SIAM J. OPTIM.
Vol. 32, No. 2, pp. 545–572

A SUBSPACE ACCELERATION METHOD FOR MINIMIZATION INVOLVING A GROUP SPARSITY-INDUCING REGULARIZER*

FRANK E. CURTIS†, YUTONG DAI†, AND DANIEL P. ROBINSON†

Abstract. We consider the problem of minimizing an objective function that is the sum of a convex function and a group sparsity-inducing regularizer. Problems that integrate such regularizers arise in modern machine learning applications, often for the purpose of obtaining models that are easier to interpret and that have higher predictive accuracy. We present a new method for solving such problems that utilizes subspace acceleration, domain decomposition, and support identification. Our analysis provides the global iteration complexity of obtaining an ϵ-accurate solution and shows that, under common assumptions, the iterates locally converge superlinearly. Numerical results on regularized logistic and linear regression problems show that our approach is efficient and reliable and outperforms state-of-the-art methods on interesting classes of problems, especially when the number of data points is larger than the number of features. For solving problems when the number of data points is smaller than the number of features, algorithms that focus on solving a dual problem may be more efficient than our approach, which solves the primal problem.

Key words. nonlinear optimization, convex optimization, worst-case iteration complexity, regularization, group regularizer, sparsity, logistic regression, linear regression, subspace acceleration

AMS subject classifications. 49M37, 65K05, 65K10, 65Y20, 68Q25, 90C30, 90C60

DOI. 10.1137/21M1411111

1. Introduction.

We consider the minimization of a function that may be written as the sum of a convex function and a nonoverlapping group sparsity-inducing regularizer. Specifically, given a convex and twice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, a collection of $n_{\mathcal{G}} > 0$ nonoverlapping groups $\mathcal{G} := \{\mathcal{G}_i\}_{i=1}^{n_{\mathcal{G}}}$ that forms a partition of $\{1, 2, \ldots, n\}$ (i.e., $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset$ for all $i \neq j$ and $\cup_{i=1}^{n_{\mathcal{G}}} \mathcal{G}_i = \{1, 2, \ldots, n\}$), and groupwise weighting parameters $\{\lambda_i\}_{i=1}^{n_{\mathcal{G}}} > 0$, our algorithm solves the problem

$$\min_{x \in \mathbb{R}^n} \{f(x) + r(x)\}, \text{ where } r(x) := \sum_{i=1}^{n_{\mathcal{G}}} \lambda_i \, \|[x]_{\mathcal{G}_i}\|_2 \tag{1.1}$$

and $[x]_{\mathcal{G}_i}$ is the subvector of x corresponding to elements in $\mathcal{G}_i$. The regularizer r generalizes the ℓ_1-norm, which is recovered by choosing $\mathcal{G}_i = \{i\}$ for all $i \in \{1, 2, \ldots, n\}$.

Despite the successes of ℓ_1-norm regularization, its inadequacy in the context of many modern machine learning applications has been noticed by researchers and is one motivation for the use of group regularization. In some machine learning applications the covariates come in groups (e.g., genes that regulate hormone levels in microarray data [23]), in which case one may wish to select them jointly. Also, integrating group information into the modeling process can improve both the interpretability and accuracy [36] of the resulting model. Yuan and Lin [35] observed that in the multifactor analysis-of-variance problem, where each factor is expressed through a set

*Received by the editors April 9, 2021; accepted for publication (in revised form) September 26, 2021; published electronically April 27, 2022.
https://doi.org/10.1137/21M1411111
Funding: This material is based upon work supported by the U.S. National Science Foundation under the Division of Computing and Communication Foundations (award CCF-1740796) and the Division of Mathematical Sciences (award DMS-2012243).
†Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015 USA (frank.e.curtis@gmail.com, yud319@lehigh.edu, daniel.p.robinson@gmail.com).

of dummy variables, deleting an irrelevant factor is equivalent to deleting a *group* of dummy variables; the ℓ_1-norm regularizer fails to achieve this goal.

1.1. State-of-the art methods. There is a long history of algorithms for solving regularized problems of the form (1.1) (see [1] and the references therein). Here, we review some of the state-of-the-art approaches for solving sparsity-promoting problems that are most closely related to our proposed approach.

First-order methods. Proximal methods are designed to solve problems of the form (1.1) and have received attention in the machine learning community [3, 8, 32]. A well-known example for ℓ_1-norm regularized problems is the iterative shrinkage-thresholding algorithm (ISTA), which is obtained by applying a proximal gradient (PG) iteration to minimize a smooth function plus the ℓ_1-norm regularizer [11, 13]. Under certain assumptions, one can prove a worst-case complexity bound on the number of iterations required by the PG method before it correctly identifies the support of the optimal solution [28]. Combined with the acceleration technique proposed by Nesterov [26, 27], one obtains the algorithm FISTA [3]. One obtains a related but distinct approach from ISTA by posing an equivalent smooth reformulation of the problem—separating the positive and negative parts of the variables—and applying a gradient projection method to the resulting formulation [14, 15]. All of these approaches have been shown to work well in practice, at least compared to other first-order methods such as the subgradient algorithm. However, these algorithms are often inferior in practice compared to alternative approaches that employ space decomposition techniques and/or second-order derivatives [6, 7, 18].

As an alternative to PG and gradient projection techniques, researchers have considered (block) coordinate descent for solving ℓ_1-norm regularized problems. Such a strategy is appealing, since when minimizing an ℓ_1-norm regularized objective along coordinate directions, it is common that the objective is minimized with variables being zero. These approaches are also easy to implement to exploit parallel computing; see, e.g., the accelerated randomized proximal coordinate gradient method in [20], the parallel coordinate descent methods in [29], and the asynchronous coordinate descent technique in [22]. A downside of these approaches is that the space decomposition is performed in a prescribed manner, rather than in an adaptive way that can benefit from information acquired during the solution process. Also, these approaches do not effectively exploit second-order derivative information and require exact minimization along coordinate directions. An exception to this latter criticism is the inexact coordinate descent algorithm from [30], although this approach does not effectively exploit second-order derivatives and uses a prescribed space decomposition strategy.

Various other approaches have been proposed for solving problems using specific loss functions and/or regularizers. In [21], the authors discuss methods for sparse learning that make use of projection techniques. A well-known package is `GLMNET` [16], which is designed for solving problems with the elastic-net regularization. Finally, let us mention the work in [33], which proposes and tests a groupwise-majorization-descent algorithm (called `gglasso`) for solving problems involving the group ℓ_1-norm regularizer. A potential downside of this approach is that it updates variables by groups in a cycle, rather than by using an adaptive space decomposition technique.

Second-order methods. In [17], an accelerated regularized Newton scheme is proposed. A similar proximal-Newton method is proposed in [19], which (under assumptions) converges locally superlinearly. Although effective in practice, these methods appear to lack good worst-case guarantees in terms of identification of the optimal solution support. Other approaches, such as the orthant-based method in [18],

can predict the solution support but in practice are often outperformed by the related method `FaRSA` [6, 7]. In [31], block-coordinate PG calculations are combined with manifold identification and manifold accelerated calculations (i.e., solving reduced Newton systems). The author analyzed a generic framework, then tested a particular instantiation of the framework designed for ℓ_1-norm regularized problems. Recently, a semismooth Newton method was considered in [37] based on a dual approach for solving (1.1). The semismooth Newton method is used to solve a sequence of augmented Lagrangian problems. Numerical results illustrate the method's efficiency and robustness, although the analysis and algorithm are tailored to the least-squares loss function. As for publicly available solvers based on second-order methods, most have been designed for specific loss functions and regularizers. For example, `newGLMNET` in [34] is designed for ℓ_1-norm regularized logistic regression, and the method in [14] is designed for regularized logistic regression and support vector machines.

Other papers consider stochastic functions and distributed settings, where the evaluation of the (deterministic) gradient is costly or the data is too large to store on a single machine. However, such methods are outside the focus of this paper.

1.2. Contributions. In this paper, we present a framework for solving problem (1.1) that utilizes domain decomposition, support identification, and subspace acceleration. It extends the work in [6, 7], which consider only the traditional ℓ_1-norm regularizer (i.e., not the group ℓ_1-norm case). Although our algorithmic framework is similar to those in these prior papers, the framework proposed in this paper differs in several crucial respects that we now enumerate. (i) Instead of decomposing the domain based on zero and nonzero components of the current iterate as proposed in [6, 7], we partition variables in a way that incorporates the support prediction property of the PG method and tackles the challenge that the gradient of the function being optimized in the reduced space is not Lipschitz continuous. (This challenge is absent in the ℓ_1-norm case). To achieve both goals, a new analysis is performed. (ii) We design a specialized projection procedure for the group ℓ_1-norm regularizer that allows us to prove convergence guarantees and obtain strong numerical performance. This contribution is critical because naive adaptations of the orthant-like projections considered in [6, 7, 18] to the group ℓ_1-norm case would cause the convergence analysis to fail and would lead to abysmal numerical performance. The reason for these failings is that orthant-like projections focus on *individual* variables switching sign during the line search along the Newton-like direction to indicate which variables should be projected to zero. However, the concept of "switching signs" loses its meaning for the *group* ℓ_1-norm case. (One might interpret "switching signs" to mean that a *block* of variables switches from nonzero to zero during the line search along the Newton-like direction, but this generally never happens in theory or in practice.) We address this challenge by projecting to zero groups that are "close enough" to zero, where "close enough" is carefully defined using a ball with radius related to a certain optimality measure that we employ. Moreover, our new projection procedure is designed in a manner that accommodates the domain decomposition approach described above without hindering the global convergence analysis and local superlinear convergence rate. (iii) A worst-case iteration complexity bound is proved along with a simple but principled way of adjusting the PG step size that allows for support identification in finite iterations. Complexity results were not considered in [6, 7]. (iv) Numerical results on regularized logistic and linear regression problems show that our approach is efficient and reliable and outperforms state-of-the-art methods on interesting classes of problems, especially when the number of data points is larger than the number

of features. For solving problems when the number of data points is smaller than the number of features, algorithms such as SSNAL [37] that focus on solving a dual problem may be more efficient than our approach, which solves the primal problem.

1.3. Notation and assumptions. Let $\mathbb{R}$ denote the set of real numbers, $\mathbb{R}^n$ denote the set of n-dimensional real vectors, and $\mathbb{R}^{m\times n}$ denote the set of m-by-n-dimensional real matrices. The set of natural numbers is denoted as $\mathbb{N} := \{0, 1, 2, \dots\}$. For any set $\mathcal{I} \subseteq \{1, 2, \dots, n\}$, we define the projection of $x \in \mathbb{R}^n$ onto the subspace spanned by the coordinate vectors indexed by the entries of $\mathcal{I}$ as $P_{\mathcal{I}}(x)$, so that

$$[P_{\mathcal{I}}(x)]_i := \begin{cases} x_i & \text{if } i \in \mathcal{I}, \\ 0 & \text{if } i \notin \mathcal{I}. \end{cases}$$

For a function $h : \mathbb{R}^n \to \mathbb{R}$, vector $x \in \mathbb{R}^n$, and direction $d \in \mathbb{R}^n$, the directional derivative of h at x in the direction d is denoted by $D_h(x; d)$.

The following assumption is assumed to hold throughout the paper.

Assumption 1.1. The function $f : \mathbb{R}^n \to \mathbb{R}$ used in the definition of the objective function of problem (1.1) is convex and continuously differentiable. Moreover, there exists a positive real number L_f such that $\|\nabla f(x)\|_2 \leq L_f$ for all $x \in \mathcal{L} := \{x \in \mathbb{R}^n : f(x) + r(x) \leq f(x_0) + r(x_0)\}$, where x_0 is a given initial estimate of a solution to problem (1.1). The objective function $f + r$ is bounded below, and the gradient function ∇f is Lipschitz continuous on $\mathcal{L}$ with Lipschitz constant L_g.

2. Preliminaries. In this section, we discuss preliminary material related to the objective function $f + r$ and its associated PG calculations. For any $\bar{x} \in \mathbb{R}^n$ and $\bar{\alpha} > 0$, we define the PG *update* as

$$T(\bar{x}, \bar{\alpha}) := \operatorname*{argmin}_{x\in\mathbb{R}^n} \left\{ \tfrac{1}{2\bar{\alpha}} \|x - \big(\bar{x} - \bar{\alpha}\nabla f(\bar{x})\big)\|_2^2 + r(x) \right\} \tag{2.1}$$

and the associated PG *step* as

$$s(\bar{x}, \bar{\alpha}) := T(\bar{x}, \bar{\alpha}) - \bar{x}. \tag{2.2}$$

The PG update defined in (2.1) can be computed groupwise for each $\mathcal{G}_i \in \mathcal{G}$ by

$$[T(\bar{x}, \bar{\alpha})]_{\mathcal{G}_i} = \max\left\{1 - \frac{\bar{\alpha}\lambda_i}{\|[\bar{x}]_{\mathcal{G}_i} - \bar{\alpha}\nabla_{\mathcal{G}_i} f(\bar{x})\|_2}, 0\right\} \Big([\bar{x}]_{\mathcal{G}_i} - \bar{\alpha}\nabla_{\mathcal{G}_i} f(\bar{x})\Big). \tag{2.3}$$

The next result shows that the directional derivative of $f + r$ along the PG step is negative with magnitude proportional to the squared norm of the PG direction.

LEMMA 2.1. *For any $\bar{x} \in \mathbb{R}^n$ and $\bar{\alpha} > 0$, the PG step $s(\bar{x}, \bar{\alpha})$ in (2.2) satisfies $D_{f+r}(\bar{x}; s(\bar{x}, \bar{\alpha})) \leq -\frac{1}{\bar{\alpha}}\|s(\bar{x}, \bar{\alpha})\|_2^2$. Moreover, if $\mathcal{I}$ is equal to the union of a subset of $\{\mathcal{G}_i\}_{i=1}^{n_{\mathcal{G}}}$, then $D_{f+r}\big(\bar{x}; P_{\mathcal{I}}(s(\bar{x}, \bar{\alpha}))\big) \leq -\frac{1}{\bar{\alpha}}\|P_{\mathcal{I}}(s(\bar{x}, \bar{\alpha}))\|_2^2$.*

Proof. Let $x_+ = T(\bar{x}, \bar{\alpha})$ denote the PG update in (2.1) so that $x_+ = \bar{x} + s(\bar{x}, \bar{\alpha})$. The optimality conditions for the problem in (2.1) give some $g_+ \in \partial r(x_+)$ such that

$$x_+ - \bar{x} + \bar{\alpha}\nabla f(\bar{x}) + \bar{\alpha} g_+ = 0. \tag{2.4}$$

Next, for an arbitrary $g_{f+r} \in \partial(f + r)(\bar{x})$, it follows from Assumption 1.1 and [5, Proposition 5.4.6] that there exits $g_r \in \partial r(\bar{x})$ satisfying $g_{f+r} = \nabla f(\bar{x}) + g_r$. From the definitions of g_r and g_+ and convexity of r, it follows that $r(x_+) \geq r(\bar{x}) + g_r^T(x_+ - \bar{x})$ and $r(\bar{x}) \geq r(x_+) + g_+^T(\bar{x} - x_+)$. Adding these equations yields $(g_r - g_+)^T(x_+ - \bar{x}) \leq 0$,

which when combined with the definition of g_{f+r} and (2.4) yields $s(\overline{x},\overline{\alpha})^T g_{f+r} = (x_+ - \overline{x})^T(\nabla f(\overline{x}) + g_r) = \frac{1}{\overline{\alpha}}(x_+ - \overline{x})^T(\overline{x} - x_+ - \overline{\alpha}g_+ + \overline{\alpha}g_r) = -\frac{1}{\overline{\alpha}}\|x_+ - \overline{x}\|_2^2 + (x_+ - \overline{x})^T(g_r - g_+) \leq -\frac{1}{\overline{\alpha}}\|s(\overline{x},\overline{\alpha})\|_2^2$. Since $g_{f+r} \in \partial(f+r)(\overline{x})$ was arbitrary, [25, Theorem 2.87] and the previous string of inequalities together yield $D_{f+r}(\overline{x}; s(\overline{x},\overline{\alpha})) = \sup_{g\in\partial(f+r)(\overline{x})} s(\overline{x},\overline{\alpha})^T g \leq -\frac{1}{\overline{\alpha}}\|s(\overline{x},\overline{\alpha})\|_2^2$, as claimed. The final conclusion in the lemma follows using the same argument, but restricting the quantities to $\mathcal{I}$. ∎

Next, we quantify the decrease in $f+r$ obtained by taking a PG step $s(\overline{x},\overline{\alpha})$, provided the PG parameter $\overline{\alpha}$ is sufficiently small. The proof for the case $\mathcal{I} = \{1, 2, \ldots, n\}$ is found in [2, Lemma 10.4], and the proof for the general case, i.e., when $\mathcal{I}$ is equal to the union of a subset of $\{\mathcal{G}_i\}_{i=1}^{n_\mathcal{G}}$, follows using the same logic as in the proof of [2, Lemma 10.4] but with straightforward modifications to handle the definition of $\mathcal{I}$.

LEMMA 2.2. *If $\overline{x} \in \mathbb{R}^n$, $\overline{\alpha} \in (0, 2/L)$, and $\mathcal{I}$ is equal to the union of a subset of $\{\mathcal{G}_i\}_{i=1}^{n_\mathcal{G}}$, then $f(\overline{x}+P_\mathcal{I}(\overline{x},\bar{s}))+r(\overline{x}+P_\mathcal{I}(\overline{x},\bar{s})) \leq f(\overline{x})+r(\overline{x})-(\frac{1}{\overline{\alpha}}-\frac{L}{2})\|P_\mathcal{I}\big(s(\overline{x},\overline{\alpha})\big)\|_2^2$.*

The next result shows that, when restricted to certain groups, the size of the PG step is bounded above by the gradient of the objective function.

