

Real-time implementation of MPC for tracking in embedded systems: Application to a two-wheeled inverted pendulum

Pablo Krupa, Jose Camara, Ignacio Alvarado, Daniel Limon, Teodoro Alamo

Abstract—This article presents the real-time implementation of the model predictive control for tracking formulation to control a two-wheeled inverted pendulum robot. This formulation offers several advantages over standard MPC formulations at the expense of the addition of a small number of decision variables, which complicates the inner structure of the matrices of the optimization problem. We implement a sparse solver, based on an extension of the alternating direction method of multipliers, in the system’s embedded hardware. The results indicate that the solver is suitable for controlling a real system with sample times in the range of milliseconds using current, readily-available hardware.

Keywords: Model predictive control, embedded system, extended ADMM

I. INTRODUCTION

The implementation of model predictive control (MPC) in embedded systems has been a widely researched topic in recent years due to the interest of being able to use this advanced control strategy to control real systems using the currently available embedded hardware. One of the main challenges that needs to be overcome is the fact that MPC requires solving an optimization problem at each sample time, which can become an issue in systems with fast dynamics, especially when considering the low computational and memory resources typically available in embedded systems.

Recently, significant advances have been made in this field thanks to the development of optimization algorithms suitable for their implementation in embedded systems. Some examples of these tools being used to implement MPC in embedded systems include [1], [2], [3]. Additionally, other authors propose algorithms that are particularly tailored to the MPC optimization problem, such as in [4], [5], [6], [7]. Finally, another approach is to use *explicit* MPC [8], which computes the solution of the parametric MPC optimization problem offline and stores it online as a lookup table. However, this is only suitable for systems with few states and a moderate number of constraints.

A common theme among the current research on this topic, which is shared by the previously cited papers, is that they typically only consider standard MPC formulations. This paper, on the other hand, presents an implementation of a non-standard MPC formulation called MPC for tracking (MPCT) [9], which adds an *artificial reference* as an additional decision variable of the optimization problem. This

formulation provides a series of advantages that make its implementation in embedded systems particularly interesting.

Firstly, a common issue of standard MPC formulations with stability guarantees is that the domain of attraction of the controller can become insufficient if the prediction horizon is chosen too small. However, the use of small prediction horizons is desirable in order to help overcome the computational and memory limitations typically imposed by embedded systems. The MPCT formulation provides significantly larger domains of attraction than standard MPC formulations [10], especially for small prediction horizons.

Secondly, it intrinsically deals with references that are not attainable (i.e. that are not a steady state of the system or that violate the system constraints) [10]. In this case, it will steer the system to the “closest” attainable steady state to the reference, where the “closeness” is determined by the selection of its cost function matrices. Additionally, it also guarantees recursive feasibility of the closed-loop system even in the event of a sudden reference change [9].

However, these advantages come at the cost of the addition of the *artificial reference* as new decision variables, which complicates the inner structure of the matrices of the quadratic programming problem when compared to the standard MPC formulations.

In [11] the authors presented a sparse solver for the MPCT formulation based on an extension of the classical alternating direction method of multipliers (ADMM) [12] to problems with three separable functions in the objective function [13]. The use of this method resulted in the ingredients of the algorithm having simple structures that could be exploited using a similar approach to the one used in [4], which presented sparse solvers for standard MPC formulations. This led to a sparse solver with a small iteration complexity and a small memory footprint that was included in the Spcies toolbox [14] for Matlab, which is available at <https://github.com/GepocUS/Spcies>.

This paper presents the implementation of the above MPCT solver in a Raspberry Pi to control a two-wheeled inverted pendulum robot with a sample time of 20 milliseconds. The closed-loop results suggest that the solver is suitable for its implementation in current, readily-available hardware for controlling systems with fast dynamics.

This paper is organized as follows. Section II provides the problem formulation. The MPCT formulation is described in Section III. For completeness, a brief description of the solver is presented in Section IV. The two-wheeled inverted pendulum robot and the closed-loop results are shown in Section V. Finally, conclusions are provided in Section VI.

Systems Engineering and Automation department, University of Seville, Spain. The corresponding author is Pablo Krupa (pkrupa@us.es).

