Real-time implementation of predictive control using programmable logic controllers

Marián Mrosko and Eva Miklovičová

***Abstract*—** In this paper the real-time aspects of digital control system implementation are investigated. The control system design is based on the model predictive control, which is one of the most popular advanced control design techniques. Two alternatives of predictive control implementation using the programmable logic controllers (PLC) are proposed and compared. In the first case the control law design and execution is performed in PC using the MATLAB/Simulink environment and the PLC is used only to accomplish the data acquisition and control input implementation. The communication between PC and PLC is ensured by the OPC communication protocol. In the second case the control law is directly implemented in PLC using the available instructions. The PC is not needed for the real-time control execution; it only supports the control design and the signal processing and visualization. In both cases controlled plant is wired to PLC.

***Keywords*—** model predictive control, programmable logic controller, OPC communication, real-time system.

I. Introduction

Control of industrial processes is often realized using small digital computers called the programmable logic controllers (PLCs), where the hardware and software are specifically adapted to industrial environment. PLCs represent a cost effective solution for control of complex systems. They offer many advantages, such as flexibility (they can be reapplied to control other systems quickly and easily) and reliability (immunity to electrical noise, resistance to vibration and impact). However, this type of controllers usually offers only simple control structures, such as on-off control or PID control loops. In [1] an approach that models programmable logic controllers (PLCs) for their effective deployment in industrial control processes is presented.

Wide development of control theory brought many advanced control methods with improved control performances, robustness or stability. Control laws of these methods usually differ from the control structures available in PLC, so their real-time implementation is more complicated. Reference [2] describes a hardware-in-the-loop environment for realistic experimentation of advanced controllers in order to facilitate its study and posterior implementation in productive processes. The experimentation environment is formed by an industrial PC that acts as real time controller and other PC or real actuators like process plants. A hardware/software environment for hard real time simulation of dynamical systems and implementation and testing of robust controllers, based on Real Time Linux (Linux-RTAI) and COMEDI project, has been developed in [3].

The real-time control aspects play an important role in implementation of digital control systems. A real-time system can be defined as a system in which the correctness of a result not only depends on the logical correctness of the calculation but also upon the time at which the result is made available. The real-time computing is not equivalent to fast computing. Fast computing aims at getting the results as quickly as possible, while real-time computing aims at getting the results at a prescribed point of time within defined time tolerances [4].

This paper deals with the real-time implementation of one of the most popular advanced control design approaches, namely the model predictive control, by means of PLC Simatic S7-200 [5]. Model predictive control (MPC) has developed greatly over the last decades both within the research control community and in industry. MPC refers to a family of advanced control methods which make explicit use of the process model to predict the future process behavior and to calculate a future control sequence minimizing an objective function [6]. MPC has proved its effectiveness in many industrial application areas including petrochemical, chemical and food processing, automotive and aerospace industries [7].

Predictive algorithms are available in various commercial control packages but their implementation costs could be considerable. It could be desirable to realize the MPC control laws on simple and more affordable controllers, such as PLC. In [8] the model predictive control implementation based on the conventional PID controller structure available in PLC has been proposed, provided that the plant model structure has been properly chosen. The real-time implementation of predictive control using the PLC B&R System 2005 and Process Visualization Interface has been presented in [9].

Two variants of MPC implementation using the PLC are presented in the paper. The first one consists in using another control system, for example PC, where the control algorithm

Manuscript received December 2, 2011: Revised version received December 2, 2011. This work was supported by the Slovak Scientific Grant Agency, Grant No. 1/2256/12 and by the Slovak Research and Development Agency under the contract APVV-0211-10.

Authors are with the Institute of Control and Industrial Informatics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Ilkovičova 3, 812 19 Bratislava, Slovak republic

(e-mails: eva.miklovicova@stuba.sk, marian.mrosko@stuba.sk)

is designed and executed. In this case it is necessary to assure the communication between this control system and PLC. The communication protocol OPC can be used for this purpose. The second possibility is direct implementation of predictive control algorithm in PLC using the available instructions. In the following the pros and cons of both solutions will be analyzed.

The paper is organized as follows. In the next section the OPC communication and the problems concerning the real-time experiments are presented. The model predictive control design is briefly described in Section III. Two alternatives of MPC real-time implementation using the simple industrial controllers are proposed and experimentally evaluated in Section IV. Finally, some conclusions are given.

II. OPC

OPC (originally OLE for Process Control) which stands for Object Linking and Embedding (OLE) for Process Control, is an industrial standard created with the collaboration of a number of leading worldwide automation hardware and software suppliers, working in cooperation with Microsoft. The standard is maintained by OPC Foundation [10] and widely used within the industrial automation to facilitate the interoperability of control devices from different manufacturers. It specifies the mechanism for communication of different data sources and client applications within the process control. The data source can be a process control system, a database or a supervisory control application. Reference [11] gives detailed information about OPC, and how OPC can be beneficial for research and development and presents an overview of the latest developments and standards.