LEMMA 2.3. *If the pair $(\overline{x},\overline{\alpha})$ and group $\mathcal{G}_i$ satisfy $\overline{\alpha} \in (0, 1]$, $[\overline{x}]_{\mathcal{G}_i} \neq 0$, and $[\overline{x} + s(\overline{x},\overline{\alpha})]_{\mathcal{G}_i} \neq 0$, where $s(\overline{x},\overline{\alpha})$ is defined in (2.2), then $\|\nabla_{\mathcal{G}_i}(f+r)(\overline{x})\|_2 \geq \|[s(\overline{x},\overline{\alpha})]_{\mathcal{G}_i}\|_2$.*

Proof. Denote $g_i := \nabla_{\mathcal{G}_i} f(\overline{x})$, $x_i = [\overline{x}]_{\mathcal{G}_i}$, and $s_i = [s(\overline{x},\overline{\alpha})]_{\mathcal{G}_i}$. Since $f + r$ is differentiable with respect to variables in $\mathcal{G}_i$ at $\overline{x}$ since $[\overline{x}]_{\mathcal{G}_i} \neq 0$, we have $\|\nabla_{\mathcal{G}_i}(f + r)(\overline{x})\|_2^2 = \|g_i + \lambda_i x_i/\|x_i\|_2\|_2^2 = \|g_i\|_2^2 + 2\lambda_i\frac{g_i^T x_i}{\|x_i\|_2} + \lambda_i^2$, so it is sufficient to prove that $\|g_i\|_2^2 + 2\lambda_i\frac{g_i^T x_i}{\|x_i\|_2} + \lambda_i^2 \geq \|s_i\|_2^2$. Since $x_i + s_i \neq 0$ by assumption, s_i (see (2.3)) satisfies $s_i = \big(1 - \frac{\overline{\alpha}\lambda_i}{\|x_i - \overline{\alpha}g_i\|_2}\big)(x_i - \overline{\alpha}g_i) - x_i = x_i - \overline{\alpha}g_i - \frac{\overline{\alpha}\lambda_i(x_i - \overline{\alpha}g_i)}{\|x_i - \overline{\alpha}g_i\|_2} - x_i = -\overline{\alpha}\big(g_i + \frac{\lambda_i(x_i - \overline{\alpha}g_i)}{\|x_i - \overline{\alpha}g_i\|_2}\big)$ so that $\|s_i\|_2^2 = \overline{\alpha}^2\big(\|g_i\|_2^2 + 2\lambda_i\frac{g_i^T(x_i - \overline{\alpha}g_i)}{\|x_i - \overline{\alpha}g_i\|_2} + \lambda_i^2\big)$. Thus, it is sufficient to prove that $\|g_i\|_2^2 + 2\lambda_i\frac{g_i^T x_i}{\|x_i\|_2} + \lambda_i^2 \geq \overline{\alpha}^2\big(\|g_i\|_2^2 + 2\lambda_i\frac{g_i^T(x_i - \overline{\alpha}g_i)}{\|x_i - \overline{\alpha}g_i\|_2} + \lambda_i^2\big)$. We consider two cases and note that $x_i \neq 0$ by assumption and that $x_i - \overline{\alpha}g_i \neq 0$ as a consequence of (2.3) and the assumption that $x_i + s_i \neq 0$.

Case 1: $\overline{\alpha} = 1$. In this case, the desired inequality simplifies to

$$\frac{g_i^T x_i}{\|x_i\|_2} \geq \frac{g_i^T(x_i - g_i)}{\|x_i - g_i\|_2}. \tag{2.5}$$

We now consider the following two subcases.

Case 1a: $g_i^T x_i \geq 0$. The desired inequality clearly holds if $g_i^T(x_i - g_i) \leq 0$. Thus, for the remainder of this subcase, we assume that $g_i^T(x_i - g_i) > 0$, which equivalently means that $g_i^T x_i > \|g_i\|_2^2$, which implies that $-2x_i^T g_i + \|g_i\|_2^2 < 0$. It follows from this inequality and the fact that $(g_i^T x_i)^2 \leq \|g_i\|_2^2\|x_i\|_2^2$ (by Cauchy–Schwarz) that $(g_i^T x_i)^2(-2x_i^T g_i + \|g_i\|_2^2) \geq (-2x_i^T g_i + \|g_i\|_2^2)\|g_i\|_2^2\|x_i\|_2^2 = \big(\|g_i\|_2^4 - 2g_i^T x_i\|g_i\|_2^2\big)\|x_i\|_2^2$. We can now add the term $(g_i^T x_i)^2\|x_i\|_2^2$ to both sides to obtain $(g_i^T x_i)^2(\|x_i\|_2^2 - 2x_i^T g_i + \|g_i\|_2^2) \geq \big((g_i^T x_i)^2 + \|g_i\|_2^4 - 2g_i^T x_i\|g_i\|_2^2\big)\|x_i\|_2^2$, which can be written equivalently as $(g_i^T x_i)^2\|x_i - g_i\|_2^2 \geq (g_i^T x_i - \|g_i\|_2^2)^2\|x_i\|_2^2 = (g_i^T(x_i - g_i))^2\|x_i\|_2^2$. After taking the square root of both sides, we obtain (2.5).

Case 1b: $g_i^T x_i < 0$. Using $g_i^T x_i < 0$ and $(g_i^T x_i)^2 \leq \|g_i\|_2^2\|x_i\|_2^2$ together implies that $(g_i^T x_i)^2(-2x_i^T g_i + \|g_i\|_2^2) \leq (-2x_i^T g_i + \|g_i\|_2^2)\|g_i\|_2^2\|x_i\|_2^2 = \big(\|g_i\|_2^4 - 2g_i^T x_i\|g_i\|_2^2\big)\|x_i\|_2^2$. We can now add the term $(g_i^T x_i)^2\|x_i\|_2^2$ to both sides to obtain $(g_i^T x_i)^2(\|x_i\|_2^2 - 2x_i^T g_i + \|g_i\|_2^2) \leq \big((g_i^T x_i)^2 + \|g_i\|_2^4 - 2g_i^T x_i\|g_i\|_2^2\big)\|x_i\|_2^2$, which can be written equivalently as $(g_i^T x_i)^2\|x_i - g_i\|_2^2 \leq (g_i^T x_i - \|g_i\|_2^2)^2\|x_i\|_2^2 = (g_i^T(x_i - g_i))^2\|x_i\|_2^2$. After taking the square root of both sides and rearranging, we obtain $|g_i^T x_i|/\|x_i\|_2 \leq$

Algorithm 3.1. `FaRSA-Group` for solving problem (1.1).

1: **Input:** x_0
2: **Constants:** $\{\varphi, \xi, \eta, \zeta\} \subset (0,1)$, $\{\kappa_1, \kappa_2, p\} \subset (0,\infty)$, $\theta \in (0, \pi/2)$, and $q \in [1,2]$.
3: Choose any initial PG parameter $\alpha_0 \in (0,1]$.
4: **for** $k = 0, 1, 2, \ldots$ **do**
5: Compute the step s_k from (3.1) and the set $\bar{\mathcal{I}}_k^{\mathrm{m}}$ from (3.2).
6: Compute $\mathcal{I}_k^{\mathrm{m}}$ and $\mathcal{I}_k^{\mathrm{pg}}$ and their optimality measures χ_k^{m} and χ_k^{pg} from (3.4).
7: **if** $\chi_k^{\mathrm{pg}} \leq \chi_k^{\mathrm{m}}$ **then**
8: Choose any $\mathcal{I}_k \subseteq \mathcal{I}_k^{\mathrm{m}}$ such that

$$\|[s_k]_{\mathcal{I}_k}\|_2 \geq \varphi \|[s_k]_{\mathcal{I}_k^{\mathrm{m}}}\|_2 \equiv \varphi \chi_k^{\mathrm{m}} \text{ and } \mathcal{I}_k \text{ is the union of some } \{\mathcal{G}_j\}.$$

9: Set $g_k \leftarrow \nabla_{\mathcal{I}_k}(f+r)(x_k)$ and pick a positive-definite $H_k \in \mathbb{R}^{|\mathcal{I}_k| \times |\mathcal{I}_k|}$.
10: Call Algorithm 3.2 to obtain $\bar{d}_k \leftarrow$ `m_direction`(g_k, H_k).
11: Set $[d_k]_{\mathcal{I}_k} \leftarrow \bar{d}_k$ and $[d_k]_{\mathcal{I}_k^c} \leftarrow 0$.
12: Call Algorithm 3.3 to obtain $(x_{k+1}, \mathrm{flag}_k^{\mathrm{m}}) \leftarrow$ `m_update`$(x_k, d_k, \mathcal{I}_k)$.
13: Set $\alpha_{k+1} \leftarrow \alpha_k$.
14: **else**
15: Choose any $\mathcal{I}_k \subseteq \mathcal{I}_k^{\mathrm{pg}}$ such that

$$\|[s_k]_{\mathcal{I}_k}\|_2 \geq \varphi \|[s_k]_{\mathcal{I}_k^{\mathrm{pg}}}\|_2 \equiv \varphi \chi_k^{\mathrm{pg}} \text{ and } \mathcal{I}_k \text{ is the union of some } \{\mathcal{G}_j\}.$$

16: Call Algorithm 3.4 to obtain $(x_{k+1}, \mathrm{flag}_k^{\mathrm{pg}}) \leftarrow$ `pg_update`$(x_k, s_k, \alpha_k, \mathcal{I}_k)$.
17: **if** $\mathrm{flag}_k^{\mathrm{pg}} =$ `decrease_`α **then** $\alpha_{k+1} \leftarrow \zeta \alpha_k$ **else** $\alpha_{k+1} \leftarrow \alpha_k$.

$|g_i^T(x_i - g_i)|/\|x_i - g_i\|_2$. Combining this result with $0 > g_i^T x_i \geq g_i^T(x_i - g_i)$ gives (2.5), as claimed.

Case 2: $\overline{\alpha} \in (0,1)$. The proof of follows from Case 1 and [2, Theorem 10.9], which in our notation from (2.2) proves that $\|s(\overline{x}, \overline{\alpha})\|_2 \leq \|s(\overline{x}, 1)\|_2$ when $\overline{\alpha} \in (0,1)$. ∎

3. Proposed algorithm framework. We propose Algorithm 3.1, which we call `FaRSA-Group` (`F`ast `R`educed-Space `A`lgorithm for `Group` sparsity-inducing regularization), for solving problem (1.1) that uses ideas related to domain decomposition, subspace acceleration, and support identification. An overview of the algorithm is given in section 3.1. During each iteration of our method, at least one of three subroutines is called. The three subroutines are described in sections 3.2–3.4.

3.1. Main algorithm (Algorithm 3.1). Our main algorithm is formally stated as Algorithm 3.1. At the beginning of the kth iteration, x_k and $\alpha_k > 0$ denote the current solution estimate for problem (1.1) and the PG parameter, respectively. We then compute s_k in line 5 as the PG step associated with problem (1.1), namely,

$$s_k := s(x_k, \alpha_k) \quad \text{with} \quad s(x_k, \alpha_k) \text{ defined in (2.2).} \tag{3.1}$$

Although the repeated computation of PG steps is the basis for a first-order method, here we primarily use it to *predict* the zero/nonzero structure of a solution and to formulate optimality measures. Specifically, in line 5 we compute the index set

$$\bar{\mathcal{I}}_k^{\mathrm{m}} := \{j \in \mathcal{G}_i : [x_k]_{\mathcal{G}_i} \neq 0,\ [x_k + s_k]_{\mathcal{G}_i} \neq 0,\ \|[x_k]_{\mathcal{G}_i}\|_2 \geq \kappa_1 \|\nabla_{\mathcal{G}_i}(f+r)(x_k)\|_2\} \tag{3.2}$$

for some $\kappa_1 \in (0,\infty)$. The groups of variables that compose $\bar{\mathcal{I}}_k^{\mathrm{m}}$ are *candidates* for use in a Newton-type calculation aimed to accelerate convergence. Before using them,

however, we first check to see if each candidate block is sufficiently far from zero, and those that are not are removed. Specifically, we first define

$$\mathcal{I}_k^{\text{small}} := \{j \in \mathcal{G}_i : \mathcal{G}_i \subseteq \bar{\mathcal{I}}_k^{\text{m}} \text{ and } \|[x_k]_{\mathcal{G}_i}\|_2 < \kappa_2 \|\nabla_{\bar{\mathcal{I}}_k^{\text{m}}}(f+r)(x_k)\|_2^p\} \tag{3.3}$$

for some $\{\kappa_2, p\} \subset (0, \infty)$, then define in line 6 the sets and optimality measures

$$\left.\begin{array}{l} \mathcal{I}_k^{\text{m}} := \bar{\mathcal{I}}_k^{\text{m}} \setminus \mathcal{I}_k^{\text{small}} \\ \mathcal{I}_k^{\text{pg}} := \{1, 2, \ldots, n\} \setminus \mathcal{I}_k^{\text{m}} \end{array}\right\} \quad \text{and} \quad \left\{\begin{array}{l} \chi_k^{\text{m}} := \|[s_k]_{\mathcal{I}_k^{\text{m}}}\|_2 \\ \chi_k^{\text{pg}} := \|[s_k]_{\mathcal{I}_k^{\text{pg}}}\|_2 \end{array}\right\}, \tag{3.4}$$

where by convention $\|[\,\cdot\,]_\emptyset\|_2 = 0$. (See Lemma 4.1 for a justification that these sets together represent a measure of optimality.) This construction of sets also ensures that the subvector of x_k that corresponds to $\mathcal{G}_i$ for each $\mathcal{G}_i \subseteq \mathcal{I}_k^{\text{m}}$ is at least a distance

$$\rho_{k,i} := \max\{\kappa_1 \|\nabla_{\mathcal{G}_i}(f+r)(x_k)\|_2, \kappa_2 \|\nabla_{\mathcal{I}_k^{\text{m}}}(f+r)(x_k)\|_2^p\} \tag{3.5}$$

away from zero (see Lemma 4.5(i)), which is crucial in our analysis.

Armed with χ_k^{pg} and χ_k^{m}, Algorithm 3.1 seeks decrease in the objective function in a subspace that is likely to allow for significant progress. We consider two cases.

Case 1: the condition $\chi_k^{\text{pg}} \le \chi_k^{\text{m}}$ checked in line 7 holds. In this case, the inequality $\chi_k^{\text{pg}} \le \chi_k^{\text{m}}$ indicates that significant reduction in the objective function can be achieved by focusing on variables in the set $\mathcal{I}_k^{\text{m}}$. Therefore, in line 8 we choose any index set $\mathcal{I}_k$ that is (i) a subset of $\mathcal{I}_k^{\text{m}}$, (ii) equal to the union of some subset of groups from $\mathcal{G}$, and (iii) the size of the PG step restricted to the index set $\mathcal{I}_k$ is at least a fraction of the size of the PG step when restricted to the index set $\mathcal{I}_k^{\text{m}}$. The easiest choice that satisfies these conditions is $\mathcal{I}_k \equiv \mathcal{I}_k^{\text{m}}$, but for large-scale problems it may be beneficial to restrict $|\mathcal{I}_k|$. The opposite extreme choice is selecting $\mathcal{I}_k$ as the group $\mathcal{G}_i$ contained in $\mathcal{I}_k^{\text{m}}$ with largest associated PG step, in which case one would choose $\varphi = 1/\sqrt{n_{\mathcal{G}}}$ for the user-defined parameter in line 8. Once $\mathcal{I}_k$ has been selected, a *reduced-space* gradient g_k and *reduced-space* positive-definite matrix H_k are defined in line 9, where the derivatives are taken with respect to variables in $\mathcal{I}_k$. (In practice, H_k could be selected based on $\nabla^2_{\mathcal{I}_k\mathcal{I}_k}(f+r)(x_k)$ to promote a fast local convergence rate.) Note that g_k exists since by construction $\mathcal{I}_k \subseteq \mathcal{I}_k^{\text{m}} \subseteq \bar{\mathcal{I}}_k^{\text{m}}$, and from (3.2) the objective function $f+r$ is differentiable with respect to groups of variables in $\bar{\mathcal{I}}_k^{\text{m}}$. Next, g_k and H_k are used to compute a direction $\bar{d}_k$ of sufficient descent for $f+r$ by calling the subroutine `m_direction` (see section 3.2). Once a full-space vector d_k is obtained by padding $\bar{d}_k$ with zeros in line 11, a *projected* line search is performed by calling subroutine `m_update` in line 12 (see section 3.3).

Case 2: the condition $\chi_k^{\text{pg}} \le \chi_k^{\text{m}}$ checked in line 7 does not hold. In this case, the inequality $\chi_k^{\text{pg}} > \chi_k^{\text{m}}$ indicates that significant reduction in the objective function can be achieved by focusing on variables in the set $\mathcal{I}_k^{\text{pg}}$. Therefore, in line 15, we choose any index set $\mathcal{I}_k$ that is (i) a subset of $\mathcal{I}_k^{\text{pg}}$, (ii) equal to the union of some subset of groups from $\mathcal{G}$, and (iii) the size of the PG step restricted to the index set $\mathcal{I}_k$ is at least a fraction of the size of the PG step restricted to the index set $\mathcal{I}_k^{\text{pg}}$. The easiest choice that satisfies these conditions is $\mathcal{I}_k \equiv \mathcal{I}_k^{\text{pg}}$. Once $\mathcal{I}_k$ has been chosen, the next iterate is obtained by performing a line search along the PG direction in line 16 by calling the subroutine `pg_update` (for details, see section 3.4). If the subroutine returns $\text{flag}_k^{\text{pg}} =$ `decrease_`α, the PG parameter is decreased for the next iteration.

3.2. Computing the m-direction (Algorithm 3.2). This subroutine returns a reduced-space direction $\bar{d}_k$ that satisfies the conditions in line 22. We call it

a reduced-space vector because the inputs g_k and H_k are elements in $\mathbb{R}^{|\mathcal{I}_k|}$ and $\mathbb{R}^{|\mathcal{I}_k|\times|\mathcal{I}_k|}$, respectively, where $\mathcal{I}_k$ is computed in line 8 of Algorithm 3.1. The first condition in line 22 ensures that $\bar{d}_k$ is a descent direction for the objective function as a consequence of how the reference direction d_k^R is computed in line 21. The second condition in line 22 ensures that $\bar{d}_k$ reduces the model m_k at least as much as a zero step. Finally, the third condition in line 22 promotes fast *local* convergence of the iterate sequence $\{x_k\}$ (see section 4.2), but its enforcement (or lack of enforcement) is irrelevant with respect to the complexity result that we prove in section 4.1. In our numerical implementation we apply the linear CG algorithm to the system $H_k d = -g_k$ associated with the model m_k in line 20, although other options include a blockwise coordinate descent method applied to the model m_k. In particular, the direction associated with every iteration of the CG algorithm satisfies the first two conditions in line 22, and the third condition in line 22 is satisfied by all sufficiently large CG iterations. (The fact that the first condition in line 22 holds for every iteration within CG is not a commonly mentioned result, but it follows from the updates that define CG.) Thus, the requirements of this subroutine can always be met.

Algorithm 3.2. Computing $\bar{d}_k$ in line 10 of Algorithm 3.1.

