Department of Computer Science, University of British Columbia
DRAFT Technical Report TR-2010-nn, July 30, 2012

A DUAL ACTIVE-SET QUADRATIC PROGRAMMING METHOD FOR FINDING SPARSE LEAST-SQUARES SOLUTIONS

MICHAEL P. FRIEDLANDER* AND MICHAEL A. SAUNDERS†

Abstract. Many imaging and compressed sensing applications seek sparse solutions to large under-determined least-squares problems. The basis pursuit (BP) approach minimizes the 1-norm of the solution, and the BP denoising (BPDN) approach balances it against the least-squares fit. The duals of these problems are conventional linear and quadratic programs. We introduce a modified parameterization of the BPDN problem and explore the effectiveness of active-set methods for solving its dual. Our algorithm for solving a generalized form of the BP dual unifies several existing algorithms and is applicable to large-scale examples.

Key words. basis pursuit, basis pursuit denoising, active-set method, quadratic program, convex program, duality, regularization, sparse solution, sparse recovery, one-norm, elastic bounds

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

DOI. xxx/xxxxxxx

1. Introduction. Consider the linear system $Ax + r = b$, where A is an m-by-n matrix and b is a m-vector. In many statistical and signal processing applications the aim is to obtain a solution (x, r) such that the residual vector r is small in norm and the vector x is sparse. In model selection, for example, the aim is to find a parsimonious set of regressors that approximately fit the data (Osborne et al. [33], Efron et al. [16]). In these cases, typically $m > n$. In signal processing, the columns of A represent a large collection of atoms in a dictionary, and the aim is to select a small subset that admits an accurate reconstruction of the observed signal b (Chen et al. [7,8]). In contrast to traditional linear-regression applications, typically $m \ll n$ and the problem is ill-posed.

Regardless of the shape of A, many techniques for computing sparse solutions are based on using the one-norm function, or some variation of it, to regularize a least-squares (LS) problem. Consider the "primal" problem

$$\begin{array}{ll} \underset{x,\,y}{\text{minimize}} & \|x\|_1 + \frac{1}{2}\lambda\|y\|_2^2 \\ \text{subject to} & Ax + \lambda y = b \end{array} \tag{1.1}$$

and its dual

$$\begin{array}{ll} \underset{y}{\text{maximize}} & b^T y - \frac{1}{2}\lambda\|y\|_2^2 \\ \text{subject to} & -e \le A^T y \le e, \end{array} \tag{1.2}$$

where $\lambda \ge 0$ is a scalar parameter and e is a vector of ones. When $\lambda = 0$, (1.1) is equivalent to the basis pursuit (BP) problem [7,8]. It insists on a zero residual and often yields a sparse solution x. In some cases it yields the sparsest solution possible (Candès et al. [5], Donoho [12]). When $\lambda > 0$, (1.1) is equivalent to the basis pursuit

*Department of Computer Science, University of British Columbia, Vancouver V6T 1Z4, BC, Canada (mpf@cs.ubc.ca). This author's research was partially supported by NSERC Collaborative Research and Development Grant 334810-05.

†Systems Optimization Laboratory, Department of Management Science and Engineering, Stanford University, Stanford, CA 94305-4026, USA (saunders@stanford.edu). This author's research was partially supported by Office of Naval Research grant N00014-08-1-0191 and by AHPCRC.

denoising (BPDN) problem [7, 8]. It allows a nonzero residual, but the sparsity of x remains of prime importance. At the extreme, $x = 0$ and $r = b$ for all $\lambda \geq \|A^T b\|_\infty$.

Problems (1.1) and (1.2) are duals of each other in the sense that the Karush-Kuhn-Tucker (KKT) conditions for optimality for each problem are satisfied by the same vector pair (x, y). (The KKT conditions require the constraints in each problem to be satisfied and the objective values to be equal.) In particular, the optimal x values for (1.1) are optimal Lagrange multipliers for the inequality constraints of (1.2).

For all vectors with infinity-norm of one or less, the one-norm is the convex envelope of the "zero-norm" $\|x\|_0$, which counts number of nonzero elements of x. Problem (1.1) is thus often used as a convex relaxation of the nonconvex problem

$$\begin{array}{ll} \underset{x,\,y}{\text{minimize}} & \|x\|_0 + \frac{1}{2}\lambda\|y\|_2^2 \\ \text{subject to} & Ax + \lambda y = b. \end{array}$$

In some applications this is called the *sparse recovery problem*.

The term $\|x\|_1$ in the primal problem (1.1) is nonsmooth at points for which any component of x is zero. A standard device is to split x into positive and negative parts and solve the equivalent smooth problem

$$\begin{array}{ll} \underset{v,\,w,\,y}{\text{minimize}} & e^T(v + w) + \frac{1}{2}\lambda\|y\|_2^2 \\ \text{subject to} & A(v - w) + \lambda y = b, \quad v, w \geq 0, \end{array}$$

where $x = v - w$. Numerical methods for linear programming ($\lambda = 0$) and convex quadratic programming ($\lambda > 0$) may be applied.

1.1. Generalized formulation. Since the dual problem (1.2) is a conventional convex quadratic program, and since a sparse primal solution x implies relatively few active constraints in the dual, our interest has been to explore active-set methods for solving the dual. For practical reasons it has proved important to generalize the constraints $-e \leq A^T y \leq e$ to be $\ell \leq A^T y \leq u$ for given vectors satisfying $\ell \leq u$. For later reference we state the generalized primal and dual problems as

BP_λ:
$$\begin{array}{ll} \underset{x,\,y}{\text{minimize}} & c(x)^T x + \frac{1}{2}\lambda\|y\|_2^2 \\ \text{subject to} & Ax + \lambda y = b \end{array}$$

and

QP_λ:
$$\begin{array}{ll} \underset{y}{\text{minimize}} & \frac{1}{2}\lambda\|y\|_2^2 - b^T y \\ \text{subject to} & \ell \leq A^T y \leq u, \end{array}$$

where the jth component of the piecewise-linear function $c(x) : \mathbb{R}^n \to \mathbb{R}^n$ used in the primal objective is defined by

$$c_j(x) = \begin{cases} \ell_j & \text{if } x_j \leq 0, \\ u_j & \text{if } x_j > 0, \end{cases} \qquad \text{for} \quad j = 1, \ldots, n.$$

Certain choices for ℓ and u lead to the weighted one-norm regularization function ($u = -\ell = w$ for $w > 0$), or to a "one-sided" regularization function ($u = \infty$ or

$\ell = -\infty$). This formulation is general enough to capture a wide range of problems. For example, the generalized one-norm problem

$$\begin{array}{ll} \underset{x,y}{\text{minimize}} & \|Bx\|_1 + \frac{1}{2}\lambda\|y\|_2^2 \\ \text{subject to} & Ax + \lambda y = b, \end{array} \tag{1.3}$$

where B is a general matrix, is used for the "analysis" approach for sparse recovery [17], and the fused Lasso [39], among others. (This is equivalent to the generalized Lasso problem of Tibshirani and Taylor [40], who vary B to obtain the fused Lasso, trend filtering, wavelet smoothing, and outlier detection, and who emphasize the benefits of solving the dual of problem (1.3).) In order to fit this into the BP_λ formulation, we first consider the weighted least-squares problem

$$\underset{x,y}{\text{minimize}} \quad c(x)^T x + \frac{1}{2}\lambda\|Dy\|_2^2 \quad \text{subject to} \quad Ax + \lambda D^2 y = b, \tag{1.4}$$

where D is nonsingular, and note that it can be solved by BP_λ via the identifications

$$A \Leftrightarrow D^{-1}A, \quad b \Leftrightarrow D^{-1}b, \quad y \Leftrightarrow Dy.$$

Problem (1.4) then approximates (1.3) if we make the identifications

$$A \Leftrightarrow \begin{pmatrix} A & \\ B & I \end{pmatrix}, \quad b \Leftrightarrow \begin{pmatrix} b \\ 0 \end{pmatrix}, \quad D \Leftrightarrow \begin{pmatrix} I & \\ & \delta I \end{pmatrix}, \quad \ell \Leftrightarrow \begin{pmatrix} 0 \\ -e \end{pmatrix}, \quad u \Leftrightarrow \begin{pmatrix} 0 \\ e \end{pmatrix},$$

where δ is a small positive parameter. An acceptable numerical solution might be obtained if δ is not too small (say $\delta \geq 10^{-4}$ on a typical machine).

For such variations, the basic form of our proposed algorithm remains the same. The vital property of the active-set approach is that for sparse optimization problems, the number of iterations required is often directly proportional to the number of nonzeros in the primal solution vector x, independent of the size of the problem. The flexibility of our solver `BPdual` provides a base from which more involved algorithms can be easily implemented.

1.2. Overview. Our aim is present an active-set QP solver and demonstrate its potential to solve effectively a wide variety of sparse optimization problems. The main components of the paper are as follows.

Active-set quadratic programming (section 2). We describe an active-set method for solving problem QP_λ. It follows many standard procedures in the optimization tool kit. In particular, it maintains a sequence of iterates $\{y^k\}$ that are *feasible* for QP_λ (so that every y^k satisfies the constraints $\ell \leq A^T y^k \leq u$). This is the base algorithm for later sections.

Elastic constraints (section 3). In some applications we may have a good estimate of the optimal y^* that nevertheless violates the constraints. We adapt the method of Conn and Sinclair [10] (also described by Gould and Toint [24]), which allows the iterates to become infeasible, and provides a mechanism to ensure that the iterates eventually satisfy all the constraints. As we describe, the elastic-constraints variation allows for warm starts from infeasible points, and a mechanism for maintaining a well-conditioned subset of constraints in the active set. By judiciously truncating the solution process, it can also be used to solve BP_λ with the additional constraints $-w \leq x \leq w$ for some specified positive vector w.

Table 1.1: Solvers in the active set pursuit (ASP) MATLAB software library.

