Complexity Certification of Proximal-Point Methods for Numerically Stable Quadratic Programming

Daniel Arnström, Alberto Bemporad and Daniel Axehill

Arnström, D., Bemporad, A., Axehill, D., (2021), Complexity Certification of Proximal-Point Methods for Numerically Stable Quadratic Programming, *IEEE CONTROL SYSTEMS LETTERS*, 5(4), 1381-1386. https://doi.org/10.1109/LCSYS.2020.3038035

Complexity Certification of Proximal-Point Methods for Numerically Stable Quadratic Programming

Daniel Arnström, Alberto Bemporad and Daniel Axehill

***Abstract*— When solving a quadratic program (QP), one can improve the numerical stability of any QP solver by performing proximal-point outer iterations, resulting in solving a sequence of better conditioned QPs. In this paper we present a method which, for a given multi-parametric quadratic program (mpQP) and any polyhedral set of parameters, determines which sequences of QPs will have to be solved when using outer proximal-point iterations. By knowing this sequence, bounds on the worst-case complexity of the method can be obtained, which is of importance in, for example, real-time model predictive control (MPC) applications. Moreover, we combine the proposed method with previous work on complexity certification for active-set methods to obtain a more detailed certification of the proximal-point method's complexity, namely the total number of inner iterations.**

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

I. Introduction

In model predictive control (MPC) an optimization problem must be solved at each time step. When MPC is used on embedded systems in real-time applications, it is important that the optimization methods used for solving these problems are efficient, reliable, and simple. In linear MPC, where the dynamics of the plant is modeled as linear, the optimization problems in question are often quadratic programs (QPs) which depend on the state of the plant and the set-point, making them multi-parametric quadratic programs (mpQPs).

A class of methods which are suitable for solving QPs originating from MPC problems is active-set methods [1][2][3, Ch. 16.5][4][5]. In particular, the active-set method in [1] is simple to implement and has been shown to be competitive with state-of-the-art solvers for QPs encountered in real-time MPC. A limitation of the method is, however, that it can be unreliable for ill-conditioned problems, which lead to the extension presented in [6]. Therein, the QP method is amended with outer proximal-point iterations, resulting in solving a sequence of better conditioned QPs which largely improves the numerical stability of the method.

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

A. Bemporad is with the IMT School for Advanced Studies Lucca, Lucca, Italy alberto.bemporad@imtlucca.it

Another recent development for improving the reliability of active-set QP methods are complexity certification methods which, given an mpQP, determines *exactly* which sequence of active-set changes will occur during solution for any polyhedral set of parameters. In particular, complexity certification methods for the active-set QP methods in [1]–[4] have been presented in [7]–[10], respectively. Furthermore, a general complexity certification framework which encapsulates [7]–[9] has been presented in [11].

In this paper we present a complexity certification method which extends certification methods covered by [11] to handle outer proximal-point iterations, akin to how [6] extends [1]. By using the proposed method, the reliability of the QP method in [6] can be increased further through guarantees on worst-case computational complexity.

Given an mpQP, the method first identifies which sequence of QPs will be solved when the proximal-point method is used. Then, for each such QP, the complexity certification methods covered by [11] are used to determine which sequence of systems of linear equations will be solved when an active-set method is used as an inner solver. Hence, the presented method certifies, but is not limited to, the QP method in [6]. More broadly, the proposed method certifies the worst-case number of outer proximal-point iterations when *any* inner QP solver is used. If there, in addition, exists a complexity method for the inner solver, similar to the one covered in [11], the proposed method can also determine bounds on the total number of inner QP iterations and, ultimately, floating-point operations.

The main contribution is, hence, a method which can be used to determine worst-case bounds on iterations and, ultimately, floating-point operations for QP solvers which use outer proximal-point iterations. Such bounds are important when, e.g., MPC is used in real-time applications.

II. Problem formulation and notation

It is well-known, c.f. [12], that linear MPC problems can be cast as mpQPs of the form

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

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 and the set-point. The objective function $J(x,\theta)$ is defined by $H \in \mathbb{S}^n_+, f \in \mathbb{R}^n$, and $f_\theta \in \mathbb{R}^{n \times p}$, while the feasible set is

defined by $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$, and $W \in \mathbb{R}^{m\times p}$. Here we will consider the case in which Θ_0 is a polytope.

Remark 1: All subsequent results also hold when linear *equality* constraints are present in (1), but we limit our discussion to inequality constraints for a clean presentation.

Problem (1) can also be recast as $\min_x q_\theta(x)$ with

$$q_\theta(x) \triangleq \left(J(x,\theta) + \sum_{i=1}^{m} \mathcal{I}^+([b]_i + [W]_i\theta - [A]_i x) \right), \quad (2)$$

where $\mathcal{I}^+$ denotes the indicator function of the nonnegative real-axis, i.e., $\mathcal{I}^+(z) = 0$ if $z \geq 0$ and ∞ otherwise. Recasting problem (1) into the unconstrained optimization problem $\min_x q_\theta(x)$ will be used to motivate the proximal-point method introduced in Section III.