This work was supported in part by the Agencia Estatal de Investigación (AEI) under Grant PID2019-106212RB-C41/AEI/10.13039/501100011033, by MINERCO-Spain and FEDER funds under Grant DPI2016-76493-C3-1-R and by the MCIU-Spain and FSE under Grant FPI-2017.

Notation: Given two integers i and j with $j \geq i$, \mathbb{Z}_i^j denotes the set of integer numbers from i to j , i.e. $\mathbb{Z}_i^j \doteq \{i, i+1, \dots, j-1, j\}$. Given two vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$, $x \leq (\geq) y$ denotes componentwise inequalities; and $\langle x, y \rangle$ denotes their standard inner product. For a vector $x \in \mathbb{R}^n$ and a positive definite matrix $A \in \mathbb{R}^{n \times n}$, $\|x\| \doteq \sqrt{\langle x, x \rangle}$, $\|x\|_A$ is its weighted Euclidean norm $\|x\|_A \doteq \sqrt{\langle x, Ax \rangle}$, 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. We denote by (x_1, x_2, \dots, x_N) the column vector formed by the concatenation of column vectors x_1 to x_N . Given scalars and/or matrices M_1, M_2, \dots, M_N , 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 .

II. PROBLEM FORMULATION

We consider a system described by a 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, that are subject to the box constraints

$$\underline{x} \leq x_k \leq \bar{x}, \quad \underline{u} \leq u_k \leq \bar{u}. \quad (2)$$

The control objective is to steer the system to the reference (x_r, u_r) given by the user. This will only be possible if the reference is a steady state of the system that satisfies the constraints (2), i.e., if it is an *admissible* steady state. Otherwise, we wish to steer the system to the closest admissible steady state to (x_r, u_r) , for a certain criterion of closeness.

III. MODEL PREDICTIVE CONTROL FOR TRACKING

This section describes the particular MPC formulation that we consider in this paper, which is called *MPC for tracking* [9]. For a given control horizon N , a current state $x \in \mathbb{R}^n$ and a reference (x_r, u_r) , the control law of the MPCT controller is derived from the solution of the optimization problem

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

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

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

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

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

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

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

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

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

where $\mathbf{x} = (x_0, x_1, \dots, x_{N-1})$ and $\mathbf{u} = (u_0, u_1, \dots, u_{N-1})$ are the predicted states and control actions throughout the prediction horizon, respectively; (x_s, u_s) is the artificial

reference; $\varepsilon_x \in \mathbb{R}^n$ and $\varepsilon_u \in \mathbb{R}^m$ are vectors with arbitrarily small positive components which are added to avoid a (possible) loss of controllability in the event of active constraints at the equilibrium point [10]; and the positive definite matrices Q , R , T and S are the cost function matrices.

The main difference between MPCT and standard MPC formulations is the introduction of the artificial reference (x_s, u_s) as additional decision variables. As can be seen in (3), the discrepancy between the predicted states and control actions with the artificial reference is penalized with matrices Q and R . Additionally, the discrepancy between the artificial reference and the reference (x_r, u_r) given by the user is penalized with matrices T and S .

IV. EMBEDDED SOLVER FOR MPCT

This section briefly presents the sparse solver implemented in the embedded system, which is a particularization of the *extended ADMM* algorithm [13] to the optimization problem (3). This solver, which was originally presented in [11] and is available at [14], exploits the structure of the problem to attain a very small memory footprint and an efficient implementation. Due to space considerations, and to not repeat the results presented in [11], only a very brief outline of the solver is presented here. We refer the reader to the above reference for an in-depth explanation.

A. Extended ADMM

The extended ADMM algorithm is, as its name suggests, an extension of the classical ADMM algorithm [12] to problems with more than two separable functions in the objective function. Specifically, we show its application to objective functions that are the sum of three separable functions [13].

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

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

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

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

where $z_i \in \mathbb{R}^{n_i}$ for $i \in \mathbb{Z}_1^3$ are the decision variables, and let its augmented Lagrangian $\mathcal{L}_\rho(z_1, z_2, z_3, \lambda)$ be given by

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

where $\lambda \in \mathbb{R}^{m_z}$ are the dual variables and the scalar $\rho > 0$ is the penalty parameter.

