Exact Complexity Certification of Active-Set Methods for Quadratic Programming

Gionata Cimini, *Student Member, IEEE*, and Alberto Bemporad, *Fellow, IEEE*

***Abstract*—Active-set methods are recognized to often outperform other methods in terms of speed and solution accuracy when solving small-size quadratic programming (QP) problems, making them very attractive in embedded linear model predictive control (MPC) applications. A drawback of active-set methods is the lack of tight bounds on the worst-case number of iterations, a fundamental requirement for their implementation in a real-time system. Extensive simulation campaigns provide an indication of the expected worst-case computation load, but not a complete guarantee. This paper solves such a certification problem by proposing an algorithm to compute the exact bound on the maximum number of iterations and floating point operations required by a state-of-the-art dual active-set QP solver. The algorithm is applicable to a given QP problem whose linear term of the cost function and right-hand side of the constraints depend linearly on a vector of parameters, as in the case of linear MPC. In addition, a new solver is presented that combines explicit and implicit MPC ideas, guaranteeing improvements of the worst-case computation time. The ability of the approach to exactly quantify memory and worst-case computation requirements is tested on a few MPC examples, also highlighting when online optimization should be preferred to explicit MPC.**

***Index Terms*—Active-set methods, complexity certification, linear model predictive control (MPC), quadratic programming (QP).**

I. Introduction

In THE last decade, embedded model predictive control (MPC) has become increasingly important in many engineering fields, like automotive, aerospace, and power systems [1]–[5]. The control problems that arise in such contexts usually involve a high sampling frequency and/or low-power computing boards [6], [7]. MPC optimizes the closed-loop response of multivariable systems subject to state and input constraints [8]. A constrained finite-horizon optimal control problem must be solved at each time step, making the online implementation of MPC in embedded boards a challenge.

Manuscript received December 17, 2016; accepted April 3, 2017. Date of publication April 24, 2017; date of current version December 1, 2017. Recommended by Associate Editor M. Alamir. (*Corresponding author: Alberto Bemporad.)*

G. Cimini is with the Università Politecnica delle Marche, Ancona 60121, Italy, and also with ODYS S.r.l., Lucca 55100, Italy (e-mail: gionata.cimini@odys.it).

A. Bemporad is with the IMT School for Advanced Studies Lucca, Lucca 55100, Italy (e-mail: alberto.bemporad@imtlucca.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2017.2696742

The standard formulation of MPC based on a linear time-invariant model, quadratic performance index, and linear constraints can be cast into a quadratic program (QP) [9], whose linear term of the cost function and right-hand side of the constraints depend linearly on a vector of parameters, such as states and reference signals. Explicit MPC drastically reduces the computational load by presolving offline the QP and converting the MPC law into a continuous and piecewise affine (PWA) function of the parameter vector [9], [10]. Unfortunately, explicit MPC is manageable only when the dimensions of the QP are very small, due to the high memory requirements for storing the resulting polyhedral partitions and gains defining the control law.

When the complexity of explicit MPC becomes prohibitive, one must solve the QP problem online, which clearly requires to embed the QP solver in the real-time control board [6]. The literature on QP solvers is extremely rich, among several popular approaches, we mention interior-point methods [11], gradient projection methods [12], [13], the alternating directions method of multipliers [14], [15], and active-set methods [7], [16]–[18]. For the small-size QP's that arise in embedded MPC, active-set methods are usually the best in terms of speed and accuracy [4], [19], [20]. Moreover, the solution can be achieved with very high accuracy and *in a finite number of iterations*, even in the case of ill-conditioned problems, using single-precision arithmetics, and without the need of preconditioning the matrices of the problem [21].

The question is how large such a finite number of iterations can be. In embedded MPC, it is of paramount importance to certify that the QP solver always provides the solution within a certain execution time, which must be lower than the sampling interval [22]. Unfortunately, in spite of the empirical evidence of their efficiency, a theoretical computational complexity of active-set methods that is useful in practice is not available [23]–[25]. Practical experience suggests that they are polynomial algorithms on average [24], [25], but as shown in the famous Klee–Minty problem they can even display exponential worst-case number of iterations on contrived problems [26]. This aspect could prevent their use in embedded applications, in favor of interior-point methods, for which a theoretical polynomial iteration bound is available [11], [27], or gradient projection methods, for which tight worst-case bounds can be often proved [13].

This paper proposes an approach to certify the worst-case execution time of a dual active-set QP solver *exactly*. We focus on dual active-set methods as, despite the drawback of

0018-9286 © 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

infeasible subiterates, they usually require less iterations than primal methods and do not need the so-called *Phase I* computation to get a starting primal-feasible solution [16], [17]. We first demonstrate that the number of iterations of a dual active-set method depends, in a PWA fashion, on the vector of parameters perturbing the linear term of the quadratic cost function and the right-hand side of the constraints. Therefore, any given bounded operating range of parameters can be split into polyhedral regions that share the same finite number of iterations needed to solve the corresponding QP problem.

The technique proposed in the paper builds upon the machinery employed in multiparametric quadratic programming (mpQP) to get explicit MPC solutions, where the parameter space is divided into polyhedral regions that share the same optimal active set. The idea is close in spirit with the multiparametric analysis developed in [28] of the simplex method for linear programming (LP) problems. A similar approach was used in [29] to prove that the step-length and the current iterate of an active-set method for LPs are PWA functions of the parameters. Here, we consider how "parametric" steps of the dual active-set method for solving strictly convex QP's by GI [16], referred to as GI algorithm in the paper, propagate in the parameter space during iterations. The proposed algorithm is able to quantify the worst-case number of iterations and the corresponding flops.

The certification algorithm can be also used to improve the speed of execution of embedded MPC in several ways, described in the paper. First, the algorithm can be extended immediately to other dual active-set methods for QP's, including range-space or null-space-based methods [18], [30], so that one can choose the best solver for a given MPC controller based on the one that has the least worst-case execution time. Second, as dual active-set solvers have the degree of freedom of selecting which constraint enters in the current active set at each iteration [31], different selection rules for choosing the violated constraint can reduce the number of iterations and flops [32]. The best selection rule depends on the particular problem at hand and can be singled out by the certification algorithm.

Finally, we propose a novel technique that combines "implicit" (online QP) and explicit MPC, and that we refer to as worst-case partial enumeration (WCPE)-MPC. Several techniques that combine explicit and implicit MPC are available in the literature [7], [29], [33]–[36], however they focus on the average behavior of the solver, or do not guarantee optimality/feasibility of the solution. Based on the certification algorithm developed in the paper, the proposed WCPE-MPC approach certifies the worst-case improvement exactly, therefore allowing one to select the best tradeoff between memory occupancy and speed enhancement. Furthermore, despite some accelerating techniques proposed in the literature, WCPE-MPC always provides the optimal solution of the QP problem.

It is worth noticing that the type of dual QP solver, the violated constraint selection rule, and whether implicit or explicit MPC should be used are issues that are commonly tackled empirically, based on experience or on the dimension of the QP problem. Instead, given a QP problem, the use of the certification algorithm guarantees the optimality of the choice in terms of worst-case number of flops and memory occupancy. As the methods proposed in this paper rely on recursively partitioning the parameter space into polyhedral cells, their applicability are limited to MPC problems of moderate size, in general to those for which the explicit solution is computable offline, although not necessarily applicable online because of its excessive memory/CPU requirements.

The paper is organized as follows. Section II recalls the standard MPC formulation and the resulting multiparametric QP. Section III details the dual active-set algorithm GI, and demonstrates the linearity of the steps with respect to the parameters [16]. Section IV describes the proposed algorithm for computing the worst-case complexity certification. Section V presents the methods to choose the best dual active-set solver, the best violated constraint selection rule, and the WCPE-MPC strategy. Numerical results on four MPC problems are presented in Section VI, and Section VII concludes the paper.

II. MPC AND QP PROBLEM

In this paper, we consider the following linear MPC formulation:

$$\min_{\Delta u} \sum_{i=1}^{N_p} \|W_y(y_{k+i|k} - r(k))\|_2^2 +$$

$$\sum_{h=0}^{N_u-1} \|W_u(u_{k+h|k} - u_r(k))\|_2^2 + \|W_{\Delta u}\Delta u_{k+h|k}\|_2^2 \quad (1a)$$

$$\text{s.t. } x_{k+i+1|k} = Ax_{k+i|k} + Bu_{k+i|k},\ x_{k|k} = x(k) \quad (1b)$$

$$y_{k+i+1|k} = Cx_{k+i+1|k} \quad (1c)$$

$$\Delta u_{k+i|k} = u_{k+i|k} - u_{k+i-1|k} \quad (1d)$$

$$\Delta u_{k+N_u+j|k} = 0,\ j = 0, 1, \ldots, N_p - N_u - 1 \quad (1e)$$

$$u_{k+h|k} \in \mathbb{U}, \Delta u_{k+h|k} \in \mathbb{D},\ y_{k+i|k} \in \mathbb{Y} \quad (1f)$$

$$i = 0, \ldots, N_p - 1,\ h = 0, \ldots, N_u - 1$$

where $\mathbb{R}^n$ denotes the set of real vectors of dimension n, $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$, and $y \in \mathbb{R}^{n_y}$, N_p is the prediction horizon, N_u is the control horizon, W_y, W_u, and $W_{\Delta u}$ are square weight matrices, with $W_{\Delta u}$ nonsingular, the subscript $_{k+i|k}$ denotes the prediction at time $k+i$ based on the information available at time k, $x(k)$ is the current state, $\Delta u_{k+i|k}$ is the vector of the input increments, with $u_{k-1|k} = u(k-1)$, and $\mathbb{U}$, $\mathbb{D}$, and $\mathbb{Y}$ are polyhedral sets defining the constraints on inputs, input increments, and outputs, respectively.

Problem (1) can be cast into the QP

$$\begin{aligned} \min_z \quad & f(z) \triangleq \frac{1}{2} z'Hz + \theta'F'z \\ \text{s.t.} \quad & g(z) \triangleq Gz - W\theta - w \leq 0 \end{aligned} \quad (2)$$

where $\theta \in \Theta$ is the vector of parameters of dimension $n_\theta = n_y + n_x + n_u$ and $\Theta \subset \mathbb{R}^{n_\theta}$ is a bounded set of interest, $z \in \mathbb{R}^n$ is the vector of optimization variables, $H \in \mathbb{R}^{n \times n}$ is a symmetric and positive definite matrix, $F \in \mathbb{R}^{n \times n_\theta}$, $G \in \mathbb{R}^{m \times n}$,

$W \in \mathbb{R}^{m \times n_\theta}$, $w \in \mathbb{R}^m$, and given a matrix $A \in \mathbb{R}^{n \times m}$, A' denotes its transpose.

Solving problem (2) online to compute the optimal control move $u(k) = u_{k|k}$ for a given θ is often the main obstacle in embedded MPC applications, as the associated computational requirements may be excessive for a given embedded board [4]. Explicit MPC presolves problem (2) offline via mpQP [9], [10], however the required memory occupancy of the solution typically grows exponentially with the number of imposed linear constraints, limiting the approach to small MPC problems (few inputs and short control horizon, few constraints). Embedding a QP solver is therefore the only solution when explicit MPC is too complex. Next section recalls the generalities of active-set methods and focuses on the dual active-set method of Goldfarb and Idnani (GI) [16] for solving strictly convex QP's of the form (2).

III. Active-Set Solvers

A. Preliminaries

For a given finite subset $\mathcal{I} \subset \mathbb{N}$ of positive integers, let $\#\mathcal{I}$ denote its cardinality. For a vector $a \in \mathbb{R}^n$, a_i denotes the ith entry of a, $a_\mathcal{I}$ the subvector obtained by collecting the entries a_i for all $i \in \mathcal{I}$. For a matrix $A \in \mathbb{R}^{n \times m}$, A_i denotes the ith row of A, $A_\mathcal{I}$ the submatrix of A obtained by collecting the rows A_i for all $i \in \mathcal{I}$.

Definition 1: Given the QP problem (2) and $\bar{z} \in \mathbb{R}^n$, the constraint $G_i z \leq W_i\theta + w_i$ is *active* at $\bar{z}$ if the equality $G_i\bar{z} = W_i\theta + w_i$ holds, otherwise it is *inactive*.

Definition 2: Given the QP problem (2) and $\bar{z} \in \mathbb{R}^n$, the *active set* $\mathcal{A}(\bar{z})$ is

$$\mathcal{A}(\bar{z}) = \{i \in \mathcal{K} \,|\, G_i\bar{z} = W_i\theta + w_i\} \tag{3}$$

where $\mathcal{K} = \{1, \dots, m\}$ is the set of constraint indices.

We will denote by $\mathcal{I}(\bar{z}) = \{i \in \mathcal{K} \,|\, G_i\bar{z} < W_i\theta + w_i\}$ the set of inactive constraints at $\bar{z}$, where clearly $\mathcal{I} = \mathcal{K} \setminus \mathcal{A}$. The idea behind active-set methods is to iteratively make steps toward the solution by solving a reduced problem with only the equality constraints in the current active set, which we indicate as $\mathcal{A}^q$ for the qth iteration of the solver. The algorithm terminates when the current active set equals the optimal one, namely $\mathcal{A}^q \equiv \mathcal{A}^*$. At every iteration q a violated constraint p is added to the active set, and all the constraints in $\mathcal{A}^{q-1}$ that prevent p to be active are dropped from it. These constraints are said to be *blocking*.

