Efficiently Solving the Harmonic Model Predictive Control Formulation

Pablo Krupa ©, Daniel Limon ©, Alberto Bemporad ©, Fellow, IEEE, and Teodoro Alamo ©, Member, IEEE

Abstract—Harmonic model predictive control (HMPC) is a model predictive control (MPC) formulation that 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 the multipliers algorithm, presenting a solver that 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—Alternating direction method of multipliers (ADMM), convex optimization, embedded systems, harmonic model predictive control (HMPC), predictive control.

I. INTRODUCTION

In the recent publication [1] (originally presented in [2]), Krupa et al. 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 slewing-rate constraints.

These advantages, which are highlighted and discussed in detail in [1] and [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.

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 SPoCIES 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 ≥ i, Z_1 denotes the set of integers from i to j, i.e., Z_1 = {i, i + 1, ..., j − 1, j}. We denote by S_n^1 (D_n^1) the set of (diagonal) positive definite matrices in R^n×n. Given a set X ⊆ R^n, we denote by I_X its indicator function, i.e., I_X(x) = 0 if x ∈ X and I_X(x) = +∞ if x /∈ X. For vectors x_1 to x_M, (x_1, x_2, ..., x_M) denotes the column vector formed by their concatenation. Given a vector x ∈ R^n, we denote its ith component using a parentheses subindex x(i). Given two vectors x ∈ R^n and y ∈ R^m, their inner product is denoted by (x, y) = ∑_{i=1}^{n} x(i)y(i). For x ∈ R^n and A ∈ S_n^1, ∥x∥_A = √(x^T A x), ∥x∥_∞ ≤ max_{1≤i≤n} |x(i)|. The Euclidean projection of a vector x ∈ R^n onto a set A ⊆ R^n is denoted by P_A(x), i.e., P_A(x) = arg min_{v ∈ A} ∥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-SOC): A shifted-SOC K_A(c) ⊆ R^n is a set given by

K_A(c) = \{ z = (z_0, z_1) ∈ R × R^{n−1} : ∥z_1∥ ≤ α(z_0 − c) \}

(1)
Algorithm 1: Dykstra’s Algorithm for \( C = C_1 \cap C_2 \).

\[
\begin{array}{l}
\text{Input: } z \in \mathbb{R}^n \\
1 \quad w^0 \leftarrow z, \; p^0 \leftarrow 0_n, \; q^0 \leftarrow 0_n \\
2 \quad \text{foreach } k \geq 1 \text{ do} \\
3 \quad \quad v^k \leftarrow \mathcal{P}_{C_1}(w^{k-1} + p^{k-1}) \\
4 \quad \quad p^k \leftarrow w^{k-1} - v^k \\
5 \quad \quad w^k \leftarrow \mathcal{P}_{C_2}(v^k + q^{k-1}) \\
6 \quad \quad q^k \leftarrow v^k + q^{k-1} - w^k \\
\end{array}
\]

where \( \alpha \in \{1, -1\} \) and \( c \in \mathbb{R} \). For convenience, let us denote by \( \mathcal{K}_c = C_{\alpha \cdot c} \) and \( \mathcal{K}_c = C_{\alpha \cdot c} \), 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 \( \mathcal{K}_c \). Its proof is inspired by the proof of [10, Th. 3.3.6], which proves an explicit solution onto (1) for \( c = 0 \) and \( \alpha > 0 \).

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

\[
\mathcal{P}_c(z) = \begin{cases} 
\begin{cases} 
\text{if } ||z_1|| \leq \alpha \cdot ||z_0 - c|| & (2a) \\
\tilde{\alpha} \cdot c + \frac{z_1}{\alpha} & (2c) \\
\end{cases} \\
\text{if } ||z_1|| \leq -\alpha \cdot ||z_0 - c|| & (2b) \\
\end{cases}
\end{cases}
\]

where \( \tilde{\alpha} = \frac{1}{\alpha}(\alpha \cdot ||z_0 - c|| + ||z_1||) \).

