Implementation of model predictive control for tracking in embedded systems using a sparse extended ADMM algorithm

Pablo Krupa, Ignacio Alvarado, Daniel Limon, Teodoro Alamo

Abstract—This article presents a sparse, low-memory footprint optimization algorithm for the implementation of the model predictive control (MPC) for tracking formulation in embedded systems. This MPC formulation has several advantages over standard MPC formulations, such as an increased domain of attraction and guaranteed recursive feasibility even in the event of a sudden reference change. However, this comes at the expense of the addition of a small amount of decision variables to the MPC’s optimization problem that complicates the structure of its matrices. We propose a sparse optimization algorithm, based on an extension of the alternating direction method of multipliers, that exploits the structure of this particular MPC formulation. We describe the controller formulation and detail how its structure is exploited by means of the aforementioned optimization algorithm. We show closed-loop simulations comparing the proposed solver against other solvers and approaches from the literature.

Index Terms—Model predictive control, embedded optimization, embedded systems, extended ADMM

I. INTRODUCTION

Model Predictive Control (MPC) is an advanced control strategy in which the control action is obtained, at each sample time, from the solution of an optimization problem where a prediction model is used to forecast the evolution of the system over a finite prediction horizon. One of the main advantages of MPC over other control strategies is that it inherently considers and satisfies system constraints [1].

There are many different MPC formulations in the literature, each of which is defined by an optimization problem with different objective function and/or set of constraints [2]. In general, the optimization problem is posed as a minimization problem in which the objective function penalizes the distance between the reference and the predicted system evolution over the prediction horizon. In this paper we focus on linear MPC controllers for tracking whose optimization problem can be posed as a quadratic programming (QP) problem.

Since an MPC controller requires solving an optimization problem at each sample time, its use has historically been confined to computationally powerful devices, such as PCs. However, as evidenced by the references provided below, there is a growing interest in the literature in the implementation of these controllers in devices with very limited computational and memory resources, known as embedded systems.

One possible approach for implementing MPC in embedded systems is to use explicit MPC [3], in which the solution of the MPC optimization problem is stored as a lookup table that is computed offline. However, this lookup table can become prohibitively large for medium to large-sized systems and/or for MPC problems with many constraints. Some examples of it being implemented in embedded systems are [4]. [5].

Another approach comes from the recent development of optimization solvers for QP problems that are tailored to embedded systems. A few of the most widespread ones include OSQP [6], qpOASES [7], CVXGEN [8], FiOrdOs [9] and FORCES [10]. These solvers, although used to successfully implement MPC controllers in embedded systems (see [11], [12], [13], [14] for a few examples), are for generic QP problems. Therefore, the development of optimization algorithms tailored to the specific MPC optimization problem can potentially provide better results. Some examples of solvers that fall into this category include PRESAS [15], FalcOpt [16], HPMPC [17] or μAO-MPC [18]. Other noteworthy examples of MPC-tailored solvers implemented in embedded systems include [19], [20], [21] for implementations in FPGAs, [22] for microcontrollers and [23], [24], [25], [26], [27] for programmable logic controllers. Specifically, the authors proposed in [23] and [24] a method for the embedded implementation of MPC controllers which takes advantage of the structure of the matrices of the QP problem. This led to the development of a sparse optimization algorithm, based either on the FISTA [28] or on the alternating direction method of multipliers (ADMM) [29] algorithms.

The references on embedded MPC provided above, as well as most publications in this field, deal with standard MPC formulations, presenting efficient ways to solve them. In this paper, however, we are concerned with providing an efficient solver for a more intricate linear MPC formulation known as MPC for tracking (MPCT) [30], which has several advantages over standard MPC formulations that makes it especially attractive for its implementation in embedded systems. For instance, it offers a significantly larger domain of attraction than standard MPC formulations with the same prediction
horizon, which is useful in embedded systems since the use of smaller prediction horizons reduces the computational and memory footprint of the controller. Additionally, it provides recursive feasibility even in the event of a sudden reference change \cite{31}.

However, these advantages come at the cost of a slightly more complicated QP problem due to the inclusion of additional decision variables. In this paper we propose a sparse solver for MPCT based on the extended ADMM (EADMM) algorithm \cite{32}. The use of this algorithm leads to sparse matrices whose simple structure can be efficiently exploited to attain a computationally and memory efficient implementation following a similar approach to the one used in \cite{32}, but for a more intricate MPC formulation. In particular, we avoid representing the sparse matrices using the typical sparse-matrix representations (e.g., \textit{compressed sparse column}).

We showcase the benefits of the proposed solver by comparing it to other ADMM-based approaches to control a simulated ball and plate system. Additionally, the results of its implementation in a Raspberry Pi to control a two-wheeled inverted pendulum robot have been reported in \cite{33}, providing further evidence of its good performance.

