Automatica 100 (2019) 29–37

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Complexity and convergence certification of a block principal pivoting method for box-constrained quadratic programs☆

Gionata Cimini [a], Alberto Bemporad [b,*]

[a] ODYS S.r.l., Via Pastrengo, 14, 20159 Milano, Italy
[b] IMT School for Advanced Studies Lucca, Piazza San Francesco 19, 55100 Lucca, Italy

ARTICLE INFO

Article history:
Received 3 February 2018
Received in revised form 27 July 2018
Accepted 1 October 2018

Keywords:
Quadratic programming
Active-set methods
Block principal pivoting methods
Linear model predictive control
Complexity certification

ABSTRACT

Active-set (AS) methods for quadratic programming (QP) are particularly suitable for real-time optimization, as they provide a high-quality solution in a finite number of iterations. However, they add or remove one constraint at each iteration, which makes them inefficient when the number of constraints and variables grows. Block principal pivoting (BPP) methods perform instead multiple changes in the working set in a single iteration, resulting in a much faster execution time. The infeasible primal–dual method proposed by Kunisch and Rendl (KR) (Kunisch and Rendl, 2003) is a BPP method for box-constrained QP that is particularly attractive when reducing the time for finding an accurate solution is crucial, such as in linear model predictive control (MPC) applications. However, the method is guaranteed to converge only under very restrictive sufficient conditions, and tight bounds on the worst-case complexity are not available. For a given set of box-constrained QP's that depend on a vector of parameters, such as those that arise in linear MPC, this paper proposes an algorithm that computes offline the *exact* number of iterations and flops needed by the KR method in the worst-case, and the region of the parameter space for which the method converges or is proved to cycle.

1. Introduction

Model predictive control (MPC) is gaining increasing popularity for controlling multivariable systems subject to input and output constraints (Mayne, 2014). In case of linear prediction models, quadratic costs and linear constraints, the MPC control law is obtained by solving a quadratic programming (QP) problem. Guaranteeing that the command input can be always computed within the allocated sampling time is the most challenging task for the implementation of MPC in a control board. Many QP solvers tailored to embedded applications exist in the literature, such as interior-point (Nesterov & Nemirovskii, 1994; Vanderbei, 1999), operator splitting (Patrinos & Bemporad, 2014; Richter, Jones, & Morari, 2012; Stellato, Banjac, Goulart, Bemporad, & Boyd, 2017) and active-set (AS) (Bartlett & Biegler, 2006; Ferreau, Bock, & Diehl, 2008; Gill, Gould, Murray, Saunders, & Wright, 1984; Goldfarb & Idnani, 1983) methods. The small-size QP problems that arise in most embedded MPC applications encourage the use of AS methods, as they usually outperform the others in terms of speed and solution accuracy. The standard AS method makes steps towards the optimal solution by adding (or removing) one violated (or blocking) constraint from the active set at each iteration. This incremental update of the active set becomes inefficient when the number of constraints and optimization variables grows. For this reason, QP solvers based on block principal pivoting (BPP) have been investigated (Bing, Shiji, Wei, & Keyou, 2015; Buchheim & Trieu, 2014; Curtis, Han, & Robinson, 2014; Hungerlnder & Rendl, 2015; Kunisch & Rendl, 2003), allowing multiple changes in the active set at each iteration and drastically reducing the number of iterations. This paper focuses on one of these methods, the primal–dual infeasible method of Kunisch and Rendl (2003), that we refer to as the KR algorithm. This solver is very simple to code and fast to execute to obtain a solution of a box-constrained QP, such as the one arising in linear MPC with input saturation constraints, or in linear MPC based on I/O models in which both inputs and outputs are treated as optimization variables (Saraf & Bemporad, 2017). Algorithm 1 describes the KR method to solve the box-constrained QP

$$\begin{aligned} \min_z \quad & \frac{1}{2} z'Hz + f'z \\ \text{s.t.} \quad & s \le z \le w \end{aligned} \tag{1}$$

where $z \in \mathbb{R}^n$ is the optimization vector, $H \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix, $s, w, f \in \mathbb{R}^n$, with $s_i < w_i$, $i = 1, \ldots, n$. In

☆ The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Riccardo Scattolini under the direction of Editor Ian R. Petersen.

* Corresponding author.
E-mail addresses: gionata.cimini@odys.it (G. Cimini), alberto.bemporad@imtlucca.it (A. Bemporad).

https://doi.org/10.1016/j.automatica.2018.10.050
0005-1098/

Algorithm 1 z_i denotes the ith component of z, $z_{\mathcal{I}}$ the subvector obtained by collecting the entries z_i for all i in a finite set $\mathcal{I} \subseteq \{1, \dots, n\}$, $A_{\mathcal{I},\mathcal{J}}$ denotes the submatrix obtained by collecting the elements $A_{i,j}$ for all $i \in \mathcal{I}$ and for all $j \in \mathcal{J}$ of a given matrix A.

In spite of its strong attractiveness for embedded MPC, the KR method has the main drawback that it can be guaranteed to converge only under very restrictive conditions, and when it converges it is not possible to estimate beforehand an upper bound on the number of iterations it takes. Complexity certification is a key aspect for implementation of MPC requiring on-line QP optimization, especially in embedded applications where it is crucial to guarantee a solution time within a prescribed limit (Necoara, 2015).

To overcome the lack of tight theoretical bounds on computation complexity, in Cimini and Bemporad (2017) we proposed an algorithm to exactly compute the worst-case iterations and flops of a dual AS solver. Regarding BPP methods, no exact complexity certification exists to the best of our knowledge. Indeed, for what concerns the KR algorithm, the authors in Kunisch and Rendl (2003) claimed an experimental complexity of no more than 12 iterations. In Hintermller, Ito, and Kunisch (2002) the authors prove superlinear convergence of the KR method when the starting point is sufficiently close to the optimal solution, due to the tight relation of BPP methods with semi-smooth Newton methods (Li & Swetits, 1997; Patrinos, Sopasakis, & Sarimveis, 2011). Unfortunately, these results are far to be useful to exactly certify the worst-case execution time of the KR method.

An even more serious issue is the impossibility of guaranteeing convergence of the KR algorithm when solving QP's with more than three variables (Curtis et al., 2014; Gharbia & Gilbert, 2012). Global convergence results, that is convergence from an arbitrary initial guess, have been shown only under the very restrictive assumption that the Hessian matrix of the QP problem is an M-matrix (Hintermller et al., 2002), or it has diagonal dominance (Kunisch & Rendl, 2003). Safeguards and modifications to the basic KR algorithm have been proposed to safely extend its use to a wider class of problems, however these resulted in slower and/or more complex algorithms (Hungerlnder & Rendl, 2015, 2016).

This paper solves the worst-case complexity and the global convergence issues of the KR algorithm, when solving QP problems that depend on a vector θ of parameters, similar to those arising in linear MPC. By following an approach close in spirit with the one reported in Cimini and Bemporad (2017) for a dual AS method, we provide an algorithm that splits the set of parameters θ into polyhedral regions sharing the same number N of iterations required by the solver to compute the solution. Consequently, the largest of such values N provides the exact amount of flops required in the worst-case to solve the given set of QP problems for all possible values of θ within a set Θ of interest. This is helpful not only to exactly certify the complexity of the KR algorithm, but also to help the control designer selecting between MPC based on QP or "explicit" MPC (Bemporad, Morari, Dua, & Pistikopoulos, 2002), and between the KR algorithm and the dual active-set method certified in Cimini and Bemporad (2017), by comparing the requirements in terms of memory and speed. Moreover, the proposed method establishes if the KR algorithm converges for all $\theta \in \Theta$ without any restrictions on the Hessian matrix H other than positive definiteness, or identifies polyhedral subsets of Θ where the KR algorithm cycles. The possibility to add or drop multiple constraints at a time, and to iterate a possible primal and dual infeasible solution, makes the theoretical analysis completely different with respect to solvers that can add only one constraint at a time, and which iterate by maintaining either primal or dual feasibility at intermediate iterations, such as Cimini and Bemporad (2017). Moreover, contrary to Cimini and Bemporad (2017), this algorithm solves the lack of convergence guarantee of the QP solver and takes into account implementation details directly in the convergence analysis, such as primal and dual feasibility tolerances.

Algorithm 1 KR box-constrained QP solver [18]

Input: Tuple (H, f, s, w) defining the QP (1) and initial guess $\mathcal{L}^0, \mathcal{U}^0$ of the active set, feasibility tolerance $\epsilon > 0$.

1: $j \leftarrow 0$;
2: $\mathcal{A} \leftarrow \mathcal{L}^j \cup \mathcal{U}^j, \mathcal{I} \leftarrow \mathcal{K} \setminus \mathcal{A}$;
3: $\begin{bmatrix} z^j_{\mathcal{L}^j} \\ z^j_{\mathcal{U}^j} \end{bmatrix} \leftarrow \begin{bmatrix} s_{\mathcal{L}^j} \\ w_{\mathcal{U}^j} \end{bmatrix}$; $z^j_{\mathcal{I}} \leftarrow -H^{-1}_{\mathcal{I},\mathcal{I}}(f_{\mathcal{I}} + H_{\mathcal{I},\mathcal{A}} z^j_{\mathcal{A}})$;
$\pi^j_{\mathcal{I}} \leftarrow 0$; $\pi^j_{\mathcal{A}} \leftarrow -f_{\mathcal{A}} - H_{\mathcal{A},\mathcal{K}} z^j$;
4: $\mathcal{L}^{j+1} \leftarrow \{i \in \mathcal{K} \setminus \mathcal{L} : z^j_i \le s_i - \epsilon\} \cup \{i \in \mathcal{L}^j : \pi^j_i \le -\epsilon\}$;
$\mathcal{U}^{j+1} \leftarrow \{i \in \mathcal{K} \setminus \mathcal{U} : z^j_i \ge w_i + \epsilon\} \cup \{i \in \mathcal{U}^j : \pi^j_i \ge \epsilon\}$;
5: **if** $(\mathcal{L}^{j+1} \cup \mathcal{U}^{j+1}) = \emptyset$ **return** $z^* \leftarrow z^j$, **end if**;
6: **if** $(\mathcal{L}^{j+1}, \mathcal{U}^{j+1}) = (\mathcal{L}^h, \mathcal{U}^h)$ for some $h \le j$ **return "cycling"**, **end if**;
7: $j \leftarrow j + 1$;
8: **go to** Step 2;

Output: solution z^* and optimal active sets $\mathcal{L}^* \leftarrow \mathcal{L}^j, \mathcal{U}^* \leftarrow \mathcal{U}^j$ (within the feasibility tolerance ϵ) or "cycling" condition detected, number $N = j + 1$ of iterations.