Algorithm 1 shows the implementation of the extended ADMM algorithm. It returns a suboptimal solution $(\tilde{z}_1^*, \tilde{z}_2^*, \tilde{z}_3^*)$ of problem (4) (assuming a solution point exists) as well as a suboptimal dual variable $\tilde{\lambda}^*$, where the suboptimality is determined by the exit tolerance $\epsilon > 0$, since the exit conditions of step 9 serve as a measure of the

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: $\tilde{z}_1^* \leftarrow z_1^k, \tilde{z}_2^* \leftarrow z_2^k, \tilde{z}_3^* \leftarrow z_3^k, \lambda^* \leftarrow \lambda^k$

optimality of the current iterate [13, §5]. The superscript k is used to indicate the value of the variable at iteration k of the algorithm.

The extended ADMM does not necessarily converge under the same assumptions as standard ADMM, as shown in [15]. In order to prove its convergence, additional conditions are required. In particular, in [13, Theorem 3.1] it was shown that the extended ADMM algorithm applied to (4) converges under the following assumption if ρ is chosen appropriately (as stated in the cited theorem).

Assumption 1 ([13], Assumption 3.1). *The functions θ_1 and θ_2 are convex; function θ_3 is strongly convex; and C_1 and C_2 are full column rank.*

B. Solving MPCT using EADMM

This section explains how problem (3) can be recast into (4) by a proper selection of decision variables. We do so by defining variables $\tilde{x}_i \doteq x_i - x_s$ and $\tilde{u}_i \doteq u_i - u_s$, which lets us rewrite (3) as:

$$\min_{\substack{\tilde{x}, \tilde{u}, x_s \\ \mathbf{u}, x_s, u_s}} \sum_{i=0}^N \|\tilde{x}_i\|_Q^2 + \|\tilde{u}_i\|_R^2 + \|x_s - x_r\|_T^2 + \|u_s - u_r\|_S^2 \quad (5a)$$

$$s.t. \ x_0 = x \quad (5b)$$

$$\tilde{x}_{i+1} = A\tilde{x}_i + B\tilde{u}_i, \ i \in \mathbb{Z}_0^{N-1} \quad (5c)$$

$$\underline{x} \leq x_i \leq \bar{x}, \ i \in \mathbb{Z}_1^{N-1} \quad (5d)$$

$$\underline{u} \leq u_i \leq \bar{u}, \ i \in \mathbb{Z}_0^{N-1} \quad (5e)$$

$$\underline{x} + \varepsilon_x \leq x_N \leq \bar{x} - \varepsilon_x \quad (5f)$$

$$\underline{u} + \varepsilon_u \leq u_N \leq \bar{u} - \varepsilon_u \quad (5g)$$

$$x_s = Ax_s + Bu_s \quad (5h)$$

$$\tilde{x}_i + x_s - x_i = 0, \ i \in \mathbb{Z}_0^N \quad (5i)$$

$$\tilde{u}_i + u_s - u_i = 0, \ i \in \mathbb{Z}_0^N \quad (5j)$$

$$x_N = x_s \quad (5k)$$

$$u_N = u_s, \quad (5l)$$

where we are introducing the new decision variables $\tilde{\mathbf{x}} = (\tilde{x}_0, \dots, \tilde{x}_N)$ and $\tilde{\mathbf{u}} = (\tilde{u}_0, \dots, \tilde{u}_N)$. Note that we have

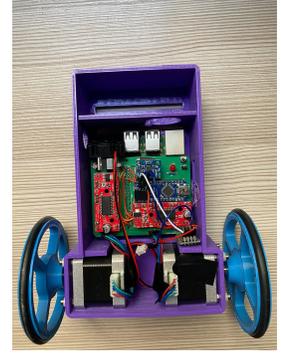
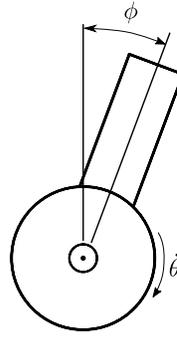


Fig. 1: Two-wheeled inverted pendulum robot. The angle ϕ is zero if the pendulum is in a vertical position.

extended the summations in the cost function (and some constraints) to $i = N$. This is necessary to be able to construct matrices C_i of (4) with a simple structure. However, note that this additional term does not affect the optimization problem due to constraints (5k) and (5l).

