chips' performance significantly.

This issue prevented detailed performance analysis (although some components, like the LNA and the digital logic system, were verified as working correctly). Without the shorts, the expected power draw was 700 mW, the dynamic range of the chip itself was 45 to 50 dB, operating from 0.1 to 3 GHz. The die area of the chip was 2 mm $\times$ 2 mm. Signals were successfully run through the analog signal path, but at low gain, and no reconstructions were made.

Nonetheless, the chip was still useful. The data acquisition hardware was built, including the chip's PC board and the data-acquisition FPGA. The design and failures of the chip informed the design of the "version 2" chip. Furthermore, using a detailed SPICE simulation, it was demonstrated that the version 1 chip correctly recovers a pulse; see Figure 2.62.

2.8.3.3 Version 2

The second chip (see Figure 2.72) was designed in 2009 and 2010, and arrived back from fabrication in fall 2010. All major components were upgraded. One of the biggest changes was the design of the integrating filter. Simulations predicted that a second-order filter would work, as long as the second pole was not too close to the first, allowing a few hundred MHz of first-order behavior. Allowing this second pole means that the first pole can be situated much closer to DC, and this has an enormous

beneficial effect. The version 1 integrator's first pole was at 44 MHz, while the version 2 integrator's first pole is 300 KHz; see Table 2.4.

Like version 1, the chip was fabricated in IBM's 90 nm CMOS9SF `06_02_00_LB` process. It consumes 830 mW of power (see Table 2.5 for a breakdown), and has a bandwidth of 2.5 GHz. It has a full integrated RF and baseband signal path, so it functions as both a receiver and an ADC. All controls are fully digitally programmable, so the chip sequence can be controlled by the FPGA, which is in turn controlled by a standard computer. Each channel's supply domain is completely isolated. An extra ninth "test" channel was built into the chip, with internal nodes routed out to pads to allow testing. All block biases included programmable power up and down circuitry, allowing for considerable testing capabilities including aggressor/victim cross-talk tests. The test channel also incorporated a replica clock distribution, and digital shift register nodes routed out to pads; this allows characterization of noise and jitter.

The version 2 hardware is currently being calibrated, so the RMPI has not yet been used to recover real pulses, but this is only a few days or weeks away, and will be reported in upcoming papers.

Below we describe some specific details. This section was written primarily by Juhwan Yoo and Azita Emami. Dynamic range is described as a function of the measurements, and is not the same as the dynamic range achieved via reconstruction. Detailed power consumption figures are in Table 2.5.

Due to the large bandwidth requirement of the RMPI receiver, the noise floor is relatively high, and this effect cannot be mitigated due to the fundamental nature of its source, thus limiting receiver sensitivity. With respect to non-linearity, the RMPI is similarly constrained by the required speed of operation due to the fact that the process technology that is required to achieve the necessary broadband operation places a hard constraint on the supply voltages which can be used to power the receiver. Thus, as in the case of the version 1 design, when using a voltage-mode signal propagation methodology, nonlinearity plays a significant role due to an analog design's fundamental reliance on operating receivers around linearized operating points. These two constraints limited the A2I version 1 design to approximately 40–45 dB of dynamic range when operating the signal path with 1.8 V and 2.5 V supplies.

In the version 2 design, in order to mitigate the effects of non-linearity induced by large voltage amplitude operation, a current domain approach is used. Most of the RF signal processing path is designed to handle very high currents while maintaining extremely low-impedance at each node, thus limiting the voltage swing at each node. The current is then converted back to a voltage signal in the integrating stage using transimpedance amplifier-based integrators with very high-loop-gain feedback. The high-loop gain, along with the use of current-mode propagation of the RF signal, mitigated the effects of non-linearity. This allowed the achievement of dynamic range of approximately 55 dB.

Component	Power consumption per units	Quantity	Total Power
LNA/Balun	24 mW	1	24 mW
Transconductor (Gm)	18 mW	8	144 mW
Common Gate TIA	3 mW	8	24 mW
Integrator	12 mW	8	96 mW
Mixer	0.3 mW	8	2.4 mW
Output buffers	30 mW	8	240 mW
Shift registers	30 mW	8	240 mW
Clock distribution	NA	NA	60 mW
ADC (12 bit AD9235)	350 mW (worst-case conditions)	8	2.8 W
Total w/o ADCs			830 mW
Total w/ ADCs			3.63 W

Table 2.5: Power consumption at 1.5 V

The clock distribution is a vital part of the RMPI system. This is due to the fact that it is used to clock the high-speed shift registers which store the PRBS sequence with which the input signal is mixed. The Simulink simulations conducted indicate that the system dynamic range is particularly sensitive to the jitter performance of the system clock. Thus the clock distribution uses a low amplitude sinusoid as opposed to an inverter-based clock-tree, and then performs a CML-to-CMOS level conversion locally at each channel thus achieving very minimal duty-cycle distortion as well as low jitter. The analog distribution is designed to utilize current-mode open drain transmitters and transimpedance amplifier-based receivers with 50 Ω matched transmission lines. Several repeaters are inserted to maintain signal integrity and a H-tree type symmetric clock distribution is used.

2.9 Recommendations

We summarize the important findings; for future directions, see Chapter 5.

Integrator. The integrator should have a very low first pole, at least under 50 MHz, but preferably under 1 MHz, as we have done with the version 2 design. The outer poles are less important, and we set ours at 300 MHz, but it may be possible to bring this down even lower to about 100 MHz.

Channels. We have found that more channels, and using stagger, is beneficial. But too many channels complicates the design and reconstruction (more lost samples due to boundary effects). The NG design uses 4, and the Caltech design uses 8. We recommend between 8 and 12 channels as a good trade off.

Chip sequence. This is quite important. Many errors can be traced to chip sequences with short repetition times N_{chip}. It appears that periods greater than about 60 work, and ours is set

at length 128 to be conservative. However, it may be wise to move this even higher, e.g., 256 or 512. One issue which is not fully determined is how N_{chip} and the integrator period N_{int} should be related; specifically, should they be co-prime? It seems that co-prime numbers will give the best reconstruction problems. The difficulty with this is that the Φ matrix is not fully characterized until it has $N_{\text{min}} = N_{\text{chip}}N_{\text{int}}$ columns, since there are no repetitions; hence, calibration is much slower. For the current design, $N_{\text{chip}} = 128$ and $N_{\text{int}} = 100$ so the system is fully characterized by $N = 3200$ columns, which is reasonable. We strongly advise against setting one of N_{chip} or N_{int} to be a multiple of the other.

Reset. Reset is not too much of an issue according to the analysis. However, a design that alternates between integrators and allows a reset every period may be worth the power, especially for a robust real-time system. In our academic setting, this appears to be unnecessary.

Mixer. Mixer jitter is significant. It took much redesign to get the design to have less than 0.5 ps rms jitter. We advise other systems to also reduce rms jitter to under 1 ps, since otherwise it will kill dynamic range when a large and small signal are both present.

ADC. If the ADCs have more than about 12 bits, there is no noticeable effect for even small signals, assuming that the ADCs are calibrated to have full range with large signals. The RMPI uses 12-bit ADC but these can be switched out for 14-bit ADCs if necessary. We recommend implementing a sample-and-hold block (on-chip) before the ADC (off-chip) so that the ADC does not need a highly accurate clock.

Recovery. The matched filter techniques are quite promising, since they are fast and only attempt to estimate a few parameters. For recovery of single tones, they are recommended. For complex signal environments, we recommend using reweighted ℓ_1 analysis with a very redundant Gabor dictionary.

Chapter 3

NESTA

This chapter, with the exception of §3.7, was jointly written with Jérôme Bobin and Emmanuel J. Candès, and is published in "NESTA: A Fast and Accurate First-Order Method for Sparse Recovery", *SIAM Journal on Imaging Sciences*, **4**, 1, January 2011 [BBC11].

This chapter applies a smoothing technique and an accelerated first-order algorithm, both from Nesterov [Nes05], and demonstrates that this approach is ideally suited for solving large-scale compressed sensing reconstruction problems as 1) it is computationally efficient, 2) it is accurate and returns solutions with several correct digits, 3) it is flexible and amenable to many kinds of reconstruction problems, and 4) it is robust in the sense that its excellent performance across a wide range of problems does not depend on the fine tuning of several parameters. Comprehensive numerical experiments on realistic signals exhibiting a large dynamic range show that this algorithm compares favorably with recently proposed state-of-the-art methods. We also apply the algorithm to solve other problems for which there are fewer alternatives, such as total-variation minimization, and convex programs seeking to minimize the ℓ_1 norm of Wx under constraints, in which W is not diagonal. The code is available online as a free package in the MATLAB language.

3.1 Introduction

Compressed sensing (CS) [CRT06, CT06, Don06] is a novel sampling theory, which is based on the discovery that one can exploit sparsity or compressibility when acquiring signals of general interest. In a nutshell, compressed sensing designs nonadaptive sampling techniques that condense the information in a compressible signal into a small amount of data. There are some indications that because of the significant reduction in the number of measurements needed to recover a signal accurately, engineers are changing the way they think about signal acquisition in areas ranging from analog-to-digital conversion [Hea07], digital optics, magnetic resonance imaging [LDP07], seismics [LH07], and astronomy [BSO08].

In this field, a signal $x^0 \in \mathbb{R}^n$ is acquired by collecting data of the form

$$b = Ax^0 + z,$$

where x^0 is the signal of interest (or its coefficient sequence in a representation where it is assumed to be fairly sparse), A is a known $m \times n$ "sampling" matrix, and z is a noise term. In compressed sensing and elsewhere, a standard approach attempts to reconstruct x^0 by solving

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & \|b - Ax\|_{\ell_2} \leq \epsilon, \end{array} \tag{3.1.1}$$

where ϵ^2 is an estimated upper bound on the noise power. The choice of the regularizing function f depends on prior assumptions about the signal x^0 of interest: if x^0 is (approximately) sparse, an appropriate convex function is the ℓ_1 norm (as advocated by the CS theory); if x^0 is a piecewise constant object, the total-variation norm provides accurate recovery results, and so on.

Solving large-scale problems such as (3.1.1) (think of x^0 as having millions of entries, as in mega-pixel images) is challenging. Although one cannot review the vast literature on this subject, the majority of the algorithms that have been proposed are unable to solve these problems *accurately* with low computational complexity. On the one hand, standard second-order methods such as interior-point methods [CR07b, KKB07, SK02] are accurate but problematic, for they need to solve large systems of linear equations to compute the Newton steps. On the other hand, inspired by iterative thresholding ideas [DDM04, FN03, CW05], we have now available a great number of first-order methods, see [WNF09, COS09, HYZ07, HYZ08] and the many earlier references therein, which may be faster but not necessarily accurate. Indeed, these methods are shown to converge slowly, and typically need a very large number of iterations when high accuracy is required.

We would like to pause on the demand for high accuracy since this is the main motivation of the present chapter. While in some applications, one may be content with one or two digits of accuracy, there are situations in which this is simply unacceptable. Imagine that the matrix A models a device giving information about the signal x^0, such as an analog-to-digital converter, for example. Here, the ability to detect and recover low-power signals that are barely above the noise floor, and possibly further obscured by large interferers, is critical to many applications. In mathematical terms, one could have a superposition of high power signals corresponding to components $x^0[i]$ of x^0 with magnitude of order 1, and low power signals with amplitudes as far as 100 dB down, corresponding to components with magnitude about 10^{-5}. In this regime of high-dynamic range, very high accuracy is required. In the example above, one would need at least five digits of precision as otherwise, the low power signals would go undetected.

Another motivation is solving (3.1.1) accurately when the signal x^0 is not exactly sparse, but

rather approximately sparse, as in the case of real-world compressible signals. Since exactly sparse signals are rarely found in applications—while compressible signals are ubiquitous—it is important to have an accurate first-order method to handle realistic signals.

3.1.1 Contributions

A few years ago, Nesterov [Nes05] published a seminal paper which couples smoothing techniques with an improved gradient method to derive first-order methods[1] which achieve a convergence rate he had proved to be optimal [Nes83] two decades earlier. As a consequence of this breakthrough, a few recent works have followed up with improved techniques in signal or image processing: in [BT09] a fast iterative thresholding algorithm was introduced to solve a class of composite functions that include (QP_λ), and [DHJJ10, WBFA09, Auj09] focused on total-variation minimization problems. In truth, these novel algorithms demonstrate great promise; they are fast, accurate, and robust in the sense that their performance does not depend on the fine tuning of various controlling parameters.

This chapter also focuses on applying some of Nesterov's works [Nes05] discussed just above, and proposes an algorithm—or, better said, a class of algorithms—for solving recovery problems from incomplete measurements. We refer to this algorithm as NESTA—a shorthand for Nesterov's algorithm—to acknowledge the fact that it is based on his method. The main purpose and the contribution of this chapter consist in showing that NESTA obeys the following desirable properties:

1. *Speed:* we will emphasize the case where A^*A is an orthogonal projector. Though being a special case, it is widespread in compressed sensing applications. In that case, NESTA is an iterative algorithm where each iteration is decomposed into three steps, each involving only a few matrix-vector operations. This, together with the accelerated convergence rate of Nesterov's algorithm [Nes05, BT09], makes NESTA a method of choice for solving large-scale problems. Furthermore, NESTA's convergence is mainly driven by a heuristically chosen single smoothing parameter μ introduced in Section 3.2.1. One can use continuation techniques [HYZ07, HYZ08] to dynamically update this parameter to substantially accelerate this algorithm.

2. *Accuracy:* NESTA depends on a few parameters that can be set in a very natural fashion. In fact, there is a clear relationship between the value of these parameters and the desired accuracy. Furthermore, our numerical experiments demonstrate that NESTA can find up to the first 4 or 5 significant digits of the optimal solution to (3.1.1), where $f(x)$ is the ℓ_1 norm or the total-variation norm of x, in a few hundred iterations. This makes NESTA amenable to solve recovery problems involving signals of very large sizes that also exhibit a great dynamic range.

[1] By "first-order", we mean methods which assume only that the local gradient information is available from a black-box oracle, but that once queried, the information persists; this is in contrast to a "pure" first-order method that only uses information about the current gradient.

3. *Flexibility:* NESTA can be adapted to solve many problems beyond ℓ_1 minimization with the same efficiency, such as total-variation (TV) minimization problems. In this chapter, we will also discuss applications in which f in (3.1.1) is given by $f(x) = \|Wx\|_{\ell_1}$, where one may think of W as a short-time Fourier transform also known as the Gabor transform, a curvelet transform, an undecimated wavelet transform, and so on, or a combination of these, or a general arbitrary dictionary of waveforms (note that this class of recovery problems also includes weighted ℓ_1 methods [CWB08]). This is particularly interesting because recent work [EMR07] suggests the potential advantage of this analysis-based approach over the classical *basis pursuit* in solving important inverse problems [EMR07].

A consequence of these properties is that NESTA, and more generally Nesterov's method, may be of interest to researchers working in the broad area of signal recovery from indirect and/or undersampled data.

Another contribution of this chapter is that it also features a fairly wide range of numerical experiments comparing various methods against problems involving realistic and challenging data. By challenging, we mean problems of very large scale where the unknown solution exhibits a large dynamic range; that is, problems for which classical second-order methods are too slow, and for which standard first-order methods do not provide sufficient accuracy. More specifically, Section 3.5 presents a comprehensive series of numerical experiments which illustrate the behavior of several state-of-the-art methods including interior point methods [KKB07], projected gradient techniques [HYZ07, vdBF08, FNW07], fixed point continuation, and iterative thresholding algorithms [HYZ07, YOGD08, BT09]. Most of these methods have either been perfected after several years of research [KKB07, FNW07], or are very recent state-of-the-art methods building on insight from the past few years. For example, the fixed point continuation method with active set [HYZ08], which represents a notable improvement over existing ideas, was released while we were working on the paper (and [ABDF11] was released while we were revising the paper).

3.1.2 Organization of the chapter and notations

As emphasized earlier, NESTA is based on Nesterov's ideas and Section 3.2.1 gives a brief but essential description of Nesterov's algorithmic framework. The proposed algorithm is introduced in Section 3.3. Inspired by continuation-like schemes, an accelerated version of NESTA is described in Section 3.3.6. We report on extensive and comparative numerical experiments in Section 3.5. Section 3.6 covers extensions of NESTA to minimize the ℓ_1 norm of Wx under data constraints (Section 3.6.1), and includes realistic simulations in the field of radar pulse detection and estimation. Section 3.6.3 extends NESTA to solve total-variation problems and presents numerical experiments which also demonstrate its remarkable efficiency there as well. Finally, we conclude with Section 3.8

discussing further extensions, which would address an even wider range of linear inverse problems.

Notations. Before we begin, it is best to provide a brief summary of the notations used throughout the chapter. As usual, vectors are written in small letters and matrices in capital letters. The ith entry of a vector x is denoted $x[i]$ and the (i,j)th entry of the matrix A is $A[i,j]$.

It is convenient to introduce some common optimization problems that will be discussed throughout. Solving sparse reconstruction problems can be approached via several different equivalent formulations. In this chapter, we particularly emphasize the quadratically constrained ℓ_1-minimization problem

$$\begin{array}{lll} (\mathrm{BP}_\epsilon) & \text{minimize} & \|x\|_{\ell_1} \\ & \text{subject to} & \|b - Ax\|_{\ell_2} \leq \epsilon, \end{array} \tag{3.1.2}$$

where ϵ quantifies the uncertainty about the measurements b as in the situation where the measurements are noisy. This formulation is often preferred because a reasonable estimate of ϵ may be known. A second frequently discussed approach considers solving this problem in Lagrangian form, i.e.,

$$(\mathrm{QP}_\lambda) \quad \text{minimize} \quad \lambda\|x\|_{\ell_1} + \frac{1}{2}\|b - Ax\|_{\ell_2}^2, \tag{3.1.3}$$

and is also known as the basis pursuit denoising problem (BPDN) [CDS98]. This problem is popular in signal and image processing [DDM04, FN03]; it is often interpreted as a maximum *a posteriori* estimate in a Bayesian setting or, simply, as a regularized least-squares problem. In statistics, the same problem is more well-known as the LASSO [Tib96]

$$\begin{array}{lll} (\mathrm{LS}_\tau) & \text{minimize} & \|b - Ax\|_{\ell_2} \\ & \text{subject to} & \|x\|_{\ell_1} \leq \tau. \end{array} \tag{3.1.4}$$

Standard optimization theory [Roc70] asserts that these three problems are of course equivalent provided that ϵ, λ, τ obey some special relationships. With the exception of the case where the matrix A is orthogonal, this functional dependence is hard to compute [vdBF08]. Because it is usually more natural to determine an appropriate ϵ rather than an appropriate λ or τ, the fact that NESTA solves (BP_ϵ) is a significant advantage. Further, note that theoretical equivalence of course does not mean that all three problems are just as easy (or just as hard) to solve. For instance, the constrained problem (BP_ϵ) may be harder to solve than (QP_λ). Hence, the fact that NESTA turns out to be competitive with algorithms that only solve (QP_λ) is quite remarkable.

3.2 NESTA

3.2.1 Nesterov's method to minimize smooth convex functions

In [Nes04, Nes83], Nesterov introduces a subtle algorithm to minimize any smooth convex function f on the convex set $\mathcal{Q}_p$,

$$\min_{x \in \mathcal{Q}_p} f(x). \tag{3.2.1}$$

We will refer to $\mathcal{Q}_p$ as the primal feasible set. The function f is assumed to be differentiable and its gradient $\nabla f(x)$ is Lipschitz and obeys

$$\forall x, y \in \mathrm{dom}(f), \quad ||\nabla f(x) - \nabla f(y)||_{\ell_2} \leq L \|x - y\|_{\ell_2}; \tag{3.2.2}$$

where $\mathrm{dom}(f) \equiv \mathbb{R}^n$ and L is an upper bound on the Lipschitz constant. With these assumptions, Nesterov's algorithm minimizes f over $\mathcal{Q}_p$ by iteratively estimating three sequences $\{x_k\}$, $\{y_k\}$, and $\{z_k\}$ while smoothing the feasible set $\mathcal{Q}_p$. The algorithm depends on two scalar sequences $\{\alpha_k\}$ and $\{\tau_k\}$, discussed below, and takes the following form (this specific form is based on [Nes05]):

Initialize x_0. For $k \geq 0$,

1. Compute $\nabla f(x_k)$.

2. Compute y_k:

   $y_k = \mathrm{argmin}_{x \in \mathcal{Q}_p} \frac{L}{2} \|x - x_k\|_{\ell_2}^2 + \langle \nabla f(x_k), x - x_k \rangle$.

3. Compute z_k:

   $z_k = \mathrm{argmin}_{x \in \mathcal{Q}_p} L p_p(x) + \sum_{i=0}^{k} \alpha_i \langle \nabla f(x_i), x - x_i \rangle$.

4. Update x_k:

   $x_{k+1} = \tau_k z_k + (1 - \tau_k) y_k$.

Stop when a given criterion is valid.

At step k, y_k is the current guess of the optimal solution. If we only performed the second step of the algorithm with y_{k-1} instead of x_k, we would obtain a standard first-order technique with convergence rate $\mathcal{O}(1/k)$. This second step can be seen as minimizing an approximate Taylor expansion of f about x_k, by taking an upper bound on the Hessian.

The novelty is that the sequence z_k "keeps in mind" the previous iterations since Step 3 involves a weighted sum of already computed gradients[2]. Another aspect of this step is that—borrowing ideas from smoothing techniques in optimization —it makes use of a *prox-function* $p_p(x)$ for the primal feasible set $\mathcal{Q}_p$. This function is strongly convex with parameter σ_p. Without loss of generality, we will set $\sigma_p = 1$. Assuming that $p_p(x)$ vanishes at the prox-center $x_p^c = \operatorname{argmin}_x p_p(x)$, this gives

$$p_p(x) \geq \frac{1}{2}\|x - x_p^c\|_{\ell_2}^2.$$

The prox-function is usually chosen so that $x_p^c \in \mathcal{Q}_p$, thus discouraging z_k from moving too far away from the center x_p^c.

The point x_k, at which the gradient of f is evaluated, is a weighted average between z_k and y_k. This is motivated by a theoretical analysis [Nes05, Tse08], which shows that if [3] $\alpha_k = 1/2(k+1)$ and $\tau_k = 2/(k+3)$, then the sequence $\{f(y_k)\}$ (y_k is feasible) converges to $f(x^\star)$ where $x^\star = \operatorname{argmin}_{x \in \mathcal{Q}_p} f(x)$, with the convergence rate (see [Nes05, Theorem 2])

$$f(y_k) - f(x^\star) \leq \frac{4Lp_p(x^\star)}{(k+1)^2}. \tag{3.2.3}$$

This decay is far better than what is achievable via standard gradient-based optimization techniques since we have an approximation scaling like L/k^2 instead of L/k.

3.3 Application to compressed sensing

We now apply Nesterov's algorithm to solve compressed sensing recovery problems, and refer to this extension as NESTA. For now, we shall be concerned with solving the quadratically constrained ℓ_1-minimization problem (3.1.2).

3.3.1 NESTA

We wish to solve (3.1.2), i.e., minimize $\|x\|_{\ell_1}$ subject to $\|b - Ax\|_{\ell_2} \leq \epsilon$, where $A \in \mathbb{R}^{m \times n}$ is singular ($m < n$).

[2] Using the sequence $\{\nabla f(x_i)\}_{1 \leq i \leq k}$ to update the current estimate x_k is not new. For instance, in algorithms such as SESOP [EMZ07], the search direction is optimally evaluated at each iteration in the subspace spanned by previous gradients; this requires storing a set of previous values, which may not be practical for large-scale problems.

[3] Other choices for α_k and τ_k are possible [Nes05].

In this chapter, with the exception of §3.7, we assume that A^*A is an orthogonal projector, i.e., the rows of A are orthonormal. This is often the case in compressed sensing applications where it is common to take A as a submatrix of a unitary transformation which admits a fast algorithm for matrix-vector products; special instances include the discrete Fourier transform, the discrete cosine transform, the Hadamard transform, the noiselet transform, and so on. Basically, collecting incomplete structured orthogonal measurements is the prime method for efficient data acquisition in compressed sensing. Section 3.8 discusses different approaches for relaxing this assumption.

From non-smooth to smooth functions

The algorithm described in the previous paragraph has been introduced to minimize a smooth function f; it cannot be applied directly to solve (BP_ϵ) as the function $f(x) = \|x\|_{\ell_1}$ is not smooth. In an innovative paper [Nes05], Nesterov recently extended this framework to deal with a certain class of non-smooth convex functions. Suppose that the function f to be minimized has the form

$$f(x) = \max_{u \in \mathcal{Q}_d} \langle u, Wx \rangle, \tag{3.3.1}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^p$, and $W \in \mathbb{R}^{p \times n}$. W is assumed to have full column rank. We will refer to $\mathcal{Q}_d$ as the dual feasible set, and assume it is convex. In [Nes05], Nesterov proposed substituting f by the smooth approximation

$$f_\mu(x) = \max_{u \in \mathcal{Q}_d} \langle u, Wx \rangle - \mu\, p_d(u), \tag{3.3.2}$$

where $p_d(u)$ is a *prox-function* for $\mathcal{Q}_d$; that is, $p_d(u)$ is continuous and strongly convex on $\mathcal{Q}_d$, with convexity parameter σ_d which we will take to be 1. Nesterov proved that f_μ is continuously differentiable, and that its gradient obeys

$$\nabla f_\mu(x) = W^* u_\mu(x), \tag{3.3.3}$$

where $u_\mu(x)$ is the optimal solution of (3.3.2). Furthermore, ∇f_μ is shown to be Lipschitz with constant

$$L = \frac{1}{\mu \sigma_d} \|W\|^2 \tag{3.3.4}$$

($\|W\|$ is the operator norm of W). It is now possible to apply Nesterov's accelerated method [Nes05] to the smoothed function f_μ. This produces a feasible sequence of points that will weakly converge to the minimizer of the smoothed problem at rate $\mathcal{O}(k^{-2})$, and empirically, the minimizers of the smoothed problem and the unsmoothed problem can be made arbitrarily close by taking $\mu \to 0$.

The particular case $f(x) = \|x\|_{\ell_1}$

In this setting, the ℓ_1 norm can be formulated as in (3.3.1) since

$$\|x\|_{\ell_1} = \max_{u \in \mathcal{Q}_d} \langle u, x \rangle,$$

where the dual feasible set $\mathcal{Q}_d$ is the ℓ_∞ ball $\mathcal{Q}_d = \{u : \|u\|_\infty \leq 1\}$. Therefore, a natural smooth approximation to the ℓ_1 norm is

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

where $p_d(u)$ is our dual prox-function. For $p_d(u)$, we would like a strongly convex function, which is known analytically and takes its minimum value (equal to zero) at some $u_d^c \in \mathcal{Q}_d$. It is also usual to have $p_d(u)$ separable. Taking these criteria into account, a convenient choice is $p_d(u) = \frac{1}{2}\|u\|_{\ell_2}^2$ whose strong convexity parameter is equal to 1. With this prox-function, f_μ is the well-known Huber function[4] and ∇f_μ is Lipschitz with constant $1/\mu$. In particular, $\nabla f_\mu(x)$ is given by

$$\nabla f_\mu(x)[i] = \begin{cases} \mu^{-1}\, x[i], & \text{if } |x[i]| < \mu, \\ \operatorname{sgn}(x[i]), & \text{otherwise.} \end{cases} \tag{3.3.5}$$

As proposed in [Nes05], Nesterov's algorithm can then be applied to solve

$$\min_{x \in \mathcal{Q}_p} f_\mu(x), \tag{3.3.6}$$

where $\mathcal{Q}_p = \{x : \|b - Ax\|_{\ell_2} \leq \epsilon\}$. Let us note that this problem is an approximation to (BP_ϵ) with $0 \leq f(x) - f_\mu(x) \leq n\mu/2$ for all $x \in \mathbb{R}^n$; this bound is overly pessimistic if x is s-sparse, since $f(0) = f_\mu(0)$, so the expected approximation error is closer to $s\mu/2$ for these signals.

The only remaining details of the algorithm are about the auxiliary updates of y_k and z_k in Steps 2 and 3. In the next section, we show that these steps are computationally very cheap when A^*A is assumed to be an orthogonal projector.[5]

3.3.2 Updating y_k

To compute y_k, we need to solve a problem of the form

$$y_k = \operatorname*{argmin}_{x \in \mathcal{Q}_p} \frac{L}{2}\|d - x\|_{\ell_2}^2 + \langle c, x - x_k \rangle, \tag{3.3.7}$$

[4]The smoothed version of the total-variation norm is not known to have a specific name.

[5]Taking $x_0 = A^*b$ which is feasible and observing that $f(x^\star) \leq f(A^*b)$ for any minimizer $x^\star$ implies the boundedness of $x^\star$ and, hence, that of $p_p(x^\star)$ in (3.2.3). This follows from $f(x) \leq \|x\|_{\ell_1} \leq f(x) + \mu n/2$ valid for all x, giving $\|x^\star\|_{\ell_1} \leq f(A^*b) + \mu n/2$.

where x_k is from the previous iterate and L is given by (3.3.4), $d = x_k$ and $c = \nabla f_\mu(x_k)$. The Lagrangian for this problem is

$$\mathcal{L}(x, \lambda) = \frac{L}{2}\|d - x\|_{\ell_2}^2 + \frac{\lambda}{2}\left(\|b - Ax\|_{\ell_2}^2 - \epsilon^2\right) + \langle c, x - x_k\rangle, \tag{3.3.8}$$

and at the primal-dual solution (y_k, λ_ϵ), the Karush-Kuhn-Tucker (KKT) conditions [Roc70] read

$$\begin{aligned} \|b - Ay_k\|_{\ell_2} &\leq \epsilon, \\ \lambda_\epsilon &\geq 0, \\ \lambda_\epsilon\left(\|b - Ay_k\|_{\ell_2}^2 - \epsilon^2\right) &= 0, \\ L(y_k - d) + \lambda_\epsilon A^*(Ay_k - b) + c &= 0. \end{aligned}$$

From the stationarity condition, y_k is the solution to the linear system

$$\left(I + \frac{\lambda_\epsilon}{L}A^*A\right) y_k = \frac{\lambda_\epsilon}{L}A^*b + d - \frac{1}{L}c. \tag{3.3.9}$$

As discussed earlier, our assumption is that A^*A is an orthogonal projector so that

$$y_k = \left(I - \frac{\lambda_\epsilon}{\lambda_\epsilon + L}A^*A\right)\left(\frac{\lambda_\epsilon}{L}A^*b + d - \frac{1}{L}c\right). \tag{3.3.10}$$

In this case, computing y_k is cheap since no matrix inversion is required—only a few matrix-vector products are necessary. Moreover, from the KKT conditions, the value of the optimal Lagrange multiplier is obtained explicitly, and equals

$$\lambda_\epsilon = \max(0, L\epsilon^{-1}\|b - Aq\|_{\ell_2} - L), \quad q = d - L^{-1}c \tag{3.3.11}$$

where $c = \nabla f_\mu(x_k)$ and $d = x_k$. Observe that this can be computed beforehand since it only depends on x_k and $\nabla f_\mu(x_k)$.

3.3.3 Updating z_k

To compute z_k, we need to solve

$$z_k = \underset{x \in \mathcal{Q}_p}{\operatorname{argmin}}\, Lp_p(x) + \langle \sum_{i \leq k} \alpha_i \nabla f_\mu(x_i), x - x_k\rangle, \tag{3.3.12}$$

where $p_p(x)$ is the primal prox-function. The point z_k differs from y_k since it is computed from a weighted cumulative gradient $\sum_{i \leq k} \alpha_i \nabla f_\mu(x_i)$, making it less prone to zig-zagging, which typically occurs when we have highly elliptical level sets. This step keeps a memory from the previous steps

and forces z_k to stay near the prox-center. A good primal prox-function is a smooth and strongly convex function that is likely to have some positive effect near the solution. In the setting of (3.1.1), a suitable smoothing prox-function may be

$$p_p(x) = \frac{1}{2}\|x - x_0\|_{\ell_2}^2 \tag{3.3.13}$$

for some $x_0 \in \mathbb{R}^n$, e.g., an initial guess of the solution. Other choices of primal feasible set $\mathcal{Q}_p$ may lead to other choices of prox-functions. For instance, when $\mathcal{Q}_p$ is the standard simplex, choosing an entropy distance for $p_p(x)$ is smarter and more efficient, see [Nes05]. In this chapter, the primal feasible set is quadratic, which makes the Euclidean distance a reasonable choice. What is more important, however, is that this choice allows very efficient computations of z_k, while other choices may considerably slow down each Nesterov iteration. Finally, notice that the bound on the error at iteration k in (3.2.3) is proportional to $p_p(x^\star)$; choosing x_0 wisely (a good first guess) can make $p_p(x^\star)$ small. When nothing is known about the solution, a natural choice may be $x_0 = A^*b$; this idea will be developed in Section 3.3.6.

With (3.3.13), the strong convexity parameter of $p_p(x)$ is equal to 1, and to compute z_k we need to solve a quadratically constrained quadratic program that has exactly the same formulation as (3.3.7), where $d = x_0$ and $c = \sum_{i \le k} \alpha_i \nabla f_\mu(x_i)$. Updating z_k is then just as cheap as updating y_k. Furthermore, many of the matrix-vector multiplications from the y_k update can be reused.

3.3.4 Computational complexity

The computational complexity of each of NESTA's steps is clear. In large-scale problems, most of the work is in the application of A and A^*. Define $\mathcal{C}_A$ to be the complexity of applying A or A^*. The first step, namely, computing ∇f_μ, only requires vector operations whose complexity is $\mathcal{O}(n)$. Steps 2 and 3 require the application of A or A^* three times each (we only need to compute A^*b once). Hence, the total complexity of a single NESTA iteration is $6\,\mathcal{C}_A + \mathcal{O}(n)$ where $\mathcal{C}_A$ is dominant.

The calculations above are in some sense overly pessimistic. In compressed sensing applications, it is common to choose A as a submatrix of a unitary transformation U, which admits a fast algorithm for matrix-vector products. In the sequel, it might be useful to think of A as a subsampled DFT. In this case, letting R be the $m \times n$ matrix extracting the observed measurements, we have $A = RU$. The trick then is to compute in the U-domain directly. Making the change of variables $x \leftarrow Ux$, our problem is

$$\begin{array}{ll} \text{minimize} & \hat{f}_\mu(x) \\ \text{subject to} & \|b - Rx\|_{\ell_2} \le \epsilon, \end{array}$$

where $\hat{f}_\mu = f_\mu \circ U^*$. The gradient of $\hat{f}_\mu$ is then

$$\nabla \hat{f}_\mu(x) = U \, \nabla f_\mu(U^* x).$$

With this change of variables, Steps 2 and 3 do not require applying U or U^* since

$$y_k = \left(I - \frac{\lambda}{\lambda + L} R^* R \right) \left(\frac{\lambda}{L} R^* b + x_k - \frac{1}{L} \nabla \hat{f}_\mu(x_k) \right),$$

where R^*R is the diagonal matrix with 0/1 diagonal entries depending on whether a coordinate is sampled or not. As before, $\lambda_\epsilon = \max(0, \|b - Rq\|_{\ell_2} - L)$ with $q = x_k - L^{-1} \nabla \hat{f}_\mu(x_k)$. The complexity of Step 2 is now $\mathcal{O}(n)$ and the same applies to Step 3.

Put $\mathcal{C}_U$ for the complexity of applying U and U^*. The complexity of Step 1 is now $2\,\mathcal{C}_U$, so that this simple change of variables reduces the cost of each NESTA iteration to $2\,\mathcal{C}_U + \mathcal{O}(n)$. For example, in the case of a subsampled DFT (or something similar), the cost of each iteration is essentially that of two FFTs. Hence, each iteration is extremely fast.

3.3.5 Parameter selection

NESTA involves the selection of a single smoothing parameter μ and of a suitable stopping criterion. For the latter, our experience indicates that a robust and fairly natural stopping criterion is to terminate the algorithm when the relative variation of f_μ is small. Define Δf_μ as

$$\Delta f_\mu := \frac{|f_\mu(x_k) - \bar{f}_\mu(x_k)|}{\bar{f}_\mu(x_k)}, \quad \bar{f}_\mu(x_k) := \frac{1}{\min\{10, k\}} \sum_{l=1}^{\min\{10,k\}} f_\mu(x_{k-l}). \tag{3.3.14}$$

Then convergence is claimed when

$$\Delta f_\mu < \delta$$

for some $\delta > 0$. In our experiments, $\delta \in \{10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$ depending upon the desired accuracy. These values might depend on the problem type and should be optimized accordingly. Clearly, δ should be small when μ is small, since this is the high accuracy situation; when μ is larger, δ may be taken to be larger as well.

The choice of μ is based on a tradeoff between the accuracy of the smoothed approximation f_μ (basically, $\lim_{\mu \to 0} f_\mu(x) = \|x\|_{\ell_1}$) and the speed of convergence (the convergence rate is proportional to μ). With noiseless data, μ is directly linked to the desired accuracy. To illustrate this, we have observed in [BC09] that when the true signal x^0 is exactly sparse and is actually the minimum solution under the equality constraints $Ax^0 = b$, the ℓ_∞ error on the nonzero entries is on the order of μ. The link between μ and accuracy will be further discussed in Section 3.4.3.

3.3.6 Accelerating NESTA with continuation

Inspired by homotopy techniques which find the solution to the LASSO problem (3.1.4) for values of τ ranging in an interval $[0, \tau_{\max}]$, [HYZ07] and [DFL08] independently introduced a fixed point continuation technique which solves ℓ_1-penalized least-square problems (3.1.3),

$$(\mathrm{QP}_\lambda) \quad \text{minimize} \quad \lambda \|x\|_{\ell_1} + \frac{1}{2}\|b - Ax\|_{\ell_2}^2,$$

for values of λ obeying $0 < \lambda < \|A^* b\|_{\ell_\infty}$. The continuation solution approximately follows the path of solutions to the problem (QP_λ) and, hence, the solutions to (3.1.1) and (3.1.4) may be found by solving a sequence of ℓ_1 penalized least-squares problems.

The point of this is that it has been noticed (see [HYZ07, OPT00, DT08, WNF09]) that solving (3.1.3) (resp. the LASSO (3.1.4)) is faster when λ is large (resp. τ is low). This observation greatly motivates the use of continuation for solving (3.1.3) for a fixed λ_f. The idea is simple: propose a sequence of problems with decreasing values of the parameter λ, $\lambda_0 > \cdots > \lambda_f$, and use the intermediate solution as a warm start for the next problem. This technique has been used with success in [FNW07, vdBF08]. Continuation has been shown to be a very successful tool to increase the speed of convergence, in particular when dealing with large-scale problems and high dynamic range signals.

Likewise, our proposed algorithm can greatly benefit from a continuation approach. Recall that to compute y_k, we need to solve

$$\begin{aligned} y_k &= \operatorname*{argmin}_{x \in \mathcal{Q}_p} \frac{L}{2}\|x - x_k\|_{\ell_2}^2 + \langle c, x \rangle \\ &= \operatorname*{argmin}_{x \in \mathcal{Q}_p} \|x - (x_k - L^{-1}c)\|_{\ell_2}^2 \end{aligned}$$

for some vector c. Thus with $\mathcal{P}_{\mathcal{Q}_p}$ the projector onto $\mathcal{Q}_p$, $y_k = \mathcal{P}_{\mathcal{Q}_p}(x_k - L^{-1}c)$. Now two observations are in order.

1. Computing y_k is similar to a projected gradient step as the Lipschitz constant L^{-1} plays the role of the step size. Since L is proportional to μ^{-1}, the larger μ, the larger the step-size, and the faster the convergence. This also applies to the sequence $\{z_k\}$.

2. For a fixed value of μ, the convergence rate of the algorithm obeys

$$f_\mu(y_k) - f_\mu(x_\mu^\star) \leq \frac{2L\|x_\mu^\star - x_0\|_{\ell_2}^2}{k^2},$$

where $x_\mu^\star$ is the optimal solution to $\min f_\mu$ over $\mathcal{Q}_p$. On the one hand, the convergence rate is proportional to μ^{-1}, so a large value of μ is beneficial. On the other hand, choosing a good

guess x_0 close to $x^\star_\mu$ provides a low value of $p_p(x^\star_\mu) = \frac{1}{2}\|x^\star_\mu - x_0\|^2_{\ell_2}$, also improving the rate of convergence. Warm-starting with x_0 from a previous solve not only changes the starting point of the algorithm, but it beneficially changes p_p as well.

These two observations motivate the following continuation-like algorithm:

Initialize μ_0, x_0 and the number of continuation steps T. For $t = 1, \cdots, T$,

1. Apply Nesterov's algorithm with $\mu = \mu^{(t)}$ and $x_0 = x_{\mu^{(t-1)}}$.

2. Decrease the value of μ: $\mu^{(t+1)} = \gamma\mu^{(t)}$ with $\gamma < 1$.

Stop when the desired value of μ_f is reached.

This algorithm iteratively finds the solutions to a succession of problems with decreasing smoothing parameters $\mu_0 > \cdots > \mu_f = \gamma^T\mu_0$ producing a sequence of—hopefully—finer estimates of $x^\star_{\mu_f}$ [6]; these intermediate solutions are cheap to compute and provide a string of convenient first guesses for the next problem. In practice, they are solved with less accuracy, making them even cheaper to compute. This kind of continuation is different from standard continuation techniques [HYZ07, OPT00, DT08, WNF09]. Indeed, while standard continuation solves a sequence of problems (QP_λ) with different values of the Lagrange multiplier λ (so that intermediate solutions may have a meaningful interpretation if λ is not decreased too quickly), the proposed continuation technique acts internally by changing the smoothing parameter μ while keeping ϵ and, therefore, the feasible set, fixed. While intermediate solutions may have a less meaningful interpretation, the continuation may actually be more effective than traditional continuation, since not only is the new step warm-started, but the primal prox-function is improved.

The value of μ_f is based on a desired accuracy as explained in Section 3.3.5. As for an initial value μ_0, (3.3.5) makes clear that the smoothing parameter plays a role similar to a threshold. A first choice may then be $\mu_0 = 0.9\|A^*b\|_{\ell_\infty}$.

We illustrate the good behavior of the continuation-inspired algorithm by applying NESTA with continuation to solve a sparse reconstruction problem from partial frequency data. In this series of experiments, we assess the performance of NESTA while the dynamic range of the signals to be recovered increases.

The signals x are s-sparse signals—that is, have exactly s nonzero components—of size $n = 4096$ and $s = m/40$. Put Λ for the indices of the nonzero entries of x; the amplitude of each nonzero entry is distributed uniformly on a logarithmic scale with a fixed dynamic range. Specifically, each

[6] The parameter γ does not define a new parameter; it depends upon μ_f, μ_0, and T via $\gamma = (\mu_f/\mu_0)^{1/T}$.

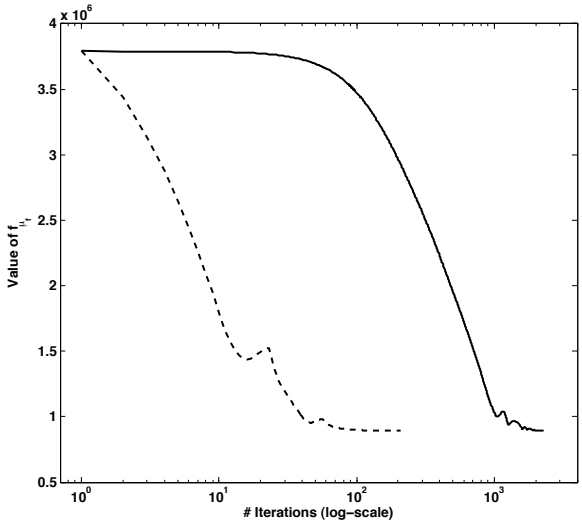


Figure 3.1: Value of $f_{\mu_f}(x_k)$ as a function of iteration k. Solid line: without continuation. Dashed line: with continuation. Here, the test signal has 100 dB of dynamic range.

nonzero entry is generated as follows:

$$x[i] = \eta_1[i] 10^{\alpha \eta_2[i]}, \tag{3.3.15}$$

where $\eta_1[i] = \pm 1$ with probability $1/2$ (a random sign) and $\eta_2[i]$ is uniformly distributed in $[0, 1]$. The parameter α quantifies the dynamic range. Unless specified otherwise, a dynamic range of d dB means that $\alpha = d/20$ (since for large signals α is approximately the logarithm base 10 of the ratio between the largest and the lowest magnitudes). For instance, 80 dB signals are generated according to (3.3.15) with $\alpha = 4$.

The measurements Ax consist of $m = n/8$ random discrete cosine measurements so that A^*A is diagonalized by the DCT. Finally, b is obtained by adding a white Gaussian noise term with standard deviation $\sigma = 0.1$. The initial value of the smoothing parameter is $\mu_0 = \|A^*b\|_{\ell_\infty}$ and the terminal value is $\mu_f = 2\sigma$. The algorithm terminates when the relative variation of f_μ is lower than $\delta = 10^{-5}$. NESTA with continuation is applied to 10 random trials for varying number of continuation steps T and various values of the dynamic range. Figure 3.1 graphs the value of f_{μ_f} while applying NESTA with and without continuation as a function of the iteration count. The number of continuation steps is set to $T = 4$.

One can observe that computing the solution to $\min f_{\mu_f}$ (solid line) takes a while when computed with the final value μ_f; notice that NESTA seems to be slow at the beginning (number of iterations lower than 15). In the meantime NESTA with continuation rapidly estimates a sequence of coarse intermediate solutions that converges to the solution to $\min f_{\mu_f}$ In this case, continuation clearly enhances the global speed of convergence with a factor 10. Figure 3.2 provides deeper insights into

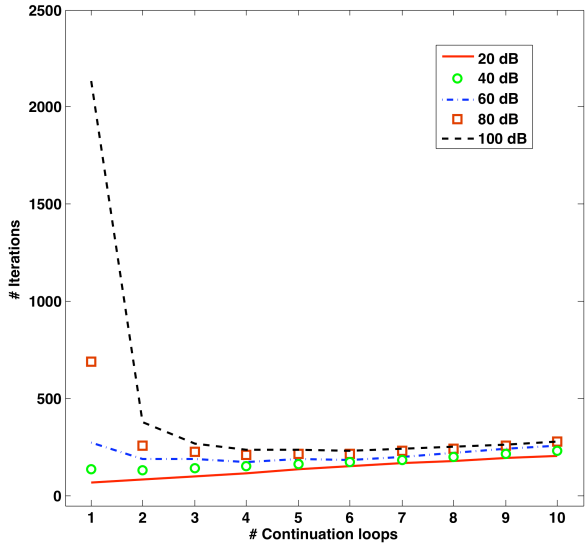


Figure 3.2: Total number of iterations required for convergence with a varying number of continuation steps and for different values of the dynamic range

the behavior of continuation with NESTA and shows the number of iterations required to reach convergence for varying values of the continuation steps T for different values of the dynamic range.

When the ratio μ_0/μ_f is low or when the required accuracy is low, continuation is not as beneficial: intermediate continuation steps require a number of iterations which may not speed up overall convergence. The step-size which is about $L_{\mu_f}^{-1}$ works well in this regime. When the dynamic range increases and we require more accuracy, however, the ratio μ_0/μ_f is large, since $\mu_0 = .9\|A^*b\|_{\ell_\infty} \approx \|x\|_{\ell_\infty} \gg \sigma$, and continuation provides considerable improvements. In this case, the step size $L_{\mu_f}^{-1}$ is too conservative and it takes a while to find the large entries of x. Empirically, when the dynamic range is 100 dB, continuation improves the speed of convergence by a factor of 8. As this factor is likely to increase exponentially with the dynamic range (when expressed in dB), NESTA with continuation seems to be a better candidate for solving sparse reconstruction problems with high accuracy.

Interestingly, the behavior of NESTA with continuation seems to be quite stable: increasing the number of continuation steps does not increase dramatically the number of iterations. In practice, although the ideal T is certainly signal dependent, we have observed in our problem settings that choosing $T \in \{4, 5, 6\}$ leads to reasonable results. The value of T should certainly be optimized for particular problem types.

3.3.7 Some theoretical considerations

The convergence of NESTA with and without continuation is straightforward. The following theorem states that at each continuation step with $\mu = \mu^{(t)}$, the sequence $f_\mu(y_k)$ converges to $f_\mu(x_\mu^\star)$. Global

convergence is proved by applying this theorem to $t = T$.