The OPC Specification is a non-proprietary technical specification that defines a set of standard interfaces based upon Microsoft's OLE/COM/DCOM platform and .NET technology. The application of the OPC standard interface enables the interoperability between automation/control applications, field systems/devices and business/office applications.

Traditionally, each software or application developer was required to write a custom interface or server/driver, to exchange data with hardware field devices. OPC eliminates this requirement by defining a common, high performance interface that permits this work to be done once, and then easily reused by HMI, SCADA, Control and custom applications.

A. OPC server

OPC server is the software application which operates as the application programming interface (API) or as the protocol converter. OPC Server is connected to a device such as PLC, distributed control system (DCS), remote terminal unit (RTU) or the data source (database or user interface) and translates the data into a standard-based OPC format. OPC compliant applications such as a human machine interface (HMI), historian, spreadsheet, trending application, etc. can be connected to the OPC Server and then they can use it to read and write the device data. The OPC Server is based on a Server/Client architecture.

B. OPC Toolbox

MATLAB® [12] is a high-level language and interactive environment that enables to perform computationally intensive tasks. It is often used in academic community for design, analysis and simulation of advanced control techniques.

The OPC Toolbox™ [12] extends MATLAB® and Simulink® with tools for interacting with OPC servers. It enables to read, write, and log OPC data from devices that conform to the OPC Foundation Data Access standard, such as distributed control systems, supervisory control and data acquisition systems, and programmable logic controllers. The toolbox enables MATLAB and Simulink products to respond to an OPC server- or OPC Toolbox software-initiated event, such as a shutdown, server error, or item value change. MATLAB with OPC Toolbox can be used in process industries for data analysis, visualization, simulation, and rapid prototyping of algorithms on real processes.

The OPC Toolbox provides three ways to implement an OPC Data Access Client:

1. Execute all OPC Toolbox functions directly from the MATLAB command line or incorporate them into the MATLAB applications.
2. Use the graphical user interface (GUI) to rapidly connect to OPC servers, create and configure OPC Toolbox objects, and read, write, and log data.
3. Use the Simulink Blockset library to read and write data to and from the OPC server while simulating a system.

In [13] the OPC configuration has been described in detail and experimentally tested.

C. Hierarchical OPC data access object

When used in MATLAB, the toolbox employs an intuitive, hierarchical object structure to help you manage connections to OPC servers and collections of server items or tags. You create an OPC Data Access Client object to connect to an OPC server. This connection lets you browse the server name space and retrieve properties of each item stored on the server. You create Data Access Group objects to control sets of Data Access Item objects, which represent server items. The toolbox allows configuring and controlling all client, group, and item objects by modifying their properties. OPC Tool shown in Fig. 1 enables to browse the server's name space, configure the objects, and read and write the OPC data. It also enables to log OPC data into MATLAB for analysis and plotting.

Simulink's OPC toolbox offers a configuration block to specify the OPC clients used in the model, to define the behavior for OPC errors and events and to set the real-time behavior. During the simulation, the model executes in pseudo real-time, matching the system clock as closely as possible by automatically slowing the simulation. The block parameters

can also be configured so that the simulation runs more slowly than the system clock. Fig. 2 shows the OPC Configuration window and the window for OPC Client management can be seen in Fig. 3.

