Mathematical Programming Computation (2020) 12:1–38
https://doi.org/10.1007/s12532-019-00163-5

FULL LENGTH PAPER

A hybrid quasi-Newton projected-gradient method with application to Lasso and basis-pursuit denoising

Ewout van den Berg[1]

Received: 29 August 2017 / Accepted: 6 May 2019 / Published online: 4 June 2019

Abstract
We propose a new algorithm for the optimization of convex functions over a polyhedral set in $\mathbb{R}^n$. The algorithm extends the spectral projected-gradient method with limited-memory BFGS iterates restricted to the present face whenever possible. We prove convergence of the algorithm under suitable conditions and apply the algorithm to solve the Lasso problem, and consequently, the basis-pursuit denoise problem through the root-finding framework proposed by van den Berg and Friedlander (SIAM J Sci Comput 31(2):890–912, 2008). The algorithm is especially well suited to simple domains and could also be used to solve bound-constrained problems as well as problems restricted to the simplex.

Keywords ℓ_1 minimization · Quasi-Newton method · Projected gradient method

Mathematics Subject Classification 90C20 · 90C25 · 90C53 · 52B12

1 Introduction

In this paper we propose an algorithm for optimization problems of the form

$$\underset{x}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad x \in \mathcal{C}, \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex, twice continuously differentiable function, and $\mathcal{C}$ is a polyhedral set in $\mathbb{R}^n$. The main focus of the paper is the application and specialization of the framework to the Lasso problem [1]:

This work was partially supported by National Science Foundation Grant DMS 0906812 (American Reinvestment and Recovery Act).

✉ Ewout van den Berg
evandenberg@us.ibm.com

[1] IBM T.J. Watson Research Center, 1101 Kitchawan Rd, Yorktown Heights, NY 10598, USA

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|Ax-b\|_2^2 \quad \text{subject to} \quad \|x\|_1 \le \tau, \tag{LS$_\tau$}$$

and its extension to the weighted one-norm. The work in this paper is motivated by the need for an efficient and accurate solver for the Lasso subproblems appearing in the SPGL1 [2] solver for basis-pursuit denoise [3] problems of the form

$$\underset{x}{\text{minimize}} \quad \|x\|_1 \quad \text{subject to} \quad \tfrac{1}{2}\|Ax-b\|_2^2 \le \sigma. \tag{BP$_\sigma$}$$

Both the (BP$_\sigma$) and (LS$_\tau$) formulations are central to compressed sensing [4,5] as a means of recovering sparse or approximately sparse vectors x_0 from linearly compressed and often noisy observations $b = Ax_0 + z$. The basis-pursuit denoise formulation (BP$_\sigma$) is often a more natural choice in practice compared to the Lasso formulation, as it is parameterized in the noise level σ, rather than in τ, the one-norm of the unknown signal x_0, which may be more difficult to determine.

It was shown in [2] that basis-pursuit denoise and Lasso are connected through the Pareto curve

$$p(\tau) = \min_x \quad \|Ax-b\|_2 \quad \text{subject to} \quad \|x\|_1 \le \tau,$$

and that solving (BP$_\sigma$) can be reduced to finding the smallest τ for which the Lasso solution x_τ^* satisfies $\|Ax_\tau^* - b\| \le \sigma$. Denoting by τ_σ this critical value of τ and assuming that b lies in the range space of A it was shown in [2] that the Pareto curve is convex and differentiable at all $\tau \in [0, \tau_0)$ with gradient $\|A^T r\|_\infty / \|r\|_2$ where r denotes the misfit $Ax_\tau^* - b$. Evaluation of both $p(\tau)$ and $p'(\tau)$ relies on the misfit r, which can be obtained by solving (LS$_\tau$). The SPGL1 solver proposed in [2] applies root finding on the Pareto curve, as illustrated in Fig. 1, to solve $p(\tau) = \sigma$ and thereby reduce basis-pursuit denoise to a series of Lasso problems. In SPGL1 these subproblems are solved using the spectral projected-gradient (SPG) algorithm [6], which we discuss in more detail in Sect. 2.

For certain problems it was observed that SPG generates long sequences of iterates that all lie on the same face of the polyhedral set of feasible points. This suggests the use of an active-set type method in which a quasi-Newton method, such as the limited-memory BFGS (L-BFGS) method [7], is used to minimize the problem restricted to the currently active face. In order to obtain an efficient solver it is important to avoid unnecessarily costly subproblem solves or complicated heuristics to determine when to switch between the solvers. In this paper we therefore propose a hybrid algorithm that switches between the two methods in a seamless and lightweight fashion.

1.1 Paper outline

In Sect. 2 we provide a concise background on the SPG and L-BFGS methods along with some of their theoretical properties. We then describe the proposed algorithm for the general problem formulation (1) in Sect. 3. In Sect. 4 we study the geometry of the constraints in the Lasso problem, and develop the tools needed for an efficient

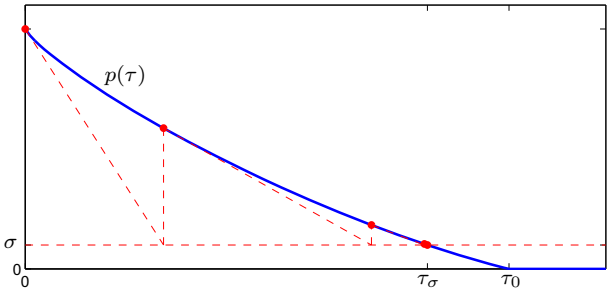

Fig. 1 Example of root finding on the Pareto curve $p(\tau)$

implementation of the framework for Lasso. Numerical experiments are provided in Sect. 5, followed by the conclusions in Sect. 6.

1.2 Notation and definitions

We use caligraphic capital letters for sets. Given any two set $\mathcal{S}_1$ and $\mathcal{S}_2$, we write $\mathcal{S}_1 + \mathcal{S}_2$ for $\{x_1 + x_2 \mid x_1 \in \mathcal{S}_1,\ x_2 \in \mathcal{S}_2\}$, and likewise for $\mathcal{S}_1 - \mathcal{S}_2$. For a seeming lack of established terminology, we define the *difference hull* of a set $\mathcal{S}$, diff hull$(\mathcal{S})$, as the linear hull of differences, span$\{u_1 - u_2 \mid u_1, u_2 \in \mathcal{S}\}$. The difference hull can be seen as the linear subspace corresponding to the affine hull of $\mathcal{S}$ translated to contain the origin. For any x in a polyhedral set $\mathcal{C}$, we define $\mathcal{F}(x)$ to be the unique face $\mathcal{F}$ of $\mathcal{C}$ for which $x \in \text{relint}(\mathcal{F})$; this may be $\mathcal{C}$ itself. The normal cone of $\mathcal{C}$ at x is given by $\mathcal{N}(x) := \{d \in \mathbb{R}^n \mid \mathcal{P}(x + d) = x\}$. The normal cones $\mathcal{N}(x)$ are identical for all $x \in \text{relint}(\mathcal{F})$, and we can therefore define the normal cone of a face $\mathcal{N}(\mathcal{F})$ as $\mathcal{N}(x)$ evaluated at any arbitrary $x \in \text{relint}(\mathcal{F})$. Orthogonal projection of any vector $v \in \mathbb{R}^n$ onto $\mathcal{C}$ is defined as

$$\mathcal{P}(v) := \arg\min_x \quad \|x - v\|_2 \quad \text{subject to} \quad x \in \mathcal{C}.$$

Given any point $x \in \text{relint}(\mathcal{F})$ we are interested in the set of directions $d \in \mathbb{R}^n$ for which there exists an $\epsilon > 0$ such that the projection of $x + \epsilon d$ lies in $\mathcal{F}$. It can be verified that

$$\begin{aligned}\mathcal{S}(x) &:= \{d \in \mathbb{R}^n \mid \exists\, \epsilon > 0 : \mathcal{F}[\mathcal{P}(x + \epsilon d)] = \mathcal{F}(x)\} \\ &= \mathcal{N}(x) + \text{diff hull}(\mathcal{F}(x)).\end{aligned}$$

Because the normal cone $\mathcal{N}(x)$ is the same for all $x \in \text{relint}(\mathcal{F})$ we can define the *self-projection cone* of a face $\mathcal{F}$ as $\mathcal{S}(\mathcal{F}) := \mathcal{N}(\mathcal{F}) + \text{diff hull}(\mathcal{F})$. Note that $\mathcal{N}(\mathcal{F}) \perp$ diff hull$(\mathcal{F})$, or stronger yet, that the difference hull of $\mathcal{F}$ is the orthogonal complement of the linear hull of $\mathcal{N}(\mathcal{F})$. For any k-face $\mathcal{F}$ of $\mathcal{C}$, $k \geq 1$, we denote by $\Phi(\mathcal{F}) \in \mathbb{R}^{n \times k}$ an arbitrary but fixed orthonormal basis for diff hull$(\mathcal{F})$. We will never use $\Phi(\mathcal{F})$ when $\mathcal{F}$ is a vertex and therefore leave it undefined. We denote by e_i the ith column of

an identity matrix whose size is clear from the context. The proximation of a function f is defined as $\text{prox}_f(u) := \arg\min_x f(x) + \frac{1}{2}\|x - y\|_2^2$.

2 Background

2.1 The nonmonotone spectral projected-gradient method

Algorithm 1 *The nonmonotone spectral projected-gradient method (SPG)*

Inputs: A, b, τ, x_0
Initialize $i \leftarrow 0$, choose $\alpha_0 \in [\alpha_{\min}, \alpha_{\max}]$
while not done **do**
  Compute $g_i = \nabla f(x_i)$

  # Compute the Barzilai-Borwein scaling parameter
  if $i > 0$ **then**
    Set $s \leftarrow x_i - x_{i-1}$, $y \leftarrow g_i - g_{i-1}$
    if $s^T y > 0$ **then**
      $\alpha_i \leftarrow \max(\min(\frac{s^T s}{s^T y}, \alpha_{\max}), \alpha_{\min})$
    else
      $\alpha_i \leftarrow \alpha_{\max}$
    end if
  end if

  # Non-monotone curvilinear Armijo line-search
  Initialize $k \leftarrow 0$
  while condition (3) is not satisfied **do**
    $k \leftarrow k + 1$
  end while

  # Proceed to the next iteration
  Set $x_{i+1} = x(\beta^k \alpha_i)$
  Update $i \leftarrow i + 1$
end while

The nonmonotone spectral projected-gradient method (SPG), outlined in Algorithm 1, was introduced by Birgin et al. [6] for problems of the form (1), with $\mathcal{C}$ a closed convex set in $\mathbb{R}^n$, and $f : \mathbb{R}^n \to \mathbb{R}$ a function with continuous partial derivatives on an open set that contains $\mathcal{C}$. The SPG algorithm is based on the curvilinear projected-gradient method, which performs a line search along the curvilinear trajectory (see also [8]):

$$x(\alpha) = \mathcal{P}(x_i - \alpha \nabla f(x_i)), \quad \alpha \geq 0. \tag{2}$$

In order to speed up the method, two important modifications were introduced in [6]. The first modification allows a limited level of nonmonotonicity in the objective value. Given $\mu, \gamma \in (0, 1)$, the Armijo-type line search starts with an initial step length α_i, and then finds the first nonnegative integer k such that

$$f(x(\mu^k\alpha_i)) \leq \max_{0\leq j\leq \min\{i,M-1\}}\{f(x_{i-j})\} + \gamma(\nabla f(x_i))^T(x(\mu^k\alpha_i) - x_i). \tag{3}$$

The right-hand side of this condition ensures sufficient descent, but only with respect to the maximum of up to M of the most recent objective values. In case $M = 1$ this reduces to the standard Armijo line-search condition. The second modification is the use of the spectral step length, as proposed by Barzilai and Borwein [9]. Given $s = x_i - x_{i-1}$ and $y = \nabla f(x_i) - \nabla f(x_{i-1})$, the initial step length at iteration i is defined as

$$\alpha_i = \begin{cases} \max\left(\min\left(\frac{s^Ts}{s^Ty},\ \alpha_{\max}\right),\ \alpha_{\min}\right) & \text{if } s^Ty > 0, \\ \alpha_{\max} & \text{otherwise,} \end{cases}$$

where $0 < \alpha_{\min} < \alpha_{\max}$ are fixed parameters. More information on the motivation behind this particular choice of step length can be found in [6,10]. It was shown in [6] that any accumulation point x^* of the sequence $\{x_i\}$ lies in $\mathcal{C}$ and satisfies

$$\|\mathcal{P}(x^* - \nabla f(x^*)) - x^*\|_2 = 0. \tag{4}$$

Practical implementations of Algorithm 1 may use a relaxed version of (4) along with other conditions as a stopping criterion.

2.2 Limited-memory BFGS

The L-BFGS algorithm by Liu and Nocedal [7] is a popular quasi-Newton method for unconstrained minimization of smooth functions $\hat{f} : \mathbb{R}^n \to \mathbb{R}$:

$$\underset{x}{\text{minimize}} \quad \hat{f}(x).$$

At each iteration, the algorithm constructs a positive-definite approximation H_i of the inverse Hessian of $\hat{f}$ at x_i, based on an initial positive-definite matrix H along with $\hat{n} = \min\{i, N\}$ of the most recent vector pairs $\{s_{i-j}, y_{i-j}\}_{j=0}^{\hat{n}-1}$, where

$$s_j = x_j - x_{j-1}, \quad \text{and} \quad y_j = \nabla\hat{f}(x_j) - \nabla\hat{f}(x_{j-1}), \tag{5}$$

and N denotes the memory size for L-BFGS. The iterates in L-BFGS are of the form $x_{i+1} = x_i + \alpha_i d_i$, where the search direction d_i is given by

$$d_i = -H_i \cdot \nabla\hat{f}(x_i), \tag{6}$$

and the step size α_i is chosen to satisfy the Wolfe conditions:

$$\hat{f}(x_i + \alpha_i d_i) \leq \hat{f}(x_i) + \gamma_1\alpha_i(\nabla\hat{f}(x_i))^T d_i, \tag{7a}$$
$$(\nabla\hat{f}(x_i + \alpha_i d_i))^T d_i \geq \gamma_2(\nabla\hat{f}(x_i))^T d_i. \tag{7b}$$

Parameters γ_1 and γ_2 are chosen such that $0 < \gamma_1 < \frac{1}{2}$, and $\gamma_1 < \gamma_2 < 1$. For details on the structure of the inverse approximation H_i and efficient ways of evaluating the matrix-vector product in (6), see [7,11].

2.2.1 Convergence results

For the analysis of the L-BFGS algorithm, Liu and Nocedal make the following assumptions [7]:

Assumption 1 Given starting point x_0, we have:

(1) The objective function $\hat{f}$ is twice continuously differentiable;
(2) The level set $\mathcal{D} := \{x \in \mathbb{R}^n \mid \hat{f}(x) \leq f(x_0)\}$ is convex; and
(3) There exist positive constants μ_1 and μ_2 such that

$$\mu_1 \|v\|^2 \leq v^T [\nabla^2 \hat{f}(x)]\, v \leq \mu_2 \|v\|^2, \tag{8}$$

for all $x \in \mathcal{D}$ and $v \in \mathbb{R}^n$.

