Real-time Model Predictive Control

MARIÁN MROSKO, EVA MIKLOVIČOVÁ
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-mail: marian.mrosko@stuba.sk, eva.miklovicova@stuba.sk,

Abstract: - In this paper two possibilities of real-time implementation of advanced control algorithms using the programmable logic controllers are presented and compared. In the first one the control law is designed and executed in another control system which is connected through the OPC communication to the PLC. The second possibility is direct implementation of predictive control algorithm in PLC using the available instructions.

Key-Words: real-time control, model predictive control, programmable logic controller, OPC

1 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. This type of controllers usually offers only simple control structures, such as on-off control or PID control loops. The wide development of control theory brought many advanced control methods with improved control performances, robustness or stability. The control laws of these methods usually differ from the control structures available in PLC, so their real-time implementation is more complicated.

In this paper the real-time implementation of one of the most popular advanced control design approaches, namely the model predictive control, by means of the PLC is dealt with. 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 [1]. MPC has proved its effectiveness in many industrial applications areas including petrochemical, chemical and food processing, automotive and aerospace industries [2].

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 controllers, such as PLC, which are more affordable. In [3] the model predictive control implementation based on the conventional PID controller structure has been proposed, provided that the plant model structure is properly chosen.

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 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 are analyzed.

The paper is organized as follows. First the OPC communication and the problems concerning the real-time experiments are presented. The model predictive control design is briefly described. Then two alternatives of the model predictive control real-time implementation based on the simple industrial controllers are proposed and experimentally evaluated on a simple laboratory plant.

2 OPC

OPC (originally OLE for Process Control) which stands for Object Linking and Embedding (OLE) for Process Control, is an industrial standard which is maintained by OPC Foundation [4] and widely used within the industrial automation to facilitate the interoperability of control devices from different manufacturers. The standard 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.

2.1 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.

2.2 OPC Toolbox

MATLAB® [5] 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™ [5] 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 an OPC server while simulating a system.

In [6] the OPC configuration has been described in detail and experimentally tested.

2.3 Real-time Experiments

Simulink's OPC toolbox uses a configuration block to specify the OPC clients to use in the model, to define the behaviour for OPC errors and events and to set the real-time behaviour. During the simulation, the model executes in pseudo real-time, matching the system clock as closely as possible by automatically slowing the simulation. However, 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. 1 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 is longer than 1 s [6].

Latency represents the sample time margin; the larger latency means larger margin, i.e. the sample time can be more decreased.

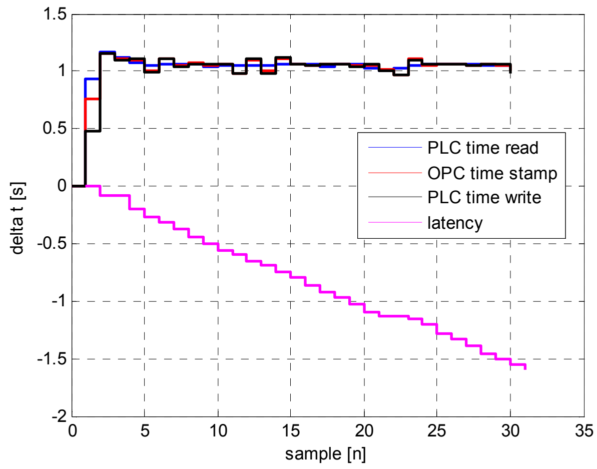


Fig. 1 Sample time and latency

There are three factors that would allow to get the execution time near the value set in the Simulink scheme:

1. Data transfer rate in the network (baudrate), usually measured in kbaud or Mbaud. It is the

amount of digital data transferred in a given time.

2. The number of OPC read and OPC write blocks in the Simulink scheme. Increase of the OPC read blocks number has less influence on the latency decrease than increase of the OPC write blocks number.
3. The number of items in OPC read and OPC write blocks; however the influence on the latency is not significant.

