

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.

Department of Systems Engineering and Automation, Universidad de Sevilla, 41092, Seville, Spain (e-mails: pkrupa@us.es, ialvarado@us.es, dml@us.es, talamo@us.es). Corresponding author: Pablo Krupa.

This work was supported in part by the Agencia Estatal de Investigación (AEI) under Grant PID2019-106212RB-C41/AEI/10.13039/501100011033, in part by the MCIU-Spain and FSE under Grant FPI-2017, and in part by the Junta de Andalucía and the Fondos Europeos para el Desarrollo Regional (FEDER) under Grant P20_00546.

Accepted version of the article published in IEEE Transactions on Control Systems Technology. DOI: 10.1109/TCST.2021.3128824

©2020 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

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 μ AO-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 [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 [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 [23], but for a more intricate MPC formulation. In particular, we avoid representing the sparse matrices using the typical sparse-matrix representations (e.g., *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 [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” [34], which is available at <https://github.com/GepocUS/Spies>.

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

Notation: Given two vectors $x, y \in \mathbb{R}^n$, $x \leq (\geq) y$ denotes componentwise inequalities and $\langle x, y \rangle$ is their standard inner product. For a vector $x \in \mathbb{R}^n$ and a positive definite matrix $A \in \mathbb{R}^{n \times n}$, $\|x\| \doteq \sqrt{\langle x, x \rangle}$, $\|x\|_A \doteq \sqrt{\langle x, Ax \rangle}$ is its weighted Euclidean norm, and $\|x\|_\infty \doteq \max_{i=1 \dots n} |x_{(i)}|$, where $x_{(i)}$ is the i -th element of x , is its ℓ_∞ -norm. For a symmetric matrix A , $\|A\|$ denotes its spectral norm. Given scalars and/or matrices M_1, M_2, \dots, M_N (not necessarily of the same dimensions), we denote by $\text{diag}(M_1, M_2, \dots, M_N)$ the block diagonal matrix formed by the diagonal concatenation of M_1 to M_N . Given a matrix $A \in \mathbb{R}^{n \times m}$, $A_{i,j}$ denotes its (i, j) -th element, A^\top its transposed and A^{-1} its inverse (if it is non-singular). (x_1, x_2, \dots, x_N) is a column vector formed by the concatenation of column vectors x_1 to x_N . 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. $\mathbb{Z}_i^j \doteq \{i, i+1, \dots, j-1, j\}$.

Definition 1. A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is said to be μ -strongly convex, for some scalar $\mu > 0$, if $f(x) - \frac{\mu}{2}\|x\|^2$ is convex.

II. PROBLEM FORMULATION

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

$$x_{k+1} = Ax_k + Bu_k, \quad (1)$$

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

$$\underline{x} \leq x_k \leq \bar{x}, \quad (2a)$$

$$\underline{u} \leq u_k \leq \bar{u}. \quad (2b)$$

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

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 (1) subject to (2) if

- (i) $x_a = Ax_a + Bu_a$, i.e., it is a steady state of system (1),
- (ii) $\underline{x} \leq x_a \leq \bar{x}$,
- (iii) $\underline{u} \leq u_a \leq \bar{u}$.

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

III. MODEL PREDICTIVE CONTROL FOR TRACKING

The MPCT formulation [30] differs from other standard MPC formulations in the inclusion of a pair of decision variables (x_s, u_s) known as the *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

$$\min_{x_s, u_s} \sum_{i=0}^{N-1} (\|x_i - x_s\|_Q^2 + \|u_i - u_s\|_R^2) + \|x_s - x_r\|_T^2 + \|u_s - u_r\|_S^2 \quad (3a)$$

$$\text{s.t. } x_0 = x \quad (3b)$$

$$x_{i+1} = Ax_i + Bu_i, \quad i \in \mathbb{Z}_0^{N-1} \quad (3c)$$

$$\underline{x} \leq x_i \leq \bar{x}, \quad i \in \mathbb{Z}_1^{N-1} \quad (3d)$$

$$\underline{u} \leq u_i \leq \bar{u}, \quad i \in \mathbb{Z}_0^{N-1} \quad (3e)$$

$$x_s = Ax_s + Bu_s \quad (3f)$$

$$\underline{x} + \varepsilon_x \leq x_s \leq \bar{x} - \varepsilon_x \quad (3g)$$

$$\underline{u} + \varepsilon_u \leq u_s \leq \bar{u} - \varepsilon_u \quad (3h)$$

$$x_N = x_s, \quad (3i)$$

where the decision variables are the predicted states and inputs $\mathbf{x} = (x_0, \dots, x_N)$, $\mathbf{u} = (u_0, \dots, u_{N-1})$ 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 [31].

