Slovak University of Technology in Bratislava
Institute of Information Engineering, Automation, and Mathematics

PROCEEDINGS

of the 18th International Conference on Process Control

Hotel Titris, Tatranská Lomnica, Slovakia, June 14 – 17, 2011

ISBN 978-80-227-3517-9

http://www.kirp.chtf.stuba.sk/pc11

Editors: M. Fikar and M. Kvasnica

Rauová, I., Valo, R., Kvasnica, M., Fikar, M.: Real-Time Model Predictive Control of a Fan Heater via PLC, Editors: Fikar, M., Kvasnica, M., In *Proceedings of the 18th International Conference on Process Control*, Tatranská Lomnica, Slovakia, 388–393, 2011.

Full paper online: http://www.kirp.chtf.stuba.sk/pc11/data/abstracts/080.html

Real-Time Model Predictive Control of a Fan Heater via PLC

Ivan Rauová *, **Richard Valo** *, **Michal Kvasnica** *,
Miroslav Fikar *

* *Institute of Automation, Information Engineering and Mathematics,
Slovak University of Technology in Bratislava,
812 37 Bratislava, Slovakia,
http://www.kirp.chtf.stuba.sk
{ivana.rauova,richard.valo,michal.kvasnica,miroslav.fikar}@stuba.sk*

Abstract: This paper deals with real-time implementation of Model Predictive Control (MPC) of a fan heater system using Programmable Logic Controller (PLC) platform. The MPC problem is solved using parametric programming techniques, which encode the optimal control moves as a lookup table. The challenge then becomes how to implement such a table on a memory-restricted device. The proposed design procedure is illustrated on real-time control of a laboratory heat exchange plant.

Keywords: model predictive control, programmable logic controllers

1. INTRODUCTION

Model predictive control (MPC) is an attractive approach widely used in industry to control a broad range of the systems due to its ability to provide optimal performance while taking process constraints into account (Maciejowski, 2002). In MPC the control objectives are translated into an optimization problem, which is formulated over a finite prediction horizon. The result of the optimization is a sequence of optimal control moves which drives system states towards a given reference point while respecting system constraints (such as upper and lower limits on the inputs and states) and optimizing a selected performance criterion. Traditionally, MPC is implemented in a so-called Receding Horizon fashion where the optimal control move is achieved by solving optimization problem in each time instance for a newly measured state. This approach induces lot of computation load at each sampling time, which might be prohibitive if not enough computation power is available or if sampling time is too short.

If less powerful control platforms are employed, additional care has to be taken to respect real-time constraints. One approach to decrease computational load involved in obtaining of optimal control action u^* for a particular value of x is to "pre-compute" the optimal solution to a given optimization problem for all possible initial conditions of x using parametric programming techniques (Bemporad et al., 2002). The optimal control can be then found as an explicit function $u^*(x)$ mapping the states to the control inputs. The function is computed off-line and takes a form of lookup table. Implementation of such a table can be done very efficiently on-line, as the evaluation of the feedback law involves only matrix multiplications, additions and logic comparisons. As a consequence, real-time implementation of such an *explicit* MPC can be done much faster compared to traditional on-line MPC fashion.

In this work we aim at implementing explicit MPC on a Programmable Logic Controller (PLC) restricted to 1024 bytes of memory. Three factors determine whether the design procedure will be successful:

(i) whether it is possible to construct the explicit MPC controller off-line in an automated fashion;
(ii) whether the controller is reasonably small as not to exceed the memory capabilities of the PLC;
(iii) whether the controller can be implemented using programming instructions which the control device understands.

In the paper we illustrate how to synthesize the parametric solution to MPC optimization problem using the Multi-Parametric (MPT) Toolbox (Kvasnica et al., 2004) and how to implement it on a PLC. First, we introduce the controlled plant and derive its mathematical model. Then we show which commands have to be used to set up the MPC optimization problem and how to synthesize the explicit MPC controller using MPT. Having a lookup table we introduce an algorithm capable to transform it into a binary search tree (BST), which can be downloaded directly to the PLC. At the end we show results of the laboratory plant control using explicit MPC controller provided on the PLC.

2. PHYSICAL SETUP

2.1 Controlled Plant

The laboratory Air-stream and Temperature Control Plant LTR 700 (Svetíková et al., 2003) is produced by the German company Amira. It consists of a fan, a heating coil, a differential pressure sensor, a temperature sensor, and an actuator box.

