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.

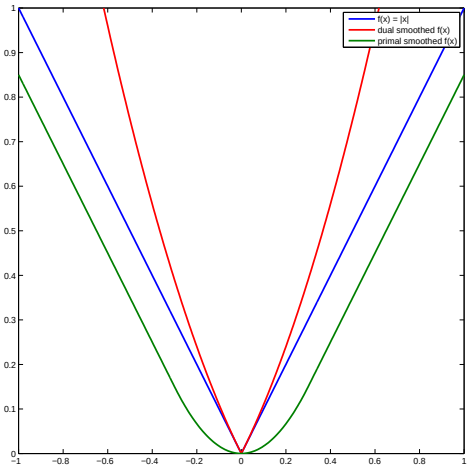

Figure 4.1: The original objective $f(x) = |x|$ (blue), our modification (red), and Nesterov's smoothing (green)

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

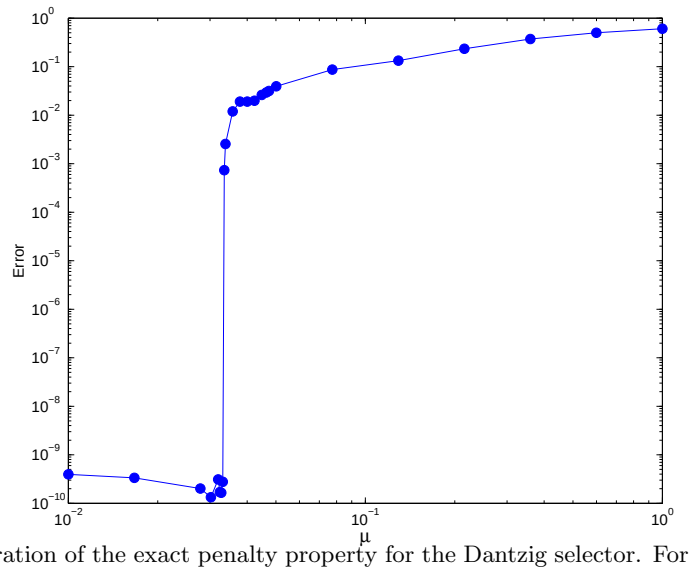

Figure 4.2: Demonstration of the exact penalty property for the Dantzig selector. For $\mu \leq \mu_0 \approx 0.025$, the solution to the smoothed model coincides with the original model to solver precision.

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.

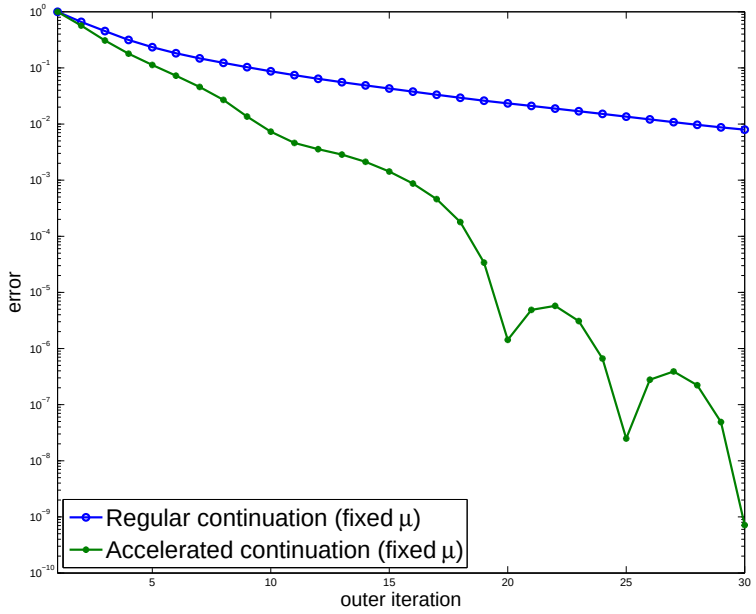

Figure 4.3: A comparison of simple continuation and accelerated continuation applied to a small-scale LASSO model. The horizontal axis gives the number of continuation steps taken; the vertical axis gives the error between the continuation solution and the original, unsmoothed model. The error is normalized 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)