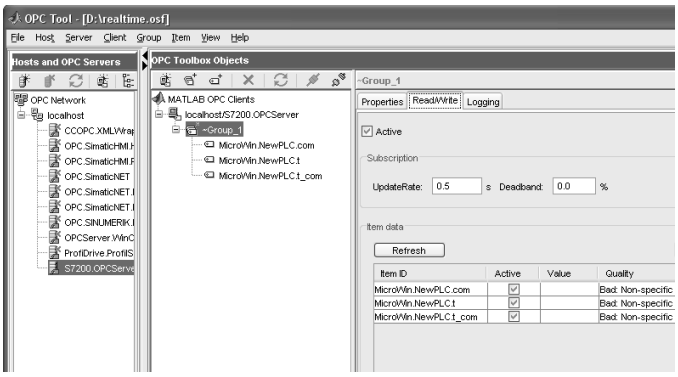

Fig. 1 OPC Tool

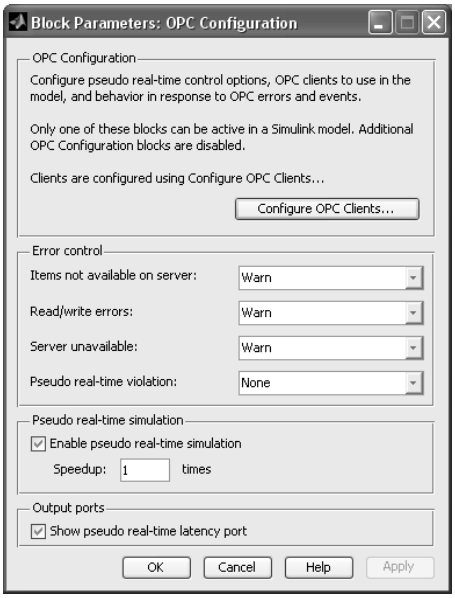

Fig. 2 OPC Configuration

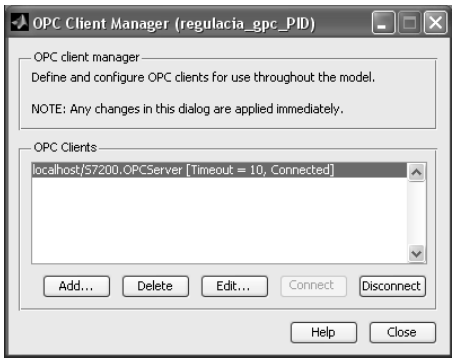

Fig. 3 OPC Client Manager

In the OPC Configuration window, *Pseudo real-time simulation* panel allows configuring options for running the simulation in pseudo real-time. When checkbox "Enable pseudo real-time simulation" is checked, the model execution time matches the system clock as closely as possible by slowing down the simulation appropriately. The "Speedup" setting determines how many times faster the simulation runs comparing to the system clock. For example, when "Speedup" is set to 2, it means that a 10-second simulation will take 5 seconds to complete.

Note that the real-time control settings do not guarantee real-time behavior. If the model runs slower than real time, a pseudo real-time latency violation error occurs. You can control how Simulink responds to a pseudo real-time latency violation using the settings in the *Error control* panel. The "Show pseudo real-time latency port" check box enables to output the model latency. When it is checked, the pseudo real-time latency (in seconds) appears as an output port of the "OPC Configuration" block. Pseudo real-time latency is the time spent by waiting for the system clock during each step. If this value is negative, the simulation runs slower than real time and the Simulink action is determined in the "Pseudo real-time violation" setting.

D. Reading and writing OPC data

Once a group object containing item objects is created, you can read from or write to individual items or all the items in the group simultaneously. In MATLAB, read and write operations can occur either synchronously (MATLAB execution is blocked until the operation is complete) or asynchronously (MATLAB can continue processing while the operation is in progress).

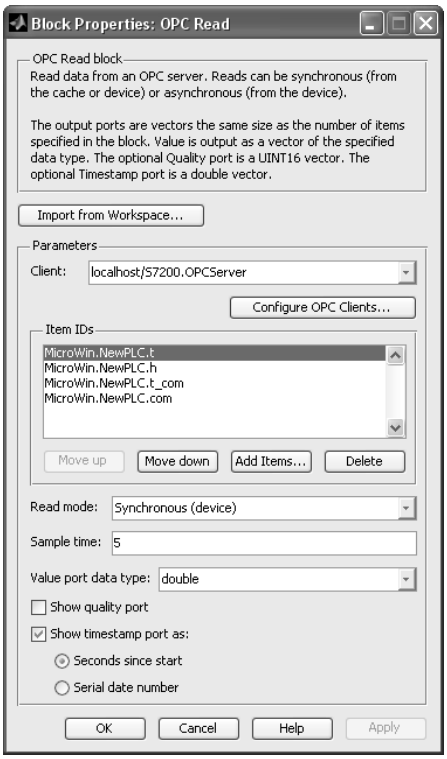

Fig. 4 OPC Read

In Simulink, the read and write blocks retrieve and transmit data synchronously or asynchronously to and from the OPC server. The blocks contain a client manager that makes possible to specify and manage the OPC server, select items, and define block sample times. The OPC Read Block shown in Fig. 4 enables to choose items from the OPC server and to read online plant data directly to the Simulink model. The OPC Write Block (Fig. 5) enables to choose items from the Simulink model and to write online plant data directly to the OPC server.

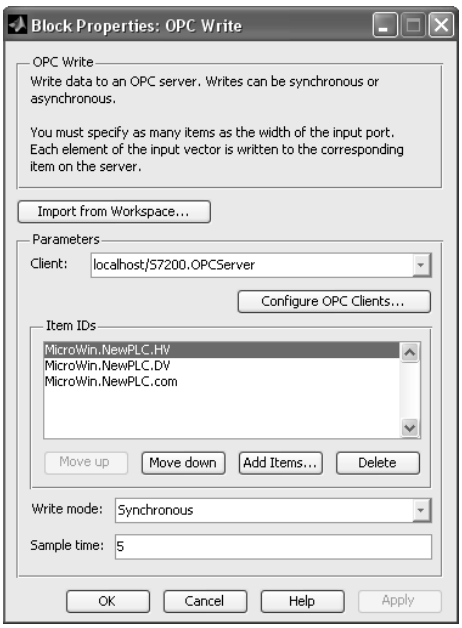

Fig. 5 OPC Write

E. Logging OPC Data

The toolbox enables to log data as it changes over time. Data can be logged to a memory or to a disc. The MATLAB and add-on toolboxes can then be used to analyze and visualize the data, to design the control systems or to optimize the plant performances.

