Constrained Model Predictive Control on a Programmable Automation System Exploiting Code Generation: Practical Considerations

B. Huyck [*,**,***] **F. Logist** [**] **J. De Brabanter** [*,***]
J. Van Impe [**] **B. De Moor** [***]

[*] *Department of Industrial Engineering, KaHo Sint Lieven, Gent, Belgium (e-mail: bart.huyck@esat.kuleuven.be)*
[**] *Department of Chemical Engineering (CIT - BioTeC), K.U.Leuven, Leuven, Belgium*
[***] *Department of Electrical Engineering (ESAT - SCD), K.U.Leuven, Leuven, Belgium*

Abstract: This paper examines the use of code generators for Model Predictive Control (MPC) on a Programmable Automation Controller (PAC). Recently, a strong interest has been observed in code generators able to produce dedicated code for solving optimization problems on embedded devices. This interest has consecutively raised questions about usability, flexibility and user-friendliness of these code generators in more standard industrial hardware or lab equipment. To this end, the current paper investigates as a benchmark example the constrained model predictive control of a real-life Two Input - Single Output hair dryer system that is controlled with a CompactRIO controller (National Instruments). The aim is to provide practical information throughout the description of the entire development process (i.e., from model identification over code generation to MPC implementation and validation). Hence, advantages as well as possible limitations and pitfalls are explored.

Keywords: System identification; Linear MISO; Automatic Process Control; Model-based Control; code generator

1. INTRODUCTION

Given the growing processor power of current micro controllers, much research is performed on the development of techniques capable of solving optimization problems on these types of devices rapidly and efficiently. To this end, different strategies have been followed. The first and classic approach involves the adaptation of currently available fast optimization algorithms in a language that can be understood by compilers of micro controllers. In the case of Model Predictive Control (MPC), this typically relates to Quadratic Program (QP) solvers. With respect to the languages, not only common languages as C or C++ (Ferreau et al., 2008) but also specific languages as VHDL for FPGA devices (Currie and Wilson, 2010) may be required depending on the controller type. The correct adaptation is generally a rather fast process but creates computational overhead. Another approach is not only to adapt, but also to rewrite the code in order to remove this overhead as much as possible. The elimination of multiplications by zero matrix-entries, for instance, saves a lot of runtime for a processor. It should, however, be noted that the removal of these unnecessary calculations has to be done for each (re)formulation of a specific problem. As this process is time-consuming and repetitive, it has to be automated with code generators (e.g., CVXGEN (Mattingley et al., 2010) and ACADO (Houska et al., 2010)). Code generation claims to have reached the kHz range for solving optimization problems on standard PCs. However, when computation speed is less critical, these codes can be used in an equally efficient manner on computationally less powerful industry standard controlling hardware, e.g., Programmable Logic Controllers (PLCs) or Programmable Automation Controllers (PACs), to control industrial processes in practice.

Constrained Model Predictive Control (MPC) is in fact an optimization problem. The calculated inputs are the result of a Quadratic Programming problem (Maciejowski, 2002; Camacho and Bordons, 2003). Recent contributions to the field discussed, e.g., the unconstrained MPC of Single Input - Single Output (SISO) systems on a PLC without the use of code generation (Valencia-Palomo et al., 2009). The current paper, however, investigates the use of the code generator CVXGEN (Mattingley et al., 2010) for constrained MPC on a real life Multiple-Input Single-Output (MISO) hairdryer set-up using a CompactRIO (National Instruments) controller. The aim is to provide practical information throughout the description of the entire development process (i.e., from model identification over code generation to MPC implementation and validation) and to discuss current advantages and limitations. In addition,

the MPC created with CVXGEN is also compared to the built-in LabVIEW MPC controller.

This paper is structured as follows. The next section briefly reviews the basics of MPC. Section 3 describes the experimental set-up and hardware used. Section 4 presents the model identification procedure and consecutively Section 5 describes and discusses the MPC implementation. Finally, Section 6 summarizes the main conclusions.

