Preprints of the 2012 20th Mediterranean Conference on Control & Automation (MED), Barcelona, Spain, July 3-6, 2012

Implementation and Experimental Validation of Classic MPC on Programmable Logic Controllers

B. Huyck, L. Callebaut, F. Logist, H. J. Ferreau, M. Diehl, J. De Brabanter, J. Van Impe and B. De Moor

Abstract— **Over the last years, a number of publications were written about Model Predictive Control (MPC) on industrial Programmable Logic Controllers (PLC). They focussed on explicit MPC strategies to provide a fast solution. When sufficient time is available to solve a classic MPC problem, an online solution to the corresponding Quadratic Problem (QP) can be provided. This paper investigates the use of an online quadratic programming solver to exploit MPC on a PLC. This will be illustrated with the classic Hildreth QP algorithm and qpOASES, a recently developed online active set strategy. These algorithms will be investigated on a MISO system.**

I. INTRODUCTION

Model Predictive Control has become a mature control technique. It is widely spread for complex and large installations, particularly in the process industry. In the recent years, the MPC community evolved towards high speed MPC and embedded systems. The hardware used are typical FPGA devices and microcontrollers. A PLC is a widely used microcontroller adapted to meet industry standard in- and outputs and communication protocols. These devices can be easily programmed and expanded to meet the needs of an industrial control or automation problem. The common PLC programming languages are standardized and differ from the typical PC programming languages. In order to use algorithms from literature, a manual time-consuming translation of the algorithm is needed. This is one of the reasons why trials to run MPC on PLCs employ explicit MPC [1], [2], [3] as less PLC code is needed. This approach uses a precomputed solution stored in look-up tables. For MIMO systems with a large control horizon, i.e. more than 10, these solutions are sub-optimal as the time to search the look-up tree becomes prohibitive. These methods therefore seem perfectly suited for small SISO (Single Input - Single Output) systems. For large MIMO systems *Pannocchia et al.* [4] suggested to use online QP solvers. This paper investigates the possibility of using standard industry hardware for classic MPC applications on small MIMO (Multiple Input - Multiple Output) systems. Classic MPC solves the corresponding quadratic problem online. This article examines two online QP algorithms to solve the MPC problem online. First, the Hildreth QP algorithm [5], a classic but easy to implement algorithm, will be used. It will be compared to qpOASES, a state-of-the-art online active set algorithm [6]. Parts of the code which are not necessary for this application have been removed. The application consists of a heating device where fan speed and resistor power can be manipulated independently. Although small scale, it is an interesting test application [1]. Both algorithms will be compared with regard to their applicability and the time needed to solve the online MPC problem for this application. This paper is structured as follows: Section II deals with the MPC formulation used in this paper. Section III describes the experimental set-up and the applied model. Section IV contains the practical implementation of the MPC algorithm on a Siemens PLC. Results are discussed in Section V and finally, the main conclusions will be summarized in Section VI .

Bart Huyck, Lennert Callebaut and Jos De Brabanter are with the Department of Industrial Engineering, KAHO St-Lieven, B-9000 Gent, Belgium prename.name@kahosl.be

Bart Huyck, Hans Joachim Ferreau, Jos De Brabanter, Moritz Diehl and Bart De Moor are with the Department of Electrical Engineering, (ESAT - SCD), KU Leuven, B-3001 Leuven, Belgium prename.name@esat.kuleuven.be

Bart Huyck, Filip Logist and Jan Van Impe are with the Department of Chemical Engineering (CIT - BioTeC), KU Leuven, B-3001 Leuven, Belgium prename.name@cit.kuleuven.be

II. MODEL PREDICTIVE CONTROL FORMULATION

Linear Model Predictive Control is well known in literature [7], [8], [9] and the reader is invited to read these works for a detailed description. The basic formulation needed for this paper is given below.

