Efficiently solving the harmonic model predictive control formulation

Pablo Krupa†, Daniel Limon†, Alberto Bemporad*, Teodoro Alamo†

***Abstract*—Harmonic model predictive control (HMPC) is a model predictive control (MPC) formulation which displays several benefits over other MPC formulations, especially when using a small prediction horizon. These benefits, however, come at the expense of an optimization problem that is no longer the typical quadratic programming problem derived from most linear MPC formulations due to the inclusion of a particular class of second order cone constraints. This article presents a method for efficiently dealing with these constraints in operator splitting methods, leading to a computation time for solving HMPC in line with state of the art solvers for linear MPC. We show how to apply this result to the alternating direction method of multipliers algorithm, presenting a solver which we compare against other solvers from the literature, including solvers for other linear MPC formulations. The results show that the proposed solver, and by extension the HMPC formulation, is suitable for its implementation in embedded systems.**

***Index Terms*—predictive control, harmonic model predictive control, convex optimization, embedded systems, ADMM.**

I. Introduction

In the recent publication [1] (originally presented in [2]), the authors proposed a novel model predictive control (MPC) [3] formulation labeled *harmonic model predictive control* (HMPC), which has several advantages over other MPC formulations, such as guaranteed asymptotic stability, recursive feasibility even in the event of a sudden reference change, an increased domain of attraction with respect to other MPC formulations, does not require a positive invariant set of the system, and displays an improved performance of the closed-loop system when using a small prediction horizon, especially for systems with integrator states or slew-rate constraints.

These advantages, which are highlighted and discussed in detail in [1], [2], indicate that HMPC is an ideal candidate for its use as an embedded controller, i.e., for its implementation in devices with low computation and memory resources. The main drawback, however, is that its optimization problem is not the typical quadratic programming (QP) problem derived from most linear MPC formulations. This is due to the inclusion of several constraints that can be imposed as second-order cone (SOC) constraints.

† Department of Systems Engineering and Automation, Universidad de Sevilla, 41092, Sevilla, Spain. E-mails: pkrupa@us.es, dlm@us.es, talamo@us.es. Corresponding author: Pablo Krupa.

* IMT School for Advanced Studies, Piazza San Francesco 19, Lucca, Italy. Email: alberto.bemporad@imtlucca.it

This work was supported in part by Grant PDC2021-121120-C21 funded by MCIN/AEI/10.13039/501100011033 and by the "European Union NextGenerationEU/PRTR", and in part by Grant Margarita Salas (grant number 20122) funded by the Ministerio de Universidades and the European Union (NextGenerationEU).

This article shows that in spite of this drawback, the HMPC formulation can be solved in a computation time comparable to that of MPC formulations whose control law is derived from QP problems using state of the art solvers.

We show how to efficiently deal with the SOC-like constraints of the HMPC formulation by grouping them in pairs and considering their intersection. We prove an explicit solution to the Euclidean projection onto this intersection of pairs of SOC-like constraints; a fact that can be exploited by several first-order methods. In particular, the operator splitting methods considered in solvers such as [4], [5] or [6] can make good use of this, since the pairing (along with the explicit solution of the projection operator) leads to a reduction of the number of decision variables and dimensions of the matrices involved in the solver, leading to a reduction of the computation time when compared to simply solving the original problem by considering SOC constraints.

To show how to solve the resulting optimization problem, we present a solver based on the alternating direction method of multipliers (ADMM) algorithm [7] using ideas and approaches taken from state of the art solvers [4], [5], [8]. The resulting solver, which is available in the SPCIES toolbox [9], is well suited for its implementation in embedded systems, especially considering that the HMPC formulation is particularly suited for its use with small prediction horizons.

The remainder of this article is structured as follows. In Section II we describe the SOC-like sets that we deal with in subsequent sections and present explicit solutions for the Euclidean projection onto them. We briefly recall the HMPC formulation in Section III. Section IV presents the ADMM algorithm and its particularization to HMPC. Numerical results are presented in Section V. Concluding remarks and a discussion of the computational results are provided in Section VI.

Notation: Given two integers i and j with $j \geq i$, $\mathbb{Z}_i^j$ denotes the set of integer numbers from i to j, i.e. $\mathbb{Z}_i^j \doteq \{i, i+1, \dots, j-1, j\}$. We denote by $\mathbb{S}_{++}^n$ ($\mathbb{D}_{++}^n$) the set of (diagonal) positive definite matrices in $\mathbb{R}^{n\times n}$. Given a set $\mathcal{X} \subseteq \mathbb{R}^n$, we denote by $\mathcal{I}_\mathcal{X}$ its indicator function, i.e., $\mathcal{I}_\mathcal{X}(x) = 0$ if $x \in \mathcal{X}$ and $\mathcal{I}_\mathcal{X}(x) = +\infty$ if $x \notin \mathcal{X}$. For vectors x_1 to x_N, $(x_1, x_2, \dots, x_N)$ denotes the column vector formed by their concatenation. Given a vector $x \in \mathbb{R}^n$, we denote its i-th component using a parenthesized subindex $x_{(i)}$. Given two vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$, their standard inner product is denoted by $\langle x, y\rangle \doteq \sum_{i=1}^n x_{(i)}y_{(i)}$. For $x \in \mathbb{R}^n$ and $A \in \mathbb{S}_{++}^n$, $\|x\| \doteq \sqrt{\langle x, x\rangle}$, $\|x\|_A \doteq \sqrt{\langle x, Ax\rangle}$, $\|x\|_\infty \doteq \max_{i=1\dots n} |x_{(i)}|$. The Euclidean projection of a vector $x \in \mathbb{R}^n$ onto a set $\mathcal{X} \subseteq \mathbb{R}^n$ is denoted by $\mathcal{P}_\mathcal{X}(x)$, i.e., $\mathcal{P}_\mathcal{X}(x) = \arg\min_{v\in\mathcal{X}} \|v - x\|^2$.

II. Shifted second order cones

This section describes a class of closed convex sets, which we denote by *shifted second order cones* (shifted-SOCs). We prove an explicit solution for the Euclidean projection onto them and onto the intersection of two "opposed" shifted-SOCs. The results and definitions of this section will play a major role in subsequent developments.

Definition 1 (Shifted second order cone)**.** *A shifted second order cone (shifted-SOC) $\mathcal{K}_\alpha(c) \subset \mathbb{R}^n$ is a set given by*

$$\mathcal{K}_\alpha(c) = \{z = (z_0, z_1) \in \mathbb{R}\times\mathbb{R}^{n-1} : \|z_1\| \leq \alpha(z_0 - c)\}, \quad (1)$$

where $\alpha \in \{1, -1\}$ and $c \in \mathbb{R}$. For convenience, let us denote by $\mathcal{K}_-(c) \doteq \mathcal{K}_{-1}(c)$ and $\mathcal{K}_+(c) \doteq \mathcal{K}_1(c)$, where we may drop the "(c)" if it is clear from the context.

The following theorem provides an explicit solution for the Euclidean projection onto $\mathcal{K}_\alpha(c)$. Its proof is heavily inspired by the proof of [10, Theorem 3.3.6], which proves an explicit solution onto (1) for $c = 0$ and $\alpha > 0$.

Theorem 1. *Let $z = (z_0, z_1) \in \mathbb{R} \times \mathbb{R}^{n-1}$ and $\mathcal{K}_\alpha(c) \subset \mathbb{R}^n$ be given by Definition 1 for some $\alpha \in \{1, -1\}$ and $c \in \mathbb{R}$. The Euclidean projection of z onto $\mathcal{K} \doteq \mathcal{K}_\alpha(c)$ is given by*

$$\mathcal{P}_\mathcal{K}(z) = \begin{cases} z & \text{if } \|z_1\| \leq \alpha(z_0 - c) \quad (2a) \\ (c, 0) & \text{if } \|z_1\| \leq -\alpha(z_0 - c) \quad (2b) \\ \left(\tau\alpha + c, \dfrac{\tau z_1}{\|z_1\|}\right) & \text{otherwise,} \quad (2c) \end{cases}$$

where $\tau = \dfrac{1}{2}(\alpha(z_0 - c) + \|z_1\|)$.