2. MODEL PREDICTIVE CONTROL AS OPTIMIZATION PROBLEM

Linear model predictive control is well known and investigated in depth in literature (Maciejowski, 2002; Camacho and Bordons, 2003) 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 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 to find 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:

$$\begin{aligned} J = &\sum_{i=1}^{H_p} \|\mathbf{y}(k+i|k) - \mathbf{r}(k+i|k)\|_Q^2 \\ &+ \sum_{i=1}^{H_c} \|\mathbf{u}(k+i|k)\|_R^2 , \end{aligned} \tag{2}$$

given the constraints for the input: $\mathbf{u}_{Min} \leq \mathbf{u} \leq \mathbf{u}_{Max}$. H_p and H_c are respectively the prediction and control horizon of the controller. $H_c \leq H_p$ is assumed. $Q \in \mathbb{R}^{mxm}$ and $R \in \mathbb{R}^{pxp}$ are positive definite weighting matrices. The notation $\|\mathbf{x}\|_S^2$ corresponds to $\mathbf{x}^T S\mathbf{x}$ for $\mathbf{x} \in \mathbb{R}^n$ and $S > 0 \in \mathbb{R}^{nxn}$. Output constraints are not taken into account in this paper. The formulation $\mathbf{y}(k+i|k)$ represents the vector $\mathbf{y}$ on time $k+i$ at calculation time k. Hence, the optimization problem can be formulated as:

$$\begin{aligned} \text{Minimize } J = &\sum_{i=1}^{H_p} \|\mathbf{y}(k+i|k) - \mathbf{r}(k+i|k)\|_Q^2 \\ &+ \sum_{i=1}^{H_c} \|\mathbf{u}(k+i|k)\|_R^2 \end{aligned} \tag{3}$$

$$\text{Subject to} \quad \mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k)$$

$$\mathbf{y}(k) = C\mathbf{x}(k) \tag{4}$$

$$\mathbf{u}(i) < \mathbf{u}_{Max} \tag{5}$$

$$\mathbf{u}(i) > \mathbf{u}_{Min} \tag{6}$$

This formulation of the problem can now be offered in an appropriate form to the code generator. The generator will then provide code that calculates the minimized input vector $\mathbf{u}$ where only the first input has to be applied to the system.

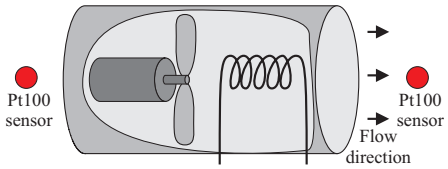


Fig. 1. Schematic overview of the hair dryer set-up.

3. HARD- AND SOFTWARE

3.1 Hardware

The set-up consists of a hairdryer whose connections to the resistor and the fan are made accessible. The fan is driven by a 24V DC motor and the resistor has a maximum power of 1400 W taken out of the 230V AC grid. The heating power delivered by the resistor can be adapted by a solid state relais with analog control (Gefran GTT 25A 480VAC - analog control voltage 0-10 V). The fan is controlled by a DC brushed Servo Drive (National Instruments NI9505 module). A National Instruments CompactRio controller system is built with the following elements: a NI cRio– 9024 Real-Time Controller, a cRIO–9114 Reconfigurable Chassis, a NI 9265 Analog Current Output module, a NI 9505 DC brushed Servo Drive module and a NI 9217 RTD 24-Bit Analog Input Module. Temperature sensors measure the environmental temperature and the temperature of the heated air as indicated in Fig. 1. Both sensors types are PT100 class B.

3.2 Software

The CompactRio is programmed with LabVIEW 2010 (Austin, Texas). This graphical language is suited for rapid application development. The MPC optimization problem is formulated in CVXGEN (http://cvxgen.com/) and the code generator provides a number of files written in plain C-code. The produced C-files are adapted slightly to make a library of this code. This is an easy way to call C-code from within the graphical programming language LabVIEW. The NI cRio– 9024 Real-Time Controller is equipped with the VxWorks (Wind River Systems, California) real-time operating system. A *gcc*-compiler is available to compile the library function for the VxWorks real-time operation system.

4. MODEL IDENTIFICATATION

MPC requires a model. Although physical modeling can be executed rather easily for this set-up, this paper uses an identified model of the hairdryer set-up. The aim is to create a two input (fan speed and resistor power) and single output (temperature) model. For this purpose, the System Identification Toolbox (Ljung, 2009) from Matlab is used. Two parametric model classes are used: (*i*) continuous transfer function models, where only first-order models are fitted and, as a state-space representation is recommended, (*ii*) a subspace state-space identification (Van Overschee and De Moor, 1996).

