Embedded optimization-based controllers for industrial processes

Carlos Ibañez, Carlos Ocampo-Martinez, *Senior Member*, IEEE, Brais Gonzalez
Universitat Politècnica de Catalunya, Institut de Robòtica i Informàtica Industrial (CSIC-UPC),
Llorens i Artigas 4-6, Planta 2, 08028 Barcelona, Spain
Corresponding author: cibanez@iri.upc.edu

***Abstract*—Due to the growth of computational capabilities and the proliferation of field-programmable gate arrays (FPGA), the utilization of model predictive control (MPC) for embedded applications in the industry has become a possibility and a fact. This paper presents and discusses the possibilities of the use of online MPC, embedded in an educational device from National Instruments, using two different optimization algorithms and code generators, which have come out in recent years by the academia: CVXGEN, which implements a primal-dual interior-point algorithm, and qpOASES, which relies on the online active-set strategy algorithm. Both algorithms have been tested both in simulation and in real-time experimentation to control a four-tank pilot plant.**

I. Introduction

Numerous modern control approaches depend on solving an optimization problem that is updated with information of the system, which progresses in time, while dealing with real-time control constraints. In this context, real time (in the control context) refers to the fact that, to apply the control approach, it must be taken into account that there is a small time window to gather the data from the system, compute the control signal, and, finally, update the system. The key issue therefore is how to deal with this time window since, if not done with caution, the optimization problem embedded in the controller may be out of date in the face of the evolution of the real system if the controller is delayed, or perform unnecessary calculations if the controller is advanced. The suitability of a controller should not be evaluated just in a single optimization from the optimality sense, but should be evaluated according to the controller-system behavior over time [10].

One of the optimization-based control techniques that has captured the interest, both from academia and industry, has been model predictive control (MPC). This technique allows to deal with (i) multivariable systems, (ii) optimal inputs and (iii) system constraints [12]. This control approach usually involves the resolution of a Quadratic Programming (QP) problem. This type of problem has been extensively studied by the academia and, as a result, numerous resolution solvers have been presented and, thanks to the computational speed of the used algorithm, MPC can be embedded in either a microcontroller or an FPGA to solve online optimization problems managing to meet the real-time constraints. Lately, a lot of open-source solvers have been introduced by researchers, e.g., CVXGEN, FORCES, FiOrdOs, qpOASES. Moreover, some of these solvers are also code generators with the goal of providing optimized code to build embedded applications [11].

The MPC controller can be implemented in two different ways: the explicit and online way. Explicit MPC prevents the need of solving an optimization problem online to compute the control signal at each time instant, allowing to solve offline the optimization problem for a range of operating conditions of interest [1]. Usually, this MPC approach ends up by mapping the system dynamics into a table of linear gains. With this method, the most costly computational operation (the solution of the optimization problem) is done offline without the need for the embedded controller to solve online the optimization problem at each time instant.

On the other hand, online MPC should solve an optimization problem at each step, which requires an important computational power. To deal with this problem, the academia has focused on mathematical algorithms to solve QP problems in a reliable and fast way, e.g., [18][20][6].

Embedded optimization, used to design and implement real-time MPC controllers, is becoming a frequent approach in different industrial applications, e.g., robotics, automotive, aerospace. For example, in [3] an energy management approach using MPC for hybrid vehicles is proposed. In [7], a control of a step-down DC-DC converter using an hybrid model and solving a constrained optimal control is proposed. Besides microprocessors and FPGAs, programmable logic controllers (PLCs) have been studied for the implementation of optimization-based controllers given their ability to work in harsh industrial and environmental conditions, e.g., [8] implements an online linear MPC controller embedded in a PLC to control a pilot-scale distillation column.

The aim of this paper is to illustrate the practical implementation of an online MPC controller, embedded in a device composed of both an FPGA and a microprocessor, using two different QP solvers: CVXGEN (also a code generator) and qpOASES[1], which implement a dual interior-point and an online active set strategy, respectively, and perform a comparative assessment between the resolution algorithms about their influence over the closed-loop performance.

The remainder of this paper is structured as follows: in Section II, the background on MPC and the QP solvers used

[1]FiOrdOs [19] was also tested, but due to computational aspects no valid experimental results could be obtained. Refer to Section IV to get a closer look.

are briefly explained. In Section III, the problem statement studied along this document is presented and complemented with some simulation results. In Section IV, the real implementation is presented along with the used plant, the used device and the explanation of embedding the controller. Finally, the conclusions are drawn in Section V.

