EFFICIENT EVALUATION OF SCALED PROXIMAL OPERATORS[*]

MICHAEL P. FRIEDLANDER[†] AND GABRIEL GOH[‡]

Abstract. Quadratic-support functions [Aravkin, Burke, and Pillonetto; J. Mach. Learn. Res. 14(1), 2013] constitute a parametric family of convex functions that includes a range of useful regularization terms found in applications of convex optimization. We show how an interior method can be used to efficiently compute the proximal operator of a quadratic-support function under different metrics. When the metric and the function have the right structure, the proximal map can be computed with cost nearly linear in the input size. We describe how to use this approach to implement quasi-Newton methods for a rich class of nonsmooth problems that arise, for example, in sparse optimization, image denoising, and sparse logistic regression.

Key words. support functions, proximal-gradient, quasi-Newton, interior method

AMS subject classifications. 90C15, 90C25

1. Introduction.

The proximal operator is a key ingredient in many convex optimization algorithms for problems with nonsmooth objective functions. Proximal-gradient algorithms in particular are prized for their applicability to a broad range of convex problems that arise in machine learning and signal processing, and for their good theoretical convergence properties. Under reasonable hypotheses they are guaranteed to achieve, within k iterations, a function value that is within $\mathcal{O}(1/k)$ of the optimal value using a constant step size, and within $\mathcal{O}(1/k^2)$ using an accelerated variant. Tseng [33] outlines a unified view of the many proximal-gradient variations, including accelerated versions such as FISTA [6].

In its canonical form, the proximal-gradient algorithm applies to convex optimization problems of the form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(x)+g(x), \tag{1.1}$$

where the functions $f:\mathbb{R}^n\to\mathbb{R}$ and $g:\mathbb{R}^n\to\mathbb{R}\cup\{+\infty\}$ are convex. We assume that f has a Lipschitz-continuous gradient, and that g is lower semicontinuous [29, Definition 1.5]. Typically, f is a loss function that penalizes incorrect predictions of a model, and g is a regularizer that encourages desirable structure in the solution. Important examples of nonsmooth regularizers are the 1-norm and total variation, which encourage sparsity in either x or its gradient.

Suppose that H is a positive-definite matrix. The iteration

$$x^+ = \mathbf{prox}_g^H(x - H^{-1}\nabla f(x)) \tag{1.2}$$

underlies the prototypical proximal-gradient method, where x is most recent estimate of the solution, and

$$\mathbf{prox}_g^H(z) := \underset{x\in\mathbb{R}^n}{\arg\min}\left\{\tfrac{1}{2}\|z-x\|_H^2 + g(x)\right\} \tag{1.3}$$

is the (scaled) proximal operator of g. The scaled diagonal $H=\alpha I$ is typically used to define the proximal operator because it leads to an inexpensive proximal

[*]May 25, 2017. Research supported by ONR award N00014-17-1-2009.

[†]Departments of Computer Science and Mathematics, University of British Columbia, Vancouver, BC, V6T 1Z4, Canada (mpf@cs.ubc.ca).

[‡]Department of Mathematics, University of California, Davis (gabgohjk@gmail.com).

Table 1.1
Preconditioned proximal-gradient methods.

Method	Reference	H
iterative soft thresholding	[6]	αI
symmetric rank 1	[7]	$\alpha I + ss^T$
identity-minus-rank-1	[19]	$\alpha I - ss^T$
proximal L-BFGS	[30, 31, 38]	$\Lambda + SDS^T$
proximal Newton	[13, 21, 32]	$\nabla^2 f$

iteration, particularly in the case where gT is separable. (In that case, α has a natural interpretation as a steplength.) More general matrices H, however, may lead to proximal operators that dominate the computation. The choice of H remains very much an art. In fact, there is an entire continuum of algorithms that vary based on the choice of H, as illustrated by Table 1.1. The table lists a set of algorithms roughly in order of the accuracy with which H approximates the Hessian—when it exists—of the smooth function f. At one extreme is iterative soft thresholding (IST), which approximates the Hessian using a constant diagonal. At the other extreme is proximal Newton, which uses the true Hessian. The quality of the approximation induces a tradeoff between the number of expected proximal iterations and the computational cost of evaluating the proximal operator at each iteration. Thus H can be considered a preconditioner for the proximal iteration. The proposals offered by Tran-Dinh et al. [32] and Byrd et al. [13] are flexible in the choice of H, and so those references might also be considered to apply to other methods listed in Table 1.1.

The main contribution of this paper is to show how for an important family of functions g and H, the proximal operator (1.3) can be computed efficiently via an interior method. This approach builds on the work of Aravkin et al. [2], who define the class of quadratic-support functions and outline a particular interior algorithm for their optimization. Our approach is specialized to the case where the quadratic-support function appears inside of a proximal computation. Together with the correct dualization approach (§4), this yields a particularly efficient interior implementation when the data that define g and H have special structure (§6). The proximal quasi-Newton method serves as a showcase for how this technique can be used within a broader algorithmic context (§7).

2. Quadratic-support functions.

Aravkin et al. [2] introduce the notion of a quadratic-support (QS) function, which is a generalization of sublinear support functions [29, Ch. 8E]. Here we introduce a slightly more general definition than the version implemented by Aravkin et al. We retain the "QS" designation because the quadratic term, which is an essential feature of their definition, can also be expressed by the version we use here.

Let $\mathbb{B}_p = \{\, z \mid \|z\|_p \le 1 \,\}$ and $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_k$, where each cone $\mathcal{K}_i$ is either a nonnegative orthant $\mathbb{R}^m_+$ or a second-order cone $\mathbb{Q}^m = \{\, (\tau, z) \in \mathbb{R} \times \mathbb{R}^{m-1} \mid z \in \tau\mathbb{B}_2 \,\}$. (The size m of the cones may of course be different for each index i.) The notation $Ay \succeq_{\mathcal{K}} b$ means that $Ay - b \in \mathcal{K}$, and $\tau\mathbb{B}_p \equiv \{\, \tau z \mid z \in \mathbb{B}_p \,\}$. The indicator on a convex set U is denoted as

$$\delta(x \mid U) = \begin{cases} 0 & \text{if } x \in U, \\ +\infty & \text{otherwise.} \end{cases}$$

Unless otherwise specified, x is an n-vector. To help disambiguate dimensions, the p-by-p identity matrix is denoted by I_p, and the p-vector of all ones by $\mathbf{1}_p$.

We consider the class of functions $g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ that have the conjugate representation

$$g(x) = \sup_y \{\, y^T(Bx + d) \mid y \in \mathcal{Y} \,\}, \quad \text{where} \quad \mathcal{Y} = \{y \in \mathbb{R}^\ell \mid Ay \succeq_{\mathcal{K}} b\}. \tag{2.1}$$

(The term "conjugate" alludes to the implicit duality, since g may be considered as the conjugate of the indicator function to the set $\mathcal{Y}$.) We assume throughout that the feasible set $\mathcal{Y}$ is nonempty. If $\mathcal{Y}$ contains the origin, the QS function g is nonnegative for all x, and we can then consider it to be a penalty function. This is automatically true, for example, if $b \le 0$.

The formulation (2.1) is close to the standard definition of a sublinear support function [28, §13], which is recovered by setting $d = 0$ and $B = I$, and letting $\mathcal{Y}$ be any convex set. Unlike a standard support function, g is not positively homogeneous if $d \neq 0$. This is a feature that allows us to capture important classes of penalty functions that are not positively homogeneous, such as piecewise quadratic functions, the "deadzone" penalty, or indicators on certain constraint sets. These are examples that are not representable by the standard definition of a support function. Our definition springs from the quadratic-support function definition introduced by Aravkin et al. [2], who additionally allow for an explicit quadratic term in the objective and for $\mathcal{Y}$ to be any nonempty convex set. The concrete implementation considered by Aravkin et al., however, is restricted to the case where $\mathcal{Y}$ is polyhedral. In contrast, we also allow $\mathcal{K}$ to contain second-order cones. Therefore, any quadratic objective terms in the Aravkin et al. definition can be "lifted" and turned into a linear term with a second-order-cone constraint (see Example 2.2). Our definition is thus no less general.

This expressive class of functions includes many penalty functions commonly used in machine learning; Aravkin et al. give many other examples. In addition, they show how to interpret QS functions as the negative log of an associated probability density, which makes these functions relevant to maximum a posteriori estimation. In the remainder of this section we provide some examples that illustrate various regularizing functions and constraints that can be expressed as QS functions.