18: **procedure** $\bar{d}_k = \texttt{m\_direction}(g_k, H_k)$
19: **Constant:** q is provided by Algorithm 3.1.
20: Define the model $m_k(d) := g_k^T d + \frac{1}{2} d^T H_k d$.
21: Compute the reference direction (an approximate minimizer of m_k) as

$$d_k^R \leftarrow -\beta_k g_k, \text{ where } \beta_k \leftarrow \|g_k\|_2^2/(g_k^T H_k g_k).$$

22: Choose $\mu_k \in (0, 1]$ and then compute any $\bar{d}_k \approx \operatorname{argmin}_d m_k(d)$ that satisfies

$$g_k^T \bar{d}_k \leq g_k^T d_k^R, \quad m_k(\bar{d}_k) \leq m_k(0), \quad \text{and} \quad \|H_k \bar{d}_k + g_k\|_2 \leq \mu_k \|g_k\|_2^q.$$

23: **return** $\bar{d}_k$

3.3. Reduced-space search using an m-direction (Algorithm 3.3). This subroutine searches along the direction d_k returned by the subroutine `m_direction` in line 10 of Algorithm 3.1. For an illustration of this search, which incorporates projections, see Figure 3.1. The approach uses the direction d_k, without modification, for each block of variables $\mathcal{G}_i$ such that the ray $\{[x_k + \tau d_k]_{\mathcal{G}_i} : \tau \geq 0\}$ does not intersect the ball centered at zero of radius $\bar{\rho}_{k,i} = \min\{\rho_{k,i}, \sin(\theta)\|[x_k]_{\mathcal{G}_i}\|_2\}$, where $\rho_{k,i}$ is defined in (3.5) and $\theta \in (0, \pi/2)$ is a user-defined parameter. When they do intersect, we first compute $\tau_{k,i}$ as the smallest step along d_k (restricted to block $\mathcal{G}_i$) that intersects the ball. Then, during the search that follows, any time the trial step size ξ^j is larger than $\tau_{k,i}$, the trial step for block $\mathcal{G}_i$ is set to zero; otherwise, d_k is used so that the trial step (with respect to block $\mathcal{G}_i$) is $[x_k + \xi^j d_k]_{\mathcal{G}_i}$ (see line 37). If termination occurs in line 38, a new block of variables will become zero, in which case we require the objective function not to increase (see line 39). On the other hand, if termination occurs in line 44, it indicates that the objective function has been sufficiently reduced (see line 43) and no new groups of zeros have been formed.

3.4. Reduced-space line search along a PG direction (Algorithm 3.4). This subroutine performs a line search along the PG direction $P_{\mathcal{I}}(s_k)$. The search ensures that the step yields decrease in the objective of at least $(\eta\xi^j/\alpha_k)\|P_{\mathcal{I}_k}(s_k)\|_2^2$

Algorithm 3.3. Computing x_{k+1} in line 12 of Algorithm 3.1.

24: **procedure** $(x_{k+1}, \text{flag}_k^{\text{m}}) = \texttt{m\_update}(x_k, d_k, \mathcal{I}_k)$
25: **Constants:** η, ξ, and θ provided by Algorithm 3.1.
26: **for** each i such that $\mathcal{G}_i \subseteq \mathcal{I}_k$ **do**
27: Compute $\rho_{k,i}$ as defined in (3.5).
28: Set $\bar{\rho}_{k,i} \leftarrow \min\{\rho_{k,i}, \sin(\theta)\|[x_k]_{\mathcal{G}_i}\|_2\}$.
29: **if** $\{[x_k + \tau d_k]_{\mathcal{G}_i} : \tau \geq 0\} \cap \{x \in \mathbb{R}^{|\mathcal{G}_i|} : \|x\|_2 \leq \bar{\rho}_{k,i}\} = \emptyset$ **then**
30: Set $\tau_{k,i} \leftarrow \infty$.
31: **else**
32: Set $\tau_{k,i}$ as the smallest positive root of $\|[x_k + \tau d_k]_{\mathcal{G}_i}\|_2 = \bar{\rho}_{k,i}$.
33: Set $j \leftarrow 0$ and $\tau_k := \min_i\{\tau_{k,i} : \mathcal{G}_i \subseteq \mathcal{I}_k\}$.
34: **while** $\xi^j \geq \tau_k$ **do**
35: Set $[y_j]_{\mathcal{I}_k^c} \leftarrow [x_k]_{\mathcal{I}_k^c}$.
36: **for** each i such that $\mathcal{G}_i \subseteq \mathcal{I}_k$ **do**
37: Set $[y_j]_{\mathcal{G}_i} \leftarrow \begin{cases} [x_k]_{\mathcal{G}_i} + \xi^j [d_k]_{\mathcal{G}_i} & \text{if } \xi^j < \tau_{k,i}, \\ 0 & \text{if } \xi^j \geq \tau_{k,i}. \end{cases}$
38: **if** $f(y_j) + r(y_j) \leq f(x_k) + r(x_k)$ **then**
39: **return** $x_{k+1} \leftarrow y_j$ and $\text{flag}_k^{\text{m}} \leftarrow \texttt{new\_zero}$
40: Set $j \leftarrow j + 1$.
41: **loop**
42: Set $y_j \leftarrow x_k + \xi^j d_k$.
43: **if** $f(y_j) + r(y_j) \leq f(x_k) + r(x_k) + \eta \xi^j \nabla_{\mathcal{I}_k}(f + r)(x_k)^T [d_k]_{\mathcal{I}_k}$ **then**
44: **return** $x_{k+1} \leftarrow y_j$ and $\text{flag}_k^{\text{m}} \leftarrow \texttt{suff\_descent}$
45: Set $j \leftarrow j + 1$.

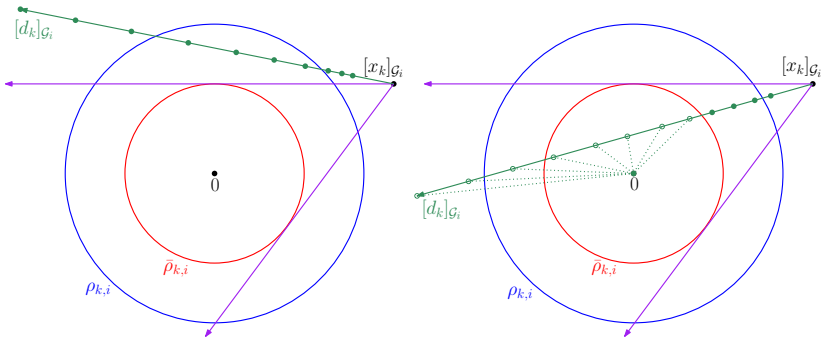

FIG. 3.1. *The reduced-space projected search based on the m-direction d_k described in section 3.3. In the figure on the left, the direction d_k does not intersect the ball of radius $\bar{\rho}_{k,i}$. In this case, standard backtracking is used, as indicated by the solid green dots. In the figure on the right, the direction d_k does intersect the ball of radius $\bar{\rho}_{k,i}$. In this case, all points after the first point of intersection (indicated by hollow green circles) are projected to zero. Once the backtracking points leave the ball of radius $\bar{\rho}_{k,i}$ (indicated as solid green dots), standard backtracking is resumed.*

for some positive integer j computed within the while loop in line 49. Once the while loop terminates, the update $\text{flag}_k^{\text{pg}} \leftarrow \texttt{same\_}\alpha$ is made if $j = 0$ and set as $\text{flag}_k^{\text{pg}} \leftarrow \texttt{decrease\_}\alpha$ otherwise. The motivation for this update is Lemma 2.2, which shows that the while loop in line 49 will terminate with $j = 0$ if the PG parameter

α_k is sufficiently small. Therefore, any time $j > 0$, Algorithm 3.4 returns $\text{flag}_k^{\text{pg}} \leftarrow$ `decrease_`α to Algorithm 3.1 in line 16 so that the PG parameter value for the next iteration is reduced by a factor of $\xi \in (0, 1)$ in line 17.

Algorithm 3.4. Computing x_{k+1} in line 16 of Algorithm 3.1.

46: **procedure** $(x_{k+1}, \text{flag}_k^{\text{pg}}) =$ `pg_update`$(x_k, s_k, \alpha_k, \mathcal{I}_k)$
47: **Constants:** η and ξ provided by Algorithm 3.1.
48: Set $j \leftarrow 0$ and $y_0 \leftarrow x_k + P_{\mathcal{I}_k}(s_k)$.
49: **while** $f(y_j) + r(y_j) > f(x_k) + r(x_k) - \eta\xi^j \frac{1}{\alpha_k}\|P_{\mathcal{I}_k}(s_k)\|_2^2$ **do**
50: Set $j \leftarrow j + 1$ and then $y_j \leftarrow x_k + \xi^j P_{\mathcal{I}_k}(s_k)$.
51: **if** $j = 0$ **then**
52: **return** $x_{k+1} \leftarrow y_j$ and $\text{flag}_k^{\text{pg}} \leftarrow$ `same_`α
53: **else**
54: **return** $x_{k+1} \leftarrow y_j$ and $\text{flag}_k^{\text{pg}} \leftarrow$ `decrease_`α

4. Analysis. Our analysis considers worst-case complexity (section 4.1) and local convergence (section 4.2) properties of Algorithm 3.1. To identify an approximate solution to problem (1.1), we use the measure $\max\{\chi_k^{\text{pg}}, \chi_k^{\text{m}}\}$, as we now justify.

LEMMA 4.1. *Let* $\mathcal{K} \subseteq \mathbb{N}$ *be such that* $\lim_{k\in\mathcal{K}} x_k = x_*$ *and* $\lim_{k\in\mathcal{K}} \alpha_k = \alpha_* > 0$. *Then,* x_* *is a solution to problem* (1.1) *if and only if* $\lim_{k\in\mathcal{K}} \max\{\chi_k^{\text{pg}}, \chi_k^{\text{m}}\} = 0$.

Proof. First, we may apply [9, Theorem 3.2.8], with the choice $y = (\overline{x}, \overline{\alpha})$ and the set map $\mathcal{C}(y) = \mathbb{R}^n$, to the objective function appearing in (2.1) to conclude that $T(\overline{x}, \overline{\alpha})$ is continuous on $\mathbb{R}^n \times (0, \infty)$. Combining this property with the definition of T in (2.1) and the assumption that $\lim_{k\in\mathcal{K}}(x_k, \alpha_k) = (x_*, \alpha_*)$ with $\alpha_* > 0$ shows that $\lim_{k\in\mathcal{K}} s_k = \lim_{k\in\mathcal{K}} \big(T(x_k, \alpha_k) - x_k\big) = T(x_*, \alpha_*) - x_*$. It follows from this limit and the fact that Assumption 1.1 and [2, Theorem 10.7] together show that x_* is a solution to problem (1.1) if and only if $T(x_*, \alpha_*) = x_*$. ∎

Suppose that $\max\{\chi_k^{\text{m}}, \chi_k^{\text{pg}}\} = 0$ for some $k \in \mathbb{N}$. By defining the sequences $\{x_j\}$ and $\{\alpha_j\}$ such that $x_j = x_k$ and $\alpha_j = \alpha_k$ for all $j \geq 1$, we may apply Lemma 4.1 (with k replaced by j) to conclude that x_k is a solution to problem (1.1). Hence, for the remainder of this section, we make the following assumption.

Assumption 4.1. For all iterations $k \in \mathbb{N}$, it holds that $\max\{\chi_k^{\text{m}}, \chi_k^{\text{pg}}\} > 0$.

Since our analysis considers the properties of the sequence of iterates, it is convenient to define the following partition of iterations performed by Algorithm 3.1:

$$\begin{aligned}
\mathcal{K}^{\text{m}} &:= \{k \in \mathbb{N} : \text{line 12 is reached during the } k\text{th iteration}\},\\
\mathcal{K}_0^{\text{m}} &:= \{k \in \mathcal{K}^{\text{m}} : \text{subroutine } \texttt{m_update} \text{ returns } \text{flag}_k^{\text{m}} = \texttt{new_zero} \text{ in line 12}\},\\
\mathcal{K}_{\text{sd}}^{\text{m}} &:= \{k \in \mathcal{K}^{\text{m}} : \text{subroutine } \texttt{m_update} \text{ returns } \text{flag}_k^{\text{m}} = \texttt{suff_descent} \text{ in line 12}\},\\
\mathcal{K}^{\text{pg}} &:= \{k \in \mathbb{N} : \text{line 16 is reached during the } k\text{th iteration}\},\\
\mathcal{K}_{\rightarrow}^{\text{pg}} &:= \{k \in \mathcal{K}^{\text{pg}} : \text{subroutine } \texttt{pg_update} \text{ returns } \text{flag}_k^{\text{pg}} = \texttt{same_}\alpha \text{ in line 16}\}, \text{ and}\\
\mathcal{K}_{\downarrow}^{\text{pg}} &:= \{k \in \mathcal{K}^{\text{pg}} : \text{subroutine } \texttt{pg_update} \text{ returns } \text{flag}_k^{\text{pg}} = \texttt{decrease_}\alpha \text{ in line 16}\},
\end{aligned}$$

so that $\mathcal{K}^{\text{m}} = \mathcal{K}_0^{\text{m}} \cup \mathcal{K}_{\text{sd}}^{\text{m}}$, $\mathcal{K}^{\text{pg}} = \mathcal{K}_{\rightarrow}^{\text{pg}} \cup \mathcal{K}_{\downarrow}^{\text{pg}}$, and $\mathbb{N} = \mathcal{K}^{\text{m}} \cup \mathcal{K}^{\text{pg}}$.

Finally, we assume that the symmetric and positive-definite matrices required in line 9 are chosen to be bounded and uniformly positive definite.

Assumption 4.2. The matrix sequence $\{H_k\}_{k\in\mathcal{K}^{\mathrm{m}}}$ chosen in line 9 is bounded and uniformly positive definite. That is, there exist constants $0 < \mu_{\min} \le \mu_{\max} < \infty$ such that $\mu_{\min}\|v\|_2^2 \le v^T H_k v \le \mu_{\max}\|v\|_2^2$ for all $k \in \mathcal{K}^{\mathrm{m}}$ and $v \in \mathbb{R}^{|\mathcal{I}_k|}$.

4.1. Complexity result. We first focus our attention on iterations in $\mathcal{K}^{\mathrm{pg}}$. The next result shows that Algorithm 3.4 is well posed and that the new iterate that it produces satisfies a decrease property that will be useful for our complexity analysis.

LEMMA 4.2. *For each $k \in \mathcal{K}^{\mathrm{pg}}$, Algorithm* 3.4 *is called in line* 16 *and successfully returns x_{k+1} and $\mathrm{flag}_k^{\mathrm{pg}}$. Moreover, the value of $\mathrm{flag}_k^{\mathrm{pg}}$ indicates whether $k \in \mathcal{K}_{\downarrow}^{\mathrm{pg}}$ or $k \in \mathcal{K}_{\rightarrow}^{\mathrm{pg}}$, and for these respective cases the following properties hold:*

(i) *If $k \in \mathcal{K}_{\rightarrow}^{\mathrm{pg}}$, then $\alpha_{k+1} = \alpha_k$ and $f(x_{k+1})+r(x_{k+1}) \le f(x_k)+r(x_k)-\frac{\eta\varphi^2}{\alpha_k}(\chi_k^{\mathrm{pg}})^2$.*

(ii) *If $k \in \mathcal{K}_{\downarrow}^{\mathrm{pg}}$, then $\alpha_{k+1} = \xi\alpha_k$ and $f(x_{k+1}) + r(x_{k+1}) < f(x_k) + r(x_k)$.*

Proof. Since $k \in \mathcal{K}^{\mathrm{pg}}$, we know that the condition tested in line 7 of Algorithm 3.1 must not hold, meaning that $\chi_k^{\mathrm{pg}} > \chi_k^{\mathrm{m}}$. Combining this observation with line 15 of Algorithm 3.1 shows that the set $\mathcal{I}_k$ defined in line 15 satisfies

$$\|P_{\mathcal{I}_k}(s_k)\|_2 = \|[s_k]_{\mathcal{I}_k}\|_2 \ge \varphi\chi_k^{\mathrm{pg}} > 0. \tag{4.1}$$

Combining this result with Lemma 2.1 (using $\mathcal{I} = \mathcal{I}_k$, $\overline{x} = x_k$, and $\overline{\alpha} = \alpha_k$) yields

$$D_{f+r}(x_k; P_{\mathcal{I}_k}(s_k)) \le -\tfrac{1}{\alpha_k}\|P_{\mathcal{I}_k}(s_k)\|_2^2 < 0. \tag{4.2}$$

It is possible that Algorithm 3.4 terminates in line 52 because the inequality in line 49 does not hold for $j = 0$. In this case, Algorithm 3.4 successfully returns $x_{k+1} = y_0 = x_k + P_{\mathcal{I}_k}(s_k)$ and $\mathrm{flag}_k^{\mathrm{pg}} = \texttt{same\_}\alpha$, also indicating that $k \in \mathcal{K}_{\rightarrow}^{\mathrm{pg}}$. Since the while loop in line 49 terminates with $j = 0$, we can conclude that

$$f(x_{k+1}) + r(x_{k+1}) \equiv f(y_0) + r(y_0) \le f(x_k) + r(x_k) - \tfrac{\eta}{\alpha_k}\|P_{\mathcal{I}_k}(s_k)\|_2^2. \tag{4.3}$$

Combining this bound with (4.1) yields Lemma 4.2(i). Finally, since $\mathrm{flag}_k^{\mathrm{pg}} = \texttt{same\_}\alpha$, it follows from line 17 that $\alpha_{k+1} = \alpha_k$, completing the proof in this case.

It remains to consider the case when Algorithm 3.4 is unable to terminate in line 52 because the inequality in line 49 holds for $j = 0$. In this case, it follows from (4.2) and standard results for a backtracking Armijo line search that, for all sufficiently large j, the vector $y_j \leftarrow x_k + \xi^j P_{\mathcal{I}_k}(s_k)$ defined in line 50 of Algorithm 3.4 satisfies $f(y_j) + r(y_j) \le f(x_k) + r(x_k) + \eta\xi^j D_{f+r}(x_k; P_{\mathcal{I}_k}(s_k)) \le f(x_k) + r(x_k) - \eta\xi^j \frac{1}{\alpha_k}\|P_{\mathcal{I}_k}(s_k)\|_2^2$. This inequality shows that the while loop starting in line 49 of Algorithm 3.4 will terminate finitely, and thus Algorithm 3.4 successfully returns $x_{k+1} = y_j = x_k + \xi^j P_{\mathcal{I}_k}(s_k)$ for some $j > 0$ and $\mathrm{flag}_k^{\mathrm{pg}} = \texttt{decrease\_}\alpha$, also indicating that $k \in \mathcal{K}_{\downarrow}^{\mathrm{pg}}$. Moreover, that inequality may be combined with $y_j = x_{k+1}$, and (4.2) proves that $f(x_{k+1}) + r(x_{k+1}) < f(x_k) + r(x_k)$, as claimed. Finally, since $\mathrm{flag}_k^{\mathrm{pg}} = \texttt{decrease\_}\alpha$, we see in line 17 that $\alpha_{k+1} = \xi\alpha_k$. ∎

Next, we prove that the PG parameter remains bounded away from zero.

LEMMA 4.3. *The PG parameter sequence generated by Algorithm* 3.1 *satisfies $1 \ge \alpha_k \ge \alpha_{\min} := \min\left\{\alpha_0, \frac{2\xi(1-\eta)}{L}\right\} > 0$ for all $k \in \mathbb{N}$. Moreover, a bound on the number of times the PG parameter is decreased is given by*

$$|\mathcal{K}_{\downarrow}^{\mathrm{pg}}| \le c_{\downarrow}^{\alpha} := \max\left\{0, \left\lceil \log\left(\tfrac{\alpha_0 L}{2(1-\eta)}\right) / \log(\xi^{-1}) \right\rceil\right\}. \tag{4.4}$$

Proof. Since $\alpha_0 \in (0,1]$ in line 3 and $\alpha_{k+1} \leq \alpha_k$ for all $k \in \mathbb{N}$, we need only prove the lower bound on α_k. With that goal in mind, for the purpose of obtaining a contradiction, suppose that there exists an iteration k satisfying $\alpha_k \leq 2(1-\eta)/L < 2/L$, with the latter inequality holding since $\eta \in (0,1)$.