Solvers	Purpose	Section
`BPdual`	feasible/infeasible active-set method for QP_λ	§§2–3
`asp_bpdn`	basis pursuit denoising (including $\lambda = 0$)	§5
`asp_omp`	orthogonal matching pursuit	§6
`asp_topy`	homotopy version of basis pursuit denoising (for all λ)	§7
`asp_rwbp`	reweighted basis pursuit for approximating ℓ_0 solutions	§8
`asp_seqcs`	sequential compressed sensing (adding rows to A and b)	§9
`asp_nnls`	nonnegative least-squares	§10
`asp_l1l1`	sparse-residual and sparse-solution regression (ℓ_1-ℓ_1)	§11
`asp_gl1`	generalized Lasso for sparsity in Bx	§12

Implementation (section 4). We review efficient procedures to update a triangular matrix used for solving the LS subproblem associated with each iteration of the active-set method.

Applications (sections 5–12). We describe the use of `BPdual` on a range of problems that have been treated by other solvers; see Table 1.1.

1.3. Software implementation. The methods described above have been implemented as a MATLAB software package named ASP (*active-set pursuit*). The primary solver `BPdual` implements the feasible active-set method described in section 2, and also implements as an option the elastic active-set method described in section 3. This routine forms the kernel from which solvers for certain problems can be implemented using one or more calls to `BPdual`. Table 1.1 lists the routines implemented within the ASP software library. The routine `asp_omp`, which implements the orthogonal matching pursuit (OMP) algorithm, is an exception and does not make use of `BPdual`; it is included only for purposes of comparison.

In many applications that involve sparse optimization, the matrix A is not available explicitly. Without exception, `BPdual` and its dependent routines rely only on products with A and its transpose. These routines accept as input matrix-like objects that define methods for multiplication with vectors. This feature can be used to accommodate fast algorithms that might be available for products with the linear operator, such as when A is defined by a Fourier operator, for example.

1.4. Reproducible research. Following the discipline of reproducible research, the ASP software library, and the source code and data files required to reproduce all of the experimental results of this paper, including the figures and tables, can be downloaded from `http://www.cs.ubc.ca/~mpf/asp`.

1.5. Notation. Optimal solutions of BP_λ and QP_λ are denoted by x^* and y^*. The objective function of QP_λ and its gradient vector are defined as

$$\phi(y) = \tfrac{1}{2}\lambda\|y\|_2^2 - b^T y, \qquad g(y) = \lambda y - b.$$

If $\lambda > 0$, $\phi(y)$ is strictly convex and thus y^* is unique. The special vectors a_j and e_j denote the jth columns of A and the identity matrix. Otherwise, z_j denotes the jth component of a vector z. The support of a vector x is the set of indices j for which $x_j \neq 0$. We use the index set $\mathcal{S}$ to denote the support of x, meaning every component of the subvector $x_{\mathcal{S}}$ is nonzero.

Algorithm 1: Feasible active-set algorithm for QP_λ (`BPdual`).

input: A, b, ℓ, u, λ
$x \leftarrow [\,]$, $y \leftarrow 0$, $z \leftarrow 0$, $R \leftarrow [\,]$, $\mathcal{S} \leftarrow \{\,\}$ `[Initialize iterates]`
while true **do**
  $h \leftarrow b - \lambda y$ `[negative gradient of` ϕ`]`
  $x \leftarrow \arg\min \|h - Sx\|_2$ `[`x `is really` $x_{\mathcal{S}}$`; solve LS problem using` R`]`
  $\Delta y \leftarrow (h - Sx)/\lambda$
  $\Delta z \leftarrow A^T \Delta y$
  $(\alpha_{\max}, p) \leftarrow$ `linesearch`(Δz) `[step to closest constraint` p`; see (2.5)]`
  $\alpha \leftarrow \min\{1, \alpha_{\max}\}$
  $y \leftarrow y + \alpha \Delta y$
  $z \leftarrow z + \alpha \Delta z$
  if $\alpha < 1$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{p\}$ `[add constraint` p `to the active set]`
    $R \leftarrow$ `QRaddcol`(R, Ae_p) `[add` p`th column of` A `to QR factor; see §4.2]`
  else
    $\mathcal{C}_\ell \leftarrow \{r \mid x_r < 0 \quad \text{and} \quad z_j = \ell_j,\ j = \mathcal{S}_r\}$ `[lower deletion candidates]`
    $\mathcal{C}_u \leftarrow \{r \mid x_r > 0 \quad \text{and} \quad z_j = u_j,\ j = \mathcal{S}_r\}$ `[upper deletion candidates]`
    if $\mathcal{C}_\ell \cap \mathcal{C}_u = \emptyset$ **then**
      break `[optimal solution found; exit]`
    else
      $r \leftarrow \underset{r \in \mathcal{C}_1 \cap \mathcal{C}_2}{\arg\max} |x_r|, \quad q \leftarrow \mathcal{S}_r$
      $\mathcal{S} \leftarrow \mathcal{S} \setminus \{q\}$ `[delete constraint` q `from the working set]`
      $R \leftarrow$ `QRdelcol`(R, r) `[delete column` r `from QR factor; see §4.2]`
output: x, y, z, $\mathcal{S}$, R

2. An active-set method for QP_λ. Papers by Gill et al. [21], Fletcher [19], and others survey methods for general QPs. Many of those techniques simplify considerably when applied to convex problems and when the quadratic term in the objective $\phi(y)$ is simply $y^T y$. Algorithm 1 summarizes an active-set method for solving the specialized quadratic program QP_λ. For simplicity we assume that a feasible initial point y is known. (For example, $y = 0$ is feasible for the dual BPDN problem (1.2).) The algorithm building blocks are detailed in the following sections.

2.1. The active set. Let y be the current estimate of y^*. The jth constraint of QP_λ is *active* if $a_j^T y$ is at one of its bounds: $a_j^T y = \ell_j$ or u_j. We maintain an index set $\mathcal{S}$ of constraints that are active at y and whose columns of A form a submatrix S of full column rank. Thus for some permutation P,

$$AP = \begin{bmatrix} S & N \end{bmatrix} \quad \text{and} \quad S^T y = c,$$

where each component of c is an element of $\ell_{\mathcal{S}}$ or $u_{\mathcal{S}}$.

We stress that $\mathcal{S}$ may not contain all indices of constraints that are currently active. There may exist *degenerate* points y for which additional active constraints have normals that are linearly dependent on S. For this reason, active-set methods are often described as *working-set methods* [21, 24] to clarify the distinction between *all* active constraints and those in $\mathcal{S}$. Various practical strategies are available for systematically dealing with degeneracy, and are vital for any successful implementation [3, 20, 37]. In order to streamline later discussion, we make the simplifying assumption that there are no degenerate points.

2.2. Stationary points. The current y is *stationary* for QP_λ if it is feasible (i.e., $\ell \le A^T y \le u$) and if the objective gradient is a linear combination of the gradients of the constraints in the active set; that is, if there exists a vector $x_{\mathcal{S}}$ such that $Sx_{\mathcal{S}} = -g(y) = b - \lambda y$. Thus,

$$Sx_{\mathcal{S}} + \lambda y = b \quad \text{and} \quad S^T y = c. \tag{2.1}$$

These are the optimality conditions for the equality-constrained QP that minimizes $\phi(y)$ on the active set. The first condition in (2.1) implies that the pair (x, y) is feasible for BP_λ.

2.3. Search directions. Let $(x_{\mathcal{S}}, y)$ be current solution estimates, with y lying on the current active set (so that $S^T y = c$). A new point $(x_{\mathcal{S}} + \Delta x_{\mathcal{S}}, y + \Delta y)$ will be stationary if it is feasible and satisfies (2.1):

$$S(x_{\mathcal{S}} + \Delta x_{\mathcal{S}}) + \lambda(y + \Delta y) = b \quad \text{and} \quad S^T(y + \Delta y) = c. \tag{2.2}$$

This means that we can obtain search directions $\Delta x_{\mathcal{S}}$ and Δy by solving the system

$$\begin{bmatrix} \lambda I & S \\ S^T & 0 \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta x_{\mathcal{S}} \end{bmatrix} = \begin{bmatrix} h \\ 0 \end{bmatrix}, \qquad h \equiv b - \lambda y - Sx_{\mathcal{S}}, \tag{2.3}$$

or equivalently

$$\min_{\Delta x_{\mathcal{S}}} \|h - S\Delta x_{\mathcal{S}}\|, \quad \Delta y = (h - S\Delta x_{\mathcal{S}})/\lambda. \tag{2.4}$$

Note that h becomes small as $(x_{\mathcal{S}}, y)$ approaches a stationary point for the active set. Also, S having full column rank allows us to say that the search directions are zero if and only if $h = 0$.

Each iteration of the active-set algorithm needs the solution of (2.3) or (2.4), where S changes by only one column (added or deleted). This property opens the door to efficient implementations based on matrix-factorization updates. In particular, the sequence of LS problems (2.4) may be solved by maintaining the triangular part of a QR factorization of S. This is described in section 4.2.

2.4. The linesearch. As in more general active-set algorithms for optimization, we want to choose a steplength $\alpha > 0$ and take a step

$$x_{\mathcal{S}} \leftarrow x_{\mathcal{S}} + \alpha \Delta x_{\mathcal{S}} \quad \text{and} \quad y \leftarrow y + \alpha \Delta y$$

in order to reduce the QP_λ objective $\phi(y)$. From (2.2) and because $\phi(y)$ is strictly convex and quadratic, we know that the steplength $\alpha = 1$ would minimize $\phi(y)$ on the current active set. Thus we take $\alpha = 1$ if possible, then start a new iteration. If a step of $\alpha = 1$ violates constraints that are not in the active set, a linesearch determines which constraint would be violated first as α increases from zero. We add this constraint index to $\mathcal{S}$, add a column to S, then start a new iteration.