The rest of the paper is organized as follows. After detailing the KR algorithm in Section 2, Section 3 presents the certification algorithm. Section 4 shows the application of our approach to MPC problems, and Section 5 concludes the paper.

2. KR algorithm

In this section we describe in detail the KR Algorithm 1 when applied to solve the parameter-dependent box-constrained QP problem

$$\begin{aligned} \min_z \quad & \frac{1}{2} z' H z + (F\theta + f)' z \\ \text{s.t.} \quad & S\theta + s \le z \le W\theta + w \end{aligned} \tag{2}$$

where $\theta \in \mathbb{R}^{n_\theta}$ is a parameter vector within a given polyhedral set $\Theta = \{\theta \in \mathbb{R}^{n_\theta} \mid E_\theta \theta \le e_\theta\}$ of interest, E_θ, e_θ are matrices of appropriate dimensions, $F \in \mathbb{R}^{n \times n_\theta}$ is a given matrix, $S, W \in \mathbb{R}^{n \times n_\theta}$, and $S_i\theta + s_i < W_i\theta + w_i$, for all $\theta \in \Theta$ and $i = 1, \dots, n$, with S_i denoting the ith row of matrix S. In Algorithm 1, $\mathcal{K} = \{1, \dots, n\}$ denotes the set of indices corresponding to the n optimization variables, and f is substituted by $F\theta + f$, s by $S\theta + s$, w by $W\theta + w$ when solving (2) for a given $\theta \in \Theta$. In Algorithm 1 we have introduced Step 6 to detect possible "cycling" conditions, under which the algorithm fails in finding a solution, and made an explicit reference to the tolerance ϵ used in recognizing violation of primal and dual feasibility.

Definition 1. Given the QP problem (2) and an intermediate solution $\bar{z} \in \mathbb{R}^n$, the lower constraint $S_i\theta + s_i \le z_i$ is *active* at $\bar{z}$ if the equality $S_i\theta + s_i = \bar{z}_i$ holds, otherwise it is *inactive*. Similarly, the upper constraint $W_i\theta + w_i \ge z_i$ is active at $\bar{z}$ if $W_i\theta + w_i = \bar{z}_i$ holds, otherwise it is inactive.

Definition 2. Given the QP problem (2) and the intermediate primal solution $\bar{z} \in \mathbb{R}^n$, the *lower active set* $\mathcal{L}(\bar{z})$ and the *upper active set* $\mathcal{U}(\bar{z})$ are defined as:

$$\mathcal{L}(\bar{z}) = \{i \in \mathcal{K} \mid \bar{z} = S_i\theta + s_i\} \tag{3a}$$

$$\mathcal{U}(\bar{z}) = \{i \in \mathcal{K} \mid \bar{z} = W_i\theta + w_i\} \tag{3b}$$

where $\mathcal{L}(\bar{z}) \cap \mathcal{U}(\bar{z}) = \emptyset$ by construction. Accordingly, $\mathcal{A}(\bar{z}) = \mathcal{L}(\bar{z}) \cup \mathcal{U}(\bar{z})$ is defined as the *active set*.

Definition 3. Given the QP problem (2), the intermediate primal solution $\bar{z} \in \mathbb{R}^n$ and the sets of lower and upper active constraints (3), $\mathcal{I}(\bar{z})$ is the *inactive set*

$$\mathcal{I}(\bar{z}) = \{i \in \mathcal{K} \mid \bar{z}_i \neq S_i\theta + s_i,\ \bar{z}_i \neq W_i\theta + w_i\} \tag{4}$$

or equivalently $\mathcal{I}(\bar{z}) = \mathcal{K} \setminus \mathcal{A}(\bar{z})$.

The Karush–Kuhn–Tucker (KKT) optimality conditions associated with problem (2) are

$$Hz + F\theta + f + \alpha + \beta = 0 \tag{5a}$$
$$z_i - S_i\theta - s_i \geq 0, \quad z_i - W_i\theta - w_i \leq 0 \tag{5b}$$
$$\alpha_i(z_i - S_i\theta - s_i) = 0, \quad \beta_i(z_i - W_i\theta - w_i) = 0 \tag{5c}$$
$$\alpha_i \leq 0, \quad \beta_i \geq 0, \quad i = 1, \ldots, n \tag{5d}$$

where $\alpha \in \mathbb{R}^n$ and $\beta \in \mathbb{R}^n$ are the Lagrange multipliers for the lower and the upper bounds, respectively. As $\alpha_i\beta_i = 0$ due to the assumption that $S_i\theta + s_i < W_i\theta + w_i, \forall\theta \in \Theta$ and $i = 1, \ldots, n$, we define $\pi = \alpha + \beta$ and rewrite (5) in the equivalent form

$$Hz + F\theta + f + \pi = 0 \tag{6a}$$
$$z_i - S_i\theta - s_i \geq 0, \quad z_i - W_i\theta - w_i \leq 0 \tag{6b}$$
$$\min\{\pi_i, 0\}(z_i - S_i\theta - s_i) = 0 \tag{6c}$$
$$\max\{\pi_i, 0\}(z_i - W_i\theta - w_i) = 0, \quad i = 1, \ldots, n \tag{6d}$$

which is simpler to handle, requiring the use of a single vector π of dual variables. For given active sets $\mathcal{L}, \mathcal{U}$, by (6) we have

$$z_{\mathcal{L}} = S_{\mathcal{L}}\theta + s_{\mathcal{L}} \tag{7a}$$
$$z_{\mathcal{U}} = W_{\mathcal{U}}\theta + w_{\mathcal{U}} \tag{7b}$$
$$\pi_{\mathcal{I}} = 0 \tag{7c}$$

where $S_{\mathcal{L}}$ defines the submatrix obtained by collecting the rows S_i for all $i \in \mathcal{L}$ of a given matrix S. By expanding (6a) to

$$\begin{bmatrix} H_{\mathcal{A},\mathcal{A}} & H_{\mathcal{A},\mathcal{I}} \\ H_{\mathcal{I},\mathcal{A}} & H_{\mathcal{I},\mathcal{I}} \end{bmatrix} \begin{bmatrix} z_{\mathcal{A}} \\ z_{\mathcal{I}} \end{bmatrix} + \begin{bmatrix} F_{\mathcal{A}}\theta + f_{\mathcal{A}} \\ F_{\mathcal{I}}\theta + f_{\mathcal{I}} \end{bmatrix} + \begin{bmatrix} \pi_{\mathcal{A}} \\ \pi_{\mathcal{I}} \end{bmatrix} = 0 \tag{8}$$

with $H_{\mathcal{I},\mathcal{I}}$ invertible being a principal submatrix of a positive definite matrix, from (7c)–(8) we get

$$z_{\mathcal{I}} = -H_{\mathcal{I},\mathcal{I}}^{-1}\left(F_{\mathcal{I}}\theta + f_{\mathcal{I}} + H_{\mathcal{I},\mathcal{A}}z_{\mathcal{A}}\right) \tag{9a}$$
$$\pi_{\mathcal{A}}^j = -F_{\mathcal{A}}\theta - f_{\mathcal{A}} - H_{\mathcal{A}}z \tag{9b}$$

Starting from an arbitrary initial guess of the active set $\mathcal{A}^0 = \mathcal{L}^0 \cup \mathcal{U}^0$, Algorithm 1 iteratively solves the KKT-conditions (6) until the optimal active set $\mathcal{A}^*$ is found. At each iteration *all* the violated constraints on primal and dual variables are added to the current active set, therefore allowing for multiple changes, see Step 4.

Contrary to most active-set methods, in Algorithm 1 *any* initial guess for $\mathcal{L}^0$ and $\mathcal{U}^0$ can be used, as long as it satisfies the condition $\mathcal{L}^0 \cap \mathcal{U}^0 = \emptyset$. Experimental evidence shows that the number of iterations is affected only marginally by the initial guess (Hungerlnder & Rendl, 2015; Kunisch & Rendl, 2003). For this reason, we assume in this paper to always start with $\mathcal{L}^0 \cup \mathcal{U}^0 = \mathcal{K}$, so to avoid solving the linear system (9a) at the first iteration, which is the most costly operation of the KR algorithm.

While extensive experimental results reported in Gharbia and Gilbert (2012), Hintermller et al. (2002), Hungerlnder and Rendl (2015) and Kunisch and Rendl (2003) show promising performance of the KR algorithm, its global convergence cannot be guaranteed, as demonstrated in Gharbia and Gilbert (2012), unless sufficient conditions on the problems are imposed (Hintermller et al., 2002; Kunisch & Rendl, 2003). We recall such conditions in the following proposition.

