A Unifying Complexity Certification Framework for Active-Set Methods for Convex Quadratic Programming

Daniel Arnström and Daniel Axehill

Arnström, D., Axehill, D., (2022), A Unifying Complexity Certification Framework for Active-Set Methods for Convex Quadratic Programming, *IEEE Transactions on Automatic Control*, 67(6), 2758-2770. https://doi.org/10.1109/TAC.2021.3090749

A Unifying Complexity Certification Framework for Active-Set Methods for Convex Quadratic Programming

Daniel Arnström and Daniel Axehill

***Abstract*—In model predictive control (MPC) an optimization problem has to be solved at each time step, which in real-time applications makes it important to solve these efficiently and to have good upper bounds on worst-case solution time. Often for linear MPC problems, the optimization problem in question is a quadratic program (QP) that depends on parameters such as system states and reference signals. A popular class of methods for solving such QPs is active-set methods, where a sequence of linear systems of equations is solved. We propose an algorithm for computing which sequence of subproblems an active-set algorithm will solve, for every parameter of interest. These sequences can be used to set worst-case bounds on how many iterations, floating-point operations and, ultimately, the maximum solution time, the active-set algorithm requires to converge. The usefulness of the proposed method is illustrated on a set of QPs originating from MPC problems, by computing the exact worst-case number of iterations primal and dual active-set algorithms require to reach optimality.**

***Index Terms*—Quadratic Programming, Optimization algorithms, Predictive control for linear systems**

I. Introduction

In model predictive control (MPC) an optimization problem has to be solved at each time step, which for linear MPC often is a quadratic program (QP) which depends on parameters such as system states and reference signals, making it a multi-parametric QP (mpQP). Often, these mpQPs are solved offline parametrically for a set of parameters and the pre-computed solution is then used online [1]. However, the pre-computed solution grows exponentially in complexity with the dimensions of the problem and, for high-dimensional problems, limited memory can restrict the use of a pre-computed solution online. For such problems, the QP has to be solved online and the limited time and computational resources often at hand in real-time MPC require the employed QP solver to be efficient and to have guarantees on the time needed to solve the QPs within a given tolerance.

Popular methods for solving QPs encountered in MPC are active-set methods [2]–[6], interior-point methods [7], [8] and gradient projection methods [9]–[11]. Active-set methods easily integrate warm-starting of the solver, i.e., the use of a previous solution to start the solver in the next iteration, which often reduces the number of iterations needed by the solver [12], [13]. A well-known drawback of active-set methods is, however, that the complexity can be exponential in the worst-case [14], although, polynomial complexity is often observed in practice [15]. In contrast to active-set methods, theoretical polynomial bounds on the computational complexity of some interior-point and gradient projection methods have been proven in, e.g., [7], [9], [16], [17].

To close the gap between the possible exponential complexity and the often experienced polynomial complexity, methods for determining the exact complexity of the active-set QP methods presented in [2], [3] and [4] have been proposed in [18], [19] and [20], respectively. Similarly, a method for determining the complexity of a primal active-set method for linear programs (LPs) has been proposed in [12]. This paper extends the result in [18], which handles the strictly convex case, to also handle positive semi-definite mpQPs, leading to additional theoretical as well as numerical results. In addition to being able to certify the complexity of primal active-set methods applied to positive semi-definite mpQPs, we show that this extension allows for certification of dual active-set QP methods and active-set LP methods, enabling the results in [18], [19] and [12] to be viewed in a unified framework.

The main contribution is, hence, a method for analyzing *exactly* which subproblems a primal active-set algorithm will solve in order to compute an optimal solution, for any set of parameters in a positive *semi-definite* mpQP. The knowledge of these subproblems can be used to determine worst-case bounds on number of iterations, floating-point operations and, ultimately, solution time. These worst-case bounds can, hence, be determined *before* the active-set algorithm is employed online by using the proposed method offline on a given mpQP. Furthermore, exact knowledge about the encountered subproblems can be used to reduce the online computational cost by tailoring the solver for the mpQP at hand.

A challenging aspect of the analysis of the primal active-set QP algorithm considered in this work is that all iterates are not necessarily affine in the parameter, in contrast to the methods studied in [19], [20] and [12]. Nonaffine iterates are shown to lead to a partition of the parameter space consisting of both linear and quadratic inequalities, in contrast to only linear inequalities which is the case in [19], [20] and [12]. We also show how these quadratic inequalities can be dealt with and, in some cases, circumvented.

II. Preliminaries

It is well-known (see, e.g., [1]) that a linear MPC problem can be cast into an mpQP on the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2}x^T H x + (f^T + \theta^T f_\theta^T)x \\ \text{subject to} \quad & Ax \leq b + W\theta, \end{aligned} \tag{1}$$

This work was supported by the Swedish Research Council (VR) under contract number 2017-04710.

D. Arnström and D. Axehill are with the Division of Automatic Control, Linköping University, Sweden daniel.{arnstrom,axehill}@liu.se

where the iterate $x \in \mathbb{R}^n$ is related to the control action and the parameter $\theta \in \Theta_0 \subseteq \mathbb{R}^p$ is related to the state of the plant. The feasible set of the problem is defined by $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $W \in \mathbb{R}^{m \times p}$ and the objective function is defined by $f \in \mathbb{R}^n$, $f_\theta \in \mathbb{R}^{n \times p}$, and $H \in \mathbb{S}^n_+$. For convenience, we also introduce the compact notation $b(\theta) = b + W\theta$ and $f(\theta) = f + f_\theta \theta$ to clean up some expressions.

Another way of expressing the feasible set is in terms of each constraint as $[A]_i x \leq [b]_i + [W]_i \theta, i \in \mathcal{K}$, where the notation $[\cdot]_i$ extracts the i:th row and $\mathcal{K} \triangleq \{1, 2, ..., m\}$. Finally, a constraint holding with equality is said to be *active*.

The active-set algorithm considered is an iterative algorithm which searches for the active constraints at the optimum, motivating the following notation. The iterate at iteration k is denoted x_k and $\mathcal{W}_k$ denotes a subset of the constraints, called the *working set*, that are active at x_k. Moreover, we define A_k, b_k and W_k to denote the rows of the matrices indexed by $\mathcal{W}_k$ and we denote the complement of $\mathcal{W}_k$ as $\bar{\mathcal{W}}_k \triangleq \mathcal{K} \setminus \mathcal{W}_k$.

A. Equality constrained mpQP

The active-set algorithm considered in this paper solves a sequence of equality constrained QPs (EQPs) on the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2} x^T H x + f(\theta)^T x \\ \text{subject to} \quad & A_k x = b_k(\theta). \end{aligned} \tag{2}$$

The optimizer x_k^* of this subproblem, which we will call a *constrained stationary point* (CSP), and the dual variable λ_k can be obtained by solving the following linear system of equations, also known as a KKT system,

$$\begin{pmatrix} H & A_k^T \\ A_k & 0 \end{pmatrix} \begin{pmatrix} x_k^* \\ \lambda_k \end{pmatrix} = \begin{pmatrix} -f(\theta) \\ b_k(\theta) \end{pmatrix}. \tag{3}$$

If there exists a unique solution to (3) the inverse of the KKT matrix can be partitioned as

$$\begin{pmatrix} H & A_k^T \\ A_k & 0 \end{pmatrix}^{-1} = \begin{pmatrix} H_k^* & T_k \\ T_k^T & U_k \end{pmatrix} \tag{4}$$

and the solution to (3) is, hence, given by

$$x_k^* = -H_k^* f(\theta) + T_k b_k(\theta), \;\; \lambda_k = -T_k^T f(\theta) + U_k b_k(\theta). \tag{5}$$

The inverse in (4) exists iff the so-called *reduced Hessian* $\tilde{H}_k \triangleq Z_k^T H Z_k$ is nonsingular (see, e.g. Theorem 16.2 in [2]), where Z_k is a matrix with columns forming a basis for the null space of A_k. Furthermore, if the reduced Hessian is nonsingular H_k^*, T_k and U_k are given by

$$\begin{aligned} H_k^* &= Z_k (Z_k^T H Z_k)^{-1} Z_k^T, \\ T_k &= Y_k - Z_k (Z_k^T H Z_k)^{-1} Z_k^T H Y_k, \\ U_k &= Y_k^T H Z_k (Z_k^T H Z_k)^{-1} Z_k^T H Y_k - Y_k^T H Y_k, \end{aligned} \tag{6}$$

where Y_k is a matrix with columns spanning the range space of A_k and satisfying $Y_k^T A_k = I$ [21].

A well-known property of the solution in (5), which is exploited in explicit MPC [1], is that it is affine in θ

$$x_k^* = F_k^* \theta + G_k^*, \quad \lambda_k = F_k^\lambda \theta + G_k^\lambda, \tag{7}$$

where $F_k^*, G_k^*, F_k^\lambda$, and G_k^λ are given by

$$F_k^* \triangleq -H_k^* f_\theta + T_k W_k, \qquad G_k^* \triangleq -H_k^* f + T_k b_k, \tag{8a}$$

$$F_k^\lambda \triangleq -T_k^T f_\theta + U_k W_k, \qquad G_k^\lambda \triangleq -T_k^T f + U_k b_k. \tag{8b}$$

B. A primal active-set algorithm

There are plenty of different primal active-set methods in the litterature, e.g., [2][22][23], and numerous of these are mathematically equivalent [24], in the sense that they produce the same iterates given the same starting conditions, but differ numerically. In this paper we consider the primal active-set algorithm given by Algorithm 1, described in detail below. This algorithm formulation is chosen to make the certification method, described in Section IV, more succinct and the definition of an iteration of the algorithm sound, which is important for the certification. However, it would be possible to instead consider any other equivalent formulation, such as any of the primal active-set methods cited above. For example, this is done in [18] where the algorithm formulation presented in [2, Sec.16.5] is considered. The algorithm in [2] is, however, limited to strictly convex QPs, while Algorithm 1 also works for $H \succeq 0$.