This plant is designed for heating the entering medium. Commonly, air is the medium which is intake to the plant

thanks to the fan. The entering air is further heated by heating coil. In order to obtain hot air of desired amount and temperature, we can manipulate the fan speed or the amount of heat generated by heating coil. In this work, the heating coils output is set to constant value (50%). We aim at controlling only the air-flow rate (manipulated variable) by fan speed (control variable).

The plant is schematically illustrated in Fig. 1. Actual temperature and air-flow can be measured by sensors (TI, FI).

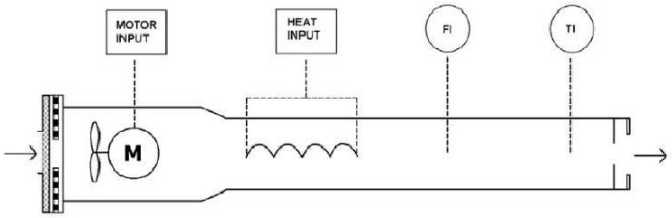

Fig. 1. Sensors in the air heater.

It is possible to implement several control configurations ranging from simple feedback loops, through cascade loops, up to multivariable control with two inputs and two outputs.

Airflow The plant converts air-flow value to a current signal in a range of 10.4-20 mA. This output signal is connected to the 3rd input of analog I/O module which is shown in Fig. 2. For our control program, the connected signal is converted into an integer number with physical address $AIW4$ (A – analog, I – input of the PLC, W - memory size of 16 bits, $4-3^{\text{rd}}$ input to PLC). This integer is converted into a corresponding quantity expressed in mA units by the following relation:

$$x_{[\text{mA}]} = \frac{AIW4 - 0.596}{1583.026}, \tag{1}$$

Corresponding value of the state in a percentage range is achieved by

$$x_{[\%]} = \frac{x_{[\text{mA}]} - 10.4}{0.096}, \tag{2}$$

Fan speed Formula which relates actual control action $u_{[\%]}$, expressed in percentage rage, and a current signal output to the fan engine in mA units is

$$u_{[\text{mA}]} = 0.2u_{[\%]}, \tag{3}$$

Therefore, corresponding integer representation of the output is

$$AQW0 = 1585.917u_{[\text{mA}]} + 51.23, \tag{4}$$

where $AQW0$ is a physical address (A – analog, Q – output of the PLC, W – memory size of 16 bits, $0-1^{\text{st}}$ output from PLC). This input signal is connected to the output of analog I/O module which is shown in Fig. 2.

The mathematical model of the fan airflow can be captured by one differential equation of the following form

$$\frac{\mathrm{d}V}{\mathrm{d}t} = kD_M^3 f - V \tag{5}$$

Here, V represents volume of air flow, k denotes a proportionality coefficient as a function of the Reynolds number of blender, D_M is blender diameter and f represents blender rotation frequency. By linearizing (5) around the steady state f^s and V^s, the following transfer function model can be derived

$$G = \frac{kD_M^3}{s+1} = \frac{Z}{T_v s + 1}, \tag{6}$$

where Z denotes gain of the system and T_v represents time constant.

Corresponding state-space representation of the fan is

$$\dot{x} = Ax + Bu \tag{7a}$$

$$y = Cx + Dy \tag{7b}$$

where $x = V - V^s$ is the state and $u = f - f^s$ is the input. Based on the steady state values of the variables mentioned above, the matrices are defined by

$$A = -\frac{1}{T_v},\ B = \frac{1}{Z},\ C = 1,\ D = 0.$$

This linear state-space representation can then be used to find a closed-form representation of the MPC feedback law by using techniques of parametric programming as described in Section 3.

2.2 PLC

A programmable logic controller (PLC) is a special digital computer often used in process automation such as for control of machinery on factory assembly lines, amusement rides, or lighting fixtures. Unlike general-purpose computers, PLCs are designed for multiple input and output arrangements, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact. Programs to control machine operation are performed in constant length cycles (Siemens, 2008).

In this work, we have used the SIMATIC S7-200 micro PLC, which is exceptionally compact, remarkably capable, fast and comes with easy to operate hardware and software. It has a modular design, still open-ended enough to be expanded. The main components of the selected PLC are briefly described next.