Under these conditions, and with some simplifications, they prove that

Theorem 2 (Liu and Nocedal [7]) *The L-BFGS algorithm generates a sequence* $\{x_i\}$ *that converges to the unique minimizer* x^* *in* $\mathcal{D}$. *Moreover, there exists a constant* $c > 0$ *such that*

$$\hat{f}(x_{i+1}) - \hat{f}(x^*) \leq (1 - c)(\hat{f}(x_i) - \hat{f}(x^*)). \tag{9}$$

3 Proposed algorithm

The proposed algorithm can be seen as a modification of the SPG method that allows the use of quasi-Newton steps over a currently active face. The basic idea is that whenever two successive iterates x_i and x_{i-1} lie on the same face, we can form or update a quadratic model of $\hat{f}$, the objective function restricted to the face. Whenever a model for the current face is available, the algorithm will attempt a quasi-Newton step that is restricted to the face and satisfies the Wolfe conditions (7). If the quasi-Newton step fails, or is otherwise abandoned, the algorithm simply falls back to the SPG method and takes a regular spectral projected-gradient step. After each step—regardless of the type—we check the conditions required to update the quadratic model and initiate the quasi-Newton step:

$$\mathcal{F}(x_i) = \mathcal{F}(x_{i-1}) \quad \text{and} \quad -\nabla f(x_i) \in \text{self proj}(\mathcal{F}_i). \tag{10}$$

If these conditions are not met, we discard the Hessian approximation used in the quadratic model. The self-projection criterion in (10) is essential and omitting it could cause the algorithm to take repeated quasi-Newton iterations converging to a minimum on the relative interior of the face rather than the global minimum.

3.1 Quasi-Newton over a face

One way of performing quasi-Newton over a face is by maintaining an inverse Hessian approximation using the update vectors in (5), and computing the search direction d_i using (6). However, this approach has some major disadvantages. First, we may have that $d_i \notin \text{diff hull}(\mathcal{F}_i)$, which means that $x_i + \alpha d_i \notin \mathcal{F}_i$ for all nonzero α. This could be partially solved by projection onto the face, but such a projected direction is no longer guaranteed to be a descent direction [12]. This too could be addressed by modifying the Hessian, but doing so would further complicate the algorithm. A second disadvantage is that we maintain the inverse Hessian approximation for the higher-dimensional ambient space, and the approximation may therefore not be very accurate along $\text{aff}(\mathcal{F}_i)$.

Algorithm 2 *Outline of the proposed hybrid quasi-Newton projected-gradient method*

Inputs: A, b, τ, x_0
Initialize $H_0 = \emptyset$, $\mathcal{F}_0 = \mathcal{F}(x_0)$, $g_0 = \nabla f(x_0)$
Set $t \leftarrow 0$, choose $\alpha_0 \in [\alpha_{\min}, \alpha_{\max}]$
while not done **do**
  # Quasi-Newton step on current face
  flagUpdated $\leftarrow$ false
  if ($H_t \neq \emptyset$) **then**
    $d_t = -\Phi_t H_k \Phi_t^T g_t$
    Perform Wolfe-type line-search on $x(\gamma) := x + \gamma d$ over $\{\gamma \mid x(\gamma) \in \mathcal{F}_t\}$
    if (line search successful) **then**
      Reset objective function history
      $x_{t+1} = x_t + \gamma_t d_t$
      flagUpdated $\leftarrow$ true
    end if
  end if

  # Projected-gradient step
  if (flagUpdated = false) **then**
    Compute Barzilai-Borwein scaling parameter
    Nonmonotone curvilinear Armijo line-search along $x(\gamma) := \mathcal{P}(x_t - \gamma g_t)$
    $x_{t+1} = x(\gamma_t)$
  end if

  $t \leftarrow t + 1$, $g_t = \nabla f(x_t)$, $\mathcal{F}_t = \mathcal{F}(x_t)$

  # Update the quadratic model of the current face
  if ($\mathcal{F}_t = \mathcal{F}_{t-1}$ and $-g_t \in$ self proj($\mathcal{F}_t$)) **then**
    if ($H_{t-1} = \emptyset$) **then**
      Initialize $H_{t-1} \leftarrow \mu I$
      Determine orthonormal basis $\Phi_t = \Phi(\mathcal{F}_t)$
    else
      Set $\Phi_t = \Phi_{t-1}$
    end if
    Set H_t as L-BFGS update to H_{t-1} using $s_t = \Phi_t^T (x_t - x_{t-1})$ and $y_t = \Phi_t^T (g_t - g_{t-1})$
  else
    $H_t = \emptyset$
  end if
end while

The solution of the above problems is straightforward: we simply work with a representation for the function restricted to $\text{aff}(\mathcal{F}_i)$. Let $\mathcal{F}_i$ be a k-dimensional face with $k > 0$. Then we can find an orthonormal basis $\Phi := \Phi(\mathcal{F}_i) \in \mathbb{R}^{n\times k}$ whose span coincides with $\text{diff hull}(\mathcal{F}_i)$. Using Φ we can write any point $x \in \mathcal{F}_i$ as $x = v + \Phi c$, where $v \in \mathbb{R}^n$ is an arbitrary but fixed point in $\mathcal{F}_i$, and $c \in \mathbb{R}^k$ is a coefficient in the lower-dimensional space. The function $\hat{f} : \mathbb{R}^k \to \mathbb{R}$, which restricts f to the current face, is then given by $\hat{f}(c) = f(v + \Phi c)$. The idea then is to form the inverse Hessian approximation over $\hat{f}$, and use it to obtain a search direction $\hat{d} \in \mathbb{R}^k$, which can then be mapped back to the ambient space for the actual line search. In particular, we can form the approximate inverse Hessian $H_i \in \mathbb{R}^{k\times k}$ by updating an initial positive definite H using

$$s_i = \Phi^T(x_i - x_{i-1}) \quad \text{and} \quad y_i = \Phi^T(\nabla f(x_i) - \nabla f(x_{i-1})). \tag{11}$$

In order to obtain the search direction we first compute $\nabla \hat{f}(\Phi^T(x_i - v)) = \Phi^T \nabla f(x_i)$ by projecting the gradient $\nabla f(x_i)$ onto the lower-dimensional space. We then apply the inverse Hessian followed by back projection, giving:

$$d_i = \Phi H_i \Phi^T(-\nabla f(x_i)).$$

The resulting vector clearly lies in $\text{diff hull}(\mathcal{F}_i)$ and we therefore have that $x_i + \alpha d_i \in \mathcal{F}_i$ for all step sizes $\alpha \in [0, \alpha_{\text{bnd}}]$, for some $\alpha_{\text{bnd}} > 0$, possibly with $\alpha_{\text{bnd}} = +\infty$. Line search is equivalently done in the projected or ambient dimension, and we aim to find a step size α within the above range, such that the Wolfe conditions (7) are met. For the line search we could start with a unit step length, whenever $\alpha_{\text{bnd}} \geq 1$, or we could try $\alpha = \alpha_{\text{bnd}}$ first to encourage exploring lower-dimensional faces, provided of course that $\alpha_{\text{bnd}} < \infty$. If no suitable step length can be found, or a certain maximum number of trial steps is taken, we abandon the quasi-Newton step and take a spectral projected-gradient step instead. As a final remark, note that condition (10) should never be met for vertices, since that would imply not only that $x_i = x_{i-1}$, but also that $-\nabla f(x_{i-1}) \in \text{self proj}(\mathcal{F}_{i-1}) = \mathcal{N}(\mathcal{F}_{i-1})$, which means that the optimality condition given in (4) would already have been satisfied at x_{i-1}.

3.2 Convergence

For the convergence analysis of Algorithm 2 we rely on the results in [6] and [7], and add a step in the algorithm that resets the objective-value history used by SPG after each series of successful quasi-Newton iterations to ensure that any subsequent iteration has a lower objective value. We use the following assumptions, which are somewhat more restrictive than those in the aforementioned two papers (see for example Assumption 1):

Assumption 3 We assume that

(1) Function f is twice continuously differentiable, and bounded below;

(2) There exist constants $0 < \mu_1 \leq \mu_2 < \infty$ such that for all $x, v \in \mathbb{R}^n$:

$$\mu_1 \|v\|_2^2 \leq v^T \nabla^2 f(x) v \leq \mu_2 \|v\|_2^2. \tag{12}$$

Under these assumptions, which imply convexity of f, we have the following result:

Theorem 4 *Let $f(x)$ satisfy Assumption 3 and let $x_0 \in \mathcal{C}$. Then the sequence $\{x_t\}$ generated by Algorithm 2 converges to the unique minimizer of (1).*

Proof Assumption 3 ensures the existence of a unique minimizer x^* to (1), which satisfies

$$-\nabla f(x^*) \in \mathcal{N}(x^*).$$

If there exists a finite t for which $x_t = x^*$, we are done. Suppose, therefore that $x_t \neq x^*$ for all t. We consider two cases. First, if there are finitely many quasi-Newton steps, there must a $\bar{t}$ such that all iterations $t > \bar{t}$ are of the projected gradient type. In this case the result follows directly from the analysis in [6]. Second, consider the case where there are infinitely many quasi-Newton steps. For a fixed face $\mathcal{F}$, minimizing $\hat{f}$ is equivalent to minimizing the objective over the affine hull of the current face $\mathcal{F}$:

$$\underset{x}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad x \in \text{aff}(\mathcal{F}). \tag{13}$$

For any successful step it follows from Theorem 2 that there exists a constant $c > 0$ such that the quasi-Newton step satisfies

$$f(x_{t+1}) - f(x_{\mathcal{F}}^*) \leq (1 - c)(f(x_t) - f(x_{\mathcal{F}}^*)), \tag{14}$$

where $x_{\mathcal{F}}^*$ denotes the minimizer of (13). Because the history of the M most recent objective values is reset after each successful quasi-Newton step, any intermediate projected-gradient step will not increase the objective. Based on this, Lemma 1 below, shows that the number of quasi-Newton iterates on any $\mathcal{F}$ that does not contain x^* is finite. By polyhedrality of the domain, the number of faces is bounded, and we must therefore take infinitely many iterations on at least one face that contains x^*. Repeated application of (14) then shows that the objective value converges to $f(x_{\mathcal{F}}^*) = f(x^*)$. From Assumption 3 it then follows that $\{x_t\}$ converges to x^*. □

Lemma 1 *Let $\mathcal{F}$ be a face of $\mathcal{C}$ such that $x^* \notin \mathcal{F}$. Then the number of quasi-Newton steps on $\mathcal{F}$ taken by Algorithm 2 is finite.*

Proof The quasi-Newton method is applied to $\hat{f}$, which is equivalent to f restricted to $\mathcal{F}$. We therefore focus on (13). Let $x_{\mathcal{F}}^*$ be the solution to (13), and denote by $x_{[j]}$ and $x_{[j]+1}$ the starting, respectively ending, point for the jth quasi-Newton step on $\mathcal{F}$. It follows from Theorem 4 that

$$f(x_{[j]}) - f(x_{\mathcal{F}}^*) \leq f(x_{[j-1]+1}) - f(x_{\mathcal{F}}^*) \leq (1 - c)(f(x_{[j-1]}) - f(x_{\mathcal{F}}^*)), \tag{15}$$

for $j \geq 2$. This holds since any intermediate quasi-Newton iteration can only reduce the objective, and likewise for projected-gradient steps, as a consequence of resetting the function-value history.

We consider two cases. First assume that $x^*_{\mathcal{F}} \notin \mathcal{F}$. Let $\bar{f}$ be the minimum of $f(x)$ over $x \in \mathcal{F}$. Repeated application of (15) gives

$$f(x_{[j]+1}) - f(x^*_{\mathcal{F}}) \leq (1-c)^j (f(x_{[1]}) - f(x^*_{\mathcal{F}})). \tag{16}$$

For sufficiently large, but finite j, the right-hand side in (16) must fall below $\bar{f} - f(x^*_{\mathcal{F}})$, which is strictly positive. Since every successful quasi-Newton step results in a vector $x_{[j]+1} \in \mathcal{F}$ by construction, it follows that the number of quasi-Newton iterates on $\mathcal{F}$ must be bounded.

For the second case, assume that $x^*_{\mathcal{F}} \in \mathcal{F}$. Because optimization is done over $\text{aff}(\mathcal{F})$, it holds that $-\nabla f(x^*_{\mathcal{F}}) \perp \text{diff hull}(\mathcal{F})$. For $-\nabla f(x^*_{\mathcal{F}}) \in \text{self proj}(\mathcal{F})$, we must therefore have $-\nabla f(x^*_{\mathcal{F}}) \in \mathcal{N}(x^*_{\mathcal{F}})$, but this cannot be the case since it would imply that $x^*_{\mathcal{F}}$ is a global minimizer. (The same holds when $x^*_{\mathcal{F}}$ lies on a lower-dimensional subface on the boundary of $\mathcal{F}$.) Since f is continuously differentiable by assumption, it follows that $-\nabla f(x) \notin \text{self proj}(\mathcal{F})$ for all points $x \in \mathcal{F}$ sufficiently close to $x^*_{\mathcal{F}}$. Assumption 3 then allows us to define a sufficiently close neighborhood as the level set $f(x) \leq \bar{f}$ over $x \in \mathcal{F}$, where $\bar{f} > f(x^*_{\mathcal{F}})$. Applying the same argument we used above shows that the right-hand side of (16) again falls below $\bar{f} - f(x^*_{\mathcal{F}})$ for sufficiently large j. Once this happens all following iterates $x_t \in \mathcal{F}$ must have $f(x_t) \leq \bar{f}$. Since the self-projection cone condition $-\nabla f(x) \in \text{self proj}(\mathcal{F})$ does not hold at these points, no more quasi-Newton steps are taken on $\mathcal{F}$. □

A similar analysis holds when the spectral projected-gradient method is replaced by another convergent algorithm, provided that the iterates do not exceed the initial objective value.

4 Application to Lasso

The proposed algorithm depends on a number of operations on the constraint set. In particular, it has to determine the face in which the current iterate lies, check membership of the self-projection cone, and determine an orthonormal basis for the current face. For the algorithm to be of practical use, the constraint set must therefore be simple enough to allow efficient evaluation of these operations. As this work was motivated by improving the Lasso problem, we focus on the weighted one-norm ball (which for unit weights is also known as the cross-polytope or n-octahedron [13]):

$$\mathcal{C}_{w,1} = \{x \in \mathbb{R}^n \mid \|x\|_{w,1} \leq \tau\},$$

where $\|x\|_{w,1} := \sum_i w_i |x_i|$ positive w_i. The proposed framework also applies naturally to bound or simplex constrained problems, but these are outside the scope of this paper. The objective function we consider throughout this section is

$$f(x) = \tfrac{1}{2}\|Ax - b\|_2^2 + \tfrac{\mu}{2}\|x\|_2^2 + c^T x, \tag{17}$$

which can also be written in the form $\frac{1}{2}\|Ax - b\|_2^2 + c^T x$, with A and b appropriately redefined. The benefit of having an objective function of the form (17) is that it permits closed-form expressions for step lengths satisfying certain conditions. In the remainder of this section we discuss practical considerations for the line-search conditions and look at the specific structure and properties of the set $\mathcal{C}_{w,1}$.

