PREDICTIVE FUNCTIONAL CONTROL OF THERMAL SYSTEM

B. Kleštinec, E. Miklovičová

Slovak University of Technology, Faculty of Electrical Engineering and Information Technology Ilkovičova 3, 812 19 Bratislava, Slovak Republic

Abstract

In this paper laboratory thermal system control using a simple model predictive control algorithm called Predictive Functional Control (PFC) is presented. This algorithm has been proposed as an alternative to PID controllers for demanding processes to be comparatively cheap to implement and commission and relatively easy to tune and understand. PFC control scheme has been implemented in MATLAB®/Simulink® environment. In real-time PFC implementation, the process has been connected through I/O wires to a programmable logic controller (PLC), which has assured data acquisition and control input implementation. For the communication between PLC and PC the OPC communication protocol (OLE for Process Control) has been used.

1 Introduction

Model predictive control systems belong to advanced control strategies and are mainly used for complex optimizing control. They make explicit use of plant model to predict future plant behavior and to calculate the future control sequence minimizing an objective function. Depending on considered method, the control variable calculation can be relatively complex requiring large matrix inversion or online iterative optimization. Predictive algorithms proved their effectiveness in many industrial applications and are available in various commercial control packages.

Despite the wide development of advanced control methods, PID controllers are still commonly used in industry for its structural simplicity and design rules of thumb. Any control strategy that is proposed as an alternative to the industrial PID must be easy to understand, tune and implement in order to be accepted by PID users. Motivated by the technological and economic requirements of industry, model predictive control technique called Predictive Functional Control (PFC) [1], [2] has been proposed in the 1970s and a number of its successful applications to a large variety of processes exists to date. PFC algorithm is a very effective simple controller with low real-time computation, which does not require any matrix inversion; it ensures quick and accurate reference trajectory tracking and can also handle constraints easily. PFC controller parameters have a physical meaning and they are easy to tune and to understand.

In [3] a relation between PI(D) and PFC controller has been analytically derived for the first-order and the second-order processes without dead time and it has been shown that PI(D) controllers can be tuned in such a way that they are equivalent to the PFC controller.

In PFC, like in other model predictive control strategies, a sequence of several future control actions up to a given control horizon is calculated at each sampling time, which makes it suitable to use the precalculated control values in networked control systems to solve the packet losses problem, as proposed in [4].

PFC algorithm is based on assumption that the linear model effectively characterizes the process dominant behavior. The good tracking and robustness properties heavily depend on the accuracy of this plant model. If the model uncertainties and disturbances are significant, the tracking performances can deteriorate. In [5] a dual loop control scheme with a disturbance observer has been proposed in order to improve the PFC tracking performance in the presence of high order disturbances. The PFC parameter optimization based on differential evolution algorithm has been proposed in [6] which allows to prevent and solve the plant-model mismatch and to realize fast accurate tracking control. For nonlinear processes with occasional abrupt changes of parameters the adaptive fuzzy predictive functional control has been proposed in [7]. PFC algorithm using the supported vector machine for the nonlinear system approximation has been derived in [8].

In this paper the basic PFC algorithm is presented and applied to a temperature control of laboratory thermal system. The control scheme has been implemented in MATLAB®/Simulink® environment and the control input implementation and data acquisition have been performed by means of simple programmable logic controller (PLC).

2 Predictive Functional Control Design

Consider a first-order process defined by a gain K and a time constant T represented by a model in discrete form sampled at a rate T_s

$$y_m(n) = a y_m(n-1) + K\,b\,u(n-1) \tag{1}$$

where $y_m(n)$ is the model output, $u(n)$ is the input and $a = e^{\frac{T_s}{T}}$, $b = 1 - a$. This model structure represents an acceptable approximation for many industrial processes.

A reference trajectory represents a suggested path by which the process output $y_p(n)$ should converge to a set-point $y_{sp}(n)$, i.e. it defines the desired closed-loop behavior. In PFC design, an exponential function is used to implement the reference trajectory, which is easy to calculate in real-time

$$y(n) = e^{\frac{-nT_s}{T}} \tag{2}$$

where the decrement is $\lambda = e^{\frac{T_s}{T}}$. The reference trajectory is re-initialized at each sampling point using the measured process output, thus ensuring the closure of the feedback loop.

The principle of PFC is that the process output achieves the reference trajectory only at one coincidence point $n+h$ using one change in the input variable (Fig. 1). The desired input change during the prediction horizon is calculated from the change of the reference trajectory compared to the predicted change of the model output.