Example 2.1 (1-norm regularizer). The 1-norm has the QS representation

$$\|x\|_1 = \sup_y \{\, y^T x \mid y \in \mathbb{B}_\infty \,\},$$

where

$$A = \begin{pmatrix} I_n \\ -I_n \end{pmatrix}, \quad b = -\begin{pmatrix} \mathbf{1}_n \\ \mathbf{1}_n \end{pmatrix}, \quad d = 0, \quad B = I_n, \quad \mathcal{K} = \mathbb{R}^{2n}_+, \tag{2.2}$$

□

Example 2.2 (2-norm). This simple example illustrates how the QS representation (2.1) can represent the 2-norm, which is not possible using the QS formulation described by Aravkin et al. [2] where the constraints are polyhedral. With our definition, the 2-norm has the QS representation

$$\|x\|_2 = \sup_y \{\, y^T x \mid y \in \mathbb{B}_2 \,\} = \sup_y \{\, y^T x \mid (1, y) \succeq_{\mathcal{K}} 0 \,\},$$

where

$$A = \begin{pmatrix} 0 \\ I_n \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad d = 0, \quad B = I_n, \quad \mathcal{K} = \mathbb{Q}^{n+1}. \tag{2.3}$$

□

Example 2.3 (Polyhedral norms). Any polyhedral seminorm is a support function, e.g., $\|Bx\|_1$ for some matrix B. In particular, if the set $\{\, y \mid Ay \geq b \,\}$ contains the origin and is centro-symmetric, then

$$\|x\| := \sup_y \{\, y^T Bx \mid Ay \geq b \,\}$$

defines a norm if B is nonsingular, and a seminorm otherwise. This is a QS function with $d := 0$ (as will be the case for any positively homogeneous QS function) and $\mathcal{Y} := \{\, y \mid Ay \geq b \,\}$. □

Example 2.4 (Quadratic function). This example justifies the term "quadratic" in our modified definition, even though there are no explicit quadratic terms. It also illustrates the roles of the terms B and d. The quadratic function can be written as

$$\tfrac{1}{2}\|x\|_2^2 = \sup_{y,\,t} \left\{ y^T x - \tfrac{1}{2} t \;\middle|\; \|y\|_2^2 \leq t \right\} = \sup_{y,\,t} \left\{ \begin{pmatrix} y \\ t \end{pmatrix}^T \left[\begin{pmatrix} I_n \\ 0 \end{pmatrix} x - \begin{pmatrix} 0 \\ 1/2 \end{pmatrix} \right] \;\middle|\; \|y\|_2^2 \leq t \right\}.$$

Use the derivation in Appendix A to obtain the QS representation with parameters

$$A = \begin{pmatrix} 0 & 1/2 \\ 0 & 1/2 \\ I_n & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 1/2 \\ -1/2 \\ 0 \end{pmatrix}, \quad d = \begin{pmatrix} 0 \\ 1/2 \end{pmatrix}, \quad B = \begin{pmatrix} I_n \\ 0 \end{pmatrix}, \quad \mathcal{K} = \mathbb{Q}^{n+2}.$$

□

Example 2.5 (1-norm constraint). This example is closely related to the 1-norm regularizer in Example 2.1, except that the QS function is used to express the constraint $\|x\|_1 \leq 1$ via an indicator function $g = \delta(\,\cdot \mid \mathbb{B}_1)$. Write the indicator to the 1-norm ball as the conjugate of the infinity norm, which gives

$$\begin{aligned} \delta(x \mid \mathbb{B}_1) &= \sup_y \{\, y^T x - \|y\|_\infty \,\} \\ &= \sup_{y,\,\tau} \{\, y^T x - \tau \mid y \in \tau \mathbb{B}_\infty \,\} = \sup_{y,\,\tau} \{\, y^T x - \tau \mid -\tau \mathbf{1}_n \leq y \leq \tau \mathbf{1}_n \,\}. \end{aligned} \tag{2.4}$$

This is a QS function with parameters

$$A = \begin{pmatrix} -I_n & \mathbf{1}_n \\ I_n & \mathbf{1}_n \end{pmatrix}, \quad b = 0, \quad d = \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \quad B = \begin{pmatrix} I_n \\ 0 \end{pmatrix}, \quad \mathcal{K} = \mathbb{R}^{2n}.$$

□

Example 2.6 (Indicators on polyhedral cones). Consider the following polyhedral cone and its polar:

$$U = \{\, x \mid Bx \leq 0 \,\} \quad \text{and} \quad U^\circ = \{\, B^T y \mid y \leq 0 \,\}.$$

Use the support-function representation of a cone in terms of its polar to obtain

$$\delta(x \mid U) = \delta^*(\cdot \mid U^\circ)(x) = \sup_y \{\, y^T Bx \mid y \leq 0 \,\}, \tag{2.5}$$

which is an example of an elementary QS function. (See Rockafellar and Wets [29] for definitions of the polar of a convex set, and the convex conjugate.) A concrete example is the positive orthant, obtained by choosing $B = I_n$. An important example, used in isotonic regression [10], is the monotonic cone

$$U := \{\, x \mid x_i \geq x_j,\ \forall (i,j) \in \mathcal{E} \,\},$$

Here, $\mathcal{E}$ is the set of edges in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that describes the relationships between variables in $\mathcal{V}$. If we set B to be the incidence matrix for the graph, (2.5) then corresponds to the indicator on the monotonic cone U. $\square$

Example 2.7 (Distance to a cone). The distance to a cone U that is a combination of polyhedral and second-order cones can be represented as a QS function:

$$\begin{aligned}\inf_{x\in U} \|x-y\|_2 &= \inf_x \ \{\, \|x-y\|_2 + \delta(x \mid U)\,\} \\ &= \left[\delta(\cdot \mid \mathbb{B}_2) + \delta(\cdot \mid U^\circ)\right]^*(y) = \ \sup\left\{ y^T x \mid y \in \mathbb{B}_2 \cap U^\circ \right\}.\end{aligned}$$

The second equality follows from the relationship between infimal convolution and conjugates [28, §16.4]. When U is the positive orthant, for example, $g(x) = \|\max\{0,\, x\}\|_2$, where the *max* operator is taken elementwise. $\square$

3. Building quadratic-support functions. Quadratic-support functions are closed under addition, composition with an affine map, and infimal convolution with a quadratic function. In the following, let g_i be QS functions with parameters A_i, b_i, d_i, B_i, and $\mathcal{K}_i$ (with $i = 0, 1, 2$). The rules for addition and composition are described in [2], which are here summarized and amplified.

Addition rule. The function

$$h(x) := g_1(x) + g_2(x)$$

is QS with parameters

$$A = \begin{pmatrix} A_1 & \\ & A_2 \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \quad d = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix}, \quad B = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix}, \quad \mathcal{K} = \mathcal{K}_1 \times \mathcal{K}_2.$$

Concatenation rule. The function

$$h(x) := g_0(x^1) + \cdots + g_0(x^k),$$

where each partition $x^i \in \mathbb{R}^n$, is QS with parameters

$$(A,\, B) = I_k \otimes (A_0,\, B_0), \qquad (b,\, d) = \mathbf{1}_k \otimes (b_0,\, d_0), \qquad \mathcal{K} = \mathcal{K}_0 \times \overset{(k)}{\cdots} \times \mathcal{K}_0. \tag{3.1}$$

where the symbol $\otimes$ denotes the Kronecker product. The rule for concatenation follows from the rule for addition of QS functions.

Affine composition rule. The function

$$h(x) := g_0(Px - p)$$

is QS with parameters

$$A = A_0, \quad b = b_0, \quad d = d_0 - B_0 p, \quad B = B_0 P.$$

Moreau-Yosida regularization. The Moreau-Yosida envelope of g_0 is the value of the proximal operator, i.e.,

$$\mathbf{env}^{H}_{g_0}(z) := \inf_x \left\{\, \tfrac{1}{2}\|z - x\|^2_H + g_0(x)\,\right\}. \tag{3.2}$$

It follows from Burke and Hoheisel [12, Proposition 4.10] that

$$\mathbf{env}^{H}_{g_0}(z) = \sup_y \left\{\, y^T(B_0 x + d_0) - \tfrac{1}{2} y^T B_0 H^{-1} B_0^T y \mid A_0 y \succeq_{\mathcal{K}_0} b_0 \,\right\},$$

which is a QS function with parameters

$$A = \begin{pmatrix} 0 & 1/2 \\ 0 & 1/2 \\ R & 0 \\ A_0 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 1/2 \\ -1/2 \\ 0 \\ b_0 \end{pmatrix}, \quad d = \begin{pmatrix} d_0 \\ -1/2 \end{pmatrix}, \quad B = \begin{pmatrix} B_0 \\ 0 \end{pmatrix}, \quad \mathcal{K} = \mathbb{Q}^{n+2} \times \mathcal{K}_0,$$

with Cholesky factorization $R^T R = B_0 H^{-1} B_0^T$. The derivation is given in Appendix A, where we take $Q = B_0 H^{-1} B_0^T$.

Example 3.1 (Sums of norms). In applications of group sparsity [18, 37], various norms are applied to all partitions of $x = (x^1, \dots, x^p)$, which possibly overlap. This produces the QS function

$$g(x) = \|x^1\| + \cdots + \|x^p\|, \tag{3.3}$$

where each norm in the sum may be different. In particular, consider the case of adding two norms $g(x) = \|x^1\|_\triangledown + \|x^2\|_\vartriangle$. (The extension to adding three or more norms follows trivially.) First, we introduce matrices P_i that restrict x to partition i, i.e., $x^i = P_i x$, for $i = 1, 2$. Then

$$g(x) = \|P_1 x\|_\triangledown + \|P_2 x\|_\vartriangle.$$

Then we apply the affine-composition and addition rules to determine the corresponding quantities that define the QS representation of g:

$$A = \begin{pmatrix} A_\triangledown & \\ & A_\vartriangle \end{pmatrix}, \quad b = \begin{pmatrix} b_\triangledown \\ b_\vartriangle \end{pmatrix}, \quad d = 0, \quad B = \begin{pmatrix} B_\triangledown P_1 \\ B_\vartriangle P_2 \end{pmatrix}, \quad \mathcal{K} = \mathcal{K}_\triangledown \times \mathcal{K}_\vartriangle,$$

where A_i, b_i, B_i, and $\mathcal{K}_i$ (with $i = \triangledown, \vartriangle$) are the quantities that define the QS representation of the individual norms. (Necessarily, $d = 0$ because the result is a norm and therefore positive homogeneous.) In the special case where $\|\cdot\|_\triangledown$ and $\|\cdot\|_\vartriangle$ are both the 2-norm, then A_i, b_i, B_i, and $\mathcal{K}_i$ are given by (2.3) in Example 2.2. □