Algorithm 1 Primal Active-Set Method for QP

Input: $x_0, \mathcal{W}_0, \theta, k = 1$, dual tolerance $\epsilon_d \geq 0$
Output: $x_k^*, \lambda_k, \mathcal{W}_k$

$s_0 \leftarrow b + W\theta - Ax_0$
while true **do**
  if (2) is unbounded **then** SINGULARITERATION
  else
    Compute x_k^* by solving (3)
    if $s_k^* \geq 0$ **then**
      Compute λ_k by solving (11)
      if $\lambda_k \geq -\epsilon_d$ **then return** $x_k^*, \lambda_k, \mathcal{W}_k$
      else $l \leftarrow \underset{i \in \mathcal{W}_k}{\text{argmin}} \; [\lambda_k]_i$; $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{l\}$
        $x_{k+1} \leftarrow x_k^*$; $s_{k+1} \leftarrow s_k^*$
    else $p_k \leftarrow x_k^* - x_k$; $[\sigma_k]_{\bar{\mathcal{W}}_k} \leftarrow [A]_{\bar{\mathcal{W}}_k} p_k$
      $m \leftarrow \underset{i \in \bar{\mathcal{W}}_k : [s_k^*]_i < 0}{\text{argmin}} \frac{[s_k]_i}{[\sigma_k]_i}$; $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \cup \{m\}$
      $x_{k+1} \leftarrow x_k + \alpha_k^m p_k$; $s_{k+1} \leftarrow s_k - \alpha_k^m \sigma_k$
    $k \leftarrow k + 1$

procedure SINGULARITERATION
  Compute $\tilde{p}_k$ from (12)
  $[\tilde{\sigma}_k]_{\bar{\mathcal{W}}_k} \leftarrow [A]_{\bar{\mathcal{W}}_k} \tilde{p}_k$
  if $\tilde{\sigma}_k \geq 0$ **then**
    break unbounded
  else $m \leftarrow \underset{i \in \bar{\mathcal{W}}_k : [\tilde{\sigma}_k]_i < 0}{\text{argmin}} \frac{[s_k]_i}{[\tilde{\sigma}_k]_i}$; $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \cup \{m\}$
    $x_{k+1} \leftarrow x_k + \alpha_k^m \tilde{p}_k$; $s_{k+1} \leftarrow s_k - \alpha_k^m \tilde{\sigma}_k$
  $k \leftarrow k + 1$

Algorithm 1 starts with a feasible point x_0 and a corresponding working set $\mathcal{W}_0$, containing a subset of the constraints that are active at x_0. For the later analysis, we allow x_0 to be affine in the parameter θ, i.e., $x_0 = F_0 \theta + G_0$.

In an iteration of the algorithm, constraints are added to or removed from the working set $\mathcal{W}_k$ while maintaining primal feasibility and updating the iterate x_k. The update of the iterate

is different depending on if the EQP defined by the current working set $\mathcal{W}_k$ is unbounded or not.

1) Bounded EQP subproblem: If $\tilde{H}_k \succ 0$, the EQP in (2) (defined by $\mathcal{W}_k$) has a unique solution and the search direction p_k is defined as the Newton step direction, given by $p_k \triangleq x_k^* - x_k$, where x_k^* is the solution to (2). In an iteration, primal feasibility is retained while attempting to move along the line segment from x_k to x_k^*. Because of the convexity of $Ax \leq b$, primal feasibility can be retained if x_k^* is primal feasible, i.e., if $b(\theta) - Ax_k^* \geq 0$. Let the primal slack of x_k^* be denoted

$$s_k^* \triangleq b(\theta) - Ax_k^* = b(\theta) - Ax_k - Ap_k = s_k - \sigma_k, \quad (9)$$

where we in the last equality have defined $s_k \triangleq b(\theta) - Ax_k$ (the primal slack of the current iterate) and $\sigma_k \triangleq Ap_k$ (how much the step p_k affects the primal slack). With this notation, x_k^* being primal feasible $\Leftrightarrow s_k^* \geq 0$.

If x_k^* is primal infeasible, i.e., if $s_k^* \ngeq 0$, there will be at least one hyper-plane corresponding to an inactive constraint that separates x_k and x_k^*. The move from x_k to x_k^* cannot, hence, be completed without breaking feasibility. Instead, a step is taken in the direction of p_k until the first blocking constraint $m \in \bar{\mathcal{W}}_k$ is encountered. The maximal step length α_k that retains feasibility is explicitly given as

$$\alpha_k = \min_{i \in \bar{\mathcal{W}}_k : [s_k^*]_i < 0} \alpha_k^i, \quad \alpha_k^i \triangleq \frac{[s_k]_i}{[\sigma_k]_i} = \frac{[b]_i + [W]_i\theta - [A]_i x_k}{[A]_i p_k}, \quad (10)$$

where α_k^j can be seen as a measure of the distance from x_k to the hyper-plane $[A]_j x = [b(\theta)]_j$ in the direction of p_k.

In addition to updating the iterate, $x_{k+1} = x_k + \alpha_k p_k$, the working set is updated by adding the first blocking constraint, i.e., the minimizing index of (10). Concretely, if m is the minimizing index in (10), $\mathcal{W}_{k+1} = \mathcal{W}_k \cup \{m\}$.

If x_k^* is feasible, i.e., if $s_k^* \geq 0$, global optimality for x_k^* is checked by examining the dual variables λ_k. x_k^* will be a global optimum if λ_k is dual feasible, i.e., if $[\lambda_k]_i \geq -\epsilon_d$, $\forall i \in \mathcal{W}_k$, where ϵ_d is the dual feasibility tolerance. From the first row in (3), λ_k can be obtained by solving

$$A_k^T \lambda_k = -(Hx_k^* + f(\theta)). \quad (11)$$

If the $\lambda_k \ngeq -\epsilon_d$, a constraint corresponding to the most negative dual variable $[\lambda_k]_l$ is removed from the working set, resulting in $\mathcal{W}_{k+1} = \mathcal{W}_k \setminus \{l\}$.

After the working set has been updated, a new search direction is computed by solving (3) with the new working set and the algorithm reiterates the steps described above until global optimality is ensured.

Remark 1: A straightforward way for terminating the algorithm earlier is to increase ϵ_d, which is considered in [25].

2) Unbounded EQP subproblem: If the EQP in (2) is unbounded at iteration k, which is equivalent to $\tilde{H}_k \nsucc 0$, the Newton direction is not well-defined and the search direction must, hence, be determined in another way. Instead of taking a step in the Newton direction, a step is taken in an unbounded direction of (2). Such a direction $\tilde{p}_k$ satisfies

$$H\tilde{p}_k = 0, \quad A_k\tilde{p}_k = 0, \quad (Hx_k + f(\theta))^T \tilde{p}_k < 0, \quad (12)$$

where the condition $(Hx_k + f(\theta))^T \tilde{p}_k < 0$ ensures that $\tilde{p}_k$ is a descent direction.

When deciding the step length along the unbounded direction, two different scenarios can occur. If there is a blocking constraint along the ray $x_k + \alpha\tilde{p}_k, \alpha > 0$, the first blocking constraint can be added to $\mathcal{W}_k$, similar to the line search in the bounded case, and the iterations can proceed as usual. Otherwise, if there are no blocking constraints along the ray, the objective function can be decreased by an arbitrary amount by moving along the descent direction $\tilde{p}_k$, resulting in an unbounded problem. Concretely, there will be no blocking constraint if $\tilde{\sigma}_k \triangleq A\tilde{p}_k \geq 0$ since then the updated slack $s_{k+1} = s_k + \alpha\tilde{\sigma}_k$ cannot become negative for any positive step length α, i.e., any positive α gives a primal feasible iterate.

III. Properties of primal active-set algorithms

To certify the complexity of Algorithm 1, we will make use of some inherent properties of its search directions and iterates. These properties are derived in this section and will later be used in Section IV, where the proposed complexity certification method is presented.

We consider properties of search directions and iterates after constraints are added to $\mathcal{W}$ (Section III-A), removed from $\mathcal{W}$ (Section III-B), and when the reduced Hessian is singular (Section III-C).

Remark 2: Since some of the properties are similar to the ones used in [18], where $H \succ 0$ was assumed, the reader is referred to [18] for some of the proofs and only brief comments about how to amend these to also handle the semi-definite case are given.

A. Addition of a constraint to $\mathcal{W}$

When a constraint is added to $\mathcal{W}$ there will be a relationship between the subsequent and previous search direction in terms of H_k^* (recall that H_k^* was defined in (6)).

Property 1: If a constraint is added to $\mathcal{W}$ in iteration k, $p_{k+1} = (1 - \alpha_k)H_k^* H p_k$

Proof: Cf. Lemma 1 in [18] and replace $\tilde{\pi}_k$ with $H_k^* H$. ∎

A lot of important properties of the search directions follow from Property 1 and the following projective property of H_k^*.

Lemma 1: $\mathcal{W}_k \subseteq \mathcal{W}_{k+1} \implies H_{k+1}^* H H_k^* = H_{k+1}^*$.

Proof: If $\mathcal{W}_k \subseteq \mathcal{W}_{k+1}$, a null space basis for A_{k+1} is a subset of a null space basis for A_k, i.e., $Z_k = [Z_{k+1},\ Z_+]$. Using the inversion formula for 2×2 block matrices then gives

$$\begin{aligned}(Z_k^T H Z_k)^{-1} &= \begin{bmatrix} Z_{k+1}^T H Z_{k+1} & Z_{k+1}^T H Z_+ \\ Z_+^T H Z_{k+1} & Z_+^T H Z_+ \end{bmatrix}^{-1} \triangleq \begin{bmatrix} U & V \\ V^T & W \end{bmatrix}^{-1} \\ &= \begin{bmatrix} U^{-1}(I + V\tilde{S}V^T U^{-1}) & -U^{-1}V\tilde{S} \\ -\tilde{S}V^T U^{-1} & \tilde{S} \end{bmatrix}, \end{aligned} \quad (13)$$

with $\tilde{S} \triangleq (W - V^T U V)^{-1}$ being the inverse of a Schur complement. Multiplication with Z_k^T from the right then gives

$$(Z_k^T H Z_k)^{-1} Z_k^T = \begin{bmatrix} U^{-1}\left(Z_{k+1}^T + V\tilde{S}\tilde{Z}\right) \\ -\tilde{S}\tilde{Z} \end{bmatrix}, \quad (14)$$

with $\tilde{Z} \triangleq (V^T U^{-1} Z_{k+1}^T - Z_+)$. By definition we have that $Z_{k+1}^T H Z_k = [U,\ V]$ and multiplying (14) from the left with

this expression and recalling the definition of H_k^* from (6) gives, after some cancellations,

$$Z_{k+1}^T H H_k^* = Z_{k+1}^T H Z_k (Z_k^T H Z_k)^{-1} Z_k^T = Z_{k+1}^T. \quad (15)$$

Finally, we get the desired result by recalling the definition of H_{k+1}^* from (6) together with (15), resulting in $H_{k+1}^* H H_k^* = Z_{k+1}^T (Z_{k+1}^T H Z_{k+1})^{-1} Z_{k+1}^T H H_k^* = Z_{k+1}^T (Z_{k+1}^T H Z_{k+1})^{-1} Z_{k+1}^T = H_{k+1}^*$. ∎

Remark 3: When $H \succ 0$, $H_k^* H$ will be equivalent to the operator $\tilde{\pi}_k$ used in [18].

