Chapter 3

NESTA

This chapter, with the exception of §3.7, was jointly written with Jérôme Bobin and Emmanuel J. Candès, and is published in "NESTA: A Fast and Accurate First-Order Method for Sparse Recovery", *SIAM Journal on Imaging Sciences*, **4**, 1, January 2011 [BBC11].

This chapter applies a smoothing technique and an accelerated first-order algorithm, both from Nesterov [Nes05], and demonstrates that this approach is ideally suited for solving large-scale compressed sensing reconstruction problems as 1) it is computationally efficient, 2) it is accurate and returns solutions with several correct digits, 3) it is flexible and amenable to many kinds of reconstruction problems, and 4) it is robust in the sense that its excellent performance across a wide range of problems does not depend on the fine tuning of several parameters. Comprehensive numerical experiments on realistic signals exhibiting a large dynamic range show that this algorithm compares favorably with recently proposed state-of-the-art methods. We also apply the algorithm to solve other problems for which there are fewer alternatives, such as total-variation minimization, and convex programs seeking to minimize the ℓ_1 norm of Wx under constraints, in which W is not diagonal. The code is available online as a free package in the MATLAB language.

3.1 Introduction

Compressed sensing (CS) [CRT06, CT06, Don06] is a novel sampling theory, which is based on the discovery that one can exploit sparsity or compressibility when acquiring signals of general interest. In a nutshell, compressed sensing designs nonadaptive sampling techniques that condense the information in a compressible signal into a small amount of data. There are some indications that because of the significant reduction in the number of measurements needed to recover a signal accurately, engineers are changing the way they think about signal acquisition in areas ranging from analog-to-digital conversion [Hea07], digital optics, magnetic resonance imaging [LDP07], seismics [LH07], and astronomy [BSO08].

In this field, a signal $x^0 \in \mathbb{R}^n$ is acquired by collecting data of the form

$$b = Ax^0 + z,$$

where x^0 is the signal of interest (or its coefficient sequence in a representation where it is assumed to be fairly sparse), A is a known $m \times n$ "sampling" matrix, and z is a noise term. In compressed sensing and elsewhere, a standard approach attempts to reconstruct x^0 by solving

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & \|b - Ax\|_{\ell_2} \leq \epsilon, \end{array} \tag{3.1.1}$$

where ϵ^2 is an estimated upper bound on the noise power. The choice of the regularizing function f depends on prior assumptions about the signal x^0 of interest: if x^0 is (approximately) sparse, an appropriate convex function is the ℓ_1 norm (as advocated by the CS theory); if x^0 is a piecewise constant object, the total-variation norm provides accurate recovery results, and so on.

Solving large-scale problems such as (3.1.1) (think of x^0 as having millions of entries, as in mega-pixel images) is challenging. Although one cannot review the vast literature on this subject, the majority of the algorithms that have been proposed are unable to solve these problems *accurately* with low computational complexity. On the one hand, standard second-order methods such as interior-point methods [CR07b, KKB07, SK02] are accurate but problematic, for they need to solve large systems of linear equations to compute the Newton steps. On the other hand, inspired by iterative thresholding ideas [DDM04, FN03, CW05], we have now available a great number of first-order methods, see [WNF09, COS09, HYZ07, HYZ08] and the many earlier references therein, which may be faster but not necessarily accurate. Indeed, these methods are shown to converge slowly, and typically need a very large number of iterations when high accuracy is required.

We would like to pause on the demand for high accuracy since this is the main motivation of the present chapter. While in some applications, one may be content with one or two digits of accuracy, there are situations in which this is simply unacceptable. Imagine that the matrix A models a device giving information about the signal x^0, such as an analog-to-digital converter, for example. Here, the ability to detect and recover low-power signals that are barely above the noise floor, and possibly further obscured by large interferers, is critical to many applications. In mathematical terms, one could have a superposition of high power signals corresponding to components $x^0[i]$ of x^0 with magnitude of order 1, and low power signals with amplitudes as far as 100 dB down, corresponding to components with magnitude about 10^{-5}. In this regime of high-dynamic range, very high accuracy is required. In the example above, one would need at least five digits of precision as otherwise, the low power signals would go undetected.

Another motivation is solving (3.1.1) accurately when the signal x^0 is not exactly sparse, but

rather approximately sparse, as in the case of real-world compressible signals. Since exactly sparse signals are rarely found in applications—while compressible signals are ubiquitous—it is important to have an accurate first-order method to handle realistic signals.

3.1.1 Contributions

A few years ago, Nesterov [Nes05] published a seminal paper which couples smoothing techniques with an improved gradient method to derive first-order methods[1] which achieve a convergence rate he had proved to be optimal [Nes83] two decades earlier. As a consequence of this breakthrough, a few recent works have followed up with improved techniques in signal or image processing: in [BT09] a fast iterative thresholding algorithm was introduced to solve a class of composite functions that include (QP_λ), and [DHJJ10, WBFA09, Auj09] focused on total-variation minimization problems. In truth, these novel algorithms demonstrate great promise; they are fast, accurate, and robust in the sense that their performance does not depend on the fine tuning of various controlling parameters.

This chapter also focuses on applying some of Nesterov's works [Nes05] discussed just above, and proposes an algorithm—or, better said, a class of algorithms—for solving recovery problems from incomplete measurements. We refer to this algorithm as NESTA—a shorthand for Nesterov's algorithm—to acknowledge the fact that it is based on his method. The main purpose and the contribution of this chapter consist in showing that NESTA obeys the following desirable properties:

1. *Speed:* we will emphasize the case where A^*A is an orthogonal projector. Though being a special case, it is widespread in compressed sensing applications. In that case, NESTA is an iterative algorithm where each iteration is decomposed into three steps, each involving only a few matrix-vector operations. This, together with the accelerated convergence rate of Nesterov's algorithm [Nes05, BT09], makes NESTA a method of choice for solving large-scale problems. Furthermore, NESTA's convergence is mainly driven by a heuristically chosen single smoothing parameter μ introduced in Section 3.2.1. One can use continuation techniques [HYZ07, HYZ08] to dynamically update this parameter to substantially accelerate this algorithm.

2. *Accuracy:* NESTA depends on a few parameters that can be set in a very natural fashion. In fact, there is a clear relationship between the value of these parameters and the desired accuracy. Furthermore, our numerical experiments demonstrate that NESTA can find up to the first 4 or 5 significant digits of the optimal solution to (3.1.1), where $f(x)$ is the ℓ_1 norm or the total-variation norm of x, in a few hundred iterations. This makes NESTA amenable to solve recovery problems involving signals of very large sizes that also exhibit a great dynamic range.

[1] By "first-order", we mean methods which assume only that the local gradient information is available from a black-box oracle, but that once queried, the information persists; this is in contrast to a "pure" first-order method that only uses information about the current gradient.

3. *Flexibility:* NESTA can be adapted to solve many problems beyond ℓ_1 minimization with the same efficiency, such as total-variation (TV) minimization problems. In this chapter, we will also discuss applications in which f in (3.1.1) is given by $f(x) = \|Wx\|_{\ell_1}$, where one may think of W as a short-time Fourier transform also known as the Gabor transform, a curvelet transform, an undecimated wavelet transform, and so on, or a combination of these, or a general arbitrary dictionary of waveforms (note that this class of recovery problems also includes weighted ℓ_1 methods [CWB08]). This is particularly interesting because recent work [EMR07] suggests the potential advantage of this analysis-based approach over the classical *basis pursuit* in solving important inverse problems [EMR07].

A consequence of these properties is that NESTA, and more generally Nesterov's method, may be of interest to researchers working in the broad area of signal recovery from indirect and/or undersampled data.

Another contribution of this chapter is that it also features a fairly wide range of numerical experiments comparing various methods against problems involving realistic and challenging data. By challenging, we mean problems of very large scale where the unknown solution exhibits a large dynamic range; that is, problems for which classical second-order methods are too slow, and for which standard first-order methods do not provide sufficient accuracy. More specifically, Section 3.5 presents a comprehensive series of numerical experiments which illustrate the behavior of several state-of-the-art methods including interior point methods [KKB07], projected gradient techniques [HYZ07, vdBF08, FNW07], fixed point continuation, and iterative thresholding algorithms [HYZ07, YOGD08, BT09]. Most of these methods have either been perfected after several years of research [KKB07, FNW07], or are very recent state-of-the-art methods building on insight from the past few years. For example, the fixed point continuation method with active set [HYZ08], which represents a notable improvement over existing ideas, was released while we were working on the paper (and [ABDF11] was released while we were revising the paper).

3.1.2 Organization of the chapter and notations

As emphasized earlier, NESTA is based on Nesterov's ideas and Section 3.2.1 gives a brief but essential description of Nesterov's algorithmic framework. The proposed algorithm is introduced in Section 3.3. Inspired by continuation-like schemes, an accelerated version of NESTA is described in Section 3.3.6. We report on extensive and comparative numerical experiments in Section 3.5. Section 3.6 covers extensions of NESTA to minimize the ℓ_1 norm of Wx under data constraints (Section 3.6.1), and includes realistic simulations in the field of radar pulse detection and estimation. Section 3.6.3 extends NESTA to solve total-variation problems and presents numerical experiments which also demonstrate its remarkable efficiency there as well. Finally, we conclude with Section 3.8

discussing further extensions, which would address an even wider range of linear inverse problems.

Notations. Before we begin, it is best to provide a brief summary of the notations used throughout the chapter. As usual, vectors are written in small letters and matrices in capital letters. The ith entry of a vector x is denoted $x[i]$ and the (i,j)th entry of the matrix A is $A[i,j]$.

It is convenient to introduce some common optimization problems that will be discussed throughout. Solving sparse reconstruction problems can be approached via several different equivalent formulations. In this chapter, we particularly emphasize the quadratically constrained ℓ_1-minimization problem

$$\begin{array}{lll} (\mathrm{BP}_\epsilon) & \text{minimize} & \|x\|_{\ell_1} \\ & \text{subject to} & \|b - Ax\|_{\ell_2} \leq \epsilon, \end{array} \tag{3.1.2}$$

where ϵ quantifies the uncertainty about the measurements b as in the situation where the measurements are noisy. This formulation is often preferred because a reasonable estimate of ϵ may be known. A second frequently discussed approach considers solving this problem in Lagrangian form, i.e.,

$$(\mathrm{QP}_\lambda) \quad \text{minimize} \quad \lambda\|x\|_{\ell_1} + \frac{1}{2}\|b - Ax\|_{\ell_2}^2, \tag{3.1.3}$$

and is also known as the basis pursuit denoising problem (BPDN) [CDS98]. This problem is popular in signal and image processing [DDM04, FN03]; it is often interpreted as a maximum *a posteriori* estimate in a Bayesian setting or, simply, as a regularized least-squares problem. In statistics, the same problem is more well-known as the LASSO [Tib96]

$$\begin{array}{lll} (\mathrm{LS}_\tau) & \text{minimize} & \|b - Ax\|_{\ell_2} \\ & \text{subject to} & \|x\|_{\ell_1} \leq \tau. \end{array} \tag{3.1.4}$$

Standard optimization theory [Roc70] asserts that these three problems are of course equivalent provided that ϵ, λ, τ obey some special relationships. With the exception of the case where the matrix A is orthogonal, this functional dependence is hard to compute [vdBF08]. Because it is usually more natural to determine an appropriate ϵ rather than an appropriate λ or τ, the fact that NESTA solves (BP_ϵ) is a significant advantage. Further, note that theoretical equivalence of course does not mean that all three problems are just as easy (or just as hard) to solve. For instance, the constrained problem (BP_ϵ) may be harder to solve than (QP_λ). Hence, the fact that NESTA turns out to be competitive with algorithms that only solve (QP_λ) is quite remarkable.

3.2 NESTA

3.2.1 Nesterov's method to minimize smooth convex functions

In [Nes04, Nes83], Nesterov introduces a subtle algorithm to minimize any smooth convex function f on the convex set $\mathcal{Q}_p$,

$$\min_{x \in \mathcal{Q}_p} f(x). \tag{3.2.1}$$

We will refer to $\mathcal{Q}_p$ as the primal feasible set. The function f is assumed to be differentiable and its gradient $\nabla f(x)$ is Lipschitz and obeys

$$\forall x, y \in \mathrm{dom}(f), \quad ||\nabla f(x) - \nabla f(y)||_{\ell_2} \leq L \|x - y\|_{\ell_2}; \tag{3.2.2}$$

where $\mathrm{dom}(f) \equiv \mathbb{R}^n$ and L is an upper bound on the Lipschitz constant. With these assumptions, Nesterov's algorithm minimizes f over $\mathcal{Q}_p$ by iteratively estimating three sequences $\{x_k\}$, $\{y_k\}$, and $\{z_k\}$ while smoothing the feasible set $\mathcal{Q}_p$. The algorithm depends on two scalar sequences $\{\alpha_k\}$ and $\{\tau_k\}$, discussed below, and takes the following form (this specific form is based on [Nes05]):

Initialize x_0. For $k \geq 0$,

1. Compute $\nabla f(x_k)$.

2. Compute y_k:

   $y_k = \mathrm{argmin}_{x \in \mathcal{Q}_p} \frac{L}{2} \|x - x_k\|_{\ell_2}^2 + \langle \nabla f(x_k), x - x_k \rangle$.

3. Compute z_k:

   $z_k = \mathrm{argmin}_{x \in \mathcal{Q}_p} L p_p(x) + \sum_{i=0}^{k} \alpha_i \langle \nabla f(x_i), x - x_i \rangle$.

4. Update x_k:

   $x_{k+1} = \tau_k z_k + (1 - \tau_k) y_k$.

Stop when a given criterion is valid.

At step k, y_k is the current guess of the optimal solution. If we only performed the second step of the algorithm with y_{k-1} instead of x_k, we would obtain a standard first-order technique with convergence rate $\mathcal{O}(1/k)$. This second step can be seen as minimizing an approximate Taylor expansion of f about x_k, by taking an upper bound on the Hessian.

The novelty is that the sequence z_k "keeps in mind" the previous iterations since Step 3 involves a weighted sum of already computed gradients[2]. Another aspect of this step is that—borrowing ideas from smoothing techniques in optimization —it makes use of a *prox-function* $p_p(x)$ for the primal feasible set $\mathcal{Q}_p$. This function is strongly convex with parameter σ_p. Without loss of generality, we will set $\sigma_p = 1$. Assuming that $p_p(x)$ vanishes at the prox-center $x_p^c = \operatorname{argmin}_x p_p(x)$, this gives

$$p_p(x) \geq \frac{1}{2}\|x - x_p^c\|_{\ell_2}^2.$$

The prox-function is usually chosen so that $x_p^c \in \mathcal{Q}_p$, thus discouraging z_k from moving too far away from the center x_p^c.

The point x_k, at which the gradient of f is evaluated, is a weighted average between z_k and y_k. This is motivated by a theoretical analysis [Nes05, Tse08], which shows that if [3] $\alpha_k = 1/2(k+1)$ and $\tau_k = 2/(k+3)$, then the sequence $\{f(y_k)\}$ (y_k is feasible) converges to $f(x^\star)$ where $x^\star = \operatorname{argmin}_{x \in \mathcal{Q}_p} f(x)$, with the convergence rate (see [Nes05, Theorem 2])

$$f(y_k) - f(x^\star) \leq \frac{4Lp_p(x^\star)}{(k+1)^2}. \tag{3.2.3}$$

This decay is far better than what is achievable via standard gradient-based optimization techniques since we have an approximation scaling like L/k^2 instead of L/k.

3.3 Application to compressed sensing

We now apply Nesterov's algorithm to solve compressed sensing recovery problems, and refer to this extension as NESTA. For now, we shall be concerned with solving the quadratically constrained ℓ_1-minimization problem (3.1.2).