A. Notation

For a polyhedron $R = \{\theta \in \mathbb{R}^p : A\theta \leq b\}$ we denote its interior $\mathring{R} \triangleq \{\theta \in \mathbb{R}^p : A\theta < b\}$. A collection of objects, e.g., polyhedra, $\{R^1, R^2, \ldots, R^{N-1}, R^N\}$ is often written as $\{R^i\}_{i=1}^N$ or more compactly $\{R^i\}_i$ if its cardinality is unimportant. For any given integer $N \in \mathbb{N}$ we call $\mathbb{N}_n$ the set of integers from 1 to N.

III. The proximal-point method

In this paper we are interested in analyzing a proximal-point method which solves (1) by iteratively applying the so-called *proximal operator* of $q_\theta(x)$.

Definition 1: For a convex function $g : X \to (-\infty, \infty]$, its *proximal operator* $\mathbf{prox}_g : X \to X$ is the mapping

$$\mathbf{prox}_g(v) \triangleq \underset{x\in X}{\operatorname{argmin}} \left(g(x) + \frac{1}{2}||x - v||_2^2 \right), \forall v \in X. \quad (3)$$

Many interesting interpretations and properties of the proximal operator are given in [13], where also interpretations of the proximal-point method, soon to be introduced, are given.

Remark 2: $\mathbf{prox}_g$ is in general a set-valued function, but if g is proper, closed, and convex it outpus a singleton.

Here, we will, in particular, make use of the proximal operator for γq_θ, given by

$$\mathbf{prox}_{\gamma q_\theta}(v) \triangleq \underset{x}{\operatorname{argmin}} \left(q_\theta(x) + \frac{1}{2\gamma}||x - v||_2^2 \right), \quad (4)$$

where $\gamma > 0$ can be interpreted as a regularization parameter since $\mathbf{prox}_{\gamma q_\theta}(v)$ is the optimum of the regularized QP

$$\begin{aligned} \min_x \quad & \frac{1}{2}x^T \left(H + \tfrac{1}{\gamma} I \right) x + \left(f + f_\theta \theta - \tfrac{1}{\gamma} v \right)^T x \\ \text{s.t.} \quad & Ax \leq b + W\theta. \end{aligned} \quad (5)$$

Importantly, (5) is better conditioned than (1) since a positive diagonal matrix is added to the Hessian. The smaller γ is, the better conditioned the Hessian in (5) is.

Our interest in $\mathbf{prox}_{\gamma q_\theta}$ originates from the relation between its fixed points and the optimizers of (1).

Lemma 1: For $H \succeq 0$ and $\gamma > 0$, let x_k be updated by

$$x_{k+1} = \mathbf{prox}_{\gamma q_\theta}(x_k). \quad (6)$$

Then, given a starting iterate $x_0(\theta)$ and $\theta \in \Theta_0$, $\lim_{k\to\infty} x_k(\theta) = x^*(\theta)$, where $x^*(\theta)$ is the optimizer of (1).

Proof: See [6, Corollary 1]. ■

Hence, for a given θ, problem (1) can be solved by iteratively applying $\mathbf{prox}_{\gamma q_\theta}$, corresponding to solving a regularized QP problem, until a fixed point has been reached within a prescribed tolerance, as is summarized in Algorithm 1. The algorithm is a *proximal-point method* and terminates if the change between two iterates is less than or equal to $\nu > 0$ in some norm, meaning that a fixed point of $\mathbf{prox}_{\gamma q_\theta}$ has approximately been reached.

Algorithm 1 Proximal-point method for solving (1)

```
x_0 ← starting guess, θ given, k ← 0.
while true do
    x_{k+1} ← prox_{γq_θ}(x_k)
    if ||x_{k+1} − x_k|| ≤ ν then
        return x_{k+1}
    k ← k + 1
```

Algorithm 1 has a convergence rate of $\mathcal{O}(1/k)$. More concretely, it can be shown (see, e.g., Theorem 10.28 in [14]) that its iterates satisfy

$$J_k(\theta) - J^*(\theta) \leq \frac{1}{2\gamma k}||x_0(\theta) - x^*(\theta)||_2^2, \quad (7)$$

with $J_k(\theta) \triangleq J(x_k(\theta), \theta)$ and $J^*(\theta) \triangleq J(x^*(\theta), \theta)$.

Remark 3: An inherent trade-off when picking γ can be seen in (5) and (7): a smaller γ leads to better conditioned QP subproblems to solve in each iteration of Algorithm 1, but a smaller γ also leads to slower convergence, i.e., more iterations of Algorithm 1 must be executed.