4.1 Experiment

The excitation signal is a multisine where both inputs use different frequencies within the range 0.05 - 0.00125 Hz. An experiment of length 500 seconds is repeated 4 times. These 4 experiments are averaged to eliminate disturbances. After detrending and normalisation, this dataset, recorded at a sample rate of 0.1 s, is divided in an estimation and validation subset with each a length of 250 s.

4.2 Model Classes

Two model classes will be employed to construct a linear MISO black-box model for the hair dryer: a continuous time transfer function and a subspace discrete state space model. From first principles, the subsystems of the hairdryer are known to be of first-order. Consequently, a linear, low-order, continuous-time transfer function of the form:

$$G(s) = \frac{K_p}{1 + T_{p1}s} e^{-T_d s} \tag{7}$$

is fitted to the data. Afterwards, this model is discretized and converted to state-space and is called *PEM P1D*.

The state-space formulation of a linear system is

$$\begin{aligned} \mathbf{x}(kT + T) &= A\mathbf{x}(kT) + B\mathbf{u}(kT) + K\mathbf{e}(kT) \\ \mathbf{y}(kT) &= C\mathbf{x}(kT) + D\mathbf{u}(kT) + \mathbf{e}(kT) \end{aligned} \tag{8}$$

with parameter matrices A, B, C, D and K. The measurements are sampled at time instances $t = kT$, with $k = 1, 2, \ldots$. The parameters in the general formulation (8) are identified using the subspace identification method (Van Overschee and De Moor, 1996). The subspace identified model is called N4SID.

4.3 Validation

As model validation, the simulation of the dataset is compared with the data based on a fit measure defined as:

$$\text{fit} = 100\% \left(1 - \frac{\|\hat{y}(t) - y(t)\|_2}{\|y(t) - \bar{y}(t)\|_2}\right) \tag{9}$$

where $\hat{y}(t)$ is the simulated output, $y(t)$ the measured output and $\bar{y}(t)$ the mean of the measured output. A fit value of 100% means that the simulation is the same as the measured output. If the predicted value is the mean value, the result is 0%.

4.4 Identification results

In Fig. 2 the validation for the two model classes is displayed on the validation dataset. The models are simulated over the horizon of the validation experiment. The first-order model P1D reaches a Fit of 78%, while the first-order subspace state-space model ends at 67%. Based on this observation, the P1D model is selected.

In an industrial environment, set-point changes are common. To account for this factor, a new validation dataset containing only steps is recorded. The sequence displayed

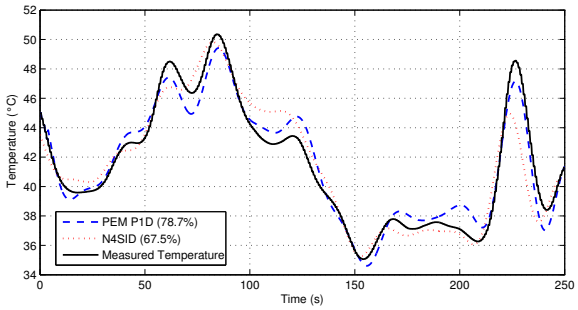
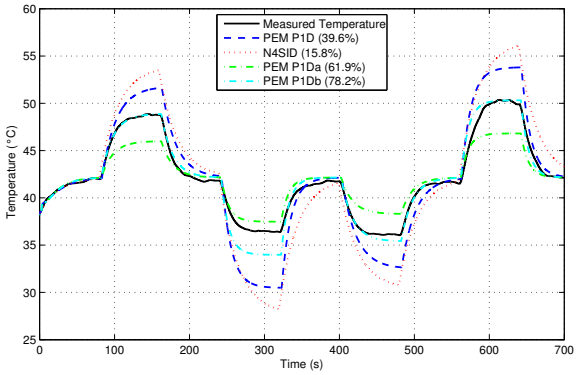
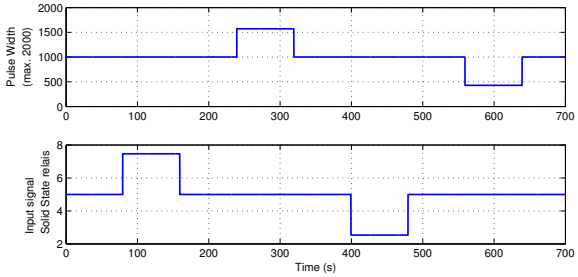


Fig. 2. Validation on multisine validation data. The selected model is the P1D is the first-order transfer function model.



