SIAM J. SCI. COMPUT.
Vol. 31, No. 2, pp. 890–912

PROBING THE PARETO FRONTIER FOR BASIS PURSUIT SOLUTIONS*

EWOUT VAN DEN BERG† AND MICHAEL P. FRIEDLANDER†

Abstract. The basis pursuit problem seeks a minimum one-norm solution of an underdetermined least-squares problem. Basis pursuit denoise (BPDN) fits the least-squares problem only approximately, and a single parameter determines a curve that traces the optimal trade-off between the least-squares fit and the one-norm of the solution. We prove that this curve is convex and continuously differentiable over all points of interest, and show that it gives an explicit relationship to two other optimization problems closely related to BPDN. We describe a root-finding algorithm for finding arbitrary points on this curve; the algorithm is suitable for problems that are large scale and for those that are in the complex domain. At each iteration, a spectral gradient-projection method approximately minimizes a least-squares problem with an explicit one-norm constraint. Only matrix-vector operations are required. The primal-dual solution of this problem gives function and derivative information needed for the root-finding method. Numerical experiments on a comprehensive set of test problems demonstrate that the method scales well to large problems.

Key words. basis pursuit, convex program, duality, root-finding, Newton's method, projected gradient, one-norm regularization, sparse solutions

AMS subject classifications. 49M29, 65K05, 90C25, 90C06

DOI. 10.1137/080714488

1. Basis pursuit denoise.

The basis pursuit problem aims to find a sparse solution of the underdetermined system of equations $Ax = b$, where A is an m-by-n matrix and b is an m-vector. Typically, $m \ll n$, and the problem is ill-posed. The approach advocated by Chen, Donoho, and Saunders [15] is to solve the convex optimization problem

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

In the presence of noisy or imperfect data, however, it is undesirable to exactly fit the linear system. Instead, the constraint in (BP) is relaxed to obtain the basis pursuit denoise (BPDN) problem

$$\text{(BP}_\sigma\text{)} \qquad \underset{x}{\text{minimize}} \quad \|x\|_1 \quad \text{subject to} \quad \|Ax - b\|_2 \le \sigma,$$

where the positive parameter σ is an estimate of the noise level in the data. The case $\sigma = 0$ corresponds to a solution of (BP)—i.e., a basis pursuit solution.

There is now a significant body of work that addresses the conditions under which a solution of this problem yields a *sparse* approximation to a solution of the underdetermined system; see Candès, Romberg, and Tao [11], Donoho [24], and Tropp [48], and references therein. The sparse approximation problem is of vital importance to many applications in signal processing and statistics. Some important applications

*Received by the editors January 28, 2008; accepted for publication (in revised form) June 2, 2008; published electronically November 26, 2008. This work was supported by NSERC Collaborative Research and Development Grant 334810-05.
http://www.siam.org/journals/sisc/31-2/71448.html.
†Department of Computer Science, University of British Columbia, Vancouver V6T 1Z4, BC, Canada (ewout@cs.ubc.ca, mpf@cs.ubc.ca).

include image reconstruction, such as MRI [36, 37] and seismic [31, 32] images, and model selection in regression [26]. In many of these applications, the data sets are large, and the matrix A is available only as an operator. In compressed sensing [10, 11, 12, 23], for example, the matrices are often fast operators such as Fourier or wavelet transforms. It is therefore crucial to develop algorithms for the sparse reconstruction problem that scale well and work effectively in a matrix-free context.

We present an algorithm, suitable for large-scale applications, that is capable of finding solutions of (BP_σ) for any value of $\sigma \geq 0$. Our approach is based on recasting (BP_σ) as a problem of finding the root of a single-variable nonlinear equation. At each iteration of our algorithm, an estimate of that variable is used to define a convex optimization problem whose solution yields derivative information that can be used by a Newton-based root-finding algorithm.

1.1. One-norm regularization. The convex optimization problem (BP_σ) is only one possible statement of the one-norm regularized least-squares problem. In fact, the BPDN label is typically applied to the penalized least-squares problem

$$(\mathrm{QP}_\lambda) \qquad \underset{x}{\text{minimize}} \quad \|Ax - b\|_2^2 + \lambda \|x\|_1,$$

which is the problem statement proposed by Chen, Donoho, and Saunders [14, 15]. A third formulation,

$$(\mathrm{LS}_\tau) \qquad \underset{x}{\text{minimize}} \quad \|Ax - b\|_2 \quad \text{subject to} \quad \|x\|_1 \leq \tau,$$

has an explicit one-norm constraint and is often called the Lasso problem [46]. The parameter λ is related to the Lagrange multiplier of the constraint in (LS_τ) and to the reciprocal of the multiplier of the constraint in (BP_σ). Thus, for appropriate parameter choices of σ, λ, and τ, the solutions of (BP_σ), (QP_λ), and (LS_τ) coincide, and these problems are in some sense equivalent. However, except for special cases—such as A orthogonal—the parameters that make these problems equivalent cannot be known a priori.

The formulation (QP_λ) is often preferred because of its close connection to convex quadratic programming, for which many algorithms and software are available; some examples include iteratively reweighted least squares [7, section 4.5] and gradient projection [27]. For the case where an estimate of the noise-level σ is known, Chen, Donoho, and Saunders [15, section 5.2] argue that the choice $\lambda = \sigma\sqrt{2 \log n}$ has important optimality properties. However, this argument hinges on the orthogonality of A.

We focus on the situation where σ is approximately known—such as when we can estimate the noise levels inherent in an underlying system or in the measurements taken. In this case it is preferable to solve (BP_σ), and here this is our primary goal. An important consequence of our approach is that it can also be used to efficiently solve the closely related problems (BP) and (LS_τ). Our algorithm also applies to all three problems in the complex domain, which can arise in signal processing applications.

1.2. Approach. At the heart of our approach is the ability to efficiently solve a sequence of (LS_τ) problems using a spectral projected-gradient (SPG) algorithm [5, 6, 18]. As with (QP_λ), this problem is parameterized by a scalar; the crucial difference, however, is that the dual solution of (LS_τ) yields vital information on how to update τ so that the next solution of (LS_τ) is much closer to the solution of (BP_σ).

Let x_τ denote the optimal solution of (LS_τ). The single-parameter function

$$\phi(\tau) = \|r_\tau\|_2, \quad \text{with} \quad r_\tau := b - Ax_\tau \tag{1.1}$$

gives the optimal value of (LS_τ) for each $\tau \geq 0$. As we describe in section 2, its derivative is given by $-\lambda_\tau$, where $\lambda_\tau \geq 0$ is the unique dual solution of (LS_τ). Importantly, this dual solution can easily be obtained as a by-product of the minimization of (LS_τ); this is discussed in section 2.1. Our approach is then based on applying Newton's method to find a root of the nonlinear equation

$$\phi(\tau) = \sigma, \tag{1.2}$$

which defines a sequence of regularization parameters $\tau_k \to \tau_\sigma$, where x_{τ_σ} is a solution of (BP_σ). In other words, τ_σ is the parameter that causes (LS_τ) and (BP_σ) to share the same solution.

There are four distinct components to this paper. The first two are related to the root-finding algorithm for (BP_σ). The third is an efficient algorithm for solving (LS_τ)—and hence for evaluating the function ϕ and its derivative ϕ'. The fourth gives the results of a series of numerical experiments.

Pareto curve (section 2). The Pareto curve defines the optimal trade-off between the two-norm of the residual r and the one-norm of the solution x. The problems (BP_σ) and (LS_τ) are two distinct characterizations of the same curve. Our approach uses the function ϕ to parameterize the Pareto curve by τ. We show that, for all points of interest, ϕ—and hence the Pareto curve—is continuously differentiable. We are also able to give an explicit expression for its derivative. This surprising result permits us to use a Newton-based root-finding algorithm to find roots of the nonlinear equation (1.2), which correspond to points on the Pareto curve. Thus we can find solutions of (BP_σ) for any $\sigma \geq 0$.

Root finding (section 3). Each iteration of the root-finding algorithm for (1.2) requires the evaluation of ϕ and ϕ' at some τ, and hence the minimization of (LS_τ). This is a potentially expensive subproblem, and the effectiveness of our approach hinges on the ability to solve this subproblem only approximately. We present rate-of-convergence results for the case where ϕ and ϕ' are known only approximately. This is in contrast to the usual inexact-Newton analysis [22], which assumes that ϕ is known exactly. We also give an effective stopping rule for determining the required accuracy of each minimization of (LS_τ).

Projected gradient for Lasso (section 4). We describe an SPG algorithm for solving (LS_τ). Each iteration of this method requires an orthogonal projection of an n-vector onto the convex set $\|x\|_1 \leq \tau$. In section 4.2 we give an algorithm for this projection with a worst-case complexity of $\mathcal{O}(n \log n)$. In many important applications, A is a Fourier-type operator, and matrix-vector products with A and A^T can be obtained with $\mathcal{O}(n \log n)$ cost. The projection cost is typically much smaller than the worst case, and the dominant cost in our algorithm consists of the matrix-vector products, as it does in other algorithms for BPDN. We also show how the projection algorithm can easily be extended to project complex vectors, which allows us to extend the SPG algorithm to problems in the complex domain.

Implementation and numerical experiments (sections 5 and 6). To demonstrate the effectiveness of our approach, we apply our algorithm on a set of benchmark problems and compare it to other state-of-the-art solvers. In sections 6.1 and 6.2 we report numerical results on a series of (BP_σ) and (BP) problems, which are normally considered as distinct problems. In section 6.3 we report numerical results on a series of (LS_τ) problems for various values of τ, and compare against the equivalent (QP_λ) for-

mulations. In section 6.4 we show how to capitalize on the smoothness of the Pareto curve to obtain quick and approximate solutions to (BP_σ).

1.3. Assumption. We make the following blanket assumption throughout.

Assumption 1.1. The vector $b \in \mathrm{range}(A)$, and $b \neq 0$.

This assumption is only needed in order to simplify the discussion, and it implies that (BP_σ) is feasible for all $\sigma \geq 0$. In many applications, such as compressed sensing [10, 11, 12, 23], A has full row rank, and therefore this assumption is satisfied automatically.

1.4. Related work.

Homotopy approaches. A number of approaches have been suggested for solving (BP_σ), many of which are based on repeatedly solving (QP_λ) for various values of λ. Notable examples of this approach are HOMOTOPY [41, 42] and LARS [26], which solve (QP_λ) for essentially all values of λ. In this way, they eventually find the value of λ that recovers a solution of (BP_σ). These active-set continuation approaches begin with $\lambda = \|A^T b\|_\infty$ (for which the corresponding solution is $x_\lambda = 0$) and gradually reduce λ in stages that predictably change the sparsity pattern in x_λ. The remarkable efficiency of these continuation methods follows from their ability to systematically update the resulting sequence of solutions. (See Donoho and Tsaig [25] and Friedlander and Saunders [28] for discussions of the performance of these methods.) The computational bottleneck for these methods is the accurate solution at each iteration of a least-squares subproblem that involves a subset of the columns of A. In some applications (such as the seismic image reconstruction problem [32]) the size of this subset can become large, and thus the least-squares subproblems can become prohibitively expensive. Moreover, even if the correct value λ_σ is known a priori, the method must necessarily begin with $\lambda = \|A^T b\|_\infty$ and traverse all critical values of λ down to λ_σ.