A linear discrete-time system is described by:

$$\begin{aligned} \mathbf{x}(k+1) &= A\mathbf{x}(k) + B\mathbf{u}(k) \\ \mathbf{y}(k) &= C\mathbf{x}(k), \end{aligned} \tag{1}$$

where $A \in \mathbb{R}^{nxn}$, $B \in \mathbb{R}^{nxm}$ and $C \in \mathbb{R}^{pxn}$ with m, n and p respectively the number of inputs, states and outputs. The objective of the controller is finding the optimal input for this system by means of minimizing a cost function, which for the purpose of this paper, is assumed to be the following:

$$\begin{aligned} J &= \sum_{i=1}^{H_p} \|\mathbf{y}(k+i|k) - \mathbf{r}(k+i|k)\|^2_{W_y} \\ &\quad + \sum_{j=1}^{H_c} \|\mathbf{u}(k+j|k)\|^2_{W_u} . \end{aligned} \tag{2}$$

H_p and H_c are respectively the prediction and control horizon of the controller. $H_c \leq H_p$ is assumed. $W_y \in \mathbb{R}^{pxp}$ and $W_u \in \mathbb{R}^{mxm}$ are positive definite weighting matrices and $\mathbf{r}$ is the output reference. The formulation $\mathbf{y}(k+i|k)$ represents the vector $\mathbf{y}$ on time $k+i$ at calculation time k.

The power of MPC lies in the possibility of adding constraints. For this paper only input constraints are taken into account. Output and state constraints are omitted. Given

978-1-4673-2529-5/12/$31.00 ©2012 IEEE

the constraints for the input: $\mathbf{u}_{Min} \leq \mathbf{u} \leq \mathbf{u}_{Max}$, the optimization problem can be formulated as:

$$\begin{aligned} \text{Minimize } J &= \sum_{i=1}^{H_p} \|\mathbf{y}(k+i|k) - \mathbf{r}(k+i|k)\|^2_{W_y} \\ &\quad + \sum_{j=1}^{H_c} \|\mathbf{u}(k+j|k)\|^2_{W_u} \quad (3) \\ \text{Subject to} \quad & \mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k) \quad (4) \\ & \mathbf{y}(k) = C\mathbf{x}(k) \\ & \mathbf{u}(j) \leq \mathbf{u}_{Max} \\ & \mathbf{u}(j) \geq \mathbf{u}_{Min} \end{aligned}$$

In order to solve this problem, the optimization problem is reformulated by elimination of the states in the form of a Quadratic Problem:

$$\begin{aligned} \text{Minimize } J &= \theta^T H \theta + g^T \theta \quad (5) \\ \text{Subject to} \quad & P\theta \leq \alpha \quad (6) \end{aligned}$$

To this end, following steps are taken. First the state space model is rewritten in terms of a new input $\mathbf{u}' = [\Delta\mathbf{u}\ y]^T$ to add integration. The prediction along the prediction horizon is written in matrix formulation and is be formulated as:

$$\mathbf{Y} = F\mathbf{x}(k) + H\mathbf{U}', \quad (7)$$

with $\mathbf{Y}$ and $\mathbf{U}'$ column matrices of respectively outputs and inputs. E.g. $\mathbf{U}' \in \mathbb{R}^{(p+m)H_c \text{x} 1}$ composed of $\mathbf{u}'(k|k)$ to $\mathbf{u}'(k+H_c|k)$. The matrices F and H can be found in many works on MPC [7], [9]. The matrix H is postprocessed by including the weight matrices as follows:

$$H' = H^T W_{y_{bd}} H + W_{u_{bd}} \quad (8)$$

with $W_{y_{bd}}$ and $W_{u_{bd}}$ blockdiagonal matrices of respectively W_y and W_u. As for the purpose of this work, the aim is to minimize the online calculation work, matrices that can be computed in advance and are calculated offline. Matrix H' is one of the precomputed matrices that does not change during runtime. Vector g from (5) has to be calculated online and contains two parts depending on the current state and the reference of the in- and outputs. So, g has to be calculated according to:

$$g = G1\mathbf{x} - G2\mathbf{Y}_{\mathbf{ref}} \quad (9)$$

where $G1$ and $G2$ are gradient matrices. $\mathbf{Y}_{\mathbf{ref}}$ is an $\mathbb{R}^{pH_c \text{x} 1}$ matrix of the references $\mathbf{r}(k|k)$ to $\mathbf{r}(k+H_c|k)$. The gradient matrices are constant and are computed offline:

$$\begin{aligned} G1 &= H'^T W^T_{y_{bd}} F; \quad (10) \\ G2 &= H'^T W^T_{y_{bd}}; \quad (11) \end{aligned}$$