Let $\mathcal{V}^q$ be the set of violated constraints at iteration q. After selecting a new violated constraint $p^q \in \mathcal{V}^q \subseteq \mathcal{I}^q$, the step size and direction are derived by solving the equality constrained QP problem

$$\begin{aligned} \min_z \quad & \frac{1}{2} z^{q\prime} H z^q + \theta' F' z^q \\ \text{s.t.} \quad & G_{\mathcal{A}^q} z^q = W_{\mathcal{A}^q}\theta + w_{\mathcal{A}^q} \end{aligned} \tag{4}$$

where the updated working set is $\mathcal{A}^q = \{\mathcal{A}^{q-1} \setminus \mathcal{B}^q\} \cup p^q$, with $\mathcal{B}^q$ the set of blocking constraints removed at iteration q from $\mathcal{A}^{q-1}$. The primal–dual pair (z^q, π^q) is the solution of problem (4) if it solves the Karush–Kuhn–Tucker (KKT) system

$$\underbrace{\begin{bmatrix} H & G'_{\mathcal{A}^q} \\ G_{\mathcal{A}^q} & 0 \end{bmatrix}}_{\text{KKT}(\mathcal{A}^q)} \begin{bmatrix} z^q \\ \pi^q \end{bmatrix} = \begin{bmatrix} -F\theta \\ W_{\mathcal{A}^q}\theta + w_{\mathcal{A}^q} \end{bmatrix} \tag{5}$$

with $\text{KKT}(\mathcal{A}^q)$ the so-called *KKT matrix* and π the vector of dual variables. *Primal feasible* active-set methods iteratively solve the KKT system (5) starting from a primal feasible z^0, until dual feasibility is reached [18], [31], and maintain primal feasibility in the subiterates. Phase I is needed to find a primal feasible z^0. On the contrary, *dual feasible* active-set solvers start from a dual feasible point, which is readily available (e.g., $\pi^0 = 0$, $z^0 = -H^{-1}F\theta$, where for a square matrix $A \in \mathbb{R}^{n \times n}$, A^{-1} denotes the inverse of A if it exists) and iterate (5) to reach primal feasibility, maintaining dual feasibility in the subiterates [16], [17]. Despite the drawback of infeasible subiterates, dual active-set methods usually require less iterations and do not need a Phase I computation [16], [17]. Active-set methods are further categorized depending on how they solve (5). As far as the direct solution of KKT matrix is concerned, *range-space* or *null-space* methods are the most common strategies [16], [18], [37], where the name refers to the subspaces they are working with.

B. Goldfarb–Idnani Algorithm

In this section, we briefly recall the GI algorithm [16], so as to analyze in Section IV how its iterations for solving (2) depend on θ. Being a dual, range-space solver, it is particularly suited for QPs that arise from MPC problems, where typically $m > n$. Range-space approaches solve (5) by the explicit inversion of $\text{KKT}(\mathcal{A})$. Equation (5) is iteratively solved starting from $\mathcal{A}^{q-1}$ and dropping the blocking constraints one-by-one, until constraint p can be added to the active set without violating dual feasibility. Note that here we denote by q the number of QP iterations, which is increased each time a constraint is added to the active set. In the sequel, we denote instead by j the index that is increased each time a constraint is either added or dropped from the active set, therefore accounting also for subiterations of the algorithm, where clearly $j \geq q$.

The explicit inversion $\text{KKT}(\mathcal{A}^j)^{-1}$ is

$$\text{KKT}(\mathcal{A}^j)^{-1} = \begin{bmatrix} I & -H^{-1}G'_{\mathcal{A}^j} \\ 0 & I \end{bmatrix} \begin{bmatrix} H^{-1} & 0 \\ 0 & \mathcal{S}_H^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -G_{\mathcal{A}^j}H^{-1} & I \end{bmatrix} \tag{6}$$

where $\mathcal{S}_H = -G_{\mathcal{A}^j}H^{-1}G'_{\mathcal{A}^j}$ is the Schur complement of H. From (6) we define the two operators

$$G^*_{\mathcal{A}^j} = \left(G_{\mathcal{A}^j}H^{-1}G'_{\mathcal{A}^j}\right)^{-1} G_{\mathcal{A}^j}H^{-1} \tag{7a}$$

$$\mathcal{H}^j = H^{-1}\left(I - G'_{\mathcal{A}^j}G^*_{\mathcal{A}^j}\right) \tag{7b}$$

where $G^*_{\mathcal{A}^j}$ is the Moore–Penrose pseudoinverse of $G_{\mathcal{A}^j}$ under the transformation $y = H^{1/2}x$ and $\mathcal{H}$ is the inverse Hessian operator in the active-set space.

The core of the GI algorithm, summarized in Algorithm 1, is the method to find a new vector z^j and the set of active

Algorithm 1: Goldfarb-Idnani (GI) QP Solver, [16].

Input: Matrices H, F, G, W, w, θ defining problem (2).

1: Compute the Cholesky factorization $LL' = H$;
2: $J_2^0 \leftarrow L^{-T}$; $z^0 \leftarrow -J_2^0 J_2^{0\prime} F\theta$; $\mathcal{A}^0 \leftarrow \emptyset$; $q, j \leftarrow 0$;
3: choose

$$p \leftarrow \begin{cases} 0 \text{ if } g_i(z^j) \le 0,\ \forall i \in \mathcal{I} \\ h \in \mathcal{I} \,|\, g_h(z^j) > 0 \text{ otherwise;} \end{cases} \quad \text{(A.1)}$$

4: **if** $p = 0$ **then return** $\mathcal{A}^* \leftarrow \mathcal{A}^j$; **end if**;
5: $q \leftarrow q + 1$, $\pi_p^j \leftarrow 0$;
6: $j \leftarrow j + 1$, $\Delta z^j \leftarrow -J_2^j (J_2^j)' G_p'$;
7: **if** $\#\mathcal{A}^j > 0$ **then** $\Delta\pi^j \leftarrow -R^{-1,j}(J_1^j)' G_p'$; **end if**;
8: **if** $\Delta\pi^j \ge 0$ **or** $\#\mathcal{A}^j = 0$ **then** $\alpha_1^j \leftarrow \infty$;
9: **else**

$$k \leftarrow \min \underset{l=1,\dots,\#\mathcal{A}^{j-1}}{\arg\min} \left(-\frac{\pi_l^{j-1}}{\Delta\pi_l^j} \middle| \Delta\pi_l^j < 0 \right),$$

$$\alpha_1^j \leftarrow -\frac{\pi_k^{j-1}}{\Delta\pi_k^j}; \quad \text{(A.2)}$$

10: **end if**;
11: **if** $\Delta z^j = 0$ **then** $\alpha_2^j \leftarrow \infty$;
12: **else** $\alpha_2^j \leftarrow \dfrac{W_p\theta + w_p - G_p z^{j-1}}{G_p \Delta z^j}$;
13: **end if**;
14: $\alpha^j \leftarrow \min(\alpha_1^j, \alpha_2^j)$;
15: **if** $\alpha^j = \infty$ **then return infeasible**; **end if**;
16: **if** $(\alpha_2^j = \infty)$ **then**
17: $\pi^j \leftarrow \pi^{j-1} + \alpha^j \Delta\pi^j$;
18: $\mathcal{A}^j \leftarrow \mathcal{A}^{j-1} \setminus \{k\}$, update J_1^j, J_2^j, R^j, **go to** Step 6;
19: **end if**;
20: Set

$$z^j \leftarrow z^{j-1} + \alpha^j \Delta z^j; \quad \text{(A.3a)}$$

$$\pi^j \leftarrow \begin{bmatrix} \pi^{j-1} p^{j-1} \end{bmatrix} + \alpha^j \begin{bmatrix} \Delta\pi^j \\ 1 \end{bmatrix}; \quad \text{(A.3b)}$$

21: **if** $\alpha^j = \alpha_2^j$ **then**
22: $\mathcal{A}^j \leftarrow \mathcal{A}^{j-1} \cup \{p\}$, update J_1^j, J_2^j, R^j, **go to** Step 3;
23: **else if** $\alpha^j = \alpha_1^j$ **then**
24: $\mathcal{A}^j \leftarrow \mathcal{A}^{j-1} \setminus \{k\}$, Update J_1^j, J_2^j, R^j, **go to** Step 6;
25: **end if**.

Output: Vectors $z^* \leftarrow z^j$ (primal solution), $\pi^* \leftarrow \pi^j$ (dual solution), and $\mathcal{A}^* \leftarrow \mathcal{A}^j$ (optimal active set), or infeasibility status; $N \leftarrow q$ (number of iterations).

constraints $\mathcal{A}^j$ starting from $(z^{j-1}, \mathcal{A}^{j-1})$ and a violated constraint p^q. This is done by setting z^j and π^j as in (A.3), where $\alpha^j \in \mathbb{R}_+$ is the step size, and Δz^j, $\Delta\pi^j$ are the update directions in the primal and dual space, respectively. The primal and dual directions are chosen as

$$\Delta z^j = -\mathcal{H}^{j-1} G_p' \quad \text{(8a)}$$

$$\Delta\pi^j = -G_{\mathcal{A}^{j-1}}^* G_p' \quad \text{(8b)}$$

and the step size $\alpha^j = \min\{\alpha_1^j, \alpha_2^j\}$ with

$$\alpha_1^j = \min \left\{ \min_{l \in \{1,\dots,\#\mathcal{A}^{j-1}\}} \left\{ -\frac{\pi_l^{j-1}}{\Delta\pi_l^j} \middle| \Delta\pi_l^j < 0 \right\}, \infty \right\} \quad \text{(9a)}$$

$$\alpha_2^j = \frac{W_p\theta + w_p - G_p z^{j-1}}{G_p \Delta z^j} \quad \text{(9b)}$$

is chosen such that $\pi^j \ge 0$ and, if possible, the pth constraint becomes active, that is, $g_p(z^j) = 0$. In [16], Goldfarb and Idnani prove that, being $g_p(z^j)$ a violated constraint, the following relations hold:

$$g_p(z^j) \ge g_p(z^{j+1}) \ge g_p(z^{j+h}) = 0 \quad \text{(10a)}$$

$$f(z^j) \le f(z^{j+1}) \le f(z^{j+h}) \quad \text{(10b)}$$

between two iterations j and $j + h$ of Algorithm 1 in which constraints are dropped for h subiterations, that is, $\alpha^{j+i} = \alpha_1^{j+i} < \infty$, $\forall i = 0, \dots, h-1$, and constraint $g_p(z^j)$ is added at subiteration $j + h$, that is, $\alpha^{j+h} = \alpha_2^{j+h} < \infty$. Goldfarb and Idnani [16] also show that strict inequalities hold in (10) every time the step size α^j is positive.

The following three situations can occur during the jth subiteration.

1) If $\alpha^j \equiv \alpha_1^j$ only a partial step can be taken, the kth constraint is blocking and it must be dropped.
2) If $\alpha^j \equiv \alpha_2^j$ a full step is taken and $\mathcal{A}^j = \mathcal{A}^{j-1} \cup \{p^j\}$
3) If $\alpha^j \equiv \infty$ the problem is infeasible.

To avoid computing $\mathcal{H}$ and $G_{\mathcal{A}}^*$ explicitly, the Cholesky factorization of $\mathcal{H}$ and the QR factorization of $G_{\mathcal{A}}^*$ are iteratively updated by Algorithm 1. Let $H = LL'$ be the Cholesky factorization of the primal Hessian and define

$$L^{-1} G_{\mathcal{A}^j} = \begin{bmatrix} Q_1^j & Q_2^j \end{bmatrix} \begin{bmatrix} R^j \\ 0 \end{bmatrix} \quad \text{(11a)}$$

$$J^j = \begin{bmatrix} J_1^j & J_2^j \end{bmatrix} = \begin{bmatrix} L^{-T} Q_1^j & L^{-T} Q_2^j \end{bmatrix}. \quad \text{(11b)}$$

Then the two operators $\mathcal{H}^j$, $G_{\mathcal{A}^j}$ can be written as

$$\mathcal{H}^j = J_2^j J_2^{j\prime} \quad \text{(12a)}$$

$$G_{\mathcal{A}^j}^* = R^{-1,j} J_1^{j\prime} \quad \text{(12b)}$$

where $J_1^j \in \mathbb{R}^{n \times \#\mathcal{A}^j}$, $J_2 \in \mathbb{R}^{n \times \#\mathcal{I}^j}$, $J^0 = L^{-T}$, $R^0 = [\,]$, and $[\,]$ is the empty matrix ($n = 0$ or $m = 0$). Thus, the GI algorithm only needs to iteratively update J^j and R^j, so to compute the primal and dual directions as (cf. Steps 6 and 7 of Algorithm 1)

$$\Delta z^j = -J_2^j J_2^{j\prime} G_p' \quad \text{(13a)}$$

$$\Delta\pi^j = -R^{-1,j} J_1^{j\prime} G_p' \quad \text{(13b)}$$

and without requiring the explicit computation of $\mathcal{H}^j$, $G_{\mathcal{A}^j}$.

Dual active-set methods have the degree of freedom to select which constraint becomes active at each iteration, see Step 3 of Algorithm 1. The order the constraints are added to the active set changes the number of iterations, although it does not affect convergence [38], as condition (11) guarantees convergence no

matter how the next violated constraint $g_p(z^j) > 0$ is selected. Although the selection can be arbitrary, in the literature two techniques are commonly used: the *most-violated* constraint selection rule

$$p \leftarrow \begin{cases} 0 \text{ if } g_i(z^j) \leq 0,\ \forall i \in \mathcal{I} \\ \min \arg\max_{i:\ g_i(z^j)>0} \{g_i(z^j)\} \text{ otherwise} \end{cases} \tag{14a}$$

and the *first-violated* constraint selection rule

$$p \leftarrow \begin{cases} 0 \text{ if } g_i(z^j) \leq 0,\ \forall i \in \mathcal{I} \\ \min_{i \in \mathcal{K}} \{i |\ g_i(z^j) > 0\} \text{ otherwise.} \end{cases} \tag{14b}$$

In (15a), the "min" before the "arg max" imposes to select the smallest index in case of multiple maximizers (the same rule is adopted in (A.2) for choosing the index k in case of multiple minima).

IV. Complexity Certification Analysis

We now derive results that allow the exact characterization of the worst-case number of iterations performed by Algorithm 1 parametrically with respect to θ, for a given set Θ of parameters that perturbs problem (2), in case the violated constraints are selected according to the most common rules (15a) or (15b).

Lemma IV.1. Let $\Theta \subseteq \mathbb{R}^{n_\theta}$ be a polyhedron. Then, a finite number $N_{\max}$ exists, such that

$$N_{\max} = \max_{\theta \in \Theta} N(\theta) \tag{15}$$

that is Algorithm 1 terminates in at most $N_{\max}$ steps for all $\theta \in \Theta$.