BPDN as a cone program. The problem (BP_σ) with $\sigma > 0$ can be considered as a special case of a generic second-order cone program [8, Chapter 5]. Interior-point (IP) algorithms for general conic programs can be very effective if the matrices are available explicitly. Examples of general-purpose software for cone programs include SEDUMI [45] and MOSEK [39]. The software package ℓ_1-MAGIC [9] contains an IP implementation specially adapted to (BP_σ). In general, the efficiency of IP implementations relies ultimately on their ability to efficiently solve certain linear systems that involve highly ill-conditioned matrices.

Basis pursuit as a linear program. The special case $\sigma = 0$ corresponding to (BP) can be reformulated and solved as a linear program. Again, IP methods are known to be effective for general linear programs, but many IP implementations for general linear programming, such as CPLEX [16] and MOSEK, require explicit matrices. The solver PDCO [44], available within the SPARSELAB package, is capable of using A as an operator only, but it often requires many matrix-vector multiplications to converge, and as we report in section 6.2, it is not generally competitive with other approaches. We are not aware of simplex-type implementations that do not require A explicitly.

Sampling the Pareto curve. A common approach for obtaining approximate solutions to (BP_σ) is to sample various points on the Pareto curve; this is often accomplished by solving (QP_λ) for a decreasing sequence of values of λ. As noted by Das and Dennis [19], and more recently by Leyffer [35], a uniform distribution of weights λ can lead to an uneven sampling of the Pareto curve. In contrast, by instead parameterizing the Pareto curve by σ or τ (via the problem (BP_σ) or (LS_τ)), it is possible to obtain a more uniform sample of the Pareto curve; see section 6.4.

Projected gradient. Our application of the SPG algorithm to solve (LS_τ) follows Birgin, Martínez, and Raydan [5] closely for the minimization of general nonlinear functions over arbitrary convex sets. The method they propose combines projected-gradient search directions with the spectral step length that was introduced by Barzilai and Borwein [1]. A nonmonotone line search is used to accept or reject steps. The key ingredient of Birgin, Martínez, and Raydan's algorithm is the projection of the gradient direction onto a convex set, which in our case is defined by the constraint in (LS_τ). In their recent report, Figueiredo, Nowak, and Wright [27] describe the remarkable efficiency of an SPG method specialized to (QP_λ). Their approach builds on the earlier report by Dai and Fletcher [18] on the efficiency of a specialized SPG method for general bound-constrained quadratic programs (QPs).

2. The Pareto curve.

The function ϕ defined by (1.1) yields the optimal value of the constrained problem (LS_τ) for each value of the regularization parameter τ. Its graph traces the optimal trade-off between the one-norm of the solution x and the two-norm of the residual r, which defines the Pareto curve. Figure 2.1 shows the graph of ϕ for a typical problem.

The Newton-based root-finding procedure that we propose for locating specific points on the Pareto curve—e.g., finding roots of (1.2)—relies on several important properties of the function ϕ. As we show in this section, ϕ is a convex and differentiable function of τ. The differentiability of ϕ is perhaps unintuitive, given that the one-norm constraint in (LS_τ) is not differentiable. To deal with the nonsmoothness of the one-norm constraint, we appeal to Lagrange duality theory. This approach yields significant insight into the properties of the trade-off curve. We discuss the most important properties below.

2.1. The dual subproblem.

The dual of the Lasso problem (LS_τ) plays a prominent role in understanding the Pareto curve. In order to derive the dual of (LS_τ), we first recast (LS_τ) as the equivalent problem

$$\underset{r,x}{\text{minimize}} \quad \|r\|_2 \quad \text{subject to} \quad Ax + r = b, \quad \|x\|_1 \le \tau. \tag{2.1}$$

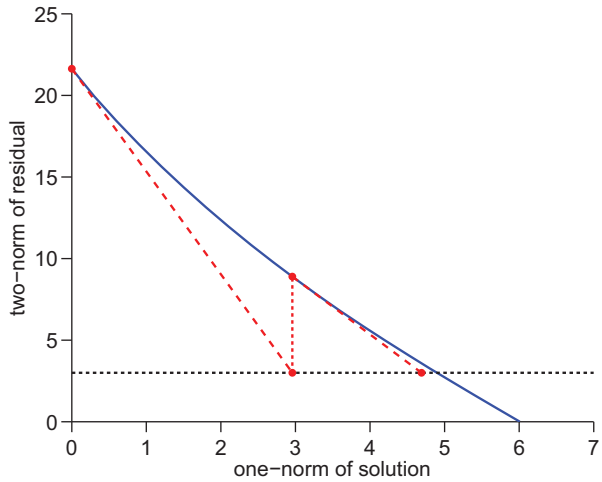


FIG. 2.1. *A typical Pareto curve (solid line) showing two iterations of Newton's method. The first iteration is available at no cost.*

The dual of this convex problem is given by

$$\underset{y,\lambda}{\text{maximize}} \quad \mathcal{L}(y,\lambda) \quad \text{subject to} \quad \lambda \geq 0, \tag{2.2}$$

where

$$\mathcal{L}(y,\lambda) = \inf_{x,r} \{\|r\|_2 - y^T(Ax + r - b) + \lambda(\|x\|_1 - \tau)\}$$

is the Lagrange dual function, and the m-vector y and scalar λ are the Lagrange multipliers (e.g., dual variables) corresponding to the constraints in (2.1). We use the separability of the infimum in r and x to rearrange terms and arrive at the equivalent statement

$$\mathcal{L}(y,\lambda) = b^T y - \tau\lambda - \sup_r \{y^T r - \|r\|_2\} - \sup_x \{y^T A x - \lambda\|x\|_1\}.$$

We recognize the suprema above as the conjugate functions of $\|r\|_2$ and of $\lambda\|x\|_1$, respectively. For an arbitrary norm $\|\cdot\|$ with dual norm $\|\cdot\|_*$, the conjugate function of $f(x) = \alpha\|x\|$ for any $\alpha \geq 0$ is given by

$$f_*(y) := \sup_x \{y^T x - \alpha\|x\|\} = \begin{cases} 0 & \text{if } \|y\|_* \leq \alpha, \\ \infty & \text{otherwise;} \end{cases} \tag{2.3}$$

see Boyd and Vandenberghe [8, section 3.3.1]. With this expression of the conjugate function, it follows that (2.2) remains bounded if and only if the dual variables y and λ satisfy the constraints $\|y\|_2 \leq 1$ and $\|A^T y\|_\infty \leq \lambda$. The dual of (2.1), and hence of (LS_τ), is then given by

$$\underset{y,\lambda}{\text{maximize}} \quad b^T y - \tau\lambda \quad \text{subject to} \quad \|y\|_2 \leq 1, \ \|A^T y\|_\infty \leq \lambda; \tag{2.4}$$

the nonnegativity constraint on λ is implicitly enforced by the second constraint.

Importantly, the dual variables y and λ can easily be computed from the optimal primal solutions. To derive y, first note that, from (2.3),

$$\sup_r \{y^T r - \|r\|_2\} = 0 \quad \text{if} \quad \|y\|_2 \leq 1. \tag{2.5}$$

Therefore, $y = r/\|r\|_2$, and we can without loss of generality take $\|y\|_2 = 1$ in (2.4). To derive λ, note that, as long as $\tau > 0$, λ must be at its lower bound, as implied by the constraint $\|A^T y\|_\infty \leq \lambda$. Hence, we take $\lambda = \|A^T y\|_\infty$. (If $r = 0$ or $\tau = 0$, the choice of y or λ, respectively, is arbitrary.) The dual variable y can then be eliminated, and we arrive at the following necessary and sufficient optimality conditions for the primal-dual solution $(r_\tau, x_\tau, \lambda_\tau)$ of (2.1):

$$Ax_\tau + r_\tau = b, \quad \|x_\tau\|_1 \leq \tau \qquad \text{(primal feasibility);} \tag{2.6a}$$

$$\|A^T r_\tau\|_\infty \leq \lambda_\tau \|r_\tau\|_2 \qquad \text{(dual feasibility);} \tag{2.6b}$$

$$\lambda_\tau(\|x_\tau\|_1 - \tau) = 0 \qquad \text{(complementarity).} \tag{2.6c}$$

2.2. Convexity and differentiability of the Pareto curve. Let τ_{BP} be the optimal objective value of the problem (BP). This corresponds to the smallest value of τ such that (LS_τ) has a zero objective value. As we show below, ϕ is nonincreasing,

and therefore τ_{BP} is the first point at which the graph of ϕ touches the horizontal axis. Our assumption that $0 \neq b \in \text{range}(A)$ implies that (BP) is feasible and that $\tau_{\text{BP}} > 0$. Therefore, at the endpoints of the interval of interest,

$$\phi(0) = \|b\|_2 > 0 \quad \text{and} \quad \phi(\tau_{\text{BP}}) = 0. \tag{2.7}$$

As the following result confirms, the function is convex and strictly decreasing over the interval $\tau \in [0, \tau_{\text{BP}}]$. It is also continuously differentiable on the interior of this interval—this is a crucial property.

THEOREM 2.1.

(a) *The function ϕ is convex and nonincreasing.*

(b) *For all $\tau \in (0, \tau_{\text{BP}})$, ϕ is continuously differentiable, $\phi'(\tau) = -\lambda_\tau$, and the optimal dual variable $\lambda_\tau = \|A^T y_\tau\|_\infty$, where $y_\tau = r_\tau / \|r_\tau\|_2$.*

(c) *For $\tau \in [0, \tau_{\text{BP}}]$, $\|x_\tau\|_1 = \tau$, and ϕ is strictly decreasing.*

Proof. (a) The function ϕ can be restated as

$$\phi(\tau) = \inf_x f(x, \tau), \tag{2.8}$$

where

$$f(x,\tau) := \|Ax - b\|_2 + \psi_\tau(x) \quad \text{and} \quad \psi_\tau(x) := \begin{cases} 0 & \text{if } \|x\|_1 \leq \tau, \\ \infty & \text{otherwise.} \end{cases}$$

Note that, by (2.3), $\psi_\tau(x) = \sup_z \{x^T z - \tau \|z\|_\infty\}$, which is the pointwise supremum of an affine function in (x, τ). Therefore, it is convex in (x, τ). Together with the convexity of $\|Ax - b\|_2$, this implies that f is convex in (x, τ). Consider any nonnegative scalars τ_1 and τ_2, and let x_1 and x_2 be the corresponding minimizers of (2.8). For any $\beta \in [0, 1]$,

$$\begin{aligned} \phi(\beta\tau_1 + (1-\beta)\tau_2) &= \inf_x f(x, \beta\tau_1 + (1-\beta)\tau_2) \\ &\leq f\big(\beta x_1 + (1-\beta)x_2\,,\, \beta\tau_1 + (1-\beta)\tau_2\big) \\ &\leq \beta f(x_1, \tau_1) + (1-\beta) f(x_2, \tau_2) \\ &= \beta\phi(\tau_1) + (1-\beta)\phi(\tau_2). \end{aligned}$$

Hence, ϕ is convex in τ. Moreover, ϕ is nonincreasing because the feasible set enlarges as τ increases.

