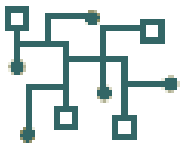

ISE

Industrial and Systems Engineering

Globally Convergent Primal-Dual Active-Set Methods with Inexact Subproblem Solves

Frank E. Curtis and Zheng Han

Department of Industrial and Systems Engineering
Lehigh University, Bethlehem, PA, USA

COR@L Technical Report 14T-10

LEHIGH UNIVERSITY

COR@L
COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

GLOBALLY CONVERGENT PRIMAL-DUAL ACTIVE-SET METHODS WITH INEXACT SUBPROBLEM SOLVES

FRANK E. CURTIS[†‡] AND ZHENG HAN[†§]

Abstract. We propose primal-dual active-set (PDAS) methods for solving large-scale instances of an important class of convex quadratic optimization problems (QPs). The iterates of the algorithms are partitions of the index set of variables, where corresponding to each partition there exist unique primal-dual variables that can be obtained by solving a (reduced) linear system. Algorithms of this type have recently received attention when solving certain QPs and linear complementarity problems (LCPs) since, with rapid changes in the active set estimate, they often converge in few iterations. Indeed, as discussed in this paper, convergence in a finite number of iterations is guaranteed when a basic PDAS method is employed to solve QPs in which the Hessian of the objective function is (a perturbation of) an M-matrix. We propose three PDAS algorithms. The novelty of the algorithms that we propose is that they allow inexactness in the (reduced) linear system solves at all partitions (except optimal ones). Such a feature is particularly important in large-scale settings when one employs iterative Krylov subspace methods to solve these systems. Our first algorithm is convergent on problems for which properties of the Hessian can be exploited to derive explicit bounds to be enforced on the (reduced) linear system residuals, whereas our second and third algorithms employ dynamic parameters to control the residuals (to avoid the computation of such explicit bounds). We prove that, when applied to solve an important class of convex QPs, our algorithms converge from any initial partition. We also illustrate their practical behavior by providing the results of numerical experiments on a pair of discretized optimal control problems.

Key words. convex quadratic optimization, large-scale optimization, active-set methods, semi-smooth Newton methods, inexact Newton methods, Krylov subspace methods

AMS subject classifications. 49M05, 49M15, 65K05, 65K10, 65K15

1. Introduction.

Convex quadratic optimization problems (QPs) arise in numerous areas of applied mathematics [8, 9, 22, 30, 31, 34, 37, 44, 47]. Consequently, algorithms for solving such problems have been studied for decades. These algorithms generally fall into the categories of active-set [6, 16, 40] and interior-point methods [33, 39, 46, 50]. There are also a variety of methods designed exclusively for bound-constrained QPs, which represent an important subclass of the class of QPs considered in this paper. These include active-set [17, 20], interior-point [11, 25], gradient projection [5, 13, 18, 40], or some combination of these methods [7, 24, 38].

In this paper, we propose three primal-dual active-set (PDAS) methods for solving large-scale instances of an important class of QPs. We consider our methods to be enhancements of the method proposed in [26], but remark that our methods are also related to the work in [1, 4, 30, 35], the subsequent work in [12, 31, 32], and the numerous other articles on the use of PDAS methods [14, 23, 27, 28, 29, 36, 43]. The key feature of the algorithm in [26] (and others just cited) is that it allows any number of changes in the active-set estimate during each iteration. This is in contrast to classical primal or dual active-set methods, where the worst-case performance of each can be plagued by slow adaptation of the active-set estimate [3, 19, 21, 40]. The method in [26] is one in which each iterate corresponds to a partition of an index set

[†]Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA.
This material is based upon work supported by the U.S. Department of Energy, Office of Science, Applied Mathematics, Early Career Research Program under Award Number DE–SC0010615 and by the U.S. National Science Foundation, Division of Mathematical Sciences, Computational Mathematics Program under Award Number DMS–1016291.
[‡]E-mail: frank.e.curtis@lehigh.edu
[§]E-mail: zhh210@lehigh.edu

of variables into an active set and an inactive set. Such a partition corresponds to a unique primal-dual solution estimate via a reduced linear system whose dimension depends on the size of the inactive set. Solving this reduced linear system represents the main computational expense during each iteration of the algorithm.

Our contributions in this paper are two-fold. First, we present a convergence result for a flexible PDAS framework that is applicable to a broad class of QPs. In particular, our convergence result applies for problems with certain sets of equality constraints and upper bounds on a subset of variables, and assumes only that a reduced Hessian of the objective has certain structure—which it does in applications of interest, such as certain optimal control problems (see §5). Second, with the goal of enhancing the algorithm in [26], a novel feature of our algorithms is that the reduced linear systems may be solved *inexactly*. This is of particular interest in large-scale settings when the systems are to be solved iteratively, such as with Krylov subspace methods [10, 41, 42, 45]. We present three algorithms, each involving a set of easily implementable conditions to control inexactness in such a way that our convergence guarantees are ensured. These conditions always allow inexactness in the system solves for any partition that is suboptimal. (If a given partition is optimal in that the corresponding primal-dual solution is optimal for the QP, then our algorithms require sufficient accuracy in the subproblem solve so that the overall algorithm will terminate. Clearly, such a requirement is reasonable whenever the partition is optimal.)

We remark at the outset that a straightforward heuristic involving a dynamic accuracy tolerance that decreases as the optimization process proceeds will *not* ensure convergence without strong assumptions on the employed linear system solver. Indeed, such is the approach employed in our third algorithm, for which our convergence guarantees are significantly weaker than for our first two algorithms. (We also provide an example illustrating why the convergence guarantees for such an algorithm must be weaker.) In short, our first two algorithms are able to attain stronger convergence guarantees as they involve procedures for computing an upper bound on the norm of the inverse of a particular submatrix during each iteration; our first algorithm computes such an upper bound explicitly, whereas our second algorithm incorporates a dynamic parameter that (effectively) replaces this upper bound.

This paper is organized as follows. In §2, we state our problem of interest, basic concepts and definitions, and outline our algorithmic framework. In §3, we present our proposed algorithms and corresponding subroutines. We also prove that the algorithms attain convergence guarantees for a certain class of problems of interest. We then discuss an implementation of our algorithm in §4 and provide the results of numerical experiments on discretized optimal control problems in §5 to show that our inexact PDAS methods have advantages over a similar strategy that employs exact linear system solves. Concluding remarks are provided in §6.

Notation. We use index sets as subscripts to denote the subvector or submatrix corresponding to the indices in the given sets. For example, given an ordered set of indices $\mathcal{S}$, by $x_{\mathcal{S}}$ we denote the subvector of the vector x corresponding to the indices in $\mathcal{S}$, and, with another ordered set of indices $\mathcal{T}$, by $H_{\mathcal{S}\mathcal{T}}$ we denote the submatrix of the matrix H with row indices in $\mathcal{S}$ and column indices from $\mathcal{T}$. We also occasionally denote a vector composed of stacked subvectors as an ordered tuple of vectors, i.e., for vectors a and b we occasionally write $(a, b) := [a^T \ b^T]^T$. For a square matrix S, we write $S \succ 0$ ($S \succeq 0$) to indicate that S is positive definite (semidefinite). Finally, e denotes a vector of ones whose size is determined by the context in which it appears.

2. Fundamentals. For a positive integer n and nonnegative integer m, we define an index set of (upper) bounded variables $\mathcal{N} := \{1, \ldots, n\}$, index set of free variables $\mathcal{F} := \{n+1, \ldots, n+m\}$, and index set of equality constraints $\mathcal{M} := \{1, \ldots, m\}$. Then, given problem data in terms of $c \in \mathbb{R}^{n+m}$, $H \in \mathbb{R}^{(n+m)\times(n+m)}$, $A \in \mathbb{R}^{m\times(n+m)}$, $b \in \mathbb{R}^m$, and $u \in \mathbb{R}^n$, we consider the quadratic optimization problem

$$\min_{x\in\mathbb{R}^{n+m}} \; c^T \begin{bmatrix} x_{\mathcal{N}} \\ x_{\mathcal{F}} \end{bmatrix} + \tfrac{1}{2} \begin{bmatrix} x_{\mathcal{N}} \\ x_{\mathcal{F}} \end{bmatrix}^T H \begin{bmatrix} x_{\mathcal{N}} \\ x_{\mathcal{F}} \end{bmatrix} \quad \text{s.t.} \quad A \begin{bmatrix} x_{\mathcal{N}} \\ x_{\mathcal{F}} \end{bmatrix} = b, \; x_{\mathcal{N}} \leq u. \tag{QP}$$

Throughout the paper, we make the following assumption about the matrices in the problem data for a given instance of (QP).

ASSUMPTION 2.1. *In (QP), the Hessian of the objective function satisfies $H \succeq 0$ and $H_{\mathcal{N}\mathcal{N}} \succ 0$, and the constraint data submatrix $A_{\mathcal{M}\mathcal{F}}$ is invertible.*

Under Assumption 2.1, it follows that (QP) is feasible and there exists a unique primal point x and unique Lagrange multipliers (y, z) satisfying the Karush-Kuhn-Tucker (KKT) optimality conditions for (QP), which can be written as

$$0 = \mathrm{KKT}(x, y, z) := \left(c + H \begin{bmatrix} x_{\mathcal{N}} \\ x_{\mathcal{F}} \end{bmatrix} + A^T y + \begin{bmatrix} z \\ 0 \end{bmatrix}, \; A \begin{bmatrix} x_{\mathcal{N}} \\ x_{\mathcal{F}} \end{bmatrix} - b, \; \min\{u - x_{\mathcal{N}}, z\} \right).$$

We define a partition $(\mathcal{A}, \mathcal{I})$ of the index set of bounded variables as a pair of mutually exclusive and exhaustive subsets of $\mathcal{N}$, where $\mathcal{A}$ represents an *active* set of variables (i.e., variables equal to their upper bounds) and $\mathcal{I} = \mathcal{N} \setminus \mathcal{A}$ represents the corresponding *inactive* set. Corresponding to a partition $(\mathcal{A}, \mathcal{I})$, we define a *subspace solution*, call it (x, y, z), by the following sequence of operations:

$$\text{Set} \quad x_{\mathcal{A}} \leftarrow u_{\mathcal{A}} \quad \text{and} \quad z_{\mathcal{I}} \leftarrow 0, \tag{2.1a}$$

$$\text{then solve} \quad \begin{bmatrix} H_{\mathcal{I}\mathcal{I}} & H_{\mathcal{I}\mathcal{F}} & [A_{\mathcal{M}\mathcal{I}}]^T \\ H_{\mathcal{F}\mathcal{I}} & H_{\mathcal{F}\mathcal{F}} & [A_{\mathcal{M}\mathcal{F}}]^T \\ A_{\mathcal{M}\mathcal{I}} & A_{\mathcal{M}\mathcal{F}} & 0 \end{bmatrix} \begin{bmatrix} x_{\mathcal{I}} \\ x_{\mathcal{F}} \\ y \end{bmatrix} = - \begin{bmatrix} c_{\mathcal{I}} \\ c_{\mathcal{F}} \\ -b \end{bmatrix} - \begin{bmatrix} H_{\mathcal{I}\mathcal{A}} \\ H_{\mathcal{F}\mathcal{A}} \\ A_{\mathcal{M}\mathcal{A}} \end{bmatrix} u_{\mathcal{A}} \tag{2.1b}$$

$$\text{for} \quad (x_{\mathcal{I}}, x_{\mathcal{F}}, y),$$

$$\text{then set} \quad z_{\mathcal{A}} \leftarrow -H_{\mathcal{A}\mathcal{N}} x_{\mathcal{N}} - H_{\mathcal{A}\mathcal{F}} x_{\mathcal{F}} - [A_{\mathcal{M}\mathcal{A}}]^T y - c_{\mathcal{A}}. \tag{2.1c}$$

Under Assumption 2.1, the matrix on the left-hand side of (2.1b) is nonsingular, and hence the subspace solution corresponding to $(\mathcal{A}, \mathcal{I})$ is unique. We call $(\mathcal{A}, \mathcal{I})$ an *optimal partition* if its subspace solution (x, y, z) satisfies $\mathrm{KKT}(x, y, z) = 0$, i.e., if (x, y, z) is the optimal primal-dual solution for (QP). Otherwise, the partition $(\mathcal{A}, \mathcal{I})$ and its corresponding subspace solution (x, y, z) are *suboptimal*. We remark that while the optimal primal-dual solution is unique for an instance of (QP), there may be more than one optimal partition. Indeed, more generally, for a given instance of (QP), multiple partitions may correspond to the same subspace solution.

Given a partition $(\mathcal{A}, \mathcal{I})$ and its corresponding subspace solution (x, y, z), we define the *violated sets* of indices of variables as given by

$$\mathcal{V}_P := \{i \in \mathcal{I} : x_i > u_i\} \quad \text{and} \quad \mathcal{V}_D := \{i \in \mathcal{A} : z_i < 0\}. \tag{2.2}$$

(Clearly, these violated sets depend on the partition $(\mathcal{A}, \mathcal{I})$. However, for brevity in our presentation, we do not indicate this dependence in the notation for $\mathcal{V}_P$ and $\mathcal{V}_D$. In all cases, the partition of interest will be clear from the context.) The following result has important consequences that we will use extensively.