Proof: See Appendix A.

We are now interested in the intersection of two “opposed” shifted-SOCs. Let us denote by \( D(\pi, z) \subset \mathbb{R}^n \), where \( \pi, z \in \mathbb{R} \), the set given by

\[
D(\pi, z) = \{ z \in \mathbb{R}^n : z \in \mathcal{K}_c(z) \}. 
\]

Once again, we may drop the “(\pi, z)” if it is clear from the context. Let \( D \) be 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 nonempty if \( \pi \leq \pi \), as we state in the following lemma.

Lemma 1: Let \( D(\pi, z) \subset \mathbb{R}^n \) be given by (3) for some \( \pi, z \in \mathbb{R} \). Then, \( D \) is nonempty if \( \pi \leq \pi \).

Proof: First, assume that \( D \) is nonempty and take any \( z \in D \). Then, from \( z \in \mathcal{K}_c \) and \( z \in \mathcal{K}_c \), we have that \( ||z_1|| \leq \pi - z_0 \) and \( ||z_1|| \leq z_0 - \pi \), which leads to \( \pi - z_0 \geq 2||z_1|| \geq 0 \). Next, assume that \( \pi \leq \pi \) and consider the vector \( z = ((\pi + \pi)/2, 0) \in \mathbb{R} \times \mathbb{R}^{-1} \). It is easy to verify that \( z \in \mathcal{K}_c \). Therefore, \( \pi \leq \pi \) and \( z \in \mathcal{K}_c \), thus \( z \in D \).

The projection onto set \( D \) 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 nonempty closed convex set \( C = C_1 \cap C_2 \cdots \cap C_r \), where \( r > 0 \) is finite, and it is assumed that the sets \( C_i \) are closed and convex and that \( \mathcal{P}_{C_i} \) has a known solution for \( i \in \mathbb{Z}^r \). 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 \( C_i \) in order, even if \( r = 2 \).

However, we will show that this is not the case for the projection onto set \( D \), which can be obtained by first projecting onto \( K_0 \) and then projecting the resulting vector onto \( K_0 \), both of which have simple explicit solutions given by Theorem 1. This result will allow us to directly use sets \( D \) in future developments without having to resort 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 nonempty set \( D \). 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 \( C = C_1 \cap C_2 \). It generates iterates \( v^k \) and \( w^k \) satisfying \( \|v^k - \mathcal{P}_c(z)\| \to 0 \) and \( \|w^k - \mathcal{P}_c(z)\| \to 0 \) as \( k \to +\infty \) [12, Th. 2].

Lemma 2: Let \( C = C_1 \cap C_2 \) be a nonempty closed convex set and \( C_1, C_2 \subseteq \mathbb{R}^n \) be closed convex sets. Consider Algorithm 1 for finding \( \mathcal{P}_c(z) \) for \( z \in \mathbb{R}^n \). Then, \( v^2 = w^1 \Rightarrow w^1 = \mathcal{P}_c(z) \).

Proof: The first iterate of Algorithm 1 satisfies

\[
v^1 = P_{C_1}(z), \quad p^1 = z - v^1, \quad w^1 = P_{C_2}(v^1), \quad q^1 = v^1 - w^1.
\]

Then, if \( v^2 = w^1 \), we have that

\[
v^2 = P_{C_1}(w^1 + p^1) = w^1, \quad p^2 = w^1 + p^1 - w^1 = p^1 \\
w^2 = P_{C_2}(v^2 + v^1 - w^1) = P_{C_2}(v^1) = w^1 \\
q^2 = v^2 + q^1 - w^2 = q^1.
\]

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

Theorem 2: Let \( z = (z_0, z_1) \in \mathbb{R} \times \mathbb{R}^{-1} \) and \( D(z, \pi) \) be the set given by (3) for some \( \pi, z \in \mathbb{R} \) satisfying \( \pi \leq \pi \). Then, the projection of \( z \) onto \( D \) is given by \( \mathcal{P}_D(z) = \mathcal{P}_{\pi}(\mathcal{P}_{\pi}(z)) \).