Theorem 3.3.1. *At each continuation step t, $\lim_{k\to\infty} f_{\mu^{(t)}}(y_k) = f_{\mu^{(t)}}(x^\star_{\mu^{(t)}})$, and*

$$f_{\mu^{(t)}}(y_k) - f_{\mu^{(t)}}(x^\star_{\mu^{(t)}}) \leq \frac{2L_{\mu^{(t)}}\|x^\star_{\mu^{(t)}} - x_{\mu^{(t-1)}}\|^2_{\ell_2}}{k^2}.$$

Proof. Immediate by using [Nes05, Theorem 2]. □

As mentioned earlier, continuation may be valuable for improving the speed of convergence. Let each continuation step t stop after $\mathcal{N}^{(t)}$ iterations with

$$\mathcal{N}^{(t)} = \sqrt{\frac{2L_{\mu^{(t)}}}{\gamma^t \delta_0}}\|x^\star_{\mu^{(t)}} - x^\star_{\mu^{(t-1)}}\|_{\ell_2}$$

so that we have

$$f_{\mu^{(t)}}(y_k) - f_{\mu^{(t)}}(x^\star_{\mu^{(t)}}) \leq \gamma^t \delta_0,$$

where the accuracy $\gamma^t\delta_0$ becomes tighter as t increases. Then summing up the contribution of all the continuation steps gives

$$\mathcal{N}_c = \sqrt{\frac{2}{\mu_0\delta_0}}\sum_{t=1}^{T}\gamma^{-t}\|x^\star_{\mu^{(t)}} - x^\star_{\mu^{(t-1)}}\|_{\ell_2}.$$

When NESTA is applied without continuation, the number of iterations required to reach convergence is

$$\mathcal{N} = \sqrt{\frac{2}{\mu_0\delta_0}}\gamma^{-T}\|x^\star_{\mu_f} - x_0\|_{\ell_2}.$$

Now the ratio $\mathcal{N}_c/\mathcal{N}$ is given by

$$\frac{\mathcal{N}_c}{\mathcal{N}} = \sum_{t=1}^{T}\gamma^{T-t}\frac{\|x^\star_{\mu^{(t)}} - x^\star_{\mu^{(t-1)}}\|_{\ell_2}}{\|x^\star_{\mu_f} - x_0\|_{\ell_2}}. \tag{3.3.16}$$

Continuation is definitely worthwhile when the right-hand side is smaller than 1. Interestingly, this quantity is directly linked to the path followed by the sequence $x_0 \to x_{\mu^{(1)}} \to \cdots \to x_{\mu_f}$. More precisely, it is related to the smoothness of this path; for instance, if all the intermediate points $x_{\mu^{(t)}}$ belong to the segment $[x_0, x_{\mu_f}]$ in an ordered fashion, then $\sum_t \|x^\star_{\mu^{(t)}} - x_{\mu^{(t-1)}}\|_{\ell_2} = \|x^\star_{\mu_f} - x_0\|_{\ell_2}$. Hence, $\frac{\mathcal{N}_c}{\mathcal{N}} < 1$ and continuation improves the convergence rate.

Figure 3.3 illustrates two typical solution paths with continuation. When the sequence of solutions obeys $\|x_0\|_{\ell_1} \geq \ldots \|x^\star_{\mu^{(t)}}\|_{\ell_1} \ldots \geq \|x^\star_{\mu_f}\|_{\ell_1}$ (this is the case when $x_0 = A^*b$ and $\mu_1 \geq \ldots \mu^{(t)} \ldots \geq \mu_f$), the solution path is likely to be "smooth"; that is, the solutions obey $\|x^\star_{\mu^{(t)}} - x^\star_{\mu_f}\|_{\ell_2} \geq \|x^\star_{\mu^{(t+1)}} - x^\star_{\mu_f}\|_{\ell_2}$ as on the left of Figure 3.3. The "non-smooth" case on the right of

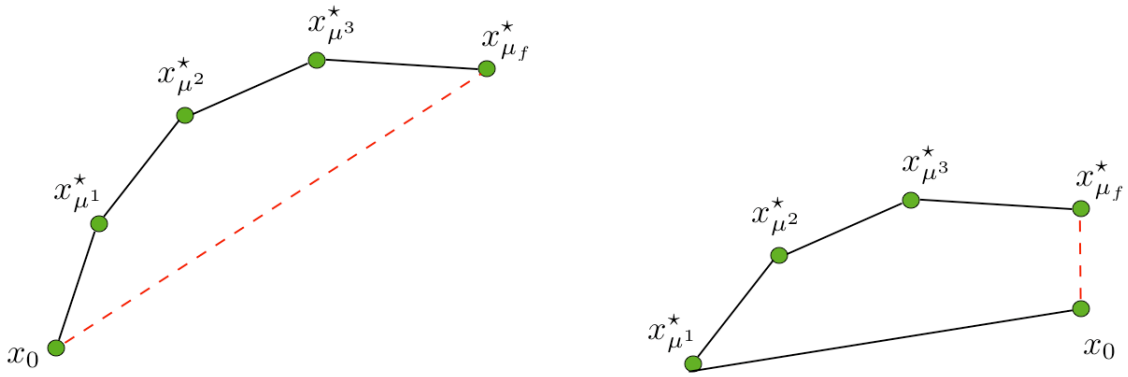


Figure 3.3: Typical solution paths. Left: smooth solution path. Right: non-smooth solution path

Figure 3.3 arises when the sequence of smoothing parameters does not provide estimates of $x^\star_{\mu_f}$ that are all better than x_0. Here, computing some of the intermediate points $\{x^\star_{\mu^{(t)}}\}$ is wasteful and continuation fails to be faster.

3.4 Accurate optimization

A significant fraction of the numerical part of this chapter focuses on comparing different sparse recovery algorithms in terms of speed and accuracy. In this section, we first demonstrate that NESTA can easily recover the exact solution to (BP_ϵ) with a precision of 5 to 6 digits. Speaking of precision, we shall essentially use two criteria to evaluate accuracy.

1. The first is the (relative) error on the objective functional

$$\frac{\|x\|_{\ell_1} - \|x^\star\|_{\ell_1}}{\|x^\star\|_{\ell_1}}, \tag{3.4.1}$$

   where $x^\star$ is the optimal solution to (BP_ϵ) .

2. The second is the accuracy of the optimal solution itself and is measured via

$$\|x - x^\star\|_{\ell_\infty}, \tag{3.4.2}$$

   which gives a precise value of the accuracy per entry.

3.4.1 Is NESTA accurate?

For general problem instances, the exact solution to (BP_ϵ) (or equivalently (QP_λ)) cannot be computed analytically. Under some conditions, however, a simple formula is available when the optimal

Table 3.1: Assessing FISTA's and NESTA's accuracy when the optimal solution is known. The relative error on the optimal value is given by (3.4.1) and the ℓ_∞ error on the optimal solution by (3.4.2). $\mathcal{N}_A$ is the number of calls to A or A^* to compute the solution.

Method	ℓ_1-norm	Rel. error ℓ_1-norm	ℓ_∞ error	$\mathcal{N}_A$
$x^\star$	3.33601e+6			
FISTA	3.33610e+6	2.7e-5	0.31	40000
NESTA $\mu = 0.02$	3.33647e+6	1.4e-4	0.08	513

solution has exactly the same support and the same sign as the unknown (sparse) x^0 (recall the model $b = Ax^0 + z$). Denote by I the support of x^0, $I := \{i : |x^0[i]| > 0\}$. Then if x^0 is sufficiently sparse and if the nonzero entries of x^0 are sufficiently large, the solution $x^\star$ to (QP_λ) is given by

$$x^\star[I] = (A[I]^* A[I])^{-1}(A[I]^* b - \lambda \operatorname{sgn}(x^0[I])), \tag{3.4.3}$$

$$x^\star[I^c] = 0, \tag{3.4.4}$$

see [CP07a] for example. In this expression, $x[I]$ is the vector with indices in I and $A[I]$ is the submatrix with columns indices in I.

To evaluate NESTA's accuracy, we set $n = 262{,}144$, $m = n/8$, and $s = m/100$ (this is the number of nonzero coordinates of x_0). The absolute values of the nonzero entries of x_0 are distributed between 1 and 10^5 so that we have about 100 dB of dynamic range. The measurements Ax^0 are discrete cosine coefficients selected uniformly at random. We add Gaussian white noise with standard deviation $\sigma = 0.01$. We then compute the solution (3.4.3), and make sure it obeys the KKT optimality conditions for (QP_λ) so that the optimal solution is known.

We run NESTA with continuation with the value of $\epsilon := \|b - Ax^\star\|$. We use $\mu = 0.02$, $\delta = 10^{-7}$ and the number of continuation steps is set to 5. Table 3.1 reports on numerical results. First, the value of the objective functional is accurate up to 4 digits. Second, the computed solution is very accurate since we observe an ℓ_∞ error of 0.08. Now recall that the nonzero components of $x^\star$ vary from about 1 to 10^5 so that we have high accuracy over a huge dynamic range. This can also be gleaned from Figure 3.4 which plots NESTA's solution versus the optimal solution, and confirms the excellent precision of our algorithm.

3.4.2 Setting up a reference algorithm for accuracy tests

In general situations, a formula for the optimal solution is of course unavailable, and evaluating the accuracy of solutions requires defining a method of reference. In this chapter, we will use FISTA [BT09] as such a reference since it is an efficient algorithm that also turns out to be extremely easy to use; in particular, no parameter has to be tweaked, except for the standard stopping criterion (maximum number of iterations and tolerance on the relative variation of the objective function).

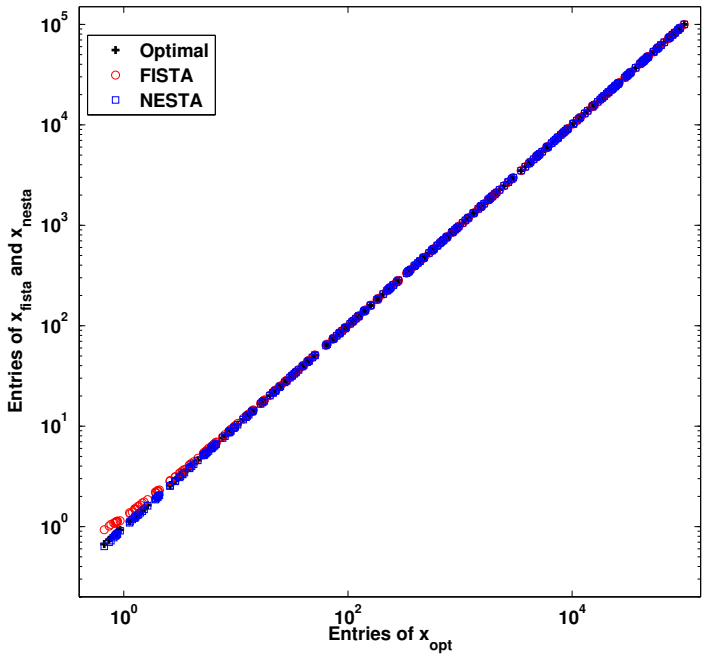


Figure 3.4: Entries of the computed solutions versus the optimal solution. The absolute values of the entries on the support of the optimal solution are plotted.

We run FISTA with 20,000 iterations on the same problem as above, and report its accuracy in Table 3.1. The ℓ_1-norm is exact up to 4 digits. Furthermore, Figure 3.4 shows the entries of FISTA's solution versus those of the optimal solution, and one observes a very good fit (near perfect when the magnitude of a component of $x^\star$ is higher than 3). The ℓ_∞ error between FISTA's solution and the optimal solution $x^\star$ is equal to 0.31; that is, the entries are exact up to ± 0.31. Because this occurs over an enormous dynamic range, we conclude that FISTA also gives very accurate solutions provided that sufficiently many iterations are taken. We have observed that running FISTA with a high number of iterations—typically greater than 20,000—provides accurate solutions to (QP_λ), and this is why we will use it as our method of reference in the forthcoming comparisons from this section and the next.

3.4.3 The smoothing parameter μ and NESTA's accuracy

By definition, μ fixes the accuracy of the approximation f_μ to the ℓ_1 norm and, therefore, NESTA's accuracy directly depends on this parameter. We now propose to assess the accuracy of NESTA for different values of μ. The problem sizes are as before, namely, $n = 262{,}144$ and $m = n/8$, except that now the unknown x^0 is far less sparse with $s = m/5$. The standard deviation of the additive Gaussian white noise is also higher, and we set $\sigma = 0.1$.

Because of the larger value of s and σ, it is no longer possible to have an analytic solution from (3.4.3). Instead, we use FISTA to compute a reference solution x_F, using 20,000 iterations and with

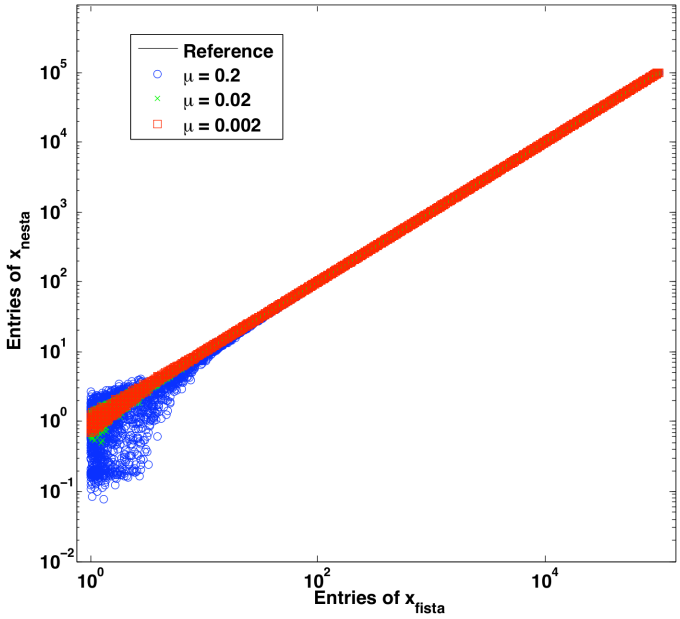


Figure 3.5: Entries of the computed solutions versus the optimal solution. We plot the absolute values of the entries on the set where the magnitude of the optimal solution exceeds 1.

$\lambda = 0.0685$, which gives $\|b - Ax_F\|_{\ell_2}^2 \simeq (m + 2\sqrt{2m})\sigma^2$. To be sure that FISTA's solution is very close to the optimal solution, we check that the KKT stationarity condition is nearly verified. If $I_\star$ is the support of the optimal solution $x^\star$, this condition reads

$$A[I_\star]^*(b - Ax^\star) = \lambda \operatorname{sgn}(x^\star[I_\star]),$$
$$\|A[I_\star^c]^*(b - Ax^\star)\|_{\ell_\infty} \leq \lambda.$$

Now define I to be the support of x_F. Then, here, x_F obeys

$$\|A[I]^*(b - Ax_F) - \lambda \operatorname{sgn}(x_F[I])\|_{\ell_\infty} = 2.6610^{-10}\lambda,$$
$$\|A[I^c]^*(b - Ax_F)\|_{\ell_\infty} \leq 0.99\lambda.$$

This shows that x_F is extremely close to the optimal solution.

NESTA is run with $T = 5$ continuation steps for three different values of $\mu \in \{0.2, 0.02, 0.002\}$ (the tolerance δ is set to 10^{-6}, 10^{-7}, and 10^{-8} respectively). Figure 3.5 plots the solutions given by NESTA versus the "optimal solution" x_F. Clearly, when μ decreases, the accuracy of NESTA increases just as expected. More precisely, notice in Table 3.2 that for this particular experiment, decreasing μ by a factor of 10 gives about 1 additional digit of accuracy on the optimal value.

According to this table, $\mu = 0.02$ seems a reasonable choice for this particular signal to guarantee an accurate solution, since one has between 4 and 5 digits of accuracy on the optimal value, and

Table 3.2: NESTA's accuracy. The errors and number of function calls $\mathcal{N}_A$ have the same meaning as in Table 3.1.

Method	ℓ_1-norm	Rel. error ℓ_1-norm	ℓ_∞ error	$\mathcal{N}_A$
FISTA	5.71539e+7			
NESTA $\mu = 0.2$	5.71614e+7	1.3e-4	3.8	659
NESTA $\mu = 0.02$	5.71547e+7	1.4e-5	0.96	1055
NESTA $\mu = 0.002$	5.71540e+7	1.6e-6	0.64	1537

since the ℓ_∞ error is lower than 1. Observe in Figure 3.5 that this value separates the nonzero entries from the noise floor (when $\sigma = 0.01$). In the extensive numerical experiments of Section 3.5, we shall set $\mu = 0.02$ and $\delta = 10^{-7}$ as default values.

3.5 Numerical comparisons

This section presents numerical experiments comparing several state-of-the-art optimization techniques designed to solve (3.1.2) or (3.1.3). To be as fair as possible, we propose comparisons with methods for which software is publicly available online.

Some extensive numerical results have been reported in [vdBF08, WNF09, Lor09, LBDM$^+$09], but to the best of our knowledge, extensive tests comparing algorithms for solving (QP_λ) *and* (BP_ϵ) are currently unavailable, and one novelty of our experimental study is that it uses fair stopping criteria introduced in Section 3.5.3. Moreover, whereas publications sometimes test algorithms on relatively easy and academic problems, we will subject optimization methods to hard but realistic ℓ_1 reconstruction problems.

In our view, a challenging problem involves some or all of the characteristics below.

1. *High dynamic range.* As mentioned earlier, most optimization techniques are able to find (more or less rapidly) the most significant entries (those with a large amplitude) of the signal x. Recovering the entries of x that have low magnitudes accurately is more challenging. Methods that can accurately recover high-dynamic range signals is of crucial importance in some real-world applications, see Section 3.6.1.

2. *Approximate sparsity.* Realistic signals are seldom exactly sparse and, therefore, coping with approximately sparse signals is of paramount importance. In signal or image processing for example, wavelet coefficients of natural images contain lots of low level entries that are worth retrieving.

3. *Large scale.* Some standard optimization techniques, such as interior point methods, are known to provide accurate solutions. However, these techniques are not applicable to large-scale problems due to the large cost of solving linear systems. Further, many existing software

packages fail to take advantage of fast-algorithms for applying A. We will focus on large-scale problems in which the number of unknowns n is over a quarter of a million, i.e., $n = 262{,}144$.

3.5.1 State-of-the-art methods

Most of the algorithms discussed in this section are considered to be state-of-art in the sense that they are the most competitive among sparse reconstruction algorithms. Some of these methods have been improved after some years of research [KKB07, FNW07]. Others make careful use of the fact that ℓ_1 minimization produces sparse solutions, and activate special code when they estimate that a fixed active set has been found [vdBF08, WYGZ10]. Finally, our focus is on rapid algorithms so that we are interested in methods which can take advantage of fast algorithms for applying A to a vector. This is why we have not tested other good methods, such as [FHT10], for example.

Most of the following algorithms solve (QP_λ), and require gradient or sub-gradient information at the current iterate. Because of the quadratic penalty, the gradient requires one application of A and one of A^*, so the dominant cost per iteration is $2\mathcal{C}_A$.

3.5.1.1 NESTA

Below, we applied NESTA with the following default parameters

$$x_0 = A^*b, \quad \mu = 0.02, \quad \delta = 10^{-7}$$

(recall that x_0 is the initial guess). The maximal number of iterations is set to $\mathcal{I}_{\max} = 10{,}000$; if convergence is not reached after $\mathcal{I}_{\max}$ iterations, we record that the algorithm did not convergence (DNC). Because NESTA requires 2 calls to either A or A^* per iteration, this is equivalent to declaring DNC after $\mathcal{N}_A = 20{,}000$ iterations where $\mathcal{N}_A$ refers to the total number of calls to A or A^*; hence, for the other methods, we declare DNC when $\mathcal{N}_A > 20{,}000$. When continuation is used, extra parameters are set up as follows:

$$T = 4, \quad \mu_0 = \|x_0\|_{\ell_\infty}, \quad \gamma = (\mu/\mu_0)^{1/T},$$

and for $t = 1, \ldots, T$,

$$\mu_t = \gamma^t \mu_0, \quad \delta_t = 0.1 \cdot (\delta/0.1)^{t/T},$$

where δ_t fixes the stopping criterion in the t-th continuation step. Numerical results are reported and discussed in Section 3.5.4.

3.5.1.2 Gradient projections for sparse reconstruction (GPSR)

GPSR has been introduced in [FNW07] to solve the standard ℓ_1 minimization problem in Lagrangian form (QP_λ). GPSR uses a standard linear programming change-of-variables trick to recast the variable $x = v_1 - v_2$, with the requirement that $v_1, v_2 \geq 0$. This is a different problem, but it clearly has the same solution set. The advantage is that the ℓ_1 norm is replaced by a linear functional, so that the objective is now smooth. Projecting v_1 and v_2 onto the non-negative orthant is trivial. Different techniques for choosing the step-size α_k (backtracking, Barzilai-Borwein [BB88], and so on) are discussed in [FNW07]. The code is available at `http://www.lx.it.pt/~mtf/GPSR/`. In the forthcoming experiments, the parameters are set to their default values.

GPSR also implements continuation, and we test this version as well. All parameters were set to defaults except, per the recommendation of one of the GPSR authors to increase performance, the number of continuation steps was set to 40, the `ToleranceA` variable was set to 10^{-3}, and the `MiniterA` variable was set to 1. In addition, the code itself was tweaked a bit; in particular, the stopping criteria for continuation steps (other than the final step) was changed. Future releases of GPSR will probably contain a similarly updated continuation stopping criteria.

3.5.1.3 Sparse reconstruction by separable approximation (SpaRSA)

SpaRSA [WNF09] is an algorithm to minimize composite functions $\phi(x) = f(x) + \lambda c(x)$ composed of a smooth term f and a separable non-smooth term c, e.g., (QP_λ). At every step, a sub-problem of the form

$$\text{minimize} \quad \|x - y\|_{\ell_2}^2 + \frac{\lambda}{\alpha} c(x)$$

with optimization variable x must be solved; this is the same as computing the proximity operator corresponding to c. For (QP_λ), the solution is given by shrinkage. In this sense, SpaRSA is an iterative shrinkage/thresholding (IST) algorithm, much like FISTA (though without the accelerated convergence) and FPC. Also like FPC, continuation is used to speed convergence. Code for SpaRSA may be obtained at `http://www.lx.it.pt/~mtf/SpaRSA/`. Parameters were set to default except the number of continuation steps was set to 40 and the `MiniterA` variable was set to 1 (instead of the default 5), as per the recommendations of one of the SpaRSA authors—again, to increase performance.

3.5.1.4 ℓ_1 regularized least-squares (l1_ls)

This method [KKB07] solves the standard unconstrained ℓ_1 minimization problem, and is an interior point method (with log-barrier) using preconditioned conjugate gradient (PCG) to accelerate convergence and stabilize the algorithm. The preconditioner used in the PCG step is a linear combination of the diagonal approximation of the Hessian of the quadratic term and of the Hessian of

the log-barrier term. l1_ls is shown to be faster than usual interior point methods; nevertheless, each step requires solving a linear system of the form $H\Delta x = g$. Even if PCG makes the method more reliable, l1_ls is still problematic for large-scale problems. In the next comparisons, we provide some typical values of its computational complexity compared to the other methods. The code is available at `http://www.stanford.edu/~boyd/l1_ls/`.

3.5.1.5 Spectral projected gradient (SPGL1)

In 2008, van den Berg et al. [vdBF08] adapted the spectral projection gradient algorithm introduced in [BMR00] to solve the LASSO (LS_τ). Interestingly, they introduced a clever root finding procedure such that solving a few instances of (LS_τ) for different values of τ enables them to equivalently solve (BP_ϵ). Furthermore, if the algorithm detects a nearly-sparse solution, it defines an active set and solves an equation like (3.4.3) on this active set. In the next experiments, the parameters are set to their default values. The code is available at `http://www.cs.ubc.ca/labs/scl/SPGL11/`.

3.5.1.6 Fixed point continuation method (FPC)

The fixed point continuation method [HYZ07, HYZ08] is a recent first-order algorithm for solving (QP_λ) and simple generalizations of (QP_λ). The main idea is based on a fixed point equation, $x = F(x)$, which holds at the solution (derived from the subgradient optimality condition, where F is a composition of shrinkage and a gradient step). For appropriate parameters, F is a contraction, and thus the algorithm $x_{k+1} = F(x_k)$ converges (q-linearly). The parameter λ in (QP_λ) determines the amount of shrinkage and, therefore, the speed of convergence; thus in practice, λ is decreased in a continuation scheme. Code for FPC is available at `http://www.caam.rice.edu/~optimization/L1/fpc/`. Also available is a state-of-the-art version of FPC from 2008 that uses Barzilai-Borwein [BB88] steps to accelerate performance. In the numerical tests, the Barzilai-Borwein version (referred to as FPC-BB) significantly outperforms standard FPC. All parameters were set to default values.

3.5.1.7 FPC active set (FPC-AS)

In 2009, inspired by both first-order algorithms, such as FPC, and greedy algorithms [DTDS06, NT09], Wen et al. [WYGZ10] extended FPC into the two-part algorithm FPC active set to solve (QP_λ). In the first stage, FPC-AS calls an improved version of FPC that allows the step-size to be updated dynamically, using a non-monotone exact line search to ensure r-linear convergence, and also incorporating a Barzilai-Borwein [BB88] heuristic. After a given stopping criterion, the current value, x_k, is hard-thresholded to determine an active set. On the active set, $\|x\|_{\ell_1}$ is replaced by c^*x, where $c = \text{sgn}(x_k)$, with the constraints that $x[i] \cdot c[i] > 0$ for all the indices i belonging to the active set. This sub-problem has a smooth objective, so it can be solved using smooth optimization techniques. This two-step process is then repeated for a smaller value of λ in a continuation scheme. We tested

FPC-AS using both L-BFGS (the default) and CG (which we refer to as FPC-AS-CG) to solve the sub-problem; both of these solvers do not actually enforce the $x[i] \cdot c[i] > 0$ constraint on the active set. Code for FPC-AS is available at `http://www.caam.rice.edu/~optimization/L1/FPC_AS/`.

For s-sparse signals, all parameters were set to defaults except for the stopping criteria (as discussed in Section 3.5.3). For approximately sparse signals, FPC-AS performed poorly ($> 10,000$ iterations) with the default parameters. By changing a parameter that controls the *estimated* number of nonzeros from $m/2$ (default) to n, the performance improved dramatically, and this is the performance reported in the tables. The maximum number of subspace iterations was also changed from the default to 10, as recommended in the help file.

3.5.1.8 Bregman

The Bregman iterative algorithm, motivated by the Bregman distance, has been shown to be surprisingly simple [YOGD08]. The first iteration solves (QP_λ) for a specified value of λ; subsequent iterations solve (QP_λ) for the same value of λ, with an updated observation vector b. Typically, only a few outer iterations are needed (e.g., 4), but each iteration requires a solve of (QP_λ), which is costly. The original Bregman algorithm calls FPC to solve these sub-problems; we test Bregman using FPC and FPC-BB as sub-problem solvers.

A version of the Bregman algorithm, known as the linearized Bregman algorithm [OMDY10, COS09], takes only one step of the inner iteration per outer iteration; consequently, many outer iterations are taken, in contrast to the regular Bregman algorithm. It can be shown that linearized Bregman is equivalent to gradient ascent on the dual problem. Linearized Bregman was not included in the tests because no standardized public code is available. Code for the regular Bregman algorithm may be obtained at `http://www.caam.rice.edu/~optimization/L1/2006/10/bregman-iterative-algorithms-for.html`. There are quite a few parameters, since there are parameters for the outer iterations and for the inner (FPC) iterations; for all experiments, parameters were set to defaults. In particular, we noted that using the default stopping criteria for the inner solve, which limited FPC to 1,000 iterations, led to significantly better results than allowing the sub-problem to run to 10,000 iterations.

3.5.1.9 Fast iterative soft-thresholding algorithm (FISTA)

FISTA [BT09] is based upon Nesterov's work but departs from NESTA in two important ways: 1) FISTA solves the sparse unconstrained reconstruction problem (QP_λ); 2) FISTA is a proximal subgradient algorithm, which only uses two sequences of iterates. In some sense, FISTA is a simplified version of the algorithm previously introduced by Nesterov to minimize composite functions [Nes07]. The theoretical rate of convergence of FISTA is $\mathcal{O}(1/k^2)$.

For each test, FISTA is run twice: it is first run until the relative variation in the function value

is less than 10^{-14}, with no limit on function calls, and this solution is used as the reference solution. The second time, it is run using the same stopping criteria as the other algorithms, as explained in Section 3.5.3.

3.5.2 Constrained versus unconstrained minimization

We would like to briefly highlight the fact that these algorithms are not solving the same problem. SPGL1 solves the constrained problem (BP_ϵ) and NESTA solves an approximate solution to (BP_ϵ), while all other methods tested solve the unconstrained problem (QP_λ). NESTA and SPGL1 can also solve (QP_λ). Solving a constrained problem may sometimes be more challenging than similar unconstrained problems (witness the popularity of penalty functions and augmented Lagrangian methods), and given the numerous algorithms to solve (QP_λ) and the relatively few algorithms to solve (BP_ϵ), it seems empirically that (BP_ϵ) is the harder problem.[7] For example, it may be hard to even find a feasible point for (BP_ϵ), since the pseudo-inverse of A, when A is not a projection, may be difficult to compute. Thus, we emphasize that SPGL1 and NESTA are actually more general than the other algorithms (and as Section 3.6 shows, NESTA is even more general because it handles a wide variety of constrained problems); this is especially important because from a practical viewpoint, it may be easier to estimate an appropriate ϵ than an appropriate value of λ. Furthermore, as will be shown in Section 3.5.4, SPGL1 and NESTA with continuation are also the most robust methods for arbitrary signals (i.e., they perform well even when the signal is not exactly sparse, and even when it has high dynamic range). Combining these two facts, we feel that these two algorithms are extremely useful for real-world applications.

3.5.3 Experimental protocol

In these experiments, we compare NESTA with other efficient methods. There are two main difficulties with comparisons which might explain why broad comparisons have not been offered before. The first problem is that some algorithms, such as NESTA, solve (BP_ϵ), whereas other algorithms solve (QP_λ). Given ϵ, it is difficult to compute $\lambda(\epsilon)$ that gives an equivalence between the problems; in theory, the KKT conditions give λ, but we have observed in practice that because we have an approximate solution (albeit a very accurate one), computing λ in this fashion is not stable.

Instead, we note that given λ and a solution x_λ to (QP_λ), it is easy to compute a very accurate $\epsilon(\lambda)$ since $\epsilon = \|Ax_\lambda - b\|_{\ell_2}$. Hence, we use a two-step procedure. In the first step, we choose a value of $\epsilon_0 = \sqrt{m + 2\sqrt{2m}\sigma}$ based on the noise level σ (since a value of λ that corresponds to σ is less clear), and use SPGL1 to solve (BP_ϵ). From the SPGL1 dual solution, we have an estimate

[7] Standard texts explain that (BP_ϵ) is equivalent to minimizing $\|x\|_{\ell_1} + \chi_{\mathcal{Q}_p}(x)$ where $\mathcal{Q}_p$ is the feasible set $\{x : \|Ax - b\|_{\ell_2} \leq \epsilon\}$, and $\chi_{\mathcal{Q}_p}(x) = 0$ if $x \in \mathcal{Q}_p$ and $+\infty$ otherwise. Hence, the unconstrained problem has a discontinuous objective functional.

of $\lambda = \lambda(\epsilon_0)$. As noted above, this equivalence may not be very accurate, so the second step is to compute $\epsilon_1 = \epsilon(\lambda)$ via FISTA, using a very high accuracy of $\delta = 10^{-14}$. The pair (λ, ϵ_1) now leads to nearly equivalent solutions of (QP_λ) and (BP_ϵ). The solution from FISTA will also be used to judge the accuracy of the other algorithms.

The other main difficulty in comparisons is a fair stopping criterion. Each algorithm has its own stopping criterion (or may offer a choice of stopping criteria), and these are not directly comparable. To overcome this difficulty, we have modified the codes of the algorithms to allow for two new stopping criterion that we feel are the only fair choices. The short story is that we use NESTA to compute a solution x_N and then ask the other algorithms to compute a solution that is at least as accurate.

Specifically, given NESTA's solution x_N (using continuation), the other algorithms terminate at iteration k when the solution $\hat{x}_k$ satisfies

$$\text{(Crit. 1)} \quad \|\hat{x}_k\|_{\ell_1} \leq \|x_N\|_{\ell_1} \quad \text{and} \quad \|b - A\hat{x}_k\|_{\ell_2} \leq 1.05\,\|b - Ax_N\|_{\ell_2}, \tag{3.5.1}$$

or

$$\text{(Crit. 2)} \quad \lambda\|\hat{x}_k\|_{\ell_1} + \frac{1}{2}\|A\hat{x}_k - b\|_{\ell_2}^2 \leq \lambda\|x_N\|_{\ell_1} + \frac{1}{2}\|Ax_N - b\|_{\ell_2}^2. \tag{3.5.2}$$

We run tests with both stopping criteria to reduce any potential bias from the fact that some algorithms solve (QP_λ), for which Crit. 2 is the most natural, while others solve (BP_ϵ), for which Crit. 1 is the most natural. In practice, the results when applying Crit. 1 or Crit. 2 are not significantly different.

3.5.4 Numerical results

3.5.4.1 The case of exactly sparse signals

This first series of experiments tests all the algorithms discussed above in the case where the unknown signal is s-sparse with $s = m/5$, $m = n/8$, and $n = 262{,}144$. This situation is close to the limit of perfect recovery from noiseless data. The s nonzero entries of the signals x^0 are generated as described in (3.3.15). Reconstruction is performed with several values of the dynamic range $d = 20, 40, 60, 80, 100$ in dB. The measurement operator is a randomly subsampled discrete cosine transform, as in Section 3.4.1 (with a different random set of measurements chosen for each trial). The noise level is set to $\sigma = 0.1$. The results are reported in Tables 3.3 (Crit. 1) and 3.4 (Crit. 2); each cell in these tables contains the mean value of $\mathcal{N}_A$ (the number of calls of A or A^*) over 10 random trials, and, in smaller font, the minimum and maximum value of $\mathcal{N}_A$ over the 10 trials. When convergence is not reached after $\mathcal{N}_A = 20{,}000$, we report DNC (did not converge). As expected, the number of calls needed to reach convergence varies a lot from an algorithm to another.

Table 3.3: Number of function calls $\mathcal{N}_A$ averaged over 10 independent runs. The sparsity level $s = m/5$ and the stopping rule is Crit. 1 (3.5.1).

Method	20 dB	40 dB	60 dB	80 dB	100 dB
NESTA	446 351/491	880 719/951	1701 1581/1777	4528 4031/4749	14647 7729/15991
NESTA + Ct	479 475/485	551 539/559	605 589/619	658 635/679	685 657/705
GPSR	56 44/62	733 680/788	5320 4818/5628	DNC	DNC
GPSR + Ct	305 293/311	251 245/257	497 453/531	1816 1303/2069	9101 7221/10761
SpaRSA	345 327/373	455 435/469	542 511/579	601 563/629	708 667/819
SPGL1	54 37/61	128 102/142	209 190/216	354 297/561	465 380/562
FISTA	68 66/69	270 261/279	935 885/969	3410 2961/3594	13164 11961/13911
FPC AS	156 111/177	236 157/263	218 215/239	351 247/457	325 313/335
FPC AS (CG)	312 212/359	475 301/538	434 423/481	641 470/812	583 567/595
FPC	414 394/436	417 408/422	571 546/594	945 852/1038	3945 2018/4734
FPC-BB	148 140/152	166 158/168	219 208/250	264 252/282	520 320/800
Bregman-BB	211 203/225	270 257/295	364 355/393	470 429/501	572 521/657

Table 3.4: Number of function calls $\mathcal{N}_A$ averaged over 10 independent runs. The sparsity level $s = m/5$ and the stopping rule is Crit. 2 (3.5.2).

Method	20 dB	40 dB	60 dB	80 dB	100 dB
NESTA	446 351/491	880 719/951	1701 1581/1777	4528 4031/4749	14647 7729/15991
NESTA + Ct	479 475/485	551 539/559	605 589/619	658 635/679	685 657/705
GPSR	59 44/64	736 678/790	5316 4814/5630	DNC	DNC
GPSR + Ct	305 293/311	251 245/257	511 467/543	1837 1323/2091	9127 7251/10789
SpaRSA	345 327/373	455 435/469	541 509/579	600 561/629	706 667/819
SPGL1	55 37/61	138 113/152	217 196/233	358 300/576	470 383/568
FISTA	65 63/66	288 279/297	932 882/966	3407 2961/3591	13160 11955/13908
FPC AS	176 169/183	236 157/263	218 215/239	344 247/459	330 319/339
FPC AS (CG)	357 343/371	475 301/538	434 423/481	622 435/814	588 573/599
FPC	416 398/438	435 418/446	577 558/600	899 788/962	3866 1938/4648
FPC-BB	149 140/154	172 164/174	217 208/254	262 248/286	512 308/790
Bregman-BB	211 203/225	270 257/295	364 355/393	470 429/501	572 521/657

The careful reader will notice that Tables 3.3 and 3.4 do not feature the results provided by l1_ls; indeed, while it seems faster than other interior point methods, it is still far from being comparable to the other algorithms reviewed here. In these experiments l1_ls typically needed 1500 calls to A or A^* for reconstructing a 20 dB signal with $s = m/100$ nonzero entries. For solving the same problem with a dynamic range of 100 dB, it took 5 hours to converge on a dual core MacPro G5 clocked at 2.7GHz.

GPSR performs well in the case of low-dynamic range signals; its performance, however, decreases dramatically as the dynamic range increases; Table 3.4 shows that it does not converge for 80 and 100 dB signals. GPSR with continuation does worse on the low dynamic range signals (which is not surprising). It does much better than the regular GPSR version on the high dynamic range signals, though it is slower than NESTA with continuation by more than a factor of 10. SpaRSA performs well at low dynamic range, comparable to NESTA, and begins to outperform GSPR with continuation as the dynamic range increases, although it begins to underperform NESTA with continuation in this regime. SpaRSA takes over twice as many function calls on the 100 dB signal as on the 20 dB signal.

SPGL1 shows good performance with very sparse signals and low dynamic range. Although it has fewer iteration counts than NESTA, the performance decreases much more quickly than for

NESTA as the dynamic range increases; SPGL1 requires about 9× more calls to A at 100 dB than at 20 dB, whereas NESTA with continuation requires only about 1.5× more calls. FISTA is almost as fast as SPGL1 on the low dynamic range signal, but degrades very quickly as the dynamic range increases, taking about 200× more iterations at 100 dB than at 20 dB. One large contributing factor to this poor performance at high dynamic range is the lack of a continuation scheme.

FPC performs well at low dynamic range, but is very slow on 100 dB signals. The Barzilai-Borwein version was consistently faster than the regular version, but also degrades much faster than NESTA with continuation as the dynamic range increases. Both FPC active set and the Bregman algorithm perform well at all dynamic ranges, but again, degrade faster than NESTA with continuation as the dynamic range increases. There is a slight difference between the two FPC active set versions (using L-BFGS or CG), but the dependence on the dynamic range is roughly similar.

The performances of NESTA with continuation are reasonable when the dynamic range is low. When the dynamic range increases, continuation becomes much more helpful. For some 100 dB signals, using continuation reduces the number of calls by a factor of 20. In these experiments, the tolerance δ is consistently equal to 10^{-7}; while this choice is reasonable when the dynamic range is high, it seems too conservative in the low dynamic range case. Setting a lower value of δ should improve NESTA's performance in this regime. In other words, NESTA with continuation might be tweaked to run faster on the low dynamic range signals. However, this is not in the spirit of this chapter and this is why we have not researched further refinements.

In summary, for exactly sparse signals exhibiting a significant dynamic range, 1) the performance of NESTA with continuation—but otherwise applied out-of-the-box—is comparable to that of state-of-the-art algorithms, and 2) most state-of-the-art algorithms are efficient on these types of signals.

3.5.4.2 Approximately sparse signals

We now turn our attention to approximately sparse signals. Such signals are generated via a permutation of the Haar wavelet coefficients of a 512×512 natural image. The data b are $m = n/8 = 32{,}768$ discrete cosine measurements selected at random. White Gaussian noise with standard deviation $\sigma = 0.1$ is then added. Each test is repeated 5 times, using a different random permutation every time (as well as a new instance of the noise vector). Unlike in the exactly sparse case, the wavelet coefficients of natural images mostly contain mid-range and low-level coefficients (see Figure 3.6) which are challenging to recover.

The results are reported in Tables 3.5 (Crit. 1) and 3.6 (Crit. 2); the results from applying the two stopping criteria are nearly identical. In these series of experiments, the performance of SPGL1 is quite good but seems to vary a lot from one trial to another (Table 3.6). Notice that the concept of an active set is ill defined in the approximately sparse case; as a consequence, the active-set version of FPC is not much of an improvement over the regular FPC version. FPC is very fast for

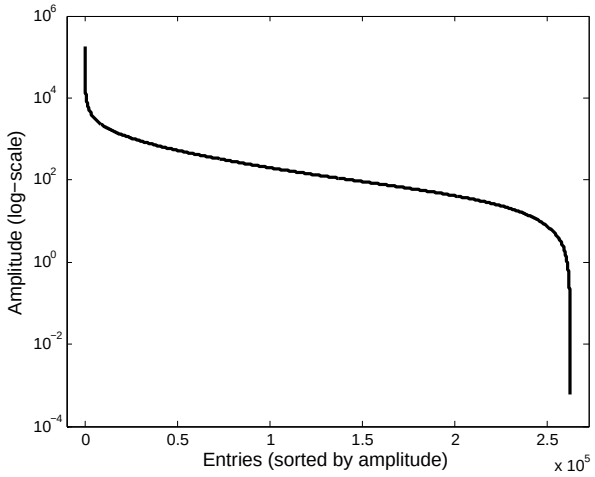


Figure 3.6: Sorted wavelet coefficients of the natural image used in the experiments

s-sparse signals but lacks the robustness to deal with less ideal situations in which the unknown is only approximately sparse.

FISTA and SpaRSA converge for these tests, but are not competitive with the best methods. It is reasonable to assume that FISTA would also improve if implemented with continuation. SpaRSA already uses continuation but does not match its excellent performance on exactly sparse signals.

Bregman, SPGL1, and NESTA with continuation all have excellent performances (continuation really helps NESTA) in this series of experiments. NESTA with continuation seems very robust when high accuracy is required. The main distinguishing feature of NESTA is that it is less sensitive to dynamic range; this means that as the dynamic range increases, or as the noise level σ decreases, NESTA becomes very competitive. For example, when the same test was repeated with more noise ($\sigma = 1$), all the algorithms converged faster. In moving from $\sigma = 1$ to $\sigma = 0.1$, SPGL1 required 90% more iterations and Bregman required 20% more iterations, while NESTA with continuation required only 5% more iterations.

One conclusion from these tests is that SPGL1, Bregman, and NESTA (with continuation) are the only methods dealing with approximately sparse signals effectively. The other methods, most of which did very well on exactly sparse signals, take over 10,000 function calls or even do not converge in 20,000 function calls; by comparison, SPGL1, Bregman, and NESTA with continuation converge in about 2,000 function calls. It is also worth noting that Bregman is only as good as the sub-problem solver; though not reported here, using the regular FPC (instead of FPC-BB) with Bregman leads to much worse performance.

Table 3.5: Recovery results of an approximately sparse signal with Crit. 1 as a stopping rule

Method	$< \mathcal{N}_A >$	$\min \mathcal{N}_A$	$\max \mathcal{N}_A$
NESTA	18912	18773	19115
NESTA + Ct	2667	2603	2713
GPSR	DNC	DNC	DNC
GPSR + Ct	DNC	DNC	DNC
SpaRSA	10019	8369	12409
SPGL1	1776	1073	2464
FISTA	10765	10239	11019
FPC Active Set	DNC	DNC	DNC
FPC Active Set (CG)	DNC	DNC	DNC
FPC	DNC	DNC	DNC
FPC-BB	DNC	DNC	DNC
Bregman-BB	2045	2045	2045

Table 3.6: Recovery results of an approximately sparse signal with Crit. 2 as a stopping rule

Method	$< \mathcal{N}_A >$	$\min \mathcal{N}_A$	$\max \mathcal{N}_A$
NESTA	18912	18773	19115
NESTA + Ct	2667	2603	2713
GPSR	DNC	DNC	DNC
GPSR + Ct	DNC	DNC	DNC
SpaRSA	10021	8353	12439
SPGL1	1776	1073	2464
FISTA	10724	10197	10980
FPC Active Set	DNC	DNC	DNC
FPC Active Set (CG)	DNC	DNC	DNC
FPC	DNC	DNC	DNC
FPC-BB	DNC	DNC	DNC
Bregman-BB	2045	2045	2045

The algorithms which did converge all achieved a mean relative ℓ_1 error (using (3.4.1) and the high accuracy FISTA solution as the reference) less than $2 \cdot 10^{-4}$ and sometimes as low as 10^{-5}, except SPGL1, which had a mean relative error of $1.1 \cdot 10^{-3}$. Of the algorithms that did not converge in 20,000 function calls, FPC and FPC-BB had a mean ℓ_1 relative error about $5 \cdot 10^{-3}$, GPSR with continuation had errors about $5 \cdot 10^{-2}$, and the rest had errors greater than 10^{-1}.

3.6 An all-purpose algorithm

A distinguishing feature is that NESTA is able to cope with a wide range of standard regularizing functions. In this section, we present two examples: non-standard ℓ_1 minimization and total-variation minimization.

3.6.1 Non-standard sparse reconstruction: ℓ_1 analysis

Suppose we have a signal $x \in \mathbb{R}^n$, which is assumed to be approximately sparse in a transformed domain such as the wavelet, the curvelet or the time-frequency domains. Let W be the corresponding

synthesis operator whose columns are the waveforms we use to synthesize the signal $x = W\alpha$ (real-world signals do not admit an exactly sparse expansion); e.g., the columns may be wavelets, curvelets, and so on, or both. We will refer to W^* as the analysis operator. As before, we have (possibly noisy) measurements $b = Ax^0 + z$. The *synthesis* approach attempts reconstruction by solving

$$\begin{aligned} &\text{minimize} && \|\alpha\|_{\ell_1} \\ &\text{subject to} && \|b - AW\alpha\|_{\ell_2} \leq \epsilon, \end{aligned} \tag{3.6.1}$$

while the *analysis* approach solves the related problem

$$\begin{aligned} &\text{minimize} && \|W^*x\|_{\ell_1} \\ &\text{subject to} && \|b - Ax\|_{\ell_2} \leq \epsilon. \end{aligned} \tag{3.6.2}$$

If W is orthonormal, the two problems are equivalent, but in general, these give distinct solutions and current theory explaining the differences is still in its infancy. The article [EMR07] suggests that synthesis may be overly sensitive, and argues with geometric heuristics and numerical simulations that analysis is sometimes preferable.

Solving ℓ_1-analysis problems with NESTA is straightforward as only Step 1 needs to be adapted. We have

$$f_\mu(x) = \max_{u \in Q_p} \langle u, W^*x \rangle - \frac{\mu}{2}\|u\|_{\ell_2}^2,$$

and the gradient at x is equal to

$$\nabla f_\mu(x) = W u_\mu(x);$$

here, $u_\mu(x)$ is given by

$$(u_\mu(x))[i] = \begin{cases} \mu^{-1}(W^*x)[i], & \text{if } |(W^*x)[i]| < \mu, \\ \text{sgn}((W^*x)[i]), & \text{otherwise.} \end{cases}$$

Steps 2 and 3 remain unchanged. The computational complexity of the algorithm is then increased by an extra term, namely $2\,\mathcal{C}_W$ where $\mathcal{C}_W$ is the cost of applying W or W^* to a vector. In practical situations, there is often a fast algorithm for applying W and W^*, e.g. a fast wavelet transform [Mal08], a fast curvelet transform [CDDY06], a fast short-time Fourier transform [Mal08] and so on, which makes this a low-cost extra step[8].

[8] The ability to solve the analysis problem also means that NESTA can easily solve reweighted ℓ_1 problems [CWB08] with no change to the code.

3.6.2 Numerical results for non-standard ℓ_1 minimization

Because NESTA is one of very few algorithms that can solve both the analysis and synthesis problems efficiently, we tested the performance of both analysis and synthesis on a simulated real-world signal from the field of radar detection. The test input is a superposition of three signals. The first signal, which is intended to make recovery more difficult for any smaller signals, is a plain sinusoid with amplitude of 1000 and frequency near 835 MHz.