4.1 Line search

For most objective functions the line search is done by evaluating $f(\mathcal{P}(x + \alpha d))$ or $f(x + \alpha d)$ for a series of α values until all required conditions, such as Armijo and Wolfe, are satisfied. The objective function in (17) has closed-form solutions for some of the problems arising in the line search, thereby allowing us to simplify the algorithms and improve their performance.

4.1.1 Optimal unconstrained step size

As a start we look at the step length that minimizes the objective along $f(x + \alpha d)$:

$$\alpha_{\text{opt}} = \arg\min_{\alpha} \quad f(x + \alpha d)$$

Differentiating f with respect to α and equating to zero leads to the following expression:

$$\alpha_{\text{opt}} = -\frac{(A^T r + \mu x + c)^T d}{\|Ad\|_2^2 + \mu\|d\|_2^2},$$

with $r = Ax - b$. When $\mu = 0$ and $c = 0$ this reduces to $\alpha_{\text{opt}} = -r^T Ad/\|Ad\|_2^2$.

4.1.2 Wolfe line search conditions

The maximum step length for which the Armijo condition (7a) is satisfied can be found by expanding the terms and simplifying. Doing so gives the following bound:

$$\alpha \leq \alpha_{\max} = 2(1 - \gamma_1)\alpha_{\text{opt}}.$$

Likewise, the gradient condition (7b) reduces to

$$\alpha \geq \alpha_{\min} = (1 - \gamma_2)\alpha_{\text{opt}}.$$

4.1.3 Projection arc

Line search in gradient projection methods is often done by backtracking from a single projection $p(\alpha) = x + \alpha(\mathcal{P}_{\mathcal{C}}(x + d))$, with $\alpha \in [0, 1]$, or by search over the projection arc $p(\alpha) = \mathcal{P}_{\mathcal{C}}(x + \alpha d)$, with $\alpha \geq 0$. The trajectory of the first method depends strongly on the scaling of d and is more likely to generate points on the interior of the feasible set. The second method better captures the structure of the domain, but can also be more expensive computationally. In an earlier version of the present work [14] we studied line search based on the entire projection arc (that is, over all values $\alpha \geq 0$) along with its properties. Given the high computational complexity of the line search in practice and the lengthy treatise we omit the details here and refer the interested reader to [14].

4.2 Properties of the weighted one-norm ball

4.2.1 Facial structure

The weighted one-norm ball of radius τ is the convex hull of vertices $\{\pm\tau/w_i \cdot e_i\}_i$. Every proper k-face $\mathcal{F}$ of the weighted one-norm ball $\mathcal{C}_{w,1}$ can be written as the convex hull of $\{\sigma_i/w_i \cdot e_i\}_{i\in\mathcal{I}}$, where $\mathcal{I}$ is a subset of $\{1, \ldots, n\}$ with cardinality $k + 1$, and $\sigma_i \in \{-1, +1\}$. Throughout this section we assume that $\tau > 0$.

Given an $x \in \mathcal{C}$ we can determine $\mathcal{F}(x)$ as follows. First, we need to check whether $\|x\|_{w,1} < \tau$, in which case $\mathcal{F}(x) = \mathcal{C}$. Otherwise, x lies on a proper face, which can be uniquely characterized by the sign vector $\operatorname{sgn}(x)$ whose ith entry is given by $\operatorname{sgn}(x_i)$. Determining $\mathcal{F}(x)$ and checking equality of faces can therefore be done in $\mathcal{O}(n)$ time.

4.2.2 Projection onto the feasible set

Projection onto the weighted one-norm ball is discussed in [15] and is based on the solution of the prox function

$$x_\lambda(u) = \operatorname{prox}_{\lambda\|\cdot\|_{w,1}}(u) := \arg\min_x \ \tfrac{1}{2}\|x - u\|_2^2 + \lambda\|x\|_{w,1} = \operatorname{sign}(u) \cdot [|u| - \lambda w]_+ , \tag{18}$$

where $[\cdot]_+ = \max(0, \cdot)$,the absolute value, sign function, and multiplication in the right-hand side are evaluated elementwise. Projection onto $\mathcal{C}_{w,1}$ then amounts to finding the smallest $\lambda \geq 0$ for which $\|x_\lambda(u)\|_{w,1} \leq \tau$. The entries in $x_\lambda(u)$, and therefore $\|x_\lambda(u)\|_{w,1}$, are continuous and piecewise linear in λ with break points occurring at $\lambda = |u_i|/w_i$. We can obtain an $\mathcal{O}(n \log n)$ algorithm that finds the optimal λ and subsequent projection by sorting the break points [15]. This can be reduced to an expected $\mathcal{O}(n)$ algorithm [16] by avoiding the explicit sorting step.

4.2.3 Self-projection cone of a face

Given $x \in \mathcal{C}_{w,1}$ and search direction $d \in \mathbb{R}^n$, we want to know if $d \in \text{self proj}(\mathcal{F}(x))$. When $\|x\|_{w,1} < \tau$ it follows that x lies in the interior of $\mathcal{C}_{w,1}$ meaning that $\mathcal{F}(x) = \mathcal{C}_{w,1}$ and $d \in \text{self proj}(\mathcal{C}_{w,1}) = \mathbb{R}^n$, trivially. For the case where $\|x\|_{w,1} = \tau$, consider the support $\mathcal{I} = \{i \mid x_i \neq 0\}$.

We first show that $\mathcal{I}$ is included in the support of $x(\alpha) = \mathcal{P}(x + \alpha d)$ for all $\alpha \in [0, \bar{\alpha})$, for some $\bar{\alpha} > 0$. Since

$$\|x + \alpha d\|_{w,1} \leq \|x\|_{w,1} + \alpha \|d\|_{w,1} = \tau + \alpha \|d\|_{w,1}, \tag{19}$$

it follows that the projection threshold $\lambda(\alpha) \leq \alpha \|d\|_{w,1} / \min_i\{w_i\} =: \nu\alpha$. At the same time, a necessary condition for $i \in \mathcal{I}$ to be removed from the support is that $\lambda(\alpha) \geq (|x_i| - \alpha|d_i|)/w_i$. Combined with (19) this gives the necessary condition $(\nu + |d_i|/w_i)\alpha \geq |x_i|/w_i$, which allows us to choose

$$\bar{\alpha} = \min_{i \in \mathcal{I}} \left\{ |x_i|/(w_i \nu + |d_i|) \right\} > 0.$$

For d to be in the self-projection cone we therefore need to show that (1) $x + \alpha d$ does not move into the polytope, and (2) the support does not increase. It can be verified that the first condition is satisfied if and only if the directional derivative $(\|\cdot\|_{w,1})'(x; d)$ of $\|x\|_{w,1}$ in direction d is nonnegative:

$$\sum_{i \in \mathcal{I}} \text{sign}(x_i) d_i w_i + \sum_{i \notin \mathcal{I}} |d_i| w_i \geq 0. \tag{20}$$

For the second condition to be satisfied we require for all $i \notin \mathcal{I}$ and sufficiently small α that the absolute value of the entries remains less than or equal to the threshold value, namely $\alpha|d_i| \leq w_i \lambda(\alpha)$. When the support remains the same we find $\lambda(\alpha)$ by solving

$$\sum_{i \in \mathcal{I}} w_i (|x_i + \alpha d_i| - w_i \lambda(\alpha)) = \tau, \quad \text{which gives} \quad \lambda(\alpha) = \alpha \cdot \frac{\sum_{i \in \mathcal{I}} w_i \, \text{sign}(x_i) d_i}{\sum_{i \in \mathcal{I}} w_i^2},$$

after writing $|x_i + \alpha d_i| = |x_i| + \alpha \, \text{sign}(x_i) d_i$ and recalling that $\|x\|_{w,1} = \tau$. A necessary (and sufficient) condition for the support to remain the same is therefore that

$$\max_{i \notin \mathcal{I}} |d_i|/w_i \leq \frac{\sum_{i \in \mathcal{I}} w_i \, \text{sign}(x_i) d_i}{\sum_{i \in \mathcal{I}} w_i^2}. \tag{21}$$

Summarizing the above we have:

Theorem 5 *Given $x \in \mathcal{C}_{w,1}$ with support $\mathcal{I} = \{i \mid x_i \neq 0\}$, then $d \in$ self proj$(\mathcal{F}(x))$ if and only if $\|x\|_{w,1} < \tau$, or $\|x\|_{w,1} = \tau$ and*

$$\sum_{i\in\mathcal{I}} \operatorname{sign}(x_i) d_i w_i + \sum_{i\notin\mathcal{I}} |d_i| w_i \geq 0, \quad \text{and} \quad \max_{i\notin\mathcal{I}} |d_i|/w_i \leq \frac{\sum_{i\in\mathcal{I}} w_i \operatorname{sign}(x_i) d_i}{\sum_{i\in\mathcal{I}} w_i^2}.$$

4.2.4 Orthogonal basis for a face

For the construction of a quadratic approximation of objective function f over a face $\mathcal{F}$, we require an orthogonal basis Φ for diff hull$(\mathcal{F})$. For simplicity, consider the facet of the unit cross polytope lying in the positive orthant in $\mathbb{R}^3$. In other words, consider the unit simplex given by conv$\{e_1, e_2, e_3\}$. A first vector for the basis can then be obtained by normalizing $e_2 - e_1$ to have unit norm. A second vector orthogonal to the first can be obtained by connecting the point halfway on the line segment e_1–e_2 to e_3, that is, $e_3 - (e_1 + e_2)/2$, followed again by normalization. Extending this to general k-simplices we find $(k + 1) \times k$ bases $Q = [q_1, q_2, \ldots, q_k]$ with

$$q_j = \left(e_{j+1} - \frac{1}{j} \sum_{i=1}^{j} e_i \right) \Big/ \sqrt{1 + 1/j}.$$

In other words

$$Q_{i,j} = \begin{cases} -\sqrt{1/(j^2 + j)} & i \leq j \\ \sqrt{j/(j + 1)} & i = j + 1 \\ 0 & \text{otherwise.} \end{cases}$$

It can be seen that the above procedure amounts to taking a QR factorization of the $k + 1 \times k$ matrix $[-e,\ I]^T$ of differences between the first vertex and all others of the k-simplex, and discarding the last column in Q whose entries are all equal to $1/\sqrt{n}$. The special structure of Q allows us to evaluate matrix-vector products with Q and its transpose in $\mathcal{O}(k)$ time, without having to form the matrix explicitly. For the general case, let $\mathcal{F} = \mathcal{F}(x)$. For the case where $\mathcal{F} = \mathcal{C}$ no projection is needed and we can simply choose $\Phi = I$. Otherwise, let $\mathcal{I} = \{i \mid x_i \neq 0\}$ denote the support of x. The desired basis can then be obtained by first restricting the vector to its support and then normalizing the sign pattern, thus giving:

$$\Phi = I_{\mathcal{I}} \cdot \operatorname{diag}(\operatorname{sgn}(x_{\mathcal{I}})) \cdot Q.$$

Matrix-vector products with Φ can be evaluated in $\mathcal{O}(n)$ time, again without forming the matrix.

Generic weighted one-norm ball For the weighted one-norm ball we consider a simplicial face given by conv$(w_0 e_1, w_1 e_2, \ldots, w_n e_n)$, with nonzero weights w_0 to w_n. (Throughout this paragraph it is more convenient to work with a weight vector whose

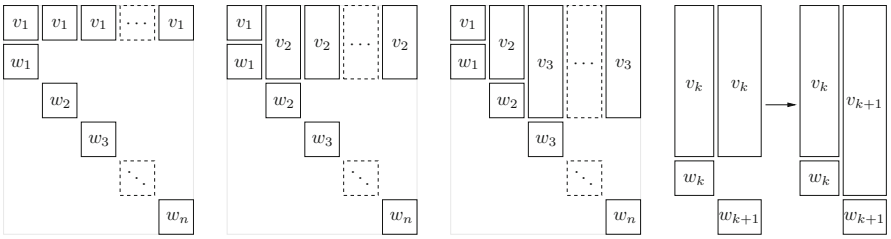

Fig. 2 Stages of the orthogonalization process

elements are the inverse of the weights appearing in the weighted one norm; the actual vertices of the weighted one norm ball are $\pm w_i^{-1} e_i$, not $\pm w_i e_i$.) The orthonormal basis corresponding to the face can again be found by applying the standard QR-algorithm to the matrix of differences between the vertices, and taking advantage of the special structure of the matrix. The initial setup is illustrated in Fig. 2a with $v_1 = -w_0$. The two operations in this process are projecting out the contributions of all subsequent columns and normalizing the columns to unit norm. We do not normalize until the very end but do keep track of the squared two norm of the completed columns. Given vectors a and b we obtain the component in b orthogonal to a by evaluating $b - \frac{\langle a,b\rangle}{\langle a,a\rangle} a$. In the first step of the factorization (we are interested only in Q) we orthogonalize with respect to the first column a. The inner product of each column with a is identical and equal to $\alpha_1 = \langle v_1, v_1\rangle = \|v_1\|_2^2$. Using this we also compute the squared two norm of the first column as $\gamma_1 = \alpha_1 + w_1^2$. After the sweep with the first column we are left with the matrix shown in Fig. 2b where

$$v_2 = \begin{bmatrix} v_1 - \frac{\alpha_1}{\gamma_1} v_1 \\ -\frac{\alpha_1}{\gamma_1} w_1 \end{bmatrix} = \begin{bmatrix} \frac{\gamma_1 - \alpha_1}{\gamma_1} v_1 \\ -\frac{\alpha_1}{\gamma_1} w_1 \end{bmatrix} = \begin{bmatrix} \frac{w_1^2}{\gamma_1} v_1 \\ -\frac{\alpha_1}{\gamma_1} w_1 \end{bmatrix}.$$

The next step is to sweep with the updated second column. For this we compute the inner product with the remaining columns and itself, yielding $\alpha_2 = \|v_2\|_2^2$ and $\gamma_2 = \alpha_2 + w_2^2$, respectively. After this sweep we arrive at the matrix given in Fig. 2c. Proceeding like this we successively sweep with each of the column until we are done. Consider now the sweep with some column k, illustrated in Fig. 2d. Given $\alpha_k = \|v_k\|_2^2$ and $\gamma_k = \alpha_k + w_k^2$ we find

$$v_{k+1} = \begin{bmatrix} v_k - \frac{\alpha_k}{\gamma_k} v_k \\ -\frac{\alpha_k}{\gamma_k} w_k \end{bmatrix} = \begin{bmatrix} \frac{w_k^2}{\gamma_k} v_k \\ -\frac{\alpha_k}{\gamma_k} w_k \end{bmatrix},$$

from which we derive recurrence relations $\gamma_{k+1} = \alpha_{k+1} + w_{k+1}^2$ and

$$\begin{aligned}\alpha_{k+1} &= \left(\frac{w_k^2}{\gamma_k}\right)^2 \cdot \|v_k\|_2^2 + \left(\frac{\alpha_k}{\gamma_k}\right)^2 \cdot w_k^2 = \frac{w_k^2 w_k^2}{\gamma_k^2}\alpha_k + \frac{\alpha_k^2}{\gamma_k^2} w_k^2 \\ &= \alpha_k w_k^2 \frac{\alpha_k + w_k^2}{\gamma_k^2} = \frac{\alpha_k w_k^2}{\gamma_k}. \end{aligned} \tag{22}$$