To summarize, the advantage of using a proximal-point method for solving (1) is that a sequence of *well-conditioned* QPs can be solved to solve the original QP, which might be *ill-conditioned*. This allows solvers which behave particularly bad for ill-conditioned problems, or solvers which demand the problem to be strictly convex, to be used to solve (1).

A. Evaluating the proximal operator

Using the proximal-point method might seem expensive since we have to solve numerous QPs instead of a single one. However, as is shown in [6], performing outer proximal-point iterations instead of trying to solve the QP directly can, in addition to improving numerical stability, improve the worst-case CPU time. This improvement can partly be explained by the subproblems being better conditioned than the original QP problem, making them easier to solve. Furthermore, only the linear term in the objective function of (5) changes in the regularized QP subproblems, i.e., the solution between subproblems will be similar. Hence, the solution from the previous subproblem can be used as a warm start in the following QP so that, possibly required, matrix factorizations can be reused which, often, reduces the computational complexity significantly.

A class of QP methods that can be efficiently warm-started is active-set methods. We will therefore mainly focus on using active-set methods for solving the QP needed to evaluate $\mathbf{prox}_{q_\theta}$. However, most of the results are applicable to the case

in which other QP methods, such as interior-point methods, are used to evaluate $\mathbf{prox}_{\gamma q_\theta}$.

IV. Parametric Behaviour

We are now interested in how Algorithm 1 behaves for different values of the parameter θ, for example, how the number of iterations k depends on θ. When analyzing the parametric properties of the algorithm, the following definitions will prove useful.

Definition 2: A collection of polyhedra $\{R^i\}_{i=1}^N$ is said to be a *polyhedral partition* of Θ if $\cup_i R^i = \Theta$ and if $\mathring{R}^i \cap \mathring{R}^j = \emptyset, \forall i \neq j$.

Definition 3: A function $x(\theta) : \Theta \to X$ is *piecewise affine* (PWA) on Θ if there exists a polyhedral partition $\{R^i\}_{i=1}^N$ of Θ and if, $\forall i \in \mathbb{N}_N$, $x(\theta) = G^i\theta + h^i$, $\forall \theta \in R^i$, with $G^i \in \mathbb{R}^{n\times p}$ and $h^i \in \mathbb{R}^n$. If in addition $G^i = 0$, $\forall i \in \mathbb{N}_N$, $x(\theta)$ is *piecewise constant* (PWC) on Θ.

The main property which allows for the parametric behaviour of Algorithm 1 to be parametrically tractable stems from $\mathbf{prox}_{\gamma q_\theta}$ retaining PWA structures, formalized in the following lemma.

Lemma 2: If $v(\theta)$ is PWA on Θ, then $\mathbf{prox}_{\gamma q_\theta}(v(\theta))$ is also PWA on θ.

Proof: Since $v(\theta)$ is PWA, $v(\theta) = G^i\theta + h^i, \forall \theta \in R^i$ for some polyhedral partition $\{R^i\}_{i=1}^N$ of Θ. By considering $\theta \in R^i$ and inserting the affine expression of $v(\theta)$ into (5), $\mathbf{prox}_{\gamma q_\theta}(v(\theta))$ is the solution to

$$\begin{aligned} \min_x \quad & \frac{1}{2}x^T\left(H + \tfrac{1}{\gamma}I\right)x + \left(f - \tfrac{1}{\gamma}h^i + (f_\theta - \tfrac{1}{\gamma}G^i)\theta\right)^T x \\ \text{s.t.} \quad & Ax \leq b + W\theta, \quad \theta \in R^i, \end{aligned} \tag{8}$$

which is an mpQP and, hence, has a PWA solution, see, e.g., Theorem 4 in [12]. Let this solution be denoted as $G^{i,j}\theta + h^{i,j}, \forall \theta \in R^{i,j}$, where $\{R^{i,j}\}_{j=1}^{N^i}$ is a polyhedral partition of R^i. What remains to prove is that $\{R^{i,j}\}_{i,j}$ is a polyhedral partition of Θ. First we note that $\cup_{i,j} R^{i,j} = \cup_i R^i = \Theta$, where the first equality follows from $\{R^{i,j}\}_j$ being a polyhedral partition of R^i for each fixed i and the last equality follows from $\{R^i\}_i$ being a polyhedral partition of Θ. Next, we have that $\mathring{R}^{i,j} \cap \mathring{R}^{\tilde{i},\tilde{j}} = \emptyset$ for $i \neq \tilde{i}$ and $j \neq \tilde{j}$, which follows from $\mathring{R}^{i,j} \cap \mathring{R}^{i,\tilde{j}} = \emptyset$ for $j \neq \tilde{j}$ and any i since $\{R^{i,j}\}_{i,j}$ is a polyhedral partition of R^i. Furthermore, $\mathring{R}^{i,j} \cap \mathring{R}^{\tilde{i},k} = \emptyset$ for $i \neq \tilde{i}$ since $R^{i,j} \subseteq R^i$ and $\mathring{R}^i \cap \mathring{R}^{\tilde{i}} = \emptyset$ since $\{R^i\}_i$ is a polyhedral partition of Θ. ■

Figure 1 illustrates how the proximal operator of q_θ propagates polyhedral partitions.