A second signal, similar to a Doppler pulse radar, is at a carrier frequency of 2.33 GHz with maximum amplitude of 10, a pulse width of 1 μs and a pulse repetition interval of 10 μs; the pulse envelope is trapezoidal, with a 10 ns rise time and 40 ns fall time. This signal is more than 40 dB lower than the pure sinusoid, since the maximum amplitude is 100$\times$ smaller, and since the radar is nonzero only 10% of the time. The Doppler pulse was chosen to be roughly similar to a realistic weather Doppler radar. In practice, these systems operate at 5 cm or 10 cm wavelengths (i.e., 6 or 3 GHz) and send out short trapezoidal pulses to measure the radial velocity of water droplets in the atmosphere using the Doppler effect.

The third signal, which is the signal of interest, is a frequency-hopping radar pulse with maximum amplitude of 1 (so about 20 dB beneath the Doppler signal, and more than 60 dB below the sinusoid). For each instance of the pulse, the frequency is chosen uniformly at random from the range 200 MHz to 2.4 GHz. The pulse duration is 2 μs and the pulse repetition interval is 22 μs, which means that some, but not all, pulses overlap with the Doppler radar pulses. The rise time and fall time of the pulse envelope are comparable to the Doppler pulse. Frequency-hopping signals may arise in applications because they can be more robust to interference and because they can be harder to intercept. When the carrier frequencies are not known to the listener, the receiver must be designed to cover the entire range of possible frequencies (2.2 GHz in our case). While some current analog-to-digital converters (ADC) may be capable of operating at 2.2 GHz, they do so at the expense of low precision. Hence this situation may be particularly amenable to a compressed sensing setup by using several slower (but accurate) ADC to cover a large bandwidth.

We consider the exact signal to be the result of an infinite-precision ADC operating at 5 GHz, which corresponds to the Nyquist rate for signals with 2.5 GHz of bandwidth. Measurements are taken using an orthogonal Hadamard transform with randomly permuted columns, and these measurements were subsequently sub-sampled by randomly choosing $m = .3n$ rows of the transform (so that we undersample Nyquist by 10/3). Samples are recorded for $T = 209.7\mu$s, which corresponds to $n = 2^{20}$. White noise was added to the measurements to make a 60 dB signal-to-noise ratio (SNR) (note that the effective SNR for the frequency-hopping pulse is much lower). The frequencies of the sinusoid and the Doppler radar were chosen such that they were not integer multiples of the lowest recoverable frequency $f_{min} = 1/(2T)$.

For reconstruction, the signal is analyzed with a tight frame of Gabor atoms that is approxi-

mately 5.5× overcomplete. The particular parameters of the frame are chosen to give reasonable reconstruction, but were not tweaked excessively. It is likely that differences in performance between analysis and synthesis are heavily dependent on the particular dictionary.

To analyze performance, we restrict our attention to the frequency domain in order to simplify comparisons. The top plot in Figure 3.7 shows the frequency components of the original, noiseless signal. The frequency-hopping pulse barely shows up since the amplitude is 1000× smaller than the sinusoid and since each frequency only occurs for 1 μs (of 210 μs total).

The bottom plots in Figure 3.7 show the spectrum of the recovered signal using analysis and synthesis, respectively. For this test, analysis does a better job at finding the frequencies belonging to the small pulse, while synthesis does a better job recreating the large pulse and the pure tone. The two reconstructions used slightly different values of μ to account for the redundancy in the size of the dictionary; otherwise, algorithm parameters were the same. In the analysis problem, NESTA took 231 calls to the analysis/synthesis operator (and 231 calls to the Hadamard transform); for synthesis, NESTA took 1378 calls to the analysis/synthesis operator (and 1378 to the Hadamard transform). With NESTA, synthesis is more computationally expensive than analysis since no change of variables trick can be done; in the synthesis case, W and W^* are used in Step 2 and 3 while in the analysis case, the same operators are used once in Step 1 (this is accomplished by the previously mentioned change-of-variables for partial orthogonal measurements).

As emphasized in [EMR07], when W is overcomplete, the solution computed by solving the analysis problems is likely to be denser than in the synthesis case. In plain English, the analysis solution may seem "noisier" than the synthesis solution. But the compactness of the solution of the synthesis problem may also be its weakness: an error on one entry of α may lead to a solution that differs a lot. This may explain why the frequency-hopping radar pulse is harder to recover with the synthesis prior.

Because all other known first-order methods solve only the synthesis problem, NESTA may prove to be extremely useful for real-world applications. Indeed, this simple test suggests that analysis may sometimes be much preferable to synthesis, and given a signal with 2^{20} samples (too large for interior point methods), we know of no other algorithm that can return the same results.

3.6.3 Total-variation minimization

Nesterov's framework also makes total-variation minimization possible. The TV norm of a 2D digital object $x[i,j]$ is given by

$$\|x\|_{TV} := \sum_{i,j} \|\nabla x[i,j]\|, \quad \nabla x[i,j] = \begin{bmatrix} (D_1 x)[i,j] \\ (D_2 x)[i,j] \end{bmatrix},$$

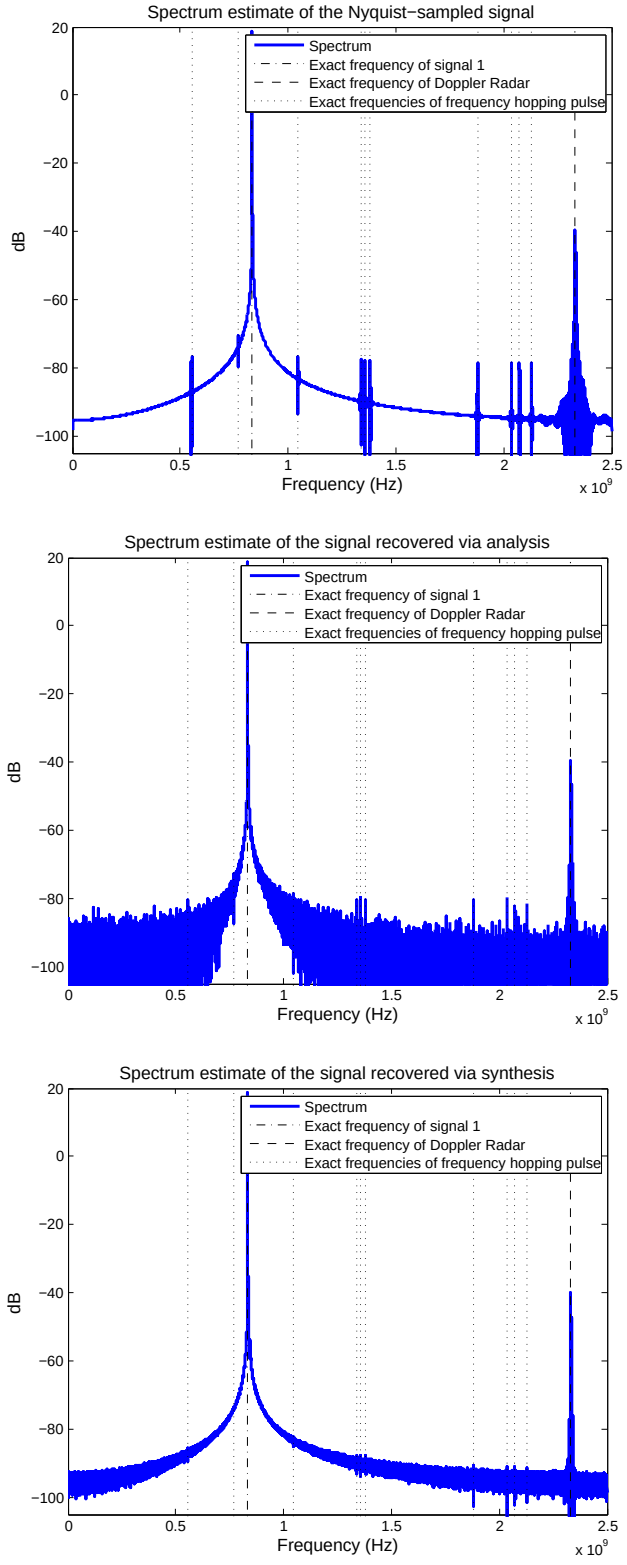


Figure 3.7: **Top:** spectrum estimate of the exact signal, no noise. The pure tone at 60 dB and the Doppler radar at 20 dB dominate the 0 dB frequency-hopping pulses. **Middle:** spectrum estimate of the recovered signal using analysis prior, with 60 dB SNR. The spectrum appears noisy, but the frequency-hopping pulses stand out. **Bottom:** spectrum estimate of the recovered signal using synthesis prior, with 60 dB SNR. The spectrum appears cleaner, but the small 0 dB pulses do not appear.

where D_1 and D_2 are the horizontal and vertical differences

$$(D_1 x)[i,j] = x[i+1,j] - x[i,j],$$
$$(D_2 x)[i,j] = x[i,j+1] - x[i,j].$$

Now the TV norm can be expressed as follows:

$$\|x\|_{TV} = \max_{u \in \mathcal{Q}_d} \langle u, Dx \rangle, \tag{3.6.3}$$

where $u = [u_1, u_2]^* \in \mathcal{Q}_d$ if and only for each (i,j), $u_1^2[i,j] + u_2^2[i,j] \leq 1$, and $D = [D_1, D_2]^*$. The key feature of Nesterov's work is to smooth a well-structured non-smooth function as follows (notice in (3.6.3) the similarity between the TV norm and the ℓ_1 norm):

$$\max_{u \in \mathcal{Q}_d} \langle u, Dx \rangle - \mu p_d(u).$$

Choosing $p_d(u) = \frac{1}{2}\|u\|_{\ell_2}^2$ provides a reasonable prox-function that eases the computation of ∇f_μ. Just as before, changing the regularizing function only modifies Step 1 of NESTA. Here,

$$f_\mu(x) = \max_{u \in \mathcal{Q}_d} \langle u, Dx \rangle - \frac{\mu}{2}\|u\|_{\ell_2}^2.$$

Then as usual,

$$\nabla f_\mu(x) = D^* u^{(\mu)}(x),$$

where $u^{(\mu)}(x)$ is of the form $[u_1^{(\mu)}, u_2^{(\mu)}]^*$ and for each $a \in \{1,2\}$,

$$u_a^{(\mu)}[i,j] = \begin{cases} \mu^{-1}(D_a x)[i,j], & \text{if } \|\nabla x[i,j]\| < \mu, \\ \|\nabla x[i,j]\|^{-1}(D_a x)[i,j], & \text{otherwise.} \end{cases}$$

The application of D and D^* leads to a negligible computational cost (sparse matrix-vector multiplications).

3.6.4 Numerical results for TV minimization

We are interested in solving

$$\begin{array}{ll} \text{minimize} & \|x\|_{TV} \\ \text{subject to} & \|b - Ax\|_{\ell_2} \leq \epsilon. \end{array} \tag{3.6.4}$$

To be sure, a number of efficient TV-minimization algorithms have been proposed to solve (3.6.4) in the special case $A = I$ (denoising problem), see [Cha04,DS05,GO09,ZC08]. In comparison, only a few

methods have been proposed to solve the more general problem (3.6.4) even when A is a projector. Known methods include interior point methods (ℓ_1-magic) [CR07b], proximal-subgradient methods [BDF07, CP08], Split-Bregman [GO09], and the very recently introduced RecPF[9] [YZY10], which operates in the special case of partial Fourier measurements. Applications of some of Nesterov's works have also been considered in different settings [WBFA09, DHJJ10, Auj09].

Again, we compare NESTA with algorithms with publicly available implementations. Therefore, the two algorithms we tested are RecPF (the newest member of the family) and TwIST [BDF07]. (ℓ_1-magic is based on an interior point method, and is not yet applicable to this large-scale problem.)

Evaluations are made by comparing the performances of NESTA (with continuation), RecPF, and TwIST on a set of images composed of random squares. As in Section 3.5, the dynamic range of the signals (amplitude of the squares) varies in a range from 20 to 40 dB. The size of each image x is 1024×1024; one of these images is displayed in the top panel of Figure 3.8. The data b are partial Fourier measurements as in [CRT06]; the number of measurements $m = n/10$. White Gaussian noise of standard deviation $\sigma = 0.1$ is added. The parameters of NESTA are set up as follows:

$$x_0 = A^* b, \quad \mu = 0.2, \quad \delta = 10^{-5}, \quad T = 5,$$

and the initial value of μ is

$$\mu_0 = 0.9 \max_{ij} \|\nabla x_0[i,j]\|.$$

The maximal number of iterations is set to $\mathcal{I}_{\max} = 4{,}000$. As it turns out, TV minimization from partial Fourier measurements is of significant interest in the field of magnetic resonance imaging (MRI) [LDP07].

As discussed above, RecPF has been designed to solve TV-minimization reconstruction problems from partial Fourier measurements. We set the parameters of RecPF to their default values except for the parameter `tol_rel_inn` that is set to 10^{-5}. TwIST calls Chambolle's algorithm [Cha04] at each iteration to compute the proximity operator of TV; the maximum number of iterations in Chambolle's algorithm is set to 10. With these parameter selections, TwIST and RecPF converge to a solution close enough to NESTA's output—at least in our experiments. Figure 3.8 shows the the solution computed by RecPF (top right), TwIST (bottom left), and NESTA (bottom left).

The curves in Figure 3.9 show the number of calls to A or A^*; mid-points are averages over 5 random trials, with error bars indicating the minimum and maximum number of calls. Here, RecPF

[9] Available at `http://www.caam.rice.edu/~optimization/L1/RecPF/`

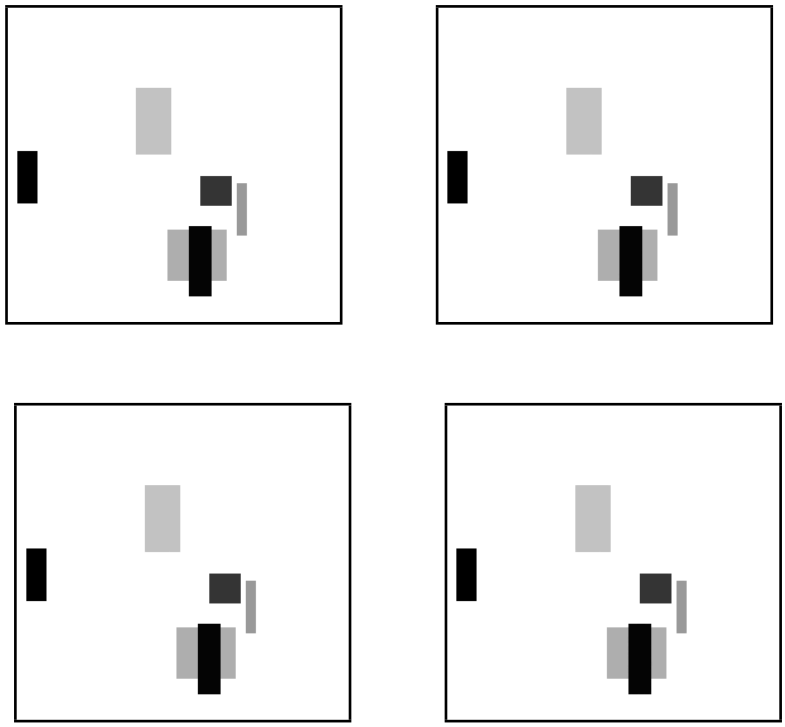


Figure 3.8: Top-Left: original image of size 1024×1024 with a dynamic range of about 40 dB. Top-Right: RecPF solution. Bottom-Left: TwIST solution. Bottom-Right: NESTA solution

is stopped when

$$\|x_{\text{RecPF}}\|_{TV} \leq 1.05\|x_N\|_{TV},$$
$$\|b - Ax_{\text{RecPF}}\|_{\ell_2} \leq 1.05\|b - Ax_N\|_{\ell_2},$$

where x_N is the solution computed via NESTA. As before continuation is very efficient when the dynamic range is high (typically higher than 40 dB). An interesting feature is that the numbers of calls are very similar over all five trials. When the dynamic range increases, the computational costs of both NESTA and RecPF naturally increase. Note that in the 60 and 80 dB experiments, RecPF did not converge to the solution and this is the reason why the number of calls saturates. While both methods have a similar computational cost in the low-dynamic range regime, NESTA has a clear advantage in the higher-dynamic range regime. Moreover, the number of iterations needed to reach convergence with NESTA with continuation is fairly low—300–400 calls to A and A^*—and so this algorithm is well suited to large-scale problems.

Figure 3.9 suggests that TwIST performs well, in terms of function calls, on the problems with low dynamic range, but deteriorates as the dynamic range increases. However, this hides a fundamental complication of the TwIST algorithm: the function calls to A and A^* are not always the dominant computational cost. TwIST is one of many TV methods that is a proximal gradient algorithm,

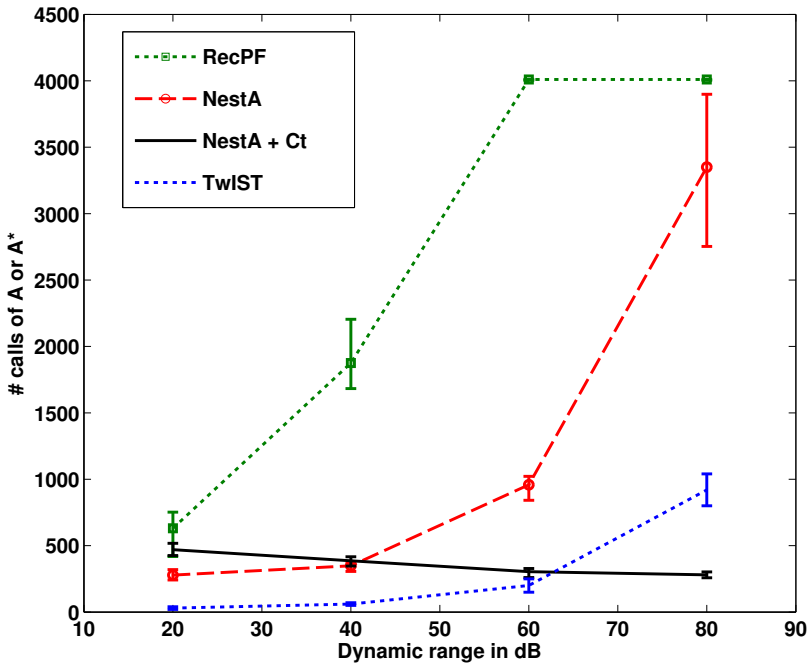


Figure 3.9: Number of calls to A and A^* as a function of the dynamic range. Solid line: NESTA with continuation. Dashed line: NESTA. Dotted line: RecPF. Dash-dotted: maximum number of iterations. In the 60 and 80 dB experiments, RecPF did not converge to the solution and this is the reason why the number of calls saturates. For TwIST, it is important not to take the number of function calls to A and A^* as a proxy for the computational time of the algorithm. Indeed, the dominant computational cost in TwIST is in evaluating the proximity operator, see Table 3.7.

Table 3.7: Comparison of computation times, mean value over 5 trials (in seconds)

Method	20dB	40dB	60dB	80dB
NESTA + Ct	750	704	553	542
RecPF	725	1918	DNC	DNC
TwIST	203	553	1675	9510

meaning that at each step, it relies on the solution to a proximity operator:

$$\text{Prox}_{TV,\gamma}(z) = \underset{x}{\text{argmin}}\, \gamma\|x\|_{TV} + \frac{1}{2}\|x - z\|_{\ell_2}^2;$$

see [CW05] and references therein. Evaluating the proximity operator at z is equivalent to solving a TV-denoising problem, which is far from trivial[10]. In [BDF07], the authors advocate the use of a side algorithm (for instance Chambolle's algorithm [Cha04]) to do this.

There are a few issues with this approach. The first is that side algorithms depend on various parameters, which adds complexity. The second is that these denoising algorithms are computationally demanding, which makes them hard to apply to large-scale problems. To illustrate this phenomenon, we compared the computation times of NESTA + Ct, RecPF, and TwIST on the same kind of synthetic 1024×1024 images. The results are reported in Table 3.7. Each line displays the computation time before convergence is reached. We observe that while TwIST needs fewer calls to A or A^* than NESTA when the dynamic range is low, the computational cost of each step is higher (and increases as the problem size increases) because the dominant computational cost in TwIST is in evaluating the proximity operator.

3.7 Handling non-projectors

Recall the derivation in §3.3.2, which assumed that $AA^* = I$. This section relaxes that assumption. The case $\epsilon = 0$ is simpler, and we offer a few methods. If $\epsilon > 0$, the only method seems to require knowledge of the SVD decomposition of A: $A = U\Sigma V^*$. We also cover the $AA^* = I$ case in more detail, since the author has received several inquiries about this.

[10] For ℓ_1 minimization, the equivalent proximity operator is given by by shrinkage, and can be solved in linear time. In the case of the weighted ℓ_1 problem, using $\|Wx\|_1$ in the objective, the proximity operator is not known in closed form unless W is diagonal. Since most first-order ℓ_1 algorithms rely on the proximity operator, this is the reason that no other algorithms can solve the analysis problem, in contrast to NESTA.

3.7.1 Revisiting the projector case

To compute y_k, we need to solve a problem of the form (with $q = d - L^{-1}c$)

$$\left(I + \frac{\lambda_\epsilon}{L} A^* A\right) y_k = \frac{\lambda_\epsilon}{L} A^* b + q \tag{3.7.1}$$

which has solution (3.3.10)

$$y_k = \left(I - \frac{\lambda_\epsilon}{\lambda_\epsilon + L} A^* A\right) \left(\frac{\lambda_\epsilon}{L} A^* b + qc\right). \tag{3.7.2}$$

To see this, it is possible to simply check that this solution works. For deriving it, one method is using the binomial inverse theorem (equivalently, the Sherman-Morrison-Woodbury formula). For matrices $U, V^* \in \mathbb{R}^{p \times q}$, with I_p and I_q the $p \times p$ and $q \times q$ identity matrices, respectively, the inverse theorem states

$$(I_p + UV)^{-1} = I_p - U(I_q + VU)^{-1}V.$$

Unless $\lambda_\epsilon = 0$, by complementary slackness it holds that $\|Ay_k - b\| = \epsilon$, so plugging in y_k and using $AA^* = I$ to simplify gives the correct value of λ_ϵ to make this hold.

3.7.2 Non-projectors for $\epsilon = 0$ case

Consider the problem in a more abstract setting. The desire is to project onto $\mathcal{Q}_p$, which is equivalent to

$$y_k = \underset{y : \|Ay - b\|_2 \leq \epsilon}{\operatorname{argmin}} \frac{1}{2}\|y - q\|_2^2 \tag{3.7.3}$$

for $q = d - L^{-1}c$.

When $\epsilon = 0$, the constraint is simply $Ay = b$. There are only two KKT conditions (now, λ is a vector, unlike λ_ϵ):

$$0 = (y - q) + A^*\lambda \qquad \text{(stationarity)} \tag{3.7.4}$$

$$b = Ay \qquad \text{(primal feasibility)}\,. \tag{3.7.5}$$

Multiplying the first equation by A gives

$$0 = b - Aq + AA^*\lambda.$$

The key computation is inverting AA^* to find λ, since once λ is found it is easy to get y.

In practice, this means the user must specify a method of calculating $(AA^*)^{-1}$. If A is large and has a fast multiply, then the conjugate-gradient algorithm is a natural choice. If A is small enough

that it can be factored, then a Cholesky factorization $RR^* = AA^*$ can be computed once at the beginning of the algorithm, producing the triangular matrix R, and subsequently $(AA^*)^{-1}(b - Aq)$ can be calculated by two back-substitution operations. The Cholesky factorization exists as long as R has full row-rank so that AA^* is positive definite. The one-time factorization cost is $\mathcal{O}(m^3)$ and the back-substitution cost is $\mathcal{O}(m^2)$ which is even less than the cost of one dense matrix-vector multiplication ($\mathcal{O}(mn)$).

An entirely different method is to take the system of equations $Ax = b$ and convert to the equivalent system (assuming A has full row-rank) $\tilde{A}x = \tilde{b}$ where $\tilde{A}\tilde{A}^* = I$. To see how this is possible, take the QR factorization $QR = A^*$ to get an orthogonal matrix Q and a triangular matrix R. Then $\tilde{A} = Q^*$ and $\tilde{b} = R^{-*}b$ satisfy $\tilde{A}x = \tilde{b}$. It is possible to do this with $\epsilon > 0$ constraints as well, but this will affect the properties of the residual, so caution must be used. The cost of the QR decomposition is $\mathcal{O}(mn^2)$ so there is not much advantage over using the SVD method described in the next subsection.

3.7.3 Non-projectors for $\epsilon > 0$ case

If there is a feasible primal vector, then Slater's conditions tell us that there is a Lagrange multiplier scalar λ_ϵ such that (3.7.3) is equivalent to the following unconstrained minimization:

$$y_k = \operatorname*{argmin}_y \frac{1}{2}\|y - q\|_2^2 + \frac{\lambda_\epsilon}{2}\|Ay - b\|_2^2. \tag{3.7.6}$$

Let $A = U\Sigma V^*$ be the full singular value decomposition (SVD) of A, so that U and V are orthogonal (or unitary in the complex-valued case) matrices. The ℓ_2 norm is unitary-invariant, so $\|Ay - b\| = \|U^*(Ay - b)\|$. Make the change-of-variables $x = V^*(y - q)$ and $\tilde{b} = U^*(b - Ay)$, so the problem is

$$\min_x \frac{1}{2}\|x\|^2 + \frac{\lambda_\epsilon}{2}\|\Sigma x - \tilde{b}\|^2.$$

Since Σ is a diagonal matrix (with diagonal $(\sigma_1, \ldots, \sigma_r)$ where $r = \min(m, n)$), this problem is separable, and the solution is easy to compute since this amounts to solving a scalar quadratic equation:

$$(x_{\lambda_\epsilon})_i = \frac{\lambda_\epsilon \sigma_i \tilde{b}_i}{1 + \lambda_\epsilon \sigma_i^2}, \quad i = 1, \ldots, n.$$

The issue is that the value of λ_ϵ is unknown. It should be chosen so that

$$\|Ay - b\|_2 = \|\Sigma x_{\lambda_\epsilon} - \tilde{b}\| = \epsilon.$$

For a given λ, define

$$h(\lambda) = \|\Sigma x_\lambda - \tilde{b}\|^2 - \epsilon^2 = \sum_{i=1}^{n} \left(\sigma_i \frac{\lambda \sigma_i b_i}{1+\lambda\sigma_i^2} - b_i\right)^2 - \epsilon^2 = \sum_{i=1}^{n} \frac{b_i^2}{(1+\lambda\sigma_i^2)^2} - \epsilon^2.$$

Thus we search for a root λ_ϵ of h. The derivative of h is smooth and easy to compute:

$$\frac{dh}{d\lambda}(\lambda) = -2\sum_{i=1}^{n} \frac{(b_i\sigma_i)^2}{(1+\lambda\sigma_i^2)^3}$$

so the natural choice is Newton's method. In a sense, this is a simplified version of the SPGL1 algorithm. The Newton's method is well-posed in this case, since h is a non-increasing function of λ so it can have at most one root.

In the NESTA code, we use a safe-guarded Newton method that does a brief check to keep $\lambda \geq 0$ at every inner iteration. It is possible that $\lambda_\epsilon = 0$ which signifies that $q \in \mathcal{Q}_p$ already so $y = q$ is feasible. During the Newton's method algorithm, all the computations are $\mathcal{O}(n)$ so it is relatively fast. Furthermore, since A, b, and ϵ are constants, and q changes only slightly from iteration to iteration, the value of λ from the previous iteration can be used to warm-start the new Newton's method search. Except for pathological test cases, this extra optimization problem adds no significant running time to NESTA. Similar approaches that use Newton's method to solve the projection have been independently proposed in [DHJJ10] and [WBFA09]. These papers discuss the TV minimization problem and, like NESTA, are based off [Nes05].

In practice, the full SVD is not needed; a "thin" or "skinny" SVD is sufficient. If A does not have full row rank, then b need not be in the range of A. To deal with this, split b into two parts, $b = b_R + b_N$ where $b_R = UU^*b$ is in the range of A and thus $b_N = b - b_R$ is in the null space of A^*. Then the ϵ and b terms are updated

$$\epsilon \leftarrow \sqrt{\epsilon^2 - \|b_N\|^2}, \quad b \leftarrow b_R$$

and the algorithm is run as before.

These improvements have been included in the version "1.1" release of NESTA, and are used to solve the RMPI problems discussed in Chapter 2. For the RMPI problem, W is an overcomplete Gabor dictionary which is costly to compute; on the other-hand, the sensing matrix A (called Φ in the RMPI chapter) is of modest size, such as 80×1024. The largest matrices tested in RMPI have been 8192 columns, and even this is manageable (see Figure 3.10), since the FFT calls in the Gabor dictionary will dominate. So the NESTA algorithm is extremely well suited to the largescale analysis problems that arise in RMPI; indeed, this was a motivation for creating NESTA.

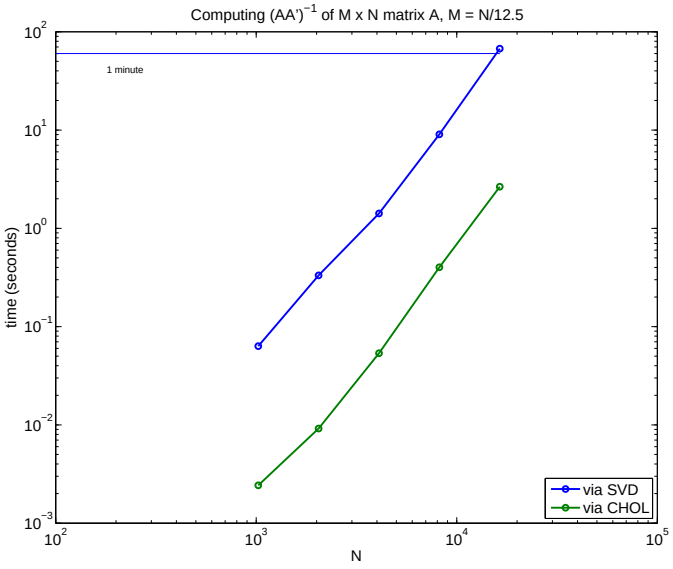


Figure 3.10: The computation time for calculating the Cholesky decomposition and thin SVD decomposition of a matrix, as a function of the columns n of the matrix. Computed in MATLAB on an Intel i7 laptop. This computation needs only be done once per matrix A, so it can be amortized over all iterations or even all experiments, which is the case if multiple problems use the same A matrix (with perhaps different b or ϵ parameters). Working with a matrix of 10000 columns is quite reasonable, though obviously this method scales poorly if the dimensions get really large since both SVD and Cholesky scale at $\mathcal{O}(m^2 n)$ and $\mathcal{O}(m^3)$, respectively. For this plot, $m = n/12.5$ which is the same ratio used in the RMPI device of Chapter 2.

3.8 Discussion

In this chapter, we have proposed an algorithm for general sparse recovery problems, which is based on Nesterov's method. This algorithm is accurate and competitive with state-of-the-art alternatives. In fact, in applications of greatest interest such as the recovery of approximately sparse signals, it outperforms most of the existing methods we have used in our comparisons and is comparable to the best. Further, what is interesting here, is that we have not attempted to optimize the algorithm in any way. For instance, we have not optimized the parameters $\{\alpha_k\}$ and $\{\tau_k\}$, or the number of continuation steps as a function of the desired accuracy δ, and so it is expected that finer tuning would speed up the algorithm. Another advantage is that NESTA is extremely flexible in the sense that minor adaptations lead to efficient algorithms for a host of optimization problems that are crucial in the field of signal/image processing.

3.8.1 Extensions

This chapter focused on the situation in which A^*A is a projector (the rows of A are orthonormal). This stems from the facts that 1) the most computationally friendly compressed sensing are of this form, and 2) it allows fast computations of the two sequence of iterates $\{y_k\}$ and $\{z_k\}$. It is important, however, to extend NESTA as to be able to cope with a wider range of problem in which A^*A is not a projection (or not diagonal).

In order to do this, observe that in Steps 2 and 3, we need to solve problems of the form

$$y_k = \underset{x \in \mathcal{Q}_p}{\operatorname{argmin}} \|x - q\|_{\ell_2}^2,$$

for some q, and we have seen that the solution is given by $y_k = \mathcal{P}_{\mathcal{Q}_p}(q)$, where $\mathcal{P}_{\mathcal{Q}_p}$ is the projector onto $\mathcal{Q}_p := \{x : \|Ax - b\|_{\ell_2} \leq \epsilon\}$. The solution is given by

$$y_k = (I + \lambda A^* A)^{-1}(q + \lambda A^* b) \tag{3.8.1}$$

for some $\lambda \geq 0$. When the eigenvalues of A^*A are well clustered, the right-hand side of (3.8.1) can be computed very efficiently via a few conjugate gradients (CG) steps. Note that this is of direct interest in compressed sensing applications in which A is a random matrix since in all the cases we are familiar with, the eigenvalues of A^*A are tightly clustered. Hence, NESTA may be extended to general problems while retaining its efficiency, with the proviso that a good rule for selecting λ in (3.8.1) is available; i.e., such that $\|Ay_k - b\|_{\ell_2} = \epsilon$ unless $q \in \mathcal{Q}_p$. Of course, one can always eliminate the problem of finding such a λ by solving the unconstrained problem (QP_λ) instead of (BP_ϵ). In this case, each NESTA iteration is actually very cheap, no matter what A looks like.

It is worth noting that if A is relatively small, so that is computationally possible to perform a one-time singular value decomposition of A (this may be the case if A is not too large, but the analysis dictionary W is very overcomplete and the bottleneck for computation), then the inversion just requires two matrix multiplies. The correct value of λ is given by a scalar equation that may be solved efficiently by Newton's method (see also [WBFA09] for a similar approach to selecting λ). This has been discussed in detail in §3.7.

Finally, we also observe that Nesterov's framework is likely to provide efficient algorithms for related problems, which do not have the special $\ell_1 + \ell_2^2$ structure. One example might be the Dantzig selector, which is a convenient and flexible estimator for recovering sparse signals from noisy data [CT07a]:

$$\begin{array}{ll} \text{minimize} & \|x\|_{\ell_1} \\ \text{subject to} & \|A^*(b - Ax)\|_{\ell_\infty} \leq \delta. \end{array} \tag{3.8.2}$$

This is of course equivalent to the unconstrained problem

$$\text{minimize} \quad \lambda\|x\|_{\ell_1} + \|A^*(b - Ax)\|_{\ell_\infty}$$

for some value of λ. Clearly, one could apply Nesterov's smoothing techniques to smooth both terms in the objective functional together with Nesterov's accelerated gradient techniques, and derive a novel and efficient algorithm for computing the solution to the Dantzig selector. This is an example among many others. Another might be the minimization of a sum of two norms, e.g., an ℓ_1 and a

TV norm, under data constraints.

3.8.2 Software

In the spirit of reproducible research [DMR+09], a Matlab version of NESTA is available at: `http://www.acm.caltech.edu/~nesta/`.

Acknowledgements

The authors are grateful to the anonymous reviewers for helpful comments and for pointing out [EMZ07, Lor09, LBDM+09]. S. Becker wishes to thank Peter Stobbe for the use of his Hadamard Transform and Gabor frame code, and Wotao Yin for helpful discussions about RecPF. J. Bobin wishes to thank Hamza Fawzi for fruitful discussions, and E. Candès would like to thank Jalal Fadili for his suggestions. We are grateful to Stephen Wright for his comments on an earlier version of the paper that this chapter is based on, for suggesting use of a better version of GPSR, and encouraging us to test SpaRSA. Thanks Stephen!

Chapter 4

TFOCS

Most of this chapter is part of a journal article submission "Templates for Convex Cone Problems with Applications to Sparse Signal Recover" [BCG10a], and was jointly written with Emmanuel Candès and Michael Grant. An accompanying software package called "Templates for First-Order Conic Solvers" (TFOCS) is available for download [BCG10b].

Parts of this chapter are new and do not appear in the journal version. The section §4.6 re-derives the dual problem in a dual function framework, as opposed to a dual cone framework. The end algorithm is the same, but the new formulation has the benefit of making certain requirements very clear. This also allows for certain new convergence results, which are presented in §4.6.4. The appendix §4.12 has been rewritten and also incorporates work in the literature which very recently appeared. A section on extensions, §4.8, considers a method for automatically determining the restart parameter (§4.8.1), and includes discussion (§4.8.2) on the special problems of noiseless basis pursuit, conic programs in standard form, and matrix completion.

This chapter develops a general framework for solving a variety of convex cone problems that frequently arise in signal processing, machine learning, statistics, and other fields. The approach works as follows: first, determine a conic formulation of the problem; second, determine its dual; third, apply smoothing; and fourth, solve using an optimal first-order method. A merit of this approach is its flexibility: for example, all compressed sensing problems can be solved via this approach. These include models with objective functionals such as the total-variation norm, $\|Wx\|_1$ where W is arbitrary, or a combination thereof. In addition, the chapter introduces a number of technical contributions such as a novel continuation scheme and a novel approach for controlling the step size, and applies results showing that the smooth and unsmoothed problems are sometimes formally equivalent. Combined with our framework, these lead to novel, stable, and computationally efficient algorithms. For instance, our general implementation is competitive with state-of-the-art methods for solving intensively studied problems such as the LASSO. Further, numerical experiments show that one can solve the Dantzig selector problem, for which no efficient large-scale solvers exist, in a few hundred iterations. Finally, the chapter is accompanied with a software release. This

software is not a single, monolithic solver; rather, it is a suite of programs and routines designed to serve as building blocks for constructing complete algorithms.

4.1 Introduction

4.1.1 Motivation

This chapter establishes a general framework for constructing optimal first-order methods for solving certain types of convex optimization programs that frequently arise in signal and image processing, statistics, computer vision, and a variety of other fields.[1] In particular, we wish to recover an unknown vector $x_0 \in \mathbb{R}^n$ from the data $y \in \mathbb{R}^m$ and the model

$$y = Ax_0 + z; \tag{4.1.1}$$

here, A is a known $m \times n$ design matrix and z is a noise term. To fix ideas, suppose we find ourselves in the increasingly common situation where there are fewer observations/measurements than unknowns, i.e., $m < n$. While this may seem *a priori* hopeless, an impressive body of recent works has shown that accurate estimation is often possible under reasonable sparsity constraints on x_0. One practically and theoretically effective estimator is the Dantzig selector introduced in [CT07a]. The idea of this procedure is rather simple: find the estimate which is consistent with the observed data and has minimum ℓ_1 norm (thus promoting sparsity). Formally, assuming that the columns of A are normalized,[2] the Dantzig selector is the solution to the convex program

$$\begin{array}{ll} \text{minimize} & \|x\|_1 \\ \text{subject to} & \|A^*(y - Ax)\|_\infty \leq \delta, \end{array} \tag{4.1.2}$$

where δ is a scalar. Clearly, the constraint is a data fitting term since it asks that the correlation between the residual vector $r = y - Ax$ and the columns of A is small. Typically, the scalar δ is adjusted so that the true x_0 is feasible, at least with high probability, when the noise term z is stochastic; that is, δ obeys $\|A^*z\|_\infty \leq \delta$ (with high probability). Another effective method, which we refer to as the LASSO [Tib96] (also known as basis pursuit denoising, or BPDN), assumes a different fidelity term and is the solution to

$$\begin{array}{ll} \text{minimize} & \|x\|_1 \\ \text{subject to} & \|y - Ax\|_2 \leq \epsilon, \end{array} \tag{4.1.3}$$

[1] The meaning of the word "optimal" shall be made precise later.

[2] There is a slight modification when the columns do not have the same norm, namely, $\|D^{-1}A^*(y - Ax)\|_\infty \leq \delta$, where D is diagonal and whose diagonal entries are the ℓ_2 norms of the columns of A.

where ϵ is a scalar, which again may be selected so that the true vector is feasible. Both of these estimators are generally able to accurately estimate nearly sparse vectors and it is, therefore, of interest to develop effective algorithms for each that can deal with problems involving thousands or even millions of variables and observations.

There are of course many techniques, which are perhaps more complicated than (4.1.2) and (4.1.3), for recovering signals or images from possibly undersampled noisy data. Suppose for instance that we have noisy data y (4.1.1) about an $n \times n$ image x_0; that is, $[x_0]_{ij}$ is an n^2-array of real numbers. Then to recover the image, one might want to solve a problem of this kind:

$$\begin{array}{ll} \text{minimize} & \|Wx\|_1 + \lambda \|x\|_{\text{TV}} \\ \text{subject to} & \|y - Ax\|_2 \leq \epsilon, \end{array} \tag{4.1.4}$$

where W is some (possibly non-orthogonal) transform such as an undecimated wavelet transform enforcing sparsity of the image in this domain, and $\|\cdot\|_{\text{TV}}$ is the isotropic total-variation norm introduced in [ROF92] defined as

$$\|x\|_{TV} := \sum_{i,j} \sqrt{|x[i+1,j] - x[i,j]|^2 + |x[i,j+1] - x[i,j]|^2}.$$

The motivation for (4.1.4) is to look for a sparse object in a transformed domain while reducing artifacts due to sparsity constraints alone, such as Gibbs oscillations, by means of the total-variation norm [ROF92, CG02, CR05]. The proposal (4.1.4) appears computationally more involved than both (4.1.2) and (4.1.3), and our goal is to develop effective algorithms for problems of this kind as well.

To continue our tour, another problem that has recently attracted a lot attention concerns the recovery of a low-rank matrix X_0 from undersampled data

$$y = \mathcal{A}(X_0) + z, \tag{4.1.5}$$

where $\mathcal{A} : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^m$ is a linear operator supplying information about X_0. An important example concerns the situation where only some of the entries of X_0 are revealed, $\mathcal{A}(X_0) = [X_0]_{ij} : (i,j) \in E \subset [n_1] \times [n_2]$, and the goal is to predict the values of all the missing entries. It has been shown [CR09, CT10, Gro11] that an effective way of recovering the missing information from y and the model (4.1.5) is via the convex program

$$\begin{array}{ll} \text{minimize} & \|X\|_* \\ \text{subject to} & X \in \mathcal{C}. \end{array} \tag{4.1.6}$$

Here, $\|X\|_*$ is the sum of the singular values of the matrix X, a quantity known as the *nuclear norm* of X. ($\|X\|_*$ is also the dual of the standard operator norm $\|X\|$, given by the largest singular

value of X). Above, $\mathcal{C}$ is a data fitting set, and might be $\{X : \mathcal{A}(X) = y\}$ in the noiseless case, or $\{X : \|\mathcal{A}^*(y - \mathcal{A}(X))\|_\infty \leq \delta\}$ (Dantzig selector-type constraint), or $\{X : \|y - \mathcal{A}(X)\|_2 \leq \epsilon\}$ (LASSO-type constraint) in the noisy setup. We are again interested in computational solutions to problems of this type.

4.1.2 The literature

There is of course an immense literature for solving problems of the types described above. Consider the LASSO, for example. Most of the works [HYZ08,WYGZ10,OMDY10,YOGD08,FNW07,WNF09, BT09, FHT10] are concerned with the unconstrained problem

$$\text{minimize} \qquad \tfrac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1, \tag{4.1.7}$$

which differs from (4.1.3) in that the hard constraint $\|Ax - b\|_2 \leq \epsilon$ is replaced with a quadratic penalty $\frac{1}{2}\lambda^{-1}\|Ax - b\|_2^2$. There are far fewer methods specially adapted to (4.1.3); let us briefly discuss some of them. SPGL1 [vdBF08] is a solver specifically designed for (4.1.3), and evidence from [BBC11] suggests that it is both robust and efficient. The issue is that at the moment, it cannot handle important variations such as

$$\begin{aligned} &\text{minimize} && \|Wx\|_1 \\ &\text{subject to} && \|y - Ax\|_2 \leq \epsilon, \end{aligned}$$

where W is an over-complete (i.e., more columns than rows) transform as in (4.1.4). The main reason is that SPGL1—as with almost all first-order methods for that matter—relies on the fact that the proximity operator associated with the ℓ_1 norm,

$$x(z;t) \triangleq \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \quad \tfrac{1}{2}t^{-1}\|x - z\|_2^2 + \|x\|_1, \tag{4.1.8}$$

is efficiently computable via soft-thresholding. This is not the case, however, when $\|x\|_1$ is replaced by a general term of the form $\|Wx\|_1$, except in the special cases when $WW^* = I$ [CP07b], and these special cases cannot occur if W is over-complete. NESTA [BBC11] can efficiently deal with an objective functional of the form $\|Wx\|_1$—that is, it works for any W and the extra computational cost is just one application of W and W^* per iteration—but it requires repeated projections onto the feasible set; see also [ABDF11] for a related approach, and [WBFA09] for a similar approach specialized for minimizing total variation. Hence, NESTA is efficient when AA^* is a projector or, more generally, when the eigenvalues of AA^* are well clustered. Other types of algorithms such as LARS [EHJT04] are based on homotopy methods, and compute the whole solution path; i.e., they find the solution to (4.1.7) for all values of the regularization parameter λ and, in doing so, find the

solution to the constrained problem (4.1.3). These methods do not scale well with problem size, however, especially when the solution is not that sparse.

The approach taken in this chapter, described in §4.1.3, is based off duality and smoothing, and these concepts have been widely studied in the literature. At the heart of the method is the fact that solving the dual problem eliminates difficulties with affine operators A, and this observation goes back to Uzawa's method [Cia89]. More recently, [MZ09, CP10c, CDV10] discuss dual method approaches to signal processing problems. In [MZ09], a non-negative version of (4.1.3) (which can be extended to the regular version by splitting x into positive and negative parts) is solved using inner and outer iterations. The outer iteration allows the inner problem to be smoothed, which is similar to the continuation idea presented in §4.5.5. The outer iteration is proved to converge rapidly, but depends on exactly solving the inner iteration. The work in [LST11] applies a similar outer iteration, which they recognize as the proximal point algorithm, applied to nuclear norm minimization. The method of [CP10c] applies a primal-dual method to unconstrained problems such as (4.1.7), or simplified and unconstrained versions of (4.1.4). Notably, they prove that when the objective function contains an appropriate strongly convex term, then the primal variable x_k converges with the bound $\|x_k - x^\star\|_2^2 \leq \mathcal{O}(1/k^2)$. The approach of [CDV10] considers smoothing (4.1.3) in a similar manner to that discussed in §4.2.4, but does not use continuation to diminish the effect of the smoothing. The dual problem is solved via the forward-backward method, and this allows the authors to prove that the primal variable converges, though without a known bound on the rate.

Turning to the Dantzig selector, solution algorithms are scarce. The standard way of solving (4.1.2) is via linear programming techniques [CR07b] since it is well known that it can be recast as a linear program [CT07a]. Typical modern solvers rely on interior-point methods (IPM) which are somewhat problematic for large-scale problems, since they do not scale well with size. Another way of solving (4.1.2) is via the new works [JRL09, Rom08], which use homotopy methods inspired by LARS to compute the whole solution path of the Dantzig selector. These methods, however, are also unable to cope with large problems. As an example of the speed of these methods, the authors of l1ls [KKB07], which is a log-barrier interior-point method using preconditioned conjugate-gradients to solve the Newton step, report that on an instance of the (4.1.7) problem, their method is 20× faster than the commercial IPM MOSEK [Mos02], 10× faster than the IPM PDCO [SK02], 78× faster than IPM l1Magic [CR07b], and 1.6× faster than Homotopy [DT08]. Yet the first-order method FPC [HYZ08] reports that FPC is typically 10 to 20× faster than l1ls, and sometimes even 1040× faster. These results are for (4.1.7) and not the Dantzig selector, but it is reasonable to conclude that in general IPM and homotopy do not seem reasonable methods for extremely large problems.

The accelerated first-order algorithm recently introduced in [Lu09] can handle large Dantzig

selector problems, but with the same limitations as NESTA since it requires inverting AA^*. A method based on operator splitting has been suggested in [FS09] but results have not yet been reported. Another alternative is adapting SPGL1 to this setting, but this comes with the caveat that it does not handle slight variations as discussed above.

Finally, as far as the mixed norm problem (4.1.4) is concerned, we are not aware of efficient solution algorithms. One can always recast this problem as a second-order cone program (SOCP) which one could then solve via an interior-point method; but again, this is problematic for large-scale problems.