Example 3.2 (Graph-based 1-norm and total variation). A variation of Example 3.1 can be used to define a variety of interesting norms, including the graph-based 1-norm regularizer used in machine learning [14], and the isotropic and anisotropic versions of total variation (TV), important in image processing [24]. Let

$$g(x) = \|Nx\|_G \qquad \text{with} \qquad \|z\|_G = \sum_{i=1}^{p} \|z^i\|_2,$$

where z^i is a partition of z and N is an m-by-n matrix. For anisotropic TV and the graph-based 1-norm regularizer, N is the adjacency matrix of a graph, and each partition z^i has a single unique element, so $g(x) = \|Nx\|_1$. For isotropic TV, each partition captures neighboring pairs of variables, and N is a finite-difference matrix. The QS addition and affine-composition rules can be combined to derive the parameters of g. When $p = m$ (i.e., each z^i is a scalar), we are summing n absolute-value functions, and we use (2.2) and (3.1) to obtain

$$A = I_m \otimes \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad b = \mathbf{1}_m \otimes \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \quad d = 0, \quad B = N, \quad \mathcal{K} = \mathbb{R}_+^{2m}. \tag{3.4}$$

Now consider the variation where $p = m/2$, (i.e., each partition has size 2), which corresponds to summing $m/2$ two-dimensional 2-norms. Use (2.3) to obtain

$$A = I_{m/2} \otimes \begin{pmatrix} 0 \\ I_2 \end{pmatrix}, \quad b = \mathbf{1}_{m/2} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad d = 0, \quad B = N, \quad \mathcal{K} = \mathbb{Q}^2 \times \overset{(m/2)}{\cdots} \times \mathbb{Q}^2.$$

□

4. The proximal operator as a conic QP. We describe in this section how the proximal map (1.2) can be obtained as the solution of a quadratic optimization problem (QP) over conic constraints,

$$\underset{y}{\text{minimize}} \quad \tfrac{1}{2} y^T Q y - c^T y \quad \text{subject to} \quad Ay \succeq_{\mathcal{K}} b, \tag{4.1}$$

for some positive semidefinite ℓ-by-ℓ matrix Q and a convex cone $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_k$. The transformation to a conic QP is not immediate because the definition of the QS function implies that the proximal map involves nested optimization. Duality, however, furnishes a means for simplifying this problem.

Proposition 4.1. *Let g be a QS function. The following problems are dual pairs:*

$$\underset{x}{\textit{minimize}} \quad \tfrac{1}{2}\|z - x\|_H^2 + g(x), \tag{4.2a}$$

$$\underset{Ay \succeq_{\mathcal{K}} b}{\textit{minimize}} \quad \tfrac{1}{2} y^T B H^{-1} B^T y - (d + Bz)^T y. \tag{4.2b}$$

If strong duality holds, the corresponding primal-dual solutions are related by

$$Hx + B^T y = Hz. \tag{4.3}$$

Proof. Let

$$h_1(x) := \tfrac{1}{2}\|x - z\|_H^2 \quad \text{and} \quad h_2(x) := \sup_{y \in \mathcal{Y}} \{\, y^T (x + d) \,\}.$$

If strong duality holds, it follows from Bertsekas [9, Prop. 5.3.8] that

$$\inf_x \; h_1(x) + h_2(Bx) = -\inf_y \; h_1^*(-B^T y) + h_2^*(y), \tag{4.4}$$

where

$$h_1^*(y) = \tfrac{1}{2}\|y\|_{H^{-1}}^2 + z^T y \quad \text{and} \quad h_2^*(y) = \delta(z \mid \mathcal{Y}) - d^T y$$

are the Fenchel conjugates of h_1 and h_2, and the infima on both sides are attained. (See Rockafellar [28, §12] for the convex calculus of Fenchel conjugates.) The right-hand side of (4.4) is precisely the dual problem (4.2b). It also follows from Fenchel duality that the pair (x, y) is optimal only if

$$x \in \arg\min_x \{\, h_1(x) + y^T B x \,\}.$$

Differentiate this objective to obtain (4.3). □

Strong duality holds when $B \cdot \operatorname{ri} \operatorname{dom}(h_1) \cap \operatorname{ri} \operatorname{dom}(h_2) \neq \emptyset$. This holds, for example, when the interior of the domain of g is nonempty, since

$$\operatorname{int} \operatorname{dom}(g) \neq \emptyset \iff \operatorname{im} B \cap \operatorname{int} \operatorname{dom}(h_2) \neq \emptyset \Rightarrow B \cdot \operatorname{ri} \operatorname{dom}(h_1) \cap \operatorname{ri} \operatorname{dom}(h_2) \neq \emptyset.$$

In all of the examples in this paper, this condition holds true.

5. Primal-dual methods for conic QP. Proposition 4.1 provides a means of evaluating the proximal map of QS functions via conic quadratic optimization. There are many algorithms for solving convex conic QPs, but primal-dual methods offer a particularly efficient approach that can leverage the special structure that defines the class of QS functions. A detailed discussion of the implementation of primal-dual methods for conic optimization is given by Vandenberghe [34]. Here we summarize the main aspects that pertain to implementing these methods efficiently in our context.

The standard development of primal-dual methods for (4.1) is based on perturbing the optimality conditions, which can be stated as follows. The solution y, together with slack and dual vectors s and v, must satisfy

$$Qy - A^T v = c, \quad v \succeq_{\mathcal{K}} 0, \quad Sv = 0,$$

where the matrix S is block diagonal, and each m_i-by-m_i block S_i is either a diagonal or arrow matrix depending on the type of cone, i.e.,

$$S_i = \begin{cases} \mathbf{diag}(s_i) & \text{if } \mathcal{K}_i = \mathbb{R}^{m_i}_+, \\ \mathbf{arrow}(s_i) & \text{if } \mathcal{K}_i = \mathbb{Q}^{m_i}, \end{cases} \qquad \mathbf{arrow}(u) := \begin{pmatrix} u_0 & \bar{u}^T \\ \bar{u} & u_0 I \end{pmatrix} \quad \text{for} \quad u = (u_0, \bar{u}).$$

See Vandenberghe [34] for further details.

Now replace the complementarity condition $Sv = 0$ with its perturbation $Sv = \mu e$, where μ is a positive parameter and $e = (e^1, \dots, e^k)$, with each partition defined by

$$e^i = \begin{cases} (1, 1, \dots, 1) & \text{if } \mathcal{K}_i = \mathbb{R}^{m_i}_+, \\ (1, 0, \dots, 0) & \text{if } \mathcal{K}_i = Q^{m_i}. \end{cases}$$

A Newton-like method is applied to the perturbed optimality conditions, which we phrase as the root of the function

$$R_\mu : \begin{pmatrix} y \\ v \\ s \end{pmatrix} \mapsto \begin{pmatrix} r_d \\ r_p \\ r_\mu \end{pmatrix} := \begin{pmatrix} Qy - A^T v - c \\ Ay - s - b \\ Sv - \mu e \end{pmatrix}. \tag{5.1}$$

Each iteration of the method proceeds by systematically choosing the perturbation parameter μ (ensuring it decreases), and obtaining each search direction as the solution of the Newton system

$$\begin{pmatrix} Q & -A^T & 0 \\ A & 0 & -I \\ 0 & S & V \end{pmatrix} \begin{pmatrix} \Delta y \\ \Delta v \\ \Delta s \end{pmatrix} = - \begin{pmatrix} r_d \\ r_p \\ r_\mu \end{pmatrix}, \qquad \begin{pmatrix} y^+ \\ v^+ \\ s^+ \end{pmatrix} = \begin{pmatrix} y \\ v \\ s \end{pmatrix} + \alpha \begin{pmatrix} \Delta y \\ \Delta v \\ \Delta s \end{pmatrix}. \tag{5.2}$$

The steplength α is chosen to ensure that (v^+, s^+) remain in the strict interior of the cone.

One approach to solving for the Newton direction is to apply block Gaussian elimination to (5.2), and obtain the search direction via the following systems:

$$(Q + A^T S^{-1} V A)\Delta x = r_d + A^T S^{-1}(V r_p + r_\mu), \tag{5.3a}$$

$$\Delta v = S^{-1}(V r_p + r_\mu - V A \Delta x), \tag{5.3b}$$

$$\Delta s = V^{-1}\left(r_\mu - S\Delta v\right). \tag{5.3c}$$

```
Algorithm 1: Evaluating $\mathbf{prox}_g^H(x)$

    INPUT  : $x$, $H$, and QS function $g$ as defined by parameters $A$, $b$, $d$, $B$, $\mathcal{K}$
    OUTPUT : $\mathbf{prox}_g^H(x)$
    Step 1: Apply interior method to QP (4.2b) to obtain $y^\star$.
    Step 2: Return $H^{-1}(c - B^T y^\star)$.
```

In practice, the matrices S and V are rescaled at each iteration in order to yield search directions with favorable properties. In particular, the Nesterov-Todd rescaling redefines S and V so that $SV^{-1} = \mathbf{block}(u)$ for some vector u, where

$$\mathbf{block}(u)_i = \begin{cases} \mathbf{diag}(u_i) & \text{if } \mathcal{K}_i = \Re_+^{m_i}, \\ (2u_iu_i^T - [u_i^TJu_i]J)^2 & \text{if } \mathcal{K}_i = \mathbb{Q}^{m_i}, \end{cases} \qquad J = \begin{pmatrix} 1 & 0 \\ 0 & -I_{(m_i-1)} \end{pmatrix}. \tag{5.4}$$

The cost of the overall approach is therefore determined by the cost of solving, at each iteration, linear systems with the matrix

$$\mathcal{L}(u) := Q + A^T\mathbf{block}(u)^{-1}A, \tag{5.5}$$

which now defines the system (5.3a).