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

\[
x(t + 1) = Ax(t) + Bu(t)
\]

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

\[
y \leq E_z(t) + F_u(t) \leq \overline{y}
\]

where we assume that the bounds \( y, \overline{y} \in \mathbb{R}^{n_x} \) satisfy \( y < \overline{y} \).

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 \( x_i(t), u_i(t) \), called the artificial harmonic reference, whose value at each discrete time instant \( j \in Z \) is given by

\[
x_i(t) = x + x_i \sin(w(j - N)) + x_i \cos(w(j - N)) \quad (6a)
\]

\[
u_i(t) = u + u_i \sin(w(j - N)) + u_i \cos(w(j - N)) \quad (6b)
\]

i.e., to use a harmonic signal, thus the name of the formulation, with base frequency \( w \geq 0 \) parameterized by \( x_i, u_i, x, u \in \mathbb{R}^{n_x} \) and \( u_i, u, x_i, x \in \mathbb{R}^{n_u} \). Let us introduce the following notation to simplify future developments:

\[
x_H(t) \doteq (x_i(t), x_i(t)) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_x}
\]

\[
u_H(t) \doteq (u_i(t), u_i(t)) \in \mathbb{R}^{n_u} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_u}
\]

\[
y_c = Ex + Fu, \quad y_s = Ex + Fu, \quad y_c = Ex + Fu.
\]
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 \( (r_x, u_r) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \) is derived from

\[
\min_{x_H, u_H} \sum_{j=0}^{N-1} \ell_h(x^j, u^j, x_{h}^j, u_{h}^j) + V_h(x_H, u_H; x_r, u_r) \quad (7a)
\]

subject to

\[
x_{j+1}^j = Ax^j + Bu^j, \quad j \in \mathbb{Z}_0^{N-1} \quad (7b)
\]

\[
y \leq Ex^j + Fu^j \leq \gamma, \quad j \in \mathbb{Z}_0^{N-1} \quad (7c)
\]

\[
Ax^N + Bu^{N-1} = x_c \quad (7d)
\]

\[
x_c = Ax_c + Bu_c \quad (7e)
\]

\[
x_s \cos(w) - x_s \sin(w) = Ax_s + Bu_s \quad (7f)
\]

\[
x_s \sin(w) + x_c \cos(w) = Ax_c + Bu_c \quad (7g)
\]

\[
(y_{e(i)}, y_{s(i)}, y_{e(i)}) \in D \quad (7h)
\]

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

\[
\ell_h(x, u, x_h, u_h) = \|x - x_h\|^2_F + \|u - u_h\|^2_R
\]

where \( Q \in \mathbb{R}^{n_x \times n_x} \) and \( R \in \mathbb{R}^{n_u \times n_u} \); and the offset cost function

\[
V_h(j) = \|x_e - x_r\|^2_T + \|u_e - u_r\|^2_S
\]

\[
+ \|x_s\|^2_T + \|x_c\|^2_T + \|u_s\|^2_S + \|u_c\|^2_S
\]

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

As is typical in MPC, the summation of the stage cost function \( \ell_h \) 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., \( x_h^j, u_h^j \), respectively. The offset cost function is penalizing the following two conceptually different things.

1) The distances \( \|x_e - x_r\|^2_T \) and \( \|u_e - u_r\|^2_S \). This penalization will make the “center” of the artificial harmonic reference signal move toward the reference, eventually reaching it if an admissible steady-state of the system.

2) 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 \( (x_c, u_c) \) as \( k \to +\infty \), since it will converge toward \( x_s = x_c = 0 \) and \( u_s = u_c = 0 \) as \( k \to +\infty \).