(b) The function ϕ is differentiable at τ if and only if its subgradient at τ is unique [43, Theorem 25.1]. By [4, Proposition 6.5.8(a)], $-\lambda_\tau \in \partial\phi(\tau)$. Therefore, to prove the differentiability of ϕ, it is enough to show that λ_τ is unique. Note that λ appears linearly in (2.4) with coefficient $-\tau < 0$, and thus λ_τ is not optimal unless it is at its lower bound, as implied by the constraint $\|A^T y\|_\infty \leq \lambda$. Hence, $\lambda_\tau = \|A^T y_\tau\|_\infty$. Moreover, the convexity of (LS_τ) implies that its optimal value is unique, and so $r_\tau \equiv b - Ax_\tau$ is unique. Also, $\|r_\tau\| > 0$ because $\tau < \tau_{\text{BP}}$ (cf. (2.7)). As discussed in connection with (2.5), we can then take $y_\tau = r_\tau / \|r_\tau\|_2$, and so the uniqueness of r_τ implies the uniqueness of y_τ, and hence the uniqueness of λ_τ, as required. The continuity of the gradient follows from the convexity of ϕ.

(c) The assertion holds trivially for $\tau = 0$. For $\tau = \tau_{\text{BP}}$, $\|x_{\tau_{\text{BP}}}\|_1 = \tau_{\text{BP}}$ by definition. It only remains to prove part (c) on the interior of the interval. Note that $\phi(\tau) \equiv \|r_\tau\| > 0$ for all $\tau \in [0, \tau_{\text{BP}})$. Then by part (b), $\lambda_\tau > 0$, and hence ϕ is strictly decreasing for $\tau < \tau_{\text{BP}}$. But because x_τ and λ_τ both satisfy the complementarity condition in (2.6), it must hold that $\|x_\tau\|_1 = \tau$. ∎

2.3. Generic regularization. The technique used to prove Theorem 2.1 does not in any way rely on the specific norms used in the objective and regularization functions, and it can be used to prove similar properties for the generic regularized fitting problem

$$\underset{x}{\text{minimize}} \quad \|Ax-b\|_s \quad \text{subject to} \quad \|x\|_p \le \tau, \tag{2.9}$$

where $1 \le (p,s) \le \infty$ define the norms of interest, i.e., $\|x\|_p = (\sum_i |x_i|^p)^{1/p}$. More generally, the constraint here may appear as $\|Lx\|_p$, where L may be rectangular. Such a constraint defines a seminorm, and it often arises in the discrete approximations of derivative operators. In particular, least-squares with Tikhonov regularization [47], which corresponds to $p = s = 2$, is used extensively for the regularization of ill-posed problems; see Hansen [29] for a comprehensive study. In this case, the Pareto curve defined by the optimal trade-off between $\|x\|_2$ and $\|Ax-b\|_2$ is often called the *L-curve* because of its shape when plotted on a log-log scale [30].

If we define $\bar{p}$ and $\bar{s}$ such that

$$1/p + 1/\bar{p} = 1 \quad \text{and} \quad 1/s + 1/\bar{s} = 1,$$

then the dual of the generic regularization problem is given by

$$\underset{y,\lambda}{\text{maximize}} \quad b^T y - \tau\lambda \quad \text{subject to} \quad \|y\|_{\bar{s}} \le 1, \ \|A^T y\|_{\bar{p}} \le \lambda.$$

As with (2.4), the optimal dual variables are given by $y = r/\|r\|_{\bar{p}}$ and $\lambda = \|A^T y\|_{\bar{s}}$. This is a generalization of the results obtained by Dax [21], who derives the dual for p and s strictly between 1 and ∞. The corollary below, which follows from a straightfoward modification of Theorem 2.1, asserts that the Pareto curve defined for any $1 \le (p,s) \le \infty$ in (2.7) has the properties of convexity and differentiability.

COROLLARY 2.2. *Let $\theta(\tau) := \|r_\tau\|_s$, where $r_\tau := b - Ax_\tau$, and x_τ is the optimal solution of* (2.9).

(a) *The function θ is convex and nonincreasing.*
(b) *For all $\tau \in (0, \tau_{\text{BP}})$, θ is continuously differentiable, $\theta'(\tau) = -\lambda_\tau$, and the optimal dual variable $\lambda_\tau = \|A^T y_\tau\|_{\bar{p}}$, where y_τ satisfies $y^T r_\tau = \|r_\tau\|_s$.*
(c) *For $\tau \in [0, \tau_{\text{BP}}]$, $\|x_\tau\|_p = \tau$, and θ is strictly decreasing.*

3. Root finding. As we briefly outlined in section 1.2, our algorithm generates a sequence of regularization parameters $\tau_k \to \tau_\sigma$ based on the Newton iteration

$$\tau_{k+1} = \tau_k + \Delta\tau_k, \quad \text{with} \quad \Delta\tau_k := \big(\sigma - \phi(\tau_k)\big)/\phi'(\tau_k), \tag{3.1}$$

such that the corresponding solutions x_{τ_k} of (LS_{τ_k}) converge to x_σ. For values of $\sigma \in (0, \|b\|_2)$, Theorem 2.1 implies that ϕ is convex, strictly decreasing, and continuously differentiable. In that case it is clear that $\tau_k \to \tau_\sigma$ superlinearly for all initial values $\tau_0 \in (0, \tau_{\text{BP}})$ (see, e.g., Bertsekas [3, Proposition 1.4.1]).

The efficiency of our method, as with many Newton-type methods for large problems, ultimately relies on the ability to carry out the iteration described by (3.1) with only an approximation of $\phi(\tau_k)$ and $\phi'(\tau_k)$. Although the nonlinear equation (1.2) that we wish to solve involves only a single variable τ, the evaluation of $\phi(\tau)$ involves the solution of (LS_τ), which can be a large optimization problem that is expensive to solve to full accuracy.

For systems of nonlinear equations in general, inexact Newton methods assume that the Newton system analogous to the equation

$$\phi'(\tau_k)\Delta\tau_k = \sigma - \phi(\tau_k)$$

is solved only approximately, with a residual that is a fraction of the right-hand side. A constant fraction yields a linear convergence rate, and a fraction tending to zero yields a superlinear convergence rate (see, e.g., Nocedal and Wright [40, Theorem 7.2]). However, the inexact-Newton analysis does not apply to the case where the right-hand side (i.e., the function itself) is known only approximately, and it is therefore not possible to know a priori the accuracy required to achieve an inexact Newton-type convergence rate. This is the situation that we are faced with if (LS_τ) is solved approximately. As we show below, with only approximate knowledge of the function value ϕ, this inexact version of Newton's method still converges, although the convergence rate is sublinear. The rate can be made arbitrarily close to superlinear by increasing the accuracy with which we compute ϕ.

3.1. Approximate primal-dual solutions. In this section we use the duality gap to derive an easily computable expression that bounds the accuracy of the computed function value of ϕ. The algorithm for solving (LS_τ) outlined in section 4 maintains feasibility of the iterates at all iterations. Thus, an approximate solution $\bar{x}_\tau$ and its corresponding residual $\bar{r}_\tau := b - A\bar{x}_\tau$ satisfy

$$\|\bar{x}_\tau\|_1 \le \tau \quad \text{and} \quad \|\bar{r}_\tau\|_2 \ge \|r_\tau\|_2 > 0, \tag{3.2}$$

where the second set of inequalities holds because $\bar{x}_\tau$ is suboptimal and $\tau < \tau_{\text{BP}}$. We can thus construct the approximations

$$\bar{y}_\tau := \bar{r}_\tau/\|\bar{r}_\tau\|_2 \quad \text{and} \quad \bar{\lambda}_\tau := \|A^T\bar{y}_\tau\|_\infty$$

to the dual variables that are dual feasible, i.e., they satisfy (2.6b). The value of the dual problem (2.2) at any feasible point gives a lower bound on the optimal value $\|r_\tau\|_2$, and the value of the primal problem (2.1) at any feasible point gives an upper bound on the optimal value. Therefore,

$$b^T\bar{y}_\tau - \tau\bar{\lambda}_\tau \le \|r_\tau\|_2 \le \|\bar{r}_\tau\|_2. \tag{3.3}$$

We use the duality gap

$$\delta_\tau := \|\bar{r}_\tau\|_2 - (b^T\bar{y}_\tau - \tau\bar{\lambda}_\tau) \tag{3.4}$$

to measure the quality of an approximate solution $\bar{x}_\tau$. By (3.3), δ_τ is necessarily nonnegative.

Let $\bar{\phi}(\tau) := \|\bar{r}_\tau\|_2$ be the objective value of (LS_τ) at the approximate solution $\bar{x}_\tau$. The duality gap at $\bar{x}_\tau$ provides a bound on the difference between $\phi(\tau)$ and $\bar{\phi}(\tau)$. If we additionally assume that A is full rank (so that its condition number is bounded), we can also use δ_τ to provide a bound on the difference between the derivatives $\phi'(\tau)$ and $\bar{\phi}'(\tau)$. From (3.3)–(3.4) and from Theorem 2.1(b), for all $\tau \in (0, \tau_{\text{BP}})$,

$$\bar{\phi}(\tau) - \phi(\tau) < \delta_\tau \quad \text{and} \quad |\bar{\phi}'(\tau) - \phi'(\tau)| < \gamma\delta_\tau \tag{3.5}$$

for some positive constant γ that is independent of τ. It follows from the definition of ϕ' and from standard properties of matrix norms that γ is proportional to the condition number of A.

3.2. Local convergence rate. The following theorem establishes the local convergence rate of an inexact Newton method for (1.2) where ϕ and ϕ' are known only approximately.

