SIAM J. OPTIM.
Vol. 26, No. 3, pp. 1773–1798

PROJECTION ONTO A POLYHEDRON THAT EXPLOITS SPARSITY*

WILLIAM W. HAGER† AND HONGCHAO ZHANG‡

Abstract. An algorithm is developed for projecting a point onto a polyhedron. The algorithm solves a dual version of the projection problem and then uses the relationship between the primal and dual to recover the projection. The techniques in the paper exploit sparsity. Sparse reconstruction by separable approximation (SpaRSA) is used to approximately identify active constraints in the polyhedron, and the dual active set algorithm (DASA) is used to compute a high precision solution. A linear convergence result is established for SpaRSA that does not require the strong concavity of the dual to the projection problem, and an earlier R-linear convergence rate is strengthened to a Q-linear convergence property. An algorithmic framework is developed for combining SpaRSA with an asymptotically preferred algorithm such as DASA. It is shown that only the preferred algorithm is executed asymptotically. Numerical results are given using the polyhedra associated with the Netlib LP test set. A comparison is made to the interior point method contained in the general purpose open source software package IPOPT for nonlinear optimization, and to the commercial package CPLEX, which contains an implementation of the barrier method that is targeted to problems with the structure of the polyhedral projection problem.

Key words. polyhedral projection, SpaRSA, active set algorithm, dual active set algorithm, DASA, multilevel optimization

AMS subject classifications. 90C06, 90C20, 90C25, 65Y20

DOI. 10.1137/15M102825X

1. Introduction.

A dual approach is developed for computing the Euclidean projection of a point $\mathbf{y} \in \mathbb{R}^n$ onto a nonempty polyhedron $\Omega \subset \mathbb{R}^n$. Computing the projection is equivalent to solving the optimization problem

$$\min \left\{ \frac{1}{2}\|\mathbf{y} - \mathbf{x}\|^2 : \mathbf{x} \in \Omega \right\}, \tag{1.1}$$

where $\|\cdot\|$ is the Euclidean norm. Note that any quadratic programming problem whose objective's Hessian is diagonal with positive entries can be written in the form (1.1). To simplify the exposition, it is assumed that the polyhedron has the form

$$\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{l} \le \mathbf{A}\mathbf{x} \le \mathbf{u}, \quad \mathbf{x} \ge \mathbf{0}\}, \tag{1.2}$$

where $\mathbf{l}$ and $\mathbf{u} \in \mathbb{R}^n$ are given bounds with $\mathbf{l} \le \mathbf{u}$ and $\mathbf{A} \in \mathbb{R}^{m\times n}$ is a given (nonvacuous) matrix with nonzero columns. Of course, when $\mathbf{A}$ is vacuous, the projection (1.1)

*Received by the editors June 29, 2015; accepted for publication (in revised form) June 13, 2016; published electronically September 1, 2016.
http://www.siam.org/journals/siopt/26-3/M102825.html
Funding: This work was supported by the National Science Foundation under grants 1016204, 1115568, 1522629, and 1522654, by the Office of Naval Research under grants N00014-11-1-0068 and N00014-15-1-2048, by the Air Force under contract FA8651-08-D-0108, and by the Defense Advanced Research Projects Agency under contract HR0011-12-C-0011. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).
†Department of Mathematics, University of Florida, Gainesville, FL 32611-8105 (hager@ufl.edu, http://people.clas.ufl.edu/hager/).
‡Department of Mathematics, Louisiana State University, Baton Rouge, LA 70803-4918 (hozhang@math.lsu.edu, http://www.math.lsu.edu/~hozhang).

is trivial, and when any column of $\mathbf{A}$ is zero, the corresponding optimal component of $\mathbf{x}$ is easily determined. We emphasize that the particular form of the polyhedron in (1.2) is for expositional convenience. In the code PPROJ that implements the algorithms developed in this paper, the constraint $\mathbf{x} \geq \mathbf{0}$ in (1.2) is replaced by $\mathbf{lo} \leq \mathbf{x} \leq \mathbf{hi}$, where elements of $\mathbf{lo}$ could be $-\infty$ and elements of $\mathbf{hi}$ could be $+\infty$.

By introducing an additional variable $\mathbf{b} \in \mathbb{R}^m$, the projection problem (1.1)–(1.2) is expressed as

$$\min \left\{ \frac{1}{2}\|\mathbf{y}-\mathbf{x}\|^2 : \mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{l} \leq \mathbf{b} \leq \mathbf{u}, \quad \mathbf{x} \geq \mathbf{0}, \quad \mathbf{x} \in \mathbb{R}^n, \quad \mathbf{b} \in \mathbb{R}^m \right\}.$$

Let $\boldsymbol{\lambda} \in \mathbb{R}^m$ denote the Lagrange multiplier for the linear constraint $\mathbf{A}\mathbf{x} = \mathbf{b}$ in the Lagrangian

$$\mathcal{L}(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{b}) = \frac{1}{2}\|\mathbf{y}-\mathbf{x}\|^2 + \boldsymbol{\lambda}^{\mathsf{T}}(\mathbf{b} - \mathbf{A}\mathbf{x}).$$

If the bound constraints $\mathbf{l} \leq \mathbf{b} \leq \mathbf{u}$ and $\mathbf{x} \geq \mathbf{0}$ are treated explicitly, then the dual problem associated with the projection problem (1.1) is

$$\max\{L(\boldsymbol{\lambda}) : \boldsymbol{\lambda} \in \mathbb{R}^m\}, \tag{1.3}$$

where $L(\boldsymbol{\lambda})$ is the dual function defined by

$$L(\boldsymbol{\lambda}) = \min\left\{\mathcal{L}(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{b}) : \mathbf{l} \leq \mathbf{b} \leq \mathbf{u}, \quad \mathbf{x} \geq \mathbf{0}, \quad \mathbf{x} \in \mathbb{R}^n, \quad \mathbf{b} \in \mathbb{R}^m\right\}. \tag{1.4}$$

The bounds $\mathbf{l}$ and $\mathbf{u}$ are assumed finite to simplify the discussion. When components of $\mathbf{l}$ or $\mathbf{u}$ become infinite, the effective domain of the dual function changes from $\mathbb{R}^n$ to a convex subset of $\mathbb{R}^n$. The analysis still goes through but becomes more complex. The code based on this paper allows infinite values for $\mathbf{l}$ or $\mathbf{u}$.

The arguments that achieve the minimum in (1.4) are given by

$$x_j(\boldsymbol{\lambda}) = \max\{y_j + \mathbf{a}_j^{\mathsf{T}}\boldsymbol{\lambda}, 0\} \quad \text{and} \quad b_i(\boldsymbol{\lambda}) = \begin{cases} \{l_i\} & \text{if } \lambda_i > 0, \\ [l_i, u_i] & \text{if } \lambda_i = 0, \\ \{u_i\} & \text{if } \lambda_i < 0, \end{cases} \tag{1.5}$$

$1 \leq j \leq n$, and $1 \leq i \leq m$, where $\mathbf{a}_j$ is the jth column of $\mathbf{A}$. Note that $\mathbf{x}(\boldsymbol{\lambda})$ is single-valued, while $\mathbf{b}(\boldsymbol{\lambda})$ is set-valued in general. Due to the strong convexity of the objective function, there exists a unique solution $\mathbf{x}^*$ of (1.1) and by [39, p. 265],

$$\|\mathbf{x}(\boldsymbol{\lambda}_2) - \mathbf{x}(\boldsymbol{\lambda}_1)\| \leq \|\mathbf{A}^{\mathsf{T}}\|\|\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_1\|. \tag{1.6}$$

Moreover, by the polyhedral structure of the constraint in (1.1), the maximum in (1.3) is equal to the minimum in (1.1), and $\mathbf{x}(\boldsymbol{\lambda}) = \mathbf{x}^*$ whenever $\boldsymbol{\lambda} \in \mathbb{R}^m$ is a maximizer of L. In a dual approach to (1.1), we first compute a maximizer $\boldsymbol{\lambda}^*$ of L and then recover $\mathbf{x}^* = \mathbf{x}(\boldsymbol{\lambda}^*)$.

Dual strategies are used extensively in nonlinear optimization as can be seen in classic textbooks such as [8, 31, 44, 50]. A dual approach can be much more efficient than a primal approach. For example, the dual simplex method is usually faster than primal simplex, dual approaches to the separable quadratic knapsack problems were found to be very efficient in [14, 24], and the dual active set algorithm was found to be very efficient both for quadratic network optimization [32] and for linear programming [21, 22].

A fast algorithm for the polyhedral projection problem has many important applications. For example, if Newton's method is used to find a feasible point for a system

of equations and inequalities, then the constraints are linearized and each iteration amounts to solving a polyhedral projection problem. In the gradient projection algorithm for a polyhedral constrained optimization problem, each iteration involves a gradient step followed by a projection onto the polyhedron. An important application in the signal processing literature is the basis pursuit denoising problem [10], which amounts to solving an optimization problem of the form

$$\min \ \frac{1}{2}\|\mathbf{A}^{\mathsf{T}}\boldsymbol{\lambda} - \mathbf{y}\|^2 + \sigma \sum_{i=1}^{m} |\lambda_i|, \tag{1.7}$$

where $\sigma > 0$. For recent results in this area see [9]. The problem (1.7) is equivalent to the dual of the following polyhedral projection problem:

$$\min \left\{ \frac{1}{2}\|\mathbf{y} - \mathbf{x}\|^2 : \mathbf{A}\mathbf{x} = \mathbf{b}, \quad -\sigma\mathbf{1} \le \mathbf{b} \le \sigma\mathbf{1}, \quad \mathbf{x} \in \mathbb{R}^n, \quad \mathbf{b} \in \mathbb{R}^m \right\}, \tag{1.8}$$

where $\mathbf{1}$ is the vector whose entries are all 1. More precisely, if the dual of the polyhedral projection problem (1.8) is changed from a maximization problem to a minimization problem by reversing the sign of the objective, and if a constant $\|\mathbf{y}\|^2/2$ is added to the objective, then we obtain the optimization problem (1.7). Finally, we note that in the GALAHAD optimization package [29], the quadratic programming solver QPC uses a projection onto the constraint polyhedron as a starting point. The projection is computed by a primal-dual interior point method based on the algorithm in [58] with enhancements described in [15]. A fast algorithm for projecting a point onto a polyhedron would be useful in any of these applications.

Since the Lagrangian $\mathcal{L}$ is separable in $\mathbf{x}$ and $\mathbf{b}$, the dual function is the sum of two terms, a smooth term f and a nonsmooth term ψ:

$$L(\boldsymbol{\lambda}) = f(\boldsymbol{\lambda}) + \psi(\boldsymbol{\lambda}), \tag{1.9}$$

$$f(\boldsymbol{\lambda}) = \sum_{j=1}^{n} f_j(\boldsymbol{\lambda}), \quad f_j(\boldsymbol{\lambda}) = \frac{1}{2}\left(y_j^2 - x_j(\boldsymbol{\lambda})^2\right), \tag{1.10}$$

$$\psi(\boldsymbol{\lambda}) = \sum_{i=1}^{m} \psi_i(\boldsymbol{\lambda}), \quad \psi_i(\boldsymbol{\lambda}) = \min\{l_i\lambda_i,\ u_i\lambda_i\}. \tag{1.11}$$

An easy way to derive (1.10), pointed out by a referee, is to first observe that

$$x_j(\boldsymbol{\lambda})(y_j + \mathbf{a}_j^{\mathsf{T}}\boldsymbol{\lambda}) = x_j(\boldsymbol{\lambda})^2.$$

This holds trivially when $x_j(\boldsymbol{\lambda}) = 0$ and by (1.5) when $x_j(\boldsymbol{\lambda}) > 0$. With this substitution in the Lagrangian, we obtain (1.9)–(1.10).

Based on the formula (1.5) for $x_j(\boldsymbol{\lambda})$, f_j is a piecewise quadratic function:

$$f_j(\boldsymbol{\lambda}) = \begin{cases} \frac{1}{2}\left[y_j^2 - (y_j + \mathbf{a}_j^{\mathsf{T}}\boldsymbol{\lambda})^2\right] & \text{when } y_j + \mathbf{a}_j^{\mathsf{T}}\boldsymbol{\lambda} \ge 0, \\ \frac{1}{2}y_j^2 & \text{otherwise.} \end{cases}$$

Similarly, ψ_i is a piecewise linear function:

$$\psi_i(\boldsymbol{\lambda}) = \begin{cases} l_i\lambda_i & \text{when } \lambda_i \ge 0, \\ u_i\lambda_i & \text{otherwise.} \end{cases}$$

Hence, L is a piecewise quadratic since its domain can be partitioned into a finite number of polyhedra, and on each polyhedron, L is quadratic. The different polyhedra correspond to the intersection of half spaces of the form $\pm(y_j + \mathbf{a}_j^\mathsf{T}\boldsymbol{\lambda}) \le 0$ for each j (associated with f_j) and with half spaces of the form $\pm\lambda_i \le 0$ for each i (associated with ψ_i). Altogether, L could be formed from as many as 2^{m+n} different quadratics with each quadratic defined on a different polyhedron in $\mathbb{R}^m$.

The function f is continuously differentiable, as expected by [12, Thm. 2.1] or [17], with derivative given by

$$\nabla f(\boldsymbol{\lambda}) = -\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}). \tag{1.12}$$

Since $\mathbf{x}(\boldsymbol{\lambda})$ is a Lipschitz continuous function of $\boldsymbol{\lambda}$ by (1.6), ∇f is also Lipschitz continuous. Moreover, by (1.6), we have

$$\|\nabla f(\boldsymbol{\lambda}_2) - \nabla f(\boldsymbol{\lambda}_1)\| \le \|\mathbf{A}\|\|\mathbf{A}^\mathsf{T}\|\|\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_1\| = \|\mathbf{A}\|^2\|\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_1\|. \tag{1.13}$$

The last equality is due to the fact that the matrix norm induced by the Euclidean norm is the largest singular value, and the largest singular values of $\mathbf{A}$ and $\mathbf{A}^\mathsf{T}$ are the same. In contrast to the smoothness of f and ∇f, the derivative of ψ_i is piecewise constant with a discontinuity at the origin.

Sparse reconstruction by separable approximation (SpaRSA), proposed in [57] by Wright, Nowak, and Figueiredo, applies to the maximization of a concave functional like L that can be expressed as the sum of a smooth term and a nonsmooth term. In [34] we establish an $O(1/k)$ convergence rate for SpaRSA, and an R-linear convergence rate when the objective is strongly concave at a unique optimal solution. Typically the solution of the dual (1.3) of the projection problem is not unique. In this paper, we remove the strong concavity assumption and establish a Q-linear convergence property for the SpaRSA reference function values by applying a strong concavity property relative to the solution set. The analysis employs the GLL [30] reference value which is the minimum function value over the M most recent iterates for some fixed integer $M > 0$. This strengthens a previous R-linear convergence result. We note that R-linear convergence is obtained in [46] for a gradient projection method applied to smooth convex minimization problems, while [54] obtains Q-linear convergence for a class of monotone descent methods applied to a smooth convex minimization problem. Although the results in [54] fit the SpaRSA framework, both the smoothness and the convexity properties of the polyhedral dual function L violate the assumptions in [54]. Also, in [49] a Q-linear convergence result is established for an algorithm similar to SpaRSA but with a different line search.