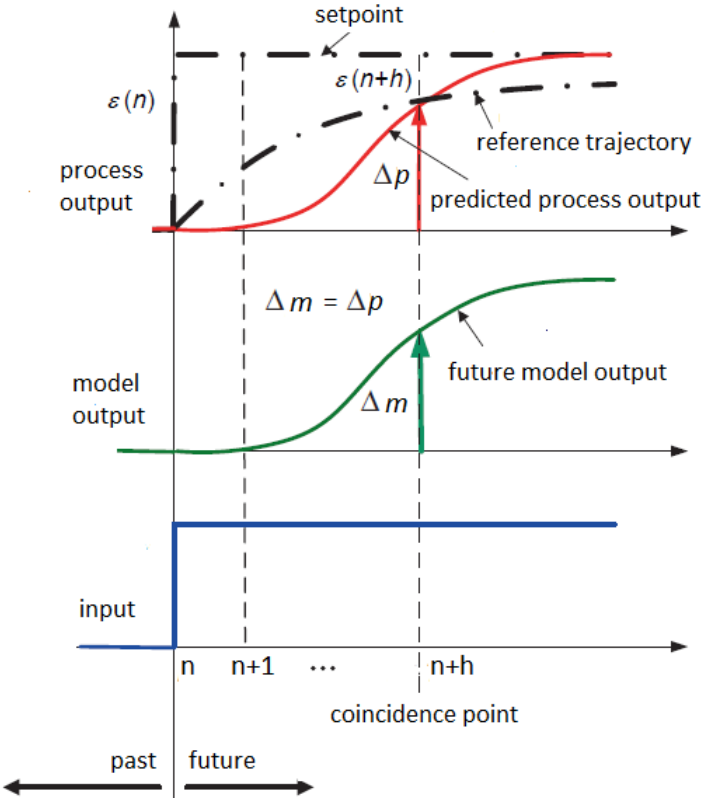

Figure 1: Reference trajectory

The current error signal is

$$\varepsilon(n) = y_{sp}(n) - y_p(n) \tag{3}$$

and at time $n+h$ it is desired that

$$\varepsilon(n+h) = \varepsilon(n)\lambda^h \tag{4}$$

so the desired increment of process output Δp at time $n+h$ is given by

$$\Delta p = \varepsilon(n) - \varepsilon(n)\lambda^h = \varepsilon(n)\left(1-\lambda^h\right) \tag{5}$$

The objective is to calculate the control signal $u(n)$ that will produce a model output increment Δm equivalent to the process output increment (5), i.e.

$$y_m(n+h) = y_m(n) + \varepsilon(n)\left(1-\lambda^h\right) \tag{6}$$

The model output future values (predictions) are composed of two parts:

- free response – starting at a value $y_m(n)$ the model output responds to $u(n)$ while it is assumed that $u(n+i) = 0,\ i \geq 0$

$$y_{m\_free}(n+h) = y_m(n)a^h \tag{7}$$

- forced response – the past values of input and output are zero and the model output responds to future control signal values $u(n+i) = u(n),\ i \geq 0$

$$y_{m\_forced}(n+h) = u(n)K\left(1-a^h\right) \tag{8}$$

The resulting future model output is a sum of (7) and (8)

$$y_m(n+h) = y_{m\_free}(n+h) + y_{m\_forced}(n+h) = y_m(n)a^h + u(n)K\left(1-a^h\right) \tag{9}$$

Comparing equations (6) and (9) it yields that

$$\left(y_{sp}(n) - y_p(n)\right)\left(1-\lambda^h\right) = y_m(n)a^h + u(n)K\left(1-a^h\right) - y_m(n) \tag{10}$$

Extracting the control input from (10) we obtain the PFC control law in the form

$$\begin{aligned} u(n) &= \frac{\left(y_{sp}(n) - y_p(n)\right)\left(1-\lambda^h\right) + y_m(n)\left(1-a^h\right)}{K\left(1-a^h\right)} = k_1\varepsilon(n) + k_2 y_m(n) \\ k_1 &= \frac{\left(1-\lambda^h\right)}{K\left(1-a^h\right)} \qquad k_2 = \frac{1}{K} \end{aligned} \tag{11}$$

This control law has two tuning parameters:

- Closed-Loop Time Response (CLTR) which may be defined as a time for the process output to reach 95% of the step setpoint change. Since for the first-order system 95% time response equals to three times its time constant, the parameter λ is related to CLTR as follows

$$\lambda = e^{-\frac{3T_S}{CLTR}} \tag{12}$$

- Coincidence point h which is generally taken at the inflexion point of the process step response. Smaller value of h results in accurate tracking of the reference trajectory with very active control signal. If the coincidence point is distant, the tracking performances are worse and the control signal is less active.