The proposed solver is part of the Matlab toolbox “SPCIES: Suite of Predictive Controllers for Industrial Embedded Systems” \cite{34}, available at \url{https://github.com/GepocUS/Species}.

The remainder of this paper is structured as follows. Section \textbf{II} describes the problem formulation and control objective. The MPC for tracking formulation is described in Section \textbf{III}. The extended ADMM algorithm is detailed in Section \textbf{IV}. Section \textbf{V} describes how the MPCT optimization problem is recast into a problem solvable by the EADMM algorithm. Section \textbf{VI} shows the numerical results. Finally, conclusions are drawn in Section \textbf{VII}.

\textit{Notation:} Given two vectors $x,y \in \mathbb{R}^n$, $x \leq (\geq) y$ denotes componentwise inequalities and $(x,y)$ is their standard inner product. For a vector $x \in \mathbb{R}^n$ and a positive definite matrix $A \in \mathbb{R}^{n \times n}$, $\|x\|_2 = \sqrt{x^T A x}$ is its weighted Euclidean norm, and $\|x\|_\infty = \max_{i=1,\ldots,n} |x_i|$.

\section{Problem formulation}

We consider a controllable system described by a discrete linear time-invariant state-space model

$$x_{k+1} = Ax_k + Bu_k,$$

where $x_k \in \mathbb{R}^n$ and $u_k \in \mathbb{R}^m$ are the state and input of the system at sample time $k$, respectively. Additionally, we consider that the system is subject to the box constraints

$$x \leq x_k \leq \pi,$$  

$$u \leq u_k \leq \pi.$$  

The control objective is to steer the system to the given reference $(x_r,u_r)$ while satisfying the system constraints \cite{2}. This will only be possible if the reference is an admissible steady state of the system, which we formally define as follows.

\textbf{Definition 2.} An ordered pair $(x_a, u_a) \in \mathbb{R}^n \times \mathbb{R}^m$ is said to be an admissible steady state of system (\ref{eq:1}) subject to (\ref{eq:2}) if

(i) $x_a = Ax_a + Bu_a$, i.e., it is a steady state of system (\ref{eq:1}),

(ii) $x \leq x_a \leq \pi$,

(iii) $u \leq u_a \leq \pi$.

If the given reference is not an admissible steady state of system (\ref{eq:1}) subject to (\ref{eq:2}), then we wish to steer the system to the closest admissible steady state, for some given criterion of closeness.

\section{Model Predictive Control for Tracking}

The MPCT formulation \cite{30} differs from other standard MPC formulations in the inclusion of a pair of decision variables $(x_s,u_s)$ known as the \textit{artificial reference}. The cost function penalizes, on one hand, the difference between the predicted states and control actions with this artificial reference, and on the other, the discrepancy between the artificial reference and the reference $(x_r,u_r)$ given by the user. In particular, this paper focuses on the MPCT formulation shown below, which uses a terminal equality constraint.

For a given prediction horizon $N$, the MPCT control law for a given state $x$ and reference $(x_r,u_r)$ is derived from the solution of the following convex optimization problem

$$\min_{x_s,u_s} \sum_{k=0}^{N-1} \left( \|x_i - x_s\|^2_Q + \|u_i - u_s\|^2_S \right) + \|x_s - x_r\|^2_T + \|u_s - u_r\|^2_S$$

\textit{s.t.} $x_0 = x$ \hspace{1cm} \textit{(3a)}

$$x_{i+1} = Ax_i + Bu_i, \quad i \in \mathbb{Z}^{N-1} \hspace{1cm} \textit{(3b)}$$

$$x \leq x_i \leq \pi, \quad i \in \mathbb{Z}^{N-1} \hspace{1cm} \textit{(3c)}$$

$$u \leq u_i \leq \pi, \quad i \in \mathbb{Z}^{N-1} \hspace{1cm} \textit{(3d)}$$

$$x_s = Ax_s + Bu_s \hspace{1cm} \textit{(3e)}$$

$$x_{N-1} = x_{N-1} \hspace{1cm} \textit{(3f)}$$

where the decision variables are the predicted states and inputs $x = (x_0,\ldots,x_N)$, $u = (u_0,\ldots,u_{N-1})$ and the artificial reference $(x_s,u_s)$; the positive definite matrices $Q \in \mathbb{R}^{n \times n}$, $R \in \mathbb{R}^{m \times m}$, $T \in \mathbb{R}^{n \times n}$ and $S \in \mathbb{R}^{m \times m}$ are the cost function matrices; and $\varepsilon_x \in \mathbb{R}^n$, $\varepsilon_u \in \mathbb{R}^m$ are arbitrarily small positive vectors which are added to avoid a possible loss of controllability when the constraints are active at the equilibrium point \cite{31}.
One of the properties of the MPCT formulation is that it will steer the closed-loop system to the admissible steady state \((x_a, u_a)\) that minimizes the cost \(\|x_a - x_r\|_T^2 + \|u_a - u_r\|_S^2\) [30], [31]. Additionally, as previously mentioned in the introduction, this formulation provides significantly larger domains of attraction than other standard MPC formulations, and guarantees recursive feasibility even in the event of a sudden reference change.