4.1.3 Our approach

In this chapter, we develop a template for solving a variety of problems such as those encountered thus far. The template proceeds as follows: first, determine an equivalent conic formulation; second, determine its dual; third, apply smoothing; and fourth, solve using an optimal first-order method.

4.1.3.1 Conic formulation

In reality, our approach can be applied to general models expressed in the following canonical form:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & \mathcal{A}(x) + b \in \mathcal{K}. \end{array} \tag{4.1.9}$$

The optimization variable is a vector $x \in \mathbb{R}^n$, and the objective function f is convex, possibly extended-valued, and not necessarily smooth. The constraint is expressed in terms of a linear operator $\mathcal{A} : \mathbb{R}^n \to \mathbb{R}^m$, a vector $b \in \mathbb{R}^m$, and a closed, convex cone $\mathcal{K} \subseteq \mathbb{R}^m$. We shall call a model of the form (4.1.9) that is equivalent to a given convex optimization model $\mathcal{P}$ a conic form for $\mathcal{P}$.

The conic constraint $\mathcal{A}(x) + b \in \mathcal{K}$ may seem specialized, but in fact any closed convex subset of $\mathbb{R}^n$ may be represented in this fashion; and models involving complex variables, matrices, or other vector spaces can be handled by defining appropriate isomorphisms. Of course, some constraints are more readily transformed into conic form than others; included in this former group are linear equations, linear inequalities, and convex inequalities involving norms of affine forms. Thus virtually every convex compressed sensing model may be readily converted. Almost all models admit multiple conic forms, and each results in a different final algorithm.

For example, the Dantzig selector (4.1.2) can be mapped to conic form as follows:

$$f(x) \to \|x\|_1, \qquad \mathcal{A}(x) \to (A^*Ax, 0), \qquad b \to (-A^*y, \delta), \qquad \mathcal{K} \to \mathcal{L}_\infty^n, \tag{4.1.10}$$

where $\mathcal{L}_\infty^n$ is the epigraph of the ℓ_∞ norm: $\mathcal{L}_\infty^n = \{(y, t) \in \mathbb{R}^{n+1} : \|y\|_\infty \le t\}$.

4.1.3.2 Dualization

The conic form (4.1.9) does not immediately lend itself to efficient solution using first-order methods for two reasons: first, because f may not be smooth; and second, because projection onto the set $\{x \,|\, \mathcal{A}(x)+b \in \mathcal{K}\}$, or even the determination of a single feasible point, can be expensive. We propose to resolve these issues by solving either the dual problem, or a carefully chosen approximation of it. Recall that the dual of our canonical form (4.1.9) is given by

$$\begin{array}{ll} \text{maximize} & g(\lambda) \\ \text{subject to} & \lambda \in \mathcal{K}^*, \end{array} \tag{4.1.11}$$

where $g(\lambda)$ is the Lagrange dual function

$$g(\lambda) \triangleq \inf_x \mathcal{L}(x, \lambda) = \inf_x f(x) - \langle \lambda, \mathcal{A}(x) + b \rangle,$$

and $\mathcal{K}^*$ is the dual cone defined via

$$\mathcal{K}^* = \{\lambda \in \mathbb{R}^m : \langle \lambda, x \rangle \geq 0 \text{ for all } x \in \mathcal{K}\}.$$

The dual form has an immediate benefit that for the problems of interest, projections onto the dual cone are usually tractable and computationally very efficient. For example, consider the projection of a point onto the feasible set $\{x : \|Ax - y\|_2 \leq \epsilon\}$ of the LASSO, an operation which may be expensive. However, one can recast the constraint as $\mathcal{A}(x) + b \in \mathcal{K}$ with

$$\mathcal{A}(x) \to (Ax, 0), \qquad b \to (-y, \epsilon) \qquad \mathcal{K} \to \mathcal{L}_2^m, \tag{4.1.12}$$

where $\mathcal{L}_2^m$ is the second-order cone $\mathcal{L}_2^m = \{(y, t) \in \mathbb{R}^{m+1} : \|y\|_2 \leq t\}$. This cone is self dual, i.e., $(\mathcal{L}_2^m)^* = \mathcal{L}_2^m$, and projection onto $\mathcal{L}_2^m$ is trivial: indeed, it is given by

$$(y, t) \mapsto \begin{cases} (y, t), & \|y\|_2 \leq t, \\ c(y, \|y\|_2), & -\|y\|_2 \leq t \leq \|y\|_2, \quad c = \dfrac{\|y\|_2 + t}{2\|y\|_2}. \\ (0, 0), & t \leq -\|y\|_2, \end{cases} \tag{4.1.13}$$

And so we see that by eliminating the affine mapping, the projection computation has been greatly simplified. Of course, not every cone projection admits as simple a solution as (4.1.13); but as we will show, all of the cones of interest to us do indeed.

4.1.3.3 Smoothing

Unfortunately, because of the nature of the problems under study, the dual function is usually not differentiable either, and direct solution via subgradient methods would converge too slowly. Our solution is inspired by the smoothing technique due to Nesterov [Nes05]. We shall see that if one modifies the primal objective $f(x)$ and instead solves

$$\begin{array}{ll} \text{minimize} & f_\mu(x) \triangleq f(x) + \mu d(x) \\ \text{subject to} & \mathcal{A}(x) + b \in \mathcal{K}, \end{array} \tag{4.1.14}$$

where $d(x)$ is a strongly convex function to be defined later and μ a positive scalar, then the dual problem takes the form

$$\begin{array}{ll} \text{maximize} & g_\mu(\lambda) \\ \text{subject to} & \lambda \in \mathcal{K}^*, \end{array} \tag{4.1.15}$$

where g_μ is a smooth approximation of g. This approximate model can now be solved using first-order methods. As a general rule, higher values of μ improve the performance of the underlying solver, but at the expense of accuracy. Techniques such as continuation can be used to recover the accuracy lost, however, so the precise trade-off is not so simple.

In many cases, the smoothed dual can be reduced to an unconstrained problem of the form

$$\text{maximize} \qquad -g_{\text{smooth}}(z) - h(z), \tag{4.1.16}$$

with optimization variable $z \in \mathbb{R}^m$, where g_{smooth} is convex and smooth and h convex, non-smooth, and possibly extended-valued. For instance, for the Dantzig selector (4.1.2), $h(z) = \delta\|z\|_1$. As we shall see, this so-called composite form can also be solved efficiently using optimal first-order methods. In fact, the reduction to composite form often simplifies some of the central computations in the algorithms.

4.1.3.4 First-order methods

Optimal first-order methods are proper descendants of the classic projected gradient algorithm. For the smoothed dual problem (4.1.15), a prototypical projected gradient algorithm begins with a point $\lambda_0 \in \mathcal{K}^*$, and generates updates for $k = 0, 1, 2, ...$ as follows:

$$\lambda_{k+1} \leftarrow \operatorname*{argmin}_{\lambda \in \mathcal{K}^*} \|\lambda_k + t_k \nabla g_\mu(\lambda_k) - \lambda\|_2, \tag{4.1.17}$$

given step sizes $\{t_k\}$. The method has also been extended to composite problems like (4.1.16) [Wri07, Nes07, Tse08]; the corresponding iteration is

$$z_{k+1} \leftarrow \operatorname*{argmin}_y g_{\text{smooth}}(z_k) + \langle \nabla g_{\text{smooth}}(z_k), z - z_k \rangle + \tfrac{1}{2t_k}\|z - z_k\|^2 + h(z). \tag{4.1.18}$$

Note the use of a general norm $\|\cdot\|$ and the inclusion of the non-smooth term h. We call the minimization in (4.1.18) a generalized projection, because it reduces to a standard projection (4.1.17) if the norm is Euclidean and h is an indicator function. This generalized form allows us to construct efficient algorithms for a wider variety of models.

For the problems under study, the step sizes $\{t_k\}$ above can be chosen so that ϵ-optimality (that is, $\sup_{\lambda \in \mathcal{K}^*} g_\mu(\lambda) - g_\mu(\lambda_k) \leq \epsilon$) can be achieved in $\mathcal{O}(1/\epsilon)$ iterations [Nes04]. In 1983, Nesterov reduced this cost to $\mathcal{O}(1/\sqrt{\epsilon})$ using a slightly more complex iteration

$$\lambda_{k+1} \leftarrow \operatorname*{argmin}_{\lambda \in \mathcal{K}^*} \|\nu_k + t_k \nabla g_\mu(\nu_k) - \lambda\|_2, \quad \nu_{k+1} \leftarrow \lambda_{k+1} + \alpha_k(\lambda_{k+1} - \lambda_k), \tag{4.1.19}$$

where $\nu_0 = \lambda_0$ and the sequence $\{\alpha_k\}$ is constructed according to a particular recurrence relation. Previous work by Nemirosvski and Yudin had established $\mathcal{O}(1/\sqrt{\epsilon})$ complexity as the best that can be achieved for this class of problems [NY83], so Nesterov's modification is indeed optimal. Many alternative first-order methods have since been developed [Nes88, Nes05, Nes07, Tse08, AT06, LLM09], including methods that support generalized projections. We examine these methods in more detail in §4.5.

We have not yet spoken about the complexity of computing g_μ or g_{smooth} and their gradients. For now, let us highlight the fact that $\nabla g_\mu(\lambda) = -\mathcal{A}(x(\lambda)) - b$, where

$$x(\lambda) \triangleq \operatorname*{argmin}_x \mathcal{L}_\mu(x, \lambda) = \operatorname*{argmin}_x f(x) + \mu d(x) - \langle \mathcal{A}(x) + b, \lambda \rangle, \tag{4.1.20}$$

and $d(x)$ is a selected proximity function. In the common case that $d(x) = \frac{1}{2}\|x - x_0\|^2$, the structure of (4.1.20) is identical to that of a generalized projection. Thus we see that the ability to efficiently minimize the sum of a linear term, a proximity function, and a non-smooth function of interest is the fundamental computational primitive involved in our method. Equation (4.1.20) also reveals how to recover an approximate primal solution as λ approaches its optimal value.

4.1.4 Contributions

The formulation of compressed sensing models in conic form is not widely known. Yet the convex optimization modeling framework `CVX` [GB10] converts *all* models into conic form; and the compressed sensing package ℓ_1-Magic [CR07b] converts problems into second-order cone programs (SOCPs). Both systems utilize interior-point methods instead of first-order methods, however. As mentioned

above, the smoothing step is inspired by [Nes05], and is similar in structure to traditional Lagrangian augmentation. As we also noted, first-order methods have been a subject of considerable research.

Taken separately, then, none of the components in this approach is new. However their combination and application to solve compressed sensing problems leads to effective algorithms that have not previously been considered. For instance, applying our methodology to the Dantzig selector gives a novel and efficient algorithm (in fact, it gives several novel algorithms, depending on which conic form is used). Numerical experiments presented later in the chapter show that one can solve the Dantzig selector problem with a reasonable number of applications of A and its adjoint; the exact number depends upon the desired level of accuracy. In the case of the LASSO, our approach leads to novel algorithms which are competitive with state-of-the-art methods such as SPGL1.

Aside from good empirical performance, we believe that the primary merit of our framework lies in its flexibility. To be sure, all the compressed sensing problems listed at the beginning of this chapter, and of course many others, can be solved via this approach. These include total-variation norm problems, ℓ_1-analysis problems involving objectives of the form $\|Wx\|_1$ where W is neither orthogonal nor diagonal, and so on. In each case, our framework allows us to construct an effective algorithm, thus providing a computational solution to almost every problem arising in sparse signal or low-rank matrix recovery applications.

Furthermore, in the course of our investigation, we have developed a number of additional technical contributions. For example, we will show that certain models, including the Dantzig selector, exhibit an exact penalty property: the exact solution to the original problem is recovered even when some smoothing is applied. We have also developed a novel continuation scheme that allows us to employ more aggressive smoothing to improve solver performance while still recovering the exact solution to the unsmoothed problem. The flexibility of our template also provides opportunities to employ novel approaches for controlling the step size.

4.1.5 Software

This chapter is accompanied with a software release [BCG10b], including a detailed user guide which gives many additional implementation details not discussed in this chapter. Since most compressed sensing problems can be easily cast into standard conic form, our software provides a powerful and flexible computational tool for solving a large range of problems researchers might be interested in experimenting with.

The software is not a single, monolithic solver; rather, it is a suite of programs and routines designed to serve as building blocks for constructing complete algorithms. Roughly speaking, we can divide the routines into three levels. On the first level is a suite of routines that implement a variety of known first-order solvers, including the standard projected gradient algorithm and known optimal variants by Nesterov and others. On the second level are wrappers designed to accept

problems in conic standard form (4.1.9) and apply the first-order solvers to the smoothed dual problem. Finally, the package includes a variety of routines to directly solve the specific models described in this chapter and to reproduce our experiments.

We have worked to ensure that each of the solvers is as easy to use as possible, by providing sensible defaults for line search, continuation, and other factors. At the same time, we have sought to give the user flexibility to insert their own choices for these components. We also want to provide the user with the opportunity to compare the performance of various first-order variants on their particular application. We do have some general views about which algorithms perform best for compressed sensing applications, however, and will share some of them in §4.7.

4.1.6 Organization of the chapter

In §4.2, we continue the discussion of conic formulations, including a derivation of the dual conic formulation and details about the smoothing operation. Section 4.3 instantiates this general framework to derive a new algorithm for the Dantzig selector problem. In §4.4, we provide further selected instantiations of our framework including the LASSO, total-variation problems, ℓ_1-analysis problems, and common nuclear-norm minimization problems. In §4.5, we review a variety of first-order methods and suggest improvements. Section 4.6 derives the dual formulation using the machinery of dual functions instead of dual cones, and covers known convergence results. Section 4.7 presents numerical results illustrating both the empirical effectiveness and the flexibility of our approach. Some interesting special cases are discussed in 4.8, such as solving linear program in standard form. Section 4.9 provides a short introduction to the software release accompanying this chapter. Finally, the appendix gives a proof of the exact penalty property for general linear programs (which itself is not a novel result), and thus for Dantzig selector and basis pursuit models, and describes a unique approach we use to generate test models so that their exact solution is known in advance.

4.2 Conic formulations

4.2.1 Alternate forms

In the introduction, we presented our standard conic form (4.1.9) and a specific instance for the Dantzig selector in (4.1.10). As we said then, conic forms are rarely unique; this is true even if one disregards simple scalings of the cone constraint. For instance, we may express the Dantzig selector constraint as an intersection of linear inequalities, $-\delta\mathbf{1} \preceq A^*(y - Ax) \preceq \delta\mathbf{1}$, suggesting the following alternative:

$$f(x) \to \|x\|_1, \qquad \mathcal{A}(x) \to \begin{bmatrix} -A^*A \\ A^*A \end{bmatrix} x, \qquad b \to \begin{bmatrix} \delta\mathbf{1} + A^*y \\ \delta\mathbf{1} - A^*y \end{bmatrix}, \qquad \mathcal{K} \to \mathbb{R}^{2n}_+. \tag{4.2.1}$$

We will return to this alternative later in §4.3.5. In many instances, a conic form may involve the manipulation of the objective function as well. For instance, if we first transform (4.1.2) to

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \|x\|_1 \leq t \\ & \|A^*(y - Ax)\|_\infty \leq \delta, \end{array}$$

then yet another conic form results:

$$f(x,t) \to t, \qquad \mathcal{A}(x,t) \to (x, t, A^*Ax, 0), \qquad b \to (0, 0, -A^*y, \delta), \qquad \mathcal{K} \to \mathcal{L}_1^n \times \mathcal{L}_\infty^n, \tag{4.2.2}$$

where $\mathcal{L}_1^n$ is the epigraph of the ℓ_1 norm, $\mathcal{L}_1^n = \{(y,t) \in \mathbb{R}^{n+1} : \|y\|_1 \leq t\}$.

Our experiments show that different conic formulations yield different levels of performance using the same numerical algorithms. Some are simpler to implement than others as well. Therefore, it is worthwhile to at least explore these alternatives to find the best choice for a given application.

4.2.2 The dual

To begin with, the conic Lagrangian associated with (4.1.9) is given by

$$\mathcal{L}(x, \lambda) = f(x) - \langle \lambda, \mathcal{A}(x) + b \rangle, \tag{4.2.3}$$

where $\lambda \in \mathbb{R}^m$ is the Lagrange multiplier, constrained to lie in the dual cone $\mathcal{K}^*$. The dual function $g : \mathbb{R}^m \to (\mathbb{R} \cup \{-\infty\})$ is, therefore,

$$g(\lambda) = \inf_x \mathcal{L}(x, \lambda) = -f^*(\mathcal{A}^*(\lambda)) - \langle b, \lambda \rangle. \tag{4.2.4}$$

Here, $\mathcal{A}^* : \mathbb{R}^m \to \mathbb{R}^n$ is the adjoint of the linear operator $\mathcal{A}$ and $f^* : \mathbb{R}^n \to (\mathbb{R} \cup \{+\infty\})$ is the convex conjugate of f defined by

$$f^*(z) = \sup_x \langle z, x \rangle - f(x).$$

Thus the dual problem is given by

$$\begin{array}{ll} \text{maximize} & -f^*(\mathcal{A}^*(\lambda)) - \langle b, \lambda \rangle \\ \text{subject to} & \lambda \in \mathcal{K}^*. \end{array} \tag{4.2.5}$$

Given a feasible primal/dual pair (x, λ), the *duality gap* is the difference between their respective objective values. The non-negativity of the duality gap is easily verified:

$$f(x) - g(\lambda) = f(x) + f^*(\mathcal{A}^*(\lambda)) + \langle b, \lambda \rangle \geq \langle x, \mathcal{A}^*(\lambda) \rangle + \langle b, \lambda \rangle = \langle \mathcal{A}(x) + b, \lambda \rangle \geq 0. \tag{4.2.6}$$

The first inequality follows from the definition of conjugate functions, while the second follows from the definition of the dual cone. If both the primal and dual are strictly feasible—as is the case for all problems we are interested in here—then the minimum duality gap is exactly zero, and there exists an optimal pair $(x^\star, \lambda^\star)$ that achieves $f(x^\star) = g(\lambda^\star) = \mathcal{L}(x^\star, \lambda^\star)$. It is important to note that the optimal points are not necessarily unique; more about this in §4.2.4. But any optimal primal/dual pair will satisfy the KKT optimality conditions [BV04]

$$\mathcal{A}(x^\star) + b \in \mathcal{K}, \quad \lambda^\star \in \mathcal{K}^*, \quad \langle \mathcal{A}(x^\star) + b, \lambda^\star \rangle = 0, \quad \mathcal{A}^*(\lambda^\star) \in \partial f(x^\star), \tag{4.2.7}$$

where ∂f refers to the subgradient of f.

4.2.3 The differentiable case

The dual function is of course concave; and its derivative (when it exists) is given by

$$\nabla g(\lambda) = -\mathcal{A}(x(\lambda)) - b, \quad x(\lambda) \in \operatorname*{argmin}_x \mathcal{L}(x, \lambda). \tag{4.2.8}$$

It is possible that the minimizer $x(\lambda)$ is not unique, so in order to be differentiable, all such minimizers must yield the same value of $-\mathcal{A}(x(\lambda)) - b$ [BNO03].

If g is finite and differentiable on the entirety of $\mathcal{K}^*$, then it becomes trivial to locate an initial dual point (e.g., $\lambda = 0$); and for many genuinely useful cones $\mathcal{K}^*$, it becomes trivial to project an arbitrary point $\lambda \in \mathbb{R}^m$ onto this feasible set. If the argmin calculation in (4.2.8) is computationally practical, we may entertain the construction of a projected gradient method for solving the dual problem (4.2.5) directly; i.e., without our proposed smoothing step. Once an optimal dual point $\lambda^\star$ is recovered, an optimal solution to the original problem (4.1.9) is recovered by solving $x^\star \in \operatorname{argmin}_x \mathcal{L}(x, \lambda^\star) \cap \{x : \mathcal{A}x + b \in \mathcal{K}\}$. If $\operatorname{argmin}_x \mathcal{L}(x, \lambda^\star)$ is unique, which is the case when f is strictly convex, then this minimizer is necessarily feasible and is the primal optimal solution.

Further suppose that f is strongly convex[3] ; that is, it satisfies for some constant $m_f > 0$,

$$f((1-\alpha)x + \alpha x') \leq (1-\alpha)f(x) + \alpha f(x') - m_f \alpha(1-\alpha)\|x - x'\|_2^2/2 \tag{4.2.9}$$

for all $x, x' \in \operatorname{dom}(f) = \{x : f(x) < +\infty\}$ and $0 \leq \alpha \leq 1$. Then assuming the problem is feasible, it admits a unique optimal solution. The Lagrangian minimizers $x(\lambda)$ are unique for all $\lambda \in \mathbb{R}^n$; so g is differentiable everywhere. Furthermore, [Nes05] proves that the gradient of g is Lipschitz continuous, obeying

$$\|\nabla g(\lambda') - \nabla g(\lambda)\|_2 \leq m_f^{-1} \|\mathcal{A}\|^2 \|\lambda' - \lambda\|_2, \tag{4.2.10}$$

[3] We use the Euclidean norm, but any norm works as long as $\|\mathcal{A}\|$ in (4.2.10) is appropriately defined; see [Nes05].

where $\|\mathcal{A}\| = \sup_{\|x\|_2=1} \|\mathcal{A}(x)\|_2$ is the induced operator norm of $\mathcal{A}$. So when f is strongly convex, then provably convergent, accelerated gradient methods in the Nesterov style are possible.

4.2.4 Smoothing

Unfortunately, it is more likely that g is not differentiable (or even finite) on all of $\mathcal{K}^*$. So we consider a smoothing approach similar to that proposed in [Nes05] to solve an approximation of our problem. Consider the following perturbation of (4.1.9):

$$\begin{array}{ll} \text{minimize} & f_\mu(x) \triangleq f(x) + \mu d(x) \\ \text{subject to} & \mathcal{A}(x) + b \in \mathcal{K} \end{array} \tag{4.2.11}$$

for some fixed smoothing parameter $\mu > 0$ and a strongly convex function d obeying

$$d(x) \geq d(x_0) + \tfrac{1}{2}\|x - x_0\|^2 \tag{4.2.12}$$

for some fixed point $x_0 \in \mathbb{R}^n$. Such a function is usually called a proximity function.

The new objective f_μ is strongly convex with $m_f = \mu$, so the full benefits described in §4.2.3 now apply. The Lagrangian and dual functions become[4]

$$\mathcal{L}_\mu(x, \lambda) = f(x) + \mu d(x) - \langle \lambda, \mathcal{A}(x) + b \rangle \tag{4.2.13}$$

$$g_\mu(\lambda) \triangleq \inf_x \mathcal{L}_\mu(x, \lambda) = -(f + \mu d)^*(\mathcal{A}^*(\lambda)) - \langle b, \lambda \rangle. \tag{4.2.14}$$

One can verify that for the affine objective case $f(x) \triangleq \langle c_0, x \rangle + d_0$, the dual and smoothed dual function take the form

$$\begin{aligned} g(\lambda) &= d_0 - I_{\ell_\infty}(\mathcal{A}^*(\lambda) - c_0) - \langle b, \lambda \rangle, \\ g_\mu(\lambda) &= d_0 - \mu d^*(\mu^{-1}(\mathcal{A}^*(\lambda) - c_0)) - \langle b, \lambda \rangle, \end{aligned}$$

where I_{ℓ_∞} is the indicator function of the ℓ_∞ norm ball; that is,

$$I_{\ell_\infty}(y) \triangleq \begin{cases} 0, & \|y\|_\infty \leq 1, \\ +\infty, & \|y\|_\infty > 1. \end{cases}$$

The new optimality conditions are

$$\mathcal{A}(x_\mu) + b \in \mathcal{K}, \quad \lambda_\mu \in \mathcal{K}^*, \quad \langle \mathcal{A}(x_\mu) + b, \lambda_\mu \rangle = 0, \quad \mathcal{A}^*(\lambda_\mu) - \mu \nabla d(x_\mu) \in \partial f(x_\mu). \tag{4.2.15}$$

[4] One can also observe the identity $g_\mu(\lambda) = \sup_z -f^*(\mathcal{A}^*(\lambda) - z) - \mu d^*(\mu^{-1} z) - \langle b, \lambda \rangle$.

Because the Lipschitz bound (4.2.10) holds, first-order methods may be employed to solve (4.2.11) with provable performance. The iteration counts for these methods are proportional to the square root of the Lipschitz constant, and therefore proportional to $\mu^{-1/2}$. There is a trade off between the accuracy of the approximation and the performance of the algorithms that must be explored.[5]

For each $\mu > 0$, the smoothed model obtains a single minimizer x_μ; and the trajectory traced by x_μ as μ varies converges to an optimal solution $x^\star \triangleq \lim_{\mu \to 0+} x_\mu$. Henceforth, when speaking about the (possibly non-unique) optimal solution $x^\star$ to the original model, we will be referring to this uniquely determined value. Later we will show that for some models, including the Dantzig selector, the approximate model is *exact*: that is, $x_\mu = x^\star$ for sufficiently small but nonzero μ.

Roughly speaking, the smooth dual function g_μ is what we would obtain if the Nesterov smoothing method described in [Nes05] were applied to the dual function g. It is worthwhile to explore how things would differ if the Nesterov approach were applied directly to the primal objective $f(x)$. Suppose that $f(x) = \|x\|_1$ and $d(x) = \frac{1}{2}\|x\|_2^2$. The Nesterov approach yields a smooth approximation f_μ^{N} whose elements can be described by the formula

$$\left[f_\mu^{\mathrm{N}}(x)\right]_i = \sup_{|z| \le 1} z x_i - \tfrac{1}{2}\mu z^2 = \begin{cases} \frac{1}{2}\mu^{-1} x_i^2, & |x_i| \le \mu, \\ |x_i| - \frac{1}{2}\mu, & |x| \ge \mu, \end{cases} \quad i = 1, 2, \ldots, n. \tag{4.2.16}$$

Readers may recognize this as the Huber penalty function with half-width μ; a graphical comparison with f_μ is provided in Figure 4.1. Its smoothness may seem to be an advantage over our choice $f_\mu(x) = \|x\|_1 + \frac{1}{2}\|x\|_2^2$, but the difficulty of projecting onto the set $\{x \,|\, \mathcal{A}(x) + b \in \mathcal{K}\}$ remains; so we still prefer to solve the dual problem. Furthermore, the quadratic behavior of f_μ^{N} around $x_i = 0$ eliminates the tendency towards solutions with many zero values. In contrast, $f_\mu(x)$ maintains the sharp vertices from the ℓ_1 norm that are known to encourage sparse solutions.

4.2.5 Composite forms

In most of the cases under study, the dual variable can be partitioned as $\lambda \triangleq (z, \tau) \in \mathbb{R}^{m-\bar{m}} \times \mathbb{R}^{\bar{m}}$ such that the smoothed dual $g_\mu(z, \tau)$ is linear in τ; that is,

$$g_\mu(\lambda) = -g_{\mathrm{smooth}}(z) - \langle v_\mu, \tau \rangle \tag{4.2.17}$$

for some smooth convex function g_{smooth} and a constant vector $v_\mu \in \mathbb{R}^{\bar{m}}$. An examination of the Lagrangian $\mathcal{L}_\mu$ (4.2.13) reveals a precise condition under which this occurs: when the linear operator $\mathcal{A}$ is of the form $\mathcal{A}(x) \to (\bar{\mathcal{A}}(x), \mathbf{0}_{\bar{m} \times 1})$, as seen in the conic constraints for the Dantzig selector (4.1.10) and LASSO (4.1.12). If we partition $b = (\bar{b}, b_\tau)$ accordingly, then evidently $v_\mu = b_\tau$.

[5] In fact, even when the original objective is strongly convex, further adding a strongly convex term may be worthwhile to improve performance.

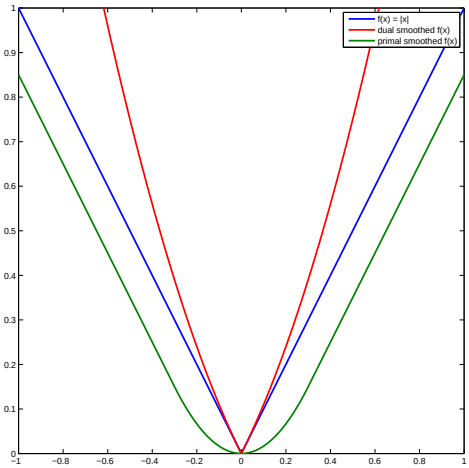


Figure 4.1: The original objective $f(x) = |x|$ (blue), our modification (red), and Nesterov's smoothing (green)

Under such conditions, it is more natural to work with $\bar{\mathcal{A}}$, $\bar{b}$, and b_τ directly, and exploit some useful simplifications. Specifically, let us define the function

$$h : \mathbb{R}^{m-\bar{m}} \to (\mathbb{R} \cup \{+\infty\}), \quad h(z) \triangleq \inf\{\langle b_\tau, \tau \rangle \,|\, (z, \tau) \in \mathcal{K}^*\}. \tag{4.2.18}$$

Then the dual problem reduces to a non-smooth unconstrained maximization

$$\text{maximize} \quad \bar{g}_\mu(z) \triangleq -g_{\text{smooth}}(z) - h(z).$$

The gradient of g_{smooth} is $\nabla g_{\text{smooth}}(z) = \bar{\mathcal{A}}(x(z)) + \bar{b}$, where $x(z)$ is the minimizer of a reduced Lagrangian

$$\bar{\mathcal{L}}_\mu(x, z) = f(x) + \mu d(x) - \langle z, \bar{\mathcal{A}}(x) + \bar{b} \rangle. \tag{4.2.19}$$

4.2.6 Projections

A standard gradient projection step for the smoothed dual problem is

$$\lambda_{k+1} \leftarrow \underset{\lambda \in \mathcal{K}^*}{\text{argmin}} \, \|\lambda - \lambda_k - t_k \nabla g_\mu(\lambda_k)\|_2. \tag{4.2.20}$$

For the composite version of the same problem, the corresponding generalized projection is

$$z_{k+1} \leftarrow \underset{z}{\text{argmin}} \, g_{\text{smooth}}(z_k) + \langle \nabla g_{\text{smooth}}(z_k), z - z_k \rangle + \tfrac{1}{2t_k} \|z - z_k\|^2 + h(z). \tag{4.2.21}$$

Integrating the definition of $\nabla g_{\text{smooth}}(z)$ into (4.2.21) and simplifying yields a two-sequence recursion:

$$\begin{aligned} x_k &\leftarrow \operatorname*{argmin}_x f(x) + \mu d(x) - \langle \bar{\mathcal{A}}^*(z_k), x \rangle \\ z_{k+1} &\leftarrow \operatorname*{argmin}_z h(z) + \tfrac{1}{2t_k}\|z - z_k\|^2 + \langle \bar{\mathcal{A}}(x_k) + \bar{b}, z \rangle. \end{aligned} \tag{4.2.22}$$

Note the similarity in computational structure of the two formulae. This similarity is even more evident in the common scenario where $d(x) = \frac{1}{2}\|x - x_0\|^2$ for some fixed $x_0 \in \mathbb{R}^n$.

Let Σ be the matrix composed of the first $m - \bar{m}$ rows of the $m \times m$ identity matrix, so that $\Sigma\lambda \equiv z$ for all $\lambda = (z, \tau)$. Then (4.2.21) can also be written in terms of $\mathcal{K}^*$ and g_μ:

$$z_{k+1} \leftarrow \Sigma \cdot \operatorname*{argmax}_{\lambda \in \mathcal{K}^*} g_\mu(\lambda_k) + \langle \nabla g_\mu(\lambda_k), \lambda - \lambda_k \rangle - \tfrac{1}{2t_k}\|\Sigma(\lambda - \lambda_k)\|^2, \tag{4.2.23}$$

where $\lambda_k \triangleq (z_k, h(z_k))$. If $m = 0$ and the norm is Euclidean, then $\Sigma = I$ and the standard projection (4.2.20) is recovered. So (4.2.21) is indeed a true generalization, as claimed in §4.1.3.4.

The key feature of the composite approach, then, is the removal of the linear variables τ from the proximity term. Given that they are linearly involved to begin with, this yields a more accurate approximation of the dual function, so we might expect a composite approach to yield improved performance. In fact, the theoretical predictions of the number of iterations required to achieve a certain level of accuracy are identical; and in our experience, any differences in practice seem minimal at best. The true advantage to the composite approach is that generalized projections more readily admit analytic solutions and are less expensive to compute.

4.3 A novel algorithm for the Dantzig selector

We now weave together the ideas of the last two sections to develop a novel algorithm for the Dantzig selector problem (4.1.2).

4.3.1 The conic form

We use the standard conic formulation (4.1.9) with the mapping (4.1.10) as discussed in the introduction, which results in the model

$$\begin{array}{ll} \text{minimize} & \|x\|_1 \\ \text{subject to} & (A^*(y - Ax), \delta) \in \mathcal{L}^n_\infty, \end{array} \tag{4.3.1}$$

where $\mathcal{L}_\infty^m$ is the epigraph of the ℓ_∞ norm. The dual variable λ, therefore, will lie in the dual cone $(\mathcal{L}_\infty^n)^* = \mathcal{L}_1^n$, the epigraph of the ℓ_1 norm. Defining $\lambda = (z, \tau)$, the conic dual (4.2.5) is

$$\begin{array}{ll} \text{maximize} & -I_{\ell_\infty}(-A^*Az) - \langle A^*y, z\rangle - \delta\tau \\ \text{subject to} & (z,\tau) \in \mathcal{L}_1^n, \end{array} \tag{4.3.2}$$

where $f^* = I_{\ell_\infty}$ is the indicator function of the ℓ_∞ norm ball as before. Clearly the optimal value of τ must satisfy $\|z\|_1 = \tau$,[6] so eliminating it yields

$$\text{maximize} \qquad -I_{\ell_\infty}(-A^*Az) - \langle A^*y, z\rangle - \delta\|z\|_1. \tag{4.3.3}$$

Neither (4.3.2) nor (4.3.3) has a smooth objective, so the smoothing approach discussed in §4.2.4 will indeed be necessary.

4.3.2 Smooth approximation

We augment the objective with a strongly convex term

$$\begin{array}{ll} \text{minimize} & \|x\|_1 + \mu d(x) \\ \text{subject to} & (A^*(y - Ax), \delta) \in \mathcal{K} \triangleq \mathcal{L}_\infty^n. \end{array} \tag{4.3.4}$$

The Lagrangian of this new model is

$$\mathcal{L}_\mu(x; z, \tau) = \|x\|_1 + \mu d(x) - \langle z, A^*(y - Ax)\rangle - \delta\tau.$$

Letting $x(z)$ be the unique minimizer of $\mathcal{L}_\mu(x; z, \tau)$, the dual function becomes

$$g_\mu(z, \tau) = \|x(z)\|_1 + \mu d(x(z)) - \langle z, A^*(y - Ax(z))\rangle - \tau\delta.$$

Eliminating τ per §4.2.5 yields a composite form $\bar{g}_\mu(z) = -g_{\text{smooth}}(z) - h(z)$ with

$$g_{\text{smooth}}(z) = -\|x(z)\|_1 - \mu d(x(z)) + \langle z, A^*(y - Ax(z))\rangle, \quad h(z) = \delta\|z\|_1.$$

The gradient of g_{smooth} is $\nabla g_{\text{smooth}}(z) = A^*(y - Ax(z))$.

The precise forms of $x(z)$ and ∇g_{smooth} depend of course on our choice of proximity function $d(x)$. For our problem, the simple convex quadratic

$$d(x) = \tfrac{1}{2}\|x - x_0\|_2^2,$$

[6] We assume $\delta > 0$ here; if $\delta = 0$, the form is slightly different.

for a fixed center point $x_0 \in \mathbb{R}^n$, works well, and guarantees that the gradient is Lipschitz continuous with a constant at most $\mu^{-1}\|A^*A\|^2$. With this choice, $x(z)$ can be expressed in terms of the *soft-thresholding* operation which is a common fixture in algorithms for sparse recovery. For scalars x and $s \geq 0$, define

$$\text{SoftThreshold}(x, s) = \text{sgn}(x) \cdot \max\{|x| - s, 0\} = \begin{cases} x + s, & x \leq -s, \\ 0, & |x| \leq s, \\ x - s, & x \geq s. \end{cases}$$

When the first input x is a vector, the soft-thresholding operation is to be applied componentwise. Armed with this definition, the formula for $x(z)$ becomes

$$x(z) = \text{SoftThreshold}(x_0 - \mu^{-1}A^*Az, \mu^{-1}).$$

If we substitute $x(z)$ into the formula for $g_{\text{smooth}}(z)$ and simplify carefully, we find that

$$g_{\text{smooth}}(z) = -\tfrac{1}{2}\mu^{-1}\|\,\text{SoftThreshold}(\mu x_0 - A^*Az, 1)\|_2^2 + \langle A^*y, z\rangle + c,$$

where c is a term that depends only on constants μ and x_0. In other words, to within an additive constant, $g_{\text{smooth}}(z)$ is a smooth approximation of the non-smooth term $I_{\ell_\infty}(-A^*Az) + \langle A^*y, z\rangle$ from (4.3.2), and indeed it converges to that function as $\mu \to 0$.

For the dual update, the generalized projection is

$$z_{k+1} \leftarrow \operatorname*{argmin}_z g_{\text{smooth}}(z_k) + \langle \nabla g_{\text{smooth}}(z_k), z - z_k\rangle + \tfrac{1}{2t_k}\|z - z_k\|_2^2 + \delta\|z\|_1. \tag{4.3.5}$$

A solution to this minimization can also be expressed in terms of the soft thresholding operation:

$$z_{k+1} \leftarrow \text{SoftThreshold}(z_k - t_kA^*(y - Ax(z_k)), t_k\delta).$$

4.3.3 Implementation

To solve the model presented in §4.3.2, we considered first-order projected gradient solvers. After some experimentation, we concluded that the Auslender and Teboulle first-order variant [AT06, Tse08] is a good choice for this model. We discuss this and other variants in more detail in §4.5, so for now we will simply present the basic algorithm in Listing 1 below. Note that the dual update used differs slightly from (4.3.5) above: the gradient ∇g_{smooth} is evaluated at y_k, not z_k, and the step size in the generalized projection is multiplied by θ_k^{-1}. Each iteration requires two applications of both A and A^*, and is computationally inexpensive when a fast matrix-vector multiply is available.

Listing 1 Algorithm for the smoothed Dantzig selector

Require: $z_0, x_0 \in \mathbb{R}^n$, $\mu > 0$, step sizes $\{t_k\}$

1: $\theta_0 \leftarrow 1$, $v_0 \leftarrow z_0$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: $\quad y_k \leftarrow (1 - \theta_k)v_k + \theta_k z_k$
4: $\quad x_k \leftarrow \text{SoftThreshold}(x_0 - \mu^{-1}A^*Ay_k, \mu^{-1})$.
5: $\quad z_{k+1} \leftarrow \text{SoftThreshold}(z_k - \theta_k^{-1}t_k A^*(y - Ax_k), \theta_k^{-1}t_k\delta)$
6: $\quad v_{k+1} \leftarrow (1 - \theta_k)v_k + \theta_k z_{k+1}$
7: $\quad \theta_{k+1} \leftarrow 2/(1 + (1 + 4/\theta_k^2)^{1/2})$

It is known that for a fixed step size $t_k \triangleq t \leq \mu/\|A^*A\|_2^2$, the above algorithm converges in the sense that $\bar{g}_\mu(z^*) - \bar{g}_\mu(z_k) = \mathcal{O}(k^{-2})$. Performance can be improved through the use of a backtracking line search for z_k, as discussed in §4.5.3 (see also [BT09]). Further, fairly standard arguments in convex analysis show that the sequence $\{x_k\}$ converges to the unique solution to (4.3.4).

4.3.4 Exact penalty

Theorem 3.1 in [CCS10] can be adapted to show that as $\mu \to 0$, the solution to (4.3.4) converges to a solution to (4.1.2). But an interesting feature of the Dantzig selector model in particular is that if $\mu < \mu_0$ for μ_0 sufficiently small, the solutions to the Dantzig selector and to its perturbed variation (4.3.4) coincide; that is, $x^\star = x_\mu^\star$. In fact, this phenomenon holds for any linear program (LP).

Theorem 4.3.1 (Exact penalty)**.** *Consider an arbitrary LP with objective $\langle c, x\rangle$ and having an optimal solution (i.e., the optimal value is not $-\infty$) and let Q be a positive semidefinite matrix. Then there is a $\mu_0 > 0$ such that if $0 < \mu \leq \mu_0$, any solution to the perturbed problem with objective $\langle c, x\rangle + \frac{1}{2}\mu\langle x - x_0, Q(x - x_0)\rangle$ is a solution to LP. Among all the solutions to LP, the solutions to the perturbed problem are those minimizing the quadratic penalty. In particular, in the (usual) case where the LP solution is unique, the solution to the perturbed problem is unique and they coincide.*

The theorem is not difficult to prove, and first appeared in 1979 due to Mangasarian [MM79]. The special case of noiseless basis pursuit was recently analyzed in [Yin10] using different techniques. More general results, allowing a range of penalty functions, were proved in [FT07]. Our theorem is a special case of [FT07] and the earlier results, but we present the proof in Appendix 4.11 since it uses a different technique than [FT07], is directly applicable to our method in this form, and provides useful intuition. The result, when combined with our continuation techniques in §4.5.5, is also a new look at known results about the finite termination of the proximal point algorithm when applied to linear programs [PT74, Ber75].

As a consequence of the theorem, the Dantzig selector and noiseless basis pursuit, which are

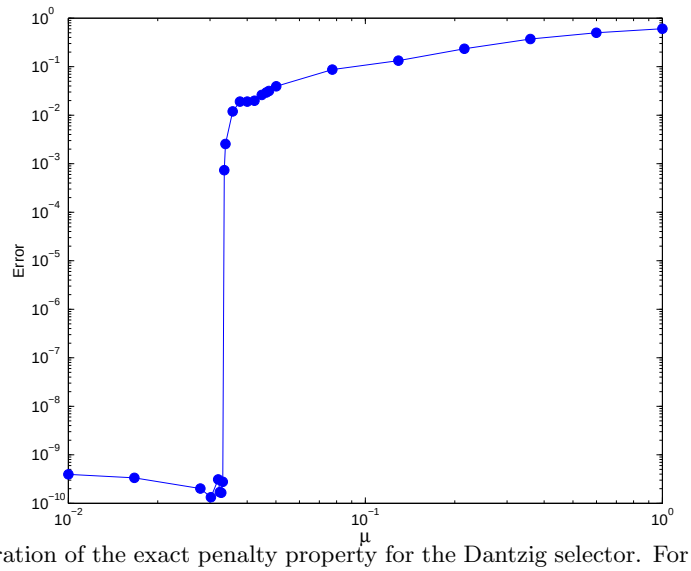


Figure 4.2: Demonstration of the exact penalty property for the Dantzig selector. For $\mu \leq \mu_0 \approx 0.025$, the solution to the smoothed model coincides with the original model to solver precision.

both linear programs, have the exact penalty property. To see why it holds for the Dantzig selector, recast it as the LP

$$\begin{array}{ll} \text{minimize} & \langle \mathbf{1}, u \rangle \\ \text{subject to} & -u \leq x \leq u \\ & -\delta \mathbf{1} \leq A^*(y - Ax) \leq \delta \mathbf{1}, \end{array}$$

with optimization variables $(x, u) \in \mathbb{R}^{2n}$. Then the perturbation $\frac{1}{2}\mu\|x - x_0\|^2$ corresponds to a quadratic perturbation with a diagonal $Q \succeq 0$ obeying $Q_{ii} = 1$ for $1 \leq i \leq n$ and $Q_{ii} = 0$ for $n + 1 \leq i \leq 2n$.

An illustration of this phenomenon is provided in Figure 4.2. For this example, a Dantzig selector model was constructed for a data set built from a DCT measurement matrix of size 512×4096. The exact solution $x^\star$ was constructed to have 60 dB of dynamic range and 129 nonzeros, using the techniques of Appendix 4.12.[7] The figure plots the smoothing error $\|x^\star - x_\mu\|_2 / \|x^\star\|_2$ as a function of μ; below approximately $\mu \approx 0.025$, the error drops rapidly to solver precision.

Unfortunately, compressed sensing models that are not equivalent to linear programs do not in general have the exact penalty property. For instance, it is possible to construct a counter-example for LASSO where the constraint is of the form $\|y - Ax\| \leq \epsilon$. However, the result still suggests that for small ϵ and μ, the LASSO and its smoothed version are similar.

[7] In this example, we verified that the optimal solution was unique by converting to a standard form linear program and used a dual optimal point as a certificate to prove the primal optimal variable is always a vertex.

4.3.5 Alternative models

As previously mentioned, different conic formulations result in different algorithms. To illustrate, consider the first alternative (4.2.1) proposed in §4.2.1, which represents the Dantzig selector constraint via linear inequalities. The conic form is

$$\begin{array}{ll} \text{minimize} & \|x\|_1 \\ \text{subject to} & \begin{bmatrix} \delta\mathbf{1} + A^*(y - Ax) \\ \delta\mathbf{1} - A^*(y - Ax) \end{bmatrix} \in \mathbb{R}^{2n}_+. \end{array} \tag{4.3.6}$$

The dual variable $\lambda \triangleq (\lambda_1, \lambda_2)$ must also lie in $\mathbb{R}^{2n}_+$. The Lagrangian of the smoothed model is

$$\mathcal{L}_\mu(x; \lambda) = \|x\|_1 + \tfrac{1}{2}\mu\|x - x_0\|_2^2 - \langle \lambda_1, \delta\mathbf{1} + A^*(y - Ax)\rangle - \langle \lambda_2, \delta\mathbf{1} - A^*(y - Ax)\rangle$$

and its unique minimizer is given by the soft-thresholding operation

$$x(\lambda) = \text{SoftThreshold}(x_0 - \mu^{-1}A^*A(\lambda_1 - \lambda_2), \mu^{-1}).$$

We cannot eliminate any variables by reducing to composite form, so we stay with the standard smoothed dual function $g_\mu(\lambda) = \inf_x L_\mu(x; \lambda)$, whose gradient is

$$\nabla g_\mu(\lambda) = \begin{bmatrix} -\delta\mathbf{1} - A^*(y - Ax(\lambda)) \\ -\delta\mathbf{1} + A^*(y - Ax(\lambda)) \end{bmatrix}.$$

The dual update is a true projection

$$\begin{aligned} \lambda_{k+1} &= \operatorname*{argmin}_{\lambda \in \mathbb{R}^{2n}_+} -g_\mu(\lambda_k) - \langle \nabla g_\mu(\lambda_k), \lambda - \lambda_k\rangle + \tfrac{1}{2}t_k^{-1}\|\lambda - \lambda_k\|_2^2 \\ &= \operatorname*{argmin}_{\lambda \in \mathbb{R}^{2n}_+} \|\lambda - \lambda_k - t_k\nabla g_\mu(\lambda_k)\|_2 \end{aligned} \tag{4.3.7}$$

whose solution is simply the non-negative portion of a standard gradient step:

$$\lambda_{k+1} = \text{Pos}(\lambda_k + t_k\nabla g_\mu(\lambda_k)), \quad [\text{Pos}(z)]_i \triangleq \begin{cases} z_i, & z_i > 0, \\ 0, & z_i \le 0. \end{cases}$$

In order to better reveal the similarities between the two models, let us define $\bar{z} \triangleq \lambda_1 - \lambda_2$ and $\bar{\tau} \triangleq \mathbf{1}^*(\lambda_1 + \lambda_2)$. Then we have $\|\bar{z}\|_1 \leq \bar{\tau}$, and the Lagrangian and its minimizer become

$$\mathcal{L}_\mu(x;\lambda) = \|x\|_1 + \tfrac{1}{2}\mu\|x - x_0\|_2^2 - \langle \bar{z}, A^*(y - Ax)\rangle - \delta\bar{\tau},$$
$$x(\lambda) = \text{SoftThreshold}(x_0 - \mu^{-1}A^*A\bar{z}, \mu^{-1}),$$

which are actually identical to the original norm-based model. The difference lies in the dual update. It is possible to show that the dual update for the original model is equivalent to

$$z_{k+1} = \Sigma \cdot \underset{\lambda \in \mathbb{R}^{2n}_+}{\text{argmin}} -g_\mu(\lambda_k) - \langle \nabla g_\mu(\lambda_k), \lambda - \lambda_k\rangle + \tfrac{1}{2}t_k^{-1}\|\Sigma(\lambda - \lambda_k)\|_2^2 \tag{4.3.8}$$

for $\Sigma = [+I, -I]$. In short, the dual function is linear in the directions of $\lambda_1 + \lambda_2$, so eliminating them from the proximity term would yield true numerical equivalence to the original model.