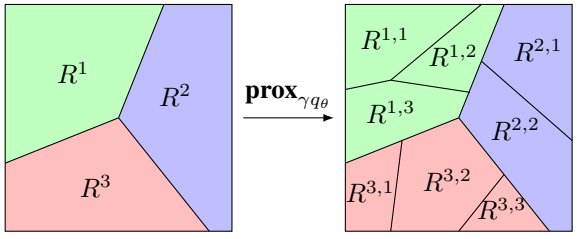


Fig. 1: Example of polyhedral partitions before and after $\mathbf{prox}_{\gamma q_\theta}$ is applied to a PWA function.

Now, since Algorithm 1 iteratively applies $\mathbf{prox}_{\gamma q_\theta}$, Lemma 2 can be applied iteratively to analyze how its iterates change parametrically.

Lemma 3: Let x_0 be PWA on Θ, then $x_k(\theta)$ in Algorithm 1 will be PWA on Θ at all iterations k.

Proof: Directly follows from Lemma 2 and by induction, since the iterates of Algorithm 1 are updated by iteratively applying $\mathbf{prox}_{\gamma q_\theta}$. ■

So far we have shown that the parametric behaviour of Algorithm 1 will be similar on polyhedral regions of the parameter space if the starting iterate is PWA. We are now interested in analyzing the termination criterion $||x_{k+1} - x_k|| \leq \nu$ parametrically, which evidently depends on which norm is used. The following lemma shows that if $||\cdot||_\infty$ is used, a polyhedral partitioning of the parameter space can be retained.

Lemma 4: Let $\Theta_k^* \subseteq \Theta_0$ be the set of parameters in Θ_0 which results in the termination criterion $||x_k - x_{k-1}|| \leq \nu$ being satisfied in iteration k of Algorithm 1, i.e.,

$$\Theta_k^* \triangleq \{\theta \in \Theta_0 : ||x_k(\theta) - x_{k-1}(\theta)|| \leq \nu\}. \tag{9}$$

Furthermore, let x_0 be PWA on Θ_0 and let the $||\cdot||_\infty$ be used for the termination criterion. Then there exist two collections of polyhedra $\{\Psi^i\}_i$ and $\{\Phi^i\}_i$ such that

(i) $\Theta_k^* = \cup_i \Psi_k^i$,
(ii) $\Theta_0 \setminus \Theta_k^* = \cup_i \Phi_k^i$.

Proof: First, we show that the difference $\Delta x_k(\theta) \triangleq x_k(\theta) - x_{k-1}(\theta)$ is PWA. From Lemma 3 we have that x_{k-1} is PWA for any k, i.e., $x_{k-1} = G_{k-1}^i\theta + h_{k-1}^i, \forall \theta \in R^i$, where $\{R^i\}_i$ is a polyhedral partition of Θ_0. Furthermore, by using the same notation as in the proof of Lemma 2, $x_k = G_k^{i,j}\theta + h_k^{i,j}, \forall \theta \in R^{i,j}$, where $\{R^{i,j}\}_{i,j}$ is a polyhedral partition of Θ_0 and $\{R^{i,j}\}_j$ is a polyhedral partition of R^i. The difference then becomes

$$\begin{aligned} \Delta x_k(\theta) &= G_k^{i,j}\theta + h_k^{i,j} - \left(G_{k-1}^i\theta + h_{k-1}^i\right), \text{ if } \theta \in R^{i,j} \\ &= \Delta G_k^{i,j}\theta + \Delta h_k^{i,j}, \text{ if } \theta \in R^{i,j} \end{aligned} \tag{10}$$

where we have used $R^{i,j} \subseteq R^i$ in the first equality and have defined $\Delta G_k^{i,j} \triangleq G_k^{i,j} - G_{k-1}^i$ and $\Delta h_k^{i,j} = h_k^{i,j} - h_{k-1}^i$ in the second equality.

With $\Delta x_k(\theta)$ being PWA established, we analyze the termination criterion $||\Delta x_k(\theta)||_\infty \leq \nu$.