Although SpaRSA has a Q-linear convergence property, we find that it is asymptotically slower than other approaches for solving (1.1). In the dual approach developed in this paper, we combine SpaRSA with another algorithm with asymptotically faster convergence. Our dual active set strategy (DASS) provides the rules that determine whether SpaRSA or the asymptotically preferred algorithm is executed. We show that only the preferred algorithm is executed asymptotically. DASS has the same general structure as the framework we develop in [35] for combining the gradient projection algorithm with the asymptotically preferred conjugate gradient algorithm when solving box constrained optimization problems. However, both the details of DASS and the analysis are quite different from that of the earlier paper due to the huge difference between box constraints and general polyhedral constraints and due to the special form of the objective function in the polyhedral projection problem. A specific implementation is given in which the asymptotically preferred

algorithm is the dual active set algorithm (DASA) [21, 22, 36, 37, 38, 39, 40, 32], either the factorization-based version [21, 22, 32] or the iterative version [40]. When the linear algebra is implemented using the update-downdate techniques developed in [18, 19, 20, 23] and embedded in CHOLMOD [11, 23], we obtain an extremely fast algorithm for solving the polyhedral projection problem (1.1).

The paper is organized as follows: In section 2 we review SpaRSA [57]. In section 3, a Q-linear convergence property is established for the SpaRSA reference function values by exploiting a strong concavity property for piecewise quadratics along the line segment connecting a point to its projection on the dual solution set. In a sense, SpaRSA identifies the active constraints in a finite number of iterations. Section 4 presents the DASS which combines SpaRSA with an asymptotically preferred algorithm (AP) to accelerate convergence. Rules are given for switching between SpaRSA and AP. Sections 5 and 6 give the global and local asymptotic convergence analysis, respectively. In a rather general setting, it is shown in section 6.3 that when the asymptotically preferred algorithm satisfies some basic properties, only the preferred algorithm is executed asymptotically. In section 7 we give an AP based on the DASA. It is observed that DASS with a DASA based AP converges to a solution of the polyhedral projection problem in a finite number of iterations. Finally, section 8 gives numerical results for a test set composed of the polyhedra associated with the Netlib LP test problems. Comparisons are given between DASA, DASS with a DASA based AP, the general purpose open source software IPOPT [53] (version 3.11) in the COIN-OR library, and the commercial package CPLEX version 12.6, which contains a barrier method targeted to a positive definite quadratic program with a diagonal Hessian, such as the polyhedral projection problem (1.1).

Notation. Let $\nabla f(\boldsymbol{\lambda})$ denote the gradient of f, a column vector, evaluated at $\boldsymbol{\lambda}$, and let $\nabla_i f(\boldsymbol{\lambda})$ be the ith component of the gradient. Let $\partial L(\boldsymbol{\lambda})$ denote the subdifferential set at $\boldsymbol{\lambda}$ and let $\partial_i L(\boldsymbol{\lambda})$ be the ith component of the subdifferential. If $\partial_i L(\boldsymbol{\lambda})$ contains a single element, then $\partial_i L(\boldsymbol{\lambda})$ is simply the element of this set. The Euclidean norm of $\boldsymbol{\lambda}$ is denoted $\|\boldsymbol{\lambda}\|$. The sup-norm (maximum absolute component) of $\boldsymbol{\lambda}$ is denoted $\|\boldsymbol{\lambda}\|_\infty$, while

$$\|\partial L(\boldsymbol{\lambda})\|_{\min} := \min\{\|\mathbf{g}\|_\infty : \mathbf{g} \in \partial L(\boldsymbol{\lambda})\}.$$

We let $\mathbf{g}(\boldsymbol{\lambda})$, a row vector, denote the minimum ∞-norm subgradient in $\partial L(\boldsymbol{\lambda})$. The standard signum function is defined by

$$\operatorname{sgn}(x) = \begin{cases} +1 & \text{if } x > 0, \\ 0 & \text{if } x = 0, \\ -1 & \text{if } x < 0. \end{cases}$$

If $\mathbf{x} \in \mathbb{R}^n$, then $\mathbf{x}^+$ is the positive part of $\mathbf{x}$ defined by $x_i^+ = \max\{0, x_i\}$ for each i. L^* denotes the maximum value for L and Λ^* denotes the set of solutions of (1.3). If $\boldsymbol{\lambda}_k$ is an iterate for an algorithm, then λ_{ki} denotes the ith component of $\boldsymbol{\lambda}_k$ and $\boldsymbol{\lambda}_k^*$ is the projection of $\boldsymbol{\lambda}_k$ onto Λ^*. Since Λ^* is a closed convex set, the projection $\boldsymbol{\lambda}_k^*$ exists and is unique [45, p. 69]. The set $\mathcal{S} := \{i \in [1, m] : l_i < u_i\}$ contains the indices associated with strict inequalities; if $i \in \mathcal{S}^c$, where the superscript c denotes set complement, the ith inequality is really an equality and $l_i = u_i$. Let $\mathcal{Z}(\boldsymbol{\lambda})$ denote the set of indices of zero components of $\boldsymbol{\lambda}$ associated with inequality constraints:

$$\mathcal{Z}(\boldsymbol{\lambda}) = \{i \in \mathcal{S} : \lambda_i = 0\}.$$

If $\mathcal{F} \subset \{1, 2, \ldots n\}$, then $\mathbf{A}_\mathcal{F}$ is the submatrix of $\mathbf{A}$ corresponding to those column indices in $\mathcal{F}$, while $\mathbf{x}_\mathcal{F}$ is the subvector of $\mathbf{x}$ corresponding to indices in $\mathcal{F}$.

Algorithm 2.1 Sparse Reconstruction by Separable Approximation (SpaRSA)

Given $\rho > 1, \sigma \in (0,1)$, $[\alpha_{\min}, \alpha_{\max}] \subset (0,\infty)$, and starting guess $\boldsymbol{\lambda}_1$.
Set $k = 1$.
Step 1. Choose $\alpha_0 \in [\alpha_{\min}, \alpha_{\max}]$
Step 2. Set $\alpha = \rho^j \alpha_0$, where $j \geq 0$ is smallest integer such that

$$L(\boldsymbol{\lambda}_{k+1}) \geq L_k^R + \tfrac{\sigma\alpha}{2}\|\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k\|^2, \text{ where}$$

$$\boldsymbol{\lambda}_{k+1} = \arg\max\{\nabla f(\boldsymbol{\lambda}_k)^{\mathsf{T}}\boldsymbol{\lambda} - \tfrac{\alpha}{2}\|\boldsymbol{\lambda} - \boldsymbol{\lambda}_k\|^2 + \psi(\boldsymbol{\lambda}) : \boldsymbol{\lambda} \in \mathbb{R}^n\}.$$

Step 3. If a stopping criterion is satisfied, terminate.
Step 4. Set $k = k + 1$ and go to Step 1.

2. SpaRSA. In this section, we review the SpaRSA algorithm and previous convergence results. Algorithm 2.1 is SpaRSA [57] applied to (1.3), with L decomposed as in (1.9)–(1.11). In [57], the reference value L_k^R is the GLL [30] reference value $L_k^{\min}$ defined by

$$L_k^{\min} = \min\{L(\boldsymbol{\lambda}_{k-j}) : 0 \leq j < \min(k, M)\}. \tag{2.1}$$

In [34] we introduce other ways to choose the reference value which often yield better performance; however, for the analysis in this paper, we assume that $L_k^R = L_k^{\min}$, the GLL reference value (2.1). For the numerical experiments of section 8, the BB formula [3] was used in Step 1 to generate α_0. That is, we first evaluated

$$\phi = \frac{(\nabla f(\boldsymbol{\lambda}_k) - \nabla f(\boldsymbol{\lambda}_{k-1}))^{\mathsf{T}}(\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1})}{\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1}\|^2}$$

and then set $\alpha_0 = \text{mid}\,\{\alpha_{\min}, \phi, \alpha_{\max}\}$, where mid is the median. An algorithm closely related to SpaRSA is the proximal gradient method [4, 5, 13]; the only difference with SpaRSA is in the line search. SpaRSA is usually presented with a nonmonotone line search while the proximal gradient method is usually given with a monotone line search, and the termination conditions inside the line search are slightly different. Nonetheless, the analysis that we give for SpaRSA should also apply to the proximal gradient method.

Note that there is an explicit formula for the $\boldsymbol{\lambda}_{k+1}$ update in Step 2 of SpaRSA since $\nabla f(\boldsymbol{\lambda}) = -\mathbf{A}\mathbf{x}(\boldsymbol{\lambda})$. In particular, the components of $\boldsymbol{\lambda}_{k+1}$ can be expressed as

$$\lambda_{(k+1)i} = \begin{cases} \lambda_{ki}^l & \text{if } \lambda_{ki}^l \geq 0, \\ 0 & \text{if } \lambda_{ki}^l \leq 0 \leq \lambda_{ki}^u, \\ \lambda_{ki}^u & \text{if } \lambda_{ki}^u \leq 0, \end{cases} \tag{2.2}$$

where

$$\boldsymbol{\lambda}_k^{\mathbf{l}} := \boldsymbol{\lambda}_k + (\mathbf{l} - \mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k))/\alpha \quad \leq \quad \boldsymbol{\lambda}_k^{\mathbf{u}} := \boldsymbol{\lambda}_k + (\mathbf{u} - \mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k))/\alpha. \tag{2.3}$$

From the analysis of [57], when the SpaRSA iterates are bounded, we have

$$\lim_{k\to\infty} L(\boldsymbol{\lambda}_k) = L^*, \tag{2.4}$$

where L^* is the maximum in the dual problem (1.3). Moreover, if for some solution $\boldsymbol{\lambda}^*$ of the dual problem (1.3) there is a constant $\mu > 0$ such that

$$L^* - L(\boldsymbol{\lambda}) \geq \mu\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^*\|^2 \tag{2.5}$$

for all $\boldsymbol{\lambda} \in \mathbb{R}^m$, then by Theorem 4.1 in [34], there are constants $\theta \in [0, 1)$ and c such that

$$\mu\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}^*\|^2 \leq L^* - L(\boldsymbol{\lambda}_k) \leq c\theta^k(L^* - L(\boldsymbol{\lambda}_1)).$$

In other words, SpaRSA is R-linearly convergent.

3. Convergence properties for SpaRSA. The dual function L for the polyhedral projection problem usually does not satisfy the strong concavity condition (2.5). Instead the dual function possesses a strong concavity property relative to the projection onto the dual solution set. This property, established by Li in [43, Thm. 2.7] for a general piecewise quadratic, is restated here in the context of the dual function. As noted in the introduction, the dual function is a concave piecewise quadratic, so Li's result is applicable.

THEOREM 3.1. *For any $\boldsymbol{\lambda}_1 \in \mathbb{R}^m$, there exists a constant $\mu > 0$ such that*

$$\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^*\|^2 \leq \frac{L^* - L(\boldsymbol{\lambda})}{\mu} \quad \forall \boldsymbol{\lambda} \in \mathbb{R}^m \text{ with } L(\boldsymbol{\lambda}) \geq L(\boldsymbol{\lambda}_1), \tag{3.1}$$

where $\boldsymbol{\lambda}^*$ is the projection of $\boldsymbol{\lambda}$ onto Λ^*.

Based on this theorem, we have the following corollary.

COROLLARY 3.2. *For any $\boldsymbol{\lambda}_1 \in \mathbb{R}^m$, there exists a constant $\mu > 0$ such that for all $\boldsymbol{\lambda} \in \mathbb{R}^m$ with $L(\boldsymbol{\lambda}) \geq L(\boldsymbol{\lambda}_1)$, we have*

$$L^* - L(\boldsymbol{\lambda}) \leq \frac{\|\partial L(\boldsymbol{\lambda})\|_{\min}^2}{\mu} \tag{3.2}$$

and

$$\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^*\| \leq \frac{\|\partial L(\boldsymbol{\lambda})\|_{\min}}{\mu}, \tag{3.3}$$

where $\boldsymbol{\lambda}^*$ is the projection of $\boldsymbol{\lambda}$ onto Λ^*.

Proof. First, by (3.1) and the concavity of L, we have

$$\mu\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^*\|^2 \leq L^* - L(\boldsymbol{\lambda}) \leq \|\partial L(\boldsymbol{\lambda})\|_{\min}\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^*\|. \tag{3.4}$$

This inequality gives (3.3). The second inequality in (3.4) along with (3.3) yield (3.2). ∎

We use Theorem 3.1 and Corollary 3.2 to establish a Q-linear convergence property for the SpaRSA reference function values. First, we give a global convergence result. From the analysis in [57], the SpaRSA iterates form a maximizing sequence satisfying (2.4) when the iterates are bounded. Since the set of optimal dual solutions could be unbounded, we drop the boundedness condition in the following theorem.