Problem (5) can then be recast as (4) by defining

$$z_1 = (x_0, u_0, x_1, u_1, \dots, x_{N-1}, u_{N-1}, x_N, u_N), \quad (6a)$$

$$z_2 = (x_s, u_s), \quad (6b)$$

$$z_3 = (\tilde{x}_0, \tilde{u}_0, \tilde{x}_1, \tilde{u}_1, \dots, \tilde{x}_{N-1}, \tilde{u}_{N-1}, \tilde{x}_N, \tilde{u}_N), \quad (6c)$$

and casting the constraints (5b), (5i), (5j), (5k) and (5l) into C_i and b ; constraints (5d)-(5g) into \mathcal{Z}_1 , (5h) into \mathcal{Z}_2 and (5c) into \mathcal{Z}_3 . Then, the functions θ_i are given by $\theta_1(z_1) = 0$

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

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

We note that our selection of z_i and C_i for $i \in \mathbb{Z}_1^3$ leads to a problem (4) that satisfies Assumption 1.

The selection of the decision variables z_1 , z_2 and z_3 as shown in (6) leads to matrices C_i with very simple structures (see [11, Eq. (9)]) and to the optimization problems solved in steps 3, 4 and 5 of Algorithm 1 to have explicit and computationally efficient solutions [11, §5.2].

Remark 1 ([11], Remark 1). *In [16, §5.2] it was shown that the performance of ADMM can be improved if different values of ρ are used to penalize some constraints more than others, i.e., by taking ρ as a diagonal positive definite matrix. In particular, for problem (5), the convergence improves significantly if the equality constraints (5b), (5k), (5l), (5j) for $i = N$, (5i) for $i = 0$ and $i = N$, are penalized more than the others.*

V. CASE STUDY

A. Two-wheeled inverted pendulum robot

The two-wheeled inverted pendulum robot, which is shown in Figure 1, is a two-wheeled vehicle based on the inverted pendulum configuration. The control objective is to control the horizontal speed of the vehicle whilst keeping it from toppling. Due to construction limitations of the robot, in

this paper we only consider forward and backward velocities, i.e., both wheels have the same speed, making the robot incapable of rotating sideways.

The specifics of the robot, including its construction and components, are very similar to the description provided in [17]. The chassis is made using a 3D printer, and its main components are: a Raspberry Pi, an inertial measuring unit MPU6050, an Arduino NANO, a microstepping motor driver A3967 and two step motors Nema 17. The main difference between this robot and the one described in [17] is the inclusion of the Raspberry Pi for monitoring and controlling the system, as we explain in further detail in Section V-B.

The non-linear dynamics of the systems, which are obtained by applying Lagrange's equation as in [18, Appendix A], are given by the ordinary differential equation [17, §6]

$$(2a + c \cos(\phi + \phi_0))\ddot{\theta} + (c \cos(\phi + \phi_0) + 2b)\dot{\phi} - c\dot{\phi}^2 \sin(\phi + \phi_0) - d \sin(\phi + \phi_0) = 0, \quad (7)$$

where ϕ is the tilt of the robot, θ is the angle of the wheels, $a = (3/2m_r + 1/2)R^2$, $b = ML^2$, $c = RML$, $d = MgL$, m_r is the mass of the wheels, M is the mass of the robot without the wheels, R is the wheel's radius, L is the distance between the rotation axis of the wheels and the center of mass, $g = 9.81\text{m/s}^2$ is the gravitational acceleration and ϕ_0 is the angle between the center of mass and the geometrical center. The robot used in this case study, which is shown in the right-hand-side of Figure 1, has the following values of the parameters: $m_r = 0.064\text{Kg}$, $M = 0.975\text{Kg}$, $R = 0.05\text{m}$, $L = 0.05\text{m}$ and ϕ_0 is unmeasured but known to be small.

The state of the system is given by $x = (\phi, \dot{\phi}, \dot{\theta})$ and the control input is the angular acceleration of the wheels $u = \ddot{\theta}$. We consider the following constraints on the state and control input,

$$\begin{bmatrix} -\frac{90}{360}2\pi \\ -4 \\ -60 \end{bmatrix} \leq \begin{bmatrix} \phi \\ \dot{\phi} \\ \dot{\theta} \end{bmatrix} \leq \begin{bmatrix} \frac{90}{360}2\pi \\ 4 \\ 60 \end{bmatrix}, \quad -80 \leq \ddot{\theta} \leq 80,$$

where the units are given in radians and seconds, accordingly.