4.4 Further instantiations

Now that we have seen the mechanism for instantiating a particular instance of a compressed sensing problem, let us show how the same approach can be applied to several other types of models. Instead of performing the full, separate derivation for each case, we first provide a template for our standard form. Then, for each specific model, we show the necessary modifications to the template to implement that particular case.

4.4.1 A generic algorithm

A careful examination of our derivations for the Dantzig selector, as well as the developments in §4.2.6, provide a clear path to generalizing Listing 1 above. We require implementations of the linear operators $\bar{\mathcal{A}}$ and its adjoint $\bar{\mathcal{A}}^*$, and values of the constants $\bar{b}$, b_τ; recall that these are the partitioned versions of $\mathcal{A}$ and b as described in §4.2.5. We also need to be able to perform the two-sequence recursion (4.2.22), modified to include the step size multiplier θ_k and an adjustable centerpoint x_0 in the proximity function.

Armed with these computational primitives, we present in Listing 2 a generic equivalent of the algorithm employed for the Dantzig selector in Listing 1. It is important to note that this particular variant of the optimal first-order methods may not be the best choice for every model; nevertheless each variant uses the same computational primitives.

Listing 2 Generic algorithm for the conic standard form

Require: $z_0, x_0 \in \mathbb{R}^n$, $\mu > 0$, step sizes $\{t_k\}$

1: $\theta_0 \leftarrow 1$, $v_0 \leftarrow z_0$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: $\quad y_k \leftarrow (1 - \theta_k)v_k + \theta_k z_k$
4: $\quad x_k \leftarrow \operatorname{argmin}_x f(x) + \mu d(x - x_0) - \langle \bar{\mathcal{A}}^*(y_k), x \rangle$
5: $\quad z_{k+1} \leftarrow \operatorname{argmin}_z h(z) + \frac{\theta_k}{2t_k}\|z - z_k\|^2 + \langle \bar{\mathcal{A}}(x_k) + \bar{b}, z \rangle$
6: $\quad v_{k+1} \leftarrow (1 - \theta_k)v_k + \theta_k z_{k+1}$
7: $\quad \theta_{k+1} \leftarrow 2/(1 + (1 + 4/\theta_k^2)^{1/2})$

In the next several sections, we show how to construct first-order methods for a variety of models. We will do so by replacing lines 4–5 of Listing 2 with appropriate substitutions and simplified expressions for each.

4.4.2 The LASSO

The conic form for the smoothed LASSO is

$$\begin{array}{ll} \text{minimize} & \|x\|_1 + \frac{1}{2}\mu\|x - x_0\|_2^2 \\ \text{subject to} & (y - Ax, \epsilon) \in \mathcal{L}_2^n, \end{array} \tag{4.4.1}$$

where $\mathcal{L}_2^n$ is the epigraph of the Euclidean norm. The Lagrangian is

$$\mathcal{L}_\mu(x; z, \tau) = \|x\|_1 + \tfrac{1}{2}\mu\|x - x_0\|_2^2 - \langle z, y - Ax \rangle - \epsilon\tau.$$

The dual variable $\lambda = (z, \tau)$ is constrained to lie in the dual cone, which is also $\mathcal{L}_2^n$. Eliminating τ (assuming $\epsilon > 0$) yields the composite dual form

$$\text{maximize} \quad \inf_x \|x\|_1 + \tfrac{1}{2}\mu\|x - x_0\|_2^2 - \langle z, y - Ax \rangle - \epsilon\|z\|_2.$$

The primal projection with $f(x) = \|x\|_1$ is the same soft-thresholding operation used for the Dantzig selector. The dual projection involving $h(z) = \epsilon\|z\|_2$, on the other hand, is

$$z_{k+1} = \operatorname*{argmin}_z \epsilon\|z\|_2 + \tfrac{\theta_k}{2t_k}\|z - z_k\|_2^2 + \langle \tilde{x}, z \rangle = \text{Shrink}(z_k - \theta_k^{-1} t_k \tilde{x}, \theta_k^{-1} t_k \epsilon),$$

where $\tilde{x} \triangleq y - Ax_k$ and Shrink is an ℓ_2-shrinkage operation

$$\text{Shrink}(z,t) \triangleq \max\{1 - t/\|z\|_2, 0\} \cdot z = \begin{cases} 0, & \|z\|_2 \leq t, \\ (1 - t/\|z\|_2) \cdot z, & \|z\|_2 > t. \end{cases}$$

The resulting algorithm excerpt is given in Listing 3.

Listing 3 Algorithm excerpt for LASSO

4: $x_k \leftarrow \text{SoftThreshold}(x_0 - \mu^{-1} A^* y_k, \mu^{-1})$

5: $z_{k+1} \leftarrow \text{Shrink}(z_k - \theta_k^{-1} t_k (y - Ax_k), \theta_k^{-1} t_k \epsilon)$

4.4.3 Nuclear-norm minimization

Extending this approach to the nuclear-norm minimization problem

$$\begin{aligned} &\text{minimize} && \|X\|_* \\ &\text{subject to} && \|y - \mathcal{A}(X)\|_2 \leq \epsilon \end{aligned} \tag{4.4.2}$$

is straightforward. The composite smoothed dual form is

$$\text{maximize} \quad \inf_X \|X\|_* + \mu d(X) - \langle z, y - \mathcal{A}(X) \rangle - \epsilon \|z\|_2,$$

and the dual projection corresponds very directly to the LASSO. Choosing $d(X) = \frac{1}{2}\|X - X_0\|_F^2$ leads to a primal projection given by the soft-thresholding of singular values:

$$X_k = \text{SoftThresholdSingVal}(X_0 - \mu^{-1} \mathcal{A}^*(y_k), \mu^{-1}). \tag{4.4.3}$$

The SoftThresholdSingVal operation obeys [CCS10]

$$\text{SoftThresholdSingVal}(X,t) = U \cdot \text{SoftThreshold}(\Sigma, t) \cdot V^*,$$

where $X = U\Sigma V^*$ is any singular value decomposition of Z, and SoftThreshold(Σ) applies soft-thresholding to the singular values (the diagonal elements of Σ). This results in the algorithm excerpt presented in Listing 4.

Listing 4 Algorithm for nuclear-norm minimization (LASSO constraint)

4: $X_k \leftarrow \text{SoftThresholdSingVal}(X_0 - \mu^{-1} \mathcal{A}^*(y_k), \mu^{-1})$

5: $z_{k+1} \leftarrow \text{Shrink}(z_k - \theta_k^{-1} t_k (y - \mathcal{A}(X_k)), \theta_k^{-1} t_k \epsilon)$

Another constraint of interest is of Dantzig-selector type [CP10b] so that one is interested in

$$\begin{array}{ll} \text{minimize} & \|X\|_* \\ \text{subject to} & \|\mathcal{A}^*(\mathcal{A}(X) - y)\| \leq \delta. \end{array} \tag{4.4.4}$$

The cone of interest is, therefore, $\mathcal{K} = \{(X,t) : \|X\| \leq t\}$ and the dual cone is $\mathcal{K}^* = \{(X,t) : \|X\|_* \leq t\}$. The derivation proceeds as before, and the composite dual problem becomes

$$\text{maximize} \quad \inf_X \|X\|_* + \tfrac{1}{2}\mu\|X - X_0\|_F^2 - \langle Z, \mathcal{A}^*(y - \mathcal{A}(X))\rangle - \delta\|Z\|_*.$$

The gradient of the smooth portion has a Lipschitz continuous gradient with constant at most $\mu^{-1}\|\mathcal{A}^*\mathcal{A}\|^2$, and singular value thresholding is now used to perform the dual projection. The resulting excerpt is given in Listing 5.

Listing 5 Algorithm for nuclear-norm minimization (Dantzig-selector constraint)

```
4: X_k ← SoftThresholdSingVal(X_0 − μ^{-1}A*(A(Y_k)), μ^{-1})
5: Z_{k+1} ← SoftThresholdSingVal(Z_k − θ_k^{-1} t_k A*(y − A(X_k)), θ_k^{-1} t_k δ)
```

4.4.4 ℓ_1-analysis

We are now interested in

$$\begin{array}{ll} \text{minimize} & \|Wx\|_1 \\ \text{subject to} & \|y - Ax\|_2 \leq \epsilon, \end{array} \tag{4.4.5}$$

where the matrix W is arbitrary. This problem is frequently discussed in signal processing and is sometimes referred to as the method of ℓ_1-*analysis*. As explained in the introduction, this is a challenging problem as stated, because a generalized projection for $f(x) = \|Wx\|_1$ does not have an analytical form.

Let us apply our techniques to an alternative conic formulation

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \|Wx\|_1 \leq t, \\ & \|y - Ax\|_2 \leq \epsilon, \end{array}$$

where t is a new scalar variable. The dual variables are $\lambda = (z^{(1)}, \tau^{(1)}, z^{(2)}, \tau^{(2)})$, where

$$\|z^{(1)}\|_\infty \leq \tau^{(1)}, \quad \|z^{(2)}\|_2 \leq \tau^{(2)},$$

and the Lagrangian is given by

$$\mathcal{L}(x,t;z^{(1)},\tau^{(1)},z^{(2)},\tau^{(2)}) = t - \langle z^{(1)}, Wx\rangle - \tau^{(1)}t - \langle z^{(2)}, y - Ax\rangle - \epsilon\tau^{(2)}.$$

The Lagrangian is unbounded unless $\tau^{(1)} = 1$; and we can eliminate $\tau^{(2)}$ in our standard fashion as well. These simplifications yield a dual problem

$$\begin{array}{ll} \text{maximize} & \langle y, z^{(2)}\rangle - \epsilon\|z^{(2)}\|_2 \\ \text{subject to} & A^*z^{(2)} - W^*z^{(1)} = 0, \\ & \|z^{(1)}\|_\infty \leq 1. \end{array}$$

To apply smoothing to this problem, we use a standard proximity function $d(x) = \frac{1}{2}\|x - x_0\|^2$. (Setting $\tau^{(1)} = 1$ causes t to be eliminated from the Lagrangian, so it need not appear in our proximity term.) The dual function becomes

$$g_\mu(z^{(1)}, z^{(2)}) = \inf_x \tfrac{1}{2}\mu\|x - x_0\|_2^2 - \langle z^{(1)}, Wx\rangle - \langle z^{(2)}, y - Ax\rangle - \epsilon\|z^{(2)}\|$$

and the minimizer $x(z)$ is simply

$$x(z) = x_0 + \mu^{-1}(W^*z^{(1)} - A^*z^{(2)}).$$

Now onto the dual projection

$$z_{k+1} = \operatorname*{argmin}_{z:\ \|z^{(2)}\|_\infty \leq 1} \epsilon\|z^{(2)}\|_2 + \tfrac{\theta_k}{2t_k}\|z - z_k\|^2 + \langle \tilde{x}, z\rangle,$$

where $\tilde{x} = (Wx(z), y - Ax(z))$. This will certainly converge if the step sizes t_k are chosen properly. However, if W and A have significantly different scaling, the performance of the algorithm may suffer. Our idea is to apply different step sizes $t_k^{(i)}$ to each dual variable

$$z_{k+1} = \operatorname*{argmin}_{z:\ \|z^{(2)}\|_\infty \leq 1} \epsilon\|z^{(2)}\|_2 + \langle \tilde{x}, z\rangle + \tfrac{1}{2}\theta_k \sum_{i=1}^{2} (t_k^{(i)})^{-1}\|z^{(i)} - z_k^{(i)}\|_2^2$$

in a fashion that preserves the convergence properties of the method. The minimization problem over z is separable, and the solution is given by

$$z_k^{(1)} = \text{Trunc}(y_k^{(1)} - \theta_k^{-1}t_k^{(1)}\tilde{x}^{(1)}, \theta_k^{-1}t_k^{(1)}) \tag{4.4.6a}$$

$$z_k^{(2)} = \text{Shrink}(y_k^{(2)} - \theta_k^{-1}t_k^{(2)}\tilde{x}^{(2)}, \theta_k^{-1}t_k^{(2)}\epsilon), \tag{4.4.6b}$$

where the truncation operator is given element-wise by

$$\text{Trunc}(z,\tau) = \text{sgn}(z)\cdot\min\{|z|,\tau\} = \begin{cases} z, & |z|\le\tau, \\ \tau\ \text{sgn}(z), & |z|\ge\tau. \end{cases}$$

In our current tests, we fix $t_k^{(2)} = \alpha t_k^{(1)}$, where we choose $\alpha = \|W\|^2/\|A\|^2$, or some estimate thereof. This is numerically equivalent to applying a single step size to a scaled version of the original problem, so convergence guarantees remain. In future work, however, we intend to develop a practical method for adapting each step size separately.

The algorithm excerpt is given in Listing 6.

Listing 6 Algorithm excerpt for ℓ_1-analysis

4: $x_k \leftarrow x_0 + \mu^{-1}(Wy_k^{(1)} - A^*y_k^{(2)})$
5: $z_{k+1}^{(1)} \leftarrow \text{Trunc}(y_k^{(1)} - \theta_k^{-1}t_k^{(1)}Wx_k, \theta_k^{-1}t_k^{(1)})$
$z_{k+1}^{(2)} \leftarrow \text{Shrink}(y_k^{(2)} - \theta_k^{-1}t_k^{(2)}(y - Ax_k), \theta_k^{-1}t_k^{(2)}\epsilon)$

4.4.5 Total-variation minimization

We now wish to solve

$$\begin{aligned} &\text{minimize} && \|x\|_{\text{TV}} \\ &\text{subject to} && \|y - Ax\|_2 \le \epsilon \end{aligned} \tag{4.4.7}$$

for some image array $x \in \mathbb{R}^{n^2}$ where $\|x\|_{\text{TV}}$ is the total-variation introduced in §4.1.1. We can actually cast this as a *complex* ℓ_1-analysis problem

$$\begin{aligned} &\text{minimize} && \|Dx\|_1 \\ &\text{subject to} && \|y - Ax\|_2 \le \epsilon \end{aligned}$$

where $D : \mathbb{R}^{n^2} \to \mathbb{C}^{(n-1)^2}$ is a matrix representing the linear operation that places horizontal and vertical differences into the real and imaginary elements of the output, respectively:

$$[Dx]_{ij} \triangleq (x_{i+1,j} - x_{i,j}) + \sqrt{-1}\cdot(x_{i,j+1} - x_{i,j}), \quad 1 \le i < n,\ 1 \le j < n.$$

Writing it in this way allows us to adapt our ℓ_1-analysis derivations directly. The smoothed dual function becomes

$$g_\mu(z^{(1)}, z^{(2)}) = \inf_x \tfrac{1}{2}\mu\|x - x_0\|_2^2 - \langle z^{(1)}, Dx\rangle - \langle z^{(2)}, y - Ax\rangle - \epsilon\|z^{(2)}\|_2,$$

where $z^{(2)} \in \mathbb{R}^m$ is identical to the previous problem, and $z^{(1)} \in \mathbb{C}^{(n-1)^2}$ satisfies $\|z^{(1)}\|_\infty \leq 1$. Supporting a complex $z^{(1)}$ requires two modifications. First, we must be careful to use the real-valued inner product

$$\langle z^{(1)}, Dx \rangle \triangleq \Re((z^{(1)})^H Dx) = (\Re(D^H z^{(1)}))^T x.$$

Second, the projection requires a complex version of the truncation operation:

$$[\mathrm{CTrunc}(z, \tau)]_k = \min\{1, \tau/|z_k|\} \cdot z_k = \begin{cases} z_k, & |z_k| \leq \tau, \\ \tau z_k/|z_k|, & |z_k| \geq \tau. \end{cases}$$

The algorithm excerpt is given in Listing 7.

Listing 7 Algorithm excerpt for TV minimization

```
4: x_k ← x_0 + μ^{-1}(ℜ(D* y_k^(1)) − A* y_k^(2))
5: z_{k+1}^(1) ← CTrunc(y_k^(1) − θ_k^{-1} t_k^(1) D x_k, θ_k^{-1} t_k^(1))
   z_{k+1}^(2) ← Shrink(y_k^(2) − θ_k^{-1} t_k^(2) (y − A x_k), θ_k^{-1} t_k^(2) ϵ)
```

4.4.6 Combining ℓ_1 analysis and total-variation minimization

We could multiply our examples indefinitely, and we close this section by explaining how one could solve the problem (4.1.4), namely that of finding the minimum of the weighted combination $\|Wx\|_1 + \lambda\|x\|_{\mathrm{TV}}$ subject to quadratic constraints. This problem can be recast as

$$\begin{aligned} \text{minimize} \quad & t + \lambda s \\ \text{subject to} \quad & \|Wx\|_1 \leq t \\ & \|Dx\|_1 \leq s \\ & \|Ax - y\|_2 \leq \epsilon \end{aligned} \tag{4.4.8}$$

and the strategy is exactly the same as before. The only difference with §4.4.4 and §4.4.5 is that the dual variable now belongs to a direct product of three cones instead of two. Otherwise, the strategy is the same, and the path is so clear that we prefer leaving the details to the reader, who may also want to consult the user guide which accompanies the software release [BCG10b].

4.5 Implementing first-order methods

So far we have demonstrated how to express compressed sensing problems in a specific conic form that can be solved using optimal first-order methods. In this section, we discuss a number of practical matters that arise in the implementation of optimal first-order methods. This work applies

to a wider class of models than those presented in this chapter; therefore, we will set aside our conic formulations and present the first-order algorithms in their more native form.

4.5.1 Introduction

The problems of interest in this chapter can be expressed in an unconstrained composite form

$$\text{minimize} \quad \phi(z) \triangleq g(z) + h(z), \tag{4.5.1}$$

where $g, h : \mathbb{R}^n \to (\mathbb{R} \cup +\infty)$ are convex functions with g smooth and h non-smooth. (To be precise, the dual functions in our models are concave, so we consider their convex negatives here.) Convex constraints are readily supported by including their corresponding indicator functions into h.

First-order methods solve (4.5.1) with repeated calls to a generalized projection, such as

$$z_{k+1} \leftarrow \operatorname*{argmin}_z g(z_k) + \langle \nabla g(z_k), z - z_k \rangle + \tfrac{1}{2t_k}\|z - z_k\|^2 + h(z), \tag{4.5.2}$$

where $\|\cdot\|$ is a chosen norm and t_k is the step size control. Proofs of global convergence depend upon the right-hand approximation satisfying an upper bound property

$$\phi(z_{k+1}) \leq g(z_k) + \langle \nabla g(z_k), z_{k+1} - z_k \rangle + \tfrac{1}{2t_k}\|z_{k+1} - z_k\|^2 + h(z_{k+1}). \tag{4.5.3}$$

This bound is certain to hold for sufficiently small t_k; but to ensure global convergence, t_k must be bounded away from zero. This is typically accomplished by assuming that the gradient of g satisfies a generalized Lipschitz criterion,

$$\|\nabla g(x) - \nabla g(y)\|_* \leq L\|x - y\| \quad \forall x, y \in \operatorname{dom} \phi, \tag{4.5.4}$$

where $\|\cdot\|_*$ is the dual norm; that is, $\|w\|_* = \sup\{\langle z, w \rangle \,|\, \|z\| \leq 1\}$. Then the bound (4.5.3) is guaranteed to hold for any $t_k \leq L^{-1}$. Under these conditions, convergence to ϵ accuracy—that is, $\phi(z_k) - \inf_z \phi(z) \leq \epsilon$—is obtained in $\mathcal{O}(L/\epsilon)$ iterations for a simple algorithm based on (4.5.2) known variously as the forward-backward algorithm or proximal gradient descent, which dates to at least [FM81]. This bound on the iterations improves to $\mathcal{O}(\sqrt{L/\epsilon})$ for the so-called optimal or accelerated methods [Nes83,Nes88,Nes07,Tse08]. These optimal methods vary the calculation (4.5.2) slightly, but the structure and complexity remain the same.

4.5.2 The variants

Optimal first-order methods have been a subject of much study in the last decade by many different authors. In 2008, Tseng presented a nearly unified treatment of the most commonly cited methods,

and provided simplified proofs of global convergence and complexity [Tse08]; this paper was both an aid and an inspiration for the modular template framework we present.

We constructed implementations of five of the optimal first-order variants as well as a standard projected gradient algorithm. To simplify discussion, we have given each variant a 2–3 character label. Listing 8 depicts N07, a variation of the method described by Nesterov in [Nes04, Nes07].

Listing 8 Nesterov's 2007 algorithm (N07)

Require: $z_0 \in \operatorname{dom} \phi$, Lipschitz estimate L

1: $\bar{z}_0 \leftarrow z_0$, $\theta_0 \leftarrow 1$

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

3: $\quad y_k \leftarrow (1 - \theta_k) z_k + \theta_k \bar{z}_k$

4: $\quad \bar{z}_{k+1} \leftarrow \operatorname{argmin}_z \langle \theta_k^2 \sum_{i=0}^{k} \theta_i^{-1} \nabla g(y_i), z \rangle + \frac{1}{2} \theta_k^2 L \|z - z_0\|^2 + h(z)$

5: $\quad z_{k+1} \leftarrow \operatorname{argmin}_z \langle \nabla g(y_k), z \rangle + \frac{1}{2} L \|z - y_k\|^2 + h(z)$

6: $\quad \theta_{k+1} \leftarrow 2/(1 + (1 + 4/\theta_k^2)^{1/2})$

The other variants can be described described simply by replacing lines 4–5 as follows.

- TS: Tseng's single-projection simplification of N07 [Tse08]

  4: $\bar{z}_{k+1} \leftarrow \operatorname{argmin}_z \langle \theta_k^2 \sum_{i=0}^{k} \theta_i^{-1} \nabla g(y_i), z \rangle + \frac{1}{2} \theta_k^2 L \|z - z_0\|^2 + h(z)$

  5: $z_{k+1} \leftarrow (1 - \theta_k) z_k + \theta_k \bar{z}_{k+1}$

- LLM: Lan, Lu, and Monteiro's modification of N07 [LLM09]

  4: $\bar{z}_{k+1} \leftarrow \operatorname{argmin}_z \langle \nabla g(y_k), z \rangle + \frac{1}{2} \theta_k L \|z - \bar{z}_k\|^2 + h(z)$

  5: $z_{k+1} \leftarrow \operatorname{argmin}_z \langle \nabla g(y_k), z \rangle + \frac{1}{2} L \|z - y_k\|^2 + h(z)$

- AT: Auslender and Teboulle's method from [AT06]

  4: $\bar{z}_{k+1} \leftarrow \operatorname{argmin}_z \langle \nabla g(y_k), z \rangle + \frac{1}{2} \theta_k L \|z - \bar{z}_k\|^2 + h(z)$

  5: $z_{k+1} \leftarrow (1 - \theta_k) z_k + \theta_k \bar{z}_{k+1}$

- N83: Nesterov's 1983 method [Nes83,Nes05]; see also FISTA [BT09]

  4: $z_{k+1} \leftarrow \operatorname{argmin}_z \langle \nabla g(y_k), z \rangle + \frac{1}{2} L \|z - y_k\|^2 + h(z)$

  5: Compute $\bar{z}_{k+1}$ to satisfy $z_{k+1} = (1 - \theta_k) z_k + \theta_k \bar{z}_{k+1}$.

- GRA: The classical projected gradient generalization

  4: $z_{k+1} \leftarrow \operatorname{argmin}_z \langle \nabla g(y_k), z \rangle + \frac{1}{2} L \|z - y_k\|^2 + h(z)$

  5: $\bar{z}_{k+1} \leftarrow z_{k+1}$

Following Tseng's lead, we have rearranged steps and renamed variables, compared to their original sources, so that the similarities are more apparent. This does mean that simpler expressions of some

of these algorithms are possible, specifically for TS, AT, N83, and GRA. Note in particular that GRA does not use the parameter θ_k.

Given their similar structure, it should not be surprising that these algorithms, except GRA, achieve nearly identical theoretical iteration performance. Indeed, it can be shown that if $z^\star$ is an optimal point for (4.5.1), then for any of the optimal variants,

$$\phi(z_{k+1}) - \phi(z^\star) \leq \tfrac{1}{2} L \theta_k^2 \|z_0 - z^\star\|^2 \leq 2Lk^{-2}\|z_0 - z^\star\|^2. \tag{4.5.5}$$

Thus the number of iterations required to reach ϵ optimality is at most $\lceil \sqrt{2L/\epsilon}\|z_0 - z^\star\|^2 \rceil$ (again, except GRA). Tighter bounds can be constructed in some cases but the differences remain small.

Despite their obvious similarity, the algorithms do have some key differences worth noting. First of all, the sequence of points y_k generated by the N83 method may sometimes lie outside of $\operatorname{dom}\phi$. This is not an issue for our applications, but it might be for those where $g(z)$ may not be differentiable everywhere. Secondly, N07 and LLM require two projections per iteration, while the others require only one. Two-projection methods would be preferred only if the added cost results in a comparable reduction in the number of iterations required. Theory does not support this trade-off, but the results in practice may differ; see §4.7.1 for a single comparison.

4.5.3 Step size adaptation

All of the algorithms involve the global Lipschitz constant L. Not only is this constant often difficult or impractical to obtain, the step sizes it produces are often too conservative, since the global Lipschitz bound (4.5.4) may not be tight in the neighborhood of the solution trajectory. Reducing L artificially can improve performance, but reducing it too much can cause the algorithms to diverge. Our experiments suggest that the transition between convergence and divergence is very sharp.

A common solution to such issues is backtracking: replace the global constant L with a per-iteration estimate L_k that is increased as local behavior demands it. Examining the convergence proofs of Tseng reveals that the following condition is sufficient to preserve convergence (see [Tse08], Propositions 1, 2, and 3):

$$g(z_{k+1}) \leq g(y_k) + \langle \nabla g(y_k), z_{k+1} - y_k \rangle + \tfrac{1}{2} L_k \|z_{k+1} - y_k\|^2. \tag{4.5.6}$$

If we double the value of L_k every time a violation of (4.5.6) occurs, for instance, then L_k will satisfy $L_k \geq L$ after no more than $\lceil \log_2(L/L_0) \rceil$ backtracks, after which the condition must hold for all subsequent iterations. Thus strict backtracking preserves global convergence. A simple improvement to this approach is to update L_k with $\max\{2L_k, \hat{L}\}$, where $\hat{L}$ is the smallest value of L_k that would satisfy (4.5.6). To determine an initial estimate L_0, we can select any two points z_0, z_1 and use the

formula

$$L_0 = \|\nabla g(z_0) - \nabla g(z_1)\|_* / \|z_0 - z_1\|.$$

Unfortunately, our experiments reveal that (4.5.6) suffers from severe cancellation errors when $g(z_{k+1}) \approx g(y_k)$, often preventing the algorithms from achieving high levels of accuracy. More traditional Armijo-style line search tests also suffer from this issue. We propose an alternative test that maintains fidelity at much higher levels of accuracy:

$$|\langle y_k - z_{k+1}, \nabla g(z_{k+1}) - \nabla g(y_k)\rangle| \leq \tfrac{1}{2} L_k \|z_{k+1} - y_k\|_2^2. \tag{4.5.7}$$

It is not difficult to show that (4.5.7) implies (4.5.6), so provable convergence is maintained. It is a more conservative test, however, producing smaller step sizes. So for best performance we prefer a hybrid approach: for instance, use (4.5.6) when $g(y_k) - g(z_{k+1}) \geq \gamma g(z_{k+1})$ for some small $\gamma > 0$, and use (4.5.7) otherwise to avoid the cancellation error issues.

A closer study suggests a further improvement. Because the error bound (4.5.5) is proportional to $L_k \theta_k^2$, simple backtracking will cause it to rise unnecessarily. This anomaly can be rectified by modifying θ_k as well as L_k during a backtracking step. Such an approach was adopted by Nesterov for N07 in [Nes07]; and with care it can be adapted to any of the variants. Convergence is preserved if

$$L_{k+1}\theta_{k+1}^2/(1 - \theta_{k+1}) \geq L_k \theta_k^2 \tag{4.5.8}$$

(see [Tse08], Proposition 1), which implies that the θ_k update in Line 6 of Listing 8 should be

$$\theta_{k+1} \leftarrow 2/(1 + (1 + 4L_{k+1}/\theta_k^2 L_k)^{1/2}). \tag{4.5.9}$$

With this update the monotonicity of the error bound (4.5.5) is restored. For N07 and TS, the update for $\bar{z}_{k+1}$ must also be modified as follows:

$$\bar{z}_{k+1} \leftarrow \operatorname*{argmin}_z \langle \theta_k^2 L_k \textstyle\sum_{i=0}^{k} (L_i \theta_i)^{-1} \nabla g(y_i), z\rangle + \tfrac{1}{2}\theta_k^2 L_k \|z - z_0\|^2 + h(z). \tag{4.5.10}$$

Finally, to improve performance we may consider *decreasing* the local Lipschitz estimate L_k when conditions permit. We have chosen a simple approach: attempt a slight decrease of L_k at each iteration; that is, $L_k = \alpha L_{k-1}$ for some fixed $\alpha \in (0, 1]$. Of course, doing so will guarantee that occasional backtracks occur. With judicious choice of α, we can balance step size growth for limited amounts of backtracking, minimizing the total number of function evaluations or projections. We have found that $\alpha = 0.9$ provides good performance in many applications.

4.5.4 Linear operator structure

Let us briefly reconsider the special structure of our compressed sensing models. In these problems, it is possible to express the smooth portion of our composite function in the form

$$g(z) = \bar{g}(\mathcal{A}^*(z)) + \langle b, z \rangle,$$

where $\bar{g}$ remains smooth, $\mathcal{A}$ is a linear operator, and b is a constant vector (see §4.2.5; we have dropped some overbars here for convenience). Computing a value of g requires a single application of $\mathcal{A}^*$, and computing its gradient also requires an application of $\mathcal{A}$. In many of our models, the functions $\bar{g}$ and h are quite simple to compute, so the linear operators account for the bulk of the computational costs. It is to our benefit, then, to utilize them as efficiently as possible.

For the prototypical algorithms of Section 4.5.2, each iteration requires the computation of the value and gradient of g at the single point y_k, so all variants require a single application each of $\mathcal{A}$ and $\mathcal{A}^*$. The situation changes when backtracking is used, however. Specifically, the backtracking test (4.5.6) requires the computation of g at a cost of one application of $\mathcal{A}^*$; and the alternative test (4.5.7) requires the gradient as well, at a cost of one application of $\mathcal{A}$. Fortunately, with a careful rearrangement of computations we can eliminate both of these additional costs for single-projection methods TS, AT, N83, and GRA, and one of them for the two-projection methods N07 and LLM.

Listing 9 AT variant with improved backtracking

Require: $z_0 \in \operatorname{dom}\phi$, $\hat{L} > 0$, $\alpha \in (0,1]$, $\beta \in (0,1)$
1: $\bar{z}_0 \leftarrow z_0$, $z_{\mathcal{A}0}, \bar{z}_{\mathcal{A}0} \leftarrow \bar{\mathcal{A}}^*(z_0)$, $\theta_{-1} = +\infty$, $L_{-1} = \hat{L}$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: $\quad L_k \leftarrow \alpha L_{k-1}$
4: $\quad$ **loop**
5: $\quad\quad \theta_k \leftarrow 2/(1 + (1 + 4L_k/\theta_{k-1}^2 L_{k-1})^{1/2})$ $(\theta_0 \triangleq 1)$
6: $\quad\quad y_k \leftarrow (1-\theta_k)z_k + \theta_k \bar{z}_k$, $y_{\mathcal{A},k} \leftarrow (1-\theta_k)z_{\mathcal{A},k} + \theta_k \bar{z}_{\mathcal{A},k}$
7: $\quad\quad \bar{g}_k \leftarrow \nabla \bar{g}(y_{\mathcal{A},k})$, $g_k \leftarrow \mathcal{A}\bar{g}_k + b$
8: $\quad\quad \bar{z}_{k+1} \leftarrow \operatorname{argmin}_z \langle g_k, z\rangle + h(z) + \frac{\theta_k}{2t_k}\|z - y_k\|^2$, $\bar{z}_{\mathcal{A},k+1} \leftarrow \mathcal{A}^*(z_{k+1})$
9: $\quad\quad z_{k+1} \leftarrow (1-\theta_k)z_k + \theta_k \bar{z}_{k+1}$, $z_{\mathcal{A},k+1} \leftarrow (1-\theta_k)z_{\mathcal{A},k} + \theta_k \bar{z}_{\mathcal{A},k}$
10: $\quad\quad \hat{L} \leftarrow 2\,|\langle y_{\mathcal{A},k} - z_{\mathcal{A},k+1}, \nabla\bar{g}(z_{\mathcal{A},k+1}) - \bar{g}_k\rangle| \,/\, \|z_{k+1} - y_k\|_2^2$
11: $\quad\quad$ **if** $L_k \geq \hat{L}$ **then break endif**
12: $\quad\quad L_k \leftarrow \max\{L_k/\beta, \hat{L}\}$

Listing 9 depicts this more efficient use of linear operators, along with the step size adaptation approach, using the AT variant. The savings come from two different additions. First, we maintain additional sequences $z_{\mathcal{A},k}$ and $\bar{z}_{\mathcal{A},k}$ to allow us to compute $y_{\mathcal{A},k} = \mathcal{A}^*(y_k)$ without a call to $\mathcal{A}^*$. Secondly, we take advantage of the fact that

$$\langle y_k - z_{k+1}, \nabla g(z_{k+1}) - \nabla g(y_k)\rangle = \langle y_{\mathcal{A},k} - z_{\mathcal{A},k+1}, \nabla\bar{g}(z_{\mathcal{A},k+1}) - \nabla\bar{g}(y_{\mathcal{A},k})\rangle,$$

which allows us to avoid having to compute the full gradient of g; instead we need to compute only the significantly less expensive gradient of $\bar{g}$.

4.5.5 Accelerated continuation

Several recent algorithms, such as FPC and NESTA [HYZ08, BBC11], have empirically found that continuation schemes greatly improve performance. The idea behind continuation is that we solve the problem of interest by solving a sequence of similar but easier problems, using the results of each subproblem to initialize or *warm start* the next one. Listing 10 below depicts a standard continuation loop for solving the generic conic problem in (4.1.14) with a proximity function $d(x) = \frac{1}{2}\|x - x_0\|_2^2$. We have used a capital X and loop count j to distinguish these iterates from the *inner loop* iterates x_k generated by a first-order method.

Listing 10 Standard continuation

Require: Y_0, $\mu_0 > 0$, $\beta < 1$

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

2: $\quad X_{j+1} \leftarrow \operatorname{argmin}_{\mathcal{A}(x)+b \in \mathcal{K}} f(x) + \frac{\mu_j}{2}\|x - Y_j\|_2^2$

3: $\quad Y_{j+1} \leftarrow X_{j+1}$ or $Y_{j+1} \leftarrow Y_j$

4: $\quad \mu_{j+1} \leftarrow \beta\mu_j$

Note that Listing 10 allows for the updating of both the smoothing parameter μ_j and the proximity center Y_j at each iteration. In many implementations, $Y_j \equiv X_0$ and only μ_j is decreased, but updating Y_j as well will almost always be beneficial. When Y_j is updated in this manner, the algorithm is known as the proximal point method, which has been studied since at least [Roc76]. Indeed, one of the accelerated variants [AT06] that is used in our solvers uses the proximal point framework to analyze gradient-mapping type updates. It turns out we can do much better by applying the same acceleration ideas already mentioned.

Let us suggestively write

$$h(Y) = \min_{x \in C} f(x) + \frac{\mu}{2}\|x - Y\|_2^2, \tag{4.5.11}$$

where $\mu > 0$ is fixed and C is a closed convex set. This is an infimal convolution, and h is known as the Moreau-Yosida regularization of f [HUL93]. Define

$$X_Y = \operatorname*{argmin}_{x \in C} f(x) + \frac{\mu}{2}\|x - Y\|_2^2. \tag{4.5.12}$$

The map $Y \mapsto X_Y$ is a proximity operator [Mor65].

We now state a very useful theorem.

Theorem 4.5.1. *The function h (4.5.11) is continuously differentiable with gradient*

$$\nabla h(Y) = \mu(Y - X_Y). \tag{4.5.13}$$

The gradient is Lipschitz continuous with constant $L = \mu$. Furthermore, minimizing h is equivalent to minimizing $f(x)$ subject to $x \in C$.

The proof is not difficult and can be found in Proposition I.2.2.4 and §XV.4.1 in [HUL93]; see also exercises 2.13 and 2.14 in [BNO03], where h is referred to as the *envelope* function of f. The Lipschitz constant is μ since X_Y is a proximity operator P, and $I - P$ is non-expansive for any proximity operator.

The proximal point method can be analyzed in this framework. Minimizing h using gradient descent, with step size $t = 1/L = 1/\mu$, gives

$$Y_{j+1} = Y_j - t\nabla h(Y_j) = Y_j - \frac{1}{\mu}\mu(Y_j - X_{Y_j}) = X_{Y_j},$$

which is exactly the proximal point algorithm. But since h has a Lipschitz gradient, we can use the accelerated first-order methods to achieve a convergence rate of $\mathcal{O}(j^{-2})$, versus $\mathcal{O}(j^{-1})$ for standard continuation. In Listing 11, we offer such an approach using a fixed step size $t = 1/L$ and the accelerated algorithm from Algorithm 2 in [Tse08]. This accelerated version of the proximal point algorithm has been analyzed in [G̈92] where it was shown to be stable with respect to inexact solves.

Listing 11 Accelerated continuation

Require: Y_0, $\mu_0 > 0$

1: $X_0 \leftarrow Y_0$
2: **for** $j = 0, 1, 2, \ldots$ **do**
3: $\quad X_{j+1} \leftarrow \operatorname{argmin}_{\mathcal{A}(x)+b \in \mathcal{K}} f(x) + \frac{\mu_j}{2}\|x - Y_j\|_2^2$
4: $\quad Y_{j+1} \leftarrow X_{j+1} + \frac{j}{j+3}(X_{j+1} - X_j)$
5: $\quad$ (optional) increase or decrease μ_j

Figure 4.3 provides an example of the potential improvement offered by accelerated continuation. In this case, we have constructed a LASSO model with a 80×200 i.i.d. Gaussian measurement matrix. The horizontal axis gives the number of continuation steps taken, and the vertical axis gives the error $\|X_j - x^\star\|_2/\|x^\star\|_2$ between each continuation iterate and an optimal solution to the unsmoothed model; empirically, the optimal solution appears to be unique. Both simple and accelerated continuation have been employed, using a fixed smoothing parameter $\mu_j \equiv \mu_0$.[8] A clear advantage is demonstrated for accelerated continuation in this case.

[8] If μ is fixed, this is not *continuation* per se, but we still use the term since it refers to an *outer* iteration.

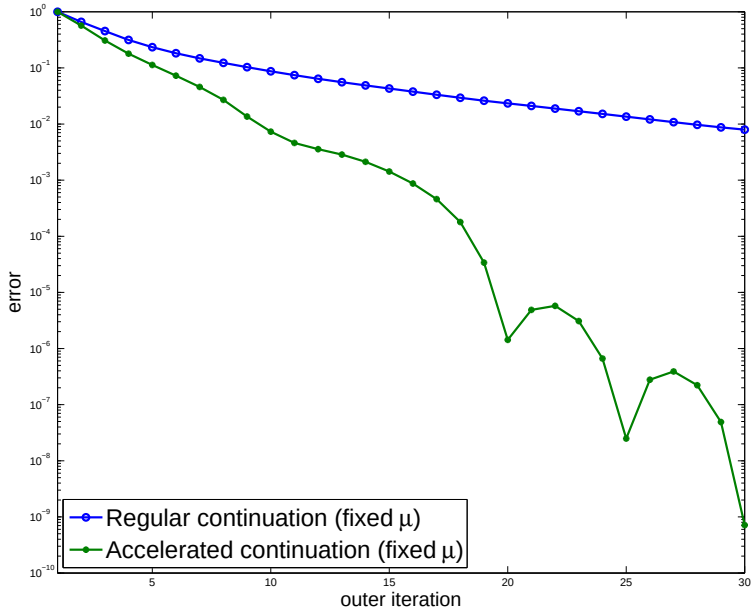


Figure 4.3: A comparison of simple continuation and accelerated continuation applied to a small-scale LASSO model. The horizontal axis gives the number of continuation steps taken; the vertical axis gives the error between the continuation solution and the original, unsmoothed model. The error is normalized to 1 at iteration 1.

For models that exhibit the exact penalty property, updating the proximity center Y_j at each iteration yields an interesting result. Examining the proof in Appendix 4.11, we see that the property depends not on the size of μ but rather on the size of $\mu\|x_j^\star - Y_j\|$, where $x_j^\star$ is projection of Y_j on the optimal set (e.g., if $x^\star$ is unique, then $x_j^\star \equiv x^\star$). Therefore, for *any* positive μ, the exact penalty property will be obtained if Y_j is sufficiently close to the optimal set. This has some obvious and very useful consequences.

The use of accelerated continuation does require some care, however, particularly in the dual conic framework. The issue is that we solve the dual problem, but continuation is with the primal variable. When Y is updated in a complicated fashion, as in Listing 11, or if μ changes, we no longer have a good estimate for a starting dual variable λ_j, so the subproblem will perhaps take many inner iterations. In contrast, if Y is updated in a simple manner and μ is fixed, as in Listing 10, then the old value of the dual variable is a good initial guess for the new iteration, and this warm start results in fewer inner iterations. For this reason, it is sometimes advantageous to use Listing 10.

Figure 4.4 shows an example where accelerated continuation does not offer a clear benefit. The model solved is the Dantzig selector, with a 64×256 partial DCT measurement matrix, and the exact solution (see Appendix 4.12) has 50 nonzeros and a dynamic range of 77 dB which makes the test quite challenging, for the minimum ℓ_1 solution is not the sparsest. The algorithm is tested without continuation, using $\mu = \{\mu_0/10, \mu_0, 10\mu_0\}$ for $\mu_0 = 10^{-2}$, and also with both types of continuation, using $\mu = 50\mu_0$. The horizontal axis gives the total number of *inner* iterations, and the vertical axis gives the error $\|x_k - x^\star\|_2/\|x^\star\|_2$ between the current iterate and the solution to the (unsmoothed)

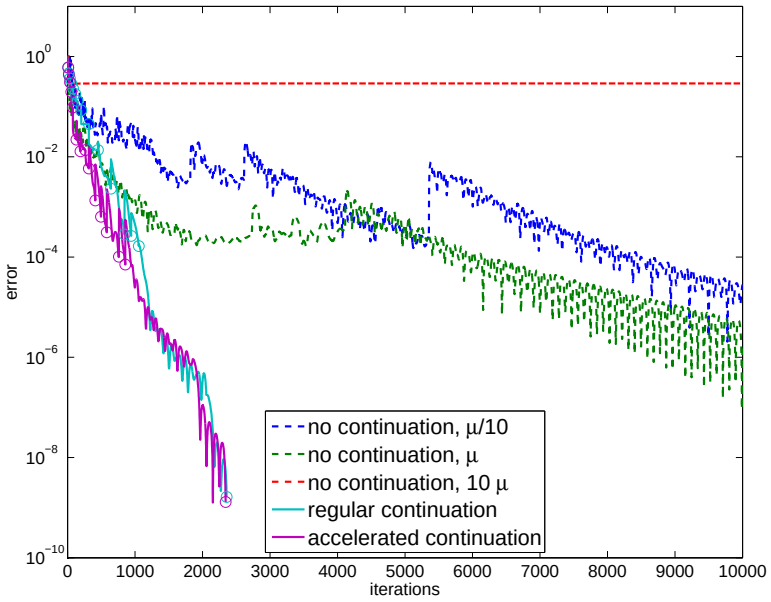


Figure 4.4: Comparing fixed smoothing and continuation strategies for a Dantzig selector model. The horizontal axis gives the number of inner iterations of each first-order method. For the two continuation strategies, the circles depict the completion of each continuation step.

model. When no continuation is employed, the trade-off between solution accuracy (small μ) and the number of iterations to converge (large μ) is evident. Continuation helps drastically, and high levels of accuracy can be obtained quickly. In this example, both forms of continuation perform similarly.

In our experience, accelerated continuation usually matches or outperforms regular continuation, but the matter requires further study. In particular, performance is affected by the stopping criteria of the inner iteration, and we plan to investigate this in future work. The experiments in Figure 4.4 decreased the tolerance by a factor of two every iteration, with the exception of the final iteration which used a stricter tolerance.

4.5.6 Strong convexity

Suppose that our composite function ϕ is strongly convex; that is,

$$\phi(z) \geq \phi(z_0) + \tfrac{1}{2} m_\phi \|z - z_0\|_2^2 \quad \forall z \in \operatorname{dom} \phi$$

for some fixed $m_\phi > 0$. It is well known that a standard gradient method will achieve linear convergence in such a case. Unfortunately, without modification, none of the so-called optimal methods will do so. Thus in the presence of strong convexity, standard gradient methods can actually perform better than their so-called optimal counterparts.

If the strong convexity parameter $m_\phi \leq L_\phi$ is known or can be bounded below, the N83 algorithm can be modified to recover linear convergence [Nes05, Nes07], and will recover its superior

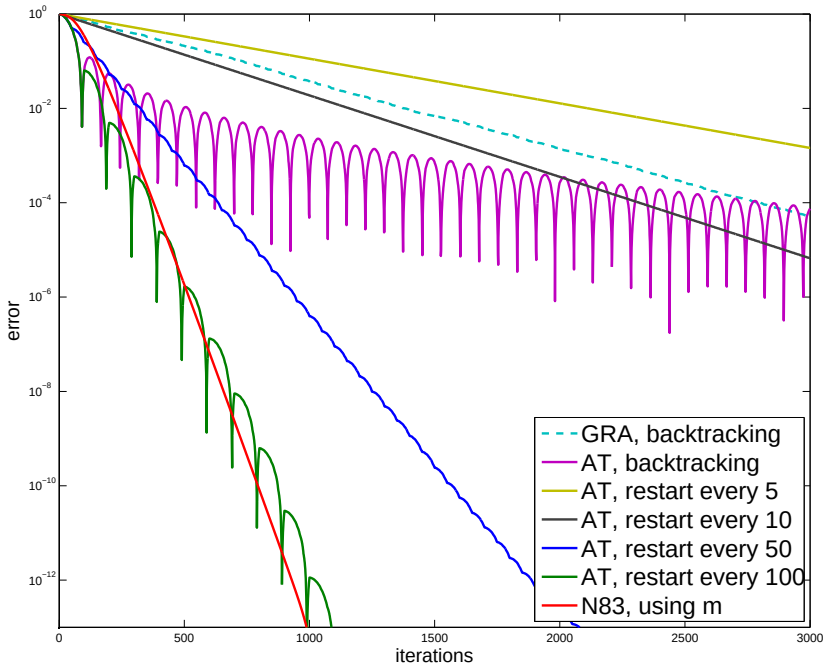


Figure 4.5: First-order methods applied to a strongly convex problem: GRA, AT without restart, AT with restart, and N83 tuned with knowledge of the strong convexity parameter. The y-axis is $\|x_k - x^\star\|_2/\|x^\star\|_2$.

performance compared to standard gradient methods. The challenge, of course, is that this parameter rarely *is* known. In [Nes07], Nesterov provides two approaches for dealing with the case of unknown m_ϕ, one of which is readily adapted to all of the first-order variants here. That approach is a so-called *restart* method: the algorithm is restarted after a certain number of iterations, using the current iterate as the starting point for the restart; or, equivalently, the acceleration parameter θ_k is reset to $\theta_0 = 1$. In theory, the optimal number of iterations between restarts depends on m_ϕ/L_ϕ, but linear convergence can be recovered even for sub-optimal choices of this iteration count [GLW09].

To illustrate the impact of strong convexity on performance, we constructed a strongly quadratic unconstrained function with $m_\phi = 0.07$ and $L_\phi = 59.1$, and minimized it using the gradient method (GRA), the AT first-order method with and without restart, and the N83 method tuned to the exact values of (m_ϕ, L_ϕ). Backtracking was employed in all cases; we have verified experimentally that doing so consistently improves performance and does not compromise the exploitation of strong convexity. As can be seen in Figure 4.5, while AT without restart initially performs much better than GRA, the linear convergence obtained by GRA will eventually overtake it. The N83 method is clearly superior to either of these, achieving linear convergence with a much steeper slope. When restart is employed, AT recovers linear convergence, and achieves near parity with N83 when restarted every 100 iterations (the optimal value predicted by [GLW09] is every 112 iterations).