THEOREM 2.2. *Let (x, y, z) be the subspace solution corresponding to a given partition $(\mathcal{A}, \mathcal{I})$. Then, $(\mathcal{A}, \mathcal{I})$ is optimal for (QP) if and only if $\mathcal{V}_P \cup \mathcal{V}_D = \emptyset$.*

Proof. By straightforward verification of the KKT conditions for (QP), the subspace solution defined by (2.1) satisfies all KKT conditions, except perhaps subsets of $\min\{u - x_{\mathcal{N}}, z\} = 0$ corresponding to the bounds $x_{\mathcal{I}} \leq u_{\mathcal{I}}$ and $z_{\mathcal{A}} \geq 0$. Subsets of these bounds are violated if and only if the set $\mathcal{V}_P \cup \mathcal{V}_D$ is nonempty. ∎

Consider the framework for solving (QP) that is stated as Algorithm 1 below. Each iteration of Algorithm 1 involves the computation of a subspace solution as defined in (2.1). If the corresponding primal-dual solution estimate yields a zero (or sufficiently small, corresponding to an arbitrary vector norm $\|\cdot\|$) KKT residual, then the solution is (approximately) optimal and the algorithm terminates. Otherwise, subsets of the corresponding violated sets—the union of which is guaranteed by Theorem 2.2 to be nonempty—are chosen, the indices of which are switched from active to inactive, or vice versa, to create a new partition. This algorithm represents a generic framework that allows much flexibility, such as in the choices for $\mathcal{C}_P$ and $\mathcal{C}_D$ in Step 7. In the subsequent sections of this paper, we propose three algorithms related to Algorithm 1 that allow inexactness in each subspace solution; along with these algorithms, details are provided for all algorithmic computations, including how one may consider choosing the sets $\mathcal{C}_P$ and $\mathcal{C}_D$.

Algorithm 1 Primal-Dual Active-Set (PDAS) Framework

1: Input an initial partition $(\mathcal{A}, \mathcal{I})$ and optimality tolerance $\varepsilon_{opt} \geq 0$.
2: **loop**
3: Compute the subspace solution (x, y, z) by (2.1).
4: **if** $\|\text{KKT}(x, y, z)\| \leq \varepsilon_{opt}$ **then**
5: Terminate and **return** (x, y, z).
6: Set $\mathcal{V}_P$ and $\mathcal{V}_D$ by (2.2).
7: Choose $\mathcal{C}_P \subseteq \mathcal{V}_P$ and $\mathcal{C}_D \subseteq \mathcal{V}_D$ such that $\mathcal{C}_P \cup \mathcal{C}_D \neq \emptyset$.
8: Set $\mathcal{A} \leftarrow (\mathcal{A} \backslash \mathcal{C}_D) \cup \mathcal{C}_P$ and $\mathcal{I} \leftarrow (\mathcal{I} \backslash \mathcal{C}_P) \cup \mathcal{C}_D$.

The algorithm in [26] can be viewed as a special case of Algorithm 1. In particular, for the case when $m = 0$ (i.e., $\mathcal{F} = \emptyset$ and $\mathcal{M} = \emptyset$), it corresponds to Algorithm 1 with the choice $\mathcal{C}_P \leftarrow \mathcal{V}_P$ and $\mathcal{C}_D \leftarrow \mathcal{V}_D$ in Step 7 in each iteration. The authors of [26] provide convergence results for their algorithm that are similar to those in Theorem 2.3 below. For our purposes, we state the theorem in a more general setting so that it applies for Algorithm 1 above. A proof is given in Appendix A. For the result, recall that a real symmetric matrix is a P-matrix if all of its principal minors are positive (implying that the matrix is positive definite), and a P-matrix is called an M-matrix if all of its off-diagonal entries are nonpositive. We define $[A]_+ := \max\{0, A\}$ (where the max should be understood component-wise), the KKT system matrix

$$K := \begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix}, \tag{2.3}$$

and the following submatrix of K with its Schur complement with respect to K:

$$Q := \begin{bmatrix} H_{\mathcal{F}\mathcal{F}} & [A_{\mathcal{M}\mathcal{F}}]^T \\ A_{\mathcal{M}\mathcal{F}} & 0 \end{bmatrix} \quad \text{and} \quad R := H_{\mathcal{N}\mathcal{N}} - \begin{bmatrix} H_{\mathcal{F}\mathcal{N}} \\ A_{\mathcal{M}\mathcal{N}} \end{bmatrix}^T Q^{-1} \begin{bmatrix} H_{\mathcal{F}\mathcal{N}} \\ A_{\mathcal{M}\mathcal{N}} \end{bmatrix}. \tag{2.4}$$

THEOREM 2.3. *Suppose that Assumption 2.1 holds and that the matrix R in (2.4) satisfies at least one of the following conditions:*

(a) R is a P-matrix and, corresponding to any partition $(\mathcal{A},\mathcal{I})$, we have that $\|[R_{\mathcal{I}\mathcal{I}}^{-1}R_{\mathcal{I}\mathcal{A}}]_+\|_1 < 1$ and $e^T R_{\mathcal{I}\mathcal{I}}^{-1} w \geq 0$ for any $w \geq 0$, where the latter inequality holds strictly, i.e., $e^T R_{\mathcal{I}\mathcal{I}}^{-1} w > 0$, whenever $w \neq 0$.

(b) $R = M + E$, where M is an M-matrix and $\|E\|_1$ is sufficiently small.

Then, with $\varepsilon_{opt} \geq 0$ and any initial partition, Algorithm 1 terminates in a finite number of iterations. In particular, if $\varepsilon_{opt} = 0$, then Algorithm 1 terminates in a finite number of iterations with a KKT point for (QP).

We remark that, under condition (b) in Theorem 2.3, a notion of how small $\|E\|_1$ must be is revealed in the proof in Appendix A.

3. Algorithm Descriptions. In this section, we propose three algorithms for solving (QP). Each algorithm has the same basic structure as Algorithm 1, but allows inexactness in the reduced linear system solves. In the first algorithm that we propose, a tolerance for inexactness is set based on an upper bound on a norm of a particular submatrix. We illustrate that such a bound can be computed efficiently in certain cases of interest. In the second and third algorithms, the inexactness tolerance is set based on a parameter that is updated dynamically within the algorithm. For all algorithms, we prove that the guarantees of Theorem 2.3 are maintained.

The algorithms in this section employ an extension of the operations specified in (2.1). In particular, corresponding to a partition $(\mathcal{A},\mathcal{I})$, we define an *inexact subspace solution*, call it $(\tilde{x},\tilde{y},\tilde{z})$, by the following operations (where by "$\approx$" in (3.1b) we are indicating that the "solve" may be approximate):

$$\text{Set} \quad \tilde{x}_{\mathcal{A}} \leftarrow u_{\mathcal{A}} \quad \text{and} \quad \tilde{z}_{\mathcal{I}} \leftarrow 0, \tag{3.1a}$$

$$\text{then solve} \quad \begin{bmatrix} H_{\mathcal{I}\mathcal{I}} & H_{\mathcal{I}\mathcal{F}} & [A_{\mathcal{M}\mathcal{I}}]^T \\ H_{\mathcal{F}\mathcal{I}} & H_{\mathcal{F}\mathcal{F}} & [A_{\mathcal{M}\mathcal{F}}]^T \\ A_{\mathcal{M}\mathcal{I}} & A_{\mathcal{M}\mathcal{F}} & 0 \end{bmatrix} \begin{bmatrix} x_{\mathcal{I}} \\ x_{\mathcal{F}} \\ y \end{bmatrix} \approx - \begin{bmatrix} c_{\mathcal{I}} \\ c_{\mathcal{F}} \\ -b \end{bmatrix} - \begin{bmatrix} H_{\mathcal{I}\mathcal{A}} \\ H_{\mathcal{F}\mathcal{A}} \\ A_{\mathcal{M}\mathcal{A}} \end{bmatrix} u_{\mathcal{A}} \tag{3.1b}$$

$$\text{for} \quad (\tilde{x}_{\mathcal{I}},\tilde{x}_{\mathcal{F}},\tilde{y}),$$

$$\text{then set} \quad \tilde{z}_{\mathcal{A}} \leftarrow -H_{\mathcal{A}\mathcal{N}}\tilde{x}_{\mathcal{N}} - H_{\mathcal{A}\mathcal{F}}\tilde{x}_{\mathcal{F}} - [A_{\mathcal{M}\mathcal{A}}]^T\tilde{y} - c_{\mathcal{A}} \tag{3.1c}$$

$$\text{and} \quad \begin{bmatrix} \tilde{r}_{\mathcal{N}} \\ \tilde{r}_{\mathcal{F}} \\ \tilde{t} \end{bmatrix} \leftarrow \begin{bmatrix} c_{\mathcal{N}} \\ c_{\mathcal{F}} \\ -b \end{bmatrix} + \begin{bmatrix} H_{\mathcal{N}\mathcal{N}} & H_{\mathcal{N}\mathcal{F}} & [A_{\mathcal{M}\mathcal{N}}]^T \\ H_{\mathcal{F}\mathcal{N}} & H_{\mathcal{F}\mathcal{F}} & [A_{\mathcal{M}\mathcal{F}}]^T \\ A_{\mathcal{M}\mathcal{N}} & A_{\mathcal{M}\mathcal{F}} & 0 \end{bmatrix} \begin{bmatrix} \tilde{x}_{\mathcal{N}} \\ \tilde{x}_{\mathcal{F}} \\ \tilde{y} \end{bmatrix} + \begin{bmatrix} \tilde{z} \\ 0 \\ 0 \end{bmatrix}. \tag{3.1d}$$

The vector $(\tilde{r},\tilde{t})$ is the residual in the (reduced) linear system solve that produces $(\tilde{x},\tilde{y},\tilde{z})$ via (3.1b). Under Assumption 2.1, we find by comparing (2.1) and (3.1) that $(\tilde{x},\tilde{y},\tilde{z}) = (x,y,z)$ if and only if $(\tilde{r},\tilde{t}) = 0$.

In each of the algorithms proposed in this section, we iteratively solve the linear system in (3.1b) until either it is verified that the partition $(\mathcal{A},\mathcal{I})$ is optimal (with respect to a tolerance $\varepsilon_{opt} \geq 0$) or the inexact subspace solution is sufficiently accurate in that it leads to a productive update of the partition. In our first algorithm, we provide a strategy in which we identify subsets of the violated sets $\mathcal{V}_P$ and $\mathcal{V}_D$ corresponding to the *exact* subspace solution (x,y,z) *without* having to explicitly compute this exact solution. In this manner, the algorithm fits into the framework of Algorithm 1. In our other algorithms, we do not necessarily identify subsets of these violated sets, though we can still ensure convergence guarantees by employing and appropriately updating a dynamic algorithmic parameter.

3.1. An Algorithm with a Partition-Defined Subproblem Tolerance. Our first algorithm imposes a tolerance on the residual $(\tilde{r},\tilde{t})$ defined in (3.1d) that is based on a partition-defined value with which we can guarantee that at any suboptimal

partition a subset of the union $\mathcal{V}_P \cup \mathcal{V}_D$ corresponding to the *exact* subspace solution (x, y, z) will be identified. In order to motivate the tolerance that we employ, we first explore, for a given partition $(\mathcal{A}, \mathcal{I})$, the relationship between the subspace solution (x, y, z) defined by (2.1) and an inexact subspace solution $(\tilde{x}, \tilde{y}, \tilde{z})$ defined by (3.1). Then, once the tolerance is established, we present our first inexact PDAS algorithm, followed by subsections in which we discuss details of subroutines of the algorithm. It is important to note that we assume that a subroutine is available for computing *exact* solutions of linear systems with Q in (2.4); i.e., we assume products with Q^{-1} can be computed. This is a reasonable assumption in certain applications, especially since the matrix Q is fixed, i.e., it does not depend on the partition.

Given a partition $(\mathcal{A}, \mathcal{I})$ and recalling (2.3), consider the decomposition

$$K = \begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} H_{\mathcal{A}\mathcal{A}} \end{bmatrix} & \begin{bmatrix} H_{\mathcal{A}\mathcal{I}} & S_{[\mathcal{A}]}^T \end{bmatrix} \\ \begin{bmatrix} H_{\mathcal{I}\mathcal{A}} \\ S_{[\mathcal{A}]} \end{bmatrix} & \begin{bmatrix} H_{\mathcal{I}\mathcal{I}} & S_{[\mathcal{I}]}^T \\ S_{[\mathcal{I}]} & Q \end{bmatrix} \end{bmatrix}, \tag{3.2}$$

where (using a subscript $[\cdot]$ to indicate dependence on an index set) we define

$$K_{[\mathcal{I}]} := \begin{bmatrix} H_{\mathcal{I}\mathcal{I}} & S_{[\mathcal{I}]}^T \\ S_{[\mathcal{I}]} & Q \end{bmatrix}, \quad S_{[\mathcal{A}]} := \begin{bmatrix} H_{\mathcal{F}\mathcal{A}} \\ A_{\mathcal{M}\mathcal{A}} \end{bmatrix}, \quad \text{and} \quad S_{[\mathcal{I}]} := \begin{bmatrix} H_{\mathcal{F}\mathcal{I}} \\ A_{\mathcal{M}\mathcal{I}} \end{bmatrix}.$$