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 in relation to the number of communication items (the number of exchanged process and control variables) and taking into account the sufficient computational reserve in the case of the incidental processor overload by another process.

Testing or implementation of real-time control strategies in Simulink and OPC communication meets some restrictions that are critical for processes with fast dynamics. The order of block execution in Simulink is highly influenced by the time difference between the samples 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 to avoid the problems in implement-tation of real-time control systems

3 Model Predictive Control

3.1 Control Design

Generalized predictive control (GPC) proposed in [7] belongs to the most popular predictive 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 plant to be controlled in the discrete-time form

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

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

with

$$A(z^{-1}) = 1 + a_1 z^{-1} + \ldots + a_{na} z^{-na}$$
$$B(z^{-1}) = b_0 + b_1 z^{-1} + \ldots + b_{nb} z^{-nb}$$

$$\begin{aligned} C(z^{-1}) &= 1 + c_1 z^{-1} + \ldots + c_{nc} z^{-nc} \\ D(z^{-1}) &= 1 - z^{-1} \end{aligned} \tag{3}$$

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

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

subject to:

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

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 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.

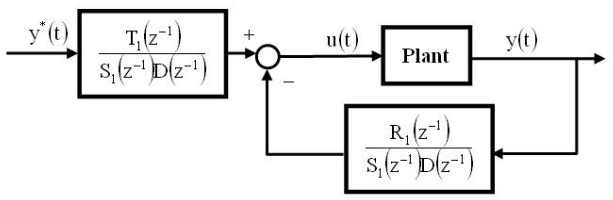
The GPC control law may be implemented using the standard pole-placement control structure (as shown in Fig. 2)

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

with

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

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



$y^*(t)$
$\frac{T_1(z^{-1})}{S_1(z^{-1})D(z^{-1})}$
+
$u(t)$
Plant
$y(t)$
–
$\frac{R_1(z^{-1})}{S_1(z^{-1})D(z^{-1})}$

Fig.2 Control scheme

Thus for the calculation of the control input at the time *t* only the output and control inputs available up to this time together with the reference value $y^*(t)$ are necessary. The orders of $R(z^{-1})$ and $S(z^{-1})$ 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 $R(z^{-1})$ and $S(z^{-1})$ 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 $T(z^{-1})$ 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

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

3.2 Control Implementation

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

$$u(t) = \frac{T(z^{-1})}{S(z^{-1})D(z^{-1})} r(t+1) - \frac{R(z^{-1})}{S(z^{-1})D(z^{-1})} y(t) \qquad (10)$$

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

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

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

$$\begin{aligned} SD(z^{-1}) &= S(z^{-1})D(z^{-1}) = \\ &= sd_0 + sd_1 z^{-1} + sd_2 z^{-2} \end{aligned} \qquad (13)$$

The control input at time t is then has the form

$$\begin{aligned} u(t) = {} & \frac{-sd_1.u(t-1) - sd_2.u(t-2)}{sd_0} + \\ & - \frac{y(t).r_0 + r_1.y(t-1) + r_2.y(t-2)}{sd_0} \\ & + \frac{t_0.r(t+1)}{sd_0} - \end{aligned} \qquad (14)$$

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.

4 Real-time Implementation of Model Predictive Control

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

4.1 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. More specifically, the MATLAB and Simulink with the OPC Toolbox can be used. In this software environment also the data processing and visualization can be realized.

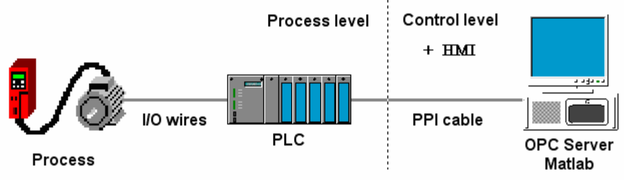


Fig.3 PLC + PC control

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.

4.2 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 coefficients, 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.