THEOREM 3.1. *Suppose that A has full rank, $\sigma \in (0, \|b\|_2)$, and $\delta_k := \delta_{\tau_k} \to 0$. Then if τ_0 is close enough to τ_σ, the iteration (3.1)—with ϕ and ϕ' replaced by $\bar{\phi}$ and $\bar{\phi}'$—generates a sequence $\tau_k \to \tau_\sigma$ that satisfies*

$$|\tau_{k+1} - \tau_\sigma| = \gamma\delta_k + \eta_k|\tau_k - \tau_\sigma|, \tag{3.6}$$

where $\eta_k \to 0$ and γ is a positive constant.

Proof. Because $\phi(\tau_\sigma) = \sigma \in (0, \|b\|_2)$, (2.7) implies that $\tau_\sigma \in (0, \tau_{\text{BP}})$. By Theorem 2.1 we have that $\phi(\tau)$ is continuously differentiable for all τ close enough to τ_σ, and so by Taylor's theorem,

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

where the remainder ω satisfies

$$\omega(\tau_k, \tau_\sigma)/|\tau_k - \tau_\sigma| \to 0 \quad \text{as} \quad |\tau_k - \tau_\sigma| \to 0. \tag{3.7}$$

By (3.5) and because (3.2) holds for $\tau = \tau_k$, there exist positive constants γ_1 and γ_2, independent of τ_k, such that

$$\left| \frac{\phi(\tau_k) - \sigma}{\phi'(\tau_k)} - \frac{\bar{\phi}(\tau_k) - \sigma}{\bar{\phi}'(\tau_k)} \right| \le \gamma_1\delta_k \quad \text{and} \quad |\phi'(\tau_k)|^{-1} < \gamma_2.$$

Then, because $\Delta\tau_k = \big(\sigma - \bar{\phi}(\tau_k)\big)/\bar{\phi}'(\tau_k)$,

$$\begin{aligned}
|\tau_{k+1} - \tau_\sigma| &= |\tau_k - \tau_\sigma + \Delta\tau_k| \\
&= \left| -\frac{\bar{\phi}(\tau_k) - \sigma}{\bar{\phi}'(\tau_k)} + \frac{1}{\phi'(\tau_k)}\big[\phi(\tau_k) - \sigma - \omega(\tau_k, \tau_\sigma)\big] \right| \\
&\le \left| \frac{\phi(\tau_k) - \sigma}{\phi'(\tau_k)} - \frac{\bar{\phi}(\tau_k) - \sigma}{\bar{\phi}'(\tau_k)} \right| + \left| \frac{\omega(\tau_k, \tau_\sigma)}{\phi'(\tau_k)} \right| \\
&= \gamma_1\delta_k + \gamma_2|\omega(\tau_k, \tau_\sigma)| \\
&= \gamma_1\delta_k + \eta_k|\tau_k - \tau_\sigma|,
\end{aligned}$$

where $\eta_k := \gamma_2|\omega(\tau_k, \tau_\sigma)|/|\tau_k - \tau_\sigma|$. With τ_k sufficiently close to τ_σ, (3.7) implies that $\eta_k < 1$. Apply the above inequality recursively $\ell \ge 1$ times to obtain

$$|\tau_{k+\ell} - \tau_\sigma| \le \gamma_1 \sum_{i=1}^{\ell} (\gamma_1)^{\ell-i}\delta_{k+i-1} + (\eta_k)^\ell |\tau_k - \tau_\sigma|,$$

and because $\delta_k \to 0$ and $\eta_k < 1$, it follows that $\tau_{k+\ell} \to \tau_\sigma$ as $\ell \to \infty$. Thus $\tau_k \to \tau_\sigma$, as required. By again applying (3.7), we have that $\eta_k \to 0$. ∎

Note that if (LS_τ) is solved exactly at each iteration, such that $\delta_k = 0$, then Theorem 3.1 shows that the convergence rate is superlinear, as we expect of a standard Newton iteration. In effect, the convergence rate of the algorithm depends on the rate at which $\delta_k \to 0$. If A is rank deficient, then the constant γ in (3.6) is infinite; we thus expect that ill-conditioning in A leads to slow convergence unless $\delta_k = 0$, i.e., ϕ is evaluated accurately at every iteration.

Algorithm 1: Spectral projected gradient for (LS_τ).

Input: x, τ, δ
Output: x_τ, r_τ

Set minimum and maximum step lengths $0 < \alpha_{\min} < \alpha_{\max}$.
Set initial step length $\alpha_0 \in [\alpha_{\min}, \alpha_{\max}]$ and sufficient descent parameter $\gamma \in (0, 1)$.
Set an integer line search history length $M \geq 1$.
Set initial iterates: $x_0 \leftarrow P_\tau[x],\ r_0 \leftarrow b - Ax_0,\ g_0 \leftarrow -A^T r_0$.
$\ell \leftarrow 0$
begin

1 $\quad \delta_\ell \leftarrow \|r_\ell\|_2 - (b^T r_\ell - \tau\|g_\ell\|_\infty)/\|r_\ell\|_2$ `[compute duality gap]`
2 $\quad$ **if** $\delta_\ell < \delta$ **then** break `[exit if converged]`
3 $\quad \alpha \leftarrow \alpha_\ell$ `[initial step length]`
$\quad$ **begin**
4 $\quad\quad \bar{x} \leftarrow P_\tau[x_\ell - \alpha g_\ell]$ `[candidate line search iterate]`
5 $\quad\quad \bar{r} \leftarrow b - A\bar{x}$ `[update the corresponding residual]`
6 $\quad\quad$ **if** $\|\bar{r}\|_2^2 \leq \max_{j\in[0,\min\{k,M-1\}]} \|r_{\ell-j}\|_2^2 + \gamma(\bar{x} - x_\ell)^T g_\ell$ **then**
7 $\quad\quad\quad$ break `[exit line search]`
$\quad\quad$ **else**
8 $\quad\quad\quad \alpha \leftarrow \alpha/2$ `[decrease step length]`
$\quad$ **end**
9 $\quad x_{\ell+1} \leftarrow \bar{x},\ r_{\ell+1} \leftarrow \bar{r},\ g_{\ell+1} \leftarrow -A^T r_{\ell+1}$ `[update iterates]`
10 $\quad \Delta x \leftarrow x_{\ell+1} - x_\ell,\ \Delta g \leftarrow g_{\ell+1} - g_\ell$
11 $\quad$ **if** $\Delta x^T \Delta g \leq 0$ **then** `[Update the Barzilai-Borwein step length]`
12 $\quad\quad \alpha_{\ell+1} \leftarrow \alpha_{\max}$
$\quad$ **else**
13 $\quad\quad \alpha_{\ell+1} \leftarrow \min\{\alpha_{\max}, \max[\alpha_{\min}, (\Delta x^T \Delta x)/(\Delta x^T \Delta g)]\}$
$\quad \ell \leftarrow \ell + 1$
end
return $x_\tau \leftarrow x_\ell$, $r_\tau \leftarrow r_\ell$

4. Solving the Lasso problem (evaluating ϕ). Each iteration of the Newton root-finding method described in section 3 requires the (approximate) evaluation of $\phi(\tau)$, and therefore a procedure for minimizing (LS_τ). In this section we outline an SPG algorithm for this purpose.

4.1. Spectral projected gradient. The SPG procedure that we use for solving (LS_τ) closely follows Birgin, Martínez, and Raydan [5, Algorithm 2.1], and is outlined in Algorithm 1. The method depends on the ability to project iterates onto the feasible set $\{x \mid \|x\|_1 \leq \tau\}$. This is accomplished via the operator

$$P_\tau[c] := \Big\{ \arg\min_x \ \|c - x\|_2 \ \text{ subject to } \ \|x\|_1 \leq \tau \Big\}, \tag{4.1}$$

which gives the projection of an n-vector c onto the one-norm ball with radius τ.

Each iteration of the algorithm searches the projected gradient path $P_\tau[x_\ell - \alpha g_\ell]$, where g_ℓ is the current gradient for the function $\|Ax - b\|_2^2$ (which is the square of the (LS_τ) objective); see steps 4–8. Because the feasible set is polyhedral, the projected gradient path is piecewise linear. The criterion used in step 6 results in a nonmonotone line search, which ensures that at least every M iterations yield a sufficient decrease in the objective function.

The initial candidate iterate in step 4 is determined by the step length computed in steps 10–13. Birgin, Martínez, and Raydan [5, Algorithm 2.1] relate this step length, introduced by Barzilai and Borwein [1], to the eigenvalues of the Hessian of the objective. (In this case, the Hessian is A^TA.) They prove that the method is globally convergent. The effectiveness in practice of the scaling suggested by Barzilai and Borwein has led many researchers to continue to explore enhancements to this choice of step length; for examples, see Dai and Fletcher [18] and Dai et al. [17]. The method proposed by Daubechies, Fornasier, and Loris [20] is related to Algorithm 1.

4.2. One-norm projection. There are three potentially expensive steps in Algorithm 1: steps 5 and 9 compute the matrix vector products Ax and A^Tr, and step 4 computes the projection $P_\tau[\cdot]$ of a candidate iterate. In this section we give an algorithm for computing the projection defined in (4.1). The algorithm has a worst-case complexity of $\mathcal{O}(n \log n)$, but numerical experiments presented in section 6 suggest that the overall work on average is much less than for the worst case.

In order to simplify the following discussion, we assume that the entries of the n-vector c are nonnegative. This does not lead to any loss of generality: note that if the entries of c had different signs, then it would be possible to replace the objective in (4.1) with the equivalent objective $\|Dc - Dx\|_2$, where the diagonal matrix $D = \mathrm{diag}(\mathrm{sgn}(c))$. The true solution can then recovered by applying D^{-1}.

Our algorithm for solving (4.1) is motivated as follows. We begin with the trial solution $x \leftarrow c$. If this is feasible for (4.1), then we exit immediately with $P_\tau[c] := x^* = c$. Otherwise, we attempt to decrease the norm of the trial x by

$$\nu := \|x\|_1 - \tau, \tag{4.2}$$

which is the amount of infeasibility. Therefore, we must find a vector d such that $\|x - d\|_1 = \tau$ and, in order to minimize the potential increase in the objective value, choose d so that $\|d\|_2$ is minimal. The correction d must therefore solve

$$\underset{d}{\text{minimize}} \quad \|d\|_2 \quad \text{subject to} \quad d \geq 0,\ \|d\|_1 = \nu.$$

It is straightforward to verify that

$$d^* = \gamma e, \quad \text{with} \quad \gamma = \nu/n, \tag{4.3}$$

is a solution of this subproblem.

However, we cannot exit with $x \leftarrow c - d^*$ if some of these entries are negative because doing so increases the value of $\|x\|_1$—i.e., the projection must preserve the sign pattern of c. Therefore,

$$\text{if each } d_i^* < c_{\min} := \min_i c_i, \quad \text{set } x \leftarrow c - d^* \tag{4.4}$$

and exit with the solution of (4.1). Otherwise, we enforce

$$x_i = 0 \quad \text{for all} \quad i \in \mathcal{I} := \{i \mid d_i^* \geq c_{\min}\}, \tag{4.5}$$

and then recursively repeat the process described above for the remaining variables $\{1, \ldots, n\} \backslash \mathcal{I}$.