The linesearch makes use of the vectors $z = A^T y$ and $\Delta z = A^T \Delta y$. If $\Delta z_j < 0$ for some j, the step $y \leftarrow y + \alpha \Delta y$ moves z_j toward its lower bound ℓ_j, and if $\Delta z_j > 0$, the step moves z_j toward its upper bound u_j. (Because $z_j = 0$ for all $j \in \mathcal{S}$, searching along Δy can only encounter constraints that are not already in $\mathcal{S}$.) The maximum step α_j that can be taken without violating a constraint $j \notin \mathcal{S}$ is

$$\alpha_j = \begin{cases} (u_j - z_j)/\Delta z_j & \text{if } \Delta z_j > 0, \\ (\ell_j - z_j)/\Delta z_j & \text{if } \Delta z_j < 0, \\ +\infty & \text{otherwise.} \end{cases}$$

The steplength along Δy that reaches the nearest constraint, and the corresponding constraint index, are therefore given by

$$\alpha_{\max} = \min_{j \notin \mathcal{S}} \alpha_j \quad \text{and} \quad p = \arg\min_{j \notin \mathcal{S}} \alpha_j, \tag{2.5}$$

and the required steplength is $\alpha = \min\{1, \alpha_{\max}\}$.

2.5. Adding constraints. If $\alpha < 1$, one of the constraints $a_p^T y \geq \ell_p$ or $a_p^T y \leq u_p$ is encountered before $\phi(y)$ reaches a minimum on the current active set. That constraint is added to S. Otherwise, the full step Δy is taken.

A crucial implication of the definition of the search direction is that if a new constraint is encountered, it must be linearly independent of those in the active set. Recall that $S^T \Delta y = 0$ and that any constraint encountered during the linesearch must have a normal a_p that satisfies $a_p^T \Delta y \neq 0$. Hence, a_p must be linearly independent of S. We conclude that S always has full column rank.

This is true even in the degenerate case, when $\alpha = 0$. In all cases, it is important to break ties (when $\alpha_{\max} = \alpha_j$ for more than one j) in favor of large $|\Delta z_j|$.

2.6. Deleting constraints. If the linesearch returns $\alpha = 1$, it did not encounter any new constraint, and the new iterate $y \leftarrow y + \Delta y$ is a minimizer for $\phi(y)$ on the current active set. From (2.1) we know that $S x_{\mathcal{S}} = -g(y)$.

Further progress may be possible by moving away from one of the constraints in the active set. Suppose that an inequality constraint $q = \mathcal{S}_p$ is at its lower bound ($a_q^T y = \ell_q < u_q$) and consider a direction Δy satisfying $S^T \Delta y = e_p$, so that Δy is a feasible direction that moves away from the active lower bound ℓ_q. Note that

$$g^T \Delta y = (-S x_{\mathcal{S}})^T \Delta y = -x_{\mathcal{S}}^T S^T \Delta y = -(x_{\mathcal{S}})_p.$$

Thus, if $(x_{\mathcal{S}})_p > 0$, the objective will improve if we move in the direction Δy. We allow this by removing constraint q from the active set.

Similarly, if constraint q is at its upper bound and $(x_{\mathcal{S}})_p < 0$, the objective will improve if we move along a direction for which $S^T \Delta y = -e_p$. Again we allow this by removing constraint q from the active set.

Thus, the Lagrange multipliers $x_{\mathcal{S}}$ reveal which active constraint indices should be removed from $\mathcal{S}$ to allow reduction of the objective function. If no elements of $x_{\mathcal{S}}$ have the correct sign, the current point (x, y) is optimal.

3. Elastic bounds. The basic active-set method describe in section 2 requires a feasible starting point and maintains feasibility thereafter. For problems that originate from (possibly weighted) one-norm regularization, $\ell < 0 < u$, and so the trivial starting point $y_0 = 0$ is always available. However, more general bounds, and the ability to warm-start the algorithm from arbitrary points, require either a preliminary solution phase to obtain a feasible vector (such as the "phase 1" procedure common in linear programming [26]), or a method that allows infeasible intermediate iterates. Sparsity is paramount for the applications that we consider, and it may be useful to obtain sparse solutions even at the expense of suboptimality of BP_λ, which is the implication of obtaining infeasible solutions of QP_λ.

3.1. An exact penalty function. To this end, we consider a relaxed version of QP_λ that discards the explicit constraints and replaces them with penalties on constraint violations. The elastic-bounds problem is defined to be

$\text{QP}_{\lambda w}$:	$\underset{y}{\text{minimize}} \quad \phi_w(y) := \phi(y) + \sum_{j=1}^n w_j \sigma_j(a_j^T y)$,

where $\phi(y) = \frac{1}{2}\lambda\|y\|_2^2 - b^T y$ as before, w is a vector of positive penalty parameters (each $w_j > 0$), and each function σ_j measures violation in the jth constraint of QP_λ:

$$\sigma_j(\zeta) = \begin{cases} \ell_j - \zeta & \text{if } \zeta < \ell_j, \\ \zeta - u_j & \text{if } \zeta > u_j, \\ 0 & \text{otherwise.} \end{cases}$$

A key feature of this ℓ_1-penalty function is that it is *exact*: sufficiently large values of w_j ensure that minimizers of ϕ_w coincide with the solution of the explicitly constrained problem QP_λ, assuming that it is feasible; see, e.g., [18, §14.3].

The dual of the elastic problem $\mathrm{QP}_{\lambda w}$ is illuminating:

$$\begin{array}{|ll|} \hline \mathrm{BP}_{\lambda w}\text{:} & \underset{x}{\text{minimize}} \quad c(x)^T x + \frac{1}{2}\lambda\|y\|_2^2 \\ & \text{subject to} \quad Ax + \lambda y = b, \quad -w \le x \le w. \\ \hline \end{array}$$

This is problem BP_λ with additional constraints on x. Clearly there are finite values of the components of w—namely, $w_j > |x_j^*|$—that permit recovery of the original problem solution. Thus, one role of the elastic problem is to *bound the primal solution*.

The following alternative algorithm for QP_λ suggests itself: solve a sequence of problems $\mathrm{QP}_{\lambda w}$ with increasing values of the components of w. If QP_λ is feasible, then the solution of $\mathrm{QP}_{\lambda w}$ will be feasible for QP_λ after finitely many increases to w, and will coincide with the solution of QP_λ. If, on the other hand, the components of w are increased beyond some prescribed maximum value and $\mathrm{QP}_{\lambda w}$ still does not yield a solution feasible for QP_λ, then QP_λ is declared infeasible. We note that it is not necessary to solve each elastic subproblem to optimality in order for the overall algorithm to converge.

3.2. An elastic-bounds algorithm. For problem $\mathrm{QP}_{\lambda w}$, we consider the elastic-bounds approach first proposed by Conn and Sinclair [10], which is the basis for the active-set large-scale nonconvex QP solver QPA in the GALAHAD optimization library [23, 24]. For the structured problems that we consider, the elastic-bounds algorithm can be implemented as a straightforward modification of Algorithm 1.

The active set serves the same role as before, except that search directions are defined by the solution of the equality-constrained QP

$$\underset{\Delta y}{\text{minimize}} \quad \tfrac{1}{2}\lambda\|\Delta y\|_2^2 + g_\rho^T \Delta y \quad \text{subject to} \quad S^T \Delta y = 0,$$

where

$$g_w = g + A\bar{w}, \quad \text{with} \quad \bar{w}_j = \begin{cases} -w_j & \text{if } a_j^T y < \ell_j \\ +w_j & \text{if } a_j^T y > u_j \\ 0 & \text{otherwise,} \end{cases} \tag{3.1}$$

is a subgradient of the composite function ϕ_w. The definition of $\bar{w}$ assumes nondegeneracy: in that case, constraints that are active—but not in $\mathcal{S}$—would contribute a nonzero element to $\bar{w}$ in anticipation that the subsequent search direction may cause these constraints to become infeasible. In contrast, constraints $j \in \mathcal{S}$ always satisfy $a_j^T \Delta y = 0$, and hence do not contribute to $\bar{w}$ because these constraints cannot become infeasible. Algorithm 2 summarizes the elastic active-set algorithm.

The step along the search direction Δy (step 5 of Algorithm 2) is computed by optimizing the one-variable piecewise-quadratic function $\psi(\alpha) := \phi_w(y + \alpha\Delta y)$. This

Algorithm 2: Elastic algorithm for $\mathrm{QP}_{\lambda w}$ (`BPdual_elastic`).