Proof: Convergence of Algorithm 1 for each given $\theta \in \Theta$ is proved in [16, Theorem 3], showing that because of (10) the same combination $\mathcal{A}^{j-1}$ of active constraints cannot be generated during the iterations of the algorithm. Since $q \leq j$ and the number $N_{\mathcal{A}}$ of combinations is finite, the algorithm stops in at most $N_{\max} = \max_{\theta \in \Theta}\{N(\theta)\} \leq N_{\mathcal{A}}$ iterations for all $\theta \in \Theta$. ■

Lemma IV.1 proves only an existence result for $N_{\max}$. The value $N_{\max}$ depends on the selection rule (3) for violated constraints, as this determines the sequence of active-set guesses $\mathcal{A}^j$. In most cases $N_{\max}$ is much smaller than the number $N_{\mathcal{A}}$ of all possible combinations of active constraints. In the remaining part of this section, we quantify the exact value of $N_{\max}$.

Definition 3: Given a polyhedron $\Theta \subseteq \mathbb{R}^{n_\theta}$, and denoting by $\mathring{\Theta}$ its interior, the collection of sets $\{\Theta_1, \ldots, \Theta_s\}$ is said a *polyhedral partition* of Θ if Θ_i is a polyhedron, $\Theta_i \subseteq \mathbb{R}^{n_\theta}, \forall i = 1, \ldots, s$, $\cup_{i=1}^s \Theta_i = \Theta$, and $\mathring{\Theta}_i \cap \mathring{\Theta}_j = \emptyset, \forall i, j = 1, \ldots, s, i \neq j$.

Definition 4: A function $n : \Theta \to \mathbb{N}, \Theta \subseteq \mathbb{R}^{n_\theta}$, is said *integer piecewise constant* (IPWC) if there exist a polyhedral partition $\Theta_1, \ldots, \Theta_s$ of Θ and a set of integers $\{n_1, \ldots, n_s\} \subset \mathbb{N}$, such that

$$n(\theta) = \min_{i \in \{1,\ldots,s\}:\ \theta \in \Theta_i} \{n_i\} \tag{16}$$

for all $\theta \in \Theta$.

Note that the "min" in (17) avoids possible multiple definitions of $n(\theta)$ on overalapping boundaries $\Theta_i \cap \Theta_j \neq \emptyset$.

Definition 5: A function $h : \Theta \to \mathbb{R}^n,\ \Theta \subseteq \mathbb{R}^{n_\theta}$, is said PWA if there exist an IPWC function $n : \Theta \to \{n_1, \ldots, n_s\}$ defined over a polyhedral partition $\Theta_1, \ldots, \Theta_s$ of Θ and s pairs (F_i, f_i), $F_i \in \mathbb{R}^{n \times n_\theta}$, $f_i \in \mathbb{R}^n$, $i \in \{n_1, \ldots, n_s\}$, such that

$$h(\theta) = F_{n(\theta)}\theta + f_{n(\theta)} \tag{17a}$$

for all $\theta \in \Theta$. It is said *piecewise constant* if $F_i = 0,\ \forall i \in \{n_1, \ldots, n_s\}$.

Lemma IV.2. Let $f, g : \Theta \to \mathbb{R}$ be PWA functions over a polyhedron $\Theta \subseteq \mathbb{R}^n$. Then, $h = \min\{f, g\}$ is also PWA.

Proof: Let $\{\Theta_1, \ldots, \Theta_s\}$ and $\{\Phi_1, \ldots, \Phi_t\}$ be the polyhedral partitions associated with f and g, respectively, and $\{p_1, \ldots, p_s\}$, $\{k_1, \ldots, k_t\}$ define the corresponding IPWC functions n and m. Let $F_i, f_i,\ i = 1, \ldots, s$ and $G_j, g_j,\ j = 1, \ldots, t$, define f and g, respectively. Consider the polyhedra

$$\Psi_{ij}^f = \{\theta \in \Theta : \theta \in \Theta_i \cap \Phi_j,\ (F_i - G_j)\theta \leq g_j - f_i\} \tag{18a}$$

$$\Psi_{ij}^g = \{\theta \in \Theta : \theta \in \Theta_i \cap \Phi_j,\ (G_i - F_j)\theta \leq f_j - g_i\} \tag{18b}$$

and let $\{\Psi_1, \ldots, \Psi_v\}$ be the set of all polyhedra Ψ_{ij}^f, Ψ_{ij}^g having a nonempty interior. It is easy to show that $\{\Psi_1, \ldots, \Psi_v\}$ defines a polyhedral partition of Θ. By letting $\ell_i = i, i = 1, \ldots, v$, and defining the IPWC function $\ell : \Theta \to \{1, \ldots, v\}$

$$\ell(\theta) = \min_{i \in \{1,\ldots,v\}:\ \theta \in \Psi_i} \{\ell_i\}$$

we have that

$$h(\theta) = \begin{cases} F_i\theta + f_i & \text{if} \quad \theta \in \Psi_{\ell(\theta)} = \Psi_{ij}^f \ \text{ for some } j \\ & \qquad\qquad\quad j \in \{1, \ldots, t\} \\ G_j\theta + g_j & \text{if} \quad \theta \in \Psi_{\ell(\theta)} = \Psi_{ij}^g \ \text{ for some } i \\ & \qquad\qquad\quad i \in \{1, \ldots, s\} \end{cases}$$

and therefore that h is PWA. ■

The following Lemma IV.3 characterizes the behavior of one iteration j of Algorithm 1 as a function of the parameter vector θ.

Lemma IV.3. Let $z^{j-1} : \Theta^j \to \mathbb{R}^n$ and $\pi^{j-1} : \Theta^j \to \mathbb{R}^m$ be affine functions on a polyhedron $\Theta^j \subseteq \mathbb{R}^{n_\theta}$, and let $\mathcal{A}^{j-1} \subseteq \mathcal{K}$, $\mathcal{V}^{j-1} \subseteq \mathcal{K} \setminus \mathcal{A}^{j-1}$. The following properties hold.

1) By letting $p(\theta) = 0$ if no constraint is violated ($\mathcal{V}^{q-1} = \emptyset$), the index $p : \Theta^j \to \mathbb{N} \cup \{0\}$ selected according to (14a) or (14b) is IPWC.
2) The step directions $\Delta z^j, \Delta\pi^j$ obtained from (8) are PWC.
3) The index $k : \Theta^j \to \mathbb{N}$ selected according to (1) is IPWC.
4) The step size $\alpha^j : \Theta^j \to \mathbb{R} \cup \{+\infty\}$ defined in (9) is PWA.
5) The functions $z^j : \Theta^j \to \mathbb{N}^n$, $\pi^j : \Theta^j \to \mathbb{R}^m$ defined by (A.3) are PWA.

Proof: 1) Let $z^j(\theta) = A_z\theta + b_z$ define the affine function z^j over Θ^j. Then, $g_\ell(z^j) = G_\ell z^j - W_\ell\theta - w_\ell = (G_\ell A_z - W_\ell)\theta + G_\ell b_z - w_\ell$ is also affine, $\forall \ell \in \mathcal{I}$. Let $\bar{G}_\ell = G_\ell A_z - W_\ell$, $\bar{w}_\ell = w_\ell - G_\ell b_z$ and

$$\Theta_\ell^1 = \{\theta \in \Theta^j :\ (\bar{G}_\ell - \bar{G}_h)\theta \geq \bar{w}_h - \bar{w}_\ell,\ \bar{G}_\ell\theta \geq \bar{w}_\ell,\ \forall h \in \mathcal{I},\ h \neq \ell\} \tag{19a}$$

$$\Theta_\ell^2 = \{\theta \in \Theta^j :\ \bar{G}_h\theta \leq \bar{w}_h,\ \bar{G}_\ell\theta \geq \bar{w}_\ell\ \ \forall h \in \mathcal{I} \cap \{1, \ldots, \ell - 1\}\}. \tag{19b}$$

Consider the polyhedra

$$\Theta_0 = \{\theta \in \Theta^j : \bar{G}_\ell \theta \leq \bar{w}_\ell, \forall \ell \in \mathcal{I}\}$$
$$\Theta_\ell = \begin{cases} \Theta_j^1 & \text{if} \quad p = \arg\max\{\bar{G}_\ell \theta - \bar{w}_\ell\} \\ \Theta_j^2 & \text{if} \quad p = \min_{i \in \mathcal{K}}\{i |\ g_i(z^j) > 0\}. \end{cases} \tag{20}$$

The polyhedra $\{\Theta_\ell\}_{\ell=0}^m$ define a polyhedral partition of Θ^j as a) they are included in Θ^j by construction, b) each $\theta \in \Theta_{p(\theta)}$, where $p(\theta) = 0$ if $\bar{G}_\ell \theta \geq \bar{w}_\ell$, $\forall \ell \in \mathcal{I}$, or $p = \arg\max\{\bar{G}_\ell \theta - \bar{w}_\ell\}$ if rule (14a) is used, or $p = \min_{i \in \mathcal{K}}\{i |\ g_i(z^j) > 0\}$ in case (15b) is adopted, 3) the interiors $\mathring{\Theta}_\ell$ are obtained by changing "$\geq$" to "$>$" in (21), so no θ can belong simultaneously to two different interiors of polyhedra. Since both rules in (15) are equivalent to setting

$$p(\theta) = \min_{\ell \in \{0,\ldots,m\}:\ \theta \in \Theta_\ell} \{\ell\} \tag{21}$$

function p is IPWC.

2) From (8) we have that $\mathcal{H}^{j-1}$ and $G^*_{\mathcal{A}^{j-1}}$ are independent from θ, so the same holds for J_1^j, J_2^j and $R^{-1,j}$. Since G_p depends on p that by (21) is IPWC, the step directions obtained from (8) are PWC functions.

3) Since π^{j-1} is an affine function of θ, say $\pi^{j-1}(\theta) = A_\pi \theta + b_\pi$, and $\Delta\pi^j$ is PWC, their negative ratio $-\frac{\pi^{j-1}(\theta)}{\Delta\pi_l^j}$ is PWC over the partition of Θ^j defined by (20), for all $l \in \mathcal{P}^{j-1} = \{k_1, \ldots, k_t\} = \{l \in \mathcal{A}^{j-1} :\ \Delta\pi_l^j < 0\}$, where $t = \#\,\mathcal{P}^{j-1}$. Let $(C_{l,1}, d_{l,1})$, …, $(C_{l,s}, d_{l,s})$ define the corresponding affine terms of the PWC ratios and consider the polyhedra

$$\Phi_{i,\ell} = \{\theta \in \Theta_\ell :\ (C_{k_i,\ell} - C_{k_h,\ell})\theta \leq d_{k_h,\ell} - d_{k_i,\ell}, \quad \forall h = 1, \ldots, t,\ h \neq i,\ \ell = 1, \ldots, s\}. \tag{22}$$

Each set of polyhedra $\{\Phi_{i,\ell}\}_{i=1}^t$ also defines a polyhedral partition of Θ_ℓ, for all $\ell = 1, \ldots, s$. Since (A.2) is equivalent to setting

$$k(\theta) = \min_{i \in \{1,\ldots,t\}:\ \theta \in \cup_{\ell \in \mathcal{P}^{j-1}} \Phi_{i,\ell}} \{k_i\} \tag{23}$$

k is also an IPWC function on Θ^j.

4) If $\#\,\mathcal{A}^{j-1} = 0$ or $\Delta\pi^j \geq 0$, then $\alpha_1^j(\theta) = +\infty$ for all $\theta \in \Theta^j$, cf. Step 1 of Algorithm 1. Otherwise

$$\alpha_1^j(\theta) = C_{k(\theta)}\theta + d_{k(\theta)} \tag{24}$$

and therefore α_1^j is PWA. Similarly, since z^{j-1} is affine, we have that

$$\alpha_2^j(\theta) = C_{h(\theta)}\theta + d_{h(\theta)} \tag{25}$$

with

$$C_{h(\theta)} = \frac{W_{p(\theta)} - G_{p(\theta)}A_z}{G_{p(\theta)}\Delta z^j}, \quad d_{h(\theta)} = \frac{w_{p(\theta)} - G_{p(\theta)}b_z}{G_{p(\theta)}\Delta z^j} \tag{26}$$

is also PWA. If $\alpha_1^j(\theta) = +\infty$, then $\alpha^j(\theta) = \alpha_2^j(\theta)$ for all $\theta \in \Theta^j$ and so α^j is PWA. Otherwise, Lemma IV.2 proves that $\alpha^j = \min\{\alpha_1^j, \alpha_2^j\}$ is PWA. As a result, we consider the set of polyhedra

$$\tilde{\Phi}_{0,\ell} = \{\theta \in \Theta_\ell :\ C_{k_j,\ell} - C_{h,\ell} \leq d_{h,\ell} - d_{k_j,\ell}, \quad \forall j = 1, \ldots, t\}$$
$$\tilde{\Phi}_{i,\ell} = \Phi_{i,\ell} \setminus \tilde{\Phi}_{0,\ell},\ \forall i = 1, \ldots, t \tag{27}$$

such that $\{\tilde{\Phi}_{i,l}\}_{i=0}^t$ defines a polyhedral partition of Θ_ℓ.

5) The functions z^j and π^j defined in (A.3) are the sum of an affine and PWA function, and are therefore PWA.

Theorem IV.4. Let $\Theta \subseteq \mathbb{R}^{n_\theta}$ be a polyhedron and z^0, π^0 be affine functions of θ on Θ. Then each iterate z^j, π^j defined in (A.3) is PWA for all $j \in \mathbb{N}$, such that Algorithm 1 is executed. Moreover, $N : \Theta \to \{0, \ldots, N_{\max}\}$ is IPWC.

Proof: We prove the theorem by induction. Since z^0, π^0 are affine, they are also PWA. Assume z^{j-1}, π^{j-1} are PWA functions defined over a polyhedral partition $\{\Theta_i^{j-1}\}_{i=1}^{s^{j-1}}$ of Θ. Therefore, z^{j-1} and π^{j-1} are affine on each polyhedron Θ_i^{j-1}, and by property 5) of Lemma IV.3 we have that z^j, π^j are PWA functions defined over a polyhedral partition $\{\Theta_{ih}^{j-1}\}_{h=1}^{t_i^{j-1}}$ of Θ_i^{j-1}. As the collection of sets

$$\left\{\Psi_i^j\right\}_{i=1}^{L^j} = \left\{\Theta_{1h}^{j-1}\right\}_{h=1}^{t_1^{j-1}} \cup \ldots \cup \left\{\Theta_{s^{j-1}h}^{j-1}\right\}_{h=1}^{t_{s^{j-1}}^{j-1}}$$

where $L^j = \sum_{i=1}^{s^{j-1}} t_i^{j-1}$, is a polyhedral partition of Θ^{j-1}, it follows that z^j, π^j are PWA over Θ^{j-1}.