IV. EXTENDED ADMM

This section introduces the extended ADMM algorithm [32], which, as its name suggests, is an extension of the ADMM algorithm [29] to optimization problems with more than two separable functions in the objective function. In particular, we focus on the following class of optimization problem.

Let \(θ_i : \mathbb{R}^{m_i} \to \mathbb{R}\) for \(i \in \mathbb{Z}_3^3\) be convex functions, \(Z_1 \subseteq \mathbb{R}^{m_1}\) for \(i \in \mathbb{Z}_3^3\) be closed convex sets, \(C_i \in \mathbb{R}^{m_i \times n_i}\) for \(i \in \mathbb{Z}_3^3\) and \(b \in \mathbb{R}^{m_3}\). Consider the optimization problem

\[
\begin{align*}
\min_{z_1, z_2, z_3} & \sum_{i=1}^3 θ_i(z_i) \\
\text{s.t.} & \sum_{i=1}^3 C_i z_i = b \\
& z_i \in Z_i, \ i \in \mathbb{Z}_3^3.
\end{align*}
\]

and let its augmented Lagrangian \(L_ρ(z_1, z_2, z_3, λ)\) be given by

\[
L_ρ(\cdot) = \sum_{i=1}^3 θ_i(z_i) + (λ, \sum_{i=1}^3 C_i z_i - b) + \frac{ρ}{2} \sum_{i=1}^3 \|C_i z_i - b\|_2^2,
\]

where \(λ \in \mathbb{R}^{m_3}\) are the dual variables and \(ρ > 0\) is the penalty parameter. We denote a solution point of (4) by \((z_1^∗, z_2^∗, z_3^∗, λ^∗)\), assuming that one exists.

Algorithm 1 shows the implementation of the extended ADMM algorithm for a given exit tolerance \(ε > 0\) and initial points \((z_0^0, z_0^1, z_0^2, λ^0)\). The superscript \(k\) indicates the value of the variable at iteration \(k\). We note that step \(9\) uses the \(ℓ∞\)-norm, although any other norm can be used. Algorithm 1 returns an \(ε\)-suboptimal solution \((z_k^0, z_k^1, z_k^2, λ_k)\) of problem (4).

\[
\begin{align*}
2 & \text{repeat} \\
3 & z_{k+1}^i \leftarrow \text{arg min}_{z_i} \{L_ρ(z_1^k, z_2^k, z_3^k, λ_k^i) \mid z_i \in Z_i \} \\
4 & z_{k+1}^2 \leftarrow \text{arg min}_{z_2} \{L_ρ(z_1^k, z_2^k, z_3^k, λ_{k+1}) \mid z_2 \in Z_2 \} \\
5 & z_{k+1}^3 \leftarrow \text{arg min}_{z_3} \{L_ρ(z_1^k, z_2^k, z_3^k, λ_{k+1}) \mid z_3 \in Z_3 \} \\
6 & \Gamma \leftarrow \sum_{i=1}^3 C_i z_{k+1}^i - b \\
7 & λ_{k+1} \leftarrow λ_k + ρ\Gamma \\
8 & k \leftarrow k + 1 \text{ until } \|Γ\|_∞ ≤ ε, \ \|z_k^2 - z_{k-1}^2\|_∞ ≤ ε, \ \|z_k^3 - z_{k-1}^3\|_∞ ≤ ε.
\end{align*}
\]

Output: \(z_k^0, z_k^1, z_k^2, z_k^3, λ_k\)

Remark 1. The exit condition given in step \(9\) of Algorithm 1 serves as an indicator of the (sub-)optimality of the current iterate [32] §5.

V. SOLVING MPCT USING EXTENDED ADMM

This section describes how problem (3) is solved using Algorithm 1. The objective is to develop a memory and computationally efficient algorithm so that it can be implemented in an embedded system. To this end, we recast problem (3) so that steps \(3\) and \(6\) are easy to follow using a similar approach to the one taken in [23]. Algorithm 2 shows the particularization of Algorithm 1 that results from this effort.