With α_1 and γ_1 as given above, this allows us to compute all α and γ values. Ultimately we are interested in the final orthonormal Q matrix. Defining scaling factors

$$\mu_{i,j} = \prod_{k=i}^{j} \frac{w_k^2}{\gamma_k} \quad \text{for } 1 \le i \le j \le n, \tag{23}$$

as well as factors $u_i = -\alpha_i/\gamma_i$ for $1 \le i \le n$ and $u_0 := -1$, it can be found based on the structure of vectors v that

$$Q = \operatorname{diag}(w) \cdot \begin{bmatrix} u_0 & \mu_{1,1}u_0 & \mu_{1,2}u_0 & \mu_{1,3}u_0 & \cdots & \mu_{1,n-1}u_0 \\ 1 & u_1 & \mu_{2,2}u_1 & \mu_{2,3}u_1 & \cdots & \mu_{2,n-1}u_1 \\ & 1 & u_2 & \mu_{3,3}u_2 & \cdots & \mu_{3,n-1}u_2 \\ & & \ddots & \ddots & \ddots & \vdots \\ & & & 1 & u_{n-2} & \mu_{n-1,n-1}u_{n-2} \\ & & & & 1 & u_{n-1} \\ & & & & & 1 \end{bmatrix} \cdot \operatorname{diag}(1/\sqrt{\gamma}).$$

Multiplication with this matrix and its transpose may still seem expensive but we now show how the structure enables $\mathcal{O}(n)$ algorithms for both operations. Multiplication with the diagonal matrices is trivial so we focus only on multiplication with the central part of the matrix. Looking at a small example we can decompose this matrix as

$$\begin{bmatrix} u_0 & \mu_{1,1}u_0 & \mu_{1,2}u_0 \\ 1 & u_1 & \mu_{2,2}u_1 \\ & 1 & u_2 \\ & & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix} + \operatorname{diag}(u) \begin{bmatrix} 1 & \mu_{1,1} & \mu_{1,2} \\ & 1 & \mu_{2,2} \\ & & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix} + \operatorname{diag}(u) \begin{bmatrix} M \\ 0 \end{bmatrix}.$$

The key part is multiplication with the last matrix M. To evaluate $y = Mv$ we initialize $y_3 = v_3$ and then work upwards. Direct evaluation gives $y_2 = v_2 + \mu_{2,2}v_3$, which can be rewritten as $y_2 = v_2 + \mu_{2,2}y_3$. A pattern emerges when looking at the computation of y_1:

$$\begin{aligned} y_1 &= v_1 + \mu_{1,1}v_2 + \mu_{1,2}v_3 \\ &= v_1 + \mu_{1,1}(v_2 + \mu_{2,2}v_3) \\ &= v_1 + \mu_{1,1}y_2, \end{aligned}$$

where $\mu_{1,2} = \mu_{1,1}\mu_{2,2}$, or more generally $\mu_{i,k} = \mu_{i,j}\mu_{j+1,k}$ for $i \le j \le k$, follows from the definition of μ in (23). Given $y_n = v_n$, we therefore obtain the recurrence $y_k = v_k + \mu_{k,k} y_{k+1}$, which allows us to evaluate $y = Mv$ in linear time. With v appropriately redefined we now look at $y = M^T v$:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & & \\ \mu_{1,1} & 1 & \\ \mu_{1,2} & \mu_{2,2} & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}.$$

Starting with $y_1 = v_1$ we find $y_2 = \mu_{1,1} y_1 + v_2$ and $y_3 = \mu_{2,2} y_2 + v_3$, using $\mu_{1,2} = \mu_{1,1}\mu_{2,2}$. This gives the recurrence $y_{k+1} = \mu_{k,k} y_k + v_{k+1}$. We summarize the initialization and multiplication with Q and Q^T in Algorithm 3. Note that these algorithms use a different indexing scheme for a convenient implementation. For practical implementations we can precompute and store $1/\sqrt{\gamma_k}$ instead of γ_k and avoid storing α since it is not used during the evaluation of matrix-vector products. Alternatively, we can reduce the memory footprint at the cost of increased computation by storing only α and re-computing μ_k, u_k, and γ_k whenever they are needed.

Algorithm 3 *Implicit construction of Q and matrix-vector multiplication with Q and Q^T*

(**a**) Precomputing α, γ, u, and μ

Input: Weights $w = [w_1, \ldots, w_n]$
Initialize $\alpha_1 = w_1^2$, $u_1 = -1$
for $k = 1$ to $n-1$ **do**
  $\gamma_k = \alpha_k + w_{k+1}^2$
  $\mu_k = w_{k+1}^2/\gamma_k$
  $u_{k+1} = -\alpha_k/\gamma_k$
  $\alpha_{k+1} = \alpha_k \mu_k$
end for

(**b**) Evaluating $y = Qv$

Input: γ, u, w, μ; $v \in \mathbb{R}^{n-1}$
$s \leftarrow v_{n-1}/\sqrt{\gamma_{n-1}}$, $t \leftarrow 0$
$y_n = w_n s$
for $k = n-1$ down to 2 **do**
  $t \leftarrow \mu_k t + s$
  $s \leftarrow v_{k-1}/\sqrt{\gamma_{k-1}}$
  $y_k = w_k(u_k t + s)$
end for
$y_1 = w_1 u_1 (\mu_1 t + s)$

(**c**) Evaluating $y = Q^T v$

Input: γ, u, w, μ; $v \in \mathbb{R}^n$
$t \leftarrow w_1 v_1 u_1$
$s \leftarrow w_2 v_2$
$y_1 = (t+s)/\sqrt{\gamma_1}$
for $k = 2$ to $n-1$ **do**
  $t \leftarrow \mu_{k-1} t + u_k s$
  $s \leftarrow w_{k+1} v_{k+1}$
  $y_k = (t+s)/\sqrt{\gamma_k}$
end for

4.2.5 Maximum step length along a face

Given a feasible search direction d it is useful to know the maximum α for which $x + \alpha d \in \mathcal{C}_{w,1}$. When x lies in the interior of $\mathcal{C}_{w,1}$ or when (20) is violated and $x + \alpha d$ moves into the interior, we need to compute the first intersection with the boundary. When x lies on a proper face of $\mathcal{C}_{w,1}$ and d moves along the face or onto a higher dimensional face, the maximum step length is determined by the first element to reach zero. More details can be found in [14].

4.3 Stopping criteria

We now look at stopping criteria for optimizing $f(x)$ as defined in (17) over the weighted one-norm ball. A common stopping criterion for problem of this type is to look at the relative norm of the projected gradient:

$$\rho(x) := \frac{\|\mathcal{P}_{\mathcal{C}}(x - \nabla f(x)) - x\|_2}{\max\{1, \|\nabla f(x)\|\}},$$

which is zero if and only if x is optimal. In addition to this we can look at the relative duality gap, which we define as the difference δ between $f(x)$ and any dual feasible objective, divided by $\max\{1, f(x)\}$. Given that the dual feasible point will typically not be optimal, the relative duality gap merely provides an upper bound on the actual gap. Improving the estimate of the duality gap can cause the iterates to satisfy the stopping criteria earlier and thereby help reduce the runtime.

For the derivation of the dual problem we follow [2,15] and rewrite the original problem as:

$$\underset{x,r}{\text{minimize}} \quad \tfrac{1}{2}\|r\|_2^2 + c^T x + \tfrac{\mu}{2}\|x\|_2^2 \quad \text{subject to} \quad Ax + r - b = 0, \ \|x\|_{w,1} \le \tau.$$

The dual of this problem is given by

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

where the Lagrange dual function $\mathcal{L}$ is given by

$$\begin{aligned}
\mathcal{L}(y,\lambda) &:= \inf_{x,r} \left\{ \tfrac{1}{2}\|r\|_2^2 + c^T x + \tfrac{\mu}{2}\|x\|_2^2 - y^T(Ax + r - b) + \lambda(\|x\|_{w,1} - \tau) \right\} \\
&= y^T b - \tau\lambda + \inf_r \left\{ \tfrac{1}{2}\|r\|_2^2 - y^T r \right\} + \inf_x \left\{ (c - A^T y)^T x + \tfrac{\mu}{2}\|x\|_2^2 + \lambda\|x\|_{w,1} \right\} \\
&= y^T b - \tau\lambda - \tfrac{1}{2}\|y\|_2^2 + \inf_x \left\{ (c - A^T y)^T x + \tfrac{\mu}{2}\|x\|_2^2 + \lambda\|x\|_{w,1} \right\}. \qquad (24)
\end{aligned}$$

Here, the infimum over r is solved by equating the gradient to zero, giving $y = r$ and $y^T r = \|y\|_2^2$. For the infimum over x we consider two cases, based on the value of μ.

Dual when $\mu = 0$. Following the derivation given in [15], with minor changes to account for the c term, it can be shown that

$$\inf_x \{(c - A^T y)^T x + \lambda\|x\|_{w,1}\} = \begin{cases} 0 & \|A^T y - c\|_{\frac{1}{w},\infty} \le \lambda \\ -\infty & \text{otherwise.} \end{cases}$$

From this we then obtain the dual problem:

$$\underset{y,\ \lambda \ge 0}{\text{maximize}} \quad y^T b - \tau\lambda - \tfrac{1}{2}\|y\|_2^2 \quad \text{subject to} \quad \|A^T y - c\|_{\frac{1}{w},\infty} \le \lambda. \qquad (25)$$

As a dual-feasible point we can choose $y = r$. For any given y it can be verified that choosing $\lambda = \|A^T y - c\|_{\frac{1}{w},\infty}$ always gives the largest dual objective value. Given x and the corresponding residual $r = b - Ax$ we therefore obtain the following duality gap:

$$\delta = \|r\|_2^2 + c^T x - r^T b + \tau\|A^T r - c\|_{\frac{1}{w},\infty}$$

Dual when $\mu > 0$. The simplest way of dealing with $\mu > 0$ is to rewrite the problem as:

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|\tilde{A}x - \tilde{b}\|_2^2 \quad \text{subject to} \quad \|x\|_{w,1} \leq \tau$$

with $\tilde{A} = [A;\ \sqrt{\mu}I]$, and $\tilde{b} = [b;\ 0]$. This reduces the problem to the form where $\mu = 0$ and we can therefore directly use the dual formulation (25). Choosing $y = \tilde{r}$, with $\tilde{r} = [r;\ -\sqrt{\mu}x]$, and applying the same derivation as given above, we obtain a dual objective value of

$$r^T b - \tau\lambda - \tfrac{1}{2}\|r\|_2^2 - \tfrac{\mu}{2}\|x\|_2^2, \quad \text{with} \quad \lambda = \|A^T r - \mu x - c\|_{\frac{1}{w},\infty}, \tag{26}$$

and a corresponding duality gap of

$$\delta = \|r\|_2^2 + c^T x - r^T b + \mu\|x\|_2^2 + \tau\|A^T r - \mu x - c\|_{\frac{1}{w},\infty}. \tag{27}$$

Another approach is to solve the original infimum over x in (24) for the case where $\mu > 0$. For a fixed y and λ we have

$$\begin{aligned} m(y,\lambda) &:= \inf_x \left\{ (c - A^T y)^T x + \tfrac{\mu}{2}\|x\|_2^2 + \lambda\|x\|_{w,1} \right\} \\ &= \mu \inf_x \left\{ -\tfrac{1}{\mu}(A^T y - c)^T x + \tfrac{1}{2}\|x\|_2^2 + \tfrac{\lambda}{\mu}\|x\|_{w,1} \right\} \end{aligned} \tag{28}$$

When $\lambda = 0$ it is easily seen that $x^* = \frac{1}{\mu}(A^T y - c)$, thus giving $m(x) = -\frac{1}{2\mu}\|A^T y - c\|_2^2$. For the more general case where $\lambda > 0$, we first reformulate (28) as

$$m(y,\lambda) = \mu \inf_x \left\{ -v^T x + \tfrac{1}{2}\|x\|_2^2 + h(x) \right\}. \tag{29}$$

with $h(x) = \frac{\lambda}{\mu}\|x\|_{w,1} = \|x\|_{\frac{\lambda w}{\mu},1}$ and $v = \frac{1}{\mu}(A^T y - c)$. For problems of the form (29) we have:

Theorem 6 *Let* $h(\cdot)$ *be any norm then*

$$\inf_x \ -v^T x + \tfrac{1}{2}\|x\|_2^2 + h(x) = -\tfrac{1}{2}\|\text{prox}_h(v)\|_2^2$$

Proof Note that the objective is coercive and therefore attains the minimum. This allows us to rewrite and solve the objective as follows:

$$u = \arg\min_x \ \tfrac{1}{2}\|x - v\|_2^2 + h(x) = \text{prox}_h(v).$$

We then need to show that

$$-v^T u + \tfrac{1}{2}\|u\|_2^2 + h(u) = -\tfrac{1}{2}\|u\|_2^2$$

From the Moreau decomposition [17] we have $v = \text{prox}_h(v) + \text{prox}_{h^*}(v)$, where h^* is the conjugate of h. Using $\text{prox}_{h^*}(v) = v - u$ we have

$$\begin{aligned}
&-v^T u + \tfrac{1}{2}\|u\|_2^2 + h(u) = -(u + (v-u))^T u + \tfrac{1}{2}u^T u + h(u) \\
&= -\tfrac{1}{2}\|u\|_2^2 - (v-u)^T u + h(u) \\
&= -\tfrac{1}{2}\|\text{prox}_h(v)\|_2^2 - \text{prox}_{h^*}(v)^T \text{prox}_h(v) + h(\text{prox}_h(v)) \\
&\overset{(a)}{=} -\tfrac{1}{2}\|\text{prox}_h(v)\|_2^2 - h^*(\text{prox}_{h^*}(v)) \\
&\overset{(b)}{=} -\tfrac{1}{2}\|\text{prox}_h(v)\|_2^2,
\end{aligned}$$

where (a) follows from [18, Lemma 2.10], and (b) follows from the fact that h^* is the indicator function for the dual norm of h, which implies that $\text{prox}_{h^*}(v)$ is an element of the dual norm ball, and $h^*(\text{prox}_{h^*}(v)) = 0$. □

Application of Theorem 6 to (29) with proximal operator [see also (18)]

$$\text{prox}_h(v) = \text{sign}(v)\big[|v| - \tfrac{\lambda w}{\mu}\big]_+,$$

we obtain

$$\begin{aligned}
m(y,\lambda) &= -\frac{\mu}{2}\left\|\text{sign}(\tfrac{1}{\mu}(A^T y - c))\left[\left|\tfrac{1}{\mu}(A^T y - c)\right| - \frac{\lambda w}{\mu}\right]_+\right\|_2^2 \\
&= -\frac{1}{2\mu}\left\|\Big[\big|A^T y - c\big| - \lambda w\Big]_+\right\|_2^2.
\end{aligned}$$

The same expression holds for $\lambda = 0$ and substitution into (24) therefore gives the following dual problem:

$$\underset{y,\ \lambda \geq 0}{\text{maximize}} \quad y^T b - \tau\lambda - \tfrac{1}{2}\|y\|_2^2 - \tfrac{1}{2\mu}\left\|\Big[\big|A^T y - c\big| - \lambda w\Big]_+\right\|_2^2. \tag{30}$$

Even when restricting y to the current residual r in the primal formulation, it can be seen that the value of (30) is never smaller than that of (26) and, consequently, that the duality gap never exceeds the value in (27). In particular, by choosing $\lambda = \|A^T y + \mu x - c\|_{\frac{1}{w},\infty}$, it follows that for any index i we have

$$\lambda \geq \tfrac{1}{w_i}\Big|[A^T x - c]_i + \mu x_i\Big| \geq \tfrac{1}{w_i}\Big|[A^T x - c]_i\Big| - \tfrac{\mu}{w_i}|x_i|.$$

Multiplying either side by w_i and rearranging gives

$$\Big|[A^T y - c]_i\Big| - \lambda w_i \leq \mu|x_i|.$$

Because the right-hand side is always nonnegative, this continues to hold when applying the $[\cdot]_+$ operator on the left-hand side, and as a result we have

$$\tfrac{1}{2\mu}\left\|\left[\left|A^T y - c\right| - \lambda w\right]_+\right\|_2^2 \leq \tfrac{\mu}{2}\|x\|_2^2,$$

from which the desired result immediately follows.