By using the projective property of H_k^*, Property 1 can be used iteratively for a relationship between search directions when constraints are added to $\mathcal{W}$ in consecutive iterations.

Property 2: If constraints are added to $\mathcal{W}$ from iteration k until iteration $k+N$, $p_{k+N} = (1-\tau(\theta)) H_{k+N}^* H p_k$ for some $\tau(\theta) \in [0, 1)$.

Proof: Cf. Corollary 1 in [18] and replace $\tilde{\pi}_k$ with $H_k^* H$ and iteratively apply Lemma 1. ∎

If only additions to $\mathcal{W}$ have been made since the start of Algorithm 1 up until iteration k, Property 2 can be used to get an explicit expression of x_k in terms of x_0 and p_0.

Property 3: If constraints are added to $\mathcal{W}$ from iteration 0 until iteration k, $x_k = H_k^* H (x_0 + \tau p_0) + T_k b(\theta)$

Proof: Cf. Corollary 2 in [18] and replace $\tilde{\pi}_k$ with $H_k^* H$ and $\pi_k(\cdot)$ with $-H_k^* H(\cdot) + T_k b_k(\theta)$. ∎

B. Removal of a constraint from $\mathcal{W}$

When a constraint is removed there will be a relationship between the subsequent search direction and the normal of the removed half-plane.

Property 4: If constraint l is removed from $\mathcal{W}_k$ in iteration k and $\mathcal{W}_k \setminus \{l\}$ leads to the EQP in (2) being bounded, $p_{k+1} = -[\lambda_k]_l H_{k+1}^* [A]_l^T$.

Proof: Cf. Lemma 2 in [18] and replace $\tilde{\pi}_k$ with $H_k^* H$. ∎

Property 4 together with Property 2 give the following, fundamental, property of the search directions computed by Algorithm 1.

Property 5: At iteration $k+N$, let l be the index of the latest removed constraint from $\mathcal{W}$, removed in iteration k. Then $p_{k+N} = -(1-\tau(\theta))[\lambda_k]_l H_{k+N}^* [A]_l^T$ for some $\tau(\theta) \in [0, 1]$.

Proof: The property follows by combining Property 2 and Property 4 together with Lemma 1. ∎

Since the direction of the step in Property 5 only depends on H_{k+N}^* and the normal $[A]_l^T$ of the latest removed constraint, the parameter θ does not affect the *direction* of the step after a constraint has been removed, only its magnitude.

Not only the search directions, but also the iterates have additional structure after a constraint has been removed.

Property 6: If a constraint is removed in iteration κ, $x_k = F_k \theta + G_k$, $\forall k > \kappa$ for some $F_k \in \mathbb{R}^{n \times p}, G_k \in \mathbb{R}^n$.

Proof: Cf. Lemma 4 in [18] and use Property 5. ∎

Explicitly, if the normal of the latest removed constraint is denoted $\hat{p}$ and we add constraint j to $\mathcal{W}$ in iteration k, $x_{k+1} = (F_k + \Delta F_k)\theta + (G_k + \Delta G_k)$ with

$$\Delta F_k = H_k^* \hat{p} \frac{[W]_j - [A]_j F_k}{[A]_j H_k^* \hat{p}}, \quad \Delta G_k = \frac{[b]_j - [A]_j G_k}{[A]_j H_k^* \hat{p}} H_k^* \hat{p}. \quad (16)$$

C. Parameter independence of singular search directions

Another property which will be important in Section IV is that the search direction when $\tilde{H}$ is singular can be determined independently of the parameter θ.

Property 7: Assume that constraint l was removed from $\mathcal{W}_{k-1}$ at iteration $k-1$ and let $\tilde{p}_k$ be a solution to (12). Then $(Hx_k + f(\theta))^T \tilde{p}_k < 0 \Leftrightarrow [A]_l^T \tilde{p}_k < 0$.

Proof: Since a constraint was removed in the previous iteration it holds that $x_k = x_{k-1}^*$, which satisfies $H x_{k-1}^* + A_{k-1}^T \lambda_{k-1} = -f(\theta)$. Hence, $H x_{k-1}^* + f(\theta) = -A_{k-1}^T \lambda_{k-1}$ and, moreover, $(Hx_k + f(\theta))^T \tilde{p}_k = -\lambda_{k-1}^T A_{k-1} \tilde{p}_k = -[\lambda_{k-1}(\theta)]_l [A]_l^T \tilde{p}_k$, where the last equality follows from $A_k = [A_{k-1}^T a_l^T]^T$ since l was removed in iteration $k-1$ and from $A_k \tilde{p}_k = 0$ since $\tilde{p}_k$ is a solution to (12).

Taken together, $(Hx_k + f(\theta))^T \tilde{p}_k = -[\lambda_{k-1}]_l [A]_l^T \tilde{p}_k < 0$ and, since constraint l was removed in the previous iteration, $[\lambda_{k-1}]_l < 0$, resulting in $[A]_l^T \tilde{p}_k < 0$. ∎

From (12) and Property 7 the singular direction $\tilde{p}_k$ is, hence, a solution to $H\tilde{p}_k = 0, A_k \tilde{p}_k = 0, [A]_l^T \tilde{p}_k < 0$, which is independent of the parameter θ.

IV. Complexity Certification

In this section we propose a method which exactly identifies which working-set sequence $\{\mathcal{W}_k(\theta)\}_{k=0}^{\bar{k}(\theta)}$ different parameters θ will generate when Algorithm 1 is applied to the mpQP in (1). As a by-product the method, hence, also indentifies the number of iterations $\bar{k}(\theta)$ Algorithm 1 will perform as a function of θ. The method is an extension of [18] and similar to the ones presented in [19], [20], and [12], in the sense that the parameter space is iteratively partitioned depending on how the working set changes in each iteration. In fact, as will be described in Section VI, the certification methods covered in [18], [19] and [12] can all be seen as special cases of the proposed method.

A. Overview

The main idea behind the complexity certification method is to track how the working set $\mathcal{W}$ changes parametrically in terms of θ. An interpretation of this is running Algorithm 1 parametrically and partitioning Θ_0 into regions containing parameters which yield the same working-set changes. To do this partitioning, Algorithm 1 is decomposed into three distinct modes (illustrated in Figure 1):

a) Checking for global optimality and removing constraints, performed at Steps 7-10.
b) Checking for local optimality and adding constraints, performed at Steps 5-6 and 11-13.
c) Checking for unboundedness and adding constraints, performed at Steps 16-21.

A switch from mode a) → b) or c) occurs when a constraint is removed, depending on if the new reduced Hessian is positive definite or not, respectively. Likewise, the algorithm goes from mode b) → a) when a constrained stationary point is primal feasible. Finally, a switch from mode c) → b) occurs after a constraint is added in mode c).

In each mode, a parameter region is partitioned into finer regions, where all parameters contained in a certain region

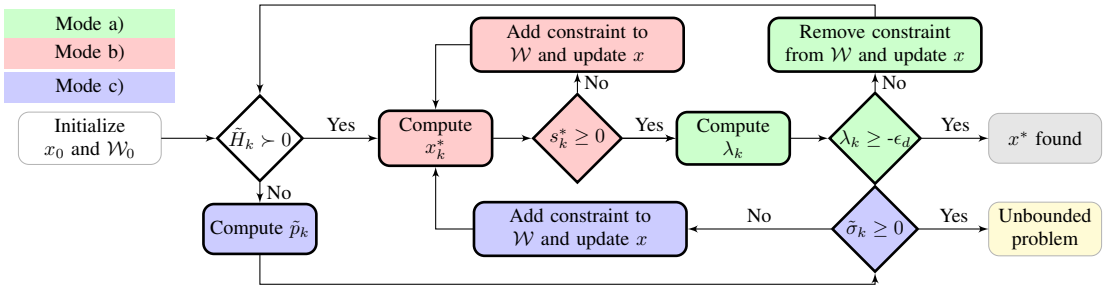


Fig. 1: Flowchart characterizing Algorithm 1.

indicate that they generate the same change to the working set. For each finer region, the partitioning is repeated recursively, corresponding to executing iterations of Algorithm 1 parametrically, leading to an increasingly finer partitioning. Explicitly, the partitioning done in each mode are:

In mode a) a region Θ is partitioned into the regions

- Θ^*: Global optimality obtained
- Θ^{-j}: j removed from $\mathcal{W}$

Likewise, in mode b) a region Θ is partitioned into the regions

- Θ^{CSP}: A primal feasible CSP reached
- Θ^{+j}: j added to $\mathcal{W}$

Finally, in mode c) a region Θ is partitioned into the regions

- Θ^{∞}: The problem is unbounded
- Θ^{+j}: j added to $\mathcal{W}$

Each new region will enter a new mode for further partitioning or be marked as optimal or unbounded. If marked optimal or unbounded, the region will not be partitioned further. The partitioning made in each mode is illustrated in Figure 2 and will be discussed in more detail in the upcoming sections.