A. Recasting the MPCT problem

Let us define \(\tilde{x}_i \equiv x_i - x_s\) and \(\tilde{u}_i \equiv u_i - u_s\). Then, we can rewrite (3) as

\[
\begin{align*}
\min_{x_i, u_i, x_s, u_s} & \sum_{i=0}^{N-1} \left(\|\tilde{x}_i\|_Q^2 + \|\tilde{u}_i\|_R^2 + \|x_s - x_r\|_T^2 + \|u_s - u_r\|_S^2\right) \\
\text{s.t.} & x_0 = x \\
& \tilde{x}_{i+1} = A\tilde{x}_i + Bu_i, \ i \in \mathbb{Z}_0^{N-1} \\
& \tilde{x}_{i+1} \leftarrow x_{i+1}, \ i \in \mathbb{Z}_0^{N-1} \\
& \|\tilde{x}_i\|_∞ \leq \|x_{i+1}\|_∞ \leq \|x_i\|_∞ \\
& \|\tilde{x}_i\|_∞ \leq \|u_{i+1}\|_∞ \leq \|u_i\|_∞ \\
& \|\tilde{x}_i\|_∞ \leq \|x_{i+1}\|_∞ \leq \|x_i\|_∞ \\
& \|\tilde{x}_i\|_∞ \leq \|u_{i+1}\|_∞ \leq \|u_i\|_∞ \\
& x_s = Ax_s + Bu_s \\
& \tilde{x}_i + x_s - x_i = 0, \ i \in \mathbb{Z}_0^N \\
& \tilde{u}_i + u_s - u_i = 0, \ i \in \mathbb{Z}_0^N \\
& x_N = x_s \\
& u_N = u_s,
\end{align*}
\]

where the decision variables are \(\tilde{x} = (\tilde{x}_0, ..., \tilde{x}_N), \ \tilde{u} = (\tilde{u}_0, ..., \tilde{u}_N), \ x = (x_0, ..., x_N), \ u = (u_0, ..., u_N), \ x_s\) and \(u_s\). Equality constraints (6) and (7) impose the congruence of the decision variables with the original problem. We note that inequalities (8) and (9) are omitted.
because they are already imposed by (6f) and (6g) alongside the inclusion of (6k) and (6l).

Specifically, the first inclusion of (6k) and (6l) because they are already imposed by (6f) and (6g) alongside which results in an optimization problem due to the inclusion of (6k) and (6l).

We can now obtain a problem of form (4) by taking

\[
\begin{align*}
    z_1 &= (x_0, u_0, x_1, u_1, \ldots, x_{N-1}, u_{N-1}, x_N, u_N), \quad (7a) \\
    z_2 &= (x_s, u_s), \quad (7b) \\
    z_3 &= (\tilde{x}_0, \tilde{u}_0, \tilde{x}_1, \ldots, \tilde{x}_{N-1}, \tilde{u}_{N-1}, \tilde{x}_N, \tilde{u}_N), \quad (7c)
\end{align*}
\]

which leads to \(\theta_1(z_1) = 0\),

\[
\begin{align*}
    \theta_2(z_2) &= \frac{1}{2}z_2^T \text{diag}(T, S) z_2 - (Tx_r, Su_r)^T z_2, \\
    \theta_3(z_3) &= \frac{1}{2}z_3^T \text{diag}(Q, R, R, \ldots, Q, R) z_3,
\end{align*}
\]

\[
\begin{align*}
    C_1 &= \begin{bmatrix} I_n & 0 & 0 \\ -I_{n+m} & 0 & 0 \\ 0 & \cdots & 0 \\ 0 & 0 & -I_{n+m} \\ 0 & 0 & 0 \end{bmatrix}, \\
    C_2 &= \begin{bmatrix} 0 \\ I_{n+m} \\ \vdots \\ I_{n+m} \\ 0 \end{bmatrix}, \\
    C_3 &= \begin{bmatrix} I_n & 0 & 0 \\ 0 & \cdots & 0 \\ 0 & 0 & I_{n+m} \\ 0 & \cdots & 0 \end{bmatrix}, \\
    b &= \begin{bmatrix} x \\ 0 \\ \vdots \\ 0 \end{bmatrix}
\end{align*}
\]

Matrices \(C_1, C_2\) and \(C_3\) contain the equality constraints (6b), (6f), (6g), (6k) and (6l). Specifically, the first \(n\) rows impose constraint (6b), the last \(n + m\) rows impose the constraints (6f) and (6g), and the rest of the rows impose the constraints (6k) and (6l). Set \(Z_1\) is the set of vectors \(z_1\) (7a) for which the box constraints (6d)-(6g) are satisfied; set \(Z_2\) is the set of vectors \(z_2\) (7b) that satisfy the equality constraint (6b); and set \(Z_3\) is the set of vectors \(z_3\) (7c) that satisfy the equality constraints (6c).

We note that our selection of \(z_i\) and \(C_i\) for \(i \in Z_1^3\) results in an optimization problem that satisfies Assumption 1. Therefore, under a proper selection of \(\rho\), the iterates of the EADMM algorithm will converge to the optimal solution of the MPCT controller. In practice, the parameter \(\rho\) may be selected outside the range shown in Theorem 1 in order to improve the convergence rate of the algorithm [32]. In this case, the convergence will not be guaranteed and will have to be extensively checked with simulations.