(a) P1D is the identified model. For robustness, a too *small* model P1Da is created. The model implemented in the controller is P1Db. N4SID is the state-space model.



(b) Input signal for the above validation data

Fig. 3. Validation on validation data containing steps.

in Fig. 3 contains successively: (*i*) an increase of power, (*ii*) an increase of fan speed, (*iii*) a decrease of power and finally (*iv*) a decrease of fan speed. Based on this plot, it is clear that the gain of both models is too large. Though not so clearly, steps causing an increase of the temperature above the mean temperature do have a smaller difference then the steps causing a decrease of temperature. This clarifies that some nonlinear effects, e.g., heat losses, have to be taken into account when developing a robust controller for this set-up.

The identified model, employed for further development of the MPC controller, is the P1D model (10). Validation on step data (Fig. 3) demonstrates this model is *too large*. To ensure robustness two other models are created: model PEM P1Da with gain 0.4 times the gain of the original model and T_{p1} and T_d 0.8 times the original parameters which is *too small*. This model and the identified model are employed to determine the robust settings for the

controller. The implemented model is named PEM P1Db. The gain is 0.7 times the original PEM P1D gain.

The final identified P1D model is:

$$T_n = \begin{bmatrix} \frac{-1.33}{1+12.35s} e^{-3.59s} & \frac{1.1}{1+18.9s} e^{-3.74s} \end{bmatrix} \begin{bmatrix} u_{Fan,n} \\ u_{Power,n} \end{bmatrix} \quad (10)$$

where the index n indicates a normalised variable. $u_{Fan,n}$ and $u_{Power,n}$ are respectively the normalised and detrended Pulse Width Modulation (PWM) signal for the motor and the normalised and detrended voltage applied to the solid state relais of the resistor. Conversion to state space of the P1Db model results in a state space model of order 4. The discretisation interval is 2.5 s. This model is controllable, observable and stable.

4.5 Conclusion

A model is identified for the hairdryer. A transfer function model is selected as this is the best performing model, but also because this model class can be tuned by hand to fit step validation data closer.

5. MPC IMPLEMENTATION

The optimization problem formulated in Section 2 is introduced into the CVXGEN code generator available on the internet (http://cvxgen.com). Comparison of this controller with LabVIEW MPC, results and comments can be found in this section.

5.1 MPC controller design

Before code can be generated, some design decisions have to be made. First of all, a prediction and control horizon is selected. In literature (Valencia-Palomo and Rossiter, 2010; Shridhar and Cooper, 1998) it is recommended to take the prediction horizon longer than the settling time. With a settling time that is three time the longest time constant, the hairdryer needs a prediction horizon of minimum 62 s. Divided by a chosen discretization time of 2.5 s, the prediction horizon H_p has to be at least 25. Although this is preferred, the designers chose $H_p = 16$, this to meet the limit of a maximum 2000 non-zero Karush–Kuhn–Tucker (KKT) matrix entries forced by the CVXGEN code generator. Even though the limit of the KKT matrix entries might increase in the future, the limit of 2000 KKT matrix entries seems also a hard constraint for the compiler for the CompactRio device. Larger code results in compilation or runtime errors. Fortunately, simulation on computer shows no performance degradation between $H_p = 16$ and $H_p = 25$. Both controlled temperatures are the same even though the system model is different from the controller model. Despite there is no sign of performance degradation for this set-up, other systems with larger system matrices or a longer prediction horizon, will get into trouble if the limit is crossed. A control horizon of 7 time steps is selected.

The optimization problem formated in the CVX format is given below.

```
dimensions
  m = 2
  n = 4
  p = 1
  Hc = 7
  Hp = 16
end

parameters
  A (n,n)
  B (n,m)
  C (p,n)
  Q (p,p) psd
  R (m,m) psd
  x[0] (n)
  yref[t] (p), t=0..Hp
  S nonnegative  # slew rate limit.
  uMax nonnegative
  uMin nonnegative
end

variables
  x[t] (n), t=1..Hp+1
  u[t] (m), t=0..Hp
  y[t] (p), t=0..Hp
end

minimize
  sum[t=0..Hp](quad((y[t]-yref[t]), Q))
  + sum[t=0..Hc](quad(u[t], R))
subject to
  x[t+1] == A*x[t] + B*u[t], t=0..Hp
  y[t] == C*x[t], t=0..Hp
  u[t] < uMax, t=0..Hc
  u[t] > uMin, t=0..Hc
  u[t] == u[7], t = Hc+1..Hp
  norminf(u[t+1] - u[t]) <= S, t=0..Hc-1
end
```