More information about the PFC tuning and the influence of its parameters on accuracy, dynamics and robustness can be found in [1].

It is apparent that the control law (11) includes no explicit integrator, which is favorable from the practical implementation point of view, as it eliminates the issues like the integrator initialization or wind-up effect. However, it may be shown that there is an implicit integrator effect present in the controller which ensures the offset-free performances in steady state in the case of state disturbances or process /model mismatch.

The PFC design can be extended also to higher-order processes. Any higher-order aperiodic process can be approximated by a parallel connection of first-order processes with different time constants [2]. For example, a second-order transfer function

$$G(s)=\frac{K}{(T_1 s+1)(T_2 s+1)} \tag{13}$$

can be replaced by two first-order submodels as follows

$$G(s)=\frac{K}{(T_1 s+1)(T_2 s+1)}=\frac{K_1}{T_1 s+1}+\frac{K_2}{T_2 s+1}=\frac{K_1(T_2 s+1)+K_2(T_1 s+1)}{(T_1 s+1)(T_2 s+1)} \tag{14}$$

where K_1 and K_2 are obtained from equations

$$\begin{aligned} K_1+K_2&=K \\ K_1\cdot T_2+K_2\cdot T_1&=0 \end{aligned} \tag{15}$$

If the process model is approximated by (14) then its output be can calculated in the following way

$$\begin{aligned} y_m(n)&=y_{m1}(n)+y_{m2}(n) \\ &=y_{m1}(n-1)y_{m1}+(1-a_{m1})K_1u(n-1)+y_{m2}(n-1)a_{m2}+(1-a_{m2})K_2u(n-1) \end{aligned} \tag{16}$$

where $y_{m1}(n)$ and $y_{m2}(n)$ are the outputs of the first-order submodels and $a_{m1}=e^{-\frac{T_s}{T_1}}$, $a_{m2}=e^{-\frac{T_s}{T_2}}$.

Then the PFC control law has the form

$$u(n)=\frac{(y_{sp}(n)-y_p(n))(1-\lambda^h)+y_{m1}(n)(1-a_{m1}^h)+y_{m2}(n)(1-a_{m2}^h)}{K_1(1-a_{m1}^h)+K_2(1-a_{m2}^h)} \tag{17}$$

3 Thermal System Control

PFC algorithm has been used to control a laboratory thermal system which consists of an insulated box divided into two parts, both containing a heating coil controlled by a voltage U_C within the range 0 – 10 V. Air with a room temperature is blown into the box by means of a fan controlled by a voltage $U_V \in \langle 3.5;10 \rangle\ V$. Inside the box four temperature sensors are installed (T_1, T_2, T_3, T_4); their range is 0 – 10 V, too. The control objective is to control the box internal temperature by means of heating coil voltage U_C. Fan voltage U_V has been used to generate the air mass flow rate disturbance. The control scheme is shown in Fig. 2.

The process model identification and PFC algorithm implementation have been performed in the MATLAB®/Simulink® environment. The process has been connected through I/O wires to a programmable logic controller (PLC) in order to assure data acquisition and control input implementation. The OPC communication protocol [9], [10] has been used for the communication between PLC and PC. OPC, which stands for Object Linking and Embedding (OLE) for Process Control, is an industrial standard widely used within the industrial automation to specify the mechanism for communication of different data sources and client applications. The OPC server is a software program based on a Server/Client architecture that converts the hardware communication protocol used by PLC into the OPC protocol and thus enables the OPC client to read or write the PLC data. In our application the PLC Simatic S7-200 with OPC server PC Access V1.0 has been used.

The OPC Toolbox™ [11] extends MATLAB® and Simulink® with tools for interacting with OPC servers and can be used in process industries for data analysis, visualization, simulation, and rapid prototyping of algorithms on real processes. The toolbox includes four Simulink blocks. The OPC Configuration block lets us to create and configure an OPC Data Access (DA) Client object, to define the behavior for OPC errors and events and to set the pseudo real-time behavior. The OPC Read and OPC Write blocks retrieve and transmit data synchronously or asynchronously to and from the OPC DA server. OPC Quality Parts block converts OPC quality ID into vendor, major, minor, and limit status. In [12] the OPC configuration has been described in detail and experimentally tested.