THEOREM 3.3. *If $\boldsymbol{\lambda}_k$ is generated by SpaRSA, then we have*

$$\lim_{k\to\infty} \|\partial L(\boldsymbol{\lambda}_k)\|_{\min} = 0, \quad \lim_{k\to\infty} \|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\| = 0, \quad \text{and } \lim_{k\to\infty} L(\boldsymbol{\lambda}_k) = L^*. \tag{3.5}$$

Proof. By the definition of $\boldsymbol{\lambda}_{k+1}$ in SpaRSA, we know that

$$\mathbf{0} \in \nabla f(\boldsymbol{\lambda}_k) - \alpha(\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k) + \partial\psi(\boldsymbol{\lambda}_{k+1}).$$

This is equivalent to

$$\nabla f(\boldsymbol{\lambda}_{k+1}) - \nabla f(\boldsymbol{\lambda}_k) + \alpha(\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k) \in \nabla f(\boldsymbol{\lambda}_{k+1}) + \partial\psi(\boldsymbol{\lambda}_{k+1}) = \partial L(\boldsymbol{\lambda}_{k+1}).$$

Hence, we have

$$\|\partial L(\boldsymbol{\lambda}_{k+1})\|_{\min} \le \|\nabla f(\boldsymbol{\lambda}_{k+1}) - \nabla f(\boldsymbol{\lambda}_k) + \alpha(\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k)\|. \tag{3.6}$$

By (1.13), ∇f is Lipschitz continuous with Lipschitz constant $\|\mathbf{A}\|^2$. It follows from (3.6) that

$$\|\partial L(\boldsymbol{\lambda}_{k+1})\|_{\min} \le (\|\mathbf{A}\|^2 + \alpha)\|\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k\|. \tag{3.7}$$

By [34, Prop. 2.1] and the Lipschitz continuity of ∇f, the stepsize α in SpaRSA is bounded from above uniformly in k:

$$\alpha \le \beta := \frac{\rho\|\mathbf{A}\|^2}{1-\sigma}. \tag{3.8}$$

Here ρ and σ are parameters appearing in the statement of SpaRSA. Replacing α by its upper bound β in (3.7) yields

$$\|\partial L(\boldsymbol{\lambda}_{k+1})\|_{\min} \le (\|\mathbf{A}\|^2 + \beta)\|\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k\|. \tag{3.9}$$

On the other hand, in [57, eq. (35)] it is shown that

$$\lim_{k\to\infty} \boldsymbol{\lambda}_{\ell(k)} - \boldsymbol{\lambda}_{\ell(k)-1} = \mathbf{0}, \quad \text{where } \ell(k) = \arg\min\{L(\boldsymbol{\lambda}_j) : \max(0, k-M) < j \le k\}.$$

Hence, by (3.9) and (3.2), we have

$$\lim_{k\to\infty} \|\partial L(\boldsymbol{\lambda}_{\ell(k)})\|_{\min} = 0 \quad \text{and} \quad \lim_{k\to\infty} L(\boldsymbol{\lambda}_{\ell(k)}) = L^*.$$

The stepsize criterion in Step 2 of SpaRSA implies that

$$\frac{\sigma\alpha_{\min}}{2}\|\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k\|^2 \le L(\boldsymbol{\lambda}_{k+1}) - L(\boldsymbol{\lambda}_{\ell(k)}) \le L^* - L(\boldsymbol{\lambda}_{\ell(k)}).$$

Since $L(\boldsymbol{\lambda}_{\ell(k)})$ approaches L^*, we deduce that

$$\lim_{k\to\infty} \|\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k\| = 0,$$

which combines with (3.9) to give the first result in (3.5). Since $L(\boldsymbol{\lambda}_k) \ge L(\boldsymbol{\lambda}_1)$ for all $k \ge 1$ in SpaRSA, the remaining convergence results in (3.5) follow from Corollary 3.2 and the convergence of $\|\partial L(\boldsymbol{\lambda}_{k+1})\|_{\min}$ to zero. ∎

The strong concavity property in Theorem 3.1 could be used to establish the R-linear convergence of SpaRSA applied to the dual function L as in [34, Thm. 4.1]; the only change is to replace the unique optimizer in [34, Thm. 4.1] by the projection onto the set of optima. In the following theorem, we establish a Q-linear convergence property for the reference function values L_k^R in SpaRSA. When $M = 1$ in (2.1), this implies Q-linear convergence of the sequence $L(\boldsymbol{\lambda}_k)$.

THEOREM 3.4. *If $\boldsymbol{\lambda}_k$ is a sequence generated by SpaRSA, then there exists $\theta \in [0, 1)$ such that*

$$L^* - L_k^R \le \theta\left(L^* - L_{k-M}^R\right)$$

for all $k > M$.

Proof. If SpaRSA converges in a finite number of iterations, then the theorem holds with $\theta = 0$. Hence, we assume that $\boldsymbol{\lambda}_k \notin \Lambda^*$ for all k. Since $L(\boldsymbol{\lambda}_k) \geq L(\boldsymbol{\lambda}_1)$ for all $k \geq 1$ in SpaRSA, Corollary 3.2 yields

$$L^* - L(\boldsymbol{\lambda}_k) \leq \frac{1}{\mu} \|\partial L(\boldsymbol{\lambda}_k)\|_{\min}^2. \tag{3.10}$$

If $\kappa := \sigma\alpha_{\min}/2$ is a lower bound for the parameter appearing in the SpaRSA stepsize rule and if $j \in [0, M-1]$ is chosen so that $L_k^R = L(\boldsymbol{\lambda}_{k-j})$, then by the stopping condition in Step 2 of SpaRSA, we have

$$L_k^R = L(\boldsymbol{\lambda}_{k-j}) \geq L_{k-j-1}^R + \kappa \|\mathbf{s}_{k-j-1}\|^2, \quad \mathbf{s}_{k-j-1} := \boldsymbol{\lambda}_{k-j} - \boldsymbol{\lambda}_{k-j-1},$$

which implies that

$$L^* - L_k^R \leq L^* - L_{k-j-1}^R - \kappa \|\mathbf{s}_{k-j-1}\|^2. \tag{3.11}$$

By the SpaRSA stepsize rule, $L(\boldsymbol{\lambda}_{k+1}) > L_k^R$ for each k. It follows that $L_{k+1}^R \geq L_k^R$ for each k, that is, the L_k^R are monotone nondecreasing. Hence, by (3.11), we have

$$L^* - L_k^R \leq L^* - L_{k-M}^R - \kappa \|\mathbf{s}_{k-j-1}\|^2 \tag{3.12}$$

since $k - M \leq k - j - 1$.

Let $c > 0$ be chosen small enough that

$$c \left(\frac{(\|\mathbf{A}\|^2 + \beta)^2}{\mu} \right) < 1. \tag{3.13}$$

If $\|\mathbf{s}_{k-j-1}\|^2 \geq c(L^* - L_{k-M}^R)$, then by (3.12),

$$L^* - L_k^R \leq (1 - c\kappa) \left(L^* - L_{k-M}^R \right). \tag{3.14}$$

Conversely, if $\|\mathbf{s}_{k-j-1}\|^2 < c(L^* - L_{k-M}^R)$, then by (3.9) and (3.10), we have

$$\begin{aligned} L^* - L_k^R = L^* - L(\boldsymbol{\lambda}_{k-j}) &\leq \left(\frac{(\|\mathbf{A}\|^2 + \beta)^2}{\mu} \right) \|\mathbf{s}_{k-j-1}\|^2 \\ &\leq c \left(\frac{(\|\mathbf{A}\|^2 + \beta)^2}{\mu} \right) [L^* - L_{k-M}^R]. \end{aligned} \tag{3.15}$$

Let θ be the smaller of the factors in (3.14) and (3.15). Due to (3.13), $\theta < 1$ and the proof is complete. ∎

Remark 3.1. In [49] a Q-linear convergence result is established under an error bound condition for an algorithm similar to SpaRSA but with a different line search. By [51, Thm. 4], the dual function L satisfies the error bound condition of [49]. Further development of error bound conditions is given in [46, 47, 48].

Note that the dual problem (1.3) can have multiple solutions. In contrast, the solution of the primal problem (1.1) is a unique point $\mathbf{x}^*$ due to the strong convexity of the objective function. By the relationship between the primal and dual solutions, we know that $\mathbf{x}^* = \mathbf{x}(\boldsymbol{\lambda})$ for all $\boldsymbol{\lambda} \in \Lambda^*$. Let $\mathcal{S}$ be the set of strict inequalities defined by

$$\mathcal{S} = \{i \in [1, m] : l_i < u_i\}.$$

We partition the indices of $\mathcal{S}$ into three sets:

$$\begin{aligned}\mathcal{S}_0 &= \{i \in \mathcal{S} : l_i < (\mathbf{A}\mathbf{x}^*)_i < u_i\},\\ \mathcal{S}_+ &= \{i \in \mathcal{S} : (\mathbf{A}\mathbf{x}^*)_i = l_i\},\\ \mathcal{S}_- &= \{i \in \mathcal{S} : (\mathbf{A}\mathbf{x}^*)_i = u_i\}.\end{aligned}$$

Since $\mathbf{x}^*$ is feasible for (1.1), each index $i \in \mathcal{S}$ lies in one of these sets. By the first-order optimality conditions for (1.1), each $\boldsymbol{\lambda} \in \Lambda^*$ satisfies

$$\lambda_i = 0 \text{ for } i \in \mathcal{S}_0, \quad \lambda_i \geq 0 \text{ for } i \in \mathcal{S}_+, \quad \lambda_i \leq 0 \text{ for } i \in \mathcal{S}_-. \tag{3.16}$$

We now show that the SpaRSA iterates satisfy the first-order conditions (3.16) when $\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|$ is sufficiently small.

PROPOSITION 3.5. *There exist $\eta > 0$ with the property that $\boldsymbol{\lambda} = \boldsymbol{\lambda}_{k+1}$ satisfies* (3.16) *whenever $\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\| \leq \eta$, where $\boldsymbol{\lambda}_{k+1}$ is generated by SpaRSA and $\boldsymbol{\lambda}_k^*$ is the projection of $\boldsymbol{\lambda}_k$ onto Λ^*.*

Proof. If $i \in \mathcal{S}_0$, then

$$(\mathbf{l} - \mathbf{A}\mathbf{x}^*)_i < 0 < (\mathbf{u} - \mathbf{A}\mathbf{x}^*)_i.$$

Observe that

$$\begin{aligned}[\mathbf{l} - \mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k)]_i &= [\mathbf{l} - \mathbf{A}\mathbf{x}^*]_i + [\mathbf{A}(\mathbf{x}^* - \mathbf{x}(\boldsymbol{\lambda}_k))]_i\\ &= [\mathbf{l} - \mathbf{A}\mathbf{x}^*]_i + [\mathbf{A}(\mathbf{x}(\boldsymbol{\lambda}_k^*) - \mathbf{x}(\boldsymbol{\lambda}_k))]_i\\ &\leq [\mathbf{l} - \mathbf{A}\mathbf{x}^*]_i + \|\mathbf{A}\|^2\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|\\ &\leq [\mathbf{l} - \mathbf{A}\mathbf{x}^*]_i + \eta\|\mathbf{A}\|^2\end{aligned} \tag{3.17}$$

when $\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\| \leq \eta$. The first inequality in (3.17) is due to the Lipschitz continuity (1.6) of $\mathbf{x}$. Moreover, for $i \in \mathcal{S}_0$, we have

$$\lambda_{ki} \leq |\lambda_{ki}| = |\lambda_{ki} - \lambda_{ki}^*| \leq \|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|.$$

We combine these inequalities to obtain, for $i \in \mathcal{S}_0$ and $\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\| \leq \eta$, the relation

$$\lambda_{ki}^l \leq [\mathbf{l} - \mathbf{A}\mathbf{x}^*]_i / \beta + \eta\left(1 + \|\mathbf{A}\|^2/\beta\right),$$

where β is given in (3.8) and $\boldsymbol{\lambda}_k^{\mathbf{l}}$ is defined in (2.3). In a similar fashion, we have

$$\lambda_{ki}^u \geq [\mathbf{u} - \mathbf{A}\mathbf{x}^*]_i / \beta - \eta\left(1 + \|\mathbf{A}\|^2/\beta\right).$$

Choose $\eta > 0$ small enough that the upper bound for λ_{ki}^l is negative and the lower bound for λ_{ki}^u is positive. Hence, by (2.2), $\lambda_{(k+1)i} = 0$ when $\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\| \leq \eta$ and $i \in \mathcal{S}_0$.

Next, suppose that $i \in \mathcal{S}_-$, which implies that $\lambda_{ki}^* \leq 0$ for all k. We need to show that $\lambda_{(k+1)i} \leq 0$. Observe that

$$0 \geq \lambda_{ki}^* = (\lambda_{ki}^* - \lambda_{ki}) + \lambda_{ki} \geq -\|\boldsymbol{\lambda}_k^* - \boldsymbol{\lambda}_k\| + \lambda_{ki},$$

which yields

$$\lambda_{ki} \leq \|\boldsymbol{\lambda}_k^* - \boldsymbol{\lambda}_k\|. \tag{3.18}$$

If $i \in \mathcal{S}_-$, then $u_i = (\mathbf{A}\mathbf{x}^*)_i = [\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k^*)]_i$. This substitution in (3.17) yields

$$[\mathbf{l} - \mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k)]_i \leq (l_i - u_i) + \eta\|\mathbf{A}\|^2 \tag{3.19}$$

when $\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\| \leq \eta$. Hence, by (3.18) and (3.19), it follows that for $i \in \mathcal{S}_-$, we have

$$\lambda_{ki}^l = \lambda_{ki} + [\mathbf{l} - \mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k)]_i / \alpha \leq \eta + (l_i - u_i + \eta\|\mathbf{A}\|^2)/\beta.$$

Since $i \in \mathcal{S}, l_i - u_i < 0$; hence, we choose η small enough to ensure that $\lambda_{ki}^l \leq 0$, which implies that $\lambda_{(k+1)i} \leq 0$ by (2.2). The analysis of $i \in \mathcal{S}_+$ is similar. ∎

4. DASS. The linear convergence of SpaRSA may be acceptable when computing a low accuracy solution of (1.1), but there are more efficient techniques for computing high accuracy solutions with better local convergence rates. In this section, we propose a dual strategy that combines SpaRSA with an asymptotically preferred algorithm denoted as AP. We develop rules for switching between SpaRSA and AP so that SpaRSA will be used far from the solution, while only AP is executed asymptotically. The hope is that SpaRSA will approximately identify proper signs for λ_i^* in accordance with Proposition 3.5, at which point the dual problem becomes essentially an equality constrained problem where the AP has better performance.