(Note that $K_{[\mathcal{N}]} = K$ and $R_{\mathcal{I}\mathcal{I}} = H_{\mathcal{I}\mathcal{I}} - S_{[\mathcal{I}]}^T Q^{-1} S_{[\mathcal{I}]}$ is the Schur complement of Q with respect to $K_{[\mathcal{I}]}$.) Observing (2.1) and (3.1), it follows that $\tilde{x}_{\mathcal{A}} = u_{\mathcal{A}} = x_{\mathcal{A}}$ and $\tilde{z}_{\mathcal{I}} = 0 = z_{\mathcal{I}}$. In addition, defining the residual subvector $\tilde{v} := (\tilde{r}_{\mathcal{F}}, \tilde{t})$, we have

$$x_{\mathcal{I}} = \tilde{x}_{\mathcal{I}} - R_{\mathcal{I}\mathcal{I}}^{-1} \tilde{r}_{\mathcal{I}} + R_{\mathcal{I}\mathcal{I}}^{-1} S_{[\mathcal{I}]}^T Q^{-1} \tilde{v} \tag{3.3a}$$

$$\begin{aligned} \text{and} \quad z_{\mathcal{A}} = \tilde{z}_{\mathcal{A}} &+ (H_{\mathcal{A}\mathcal{I}} - S_{[\mathcal{A}]}^T Q^{-1} S_{[\mathcal{I}]}) R_{\mathcal{I}\mathcal{I}}^{-1} \tilde{r}_{\mathcal{I}} \\ &- (H_{\mathcal{A}\mathcal{I}} R_{\mathcal{I}\mathcal{I}}^{-1} S_{[\mathcal{I}]}^T - S_{[\mathcal{A}]}^T - S_{[\mathcal{A}]}^T Q^{-1} S_{[\mathcal{I}]} R_{\mathcal{I}\mathcal{I}}^{-1} S_{[\mathcal{I}]}^T) Q^{-1} \tilde{v}. \end{aligned} \tag{3.3b}$$

Observing (3.3), it follows that the violated sets $\mathcal{V}_P$ and $\mathcal{V}_D$ corresponding to (x, y, z) can be defined in terms of an inexact subspace solution $(\tilde{x}, \tilde{y}, \tilde{z})$ and its residual $(\tilde{r}, \tilde{t})$. In particular, for an index $i \in \mathcal{I}$, the ith element of x violates its upper bound if the corresponding element on the right-hand side of (3.3a) is greater than u_i, and, for an index $i \in \mathcal{A}$, the ith element of z violates its lower bound (of zero) if the corresponding element on the right-hand side of (3.3b) is negative. This revised viewpoint of the elements of $x_{\mathcal{I}}$ and $z_{\mathcal{A}}$ does not immediately yield any benefits since the evaluation of the terms on the right-hand sides of (3.3) is equivalent to that of solving (3.1b) exactly. However, it reveals that with an inexact subspace solution $(\tilde{x}, \tilde{y}, \tilde{z})$ and bounds on the residual terms in (3.3), we may identify subsets of $\mathcal{V}_P$ and $\mathcal{V}_D$ without computing (x, y, z) explicitly. The following lemma suggests a strategy for such a procedure.

LEMMA 3.1. *Given a partition $(\mathcal{A}, \mathcal{I})$, let (x, y, z) be the corresponding subspace solution and let $(\tilde{x}, \tilde{y}, \tilde{z})$ be an inexact subspace solution with residual $(\tilde{r}, \tilde{t})$. Furthermore, suppose that with $\tilde{v} := (\tilde{r}_{\mathcal{F}}, \tilde{t})$ we have $\alpha_{\mathcal{I}}$ and $\beta_{\mathcal{A}}$ satisfying*

$$\alpha_{\mathcal{I}} \geq R_{\mathcal{I}\mathcal{I}}^{-1} \tilde{r}_{\mathcal{I}} - R_{\mathcal{I}\mathcal{I}}^{-1} S_{[\mathcal{I}]} Q^{-1} \tilde{v} \tag{3.4a}$$

$$\begin{aligned} \textit{and} \quad \beta_{\mathcal{A}} \geq (H_{\mathcal{A}\mathcal{I}} &- S_{[\mathcal{A}]}^T Q^{-1} S_{[\mathcal{I}]}) R_{\mathcal{I}\mathcal{I}}^{-1} \tilde{r}_{\mathcal{I}} \\ &- (H_{\mathcal{A}\mathcal{I}} R_{\mathcal{I}\mathcal{I}}^{-1} S_{[\mathcal{I}]}^T - S_{[\mathcal{A}]}^T - S_{[\mathcal{A}]}^T Q^{-1} S_{[\mathcal{I}]} R_{\mathcal{I}\mathcal{I}}^{-1} S_{[\mathcal{I}]}^T) Q^{-1} \tilde{v}. \end{aligned} \tag{3.4b}$$

Then, for the violated sets $\mathcal{V}_P$ and $\mathcal{V}_D$ corresponding to (x, y, z), we have

$$\tilde{\mathcal{V}}_P := \{i \in \mathcal{I} : \tilde{x}_i - \alpha_i > u_i\} \subseteq \mathcal{V}_P \quad \textit{and} \quad \tilde{\mathcal{V}}_D := \{i \in \mathcal{A} : \tilde{z}_i + \beta_i < 0\} \subseteq \mathcal{V}_D. \tag{3.5}$$

Moreover, if $(\mathcal{A},\mathcal{I})$ *is suboptimal, then there exists* $\varepsilon > 0$ *such that* (3.4)–(3.5) *with*

$$\|\alpha_{\mathcal{I}}\| = \mathcal{O}(\|(\tilde{r},\tilde{t})\|) \quad \textit{and} \quad \|\beta_{\mathcal{A}}\| = \mathcal{O}(\|(\tilde{r},\tilde{t})\|) \tag{3.6}$$

yields $\tilde{\mathcal{V}}_P = \mathcal{V}_P$ *and* $\tilde{\mathcal{V}}_D = \mathcal{V}_D$ *for any inexact subspace solution with* $\|(\tilde{r},\tilde{t})\| \le \varepsilon$.

Proof. By (3.4) and the relationships in (3.3), we have that for $i \in \mathcal{I}$ the inequality $\tilde{x}_i - \alpha_i > u_i$ implies $x_i > u_i$, and for $i \in \mathcal{A}$ the inequality $\tilde{z}_i + \beta_i < 0$ implies $z_i < 0$. Hence, with (3.5), it follows that $\tilde{\mathcal{V}}_P \subseteq \mathcal{V}_P$ and $\tilde{\mathcal{V}}_D \subseteq \mathcal{V}_D$.

Now suppose that $(\mathcal{A},\mathcal{I})$ is suboptimal, from which it follows by Theorem 2.2 that $\mathcal{V}_P \cup \mathcal{V}_D \neq \emptyset$. If $\mathcal{V}_P \neq \emptyset$, then for any $i \in \mathcal{V}_P$ we have $x_i > u_i$. Moreover, by continuity of the linear transformation defined by the inverse of the matrix on the left-hand side of (3.1b), for this $i \in \mathcal{V}_P$ there exists $\varepsilon_i > 0$ such that for any $(\tilde{r},\tilde{t})$ with $\|(\tilde{r},\tilde{t})\| \le \varepsilon_i$, the condition (3.6) implies $\tilde{x}_i - \alpha_i > u_i$. Similar analysis shows that if $\mathcal{V}_D \neq \emptyset$, then for any $i \in \mathcal{V}_D$ there exists $\varepsilon_i > 0$ such that for any $(\tilde{r},\tilde{t})$ with $\|(\tilde{r},\tilde{t})\| \le \varepsilon_i$, the condition (3.6) implies $\tilde{z}_i + \beta_i < 0$. Since $\tilde{\mathcal{V}}_P \subseteq \mathcal{V}_P$ and $\tilde{\mathcal{V}}_D \subseteq \mathcal{V}_D$, it follows that with $\varepsilon := \min\{\varepsilon_i : i \in \mathcal{V}_P \cup \mathcal{V}_D\} > 0$, we have $\tilde{\mathcal{V}}_P = \mathcal{V}_P$ and $\tilde{\mathcal{V}}_D = \mathcal{V}_D$. ∎

Lemma 3.1 proves that at any suboptimal partition $(\mathcal{A},\mathcal{I})$, a subset of the union of violated sets $\mathcal{V}_P \cup \mathcal{V}_D$ can be identified as long as upper bounds $\alpha_{\mathcal{I}}$ and $\beta_{\mathcal{A}}$ are available and are proportional (in terms of any given norm $\|\cdot\|$) to the residual vector $(\tilde{r},\tilde{t})$. On the other hand, if a partition $(\mathcal{A},\mathcal{I})$ is optimal, then with a sufficiently small residual we obtain a sufficiently accurate primal-dual solution. Motivated by these observations, we propose the inexact PDAS framework presented as Algorithm 2.

Algorithm 2 PDAS Framework with Inexact Subspace Solutions

1: Input an initial partition $(\mathcal{A},\mathcal{I})$ and optimality tolerance $\varepsilon_{opt} \ge 0$.
2: **loop**
3: Compute an inexact subspace solution $(\tilde{x},\tilde{y},\tilde{z})$ with residual $(\tilde{r},\tilde{t})$ by (3.1).
4: **if** $\|\mathrm{KKT}(\tilde{x},\tilde{y},\tilde{z})\| \le \varepsilon_{opt}$ **then**
5: Terminate and **return** $(\tilde{x},\tilde{y},\tilde{z})$.
6: Compute $\alpha_{\mathcal{I}}$ and $\beta_{\mathcal{A}}$ satisfying (3.4) and (3.6).
7: Set $\tilde{\mathcal{V}}_P$ and $\tilde{\mathcal{V}}_D$ by (3.5).
8: **if** $\tilde{\mathcal{V}}_P \cup \tilde{\mathcal{V}}_D \neq \emptyset$ **then**
9: Choose $\mathcal{C}_P \subseteq \tilde{\mathcal{V}}_P$ and $\mathcal{C}_D \subseteq \tilde{\mathcal{V}}_D$ such that $\mathcal{C}_P \cup \mathcal{C}_D \neq \emptyset$.
10: Set $\mathcal{A} \leftarrow (\mathcal{A}\backslash\mathcal{C}_D) \cup \mathcal{C}_P$ and $\mathcal{I} \leftarrow (\mathcal{I}\backslash\mathcal{C}_P) \cup \mathcal{C}_D$.

We have the following result, indicating conditions on the inexact subspace solutions computed in the algorithm—or, more precisely, on their corresponding residual vectors—that are necessary to ensure convergence.

THEOREM 3.2. *Suppose that the conditions of Theorem 2.3 hold for the partitions generated by Algorithm 2. Then, the following hold:*

(i) If, for any partition, repeated executions of Step 3 yield $(\tilde{r},\tilde{t}) \to 0$*, then, with* $\varepsilon_{opt} > 0$*, Algorithm 2 terminates after a finite number of partition updates.*

(ii) If there exists a positive integer J *such that, for any partition, at most* J *executions of Step 3 yields* $(\tilde{r},\tilde{t}) = 0$*, then, with* $\varepsilon_{opt} \ge 0$*, Algorithm 2 terminates in a finite number of iterations. In particular, if* $\varepsilon_{opt} = 0$*, then Algorithm 2 terminates in a finite number of iterations with a KKT point for* (QP).

Proof. Given any partition, it follows by the conditions in (i) and Lemma 3.1 that after a finite number of executions of Step 3, the sets $\tilde{\mathcal{V}}_P \subseteq \mathcal{V}_P$ and $\tilde{\mathcal{V}}_D \subseteq \mathcal{V}_D$ defined by (3.5) satisfy $\tilde{\mathcal{V}}_P \cup \tilde{\mathcal{V}}_D \neq \emptyset$. The result in (i) then follows from Theorem 2.3 and

the fact that after a finite number of partition updates, a partition is identified such that repeated executions of Step 3 yield $(\tilde{x}, \tilde{y}, \tilde{z})$ satisfying the condition in Step 5 of the algorithm. The result in (ii) follows in a similar manner due to the additional observation that, given any partition, at most J executions of Step 3 occur before the condition in Step 5 is satisfied or the partition is modified. □

There remain many details that need to be specified for a practical implementation of Algorithm 2. These details are the subjects of the following three subsections. First, due to the observation that upper bounds $\alpha_{\mathcal{I}}$ and $\beta_{\mathcal{A}}$ in Step 6 are easily computed once one obtains an upper bound for the norm of the matrix $R_{\mathcal{I}\mathcal{I}}^{-1}$, we present an algorithm for computing such a bound in certain cases of interest. Second, we present a generic technique for computing $\alpha_{\mathcal{I}}$ and $\beta_{\mathcal{A}}$ once such a bound is obtained. Third, we outline conditions that one may choose to impose—in addition to the condition that $\tilde{\mathcal{V}}_P \cup \tilde{\mathcal{V}}_D \neq \emptyset$—in the **if** statement in Step 8 of Algorithm 2.

3.1.1. Obtaining an upper bound for $\|R_{\mathcal{I}\mathcal{I}}^{-1}\|_1$. In this subsection, given an index set $\mathcal{I}$, we present a technique for computing an upper bound for $\|R_{\mathcal{I}\mathcal{I}}^{-1}\|_1$. Our technique amounts to solving a linear system of equations by an iterative process, where the computed upper bound may become tighter as the linear system is solved more accurately. (Indeed, under certain conditions, an exact solution of the linear system reveals $\|R_{\mathcal{I}\mathcal{I}}^{-1}\|_1$.) In this manner, it is clear how the upper bounds $\alpha_{\mathcal{I}}$ and $\beta_{\mathcal{A}}$ required in Step 6 may be improved during the loop of Algorithm 2: For a given partition $(\mathcal{A}, \mathcal{I})$, repeated executions of the algorithm in this subsection eventually lead to a tighter upper bound for $\|R_{\mathcal{I}\mathcal{I}}^{-1}\|_1$, which in turn eventually lead to tighter upper bounds $\alpha_{\mathcal{I}}$ and $\beta_{\mathcal{A}}$ via the algorithm in the following subsection.