Since by Lemma IV.1 the number of recursive partitioning is finite, Θ gets partitioned in a finite number of polyhedral sets $\Psi_1, \ldots, \Psi_M$. Each polyhedron Ψ_i is characterized by either $\mathcal{V}^j = \emptyset$ (optimal solution found) or $\alpha^j = \alpha_1^j = \infty$ (infeasibility) and by the number $N_i \leq N_{\max}$ of recursive splitting it took to get defining Ψ_j. Then, the function N such that $N(\theta) = N_i$ if $\theta \in \Psi_i$, $i = 1, \ldots, M$ is IPWC. ■

The following corollary confirms a well-known property of the optimizer z^* proved in [9, Theorem 4].

Corollary IV.5. The multiparametric QP solution vector z^* of (2) is PWA with respect to θ over a subset of Θ.

Proof: Easily follows since, for each $i = 1, \ldots, M$, either $z^*(\theta) = z^{N_i}(\theta)$ or the problem is infeasible, for all $\theta \in \Psi_i$. ■

A. Complexity Certification Algorithm

We derive next an algorithm that iteratively constructs two lists of tuples $\mathbb{T}$ and $\bar{\mathbb{T}}$, corresponding to parameters $\theta \in \Theta$ for which the QP (2) has an optimal solution or is infeasible, respectively. Each tuple is defined as

$$T^h = (\Theta^h, \mathcal{A}^h, A_z^h, b_z^h, A_\pi^h, b_\pi^h, J_1^h, J_2^h, R^h, q^h) \tag{28}$$

where Θ^h are polyhedra that we will show provide a partition of Θ.

The lists are constructed by partitioning the given set Θ of parameters depending on the behavior of each consecutive iteration q made by Algorithm 1. The core of the certification algorithm is the way to split a given polyhedron Θ_ℓ, obtained as in (20) for the iteration q, in all the possible polyhedra that are either *feasible*, with constraint $p(\theta)$ active at $q+1$, or *infeasible*. For a given Θ_ℓ, let Γ_ℓ^i and $\bar{\Gamma}_\ell^j$ define the ith feasible and the jth infeasible polyhedra. From Theorem IV.4, it

Algorithm 2: Parametric GI's dual QP Iteration.

Input: Tuple $T^q = (\Theta^q, \mathcal{A}^q, A_z^q, b_z^q, A_\pi^q, b_\pi^q, J_1^q, J_2^q, R^q, q)$
Matrices G, W, w

Define $\{\Theta_\ell\}_{\ell=0}^m$ as in (20) and p as in (21);
if $\mathring{\Theta}_0 \neq \emptyset$ **then**
  $T_o = (\Theta_0, \mathcal{A}^q, A_z^q, b_z^q, A_\pi^q, b_\pi^q, J_1^q, J_2^q, R^q, q)$;
end if
for $\ell = 1, \ldots, m$ such that $\mathring{\Theta}_\ell \neq \emptyset$ **do**
  $\mathcal{T}_\ell \leftarrow \{T^q\}$, $\mathcal{T}_{\Gamma,\ell} \leftarrow \emptyset$, $\mathcal{T}_{\bar{\Gamma},\ell} \leftarrow \emptyset$;
  while $\mathcal{T}_\ell \neq \emptyset$ **do**
    extract from $\mathcal{T}_\ell$ a tuple
    $T_\ell^j \leftarrow (\Theta_\ell^j, \mathcal{A}_\ell^j, A_{z,\ell}^j, b_{z,\ell}^j, A_{\pi,\ell}^j, b_{\pi,\ell}^j, J_{1,\ell}^j, J_{2,\ell}^j, R_\ell^j, q)$;
    $\Delta z_\ell^j \leftarrow -J_{2,\ell}^j (J_{2,\ell}^j)' G_p$, $\Delta\pi_\ell^j \leftarrow -R_\ell^{-1,j} (J_{1,\ell}^j)' G_p'$;
    if $(\Delta\pi_\ell^j \geq 0 \quad \textbf{or} \quad \#\mathcal{A}^j = 0)$ **then** $\alpha_{1,\ell}^j \leftarrow \infty$;
      if $\Delta z_\ell^j = 0$ **then** $\mathcal{T}_{\bar{\Gamma},\ell} \leftarrow \mathcal{T}_{\bar{\Gamma},\ell} \cup \{T_\ell^j\}$; **else**
        Update $A_{z,\ell}^{j;+;1}, b_{z,\ell}^{j;+;1}, A_{\pi,\ell}^{j;+;1}, b_{\pi,\ell}^{j;+;1}$ as in (30)–(31);
        Update $J_{1,\ell}^{j;+;1}, J_{2,\ell}^{j;+;1}, R_\ell^{j;+;1}, \mathcal{A}_\ell^{j;+;1}$ adding constraint p;
        Set the tuple
        $T_{1,\ell}^{j;+;1} \leftarrow (\Theta_\ell^j, \mathcal{A}_\ell^{j;+;1}, A_{z,\ell}^{j;+;1}, b_{z,\ell}^{j;+;1}, A_{\pi,\ell}^{j;+;1}, b_{\pi,\ell}^{j;+;1}, J_{1,\ell}^{j;+;1}, J_{2,\ell}^{j;+;1}, R_\ell^{j;+;1}, q;+;1)$;
        $\mathcal{T}_{\Gamma,\ell} \leftarrow \mathcal{T}_{\Gamma,\ell} \cup \{T_{1,\ell}^{j;+;1}\}$;
      end if;
    else
      partition Θ_ℓ as in (27), with k as in (23);
      if $\Delta z_\ell^j > 0$ **then**
        Update $A_{z,\ell}^{j;+;1}, b_{z,\ell}^{j;+;1}, A_{\pi,\ell}^{j;+;1}, b_{\pi,\ell}^{j;+;1}$ as in (30)–(31);
        Update $J_{1,\ell}^{j;+;1}, J_{2,\ell}^{j;+;1}, R_\ell^{j;+;1}, \mathcal{A}_\ell^{j;+;1}$ adding constraint p;
        Set the tuple
        $T_{1,\ell}^{j;+;1} \leftarrow (\tilde{\Phi}_{0,\ell}^j, \mathcal{A}_\ell^{j;+;1}, A_{z,\ell}^{j;+;1}, b_{z,\ell}^{j;+;1}, A_{\pi,\ell}^{j;+;1}, b_{\pi,\ell}^{j;+;1}, J_{1,\ell}^{j;+;1}, J_{2,\ell}^{j;+;1}, R_\ell^{j;+;1}, q;+;1)$;
        $\mathcal{T}_{\Gamma,\ell} \leftarrow \mathcal{T}_{\Gamma,\ell} \cup \{T_{1,\ell}^{j;+;1}\}$;
      end if;
      for $i = 1, \ldots, t$ with $\mathring{\Phi}_{i,\ell} \neq \emptyset$ **do**
        Update $A_{\pi,\ell}^{j;+;1}, b_{\pi,\ell}^{j;+;1}$ as in (32);
        Update $J_{1,\ell}^{j;+;1}, J_{2,\ell}^{j;+;1}, R_\ell^{j;+;1}, \mathcal{A}_\ell^{j;+;1}$ removing constraint k;
        if $\Delta z_\ell^j = 0$ **then**
          Set the tuple
          $T_{2,\ell}^{j;+;1} \leftarrow (\tilde{\Phi}_{i,\ell}^j, \mathcal{A}_\ell^{j;+;1}, A_{z,\ell}^j, b_{z,\ell}^j, A_{\pi,\ell}^{j;+;1}, b_{\pi,\ell}^{j;+;1}, J_{1,\ell}^{j;+;1}, J_{2,\ell}^{j;+;1}, R_\ell^{j;+;1}, q)$;
        else
          Update $A_{z,\ell}^{j;+;1}, b_{z,\ell}^{j;+;1}$ as in (30);
          Set the tuple
          $T_{2,\ell}^{j;+;1} \leftarrow (\tilde{\Phi}_{i,\ell}^j, \mathcal{A}_\ell^{j;+;1}, A_{z,\ell}^{j;+;1}, b_{z,\ell}^{j;+;1}, A_{\pi,\ell}^{j;+;1}, b_{\pi,\ell}^{j;+;1}, J_{1,\ell}^{j;+;1}, J_{2,\ell}^{j;+;1}, R_\ell^{j;+;1}, q)$;
        end if;
        $\mathcal{T}_\ell = \mathcal{T}_\ell \cup \{T_{2,\ell}^{j;+;1}\}$;
      end for;
    end if;
  end while
end for.

Output: Tuple T_o; sets of tuples $\{\mathcal{T}_{\Gamma,\ell}^i\}_{\ell=1,i=1}^{m,\ n_{\gamma,\ell}}$, $\{\mathcal{T}_{\bar{\Gamma},\ell}^j\}_{\ell=1,j=1}^{m,\ n_{\bar{\gamma},\ell}}$.

follows that $\{\Gamma_\ell^i\}_{i=1}^{n_\gamma} \cup \{\bar{\Gamma}_\ell^j\}_{j=1}^{n_{\bar{\gamma}}}$ define a polyhedral partition of Θ_ℓ, where the polyhedra will differ from each other for the sequence of constraints dropped before adding the constraint $p(\theta)$ to the current active set, or before verifying an infeasibility status. Therefore, each tuple at iteration q is iteratively split into two sets of tuples $\{\mathcal{T}_{\Gamma,\ell}^i\}_{\ell=1,i=1}^{m,\ n_{\gamma,\ell}}$ and $\{\mathcal{T}_{\bar{\Gamma},\ell}^j\}_{\ell=1,j=1}^{m,\ n_{\bar{\gamma},\ell}}$, associated with the sets of polyhedra $\{\Gamma_\ell^i\}_{\ell=1,i=1}^{m,\ n_{\gamma,\ell}}$ and $\{\bar{\Gamma}_\ell^j\}_{\ell=1,j=1}^{m,\ n_{\bar{\gamma},\ell}}$. For each new tuple created from T^{j-1}, a parametric primal–dual update is performed, such that by considering the relation

$$(\tilde{C}, \tilde{d}) = \begin{cases} (C_k, d_k) \text{ if } \alpha^j \equiv \alpha_1^j \\ (C_h, d_h) \text{ if } \alpha^j \equiv \alpha_2^j \end{cases} \tag{29}$$

the parametric primal update becomes

$$\begin{aligned} A_z^j &= A_z^{j-1} + \tilde{C}\Delta z^j \\ b_z^j &= b_z^{j-1} + \tilde{d}\Delta z^j \end{aligned} \tag{30}$$

where the parametric dual update in the case of an added constraint is defined by

$$\begin{aligned} A_\pi^j &= \begin{bmatrix} A_\pi^{j-1} \\ 0 \end{bmatrix} + \tilde{C} \begin{bmatrix} \Delta\pi^j \\ 1 \end{bmatrix} \\ b_\pi^j &= \begin{bmatrix} b_\pi^{j-1} \\ 0 \end{bmatrix} + \tilde{d} \begin{bmatrix} \Delta\pi^j \\ 1 \end{bmatrix} \end{aligned} \tag{31}$$

and the parametric dual update in the case of a blocking constraint removal is computed as

$$\begin{aligned} A_\pi^j &= A_\pi^{j-1} + \tilde{C}\Delta\pi^j \\ b_\pi^j &= b_\pi^{j-1} + \tilde{d}\Delta\pi^j. \end{aligned} \tag{32}$$

Algorithm 2 characterizes the way a given polyhedron Θ^q is split at a generic iteration q. Next, Algorithm 3 executes Algorithm 2 iteratively to characterize the entire parameter set Θ.

Algorithm 3 provides the list of optimal and infeasible tuples that partition the set Θ into polyhedra characterized by different steps of a dual active-set algorithm to reach the optimal solution, or to detect infeasibility. Note that parallel computations can be easily used to speed up the execution of Algorithm 3, by creating an independent task for each T^q extracted from $\mathcal{C}$.

Algorithm 3: GI's QP Solver Certification.

Input: Matrices H, F, G, W, w defining problem (2) and polyhedral set Θ of parameters.

1: Compute Cholesky factorization $LL' = H$, $J_2^0 \leftarrow L^{-T}$;
2: $\mathbb{T} \leftarrow \emptyset$, $\bar{\mathbb{T}} \leftarrow \emptyset$
3: $\mathcal{C} \leftarrow \{(\Theta, \emptyset, -H^{-1}F, 0, [\,], [\,], [\,], J_2^0, [\,], 0)\}$;
4: **while** $\mathcal{C} \neq \emptyset$ **do**
5: $T^q \leftarrow$ extract tuple from $\mathcal{C}$;
6: Execute Algorithm 2 with input data from T^q;
7: $\mathbb{T} \leftarrow \mathbb{T} \cup \{T_o\}$;
8: $\bar{\mathbb{T}} \leftarrow \bar{\mathbb{T}} \cup \{\mathcal{T}^i_{\Gamma,\ell}\}_{\ell=1,i=1}^{m,\ n_{\bar{\gamma},\ell}}$;
9: $\mathcal{C} \leftarrow \{\mathcal{T}^i_{\Gamma,\ell}\}_{\ell=1,i=1}^{m,\ n_{\gamma,\ell}}$;
10: **end while**

Output: List of *optimal* tuples $\mathbb{T}$, list of *infeasible* tuples $\bar{\mathbb{T}}$.

The following key result of the paper is therefore proved.

Proposition IV.6. Consider the QP problem (2) for $\theta \in \Theta$ and let $\mathbb{T}$, $\bar{\mathbb{T}}$ be the lists of tuples generated by Algorithm 3. Then, for any $\theta \in \Theta$, the dual active-set method of Algorithm 1 takes no more than

$$N_{\max} = \max_{T \in \mathbb{T} \cup \bar{\mathbb{T}}} \{q(T)\} \tag{33}$$

iterations to solve (2) or to detect infeasibility, and exactly $N_{\max}$ iterations for all $\theta \in \Theta(T)$ such that $q(T) = N_{\max}$, where $q(T)$ and $\Theta(T)$ denote the value q and set Θ associated with tuple T, respectively. ■

Remark 1: The bound $N_{\max}$ also applies to other dual active-set methods that select the violated constraint p with the same rule used for generating $\{\Theta_\ell\}_{\ell=0}^m$ in (21), thanks to the fact that different dual active-set QP solvers share the same path to reach the solution. See Section V-A.