B. Particularizing EADMM to the MPCT problem

By taking \(z_i\) and \(C_i\) for \(i \in Z_1^3\) as shown in Section 4A, we can particularize Algorithm 1 to the MPCT problem, resulting in Algorithm 2. This algorithm requires solving three QP problems (which we label \(P_1\), \(P_2\) and \(P_3\) in the following) with explicit solutions at each iteration. The control action to be applied to the system are the elements \(u_0\) of the variable \(z_1^*\) (7a) returned by Algorithm 2.

Step 3 minimizes the Lagrangian (5) over \(z_1\), resulting in the following box-constrained QP problem,

\[
\begin{align*}
    P_1(z_2^*, z_3^*, \lambda^k) : \min_{z_1} & \quad \frac{1}{2} z_1^T H_1 z_1 + q_1^T z_1 \\
    \text{s.t.} & \quad z_1 \leq z_1^* \\
    \end{align*}
\]

where \(H_1 = \rho C_1^T C_1\),

\[
\begin{align*}
    q_1 &= \rho C_1^T C_2 \tilde{x}_2^k + \rho C_1^T C_3 z_3^k + C_1^T \lambda^k - \rho C_1^T b, \\
    z_1 &= (M_n, u_0, x_1, \ldots, x_N + \varepsilon, x_N + u), \\
    z_1^* &= (M_n, u_0, x_1, \ldots, x_N, x_N + \varepsilon, x_N + u),
\end{align*}
\]

and \(M_n \in \mathbb{R}^n > 0\) has arbitrarily large components.

Due to the structure of \(C_1\), matrix \(H_1\) is a positive definite diagonal matrix. As such, each element \(j \in Z_1^3\) of the optimal solution of \(P_1\) denoted by \((z_1^*)_j\), can be explicitly computed as,

\[
(z_1^*)_j = \max \left\{ -\frac{(q_1)_j}{(H_1)_j}, (z_1)_j \right\}, \quad (z_1)_j
\]

Remark 2. In this article we consider box constraints (2) so that \(z_1^{k+1}\) is very simple to update, as shown in (8). However, the use of more generic constraints \(y \leq Cx + Du \leq y\) would also be possible. In this case, (8) would result in \(N + 1\) decoupled inequality-constrained QPs, which would have to be solved at each iteration of the algorithm, but that, given their small scale, may be solved rather cheaply using interior point or active set methods.

Step 4 minimizes the Lagrangian (5) over \(z_2 = (x_s, u_s)\), resulting in the following equality-constrained QP problem,

\[
\begin{align*}
    P_2(z_1^k, z_3^k, \lambda^k) : \min_{z_2} & \quad \frac{1}{2} z_2^T H_2 z_2 + q_2^T z_2 \\
    \text{s.t.} & \quad G_{22} z_2 = b_2,
\end{align*}
\]

where \(H_2 = \text{diag}(T, S) + \rho C_2^T C_2, G_2 = [(A - I_n) B],\)

\[
\begin{align*}
    b_2 &= 0 \quad \text{and} \\
    q_2 &= -(Tx_r, Su_r)^T + \rho C_2^T C_1 z_1^{k+1} + \rho C_2^T C_3 z_3^k + C_2^T \lambda^k - \rho C_2^T b
\end{align*}
\]

This problem has an explicit solution derived from the following proposition [30] §10.11.

Proposition 1. Consider an optimization problem \(\min(1/2) z^T H z + q^T z, \ s.t. \ G z = b\), where \(H\) is positive definite. A vector \(z^*\) is an optimal solution of this problem if and only if there exists a vector \(\mu\) such that,

\[
\begin{align*}
    G z^* &= b \\
    H z^* + q + G^T \mu &= 0,
\end{align*}
\]

which using simple algebra and defining \(W_H = G H^{-1} G^T\), leads to

\[
\begin{align*}
    W_{H \mu} &= -(G H^{-1} q + b) \quad (9a) \\
    z^* &= -H^{-1}(G^T \mu + q). \quad (9b)
\end{align*}
\]

The optimal solution \(z_2^*\) of problem (7c) can be obtained by substituting (9a) into (9b), which leads to the expression

\[
\begin{align*}
    z_2^* &= M_2 q_2, \quad (10)
\end{align*}
\]
where \( M_2 = H_2^{-1}G_2^T(G_2H_2^{-1}G_3^2)^{-1}G_2H_2^{-1} - H_2^{-1} \in \mathbb{R}^{(n+m) \times (n+m)} \). This matrix, which has a relatively small dimension, is computed offline and stored in the embedded system. Vector \( b_2 \) does not appear in the above expression because it is equal to zero.

