Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Exact Complexity Certification of an Early-Terminating Standard Primal Active-Set Method for Quadratic Programming

Daniel Arnström * **Daniel Axehill** *

* *Division of Automatic Control, Linköping University, Sweden (e-mail: daniel.{arnstrom,axehill}@liu.se).*

Abstract: In this paper we present a method to exactly certify the iteration complexity of a primal active-set algorithm for quadratic programs which is terminated early, given a specific multi-parametric quadratic program. The primal active-set algorithm's real-time applicability is, hence, improved by early termination, increasing its computational efficiency, and by the proposed certification method, providing guarantees on worst-case behaviour. The certification method is illustrated on a multi-parametric quadratic program originating from model predictive control of an inverted pendulum, for which the relationship between allowed suboptimality and iterations needed by the primal active-set algorithm is presented.

Keywords: Quadratic programming, Active-set methods, Linear model predictive control, Suboptimal control, Complexity certification

1. INTRODUCTION

In model predictive control (MPC) the control problem is formulated as a parametric optimization problem which is solved in each sample to introduce feedback. In linear MPC, the problem is often formulated in a way that results in the optimization problems being quadratic programs (QPs) that depend on parameters such as the current state of the system, making them multi-parametric quadratic programs (mpQPs). In embedded MPC it is important that these optimization problems can be solved under real-time constraints, especially when the computational resources needed are close to the computational resources at hand.

A class of methods for solving QPs are active-set methods such as the primal method presented in Nocedal and Wright (2006), the dual method presented in Goldfarb and Idnani (1983) and the primal-dual method presented in Kunisch and Rendl (2003). Methods for *exactly* certifying the complexity of these methods, when the QP is solved to optimality, have been presented in Arnström and Axehill (2019), Cimini and Bemporad (2017) and Cimini and Bemporad (2019), respectively. Complexity certification of a primal active-set method for linear programs (LPs) has been presented in Zeilinger et al. (2011).

A property of *primal* active-set methods which make them desirable for real-time optimization is that all the iterates are primal feasible. Hence, the algorithm can be terminated before optimality has been reached, reducing the computations needed, while still providing a solution that is admissible. In the context of MPC, solving the QPs to optimality is often unnecessary to achieve sufficient control performance. Furthermore, if the suboptimality is moderate and the suboptimal solution is primal feasible, stability can be guaranteed, Scokaert et al. (1999), Michalska and Mayne (1993), Graichen and Kugi (2010).

In this paper we present a method for exactly certifying the worst-case number of iterations a standard primal active-set QP algorithm, presented in Nocedal and Wright (2006), needs for an iterate to be sufficiently close to the optimal solution. Given a user-defined tolerance for the suboptimality of the solution, the proposed method determines the worst-case number of iterations needed by the QP algorithm for the iterate to be within this tolerance. The method is an extension to the certification method presented in Arnström and Axehill (2019), where the same primal active-set algorithm is analyzed when being applied until it converges to optimality. The extension makes it possible to certify exactly how many iterations are required, for any parameter value, until the iterate is within the user-defined tolerance from the optimal solution. Ultimately the method provides information about exactly how much computational effort can be saved online if the solution is allowed to be suboptimal to a certain degree.

The outline of the paper is as follows. Section 2 presents some preliminary information about mpQPs, the primal active-set algorithm considered and the certification method presented in Arnström and Axehill (2019) while at the same time introducing notation used throughout the paper. Section 3 presents two different ways of terminating the active-set algorithm early and how to incorporate these in the certification method. Finally, Section 4 illustrates the proposed method on an mpQP originating from the control of an inverted pendulum using MPC.

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

2. PRELIMINARIES

Consider the multi-parametric quadratic program (mpQP)

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

where $x \in \mathbb{R}^n$ and the parameter $\theta \in \Theta_0 \subseteq \mathbb{R}^p$, with Θ_0 being a polyhedron. The mpQP is given by $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $W \in \mathbb{R}^{m\times p}$, $f \in \mathbb{R}^n$, $f_\theta \in \mathbb{R}^{n\times p}$, and $H \in \mathbb{S}^n_{++}$. The minimizing argument of (1) given θ is denoted $x^*(\theta)$ and the minimizing value of the objective function given θ is called the value function and is denoted $J^*(\theta)$.

A linear MPC problem using a quadratic cost can be cast in the form (1), where the parameter θ contains the measured/estimated state, Bemporad et al. (2002).

Another way of expressing the feasible set in (1) is in terms of each constraint as $[A]_i x \leq [b]_i + [W]_i\theta, \forall i \in \mathcal{K}$, where the notation $[M]_i$ denotes the i:th row of the matrix M and $\mathcal{K} \triangleq \{1,\dots,m\}$ is the set of all constraint indices. A constraint which holds with equality is said to be *active*.

Since the primal active-set algorithm considered in this paper is iterative, we introduce notation for quantities that change in each iteration. x_k and λ_k denote the primal and dual iterates, respectively, at iteration k. $\mathcal{W}_k$ denotes the working set at iteration k, which contains a subset of the constraints that are active at x_k. Furthermore, $\mathcal{C}_k \triangleq \mathcal{K}\backslash\mathcal{W}_k$ denotes the complement to the working set at iteration k.