The constraints, i.e. the minimum and maximum admissible value for $\mathbf{U}'$, are calculated online. For this paper, the constraints are constant during the experiment. Finally, the QP problem to be solved is:

$$\begin{aligned} \min_{\mathbf{U}'} \quad & \tfrac{1}{2}\mathbf{U}'^T H' \mathbf{U}' + g\mathbf{U}' \quad (12) \\ \text{Subject to} \quad & \mathbf{U}'_{Min} \leq \mathbf{U}' \leq \mathbf{U}'_{Max} \quad (13) \\ & g = G1\mathbf{x} - G2\mathbf{Y}_{\mathbf{ref}} \quad (14) \end{aligned}$$

Both the Hildreth [5] and the qpOASES [6] algorithm will be used to solve this problem.

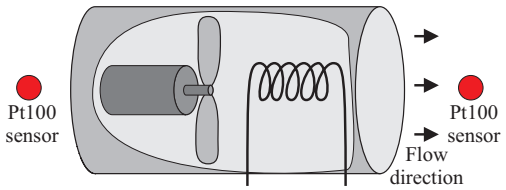

Fig. 1. Schematic overview of the temperature control set-up.

III. Experimental temperature control Set-up

A. Hardware and software

The temperature control set-up consists of a resistor and a fan which can be controlled separately. The fan is driven by a 24V DC motor and the resistor has a maximum power of 1400 W. The heating power delivered by the resistor can be adapted by a solid state relay with analog control (Gefran GTT 25A 480VAC - analog control voltage 0-10 V). The fan is controlled by a custom made DC Drive based on a TEXAS INSTRUMENTS DRV102T chip and adapted for an analog control voltage of 0-10 V. Temperature sensors measure the environmental temperature and the temperature of the heated air as indicated in Fig. 1. Both sensors are of the PT100 class B type. The controller is a Siemens CPU319-3DP/PN equipped with an 8Mb memory card and an additional SM334 analog card to connect to the solid state relay and DC drive. The Siemens CPU is programmed using Step 7 Professional 2010. To code the problem, the *Structured Control Language* (S7-SCL) is used.

B. Model Identification

As MPC requires a model, this temperature control set-up is modeled. Based on a multisine input signal, a two-input (fan speed and resistor power) and single output (temperature) model is created. Both the construction of the input signal and the identification are performed using the Matlab System Identification Toolbox [10]. The resulting model consists of first order transfer functions:

$$T_n = \begin{bmatrix} \frac{-0.98}{1+5.17s}e^{-1.34s} & \frac{0.83}{1+7.17s}e^{-1.53s} \end{bmatrix} \begin{bmatrix} u_{Fan,n} \\ u_{Power,n} \end{bmatrix} \quad (15)$$

In (15) the index n indicates a normalized variable. $u_{Fan,n}$ and $u_{Power,n}$ are respectively the normalized and detrended input signal for the motor drive and the normalized and detrended voltage applied to the solid state relay of the resistor. After detrending, a zero output of the model corresponds to

42°C. For the inputs, the zero input corresponds to 5 V. More information about this set-up and this identification procedure can be found in [11].

C. Controller Design

Model (15) is converted to a discrete state space model of order 4. The sampling rate is 1 Hz. It has been verified that the model is controllable, observable and stable. The control horizon H_c of the controller is set to 7 and the prediction horizon H_p is 22. These horizons have been chosen similar to those in [11] to compare the different controllers and control algorithms for this temperature control set-up.

IV. Implementation on the hardware

The current evolution towards code generation and MPC toolkits that produce ready-to-run C/C++-code, makes it possible to quickly implement MPC algorithms on micro-controllers and computers. This type of devices are often lab equipment or development tools. A lot of standard industrial hardware is not able to run this generated code as a different programming standard is employed or compilers are unavailable. Closed software or the fact that the real-time behavior of the device cannot be guaranteed are often reasons for this situation. Siemens S7-300 PLC CPUs do not have a port to C/C++. Therefore, to run MPC on these devices, the applied QP solvers have to be translated in a suitable supported language. In this paper S7-SCL is used.