Proof. See Appendix A.

We are now interested in the following set, obtained from the intersection of two "opposed" shifted-SOCs. Let us denote by $\mathcal{D}(\overline{z}, \underline{z}) \subset \mathbb{R}^n$, where $\overline{z}, \underline{z} \in \mathbb{R}$, the set given by

$$\mathcal{D}(\overline{z}, \underline{z}) \doteq \{z \in \mathbb{R}^n : z \in \mathcal{K}_-(\overline{z}) \cap \mathcal{K}_+(\underline{z})\}. \quad (3)$$

Once again, we may drop the "$(\overline{z}, \underline{z})$" if it is clear from the context. Set $\mathcal{D}$ is closed and convex, since it is the intersection of two closed convex sets [11, Prop. 1.1.1(a)], [11, Prop. A.2.4(b)]. Additionally, it is non-empty if $\underline{z} \leq \overline{z}$, as we state in the following lemma.

Lemma 1. *Let $\mathcal{D}(\overline{z}, \underline{z}) \subset \mathbb{R}^n$ be given by* (3) *for some $\overline{z}, \underline{z} \in \mathbb{R}$. Then, $\mathcal{D}$ is non-empty iff $\underline{z} \leq \overline{z}$.*

Proof. First, assume that $\mathcal{D}$ is non-empty and take any $z \in \mathcal{D}$. Then, from $z \in \mathcal{K}_-(\overline{z})$ and $z \in \mathcal{K}_+(\underline{z})$ we have that $\|z_1\| \leq \overline{z} - z_0$ and $\|z_1\| \leq z_0 - \underline{z}$, which leads to $\overline{z} - \underline{z} \geq 2\|z_1\| \geq 0$. Next, assume that $\underline{z} \leq \overline{z}$ and consider the vector $z = ((\overline{z}+\underline{z})/2, 0) \in \mathbb{R}\times\mathbb{R}^{n-1}$. It is easy to verify that $z \in \mathcal{K}_-$ and $z \in \mathcal{K}_+$, thus $z \in \mathcal{D}$. ∎

The projection onto set $\mathcal{D}$ could be performed using one of many methods from the literature for projecting onto the intersections of convex sets [12], [13]. These methods typically consider the Euclidean projection of a vector onto a non-empty closed convex set $\mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2 \cap \cdots \cap \mathcal{C}_r$, where $r > 0$ is finite, and it is assumed that the sets $\mathcal{C}_i$ are closed and convex and that $\mathcal{P}_{\mathcal{C}_i}$ has a known solution for $i \in \mathbb{Z}_1^r$. They are employed because, in general, the projection onto the intersection of convex sets is not guaranteed to be the result of projecting onto each set $\mathcal{C}_i$ in order, even if $r = 2$.

Algorithm 1: Dykstra's algorithm for $\mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2$.

Input: $z \in \mathbb{R}^n$
$w^0 \leftarrow z$, $p^0 \leftarrow \mathbf{0}_n$, $q^0 \leftarrow \mathbf{0}_n$
foreach $k \geq 1$ **do**
  $v^k \leftarrow \mathcal{P}_{\mathcal{C}_1}(w^{k-1} + p^{k-1})$
  $p^k \leftarrow w^{k-1} + p^{k-1} - v^k$
  $w^k \leftarrow \mathcal{P}_{\mathcal{C}_2}(v^k + q^{k-1})$
  $q^k \leftarrow v^k + q^{k-1} - w^k$

However, we will show that this is not the case for the projection onto set $\mathcal{D}$, which can be obtained by first projecting onto $\mathcal{K}_+$ and then projecting the resulting vector onto $\mathcal{K}_-$, both of which have simple explicit solutions given by Theorem 1. This result will allow us to directly use sets $\mathcal{D}$ in future developments without having to result to an iterative method to compute the projection onto them, which would be computationally expensive, especially if a good approximation of the projection is required.

The following theorem states a direct solution of the projection onto a non-empty set $\mathcal{D}$. Its proof is based on making use of the following lemma, which states the condition for the convergence after a single iteration of Algorithm 1, obtained from [14, §3], which is a particularization of Dykstra's algorithm [12] to finding the projection of $z \in \mathbb{R}^n$ onto $\mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2$. It generates iterates v^k and w^k satisfying $\|v^k - \mathcal{P}_\mathcal{C}(z)\| \to 0$ and $\|w^k - \mathcal{P}_\mathcal{C}(z)\| \to 0$ as $k \to +\infty$ [12, Theorem 2].

Lemma 2. *Let $\mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2$, be a non-empty closed convex set and $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathbb{R}^n$ be closed convex sets. Consider Algorithm 1 for finding $\mathcal{P}_\mathcal{C}(z)$ for $z \in \mathbb{R}^n$. Then, $v^2 = w^1 \Rightarrow w^1 = \mathcal{P}_\mathcal{C}(z)$.*

Proof. The first iterate of Algorithm 1 satisfies

$$v^1 = \mathcal{P}_{\mathcal{C}_1}(z), \quad p^1 = z - v^1, \quad w^1 = \mathcal{P}_{\mathcal{C}_2}(v^1), \quad q^1 = v^1 - w^1.$$

Then, if $v^2 = w^1$, we have that

$$\begin{aligned} &v^2 = \mathcal{P}_{\mathcal{C}_1}(w^1 + p^1) = w^1, \;\; p^2 = w^1 + p^1 - w^1 = p^1, \\ &w^2 = \mathcal{P}_{\mathcal{C}_2}(v^2 + v^1 - w^1) = \mathcal{P}_{\mathcal{C}_2}(v^1) = w^1, \\ &q^2 = v^2 + q^1 - w^2 = q^1. \end{aligned}$$

Therefore, iterations $k > 2$ will return the same results as $k = 2$, meaning we have reached a fixed point of the algorithm, and thus $w^2 = w^1 = \mathcal{P}_\mathcal{C}(z)$ [12, Theorem 2]. ∎

Theorem 2. *Let $z = (z_0, z_1) \in \mathbb{R} \times \mathbb{R}^{n-1}$ and $\mathcal{D}(\overline{z}, \underline{z})$ be the set given by* (3) *for some $\overline{z}, \underline{z} \in \mathbb{R}$ satisfying $\underline{z} \leq \overline{z}$. Then, the projection of z onto $\mathcal{D}$ is given by $\mathcal{P}_\mathcal{D}(z) = \mathcal{P}_{\mathcal{K}_-}\left(\mathcal{P}_{\mathcal{K}_+}(z)\right)$.*

Proof. See Appendix B.

III. Harmonic model predictive control

The HMPC formulation [1], [2], considers a controllable linear time-invariant system described by the discrete state space model

$$x(t+1) = Ax(t) + Bu(t), \quad (4)$$

where $x(t) \in \mathbb{R}^{n_x}$ and $u(t) \in \mathbb{R}^{n_u}$ are the state and control input at the discrete time instant t, respectively, subject to

$$\underline{y} \leq Ex(t) + Fu(t) \leq \overline{y}, \tag{5}$$

where we assume that the bounds $\underline{y}, \overline{y} \in \mathbb{R}^{n_y}$ satisfy $\underline{y} < \overline{y}$.