Given a symmetric $p \times p$ matrix $B \succ 0$, consider the set

$$\mathcal{P}(B) := \{w \in \mathbb{R}^p : w > 0,\ Bw > 0,\ \|w\|_\infty = 1\}$$

and the symmetric $p \times p$ matrix $\mathfrak{M}(B)$ defined, for all $\{i, j\} \subseteq \{1, \ldots, p\}$, by

$$[\mathfrak{M}(B)]_{ij} = \begin{cases} |B_{ii}| & \text{if } i = j \\ -|B_{ij}| & \text{if } i \neq j. \end{cases}$$

The matrix B is called an H-matrix if and only if $\mathfrak{M}(B)$ is a nonsingular M-matrix. Moreover, according to [49, eq. (5)], the following three statements are equivalent:

1. B is an H-matrix.
2. $\mathfrak{M}(B)$ is an M-matrix.
3. $\mathcal{P}(B)$ is nonempty.

We also have the following result, which is a special case of [49, Theorem 1]. (In [49], the author discusses upper bounds for the ℓ_∞-norm of a matrix inverse. However, since our matrix is symmetric, we can equivalently refer to its ℓ_1-norm.)

LEMMA 3.3. *If $B \in \mathbb{R}^{p \times p}$ is a symmetric H-matrix, then, for any $w \in \mathcal{P}(B)$,*

$$\|B^{-1}\|_1 \leq \|\mathfrak{M}(B)^{-1}\|_1 \leq (\min\{[Aw]_i : 1 \leq i \leq p\})^{-1},$$

where the second inequality holds as an equality when $w = B^{-1}e/\|B^{-1}e\|_\infty \in \mathcal{P}(B)$.

By Lemma 3.3, if $R_{\mathcal{I}\mathcal{I}}$ is an H-matrix, then we can obtain an upper bound for $\|R_{\mathcal{I}\mathcal{I}}^{-1}\|_1$ by iteratively solving the linear system $\mathfrak{M}(R_{\mathcal{I}\mathcal{I}})w = e$ for $w \in \mathbb{R}^{|\mathcal{I}|}$, terminating whenever an element of $\mathcal{P}(R_{\mathcal{I}\mathcal{I}})$ is obtained. We formalize this strategy as Algorithm 3, for which we have the following result.

Algorithm 3 Subroutine for Obtaining an Upper Bound of $\|R_{\mathcal{I}\mathcal{I}}^{-1}\|_1$

1: Input $R_{\mathcal{I}\mathcal{I}}$.
2: **loop**
3: Compute an inexact solution w of $\mathfrak{M}(R_{\mathcal{I}\mathcal{I}})w = e$.
4: **if** $\mathfrak{M}(R_{\mathcal{I}\mathcal{I}})w > 0$ **then**
5: Terminate and **return**

$$\|w\|_\infty(\min\{[R_{\mathcal{I}\mathcal{I}}w]_i : 1 \le i \le |\mathcal{I}|\})^{-1}.$$

LEMMA 3.4. *Suppose Assumption 2.1 holds and R satisfies $R = M + E$ where M is an M-matrix and $\|E\|_1$ is sufficiently small. Then, for any $\mathcal{I} \subseteq \mathcal{N}$, if repeated executions of Step 3 of Algorithm 3 yield $\mathfrak{M}(R_{\mathcal{I}\mathcal{I}})w \to e$, then Algorithm 3 will terminate and return an upper bound for $\|R_{\mathcal{I}\mathcal{I}}^{-1}\|_1$.*

Proof. Since $R = M + E$, we have $R_{\mathcal{I}\mathcal{I}} = M_{\mathcal{I}\mathcal{I}} + E_{\mathcal{I}\mathcal{I}}$ for some M-matrix $M_{\mathcal{I}\mathcal{I}}$. Since $M_{\mathcal{I}\mathcal{I}}$ is an M-matrix, it is also an H-matrix since $\mathfrak{M}(M_{\mathcal{I}\mathcal{I}}) = M_{\mathcal{I}\mathcal{I}}$. Then, since $M_{\mathcal{I}\mathcal{I}}$ is an H-matrix, it follows that for sufficiently small $\|E\|_1$ we have sufficiently small $\|E_{\mathcal{I}\mathcal{I}}\|_1$ such that $R_{\mathcal{I}\mathcal{I}}$ is also an H-matrix. The result then follows by Lemma 3.3 since, under the conditions of the lemma, the algorithm eventually computes w satisfying $\mathfrak{M}(R_{\mathcal{I}\mathcal{I}})w > 0$. ∎

We close this subsection by remarking that if $R_{\mathcal{I}\mathcal{I}}$ is strictly diagonally dominant, then by computing $w = e$ in Step 3, Algorithm 3 would terminate in the first iteration of the **loop** and return the upper bound given by

$$\|R_{\mathcal{I}\mathcal{I}}^{-1}\|_1 \le (\min\{[R_{\mathcal{I}\mathcal{I}}e]_i : 1 \le i \le |\mathcal{I}|\})^{-1}.$$

This is known as the Ahlberg-Nilson-Varah bound [2, 48].

3.1.2. Obtaining upper bounds $\alpha_\mathcal{I}$ and $\beta_\mathcal{A}$. Given a partition $(\mathcal{A}, \mathcal{I})$, an inexact subspace solution $(\tilde{x}, \tilde{y}, \tilde{z})$ with residual $(\tilde{r}, \tilde{t})$, and an upper bound for a norm of $R_{\mathcal{I}\mathcal{I}}^{-1}$ (say, computed via Algorithm 3), in this subsection we present a generic method for computing upper bounds $\alpha_\mathcal{I}$ and $\beta_\mathcal{A}$ satisfying (3.4) and (3.6).

We present our technique as the following algorithm. For the approach, we recall that since Q is independent of the partition, we suppose that this matrix is formed and factorized at the start of the optimization process so that products with Q^{-1} are available. Thus, the computational cost of executing the algorithm is relatively low, especially if the matrix $Q^{-1}S_{[\mathcal{N}]}$ is precomputed.

Algorithm 4 Subroutine for Obtaining $\alpha_\mathcal{I}$ and $\beta_\mathcal{A}$ Satisfying (3.4) and (3.6)

1: Input $p \ge 1$, $q \ge 1$, and $\gamma \ge \|R_{\mathcal{I}\mathcal{I}}^{-1}\|_p$ such that $\|\cdot\|_p$ and $\|\cdot\|_q$ are dual norms.
2: Terminate and **return**

$$\alpha_\mathcal{I} \leftarrow (\gamma\|\tilde{r}_\mathcal{I} - S_{[\mathcal{I}]}Q^{-1}\tilde{v}\|_q)e$$
$$\text{and} \quad \beta_\mathcal{A} \leftarrow [H_{\mathcal{A}\mathcal{I}} - S_{[\mathcal{A}]}^T Q^{-1}S_{[\mathcal{I}]}]_+\alpha_\mathcal{I} - [H_{\mathcal{A}\mathcal{I}} - S_{[\mathcal{A}]}^T Q^{-1}S_{[\mathcal{I}]}]_-\alpha_\mathcal{I} + S_{[\mathcal{A}]}Q^{-1}\tilde{v}.$$

The fact that the upper bounds $\alpha_\mathcal{I}$ and $\beta_\mathcal{A}$ generated by Algorithm 4 satisfy (3.4) and (3.6) follows as a consequence of the following lemma. Recall that we define e as

a vector of ones whose length is determined by the context in which it appears. For the proof of the lemma, we also define e_i as the corresponding ith unit vector.

LEMMA 3.5. *Consider $B \in \mathbb{R}^{m\times n}$ with ith row denoted by B_i for $i \in \{1,\dots,m\}$, a vector $w \in \mathbb{R}^n$, and a vector norm $\|\cdot\|_p$ with corresponding dual norm $\|\cdot\|_q$. Then, $B_i w \le \|B_i^T\|_p \|w\|_q$ for all $i \in \{1,\dots,m\}$. Furthermore, $Bw \le \|B^T\|_p \|w\|_q e$.*

Proof. Hölder's inequality implies that $|B_i w| \le \|B_i^T\|_p \|w\|_q$ for all $i \in \{1,\dots,m\}$, as desired. Then, the remainder of the results follows since

$$|[Bw]_i| = |B_i w| \le \|B_i^T\|_p \|w\|_q \le \|B^T\|_p \|w\|_q \quad \text{for all } i \in \{1,\dots,m\},$$

where the last inequality follows from the fact that $\|B_i\|_p = \|B^T e_i\|_p \le \|B^T\|_p$. ∎

We remark that when employing Algorithm 3 to compute an upper bound for $\|R_{\mathcal{I}\mathcal{I}}^{-1}\|_1$, it is natural to employ Algorithm 4 with $p = 1$ and $q = \infty$. This is the approach used in our implementation and numerical experiments.

3.1.3. Conditions for executing a partition update. We close our discussion of Algorithm 2 by describing conditions that one may choose to impose—in addition to the (generally very loose) requirement that $\tilde{\mathcal{V}}_P \cup \tilde{\mathcal{V}}_D \neq \emptyset$—in the **if** statement in Step 8. As already shown in Theorem 3.2, the convergence of the algorithm is guaranteed with the condition as it is stated in the algorithm. Therefore, the addition of extra conditions is not necessary to ensure convergence. However, in our experience, it is worthwhile to impose extra conditions to ensure that any update to the partition that is performed will lead to substantial progress toward a solution, which can be expected to be the case when either the inexact subspace solution is sufficiently accurate and/or the modification to the partition will involve a large number of indices switching from active to inactive, or vice versa. We have found the conditions that we state in this section to work well in practice, though one may imagine other possible conditions that could be imposed.

Let $(\tilde{x}', \tilde{y}', \tilde{z}')$ be a given inexact subspace solution with residual $(\tilde{r}', \tilde{t}')$. For example, one may consider $(\tilde{x}', \tilde{y}', \tilde{z}') = (0, 0, 0)$ or the inexact subspace solution corresponding to primal-dual variable values as computed in the previous iteration of Algorithm 2. Given a tolerance $\epsilon_{res} \in (0, 1)$ and a vector norm $\|\cdot\|$, a condition that one may choose to impose is the following, similar to conditions typically found in inexact Newton methods for solving systems of equations [15]:

$$\|(\tilde{r}, \tilde{t})\| \le \epsilon_{res} \|(\tilde{r}', \tilde{t}')\|. \tag{3.7}$$

That is, one may choose not to modify the partition until the residual vector $(\tilde{r}, \tilde{t})$ is sufficiently small in norm compared to the reference residual $(\tilde{r}', \tilde{t}')$ corresponding to the reference solution $(\tilde{x}', \tilde{y}', \tilde{z}')$. If the right-hand side of (3.7) is zero, then $(x, y, z) = (\tilde{x}', \tilde{y}', \tilde{z}')$; otherwise, (3.7) will eventually be satisfied as long as the employed iterative solver ensures that the residual vanishes, i.e., $(\tilde{r}, \tilde{t}) \to 0$.

In our experience, we have also found it beneficial to avoid modifying the partition until there is consistency between the sets $\tilde{\mathcal{V}}_P$ and $\tilde{\mathcal{V}}_D$ in (3.5) and

$$\tilde{\mathcal{V}}_P' := \{i \in \mathcal{I} : \tilde{x}_i > u_i\} \quad \text{and} \quad \tilde{\mathcal{V}}_D' := \{i \in \mathcal{A} : \tilde{z}_i < 0\}, \tag{3.8}$$

potentially with the latter set replaced by

$$\tilde{\mathcal{V}}_D' := \{i \in \mathcal{A} : \tilde{z}_i + [S_{[\mathcal{A}]} Q^{-1} \tilde{v}]_i < 0\}. \tag{3.9}$$

Specifically, we have found it beneficial to avoid modifying the partition until the number of elements of the violated sets that have been identified (as measured by

$|\tilde{\mathcal{V}}_P \cup \tilde{\mathcal{V}}_D|$) is proportional to the number of elements of the primal-dual components that violate their bounds (as measured by $|\tilde{\mathcal{V}}'_P \cup \tilde{\mathcal{V}}'_D|$). Given a parameter $\theta \in (0, 1]$, we write this condition as

$$|\tilde{\mathcal{V}}_P \cup \tilde{\mathcal{V}}_D| \geq \theta |\tilde{\mathcal{V}}'_P \cup \tilde{\mathcal{V}}'_D|. \tag{3.10}$$

Observe that Algorithm 4 yields $\alpha_{\mathcal{I}} \geq 0$ and $\beta_{\mathcal{A}} \geq 0$, from which it follows that $|\tilde{\mathcal{V}}_P \cup \tilde{\mathcal{V}}_D| \leq |\tilde{\mathcal{V}}'_P \cup \tilde{\mathcal{V}}'_D|$. This justifies the restriction that $\theta \in (0, 1]$.

3.2. Algorithms with Dynamic Subproblem Tolerances. We are now prepared to present our second and third inexact PDAS algorithms. For a given partition $(\mathcal{A}, \mathcal{I})$, the main idea underlying our first method—i.e., Algorithm 2 presented in §3.1—was to use properties of inexact subspace solutions and their corresponding residuals in order to construct explicit subsets of the violated sets $\mathcal{V}_P$ and $\mathcal{V}_D$ corresponding to the exact subspace solution, all without having to explicitly compute this exact solution. Unfortunately, however, the procedure that we proposed required an explicit upper bound for a norm of $R_{\mathcal{I}\mathcal{I}}^{-1}$, which may be expensive to compute in certain situations, especially when a tight bound is needed to identify elements of the violated sets. By contrast, the algorithms that we propose in this section do not require explicit upper bounds of this type; instead, they involve dynamic parameters either to estimate such an upper bound or control inexactness directly.