Finding a dual-feasible solution It follows from Slater's condition and strong duality that, at the solution (x, r) for (25), we have $y = r$ and, without loss of generality, $\lambda = \|A^T r - c\|_{\frac{1}{w},\infty}$. When r is not optimal, we can still choose $y = r$ and obtain a dual-feasible solution. For (30) we can also take $y = r$, but finding λ requires some more work. In general, given any y we want to find a λ that maximizes the objective. Writing $z = |A^T r - c|$ and ignoring constant terms, this is equivalent to solving

$$\lambda^* := \operatorname*{arg\,min}_{\lambda \geq 0} \quad \tau\lambda + \tfrac{1}{2\mu}\|[z - \lambda w]_+\|_2^2.$$

With $\mathcal{I}(\lambda) := \{i \mid z_i \geq \lambda w_i\}$ we can write the objective as

$$f(\lambda) = \tau\lambda + \tfrac{1}{2\mu}\sum_{i \in \mathcal{I}(\lambda)} (z_i - \lambda w_i)^2$$

Discarding all zero terms with $z_i = 0$, this function is piecewise quadratic with breakpoints at $\lambda_i = w_i / z_i$. We can write the sequence of breakpoints in non-decreasing order as $\lambda_{[i]}$ for $i = 0, \ldots, n$, with $\lambda_{[0]} := 0$. The gradient between successive breakpoints is linear and continuously increases from $f'(0) = \tau - 1/\mu \sum_i w_i z_i$ to $f'(\lambda_{[n]}) = \tau$. In order to find the optimal point λ^*, we consider two cases. In the first case we have $f'(0) \geq 0$, or equivalently $\tau \geq 1/\mu \sum_i w_i z_i$, which means that the function is non-decreasing and we find $\lambda^* = 0$. In the second case we need to find λ for which the gradient vanishes. This can be done by traversing the breakpoints until the first breakpoint is found where the gradient is nonnegative. The desired solution λ^* is then found by linear interpolation over the last segment. Including sorting this can be done in $\mathcal{O}(n \log n)$ time. This problem is very similar to projection onto the one-norm ball, and can also be evaluated in expected $\mathcal{O}(n)$ time using an algorithm similar to that proposed in [16].

Primal-dual pairs Using the methods described above, we can compute an upper bound on the duality gap given a feasible x and the corresponding residual r. We can do at least as good, and often better, by maintaining the maximum dual objective found so far, and using this to determine the relative duality gap. This way it is possible for the primal objective for the current iterate x to attain the desired optimality tolerance while the corresponding dual estimate is far from optimal. Within the root-finding framework this means that we cannot simply use the latest residual r to evaluate the gradient of the Pareto curve. Instead we should maintain the value of λ corresponding

to the best dual solution at any point, and use this as the gradient approximation. In other words, we need to keep track of the best primal and dual variables separately.

4.4 Convergence

In many practical situations, the $m \times n$ matrix A in (LS_τ) will have more columns than rows ($m < n$). Due to the presence of a null space, it follows that Assumption 3 is not satisfied, and the convergence result in Theorem 4 does not apply as is. For the result to apply it suffices to ensure that condition (12) applies to the restriction $\hat{f}$ of f on each of the faces of $\mathcal{C}$. A sufficient condition for this is that the columns in A are in general position, such that there does not exist any subset of m or fewer columns of A that is rank deficient. The hybrid algorithm can be updated to check for the self-projection cone and perform quasi-Newton steps only when the number of nonzeros in the current iterate are at most m.

5 Numerical experiments

In this section we evaluate the performance of the hybrid approach on the Lasso problem (LS_τ) both independently and within the SPGL1 root-finding framework [2] described in the introduction. The SPGL1 solver can be used both for stand-alone Lasso problems, as well as for basis-pursuit denoise (BP_σ) problems. For the hybrid method we are mostly concerned with the performance of the former and we therefore changed SPGL1 in two stages. First we modified the stopping criteria used in the Lasso mode, now declaring a solve successful only if the relative duality falls below a certain tolerance level. We then added all modifications needed for the implementation of the hybrid approach. To distinguish between the different algorithms, we use the convention that SPGL1 is used only to refer to the existing implementation provided by [2]. We refer to the version of SPGL1 with the more stringent stopping criteria as the SPG method, which is then extended with the techniques described in this paper to obtain the hybrid method. For the hybrid method we use an L-BFGS memory size of eight for all experiments.

When used in the root-finding mode to solve (BP_σ), SPGL1 uses several different criteria to decide when to update τ. Each subproblems in SPGL1 is considered solved when the relative change in objective is small, and at least one iteration was taken within the current subproblem. The overall problem is declared solved when $\|A^T y\|_\infty$, the relative difference between $\|r\|_2$ and σ, or the relative duality gap is sufficiently small. For the basis-pursuit denoise experiments based on the SPG and hybrid algorithms, we use a separate implementation of the root-finding framework in which each Lasso subproblem is fully solved before updating τ. The differences in stopping criteria, and especially the lack of guarantees on the duality gap for the final subproblem in SPGL1, make it difficult to compare the performances directly. We therefore focus predominantly on how the performance of the hybrid method differs from the reference SPG method.

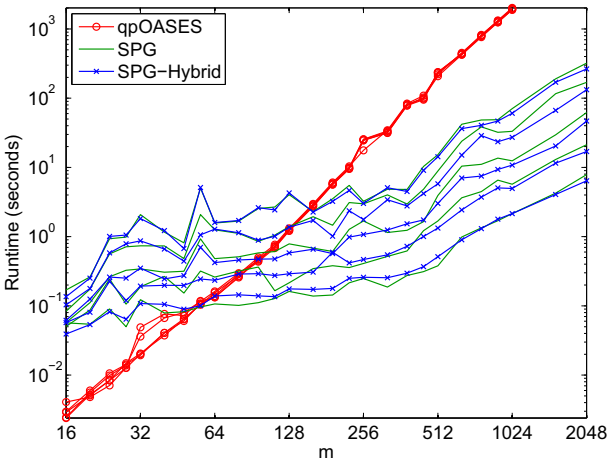

Fig. 3 Comparison of qpOASES with SPG and SPG-hybrid on random gaussian problems of size $m \times 2\,\mathrm{m}$. The different curves from bottom to top correspond to increasing τ values. In qpOASES the curves overlap

5.1 Lasso problems

5.1.1 Active-set type method

Given their similarity, we first compare the performance of the hybrid method with an active-set type method. For this, we rewrite the Lasso subproblem (LS_τ) as a quadratic program (QP) by splitting x into its positive and negative components:

$$\min_{\bar{x}} \quad \tfrac{1}{2}\|[A, -A]\bar{x} - b\|_2^2 \quad \text{subject to} \quad \bar{x} \geq 0,\ e^T\bar{x} \leq \tau. \tag{31}$$

As a solver we choose qpOASES [19], a parametric active-set method that relies on solving linear systems of equations, and therefore generally works best for small to medium scale problems.

For the comparison we generate 20 random $m \times 2\,\mathrm{m}$ matrices A and vectors b with i.i.d. random gaussian elements and columns normalized to unit length. For each problem instance we use SPGL1 to solve the basis-pursuit denoise problem with $\sigma \in \{0.2, 0.1, 0.05, 0.02, 0.01\}$ to get corresponding τ values. We then solve the first Lasso problem with qpOASES, as well as SPG and the hybrid method, both with relative duality tolerance 10^{-6}. In preliminary runs it was found that the runtime for qpOASES is insensitive to the problem instance and we therefore ran only the first problem instance for each τ value. For the SPG and hybrid methods we ran all 20 instances. The (average) runtime for the three methods is plotted against problem size m in Fig. 3. The first thing to notice is that the runtime of qpOASES is insensitive to the value of τ, whereas the runtime for SPG and the hybrid method increases with τ (corresponding to smaller σ). As expected, qpOASES performs well on small problems. However, it scales poorly with increasing problem size, and becomes uncompetitive beyond $m \approx$

200. Using the procedure described in Sect. 4.3 we generated dual-feasible solutions for each of the primal solutions obtains with qpOASES. This showed that the relative duality gap for solutions by qpOASES was around 10^{-12}, which is much smaller than the threshold used for SPG and the hybrid method. To ensure the comparison is fair we modified qpOASES to keep track of the relative duality gap at each iteration to see if early stopping was possible. This turned out not to be the case, since the one-norm of the iterates was found to substantially exceed τ for all but the last several iterations, thereby prohibiting early stopping.

The total runtime over all problem instances using qpOASES amounted to 24,725 s. Even with additional problem instances for $m = 1536$ and $m = 2048$, the total of the (average) runtime for SPG and the hybrid method were 1492 s and 1206 s. The total runtime for the hybrid method is 19% less than that of SPG.

5.1.2 Comparison with PNOPT

The PNOPT algorithm, introduced by Lee et al. [20], is a proximal Newton-type method for solving problems of the form

$$\min_x \quad f(x) + h(x),$$

where $f(x)$ is convex and continuously differentiable, and $h(x)$ is convex, but not necessarily differentiable. By choosing $f(x) = \frac{1}{2}\|Ax - b\|_2^2$ and setting $h(x)$ to the indicator function of the one-norm ball of radius τ, PNOPT can solve the Lasso formulation (LS_τ). We configure PNOPT to use an L-BFGS quadratic model for $f(x)$ with the default memory size of 50 (using a memory size of eight, as used in the hybrid method, gave similar results). Differences in the stopping criteria make it difficult to directly compare the performance of PNOPT with SPG and the hybrid method. We determine relative duality gap tolerance values and PNOPT runtimes using the following steps:

1. We run PNOPT on each of the problem instances from Sect. 5.1.1 using solver-specific tolerance values of 10^{-4}, 10^{-5}, and 10^{-6}.
2. Given the solutions for each of the three settings, we determine the relative duality gap, as described in Sect. 4.3, and choose this value as the reference tolerance. The median of the resulting tolerance values are plotted in Fig. 4a.
3. In order to ensure that PNOPT did not reach the stopping criterion much earlier, we created a modified version of PNOPT to evaluate the duality gap at each iteration to determine the first iteration at which the reference tolerance is attained. In most cases this number is equal to the number of iterations in step 1, in the remaining cases it is less due to the choice of the threshold value.
4. Finally, we rerun the original PNOPT solver with the number of iterations limited to the value found in the previous step. This is the first iteration at which the relative duality gap tolerance is attained.
5. The time reported for each problem instance and setting for PNOPT is then chosen as the minimum of the runs in steps 1, 3, and 4. Note that this approach is advan-

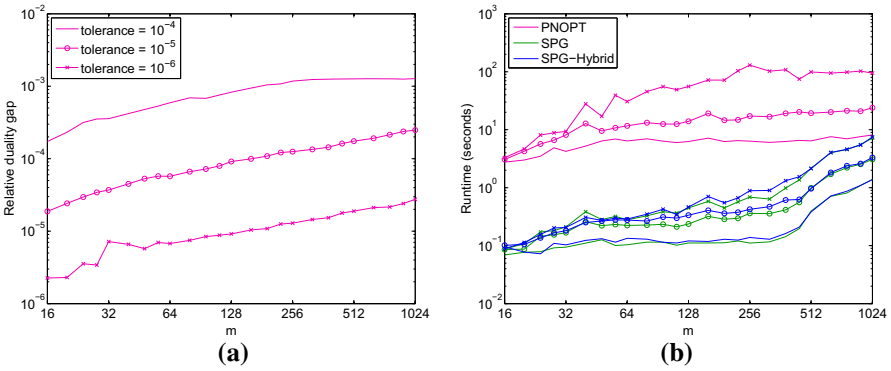

Fig. 4 Plots of **a** the median relative duality gap for ten problem instances solved with PNOPT, and **b** the median runtime over the problem instances using PNOPT, SPG, and the hybrid method with a relative duality gap threshold corresponding to the settings in plot (**a**)

tageous to PNOPT because it effectively excludes the time needed to evaluate the relative duality gap.

For SPG and the hybrid method we use the reference tolerance found in step 2 above as the stopping criterion. Figure 4b plots the median runtimes over 10 problem instances for the problem sizes in Sect. 5.1.1 and the relative duality tolerances corresponding to the three original tolerance values (we plot the median rather than the average runtime due to the presence of problem instances that take excessive time to solve using PNOPT; for SPG and the hybrid method the mean and median are quite similar). It can be seen that both SPG and the hybrid method are much faster than PNOPT for all problem sizes. Between SPG and the hybrid method we see that the hybrid method is slightly slower than SPG, especially on the small problem sizes. This can be attributed to the large tolerance values for the relative duality gap and the low number of iterations needed to reach it, which means that the quasi-Newton method either never activates or only activates several iterations before SPG finishes. In this case, the overhead associated with maintaining the support cannot be offset by the benefits of any quasi-Newton steps.