Proposition 1. *Consider the QP problem (2) and let $\lambda_{\min}(H)$ and $\lambda_{\max}(H)$ be respectively the smallest and the largest eigenvalues of H, with $\lambda_{\min}(H) > 0$. Define $v = \arg\max\{\|H_{\mathcal{N}_i,\mathcal{K}\setminus\mathcal{N}_i}\| : \mathcal{N}_i \in 2^{\mathcal{K}} \setminus \{\emptyset, \mathcal{K}\}\}$, where $2^{\mathcal{K}}$ denotes the powerset of $\mathcal{K}$ and $\|H\|$ the ℓ^2-norm of H. Let $D = \operatorname{diag}(H)$ be the diagonal matrix collecting the diagonal elements h_{ii} of H. Define $h^{\min} = \min_{i\in\mathcal{K}}\{h_{ii}\}$ and $r = \|D - H\|$. Then, Algorithm 1 is guaranteed to converge globally in a finite number steps if at least one of the following sufficient conditions holds:*

(C_1) $\frac{\lambda_{\max}(H)}{\lambda_{\min}(H)} < \left(\frac{\lambda_{\min}(H)}{v}\right)^2 - 1$;

(C_2) $\frac{\lambda_{\max}(H)}{\lambda_{\min}(H)} < \left(\frac{h^{\min}}{r}\right)^2 - 1$;

(C_3) *H is an M-matrix, that is $H_{i,j} \leq 0$, $\operatorname{Re}(\lambda_i(H)) \geq 0$, $\forall i, j = 1, \ldots, n$, $i \neq j$, with $\operatorname{Re}(\lambda)$ the real part of λ.*

The conditions of Proposition 1 are restrictive for problems coming from MPC formulations, as rarely the Hessian matrix is diagonally dominant. Moreover, even though the convergence proof in Proposition 1 does not depend on assumptions on strict complementarity, the algorithm is always implemented by considering the small tolerance ϵ for checking primal and dual feasibility, as described in Algorithm 1, which makes strict complementarity a necessary condition for Proposition 1 to hold. Despite the unavailability of a convergence proof, experiments on a large set of contrived problems show that the KR algorithm often converges even when the conditions in Proposition 1 do not hold, and when strict complementarity is not verified (Hungerlnder & Rendl, 2015; Kunisch & Rendl, 2003). In the next section we solve the question on whether the KR algorithm converges (or cycles), given the implementation in Algorithm 1, and estimate tightly how many iterations N it takes when it does, so that it can be safely and effectively used in embedded applications.

3. Complexity certification of KR algorithm

We want to characterize the parametric behavior of Algorithm 1 when f is replaced by $F\theta + f$, s by $S\theta + s$, and w by $W\theta + w$. To this end we define the functions $z^j, \pi^j : \mathbb{R}^{n_\theta} \to \mathbb{R}^n$ such that $z^j(\theta), \pi^j(\theta)$ are the values of z^j, π^j, respectively, generated by the algorithm for a given θ at the jth iteration. Similarly, we define $z^*, \pi^* : \mathbb{R}^{n_\theta} \to (\mathbb{R}\cup\{+\infty\})^n$ the optimal primal and dual solutions (within the feasibility tolerance ϵ) determined by the algorithm, where we set $z_i^*(\theta) = \pi_i^*(\theta) = +\infty$ for all $i = 1, \ldots, n$ in case of cycling. We first introduce the following definitions (cf. Cimini & Bemporad, 2017).

Definition 4. Given a polyhedron $\Theta \subseteq \mathbb{R}^{n_\theta}$, the collection of sets $\{\Theta_1, \ldots, \Theta_s\}$ is 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 $\mathring{\Theta}_i \cap \mathring{\Theta}_j = \emptyset$, $\forall i, j = 1, \ldots, s$, $i \neq j$, where $\mathring{\Theta}_i$ denotes the interior of Θ_i.

Definition 5. A function $n : \Theta \to \mathbb{N}$, $\Theta \subseteq \mathbb{R}^{n_\theta}$, is *integer piecewise constant* (IPWC) if there exist a polyhedral partition $\Theta_1, \ldots, \Theta_s$ of Θ and a function $\sigma : 2^{\{1,\ldots,s\}} \setminus \emptyset \to \{1, \ldots, s\}$, where $2^{\{1,\ldots,s\}}$ is the powerset of $\{1, \ldots, s\}$, such that

$$\begin{aligned} n(\theta) &= \sigma(\mathcal{J}(\theta)) \\ \mathcal{J}(\theta) &= \{i \in \{1, \ldots, s\} : \theta \in \Theta_i\} \end{aligned} \tag{10}$$

for all $\theta \in \Theta$, with $\sigma(\{i\}) = i$, for all $i = 1, \ldots, s$.

Note that the selection function σ in (10) avoids possible multiple definition of $n(\theta)$ on overlapping boundaries $\Theta_i \cap \Theta_j \neq \emptyset$.

Definition 6. A function $\gamma : \Theta \to \mathbb{R}^n$, $\Theta \subseteq \mathbb{R}^{n_\theta}$, is *piecewise affine* (PWA) if there exists an IPWC function $n : \Theta \to \{n_1, \ldots, n_s\}$

$$\Omega(\bar{\mathcal{L}},\bar{\mathcal{U}}) = \left\{ \theta \in \bar{\Theta} : \begin{array}{l} z_i \le S_i\theta + s_i - \epsilon,\ \forall i \in \mathcal{L}_2, \\ z_i > S_i\theta + s_i - \epsilon,\ \forall i \in \mathcal{L}_4, \\ \pi_j \le -\epsilon,\ \forall j \in \mathcal{L}_1 \\ \pi_j > -\epsilon,\ \forall j \in \mathcal{L}_3 \\ z_i \ge W_i\theta + w_i + \epsilon,\ \forall i \in \mathcal{U}_1, \\ z_i < W_i\theta + w_i + \epsilon,\ \forall i \in \mathcal{U}_3, \\ \pi_j \ge \epsilon,\ \forall j \in \mathcal{U}_2 \\ \pi_j < \epsilon,\ \forall j \in \mathcal{U}_4 \end{array} \right\} = \left\{ \theta \in \bar{\Theta} : \begin{array}{l} \begin{bmatrix} A_{\mathcal{L}_2} - S_{\mathcal{L}_2} \\ B_{\mathcal{L}_1} \\ W_{\mathcal{U}_1} - A_{\mathcal{U}_1} \\ -B_{\mathcal{U}_2} \end{bmatrix} \theta \le \begin{bmatrix} s_{\mathcal{L}_2} - a_{\mathcal{L}_2} \\ -b_{\mathcal{L}_1} \\ a_{\mathcal{U}_1} - w_{\mathcal{U}_1} \\ b_{\mathcal{U}_2} \end{bmatrix} - \epsilon\mathbf{1}, \\ \begin{bmatrix} S_{\mathcal{L}_4} - A_{\mathcal{L}_4} \\ -B_{\mathcal{L}_3} \\ A_{\mathcal{U}_3} - W_{\mathcal{U}_3} \\ B_{\mathcal{U}_4} \end{bmatrix} \theta < \begin{bmatrix} a_{\mathcal{L}_4} - s_{\mathcal{L}_4} \\ b_{\mathcal{L}_3} \\ w_{\mathcal{U}_3} - a_{\mathcal{U}_3} \\ -b_{\mathcal{U}_4} \end{bmatrix} - \epsilon\mathbf{1} \end{array} \right\} \tag{12}$$

Box I.

defined over a polyhedral partition $\Theta_1, \dots, \Theta_s$ of Θ and s pairs (G_i, g_i), $G_i \in \mathbb{R}^{n\times n_\theta}$, $g_i \in \mathbb{R}^n$, $i \in \{n_1, \dots, n_s\}$, such that

$$\gamma(\theta) = G_{n(\theta)}\theta + g_{n(\theta)} \tag{11}$$

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

The following Lemma 1 characterizes the generic iteration j of the certification algorithm.

Lemma 1. *Let $\bar{\Theta}$ be a polyhedron, $\bar{\Theta} \subseteq \mathbb{R}^{n_\theta}$, and let $\mathcal{L}^j, \mathcal{U}^j \subseteq \mathcal{K}$ such that $\mathcal{L}^j \cap \mathcal{U}^j = \emptyset$. Then*

(i) *$z^j(\theta)$, $\pi^j(\theta)$ defined by Step 3 of Algorithm 1 are affine functions of θ.*
(ii) *For all combinations $\bar{\mathcal{L}}, \bar{\mathcal{U}} \subseteq \mathcal{K}$ such that $\bar{\mathcal{L}} \cap \bar{\mathcal{U}} = \emptyset$ the closure $\bar{\Omega}(\bar{\mathcal{L}}, \bar{\mathcal{U}})$ of the corresponding set $\Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}}) \triangleq \{\theta \in \bar{\Theta}: \mathcal{L}^{j+1}(\theta) = \bar{\mathcal{L}}, \mathcal{U}^{j+1}(\theta) = \bar{\mathcal{U}}\}$ with $\mathcal{L}^{j+1}(\theta), \mathcal{U}^{j+1}(\theta)$ defined by Step 4 of Algorithm 1 is a (possibly empty) polyhedron.*
(iii) *The collection of all nonempty closed polyhedra $\{\bar{\Omega}(\bar{\mathcal{L}}, \bar{\mathcal{U}})\}$ is a polyhedral partition of $\bar{\Theta}$.*
(iv) *$z^{j+1}, \pi^{j+1} : \bar{\Theta} \to \mathbb{R}^n$ are piecewise affine functions of θ.*

Proof. (i) The fact that $z^j(\theta) = A\theta + a$, $\pi^j(\theta) = B\theta + b$ for some A, a, B, b immediately follows from the way they are defined by Step 3 of Algorithm 1. In particular $z^j_{\mathcal{L}^j}(\theta) = S_{\mathcal{L}^j}\theta + s_{\mathcal{L}^j}$, $z^j_{\mathcal{U}^j}(\theta) = S_{\mathcal{U}^j}\theta + s_{\mathcal{U}^j}$, $\pi^j_{\mathcal{I}^j}(\theta) = 0$, and consequently $z_{\mathcal{I}^j} = C\theta + c$ and $\pi_{\mathcal{A}^j} = D\theta + d$ for all $\theta \in \bar{\Theta}$, with $C = -H^{-1}_{\mathcal{I}^j,\mathcal{I}^j}(H_{\mathcal{I}^j,\mathcal{A}^j}\begin{bmatrix} S_{\mathcal{L}^j} \\ S_{\mathcal{U}^j} \end{bmatrix} + F_{\mathcal{I}^j})$, $c = -H^{-1}_{\mathcal{I}^j,\mathcal{I}^j}(H_{\mathcal{I}^j,\mathcal{A}^j}\begin{bmatrix} s_{\mathcal{L}^j} \\ s_{\mathcal{U}^j} \end{bmatrix} + f_{\mathcal{I}^j})$, $D = -(H_{\mathcal{A}^j}A + F_{\mathcal{A}^j})$ and $d = -H_{\mathcal{A}^j}a - f_{\mathcal{A}^j}$.