input: A, b, ℓ, u, λ, w
$x \leftarrow [\,]$, $y \leftarrow 0$, $z \leftarrow 0$, $R \leftarrow [\,]$, $\mathcal{S} \leftarrow \{\,\}$ `[initialize iterates]`
while `true` **do**
1 $h \leftarrow b - \lambda y - A\bar{w}$ `[negative subgradient of` ϕ_w`; see (3.1)]`
2 $x \leftarrow \arg\min_x \|h - Sx\|_2$ `[solve least-squares problem using` R`]`
3 $\Delta y \leftarrow (h - Sx)/\lambda$
4 $\Delta z \leftarrow A^T \Delta y$
5 $(\alpha, p) \leftarrow$ `linesearch`(Δz) `[exact linesearch on` $\phi_w(y + \alpha \Delta y)$`]`
$y \leftarrow y + \alpha \Delta y$
$z \leftarrow z + \alpha \Delta z$
if $\alpha < 1$ **then**
$\mathcal{S} \leftarrow \mathcal{S} \cup \{p\}$ `[add constraint` p `to the active set]`
6 $R \leftarrow$ `QRaddcol`(R, Ae_p) `[add` p`th column of` A `to QR factor; see §4.2]`
else
$\mathcal{C}_\ell \leftarrow \{r \mid x_r < 0 \text{ or } x_r > +\rho; \quad \text{and} \quad z_j = \ell_j,\ j = \mathcal{S}_r\}$
$\mathcal{C}_u \leftarrow \{r \mid x_r > 0 \text{ or } x_r < -\rho; \quad \text{and} \quad z_j = u_j,\ j = \mathcal{S}_r\}$
if $\mathcal{C}_\ell \cap \mathcal{C}_u = \emptyset$ **then**
7 break `[optimal solution found; exit]`
else
$r \leftarrow \arg\max_{r \in \mathcal{C}_1 \cap \mathcal{C}_2} |x_r|, \quad q \leftarrow \mathcal{S}_r$
$\mathcal{S} \leftarrow \mathcal{S} \setminus \{q\}$ `[delete constraint` q `from the working set]`
$R \leftarrow$ `QRdelcol`(R, r) `[delete column` r `from QR factor; see §4.2]`
output: x, y, z, $\mathcal{S}$, R

can be accomplished by computing and sorting the *breakpoints* α_j that define the beginning of each quadratic piece, and sequentially optimizing over each piece until the optimum α^* is found. If the optimal step occurs in the first piece (i.e., $\alpha^* = 1 < \alpha_1$, where α_1 is the first breakpoint), then $y + \Delta y$ optimizes ϕ_w over the current subspace and is a stationary point. If the optimal step occurs in the strict interior of a piece (i.e., $\alpha_j < \alpha^* < \alpha_{j+1}$), then $y + \alpha^* \Delta y$ is not necessarily a stationary point, and the active set does not change. The final possibility is that the optimal step occurs at a breakpoint (i.e., $\alpha^* = \alpha_{j+1}$); in that case, the constraint corresponding to this index is added to the active set.

For a fixed value of the penalty vector w, Algorithm 2 will converge to a minimizer of ϕ_w that may not be feasible for the original problem. If the resulting solution is infeasible for QP_λ, then w is increased (e.g., $w \leftarrow 2w$) and the elastic problem is resolved. Because the penalty function is exact, w will increase only finitely many times if QP_λ is feasible. If $\|w\|_\infty$ reaches a specified maximum value, QP_λ is declared infeasible.

4. Implementation. We assume that A is available as an operator. Each iteration requires one multiplication with A^T and one with A; see steps 4 and 6 of Algorithm 2. If the current y is infeasible, the elastic active-set algorithm requires one additional product in step 1.

We also maintain a "Q-less" QR factorization of the s-by-m submatrix S by updating a dense triangular s-by-s matrix R, whose dimension increases or decreases by 1 each iteration. Under the assumption—valid for many sparse-recovery applications—that s is small, the $O(s^2)$ storage for R is relatively small. On the other hand, s may

Table 4.1: *Key to symbols used in tables.*

m, n	number of rows and columns in A
$\|r\|_2$	two-norm of the final residual $r = b - Ax$
$\|\mathcal{S}\|$	number of indices in the final active set
nnz(x)	number of nonzeros in the computed solution; see (4.1)
itns	number of iterations
nMat	total number of matrix-vector products with A and A^T
cond(S)	condition number of the active-set matrix S

Table 4.2: *The Sparco test problems used.*

Problem	ID	m	n	$\|b\|_2$	operator
`blocksig`	2	1024	1024	7.9e+1	wavelet
`cosspike`	3	1024	2048	1.0e+2	DCT
`dcthdr`	12	2000	8192	2.3e+3	restricted DCT
`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
`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
`yinyang`	603	1024	4096	2.5e+1	wavelet

be large for truly large problems (even if $s \ll n$) and the storage requirements may become prohibitive.

For simplicity we store the submatrix S from the current active set. Storage for this matrix could be foregone at the expense of two additional products with A, which would take place in step 3 and inside `QRaddcol` in step 6 (see section 4.2).

4.1. Numerical experiments. The numerical experiments described in the following sections include (among others) a selection of nine relevant problems from the Sparco [42] collection of benchmark sparse-recovery problems of the "primal" form (1.1). Each problem includes a linear operator A and a right-hand side vector b. We selected problems where the BP solution x is known to have fewer than 1000 nonzero entries.

Table 4.1 defines the symbols used in the tables of results, with nnz(x) counting the significant nonzero entries in the solution vector x. This is computed as the number of nonzero entries that carry 99.9% of the one-norm of the vector:

$$\text{nnz}(x) = \{\min r \quad \text{such that} \quad \textstyle\sum_i^r |x_{\lfloor i \rfloor}| \geq 0.999\|x\|_1\}, \tag{4.1}$$

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

Table 4.2 lists the selected problems, includes the problem name, the Sparco ID, and a brief description of the A, which is often a compound operator.

4.2. Least-squares updates. Following (2.4), each search direction is computed by solving an LS problem $\min_x \|h - Sx\|_2$ and setting $\Delta y = (h - Sx)/\lambda$. Because we choose not to store A, we are restricted to obtaining the solution of the LS problem via

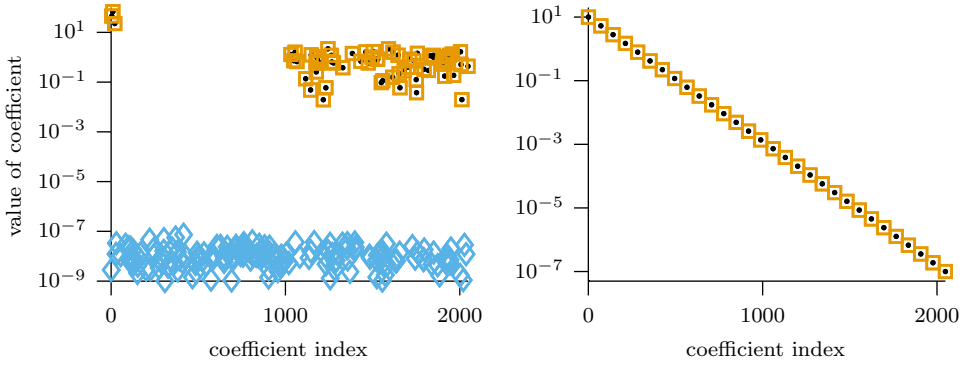


Fig. 4.1: The result of trimming the active set on two test problems, showing true (•), recovered (□), and trimmed (◊) coefficients. The left panel (Sparco problem `gcosspike`) shows that the trimming procedure eliminated all of the extraneous coefficients. The right panel (a problem whose true coefficients span 8 orders of magnitude) shows that none of the small (and correct) coefficients were trimmed.

the semi-normal equation $R^T Rx = S^T h$. As described by Björck [2, §6.6.5], one step of iterative refinement—the corrected semi-normal equation method—may be needed to improve accuracy. (In the case of compressed sensing, however, where A is expected to satisfy a restricted isometry property [4] or to have a low mutual coherence [13, 14], S is expected to be extremely well conditioned, and the correction step is not needed.)

Utilities `QRaddcol` and `QRdelcol` implement standard procedures [2, §3.2] for updating R when columns of S are added or deleted without storing or updating the orthogonal part of the QR factorization.

4.3. Trimming the active set. The optimal active set $\mathcal{S}$ returned by Algorithm 1 (`BPdual`) is often as important as the final solution. For some applications, the main aim is to discover a *minimal* set of columns of A that need to participate in the reproduction of the right-hand side. Indeed, an optimal $\mathcal{S}$ is sufficient—via a conventional linear least-squares solution—to recover an optimal solution x.

The left panel in Figure 4.1 shows that `BPdual` may yield solutions x with many small and insignificant entries, i.e., elements in $\mathcal{S}$ that are "weakly" active. The naive approach of simply trimming elements from $\mathcal{S}$ that correspond to small coefficients x suffers from the ambiguity in having to define a suitable threshold that doesn't also eliminate small elements that should be in the active set. To illustrate this point, the right panel shows a contrived example whose true coefficients smoothly span eight orders of magnitude, and in that case there is no well defined threshold.

Algorithm 3 describes a simple trimming procedure that systematically deletes elements from $\mathcal{S}$ that correspond to small elements in x. The algorithm assumes that the inputs are optimal—i.e., they are the results of `BPdual`—and exits when another constraint deletion would result in loss of optimality (the multipliers x no longer have the correct sign) or feasibility. A concession to efficiency appears in line 1: a rigorous test would compute $\Delta z = A^T \Delta y$ (see line 5 of Algorithm 1) at each iteration in order to ensure feasibility, but we wish to keep the iterations lightweight and avoid products with A (and thus implicitly assume that $\|A\| \approx 1$).

Figure 4.1 shows the results of applying Algorithm 3 to two different problems. In the left panel, 117 of 237 elements from $\mathcal{S}$, corresponding to small elements in x, are

Algorithm 3: Trimming procedure for the working set.

input: $\mathcal{S}$, R, x, λ
$h \leftarrow b - \lambda y$ [h is the negative gradient of ϕ]
$r \leftarrow \arg\min_r |x_r|$ [r is the most weakly active constraint]
while $|x_r| <$ `tol` **do**
  $q \leftarrow \mathcal{S}_r, \quad \mathcal{S} \leftarrow \mathcal{S} \setminus \{q\}$ [delete constraint r from the working set]
  $R \leftarrow$ `QRdelcol`(R, r) [delete column r from QR factor; see §4.2]
  $\bar{x} \leftarrow \arg\min_x \|h - Sx\|_2$ [solve least-squares problem using R]
  $\Delta y \leftarrow (h - S\bar{x})/\lambda$
1 **if** $\operatorname{sgn}(\bar{x}) \neq \operatorname{sgn}(x_{-r})$ or $\|\Delta y\| >$ `tol` **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{q\}$ [add constraint q back into the active set]
    $R \leftarrow$ `QRaddcol`(R, Ae_q) [add qth column of A to QR factor; see §4.2]
    break [no more elements to trim; exit]
  $x \leftarrow \bar{x}, \quad r \leftarrow \arg\min_r |x_r|$
output: x, $\mathcal{S}$

eliminated; in the right panel, no constraints are trimmed. Table 5.1 gives the number of elements trimmed for each of the Sparco test problems.