A. Hildreth algorithm

This algorithm has been chosen due to its easy implementation which does not require the use of matrix operations like, e.g. transpose. The employed PLC has a limited computational power, so constant parts of the algorithms which be calculated offline are calculated offline. The data is stored in matrices to keep the link with the theory, but the computation of the optimal input is executed element-wise.

The Hildreth algorithm calculates the solution in two steps [9]. First, the unconstrained solution is calculated and if no constraints are violated, this solution is presented at the outputs of the PLC. If a constraint is violated, a QP will be solved. The solution of the QP is then passed to the inputs of the heating device. For more information about the solution routing, see [5], [9], [12]. Important for the online use of this algorithm is that there is no guaranteed solution that fulfills the constraints if it is stopped early. So, if the time needed to calculate the solution exceeds the available time, the algorithm will be stopped, but it cannot be ensured that the constraints are satisfied. In such a case, the solution of the previous QP is used.

B. qpOASES algorithm

qpOASES is an open-source C++ implementation of the recently proposed online active set strategy [6]. It builds on the idea that the optimal sets of active constraints do not differ much from one QP to the next. At each sampling instant, it starts from the optimal solution of the previous QP and follows a homotopy path towards the solution of the current QP. Along this path, constraints may become active or inactive as in any active set QP solver and the internal matrix factorizations are adapted accordingly. While moving along the homotopy path, the online active set strategy delivers sub-optimal solutions in a transparent way. Therefore, such sub-optimal feedback can be reasonably passed to the process in case the maximum number of iterations is reached.

As no port to C/C++ exists, a simplified version of qpOASES has been translated to S7-SCL and was used for controlling the heating device. Note that our simplified implementation does not allow for hotstarting the QP solution and is not fully optimized for speed. Moreover, it only handles bounds on the control inputs but no general constraints.

C. Steps to compute a solution

To compute the inputs, the following sequence of actions presented in Algorithm 1 are programmed. In advance, constant matrices are precomputed and the reference trajectory for the output temperature is defined. The input reference is set to zero for the algorithm, which corresponds to 5 V for both inputs.

```
1  Offline: Calculate H', G1, G2 and G3;
2  Online: Start PLC;
3  Store all precomputed matrices in working memory,
   together with the reference for inputs and output;
4  while PLC is running do
5      if 1 second passed since last call then
6          Calculate current state with current information
           of the system in- and outputs;
7          Calculate g, U'_min and U'_max;
8          while maximum numbers of iterations is not
           reached and solution not found do
9              Solve one iteration of the QP;
10             Check if solution is found or maximum
               numbers of iterations is reached;
11         end
12         switch max. numbers of iterations is reached do
13             case Hildreth
14                 Use solution of the former QP;
15             endsw
16             case qpOASES
17                 Use last sub-optimal solution;
18             endsw
19         endsw
20         Apply the inputs to the system;
21     end
22 end
```

Algorithm 1: Steps to compute the inputs of the system.

D. Programming the PLC

To calculate the appropriate inputs of the system and solve the QP, following build-in function blocks (FB) and organization blocks (OB) are programmed.

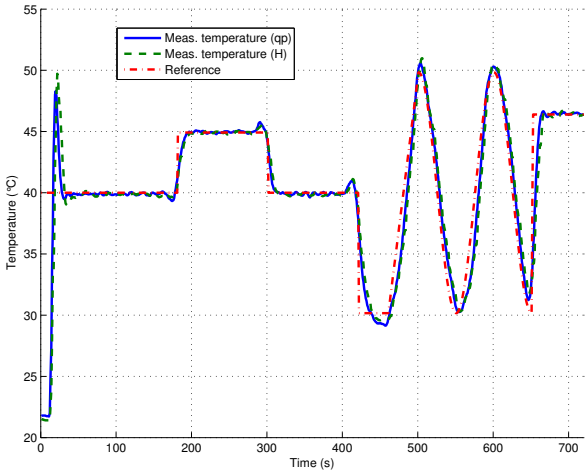

Fig. 2. Measured temperature for both the Hildreth (H) and qpOASES (qp) algorithm. The weight matrix for the inputs of the systems is the identity matrix (W_u^{11}).