The result of this, as stated and proven in [1, Th. 3], is that the closed-loop system will asymptotically converge to \((x_c, u_c)\) if it is an admissible steady-state of the system, or to the admissible steady-state \((x_h^j, u_h^j)\) that minimizes the distance \( \|x - x_h\|^2_T + \|u - u_h\|^2_S \) 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., \( x^N = x_N \), and (7f) that \((x_c, u_c)\) 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 \( x_{h+1}^j = Ax_h^j + Bu_h^j, \forall j \) (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] and [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 \( K_+ \) and one on \( K_- \). 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 \( D \) 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], and [6].

IV. ADMM SOLVER FOR HMPC

This section shows how the HMPC formulation can be solved by 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

\[
\min_{z,s} \frac{1}{2} z^\top Hz + q^\top z \quad (8a)
\]

subject to

\[
Gz = b, \quad s \in S \quad (8b)
\]

\[
Cz + s = d \quad (8c)
\]

where \( z \in \mathbb{R}^n \) are the primal decision variables, \( s \in \mathbb{R}^m \) are the primal slack variables, \( H \in \mathbb{R}^{n \times n} \), \( q \in \mathbb{R}^n \), \( G \in \mathbb{R}^{m \times n} \), \( b \in \mathbb{R}^m \), \( C \in \mathbb{R}^{m \times m} \), \( d \in \mathbb{R}^m \) and \( S \) is a Cartesian product of the form \( S = S_1 \times S_2 \times \cdots \times S_p \), where \( S_i \subseteq \mathbb{R}^{m_i} \), \( i = 1, \ldots, p \), \( m = \sum_{i=1}^p m_i \) are nonempty closed convex sets, i.e., we consider a partition of \( s \) given by \( s = (s_1, s_2, \ldots, s_p) \), where each \( s_i \in \mathbb{R}^{m_i} \), \( i \in \mathbb{Z}_+ \), must belong to \( S_i \).

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

\[
\min_{s,z} f(z) + g(s)
\]

subject to

\[
Cz + s = d
\]

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

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

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 \((z^0, s^0, \lambda^0)\), the iterations of ADMM consist of the following steps [7]:

\[
\begin{align*}
    z^{k+1} &= \arg \min_z \mathcal{L}(z, s^k, \lambda^k) \\
    s^{k+1} &= \arg \min_s \mathcal{L}(z^{k+1}, s, \lambda^k) \\
    \lambda^{k+1} &= \lambda^k + \rho (Cz^k + s^{k+1} - d)
\end{align*}
\]  

(9a)

(9b)

(9c)

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

\[
    z^{k+1} = \arg \min_z \left\{ \frac{1}{2} \|z - \hat{H}z + (\hat{q})^\top z \|_2 \text{s.t. } Gz = b \right\}
\]  

(10)

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

\[
    s^{k+1} = \arg \min_{s \in S} \rho \frac{1}{2} \|Cz^k + s - d + \frac{1}{\rho} \lambda^k\|_2^2
\]  

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

B. Partialization to the HMPC’s Optimization Problem

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

\[
    z = (u^0, x^1, u^1, \ldots, x^{N-1}, u^{N-1}, x_c, x_i, u_c, u_s)
\]

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

\[
    S = C \times C \times \cdots \times C \times D_1 \times D_2 \times \cdots \times D_N
\]

where \(D_i \doteq \{ \gamma(i) : \gamma(i) \} \subseteq \mathbb{R}^{3}, i \in \mathbb{Z}_N^{N_y}\), as defined in (3), and \(C \doteq \{ y \in \mathbb{R}^{N_y} : -y \leq y \leq \eta \}\). 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 (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 \(C\), which is a simple projection onto a box, and the second is a projection onto \(D_i\) for each \(i \in \mathbb{Z}_N^{n_y}\), 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 LDLT 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

\[
    z^{k+1} = M_q \hat{q}^k + M_k b
\]  

(11)

where \(M_q\) and \(M_k\) are the matrices given by

\[
    M_q = \hat{H}^{-1}G^\top \left( G\hat{H}^{-1}G^\top \right)^{-1} G\hat{H}^{-1} - \hat{H}^{-1}
\]

\[
    M_k = \hat{H}^{-1}G^\top \left( G\hat{H}^{-1}G^\top \right)^{-1}.
\]

Although matrices \(M_q\) and \(M_k\) 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 \(n_y\) columns of \(M_k\) are required, since all the elements of \(b\) are always zero with the exception of the first \(n_x\).