In summary, Θ_0 will iteratively be partitioned into the above-mentioned subsets until all parameters have reached global optimality or have been identified to result in an unbounded problem. In the final partition, parameters in the same region signify that they produce the same sequence of working-set changes before terminating. The method is summarized in Algorithm 2, where each region of the partition is represented by a tuple $(\Theta, \mathcal{W}, F, G, s, k, \hat{p})$ containing the following data:

- $\Theta \subseteq \Theta_0 \subseteq \mathbb{R}^p$ - The subset of the parameter space that defines the region.
- $\mathcal{W}$ - The working set in the region.
- $F \in \mathbb{R}^{n \times p}$ and $G \in \mathbb{R}^{n \times 1}$ - Matrices that define the affine mapping $x_k = F\theta + G$ for $\theta \in \Theta$.
- s - A status flag that marks if the region has reached a CSP 'CSP', global optimality 'OPT', results in an unbounded problem 'UNB', removed a constraint in last iteration 'RM' or neither 'NUL'.
- k - Number of iterations performed by Algorithm 1 to reach the current state.
- $\hat{p}$ - The normal of the latest removed constraint.

Furthermore, S in Algorithm 2 is a stack containing tuples corresponding to regions of Θ_0 that are yet to terminate.

The subroutines ModeA, ModeB and ModeC in Algorithm 2 partition the parameter space depending on what happens in mode a), mode b), or mode c), respectively. These subroutines will now be described in detail in Section IV-B, IV-C and IV-D, respectively.

Algorithm 2 Partition Θ_0 based on working-set changes

Input: $\Theta_0, \mathcal{W}_0, F_0, G_0$, mpQP
Output: FinalPartition, UnboundPartition

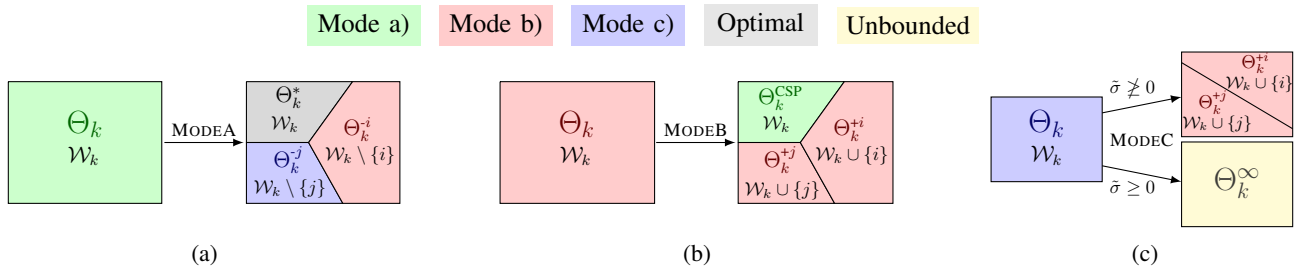
```
Push (Θ_0, W_0, F_0, G_0, 'NUL', 0, NaN) to S
while S is not empty do
    Pop p_c from S
    if reduced Hessian for p_c is positive definite then
        if p_c has reached a CSP then
            Partition = ModeA(p_c, mpQP)
        else Partition = ModeB(p_c, mpQP)
    else
        Partition = ModeC(p_c, mpQP)
    for p in Partition do
        if p is global optimum then
            Append p to FinalPartition
        else if p is unbounded then
            Append p to UnboundPartition
        else Push p to S
return FinalPartition, UnboundPartition
```

B. Removing constraints and checking for global optimality

First we consider how the parameter space is partitioned in mode a). If mode a) is entered at iteration k, the dual variable λ_k determines whether global optimality has been reached or if a constraint has to be removed. Recall from Algorithm 1 that a global optimum has been found if all components of $\lambda_k(\theta)$ are nonnegative (within a given tolerence ϵ_d). Otherwise, a constraint l corresponding to a negative component of λ_k is removed from the working set. From Algorithm 1 Step 9, l is chosen as the most negative component of $\lambda_k(\theta)$, i.e., $l = \text{argmin}_{i \in \mathcal{W}_k} [\lambda_k(\theta)]_i$ (Dantzig's selection rule). The set Θ_k^{-j} of all parameters in iteration k resulting in constraint $j \in \mathcal{W}_k$ being removed from the working set is, hence, given by

$$\Theta_k^{-j} = \left\{ \theta \in \Theta_k \middle| [\lambda_k(\theta)]_j < \begin{pmatrix} -\epsilon_d \\ [\lambda_k(\theta)]_i \end{pmatrix}, \forall i \in \mathcal{W}_k \setminus \{j\} \right\}, \tag{17}$$

i.e., θ for which the dual variable corresponding to constraint j is negative and more negative than any other dual variable.



(a) (b) (c)

Fig. 2: Conceptual illustrations of the partitioning of a parameter region Θ_k done in mode a), b), and c). In mode a), removing j results in a singular reduced Hessian while removing i does not.

Likewise, the set Θ_k^* of all parameters in iteration k resulting in a global optimum is given by

$$\Theta_k^* = \{\theta \in \Theta_k | [\lambda_k(\theta)]_i \geq \text{-}\epsilon_d, \ \forall i \in \mathcal{W}_k\}, \tag{18}$$

i.e., θ for which all dual variables are nonnegative.

In summary, a region Θ_k in mode a) will be partitioned into Θ_k^* and $\Theta_k^{-i}, \forall i \in \mathcal{W}_k$, illustrated in Figure 2a.

To get more explicit expressions of these sets, recall from (7) that $\lambda_k(\theta)$ is affine in θ, i.e., $\lambda_k(\theta) = F_k^\lambda \theta + G_k^\lambda$. Hence, the regions Θ_k^{-j} in (17) can be equivalently expressed as all $\theta \in \Theta_k$ such that

$$[F_k^\lambda]_j \theta + [G_k^\lambda]_j < -\epsilon_d \tag{19a}$$

$$\left([F_k^\lambda]_j - [F_k^\lambda]_i\right)\theta < \left([G_k^\lambda]_i - [G_k^\lambda]_j\right), \ \forall i \in \mathcal{W}_k \setminus \{j\} \tag{19b}$$

Likewise, the region Θ_k^* defined in (18) can be expressed as

$$\Theta_k^* = \{\theta \in \Theta_k | F_k^\lambda \theta + G_k^\lambda \geq -\epsilon_d\} \tag{20}$$

How regions of the parameter space are partitioned in mode a) is summarized in Algorithm 3.

Algorithm 3 (MODEA) Partition Θ based on if global optimality is reached or if a constraint is removed from $\mathcal{W}$

```
MODEA((Θ, W, F, G, s, k, p̂) , mpQP)
Calculate F^λ and G^λ according to (8b)
for all i in W do
    Calculate Θ^{-i} according to (19)
    if Θ^{-i} ≠ ∅ then
        Append (Θ^{-i}, W\{i}, F, G, 'RM', k+1, [A]_i^T) to P
Calculate Θ* according to (20)
if Θ* ≠ ∅ then
    Append (Θ*, W, F, G, 'OPT', k, p̂) to P
return P
```

C. Adding constraints and checking for local optimality

We now turn our attention to how the parameter space is partitioned in mode b). If j is the minimizing index of the minimization in (10), it will be added to $\mathcal{W}_{k+1}$ and $\alpha_k = \alpha_k^j$. The set Θ_k^{+j} of all parameters in iteration k leading to constraint j being added to $\mathcal{W}_{k+1}$ is, hence, given by

$$\Theta_k^{+j} \triangleq \{\theta \in \Theta_k | [s_k^*]_j < 0, \ \alpha_k^j(\theta) < \alpha_k^i(\theta), \forall i \in \bar{\mathcal{W}}_k^- \setminus \{j\}\} \tag{21}$$

where j being a blocking constraint is ensured by $[s_k^*]_j < 0$, while $\alpha_k^j(\theta) < \alpha_k^i(\theta), \ \forall i \in \bar{\mathcal{W}}_k^- \setminus \{j\}$ ensures that it is the *first* encountered blocking constraint.

Furthermore, the constrained stationary point is primal feasible if $[s_k^*]_i \geq 0 \ \forall i \in \bar{\mathcal{W}}_k$. The set Θ_k^{CSP} of all parameters in iteration k leading to a constrained stationary point being reached is, hence, given by

$$\Theta_k^{\text{CSP}} \triangleq \{\theta \in \Theta_k | [s_k^*]_i \geq 0, \forall i \in \bar{\mathcal{W}}_k\}. \tag{22}$$

In summary, a region Θ_k in mode b) will be partitioned into Θ_k^{CSP} and $\Theta_k^{+i}, \forall i \in \bar{\mathcal{W}}_k$, illustrated in Figure 2b.

The rest of this section is dedicated to deriving explicit expressions for Θ_k^{CSP} and Θ_k^{+j}. First, we formulate an explicit expression for Θ_k^{CSP}, which is straightforward since s_k^* is affine in θ, i.e.,

$$s_k^* = F_{s_k}^* \theta + G_{s_k}^*, \ F_{s_k}^* \triangleq W - AF_k^*, \ G_{s_k}^* \triangleq b - AG_k^*. \tag{23}$$

An explicit expression for Θ_k^{CSP} is, hence, all $\theta \in \Theta_k$ such that

$$[F_{s_k}^*]_i \theta + [G_{s_k}^*]_i \geq 0, \quad \forall i \in \bar{\mathcal{W}}_k. \tag{24}$$

Next we formulate an explicit expression for Θ_k^{+j}, which will prove to be more complicated. From (21), the quantities that define Θ_k^{+j} are $s_k^*(\theta)$ and $\alpha_k^i(\theta)$, where we know from above that $s_k^*(\theta)$ is affine in θ. The main complication for deriving an explicit expression for Θ_k^{+j} is, hence, to establish explicit expressions for $\alpha_k^i(\theta)$, which in turn requires explicit expressions for $p_k(\theta)$ and $x_k(\theta)$.

Because of Property 5 and 6, both p_k and x_k depend on θ in a straightforward way after a constraint has been removed from $\mathcal{W}$, while their dependence on θ will be more intricate before a constraint has been removed. Therefore, we derive explicit expressions for Θ_k^{+j} in two different cases: *Case b1* considers the case when a constraint has been removed from $\mathcal{W}$ in an earlier iteration, whereas *Case b2* considers the case when no constraint has been removed since the start of Algorithm 1.