Our second algorithm, stated as Algorithm 5 below, employs a dynamic parameter that plays a role similar to the upper bound for a norm of $R_{\mathcal{I}\mathcal{I}}^{-1}$ as employed in Algorithm 2. In the worst case, this dynamic parameter will increase large enough such that it is, in fact, an upper bound for a norm of $R_{\mathcal{I}\mathcal{I}}^{-1}$ (for any $\mathcal{I}$); indeed, the convergence guarantees that we present for Algorithm 5 are based on this feature. However, we have designed the update for this dynamic parameter such that we rarely see such behavior in practice. Indeed, in practice, we often observe that the algorithm terminates for relatively small values of this dynamic parameter. As in Algorithm 2, the norm used in the optimality test in Step 5 can be any vector norm; we also add that, in the context of this algorithm, it is natural to use the same norm in Step 12. On the other hand, the norm to which we refer in Step 7 should be the norm $\|\cdot\|_p$ with $p \geq 1$ employed in Algorithm 4 for computing the values $\alpha_{\mathcal{I}}$ and $\beta_{\mathcal{A}}$.

The purpose of the sequence $\{\text{KKT}_j\}$ computed in Algorithm 5 is to monitor progress in reducing the KKT error over the sequence of iterations in which the partition is modified. Specifically, if a KKT error computed in Step 12 is not less than the most recent $\bar{j}$ such computed KKT errors, then the dynamic parameter γ is increased. As can be seen in the procedure in Algorithm 4, this has the effect of yielding larger values for $\alpha_{\mathcal{I}}$ and $\beta_{\mathcal{A}}$, which in turn has the effect of producing more conservative estimates (i.e., $\tilde{\mathcal{V}}_P$ and $\tilde{\mathcal{V}}_D$) of the violated sets (i.e., $\mathcal{V}_P$ and $\mathcal{V}_D$).

We have the following theorem related to convergence properties of Algorithm 5.

THEOREM 3.6. *Suppose that the conditions of Theorem 2.3 hold for the partitions generated by Algorithm 5. Then, the following hold:*

- *(i) If, for any partition, repeated executions of Step 4 yield $(\tilde{r}, \tilde{t}) \to 0$, then, with $\varepsilon_{opt} > 0$, Algorithm 5 terminates after a finite number of partition updates.*
- *(ii) If there exists a positive integer J such that, for any partition, at most J executions of Step 4 yields $(\tilde{r}, \tilde{t}) = 0$, then, with $\varepsilon_{opt} \geq 0$, Algorithm 5 terminates in a finite number of iterations. In particular, if $\varepsilon_{opt} = 0$, then Algorithm 5 terminates in a finite number of iterations with a KKT point for (QP).*

Proof. Given any partition, it follows by the conditions in either (i) or (ii) that after a finite number of executions of Step 4, the sets $\tilde{\mathcal{V}}_P$ and $\tilde{\mathcal{V}}_D$ defined by (3.5)

Algorithm 5 PDAS Framework with Inexact Subspace Solutions (Dynamic)

1: Input an initial partition $(\mathcal{A}, \mathcal{I})$, optimality tolerance $\varepsilon_{opt} > 0$, dynamic parameter $\gamma > 0$, update factor $\delta_\gamma > 1$, optimality tolerance history length $\bar{j} \in \mathbb{N}$, and sufficient reduction factor $\kappa \in (0, 1)$.
2: Initialize a partition update counter $j \leftarrow 0$ and $\mathrm{KKT}_j \leftarrow \infty$ for $j \in \{-1, \ldots, -\bar{j}\}$.
3: **loop**
4: Compute an inexact subspace solution $(\tilde{x}, \tilde{y}, \tilde{z})$ with residual $(\tilde{r}, \tilde{t})$ by (3.1).
5: **if** $\|\mathrm{KKT}(\tilde{x}, \tilde{y}, \tilde{z})\| \leq \varepsilon_{opt}$ **then**
6: Terminate and **return** $(\tilde{x}, \tilde{y}, \tilde{z})$.
7: Compute $\alpha_{\mathcal{I}}$ and $\beta_{\mathcal{A}}$ by Algorithm 4 with input γ (even if $\gamma \not\geq \|R_{\mathcal{I}\mathcal{I}}^{-1}\|_p$).
8: Set $\tilde{\mathcal{V}}_P$ and $\tilde{\mathcal{V}}_D$ by (3.5) (even if $\tilde{\mathcal{V}}_P \not\subseteq \mathcal{V}_P$ and/or $\tilde{\mathcal{V}}_D \not\subseteq \mathcal{V}_D$).
9: **if** $\tilde{\mathcal{V}}_P \cup \tilde{\mathcal{V}}_D \neq \emptyset$ **then**
10: Choose $\mathcal{C}_P \subseteq \tilde{\mathcal{V}}_P$ and $\mathcal{C}_D \subseteq \tilde{\mathcal{V}}_D$ such that $\mathcal{C}_P \cup \mathcal{C}_D \neq \emptyset$.
11: Set $\mathcal{A} \leftarrow (\mathcal{A} \backslash \mathcal{C}_D) \cup \mathcal{C}_P$ and $\mathcal{I} \leftarrow (\mathcal{I} \backslash \mathcal{C}_P) \cup \mathcal{C}_D$.
12: Set $\mathrm{KKT}_j \leftarrow \|\mathrm{KKT}(\tilde{x}, \tilde{y}, \tilde{z})\|$.
13: **if** $\mathrm{KKT}_j \geq \kappa \max\{\mathrm{KKT}_{j-1}, \ldots, \mathrm{KKT}_{j-\bar{j}}\}$ **then**
14: Set $\gamma \leftarrow \delta_\gamma \gamma$.
15: Set $j \leftarrow j + 1$.

satisfy $\tilde{\mathcal{V}}_P \cup \tilde{\mathcal{V}}_D \neq \emptyset$ (despite the fact that we may have $\tilde{\mathcal{V}}_P \not\subseteq \mathcal{V}_P$ and/or $\tilde{\mathcal{V}}_D \not\subseteq \mathcal{V}_D$). Consequently, given any partition, Algorithm 5 will eventually either terminate or a partition update will be performed. If the algorithm terminates finitely, then there is nothing left to prove. Hence, in order to derive a contradiction, suppose that an infinite number of partition updates are performed. If $\{\mathrm{KKT}_j\} \to 0$, then, under the conditions in either (i) or (ii), the optimality condition in Step 5 eventually will be satisfied; this would cause the algorithm to terminate finitely, a contradiction to our supposition that an infinite number of partition updates are performed. Thus, we may assume that $\{\mathrm{KKT}_j\}$ is bounded below by a positive constant, which, by the condition in Step 13, implies that $\gamma \to \infty$. However, once

$$\gamma \geq \bar{\gamma} := \max_{\mathcal{I} \subseteq \mathcal{N}} \|R_{\mathcal{I}\mathcal{I}}^{-1}\|_p, \tag{3.11}$$

it follows that we will always have $\tilde{\mathcal{V}}_P \subseteq \mathcal{V}_P$ and $\tilde{\mathcal{V}}_D \subseteq \mathcal{V}_D$, implying that convergence can be guaranteed as the same manner as in the proof of Theorem 2. Since this contradicts our supposition that an infinite number of partition updates are performed, we have that the algorithm will terminate finitely, as desired. □

One important observation about Algorithm 5 is that it is nontrivial to choose an initial value for the dynamic parameter γ such that all iterations performed in the algorithm may be identical to those that would be performed by Algorithm 2. For example, assuming that it may be computed efficiently, one may consider an initial value of $\gamma \leftarrow \|R^{-1}\|_1$ (independently of any partition), but this value does not necessarily satisfy (3.11). To see this, consider the following example.

EXAMPLE 3.7. *Let*

$$R = \begin{bmatrix} \frac{7}{6} & \frac{1}{6} & \frac{29}{12} \\ \frac{1}{6} & \frac{1}{6} & \frac{5}{12} \\ \frac{29}{12} & \frac{5}{12} & \frac{143}{24} \end{bmatrix} \quad \textit{and} \quad \mathcal{I} = \{1, 2\}.$$

One can verify that $\|R^{-1}\|_1 = \frac{31}{4} < 8 = \|R_{\mathcal{I}\mathcal{I}}^{-1}\|_1$.

Our third inexact PDAS algorithm, presented as Algorithm 6 below, employs the straightforward heuristic of defining a dynamic tolerance for the residuals in the (reduced) linear system solves that decreases as the optimization process proceeds. In this algorithm, it is reasonable to choose the norm in Step 7 as the same norm used in the KKT residual checks in Steps 5 and 11. (We also note that a relative residual test—rather than an absolute residual test—could be employed, as we do in our implementation and numerical experiments.)

Algorithm 6 PDAS Framework with Inexact Subspace Solutions (Dynamic)

1: Input an initial partition $(\mathcal{A}, \mathcal{I})$, optimality tolerance $\varepsilon_{opt} > 0$, dynamic parameter $\zeta > 0$, update factor $\delta_\zeta > 1$, optimality tolerance history length $\bar{j} \in \mathbb{N}$, and sufficient reduction factor $\kappa \in (0, 1)$.
2: Initialize a partition update counter $j \leftarrow 0$ and $\text{KKT}_j \leftarrow \infty$ for $j \in \{-1, \ldots, -\bar{j}\}$.
3: **loop**
4: Compute an inexact subspace solution $(\tilde{x}, \tilde{y}, \tilde{z})$ with residual $(\tilde{r}, \tilde{t})$ by (3.1).
5: **if** $\|\text{KKT}(\tilde{x}, \tilde{y}, \tilde{z})\| \leq \varepsilon_{opt}$ **then**
6: Terminate and **return** $(\tilde{x}, \tilde{y}, \tilde{z})$.
7: **if** $\|(\tilde{r}, \tilde{t})\| \leq \zeta$ **then**
8: Set $\tilde{\mathcal{V}}'_P$ and $\tilde{\mathcal{V}}'_D$ by (3.8) (even if $\tilde{\mathcal{V}}'_P \not\subseteq \mathcal{V}_P$ and/or $\tilde{\mathcal{V}}'_D \not\subseteq \mathcal{V}_D$).
9: Choose $\mathcal{C}_P \subseteq \tilde{\mathcal{V}}'_P$ and $\mathcal{C}_D \subseteq \tilde{\mathcal{V}}'_D$ such that $\mathcal{C}_P \cup \mathcal{C}_D \neq \emptyset$.
10: Set $\mathcal{A} \leftarrow (\mathcal{A} \backslash \mathcal{C}_D) \cup \mathcal{C}_P$ and $\mathcal{I} \leftarrow (\mathcal{I} \backslash \mathcal{C}_P) \cup \mathcal{C}_D$.
11: Set $\text{KKT}_j \leftarrow \|\text{KKT}(\tilde{x}, \tilde{y}, \tilde{z})\|$.
12: **if** $\text{KKT}_j \geq \kappa \max\{\text{KKT}_{j-1}, \ldots, \text{KKT}_{j-\bar{j}}\}$ **then**
13: Set $\zeta \leftarrow \zeta / \delta_\zeta$.
14: Set $j \leftarrow j + 1$.

We have the following convergence result for Algorithm 6.

THEOREM 3.8. *Suppose that the conditions of Theorem 2.3 hold for the partitions generated by Algorithm 6. If, after a finite number of partition updates, the executions of Step 4 ensure that the condition in Step 7 only ever holds true when $\|(\tilde{r}, \tilde{t})\| = 0$, then, with $\varepsilon_{opt} \geq 0$, Algorithm 6 terminates in a finite number of iterations. In particular, if $\varepsilon_{opt} = 0$, then Algorithm 6 terminates in a finite number of iterations with a KKT point for (QP).*

Proof. Under the conditions of the theorem, it follows that after a finite number of partition updates the algorithm behaves as if (exact) subspace solutions were computed via (2.1). Hence, the result follows similarly as Theorem 2.3. ∎

Despite the fact that Algorithm 6 employs a straightforward heuristic for controlling inexactness and has the advantage that a factorization of Q is not required, it has two key disadvantages vis-à-vis Algorithms 2 and 5. First, as a practical matter, our experience suggests that it is much more difficult to choose (initial) values for ζ and δ_ζ that lead to good performance on a wide range of problems. Second, our convergence guarantee for Algorithm 6 is significantly weaker than those for Algorithms 2 and 5. The following example illustrates why it is not possible to obtain the same convergence guarantees in Theorem 3.8 as we have stated in Theorems 3.2 and 3.6. In particular, the example shows that in order to ensure that $\tilde{\mathcal{V}}'_P \subseteq \mathcal{V}_P$, $\tilde{\mathcal{V}}'_D \subseteq \mathcal{V}_D$,

and $\tilde{\mathcal{V}}'_P \cup \tilde{\mathcal{V}}'_D \neq \emptyset$, one may need a linear system residual that is exactly zero.