B. Embedded system: Raspberry Pi

The Raspberry Pi is a low-cost embedded system that can be operated by a Linux-based operating system. In particular, we use the Raspbian operating system provided by the manufacturer, which is based on the Debian distribution. The model used in this case study is Raspberry Pi 3 Model B, which comes with a Quad Core 1.2GHz Broadcom BCM2837 64bit CPU.

The Raspberry Pi is used as the monitoring and control device of the robot. It receives the measurements of the tilt angle ϕ and the tilt angular speed $\dot{\phi}$ from the inertial measuring unit MPU6050, and the measurement of the speed of the wheels $\dot{\theta}$ from the Arduino NANO board. The control action is sent to the Arduino NANO board, which is in charge of applying the corresponding PWM signals to the step motors.

In order to ensure the real-time operation of the control system, we employ the Xenomai software, which is

TABLE I: Performance of EADMM

	Iterations				Computation time (ms)			
	Max.	Min.	Med.	Avg.	Max.	Min.	Med.	Avg.
Fig. 2	44	1	15	15.12	8.64	0.195	2.99	2.99
Fig. 3	38	1	11	12.38	7.39	0.196	2.15	2.42

a dual-kernel configuration for Linux-based systems which considers the Linux kernel as an idle task, and that will ensure the accomplishment of the scheduled tasks within the given deadlines. In short, it provides a real-time framework to Linux-based systems, which we use to schedule the measurement and MPCT controller routines in real-time.

C. Closed-loop results

We design the MPCT controller, taking the parameters $N = 12$, $Q = 5I_3$, $R = 1$, $T = 1000I_3$ and $S = 5$. The exit tolerance of the EADMM algorithm is $\epsilon = 0.001$, and the penalty parameter is $\rho = 1000$ for the constraints listed in Remark 1, and $\rho = 5$ for the rest. The prediction model of the MPCT controller is obtained by linearizing the non-linear model (7) around the operating point $x^\circ = (0, 0, 0)$ and $u^\circ = (0)$, i.e., the stationary vertical position, for a sample time of 20ms.

The solver is obtained from the Matlab toolbox [14], which automatically generates code for solving different MPC formulations, including the solver discussed here. The toolbox requires the state space model of the system (1), its constraints (2), the ingredients of the MPCT formulation (Q , R , T , S and N), and the parameters of the EADMM algorithm (ρ and ϵ). It then generates library-free plain C code containing the sparse solver for its direct implementation in the embedded system, which we compile using the *gcc* compiler in the Raspberry Pi.

To test the performance of the proposed MPCT solver we conduct two experiments on the real system: disturbance rejection and reference tracking.

Figure 2 shows the disturbance rejection results. In this experiment, the reference is set to the operating point, i.e., $x_r = x^\circ$ and $u_r = u^\circ$. The system is initially positioned at the reference and is then repeatedly perturbed by manually pushing it in either direction. Figures 2a and 2b show the tilt of the system ϕ (in degrees), and the control action $\ddot{\theta}$, respectively. Figures 2c and 2d show the number of iterations and computation time of the EADMM algorithm at each sample time, respectively. As can be seen, the MPCT controller steers the system back to the vertical position after each push. Additionally, the control action reaches its upper and lower bounds, which are marked in red lines, during the first moments after each disturbance is applied. Note that the number of iterations of the solver increases when the control action bounds are active, as expected when using first-order methods. However, the increase is not very significant.