Let $n(i,j) : \mathbb{N}_{N^i} \times \mathbb{N}_{N^{i,j}} \to \mathbb{N}_{N^i N^{i,j}}$ be any bijective mapping and define

$$\begin{aligned} \Psi_k^{n(i,j)} \triangleq & \Theta_k^* \cap R^{i,j} \\ = & \{\theta \in R^{i,j} : ||\Delta G_k^{i,j}\theta + h_k^{i,j}||_\infty \leq \nu\}, \end{aligned} \tag{11}$$

which is the set of all parameters in $R^{i,j}$ which satisfy the termination criterion after k iterations. Ψ_k^n is a polyhedron since it is defined as an intersection between two polyhedra. Taking the union of all Ψ_k^n gives

$$\begin{aligned} \cup_n \Psi_k^n = & \cup_{i,j} \left(\Theta_k^* \cap R^{i,j}\right) = \Theta_k^* \cap \left(\cup_{i,j} R^{i,j}\right) \\ = & \Theta_k^* \cap \Theta_0 = \Theta_k^*, \end{aligned} \tag{12}$$

where the second equality follows from intersections distributing over unions, the third equality follows from $\{R^{i,j}\}_{i,j}$ being a polyhedral partition of Θ_0, and the final equality

follows from $\Theta_k^* \subseteq \Theta_0$. This proves (i) and, once (i) has been established, (ii) follows directly from Proposition 6 in [15], which states that the difference between a polyhedron and a collection of polyhedra can also be represented by a collection of polyhedra. ■

Remark 4: Θ_k^* in Lemma 4 is not necessarily a convex set since its composing polyhedra might be disconnected.

A direct consequence of Lemma 4 is that the number of iterations k will be structured if the $||\cdot||_\infty$ is used in the termination criterion, namely $k(\theta)$ will be PWC.

Corollary 1: If x_0 is PWA on Θ_0 and $||.||_\infty$ is used in the termination criterion of Algorithm 1, the number of iterations, $k(\theta) : \Theta_0 \to \mathbb{N}$, needed by Algorithm 1 will be PWC on Θ_0.

Proof: Directly follows from Lemma 4. Explicitly, $k(\theta) = \tilde{k}, \forall \theta \in \Theta_{\tilde{k}}^* \backslash \Theta_{\tilde{k}-1}^*$, where $\Theta_{\tilde{k}}^*$ is defined by (9). Again, since $\Theta_{\tilde{k}}^*$ can be represented as a collection of polyhedra, Proposition 6 in [15] can be used to show that $\Theta_{\tilde{k}}^* \setminus \Theta_{\tilde{k}-1}^*$ also can be represented as a collection of polyhedra. ■

With these parametric properties established, we set out to devise a method which certifies the worst-case complexity of Algorithm 1. Note that it can be immediately shown that the above results also hold when $\|.\|_1$ is used in Step 4 of Algorithm 1.

V. Complexity certification

The idea of the complexity certification method we propose is to execute iterations of Algorithm 1 parametrically, which will partition the parameter space into polyhedral regions. By using an affine starting iterate $x_0(\theta)$, the iterates at iteration k, $x_k(\theta)$, will be affine functions in θ, as shown by Lemma 3. Each region is defined by a tuple (Θ, k, G, h), where $\Theta \subseteq \Theta_0$ is the parameter region, k is the number of iterations that has been executed for the region to reach its current state and, finally, the iterate on the region is given by the affine mapping $x_k(\theta) = G\theta + h$. Executing another iteration of Algorithm 1 for a region given by the tuple (Θ, k, G, h) corresponds to solving an mpQP on the form

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^T \left(H + \tfrac{1}{\gamma} I\right) x + \left(f - \tfrac{1}{\gamma} h + (f_\theta - \tfrac{1}{\gamma} G)\theta\right)^T x \\ \text{s.t.} \quad & Ax \leq b + W\theta, \quad \theta \in \Theta, \end{aligned} \tag{13}$$

which will generate the tuples $\{(\Theta^i, k+1, G^i, h^i)\}_i$, where the explicit solution to this mpQP is given by

$$x = G^i\theta + h^i, \ \forall \theta \in \Theta^i.$$

Next, the termination criterion is imposed parametrically. Even though Lemma 4 and Corollary 1 can be used to track *exactly* how many proximal-point iterations are needed by Algorithm 1 to converge to the solution of (1), the cuts made by the termination criterion increase the complexity of the partition. These cuts are not necessary, however, if only a worst-case analysis is of interest, e.g., when $\max_{\theta\in\Theta_0} k(\theta)$ is of interest, which is often the case for real-time MPC. Instead of partitioning a parameter region Θ into regions which have converged and which have not, the parameter region can be treated as an atomic unit and if

$$\max_{\theta\in\Theta} ||x_{k+1}(\theta) - x_k(\theta)|| \leq \nu, \tag{14}$$

all of the parameters in Θ satisfies the termination criterion.

Remark 5: A similar approach of treating regions as an atomic unit instead of explicitly partitioning the parameter space by a termination criterion was used in [16], which considered complexity certification of an early-terminating primal active-set method.

Remark 6: In addition to circumventing cuts imposed by the termination criterion, analyzing the termination by (14) allows for other norms (such as the 2-norm) to be used in the termination criterion while retaining a polyhedral partition, which would not be the case if the partitioning were to be analyzed *exactly* for these norms.