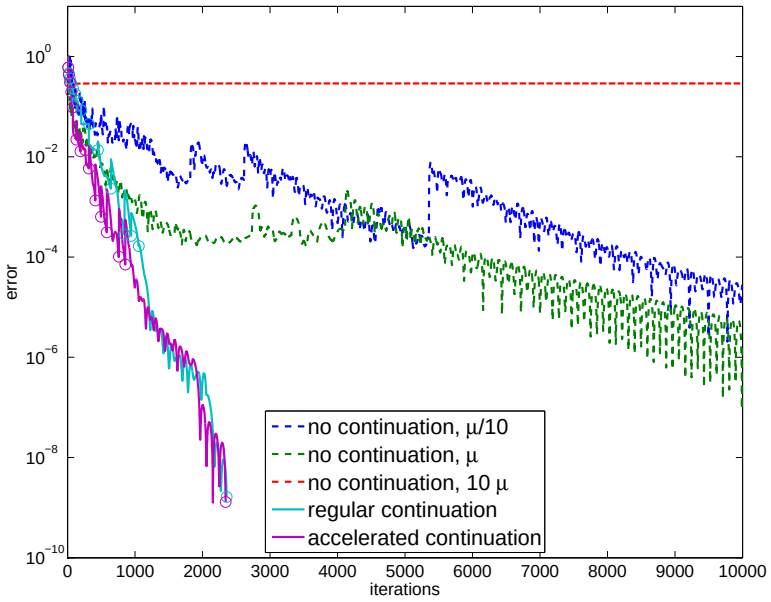

Figure 4.4: Comparing fixed smoothing and continuation strategies for a Dantzig selector model. The horizontal axis gives the number of inner iterations of each first-order method. For the two continuation strategies, the circles depict the completion of each continuation step.

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

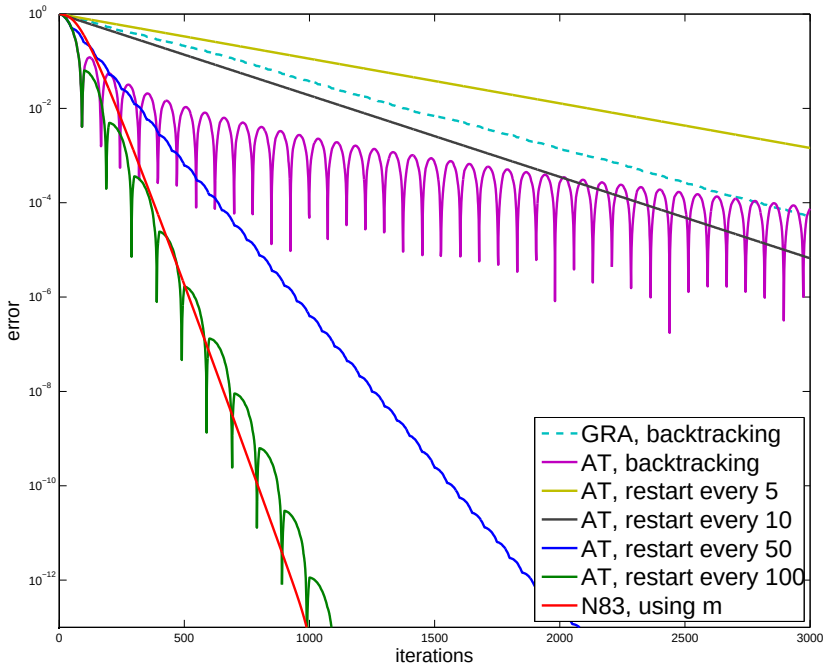

Figure 4.5: First-order methods applied to a strongly convex problem: GRA, AT without restart, AT with restart, and N83 tuned with knowledge of the strong convexity parameter. The y-axis is $\|x_k - x^\star\|_2/\|x^\star\|_2$.