The benefits of exploiting strong convexity seem clearly established. In fact, while proofs of linear convergence assume global strong convexity, our experiments in §4.7 show that applying

these methods to models with local strong convexity can improve performance as well, sometimes significantly. Unfortunately, the restart method requires manual tuning on a case-by-case basis to achieve the best performance. This is certainly acceptable in many applications, but further research is needed to develop approaches that are more automatic.

4.6 Dual-function formulation

Instead of a *conic* viewpoint, where the main requirement was the ability to project onto a cone (often an epigraph cone), we may also view the problem in a dual *function* framework. The resulting algorithm is identical; the new derivation is presented because it may be more natural for some problems. A similar framework is presented in [CDV10], which uses proximal splitting methods instead of gradient-based methods.

4.6.1 Background

In this section, convex functions are allowed to take values in $(-\infty, +\infty]$. The domain of a convex function f is the set of points $\operatorname{dom} f = \{x : f(x) < +\infty\}$ and the epigraph is $\operatorname{epi} f = \{(x, \eta) : f(x) \leq \eta\}$. A convex function is *proper* if it is not the constant function $f = +\infty$, and it is called *lower semi-continuous* (lsc) if $\operatorname{epi} f$ is closed. Let $\Gamma_0(\mathbb{R}^n)$ be the set of proper, lsc, convex functions f on $\mathbb{R}^n$. As a shorthand, we write "f is convex" to mean $f \in \Gamma_0(\mathbb{R}^n)$, since we will not discuss convex functions that are not in Γ_0. If the reader is not familiar with subgradients and standard convex analysis, then [Roc70] is recommended as the classic reference, or [BC11] for a modern perspective.

The *convex dual* function (also known as the Fenchel or Fenchel-Legendre dual) of a convex function f is given by:

$$f^*(z) = \sup_x \langle z, x \rangle - f(x).$$

The dual f^* is also convex, and since f is lsc, $f^{**} = f$.

We re-introduce the Moreau proximity operator [Mor62, Mor65] (first discussed in §4.5.5) using slightly different notation. This Moreau proximity operator is also known as a proximity function, or a "proximation" in [Roc70], and the Moreau envelope is also known as the Moreau-Yosida regularization.

Definition 4.6.1 (Moreau proximity operator)**.** *The Moreau envelope of a convex function f is*

$$e_f(z) = \min_x \; f(x) + \frac{1}{2}\|x - z\|_2^2 \tag{4.6.1}$$

and the Moreau proximity operator

$$\text{prox}_f(z) = \underset{x}{\text{argmin}} \; f(x) + \frac{1}{2}\|x - z\|_2^2. \tag{4.6.2}$$

Define

$$e_{f,\mu}(z) = \underset{x}{\text{argmin}} \; f(x) + \frac{\mu}{2}\|x - z\|_2^2$$

and define $\text{prox}_{f,\mu}$ *analogously.*

Theorem 4.5.1 shows $e_{f,\mu}(z)$ is continuously differentiable with gradient

$$\nabla_z e_{f,\mu}(z) = \mu(z - \text{prox}_{f,\mu}(z)). \tag{4.6.3}$$

and Lipschitz constant μ.

The classical orthogonal projection theorem states that for any subspace $V \subset \mathbb{R}^n$, then $x \in \mathbb{R}^n$ can be decomposed $x = x_\parallel + x_\perp$ where $x_\parallel \in V$ and $\langle x_\parallel, x_\perp \rangle = 0$. Using the Moreau proximity operator, it is possible to prove a more general projection theorem which includes the classical projection theorem as a subcase.

Lemma 4.6.2 (Moreau's decomposition [CW05])**.** *Let f be a convex function, $x \in \mathbb{R}^n$, and $\mu > 0$. Then*

$$x = x_\parallel + x_\perp \quad \textit{where} \tag{4.6.4}$$

$$x_\parallel = \mu \, \text{prox}_{f,\mu^{-1}}(x), \qquad x_\perp = \text{prox}_{f^*,\mu}(x/\mu) \tag{4.6.5}$$

$$\textit{and} \quad f(x_\parallel) + f^*(x_\perp/\mu) = \mu^{-1}\langle x_\parallel, x_\perp \rangle. \tag{4.6.6}$$

If particular, if $\mu = 1$, then

$$x_\parallel = \text{prox}_f(x), \quad x_\perp = \text{prox}_{f^*}(x), \quad f(x_\parallel) + f^*(x_\perp) = \langle x_\parallel, x_\perp \rangle.$$

4.6.2 Fenchel dual formulation

Consider the problem

$$\min_x f(x) + \frac{\mu}{2}\|x - x_0\|^2 + \sum_{i=1}^{K} \phi_i(A_i x + b_i) \tag{4.6.7}$$

where f and ϕ_i are convex functions. This allows the case when ϕ_i may be the indicator function ι to a set C:

$$\iota_C(x) = \begin{cases} 0 & x \in C \\ +\infty & x \notin C \end{cases}.$$

The set C must be closed in order for $f = \iota_C$ to be lsc. We apply the dual formulation to (4.6.7). The first step is introducing slack variables $y_i = A_i x + b_i$, so that (4.6.7) is equivalent to

$$\min_{x,y_i} f(x) + \frac{\mu}{2}\|x - x_0\|^2 + \sum_{i=1}^{K} \phi_i(y_i) \quad \text{such that} \quad y_i = A_i x + b_i \ \ \forall i \le K. \tag{4.6.8}$$

Let $z(\lambda) = \mu^{-1} \sum_i A_i^T \lambda_i$, where $\lambda_i \in \mathbb{R}^{m_i}$ if $A_i \in \mathbb{R}^{m_i \times n}$, and λ represents $\lambda_1, \ldots, \lambda_K$. The dual function is

$$\begin{aligned}
g(\lambda) &= \inf_{x,y_i} \left(f(x) + \frac{\mu}{2}\|x - x_0\|^2 + \sum_i \phi_i(y) + \langle \lambda_i, A_i x + b_i - y_i \rangle \right) \\
&= \sum_i \langle \lambda_i, b_i \rangle + \inf_x \left(f(x) + \frac{\mu}{2}\|x - x_0\|^2 + \mu \langle z(\lambda), x \rangle \right) + \sum_i \inf_{y_i} \left(\phi(y_i) - \langle \lambda_i, y_i \rangle \right) \\
&= \sum_i \langle \lambda_i, b_i \rangle - \frac{\mu}{2}\|z(\lambda)\|^2 + \mu \langle z(\lambda), x_0 \rangle + \inf_x \left(f(x) + \frac{\mu}{2}\|x - (x_0 - z(\lambda))\|^2 \right) - \sum_i \phi^*(\lambda_i) \\
&= \underbrace{\sum_i \langle \lambda_i, b_i \rangle - \frac{\mu}{2}\|z(\lambda)\|^2 + \mu \langle z(\lambda), x_0 \rangle + e_{f,\mu}(x_0 - z(\lambda))}_{g_{\text{smooth}}(\lambda)} - \underbrace{\sum_i \phi^*(\lambda_i)}_{h(\lambda)}.
\end{aligned} \tag{4.6.9}$$

The dual function is always concave. Using (4.6.3),

$$\nabla_{\lambda_i} g_{\text{smooth}}(\lambda) = b_i + A_i x(\lambda), \quad x(\lambda) = \text{prox}_{f,\mu}(x_0 - z(\lambda)).$$

Let $\mathcal{A} = [A_1; A_2; \ldots; A_K]$ and $\mathbf{b} = [b_1; b_2; \ldots; b_K]$ (both with $\sum_{i=1}^K m_i$ rows), then the total derivative is

$$\nabla_\lambda g_{\text{smooth}}(\lambda) = \mathbf{b} + A x(\lambda).$$

Since $x(\lambda)$ is computed via a proximity operator, it is a non-expansive function of its inputs $x_0 - z(\lambda)$, where $z(\lambda) = \mu^{-1}\mathcal{A}^T\lambda$, so ∇g_{smooth} is Lipschitz continuous with bound

$$L = \mu^{-1}\|\mathcal{A}\mathcal{A}^T\| \le \mu^{-1} \sum_{i=1}^{K} \|A_i^T A_i\|.$$

If, say, $\|A_1^T A_1\|$ is very large but $\|A_2^T A_2\|$ is small, then the Lipschitz bound is large and the λ_2 variable will converge slowly due to the presence of the variable λ_1 with a very different scale. It is best if all λ_i have the same scale. This can be enforced by introducing the scaling $\alpha_i = \|A_i^T A_i\|^{-1/2}$, and modifying (4.6.8) as follows:

$$\min_{x,y_i} f(x) + \frac{\mu}{2}\|x - x_0\|^2 + \sum_{i=1}^{K} \bar{\phi}_i(y_i) \quad \text{such that} \quad y_i = \alpha_i^{-1}(A_i x + b_i) \ \ \forall i \le K$$

and $\bar{\phi}_i(y) = \phi_i(\alpha_i y)$. Computing the proximity operator of $\bar{\phi}_i$ is no more difficult than computing the proximity operator of ϕ_i. From now on, we assume this scaling has already been performed.

The dual problem is

$$\max_{\lambda_i, i=1,\ldots,K} g_{\text{smooth}}(\lambda) - h(\lambda) \tag{4.6.10}$$

The dual function is an unconstrained problem (other than any implicit constraints in ϕ^*), and it can be solved with the composite formulation previously discussed. The composite formulation requires the gradient of g_{smooth} and the ability to solve the generalized projection with h; the generalized projection can be rewritten in terms of the Moreau proximity function of h. Furthermore, this is separable in λ_i. Thus (4.6.10) is easy to solve when

1. prox_f is efficient to calculate, and
2. $\text{prox}_{\phi_i^*}$ is efficient to calculate for all i.

For many functions, such as norms, the proximity operator is $\mathcal{O}(n)$ to compute. For a list of functions with known proximity operators, see [CW05].

There are three especially salient points:

1. ϕ_i^* is the dual function of $\phi_i(x)$, and *not* the dual function of $\tilde{\phi}_i(x) \equiv \phi_i(A_i x + b)$. The dual function of $\tilde{\phi}_i$ may be much more difficult to compute. This is the same observation made earlier that it is easy to project onto the set $\{x : \|x\|_2 \leq 1\}$ but it can be difficult to project onto the set $\{x : \|Ax\|_2 \leq 1\}$. This observation has been made many times in the history of optimization; see, for example, the discussion of Uzawa's algorithm in [Cia89].

2. Using Moreau's decomposition (4.6.4) and (4.6.5), it's clear that computing prox_f and prox_{f^*} are equally easy or difficult, since one can be computed via the other.

3. It is possible to define $\phi_{K+1} \leftarrow f$, $A_{K+1} \leftarrow I$, $b_{K+1} \leftarrow 0$, and $f \leftarrow 0$. That is, f can be treated specially and kept in the primal, or we can introduce one more dual variable λ_{K+1} and think of f as another ϕ_i. The result algorithms are different, although they will solve the same problem at convergence. The complexity per iteration stays nearly the same since prox_{f^*} has the same cost as prox_f, and the proximity operator of the zero function is trivial to compute. We note that the convergence properties of the two methods need not be the same, but one version may sometimes be used to bound the convergence of the other, as we show in §4.6.3.2.

This dual-function viewpoint is especially useful for problems considered in this chapter, since f and ϕ_i are typically norms or indicator functions of norm balls. Define the dual norm of any norm $\|\cdot\|$ to be $\|\cdot\|_*$ where

$$\|y\|_* \triangleq \sup_{\|x\| \leq 1} \langle y, x \rangle.$$

Then

$$h(y) = s\|y\| \quad \iff \quad h^*(z) = \iota_{\{z:\|z\|_* \leq s\}}. \tag{4.6.11}$$

If projection onto the dual-norm ball is efficient, then the dual smoothing algorithm has cheap per-iteration cost. Many signal processing problems involve only the ℓ_1, ℓ_2, and ℓ_∞ norms, for which projection onto the norm ball is $\mathcal{O}(n)$ or $\mathcal{O}(n \log n)$, and hence any minimization problem with convex combinations of these norms (with possibly affine scalings) is cheap to solve.

4.6.3 Convergence

Via the dual-function framework, it is possible to prove the convergence rate of the algorithm. There are three major steps: first, we show in §4.6.3.1 that the iterates x_k converge when solving a special variant of (4.6.7). Secondly, §4.6.3.2 extends this to cover the generic case, and thirdly, in §4.6.4.1 we incorporate continuation and prove that the value of the outer function converges.

4.6.3.1 Convergence when f is smooth

The problem in (4.6.7) allows a number of functions ϕ_i, but for simplicity let us restrict to the case of $K = 1$, so the problem of interest is

$$\min_x f(x) + \frac{\mu}{2}\|x - x_0\|^2 + \phi_1(Ax + b). \tag{4.6.12}$$

As discussed above, it is possible to treat f as one of the ϕ_i, e.g.,

$$\min_x 0 + \frac{\mu}{2}\|x - x_0\|^2 + \phi_1(y_1) + \phi_2(y_2) \quad \text{such that} \quad y_1 = Ax + b, y_2 = x \tag{4.6.13}$$

where $\phi_2(x) = f(x)$. In this case, g_{smooth} is a quadratic function, so it is both smooth and strongly convex (with parameter μ). The dual function is

$$g(\lambda_1, \lambda_2) = g_{\text{smooth}}(\lambda_1, \lambda_2) - \phi_1^*(\lambda_1) - \phi_2^*(\lambda_2)$$

and $x(\lambda)$ is a linear function of λ_1 and λ_2.

When solving the dual problem via an accelerated method, the dual objective converges at the rate

$$g^\star - g(\lambda) \leq \frac{C\mu^{-1}}{k^2}.$$

While the objective value converges, the dual variable λ need not converge, and there may not be a unique maximizer $\lambda^\star$.

Since we now think of f as ϕ_2, which acts as a "constraint" in a generalized projected-gradient algorithm, the primal objective function is just the smoothing term $\frac{\mu}{2}\|x - x_0\|^2$, which is strongly

convex. Hence it is not unreasonable to expect that the *primal* variable x does converge. Indeed, this is the case. A recent result in [FP11], using a Bregman distance technique, gives

$$\|x(\lambda_k) - x^\star\|^2 \leq \frac{2}{\mu}(g^\star - g(\lambda_k)). \tag{4.6.14}$$

See Theorem 1 and Theorem 2 in [FP11]; it is also mentioned in [CP10c].

A broad class of models, such as solving the ℓ_1-analysis problem and TV-minimization, fall into this category, so this result is useful in itself.

4.6.3.2 Convergence of inner iteration

Consider again (4.6.12). It seems unlikely that convergence is dependent on the "interpretation" of f, so we are inclined to believe that the model

$$\min_x f(x) + \frac{\mu}{2}\|x - x_0\|^2 + \phi_1(y_1) \quad \text{such that} \quad y_1 = Ax + b \tag{4.6.15}$$

should converge at the same rate. This intuition is correct, and we show how to obtain the bound.

Using the special case just discussed in (4.6.13) (refer to this as case "A"), the dual function depends on λ_1 and λ_2. Write this dual function as g_A:

$$g_A(\lambda_1, \lambda_2) = \inf_x \mathcal{L}_A(\lambda_1, \lambda_2, x) = \mathcal{L}_A(\lambda_1, \lambda_2, x_A(\lambda)) \tag{4.6.16}$$

$$\mathcal{L}_A(\lambda_1, \lambda_2, x) = \langle \lambda_1, b_1 \rangle + \left(0 + \frac{\mu}{2}\|x - x_0\|^2 + \langle A^T\lambda_1 + \lambda_2, x\rangle\right) - \phi_1^*(\lambda_1) - f^*(\lambda_2)$$

$$x_A(\lambda) = \operatorname*{argmin}_x \frac{\mu}{2}\|x - x_0\|^2 + \langle A^T\lambda_1 + \lambda_2, x\rangle. \tag{4.6.17}$$

Using the new case (4.6.15) (case "B"), we keep f in primal form, and the dual function, written as g_B to distinguish it from the other case, is only a function of λ_1:

$$g_B(\lambda_1) = \inf_x \mathcal{L}_A(\lambda_1, x) = \mathcal{L}_B(\lambda_1, x_B(\lambda)) \tag{4.6.18}$$

$$\mathcal{L}_B(\lambda_1, x) = \langle \lambda_1, b_1 \rangle + \left(f(x) + \frac{\mu}{2}\|x - x_0\|^2 + \langle A^T\lambda_1, x\rangle\right) - \phi_1^*(\lambda_1)$$

$$x_B(\lambda) = \operatorname*{argmin}_x f(x) + \frac{\mu}{2}\|x - x_0\|^2 + \langle A^T\lambda_1, x\rangle. \tag{4.6.19}$$

Our strategy is as follows. Solve "B" and bound the objective $g^\star - g_B(\lambda_1) \leq C$ (using standard convergence results on Nesterov's method and variants) for some λ_1 (really, this is $\lambda_1^{(k)}$ since it depends on the iteration, but we drop the dependence on k for simplicity). Using λ_1, we generate λ_2 so that $x_A(\lambda_1, \lambda_2) = x_B(\lambda_1)$ and $g_A(\lambda_1, \lambda_2) = g_B(\lambda_1)$. Thus we can apply existing results from the previous section that give us the bound on x_A, and hence x_B, in terms of $g^\star - g_A(\lambda_1, \lambda_2) \leq C$.

We start with a known λ_1 and x_B calculated via method "B". In the proximity calculation

(4.6.19), we know that x_B must satisfy the first-order condition

$$0 \in \partial f(x_B) + \mu(x_B - x_0) + A^T \lambda_1.$$

Define

$$\lambda_2 = -\mu(x_B - x_0) - A^T \lambda_1,$$

so $\lambda_2 \in \partial f(x_B)$. By examining (4.6.17) with this value of λ_2 (and the same value of λ_1), it's clear that $x_A = x_B$, since (4.6.17) is the minimization of an unconstrained differentiable function and has a unique minimizer characterized by

$$0 = \lambda_2 + \mu(x_A - x_0) + A^T \lambda_1$$

which is certainly also satisfied by x_B.

The remaining question is whether $\mathcal{L}_A(\lambda_1, \lambda_2, x_B) = \mathcal{L}_B(\lambda_1, x_B)$. This is verified by an application of Fenchel's equality [Roc70]

$$\begin{aligned} \mathcal{L}_B(\lambda_1, x_B) - \mathcal{L}_A(\lambda_1, \lambda_2, x_B) &= f(x_B) - \langle \lambda_2, x_B \rangle + f^*(\lambda_2) \\ &= 0 \end{aligned}$$

since $\lambda_2 \in \partial f(x_B)$. Hence any sequence generated by method "B" has the same bounds that method "A" enjoys.

4.6.4 Convergence of outer iteration

Let $F(x) = f(x) + \sum_i \phi_i(A_i x + b_i)$, so the primal problem is

$$\min_x F(x).$$

The accelerated continuation scheme solves this by solving a problem of the form

$$\min_Y \underbrace{\min_x F(x) + \frac{\mu}{2}\|x - Y\|^2}_{\psi(Y)} \tag{4.6.20}$$

by using the method in Listing 11. When the outer iteration uses an accelerated method, we call this the accelerated proximal point algorithm; if the outer iteration is gradient descent with fixed step size, this reduces to the proximal point algorithm. Let $\psi_j = \psi(Y_j)$ be the value of the outer iteration at step j. If ψ_j is known exactly, then because accelerated continuation is an optimal method, the value of ψ will converge to the optimal value $\psi(Y^\star)$ at rate $\mathcal{O}(\frac{\mu}{j^2})$.

The calculation of ψ_j is done with an inner iteration, and this will never be exact. The inner iteration is solved in the dual smoothing framework, and so the dual function $g(\lambda_k)$ converges to $g(\lambda_j^\star) = \min F(x) + \frac{\mu}{2}\|x - Y_j\|^2$ at rate $\mathcal{O}(\frac{\mu^{-1}}{k^2})$. There is a trade off in the value of μ, since a small μ will speed up the outer iteration but slow down the inner iteration; it is also possible to vary μ every outer-iteration. Is it possible to make precise this trade off, and thus suggest the best value of μ to use? To what accuracy should the sub-problems be solved? On linear programs, does this accelerated proximal-point algorithm converge in a finite number of iterations, as the regular proximal-point algorithm does [PT74, Ber75]? Using very recent results, we begin to answer these questions in §4.6.3 and §4.6.4.1; this work has not appeared in the preprint version of this chapter. Below, we review existing results, and set the stage for the final convergence proof.

In 1992, Güler [G̈92] analyzed an accelerated variant of the proximal point algorithm and provided the first robust convergence analysis. Let $X_j^\star = \operatorname{argmin} F(x) + \mu/2\|x - Y_j\|^2$. Güler proved that if every inner iteration is solved to an iterate x_k satisfying an inexact criteria, it can still be possible to recovery accelerated convergence. In the case F is a Lipschitz function and finite, then his inexact criteria simplifies to

$$\|x_k - X_j^\star\|_2 \leq \frac{C}{j^{1.5}} \tag{4.6.21}$$

and still guarantees that the outer iteration converges in $\mathcal{O}(\mu/j^2)$. His variant of the accelerated first-order method is not identical to the accelerated continuation scheme we present, but it is trivial to adapt the accelerated continuation to use his version.

A recent paper by Combettes et al. [CDV10] considers a similar problem and proves convergence of a similar scheme but using splitting methods instead of gradient-based methods, and does not obtain any convergence rate bounds. Using the special form of a smoothed problem like ours, [CDV10] shows that a bound on either $\|\lambda_k - \lambda_j^\star\|$, or $\|\nabla g_{\text{smooth}}(\lambda_k) - \nabla g_{\text{smooth}}(\lambda_j u^\star)\|$ is sufficient to bound $\|x_k - X_j^\star\|$. The accelerated composite methods used to solve the dual will guarantee that the generalized gradient mapping converges, but this does not imply that ∇g_{smooth} converges. For unconstrained dual problems, such as noiseless basis pursuit and noiseless matrix completion, the generalized gradient mapping is just the gradient, and hence we can bound the rate at which $x_k \to X_j^\star$ for these specific problems. This result covers some previously known special cases. For example, [COS09] proves convergence of linear Bregman and provides a rate of convergence; linear Bregman is the same as our dual smoothing approach applied to noiseless basis pursuit, with $x_0 = 0$ and using fixed step-size gradient descent. The work of [MZ09] considers a situation similar to [CDV10] but does not prove convergence of the primal variables.

Another known case is when the dual problem is solved via proximal gradient descent. Beck and Teboulle [BT10] prove the strong convergence of variables in proximal gradient descent, though without a known rate. Using the Combettes result, this gives a bound on the rate of the convergence

of the primal variable. Unfortunately, the strong convergence does not apply when using accelerated proximal gradient descent methods.

An alternative approach is to avoid dealing with the primal variable and use either the function value or gradient of the smoothed problem to provide a bound on the sub-optimality of the outer step.

4.6.4.1 Overall analysis

It is possible to prove the convergence of the inner iteration; see §4.6.3, which is based on a theorem of [FP11]. We assume that for a fixed j, the inner iteration converges

$$\|x_k - X_j^\star\|_2^2 \leq \frac{c_j}{\mu^2 k^2}.$$

To get this, combine (4.6.14) with the $\mathcal{O}(\frac{\mu^{-1}}{k^2})$ convergence rate of the dual function. We also need that the constants can be uniformly bounded

$$c_j \leq c \quad \forall j.$$

The only iteration-dependent aspect of the constant c_j is $\|x_0 - X_j^\star\|$ (e.g., for FISTA; see [BT10]), where x_0 is the initialization of the jth step, and is naturally chosen to be the last iterate from the $j-1$ step. It is reasonable to expect that this bound does not grow with j, since otherwise $(X_j^\star)$ is not a Cauchy sequence, but we leave the proof for future work and for now consider it an assumption.

A full analysis for general F is not yet available, but below we present a basic analysis that holds if F is Lipschitz continuous since then we use the condition (4.6.21) to imply Güler's theorem. We need the error at every outer iteration j to satisfy

$$\|x_k - X_j^\star\|_2 \leq \frac{c}{j^{1.5}}$$

for some constant c. This can be ensured by taking

$$k_j = cj^{1.5}/\mu$$

steps (for a new constant c—in this section, we allow the value of c to change, but it is always independent of j and μ).

Via the accelerated continuation scheme, convergence of the outer objective function ψ is at rate $\frac{C\mu}{j^2}$, so for any level of accuracy ϵ, it takes $J = C\sqrt{\mu/\epsilon}$ outer iterations to reach an ϵ-solution—that

is, a solution such that $\psi(Y_j) - \psi(Y^\star) \leq \epsilon$. Hence the total number of inner iterations is

$$\sum_{j=1}^{J} k_j = \frac{c}{\mu} \sum_{j=1}^{J} j^{1.5} \leq \frac{c}{\mu} \int_0^{J+1} j^{1.5} dj = \frac{c'}{\mu}(J+1)^{2.5} \lesssim c'' \frac{\mu^{1/4}}{\epsilon^{5/4}}. \tag{4.6.22}$$

This is only slightly worse than the $\mathcal{O}(1/\epsilon)$ bound used in [Nes07] which relied on a very special form of the primal function. The dependence on μ is weak, but it suggests that smaller values of μ are preferred. However, these bounds are not necessarily tight, so taking a very small μ may improve the bounds but will not necessarily improve the actual performance of the algorithm.

4.7 Numerical experiments

The templates we have introduced offer a flexible framework for solving many interesting but previously intractable problems; for example, to our knowledge, there are no first-order algorithms that can deal with non-trivially constrained problems with complicated objectives like $f(x) = \|Wx\|_1 + \|x\|_{TV}$ for an over-complete dictionary W. This section shows the templates in use to solve such real-world problems. It also describes some of the details behind the numerical experiments in previous sections.

4.7.1 Dantzig selector: comparing first-order variants

Other than Tseng's paper [Tse08], there has been little focus on comparing the various accelerated methods. Tseng's paper itself presents few simulations that differentiate the algorithms. Since our software uses interchangeable solvers with otherwise identical setups, it is easy to compare the algorithms head-to-head applied to the same model.

For this comparison, we constructed a smoothed Dantzig selector model similar to the one employed in §4.3.4 above. The model used a partial DCT measurement matrix of size 512×2048, a signal with 128 nonzero values, and an additive noise level of 30 dB SNR. The smoothing parameter was chosen to be $\mu = 0.25$, and we then employed the techniques of Appendix 4.12 to perturb the model and obtain a known exact solution. This reference solution had 341 nonzeros, a minimum magnitude of 0.002 and a maximum amplitude 8.9. The smoothed model was then solved using the 6 first-order variants discussed here, using both a fixed step size of $t = 1/L = \mu/\|\mathcal{A}\|^2$ and our proposed backtracking strategy, as well as a variety of restart intervals.

The results of our tests are summarized by two plots in Figure 4.6. The cost of the linear operator dominates, so the horizontal axes give the number of calls to either $\mathcal{A}$ or $\mathcal{A}^*$ taken by the algorithm. The vertical axes give the relative error $\|x_k - x_\mu^\star\|/\|x_\mu^\star\|$. Because this is a sparse recovery problem, we are also interested in determining when the algorithms find the correct support; that is, when

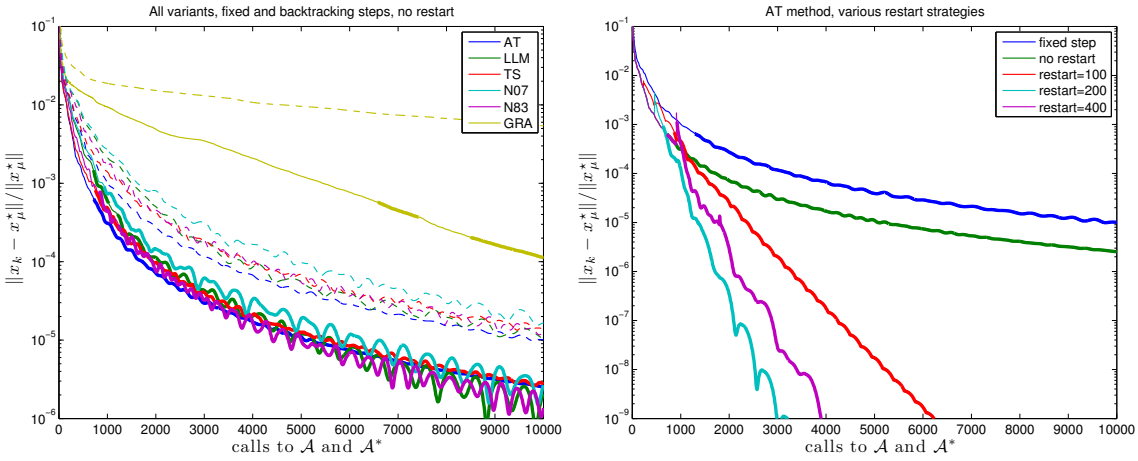


Figure 4.6: Comparing first-order methods applied to a smoothed Dantzig selector model. Left: comparing all variants using a fixed step size (dashed lines) and backtracking line search (solid lines). Right: comparing various restart strategies using the AT method

they correctly identify the locations of the 341 nonzero entries. Therefore, the lines in each plot are thicker where the computed support is correct, and thinner when it is not.

The left-hand plot compares all variants using both fixed step sizes and backtracking line search, but with no restart. Not surprisingly, the standard gradient method performs significantly worse than all of the optimal first-order methods. In the fixed step case, AT performs the best by a small margin; but the result is moot, as backtracking shows a significant performance advantage. For example, using the AT variant with a fixed step size requires more than 3000 calls to $\mathcal{A}$ or $\mathcal{A}^*$ to reach an error of 10^{-4}; with backtracking, it takes fewer than 2000. With backtracking, the algorithms exhibit very similar performance, with AT and TS exhibiting far less oscillation than the others. All of the methods except for GRA correctly identify the support (a difficult task due to the high dynamic range) within 1000 linear operations.

The right-hand plot shows the performance of AT if we employ the restart method described in §4.5.6 for several choices of the restart interval. We observe significant improvements in performance, revealing evidence of local strong convexity. A restart interval of 200 iterations yields the best results; in that case, a relative error of 10^{-4} is obtained after approximately 1000 linear operations, and the correct support after only a few hundred operations. The other variants (except GRA, which is unaffected by restart) show similar performance improvements when restart is applied, although the two-projection methods (N07 and LLM) take about 50% longer than the one-projection methods.

Of course, care should be taken when applying these results to other contexts. For instance, the cost of the projections here is negligible; when they are more costly (see, for instance, §4.7.4), two-projection methods (N07 and LLM) will be expected to fare worse. But even among Dantzig selector models, we found significant variations in performance, depending upon sparsity, noise level, and smoothing. For some models, the two-projection methods perform well; and in others, such as

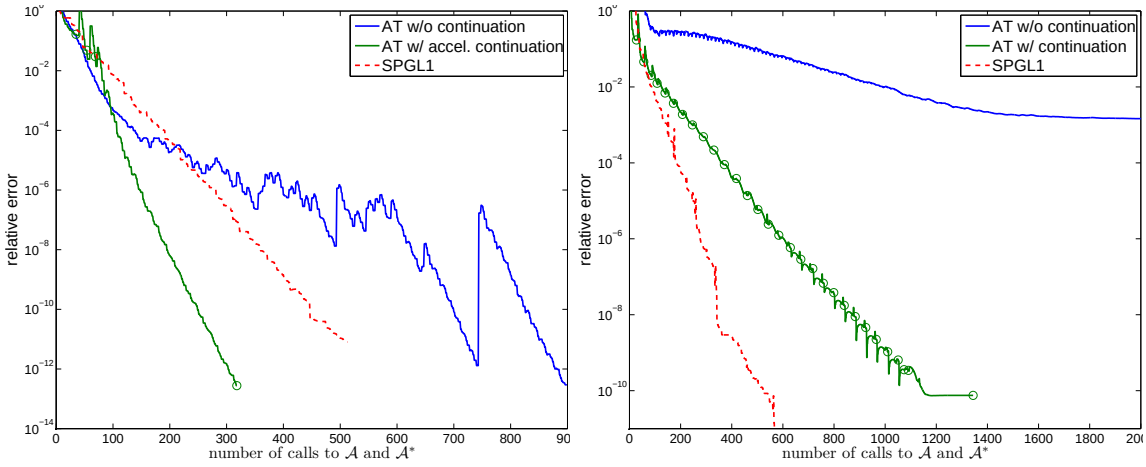


Figure 4.7: Comparisons of the dual solver with SPGL1. The plot on the left involves a noiseless basis pursuit model, while the plot on the right represents a noisy image model.

when we have local strong convexity, gradient descent performs well (when compared to the other algorithms without restart). Overall, it seems there is no *best* algorithm, but we choose the AT algorithm as our default, since in our experience it is consistently one of the best and only requires one projection per iteration.

4.7.2 LASSO: comparison with SPGL1

As mentioned in the introduction, there are numerous algorithms for solving the LASSO. Yet the algorithm produced by our dual conic approach is novel; and despite its apparent simplicity, it is competitive with the state of the art. To show this, we compared the AT first-order variant with SPGL1 [vdBF08], chosen because recent and extensive tests in [BBC11] suggest that it is one of the best available methods.

The nature of the SPGL1 algorithm, which solves a sequence of related problems in a root-finding-scheme, is such that it is fastest when the noise parameter ϵ is large, and slowest when $\epsilon = 0$. To compare performance in both regimes, we constructed two tests. The first is an "academic" test with an s-sparse signal and no noise; however, we choose s large enough so that the LASSO solution does not coincide with the sparse solution, since empirically this is more challenging for solvers. Specifically, the measurement matrix A is a $2^{13} \times 2^{14}$ partial DCT, while the optimal value $x^\star$ was constructed to have $s = 2^{12}$ nonzeros. The second test uses Haar wavelet coefficients from the "cameraman" test image (Figure 4.8 (a)) which decay roughly according to a power law, and adds noise with a signal-to-noise ratio of 30 dB. The measurement matrix is also a partial DCT, this time of size $0.3 \cdot 2^{16} \times 2^{16}$.

Figure 4.7 shows the results from both tests, each plot depicting relative error $\|x_k - x^\star\|/\|x^\star\|$ versus the number of linear operations. We see that both methods achieve several digits of accuracy

in just a few hundred applications of A and its adjoint. SPGL1 outperforms a regular AT solver in the left-hand "academic" test; however, AT with accelerated continuation solves the problem significantly faster than SPGL1. The noiseless case exploits our method's strength since the exact penalty property holds.

For the wavelet test, SPGL1 outperforms our method, even when we use continuation. Although AT with continuation achieves high levels of accuracy in fewer than 1000 operations, other tests confirmed that SPGL1 is often a little better than our method, especially for large ϵ. But the dual conic approach is competitive in many cases, and can be applied to a wider class of problems[9].

4.7.3 Wavelet analysis with total-variation

The benefit of our approach is highlighted by the fact that we can solve complicated composite objective functions. Using the solver templates, it is easy to solve the ℓ_1-analysis and TV problem from §4.4.6. We consider here a denoising problem with full observations; i.e., $A = I$. Figure 4.8 (a) shows the original image x_0, to which noise is added to give an image $y = x_0 + z$ with a signal-to-noise ratio of 20 dB (see subplot (b)). In the figure, error is measured in peak-signal-to-noise ratio (PSNR), which for an $n_1 \times n_2$ image x with pixel values between in $[0, 1]$ is defined as

$$\text{PSNR}(x) = 20 \log_{10} \left(\frac{\sqrt{n_1 n_2}}{\|x - x_0\|_F} \right)$$

where x_0 is the noiseless image.

To denoise, we work with a $9/7$ bi-orthogonal wavelet transform W (similar to that in JPEG-2000) with periodic boundary conditions (the periodicity is not ideal to achieve the lowest distortion). A simple denoising approach is to hard-threshold the wavelet coefficients Wx and then invert with W^{-1}. Figure 4.8 (c) shows the result, where the hard-threshold parameter was determined experimentally to give the best PSNR. We refer to this as "oracle thresholding" since we used the knowledge of x_0 to determine the threshold parameter. As is common with wavelet methods [SNM03], edges in the figure induce artifacts.

Figures 4.8 (d), (e), and (f) are produced using the solvers in this chapter, solving

$$\begin{array}{ll} \text{minimize} & \alpha\|Wx\|_1 + \beta\|x\|_{\text{TV}} + \frac{\mu}{2}\|x - y\|_F^2 \\ \text{subject to} & \|Ax - y\|_2 \leq \epsilon. \end{array} \tag{4.7.1}$$

Figure 4.8 (d) employs wavelet analysis only ($\alpha = 1, \beta = 0$), (e) just TV ($\alpha = 0, \beta = 1$), and (f) both ($\alpha = 1, \beta = 5$). For the best performance, the matrix A was re-scaled so that all the dual variables are of the same order; see [BCG10b] for further discussion of scaling.

[9] SPGL1 is also flexible in the choice of norm, but notably, it cannot solve the analysis problem due to difficulties in the primal projection.

(a) Original

(b) Noisy version (25.6 dB PSNR)

(c) Wavelet thresholded (28.3 dB PSNR)

(d) Wavelet ℓ_1-analysis (29.0 dB PSNR)

(e) TV minimization (30.9 dB PSNR)

(f) Wavelet analysis + TV minimization (31.0 dB PSNR)

Figure 4.8: Denoising an $n = 256^2$ image

The Frobenius term in (4.7.1) is of course for smoothing purposes, and it is possible to minimize its effect by choosing μ small or using the continuation techniques discussed. But for denoising, its presence makes little difference; in fact, it may give more visually pleasing results to use a relatively large μ. So to determine μ, we started with an estimate like

$$\mu = \max(\alpha\|Wy\|_1, \beta\|y\|_{TV})/c$$

with $c \simeq 500$ and then adjusted to give reasonable results. We ultimately employed $\mu = 1$ for (d), $\mu = 50$ for (e), and $\mu = 160$ for (f).

The wavelet analysis run took 26 iterations, and was complete in about 5 seconds. As shown in image (d), it produced boundary effects that are noticeable to the eye, and similar to those produced by thresholded image (c). The TV model (e) and TV with wavelets model (f) took 37 iterations (3 seconds) and 30 iterations (8 seconds), respectively. Both produced better reconstructions, both by PSNR and by visual inspection. The additional wavelet analysis term in plot (f) offers only minimal improvement over TV alone, but this may be due to our simple choice of wavelet transform. For example, undecimated wavelets are common in denoising and may give better results, but our point here is simplicity and to point out the flexibility of the framework.

4.7.4 Matrix completion: expensive projections

We consider the nuclear-norm minimization problem (4.4.2) of a matrix $X \in \mathbb{R}^{n_1 \times n_2}$ in the conic dual smoothing approach. For matrix completion, the linear operator $\mathcal{A}$ is the subsampling operator revealing entries in some subset $E \subset [n_1] \times [n_2]$. With equality constraints ($\epsilon = 0$) and $X_0 = 0$, gradient ascent on the dual is equivalent to the SVT algorithm of [CCS10], a reference which also considered non-equality constraints, e.g., of the form (4.4.2).

In addition to our own interest in this problem, one of the reasons we chose it for this article is that it differs from the others in one key respect: its computational cost is dominated by one of the projections, not by the linear operators. After all, the linear operator in this case is no more than a set of memory accesses, while the primal projection requires the computation of at least the largest singular values of a large matrix. As a result, the considerations we bring to the design of an efficient solver are unique in comparison to the other examples presented.

There are a number of strategies we can employ to reduce the cost of this computation. The key is to exploit the fact that a nuclear-norm matrix completion model is primarily of interest when its optimal value $X^\star$ is expected to have low rank [CR09]. Recall from §4.4.3 that the update of X_k takes the form

$$X_k = \text{SoftThresholdSingVal}\big(X_0 - \mu^{-1}\mathcal{A}^*(\lambda), \mu^{-1}\big).$$

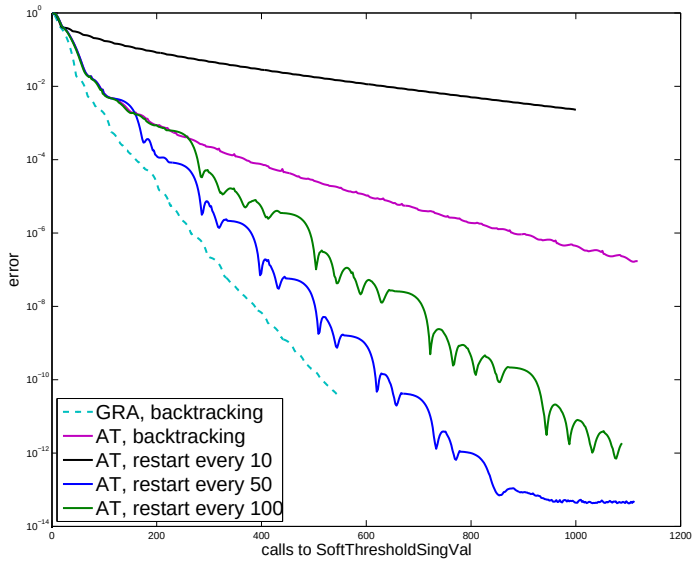


Figure 4.9: Noiseless matrix completion using various first-order methods

Our numerical experiments show that if μ is sufficiently small, the ranks $r_k = \text{rank}(X_k)$ will remain within the neighborhood of $r^\star = \text{rank}(X^\star)$. In fact, with μ sufficiently small and $X_0 = 0$, the rank grows monotonically.

There are a variety of ways we can exploit the low-rank structure of the iterates X_k. By storing $X_k = U_k \Sigma_k V_k$ in factored form, we can reduce the storage costs from $\mathcal{O}(n_1 n_2)$ to $\mathcal{O}(r_k(n_1 + n_2))$. The quantity $\mathcal{A}(X_k)$, used in the computation of the gradient of the dual function, can be computed efficiently from this factored form as well. Finally, using a Lanczos method such as PROPACK [Lar04], the cost of computing the necessary singular values will be roughly proportional to r_k. By combining these techniques, the overall result is that the cost of singular value thresholding is roughly linear in the rank of its result. These techniques are discussed in more detail in [BCG10b].

Smaller values of μ, therefore, reduce the ranks of the iterates X_k, thereby reducing the computational cost of each iteration. This leads to a unique trade off, however: as we already know, smaller values of μ increase the number of iterations required for convergence. In practice, we have found that it is indeed best to choose a small value of μ, and not to employ continuation.

For our numerical example, we constructed a rank-10 matrix of size $n_1 = n_2 = 1000$, and randomly selected 10% of the entries for measurement—about 5 times the number of degrees of freedom in the original matrix. We solved this problem using three variations of GRA and five variations of AT, varying the step size choices and the restart parameter. The smoothing parameter was chosen to be $\mu = 10^{-4}$, for which all of the methods yield a monotonic increase in the rank of the primal variable. The pure gradient method took about 1.8 minutes to reach 10^{-4} relative error, while the AT method without restart took twice the time; the error is in the Frobenius norm, comparing against the true low-rank matrix, which is the solution to the unperturbed problem.

The results of our experiments are summarized in Figure 4.9. The horizontal axis now gives the number of SoftThresholdSingVal operations, a more accurate measure of the cost in this case; the cost of a SoftThresholdSingVal call is not fixed, but we observe that the rank of the iterates quickly reaches 10 and most SoftThresholdSingVal calls have roughly the same cost. The salient feature of this figure is that while AT initially outperforms GRA, the linear convergence exhibited by GRA allows it to overtake AT at moderate levels of precision. Employing a restart method with AT improves its performance significantly; and for a restart intervals of 50 iterations, its performance approaches that of GRA, but it does not overtake it.

What accounts for this performance reversal? First, we observe that AT requires two projections per iteration while GRA requires only one. This difference is inconsequential in our other examples, but not for this one; and it is due to our use of the backtracking line search. Switching to a fixed step size eliminated the extra projection, but the overall performance suffered significantly. We hope to identify a new line search approach that avoids the added cost revealed here.

Second, the similarities of Figures 4.5 and 4.9 suggest the presence of strong convexity. Using an estimate of the decay rate for gradient descent with step size $t = 1/L_f$, and comparing with known decay rate estimates [Nes04] gives an estimate of $m_f = 0.0024$. For this value of m_f (and with $L_f = 1$), the optimal restart number K_{opt} from [GLW09] is about 80, which is consistent with the plot. It is easy to verify, however, that the smoothed dual function is *not* strongly convex.

The results suggest, then, that *local* strong convexity is present. The authors in [GLW09] argue that for compressed sensing problems whose measurement matrices satisfy the restricted isometry property, the primal objective is locally strongly convex when the primal variable is sufficiently sparse. The same reasoning may apply to the matrix completion problem; the operator $\mathcal{A}^*\mathcal{A}$ is nearly isometric (up to a scaling) when restricted to the set of low-rank matrices. The effect may be amplified by the fact that we have taken pains to ensure that the iterates remain low-rank. Overall, this effect is not yet well understood, but will also be explored in later work.

We note that if the underlying matrix is not low-rank, then the local strong convexity effect is not present, and gradient descent does not outperform the AT method. In this case, our experiments also suggest that the restart method has little effect. Likewise, when inequality constraints are added we observe that the unusually good performance of gradient descent vanishes. For both the non-low-rank and noisy cases, then, it may be beneficial to employ an optimal first-order method and to use continuation.

Figure 4.10 demonstrates these claims on a noisy matrix completion problem. We constructed a 50×45 matrix of rank 20, sampled 67% of the entries, and added white noise to yield a 30 dB SNR. By using this small problem size, we are able to use CVX [GB10] to compute a reference solution; this took 5.4 minutes. The figure depicts three different approaches to solving the problem: GRA and AT each with $\mu = 5 \cdot 10^{-4}$ and no continuation; and AT with $\mu = 10^{-2}$ and accelerated

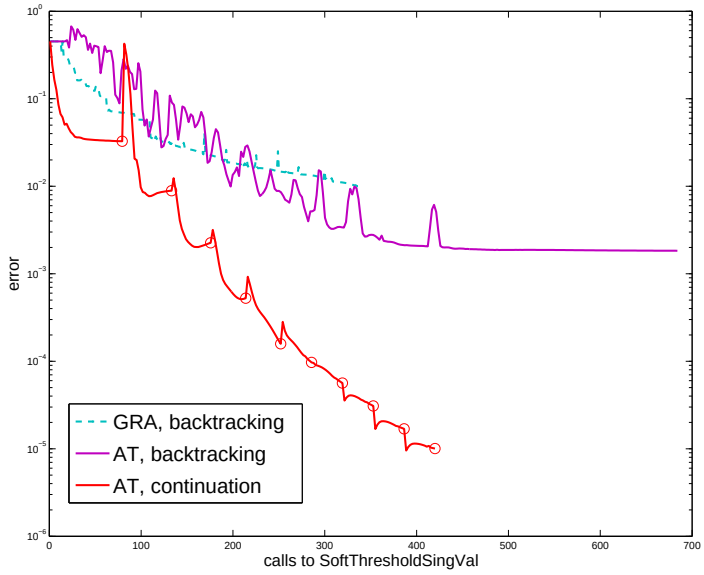


Figure 4.10: Noisy matrix completion on a non-low-rank matrix, using various first-order methods

continuation. The restart approach no longer has a beneficial effect, so it is not shown. Each solver was run for 500 iterations, taking between 2.5 and 4 seconds, and the plot depicts relative error versus the number of singular value decompositions. As we can see, the advantage of GRA has been lost, and continuation provides significant improvement: to achieve a relative error of 10^{-2}, the continuation method required only 100 partial SVDs.

4.7.5 ℓ_1-analysis

We present a brief example of solving the ℓ_1-analysis problem (4.4.5), which is used in sparse recovery when the primal variable x is not sparse itself but rather sparse or compressible in some other domain. The domain may be well-known, such as frequency space (W is a DFT) or a wavelet basis (W is a wavelet transform), and for these cases W is invertible and even orthogonal and the problem may be solved using a method like SPGL1. However, these bases are often chosen out of convenience and not because they best represent the signal. More appropriate choices may be *overcomplete* dictionaries $W \in \mathbb{R}^{p \times n}$ with $p \gg n$, such as the undecimated wavelet transform, or the multilevel Gabor dictionary. The experiment below uses the Gabor dictionary with $p = 28n$.