5. Basis pursuit (`asp_bpdn`). The BP problem aims to find a sparse solution to the system $Ax = b$, which corresponds to solving BP_λ with $\lambda = 0$. In this case the dual problem QP_0 is a linear program. However, the active-set algorithm is well defined only for positive values of λ. In particular, step 4 of Algorithm 1 defines the search direction Δy as the residual of an LS problem (computed in the previous step) scaled by λ.

Our approach for applying Algorithm 1 to the BP problem is based on the fact that it is not necessary to set $\lambda = 0$ in order to obtain a solution of QP_0. As shown by Mangasarian and Meyer [29], there exists a positive parameter $\bar{\lambda}$ such that for all $\lambda \in (0, \bar{\lambda})$ the solution y of QP_λ coincides with the unique least-norm solution of QP_0. Note that for problems with sparse solutions, we expect the solution of BP_0 (the dual of QP_0) to have many fewer than m nonzero elements; this situation translates into nonuniqueness of y. In the parlance of linear programming, BP_0 is expected to be highly *primal degenerate*, which means that its set of dual solutions is expected to have high dimensionality [22]. In this light, solving QP_λ with a small but positive value of λ corresponds to a solving a regularized version of QP_0 that ensures that the solution y is unique (and of minimimum length). The resulting approach is similar to Plumbley's polytope faces pursuit algorithm for BP [36], which seems to have been developed without reference to existing active-set methods.

In Table 5.1, we summarize experiments in which we apply Algorithm 1 with $\ell = -e$, $u = e$, and $\lambda = \sqrt{\epsilon}$ (where ϵ is the relative rounding error in double-precision floating-point arithmetic) to obtain BP solutions to the problems in Table 4.2. For most problems, the number of iterations is proportional to $\text{nnz}(x)$. In particular, for all problems except `p3poly`, `soccer1`, and `yinyang`, Algorithm 1 behaves in a perfectly greedy manner, meaning that elements are only added to the active set. However, note that for some of these problems—notably `cosspike`, `dcthdr`, `gcosspike`, and `spiketrn`—some of the entries in the final support are relatively insignificant (as suggested by an iteration count larger than $|\mathcal{S}|$; cf. (4.1)). Thus, although the solution support grows greedily, the final value of some of the coefficients may be negligible, indicating that the greedy approach is not necessarily optimal. Donoho and Tsaig [15]

Table 5.1: *Performance of the active-set method on basis pursuit problems.*

	asp_bpdn		asp_topy		both solutions			
Problem	itns	trim	itns	trim	nnz(x)	$\lvert S\rvert$	$\lVert r\rVert_2$	cond(S)
blocksig	1024	0	1025	0	1017	1024	5e−07	1e+00
cosspike	125	3	126	3	116	122	8e−15	1e+00
dcthdr	357	57	364	49	282	300	1e−12	2e+00
gcosspike	237	174	259	117	59	63	8e−14	2e+00
jitter	3	0	4	0	2	3	6e−08	1e+00
p3poly	2414	0	1027	0	581	600	1e−08	5e+02
sgnspike	20	0	21	0	19	20	1e−07	1e+00
soccer1	14475	0	4935	0	2992	3199	2e−06	3e+03
spiketrn	100	88	138	57	11	12	2e−14	4e+00
yinyang	4370	0	1763	0	994	1024	7e−07	2e+03

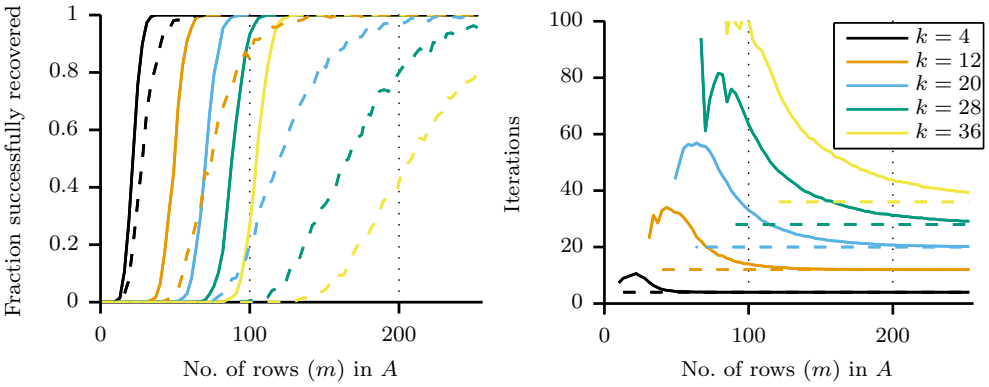
Algorithm 4: Orthogonal matching pursuit (`as_omp`).

input: A, b
$r \leftarrow b$, $R \leftarrow [\,]$, $\mathcal{S} \leftarrow \{\}$ `[initialize iterates]`
while $|\mathcal{S}| < m$ or $\|r\| > 0$ **do**
  $z \leftarrow A^T r$
  $p \leftarrow \arg\max |z_p|$, $j \leftarrow \mathcal{S}_p$ `[choose entering variable` j`]`
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{j\}$ `[add` j`th variable to the working set]`
  $R \leftarrow$ `QRaddcol`(R, Ae_j) `[add` j`th column of` A `to QR factor; see §4.2]`
  $x \leftarrow \arg\min_x \|b - Sx\|_2$ `[solve least-squares problem using` R`]`
  $r \leftarrow b - Sx$ `[update the residual vector]`
output: x, $\mathcal{S}$

noted the greedy behavior, which they call the "k-step" property, of the homotopy method [33] on certain input data.

6. Orthogonal matching pursuit (`asp_omp`). The OMP algorithm introduced by Pati, Rezaifar, and Krishnaprasad [35], and popularized in the book by Mallat [28, §9.5], is used extensively in the signal-processing literature as a means for obtaining sparse solutions to underdetermined linear systems. For important cases where A is either Gaussian or Bernoulli—i.e., every element is drawn from a Gaussian or Bernoulli distribution respectively—Tropp and Gilbert [41] give conditions that are sufficient for OMP to recover the sparsest solution. While it is now well known that the BP problem can recover the sparsest solution under more general conditions than can OMP, the OMP approach is considered to be computationally cheaper [30]. Indeed, there are several proposals for closing the gap between OMP and BP while preserving the computational advantage of OMP; some examples include the CoSamP [31] and ROMP [32] algorithms.

Each iteration of OMP is based on adding one index to the active set (the variable that maximizes the gradient of the LS objective $\|Ax-b\|_2^2$) and solving the LS problem over the variables in the active set. Algorithm 4 outlines the OMP algorithm. The main work per iteration is the same as for the basic active-set algorithm. Both methods require a product with A and A^T and updating of R in the QR factors of S. The main difference is that OMP is greedy (it only adds elements to the active set).



(a) Fraction of successfuly recovered solutions

(b) Average number of iterations

Fig. 6.1: *Comparing basis pursuit (solid line) against orthogonal matching pursuit (dotted line) on random problems with different solution sparsity levels k. The left panel plots the fraction of solutions that are correctly recovered; the right panel gives the average number of iterations needed on successful solves.*

Figure 6.1 reproduces an experiment conducted by Tropp and Gilbert [41] to assess the performance of OMP in recovering the sparsest solution to an underdetermined linear system $Ax = b$. In these experiments, A is an m-by-256 matrix drawn from a Gaussian distribution, and a random solution x_0 with k nonzeros is used to generate b. OMP is applied to 1,000 instances of a problem for a given pair (m, k), where $m \in \{1, \ldots, 256\}$ and $k \in \{4, 12, 20, 28, 36\}$. The left panel gives the fraction of problems where OMP succeeded in obtaining the correct support of the solution; the right panel gives the average number of iterations, over successful solves, to obtain that solution. Overlayed (solid lines) are the results of applying the active-set algorithm to the corresponding BP problems.

The left panel confirms the well known property that the BP formulation is significantly more successful at recovering sparsest solutions. The right panel shows that, over regimes where OMP and BP both have a high probability of success (nearly 1), the BP active-set algorithm requires the same number of iterations (and therefore the same computational effort) as OMP; the active-set algorithm incurs more iterations only in regimes where OMP has a lower probability of succeeding.

7. Homotopy (`asp_topy`). A number of algorithms based on repeatedly solving a quadratic program have been proposed for obtaining sparse LS solutions. A notable example is Homotopy [33, 34], which is based on solving a sequence of Lasso [38] problems

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|Ax - b\|_2^2 \quad \text{subject to} \quad \|x\|_1 \leq \tau_k \tag{7.1}$$

with increasing values of τ_k. The algorithm begins with $\tau_0 = 0$ (with solution $x_0 = 0$) and increases τ_k in stages that predictably change the number of nonzeros in the solution x_k. Typically, the support of x_k increases by exactly one index as τ_k passes through each "breakpoint". However, the support can also contract—and in the presence of degeneracy, multiple entries may be added or deleted at each breakpoint. The norm of the optimal residuals $r_k := b - Ax_k$ decreases monotonically, and the

values of τ_k increase until the desired residual norm is reached. A crucial consequence of the Homotopy approach is that each intermediate solution x_k is optimal (7.1) for a given value of τ_k. Efficiency follows from the ability to update the solution of each subproblem systematically as τ_k passes through each breakpoint.

Efron et al. [16] interpret Homotopy from a statistical point of view. Their resulting LARS algorithm is precisely the Homotopy algorithm, except that elements are never dropped from the support, and instead elements are allowed to pass through zero and switch sign; the resulting sequence of iterates may no longer be interpreted as optimal for some optimization problem.