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

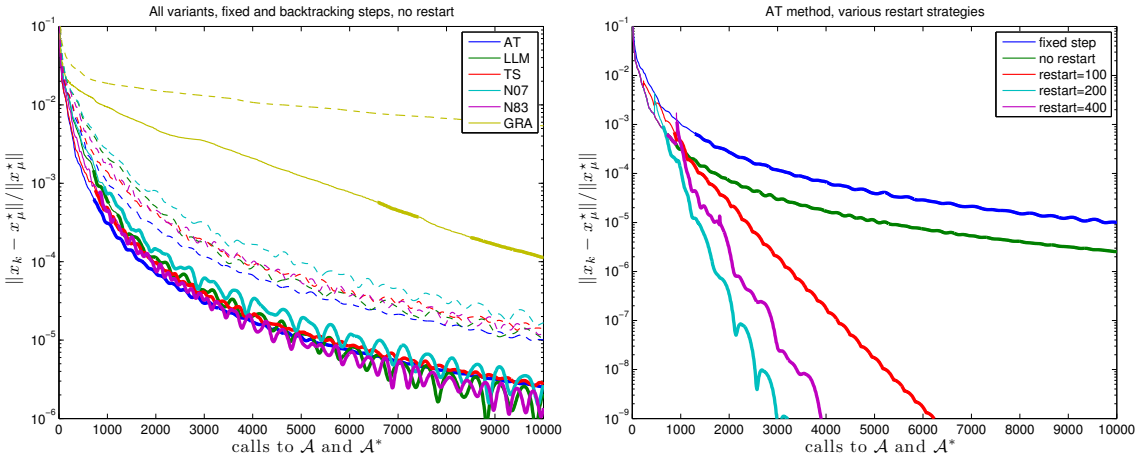

Figure 4.6: Comparing first-order methods applied to a smoothed Dantzig selector model. Left: comparing all variants using a fixed step size (dashed lines) and backtracking line search (solid lines). Right: comparing various restart strategies using the AT method

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

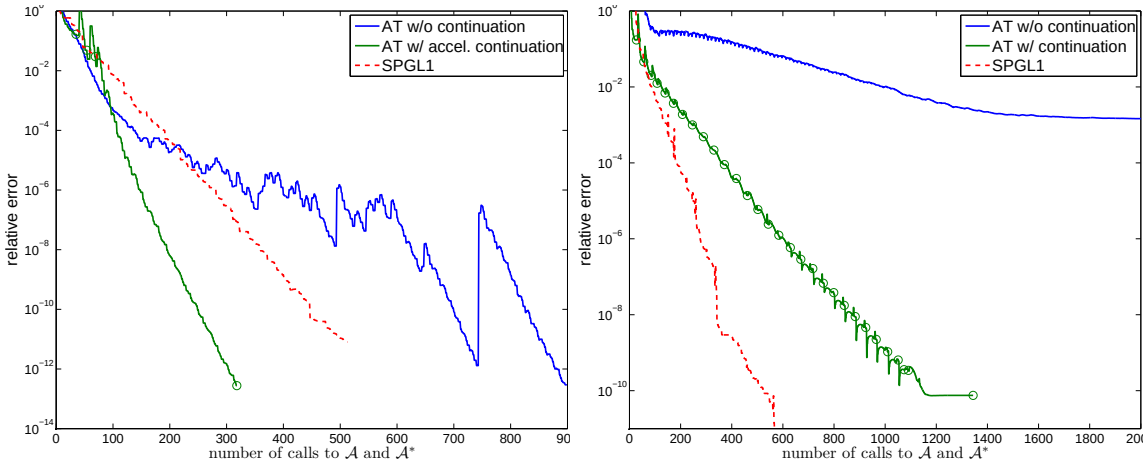

Figure 4.7: Comparisons of the dual solver with SPGL1. The plot on the left involves a noiseless basis pursuit model, while the plot on the right represents a noisy image model.

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

$$X_k = \text{SoftThresholdSingVal}\big(X_0 - \mu^{-1}\mathcal{A}^*(\lambda), \mu^{-1}\big).$$