CPU The S7-222 CPU can be seen on the left side in Fig. 2. Important to notice is that the CPU only provides 1024 bytes of memory for program data. This limit is both restrictive and challenging from the control synthesis point of view. Another limitation is that control algorithms have to be developed using so-called *ladder logic*, a visual programming language which only requires the algorithm to be composed of most basic operations (e.g. sums, products, comparisons, etc.).

Real-time data measurements can be stored on a memory cartridge (Siemens, 2008), marked by "MC" in Fig. 2. In our case it provides a 256 kB storage for measured signals, which could be captured at a 0.04 second sampling rate. The captured data can subsequently be opened in MS-Excel.

Power Source Power source (LOGOPower 6EP 1332-1SH42) (Siemens, 2010, 2008) is a standard transformer used to supply PLC from public power network (see Fig. 2), where the "Power Network" is 1-phase AC line supply with voltage rate of 100-240 V (50/60 Hz) to isolated output voltage 24 V DC.

Analog I/O Module Module EM 235 is necessary for the PLC to communicate with the controlled plant by means of analog signals. The module is shown in Fig. 2, where it is situated in between the central processor unit and the power source. The module allows 4 analog inputs and one analog output to be connected. Inputs have to be connected and configured corresponding way. The outputs in addition have to have correct HW configuration. For setup the outputs to the 0-20 mA we have to set a set of 6 switches. The set is placed right bottom corner of the EM 235 in Fig. 2. For our purpose we have set them as in Fig. 3. The PLC communicates with the master PC by a 9 pin RS 486 port, located under the memory cartridge.

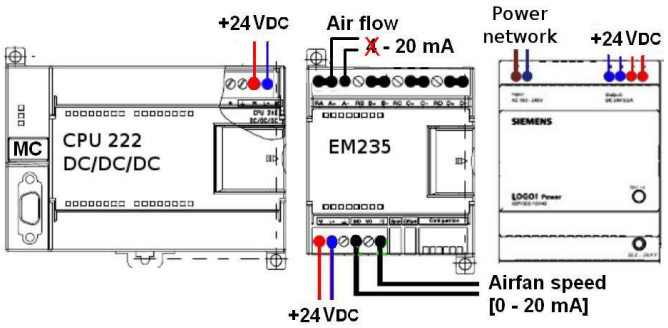

Fig. 2. PLC connections to the plant.

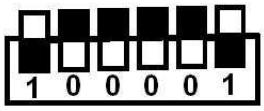

Fig. 3. Setting of the EM 235 inputs.

3. EXPLICIT MODEL PREDICTIVE CONTROL

In model predictive control (MPC), the optimal control actions are found by optimizing for plant behavior while taking process constraints into account. This is always achieved by formulating and solving an optimization problem where a given objective function is minimized subject to the constraints. A model of the plant is employed as an additional constraint to capture the predicted evolution of the plant:

We consider the discrete-time, stabilizable linear time-invariant model given by the state-space representation

$$x(k+1) = Ax(k) + Bu(k), \tag{8a}$$
$$y(k) = Cx(k). \tag{8b}$$

Here, $x(k)$ denotes state at time instance k, $x(k+1)$ is a successor state, $u(k)$ is control input, and $y(k)$ is a system output. It is assumed that variables are constrained by upper and lower limits

$$\underline{x} \leq x(k) \leq \overline{x}, \qquad \underline{u} \leq u(k) \leq \overline{u}, \qquad \underline{y} \leq y(k) \leq \overline{y} \tag{9}$$

For the system (10) consider now the constrained finite-time optimal control problem

$$\min_{\Delta u_0,\ldots,\Delta u_{N-1}} \sum_{k=0}^{N-1} \| R\Delta u_k \|_p + \| Q(y - y_{\text{ref}}) \|_p \tag{10a}$$
$$\text{s.t. } x_0 = x(t), \tag{10b}$$
$$x_{k+1} = Ax_k + Bu_k, \tag{10c}$$
$$y_k = Cx_k, \tag{10d}$$
$$x_k \in \mathcal{X}, \tag{10e}$$
$$y_k \in \mathcal{Y}, \tag{10f}$$
$$u_k \in \mathcal{U}. \tag{10g}$$

Here p denotes a matrix norm (either $p = 1$, $p = 2$ or $p = \infty$), the integer N represents the finite prediction horizon, and R, Q are weighting matrices used to tune performance of the MPC controller. The linear model in (10c) serves to predict the future states based on the knowledge of the initial state $x(t)$, which is assumed to be available at each time instance. The optimization is performed over the increments Δu_k to provide offset-free tracking of the reference trajectory y_{ref}. The state, output and input constraints represented, respectively, by the polyhedral sets $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{U}$.