2.1 A primal active-set algorithm

The primal active-set algorithm considered in this paper is given in Algorithm 1 and is described briefly below, see Nocedal and Wright (2006) for a more extensive description. The algorithm starts with a feasible starting iterate x_0 and a starting working set $\mathcal{W}_0$ which contains a subset of the constraints that are active at x_0 and the main objective of the algorithm is to iteratively update the iterate x_k and the working set $\mathcal{W}_k$ by adding/removing constraints until x^* is found. In iteration k, the algorithm computes a search direction p_k by solving an equality-constrained quadratic program (EQP), where the equality constraints of the EQP are given by the working set $\mathcal{W}_k$. A step is taken in the direction of p_k until either a constraint that is not in $\mathcal{W}_k$ becomes active or a constrained stationary point with respect to $\mathcal{W}_k$ is reached.

Definition 1. An iterate $\tilde{x}_k(\theta)$ is said to be a constrained stationary point (CSP) with respect to $\mathcal{W}_k$ if

$$\begin{aligned} \tilde{x}_k(\theta) = \text{argmin}_x \quad & J(x,\theta) \\ \text{s.t} \quad & [A]_i x = [b]_i + [W]_i\theta, \quad i \in \mathcal{W}_k. \end{aligned} \tag{2}$$

If a constraint that is not in the working set becomes active before reaching a CSP, the constraint is added to the working set to maintain primal feasibility and a new search direction is computed by solving another EQP given by the updated working set. If a CSP is reached, global optimality of the iterate is checked by analyzing the dual variables. If all dual variables are non-negative the global solution has been found, otherwise, a constraint which corresponds to a negative dual variable is removed from the working set and a new search direction is computed.

Algorithm 1 Primal Active-Set Method for solving (1).

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

1: **while** true **do**
2: Compute x_k^* and λ_k by solving EQP defined by $\mathcal{W}_k$
3: $p_k \leftarrow x_k^* - x_k$
4: **if** $p_k = 0$ **then**
5: **if** $\lambda_k \geq 0$ **then return** $x_k^*, \lambda_k, \mathcal{W}_k$
6: **else** $l \leftarrow \text{argmin}\ [\lambda_k]_i, \quad \mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{l\}$
7: **else**
8: $m \leftarrow \underset{i\in\mathcal{C}_k,\,[A]_i p_k>0}{\text{argmin}} \frac{[b+W\theta]_i - [A]_i x_k}{[A]_i p_k}$
9: $\alpha_k \leftarrow \min\{1, ([b+W\theta]_m - [A]_m x_k)/([A]_m p_k)\}$
10: $x_{k+1} \leftarrow x_k + \alpha_k p_k$
11: **if** $\alpha_k < 1$ **then** $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \cup \{m\}$
12: $k \leftarrow k + 1$

2.2 Certification of the primal active-set algorithm

We will use the certification method presented in Arnström and Axehill (2019), summarized briefly below, to certify how many iterations that are needed by Algorithm 1 to reach a certain, user-defined, level of suboptimality. The main idea of the certification method is to iteratively partition the parameter space depending on which working-set sequences that are produced by Algorithm 1 for different parameters. For each iteration, the parameter space will be partitioned depending on which constraints that are added to or removed from $\mathcal{W}$. In iteration k we will have the subregions Θ_k^i of the region of interest Θ_0 in the parameter space. For each region we also have the corresponding iterate $x_k^i(\theta)$ and working set $\mathcal{W}_k^i$. Instead of terminating when the iterates have converged completely to global optimality, as was done in Arnström and Axehill (2019), we will consider two different early termination criteria which guarantee a certain quality of the solution.

In this paper we are not concerned with determining a suboptimality which maintains a sufficient closed-loop performance for the controller. Instead, we assume that a suitable suboptimality level is given and consider how many iterations Algorithm 1 will have to perform until every possible iterate satisfy this suboptimality level.

3. EARLY TERMINATION OF THE PRIMAL ACTIVE-SET ALGORITHM

Two approaches for early termination of the primal active-set algorithm are considered. The first approach, which we will call the relaxation approach, is based on terminating when a set of relaxed KKT-conditions are satisfied, similar to what is done in Bemporad and Filippi (2003) for obtaining an approximate explicit solution. The second approach, which we will call the direct approach, analyzes the level of suboptimality in terms of the objective function directly and the algorithm is terminated when this value is sufficiently close to the value function. In practice, the relaxation method will lead to a modification of the check for global optimality at Line 5 in Algorithm 1 while the direct method will lead to the algorithm being stopped after a predetermined number of iterations has been performed.

3.1 Relaxing KKT-conditions

The KKT-conditions are necessary for optimality and for (1) they are also sufficient and explicitly given as

$$Hx + A^T\lambda = -f - f_\theta\theta, \tag{3a}$$
$$(Ax - b - W\theta)_i\lambda_i = 0, \qquad i = 1, \dots, m \tag{3b}$$
$$Ax \leq b + W\theta, \tag{3c}$$
$$\lambda \geq 0, \tag{3d}$$

where (3a) is called the stationarity condition, (3b) the complementarity condition, (3c) primal feasibility and (3d) dual feasibility.