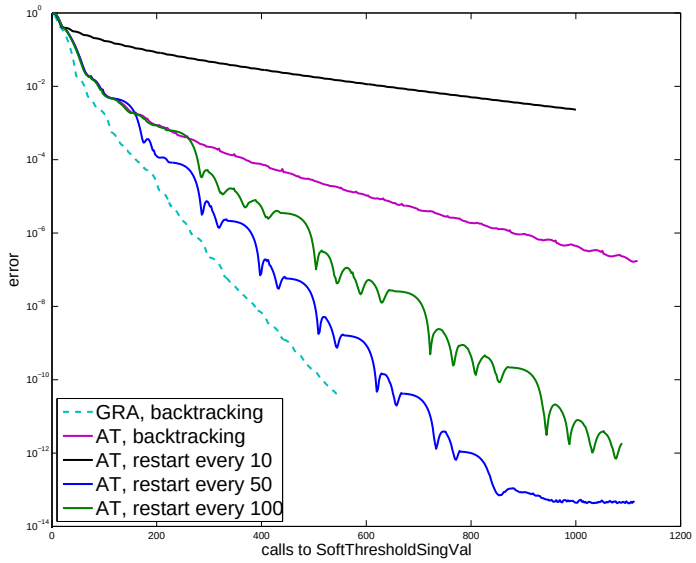

Figure 4.9: Noiseless matrix completion using various first-order methods

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

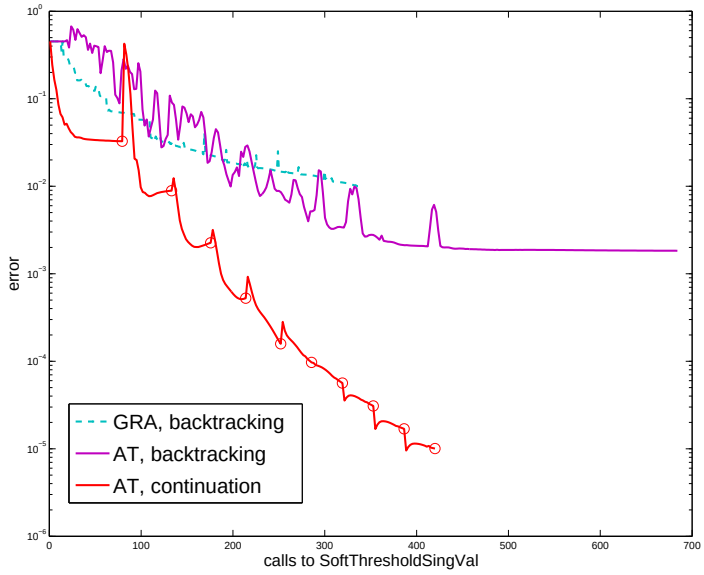

Figure 4.10: Noisy matrix completion on a non-low-rank matrix, using various first-order methods