1) Case b1 - A constraint has been removed from $\mathcal{W}$: If constraint l is the latest constraint removed from $\mathcal{W}$ at iteration $\tilde{k} < k$, p_k can from Property 5 be expressed as

$$p_k(\theta) = -(1 - \tau(\theta))[\lambda_{\tilde{k}}]_l H_k^* [A]_l^T = \gamma(\theta) H_k^* \hat{p}, \tag{25}$$

with the scaling factor $\gamma(\theta) \triangleq -(1 - \tau(\theta))[\lambda_{\tilde{k}}]_l$ and the latest removed normal $\hat{p} \triangleq [A]_l^T$. Also note that $\gamma(\theta) > 0$ since $\tau(\theta) \in [0, 1)$ and $[\lambda_{\tilde{k}}]_l < 0$.

Furthermore, from Property 6, the iterate x_k is affine in θ, i.e. $x_k = F_k \theta + G_k$.

Using these expressions for p_k and x_k, the step length α_k^j defined in (10) is given by

$$\alpha_k^j(\theta) = \frac{[F_{s_k}]_j \theta + [G_{s_k}]_j}{\gamma(\theta)[G_{\sigma_k}]_j}, \tag{26}$$

with $F_{s_k} \triangleq W - AF_k$, $G_{s_k} \triangleq b - AG_k$, and $G_{\sigma_k} \triangleq AH_k^* \hat{p}$.

By inserting expression (26) for α_k^j and expression (9) for s_k^* in (21), Θ_k^{+j} can be explicitly stated as all $\theta \in \Theta_k$ satisfying

$$K_k^{j,i}\theta < L_k^{j,i}, \quad \forall i \in \bar{\mathcal{W}}_k \setminus \{j\}, \tag{27a}$$

$$[F_{s_k}^*]_j\theta + [G_{s_k}^*]_j < 0, \tag{27b}$$

where $K_k^{j,i}$ and $L_k^{j,i}$ are given by

$$K_k^{j,i} \triangleq [G_{\sigma_k}]_i[F_{s_k}]_j - [G_{\sigma_k}]_j[F_{s_k}]_i, \tag{28a}$$

$$L_k^{j,i} \triangleq -[G_{\sigma_k}]_i[G_{s_k}]_j + [G_{\sigma_k}]_j[G_{s_k}]_i. \tag{28b}$$

Remark 4: As can be seen in (27), all partitioning in Case b2 will solely be made by *linear* inequalities.

2) Case b2 - No constraint has been removed from $\mathcal{W}$: When formulating an explicit expressions for Θ_k^{+j} when no constraint has been removed from $\mathcal{W}$, we will, instead of $\alpha_k^j(\theta)$, use the quantity $\tilde{\alpha}_k^j(\theta)$ defined as

$$\tilde{\alpha}_k^j(\theta) \triangleq \frac{[b]_j + [W]_j\theta - [A]_j(H_k^* H x_0(\theta) + T_k b(\theta))}{[A]_j H_k^* H p_0(\theta)}, \tag{29}$$

where x_0 is the starting iterate and $p_0(\theta) = x_0^*(\theta) - x_0(\theta)$. $\tilde{\alpha}_k^j$ can be seen as a measure of the distance between the half-plane $[A]_j x = [b(\theta)]_j$ and the projected starting iterate $H_k^* H x_0$ along the search direction p_k.

The main reason for considering $\tilde{\alpha}_k^j(\theta)$ instead of $\alpha_k^j(\theta)$ is that $\alpha_k^j(\theta)$ is related to θ in an intricate way, whereas $\tilde{\alpha}_k^j$ simply is a linear fraction of θ

$$\tilde{\alpha}_k^j(\theta) = \frac{[\tilde{F}_{s_k}]_j\theta + [\tilde{G}_{s_k}]_j}{[\tilde{F}_{\sigma_k}]_j\theta + [\tilde{G}_{\sigma_k}]_j}, \tag{30}$$

with $\tilde{F}_{s_k}, \tilde{G}_{s_k}, \tilde{F}_{\sigma_k}$ and $\tilde{G}_{\sigma_k}$ defined as

$$\tilde{F}_{s_k} \triangleq W - A(H_k^* H F_0 + T_k W), \tag{31a}$$

$$\tilde{G}_{s_k} \triangleq b - A(H_k^* H G_0 + T_k b), \tag{31b}$$

$$\tilde{F}_{\sigma_k} \triangleq AH_k^* H(F_0^* - F_0),\ \tilde{G}_{\sigma_k} \triangleq AH_k^* H(G_0^* - G_0). \tag{31c}$$

The following lemma makes the relationship between $\tilde{\alpha}_k^j$ and α_k^j more explicit.

Lemma 2: If no constraint has been removed by Algorithm 1 up until iteration k, $\tilde{\alpha}_k^j = \tau + (1-\tau)\alpha_k^j$, $\tau \in [0, 1)$.

Proof: If only constraints have been added to $\mathcal{W}$ since Algorithm 1 started, it follows from Property 2 and 3 that $x_k = H_k^* H(x_0 + \tau p_0) + T_k b(\theta)$ and $p_k = (1-\tau)\tilde{H}_k^* H p_0$ for some $\tau \in [0, 1)$. This inserted into (10) gives

$$\begin{aligned}\alpha_k^j(\theta) &= \frac{[b(\theta)]_j - [A]_j(H_k^* H x_0 + T_k b(\theta)) - \tau[A]_j H_k^* H p_0(\theta)}{[A]_j(1-\tau)H_k^* H p_0(\theta)} \\ &= \frac{1}{1-\tau}\tilde{\alpha}_k^j(\theta) - \frac{\tau}{1-\tau}\end{aligned}$$

which is equivalent to $\tilde{\alpha}_k^j(\theta) = \tau + (1-\tau)\alpha_k^j(\theta)$. ∎

Next, we use Lemma 2 to prove that Θ_k^{+j} can be equivalently expressed in terms of $\tilde{\alpha}_k^j$ instead of α_k^j

Theorem 1: If no constraint has been removed in Algorithm 1 up until iteration k, Θ_k^{+j} defined by (21) equals

$$\{\theta \in \Theta_k | [s_k^*]_j < 0,\ \tilde{\alpha}_k^j(\theta) < \tilde{\alpha}_k^i(\theta), \forall i \in \bar{\mathcal{W}}_k^- \setminus \{j\}\} \tag{32}$$

Proof: From Lemma 2 we have that $\tilde{\alpha}_k^j < \tilde{\alpha}_k^i \Leftrightarrow \tau + (1-\tau)\alpha_k^j < \tau + (1-\tau)\alpha_k^i \Leftrightarrow \alpha_k^j < \alpha_k^i$, where the last equivalence follows from $(1-\tau) > 0$ since $\tau \in [0, 1)$. We can, hence, replace α_k^j and α_k^i with $\tilde{\alpha}_k^j$ and $\tilde{\alpha}_k^i$ in (21). ∎

Θ_k^{+j} can now be explicitly stated, by inserting (30) in (32) and rearranging terms to remove the fractions, as all $\theta \in \Theta_k$ satisfying

$$\theta^T Q_k^{j,i}\theta + R_k^{j,i}\theta + S_k^{j,i} < 0, \quad \forall i \in \bar{\mathcal{W}}_k \setminus \{j\}, \tag{33a}$$

$$[F_{s_k}^*]_j\theta + [G_{s_k}^*]_j < 0, \tag{33b}$$

with definitions $Q_k^{j,i} \triangleq [\tilde{F}_{\sigma_k}]_i^T[\tilde{F}_{s_k}]_j - [\tilde{F}_{\sigma_k}]_j^T[\tilde{F}_{s_k}]_i$, $R_k^{j,i} \triangleq [\tilde{G}_{s_k}]_j[\tilde{F}_{\sigma_k}]_i + [\tilde{G}_{\sigma_k}]_i[\tilde{F}_{s_k}]_j - ([\tilde{G}_{s_k}]_i[\tilde{F}_{\sigma_k}]_j + [\tilde{G}_{\sigma_k}]_j[\tilde{F}_{s_k}]_i)$, and $S_k^{j,i} \triangleq [\tilde{G}_{s_k}]_j[\tilde{G}_{\sigma_k}]_i - [\tilde{G}_{s_k}]_i[\tilde{G}_{\sigma_k}]_j$. The constraints in (33b) ensures that $[A]_j x \leq [b(\theta)]_j$ is a blocking constraint and (33a) ensures that it is the *first* blocking constraint.

Remark 5: As can be seen in (33), the parameter space will be partitioned by linear *and*, in contrast to Case b1 (see Remark 4), quadratic inequalities when a constraint is added to the working set under Case b2. Quadratic inequalities make the analysis less tractable compared to only linear inequalities and we will, hence, give some alternatives to circumvent these in Section V-B and V-C.

The results from Section IV-C1 and IV-C2, i.e., how regions are partitioned in mode b), are summarized in Algorithm 4.

Algorithm 4 (MODEB) Partition Θ based on if a CSP is reached or if a constraint is added to $\mathcal{W}$. (Case b1/Case b2)

```
MODEB((Θ, W, F, G, s, k, p̂) , mpQP)
Compute F* and G* according to (8a)
for all i in W̄ do
    Calculate Θ^{+i} according to (27)/(33)
    F_+ ← F + ΔF, G_+ ← G + ΔG using (16)/(-)
    if Θ^{+i} ≠ ∅ then
        Append (Θ^{+i}, W∪{i}, F_+, G_+, 'NUL', k+1, p̂) to P
Calculate Θ^CSP according to (24)
if Θ^CSP ≠ ∅ then
    Append (Θ^CSP, W, F*, G*, 'CSP', k+1, p̂) to P
return P
```

D. Adding constraints when the EQP is unbounded

In mode c) the singular direction $\tilde{p}_k$ is central in determining both unboundedness or which constraint that will be added to $\mathcal{W}$. Since $\tilde{p}_k$ is independent of θ, see Property 7, the unboundedness is binary: Either it holds for *all* parameters or *no* parameters in Θ_k, formalized in the following theorem.

Theorem 2: Let Θ_k be a region for which $\tilde{H}_k$ became singular after constraint l was removed in iteration $k-1$. Furthermore, let $\Theta_k^\infty \subseteq \Theta_k$ be the parameters in Θ_k which result in an unbounded problem. Then either $\Theta_k^\infty = \emptyset$ or $\Theta_k^\infty = \Theta_k$.