5.2 Implementation

After code generation, the generated C-code files are compiled with *gcc* into an extern C-linkage library. This library is called from within a small LabVIEW program that scales the input- and output and exchanges the information with the correct physical in- and outputs to manipulate the fan and resistor. This program also contains a linear kalman filter to estimate the states of the black-box model. The implementation is rather easy, but be aware of the process to incorporate the code into your own program, as this takes more time than to generate appropriate code.

5.3 Experimental Results

The controller parameters are tuned in simulation based on the models P1D and P1Da. As this concerns a test set-up, no particular goals for the fan speed or resistor power are put forward. The penalties on both inputs are set equal which corresponds to $R \in \mathbb{R}^{2x2}$, $R = I$. Parameter $Q \in \mathbb{R}^{1x1}$ is adapted to fit the set-point closely. Simulation experiments have selected Q to be 15 to ensure a small steady state error without excessive overshoot. Fig. 4 displays two experiments on the set-up. The solid line is an experiment where the desired reference is followed by the CVXGEN MPC controller. The dashed line is the same reference followed by the LabVIEW MPC controller with the same settings. To verify the robustness of the controller, the initial start point of the temperature is 20°C, which is lower than the lowest temperature in the identification experiment. For the CVXGEN MPC, the temperature slowly evolves to the set-point but a steady state error of about 2°C below the set-point is noticed. This error is also observed during the temperature set-point decrease to 34°C. When changing the set-point to 50°C, an overshoot to 55°C is observed and a steady state error of about 2.5°C is seen. Similar results are

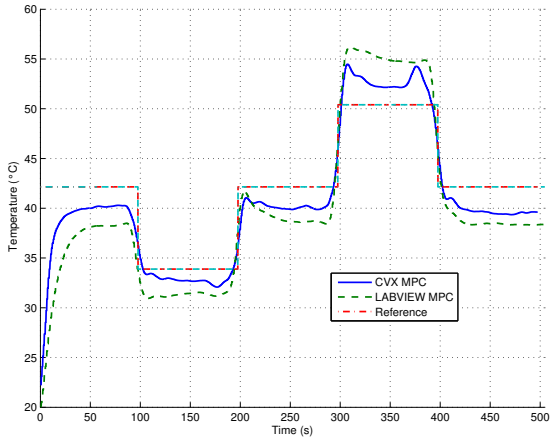


Fig. 4. An overview for the controlled temperature. The controller parameters Q and R are selected 15 and $R = I$ respectively.

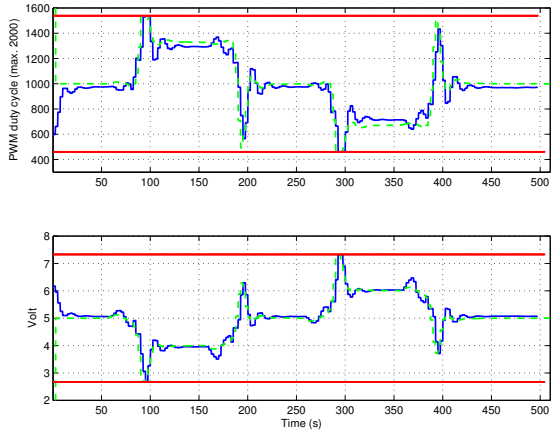


Fig. 5. An overview of the inputs. The controller parameters Q and R are selected 15 and $R = I$ respectively.

obtained for the LabVIEW MPC controller. Due to the different environmental conditions during the experiment, the reached steady state temperature is lower for the LabVIEW controller. The calculation time to solve the QP problem is between 12 and 18 ms for CVXGEN. LabVIEW is slighly faster with 7 to 10 ms for the calculation of a new input trajectory.