The original motivation for Homotopy was to obtain sparse LS solutions. Donoho and Tsaig [15] explore the effectiveness of the Homotopy approach for basis pursuit. In that context, the sequence of Lasso problems is solved with increasing values of τ_k and stops at the first breakpoint τ_k such that $Ax_k = b$; this is a basis-pursuit solution.

7.1. Parametric quadratic programming. The Homotopy algorithm is a particular case of parametric programming (see, e.g., [1]) applied to QP_λ. Suppose y is the optimal solution of QP_λ for a given λ, and let $x_\mathcal{S}$ be the vector of Lagrange multipliers for the optimal active set $\mathcal{S}$. The goal is to find the largest allowable reduction in λ for which $\mathcal{S}$ continues to be the optimal active set. If λ is reduced to $\bar{\lambda} = \lambda - \alpha$, the corresponding optimal pair $(\bar{x}_\mathcal{S}, \bar{y})$ satisfies

$$S\bar{x}_\mathcal{S} + \bar{\lambda}\bar{y} = b \quad \text{and} \quad S^T\bar{y} = c \tag{7.2}$$

(among other conditions), with $\bar{x}_\mathcal{S}$ having the correct sign pattern. We describe below how to compute the maximum allowable reduction in λ; the corresponding optimal solution can obtained as a side-product of this computation.

Subtract (2.1) from (7.2), and use the definition of $\bar{\lambda}$ to obtain

$$S(\bar{x}_\mathcal{S} - x_\mathcal{S}) + \lambda(\bar{y} - y) - \alpha\bar{y} = 0 \quad \text{and} \quad S^T(y - \bar{y}) = 0.$$

Premultiply the first equation by S^T and use the second equation to arrive at

$$S^TS(\bar{x}_\mathcal{S} - x_\mathcal{S}) = \alpha S^T\bar{y} = \alpha S^Ty.$$

Note that this equation can be interpreted as the optimality conditions for an LS problem. In particular,

$$\bar{x}_\mathcal{S} = x_\mathcal{S} + \alpha\Delta x \quad \text{where} \quad \Delta x \text{ solves minimize } \|y - S\Delta x\|_2. \tag{7.3}$$

Thus, the direction in which $x_\mathcal{S}$ moves as α increases is fixed. The maximum step α_j^x that can be taken without changing the current sign of x_j is then

$$\alpha_j^x = \begin{cases} -x_j/\Delta x_j & \text{if } j \in \mathcal{C}_u \cup \mathcal{C}_\ell \\ +\infty & \text{otherwise,} \end{cases} \tag{7.4}$$

where $\mathcal{C}_u = \{j \mid x_j < 0,\ \Delta x_j > 0\}$ and $\mathcal{C}_\ell = \{j \mid x_j > 0,\ \Delta x_j < 0\}$ are index sets of variables that are respectively positive and decreasing, or negative and increasing. Thus α cannot be larger than $\alpha^x = \min_j \alpha_j^x$.

With λ decreasing to $\bar{\lambda}$, we need to compute the effect on y, and hence on the constraints $\ell \le A^Ty \le u$. Subtract $\bar{\lambda}y$ from both sides of (7.2) and use the definition of Δx to obtain

$$(\lambda - \alpha)(\bar{y} - y) = b - \bar{\lambda}y - S\bar{x}_\mathcal{S} = (b - \lambda y - Sx) + \alpha(y - S\Delta x) = \alpha(y - S\Delta x).$$

Algorithm 5: The Homotopy method (`as_topy`).

input: A, b, ℓ, u, $\lambda_{\min}$
$\lambda \leftarrow \|A^T b\|_\infty$ `[initial λ has empty active set]`
while $\lambda > \lambda_{\min}$ **do**
  $[x, y, z, \mathcal{S}, R] \leftarrow$ `as_bpdn`(A, b, ℓ, u, λ) `[solve QP`$_\lambda$ `with current λ]`
  $\Delta x \leftarrow \arg\min_{\Delta x} \|y - S\Delta x\|_2$ `[solve (7.3) using R]`
  $\Delta y \leftarrow y - S\Delta x$
  $\Delta z \leftarrow A^T \Delta y$
  $\alpha^x \leftarrow$ `maxstepx`(Δx) `[max step to sign change; see (7.4)]`
  $\alpha^z \leftarrow$ `maxstepz`(Δz) `[max step to constraint violation; see (7.5)]`
  $\alpha \leftarrow \min\{\alpha^x, \alpha^z\}$ `[max overall step]`
  $\lambda \leftarrow \lambda - \alpha$
output: x, y, z, $\mathcal{S}$

The last equality follows from the optimality of the pair (x, y). Thus,

$$\bar{y} = y + \frac{\alpha}{\lambda - \alpha} \Delta y, \quad \text{where} \quad \Delta y := y - S\Delta x$$

is the residual of the LS problem (7.3). The change in constraints is then given by

$$\bar{z} = z + \frac{\alpha}{\lambda - \alpha} \Delta z, \qquad \text{where} \qquad z := A^T y \quad \text{and} \quad \Delta z := A^T \Delta y.$$

The maximum step α_j^z that can be taken without violating a constraint $j \notin \mathcal{S}$ is

$$\alpha_j^z = \begin{cases} \lambda(u_j - z_j)/(\Delta z_j - z_j + u_j) & \text{if } \Delta z_j > 0 \\ \lambda(\ell_j - z_j)/(\Delta z_j - z_j + \ell_j) & \text{if } \Delta z_j < 0 \\ +\infty & \text{otherwise.} \end{cases} \tag{7.5}$$

Thus α cannot be larger than $\alpha^z = \min_{j \notin \mathcal{S}} \alpha_j^z$. The overall allowed reduction in λ is then $\alpha = \min\{\alpha^x, \alpha^z\}$.

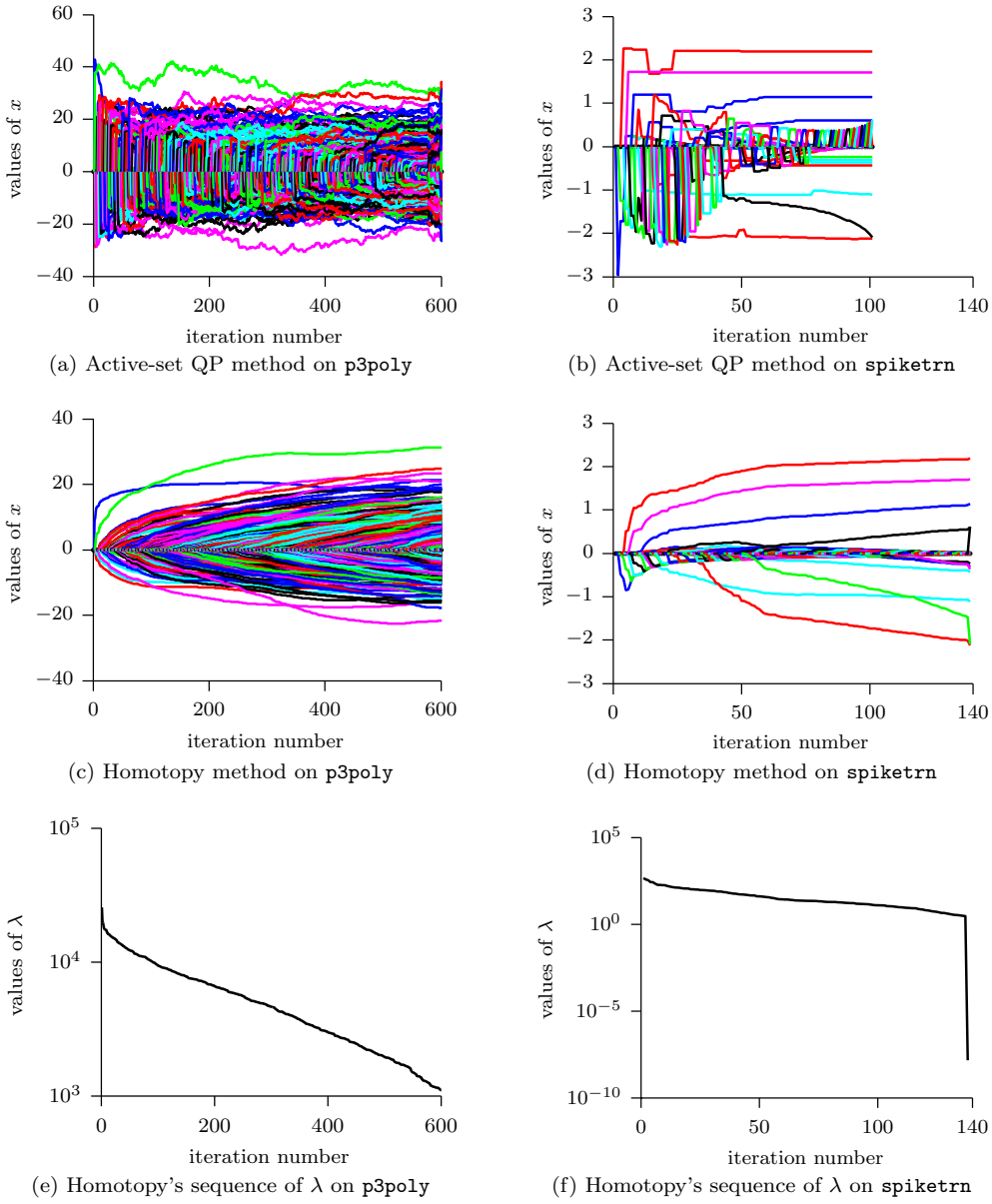
Algorithm 5 outlines the Homotopy method, which is implemented in the routine `asp_topy`.

7.2. Homotopy path. Figure 7.1 compares the Homotopy method `asp_topy` against the quadratic programming method `asp_bpdn` on the Sparco test problems `p3poly` and `spiketrn`. Comparing the top and middle panels, it is clear that the Homotopy solution path is considerably smoother and more predictable than the active-set method. This is typical behavior. The number of iterations for convergence between the two methods, however, is not always necessarily the same: on `spiketrn`, the active-set method required fewer iterations to converge. Further comparisons between the active-set and Homotopy methods are given in Table 5.1.