The certification method is summarized in Algorithm 2. Two stacks of tuples are maintained throughout the process; S stores tuples corresponding to regions for which there exist parameters which do not yet satisfy the termination criterion, while S^* contains tuples corresponding to regions for which all parameters satisfy the termination criterion. The procedure **solvempQP**(G, h, Θ) solves the mpQP in (13). For a survey of methods for solving mpQPs, c.f. [17].

Algorithm 2 Certification of outer proximal-point iterations

1: Push $(\Theta_0, 0, G_0, h_0)$ to S
2: **while** $S \neq \emptyset$ **do**
3: Pop (Θ, k, G, h) from S
4: $\{(\Theta^i, G^i, h^i)\}_{i=1}^N \leftarrow$ **solvempQP**(Θ, G, h)
5: **for** $i \in \{1, \ldots, N\}$ **do**
6: $\Delta \leftarrow \max_{\theta\in\Theta^i} ||(G^i - G)\theta + (h^i - h)||$
7: **if** $\Delta \leq \nu$ **then**
8: Push $(\Theta^i, k+1, G^i, h^i)$ to S^*
9: **else**
10: Push $(\Theta^i, k+1, G^i, h^i)$ to S
11: **return** S^*

Since Algorithm 2 uses (14) instead of explicit partitioning according to the termination criterion, we want to ensure that it terminates in finite time. This is ensured by the sequence of iterates produced by Algorithm 1 converging.

Lemma 5: Assume that $x_0(\theta)$ is given and let $\{x_i(\theta)\}_i$ be the corresponding sequence of iterates produced by Algorithm 1. Then, for any subset $\Psi \subseteq \Theta_0$, $\exists K \in \mathbb{N}$: $\max_{\theta\in\Psi} ||x_{k+1}(\theta) - x_k(\theta)|| \leq \nu$, $\forall k \geq K$.

Proof: Follows directly from Lemma 1 and Cauchy's convergence criterion. ■

With the finite termination of Algorithm 2 ensured, we state the main result of this paper: For a given mpQP, Algorithm 2 can be used to determine the worst-case iteration complexity of Algorithm 1.

Theorem 1: Let $\{(\Theta^i, \hat{k}^i, G^i, h^i)\}_i$ be the collection of tuples in S^* when Algorithm 2 has terminated and let $k(\theta) : \Theta_0 \to \mathbb{N}$ be the number of iterations needed by Algorithm 1 to terminate, as a function of θ. Then

(i) $k(\theta) \leq \hat{k}^i$, $\forall \theta \in \Theta^i$ and $\forall i$
(ii) $\max_i \hat{k}^i = \max_{\theta\in\Theta_0} k(\theta)$

Proof: (i) Consider any tuple $(\Theta^i, \hat{k}^i, G^i, h^i) \in S^*$. From Steps 6-8 of Algorithm 2, we then have that

$\max_{\theta\in\Theta^i} ||x_{\hat{k}^i}(\theta) - x_{\hat{k}^i-1}(\theta)|| \leq \nu \implies \Theta^i \subseteq \Theta^*_{\hat{k}}$, which in turn, from (9), implies that $k(\theta) \leq \hat{k}^i, \forall\theta \in \Theta^i$.

(ii) From (i), it follows that

$$\max_i \hat{k}^i \geq \max_{\theta\in\Theta_0} k(\theta), \tag{15}$$

since $\cup_i \Theta^i = \Theta_0$. Now, to prove (ii) we want to prove that this bound is tight, i.e., $\exists\tilde{\theta} \in \Theta_0 : k(\tilde{\theta}) = \max_i \hat{k}^i$. Let $j \in \text{argmax}_i \hat{k}^i$ and let $\tilde{\Theta}^j \subseteq \Theta_0$ be the parent of Θ^j in Algorithm 2, i.e., the region which was partitioned to obtain Θ^j, which implies that $\Theta^j \subseteq \tilde{\Theta}^j$. Since $\tilde{\Theta}^j$ was partitioned into Θ^j, $\max_{\theta\in\tilde{\Theta}^j} ||x_{\hat{k}^j-1}(\theta) - x_{\hat{k}^j-2}|| > \nu$, otherwise $\tilde{\Theta}^j$ would have been added to S^*. This implies from (9) that $\tilde{\Theta}^j \nsubseteq \Theta^*_{\hat{k}^j-1}$, which in turn implies that $\exists\tilde{\theta} \in \tilde{\Theta}^j : k(\tilde{\theta}) > \hat{k}^j - 1$. Combining this lower bound with the upper bound in (15) gives

$$\max_i \hat{k}^i - 1 < k(\tilde{\theta}) \leq \max_i \hat{k}^i \Leftrightarrow k(\tilde{\theta}^i) = \max_i \hat{k}^i, \tag{16}$$

where we have recalled that $\hat{k}^j = \max_i \hat{k}^i$. ■

Thus, Algorithm 2 provides upper bounds on the number of iterations needed by Algorithm 1 to terminate and a tight upper bound on the worst-case number of iterations, for any region of interest in the parameter space.