5.1.3 Lasso on sparse problems

We now take a closer look at the occurrence of line-search errors in SPG and the speed up obtained using the hybrid method. For this we generate test problems following a conventional compressed-sensing scenario and choose A to be a random 1024×2048 matrix with i.i.d. normal entries with columns normalized to unit norm. We set $b = Ax_0$, for k-sparse vectors x_0 with random support and its non-zero entries generated i.i.d. from (1) the normal distribution; (2) the uniform distribution over $[-1, 1]$; and (3) the discrete set $\{-1, +1\}$ with equal probability. We set $\tau = 0.99 \cdot \|x_0\|_1$ and terminate the algorithm whenever the relative duality gap, defined here as $(f(x) - f_{\text{dual}})/\max\{f(x), 10^{-3}\}$, falls below a given tolerance level. We run 50 instances for each of the three distributions used above and report in Table 1 the results obtained

Table 1 Comparison between the SPG and the hybrid method on exact sparse problems with k non-zero entries

k	Runtime SPG (s)			Rel.gap	✓	Runtime hybrid (s)			Rel.gap	✓	(%)
50	0.17	0.16	0.16	3.8e−7	100	0.20	0.19	0.19	4.0e−7	100	− 17
75	0.24	0.21	0.21	5.8e−7	99	0.24	0.23	0.23	4.7e−7	100	− 7
100	0.28	0.27	0.31	5.8e−7	95	0.28	0.27	0.28	5.3e−7	100	3
125	0.39	0.36	0.37	6.5e−7	91	0.34	0.32	0.31	6.2e−7	100	14
150	0.44	0.42	0.45	7.3e−7	92	0.39	0.39	0.37	6.7e−7	100	13
175	0.64	0.53	0.55	7.7e−7	81	0.46	0.44	0.45	6.9e−7	100	21
200	0.74	0.66	0.73	8.2e−7	75	0.58	0.54	0.53	7.1e−7	100	23
225	0.90	0.88	0.86	8.5e−7	71	0.71	0.68	0.65	7.5e−7	100	23
250	1.30	1.10	1.17	9.7e−7	55	0.93	0.86	0.84	7.6e−7	100	26
275	1.65	1.54	1.42	1.0e−6	50	1.30	1.14	1.07	8.3e−7	100	24
300	2.39	2.01	1.85	2.1e−6	35	1.86	1.53	1.44	8.3e−7	100	23
325	3.60	2.76	2.66	4.7e−6	26	3.08	2.04	2.00	8.8e−7	100	22
350	6.61	4.31	3.91	7.6e−6	15	5.68	3.30	3.08	9.0e−7	100	19
375	31.82	8.27	7.15	1.5e−5	7	20.09	6.19	5.68	9.1e−7	100	28
400	218.46	23.16	16.16	3.0e−5	9	129.42	14.65	10.93	9.5e−7	89	37

The first three columns for either method give the average runtime over 50 instances when the non-zero entries are sampled i.i.d. from respectively the discrete $\{-1, 1\}$, uniform $(-1,1)$, and the normal distribution. The fourth column gives the median relative duality gap at the final iteration taken over all 150 problem instances and should be compared with the optimality tolerance, which was set to 10^{-6}. The fifth column for each of the two blocks, indicated by the check mark, gives the percentage of runs that completed successfully, that is, completed without a line-search error. The right-most column gives the average of the speed-up values for each of the three distributions

with an optimality tolerance of 10^{-6}. In general we see that the runtime goes up considerable as we keep increasing k. Moreover, the results show clear differences in the runtime for the three distributions with a much higher runtime for problems based on sparse vectors with ± 1 entries. For sparsity levels up to around one hundred the number of iterations in the SPG method is relatively small (between 25 and 50). For these problems the hybrid method may complete before or soon after the first quasi-Newton step is taken. The slight overhead of the method and occasionally a small number of additional iterations make the hybrid method somewhat slower on average for these problems than the SPG method. For larger values of k, the number of iterations goes up, and the effect of the quasi-Newton steps in the hybrid method becomes apparent with average speed up values between 20 and 30%. Aside from reduced runtime we see from Table 1 that the hybrid method also manages to solve many more problems to the desired accuracy level compared to the SPG method; the number of solved problems steadily falls to around 9% with increasing k for the SPG method, but remains at 100% for all but the largest k for the hybrid method. The median relative duality gap provides further information about the level of accuracy reached before the algorithm completes or terminates with a line-search error. For the largest values of k, the SPG method fails to complete with a relative duality gap of even 10^{-5} for at least half of the problems.

5.2 Root finding

Given that most of the runtimes appearing in Table 1 are of the order of seconds, it is valid to question whether these problems are too idealized and well behaved to give a good idea about the practical performance of the algorithms. In this section we therefore look at two different types of problems. First we introduce a class of random problems that better reflect conditions found in practical problems. Second we evaluate the performance on the Sparco [21] collection of test problems for sparse reconstruction.

5.2.1 Coherent test problem generation

In the compressed-sensing literature it is well known that a random Gaussian matrix satisfies with high probability that all sufficiently small subsets of columns form a near-orthogonal basis for the subspace spanned by these columns—a property known as the restricted isometry [22]. Another quantity used to characterize matrices is the mutual coherence, defined as the maximum absolute pairwise cosine distance between the columns. In practical applications the matrix A is often quite coherent [23], and although there are no theoretical results on how this affects the complexity of one-norm minimization, it has been observed empirically that more coherent problems are harder to solve. The construction we propose for generating such problems is by means of a random walk on the $(m-1)$-sphere with a step size parameterized by γ. Starting with a unit-norm column a_1 we construct successive columns by sampling a vector v_k with i.i.d. Gaussian entries and setting $a_{k+1} = \alpha_1 a_k + \alpha_2 v_k$, where α_1 and α_2 are chosen such that $\|a_{k+1}\|_2 = 1$ and $\langle a_k, a_{k+1}\rangle = 1-\gamma$. In other words, a_{k+1} lies on the boundary of a spherical cap with center center a_k and angle θ such that $\cos(\theta) = 1-\gamma$. The mutual coherence of the resulting matrix is lower bounded by $1-\gamma$, and an example of the distribution of the pairwise cosine distance between the columns is given in Fig. 5a. An example Gram matrix, plotted in Fig. 5b, shows that aside from the banded structure, there are regions of increased coherence whenever the random walk approaches earlier locations. From Fig. 5c we see that lowering γ while keeping a_1 and v_k fixed leads to an increase of the top singular value σ_1 as the columns become more and more similar. Figure 5d illustrates that the maximum pairwise coherence μ does not necessarily have a relationship with the top singular value.

5.2.2 Highly coherent measurement matrices

We apply SPG and the hybrid method to solve (BP_σ) using the root-finding framework explained in Sect. 1. Each Lasso subproblem (LS_τ) is optimized to a certain optimality tolerance, and the overall problem is considered solved whenever the relative misfit $|\sigma - \|r\|_2|/\max(\sigma, 10^{-3})$ falls below 10^{-5}. For completeness we also compare the performance with the SPGL1 algorithm as provided by [2].

For the first set of experiments we use the highly coherent matrices described in Sect. 5.2.1. As before we create a k-sparse vector x_0 with non-zero entries sampled from different distributions, and set $b = Ax_0 + v$, where the entries in v are zero in the

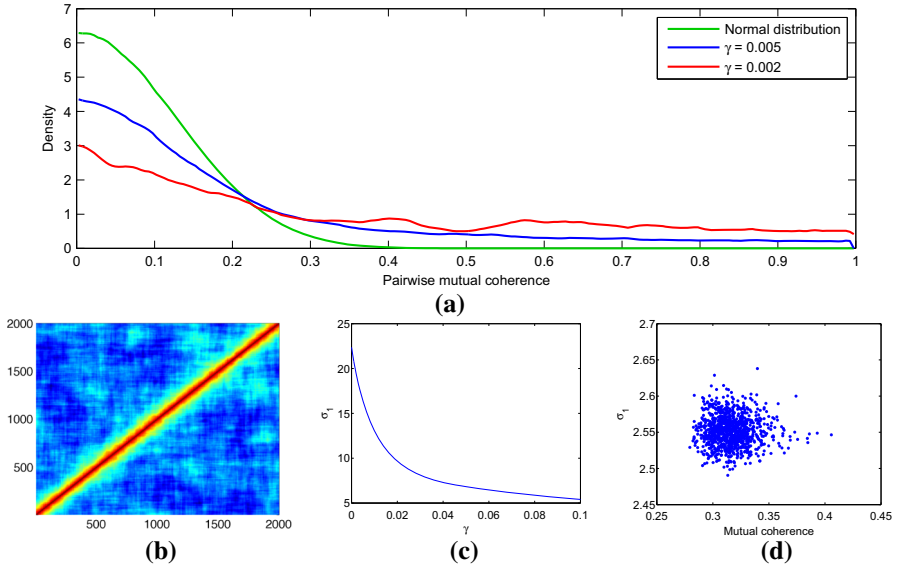

Fig. 5 **a** Distribution of pairwise mutual coherence between vectors of two types of 64×2048 matrices with unit-normalized columns generated as: (i) random vectors with i.i.d. normal entries, and (ii) a random walk over the sphere with the mutual coherence between successive columns equal to $1-\gamma$; **b** Gram matrix of a 200×2000 matrix generated with $\gamma = 0.01$; **c** the top singular value of a 200×500 matrix generated with the same a_1 and v_k for different values of λ; and **d** the mutual coherence and top singular value for 1000 random Gaussian 200×500 matrices with columns scaled to unit norm

noiseless case, and sampled i.i.d. from the normal distribution and scaled to the desired noise level otherwise. For the noiseless results in Table 2a–c we run ten instances for each of the three distributions and report the average run time over all thirty runs. The percentage reduction in runtime is computed based on the total runtime and was found to closely match the percentage obtained for each of the three signal classes independently. For the root-finding columns we solve (BP_σ) with $\sigma = 0.01\|b\|_2$ and optimality tolerance levels of 10^{-4} and 10^{-6}. For the Lasso columns we solve (LS_τ) on equivalent problems with τ set to the value obtained using the root-finding procedure. The results in Table 2d–f apply to noisy problems where $\|v\|_2$ is scaled to the given percentage of $\|Ax_0\|_2$, and σ is set accordingly. For these experiments we only consider sparse x_0 with random ± 1 entries. The percentage decrease in runtime relative to SPG, for SPGL1 and the hybrid method for all problem instances in Table 2 is given in Fig. 6. A summary of the total runtime for the different solvers along with the percentage of solutions with relative duality gap within given ranges is provided by Table 3.

The first thing to note from the results in Table 2 is that the problems generated with lower values of γ are indeed more difficult to solve for both SPG and the hybrid method. Compared to the SPG method, the hybrid method reduces the average runtime for nearly all problems, and does so by a percentage that increases as the problems get harder, a trend clearly seen in Fig. 6. From Table 3 we see that the hybrid method with optimality tolerances of 10^{-4} and 10^{-6} reduces the total runtime respectively

Table 2 Comparison between SPGL1 and root finding with strict tolerance levels using the SPG and hybrid method

γ	SPGL1 runtime	Root finding tolerance 10^{-4}			Root finding tolerance 10^{-6}			Lasso tolerance 10^{-4}			Lasso tolerance 10^{-6}		
(a) Sparsity $k = 10$													
.100	0.7	0.7	0.8	**−4**	0.7	0.8	**−4**	0.7	0.7	**−4**	0.7	0.7	**−3**
.050	1.6	1.7	1.7	**2**	1.9	1.7	**12**	1.8	1.7	**1**	1.9	1.8	**7**
.020	6.8	6.6	5.8	**12**	7.6	5.9	**22**	6.5	6.0	**7**	7.1	6.1	**14**
.010	17.8	17.7	14.7	**17**	21.4	15.0	**30**	16.7	15.5	**7**	18.3	15.6	**15**
.005	51.9	43.6	36.8	**16**	52.6	38.2	**27**	40.6	37.9	**7**	42.7	38.2	**11**
(b) Sparsity $k = 50$													
.100	1.8	3.1	2.7	**11**	5.0	3.2	**37**	1.8	1.7	**7**	2.5	1.8	**29**
.050	5.1	8.2	7.1	**13**	11.3	7.9	**30**	4.7	4.3	**8**	5.8	4.6	**21**
.020	19.8	29.4	23.2	**21**	36.5	25.8	**29**	17.5	15.5	**11**	18.9	16.1	**15**
.010	48.5	77.7	58.9	**24**	92.5	63.0	**32**	45.9	40.0	**13**	47.9	41.2	**14**
.005	145.7	221.9	144.9	**35**	263.4	153.4	**42**	127.7	101.9	**20**	134.9	104.7	**22**
(c) Sparsity $k = 100$													
.100	3.2	5.2	4.5	**13**	7.8	5.3	**32**	2.8	2.6	**8**	3.6	2.8	**23**
.050	8.6	14.9	12.4	**17**	19.4	13.8	**29**	7.6	6.9	**10**	8.8	7.3	**17**
.020	29.9	55.1	41.7	**24**	65.8	45.2	**31**	29.3	25.3	**14**	31.0	26.1	**16**
.010	72.4	158.9	110.5	**30**	188.0	115.3	**39**	84.4	68.7	**19**	88.3	70.5	**20**
.005	224.2	502.7	302.4	**40**	596.0	312.6	**48**	251.3	187.1	**26**	261.5	189.8	**27**
(d) Sparsity $k = 50$, *noise level 1%*													
.100	2.2	4.7	4.2	**12**	8.3	5.3	**37**	2.3	2.1	**9**	2.9	2.3	**22**
.050	7.3	12.5	10.0	**20**	18.1	12.0	**34**	5.8	5.4	**8**	6.7	5.7	**15**
.020	28.7	43.6	33.1	**24**	54.9	37.7	**31**	23.0	19.9	**14**	24.4	20.6	**15**
.010	68.8	118.0	84.1	**29**	140.5	93.4	**34**	59.4	51.5	**13**	62.3	52.9	**15**
.005	233.1	367.5	221.5	**40**	447.5	234.4	**48**	180.1	140.6	**22**	189.9	143.0	**25**
(e) Sparsity $k = 50$, *noise level 5%*													
.100	1.8	3.9	3.3	**13**	5.6	4.2	**26**	1.6	1.5	**4**	2.0	1.6	**16**
.050	4.6	9.3	7.7	**17**	13.1	9.3	**29**	3.8	3.6	**4**	4.5	3.9	**13**
.020	15.2	30.8	23.7	**23**	42.3	27.1	**36**	13.0	11.4	**12**	15.1	12.1	**20**
.010	32.4	78.5	55.2	**30**	109.6	62.3	**43**	33.2	27.9	**16**	39.2	28.7	**27**
.005	66.5	219.2	131.4	**40**	334.7	139.7	**58**	90.2	68.6	**24**	115.8	70.1	**39**
(f) Sparsity $k = 50$, *noise level 10%*													
.100	1.5	3.4	3.0	**10**	4.5	3.7	**18**	1.4	1.3	**8**	1.7	1.4	**13**
.050	3.3	7.9	6.7	**16**	10.9	8.2	**25**	3.2	2.8	**13**	3.8	3.1	**18**
.020	9.5	24.7	19.1	**23**	34.3	22.2	**35**	10.0	8.3	**17**	12.0	8.8	**27**
.010	19.5	61.7	44.6	**28**	85.9	50.9	**41**	24.5	19.2	**22**	29.1	20.2	**31**
.005	33.2	154.5	98.1	**37**	263.1	107.7	**59**	60.5	43.1	**29**	84.7	44.8	**47**