First suppose that $k \in \mathcal{K}^{\text{pg}}$. With $y_0 = x_k + P_{\mathcal{I}_k}(s_k)$ as in line 48 of Algorithm 3.4, it follows from Lemma 2.2 with $\overline{x} = x_k$, $\overline{\alpha} = \alpha_k$, and $s(\overline{x}, \overline{\alpha}) = s_k$ that $f(y_0) + r(y_0) \leq f(x_k) + r(x_k) - (\frac{1}{\alpha_k} - \frac{L}{2})\|P_{\mathcal{I}}(s_k)\|_2^2 \leq f(x_k) + r(x_k) - \big(\frac{1}{\alpha_k} - \frac{2(1-\eta)}{2\alpha_k}\big)\|P_{\mathcal{I}}(s_k)\|_2^2 = f(x_k) + r(x_k) - \frac{\eta}{\alpha_k}\|P_{\mathcal{I}}(s_k)\|_2^2$. This inequality implies that the condition checked in line 49 for $j = 0$ will not hold, meaning that $j = 0$ when line 51 is reached so that $\text{flag}_k^{\text{pg}} \leftarrow \texttt{same\_}\alpha$ in line 52. Thus, when line 17 in Algorithm 3.1 is reached, the update $\alpha_{k+1} \leftarrow \alpha_k$ will take place. Second, if $k \in \mathcal{K}^{\text{m}}$, then Algorithm 3.1 sets $\alpha_{k+1} \leftarrow \alpha_k$. To summarize, any time $\alpha_k \leq 2(1-\eta)/L$, the update $\alpha_{k+1} \leftarrow \alpha_k$ takes place; combining this with the fact that when the PG parameter is decreased the update $\alpha_{k+1} \leftarrow \xi\alpha_k$ is used (see line 17 in Algorithm 3.1) gives the lower bound on α_k.

We now prove (4.4). Let us observe from the first paragraph in this proof that if $\alpha_0 \leq 2(1-\eta)/L$, then $|\mathcal{K}_{\downarrow}^{\text{pg}}| = 0$, which verifies that (4.4) holds. Therefore, for the remainder of the proof, suppose that $\alpha_0 > 2(1-\eta)/L$. Combining this bound with the fact that when the PG parameter is decreased the update $\alpha_{k+1} \leftarrow \xi\alpha_k$ is used, we see that an upper bound on $|\mathcal{K}_{\downarrow}^{\text{pg}}|$ is the smallest integer ℓ such that $\alpha_0\xi^{\ell} \leq 2(1-\eta)/L$. Solving this inequality for ℓ shows that the result in (4.4) holds. ∎

We now switch our attention to iterations in $\mathcal{K}^{\text{m}}$. The next result establishes that Algorithm 3.2 is well posed and that the direction d_k that results from it when called by Algorithm 3.1 satisfies a certain descent property.

LEMMA 4.4. *For each $k \in \mathcal{K}^{\text{m}}$, Algorithm* 3.2 *is well posed. Moreover, the resulting direction $\bar{d}_k$, which is used to compute d_k in line* 11*, guarantees that d_k satisfies*

(i) $\nabla_{\mathcal{I}_k}(f+r)(x_k)^T[d_k]_{\mathcal{I}_k} \leq -\frac{1}{\mu_{\max}}\|\nabla_{\mathcal{I}_k}(f+r)(x_k)\|_2^2 < 0$ *and*

(ii) $\|d_k\|_2 \leq (2/\mu_{\min})\|\nabla_{\mathcal{I}_k}(f+r)(x_k)\|_2$,

where $\mathcal{I}_k \subseteq \mathcal{I}_k^{\text{m}}$ is the set in line 8 *used as an input to Algorithm* 3.2 *in line* 12*.*

Proof. Since $k \in \mathcal{K}^{\text{m}}$, Algorithm 3.2 is called in line 10 with input $\mathcal{I}_k$ defined in line 8. We first prove that $g_k = \nabla_{\mathcal{I}_k}(f+r)(x_k)$, as defined in line 9, is nonzero. For a proof by contradiction, suppose that $g_k = 0$ so that $\nabla_{\mathcal{G}_i}(f+r)(x_k) = 0$ for all i such that $\mathcal{G}_i \subseteq \mathcal{I}_k$. Consider arbitrary such i. Note that $[x_k]_{\mathcal{G}_i} \neq 0$ and $[x_k + s_k]_{\mathcal{G}_i} \neq 0$ since $\mathcal{G}_i \subseteq \mathcal{I}_k \subseteq \mathcal{I}_k^{\text{m}}$ (see line 8) and by how $\mathcal{I}_k^{\text{m}}$ is defined. This allows us to conclude from Lemma 2.3 that $[s_k]_{\mathcal{G}_i} = 0$, i.e., that $[s_k]_{\mathcal{I}_k} = 0$ since i with $\mathcal{G}_i \subseteq \mathcal{I}_k$ was arbitrary. This fact and line 8 yield $\chi_k^{\text{m}} = 0$, but since the inequality in line 7 must hold, we also have $\chi_k^{\text{pg}} = 0$. This contradicts Assumption 4.1, thus establishing that $g_k \neq 0$. Now, it follows from lines 9, 11, 22, and 21, $g_k \neq 0$, and Assumption 4.2 that $\nabla_{\mathcal{I}_k}(f+r)(x_k)^T[d_k]_{\mathcal{I}_k} \equiv g_k^T\bar{d}_k \leq g_k^T d_k^R = -\beta_k\|g_k\|_2^2 = -\|g_k\|_2^4/(g_k^T H_k g_k) \leq -\frac{1}{\mu_{\max}}\|g_k\|_2^2$. The result in (i) follows from this inequality and $g_k = \nabla_{\mathcal{I}_k}(f+r)(x_k) \neq 0$.

Part (ii) is precisely [6, Lemma 3.8] under our Assumption 4.2 since our conditions placed upon the step d_k are exactly the same as those used in [6]. ∎

The next lemma shows that, for $k \in \mathcal{K}^{\text{m}}$, a local Lipschitz property holds along a certain portion of the search path defined by the reduced-space m-direction.

LEMMA 4.5. *Let $k \in \mathcal{K}^{\text{m}}$ so that $\mathcal{I}_k$ is computed in line* 8*. The following hold:*

(i) *The constant $\theta \in (0, \pi/2)$ and index set $\mathcal{I}_k$ passed into Algorithm* 3.3 *satisfy, for each i such that $\mathcal{G}_i \subseteq \mathcal{I}_k$ with $\rho_{k,i}$ computed in* (3.5) *and $\bar{\rho}_{k,i}$ computed in line* 28*, the following conditions:*

(a) $\|[x_k + s_k]_{\mathcal{G}_i}\|_2 \neq 0$,
(b) $\|[x_k]_{\mathcal{G}_i}\|_2 \geq \rho_{k,i} \geq \bar{\rho}_{k,i} \geq \sin(\theta)\rho_{k,i} > 0$, *and*
(c) $\|[x_k]_{\mathcal{G}_i}\|_2 - \bar{\rho}_{k,i} \geq \kappa_2(1 - \sin(\theta))\|\nabla_{\mathcal{I}_k}(f + r)(x_k)\|_2^p$.

(ii) *For all step sizes* $\beta \in [0, \tau_k)$ *with* τ_k *computed in line* 33, *it holds, with*

$$\lambda_{\max} := \max\{\lambda_1, \lambda_2, \ldots, \lambda_{n_\mathcal{G}}\} \quad \textit{and} \quad \rho_{k,\min} := \min_i\{\rho_{k,i} : \mathcal{G}_i \subseteq \mathcal{I}_k\}, \tag{4.5}$$

that $\|\nabla_{\mathcal{I}_k}(f + r)(x_k) - \nabla_{\mathcal{I}_k}(f + r)(x_k + \beta d_k)\|_2 \leq \beta\big(L_g + \frac{\lambda_{\max}}{\rho_{k,\min}}\big)\|[d_k]_{\mathcal{I}_k}\|_2$.

Proof. We first prove part (i). Consider arbitrary i with $\mathcal{G}_i \subseteq \mathcal{I}_k$, where $\mathcal{I}_k \subseteq \mathcal{I}_k^{\rm m}$ is passed into Algorithm 3.3 and constructed to satisfy the condition in line 8. Part (a) follows from $\mathcal{I}_k^{\rm m} \subseteq \bar{\mathcal{I}}_k^{\rm m}$ and the definition of $\bar{\mathcal{I}}_k^{\rm m}$ in (3.2). The first inequality in part (b) follows from $\mathcal{I}_k^{\rm m} \subseteq \bar{\mathcal{I}}_k^{\rm m}$ and how $\mathcal{I}_k^{\rm m}$, $\mathcal{I}_k^{\rm small}$, and $\bar{\mathcal{I}}_k^{\rm m}$ are defined. The second inequality in (b) follows from how $\bar{\rho}_{k,i}$ is defined in line 28. The third inequality in (b) follows from line 28 and the first inequality in (b). To complete part (b), we prove $\rho_{k,i} > 0$. For a proof by contradiction, assume $\rho_{k,i} = 0$, which by (3.5) means that $\|\nabla_{\mathcal{I}_k^{\rm m}}(f + r)(x_k)\|_2 = 0$. This fact means that each i with $\mathcal{G}_i \subseteq \mathcal{I}_k \subseteq \mathcal{I}_k^{\rm m}$ satisfies $\|\nabla_{\mathcal{G}_i}(f + r)(x_k)\|_2 = 0$, which with Lemma 2.3 (using $\bar{x} = x_k$, $\bar{\alpha} = \alpha_k$, and $s(\bar{x}, \bar{\alpha}) = s_k$) and the definition of $\mathcal{I}_k^{\rm m}$ implies that $\|[s_k]_{\mathcal{G}_i}\|_2 = 0$ for each $\mathcal{G}_i \subseteq \mathcal{I}_k$, i.e., that $\|[s_k]_{\mathcal{I}_k}\|_2 = 0$. It now follows from line 8 that $\chi_k^{\rm m} = 0$, which with line 7 yields $\chi_k^{\rm pg} = 0$. We have reached a contradiction to Assumption 4.1, and conclude that $\rho_{k,i} > 0$, as claimed. Finally, we prove part (c). It follows from line 28, $\theta \in (0, \pi/2)$, part (b), (3.5), and the fact that $\mathcal{I}_k \subseteq \mathcal{I}_k^{\rm m}$ that $\|[x_k]_{\mathcal{G}_i}\|_2 - \bar{\rho}_{k,i} \geq \|[x_k]_{\mathcal{G}_i}\|_2 - \sin(\theta)\|[x_k]_{\mathcal{G}_i}\|_2 = (1 - \sin(\theta))\|[x_k]_{\mathcal{G}_i}\|_2 \geq (1 - \sin(\theta))\rho_{k,i} \geq \kappa_2(1 - \sin(\theta))\|\nabla_{\mathcal{I}_k^{\rm m}}(f + r)(x_k)\|_2^p \geq \kappa_2(1 - \sin(\theta))\|\nabla_{\mathcal{I}_k}(f + r)(x_k)\|_2^p$, which completes the proof of part (c).

To prove part (ii), let $\beta \in [0, \tau_k)$. It follows from part (i) and the definition of τ_k in line 33 that every point on the segment that connects $[x_k]_{\mathcal{G}_i}$ to $[x_k + \beta d_k]_{\mathcal{G}_i}$ is outside of the ball in $\mathbb{R}^{|\mathcal{G}_i|}$ centered at zero of radius $\bar{\rho}_{k,i} > 0$. This means that both $\|[x_k]_{\mathcal{G}_i}\| \geq \bar{\rho}_{k,i}$ and $\|[x_k + \beta d_k]_{\mathcal{G}_i}\| \geq \bar{\rho}_{k,i}$. It now follows that

$$\begin{aligned}
&\|\nabla_{\mathcal{G}_i} r(x_k) - \nabla_{\mathcal{G}_i} r(x_k + \beta d_k)\|_2 \\
&= \lambda_i \left\| \frac{[x_k]_{\mathcal{G}_i}}{\|[x_k]_{\mathcal{G}_i}\|_2} - \frac{[x_k + \beta d_k]_{\mathcal{G}_i}}{\|[x_k + \beta d_k]_{\mathcal{G}_i}\|_2} \right\|_2 = \frac{\lambda_i}{\bar{\rho}_{k,i}} \left\| \frac{\bar{\rho}_{k,i}[x_k]_{\mathcal{G}_i}}{\|[x_k]_{\mathcal{G}_i}\|_2} - \frac{\bar{\rho}_{k,i}[x_k + \beta d_k]_{\mathcal{G}_i}}{\|[x_k + \beta d_k]_{\mathcal{G}_i}\|_2} \right\|_2 \\
&\leq \frac{\lambda_i}{\bar{\rho}_{k,i}} \|[x_k]_{\mathcal{G}_i} - [x_k + \beta d_k]_{\mathcal{G}_i}\|_2 = \frac{\lambda_i \beta}{\bar{\rho}_{k,i}} \|[d_k]_{\mathcal{G}_i}\|_2,
\end{aligned} \tag{4.6}$$

where the (only) inequality follows from the nonexpansive property of the projection (of $[x_k]_{\mathcal{G}_i}$ and $[x_k + \beta d_k]_{\mathcal{G}_i}$) onto the ball of radius $\bar{\rho}_{k,i}$. From (4.6) we have

$$\begin{aligned}
&\|\nabla_{\mathcal{I}_k} r(x_k) - \nabla_{\mathcal{I}_k} r(x_k + \beta d_k)\|_2^2 \\
&= \sum_{i:\mathcal{G}_i \subseteq \mathcal{I}_k} \|\nabla_{\mathcal{G}_i} r(x_k) - \nabla_{\mathcal{G}_i} r(x_k + \beta d_k)\|_2^2 \leq \beta^2 \sum_{i:\mathcal{G}_i \subseteq \mathcal{I}_k} \frac{\lambda_i^2}{\bar{\rho}_{k,i}^2} \|[d_k]_{\mathcal{G}_i}\|_2^2 \\
&\leq \frac{\beta^2 \lambda_{\max}^2}{\rho_{k,\min}^2} \sum_{i:\mathcal{G}_i \subseteq \mathcal{I}_k} \|[d_k]_{\mathcal{G}_i}\|_2^2 = \frac{\beta^2 \lambda_{\max}^2}{\rho_{k,\min}^2} \|[d_k]_{\mathcal{I}_k}\|_2^2.
\end{aligned} \tag{4.7}$$

It follows from Assumption 1.1, $[d_k]_{\mathcal{I}_k^c} = 0$, the triangle inequality, and (4.7) that $\|\nabla_{\mathcal{I}_k}(f + r)(x_k) - \nabla_{\mathcal{I}_k}(f + r)(x_k + \beta d_k)\|_2 \leq \|\nabla_{\mathcal{I}_k} f(x_k) - \nabla_{\mathcal{I}_k} f(x_k + \beta d_k)\|_2 + \|\nabla_{\mathcal{I}_k} r(x_k) - \nabla_{\mathcal{I}_k} r(x_k + \beta d_k)\|_2 \leq \beta\big(L_g + \frac{\lambda_{\max}}{\rho_{k,\min}}\big)\|[d_k]_{\mathcal{I}_k}\|_2$, as claimed. ∎

We now show that Algorithm 3.3 is well posed and that the new iterate it produces satisfies a decrease property that will be used in the final complexity result.

LEMMA 4.6. *For each $k \in \mathcal{K}^{\mathrm{m}}$, Algorithm 3.3 is called in line 12 and successfully returns x_{k+1} and $\mathrm{flag}_k^{\mathrm{m}}$. Moreover, the value of $\mathrm{flag}_k^{\mathrm{m}}$ indicates whether $k \in \mathcal{K}_0^{\mathrm{m}}$ or $k \in \mathcal{K}_{\mathrm{sd}}^{\mathrm{m}}$, and for these respective cases the following properties hold:*

(i) *If $k \in \mathcal{K}_0^{\mathrm{m}}$, then $f(x_{k+1}) + r(x_{k+1}) \leq f(x_k) + r(x_k)$, and x_{k+1} has at least one additional block of zeros compared to x_k.*

(ii) *If $k \in \mathcal{K}_{\mathrm{sd}}^{\mathrm{m}}$, then*

$$f(x_{k+1}) + r(x_{k+1}) \leq f(x_k) + r(x_k) - \min\{c_1(\chi_k^{\mathrm{m}})^{1+p}, c_2(\chi_k^{\mathrm{m}})^{2+p}\}, \tag{4.8}$$

where $c_1 := \frac{\eta\xi\mu_{\min}\kappa_2\left(1-\sin(\theta)\right)\varphi^{1+p}}{2\mu_{\max}}$ *and* $c_2 := \frac{\kappa_2\mu_{\min}^2\xi\eta(1-\eta)\varphi^{2+p}}{2\mu_{\max}^2\left(L_g\kappa_2(L_f+\lambda_{\max}\sqrt{n_{\mathcal{G}}})^p+\lambda_{\max}\right)}$.

Proof. Throughout, we use $F := f + r$. It is possible that Algorithm 3.3 successfully terminates in line 39, in which case it follows from line 39 and line 38 that the returned x_{k+1} and $\mathrm{flag}_k^{\mathrm{m}}$ satisfy $F(x_{k+1}) \leq F(x_k)$ and $\mathrm{flag}_k^{\mathrm{m}} = \texttt{new\_zero}$, indicating that $k \in \mathcal{K}_0^{\mathrm{m}}$. Moreover, upon termination, the value j satisfies $\xi^j \geq \tau_k$ (see line 34), which combined with line 37 shows that at least one additional group of variables has become zero at x_{k+1}. This proves that part (i) holds.

