Embedded Real-Time MPC implementation using Rapid Prototyping Tools: a Thermal Process case study

Gabriel Harja, Andres Hernandez and Robin De Keyser
Department of Electrical energy, Systems and Automation
Ghent University, Belgium
Email: Andres.Hernandez@UGent.be

Ioan Nascu
Department of Automation, Technical University of Cluj-Napoca
Romania
Email: gabrielharja@yahoo.com

***Abstract*—Recently, a strong interest has been observed in the development of new control strategies for solving optimization problems on embedded devices. Particularly, Model Predictive Control (MPC) became quite popular nowadays due to its multiple successful implementations in real-life industrial applications. The aim of this contribution is to provide practical information throughout the use of VeriStand and CompactRIO as Rapid Prototyping Tools (RPT) for implementing MPC algorithms from Matlab/Simulink code. The effectiveness of this tool is tested on a nonlinear thermal process with variable time delay, in which three different implementations of MPC to deal with both nonlinear dynamics and time delay were evaluated.**

I. Introduction

During recent years an extreme evolution in microelectronics and informatics technology was observed, making possible the development of high performance embedded systems. At the same time researchers developed each time more complex and advanced control algorithms which required a high computational demand as it is in many cases required to solve an optimization problem in real-time, at high accuracy and fast sample rate. Therefore, many of the scientific contributions remained just for academia as its implementation was still limited. In order to close this gap between active research and real-life implementation, some companies such as Texas Instruments, dSPACE, Mathworks and National Instruments (NI), just to mention a few ones, have developed Rapid Prototyping Tools (RPT) to help engineers to speed up during the process of implementing control algorithms in embedded systems. Some of the important characteristics these systems offer are: fast code generation, flexibility, reliable hardware, easy access to I/O channels, friendly user interface, datalogging and application sustainability; the last especially is needed, as nowadays, big projects require a multidisciplinary group working for a common objective.

One of the control methodologies which is gaining more attention from both industry and academia is Model Predictive Control (MPC). The reason is that MPC can relatively easily tackle multiple and strongly coupled input-output variables, it has inherent dead-time compensation, while introducing feedforward control action in a natural way, allowing the compensation of measured disturbances. Furthermore, it is possible to extend the strategy to constrained control problems (e.g. constraints on manipulated variable, controlled variable or variation rates of these). This promising control methodology, however, requires for its implementation a good knowledge of common languages as C or C++, as mentioned in [1] but also of specific ones as VHDL for FPGA devices [2]; the choice between the different languages depends on the target. Lacks in handling these languages are often obstacles in running the algorithm on a Real-Time (RT) Target, as a time consuming operation.

978-1-4799-2228-4/13/$31.00 © 2013 IEEE

Many processes include time delay phenomena in their inner dynamics, representative examples being found in biology, chemistry, mechanics, physics, population dynamics, as well as in engineering sciences. In addition, actuators, sensors, field networks that are involved in the feedback loops usually also introduce delays, which are possibly time-varying. Thus there is an increasing interest in studying time delay systems in all scientific areas, especially in control engineering [3], [4]. The presence of time delay (dead time) in the control loop is always a serious obstacle on the need to a good performance. Hence, time delay compensators have received a considerable amount of attention in the literature. The most widely implemented method in industry is the Smith Predictor (SP) [5] and its modifications [6]. The general shortcoming of these methodologies is that they assume a stable or integrating process with *constant* time delay. There are industrial processes involving transportation of material which is directly depending on the manipulated variable. Consequently, these processes possess an inherent *variable* time delay. The complexity is increased when considering that real processes are nonlinear and it requires a nonlinear optimization problem to be solved in real-time.

In this study a nonlinear thermal process with variable time delay is considered. The first reason is to provide practical information throughout the use of NI VeriStand as Rapid Prototyping Tools to easily implement the MPC algorithm directly from the Matlab/Simulink code to a RT Controller. The second reason is to provide insight about the benefits of using MPC to control processes with nonlinearities and variable time delay. The analysis is built as a comparison between a linear and a nonlinear MPC and between the MPC with variable cost horizon and the SP version.

This paper is organized as follows. Section II gives a brief description of the proposed MPC approaches that will be used. The process to be controlled is described in Section

Authorized licensed use limited to: University of Liege (ULg). Downloaded on January 19,2026 at 16:46:30 UTC from IEEE Xplore. Restrictions apply.