The columns within the root finding and Lasso blocks are respectively the runtime in seconds of the SPG and hybrid method, and (in bold) the reduction in runtime in percent of the hybrid method compared to the SPG method

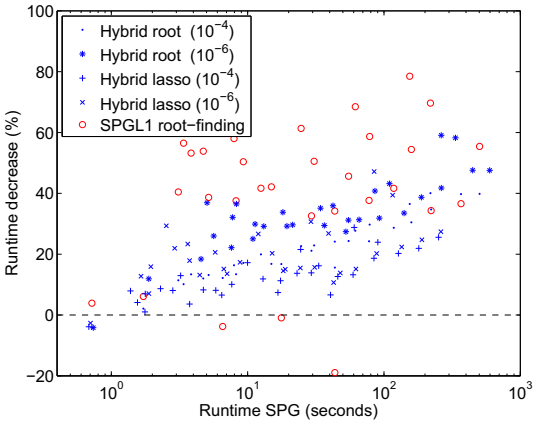

Fig. 6 Percentage runtime reduction relative to SPG for solvers SPGL1 and the hybrid method on a set of highly coherent problems. The hybrid method is run on both the root-finding and lasso problems with two optimality tolerance levels each

Table 3 Total runtime for the coherent problems with different methods and optimality tolerances, along with the percentage of instances that attain a relative duality gap in the given intervals at the final iteration

Type	Method	Tol.	Time	Relative duality gap ($\log_{10}$)					
				≤ -6	$(-6,-5]$	$(-5,-4]$	$(-4,-3]$	$(-3,-2]$	>-2
BP_σ	SPGL1	10^{-6}	7h25	0.2	1.5	4.6	15	49	29
	SPG	10^{-6}	17h29	39	33	28	0.2	–	–
	Hybrid	10^{-6}	9h53	100	–	–	–	–	–
	SPG	10^{-4}	14h02	0.2	0.6	99	0.5	–	–
	Hybrid	10^{-4}	9h18	0.2	0.8	99	–	–	–
LS_τ	SPG	10^{-6}	7h55	36	33	30	0.5	–	–
	Hybrid	10^{-6}	6h02	100	–	–	–	–	–
	SPG	10^{-4}	7h20	–	1.2	98	0.5	–	–
	Hybrid	10^{-4}	5h54	–	1.2	99	–	–	–

The reduction in runtime for the successive SPG-hybrid pairs are 43, 34, 24, and 20%, respectively

by 34% and 43% for the basis-pursuit problems, and 20% and 24% for the Lasso problems. The larger relative reduction in runtime for basis pursuit is due to the use of warm starting in the root-finding procedure, which removes a substantial number of iterations that would otherwise be identical for the hybrid and SPG methods. Despite the improvements, the hybrid method still has a larger runtime than SPGL1 on most problems. However, from Table 3 we see that SPGL1 does not even reach a relative duality gap of 10^{-3} for nearly 80% of the problems, as a result of the relaxed stopping criteria. Tightening these criteria, as done in what we label the SPG method, increases the number of solutions that attain the desired optimality tolerance. Nevertheless, the SPG method still fails to reach an optimality of 10^{-6} for some 60% of the problems. Finally, we see that the hybrid method not only improves the runtime of the SPG method, but also manages to reach the requested optimality on all problems from Table 2.

Table 4 Selected Sparco problems

Problem	ID	m	n	$\|b\|_2$	Scale	$\|x^*_{1e-2}\|_1$	$\|x^*_{1e-3}\|_1$
blurrycam	701	65,536	65,536	1.3e+2	100	2.8e+5	9.1e+5
blurspike	702	16,384	16,384	2.2e+0	100	2.5e+4	3.4e+4
soccer1	601	3200	4096	5.5e+4	1	7.9e+1	3.1e+2
spiketrn	903	1024	1024	5.7e+1	100	1.3e+3	1.3e+3
yinyang	603	1024	4096	2.5e+1	100	2.5e+4	2.6e+4
srcsep1	401	29,166	57,344	2.2e+1	1	1.0e+3	1.0e+3
srcsep2	402	29,166	86,016	2.3e+1	1	1.1e+3	1.1e+3

5.2.3 Sparco test problems

Sparco [21] provides a standard collection of test problems for compressed sensing and sparse recovery. The problems in Sparco are of the form $b = Ax + v$, where A is represented as a linear operator rather than an explicit matrix. After excluding problems that are too easy to solve or require access to third-party software, we obtain the problem selection listed in Table 4. For some problems we scale the original b to avoid a very small objective value at the solution, which causes the duality gap relative to $\max(f(x), 1)$ to be satisfied more easily. The table also lists the one-norm of the solutions found when solving with $\sigma = 0.01\|b\|_2$ and $\sigma = 0.001\|b\|_2$, respectively, for the scaled b.

We run the SPG and hybrid methods with optimality tolerances ranging from 10^{-2} down to 10^{-4}. Beyond that, some of the problems simply took too long to finish. For SPGL1 we use optimality tolerance values set to 10^{-6} and 10^{-9}. By comparison these may seem excessively small, and we certainly do not expect the relative duality gap to reach these levels. Instead, we choose the small values to help control the other stopping criteria, such as the relative change in the objective value, which are parameterized using the same tolerance parameter. The results of the experiments with the two choices of σ, are summarized in Tables 5 and 6. The hybrid method reduces the runtime of the SPG method in 42 out of the 56 settings, often considerably so. For a tolerance level of 10^{-4} the hybrid method consistently outperforms the SPG method with an average time reduction of 38%. The required optimality level is reached on all problems except for problem 903 with the smaller σ and optimality tolerance 10^{-4}. For this problem the SPG method stops with a relative duality gap of 2×10^{-4} following a line-search error. The runtime for SPGL1 with optimality tolerance 10^{-6} is very low overall, but comes at the cost of a rather large relative duality gap at the solution. Lowering the tolerance to 10^{-9} reduces the gap, but also leads to a considerable increase in runtime. In either case the number of root-finding iterations can be very large, especially if the target value of τ is exceeded and gradual reduction follows. The lowest relative duality gap reached by SPGL1 over all problems in Tables 5 and 6 is 4×10^{-3}. The varying optimality levels make it difficult to compare results, so of special interest are problem instances where SPGL1 simultaneously has a lower runtime and relative duality gap with either the SPG or hybrid method, or vice versa. From the tables we see

Table 5 Sparco problems with $\sigma = 0.01\|b\|_2$, (a) runtime in seconds, (b) relative duality gap, (c) number of root-finding iterations

	SPGL1		SPG	Hybr.	SPG	Hybr.	SPG	Hybr.
	10^{-6}	10^{-9}	Tol = 10^{-2}		Tol = 10^{-3}		Tol = 10^{-4}	
Runtime (s)	18.0	32.2	44.6	54.0	84.1	75.9	91.1	66.7
Rel.gap	3e−1	2e−2	9e−3	9e−3	9e−4	1e−3	1e−4	8e−5
Outer iterations	144	246	12	12	12	10	10	10
► **Problem 701—blurrycam**				**−21.2**		**9.7**		**26.8**
Runtime (s)	9.7	16.7	27.8	28.0	38.9	38.2	38.2	35.4
Rel.gap	6e−1	3e−2	9e−3	1e−2	1e−3	1e−3	1e−4	1e−4
Outer iterations	191	296	10	10	10	10	8	8
► **Problem 702—blurspike**				**−0.5**		**1.8**		**7.2**
Runtime (s)	133	206	28.0	20.1	34.0	17.7	38.0	17.2
Rel.gap	1.5	6e−2	7e−3	6e−3	1e−3	4e−4	1e−5	2e−5
Outer iterations	1274	1945	9	9	12	9	8	9
► **Problem 601—soccer1**				**28.2**		**47.8**		**54.7**
Runtime (s)	2.8	6.6	5.6	3.8	7.6	5.6	11.2	5.3
Rel.gap	3.6	1e−1	8e−3	8e−3	1e−3	8e−4	1e−4	1e−4
Outer iterations	448	1120	6	6	6	10	5	5
► **Problem 903—spiketrn**				**31.3**		**27.2**		**53.0**
Runtime (s)	1.8	2.7	2.5	2.9	2.9	3.1	4.2	4.0
Rel.gap	3e−1	2e−2	7e−3	6e−3	8e−4	8e−4	7e−5	8e−5
Outer iterations	38	60	7	8	7	7	7	7
► **Problem 603—yinyang**				**−13.5**		**-4.5**		**3.7**
Runtime (s)	79.0	418	299	331	1404	1047	6351	2447
Rel.gap	5e−2	1e−1	9e−3	9e−3	1e−3	1e−3	1e−4	1e−4
Outer iterations	37	213	9	20	8	9	9	8
► **Problem 401—srcsep1**				**−10.5**		**25.5**		**61.5**
Runtime (s)	114	622	441	409	1114	1023	2447	1565
Rel.gap	5e−1	1e−1	1e−2	9e−3	9e−4	1e−3	1e−4	6e−5
Outer iterations	30	198	9	9	8	7	7	9
► **Problem 402—srcsep2**				**7.1**		**8.2**		**36.1**

The percentage reduction in time of the hybrid method over SPG is given in bold next to the problem index

that SPGL1 outperforms the SPG method on both instances of problem 702. For problem 401 in Table 5, SPGL1 with an optimality tolerance of 10^{-6} is better, but aside from this problem, SPGL1 consistently has the lowest runtime, but also the largest duality gap. The SPG method with more stringent root-finding iterations dominates SPGL1 with a tolerance level of 10^{-9} on all remaining problems aside from the instance of problem 701 in Table 5. As we saw earlier, the hybrid method performs especially well when the desired relative duality gap is small. Nevertheless, even for large duality gaps it still dominates SPGL1 on nine out of the fourteen problem instances and is dominated on only one.

5.2.4 Primal-dual gap

We now consider the formulation

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|Ax - b\|_2^2 + \tfrac{\mu}{2}\|x\|_2^2 \quad \text{subject to} \quad x \in \mathcal{C}, \tag{32}$$

for $\mu > 0$. In Sect. 4.3 we described two different ways of deriving a dual formulation. In the first approach we augment A and b to account for the $\frac{\mu}{2}\|x\|_2^2$ term and reduce the problem to the standard Lasso formulation. The derivation of the dual for this formulation in [2,15] provides a way of generating a dual-feasible point $(\bar{y}, \bar{\lambda})$ from a primal-feasible x by choosing $\bar{y} = \bar{A}x - \bar{b}$ and solving a trivial optimization problem for $\bar{\lambda}$. In the second approach we deal with formulation (32) directly and obtain a dual problem parameterized in (y, λ). As before we can choose y to be equal to the residual, now in terms of the original A and b, and remain with a non-trivial optimization problem for λ that is nevertheless easily solved using the algorithm described in Sect. 4.3. We refer to the two derivations as the augmented derivation and the optimized derivation. The term 'optimized' refers to the need to solve for λ, but more importantly, to the fact that the dual objective generated from any x using the optimized derivation is never smaller than that using the augmented derivation, as shown in Sect. 4.3.

To evaluate the practical difference between the two approaches we generate a large number of randomized test problems of the form $b = Ax_0 + v$, where x_0 are random vectors with sparsity levels ranging from 50 to 350 in steps of 50 and on-support entries draw i.i.d. from the normal distribution. The $m \times n$ measurement matrices A are drawn i.i.d. from $\mathcal{N}(0, 1/m)$, with $m = 1000$ and $n = 2000$. Finally, the additive noise vectors v are drawn from the normal distribution and scaled to have Euclidean norm equal to 0, 0.01, 0.1, and 1% of that of the clean observation Ax_0. For the problem formulation we set τ to $\|x_0\|_1$ scaled by 0.7–1.2 in 0.1 increments, and range μ log-linearly from 10^{-1} to 10^{-4} in four steps. As a result of the additive $\frac{\mu}{2}\|x\|_2$ term in the objective, the solutions are no longer sparse. As a result the hybrid method tends to coincide with the SPG method, and we therefore only consider the latter for these experiments.

For each of the settings we evaluate the time required by the augmented and optimized formulations to reach a relative duality gap of 10^{-4}. Figure 7a, b plot the speed up obtained using the optimized formulation along with the runtime of the augmented formulation for different problem instances with two levels of μ. Despite the slightly more expensive evaluation of the dual, we see that the optimized formulation is around 1.5–4 times faster for $\mu = 0.01$, and up to 7 times faster for $\mu = 0.001$. For $\mu = 0.1$ (not shown in the plot) the speedup ranges from 1.2 to 3, and for $\mu = 0.001$ the speed up exceeds 10 on many problem instances and reaches a maximum of around 30.

We now take a closer look at the relative distance of the primal and dual objective to the optimum for the two circled problems in Fig. 7c,d. The progress of the primal objective over the iterations, indicated by the gray line, is the same for both formulations. For the dual objective there is a marked difference between the two. Notably,

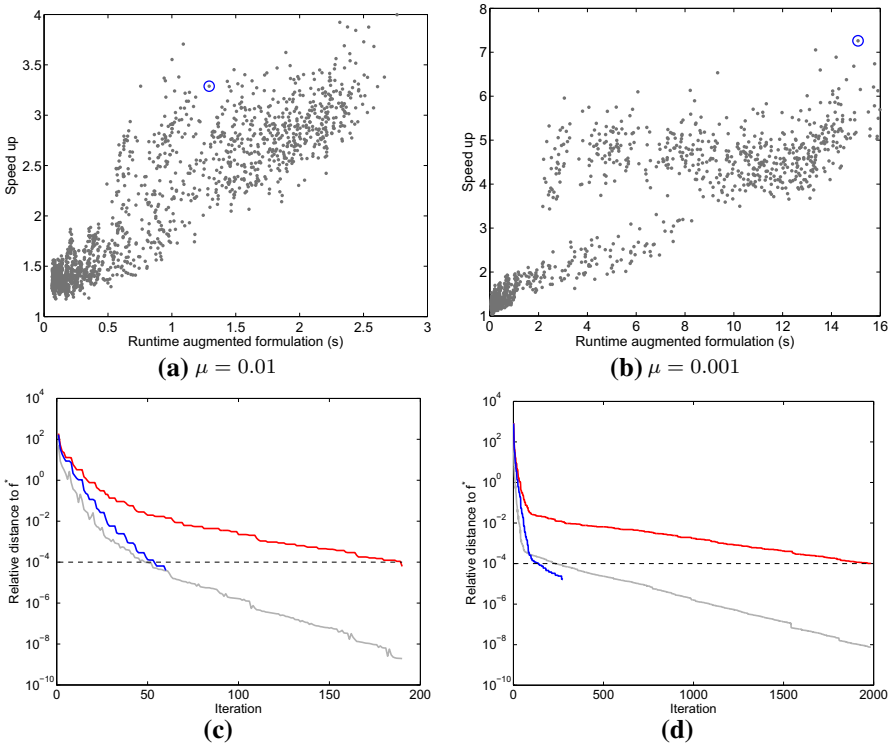

Fig. 7 Plots **a**, **b** the time required to reach the desired relative optimality gap of 10^{-4} using the augmented formulation, and the corresponding speed up obtained using the optimized formulation; each point indicates a single problem instance. Plots **c**,**d** the relative distance of the primal (gray) and dual objective (red for the augmented formulation and blue for the optimized formulation) to the optimal objective as a function of iterations for the two circled problem instances (color figure online)

the augmented formulation converges much slower than the optimized formulation, thereby preventing the stopping criterion from being satisfied for many more iterations.