A way to early terminate an iterative algorithm, such as the primal active-set algorithm of interest, is to relax these conditions and terminate when the relaxed conditions are met. In Bemporad and Filippi (2003) the KKT-conditions are relaxed by introducing slack in (3a), (3b) and (3d) and this is used to compute suboptimal explicit solutions of reduced complexity. Slack in the stationarity and complementarity condition are shown in Bemporad and Filippi (2003) to result in projections of lifted polytopes in the analysis while slack in the dual feasibility does not. In this paper we will consider a slack in the dual feasibility conditions, i.e., replacing (3d) with the condition $\lambda \geq -\epsilon_\lambda$ for a constant vector $\epsilon_\lambda > 0$.

Implementing this for early termination of Algorithm 1 is straightforward. The only modification needed is to replace $\lambda_k \geq 0$ with $\lambda_k \geq -\epsilon_\lambda$ at Line 5. This modification of Algorithm 1 calls for a modification to the certification method described in Arnström and Axehill (2019), however, the modification is minor. Since the dual variables are affine in the parameter, i.e., $\lambda = F^\lambda\theta + G^\lambda$ for some F^λ and G^λ, the partition of the parameter space originating from Line 5 in Algorithm 1 is done by half-planes, Arnström and Axehill (2019). Specifically, parameters in a region that satisfy global optimality, i.e., $\lambda \geq 0$, is restricted by the half-planes $\lambda = F^\lambda\theta + G^\lambda \geq 0 \Leftrightarrow -F^\lambda\theta \leq G^\lambda$. Adding a slack ϵ_λ modifies these half-planes to $\lambda = F^\lambda\theta + G^\lambda \geq -\epsilon_\lambda \Leftrightarrow -F^\lambda\theta \leq G^\lambda + \epsilon_\lambda$.

The dual slack will also affect the partitioning that is done when constraints are removed from the working set. For a constraint to be removed, its corresponding dual variable has to be negative. Hence, all parameters in the region that leads to the constraint corresponding to the i:th dual variable being removed from the working set are restricted by the half-plane $[\lambda]_i = [F^\lambda]_i\theta + [G^\lambda]_i < 0 \Leftrightarrow [F^\lambda]_i\theta < -[G^\lambda]_i$, where $[.]_i$ denotes the i:th row of a matrix. Adding a dual slack modifies this half-plane to $[\lambda]_i = [F^\lambda]_i\theta + [G^\lambda]_i < -\epsilon_\lambda \Leftrightarrow [F^\lambda]_i\theta < -\epsilon_\lambda - [G^\lambda]_i$.

Remark 2. The modifications of the half-planes are done linearly by the dual slack, i.e., the constant ϵ_λ enters exactly the same way as the parameters θ do. Hence, the dual slack could be considered as additional parameters and the effect on Algorithm 1 for a continuum of values of ϵ_λ can be analyzed in one shot, enabling an immediate analysis of the trade-off between numbers of iterations and suboptimality. This would result in extending the parameter space with the same number of dimensions as ϵ_λ. However, if all elements of ϵ_λ are chosen to be equal, the parameter space would only have to be extended with one additional dimension.

A limitation of the relaxation method is that it only allows early termination when the iterate is a constrained stationary point, i.e., when a constraint is removed in Algorithm 1. Thus, the early termination cannot be enforced in an arbitrary iteration. Also, note that in practice selecting a suitable ϵ_λ is problem dependent since the dual variables depend on the scaling of the constraints. Furthermore, the desired suboptimality is only stated implicitly by ϵ_λ. In practice an admissible suboptimality is often given, hence, this specification has to be translated into the choice of a suitable ϵ_λ.

In Bemporad and Filippi (2003), a bound on the difference between the value function and the objective function for the iterate when the iterate satisfies the relaxed KKT-conditions with dual slack ϵ_λ is derived

Lemma 3. (Bemporad and Filippi (2003) Lemma 4.3) If an iterate x_k satisfies (3a),(3b),(3c) and $\lambda_k \geq -\epsilon_\lambda$, the difference between the objective function and the value function is bounded by

$$J(x_k, \theta) - J(x^*, \theta) \leq \tfrac{1}{2}\epsilon_\lambda^T[A]_{\mathcal{W}_k}H^{-1}[A]_{\mathcal{W}_k}^T\epsilon_\lambda \triangleq \bar{\epsilon}_k, \tag{4}$$

where $[A]_{\mathcal{W}_k}$ denotes the rows of A indexed by $\mathcal{W}_k$.

This relationship can be used when choosing an ϵ_λ to obtain a suboptimal solution sufficiently close to the optimal one. Tight a posteriori bounds on the suboptimality level for a given ϵ_λ are also calculated in Bemporad and Filippi (2003) by comparing the objective function of the terminated iterates with the value function. A similar comparison between the objective function and the value function is done in the direct method, described below, but the comparison is done in all iterations.

3.2 Directly comparing objective function to value function

Another, more direct, termination criteria is to compute the difference between the objective function at the current iterate $J_k^i(\theta) \triangleq J(x_k^i, \theta)$ and the value function $J^*(\theta) \triangleq J(x^*, \theta)$, which is obtained by computing the optimal solution explicitly, and to terminate if this difference is small enough. Since the value function is not known a priori in practice, assuming the explicit solution is not available online, the implementation of the direct method online is to terminate Algorithm 1 after a predefined iteration limit has been passed, independent of the parameter value. This iteration limit is determined offline and guarantees that all iterates at termination, for all $\theta \in \Theta_0$, will be sufficiently close to the optimum. More concretely, using the proposed certification algorithm, the iteration limit can be obtained by calculating the difference between the current objective function values with the value function at each subregion. Subregions which result in this difference being below the specified suboptimality level is terminated, while the other subregions are partitioned further and the iterates are updated, corresponding to executing yet another iteration of Algorithm 1. When all regions have been terminated, the iteration limit is the maximum times a subregion has been partitioned.