Next, suppose that Algorithm 3.3 does not terminate in line 39. Observe from the definition of τ_k in line 33 that $\tau_k > 0$ (this follows from Lemma 4.5(i) and the definition of $\bar{\rho}_{k,i}$). Therefore, it follows that the while loop starting in line 34 will terminate with the smallest nonnegative integer $\bar{j}$ such that $\xi^{\bar{j}} < \tau_k$, and the loop in line 41 will begin with $j = \bar{j}$. We now claim that the condition in line 43 used to determine termination of the loop is satisfied for all $j \geq \bar{j}$ such that

$$\xi^j \in \left[0, \frac{2(\eta-1)\nabla_{\mathcal{I}_k}F(x_k)^T[d_k]_{\mathcal{I}_k}}{(L_g + \lambda_{\max}/\rho_{k,\min})\|[d_k]_{\mathcal{I}_k}\|_2^2}\right] \subset [0, \tau_k). \tag{4.9}$$

To see that this claim holds, we can use the integral form of Taylor's theorem and Lemma 4.5(ii) (using the fact that $\gamma\xi^j \in [0, \tau_k)$ for all $\gamma \in [0, 1]$) to obtain

$$\begin{aligned}
&|F(x_k + \xi^j d_k) - F(x_k) - \xi^j \nabla_{\mathcal{I}_k}F(x_k)^T[d_k]_{\mathcal{I}_k}| \\
&\quad \leq \left|\int_0^1 \xi^j [d_k]_{\mathcal{I}_k}^T\left(\nabla_{\mathcal{I}_k}F(x_k + \gamma\xi^j d_k) - \nabla_{\mathcal{I}_k}F(x_k)\right)\mathrm{d}\gamma\right| \\
&\quad \leq \xi^j \int_0^1 \|[d_k]_{\mathcal{I}_k}\|_2\|\nabla_{\mathcal{I}_k}F(x_k + \gamma\xi^j d_k) - \nabla_{\mathcal{I}_k}F(x_k)\|_2\mathrm{d}\gamma \\
&\quad \leq \xi^{2j}(L_g + \lambda_{\max}/\rho_{k,\min})\|[d_k]_{\mathcal{I}_k}\|_2^2 \int_0^1 \gamma\mathrm{d}\gamma = \tfrac{1}{2}\xi^{2j}(L_g + \lambda_{\max}/\rho_{k,\min})\|[d_k]_{\mathcal{I}_k}\|_2^2.
\end{aligned}$$

Combining this inequality with (4.9) yields

$$\begin{aligned}
F(x_k + \xi^j d_k) &\leq F(x_k) + \xi^j \nabla_{\mathcal{I}_k}F(x_k)^T[d_k]_{\mathcal{I}_k} + \tfrac{1}{2}\xi^{2j}(L_g + \lambda_{\max}/\rho_{k,\min})\|[d_k]_{\mathcal{I}_k}\|_2^2 \\
&= F(x_k) + \xi^j \nabla_{\mathcal{I}_k}F(x_k)^T[d_k]_{\mathcal{I}_k} + \xi^j(\eta-1)\nabla_{\mathcal{I}_k}F(x_k)^T[d_k]_{\mathcal{I}_k} \\
&= F(x_k) + \eta\xi^j \nabla_{\mathcal{I}_k}F(x_k)^T[d_k]_{\mathcal{I}_k},
\end{aligned}$$

which establishes our claim that the inequality in line 43 holds for all $j \geq \bar{j}$ such that ξ^j satisfies (4.9). This shows that the loop will successfully terminate with $\mathrm{flag}_k^{\mathrm{m}} = \texttt{suff\_descent}$ (thus indicating that $k \in \mathcal{K}_{\mathrm{sd}}^{\mathrm{m}}$) and x_{k+1} satisfying

$$F(x_{k+1}) \leq F(x_k) + \eta\hat{\xi^j}\nabla_{\mathcal{I}_k}F(x_k)^T[d_k]_{\mathcal{I}_k} \tag{4.10}$$

for some $\hat{j}$ satisfying

$$\begin{aligned}\xi^{\hat{j}} &\geq \min\left\{\xi^{\bar{j}}, \frac{2\xi(\eta-1)\nabla_{\mathcal{I}_k}F(x_k)^T[d_k]_{\mathcal{I}_k}}{(L_g+\lambda_{\max}/\rho_{k,\min})\|[d_k]_{\mathcal{I}_k}\|_2^2}\right\}\\ &\geq \min\left\{\xi\tau_k, \frac{2\xi(\eta-1)\nabla_{\mathcal{I}_k}F(x_k)^T[d_k]_{\mathcal{I}_k}}{(L_g+\lambda_{\max}/\rho_{k,\min})\|[d_k]_{\mathcal{I}_k}\|_2^2}\right\},\end{aligned} \tag{4.11}$$

where the second inequality follows from the fact that $\bar{j}$ is the *smallest* nonnegative integer such that $\xi^{\bar{j}} < \tau_k$. We now consider two cases.

Case 1. The minimum in (4.11) is $\xi\tau_k$, from which we may conclude that $\tau_k < \infty$. Using (4.10) and Lemma 4.4(i) we have that

$$F(x_{k+1}) \leq F(x_k) + \eta\xi^{\hat{j}}\nabla_{\mathcal{I}_k}F(x_k)^T[d_k]_{\mathcal{I}_k} \leq F(x_k) - \tfrac{\eta\xi}{\mu_{\max}}\tau_k\|\nabla_{\mathcal{I}_k}F(x_k)\|_2^2. \tag{4.12}$$

We now seek a lower bound on τ_k. Consider i such that $\tau_{k,i} < \infty$ when computed in Algorithm 3.3. The triangle inequality gives $\bar{\rho}_{k,i} = \|[x_k+\tau_{k,i}d_k]_{\mathcal{G}_i}\|_2 \geq \|[x_k]_{\mathcal{G}_i}\|_2 - \tau_{k,i}\|[d_k]_{\mathcal{G}_i}\|_2$, which together with Lemma 4.5(i)(c) and Lemma 4.4(ii) shows that $\tau_{k,i} \geq \frac{\|[x_k]_{\mathcal{G}_i}\|_2-\bar{\rho}_{k,i}}{\|[d_k]_{\mathcal{G}_i}\|_2} \geq \frac{\mu_{\min}\kappa_2(1-\sin(\theta))\|\nabla_{\mathcal{I}_k}F(x_k)\|_2^p}{2\|\nabla_{\mathcal{I}_k}F(x_k)\|_2} = \frac{\mu_{\min}}{2}\kappa_2(1-\sin(\theta))\|\nabla_{\mathcal{I}_k}F(x_k)\|_2^{p-1}$. From this, it follows that $\tau_k \geq \frac{1}{2}\mu_{\min}\kappa_2(1-\sin(\theta))\|\nabla_{\mathcal{I}_k}F(x_k)\|_2^{p-1}$. Using this inequality with (4.12), Lemma 2.3, and the set $\mathcal{I}_k$ from line 8 shows that $F(x_{k+1}) \leq F(x_k) - \frac{\eta\xi\mu_{\min}\kappa_2(1-\sin(\theta))}{2\mu_{\max}}\|\nabla_{\mathcal{I}_k}F(x_k)\|_2^{1+p} \leq F(x_k) - \frac{\eta\xi\mu_{\min}\kappa_2(1-\sin(\theta))}{2\mu_{\max}}\|[s_k]_{\mathcal{I}_k}\|_2^{1+p} \leq F(x_k) - \frac{\eta\xi\mu_{\min}\kappa_2(1-\sin(\theta))\varphi^{1+p}}{2\mu_{\max}}(\chi_k^{\mathrm{m}})^{1+p}$, thus completing the proof for this case.

Case 2. The minimum in (4.11) is $\frac{2\xi(\eta-1)\nabla_{\mathcal{I}_k}F(x_k)^T[d_k]_{\mathcal{I}_k}}{(L+\lambda_{\max}/\rho_{k,\min})\|[d_k]_{\mathcal{I}_k}\|_2^2}$. Combining this fact with (4.10), (4.11), Lemma 4.4(i), and Lemma 4.4(ii) shows that

$$\begin{aligned}F(x_{k+1}) &\leq F(x_k) + \eta\xi^{\hat{j}}\nabla_{\mathcal{I}_k}F(x_k)^T[d_k]_{\mathcal{I}_k}\\ &\leq F(x_k) - \frac{2\xi\eta(1-\eta)\|\nabla_{\mathcal{I}_k}F(x_k)\|_2^4}{\mu_{\max}^2(L_g+\lambda_{\max}/\rho_{k,\min})\|[d_k]_{\mathcal{I}_k}\|_2^2}\\ &\leq F(x_k) - \frac{2\mu_{\min}^2\xi\eta(1-\eta)\|\nabla_{\mathcal{I}_k}F(x_k)\|_2^4}{4\mu_{\max}^2(L_g+\lambda_{\max}/\rho_{k,\min})\|\nabla_{\mathcal{I}_k}F(x_k)\|_2^2}\\ &= F(x_k) - \frac{\mu_{\min}^2\xi\eta(1-\eta)\|\nabla_{\mathcal{I}_k}F(x_k)\|_2^2}{2\mu_{\max}^2(L_g+\lambda_{\max}/\rho_{k,\min})}.\end{aligned} \tag{4.13}$$

It follows from (4.5), (3.5), and $\mathcal{I}_k \subseteq \mathcal{I}_k^{\mathrm{m}}$ that $\rho_{k,\min} \geq \kappa_2\|\nabla_{\mathcal{I}_k}F(x_k)\|_2^p$. Combining this bound with (4.13) shows that

$$\begin{aligned}F(x_{k+1}) &\leq F(x_k) - \frac{\mu_{\min}^2\xi\eta(1-\eta)\|\nabla_{\mathcal{I}_k}F(x_k)\|_2^2}{2\mu_{\max}^2(L_g+\lambda_{\max}/\rho_{k,\min})}\\ &\leq F(x_k) - \frac{\mu_{\min}^2\xi\eta(1-\eta)\|\nabla_{\mathcal{I}_k}F(x_k)\|_2^2}{2\mu_{\max}^2\big(L_g+\lambda_{\max}/(\kappa_2\|\nabla_{\mathcal{I}_k}F(x_k)\|_2^p)\big)}\\ &= F(x_k) - \frac{\kappa_2\mu_{\min}^2\xi\eta(1-\eta)\|\nabla_{\mathcal{I}_k}F(x_k)\|_2^{2+p}}{2\mu_{\max}^2(L_g\kappa_2\|\nabla_{\mathcal{I}_k}F(x_k)\|_2^p+\lambda_{\max})}.\end{aligned} \tag{4.14}$$

Next, we know from Lemma 4.2, Lemma 4.6(i), (4.12), and (4.14) that $F(x_k) \leq F(x_0)$ for all $k \in \mathbb{N}$, i.e., $x_k \in \mathcal{L}$ for all $k \in \mathbb{N}$. Combining this fact with the triangle inequality, Assumption 1.1, the definition of r, and (4.5) gives

$$\begin{aligned}\|\nabla_{\mathcal{I}_k} F(x_k)\|_2 &\leq \|\nabla_{\mathcal{I}_k} f(x_k)\|_2 + \|\nabla_{\mathcal{I}_k} r(x_k)\|_2 = \|\nabla_{\mathcal{I}_k} f(x_k)\|_2 + \Big(\sum_{i:\mathcal{G}_i \subseteq \mathcal{I}_k} \|\nabla_{\mathcal{G}_i} r(x_k)\|_2^2\Big)^{1/2} \\ &\leq L_f + \Big(\sum_{i:\mathcal{G}_i \subseteq \mathcal{I}_k} \|\lambda_i [x_k]_{\mathcal{G}_i} / \|[x_k]_{\mathcal{G}_i}\|_2\|_2^2\Big)^{1/2} = L_f + \Big(\sum_{i:\mathcal{G}_i \subseteq \mathcal{I}_k} \lambda_i^2\Big)^{1/2} \\ &\leq L_f + \Big(\sum_{i:\mathcal{G}_i \subseteq \mathcal{I}_k} \lambda_{\max}^2\Big)^{1/2} \leq L_f + \lambda_{\max}\sqrt{n_{\mathcal{G}}}.\end{aligned}$$

We can now combine this inequality with (4.14) to obtain the bound $F(x_{k+1}) \leq F(x_k) - \Big(\frac{\kappa_2 \mu_{\min}^2 \xi \eta (1-\eta)}{2\mu_{\max}^2 (L_g \kappa_2 (L_f + \lambda_{\max}\sqrt{n_{\mathcal{G}}})^p + \lambda_{\max})}\Big) \|\nabla_{\mathcal{I}_k} F(x_k)\|_2^{2+p}$, which with Lemma 2.3 and line 8 gives $F(x_{k+1}) \leq F(x_k) - \Big(\frac{\kappa_2 \mu_{\min}^2 \xi \eta (1-\eta)}{2\mu_{\max}^2 (L_g \kappa_2 (L_f + \lambda_{\max}\sqrt{n_{\mathcal{G}}})^p + \lambda_{\max})}\Big) \|[s_k]_{\mathcal{I}_k}\|_2^{2+p} \leq F(x_k) - \Big(\frac{\kappa_2 \mu_{\min}^2 \xi \eta (1-\eta) \varphi^{2+p}}{2\mu_{\max}^2 (L_g \kappa_2 (L_f + \lambda_{\max}\sqrt{n_{\mathcal{G}}})^p + \lambda_{\max})}\Big) (\chi_k^{\mathrm{m}})^{2+p}$, thus completing the proof. ∎

The result in (4.8) motivates us to define the following subsets of $\mathcal{K}_{\mathrm{sd}}^{\mathrm{m}}$:

$$\mathcal{K}_{\mathrm{sd,big}}^{\mathrm{m}} := \{k \in \mathcal{K}_{\mathrm{sd}}^{\mathrm{m}} : \chi_k^{\mathrm{m}} \geq c_1/c_2\} \quad \text{and} \quad \mathcal{K}_{\mathrm{sd,small}}^{\mathrm{m}} := \mathcal{K}_{\mathrm{sd}}^{\mathrm{m}} \setminus \mathcal{K}_{\mathrm{sd,big}}^{\mathrm{m}}. \tag{4.15}$$

This distinction plays a role in our complexity result. First, we require a lemma.

LEMMA 4.7. *The objective function $f + r$ is monotonically decreasing over the sequence of iterates $\{x_k\}$ and $\lim_{k\to\infty} \big(f(x_k) + r(x_k)\big) =: F_{\min} > -\infty$.*

Proof. It follows from Lemma 4.2 and Lemma 4.6 that the objective function is monotonically decreasing over the iterate sequence. The remaining conclusion of the lemma follows from the monotonicity property and Assumption 1.1. ∎

The main theorem can now be stated. It gives an upper bound on the number of iterations performed by Algorithm 3.1 before an approximate solution is obtained.

THEOREM 4.8. *Let c_1 and c_2 be the constants in Lemma (4.6)(ii), and let us define $c_3 := \eta\varphi^2/\alpha_0 > 0$. For any $\epsilon > 0$, define $\mathcal{K}_\epsilon := \{k \in \mathbb{N} : \max\{\chi_k^{\mathrm{m}}, \chi_k^{\mathrm{pg}}\} > \epsilon\}$. Then,*

$$\begin{aligned} &|\mathcal{K}_{\rightarrow}^{\mathrm{pg}} \cap \mathcal{K}_\epsilon| \leq c_{\mathrm{pg}} \epsilon^{-2} + 1, \\ &|\mathcal{K}_{\mathrm{sd,big}}^{\mathrm{m}} \cap \mathcal{K}_\epsilon| \leq c_{\mathrm{big}} \epsilon^{-(1+p)} + 1, \quad \text{and} \quad |\mathcal{K}_{\mathrm{sd,small}}^{\mathrm{m}} \cap \mathcal{K}_\epsilon| \leq c_{\mathrm{small}} \epsilon^{-(2+p)} + 1, \end{aligned} \tag{4.16}$$

where $c_{\mathrm{pg}} := \big(f(x_0) + r(x_0) - F_{\min}\big)/c_3$, $c_{\mathrm{big}} := \big(f(x_0) + r(x_0) - F_{\min}\big)/c_1$, and $c_{\mathrm{small}} := \big(f(x_0) + r(x_0) - F_{\min}\big)/c_2$. Therefore, if $\epsilon \geq c_1/c_2$, then

$$|\mathcal{K}_\epsilon| \leq \big(c_\downarrow^\alpha + c_{\mathrm{pg}} \epsilon^{-2} + c_{\mathrm{big}} \epsilon^{-(1+p)} + 2\big)(1 + n_{\mathcal{G}}) + n_{\mathcal{G}}, \tag{4.17}$$

where $c_\downarrow^\alpha$ is defined in (4.4); otherwise, i.e., if $\epsilon < c_1/c_2$, then

$$|\mathcal{K}_\epsilon| \leq \big(c_\downarrow^\alpha + c_{\mathrm{pg}} \epsilon^{-2} + c_{\mathrm{big}} \epsilon^{-(1+p)} + c_{\mathrm{small}} \epsilon^{-(2+p)} + 3\big)(1 + n_{\mathcal{G}}) + n_{\mathcal{G}}. \tag{4.18}$$

Proof. Note that the definitions of $\mathcal{K}^{\mathrm{m}}$ and $\mathcal{K}^{\mathrm{pg}}$ together with line 7 show that

$$\chi_k^{\mathrm{m}} \geq \chi_k^{\mathrm{pg}} \text{ for } k \in \mathcal{K}^{\mathrm{m}} \quad \text{and} \quad \chi_k^{\mathrm{pg}} > \chi_k^{\mathrm{m}} \text{ for } k \in \mathcal{K}^{\mathrm{pg}}. \tag{4.19}$$

Define $\Delta_k := f(x_k) + r(x_k) - \big(f(x_{k+1}) + r(x_{k+1})\big)$ and $\chi_k := \max\{\chi_k^{\rm pg}, \chi_k^{\rm m}\}$. Using Lemma 4.2(i), Lemma 4.3, Lemma 4.6(ii), the definitions of c_3 and $\mathcal{K}_\epsilon$ in the statement of the theorem, and (4.19) shows for arbitrary $\bar{k} \in \mathbb{N}$ that

$$\begin{aligned}
f(x_0) + r(x_0) - \big(f(x_{\bar{k}+1}) + r(x_{\bar{k}+1})\big) &= \sum_{0\le k\le \bar{k}} \Delta_k \\
&\ge \sum_{\substack{k\in\mathcal{K}_{\rightarrow}^{\rm pg}\cap\mathcal{K}_\epsilon \\ 0\le k\le\bar{k}}} \Delta_k + \sum_{\substack{k\in\mathcal{K}_{\rm sd,big}^{\rm m}\cap\mathcal{K}_\epsilon \\ 0\le k\le\bar{k}}} \Delta_k + \sum_{\substack{k\in\mathcal{K}_{\rm sd,small}^{\rm m}\cap\mathcal{K}_\epsilon \\ 0\le k\le\bar{k}}} \Delta_k \\
&\ge \sum_{\substack{k\in\mathcal{K}_{\rightarrow}^{\rm pg}\cap\mathcal{K}_\epsilon \\ 0\le k\le\bar{k}}} c_3(\chi_k^{\rm pg})^2 + \sum_{\substack{k\in\mathcal{K}_{\rm sd,big}^{\rm m}\cap\mathcal{K}_\epsilon \\ 0\le k\le\bar{k}}} c_1(\chi_k^{\rm m})^{1+p} + \sum_{\substack{k\in\mathcal{K}_{\rm sd,small}^{\rm m}\cap\mathcal{K}_\epsilon \\ 0\le k\le\bar{k}}} c_2(\chi_k^{\rm m})^{2+p} \\
&\ge \sum_{\substack{k\in\mathcal{K}_{\rightarrow}^{\rm pg}\cap\mathcal{K}_\epsilon \\ 0\le k\le\bar{k}}} c_3\epsilon^2 + \sum_{\substack{k\in\mathcal{K}_{\rm sd,big}^{\rm m}\cap\mathcal{K}_\epsilon \\ 0\le k\le\bar{k}}} c_1\epsilon^{1+p} + \sum_{\substack{k\in\mathcal{K}_{\rm sd,small}^{\rm m}\cap\mathcal{K}_\epsilon \\ 0\le k\le\bar{k}}} c_2\epsilon^{2+p}.
\end{aligned}$$

From this inequality and Lemma 4.7, one finds that (4.16) follows.