The HMPC formulation is inspired by the *MPC for tracking* (MPCT) formulation [15], [16], whose difference with classical MPC formulations is that it includes an *artificial reference*, which is forced to be a steady-state of (4) satisfying (5), as decision variables in the optimization problem

The idea behind HMPC is to substitute this steady-state artificial reference by one in the form of the periodic signals x_h^j, u_h^j, called the *artificial harmonic reference*, whose value at each discrete time instant $j \in \mathbb{Z}$ is given by

$$x_h^j = x_e + x_s \sin(w(j-N)) + x_c \cos(w(j-N)), \tag{6a}$$
$$u_h^j = u_e + u_s \sin(w(j-N)) + u_c \cos(w(j-N)), \tag{6b}$$

i.e., to use a harmonic signal, thus the name of the formulation, with base frequency $w \geq 0$ parameterized by x_e, x_s, $x_c \in \mathbb{R}^{n_x}$ and u_e, u_s, $u_c \in \mathbb{R}^{n_u}$. Let us introduce the following notation to simplify future developments:

$$\mathbf{x}_H \doteq (x_e, x_s, x_c) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_x},$$
$$\mathbf{u}_H \doteq (u_e, u_s, u_c) \in \mathbb{R}^{n_u} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_u},$$
$$y_e = Ex_e + Fu_e,\ y_s = Ex_s + Fu_s,\ y_c = Ex_c + Fu_c.$$

For a given prediction horizon $N > 0$ and base frequency $w \geq 0$, the HMPC control law for a given state $x(t) \in \mathbb{R}^{n_x}$ and reference $(x_r, u_r) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$ is derived from

$$\min_{\substack{\mathbf{x},\mathbf{u},\\ \mathbf{x}_H,\mathbf{u}_H}} \sum_{j=0}^{N-1} \ell_h(x^j, u^j, x_h^j, u_h^j) + V_h(\mathbf{x}_H, \mathbf{u}_H; x_r, u_r) \tag{7a}$$
$$\text{s.t. } x^0 = x(t) \tag{7b}$$
$$x^{j+1} = Ax^j + Bu^j,\ j \in \mathbb{Z}_0^{N-1} \tag{7c}$$
$$\underline{y} \leq Ex^j + Fu^j \leq \overline{y},\ j \in \mathbb{Z}_0^{N-1} \tag{7d}$$
$$Ax^{N-1} + Bu^{N-1} = x_e + x_c \tag{7e}$$
$$x_e = Ax_e + Bu_e \tag{7f}$$
$$x_s \cos(w) - x_c \sin(w) = Ax_s + Bu_s \tag{7g}$$
$$x_s \sin(w) + x_c \cos(w) = Ax_c + Bu_c \tag{7h}$$
$$(y_{e(i)}, y_{s(i)}, y_{c(i)}) \in \mathcal{D}(\overline{y}_{(i)}, \underline{y}_{(i)}),\ i \in \mathbb{Z}_1^{n_y}, \tag{7i}$$

where $\mathbf{x} = (x^0, \ldots, x^{N-1})$, $\mathbf{u} = (u^0, \ldots, u^{N-1})$, and the two terms of the cost function are given by the stage cost function

$$\ell_h(x, u, x_h, u_h) = \|x - x_h\|_Q^2 + \|u - u_h\|_R^2,$$

where $Q \in \mathbb{S}_{++}^{n_x}$ and $R \in \mathbb{S}_{++}^{n_u}$; and the offset cost function

$$V_h(\cdot) = \|x_e - x_r\|_{T_e}^2 + \|u_e - u_r\|_{S_e}^2 + \|x_s\|_{T_h}^2 + \|x_c\|_{T_h}^2 + \|u_s\|_{S_h}^2 + \|u_c\|_{S_h}^2,$$

where $T_e \in \mathbb{S}_{++}^{n_x}$, $T_h \in \mathbb{D}_{++}^{n_x}$, $S_e \in \mathbb{S}_{++}^{n_u}$, and $S_h \in \mathbb{D}_{++}^{n_u}$. Note that constraint (7i) is imposing n_y constraints onto sets $\mathcal{D} \subset \mathbb{R}^3$ defined in (3).

As is typical in MPC, the summation of the stage cost function ℓ_h is penalizing the discrepancy between the predicted states x^j and inputs u^j with the reference, although in this case the discrepancy is with respect to the artificial reference at prediction time j, i.e., x_h^j and u_h^j, respectively. The offset cost function is penalizing two conceptually different things:

- The distances $\|x_e - x_r\|_{T_e}^2$ and $\|u_e - u_r\|_{S_e}^2$. This penalization will make the "center" of the artificial harmonic reference signal move towards the reference, eventually reaching it if is an admissible steady-state of the system.
- The magnitude of x_s, x_c, u_s and u_c. This will force the artificial harmonic reference signal to converge to the steady-state (x_e, u_e) as $k \to +\infty$, since it will converge towards $x_s = x_c = 0$ and $u_s = u_c = 0$ as $k \to \infty$.

The result of this, as stated and proven in [1, Theorem 3], is that the closed-loop system will asymptotically converge to (x_r, u_r) if it is an admissible steady-state of the system, or to the admissible steady-state (x_t, u_t) that minimizes the distance $\|x_t - x_r\|_{T_e}^2 + \|u_t - u_r\|_{S_e}^2$ otherwise. In both cases, the closed-loop system will satisfy the constraints (5) under nominal conditions.

Constraints (7b)-(7d) impose the typical MPC constraints: current system state, system dynamics and system constraints, respectively. Constraint (7e) forces the terminal state x^N to reach the artificial harmonic reference, i.e., $x^N = x_h^N$, and (7f) that (x_e, u_e) must be a steady-state of the system. The satisfaction of (7g)-(7h) along with (7f) results in an artificial harmonic reference (6) that satisfies the system dynamics (4), i.e., it satisfies $x_h^{j+1} = Ax_h^j + Bu_h^j$, $\forall j$ (see [1, Property 2]). Finally, the constraints (7i) enforce that the artificial harmonic reference satisfies the system constraints (5) (see [1, Property 3]). The reason for imposing the system constraints on the artificial harmonic reference this way is that the satisfaction of (7i) implies the feasibility of x_h^j and u_h^j for all j. Therefore, the system constraints (5) can be imposed on the artificial reference by only n_y constraints. A more naive approach would have resulted in a number of constraints that depends on the values of N and w. The downside, however, is that we add constraints on the intersection of shifted-SOC constraints, leading to an optimization problem that is no longer the typical QP obtained from most linear MPC formulations.

For a more detailed description and discussion of this formulation and its advantages we refer the reader to [1], [2].

Remark 1. *The HMPC formulation* (7) *can be posed as a SOC programming problem by first separating each constraint* (7i) *into two constraints, one on $\mathcal{K}_+$ and one on $\mathcal{K}_-$. In this case, problem* (7) *can be solved using SOC programming solvers such as [6]. However, by doing so, we have $2n_y$ SOC constraints instead of the n_y constraints* (7i)*. Our use of the sets $\mathcal{D}$ leads to a smaller number of constraints, which can have a major impact on the computation time of the solver when n_y is large or when using certain operator splitting approaches such as the one used in [4], [5], [6].*

IV. ADMM SOLVER FOR HMPC

This section shows how the HMPC formulation can be solved taking into account the sets and projection theorems from Section II by presenting a solver based on the ADMM algorithm [7].