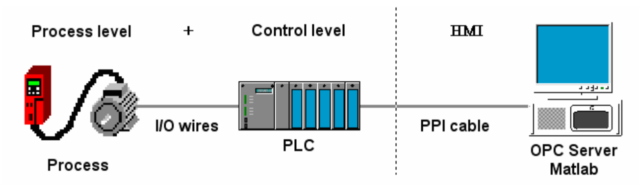


Fig.4 PLC control

5 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. 5) 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.

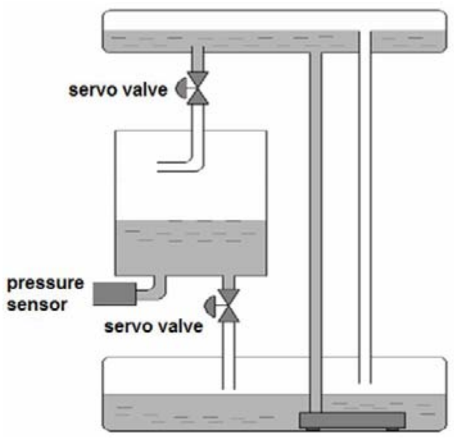


Fig. 5 Cylindrical laboratory tank

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

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

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

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

The real-time control results are shown in Fig. 6 and Fig. 7. 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.

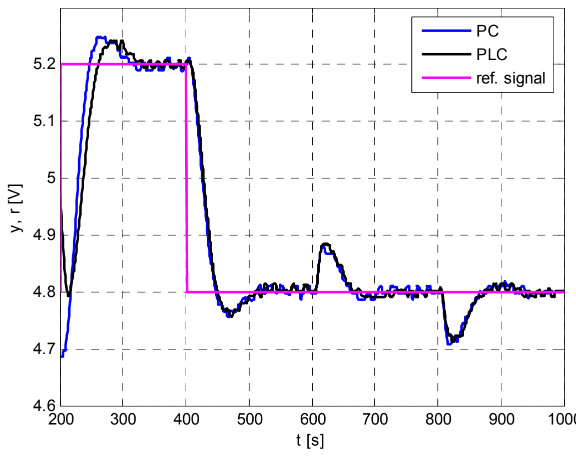


Fig. 6 Time responses of output and reference

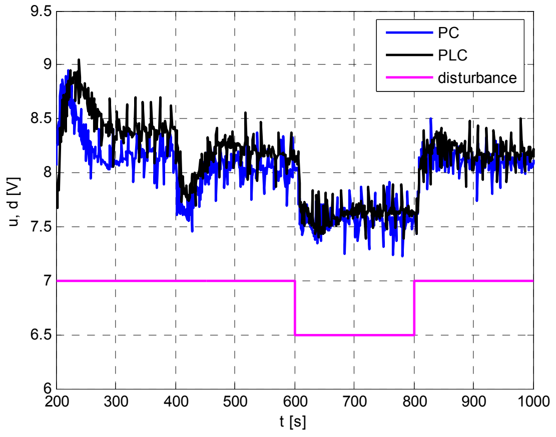


Fig. 7 Time responses of control input and disturbance

The difference in the output time responses in the beginning of simulation is due to the different initial conditions of experiments. In the further course of simulation, 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.

6 Conclusion

In this paper the real-time implementation of one of the most popular advanced control techniques has been dealt with. 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.

Acknowledgements
This work has been supported by the Slovak Scientific Grant Agency, Grant No. 1/0592/10 and by the Ministry of Education, Project No. Req-0048-0001.

References:

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

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

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

[4] OPC Foundation 1998-2010, Specifications of OPC, available on: http://www.opcfoundation.org.

[5] The Mathworks 2007, MATLAB Help, MATLAB OPC Toolbox Help, available on: http://www.mathworks.com.

[6] Mrosko, M., Mrafko, L., Körösi, L., *Real time control*, 9th International Conference Process Control 2010, Kouty nad Desnou, Czech Republic.

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

[8] S7-200 Programmable Controller System Manual, Edition 05/2003.