Proof: By combining (12) and Property 7, $\tilde{p}_k$ solves $H\tilde{p}_k = 0, A_k\tilde{p}_k = 0, [A]_l^T\tilde{p}_k < 0$, which is independent of the parameter θ. Moreover, the parameter independence of $\tilde{p}_k$ also implies that the unboundedness condition $\tilde{\sigma}_k \triangleq A\tilde{p}_k \geq 0$ is independent of θ so either $\tilde{\sigma}_k \geq 0, \forall\theta \in \Theta_k$, implying that $\Theta_k^\infty = \Theta_k$, or $\tilde{\sigma}_k \ngeq 0, \forall\theta \in \Theta_k$, implying that $\Theta_k^\infty = \emptyset$. ∎

Furthermore, the possible constraints that can be added to $\mathcal{W}_k$ are all $i \in \bar{\mathcal{W}}_k$ such that $[\tilde{\sigma}_k]_i < 0$, which, again, is parameter independent since $\tilde{\sigma}_k \triangleq A\tilde{p}_k$.

Additions to $\mathcal{W}_k$ in mode c) are similar to additions in mode b), the only difference is which step direction is used (p_k in mode b) and $\tilde{p}_k$ in mode c)). The partition will, hence, be similar to the one described in Section IV-C1. The only change needed is to replace $\hat{p}$ with $\tilde{p}_k$. Hence, if the problem is unbounded ($\tilde{\sigma}_k \not\geq 0$), the region of Θ_k in which j is added is given by

$$\Theta_k^{+j} = \left\{\theta \in \Theta_k | \alpha^j(\theta) < \alpha^i(\theta), \forall i : [\tilde{\sigma}_k]_i < 0\right\}, \qquad (34)$$

which can, similar to (27), be written as all θ in Θ_k such that

$$\tilde{K}_k^{j,i}\theta < \tilde{L}_k^{j,i}, \quad \forall i \in \mathcal{W}_k \setminus \{j\} : [\tilde{\sigma}_k]_i < 0, \qquad (35)$$

with the definitions $\tilde{K}_k^{j,i} \triangleq [\tilde{\sigma}_k]_i[F_{s_k}]_j - [\tilde{\sigma}_k]_j[F_{s_k}]_i$ and $\tilde{L}_k^{j,i} \triangleq -[\tilde{\sigma}_k]_i[G_{s_k}]_j + [\tilde{\sigma}_k]_j[G_{s_k}]_i$.

In summary, a region Θ_k in mode c) is partitioned into regions $\Theta_k^{+i}, \forall i : [\tilde{\sigma}_k]_i < 0$, or the *entire* Θ_k is marked as unbounded if $\tilde{\sigma}_k > 0$, illustrated in Figure 2c and summarized in Algorithm 5.

Algorithm 5 (MODEC) Partition Θ based on if an unbounded problem is identified or if a constraint is added to $\mathcal{W}$.

```
MODEC((Θ, W, F, G, s, k, p̂) , mpQP)
Compute p̃ by solving Hp̃ = 0,  [A]_W p̃ = 0,  p̂^T p̃ < 0
σ̃ ← Ap̃
if σ̃ ≥ 0 then
    Append (Θ, W, F, G, 'UNB', ∞, p̂) to P
    return P
for all i : [σ̃]_i < 0 do
    Calculate Θ^{+i} according to (35)
    Calculate ΔF, ΔG using (16) (with p̃ replacing H_k^* p̂)
    F_+ ← F + ΔF,   G_+ ← G + ΔG
    if Θ^{+i} ≠ ∅ then
        Append (Θ^{+i}, W ∪ {i}, F_+, G_+, 'NUL', k+1, p̂) to P
return P
```

V. EXTENSIONS AND PRACTICAL ASPECTS

A. Certifying number of floating-point operations

As previously mentioned, there are many primal active-set algorithms in the literature which are mathematically equivalent [24], in the sense that they produce the same sequence of iterates. The main difference between algorithms is how, and which, matrices are factorized for solving the KKT system. In Algorithm 2 we have deliberately abstracted away, e.g., how the KKT system is solved to cover numerous of these active-set algorithms simultaneously. To determine the flops for a specific algorithm, however, one needs a mapping $\mathcal{F}(\mathcal{W}_k)$ which, given a working set, computes the required number of flops to, e.g., solve the KKT system for the particular solver of interest. By using such a mapping, the exact number flops for the particular solver can be determined since the exact sequence of working-set changes is determined by Algorithm 2 $\forall \theta \in \Theta_0$.

Furthermore, by using multiple mappings corresponding to different solvers, their complexity can be compared *simultaneously*. Through such a simultaneous comparison, the choice of which solver is best, in terms of the least number of flops, for the specific problem at hand can be determined.

B. Special cases with simplified partition

As has been shown in (33), the application of the proposed method to a general mpQP might result in a partitioning of the parameter space using not only affine but also quadratic inequalities. The significance of this is during the pruning of empty regions, done at Step 5 and 8 of Algorithm 3, Step 6 and 9 of Algorithm 4 and Step 11 of Algorithm 5, since to check consistency of a combination of linear and quadratic constraints is non-trivial. However, there are some relevant cases when the partitioning is solely composed of affine constraints, resulting in easier feasibility checks (a single LP solve per check). Such special cases are described below.

1) No state constraints: When there are no constraints on the states, a linear MPC problem can be formulated as an mpQP with $W = 0$. Additionally, an admissible control input can be picked as a fixed starting point, i.e., $F_0 = 0$. This will result in $[\tilde{F}_{s_k}]_j = 0$ in (31a) which in turn results in $Q_k^{j,i} = 0$. Therefore, all partitioning of the parameter space will be done using half-planes, leading to a polytopic partition.

2) Starting in a constrained stationary point: When the initial point is a CSP, partitioning according to Case b2 will never occur since either the CSP will be dual feasible, resulting in the algorithm directly terminating with a global solution, or a constraint will be removed. All partitioning will, hence, be made be affine inequalities since partitioning with quadratic inequalities only occurs under Case b2 (see Remark 5). Hence, under the assumption that Θ_0 is a polyhedron, the final partition will be polytopic.

3) Reformulate QP using a quadratic penalty method: All inequality constraints that depend on parameters can be transformed to equality constraints by introducing slack variables. These equality constraints can then be moved to the objective function under a quadratic penalty, cf. e.g., [26][2, Sec.17-1]. The resulting QP will be on the form which was discussed in special case 1), described above.

C. Outer approximations of quadratic inequalities

If the problem does not fall into any of the special cases above, the comparison of step lengths $\alpha_k^i(\theta) < \alpha_k^j(\theta)$ to find the first blocking constraint when a constraint is yet to be removed from $\mathcal{W}$, i.e. under Case b2, results in the quadratic inequalities (33a) on the form $\theta^T Q\theta + R\theta + S < 0$.

An alternative to handle these quadratic constraints is to make an affine outer-approximation with the half-plane

$$R\theta < -S - \min_{\theta \in \Theta_k} \theta^T Q\theta \qquad (36)$$

where Θ_k is the current region. Hence, by solving an indefinite QP in relatively low dimension, an affine relaxation can be obtained. Ultimately, this relaxation might lead to some regions overlapping, giving a conservative result since all regions produced by the certification method might not correspond to how Algorithm 1 performs in practice.

An interpretation of relaxing $\alpha_k^i(\theta) < \alpha_k^j(\theta)$ with (36) is that the i:th constraint might not be the *first* blocking constraint for that particular parameter region in iteration k. This would result in a primal infeasible iterate, which can be used in the certification algorithm to prune some of the redundant

regions which the outer-approximation might yield. Checking for infeasibility of the iterate during Case b2 will, again, lead to quadratic regions and is therefore of no use. As soon as a constrained stationary point is reached, however, the iterates become affine in θ, see Property 6, and the affine constraint $Ax_k^*(\theta) \leq b + W\theta$ can be added to the current region to prune infeasible iterates. In the end, the only redundant regions that remain will correspond to iterates that regained primal feasibility before the first CSP was reached.

D. Constraint selection rules

In Algorithm 1, Dantzig's selection rule, i.e., picking the most negative component of λ, has been used when selecting which constraint to remove from $\mathcal{W}$. Convergence can, however, be guaranteed if *any* constraint corresponding to a negative component of λ is removed from $\mathcal{W}$. To retain the tractability of the certification method though, the partitioning made by the selection rule should not be too complex, preferably with half-planes (as Dantzig's rule). A class of selection rules which leads to partitioning with half-planes are combinatorial rules described in [27], including Bland's rule which, together with Dantzig's rule, were considered in [19].

As a final remark, the framework presented in this work could also be used to determine if it is beneficial to remove more than one constraint corresponding to a negative dual variable at a time or if it leads to more computations as the active-set algorithm progresses. This would, however, lead to some of the nice properties under Case b1, described in IV-C1, being lost, making the certification method less tractable (because of partitioning with quadratic inequalities).

E. Warm-starting

In the derivation of Algorithm 2 in Section IV, we have used an arbitrary $\mathcal{W}_0$, $x_0 = F_0\theta + G_0$ and $\Theta_0 \subset \mathbb{R}^p$ being any polyhedron. This general setup can be used for certifying the complexity of semi-explicit warm starts for a given mpQP, akin to the semi-explicit method presented in [12] for mpLPs. Moreover, the setup also allows for using the explicit solution to analyze warm starts, assuming the solution in the previous time step is used to warm-start the solver online and it is known how it is used, e.g., if it is shifted or not.

Complexity certified warm starts for active-set methods is an area for future research, where the presented certification method herein is an important first step.

VI. Relationship to other methods

A. Dual active-set methods for Quadratic Programming

As is noted in [21, p.244] and [3], the popular dual active-set method presented in [3] is mathematically equivalent to Algorithm 1 applied to the dual of (1) when $H \succ 0$. The dual problem to (1) can be stated as the mpQP

$$\underset{\lambda \geq 0}{\text{minimize}} \ \frac{1}{2}\lambda^T A^T H^{-1} A\lambda + (b(\theta) + AH^{-1}f(\theta))^T\lambda, \quad (37)$$

where the optimal primal solution x^* can be recovered from the optimal dual solution λ^* by $x^* = -H^{-1}(f(\theta) + A^T\lambda^*)$. The complexity certification method for the dual method in [3] which is provided in [19] can, hence, be seen as a special case of Algorithm 2 applied to the mpQP in (37).