II. Preliminaries

A. Model Predictive Control

Model predictive control (MPC) relies on the idea of using a *dynamic model* of the plant intended to be controlled to predict the system behavior and optimize it to obtain the best possible decision towards reaching a control objective while satisfying system constraints [17]. The model is, therefore, a central key for this control technique, and the final performance of the controller relies on the model used since it can be deterministic or stochastic, linear or nonlinear, continuous or discrete or hybrid. The choice of one model or another resides on the computational possibilities and the accuracy level wanted. Normally, the models used in simulations are intended to be the most accurate as possible while the models used in control normally are not that accurate because of computational aspects.

To compute the best possible decision, a *cost function* is defined. This cost function is used to define the goals that the controller needs to achieve, e.g., tracking a set-point or minimizing the use of a resource. Therefore, the optimal input minimizes the cost function: at the current time k, an optimization problem is solved along a finite prediction horizon and a set of control signals is calculated, but just the first control input of the optimal sequence is applied (receding horizon idea) [12].

This paper is restricted to the study of the linear MPC (LMPC). Since linear control theory is going to be applied, from now on the system will be represented as a discrete-time state-space linear model of the form

$$x_{k+1} = Ax_k + Bu_k, \tag{1a}$$
$$y_k = Cx_k + Du_k, \tag{1b}$$
$$x_{k_0} = x_0, \tag{1c}$$

where $x \in \mathbb{R}^{n_x}$ are the states, $u \in \mathbb{R}^{n_u}$ are the inputs, $y \in \mathbb{R}^p$ are the outputs and $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$, $C \in \mathbb{R}^{p \times n_x}$, $D \in \mathbb{R}^{p \times n_u}$ are time-invariant matrices defining the dynamics of the system.

The optimal control problem (OCP) behind the MPC implemented in this paper has the form

$$\begin{aligned} \min_{u_{k|k},\ldots,u_{k+H_p-1|k}} \quad & \sum_{i=1}^{H_p} ||x_{k+i|k} - x_r||_Q + \sum_{i=0}^{H_c-1} ||\Delta u_{k+i|k}||_R \\ \text{s.t.} \quad & x_{k+i+1|k} = Ax_{k+i|k} + Bu_{k+i|k}, \\ & u_{k+i|k} \in \mathcal{U}, \\ & x_{k+i|k} \in \mathcal{X}, \end{aligned} \tag{2}$$

where $x_r \in \mathbb{R}^{n_x}$ is a constant desired set-point. Matrices $Q \in \mathbb{R}^{n_x \times n_x}$ and $R \in \mathbb{R}^{n_u \times n_u}$ are penalization weights assigning prioritization for the control objectives. Both states and inputs are subject to some physical and operating constraints defined as $\mathcal{X} \triangleq \{x \in \mathbb{R}^{n_x} : \underline{x} \leq x_k \leq \bar{x},\ \forall k\}$ and $\mathcal{U} \triangleq \{u \in \mathbb{R}^{n_u} : \underline{u} \leq u_k \leq \bar{u},\ \forall k\}$, respectively, where $\underline{x}$ and $\bar{x}$ correspond to the lower and upper limits for the states, respectively. Similarly, $\underline{u}$ and $\bar{u}$ are the lower and upper limits for the control signals, respectively. Moreover, H_p and H_c refer to the prediction and control horizons respectively. It is assumed that each state is measurable, if not, a state observer should be implemented. Note that the cost function does not include the input u_k itself, but the changes Δu_k (slew rate) defined as

$$\Delta u_{k+i|k} = u_{k+i|k} - u_{k-1}, \tag{3}$$

where $u_{k-1} \in \mathbb{R}^{n_u}$ is the control input applied to the system in the previous iteration. This change is done with the aim of obtaining control actions with a smoother behavior [12].

Finally, the MPC relies on the Receding Horizon Strategy defined in Algorithm 1.