Up to this point, we have not made any assumptions on how $\textbf{prox}_{\gamma q_\theta}$ is evaluated in Algorithm 1. Next, we consider the case when an active-set method is used as an inner QP solver, resulting in more fine-grained certificates of the complexity, such as total number of iterations performed by the inner solver.

A. Certifying inner iterations

As was mentioned in Section III-A, the choice of inner solver is important for the proximal-point method to be practical. In particular, it is critical that the inner solver can be warm started, which makes active-set methods good candidates. For some active-set methods there exist algorithms which, for a given mpQP, can certify their complexity [7][8][9][10][11]. These certification methods determine which sequence of active sets is visited for any parameter or interest, allowing the worst-case number of iterations and floating-point operations to be determined. In brief, the parameter space is partitioned into polyhedral regions and for each region the worst-case number of iterations, the optimal solution as an affine function of θ and the optimal active-set, i.e., the constraints which hold with equality at the solution, are determined.

To get more fine-grained certificates on the complexity of Algorithm 1, we consider the case in which any active-set method covered by [11] is used as an inner solver. By using the corresponding complexity certification method for the inner solver, one can obtain, in addition to bounds on the outer iteration, bounds on the total number of inner iterations or even the number of floating-point operations needed by Algorithm 1. The idea is to use Algorithm 1 to determine which sequence of mpQPs will be solved for different regions of the set Θ_0 and for each such mpQP apply the complexity certification for the active-set method to determine the sequence of active-set changes made by the inner solver.

This is done by extending **solvempQP**(Θ, G, h) in Algorithm 2 with the procedure **certInner**$(\Theta, G, h, \mathcal{A})$ which applies the complexity certification method for the active-set method on the mpQP in (13) and outputs the tuples $\{(\Theta^i, \kappa^i, G^i, h^i, \mathcal{A}^i)\}_i$, where $G^i\theta + h^i, \forall\theta \in \Theta^i$ is, as before, the explicit solution of (13), κ^i is a measure of the complexity, which can either be the number of iterations or number of floating-point operations, to solve (13) when the active-set method is warm started with $\mathcal{A}$ and, finally, $\mathcal{A}^i$ is the optimal active set at region Θ^i for (13). The final algorithm is given by Algorithm 3.

Algorithm 3 Certification of total inner complexity

```
Push (Θ_0, 0, G_0, h_0, A_0) to S
while S ≠ ∅ do
    Pop (Θ, κ, G, h, A) from S
    {Θ^i, κ^i, G^i, h^i, A^i}_{i=1}^N ← certInner(Θ, G, h, A)
    for i ∈ {1, ..., N} do
        Δ ← max_{θ∈Θ^i} ||(G^i − G)θ + (h^i − h)||
        if Δ ≤ ν then
            Push (Θ^i, κ + κ^i, G^i, h^i, A^i) to S*
        else
            Push (Θ^i, κ + κ^i, G^i, h^i, A^i) to S
return S*
```

Remark 7: Even though our main focus herein is on active-set methods, Algorithm 3 works for any QP method, if there exists a complexity certification method for it.

Remark 8: The partition provided by **certInner** will be finer compared to the partition of the explicit solution. If one wants to reduce the number of regions, all regions with the same final active set can be merged into one region and the maximal κ^i can be picked for the merged region. However, the finer partition from **certInner** is not necessarily a drawback since it allows for more regions to be terminated by (14).

VI. Numerical Example

The proposed certification method is illustrated on a small-scale mpQP originating from MPC applied to control a double integrator. The problem matrices are given by

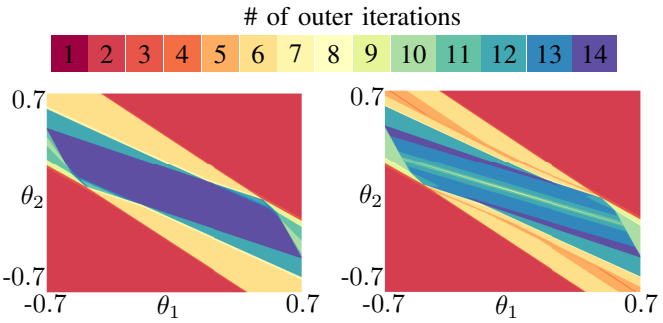
$$H = \begin{pmatrix} 5.3 & 1.8 & 8.0 \\ 1.8 & 4.4 & 6.0 \\ 8.0 & 6.0 & 22.9 \end{pmatrix} \cdot 10^{-2}, f_\theta = \begin{pmatrix} 0.36 & 0.50 & -0.50 & -0.01 \\ 0.30 & 0.41 & -0.41 & 0 \\ 0.85 & 1.0 & -1.0 & 0 \end{pmatrix}$$