MPC is usually implemented in so-called receding horizon fashion. Here the optimal solution to the problem (10) is found for particular value of $x(t)$, which results into the optimal sequence $[\Delta u_0^*, \ldots, \Delta u_{N-1}^*]$. Here $\Delta u_k = u_k - u_{k-1}$ is the control action increment, used to introduce integral action. Out of this sequence, only the first element (i.e. Δu_0^*) is actually implemented to the plant and the rest is discarded. At the next time instance, a new initial state measurement $x(t)$ is obtained and the whole procedure is repeated. This repetitive optimization is performed in order to introduce feedback control into the procedure and to deal with possible disturbances and plant model mismatches.

One can write new control action as $u_k = u_{k-1} + \Delta u_k$ which function of previous known control action and control action increment. The optimization problem (10) than can be rewritten into the state-space formulation with extended state feedback $\widetilde{x}_k = [x_k, u_{k-1}]^T$. Therefore, state-space representation reduces number of optimization variables from two (u_k and Δu_k) to one (Δu_k). The time-varying reference is constrained to be $y_{\text{ref},k+1} = y_{\text{ref},k}$, hence, state vector can be extended into three states $\widetilde{x}_k = [x_k, u_{k-1}, y_{\text{ref},k}]^T$ and state-space representation is defined in the form:

$$\widetilde{x}_{k+1} = \begin{bmatrix} A & B & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \widetilde{x}_k + \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix} \Delta u_k = \widetilde{A}\widetilde{x}_k + \widetilde{B}\Delta u_k \tag{11a}$$
$$y_k = [C \;\; D \;\; 0]\, \widetilde{x}_k + D\Delta u_k = \widetilde{C}\widetilde{x}_k + \widetilde{D}u_k \tag{11b}$$

with all the matrices of appropriate dimensions.

If the initial state $x(t)$ and the value of the reference signal y_{ref} are known, the problem (10) can be solved as a quadratic problem (QP) for $p = 2$ and for $p = 1$ or $p = \infty$ as a linear program (LP). Even though efficient polynomial-time algorithms exist to solve both type of problems, the time needed to perform the optimization can be prohibitive of the sampling time is too short, or if the implementation hardware is very simple and thus less capable. To address this issue, in their seminal work (Bemporad et al. (2002)) have shown (for a quadratic type

of performance induces) how to solve the problem (10) parametrically for all admissible initial conditions $x(t)$ by employing techniques of parametric programming. In this approach the optimal solution to (10) is found as an explicit state feedback law parametrized in the initial condition $x(t)$. The advantage of the parametric solutions is that the optimal control input can be obtained in real-time by simply evaluating a look-up table. The main result of the parametric approach is summarized by the following theorem.

Theorem 3.1. (Bemporad et al. (2002)). The optimal solution to the problem (10) is a piecewise affine function of the initial state x_0

$$\Delta u_0^* = F_r x_0 + g_r \qquad \text{if} \qquad x_0 \in \mathcal{R}_r \tag{12}$$

where $\mathcal{R}_r = \{x_0 \mid H_r x_0 \leq K_r\}$ are polytopic regions of the state space, and F_r and g_r are the matrices of the affine state-feedback law active in the r-th region.

Theorem 3.1 shows that the optimal solution for the problem (10) can be found as a look-up table consisting of r components. Therefore, once the table is calculated, MPC can be implemented in a real time by simply evaluating the table for the actual measurements of $x_0 := x(t)$. The table can be calculated efficiently using, e.g. the Multi-Parametric Toolbox (Kvasnica et al., 2004). Performance of the MPC scheme can be tuned by appropriately adjusting the weighting matrices Q and R, and by a suitable choice of prediction horizon N.