Algorithm 1 Receding Horizon Strategy

1: measure the state x_k at the time k
2: compute $u_k^*(x_k) := \left[u_{k|k}^{*\top}, \ldots, u_{k+H_p-1|k}^{*\top}\right]^\top$
3: apply the first element $u_{k|k}^{*\top}$ to the system
4: proceed to time step $k+1$
5: go to 1

B. QP Solvers

The optimization problem presented in (2) can be addressed as a QP problem of the form

$$\begin{aligned} \min_{z} \quad & \frac{1}{2} z^\top H z + g^\top z \\ \text{s.t.} \quad & a_{lb} \leq \Lambda z \leq a_{ub} \end{aligned} \tag{4}$$

where $H \in \mathbb{R}^{(n_x+n_u)\times(n_x+n_u)}$ is the Hessian matrix defined as semi-definite positive, i.e., it defines a convex optimization problem and, therefore, any locally optimal solution is also a globally optimal [2], and $z \in \mathbb{R}^{n_x+n_u}$ is the decision vector defined as $z = (x^\top \quad u^\top)^\top$. Moreover, $a_{lb}, a_{ub} \in \mathbb{R}^{n_x}$ and $\Lambda \in \mathbb{R}^{n_x \times (n_x+n_u)}$ define the feasible set and $g \in \mathbb{R}^{n_x+n_u}$ is the gradient vector.

To solve the QP problem presented in (4), two different QP solvers have been implemented and tested: CVXGEN and qpOASES.

1) CVXGEN: CVXGEN is a code generator for convex optimization problems developed by Jacob Mattingley and Stephen Boyd at Standford University [14]. CVXGEN uses a high-level and powerful language to define the optimization problem to be solved and automatically generates C flat code that can be compiled into high-speed solvers for the family of problems defined.

Once the optimization problem has been defined in the online interface at `www.cvxgen.com` and parsed into a canonical form used by CVXGEN to make the computations, CVXGEN implements a standard primal-dual interior point

method to find the solution. The main numerical algorithm was presented by Lieven Vandenberghe [2].

The CVXGEN website generates five main files in flat C. The core file is `solver.c`, where the main solver routine is implemented. Other important files are generated like `matrix_support.c`, allowing matrix and vector filling, `util.c`, where different useful functions are declared, or `ldl.c`, where the Karush-Kuhn-Tucker (KKT) factorization and solution are computed.

Besides, CVXGEN adds some files to interface the solver with MATLAB. In this paper, the controllers have been designed and simulated using this interface, while an interface with LabVIEW has been created (compiling the main C files into a Dynamic-Link Library (DLL)) for simulation and real-time implementation purposes.

2) qpOASES: qpOASES is a C++ based open-source tool for **q**uadratic **p**rogram resolution using a lately algorithm proposed: **o**nline **a**ctive **se**t **s**trategy [6]. This software was initially released by the KU Leuven within the Optimization in Engineering Center. The resolution algorithm used by qpOASES relies on the active set strategy [5] and incorporates an interface for third-party software such as MATLAB and Simulink. For the performed simulations, the interface with MATLAB has been used, while for the embedded task, a version of qpOASES, named `qpOASES_e`, has been tested since the latter is a plain C translation of the C++ based version and it is more suitable for being embedded into the device. As with CVXGEN, qpOASES_e has been compiled into a DLL to interface the main functionalities with LabVIEW for implementation purposes.

III. Problem Statement

A. Simulation and control models

The case study considered throughout this paper is the quadruple-tank process, presented by Karl H. Johansson [9] as a multivariable control process, ideal for educational purposes.

The quadruple-tank setup consists of four interacting tanks, two pumps, four level sensors and two valves (three-way valves) as shown in Figure 1. The goal is to control the water level of the two lower tanks (Tank 1 and Tank 2) while the inputs of the process are the voltages applied to the pumps (v_1 and v_2).

Tanks 1 and 2 are positioned below Tanks 3 and 4 and receive water from these latter by the gravitational action. A reservoir placed below Tanks 1 and 2 serves to accumulate the water from the tanks while the two pumps extract water from it. The water flow is split by two three-way valves whose position gives the ratio of how the flow rate is divided between upper and lower tanks.