Next, suppose that $\epsilon \ge c_1/c_2$. It follows from (4.15) and (4.19) that $\chi_k^{\rm m} = \max\{\chi_k^{\rm pg}, \chi_k^{\rm m}\} > \epsilon \ge c_1/c_2$ for all $k \in \mathcal{K}^{\rm m}$, which implies that $\mathcal{K}_{\rm sd,small}^{\rm m} \cap \mathcal{K}_\epsilon = \emptyset$. The result in (4.17) follows from this observation, (4.16), (4.4), and since (by Lemma 4.6(i)) at most $n_{\mathcal{G}}$ iterations in $\mathcal{K}_0^{\rm m}$ can occur before the first, after the last, or between any two iterations in $\mathcal{K}_{\downarrow}^{\rm pg} \cup \mathcal{K}_{\rightarrow}^{\rm pg} \cup \mathcal{K}_{\rm sd}^{\rm m}$.

The final result (4.18) follows using the same argument as in the previous paragraph, except now $\mathcal{K}_{\rm sd,small}^{\rm m} \cap \mathcal{K}_\epsilon$ is no longer necessarily empty. ∎

We see from (4.18) that, for all sufficiently small ϵ, the worst-case complexity result for Algorithm 3.1 is $n_{\mathcal{G}}\epsilon^{-(2+p)}$, which is worse than the ϵ^{-2} result that holds for the PG method. However, as is typical with well-designed second-derivative methods, although the complexity bound is worse, it typically performs better (see section 5). Also, the PG method would not converge locally superlinearly, whereas our method can exhibit this behavior, as we show in the next section.

4.2. Local convergence. We now consider the local convergence rate of the iterates generated by Algorithm 3.1. Our analysis is performed under the following additional assumption that will be assumed to hold throughout this section.

Assumption 4.3. The following conditions related to problem (1.1) hold:

(i) The function f is twice continuously differentiable, (1.1) has a unique solution x_*, and $\nabla^2 f : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is Lipschitz continuous in a neighborhood of x_*.

(ii) With $\mathcal{S}_* := \{i : [x_*]_{\mathcal{G}_i} \ne 0\}$ and $\mathcal{X}_* := \{x \in \mathbb{R}^n : [x]_{\mathcal{S}_*^c} = [x_*]_{\mathcal{S}_*^c} = 0\}$, where $\mathcal{S}_*^c$ is the complement of $\mathcal{S}_*$, there exists a scalar $\sigma_* \in (0,\infty)$ such that $p^T\nabla^2 f(x_*)p \ge \sigma_*\|p\|_2^2$ for all $p \in \mathcal{X}_*$.

(iii) f is nondegenerate at x_*, i.e., $\|[\nabla f(x_*)]_{\mathcal{G}_i}\|_2 < \lambda_i$ for all $i \notin \mathcal{S}_*$.

Assumption 4.3(ii) is a relaxation of the requirement that f is strongly convex. As for Assumption 4.3(iii), it is a more strict version of the optimality condition for problem (1.1), namely, that $\|[\nabla f(x_*)]_{\mathcal{G}_i}\|_2 \le \lambda_i$ for all $i \notin \mathcal{S}_*$.

Assumption 4.4. The following algorithmic choices are made in Algorithm 3.1:

(i) The backtracking parameter is chosen to satisfy $\eta \in (0, 1/2)$.

(ii) For all sufficiently large $k \in \mathbb{N}$, $\mathcal{I}_k$ in line 8/15 is

$$\mathcal{I}_k = \begin{cases} \mathcal{I}_k^{\mathrm{m}} & \text{if } k \in \mathcal{K}^{\mathrm{m}}, \\ \mathcal{I}_k^{\mathrm{pg}} & \text{if } k \in \mathcal{K}^{\mathrm{pg}}. \end{cases}$$

(iii) For all sufficiently large $k \in \mathcal{K}^{\mathrm{m}}$, $H_k = \nabla^2_{\mathcal{I}_k \mathcal{I}_k}(f+r)(x_k)$ is chosen in line 9.

Finally, we make the following assumption about the iterate sequence generated.

Assumption 4.5. The iterate sequence $\{x_k\}$ has a limit point.

This assumption is not too restrictive. It holds, for example, when the level set $\mathcal{L}$ is bounded due to the fact that $\{f(x_k)+r(x_k)\}$ is monotonically decreasing. The level set $\mathcal{L}$ is guaranteed to be bounded in various situations, such as when $f+r$ is coercive, which occurs, for instance, when f is nonnegative-valued (since r is coercive).

The first result of this section shows that the iterate sequence converges to x_*.

THEOREM 4.9. *The iterate sequence* $\{x_k\}$ *generated by Algorithm* 3.1 *satisfies* $\lim_{k\to\infty} x_k = x_*$ *and* $\lim_{k\to\infty} \max\{\chi_k^{\mathrm{pg}}, \chi_k^{\mathrm{m}}\} = 0$.

Proof. Theorem 4.8 gives $\lim_{k\to\infty} \max\{\chi_k^{\mathrm{pg}}, \chi_k^{\mathrm{m}}\} = 0$. Since Assumption 4.5 ensures that $\{x_k\}$ has a limit point, say $\hat{x}$, there exists an infinite $\mathcal{K} \subseteq \mathbb{N}$ such that $\lim_{k\in\mathcal{K},k\to\infty} x_k = \hat{x}$. Then, Lemma 4.1 and Lemma 4.3 imply that $\hat{x}$ is a solution to (1.1), but then Assumption 4.3 shows that $\hat{x} = x_*$, so $\lim_{k\in\mathcal{K},k\to\infty} x_k = x_*$. The fact that the *entire* sequence $\{x_k\}$ converges to x_* follows from $\lim_{k\in\mathcal{K},k\to\infty} x_k = x_*$, the uniqueness of x_* in Assumption 4.3(i), and monotonicity of $\{f(x_k) + r(x_k)\}$. ∎

We now show for groups whose variables are all equal to zero at the solution x_* that the PG step will eventually predict them to be zero.

LEMMA 4.10. *For all* $i \notin \mathcal{S}_*$ *and sufficiently large* k, *it holds that* $[x_k + s_k]_{\mathcal{G}_i} = 0$.

Proof. First note that Lemma 4.3 and the update strategy for $\{\alpha_k\}$ in Algorithm 3.1 ensure that there exists $\bar{k}_1$ such that $\alpha_k = \alpha_* > 0$ for all $k \geq \bar{k}_1$. Let $i \notin \mathcal{S}_*$ so that $[x_*]_{\mathcal{G}_i} = 0$. It follows from Assumption 4.3 that $\frac{\alpha_*\lambda_i}{\|[x_* - \alpha_* \nabla f(x_*)]_{\mathcal{G}_i}\|_2} = \frac{\lambda_i}{\|[\nabla f(x_*)]_{\mathcal{G}_i}\|_2} > 1$. Combining this with Theorem 4.9, $\alpha_k = \alpha_* > 0$ for all $k \geq \bar{k}_1$, and Assumption 1.1 gives some $\bar{k}_2 \geq \bar{k}_1$ such that $1 - \alpha_k\lambda_i / \|[x_k - \alpha_k \nabla f(x_k)]_{\mathcal{G}_i}\|_2 < 0$ for all $k \geq \bar{k}_2$. Using this fact with (2.2) and (2.3) yields $[x_k + s_k]_{\mathcal{G}_i} = 0$ for all $k \geq \bar{k}_2$. We are done since $i \notin \mathcal{S}_*$ was arbitrary and $n_{\mathcal{G}}$ is finite. ∎

We now show that, eventually, the set $\mathcal{S}_*$ determines the sets $\mathcal{I}_k^{\mathrm{pg}}$ and $\mathcal{I}_k^{\mathrm{m}}$.

LEMMA 4.11. *For all sufficiently large* k, *it holds that* $\mathcal{I}_k^{\mathrm{pg}} \equiv \{j \in \mathcal{G}_i : i \notin \mathcal{S}_*\}$ *and* $\mathcal{I}_k^{\mathrm{m}} \equiv \{j \in \mathcal{G}_i : i \in \mathcal{S}_*\}$, *where the sets* $\mathcal{I}_k^{\mathrm{pg}}$ *and* $\mathcal{I}_k^{\mathrm{m}}$ *are defined in* (3.4).

Proof. Let $\bar{k}_1$ be large enough so that the conclusion of Lemma 4.10 holds, i.e., if $k \geq \bar{k}_1$ and $i \notin \mathcal{S}_*$, then $[x_k + s_k]_{\mathcal{G}_i} = 0$. Together with (3.2), this shows that $\mathcal{G}_i \cap \bar{\mathcal{I}}_k^{\mathrm{m}} = \emptyset$ for all $k \geq \bar{k}_1$ and $i \notin \mathcal{S}_*$, and thus $\mathcal{G}_i \subseteq \mathcal{I}_k^{\mathrm{pg}}$ (see (3.4)) for all $k \geq \bar{k}_1$ and $i \notin \mathcal{S}_*$. In other words, it holds that $\{j \in \mathcal{G}_i : i \notin \mathcal{S}_*\} \subseteq \mathcal{I}_k^{\mathrm{pg}}$ for all $k \geq \bar{k}_1$.

Next, we prove that there exists $\bar{k}_2$ such that $\mathcal{I}_k^{\mathrm{pg}} \subseteq \{j \in \mathcal{G}_i : i \notin \mathcal{S}_*\}$ for all $k \geq \bar{k}_2$. For a proof by contradiction, suppose that there exist an infinite subsequence $\mathcal{K} \subseteq \mathbb{N}$ and group index $\bar{\imath}$ such that $\mathcal{G}_{\bar{\imath}} \subseteq \mathcal{I}_k^{\mathrm{pg}}$ and $\bar{\imath} \in \mathcal{S}_*$ for all $k \in \mathcal{K}$. Since $\mathcal{G}_{\bar{\imath}} \subseteq \mathcal{I}_k^{\mathrm{pg}}$ for all $k \in \mathcal{K}$, it follows from (3.2), (3.3), and (3.4) that at least one of

$$[x_k]_{\mathcal{G}_{\bar{\imath}}} = 0, \quad [x_k + s_k]_{\mathcal{G}_{\bar{\imath}}} = 0, \quad \|[x_k]_{\mathcal{G}_{\bar{\imath}}}\|_2 < \kappa_1 \|\nabla_{\mathcal{G}_{\bar{\imath}}}(f+r)(x_k)\|_2, \quad \text{or} \tag{4.20}$$

$$\|[x_k]_{\mathcal{G}_{\bar{\imath}}}\|_2 < \kappa_2 \|\nabla_{\bar{\mathcal{I}}_k^{\mathrm{m}}}(f+r)(x_k)\|_2^p \tag{4.21}$$

holds for all $k \in \mathcal{K}$. However, since $\bar{i} \in \mathcal{S}_*$, it follows from Theorem 4.9 that the first condition in (4.20) does not hold for all sufficiently large $k \in \mathcal{K}$. Also, it follows from Theorem 4.9, the facts that $\chi_k^{\rm pg} \equiv \|[s_k]_{\mathcal{I}_k^{\rm pg}}\|_2$ and $\chi_k^{\rm m} \equiv \|[s_k]_{\mathcal{I}_k^{\rm m}}\|_2$, and the fact that $\mathcal{I}_k^{\rm m} \cup \mathcal{I}_k^{\rm pg} = \{1, \ldots, n\}$ that $\lim_{k\to\infty} \|s_k\|_2 = 0$, which combined with $\bar{i} \in \mathcal{S}_*$ proves that $[x_k + s_k]_{\mathcal{G}_{\bar{i}}} \neq 0$ for all sufficiently large k. Hence, the second condition in (4.20) does not hold for all sufficiently large $k \in \mathcal{K}$. Next, from the optimality conditions for problem (1.1), the fact that $\bar{i} \in \mathcal{S}_*$, Theorem 4.9, Assumption 1.1, and the fact that $f + r$ is differentiable over the variables in $\mathcal{G}_{\bar{i}}$ for sufficiently large k that we have $\lim_{k\to\infty} \|\nabla_{\mathcal{G}_{\bar{i}}}(f + r)(x_k)\|_2 = 0$. This limit, $[x_*]_{\mathcal{G}_{\bar{i}}} \neq 0$, and Theorem 4.9 show that $\|[x_k]_{\mathcal{G}_{\bar{i}}}\|_2 \geq \kappa_1 \|\nabla_{\mathcal{G}_{\bar{i}}}(f + r)(x_k)\|_2$ for all sufficiently large k, meaning that the third condition in (4.20) does not hold for all sufficiently large $k \in \mathcal{K}$. Therefore, we must conclude that the inequality in (4.21) holds for all sufficiently large $k \in \mathcal{K}$. Combining this with $\bar{i} \in \mathcal{S}_*$ shows that there exists $\epsilon > 0$ such that

$$\|\nabla_{\bar{\mathcal{I}}_k^{\rm m}}(f + r)(x_k)\|_2 \geq \epsilon > 0 \quad \text{for all sufficiently large } k \in \mathcal{K}, \tag{4.22}$$

which in particular shows that $\bar{\mathcal{I}}_k^{\rm m} \neq \emptyset$ for all sufficiently large $k \in \mathcal{K}$. Since the optimality conditions for problem (1.1) together with Theorem 4.9, Assumption 1.1, and the fact that $f + r$ is differentiable over the variables in $\mathcal{G}_i$ for sufficiently large k imply that $\lim_{k\to\infty} \|\nabla_{\mathcal{G}_i}(f+r)(x_k)\|_2 = 0$ for all $i \in \mathcal{S}_*$, we must conclude from (4.22) that, for all sufficiently large $k \in \mathcal{K}$, there exists an $i_k \notin \mathcal{S}_*$ such that $\mathcal{G}_{i_k} \subseteq \bar{\mathcal{I}}_k^{\rm m}$. However, Lemma 4.10 yields $[x_k + s_k]_{\mathcal{G}_{i_k}} = 0$ for all sufficiently large $k \in \mathcal{K}$, which together with (3.2) shows that $\mathcal{G}_{i_k} \not\subseteq \bar{\mathcal{I}}_k^{\rm m}$, which is a contradiction. Therefore, there exists $\bar{k}_2$ such that $\mathcal{I}_k^{\rm pg} \subseteq \{j \in \mathcal{G}_i : i \notin \mathcal{S}_*\}$ for all $k \geq \bar{k}_2$.

The conclusions of the two previous paragraphs yield $\mathcal{I}_k^{\rm pg} \equiv \{j \in \mathcal{G}_i : i \notin \mathcal{S}_*\}$ for all sufficiently large k. The final assertion, namely, that $\mathcal{I}_k^{\rm m} \equiv \{j \in \mathcal{G}_i : i \in \mathcal{S}_*\}$, follows from the fact that $\mathcal{I}_k^{\rm pg}$ and $\mathcal{I}_k^{\rm m}$ partition $\{1, 2, \ldots, n\}$ for every iteration k. ∎

For k sufficiently large, the support of x_k agrees with the support of the solution.

LEMMA 4.12. *For all sufficiently large k, it holds that $[x_k]_{\mathcal{G}_i} \neq 0$ for all $i \in \mathcal{S}_*$ and $[x_k]_{\mathcal{G}_i} = 0$ for all $i \notin \mathcal{S}_*$.*

Proof. Theorem 4.9 shows that $[x_k]_{\mathcal{G}_i} \neq 0$ for all sufficiently large k and all $i \in \mathcal{S}_*$, which is the first desired result. Hence, let us proceed by considering arbitrary $i \notin \mathcal{S}_*$. Assumption 4.4(ii), Lemma 4.10, Lemma 4.11, and Lemma 4.3 ensure the existence of an iteration $\bar{k}$ such that, for all $k \geq \bar{k}$, the following hold:

$$\mathcal{G}_i \subseteq \mathcal{I}_k^{\rm pg}, \quad [x_k + s_k]_{\mathcal{G}_i} = 0, \quad \text{and } \alpha_k = \alpha_{\bar{k}}. \tag{4.23}$$

We claim that the second desired result follows from (4.23) if there exists some sufficiently large $\hat{k} \geq \bar{k}$ such that $\hat{k} \in \mathcal{K}^{\rm pg}$ and $[x_{\hat{k}+1}]_{\mathcal{G}_i} = [x_{\hat{k}} + s_{\hat{k}}]_{\mathcal{G}_i} = 0$. Indeed, since i is an arbitrary element from $\{1, \ldots, n_{\mathcal{G}}\} \setminus \mathcal{S}_*$, $n_{\mathcal{G}}$ is finite, and the second condition in (4.23) shows that values of the variables in $\mathcal{G}_i$ can only be modified if $k \in \mathcal{K}^{\rm pg}$, the existence of such $\hat{k}$ along with (4.23) shows that iteration $\hat{k} \in \mathcal{K}^{\rm pg}$ sets $[x_{\hat{k}+1}]_{\mathcal{G}_i}$ to zero, and these variables will remain zero for all future iterations.

Let us now show the existence of such $\hat{k} \geq \bar{k}$. We claim that there exists $k \geq \bar{k}$ such that $[x_k]_{\mathcal{G}_i} = 0$. For a proof by contradiction, suppose that $[x_k]_{\mathcal{G}_i} \neq 0$ for all $k \geq \bar{k}$. Combining this with Theorem 4.9, $i \notin \mathcal{S}_*$, and the fact that the variables in $\mathcal{G}_i$ can have their values changed only if $k \in \mathcal{K}^{\rm pg}$ implies that there exists $\hat{k} \geq \bar{k}$ such that $\hat{k} \in \mathcal{K}^{\rm pg}$. Now, since $\hat{k} \in \mathcal{K}^{\rm pg}$ and $\alpha_k = \alpha_{\bar{k}}$ for all $k \geq \bar{k}$, it follows from Algorithm 3.1 that $\text{flag}_{\hat{k}}^{\rm pg} = \texttt{same\_}\alpha$ is returned in line 16. Using this fact, the update used in line 52, and (4.23) shows that $[x_{\hat{k}+1}]_{\mathcal{G}_i} = [x_{\hat{k}} + s_{\hat{k}}]_{\mathcal{G}_i} = 0$. ∎

We require one more lemma that shows that eventually all iterations are in $\mathcal{K}^{\mathrm{m}}_{\mathrm{sd}}$.