Figure 3 shows the reference tracking results. In this experiment, the system is started at the operating point and then the reference for the wheel angular speed $\dot{\theta}$ is changed

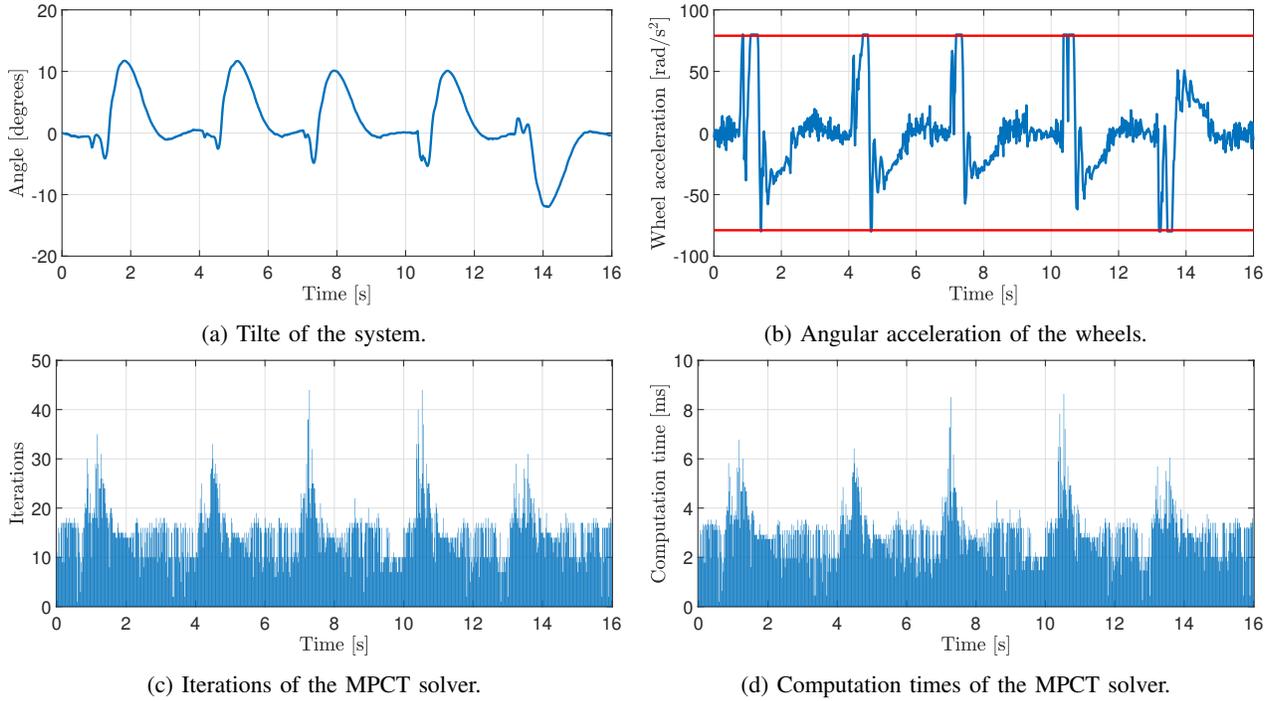


Fig. 2: Closed-loop results of the robot against external disturbances.