A. Alternating direction method of multipliers

Let us consider the optimization problem

$$\min_{z,s} \frac{1}{2} z^\top H z + q^\top z \tag{8a}$$
$$\text{s.t. } Gz = b,\ s \in \mathcal{S} \tag{8b}$$
$$Cz + s = d, \tag{8c}$$

where $z \in \mathbb{R}^n$ are the *primal decision variables*, $s \in \mathbb{R}^m$ are the *primal slack variables*, $H \in \mathbb{S}^n_{++}$, $q \in \mathbb{R}^n$, $G \in \mathbb{R}^{n_{eq} \times n}$, $b \in \mathbb{R}^{n_{eq}}$, $C \in \mathbb{R}^{m \times n}$, $d \in \mathbb{R}^m$ and $\mathcal{S}$ is a Cartesian product of the form $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_p$, where $\mathcal{S}_i \subseteq \mathbb{R}^{m_i}$, $i \in \mathbb{Z}_1^p$, with $\sum_{i=1}^p m_i = m$, are non-empty closed convex sets, i.e., we consider a partition of s given by $s = (s_1, s_2, \ldots, s_p)$, where each $s_i \in \mathbb{R}^{m_i}$, $i \in \mathbb{Z}_1^p$, must belong to $\mathcal{S}_i$.

Problem (8) can be solved by considering (8c) as the linear constraint relating the two separable decision variables z and s, and including (8b) as indicator functions in the objective function. That is, to consider the classical ADMM optimization problem

$$\min_{z,s} f(z) + g(s)$$
$$\text{s.t. } Cz + s = d,$$

where $f(z) = \frac{1}{2} z^\top H z + q^\top z + \mathcal{I}_{(G,b)}(z)$, $g(s) = \mathcal{I}_{\mathcal{S}}(s)$, and $\mathcal{I}_{(G,b)}$ is the indicator functions of set $\{z \in \mathbb{R}^n : Gz = b\}$. This leads to the Lagrangian function

$$\mathcal{L}(z, s, \lambda) = f(z) + g(s) + \frac{\rho}{2} \|Cz + s - d + \frac{1}{\rho}\lambda\|^2,$$

where $\lambda \in \mathbb{R}^m$ are the dual variables for constraint (8c) and $\rho > 0$ is the penalty parameter.

Starting at an initial point (z^0, s^0, λ^0), the iterations of ADMM consist of the following steps [7]:

$$z^{k+1} = \arg\min_z \mathcal{L}(z, s^k, \lambda^k) \tag{9a}$$
$$s^{k+1} = \arg\min_s \mathcal{L}(z^{k+1}, s, \lambda^k) \tag{9b}$$
$$\lambda^{k+1} = \lambda^k + \rho(Cz^{k+1} + s^{k+1} - d), \tag{9c}$$

where $k \geq 0$ is the iteration counter. Step (9a) solves

$$z^{k+1} = \arg\min_z \left\{ \frac{1}{2} z^\top \hat{H} z + (\hat{q}^k)^\top z,\ \text{ s.t. } Gz = b \right\}, \tag{10}$$

where $\hat{H} = H + \rho C^\top C$ and $\hat{q}^k = q + C^\top \left(\rho(s^k - d) + \lambda^k\right)$. Step (9b) solves

$$s^{k+1} = \arg\min_{s \in \mathcal{S}} \frac{\rho}{2} \|Cz^{k+1} + s - d + \frac{1}{\rho}\lambda^k\|^2,$$

which is the Euclidean projection of $(-Cz^{k+1} + d - \frac{1}{\rho}\lambda^k)$ onto $\mathcal{S}$, that considering its separability, can be solved for each subset s_i^{k+1}, $i \in \mathbb{Z}_1^p$.

B. Particularization to the HMPC's optimization problem

Problem (7) can be recast as (8) by taking $s = (s_b, s_c)$

$$z = (u^0, x^1, u^1, \ldots, x^{N-1}, u^{N-1}, x_e, x_s, x_c, u_e, u_s, u_c),$$

where $s_b = (s_{b,0}, s_{b,1}, \ldots s_{b,N-1})$ with $s_{b,i} \in \mathbb{R}^{n_y}$, $i \in \mathbb{Z}_0^{N-1}$, and $s_c = (s_{c,1}, s_{c,2}, \ldots, s_{c,n_y})$ with $s_{c,i} \in \mathbb{R}^3$, $i \in \mathbb{Z}_1^{n_y}$. That is, $s_{b,i}$ accounts for the constraints (7d) and $s_{c,i}$ for the constraints (7i). Therefore, set $\mathcal{S}$ in (8b) is given by

$$\mathcal{S} = \underbrace{\mathcal{C} \times \mathcal{C} \times \cdots \times \mathcal{C}}_{N} \times \mathcal{D}_1 \times \mathcal{D}_2 \times \cdots \times \mathcal{D}_{n_y},$$

where $\mathcal{D}_i \doteq \mathcal{D}(\overline{y}_{(i)}, \underline{y}_{(i)}) \subset \mathbb{R}^3$, $i \in \mathbb{Z}_1^{n_y}$, as defined in (3), and $\mathcal{C} \doteq \{y \in \mathbb{R}^{n_y} : \underline{y} \leq y \leq \overline{y}\}$. Ingredients G and b account for the equality constraints (7b), (7c), (7e), (7f), (7g) and (7h); C and d for the relationship between z and s; and H and q for the cost function (7a).

Remark 2. *We note that x^0 does not appear in the above definition of z because equation* (7b) *can be implicitly considered in the optimization problem, i.e., taking the first equation of* (7c) *as $x^1 = Ax(t) + Bu^0$, as is typical in MPC solvers.*

By defining z and, in particular, s in this way, we have that the update of s^{k+1} requires two types of projections: the first is a projection onto $\mathcal{C}$, which is a simple projection onto a box, and the second is a projection onto $\mathcal{D}_i$ for each $i \in \mathbb{Z}_1^{n_y}$, whose explicit solution is provided in Theorem 2.

Finally, we must find a way of solving the equality constrained QP (10), corresponding to step (9a) of the ADMM algorithm. There are multiple ways to do this [17]. A popular approach in sparse QP solvers is the one presented in [5, §3.1], in which its KKT conditions are expressed as a linear system of equations whose solution can be sparsely computed using matrix decompositions such as the QR [18] or LDL$^\top$ factorizations [4, §2], [5, §3.1]. Other approaches make use of the Cholesky decomposition [8], leading to very sparse and simple matrices thanks to the simple structure of G in linear MPC. However, in the case of HMPC, we find that the straightforward explicit solution from [19, §10.1.1] provides the best computational results. After some simple algebraic manipulations, this approach leads to

$$z^{k+1} = M_q \hat{q}^k + M_b b, \tag{11}$$

where M_q and M_b are the matrices given by

$$M_q = \hat{H}^{-1} G^\top (G\hat{H}^{-1}G^\top)^{-1} G\hat{H}^{-1} - \hat{H}^{-1},$$
$$M_b = \hat{H}^{-1} G^\top (G\hat{H}^{-1}G^\top)^{-1}.$$

Although matrices M_q and M_b are generally dense, we find this approach is often favorable compared to sparse approaches due to the fact that HMPC is of particular interest when using a small prediction horizon, leading to relatively small-dimensional matrices, where the advantages of the sparse approaches are no longer meaningful. We also note that only the first n_x columns of M_b are required, since all the elements of b are always zero with the exception of the first n_x.

Algorithm 2 shows the particularization of ADMM to the HMPC formulation (7), where we take the same partition of vectors c and λ that we took for s (see the beginning of this subsection) in steps 8 and 11, i.e., $c = (c_b, c_c)$ and $\lambda = (\lambda_b, \lambda_c)$. We note that step 11 is making use of Theorem 1. The algorithm returns a suboptimal solution $(\tilde{z}, \tilde{s})$, where the level of suboptimality is determined by the choice of the exit tolerances $\epsilon_p > 0$ and $\epsilon_d > 0$ [7, §3.3].