Remark 4. Since Algorithm 1 defines a descent method, Nocedal and Wright (2006), terminating the algorithm in an iteration when the suboptimality is below a given threshold also guarantees that the quality of the subop-

timal solution in later iterations is no worse than the specified suboptimality.

Remark 5. In contrast to the relaxation method where the suboptimality is stated implicitly in terms of the dual slack ϵ_λ, the subotimality for the direct method is stated explicitly. Moreover, since the direct method leads to termination after a fixed number of iterations independently of the parameter value, the number of iterations is only reduced for regions leading to the worst-case number of iterations or more. This is not the case for the relaxation method which can lead to regions with relatively few iterations terminating early. However, from a real-time perspective it is only the worst-case number of iterations that is of interest.

Two approaches, one implicit and one explicit approach, can be considered when analyzing suboptimality by comparing J_k^i with J^*.

Implicit approach In the implicit approach, we consider a region Θ_k^i as an atomic unit when suboptimality is checked, similar to Axehill et al. (2014). We mark the current region Θ_k^i as optimal if the difference between the current objective function value J_k^i and the value function J^* is small enough for the *entire* region. Formally this can be expressed in terms of ϵ-optimality as

Definition 6. Θ_k^i is said to be ϵ-optimal if

$$J_k^i(\theta) - J^*(\theta) \leq \epsilon, \quad \forall \theta \in \Theta_k^i. \tag{5}$$

In this case, *all* parameters $\theta \in \Theta_k^i$ have to result in $J_k^i(\theta)$ being sufficiently close to $J^*(\theta)$. Thus, a single pathological parameter in the region dismisses the entire region from early termination. Deciding a suitable value of ϵ can be difficult since the scaling of J is problem dependent. Therefore, it is often more convenient to consider the relative error $(J_k^i(\theta) - J^*(\theta))/J^*(\theta)$.

Definition 7. Θ_k^i is said to be ϵ_r-optimal if

$$\frac{J_k^i(\theta) - J^*(\theta)}{J^*(\theta)} \leq \epsilon_r, \quad \forall \theta \in \Theta_k^i. \tag{6}$$

Remark 8. Note that ϵ_r-optimality is only well-defined when $J^*(\theta) > 0$. However, J can always be modified to satisfy this, without changing the optimizer, by adding terms which are constant or only depend on θ.

Explicit approach In the explicit approach, Θ_k^i is explicitly partitioned based on the difference between J_k^i and J_k^* in the ϵ-optimal region $\Theta_k^{i\,*} \triangleq \{\theta \in \Theta_k^i | J_k^i(\theta) - J^*(\theta) \leq \epsilon\}$ and in $\bar{\Theta}_k^{i\,*} \triangleq \{\theta \in \Theta_k^i | J_k^i(\theta) - J^*(\theta) > \epsilon\}$. If ϵ is the user-defined suboptimality tolerance, the algorithm can be terminated in Θ_k^* since its corresponding iterates are sufficiently close to the optimal solution while $\bar{\Theta}_k^*$ is further partitioned according to the certification algorithm. A similar partitioning can be done if the relative error is considered. Partitioning Θ_k^i into Θ_k^* and $\bar{\Theta}_k^*$ introduces nonlinear inequalities to the partitioning of the parameter space, since $J_k^i(\theta) - J^*(\theta)$ is nonlinear, which makes the certification algorithm less tractable and is not considered further in this paper.

Determining ϵ- and ϵ_r-optimality Deciding if a region Θ_k^i is ϵ-optimal can be done by solving an optimization problem as shown by the following lemma

Lemma 9. Given ϵ and a region Θ_k^i, let $\Delta^* J_k^i(\epsilon)$ be the optimal value to the optimization problem

$$\Delta^* J_k^i(\epsilon) \triangleq \underset{\theta \in \Theta_k^i}{\text{maximize}} \; J_k^i(\theta) - J^*(\theta) - \epsilon. \tag{7}$$

Then $\Delta^* J_k^i(\epsilon) \leq 0 \Leftrightarrow \Theta_k^i$ is ϵ-optimal.

Proof. If $\Delta^* J_k^i(\epsilon) \leq 0$ it holds that $J_k^i(\theta) - J^*(\theta) - \epsilon \leq 0, \forall \theta \in \Theta_k^i \Leftrightarrow J_k^i(\theta) - J^*(\theta) \leq \epsilon, \forall \theta \in \Theta_k^i$, which is the definition of Θ_k^i being ϵ-optimal. Now, assume instead that $\Delta^* J_k^i(\epsilon) > 0$, then $\exists \tilde{\theta} \in \Theta_k^i$ such that

$$J_k^i(\tilde{\theta}) - J^*(\tilde{\theta}) - \epsilon > 0 \Leftrightarrow J_k^i(\tilde{\theta}) - J^*(\tilde{\theta}) > \epsilon,$$

which makes it impossible for Θ_k^i to be ϵ-optimal since (5) has to hold for all $\theta \in \Theta_k^i$. □

As is shown in Arnström and Axehill (2019), after a constrained stationary point has been reached, the iterates for the algorithm considered will be affine in θ, resulting in $J_k^i(\theta)$ being a quadratic function of θ. Furthermore, it is well-known, Bemporad et al. (2002), that the value function of an mpQP is polyhedral piecewise quadratic (PWQ), i.e., there exist U_n, V_n and w_n such that

$$J^*(\theta) = \theta^T U_n \theta + V_n \theta + w_n, \quad \theta \in R_n, \tag{8}$$

where $\mathcal{R} = \{R_n\}_{n=1}^N$ is a polyhedral partition of Θ_0.

