Real-Time Communication between MATLAB/Simulink and PLC via Process Visualization Interface

PETR PIVOŇKA, VOJTĚCH MIKŠÁNEK
Department of Control and Instrumentation
Brno University of Technology
Kolejní 4, 612 00 Brno
CZECH REPUBLIC

Abstract: - This paper shows real-time implementation of control algorithms from a simulation environment into the Programmable Logic Controller (PLC). Development of control algorithms is necessary for their use in a simulation environment and verification on simulation models. The real control system should be connected to the simulation environment with a PLC to test the control algorithms on a real physical model. In this case the PLC is used only as an IO device and all computations are performed in the simulation environment – MATLAB/Simulink. This connection is ensured via a general process visualization interface, which can transmit a data via different communication protocols: Ethernet, RS232, CAN or ProfiBus. Finally, the control algorithm should be implemented into the PLC and MATLAB/Simulink can be used for monitoring of the process input, output or others parameters. All implementation steps are shown on one of advanced control algorithms – the Generalized Predictive Control.

Key-Words: - PLC, Real-Time, MATLAB, Simulink, Predictive Control

1 Introduction

At first, the control algorithms is developed in a simulation environment (MATLAB/Simulink) and tested on simulation models. After that MATLAB/Simulink can be connected with a PLC, and the algorithm is tested on a physical model. This connection provides real-time communication between MATLAB/Simulink and the PLC (B&R 2005), which is an advantage for the development of control algorithms, especially extended and advanced control algorithms. Control algorithms have to be written in a universal programmable language supported by both MATLAB and PLC, because of its transmission into the PLC.

2 PLC B&R System 2005

The programming of the PLC is carried out using the B&R Automation Studio and several programming languages are available: Automation Basic, ANSI C and languages in international standard IEC 1131-3 (Ladder Diagram, Sequential Function Chart, Structured Text and Instruction List).

2.1 Process Visualization Interface

The PVI is a modular environment for operation systems MS Windows. This environment provides communication between PLC B&R and different user's applications, including B&R Automation Studio. The PVI can transmit a data via different communication protocols (RS232, Ethernet, CAN, etc.). By using the PVI the MATLAB/Simulink is able to have access to automation components.

2.2 CLIENT mk_pvi

The standard programmable tool (e.g. MS Visual C++) and libraries PVI enable us to make own tools for communication with PLC (clients). The client *mk_pvi*, which is used for real-time communication was developed in [3].

3 MATLAB/Simulink

MATLAB is a high level language and simulation environment for technical computing. MATLAB is foundation for Simulink – the graphical platform for simulation and model-based design for dynamic system. Besides the prepared blocks from Simulink toolboxes, it is possible to write user's own blocks and add them to Simulink models. Those blocks are called S-functions and can be written in MATLAB (M-file S-function) or in ANSI C (C MEX S-function). C MEX S-function has to be compiled to Microsoft Windows 32-bit linked libraries (DLL). Moreover, the control algorithm from C MEX S-function is directly transmittable into the PLC.

3.1 FIRST PART

The development of control algorithms can be divided in three parts. In the first part, the control algorithm is developed in a simulation environment MATLAB/Simulink in programmable user's blocks – S-functions. S-functions can be written in M-file or in

ANSI C. A big advantage of M-file S-functions is simplification of matrix algebra implementation although writing of control algorithms in C is also necessary because of its transmission into the PLC. In this part, the control algorithms are tested and verified on simulation models. The block diagram of the first part of implementation is shown in Fig. 1.