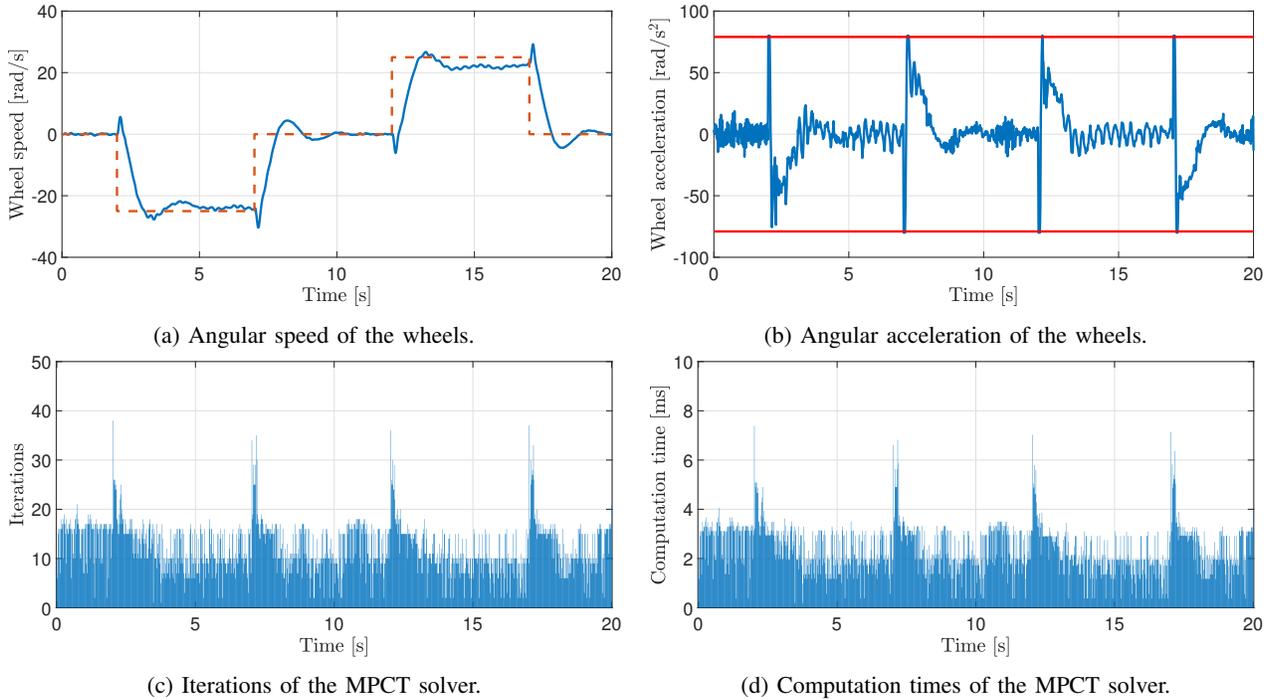


Fig. 3: Closed-loop results of the robot for changing reference of the angular speed.

in multiple occasions. Figures 3a and 3b show the speed of the wheels $\dot{\theta}$, and the control action $\ddot{\theta}$, respectively. Figures 3c and 3d show the number of iterations and the computation time of the EADMM algorithm at each sample time, respectively. As can be seen, the MPCT controller steers the system to the reference. A slight offset can be observed for references other than the operating point due to

the difference between the prediction model (1) and the real system. This offset could be corrected with the inclusion of a state and disturbance estimator [4]. Once again, the control action reaches its upper and lower bounds during the first moments after each reference change, without having a significant impact on the number of iterations.

Table I shows a detailed analysis of the number of itera-

tions and computation times of the algorithm during the two experiments. We show the maximum, minimum, median and average number of iterations and computation times.

VI. CONCLUSIONS

This paper presents the results of implementing the sparse solver for the MPC for tracking formulation presented in [11] in a Raspberry Pi to control an inverted pendulum robot with fast dynamics. The proposed solver is available in the Spcies toolbox [14] for Matlab at <https://github.com/GepocUS/Spcies>.

The results indicate that the solver, which is based on an extension of the ADMM algorithm, is suitable for its implementation in embedded systems to control systems with sample times in the order of milliseconds in real-time.

REFERENCES

- [1] 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.
- [2] 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.
- [3] P. Krupa, N. Saraf, D. Limon, and A. Bemporad, "PLC implementation of a real-time embedded MPC algorithm based on linear input/output models," in *21st IFAC World Congress*, 2020.
- [4] P. Krupa, D. Limon, and T. Alamo, "Implementation of model predictive control in programmable logic controllers," *IEEE Transactions on Control Systems Technology*, 2020.
- [5] 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.
- [6] 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. 14 386–14 391, 2017.
- [7] H. Park, J. Sun, and I. Kolmanovsky, "A tutorial overview of IPASQP approach for optimization of constrained nonlinear systems," in *Proceeding of the 11th World Congress on Intelligent Control and Automation*. IEEE, 2014, pp. 1735–1740.
- [8] 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.
- [9] 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.
- [10] 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.
- [11] 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 preprint: 2008.09071v2*, submitted to *Transactions on Control Systems Technology*, 2020.
- [12] 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.
- [13] 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.
- [14] P. Krupa, D. Limon, and T. Alamo, "Sppecies: Suite of Predictive Controllers for Industrial Embedded Systems," Available at <https://github.com/GepocUS/Sppecies>, Dec 2020.
- [15] 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.
- [16] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *arXiv preprint:1711.08013v4*, 2020.
- [17] J. A. Borja, I. Alvarado, and D. Muñoz de la Peña, "Low cost two-wheels self-balancing robot for control education powered by stepper motors," in *IFAC World Congress*, 2020.
- [18] C. González, I. Alvarado, and D. Muñoz La Peña, "Low cost two-wheels self-balancing robot for control education," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 9174–9179, 2017.