III. Next the hardware and software tools chosen for the Rapid Prototyping Process are presented also accompanied by some guidelines. In Section V real experiments are used to compare the performance of different control algorithms. Finally, Section VI summarizes the main conclusions.

II. Model Predictive Control

Model Predictive Control (MPC) techniques have been successfully implemented in both academia and industry. In [7] an introduction to theoretical and practical aspects of the most commonly used MPC strategies is provided. Among the different MPC methodologies available in literature, the Extended Prediction Self-Adaptive Control (EPSAC) proposed by [8] has been chosen as it presents interesting features, especially in its nonlinear version (NEPSAC). This sections offers a brief introduction of this method.

A. EPSAC algorithm

In the EPSAC algorithm, the model output $x(t)$ represents the effect of the control input $u(t)$ on the process output $y(t)$. It can be described by the following equation:

$$x(t) = f[x(t-1), x(t-2), \ldots, u(t-1), u(t-2), \ldots] \quad (1)$$

Notice that $x(t)$ represents here the model output, not the state vector. Also important is the fact that f can be either a linear or a nonlinear function. The generic model of the EPSAC algorithm is:

$$y(t) = x(t) + n(t) \quad (2)$$

where $y(t)$ is the measured output of the process, $x(t)$ is the model output and $n(t)$ represents model/process disturbance, all at discrete-time index t. The disturbance $n(t)$ can be modeled as coloured noise through a filter with the transfer function:

$$n(t) = \frac{C(q^{-1})}{D(q^{-1})} e(t) \quad (3)$$

with $e(t)$ uncorrelated (white) noise with zero-mean and C, D monic polynomials in the backward shift operator q^{-1}. The disturbance model must be designed to achieve robustness of the control loop against unmeasured disturbances and modeling errors. A 'default' choice to remove steady-state control offsets is $n(t) = \frac{1}{1-q^{-1}} e(t)$ [9].

A fundamental step in the MPC methodology consists of the prediction. Using the generic process model (2), the predicted values of the output are:

$$y(t+k|t) = x(t+k|t) + n(t+k|t) \quad (4)$$

for $k = N_1 \ldots N_2$, where N_1 and N_2 are the minimum and the maximum prediction horizons. The prediction of the process output is based on the measurements available at sampling instant t, $\{y(t), y(t-1), \ldots, u(t-1), u(t-2), \ldots\}$ and future (postulated) values of the input signal $\{u(t|t), u(t+1|t), \ldots\}$. The future response can then be expressed as:

$$y(t+k|t) = y_{base}(t+k|t) + y_{opt}(t+k|t) \quad (5)$$

The two contributing factors have the following origin:

- $y_{base}(t+k|t)$ is the effect of the past inputs $u(t-1), u(t-2) \ldots$, a future base control sequence $u_{base}(t+k|t)$ (which is pre-specified, ref. section II-B) and the predicted disturbance $n(t+k|t)$.

- $y_{opt}(t+k|t)$ is the effect of the optimizing control actions $\delta u(t|t), \ldots, \delta u(t+N_u-1|t)$ with $\delta u(t+k|t) = u(t+k|t) - u_{base}(t+k|t)$, in a control horizon N_u.

The optimized output can be expressed as the discrete-time convolution of the unit impulse response coefficients $h_1, \ldots, h_{N_2}$ and unit step response coefficients $g_1, \ldots, g_{N_2}$ of the system as follows:

$$\begin{aligned} y_{opt}(t+k|t) = h_k \delta u(t|t) + h_{k-1} \delta u(t+1|t) + \ldots \\ + g_{k-N_u+1} \delta u(t+N_u-1|t) \end{aligned} \quad (6)$$

Using (5) and (6), the key EPSAC-MPC formulation becomes:

$$\mathbf{Y} = \overline{\mathbf{Y}} + \mathbf{G}.\mathbf{U} \quad (7)$$

where:

$$\begin{aligned} \mathbf{Y} &= [y(t+N_1|t) \ldots y(t+N_2|t)]^T \\ \overline{\mathbf{Y}} &= [y_{base}(t+N_1|t) \ldots y_{base}(t+N_2|t)]^T \\ \mathbf{U} &= [\delta u(t|t) \ldots \delta u(t+N_u-1|t)]^T \end{aligned} \quad (8)$$

$$\mathbf{G} = \begin{bmatrix} h_{N_1} & h_{N_1-1} & \ldots & g_{N_1-N_u+1} \\ h_{N_1+1} & h_{N_1} & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ h_{N_2} & h_{N_2-1} & \ldots & g_{N_2-N_u+1} \end{bmatrix} \quad (9)$$