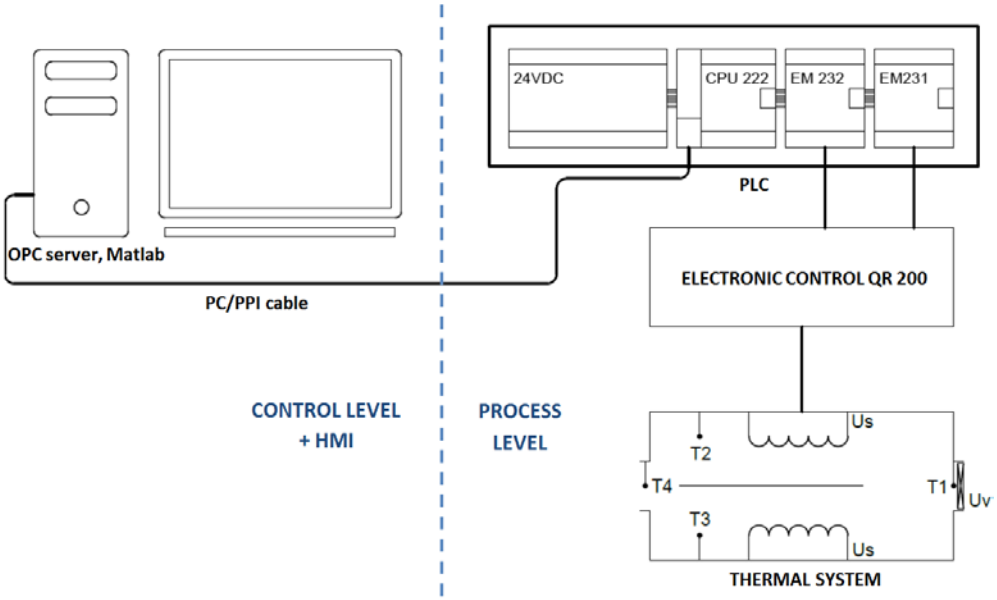

Figure 2: Thermal system control

The process model identification has been performed in the operating point given by $U_C = 5V, U_V = 5V$ with the sampling period $T_S = 5V$. The following first-order and second-order discrete-time models have been obtained:

$$G(z)_{arx111} = \frac{0.02159}{z - 0.9559} \tag{18}$$

$$G(z)_{arx221} = \frac{0.01226z + 0.008446}{z^2 - 1.046z + 0.08808} \tag{19}$$

and their continuous-time equivalents are

$$G(s)_{arx111} = \frac{0.004415}{s + 0.009031} = \frac{0.4888}{110.7297s + 1} \tag{20}$$

$$G(s)_{arx221} = \frac{-0.001049s + 0.002226}{s^2 + 0.4859s + 0.004566} = \frac{-0.0123}{2.0995s + 1} + \frac{0.4998}{104.3175s + 1} \tag{21}$$

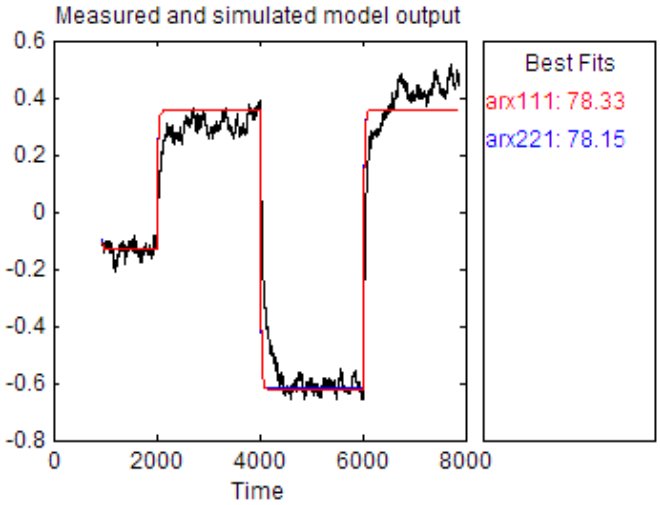

Figure 3: Identification: process and model outputs

The comparison of the measured process output and the identified model outputs is in Fig. 3. It can be seen that both models have approximately the same dynamics, the process output is affected by a measurement noise and the models do not exactly capture the process behavior.