Algorithm 2 shows the partialization of ADMM to the HMPC formulation (7), where we take the same partition of vectors \(c\) and \(\lambda\) 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 \((\bar{z}, \bar{s})\), where the level of suboptimality is determined by the choice of the exit tolerances \(\epsilon_p > 0\) and \(\epsilon_d > 0\) [7, §3.3].

\[
    \epsilon_p < \epsilon_p \quad \text{and} \quad \|s^k - s^{k-1}\|_\infty \leq \epsilon_d
\]

Output: \(\bar{z} \leftarrow s^k, \bar{s} \leftarrow s^k\)

V. Numerical Results

This section shows two case studies evaluating the performance of the proposed HMPC solver. The first one compares it against other
savers and MPC formulations from the literature. The second one highlights the benefits obtained by considering the sets $D$ 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 in 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).

1) 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 constraint in (7i).

2) 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.

3) 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.

We take the options of the solvers were 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 $10^{-5}$.

Fig. 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. Fig. 1(a) shows the position of the ball on axis 1, which converges to its reference $1.8$, and Fig. 1(b) shows the control action on axis 1, whose reference is $0$. Fig. 1(c) shows the computation times and the 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 i5-8250 U operating at 1.60 GHz in Matlab using the C-MEX interface of the solvers.

B. Benefits of Considering the Sets $D$

The results from the previous subsection seem to indicate that there is little benefit to using the sets $D$ 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.
Comparison of using the constraints for increasing values of \( n \) becomes sufficiently large.

... containing two nonzero elements, thus and are performed without taking advantage of the SOC constraints. Lines show averages when using the SOC constraints divided by averages when using the \( D \) constraints for increasing values of \( l \). Solid blue line represents the total computation time, dashed red represents the number of iterations, and dotted green represents the computation time per iteration.

![Fig. 2](image)

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

However, for larger dimensions of \( E \) and \( F \) the advantage of using the sets \( D \) becomes more pronounced. To show this, we take the exact same setup as in the previous section, 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 nonzero elements, thus increasing \( n_y \) by one.

Fig. 2 shows a comparison between the ADMM algorithm using the \( D \) 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 \( D \) 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 \( D \) constraints this dimension is given by \( N n_y + 3n_y \), whereas using the SOC constraints the dimension is given by \( N n_y + 6n_y \). 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 nonzero 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 Fig. 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 \( D \) 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 that must converge to a vicinity of their optimal values. Thus, by using the sets \( D \), 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 Fig. 2. Additionally, the use of sets \( D \) 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 \( D \) 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 article discusses how to efficiently solve the HMPC formulation by imposing its SOC-like constraints by using the sets \( D \) defined in (3). The main conclusion is that the HMPC formulation is a 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, which are as follows.

1) 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.

2) A comparison between the ADMM solvers using the \( D \) or SOC constraints shows that the use of the sets \( D \) 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 \( D \) 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.

3) 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 article could be very useful in other optimization settings involving constraints \( D \). 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.
This proof makes use of the following well-known projection theorem (see [11, Prop. 1.1.9]).

Theorem 3 (Projection Theorem): Let 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 \( v \in C \), called the projection of \( z \) on \( C \). Furthermore, a vector \( v' \in C \) is the projection of \( z \) on \( C \) if and only if \( \langle v - v', z - v' \rangle \leq 0 \), \( \forall v \in C \).