The certification of dual active-set methods which are not started in the unconstrained optimum is not considered in [19]. However, viewing the method in [19] as Algorithm 2 applied to (37) allows for complexity certification of dual active-set methods with arbitrary, dual feasible, starting iterates. Being able to do the certification from an arbitrary starting iterate is necessary when analyzing warm starts.

In [19] the certified number of iterations is shown to be constant over a polyhedral partition. This is in contrast with the results in Section IV where, in general, both affine *and* quadratic inequalities partitions the parameter space for Algorithm 2. There are two factors that, separately, lead to the lack of quadratic partitioning in [19]. First, since the dual active-set method considered in [19] is always initialized in the unconstrained optimum, it falls into the special case described in Section V-B2. Another reason reason for a final polyhedral partition is that (37) has more structure than the generic mpQP in (1), i.e., that there is no parameter dependence in the constraints. This additional structure can, with the same reasoning as in the special case described in Section V-B1, lead to a polyhedral partition.

B. Active-set methods for Linear Programming

Using another formulation, more concretely using the 1- and ∞-norm instead of the 2-norm in the cost function, linear MPC problems can be cast as mpLPs, see, e.g., [28, Sec.2-3]. mpLPs can be seen as a special class of mpQPs with $H = 0$.

A well-renowned method for solving LPs is the simplex method [23, Sec. 5] which can also be seen as an active-set method. In fact, Algorithm 1 applied to an LP is mathematically, but not numerically, equivalent to the simplex method with Dantzig's pivot rule [29]. The iterates of the simplex method are vertices of the feasible set and we will now briefly describe how this translates to the behaviour of Algorithm 1. Since a vertex is a constrained stationary point, we will check for optimality or remove a constraint from our working set (mode a)). Removing a constraint leads to mode c) being entered since the reduced Hessian is trivially singular ($H = 0$). This will either lead to unboundedness being detected or a constraint being added. In the latter case, the iterate is a new vertex and mode a) is entered again. Because of the above-mentioned equivalence, the complexity of the simplex method can, hence, also be certified using Algorithm 2.

An alternative to the simplex method for solving LPs are other active-set algorithms which do not restrict all iterates to vertices. Such a method is considered in [12] and uses the gradient of the objective function as search direction. Hence, by computing the search direction by solving the KKT system

$$\begin{pmatrix} I & A_k^T \\ A_k & 0 \end{pmatrix} \begin{pmatrix} p_k \\ \lambda_k \end{pmatrix} = \begin{pmatrix} -f(\theta) \\ 0 \end{pmatrix}, \quad (38)$$

and modifying Algorithm 2 accordingly, the LP method in [12] can also be certified using Algorithm 2.

C. Explicitly solving mpQP

The proposed method is also similar to computing the explicit solution to the mpQP, which is used in explicit MPC [1], since the explicit solution is a by-product in the final partition of Algorithm 2. The method therefore shares the complexity of explicit MPC which has the well-known drawback of growing quickly as the problem size grows.

An important advantage of the proposed certification method is, however, that this growth in complexity only has to be dealt with offline, while it has to be dealt with offline *and* online for explicit MPC (since the critical regions have to be stored and looked up online). Not only is the storage a limitation for the use of explicit MPC on embedded system with limited memory, but the point location can also be more demanding than solving the problem online, as is shown in [19]. The proposed method is therefore applicable to problems which are out-of-bounds for explicit MPC, for example, one could use Algorithm 2 on problems with a vast number of critical regions which might take hours, or even days, to certify. Once Algorithm 2 completes, however, the active-set method can be used online with real-time guarantees. The additional memory and computational requirements online due to that the QP algorithm has been certified is zero, i.e., the requirements are identical to if the online QP algorithm would not have been certified.

Moreover, since the complexity is in offline computations for the proposed method, high-performance computing (HPC) can be used to further increase the problem sizes which can be handled, especially since Algorithm 2 is well-suited for parallelization by distributing the stack S over multiple processors.

VII. NUMERICAL EXAMPLES

Some benchmark problems from the MATLAB Model Predictive Control Toolbox will be considered to test the proposed certification method. These MPC problems are the control of a double integrator, a DC-motor, an inverted pendulum, a linearized nonlinear multiple-input-multiple-output system and an ATFI-F16 aircraft. The tracking problem is considered, resulting in a parameter vector θ containing the state vector, the previous control input and the reference signal. The same problems were also considered in the context of real-time certification for other QP methods in [19] and [20], where they were considered a good representation of the kind of problems encountered in real-time MPC. For further details about the problems see [19] and [20]. Additionally, the method is tested on a randomly generated mpQP to accentuate the possibility of quadratic partitioning of the parameter space. This problem will be called "Contrived mpQP" and is given by

$$H = \begin{pmatrix} 0.97 & 0.19 & 0.15 \\ 0.19 & 0.98 & 0.05 \\ 0.15 & 0.05 & 0.99 \end{pmatrix}, \quad A = \begin{pmatrix} 0.38 & 2.20 & 0.43 \\ 0.49 & 0.57 & 0.22 \\ 0.77 & 0.46 & 0.41 \end{pmatrix},$$
$$f = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T, \qquad b = \begin{pmatrix} 4.1 & 3.7 & 4.3 \end{pmatrix}^T,$$
$$W = \begin{pmatrix} 0.19 & -0.89 \\ 0.62 & -1.54 \\ -0.59 & -1.01 \end{pmatrix}, \qquad f_\theta = \begin{pmatrix} 11.3 & -44.3 \\ -3.66 & -11.9 \\ -32.6 & 7.81 \end{pmatrix}.$$

The certification method presented in Section IV is applied to the resulting primal mpQP problems on the form (1) for all of the MPC examples, with the starting iterate being the origin, i.e. $x_0 = (0, ..., 0)^T$ and the starting working set being the empty set, i.e. $\mathcal{W}_0 = \emptyset$. Since the DC motor and ATFI-F16 aircraft examples contain state constraints, these constraints are softened (see, e.g., [30]) to ensure the existence of primal feasible solutions. Furthermore, the initial slack is set large enough to ensure primal feasibility of the origin for all parameters of interest.

In addition to the primal problems, the certification method is applied to the dual problems on the form (37), which are positive semi-definite. For all of the dual experiments, the starting iterate is chosen as $\lambda_0 = (0, ..., 0)^T$ and all constraints of the dual problem are active in the initial working set, i.e., $\mathcal{W}_0 = \mathcal{K} = \{1, \ldots, m\}$, corresponding to starting in the unconstrained primal solution.

Gurobi 9.0 [31] was used to decide if regions described by both linear and quadratic inequalities were empty or not.

A. Complexity certification

To give a taste of the final result from Algorithm 2, Figure 3 depicts a low-dimensional slice of the resulting regions which lead to the same number of QP iterations when the primal problems are solved with Algorithm 1, determined by Algorithm 2. However, this is only a subset of the information contained in the final partition since every region also contains the exact sequence of working-set changes performed to reach the solution. As an example, the parameters in the final region of the contrived mpQP example which contains $\theta = [0.5, 0.5]^T$, (the purple region in the middle of Figure 3a), have undergone the following working-set changes: $\emptyset \to \{1\} \to \{1, 3\} \to \{3\}$ before reaching optimality.

The dimensions of the resulting mpQPs for the examples are shown in Table I together with the maximum number of QP iterations $N^{\max}_{\text{primal}}$ and $N^{\max}_{\text{dual}}$ needed for the active-set algorithm to provide a solution when solving the primal and dual problem, respectively, determined by Algorithm 2. The table also includes the time taken for the certification t^{cert} and the number of regions N^{reg} in the final partition. Furthermore, the maximum number of QP iterations observed when running Monte Carlo (MC) simulations, denoted $N^{\max}_{\text{MC}}$, were obtained by random sampling of Θ_0 and applying Algorithm 1 to the resulting QPs. For the MC simulations, as many samples as possible were drawn during t^{cert} to compare with the certification method.

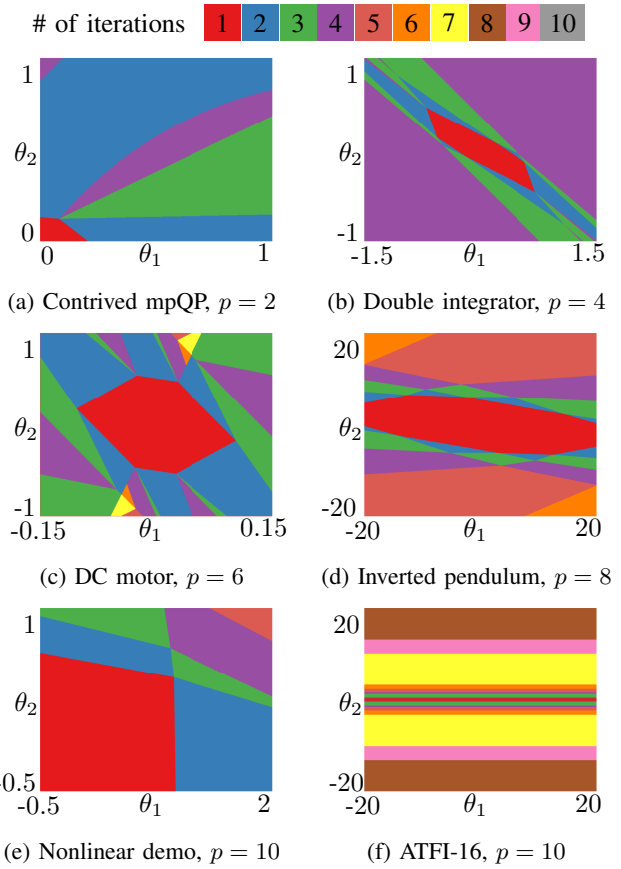
By comparing $N^{\max}_{\text{prim}}$ with $N^{\max}_{\text{dual}}$ in Table I it can be seen that the dual method needs fewer iterations in the worst-case for most of the examples, which is in accordance with what is noted in [3]. However, for the ATFI-F16 example the primal method needs fewer iterations in the worst-case. Hence, whether the primal or dual active-set approach is to be preferred, from a real-time perspective, is, not surprisingly, problem dependent and the proposed certification method can be used to decide which one gives the fewest iterations in the worst-case for a given problem.