F. Simulink block execution

During the simulation of real-time (or pseudo real-time) control strategies in Simulink, an issue of block execution order can occur. Without explicitly defining block priorities in Simulink, the blocks are executed in order given by so called "sorted order". There are two basic rules, which affect the block order:

1. each block, which drives another block's "direct-feedthrough" ports must precede this block,
2. blocks without direct-feedthrough ports must give precedence to any blocks that drive the direct-feedthrough ports.

Direct-feedthrough port is an input port, whose current value determines current value of one of the block's outputs. Examples of blocks that have direct-feedthrough ports include the Gain, Product, and Sum blocks. Examples of blocks that have non-direct-feedthrough inputs include the Integrator block (its output is a function purely of its state), the Constant block (it does not have an input) and the Memory block (its output is dependent on its input in the previous time step).

In Simulink there is also a possibility to assign block's priority, by which it is possible to influence the blocks sorted order. Block's priority can be assigned interactively using the "Priority" field of the block's "Block Properties" dialog box or programmatically using "set_param" command. The lower the number, the higher the priority; that is, 2 means the higher priority than 3. Higher priority blocks appear before lower priority blocks in the sorted order, though not necessarily before blocks that have no assigned priority.

G. Real-time experiments

During the implementation of real-time experiments in Simulink through the OPC communication it can happen that the real execution time is longer than the simulation time set in the Simulink configuration parameters. Fig. 6 shows the results of the real-time simulation where the real execution time of 30 s simulation was 31.6 s. The latency at each step is negative, which means that each sample period is longer than 1 s [6].

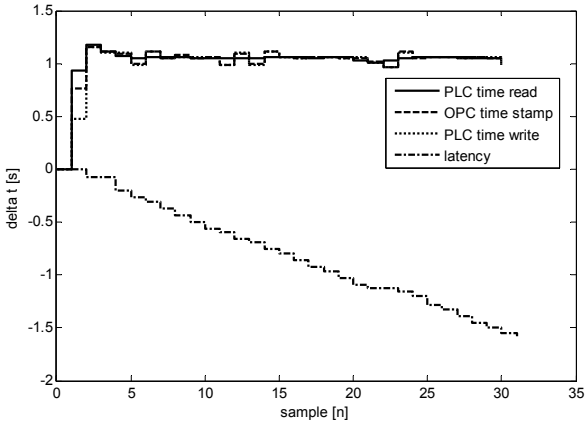

Fig. 6 Sample time and latency

Latency represents the sample time margin; the larger latency means larger margin, i.e. the sample time can be more decreased. There are several factors that allow getting the execution time near the value set in the Simulink scheme.

1) Baud rate

The speed of data transfer across the network is given by the baudrate, which is typically measured in units of kilobaud (kbaud) or megabaud (Mbaud). The baudrate measures how much data can be transmitted within a given amount of time. Latency of samples with different baud rate is shown in Fig. 7.

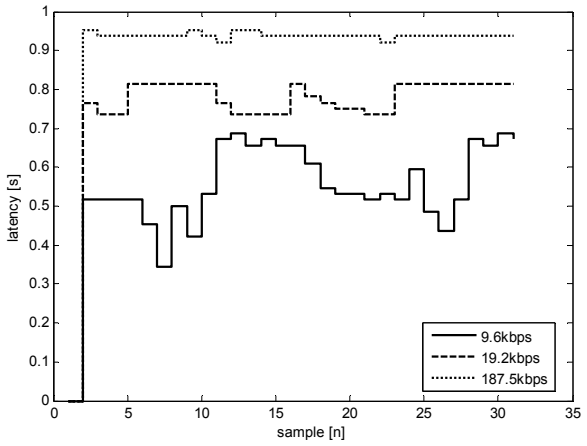

Fig. 7 Latency dependence on baudrate

2) Number of OPC Read and OPC Write blocks

Latency also depends on the number of OPC Read and OPC Write blocks as shown in Fig. 8. It can be seen that the number of OPC Write blocks influences the latency more than the number of OPC Read blocks.

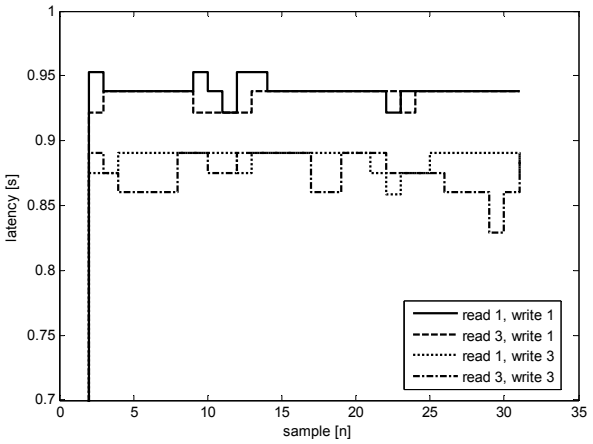

Fig. 8 Latency dependence on the number of OPC Read and Write blocks

3) Number of items for OPC Read and OPC Write

In Fig. 9 dependence of latency on the number of OPC Read and OPC Write items is investigated. It can be concluded, that the number of OPC Read and Write items does not significantly influence the latency.