One of the properties of the MPCT formulation (3) 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 $\theta_i : \mathbb{R}^{n_i} \rightarrow \mathbb{R}$ for $i \in \mathbb{Z}_1^3$ be convex functions, $\mathcal{Z}_i \subseteq \mathbb{R}^{n_i}$ for $i \in \mathbb{Z}_1^3$ be closed convex sets, $C_i \in \mathbb{R}^{m_z \times n_i}$ for $i \in \mathbb{Z}_1^3$ and $b \in \mathbb{R}^{m_z}$. Consider the optimization problem

$$\min_{z_1, z_2, z_3} \sum_{i=1}^3 \theta_i(z_i) \quad (4a)$$

$$s.t. \sum_{i=1}^3 C_i z_i = b \quad (4b)$$

$$z_i \in \mathcal{Z}_i, i \in \mathbb{Z}_1^3. \quad (4c)$$

and let its augmented Lagrangian $\mathcal{L}_\rho(z_1, z_2, z_3, \lambda)$ be given by

$$\mathcal{L}_\rho(\cdot) = \sum_{i=1}^3 \theta_i(z_i) + \langle \lambda, \sum_{i=1}^3 C_i z_i - b \rangle + \frac{\rho}{2} \left\| \sum_{i=1}^3 C_i z_i - b \right\|^2, \quad (5)$$

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

Algorithm 1 shows the implementation of the extended ADMM algorithm for a given exit tolerance $\epsilon > 0$ and initial points $(z_2^0, z_3^0, \lambda^0)$. 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 $(\tilde{z}_1^*, \tilde{z}_2^*, \tilde{z}_3^*, \tilde{\lambda}^*)$ of problem (4). As shown in [35], the EADMM algorithm is not necessarily convergent under the typical assumptions of the classical ADMM algorithm. However, multiple results have shown its convergence under additional assumptions [32], [36], [37] or by adding additional steps [38], [39]. In particular, [32] proved its convergence under the following assumption, as stated in the following theorem.

Assumption 1 ([32], Assumption 3.1). *Functions θ_1 and θ_2 are convex; function θ_3 is μ_3 -strongly convex for some $\mu_3 > 0$; and C_1 and C_2 are full column rank.*

Theorem 1 (Convergence of EADMM; [32], Theorem 3.1). *Suppose that Assumption 1 holds and that the penalty parameter $\rho \in \left(0, \frac{6\mu_3}{17\|C_1^T C_3\|}\right)$. Then, the sequence of points (z_1^k, z_2^k, z_3^k) generated by Algorithm 1 converges to a point in the optimal set of problem (4) as $k \rightarrow \infty$.*

Algorithm 1: Extended ADMM

Require : $z_2^0, z_3^0, \lambda^0, \rho > 0, \epsilon > 0$
1 $k \leftarrow 0$
2 repeat
3 $z_1^{k+1} \leftarrow \arg \min_{z_1} \{\mathcal{L}_\rho(z_1, z_2^k, z_3^k, \lambda^k) \mid z_1 \in \mathcal{Z}_1\}$
4 $z_2^{k+1} \leftarrow \arg \min_{z_2} \{\mathcal{L}_\rho(z_1^{k+1}, z_2, z_3^k, \lambda^k) \mid z_2 \in \mathcal{Z}_2\}$
5 $z_3^{k+1} \leftarrow \arg \min_{z_3} \{\mathcal{L}_\rho(z_1^{k+1}, z_2^{k+1}, z_3, \lambda^k) \mid z_3 \in \mathcal{Z}_3\}$
6 $\Gamma \leftarrow \sum_{i=1}^3 C_i z_i^{k+1} - b$
7 $\lambda^{k+1} \leftarrow \lambda^k + \rho \Gamma$
8 $k \leftarrow k + 1$
9 until $\|\Gamma\|_\infty \leq \epsilon, \|z_2^k - z_2^{k-1}\|_\infty \leq \epsilon, \|z_3^k - z_3^{k-1}\|_\infty \leq \epsilon$
Output: $\tilde{z}_1^* \leftarrow z_1^k, \tilde{z}_2^* \leftarrow z_2^k, \tilde{z}_3^* \leftarrow z_3^k, \tilde{\lambda}^* \leftarrow \lambda^k$