Algorithm 2: ADMM for solving HMPC

Require: $x(k) \in \mathbb{R}^{n_x}$, $(x_r, u_r) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$, $z^0 \in \mathbb{R}^n$, $s^0 \in \mathbb{R}^m$, $\lambda^0 \in \mathbb{R}^m$, $\epsilon_p > 0$, $\epsilon_d > 0$

1 $k \leftarrow 0$
2 Update q with x_r and u_r and b with $x(t)$
3 **repeat**
4 $\quad \hat{q}^k \leftarrow q + C^\top \left(\rho(s^k - d) + \lambda^k\right)$
5 $\quad z^{k+1} \leftarrow$ Solution of (10) for $\hat{q}^k$
6 $\quad c \leftarrow Cz^{k+1} - d$
7 $\quad$ **for** $i \in \mathbb{Z}_0^{N-1}$ **do**
8 $\quad\quad s_{b,i}^{k+1} \leftarrow \max(\min(-c_{b,i} - \frac{1}{\rho}\lambda_{b,i}^k,\ \overline{y}),\ \underline{y})$
9 $\quad$ **end for**
10 $\quad$ **for** $i \in \mathbb{Z}_1^{n_y}$ **do**
11 $\quad\quad s_{c,i}^{k+1} \leftarrow \mathcal{P}_{\mathcal{K}_-(\overline{y}_{(i)})}\left(\mathcal{P}_{\mathcal{K}_+(\underline{y}_{(i)})}(-c_{c,i} - \frac{1}{\rho}\lambda_{c,i}^k)\right)$
12 $\quad$ **end for**
13 $\quad c \leftarrow c + s^{k+1}$
14 $\quad \lambda^{k+1} \leftarrow \lambda^k + \rho c$
15 $\quad k \leftarrow k + 1$
16 **until** $\|c\|_\infty \leq \epsilon_p$ **and** $\|s^k - s^{k-1}\|_\infty \leq \epsilon_d$
Output: $\tilde{z} \leftarrow z^k$, $\tilde{s} \leftarrow s^k$

Vector c is used to store the values of $Cz^{k+1} - d$ and $Cz^{k+1} + s - d$. We use it to reduce to one the number of times the operation Cz^{k+1} is performed in each iteration of the algorithm. Even though C is sparse and the multiplication operations in steps 4 and 6 in which it is involved are performed sparsely by storing the matrix using the *compressed sparse row* representation, this reduction can have a significant impact on the computation time of the algorithm, especially if E and F are dense and/or n_y is large.

Remark 3. *The key point of Algorithm 2 is that we are able to project directly onto the sets $\mathcal{D}(\overline{y}_{(i)}, \underline{y}_{(i)})$ thanks to the explicit solution provided by Theorem 2. Otherwise, we would have had to consider the sets $\mathcal{K}_+(\underline{y}_{(i)})$ and $\mathcal{K}_-(\overline{y}_{(i)})$ separately for each $i \in \mathbb{Z}_1^{n_y}$, which would have increased the dimension of s and therefore the computational complexity of the solver.*

V. Numerical results

This section shows two case studies evaluating the performance of the proposed HMPC solver. The first one compares it against other solvers and MPC formulations from the literature. The second one highlights the benefits obtained by considering the sets $\mathcal{D}$ instead of SOC constraints.

A. Application of HMPC to a ball and plate system

In this section we solve the problem from the case study of the original HMPC article [1] using the results obtained in the previous sections as well as with the SCS solver [6]. We also apply state-of-the-art solvers for the MPC for tracking (MPCT) formulation [15] and for the standard MPC formulation with a terminal equality constraint (equMPC) described in [8, Eq. (9)], which are the MPC formulations used to highlight the benefits of HMPC in [1] and [2].

The system under consideration, described in detail in [1, §V.A], is a ball and plate system where the control objective is to steer the position of the ball to a given point by acting on the tilt of the plate through two motors on its main axes whose acceleration we can manipulate.

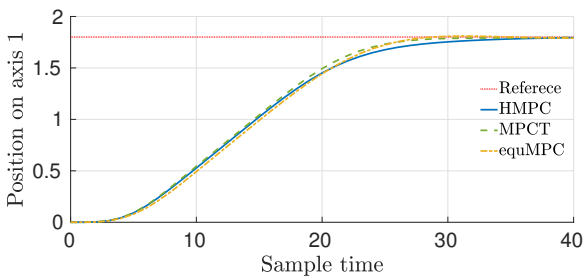
We maintain the same parameters, constraints and setup from [1, §V], including the prediction horizon $N = 5$ and base frequency $w = 0.3254$ (which is selected according to the criteria presented in [1, §VI]) for the HMPC formulation, with two exceptions. First, to improve the numerical conditioning of the resulting optimization problems, we scale the states corresponding to the position of the ball on each axis by a factor of 0.1. Second, due to the previous change, we reduced T_h and S_h by a factor of 10 to maintain nearly indistinguishable closed-loop trajectories from the ones presented in [1, §V]. This resulted in an improvement on the number of iterations of the solvers by up to two orders of magnitude while maintaining the original results.

We show results using the following solvers (formulations):

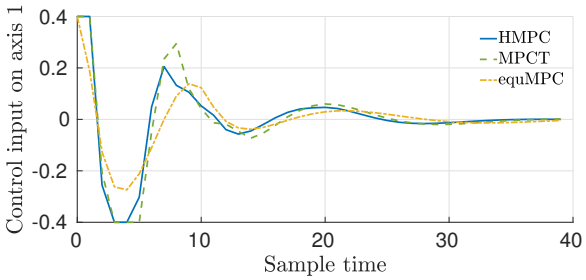
- SCS `v3.0.0` (HMPC) [6]. This state-of-the-art operator splitting solver can be applied to HMPC by imposing constraints (7i) using SOC constraints. In particular, two SOC constraints are required for each constraints in (7i).
- SPCIES `v0.3.7` (HMPC) [9]. This solver implements the ADMM algorithm described in Section IV. We also show the results of a version in which we do not make use of the results presented in Section II, i.e., in which we impose constraints (7i) using SOC constraints as in the SCS solver. Step 5 is solved using the dense method (11), since it provided better computational results.
- SPCIES `v0.3.7` (MPCT and equMPC) [9]. The MPCT formulation is solved using the extended ADMM algorithm presented in [20]. The equMPC formulation is solved using the ADMM algorithm presented in [8] (see [21, §5.4.2] for a more in-depth explanation). The MPCT formulation uses the same parameters as the ones from [1, §V], including the prediction horizon $N = 15$, which is the smallest one for which the closed-loop performance of the system is similar to the one obtained using HMPC with $N = 5$. The prediction horizon of the equMPC formulation is taken as $N = 30$ for this very same reason. We take its Q and R ingredients as the ones used by the other two formulations.

The options of the solvers where left at their default values except for the scaling option of the SCS solver, which was set to 1 since it significantly improved its performance, and the exit tolerances, which were set to 10^{-5}.

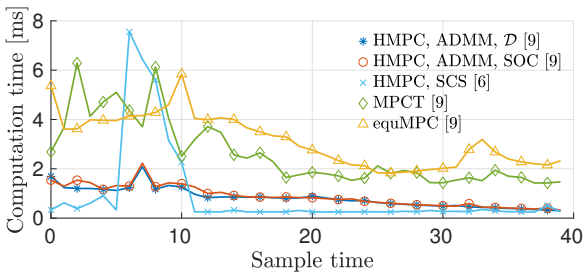
Figure 1 shows the closed-loop results of the linear system with the three MPC formulations. We refer the reader to [1, §V.B] for some results on the application of HMPC to the nonlinear system. Figure 1a shows the position of the ball on axis 1, which converges to its reference 1.8, and Figure 1b shows the control action on axis 1, whose reference is 0. Figures 1c and 1d show the computation times and number of iterations of each solver, with Table I showing relevant information about them. It also shows the penalty parameters used for the solvers from the SPCIES toolbox. The tests are performed on an Intel Core i5-8250U operating at 1.60GHz in Matlab using the C-MEX interface of the solvers.