$$f = \mathbf{0}_{3\times 1},\ A = \begin{pmatrix} I_3 \\ -I_3 \end{pmatrix},\ b = \mathbf{1}_{6\times 1},\ W = \mathbf{0}_{6\times 4},$$

where I_n denotes the n-dimensional identity matrix and $\mathbf{0}_{n\times m}$ and $\mathbf{1}_{n\times m}$ denote $n \times m$ matrices with all elements equal to 0 or 1, respectively. The regularization parameter was set to $\gamma = 10^2$ and for the termination criterion the $||\cdot||_\infty$ was used with a tolerance $\nu = 10^{-6}$. Moreover, the starting iterate was chosen as the origin, i.e., $x_0 = (0, 0, 0)^T$.

Gurobi [18] was used for solving the nonconvex optimization problems in Step 6 of Algorithm 2 and 3.

A representative two-dimensional slice of the final partition of Θ_0 with the estimated number of outer iterations $\hat{k}$ produced by Algorithm 2 is shown in Figure 2, together with results from



(a) Certification (b) Simulation

Fig. 2: Number of outer iterations determined by: (a) Applying the Algorithm 2 on the mpQP; (b) Executing Algorithm 1 over a two-dimensional grid in the parameter space. The slice is given by $\theta_3 = \theta_4 = 0$. As is stated in Theorem 1, the outer iterations of Algorithm 1, shown in (b), is upper bounded by the result from Algorithm 2, shown in (a).

(a) $\gamma = 10^2$ (b) $\gamma = 2 \cdot 10^2$ (c) $\gamma = 2 \cdot 10^3$

Fig. 3: Number of outer iterations determined by Algorithm 2 for different values of the regularization parameter γ.

simulation. These simulation results were obtained by sampling Θ_0 and, for each parameter sample, applying Algorithm 1 to the resulting QP. The execution time for the complexity certification was 42 seconds for a MATLAB implementation of the algorithm executed on an Intel 2.7 GHz i7-7500U CPU.

The properties of Algorithm 2 which were proven in Theorem 1 can be seen in Figure 2, namely that $\hat{k}(\theta)$ is an upper bound on $k(\theta)$ for all parameters in Θ_0 and that the worst-case outer iteration bound is tight, which for the example is 14 outer iterations.

One can also see that the complexity of the exact partition produced is more complex than the one produced by Algorithm 2, highlighting the advantage of using (14) instead of tracking the termination exactly by explicit partitioning of the parameter space through the termination criterion.

Figure 3 shows partitions generated by Algorithm 2 for different values of the regularization parameter γ, where a larger γ results in fewer outer iterations (in accordance with Remark 3). The execution time for Algorithm 2 for values of γ in Figure 3a, 3b, and 3c were, 42 seconds, 22 seconds, and 2 seconds, respectively.

VII. Conclusion

In this paper we have proposed a complexity certification method for when outer proximal-point iterations are performed to improve the numerical stability of QP solvers. Given an mpQP, the method determines exactly which set of regularized QPs will be solved when the proximal-point method is used, which can be applied to determine the worst-case number of outer proximal-point iterations. We have also shown how the proposed method can be combined with previous work on complexity certification methods for active-set methods to determine worst-case bounds on the total number of inner iterations and floating-point operations. The ability of the proposed method to determine the exact worst-case number of outer iterations performed by the proximal-point method was illustrated on an mpQP originating from controlling a double integrator through MPC.

References

[1] 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.

[2] 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.

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

[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 numerically stable solver for positive semidefinite quadratic programs based on nonnegative least squares," *IEEE Transactions on Automatic Control*, vol. 63, no. 2, pp. 525–531, 2017.

[7] D. Arnström, A. Bemporad, and D. Axehill, "Exact complexity certification of a nonnegative least-squares method for quadratic programming," *IEEE Control Systems Letters*, vol. 4, no. 4, pp. 1036–1041, 2020.

[8] 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.

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

[10] G. Cimini and A. Bemporad, "Complexity and convergence certification of a block principal pivoting method for box-constrained quadratic programs," *Automatica*, vol. 100, pp. 29–37, 02 2019.

[11] D. Arnström and D. Axehill, "A unifying complexity certification framework for active-set methods for convex quadratic programming," *ArXiv e-prints*, Mar. 2020.

[12] 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.

[13] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[14] A. Beck, *First-order methods in optimization*. SIAM, 2017.

[15] S. V. Rakovic, E. C. Kerrigan, D. Q. Mayne, and J. Lygeros, "Reachability analysis of discrete-time systems with disturbances," *IEEE Transactions on Automatic Control*, vol. 51, no. 4, pp. 546–561, 2006.

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

[17] A. Bemporad, "Explicit model predictive control," in *Encyclopedia of Systems and Control*, J. Baillieul and T. Samad, Eds. London: Springer London, 2019, pp. 1–7.

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