6. Evaluating the proximal operator. We now describe how to use Proposition 4.1 to transform a proximal operator (1.3) into a conic QP that can be solved by the interior algorithm described in §5. In particular, to evaluate $\mathbf{prox}_g^H(x)$ we solve the conic QP (4.1) with the definitions

$$Q := BH^{-1}B^T, \qquad c := d + Bx; \tag{6.1}$$

the other quantities A, b, and the cone $\mathcal{K}$, appear verbatim. Algorithm 1 summarizes the procedure. As we note in §5, the main cost of this procedure is incurred in Step 1, which requires repeatedly solving linear systems that involve the linear operator (5.5). Together with (6.1), these matrices have the form

$$\mathcal{L}(u) = BH^{-1}B^T + A^T\mathbf{block}(u)^{-1}A. \tag{6.2}$$

Below we offer a tour of several examples, ordered by level of simplicity, to illustrate the details involved in the application of our technique. The Sherman-Woodbury (SW) identity

$$(D + UMU^T)^{-1} = D^{-1} - D^{-1}U(M^{-1} + U^TD^{-1}U)^{-1}U^TD^{-1},$$

valid when $M^{-1} + U^TD^{-1}U$ is nonsingular, proves useful for taking advantage of certain structured matrices that arise when solving (6.2). Some caution is needed, however, because it is known that the SW identity can be numerically unstable [36].

For our purposes, it is useful to think of the SW formula as a routine that takes the elements (D, U, M) that define a linear operator $D + UMU^T$, and returns the elements (D_1, U_1, M_1) that define the inverse operator $D_1 + U_1M_1U_1^T = (D + UMU^T)^{-1}$. We assume that D and M are nonsingular. Algorithm 2 summarizes the operations needed to compute the elements of the inverse operator.

Algorithm 2: Inverse via the Sherman-Woodbury identity

```
function SWinv($D, U, M$)
1   $D_1 \leftarrow D^{-1}$
2   $U_1 \leftarrow D_1 U$
3   $M_1 \leftarrow (M^{-1} + U^T U_1)^{-1}$
4   return $D_1$, $U_1$, $M_1$
end
```

Typically, D is a structured operator that admits a fast algorithm for solving linear systems with any right-hand side, and U and M are stored explicitly as dense matrices. Step 1 computes a new operator D_1 that simply interchanges the multiplication and inversion operations of D. Step 2 applies the operator D_1 to every column of U (typically a tall matrix with few columns). Step 3 requires inverting a small matrix.

Example 6.1 (1-norm regularizer; cf. Example 2.1). Example 2.1 gives the QS representation for $g(x) = \|x\|_1$, and the required expressions for A, B, and $\mathcal{K}$. Because $\mathcal{K}$ is the nonnegative orthant, $\mathbf{block}(u) = \mathbf{diag}(u)$; cf. (5.4). With the definitions of A and B, the linear operator $\mathcal{L}$ in (6.2) simplifies to

$$\mathcal{L}(u) = H^{-1} + A^T \mathbf{diag}(u) A = H^{-1} + \Sigma,$$

where Σ is a positive-definite diagonal matrix that depends on u. If it happens that the preconditioner H has a special structure such that $H + \Sigma^{-1}$ is easily invertible, it may be convenient to apply the SW identity to obtain equivalent formulas for the inverse

$$\mathcal{L}(u)^{-1} = (H^{-1} + \Sigma)^{-1} = H - H(H + \Sigma^{-1})^{-1} H.$$

Banded, chordal, and diagonal-plus-low-rank matrices are examples of specially structured matrices that make one of these formulas for $\mathcal{L}^{-1}$ efficient. They yield the efficiency because subtracting the diagonal matrix Σ preserves the structure of either H or H^{-1}. □

In the important special case where $H = \mathbf{diag}(h)$ is diagonal, each component i of the proximal operator for the 1-norm can be obtained directly via the formula

$$[\mathbf{prox}_g^H(x)]_i = \operatorname{sign}(x_i) \cdot \max\{|x_i| - 1/h_{ii},\, 0\},$$

where h_{ii} are the diagonal elements of H. This corresponds to the well-known soft-thresholding operator. No simple formula exists, however, for more general matrices.

Example 6.2 (Graph-based 1-norm). Consider the graph-based 1-norm function from Example 3.2 induced by a graph $\mathcal{G}$ with adjacency matrix N. Substitute the definitions of A and B from (3.4) into the formula for $\mathcal{L}$ and simplify to obtain

$$\mathcal{L}(u) = N H^{-1} N^T + A^T \mathbf{diag}(u) A = N H^{-1} N^T + \Sigma,$$

where $\Sigma := A^T \mathbf{diag}(u) A$ is a positive-definite diagonal matrix. (As with Example 6.1, $\mathcal{K}$ is the positive orthant, and thus $\mathbf{block}(u) = \mathbf{diag}(u)$.) Linear systems of the form $\mathcal{L}(u)p = q$ then can be solved with the following sequence of operations outlined in Algorithm 3, in which we assume that $H = \Lambda + UMU^T$, where Λ is diagonal.

Algorithm 3: Solving the system $\mathcal{L}(u)p = q$ for the graph-based 1-norm.

```
1 $(\Lambda_1, U_1, M_1) \leftarrow$ SWinv$(\Lambda, U, M)$          [ $H^{-1} \equiv \Lambda_1 + U_1 M_1 U_1^T$ ]
2 $\Sigma_1 \leftarrow N\Lambda_1 N^T + \Sigma$
3 $(\Sigma_2, U_2, M_2) \leftarrow$ SWinv$(\Sigma_1, NU_1, M_1)$          [ $\mathcal{L}(u)^{-1} \equiv \Sigma_2 + U_2 M_2 U_2^T$ ]
4 $p \leftarrow \Sigma_2 q + U_2 M_2 U_2^T q$          [solve $\mathcal{L}(u)p = q$]
5 return $p$
```

Observe from the definition of H and the definition of Σ_1 in Step 2 that

$$\mathcal{L}(u) = \Sigma_1 + NU_1 M_1 U_1^T N^T,$$

and then Step 3 computes the quantities that define the inverse of $\mathcal{L}$. The bulk of the work in the above algorithm happens in Step 3, where $\Sigma_2 \equiv \Sigma_1^{-1}$ is applied to each column of NU_1 (see Step 2 of the `SWinv` function), and in Step 4, where Σ_2 is applied to q. Below we give two special cases where it is possible to take advantage of the structure of N and H in order to apply Σ_2 efficiently to a vector.

1-dimensional total variation. Suppose that the graph $\mathcal{G}$ is a path. Then the $(n-1)\times n$ adjacecy matrix is given by

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

The matrix $\Sigma_1 := N\Lambda^{-1}N^T + \Sigma$ (see Step 2 of the above algorithm) is tridiagonal, and hence equations of the form $\Sigma_1 q = p$ can be solved efficiently using standard techniques, e.g., Golub and Loan [16, Algorithm 4.3.6].

Chordal graphs. If the graph $\mathcal{G}$ is chordal, than the matrix $N^T DN$ is also chordal when D is diagonal. This implies that it can be factored in time linear with the number of edges of the graph [1]. We can use this fact to apply $\Sigma_2 \equiv \Sigma_1^{-1}$ efficiently, as follows: let $(\Sigma_3, U_3, M_3) = \texttt{SWinv}(\Sigma, N, \Lambda_1)$, which implies

$$\Sigma_2 := \Sigma_3 + U_3 M_3 U_3^T, \quad \text{where} \quad \Sigma_3 := \Sigma^{-1}, \quad M_3 := (N^T\Sigma^{-1}N + \Lambda_1)^{-1}.$$

Because $N^T\Sigma^{-1}N$ is chordal, so is M_3, and any methods efficient for solving with chordal matrices can be used when applying Σ_2. □

Example 6.3 (1-norm constraint; cf. Example 2.5). Example 2.5 gives the QS representation for the indicator function on the 1-norm ball. Because the constraints on y in (2.4) involve only bound constraints, $\mathbf{block}(u) = \mathbf{diag}(u)$. With the definitions of A and B from Example 2.5, the linear operator $\mathcal{L}$ has the form

$$\mathcal{L}(u) = \begin{pmatrix} 0 \\ I_n \end{pmatrix} H^{-1} \begin{pmatrix} 0 & I_n \end{pmatrix} + \begin{pmatrix} \mathbf{1}_n^T & \mathbf{1}_n^T \\ -I_n & I_n \end{pmatrix} \begin{pmatrix} \mathbf{diag}(u_1) & \\ & \mathbf{diag}(u_2) \end{pmatrix} \begin{pmatrix} \mathbf{1}_n & -I_n \\ \mathbf{1}_n & I_n \end{pmatrix},$$

where $u = (u_1, u_2)$. Thus, $\mathcal{L}$ simplifies to

$$\mathcal{L}(u) = \begin{pmatrix} \mathbf{1}_n^T u & (u^-)^T \\ u^- & H^{-1} + \Sigma \end{pmatrix} \qquad \text{where} \qquad \Sigma := \mathbf{diag}(u^+), \qquad \begin{aligned} u^+ &:= u_1 + u_2, \\ u^- &:= -u_1 + u_2. \end{aligned}$$

Systems that involve $\mathcal{L}$ can be solved by pivoting on the block $(H^{-1}+\Sigma)$. The cases where this approach is efficient are exactly those that are efficient in the case of Example 6.1. $\square$

Example 6.4 (2-norm; cf. Example 2.2). Example 2.2 gives the QS representation for the 2-norm function. Because $\mathcal{K}=Q^n$, then $\mathbf{block}(u)=(2uu^T-[u^TJu]J)^2$, where $u=(u_0,\bar{u})$ and J is specified in (5.4). With the expressions for A and B from Example 2.2, the linear operator $\mathcal{L}$ reduces to

$$\mathcal{L}(u)=H^{-1}+\alpha I_n+vv^T, \quad \text{with} \quad \alpha=(u^TJu)^2,\ v=\sqrt{8u_0}\cdot\bar{u}. \tag{6.3}$$

This amounts to a perturbation of H^{-1} by a multiple of the identity, followed by a rank-1 update. Therefore, systems that involve $\mathcal{L}$ can be solved at the cost of solving systems with $H+\alpha I_n$ (for some scalar α).