Once the output is predicted, it is possible to optimize the control signal $\mathbf{U}$ by minimizing the cost function:

$$V(\mathbf{U}) = \sum_{k=N_1}^{N_2} [r(t+k|t) - y(t+k|t)]^2 \quad (10)$$

Notice that the controller cost function (10) can be easily extended to many alternative cost functions (similar to the approach in optimal control theory) as described in [8]. The *horizons* N_1, N_2 are design parameters and $r(t+k|t)$ is the desired *reference trajectory*, chosen here as a 1^{st}-order reference trajectory as specific implementation example:

$$r(t+k|t) = \alpha\, r(t+k-1|t) + (1-\alpha) w(t+k|t) \quad (11)$$

for $k = 1 \ldots N_2$ and initialization $r(t|t) = y(t)$. The signal $w(t)$ represents the setpoint and α a design parameter that plays an important role in tuning the MPC performance [10]. The optimal input solution of the EPSAC algorithm can be written in matrix form:

$$\mathbf{U}^* = [\mathbf{G^T G}]^{-1} [\mathbf{G^T} (\mathbf{R} - \overline{\mathbf{Y}})] \quad (12)$$

with R being the vector notation of the reference trajectory, $\mathbf{R} = [r(t+N_1|t) \ldots r(t+N_2|t)]^T$ and $[G^T G]$ of dimension N_uxN_u. Only the first optimal control input $u(t) = u_{base}(t/t) + \delta u(t/t)$ is applied to the plant and the whole procedure is repeated again at the next sampling instant $t+1$ (receding horizon strategy).

Authorized licensed use limited to: University of Liege (ULg). Downloaded on January 19,2026 at 16:46:30 UTC from IEEE Xplore. Restrictions apply.

B. NEPSAC algorithm

The calculation of the predicted output with (5) involves the superposition principle. When a nonlinear system model $f[.]$ is used in (1), above strategy is only valid -from a practical point of view - if the term $y_{opt}(t+k|t)$ in (5) is small enough compared to the term $y_{base}(t+k|t)$. When this term would be zero, the superposition principle would no longer be involved. The term $y_{opt}(t+k|t)$ will be small if $\delta u(t+k|t)$ is small, see (6).

This can be realized iteratively, by executing the following steps at each controller sampling instant:

1) Initialize $u_{base}(t+k|t)$ as: $u^1_{base}(t+k|t) = u^*(t+k|t-1)$, i.e. the optimal control sequence as computed during the previous sampling instant; in other words: $u^*(t+k|t-1)$ is used as a first estimate for $u^*(t+k|t)$.
2) Calculate $\delta u^1(t+k|t)$ using the linear EPSAC algorithm.
3) Calculate the corresponding $y^1_{opt}(t+k|t)$ with (6) and compare it to $y^1_{base}(t+k|t)$, which is the result of $u^1_{base}(t+k|t)$.
4) In case $y^1_{opt}(t+k|t)$ is *not* small enough compared to $y^1_{base}(t+k|t)$: re-define $u_{base}(t+k|t)$ as $u^2_{base}(t+k|t) = u^1_{base}(t+k|t) + \delta u^1(t+k|t)$ and go to 2. The underlying idea is that $u^1_{base}(t+k|t) + \delta u^1(t+k|t)$ - which is the optimal $u^*(t+k|t)$ for a linear system - can act as a second estimate for the optimal $u^*(t+k|t)$ in case of a nonlinear system.
5) In case $y^i_{opt}(t+k|t)$ is small enough compared to $y^i_{base}(t+k|t)$: use $u(t) = u^i_{base}(t+k|t) + \delta u^i(t+k|t)$ as the resulting control action of the current sampling instant (notice that $i = 1, 2, \ldots,$) according to the number of iterations)

This algorithm results after convergence to the optimal solution for the underlying nonlinear predictive control problem. The number of required iterations depends on how far the optimal $u^*(t+k|t)$ is away with respect to $u^*(t+k|t-1)$. In quasi-steady-state situations, the number of iterations is low $(1 \ldots 2)$. On the other hand, during transients the number of iterations might raise to 10.