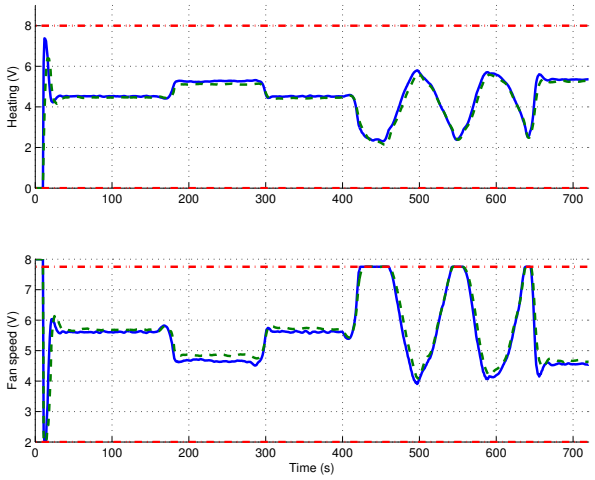

Fig. 3. The applied inputs to the system for both the Hildreth (H) and qpOASES (qp) algorithm. The weight matrix is the identity matrix (W_u^{11}).

1) OB 100 - Cold Start: This block is called once when the controller is started. It calls function FB 1, which is used to initialize the precomputed matrices which are stored in FB 2. This procedure is followed to reduce the look-up time of the variables during the calculation of the solution and to overcome the limitation that an array of constants cannot be larger than 256 elements at compilation time. In the case a matrix needs to contain more than 256 elements, more arrays are combined in this block at runtime into a combined array. For these experiments, only matrix $G2$ is initialized in this function.

2) OB 1 - main loop: This loop is started as soon as OB 100 is finished. When this function finishes, it restarts again. This loop is used to program standard tasks of the PLC. In this experiment, it is not used. It will run during the idle time of the CPU between two OB35 calls.

3) OB 35 - timed loop: This organization block is called every second. It contains the necessary code to read the current controlled and environmental temperature. This information is scaled and employed to calculate the current state (FB 3). Together with the reference for the in- and outputs, the state is used to update matrix g (FB 2). Now the QP is solved and the scaled solution will be passed to the outputs of the PLC. The absolute maximum execution time of an OB is six seconds, otherwise the system will raise a system failure. This is a hardware setting that cannot be overruled by the user. For the current experiments, the maximum execution time for OB 35 is set to 980 ms.

V. Results

Two cases with a different weight matrix W_u in the cost function are presented. In the first case, the weight matrix is the identity matrix $W_u = I_{2x2}$. We will use W_u^{11} to refer to this case. W_u^{15} will be used for the second case where the weight matrix is:

$$W_u = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix}. \tag{16}$$

Fig. 2 depicts the measured temperature and its reference temperature for both the Hildreth and qpOASES algorithms in case W_u^{11}. This reference temperature is a constant temperature of 40°C during 200 s, followed by a jump towards 45°C during 120 s, again a temperature of 40°C during 120 s and a set-point of 45°C for 60 s. From here, the temperature ramps up to 50°C at a rate of one degree per second. This ramp is followed by a sine wave for one and a half period with a period length of 50 s. After a large jump towards 47°C, the temperature is kept constant until the end of the experiment. The corresponding inputs of the system are displayed in Fig. 3. Fig. 4 depicts the same reference trajectory as Fig 2 for W_u^{15}. In this case, the deviation from the input corresponding to the heating is punished five times more than the deviation from the reference of the fan speed. The inputs of the corresponding experiment are plotted in Fig. 5. The logging of the experiments is done on a computer. Every experiment starts when the PLC is switched on. This means a connection has to be established between PLC and PC at the start of each experiment. Hence, it is unknown in advance when the logging starts. To deal with this problem, every experiment begins with a fixed fan speed and switched-off heating for 30 s. This shortens the 200 s starting set-point at 40°C to 170 s. All plots only show the last 5 s of these startup period. This start-up phase also gives the estimator the time to calculate the best possible current state of the system.