The dynamic model is the result of applying mass balances and Bernoulli's law [9] and is represented by the following continuous-time differential equations:

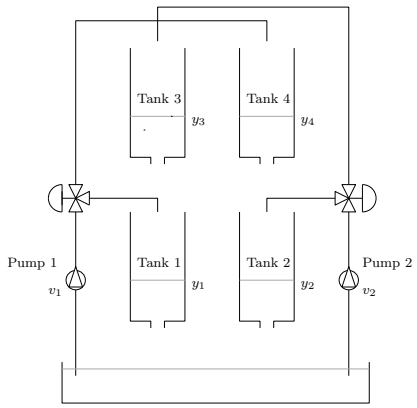


Figure 1: Schematic diagram of the quadruple-tank process

$$\frac{dh_1(t)}{dt} = -\frac{a_1}{A_1}\sqrt{2gh_1(t)} + \frac{a_3}{A_1}\sqrt{2gh_3(t)} + \frac{\gamma_1 k_1}{A_1}v_1(t), \quad (5a)$$

$$\frac{dh_2(t)}{dt} = -\frac{a_2}{A_2}\sqrt{2gh_2(t)} + \frac{a_4}{A_2}\sqrt{2gh_4(t)} + \frac{\gamma_2 k_2}{A_2}v_2(t), \quad (5b)$$

$$\frac{dh_3(t)}{dt} = -\frac{a_3}{A_3}\sqrt{2gh_3(t)} + \frac{(1-\gamma_2)k_2}{A_3}v_2(t), \quad (5c)$$

$$\frac{dh_4(t)}{dt} = -\frac{a_4}{A_4}\sqrt{2gh_4(t)} + \frac{(1-\gamma_1)k_1}{A_4}v_1(t), \quad (5d)$$

where

A_i	cross-section of Tank i,
a_i	cross-section of the outlet hole of Tank i,
h_i	water level of Tank i,
g	acceleration of gravity,
γ_i	constant of Valve i,
k_i	gain of the pump i.

To control the experimental setup, an identification of the nonlinear model has been made. The model parameters γ_i, k_i have been adjusted using a *nonlinear least squares* algorithm of the form

$$\min_{\hat{Y}_i} \quad \sum_{i=1}^{n} ||Y_i - \hat{Y}_i||^2, \quad (6)$$

where Y_i is the data measured and $\hat{Y}_i$ is the value predicted at the i-th data point [13] and the sampling time of the identification has been set to 0.1 s.

The resultant parameters from the identification procedure are summarized in Table I.

Table I: Identified and used parameters

Parameter	Value
A_i [cm^2]	138.9
a_i [cm^2]	0.5027
k_1 [cm^3/Vs]	26.00
k_2 [cm^3/Vs]	22.94
γ_1	0.836
γ_2	0.897
g [cm/s^2]	981

Note that the problem is not symmetric since the two pumps do not have the same gain k_i, just like the constants referring the overture of the valves γ_i do not have the same value.

The system behavior according to (5) has nonlinear dynamics, whose model will act as the simulation oriented model (SOM). Since an LMPC will be designed, the model in (5) should be linearized and discretized in order to obtain the corresponding control oriented model (COM). Finally, an implementation for simulation and experimental purposes, comparing the performance of the closed loop when two different solvers are used to solve the optimization problem behind the MPC controller, is done.

The COM, once linearized at the operating point x_{op}, u_{op} where

$$x_{op} = \begin{bmatrix} 10 & 10 & 0.0892 & 0.3156 \end{bmatrix}, \tag{7}$$

$$u_{op} = \begin{bmatrix} 2.9336 & 2.8140 \end{bmatrix}, \tag{8}$$

with a sampling time of 1.430 s, has the form of (1) with

$$A = \begin{bmatrix} 0.9644 & 0 & 0.3127 & 0 \\ 0 & 0.9644 & 0 & 0.1811 \\ 0 & 0 & 0.6812 & 0 \\ 0 & 0 & 0 & 0.8154 \end{bmatrix}, \tag{9}$$

$$B = \begin{bmatrix} 0.2198 & 0.0041 \\ 0.0041 & 0.2081 \\ 0 & 0.0202 \\ 0.0397 & 0 \end{bmatrix}, \tag{10}$$

where the states are the heights of the tanks, the control inputs are the voltages applied to the pumps and the outputs are the states themselves since a full-state feedback is implemented and no state observer is designed (four level sensors are used). Finally, the LMPC controller implemented should solve at each time instant the following optimization problem:

$$\begin{aligned} \min_{u_{k|k},\ldots,u_{k+H_p-1|k}} \quad & \sum_{i=1}^{H_p} ||x_{k+i|k} - x_r||_Q + \sum_{i=0}^{H_c-1} ||\Delta u_{k+i|k}||_R \\ \text{s.t.} \quad & COM, \\ & u_{k+i|k} \in \mathcal{U}, \\ & x_{k+i|k} \in \mathcal{X}, \end{aligned} \tag{11}$$