In embedded applications what counts is the CPU runtime rather than the number of QP iterations, which may differ from one iteration to another. Indeed, given a polyhedron Θ_ℓ at iteration q, the execution of Algorithm 2 computes the list of tuples $\{\mathcal{T}^i_{\Gamma,\ell}\}_{\ell=1,i=1}^{m,\ n_{\gamma,\ell}}$ that, despite their equivalence in terms of iteration complexity $(q+1)$, have a different execution time, due to the different sequence of dropped constraints. This issue is addressed by counting the exact number of flops n_{imp} required to reach the optimal solution in each tuple T^h

$$T^h = (\Theta^h, \mathcal{A}^h, A_z^h, b_z^h, A_\pi^h, b_\pi^h, J_1^h, J_2^h, R^h, q^h, n_{\text{imp}}^h) \tag{34}$$

that is incrementally updated at every subiteration j. Note that n_{imp}^h may be even further differentiated into the number of atomic and nonatomic operations, like square-roots. This allows the exact certification of the worst-case computational complexity in terms of CPU flops

$$n_{\text{imp}}^{\max} = \max_{T \in \mathbb{T} \cup \bar{\mathbb{T}}} \{n_{\text{imp}}(T)\} \tag{35}$$

where $n_{\text{imp}}(T)$ associates the value n_{imp} with tuple T, from which the maximum execution time can be computed for a given hardware architecture. Note that not necessarily $q(T) = N_{\max}$ implies $n_{\text{imp}}(T) = n_{\text{imp}}^{\max}$.

In case of degeneracy, active-set methods can suffer from the *cycling* problem. Due to the possibility of choosing the constraint to add to the active set at each iteration, dual methods are much less affected by this problem [39] with respect to primal methods. However, even if in [16], Goldfarb and Idnani have not encountered any example of cycling of the GI algorithm, the impossibility of degeneracy has not been proved yet. The certification algorithm is able to exactly verify the possible income of degeneracy. Indeed, by iteratively storing the active sets at all the previous iterations for each region of parameters, it is straightforward to verify if the same active set is visited more than once during the execution of the solver. Moreover, the certification algorithm not only verifies the possibility of cycling, but also the regions where this can happen. Therefore, appropriate procedures, such as small perturbations of the constraints, can be applied only for the regions where cycling happens.

B. Simple Numerical Example

We illustrate the iterative steps performed by Algorithm 3 to certify the computational complexity of the toy QP problem defined by the following matrices:

$$H = \begin{bmatrix} 0.9916 & -0.0033 & 0.0191 \\ -0.0033 & 1.0539 & 0.0003 \\ 0.0191 & 0.0003 & 0.9572 \end{bmatrix},$$

$$F = \begin{bmatrix} 38.4790 & 27.6810 \\ 28.9098 & 35.3639 \\ -142.6567 & -176.6410 \end{bmatrix}$$

$$G = \begin{bmatrix} -1.1512 & 1.2484 & -0.6112 \\ 1.4613 & 0.4288 & -1.9000 \\ -0.5442 & -2.2504 & 0.6687 \\ -1.1315 & -0.5512 & 1.0281 \\ -0.3603 & -1.5912 & 0.8453 \end{bmatrix},$$

$$W = \begin{bmatrix} 0.1304 & 41.8883 \\ 0.6998 & 59.3813 \\ 16.7784 & 27.6981 \\ 15.0436 & 59.2827 \\ 51.5494 & 58.5140 \end{bmatrix}$$

$$w = \begin{bmatrix} 82.2717 \\ 76.4544 \\ 12.0567 \\ 38.4332 \\ 2.7923 \end{bmatrix}, \ E_\theta = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}, \ e_\theta = \begin{bmatrix} 0.9 \\ -0.6 \\ -0.6 \\ 1 \end{bmatrix} \tag{36}$$

where E_θ and e_θ define the set of parameters $\Theta = \{\theta \in \mathbb{R}^{n_\theta} | E_\theta \theta \leq e_\theta\}$. Algorithm 3 iteratively partitions Θ into poly-

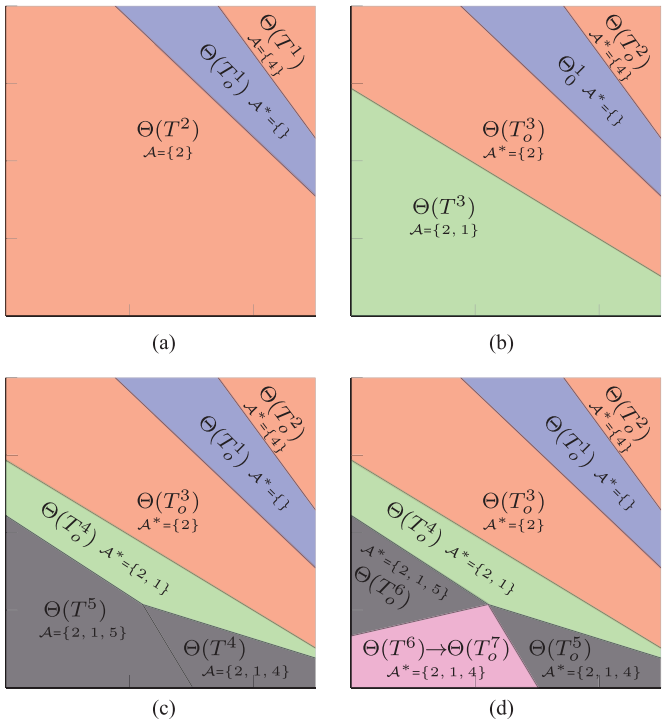

Fig. 1. Iterations of Algorithm 3, with most-violated selection rule, to certify the QP problem (36). Different colors mean different number of iterations, namely blue=unconstrained, red = 1, green = 2, gray = 3, and pink = 4 QP iterations. (a) Iterate tuples with $q(T^h) = 0$. (b) Iterate tuples with $q(T^h) = 1$. (c) Iterate tuples with $q(T^h) = 2$. (d) Iterate tuples with $q(T^h) = 3$.

hedral regions as depicted in Fig. 1. Specifically, each subfigure represents consecutively the results of the execution of subroutine 2 for all the nonoptimal tuples at the previous step. The regions corresponding to the optimal tuples are labeled as $\Theta(T_o)$, with optimal active set $\mathcal{A}^*$. Those regions corresponding to nonoptimal tuples are labeled as $\Theta(T)$, with current active set $\mathcal{A}$. Fig. 1(d) shows the partitioning of Θ after the complete execution of the algorithm, and Table I collects the list $\mathbb{T}$ of the seven optimal tuples characterizing the problem, together with the number of iterations and the associated flops. By using the proposed certification algorithm, we can exactly calculate that the GI solver applied to the QP problem (37) will perform four iterations and 528 flops in the worst case, with three nonatomic operations (square-roots in this case).

The complete map of the GI solver execution in the parameter space provides additional useful information. The average computation time can be also certified, which is useful for those applications where the time spent by the CPU in the wake-up mode is more important than the hard real time. Furthermore, the list of constraints never added to the active set can be easily retrieved, and those can be removed from the QP problem to improve memory occupancy and computational complexity. For instance, Table I shows that the third constraint can be safely removed from problem (37). Note that a simple redundancy check of $\{\begin{bmatrix} z \\ \theta \end{bmatrix} : Gz - W\theta \leq w\}$ would have labeled the same constraint as nonredundant.

V. Certified Fast Embedded MPC

The approach developed in the previous section is also useful to significantly improve the performance of the embedded dual active-set solver, as described in the following sections.

A. Certification of Different Dual Active-Set Algorithms

Dual active-set algorithms that add the violated constraints one-by-one into the active set perform the same number of iterations, as long as the selection rule for adding the violated constraint is the same [40]. The difference is how they solve (5), that translates into a different complexity per iteration [16]–[18], [20]. Usually, the appropriate QP solver is selected heuristically based on the dimensions of the QP problem (2). Indeed, it is recognized that range-space methods work better with few constraints into the active set [16], null-space methods are the favorite when the number of constraints in the active set is larger [37], and methods based on Schur decomposition are preferable with large-scale QP problems [17]. However, selecting the best solver on the basis of such general rules of thumb is not always easy, and it is especially difficult to predict the worst-case complexity precisely.

The certification algorithm can be easily extended to deal with such a solver selection problem. Indeed, the tuples created by the algorithm would be exactly the same for each solver and it is sufficient to run the algorithm only once, associating at each tuple one flops counter n_{imp}^i for each ith solver to be tested. Despite their equivalent number of iterations $N_{\max}$, the computational load $n_{\text{imp}}^{i,\max}$ of the active-set solvers will be in general different, allowing one to select the best one for the given QP problem.

B. Certification of Different Violated Constraints Selection Rules

Lemma IV.1 guarantees the convergence of a dual active-set QP solver if a violated constraint is added at each iteration to the current active set, no matter how the violated constraint is selected. Lemma IV.3 proves that the index $p : \Theta^j \to \mathbb{N} \cup \{0\}$ selected according to either the most-violated (15a) or the first-violated (15b) rule is IPWC, and therefore, for each iteration q of the algorithm, it defines the polyhedral partition $\{\Theta_\ell\}_{\ell=0}^m$.

The criterion (15a) is commonly preferred, as it usually turns into fewer iterations with respect to other selection rules, and it reduces the possibility to pass through nearly linearly dependent intermediate active sets [16]. However, checking all constraints is a costly operation, as $G_i z - b_i$ must be computed for all rows $i \in \mathcal{I}$. In MPC, where the number of constraints m is usually larger with respect to the maximum active set size n, this operation plays an important role.

The first violated rule (15b) avoids this costly operation, at the price of possibly increasing the number of iterations. Depending on the particular QP problem, the worst case can be arbitrarily affected by different selection rules, thus the certification algorithm can be used to select the best one. Similarly, other selection rules can be certified, provided that p remains IPWC. A run of the certification algorithm is necessary for

every selection rule, using each time a different rule to build $\{\Theta_\ell\}_{\ell=0}^m$, and the one with the lowest $n_{\text{imp}}^{\max}$ can then be selected for minimum worst-case execution time.

C. Worst-Case Analysis of Explicit MPC

As a by-product, Algorithm 3 also provides the multiparametric solution of (2), as the convex polyhedra associated with each optimal active set $\mathcal{A}$, corresponding to the union of the regions $\theta(T), T \in \mathbb{T}$, such that $\mathcal{A}(T) = \mathcal{A}$, can be immediately computed as in [9], by imposing the primal and dual feasibility conditions

$$(GA_z(T) - W)\theta \leq w - Gb_z(T) \tag{37a}$$

$$-A_\pi(T)\theta \leq b_\pi(T) \tag{37b}$$

for some T such that $\mathcal{A}(T) = \mathcal{A}$.

Let the explicit optimal control law be the PWA function

$$u_{\text{exp}}^*(\theta) = \{K_i\theta + c_i, \quad \forall\theta \in \mathcal{P}_i, \quad i = 1, \ldots, n_r\} \tag{38}$$

with $K_i \in \mathbb{R}^{n_u \times n_\theta}, c_i \in \mathbb{R}^{n_u}, \{\mathcal{P}_i\}_{i=1}^{n_r}$ a polyhedral partition of Θ such that $\mathcal{P}_i = \{\theta \in \mathbb{R}^{n_\theta} | E_i\theta \leq e_i, \forall i = 1, \ldots, n_r\}$, where $E_i \in \mathbb{R}^{n_e^i \times \theta}, e_i \in \mathbb{R}^{n_e^i}$ are a minimal polyhedral representation of (37). For each polyhedron $\mathcal{P}_i$ defining the PWA function (38), let

$$C_i = \{h \,|\, \mathcal{P}_h \text{ is a neighbor of } \mathcal{P}_i,\ h = 1, \ldots, n_r\} \tag{39}$$

be the list of "neighboring" polyhedra of $\mathcal{P}_i$ ($\mathcal{P}_h$ is a neighbor of $\mathcal{P}_i$ if they share a facet), with $C_i \in \mathbb{N}_+^{n_{c_i}}$. The worst-case number of flops $n_{\text{exp}}^{\max}$, needed to find and apply the optimal solution through explicit MPC, and the memory occupancy m_{exp} in kilobytes to store the PWA control law and the needed information for point location, can be calculated as

$$n_{\text{exp}}^{\max} = 2n_\theta(n_u + n_r) - n_r + \sum_{i=1}^{n_r} n_c^i \tag{40a}$$

$$m_{\text{exp}} = \frac{((n_\theta + 1)n_u + n_\theta + 1)n_r}{p_f} + 2\frac{\sum_{i=1}^{n_r} n_c^i}{p_c} \tag{40b}$$

with $p_c = 512$, and $p_f = 128$ for double precision or $p_f = 256$ for single precision, under the assumption of using the explicit algorithm presented in [41].

Having both the exact bound on the worst-case number of flops of a dual active-set solver and of the multiparametric solution, for a given linear MPC problem one can evaluate whether to adopt the online solver or the explicit solution, as the required memory occupancy and flops that need to be allocated in both cases can be computed *exactly* (see Section VI for examples).

D. Worst-Case Partial Enumeration

Based on the partial enumeration idea proposed in [33], where the explicit solution is partially stored as a table of fixed dimensions and updated online, we derive next a novel online technique for embedded MPC that combines implicit and explicit MPC. In [33], the table contains the optimal active sets at the most recent decision time-points, and, besides the suboptimal solution obtained when the table needs to be updated, its main drawback is that even if the average behavior is improved, there is no guarantee regarding the worst case. Another approach is presented in [29], where the partial explicit solution is used to warm start a primal active-set LP solver. This iterates for a fixed number of iterations, and therefore provides a possibly suboptimal solution.

The method proposed next, referred to as WCPE-MPC, builds instead upon the complexity certification algorithm of Section IV to identify those regions that, if stored as an explicit solution, will help improving the worst-case number of flops of an implicit approach. Being interested in the worst-case improvement, the partial solution that must be stored is of fixed dimensions, and not updated online. The basic idea of the proposed WCPE-MPC approach is to first search the optimal solution in the partial explicit PWA law, and, if the search fails, the online active-set solver is executed, as described below.