(ii) Let $\mathcal{L}_1 = \bar{\mathcal{L}} \cap \mathcal{L}^j$, $\mathcal{L}_2 = \bar{\mathcal{L}} \cap (\mathcal{K} \setminus \mathcal{L}^j)$, $\mathcal{L}_3 = (\mathcal{K} \setminus \bar{\mathcal{L}}) \cap \mathcal{L}^j$, $\mathcal{L}_4 = (\mathcal{K}\setminus\bar{\mathcal{L}})\cap(\mathcal{K}\setminus\mathcal{L}^j)$, $\mathcal{U}_1 = \bar{\mathcal{U}}\cap\mathcal{U}^j$, $\mathcal{U}_2 = \bar{\mathcal{U}}\cap(\mathcal{K}\setminus\mathcal{U}^j)$, $\mathcal{U}_3 = (\mathcal{K}\setminus\bar{\mathcal{U}})\cap\mathcal{U}^j$, and $\mathcal{U}_4 = (\mathcal{K}\setminus\bar{\mathcal{U}})\cap(\mathcal{K}\setminus\mathcal{U}^j)$, where clearly $\bar{\mathcal{L}} = \mathcal{L}_1 \cup \mathcal{L}_2$, $\bar{\mathcal{U}} = \mathcal{U}_1 \cup \mathcal{U}_2$, $\mathcal{K} \setminus \bar{\mathcal{L}} = \mathcal{L}_3 \cup \mathcal{L}_4$, $\mathcal{K} \setminus \bar{\mathcal{U}} = \mathcal{U}_3 \cup \mathcal{U}_4$. Then, the set $\Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}})$ is a polyhedron defined as in (12) in Box I, where $\mathbf{1}$ denotes a vector whose entries are all one.

(iii) We need to prove that the union of $\{\Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}})\}$ includes Θ, and that their interiors $\{\mathring{\Omega}(\bar{\mathcal{L}}, \bar{\mathcal{U}})\}$ do not overlap. Regarding the first, for any given $\theta \in \Omega$ we also have at least that $\theta \in \Omega(\mathcal{L}^{j+1}(\theta), \mathcal{U}^{j+1}(\theta))$. Consider two arbitrary combinations $(\bar{\mathcal{L}}, \bar{\mathcal{U}}) \neq (\tilde{\mathcal{L}}, \tilde{\mathcal{U}})$, $\bar{\mathcal{L}}\cap\bar{\mathcal{U}} = \emptyset$, $\tilde{\mathcal{L}}\cap\tilde{\mathcal{U}} = \emptyset$, such that $\Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}})$, $\Omega(\tilde{\mathcal{L}}, \tilde{\mathcal{U}}) \neq \emptyset$. Since $(\bar{\mathcal{L}}, \bar{\mathcal{U}}) \neq (\tilde{\mathcal{L}}, \tilde{\mathcal{U}})$, there exists an index i such that (a) $i \in \bar{\mathcal{L}}, i \notin \tilde{\mathcal{L}}$, or (b) $i \in \tilde{\mathcal{L}}, i \notin \bar{\mathcal{L}}$, or (c) $i \in \bar{\mathcal{U}}, i \notin \tilde{\mathcal{U}}$, or (d) $i \in \tilde{\mathcal{U}}, i \notin \bar{\mathcal{U}}$. In case (a), either $i \in \mathcal{L}_1$ or $i \in \mathcal{L}_2$, and either $i \in \tilde{\mathcal{L}}_3 \triangleq (\mathcal{K} \setminus \tilde{\mathcal{L}}) \cap \mathcal{L}^j$ or $i \in \tilde{\mathcal{L}}_4 \triangleq (\mathcal{K} \setminus \tilde{\mathcal{L}}) \cap (\mathcal{K} \setminus \mathcal{L}^j)$. If $i \in \mathcal{L}_1$ then $\pi_i(\theta) \le -\epsilon$ for all $\theta \in \Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}})$, but on the other hand either $\pi_i(\theta) > -\epsilon$ ($i \in \tilde{\mathcal{L}}_3$) or $\pi_i(\theta) = 0$ ($i \in \tilde{\mathcal{L}}_4$). Therefore, $\Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}}) \cap \Omega(\tilde{\mathcal{L}}, \tilde{\mathcal{U}}) = \emptyset$, which also implies $\mathring{\Omega}(\bar{\mathcal{L}}, \bar{\mathcal{U}}) \cap \mathring{\Omega}(\tilde{\mathcal{L}}, \tilde{\mathcal{U}}) = \emptyset$. The other three cases are similar.

(iv) If $\theta \in \mathring{\Omega}(\bar{\mathcal{L}}, \bar{\mathcal{U}})$ for only one pair $(\bar{\mathcal{L}}, \bar{\mathcal{U}})$ such that $\bar{\mathcal{L}} \cap \bar{\mathcal{U}} = \emptyset$ then z^{j+1}, π^{j+1} is uniquely defined and, as shown in (i), affine. It remains to characterize $z^{j+1}(\theta)$, $\pi^{j+1}(\theta)$ on overlapping boundaries. Consider a generic intersection $P = \cap_{i=1}^K \bar{\Omega}(\mathcal{L}_i, \mathcal{U}_i)$, $K \ge 2$. For all $\theta \in P$ Step 4 of Algorithm 1 always selects the same new combination $\mathcal{L}^{j+1}(\theta) = \mathcal{L}_i, \mathcal{U}^{j+1}(\theta) = \mathcal{U}_i$ that has the largest cardinality, which defines a selection rule σ. Therefore, a piecewise constant function $n : \mathbb{R}^{n_\theta} \to \mathcal{N}$ exists such that $\mathcal{L}^{j+1}(\theta) = \mathcal{L}_{n(\theta)}$, $\mathcal{U}^{j+1}(\theta) = \mathcal{U}_{n(\theta)}$. Hence, the corresponding functions z^{j+1}, π^{j+1} are piecewise affine. ■

3.1. Algorithm for complexity certification

Let $\Theta \subseteq \mathbb{R}^{n_\theta}$ be the set of parameters of interest for which we want to analyze problem (2). We present a recursive algorithm that analyzes the behavior of the KR solver in a parametric way. The idea is to apply recursively Lemma 1 on each nonempty set $\bar{\Omega}(\bar{\mathcal{L}}, \bar{\mathcal{U}})$ generated as in (12). We start with $\bar{\Theta} = \Theta$ and initial combinations $\mathcal{L}^0, \mathcal{U}^0$. At the next recursion, Lemma 1 is applied with $\bar{\Theta} = \bar{\Omega}(\bar{\mathcal{L}}, \bar{\mathcal{U}})$ and $\mathcal{L}^j = \bar{\mathcal{L}}, \mathcal{U}^j = \bar{\mathcal{U}}$ for each nonempty set $\bar{\Omega}(\bar{\mathcal{L}}, \bar{\mathcal{U}})$.

Let us consider the tuple $T = (\Theta, \mathcal{L}, \mathcal{U}, \tilde{\mathcal{A}}, N)$, where Θ is the polyhedron currently analyzed, $\mathcal{L}, \mathcal{U}$ the current combination of $\mathcal{L}^j$ and $\mathcal{U}^j$, N is the number of recursions performed to get such Θ, $\mathcal{L}^j, \mathcal{U}^j$, and $\tilde{\mathcal{A}}^j = \{(\mathcal{L}^h, \mathcal{U}^h)\}_{h=0}^{N-1}$ denotes the collection of lower and upper active-set pairs considered during the preceding $N-1$ recursions.

Lemma 2. *Under the hypothesis of Lemma 1, the following holds:*

(i) *if $\Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}}) \neq \emptyset$ and $(\bar{\mathcal{L}}, \bar{\mathcal{U}}) = (\mathcal{L}^h, \mathcal{U}^h)$ for some past recursion h then Algorithm 1 cycles for all $\theta \in \Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}})$;*
(ii) *if $\Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}}) = \emptyset$ for all $\bar{\mathcal{L}}, \bar{\mathcal{U}} \subseteq \mathcal{K}$ such that $\bar{\mathcal{L}}\cap\bar{\mathcal{U}} = \emptyset, \bar{\mathcal{L}}, \bar{\mathcal{U}} \neq \emptyset$, then $z^*(\theta) = z^j(\theta), \pi^*(\theta) = \pi^j(\theta), \forall\theta \in \bar{\Theta}$.*

Proof. Property (i) trivially follows from Step 6, (ii) from Step 5. ■

According to Lemma 2, a recursion is stopped on a set $\Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}}) \neq \emptyset$ if $(\bar{\mathcal{L}}, \bar{\mathcal{U}}) = (\mathcal{L}^h, \mathcal{U}^h)$ for some $(\mathcal{L}^h, \mathcal{U}^h) \in \tilde{\mathcal{A}}^j$ (cycling condition), and no further recursion occurs from the current tuple when $\Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}}) = \emptyset$ for all $\bar{\mathcal{L}}, \bar{\mathcal{U}} \subseteq \mathcal{K}$ such that $\bar{\mathcal{L}}\cap\bar{\mathcal{U}} = \emptyset, \bar{\mathcal{L}}, \bar{\mathcal{U}} \neq \emptyset$, in which case the affine expression of the optimal solution $z^*(\theta) = z^j(\theta), \pi^*(\theta) = \pi^j(\theta)$ and the number of iterations $N(\theta), \theta \in \bar{\Theta}$, are also available.