Of course, the proximal map of the 2-norm is easily computed by other means; our purpose here is to use this as a building block for more useful penalties, such as Example 3.1, which involves the sum-of-norms function shown in (3.3). Suppose that the p partitions do not overlap, and have size n_i for $i=1,\dots,p$. The operator $\mathcal{L}$ in (6.3) generalizes to

$$\mathcal{L}(u)=H^{-1}+\underbrace{\begin{pmatrix}\alpha_1 I_{n_1}+v^1(v^1)^T & & \\ & \ddots & \\ & & \alpha_p I_{n_p}+v^p(v^p)^T\end{pmatrix}}_{W}, \qquad \begin{aligned} u^i&=(u_0^i,\bar{u}^i)\\ \alpha_i&=(u^{iT}Ju^i)^2\\ v^i&=\sqrt{8u_0^i}\cdot\bar{u}^i,\end{aligned}$$

where each vector u_i has size n_i+1.

When p is large, we can treat each diagonal block of W as an individual (small) diagonal-plus-rank-1 matrix. If H^{-1} is diagonal-plus-low-rank, for example, the diagonal part of H^{-1} can be subsumed into W. In that case, each diagonal block in W remains diagonal-plus-rank-1, which can be inverted in parallel by handling each block individually. Subsequently, the inverse of $\mathcal{L}$ can be obtained by a second correction.

Another approach, when p is small, is to consider W as a diagonal-plus-rank-p matrix:

$$W=\begin{pmatrix}\alpha_1 I_{n_1} & & \\ & \ddots & \\ & & \alpha_p I_{n_p}\end{pmatrix}+\begin{pmatrix}v^1 & & \\ & \ddots & \\ & & v^p\end{pmatrix}\begin{pmatrix}v^1 & & \\ & \ddots & \\ & & v^p\end{pmatrix}^T. \qquad \square$$

This representation is convenient: systems involving $\mathcal{L}$ can be solved efficiently in a manner identical to that of Example 6.1 because W is a diagonal-plus-low-rank matrix.

Example 6.5 (separable QS functions). Suppose that g is separable, i.e.,

$$g(x)=\gamma(x_1)+\cdots+\gamma(x_n),$$

where $\gamma:\mathbb{R}\to\mathbb{R}$ is a QS function with parameters $(A_\gamma,b_\gamma,B_\gamma,d_\gamma,\mathbb{R}^{np}_+)$, and p is an integer parameter that depends on γ. The parameters A and B for g follow from the concatenation rule (3.1), and $A=(I_n\otimes A_\gamma)$ and $B=(I_n\otimes B_\gamma)$. Thus, the linear operator $\mathcal{L}$ is given by

$$\mathcal{L}(u)=(I_n\otimes B_\gamma)H^{-1}(I_n\otimes B_\gamma)^T+(I_n\otimes A_\gamma)^T\,\mathbf{diag}(u)(I_n\otimes A_\gamma).$$

Apply the SW identity to obtain

$$\mathcal{L}(u)^{-1} = \Lambda^{-1} - \Lambda^{-1}(I_n \otimes B_\gamma)(H+\Sigma)^{-1}(I_n \otimes B_\gamma)^T\Lambda^{-1},$$

where $\Lambda = \mathbf{diag}(\Lambda_1, \dots, \Lambda_n)$,

$$\Lambda_i = A_\gamma^T \mathbf{diag}(u^i) A_\gamma, \qquad \text{and} \qquad \Sigma = \mathbf{diag}(B_\gamma^T \Lambda_1^{-1} B_\gamma, \dots, B_\gamma^T \Lambda_n^{-1} B_\gamma).$$

Because the function γ takes a scalar input, B_γ is a vector. Hence Σ is a diagonal matrix. Note too that Λ is a block diagonal matrix with n blocks each of size p. We can then solve the system $\mathcal{L}(u)p = q$ with the following steps:

```
1 $q_1 \leftarrow (I_n \otimes B_\gamma)^T \Lambda^{-1} q$
2 $q_2 \leftarrow (H + \Sigma)^{-1} q_1$
3 $q_3 \leftarrow \Lambda^{-1} q_2 - \Lambda^{-1}(I_n \otimes B_\gamma) q_2$
```

The cost of solving systems with the operator $\mathcal{L}$ is dominated by solves with the block diagonal matrix Λ (Steps 1 and 3) and $H + \Sigma$ (Step 2). The cost of the latter linear solve is explored in Example 6.1. □

7. A proximal quasi-Newton method.

We now turn to the proximal-gradient method discussed in §1. Our primary goal is to demonstrate the feasibility of the interior approach for evaluating proximal operators of QS functions. A secondary goal is to illustrate how this technique leads to an efficient extension of the quasi-Newton method for nonsmooth problems of practical interest.

We follow Scheinberg and Tang [30] and implement a limited-memory BFGS (L-BFGS) variant of the proximal-gradient method that has no linesearch and that approximately evaluates the proximal operator. Scheinberg and Tang establish a sublinear rate of convergence for this method when the Hessian approximations are suitably modified by adding a scaled identity matrix, and when the scaled proximal maps are evaluated with increasing accuracy. In their proposal, the accuracy of the proximal evaluation is based on bounding the value of the approximation to (3.2). We depart from this criterion, however, and instead use the residual (5.1) obtained by the interior solver to determine the required accuracy. In particular, we require that the optimality criterion of the interior algorithm used to evaluate the operator is a small multiplicative constant κ of the current optimality of the outer proximal-gradient iterate, i.e.,

$$\|R_\mu(y, v, s)\| \le \kappa \|x_k - \mathbf{prox}_g(x_k - \nabla f(x_k))\|.$$

This heuristic is reminiscent of the accuracy required of the linear solves used by an inexact Newton method for root finding [15]. Note that the proximal map $\mathbf{prox}_g \equiv \mathbf{prox}_g^I$ used above is unscaled, which in many cases can be easily computed when g is seperable.

7.1. Limited-memory BFGS updates.

Here we give a brief outline the L-BFGS method for obtaining Hessian approximations of a smooth function f. We follow the notation of Nocedal and Wright [25, §6.1], who use H_k to denote the current approximation to the *inverse* of the Hessian of f. Let x_k and x_{k+1} be two consecutive iterates, and define the vectors

$$s_k = x_{k+1} - x_k, \qquad \text{and} \qquad y_k = \nabla f(x_{k+1}) - \nabla f(x_k).$$

A "full memory" BFGS method updates the approximation H_k via the recursion

$$H_0 = \sigma I, \qquad H_{k+1} = H_k - \frac{H_k s_k s_k^T H_k}{s_k^T H_k s_k} + \frac{y_k y_k^T}{y_k^T s_k},$$

for some positive parameter σ that defines the initial approximation. The limited-memory variant of the BFGS update (L-BFGS) maintains the most recent m pairs (s_k, y_k), discarding older vectors. In all cases, $m \ll n$, e.g., $m = 10$. The globalization strategy advocated by Scheinberg and Tang [30] may add a small multiple of the identity to H_k. This modification takes the place of a potentially expensive linesearch, and the correction is increased at each iteration if a certain condition for decrease is not satisfied.

Each interior iteration for evaluating the proximal operator depends on solving linear systems with $\mathcal{L}$ in (6.2). In all of the experiments presented below, each interior iteration has a cost that is linear in the number of variables n.

8. Numerical experiments.

We have implemented the proximal quasi-Newton method as a Julia package [11], called `QSip`, designed for problems of the form (1.1), where f is smooth and g is a QS function. The code is available at the URL

`https://github.com/MPF-Optimization-Laboratory/QSip.jl`

A primal-dual interior method, based on ideas from the CVXOPT software package [1], is used for Algorithm 1. We consider below several examples. The first three examples apply the `QSip` solver to minimize benchmark least-squares problems with different nonsmooth regularizers that are QS representable; the last example applies the solver to a sparse logistic-regression problem on a standard data set.

8.1. Timing the proximal operator.

The examples that we explored in §6 have a favorable structure that allows each interior iteration for evaluating the proximal map $\mathbf{prox}_g^H(x)$ to scale linearly with problem size. In this section we verify this behavior empirically for problems with the structure

$$H = I + UU^T, \qquad g(x) = \|x\|_1, \qquad U \in \mathbb{R}^{n\times k} \tag{8.1}$$

for different values of k and n. This choice of diagonal-plus-low-rank matrices is designed to mimic the structure of matrices that appear in L-BFGS. Here U and x are chosen with random normal entries. As described in Example 2.1, the system $\mathcal{L}(u)$ is inverted in linear time using the SW identity.

We evaluate the proximal map on 100 random instances for each combination of k and n, and plot in Figure 1 the average time needed to reach an accuracy of 10^{-7}, as measured by the optimality conditions in the interior algorithm. Because in practice the number of iterations of the interior method is almost independent of the size of the problem, the time taken to compute the proximal map is a predictable, linear function of the size of the problem.

8.2. Synthetic least-square problems.

The next set of examples all involve the least-squares objective

$$f(x) = \tfrac{1}{2}\|Ax - b\|_2^2. \tag{8.2}$$

Two different procedures are used to construct matrices A, as described in the following sections. In all cases, we follow the testing approach described by Lorenz [23] for constructing a test problem with a known solution: fix a vector $x^\star$ and choose $b = Ax^\star - A^{-T}v$, where $v \in \partial g(x^\star)$. Note that

$$\partial(f+g)(x^\star) = A^T(Ax^\star - [Ax^\star - A^{-T}v]) + \partial g(x^\star) = \partial g(x^\star) - v.$$

Because $v \in \partial g(x^\star)$, the above implies that $0 \in \partial(f+g)(x^\star)$, and hence $x^\star$ minimizes the objective $f + g$. In the next three sections, we apply `QSip` in turn to problems with g equal to the 1-norm, the group LASSO (i.e., sum of 2-norm functions), and total variation.