EXAMPLE 3.9. *Let*

$$m = 0, \quad H = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}, \quad c = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, \quad \text{and} \quad u = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

With $(\mathcal{A}, \mathcal{I}) = (\{1,2\}, \{3\})$, *it follows from* (2.1) *that the subspace solution is*

$$x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{and} \quad z = \begin{bmatrix} -3 \\ 0 \\ 0 \end{bmatrix},$$

from which it follows that $\mathcal{V}_P = \emptyset$ *and* $\mathcal{V}_D = \{1\}$. *In particular,* $(\mathcal{A}, \mathcal{I})$ *is suboptimal. Moreover, from* (3.1), *any inexact subspace solution* $(\tilde{x}, \tilde{z})$ *satisfies*

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \tilde{x}_3 - x_3 \end{bmatrix} + \begin{bmatrix} \tilde{z}_1 - z_1 \\ \tilde{z}_2 - z_2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \tilde{r}_3 \end{bmatrix},$$

which implies that $(\tilde{z}_1 - z_1) = 0$ *and* $\tilde{r}_3 = 2(\tilde{x}_3 - x_3) = 2(\tilde{z}_2 - z_2)$. *Hence, in order to have* $\tilde{\mathcal{V}}'_P \subseteq \mathcal{V}_P$, $\tilde{\mathcal{V}}'_D \subseteq \mathcal{V}_D$, *and* $\tilde{\mathcal{V}}'_P \cup \tilde{\mathcal{V}}'_D \neq \emptyset$, *one must have*

$$\tilde{x}_3 = x_3 + \tfrac{1}{2}\tilde{r}_3 = 1 + \tfrac{1}{2}\tilde{r}_3 \leq 1$$
$$\text{and} \quad \tilde{z}_2 = z_2 + \tfrac{1}{2}\tilde{r}_2 = 0 + \tfrac{1}{2}\tilde{r}_3 \geq 0,$$

i.e., one must have $\tilde{r}_3 = 0$. *(On the other hand, one can verify that with* $\gamma = 1 \geq \|H_{33}^{-1}\|_1$, *Algorithms 2 and 5 obtain* $\tilde{\mathcal{V}}_P = \emptyset = \mathcal{V}_P$ *and* $\tilde{\mathcal{V}}_D = \{1\} = \mathcal{V}_D$ *for* $|\tilde{r}_3| < 3$.*)*

4. An Implementation. We have written implementations of Algorithms 1, 2, 5, and 6 along with the subroutines described as Algorithms 3 and 4. We discuss common and distinguishing details of the implementations in this section.

All algorithms are implemented in a single piece of software in Python 2.7.3. The software uses the infrastructure in `cvxopt` for matrix storage and manipulation, as well as the implementation of MINRES [41] provided in `Scipy` for (approximately) solving the linear systems (2.1b) and (3.1b). For Algorithms 2 and 5, the matrix Q is factored at the start of each run using `cvxopt`'s interface to `LAPACK`; specifically, the `sytrf` subroutine is called to compute an LDL^T factorization of Q, the factors of which are employed to compute products with Q^{-1} as they are needed.

Our use of MINRES as the iterative solver for the linear systems (2.1b) and (3.1b) is not required; any iterative method for solving symmetric indefinite systems could be employed. It should also be noted that while preconditioning would be a critical aspect of any efficient implementation of either of our algorithms, we did not implement a preconditioner in our software. We claim that this is reasonable as the purpose of our numerical experiments is merely to illustrate the convergence behavior of our algorithms despite inexactness in the subspace solutions; in particular, the speed at which these inexact subspace solutions are obtained is not a focus of our experiments, which only serve as a "proof of concept" for our algorithms.

In all algorithms, the initial point provided to MINRES in the first PDAS iteration is a randomly generated vector, whereas, in subsequent PDAS iterations, the initial point is set by extracting the corresponding elements of the primal-dual solution corresponding to the previous PDAS iterate. For Algorithm 1 in which "exact" solutions

are required, each run of MINRES terminates once the ℓ_∞-norm of the residual for the linear system (2.1b) is reduced below a prescribed tolerance $\epsilon_{num} > 0$. Similarly, for Algorithm 6, MINRES terminates once either (3.7) holds (with ϵ_{res} replaced by ζ) or the ℓ_∞-norm of the residual for the linear system (3.1b) is reduced below ϵ_{num}. Finally, for Algorithms 2 and 5, MINRES terminates either once (3.7) and (3.10) both hold or the ℓ_∞-norm of the residual for the linear system (3.1b) is reduced below ϵ_{num}. (The only exceptions occur when $\|(\tilde{r}, \tilde{t})\|_\infty \leq \epsilon_{num}$, but $|\tilde{\mathcal{V}}_P \cup \tilde{\mathcal{V}}_D| = |\tilde{\mathcal{V}}'_P \cup \tilde{\mathcal{V}}'_D| = 0$ and $\|\mathrm{KKT}(\tilde{x}, \tilde{y}, \tilde{z})\|_\infty > \epsilon_{opt}$, in which case MINRES is forced to continue.)

In terms of Algorithms 2 and 5, the evaluation of vectors in Algorithm 4 requires products with Q^{-1} (i.e., solves with the factors of Q), meaning that it is not economical to run this subroutine after every MINRES iteration. Hence, our software performs 100 MINRES iterations in Step 3 of Algorithm 2 and Step 4 of Algorithm 5. In both cases, we use $p = 1$ in Algorithm 4.

Finally, for Algorithm 5, we implemented an additional strategy for updating γ that utilizes intermediate vectors computed by MINRES. (The purpose of this strategy is to use problem information to quickly adjust γ if the initial value is set too low for a given run of the algorithm.) In particular, with an intermediate solution $\tilde{x}_\mathcal{I}$ and product $R_{\mathcal{I}\mathcal{I}}\tilde{x}_\mathcal{I}$, both provided at no extra cost by MINRES, we set

$$\gamma \leftarrow \max\left\{\gamma, \frac{\|\tilde{x}_\mathcal{I}\|_1}{\|R_{\mathcal{I}\mathcal{I}}\tilde{x}_\mathcal{I}\|_1}\right\}.$$

This update is motivated by the fact that

$$\|R_{\mathcal{I}\mathcal{I}}^{-1}\|_1 = \max_{\|x\|_1=1} \|R_{\mathcal{I}\mathcal{I}}x\|_1^{-1}. \tag{4.1}$$

5. Numerical Results. In this section, we report on the performance of our implementations of Algorithms 1, 2, 5, and 6 when they were employed to solve two optimal control test problems. In fact, the problems we consider are the same as those considered in [26]. We remark at the outset that, for consistency with notation commonly used in the context of optimal control, the decision variables in the test problems presented in this section are a state variable y and a control variable u. In addition, we use $x = (x_1, x_2)$ to denote the coordinate axes in $\mathbb{R}^2$. Overall, the reader should be aware that the pair (x, y) in this section should not be confused with the primal-dual variable pair (x, y) used in the previous sections.

Given a domain $\Omega \in \mathbb{R}^2$, reference function $z \in L^2(\Omega)$, upper bound function $\psi \in L^2(\Omega)$, and regularization parameter $\beta > 0$, we consider the test problems

$$\begin{aligned} &\min_{y,u} \ \tfrac{1}{2}\|y - z\|^2_{L^2(\Omega)} + \tfrac{\beta}{2}\|u\|^2_{L^2(\Omega)} \\ &\text{s.t.} \ \begin{cases} -\Delta y = u & \text{in } \Omega \\ y = 0 & \text{on } \partial\Omega \\ u \leq \psi & \text{in } \Omega, \end{cases} \end{aligned} \tag{5.1}$$

and, with n denoting the unit outer normal to Ω along $\partial\Omega$,

$$\begin{aligned} &\min_{y,u} \ \tfrac{1}{2}\|y - z\|^2_{L^2(\Omega)} + \tfrac{\beta}{2}\|u\|^2_{L^2(\partial\Omega)} \\ &\text{s.t.} \ \begin{cases} -\Delta y + y = 0 & \text{in } \Omega \\ \frac{\partial y}{\partial n} = u & \text{on } \partial\Omega \\ u \leq \psi & \text{on } \partial\Omega. \end{cases} \end{aligned} \tag{5.2}$$

In particular, as in [26], we let

$$\Omega = [0,1]^2, \quad z(x_1, x_2) = \sin(5x_1) + \cos(4x_2), \quad \psi = 0, \quad \text{and} \quad \beta = 10^{-5}.$$

In order to illustrate the performance of our implementation of our algorithms on instances of various sizes, we generated discretized versions of problems (5.1) and (5.2) at various levels of discretization. In particular, we generated instances of both problems with the numbers of grid points along each dimension in the set $\{20, 40, 60, 80, 100\}$. A five-point-star discretization of Δ was used and the functions z, ψ, y, and u were discretized by means of grid functions at the nodal points. It is easily verified that all of the resulting problem instances have the form (QP) satisfying Assumption 2.1. Table 5 contains the sizes of each problem instance in terms of the numbers of grid points per dimension (`g`), variables (`n`), and equality constraints (`m`). For each instance of each problem, all algorithms were initialized with the same initial partition, which was generated randomly for all problem instances.

TABLE 5.1
Problem sizes.

	(5.1)		(5.2)	
g	n	m	n	m
20	800	400	560	480
40	3200	1600	1920	1760
60	7200	3600	4080	3840
80	12800	6400	7040	6720
100	20000	10000	10800	10400

For our experiments, we used the input parameters in Table 5.2. These values were used as they lead to good performance in our experiments, though it should be noted that, in any application, these parameters should be tuned for optimal performance. As for the dynamic parameter $\gamma > 0$ in Algorithm 5, it was initialized to 10^2, and as for the dynamic parameter $\zeta > 0$ in Algorithm 6, it was initialized to ϵ_{res}.

TABLE 5.2
Input parameters for our implementations of Algorithms 1, 2, 5, and 6.

Parameter	Value	Algorithm(s)
ϵ_{opt}	10^{-6}	1, 2, 5, 6
ϵ_{num}	10^{-6}	1, 2, 5, 6
ϵ_{res}	$\{10^{-2}, 10^{-3}\}$	2, 5, 6
θ	0.5	2, 5
δ_γ	1.2	5
δ_ζ	1.2	6
$\bar{j}$	5	5, 6
κ	0.9	5, 6

The performance of the algorithms in solving problem (5.1) is reported in Tables 5.3–5.9 below. In each table, we report the numbers of grid points per dimension and PDAS iterations required before termination (`Iter.`) for each instance. We also report, to illustrate the accuracy with which the (reduced) linear systems were solved in each run of the algorithm, the minimum (`min`), median (`med`), and maximum (`min`) relative residual (`Rel.Res.`) and absolute residual (`Abs.Res.`) over all PDAS iterations. (It should be noted that, due to diagonal dominance of the matrix involved,

all runs of Algorithm 2 required only one Krylov iteration per PDAS iteration to compute the upper bound on $\|R_{\mathcal{I}\mathcal{I}}^{-1}\|_1$ in Algorithm 3. Hence, this subroutine did not lead to a significant increase in computational expense.)

TABLE 5.3
Algorithm 1 when solving problem (5.1).

		Rel.Res.			Abs.Res.		
g	Iter.	min	med	max	min	med	max
20	7	2.82e-07	1.88e-06	2.94e-05	7.57e-07	9.22e-07	9.98e-07
40	8	2.44e-07	7.74e-06	5.08e-03	7.30e-07	9.88e-07	9.99e-07
60	8	2.92e-07	1.75e-05	1.71e-03	8.75e-07	9.97e-07	1.00e-06
80	8	3.20e-07	2.96e-05	4.97e-03	9.61e-07	9.97e-07	1.00e-06
100	8	3.22e-07	3.55e-05	6.99e-03	9.66e-07	9.99e-07	1.00e-06

TABLE 5.4
Algorithm 2 when solving problem (5.1) with $\epsilon_{res} = 10^{-2}$.

		Rel.Res.			Abs.Res.		
g	Iter.	min	med	max	min	med	max
20	10	1.71e-04	8.47e-04	8.96e-03	8.51e-07	2.84e-04	2.39e-02
40	12	3.21e-04	3.41e-03	1.31e-02	9.63e-07	2.44e-04	2.89e-02
60	11	8.70e-04	3.33e-03	5.08e-01	9.95e-07	9.98e-05	2.55e-02
80	12	2.10e-03	4.67e-03	8.11e-02	9.95e-07	8.76e-05	2.77e-02
100	13	3.30e-03	8.44e-03	4.81e-01	9.99e-07	1.21e-04	2.58e-02

TABLE 5.5
Algorithm 2 when solving problem (5.1) with $\epsilon_{res} = 10^{-3}$.

		Rel.Res.			Abs.Res.		
g	Iter.	min	med	max	min	med	max
20	9	1.71e-04	6.17e-04	9.20e-04	8.51e-07	2.28e-04	1.90e-03
40	10	3.24e-04	7.10e-04	1.31e-02	9.63e-07	4.86e-05	2.24e-03
60	10	1.60e-04	8.43e-04	5.08e-01	9.95e-07	2.37e-05	2.14e-03
80	9	2.26e-04	6.95e-04	9.49e-03	9.95e-07	1.53e-05	2.98e-03
100	9	2.89e-04	7.53e-04	2.65e-02	9.99e-07	1.02e-05	2.94e-03

TABLE 5.6
Algorithm 5 when solving problem (5.1) with $\epsilon_{res} = 10^{-2}$.

		Rel.Res.			Abs.Res.		
g	Iter.	min	med	max	min	med	max
20	9	3.05e-05	6.25e-03	8.96e-03	8.51e-07	1.26e-03	2.39e-02
40	9	6.43e-04	8.75e-03	1.03e-02	9.97e-07	4.34e-04	2.89e-02
60	10	1.91e-03	6.91e-03	4.65e-02	9.95e-07	2.52e-04	2.55e-02
80	9	2.29e-03	6.26e-03	9.39e-03	9.95e-07	1.24e-04	2.77e-02
100	9	3.40e-03	8.36e-03	9.31e-03	1.00e-06	1.79e-04	2.58e-02

TABLE 5.7
Algorithm 5 when solving problem (5.1) with $\epsilon_{res} = 10^{-3}$.