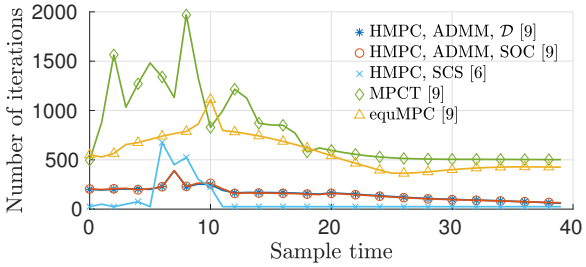
(a) Position of ball on axis 1.



(b) Control input for axis 1.



(c) Computation times.



(d) Number of iterations.

Fig. 1: Closed-loop results on the ball and plate system.

	Computation time [ms]				Number of iterations					
	Average	Median	Maximum	Minimum	Average	Median	Maximum	Minimum	Avrg. [μs/iters.]	Penalty parameters
HMPC, ADMM, $\mathcal{D}$	0.83	0.82	2.09	0.30	154.6	158.0	389	60	5.36	$\rho = 15$
HMPC, ADMM, SOC	0.87	0.83	2.23	0.33	153.8	152.5	390	64	5.77	$\rho = 15$
HMPC, SCS	0.91	0.26	7.56	0.24	78.8	25.0	675	25	11.5	-
MPCT	2.61	1.94	6.29	1.43	789.8	577.0	1969	496	3.31	$\rho_1 = 3000$, $\rho_2 = 100$
equMPC	3.14	3.05	5.84	1.83	567.5	535.0	1112	363	5.54	$\rho = 45$

Penalty parameter ρ_1 is for the constraints listed in [20, Remark 4], and ρ_2 for the rest.

TABLE I: Comparison between the different solvers during the simulation shown in Figure 1.

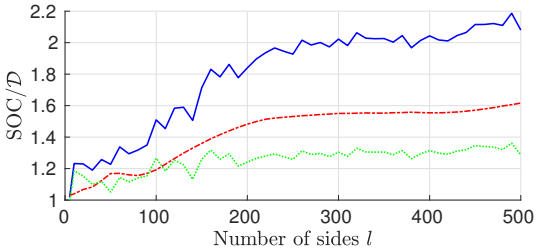


Fig. 2: Comparison of using the $\mathcal{D}$ or SOC constraints. Lines show averages when using the SOC constraints divided by averages when using the $\mathcal{D}$ constraints for increasing values of l. Solid blue line represents the total computation time, dashed red the number of iterations and dotted green the computation time per iteration.

B. Benefits of considering the sets $\mathcal{D}$

The results from the previous subsection seem to indicate that there is little benefit to using the sets $\mathcal{D}$ over the alternative of the SOC constraints. However, this is due to the small dimension and sparsity of matrices E and F in the previous example, which are simply imposing box constraints on four of the states and the two control inputs.

However, for larger dimensions of E and F the advantage of using the sets $\mathcal{D}$ becomes more pronounced. To show this, we take the exact same setup as in the previous subsection, but we include constraints on the position of the ball on the plate in the form of a regular l-sided polygon centered at the origin whose vertices are at a distance of 2 decimeters from the origin. Note that each side of the polygon adds an additional row to the matrices E and F containing two non-zero elements, thus increasing n_y by one.

Figure 2 shows a comparison between the ADMM algorithm using the $\mathcal{D}$ constraints and the SOC constraints for increasing values of l starting from $l = 5$. As can be seen, the average total computation time using the SOC constraints becomes over twice as long as using the $\mathcal{D}$ constraints when n_y becomes sufficiently large.

There are two factors at play that produce these results. On one hand, increasing n_y increases the dimension of s, and thus C. When using the $\mathcal{D}$ constraints this dimension is given by $Nn_y + 3n_y$. Then using the SOC constraints the dimension is given by $Nn_y + 6n_y$. For small values of N, where the HMPC formulation excels, this difference can become significant, becoming more prominent the larger the value of n_y and the larger the number of non-zero elements in E and F, since the cost of the multiplications by C becomes the main computational burden of the algorithm in this scenario. This reduction of the computational cost per iteration leads to the result presented by the dotted green line of Figure 2.

Remark 4. *We note that the above discussion applies when*

the multiplications involving C are performed without taking advantage of its structure, since if sets $\mathcal{D}$ are not used, then the rows of C related to the SOC constraints are (nearly) duplicated. This can be used to reduce the computational burden of performing the operation Cz^{k+1} in Step 6 of Algorithm 2. However, the same cannot be done in Step 4. Thus, our approach still provides a computational reduction even if the particular structure of C is exploited.

The second factor that leads to an increase of the total computation time is the fact that the algorithm requires more iterations to satisfy the exit condition for larger values of s. This is due to the fact that there are more primal and dual variables which must converge to a vicinity of their optimal values. Thus, by using the sets $\mathcal{D}$ we reduce these variables and thus the expected number of iterations of the algorithm. This effect can be seen in the dashed red line of Figure 2. Additionally, the use of sets $\mathcal{D}$ may also lead to a reduction of the number of active constraints, which typically also reduces the number of ADMM iterations.

Remark 5. *The use of sets $\mathcal{D}$ is motivated by our consideration of the system constraints* (5)*. In systems that do not have upper and lower bounds, but instead only have one of the two, the proposed approach is meaningless, since this case would not result in pairs of opposed SOC constraints. We note, however, that* (5) *is a very common constraint in MPC and that our approach also applies to the case of box constraints on states and inputs.*

VI. Discussion and conclusions

This paper discusses how to efficiently solve the HMPC formulation by imposing its SOC-like constraints by using the sets $\mathcal{D}$ defined in (3). The main conclusion is that the HMPC formulation is suitable candidate for its implementation in embedded systems, since we obtain computational results that are in line (or even better) than the ones obtained for other linear MPC formulations. In particular, we derive a few interesting conclusions from the numerical results:

- A comparison between the solvers for HMPC and the ones for MPCT and equMPC shows that a solution of HMPC can be obtained in computation times comparable to the ones for MPC formulations whose optimization problem is a QP using state-of-the-art solvers when comparing prediction horizons for which the closed-loop performances are similar.
- A comparison between the ADMM solvers using the $\mathcal{D}$ or SOC constraints shows that the use of the sets $\mathcal{D}$ can have a significant impact on the computation times, especially for dense matrices E and F and sufficiently large values of n_y. For large prediction horizons, the reduction obtained on matrix C when using sets $\mathcal{D}$ may not be so noticeable, but in the case of HMPC, which is designed to be used with small prediction horizons, it can have a noticeable impact.
- The results using the SCS solver indicate that lower computation times may be obtained if additional aspects, such as numerical preconditioning, adaptation of the penalty parameter, etc., were to be included in the algorithm.

We also note that the results of this paper could be very useful in other optimization settings involving constraints $\mathcal{D}$. An interesting future research line is to extend the solver to be able to deal with the complications that arise in a practical setting due to linearization errors or external disturbances. In particular, the extension of HMPC to the robust case may be a viable and interesting topic for further future research.

Appendix

A. Proof of Theorem 1

This proof makes use of the following well known *projection theorem* (see [11, Prop. 1.1.9]).