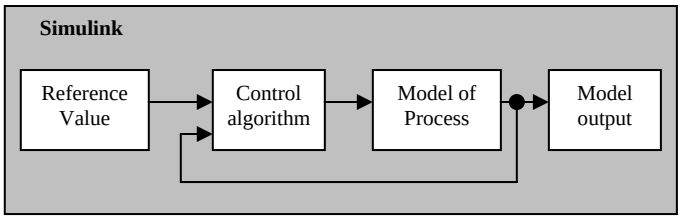

Fig. 1: Block diagram of the fist part of implementation

3.2 SECOND PART

After testing and verification of the control algorithm on a simulation model, it is necessary to test it on a physical model. For real-time communication between MATLAB/Simulink and the physical model controlled by the PLC the developed PVI client *mk_pvi* [3] is used. In this part of development the PLC B&R 2005 is used only as an IO card and MATLAB/Simulink receives a data from the physical model output and sends the action value to the physical model input. MATLAB/Simulink and PLC are connected via Ethernet. In this part, the S-function must be written in C. For debugging C MEX S-function Microsoft Visual C/C++ .NET [2] can be used. The second part of implementation is shown in Fig. 2.

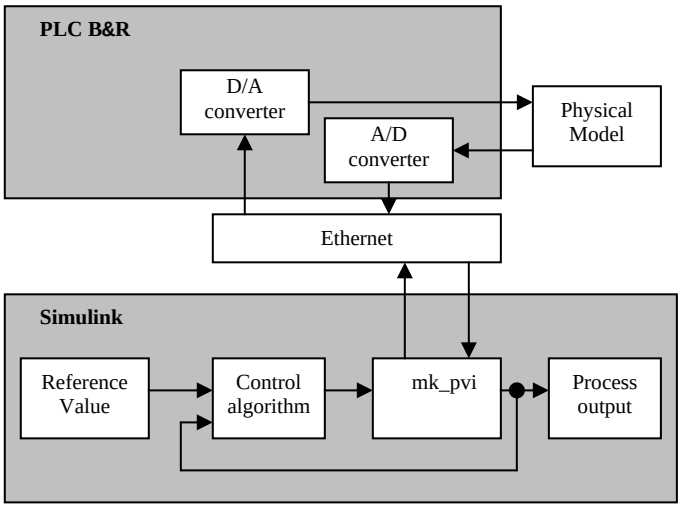

Fig. 2: Block diagram of the second part of implementation

3.3 THIRD PART

In the last step, the control algorithm is transmitted and the real model directly controlled by the PLC. The connection between MATLAB/Simulink and PLC can be used only for real-time monitoring of process inputs, process outputs or other control parameters. This monitoring of PLC by MATLAB/Simulink is shown in Fig. 3.

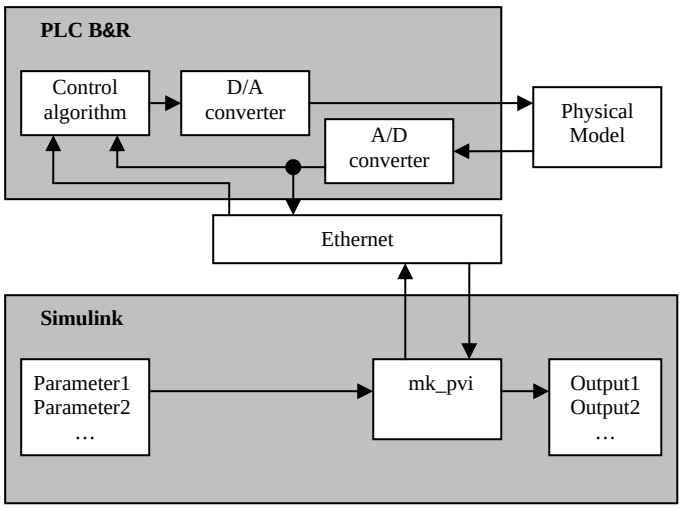

Fig. 3: Block diagram of the third part of implementation

Both S-functions and B&R Automation Studio Project are generally separable in two main procedures:

- `mdlInitializeSizes` (M-file S-function), `static void mdlStart` (C MEX S-function) and `_INIT void init` (PLC) procedures provide initialization of variables.
- `mdlOutputs` (M-file S-function), `static void mdlOutputs` (C MEX S-function) and `_CYCLIC void _cyclic` (PLC) procedures provide the control algorithm each sampling period.