Based on the models (20) and (21) the PFC control laws (11) and (17) have been designed with $h = 30$ and $CLTR = 200\ s$ and implemented in MATLAB® and Simulink®. At first the control system performances have been verified by means of simulations using the process models and subsequently the real-time experiments have been performed. The Simulink real-time control scheme for the first-order process model is shown in Fig. 4.

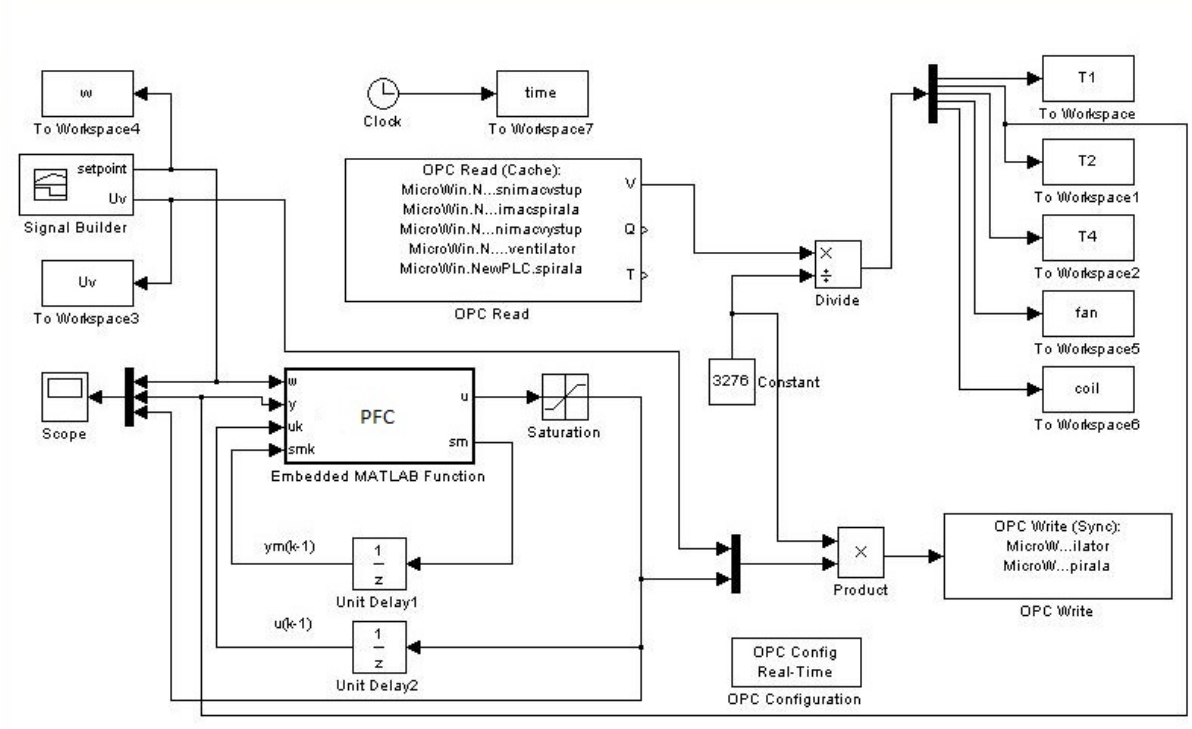

Figure 4: Simulink control scheme

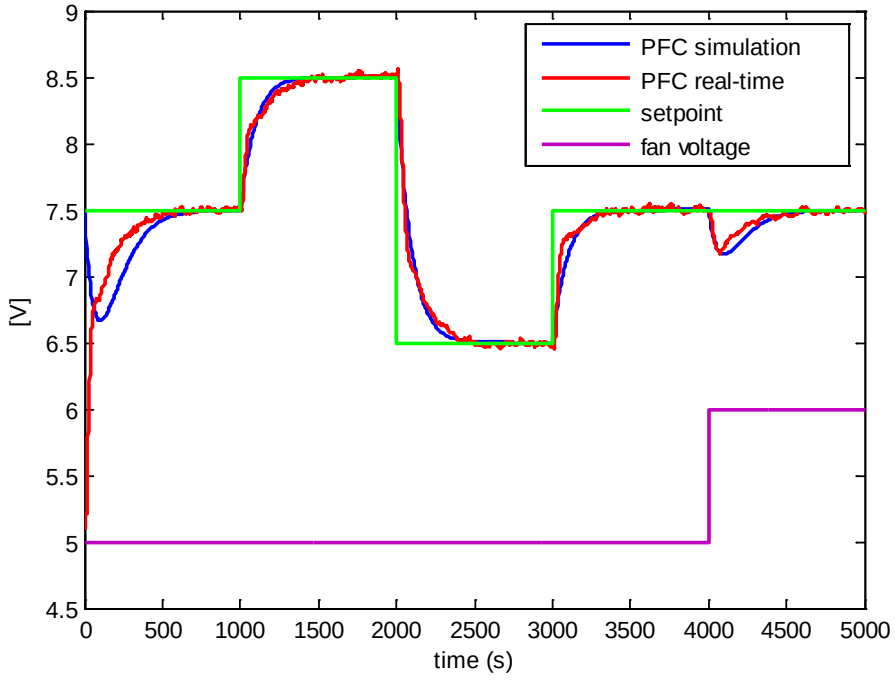

Figure 5: Comparison of simulation and real-time control results – output, setpoint and disturbance

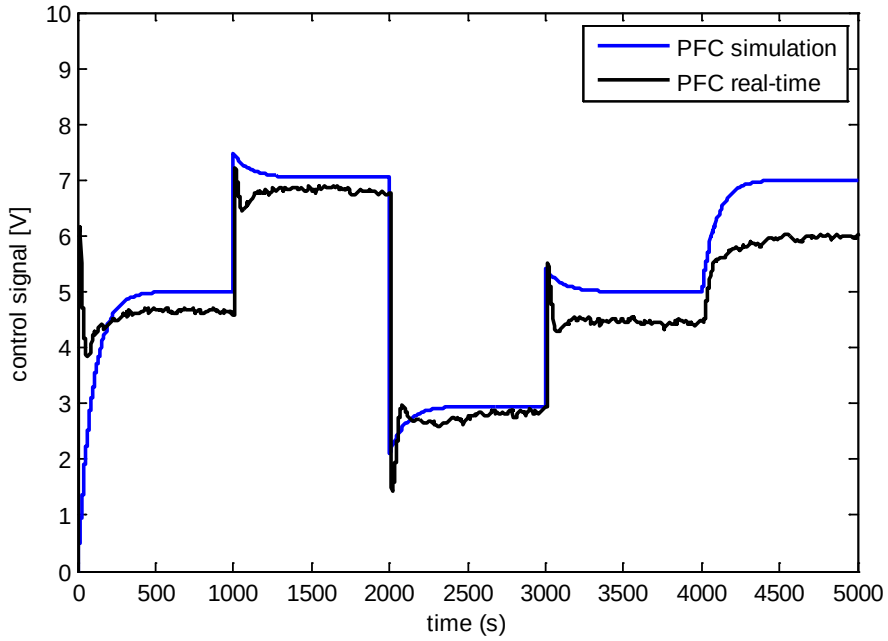

Figure 6: Comparison of simulation and real-time control results – control signal