In Fig. 5 displays the inputs of both experiments. They stay within the region defined by the constraints and only at time 40 s and 120 s, the constraints are touched, but not violated. An important difference is the different input profile at the start of the experiment. The LabVIEW MPC keeps the inputs to their steady state value, although the measured temperature is far away from the desired temperature. Consequently, the temperature evolves slower to the set-point then possible. In fact, the set-point is never reached due to model mismatch. To eliminate the steady state error, the MPC problem needs to be reformulated with an additional integrator to create a so-called augmented system (see, i.e. Wang (2009)). The LabVIEW MPC works with an augmented model. Another possibility is to add an external integrator not incorporated into the model. This keeps the model, and hence, the QP problem smaller than with an incorporated integrator. This procedure is followed for the CVXGEN MPC. Although a code generator for MPC takes a lot of work out the hand of the developer, only a properly designed problem formulation can generate appropriate code.

5.4 Experimental results with integrator

In this work the procedure described by Maciejowski (2002), is implemented to eliminate the steady state error. The reference temperature is replaced by the reference corrected by the difference of the predicted and measured temperature:

$$y_{\text{ref,int}} = y_{\text{ref}} - (y_{\text{predicted}} - y_{\text{measured}}). \tag{11}$$

The result is displayed with a solid line in Fig. 6. Parameter Q is selected by simulation and set to 5 as the controller is less stable caused by the error correction. Experimental results in simulation, as well as on the hairdryer set-up resulted in an oscillatory behavior of the temperature with higher Q, e.g., $Q = 15$, which is of course undesired. The temperature at the start is the environmental temperature, which is situated outside the working area of the model. To reach the set-point temperature at the start, both controllers introduce a large, but similar, overshoot. For further set-point changes, only at the transition from 42°C to 50°C a small overshoot is observed. Both MPC controllers behave similar. There is an offset for each set-point temperature and for the temperatures higher then 40°C, small oscillations are observed for 50 s after a set-point change. The offset of the LabVIEW controller increases when the temperature decreases from 100–200 s. For all other temperatures the offset is more or less constant. The offset is never removed completely. For the CVX MPC controller, this can be explained by the use of Eq. 11. For the LabVIEW MPC controller the cause is not clear. Simulations make clear it is not the controller. Depending on the needed heating power, the resistor is switched ON and OFF during some periods the current sine wave. This way of heating might disturb the MPC controller an maintain the offset. The ripple seen after set-point changes, is caused by the difference between the controller model and the real system. The real temperature increase is faster than the decrease and the linear model is a *best fit* model. A small time shift is observed between both experiments. The peak at the start is shifted one controller step behind the CVX MPC controller. This is caused by the initialization of the controller as the former control output is not available at the start.

Fig. 7 plots the inputs for the experiment with integrator. For CVXGEN MPC, the results are very similar to the experiment without integrator: the controller saturates at the constraint for time 40 s and 120 s. In this experiment, the constraints are reached at the starting point of the experiment, causing the large overshoot in the beginning compared to overshoot further in the experiment. The inputs calculated by the LabVIEW MPC vary less when a set-point has the be kept. This is clearly seen at the timeslots between 100–200 s and 300–400 s.

5.5 Conclusions and remarks

Implementing a new algorithm is only useful when the new algorithm performs *better* than what is already available.

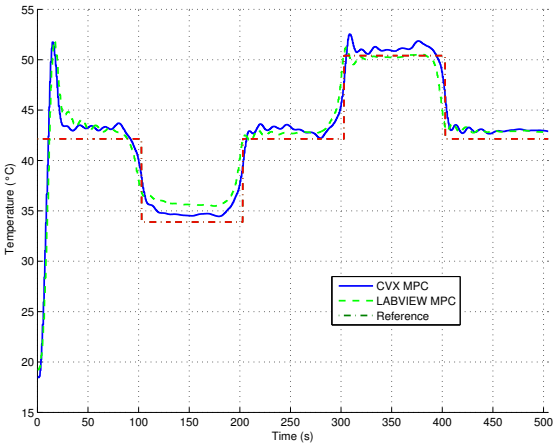


Fig. 6. An overview of the measured temperature with integration in the controller. The controller parameters Q and R are selected 5 and $R = I$ respectively.