Thus, the objective function in (7) will be PWQ after a constrained stationary point has been reached. For the general case Θ_k^i is defined by both affine and quadratic constraints, making (7) a non-convex QCQP. However, the problem reduces to an indefinite QP when Θ_k^i is a polyhedron, which is shown in Arnström and Axehill (2019) to be the case when there is no parameter dependence in the constraints or if Algorithm 1 is started in a CSP.

Also in the case when Θ_k^i is not a polyhedron, a conservative result can be obtained by neglecting the quadratic constraints, resulting in (7) becoming an indefinite QP. The result becomes conservative since removing constraints makes Θ_k^i a subset of the parameters investigated for suboptimality. Hence, a parameter $\tilde{\theta}$ not in Θ_k^i but in the relaxed feasible set might result in $J_k^i(\tilde{\theta}) - J^*(\tilde{\theta}) > \epsilon$.

Similar to ϵ-optimality, ϵ_r-optimality of a region Θ_k^i can be determined by solving an optimization problem shown in the following lemma

Lemma 10. Assume that $J^*(\theta) > 0$. Given ϵ_r and Θ_k^i, let $\Delta_r^* J_k^i(\epsilon_r)$ be the optimal value to the optimization problem

$$\Delta_r^* J_k^i(\epsilon_r) \triangleq \underset{\theta \in \Theta_k^i}{\text{maximize}} \; J_k^i(\theta) - (1 + \epsilon_r) J^*(\theta). \tag{9}$$

Then $\Delta_r^* J_k^i(\epsilon_r) \leq 0 \Leftrightarrow \Theta_k^i$ is ϵ_r-optimal.

Proof. Similar to the proof of Lemma 9. □

The properties of (9) coincide with the ones described above for (7).

Remark 11. If (7) and (9) are solved with an optimization method which maintains upper and lower bounds, such as branch and bound, they do not have to be solved to optimality to determine ϵ- or ϵ_r-optimality. The solver can terminate immediately if a negative global upper bound is found, since this guarantees ϵ/ϵ_r-optimality. Likewise, a positive global lower bound guarantees that Θ_k^i is not ϵ/ϵ_r-optimal.

Algorithm 2 describes how to determine whether a region Θ_k^i is ϵ_r-optimal, provided that the explicit solution on Θ_k^i, consisting of the polyhedral partition $\mathcal{R} = \{R_n\}_{n=1}^N$, is available.

Algorithm 2 Check if the current region Θ_k^i is ϵ_r-optimal

1: **for** R_n in $\mathcal{R}$ **do**
2: **if** $R_n \cap \Theta_k^i \neq \emptyset$ **then**
3: $\Delta_r^* J_k^i(\epsilon_r) \leftarrow \underset{\theta \in R_n \cap \Theta_k^i}{\text{maximize}}\ J_k^i(\theta) - (1+\epsilon_r) J^*(\theta)$
4: **if** $\Delta_r^* J_k^i(\epsilon_r) > 0$ **then**
5: **return** false
6: **return** true

Remark 12. In Algorithm 2 the *entire* explicit solution is not necessarily needed, it only has to be computed on $\Theta_k^i \subseteq \Theta_0$. However, since Algorithm 2 will be applied to multiple regions in practice, which might be subsets of the same optimal region in the explicit solution, it might ultimately be cheaper to compute the entire Θ_0, or at least keep track of optimal regions that have been computed to avoid redundant computations.

3.3 Direct method with an underestimator of J^*

The direct method described in Algorithm 2 assumes that the explicit solution is available. However, instead of using $J^*(\theta)$ in the direct method, an underestimator $\underline{J}(\theta)$ of $J^*(\theta)$ on Θ_k^i can be used to give a conservative result, i.e., to determine an upper bound on the worst-case number of iterations needed that is not necessarily tight.

Definition 13. $\underline{J}(\theta)$ is said to be an underestimator of $J^*(\theta)$ on Θ_k^i if $\underline{J}(\theta) \leq J^*(\theta)$, $\forall \theta \in \Theta_k^i$.

Algorithm 3 describes a method for iteratively constructing an underestimator of $J^*(\theta)$ while at the same time testing for ϵ_r-optimality of Θ_k^i. The method shares similarities with the one presented in Jones and Morari (2010), where the double description (DD) method, Fukuda and Prodon (1995), is used to calculate approximate explicit solutions to mpLPs by iteratively constructing outer- and inner approximations of $J^*(\theta)$. Algorithm 3 also constructs an outer approximation of $J^*(\theta)$ iteratively but for our purpose, in contrast to Jones and Morari (2010), no inner approximation needs to be constructed.

Algorithm 3 Check if the current region Θ_k^i is ϵ_r-optimal while iteratively constructing an underestimator of J^*.