Algorithm 2 is a distillation of this procedure. In order to make it efficient and to reduce overhead due to bookkeeping, we apply the procedure to a sequence of subelements of c: the first iteration starts with a single element that is largest in

Algorithm 2: Real projection onto the set $\{x \in \mathbb{R}^n \mid \|x\|_1 \le \tau\}$.

Input: c, τ
Output: x

1 **if** $\|c\|_1 \le \tau$ **then return** c [quick exit if c is feasible]
2 $\gamma \leftarrow 0,\ \delta \leftarrow 0,\ \nu \leftarrow -\tau$ [initialization]
3 $\bar{c} \leftarrow$ `BuildHeap`$(|c|)$ [$\bar{c}$ is a heapified copy of the absolute value of c]
4 **for** $j \leftarrow 1$ **to** n **do**
5 $\quad c_{\min} \leftarrow \bar{c}[1]$ [extract the next largest element from $\bar{c}$]
6 $\quad \nu \leftarrow \nu + c_{\min}$ [accumulate the infeasibility; see (4.2)]
7 $\quad \gamma \leftarrow \nu/j$ [define the current solution of (4.2); see (4.3)]
8 $\quad$ **if** $\gamma \ge c_{\min}$ **then** break [remaining iterations all satisfy (4.5)]
9 $\quad \bar{c} \leftarrow$ `DeleteMax`$(\bar{c})$ [remove $x_{\max}$ from the heap and re-heapify]
10 $\quad \delta \leftarrow \gamma$ [record the latest solution of (4.3)]
11 $x \leftarrow$ `SoftThreshold`$(c,\ \delta)$ [soft-threshold input vector c; see (4.6)]
return x

magnitude, and each subsequent iteration adds one more element that is next largest in magnitude. This approach avoids having to sort the entire vector c. Instead, step 3 of Algorithm 2 uses `BuildHeap` to build a binomial heap structure in which the first element of the heap is largest in magnitude. The cost of `BuildHeap` is $\mathcal{O}(n)$, where n is the length of the vector c. The cost in subsequent iterations is dominated by step 9, where the function `DeleteMax` removes the current largest element from the heap and restores the heap property. The scalar $c_{\min}$, set in step 5, is the smallest element of the current subvector; cf. (4.4). (At iteration j, $c_{\min}$ corresponds to the element of the overall vector c that is jth greatest in magnitude.) If step 8 tests true, then the algorithm can exit the for-loop because all remaining iterates must satisfy (4.5).

The concluding step uses rules (4.4) and (4.5) to generate the final solution. This is accomplished in step 11 by applying the soft-thresholding operation

$$x \leftarrow \texttt{SoftThreshold}(c,\delta) \quad \Longleftrightarrow \quad x_i \leftarrow \operatorname{sgn}(c_i) \cdot \max\{0, |c_i| - \delta\} \tag{4.6}$$

componentwise to the original vector c; the scalar δ is obtained in step 10. This operation damps elements that are larger in magnitude than δ, and sets to zero any elements that are smaller in magnitude than δ.

In the worst case, Algorithm 2 will proceed for the full n iterations, and the dominant cost is n calls to `DeleteMax`. The overall cost of the algorithm is therefore $\mathcal{O}(n \log n)$ in the worst case.

Note that the soft-thresholding operator we have just defined yields the solution of the separable convex optimization problem

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|x - c\|_2^2 + \delta\|x\|_1, \tag{4.7}$$

as shown by Chambolle et al. [13, section III]. In this light, Algorithm 2 can be interpreted as a procedure for finding the optimal dual variable δ associated with the constraint in (4.1).

4.3. Complex one-norm projection. Chambolle et al.'s [13] derivation of the soft-thresholding operation (4.6) as a solution of (4.7) applies to the case in which the minimization is done over the complex domain. If $c \in \mathbb{C}^n$, then the soft-thresholding operation damps the modulus of each element of c that is larger than δ and sets to zero any elements of c that have modulus smaller than δ. (When applied to a complex

Algorithm 3: Complex projection onto the set $\{z \in \mathbb{C}^n \mid \|z\|_1 \le \tau\}$.

Input: c, τ
Output: z

1 $r \leftarrow |c|$ [compute the componentwise modulus of c]
2 $\bar{r} \leftarrow P_\tau[r]$ [apply Algorithm 2; see (4.1)]
foreach $i = 1$ **to** n **do**
3 **if** $r_i > 0$ **then** $z_i \leftarrow c_i(\bar{r}_i/r_i)$ [compute $\operatorname{sgn}(c_i) \cdot \bar{r}_i$; see (4.6)]
4 **else** $z_i \leftarrow 0$ [the element c_i was zero; keep it]
return z

number z, the signum function $\operatorname{sgn}(z) = z/|z|$ projects z onto the unit circle of the complex domain; by convention, $\operatorname{sgn}(0) = 0$.) Thus, the thresholding operation on c acts only on the modula of each component, leaving the phases untouched.

We can use Algorithm 2 (projection of a real vector) to bootstrap an efficient algorithm for projection of complex vectors. Algorithm 3 outlines this approach. First, the vector of modula $r = (|c_1| \cdots |c_n|)$ is computed (step 1) and then is projected onto the (real) one-norm ball with radius τ (step 2). Next, the soft-thresholding operation of (4.6) is applied in steps 3–4.

The SPG method outlined in Algorithm 1 requires only an efficient procedure for the projection onto the convex feasible set—the form of that convex set has no effect on the rest of the algorithm. An important benefit of this is that, with the real- and complex-projection Algorithms 2 and 3, the SPG method applies equally well to problems in the real and complex domains.

5. Implementation. The methods that we describe in this paper have been implemented as a single MATLAB [38] software package called SPGL1. It implements the root-finding algorithm described in section 3 and the spectral projected-gradient algorithm described in section 4; the latter is, in fact, the computational kernel.

The SPGL1 implementation is structured around major and minor iterations. Each major iteration is responsible for determining the next element of the sequence $\{\tau_k\}$ and for invoking the SPG method described in Algorithm 1 to determine approximate values of $\phi(\tau_k)$ and $\phi'(\tau_k)$. For each major iteration k, there is a corresponding set of minor iterates converging to (x_{τ_k}, r_{τ_k}) that comprise the iterates of Algorithm 1. Unless the user can provide a good estimate for the solution τ_σ of (1.2), the root-finding algorithm chooses $\tau_0 = 0$. This leads to an essentially "free" first major iteration because $\phi(0) = \|b\|_2$ and $\phi'(0) = \|A^T b\|_\infty$; with (3.1), it holds immediately that the next Newton iterate $\tau_1 = (\sigma - \|b\|_2)/\|A^T b\|_\infty$ exactly.

A small modification to Algorithm 1 is needed before we can implement the inexact Newton method of the outer iterations. Instead of using a fixed threshold δ, we compare the duality-gap test in step 2 to the current relative error in satisfying (1.2). Thus, the (LS_τ) subproblem is solved to low accuracy during early major iterations (when τ_σ in known only roughly), but more accurately as the error in satisfying (1.2) decreases. If only the solution of (LS_τ) is required, then a single call to Algorithm 1 is made with δ held at some fixed value.

6. Numerical experiments. This section summarizes a series of numerical experiments in which we apply SPGL1 to basis pursuit, basis pursuit denoise (BP_σ), and Lasso (LS_τ) problems. The experiments include a selection of sixteen relevant problems from the Sparco collection [2] of test problems. The chosen problems are all real-valued and suited to one-norm regularization. We exclude problems that are

TABLE 6.1
Key to symbols used in Tables 6.2–6.4.

m, n	number of rows and columns of A
$\|b\|_2$	two-norm of the right-hand-side vector
$\|r\|_2$	two-norm of the computed residual
$\|x\|_1$	one-norm of the computed solution
nnz(x)	number of nonzeros in the computed solution; see (6.1)
nMat	total number of matrix-vector products with A and A^T
f	solver failed to return a feasible solution
t	solver failed to converge within allotted CPU time

TABLE 6.2
The Sparco test problems used.

Problem	ID	m	n	$\|b\|_2$	Operator
`blocksig`	2	1024	1024	7.9e+1	wavelet
`blurrycam`	701	65536	65536	1.3e+2	blurring, wavelet
`blurspike`	702	16384	16384	2.2e+0	blurring
`cosspike`	3	1024	2048	1.0e+2	DCT
`dcthdr`	12	2000	8192	2.3e+3	restricted DCT
`finger`	703	11013	125385	5.5e+1	2D curvelet
`gcosspike`	5	300	2048	8.1e+1	Gaussian ensemble, DCT
`jitter`	902	200	1000	4.7e−1	DCT
`p3poly`	6	600	2048	5.4e+3	Gaussian ensemble, wavelet
`seismic`	901	41472	480617	1.1e+2	2D curvelet
`sgnspike`	7	600	2560	2.2e+0	Gaussian ensemble
`soccer1`	601	3200	4096	5.5e+4	binary ensemble, wavelet
`spiketrn`	903	1024	1024	5.7e+1	1D convolution
`srcsep1`	401	29166	57344	2.2e+1	windowed DCT
`srcsep2`	402	29166	86016	2.3e+1	windowed DCT
`yinyang`	603	1024	4096	2.5e+1	wavelet

complex-valued, that are better suited to total-variation regularization, or that are pathological examples designed only for debugging codes. Each problem in the collection includes a linear operator A and a right-hand-side vector b.

Table 6.1 defines the symbols used in Tables 6.2–6.4. The quantity nnz(x) counts the number of nonzero entries in the vector x. Because it can be difficult to judge whether small entries in a solution vector are significantly different from zero, we compute the number of nonzeros in a vector x as the minimum number of entries that carry 99.9% of the one-norm of the vector, i.e.,

$$\text{nnz}(x) = \{\min\ r \text{ such that } \textstyle\sum_i^r |x_{\lfloor i \rfloor}| \geq 0.999 \cdot \|x\|_1\}, \tag{6.1}$$

where $|x_{\lfloor 1 \rfloor}| \geq \cdots \geq |x_{\lfloor n \rfloor}|$ are the n elements of x sorted by absolute value.

Table 6.2 summarizes the selected problems: following the problem name and Sparco ID are the number of rows (m) and columns (n) of A, and the two-norm of b. The last column is a brief description of A, which is often a compound operator. The operator's *wavelet*, *DCT* (discrete cosine transform), *blurring*, *curvelet*, and *convolution* are all available implicitly, and the products Ax and A^Ty are computed using fast algorithms. The operators *Gaussian ensemble* and *binary ensemble* are explicit matrices with normally distributed random entries and random zero-one entries, respectively.