A. Hildreth Algorithm

For W_u^{11}, the Hildreth algorithm follows the desired reference temperature correctly with some small disturbances caused by the environmental conditions. A large overshoot is observed at the start caused by the inaccurate state estimation of the model. After all, the model is a linearized model around 42°C. The environmental temperature is almost three times the standard deviation away form its nominal value resulting in a bad state estimation. Every further step is taken smoothly and over- of undershoots are part of the properties

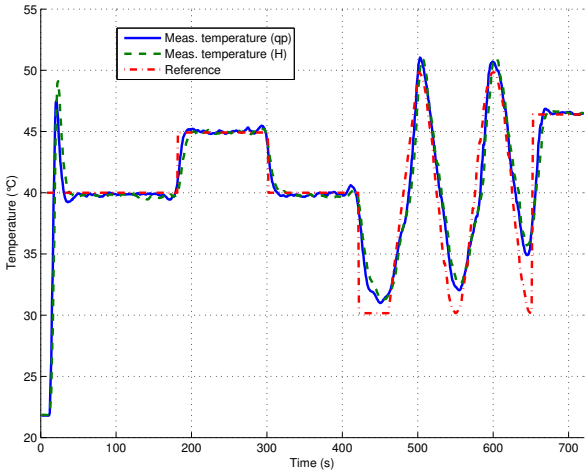

Fig. 4. Measured temperature for both the Hildreth (H) and qpOASES (qp) algorithm. The weight matrix for the inputs of the systems W_u^{15}.

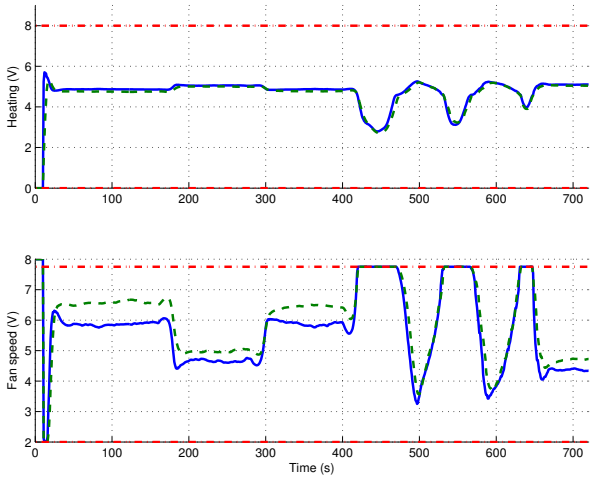

Fig. 5. The applied inputs to the system for both the Hildreth (H) and qpOASES (qp) algorithm. The weight matrix is W_u^{15}.

of the algorithm. The ramp and sine wave are followed close by both experiments, but a small delay appears. For W_u^{15}, the first part of the experiment until 450 s, is very similar to the W_u^{11}. The error for the ramp and sine wave is clearly larger, certainly in the lower temperature range. The sum of squared error is nearly doubled as printed in Table I.

TABLE I

THE SUM OF SQUARED ERRORS FOR THE RAMP AND SINE WAVE (FROM 421 S UNTIL 651 S) FOR THE EXPERIMENTS DEPICTED IN FIG. 2 AND 4

	Hildreth	qpOASES
W_u^{11}	1397	762
W_u^{15}	2406	1406

For W_u^{11}, the two inputs remain closely to their reference (5 V, zero for the algorithm). The constraints are hit four times by the fan speed. Fig. 6 depicts the required number of iterations of both algorithms and Fig. 7 plots the corresponding calculation time. For W_u^{15}, the constraints are also hit four times, but for a longer time as deviation of the reference for the heating input is punished five times more than in the first case. For both W_u^{11} and W_u^{15}, the Hildreth algorithm needs between 10 and 25 iterations if a QP has to be solved. Each iteration takes approximately 1 ms resulting in a maximum runtime of 25 ms for these experiments. It has to be noted that in the described experiment only one constraint is active.

B. qpOASES Algorithm

For both W_u^{11} and W_u^{15}, the reference is followed in a similar way for both the Hildreth and qpOASES algorithm. The differences can only be explained by differences in the solution of the QP and different environmental conditions. From both Fig 2 and 4, it can be noticed that the Hildreth algorithm reacts a little bit slower. It is typically one second behind the qpOASES solution, resulting in e.g. a larger overshoot at the start, although both algorithms start at the same moment in similar circumstances.