4. IMPLEMENTATION ON PLC

As already mentioned, typical PLCs have severe memory limitations. Our PLC, in particular, only allows 1024 bytes of memory storage. A special care has thus to be taken when evaluating the explicit MPC feedback (encoded as a lookup table composed of feedbacks F_r, g_r, and regions $\mathcal{R}_r$) on such a device. To perform this task efficiently, we employ the *binary search tree* (BST) algorithm.

The basic idea of BST algorithm is to hierarchically organize the controller regions into a tree structure where, at level of the tree, the number of regions to consider is decreased by a factory of two. Therefore the table traversal can be performed in time logarithmic in the number of regions. The tree is constructed in an iterative fashion. At each iteration an optimal separating hyperplane $h_i x(t) \leq k_i$ is selected such that the set of all regions processed at the i-th iteration is divided into two smaller subsets: regions $\mathcal{R}_i^+$ residing on one side of the hyperplane and $\mathcal{R}_i^-$ on the other side. A new node in the tree is then created which contains information about the hyperplane and two pointers to child nodes. The left child is created by recursively calling the algorithm for regions $\mathcal{R}_i^+$ and the right child for the regions $\mathcal{R}_i^-$. The exploration of a given tree branch stops when no further subdivision can be achieved. In such a case a leaf is created which points to the region which contains $x(t)$. The resulting tree is then composed of the set of separating hyperplanes linked to the actual regions through a set of pointers.

To be able to use a BST-encoded tree on a PLC, the tree is transformed into a so-called "data-block". In this data table, first M entries represent one hyperplane and pointers to next line which should be explored. Obtaining

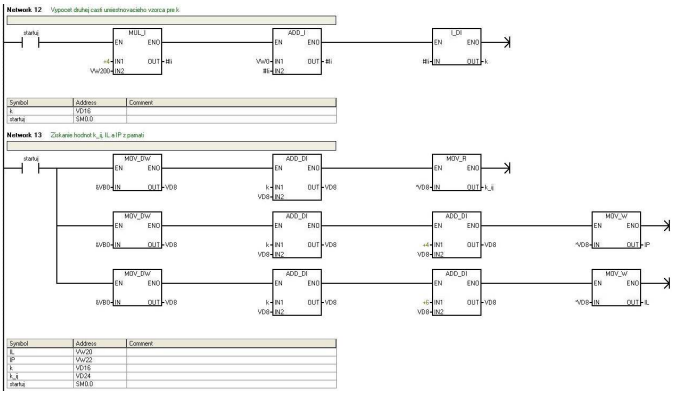

Fig. 4. A short excerpt of the LAD implementation of Algorithm 1

the optimal control action for a particular value of x then reduces to traversing the binary search tree using Algorithm 1.

Algorithm 1 Table traversal via binary search tree

INPUT: BST tree composed of separating hyperplanes $h_i x \leq k_i, i = 1, \ldots, M$ and linked nodes, state measurements $x(t)$
OUTPUT: Optimal control input $u^*(x)$

```
r ← 1
repeat
  if h_r x ≤ k_r then
    r ← index of the left child node (negative index)
  else
    r ← index of the right child node (positive index)
  end if
until r is a leaf node (positive index).
u_0^*(x(t)) = F_r x(t) + g_r
```

The PLC version of Algorithm 1, implemented using the Ladder Logic (LAD) programming language, is universal and can process any kind of lookup tables described by binary search trees. The LAD diagram consists of several routines and subroutines, a short excerpt of which is shown in Fig. 4. The program allocates 74 bytes of global memory in main routine and at most 34 bytes of temporary memory in subroutines. The total amount of memory allocated for controller is 874 bytes, the rest (150 bytes) remains to user.

5. EXPERIMENTAL RESULTS

In this section we show how MPC could be used for control of the fan heater described in the Section 2 using PLC. The control objective is to drive the volume of air flow to a time varying reference y_{ref} while respecting motor capacity $0\% \leq f \leq 100\%$ and volume of the air flow $0\% \leq V \leq 100\%$. The following mathematical model of the fan heater was obtained using identification methods:

$$G = \frac{5.12}{0.4726s + 1} \mathrm{e}^{-0.3s}. \tag{13}$$

MPC synthesis using the Multi-Parametric Toolbox begins with a definition of the prediction model:

```
>> A=-0.1953; B=0.0923; C=1; D=0;
>> fan=ss(A,B,C,D)
>> Ts=0.25
>> model=mpt_sys(fan,Ts)
```

where the model is converted into the discrete-time domain using sampling time T_s. Time delay of the model can be omitted as it is less than the time constant and the real time verification proved such model to be satisfactory. Next, constraints on state, input and output are defined:

```
>> model.umax=100-fs; model.umin=0-fs;
>> model.xmax=100-Vs; model.xmin=0-Vs;
>> model.ymax=100-Vs; model.ymin=0-Vs;
```

Notice that the constraints are imposed on the deviation variables with linearization points $f^s = 15\%$ and $V^s = 30\%$.