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, 4 and 5 are easy to solve following 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 $\tilde{x}_i \doteq x_i - x_s$ and $\tilde{u}_i \doteq u_i - u_s$. Then, we can rewrite (3) as

$$\min_{\tilde{\mathbf{x}}, \tilde{\mathbf{u}}, \mathbf{x}, \mathbf{u}, x_s, u_s} \sum_{i=0}^N (\|\tilde{x}_i\|_Q^2 + \|\tilde{u}_i\|_R^2) + \|x_s - x_r\|_T^2 + \|u_s - u_r\|_S^2 \quad (6a)$$

$$s.t. x_0 = x \quad (6b)$$

$$\tilde{x}_{i+1} = A\tilde{x}_i + B\tilde{u}_i, i \in \mathbb{Z}_0^{N-1} \quad (6c)$$

$$\underline{x} \leq x_i \leq \bar{x}, i \in \mathbb{Z}_1^{N-1} \quad (6d)$$

$$\underline{u} \leq u_i \leq \bar{u}, i \in \mathbb{Z}_0^{N-1} \quad (6e)$$

$$\underline{x} + \varepsilon_x \leq x_N \leq \bar{x} - \varepsilon_x \quad (6f)$$

$$\underline{u} + \varepsilon_u \leq u_N \leq \bar{u} - \varepsilon_u \quad (6g)$$

$$x_s = Ax_s + Bu_s \quad (6h)$$

$$\tilde{x}_i + x_s - x_i = 0, i \in \mathbb{Z}_0^N \quad (6i)$$

$$\tilde{u}_i + u_s - u_i = 0, i \in \mathbb{Z}_0^N \quad (6j)$$

$$x_N = x_s \quad (6k)$$

$$u_N = u_s, \quad (6l)$$

where the decision variables are $\tilde{\mathbf{x}} = (\tilde{x}_0, \dots, \tilde{x}_N)$, $\tilde{\mathbf{u}} = (\tilde{u}_0, \dots, \tilde{u}_N)$, $\mathbf{x} = (x_0, \dots, x_N)$, $\mathbf{u} = (u_0, \dots, u_N)$, x_s and u_s . Equality constraints (6i) and (6j) impose the congruence of the decision variables with the original problem. We note that inequalities (3g) and (3h) are omitted

because they are already imposed by (6f) and (6g) alongside the inclusion of (6k) and (6l).

Note that the summations in the cost function (6a) now include $i = N$. However, this does not change the solution of the optimization problem due to the inclusion of (6k) and (6l).

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

$$z_1 = (x_0, u_0, x_1, u_1, \dots, x_{N-1}, u_{N-1}, x_N, u_N), \quad (7a)$$

$$z_2 = (x_s, u_s), \quad (7b)$$

$$z_3 = (\tilde{x}_0, \tilde{u}_0, \tilde{x}_1, \tilde{u}_1, \dots, \tilde{x}_{N-1}, \tilde{u}_{N-1}, \tilde{x}_N, \tilde{u}_N), \quad (7c)$$

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

$$\theta_2(z_2) = \frac{1}{2} z_2^\top \text{diag}(T, S) z_2 - (T x_r, S u_r)^\top z_2,$$

$$\theta_3(z_3) = \frac{1}{2} z_3^\top \text{diag}(Q, R, Q, R, \dots, Q, R) z_3,$$

$$C_1 = \begin{bmatrix} [I_n \ 0_{n,m}] & 0 & 0 \\ -I_{n+m} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & -I_{n+m} \\ 0 & 0 & -I_{n+m} \end{bmatrix}, \quad C_2 = \begin{bmatrix} 0 \\ I_{n+m} \\ \vdots \\ I_{n+m} \\ I_{n+m} \end{bmatrix},$$

$$C_3 = \begin{bmatrix} 0 & \dots & 0 \\ I_{n+m} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & I_{n+m} \\ 0 & \dots & 0 \end{bmatrix}, \quad b = \begin{bmatrix} x \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}.$$