When implementing the experiments with real processes it is important to properly set the communication parameters, more specifically, the data transfer rate, which is often forgotten in practice. Another very important factor is the sample time, which should be chosen taking into account the number of communication items (the number of exchanged process and control variables) and the sufficient computational reserve in case of the incidental processor overload by another process.

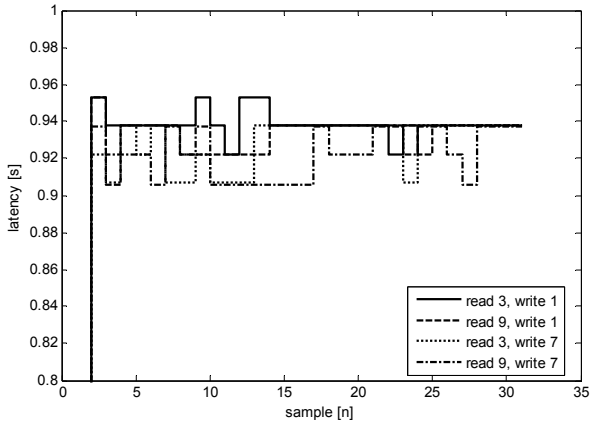

Fig. 9 Latency dependence on the number of OPC Read and Write items

Implementation of real-real-time control strategies in Simulink and OPC communication meets some restrictions that are critical for processes with fast dynamics. The order of blocks execution in Simulink is highly influenced by the time difference between the sample reading and writing. It is essential to analyze the latency and the sample time variance, which is often omitted in setting the pseudo-real time properties in Simulink. This allows avoiding the problems in implementation of real-time control systems.

III. Model Predictive Control

A. Control design

Generalized predictive control (GPC) proposed in [14] belongs to the most popular model predictive control algorithms. It can handle various control problems for a wide range of plants, its implementation is relatively simple and due to several design parameters it can be tuned to the specific applications. GPC design is based on the linear parametric model of the controlled plant in the discrete-time form

$$A(z^{-1})y(t) = B(z^{-1})u(t-d-1) + v(t) \quad (1)$$

$$D(z^{-1})v(t) = C(z^{-1})\xi(t) \quad (2)$$

with

$$\begin{aligned} A(z^{-1}) &= 1 + a_1 z^{-1} + ... + a_{na} z^{-na} \\ B(z^{-1}) &= b_0 + b_1 z^{-1} + ... + b_{nb} z^{-nb} \\ C(z^{-1}) &= 1 + c_1 z^{-1} + ... + c_{nc} z^{-nc} \\ D(z^{-1}) &= 1 - z^{-1} \end{aligned} \quad (3)$$

where *u(t)* is the control variable, *y(t)* is the measured plant output, *d* denotes the minimum plant model time-delay in sampling periods, *v(t)* represents the external disturbances and *ξ(t)* is the random variable with zero mean value and finite variance. For simplicity in the following the $C(z^{-1})$ polynomial is chosen to be 1.

The key idea of MPC is to use the process model (1) – (3) to predict the future process behavior and to calculate the future control sequence minimizing an objective function. The GPC control objective is to compute the future control sequence in such a way that the future plant output is driven close to the prescribed reference trajectory

$$J = E\left\{\sum_{j=sh}^{ph}\left(\hat{y}(t+j/t) - y^*(t+j)\right)^2 + \rho\left(D\left(z^{-1}\right)u(t+j-sh)\right)^2\right\} \quad (4)$$

subject to

$$D\left(z^{-1}\right)u(t+i) = 0 \text{ for } ch \le i \le ph \quad (5)$$

where *sh*, *ph* and *ch* are positive scalars defining the starting horizon, prediction horizon and control horizon, respectively, ρ is a nonnegative control weighting scalar. $\hat{y}(t+j/t)$ denotes the *j*-step ahead prediction of *y(t)* based on data available up to the time t and $y^*(t+j)$ is the future reference trajectory.

The GPC control design consists in performing the following three steps:

1. Compute the j-step ahead prediction of output $\hat{y}(t+j/t)$ for $j \in \langle sh, ph \rangle$.
2. Minimize with respect to the future control inputs sequence the cost function (4) – (5).
3. Determine the control law in a receding horizon sense – only the first term of the future control sequence is used at each sampling instant and the control sequence is calculated again at the next sampling time. This allows to incorporate a feedback into the control loop and to improve the control performances in the presence of disturbances and unmodelled dynamics.