To solve the LASSO problem using a dictionary W, the two common approaches are *analysis*:

$$\begin{array}{ll} \text{minimize} & \|Wx\|_1 \\ \text{subject to} & \|y - Ax\|_2 \leq \epsilon \end{array} \tag{4.7.2}$$

and *synthesis*:

$$\begin{array}{ll} \text{minimize} & \|\alpha\|_1 \\ \text{subject to} & \|y - AW^*\alpha\|_2 \leq \epsilon, \end{array} \tag{4.7.3}$$

with decision variable α. Similarly, the Dantzig selector approach would have the same objectives but constraints of the form $\|A^*(y - Ax)\|_\infty \leq \delta$ (analysis) and $\|A^*(y - AW^*\alpha)\|_\infty \leq \delta$ (synthesis). When W is not orthogonal, the two approaches are generally different in non-trivial ways, and furthermore, the solutions to *synthesis* may be overly sensitive to the data [EMR07].

The differences between analysis and synthesis are not very well understood at the moment for two reasons. The first is that the analysis problem has not been studied theoretically. An exception is the very recent paper [CENR11] which provides the first results for ℓ_1-analysis. The second is that there are no existing efficient first-order algorithms to solve the analysis problem, with the exception of the recent NESTA [BBC11] and C-SALSA [ABDF11] algorithms, which both work on the LASSO version, and only when $AA^* = I$.

With the dual conic approach, it is now possible to solve the smoothed analysis problem, and by using continuation techniques, the effect of the smoothing is negligible. To illustrate, we constructed a realistic test of the recovery of two radio-frequency radar pulses that overlap in time, as depicted in Figure 4.11. The first pulse is large, while the second pulse has 60 dB smaller amplitude. Both have carrier frequencies and phases that are chosen uniformly at random, and noise is added so that the small pulse has a signal-to-noise ratio of 0.1 dB.

The signal is recovered at Nyquist rate resolution for 2.5 GHz bandwidth, and the time period is a little over 1600 ns, so that n = 8,192. The sensing matrix A is modeled as a block-diagonal matrix with ± 1 entries on the blocks, representing a system that randomly mixes and integrates the input signal, taking 8 measurements every 100 ns, which is 12.5× below the Nyquist rate. Thus A is a 648 × 8,192 matrix and W is a 228,864 × 8,192 matrix. We applied both the Dantzig selector and LASSO models to this problem.

To solve the problems, we employed the AT variant with accelerated continuation. At each iteration, the stopping tolerance is decreased by a factor of 1.5, and the continuation loop is ended when it no longer produces significantly different answers, which is usually between 2 and 8 iterations. The value of μ is set to

$$\mu = 0.1 \frac{\|Wx_{LS}\|_2}{\frac{1}{2}\|x_{LS}\|^2}, \tag{4.7.4}$$

where x_{LS} is the least-squares solution to $Ax = y$, which is easy to calculate since $m \ll n$ (and independent of p).

To enhance the results, we employed an outer *reweighting* loop [CWB08], in which we replace the $\|Wx\|_1$ term with $\|RWx\|_1$ for some diagonal weight matrix R. Each reweighting involves a full solve of either the Dantzig selector or the LASSO. This loop is run until convergence, which is

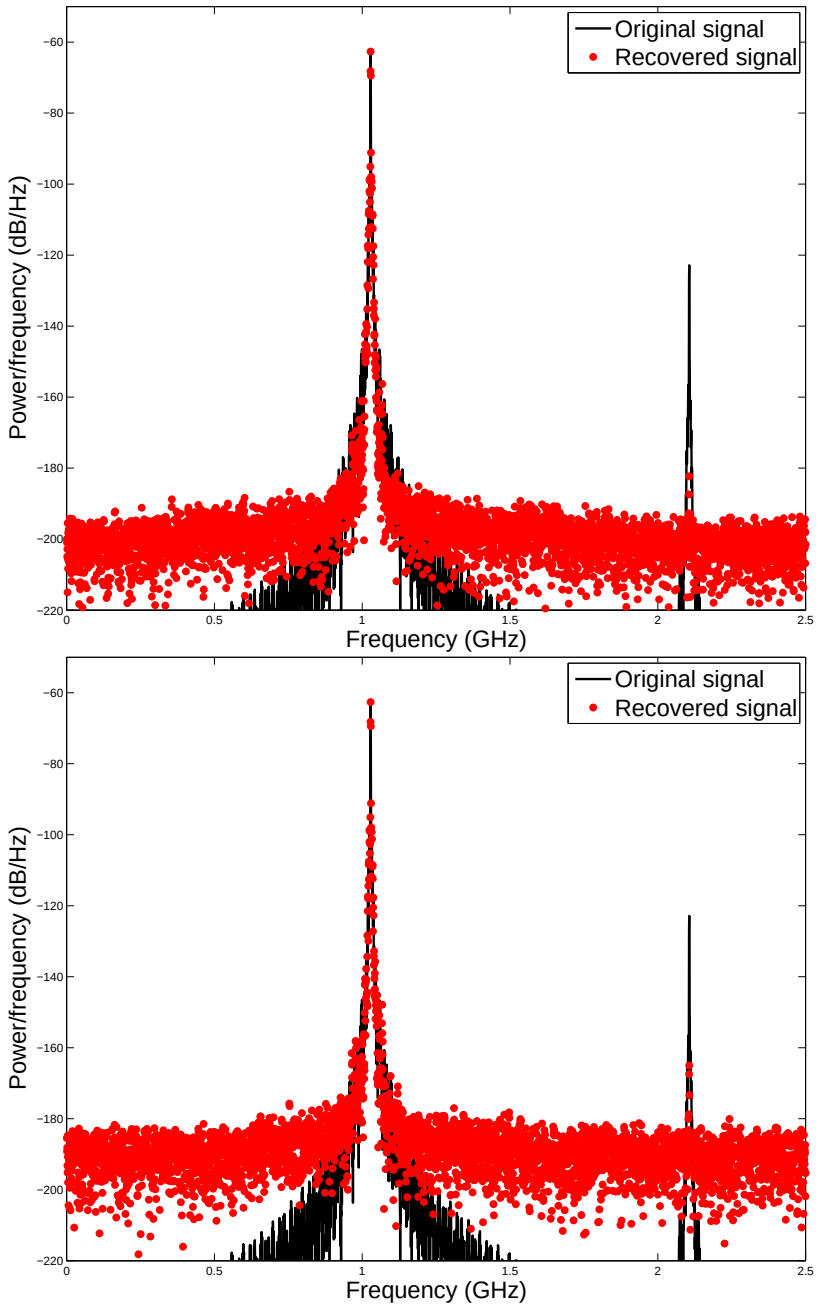


Figure 4.11: Recovery of a small pulse 60 dB below a large pulse. The first plot employs a Dantzig selector, the bottom employs LASSO.

	Dantzig selector				LASSO			
Reweighting step	# cont.	iter.	time	error	# cont.	iter.	time	error
0	7	821	89.2 s	$2.7 \cdot 10^{-3}$	6	569	54.8 s	$2.2 \cdot 10^{-3}$
1	8	1021	114.6 s	$2.1 \cdot 10^{-3}$	7	683	67.8 s	$1.8 \cdot 10^{-3}$

Table 4.1: Details on the simulation used in Figure 4.11. For both the Dantzig selector and LASSO versions, the algorithm was run with continuation and reweighting. The "cont." column is the number of continuation steps, the "iter." column is the number of iterations (over all the continuation steps), the "time" column is in seconds, and the "error" column is the relative ℓ_2 error. Each row is one solve of the Dantzig selector or the LASSO.

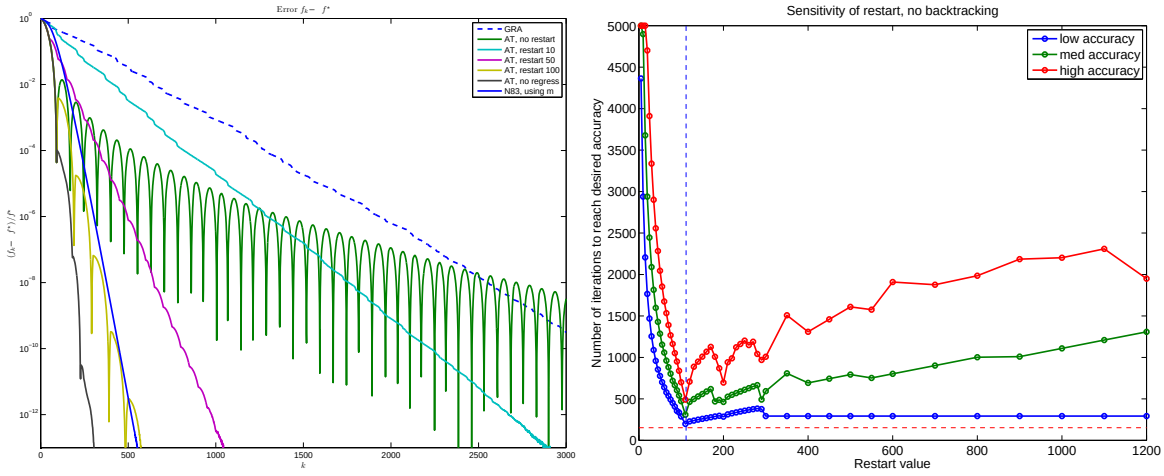


Figure 4.12: Plot (a) shows the Auslender-Teboulle method with and without restart, using various restart parameters. Also shown is the "no regress" restart option, which requires no parameters, and the N83 method using oracle knowledge of the strong convexity constant. Plot (b) shows the number of iterations to reach a given accuracy level, as a function of the restart parameter (using the AT algorithm). The vertical line near 110 is the optimal value of restart according to [GLW09]. A restart value of 0 corresponds to gradient descent, and a restart value of ∞ corresponds to the original AT method.

typically about 2 to 5 iterations. The results are plotted in the frequency domain in Figure 4.11. The large pulse is the dominant spike at about 1.1 GHz, and it is easily recovered to very high precision (the relative ℓ_2 error is less than $5 \cdot 10^{-3}$). The small pulse is at about 2.2 GHz, and because it has an SNR of 0.1 dB and the measurements are undersampled by a factor of 12, it is not possible to recover it exactly, but we can still detect its presence and accurately estimate its carrier frequency.

Table 4.1 reports the computational results of the test, first solving either the Dantzig selector or the LASSO, and then taking a single reweighting step and re-solving. Each call of the algorithm takes about 100 iterations, and the Dantzig selector or the LASSO is solved in about 1 minute, which is impressive since, due to the extremely large size of W, this problem is intractable using an interior-point method.

4.8 Extensions

4.8.1 Automatic restart

When a function is strongly convex, gradient descent converges linearly. If the strong convexity constant m_f is known, then accelerated methods such as N83 also converge linearly, and with a better rate than for gradient descent. However, if m_f is unknown, the accelerated methods do not converge linearly. This is shown in Figure 4.12(a). Our proposed solution from §4.5.6 is to restart the method every R iterations, which then guarantees linear convergence [Nes07]. The optimal rate of convergence is obtained when $R = R(m_f)$, which is unknown *a priori*, so R must be estimated. Figure 4.12(b) shows that as long as $R > R(m_f)$ ($R(m_f)$ is shown as the vertical line), then the method still converges quickly, and hence R is not an extremely sensitive parameter.

However, R may not be scale-invariant, and estimating R is still an issue. One solution is an automatic restart, which requires no parameter. Whenever the objective shows non-monotonic behavior, a restart is forced. Figure 4.12(a) shows this method, called "no regress", compared to the N83 method using "oracle" knowledge of m_f. The automatic method performs extremely well. We note that many restart ideas have been explored in the non-linear conjugate-gradient (CG) literature since the late 1970s [Pow77]. The prevailing intuition for CG, according to [NW06], is that restart should be performed once the iterate is in an approximately quadratic region surrounding the minimum, since then the non-linear CG will perform similarly to linear CG, assuming the line search is good, and convergence will be rapid.

4.8.2 Specialized solvers for certain problems

We consider some specific optimization problems for which there is special structure to exploit. One of the advantages of TFOCS is that the solver is a module that can be swapped, so it is easy to implement specialized algorithms.

4.8.2.1 Noiseless basis pursuit

The noiseless basis pursuit problem,

$$\min_x \|x\|_1 \quad \text{such that} \quad Ax = b$$

has an unconstrained differentiable dual after smoothing is applied; it is also a linear program, so it enjoys the exact-penalty property. Because the dual is unconstrained, it is easy to use a solver like SESOP [EMZ07], BFGS or L-BFGS [BLN95], or a non-linear conjugate-gradient algorithm. Figure 4.13 shows an experiment on a basis pursuit problem with $N = 2048$ variables and a partial DCT measurement matrix A with 1024 rows. The original signal has 146 nonzeros. We compare

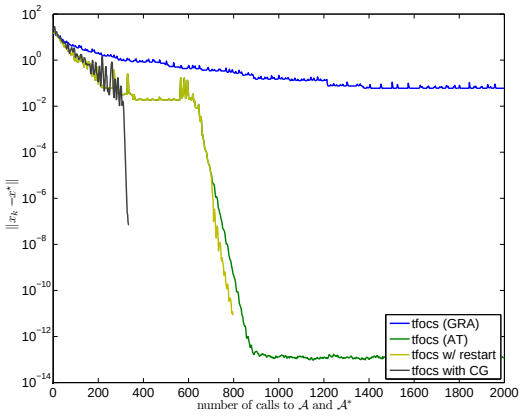


Figure 4.13: Noiseless basis pursuit, comparing a non-linear CG method to an optimal first-order method.

gradient descent, Auslender-Teboulle's (AT) algorithm, the AT algorithm with restart, and a modern non-linear CG method called CG-851 [HZ06]. All of the AT methods are efficient, but the non-linear CG method is even faster. This shows the flexibility of TFOCS and suggests that the long history of work put into CG methods may be applicable to a few special inverse problems. The main difficulty with CG is that standard CG algorithms do not handle constraints, and that the algorithms need appropriate line search. The performance of CG is extremely sensitive to the type of line search; we thank Michael McCoy for providing the line search code used in this experiment. It may also be possible to use specific exact line search algorithms; for example, it is easy to derive an exact line search when the objective contains a ℓ_1 norm and a quadratic term (see, e.g., [WYGZ10]).

We also note recent work on the performance of BFGS on non-smooth problems [LO09, YVGS10] (that is, when the objective is differentiable but not second-order differentiable). This is the case for noiseless basis pursuit (and also for noiseless matrix completion with the nuclear norm).

4.8.2.2 Conic problems in standard form

Consider the conic problem

$$\underset{x}{\text{minimize}} \ \langle c, x \rangle \quad \text{such that} \quad x \geq_{\mathcal{K}} 0, \ Ax = b \tag{4.8.1}$$

where $x \geq_{\mathcal{K}} 0$ means $x \in \mathcal{K}$. If $\mathcal{K} = \mathbb{R}^n_+$, then this reduces to $x \geq 0$ component-wise, and the problem is a linear program (LP). If $\mathcal{K}$ is the Lorentz cone, then the problem is a second-order cone program (SOCP). If $\mathcal{K}$ is the positive semi-definite cone, in which case x and c are Hermitian matrices and the inner product $\langle c, x \rangle = \operatorname{trace} cx$ is the inner product which induces the Hilbert-Schmidt norm, and then the problem is a semi-definite program (SDP).

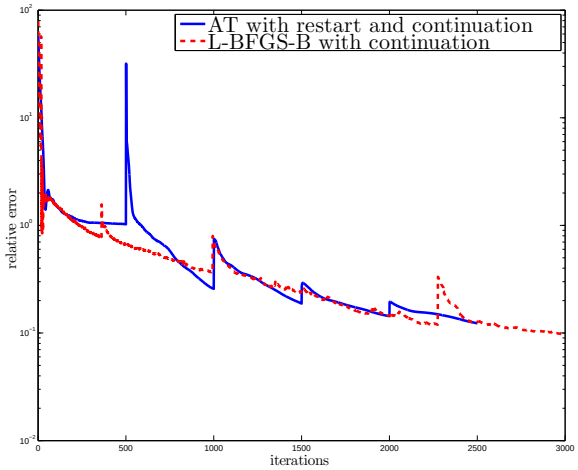


Figure 4.14: Comparison of AT method and L-BFGS-B on a standard form linear program. Both methods behave similarly.

The dual problem is

$$\max_{\nu,\lambda} \; -\langle b, \nu \rangle \quad \text{such that} \quad \lambda \geq_{\mathcal{K}} 0, \quad \lambda = c + A^* \nu$$

which may not be easier to solve than the primal. It is possible to eliminate the variable λ, but we prefer to keep it since it makes it clear that there is a coupling between the inequality and equality constraints.

To apply the dual formulation, we smooth the objective by adding a term $\frac{\mu}{2}\|x - x_0\|^2$ where $\|\cdot\|$ is the norm induced by the appropriate inner-product. This has the effect of uncoupling the dual variables. The dual problem is now

$$\max_{\nu,\lambda} \; -\langle b, \nu \rangle - \frac{1}{2\mu}\|c - \lambda + A^*\nu\|^2 + \langle c - \lambda + A^*\nu, x_0 \rangle \quad \text{such that} \quad \lambda \geq_{\mathcal{K}} 0.$$

The dual is quadratic with no equality constraints and a simple inequality constraint. For $\mathcal{K} = \mathbb{R}^n_+$, this is part of a box-constraint, and many algorithms take advantage of its simplicity. Figure 4.14 shows a comparison between TFOCS using the AT method (with restart and continuation) and the L-BFGS-B method [BLN95, ZBLN97]. L-BFGS-B is a box constrained version of the limited memory BFGS quasi-Newton method. The BFGS method is well-regarded as one of the fastest methods, when it is applicable. In the figure, we see that L-BFGS-B performs about the same as the optimal method, which is slightly discouraging. There may be tweaks or special considerations which can make L-BFGS-B achieve the same performance as BFGS.

The technique of adding a quadratic term to a linear program has been used since at least Mangasarian in the early 1980s [Man81].

4.8.2.3 Matrix completion problems

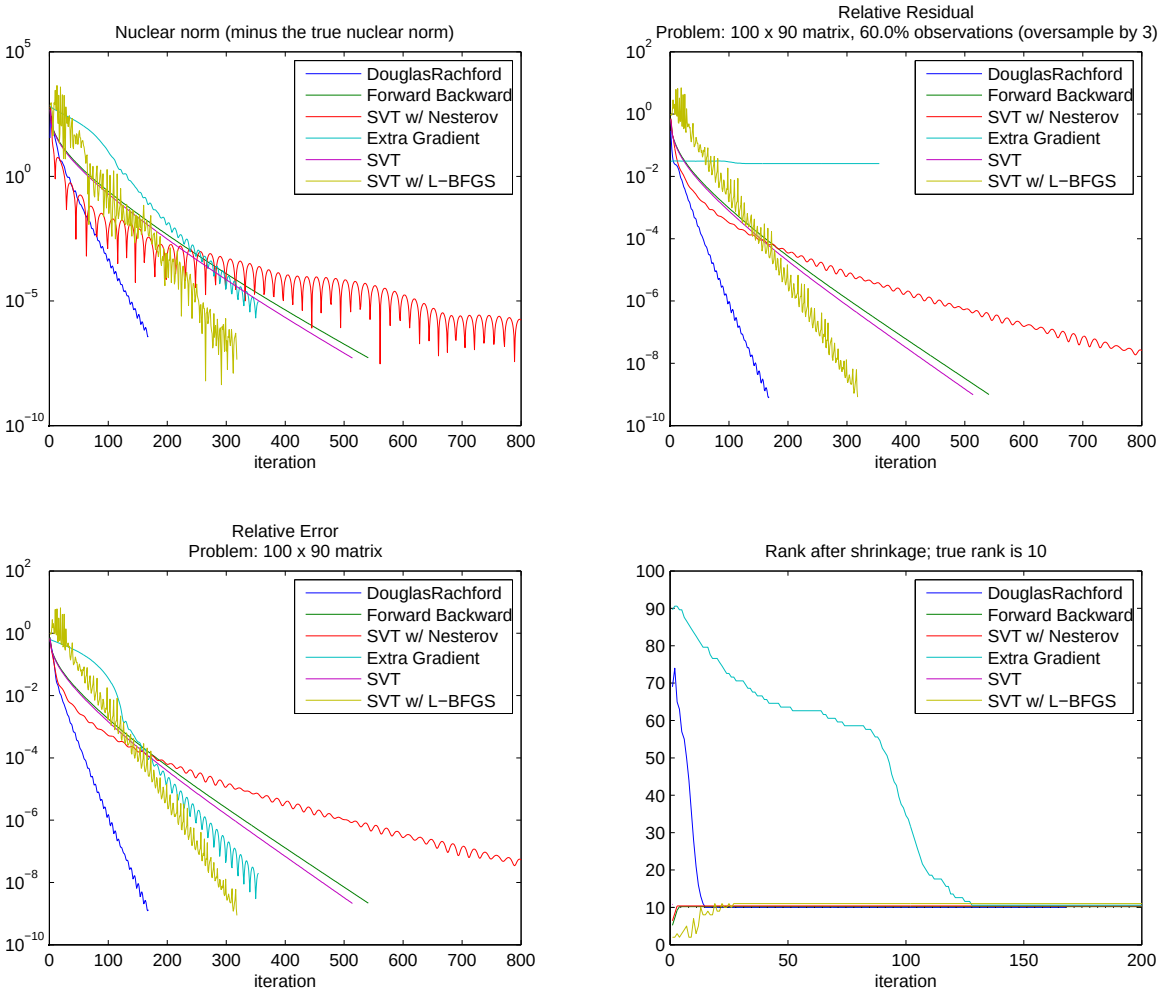


Figure 4.15: Comparison of various algorithms for noiseless matrix completion.

The noiseless matrix completion problem has many similarities with the noiseless basis pursuit problem; in particular, the smoothed dual function is unconstrained. This opens the possibility of using many algorithms, such as non-linear CG, SESOP, and BFGS. Figure 4.15 compares the performance of 5 algorithms:

- Singular value thresholding (SVT) [CCS10], which is equivalent to TFOCS using gradient descent and a fixed step size, and without continuation.
- SVT with the N83 Nesterov method, at a fixed step size and without continuation.
- SVT with limited memory BFGS (L-BFGS) [NW06] and fixed step size (no line search).
- The extra gradient algorithm [Kor76], which is a splitting method with an additional variable.
- The forward-backward splitting method, which is equivalent to the formulation in FPCA [MGC09].

- The Douglas-Rachford splitting method [Com04].

To be fair, the comparisons use fixed-step sizes for all algorithms; of course, most of the algorithms benefit enormously from using line search, as we have shown in this chapter, so these results are not necessarily "real-world" results.

The special consideration in the matrix completion problem is that the matrix shrinkage step is expensive to compute; see the discussion in §4.7.4. One method, used in [MGC09], is to use randomized linear algebra to compute an approximate SVD. Another method is using a Lanczos-based method such as PROPACK [Lar04]. For a Lanczos-based method, only singular values above a threshold size are needed; the number of iterations in the SVD computation is directly proportional to the number of singular values above this threshold. Furthermore, the cost of each matrix-vector multiply depends on the rank of the previous iterate, since the current iterate is a combination of a sparse and low-rank matrix.

Figure 4.15 shows results using a dense SVD factorization, but also records the rank. All the algorithms converge quickly, but Douglas-Rachford and SVT-with-L-BFGS are considerably faster than the rest. The lower-right plot shows the ranks of the variables at every iteration. The benefit of SVT-based methods is that the rank typically starts at 0 and increases slowly to a steady-state value (this need not be true with line search variants). The Douglas-Rachford algorithm, on the other hand, takes a very aggressive first few steps and the rank is very high, before coming lower. On a large-scale problem, this means the first few iterations might be too expensive to even compute. The extra-gradient method has even slower decay toward the true rank.

Despite the issues with rank, the Douglas-Rachford method is promising due to the rapid convergence, so it may be possible to adjust it to keep the ranks smaller. Thanks to Jalal Fadili for ideas and code regarding these splitting methods.

With matrix completion problems, there is tremendous room for improvement, since high-accuracy SVDs are unnecessary in the initial steps of the algorithm. There has been some reported success of using methods from [HMT11] with the matrix completion problem because these types of approximate SVDs can take advantage of warm-starts. Another recent paper mentions improvements using warm-started Lanczos methods [LW10].

4.9 Software: TFOCS

The work described in this paper has been incorporated into a software package, Templates for First-Order Conic Solvers (TFOCS, pronounced *tee-fox*), which is publicly available at `http://tfocs.stanford.edu`. As its name implies, this package is a set of templates, or building blocks, that can be used to construct efficient, customized solvers for a variety of models.

To illustrate the usage of the software, let us show how to construct a simple solver for the

smoothed Dantzig selector model described in §4.3. We will begin by assuming that we are given the problem data `A`, `b`, `delta`, and a fixed smoothing parameter `mu`. The basic solver templates require two functions to complete their work. The first function computes the value and gradient of g_{smooth}, the smooth component of the composite dual:

```
function [ val, grad, x ] = g_dantzig( A, y, x0, mu, z )
x    = SoftThreshold( x0 - (1/mu) * A' * ( A * z ), 1/mu );
grad = A' * ( y - A * x );
val  = z' * grad - norm( x, 1 ) - 0.5 * mu * norm( x - x0 ) .^ 2;
```

The second function computes the generalized projection associated with the nonsmooth component of the dual, which is in this case $h(z) = \delta\|z\|_1$.

```
h = prox_scale( tfocs_prox( @(z)norm(z,1), @SoftThreshold ), delta );
```

The `tfocs_prox` function combines h and the proximity operator of h in a fashion that the software understands, and `prox_scale` adjusts h from $\|z\|_1$ to $\delta\|z\|_1$ and also updates the proximity operator. The functions h and g depend on the soft-thresholding operator:

```
function y = SoftThreshold( x, t )
y = sign( x ) .* max( abs( x ) - t, 0 );
```

Armed with these functions, the following code solves the smoothed Dantzig selector, using the Auslender/Teboulle first-order variant and the default choices for line search and stopping criteria.

```
function x = Dantzig_smoothed( A, y, delta, mu )
[m,n] = size(A);
x0    = zeros(n,1); z0 = zeros(n,1);
g_sm  = @(z) g_dantzig( A, y, x0, mu, z );
affine = [];
z = tfocs_AT( g_sm_op, affine, h, z0 );
x = SoftThreshold( x0 - (1/mu) * A' * ( A * z ), 1/mu );
```

The second-to-last line of code calls the AT solver.

This simple solver is likely to be unnecessarily inefficient if `A` exhibits any sort of fast operator structure, but this can be remedied rather simply. Note that the solver itself needs to have no knowledge of `A` or `y` above; its interaction with these quantities comes only through calls to `g_dantzig`. Therefore, we are free to *rewrite* `g_dantzig` in a more numerically efficient manner: for instance, if `A` is derived from a Fourier transform, we may substitute fast Fourier transform operations for matrix-vector multiplications. This simple change will reduce the cost of each iteration from $\mathcal{O}(mn)$

to $\mathcal{O}(n \log n)$, and the storage requirements from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. For large-scale problems these savings can be quite significant.

A further improvement in performance is possible through careful management of the linear operator calculations as described in §4.5.4. The TFOCS solver templates can perform this management automatically. To take advantage of it, we replace `g_dantzig` with two functions: one which implements the underlying linear operator, and one which implements the remainder of the smooth function. The details of how to accomplish this are best left to the user guide [BCG10b].

Of course, as part of the complete package, we have supplied a solver for the Dantzig selector that exploits each of these efficiencies, and others. Similar solvers have been created for the LASSO, TV, and other models discussed here. But the flexibility of the lower-level templates will allow these same efficiencies to be achieved for many models that we have not discussed here. In fact, evidently the templates are not restricted to our specific conic form (4.1.9) or its dual, and we hope the software finds application outside of the compressed sensing domain as well.

With the software release is a detailed user guide [BCG10b] that covers the usage, documents the most popular problem formulations, and provides some examples. We refer the reader to the user guide for further software details.

4.10 Discussion

We have developed a convenient framework for constructing first-order methods, which is flexible and handles a variety of convex cone problems, including problems that did not have efficient algorithms before. On the implementation side, we have introduced ideas which lead to novel, stable, and efficient algorithms. When comparing our implementation on specific problems such as the LASSO, which have been intensively studied, our techniques appear surprisingly competitive with the state of the art.

The templates from this chapter are flexible in a manner that has not yet been seen in sparse recovery software. Not only are the solvers interchangeable and hence easy to benchmark one algorithm against another, but they work with a wide variety of formulations which we hope will greatly enable other researches. It is also our goal that the software will be easy to use for non-experts, and to this end our future research will be to improve the usability of the software and to provide sensible default parameters. One major topic to pursue is choosing the smoothing parameter μ. Further efforts will also be made to improve the line search to take advantage of strong convexity, to better manage expensive projections, and to use scaled norms so that dual variables are all on the same scale (this is an issue only for objectives with several additive terms, as in the TV with analysis problem in §4.4.6 and §4.7.3).

A subsequent paper will cover these issues, as well as further investigating local strong convexity

and how to take advantage of this in an optimal manner, and improving the accelerated continuation scheme by taking into account the inexact solves. The software and user guide [BCG10b] will be kept up-to-date and supported.

4.11 Appendix: exact penalty

The general approach proposed in this chapter is to add a strongly convex term to the primal objective function in order to smooth the dual objective. We now draw a parallel with augmented Lagrangian and penalty function methods that eliminate constraints by incorporating a penalty term p into the objective, e.g., $\min_{x:Ax=b} f(x)$ becomes $\min_x f(x) + \lambda p(Ax - b)$. For reasonable choices of p, the two problems become equivalent as $\lambda \to \infty$. Remarkably, for some special choices of p, the two problems are equivalent for a large but *finite* value of λ. Usually, p is a non-differentiable function, such as $p = \|\cdot\|_1$ or $p = \|\cdot\|_2$ (not $p = \|\cdot\|_2^2$); see Bertsekas' book [BNO03] for a discussion.

Our approach uses a different type of perturbation, since our goal is not to eliminate constraints but rather to smooth the objective. But we use the term "exact penalty" because, for some problems, we have a similar result: the smoothed and unsmoothed problems are (nearly) equivalent for some $\mu > 0$. The result below also departs from traditional exact penalty results because our perturbation $\frac{1}{2}\|x - x_0\|_2^2$ *is* smooth.

Below we present the proof of Theorem 4.3.1, which exploits the polyhedral constraint set of linear programs. This exact proof of the theorem may be original, but the theorem itself is not a new result, as explained in the text.

Proof of Theorem 4.3.1. We consider the linear program

$$\begin{array}{ll} \text{minimize} & \langle c, x\rangle \\ \text{subject to} & x \in \mathcal{P}, \end{array} \tag{LP}$$

where $\mathcal{P}$ is a convex polyhedron, and its perturbed version

$$\begin{array}{ll} \text{minimize} & \langle c, x\rangle + \frac{1}{2}\mu\langle x - x_0, x - x_0\rangle_Q \\ \text{subject to} & x \in \mathcal{P}, \end{array} \tag{QP}$$

where $\langle x, y\rangle_Q \triangleq \langle x, Qy\rangle$ for some positive semidefinite Q. Let $E^\star$ be the solution set of (LP) ($E^\star$ is a vertex or a face of the feasible polyhedron) and let $x^\star$ be any point in $E^\star$ such that $\langle x - x_0, x - x_0\rangle_Q$ is minimum. (Note that when Q is the identity, $x^\star$ is the usual projection onto the convex set $E^\star$.) With $f_\mu(x) \triangleq \langle c, x\rangle + \frac{1}{2}\mu\langle x - x_0, x - x_0\rangle_Q$, it is of course sufficient to show that $x^\star$ is a local minimizer of f_μ to prove the theorem. Put differently, let $x \in \mathcal{P}$ and consider $x^\star + t(x - x^\star)$ where $0 \le t \le 1$.

Then it suffices to show that $\lim_{t\to 0^+} f_\mu(x^\star + t(x - x^\star)) \geq f_\mu(x^\star)$. We now compute

$$\begin{aligned} f_\mu(x^\star + t(x - x^\star)) &= \langle c, x^\star + t(x - x^\star)\rangle + \tfrac{1}{2}\mu\langle x^\star + t(x - x^\star) - x_0, x^\star + t(x - x^\star) - x_0\rangle_Q \\ &= f_\mu(x^\star) + t\langle c, x - x^\star\rangle + \mu t\langle x - x^\star, x^\star - x_0\rangle_Q + \tfrac{1}{2}\mu t^2\langle x - x^\star, x - x^\star\rangle_Q. \end{aligned}$$

Therefore, it suffices to establish that for all $x \in \mathcal{P}$,

$$\langle c, x - x^\star\rangle + \mu\langle x - x^\star, x^\star - x_0\rangle_Q \geq 0,$$

provided μ is sufficiently small. Now $x \in \mathcal{P}$ can be expressed as a convex combination of its extreme points (vertices) plus a non-negative combination of its extreme directions (in case $\mathcal{P}$ is unbounded). Thus, if $\{v_i\}$ and $\{d_j\}$ are the finite families of extreme points and directions, then

$$x = \sum_i \lambda_i v_i + \sum_j \rho_j d_j,$$

with $\lambda_i \geq 0$, $\sum_i \lambda_i = 1$, $\rho_j \geq 0$. The extreme directions—if they exist—obey $\langle c, d_j\rangle \geq 0$ as otherwise, the optimal value of our LP would be $-\infty$. Let I denote those vertices which are *not* solutions to (LP), i.e., such that $\langle c, v_i - \hat{x}\rangle > 0$ for any $\hat{x} \in E^\star$. Likewise, let J be the set of extreme directions obeying $\langle c, d_j\rangle > 0$. With this, we decompose $x - x^\star$ as

$$x - x^\star = \Big(\sum_{i\notin I} \lambda_i(v_i - x^\star) + \sum_{j\notin J} \rho_j d_j\Big) + \Big(\sum_{i\in I} \lambda_i(v_i - x^\star) + \sum_{j\in J} \rho_j d_j\Big).$$

It is not hard to see that this decomposition is of the form

$$x - x^\star = \alpha(\hat{x} - x^\star) + \Big(\sum_{i\in I} \lambda_i(v_i - x^\star) + \sum_{j\in J} \rho_j d_j\Big),$$

where $\hat{x} \in E^\star$ and α is a non-negative scalar. This gives

$$\langle c, x - x^\star\rangle = \sum_{i\in I} \lambda_i\langle c, v_i - x^\star\rangle + \sum_{j\in J} \rho_j\langle c, d_j\rangle.$$

Let $\alpha_i \geq 0$ (resp. $\beta_j \geq 0$) be the cosine of the angle between c and $v_i - x^\star$ (resp. c and d_j). Then for $i \in I$ and $j \in J$, we have $\alpha_i > 0$ and $\beta_j > 0$. We can write

$$\langle c, x - x^\star\rangle = \sum_{i\in I} \lambda_i\alpha_i\|c\|_2\|v_i - x^\star\|_2 + \sum_{j\in J} \rho_j\beta_j\|c\|_2\|d_j\|_2,$$

which is strictly positive. Furthermore,

$$\begin{aligned}\langle x - x^\star, x^\star - x_0\rangle_Q &= \alpha\,\langle \hat{x} - x^\star, x^\star - x_0\rangle_Q + \Big\langle \sum_{i\in I} \lambda_i (v_i - x^\star) + \sum_{j\in J} \rho_j d_j, x^\star - x_0 \Big\rangle_Q \\ &\geq \Big\langle \sum_{i\in I} \lambda_i (v_i - x^\star) + \sum_{j\in J} \rho_j d_j, x^\star - x_0 \Big\rangle_Q \\ &\geq -\|Q(x^\star - x_0)\|_2 \Big(\sum_{i\in I} \lambda_i \|v_i - x^\star\|_2 + \sum_{j\in J} \rho_j \|d_j\|_2 \Big).\end{aligned}$$

The second inequality holds because $x^\star$ minimizes $\langle \hat{x} - x_0, \hat{x} - x_0\rangle_Q$ over $E^\star$, which implies $\langle \hat{x} - x^\star, x^\star - x_0\rangle_Q \geq 0$. The third is a consequence of the Cauchy-Schwartz inequality. In conclusion, with $\mu_Q \triangleq \mu\|Q(x^\star - x_0)\|_2$, we have

$$\langle c, x - x^\star\rangle + \mu\langle x - x^\star, x^\star - x_0\rangle_Q \geq \sum_{i\in I} (\alpha_i \|c\|_2 - \mu_Q)\lambda_i \|v_i - x^\star\|_2 + \sum_{j\in J} (\beta_j \|c\|_2 - \mu_Q)\rho_j \|d_j\|_2,$$

and it is clear that since $\min_i \alpha_i > 0$ and $\min_j \beta_j > 0$, selecting μ small enough guarantees that the right-hand side is non-negative. □

4.12 Appendix: creating a synthetic test problem

It is desirable to use test problems that have a precisely known solution. In some cases, such as compressed sensing problems in the absence of noise, the solution may be known, but in general this is not true. A common practice is to solve problems with an interior-point method (IPM) solver, since IPM software is mature and accurate. However, IPMs do not scale well with the problem size, and cannot take advantage of fast algorithms to compute matrix-vector products. Another disadvantage is that the output of an IPM is in the interior of the feasible set, which in most cases means that it is not sparse.

Below, we outline procedures that show how to generate problems with known exact solutions (to machine precision) for several common problems. The numerical experiments earlier in this chapter used this method to generate the test problems. This is inspired by [Nes07], but we use a variation that gives much more control over the properties of the problem and the solution.

Basis pursuit. Consider the basis pursuit problem and its dual

$$\begin{array}{lll} \text{minimize} & \|x\|_1 & \qquad \text{maximize} \quad \langle y, \lambda\rangle \\ \text{subject to} & Ax = y, & \qquad \text{subject to} \quad \|A^*\lambda\|_\infty \leq 1. \end{array}$$

At the optimal primal and dual solutions $x^\star$ and $\lambda^\star$, the KKT conditions hold:

$$Ax^\star = y \quad \|A^*\lambda^\star\|_\infty \leq 1 \quad (A^*\lambda^\star)_T = \operatorname{sign}(x^\star_T),$$

where $T = \operatorname{supp}(x^\star)$.

To generate the exact solution, the first step is to choose A and y. For example, after choosing A, y may be chosen as $y = A\tilde{x}$ for some $\tilde{x}$ that has interesting properties (e.g., $\tilde{x}$ is s-sparse or is the wavelet coefficient sequence of an image). Then any primal dual solver is run to high accuracy to generate $x^\star$ and $\lambda^\star$. These solutions are usually accurate, but not quite accurate to machine precision.

The idea is that $x^\star$ and $\lambda^\star$ are exact solutions to a slightly perturbed problem. Define $T = \operatorname{supp}(\hat{x})$; in sparse recovery problems, or for any linear programming problem, we have $|T| \leq m$ where m is the length of the data vector y. The matrix A is modified slightly by defining $\tilde{A} \leftarrow AD$ where D is a diagonal matrix. D is calculated to ensure that $\|(DA^*\lambda^\star)_{T^c}\|_\infty < 1$ and $(DA^*\lambda^\star)_T = \operatorname{sign}(x^\star)$. If the original primal dual solver was run to high accuracy, D is very close to the identity. In practice, we observe that the diagonal entries of D are usually within .01 of 1.

The primal variable $x^\star$ is cleaned by solving $\tilde{A_T}x^\star_T = y$; this is unique, assuming A_T has full column rank. Alternatively, $x^\star$ can be made arbitrary (e.g., hard thresholded, to give it a well-defined support) and then y redefined as $y = \tilde{A}x^\star$. If the original problem was solved to high accuracy (in which case D will have all positive entries), then the cleaning-up procedure does not affect the sign of $x^\star$. The vectors $x^\star$ and $\lambda^\star$ are now optimal solutions to the basis pursuit problem using $\tilde{A}$ and y.

The LASSO. The inequality constrained version of basis pursuit, called variously the LASSO or basis pursuit denoising (BPDN), admits exact solutions in a similar fashion. The problem and its dual are

$$\begin{array}{llll} \text{minimize} & \|x\|_1 & \text{maximize} & \langle y, \lambda\rangle - \varepsilon\|\lambda\|_2 \\ \text{subject to} & \|Ax - y\|_2 \leq \varepsilon, & \text{subject to} & \|A^*\lambda\|_\infty \leq 1. \end{array} \tag{4.12.1}$$

Write $z = y - Ax$. At optimal solutions $x^\star, \lambda^\star, z^\star$, the KKT conditions hold:

$$\|z^\star\|_2 \leq \varepsilon, \quad A^T\lambda^\star \in \partial\|x^\star\|_1, \quad \|A^T\lambda^\star\|_\infty \leq 1, \quad \langle\lambda^\star, z\rangle = \varepsilon\|\lambda^\star\|_2.$$

If the sub-gradient condition $A^T\lambda^\star \in \partial\|x^\star\|_1$ holds for some $\lambda^\star$ and $x^\star$, then defining $z = \frac{\varepsilon}{\|\lambda^\star\|}\lambda^\star$ will ensure that all the other KKT conditions hold. Modifying z is equivalent to modifying y.

In practice, given data A, y, and ε for which it is desired to have a known solution, a primal dual solver runs to high (but not perfect) accuracy to generate $\lambda^\star$ and $x^\star$ so that $A^T\lambda^\star$ is *almost* in $\partial\|x^\star\|_1$. The primal variable x can be "cleaned-up" to a new version $x^\star$ (e.g., hard-thresholded), and then A is modified by a diagonal matrix $\tilde{A} \leftarrow AD$ so that now $A^T\lambda^\star \in \partial\|x^\star\|_1$ is exactly true.

The variable z is set as above, implicitly defining $\tilde{y}$.

Unconstrained variants. The unconstrained problem (4.1.7) (which is sometimes referred to as the LASSO; to keep this distinct from (4.12.1), we call them the *unconstrained LASSO* and *constrained LASSO*, respectively),

$$\text{minimize} \qquad \tfrac{1}{2}\|Ax - y\|_2^2 + \lambda\|x\|_1, \tag{4.12.2}$$

admits exact test problems in a fashion very similar to the BPDN formulation described above. Here, λ is a fixed parameter, not a dual variable. We note that given an exact unconstrained LASSO problem with data A, y, and λ, and known exact solution $x^\star$, you also have a BPDN problem with known solution if the ε parameter in BPDN is set $\varepsilon_\lambda = \|Ax^\star - b\|_2$; the converse is also true, i.e., given a BPDN problem with solution $x^\star$, the parameter of the unconstrained LASSO problem is just $\|z\|_\infty$.

We also remark on the very recent work [Lor11] which appeared a week before this thesis was submitted. In that work, the author proposes a method for finding exact test problems for the unconstrained LASSO. Given a desired solution $x^\star$ and matrix A, the main burden is finding $\lambda^\star$ such that $A^T\lambda^\star \in \partial\|x^\star\|_1$, and several methods are proposed, such as solving a quadratic program. One key benefit is that $x^\star$ and A can be completely specified, but the downside is less control over y.

One of the proposed methods in [Lor11] is finding a vector w such that $A^T\lambda = w$ and $w \in \partial\|x^\star\|_1$. Thus w is in the intersection of two convex sets: the range of A^T and the subgradient set. The projection onto convex sets (POCS) algorithm (i.e., alternating projection) method solves this, provided the intersection is non-empty (which is not guaranteed if $x^\star$ is chosen arbitrarily). A suitable $\lambda_{w^\star}$ can be recovered at the end by $\lambda = A^\dagger w$.

Using this POCS method, we propose a hybrid method to find a w in the two sets. The idea is to use $w = A^T\lambda^\star$ as the initial guess. Thus instead of modifying the columns of A to make a perturbed matrix, the dual solution $\lambda^\star$ is perturbed to $\lambda_{w^\star}$. In practice, this works as well as the initial perturbation idea, but without the drawback of perturbing A to $\tilde{A}$, and it works better than using an arbitrary initial value for w since the residual $Ax - b$ is changed less.

Other problems. For basis pursuit and the constrained LASSO, it is possible to obtain exact solutions to the *smoothed* problem (with $d(x) = \frac{1}{2}\|x - x_0\|_2^2$). For the Dantzig selector, an exact solution for the smoothed problem can also be obtained in a similar fashion. To find an exact solution to the unsmoothed problem, we take advantage of the exact penalty property from §4.3.4 and simply find the smoothed solution for a sequence of problems to get a good estimate of x_0 and then solve for a very small value of μ.

Chapter 5

Conclusion

Since each chapter has its own conclusion, we conclude this thesis not with a summary, but with possibilities for future work. Most of the issues raised by NESTA are addressed by the TFOCS algorithm, so we do not discuss extensions of NESTA.

5.1 Future of A2I

The current chip is expected to be fully calibrated by Summer 2011, and this will give many new insights into the design. The RMPI is necessarily a prototype chip and is programmable in many ways, allowing us to modify it as necessary; future versions will be more streamlined and have better performance. Since calibration is on-going, this is an active area of research, and new methods are being tested weekly. The end-goal is not just an accurate calibration for academic purposes, but a calibration scheme that is robust, easy, and fast.

The biggest advance in an A2I system would be a novel architecture that is specialized to receive a specific signal class. For example, a pulse *detection* system that only needs to detect the presence of a pulse, or perhaps only estimate a frequency, allows much more freedom than a system that tries to reconstruct the input signal exactly. Putting a matched-filter system on-chip would be a major step in this direction. Another ambitious target would be to implement special purpose coding matrices, such as expander graphs, and decode on-chip in real-time. It may be possible to make an adaptive system that uses a crude detection first-step in order to give information, in real-time, to a second-stage system which "zooms" to the frequencies of interest.

Another hardware improvement would be to use different types of chipping signals. This is governed by both theoretical considerations (a random sum of shifted sinc-like functions would be nice) and by practical considerations, since the chipping signal should put as little strain on the hardware as possible, and be as robust as possible to clock jitter. We note recent work on improving the basic ± 1 chipping signal [HBC11, ME09b], and the work of the modulated wide-band converter [MEE09] which has many good ideas.

For reconstruction, the NESTA and TFOCS algorithms have made large scale reconstructions possible at an academic level, but real-time reconstruction is still far away. The best method to achieve ultra-fast processing is with the compressive matched filter [ERW11], though the performance degrades when the input signal is complicated, since the parameter space becomes large so the algorithm slows down.

To improve the dynamic range of the system, it seems a tighter signal model is needed. Work on model-based compressive sensing [BCDH10,Cev09] suggests tree-based models can improve recovery, though the exact amount of improvement is likely to depend strongly on the specific signal class involved. Using a reweighted scheme with weights picked in accordance with a hierarchical structure, such as in [DWB08], is an obvious way to apply these models to ℓ_1 techniques without having to use greedy methods. Other ways to improve the system performance is to exploit all prior knowledge of the errors, such as estimating the covariance matrix of the noise, and using iterative algorithms to reduce the effect of non-linearities and jitter [WG11].

Other work on recovery is to make the system capable of studying a continuous stream of data, rather than a single window at a time. To this end, a windowing scheme will be essential, and the exact details still need to be worked out.

5.2 Improvements to TFOCS

Our biggest goal for TFOCS is to develop a robust convergence theory of the accelerated continuation scheme. This is not of just theoretical interest, but is also useful in practice since it may suggest what value to set the smoothing parameter μ, and determine what the stopping tolerance of the inner iterations should be. Ultimately, the user should not need to provide any parameters. The specialized line search has already obviated the need for a step-size parameter, and we are close to finding an automatic way to determine when to restart the algorithm.

Certain special problems, such as matrix completion, provide many avenues for future research. The SVD computation in matrix completion only requires the singular vectors that correspond to singular values above a given threshold, so the entire SVD is not needed. We predict that randomized linear algebra will play a big role in speeding up computations. Due to non-negligible errors, optimization algorithms need to gracefully accept approximate computation of gradients and still guarantee convergence.

We plan to continue support of the software package and incorporate new solvers, such as BFGS and Douglas-Rachford. "Splitting methods" such as Douglas-Rachford, though not well-known in some communities, may be very efficient at certain problems. Other planned improvements include automatic diagonal scaling to improve performance on "stiff" problems, and perhaps incorporating the solvers into a modeling environment like CVX [GB10] or porting the software to a free language

like R or Python.

Bibliography

[ABDF11] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo, *An augmented Lagrangian approach to the constrained optimization formulation of imaging inverse problems*, IEEE Trans. Image Process. **20** (2011), no. 3, 681–695.