III. Thermal Process

A. Process description

The process considered in this paper consists of a heated tank of which the level is controlled by a mechanical float switch, resulting in a constant water volume V_{tank} (Fig. 1). A submerged electrical heater delivers a *constant* heat flow Q, which causes the liquid to warm up. Temperature control of the outlet water is achieved by changing the outflow $q(t)$ of hot tank water, which allows an equal amount of cold tap water to flow in. A variable transport delay, which depends on the outflow $q(t)$ (the manipulated variable), is introduced in the system by measuring the temperature of the effluent stream at a distance L from the tank.

The experimental setup - based on ideas presented in [11] and on the operation of a solar collector field [12]- has been designed and implemented at Ghent University [13], [14]. It has a tank volume $V_{tank} = 0.9$ l, a heat input $Q = 2000$ W and

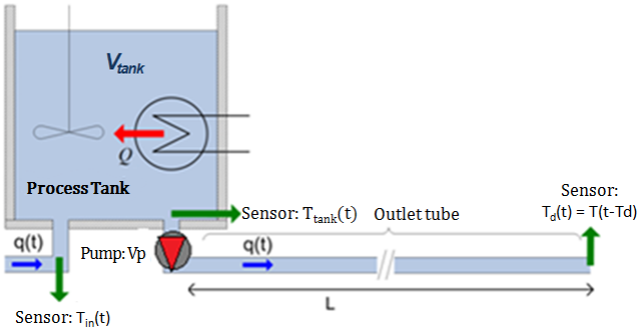

Fig. 1. Representation of the variable time delay process

an outlet tube length $L = 16$ m (tube volume $V_{tube} = 1.65$ l). The flow range is $0.0075 \leq q \leq 0.0487$ l/s, or equivalently, $0.45 \leq q \leq 2.9$ l/min.

It is important to notice that by properly choosing the length of the tube, it is possible to manipulate the value of the time delay according to:

$$T_d(t) = \frac{LS}{q(t)} = \frac{V_{tube}}{q(t)} \tag{13}$$

where S denotes the cross-section of the tube. Therefore, the time delay can be chosen to be bigger, equal or smaller than the time constant of the system τ, which is described by the relationship:

$$\tau(t) = \frac{V_{tank}}{q(t)} \tag{14}$$

Notice how an extra difficulty is introduced in the control problem: we are facing a variable time delay (13) but also a variable time constant (14) in the system as function of the manipulated variable $q(t)$. This effect is illustrated in Fig. 2 by the responses of a staircase experiment when $V_{tank} = 0.9$ l and $V_{tube} = 1.65$ l (long time delay). Although the steps applied to the flow are of the same magnitude, these have a different effect on the temperature depending on the current state of the system. As an example, it is observed a difference of the temperature change and the time constant between the steps applied at 400 and 2800 seconds. During the first step temperature decreases $16°C$ in 235 seconds while during the second one it decreases only $1.3°C$ in 79 seconds.

Accurate flow control is achieved using a peristaltic pump driven by a 24 V DC-motor, that provides linearity between control voltage and flow. Finally, three Pt100 sensors measure the temperature of the: (1) hot effluent water $T_{out}(t)$, (2) cold incoming tap water $T_{in}(t)$ and (3) tank $T_{tank}(t)$. Only T_{out} is used for identification and control purposes; the other measurements are used for the analysis of the results.

B. Process modelling

In this contribution the process is modeled as the series connection of the tank model and tube model, which consists of process dynamics and time delay. Physical modeling is used to determine the tank model, while the tube dynamics are identified experimentally.

Authorized licensed use limited to: University of Liege (ULg). Downloaded on January 19,2026 at 16:46:30 UTC from IEEE Xplore. Restrictions apply.