The GPC control law may be implemented using the standard pole-placement control structure (as shown in Fig. 10)

$$S\left(z^{-1}\right)D\left(z^{-1}\right)u(t) + R\left(z^{-1}\right)y(t) = T\left(z^{-1}\right)y^*(t) \quad (6)$$

with

$$R(z^{-1}) = r_0 + r_1 z^{-1} + \ldots + r_{nr} z^{-nr} \quad (7)$$

$$S(z^{-1}) = 1 + s_1 z^{-1} + \ldots + s_{ns} z^{-ns} \quad (8)$$

Thus for the calculation of the control input at time t only the output and control inputs available up to this time together with the reference value $y^*(t)$ are necessary.

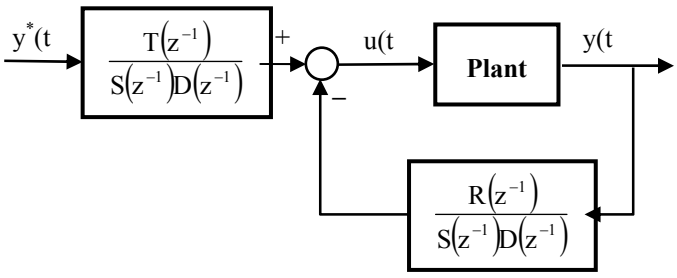

Fig. 10 Control scheme

The orders of $R\left(z^{-1}\right)$ and $S\left(z^{-1}\right)$ polynomials are given by orders of the plant model numerator and denominator. The coefficients of these polynomials depend on the plant model parameters as well as on the choice of the tuning parameters *sh*, *ph*, *ch* and ρ. The choice of this control tuning parameters influences the resulting closed loop performances and stability properties. Many design guidelines and rules of thumb concerning these parameters have been proposed in literature. To calculate the coefficients of $R\left(z^{-1}\right)$ and $S\left(z^{-1}\right)$ polynomials the recursive solution of Diophantine equations is needed which may require a great amount of calculations especially in case of large prediction horizons. However, in a fixed-parameter control case these calculations need to be performed only once before the control design stage.

The $T\left(z^{-1}\right)$ polynomial plays role in attenuation of the plant – model mismatch effects and it can also influence the robust stability. Some guidelines have been proposed for the selection of this polynomial. In this paper we assume the following simple form

$$T(z^{-1}) = \sum_{j=sh}^{ph} \gamma_j = t_0 \quad (9)$$

B. Control implementation

Based on the control law (6) the control input at time t can be expressed as follows

$$u(t) = \frac{T\left(z^{-1}\right)}{S\left(z^{-1}\right)D\left(z^{-1}\right)}y^*(t) - \frac{R\left(z^{-1}\right)}{S\left(z^{-1}\right)D\left(z^{-1}\right)}y(t) \quad (10)$$

Dynamic behavior of many industrial processes with non-oscillatory dynamics around an operating point can be described by the second order model, i.e. *na*=2, *nb*=1. In this case the polynomials in the control law (10) take the following form

$$R\left(z^{-1}\right) = r_0 + r_1 z^{-1} + r_2 z^{-2} \quad (11)$$

$$S\left(z^{-1}\right) = s_0 + s_1 z^{-1} \quad (12)$$

$$SD\left(z^{-1}\right) = S\left(z^{-1}\right)D\left(z^{-1}\right) = \\ = sd_0 + sd_1 z^{-1} + sd_2 z^{-2} \quad (13)$$

Then the control input at time t has the form

$$u(t) = \frac{-sd_1.u(t-1) - sd_2.u(t-2)}{sd_0} - \frac{r_0.y(t) + r_1.y(t-1) + r_2.y(t-2)}{sd_0} + \frac{t_0.y^*(t)}{sd_0} \quad (14)$$

This choice of the plant model orders is not restrictive; the more complex model structure leads only to higher orders of $R(z^{-1})$ and $S(z^{-1})$ polynomials. As the consequence, more past values of control input and output signals have to be stored for the control law evaluation.

IV. Real-time Implementation of Model Predictive Control

In this section two alternatives of the GPC control law real-time implementation using the simple industrial controllers are proposed.

A. PLC + PC control

The process is connected through I/O wires to the PLC. Control design is performed in the second control system (PC) where also the control law (14) is implemented. For example, the MATLAB and Simulink with the OPC Toolbox can be used. In this software environment also the data processing and visualization can be realized.

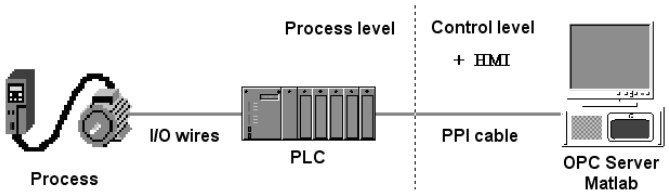

Fig. 11 PLC + PC control

The PLC assures the data acquisition and control input implementation. The advantage is that the control law design and calculation is comfortable without the need of PLC programming. On the other hand, the problems concerning the real-time execution described in section 2 must be born in mind.