3.3.1 NESTA

We wish to solve (3.1.2), i.e., minimize $\|x\|_{\ell_1}$ subject to $\|b - Ax\|_{\ell_2} \leq \epsilon$, where $A \in \mathbb{R}^{m \times n}$ is singular ($m < n$).

[2] Using the sequence $\{\nabla f(x_i)\}_{1 \leq i \leq k}$ to update the current estimate x_k is not new. For instance, in algorithms such as SESOP [EMZ07], the search direction is optimally evaluated at each iteration in the subspace spanned by previous gradients; this requires storing a set of previous values, which may not be practical for large-scale problems.

[3] Other choices for α_k and τ_k are possible [Nes05].

In this chapter, with the exception of §3.7, we assume that A^*A is an orthogonal projector, i.e., the rows of A are orthonormal. This is often the case in compressed sensing applications where it is common to take A as a submatrix of a unitary transformation which admits a fast algorithm for matrix-vector products; special instances include the discrete Fourier transform, the discrete cosine transform, the Hadamard transform, the noiselet transform, and so on. Basically, collecting incomplete structured orthogonal measurements is the prime method for efficient data acquisition in compressed sensing. Section 3.8 discusses different approaches for relaxing this assumption.

From non-smooth to smooth functions

The algorithm described in the previous paragraph has been introduced to minimize a smooth function f; it cannot be applied directly to solve (BP_ϵ) as the function $f(x) = \|x\|_{\ell_1}$ is not smooth. In an innovative paper [Nes05], Nesterov recently extended this framework to deal with a certain class of non-smooth convex functions. Suppose that the function f to be minimized has the form

$$f(x) = \max_{u \in \mathcal{Q}_d} \langle u, Wx \rangle, \tag{3.3.1}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^p$, and $W \in \mathbb{R}^{p \times n}$. W is assumed to have full column rank. We will refer to $\mathcal{Q}_d$ as the dual feasible set, and assume it is convex. In [Nes05], Nesterov proposed substituting f by the smooth approximation

$$f_\mu(x) = \max_{u \in \mathcal{Q}_d} \langle u, Wx \rangle - \mu\, p_d(u), \tag{3.3.2}$$

where $p_d(u)$ is a *prox-function* for $\mathcal{Q}_d$; that is, $p_d(u)$ is continuous and strongly convex on $\mathcal{Q}_d$, with convexity parameter σ_d which we will take to be 1. Nesterov proved that f_μ is continuously differentiable, and that its gradient obeys

$$\nabla f_\mu(x) = W^* u_\mu(x), \tag{3.3.3}$$

where $u_\mu(x)$ is the optimal solution of (3.3.2). Furthermore, ∇f_μ is shown to be Lipschitz with constant

$$L = \frac{1}{\mu \sigma_d} \|W\|^2 \tag{3.3.4}$$

($\|W\|$ is the operator norm of W). It is now possible to apply Nesterov's accelerated method [Nes05] to the smoothed function f_μ. This produces a feasible sequence of points that will weakly converge to the minimizer of the smoothed problem at rate $\mathcal{O}(k^{-2})$, and empirically, the minimizers of the smoothed problem and the unsmoothed problem can be made arbitrarily close by taking $\mu \to 0$.

The particular case $f(x) = \|x\|_{\ell_1}$

In this setting, the ℓ_1 norm can be formulated as in (3.3.1) since

$$\|x\|_{\ell_1} = \max_{u \in \mathcal{Q}_d} \langle u, x \rangle,$$

where the dual feasible set $\mathcal{Q}_d$ is the ℓ_∞ ball $\mathcal{Q}_d = \{u : \|u\|_\infty \leq 1\}$. Therefore, a natural smooth approximation to the ℓ_1 norm is

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

where $p_d(u)$ is our dual prox-function. For $p_d(u)$, we would like a strongly convex function, which is known analytically and takes its minimum value (equal to zero) at some $u_d^c \in \mathcal{Q}_d$. It is also usual to have $p_d(u)$ separable. Taking these criteria into account, a convenient choice is $p_d(u) = \frac{1}{2}\|u\|_{\ell_2}^2$ whose strong convexity parameter is equal to 1. With this prox-function, f_μ is the well-known Huber function[4] and ∇f_μ is Lipschitz with constant $1/\mu$. In particular, $\nabla f_\mu(x)$ is given by

$$\nabla f_\mu(x)[i] = \begin{cases} \mu^{-1}\, x[i], & \text{if } |x[i]| < \mu, \\ \operatorname{sgn}(x[i]), & \text{otherwise.} \end{cases} \tag{3.3.5}$$

As proposed in [Nes05], Nesterov's algorithm can then be applied to solve

$$\min_{x \in \mathcal{Q}_p} f_\mu(x), \tag{3.3.6}$$

where $\mathcal{Q}_p = \{x : \|b - Ax\|_{\ell_2} \leq \epsilon\}$. Let us note that this problem is an approximation to (BP_ϵ) with $0 \leq f(x) - f_\mu(x) \leq n\mu/2$ for all $x \in \mathbb{R}^n$; this bound is overly pessimistic if x is s-sparse, since $f(0) = f_\mu(0)$, so the expected approximation error is closer to $s\mu/2$ for these signals.

The only remaining details of the algorithm are about the auxiliary updates of y_k and z_k in Steps 2 and 3. In the next section, we show that these steps are computationally very cheap when A^*A is assumed to be an orthogonal projector.[5]

3.3.2 Updating y_k

To compute y_k, we need to solve a problem of the form

$$y_k = \operatorname*{argmin}_{x \in \mathcal{Q}_p} \frac{L}{2}\|d - x\|_{\ell_2}^2 + \langle c, x - x_k \rangle, \tag{3.3.7}$$

[4]The smoothed version of the total-variation norm is not known to have a specific name.

[5]Taking $x_0 = A^*b$ which is feasible and observing that $f(x^\star) \leq f(A^*b)$ for any minimizer $x^\star$ implies the boundedness of $x^\star$ and, hence, that of $p_p(x^\star)$ in (3.2.3). This follows from $f(x) \leq \|x\|_{\ell_1} \leq f(x) + \mu n/2$ valid for all x, giving $\|x^\star\|_{\ell_1} \leq f(A^*b) + \mu n/2$.

where x_k is from the previous iterate and L is given by (3.3.4), $d = x_k$ and $c = \nabla f_\mu(x_k)$. The Lagrangian for this problem is

$$\mathcal{L}(x, \lambda) = \frac{L}{2}\|d - x\|_{\ell_2}^2 + \frac{\lambda}{2}\left(\|b - Ax\|_{\ell_2}^2 - \epsilon^2\right) + \langle c, x - x_k\rangle, \tag{3.3.8}$$

and at the primal-dual solution (y_k, λ_ϵ), the Karush-Kuhn-Tucker (KKT) conditions [Roc70] read

$$\begin{aligned} \|b - Ay_k\|_{\ell_2} &\leq \epsilon, \\ \lambda_\epsilon &\geq 0, \\ \lambda_\epsilon\left(\|b - Ay_k\|_{\ell_2}^2 - \epsilon^2\right) &= 0, \\ L(y_k - d) + \lambda_\epsilon A^*(Ay_k - b) + c &= 0. \end{aligned}$$

From the stationarity condition, y_k is the solution to the linear system

$$\left(I + \frac{\lambda_\epsilon}{L}A^*A\right) y_k = \frac{\lambda_\epsilon}{L}A^*b + d - \frac{1}{L}c. \tag{3.3.9}$$

As discussed earlier, our assumption is that A^*A is an orthogonal projector so that

$$y_k = \left(I - \frac{\lambda_\epsilon}{\lambda_\epsilon + L}A^*A\right)\left(\frac{\lambda_\epsilon}{L}A^*b + d - \frac{1}{L}c\right). \tag{3.3.10}$$

In this case, computing y_k is cheap since no matrix inversion is required—only a few matrix-vector products are necessary. Moreover, from the KKT conditions, the value of the optimal Lagrange multiplier is obtained explicitly, and equals

$$\lambda_\epsilon = \max(0, L\epsilon^{-1}\|b - Aq\|_{\ell_2} - L), \quad q = d - L^{-1}c \tag{3.3.11}$$

where $c = \nabla f_\mu(x_k)$ and $d = x_k$. Observe that this can be computed beforehand since it only depends on x_k and $\nabla f_\mu(x_k)$.

3.3.3 Updating z_k

To compute z_k, we need to solve

$$z_k = \underset{x \in \mathcal{Q}_p}{\operatorname{argmin}}\, Lp_p(x) + \langle \sum_{i \leq k} \alpha_i \nabla f_\mu(x_i), x - x_k\rangle, \tag{3.3.12}$$

where $p_p(x)$ is the primal prox-function. The point z_k differs from y_k since it is computed from a weighted cumulative gradient $\sum_{i \leq k} \alpha_i \nabla f_\mu(x_i)$, making it less prone to zig-zagging, which typically occurs when we have highly elliptical level sets. This step keeps a memory from the previous steps

and forces z_k to stay near the prox-center. A good primal prox-function is a smooth and strongly convex function that is likely to have some positive effect near the solution. In the setting of (3.1.1), a suitable smoothing prox-function may be

$$p_p(x) = \frac{1}{2}\|x - x_0\|_{\ell_2}^2 \tag{3.3.13}$$

for some $x_0 \in \mathbb{R}^n$, e.g., an initial guess of the solution. Other choices of primal feasible set $\mathcal{Q}_p$ may lead to other choices of prox-functions. For instance, when $\mathcal{Q}_p$ is the standard simplex, choosing an entropy distance for $p_p(x)$ is smarter and more efficient, see [Nes05]. In this chapter, the primal feasible set is quadratic, which makes the Euclidean distance a reasonable choice. What is more important, however, is that this choice allows very efficient computations of z_k, while other choices may considerably slow down each Nesterov iteration. Finally, notice that the bound on the error at iteration k in (3.2.3) is proportional to $p_p(x^\star)$; choosing x_0 wisely (a good first guess) can make $p_p(x^\star)$ small. When nothing is known about the solution, a natural choice may be $x_0 = A^*b$; this idea will be developed in Section 3.3.6.

With (3.3.13), the strong convexity parameter of $p_p(x)$ is equal to 1, and to compute z_k we need to solve a quadratically constrained quadratic program that has exactly the same formulation as (3.3.7), where $d = x_0$ and $c = \sum_{i \le k} \alpha_i \nabla f_\mu(x_i)$. Updating z_k is then just as cheap as updating y_k. Furthermore, many of the matrix-vector multiplications from the y_k update can be reused.

3.3.4 Computational complexity

The computational complexity of each of NESTA's steps is clear. In large-scale problems, most of the work is in the application of A and A^*. Define $\mathcal{C}_A$ to be the complexity of applying A or A^*. The first step, namely, computing ∇f_μ, only requires vector operations whose complexity is $\mathcal{O}(n)$. Steps 2 and 3 require the application of A or A^* three times each (we only need to compute A^*b once). Hence, the total complexity of a single NESTA iteration is $6\,\mathcal{C}_A + \mathcal{O}(n)$ where $\mathcal{C}_A$ is dominant.

The calculations above are in some sense overly pessimistic. In compressed sensing applications, it is common to choose A as a submatrix of a unitary transformation U, which admits a fast algorithm for matrix-vector products. In the sequel, it might be useful to think of A as a subsampled DFT. In this case, letting R be the $m \times n$ matrix extracting the observed measurements, we have $A = RU$. The trick then is to compute in the U-domain directly. Making the change of variables $x \leftarrow Ux$, our problem is

$$\begin{array}{ll} \text{minimize} & \hat{f}_\mu(x) \\ \text{subject to} & \|b - Rx\|_{\ell_2} \le \epsilon, \end{array}$$

where $\hat{f}_\mu = f_\mu \circ U^*$. The gradient of $\hat{f}_\mu$ is then

$$\nabla \hat{f}_\mu(x) = U \, \nabla f_\mu(U^* x).$$

With this change of variables, Steps 2 and 3 do not require applying U or U^* since

$$y_k = \left(I - \frac{\lambda}{\lambda + L} R^* R \right) \left(\frac{\lambda}{L} R^* b + x_k - \frac{1}{L} \nabla \hat{f}_\mu(x_k) \right),$$

where R^*R is the diagonal matrix with 0/1 diagonal entries depending on whether a coordinate is sampled or not. As before, $\lambda_\epsilon = \max(0, \|b - Rq\|_{\ell_2} - L)$ with $q = x_k - L^{-1} \nabla \hat{f}_\mu(x_k)$. The complexity of Step 2 is now $\mathcal{O}(n)$ and the same applies to Step 3.

Put $\mathcal{C}_U$ for the complexity of applying U and U^*. The complexity of Step 1 is now $2\,\mathcal{C}_U$, so that this simple change of variables reduces the cost of each NESTA iteration to $2\,\mathcal{C}_U + \mathcal{O}(n)$. For example, in the case of a subsampled DFT (or something similar), the cost of each iteration is essentially that of two FFTs. Hence, each iteration is extremely fast.

3.3.5 Parameter selection

NESTA involves the selection of a single smoothing parameter μ and of a suitable stopping criterion. For the latter, our experience indicates that a robust and fairly natural stopping criterion is to terminate the algorithm when the relative variation of f_μ is small. Define Δf_μ as

$$\Delta f_\mu := \frac{|f_\mu(x_k) - \bar{f}_\mu(x_k)|}{\bar{f}_\mu(x_k)}, \quad \bar{f}_\mu(x_k) := \frac{1}{\min\{10, k\}} \sum_{l=1}^{\min\{10,k\}} f_\mu(x_{k-l}). \tag{3.3.14}$$

Then convergence is claimed when

$$\Delta f_\mu < \delta$$

for some $\delta > 0$. In our experiments, $\delta \in \{10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$ depending upon the desired accuracy. These values might depend on the problem type and should be optimized accordingly. Clearly, δ should be small when μ is small, since this is the high accuracy situation; when μ is larger, δ may be taken to be larger as well.

The choice of μ is based on a tradeoff between the accuracy of the smoothed approximation f_μ (basically, $\lim_{\mu \to 0} f_\mu(x) = \|x\|_{\ell_1}$) and the speed of convergence (the convergence rate is proportional to μ). With noiseless data, μ is directly linked to the desired accuracy. To illustrate this, we have observed in [BC09] that when the true signal x^0 is exactly sparse and is actually the minimum solution under the equality constraints $Ax^0 = b$, the ℓ_∞ error on the nonzero entries is on the order of μ. The link between μ and accuracy will be further discussed in Section 3.4.3.

3.3.6 Accelerating NESTA with continuation

Inspired by homotopy techniques which find the solution to the LASSO problem (3.1.4) for values of τ ranging in an interval $[0, \tau_{\max}]$, [HYZ07] and [DFL08] independently introduced a fixed point continuation technique which solves ℓ_1-penalized least-square problems (3.1.3),

$$(\mathrm{QP}_\lambda) \quad \text{minimize} \quad \lambda \|x\|_{\ell_1} + \frac{1}{2}\|b - Ax\|_{\ell_2}^2,$$

for values of λ obeying $0 < \lambda < \|A^* b\|_{\ell_\infty}$. The continuation solution approximately follows the path of solutions to the problem (QP_λ) and, hence, the solutions to (3.1.1) and (3.1.4) may be found by solving a sequence of ℓ_1 penalized least-squares problems.

The point of this is that it has been noticed (see [HYZ07, OPT00, DT08, WNF09]) that solving (3.1.3) (resp. the LASSO (3.1.4)) is faster when λ is large (resp. τ is low). This observation greatly motivates the use of continuation for solving (3.1.3) for a fixed λ_f. The idea is simple: propose a sequence of problems with decreasing values of the parameter λ, $\lambda_0 > \cdots > \lambda_f$, and use the intermediate solution as a warm start for the next problem. This technique has been used with success in [FNW07, vdBF08]. Continuation has been shown to be a very successful tool to increase the speed of convergence, in particular when dealing with large-scale problems and high dynamic range signals.