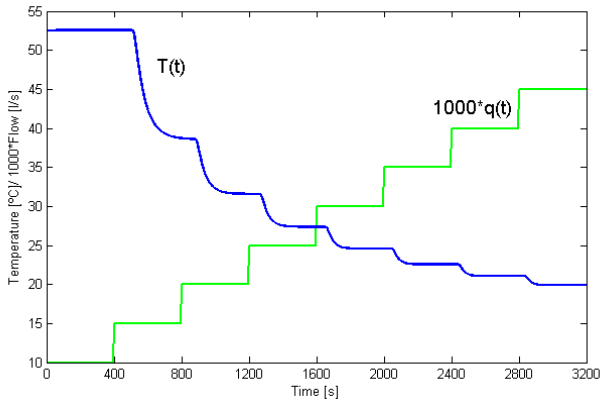

Fig. 2. Staircase experiment over the full range of the system.

The mathematical model that describes the relationship between the outflow $q(t)$ and the tank temperature $T_{tank}(t)$ follows from the energy balance equation:

$$\rho\, c_p\, V_{tank} \frac{dT_{tank}(t)}{dt} = Q + \rho\, c_p\, q(t)\,(T_{in}(t) - T_{tank}(t)) \quad (15)$$

where ρ and c_p are respectively the density and the specific heat of water, V_{tank} is the volume in the tank and Q is the constant amount of supplied heat. The small heat losses to the environment are negligible. Notice that the model is nonlinear, as $q(t)$ is the control input (manipulated variable).

Experimentally, a first order transfer function in the Laplace domain s has been fitted to the tube dynamics:

$$\frac{T(s)}{T_{tank}(s)} = \frac{K_t}{\tau_t\, s + 1} \quad (16)$$

where $T(t)$ is a virtual signal which is related to the outlet temperature by

$$T_{out}(t) = T(t - T_d(t)) \quad (17)$$

and $T_d(t)$ represents the variable time delay.

From identification experiments, reasonable results have been obtained with a constant gain $K_t = 0.99$ (small heat loss) and time constant $\tau_t = 11.1$ sec.

The temperature sensor is located in the outlet tube at a distance L from the tank. Due to the variable flow, the time the water needs to travel from the tank to the sensor varies as well and is given by

$$\int_{t-T_d(t)}^{t} q(\tau)\, \mathrm{d}\tau = \mathrm{LS} \quad (18)$$

with L the length and S the cross-section of the tube. From a physical point of view, this is equivalent to filling the tube at varying flow rates. Discretizing (18) allows the calculation of the discrete-time variable delay N_d:

$$\sum_{i=1}^{N_d} q(t-i) = \frac{LS}{T_s} \quad (19)$$

where t now denotes the discrete time index and T_s the sampling period. Consequently, at each sampling instant, N_d is calculated as the number of flow samples to be summed before the total sum exceeds $\frac{LS}{T_s}$. This means that the variable time delay depends on flows that have been applied in the past $[q(t-1), q(t-2), \ldots, q(t-N_d)]$.

IV. Rapid Control Prototyping Tools

A. Hardware

A system that runs the control algorithm loop in Real-Time is required. Also considering a good and easy connection with the selected software, described in IV-B, a CompactRIO controller from National Instruments® is used, consisting of the following elements: a NI cRio-9074 Integrated 400 MHz Real-Time Controller, a cRio-9114 reconfigurable chassis, a NI 9201 12-Bit Analog Voltage Input module and a NI 9263 16-Bit Analog Voltage Output module. Analog Input module is used to get data from the sensors and through the Analog Output module the input voltage of the pump is manipulated. The communication to the target is done by means of ethernet connection, making possible to access the controller remotely.

B. Software

In the present paper the Real-Time Controller is programmed with NI VeriStand 2012.[1] VeriStand is a configuration-based testing software, allowing to develop, prototype, and test control systems using hardware I/O and simulation models. Considering their popularity in both industrial and especially educational environment, MathWorks® products Matlab and Simulink are used to design, and simulate the control algorithms.

The process of deploying the VeriStand project in the target, starts by integrating the m-file code into an embedded Matlab function in Simulink. In order to read and write values to external blocks or functions, *In* and *Out* Simulink blocks must be added e.g. to read the setpoint and sensor value and to apply the control action. Thus, resulting in the scheme depicted in Fig. 3.