The above reasoning is formulated in Algorithm 2, that recursively builds two collections $\mathbb{T}$ and $\bar{\mathbb{T}}$ of tuples by dividing Θ into polyhedra where the KR algorithm finds the optimal solution or cycles. In the sequel, we denote by $\mathbb{T}$ the set of tuples corresponding to regions where KR converges, and by $\bar{\mathbb{T}}$ the set of tuples where it does not. For a given leaf tuple $T = (\bar{\Theta}, \mathcal{L}^j, \mathcal{U}^j, \phi, j)$, with $T \in \mathbb{T}\cup\bar{\mathbb{T}}$, $N(T)$ denotes the number j of iterations performed by KR, and $\Phi(T)$ the corresponding number ϕ of flops, for all $\theta \in \bar{\Theta}$.

Since in embedded optimization the maximum *time* needed to solve problem (2) rather than the maximum number of *iterations* is of interest, Algorithm 2 also keeps track at Step 4.3 of the computational complexity (number of CPU flops) of each iteration j. This may be different from one iteration to another, as it depends

Algorithm 2 Certification of the KR algorithm

Input: H, F, f, S, s, W, w defining problem (2), polyhedron Θ of parameters, initial active-set guess $\mathcal{L}^0, \mathcal{U}^0$.

1. $\mathbb{T} \leftarrow \emptyset, \bar{\mathbb{T}} \leftarrow \emptyset$;
2. **execute** *analyze*$(\Theta, \mathcal{L}^0, \mathcal{U}^0, \emptyset, 0, 1)$;
3. **end**.

4. **procedure** *analyze*$(\bar{\Theta}, \mathcal{L}^j, \mathcal{U}^j, \tilde{\mathcal{A}}, \phi, j)$

4.1. $\mathcal{A} \leftarrow \mathcal{L}^j \cup \mathcal{U}^j, \mathcal{I} \leftarrow \mathcal{K} \setminus \mathcal{A}$;

4.2. $A_{\mathcal{L}^j} \leftarrow S_{\mathcal{L}^j}, a_{\mathcal{L}^j} \leftarrow s_{\mathcal{L}^j}$;
$A_{\mathcal{U}^j} \leftarrow W_{\mathcal{U}^j}, a_{\mathcal{U}^j} \leftarrow w_{\mathcal{U}^j}$;
$A_{\mathcal{I}} \leftarrow -H_{\mathcal{I},\mathcal{I}}^{-1}(F_{\mathcal{I}} + H_{\mathcal{I},\mathcal{A}}A_{\mathcal{A}})$;
$a_{\mathcal{I}} \leftarrow -H_{\mathcal{I},\mathcal{I}}^{-1}(f_{\mathcal{I}} + H_{\mathcal{I},\mathcal{A}}a_{\mathcal{A}})$;
$B_{\mathcal{K}\setminus\mathcal{L}^j} \leftarrow 0, b_{\mathcal{K}\setminus\mathcal{L}^j} \leftarrow 0$;
$B_{\mathcal{K}\setminus\mathcal{U}^j} \leftarrow 0, b_{\mathcal{K}\setminus\mathcal{U}^j} \leftarrow 0$;
$B_{\mathcal{A}} \leftarrow -F_{\mathcal{A}} - H_{\mathcal{A},\mathcal{K}}A$;
$b_{\mathcal{A}} \leftarrow -f_{\mathcal{A}} - H_{\mathcal{A},\mathcal{K}}a$;

4.3. $\phi \leftarrow \phi + \varphi(\mathcal{L}^j, \mathcal{U}^j)$;

4.4. **enumerate** all $(\bar{\mathcal{L}}, \bar{\mathcal{U}})$ such that $\Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}}) \neq \emptyset$, where $\Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}})$ is defined by (12);

4.5. **if** all polyhedra $\Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}})$ are empty **set** $\mathbb{T} \leftarrow \mathbb{T} \cup \{(\bar{\Theta}, \mathcal{L}^j, \mathcal{U}^j, j)\}$ and **return**;

4.6. For all $(\bar{\mathcal{L}}, \bar{\mathcal{U}})$ such that $\Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}}) \neq \emptyset$, **do**:

4.6.1. **if** $(\bar{\mathcal{L}}, \bar{\mathcal{U}}) \in \tilde{\mathcal{A}}$ **set** $\bar{\mathbb{T}} \leftarrow \bar{\mathbb{T}} \cup \{(\Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}}), \bar{\mathcal{L}}, \bar{\mathcal{U}}, j)\}$ and **return**;

4.6.2. **else** $\tilde{\mathcal{A}} \leftarrow \tilde{\mathcal{A}} \cup \{(\bar{\mathcal{L}}, \bar{\mathcal{U}})\}$ and **execute** *analyze*$(\Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}}), \bar{\mathcal{L}}, \bar{\mathcal{U}}, \tilde{\mathcal{A}}, \phi, j+1)$;

4.6.3. **end if**

4.7. **end for**

Output: list $\mathbb{T}$ of *optimal* tuples, list $\bar{\mathbb{T}}$ of tuples where the algorithm cycles.

on the dimension of the current active set $\mathcal{L}^j, \mathcal{U}^j$. We denote by $\varphi(\mathcal{L}, \mathcal{U})$ the number of flops required to compute Step 3 of Algorithm 1 for a given combination $(\mathcal{L}, \mathcal{U})$. As the number of possible combinations $(\bar{\mathcal{L}}, \bar{\mathcal{U}})$ is finite, Step 4.6.1 guarantees that Algorithm 2 always terminates in a finite number of recursions. When it stops, by property (iii) of Lemma 1 the entire given polyhedron Θ gets recursively partitioned into polyhedral subsets $\Omega(\bar{\mathcal{L}}, \bar{\mathcal{U}})$.

Based on the results proved in the previous lemmas, next Theorem 1 summarizes the property of Algorithm 2 of fully characterizing the behavior of Algorithm 1 in solving problem (2).

Theorem 1. *Consider the QP problem (2) with $\theta \in \Theta$ and let $\mathbb{T}, \bar{\mathbb{T}}$ be the lists of leaf tuples generated by Algorithm 2, then the following properties are verified:*

(i) *Algorithm 1 terminates in at most*

$$N_{KR}^{\max} = \max_{T \in \mathbb{T} \cup \bar{\mathbb{T}}} N(T) \tag{13a}$$

iterations, and executes at most

$$n_{KR}^{\max} = \max_{T \in \mathbb{T} \cup \bar{\mathbb{T}}} \Phi(T) \tag{13b}$$

flops.

(ii) *Algorithm 1 converges for all $\theta \in \Theta$ if and only if $\bar{\mathbb{T}} = \emptyset$. Otherwise, Algorithm 1 cycles for all $\theta \in \Theta$ such that $\theta \in \bar{\Theta}$ for some $(\bar{\Theta}, \mathcal{L}^j, \mathcal{U}^j, \phi, j) \in \bar{\mathbb{T}}$.*

(iii) *Let $\theta \in \bar{\Theta}$ for some $T = (\bar{\Theta}, \mathcal{L}^j, \mathcal{U}^j, \phi, j) \in \mathbb{T} \cup \bar{\mathbb{T}}$. Then Algorithm 1 terminates after executing exactly $n_{KR} \triangleq \Phi(T)$ flops.*

Proof. (i) Since $\mathbb{T}$ and $\bar{\mathbb{T}}$ are finite sets, and $N(T), \Phi(T)$ are bounded for all $T \in \mathbb{T} \cup \bar{\mathbb{T}}$, the maxima in (13) exist. (ii) trivially follows from the definition of $\bar{\mathbb{T}}$. (iii) trivially follows from the definition of $\phi = \Phi(T)$ and Step 4.3 of Algorithm 2. ■

The number of flops $\varphi(\mathcal{L}, \mathcal{U})$ at a given iteration depends on the particular way the linear system (9) is solved in Algorithm 1. In our implementation, we use the LDL^T decomposition of $H_{\mathcal{I},\mathcal{I}}$. Contrary to active-set methods, in which the active-set changes incrementally, in the block-pivoting Algorithm 1 recursive updates of the LDL^T factorization (Bemporad, 2016; Nocedal & Wright, 1999) may be less efficient than batch factorizations, as the set $\mathcal{I}$ may completely change between two consecutive iterations. For this reason we opted for an implementation where the factorization is computed from scratch at each iteration, which also favors the numerical stability of the algorithm.

Algorithm 2 takes into account the specific value of the feasibility threshold ϵ, allowing one to certify convergence and quantify exactly the number of flops in the real case of using tolerances in the implementation. Usually ϵ is an arbitrary small number, but one can exploit the certification algorithm to also tune ϵ so to be as small as possible without incurring in cycling conditions.

As a final remark, besides certifying the convergence and computational complexity of the KR solver, Algorithm 2 also provides as a by-product the multiparametric solution of problem (2) (Bemporad, 2015; Bemporad et al., 2002). In fact, this can be immediately obtained by taking all the tuples $T \in \mathbb{T}$ and collect the corresponding polyhedra $\bar{\Theta}$ with the associated matrices A, a, B, b and active sets $\mathcal{L}^j, \mathcal{U}^j$.

4. Examples

We apply the complexity certification of the KR algorithm to different QP examples. We first show the results regarding three benchmark MPC problems, taken from the Model Predictive Control Toolbox for MATLAB® (Bemporad, Morari, & Ricker, 2014) demos library. Secondly, we deal with the problem of global convergence of the KR algorithm, showing the certification of a toy QP problem where the KR algorithm is known to fail.