Likewise, our proposed algorithm can greatly benefit from a continuation approach. Recall that to compute y_k, we need to solve

$$\begin{aligned} y_k &= \operatorname*{argmin}_{x \in \mathcal{Q}_p} \frac{L}{2}\|x - x_k\|_{\ell_2}^2 + \langle c, x \rangle \\ &= \operatorname*{argmin}_{x \in \mathcal{Q}_p} \|x - (x_k - L^{-1}c)\|_{\ell_2}^2 \end{aligned}$$

for some vector c. Thus with $\mathcal{P}_{\mathcal{Q}_p}$ the projector onto $\mathcal{Q}_p$, $y_k = \mathcal{P}_{\mathcal{Q}_p}(x_k - L^{-1}c)$. Now two observations are in order.

1. Computing y_k is similar to a projected gradient step as the Lipschitz constant L^{-1} plays the role of the step size. Since L is proportional to μ^{-1}, the larger μ, the larger the step-size, and the faster the convergence. This also applies to the sequence $\{z_k\}$.

2. For a fixed value of μ, the convergence rate of the algorithm obeys

$$f_\mu(y_k) - f_\mu(x_\mu^\star) \leq \frac{2L\|x_\mu^\star - x_0\|_{\ell_2}^2}{k^2},$$

where $x_\mu^\star$ is the optimal solution to $\min f_\mu$ over $\mathcal{Q}_p$. On the one hand, the convergence rate is proportional to μ^{-1}, so a large value of μ is beneficial. On the other hand, choosing a good

guess x_0 close to $x^\star_\mu$ provides a low value of $p_p(x^\star_\mu) = \frac{1}{2}\|x^\star_\mu - x_0\|^2_{\ell_2}$, also improving the rate of convergence. Warm-starting with x_0 from a previous solve not only changes the starting point of the algorithm, but it beneficially changes p_p as well.

These two observations motivate the following continuation-like algorithm:

Initialize μ_0, x_0 and the number of continuation steps T. For $t = 1, \cdots, T$,

1. Apply Nesterov's algorithm with $\mu = \mu^{(t)}$ and $x_0 = x_{\mu^{(t-1)}}$.

2. Decrease the value of μ: $\mu^{(t+1)} = \gamma\mu^{(t)}$ with $\gamma < 1$.

Stop when the desired value of μ_f is reached.

This algorithm iteratively finds the solutions to a succession of problems with decreasing smoothing parameters $\mu_0 > \cdots > \mu_f = \gamma^T\mu_0$ producing a sequence of—hopefully—finer estimates of $x^\star_{\mu_f}$ [6]; these intermediate solutions are cheap to compute and provide a string of convenient first guesses for the next problem. In practice, they are solved with less accuracy, making them even cheaper to compute. This kind of continuation is different from standard continuation techniques [HYZ07, OPT00, DT08, WNF09]. Indeed, while standard continuation solves a sequence of problems (QP_λ) with different values of the Lagrange multiplier λ (so that intermediate solutions may have a meaningful interpretation if λ is not decreased too quickly), the proposed continuation technique acts internally by changing the smoothing parameter μ while keeping ϵ and, therefore, the feasible set, fixed. While intermediate solutions may have a less meaningful interpretation, the continuation may actually be more effective than traditional continuation, since not only is the new step warm-started, but the primal prox-function is improved.

The value of μ_f is based on a desired accuracy as explained in Section 3.3.5. As for an initial value μ_0, (3.3.5) makes clear that the smoothing parameter plays a role similar to a threshold. A first choice may then be $\mu_0 = 0.9\|A^*b\|_{\ell_\infty}$.

We illustrate the good behavior of the continuation-inspired algorithm by applying NESTA with continuation to solve a sparse reconstruction problem from partial frequency data. In this series of experiments, we assess the performance of NESTA while the dynamic range of the signals to be recovered increases.

The signals x are s-sparse signals—that is, have exactly s nonzero components—of size $n = 4096$ and $s = m/40$. Put Λ for the indices of the nonzero entries of x; the amplitude of each nonzero entry is distributed uniformly on a logarithmic scale with a fixed dynamic range. Specifically, each

[6] The parameter γ does not define a new parameter; it depends upon μ_f, μ_0, and T via $\gamma = (\mu_f/\mu_0)^{1/T}$.

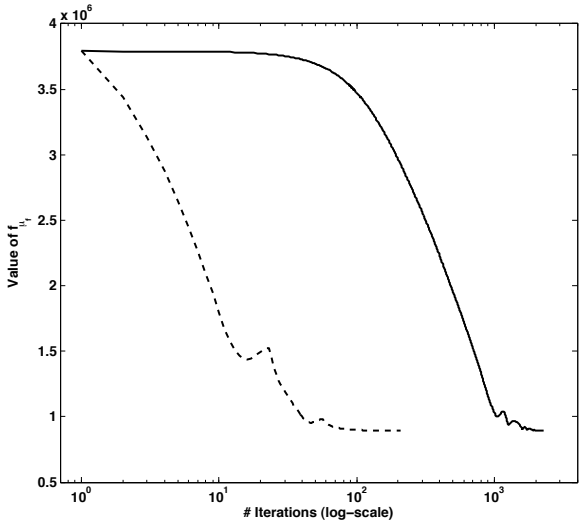

Figure 3.1: Value of $f_{\mu_f}(x_k)$ as a function of iteration k. Solid line: without continuation. Dashed line: with continuation. Here, the test signal has 100 dB of dynamic range.

nonzero entry is generated as follows:

$$x[i] = \eta_1[i] 10^{\alpha \eta_2[i]}, \tag{3.3.15}$$

where $\eta_1[i] = \pm 1$ with probability $1/2$ (a random sign) and $\eta_2[i]$ is uniformly distributed in $[0, 1]$. The parameter α quantifies the dynamic range. Unless specified otherwise, a dynamic range of d dB means that $\alpha = d/20$ (since for large signals α is approximately the logarithm base 10 of the ratio between the largest and the lowest magnitudes). For instance, 80 dB signals are generated according to (3.3.15) with $\alpha = 4$.

The measurements Ax consist of $m = n/8$ random discrete cosine measurements so that A^*A is diagonalized by the DCT. Finally, b is obtained by adding a white Gaussian noise term with standard deviation $\sigma = 0.1$. The initial value of the smoothing parameter is $\mu_0 = \|A^*b\|_{\ell_\infty}$ and the terminal value is $\mu_f = 2\sigma$. The algorithm terminates when the relative variation of f_μ is lower than $\delta = 10^{-5}$. NESTA with continuation is applied to 10 random trials for varying number of continuation steps T and various values of the dynamic range. Figure 3.1 graphs the value of f_{μ_f} while applying NESTA with and without continuation as a function of the iteration count. The number of continuation steps is set to $T = 4$.

One can observe that computing the solution to $\min f_{\mu_f}$ (solid line) takes a while when computed with the final value μ_f; notice that NESTA seems to be slow at the beginning (number of iterations lower than 15). In the meantime NESTA with continuation rapidly estimates a sequence of coarse intermediate solutions that converges to the solution to $\min f_{\mu_f}$ In this case, continuation clearly enhances the global speed of convergence with a factor 10. Figure 3.2 provides deeper insights into

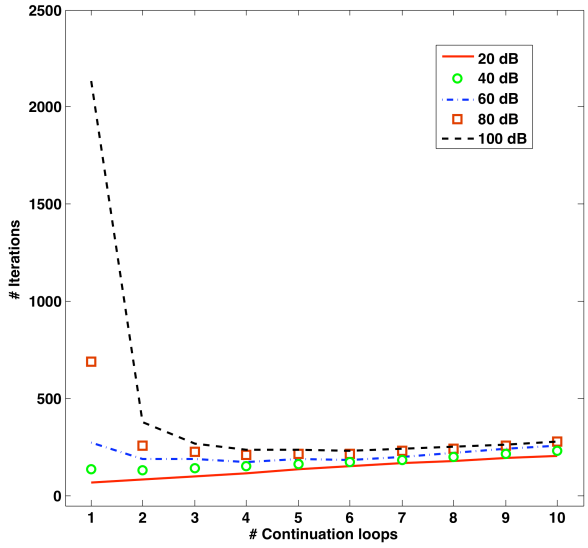

Figure 3.2: Total number of iterations required for convergence with a varying number of continuation steps and for different values of the dynamic range

the behavior of continuation with NESTA and shows the number of iterations required to reach convergence for varying values of the continuation steps T for different values of the dynamic range.

When the ratio μ_0/μ_f is low or when the required accuracy is low, continuation is not as beneficial: intermediate continuation steps require a number of iterations which may not speed up overall convergence. The step-size which is about $L_{\mu_f}^{-1}$ works well in this regime. When the dynamic range increases and we require more accuracy, however, the ratio μ_0/μ_f is large, since $\mu_0 = .9\|A^*b\|_{\ell_\infty} \approx \|x\|_{\ell_\infty} \gg \sigma$, and continuation provides considerable improvements. In this case, the step size $L_{\mu_f}^{-1}$ is too conservative and it takes a while to find the large entries of x. Empirically, when the dynamic range is 100 dB, continuation improves the speed of convergence by a factor of 8. As this factor is likely to increase exponentially with the dynamic range (when expressed in dB), NESTA with continuation seems to be a better candidate for solving sparse reconstruction problems with high accuracy.

Interestingly, the behavior of NESTA with continuation seems to be quite stable: increasing the number of continuation steps does not increase dramatically the number of iterations. In practice, although the ideal T is certainly signal dependent, we have observed in our problem settings that choosing $T \in \{4, 5, 6\}$ leads to reasonable results. The value of T should certainly be optimized for particular problem types.

3.3.7 Some theoretical considerations

The convergence of NESTA with and without continuation is straightforward. The following theorem states that at each continuation step with $\mu = \mu^{(t)}$, the sequence $f_\mu(y_k)$ converges to $f_\mu(x_\mu^\star)$. Global

convergence is proved by applying this theorem to $t = T$.

Theorem 3.3.1. *At each continuation step t, $\lim_{k\to\infty} f_{\mu^{(t)}}(y_k) = f_{\mu^{(t)}}(x^\star_{\mu^{(t)}})$, and*

$$f_{\mu^{(t)}}(y_k) - f_{\mu^{(t)}}(x^\star_{\mu^{(t)}}) \leq \frac{2L_{\mu^{(t)}}\|x^\star_{\mu^{(t)}} - x_{\mu^{(t-1)}}\|^2_{\ell_2}}{k^2}.$$

Proof. Immediate by using [Nes05, Theorem 2]. □

As mentioned earlier, continuation may be valuable for improving the speed of convergence. Let each continuation step t stop after $\mathcal{N}^{(t)}$ iterations with

$$\mathcal{N}^{(t)} = \sqrt{\frac{2L_{\mu^{(t)}}}{\gamma^t \delta_0}}\|x^\star_{\mu^{(t)}} - x^\star_{\mu^{(t-1)}}\|_{\ell_2}$$

so that we have

$$f_{\mu^{(t)}}(y_k) - f_{\mu^{(t)}}(x^\star_{\mu^{(t)}}) \leq \gamma^t \delta_0,$$

where the accuracy $\gamma^t\delta_0$ becomes tighter as t increases. Then summing up the contribution of all the continuation steps gives

$$\mathcal{N}_c = \sqrt{\frac{2}{\mu_0\delta_0}}\sum_{t=1}^{T}\gamma^{-t}\|x^\star_{\mu^{(t)}} - x^\star_{\mu^{(t-1)}}\|_{\ell_2}.$$

When NESTA is applied without continuation, the number of iterations required to reach convergence is

$$\mathcal{N} = \sqrt{\frac{2}{\mu_0\delta_0}}\gamma^{-T}\|x^\star_{\mu_f} - x_0\|_{\ell_2}.$$

Now the ratio $\mathcal{N}_c/\mathcal{N}$ is given by

$$\frac{\mathcal{N}_c}{\mathcal{N}} = \sum_{t=1}^{T}\gamma^{T-t}\frac{\|x^\star_{\mu^{(t)}} - x^\star_{\mu^{(t-1)}}\|_{\ell_2}}{\|x^\star_{\mu_f} - x_0\|_{\ell_2}}. \tag{3.3.16}$$

Continuation is definitely worthwhile when the right-hand side is smaller than 1. Interestingly, this quantity is directly linked to the path followed by the sequence $x_0 \to x_{\mu^{(1)}} \to \cdots \to x_{\mu_f}$. More precisely, it is related to the smoothness of this path; for instance, if all the intermediate points $x_{\mu^{(t)}}$ belong to the segment $[x_0, x_{\mu_f}]$ in an ordered fashion, then $\sum_t \|x^\star_{\mu^{(t)}} - x_{\mu^{(t-1)}}\|_{\ell_2} = \|x^\star_{\mu_f} - x_0\|_{\ell_2}$. Hence, $\frac{\mathcal{N}_c}{\mathcal{N}} < 1$ and continuation improves the convergence rate.

Figure 3.3 illustrates two typical solution paths with continuation. When the sequence of solutions obeys $\|x_0\|_{\ell_1} \geq \ldots \|x^\star_{\mu^{(t)}}\|_{\ell_1} \ldots \geq \|x^\star_{\mu_f}\|_{\ell_1}$ (this is the case when $x_0 = A^*b$ and $\mu_1 \geq \ldots \mu^{(t)} \ldots \geq \mu_f$), the solution path is likely to be "smooth"; that is, the solutions obey $\|x^\star_{\mu^{(t)}} - x^\star_{\mu_f}\|_{\ell_2} \geq \|x^\star_{\mu^{(t+1)}} - x^\star_{\mu_f}\|_{\ell_2}$ as on the left of Figure 3.3. The "non-smooth" case on the right of

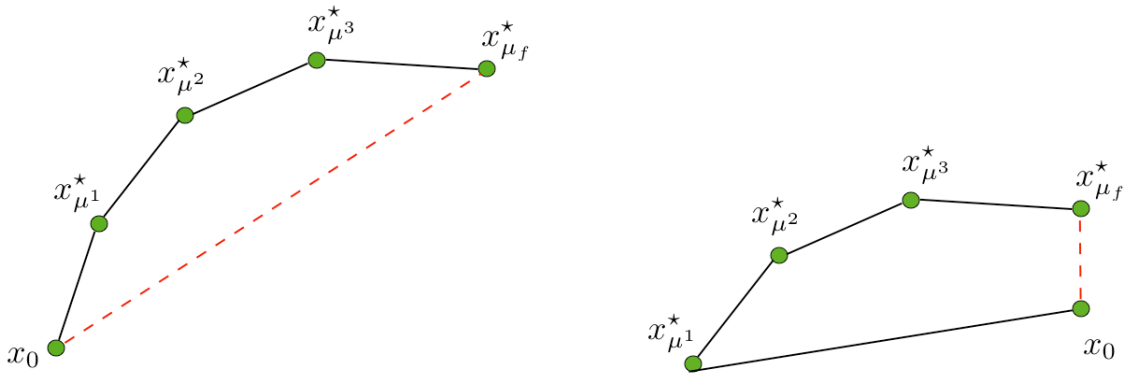

Figure 3.3: Typical solution paths. Left: smooth solution path. Right: non-smooth solution path

Figure 3.3 arises when the sequence of smoothing parameters does not provide estimates of $x^\star_{\mu_f}$ that are all better than x_0. Here, computing some of the intermediate points $\{x^\star_{\mu^{(t)}}\}$ is wasteful and continuation fails to be faster.

3.4 Accurate optimization

A significant fraction of the numerical part of this chapter focuses on comparing different sparse recovery algorithms in terms of speed and accuracy. In this section, we first demonstrate that NESTA can easily recover the exact solution to (BP_ϵ) with a precision of 5 to 6 digits. Speaking of precision, we shall essentially use two criteria to evaluate accuracy.