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

\[
((y_0, y_1) - (c, 0), (z_0, z_1) - (c, 0)) = (y_0 - c, y_1, (z_0 - c, z_1)) = (y_1, (z_0 - c) + (y_0 - c)(z_0 - c))
\]

\[
\leq \| y_1 \| \| z_1 \| + (y_0 - c)(z_0 - c)
\]

\[
\leq -\alpha^2(y_0 - c)(z_0 - c) + (y_0 - c)(z_0 - c) = 0
\]

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

\[
\tilde{z}_0 \triangleq \alpha \tau, \quad \tilde{z} \triangleq \frac{z_1}{\| z_1 \|}, \quad \tilde{z}_1 \triangleq \alpha \tilde{z}_0 \tilde{z},
\]

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

\[
\| \tilde{z}_0 \| = \| \tilde{z}_1 \| \quad | \tau | (i) = \tau, \quad \alpha \tilde{z}_0 = \tau.
\]

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

Since we assume that \( \tilde{z} \leq \tilde{z} \), we have from Lemma 1 that \( D \) is nonempty. Thus, the projection of \( z \) onto \( D \) 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 \( D \) [12, Th. 2]. We will now show that, in fact, Algorithm 1 will converge after a single iteration. That is, taking first the projection onto \( K_+ \) and then onto \( K_- \), we show that \( v^2 = w^1 \), which along with Lemma 2 proves the claim.

Let us denote \( v^1 = (v_{0}^1, v_{1}^1) \in \mathbb{R} \times \mathbb{R}^n \) and \( w^k = (w_{0}^k, w_{1}^k) \in \mathbb{R} \times \mathbb{R}^n \). We divide the proof into several cases.

Case 1: \( z \in K_- \) and \( z \in K_+ \). This case is trivial, since \( z \in D \).

Case 2: \( z \in K_+ \) and \( \mathcal{P}_K(v^1) \) is obtained from (2b). Then, \( v^1 = z, p^1 = 0, w^1 = (\tau, 0) \). See (2d), it is easy to see that \( (\tau, 0) \in K_+ \).

Therefore, \( v^2 = \mathcal{P}_K((\tau, 0) + p^1) = (\tau, 0) = w^1 \).

Case 3: \( z \in K_- \) and \( \mathcal{P}_K(v^1) \) is obtained from (2c). Then, \( v^1 = z, p^1 = 0 \)

\[
w^1 = \frac{1}{2}(\tau - z_0 + ||z_1||) - (\tau, z_0 - ||z_1||) + (\tau, 0)
\]

and \( v^2 = \mathcal{P}_K(v^1 + p^1) = \mathcal{P}_K(w^1) \). All that remains is to show that \( w^1 \in K_+ \), which we prove by contradiction. Assume that \( w^1 \not\in K_+ \), i.e., \( ||w^1|| > ||v^1 - z|| \), which can be expressed as

\[
\frac{1}{2}(\tau - z_0 + ||z_1||) > \frac{1}{2}(z_0 - \tau - ||z_1||) + \tau - \tilde{z}
\]

(12)

If \( \frac{1}{2}(\tau - z_0 + ||z_1||) > 0 \), then (12) leads to the contradiction \( ||z_1|| > z_0 - \tilde{z} \), since we assume that \( z \in K_+ \). If it is smaller of equal to 0, then (12) leads to the contradiction \( \tilde{z} > \tau \).

Case 4: \( \mathcal{P}_K(z) \) is obtained from (2b). Then, \( v^1 = (\tau, 0), p^1 = z - (\tau, 0) \). Since \( \tilde{z} \leq \tilde{z} \), it is easy to see that \( (\tau, 0) \in K_+ \). Therefore, \( v^1 = \mathcal{P}_K(v^1) = (\tau, 0) \), which leads to

\[
v^2 = \mathcal{P}_K(v^1 + p^1) = \mathcal{P}_K((\tau, 0) + z - (\tau, 0)) = \mathcal{P}_K(z) = (\tau, 0) = w^1.
\]

Case 5: \( \mathcal{P}_K(z) \) is obtained from (2c). Then

\[
v^1 = \frac{1}{2}(z_0 - \tau + ||z_1||) - (\tau, z_0 - ||z_1||) + (\tau, 0)
\]

(13)

and \( p^1 = z - v^1 \). We now distinguish between two subcases.