We now take another look at the Sparco problems from Tables 5 and 6. For each of the settings we record the optimal τ and then run the hybrid solver with a target optimality tolerance of 10^{-8} to obtain a best-effort optimum (for some problems the line search failed before reaching the desired tolerance). We then run the SPG and hybrid solvers with a target accuracy of 10^{-5} and record the relative distance of the primal and dual objective to the optimum at every iteration. The results for four representative problems are plotted in Fig. 8. From the plots we see that the iterates of the hybrid method initially coincide or otherwise closely follow those of the SPG method. Once the hybrid method starts using quasi-Newton iterates increasingly often we see a sharp decrease in the relative distance to the optimum of the primal and dual iterates. The iterates of the SPG method, by contrast, continue to decrease very slowly. Indeed, of the fourteen problem settings, the SPG method managed to solve only two to the desired level of accuracy. Of the remaining problems, two reach the default iteration limit of ten times the number of rows in A, while all other problems

Table 6 Sparco problems with $\sigma = 0.001\|b\|_2$, (a) runtime in seconds, (b) relative duality gap, (c) number of root-finding iterations

	SPGL1		SPG	Hybr.	SPG	Hybr.	SPG	Hybr.
	10^{-6}	10^{-9}	Tol = 10^{-2}		Tol = 10^{-3}		Tol = 10^{-4}	
Runtime (s)	121	200	272	300	316	342	430	333
Rel.gap	2.1	1e−1	9e−3	6e−3	1e−3	8e−4	9e−5	9e−5
Outer iterations	485	640	12	12	11	12	11	11
► Problem 701—blurrycam				**−10.4**		**−8.2**		**22.5**
Runtime (s)	28.8	34.9	45.8	49.3	56.1	59.8	74.4	68.1
Rel.gap	4e−1	7e−2	9e−3	1e−2	9e−4	9e−4	9e−5	9e−5
Outer iterations	127	151	11	11	10	10	10	10
► Problem 702—blurspike				**−7.7**		**−6.5**		**8.5**
Runtime (s)	278	368	155	56.4	203	53.2	257	47.8
Rel.gap	1.1	7e−2	9e−3	8e−3	1e−3	7e−4	1e−4	7e−5
Outer iterations	2059	2707	11	10	9	9	9	10
► Problem 601—soccer1				**63.6**		**73.8**		**81.4**
Runtime (s)	4.8	9.4	9.2	6.2	11.7	6.6	13.5	6.6
Rel.gap	22.5	1.3	1e−2	1e−2	1e−3	9e−4	2e−4	1e−4
Outer iterations	700	1253	8	8	6	8	5	5
► Problem 903—spiketrn				**33.1**		**43.7**		**51.0**
Runtime (s)	9.0	19.0	37.4	26.8	55.1	29.4	71.8	35.8
Rel.gap	1.2	9e−1	1e−2	7e−3	9e−4	1e−3	1e−4	1e−4
Outer iterations	90	190	8	10	8	8	8	8
► Problem 603—yinyang				**28.4**		**46.7**		**50.1**
Runtime (s)	*241*	3309	394	360	2803	2253	21,829	11,193
Rel.gap	*3e−3*	4e−3	1e−2	9e−3	1e−3	1e−3	1e−4	1e−4
Outer iterations	*32*	264	13	12	20	11	10	10
► Problem 401—srcsep1				**8.5**		**19.6**		**48.7**
Runtime (s)	*272*	5999	518	477	2041	1787	8631	6486
Rel.gap	*6e−3*	2e−2	8e−3	8e−3	1e−3	9e−4	1e−4	9e−5
Outer iterations	*18*	324	11	14	9	10	10	10
► Problem 402—srcsep2				**8.0**		**12.5**		**24.9**

The percentage reduction in time of the hybrid method over SPG is given in bold next to the problem index. Entries marked in italic indicate a solution that is not a root

fail with a line-search error. The hybrid method manages to solve all problems except for problem 401 with multiplier 10^{-3}. This problem reached the iteration limit, but could otherwise be solved successfully to a tolerance level of even 10^{-8}.

As before, we see that the dual objective converges to the optimum much slower than the primal, and unfortunately, there is no clear way to extend the optimized dual formulation from Sect. 4.3 to the standard Lasso formulation where $\mu = 0$. Given that the satisfaction of the optimality condition depends almost entirely by the dual objective value, it makes sense to look at the speed up that would be attained if the

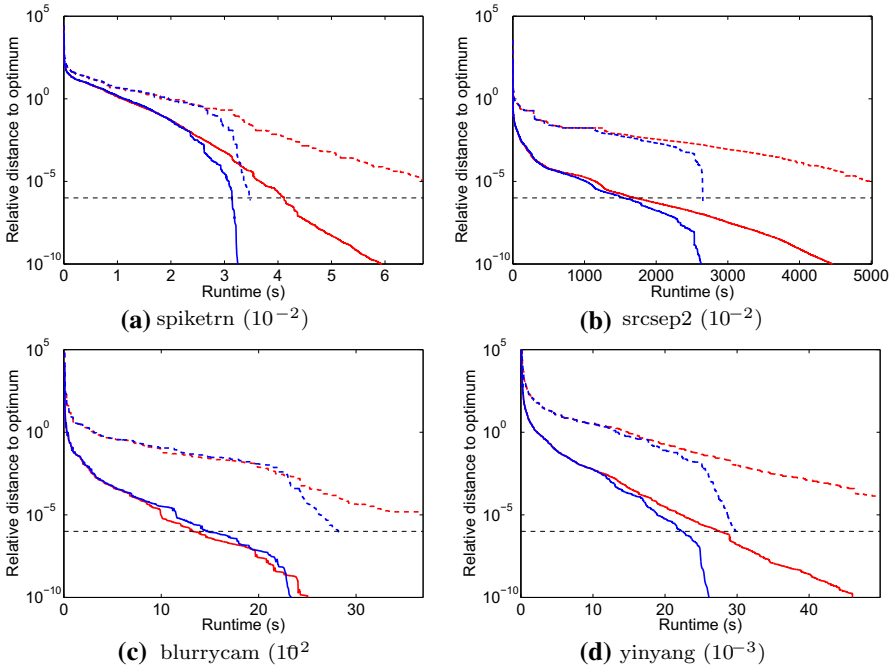

Fig. 8 Relative distance of primal (solid) and dual iterates (dashed) to the optimum as a function of time using the SPG (red) and hybrid method (blue) on four Sparco problems for fixed τ corresponding to the root for σ equal to the given multiple of $\|b\|_2$ (color figure online)

optimal objective value were to be known and satisfaction of the optimality condition depends only on the primal objective. In Table 7 we provide the attainable speed up for the different Sparco problems with varying optimality tolerance levels. Clearly, both the SPG and hybrid methods would benefit greatly from an improved dual, although the effect is less for the hybrid method, due to the already fast convergence of the dual objective in the final iterations.

6 Conclusions

In this paper we have presented a hybrid algorithm for minimization of quadratic functions over weighted one-norm balls. The method extends the spectral projected gradient method with L-BFGS iterations applied to reparameterizations of the objective function over active faces of the one-norm ball. For the decision of the iteration type we introduce the self-projection cone of a face and provide a complete characterization of this cone for weighted one-norm balls. The reparameterization uses an implicit orthonormal basis for the current face, and we provide an efficient algorithm for matrix-vector multiplication with this basis and its transpose.

As part of the numerical experiments we propose a challenging class of test problems in which the columns of the $m \times n$ measurement matrix A are generated based on a random walk over the $(m-1)$-sphere. Based on extensive numerical experiments on

Table 7 Speed up for different tolerance levels when the optimal objective value is given and satisfaction of the optimality condition depends only on the primal objective value

	($\sigma = 0.01$)				($\sigma = 0.001$)			
	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
(a) SPG								
701—blurrycam	5.93	3.79	> 3.66	> 2.79	2.73	> 2.21	> 1.94	> 1.61
702—blurspike	2.39	2.16	> 1.93	> 1.66	2.48	2.20	2.07	> 1.81
601—soccer1	1.25	1.07	1.02	1.02	2.84	2.88	> 2.57	> 2.09
903—spiketrn	1.64	1.73	> 1.83	> 1.63	1.53	> 1.64	> 1.56	> 1.44
603—yinyang	1.70	1.62	1.64	1.52	3.02	> 2.76	> 2.24	> 1.81
401—srcsep1	40.72	26.71	> 9.49	> 4.15	75.78	> 49.75	> 12.30	> 2.38
402—srcsep2	15.96	9.81	> 4.35	> 2.94	56.53	22.74	9.88	> 2.78
(b) Hybrid								
701—blurrycam	5.51	3.20	2.27	1.93	3.62	2.73	1.90	1.48
702—blurspike	2.20	1.81	1.56	1.45	2.30	1.82	1.61	1.42
601—soccer1	1.19	1.08	1.01	1.02	1.11	1.03	1.02	1.02
903—spiketrn	1.23	1.14	1.09	1.11	1.13	1.13	1.11	1.09
603—yinyang	1.80	1.64	1.55	1.45	2.12	1.63	1.52	1.33
401—srcsep1	24.83	11.83	3.72	1.81	65.28	46.18	> 13.44	> 2.61
402—srcsep2	11.62	6.45	2.60	1.72	44.70	16.92	6.07	1.99

We give a lower bound (indicated by the '> ' sign) when the dual objective failed to reach the given optimality level, either because the maximum number of iterations was reached, or because a line-search error occurred

these and other test problems we showed that the hybrid method outperforms the original spectral projected gradient methods on a large fraction of the problems. Especially for medium to high accuracy solves and more challenging problems the SPG method was found to either take much more time to reach the desired level of accuracy, or fail prematurely due to line-search problems. Both methods performed favorably against the parametric active-set solver qpOASES [19] and the proximal Newton-type solver PNOPT [20].

The current stopping criterion used in both SPG and the hybrid method relies on the generation of a dual feasible point from the primal iterate to determine the relative optimality of the iterate. From the experiments we found that the primal objective converges to the optimum much faster than the dual objective, and that satisfaction of the stopping criterion therefore depends entirely on the dual objective reaching the critical threshold. The performance of both methods could therefore be improved substantially given a better dual estimate.

In this paper we have studied the application of the hybrid method to the Lasso problem. Other important problems that may benefit from the approach but were not discussed in this paper include box-constrained optimization and minimization of quadratic functions over the simplex.

Acknowledgements The implementation of the code uses L-BFGS update routines kindly provided by Michael P. Friedlander. The author would like to thank the anonymous referees for useful suggestions.

References

1. Tibshirani, R.: Regression shrinkage and selection via the lasso. J. R. Stat. Soc. Ser. B (Methodol.) **58**(1), 267–288 (1996)
2. van den Berg, E., Friedlander, M.P.: Probing the Pareto frontier for basis-pursuit solutions. SIAM J. Sci. Comput. **31**(2), 890–912 (2008)
3. Chen, S.S., Donoho, D.L., Saunders, M.A.: Atomic decomposition by basis pursuit. SIAM J. Sci. Comput. **20**(1), 33–61 (1998)
4. Candès, E.J., Romberg, J., Tao, T.: Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. IEEE Trans. Inf. Theory **52**(2), 489–509 (2006)
5. Donoho, D.L.: Compressed sensing. IEEE Trans. Inf. Theory **52**(4), 1289–1306 (2006)
6. Birgin, E.G., Martínez, J.M., Raydan, M.: Nonmonotone spectral projected gradient methods on convex sets. SIAM J. Optim. **10**(4), 1196–1211 (2000)
7. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Math. Program. Ser. B **45**(1–3), 503–528 (1989)
8. Bertsekas, D.P.: On the Goldstein–Levitin–Polyak gradient projection method. IEEE Trans. Autom. Control **AC–21**(2), 174–184 (1976)
9. Barzilai, J., Borwein, J.M.: Two-point step size gradient methods. IMA J. Numer. Anal. **8**(1), 141–148 (1988)
10. Fletcher, R.: A limited memory steepest descent method. Math. Program. Ser. A **135**, 413–436 (2012)
11. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer Series in Operations Research and Financial Engineering, 2nd edn. Springer, Berlin (2006)
12. Bertsekas, D.P.: Nonlinear Programming, 2nd edn. Athena Scientific, Belmont (2003)
13. Grünbaum, B.: Convex Polytopes, Volume 221 of Graduate Texts in Mathematics, 2nd edn. Springer, Berlin (2003)
14. van den Berg, E.: A hybrid quasi-Newton projected-gradient method with application to Lasso and basis-pursuit denoise. arXiv:1611.05483 (2016)
15. van den Berg, E., Friedlander, M.P.: Sparse optimization with least-squares constraints. SIAM J. Optim. **21**(4), 1201–1229 (2011)
16. Duchi, J., Shalev-Shwartz, S., Singer, Y., Chandra, T.: Efficient projections onto the ℓ_1-ball for learning in high dimensions. In: Proceedings of the 25th International Conference on Machine Learning, pp. 272–279 (2008)
17. Moreau, J.J.: Proximité et dualité dans un espace hilbertien. Bulletin de la Société Mathématique de France **93**, 273–299 (1965)
18. Combettes, P.L., Wajs, V.R.: Signal recovery by proximal forward–backward splitting. Multiscale Model. Simul. **4**(4), 1168–1200 (2005)
19. Ferreau, H.J., Kirches, C., Potschka, A., Bock, H.G., Diehl, M.: qpOASES: a parametric active-set algorithm for quadratic programming. Math. Program. Comput. **6**(4), 327–363 (2014)
20. Lee, J.D., Sun, Y., Saunders, M.A.: Proximal newton-type methods for minimizing composite functions. SIAM J. Optim. **24**(3), 1420–1443 (2014)
21. van den Berg, E., Friedlander, M.P., Hennenfent, G., Herrmann, F.J., Saab, R., Yılmaz, Ö.: Algorithm 890: sparco—a testing framework for sparse reconstruction. ACM Trans. Math. Softw. **35**(4), 1–16 (2009)
22. Candès, E.J., Tao, T.: Decoding by linear programming. IEEE Trans. Inf. Theory **51**(2), 4203–4215 (2005)
23. Candès, E., Eldar, Y.C., Needell, D., Randall, P.: Compressed sensing with coherent and redundant dictionaries. Appl. Comput. Harmonic Anal. **31**(1), 59–73 (2011)