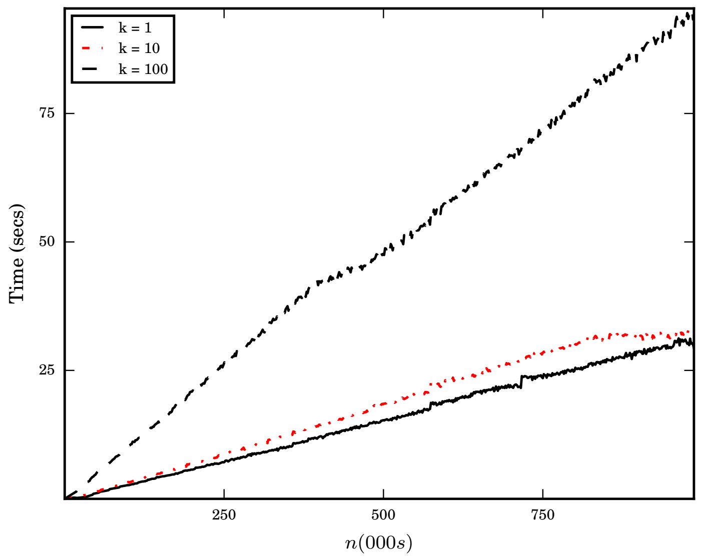

FIG. 8.1. *Time taken to compute* $\mathbf{prox}_g^H(x)$ *versus* n*, for* $k = 1, 10, 100$*; see* (8.1).

8.2.1. One-norm regularization. In this experiment we choose $g = \|\cdot\|_1$, which gives the 1-norm regularized least-squares problem, often used in applications of sparse optimization. Following the details in Example 2.1, the system $\mathcal{L}(u)$ is a diagonal-plus-low-rank matrix, which we invert using the SW identity.

The matrix A in (8.2) is a 2000-by-2000 lower triangular matrix with all nonzero entries equal to 1. The bandwidth p of A is adjustable, and determines its coherence

$$\text{coherence}(A) = \max_{i \neq j} \frac{a_i^T a_j}{\|a_i\|\|a_j\|} = \sqrt{\frac{p-1}{p}},$$

where a_i is the ith column. As observed by Lorenz [23], the difficulty of 1-norm regularized least-squares problems are strongly influenced by the coherence. Our experiments use matrices A with bandwidth $p = 500, 1000, 2000$.

Figure 8.2 shows the results of applying the `QSip` solver with a memories $k = 1, 10$, labeled "QSIP mem $= k$". We also consider comparisons against two competitive proximal-based methods. The first is a proximal-gradient algorithm that uses the Barzilai-Borwein steplength [5, 35]. This is our own implementation of the method, and is labeled "Barzilai-Borwein" in the figures. The second is the proximal quasi-Newton method implemented by Becker and Fadili [7], which is based on a symmetric-rank-1 Hessian approximation; this code is labeled "PG-SR1". The `QSip` solver with memory of 10 outperforms the other solvers. The quasi-Newton approximation benefits problems with high coherence (p large) more than problems with low coherence (p small). In all cases, the experiments reveal that the additional cost involved in evaluating a proximal operator (via an interior method) is balanced by the overall cost of the algorithm, both in terms of iterations (i.e., matrix-vector products with A) and time.

8.2.2. The effect of conditioning. It is well known that the proximal-gradient method converges slowly for ill conditioned problems. The proximal L-BFGS method may help to improve convergence in such situations. We investigate the observed convergence rate of the proximal L-BFGS approach on a family of least-squares

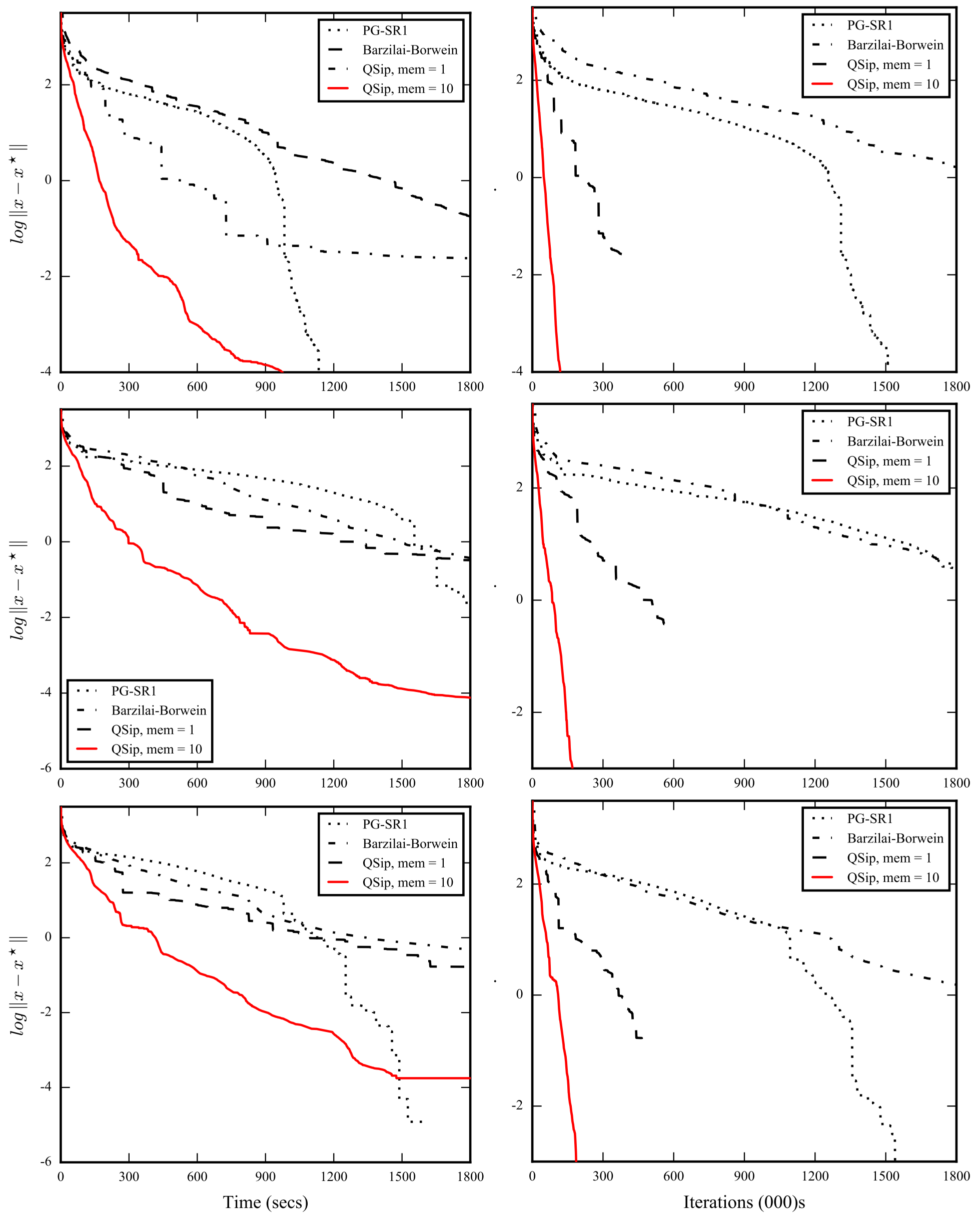

Fig. 8.2. *Performance of solvers applied to 1-norm regularized least-squares problems of increasing difficulty. The left and right columns, respectively, track the distance of the current solution estimate to the true solution versus time and iteration number. Top row:* $p = 2000$ *(highest coherence); middle row:* $p = 1000$*; bottom row:* $p = 500$ *(lowest coherence).*

problems with 1-norm regularization with varying degrees of ill conditioning. For these experiments, we take A in (8.2) as the 2000-by-2000 matrix

$$A = \alpha_L \begin{pmatrix} T & 0 \\ 0 & 0 \end{pmatrix} + \alpha_\mu I,$$

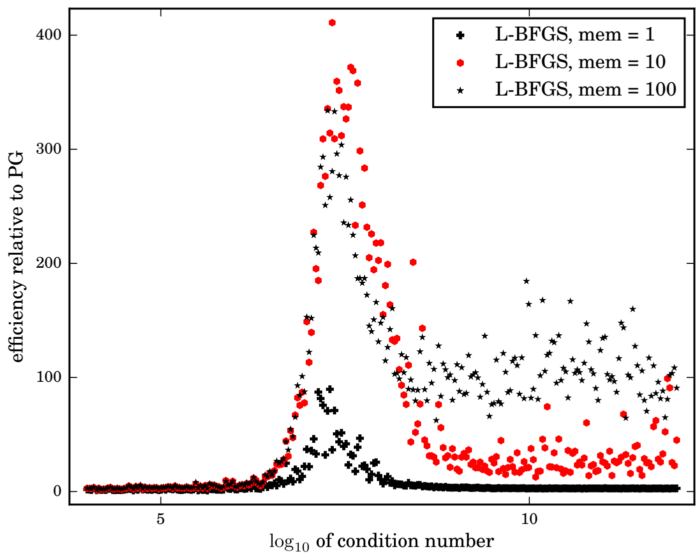

FIG. 8.3. *Performance of the proximal quasi-Newton method relative to proximal gradient for problems of varying condition number.*

where T is a 1000-by-1000 tridiagonal matrix with constant diagonal entries equal to 2, and constant sub- and super-diagonal entries equal to -1. The parameter α_L/α_μ controls the conditioning of A, and hence the conditioning of the Hessian A^TA of f.