LEMMA 4.13. *For all k sufficiently large, it holds that $k \in \mathcal{K}^{\mathrm{m}}_{\mathrm{sd}}$.*

Proof. We first show that all sufficiently large k are in $\mathcal{K}^{\mathrm{m}}$. It follows from Lemma 4.11 that $\mathcal{I}^{\mathrm{pg}}_k \equiv \{j \in \mathcal{G}_i : i \notin \mathcal{S}_*\}$ for all sufficiently large k. Combining this with Lemma 4.12 and Lemma 4.10 shows that there exists an iteration $\bar{k}$ such that $[x_k]_{\mathcal{I}^{\mathrm{pg}}_k} = 0$ and $[x_k + s_k]_{\mathcal{I}^{\mathrm{pg}}_k} = 0$ for all $k \geq \bar{k}$, which means that $\chi^{\mathrm{pg}}_k = \|[s_k]_{\mathcal{I}^{\mathrm{pg}}_k}\|_2 = 0$ for all $k \geq \bar{k}$. It follows from this fact, line 7, and Assumption 4.1 that $k \in \mathcal{K}^{\mathrm{m}}$ for all $k \geq \bar{k}$. Now, notice that at most $n_{\mathcal{G}} - 1$ iterations from $\bar{k}$ onward can be in $\mathcal{K}^{\mathrm{m}}_0$ because of Lemma 4.6(i). (Every iteration $k \in \mathcal{K}^{\mathrm{m}}_0$ fixes at least one new group of variables to zero, and if they ever all become zero so that $\mathcal{I}^{\mathrm{m}}_k = \emptyset$, then the contradiction $k \in \mathcal{K}^{\mathrm{pg}}$ is reached.) Therefore, it follows that all sufficiently large k must be in $\mathcal{K}^{\mathrm{m}}_{\mathrm{sd}}$. ∎

We can now state our main local convergence result.

THEOREM 4.14. *If in Algorithm 3.2 we choose either $q \in (1, 2]$, or $q = 1$ and $\{\mu_k\} \to 0$, then $\{x_k\} \to x_*$ at a superlinear rate. In particular, if we choose $q = 2$, then the rate of convergence is quadratic.*

Proof. It follows from Lemma 4.11, Lemma 4.12, and Lemma 4.13 that, for all sufficiently large k, the iterates generated by Algorithm 3.1 satisfy the recurrence $x_{k+1} = x_k + \xi^{j_k} d_k$, where j_k is the result of the backtracking Armijo line search in line 43, $\|[x_k]_{\mathcal{I}^{\mathrm{pg}}_k}\|_2 = \|[d_k]_{\mathcal{I}^{\mathrm{pg}}_k}\|_2 = 0$, and $[d_k]_{\mathcal{I}^{\mathrm{m}}_k} = \bar{d}_k$ with $\bar{d}_k$ computed by Algorithm 3.2 to satisfy the third condition in line 22. In other words, for all sufficiently large k, we have $[x_k]_{\mathcal{I}^{\mathrm{pg}}_k} = [x_*]_{\mathcal{I}^{\mathrm{pg}}_k} = 0$, and the variables in $\mathcal{I}^{\mathrm{m}}_k \equiv \{j \in \mathcal{G}_i : i \in \mathcal{S}_*\}$ are updated exactly as those of an inexact Newton method for computing a root of $\nabla_{\mathcal{I}^{\mathrm{m}}_k}(f + r)$). Since, by Theorem 4.9, we have $\lim_{k\to\infty} x_k = x_*$, the desired conclusions follow under the stated conditions from [12, Theorem 3.3] (also recall the local strong convexity restricted to the support of x_* in Assumption 4.3(ii)) and noting the well-known result that the unit step size $\xi^{j_k} = 1$ is accepted (asymptotically) by a backtracking Armijo line search when $\eta \in (0, 1/2)$ (see Assumption 4.4(i)). ∎

5. Numerical results. In this section, we present the results of numerical experiments with an implementation of `FaRSA-Group` (Algorithm 3.1) applied to solve two classes of regularized regression problems. The first class is the regularized logistic regression problem of the form $\min_{x\in\mathbb{R}^n} \frac{1}{N}\sum_{i=1}^{N} \log(1 + e^{-y_i x^T d_i}) + \sum_{i=1}^{n_{\mathcal{G}}} \lambda_i \|[x]_{\mathcal{G}_i}\|_2$, where $d_i \in \mathbb{R}^n$ is the ith data point, N is the number of data points, $y_i \in \{-1, 1\}$ is the class label for the ith data point, and λ_i is the weight for the ith group. The second problem is the regularized linear regression problem $\min_{x\in\mathbb{R}^n} \frac{1}{N}||Ax - b||_2^2 + \sum_{i=1}^{n_{\mathcal{G}}} \lambda_i \|[x]_{\mathcal{G}_i}\|_2$, where $A \in \mathbb{R}^{N\times n}$, $b \in \mathbb{R}^N$, and λ_i is the weight for the ith group.

We first describe details of our implementation of `FaRSA-Group`, then describe the data sets considered in our experiments, and finally present our experimental results.

5.1. Implementation details. We have developed a Python implementation of `FaRSA-Group` that is available upon request. The values of the input parameters for Algorithm 3.1 and Algorithm 3.2 that we used are given as follows (with some caveats that are mentioned in the following paragraph): $\varphi = 1$, $\kappa_1 = 0.1$, $\xi = 0.5$, $\kappa_2 = 10^{-2}$, $\eta = 10^{-3}$, $\theta = \pi/4$, $\zeta = 0.8$, $q = 1.5$, $p = 2$, and $\mu_k = 1$.

We initialized x_0 as the zero vector and α_0 as an estimate of the inverse of the Lipschitz constant of f at x_0. To be precise, our software randomly generated $y_0 \in \mathbb{R}^n$ such that $\|x_0 - y_0\|_2 = 10^{-8}$ and then set $\alpha_0 = \min\{1, \|x_0 - y_0\|_2/\|\nabla f(x_0) - \nabla f(y_0)\|_2\}$. Since $\varphi = 1$, it follows from Algorithm 3.1 that Assumption 4.4(ii) holds for all $k \in \mathbb{N}$. (For data sets with $N < n$, we initially chose $\varphi = 0.8$ and switched

to $\varphi = 1$ when an iteration in $\mathcal{K}^{\mathrm{m}}$ satisfied $f(x_k) - f(x_{k+1}) \leq 10^{-3}$. When $N < n$, the matrix $\nabla^2 f(x_k)$ is singular, which often led to large CG directions and multiple backtracks in the line search. These ill effects were partly remedied by this scheme for updating φ.) When defining the set $\mathcal{I}_k^{\mathrm{small}}$ in (3.3), we used $\tilde{\kappa}_{2,i} = \kappa_2 |\mathcal{G}_i| / \|\bar{\mathcal{I}}_k^{\mathrm{m}}\|$ in place of κ_2 for all i such that $\mathcal{G}_i \subseteq \bar{\mathcal{I}}_k^{\mathrm{m}}$ to account for the fact that the two different norms in (3.3) are associated with vectors of different dimension. Note that since $(1/n)\kappa_2 \leq \tilde{\kappa}_{2,i} \leq n\kappa_2$, this choice is easily incorporated into the analysis in section 4.

Instead of fixing κ_1 and κ_2, they are adaptively adjusted by the following rules to improve `FaRSA-Group`'s performance. (i) If the kth iteration finishes at line 44 with $j > 5$, then increase κ_1 and κ_2: $\kappa_1 \leftarrow \min\{10^6, 10\kappa_1\}$ and $\kappa_2 \leftarrow \min\{10^5, 10\kappa_2\}$. (ii) If for $k > 0$ we see that $k+i \in \mathcal{K}^{\mathrm{pg}}$ for all $i \in \{0, 1, 2, \ldots, 5\}$, then at the end of iteration $k + 5$ decrease κ_1 and κ_2: $\kappa_1 \leftarrow \max\{10^{-5}, \kappa_1/10\}$ and $\kappa_2 \leftarrow \max\{10^{-6}, \kappa_2/10\}$.

The first rule aims to keep more groups that are potentially zero at the solution out of the set $\mathcal{I}_{k+1}^{\mathrm{m}}$. In particular, it is driven by our empirical observation that when significant backtracking along the m-direction is performed in the reduced space, it is likely that groups of variables that are zero at the solution and nearly zero at x_k are (wrongly) included in $\mathcal{I}_k^{\mathrm{m}}$. As for the second rule, the idea is to increase the chance that Algorithm 3.2 is used during the next iteration, with the hope that it accelerates convergence. This rule is also guided by our numerical experience.

The choice of H_k in line 9 was a regularization of the exact second-derivatives of f. For the logistic regression problem, for any scalar $\delta \geq 0$, $\frac{1}{N} D^T \Sigma_\delta(x) D \approx \nabla^2 f(x)$, where $D^T := [d_1, d_2, \ldots, d_N]$ and $\Sigma_\delta(x)$ is the diagonal matrix with ith diagonal entry $[\Sigma_\delta(x)]_{ii} := \max\{\sigma_i(x)(1 - \sigma_i(x)), \delta\}$ with $\sigma_i(x) := \exp(y_i d_i^T x) / \big(1 + \exp(y_i d_i^T x)\big)$ for all $i \in \{1, 2, \ldots, N\}$. Notice that if $\delta = 0$, then $(1/N) D^T \Sigma_0(x) D \equiv \nabla^2 f(x)$. To use a small amount of regularization in our tests, we chose $\delta = 10^{-8}$. With this choice of δ, our choice of H_k in line 9 can now be written as $H_k \leftarrow [\frac{1}{N} D^T \Sigma_\delta(x_k) D]_{\mathcal{I}_k \mathcal{I}_k} + \nabla^2_{\mathcal{I}_k \mathcal{I}_k} r(x_k)$, where we remind the reader that $\nabla^2_{\mathcal{I}_k \mathcal{I}_k} r(x_k)$ is well defined because $\mathcal{I}_k \subseteq \mathcal{I}_k^{\mathrm{m}}$ ensures that $[x_k]_{\mathcal{G}_i} \neq 0$ for all $\mathcal{G}_i \subseteq \mathcal{I}_k$. For the linear regression problem, we set $H_k \leftarrow [A^T A]_{\mathcal{I}_k \mathcal{I}_k} + \nabla^2_{\mathcal{I}_k \mathcal{I}_k} r(x_k) + \delta I$, where I is the identity matrix.

In Algorithm 3.2, we applied the CG method to the system $H_k d = -g_k$ to approximately solve the optimization problem defined in line 22. As pointed out in section 3.2, the direction associated with every iteration of the CG algorithm satisfies the first two conditions in line 22, which were required to establish the complexity result in Theorem 4.8. To reduce the cost of the CG computation and limit the number of backtracking steps required by Algorithm 3.3, we terminated Algorithm 3.2 when at least one of three conditions was satisfied. To describe these conditions checked during the kth iteration, let $d_{j,k}$ denote the jth CG iterate, and let $t_{j,k} := \|H_k d_{j,k} + g_k\|_2$ denote the jth CG residual. The three conditions are given as follows: (i) $t_{j,k} \leq \max\{\min\{0.1 t_{0,k}, t_{0,k}^{1.5}\}, 10^{-10}\}$, (ii) $\|d_{j,k}\| \geq 10^3 \min\{1, \|\nabla_{\mathcal{I}_k}(f + r)(x_k)\|_2\}$, and (iii) $j = |\mathcal{I}_k|$. Outcome (i) is the ideal termination condition since it indicates that the residual of the linear system has been sufficiently reduced. Outcome (ii) serves as a trust-region constraint on the norm of the trial step d_k; in particular, when this inequality holds, the size of the CG iterate $d_{j,k}$ is relatively large, indicating that x_k is not close to an optimal solution. Therefore, we restrict its size with the intent of needing fewer backtracking steps during the subsequent line search. Outcome (iii) caps the number of CG iterations to $|\mathcal{I}_k|$ (the size of the reduced space) since, in exact arithmetic, CG converges to an exact solution in at most $|\mathcal{I}_k|$ iterations.

Algorithm 3.1 decreases the value of the PG parameter (see line 17) for the next iteration using a simple multiplicative factor when $\mathrm{flag}_k^{\mathrm{pg}} =$ `decrease_`α. However, in practice, we found an adaptation of the approach in [10] to be more efficient. To

describe this approach, let d_k and ξ^{j_k} be the search direction and step size used to obtain $x_{k+1} = x_k + \xi^{j_k} d_k$. It is well known [2, Lemma 5.7] that if $\alpha \in (0, 1/L_f]$, then $f(x_{k+1}) \leq f(x_k) + \xi^{j_k} \nabla f(x_k)^T d_k + \frac{1}{2\alpha} \|\xi^{j_k} d_k\|_2^2$. Setting this inequality to be an equality and then solving for α, one obtains $\hat{\alpha}_k := \frac{\|\xi^{j_k} d_k\|_2^2}{2\left(f(x_{k+1}) - f(x_k) - \xi^{j_k} \nabla f(x_k)^T d_k\right)}$, which can be viewed as a local Lipschitz constant estimate for f at x_k. In our tests, we updated the PG parameter at the end of each iteration of Algorithm 3.1 as $\alpha_{k+1} \leftarrow \min\{1, \hat{\alpha}_k/2\}$. Although this PG parameter update strategy worked better than the basic strategy in Algorithm 3.1 (see line 17 and line 17), it is not covered by our analysis in section 4. However, a simple modification of our analysis would be to allow this update to increase the PG parameter at most a finite number of times, say 100 times, at which point the update $\alpha_{k+1} \leftarrow \min\{\alpha_k, \hat{\alpha}_k/2\} \leq \alpha_k$ would be used. This strategy is covered by our earlier analysis (with a larger constant in the complexity result). The algorithm is terminated when at least one of the following conditions holds: (i) $\max\{\chi_k^{\mathrm{m}}, \chi_k^{\mathrm{pg}}\} \leq 10^{-6} \max\{\chi_0^{\mathrm{m}}, \chi_0^{\mathrm{pg}}, 1\}$. (ii) $k \in \mathcal{K}_{\mathrm{sd}}^{\mathrm{m}}$ and $|\nabla_{\mathcal{I}_k}(f+r)(x_k)^T [d_k]_{\mathcal{I}_k}| / (1 + f(x_k) + \lambda r(x_k)) < 10^{-16}$. (iii) The maximum allowed time limit is reached. (iv) The maximum allowed number of iterations is reached. Condition (i) implies the the algorithm terminates with the desired accuracy, while condition (ii) indicates that (numerically) no significant progress can be made.

5.2. Data sets. Data sets for the logistic regression problems were obtained from the LIBSVM repository.[1] We excluded all multiclass (greater than two) classification instances and all data sets that were too large ($\geq$ 8GB) to be loaded in memory. Finally, for the adult data (a1a–a9a) and webpage data (w1a–w8a), we used only the largest instances, namely, a9a and w8a. This left us with our final subset of 30 data sets that can be found in the top part of Table 5.1. For linear regression problems, we tested `FaRSA-Group` using all regression data sets from the LIBSVM repository and all regression data sets with more than 10000 samples from the University of California Irvine (UCI) Machine Learning Repository.[2]

Scaling of the data sets can be important. If the source of the data indicated that a data set was already scaled, then we used the data without modification. However, when the website did not indicate that scaling for a data set was used, we scaled each column of the feature data (i.e., featurewise scaling) into the range $[-1, 1]$ by dividing each of its entries by the largest entry in absolute value. Labels for some data sets (e.g., breast-cancer, covtype, liver-disorders, mushrooms, phishing, skin-nonskin and svmguide1) do not take values in $\{-1, 1\}$ but rather in $\{0, 1\}$ or $\{1, 2\}$. For these data sets, we mapped the smaller label to -1 and the larger label to 1.

5.3. Experimental setup and test results. For both problem classes, we considered four group structures and two different solution sparsity levels. Specifically, we considered the four different numbers of groups in $\{\lfloor 0.25n \rfloor, \lfloor 0.50n \rfloor, \lfloor 0.75n \rfloor, n\}$, where n is the problem dimension; notice that the last setting recovers ℓ_1-norm regularization. Then, for a given number of groups, the variables were sequentially distributed (as evenly as possible) to the groups; e.g., 10 variables among 3 groups would have been distributed as $\mathcal{G}_1 = \{1, 2, 3\}$, $\mathcal{G}_2 = \{4, 5, 6\}$, and $\mathcal{G}_3 = \{7, 8, 9, 10\}$. For the two different solution sparsity levels, we considered groups weights $\lambda_i = 0.1\lambda_{\min}\sqrt{|\mathcal{G}_i|}$ and $\lambda_i = 0.01\lambda_{\min}\sqrt{|\mathcal{G}_i|}$, where $\lambda_{\min}$ is the minimum positive λ such that the solution to the logistic problem with $\lambda_i = \lambda\sqrt{|\mathcal{G}_i|}$ is $x = 0$ (see [33, equation (23)]).

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets.

[2] https://archive.ics.uci.edu/ml/index.php.

TABLE 5.1

The first column (Data set) gives the name of the data set. The second column (N) and third column (n) indicate the number of data points and problem dimension, respectively. The fourth column (Scale) provides the featurewise scaling used: each feature is either scaled into the given interval or scaled to have mean zero ($\mu = 0$) and variance one ($\sigma^2 = 1$). The fifth column (Who) indicates whether the data set came prescaled from the LIBSVM website (website), or it did not come prescaled and we scaled it (us) as described in section 5.2.

Data set	N	n	Scale	Who
Data sets for the logistic regression problems				
a9a	32561	123	[0,1]	website
australian	690	140	[-1,1]	website
breast-cancer	683	10	[-1,1]	website
cod-rna	59535	8	[-1,1]	us
colon-cancer	62	2000	$(\mu, \sigma^2) = (0, 1)$	website
covtype.binary	581012	54	[0,1]	website
diabetes	768	8	[-1,1]	website
duke breast-cancer	44	7192	$(\mu, \sigma^2) = (0, 1)$	website
fourclass	862	2	[-1,1]	website
german-numer	1000	24	[-1,1]	website
gisette	6000	5000	[-1,1]	website
heart	270	13	[-1,1]	website
HIGGS	11000000	28	[-1,1]	us
ijcnn1	49990	22	[-1.5, 1.5]	website
ionosphere	351	34	[-1,1]	website
leukemia	38	7129	$(\mu, \sigma^2) = (0, 1)$	website
liver-disorders	145	5	[-1,1]	website
madelon	2000	500	[-1,1]	us
mushrooms	8124	112	[0,1]	website
news20.binary	19996	1355191	[0,1]	website
phishing	11055	68	[0,1]	website
rcv1.binary	20242	47236	[0,1]	website
real-sim	72309	20958	[0,1]	website
skin-nonskin	245057	3	[-1,1]	us
splice	1000	60	[-1,1]	website
sonar	208	60	[-1,1]	website
svmguide1	3089	4	[-1,1]	us
svmguide3	1243	21	[-1,1]	website
SUSY	5000000	18	[-1,1]	us
w8a	49749	300	[0,1]	website
Data sets for the linear regression problems				
abalone	4177	8	[-1,1]	website
blogData	60021	281	[-1,1]	us
bodyfat	252	14	[-1,1]	website
cadata	20640	8	[-1,1]	us
cpusmall	8192	12	[-1,1]	website
driftData	13910	128	[-1,1]	us
eunite2001	336	31	[-1,1]	us
E2006.tfidf	16087	150360	[-1,1]	website
housing	506	13	[-1,1]	website
mg	1385	6	[-1,1]	website
mpg	393	7	[-1,1]	website
pyrim	74	27	[-1,1]	website
space_ga	3107	6	[-1,1]	website
triazines	186	60	[-1,1]	website
UJIIndoorLoc	19937	520	[-1,1]	us
VirusShare	107888	482	[-1,1]	us
YearPredictionMSD	463715	90	[-1,1]	website

The experiments were conducted on the Computational Optimization Research Laboratory (COR@L) cluster at Lehigh University with an AMD Opteron Processor 6128 2.0 GHz CPU. In the following, we compared performance of different algorithms with respect to CPU time (seconds), final objective value, and solution sparsity.