Theorem 3 (Projection Theorem)**.** *Let $\mathcal{C}$ be a nonempty closed convex subset of $\mathbb{R}^n$, and let z be a vector in $\mathbb{R}^n$. There exists a unique vector that minimizes $\|v-z\|$ over $v \in \mathcal{C}$, called the projection of z on $\mathcal{C}$. Furthermore, a vector $v^* \in \mathcal{C}$ is the projection of z on $\mathcal{C}$ if and only if $\langle v-v^*, z-v^*\rangle \leq 0,\ \forall v \in \mathcal{C}$.*

The first case is obvious: if $z = (z_0, z_1) \in \mathcal{K}$, then $z = \mathcal{P}_{\mathcal{K}}(z)$. To prove the second case, i.e., $\|z_1\| \leq -\alpha(z_0 - c)$, we note that $(c, 0) \in \mathcal{K}$. We now use Theorem 3 to prove that $(c, 0) = \mathcal{P}_{\mathcal{K}}(z)$. We have that, for any $y = (y_0, y_1) \in \mathcal{K}$,

$$\begin{aligned}
&\langle (y_0, y_1) - (c, 0), (z_0, z_1) - (c, 0)\rangle \\
&= \langle (y_0 - c, y_1), (z_0 - c, z_1)\rangle = \langle y_1, z_1\rangle + (y_0 - c)(z_0 - c) \\
&\overset{(*)}{\leq} \|y_1\|\|z_1\| + (y_0 - c)(z_0 - c) \\
&\overset{(**)}{\leq} -\alpha^2(y_0 - c)(z_0 - c) + (y_0 - c)(z_0 - c) = 0,
\end{aligned}$$

where $(*)$ is due to the Cauchy-Schwarz inequality and $(**)$ holds because $\|z_1\| \leq -\alpha(z_0 - c)$ and $\|y_1\| \leq \alpha(y_0 - c)$. Next, we prove the third case. Let us introduce the notation

$$\hat{z}_0 \doteq \alpha\tau, \quad \tilde{z} \doteq \frac{z_1}{\|z_1\|}, \quad \hat{z}_1 \doteq \alpha\hat{z}_0\tilde{z},$$

where we recall that $\tau = 0.5(\alpha(z_0 - c) + \|z_1\|)$ and note that $\tilde{z}$ is well defined because $\|z_1\| \neq 0$ due to the non-satisfaction of the conditions in (2a) and (2b). We start by proving that $(\hat{z}_0 + c, \hat{z}_1) \in \mathcal{K}$ when *(i)* $\|z_1\| > \alpha(z_0 - c)$ and *(ii)* $\|z_1\| > -\alpha(z_0 - c)$, i.e., that $\|\hat{z}_1\| \leq \alpha(\hat{z}_0 + c - c) = \alpha\hat{z}_0$. Indeed,

$$\|\hat{z}_1\| = \Big\|\tau\frac{z_1}{\|z_1\|}\Big\| = |\tau| \overset{(ii)}{=} \tau, \quad \alpha\hat{z}_0 = \tau.$$

Finally, we show that $\mathcal{P}_{\mathcal{K}}(z) = (\hat{z}_0 + c, \hat{z}_1)$. Pick an arbitrary $y = (y_0, y_1) \in \mathcal{K}$ and fix $z = (z_0, z_1)$ satisfying *(i)* and *(ii)*. The proof follows from the equation displayed at the bottom of the next page, where the steps marked with $(*)$ hold because $\|z_1\| - \alpha\hat{z}_0 = \frac{1}{2}(\|z_1\| - \alpha(z_0 - c)) \overset{(i)}{>} 0$. ∎

B. Proof of Theorem 2

Since we assume that $\underline{z} \leq \overline{z}$, we have from Lemma 1 that $\mathcal{D}$ is non-empty. Thus, the projection of z onto $\mathcal{D}$ exists and is unique, since it is a closed convex set. Moreover, since it is the intersection of two closed convex sets, the iterates of Algorithm 1 will converge to the projection of z onto $\mathcal{D}$ [12, Theorem 2]. We will now show that, in fact, Algorithm 1 will converge after a single iteration. That is, taking first the projection onto $\mathcal{K}_+$ and then onto $\mathcal{K}_-$, we show that $v^2 = w^1$, which along with Lemma 2 proves the claim.

Let us denote $v^k = (v_0^k, v_1^k) \in \mathbb{R} \times \mathbb{R}^{n-1}$ and $w^k = (w_0^k, w_1^k) \in \mathbb{R} \times \mathbb{R}^{n-1}$. We divide the proof into several cases. ***Case 1:*** $z \in \mathcal{K}_-$ and $z \in \mathcal{K}_+$. This case is trivial, since $z \in \mathcal{D}$. ***Case 2:*** $z \in \mathcal{K}_+$ and $\mathcal{P}_{\mathcal{K}_-}(v^1)$ is obtained from (2b). Then, $v^1 = z$, $p^1 = 0$, $w^1 = (\overline{z}, 0)$. Since $\overline{z} \geq \underline{z}$, it is easy to see that $(\overline{z}, 0) \in \mathcal{K}_+$. Therefore, $v^2 = \mathcal{P}_{\mathcal{K}_+}((\overline{z}, 0) + p^1) = (\overline{z}, 0) = w^1$. ***Case 3:*** $z \in \mathcal{K}_+$ and $\mathcal{P}_{\mathcal{K}_-}(v^1)$ is obtained from (2c). Then, $v^1 = z$, $p^1 = 0$,

$$w^1 = \frac{1}{2}(\overline{z} - z_0 + \|z_1\|)\left(-1, \frac{z_1}{\|z_1\|}\right) + (\overline{z}, 0)\,,$$

and $v^2 = \mathcal{P}_{\mathcal{K}_+}(w^1 + p^1) = \mathcal{P}_{\mathcal{K}_+}(w^1)$. All that remains is to show that $w^1 \in \mathcal{K}_+$, which we prove by contradiction. Assume that $w^1 \notin \mathcal{K}_+$, i.e., $\|w_1^1\| > w_0^1 - \underline{z}$, which can be expressed as

$$\left|\frac{1}{2}(\overline{z} - z_0 + \|z_1\|)\right| > \frac{1}{2}(z_0 - \overline{z} - \|z_1\|) + \overline{z} - \underline{z}. \qquad (12)$$

If $\frac{1}{2}(\overline{z} - z_0 + \|z_1\|) > 0$, then (12) leads to the contradiction $\|z_1\| > z_0 - \underline{z}$, since we assume that $z \in \mathcal{K}_+$. If it is smaller of equal to 0, then (12) leads to the contradiction $\underline{z} > \overline{z}$.
Case 4: $\mathcal{P}_{\mathcal{K}_+}(z)$ is obtained from (2b). Then, $v^1 = (\underline{z}, 0)$, $p^1 = z - (\underline{z}, 0)$. Since $\underline{z} \leq \overline{z}$, it is easy to see that $(\underline{z}, 0) \in \mathcal{K}_-$. Therefore, $w^1 = \mathcal{P}_{\mathcal{K}_-}(v^1) = (\underline{z}, 0)$, which leads to

$$\begin{aligned} v^2 &= \mathcal{P}_{\mathcal{K}_+}(w^1 + p^1) = \mathcal{P}_{\mathcal{K}_+}((\underline{z}, 0) + z - (\underline{z}, 0)) \\ &= \mathcal{P}_{\mathcal{K}_+}(z) = (\underline{z}, 0) = w^1. \end{aligned}$$

Case 5: $\mathcal{P}_{\mathcal{K}_+}(z)$ is obtained from (2c). Then,

$$v^1 = \frac{1}{2}(z_0 - \underline{z} + \|z_1\|)\left(1, \frac{z_1}{\|z_1\|}\right) + (\underline{z}, 0)\,, \qquad (13)$$

and $p^1 = z - v^1$. We now distinguish between two subcases.