Define $\bar{\mathbb{P}} = \mathbb{P}(\{1, \ldots, n_r\}) \setminus (\emptyset, \{1, \ldots, n_r\})$, with $\mathbb{P}(X)$ the power-set of X, and consider the partial explicit control law derived from (39)

$$u_{\text{exp},\mathcal{N}}^*(\theta) = \{K_j\theta + c_j, \quad \forall\theta \in \mathcal{P}_j, \quad \forall j \in \mathcal{N}\} \tag{41}$$

where $\mathcal{N} \in \bar{\mathbb{P}}$ is the set of indices corresponding to the affine functions from (39) to be included in the explicit solution, and $\#\mathcal{N} < n_r$ and $\cup_{j\in\mathcal{N}}\{\mathcal{P}_j\} \subset \Theta$.

Consider the set of indices

$$\mathcal{M} = \{h \mid h \in C_j,\ \forall j \in \mathcal{N}\} \tag{42}$$

and define the scalar PWA *descriptor function* $r(\theta) : \mathbb{R}^{n_\theta} \to \mathbb{R}$

$$r(\theta) = \{r_i(\theta) = \bar{K}_i\theta + \bar{c}_i \mid \theta \in \mathcal{P}_i, \quad i \in \mathcal{M}\} \tag{43}$$

where $\bar{K}_i \in \mathbb{R}^{n_\theta}$, $\bar{c}_i \in \mathbb{R}$, and $\bar{K}_j \neq \bar{K}_h, \forall h \in C_j,\ j \in \mathcal{N}$. The PWA descriptor function $r(\theta)$ is derived from the PWA optimal control law (38), cf. [42]. For each $\mathcal{P}_j$, let $O_j(\theta) = \{o_i^j(\theta) \mid i \in C_j\}$ be an *ordering function*, such that

$$o_i^j(\theta) = \begin{cases} +1 & \text{if } r_j(\theta) \geq r_i(\theta) \\ -1 & \text{if } r_j(\theta) < r_i(\theta) \end{cases} \tag{44}$$

with $j \in \mathcal{N}$ and $i \in \mathcal{M}$, and assume that a set of precalculated $S_j = C_j(\bar{\theta})$, with $\bar{\theta} \in \mathring{\mathcal{P}}_j$, $\forall j \in \mathcal{N}$. Given the partial explicit PWA control law (42), the scalar PWA descriptor function (44) and the set of precomputed S_j, $\forall j \in \mathcal{N}$, Algorithm 4 presents the steps of WCPE-MPC, executed online at each time step in order to obtain the command input u^*.

The challenge in designing Algorithm 4 is the selection of $u_{\text{exp},\mathcal{N}}^*(\theta)$, such that the total worst-case number of flops is improved with respect to the implicit worst-case $n_{\text{imp}}^{\max}$. The number of possible partial explicit solutions that can be derived from (38) is $2^{n_r} - 2$. We discuss next how to select only those that can improve the implicit worst-case $n_{\text{imp}}^{\max}$.

Definition 6: Given a set of indices $\mathcal{N} \in \bar{\mathbb{P}}$, let $n_{\mathcal{N}}^{\max}$ be the worst-case number of flops of WCPE-MPC Algorithm 4 with input argument the partial explicit solution $u_{\text{exp},\mathcal{N}}^*$, such that

$$n_{\mathcal{N}}^{\max} = n_{\text{exp},\mathcal{N}}^{\max} + n_{\text{imp},\mathcal{N}}^{\max} \tag{45}$$

Algorithm 4: WCPE-MPC Algorithm.

Input: Matrices H, F, G, W, w, θ defining problem (2).
$u^*_{\text{exp},\mathcal{N}}(\theta) = \{K_j\theta + c_j, \quad \forall\theta \in \mathcal{P}_j, \quad \forall j \in \mathcal{N}\}$,
$r(\theta) = \{r_i(\theta) = \bar{K}_i\theta + \bar{c}_i \mid \theta \in \mathcal{P}_i, \quad i \in \mathcal{M}\}$, and
$\{S_j\}_{j\in\mathcal{N}}$
1: $j \leftarrow 1$, $f \leftarrow$ TRUE;
2: **while** f **or** $j \leq \#\mathcal{N}$ **do**
3: compute $O_j(\theta)$ as in (44);
4: **if** $O_j(\theta) = S_j$ **then**
5: $f \leftarrow$ FALSE;
6: **else**
7: $j \leftarrow j + 1$;
8: **end if**
9: **end while**
10: **if** $f =$FALSE **then**
11: **return** $u^* \leftarrow K_j\theta + c_j$;
12: **else**
13: $z^* \leftarrow$ execute Algorithm 1 with inputs H, F, G, W, w, θ;
14: **return** $u^* \leftarrow$ first n_u rows of z^*;
15: **end if**
Output: Optimal control input sequence u^*.

where $n^{\max}_{\text{exp},\mathcal{N}}$ is the required number of flops, in the worst case, for locating a point in the partially explicit solution, and $n^{\max}_{\text{imp},\mathcal{N}}$ is the worst-case number of flops for the online solver.

Let $\bar{\mathbb{P}}$ be the collection of all the possible sets of indices to build the partial explicit solution (42), and let $\{\mathcal{N}_i\}_{i=1}^{n_r-1}$, with $\mathcal{N}_i \in \bar{\mathbb{P}}, \forall\, i \in \{1, \dots, n_r - 1\}$, be the reduced collection of sets of indices defined such that

$$\mathcal{N}_j = \mathcal{N}_i \cup h, \quad \forall j > i, \quad h \in \{1, \dots, n_r\} \setminus \mathcal{N}_i \tag{46a}$$

$$n^{\max}_{\text{imp},\mathcal{N}_j} \leq n^{\max}_{\text{imp},\mathcal{N}_i}, \quad \forall j > i \tag{46b}$$

with $i \in \{1, \dots, n_r - 2\}$ and $j \in \{2, \dots, n_r - 1\}$. Consider the vth set of indices $\mathcal{N}_v$ that defines the partial explicit solution $u^*_{\text{exp},\mathcal{N}_v}(\theta)$. Let $n^{\max}_{\mathcal{N}_v} = n^{\max}_{\text{exp},\mathcal{N}_v} + n^{\max}_{\text{imp},\mathcal{N}_v}$ be the worst-case number of flops of Algorithm 4 with $u^*_{\text{exp},\mathcal{N}_v}(\theta)$ as the input argument, and define the list of tuples $\mathbb{T}_{\mathcal{N}_v} = \{T^i_{\mathcal{N}_v}\}_{i=1}^{\#\,\mathbb{T}_{\mathcal{N}_v}}$, such that

$$\mathbb{T}_{\mathcal{N}_v} = \{T^i \in \mathbb{T} \mid \Theta(T^i) \cap \{\mathcal{P}_j\}_{j\in\mathcal{N}_v} = \emptyset,\ \forall i = 1, \dots, \#\,\mathbb{T}\} \tag{47}$$

with $\mathbb{T}$ the list of optimal tuples obtained from Algorithm 3. The flops contribution of the implicit solver to the worst-case $n^{\max}_{\mathcal{N}_v}$ is

$$n^{\max}_{\text{imp},\mathcal{N}_v} = \max\{n_{\text{imp}}(T^i_{\mathcal{N}_v}) |\ i = 1, \dots, \#\,\mathbb{T}_{\mathcal{N}_v}\} \tag{48}$$

with $n^{\max}_{\text{imp},\mathcal{N}_v} = n_{\text{imp}}(T^s_{\mathcal{N}_v})$. Consider then $\mathcal{N}_h = \mathcal{N}_v \cup \{h\}$, with h an index to be chosen, and the corresponding partial explicit law $u^*_{\text{exp},\mathcal{N}_h}(\theta)$. The relation $n^{\max}_{\text{exp},\mathcal{N}_h} > n^{\max}_{\text{exp},\mathcal{N}_v}$ holds by construction for every h. Therefore, the necessary condition to meet the requirement $n^{\max}_{\mathcal{N}_h} < n^{\max}_{\mathcal{N}_v}$ is that $n^{\max}_{\text{imp},\mathcal{N}_h} < n^{\max}_{\text{imp},\mathcal{N}_v}$ and h is selected such that $T^s_{\mathcal{N}_v} \subset \mathcal{P}_h$. In conclusion, $\{\mathcal{N}_i\}_{i=1}^{n_r-1}$ are the only set of indices for which the corresponding set of partial explicit solutions $u^*_{\text{exp},\mathcal{N}_i}(\theta)$ can guarantee that $n^{\max}_{\mathcal{N}_i} \leq n^{\max}_{\text{imp}}$ holds true.

Algorithm 5: WCPE Reduction Algorithm.

Input: List of optimal tuples $\mathbb{T} = \{T^i\}_{i=1}^{\#\,\mathbb{T}}$ from Algorithm 3,
$u^*_{\text{exp}}(\theta) = \{K_i\theta + c_i,\ \forall\theta \in \mathcal{P}_i,\ i = 1, \dots, n_r\}$ from (38),
$r(\theta) = \{r_i(\theta) = \bar{K}_i\theta + \bar{c}_i \mid \theta \in \mathcal{P}_i, \quad i \in \mathcal{M}\}$, and the memory occupancy of implicit MPC m_{imp}
1: $\mathcal{N} \leftarrow \emptyset$, $u^*_{\text{exp},\mathcal{N}} \leftarrow \emptyset$, $n^{\max}_{\mathcal{N}} \leftarrow \emptyset$, $m_{\mathcal{N}} \leftarrow \emptyset$, $i \leftarrow 0$, $t \leftarrow \emptyset$;
2: **while** $i < n_r$ **do**
3: $s = \arg\max_{h=\{1,\dots,\#\,\mathbb{T}\}}\{n_{\text{imp}}(T^h)\}$
4: Extract from $\mathbb{T}$ the tuple $T^s = (\Theta^s, \mathcal{A}^s, A^s_z, b^s_z, A^s_\pi, b^s_\pi, J^s_1, J^s_2, R^s, q^s, n^s_{\text{imp}})$;
5: Remove tuple T^s from $\mathbb{T}$;
6: **for** $v = 1, \dots, \#\,\mathbb{P}$ **do**
7: **if** $\Theta^s \cap \mathcal{P}_v \neq \emptyset$ **then**
8: $i \leftarrow i + 1$, $t \leftarrow \emptyset$;
9: $\mathcal{N}_i \leftarrow \mathcal{N}_{i-1} \cup \{v\}$;
10: **for** $j = 1, \dots, \#\,\mathbb{T}$ **do**
11: Extract from$\mathbb{T}$ the tuple $T^j = (\Theta^j, \mathcal{A}^j, A^j_z, b^j_z, A^j_\pi, b^j_\pi, J^j_1, J^j_2, R^j, q^j, n^j_{\text{imp}})$;
12: **if** $\mathcal{A}^j = \mathcal{A}^s$ **then**
13: $t \leftarrow t \cup j$;
14: **end if**
15: **end for**
16: Remove from $\mathbb{T}$ all the tuples indexed by t;
17: $u^*_{\text{exp},\mathcal{N}_i}(\theta) = \{K_j\theta + c_j, \quad \forall\theta \in \mathcal{P}_j, \quad \forall j \in \mathcal{N}_i\}$;
18: Compute $n^{\max}_{\text{exp},\mathcal{N}_i}$, $m_{\text{exp},\mathcal{N}_i}$ as in (49);
19: $l = \arg\max_{h=\{1,\dots,\#\,\mathbb{T}\}}\{n_{\text{imp}}(T^h)\}$
20: Extract from $\mathbb{T}$ the tuple $T^l = (\Theta^l, \mathcal{A}^l, A^l_z, b^l_z, A^l_\pi, b^l_\pi, J^l_1, J^l_2, R^l, q^l, n^l_{\text{imp}})$;
21: $n^{\max}_{\mathcal{N}_i} \leftarrow n^{\max}_{\text{exp},\mathcal{N}_i} + n^l_{\text{imp}}$;
22: $m_{\mathcal{N}_h} \leftarrow m_{\text{imp}} + m_{\text{exp},\mathcal{N}_i}$;
23: **go to** Step 2
24: **end if**
25: **end for**
26: **end while**
Output: List of partial explicit optimal solutions $\{u^*_{\text{exp},\mathcal{N}_i}(\theta)\}_{i=1}^{n_r-1}$ and corresponding $\{n^{\max}_{\mathcal{N}_i}\}_{i=1}^{n_r-1}$ and memory allocation $\{m_{\mathcal{N}_i}\}_{i=1}^{n_r-1}$

Note that we only provided the necessary condition for a partial explicit solution $u^*_{\text{exp},\mathcal{N}_j}(\theta)$ to improve the worst-case number of flops of implicit MPC, that is, $\mathcal{N}_j \in \{\mathcal{N}_i\}_{i=1}^{n_r-1}$, reducing the set of interest from $2^{n_r} - 2$ to $n_r - 1$ possibilities. However, the best partial PWA law, among the possible $n_r - 1$, depends on the particular problem, and the given computational and memory limits. To this purpose Algorithm 5 provides a way to compute all the partial explicit solutions $\{u^*_{\text{exp},\mathcal{N}_i}(\theta)\}_{i=1}^{n_r-1}$, together with the lists of corresponding worst-case number of flops $\{n^{\max}_{\mathcal{N}_i}\}_{i=1}^{n_r-1}$ and memory allocation $\{m_{\mathcal{N}_i}\}_{i=1}^{n_r-1}$. The designer can then select the best choice, considering the tradeoff memory/worst-case computations.