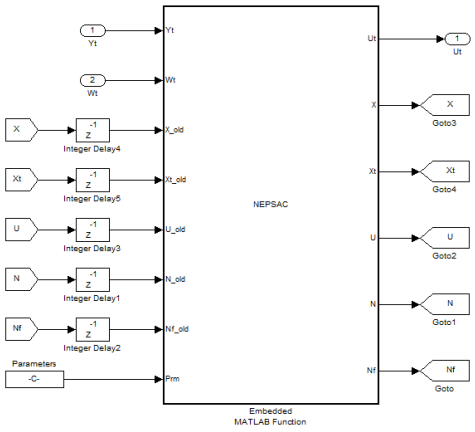

Fig. 3. Embedded Matlab Function Block.

In the MPC approach values of the previous inputs and outputs must be stored in the memory, and be shifted each

[1] http://www.ni.com/veristand/

Authorized licensed use limited to: University of Liege (ULg). Downloaded on January 19,2026 at 16:46:30 UTC from IEEE Xplore. Restrictions apply.

sampling time. This is achieved with the *GoTo* and *From* blocks, and then adding a delay of one sample with the *Integer Delay* block. Also some parameters needed in the Embedded Matlab Function must be given as parameters of the block, for example the G matrix necessary in computing the optimal input solution in the linear case of MPC.

When installing VeriStand also a NI VeriStand Server is installed. This is linked with Matlab and is used to transform the Simulink model into a appropriate format to be then used in the VeriStand project. A C compiler is necessary in this process; the conversion can be done by using the options from Real-Time Workshop subsection of the Configuration Parameters menu from Simulink. VeriStand offers a user friendly framework that allows to easily integrate the resulted model of the controller and link the I/O to the real target ones in a simple and intuitive way using the ScanEngine custom device.

In addition to VeriStand there are other useful tools that provide the possibility to interact in Real-Time with the cRio Controller, plot signals, log data, give a signal of specified profile as the input - Stimulus Profile Editor and also to check the CPU and Memory usage - Distributed System Manager (DSM). The implementation flow from Simulink model to deployment on the cRio controller is depicted in Fig. 4.

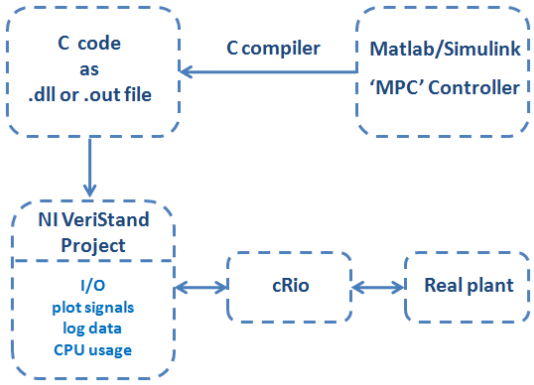

Fig. 4. Implementation flow diagram.

V. Exploring MPC Implementation

Having the necessary tools presented in Section IV it is now possible to test these control algorithms not only in simulation but also on a real target. Once the project is deployed, it is rather simple to modify the functionality of your controller as it is based on Matlab code.

In this section two cases are analized, the first regarding the performance difference between linear and nonlinear MPC (EPSAC and NEPSAC) and second related to MPC strategies to deal with systems with variable time delay, comparing the MPC with variable cost horizon to MPC with SP version.

A. EPSAC vs NEPSAC

Extended Prediction Self-Adaptive Control (EPSAC) and Nonlinear EPSAC algorithms were presented in Section II. Both require a discrete-time model of the system suitable for prediction. The first bases its prediction on the model (20), the linearization of (15), (16) and (19) around an operating point $[q^* = 0.0231/\text{s},\ T^*_{out} = 34.4\,^\circ\text{C}]$; while the second one predicts based on the nonlinear model.

$$\frac{T_{out}(s)}{q(s)} = \frac{-894}{(39.12s+1)(11.1s+1)}e^{-72s} \tag{20}$$

Both EPSAC and NEPSAC algorithms are implemented using a Smith Predictor structure. The controllers have been tuned using the following parameters: coincidence horizon $(N_1...N_2 = 20)$, control horizon $(N_u = 1)$ and $\alpha = 0.7$. The performance of the two algorithms resulted from the tests are presented in Fig. 5. As it can be seen from the plotted results both controllers perform the same near the operating point but as the setpoint departs from it, the nonlinear approach keeps the same response while the performance of the linear approach decreases. Also regarding the CPU usage it cannot be seen much difference between these two implementations. The CPU test was performed using the DSM from VeriStand.