All experiments reported in this section were run on a Mac Pro (with two 3GHz dual-core Intel Xeon processors and 4Gb of RAM) running MATLAB 7.5. Each attempt to solve a problem was limited to one hour of CPU time. The data files and MATLAB scripts used to generate all of the numerical results presented in the following subsections can be obtained at http://www.cs.ubc.ca/labs/scl/spgl1.

TABLE 6.3
Basis pursuit denoise comparisons.

	Homotopy				SPGL1			
Problem	$\|r\|_2$	$\|x\|_1$	nnz(x)	nMat	$\|r\|_2$	$\|x\|_1$	nnz(x)	nMat
`blocksig`	7.9e+0	3.8e+2	64	230	7.9e+0	3.8e+2	64	21
	7.9e−2	4.5e+2	71	246	7.9e−2	4.5e+2	71	22
`blurrycam`	1.3e+1	1.2e+3	298	1290	1.3e+1	1.2e+3	299	34
	t	t	t	t	1.3e−1	9.1e+3	59010	2872
`blurspike`	2.2e−1	1.5e+2	175	770	2.2e−1	1.5e+2	181	264
	t	t	t	t	2.3e−3	3.4e+2	15324	3238
`cosspike`	1.0e+1	1.3e+2	2	8	1.0e+1	1.3e+2	2	31
	1.0e−1	2.2e+2	113	496	1.0e−1	2.2e+2	113	77
`dcthdr`	2.3e+2	3.7e+4	133	568	2.3e+2	3.7e+4	133	34
	2.3e+0	4.4e+4	266	1436	2.3e+0	4.4e+4	266	114
`finger`	t	t	t	t	5.5e+0	4.4e+3	7968	252
	t	t	t	t	5.5e−2	5.5e+3	13564	1128
`gcosspike`	8.1e+0	1.3e+2	4	20	8.1e+0	1.3e+2	4	34
	8.1e−2	1.8e+2	94	882	8.1e−2	1.8e+2	112	434
`jitter`	4.7e−2	1.6e+0	3	12	4.7e−2	1.6e+0	3	20
	4.7e−4	1.7e+0	3	12	5.3e−4	1.7e+0	3	30
`p3poly`	5.4e+2	1.4e+3	155	708	5.4e+2	1.4e+3	155	68
	8.5e+1	1.7e+3	494	3364	5.4e+0	1.7e+3	518	478
`seismic`	1.1e+1	2.8e+3	554	4172	1.1e+1	2.8e+3	646	141
	t	t	t	t	1.1e−1	3.9e+3	14806	709
`sgnspike`	2.2e−1	1.8e+1	20	80	2.2e−1	1.8e+1	20	30
	2.2e−3	2.0e+1	20	80	2.2e−3	2.0e+1	20	44
`soccer1`	5.5e+3	5.5e+1	4	16	5.5e+3	5.5e+1	4	14
	5.5e+1	3.1e+2	769	3460	5.5e+1	3.1e+2	1073	1250
`spiketrn`	5.7e+0	1.0e+1	51	310	5.7e+0	1.0e+1	73	617
	5.7e−2	1.3e+1	35	480	5.7e−2	1.3e+1	418	4761
`srcsep1`	t	t	t	t	2.2e+0	8.0e+2	7838	160
	t	t	t	t	2.2e−2	1.0e+3	22428	1125
`srcsep2`	t	t	t	t	2.3e+0	8.6e+2	8652	246
	t	t	t	t	2.3e−2	1.1e+3	25359	947
`yinyang`	2.5e+0	1.8e+2	153	668	2.5e+0	1.8e+2	153	44
	2.5e−2	2.6e+2	881	4332	2.6e−2	2.6e+2	969	396

6.1. Basis pursuit denoise. For each of the sixteen test problems listed in Table 6.2, we generate two values of σ: $\sigma_1 = 10^{-1}\|b\|_2$ and $\sigma_2 = 10^{-3}\|b\|_2$. We thus have a total of thirty-two test problems of the form (BP_σ) to which we apply SPGL1.

As a benchmark, we also give results for the `SolveLasso` solver available within the SPARSELAB package, which we apply in its "lasso" mode (there is also an optional "lars" mode). In this mode, `SolveLasso` applies the homotopy method described in section 1.4 to solve (QP_λ) for all values of $\lambda \geq 0$. The `SolveLasso` solver begins with $\lambda = \|A^T b\|_\infty$, which has the corresponding trivial solution $x_\lambda = 0$ and reduces λ in stages so that the corresponding sequence of solutions x_λ have exactly one additional nonzero entry each. The norm of the residual $r_\lambda \equiv b - Ax_\lambda$ decreases monotonically. We set the parameters `resStop` $= \sigma$ and `lamStop` $= 0$, which together require `SolveLasso` to iterate until $\|r_\lambda\|_2 \leq \sigma$, and x_λ is therefore feasible for (BP_σ).

Table 6.3 summarizes the results. The rows in each two-row block correspond to instances of the same test problem (i.e., the same A and b) with parameters σ_1 and σ_2. The SPARSELAB results are shown under the column head "Homotopy." SPARSELAB did not obtain accurate results for nine problems—`blurrycam(2)`, `blurspike(2)`, `finger(1,2)`, `seismic(2)`, `srcsep1(1,2)`, and `srcsep2(1,2)` (the numbers in parentheses refer to the particular problem instance)—because it failed to converge within the allotted time. For these problems, the number of nonzero elements in the current

TABLE 6.4
Basis pursuit comparisons.

	PDCO		Homotopy		SPGL1	
Problem	$\|r\|_2$	$\|x\|_1$	$\|r\|_2$	$\|x\|_1$	$\|r\|_2$	$\|x\|_1$
blocksig	3.3e−04	4.5e+02	1.0e−04	4.5e+02	2.0e−14	4.5e+02
blurrycam	t	t	t	t	9.9e−05	1.0e+04
blurspike	9.1e−03	3.4e+02	t	t	9.9e−05	3.5e+02
cosspike	1.6e−04	2.2e+02	1.0e−04	2.2e+02	8.6e−05	2.2e+02
dcthdr	1.3e−05	4.4e+04	5.5e−08	4.4e+04	4.9e−05	4.4e+04
finger	t	t	t	t	8.2e−05	5.5e+03
gcosspike	1.9e−05	1.8e+02	1.0e−04	1.8e+02	9.9e−05	1.8e+02
jitter	1.3e−05	1.8e+00	1.0e−04	1.7e+00	5.3e−05	1.7e+00
p3poly	4.6e−02	1.7e+03	8.5e+01^f	1.8e+03	9.5e−05	1.7e+03
seismic	t	t	t	t	8.6e−05	3.9e+03
sgnspike	9.3e−06	2.0e+01	1.0e−04	2.0e+01	8.0e−05	2.0e+01
soccer1	t	t	t	t	1.0e−04	4.2e+02
spiketrn	3.6e−03	1.3e+01	1.0e−04	1.3e+01	9.9e−05	1.3e+01
srcsep1	8.2e−03	1.1e+03	t	t	8.6e−05	1.1e+03
srcsep2	5.5e−03	1.1e+03	t	t	1.0e−04	1.1e+03
yinyang	1.4e−03	2.6e+02	2.8e−03^f	4.7e+02	9.6e−05	2.6e+02

	PDCO		Homotopy		SPGL1	
Problem	nnz(x)	nMat	nnz(x)	nMat	nnz(x)	nMat
blocksig	71	703	71	246	71	21
blurrycam	t	t	t	t	62756	8237
blurspike	15513	59963	t	t	15585	5066
cosspike	119	2471	115	500	115	111
dcthdr	270	79911	270	1436	270	294
finger	t	t	t	t	13333	3058
gcosspike	335	14755	59	934	195	2535
jitter	678	43	3	12	3	38
p3poly	559	145053	503^f	3882	526	3047
seismic	t	t	t	t	22816	3871
sgnspike	1018	131	20	80	20	56
soccer1	t	t	t	t	3805	63233
spiketrn	30	1169	12	480	12	26406
srcsep1	40950	78385	t	t	24641	2881
srcsep2	67029	47109	t	t	26653	2432
yinyang	1733	34667	981^f	9680	1031	1198

solution vector grows very large. SPARSELAB maintains a dense factorization of the submatrix of A corresponding to these indices, and the time needed to update this factorization can grow very large as the nonzero index set grows. In contrast, the homotopy method can be very efficient when the number of nonzeros in the solution is small. SPGL1 succeeds in obtaining solutions to every problem instance, and we note that the memory requirements are constant throughout all iterations.

The quadratically constrained solver `l1qc_newton`, available within the ℓ_1-MAGIC software package [9], is also applicable to the problem (BP$_\sigma$) when A is only available as an operator. However, we do not include the results obtained with ℓ_1-MAGIC because it failed on all but ten of the thirty-two test problems—the solver either reported "stuck on cone iterations" (i.e., it could not compute a sufficiently accurate search direction within a predetermined number of conjugate-gradient iterations) or failed to return a solution within the allotted one hour of CPU time. Other quadratically constrained solvers such as MOSEK and SEDUMI are not applicable because they all require A as an explicit matrix.

6.2. Basis pursuit. The basis pursuit problem can be considered to be a special case of (BP$_\sigma$) in which $\sigma = 0$. In this section we give the results of experiments that

show that SPGL1 can be an effective algorithm for solving (BP). We apply SPGL1 to sixteen (BP) problems generated from the test set shown in Table 6.2.

As benchmarks, we also give the results of applying the solvers `SolveLasso` and `SolveBP`, which are both available within the SparseLab package. For `SolveLasso`, we again use its "lasso" mode and set the parameters `resStep` $= 10^{-4}$ and `lamStep` $= 0$. We use default parameters for `SolveBP`. Note that `SolveBP` applies the interior-point code PDCO [44] to a linear programming reformulation of (BP) and requires only matrix-vector products with A and A^T.

The top half of Table 6.4 lists the norms of the computed norms and residuals, and the bottom half lists the sparsity of the computed solutions and the number of matrix-vector products required. Both PDCO and the homotopy approach failed to converge within the allotted one hour of CPU time on problems `blurrycam`, `finger`, `seismic`, and `soccer1`. Homotopy had the same failure for problems `blurryspike`, `srcsep1`, and `srcsep2`, and in addition it failed to return feasible solutions to within the required tolerance for `p3poly` and `yinyang`. SPGL1 succeeded in all cases. Again, for problems that have very sparse solutions, homotopy can be much more efficient than SPGL1; for example, see problems `gcosspike` and `spiketrn` in Table 6.4.

Figure 6.1 shows the results of applying SPGL1 to problem `seismic`. The corrupted seismic image in Figure 6.1(a) is missing 35% of its traces (i.e., measurements); the interpolated image in Figure 6.1(b) is computed from the solution of (BP), where A is a restricted curvelet transform. Figure 6.1(c) shows a graph of the Pareto curve for this problem and the trajectory that SPGL1 follows to arrive at a (BP) solution.