1: $\underline{J}(\theta) \leftarrow 0$
2: **for** $n < N_{\max}$ **do**
3: $\bar{\theta} \leftarrow \underset{\theta \in \Theta_k^i}{\text{argmax}}\ J_k^i(\theta) - (1+\epsilon_r)\underline{J}(\theta)$
4: **if** $J_k^i(\bar{\theta}) - (1+\epsilon_r)\underline{J}(\bar{\theta}) \leq 0$ **then**
5: **return** true
6: **else**
7: Solve the QP obtained by setting $\theta = \bar{\theta}$ in (1)
8: **if** $J_k^i(\bar{\theta}) - (1+\epsilon_r)J^*(\bar{\theta}) > 0$ **then**
9: **return** false
10: Compute the tangent plane $T_n(\theta)$ of $J^*(\theta)$ at $\theta = \bar{\theta}$
11: $\underline{J}(\theta) \leftarrow \max\{\underline{J}(\theta), T_n(\theta)\}$
12: **return** false

The algorithm adds a tangent plane of $J^*(\theta)$ at the parameter $\bar{\theta}$ to $\underline{J}(\theta)$, where $\bar{\theta}$ is the parameter that results in the largest difference between $J_k^i(\theta)$ and $J^*(\theta)$ on Θ_k^i. The constructed underestimator $\underline{J}$ will be a PWA function.

Remark 14. In Algorithm 3, J is assumed to be positive, making the relative error well-defined and 0 a universal underestimator. If a non-trivial underestimator for J^* is available one can replace 0 in the initialization of $\underline{J}$ at Line 1 with this underestimator. For example, such an underestimator is at hand from a previous execution of Algorithm 3 if the region failed the suboptimality test.

At iteration n in Algorithm 3, the optimization problem at Line 3 takes the form

$$\max_{\theta \in \Theta_k^i} \quad J_k^i(\theta) - (1+\epsilon_r)\max\{0, T_1(\theta), ..., T_n(\theta)\} \tag{10}$$

Which, with an epigraph formulation, can be cast as

$$\begin{aligned} \underset{\theta \in \Theta_k^i, t \geq 0}{\text{maximize}} \quad & J_k^i(\theta) - (1+\epsilon_r)t \\ \text{subject to} \quad & t \geq T_j(\theta), \quad j \in 1, ..., n. \end{aligned} \tag{11}$$

Since $T_j(\theta)$ are affine functions in θ, (11) is an indefinte QP when Θ_k^i is a polyhedron and $J_k^i(\theta)$ is a quadratic function.

Remark 15. The higher $N_{\max}$ is chosen, the better $\underline{J}$ approximates J^*, resulting in a less conservative result. However, this comes at a computational cost since (11) has to be solved in every iteration of Algorithm 3. Since the only difference in (11) between two iterations of Algorithm 3 is the addition of a single affine constraint, warm-starting is important for the efficiency of the algorithm.

4. EXAMPLES

To illustrate the use of the proposed methods, both the direct method and the relaxation method were tested on an mpQP with $n = 5$, $m = 10$ and $p = 8$ originating from the control of an inverted pendulum, for details about the example see Cimini and Bemporad (2019). The worst-case iteration bounds obtained by the methods presented in this paper for given suboptimality thresholds and dual slacks were compared with results from Monte-Carlo (MC) simulations, i.e., applying Algorithm 1 to an extensive amount of problems defined by samples drawn from Θ_0. The algorithm was started with the iterate in the origin and with an empty working set. For the inverted pendulum example 10^8 samples were used for the MC simulations. CPLEX was used to solve the indefinite QPs needed to be solved in the direct method and MPT 3.0, Herceg et al. (2013), was used to calculate the explicit solution.

4.1 Relaxing KKT-conditions

The certification algorithm, Arnström and Axehill (2019), with the relaxation method described in Section 3.1 used for early termination was tested for different dual slacks ϵ_λ. The number of iterations $N_{\text{iter}}^{\max}$ performed by Algorithm 1 to satisfy the KKT-conditions, relaxed with different dual slacks ϵ_λ, is shown in Table 1. Note that $\epsilon_\lambda = 0$ introduces no dual slack and hence leads to Algorithm 1 computing the optimal solution. The maximum absolute error bound $\max \bar{\epsilon}_k$, obtained by taking the maximum of (4) over all regions in the final partition given by the certification algorithm, is also shown. In Section 4.2,

these bounds are compared to the bounds obtained by the direct method. As expected, increasing the dual slack, i.e., further relaxing the optimality conditions, leads to a decrease in the number of iterations performed by the active-set algorithm. Table 1 also shows that increasing the dual slack increases the bound on the absolute error $\max \bar{\epsilon}_k$ which is reasonable since terminated iterates should be further away from optimality because Algorithm 1 is a descent method and fewer iterations are performed.

Table 1. Maximum number of iterations $N_{\text{iter}}^{\max}$ required by Algorithm 1 to satisfy the relaxed KKT-conditions with dual slack ϵ_λ.