Step 5 minimizes the Lagrangian (5) over \( z_3 \), resulting in the following equality-constrained QP problem,

\[
P_3(z_1^{k+1}, z_2^{k+1}, \lambda^k) : \min_{z_3} \frac{1}{2} z_3^T H_3 z_3 + q_3^T z_3 \quad \text{s.t.} \quad G_3 z_3 = b_3,
\]

where \( H_3 = \text{diag}(Q, R, R, \ldots, Q, R) + \rho C_3^T C_3, b_3 = 0, \)

\[
q_3 = \rho C_3^T C_1 z_1^{k+1} + \rho C_3^T C_2 z_2^{k+1} + C_3^T \lambda^k - \rho C_3^T b,
\]

\[
G_3 = \begin{bmatrix}
A & B & -I_n & 0 & \cdots & \cdots & 0 & 0 \\
0 & A & B & -I_n & 0 & \cdots & \cdots & 0 \\
0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\
0 & 0 & 0 & 0 & A & B & -I_n & 0
\end{bmatrix}.
\]

This problem can be sparsely solved using the following approach. Let \( W_{H_3} = G_3 H_3^{-1} G_3^3 \). Due to the sparse structure of \( G_3 \) and the fact that \( H_3 \) is a block diagonal matrix, we have that the Cholesky factorization of \( W_{H_3} \), that is, the upper-triangular matrix \( W_{H_3,c} \) that satisfies \( W_{H_3} = W_{H_3,c} W_{H_3,c} \), has the following structure,

\[
W_{H_3,c} = \begin{pmatrix}
\beta_1^1 & \alpha_1^0 & 0 & 0 \\
. & \beta_2^1 & \alpha_2^0 & 0 \\
. & . & \ddots & \ddots \\
0 & . & \ddots & \beta_{N-1}^1 & \alpha_{N-1}^0 \\
0 & . & . & 0 & \beta_N^1
\end{pmatrix},
\]

where we define the sets of matrices \( A = \{\alpha_1^0, \ldots, \alpha_{N-1}^0\}, \alpha_i \in \mathbb{R}^{n \times n}; \) and \( B = \{\beta_1^1, \ldots, \beta_N^1\}, \beta_i \in \mathbb{R}^{n \times n} \). Note that the amount of memory required to store the sets of matrices \( A \) and \( B \) grows linearly with the prediction horizon \( N \).

Then, (9a) can be solved by consecutively solving the following two systems of equations that use the auxiliary vector \( \mu \),

\[
\begin{align*}
W_{H_3,c}^T \mu &= -G_3 H_3^{-1} q_3 \quad (12a) \\
W_{H_3,c} \mu &= \hat{\mu}, \quad (12b)
\end{align*}
\]

which are easy to solve due to \( W_{H_3,c} \) being upper-triangular. In fact, these two systems of equations can be computed sparsely by exploiting the structures of \( W_{H_3,c}, G_3 \) and \( H_3 \). The following two systems of equations use the auxiliary vector \( \tilde{\mu} \),

\[
\begin{align*}
\tilde{\mu}_3 &= -H_3^{-1}(G_3^T \mu + q_3), \quad (13)
\end{align*}
\]

which, once again, can be computed sparsely by exploiting the structures of \( G_3 \) and \( H_3^{-1} \).

**Remark 3.** Because of the controllability of the system, the fact that \( \rho > 0 \) and the positive definite nature of matrices \( Q, R, T \) and \( S \), it is easy to verify that matrices \( G_2 H_2^{-1} G_3^2 \) and \( G_3 H_3^{-1} G_3^3 \) are guaranteed to be nonsingular.

**Remark 4.** It has been shown that the performance of ADMM can be significantly improved by having different values of \( \rho \) for different constraints [6, §5.2], i.e., by considering \( \rho \) as a diagonal positive definite matrix. In particular, we find that, for our problem, the convergence improves significantly if the equality constraints (6b) are penalized more than the others.

### Algorithm 2: Extended ADMM for MPCT

**Require:** \( z_0^0, z_0^3, \lambda^0, \rho > 0, \epsilon > 0 \)

1. \( k \leftarrow 0 \)

2. **repeat**

3. Obtain \( z_2^{k+1} \) by solving \( P_1[z_2^k, z_3^k, \lambda^k] \) using (8)

4. Obtain \( z_2^{k+1} \) by solving \( P_2[z_2^k, z_3^{k+1}, \lambda^k] \) using (10)

5. Obtain \( z_3^{k+1} \) by solving \( P_3[z_1^k, z_2^{k+1}, \lambda^k] \) using (12) and (13).

6. \( \Gamma \leftarrow \sum_{i=1}^3 C_i z_i^{k+1} - b \)