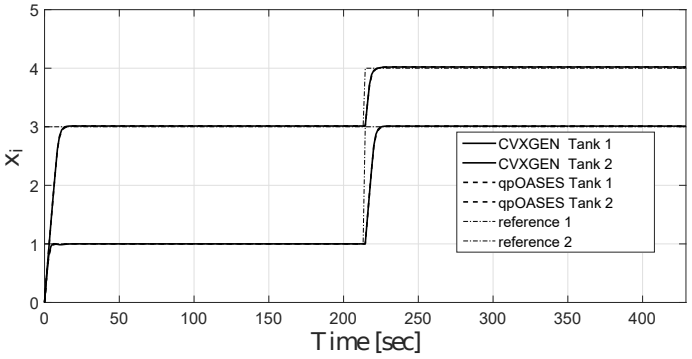
where the physical and operating constraints are summarized in Table II.

Table II: Simulation parameters

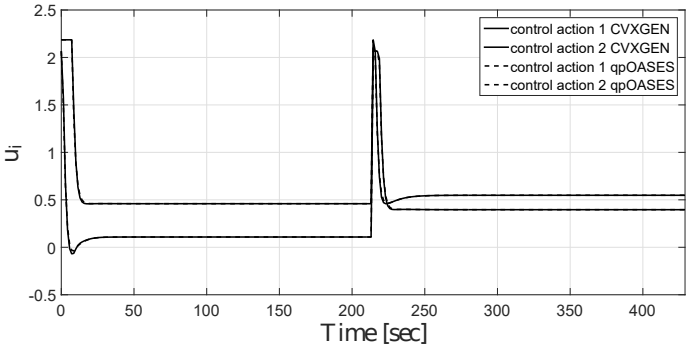
Parameter	Value
$\underline{x}$ [cm]	0
$\bar{x}$ [cm]	25
$\underline{u}$ [V]	0
$\bar{u}$ [V]	5
H_c, H_p	10

B. Simulation results

The tuning parameters for the controllers are the weighting matrices Q and R. The control objective is to track a height set-point, therefore Q must be greater than R (in the sense of the magnitude of its entries). In this case, after running some simulations, it has been set `Q=100diag[1 1]` and `R=10diag[1 1]`. The simulation results are summarized in Figure 2.



(a) h_i results for the two solvers implemented



(b) u_i results for the two solvers implemented

Figure 2: h_i and u_i solution by the different solvers in simulation

The heights of the Tanks 1 and 2 reach and track the desired set-point thanks to the greater weight given by Q as expected. It is more remarkable to take a look at the computational time spent by each solver, since it can give an idea about which solver will have faster performance into the real-time setup. For this simulation experiments, 300 iterations of the online MPC have been run by each solver, running five times the online MPC problem with the respective solvers and taking the minimum time spent among the simulations carried out, giving the results summarized in Table III.

Table III: Computational time spent by the considered solvers in simulation.

Solver	Time [s]
CVXGEN	0.084
qpOASES	0.103

CVXGEN and qpOASES have a similar performance regarding the execution time. Remember that the qpOASES versions used for simulation and implementation are not the same one. The simulations have been performed using a PC with 8 GB of RAM and a core-i7 processor running MATLAB 2015a.

Figure 3: myRIO device. Photography taken from [15]

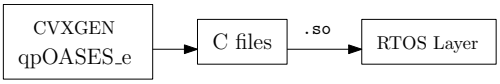


Figure 4: Embedding process scheme

IV. Experimental Setup

A. Pilot Plant

The plant to be controlled is the Coupled Tanks System 33-230 [4] from Feedback Instruments Ltd. The scheme in Figure 1 and the real setup differ in the valves since the model identified implements two three-way valves and the real plant uses two two-way valves. This difference has impact in the identification process performed.