Matrices C_1 , C_2 and C_3 contain the equality constraints (6b), (6i), (6j), (6k) and (6l). Specifically, the first n rows impose constraint (6b), the last $n + m$ rows impose the constraints (6k) and (6l), and the rest of the rows impose the constraints (6i) and (6j). Set \mathcal{Z}_1 is the set of vectors z_1 (7a) for which the box constraints (6d)-(6g) are satisfied; set \mathcal{Z}_2 is the set of vectors z_2 (7b) that satisfy the equality constraint (6h); and set \mathcal{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 $i \in \mathbb{Z}_1^3$ results in an optimization problem that satisfies Assumption 1. Therefore, under a proper selection of ρ , the iterates of the EADMM algorithm will converge to the optimal solution of the MPCT controller. In practice, the parameter ρ 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 $i \in \mathbb{Z}_1^3$ as shown in Section V-A, we can particularize Algorithm 1 to the MPCT problem, resulting in Algorithm 2. This algorithm requires solving three QP problems (which we label \mathcal{P}_1 , \mathcal{P}_2 and \mathcal{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 \tilde{z}_1^* (7a) returned by Algorithm 2.

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

$$\mathcal{P}_1(z_2^k, z_3^k, \lambda^k) : \min_{z_1} \frac{1}{2} z_1^\top H_1 z_1 + q_1^\top z_1 \quad (\mathcal{P}_1)$$

$$s.t. \ \underline{z}_1 \leq z_1 \leq \bar{z}_1,$$

where $H_1 = \rho C_1^\top C_1$,

$$q_1 = \rho C_1^\top C_2 z_2^k + \rho C_1^\top C_3 z_3^k + C_1^\top \lambda^k - \rho C_1^\top b,$$

$$\underline{z}_1 = (-M_n, \underline{u}, \underline{x}, \dots, \underline{u}, \underline{x} + \varepsilon_x, \underline{u} + \varepsilon_u),$$

$$\bar{z}_1 = (M_n, \bar{u}, \bar{x}, \dots, \bar{u}, \bar{x} - \varepsilon_x, \bar{u} - \varepsilon_u),$$

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 $j \in \mathbb{Z}_1^{(N+1)(n+m)}$ of the optimal solution of \mathcal{P}_1 , denoted by $(z_1^*)_j$, can be explicitly computed as,

$$(z_1^*)_j = \max \left\{ \min \left\{ \frac{-(q_1)_j}{(H_1)_{j,j}}, (\bar{z}_1)_j \right\}, (z_1)_j \right\}. \quad (8)$$

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 $\underline{y} \leq Cx + Du \leq \bar{y}$ 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,

$$\mathcal{P}_2(z_1^{k+1}, z_3^k, \lambda^k) : \min_{z_2} \frac{1}{2} z_2^\top H_2 z_2 + q_2^\top z_2 \quad (\mathcal{P}_2)$$

$$s.t. \ G_2 z_2 = b_2,$$

where $H_2 = \text{diag}(T, S) + \rho C_2^\top C_2$, $G_2 = [(A - I_n) \ B]$, $b_2 = 0$ and

$$q_2 = -(T x_r, S u_r) + \rho C_2^\top C_1 z_1^{k+1} + \rho C_2^\top C_3 z_3^k + C_2^\top \lambda^k - \rho C_2^\top b.$$

This problem has an explicit solution derived from the following proposition [40, §10.1.1].

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

$$Gz^* = b$$

$$Hz^* + q + G^\top \mu = 0,$$

which using simple algebra and defining $W_H \doteq GH^{-1}G^\top$, leads to

$$W_H \mu = -(GH^{-1}q + b) \quad (9a)$$

$$z^* = -H^{-1}(G^\top \mu + q). \quad (9b)$$

The optimal solution z_2^* of problem (\mathcal{P}_2) can be obtained by substituting (9a) into (9b), which leads to the expression

$$z_2^* = M_2 q_2, \quad (10)$$

where $M_2 = H_2^{-1}G_2^\top(G_2H_2^{-1}G_2^\top)^{-1}G_2H_2^{-1} - H_2^{-1} \in \mathbb{R}^{(n+m) \times (n+m)}$. 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,

$$\mathcal{P}_3(z_1^{k+1}, z_2^{k+1}, \lambda^k) : \min_{z_3} \frac{1}{2} z_3^\top H_3 z_3 + q_3^\top z_3 \quad (\mathcal{P}_3)$$

$$s.t. \quad G_3 z_3 = b_3,$$

where $H_3 = \text{diag}(Q, R, Q, R, \dots, Q, R) + \rho C_3^\top C_3$, $b_3 = 0$,

$$q_3 = \rho C_3^\top C_1 z_1^{k+1} + \rho C_3^\top C_2 z_2^{k+1} + C_3^\top \lambda^k - \rho C_3^\top b,$$

$$G_3 = \begin{bmatrix} A & B & -I_n & 0 & \dots & \dots & 0 & 0 \\ 0 & 0 & A & B & -I_n & \dots & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & A & B & -I_n & 0 \end{bmatrix}.$$