4.1. Complexity certification for MPC problems

In MPC based on a linear time-invariant (LTI) prediction model, quadratic cost function, and saturation constraints on inputs, one needs to solve an optimal control problem of the following form

$$\min_u \sum_{i=1}^{N_p} \|W_y(y_{k+i|k} - r(k))\|_2^2 + \sum_{h=0}^{N_u-1} \|W_u(u_{k+h|k} - u_{r,k})\|_2^2 + \|W_{\Delta u}\Delta u_{k+h|k}\|_2^2 \tag{14a}$$

$$\text{s.t. } x_{k+i+1|k} = Ax_{k+i|k} + B_u u_{k+i|k} + B_d d(k) \tag{14b}$$

$$x_{k|k} = \hat{x}(k) \tag{14c}$$

$$y_{k+i+1|k} = Cx_{k+i+1|k} \tag{14d}$$

$$u_{\min}(k) \le u_{k+h|k} \le u_{\max}(k) \tag{14e}$$

$$i = 0, \ldots, N_p - 1,\ h = 0, \ldots, N_u - 1 \tag{14f}$$

Table 1
Dimensions of MPC problems.

	Double integrator (16)	Inverted pendulum (17)	Nonlinear demo (18)
n_x	3	5	5
n_u	1	1	3
n_y	1	2	2
N_p	10	50	5
N_u	3	5	2
n	3	5	6
n_θ	4	8	10
m	6	10	12

where $x \in \mathbb{R}^{n_x}$ is the state vector, $\hat{x}(k)$ is the estimate of x at time k, $u \in \mathbb{R}^{n_u}$ is the input vector, $y \in \mathbb{R}^{n_y}$ is the output vector, $d \in \mathbb{R}^{n_d}$ is the vector of measured disturbances, $r \in \mathbb{R}^{n_y}$ the output reference, $u_r \in \mathbb{R}^{n_u}$ the input reference, $A \in \mathbb{R}^{n_x \times n_x}$, $B_u \in \mathbb{R}^{n_x \times n_u}$, $B_d \in \mathbb{R}^{n_x \times n_d}$ and $C \in \mathbb{R}^{n_y \times n_x}$ are the state-space matrices, N_p is the prediction horizon, N_u is the control horizon, W_y, W_u, and $W_{\Delta u}$ are square weight matrices, $x_{k+i|k}$ denotes the prediction of the variable x at time $k+i$ based on the information available at time k, $\Delta u_{k+i|k} = u_{k+i|k} - u_{k+i-1|k}$ is the vector of the input increments, with $u_{k-1|k} = u(k-1)$, and $u_{\min}(k)$, $u_{\max}(k)$ are lower and upper bounds on the input signal imposed at time k. Problem (14) can be cast as a box-constrained QP as in (1), with $\theta = [\hat{x}(k)'\ r(k)'\ u_r(k)'\ d(k)'\ u(k-1)'\ u_{\min}(k)'\ u_{\max}(k)']'$, $z = [u'_{k|k} \dots u'_{k+N_u-1|k}]'$ and m the total number of imposed constraints.

In the following examples we assume that $u_{\min}(k)$, $u_{\max}(k)$, $u_r(k)$ are constant with respect to time k, and correspondingly treat $\theta(k) = [x(k)'\ u(k-1)'\ r(k)'\ d(k)']'$ as the parameter vector at time k, with

$$\Theta = \left\{ \theta \in \mathbb{R}^{n_\theta} \mid \begin{bmatrix} x^- \\ u^- \\ r^- \\ d^- \end{bmatrix} \le \theta \le \begin{bmatrix} x^+ \\ u^+ \\ r^+ \\ d^+ \end{bmatrix} \right\} \tag{15}$$

defining the range of interest for $\theta(k)$. Table 1 collects the dimensions of three MPC problems taken from the MPC Toolbox for MATLAB® (we consider in our analysis), that we describe in more detail in the following paragraphs. These examples well represent the dimensions of QPs typically found in real-time embedded control problems.

The first MPC problem is about regulating a single-input-single-output *double-integrator* under input saturation. The prediction model, weights, constraints and range of parameters for this problem are

$$\begin{aligned} &x_{k+1} = \begin{bmatrix} 1 & 0 \\ 0.1 & 1 \end{bmatrix} x_k + \begin{bmatrix} 0.1 \\ 0.005 \end{bmatrix} u_k,\ y_k = [\,0\ 1\,]\, x_k \\ &W_{\Delta u} = 0.1,\ W_y = 1,\ -1 \le u_k \le 1, \\ &u^- = -1,\ x^- = [\,-4\ -4\,]',\ r^- = -2, \\ &u^+ = 1,\ x^+ = [\,4\ 4\,]',\ r^+ = 2. \end{aligned} \tag{16}$$

The second problem regards the regulation of an *inverted pendulum* on a cart, which is a single-input-multi-output system with one measured disturbance and input constraints. The problem specifications are:

$$\begin{aligned} x_{k+1} = &\begin{bmatrix} 1 & 9.52\text{e}{-3} & 4.74\text{e}{-4} & 1.59\text{e}{-6} & 8.01\text{e}{-9} \\ 0 & 0.9048 & 0.0934 & 4.74\text{e}{-4} & 3.19\text{e}{-6} \\ 0 & -9.67\text{e}{-4} & 1.0019 & 0.0100 & 0.0001 \\ 0 & -0.1905 & 0.3832 & 1.0019 & 0.0200 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} x_k \\ &+ \begin{bmatrix} 2.42\text{e}{-6} \\ 4.76\text{e}{-4} \\ 4.84\text{e}{-6} \\ 9.52\text{e}{-4} \\ 0 \end{bmatrix} u_k + \begin{bmatrix} 8.02e-09 \\ 3.19e-06 \\ 0.0001 \\ 0.0200 \\ 0 \end{bmatrix} d_k \\ y_k = &\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} x_k,\ W_{\Delta u} = 1,\ W_y = \begin{bmatrix} 1.2 & 0 \\ 0 & 1 \end{bmatrix} \\ -10 \le u_k \le 10,\ &u^- = -10,\ x^- = [\,-20\ -20\ -20\ -20\ -20\,]' \\ r^- = -20,\ &u^+ = 10,\ x^+ = [\,20\ 20\ 20\ 20\ 20\,]',\ r^+ = 20 \end{aligned} \tag{17}$$

Table 2
Global convergence check of KR algorithm: sufficient conditions and certification algorithm.

	Double integrator	Inverted pendulum	Nonlinear demo
c_1	✗	✗	✗
c_2	✗	✗	✗
c_3	✗	✗	✗
Algorithm 2	✓	✓	✓

The last one is the *nonlinear demo*, where a linear MPC controller regulates a multi-input multi-output nonlinear system based on the following linear prediction model

$$\begin{aligned} x_{k+1} = &\begin{bmatrix} 0.8187 & 0 & 0 & 0 & 0 \\ 0.1474 & 0.6550 & -0.1637 & 0.0489 & 0.4878 \\ 0.01637 & 0.1637 & 0.9825 & 3.43\text{e}{-3} & 0.0523 \\ 0 & 0 & 0 & 0.8013 & -0.1801 \\ 0 & 0 & 0 & 0.1801 & 0.9813 \end{bmatrix} x_k \\ &+ \begin{bmatrix} 0.1813 & 0 & 0 \\ 0.0163 & 0.1637 & 3.43\text{e}{-3} \\ 1.14\text{e}{-3} & 0.0175 & 1.77\text{e}{-4} \\ 0 & 0 & 0.1801 \\ 0 & 0 & 0.0186 \end{bmatrix} u_k,\ y_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 \end{bmatrix} x_k \\ &W_{\Delta u} = \mathrm{diag}([\,0.1\ 0.1\ 0.1\,]),\ W_y = \mathrm{diag}([\,1\ 1\,]) \\ &\begin{bmatrix} -3 \\ -2 \\ -2 \end{bmatrix} \le u_k \le \begin{bmatrix} 3 \\ 2 \\ 2 \end{bmatrix},\ u^- = [\,-3\ -2\ -2\,]' \\ &x^- = [\,-0.5\ -0.5\ -0.5\ -0.5\ -0.5\,]',\ r^- = [\,0\ 0\,]' \\ &u^+ = [\,3\ 2\ 2\,]',\ x^+ = [\,2\ 1\ 1\ 1\ 1\,]',\ r^+ = [\,1.2\ 1.2\,]' \end{aligned} \tag{18}$$

where $\mathrm{diag}(v)$ is the diagonal matrix whose (i,i)th entry is v_i.

Table 2 shows that although all the three sufficient conditions C_1, C_2 and C_3 in Proposition 1 are violated (✗) for all MPC problems, Algorithm 2 and Theorem 1 still prove that the KR solver can be safely applied (✓), as the condition $\bar{\mathbb{T}} = \emptyset$ is verified in all three examples for the given sets Θ.

Fig. 1 clarifies how the certification algorithm works, showing the projection of the polyhedra associated with the optimal tuples $\{T^j \in \mathbb{T},\ j = 1, \dots, \#\mathbb{T}\}$ obtained by running Algorithm 2 . In Fig. 1 the same color is used for polyhedral regions where the KR algorithm returns the optimal solution z^* in the same number $N(T)$ of iterations.