		Rel.Res.			Abs.Res.		
g	Iter.	min	med	max	min	med	max
20	8	1.71e-04	5.96e-04	6.66e-04	8.51e-07	9.37e-05	1.90e-03
40	8	2.19e-04	5.21e-04	1.03e-02	9.97e-07	5.42e-05	2.24e-03
60	8	1.26e-04	4.52e-04	1.72e-03	9.95e-07	2.38e-05	2.14e-03
80	8	3.87e-04	6.47e-04	5.00e-03	9.95e-07	2.09e-05	2.98e-03
100	9	2.35e-04	7.33e-04	4.81e-01	9.99e-07	1.50e-05	2.94e-03

TABLE 5.8
Algorithm 6 when solving problem (5.1) with $\epsilon_{res} = 10^{-2}$.

		Rel.Res.			Abs.Res.		
g	Iter.	min	med	max	min	med	max
20	11	5.38e-04	9.74e-03	1.07e-02	8.51e-07	1.05e-03	2.87e-02
40	9	5.26e-03	9.80e-03	9.99e-03	9.97e-07	4.58e-04	2.93e-02
60	10	9.39e-03	9.93e-03	9.55e-02	9.95e-07	2.86e-04	2.97e-02
80	11	9.53e-03	9.98e-03	6.26e-01	9.95e-07	2.18e-04	2.97e-02
100	10	9.51e-03	9.97e-03	7.66e-01	1.00e-06	2.22e-04	2.99e-02

TABLE 5.9
Algorithm 6 when solving problem (5.1) with $\epsilon_{res} = 10^{-3}$.

		Rel.Res.			Abs.Res.		
g	Iter.	min	med	max	min	med	max
20	8	7.35e-04	9.49e-04	1.07e-02	8.51e-07	2.80e-04	2.67e-03
40	9	3.38e-04	9.61e-04	1.16e-01	9.63e-07	8.33e-05	2.87e-03
60	9	9.46e-04	9.82e-04	9.39e-03	9.95e-07	4.43e-05	2.93e-03
80	8	9.62e-04	9.97e-04	1.60e-03	9.95e-07	3.29e-05	2.99e-03
100	9	9.00e-04	9.92e-04	4.81e-01	9.99e-07	2.40e-05	3.00e-03

The results in Tables 5.3–5.9 provide evidence for the computational benefits of allowing inexactness in the subspace solutions. In particular, when applied to solve instances of various sizes, Algorithms 2, 5, and 6 converge in a number of PDAS iterations that is comparable to that required by Algorithm 1; however, this convergence is attained with substantially larger relative and absolute residuals in the (reduced) linear system solves. With a reasonable preconditioner for the linear systems, such larger relative and absolute residuals would be obtainable with significantly fewer Krylov iterations, potentially yielding significant savings in computational expense.

The performance of the algorithms in solving instances of problem (5.2) is reported in Tables 5.10–5.16. (It is worthwhile to note that, again, the computational expense of applying Algorithm 3 was negligible in the context of Algorithm 2.)

TABLE 5.10
Algorithm 1 when solving problem (5.2).

		Rel.Res.			Abs.Res.		
g	Iter.	min	med	max	min	med	max
20	8	2.06e-07	4.60e-07	2.97e-06	7.81e-07	9.66e-07	1.00e-06
40	12	1.33e-07	4.02e-07	2.56e-06	8.85e-07	9.91e-07	1.00e-06
60	17	9.02e-08	3.01e-07	2.78e-06	9.53e-07	9.94e-07	1.00e-06
80	22	7.10e-08	2.61e-07	2.86e-06	9.65e-07	9.92e-07	9.99e-07
100	26	5.75e-08	2.37e-07	3.05e-06	9.64e-07	9.97e-07	1.00e-06

TABLE 5.11
Algorithm 2 when solving problem (5.2) with $\epsilon_{res} = 10^{-2}$.

		Rel.Res.			Abs.Res.		
g	Iter.	min	med	max	min	med	max
20	9	3.67e-07	1.47e-03	8.22e-03	1.18e-07	9.78e-04	3.79e-02
40	20	9.93e-08	5.50e-03	9.96e-03	3.91e-08	2.90e-03	3.69e-02
60	38	6.58e-07	3.06e-03	9.49e-03	2.18e-07	1.95e-03	3.13e-02
80	50	6.25e-07	3.50e-03	9.92e-03	1.93e-07	1.60e-03	3.53e-02
100	72	2.30e-06	5.60e-03	9.97e-03	5.52e-07	1.57e-03	2.21e-02

TABLE 5.12
Algorithm 2 when solving problem (5.2) with $\epsilon_{res} = 10^{-3}$.

		Rel.Res.			Abs.Res.		
g	Iter.	min	med	max	min	med	max
20	8	3.53e-07	7.36e-06	8.19e-04	1.18e-07	1.66e-05	3.78e-03
40	14	1.11e-07	5.68e-04	9.95e-04	3.91e-08	1.05e-03	4.79e-03
60	21	6.58e-07	7.78e-04	9.93e-04	2.18e-07	1.25e-03	4.16e-03
80	34	5.63e-07	7.33e-04	9.98e-04	1.93e-07	8.59e-04	3.45e-03
100	39	1.80e-06	7.23e-04	9.98e-04	5.52e-07	1.38e-03	4.05e-03

TABLE 5.13
Algorithm 5 when solving problem (5.2) with $\epsilon_{res} = 10^{-2}$.

		Rel.Res.			Abs.Res.		
g	Iter.	min	med	max	min	med	max
20	8	3.53e-07	1.19e-05	8.22e-03	1.18e-07	2.46e-05	3.79e-02
40	16	1.09e-07	1.19e-04	7.99e-03	3.91e-08	1.02e-04	3.69e-02
60	21	2.45e-07	3.14e-05	6.79e-03	2.18e-07	9.16e-05	3.13e-02
80	28	2.70e-07	2.27e-05	4.96e-03	1.93e-07	4.81e-05	2.29e-02
100	33	2.30e-07	5.09e-06	3.89e-03	5.52e-07	1.47e-05	1.79e-02

TABLE 5.14
Algorithm 5 when solving problem (5.2) with $\epsilon_{res} = 10^{-3}$.

		Rel.Res.			Abs.Res.		
g	Iter.	min	med	max	min	med	max
20	8	3.53e-07	1.19e-05	8.19e-04	1.18e-07	2.46e-05	3.78e-03
40	16	1.11e-07	4.67e-05	8.40e-04	3.91e-08	5.46e-05	3.88e-03
60	21	2.45e-07	2.96e-05	9.02e-04	2.18e-07	9.16e-05	4.16e-03
80	28	2.70e-07	2.23e-05	7.49e-04	1.93e-07	4.81e-05	3.45e-03
100	33	2.30e-07	4.76e-06	8.78e-04	5.52e-07	1.47e-05	4.05e-03

TABLE 5.15
Algorithm 6 when solving problem (5.2) with $\epsilon_{res} = 10^{-2}$.

		Rel.Res.			Abs.Res.		
g	Iter.	min	med	max	min	med	max
20	19	3.06e-06	6.72e-03	9.96e-03	9.16e-07	9.48e-03	4.58e-02
40	31	2.67e-06	3.34e-03	9.98e-03	9.52e-07	4.53e-03	4.60e-02
60	44	2.76e-06	2.78e-03	1.00e-02	9.84e-07	1.23e-03	1.18e-01
80	62	2.88e-06	1.93e-03	9.98e-03	9.94e-07	1.47e-03	1.26e-01
100	69	2.90e-06	1.12e-03	1.00e-02	9.77e-07	8.44e-04	1.01e-01

TABLE 5.16
Algorithm 6 when solving problem (5.2) with $\epsilon_{res} = 10^{-3}$.

		Rel.Res.			Abs.Res.		
g	Iter.	min	med	max	min	med	max
20	9	2.77e-06	9.65e-04	9.99e-04	9.16e-07	1.69e-03	5.85e-03
40	16	2.67e-06	5.77e-04	9.99e-04	9.52e-07	1.64e-03	9.19e-03
60	40	2.81e-06	4.59e-04	9.99e-04	9.84e-07	1.05e-03	4.61e-03
80	37	2.91e-06	2.12e-04	1.00e-03	9.94e-07	8.53e-04	4.60e-03
100	40	2.91e-06	1.34e-04	1.00e-03	9.77e-07	4.37e-04	4.60e-03

In contrast to those for problem (5.1), it is clear from the results for problem (5.2) that the number of PDAS iterations required before termination is mesh dependent. Thus, these results more clearly illustrate the trade-off between saving per-iteration computational costs with overall costs, as well as the sensitivity of parameter choices in our inexact PDAS algorithms. In particular, while certain runs of our algorithms involve less accurate (reduced) linear system solutions, this may may lead to a significant increase in the number of PDAS iterations required before termination. With an effective preconditioner, one should still expect to attain reduced overall computational costs by allowing inexactness in the subproblem solutions, but this may require careful selection of algorithmic parameters such as ϵ_{res}.

6. Conclusion. In this paper, we have proposed a set of primal-dual active-set algorithms for solving certain structured quadratic optimization problems. The distinguishing feature of the algorithms is that they attain global convergence guarantees while allowing inexactness in the (reduced) linear system solves in each iteration. In each iteration, the first algorithm sets a requirement for the accuracy in the linear system solve through the use of subroutines for computing an upper bound for the inverse of a particular submatrix, whereas the second and third make use of dynamic algorithmic parameters for controlling the level of inexactness in each iteration. We have implemented our algorithms and have provided the results of numerical experiments on a pair of optimal control test problems, illustrating that our algorithms can converge in a similar number of PDAS iterations as an algorithm that employs "exact" linear system solves, but with much lower per-iteration computational costs.

REFERENCES

[1] M. AGANAGIĆ, *Newton's method for linear complementarity problems*, Mathematical Programming, 28 (1984), pp. 349–362.

[2] J. H. AHLBERG AND E. N. NILSON, *Convergence properties of the spline fit*, Journal of the Society for Industrial and Applied Mathematics, 11 (1963), pp. 95–104.

[3] R. A. BARTLETT AND L. T. BIEGLER, *QPSchur: A dual, active-set, Schur-complement method for large-scale and structured convex quadratic programming*, Optimization and Engineering, 7 (2006), pp. 5–32.

[4] M. BERGOUNIOUX, K. ITO, AND K. KUNISCH, *Primal-dual strategy for constrained optimal control problems*, SIAM Journal on Control and Optimization, 37 (1999), pp. 1176–1194.

[5] D. P. BERTSEKAS, *Projected Newton methods for optimization problems with simple constraints*, SIAM Journal on Control and Optimization, 20 (1982), pp. 221–246.

[6] M. J. BEST, *An algorithm for the solution of the parametric quadratic programming problem*, in Applied Mathematics and Parallel Computing, Physica-Verlag HD, Heidelberg, Germany, 1996, pp. 57–76.

[7] E. G. BIRGIN AND J. M. MARTÍNEZ, *Large-scale active-set box-constrained optimization method with spectral projected gradients*, Computational Optimization and Applications, 23 (2002), pp. 101–125.

[8] I. M. BOMZE, *On standard quadratic optimization problems*, Journal of Global Optimization, 13 (1998), pp. 369–387.

[9] B. E. Boser, I. M. Guyon, and V. N. Vapnik, *A training algorithm for optimal margin classifiers*, in Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92, New York, NY, USA, 1992, ACM Press, pp. 144–152.

[10] P. N. Brown and Y. Saad, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM Journal on Scientific and Statistical Computing, 11 (1990), pp. 450–481.

[11] S. Cafieri, M. D'Apuzzo, M. Marino, A. Mucherino, and G. Toraldo, *Interior-point solver for large-scale quadratic programming problems with bound constraints*, Journal of Optimization Theory and Applications, 129 (2006), pp. 55–75.

[12] F. E. Curtis, Z. Han, and D. P. Robinson, *A Globally Convergent Primal-Dual Active-Set Framework for Large-Scale Convex Quadratic Optimization*, Tech. Rep. 12T-013, COR@L Laboratory, Department of ISE, Lehigh University, 2012.

[13] Y.-H. Dai and R. Fletcher, *Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming*, Numerische Mathematik, 100 (2005), pp. 21–47.

[14] J. C. De los Reyes, *A primal-dual active set method for bilaterally control constrained optimal control of the Navier-Stokes equations*, Numerical Functional Analysis and Optimization, 25 (2005), pp. 657–683.

[15] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton Methods*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 400–408.

[16] H. J. Ferreau, H. G. Bock, and M. Diehl, *An online active set strategy to overcome the limitations of explicit MPC*, International Journal of Robust and Nonlinear Control, 18 (2008), pp. 816–830.

[17] R. Fletcher, *Practical Methods of Optimization*, Wiley-Interscience, New York, NY, USA, Second ed., 2000.

[18] A. Friedlander and J. M. Martínez, *On the maximization of a concave quadratic function with box constraints*, SIAM Journal on Optimization, 4 (1994), pp. 177–192.

[19] M. Fukushima and P. Tseng, *An implementable active-set algorithm for computing a B-stationary point of a mathematical program with linear complementarity constraints*, SIAM Journal on Optimization, 12 (2002), pp. 724–739.

[20] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Emerald Group Publishing Limited, Bingley, UK, 1982.

[21] D. Goldfarb and A. Idnani, *A numerically stable dual method for solving strictly convex quadratic programs*, Mathematical Programming, 27 (1983), pp. 1–33.

[22] N. I. M. Gould and P. L. Toint, *A quadratic programming bibliography*, Tech. Rep. RAL Numerical Analysis Group Internal Report 2000-1, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2000.

[23] R. Griesse and S. Volkwein, *A primal-dual active set strategy for optimal boundary control of a nonlinear reaction-diffusion system*, SIAM Journal on Control and Optimization, 44 (2005), pp. 467–494.