This problem can be sparsely solved using the following approach. Let $W_{H_3} \doteq G_3 H_3^{-1} G_3^\top$. 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 $W_{H_3} = W_{H_3,c}^\top W_{H_3,c}$, has the following structure,

$$W_{H_3,c} = \begin{pmatrix} \beta^1 & \alpha^1 & \dots & \dots & 0 & 0 \\ \dots & \beta^2 & \alpha^2 & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \beta^{N-1} & \alpha^{N-1} \\ 0 & 0 & \dots & \dots & \dots & \beta^N \end{pmatrix}, \quad (11)$$

where we define the sets of matrices $\mathcal{A} = \{\alpha^1, \dots, \alpha^{N-1}\}$, $\alpha^i \in \mathbb{R}^{n \times n}$; and $\mathcal{B} = \{\beta^1, \dots, \beta^N\}$, $\beta^i \in \mathbb{R}^{n \times n}$. Note that the amount of memory required to store the sets of matrices \mathcal{A} and \mathcal{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 $\hat{\mu}$,

$$W_{H_3,c}^\top \hat{\mu} = -G_3 H_3^{-1} q_3 \quad (12a)$$

$$W_{H_3,c} \mu = \hat{\mu}, \quad (12b)$$

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^{-1} [41, §6]. Finally, the optimal solution z_3^* of \mathcal{P}_3 can be obtained as in (9b),

$$z_3^* = -H_3^{-1}(G_3^\top \mu + q_3), \quad (13)$$

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 $G_2 H_2^{-1} G_2^\top$ and $G_3 H_3^{-1} G_3^\top$ 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 ρ for different constraints [6, §5.2], i.e., by considering ρ as a

Algorithm 2: Extended ADMM for MPCT

Require : $z_2^0, z_3^0, \lambda^0, \rho > 0, \epsilon > 0$

- 1 $k \leftarrow 0$
 - 2 **repeat**
 - 3 Obtain z_1^{k+1} by solving $\mathcal{P}_1(z_2^k, z_3^k, \lambda^k)$ using (8)
 - 4 Obtain z_2^{k+1} by solving $\mathcal{P}_2(z_1^{k+1}, z_3^k, \lambda^k)$ using (10)
 - 5 Obtain z_3^{k+1} by solving $\mathcal{P}_3(z_1^{k+1}, z_2^{k+1}, \lambda^k)$ 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 $k \leftarrow k + 1$
 - 9 **until** $\|\Gamma\|_\infty \leq \epsilon, \|z_2^k - z_2^{k-1}\|_\infty \leq \epsilon, \|z_3^k - z_3^{k-1}\|_\infty \leq \epsilon$
- Output:** $\tilde{z}_1^* \leftarrow z_1^k, \tilde{z}_2^* \leftarrow z_2^k, \tilde{z}_3^* \leftarrow z_3^k, \tilde{\lambda}^* \leftarrow \lambda^k$
-

diagonal positive definite matrix. In particular, we find that, for our problem, the convergence improves significantly if the equality constraints (6b), (6k), (6l), (6j) for $i = N$, and (6i) for $i = 0$ and $i = N$, are penalized more than the others. As in many operator splitting methods, the selection of ρ is a mostly open problem which has a significant impact on the performance of the algorithm [42, §4]. We find, however, that penalizing the above constraints 10 times more than the others is a good starting point.

Remark 5. The theoretical upper bound for ρ provided in Theorem 1 is easily computable in this case. Indeed, we have that $C_3^\top C_3$ is the identity matrix, and therefore its spectral norm $\|C_3^\top C_3\| = 1$. Furthermore, μ_3 is the minimum eigenvalue of $\text{diag}(Q, R)$, which is simple to compute.

Remark 6. The computation of q_1, q_2, q_3, z_3^* (13) and the right-hand-side of (12a) are not performed using the matrix multiplications shown in their expressions. Instead, the particular structure of the matrices allows for a matrix-free computation of the vectors. This is also true for the forward and backward substitutions performed to solve (12). That is, we do not store all the non-zero elements of the matrices along with arrays indicating their position. Only ρ and the matrices $\alpha^i, \beta^i, (\rho Q)^{-1}, (\rho R)^{-1}, T, S, A$ and B are stored.