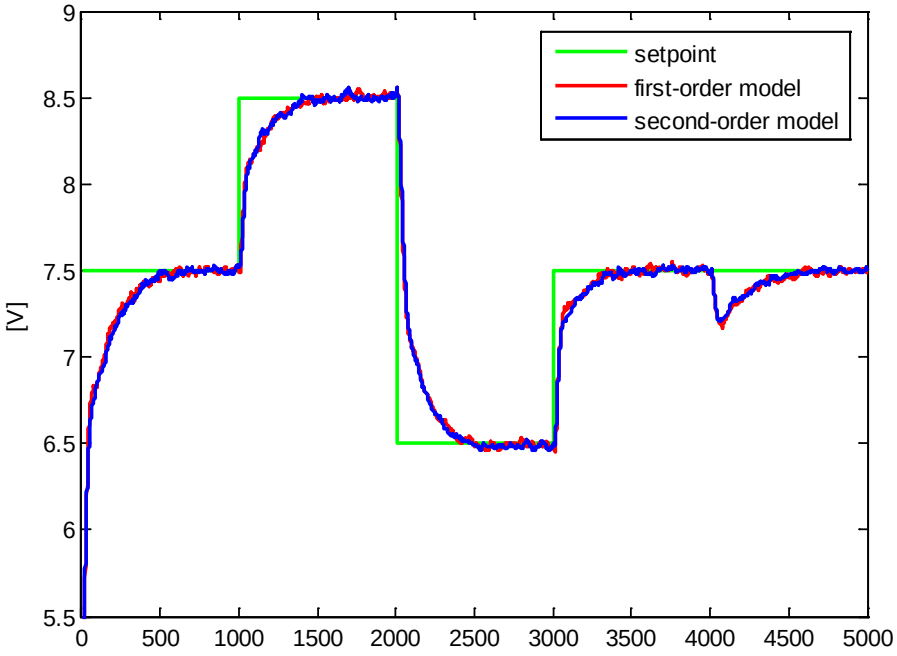

Figure 7: Comparison of real-time control results

Figures 5 and 6 show the simulation and real-time control results for the first-order process model and the control law (11). Despite the process model uncertainty and the measurement noise the real-time control performances are almost identical to those obtained by simulations. The differences in the control signal steady states are due to changed input air temperature, which is influenced by a room temperature. The real-time control results for both control laws (11) and (17) are compared in Fig. 7. It can be concluded that in both cases the obtained control performances are comparable hence the PFC controller based on the first-order process model is sufficient and suitable for this thermal system.

4 Conclusion

In this paper the application of PFC algorithm to temperature control of the laboratory thermal system has been presented. PFC algorithm is far simpler than other model-based predictive control algorithms, its parameters have a physical meaning and they are easy to tune and to understand; so this control algorithm is attractive from the implementation point of view. The PFC control scheme has been implemented in MATLAB®/Simulink® environment and the data acquisition and control input implementation have been performed by means of the PLC using the OPC communication protocol. MATLAB® and Simulink® tools facilitate the process identification, control design and analysis and also the interaction with OPC servers and real-time control execution.

Acknowledgements: This work was supported by the Slovak Scientific Grant Agency, Grant No. 1/2256/12.

References

[1] J. Richalet, D. O'Donovan. *Predictive functional control: Principles and industrial applications*. London: Springer, 2009.
[2] R. Haber, R. Bars, U. Schmitz. *Predictive Control in Process Engineering.* Wiley-VCH Verlag GmbH & Co. KgaA, 2011.
[3] K. Zabet, R. Haber. *Equivalence of PI(D) and predictive functional control for aperiodic processes without dead time.* 12th International Carpathian Control Conference (ICCC), 2011, pp. 459 - 464.
[4] Yin Yang , Xia Li , Li Hong Ke. *Research on Predictive Functional Control over Lossy Packet Networks.* 2010 International Conference on Electrical and Control Engineering, 2010, pp 1738 – 1741.
[5] T. Satoh, K. Kaneko, N. Saito. *Performance Improvement of Predictive Functional Control: A Disturbance Observer Approach*. IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society, 2011.
[6] Ma Xiao-Ping, Li Ya-Peng, Su Pi-Zhao, An Feng-Shuan. *Predictive Functional Control Based on Differential Evolution Algorithm and its Dynamic Performance Analysis.* Chinese Control and Decision Conference, 2010, pp. 65 - 68.
[7] D. Dovžan, I. Škrjanc. *Fuzzy Predictive Functional Control with Adaptive Fuzzy Model*. International Joint Conferences on Computational Cybernetics and Technical Informatics (ICCC-CONTI 2010), 2010, pp. 143 - 147.
[8] Xi-Yun Yang, Yue-Gang Lv, Da-Ping Xu, Xin-Fang Zhang. *Nonlinear Predictive Functional Control Based on Support Vector Machine*. Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, 2006.
[9] OPC Foundation 1998-2010, Specifications of OPC, available on: http://www.opcfoundation.org.
[10] M.H. Schwarz, J. Boercsoek. *A Survey on OLE for process control (OPC)*. Proc, of the 7th WSEAS International Conference on Applied Computer Science, Venice, Italy, November 2007, pp. 186-191.
[11] OPC Toolbox, http://www.mathworks.com/products/opc/, http://www.mathworks.com/help/opc/index.html
[12] M. Mrosko, L. Mrafko, L. Körösi. *Real time control*. 9th International Conference Process Control 2010, Kouty nad Desnou, Czech Republic.

Blažej Kleštinec
bklestinec@gmail.com

Eva Miklovičová
eva.miklovicova@stuba.sk