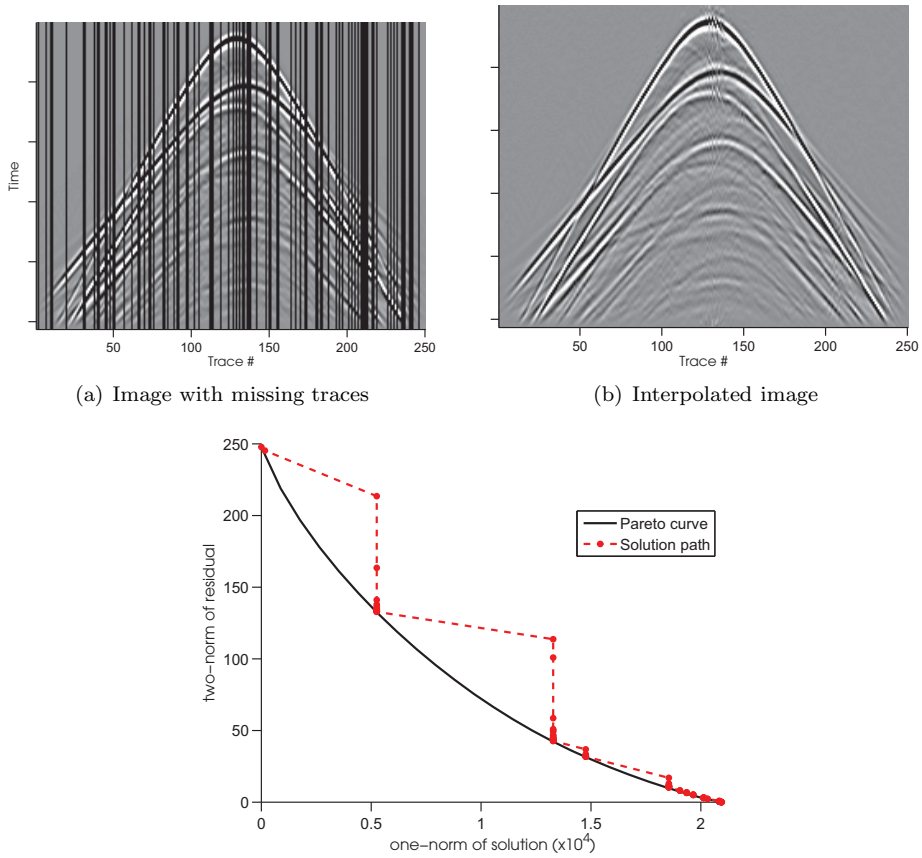
6.3. The Lasso and quadratic programs. The main goal of this work is to provide an efficient algorithm for (BP_σ). But for interest, we show here how SPGL1 can also be used to efficiently solve a single instance of the Lasso problem (LS_τ). This corresponds to solving a single instance of (QP_λ), where λ is set to the Lagrange multiplier of the Lasso constraint.

In Figure 6.2 we compare the performance and computed solutions of the solvers GPSR (version of August, 2007) [27], L1LS [34], PDCO [44], and SPGL1 applied to the problems `dcthdr` and `srcsep1`. For each of these two problems, we generate fifty values of λ and τ for which the solutions of (QP_λ) and (LS_τ) coincide, and thus generate fifty test cases each of types (QP_λ) and (LS_τ). We apply GPSR, L1LS, and PDCO to each (QP_λ) test case and apply SPGL1 to each (LS_τ) test case.

Plots (a) and (c) in Figure 6.2 show the Pareto curves for problems `dcthdr` and `srcsep1` and the norms of computed solutions versus the norms of the corresponding residuals. (Note that these curves are plotted on a semilog scale and thus are not convex.) For each point in the left-hand figure, the points in the right-hand figure give the number of matrix-vector products required to obtain the corresponding solution. Points on the Pareto curve are accurate solutions; inaccurate solutions lie above the curve, indicating that they do not solve the corresponding problem (QP_λ) or (LS_τ).

In Figure 6.2(a), almost all points lie on the Pareto curve, and thus most of the computed solutions are accurate. The corresponding points in Figure 6.2(b) indicate that SPGL1 and GPSR use only a small fraction of the matrix-vector products required by PDCO and L1LS, which are second-order methods. We note that solutions for problem `dcthdr` can range over four orders of magnitude, and yet the first-order methods appear not to be affected by this poor scaling.

In Figure 6.2(c), the computed solutions returned by PDCO and GPSR are progressively worse as λ tends to zero (and the one-norm of the solution increases). Interestingly, L1LS consistently yields very accurate solutions for all values of λ; how-



(a) Image with missing traces

(b) Interpolated image

(c) Pareto curve and solution path

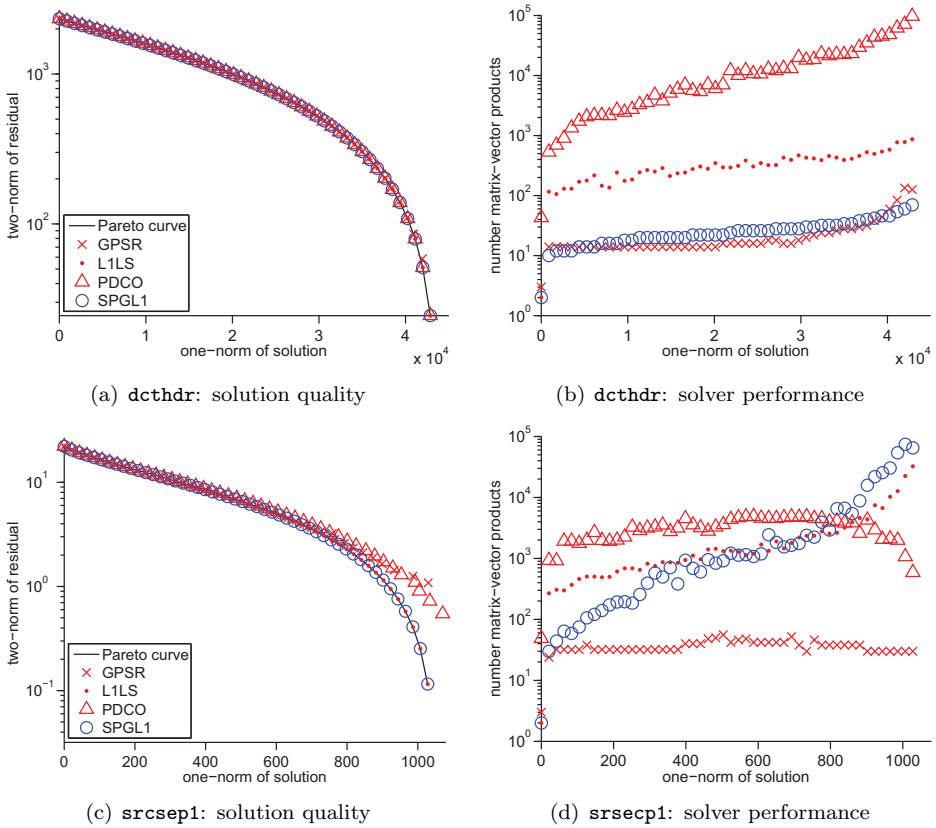
FIG. 6.1. *Corrupted and interpolated images for problem* `seismic`. *Graph* (c) *shows the Pareto curve and the solution path taken by SPGL*1.

ever, as might be expected of an interior-point method based on a conjugate-gradient linear solver, it can require many matrix-vector products.

It may be progressively more difficult to solve (QP_λ) as $\lambda \to 0$ because the regularizing effect from the one-norm term tends to become negligible, and there is less control over the norm of the solution. In contrast, the (LS_τ) formulation is guaranteed to maintain a bounded solution norm for all values of τ.

6.4. Sampling the Pareto curve. In situations where little is known about the noise level σ, it may be useful to visualize the Pareto curve in order to understand the trade-offs between the norms of the residual and the solution. In this section we aim to obtain good approximations to the Pareto curve for cases in which it is prohibitively expensive to compute it in its entirety.

We test two approaches for interpolation through a small set of samples $i = 1, \ldots, k$. In the first, we generate a uniform distribution of parameters $\lambda_i = (i/k)\|A^T b\|_\infty$ and solve the corresponding problems $(\mathrm{QP}_{\lambda_i})$. In the second, we generate a uniform distribution of parameters $\sigma_i = (i/k)\|b\|_2$ and solve the corresponding problems (BP_{σ_i}). We leverage the convexity and differentiability of the Pareto curve to approximate it with piecewise cubic polynomials that match function and derivative values at each end. When a nonconvex fit is detected, we switch to a quadratic interpolation



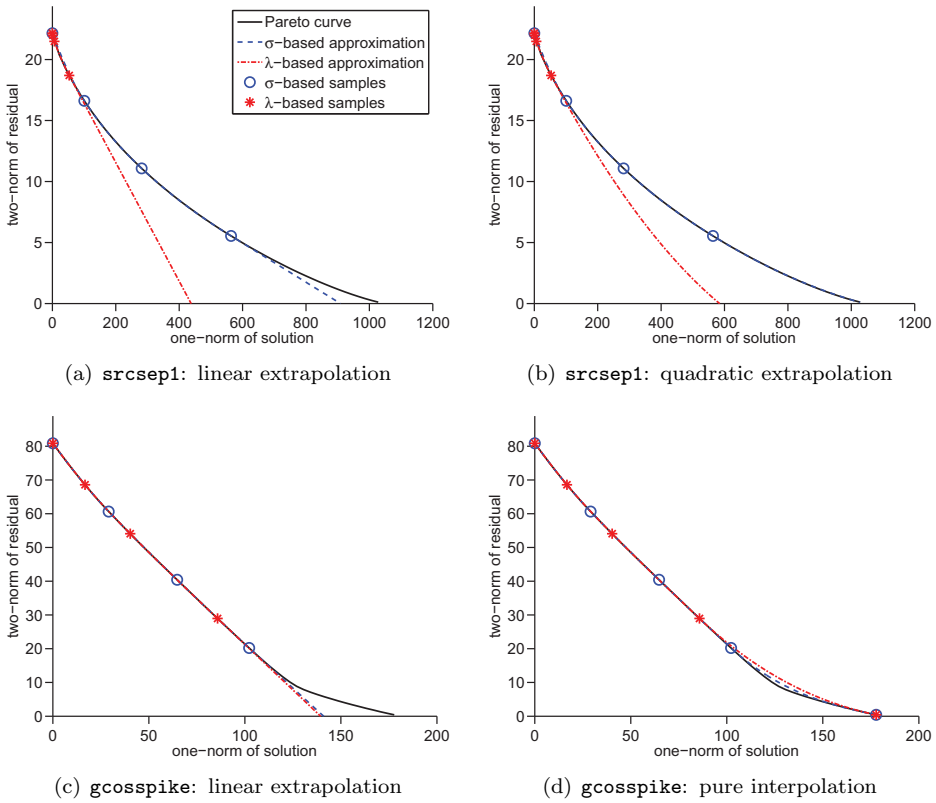
(a) `dcthdr`: solution quality (b) `dcthdr`: solver performance

(c) `srcsep1`: solution quality (d) `srsecp1`: solver performance

FIG. 6.2. *The performance of solvers on equivalent* (QP_λ) *and* (LS_τ) *problems. The top row is for problem* `dcthdr`*, and the bottom row is for problem* `srcsep1`*. The plots on the left show the norms of the computed residuals and solutions for fifty parameter values; the plots on the right give the corresponding number of required matrix-vector products. Note that the curves in the left-hand figures are not convex because they are plotted on a semilog scale.*

on the relevant interval and match derivative information only at one end. Extrapolation is based on only the function and derivative values of the last sample point. This naturally suggests that using a linear extrapolation which, due to the convexity of the Pareto curve, is guaranteed never to overestimate the curve or the value of τ_{BP}. Alternatively, we can use quadratic extrapolation through the last point and require the minimum of the extrapolation to coincide exactly with the horizontal axis. This approach works well when the Pareto curve is nearly quadratic at the end, but it is likely to overestimate the curve in other cases. Forming linear combinations of the two functions provides a balanced extrapolation that ranges from conservative to risky.