Remark 7. We note that the number of iterations of Algorithm 2 can be reduced by the application of a warmstart procedure. This paper omits discussion on this topic because it focuses on the algorithm itself. See [41, §7] for a warmstart procedure that follows the prediction-correction warmstart from [43].

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 $u = (u_{(1)}, u_{(2)}) \in \mathbb{R}^2$ is the angle acceleration of each of the axes of the plate, which we constrain to $|u_{(i)}| \leq 0.2$, $i \in \mathbb{Z}_1^2$. We start the system with the ball placed in the central point 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 $z_2^0 = 0$, $z_3^0 = 0$, $\lambda^0 = 0$, 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) [6]. Problem (3) can be directly cast as a QP problem by taking the decision variables as

$$z = (x_0, u_0, x_1, u_1, \dots, x_{N-1}, u_{N-1}, x_s, u_s).$$

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 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 $\hat{x}_i = (x_i, x_s)$ 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 (3) other than the ADMM-based approaches discussed here, including interior point methods [46], active set methods [47], and even other non-ADMM Lagrangian methods [48]. 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 $N = 30$,

$$\begin{aligned} Q &= \text{diag}(10, 0.05, 0.05, 0.05, 10, 0.05, 0.05, 0.05), \\ R &= \text{diag}(0.5, 0.5), \quad S = \text{diag}(0.3, 0.3), \\ T &= \text{diag}(600, 50, 50, 50, 600, 50, 50, 50). \end{aligned}$$

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.

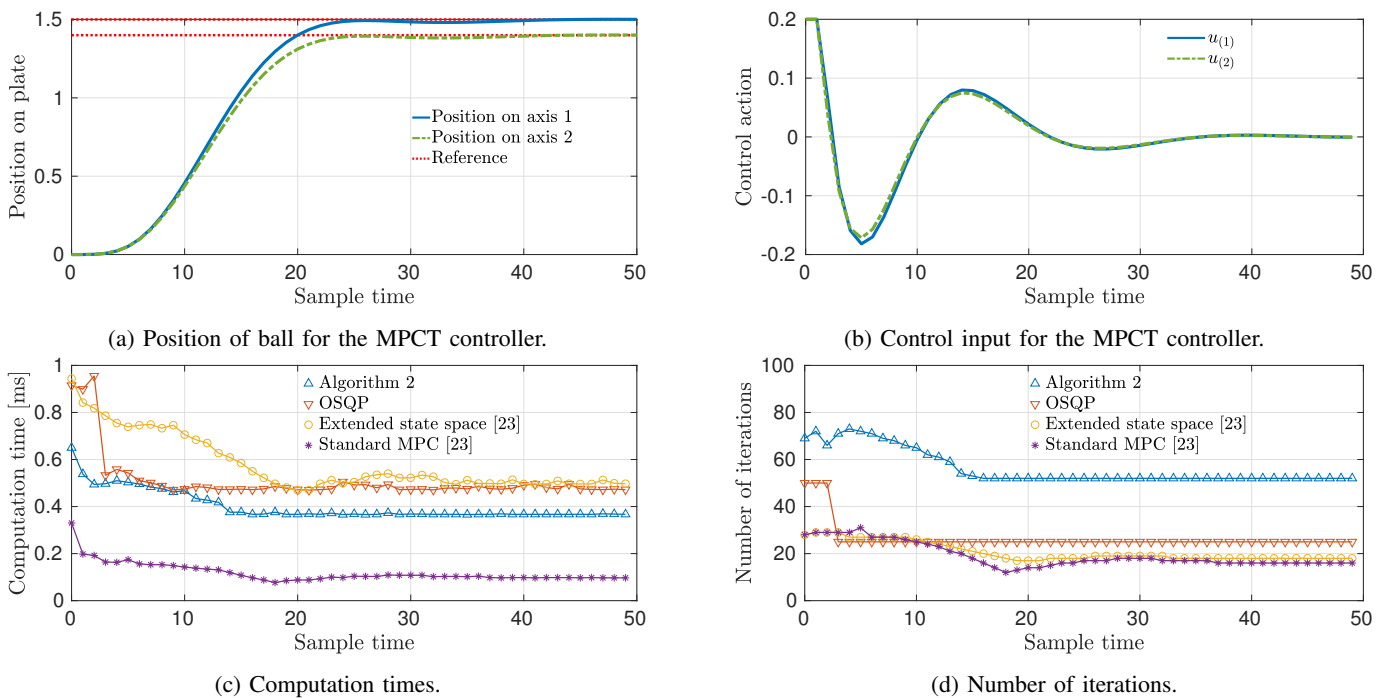


Fig. 1: Ball and plate system closed-loop tests.