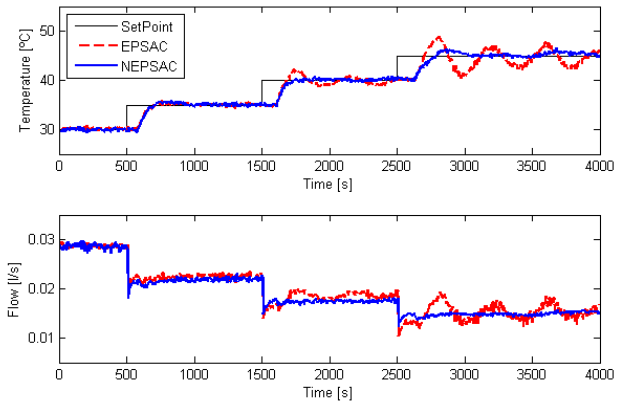

Fig. 5. Results of the EPSAC - NEPSAC comparison.

B. Variable Cost Horizons vs Smith Predictor

For a system with time delay, changes in the controlled variable are noticeable once the time delay has passed. Therefore, in order to find the optimal control sequence only output predictions occurring after the time delay should be taken in the cost function. This means that the minimum costing horizon N_1 should be equal to the time delay plus one. Notice that for systems with constant time delay the previous will lead to constant $N_1...N_2$ parameters. For a variable time delay however, the value of N_1 (and thus also N_2) varies with the time delay index. This strategy of MPC with variable cost horizon will be referred to as MPC_{VCH}.

The second strategy exploits the structure of the process model to design a predictive controller with constant design parameters (N_1, N_2) as presented in more detail in [14]. The structure of this control strategy is depicted in Fig. 6.

The model of the process consists of the nonlinear tank dynamics and the tube dynamics, which are separated from the variable time delay model. At each sampling instant, the delay-free model output $x(t)$, resulting from the process dynamics only, is calculated using the stored values

Authorized licensed use limited to: University of Liege (ULg). Downloaded on January 19,2026 at 16:46:30 UTC from IEEE Xplore. Restrictions apply.

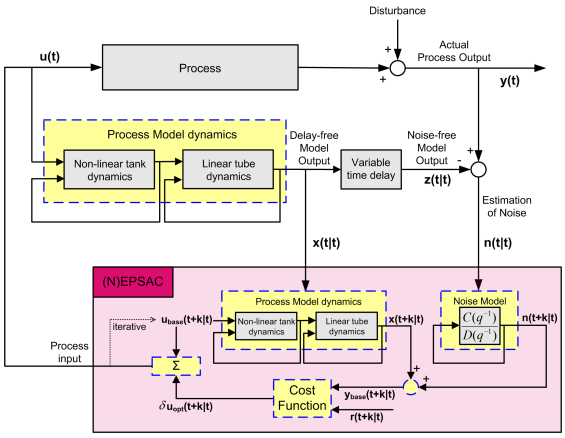

Fig. 6. NEPSAC control structure.

$[x(t-1),...,u(t-1),...]$. At the same sampling instant, the variable time delay is computed from (19). Once N_d is known, $x(t-Nd)$ can be selected out of the stored x-values, such that $z(t) = x(t-Nd)$.

In this way, the prediction procedure is thoroughly simplified, resulting in a *Smith Predictor-like* scheme, with separation of the tank and tube dynamics on one hand and the varying time delay on the other hand. In such approach the minimum prediction horizon is no longer varying and obviously equal to one; that gives a constant maximum prediction horizon. The strategy will be referred to as MPC_{SP}.

For both cases analyzed: Smith Predictor and variable cost horizons, the coincidence horizon was the same $(N_1...N_2 = 20)$, control horizon $N_u = 1$ and $\alpha = 0.7$. These parameters were chosen considering the thermal plant specifications and for a sampling time $T_s = 4$ seconds. Fig. 7 shows the results of the tests. It can be seen that controllers perform the same in the tested range.