1. The first is the (relative) error on the objective functional

$$\frac{\|x\|_{\ell_1} - \|x^\star\|_{\ell_1}}{\|x^\star\|_{\ell_1}}, \tag{3.4.1}$$

   where $x^\star$ is the optimal solution to (BP_ϵ) .

2. The second is the accuracy of the optimal solution itself and is measured via

$$\|x - x^\star\|_{\ell_\infty}, \tag{3.4.2}$$

   which gives a precise value of the accuracy per entry.

3.4.1 Is NESTA accurate?

For general problem instances, the exact solution to (BP_ϵ) (or equivalently (QP_λ)) cannot be computed analytically. Under some conditions, however, a simple formula is available when the optimal

Table 3.1: Assessing FISTA's and NESTA's accuracy when the optimal solution is known. The relative error on the optimal value is given by (3.4.1) and the ℓ_∞ error on the optimal solution by (3.4.2). $\mathcal{N}_A$ is the number of calls to A or A^* to compute the solution.

Method	ℓ_1-norm	Rel. error ℓ_1-norm	ℓ_∞ error	$\mathcal{N}_A$
$x^\star$	3.33601e+6			
FISTA	3.33610e+6	2.7e-5	0.31	40000
NESTA $\mu = 0.02$	3.33647e+6	1.4e-4	0.08	513

solution has exactly the same support and the same sign as the unknown (sparse) x^0 (recall the model $b = Ax^0 + z$). Denote by I the support of x^0, $I := \{i : |x^0[i]| > 0\}$. Then if x^0 is sufficiently sparse and if the nonzero entries of x^0 are sufficiently large, the solution $x^\star$ to (QP_λ) is given by

$$x^\star[I] = (A[I]^* A[I])^{-1}(A[I]^* b - \lambda \operatorname{sgn}(x^0[I])), \tag{3.4.3}$$

$$x^\star[I^c] = 0, \tag{3.4.4}$$

see [CP07a] for example. In this expression, $x[I]$ is the vector with indices in I and $A[I]$ is the submatrix with columns indices in I.

To evaluate NESTA's accuracy, we set $n = 262{,}144$, $m = n/8$, and $s = m/100$ (this is the number of nonzero coordinates of x_0). The absolute values of the nonzero entries of x_0 are distributed between 1 and 10^5 so that we have about 100 dB of dynamic range. The measurements Ax^0 are discrete cosine coefficients selected uniformly at random. We add Gaussian white noise with standard deviation $\sigma = 0.01$. We then compute the solution (3.4.3), and make sure it obeys the KKT optimality conditions for (QP_λ) so that the optimal solution is known.

We run NESTA with continuation with the value of $\epsilon := \|b - Ax^\star\|$. We use $\mu = 0.02$, $\delta = 10^{-7}$ and the number of continuation steps is set to 5. Table 3.1 reports on numerical results. First, the value of the objective functional is accurate up to 4 digits. Second, the computed solution is very accurate since we observe an ℓ_∞ error of 0.08. Now recall that the nonzero components of $x^\star$ vary from about 1 to 10^5 so that we have high accuracy over a huge dynamic range. This can also be gleaned from Figure 3.4 which plots NESTA's solution versus the optimal solution, and confirms the excellent precision of our algorithm.

3.4.2 Setting up a reference algorithm for accuracy tests

In general situations, a formula for the optimal solution is of course unavailable, and evaluating the accuracy of solutions requires defining a method of reference. In this chapter, we will use FISTA [BT09] as such a reference since it is an efficient algorithm that also turns out to be extremely easy to use; in particular, no parameter has to be tweaked, except for the standard stopping criterion (maximum number of iterations and tolerance on the relative variation of the objective function).

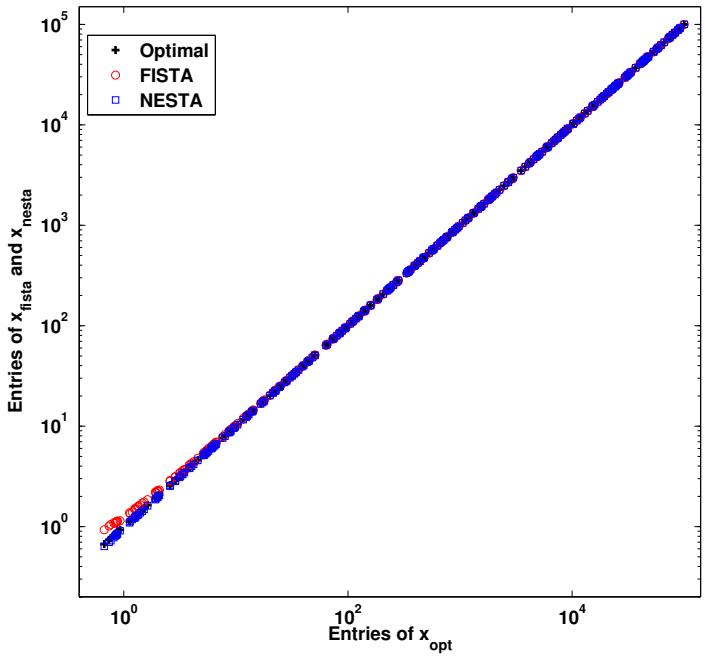

Figure 3.4: Entries of the computed solutions versus the optimal solution. The absolute values of the entries on the support of the optimal solution are plotted.

We run FISTA with 20,000 iterations on the same problem as above, and report its accuracy in Table 3.1. The ℓ_1-norm is exact up to 4 digits. Furthermore, Figure 3.4 shows the entries of FISTA's solution versus those of the optimal solution, and one observes a very good fit (near perfect when the magnitude of a component of $x^\star$ is higher than 3). The ℓ_∞ error between FISTA's solution and the optimal solution $x^\star$ is equal to 0.31; that is, the entries are exact up to ± 0.31. Because this occurs over an enormous dynamic range, we conclude that FISTA also gives very accurate solutions provided that sufficiently many iterations are taken. We have observed that running FISTA with a high number of iterations—typically greater than 20,000—provides accurate solutions to (QP_λ), and this is why we will use it as our method of reference in the forthcoming comparisons from this section and the next.

3.4.3 The smoothing parameter μ and NESTA's accuracy

By definition, μ fixes the accuracy of the approximation f_μ to the ℓ_1 norm and, therefore, NESTA's accuracy directly depends on this parameter. We now propose to assess the accuracy of NESTA for different values of μ. The problem sizes are as before, namely, $n = 262{,}144$ and $m = n/8$, except that now the unknown x^0 is far less sparse with $s = m/5$. The standard deviation of the additive Gaussian white noise is also higher, and we set $\sigma = 0.1$.

Because of the larger value of s and σ, it is no longer possible to have an analytic solution from (3.4.3). Instead, we use FISTA to compute a reference solution x_F, using 20,000 iterations and with

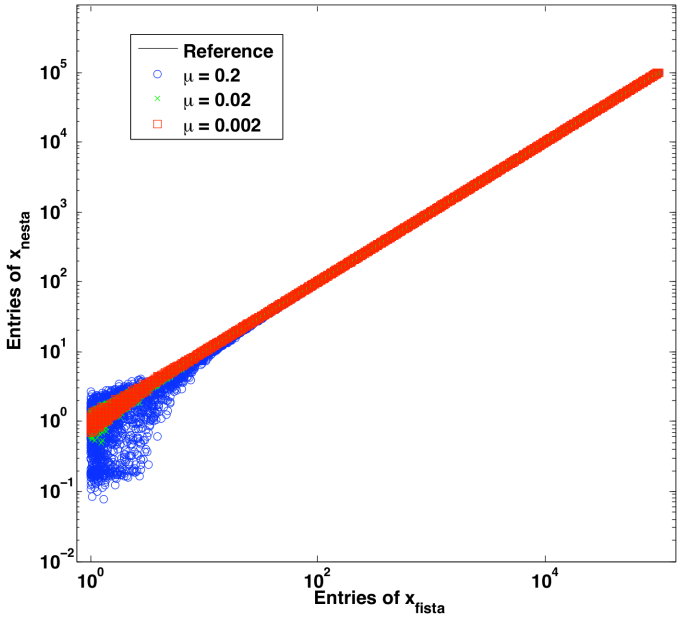

Figure 3.5: Entries of the computed solutions versus the optimal solution. We plot the absolute values of the entries on the set where the magnitude of the optimal solution exceeds 1.

$\lambda = 0.0685$, which gives $\|b - Ax_F\|_{\ell_2}^2 \simeq (m + 2\sqrt{2m})\sigma^2$. To be sure that FISTA's solution is very close to the optimal solution, we check that the KKT stationarity condition is nearly verified. If $I_\star$ is the support of the optimal solution $x^\star$, this condition reads

$$A[I_\star]^*(b - Ax^\star) = \lambda \operatorname{sgn}(x^\star[I_\star]),$$
$$\|A[I_\star^c]^*(b - Ax^\star)\|_{\ell_\infty} \leq \lambda.$$

Now define I to be the support of x_F. Then, here, x_F obeys

$$\|A[I]^*(b - Ax_F) - \lambda \operatorname{sgn}(x_F[I])\|_{\ell_\infty} = 2.6610^{-10}\lambda,$$
$$\|A[I^c]^*(b - Ax_F)\|_{\ell_\infty} \leq 0.99\lambda.$$

This shows that x_F is extremely close to the optimal solution.

NESTA is run with $T = 5$ continuation steps for three different values of $\mu \in \{0.2, 0.02, 0.002\}$ (the tolerance δ is set to 10^{-6}, 10^{-7}, and 10^{-8} respectively). Figure 3.5 plots the solutions given by NESTA versus the "optimal solution" x_F. Clearly, when μ decreases, the accuracy of NESTA increases just as expected. More precisely, notice in Table 3.2 that for this particular experiment, decreasing μ by a factor of 10 gives about 1 additional digit of accuracy on the optimal value.

According to this table, $\mu = 0.02$ seems a reasonable choice for this particular signal to guarantee an accurate solution, since one has between 4 and 5 digits of accuracy on the optimal value, and

Table 3.2: NESTA's accuracy. The errors and number of function calls $\mathcal{N}_A$ have the same meaning as in Table 3.1.

Method	ℓ_1-norm	Rel. error ℓ_1-norm	ℓ_∞ error	$\mathcal{N}_A$
FISTA	5.71539e+7			
NESTA $\mu = 0.2$	5.71614e+7	1.3e-4	3.8	659
NESTA $\mu = 0.02$	5.71547e+7	1.4e-5	0.96	1055
NESTA $\mu = 0.002$	5.71540e+7	1.6e-6	0.64	1537

since the ℓ_∞ error is lower than 1. Observe in Figure 3.5 that this value separates the nonzero entries from the noise floor (when $\sigma = 0.01$). In the extensive numerical experiments of Section 3.5, we shall set $\mu = 0.02$ and $\delta = 10^{-7}$ as default values.

3.5 Numerical comparisons

This section presents numerical experiments comparing several state-of-the-art optimization techniques designed to solve (3.1.2) or (3.1.3). To be as fair as possible, we propose comparisons with methods for which software is publicly available online.

Some extensive numerical results have been reported in [vdBF08, WNF09, Lor09, LBDM$^+$09], but to the best of our knowledge, extensive tests comparing algorithms for solving (QP_λ) *and* (BP_ϵ) are currently unavailable, and one novelty of our experimental study is that it uses fair stopping criteria introduced in Section 3.5.3. Moreover, whereas publications sometimes test algorithms on relatively easy and academic problems, we will subject optimization methods to hard but realistic ℓ_1 reconstruction problems.

In our view, a challenging problem involves some or all of the characteristics below.

1. *High dynamic range.* As mentioned earlier, most optimization techniques are able to find (more or less rapidly) the most significant entries (those with a large amplitude) of the signal x. Recovering the entries of x that have low magnitudes accurately is more challenging. Methods that can accurately recover high-dynamic range signals is of crucial importance in some real-world applications, see Section 3.6.1.

2. *Approximate sparsity.* Realistic signals are seldom exactly sparse and, therefore, coping with approximately sparse signals is of paramount importance. In signal or image processing for example, wavelet coefficients of natural images contain lots of low level entries that are worth retrieving.

3. *Large scale.* Some standard optimization techniques, such as interior point methods, are known to provide accurate solutions. However, these techniques are not applicable to large-scale problems due to the large cost of solving linear systems. Further, many existing software

packages fail to take advantage of fast-algorithms for applying A. We will focus on large-scale problems in which the number of unknowns n is over a quarter of a million, i.e., $n = 262{,}144$.

3.5.1 State-of-the-art methods

Most of the algorithms discussed in this section are considered to be state-of-art in the sense that they are the most competitive among sparse reconstruction algorithms. Some of these methods have been improved after some years of research [KKB07, FNW07]. Others make careful use of the fact that ℓ_1 minimization produces sparse solutions, and activate special code when they estimate that a fixed active set has been found [vdBF08, WYGZ10]. Finally, our focus is on rapid algorithms so that we are interested in methods which can take advantage of fast algorithms for applying A to a vector. This is why we have not tested other good methods, such as [FHT10], for example.

Most of the following algorithms solve (QP_λ), and require gradient or sub-gradient information at the current iterate. Because of the quadratic penalty, the gradient requires one application of A and one of A^*, so the dominant cost per iteration is $2\mathcal{C}_A$.

3.5.1.1 NESTA

Below, we applied NESTA with the following default parameters

$$x_0 = A^*b, \quad \mu = 0.02, \quad \delta = 10^{-7}$$

(recall that x_0 is the initial guess). The maximal number of iterations is set to $\mathcal{I}_{\max} = 10{,}000$; if convergence is not reached after $\mathcal{I}_{\max}$ iterations, we record that the algorithm did not convergence (DNC). Because NESTA requires 2 calls to either A or A^* per iteration, this is equivalent to declaring DNC after $\mathcal{N}_A = 20{,}000$ iterations where $\mathcal{N}_A$ refers to the total number of calls to A or A^*; hence, for the other methods, we declare DNC when $\mathcal{N}_A > 20{,}000$. When continuation is used, extra parameters are set up as follows:

$$T = 4, \quad \mu_0 = \|x_0\|_{\ell_\infty}, \quad \gamma = (\mu/\mu_0)^{1/T},$$

and for $t = 1, \ldots, T$,

$$\mu_t = \gamma^t \mu_0, \quad \delta_t = 0.1 \cdot (\delta/0.1)^{t/T},$$

where δ_t fixes the stopping criterion in the t-th continuation step. Numerical results are reported and discussed in Section 3.5.4.

3.5.1.2 Gradient projections for sparse reconstruction (GPSR)

GPSR has been introduced in [FNW07] to solve the standard ℓ_1 minimization problem in Lagrangian form (QP_λ). GPSR uses a standard linear programming change-of-variables trick to recast the variable $x = v_1 - v_2$, with the requirement that $v_1, v_2 \geq 0$. This is a different problem, but it clearly has the same solution set. The advantage is that the ℓ_1 norm is replaced by a linear functional, so that the objective is now smooth. Projecting v_1 and v_2 onto the non-negative orthant is trivial. Different techniques for choosing the step-size α_k (backtracking, Barzilai-Borwein [BB88], and so on) are discussed in [FNW07]. The code is available at `http://www.lx.it.pt/~mtf/GPSR/`. In the forthcoming experiments, the parameters are set to their default values.

GPSR also implements continuation, and we test this version as well. All parameters were set to defaults except, per the recommendation of one of the GPSR authors to increase performance, the number of continuation steps was set to 40, the `ToleranceA` variable was set to 10^{-3}, and the `MiniterA` variable was set to 1. In addition, the code itself was tweaked a bit; in particular, the stopping criteria for continuation steps (other than the final step) was changed. Future releases of GPSR will probably contain a similarly updated continuation stopping criteria.