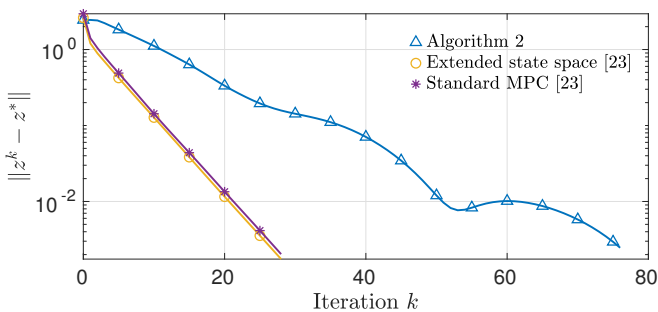


Fig. 2: Convergence of the SPCIES solvers.

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 z_1^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-Palomo 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.

- [6] 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.
- [7] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.
- [8] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.
- [9] F. Ullmann, "FiOrdOs: A Matlab toolbox for C-code generation for first order methods," *MS thesis*, 2011.
- [10] A. Domahidi, A. U. Zraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 668–674.
- [11] B. Huyck, L. Callebaut, F. Logist, H. J. Ferreau, M. Diehl, J. De Brabanter, J. Van Impe, and B. De Moor, "Implementation and experimental validation of classic MPC on programmable logic controllers," in *2012 20th Mediterranean Conference on Control & Automation (MED)*. IEEE, 2012, pp. 679–684.
- [12] B. Binder, D. K. M. Kufoalor, and T. A. Johansen, "Scalability of QP solvers for embedded model predictive control applied to a subsea petroleum production system," in *2015 IEEE Conference on Control Applications (CCA)*. IEEE, 2015, pp. 1173–1178.
- [13] E. N. Hartley, J. L. Jerez, A. Suardi, J. M. Maciejowski, E. C. Kerrigan, and G. A. Constantinides, "Predictive control using an FPGA with application to aircraft control," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 3, pp. 1006–1017, 2014.
- [14] D. Kufoalor, S. Richter, L. Imsland, T. A. Johansen, M. Morari, and G. O. Eikrem, "Embedded model predictive control on a PLC using a primal-dual first-order method for a subsea separation process," in *22nd Mediterranean Conference on Control and Automation*. IEEE, 2014, pp. 368–373.
- [15] R. Quirynen and S. Di Cairano, "PRESAS: Block-structured preconditioning of iterative solvers within a primal active-set method for fast model predictive control," *Optimal Control Applications and Methods*, vol. 41, no. 6, pp. 2282–2307, 2020.
- [16] G. Torrisi, D. Frick, T. Robbiani, S. Grammatico, R. S. Smith, and M. Morari, "FalcOpt: First-order Algorithm via Linearization of Constraints for OPTimization," <https://github.com/torrisig/FalcOpt>, May 2017.
- [17] G. Frison, H. H. B. Sørensen, B. Dammann, and J. B. Jørgensen, "High-performance small-scale solvers for linear model predictive control," in *2014 European Control Conference (ECC)*. IEEE, 2014, pp. 128–133.
- [18] P. Zometa, M. Kögel, and R. Findeisen, " μ AO-MPC: a free code generation tool for embedded real-time linear model predictive control," in *American Control Conference (ACC), 2013*. IEEE, 2013, pp. 5320–5325.
- [19] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari, "Embedded online optimization for model predictive control at megahertz rates," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3238–3251, 2014.
- [20] J. R. Sabo and A. A. Adegbege, "A primal-dual architecture for embedded implementation of linear model predictive control," in *Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 1827–1832.
- [21] H. A. Shukla, B. Khusainov, E. C. Kerrigan, and C. N. Jones, "Software and hardware code generation for predictive control using splitting methods," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 14386–14391, 2017.
- [22] J. Currie, A. Prince-Pike, and D. I. Wilson, "Auto-code generation for fast embedded model predictive controllers," in *Mechatronics and Machine Vision in Practice (M2VIP), 2012 19th International Conference*. IEEE, 2012, pp. 116–122.
- [23] 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.
- [24] —, "Implementation of model predictive controllers in programmable logic controllers using IEC 61131-3 standard," in *European Control Conference (ECC)*. IEEE, 2018, pp. 288–293.
- [25] S. Lucia, D. Navarro, Ó. Lucía, P. Zometa, and R. Findeisen, "Optimized FPGA implementation of model predictive control for embedded systems using high-level synthesis tool," *IEEE transactions on industrial informatics*, vol. 14, no. 1, pp. 137–145, 2018.