5.3.1. Logistic regression. In this section, we compare the performance of `FaRSA-Group` to `APG` ([3] adjusted to the group ℓ_1-norm), `gglasso` [33], and `PNOPT` [19] for solving logistic regression problems using the data sets in the upper part of Table 5.1; a total of 240 problem instances are tested as described above.[3] `APG` is an established accelerated PG method, and our Python implementation is based on that of Templates for First-Order Conic Solvers (TFOCS) [4]. `gglasso` is a state-of-the-art groupwise majorization descent method for which an R implementation is available with the most computationally expensive steps performed in Fortran.[4] `PNOPT` is a (quasi) proximal-Newton method that has a freely available MATLAB implementation that allows users to use either the exact Hessian matrix or (limited-memory) BFGS approximations; we test both options and refer to them as `PNOPT-Newton` and `PNOPT-LBFGS`, respectively.[5] Default parameters for `APG`, `gglasso`, and `PNOPT` are used, and x_0 is chosen as the zero vector for all algorithms and problem instances. `APG` and `PNOPT` use the same termination conditions as used in `FaRSA-Group`, but for `gglasso` we use its default termination rules since we have no control over its termination criteria through the application programming interface (API) that it provides.

In terms of running time, for each problem instance we allow a maximum of 3600 seconds. If the CPU time surpasses this limit on a problem, we terminate the run and consider the algorithm to have failed. Out of the 240 problem instances (200 for `gglasso`), `FaRSA-Group` successfully solved all instances while `APG`, `gglasso`, `PNOPT-Newton`, `PNOPT-LBFGS` failed to solve 16, 4, 13, and 26 instances, respectively. Figure 5.1 illustrates performance profiles based on [24] for comparing the computing times on problem instances that `FaRSA-Group` and/or a competing algorithm took at least 1 second to terminate. Each bar in the plot corresponds to a problem instance, with the height of the bar given by

$$-\log_2\left(\frac{\text{time required by } \texttt{FaRSA-Group}}{\text{time required by a competing algorithm}}\right). \tag{5.1}$$

Therefore, an upward pointing bar indicates that `FaRSA-Group` took less time to find the optimal solution for that problem instance, and a downward pointing bar means that the competing algorithm took less time, and in either case the size of the bar indicates the magnitude of the outperformance factor. A bar that reaches the y-axis limit is used when indicating that an algorithm was successful when solving a problem instance while the competing algorithm was unsuccessful.

For final objective function values, let $F_{\texttt{FaRSA-Group}}$ and $F_{\text{competing}}$ denote (for a given problem instance) the final objective values returned by `FaRSA-Group` and the competing algorithm, respectively. If $F_{\texttt{FaRSA-Group}} - F_{\text{competing}} < -10^{-6}$, then we consider `FaRSA-Group` to have obtained a lower objective function value; if $F_{\texttt{FaRSA-Group}} - F_{\text{competing}} > 10^{-6}$, then we consider the competing algorithm to have obtained a lower objective function value; and if $|F_{\texttt{FaRSA-Group}} - F_{\text{competing}}| \leq 10^{-6}$, then we consider them to have performed equally. For solution sparsity, we consider `FaRSA-Group` to

[3] Only 200 problem instances are tested for `gglasso` since it does not support sparse data matrix inputs. In particular, data sets HIGGS, news20.binary, rcv1.binary, real-sim, and SUSY are excluded.

[4] https://cran.r-project.org/web/packages/gglasso/gglasso.pdf.

[5] https://web.stanford.edu/group/SOL/software/pnopt/.

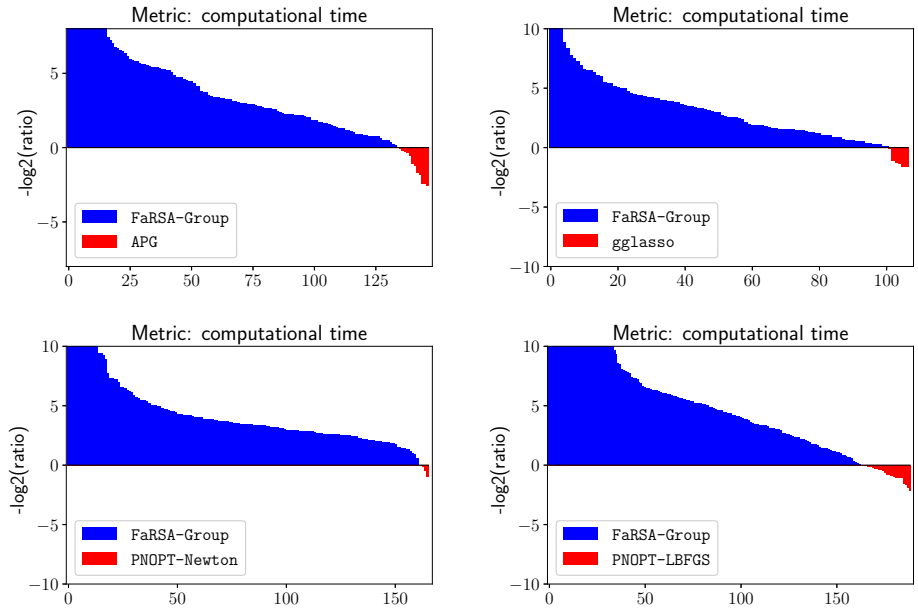

FIG. 5.1. *Performance profile for CPU time (seconds).* `FaRSA-Group` *outperforms* `APG`, `gglasso`, `PNOPT-Newton`, *and* `PNOPT-LBFGS` *on* 135 *of the* 147 *problem instances,* 101 *out of the* 107 *problem instances,* 161 *out* 166 *problem instances, and* 163 *out of the* 189 *problem instances, respectively. For each problem instance, the height of the bar is given by* (5.1).

TABLE 5.2

Solution qualities in terms of the final objective function value and the solution sparsity. The numbers under the columns labeled worse, same, and better are the number of problems for which `FaRSA-Group` *performs worse/same/better compared to the competing algorithm listed to the left. For example, comparing* `FaRSA-Group` *with* `gglasso`*, we see that out of the* 107 *test instances,* `FaRSA-Group` *performs worse, the same, and better on* 0, 40, *and* 67 *instances, respectively.*

	Total	`FaRSA-Group` obj.			`FaRSA-Group` sparsity		
Competing algorithm	# probs	Worse	Same	Better	Worse	Same	Better
`APG`	147	2	136	9	4	133	10
`gglasso`	107	0	40	67	5	74	28
`PNOPT-Newton`	166	3	154	9	6	145	15
`PNOPT-LBFGS`	189	3	171	15	6	161	22

have outperformed a competing algorithm if the following two conditions hold: All zero groups in the solution returned by the competing algorithm solution are also zero groups in the `FaRSA-Group` solution, and the solution returned by `FaRSA-Group` has at least one zero group that is not a zero group in the competing algorithm's solution; a similar criteria is used to define when a competing algorithm is considered to have outperformed `FaRSA-Group`. The results are summarized in Table 5.2.

5.3.2. Linear regression. In this section we report the results of our tests on the linear regression problems. In addition to `APG`, `gglasso`, and `PNOPT`, we also compare `FaRSA-Group` to `SSNAL` [37], which is a state-of-the-art semismooth Newton method designed to solve group-sparse lasso problems. For these tests, we use the data sets in the bottom part of Table 5.1, from which we obtain a total of 136 test instances

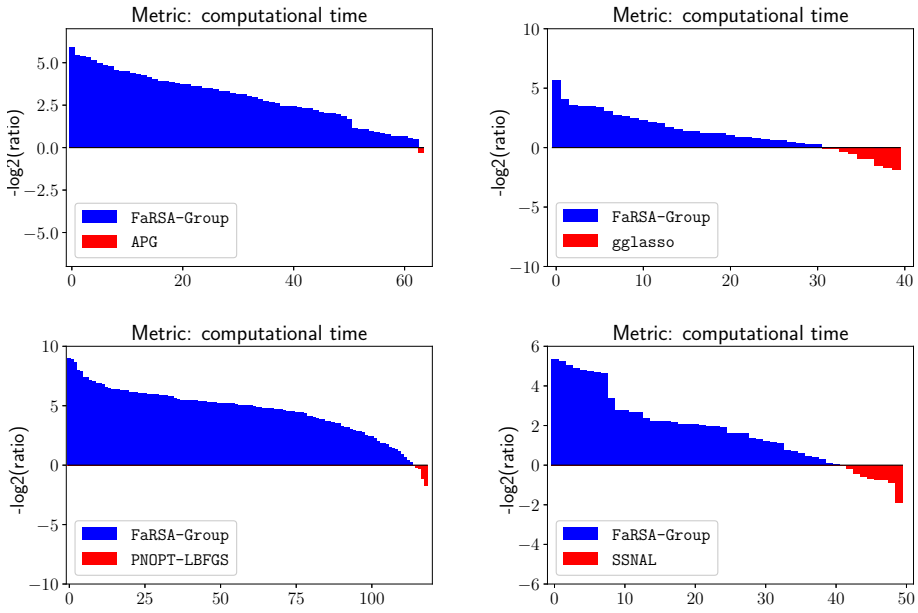

FIG. 5.2. *Performance profile for CPU time (seconds).* `FaRSA-Group` *outperforms* `APG`, `gglasso`, `PNOPT-LBFGS`, *and* `SSNAL` *on* 63 *out of the* 64, 31 *out of the* 40, 115 *out of the* 120, *and* 41 *out of the* 50 *test instances, respectively. For each problem instance, the height of the bar is given by* (5.1). *Results for* `PNOPT-Newton` *are not shown because it performs worse than* `PNOPT-LBFGS`.

as described in the beginning of section 5.3.[6] Default parameters for `SSNAL` are used, and the initial estimate x_0 is chosen to be the zero vector for both `FaRSA-Group` and `SSNAL`. We note that `SSNAL` is a dual method that terminates when the primal-dual gap is smaller than the default tolerance of 10^{-6}, which is different than the stopping condition implemented in `FaRSA-Group`. Of the 136 test instances, `FaRSA-Group` terminated 10 times because termination condition (ii) was triggered, meaning that no additional sufficient progress could be achieved.

We measure algorithm performance using the same criteria as for the tests in section 5.3.1. All methods successfully solve all instances, and Figure 5.2 illustrates performance profiles for the computational times for problem instances for which `FaRSA-Group` and/or the competing algorithm takes at least 1 second to terminate. The final objective values and solution sparsities are summarized in Table 5.3.

We remark that since `SSNAL` is a dual method, it tends to be more efficient than `FaRSA-Group` when the data matrix $A \in \mathbb{R}^{N \times n}$ satisfies $n > N$. For example, `SSNAL` often outperforms `FaRSA-Group` on the problem instances considered in [37].

6. Conclusion. We presented a new framework for solving optimization problems that incorporate group sparsity-inducing regularization by using subspace acceleration, domain decomposition, and support identification. In terms of theory, we proved a complexity result on the maximum number of iterations before an ϵ-approximate solution is computed (Theorem 4.8), and a local superlinear convergence rate (Theorem 4.14). The strong convergence theory was supported by experimental

[6] For `gglasso`, since its software does not support sparse matrix inputs, only 128 problem instances are tested. Specifically, all instances of data set E2006.tfidf are excluded.

TABLE 5.3
The meaning of this table is the same as Table 5.2 but for the linear regression problems.

	Total	`FaRSA-Group` obj.			`FaRSA-Group` sparsity		
Competing algorithm	# probs	Worse	Same	Better	Worse	Same	Better
`APG`	64	0	26	38	0	64	0
`gglasso`	40	0	27	13	0	40	0
`PNOPT-Newton`	98	0	72	26	0	98	0
`PNOPT-LBFGS`	119	1	93	25	0	119	0
`SSNAL`	50	0	45	5	0	50	0

results for minimizing a group sparsity-regularized logistic function for the task of classification and a group sparsity-regularized least-squares function for the task of regression. In terms of robustness, computational time, final objective value obtained, and solution sparsity, the numerical results showed that our proposed `FaRSA-Group` framework outperforms state-of-the-art methods, especially when the data set is larger than the number of features in the model. When the number of features that define the model is larger than the size of the data set, then `FaRSA-Group` still works well, but methods based on a dual approach may be preferable.

REFERENCES

[1] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, *Optimization with sparsity-inducing penalties*, Found. Trends Mach. Learn., 4 (2012), pp. 1–106.

[2] A. Beck, *First-Order Methods in Optimization*, SIAM, Philadelphia, 2017.

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

[4] S. R. Becker, E. J. Candès, and M. C. Grant, *Templates for convex cone problems with applications to sparse signal recovery*, Math. Program. Comput., 3 (2011), p. 165.

[5] D. P. Bertsekas, *Convex Optimization Theory*, Athena Scientific, Belmont, MA, 2009.

[6] T. Chen, F. E. Curtis, and D. P. Robinson, *A reduced-space algorithm for minimizing ℓ_1-regularized convex functions*, SIAM J. Optim., 27 (2017), pp. 1583–1610.

[7] T. Chen, F. E. Curtis, and D. P. Robinson, *FaRSA for ℓ_1-regularized convex optimization: Local convergence and numerical experience*, Optim. Methods Softw., 33 (2018), pp. 396–415.

[8] P. Combettes and J.-C. Pesquet, *Proximal splitting methods in signal processing*, in Fixed-Point Algorithms for Inverse Problems in Science and Engineering, Springer, Cham, 2011, pp. 185–212.

[9] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Trust-Region Methods*, SIAM, Philadelphia, 2000.

[10] F. E. Curtis and D. P. Robinson, *Exploiting negative curvature in deterministic and stochastic optimization*, Math. Program., 176 (2019), pp. 69–94.

[11] I. Daubechies, M. Defrise, and C. Mol, *An iterative thresholding algorithm for linear inverse problems with a sparsity constraint*, Comm. Pure Appl. Math., 58 (2004), pp. 1413–1457.

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

[13] D. Donoho, *Denoising by soft-thresholding*, IEEE Trans. Inform. Theory, 41 (1995), pp. 613–627.

[14] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, *LIBLINEAR: A library for large linear classification*, J. Mach. Learn. Res., 9 (2008), pp. 1871–1874.

[15] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright, *Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems*, IEEE J. Sel. Top. Signal Process., 1 (2007), pp. 586–597.

[16] J. Friedman, T. Hastie, and R. Tibshirani, *Regularization paths for generalized linear models via coordinate descent*, J. Statist. Softw., 33 (2010), p. 1.

[17] G. N. Grapiglia and Y. Nesterov, *Accelerated regularized Newton methods for minimizing composite convex functions*, SIAM J. Optim., 29 (2019), pp. 77–99.

[18] N. Keskar, J. Nocedal, F. Oztoprak, and A. Wächter, *A second-order method for convex ℓ_1-regularized optimization with active-set prediction*, Optim. Methods Softw., 31 (2016), pp. 605–621.
[19] J. D. Lee, Y. Sun, and M. A. Saunders, *Proximal Newton-type methods for minimizing composite functions*, SIAM J. Optim., 24 (2014), pp. 1420–1443.
[20] Q. Lin, Z. Lu, and L. Xiao, *An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization*, SIAM J. Optim., 25 (2015), pp. 2244–2273.
[21] J. Liu, S. Ji, and J. Ye, *SLEP: Sparse Learning with Efficient Projections*, Arizona State University, 2009, http://yelabs.net/software/SLEP/.
[22] J. Liu and S. J. Wright, *Asynchronous stochastic coordinate descent: Parallelism and convergence properties*, SIAM J. Optim., 25 (2015), pp. 351–376.
[23] S. Ma, X. Song, and J. Huang, *Supervised group lasso with applications to microarray data analysis*, BMC Bioinform., 8 (2007), p. 60.
[24] J. L. Morales, *A numerical study of limited memory BFGS methods*, Appl. Math. Lett., 15 (2002), pp. 481–487.
[25] B. S. Mordukhovich and N. M. Nam, *An Easy Path to Convex Analysis and Applications*, Morgan & Claypool Publishers, Williston, VT, 2013.
[26] Y. Nesterov, *A method of solving a convex programming problem with convergence rate $\mathcal{O}(1/k^2)$*, Soviet Math. Dokl., 27 (1983), pp. 372–376.
[27] Y. Nesterov, *Gradient methods for minimizing composite functions*, Math. Program., 140 (2013), pp. 125–161.
[28] J. Nutini, M. Schmidt, and W. Hare, *Active-set complexity of proximal gradient: How long does it take to find the sparsity pattern?*, Optim. Lett., 13 (2019), pp. 645–655.
[29] P. Richtárik and M. Takáč, *Parallel coordinate descent methods for big data optimization*, Math. Program., 156 (2016), pp. 433–484.
[30] R. Tappenden, P. Richtárik, and J. Gondzio, *Inexact coordinate descent: Complexity and preconditioning*, J. Optim. Theory Appl., 170 (2016), pp. 144–176.
[31] S. J. Wright, *Accelerated block-coordinate relaxation for regularized optimization*, SIAM J. Optim., 22 (2012), pp. 159–186.
[32] S. J. Wright, R. D. Nowak, and M. A. Figueiredo, *Sparse reconstruction by separable approximation*, IEEE Trans. Signal Process., 57 (2009), pp. 2479–2493.
[33] Y. Yang and H. Zou, *A fast unified algorithm for solving group-lasso penalize learning problems*, Stat. Comput., 25 (2015), pp. 1129–1141.
[34] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, *An improved GLMNET for ℓ_1-regularized logistic regression*, J. Mach. Learn. Res., 13 (2012), pp. 1999–2030.
[35] M. Yuan and Y. Lin, *Model selection and estimation in regression with grouped variables*, J. R. Stat. Soc. Ser. B Stat. Methodol., 68 (2006), pp. 49–67.
[36] Y. Zeng and P. Breheny, *Overlapping group logistic regression with applications to genetic pathway selection*, Cancer Inform., 15 (2016), pp. CIN–S40043.
[37] Y. Zhang, N. Zhang, D. Sun, and K.-C. Toh, *An efficient Hessian based algorithm for solving large-scale sparse group lasso problems*, Math. Program., 179 (2020), pp. 223–263.