Those two procedures are directly portable from C MEX S-function into PLC.

4 Simulation Experiment

In this part, the difference between the behavior of simulation and that of the physical models is shown. The simulation model was obtained by using Neural Network (NN). For training NN the Levenberg-Marquardt algorithm [8] was used. For comparison of both models one of the advanced control algorithms was chosen, especially the Model Predictive Control (MPC).

3.1 GENERALIZED PREDICTIVE CONTROL

The MPC is not a specific control strategy but an ample range of control methods where the control signal is obtained by minimizing an objective function. The Generalized Predictive Control algorithm is one of the most popular methods of predictive control, which consists in applying a control sequence that minimizes a cost function (1).

$$J(p,r,\lambda)=\sum_{j=1+d}^{p+d}\left[\hat{y}(t+j\,|\,t)-w(t+j)\right]^2+\lambda\sum_{j=1}^{r}\left[\Delta u(t+j-1)\right]^2 \quad (1)$$

where $\hat{y}(t+j\,|\,t)$ is the predicted system output in *j*-th prediction step in discrete time *t*, $w(t+j)$ is reference trajectory, $\Delta u(t+j)$ is *j*-th increment of control action, *p* is predicted horizon, *r* is control horizon, λ is cost constant and *d* is delay. The first term considers the predicted error and the second term considers penalized future control increments.

The criterion (1) can be rewritten to a matrix form [4]:

$$J=(\mathbf{G}\mathbf{u}+\hat{\mathbf{y}}-\mathbf{w})^{\mathrm{T}}(\mathbf{G}\mathbf{u}+\hat{\mathbf{y}}-\mathbf{w})+\lambda\cdot\mathbf{u}^{\mathrm{T}}\mathbf{u} \quad (2)$$

where $\hat{\mathbf{y}}$ is vector of predicted system outputs for prediction horizon, **w** is vector of future references.

$$\mathbf{u}=\begin{bmatrix}\Delta u(t)\\ \Delta u(t+1)\\ \vdots\\ \Delta u(t+r-1)\end{bmatrix}\qquad \hat{\mathbf{y}}=\begin{bmatrix}\hat{y}(t+d+1\,|\,t)\\ \hat{y}(t+d+2\,|\,t)\\ \vdots\\ \hat{y}(t+d+p\,|\,t)\end{bmatrix}$$

$$\mathbf{w}=\begin{bmatrix}w(t+d+1\,|\,t)\\ w(t+d+2\,|\,t)\\ \vdots\\ w(t+d+p\,|\,t)\end{bmatrix}$$

G is matrix of dynamics:

$$\mathbf{G}=\begin{bmatrix}\left.\frac{\partial f}{\partial u_0}\right|_{k=1} & \left.\frac{\partial f}{\partial u_1}\right|_{k=1} & \cdots & \left.\frac{\partial f}{\partial u_{r-1}}\right|_{k=1}\\ \left.\frac{\partial f}{\partial u_0}\right|_{k=2} & \left.\frac{\partial f}{\partial u_1}\right|_{k=2} & \cdots & \left.\frac{\partial f}{\partial u_{r-1}}\right|_{k=2}\\ \vdots & \vdots & \ddots & \vdots\\ \left.\frac{\partial f}{\partial u_0}\right|_{k=r-1} & \left.\frac{\partial f}{\partial u_1}\right|_{k=r-1} & \cdots & \left.\frac{\partial f}{\partial u_{r-1}}\right|_{k=r-1}\\ \vdots & \vdots & \ddots & \vdots\\ \left.\frac{\partial f}{\partial u_0}\right|_{k=p} & \left.\frac{\partial f}{\partial u_1}\right|_{k=p} & \cdots & \left.\frac{\partial f}{\partial u_{r-1}}\right|_{k=p}\end{bmatrix}$$

where $y=f(x_0,x_1,...,x_{r-1})$ and $k=1,2,...,p$, as shown in [6].