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

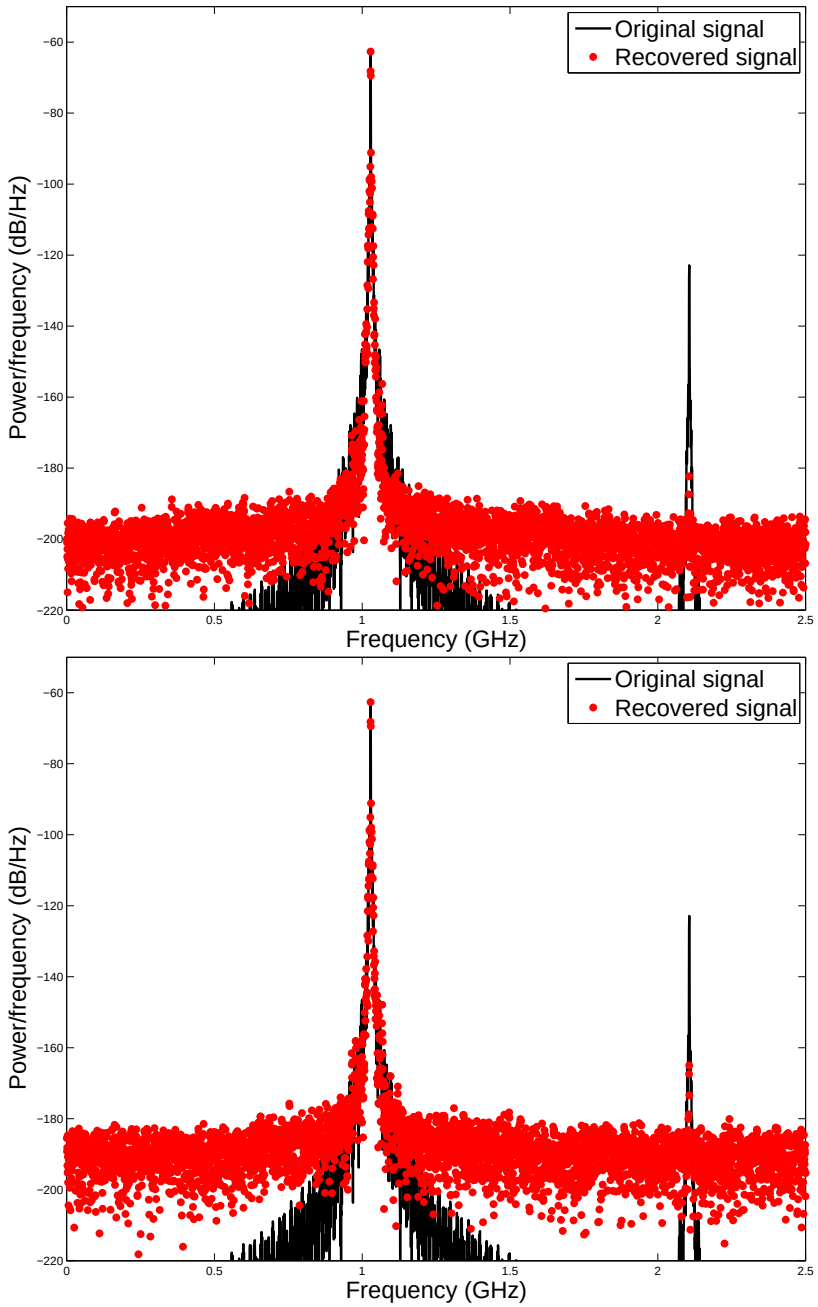

Figure 4.11: Recovery of a small pulse 60 dB below a large pulse. The first plot employs a Dantzig selector, the bottom employs LASSO.

	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.

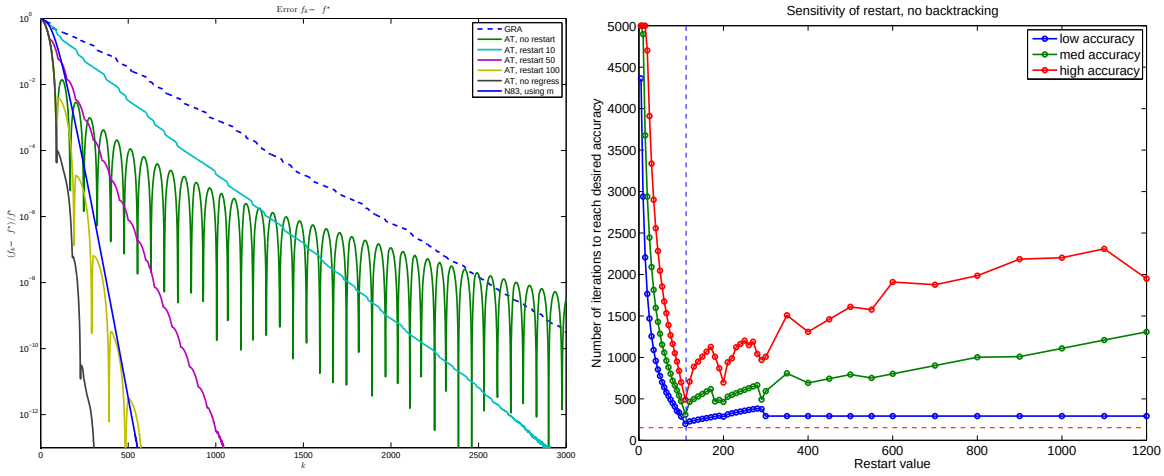

Figure 4.12: Plot (a) shows the Auslender-Teboulle method with and without restart, using various restart parameters. Also shown is the "no regress" restart option, which requires no parameters, and the N83 method using oracle knowledge of the strong convexity constant. Plot (b) shows the number of iterations 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

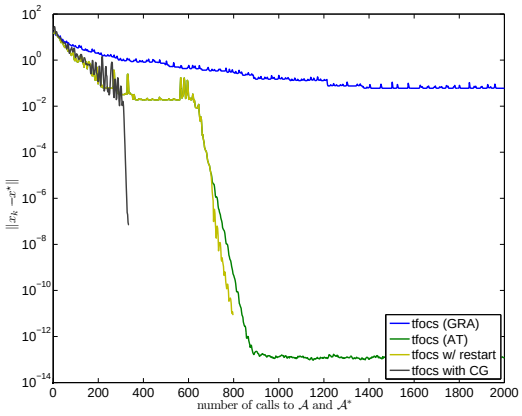

Figure 4.13: Noiseless basis pursuit, comparing a non-linear CG method to an optimal first-order method.

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).