1) \( v^1 \in K_+ \). Then, \( v^1 = v^1 \), which leads to, \( v^2 = \mathcal{P}_K(v^1 + z - v^1) = \mathcal{P}_K(z) = v^1 = w^1 \).
2) \( v^1 \notin \mathcal{K}_- \). From (13), the use of simple algebra leads to \( \|v^1\| = v_0 - z \). If \( \sigma \geq \frac{z}{\|v^1\|} \) this implies that \( \|v^1\| > v_0 - \frac{z}{\|v^1\|} \), and thus, \( w^1 \) is computed using (2b), i.e.,

\[
\begin{align*}
w^1 &= \frac{1}{2} \left( \sigma - v_0 + v^1 - \frac{z}{\|v^1\|} \right) \left( -1, \frac{v^1}{\|v^1\|} \right) + (\sigma, 0) \\
&= \frac{1}{2} \left( \sigma + \frac{z}{\|v^1\|} \right) \left( 1, \frac{v^1}{\|v^1\|} \right)
\end{align*}
\]

which along the use of simple algebra leads to \( \|w^1\| = w_0 - \frac{z}{\|v^1\|} \). If, on the other hand, \( \sigma = \frac{z}{\|v^1\|} \) then \( w^1 \) is computed using (2b), i.e.,

\[
w^1 = (\sigma, 0) = \left( \frac{z}{\|v^1\|}, 0 \right).
\]

In both cases, it is easy to see that \( w^1 \in \mathcal{K}_- \) and thus \( v^2 = \mathcal{T}_{\mathcal{K}_-}(w^1) = w^1 \).

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 Trans. Autom. Control*, vol. 67, no. 1, pp. 48–62, Jan. 2022.

2. P. Krupa, M. Pereira, D. Limon, and T. Alamo, “Single harmonic based model predictive control for tracking,” in *Proc. 58th Conf. Decis. Control*, 2019, pp. 151–156.

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

4. M. Gurska, M. Cannon, and P. Goulart, “COSMO: A conic operator splitting method for convex conic problems,” *J. Optim. Theory Appl.*, 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,” *Math. Program. Comput.*, vol. 12, no. 4, pp. 637–672, 2020.

6. B. O’Donoghue, “Operator splitting for a homogeneous embedding of the linear complementarity problem,” *SIAM J. Optim.*, vol. 31, pp. 1999–2023, Aug. 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,” *Found. Trends Mach. Learn.*, 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 Trans. Control Syst. Technol.*, vol. 29, no. 3, pp. 1117–1130, May 2021.

9. P. Krupa, D. Limon, and T. Alamo, “SPCIES: Suite of predictive controllers for industrial embedded systems,” 2020. [Online] Available: https://github.com/GepocUS/Spices

10. H. H. Bauschke, “Projection algorithms and monotone operators,” Ph.D. dissertation, Dept. Math. Statist., Simon Fraser Univ., Burnaby, BC, Canada, 1996.

11. D. P. Bertsekas, *Convex Optimization Theory*. Belmont, MA, USA: Athena Scientific, 2009.

12. J. R. 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*. Berlin, Germany: Springer, 1986, pp. 28–47.

13. M. Stošić, J. Xavier, and M. Dodig, “Projection on the intersection of convex sets,” *Linear Algebra Appl.*, vol. 509, pp. 191–205, 2016.

14. H. H. Bauschke and J. M. Borwein, “Dykstra’s alternating projection algorithm for two sets,” *J. 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 Trans. Autom. Control*, vol. 65, no. 3, pp. 1242–1247, Mar. 2019.

19. S. Boyd, *Convex Optimization*, 7th ed. Cambridge, U.K.: Cambridge Univ. 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,” *Trans. Control Syst. Technol.*, vol. 30, no. 4, pp. 1798–1805, 2022.

21. P. Krupa, “Implementation of MPC in embedded systems using first order methods,” Ph.D. dissertation, Syst. Eng., Automat., Univ. Seville, Sevilla, Spain, 2021, [arXiv:2109.02140].