[24] W. W. Hager and H. Zhang, *A new active set algorithm for box constrained optimization*, SIAM Journal on Optimization, 17 (2006), pp. 526–557.

[25] M. Heinkenschloss, M. Ulbrich, and S. Ulbrich, *Superlinear and quadratic convergence of affine-scaling interior-point Newton methods for problems with simple bounds without strict complementarity assumption*, Mathematical Programming, 86 (1999), pp. 615–635.

[26] M. Hintermüller, K. Ito, and K. Kunisch, *The primal-dual active set strategy as a semismooth Newton method*, SIAM Journal on Optimization, 13 (2003), pp. 865–888.

[27] M. Hintermüller, V. A. Kovtunenko, and K. Kunisch, *Obstacle problems with cohesion: A hemivariational inequality approach and its efficient numerical solution*, SIAM Journal on Optimization, 21 (2011), pp. 491–516.

[28] S. Hüeber, A. Matei, and B. I. Wohlmuth, *Efficient algorithms for problems with friction*, SIAM Journal on Scientific Computing, 29 (2007), pp. 70–92.

[29] S. Hüeber, G. Stadler, and B. I. Wohlmuth, *A primal-dual active set algorithm for three-dimensional contact problems with Coulomb friction*, SIAM Journal on Scientific Computing, 30 (2008), pp. 572–596.

[30] K. Ito and K. Kunisch, *Optimal control of elliptic variational inequalities*, Applied Mathematics and Optimization, 41 (2000), pp. 343–364.

[31] ———, *The primal-dual active set method for nonlinear optimal control problems with bilateral constraints*, SIAM Journal on Control and Optimization, 43 (2004), pp. 357–376.

[32] K. Ito and K. Kunisch, *Convergence of the primal-dual active set strategy for diagonally dominant systems*, SIAM Journal on Control and Optimization, 46 (2007), pp. 14–34.

[33] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, *An interior-point method for large-scale ℓ_1-regularized least squares*, IEEE Journal of Selected Topics in Signal Processing, 1 (2007), pp. 606–617.

[34] D. Krishnan, P. Lin, and A. M. Yip, *A primal-dual active-set method for non-negativity constrained total variation deblurring problems*, IEEE Transactions on Image Processing, 16 (2007), pp. 2766–2777.
[35] K. Kunisch and F. Rendl, *An infeasible active set method for quadratic problems with simple bounds*, SIAM Journal on Optimization, 14 (2003), pp. 35–52.
[36] K. Kunisch and A. Rösch, *Primal-dual active set strategy for a general class of constrained optimal control problems*, SIAM Journal on Optimization, 13 (2002), pp. 321–334.
[37] P. Lötstedt, *Solving the minimal least squares problem subject to bounds on the variables*, BIT Numerical Mathematics, 24 (1984), pp. 205–224.
[38] J. J. Moré and G. Toraldo, *Algorithms for bound constrained quadratic programming problems*, Numerische Mathematik, 55 (1989), pp. 377–400.
[39] Y. Nesterov and A. Nemirovskii, *Interior point polynomial algorithms in convex programming*, Studies in Applied and Numerical Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1994.
[40] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, Springer, New York, NY, USA, Second ed., 2006.
[41] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM Journal on Numerical Analysis, 12 (1975), pp. 617–629.
[42] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, PA, USA, Second ed., 2003.
[43] G. Stadler, *Semismooth Newton and augmented Lagrangian methods for a simplified friction problem*, SIAM Journal on Optimization, 15 (2004), pp. 39–62.
[44] J. A. K. Suykens and J. Vandewalle, *Least squares support vector machine classifiers*, Neural Processing Letters, 9 (1999), pp. 293–300.
[45] H. A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, New York, NY, USA, 2003.
[46] R. J. Vanderbei, *LOQO : An interior point code for quadratic programming*, Optimization Methods and Software, 11 (1999), pp. 451–484.
[47] V. Vapnik and C. Cortes, *Support vector networks*, Machine Learning, 20 (1995), pp. 273–297.
[48] J. Varah, *A lower bound for the smallest singular value of a matrix*, Linear Algebra and its Applications, 11 (1975), pp. 3–5.
[49] R. S. Varga, *On diagonal dominance arguments for bounding* $\|A^{-1}\|_\infty$, Linear Algebra and its Applications, 14 (1976), pp. 211–217.
[50] S. J. Wright, *Primal-Dual Interior-Point Methods*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997.

Appendix A. Convergence of Algorithm 1.

We prove Theorem 2.3 by proving two theorems; the first corresponds to condition (a) of the theorem and the second corresponds to condition (b).

Theorem A.1. *Suppose Assumption 2.1 holds, R is a P-matrix, and, corresponding to any partition $(\mathcal{A},\mathcal{I})$, we have that $\|[R_{\mathcal{I}\mathcal{I}}^{-1}R_{\mathcal{I}\mathcal{A}}]_+\|_1 < 1$ and $e^T R_{\mathcal{I}\mathcal{I}}^{-1} w \geq 0$ for any $w \geq 0$, where the latter inequality holds strictly, i.e., $e^T R_{\mathcal{I}\mathcal{I}}^{-1} w > 0$, whenever $w \neq 0$. Then, with $\varepsilon_{opt} \geq 0$ and any initial partition, Algorithm 1 terminates in a finite number of iterations. In particular, if $\varepsilon_{opt} = 0$, then Algorithm 1 terminates in a finite number of iterations with a KKT point for (QP).*

Proof. It suffices to prove the result for $\varepsilon_{opt} = 0$. Thus, in the proof, we show that Algorithm 1 yields a KKT point in a finite number of iterations.

In order to derive a contradiction, suppose that Algorithm 1 generates an infinite number of partitions. For a given partition $(\mathcal{A},\mathcal{I})$ considered in the algorithm, let $(\mathcal{A}^+,\mathcal{I}^+)$ be the subsequent partition in the algorithm. Furthermore, let (x,y,z) and (x^+,y^+,z^+), respectively, be the subspace solutions corresponding to these partitions. For any index $i \in \mathcal{A}^+$, we have by Step 8 of Algorithm 1 that either

$$\begin{aligned} i \in \mathcal{A} &\implies x_i = u_i = x_i^+ \\ \text{or } \ i \in \mathcal{I} &\implies x_i > u_i = x_i^+. \end{aligned}$$

Hence, it follows that $[x^+ - x]_{\mathcal{A}^+} \leq 0$. Similarly, for any index $i \in \mathcal{I}^+$, we have by Step 8 of Algorithm 1 that either

$$\begin{aligned} i \in \mathcal{I} &\implies z_i = 0 = z_i^+ \\ \text{or } \; i \in \mathcal{A} &\implies z_i < 0 = z_i^+. \end{aligned}$$

Hence, it follows that $[z^+ - z]_{\mathcal{I}^+} \geq 0$. Now, since (x, y, z) and (x^+, y^+, z^+) are subspace solutions, it follows from (2.1) that

$$R[x^+ - x]_{\mathcal{N}} + [z^+ - z] = 0, \tag{A.1}$$

which implies that

$$[x^+ - x]_{\mathcal{I}^+} = -R^{-1}_{\mathcal{I}^+\mathcal{I}^+}(R_{\mathcal{I}^+\mathcal{A}^+}[x^+ - x]_{\mathcal{A}^+} + [z^+ - z]_{\mathcal{I}^+}).$$

Moreover, from nonsingularity of the matrix R, it follows that if $[x^+ - x]_{\mathcal{A}^+}$ and $[z^+ - z]_{\mathcal{I}^+}$ are both zero, then $[x^+ - x]_{\mathcal{I}^+}$ and $[z^+ - z]_{\mathcal{A}^+}$ are both zero. By the partition update rule in Step 8 of Algorithm 1, this occurs if and only if the violated sets $\mathcal{V}_P$ and $\mathcal{V}_D$ corresponding to (x, y, z) satisfy $\mathcal{V}_P \cup \mathcal{V}_D$, which, by Theorem 2.2, occurs if and only if (x, y, z) is a KKT point for (QP), i.e., $\mathrm{KKT}(x, y, z) = 0$. However, in such a case, the algorithm would have terminated with (x, y, z), contradicting the supposition that an infinite number of partitions are generated. Hence, it follows that $[x^+ - x]_{\mathcal{A}^+}$ and $[z^+ - z]_{\mathcal{I}^+}$ cannot both be zero.

For brevity, we now define $\Delta x := x^+ - x$ and $\Delta z := z^+ - z$. From the discussion above and letting $e \in \mathbb{R}^n$ denote a vector of ones, we have

$$\begin{aligned} & e^T \Delta x_{\mathcal{N}} \\ &= e^T_{\mathcal{A}^+}[\Delta x]_{\mathcal{A}^+} + e^T_{\mathcal{I}^+}[\Delta x]_{\mathcal{I}^+} \\ &= e^T_{\mathcal{A}^+}[\Delta x]_{\mathcal{A}^+} - e^T_{\mathcal{I}^+}R^{-1}_{\mathcal{I}^+\mathcal{I}^+}R_{\mathcal{I}^+\mathcal{A}^+}[\Delta x]_{\mathcal{A}^+} - e^T_{\mathcal{I}^+}R^{-1}_{\mathcal{I}^+\mathcal{I}^+}[\Delta z]_{\mathcal{I}^+} \\ &\leq \; -(1 - \|[R^{-1}_{\mathcal{I}^+\mathcal{I}^+}R_{\mathcal{I}^+\mathcal{A}^+}]_+\|_1)\|[\Delta x]_{\mathcal{A}^+}\|_1 - e^T_{\mathcal{I}^+}R^{-1}_{\mathcal{I}^+\mathcal{I}^+}[\Delta z]_{\mathcal{I}^+} < 0, \end{aligned}$$

where, by the conditions of the theorem, the last inequality is strict since $[\Delta x]_{\mathcal{A}^+}$ and $[\Delta z]_{\mathcal{I}^+}$ cannot both be zero. Thus, the quantity $e^T x_{\mathcal{N}}$ strictly decreases in each iteration of Algorithm 1. However, this is a contradiction to the supposition that an infinite number of iterates are generated since x is uniquely determined by the partition and there are only a finite number of partitions of $\mathcal{N}$. Consequently, we have proved that Algorithm 1 must terminate finitely with a KKT point. □

THEOREM A.2. *Suppose Assumption 2.1 holds and R satisfies $R = M + E$, where M is an M-matrix and $\|E\|_1$ is sufficiently small. Then, with $\varepsilon_{opt} \geq 0$ and any initial partition, Algorithm 1 terminates in a finite number of iterations. In particular, if $\varepsilon_{opt} = 0$, then Algorithm 1 terminates in a finite number of iterations with a KKT point for* (QP).

Proof. As in the proof of Theorem A.1, it suffices to prove the result for $\varepsilon_{opt} = 0$. Furthermore, again as in the proof of Theorem A.1, we suppose—in order to derive a contradiction—that Algorithm 1 generates an infinite number of partitions. Borrowing notation and conclusions from the proof of Theorem A.1, we have $[\Delta x]_{\mathcal{A}^+} \leq 0$ and $[\Delta z]_{\mathcal{I}^+} \geq 0$. Moreover, for sufficiently small $\|E\|_1$, the matrix R is nonsingular,

$$R^{-1}_{\mathcal{I}^+\mathcal{I}^+}R_{\mathcal{I}^+\mathcal{A}^+} = M^{-1}_{\mathcal{I}^+\mathcal{I}^+}M_{\mathcal{I}^+\mathcal{A}^+} + \mathcal{O}(\|E\|_1) \;\text{ and }\; R^{-1}_{\mathcal{I}^+\mathcal{I}^+} = M^{-1}_{\mathcal{I}^+\mathcal{I}^+} + \mathcal{O}(\|E\|_1).$$

Since M is an M-matrix, we have $M^{-1}_{\mathcal{I}^+\mathcal{I}^+}M_{\mathcal{I}^+\mathcal{A}^+} \leq 0$ and $M^{-1}_{\mathcal{I}^+\mathcal{I}^+} \geq 0$. Hence, since $[\Delta x]_{\mathcal{A}^+}$ and $[\Delta z]_{\mathcal{I}^+}$ cannot both be zero, we have for sufficiently small $\|E\|_1$ that

$$\begin{aligned}
&e^T \Delta x_{\mathcal{N}} \\
&= e^T_{\mathcal{A}^+}[\Delta x]_{\mathcal{A}^+} + e^T_{\mathcal{I}^+}[\Delta x]_{\mathcal{I}^+} \\
&= e^T_{\mathcal{A}^+}[\Delta x]_{\mathcal{A}^+} - e^T_{\mathcal{I}^+}R^{-1}_{\mathcal{I}^+\mathcal{I}^+}R_{\mathcal{I}^+\mathcal{A}^+}[\Delta x]_{\mathcal{A}^+} - e^T_{\mathcal{I}^+}R^{-1}_{\mathcal{I}^+\mathcal{I}^+}[\Delta z]_{\mathcal{I}^+} \\
&= e^T_{\mathcal{A}^+}[\Delta x]_{\mathcal{A}^+} - e^T_{\mathcal{I}^+}M^{-1}_{\mathcal{I}^+\mathcal{I}^+}M_{\mathcal{I}^+\mathcal{A}^+}[\Delta x]_{\mathcal{A}^+} - e^T_{\mathcal{I}^+}M^{-1}_{\mathcal{I}^+\mathcal{I}^+}[\Delta z]_{\mathcal{I}^+} + \mathcal{O}(\|E\|_1) < 0.
\end{aligned}$$

The remainder of the proof follows in the same manner as that of Theorem A.1. □