8. Reweighted basis pursuit (`asp_rwbp`). The reweighted BP algorithm proposed by Candés, Wakin, and Boyd [6] aims to improve on the ability of the BP approach to recover sparse solutions. Candés et al. give empirical evidence that the algorithm can sometimes recover sparsest solutions (i.e., the so-called zero-norm solution) in cases where BP fails to do so.

The canonical reweighted BP algorithm solves a sequence of scaled BP problems

$$\underset{x}{\text{minimize}} \quad \|W_k x\|_1 \quad \text{subject to} \quad Ax = b, \tag{8.1}$$



(a) Active-set QP method on `p3poly`
(b) Active-set QP method on `spiketrn`
(c) Homotopy method on `p3poly`
(d) Homotopy method on `spiketrn`
(e) Homotopy's sequence of λ on `p3poly`
(f) Homotopy's sequence of λ on `spiketrn`

Fig. 7.1: *Solution paths of the basic active-set method (top panels) and the homotopy method (middle and bottom panels) on the Sparco problems* `p3poly` *(left panels) and* `spiketrn` *(right panels).*

where W_k is a diagonal weighting matrix defined by x_{k-1}, the solution of the previous weighted problem, i.e.,

$$W_k = \text{diag}(w_k), \qquad w_k = 1./(\epsilon_k + |x_{k-1}|);$$

the positive scalar ϵ_k is held constant or adaptively reduced toward zero. (Davis and

Algorithm 6: Reweighted basis pursuit denoise (`as_rwbp`)

input: A, b, $k_{\max}$, ϵ
$y \leftarrow 0$, $\lambda \leftarrow \|A^T b\|_\infty/2$, $w \leftarrow e$ `[initialize iterates]`
`qp_solver` $\leftarrow$ `as_bpdn` or `as_topy` `[choose Algorithm 2 or 5 for subproblem]`
while $k < k_{\max}$ **do**
  $(\mathcal{S}, x, y) \leftarrow$ `qp_solver`$(A, b, w, -w, \lambda, \mathcal{S}, y)$ `[solve` QP_λ`]`
  foreach $j = 1, \ldots, n$ **do** $w_j \leftarrow 1/(\epsilon + x_j)$ `[update the weights]`
  $\lambda \leftarrow \lambda/2$ `[reduce the regularization parameter]`
output: $\mathcal{S}$, x

Gribonval [11] describe alternative strategies for selecting the weights.) In some sense, the weights computed in one iteration serve to "precondition" the next BP problem, encouraging significant coefficients to remain in the support, and discouraging small or zero coefficients from remaining or entering the support. The positive parameter ϵ_k ensures that the weights remain bounded, so it is possible for even heavily weighted coefficients to enter the support on subsequent iterations.

The weighted BP problem (8.1) is the dual of QP_λ with $\lambda = 0$ and $u = -\ell = w_k$. Because the weights may decrease (i.e., ℓ and u may tighten), the solution of one weighted problem may become infeasible for the next. Thus we apply the elastic active-set method of Algorithm 2 to QP_λ with appropriate bounds, using the solution of each weighted problem to warm-start the next.

Rather than solving each weighted BP problem to optimality (via QP_0), we solve a sequence of problems QP_λ with decreasing values of λ. This approach has two benefits: it reduces the amount of effort spent on early subproblems (larger values of λ lead to easier solves), and it helps to control the size of the support. The approach is summarized in Algorithm 6, and implemented in the routine `asp_rwbp`.

The left panel of Figure 8.1 compares the recovery rates of the BP approach (labeled BP), the reweighted BP approach described in Candés et al. [6] (labeled RW-BP), and the reweighted BPDN described above (labeled RW-HT). The right panel summarizes for each approach the corresponding cost (measured in the average number of required matrix-vector products) of successful recoveries. We used the same experimental setup as described in section 5, except that m was held constant at 100, and k was varied between 10 and 60.

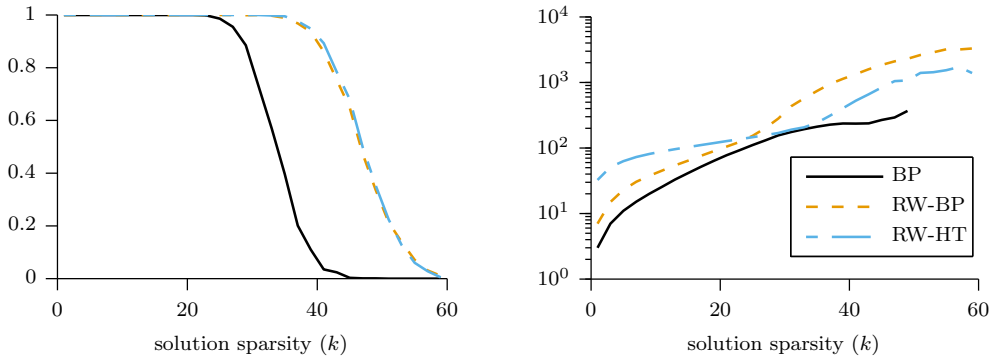
The left panel confirms the empirical observation made by Candés et al. [6] that reweighted BP can recover the sparsest solution more often than BP alone. The reweighted BP approach, using the Homotopy solver (RW-HT) seems to offer some savings over RW-BP for problems that are difficult to recover. The current RW-HT implementation naively reduces λ by half at each iteration; a more adaptive approach might further improve savings.

9. Sequential compressed sensing.

10. Sparse PageRank. Sparse solutions for $Ax \approx b$ are typically sought for over- or under-determined systems (e.g., Lasso [38] and BP), but we may also consider square systems. As described by Langville and Meyer [27], the Google PageRank eigenvector problem is equivalent to solving a sparse linear system

$$(I - \alpha H^T)x = v, \tag{10.1}$$

where H is an $n \times n$ substochastic hyperlink matrix ($H_{ij} \geq 0$, $He \leq e$), α is a scalar parameter ($0 < \alpha < 1$), v is a teleportation vector ($v \geq 0$), and the solution is



(a) Fraction of successfully recovered solutions

(b) Average number of matrix-vector products

Fig. 8.1: *Performance of elastic active-set method (Algorithm 2) used within the BP approach (BP), the reweighted BP approach (RW-BP), and the reweighted BPDN approach (RW-HT).*

nonnegative: $x = (I + \alpha H^T + (\alpha H^T)^2 + \cdots)v \geq 0$.

For the classical PageRank problem, $v = (1/n)e$ is a dense vector and we would not expect x to be sparse, although most elements will be tiny compared to the largest. For *personalized* PageRank, v is called a personalization vector and is typically $v = e_i$ for some i that might correspond to one's own homepage. We expect the exact x to have just a few significant values and the remainder to be tiny or exactly zero.

Choi and Saunders [9] suggest that personalized PageRank vectors can be approximated by basis pursuit denoising. Since x is known to be nonnegative, our simplified solver LPdual was applied to the problem

$$\underset{y}{\text{minimize}} \quad -v^T y + \tfrac{1}{2}\lambda\|y\|^2 \quad \text{subject to} \quad (I - \alpha H)y \leq e \tag{10.2}$$

in order to solve

$$\underset{x,r}{\text{minimize}} \quad \lambda e^T x + \tfrac{1}{2}\|r\|^2 \quad \text{subject to} \quad (I - \alpha H^T)x + r = v, \quad x \geq 0. \tag{10.3}$$

Small values of λ were used to approximate x in (10.1) fairly accurately. Figure 10.1 illustrates an ideal case where the exact x is sparse (only 32 nonzeros). The data for H represents the Computer Science Department of Stanford University in the year 2001. The page $i = 111$ was chosen arbitrarily; it refers to the Orientation homepage. LPdual located the nonzeros in a greedy way (32 iterations), almost in the order of largest values first. From other such examples [9] we conclude that the denoising approach to square $Ax \approx b$ can be effective when the exact x is extremely sparse.

To cover more cases, larger values of λ were used in [25] to maintain sparsity in x at the expense of larger errors in x. The aim was relaxed to finding *which elements of x are reasonably large*, without requiring accurate numerical values for those elements. Figure 10.2 illustrates using H data for all of Stanford University in 2001, with $i = 23036$ being the home page for the Computer Science Department. The true x has about 20,000 nonzeros larger than 2×10^{-3}, with just a handful of them being larger than 0.01. With $\lambda = 2 \times 10^{-3}$, LPdual chose 76 nonzeros in 76 iterations, including the significant handful. A larger λ would reduce the number of iterations but increase the absolute error already visible in the largest elements of x.