We run L-BFGS with 4 different memories ("mem"): 0 (i.e., proximal gradient with a Barzilai-Borwein steplength), 1, 10, and 100. We terminate the algorithm either when the error drops beneath 10^{-8}, or the method reaches 10^3 iterations. Our method of measuring the observed convergence (OC) computes the line of best fit to the log of optimality versus k, which results in the quantity

$$\text{Observed Convergence} := \frac{\sum_{k=0}^{N} k \cdot \log \|x_k - x_*\|}{\sum_{k=0}^{N} \log \|x_k - x_*\|},$$

where N is the total number of iterations.

The plot in Figure 8.3 shows the ratio of the OC for L-BFGS relative to the observed convergence of proximal gradient (PG). This quantity can be interpreted the amount of work that a single quasi-Newton step performs relative to the number of PG iterations. The plot reveals that the quasi-Newton method is faster at all condition numbers, but is especially effective for problems with moderate conditioning. Also, using a higher quasi-Newton memory almost always lowers the number of iterations. This benefit is most pronounced when the problem conditioning is poor.

Together with §8.1, this section gives a broad picture of the trade-off between the proximal quasi-Newton and proximal gradient methods. The time required for each proximal gradient iteration is dominated by the cost of the gradient computation because the evaluation of the unscaled proximal operator is often trivial. On the other hand, the proximal quasi-Newton iteration additionally requires evaluating the scaled proximal operator. Therefore, the proximal quasi-Newton method is most appropriate when this cost is small relative to the gradient evaluation.

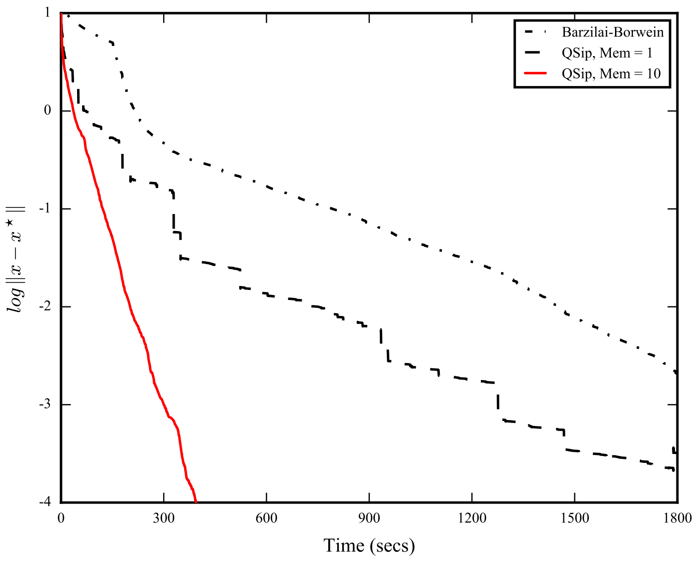

Fig. 8.4. *Performance of solvers applied to a group-Lasso problem. The horizontal axis measures elapsed time; the vertical axis measures distance to the solution.*

8.2.3. Group LASSO. Our second experiment is based on the sum-of-norms regularizer described in Examples 3.1 and 6.4. In this experiment, the n-vector (with $n = 2000$) is partitioned into $p = 5$ disjoint blocks of equal size. The matrix A is fully lower triangular.

Figure 8.4 clearly shows that the `QSip` solver outperforms the PG method with the Barzilai-Borwein step size. Although we required `QSip` to exit with a solution estimate accurate within 6 digits (i.e., $\log \|x - x^*\| \leq 10^{-6}$), the interior solver failed to achieve the requested accuracy because of numerical instability with the SW formula used for solving the Newton system. This raises the question of how to use efficient alternatives to the SW update that are numerically stable and can still leverage the structure of the problem.

8.2.4. 1-dimensional total variation. Our third experiment sets

$$g(x) = \sum_{i=1}^{n-1} |x_{i+1} - x_i|,$$

which is the anisotropic total-variation regularizer described in Examples 3.2 and 6.2. The matrix A is fully lower triangular. Figure 8.5 compares the convergence behavior of `QSip` with the Barzilai-Borwein proximal solver. The Python package `prox-tv` [3, 4] was used for the evaluation of the (unscaled) proximal operator, needed by the Barzilai-Borwein solver. The `QSip` solver, with memories of 1 and 10, outperformed the Barzilai-Borwein solver.

8.3. Sparse logistic regression. This next experiment tests `QSip` on the sparse logistic-regression problem problem

$$\underset{x}{\text{minimize}} \quad \frac{1}{N}\sum_{i=1}^{N} \log(1 + \exp[a_i^T x]) + \lambda\|x\|_1,$$

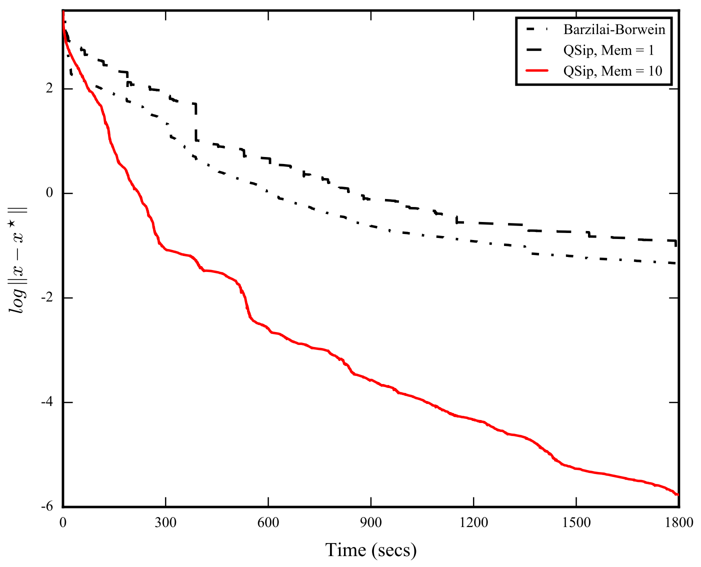

FIG. 8.5. *Performance of the `QSip` solver applied to a 1-dimensional total-variation problem.*

where N is the number of observations. The *Gisette* [17] and *Epsilon* [27] datasets, standard benchmarks from the UCI Machine Learning Repository [22], are used for the feature vectors a_i. *Gisette* has $5K$ parameters and $13.5K$ observations; *Epsilon* has $2K$ parameters with $400K$ observations. These datasets were chosen for their large size and modest number of parameters. In all of these experiments, $\lambda = 0.01$.

Figure 8.6 compares `QSip` to the Barzilai-Borwein solver, and to newGLMNet [20], a state-of-the-art solver for sparse logistic regression. (Other possible comparisons include the implementation of Scheinberg and Tang [30], which we do not include because of difficulty compiling that code.) Because we do not know a priori the solution for this problem, the vertical axis measures the log of the optimality residual $\|x_k - \mathbf{prox}_g(x_k - \nabla f(x_k))\|_\infty$ of the current iterate. (The norm of this residual necessarily vanishes at the solution.) On the *Gisette* dataset, Barzilai-Borwein and newGLMNNet are significantly faster than the proximal quasi-Newton implementation. On the *Epsilon* dataset, however, the quasi-Newton is faster at all levels of accuracy.

9. Conclusion.

Much of our discussion revolves around techniques for solving the Newton systems (5.2) that arise in the implementation of an interior method for solving QPs. The Sherman-Woodbury formula features prominently because it is a convenient vehicle for taking advantage of the structure of the Hessian approximations and the structured matrices that typically define QS functions. Other alternatives, however, may be preferable, depending on the application.

For example, we might choose to reduce the 3-by-3 matrix in (5.2) to an equivalent symmetrized system

$$\begin{pmatrix} -Q & A^T \\ A & D \end{pmatrix} \begin{pmatrix} \Delta y \\ \Delta s \end{pmatrix} = - \begin{pmatrix} -r_d \\ r_p + V^{-1} r_\mu \end{pmatrix}$$

with $D := V^{-1}S$. As described by Benzi and Wathen [8], Krylov-based method, such

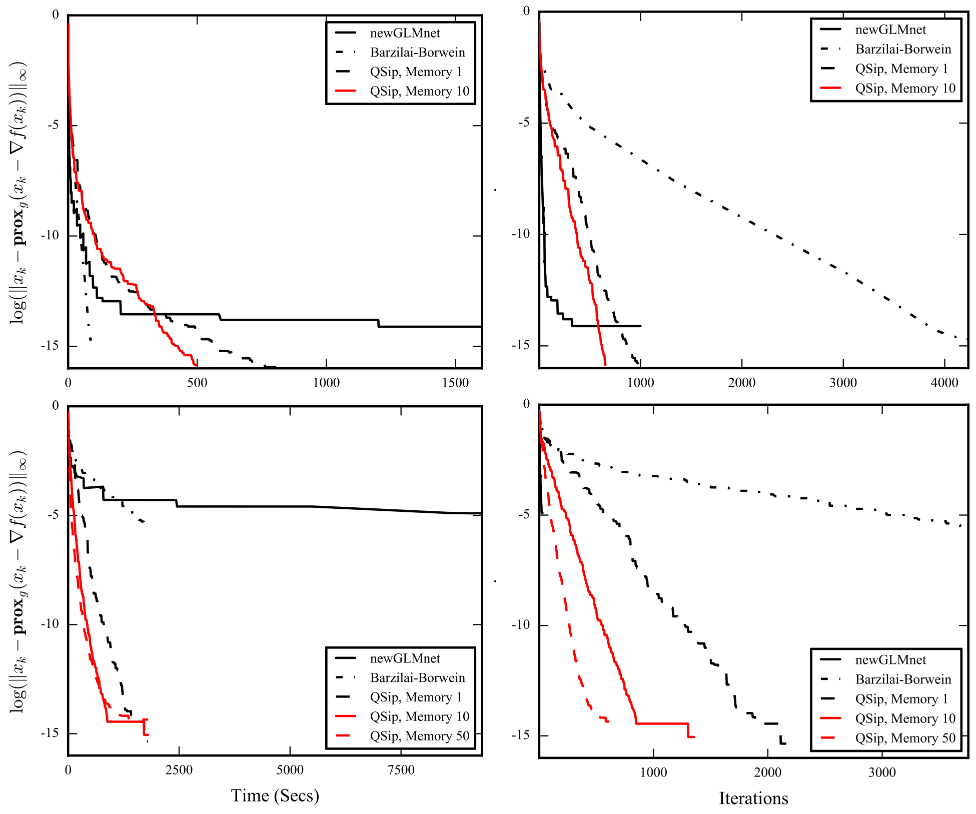

FIG. 8.6. *Performance of solvers on a sparse logistic-regression problem. Top row:* Gisette *dataset; bottom row:* Epsilon *dataset. The left and right columns, respectively, track the optimality of the current solution estimate versus elapsed time and iteration number.*