We now specify the properties of AP required for the convergence analysis. Let $\mathcal{B}(\boldsymbol{\lambda})$ be the set of indices of components of $\mathbf{x}(\boldsymbol{\lambda})$ at the lower bound, that is,

$$\mathcal{B}(\boldsymbol{\lambda}) = \{j \in [1,n] : x_j(\boldsymbol{\lambda}) = 0\} = \{j \in [1,n] : y_j + \mathbf{a}_j^\mathsf{T}\boldsymbol{\lambda} \leq 0\}.$$

For any set $\mathcal{B} \subset \{1, 2, \ldots, n\}$, we define a relaxed version of the dual function by

$$L(\boldsymbol{\lambda}, \mathcal{B}) = \min\left\{\mathcal{L}(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{b}) : \mathbf{l} \leq \mathbf{b} \leq \mathbf{u}, \quad x_j \geq 0 \ \forall j \in \mathcal{B}\right\}. \tag{4.1}$$

Note that in the relaxed dual function, x_j for $j \in \mathcal{B}^c$ is unconstrained. Hence, we have

$$L(\boldsymbol{\lambda}, \mathcal{B}_1) \leq L(\boldsymbol{\lambda}, \mathcal{B}_2) \leq L(\boldsymbol{\lambda}) \quad \text{whenever } \mathcal{B}_1 \subset \mathcal{B}_2 \tag{4.2}$$

and

$$L(\boldsymbol{\lambda}) = L(\boldsymbol{\lambda}, \{1, 2, \ldots, n\}) = L(\boldsymbol{\lambda}, \mathcal{B}(\boldsymbol{\lambda})). \tag{4.3}$$

The identity (4.3) holds since $x_j(\boldsymbol{\lambda}) > 0$ for $j \in \mathcal{B}(\boldsymbol{\lambda})^c$, and hence, the constraint $x_j \geq 0$ for $j \in \mathcal{B}(\boldsymbol{\lambda})^c$ can be ignored when evaluating $L(\boldsymbol{\lambda})$. When the AP starts at iteration j, it is assumed to generate a sequence of sets $\mathcal{B}_j, \mathcal{B}_{j+1}, \ldots$. We consider two types of APs. For one type of algorithm, which includes any unconstrained optimizer, $\mathcal{B}_k = \mathcal{B}(\boldsymbol{\lambda}_k)$ for each k. In another type of algorithm, such as the DASA, $\mathcal{B}_j = \mathcal{B}(\boldsymbol{\lambda}_j)$ in the initial iteration of the AP, while $\mathcal{B}_k \subset \mathcal{B}_{k-1}$ for all $k > j$. In this case, the set $\mathcal{B}_k$ is not generally equal to $\mathcal{B}(\boldsymbol{\lambda}_k)$. If AP starts execution at iteration j, then it is assumed that the successive iterates in AP starting from $\boldsymbol{\lambda}_j$ possess the following properties:

(AP1) $\mathcal{B}_j := \mathcal{B}(\boldsymbol{\lambda}_j)$. Either $\mathcal{B}_k = \mathcal{B}(\boldsymbol{\lambda}_k)$ for all $k > j$ or $\mathcal{B}_k \subset \mathcal{B}_{k-1}$ for all $k > j$.

(AP2) For all $k \geq j$, $\mathcal{Z}(\boldsymbol{\lambda}_k) \subset \mathcal{Z}(\boldsymbol{\lambda}_{k+1})$ and $\lambda_{ki}\lambda_{k+1,i} \geq 0$ when $i \in \mathcal{S}$. (Recall that $\mathcal{S}$ and $\mathcal{Z}(\boldsymbol{\lambda})$ are the set of indices of strict inequalities and the set of indices of zero components of $\boldsymbol{\lambda}$, respectively.)

(AP3) $L(\boldsymbol{\lambda}_{k+1}, \mathcal{B}_{k+1}) \geq L(\boldsymbol{\lambda}_k, \mathcal{B}_k)$ for all $k \geq j$.

(AP4) If $\mathcal{Z}(\boldsymbol{\lambda}_k) = \mathcal{Z}(\boldsymbol{\lambda}_l)$ for all $k \geq l$, then $\lim_{k\to\infty} \partial_i L(\boldsymbol{\lambda}_k, \mathcal{B}_k) = 0$ for all i such that $\lambda_{ki} \neq 0$ (equivalently, for all $i \in \mathcal{Z}(\boldsymbol{\lambda}_k)^c$).

(AP5) When AP is restarted, set $j = k + 1$ and generate $\boldsymbol{\lambda}_j$ by applying an iteration of SpaRSA to $\boldsymbol{\lambda}_k$.

By (AP2) the zero components of $\boldsymbol{\lambda}_k$ are also zero in $\boldsymbol{\lambda}_{k+1}$. The condition $\lambda_{ki}\lambda_{k+1,i} \geq 0$ when $i \in \mathcal{S}$ implies that a component of $\boldsymbol{\lambda}_k$ associated with a strict inequality cannot switch its sign in $\boldsymbol{\lambda}_{k+1}$. If AP is restarted in (AP5), then since one iteration of SpaRSA starting from $\boldsymbol{\lambda}_{j-1}$ can only increase the objective value, it follows from (4.3) that

$$L(\boldsymbol{\lambda}_{j-1}) \leq L(\boldsymbol{\lambda}_j) = L(\boldsymbol{\lambda}_j, \mathcal{B}(\boldsymbol{\lambda}_j)) = L(\boldsymbol{\lambda}_j, \mathcal{B}_j). \tag{4.4}$$

The first inequality is strict unless $\boldsymbol{\lambda}_{j-1} = \boldsymbol{\lambda}_j$ are optimal in the dual problem.

In (AP3) it is assumed that the relaxed dual function values are monotone nondecreasing. If $\mathcal{B}_k = \mathcal{B}(\boldsymbol{\lambda}_k)$, then $L(\boldsymbol{\lambda}_k, \mathcal{B}_k) = L(\boldsymbol{\lambda}_k)$ and the dual values themselves are nondecreasing. In DASA, on the other hand, where $\mathcal{B}_k \subset \mathcal{B}_{k-1}$, there is only monotonicity of the relaxed dual values as required by (AP3). Nonetheless, by (4.2) and (4.4), we have

$$L(\boldsymbol{\lambda}_j) = L(\boldsymbol{\lambda}_j, \mathcal{B}_j) \leq L(\boldsymbol{\lambda}_k, \mathcal{B}_k) \leq L(\boldsymbol{\lambda}_k) \quad \text{for } k \geq j.$$

Moreover, when the AP is restarted in (AP5), (4.4) yields $L(\boldsymbol{\lambda}_{j-1}) \leq L(\boldsymbol{\lambda}_j)$.

By (AP2) the set $\mathcal{Z}(\boldsymbol{\lambda}_k)$ can only grow in size, so for l sufficiently large, the condition in (AP4) that $\mathcal{Z}(\boldsymbol{\lambda}_k) = \mathcal{Z}(\boldsymbol{\lambda}_l)$ for all $k \geq l$ is satisfied. Hence, by (AP4) the derivative of the relaxed objective with respect to the nonzero components of the dual multiplier must approach 0.

DASS is given in Algorithm 4.1. Step 1 of DASS corresponds to SpaRSA, while Step 2 corresponds to AP. Recall that $\partial_i L(\boldsymbol{\lambda}_k)$ denotes the ith component of the subdifferential at $\boldsymbol{\lambda}_k$. In DASS this subdifferential is evaluated at an i where $\lambda_{ki} \neq 0$, in which case $\partial_i L(\boldsymbol{\lambda}_k)$ is a singleton. The rule for restarting the AP in Step 2 amounts to saying that the current iterate is sufficiently close to a maximizer of $L(\boldsymbol{\lambda}, \mathcal{B}_k)$ subject to $\lambda_i = 0$ for all $i \in \mathcal{Z}(\boldsymbol{\lambda}_k)$. There are many alternatives to SpaRSA for the restart in (AP5). For example, either a Wolfe line search [55, 56] or an exact line search along a minimum norm subgradient could be used; an exact line search is feasible due to the special structure of (1.1). Since this paper focuses on SpaRSA, our analysis uses the SpaRSA gradient-based line search for the restart.

An entity that influences when we switch from SpaRSA to the AP is the unidentified index set $\mathcal{U}$, which appears in Step 1(b) of DASS. Given fixed parameters $\omega \in (0, 1)$ and $\tau > 0$,

$$\mathcal{U}(\boldsymbol{\lambda}) = \{i \in \mathcal{S} : \lambda_i \neq 0 \text{ and } \operatorname{sgn}(\lambda_i)\partial_i L(\boldsymbol{\lambda}) \leq -\tau(\|\mathbf{g}(\boldsymbol{\lambda})\|_\infty)^\omega\}.$$

In the numerical experiments, $\omega = 0.5$ and $\tau = 0.1$. If $i \in \mathcal{U}(\boldsymbol{\lambda}_k)$, then the partial derivative of L with respect to λ_i is relatively large, and movement from $\boldsymbol{\lambda}_k$ along a direction of ascent pushes the ith component of the current iterate toward zero. In

Algorithm 4.1 Dual Active Set Strategy (DASS)

Given γ and $\xi \in (0,1), \epsilon \geq 0$

Step 1 While $\|\partial L(\boldsymbol{\lambda}_k)\|_{\min} \geq \epsilon$
Execute SpaRSA and in each iteration, check the following:

(a) If $|\partial_i L(\boldsymbol{\lambda}_k)| \geq \gamma \|\partial L(\boldsymbol{\lambda}_k)\|_{\min}$ for some $i \in \mathcal{Z}(\boldsymbol{\lambda}_k)^c$, then go to Step 2.

(b) Otherwise, if $\mathcal{U}(\boldsymbol{\lambda}_k) = \emptyset$, then $\gamma \leftarrow \xi\gamma$.

End

Step 2 While $\|\partial L(\boldsymbol{\lambda}_k)\|_{\min} \geq \epsilon$
Execute AP and in each iteration, check the following:

If $|\partial_i L(\boldsymbol{\lambda}_k, \mathcal{B}_k)| < \gamma \|\partial L(\boldsymbol{\lambda}_k)\|_{\min}$ for all $i \in \mathcal{Z}(\boldsymbol{\lambda}_k)^c$, then

(a) go to Step 1 if $|\partial_i L(\boldsymbol{\lambda}_k)| < \gamma \|\partial L(\boldsymbol{\lambda}_k)\|_{\min}$ for all $i \in \mathcal{Z}(\boldsymbol{\lambda}_k)^c$,

(b) otherwise restart AP using (AP5).

End

End

this case, we hope that SpaRSA will either lock $\lambda_i = 0$ or push it to the opposite side of the origin. But if $\mathcal{U}(\boldsymbol{\lambda})$ is empty in Step 1(b), then we think that the sign of λ_i matches that of λ_i^* so we encourage departure from SpaRSA by making the parameter γ smaller, that is, we multiply γ by the parameter $\xi \in (0,1)$. In the experiments, γ starts at 0.1 and $\xi = 0.5$. In Step 1(a), we would like to leave SpaRSA when $|\partial_i L(\boldsymbol{\lambda}_k)|$ for some $\lambda_{ki} \neq 0$ is relatively large when compared to the minimum norm of the subdifferential.

In Step 2, AP continues to execute until the partial derivatives $\partial_i L(\boldsymbol{\lambda}_k, \mathcal{B}_k)$ of the relaxed dual with respect to the nonzero components of $\boldsymbol{\lambda}_k$ become small in magnitude when compared to $\|\partial L(\boldsymbol{\lambda}_k)\|_{\min}$. In this case, we branch to SpaRSA if the partial derivatives $\partial_i L(\boldsymbol{\lambda}_k)$ of the actual dual are sufficiently small. Otherwise, we restart AP using an iteration of SpaRSA as in (AP5). This gives us the opportunity to unfix the components of $\boldsymbol{\lambda}_k$ which have been kept fixed at zero due to (AP2) and to reset $\mathcal{B}_j = \mathcal{B}(\boldsymbol{\lambda}_j)$ if we were using DASA. If we always choose $\mathcal{B}_k = \mathcal{B}(\boldsymbol{\lambda}_k)$ in (AP1) and if the Step 2 condition

$$|\partial_i L(\boldsymbol{\lambda}_k, \mathcal{B}_k)| < \gamma \|\partial L(\boldsymbol{\lambda}_k)\|_{\min} \ \forall i \in \mathcal{Z}(\boldsymbol{\lambda}_k)^c$$

is satisfied, then the condition in Step 2(a) is also satisfied since

$$L(\boldsymbol{\lambda}_k, \mathcal{B}_k) = L(\boldsymbol{\lambda}_k, \mathcal{B}(\boldsymbol{\lambda}_k)) = L(\boldsymbol{\lambda}_k).$$

Hence, DASS returns to Step 1.

5. Global convergence of DASS. We first provide a global convergence result for the DASS.

PROPOSITION 5.1. *If $\epsilon = 0$ and $\boldsymbol{\lambda}_k$ is a sequence generated by* DASS*, then we have*

$$\liminf_{k\to\infty} \|\partial L(\boldsymbol{\lambda}_k)\|_{\min} = 0, \quad \lim_{k\to\infty} \|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\| = 0, \quad \lim_{k\to\infty} L(\boldsymbol{\lambda}_k) = L^*, \tag{5.1}$$

and $\lim_{k\to\infty} \mathbf{x}(\boldsymbol{\lambda}_k) = \mathbf{x}^*$.

Proof. If only SpaRSA is performed for sufficiently large k, then (5.1) follows from Theorem 3.3 (with lim inf strengthened to lim). If only AP without restarts is performed for k sufficiently large, then by (AP2), $\mathcal{Z}(\boldsymbol{\lambda}_k) \subset \mathcal{Z}(\boldsymbol{\lambda}_{k+1})$ for k sufficiently large, and the sets $\mathcal{Z}(\boldsymbol{\lambda}_k)$ approach a limit as k tends to infinity. By (AP4)

$$\lim_{k\to\infty} \partial_i L(\boldsymbol{\lambda}_k, \mathcal{B}_k) = 0 \quad \forall i \in \mathcal{Z}(\boldsymbol{\lambda}_k)^c. \tag{5.2}$$

Since only AP without restarts is performed for k sufficiently large, it follows from Step 2 of DASS that for k sufficiently large,

$$|\partial_i L(\boldsymbol{\lambda}_k, \mathcal{B}_k)| \geq \gamma \|\partial L(\boldsymbol{\lambda}_k)\|_{\min} \quad \text{for some } i \in \mathcal{Z}(\boldsymbol{\lambda}_k)^c.$$

Utilizing (5.2), we deduce that $\|\partial L(\boldsymbol{\lambda}_k)\|_{\min}$ tends to zero. This gives the first equality in (5.1) with lim inf strengthened to lim. The remaining results in (5.1) follow from Corollary 3.2 and the fact that $L(\boldsymbol{\lambda}_k) \geq L(\boldsymbol{\lambda}_1)$ in DASS.

Finally, let us consider the case where AP is restarted an infinite number of times. This restart occurs either when the AP is restarted in Step 2(b) or when DASS branches from Step 1 to Step 2. If SpaRSA starts at iteration j, then for the GLL reference value, $L_k^R \geq L(\boldsymbol{\lambda}_j)$ for all $k \geq j$. Hence, if $\boldsymbol{\lambda}_k$ is the last SpaRSA iterate before a branch to AP, we have

$$L(\boldsymbol{\lambda}_k) \geq L(\boldsymbol{\lambda}_j) + \frac{\sigma\alpha}{2}\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1}\|^2, \tag{5.3}$$

where the stepsize α has the bound β given in (3.8). As explained in the discussion after (AP1)–(AP5), if the AP is restarted at iteration j, then $L(\boldsymbol{\lambda}_k) \geq L(\boldsymbol{\lambda}_j)$ for all $k \geq j$. Likewise, when SpaRSA is restarted at iteration j, $L(\boldsymbol{\lambda}_k) \geq L_k^R \geq L(\boldsymbol{\lambda}_j)$ for all $k \geq j$. Let $k_j, j = 1, 2, \ldots$, denote the terminating SpaRSA iterations, either within SpaRSA or for the (AP5) restart. The inequalities

$$L(\boldsymbol{\lambda}_k) \geq L(\boldsymbol{\lambda}_j), \quad k \geq j, \tag{5.4}$$

where j is the starting iteration either within SpaRSA or within the AP, together with the growth property (5.3) and the fact that L is bounded from above ensure that $\|\boldsymbol{\lambda}_{k_j} - \boldsymbol{\lambda}_{k_j-1}\|$ tends to zero. Moreover, by (3.7),

$$\lim_{j\to\infty} \|\partial L(\boldsymbol{\lambda}_{k_j})\|_{\min} = 0.$$

This establishes the first relation in (5.1). By (3.2) and the monotonicity property (5.4), $L(\boldsymbol{\lambda}_k)$ approaches L^*. Theorem 3.1 implies that $\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|$ tends to zero, and by the Lipschitz continuity of $\mathbf{x}(\cdot)$ in (1.6), $\mathbf{x}(\boldsymbol{\lambda}_k)$ approaches $\mathbf{x}^*$. This completes the proof. ∎

6. Asymptotic convergence. Next, we analyze the asymptotic convergence properties of iterates produced by DASS. It will be shown that asymptotically only AP is executed. Sections 6.1 and 6.2 provide some preliminary results needed in the analysis.