Figure 6.3 illustrates an approximation of the Pareto curve that is based on a small number of samples. Note that the sampling based on σ gives a better distribution of points on the curve as compared to sampling based on λ; this was observed for all problems tested and coincides with the observations made by Das and Dennis [19] and Leyffer [35]. For the curves shown in plots (a) and (b), the σ-based samples clearly lead to better approximations. For Pareto curves that are nearly linear, such as those shown in plots (c) and (d), there is little to no difference between the σ- and λ-based samples. For this test problem, the relatively sharp bend near the end of the curve makes it difficult to reconstruct the curve accurately unless σ or λ is



(a) `srcsep1`: linear extrapolation

(b) `srcsep1`: quadratic extrapolation

(c) `gcosspike`: linear extrapolation

(d) `gcosspike`: pure interpolation

FIG. 6.3. *Extrapolation of the Pareto curve based on samples obtained by solving* (BP_σ) *or* (QP_λ)*, respectively, at uniformly spaced values of* σ *and* λ*. Graphs* (a)–(c) *show results for linear and quadratic extrapolation; graph* (d) *shows the gains from using the* τ_{BP} *point and pure interpolation.*

sampled more finely, or unless the BP solution is sampled as illustrated in (d). An adaptive approach, such as that proposed by Leyffer [35], could lead to more accurate sampling.

7. Looking ahead. Our original motivation for developing the method proposed in this paper is its usefulness for inpainting applications in seismic imaging, where the problem sizes can stretch into millions of variables [32]. We are currently developing an out-of-core implementation of SPGL1 based on the SLIMPY [33] package.

We are also considering an extension of the root-finding approach for solving the nonlinear regularization problem

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

where f is a convex nonlinear function. This may open the door to applying the Pareto root-finding approach to other applications.

Acknowledgments. The authors are grateful to Chen Greif, Gilles Hennenfent, Felix Herrmann, Michael Saunders, and Özgür Yılmaz for their valuable input. We extend sincere thanks to two anonymous referees for their thoughtful suggestions.

REFERENCES

[1] J. Barzilai and J. M. Borwein, *Two-point step size gradient methods*, IMA J. Numer. Anal., 8 (1988), pp. 141–148.

[2] E. van den Berg, M. P. Friedlander, G. Hennenfent, F. Herrmann, R. Saab, and O. Yilmaz, *Algorithm* 890: *Sparco: A testing framework for sparse reconstruction*, ACM Trans. Math. Software, to appear.

[3] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed., Athena Scientific, Belmont, MA, 1999.

[4] D. P. Bertsekas, *Convex Analysis and Optimization*, Athena Scientific, Belmont, MA, 2003.

[5] E. G. Birgin, J. M. Martínez, and M. Raydan, *Nonmonotone spectral projected gradient methods on convex sets*, SIAM J. Optim., 10 (2000), pp. 1196–1211.

[6] E. G. Birgin, J. M. Martínez, and M. Raydan, *Inexact spectral projected gradient methods on convex sets*, IMA J. Numer. Anal., 23 (2003), pp. 1196–1211.

[7] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[8] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[9] E. Candès and J. Romberg, ℓ_1*-magic*, http://www.l1-magic.org/.

[10] E. J. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Trans. Inform. Theory, 52 (2006), pp. 489–509.

[11] E. J. Candès, J. Romberg, and T. Tao, *Stable signal recovery from incomplete and inaccurate measurements*, Comm. Pure Appl. Math., 59 (2006), pp. 1207–1223.

[12] E. J. Candès and T. Tao, *Near-optimal signal recovery from random projections: Universal encoding strategies?*, IEEE Trans. Inform. Theory, 52 (2006), pp. 5406–5425.

[13] A. Chambolle, R. De Vore, N.-Y. Lee, and B. Lucier, *Nonlinear wavelet image processing: Variational problems, compression, and noise removal through wavelet shrinkage*, IEEE Trans. Image Process., 7 (1998), pp. 319–335.

[14] S. S. Chen, D. L. Donoho, and M. A. Saunders, *Atomic decomposition by basis pursuit*, SIAM J. Sci. Comput., 20 (1998), pp. 33–61.

[15] S. S. Chen, D. L. Donoho, and M. A. Saunders, *Atomic decomposition by basis pursuit*, SIAM Rev., 43 (2001), pp. 129–159.

[16] *ILOG CPLEX. Mathematical Programming System*, http://www.cplex.com.

[17] Y. Dai, W. W. Hager, K. Schittkowski, and H. Zhang, *The cyclic Barzilai-Borwein method for unconstrained optimization*, IMA J. Numer. Anal., 7 (2006), pp. 604–627.

[18] Y.-H. Dai and R. Fletcher, *Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming*, Numer. Math., 100 (2005), pp. 21–47.

[19] I. Das and J. E. Dennis, *A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems*, Struct. Optim., 14 (1997), pp. 63–69.

[20] I. Daubechies, M. Fornasier, and I. Loris, *Accelereated projected gradient method for linear inverse problems with sparsity constraints*, J. Fourier Anal. Appl., 2007, to appear.

[21] A. Dax, *On regularized least norm problems*, SIAM J. Optim., 2 (1992), pp. 602–618.

[22] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[23] D. L. Donoho, *Compressed sensing*, IEEE Trans. Inform. Theory, 52 (2006), pp. 1289–1306.

[24] D. L. Donoho, *For most large underdetermined systems of linear equations the minimal* ℓ_1*-norm solution is also the sparsest solution*, Comm. Pure Appl. Math., 59 (2006), pp. 797–829.

[25] D. L. Donoho and Y. Tsaig, *Fast Solution of* l1*-norm Minimization Problems When the Solution May Be Sparse*, http://www.stanford.edu/ tsaig/research.html.

[26] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, *Least angle regression*, Ann. Statist., 32 (2004), pp. 407–499.

[27] M. Figueiredo, R. Nowak, and S. J. Wright, *Gradient Projection for Sparse Reconstruction: Application to Compressed Sensing and Other Inverse Problems*, in IEEE J. Selected Topics in Signal Process., IEEE Press, Piscataway, NJ, 2007, pp. 586–597.

[28] M. P. Friedlander and M. A. Saunders, *Discussion: The Dantzig selector: Statistical estimation when p is much larger than n*, Ann. Statist., 35 (2007), pp. 2385–2391.

[29] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems*, SIAM, Philadelphia, 1998.

[30] P. C. Hansen and D. P. O'Leary, *The use of the L-curve in the regularization of discrete ill-posed problems*, SIAM J. Sci. Comput., 14 (1993), pp. 1487–1503.

[31] G. Hennenfent and F. J. Herrmann, *Sparseness-constrained data continuation with frames: Applications to missing traces and aliased signals in* 2/3*-D*, in Proceedings of the SEG

International Exposition and 75th Annual Meeting, Houston, TX, 2005.
[32] G. Hennenfent and F. J. Herrmann, *Simply denoise: Wavefield reconstruction via jittered undersampling*, Geophys., 73 (2008), pp. V19–V28.
[33] F. Herrmann and S. Ross-Ross, *SlimPy: A Python interface to Unix-like pipe based linear operators*, http://slim.eos.ubc.ca/SLIMpy.
[34] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, *An interior-point method for large scale L_1-regularized least squares*, IEEE J. Trans. Sel. Top. Signal Process., 1 (2007), pp. 606–617.
[35] S. Leyffer, *A note on multiobjective optimization and complementarity constraints*, Preprint ANL/MCS-P1290-0905, Mathematics and Computer Science Division, Argonne National Laboratory, Illinois, Argonne, IL, 2005.
[36] M. Lustig, D. L. Donoho, and J. M. Pauly, *Sparse MRI: The application of compressed sensing for rapid MR imaging*, Magn. Resonance Med., 58 (2007), pp. 1182–1195.
[37] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, *Compressed sensing MRI*, IEEE Signal Process. Mag., 25 (2007), pp. 72–82.
[38] MathWorks, *MATLAB User's Guide*, MathWorks, Natick, MA, 1992.
[39] *Mathematical Programming System*, http://www.mosek.com.
[40] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., Springer, New York, 2006.
[41] M. R. Osborne, B. Presnell, and B. A. Turlach, *A new approach to variable selection in least squares problems*, IMA J. Numer. Anal., 20 (2000), pp. 389–403.
[42] M. R. Osborne, B. Presnell, and B. A. Turlach, *On the LASSO and its dual*, J. Comput. Graph. Statist., 9 (2000), pp. 319–337.
[43] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.
[44] M. A. Saunders, *PDCO. Matlab Software for Convex Optimization*, http://www.stanford.edu/group/SOL/software/pdco.html.
[45] J. F. Sturm, *Using SeDuMi 1.02, A Matlab Toolbox for Optimization Over Symmetric Cones (Updated for Version 1.05)*, Technical report, Department of Econometrics, Tilburg University, Tilburg, The Netherlands, 1998.
[46] R. Tibshirani, *Regression shrinkage and selection via the Lasso*, J. Roy. Statist. Soc. Ser. B., 58 (1996), pp. 267–288.
[47] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*, V. H. Winston and Sons, Washington, DC, 1977 (in Russian).
[48] J. A. Tropp, *Just relax: Convex programming methods for identifying sparse signals in noise*, IEEE Trans. Inform. Theory, 52 (2006), pp. 1030–1051.