ϵ_λ	0	1	5	10	15	20
$N_{\text{iter}}^{\max}$	26	25	23	21	17	14
$\max \bar{\epsilon}_k$	0	1.3	35	140	315	561

4.2 Direct method

The certification algorithm, Arnström and Axehill (2019), with the direct method introduced in Section 3.2 used for early termination was also applied to the problem for five different relative errors ϵ_r and five different absolute errors ϵ. To be able to compare the direct method with the relaxation method, the absolute errors were chosen as the bounds $\max \bar{\epsilon}_k$ obtained when using the relaxation method, found in Table 1. The worst-case number of iterations $N_{\text{iter}}^{\max}$ required by Algorithm 1 to compute a solution with an error below the specified relative suboptimality ϵ_r and absolute suboptimality ϵ is shown in Table 2, giving the explicit relationship between admissible suboptimality and required iterations for the given problem.

Table 2. $N_{\text{iter}}^{\max}$ number of iterations needed by Algorithm 1 to attain an error below ϵ and ϵ_r.

ϵ_r	0	0.0001	0.001	0.01	0.1
$N_{\text{iter}}^{\max}$	26	25	22	20	14
ϵ	0	1.3	35	140	561
$N_{\text{iter}}^{\max}$	26	23	17	14	14

In Figure 1, the maximum relative error observed in MC simulations are compared with the bounds computed by the direct method. As expected, the number of iterations required by Algorithm 1 to reach a certain level of suboptimality determined by the certification algorithm is always greater than or equal to the MC simulation bounds.

The discrepancy between the result of the certification algorithm and the MC simulations is a consequence of that MC simulations are unable to guarantee full coverage of the parameter space. A concrete illustration of this weakness can be seen in Figure 1 where the conclusion of the MC simulation would have been that 19 iterations are needed to obtain $\epsilon_r \leq 0.01$ while the direct method concludes that 20 iterations are needed. The direct method is able to prove the existence of, and compute, a parameter which results in a relative error which is larger than 0.01 after 19 iterations. Hence, the conclusion made by the MC solution can be shown to be optimistic by executing Algorithm 1 with this parameter.

Remark 16. Explicitly, a parameter proving the MC simulation to be optimistic can be found as the maximizing

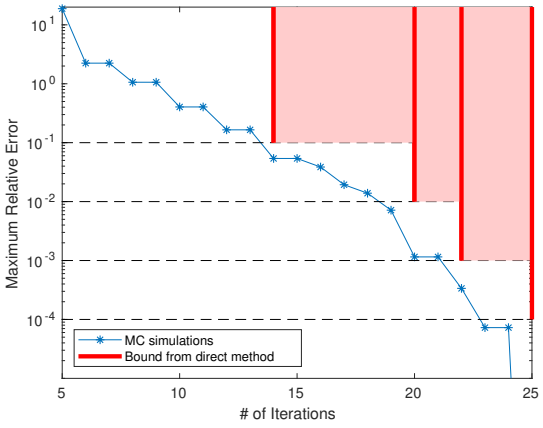

Fig. 1. Maximum relative error obtained through MC simulations compared with bounds obtained by the direct method.

argument of (9) when the optimal value is positive for a region which has required 19 iterations.

MC simulations were also considered for analyzing the absolute error. This result is shown in Figure 2 where it is compared with the bounds given by the direct method and by the relaxation method. The bound from (4) is illustrated to be conservative for the considered example.

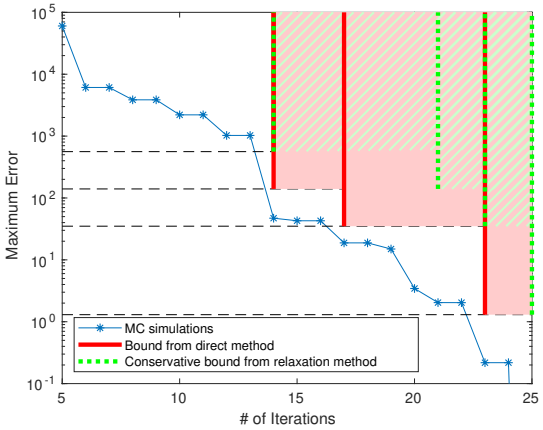

Fig. 2. Maximum absolute error obtained through MC simulations compared with bounds obtained by the direct method and the relaxation method.

To further validate the direct method, another 1000 mpQPs with $n = 5$, $m = 30$, and $p = 2$ were randomly generated. p was chosen small to make sure the probability of an MC method covering most of the parameter space was relatively high without having to pick intractably many samples. The mpQPs had the form

$$\begin{aligned} &H = I, && f = 0, && f_\theta \sim \mathcal{N}(0,1) \\ &A \sim \mathcal{N}(0,1), && b \sim \mathcal{U}([0,10]), && W = 0, \end{aligned} \tag{12}$$

where the stochastic parts should be interpreted elementwise, e.g., each element of f_θ is drawn from $\mathcal{N}(0,1)$ and is independent of the rest of the elements. The parameter set considered was $\Theta_0 = \{\theta | -10 \leq \theta \leq 10\}$. Furthermore, a term $\frac{1}{2}\theta^T H_\theta \theta$ with $H_\theta = 10I$ was added to the objective function and f_θ was redrawn if the resulting J was not positive for all $\theta \in \Theta_0$ and feasible x. This redraw was done to ensure the relative error being well-defined. To

start the algorithm, the origin was picked as the starting iterate, which is feasible by construction in the examples, and the starting working set was empty.