6.1. Properties of polyhedral projections. Given matrices $\mathbf{M}_1 \in \mathbb{R}^{m\times n_1}$ and $\mathbf{M}_2 \in \mathbb{R}^{m\times n_2}$ and a vector $\mathbf{z} \in \mathbb{R}^m$, define the set

$$\Gamma = \{(\mathbf{v}_1, \mathbf{v}_2) : \mathbf{v}_1 \in \mathbb{R}^{n_1}, \quad \mathbf{v}_2 \in \mathbb{R}^{n_2}, \quad \mathbf{M}_1\mathbf{v}_1 + \mathbf{M}_2\mathbf{v}_2 \leq \mathbf{z}\}.$$

Given $(\mathbf{u}_1, \mathbf{u}_2) \in \Gamma$, let $\Gamma_1 \subset \Gamma$ be defined by

$$\Gamma_1 = \{(\mathbf{v}_1, \mathbf{u}_2) : (\mathbf{v}_1, \mathbf{u}_2) \in \Gamma\}.$$

In the set Γ_1, the second component of the elements are $\mathbf{u}_2$. Let $\mathbf{p}$ and $\mathbf{p}_1$ denote the Euclidean projections given by

$$\mathbf{p}(\mathbf{v}) = \arg\min\{\|\mathbf{v} - \mathbf{w}\| : \mathbf{w} \in \Gamma\} \quad \text{and} \quad \mathbf{p}_1(\mathbf{v}) = \arg\min\{\|\mathbf{v} - \mathbf{w}\| : \mathbf{w} \in \Gamma_1\}.$$

LEMMA 6.1. *There exists a constant $c \geq 0$ independent of $(\mathbf{u}_1, \mathbf{u}_2) \in \Gamma$ such that for all $\mathbf{v} = (\mathbf{v}_1, \mathbf{u}_2) \in \mathbb{R}^{n_1+n_2}$, we have*

$$\|\mathbf{v} - \mathbf{p}(\mathbf{v})\| \leq \|\mathbf{v} - \mathbf{p}_1(\mathbf{v})\| \leq c\|\mathbf{v} - \mathbf{p}(\mathbf{v})\|. \tag{6.1}$$

Proof. Since $\Gamma_1 \subset \Gamma$, the lower bound $\|\mathbf{v} - \mathbf{p}(\mathbf{v})\| \leq \|\mathbf{v} - \mathbf{p}_1(\mathbf{v})\|$ is immediate. Given $\mathbf{v} = (\mathbf{v}_1, \mathbf{u}_2) \in \mathbb{R}^{n_1+n_2}$, the second component of both $\mathbf{v}$ and $\mathbf{p}_1(\mathbf{v})$ is $\mathbf{u}_2$. Consequently, by Hoffman's result [41], there exists a constant c_1, depending on $\mathbf{M}_1$, such that

$$\|\mathbf{v} - \mathbf{p}_1(\mathbf{v})\| \leq c_1\|(\mathbf{M}\mathbf{v} - \mathbf{z})^+\|, \tag{6.2}$$

where $\mathbf{M} = [\mathbf{M}_1|\mathbf{M}_2]$ is the m by n_1+n_2 matrix obtained by appending $\mathbf{M}_2$ after $\mathbf{M}_1$, and the $+$ superscript denotes the positive part of the vector. Since $\overline{\mathbf{v}} := \mathbf{p}(\mathbf{v}) \in \Gamma$, $\mathbf{M}\overline{\mathbf{v}} - \mathbf{z} \leq \mathbf{0}$. Hence, we have

$$\begin{aligned} \|(\mathbf{M}\mathbf{v} - \mathbf{z})^+\| &= \|[\mathbf{M}(\mathbf{v} - \overline{\mathbf{v}}) + \mathbf{M}\overline{\mathbf{v}} - \mathbf{z}]^+\| \\ &\leq \|[\mathbf{M}(\mathbf{v} - \overline{\mathbf{v}})]^+\| \leq \|\mathbf{M}(\mathbf{v} - \overline{\mathbf{v}})\| \leq \|\mathbf{M}\|\|\mathbf{v} - \overline{\mathbf{v}}\|. \end{aligned} \tag{6.3}$$

We combine (6.2) and (6.3) to obtain the second inequality in (6.1) with $c = c_1\|\mathbf{M}\|$. ∎

If the columns of $\mathbf{M}_1$ and $\mathbf{M}_2$ are chosen from a fixed, finite set, then there is a different c for each choice of $\mathbf{M}_1$; but since there are a finite number of different choices for $\mathbf{M}_1$, a fixed constant c can be chosen which is valid for all the potential choices of $\mathbf{M}_1$. We now give an application of Lemma 6.1 to dual sequences converging to Λ^*. Recall that $\mathcal{Z}(\boldsymbol{\lambda}) = \{i \in \mathcal{S} : \lambda_i = 0\}$.

COROLLARY 6.2. *If $\widehat{\boldsymbol{\lambda}}_k$ is the projection defined by*

$$\widehat{\boldsymbol{\lambda}}_k = \textit{arg min}\,\{\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}\| : \boldsymbol{\lambda} \in \Lambda^*, \quad \lambda_i = 0 \textit{ if } i \in \mathcal{Z}(\boldsymbol{\lambda}_k)\}, \tag{6.4}$$

then for k sufficiently large, the feasible set in (6.4) *is nonempty, $\widehat{\boldsymbol{\lambda}}_k$ exists, and*

$$\|\boldsymbol{\lambda}_k - \widehat{\boldsymbol{\lambda}}_k\| \leq c\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|, \tag{6.5}$$

where c is independent of k.

Proof. Suppose to the contrary that there exists a subsequence for which the feasible set in (6.4) is empty. Since $\mathcal{Z}(\boldsymbol{\lambda}_k)$ is contained in a finite set, there exists a fixed nonempty set $\mathcal{Z}_0$ such that $\mathcal{Z}(\boldsymbol{\lambda}_k) = \mathcal{Z}_0$ for infinitely many k. Let Λ_0^* be defined by

$$\Lambda_0^* = \{\boldsymbol{\lambda}_{\mathcal{Z}_0} : \boldsymbol{\lambda} \in \Lambda^*\}.$$

Since Λ^* is a polyhedron, so is Λ_0^*, and its distance δ to the origin must be positive since the feasible set in (6.4) was empty for those k's associated with $\mathcal{Z}_0$. Since $0 < \delta \leq \|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|$ for these k's and since $\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|$ approaches 0 by Proposition 5.1,

we have a contradiction. Hence, for k sufficiently large, the feasible set in (6.4) is nonempty and the projection $\widehat{\boldsymbol{\lambda}}_k$ exists.

The set Λ^* is a polyhedron which can be described by a finite number of linear equalities and inequalities. Hence, we write this linear system in the form $\mathbf{M}\boldsymbol{\lambda} \le \mathbf{z}$ required by Lemma 6.1. In this context, $\mathbf{M}_1$ equals the columns of $\mathbf{M}$ associated with indices in $\mathcal{Z}(\boldsymbol{\lambda}_k)^c$, $\mathbf{M}_2$ equals the columns of $\mathbf{M}$ associated with indices in $\mathcal{Z}(\boldsymbol{\lambda}_k)$, and $\mathbf{u}_2 = \mathbf{0}$. With these choices, the bound (6.5) follows from (6.1) and the fact that the columns of $\mathbf{M}_1$ are taken from a finite set. ∎

6.2. Minimum norm subgradient. In analyzing convergence, we need a way to estimate the error at any iterate. For a concave nonsmooth function such as L, $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}^*$ if and only if $\mathbf{0} \in \partial L(\boldsymbol{\lambda})$. Consequently, $\|\partial L(\boldsymbol{\lambda})\|_{\min}$, defined by

$$\|\partial L(\boldsymbol{\lambda})\|_{\min} = \min\{\|\mathbf{g}\|_\infty : \mathbf{g} \in \partial L(\boldsymbol{\lambda})\},$$

is one measure of the error. Due to the structure of L, we have $\partial L(\boldsymbol{\lambda}) = \mathbf{b}(\boldsymbol{\lambda}) - \mathbf{A}\mathbf{x}(\boldsymbol{\lambda})$, where the set-valued map $\mathbf{b}(\boldsymbol{\lambda})$ and the single-valued map $\mathbf{x}(\boldsymbol{\lambda})$ are defined in (1.5). The minimum ∞-norm subgradient can be expressed as

$$g_i(\boldsymbol{\lambda}) = \begin{cases} l_i - (\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}))_i & \text{if } i \in \mathcal{I}_+(\boldsymbol{\lambda}), \\ u_i - (\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}))_i & \text{if } i \in \mathcal{I}_-(\boldsymbol{\lambda}), \\ 0 & \text{otherwise,} \end{cases} \tag{6.6}$$

where

$$\mathcal{I}_+(\boldsymbol{\lambda}) = \{i : \lambda_i > 0 \text{ or } \lambda_i = 0 \text{ and } l_i - (\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}))_i \ge 0\},$$
$$\mathcal{I}_-(\boldsymbol{\lambda}) = \{i : \lambda_i < 0 \text{ or } \lambda_i = 0 \text{ and } u_i - (\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}))_i \le 0\}.$$

LEMMA 6.3. *Suppose that $\boldsymbol{\lambda}_k$ satisfies the first-order conditions* (3.16)*, which are repeated here for convenience:*

$$\lambda_i = 0 \text{ for } i \in \mathcal{S}_0, \quad \lambda_i \ge 0 \text{ for } i \in \mathcal{S}_+, \quad \lambda_i \le 0 \text{ for } i \in \mathcal{S}_-. \tag{6.7}$$

If $\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|$ is small enough to ensure that $l_i < [\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k)]_i < u_i$ for all $i \in \mathcal{S}_0$, $[\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k)]_i < u_i$ for all $i \in \mathcal{S}_+$, and $[\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k)]_i > l_i$ for all $i \in \mathcal{S}_-$, then for all i, we have

$$|g_i(\boldsymbol{\lambda}_k)| \le |\nabla_i f(\boldsymbol{\lambda}_k) - \nabla_i f(\boldsymbol{\lambda}_k^*)| \le \|\mathbf{A}\|^2 \|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|, \tag{6.8}$$

while for all $i \in \mathcal{Z}(\boldsymbol{\lambda}_k)^c$,

$$g_i(\boldsymbol{\lambda}_k) = \nabla_i f(\boldsymbol{\lambda}_k) - \nabla_i f(\boldsymbol{\lambda}_k^*). \tag{6.9}$$

Proof. Let us start with (6.9). If $\lambda_{ki} > 0$ and $u_i > l_i$, then $i \in \mathcal{S}_+$ since $\boldsymbol{\lambda}_k$ satisfies the first-order conditions (6.7). Since $i \in \mathcal{S}_+$, we have $l_i = [\mathbf{A}\mathbf{x}^*]_i = [\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k^*)]_i$. Hence, by (6.6) we conclude that

$$g_i(\boldsymbol{\lambda}_k) = l_i - [\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k)]_i = [\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k^*) - \mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k)]_i, \tag{6.10}$$

which reduces to (6.9) by the formula (1.12) for the gradient. An analogous argument can be used for the case where $\lambda_{ki} < 0$ and $u_i > l_i$. In the case where $l_i = u_i$, $g_i(\boldsymbol{\lambda}_k)$ is given by (6.10), which again reduces to (6.9) by the formula (1.12) for the gradient.

The second inequality in (6.8) is due to the Lipschitz continuity (1.13) of the gradient of f. Now consider the first inequality in (6.8). For $i \in \mathcal{Z}(\boldsymbol{\lambda}_k)^c$, (6.9) implies

(6.8). If $i \in \mathcal{S}_0$, then (6.8) holds trivially since the assumption $l_i < [\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k)]_i < u_i$ implies that $g_i(\boldsymbol{\lambda}_k) = 0$. Since $\boldsymbol{\lambda}_k$ satisfies the first-order conditions (6.7), we have $\mathcal{S}_0 \subset \mathcal{Z}(\boldsymbol{\lambda}_k)$. The only remaining indices to check in (6.8) are those $i \in \mathcal{Z}(\boldsymbol{\lambda}_k) \setminus \mathcal{S}_0$. These indices correspond to $i \in \mathcal{S}_+$ or $i \in \mathcal{S}_-$ with $\lambda_{ki} = 0$. Suppose that $i \in \mathcal{S}_+$ and $\lambda_{ki} = 0$. By the assumptions of the lemma, $[\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k)]_i < u_i$ for all $i \in \mathcal{S}_+$. It follows from (6.6) that either $g_i(\boldsymbol{\lambda}_k) = 0$, in which case (6.8) holds trivially, or $g_i(\boldsymbol{\lambda}_k) = l_i - [\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k)]_i$. But again for $i \in \mathcal{S}_+$, $l_i = (\mathbf{A}\mathbf{x}^*)_i = [\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k^*)]_i$ and

$$g_i(\boldsymbol{\lambda}_k) = l_i - [\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k)]_i = [\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k^*)]_i - [\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k)]_i = \nabla_i f(\boldsymbol{\lambda}_k) - \nabla_i f(\boldsymbol{\lambda}_k^*).$$

Hence, the first inequality in (6.8) is an equality. The case $i \in \mathcal{S}_-$ with $\lambda_{ki} = 0$ is treated in a similar fashion. ∎

The next result provides a lower bound for the elements of $\partial L(\boldsymbol{\lambda}_k)$ associated with $\mathcal{Z}(\boldsymbol{\lambda}_k)^c$.