3.5.1.3 Sparse reconstruction by separable approximation (SpaRSA)

SpaRSA [WNF09] is an algorithm to minimize composite functions $\phi(x) = f(x) + \lambda c(x)$ composed of a smooth term f and a separable non-smooth term c, e.g., (QP_λ). At every step, a sub-problem of the form

$$\text{minimize} \quad \|x - y\|_{\ell_2}^2 + \frac{\lambda}{\alpha} c(x)$$

with optimization variable x must be solved; this is the same as computing the proximity operator corresponding to c. For (QP_λ), the solution is given by shrinkage. In this sense, SpaRSA is an iterative shrinkage/thresholding (IST) algorithm, much like FISTA (though without the accelerated convergence) and FPC. Also like FPC, continuation is used to speed convergence. Code for SpaRSA may be obtained at `http://www.lx.it.pt/~mtf/SpaRSA/`. Parameters were set to default except the number of continuation steps was set to 40 and the `MiniterA` variable was set to 1 (instead of the default 5), as per the recommendations of one of the SpaRSA authors—again, to increase performance.

3.5.1.4 ℓ_1 regularized least-squares (l1_ls)

This method [KKB07] solves the standard unconstrained ℓ_1 minimization problem, and is an interior point method (with log-barrier) using preconditioned conjugate gradient (PCG) to accelerate convergence and stabilize the algorithm. The preconditioner used in the PCG step is a linear combination of the diagonal approximation of the Hessian of the quadratic term and of the Hessian of

the log-barrier term. l1_ls is shown to be faster than usual interior point methods; nevertheless, each step requires solving a linear system of the form $H\Delta x = g$. Even if PCG makes the method more reliable, l1_ls is still problematic for large-scale problems. In the next comparisons, we provide some typical values of its computational complexity compared to the other methods. The code is available at `http://www.stanford.edu/~boyd/l1_ls/`.

3.5.1.5 Spectral projected gradient (SPGL1)

In 2008, van den Berg et al. [vdBF08] adapted the spectral projection gradient algorithm introduced in [BMR00] to solve the LASSO (LS_τ). Interestingly, they introduced a clever root finding procedure such that solving a few instances of (LS_τ) for different values of τ enables them to equivalently solve (BP_ϵ). Furthermore, if the algorithm detects a nearly-sparse solution, it defines an active set and solves an equation like (3.4.3) on this active set. In the next experiments, the parameters are set to their default values. The code is available at `http://www.cs.ubc.ca/labs/scl/SPGL11/`.

3.5.1.6 Fixed point continuation method (FPC)

The fixed point continuation method [HYZ07, HYZ08] is a recent first-order algorithm for solving (QP_λ) and simple generalizations of (QP_λ). The main idea is based on a fixed point equation, $x = F(x)$, which holds at the solution (derived from the subgradient optimality condition, where F is a composition of shrinkage and a gradient step). For appropriate parameters, F is a contraction, and thus the algorithm $x_{k+1} = F(x_k)$ converges (q-linearly). The parameter λ in (QP_λ) determines the amount of shrinkage and, therefore, the speed of convergence; thus in practice, λ is decreased in a continuation scheme. Code for FPC is available at `http://www.caam.rice.edu/~optimization/L1/fpc/`. Also available is a state-of-the-art version of FPC from 2008 that uses Barzilai-Borwein [BB88] steps to accelerate performance. In the numerical tests, the Barzilai-Borwein version (referred to as FPC-BB) significantly outperforms standard FPC. All parameters were set to default values.

3.5.1.7 FPC active set (FPC-AS)

In 2009, inspired by both first-order algorithms, such as FPC, and greedy algorithms [DTDS06, NT09], Wen et al. [WYGZ10] extended FPC into the two-part algorithm FPC active set to solve (QP_λ). In the first stage, FPC-AS calls an improved version of FPC that allows the step-size to be updated dynamically, using a non-monotone exact line search to ensure r-linear convergence, and also incorporating a Barzilai-Borwein [BB88] heuristic. After a given stopping criterion, the current value, x_k, is hard-thresholded to determine an active set. On the active set, $\|x\|_{\ell_1}$ is replaced by c^*x, where $c = \text{sgn}(x_k)$, with the constraints that $x[i] \cdot c[i] > 0$ for all the indices i belonging to the active set. This sub-problem has a smooth objective, so it can be solved using smooth optimization techniques. This two-step process is then repeated for a smaller value of λ in a continuation scheme. We tested

FPC-AS using both L-BFGS (the default) and CG (which we refer to as FPC-AS-CG) to solve the sub-problem; both of these solvers do not actually enforce the $x[i] \cdot c[i] > 0$ constraint on the active set. Code for FPC-AS is available at `http://www.caam.rice.edu/~optimization/L1/FPC_AS/`.

For s-sparse signals, all parameters were set to defaults except for the stopping criteria (as discussed in Section 3.5.3). For approximately sparse signals, FPC-AS performed poorly ($> 10,000$ iterations) with the default parameters. By changing a parameter that controls the *estimated* number of nonzeros from $m/2$ (default) to n, the performance improved dramatically, and this is the performance reported in the tables. The maximum number of subspace iterations was also changed from the default to 10, as recommended in the help file.

3.5.1.8 Bregman

The Bregman iterative algorithm, motivated by the Bregman distance, has been shown to be surprisingly simple [YOGD08]. The first iteration solves (QP_λ) for a specified value of λ; subsequent iterations solve (QP_λ) for the same value of λ, with an updated observation vector b. Typically, only a few outer iterations are needed (e.g., 4), but each iteration requires a solve of (QP_λ), which is costly. The original Bregman algorithm calls FPC to solve these sub-problems; we test Bregman using FPC and FPC-BB as sub-problem solvers.

A version of the Bregman algorithm, known as the linearized Bregman algorithm [OMDY10, COS09], takes only one step of the inner iteration per outer iteration; consequently, many outer iterations are taken, in contrast to the regular Bregman algorithm. It can be shown that linearized Bregman is equivalent to gradient ascent on the dual problem. Linearized Bregman was not included in the tests because no standardized public code is available. Code for the regular Bregman algorithm may be obtained at `http://www.caam.rice.edu/~optimization/L1/2006/10/bregman-iterative-algorithms-for.html`. There are quite a few parameters, since there are parameters for the outer iterations and for the inner (FPC) iterations; for all experiments, parameters were set to defaults. In particular, we noted that using the default stopping criteria for the inner solve, which limited FPC to 1,000 iterations, led to significantly better results than allowing the sub-problem to run to 10,000 iterations.

3.5.1.9 Fast iterative soft-thresholding algorithm (FISTA)

FISTA [BT09] is based upon Nesterov's work but departs from NESTA in two important ways: 1) FISTA solves the sparse unconstrained reconstruction problem (QP_λ); 2) FISTA is a proximal subgradient algorithm, which only uses two sequences of iterates. In some sense, FISTA is a simplified version of the algorithm previously introduced by Nesterov to minimize composite functions [Nes07]. The theoretical rate of convergence of FISTA is $\mathcal{O}(1/k^2)$.

For each test, FISTA is run twice: it is first run until the relative variation in the function value

is less than 10^{-14}, with no limit on function calls, and this solution is used as the reference solution. The second time, it is run using the same stopping criteria as the other algorithms, as explained in Section 3.5.3.

3.5.2 Constrained versus unconstrained minimization

We would like to briefly highlight the fact that these algorithms are not solving the same problem. SPGL1 solves the constrained problem (BP_ϵ) and NESTA solves an approximate solution to (BP_ϵ), while all other methods tested solve the unconstrained problem (QP_λ). NESTA and SPGL1 can also solve (QP_λ). Solving a constrained problem may sometimes be more challenging than similar unconstrained problems (witness the popularity of penalty functions and augmented Lagrangian methods), and given the numerous algorithms to solve (QP_λ) and the relatively few algorithms to solve (BP_ϵ), it seems empirically that (BP_ϵ) is the harder problem.[7] For example, it may be hard to even find a feasible point for (BP_ϵ), since the pseudo-inverse of A, when A is not a projection, may be difficult to compute. Thus, we emphasize that SPGL1 and NESTA are actually more general than the other algorithms (and as Section 3.6 shows, NESTA is even more general because it handles a wide variety of constrained problems); this is especially important because from a practical viewpoint, it may be easier to estimate an appropriate ϵ than an appropriate value of λ. Furthermore, as will be shown in Section 3.5.4, SPGL1 and NESTA with continuation are also the most robust methods for arbitrary signals (i.e., they perform well even when the signal is not exactly sparse, and even when it has high dynamic range). Combining these two facts, we feel that these two algorithms are extremely useful for real-world applications.

3.5.3 Experimental protocol

In these experiments, we compare NESTA with other efficient methods. There are two main difficulties with comparisons which might explain why broad comparisons have not been offered before. The first problem is that some algorithms, such as NESTA, solve (BP_ϵ), whereas other algorithms solve (QP_λ). Given ϵ, it is difficult to compute $\lambda(\epsilon)$ that gives an equivalence between the problems; in theory, the KKT conditions give λ, but we have observed in practice that because we have an approximate solution (albeit a very accurate one), computing λ in this fashion is not stable.

Instead, we note that given λ and a solution x_λ to (QP_λ), it is easy to compute a very accurate $\epsilon(\lambda)$ since $\epsilon = \|Ax_\lambda - b\|_{\ell_2}$. Hence, we use a two-step procedure. In the first step, we choose a value of $\epsilon_0 = \sqrt{m + 2\sqrt{2m}\sigma}$ based on the noise level σ (since a value of λ that corresponds to σ is less clear), and use SPGL1 to solve (BP_ϵ). From the SPGL1 dual solution, we have an estimate

[7] Standard texts explain that (BP_ϵ) is equivalent to minimizing $\|x\|_{\ell_1} + \chi_{\mathcal{Q}_p}(x)$ where $\mathcal{Q}_p$ is the feasible set $\{x : \|Ax - b\|_{\ell_2} \leq \epsilon\}$, and $\chi_{\mathcal{Q}_p}(x) = 0$ if $x \in \mathcal{Q}_p$ and $+\infty$ otherwise. Hence, the unconstrained problem has a discontinuous objective functional.

of $\lambda = \lambda(\epsilon_0)$. As noted above, this equivalence may not be very accurate, so the second step is to compute $\epsilon_1 = \epsilon(\lambda)$ via FISTA, using a very high accuracy of $\delta = 10^{-14}$. The pair (λ, ϵ_1) now leads to nearly equivalent solutions of (QP_λ) and (BP_ϵ). The solution from FISTA will also be used to judge the accuracy of the other algorithms.

The other main difficulty in comparisons is a fair stopping criterion. Each algorithm has its own stopping criterion (or may offer a choice of stopping criteria), and these are not directly comparable. To overcome this difficulty, we have modified the codes of the algorithms to allow for two new stopping criterion that we feel are the only fair choices. The short story is that we use NESTA to compute a solution x_N and then ask the other algorithms to compute a solution that is at least as accurate.

Specifically, given NESTA's solution x_N (using continuation), the other algorithms terminate at iteration k when the solution $\hat{x}_k$ satisfies

$$\text{(Crit. 1)} \quad \|\hat{x}_k\|_{\ell_1} \leq \|x_N\|_{\ell_1} \quad \text{and} \quad \|b - A\hat{x}_k\|_{\ell_2} \leq 1.05\,\|b - Ax_N\|_{\ell_2}, \tag{3.5.1}$$

or

$$\text{(Crit. 2)} \quad \lambda\|\hat{x}_k\|_{\ell_1} + \frac{1}{2}\|A\hat{x}_k - b\|_{\ell_2}^2 \leq \lambda\|x_N\|_{\ell_1} + \frac{1}{2}\|Ax_N - b\|_{\ell_2}^2. \tag{3.5.2}$$

We run tests with both stopping criteria to reduce any potential bias from the fact that some algorithms solve (QP_λ), for which Crit. 2 is the most natural, while others solve (BP_ϵ), for which Crit. 1 is the most natural. In practice, the results when applying Crit. 1 or Crit. 2 are not significantly different.

3.5.4 Numerical results

3.5.4.1 The case of exactly sparse signals

This first series of experiments tests all the algorithms discussed above in the case where the unknown signal is s-sparse with $s = m/5$, $m = n/8$, and $n = 262{,}144$. This situation is close to the limit of perfect recovery from noiseless data. The s nonzero entries of the signals x^0 are generated as described in (3.3.15). Reconstruction is performed with several values of the dynamic range $d = 20, 40, 60, 80, 100$ in dB. The measurement operator is a randomly subsampled discrete cosine transform, as in Section 3.4.1 (with a different random set of measurements chosen for each trial). The noise level is set to $\sigma = 0.1$. The results are reported in Tables 3.3 (Crit. 1) and 3.4 (Crit. 2); each cell in these tables contains the mean value of $\mathcal{N}_A$ (the number of calls of A or A^*) over 10 random trials, and, in smaller font, the minimum and maximum value of $\mathcal{N}_A$ over the 10 trials. When convergence is not reached after $\mathcal{N}_A = 20{,}000$, we report DNC (did not converge). As expected, the number of calls needed to reach convergence varies a lot from an algorithm to another.

Table 3.3: Number of function calls $\mathcal{N}_A$ averaged over 10 independent runs. The sparsity level $s = m/5$ and the stopping rule is Crit. 1 (3.5.1).

Method	20 dB	40 dB	60 dB	80 dB	100 dB
NESTA	446 351/491	880 719/951	1701 1581/1777	4528 4031/4749	14647 7729/15991
NESTA + Ct	479 475/485	551 539/559	605 589/619	658 635/679	685 657/705
GPSR	56 44/62	733 680/788	5320 4818/5628	DNC	DNC
GPSR + Ct	305 293/311	251 245/257	497 453/531	1816 1303/2069	9101 7221/10761
SpaRSA	345 327/373	455 435/469	542 511/579	601 563/629	708 667/819
SPGL1	54 37/61	128 102/142	209 190/216	354 297/561	465 380/562
FISTA	68 66/69	270 261/279	935 885/969	3410 2961/3594	13164 11961/13911
FPC AS	156 111/177	236 157/263	218 215/239	351 247/457	325 313/335
FPC AS (CG)	312 212/359	475 301/538	434 423/481	641 470/812	583 567/595
FPC	414 394/436	417 408/422	571 546/594	945 852/1038	3945 2018/4734
FPC-BB	148 140/152	166 158/168	219 208/250	264 252/282	520 320/800
Bregman-BB	211 203/225	270 257/295	364 355/393	470 429/501	572 521/657

Table 3.4: Number of function calls $\mathcal{N}_A$ averaged over 10 independent runs. The sparsity level $s = m/5$ and the stopping rule is Crit. 2 (3.5.2).

Method	20 dB	40 dB	60 dB	80 dB	100 dB
NESTA	446 351/491	880 719/951	1701 1581/1777	4528 4031/4749	14647 7729/15991
NESTA + Ct	479 475/485	551 539/559	605 589/619	658 635/679	685 657/705
GPSR	59 44/64	736 678/790	5316 4814/5630	DNC	DNC
GPSR + Ct	305 293/311	251 245/257	511 467/543	1837 1323/2091	9127 7251/10789
SpaRSA	345 327/373	455 435/469	541 509/579	600 561/629	706 667/819
SPGL1	55 37/61	138 113/152	217 196/233	358 300/576	470 383/568
FISTA	65 63/66	288 279/297	932 882/966	3407 2961/3591	13160 11955/13908
FPC AS	176 169/183	236 157/263	218 215/239	344 247/459	330 319/339
FPC AS (CG)	357 343/371	475 301/538	434 423/481	622 435/814	588 573/599
FPC	416 398/438	435 418/446	577 558/600	899 788/962	3866 1938/4648
FPC-BB	149 140/154	172 164/174	217 208/254	262 248/286	512 308/790
Bregman-BB	211 203/225	270 257/295	364 355/393	470 429/501	572 521/657

The careful reader will notice that Tables 3.3 and 3.4 do not feature the results provided by l1_ls; indeed, while it seems faster than other interior point methods, it is still far from being comparable to the other algorithms reviewed here. In these experiments l1_ls typically needed 1500 calls to A or A^* for reconstructing a 20 dB signal with $s = m/100$ nonzero entries. For solving the same problem with a dynamic range of 100 dB, it took 5 hours to converge on a dual core MacPro G5 clocked at 2.7GHz.