(i) $v^1 \in \mathcal{K}_-$. Then, $w^1 = v^1$, which leads to, $v^2 = \mathcal{P}_{\mathcal{K}_+}(v^1 + z - v^1) = \mathcal{P}_{\mathcal{K}_+}(z) = v^1 = w^1$.

(ii) $v^1 \notin \mathcal{K}_-$. From (13), the use of simple algebra leads to $\|v_1^1\| = v_0^1 - \underline{z}$. If $\overline{z} > \underline{z}$ this implies that $\|v_1^1\| > v_0^1 - \overline{z}$, and thus w^1 is computed using (2c), i.e.,

$$\begin{aligned} w^1 &= \frac{1}{2}(\overline{z} - v_0^1 + v_0^1 - \underline{z})\left(-1, \frac{v_1^1}{\|v_1^1\|}\right) + (\overline{z}, 0) \\ &= \frac{1}{2}\left(\overline{z} + \underline{z}, \frac{\overline{z} - \underline{z}}{\|v_1^1\|} v_1^1\right), \end{aligned}$$

which along the use of simple algebra leads to $\|w_1^1\| = w_0^1 - \underline{z}$. If, on the other hand, $\overline{z} = \underline{z}$, then w^1 is computed using (2b), i.e., $w^1 = (\overline{z}, 0) = (\underline{z}, 0)$. In both cases it is easy to see that $w^1 \in \mathcal{K}_+$ and thus $v^2 = \mathcal{P}_{\mathcal{K}_+}(w^1) = w^1$.

These cases cover all the possibilities for projecting first onto $\mathcal{K}_+$ and then onto $\mathcal{K}_-$. ■

REFERENCES

[1] P. Krupa, D. Limon, and T. Alamo, "Harmonic based model predictive control for set-point tracking," *IEEE Transactions on Automatic Control*, vol. 67, no. 1, pp. 48–62, 2022.

[2] P. Krupa, M. Pereira, D. Limon, and T. Alamo, "Single harmonic based model predictive control for tracking," in *58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 151–156.

[3] E. F. Camacho and C. B. Alba, *Model Predictive Control*, 2nd ed. London, UK: Springer-Verlag, 2007.

[4] M. Garstka, M. Cannon, and P. Goulart, "COSMO: A conic operator splitting method for convex conic problems," *Journal of Optimization Theory and Applications*, vol. 190, no. 3, pp. 779–810, 2021.

[5] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[6] B. O'Donoghue, "Operator splitting for a homogeneous embedding of the linear complementarity problem," *SIAM Journal on Optimization*, vol. 31, pp. 1999–2023, August 2021.

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

[8] P. Krupa, D. Limon, and T. Alamo, "Implementation of model predictive control in programmable logic controllers," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 1117–1130, 2021.

[9] ——, "SPCIES: Suite of Predictive Controllers for Industrial Embedded Systems," https://github.com/GepocUS/Spcies, Dec 2020.

[10] H. H. Bauschke, "Projection algorithms and monotone operators," Ph.D. dissertation, Theses (Dept. of Mathematics and Statistics)/Simon Fraser University, 1996.

[11] D. P. Bertsekas, *Convex Optimization Theory*. Athena Scientific Belmont, 2009.

[12] J. P. Boyle and R. L. Dykstra, "A method for finding projections onto the intersection of convex sets in Hilbert spaces," in *Advances in order restricted statistical inference*. Springer, 1986, pp. 28–47.

[13] M. Stošić, J. Xavier, and M. Dodig, "Projection on the intersection of convex sets," *Linear Algebra and its Applications*, vol. 509, pp. 191–205, 2016.

[14] H. H. Bauschke and J. M. Borwein, "Dykstra's alternating projection algorithm for two sets," *Journal of Approximation Theory*, vol. 79, no. 3, pp. 418–443, 1994.

[15] A. Ferramosca, D. Limon, I. Alvarado, T. Alamo, and E. Camacho, "MPC for tracking with optimal closed-loop performance," *Automatica*, vol. 45, no. 8, pp. 1975–1978, 2009.

[16] D. Limon, I. Alvarado, T. Alamo, and E. F. Camacho, "MPC for tracking piecewise constant references for constrained linear systems," *Automatica*, vol. 44, no. 9, pp. 2382–2387, 2008.

[17] M. Benzi, G. H. Golub, and J. Liesen, "Numerical solution of saddle point problems," *Acta numerica*, vol. 14, pp. 1–137, 2005.

[18] N. Saraf and A. Bemporad, "A bounded-variable least-squares solver based on stable QR updates," *IEEE Transactions on Automatic Control*, vol. 65, no. 3, pp. 1242–1247, 2019.

[19] S. Boyd, *Convex Optimization*, 7th ed. Cambridge, UK: Cambridge University Press, 2009.

[20] P. Krupa, I. Alvarado, D. Limon, and T. Alamo, "Implementation of model predictive control for tracking in embedded systems using a sparse extended ADMM algorithm," *Transactions on Control Systems Technology*, vol. 30, no. 4, pp. 1798–1805, 2022.

[21] P. Krupa, "Implementation of MPC in embedded systems using first order methods," Ph.D. dissertation, University of Seville, 2021, available at arXiv:2109.02140.

$$\begin{aligned}
\langle (y_0,\ y_1) - (\hat{z}_0 + c, \hat{z}_1), (z_0,\ z_1) - (\hat{z}_0 + c,\ \hat{z}_1) \rangle &= \langle (y_0 - \hat{z}_0 - c,\ y_1 - \alpha\hat{z}_0\tilde{z}), (z_0 - \hat{z}_0 - c,\ (\|z_1\| - \alpha\hat{z}_0)\tilde{z}) \rangle \\
&= \langle y_1 - \alpha\hat{z}_0\tilde{z},\ (\|z_1\| - \alpha\hat{z}_0)\tilde{z} \rangle + (y_0 - \hat{z}_0 - c)(z_0 - \hat{z}_0 - c) \\
&= (\|z_1\| - \alpha\hat{z}_0)\langle y_1,\ \tilde{z} \rangle + \alpha\hat{z}_0(\alpha\hat{z}_0 - \|z_1\|)\langle \tilde{z},\ \tilde{z} \rangle + (y_0 - \hat{z}_0 - c)(z_0 - \hat{z}_0 - c) \\
&\overset{(*)}{\leq} (\|z_1\| - \alpha\hat{z}_0)\|y_1\|\|\tilde{z}\| + \alpha\hat{z}_0(\alpha\hat{z}_0 - \|z_1\|)\|\tilde{z}\|^2 + (y_0 - \hat{z}_0 - c)(z_0 - \hat{z}_0 - c) \\
&\overset{(*)}{\leq} (\|z_1\| - \alpha\hat{z}_0)\alpha(y_0 - c) + \alpha\hat{z}_0(\alpha\hat{z}_0 - \|z_1\|) + (y_0 - \hat{z}_0 - c)(z_0 - \hat{z}_0 - c) \\
&= \alpha(\|z_1\| - \alpha\hat{z}_0)(y_0 - \hat{z}_0 - c) + (y_0 - \hat{z}_0 - c)(z_0 - \hat{z}_0 - c) \\
&= (y_0 - \hat{z}_0 - c)\left(\alpha(\|z_1\| - \alpha\hat{z}_0) + z_0 - \hat{z}_0 - c\right) \\
&= (y_0 - \hat{z}_0 - c)(z_0 - c + \alpha\|z_1\| - 2\hat{z}_0) = (y_0 - \hat{z}_0 - c)(2\alpha\tau - 2\alpha\tau) = 0
\end{aligned}$$