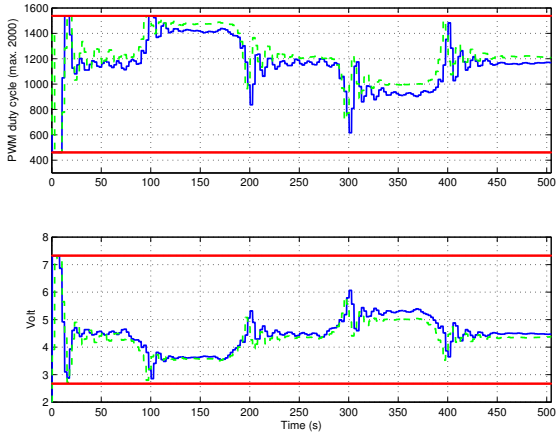


Fig. 7. An overview of the inputs of the Hairdryer with integration in the controller. The controller parameters Q and R are selected 5 and $R = I$ respectively.

Better is relative and depends of course on what is needed or wanted for a particular application. Code generators, e.g., CVXGEN and ACADO, are continuously updated and are equipped with the latest knowledge about fast QP solvers. Is code generation worth the effort? For CVXGEN, the limit of 2000 KKT matrix entries is restrictive, but the practitioner get a state-of-the-art controller. The code generation is fast and straightforward, but the practitioner has to be aware of the time consuming process of incorporating the code into his/her own program. As a code generator only provides code for the provided problem, additional problems, i.e., incorporation of the integrator in the controller for this problem, has to be done beforehand. The LabVIEW MPC is very easy to implement and well documented. A drawback is the complexity of this type of controller.

6. CONCLUSION

This paper discusses the use of a code generator on a programmable automation system. It has been demonstrated that the CVXGEN code generator delivers very reliable code that can be implemented on a programmable automation controller system. However, to generate useful code one needs to formulate the problem properly. The limit of 2000 KKT matrix entries posed by the CVXGEN code generator is restrictive, although this size should be sufficient, even for (slow) processes that need long prediction horizons. An advantage by using the code generator is the perfect knowledge of the implementation as this generation is close to the mathematical formulation. The LabVIEW MPC is fast, easy to implement, but its complexity makes it difficult to explain its behaviour.

REFERENCES

Camacho, E.F. and Bordons, C. (2003). *Model Predictive Control*. Springer.

Currie, J. and Wilson, D. (2010). Lightweight model predictive control intended for embedded applications. In *The 9th International Symposium on Dynamics and Control of Process Systems, Leuven, Belgium*.

Ferreau, H.J., Bock, H.G., and Diehl, M. (2008). An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8), 816–830.

Houska, B., Ferreau, H.J., and Diehl, M. (2010). ACADO toolkit - an open-source framework for automatic control and dynamic optimization. *Optimal Control Methods and Application*, accepted.

Ljung, L. (2009). *System Identification Toolbox Users Guide*. The MathWorks, Inc, Natick.

Maciejowski, J. (2002). *Predictive Control With Constraints*. Pearson Education Limited.

Mattingley, J., Wang, Y., and Boyd, S. (2010). Code generation for receding horizon control. In *Proceedings IEEE Multi-Conference on Systems and Control, Yokohama, Japan*, 985–992.

Shridhar, R. and Cooper, D.J. (1998). A novel tuning strategy for multivariable model predictive controle. *ISA Transactions*, 36(4), 273–280.

Valencia-Palomo, G., Hilton, K., and Rossiter, J. (2009). Predictive control implementation in a PLC using the IEC 1131.3 programming standard. In *Proceedings of The European Control Conference 2009*.

Valencia-Palomo, G. and Rossiter, J. (2010). Auto-tuned predictive control based on minimal plant information. In *Advanced Control of Chemical Processes*, volume 7.

Van Overschee, P. and De Moor, B. (1996). *Subspace Identification of Linear Systems: Theory, Implementation, Applications*. Kluwer Academic Publishers.

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

ACKNOWLEDGEMENTS

BDM is a full professor at the Katholieke Universiteit Leuven, Belgium. Research supported by: Research Council KUL: GOA/11/05 Ambiorics, GOA/10/09 MaNet , CoE EF/05/006 Optimization in Engineering (OPTEC) en PFV/10/002 (OPTEC), IOF-SCORES4CHEM, OT/09/025/TBA, OT/10/035; 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) ; EU: ERNSI; FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, FP7-EMBOCON (ICT-248940), FP7-SADCO (MC ITN-264735), ERC HIGHWIND (259 166). Contract Research: AMINAL; Other: Helmholtz: viCERP, ACCM.

JVI holds the chair Safety Engineering sponsored by the Belgian chemistry and life sciences federation essenscia.

The scientific responsibility is assumed by its authors.