The first step on the homotopy path for qpOASES takes approximately 10 ms. If no constraint is active, the optimal solution is found immediately. If a constraint is active, maximum 7 iterations were needed for these experiments by qpOASES performing cold-starts. Again, it has to be noted that only one constraint was active during the experiment. For the additional iterations, qpOASES needs approximately 15 ms resulting in a maximum runtime to get a solution of 120 ms. For W_u^{15} (Fig. 3), the inputs of the system are similar for both algorithms. qpOASES hits the constraints at the same four moments, but one second earlier than Hildreth. The constraints are also left one second earlier. For W_u^{11} (Fig. 5) the same evolutions are seen. The clearly different path of the fan speed during the first 400 s in Fig. 5 is caused by different environmental conditions.

C. Comparison of both algorithms

Both algorithms solve the QP online. Compared with respect to computing time, the Hildreth algorithm is faster. If no constraints are active, this algorithm calculates a solution 10 times faster. If constraints are hit, the number of iterations is lower for qpOASES. Due to the advanced mathematics in the algorithm, it still takes more time to get a solution than with the Hildreth algorithm. Table I lists the sum of squared errors (SSE) for the ramp and sine wave. The SSE values are only calculated for a part of the experiment, but are representative for the whole experiment. When both algorithms are compared, the SSE of qpOASES is clearly lower than the one for the Hildreth algorithm. For W_u^{15} the SSE is nearly doubled for the Hildreth algorithm, for W_u^{15} the SSE is 70% higher than for qpOASES. It is clear that qpOASES has a higher accuracy and is preferred over the Hildreth algorithm. Although not described in this paper, identical experiments are set up with different start-up conditions. During the first 30 s of the experiment, the inputs passed to the estimator were the inputs of the former QP solution, not a manually fixed set as depicted in the plots. This resulted in an estimated state corresponding

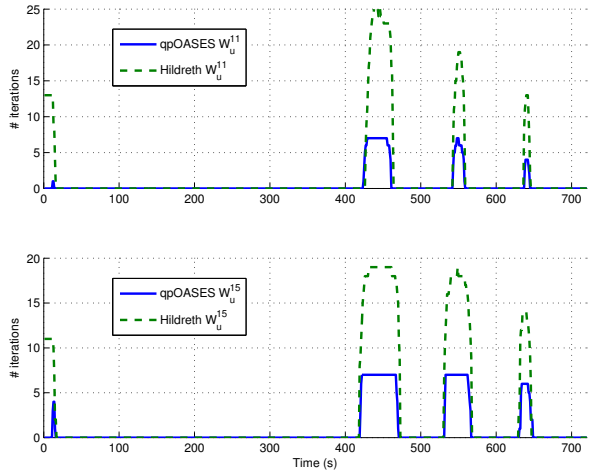

Fig. 6. Number of iterations needed to solve the QP problem. Above: W_u^{11} - both inputs are punished equally. Below W_u^{15} - the input controlling the heating is punished five times more than the input controlling the fan.

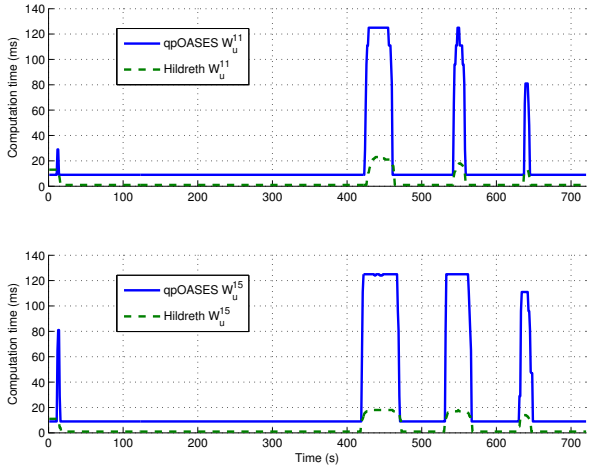

Fig. 7. Time needed to solve the QP problem. Above: W_u^{11} - both inputs are punished equal. Below W_u^{15} - the input controlling the heating is punished five times more than the input controlling the Fan.

to approximately 50°C and inputs touching both heating and fan constraints. The Hildreth algorithm needed more than 60 iterations at that point, the maximum allowed, and was stopped early. The former solution was passed to the estimator (but not to the process). qpOASES, on the other hand, calculated the solution in maximum 14 iterations and provided this to the estimator. Maximum 20 iterations were allowed. This behavior demonstrates that for the Hildreth algorithm appropriate measures need to be taken to handle these situations. For this application, the current output is not changed.