TABLE I
LIST $\mathbb{T}$ OF OPTIMAL TUPLES GENERATED BY ALGORITHM 3 FOR PROBLEM (36)

	$\mathcal{A}(T^h)$	$q(T^h)$	$n_{\text{imp}}(T^h)(\pm,*,\div)$|sqrt
T_o^1	$\emptyset$	unconstrained	35|0
T_o^2	$\{4\}$	1	178|2
T_o^3	$\{2\}$	1	174|2
T_o^4	$\{2,1\}$	2	295|3
T_o^5	$\{2,1,4\}$	3	388|3
T_o^6	$\{2,1,5\}$	3	372|3
T_o^7	$\{2,1,4\}$	4	528|3

TABLE II
DIMENSIONS OF MPC PROBLEMS

	Inverted pendulum	DC motor	Nonlinear demo	AFTI-16
n_x	5	4	5	6
n_u	1	1	3	2
n_y	2	2	2	2
N_p	50	10	5	10
N_u	5	2	2	2
n	5	3	6	5
m	10	10	18	12
n_θ	9	6	10	10

Given an index set of affine functions for the partial explicit solution, the following equations characterize the maximum number of flops and the memory occupancy of the explicit control law in WCPE-MPC

$$n_{\text{exp},\mathcal{N}}^{\max} = 2n_\theta n_u + 2n_\theta \,\#\mathcal{N} + \sum_{i\in\mathcal{N}} n_e^i \tag{49a}$$

$$m_{\text{exp}} = \frac{(n_\theta+1)\,\#\mathcal{M} + (n_u n_\theta + n_u)\,\#\mathcal{N}}{p_f} + 2\frac{\sum_{i\in\mathcal{N}} n_c^i}{p_c}. \tag{49b}$$

Result 1: Let $m_{\mathcal{N}_i}$ be the memory requirements for WCPE-MPC, when implemented with the ith partial PWA function of $\{\mathcal{N}_i\}_{i=1}^{n_r-1}$. Then, $m_{\mathcal{N}_i} > m_{\mathcal{N}_j}, \forall i > j$. Let $\bar{m}$ be the memory allocation to store the WCPE-MPC code (including GI solver code), and the matrices required by the GI algorithm. Then, $\bar{m}$ is independent from the particular selection of $\mathcal{N}_i$, and therefore $m_{\mathcal{N}_i} = \bar{m} + m_{\text{exp},\mathcal{N}_i}$.

VI. EXAMPLES

We test the algorithms developed in the previous sections on four benchmark MPC problems taken from the demos library of the MPC Toolbox for MATLAB: An *inverted pendulum* control problem, consisting of controlling a single-input-multioutput inverted pendulum on a cart, with a measured disturbance and input constraints; the *dc motor* control problem, concerning the control of a dc servomechanism under voltage and shaft torque constraints [43]; the *nonlinear demo* control problem, consisting of controlling a multi-input multioutput nonlinear plant with a linear MPC formulation; and the multivariable *AFTI-16* aircraft control problem, characterized by an open-loop unstable pole and constraints on both inputs and outputs [44]. We use here exactly the same settings used in the toolbox, and the dimensions of all the problems are listed in Table II. The dc motor and the AFTI-16 problems are detailed in [43] and [44], respectively. The problem of the inverted pendulum on a cart is described by

$$x_{k+1} = \begin{bmatrix} 1 & 0.0095 & 0.0005 & 0 \\ 0 & 0.9048 & 0.0934 & 0.0005 \\ 0 & -0.0010 & 1.0019 & 0.0100 \\ 0 & -0.1905 & 0.3832 & 1.0019 \end{bmatrix} x_k + \begin{bmatrix} 0 & 0 \\ 0 & 0.0095 \\ 0.0001 & 0.0001 \\ 0.0200 & 0.0190 \end{bmatrix} u_k$$

$$y_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x_k,\ W_{\Delta u} = I,\ W_y = I \begin{bmatrix} 1.2 & 0 \\ 0 & 1 \end{bmatrix}$$

$$-200 \le u_k \le 200 \tag{50}$$

while the nonlinear demo by the linear MPC setting

$$x_{k+1} = \begin{bmatrix} 0.8187 & 0 & 0 & 0 & 0 \\ 0.1474 & 0.6550 & -0.1637 & 0.0490 & 0.4878 \\ 0.0164 & 0.1637 & 0.9825 & 0.0034 & 0.0524 \\ 0 & 0 & 0 & 0.8013 & -0.1801 \\ 0 & 0 & 0 & 0.1801 & 0.9813 \end{bmatrix} x_k +$$

$$+ \begin{bmatrix} 0.1813 & 0 & 0 \\ 0.0164 & 0.1637 & 0.0034 \\ 0.0011 & 0.0175 & 0.0002 \\ 0 & 0 & 0.1801 \\ 0 & 0 & 0.0187 \end{bmatrix} u_k,\ y_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 \end{bmatrix} x_k,$$

$$W_{\Delta u} = 0.1I,\ W_y = I,$$

$$\begin{bmatrix} -3 & -2 & -2 \end{bmatrix}' \le u_k \le \begin{bmatrix} 3 & 2 & 2 \end{bmatrix}'. \tag{51}$$

In Sections VI-A and VI-B, we present the results of worst-case certification of implicit and explicit MPC, and the certified improvements in the worst case when applying WCPE-MPC to the same problems. In both sections, we use the most violated rule for the constraint selection. Then, Section VI-C shows how to select the best constraint selection rule, with the lowest certified worst-case number of flops.

A. Worst-Case Certification Algorithm

To better clarify the operation of Algorithm 3, Fig. 2 shows a sample two-dimensional, section of the polyhedra associated with optimal tuples $\{T^i\}_{i=1}^{\#\mathbb{T}}$ for the inverted pendulum control problem, where regions corresponding to nodes that share the same number of iterations have the same color. As all output constraints are treated as soft constraints, the QP problem is feasible over the entire set of interest. Table III shows the results of the certification algorithm for the GI solver described by Algorithm 1, in which the standard most violated rule (15a) is used for constraint selection, in terms of the worst-case number of iterations $N_{\max}$, the memory allocation m_{imp} in single and double precision, and the worst-case number $n_{\text{imp}}^{\max}$ of flops. The table also collects information on explicit MPC, namely

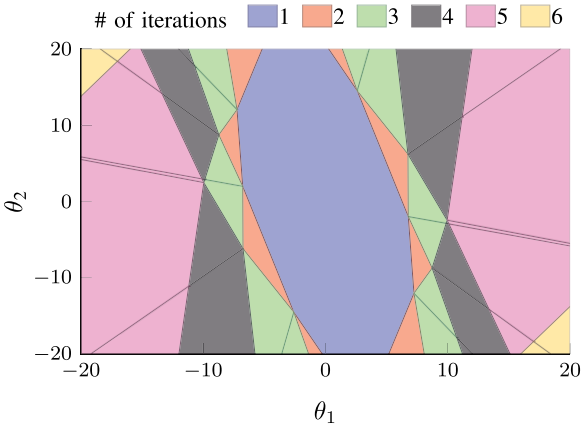

Fig. 2. Results of the explicit certification algorithm applied to the inverted pendulum problem: Partition of the parameter set Θ based on the number of iterations required by the GI QP solver (same color = same number of QP iterations).

TABLE III
RESULTS FROM THE COMPLEXITY CERTIFICATION ALGORITHM OF GI SOLVER (MAX VIOLATED SELECTION RULE (14A))

	Inv. pend.	DC motor	Nonlin. demo	AFTI-16
Explicit MPC				
n_r	87	67	214	417
$n_{\text{exp}}^{\max}$ $(\pm, *)$	3295	1622	8938	16531
m_{exp} 16| 32 bit (kb)	14.8| 21.6	8.0| 11.7	56.6| 93.4	88.2| 142.0
Implicit MPC				
$N_{\max}$	10	9	9	16
$n_{\text{imp}}^{\max}$ $(\pm, *, \div)$|sqrt	3809| 22	2082| 9.0	6109| 25	7807| 33
m_{imp} 16| 32 bit (kb)	8.5| 9.2	8.2| 8.6	9.0| 10.3	8.6| 9.4
t_{ca} [s]	198	173	287	9366

the number of regions of the multiparametric solution n_r, its memory occupancy m_{exp} in single and double precision, and the worst-case number $n_{\text{exp}}^{\max}$ of flops, computed as in (41). Memory allocation also comprises the solver code, that, in the case of explicit MPC, is about 1 kB, while Algorithm 1 takes approximately 7.7 kB. We count square roots separately from other arithmetic operations, as the associated flops depend on the computer architecture. Table III allows the designer to assess for each problem whether explicit is preferable than implicit MPC in terms of speed and memory occupancy. By referring to Table III, it turns out that for the inverted pendulum and dc motor problems explicit MPC would be the preferred choice if the worst-case speed is the sole concern. For the inverted pendulum problem, for example, if memory occupancy is also taken into account one may trade off a 15.6% increase of worst-case flops for a memory reduction of 57.4% by adopting implicit MPC. For the nonlinear and AFTI-16 problems, implicit MPC outperforms the explicit solution in terms of both memory and speed, for instance, a memory reduction of 93.4% and a worst-case flops reduction of 52.8% in the AFTI-16 problem. Table III collects also the offline time t_{ca} taken by the certification algorithm to execute on a 2.50 GHz Intel Core i7-4710MQ CPU.

TABLE IV
CHARACTERIZATION OF EXPLICIT, IMPLICIT AND WCPE-MPC WITH AN INCREASING CONTROL HORIZON FOR THE INVERTED PENDULUM PROBLEM (DOUBLE PRECISION)

	Explicit MPC			Implicit MPC			WCPE-MPC		t_{ca} [s]
N_u	n_r	$n_{\text{exp}}^{\max}$	$m_{\text{exp}}^{\max}$	N_r	$n_{\text{imp}}^{\max}$	m_{imp}	$n_{\mathcal{N}^*}^{\max}$	$m_{\mathcal{N}^*}$	
5	87	3295	21.6	10	3809| 22	9.2	3621	12.2	198
6	149	5645	36.4	12	6441| 33	9.8	5974	13.2	589
7	245	9311	59.3	14	10 054| 48	10.4	9267	13.9	3226
8	373	14 213	90.0	16	14 783| 64	11.1	13 650	14.1	6475
9	551	21 023	132.6	18	20 749| 80	11.9	19 180	15.5	14 252
10	763	29 188	183.5	20	28 095| 99	12.7	26 101	16.3	26 262

B. Results for WCPE-MPC

Table IV presents the results obtained with WCPE-MPC applied to the inverted pendulum problem, together with the performance of explicit and implicit MPC. To better understand how the certification algorithm and the three different MPC implementations scale up with the problem dimension, several tests are reported by increasing the control horizon N_u. For what concerns WCPE-MPC, the reported worst-case flops $n_{\mathcal{N}^*}^{\max}$ and memory occupancy $m_{\mathcal{N}^*}$ refer to the partial explicit solution selected according to the best tradeoff memory/speed. This is highlighted in Fig. 3, which shows three out of the six experiments collected by Table IV. The partial explicit PWA solutions $u^*_{\text{exp},\mathcal{N}_j}(\theta)$, with $\mathcal{N}_j \in \{\mathcal{N}_i\}_{i=1}^{n_r-1}$, are obtained by running Algorithm 5. From Result 1, the worst-case number of flops is represented as a function of the increasing memory occupancy $m_{\mathcal{N}_i}$ (yellow line). The results are compared to the worst-case computational cost of both explicit (dashed red line) and implicit (dashed blue line) MPC, the latter derived from Algorithm 3. The points of best tradeoff between memory worst-case number of flops of WCPE-MPC are highlighted in the figure.

For $N_u = 5$ (top figure), implicit MPC and all the possible implementations of WCPE-MPC are computationally more intensive than explicit MPC, which is therefore the preferred solution if the available memory is enough (21.6 kB in the case of double precision). However, when memory is taken into account, WCPE-MPC implemented with the partial PWA law corresponding to the point p_1 guarantees a memory reduction of 56.3%, while worsening the worst-case number of flops only by 9.9%; implicit MPC would instead increase flops by 15.6%.

By increasing the control horizon to $N_u = 8$ (middle figure), the computational cost of implicit MPC is still higher than explicit MPC, even though the difference is smaller and implicit MPC could be selected for reduced memory occupancy. In this case, however, WCPE-MPC outperforms even explicit MPC, providing 4.0% worst-case flops reduction and 84.3% less memory usage.

Finally, when $N_u = 10$ (bottom figure), the results are reversed: implicit MPC is certified to be faster than explicit MPC and therefore it is the preferred solution both in terms of memory and speed. The use of WCPE-MPC further enhances the performance of implicit MPC, by reducing its worst-case number of flops by 7% and allocating 3.6 kB more.

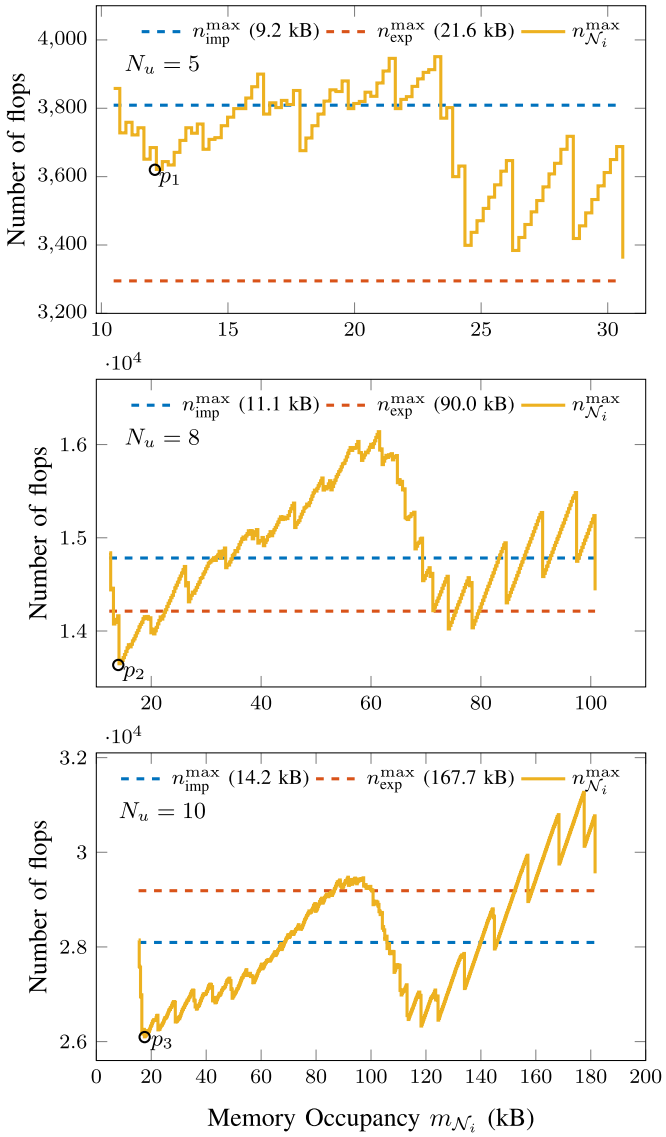

Fig. 3. Results of WCPE-MPC for the inverted pendulum problem with $N_u = 5$ (top), $N_u = 8$ (middle) and $N_u = 10$ (bottom). Computational complexity (yellow line) of WCPE-MPC is plotted as a function of the memory occupancy required to store an increasing number of regions, the complexity of implicit (blue line) and explicit (red line) MPC, and the corresponding memory requirements stored (in brackets). The circled points correspond to the best tradeoff between memory and worst-case execution time.