LEMMA 6.4. *For k sufficiently large, we have*

$$\mu\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\| \le c\|\widehat{\mathbf{g}}_k\|, \tag{6.11}$$

where c is the constant given in Corollary 6.2*, μ is the constant given in Corollary* 3.2*, and $\widehat{\mathbf{g}}_k$ is the subvector of $\partial L(\boldsymbol{\lambda}_k)$ associated with indices in $\mathcal{Z}(\boldsymbol{\lambda}_k)^c$.*

Proof. By Theorem 3.1, there exists $\mu > 0$ such that

$$\mu\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|^2 \le L^* - L(\boldsymbol{\lambda}_k) \tag{6.12}$$

for k sufficiently large. Again, let $\widehat{\boldsymbol{\lambda}}_k$ denote the projection defined in (6.4). Since $L^* = L(\widehat{\boldsymbol{\lambda}}_k)$ and L is concave, (6.12) yields

$$\mu\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|^2 \le L(\widehat{\boldsymbol{\lambda}}_k) - L(\boldsymbol{\lambda}_k) \le \mathbf{g}(\boldsymbol{\lambda}_k)(\widehat{\boldsymbol{\lambda}}_k - \boldsymbol{\lambda}_k).$$

By the definition of the projection (6.4), $\widehat{\lambda}_{ki} = \lambda_{ki} = 0$ for all $i \in \mathcal{Z}(\boldsymbol{\lambda}_k)$. Hence, we have

$$\mu\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|^2 \le \sum_{i \in \mathcal{Z}(\boldsymbol{\lambda}_k)^c} \partial_i L(\boldsymbol{\lambda}_k)(\widehat{\lambda}_{ki} - \lambda_{ki}). \tag{6.13}$$

By the Schwarz inequality,

$$\mu\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|^2 \le \|\widehat{\mathbf{g}}_k\|\|\widehat{\boldsymbol{\lambda}}_k - \boldsymbol{\lambda}_k\| \le c\|\widehat{\mathbf{g}}_k\|\|\boldsymbol{\lambda}_k^* - \boldsymbol{\lambda}_k\|,$$

where c is the constant given in Corollary 6.2 and $\widehat{\mathbf{g}}_k$ is the subvector of $\partial L(\boldsymbol{\lambda}_k)$ associated with indices in $\mathcal{Z}(\boldsymbol{\lambda}_k)^c$. If $\boldsymbol{\lambda}_k = \boldsymbol{\lambda}_k^*$, then (6.11) holds trivially. If $\boldsymbol{\lambda}_k \neq \boldsymbol{\lambda}_k^*$, then we can divide by $\|\boldsymbol{\lambda}_k^* - \boldsymbol{\lambda}_k\|$ to complete the proof. ∎

We will now prove that asymptotically, DASS only executes AP. The analysis is divided into two cases, the nondegenerate case and the general case. By a nondegenerate problem, we mean that there exists a parameter $\pi > 0$ such that for all $\boldsymbol{\lambda} \in \Lambda^*$ and $i \in \mathcal{S}_+$, we have $\lambda_i \ge \pi$, while for all $\boldsymbol{\lambda} \in \Lambda^*$ and $i \in \mathcal{S}_-$, $\lambda_i \le -\pi$.

6.3. Asymptotic behavior of DASS. We first show in the nondegenerate case that DASS asymptotically executes only the preferred algorithm.

THEOREM 6.5. *Suppose the projection problem* (1.1) *is nondegenerate and* $\boldsymbol{\lambda}_k$ *is generated by* DASS *with* $\epsilon = 0$. *Then after a finite number of iterations, either* $\boldsymbol{\lambda}_k \in \Lambda^*$, *or* DASS *performs only* AP. *Moreover, if* $\mathcal{B}_k = \mathcal{B}(\boldsymbol{\lambda}_k)$ *in each iteration of* AP, *then after a finite number of iterations,* DASS *performs only* AP *without restarts.*

Proof. Suppose that DASS performs an infinite number of iterations; otherwise there is nothing to prove. By Proposition 5.1, $\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|$ approaches 0 and $\mathbf{x}(\boldsymbol{\lambda}_k)$ approaches $\mathbf{x}^*$. Let K be chosen large enough that for all $k \geq K$, we have $l_i < (\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k))_i < u_i$ for every $i \in \mathcal{S}_0$, $\lambda_{ki} \geq \pi/2$ for every $i \in \mathcal{S}_+$, and $\lambda_{ki} \leq -\pi/2$ for every all $i \in \mathcal{S}_-$. By (6.6), $g_i(\boldsymbol{\lambda}_k) = 0$ for all $i \in \mathcal{S}_0$ since $l_i < (\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}_k))_i < u_i$. Since $\lambda_{ki} \geq \pi/2$ and $\lambda_{kj} \leq -\pi/2$ for every $i \in \mathcal{S}_+, j \in \mathcal{S}_-$, and $k \geq K$, it follows that $\mathcal{Z}(\boldsymbol{\lambda}_k) \subset \mathcal{S}_0$. Hence, we have

$$\begin{aligned} \max\{|\partial_i L(\boldsymbol{\lambda}_k)| : i \in \mathcal{Z}(\boldsymbol{\lambda}_k)^c\} &\geq \max\{|\partial_i L(\boldsymbol{\lambda}_k)| : i \in \mathcal{S}_0^c\} \\ &= \|\mathbf{g}(\boldsymbol{\lambda}_k)\|_\infty = \|\partial L(\boldsymbol{\lambda}_k)\|_{\min}. \end{aligned} \tag{6.14}$$

Since $\gamma < 1$, the condition in Step 1(a) is fulfilled and SpaRSA branches to Step 2. Likewise, in AP, the condition of Step 2(a) is never satisfied due to (6.14) and the fact that $\gamma < 1$. Hence, only AP will be executed. If $\mathcal{B}_k = \mathcal{B}(\boldsymbol{\lambda}_k)$ in AP, then $\partial_i L(\boldsymbol{\lambda}_k, \mathcal{B}_k) = \partial_i L(\boldsymbol{\lambda}_k)$ for $i \in \mathcal{Z}(\boldsymbol{\lambda}_k)^c$, and we do not check the conditions in Step 2(a) and Step 2(b). In particular, AP does not perform a restart. ∎

We now consider the general problem which is potentially degenerate. In this case, the undecided index set plays a role.

THEOREM 6.6. *If* $\boldsymbol{\lambda}_k$ *is generated by* DASS, *then after a finite number of iterations, either* $\boldsymbol{\lambda}_k \in \Lambda^*$ *or the* DASS *performs only* AP. *Moreover, if* $\mathcal{B}_k = \mathcal{B}(\boldsymbol{\lambda}_k)$ *in each iteration of* AP, *then after a finite number of iterations,* DASS *performs only* AP *without restarts.*

Proof. Let us focus on the nontrivial case where $\boldsymbol{\lambda}_k \notin \Lambda^*$ for all k. We will now utilize Lemma 6.3. By Proposition 5.1, $\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|$ approaches 0. By the structure of DASS, AP is always preceded by one or more iterations of SpaRSA, and by Proposition 3.5, when $\boldsymbol{\lambda}_k$ is generated by SpaRSA, the first-order conditions (6.7) are satisfied for k sufficiently large. Moreover, by (AP2), the components of $\boldsymbol{\lambda}_k$ do not change sign during execution of AP. Hence, $\boldsymbol{\lambda}_k$ satisfies the first-order conditions (6.7) throughout Steps 1 and 2 of DASS for k sufficiently large. For any i, it follows from the bound (6.8) of Lemma 6.3 that

$$\frac{|\mathrm{sgn}(\lambda_{ki}) g_i(\boldsymbol{\lambda}_k)|}{(\|\mathbf{g}(\boldsymbol{\lambda}_k)\|_\infty)^\omega} \leq \frac{\|\mathbf{g}(\boldsymbol{\lambda}_k)\|_\infty}{(\|\mathbf{g}(\boldsymbol{\lambda}_k)\|_\infty)^\omega} = (\|\mathbf{g}(\boldsymbol{\lambda}_k)\|_\infty)^{1-\omega} \leq \left(\|\mathbf{A}\|^2 \|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|\right)^{1-\omega}.$$

Since $\omega \in (0,1)$ and $\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|$ approaches 0, the right side approaches 0 as k tends to ∞. Consequently, $\mathcal{U}(\boldsymbol{\lambda}_k)$ is empty for k sufficiently large, and if SpaRSA is executed without branching to Step 2, then γ is successively multiplied by the factor $\xi \in (0,1)$, which drives γ toward 0.

By (6.8), we have

$$\|\partial L(\boldsymbol{\lambda}_k)\|_{\min} = \|\mathbf{g}(\boldsymbol{\lambda}_k)\|_\infty \leq \|\mathbf{A}\|^2 \|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|. \tag{6.15}$$

And by Lemma 6.4,

$$\frac{\mu}{c}\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\| \leq \|\widehat{\mathbf{g}}_k\| \leq \sqrt{m}\|\widehat{\mathbf{g}}_k\|_\infty, \tag{6.16}$$

Algorithm 7.1 The Dual Active Set Algorithm (DASA) for (1.3)

Given starting guess $\boldsymbol{\lambda}_0$, initialize $k = 0$

While $\mathbf{0} \notin \partial L(\boldsymbol{\lambda}_k)$:

Step 1. $\mathcal{I}_k^+ = \mathcal{I}_+(\boldsymbol{\lambda}_k) \cup \mathcal{S}^c$, $\mathcal{I}_k^- = \mathcal{I}_-(\boldsymbol{\lambda}_k) \cup \mathcal{S}^c$, $\mathcal{B}_k = \mathcal{B}(\boldsymbol{\lambda}_k)$,
$\mathcal{I}_k^0 = \mathcal{S} \setminus (\mathcal{I}_k^+ \cup \mathcal{I}_k^-)$.

Step 2. $\boldsymbol{\mu} \in \arg\max \left\{ L_k(\boldsymbol{\lambda}) : \boldsymbol{\lambda} \in \mathbb{R}^m, \boldsymbol{\lambda}_{\mathcal{I}_k^0} = \mathbf{0} \right\}$, $\mathbf{d} = \boldsymbol{\mu} - \boldsymbol{\lambda}_k$.

Step 3. $s_k \in \arg\max_{s \geq 0} L(T_k(\boldsymbol{\lambda}_k + s\mathbf{d}), \mathcal{B}_k)$, $\boldsymbol{\lambda}_{k+1} = T_k(\boldsymbol{\lambda}_k + s_k \mathbf{d})$.

Step 4. $k = k + 1$, $\mathcal{B}_k = \mathcal{B}_{k-1} \setminus \{j : x_j(\boldsymbol{\lambda}_k) > 0\}$,
$\mathcal{I}_k^0 = \mathcal{I}_{k-1}^0 \cup \{i \in \mathcal{S} : \boldsymbol{\lambda}_{ki} = 0 \text{ and } \boldsymbol{\lambda}_{k-1,i} \neq 0\}$,
$\mathcal{I}_k^+ = \mathcal{I}_{k-1}^+ \setminus \mathcal{I}_k^0$, $\mathcal{I}_k^- = \mathcal{I}_{k-1}^- \setminus \mathcal{I}_k^0$.

Step 5. If $\mathcal{B}_k \neq \mathcal{B}_{k-1}$ or $\mathcal{I}_k^+ \neq \mathcal{I}_{k-1}^+$ or $\mathcal{I}_k^- \neq \mathcal{I}_{k-1}^-$, go to Step 2.

End

where $\widehat{\mathbf{g}}_k$ is the subvector of $\partial L(\boldsymbol{\lambda}_k)$ associated with indices in $\mathcal{Z}(\boldsymbol{\lambda}_k)^c$. Combining (6.15) and (6.16) yields

$$\frac{\|\widehat{\mathbf{g}}_k\|_\infty}{\|\partial L(\boldsymbol{\lambda}_k)\|_{\min}} \geq \frac{\mu}{c\sqrt{m}\|\mathbf{A}\|^2}.$$

Hence, the condition in Step 1(a) of DASS is fulfilled whenever $\gamma \leq \mu/(c\sqrt{m}\|\mathbf{A}\|^2)$. If γ is this small, then the condition in Step 2(a) is never satisfied, and only AP of Step 2 is executed for k sufficiently large. If $\mathcal{B}_k = \mathcal{B}(\boldsymbol{\lambda}_k)$ in AP, then $\partial_i L(\boldsymbol{\lambda}_k, \mathcal{B}_k) = \partial_i L(\boldsymbol{\lambda}_k)$ for $i \in \mathcal{Z}(\boldsymbol{\lambda}_k)^c$, and we do not check the conditions in Step 2(a) and Step 2(b). In particular, AP does not perform a restart. ∎

7. Dual active set algorithm. For the numerical experiments, the AP is based on the DASA [21, 22, 36, 37, 38, 39, 40, 32]. Algorithm 7.1 is DASA in the context of the polyhedral projection problem (1.1). Three sets appear in the statement of DASA. The sets $\mathcal{I}_k^+$ and $\mathcal{I}_k^-$ correspond to inequalities that are treated as at their lower and upper bounds, respectively. Their initialization is based on the sets $\mathcal{I}_+(\boldsymbol{\lambda})$ and $\mathcal{I}_-(\boldsymbol{\lambda})$ associated with the minimum ∞-norm subgradient (6.6). The set $\mathcal{B}_k$ corresponds to the components of the primal variable that are treated as at their lower bound. Based on current bound choices, we define the local dual function L_k by

$$L_k(\boldsymbol{\lambda}) = \inf\{L(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{b}) : x_j = 0 \; \forall \; j \in \mathcal{B}_k, \; b_i = l_i \; \forall \; i \in \mathcal{I}_k^+, \; b_i = u_i \; \forall \; i \in \mathcal{I}_k^-\}.$$

When evaluating L_k, the components of $\mathbf{x}$ in $\mathcal{B}_k^c$ and the components of $\mathbf{b}$ in $(\mathcal{I}_k^+ \cup \mathcal{I}_k^-)^c$ are unconstrained. Consequently, $L_k(\boldsymbol{\lambda}) = -\infty$ if $\lambda_i \neq 0$ for some $i \in (\mathcal{I}_k^+ \cup \mathcal{I}_k^-)^c$. The maximization of L_k in Step 2 is equivalent to solving the linear system

$$\mathbf{A}_{\mathcal{R}\mathcal{C}} \mathbf{A}_{\mathcal{R}\mathcal{C}}^\mathsf{T} \boldsymbol{\mu}_\mathcal{R} = \mathbf{b}_\mathcal{R} - \mathbf{A}_{\mathcal{R}\mathcal{C}} \mathbf{y}_\mathcal{C}, \tag{7.1}$$

where $\mathbf{b}$ is the vector given in the definition of L_k, $\mathbf{A}_{\mathcal{R}\mathcal{C}}$ is the submatrix of $\mathbf{A}$ corresponding to rows $i \in \mathcal{R} = \mathcal{I}_k^+ \cup \mathcal{I}_k^-$ and columns $j \in \mathcal{C} = \mathcal{B}_k^c$, and $\mu_i = 0$ when $i \notin \mathcal{R}$. Since the matrix $\mathbf{A}_{\mathcal{R}\mathcal{C}} \mathbf{A}_{\mathcal{R}\mathcal{C}}^\mathsf{T}$ may be singular, we could utilize the minimum norm solution to (7.1). In practice, we modify the dual function by adding a term

of the form $0.5\epsilon\|\boldsymbol{\lambda}\|^2$, where ϵ is on the order of the computing precision. With this adjustment to L, the matrix in (7.1) becomes $\mathbf{A}_{\mathcal{RC}}\mathbf{A}_{\mathcal{RC}}^{\mathsf{T}}+\epsilon\mathbf{I}$, which is positive definite. This modification also ensures that the maximizer of L_k exists.