For linear causal systems can be **G** simplified:

$$\mathbf{G}=\begin{bmatrix}g_0 & 0 & \cdots & 0\\ g_1 & g_0 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ g_{r-1} & g_{r-2} & \cdots & g_0\\ \vdots & \vdots & \ddots & \vdots\\ g_{p-1} & g_{p-2} & \cdots & g_{p-r}\end{bmatrix}$$

where element g_j is j-th coefficient of model step response (3).

$$g_j=-\sum_{i=1}^{j}a_i g_{j-i}+\sum_{i=0}^{j}b_i \quad (3)$$

Cost function minimum (2) is obtained by making the gradient of *J* equal zero [4]. The result is equation (4), which is used for computation of the future control action increments vector.

$$\mathbf{u}=\left(\mathbf{G}^{\mathrm{T}}\mathbf{G}+\lambda\mathbf{I}\right)^{-1}\mathbf{G}^{\mathrm{T}}(\mathbf{w}-\hat{\mathbf{y}}) \quad (4)$$

But only the first increment of control action is used for control (5):

$$\Delta u(t)=\mathbf{k}(\mathbf{w}-\hat{\mathbf{y}}) \quad (5)$$

where k is the first row of the matrix $\left(\mathbf{G}^{\mathrm{T}}\mathbf{G}+\lambda\mathbf{I}\right)^{-1}\mathbf{G}^{\mathrm{T}}$. Only the first increment of control action is used for control (5). The close loop with GPC is shown in Fig. 4.

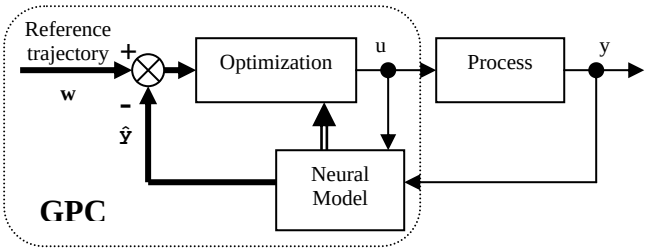

Fig. 4: Close loop with GPC controller.

3.2 APPLICATION OF GPC

The predictive control algorithm was written in ANSI C (C MEX S-function) and tested on a simulation model. The model is the cornerstone of GPC and it is obtained by using a Neural Network with the Levenberg-Marquardt training algorithm. The same neural network model is used as a model of process, too. To test of the control algorithm on a physical model, real-time communication is used between MATLAB/Simulink and PLC is via Ethernet using the *mk_pvi* client. In folowing pictures the influence of cost constant λ on regulation is shown. Fig. 5 shows control of the simulation model and Fig. 6 shows control of the real process.