VI. Conclusion

This paper illustrates that it is possible to run an online model predictive controller on a Programmable Logic Controller. Two online QP algorithms are tested to solve a Model Predictive Control (MPC) problem. Both algorithms successfully control a temperature along a reference trajectory. The Hildreth algorithm is able to present a solution in a shorter time than the qpOASES algorithm. However, the latter has a much higher performance. Hence, it depends on the specific application which QP solver is preferred.

Acknowledgements

Research supported by Research Council KUL: OT/10/035, GOA/11/05 Ambiorics, GOA/10/09 MaNet , PFV/10/002 (OPTEC), IOF-SCORES4CHEM, several PhD/postdoc & fellow grants; Flemish Government: FWO: PhD/postdoc grants, projects: G0226.06 (cooperative systems and optimization), G0321.06 (Tensors), G.0302.07 (SVM/Kernel), G.0320.08 (convex MPC), G.0558.08 (Robust MHE), G.0557.08 (Glycemia2), G.0588.09 (Brain-machine) research communities (WOG: ICCoS, ANMMM, MLDM); G.0377.09 (Mechatronics MPC) IWT: PhD Grants, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, SBO POM, O&O-Dsquare Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007-2011) ; IBBT EU: ERNSI; FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, FP7-EMBOCON (ICT-248940), FP7-SADCO (MC ITN-264735), ERC HIGHWIND (259 166), Interreg IVa-7-022-BE i-MOCCA; Contract Research: AMINAL Other: Helmholtz: viCERP ACCM The scientific responsibility is assumed by its authors. BDM is a full professor at the KU Leuven, Belgium. JVI holds the chair Safety Engineering sponsored by the Belgian chemistry and life sciences federation essenscia. The fourth author holds a PhD fellowship of the Research Foundation – Flanders (FWO). We thank Andreas Potschka for providing a simplified Matlab implementation of qpOASES.

References

[1] G. Valencia-Palomo and J. Rossiter, "Efficient suboptimal parametric solutions to predictive control for plc applications," *Control Engineering Practice*, vol. 19, pp. 732–743, 2011.

[2] G. Valencia-Palomo, K. Hilton, and J. Rossiter, "Predictive control implementation in a PLC using the IEC 1131.3 programming standard," in *Proceedings of The European Control Conference 2009*, August 23-26, 2009, Budapest, Hungary 2009.

[3] M. Kvasnica, I. Rauova, and M. Fikar, "Automatic code generation for real-time implementation of model predictive control," in *2010 IEEE International Symposium on Computer-Aided Control System Design, Yokohama, Japan, September 8-10, 2010*, 2010.

[4] G. Pannocchia, J. B. Rawlings, and S. J. Wright, "Fast, large-scale model predictive control by partial enumeration," *Automatica*, vol. 43, no. 5, pp. 852 – 860, 2007.

[5] C. Hildreth, "A quadratic programming procedure," *Naval Research Logistics Quarterly*, vol. 4, p. 7985, 1957.

[6] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.

[7] J. Maciejowski, *Predictive Control With Constraints*. Pearson Education Limited, 2002.

[8] E. F. Camacho and C. Bordons, *Model Predictive Control*. Springer, 2003.

[9] L. Wang, *Model Predictive Control System Design and Implementation Using MATLAB*. Springer-Verlag London Limited, 2009.

[10] L. Ljung, *System Identification Toolbox Users Guide*. Natick: The MathWorks, Inc, 2009.

[11] B. Huyck, F. Logist, J. D. Brabanter, J. Van Impe, and B. De Moor, "Constrained model predictive control on a programmable automation system exploiting code generation: practical considerations," in *18th World Congress of the International Federation of Automatic Control, Milano (Italy)*, August 28 - September 2 2011.

[12] D. G. Luenberger, *Optimization by vector space methods*. John Wiley & Sons, 1997.