GPSR performs well in the case of low-dynamic range signals; its performance, however, decreases dramatically as the dynamic range increases; Table 3.4 shows that it does not converge for 80 and 100 dB signals. GPSR with continuation does worse on the low dynamic range signals (which is not surprising). It does much better than the regular GPSR version on the high dynamic range signals, though it is slower than NESTA with continuation by more than a factor of 10. SpaRSA performs well at low dynamic range, comparable to NESTA, and begins to outperform GSPR with continuation as the dynamic range increases, although it begins to underperform NESTA with continuation in this regime. SpaRSA takes over twice as many function calls on the 100 dB signal as on the 20 dB signal.

SPGL1 shows good performance with very sparse signals and low dynamic range. Although it has fewer iteration counts than NESTA, the performance decreases much more quickly than for

NESTA as the dynamic range increases; SPGL1 requires about 9× more calls to A at 100 dB than at 20 dB, whereas NESTA with continuation requires only about 1.5× more calls. FISTA is almost as fast as SPGL1 on the low dynamic range signal, but degrades very quickly as the dynamic range increases, taking about 200× more iterations at 100 dB than at 20 dB. One large contributing factor to this poor performance at high dynamic range is the lack of a continuation scheme.

FPC performs well at low dynamic range, but is very slow on 100 dB signals. The Barzilai-Borwein version was consistently faster than the regular version, but also degrades much faster than NESTA with continuation as the dynamic range increases. Both FPC active set and the Bregman algorithm perform well at all dynamic ranges, but again, degrade faster than NESTA with continuation as the dynamic range increases. There is a slight difference between the two FPC active set versions (using L-BFGS or CG), but the dependence on the dynamic range is roughly similar.

The performances of NESTA with continuation are reasonable when the dynamic range is low. When the dynamic range increases, continuation becomes much more helpful. For some 100 dB signals, using continuation reduces the number of calls by a factor of 20. In these experiments, the tolerance δ is consistently equal to 10^{-7}; while this choice is reasonable when the dynamic range is high, it seems too conservative in the low dynamic range case. Setting a lower value of δ should improve NESTA's performance in this regime. In other words, NESTA with continuation might be tweaked to run faster on the low dynamic range signals. However, this is not in the spirit of this chapter and this is why we have not researched further refinements.

In summary, for exactly sparse signals exhibiting a significant dynamic range, 1) the performance of NESTA with continuation—but otherwise applied out-of-the-box—is comparable to that of state-of-the-art algorithms, and 2) most state-of-the-art algorithms are efficient on these types of signals.

3.5.4.2 Approximately sparse signals

We now turn our attention to approximately sparse signals. Such signals are generated via a permutation of the Haar wavelet coefficients of a 512×512 natural image. The data b are $m = n/8 = 32{,}768$ discrete cosine measurements selected at random. White Gaussian noise with standard deviation $\sigma = 0.1$ is then added. Each test is repeated 5 times, using a different random permutation every time (as well as a new instance of the noise vector). Unlike in the exactly sparse case, the wavelet coefficients of natural images mostly contain mid-range and low-level coefficients (see Figure 3.6) which are challenging to recover.

The results are reported in Tables 3.5 (Crit. 1) and 3.6 (Crit. 2); the results from applying the two stopping criteria are nearly identical. In these series of experiments, the performance of SPGL1 is quite good but seems to vary a lot from one trial to another (Table 3.6). Notice that the concept of an active set is ill defined in the approximately sparse case; as a consequence, the active-set version of FPC is not much of an improvement over the regular FPC version. FPC is very fast for

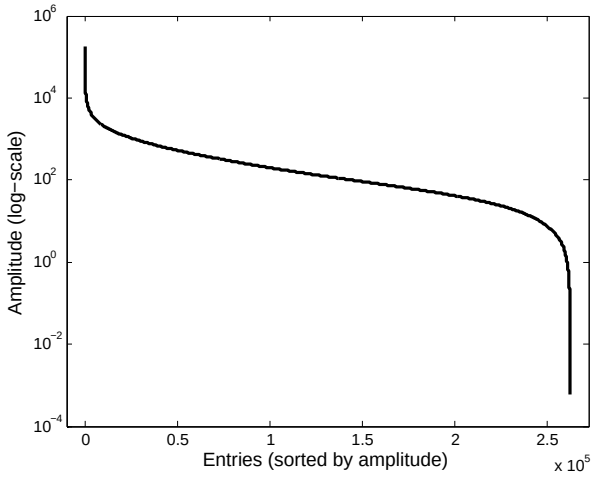

Figure 3.6: Sorted wavelet coefficients of the natural image used in the experiments

s-sparse signals but lacks the robustness to deal with less ideal situations in which the unknown is only approximately sparse.

FISTA and SpaRSA converge for these tests, but are not competitive with the best methods. It is reasonable to assume that FISTA would also improve if implemented with continuation. SpaRSA already uses continuation but does not match its excellent performance on exactly sparse signals.

Bregman, SPGL1, and NESTA with continuation all have excellent performances (continuation really helps NESTA) in this series of experiments. NESTA with continuation seems very robust when high accuracy is required. The main distinguishing feature of NESTA is that it is less sensitive to dynamic range; this means that as the dynamic range increases, or as the noise level σ decreases, NESTA becomes very competitive. For example, when the same test was repeated with more noise ($\sigma = 1$), all the algorithms converged faster. In moving from $\sigma = 1$ to $\sigma = 0.1$, SPGL1 required 90% more iterations and Bregman required 20% more iterations, while NESTA with continuation required only 5% more iterations.

One conclusion from these tests is that SPGL1, Bregman, and NESTA (with continuation) are the only methods dealing with approximately sparse signals effectively. The other methods, most of which did very well on exactly sparse signals, take over 10,000 function calls or even do not converge in 20,000 function calls; by comparison, SPGL1, Bregman, and NESTA with continuation converge in about 2,000 function calls. It is also worth noting that Bregman is only as good as the sub-problem solver; though not reported here, using the regular FPC (instead of FPC-BB) with Bregman leads to much worse performance.

Table 3.5: Recovery results of an approximately sparse signal with Crit. 1 as a stopping rule

Method	$< \mathcal{N}_A >$	$\min \mathcal{N}_A$	$\max \mathcal{N}_A$
NESTA	18912	18773	19115
NESTA + Ct	2667	2603	2713
GPSR	DNC	DNC	DNC
GPSR + Ct	DNC	DNC	DNC
SpaRSA	10019	8369	12409
SPGL1	1776	1073	2464
FISTA	10765	10239	11019
FPC Active Set	DNC	DNC	DNC
FPC Active Set (CG)	DNC	DNC	DNC
FPC	DNC	DNC	DNC
FPC-BB	DNC	DNC	DNC
Bregman-BB	2045	2045	2045

Table 3.6: Recovery results of an approximately sparse signal with Crit. 2 as a stopping rule

Method	$< \mathcal{N}_A >$	$\min \mathcal{N}_A$	$\max \mathcal{N}_A$
NESTA	18912	18773	19115
NESTA + Ct	2667	2603	2713
GPSR	DNC	DNC	DNC
GPSR + Ct	DNC	DNC	DNC
SpaRSA	10021	8353	12439
SPGL1	1776	1073	2464
FISTA	10724	10197	10980
FPC Active Set	DNC	DNC	DNC
FPC Active Set (CG)	DNC	DNC	DNC
FPC	DNC	DNC	DNC
FPC-BB	DNC	DNC	DNC
Bregman-BB	2045	2045	2045

The algorithms which did converge all achieved a mean relative ℓ_1 error (using (3.4.1) and the high accuracy FISTA solution as the reference) less than $2 \cdot 10^{-4}$ and sometimes as low as 10^{-5}, except SPGL1, which had a mean relative error of $1.1 \cdot 10^{-3}$. Of the algorithms that did not converge in 20,000 function calls, FPC and FPC-BB had a mean ℓ_1 relative error about $5 \cdot 10^{-3}$, GPSR with continuation had errors about $5 \cdot 10^{-2}$, and the rest had errors greater than 10^{-1}.

3.6 An all-purpose algorithm

A distinguishing feature is that NESTA is able to cope with a wide range of standard regularizing functions. In this section, we present two examples: non-standard ℓ_1 minimization and total-variation minimization.

3.6.1 Non-standard sparse reconstruction: ℓ_1 analysis

Suppose we have a signal $x \in \mathbb{R}^n$, which is assumed to be approximately sparse in a transformed domain such as the wavelet, the curvelet or the time-frequency domains. Let W be the corresponding

synthesis operator whose columns are the waveforms we use to synthesize the signal $x = W\alpha$ (real-world signals do not admit an exactly sparse expansion); e.g., the columns may be wavelets, curvelets, and so on, or both. We will refer to W^* as the analysis operator. As before, we have (possibly noisy) measurements $b = Ax^0 + z$. The *synthesis* approach attempts reconstruction by solving

$$\begin{aligned} &\text{minimize} && \|\alpha\|_{\ell_1} \\ &\text{subject to} && \|b - AW\alpha\|_{\ell_2} \leq \epsilon, \end{aligned} \tag{3.6.1}$$

while the *analysis* approach solves the related problem

$$\begin{aligned} &\text{minimize} && \|W^*x\|_{\ell_1} \\ &\text{subject to} && \|b - Ax\|_{\ell_2} \leq \epsilon. \end{aligned} \tag{3.6.2}$$

If W is orthonormal, the two problems are equivalent, but in general, these give distinct solutions and current theory explaining the differences is still in its infancy. The article [EMR07] suggests that synthesis may be overly sensitive, and argues with geometric heuristics and numerical simulations that analysis is sometimes preferable.

Solving ℓ_1-analysis problems with NESTA is straightforward as only Step 1 needs to be adapted. We have

$$f_\mu(x) = \max_{u \in Q_p} \langle u, W^*x \rangle - \frac{\mu}{2}\|u\|_{\ell_2}^2,$$

and the gradient at x is equal to

$$\nabla f_\mu(x) = W u_\mu(x);$$

here, $u_\mu(x)$ is given by

$$(u_\mu(x))[i] = \begin{cases} \mu^{-1}(W^*x)[i], & \text{if } |(W^*x)[i]| < \mu, \\ \text{sgn}((W^*x)[i]), & \text{otherwise.} \end{cases}$$

Steps 2 and 3 remain unchanged. The computational complexity of the algorithm is then increased by an extra term, namely $2\,\mathcal{C}_W$ where $\mathcal{C}_W$ is the cost of applying W or W^* to a vector. In practical situations, there is often a fast algorithm for applying W and W^*, e.g. a fast wavelet transform [Mal08], a fast curvelet transform [CDDY06], a fast short-time Fourier transform [Mal08] and so on, which makes this a low-cost extra step[8].

[8] The ability to solve the analysis problem also means that NESTA can easily solve reweighted ℓ_1 problems [CWB08] with no change to the code.

3.6.2 Numerical results for non-standard ℓ_1 minimization

Because NESTA is one of very few algorithms that can solve both the analysis and synthesis problems efficiently, we tested the performance of both analysis and synthesis on a simulated real-world signal from the field of radar detection. The test input is a superposition of three signals. The first signal, which is intended to make recovery more difficult for any smaller signals, is a plain sinusoid with amplitude of 1000 and frequency near 835 MHz.

A second signal, similar to a Doppler pulse radar, is at a carrier frequency of 2.33 GHz with maximum amplitude of 10, a pulse width of 1 μs and a pulse repetition interval of 10 μs; the pulse envelope is trapezoidal, with a 10 ns rise time and 40 ns fall time. This signal is more than 40 dB lower than the pure sinusoid, since the maximum amplitude is 100$\times$ smaller, and since the radar is nonzero only 10% of the time. The Doppler pulse was chosen to be roughly similar to a realistic weather Doppler radar. In practice, these systems operate at 5 cm or 10 cm wavelengths (i.e., 6 or 3 GHz) and send out short trapezoidal pulses to measure the radial velocity of water droplets in the atmosphere using the Doppler effect.

The third signal, which is the signal of interest, is a frequency-hopping radar pulse with maximum amplitude of 1 (so about 20 dB beneath the Doppler signal, and more than 60 dB below the sinusoid). For each instance of the pulse, the frequency is chosen uniformly at random from the range 200 MHz to 2.4 GHz. The pulse duration is 2 μs and the pulse repetition interval is 22 μs, which means that some, but not all, pulses overlap with the Doppler radar pulses. The rise time and fall time of the pulse envelope are comparable to the Doppler pulse. Frequency-hopping signals may arise in applications because they can be more robust to interference and because they can be harder to intercept. When the carrier frequencies are not known to the listener, the receiver must be designed to cover the entire range of possible frequencies (2.2 GHz in our case). While some current analog-to-digital converters (ADC) may be capable of operating at 2.2 GHz, they do so at the expense of low precision. Hence this situation may be particularly amenable to a compressed sensing setup by using several slower (but accurate) ADC to cover a large bandwidth.

We consider the exact signal to be the result of an infinite-precision ADC operating at 5 GHz, which corresponds to the Nyquist rate for signals with 2.5 GHz of bandwidth. Measurements are taken using an orthogonal Hadamard transform with randomly permuted columns, and these measurements were subsequently sub-sampled by randomly choosing $m = .3n$ rows of the transform (so that we undersample Nyquist by 10/3). Samples are recorded for $T = 209.7\mu$s, which corresponds to $n = 2^{20}$. White noise was added to the measurements to make a 60 dB signal-to-noise ratio (SNR) (note that the effective SNR for the frequency-hopping pulse is much lower). The frequencies of the sinusoid and the Doppler radar were chosen such that they were not integer multiples of the lowest recoverable frequency $f_{min} = 1/(2T)$.

For reconstruction, the signal is analyzed with a tight frame of Gabor atoms that is approxi-

mately 5.5× overcomplete. The particular parameters of the frame are chosen to give reasonable reconstruction, but were not tweaked excessively. It is likely that differences in performance between analysis and synthesis are heavily dependent on the particular dictionary.

To analyze performance, we restrict our attention to the frequency domain in order to simplify comparisons. The top plot in Figure 3.7 shows the frequency components of the original, noiseless signal. The frequency-hopping pulse barely shows up since the amplitude is 1000× smaller than the sinusoid and since each frequency only occurs for 1 μs (of 210 μs total).

The bottom plots in Figure 3.7 show the spectrum of the recovered signal using analysis and synthesis, respectively. For this test, analysis does a better job at finding the frequencies belonging to the small pulse, while synthesis does a better job recreating the large pulse and the pure tone. The two reconstructions used slightly different values of μ to account for the redundancy in the size of the dictionary; otherwise, algorithm parameters were the same. In the analysis problem, NESTA took 231 calls to the analysis/synthesis operator (and 231 calls to the Hadamard transform); for synthesis, NESTA took 1378 calls to the analysis/synthesis operator (and 1378 to the Hadamard transform). With NESTA, synthesis is more computationally expensive than analysis since no change of variables trick can be done; in the synthesis case, W and W^* are used in Step 2 and 3 while in the analysis case, the same operators are used once in Step 1 (this is accomplished by the previously mentioned change-of-variables for partial orthogonal measurements).

As emphasized in [EMR07], when W is overcomplete, the solution computed by solving the analysis problems is likely to be denser than in the synthesis case. In plain English, the analysis solution may seem "noisier" than the synthesis solution. But the compactness of the solution of the synthesis problem may also be its weakness: an error on one entry of α may lead to a solution that differs a lot. This may explain why the frequency-hopping radar pulse is harder to recover with the synthesis prior.