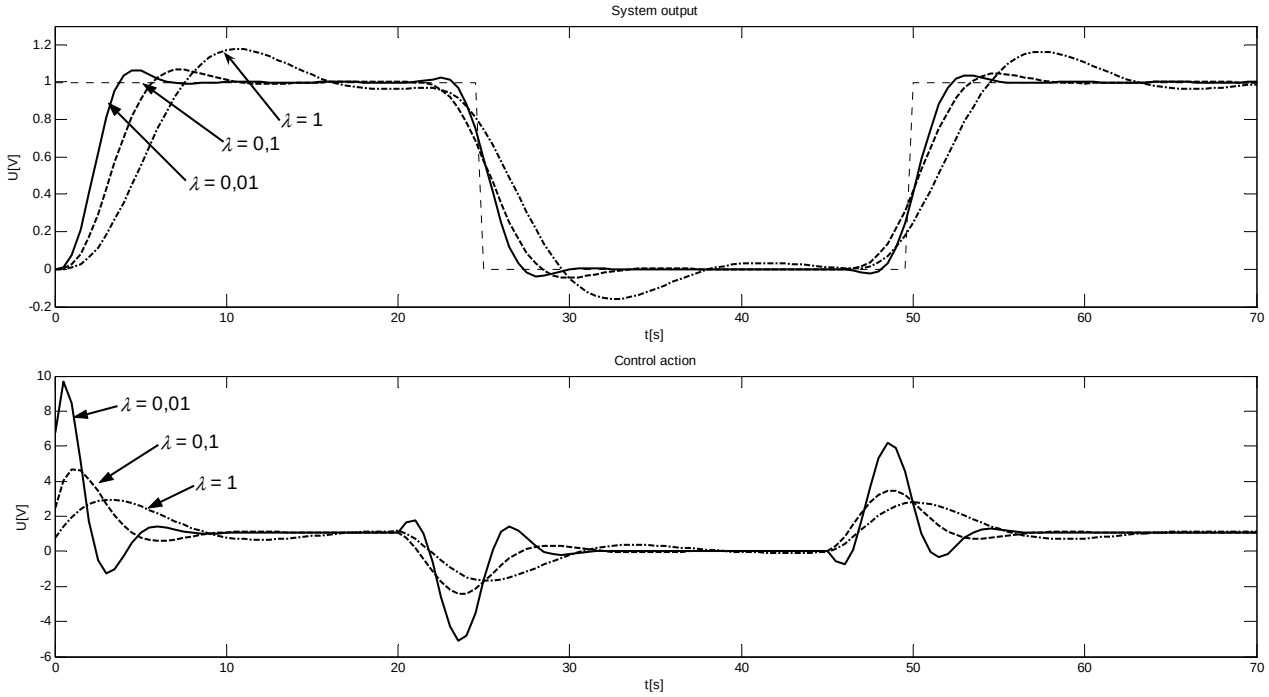

Fig. 5: GPC controller implemented in MATLAB/Simulink controls simulation model of real process.