Once the model is complete, parameters of the MPC problem to be solved could be defined by

```
>> problem.R=1; %penalty on u_k
>> problem.Q=1; %penalty on x_k
>> problem.Qy=1000; %penalty on (y_k-y_ref)
>> problem.N=5; %prediction horizon
>> problem.norm=2; %use quadratic cost
>> problem.tracking=1; %use time-varying
    reference
```

Values of the penalty matrices R and Q_y were chosen with respect to allowed number of regions (which reflect the memory footprint of the controller). The number of regions can be reduced by lowering R and increasing the value of Q_y. The upper bound on the number of regions is 26 for a controller with 1 state and 1 input, otherwise the controller footprint would exceed 1 kB.

Finally, the parametric solution to problem (10) can be calculated as a lookup table using the command

```
>> ctrl=mpt_control(model,problem)
```

Result of the composition is, in this case, a lookup table consisting of 25 regions in a 3D state-space. State-space representation used in the controller consist of the following matrices

$$\widetilde{A} = \begin{bmatrix} 0.8226 & 0.3199 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \widetilde{B} = \begin{bmatrix} 0.3199 \\ 0 \\ 0 \end{bmatrix} \tag{14a}$$

$$\widetilde{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \widetilde{D} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{14b}$$

BST tree is constructed from the lookup table using the MPT command

```
>> tree=mpt_searchTree(ctrl)
```

In our case, tree consists of 25 nodes in 7 levels, which corresponds to 724 bytes of memory. Selected parts of the data-block are depicted in Fig. 5. The number of values in one line corresponds with number of the state variables and constant.

The data are subsequently downloaded to the PLC, which then executes the table traversal at each sampling instance based on the measurements of the states. When the region for the actually measured state is found, Algorithm 1 is executed and the corresponding control input to the system is calculated as

```
// h_i, k_i, I_positive (h*x >= k), I_negative (h*x < k)
VD216   0.014993305868, 0.000000000000, 0.999887594072, 70.303400479304
VW232   -2,-3
VD236   -0.000172245543, -0.000023729104, 0.999999984884, 69.987020388607
VW252   -4,-5
VD256   -0.031289541095, 0.000555538228, -0.999510208050, 27.866553742954
VW272   -6,-7
VD276   -0.007172535481, 0.000094602731, 0.999974272562, 69.499794876778
VW292   -8,-9
VD296   -0.634115037629, 0.002361650321, 0.773235114089, 21.519222553989
VW312   -10,-11
VD316   0.077584962071, 0.008448286942, -0.996949948647, 24.734743662862
VW332   -12,-13
VD336   0.023194184544, 0.004509499517, -0.999720808135, 26.339590462389
VW352   -14,-15
VD356   -0.643222433078, 0.002393804802, 0.765675630593, 20.524844140314
VW372   13,10
VD376   -0.000382795506, -0.000020170019, 0.999999926530, 69.972416000382
VW392   -16,7
  .
  .
  .
  .
  .
// F_i, G_i
VD716   -2.571702741900, -1.000000000000, -0.000000000000, 218.847680947400
VD732   -2.551297916900, -0.996032818100, 3.877595031500, 20.925234301400
VD748   -2.548395788500, -0.995468575500, 3.630187328100, 14.321778179900
VD764   -2.542212686100, -0.994266433800, 3.363681899900, 7.300173971700
VD780   -2.504735063800, -0.990672007000, 3.075256534800, -1.470205824300
VD796   -2.504735022400, -0.990672007500, 3.075986170100, -1.521283174400
VD812   -2.504727020800, -0.990671583200, 3.054245316400, 0.000000000000
```

Fig. 5. A short excerpt of the data-block provided on the PLC.

$$u_k = u_{k-1} + \Delta u^* \tag{15}$$

with state-feedback law $\Delta u^* = F_i \widetilde{x}_k + g_i$, where $\widetilde{x}_k$ is state variable of the system (11).