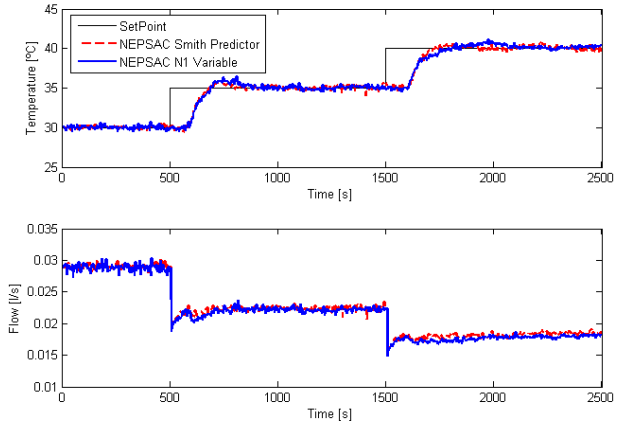

Fig. 7. Results of the NEPSAC N_1 Variable and Smith Predictor comparison.

In terms of CPU usage (tested with DSM) there was not noticeable difference between the two approaches. The only difference is from the implementation point of view. For the method with N_1 variable the prediction of the plant, noise and filtered noise must be performed also during the time delay. This requires longer vectors and it is more sensitive to programming errors unlike the Smith Predictor implementation.

VI. Conclusion

This paper discussed the use of VeriStand and CompactRIO as Rapid Prototyping Tools to easily deploy complex control algorithms directly from Matlab/Simulink code on a real-time target. There were presented three approaches of the MPC to control a nonlinear thermal process with variable time delay. Based on this implementation it was demonstrated that implementing a MPC with variable cost horizon lead to the same as the Smith Predictor structure although it requires a more careful implementation. It was also shown the effectiveness of the NEPSAC to control the system compared to its linear version EPSAC, without incurring in a higher computational demand. In order to better compare the algorithms, a further deeper study regarding the CPU and Memory performance has to be done.

References

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

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

[3] Richard, J.P.,(2003). Time-delay systems: an overview of some recent advances and open problems, *Automatica*, 39, pp. 1667–1694.

[4] Normey-Rico, J., and Camacho, E., (2007). Control of Dead-time processes, *Springer*, Berlin.

[5] Smith, O.J.M., (1957). Closer control of loops with dead time, *Chemical Engineering Progress*, 53, pp. 217–219.

[6] Palmor, Z.J., (1996). Time-delay compensation - Smith predictor and its modifications, in: *W.S. Levine (Eds.), The control handbook, Boca Raton*, pp. 224–237.

[7] E.F. Camacho and C., Bordons, (2007) Model Predictive Control. ISBN-13: 978-1852336943, *Springer*; 2nd ed.

[8] R. De Keyser. A 'Gent'le approach to predictive control, *Invited Chapter in UNESCO Encyclopaedia of Life Support Systems (EoLSS)*. Article contribution 6.43.16.1, Eolss Publishers Co Ltd, Oxford, ISBN 0 9542 989 18-26-34 (www.eolss.net), 30p.

[9] J.M. Maciejowski (2002) Predictive Control with Constraints *Pearson Education Limited*, Edinburgh.

[10] M. Sanchez and J. Rodellar (1996) Adaptive Predictive Control ISBN 0135148618, *Prentice Hall London*

[11] Cristea, S. De Prada, C., De Keyser, R., (2005). Predictive control of a process with variable dead-time, in: *Proceedings of the 16th IFAC World Congress*, Prague, Czeck Republic, 2005, CD-ROM.

[12] Normey-Rico, J., Bordons, C., Berenguel, M., and Camacho, E., (1998). A robust adaptive dead-time compensator with application to a solar collector field, in: *Proceedings of the IFAC-Workshop on Linear Time Delay Systems (LDTS'98)*, Grenoble, France, pp. 105–110.

[13] Himpe, S., and Theunynck, V., (2006). Design and Advanced Control of a Process with Variable Time Delay, Master thesis Ghent University, Department of Electrical Energy, Systems and Automation.

[14] Sbarciog, M., De Keyser, R., Cristea, S., and De Prada, C., (2008). Nonlinear predictive control of processes with variable time delay: a temperature control case study, *IEEE - Multi Conference on Systems and Control*, San Antonio USA, ISSN 978-1-4244-2223-4/08, pp. 1001–1006

Authorized licensed use limited to: University of Liege (ULg). Downloaded on January 19,2026 at 16:46:30 UTC from IEEE Xplore. Restrictions apply.