B. Controller Device and Embedding process

The hardware used to implement the MPC controller is the NI myRIO-1900 from National Instruments [15] shown in Figure 3. This device has an FPGA that uses the *LabVIEW FPGA* operating system, and a microprocessor that uses the *LabVIEW Real-Time Operating System* (LRTOS). The LRTOS is responsible of managing the hardware resources and hosting the applications defined by the user. This operating system is designed to run applications with highly-precise timing, fact that is crucial in real-time applications, adding an important degree of reliability. In terms of the controller embedding process, the LRTOS has a C and C++ code compiler whereby the QP solvers will be compiled into DLLs and embedded in the real-time processor layout. In this case, the RTOS is linux based, then the embedded QP solvers must be compiled as `.so` libraries (Figure 4). These libraries will have two main functions: the former is to compute the optimal control inputs for the pumps once the set-point defined by the user is given and the latter is to exchange the data from both analog inputs and outputs with the host PC. The FPGA layout is in charge of scaling both input and output values obtained by the A/D converter.

Finally, a low-pass filter is placed at the A/D converter output in order to obtain noise attenuation.

C. Experimental results

To make sure that the embedded controller is performing properly, a test for the two QP solvers implemented has been run with the device disconnected from the plant, just to estimate the capabilities of the myRIO to run the respective solver. The first aspect that was tested was that the optimizers solved the problem correctly. Once verified its suitable operation, the computational time spent by each algorithm to solve an optimization problem was measured yielding the results summarized in Table IV.

Table IV: Computational time spent, by each solver embedded in the real-time device, to perform a closed-loop iteration.

Solver	Time [ms]
CVXGEN	300
qpOASES_e	1430
FiOrdOs	3480

These results show that the device used can not ensure the correct performance of the online LMPC problem with the FiOrdOs solver, since its computation time per iteration is more than three seconds, producing an oscillating behavior of the system outputs around the given set-point.

The implemented strategy is collected in Algorithm 2.

Algorithm 2 Synchronous strategy implemented

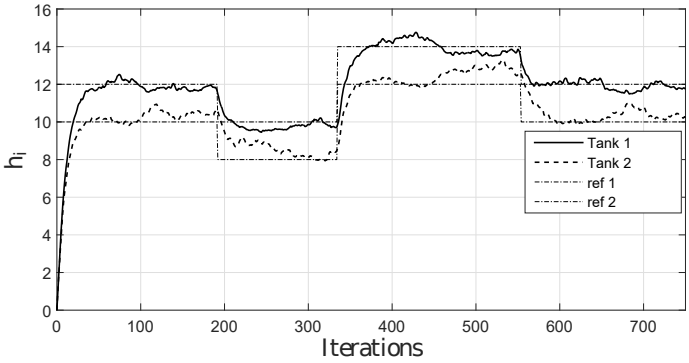
1: give the desired set-point to be tracked
2: read the analog inputs from the four tanks
3: compute $\mathbf{u}_k^*(x_k) := \left[u_{k|k}^{*\top}, \ldots, u_{k+H_p-1|k}^{*\top}\right]^\top$
4: take the first element $u_{k|k}^{*\top}$
5: perform the A/D conversion carried out by the FPGA
6: apply the result of the conversion to the system
7: go to 1

In terms of execution time capabilities of the device used, the myRIO can use a sampling rate of the analog inputs (height reads) between 1 kHz and 30 kHz, more than enough to ensure that the online MPC is updated correctly. The experiment performed, for both solvers, has been to make set-point changes that the Tanks 1 and 2 should track. The results are shown in Figures 5a and 5b for CVXGEN and qpOASES, respectively.

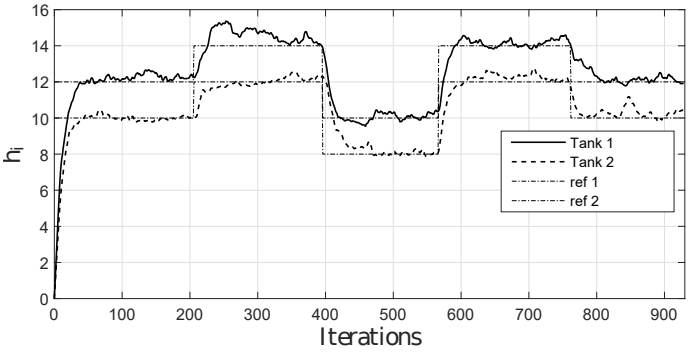
CVXGEN and qpOASES have shown a similar performance in spite of the different execution time, thanks to the chosen synchronous strategy, with which the two solvers execute the closed-loop at the same rate. It should be mentioned that when the problem grows in dimensions (H_p or the number of decision variables increase), CVXGEN becomes much slower than qpOASES, while the latter is more robust in terms of runtime and then it seems to be more suitable for being applied in large-scale problems.