The truncation operator T_k in Algorithm 7.1 essentially holds a dual multiplier fixed at zero when it reaches zero. More precisely, the components of the truncation operator are defined by

$$T_k(\boldsymbol{\lambda})_i = \begin{cases} \max\{\lambda_i, 0\} & \text{if } i \in \mathcal{I}_k^+ \cap \mathcal{S}, \\ \min\{\lambda_i, 0\} & \text{if } i \in \mathcal{I}_k^- \cap \mathcal{S}, \\ \lambda_i & \text{otherwise.} \end{cases}$$

In DASA, the sets $\mathcal{B}_k$ and $\mathcal{I}_k^{\pm}$ can only shrink in each iteration; eventually these sets do not change, at which point $\boldsymbol{\lambda}_{k+1}$ solves the maximization problem of Step 2 and DASA returns to Step 1 assuming $\mathbf{0} \notin \partial L(\boldsymbol{\lambda}_k)$. By the theory for DASA, we know that it reaches a solution of the dual problem in a finite number of iterations.

Various techniques for efficiently solving (7.1) have been developed in our earlier work [11, 18, 19, 20, 21, 23, 40]. One strategy is the direct factorization approach [11, 18, 19, 20, 23]. When there are small changes in the sets $\mathcal{R}$ or $\mathcal{C}$, then the current factorization can be updated to reflect the changes in the sets. Another approach is to approximately solve the linear system using an iterative method [40]. Our code currently exploits an SSOR preconditioned conjugate gradient iterative solver. The final approach we have developed is what we call a multilevel approach [21]. With this approach we analyze the sparsity structure of the linear system and find blocks that can be uncoupled from the rest of the problem and solved separately. These blocks are arranged in a tree, and we work our way up the tree, eventually solving the original linear system. An advantage of the multilevel approach is that the sets $\mathcal{B}_k$ and $\mathcal{I}_k^{\pm}$ are updated in a dynamic fashion as we work our way up the multilevel tree. The code developed for solving the polyhedral projection problem exploits all of these algorithms. Based on the analysis of the sparsity structure, either a single-level or a multilevel approach is utilized. Based on the estimated flops to Cholesky factor the matrix or to perform an update/downdate, we either factor the matrix from scratch or perform a series of updates and downdates. Finally, based on an estimate for the update/downdate time relative to the time of an iteration, we determine how many SSOR preconditioned conjugate gradient iterations could be beneficial.

One of the requirements for an AP was that $\mathcal{Z}(\boldsymbol{\lambda}_k) \subset \mathcal{Z}(\boldsymbol{\lambda}_{k+1})$ for each k. Note that DASA may violate this property since $\mathcal{I}_k^+$ or $\mathcal{I}_k^-$ in Step 1 of Algorithm 7.1 could contain indices i for which $\lambda_{ki} = 0$, and in the first iteration after Step 1, λ_{ki} could become nonzero. Algorithm 7.2 is an AP based on DASA in which the definition of $\mathcal{I}_0^{\pm}$ is modified to exclude all $i \in \mathcal{S}$ where $\lambda_{ki} = 0$. (AP1) holds since $\mathcal{B}_k$ is obtained by removing elements from $\mathcal{B}_{k-1}$ in Step 3. (AP2) holds since the components of $\boldsymbol{\lambda}_k$ in $\mathcal{S}$ which reach zero are kept fixed at zero. In particular, the set $\mathcal{I}_0^0$ is initialized to be the indices in $\mathcal{S}$ associated with vanishing components of $\boldsymbol{\lambda}_0$. At iteration k, L_k is maximized in Step 1 subject to the constraint $\boldsymbol{\lambda}_{\mathcal{I}_k^0} = \mathbf{0}$. During the line search of Step 2, the components of $\mathbf{d}$ and $\boldsymbol{\lambda}_k$ corresponding to $\mathcal{I}_k^0$ are zero. Moreover, if $i \in \mathcal{S}$ and $\lambda_{ki} > 0$, then the ith component of $T_k(\boldsymbol{\lambda}_k + s_k\mathbf{d})$ is zero whenever $\lambda_{ki} + s_k d_{ki} \le 0$. And in Step 3, these nonzero components that were fixed at zero during the Step 2 line search are inserted in $\mathcal{I}_{k+1}^0$. (AP3) is the key ascent property of DASA; to fully understand this property, one needs to study the convergence analysis in [21, 22, 32]. The rationale for (AP3) is roughly as follows. By the definition of $\mathcal{B}_k$, it follows that $L_k(\boldsymbol{\lambda}_k) = L(\boldsymbol{\lambda}_k, \mathcal{B}_k)$. If $\boldsymbol{\lambda}_k$ is not a maximizer of L_k, then $L_k(\boldsymbol{\mu}) > L_k(\boldsymbol{\lambda}_k) =$

Algorithm 7.2 A DASA based AP

Given starting guess $\boldsymbol{\lambda}_0$, $\mathcal{B}_0 = \mathcal{B}(\boldsymbol{\lambda}_0)$, $\mathcal{I}_0^0 = \mathcal{Z}(\boldsymbol{\lambda}_0)$,
$\mathcal{I}_0^+ = \{i \in \mathcal{S} : \lambda_{0i} > 0\} \cup \mathcal{S}^c$, $\mathcal{I}_0^- = \{i \in \mathcal{S} : \lambda_{0i} < 0\} \cup \mathcal{S}^c$

For $k = 0, 1, 2, \ldots$

Step 1. $\boldsymbol{\mu} \in \arg\max \left\{ L_k(\boldsymbol{\lambda}) : \boldsymbol{\lambda} \in \mathbb{R}^m, \boldsymbol{\lambda}_{\mathcal{I}_k^0} = \mathbf{0} \right\}$, $\mathbf{d} = \boldsymbol{\mu} - \boldsymbol{\lambda}_k$.

Step 2. $s_k \in \arg\max_{s \geq 0} L(T_k(\boldsymbol{\lambda}_k + s\mathbf{d}), \mathcal{B}_k)$, $\boldsymbol{\lambda}_{k+1} = T_k(\boldsymbol{\lambda}_k + s_k\mathbf{d})$.

Step 3. $\mathcal{B}_{k+1} = \mathcal{B}_k \setminus \{j : x_j(\boldsymbol{\lambda}_{k+1}) > 0\}$, $\mathcal{I}_{k+1}^0 = \mathcal{Z}(\boldsymbol{\lambda}_{k+1})$,
$\mathcal{I}_{k+1}^+ = \mathcal{I}_k^+ \setminus \mathcal{I}_{k+1}^0$, $\mathcal{I}_{k+1}^- = \mathcal{I}_k^- \setminus \mathcal{I}_{k+1}^0$.

End

$L(\boldsymbol{\lambda}_k, \mathcal{B}_k)$. In Step 2, we search along the line segment connecting $\boldsymbol{\lambda}_k$ and $\boldsymbol{\mu}$ using the merit function $L(T_k(\cdot), \mathcal{B}_k)$. This takes us to a new point $\boldsymbol{\lambda}_{k+1}$ satisfying

$$L(\boldsymbol{\lambda}_{k+1}, \mathcal{B}_{k+1}) = L(\boldsymbol{\lambda}_{k+1}, \mathcal{B}_k) > L(\boldsymbol{\lambda}_k, \mathcal{B}_k),$$

which ensures that (AP3) holds. (AP4) holds since within a finite number of iterations, $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\mu}$, a maximizer of L_k over the subspace where $\boldsymbol{\lambda}_{\mathcal{I}_k^0} = 0$. The proof of these properties associated with (AP3) and (AP4) is mostly contained in [32, Thm. 1]; the difference between [32] and the current paper is that in the earlier work, the constraint is an equation $\mathbf{h}(\mathbf{x}) = \mathbf{0}$, while here the corresponding constraint is a system of inequalities $\mathbf{l} \leq \mathbf{A}\mathbf{x} \leq \mathbf{u}$. The treatment of inequalities, which leads to the index sets $\mathcal{I}_k$ here, can be found in [22, sect. 5].

If the DASA-based AP given in Algorithm 7.2 is used in DASS and if the test "$|\partial_i L(\boldsymbol{\lambda}_k, \mathcal{B}_k)| < \gamma \|\partial L(\boldsymbol{\lambda}_k)\|_{\min}$ for all $i \in \mathcal{Z}(\boldsymbol{\lambda}_k)^c$" in Step 2 of DASS is replaced by the test "$\partial_i L(\boldsymbol{\lambda}_k, \mathcal{B}_k) = 0$ for all $i \in \mathcal{Z}(\boldsymbol{\lambda}_k)^c$," then DASS converges in a finite number of iterations. The reason is that the SpaRSA restart strictly improves the objective value, based on the SpaRSA line search condition and (3.9), unless optimality has been achieved. Immediately before the restart, DASA had reached a maximizer of L_k. By the monotonicity of DASA and the strict improvement of the dual function during the SpaRSA restart, the pair of sets $\mathcal{B}_k$ and $\mathcal{Z}(\boldsymbol{\lambda}_k)$ associated with the maximizer of L_k cannot repeat.

Algorithms 7.1 and 7.2 both employ a line search in the search direction $\mathbf{d}$. In certain contexts, this line search can be eliminated and we can set $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\mu}$. For example, in the dual approximations to optimal control problems studied in [33], it was found that the version of DASA without a line search was locally, quadratically convergent for a class of optimal control problems. On the other hand, in later work [32] related to network optimization, we found that the step $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\mu}$ often resulted in a nonconvergent algorithm. This led to the line search version of DASA, first presented in [36]. More recently, the step $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\mu}$ is investigated more deeply in the context of optimal control for partial differential equations, and both global and superlinear convergence results are established when the matrix possess certain properties; references include [6, 7, 52]. Very recently, in [16], a framework is developed in which the step $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\mu}$ plays a key role, and in this broader framework, a finite convergence result is established.

8. Numerical results. DASS with DASA for the AP was implemented in a C code called PPROJ. We compared the performance of PPROJ to IPOPT (version 3.11) in the COIN-OR library and to CPLEX (version 12.6). IPOPT is an open source general purpose nonlinear optimization package, initially developed by Wächter and Biegler in [53], based on an interior point method, and CPLEX is a commercial package, initially developed by Robert Bixby, which applies to linear and quadratic programming problems, and which contains algorithms based on the primal simplex method, the dual simplex method, and a barrier method targeted to positive definite quadratic programs with a diagonal Hessian as in the polyhedral projection problem (1.1). Note that IPOPT does not exploit the specific structure of the polyhedral projection problem while CPLEX does. The polyhedra used in the numerical experiments were the constraints in the Netlib LP test set. Altogether, there were 109 polyhedra with m ranging from 2 up to 39,867 and n ranging from 3 up to 224,125.

The CPLEX presolver was applied to each test problem since presolving is often crucial for the performance of barrier or interior point methods. See [1] for presolve techniques. Roughly, the presolver attempts to simplify the constraints without changing the feasible set. The vector $\mathbf{y}$ projected onto each polyhedron was a randomly generated with components uniformly distributed on the interval $[-1, +1]$. These randomly generated points were always outside the polyhedra. The PPROJ code, the test problems, the infeasible points, and the projections can be downloaded from the authors' web pages.

The starting point for IPOPT was $\mathbf{x} = \mathbf{0}$, while the starting point for PPROJ was $\boldsymbol{\lambda} = \mathbf{0}$. CPLEX generates it own starting point. PPROJ solved all the test problems to a relative error tolerance of 10^{-9}, where the relative error is the following ratio of the minimum ∞-norm subgradient to a normalization expression:

$$\frac{\max\left\{\left|l_i - \sum_{j=1}^n a_{ij}x_j(\boldsymbol{\lambda})\right|, \left|u_k - \sum_{j=1}^n a_{kj}x_j(\boldsymbol{\lambda})\right| : i \in \mathcal{I}_+(\boldsymbol{\lambda}), k \in \mathcal{I}_-(\boldsymbol{\lambda})\right\}}{\max\left\{\sum_{j=1}^n |a_{ij}x_j(\boldsymbol{\lambda})|, \sum_{j=1}^n |a_{kj}x_j(\boldsymbol{\lambda})| : i \in \mathcal{I}_+(\boldsymbol{\lambda}), k \in \mathcal{I}_-(\boldsymbol{\lambda})\right\}}.$$

The numerator is $\|\partial L(\boldsymbol{\lambda})\|_{\min}$, where the minimum norm subgradient is given in (6.6). At optimality, the numerator vanishes. The denominator measures the size of the numbers that enter into the computation of the numerator. IPOPT uses a different yet related relative error estimator [53]. The tolerance 10^{-9}, which is slightly smaller than the default tolerance 10^{-8} in IPOPT, was determined as follows: All the test problems were initially solved using PPROJ and a relative error tolerance 10^{-12}. The computed projections, which are unique due to the strong convexity of the objective function, were saved. The problems were then resolved to a lower tolerance, which was increased until the computed projections had at least 4 digit accuracy for all the test problems.

Default parameter values were used for all the software except for the parameter MAX_ITER in IPOPT, which was increased from its default value of 3,000 to 50,000. With the default value, 16 problems terminated in IPOPT with the error message "maximum number of iterations exceeded." After increasing MAX_ITER to 50,000, 7 more problems were solved.