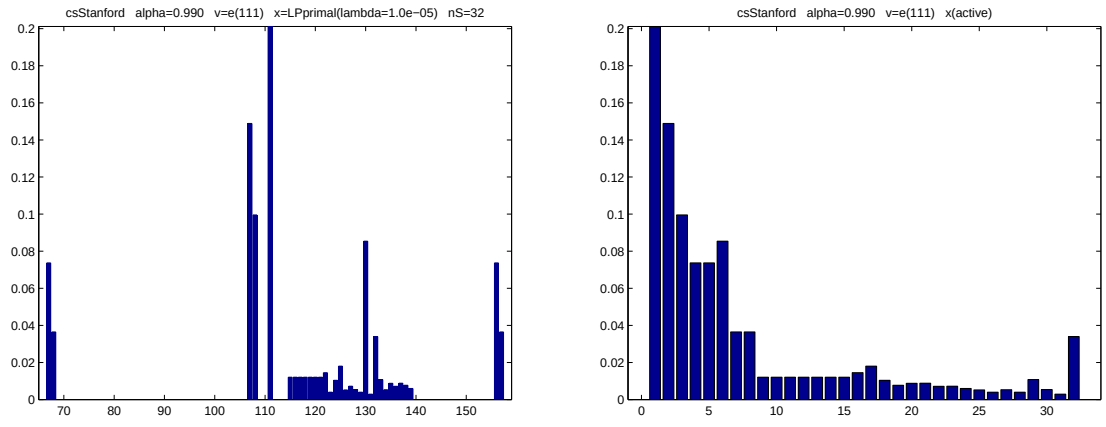


Fig. 10.1: *Stanford CS Department webmatrix H of order $n = 9914$ with $\alpha = 0.99$ and $v = e_{111}$ in* (10.1). *Left: The exact solution x with 32 nonzero entries. Right: Accurate values of those nonzeros in the order chosen by LPdual solving* (10.2)–(10.3) *with $\lambda = 10^{-5}$.*

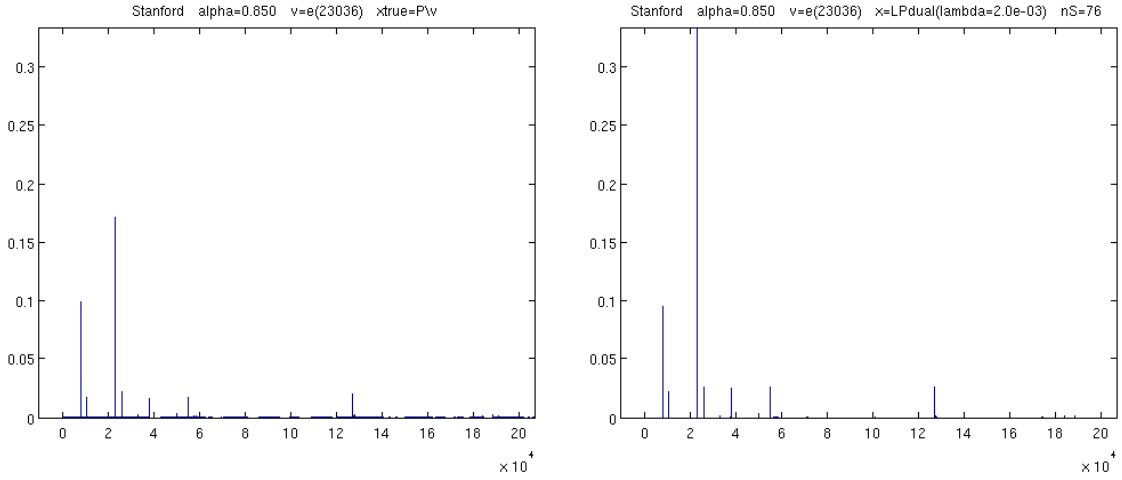


Fig. 10.2: *Stanford webmatrix H of order $n = 275000$ with $\alpha = 0.85$ and $v = e_{23036}$ in* (10.1). *Left: Significant entries in the exact solution x (about 20,000 entries larger than 2×10^{-3}). Right: Estimates of those entries chosen by LPdual solving* (10.2)–(10.3) *with $\lambda = 2 \times 10^{-3}$ (only 76 entries are nonzero).*

From more such examples [25] we conclude that BP denoising with relatively large λ can be effective as a selector for ranking the largest solution elements.

This suggests a further application to the sparse approximate inverse (SPAI) approach to computing preconditioners for iterative solvers; see xxx

11. Sparse regression with sparse solutions.

12. Generalized Lasso.

13. Other applications.

- Nonnegative least-squares: Application of `BPdual` with $\ell = -\infty$, $u = 0$ and $\lambda = 1$.
- Sparse approximate inverse (SPAI): Parallel estimation of each column of A^{-1} for some sparse matrix A, to achieve automatic computation of preconditioners for iterative solution of $Ax = b$.

REFERENCES

[1] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*, Athena Scientific, Nashua, NH, 1997.

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

[3] S. Busovaca, *Handling degeneracy in a nonlinear l1 algorithm*, Tech. Rep. CS-85-34, Computer Science Department, University of Waterloo, 1985.

[4] 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.

[5] ———, *Stable signal recovery from incomplete and inaccurate measurements*, Comm. Pure Appl. Math., 59 (2006), pp. 1207–1223.

[6] E. J. Candés, M. B. Wakin, and S. P. Boyd, *Enhancing sparsity by reweighted L1 minimization*, J. Fourier Anal. Appl., 14 (2008), pp. 877–905.

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

[8] ———, *Atomic decomposition by basis pursuit*, SIAM Rev., 43 (2001), pp. 129–159.

[9] S.-C. Choi and M. A. Saunders, *PageRank by Basis Pursuit*. Presented at ICIAM 07, Zürich, Switzerland, July 2007. `http://www.stanford.edu/group/SOL/talks.html`.

[10] A. R. Conn and J. W. Sinclair, *Quadratic programming via a nondifferentiable penalty function*, Tech. Rep. CORR 75/15, Faculty of Mathematics, University of Waterloo, Waterloo, 1975.

[11] M. E. Davies and R. Gribonval, *Restricted isometry constants where ℓ^p sparse recovery can fail for $0 < p \leq 1$*, Publication interne 1899, Institut de Recherche en Informatique et Systémes Aléatoires, July 2008.

[12] 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.

[13] D. L. Donoho, M. Elad, and V. Temlyakov, *Stable recovery of sparse overcomplete representations in the presence of noise*, IEEE Trans. Inform. Theory, 52 (2006), pp. 6–18.

[14] D. L. Donoho and X. Huo, *Uncertainty principles and ideal atomic decomposition*, IEEE Trans. Inform. Theory, 47 (2001), pp. 2845–2862.

[15] 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`, October 2006.

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

[17] M. Elad, P. Milanfar, and R. Rubinstein, *Analysis versus synthesis in signal priors*, Inverse Problems, 23 (2007), pp. 947–968.

[18] R. Fletcher, *Practical Methods of Optimization*, John Wiley and Sons, Chichester, UK, second ed., 1987.

[19] ———, *Recent developments in linear and quadratic programming*, in State of the Art in Numerical Analysis. Proceedings of the Joint IMA/SIAM Conference, A. Iserles and M. J. D. Powell, eds., Oxford University Press, Oxford, England, 1987, pp. 213–243.

[20] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, *A practical anti-cycling procedure for linearly constrained optimization*, Math. Program., 45 (1989), pp. 437–474.

[21] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, *Inertia-controlling methods for general quadratic programming*, SIAM Rev., 33 (1991), pp. 1–36.

[22] C. C. Gonzaga, *Generation of degenerate linear programming problems*, tech. rep., Dept. of Mathematics, Federal University of Santa Catarina, Santa Catarina, Brazil, April 2003.

[23] N. Gould, D. Orban, and P. Toint, *GALAHAD, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization*, ACM Trans. Math. Softw., 29 (2004), pp. 353–372.
[24] N. I. M. Gould and Ph. L. Toint, *An iterative working-set method for large-scale nonconvex quadratic programming*, Appl. Numer. Math., 43 (2002), pp. 109–128.
[25] H. H. Jin and M. A. Saunders, *Computing approximate PageRank vectors by Basis Pursuit Denoising*. Presented at SIAM Annual Meeting, San Diego, CA, July 2008. `http://www.stanford.edu/group/SOL/talks.html`.
[26] A. Koberstein and U. H. Suhl, *Progress in the dual simplex method for large scale LP problems: practical dual phase 1 algorithms*, J. Comp. Optim. Appl., 37 (2007), pp. 49–65.
[27] A. N. Langville and C. D. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*, Princeton University Press, July 2006.
[28] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 1999.
[29] O. L. Mangasarian and R. R. Meyer, *Nonlinear perturbation of linear programs*, SIAM J. Control Optim., 17 (1979), pp. 745–752.
[30] D. Needell, J. Tropp, and R. Vershynin, *Greedy signal recovery review*. Unpublished: Available at `http://arxiv.org/pdf/0812.2202`, December 2008.
[31] D. Needell and J. A. Tropp, *CoSaMP: Iterative signal recovery from incomplete and inaccurate samples*. To appear in *Appl. Comp. Harmonic Anal.*, June 2008.
[32] D. Needell and R. Vershynin, *Greedy signal recovery and uncertainty principles*, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 6814, Mar. 2008.
[33] 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.
[34] M. R. Osborne, B. Presnell, and B. A. Turlach, *On the LASSO and its dual*, J. of Computational and Graphical Statistics, 9 (2000), pp. 319–337.
[35] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, *Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition*, in Proceedings of the 27th Annual Asilomar Conference on Signals, Systems and Computers, vol. 1, Nov. 1993, pp. 40–44.
[36] M. D. Plumbley, *Recovery of sparse representations by polytope faces pursuit*, in Independent Component Analysis and Blind Signal Separation, vol. 3889/2006 of Lecture Notes in Computer Science, Springer, Berlin, 2006, pp. 206–213.
[37] D. M. Ryan and M. R. Osborne, *On the solution of highly degenerate linear programmes*, Math. Program., 41 (1986), pp. 385–392.
[38] R. Tibshirani, *Regression shrinkage and selection via the Lasso*, J. R. Statist. Soc. B., 58 (1996), pp. 267–288.
[39] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, *Sparsity and smoothness via the fused lasso*, J. Royal Statistical Society: Series B (Statistical Methodology), (2005).
[40] R. J. Tibshirani and J. Taylor, *The solution path of the generalized lasso*, Annals Statist., 39 (2011), pp. 1335–1371.
[41] J. A. Tropp and A. C. Gilbert, *Signal recovery from random measurements via orthogonal matching pursuit*, IEEE Trans. Inform. Theory, 53 (2007), pp. 4655–4666.
[42] E. van den Berg, M. P. Friedlander, G. Hennenfent, F. J. Herrmann, R. Saab, and Ö. Yilmaz, *Algorithm 890: Sparco: A testing framework for sparse reconstruction*, ACM Trans. Math. Softw., 35 (2009), pp. 29:1–29:16.