V. Conclusions

In this paper, the use of an online LMPC controller, embedded in an educational device composed of both an FPGA and a microprocessor, has been proposed and discussed using two different QP solvers of which one offers code generation



(a) h_i with CVXGEN



(b) h_i with qpOASES

Figure 5: Tracking task performed by both considered algorithms

(CVXGEN), while to embed the other, more advanced C and compilation notions are required.

A nonlinear identification of the system model has been performed to adjust the unknown parameters (grey-box identification). A linear discrete-time state-space model has been obtained linearizing the previous model around an operating point towards the design of the LMPC controller. Despite the limitations of the used device (it is an educational instrument), an online LMPC has been embedded using two different numerical algorithms that, in this paper, demonstrate similar behaviors. FiOrdOS has also been tested, but a reliable implementation has not been achieved due to the computational time spent by the fast-gradient algorithm [16].

The considered solvers, capable of being embedded in real-time applications, can be great tools in case that the needs of both execution and computational time can be met, and they can be used instead of stablished toolboxes, e.g., Control Design and Simulation Module from LabVIEW or the Model Predictive Control Toolbox from MATLAB, since these latter are black-box-based tools leaving few possibilities of interaction between the user and the problem defined.

ACKNOWLEDGMENT

This work has been supported by the project DEOCS (Ref. DPI2016-76493-C3-3-R) from the Spanish Ministry of Economy and Competitiveness.

REFERENCES

[1] Alberto Bemporad, Manfred Morari, Viek Dua, and Efstratios N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] Stefano Di Cairano, Wei Liang, Ilya V. Kolmanovsky, Ming L. Kuang, and Antony M. Phillips. Power smoothing energy management and its application to a series hybrid powertrain. *IEEE Transactions on Control Systems Technology*, 21(6):2091–2103, 2013.

[4] Feedback Instruments Ltd. *Coupled Tanks. Installation and Commissioning. 33-041-IC*, 2013.

[5] Hans J. Ferreau, Hans G. Bock, and Moritz Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International journal of robust and nonlinear control*, 18:816–830, 2007.

[6] Hans J. Ferreau, Christian Kirches, Andreas Potschka, and Moritz Diehl. qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

[7] Tobias Geger, Georgios Papafotiou, Roberto Frasca, and Manfred Morari. Constrained optimal control of the step-down DC–DC converter. *IEEE Transactions on Power Electronics*, 23(5):2454–2464, 2008.

[8] Bart Huyck, Hans J. Ferreau, Moritz Diehl, Jos De Brabanter, Jan F. M. Van Impe, Bart De Moor, and Filip Logist. Towards online model predictive control on a programmable logic controller: Practical considerations. *Mathematical Problems in Engineering*, vol.2012:20, 2012.

[9] Karl H. Johansson. The quadruple-tank process: A multivariable laboratory process with an adjustable zero. *IEEE Transactions on Control Systems Technology*, 8(3):456–465, 2000.

[10] Eric C. Kerrigan, George A. Constantinides, Andrea Suardi, Andrea Picciau, and Bulat Khusainov. Computer architectures to close the loop in real-time optimization. *IEEE 54th Annual Conference on Decision and Control*, 2015.

[11] Dimitris Kouzoupis, Andrea Zanelli, and Hans J. Ferreau. Towards proper assessment of QP algorithms for embedded model predictive control. *European Control Conference (ECC)*, 2015.

[12] Jan M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, 2001.

[13] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[14] Jacob Mattingley and Stephen Boyd. CVXGEN: A Code Generator for Embedded Convex Optimization. *Optimization and Engineering*, 13(1):1–27, 2012.

[15] National Instruments. *User Guide and Specifications. NI myRIO-1900*, 2013.

[16] Yurii Nesterov. *Introductory Lectures on Convex Optimization*. Springer, 2004.

[17] James B. Rawlings and David Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2015.

[18] Stefan Richter, Colin N. Jones, and Manfred Morari. Real-time input-constrained MPC using fast gradient methods. *IEEE Conference on Decision and Control*, 2009.

[19] Fabian Ullmann. FiOrdOs: A Matlab Toolbox for C-Code Generation for First Order Methods. Master's thesis, ETH Zurich, 2011.

[20] Yang Wang and Stephen Boyd. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, 18(2):267–268, 2010.