7. \( \lambda^{k+1} \leftarrow \lambda^k + \rho \Gamma \)

8. \( k \leftarrow k + 1 \)

**until** \( \| \Gamma \|_\infty \leq \epsilon, \| z_2^{k} - z_2^{k-1} \|_\infty \leq \epsilon, \| z_2^{k} - z_3^{k-1} \|_\infty \leq \epsilon \)

**Output:** \( z_1^k, z_2^k, z_3^k, \lambda^k \)

**VI. Numerical Results**

The approach we present in this article for solving the MPCT formulation (3) decouples the decision variables into three separate optimization problems in order to attain a simple structure that can be efficiently exploited using an approach similar to the one presented in [23]. However, there are other ways to solve (3). In this section we compare the performance of Algorithm [2] with some direct alternatives by applying them to the ball and plate system described in [44, §V.A]. This system consists of a plate that pivots around its center.
point and whose inclination can be manipulated by changing the angle of its two perpendicular axes. The system has 8 states and 2 control inputs, and its control objective is to steer the ball to a given position on the plate. The control input \( u = (u_1(0), u_2(0)) \in \mathbb{R}^2 \) is the angle acceleration of each of the axes of the plate, which we constrain to \(|u_i(0)| \leq 0.2, i \in \mathbb{Z}_1^2\). We start the system with the ball placed in the central position of the plate and set the reference to the point \((1.5, 1.4)\). For a more in-depth explanation of the system we refer the reader to [44] and [45].

We perform closed-loop tests of the above setup using:
- Algorithm 2 taking \( \hat{z}_2^1 = 0, z_2^0 = 0, \lambda_0 = 0, \) and \( \rho = 40 \) for the constraints listed in Remark 4 and \( \rho = 2 \) for the rest. We set the exit tolerance to \( \epsilon = 10^{-4} \).
- The OSQP solver (version 0.6.0) [46]. Problem (3) can be directly cast as a QP problem by taking the decision variables as
\[
z = (x_0, u_0, x_1, u_1, \ldots, x_{N-1}, u_{N-1}, x_s, u_s).
\]

In this case, the Hessian is no longer block diagonal, which means we cannot use the approach from [23]. However, the QP problem is still sparse (both the Hessian and the matrix that defines the equality constraints are sparse). Therefore, a sparse solver for general QP problems should solve it efficiently. A good choice in our case is OSQP, since it is a well known QP solver based on ADMM. We use its default options with a few exceptions: the exit tolerances are all set to \( \epsilon = 10^{-4} \) and the warmstart procedure is disabled to better compare its ADMM algorithm with the other alternatives we list.
- Extended state space: we recover a block diagonal Hessian by extending the state space with the artificial reference. That is, by defining \( z_i = (x_i, x_s) \) and \( \hat{u}_i = (u_i, u_s), \) \( i \in \mathbb{Z}_0^{N-1} \), problem (3) can be rewritten so that the resulting QP has a block diagonal Hessian at the expense of increasing the number of decision variables and constraints, including additional constraints that ensure that \( x_s \) and \( u_s \) are the same in all the prediction steps. We solve the resulting QP problem using the sparse ADMM-based solver from [23], where the matrices are stored using standard sparse-matrix representations. We take \( \rho = 1 \) and the exit tolerance as \( \epsilon = 10^{-4} \).
- Standard MPC: the ADMM-based solver presented in [23] for the standard MPC formulation shown in equation (9) of [23]. This comparison is of interest because, in terms of iteration complexity, the proposed solver is very similar to the ADMM-based solver from [23], in that each iteration of the algorithm requires (i) solving a system of equations [12] and (ii) computing [8]. The main difference between the two is that Algorithm 2 also requires the evaluation of [10] at each iteration, which is computationally cheap in comparison to the other steps. Therefore, the comparison with [23] serves as an indication of the “price-to-pay” for the inclusion of the artificial reference. We take \( \rho = 1 \) and the exit tolerance as \( \epsilon = 10^{-4} \).}

Remark 8. There are many possible efficient alternatives for solving (2) other than the ADMM-based approaches discussed here, including interior point methods [47], active set methods [48], and even other non-ADMM Lagrangian methods [49]. However, we restrict our comparison here to the use of ADMM-based approaches since our objective is to evaluate if the use of the EADMM algorithm is worthwhile when compared to similar alternatives, and not to determine if it outclasses other optimization methods.

For the MPCT formulation we take \( N = 30 \),
\[
Q = \text{diag}(10, 0.05, 0.05, 0.05, 10, 0.05, 0.05, 0.05),
\]
\[
R = \text{diag}(0.5, 0.5), \quad S = \text{diag}(0.3, 0.3),
\]
\[
T = \text{diag}(600, 50, 50, 50, 600, 50, 50, 50).
\]

For the MPC formulation from [23] we take the same \( N, Q, R \) and \( T \) shown above. The tests are run on a Linux laptop (Intel i5 processor) using Matlab mex files compiled with the gcc compiler using the -O3 flag. With the obvious exception of OSQP, all solvers were generated using version v0.1.5 of the SPCIES toolbox [34]. The penalty parameters were hand tuned to achieve a good performance of the solvers.