Because all other known first-order methods solve only the synthesis problem, NESTA may prove to be extremely useful for real-world applications. Indeed, this simple test suggests that analysis may sometimes be much preferable to synthesis, and given a signal with 2^{20} samples (too large for interior point methods), we know of no other algorithm that can return the same results.

3.6.3 Total-variation minimization

Nesterov's framework also makes total-variation minimization possible. The TV norm of a 2D digital object $x[i,j]$ is given by

$$\|x\|_{TV} := \sum_{i,j} \|\nabla x[i,j]\|, \quad \nabla x[i,j] = \begin{bmatrix} (D_1 x)[i,j] \\ (D_2 x)[i,j] \end{bmatrix},$$

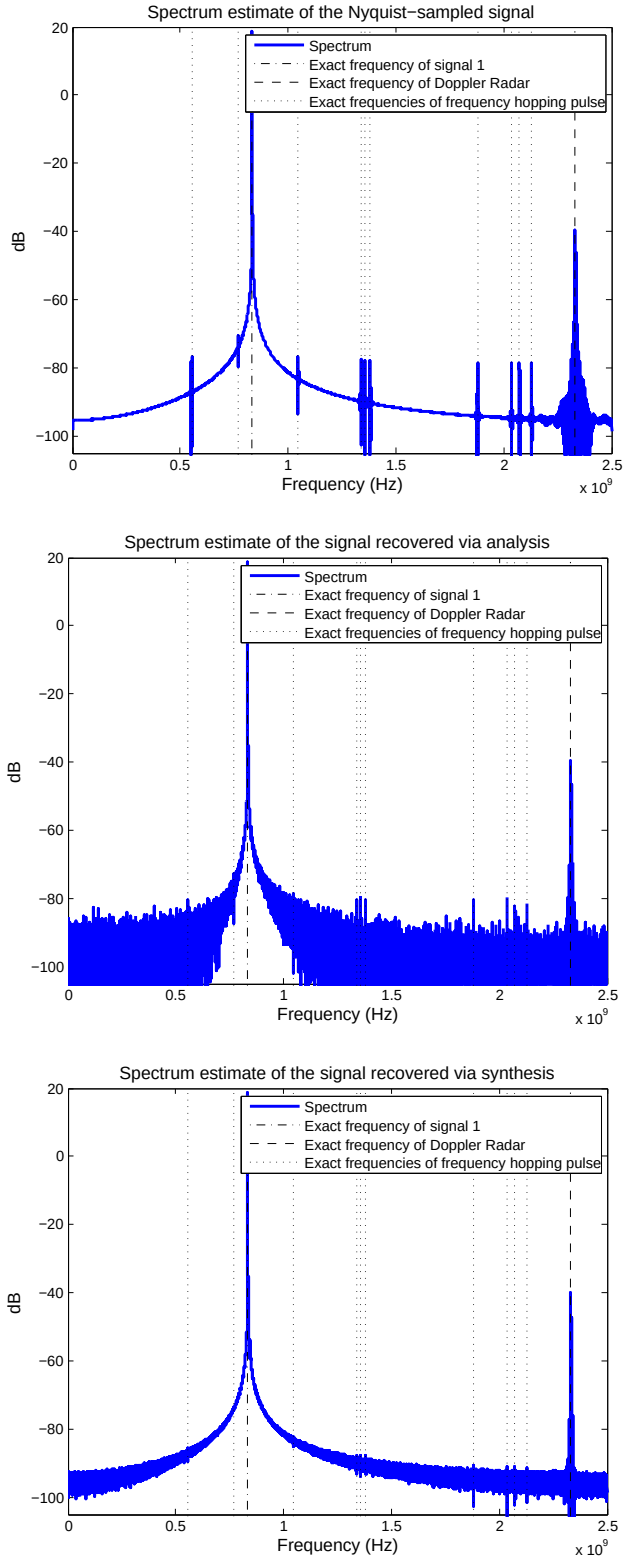

Figure 3.7: **Top:** spectrum estimate of the exact signal, no noise. The pure tone at 60 dB and the Doppler radar at 20 dB dominate the 0 dB frequency-hopping pulses. **Middle:** spectrum estimate of the recovered signal using analysis prior, with 60 dB SNR. The spectrum appears noisy, but the frequency-hopping pulses stand out. **Bottom:** spectrum estimate of the recovered signal using synthesis prior, with 60 dB SNR. The spectrum appears cleaner, but the small 0 dB pulses do not appear.

where D_1 and D_2 are the horizontal and vertical differences

$$(D_1 x)[i,j] = x[i+1,j] - x[i,j],$$
$$(D_2 x)[i,j] = x[i,j+1] - x[i,j].$$

Now the TV norm can be expressed as follows:

$$\|x\|_{TV} = \max_{u \in \mathcal{Q}_d} \langle u, Dx \rangle, \tag{3.6.3}$$

where $u = [u_1, u_2]^* \in \mathcal{Q}_d$ if and only for each (i,j), $u_1^2[i,j] + u_2^2[i,j] \leq 1$, and $D = [D_1, D_2]^*$. The key feature of Nesterov's work is to smooth a well-structured non-smooth function as follows (notice in (3.6.3) the similarity between the TV norm and the ℓ_1 norm):

$$\max_{u \in \mathcal{Q}_d} \langle u, Dx \rangle - \mu p_d(u).$$

Choosing $p_d(u) = \frac{1}{2}\|u\|_{\ell_2}^2$ provides a reasonable prox-function that eases the computation of ∇f_μ. Just as before, changing the regularizing function only modifies Step 1 of NESTA. Here,

$$f_\mu(x) = \max_{u \in \mathcal{Q}_d} \langle u, Dx \rangle - \frac{\mu}{2}\|u\|_{\ell_2}^2.$$

Then as usual,

$$\nabla f_\mu(x) = D^* u^{(\mu)}(x),$$

where $u^{(\mu)}(x)$ is of the form $[u_1^{(\mu)}, u_2^{(\mu)}]^*$ and for each $a \in \{1,2\}$,

$$u_a^{(\mu)}[i,j] = \begin{cases} \mu^{-1}(D_a x)[i,j], & \text{if } \|\nabla x[i,j]\| < \mu, \\ \|\nabla x[i,j]\|^{-1}(D_a x)[i,j], & \text{otherwise.} \end{cases}$$

The application of D and D^* leads to a negligible computational cost (sparse matrix-vector multiplications).

3.6.4 Numerical results for TV minimization

We are interested in solving

$$\begin{array}{ll} \text{minimize} & \|x\|_{TV} \\ \text{subject to} & \|b - Ax\|_{\ell_2} \leq \epsilon. \end{array} \tag{3.6.4}$$

To be sure, a number of efficient TV-minimization algorithms have been proposed to solve (3.6.4) in the special case $A = I$ (denoising problem), see [Cha04,DS05,GO09,ZC08]. In comparison, only a few

methods have been proposed to solve the more general problem (3.6.4) even when A is a projector. Known methods include interior point methods (ℓ_1-magic) [CR07b], proximal-subgradient methods [BDF07, CP08], Split-Bregman [GO09], and the very recently introduced RecPF[9] [YZY10], which operates in the special case of partial Fourier measurements. Applications of some of Nesterov's works have also been considered in different settings [WBFA09, DHJJ10, Auj09].

Again, we compare NESTA with algorithms with publicly available implementations. Therefore, the two algorithms we tested are RecPF (the newest member of the family) and TwIST [BDF07]. (ℓ_1-magic is based on an interior point method, and is not yet applicable to this large-scale problem.)

Evaluations are made by comparing the performances of NESTA (with continuation), RecPF, and TwIST on a set of images composed of random squares. As in Section 3.5, the dynamic range of the signals (amplitude of the squares) varies in a range from 20 to 40 dB. The size of each image x is 1024×1024; one of these images is displayed in the top panel of Figure 3.8. The data b are partial Fourier measurements as in [CRT06]; the number of measurements $m = n/10$. White Gaussian noise of standard deviation $\sigma = 0.1$ is added. The parameters of NESTA are set up as follows:

$$x_0 = A^* b, \quad \mu = 0.2, \quad \delta = 10^{-5}, \quad T = 5,$$

and the initial value of μ is

$$\mu_0 = 0.9 \max_{ij} \|\nabla x_0[i,j]\|.$$

The maximal number of iterations is set to $\mathcal{I}_{\max} = 4{,}000$. As it turns out, TV minimization from partial Fourier measurements is of significant interest in the field of magnetic resonance imaging (MRI) [LDP07].

As discussed above, RecPF has been designed to solve TV-minimization reconstruction problems from partial Fourier measurements. We set the parameters of RecPF to their default values except for the parameter `tol_rel_inn` that is set to 10^{-5}. TwIST calls Chambolle's algorithm [Cha04] at each iteration to compute the proximity operator of TV; the maximum number of iterations in Chambolle's algorithm is set to 10. With these parameter selections, TwIST and RecPF converge to a solution close enough to NESTA's output—at least in our experiments. Figure 3.8 shows the the solution computed by RecPF (top right), TwIST (bottom left), and NESTA (bottom left).

The curves in Figure 3.9 show the number of calls to A or A^*; mid-points are averages over 5 random trials, with error bars indicating the minimum and maximum number of calls. Here, RecPF

[9] Available at `http://www.caam.rice.edu/~optimization/L1/RecPF/`

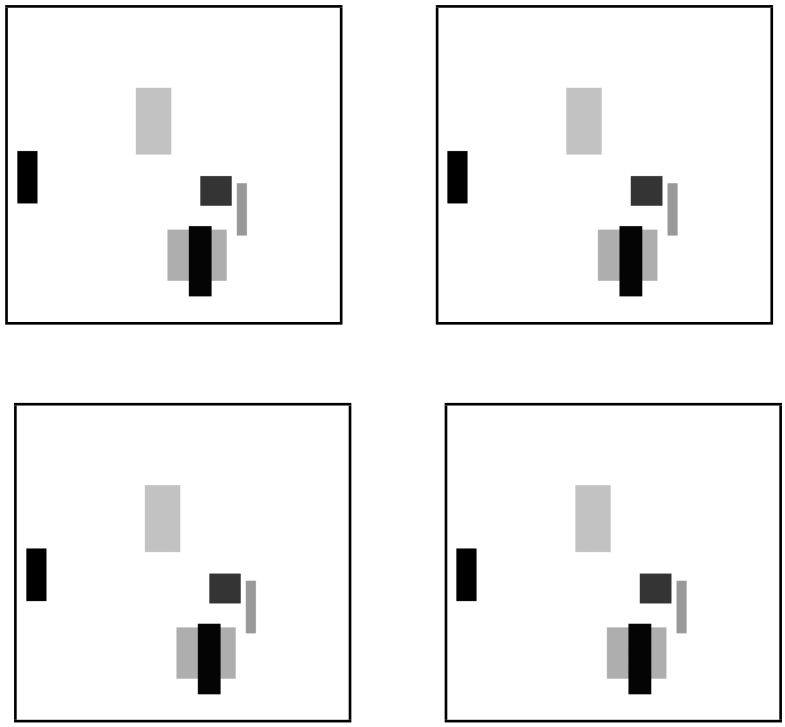

Figure 3.8: Top-Left: original image of size 1024×1024 with a dynamic range of about 40 dB. Top-Right: RecPF solution. Bottom-Left: TwIST solution. Bottom-Right: NESTA solution

is stopped when

$$\|x_{\text{RecPF}}\|_{TV} \leq 1.05\|x_N\|_{TV},$$
$$\|b - Ax_{\text{RecPF}}\|_{\ell_2} \leq 1.05\|b - Ax_N\|_{\ell_2},$$

where x_N is the solution computed via NESTA. As before continuation is very efficient when the dynamic range is high (typically higher than 40 dB). An interesting feature is that the numbers of calls are very similar over all five trials. When the dynamic range increases, the computational costs of both NESTA and RecPF naturally increase. Note that in the 60 and 80 dB experiments, RecPF did not converge to the solution and this is the reason why the number of calls saturates. While both methods have a similar computational cost in the low-dynamic range regime, NESTA has a clear advantage in the higher-dynamic range regime. Moreover, the number of iterations needed to reach convergence with NESTA with continuation is fairly low—300–400 calls to A and A^*—and so this algorithm is well suited to large-scale problems.

Figure 3.9 suggests that TwIST performs well, in terms of function calls, on the problems with low dynamic range, but deteriorates as the dynamic range increases. However, this hides a fundamental complication of the TwIST algorithm: the function calls to A and A^* are not always the dominant computational cost. TwIST is one of many TV methods that is a proximal gradient algorithm,

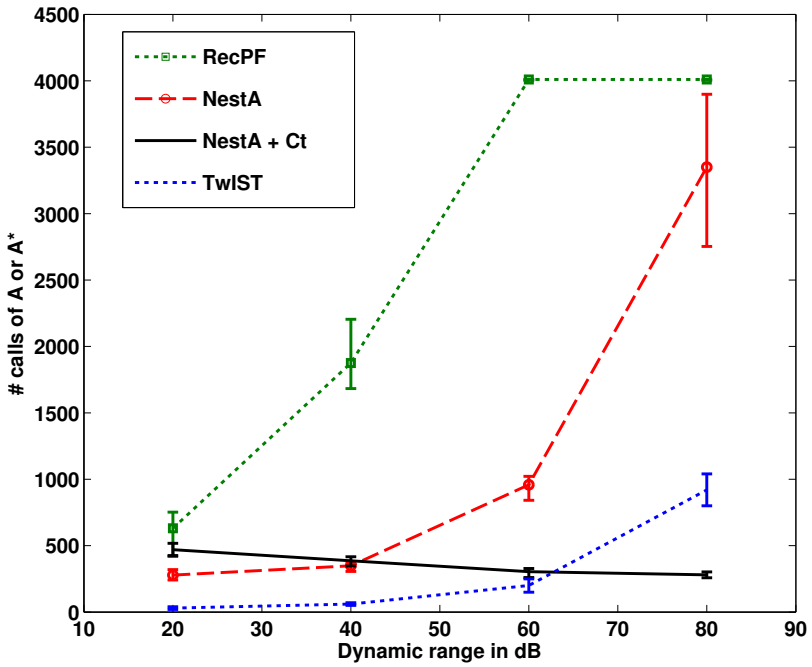

Figure 3.9: Number of calls to A and A^* as a function of the dynamic range. Solid line: NESTA with continuation. Dashed line: NESTA. Dotted line: RecPF. Dash-dotted: maximum number of iterations. In the 60 and 80 dB experiments, RecPF did not converge to the solution and this is the reason why the number of calls saturates. For TwIST, it is important not to take the number of function calls to A and A^* as a proxy for the computational time of the algorithm. Indeed, the dominant computational cost in TwIST is in evaluating the proximity operator, see Table 3.7.

Table 3.7: Comparison of computation times, mean value over 5 trials (in seconds)

Method	20dB	40dB	60dB	80dB
NESTA + Ct	750	704	553	542
RecPF	725	1918	DNC	DNC
TwIST	203	553	1675	9510

meaning that at each step, it relies on the solution to a proximity operator:

$$\text{Prox}_{TV,\gamma}(z) = \underset{x}{\text{argmin}}\, \gamma\|x\|_{TV} + \frac{1}{2}\|x - z\|_{\ell_2}^2;$$

see [CW05] and references therein. Evaluating the proximity operator at z is equivalent to solving a TV-denoising problem, which is far from trivial[10]. In [BDF07], the authors advocate the use of a side algorithm (for instance Chambolle's algorithm [Cha04]) to do this.

There are a few issues with this approach. The first is that side algorithms depend on various parameters, which adds complexity. The second is that these denoising algorithms are computationally demanding, which makes them hard to apply to large-scale problems. To illustrate this phenomenon, we compared the computation times of NESTA + Ct, RecPF, and TwIST on the same kind of synthetic 1024×1024 images. The results are reported in Table 3.7. Each line displays the computation time before convergence is reached. We observe that while TwIST needs fewer calls to A or A^* than NESTA when the dynamic range is low, the computational cost of each step is higher (and increases as the problem size increases) because the dominant computational cost in TwIST is in evaluating the proximity operator.