It can also be seen that $N^{\max}_{\text{MC,prim}} < N^{\max}_{\text{prim}}$ and $N^{\max}_{\text{MC,dual}} < N^{\max}_{\text{dual}}$ for some of the examples, highlighted

TABLE I: Dimensions of the resulting mpQPs for the examples, the worst-case number of QP-iterations $N^{\max}$ determined by Algorithm 2 and the worst-case number of QP-iterations $N^{\max}_{\text{MC}}$ determined by extensive simulation. N^{reg} is the number of regions in the final partition and t^{cert} is the time taken by a MATLAB implementation of Algorithm 2 executed on an Intel 2.7 GHz i7-7500U CPU. The subscripts "prim" or "dual" denote results when the primal or the dual QP were solved, respectively.

	p	n	m	$N^{\max}_{\text{prim}}$	$N^{\max}_{\text{dual}}$	$t^{\text{cert}}_{\text{prim}}[s]$	$t^{\text{cert}}_{\text{dual}}[s]$	$N^{\text{reg}}_{\text{prim}}$	$N^{\text{reg}}_{\text{dual}}$	$N^{\max}_{\text{MC,prim}}$	$N^{\max}_{\text{MC,dual}}$
Contrived mpQP	2	3	3	4	4	0.08	0.01	6	5	4	4
Double integrator	4	3	6	6	6	0.13	0.08	39	43	6	6
Inverted pendulum	8	5	10	19	14	15	7.6	2499	1839	19	14
DC motor*	6	3	10	14	14	46	11	2309	1865	10	10
Nonlinear demo	10	6	12	14	11	56	41	10166	8669	12	11
ATFI-F16*	10	5	12	21	24	541	558	41971	93064	14	15

* For the primal problem, quadratic inequalities were outer-approximated by affine inequalities as described in Section V-C.



(a) Contrived mpQP, $p = 2$ (b) Double integrator, $p = 4$

(c) DC motor, $p = 6$ (d) Inverted pendulum, $p = 8$

(e) Nonlinear demo, $p = 10$ (f) ATFI-16, $p = 10$

Fig. 3: 2D-slice of the resulting parameter regions with $\theta_i = 0$, $i > 2$, produced by Algorithm 2 for the primal problems. The same color means same number of QP iterations.

in red in Table I. This either means that the certification method is conservative or that the MC simulations are optimistic, (or both). However, since the certification method provides a region in parameter space for which the worst-case number of iterations is obtained, a parameter in the worst-case region for each example was extracted and by applying Algorithm 1 to the resulting QP it could be proven that the certification method did not provide a conservative result. Instead, the discrepancies are due to MC simulations not being able to cover the parameter space densely enough with samples during the allotted time. Even if more samples could be taken to improve the MC results, this would require more time than the certification method and, still, there are no guarantees for sufficient coverage for any finite number of samples. This underlines an important advantage of the proposed certification method compared to MC simulations, namely that the proposed method covers a continuum of points, which becomes increasingly beneficial as the dimension of the parameter space increases.

Remark 6: The execution time t^{cert} is based on an implementation of Algorithm 2 in MATLAB. Modifications to the implementation, such as low-rank modifications and parallelizing computations, are expected to significantly reduce t^{cert}.

B. Affine approximations of quadratic inequalities

The affine outer-approximations of quadratic constraints, described in Section V-C, were tested by using Algorithm 2 with and without these relaxations on the problems which lead to quadratic partitioning of the parameter space, i.e., the contrived mpQP, DC motor and ATFI-16 aircraft example. Table II summarizes the result, where it can be seen that approximating the quadratic constraints results in the final partition containing more regions, given by N^{reg}, for all of the examples. This is expected since, as is discussed in Section V-C, the relaxations might lead to redundant regions. For the contrived mpQP, the relaxation results in an upper bound on the number of QP iterations $N^{\max}_{\text{iter}}$ of 6 instead of the tight upper bound 4. However, for both the DC motor and ATFI-F16 example the upper bounds provided by the relaxation coincide with the tight upper bound. Table II also shows that, for large problems, the computation time t^{cert} for the certification can be significantly reduced by forming affine outer-approximations of the quadratic constraints. In conclusion, relaxing quadratic constraints with the method described in Section V-C can provide good, even tight, upper bounds on worst-case behaviour while reducing the certification time for large problems.

TABLE II: Comparison of the certification method when linear outer-approximations of quadratic constraints are/are not used.

	$N^{\max}_{\text{iter}}$	N^{reg}	$t^{\text{cert}}[s]$
Contrived mpQP	6/4	15/6	0.76/0.08
DC motor	14/14	2309/1765	46/263
ATFI-F16	21/21	41971/31831	541/4689

VIII. CONCLUSION AND FUTURE WORK

In this paper we have presented a method which extends, and unifies, complexity certification results for active-set QP

and LP methods. The method computes *exactly* which sequence of working-set changes, as a function of the parameters in an mpQP, a primal active-set QP algorithm will undergo to find an optimum. This can be used to determine an upper bound on the number of QP iterations the algorithm will need when it is applied online, which is of importance in the context of real-time MPC where hard real-time requirements have to be fulfilled. The method partitions the parameter space into regions, defined by affine and quadratic inequalities, representing parameter sets which generate the same sequence of working-set changes to reach a solution. Furthermore, by considering positive semi-definite QPs, the proposed method poses previous complexity certification results for primal and dual active-set QP methods, as well as active-set LP methods, in a unified framework. The proposed method was successfully applied to a set of linear MPC problems to illustrate how it can be used to determine the worst-case number of iterations needed by primal and dual active-set algorithms online.

Future work includes using the framework to compare the worst-case number of FLOPs different active-set algorithms require, e.g., to consider the difference between different range-space and null-space methods.

REFERENCES

[1] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[2] J. Nocedal and S. Wright, *Numerical Optimization*. Springer Science & Business Media, 2006.

[3] D. Goldfarb and A. Idnani, "A numerically stable dual method for solving strictly convex quadratic programs," *Mathematical Programming*, vol. 27, pp. 1–33, 9 1983.

[4] K. Kunisch and F. Rendl, "An infeasible active set method for quadratic problems with simple bounds," *SIAM Journal on Optimization*, vol. 14, pp. 35–52, 01 2003.

[5] 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: IFAC-Affiliated Journal*, vol. 18, no. 8, pp. 816–830, 2008.

[6] A. Bemporad, "A quadratic programming algorithm based on nonnegative least squares with applications to embedded model predictive control," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1111–1116, 2015.

[7] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *Journal of Optimization Theory and Applications*, vol. 99, no. 3, pp. 723–757, 1998.

[8] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on control systems technology*, vol. 18, no. 2, pp. 267–278, 2010.

[9] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for embedded linear model predictive control," *IEEE Transactions on Automatic Control*, vol. 59, pp. 18–33, 01 2014.

[10] D. Axehill and A. Hansson, "A dual gradient projection quadratic programming algorithm tailored for model predictive control," in *2008 47th IEEE Conference on Decision and Control*. IEEE, 2008, pp. 3057–3064.

[11] S. Richter, C. N. Jones, and M. Morari, "Real-time input-constrained MPC using fast gradient methods," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, 2009, pp. 7387–7393.

[12] M. N. Zeilinger, C. N. Jones, and M. Morari, "Real-time suboptimal model predictive control using a combination of explicit MPC and online optimization," *IEEE Transactions on Automatic Control*, vol. 56, pp. 1524–1534, 07 2011.

[13] M. Herceg, C. Jones, and M. Morari, "Dominant speed factors of active set methods for fast MPC," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 608–627, 2015.

[14] V. Klee and G. J. Minty, "How good is the simplex algorithm," *Inequalities*, vol. 3, no. 3, pp. 159–175, 1972.

[15] D. A. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," *Journal of the ACM (JACM)*, vol. 51, no. 3, pp. 385–463, 2004.

[16] S. Richter, C. N. Jones, and M. Morari, "Computational complexity certification for real-time MPC with input constraints based on the fast gradient method," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1391–1403, 2012.

[17] P. Giselsson, "Execution time certification for gradient-based optimization in model predictive control," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 3165–3170.

[18] D. Arnström and D. Axehill, "Exact complexity certification of a standard primal active-set method for quadratic programming," in *2019 IEEE 58th Conference on Decision and Control*, Dec 2019, pp. 4317–4324.

[19] G. Cimini and A. Bemporad, "Exact complexity certification of active-set methods for quadratic programming," *IEEE Transactions on Automatic Control*, vol. 62, pp. 6094–6109, 2017.

[20] ——, "Complexity and convergence certification of a block principal pivoting method for box-constrained quadratic programs," *Automatica*, vol. 100, pp. 29–37, 2019.

[21] R. Fletcher, *Practical Methods Of Optimization*. John Wiley & Son Ltd, 1987.

[22] ——, "A general quadratic programming algorithm," *IMA Journal of Applied Mathematics*, vol. 7, no. 1, pp. 76–91, 1971.

[23] G. B. Dantzig, *Linear programming and extensions*. Princeton University Press, 1963.

[24] M. J. Best, "Equivalence of some quadratic programming algorithms," *Mathematical Programming*, vol. 30, no. 1, p. 71, 1984.

[25] D. Arnström and D. Axehill, "Exact complexity certification of an early-terminating standard primal active-set method for quadratic programming," in *21st IFAC World Congress*, 2020.

[26] N. Saraf, M. Zanon, and A. Bemporad, "A fast NMPC approach based on bounded-variable nonlinear least squares," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 337–342, 2018.

[27] T. Terlaky and S. Zhang, "Pivot rules for linear programming: a survey on recent theoretical developments," *Annals of Operations Research*, vol. 46, no. 1, pp. 203–233, 1993.

[28] F. Borrelli, *Constrained optimal control of linear and hybrid systems*. Springer, 2003, vol. 290.

[29] P. E. Gill and E. Wong, "Methods for convex and general quadratic programming," *Mathematical Programming Computation*, vol. 7, no. 1, pp. 71–112, 2015.

[30] A. Zheng and M. Morari, "Stability of model predictive control with mixed constraints," *IEEE Transactions on automatic control*, vol. 40, no. 10, pp. 1818–1823, 1995.

[31] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2020. [Online]. Available: http://www.gurobi.com