The above examples clarify the versatility of WCPE-MPC, adding extra flexibility than pure explicit or implicit to enhance speed and/or reduce memory requirements.

C. Results for Different Selection Strategies

Fig. 4 shows the results relative to different selection strategies for the violated constraint p, namely the most-violated (15a) and the first-violated (15b) rules. The figure also shows that it is possible to combine both the certification of WCPE-MPC and the certification of the constraint selection strategy to best choose among all the degrees of freedom available for evaluating the MPC control law.

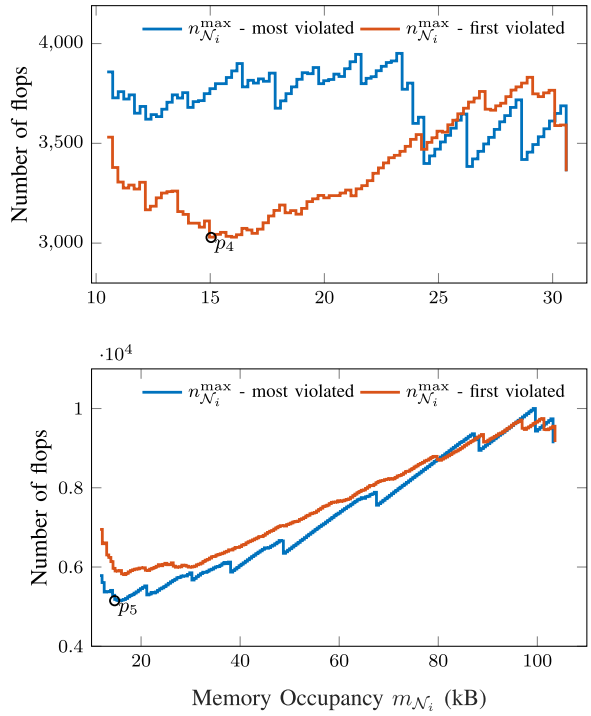

Fig. 4. Results for the certification of the most-violated and first-violated constraint selection rules. From top to bottom: the inverted pendulum on a cart and the nonlinear demo problems. The results are combined with WCPE-MPC to allow selecting the best implementation among all the design degrees of freedom available.

For the inverted pendulum example (top figure), implemented with $N_u = 5$ as in Table III, the first-violated rule outperforms not only the most-violated one, but allows WCPE-MPC to even improve the results of explicit MPC, that would have been the best choice from what observed in Section VI-B. Indeed, WCPE-MPC implemented according to the partial explicit law corresponding to "p_4" in Fig. 4, guarantees a certified 8.2% reduction of worst-case flops and a certified 30.7% reduction of memory with respect to explicit MPC.

On the other hand, for the nonlinear demo problem (bottom plot in Fig. 4), the use of the first-violated rule brings extra cost with respect to the most violated rule, and therefore the best configuration would be the implementation of WCPE-MPC with the partial explicit law selected according to "p_5" in Fig. 3. By considering the results in Table III, WCPE-MPC designed in this way enhances implicit MPC speed by 15.6%, requiring only 5.2 kB more memory.

VII. Conclusion

By analyzing a dual active-set solver for QP in a parametric way, this paper has provided methods to exactly quantify the worst-case number of flops and memory requirements of linear MPC based on online QP of moderate size. Such a characterization is a crucial requirement to certify that the MPC controller is safely implementable in a real-time system. Partial enumeration MPC schemes that combine explicit and implicit MPC can be also certified for worst-case behavior, allowing one to select the best tradeoff between memory occupancy and complexity.

The key property exploited in the certification algorithm is the PWA nature of the intermediate solution steps of the solver with respect to the vector of parameters perturbing the QP problem, that allows splitting the operating range of parameters recursively into convex polyhedral cells, each one labeled with the number of iterations and flops needed to obtain the optimizer. Current research is devoted to extending the ideas of the paper to other QP solvers for embedded MPC that enjoy a similar property.

References

[1] S. Di Cairano, H. Tseng, D. Bernardini, and A. Bemporad, "Vehicle yaw stability control by coordinating active front steering and differential braking in the tire sideslip angles domain," *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 4, pp. 1236–1248, Jul. 2013.

[2] S. Di Cairano, D. Yanakiev, A. Bemporad, I. Kolmanovsky, and D. Hrovat, "Model predictive idle speed control: Design, analysis, and experimental evaluation," *IEEE Trans. Control Syst. Technol.*, vol. 20, no. 1, pp. 84–97, Jan. 2012.

[3] E. N. Hartley and J. M. Maciejowski, "Field programmable gate array based predictive control system for spacecraft rendezvous in elliptical orbits," *Optimal Control Appl. Methods*, vol. 35, no. 7, pp. 585–607, 2015.

[4] G. Cimini, D. Bernardini, A. Bemporad, and S. Levijoki, "Online model predictive torque control for permanent magnet synchronous motors," in *Proc. IEEE Int. Conf. Ind. Technol.*, Mar. 2015, pp. 2308–2313.

[5] S. Vazquez *et al.*, "Model predictive control: A review of its applications in power electronics," *IEEE Ind. Electron. Mag.*, vol. 8, no. 1, pp. 16–31, Mar. 2014.

[6] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 2, pp. 267–278, Mar. 2010.

[7] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *Int. J. Robust Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.

[8] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

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

[10] A. Bemporad, "A multiparametric quadratic programming algorithm with polyhedral computations based on nonnegative least squares," *IEEE Trans. Automat. Control*, vol. 60, no. 11, pp. 2892–2903, Nov. 2015.

[11] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*. Philadelphia, PA, UA: Soc. Ind. Appl. Math., 1994.

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

[13] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for embedded linear model predictive control," *IEEE Trans. Automat. Control*, vol. 59, no. 1, pp. 18–33, Jan. 2014.

[14] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[15] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "osQP—The operator splitting QP solver," 2017. [Online]. Available: https://github.com/oxfordcontrol/osqp

[16] D. Goldfarb and A. Idnani, "A numerically stable dual method for solving strictly convex quadratic programs," *Math. Program.*, vol. 27, no. 1, pp. 1–33, 1983.

[17] R. A. Bartlett and L. T. Biegler, "QPSchur: A dual, active-set, Schur-complement method for large-scale and structured convex quadratic programming," *Optim. Eng.*, vol. 7, no. 1, pp. 5–32, 2006.

[18] P. Gill, N. Gould, W. Murray, M. Saunders, and M. Wright, "A weighted Gram-Schmidt method for convex quadratic programming," *Math. Program.*, vol. 30, no. 2, pp. 176–195, 1984.

[19] A. Bemporad, "A quadratic programming algorithm based on nonnegative least squares with applications to embedded model predictive control," *IEEE Trans. Automat. Control*, vol. 61, no. 4, pp. 1111–1116, Apr. 2016.

[20] A. Forsgren, P. Gill, and E. Wong, "Primal and dual active-set methods for convex quadratic programming," *Math. Program.*, vol. 159, pp. 469–508, 2016.

[21] P. Giselsson and S. Boyd, "Metric selection in fast dual forward backward splitting," *Automatica*, vol. 62, pp. 1–10, 2014.

[22] I. Necoara, "Computational complexity certification for dual gradient method: Application to embedded MPC," *Syst. Control Lett.*, vol. 81, pp. 49–56, 2015.

[23] D. Goldfarb, "On the complexity of the simplex method," in *Advances in Optimization and Numerical Analysis*, ser. Mathematics and Its Applications, S. Gomez and J.-P. Hennart, Eds. Springer Netherlands, 1994, vol. 275, pp. 25–38.

[24] K. H. Borgwardt, "Probabilistic analysis of the simplex method," in *H. Schellhaas et al. (eds) DGOR/NSOR. Operations Research Proceedings*, vol. 1987, pp. 564–575, 1988.

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

[26] V. Klee and G. J. Minty, "How good is the simplex method," *Inequalities-III*, vol. 3, pp. 159–175, 1972.

[27] C. Rao, S. Wright, and J. Rawlings, "Application of interior-point methods to model predictive control," *J. Optim. Theory Appl.*, vol. 99, no. 3, pp. 723–757, 1998.

[28] T. Gal and J. Nedoma, "Multiparametric linear programming," *Manage. Sci.*, vol. 18, pp. 406–442, 1972.

[29] M. Zeilinger, C. Jones, and M. Morari, "Real-time suboptimal model predictive control using a combination of explicit MPC and online optimization," *IEEE Trans. Automat. Control*, vol. 56, no. 7, pp. 1524–1534, Jul. 2011.

[30] R. Fletcher and S. Leyffer, "Numerical experience with lower bounds for MIQP branch-and-bound," *SIAM J. Optim.*, vol. 8, no. 2, pp. 604–616, Feb. 1998.

[31] J. Nocedal and S. Wright, *Numerical Optimization*. New York, NY, USA: Springer, 1999.

[32] R. Milman and E. Davison, "A fast MPC algorithm using nonfeasible active set methods," *J. Optim. Theory Appl.*, vol. 139, no. 3, pp. 591–616, 2008.

[33] G. Pannocchia, J. B. Rawlings, and S. J. Wright, "Fast, large-scale model predictive control by partial enumeration," *Automatica*, vol. 43, no. 5, pp. 852–860, 2007.

[34] A. Alessio and A. Bemporad, "A survey on explicit model predictive control," in *Nonlinear Model Predictive Control: Towards New Challenging Applications* (Lecture Notes in Control and Information Sciences), vol. 384. D. R. L. Magni, F. Allgower, Eds. Berlin, Germany: Springer-Verlag, 2009, pp. 345–369.

[35] M. Jost and M. Monnigmann, "Accelerating model predictive control by online constraint removal," in *Proc. IEEE Conf. Decision Control*, Dec. 2013, pp. 5764–5769.

[36] M. Jost, G. Pannocchia, and M. Mnnigmann, "Online constraint removal: Accelerating MPC with a Lyapunov function," *Automatica*, vol. 57, pp. 164–169, 2015.

[37] H. J. Ferreau, P. Ortner, P. Langthaler, L. del Re, and M. Diehl, "Predictive control of a real-world Diesel engine using an extended online active set strategy," *Annu. Rev. Control*, vol. 31, no. 2, pp. 293–301, 2007.

[38] G. Cimini, D. Bernardini, and A. Bemporad, "An efficient constraint selection strategy in dual active-set methods for model predictive control," to be published.

[39] C. E. Lemke, "A method of solution for quadratic programs," *Manage. Sci.*, vol. 8, no. 4, pp. 442–453, 1962.

[40] M. J. Best and N. Chakravarti, "Active set algorithms for isotonic regression; a unifying framework," *Math. Program.*, vol. 47, nos. 1–3, pp. 425–439, 1990.

[41] M. Baotić, F. Borrelli, A. Bemporad, and M. Morari, "Efficient on-line computation of constrained optimal control," *SIAM J. Control Optim.*, vol. 47, no. 5, pp. 2470–2489, 2008.

[42] M. Baotic, "Optimal control of piecewise affine systems—a multi-parametric approach," Ph.D. dissertation, ETH Zürich, Zurich, Switzerland, 2005.

[43] A. Bemporad and E. Mosca, "Fulfilling hard constraints in uncertain linear systems by reference managing," *Automatica*, vol. 34, no. 4, pp. 451–461, 1998.

[44] A. Bemporad, A. Casavola, and E. Mosca, "Nonlinear control of constrained linear systems via predictive reference management," *IEEE Trans. Automat. Control*, vol. AC-42, no. 3, pp. 340–349, Mar. 1997.

Gionata Cimini (S'13) received the M.Sc. degree (cum laude) in computer and automation engineering and the Ph.D. degree (cum laude) in information engineering, from Università Politecnica delle Marche, Ancona, Italy, in 2012 and 2017, respectively.

In 2014–2015, he was a Guest Ph.D. Scholar at IMT Lucca, Italy, and in 2016, he was a Visiting Ph.D. Scholar at the University of Michigan, Ann Arbor, MI, USA. In 2013, he held a Research Assistant Position at the Information Department, Universit a Politecnica delle Marche, and in 2016 he was a Research Assistant at the Automotive Research Center, University of Michigan. Since 2016, he has been an Advanced Control Specialist at ODYS Srl, Milan, Italy. He has publihed more than 30 papers in international journals, books, and refereed conference proceedings. His research interests include model predictive control, embedded optimization, nonlinear control, system identification and their application to problems in the automotive, and power electronics domains.

Alberto Bemporad (F'10) received the Master's degree in electrical engineering and the Ph.D. degree in control engineering from the University of Florence, Florence, Italy, in 1993 and 1997, respectively.

In 1996/1997, he was with the Center for Robotics and Automation, Department of Systems Science & Mathematics, Washington University, St. Louis, MO, USA. In 1997–1999, he held a Postdoctoral Position at the Automatic Control Laboratory, ETH Zurich, Zurich, Switzerland, where he collaborated as a Senior Researcher until 2002. In 1999–2009, he was with the Department of Information Engineering, University of Siena, Siena, Italy, where he became an Associate Professor in 2005. In 2010–2011, he was with the Department of Mechanical and Structural Engineering, University of Trento, Trento, Italy. Since 2011, he has been a Full Professor at the IMT School for Advanced Studies Lucca, Lucca, Italy, where he served as the Director of the institute in 2012–2015. He spent visiting periods at the University of Michigan and Zhejiang University. In 2011, he cofounded ODYS S.r.l., a consulting and software development company specialized in advanced controls and embedded optimization algorithms. He has published more than 300 papers in the areas of model predictive control, automotive control, hybrid systems, multiparametric optimization, computational geometry, robotics, and finance, and co-inventor of eight patents. He is author or coauthor of various MATLAB toolboxes for model predictive control design, including the model predictive control toolbox and the hybrid toolbox.

Dr. Bemporad was an Associate Editor of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL during 2001–2004 and the Chair of the Technical Committee on Hybrid Systems of the IEEE Control Systems Society in 2002–2010. He received the IFAC High-Impact Paper Award for the 2011–2014 triennial.