3.7 Handling non-projectors

Recall the derivation in §3.3.2, which assumed that $AA^* = I$. This section relaxes that assumption. The case $\epsilon = 0$ is simpler, and we offer a few methods. If $\epsilon > 0$, the only method seems to require knowledge of the SVD decomposition of A: $A = U\Sigma V^*$. We also cover the $AA^* = I$ case in more detail, since the author has received several inquiries about this.

[10] For ℓ_1 minimization, the equivalent proximity operator is given by by shrinkage, and can be solved in linear time. In the case of the weighted ℓ_1 problem, using $\|Wx\|_1$ in the objective, the proximity operator is not known in closed form unless W is diagonal. Since most first-order ℓ_1 algorithms rely on the proximity operator, this is the reason that no other algorithms can solve the analysis problem, in contrast to NESTA.

3.7.1 Revisiting the projector case

To compute y_k, we need to solve a problem of the form (with $q = d - L^{-1}c$)

$$\left(I + \frac{\lambda_\epsilon}{L} A^* A\right) y_k = \frac{\lambda_\epsilon}{L} A^* b + q \tag{3.7.1}$$

which has solution (3.3.10)

$$y_k = \left(I - \frac{\lambda_\epsilon}{\lambda_\epsilon + L} A^* A\right) \left(\frac{\lambda_\epsilon}{L} A^* b + qc\right). \tag{3.7.2}$$

To see this, it is possible to simply check that this solution works. For deriving it, one method is using the binomial inverse theorem (equivalently, the Sherman-Morrison-Woodbury formula). For matrices $U, V^* \in \mathbb{R}^{p \times q}$, with I_p and I_q the $p \times p$ and $q \times q$ identity matrices, respectively, the inverse theorem states

$$(I_p + UV)^{-1} = I_p - U(I_q + VU)^{-1}V.$$

Unless $\lambda_\epsilon = 0$, by complementary slackness it holds that $\|Ay_k - b\| = \epsilon$, so plugging in y_k and using $AA^* = I$ to simplify gives the correct value of λ_ϵ to make this hold.

3.7.2 Non-projectors for $\epsilon = 0$ case

Consider the problem in a more abstract setting. The desire is to project onto $\mathcal{Q}_p$, which is equivalent to

$$y_k = \underset{y : \|Ay - b\|_2 \leq \epsilon}{\operatorname{argmin}} \frac{1}{2}\|y - q\|_2^2 \tag{3.7.3}$$

for $q = d - L^{-1}c$.

When $\epsilon = 0$, the constraint is simply $Ay = b$. There are only two KKT conditions (now, λ is a vector, unlike λ_ϵ):

$$0 = (y - q) + A^*\lambda \qquad \text{(stationarity)} \tag{3.7.4}$$

$$b = Ay \qquad \text{(primal feasibility)}\,. \tag{3.7.5}$$

Multiplying the first equation by A gives

$$0 = b - Aq + AA^*\lambda.$$

The key computation is inverting AA^* to find λ, since once λ is found it is easy to get y.

In practice, this means the user must specify a method of calculating $(AA^*)^{-1}$. If A is large and has a fast multiply, then the conjugate-gradient algorithm is a natural choice. If A is small enough

that it can be factored, then a Cholesky factorization $RR^* = AA^*$ can be computed once at the beginning of the algorithm, producing the triangular matrix R, and subsequently $(AA^*)^{-1}(b - Aq)$ can be calculated by two back-substitution operations. The Cholesky factorization exists as long as R has full row-rank so that AA^* is positive definite. The one-time factorization cost is $\mathcal{O}(m^3)$ and the back-substitution cost is $\mathcal{O}(m^2)$ which is even less than the cost of one dense matrix-vector multiplication ($\mathcal{O}(mn)$).

An entirely different method is to take the system of equations $Ax = b$ and convert to the equivalent system (assuming A has full row-rank) $\tilde{A}x = \tilde{b}$ where $\tilde{A}\tilde{A}^* = I$. To see how this is possible, take the QR factorization $QR = A^*$ to get an orthogonal matrix Q and a triangular matrix R. Then $\tilde{A} = Q^*$ and $\tilde{b} = R^{-*}b$ satisfy $\tilde{A}x = \tilde{b}$. It is possible to do this with $\epsilon > 0$ constraints as well, but this will affect the properties of the residual, so caution must be used. The cost of the QR decomposition is $\mathcal{O}(mn^2)$ so there is not much advantage over using the SVD method described in the next subsection.

3.7.3 Non-projectors for $\epsilon > 0$ case

If there is a feasible primal vector, then Slater's conditions tell us that there is a Lagrange multiplier scalar λ_ϵ such that (3.7.3) is equivalent to the following unconstrained minimization:

$$y_k = \operatorname*{argmin}_y \frac{1}{2}\|y - q\|_2^2 + \frac{\lambda_\epsilon}{2}\|Ay - b\|_2^2. \tag{3.7.6}$$

Let $A = U\Sigma V^*$ be the full singular value decomposition (SVD) of A, so that U and V are orthogonal (or unitary in the complex-valued case) matrices. The ℓ_2 norm is unitary-invariant, so $\|Ay - b\| = \|U^*(Ay - b)\|$. Make the change-of-variables $x = V^*(y - q)$ and $\tilde{b} = U^*(b - Ay)$, so the problem is

$$\min_x \frac{1}{2}\|x\|^2 + \frac{\lambda_\epsilon}{2}\|\Sigma x - \tilde{b}\|^2.$$

Since Σ is a diagonal matrix (with diagonal $(\sigma_1, \ldots, \sigma_r)$ where $r = \min(m, n)$), this problem is separable, and the solution is easy to compute since this amounts to solving a scalar quadratic equation:

$$(x_{\lambda_\epsilon})_i = \frac{\lambda_\epsilon \sigma_i \tilde{b}_i}{1 + \lambda_\epsilon \sigma_i^2}, \quad i = 1, \ldots, n.$$

The issue is that the value of λ_ϵ is unknown. It should be chosen so that

$$\|Ay - b\|_2 = \|\Sigma x_{\lambda_\epsilon} - \tilde{b}\| = \epsilon.$$

For a given λ, define

$$h(\lambda) = \|\Sigma x_\lambda - \tilde{b}\|^2 - \epsilon^2 = \sum_{i=1}^{n} \left(\sigma_i \frac{\lambda \sigma_i b_i}{1+\lambda\sigma_i^2} - b_i\right)^2 - \epsilon^2 = \sum_{i=1}^{n} \frac{b_i^2}{(1+\lambda\sigma_i^2)^2} - \epsilon^2.$$

Thus we search for a root λ_ϵ of h. The derivative of h is smooth and easy to compute:

$$\frac{dh}{d\lambda}(\lambda) = -2\sum_{i=1}^{n} \frac{(b_i\sigma_i)^2}{(1+\lambda\sigma_i^2)^3}$$

so the natural choice is Newton's method. In a sense, this is a simplified version of the SPGL1 algorithm. The Newton's method is well-posed in this case, since h is a non-increasing function of λ so it can have at most one root.

In the NESTA code, we use a safe-guarded Newton method that does a brief check to keep $\lambda \geq 0$ at every inner iteration. It is possible that $\lambda_\epsilon = 0$ which signifies that $q \in \mathcal{Q}_p$ already so $y = q$ is feasible. During the Newton's method algorithm, all the computations are $\mathcal{O}(n)$ so it is relatively fast. Furthermore, since A, b, and ϵ are constants, and q changes only slightly from iteration to iteration, the value of λ from the previous iteration can be used to warm-start the new Newton's method search. Except for pathological test cases, this extra optimization problem adds no significant running time to NESTA. Similar approaches that use Newton's method to solve the projection have been independently proposed in [DHJJ10] and [WBFA09]. These papers discuss the TV minimization problem and, like NESTA, are based off [Nes05].

In practice, the full SVD is not needed; a "thin" or "skinny" SVD is sufficient. If A does not have full row rank, then b need not be in the range of A. To deal with this, split b into two parts, $b = b_R + b_N$ where $b_R = UU^*b$ is in the range of A and thus $b_N = b - b_R$ is in the null space of A^*. Then the ϵ and b terms are updated

$$\epsilon \leftarrow \sqrt{\epsilon^2 - \|b_N\|^2}, \quad b \leftarrow b_R$$

and the algorithm is run as before.

These improvements have been included in the version "1.1" release of NESTA, and are used to solve the RMPI problems discussed in Chapter 2. For the RMPI problem, W is an overcomplete Gabor dictionary which is costly to compute; on the other-hand, the sensing matrix A (called Φ in the RMPI chapter) is of modest size, such as 80×1024. The largest matrices tested in RMPI have been 8192 columns, and even this is manageable (see Figure 3.10), since the FFT calls in the Gabor dictionary will dominate. So the NESTA algorithm is extremely well suited to the largescale analysis problems that arise in RMPI; indeed, this was a motivation for creating NESTA.

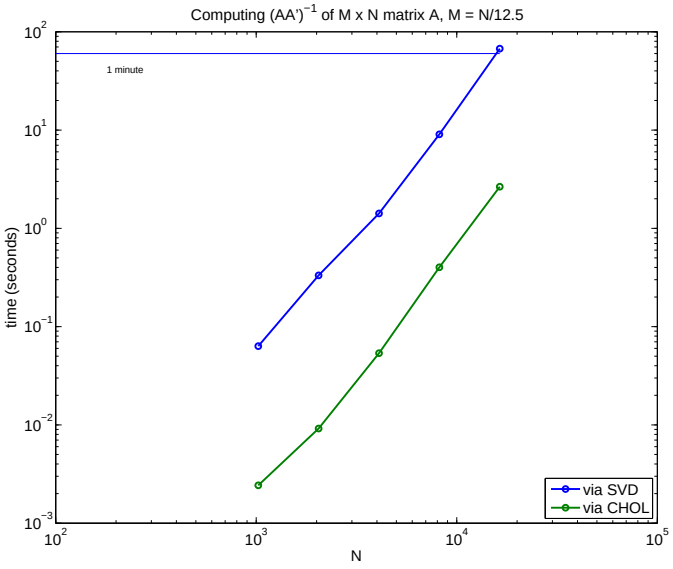

Figure 3.10: The computation time for calculating the Cholesky decomposition and thin SVD decomposition of a matrix, as a function of the columns n of the matrix. Computed in MATLAB on an Intel i7 laptop. This computation needs only be done once per matrix A, so it can be amortized over all iterations or even all experiments, which is the case if multiple problems use the same A matrix (with perhaps different b or ϵ parameters). Working with a matrix of 10000 columns is quite reasonable, though obviously this method scales poorly if the dimensions get really large since both SVD and Cholesky scale at $\mathcal{O}(m^2 n)$ and $\mathcal{O}(m^3)$, respectively. For this plot, $m = n/12.5$ which is the same ratio used in the RMPI device of Chapter 2.

3.8 Discussion

In this chapter, we have proposed an algorithm for general sparse recovery problems, which is based on Nesterov's method. This algorithm is accurate and competitive with state-of-the-art alternatives. In fact, in applications of greatest interest such as the recovery of approximately sparse signals, it outperforms most of the existing methods we have used in our comparisons and is comparable to the best. Further, what is interesting here, is that we have not attempted to optimize the algorithm in any way. For instance, we have not optimized the parameters $\{\alpha_k\}$ and $\{\tau_k\}$, or the number of continuation steps as a function of the desired accuracy δ, and so it is expected that finer tuning would speed up the algorithm. Another advantage is that NESTA is extremely flexible in the sense that minor adaptations lead to efficient algorithms for a host of optimization problems that are crucial in the field of signal/image processing.

3.8.1 Extensions

This chapter focused on the situation in which A^*A is a projector (the rows of A are orthonormal). This stems from the facts that 1) the most computationally friendly compressed sensing are of this form, and 2) it allows fast computations of the two sequence of iterates $\{y_k\}$ and $\{z_k\}$. It is important, however, to extend NESTA as to be able to cope with a wider range of problem in which A^*A is not a projection (or not diagonal).

In order to do this, observe that in Steps 2 and 3, we need to solve problems of the form

$$y_k = \underset{x \in \mathcal{Q}_p}{\operatorname{argmin}} \|x - q\|_{\ell_2}^2,$$

for some q, and we have seen that the solution is given by $y_k = \mathcal{P}_{\mathcal{Q}_p}(q)$, where $\mathcal{P}_{\mathcal{Q}_p}$ is the projector onto $\mathcal{Q}_p := \{x : \|Ax - b\|_{\ell_2} \leq \epsilon\}$. The solution is given by

$$y_k = (I + \lambda A^* A)^{-1}(q + \lambda A^* b) \tag{3.8.1}$$

for some $\lambda \geq 0$. When the eigenvalues of A^*A are well clustered, the right-hand side of (3.8.1) can be computed very efficiently via a few conjugate gradients (CG) steps. Note that this is of direct interest in compressed sensing applications in which A is a random matrix since in all the cases we are familiar with, the eigenvalues of A^*A are tightly clustered. Hence, NESTA may be extended to general problems while retaining its efficiency, with the proviso that a good rule for selecting λ in (3.8.1) is available; i.e., such that $\|Ay_k - b\|_{\ell_2} = \epsilon$ unless $q \in \mathcal{Q}_p$. Of course, one can always eliminate the problem of finding such a λ by solving the unconstrained problem (QP_λ) instead of (BP_ϵ). In this case, each NESTA iteration is actually very cheap, no matter what A looks like.

It is worth noting that if A is relatively small, so that is computationally possible to perform a one-time singular value decomposition of A (this may be the case if A is not too large, but the analysis dictionary W is very overcomplete and the bottleneck for computation), then the inversion just requires two matrix multiplies. The correct value of λ is given by a scalar equation that may be solved efficiently by Newton's method (see also [WBFA09] for a similar approach to selecting λ). This has been discussed in detail in §3.7.

Finally, we also observe that Nesterov's framework is likely to provide efficient algorithms for related problems, which do not have the special $\ell_1 + \ell_2^2$ structure. One example might be the Dantzig selector, which is a convenient and flexible estimator for recovering sparse signals from noisy data [CT07a]:

$$\begin{array}{ll} \text{minimize} & \|x\|_{\ell_1} \\ \text{subject to} & \|A^*(b - Ax)\|_{\ell_\infty} \leq \delta. \end{array} \tag{3.8.2}$$

This is of course equivalent to the unconstrained problem

$$\text{minimize} \quad \lambda\|x\|_{\ell_1} + \|A^*(b - Ax)\|_{\ell_\infty}$$

for some value of λ. Clearly, one could apply Nesterov's smoothing techniques to smooth both terms in the objective functional together with Nesterov's accelerated gradient techniques, and derive a novel and efficient algorithm for computing the solution to the Dantzig selector. This is an example among many others. Another might be the minimization of a sum of two norms, e.g., an ℓ_1 and a

TV norm, under data constraints.

3.8.2 Software

In the spirit of reproducible research [DMR+09], a Matlab version of NESTA is available at: `http://www.acm.caltech.edu/~nesta/`.

Acknowledgements

The authors are grateful to the anonymous reviewers for helpful comments and for pointing out [EMZ07, Lor09, LBDM+09]. S. Becker wishes to thank Peter Stobbe for the use of his Hadamard Transform and Gabor frame code, and Wotao Yin for helpful discussions about RecPF. J. Bobin wishes to thank Hamza Fawzi for fruitful discussions, and E. Candès would like to thank Jalal Fadili for his suggestions. We are grateful to Stephen Wright for his comments on an earlier version of the paper that this chapter is based on, for suggesting use of a better version of GPSR, and encouraging us to test SpaRSA. Thanks Stephen!