Figure 1 shows the results of the tests. Figures 1a and 1b show the trajectory of the position of the ball in each axis and the control inputs, respectively, using the MPCT controller. The trajectories are only shown once because the results using each of the MPCT solvers are indistinguishable to the naked eye. The trajectories for the standard MPC formulation are not shown because we are only interested in its computational results. They are very similar to the ones obtained with MPCT. Figures 1c and 1d show the computation time and the number of iterations of each solver, respectively.

The advantage of the proposed approach can be seen in its iteration complexity. The average computation times (in milliseconds) per iteration of the results shown in Figure 1 are 0.0071 for the proposed solver, 0.0062 for the solver from [23] applied to the standard MPC formulation, 0.0193 for OSQP and 0.0279 for the extended state space approach. That is, compared to the solver from [23], the addition of the artificial reference only increases the computation time per iteration by 15.1% (in this particular example).

However, we find that, in general, the EADMM algorithm takes more iterations to converge than the other solvers. Even so, it can still outperform other sparse solvers due to its low iteration complexity, especially if applied to larger systems and/or with large prediction horizons, where the advantage of not storing all the non-zero elements of the matrices and of performing the matrix-vector operations by direct identification of the structures becomes more prevalent. For smaller systems and short prediction horizons, the advantages of the small memory footprint and the iteration complexity may not outweigh the higher typical number of iterations. In any case, the use of one approach or another would have to be determined in a case by case basis. For instance, in the results shown in Figure 1 the proposed solver outperforms OSQP and the sparse solver based on extending the state space, in spite of it requiring more iterations.
Typically, the number of iterations of first order methods can become large if there are active constraints in the optimal solution. As shown in Figure 1b, this was not the case during the first few sample times, in spite of the control actions reaching their upper bound. However, this phenomenon may occur if there are additional active constraints along the prediction horizon (particularly so if there are active state constraints). This is true for the proposed solver as well as for any other ADMM-based approach.

Figure 2 shows the convergence of the decision variables of the solvers obtained from the SPCIES toolbox to the optimal solution of their QP problems during the first sample time of the results shown in Figure 1. For Algorithm 2, decision variable $\mathbf{z}_k$ is used to compute the distance. The optimal solutions of the QP problems are obtained using the quadprog solver from Matlab. The results show that the convergence of the EADMM algorithm is non-monotone and slower than the ones obtained using ADMM. Non-monotonicity is a phenomenon found in some classes of first order methods [49].

Further evidence of the good performance of the proposed solver can be found in [50] §5.8 and in [33], which shows its implementation in a Raspberry Pi to control a real two-wheeled inverted pendulum robot in real-time.

VII. CONCLUSIONS

This article presents a sparse solver for the linear MPC for tracking formulation based on the extended ADMM algorithm. We show how the use of this method, along with an appropriate selection of decision variables, leads to an efficient solver with simple matrix structures that can be exploited to attain a small iteration complexity and memory footprint. This statement is supported by numerical results comparing the proposed solver to other ADMM-based approaches, which suggest that, from a computational point of view, the proposed approach can provide better results. The solver is available in the SPCIES toolbox for Matlab [34] at https://github.com/GepocUS/Spcies.

REFERENCES

[1] E. F. Camacho and C. B. Alba, Model Predictive Control. Springer Science & Business Media, 2013.
[2] J. B. Rawlings, D. Q. Mayne, and M. Diehl, Model Predictive Control: Theory, Computation, and Design, 2nd ed. Nob Hill Publishing, 2017.
[3] P. Tøndel, T. A. Johansen, and A. Bemporad, “An algorithm for multi-parametric quadratic programming and explicit MPC solutions,” Automatica, vol. 39, no. 3, pp. 489–497, 2003.
[4] G. Valencia-Palomino and J. Rossiter, “Novel programmable logic controller implementation of a predictive controller based on Laguerre functions and multiparametric solutions,” IET control theory & applications, vol. 6, no. 8, pp. 1003–1014, 2012.
[5] J. Velagić and B. Šabić, “Design, implementation and experimental validation of explicit MPC in programmable logic controller,” in 2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE). IEEE, 2014, pp. 93–98.
Acceptor version of the article published in IEEE Transactions on Control Systems Technology. Changes were made to this version by the publisher prior to publication.

The final version of record is available at [http://dx.doi.org/10.1109/TCST.2021.3138824](http://dx.doi.org/10.1109/TCST.2021.3138824)