[AT06] A. Auslender and M. Teboulle, *Interior gradient and proximal methods for convex and conic optimization*, SIAM J. Optim. **16** (2006), no. 3, 697–725.

[Auj09] J.-F. Aujol, *Some first-order algorithms for total-variation based image restoration*, J. Math. Imaging Vision **34** (2009), no. 3, 307–327.

[Bar07] R. Baraniuk, *Compressive sensing*, IEEE Sig. Proc. Mag. **24** (2007), no. 4, 118–121.

[BB88] J. Barzilai and J. Borwein, *Two point step size gradient method*, IMA J. Numer. Anal. **8** (1988), 141–148.

[BBC11] S. Becker, J. Bobin, and E. J. Candès, *NESTA: a fast and accurate first-order method for sparse recovery*, SIAM J. Imaging Sci. **4** (2011), no. 1, 1–39.

[BC09] J. Bobin and E. J. Candès, *A fast and accurate first-order algorithm for compressed sensing*, Proc. IEEE ICIP (2009), 1457–1460.

[BC11] H. H. Bauschke and P. L. Combettes, *Convex analysis and monotone operator theory in hilbert spaces*, Springer-Verlag, New York, 2011.

[BCDH10] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde, *Model-based compressive sensing*, IEEE Trans. Info. Theory **56** (2010), no. 4, 1982–2001.

[BCG10a] S. Becker, E. J. Candès, and M. Grant, *Templates for convex cone problems with applications to sparse signal recovery*, Tech. Report 1009.2065, arXiv, 2010.

[BCG10b] ———, *Templates for first-order conic solvers user guide*, Tech. report, Stanford, 2010, Preprint available at `http://tfocs.stanford.edu`.

[BD09] T. Blumensath and M. E. Davies, *Iterative hard thresholding for compressed sensing*, Harm. Anal. **27** (2009), no. 3, 265–274.

[BDDW08] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, *A simple proof of the restricted isometry property for random matrices*, Constructive Approximation **28** (2008), no. 3, 253–263.

[BDF07] J. M. Bioucas-Dias and M. A. T. Figueiredo, *A new TwIST: two-step iterative shrinkage/thresholding algorithms for image restoration*, IEEE Trans. Image Process. **16** (2007), no. 12, 2992–3004.

[Ber75] D. P. Bertsekas, *Necessary and sufficient conditions for a penalty method to be exact*, Math. Program. **9** (1975), 87–99.

[BGI+08] R. Berinde, A. Gilbert, P. Indyk, H. Karloff, and M. Strauss, *Combining geometry and combinatorics: a unified approach to sparse signal recovery*, Allerton, 2008.

[BKR09] P. T. Boufounos, G. Kutyniok, and H. Rauhut, *Sparse recovery from combined fusion frame measurements*, IEEE Tran. Info. Theory, *to appear* (2009).

[BLN95] R. H. Byrd, P. Lu, and J. Nocedal, *A limited memory algorithm for bound constrained optimization*, SIAM J. Sci. Stat. Comp. **16** (1995), no. 5, 1190–1208.

[Blu11] T. Blumensath, *Compressed sensing with nonlinear observation*, IEEE Trans. Info. Theory, *under review* (2011), Available at `http://www.personal.soton.ac.uk/tb1m08/papers/B_Nonlinear.pdf`.

[BM90] N. M. Blachman and H. Mousavinezhad, *The spectrum of the square of a synchronous random pulse train*, IEEE Trans. on Comm. **38** (Jan 1990), no. 1.

[BM03] S. Burer and R.D.C. Monteiro, *A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization*, Math. Prog. (series B) **95** (2003), no. 2, 329–357.

[BM05] ———, *Local minima and convergence in low-rank semidefinite programming*, Math. Prog. (series A) **103** (2005), no. 3, 427–444.

[BMR00] G. Birgin, J. M. Martinez, and M. Raydan, *Nonmonotone spectral projected gradient methods on convex sets*, SIAM J. Optim. **10** (2000), 1196–1211.

[BNO03] D. P. Bertsekas, A. Nedić, and A. E. Ozdaglar, *Convex analysis and optimization*, Athena Scientific, 2003.

[BPC+10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, 2010, Available at `http://www.stanford.edu/~boyd/papers/admm_distr_stats.html`.

[Bre95] L. Breiman, *Better subset regression using the nonnegative garotte*, Technometrics **37** (1995), 373–384.

[BS07] H. Y. Benson and D. F. Shanno, *An exact primal-dual penalty method approach to warmstarting interior-point methods for linear programming*, Comp. Opt. Appl. **38** (2007), 371–399.

[BSB10] D. Baron, S. Sarvotham, and R. Baraniuk, *Bayesian compressive sensing via belief propagation*, IEEE Trans. Sign. Proc. **58** (2010), no. 1, 269–280.

[BSI09] BSIM Research Group, UC Berkeley, *BSIM4*, 2009, Available at `http://www-device.eecs.berkeley.edu/~bsim3/bsim4_intro.html`.

[BSO08] J. Bobin, J.-L. Starck, and R. Ottensamer, *Compressed sensing in astronomy*, IEEE J. Sel. Top. Signal Process. **2** (2008), no. 5, 718–726.

[BT09] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. on Imaging Sci. **2** (2009), no. 1, 183–202.

[BT10] ———, *Gradient-based algorithms with applications in signal recovery problems*, Convex Optimization in Signal Processing and Communications (D. Palomar and Y. Eldar, eds.), Cambridge University Press, 2010.

[BV04] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge University Press, 2004.

[Can06] E. J. Candès, *Compressive sampling*, Int. Cong. Math. **3** (2006), 1433–1452.

[Can08] ———, *The restricted isometry property and its implications for compressed sensing*, Compte Rendus de l'Academie des Sciences, Paris, Serie I **346** (2008), 589–592.

[CCS10] J-F. Cai, E. J. Candès, and Z. Shen, *A singular value thresholding algorithm for matrix completion*, SIAM J. Optim. **20** (2010), 1956–1982.

[CDDY06] E. J. Candès, L. Demanet, D. Donoho, and L. Ying, *Fast discrete curvelet transforms*, SIAM Multiscale Model. Simul. **5/3** (2006), 861–899.

[CDS98] S. S. Chen, D. L. Donoho, and M. A. Saunders, *Atomic decomposition by basis pursuit*, SIAM J. Sci. Comput. **20** (1998), 33–61.

[CDV10] P. L. Combettes, D. Dũng, and B. C. Vũ, *Dualization of signal recovery problems*, Set-Valued and Variational Analysis **18** (2010), 373–404.

[CENR11] E. J. Candès, Y. C. Eldar, D. Needell, and P. Randall, *Compressed sensing with coherent and redundant dictionaries*, Appl. Comp. Harm. Anal., *to appear* (2011).

[Cev09] V. Cevher, *Learning with compressible priors*, NIPS, 2009.

[CG02] E. J. Candès and F. Guo, *New multiscale transforms, minimum total-variation synthesis: applications to edge-preserving image reconstruction*, Signal Processing **82** (2002), no. 11, 1519–1543.

[CGM01] R. Coifman, F. Geshwind, and Y Meyer, *Noiselets*, Appl. Comput. Harmon. Anal. **10** (2001), no. 1, 27–44.

[Cha04] A. Chambolle, *An algorithm for total-variation minimization and applications*, J. Math. Imaging Vision **20** (2004), no. 1–2, 89–97.

[Cha07] R. Chartrand, *Exact reconstruction of sparse signals via nonconvex minimization*, IEEE Signal Process. Lett. **14** (2007), no. 10, 707–710.

[Cha08] V. Chandar, *A negative result concerning explicit matrices with the restricted isometry property*, Tech. report, Rice DSP website, 2008, Available at `http://dsp.rice.edu/cs`.

[Chr09] S. Chretien, *The two stage l_1 approach to the compressed sensing problem*, Tech. Report 0906.0593, arXiv, June 2009.

[Cia89] P. G. Ciarlet, *Introduction to numerical linear algebra and optimisation*, Cambridge University Press, 1989.

[CLMW09] E. J. Candès, X. Li, Y. Ma, and J. Wright, *Robust principal component analysis?*, Preprint: arXiv:0912.3599 (2009).

[CM73] J. F. Claerbout and F. Muir, *Robust modeling with erratic data*, Geophysics **38** (1973), no. 5, 826–844.

[CMA09] S. Chehrazi, A. Mirzaei, and A. Abidi, *Second-order intermodulation in current-commutating passive FET mixers*, IEEE Trans. Circ. Sys.-I **56** (2009), no. 12.

[Com04] P. L. Combettes, *Solving monotone inclusions via compositions of nonexpansive averaged operators*, Optimization **53** (2004), no. 5–6, 475–504.

[Cor09] T. J. Cornwell, *Högbom's CLEAN algorithm: impact on astronomy and beyond*, Astronomy & Astrophysics **500** (2009), no. 1, 65–66.

[COS09] J.-F. Cai, S. Osher, and Z. Shen, *Convergence of the linearized Bregman iterations for l1-norm minimization*, Math. Comp. **78** (2009), 2127–2136.

[CP07a] E. J. Candès and Y. Plan, *Near-ideal model selection by ℓ_1 minimization*, Ann. Statist. **37** (2007), no. 5A, 2145–2177.

[CP07b] P. L. Combettes and J.-C. Pesquet, *A Douglas-Rachford splitting approach to nonsmooth convex variational signal recovery*, IEEE J. Sel. Topics Signal Processing **1** (2007), no. 4, 564–574.

[CP08] P. L. Combettes and J. C. Pesquet, *A proximal decomposition method for solving convex variational inverse problems*, Inverse Problems **24** (2008), no. 6.

[CP10a] E. J. Candès and Y. Plan, *A probabilistic and RIPless theory of compressed sensing*, Preprint: arXiv:1011.3854 (2010).

[CP10b] E. J. Candès and Y. Plan, *Tight oracle bounds for low-rank matrix recovery from a minimal number of random measurements*, CoRR **abs/1001.0339** (2010).

[CP10c] A. Chambolle and T. Pock, *A first-order primal-dual algorithm for convex problems with applications to imaging*, J. Math. Imaging Vision **40** (2010), no. 1, 120–145.

[CR05] E. J. Candès and J. K. Romberg, *Practical signal recovery from random projections*, SPIE Conference on Computational Imaging, 2005, pp. 76–86.

[CR07a] E. J. Candès and J. Romberg, *Sparsity and incoherence in compressive sampling*, Inverse Prob. **23** (2007), no. 3, 969–985.

[CR07b] E. J. Candès and J. K. Romberg, ℓ_1*-magic*, Tech. report, Caltech, 2007.

[CR09] E. J. Candès and B. Recht, *Exact matrix completion via convex optimization*, Foundations of Computational Mathematics **9** (2009), no. 6, 717–772.

[CRT06] E. J. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Trans. Inform. Theory **52** (2006), no. 2, 489–509.

[CT06] E. J. Candès and T. Tao, *Near optimal signal recovery from random projections: Universal encoding strategies?*, IEEE Trans. Inform. Theory **52** (2006), no. 12, 5406–5425.

[CT07a] E. J. Candès and T. Tao, *The Dantzig selector: statistical estimation when p is much larger than n*, Ann. Statist. **35** (2007), no. 6, 2313–2351.

[CT07b] E. J. Candès and T. Tao, *Rejoinder: The Dantzig selector: Statistical estimation when p is much larger than n*, Ann. Stat. **35** (2007), no. 6, 2392–2404.

[CT10] E. J. Candès and T. Tao, *The power of convex relaxation: Near-optimal matrix completion*, IEEE Trans. Inform. Theory **56** (2010), no. 5, 2053–2080.

[CW05] P. L. Combettes and V. R. Wajs, *Signal recovery by proximal forward-backward splitting*, SIAM Multiscale Model. Simul. **4** (2005), no. 4, 1168–1200.

[CW08] E. J. Candès and M. Wakin, *An introduction to compressive sampling*, IEEE Sig. Proc. Mag. **25** (2008), no. 2, 21–30.

[CWB08] E. J. Candès, M. B. Wakin, and S. P. Boyd, *Enhancing sparsity by reweighted ℓ_1 minimization*, J. Fourier Anal. Appl. **14** (2008), no. 5–6, 877–905.

[CY08] R. Chartrand and W. Yin, *Iteratively reweighted algorithms for compressive sensing*, Proc. Int. Conf. Acoustics, Speech, Signal Processing (ICASSP), 2008, pp. 3869–3872.

[CYH+11] Xi Chen, Z. Yu, S. Hoyos, B. M. Sadler, and J. Silva-Martinez, *A sub-Nyquist rate sampling receiver exploiting compressive sensing*, IEEE Trans. Circuits and Systems I: Regular Papers **58** (2011), no. 3, 507–520.

[DB10] M. F. Duarte and R. G. Baraniuk, *Spectral compressive sensing*, IEEE Trans. Sig. Proc., *submitted.*, (2010).

[DBMS84] J. Dongarra, J. Bunch, C. Moler, and G. Stewart, *LINPACK*, netlib, 1984.

[DDFG10] I. Daubechies, R. DeVore, M. Fornasier, and C. S. Gunturk, *Iteratively re-weighted least squares minimization for sparse recovery*, Comm. Pure Appl. Math. **63** (2010), no. 1.

[DDM04] I. Daubechies, M. Defrise, and C. De Mol, *An iterative thresholding algorithm for linear inverse problems with a sparsity constraint*, Comm. Pure Appl. Math. **57** (2004), no. 11, 1413–1457.

[DDW+07] M. A. Davenport, M. F. Duarte, M. B. Wakin, J. N. Laska, D. Takhar, K. F. Kelly, and R. G. Baraniuk, *The smashed filter for compressive classification and target recognition*, Proc. Comp. Imag. V at SPIE Elect. Imag. (San Jose), 2007.

[DE03] D. L. Donoho and M. Elad, *Optimally sparse representation in general (non-orthogonal) dictionaries via ℓ^1 minimization*, Proc. Nat. Aca. Sci. **100** (2003), 2197–2202.

[DET06] D. L. Donoho, M. Elad, and V Temlyakov, *Stable recovery of sparse overcomplete representations in the presence of noise*, IEEE Trans. Inform. Theory **52** (2006), 6–18.

[DFL08] I. Daubechies, M. Fornasier, and I. Loris, *Accelerated projected gradient method for linear inverse problems with sparsity constraints*, J. Fourier Anal. Appl. **14** (2008), 764–792.

[DH93] D.-Z. Du and F. K. Hwang, *Combinatorial group testing and its applications*, World Scientific, Singapore, 1993.

[DH01] D. L. Donoho and X. Huo, *Uncertainty principles and ideal atomic decomposition*, IEEE Trans. Info. Theory **47** (2001), 2845–2862.

[DHJJ10] J. Dahl, C. Hansen, S. H. Jensen, and T. L. Jensen, *Algorithms and software for total variation image reconstruction via first-order methods*, Numer. Algorithms **53** (2010), no. 1.

[DJ94] D. L. Donoho and I. M. Johnstone, *Ideal spatial adaptation by wavelet shrinkage*, Biometrika **81** (1994), no. 3, 425–455.

[DL92] D. L. Donoho and B. F. Logan, *Signal recovery and the large sieve*, SIAM J. Appl. Math **52** (1992), no. 2, 577–592.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin, *Maximum likelihood from incomplete data via the em algorithm*, J. Roy. Statist. Soc., Ser. B **39** (1977), no. 1, 1–38.

[DMM09] D. L. Donoho, A. Maliki, and A. Montanari, *Message-passing algorithms for compressed sensing*, Proc. Nat. Acad. Sci. **106** (2009), no. 45, 18914–18919.

[DMR+09] D. L. Donoho, A. Maleki, I. Rahman, M. Shahram, and V. Stodden, *Reproducible research in computational harmonic analysis*, Computing in Sciences and Engineering **11** (2009), no. 1, 8–18.

[Don06] D. L. Donoho, *Compressed sensing*, IEEE Trans. Inform. Theory **52** (April 2006), no. 4, 1289–1306.

[DS89] D. L. Donoho and P. B. Stark, *Uncertainty principles and signal recovery*, SIAM J. Appl. Math **49** (1989), no. 3, 906–931.

[DS05] J. Darbon and M. Sigelle, *A fast and exact algorithm for total-variation minimization*, IbPRIA **3522** (2005), 351–359.

[DT08] D. L. Donoho and Y. Tsaig, *Fast solution of ℓ_1 minimization problems when the solution may be sparse*, IEEE Trans. Inform. Theory **54** (2008), no. 11, 4789–4812.

[DT10] D. L. Donoho and J. Tanner, *Precise undersampling theorems*, Proc. IEEE **98** (2010), no. 6, 913–924.

[DTDS06] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, *Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit*, IEEE Trans. Inform. Theory (2006), submitted.

[DV10] J. Dahl and L. Vandenberghe, *CVXOPT, Python software for convex optimization, version 1.13*, UCLA, 2010, Available at `http://abel.ee.ucla.edu/cvxopt`.

[DW10] M. A. Davenport and M. B. Wakin, *Analysis of orthogonal matching pursuit using the restricted isometry property*, IEEE Trans. Info. Theory **56** (2010), no. 9, 4395–4401.

[DWB08] M. F. Duarte, M. B. Wakin, and R. G. Baraniuk, *Wavelet-domain compressive signal reconstruction using a hidden Markov tree model*, ICASSP, 2008.

[EB02] M. Elad and A. M. Bruckstein, *A generalized uncertainty principle and sparse representation in pairs of bases*, IEEE Trans. Info. Theory **48** (2002), 2558–2567.

[EHJT04] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, *Least angle regression*, Ann. Statist. **32** (2004), no. 2, 407–499.

[EHT07] B. Efron, T. Hastie, and R. Tibshirani, *Discussion: The Dantzig selector: Statistical estimation when p is much larger than n*, Ann. Stat. **35** (2007), no. 6, 2358–2364.

[Eld09] Y. C. Eldar, *Compressed sensing of analog signals in shift-invariant spaces*, IEEE Trans. Sig. Proc. **57** (2009), no. 8, 2986–2997.

[Ell10] J. Ellenberg, *Fill in the blanks: Using math to turn lo-res datasets into hi-res samples*, Wired Magazine (2010), `http://www.wired.com/magazine/2010/02/ff_algorithm/`.

[EMR07] M. Elad, P. Milanfar, and R. Rubinstein, *Analysis versus synthesis in signal priors*, Inverse Problems **23** (2007), 947–968.

[EMZ07] M. Elad, B. Matalon, and M. Zibulevsky, *Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization*, Appl. Comput. Harmon. Anal **23** (2007), 346–367.

[ERW11] A. Eftekhari, J. Romberg, and M. B. Wakin, *Matched filtering from limited frequency samples*, Tech. Report 1101.2713, arXiv, 2011.

[Faz02] M. Fazel, *Matrix rank minimization with applications*, Ph.D. thesis, Stanford University, 2002.

[FBC+08] G. L. Fudge, R. E. Bland, M. A. Chivers, S. Ravindran, J. Haupt, and P. E. Pace, *A Nyquist folding analog-to-information receiver*, Signals, Systems and Computers, 2008 42nd Asilomar Conference on (Pacific Grove, CA), Oct. 2008, pp. 541–545.

[FHT10] J. Friedman, T. Hastie, and R. Tibshirani, *Regularization paths for generalized linear models via coordinate descent*, J. Stat. Software **33** (2010), no. 1.

[FM81] M. Fukushima and H. Mine, *A generalized proximal point algorithm for certain non-convex minimization problems*, Int. J. Systems Science **12** (1981), no. 8, 989–1000.

[FN03] M. A. T. Figueiredo and R.D. Nowak, *An EM algorithm for wavelet-based image restoration*, IEEE Trans. Image Process. **12** (2003), no. 8, 906–916.

[FNW07] M. A. T. Figueiredo, R. Nowak, and S. J. Wright, *Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems*, IEEE J. Sel. Top. Signal Process. **1** (2007), no. 4, 586–597.

[FP11] J.M. Fadili and G. Peyré, *Total variation projection with first order schemes*, IEEE Trans. Image Proc. **20** (2011), no. 3, 657–669.

[FR11] M. Fornasier and H. Rauhut, *Handbook of mathematical methods in imaging*, ch. Compressive Sensing, Springer, 2011.

[FS09] M.J. Fadili and J-L. Starck, *Monotone operator splitting for optimization problems in sparse recovery*, IEEE ICIP (Cairo), 2009.

[FT07] M. P. Friedlander and P. Tseng, *Exact regularization of convex programs*, SIAM J. Optim. **18** (2007), no. 4, 1326–1350.

[G̈92] O. Güler, *New proximal point algorithms for convex minimization*, SIAM J. Optim. **2** (1992), no. 4, 649–664.

[GB10] M. Grant and S. Boyd, *CVX: Matlab software for disciplined convex programming, version 1.21*, 2010, Available at `http://cvxr.com/cvx`.

[GG03] J. Gondzio and A. Grothey, *Reoptimization with the primal-dual interior point method*, SIAM J. Opt. **13** (2003), 842–864.

[GGB95] I. Gorodnitsky, J. S. George, and D. R. Bhaskar, *Neuromagnetic source imaging with focuss: a recursive weighted minimum norm algorithm*, Electroencephalography and Clinical Neurophysiology **95** (1995), 231–251.

[GLF+10] D. Gross, Y.-K. Liu, S. T. Flammia, S. Becker, and J. Eisert, *Quantum state tomography via compressed sensing*, Phys. Rev. Lett. **105** (2010), no. 15, 150401.

[GLW09] M. Gu, L.-H. Lim, and C. J. Wu, *PARNES: A rapidly convergent algorithm for accurate recovery of sparse and approximately sparse signals*, Tech. report, U.C. Berkeley, 2009, Preprint available at `http://arxiv.org/abs/0911.0492`.

[GM10] D. Goldfarb and S. Ma, *Fast multiple splitting algorithms for convex optimization*, SIAM J. Optimization, *under revision* (2010).

[GMS10] D. Goldfarb, S. Ma, and K. Scheinberg, *Fast alternating linearization methods for minimizing the sum of two convex functions*, Math. Prog. (Series A), *submitted* (2010).

[GN03] R. Gribonval and M. Nielsen, *Sparse representations in unions of bases*, IEEE Trans. Inform. Theory **49** (2003), 3320–3325.

[GO09] T. Goldstein and S. Osher, *The split Bregman method for l1 regularized problems*, SIAM J. Imaging Sci. **2** (2009), no. 2, 323–343.

[Gro11] D. Gross, *Recovering low-rank matrices from few coefficients in any basis*, IEEE Trans. Inform. Theory **57** (2011), no. 3.

[GS06a] A. C. Gilbert and M. J. Strauss, *Group testing in statistical signal recovery*, 2006, Available at `http://www.math.lsa.umich.edu/~annacg/papers/GS06.grouptesting.pdf`.

[GS06b] C. Greif and D. Schötzau, *Preconditioners for saddle point linear systems with highly singular (1,1) blocks*, Elec. Trans. Numer. Anal. **22** (2006), 114–121.

[GSS02] Y. Geerts, M. Steyaert, and W. M. C. Sansen, *Design of multi-bit delta-sigma a/d converters*, Springer, 2002.

[GST08] A. C. Gilbert, M. J. Strauss, and J. A. Tropp, *A tutorial on fast Fourier sampling*, IEEE Sig. Proc. Mag. (2008), 57–66.

[GSTV06] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin, *Algorithmic linear dimension reduction in the l1 norm for sparse vectors*, Proc. 44th Allerton Conf. Communication, Control, and Computing (Allerton, IL), 2006, Invited paper, special session: Compressed Sensing.

[GSTV07] A. Gilbert, M. Strauss, J. A. Tropp, and R. Vershynin, *One sketch for all: Fast algorithms for compressed sensing*, Proc. 39th ACM Symp. Theory of Computing (San Diego), 2007.

[HBC11] A. Harmsy, W. U. Bajwa, and R. Calderbank, *Beating Nyquist through correlations: a constrained random demodulator for sampling of sparse bandlimited signals*, ICASSP (Prague), 2011.

[Hea07] D. L. Healy, *Analog-to-Information Receiver Development Program (A-to-I)*, 2007, DARPA Broad Agency Announcement (BAA) No. 08-03. Available at `https://www.fbo.gov/spg/ODA/DARPA/CMO/BAA08-03/listing.html`.

[HH08] G. Hennenfent and F. J. Herrmann, *Simply denoise: wavefield reconstruction via jittered undersampling*, Geophysics **73** (2008).

[HL04] D. R. Hunter and K. Lange, *A tutorial on mm algorithms*, The American Statistician (2004), 30–37.

[HMT11] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions*, SIAM Rev. (2011), to appear.

[Hög74] J. A. Högbom, *Aperture synthesis with a non regular distribution of interferometer baselines*, Astronomy & Astrophysics Supp. **15** (1974), 417–426.

[HS09] M. Herman and T. Strohmer, *High-resolution radar via compressed sensing*, IEEE Trans. Sig. Proc. **57** (2009), no. 6, 2275–2284.

[Hub64] P. J. Huber, *Robust estimation of a location parameter*, Annal. Stat. **53** (1964), 73–101.

[HUL93] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex analysis and minimization algorithms*, vol. I and II, Springer-Verlag, 1993.

[HYZ07] E. T. Hale, W. Yin, and Y. Zhang, *A fixed-point continuation method for l1-regularized minimization with applications to compressed sensing*, Tech. Report CAAM/TR07-07, Rice University, Houston, TX, 2007.

[HYZ08] E. T. Hale, W. Yin, and Y. Zhang, *Fixed-point continuation for ℓ_1-minimization: Methodology and convergence*, SIAM J. Optim. **19** (2008), no. 3, 1107–1130.

[HZ06] W. W. Hager and H. Zhang, *Algorithm 851: CG_DESCENT, a conjugate gradient method with guaranteed descent*, ACM Trans. Math. Software **32** (2006), 113–137.

[JMOB10] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach, *Proximal methods for hierarchical sparse coding*, Preprint: arXiv:1009.2139 (2010).

[JRL09] G. James, P. Radchenko, and J. Lv, *DASSO: Connections Between the Dantzig Selector and Lasso*, J. Roy. Statist. Soc., Ser. B **71** (2009), 127–142.

[JXHC09] S. Jafarpour, W. Xu, B. Hassibi, and R. Calderbank, *Efficient and robust compressed sensing using optimized expander graphs*, IEEE Trans. Info. Theory **55** (2009), no. 9, 4299–4308.

[KKB07] K. Koh, S. J. Kim, and S. P. Boyd, *Solver for l1-regularized least squares problems*, Tech. report, Stanford University, 2007, `http://www.stanford.edu/~boyd/l1_ls/`.

[KLW+06] S. Kirolos, J. N. Laska, M. Wakin, M. F. Duarte, D. Baron, T. Ragheb, Y. Massoud, and R. G. Baraniuk, *Analog-to-information conversion via random demodulation*, Proc. IEEE DCAS, Sept. 2006.

[Kor76] G. M. Korpelevich, *Extrapolation gradient methods and their relation to modified Lagrange functions*, Ekonom. i Mat. Metody **19** (1976), no. 4.

[KRL+06] S. Kirolos, T. Ragheb, J. Laska, M. Duarte, Y. Massoud, and R. Baraniuk, *Practical issues in implementing analog-to-information converters*, IEEE 6th Int. Workshop on System on Chip for Real Time Applications, Sept. 2006.

[KXAH10] A. Khajehnejad, W. Xu, S. Avestimehr, and Babak Hassibi, *Improved sparse recovery thresholds with two step reweighted l1 minimization*, ISIT (Austin), June 2010.

[Lan67] H. J. Landau, *Necessary density conditions for sampling and interpolation of certain entire functions*, Acta Math. **117** (1967), 37–52.

[Lar04] R. M. Larsen, *PROPACK: Software for large and sparse SVD calculations*, Stanford, March 2004.

[LBDM+09] I. Loris, M. Bertero, C. De Mol, R. Zanella, and L. Zanni, *Accelerating gradient projection methods for ℓ_1-constrained signal recovery by steplength selection*, Appl. Comput. Harmon. Anal. **27** (2009), no. 2, 247–254.

[LDP07] M. Lustig, D. L. Donoho, and J. M. Pauly, *Sparse MRI: The application of compressed sensing for rapid MR imaging*, Magn. Reson. Med. **58** (2007), no. 6, 1182–1195.

[LH07] T. Lin and F. J. Herrmann, *Compressed wavefield extrapolation*, Geophysics **72** (2007), no. 5.

[LKD+07] J. N. Laska, S. Kirolos, M. F. Duarte, T. Ragheb, R. G. Baraniuk, and Y. Massoud, *Theory and implementation of an analog-to-information converter using random demodulation*, Proc. IEEE ISCAS, New Orleans, May 2007, pp. 1959–1962.

[LLM09] G. Lan, Z. Lu, and R. D. C. Monteiro, *Primal-dual first-order methods with $o(1/\epsilon)$ iteration-complexity for cone programming*, Math. Program. (2009).

[LO09] A. S. Lewis and M. L. Overton, *Nonsmooth optimization via BFGS*, Tech. Report 2008/12/2172, Optimization Online, 2009.

[Log65] B. F. Logan, *Properties of high-pass signals*, Ph.D. thesis, Columbia Univ., New York, 1965.

[Lor09] I. Loris, *On the performance of algorithms for the minimization of ℓ_1-penalized functionals*, Inverse Problems **25** (2009).

[Lor11] D. Lorenz, *Constructing test instances for basis pursuit denoising*, Tech. Report 1103.2897, arXiv, 2011.

[LRRB05] B. Le, T. W. Rondeau, J. H. Reed, and C. W. Bostian, *Analog-to-digital converters*, IEEE Sig. Proc. Mag. (2005), 69–77.

[LST11] Y.-J. Liu, D. Sun, and K.-C. Toh, *An implementable proximal point algorithmic framework for nuclear norm minimization*, Mathematical Programming (2011).

[Lu09] Z. Lu, *Primal-dual first-order methods for a class of cone programming*, *Submitted to* INFORMS Journal on Computing (2009), Preprint available at `http://www.math.sfu.ca/~zhaosong/ResearchPapers/pdfirst_DS_2ndrev.pdf`.

[LW10] Z. Lin and S. Wei, *A block Lanczos with warm start technique for accelerating nuclear norm minimization algorithms*, Tech. Report MSR-TR-2010-162, Microsoft, 2010.

[Mag07] MIT Tech. Rev. Magazine, *Ten emerging technologies 2007*, `http://www.technologyreview.com/specialreports/specialreport.aspx?id=19`, 2007.

[Mal08] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, Academic Press, 2008.

[Man81] O. L. Mangasarian, *Iterative solution of linear programs*, SIAM J. Numer. Anal. **18** (1981), no. 4, 606–614.

[McE02] R. McEliece, *The theory of information and coding*, Cambridge University Press, 2002.

[ME82] R. Mammone and G. Eichmann, *Restoration of discrete Fourier spectra using linear programming*, J. Opt. Soc. Am. **72** (1982), 987–992.

[ME09a] M. Mishali and Y. C. Eldar, *Blind multiband signal reconstruction: Compressed sensing for analog signals*, IEEE Trans. Signal Process. **57** (2009), no. 3, 993–1009.

[ME09b] ———, *Expected RIP: Conditioning of the modulated wideband converter*, IEEE Info. Theory Workshop, October 2009, pp. 343–347.

[ME10] ———, *From theory to practice: Sub-Nyquist sampling of sparse wideband analog signals*, IEEE J. Sel. Top. Signal Process. **4** (2010), no. 2, 375–391.

[MEDS11] M. Mishali, Y. C. Eldar, O. Dounaevsky, and E. Shoshan, *Xampling: Analog to digital at sub-Nyquist rates*, IET Cir. Dev. and Systems **5** (2011), no. 1, 8–20.

[MEE09] M. Mishali, Y. C. Eldar, and A. Elron, *Xampling–part i: Practice*, Tech. Report 0911.0519, arXiv, 2009.

[MGC09] S. Ma, D. Goldfarb, and L. Chen, *Fixed point and Bregman iterative methods for matrix rank minimization*, Math. Program., Ser A. (2009).

[Mil02] A. Miller, *Subset selection in regression*, 2 ed., Chapman and Hall/CRC, April 2002.

[MM79] O. L. Mangasarian and R. R. Meyer, *Nonlinear perturbation of linear programs*, SIAM J. Control Optim. **17** (1979), 745–752.

[Mor62] J.-J. Moreau, *Fonctions convexes duales et points proximaux dans un espace hilbertien*, R. Acad. Sci. Paris Sér. A Math. **255** (1962), 2897–2899.

[Mor65] ———, *Proximité et dualité dans un espace hilbertien*, Bull. Soc. Math. France **93** (1965), 273–299.

[Mos02] Mosek ApS, *The MOSEK optimization tools version 2.5*, 2002, `http://www.mosek.com`.

[MPAL11] T. Murray, P. Pouliquen, A. G. Andreou, and K. Lauritzen, *Design of a CMOS A2I data converter: Theory, architecture and implementation*, Conference on Information Sciences and Systems (Baltimore), March 2011.

[MSS+10] M. Mohtashemi, H. Smith, F. Sutton, D. Walburger, and J. Diggans, *Sparse sensing dna microarray-based biosensor: Is it feasible?*, IEEE Sensor. Appl., 2010.

[MT10] M. McCoy and J. A. Tropp, *Two proposals for robust pca using semidefinite programming*, Tech. Report 1012.1086, arXiv, Dec. 2010.

[MZ93] S. Mallat and Z. Zhang, *Matching pursuits with time-frequency dictionaries*, IEEE Trans. Sig. Proc. **41** (1993), 3397–3415.

[MZ09] F. Malgouyres and T. Zeng, *A predual proximal point algorithm solving a non negative basis pursuit denoising model*, Int. J. Comp. Vision **83** (2009), no. 3, 294–311.

[Nas08] G. P. Nason, *Wavelet methods in statistics with R*, Springer, 2008.

[Nee09] D. Needell, *Noisy signal recovery via iterative reweighted l1-minimization*, Proc. Asilomar Conf. Signals, Systems, and Computers, 2009.

[Nes83] Y. Nesterov, *A method for unconstrained convex minimization problem with the rate of convergence* $\mathcal{O}(1/k^2)$, Doklady AN SSSR, *translated as* Soviet Math. Docl. **269** (1983), 543–547.

[Nes88] Y. Nesterov, *On an approach to the construction of optimal methods of minimization of smooth convex functions*, Ekonomika i Mateaticheskie Metody **24** (1988), 509–517, (Russian).

[Nes04] ———, *Introductory lectures on convex optimization: A basic course*, Applied Optimization, vol. 87, Kluwer, Boston, 2004.

[Nes05] Y. Nesterov, *Smooth minimization of non-smooth functions*, Math. Program., Ser. A **103** (2005), 127–152.

[Nes07] ———, *Gradient methods for minimizing composite objective function*, Tech. report, CORE—Universite Catholique de Louvain, Louvain-la-Neuve, Belgium, 2007.

[NN94] Y. Nesterov and A. Nemirovsky, *Interior point polynomial algorithms in convex programming*, SIAM, Philadelphia, 1994.

[NT09] D. Needell and J. A. Tropp, *CoSaMP: Iterative signal recovery from incomplete and inaccurate samples*, Appl. Comput. Harmon. Anal **26** (2009), 301–321.

[NV09] D. Needell and R. Vershynin, *Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit*, Found. Comp. Math. **9** (2009), no. 3, 371–334.

[NV10] ———, *Signal recovery from inaccurate and incomplete measurements via regularized orthogonal matching pursuit*, IEEE J. Sel. Topics Sig. Proc. **4** (2010), 310–316.

[NW06] J. Nocedal and S. Wright, *Numerical optimization*, 2 ed., Springer, 2006.

[NY83] A. Nemirovski and D. Yudin, *Problem complexity and method efficiency in optimization*, Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons Ltd., 1983.

[OMDY10] S. Osher, Y. Mao, B. Dong, and W. Yin, *Fast linearized Bregman iteration for compressive sensing and sparse denoising*, Commun. Math. Sci. **8** (2010), no. 1, 93–111.

[OPT00] M. R. Osborne, B. Presnell, and B. A. Turlach, *A new approach to variable selection in least squares problems*, IMA J. Numer. Anal. **20** (2000), no. 3, 389–403.

[OSK94] M. S. O'Brien, A. N. Sinclair, and S. M. Kramer, *Recovery of a sparse spike time series by ℓ_1 norm deconvolution*, IEEE Trans. Sig. Proc. **44** (1994), 3353–3365.

[PC79] A. Papoulis and C. Chamzas, *Improvement of range resolution by spectral extrapolation*, Ultrasonic Imaging **1** (1979), no. 2, 121–135.

[Pop09] G. Pope, *Compressive sensing: A summary of reconstruction algorithms*, Master's thesis, ETH, Zürich, Feb. 2009.

[Pow77] M. J. D. Powell, *Restart procedures for the conjugate gradient method*, Math. Program. **12** (1977), 241–254.

[Pro01] J. G. Proakis, *Digital communications*, vol. 4, McGraw Hill, 2001.

[PSM82] R. L. Pickholtz, D. L. Schilling, and L B. Milstein, *Theory of spread-spectrum communications—a tutorial*, IEEE Trans. on Comm. **30** (May 1982), no. 5.

[PT74] B. T. Poljak and N. V. Tretjakov, *An iterative method for linear programming and its economic interpretation*, Matecon **10** (1974), 81–100.

[Ran09] P. A. Randall, *Sparse recovery via convex optimization*, Ph.D. thesis, California Institute of Technology, May 2009.

[Raz97] Behzad Razavi, *Rf microelectronics*, Prentice Hall, 1997.

[RG07] T. Rees and C. Greif, *A preconditioner for linear systems arising from interior point optimization methods*, SIAM J. Sci. Comput. **29** (2007), no. 5.

[RLN+08] T. Ragheb, J. Laska, H. Nejati, S. Kirolos, R. Baraniuk, and Y. Massoud, *A prototype hardware for random demodulation based compressive analog-to-digital conversion*, 51st Midwest Symp. on Circuits and Systems, MWSCAS, 2008, pp. 37–40.

[Roc70] R. T. Rockafellar, *Convex analysis*, Princeton Landmarks in Mathematics and Physics, Princeton University Press, Princeton, NJ, 1970.

[Roc76] ———, *Monotone operators and the proximal point algorithm*, SIAM Journal on Control and Optimization **14** (1976), 877–898.

[ROF92] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation noise removal algorithm*, Physica D **60** (1992), 259–268.

[Rom08] J. K. Romberg, *The Dantzig selector and generalized thresholding*, Proc. IEEE Conf. Info. Sci. and Sys. (Princeton, New Jersey), February 2008.

[Rom09] J. Romberg, *Compressive sensing by random convolution*, SIAM J. Imag. Sci. **2** (2009), no. 4, 1098–1128.

[RU] DSP Rice U., *Compressed sensing resources.*, http://www.dsp.ece.rice.edu/cs/.

[SB10] J.-L. Starck and J. Bobin, *Astronomical data analysis and sparsity: From wavelets to compressed sensing*, Proc. IEEE **98** (2010), no. 6, 1021–1030.

[SBB06] S. Sarvotham, D. Baron, and R. Baraniuk, *Fast measurement and reconstruction of sparse signals*, IEEE Int. Symp. Info. Theory (Seattle), July 2006.

[Sch78] U. J. Schwarz, *Mathematical-statistical description of the iterative beam removing technique (method CLEAN)*, Astronomy & Astrophysics (1978), 345.

[SED04] J.-L. Starck, M. Elad, and D. L. Donoho, *Redundant multiscale transforms and their application for morphological component analysis*, Advances in Imaging and Electron Physics **132** (2004), 287–348.

[SED05] J.-L Starck, M. Elad, and D. L. Donoho, *Image decomposition via the combination of sparse representation and a variational approach*, IEEE Trans. Image Process. **14** (2005), no. 10, 1570–1582.

[SK02] M.A. Saunders and B. Kim, *PDCO: Primal-dual interior method for convex objectives*, Tech. report, Stanford University, 2002, Available at `http://www.stanford.edu/group/SOL/software/pdco.html`.

[SNM03] J.-L. Starck, M. K. Ngyuen, and F. Murtagh, *Wavelets and curvelets for image deconvolution: a combined approach*, Signal Processing **83** (2003), 2279–2283.

[SOS+05] W. Skones, B. Oyama, S. Stearns, J. Romberg, and E. Candès, *Analog to information (A-to-I), technical and management proposal*, August 2005, (in response to DARPA BAA 05-35).

[SS86] F. Santosa and W. W. Symes, *Linear inversion of band-limited reflection seismograms*, SIAM J. Sci. Stat. Comput **7** (1986), no. 4, 1307–1330.

[Tao] T. Tao, *Preprints in sparse recovery/compressed sensing.*, `http://www.math.ucla.edu/~tao/preprints/sparse.html`.

[Tao05] ———, *An uncertainty principle for cyclic groups of prime order*, Math. Res. Letters **12** (2005), 121–127.

[Tao11] ———, *Topics in random matrix theory, in preparation*, (2011).

[TBM79] H. L. Taylor, S. C. Banks, and J. F. McCoy, *Deconvolution with the ℓ_1 norm*, Geophysics **44** (1979), no. 1, 39–52.

[TG07] J. A. Tropp and A. C. Gilbert, *Signal recovery from random measurements via orthogonal matching pursuit*, IEEE Tran. Info. Theory **53** (2007), no. 12.

[The08] The CMS Collaboration et al., *The CMS experiment at the CERN LHC*, J. Instrumentation **3** (2008).

[Tho82] D. J. Thomson, *Spectrum estimation and harmonic analysis*, Proc. IEEE **70** (1982), no. 9, 1055–1094.

[Tib96] R. Tibshirani, *Regression shrinkage and selection via the lasso*, J. Roy. Statist. Soc., Ser. B **58** (1996), no. 1, 267–288.

[TLD+10] J. A. Tropp, J. N. Laska, M. F. Duarte, J. Romberg, and R. G. Baraniuk, *Beyond Nyquist: Efficient sampling of sparse bandlimited signals*, IEEE Trans. Inform. Theory **56** (2010), no. 1, 520–544.

[Tro04] J. A. Tropp, *Greed is good: Algorithmic results for sparse approximation*, IEEE Trans. Inform. Theory **50** (2004), 2231–2242.

[Tro06] ———, *Just relax: Convex programming methods for identifying sparse signals*, IEEE Trans. Inform. Theory **51** (2006), no. 3, 1030–1051.

[Tse08] P. Tseng, *On accelerated proximal gradient methods for convex-concave optimization*, SIAM J. Optim. (2008).

[TV11] O. Taheri and S. A. Vorobyov, *Segmented compressed sampling for analog-to-information conversion: Method and performance analysis*, IEEE Tran. Sig. Proc. **59** (2011), no. 2.

[TW10] J. A. Tropp and S. J. Wright, *Computational methods for sparse solution of linear inverse problems*, Proc. IEEE **98** (2010), no. 6, 948–958.

[TWD+06] J. A. Tropp, M. Wakin, M. Duarte, D. Baron, and R. Baraniuk, *Random filters for compressive sampling and reconstruction*, Proc. 2006 IEEE Intl. Conf. Acoustics, Speech, and Signal Processing (ICASSP) (Toulouse, France), vol. III, May 2006, pp. 872–875.

[VB96] L. Vandenberghe and S. Boyd, *Semidefinite programming*, SIAM Review **38** (1996), 49–95.

[vdBF08] E. van den Berg and M. P. Friedlander, *Probing the Pareto frontier for basis pursuit solutions*, SIAM J. Sci. Comput. **31** (2008), no. 2, 890–912.

[Wal99] H. Walden, *Analog-to-digital converter survey and analysis*, IEEE J. Sel. Area Comm. **17** (1999), no. 4, 539–550.

[WBFA09] P. Weiss, L. Blanc-Féraud, and G. Aubert, *Efficient schemes for total variation minimization under constraints in image processing*, SIAM Journal on Scientific Computing **31** (2009), 2047–2080.

[WCW+09] Y. Wu, C. Chen, Z. Wang, P. Ye, G. R. Arce, D. Prather, and G. J. Schneider, *Fabrication and characterization of a compressive-sampling multispectral imaging system*, Opt. Eng. **48** (2009), no. 12.

[WG11] D. Weller and V. Goyal, *On the estimation of nonrandom signal coefficients from jittered samples*, IEEE Tran. Sign. Proc. **59** (2011), no. 2, 587–597.

[WGB07] R. Willett, M. Gehm, and D. Brady, *Multiscale reconstruction for computational spectral imaging*, Comp. Imag. V at SPIE Elect. Imag. (San Jose), 2007.

[WJP+09] Y. Wiaux, L. Jacques, G. Puy, A. M. M. Scaife, and P. Vandergheynst, *Compressed sensing imaging techniques for radio interferometry*, Mon. Not. R. Astron. Soc. **395** (2009), 1733–1742.

[WLD+06] M. Wakin, J. Laska, M. Duarte, D. Baron, S. Sarvotham, D. Takhar, K. Kelly, and R. Baraniuk, *An architecture for compressive imaging*, Proc. Intl. Conf. Image Proc. (Atlanta), 2006.

[WNF09] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo, *Sparse reconstruction by separable approximation*, IEEE Trans. Signal Process. **57** (2009), no. 7, 2479–2493.

[WR04] D. Wipf and B. Rao, *Sparse bayesian learning for basis selection*, IEEE Trans. Sig. Proc. **52** (2004), no. 8, 2153–2164.

[Wri97] S. J. Wright, *Primal-dual interior-point methods*, SIAM, 1997.

[Wri07] S. J. Wright, *Solving ℓ_1-regularized regression problems*, International Conference "Combinatorics and Optimization", Waterloo, 2007.

[WXT10] M. Wang, W. Xu, and A. K. Tang, *On the performance of sparse recovery via ℓ_p-minimization ($0 \leq p \leq 1$), or, Is ℓ_p- minimization ($0 \leq p < 1$) always better than ℓ_1-minimization in sparse recovery?*, 2010, Preprint available at `http://networks.ece.cornell.edu/~xu/pub/lp10.pdf`.

[WY10] Y. Wang and W. Yin, *Sparse signal reconstruction via iterative support detection*, SIAM J. Imag. Sci. **3** (2010), no. 3, 462–491.

[WYGZ10] Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang, *A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization and continuation*, SIAM J. Sci. Comput. **32** (2010), no. 4, 1832–1857.

[XH07] W. Xu and B. Hassibi, *Efficient compressive sensing with deterministic guarantees using expander graphs*, Proc. IEEE Info. Theory Workshop (Lake Tahoe), 2007.

[XH11] ———, *Compressive sensing over the grassmann manifold: a unified geometric framework*, IEEE Trans. Info. Theory, *to appear* (2011).

[YHS08] Z. Yu, S. Hoyos, and B. M. Sadler, *Mixed-signal parallel compressed sensing and reception for cognitive radio*, ICASSP, 2008, pp. 3861–3864.

[Yin10] W. Yin, *Analysis and generalizations of the linearized Bregman method*, SIAM J. Imaging Sci. **3** (2010), no. 4, 856–877.

[YOGD08] W. Yin, S. Osher, D. Goldfarb, and J. Darbon, *Bregman iterative algorithms for ℓ_1 minimization with applications to compressed sensing*, SIAM J. Imaging Sci. **1** (2008), no. 1, 143–168.

[YVGS10] J. Yu, S. V. N. Vishwanathan, S. Günter, and N. N. Schraudolph, *A quasi-Newton approach to nonsmooth convex optimization problems in machine learning*, J. Machine Learning Research **11** (2010), 1145–1200.

[YW02] E. A. Yildirim and S. J. Wright, *Warm-start strategies in interior-point methods for linear programming*, SIAM J. Opt. **12** (2002), 782–810.

[YZY10] J. Yang, Y. Zhang, and W. Yin, *A fast alternating direction method for TV ℓ_1-ℓ_2 signal reconstruction from partial Fourier data*, IEEE J. Sel. Top. Signal Process. **4** (2010), no. 2, 288–297.

[ZBLN97] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, *Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization*, ACM Trans. Math. Software **23** (1997), no. 4.

[ZC08] M. Zhu and T. Chan, *An efficient primal-dual hybrid gradient algorithm for total variation image restoration*, Tech. Report CAM08-34, UCLA, 2008, Available at `ftp://ftp.math.ucla.edu/pub/camreport/cam08-34.pdf`.

[Zou06] H. Zou, *The adaptive lasso and its oracle properties*, J. Amer. Stat. Assoc. **101** (2006), 1418–1429.