- [26] M. Pereira, D. Limon, D. M. de la Peña, and T. Alamo, "MPC implementation in a PLC based on Nesterov's fast gradient method," in *2015 23rd Mediterranean Conference on Control and Automation (MED)*. IEEE, 2015, pp. 371–376.
- [27] I. Necoara and D. Clipici, "Efficient parallel coordinate descent algorithm for convex optimization problems with separable constraints: Application to distributed MPC," *Journal of Process Control*, vol. 23, no. 3, pp. 243–253, 2013.
- [28] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.
- [29] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," vol. 3, no. 1, pp. 1–122, 2011.
- [30] 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.
- [31] 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.
- [32] X. Cai, D. Han, and X. Yuan, "On the convergence of the direct extension of ADMM for three-block separable convex minimization models with one strongly convex function," *Computational Optimization and Applications*, vol. 66, no. 1, pp. 39–73, 2017.
- [33] P. Krupa, J. Camara, I. Alvarado, D. Limon, and T. Alamo, "Real-time implementation of MPC for tracking in embedded systems: Application to a two-wheeled inverted pendulum," in *European Control Conference (ECC), 2021*, pp. 669–674.
- [34] P. Krupa, D. Limon, and T. Alamo, "SPCIES: Suite of Predictive Controllers for Industrial Embedded Systems," <https://github.com/GepocUS/Spcies>, Dec 2020.
- [35] C. Chen, B. He, Y. Ye, and X. Yuan, "The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent," *Mathematical Programming*, vol. 155, no. 1–2, pp. 57–79, 2016.
- [36] C. Chen, Y. Shen, and Y. You, "On the convergence analysis of the alternating direction method of multipliers with three blocks," in *Abstract and Applied Analysis*, vol. 2013, 2013.
- [37] T.-Y. Lin, S.-Q. Ma, and S.-Z. Zhang, "On the sublinear convergence rate of multi-block ADMM," *Journal of the Operations Research Society of China*, vol. 3, no. 3, pp. 251–274, 2015.
- [38] B. He, M. Tao, and X. Yuan, "Alternating direction method with Gaussian back substitution for separable convex programming," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 313–340, 2012.
- [39] M. Li, D. Sun, and K.-C. Toh, "A convergent 3-block semi-proximal ADMM for convex minimization problems with one strongly convex block," *Asia-Pacific Journal of Operational Research*, vol. 32, no. 04, p. 1550024, 2015.
- [40] S. Boyd, *Convex Optimization*, 7th ed. Cambridge University Press, 2009.
- [41] 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," *arXiv:2008.09071v2*, 2020.
- [42] G. Stathopoulos, H. A. Shukla, A. Szuets, Y. Pu, and C. N. Jones, "Operator splitting methods in control," *Foundations and Trends in Systems and Control*, vol. 3, no. 3, pp. 249–362, 2016.
- [43] S. Paternain, M. Morari, and A. Ribeiro, "Real-time model predictive control based on prediction-correction algorithms," in *58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 5285–5291.
- [44] P. Krupa, D. Limon, and T. Alamo, "Harmonic based model predictive control for set-point tracking," *IEEE Transactions on Automatic Control*, 2020, doi:10.1109/TAC.2020.3047579.
- [45] Y. Wang, M. Sun, Z. Wang, Z. Liu, and Z. Chen, "A novel disturbance-observer based friction compensation scheme for ball and plate system," *ISA transactions*, vol. 53, no. 2, pp. 671–678, 2014.
- [46] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *Journal of optimization theory and applications*, vol. 99, no. 3, pp. 723–757, 1998.
- [47] 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.
- [48] N.-Y. Chiang, R. Huang, and V. M. Zavala, "An augmented lagrangian filter method for real-time embedded optimization," *IEEE Transactions on Automatic Control*, vol. 62, no. 12, pp. 6110–6121, 2017.
- [49] T. Alamo, P. Krupa, and D. Limon, "Gradient based restart FISTA," in *Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 3936–3941.
- [50] P. Krupa, "Implementation of MPC in embedded systems using first order methods," Ph.D. dissertation, University of Seville, 2021, available at [arXiv:2109.02140](https://arxiv.org/abs/2109.02140).