The proposed algorithm also allows us to collect in Table 3 the worst-case computation complexity, in terms of the worst-case number $N_{\mathrm{KR}}^{\max}$ of iterations, memory occupancy m_{KR} in single and double precision arithmetic, the worst-case number $n_{\mathrm{KR}}^{\max}$ of flops, and the time t_{KR} required to certify the problem on a 2.5 GHz Intel® Core i7-4710MQ CPU. In the table we have added the certification results for the well known Goldfarb–Idnani (GI) dual active set solver (Goldfarb & Idnani, 1983), which solves efficiently QP problems with general constraints and iterates the primal solution by adding/removing one violated/blocking constraint at each iteration. The complexity certification algorithm for the GI solver has been proposed by the authors in Cimini and Bemporad (2017). Regarding the GI solver, we have collected the single and double precision memory occupancy m_{GI}, the worst-case number of flops

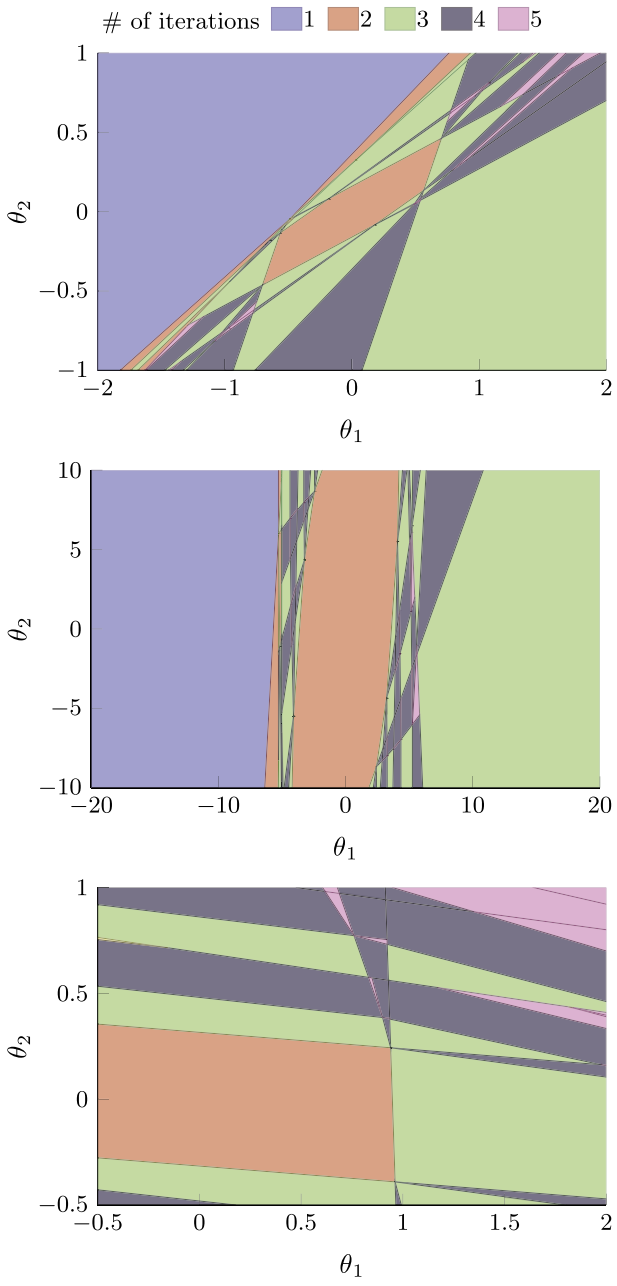

Fig. 1. Results of the certification algorithm: Partition of the parameter set based on the number of iterations required by the KR solver (same color means same number of QP iterations to get to the optimal solution). From top to bottom: double integrator, pendulum, nonlinear demo.

$n_{\text{GI}}^{\max}$ and the pair $(N^a, N^d)_{\text{GI}}^{\max}$, which is the maximum number of iterations corresponding to the worst-case execution time, with N^a the number of iterations where one constraint is added, and N^d the number of iterations where a constraint is removed. Even though the GI algorithm has been shown to be an ideal candidate for embedded optimization (Cimini & Bemporad, 2017), the KR solver has the advantage to be usually faster, thanks to the ability to add multiple constraints simultaneously, more memory efficient, and simpler to code, which translates into an easier formal verification of the code. Moreover, KR converges regardless of the initial guess of the active set, which makes the implementation of possible warm-starting strategies straightforward, contrary to GI. Table 3 shows also the features of the corresponding multiparametric solution for each problem, namely the number of regions n_r, the memory occupancy m_{exp} in single and double precision, and the worst-case number of flops $n_{\text{exp}}^{\max}$ to evaluate the explicit MPC solution.

For both explicit and implicit MPC, the memory occupancy considers both the data and the size of the *C*-code evaluating the control law, which is less than 1 kB for explicit MPC, 2.6 kB and 7.7 kB for the KR and GI algorithms in our implementation, respectively. In the case of explicit MPC, the point location algorithm to identify the region the current $\theta(k)$ belongs to can be implemented in different ways. Here the number of flops and the memory occupancy refer to the efficient implementation presented in Baotić, Borrelli, Bemporad, and Morari (2008, Algorithm 4), for which details on the computational load can be found in Cimini and Bemporad (2017, Section V.C).

As a result, the complexity certification algorithm eases the integration of an MPC controller into an embedded system, by verifying that the KR method convergences for all $\theta(k) \in \Theta$ and providing means to exactly assess which is the best solver for the particular application at hand. In fact, it accounts for the most favorable memory/flops trade-off given a specific problem, instead of determining it heuristically by massive simulations. For instance, the KR solver not only converges for all the problems considered here, but it is shown to be always faster and more memory efficient than GI algorithm. In the nonlinear demo for example, KR is 18% faster in the worst-case and requires about 60% less memory for a double precision implementation. Explicit MPC is the best approach in terms of speed only for the double integrator example, which is the smallest control problem, requiring 33% less flops in the worst-case with respect to KR, even though the memory requirement is more than twice the one of KR in double precision.

We then provide a benchmark in Table 4 regarding the performance of KR and GI algorithms on the inverted pendulum problem for different values of control horizon N_u, from 5 up to 12. This shows how the two algorithms scale in complexity with increasing problem dimension. For all the eight QP problems analyzed in this test, the KR solver is proven to always converge and the results show how it gets more efficient when the QP dimensions increase. In fact, by comparing the smaller and the larger problems, namely $N_u = 5$ and $N_u = 12$, KR requires respectively 24% and 55% less flops than GI in the worst-case. On the other hand, the memory improvement slightly decreases from 61% to 56% due to the larger impact the size of the code has on smaller problems.

Table 4 offers also a mean of comparing the computational complexity of the certification algorithm itself by showing the offline time t_{KR} needed to certify each problem. Algorithm 2 operates by recursively partitioning Θ into polyhedra, and therefore it is highly affected by the problem dimension. In the presented examples the total time needed to certify the QP problem ranges from 10.82 s to 892.35 s, therefore its applicability is limited to MPC problems of moderate size, typical of embedded control, which are the ones that most benefit from the results obtained by complexity certification.

4.2. Certification of non-convergence

In this section, the certification algorithm is applied to an optimization problem where the KR algorithm is known to cycle, that is the QP problem reported in Curtis et al. (2014, Example 1), whose linear term in the cost has been slightly modified in order to have a parametric QP problem as in the following

$$\begin{aligned} \min_z \quad & \frac{1}{2} z' \begin{bmatrix} 4 & 5 & -5 \\ 5 & 9 & -5 \\ -5 & -5 & 7 \end{bmatrix} z + z' \begin{bmatrix} 1 & 3 \\ 5 & 1 \\ 2 & 5 \end{bmatrix} \theta \\ \text{s.t.} \quad & \begin{bmatrix} -2 \\ -2 \\ -2 \end{bmatrix} \le z \le \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \end{aligned} \tag{19}$$

Table 3
Results from the complexity certification of explicit MPC and implicit MPC with KR and GI algorithms.

	Double integrator	Inverted pendulum	Nonlinear demo
Explicit MPC			
n_r	19	53	129
$n_{\text{exp}}^{\max}$ $(\pm, *)$	324	1830	5401
m_{exp} 16|32 bit [kB]	3.97|6.94	15.9|30.8	89.69|178.4
KR algorithm			
$N_{\text{KR}}^{\max}$	5	6	8
$n_{\text{KR}}^{\max}$ $(\pm, *, \div)$	489	1454	2961
m_{KR} 16|32 bit [kB]	3.19|3.33	3.39|3.70	3.51|3.94
$t_{\text{KR}}[s]$	2.81	10.82	126.62
GI algorithm			
$(N^a, N^d)_{\text{GI}}^{\max}$	(5,1)	(6,2)	(10,2)
$n_{\text{GI}}^{\max}$ $(\pm, *, \div)$|sqrt	580|5	1922|13	3622|24
m_{GI} 16|32 bit [kB]	8.21|8.56	8.63|9.38	8.90|9.90

Table 4
Benchmark of KR and GI algorithms for an increasing control horizon for the inverted pendulum problem.

N_u	KR algorithm			GI algorithm	
	$n_{\text{KR}}^{\max}$ $(\pm, *, \div)$	m_{KR} [kB] 16|32 bit	t_{KR} [s]	$n_{\text{GI}}^{\max}$ $(\pm, *, \div)$|sqrt	m_{GI} [kB] 16|32 bit
5	1454	3.39|3.70	10.82	1922|13	8.63|9.38
6	2290	3.51|3.93	31.35	2746|19	8.90|9.90
7	2875	3.65|4.20	79.16	4081|21	9.21|10.50
8	3902	3.81|4.50	231.76	5894|28	9.56|11.19
9	4616	3.98|4.82	292.54	8155|36	9.95|11.94
10	5421	4.17|5.19	391.65	10916|45	10.37|12.78
11	6296	4.37|5.57	509.91	14231|55	10.83|13.69
12	8039	4.59|6.00	892.35	18176|78	11.34|14.69

Fig. 2 shows the polyhedral regions associated with the tuples iterated by Algorithm 2. Again, the same color means the same number of iterations of the KR algorithm to find the solution. Black regions are those defining the tuples $T^i \in \bar{\mathbb{T}}$, $i = 1, \ldots, \#\bar{\mathbb{T}}$. This shows how the certification algorithm is able to detect exactly the regions where the KR algorithm is cycling, due to $\bar{\mathbb{T}} \neq \emptyset$. This result helps the control designer to eventually exclude KR from the list of candidate algorithms to solve the optimization problem, or alternatively can open the avenue to use KR algorithm even if it cycles.