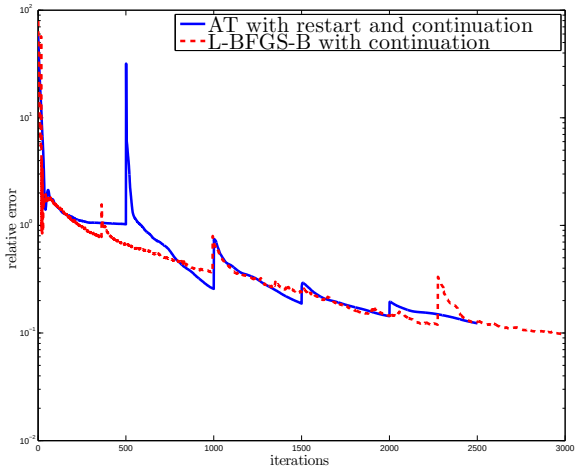

Figure 4.14: Comparison of AT method and L-BFGS-B on a standard form linear program. Both methods behave similarly.

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

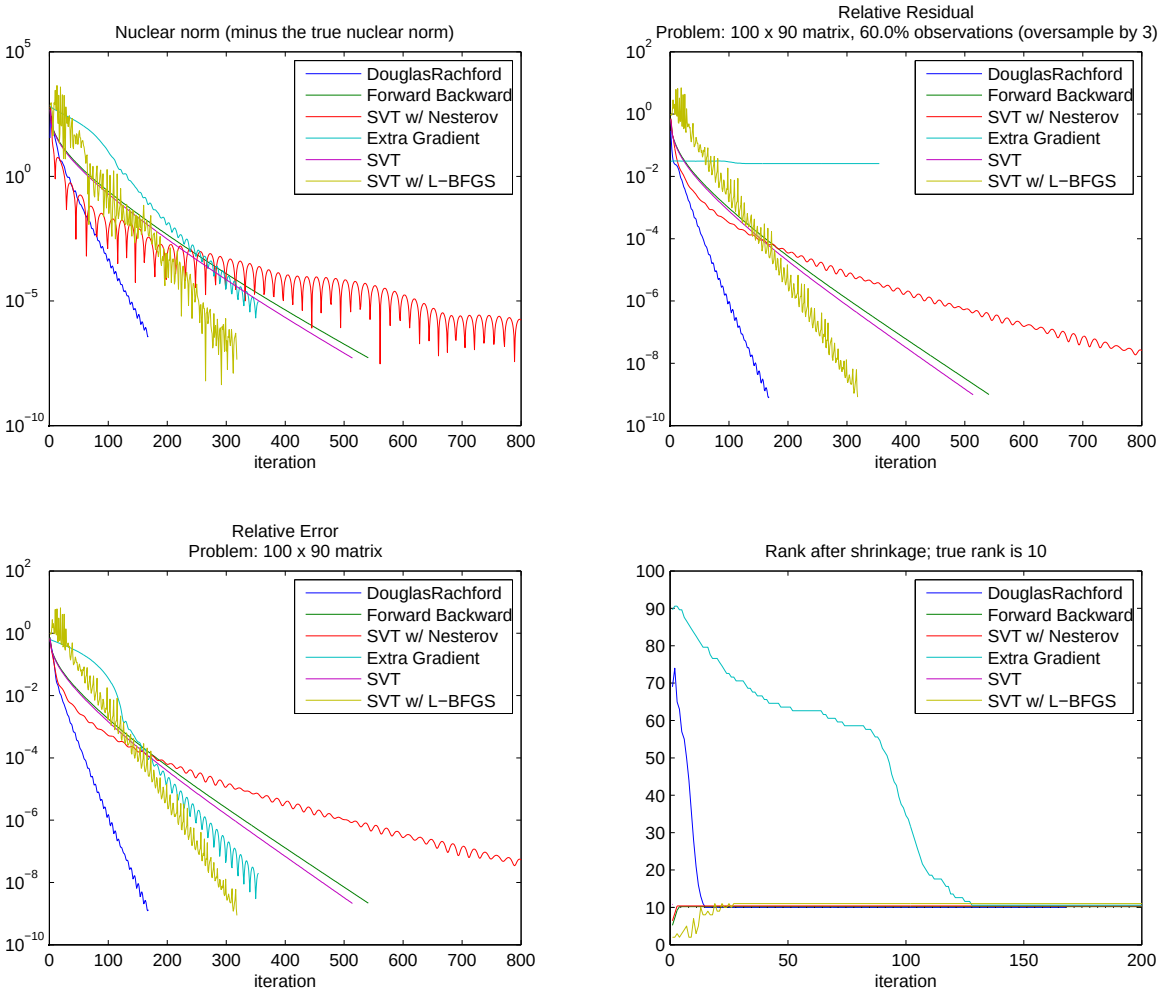

Figure 4.15: Comparison of various algorithms for noiseless matrix completion.

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.