B. PLC control

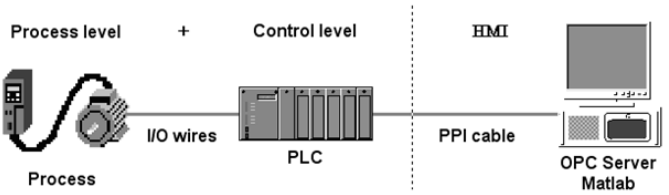

Fig. 12 PLC control

The process is connected through I/O wires to the PLC. The control design, i.e. the calculation of the $R(z^{-1})$, $S(z^{-1})$, $T(z^{-1})$ polynomials, is performed offline in the second control system (PC). The control law (14) is implemented in PLC. In this case the PC is not needed for the execution of real-time control, it only supports the design and signal processing and visualization stage. There are no problems with the control law real-time execution described in section 2.

V. Experimental Evaluation

In order to evaluate the two variants of real-time predictive control implementation described in previous section, the control of simple cylindrical laboratory tank (Fig. 13) using the PLC Simatic S7-200 has been realized. The Siemens SIMATIC S7-200 series [8] is a line of micro-programmable logic controllers that can be used to control a variety of small applications.

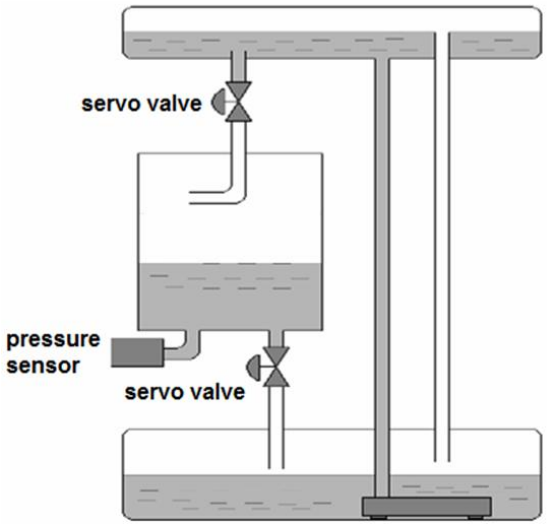

Fig. 13 Cylindrical laboratory tank

The plant output is the water height measured by a pressure sensor and the plant control input is the inflow servo valve opening. The tank has also the outflow servo valve which has been used to generate a disturbance. The servo valves are governed by voltage within the range 0 – 10 V. The pressure sensor range is 0 – 10 V, too.

The following second order plant model has been identified

$$G(z^{-1}) = \frac{-0.002254 + 0.003291z^{-1}}{1 - 1.916z^{-1} + 0.917z^{-2}} \quad (15)$$

with the sampling period $Ts = 1$ s. Based on this model the GPC controller has been designed using the following control design parameters

$$sh = 1, \quad ph = 30, \quad ch = 2, \quad \rho = 10 \quad (16)$$

The real-time control results are shown in Fig. 14 and Fig. 15. The blue line (labelled PC) corresponds to the control law implementation in PC, while the black line (labelled PLC) depicts the control law implementation in PLC.

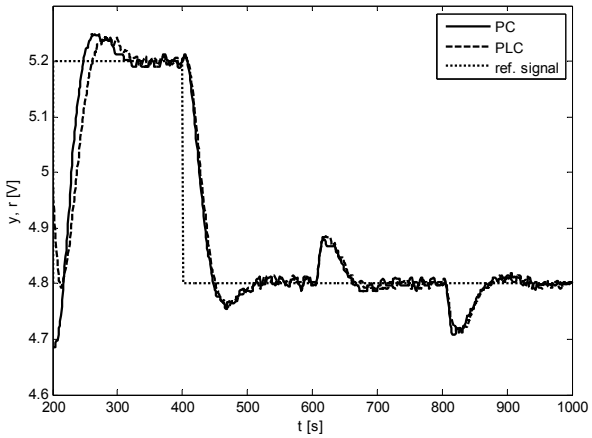

Fig. 14 Time responses of output and reference

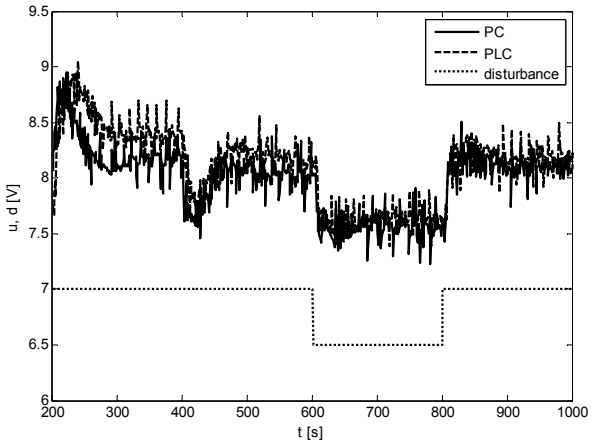

Fig. 15 Time responses of control input and disturbance

The difference in the output time responses in the beginning of experiment is due to the different initial conditions of both experiments. In the further course of experiment, the time responses are very close; the difference is induced only by the measurement noise. It can be concluded that both alternatives of the real-time control implementation are equivalent. The implementation of predictive control law expressed in the RST form in PLC is correct and is fully usable in practice without the need of another control system.