as MINRES [26], may be applied to a preconditioned system, using the preconditioner

$$P = \begin{pmatrix} -\mathcal{L}(u) & \\ & D \end{pmatrix},$$

where $\mathcal{L}(u)$ is defined in (5.5). This "ideal" preconditioner clusters the spectrum into three distinct values, so that in exact arithmetic, MINRES would converge in three iterations. The application of the preconditioner requires solving systems with $\mathcal{L}$ and D, and so all of the techniques discussed in §6 apply. One benefit, however, which we have not explored here, is that the preconditioning approach allows us to approximate $\mathcal{L}^{-1}(u)$, rather than to compute it exactly, which may yield computational efficiencies for some problems.

Acknowledgments. The authors are grateful to three anonymous referees for their thoughtful suggestions, and for a number of critical corrections. We also wish to thank Fiorella Sgallari for handling this paper as Associate Editor.

References.

[1] M. Andersen, J. Dahl, and L. Vandenberghe. Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones. *Math. Program. Comp.*, 2(3-4):167–201, 2010.

[2] A. Y. Aravkin, J. V. Burke, and G. Pillonetto. Sparse/robust estimation and Kalman smoothing with nonsmooth log-concave densities: Modeling, computation, and theory. *J. Mach. Learn. Res.*, 14(1):2689–2728, 2013.

[3] A. Barbero and S. Sra. Fast Newton-type methods for total variation regularization. In L. Getoor and T. Scheffer, editors, *Intern. Conf. on Machine Learning*, pages 313–320. Omnipress, 2011.

[4] A. Barbero and S. Sra. Modular proximal optimization for multidimensional total-variation regularization, 2014. arXiv 0902.0885.

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

[6] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imag. Sci.*, 2(1):183–202, 2009.

[7] S. Becker and J. Fadili. A quasi-newton proximal splitting method. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2618–2626. Curran Associates, Inc., 2012.

[8] M. Benzi and A. J. Wathen. Some preconditioning techniques for saddle point problems. In *Model order reduction: theory, research aspects and applications*, pages 195–211. Springer, 2008.

[9] D. P. Bertsekas. *Convex optimization theory.* Athena Scientific Belmont, MA, 2009.

[10] M. J. Best and N. Chakravarti. Active set algorithms for isotonic regression; a unifying framework. *Math. Program.*, 47(1):425–439, 1990.

[11] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing, November 2014.

[12] J. V. Burke and T. Hoheisel. Epi-convergent smoothing with applications to convex composite functions. *SIAM J. Optim.*, 23(3):1457–1479, 2013.

[13] R. H. Byrd, J. Nocedal, and F. Oztoprak. An inexact successive quadratic approximation method for L-1 regularized optimization. *Math. Program.*, 157(2): 375–396, 2016.

[14] H. H. Chin, A. Madry, G. L. Miller, and R. Peng. Runtime guarantees for regression problems. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 269–282. ACM, 2013.

[15] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19(2):400–408, 1982.

[16] G. H. Golub and C. F. V. Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, second edition, 1989.

[17] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror. Result analysis of the NIPS 2003 feature selection challenge. *Advances Neural Inform. Processing Systems*, 17: 545–552, 2004.

[18] R. Jenatton, J. Mairal, F. R. Bach, and G. R. Obozinski. Proximal methods for sparse hierarchical dictionary learning. In *Proc. 27th Intern. Confer. Machine Learning (ICML-10)*, pages 487–494, 2010.

[19] S. Karimi and S. Vavasis. IMRO: a proximal quasi-Newton method for solving l_1-regularized least squares problem, 2014.

[20] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale L1-regularized least squares. *IEEE J. Sel. Top. Signal Process.*, 1(4):606–617, 2007.

[21] J. D. Lee, Y. Sun, and M. A. Saunders. Proximal Newton-type methods for

minimizing composite functions. *SIAM J. Optim.*, 24(3):1420–1443, 2014.

[22] M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

[23] D. A. Lorenz. Constructing test instances for basis pursuit denoising. *IEEE Trans. Sig. Proc.*, 61(5):1210–1214, March 2013.

[24] Y. Lou, T. Zeng, S. Osher, and J. Xin. A weighted difference of anisotropic and isotropic total variation model for image processing. Technical report, Department of Mathematics, UCLA, 2014.

[25] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, New York, 1999.

[26] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12(4):617–629, 1975.

[27] Pascal Large Scale Learning Challenge. http://largescale.ml.tu-berlin.de/instructions, 2016. Accessed: 2016-11-22.

[28] R. T. Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, 1970.

[29] R. T. Rockafellar and R. J. B. Wets. *Variational Analysis*, volume 317. Springer, 1998. 3rd printing.

[30] K. Scheinberg and X. Tang. Practical inexact proximal quasi-Newton method with global complexity analysis. *Math. Program.*, 160(1):495–529, 2016.

[31] M. Schmidt, E. van den Berg, M. P. Friedlander, and K. Murphy. Optimizing costly functions with simple constraints: a limited-memory projected quasi-Newton algorithm. In *Proc. 12th Inter. Conf. Artificial Intelligence and Stat.*, pages 448–455, April 2009.

[32] Q. Tran-Dinh, A. Kyrillidis, and V. Cevher. Composite self-concordant minimization. *J. Machine Learning Research*, 16:371–416, 2015.

[33] P. Tseng. Approximation accuracy, gradient methods, and error bound for structured convex optimization. *Math. Program.*, 125:263–295, 2010.

[34] L. Vandenberghe. The CVXOPT linear and quadratic cone program solvers, 2010.

[35] S. J. Wright, R. D. Nowak, and M. A. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Trans. Sig. Proc.*, 57(7):2479–2493, 2009.

[36] E. Yip. A note on the stability of solving a rank-p modification of a linear system by the Sherman-Morrison-Woodbury formula. *SIAM J. Sci. Stat. Comput.*, 7(2): 507–513, 1986.

[37] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. Royal Stat. Soc. B.*, 68, 2006.

[38] K. Zhong, E. hsu Yen, I. S. Dhillon, and P. K. Ravikumar. Proximal quasi-newton for computationally intensive l1-regularized m-estimators. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances Neural Inform. Processing Systems 27*, pages 2375–2383. Curran Associates, Inc., 2014.

Appendix A. QS Representation for a quadratic.

Here we derive the QS representation of a support function that includes an explicit quadratic term:

$$g(x) = \sup_{y} \{\, y^T(B_0x + d_0) - \tfrac{1}{2}y^TQy \mid A_0y \succeq_{\mathcal{K}_0} b_0 \,\}\,.$$

Let R be such that $R^TR = Q$. We can then write the quadratic function in the objective as a constraint its epigraph, i.e.,

$$g(x) = \sup_{y,\,t} \{\, y^T(B_0x + d_0) - \tfrac{1}{2}t \mid A_0y \succeq_{\mathcal{K}} b_0,\ \|Ry\|^2 \le t \,\}\,.$$

Next we write the constraint $\|Ry\|^2 \le t$ as a second-order cone constraint:

$$\begin{aligned}
\|Ry\|^2 \le t &\iff \|Ry\|^2 \le \frac{(t+1)^2 - (t-1)^2}{4} \\
&\iff \|Ry\|^2 + \left(\frac{t-1}{2}\right)^2 \le \left(\frac{t+1}{2}\right)^2 \\
&\iff \sqrt{\|Ry\|^2 + \left(\frac{t-1}{2}\right)^2} \le \frac{t+1}{2} \\
&\iff \left\| \begin{pmatrix} 0 & 1/2 \\ R & 0 \end{pmatrix} \begin{pmatrix} y \\ t \end{pmatrix} + \begin{pmatrix} -1/2 \\ 0 \end{pmatrix} \right\| \le \frac{t+1}{2} \\
&\iff \begin{pmatrix} 0 & 1/2 \\ 0 & 1/2 \\ R & 0 \end{pmatrix} \begin{pmatrix} y \\ t \end{pmatrix} \preceq_Q \begin{pmatrix} 1/2 \\ -1/2 \\ 0 \end{pmatrix}.
\end{aligned}$$

Concatenating this with the original constraints gives a QS function with parameters

$$A = \begin{pmatrix} 0 & 1/2 \\ 0 & 1/2 \\ R & 0 \\ A_0 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 1/2 \\ -1/2 \\ 0 \\ b_0 \end{pmatrix}, \quad d = \begin{pmatrix} d_0 \\ -1/2 \end{pmatrix}, \quad B = \begin{pmatrix} B_0 \\ 0 \end{pmatrix}, \quad \mathcal{K} = \mathbb{Q}^{n+2} \times \mathcal{K}_0.$$