Indeed, by knowing exactly the regions where this happens, one could apply a semi-explicit approach similar to the one proposed in Cimini and Bemporad (2017), by storing the explicit solution only for the regions of cycling. Given the certification result for the non-cycling polyhedra, and the straightforward computation of the worst-case flops for the evaluation of the partial explicit law, one can exactly certify if the semi-explicit approach with KR as an implicit solver is superior to other algorithms.

5. Conclusion

We have presented a certification algorithm that exactly computes the worst-case number of iterations and flops, or failure to converge, of the KR primal–dual infeasible block principal pivoting method for solving a family of box-constrained QP's, that depend on a vector of parameters in the linear term of the cost function and/or in the upper and lower bounds on the optimization vector. Compared to active-set methods, the considered KR block principal pivoting algorithm can be much faster and easier to code, although it lacks guarantees of convergence for all strictly convex box-constrained QPs. The results of this paper open the opportunity to adopt the KR solver in embedded MPC applications when the certification analysis provides a positive result, or to prohibit its use by providing counter-examples where it would fail. The complexity certification and convergence verification have been successfully demonstrated on well-known MPC problems, in which the sufficient conditions for convergence available from the literature are not satisfied.

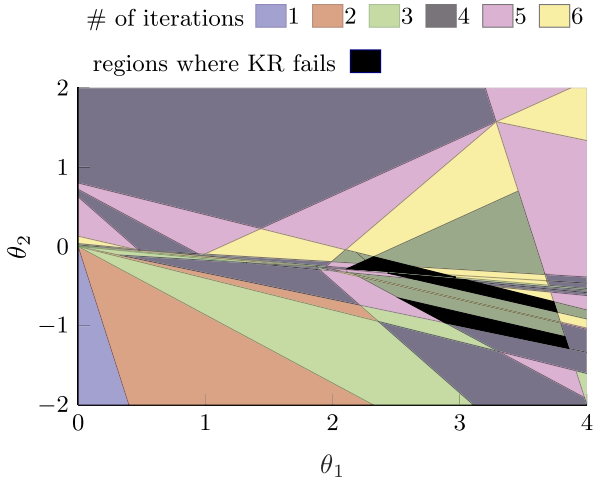

Fig. 2. Complexity certification of the KR algorithm for the toy QP problem (19). The black regions represent subsets of the parameter space for which the KR algorithm cycles.

References

Baotić, M., Borrelli, F., Bemporad, A., & Morari, M. (2008). Efficient on-line computation of constrained optimal control. *SIAM Journal on Control and Optimization*, *47*(5), 2470–2489.

Bartlett, R., & Biegler, L. (2006). QPSchur: A dual, active-set, Schur-complement method for large-scale and structured convex quadratic programming. *Optimization and Engineering*, *7*(1), 5–32.

Bemporad, A. (2015). A multiparametric quadratic programming algorithm with polyhedral computations based on nonnegative least squares. *IEEE Transactions on Automatic Control*, *60*(11), 2892–2903.

Bemporad, A. (2016). A quadratic programming algorithm based on nonnegative least squares with applications to embedded model predictive control. *IEEE Transactions on Automatic Control*, *61*(4), 1111–1116.

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

Bemporad, A., Morari, M., & Ricker, N. (2014). *Model predictive control toolbox for MATLAB 5.0*. The Mathworks, Inc., http://www.mathworks.com/access/helpdesk/help/toolbox/mpc/.

Bing, L., Shiji, S., Wei, L., & Keyou, Y. (2015). Dual active set method for support vector machines under multi-constraint activation. *Neurocomputing*, *154*, 296–304.

Buchheim, C., & Trieu, L. (2014). Active set methods with reoptimization for convex quadratic integer programming. In P. Fouilhoux, N. Gouveia, R. Mahjoub, & T. Paschos (Eds.), *Combinatorial optimization: Third international symposium, ISCO 2014, Lisbon, Portugal, March 5-7, 2014, Revised Selected Papers* (pp. 125–136). Cham: Springer International Publishing.

Cimini, G., & Bemporad, A. (2017). Exact complexity certification of active-set methods for quadratic programming. *IEEE Transactions on Automatic Control*, *62*(12), 6094–6109.

Curtis, F., Han, Z., & Robinson, D. (2014). A globally convergent primal-dual active-set framework for large-scale convex quadratic optimization. *Computational Optimization and Applications*, *60*(2), 311–341.

Ferreau, H. J., Bock, H. G., & Diehl, M. (2008). An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, *18*(8), 816–830.

Gharbia, I. B., & Gilbert, J. (2012). Nonconvergence of the plain Newton-min algorithm for linear complementarity problems with a *P*-matrix. *Mathematical Programming*, *134*(2), 349–364.

Gill, P., Gould, N., Murray, W., Saunders, M., & Wright, M. (1984). A weighted Gram-Schmidt method for convex quadratic programming. *Mathematical Programming*, *30*(2), 176–195.

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

Hintermller, M., Ito, K., & Kunisch, K. (2002). The primal-dual active set strategy as a semi-smooth Newton method. *SIAM Journal on Optimization*, *13*(3), 865–888.

Hungerlnder, P., & Rendl, F. (2015). A feasible active set method for strictly convex quadratic problems with simple bounds. *SIAM Journal on Optimization*, *25*(3), 1633–1659.

Hungerlnder, P., & Rendl, F. (2016). An infeasible active set method with combinatorial line search for convex quadratic problems with bound constraints. Optimization Group, Technical report, Alpen-Adria Universität Klagenfurt, Mathematics:TRAAUKMO160803.

Kunisch, K., & Rendl, F. (2003). An infeasible active set method for quadratic problems with simple bounds. *SIAM Journal on Optimization*, *14*(1), 35–52.

Li, W., & Swetits, J. (1997). A new algorithm for solving strictly convex quadratic programs. *SIAM Journal on Optimization*, *7*(3), 595–619.

Mayne, D. (2014). Model predictive control: Recent developments and future promise. *Automatica*, *50*(12), 2967–2986.

Necoara, I. (2015). Computational complexity certification for dual gradient method: Application to embedded MPC. *Systems & Control Letters*, *81*, 49–56.

Nesterov, Y., & Nemirovskii, A. (1994). *Interior-point polynomial algorithms in convex programming*. Society for Industrial and Applied Mathematics.

Nocedal, J., & Wright, S. (1999). *Numerical optimization*. Springer.

Patrinos, P., & Bemporad, A. (2014). An accelerated dual gradient-projection algorithm for embedded linear model predictive control. *IEEE Transactions on Automatic Control*, *59*(1), 18–33.

Patrinos, P., Sopasakis, P., & Sarimveis, H. (2011). A global piecewise smooth Newton method for fast large-scale model predictive control. *Automatica*, *47*(9), 2016–2022.

Richter, S., Jones, C., & Morari, M. (2012). Computational complexity certification for real-time MPC with input constraints based on the fast gradient method. *IEEE Transactions on Automatic Control*, *57*(6).

Saraf, N., & Bemporad, A. (2017). Fast model predictive control based on linear input/output models and bounded-variable least squares. In *Proc. 56th IEEE conf. on decision and control*, Melbourne, Australia.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., & Boyd, S. (2017). OSQP: An operator splitting solver for quadratic programs. http://arxiv.org/abs/1711.08013, Code avaliable at https://github.com/oxfordcontrol/osqp.

Vanderbei, R. (1999). LOQO: an interior point code for quadratic programming. *Optimization Methods & Software*, *11*(1–4), 451–484.

Gionata Cimini received his M.Sc. degree, cum laude, in Computer and Automation Engineering in 2012 and the Ph.D. degree, cum laude, in Information Engineering, in 2017, from Università Politecnica delle Marche, Ancona Italy. In 2014/2015 he was a guest Ph.D. scholar at IMT Alti Studi di Lucca Lucca Italy and in 2016 he was a visiting Ph.D. scholar at University of Michigan, Ann Arbor, US. In 2013 he held a research assistant position at the Information Department, Università Politecnica delle Marche and in 2016 he was a research assistant at the Automotive Research Center, University of Michigan. From 2016 he is an advanced control specialist at ODYS Srl, Milan, Italy. He has published 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 received his Master's degree in Electrical Engineering in 1993 and his Ph.D. in Control Engineering in 1997 from the University of Florence, Italy. In 1996/97 he was with the Center for Robotics and Automation, Department of Systems Science & Mathematics, Washington University, St. Louis. In 1997–1999 he held a postdoctoral position at the Automatic Control Laboratory, ETH Zurich, Switzerland, where he collaborated as a senior researcher until 2002. In 1999–2009 he was with the Department of Information Engineering of the University of Siena, Italy, becoming an Associate Professor in 2005. In 2010–2011 he was with the Department of Mechanical and Structural Engineering of the University of Trento, Italy. Since 2011 he is Full Professor at the IMT School for Advanced Studies Lucca, Italy, where he served as the Director of the institute in 2012–2015. He spent visiting periods at Stanford University, University of Michigan, and Zhejiang University. In 2011 he cofounded ODYS S.r.l., a company specialized in developing model predictive control systems for industrial production. He has published more than 300 papers in the areas of model predictive control, hybrid systems, optimization, automotive control, and is the co-inventor of 10 patents. He is author or coauthor of various MATLAB toolboxes for model predictive control design, including the Model Predictive Control Toolbox (The Mathworks, Inc.) and the Hybrid Toolbox. He was an Associate Editor of the IEEE Transactions on Automatic Control during 2001–2004 and 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 2011–14 triennial. He has been an IEEE Fellow since 2010.