VI. Conclusion

This paper dealt with the real-time implementation of one of the most popular advanced control techniques. It has been shown that the predictive control law expressed in the RST form can be implemented on simple industrial controllers without the need of advanced control packages. The model predictive control brings many advantages in comparison with the PID control; it can be used to control a great variety of processes (including time-delayed systems, the nonminimum phase or the unstable ones) and due to several design parameters it can be tuned to obtain desired control performances and robustness properties.

Acknowledgment

This work has been supported by the Slovak Scientific Grant Agency, Grant No. 1/2256/12 and by the Slovak Research and Development Agency under the contract APVV-0211-10.

References

[1] L. Ngalamou and L. Myers, "An Exploratory Method for Effective Deployment of Programmable Logic Controllers (PLCs)," *WSEAS Transactions on Systems and Control*, vol. 6, pp. 1-14, January 2011.

[2] R. Bárcena and A. Etxebarria, "Real-Time experimentation environment for digital controllers applied to industrial processes," *Proc. of the 12th WSEAS International Conference on Systems*, Heraklion, Greece, July 2008, pp. 557-562.

[3] L. Garcia, M. J. Lopez and J. Lorenzo, "Hardware-in-the-loop Environment for Control Systems evaluation under Linux/RTAI," *Proc. of the 6th WSEAS International Conference on Applied Computer Science*, Tenerife, Canary Islands, Spain, December 2006, pp. 285-290.

[4] Gambier, A., "Real-time Control Systems: A Tutorial," *Proc. of the IEEE 5th Asian Control Conference, Melbourne*, Australia, July 2004, pp. 1024-1031.

[5] *S7-200 Programmable Controller System Manua*l, Edition 05/2003

[6] E.F. Camacho and C. Bordons, *Model Predictive Control*, Springer-Verlag, London, 2004.

[7] S.J. Qin and T.A. Badgwell, "A Survey of Industrial Model Predictive Control Technology," *Control Engineering Practice*, vol. 11, 2003, pp. 733-764.

[8] E. Miklovičová, and M. Mrosko, "Implementation of Predictive Control on Industrial Controllers," *AT&P Journal Plus*, 2010, pp. 39-43.

[9] P. Pivoňka and V. Mikšánek, "Real-Time Communication between MATLAB/Simulink and PLC via Process Visualization Interface," *Proc. of the 11th WSEAS International Conference on Systems*, Agios Nikolaos, Crete Island, Greece, July 2007, pp. 28-32.

[10] *Specifications of OPC,* OPC Foundation 1998-2010. Available: http://www.opcfoundation.org.

[11] M.H. Schwarz and 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.

[12] *MATLAB Help, MATLAB OPC Toolbox Help,* The Mathworks 2007. Available: http://www.mathworks.com.

[13] M. Mrosko, L. Mrafko and L. Körösi, "Real time control," *Proc. of the 9th International Conference Process Control 2010*, Kouty nad Desnou, Czech Republic.

[14] D.W. Clarke, C. Mohtadi and P.S. Tuffs, "Generalized Predictive Control – Part I. The Basic Algorithm. Part II. Extensions and Interpretations," *Automatica*, vol. 23, 1987, pp. 137-160.

Marián Mrosko was born in Poprad, Slovakia, in May 1979. He received the MSc degree in Automatic Control systems in 2003 and PhD. degree in Automation and Control in 2010, both from the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava.

He is currently an Assistant Professor at the Institute of Control and Industrial Informatics at the Faculty of Electrical Engineering and Information Technology, STU in Bratislava. His research interests are predictive control and real-time control of systems.

Eva Miklovičová was born in Piešťany, Slovakia, in May 1967. She received the MSc. dergree in Automatic Control Systems in 1990 and the PhD. degree in Automation in 1997, both from the Slovak University of Technology in Bratislava, Faculty of Electrical Engineering and Information Technology (STU).

She is currently Associated Professor at the Institute of Control and Industrial Informatics at the Faculty of Electrical Engineering and Information Technology, STU in Bratislava. Since 2011 she has been a vice-director of the Institute of Control and Industrial Informatics. Her research interests cover predictive control, adaptive control, system modeling and identification and networked control systems.