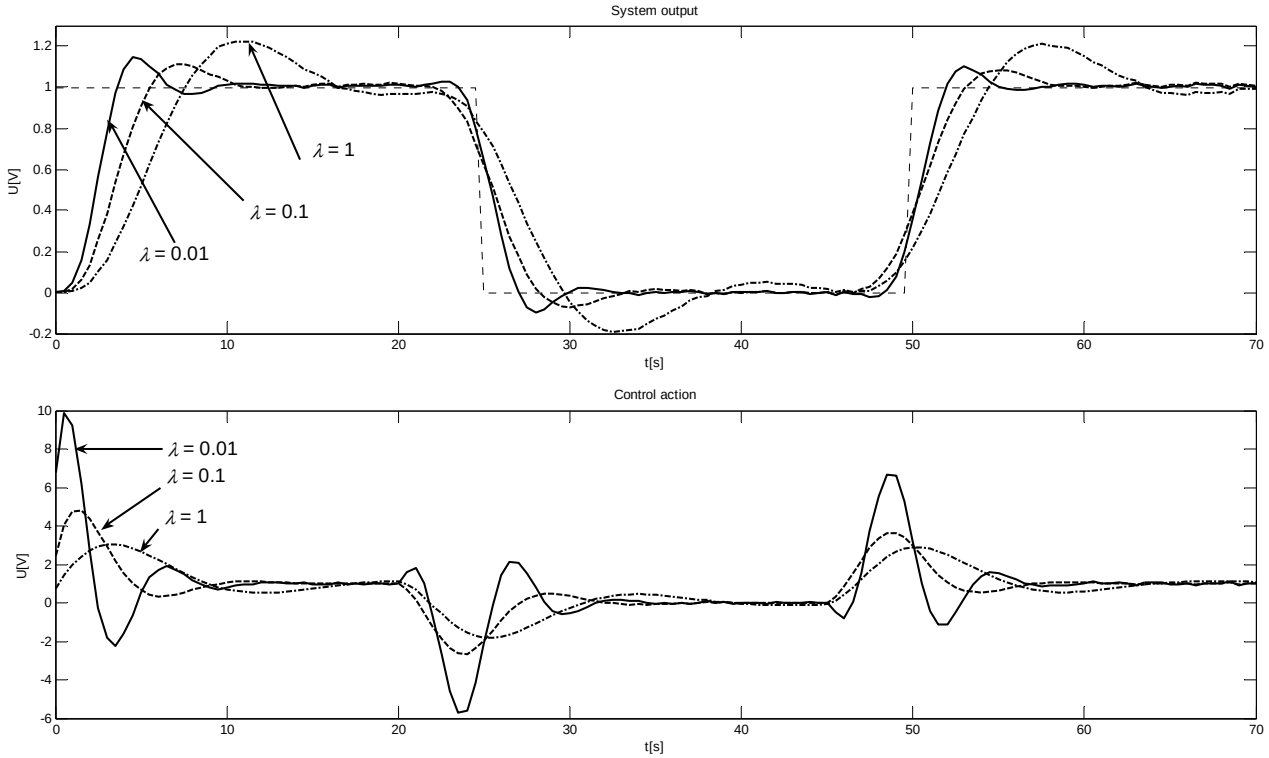

Fig. 6: GPC controller implemented in MATLAB/Simulink real-time controls real process.

4 Conclusion

In this paper is shown the way of development of control algorithms from simulation environment to direct implementation to the programmable logic controller. The simulation environment MATLAB/Simulink provides a solution to writing user's own blocks. The controls algorithms can be written in those blocks are called S-function which can be written in MATLAB (M-file S-function) or ANSI C (C MEX S-function). The control algorithm is directly portable from C MEX S-function into PLC B&R 2005.

Using the *mk_pvi* client is possible to control a real process by a control algorithm implemented in MATLAB/Simulink. In this case the MATLAB/Simulink is real-time connected with the PLC which is used only as an IO device. Another advantage of the real-time connection between MATLAB/Simulink is using of existing control blocks from MATLAB toolboxes. Using the existing control algorithms from the toolboxes is quick way to compare them with own algorithm on real process.

Verification of the control algorithm is necessary because of different behavior of the simulation model of real process and the real process. In this case the difference can be seen as a bigger overshot in real process as is shown in Fig. 5 and Fig. 6.

Acknowledgement:
The paper has been prepared as a part of the solution of Czech Science Foundation GAČR project No. 102/06/1132 Soft Computing in Control and by the Czech Ministry of Education in the frame of MSM MSM0021630529 Intelligent Systems in Automation.

References:

[1] B&R SYSTEM 2005 – *User's Manual*, http://www.br-automation.com, 2003

[2] The MathWorks, *Online Documentation*, http://www.mathworks.com

[3] Kořínek, V, *Communication MATLAB-Ethernet*, Diploma Thesis, BUT FEEC 2004

[4] Camacho, E. F., Bordons, C., *Model Predictive Control*, London, Springer 1999, ISBN 3-540-76241-8

[5] Clarke, D. W., Mohtadi, C., Tuffs, P. S, *Generalized Predictive Control – Part I. The Basic Algorithm*, Automatica, 23, s. 137-148 (1987)

[6] Vychodil, H., *Generalized Predictive Control with a Non-linear Autoregressive Model*. Automatic Control Modeling and Simulation ACMOS'05. Praha: WSEAS, 2005, pp. 85 - 89, ISBN 960-8457-12-2

[7] Švancara, K., Pivoňka, P., *The Real-Time Communication Between MATLAB and the Real Process Controlled by PLC*, TMT 2003, Lloret de Mar, Barcelona, Spain, pp. 1077 - 1080, ISBN 9958-617-18-8

[8] Hristev, R. M.: *The ANN Book*, GNU Public License, 1998