CPLEX is a commercial closed source package which is completely self contained. Both PPROJ and IPOPT require a linear system solver. For IPOPT the solver was MA57 [28] in the Harwell Subroutine Library, a code for solving sparse symmetric indefinite linear systems like those that arise in the interior point method. PPROJ utilizes CHOLMOD [11, 23], a package for solving symmetric positive definite lin-

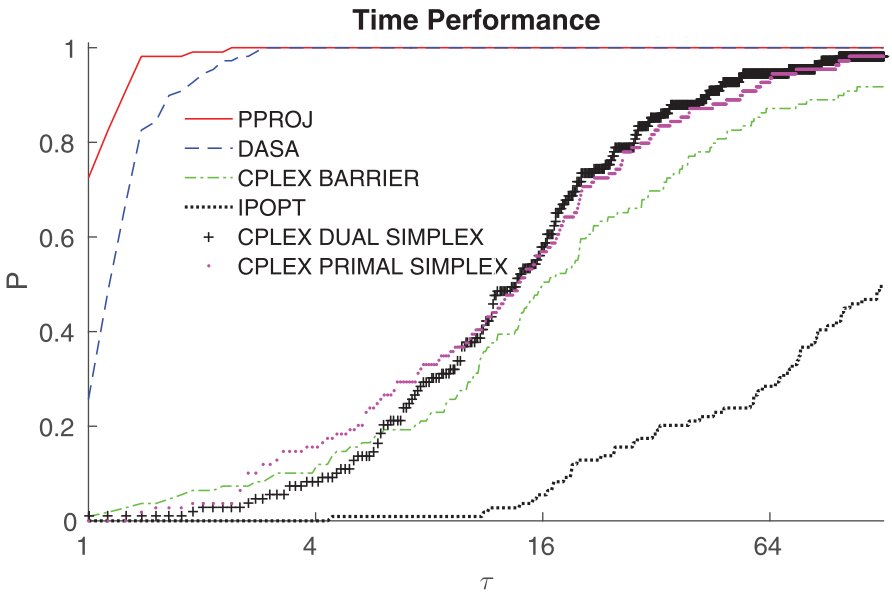

FIG. 1. *Running time performance for the Netlib LP polyhedra.*

ear systems like those arising in DASA. Both CHOLMOD and MA57 require the BLAS [27, 26, 42] and LAPACK [2]. We employed the GOTO2 BLAS of Kazushige Gotō. The computer on which the experiments were performed was a Dell Precision T7610 Workstation with a dual Intel Xeon Processor E5-2687W v2 (16 physical cores, 3.4GHz, 25.6MB cache, 192GB memory). The GOTO2 BLAS were restricted to 12 cores since better performance was achieved.

The operating system was Linux. It was found that the most reliable way to time the software was to dedicate the machine to running the codes and then time the runs using the gettimeofday routine in Linux (measuring the wall time). For IPOPT, the time to evaluate the function, gradient, Jacobian, and Hessian is a negligible part of the total running time, and the solution time is primarily the time for factoring and solving the linear system that arises in each iteration.

A performance profile comparing the running times of the codes is given in Figure 1. The data that is plotted in Figure 1 is posted on the authors' web pages. The vertical axis of the Dolan–Moré performance profile [25] gives the fraction P of problems for which any given method is within a factor τ of the best time. In the CPU time performance profile plot, the top curve is the method that solved the most problems in a time that was within a factor τ of the best time. The percentage of the test problems for which a method is fastest is given on the left axis of the plot. The right side of the plot gives the percentage of the test problems that were successfully solved by each of the methods. In essence, the right side is a measure of an algorithm's robustness. Observe that PPROJ had the best performance for the test set on which Figure 1 is based. Again, we emphasize that IPOPT is a general purpose nonlinear optimization package that does not exploit the structure of the polyhedral projection problem. Hence, Figure 1 indicates the potential benefit of software specifically targeted to the polyhedral projection problem.

To assess the benefit from combining SpaRSA with DASA, we also solved each test problem using only DASA. As seen in Figure 1, for about 75% of the problems,

the combined code was faster than the pure DASA code. Thus there was an overall benefit from combining the two algorithms; nonetheless, in about 1 of 4 problems, pure DASA was faster than PPROJ. This may be related to poor conditioning of some problems and the fact that the DASA search directions are much better than the SpaRSA directions. As a result, SpaRSA might move to a point where DASA takes more iterations than would have been used by DASA from the starting point. The better performance of PPROJ relative to the pure DASA algorithm is achieved with very few SpaRSA iterations. There are always fewer than 10 iterations, and in most cases there are just a few iterations. In contrast, there were many changes in the active constraints during these iterations. These few SpaRSA iterations that occurred at the very start of each PPROJ run generated an approximation to the active constraints for an optimal solution which generally reduced the linear algebra overhead in DASA. Our parameter values were $\gamma = 0.1$ and $\xi = 0.5$.

9. Conclusions. The paper focused on the problem of projecting a point onto a polyhedron. The projection problem has many important applications including Newton methods, gradient projection methods, basis pursuit problems that arise in sparse signal recovery, and the generation of starting points for a quadratic program solver. A new DASS was presented. DASS uses SpaRSA to approximately identify active inequality constraints in the polyhedron and an asymptotically preferred algorithm, denoted AP, to accelerate convergence. It was proved that when SpaRSA is applied to the dual of the projection problem, the dual values possess a Q-linear convergence property even though the dual problem may have multiple solutions and the objective function is not strongly concave. Asymptotically, it was shown that DASS only executes AP, not SpaRSA. When AP is an unconstrained optimization method that keeps dual multipliers fixed at zero whenever they reach zero, it was shown in Theorems 6.5 and 6.6 that asymptotically, no restarts were needed, that is, when the unconstrained optimization algorithm is used in DASS, the positive components of the dual multipliers remain positive and the negative components remain negative asymptotically. An implementation of the DASS called PPROJ was developed in which AP was based on the DASA. The performance of PPROJ was evaluated using polyhedra associated with the constraints in the Netlib LP test set. It was found that PPROJ was robust, accurate, and fast.

Acknowledgments. Constructive comments and suggestions from the reviewers and associate editor are gratefully acknowledged. In particular, the results in [43] for piecewise quadratics greatly simplified the Q-linear convergence analysis in section 3.

REFERENCES

[1] E. D. Andersen and K. D. Andersen, *Presolving in linear programming*, Math. Program., 71 (1995), pp. 221–245.

[2] E. Anderson, Z. Bai, C. H. Bischof, S. Blackford, J. W. Demmel, J. J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. C. Sorensen, *LAPACK Users' Guide*, 3rd ed., SIAM, Philadelphia, 1999.

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

[4] A. Beck and M. Teboulle, *Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems*, IEEE Trans. Image Process., 18 (2009), pp. 2419–2434.

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

[6] M. Bergounioux, K. Ito, and K. Kunisch, *Primal-dual strategy for optimal control problems*, SIAM J. Control Optim., 37 (1999), pp. 1176–1999.
[7] M. Bergounioux and K. Kunisch, *Primal-dual strategy for state-constrained optimal control problem*, Comput. Optim. Appl., 22 (2002), pp. 193–224.
[8] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, MA, 1999.
[9] R. H. Byrd, J. Nocedal, and S. Solntsev, *An Algorithm for Quadratic ℓ_1-Regularized Optimization with a Flexible Active-Set Strategy*, Technical report, Northwestern University, 2014.
[10] S. S. Chen, D. L. Donoho, and M. A. Saunders, *Atomic decomposition by basis pursuit*, SIAM J. Sci. Comput., 20 (1998), pp. 33–61.
[11] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, *Algorithm* 887: *CHOLMOD, supernodal sparse Cholesky factorization and update/downdate*, ACM Trans. Math. Software, 35 (2009).
[12] F. H. Clarke, *Generalized gradients and applications*, Trans. Amer. Math. Soc., 205 (1975), pp. 247–262.
[13] P. L. Combettes and J.-C. Pesquet, *Proximal splitting methods in signal processing*, in Fixed-Point Algorithms for Inverse Problems in Science and Engineering, H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, eds., Optim. Appl. 49, Springer, New York, 2011, pp. 185–212.
[14] R. Cominetti, W. F. Mascarenhas, and P. J. S. Silva, *A Newton's method for the continuous quadratic knapsack problem*, Math. Program Comput., 6 (2014), pp. 151–169.
[15] A. R. Conn, N. I. M. Gould, D. Orban, and P. L. Toint, *A primal-dual trust-region algorithm for minimizing a non-convex function subject to general inequality and linear equality constraints*, Math. Program., 87 (1999), pp. 219–249.
[16] F. E. Curtis, Z. Han, and D. P. Robinson, *A globally convergent primal-dual active-set framework for large-scale convex quadratic optimization*, Comput. Optim. Appl., 60 (2015), pp. 311–341, doi:10.1007/s10589-014-9681-9.
[17] J. M. Danskin, *The Theory of Max-Min and Its Applications to Weapons Allocation Problems*, Springer, New York, 1967.
[18] T. A. Davis and W. W. Hager, *Modifying a sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 606–627.
[19] T. A. Davis and W. W. Hager, *Multiple-rank modifications of a sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 997–1013.
[20] T. A. Davis and W. W. Hager, *Row modifications of a sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 621–639.
[21] T. A. Davis and W. W. Hager, *Dual multilevel optimization*, Math. Program., 112 (2008), pp. 403–425.
[22] T. A. Davis and W. W. Hager, *A sparse proximal implementation of the LP Dual Active Set Algorithm*, Math. Program., 112 (2008), pp. 275–301.
[23] T. A. Davis and W. W. Hager, *Dynamic supernodes in sparse Cholesky update/downdate and triangular solves*, ACM Trans. Math. Software, 35 (2009).
[24] T. A. Davis, W. W. Hager, and J. T. Hungerford, *The separable convex quadratic knapsack problem*, ACM Trans. Math. Software, 42 (2016), pp. 1–25, doi:10.1145/2828635.
[25] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.
[26] J. J. Dongarra, J. D. Croz, S. Hammarling, and R. J. Hanson, *Algorithm* 656: *An extended set of Fortran basic linear algebra subprograms*, ACM Trans. Math. Software, 14 (1988), pp. 1–17, 18–32.
[27] J. Dongarra, J. D. Croz, I. S. Duff, and S. Hammarling, *Algorithm* 679: *A set of level 3 basic linear algebra subprograms*, ACM Trans. Math. Software, 16 (1990), pp. 1–17, 18–28.
[28] I. S. Duff, *MA*57*—A code for the solution of sparse symmetric definite and indefinite systems*, ACM Trans. Math. Software, 30 (2004), pp. 118–144.
[29] N. I. M. Gould, D. Orban, and P. L. Toint, *GALAHAD, a library of thread-safe Fortran* 90 *packages for large-scale nonlinear optimization*, ACM Trans. Math. Software, 29 (2004), pp. 253–372.
[30] L. Grippo, F. Lampariello, and S. Lucidi, *A nonmonotone line search technique for Newton's method*, SIAM J. Numer. Anal., 23 (1986), pp. 707–716.
[31] I. Griva, S. G. Nash, and A. Sofer, *Linear and Nonlinear Optimization*, SIAM, Philadelphia, 2009.
[32] W. W. Hager and D. W. Hearn, *Application of the dual active set algorithm to quadratic network optimization*, Comput. Optim. Appl., 1 (1993), pp. 349–373.
[33] W. W. Hager and G. Ianculescu, *Dual approximations in optimal control*, SIAM J. Control Optim., 22 (1984), pp. 423–465.

[34] W. W. Hager, D. T. Phan, and H. Zhang, *Gradient-based methods for sparse recovery*, SIAM J. Imaging Sci., 4 (2011), pp. 146–165.
[35] W. W. Hager and H. Zhang, *A new active set algorithm for box constrained optimization*, SIAM J. Optim., 17 (2006), pp. 526–557.
[36] W. W. Hager, *The dual active set algorithm*, in Advances in Optimization and Parallel Computing, P. M. Pardalos, ed., North-Holland, Amsterdam, 1992, pp. 137–142.
[37] W. W. Hager, *Analysis and implementation of a dual algorithm for constrained optimization*, J. Optim. Theory Appl., 79 (1993), pp. 427–462.
[38] W. W. Hager, *The LP dual active set algorithm*, in High Performance Algorithms and Software in Nonlinear Optimization, R. D. Leone, A. Murli, P. M. Pardalos, and G. Toraldo, eds., Kluwer, Dordrecht, 1998, pp. 243–254.
[39] W. W. Hager, *The dual active set algorithm and its application to linear programming*, Comput. Optim. Appl., 21 (2002), pp. 263–275.
[40] W. W. Hager, *The dual active set algorithm and the iterative solution of linear programs*, in Novel Approaches to Hard Discrete Optimization, P. M. Pardalos and H. Wolkowicz, eds., Fields Inst. Commun. 37, 2003, pp. 95–107.
[41] A. J. Hoffman, *On approximate solutions of systems of linear inequalities*, J. Res. National Bureau Standards, 49 (1952), pp. 263–265.
[42] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, *Basic linear algebra subprograms for Fortran usage*, ACM Trans. Math. Software, 5 (1979), pp. 308–323.
[43] W. Li, *Error bounds for piecewise convex quadratic programs and applications*, SIAM J. Control Optim., 33 (1995), pp. 1510–1529.
[44] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*, Springer, New York, 2008.
[45] D. G. Luenberger, *Optimization by Vector Space Methods*, John Wiley, New York, 1969.
[46] Z.-Q. Luo and P. Tseng, *On the linear convergence of descent methods for convex essentially smooth minimization*, SIAM J. Control Optim., 30 (1992), pp. 408–425.
[47] Z.-Q. Luo and P. Tseng, *Error bounds and convergence analysis of feasible descent methods: A general approach*, Ann. Oper. Res., 46 (1993), pp. 157–178.
[48] Z.-Q. Luo and P. Tseng, *On the convergence rate of dual ascent methods for linearly constrained convex minimization*, Math. Oper. Res., 18 (1993), pp. 846–867.
[49] Z. Lu and Y. Zhang, *An augmented Lagrangian approach for sparse principal component analysis*, Math. Program., 135 (2012), pp. 149–193.
[50] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, 1999.
[51] P. Tseng and S. Yun, *A coordinate gradient descent method for nonsmooth separable minimization*, Math. Program., 117 (2009), pp. 387–423.
[52] M. Ulbrich, S. Ulbrich, and M. Heinkenschloss, *Semismooth Newton methods for operator equations in function spaces*, SIAM J. Optim., 13 (2003), pp. 805–842.
[53] A. Wächter and L. T. Biegler, *On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming*, Math. Program., 106 (2006), pp. 25–57.
[54] P.-W. Wang and C.-H. Lin, *Iteration complexity of feasible descent methods for convex optimization*, J. Mach. Learn. Res., 15 (2014), pp. 1523–1548.
[55] P. Wolfe, *Convergence conditions for ascent methods*, SIAM Rev., 11 (1969), pp. 226–235.
[56] P. Wolfe, *Convergence conditions for ascent methods* II: *Some corrections*, SIAM Rev., 13 (1971), pp. 185–188.
[57] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo, *Sparse reconstruction by separable approximation*, IEEE Trans. Signal Process., 57 (2009), pp. 2479–2493.
[58] Y. Zhang, *On the convergence of a class of infeasible interior-point methods for the horizontal linear complementarity problem*, SIAM J. Optim., 4 (1994), pp. 208–227.