Remark 17. The Hessians H were picked as the identity matrix for convenience but mpQPs defined by (12) also capture generic positive-definite Hessians since such problems can be transformed into mpQPs with $H = I$ by using a Cholesky factorization and a change of variables.

The direct method was used with $\epsilon_r = 0.01$ on the randomly generated problems and the obtained iteration bounds were compared with MC simulations using $4 \cdot 10^4$ samples. The results are summarized in Table 3.

Table 3. Comparison of the maximum number of iterations $N_{\text{cert}}^{\max}$ and $N_{\text{MC}}^{\max}$ needed by Algorithm 1 to obtain a relative error of 1% determined by the certification method and by MC simulations, respectively, for 1000 randomly generated mpQPs.

$N_{\text{cert}}^{\max} \geq N_{\text{MC}}^{\max}$	$N_{\text{cert}}^{\max} = N_{\text{MC}}^{\max}$	$N_{\text{cert}}^{\max} > N_{\text{MC}}^{\max}$
100%	90.8%	9.2%

Again, the iteration bound determined by the certification method is greater than or equal to the iteration bounds obtained by the MC simulation for all generated mpQPs, illustrating the validity of the method. For the 9.2% of problems resulting in $N_{\text{cert}}^{\max} > N_{\text{MC}}^{\max}$, Algorithm 1 was applied to the worst-case parameters computed by the certification method. The outcome from solving these additional problems proved that the results from the initial MC simulations were optimistic, due to the parameter space not being covered sufficiently well, rather than the certification method being conservative.

Moreover, the iteration bounds for $\epsilon_r = 0.01$ determined by the certification method were compared with the iteration bounds when the mpQPs were solved to optimality, i.e., when $\epsilon_r = 0$. The average difference was 5.29 iterations and the maximum and minimum difference was 16 and 0 iterations, respectively. The effectiveness of terminating Algorithm 1 early is, hence, problem dependent and the proposed certification method introduces a novel possibility to determine this efficiency exactly for a given problem.

5. CONCLUSION

In this paper we have extended the certification method presented in Arnström and Axehill (2019) to exactly certify the complexity of a primal active-set algorithm for quadratic programs when it is prematurely terminated. The certification method allows for an exact analysis of how many iterations will be needed if the solution is allowed to be suboptimal within a prespecified user-defined bound. Hence, the method determines exactly how many iterations can be saved when the requirements on the solution quality are relaxed. This allows the active-set algorithm to be terminated early, which decreases the required computational resources on-line, with guarantees on worst-case number of iterations needed to obtain a prespecified quality of the solution, extending the applicability of the primal active-set algorithm in real-time applications.

REFERENCES

Arnström, D. and Axehill, D. (2019). Exact complexity certification of a standard primal active-set method for quadratic programming. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, 4317–4324. doi: 10.1109/CDC40024.2019.9029370.

Axehill, D., Besselmann, T., Raimondo, D.M., and Morari, M. (2014). A parametric branch and bound approach to suboptimal explicit hybrid MPC. *Automatica*, 50(1), 240–246.

Bemporad, A. and Filippi, C. (2003). Suboptimal explicit receding horizon control via approximate multiparametric quadratic programming. *Journal of Optimization Theory and Applications*, 117(1), 9–38.

Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, E.N. (2002). The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1), 3–20.

Cimini, G. and Bemporad, A. (2017). Exact complexity certification of active-set methods for quadratic programming. *IEEE Transactions on Automatic Control*, 62, 6094–6109. doi:10.1109/TAC.2017.2696742.

Cimini, G. and Bemporad, A. (2019). Complexity and convergence certification of a block principal pivoting method for box-constrained quadratic programs. *Automatica*, 100, 29–37. doi: 10.1016/j.automatica.2018.10.050.

Fukuda, K. and Prodon, A. (1995). Double description method revisited. In *Franco-Japanese and Franco-Chinese Conference on Combinatorics and Computer Science*, 91–111. Springer.

Goldfarb, D. and Idnani, A. (1983). A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27, 1–33. doi: 10.1007/BF02591962.

Graichen, K. and Kugi, A. (2010). Stability and incremental improvement of suboptimal MPC without terminal constraints. *IEEE Transactions on Automatic Control*, 55(11), 2576–2580.

Herceg, M., Kvasnica, M., Jones, C., and Morari, M. (2013). Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, 502–510. Zürich, Switzerland.

Jones, C.N. and Morari, M. (2010). Polytopic approximation of explicit model predictive controllers. *IEEE Transactions on Automatic Control*, 55(11), 2542–2553.

Kunisch, K. and Rendl, F. (2003). An infeasible active set method for quadratic problems with simple bounds. *SIAM Journal on Optimization*, 14, 35–52. doi: 10.1137/S1052623400376135.

Michalska, H. and Mayne, D.Q. (1993). Robust receding horizon control of constrained nonlinear systems. *IEEE transactions on automatic control*, 38(11), 1623–1633.

Nocedal, J. and Wright, S. (2006). *Numerical Optimization*. Springer Science & Business Media.

Scokaert, P.O., Mayne, D.Q., and Rawlings, J.B. (1999). Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44(3), 648–654.

Zeilinger, M.N., Jones, C.N., and Morari, M. (2011). Real-time suboptimal model predictive control using a combination of explicit MPC and online optimization. *IEEE Transactions on Automatic Control*, 56, 1524–1534. doi:10.1109/TAC.2011.2108450.