The data-block representing the controller was downloaded to the PLC to perform real-time experiments. First, ability of the controller to follow a time-varying reference, where user can change a setpoint at any time, is documented in Figs. 6–7.

System response near the and upper bound is without oscillations within a reasonable settling time, while response around the middle and near the lower bound has longer settling time. Such response can be caused by different behavior of the plant throughout the state ranges. That means, several models are necessary to describe plant behavior sufficiently. Therefore, possibility how to eliminate oscillations, is to control the plant as a hybrid system, which is not possible due to restricted amount of the memory.

To reduce long settling time, one can approximate time delay in model (13) by Taylor series or Padé approximation. This approach results in the better but more complex model with more regions, thus impossible to apply on PLC. The maximum amount of regions is function of the number of state variables. Therefore, if one wants more regions, either a simpler model is required or region reduction techniques have to be employed (Kvasnica et al., 2011; Kvasnica and Fikar, 2010).

Real data presented in Figs. 6–7 show that the MPC controller utilizes the predictions to change the value of the input signal in the same period as the reference was changed, such that output signal is steered towards this reference. Experiment also proved that MPC controller with time-varying reference can be implemented on the PLC in a real time.

6. CONCLUSIONS

In this paper we have shown how MPC can be implemented on a programmable logic controller with severe limitations on allowed memory storage. The approach was

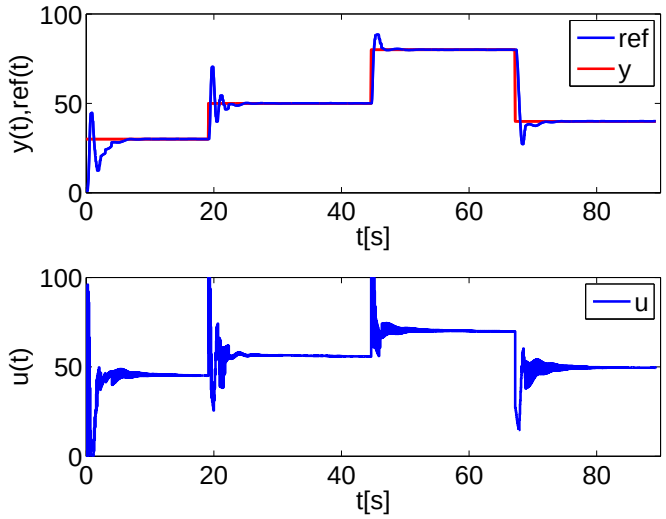

Fig. 6. Control of the fan heater tracking time-varying reference.

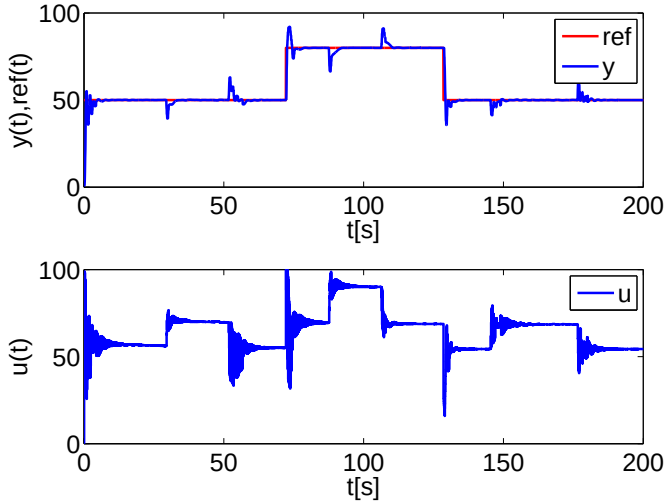

Fig. 7. Control of the fan heater with disturbance during the tracking of time-varying reference.

based on the pre-calculating the solution to the MPC optimization problem just once, for all possible initial conditions. The result is then given in the form of a lookup table. Such a table was subsequently encoded as a binary search tree for its efficient evaluation in real time. Experimental results confirm that the controller provides satisfactory performance while respecting design constraints.

ACKNOWLEDGMENT

Authors gratefully acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under the grant 1/0095/11, and the Slovak Research and Development Agency under the project APVV-0029-07. This work was also financed by a grant (No. NIL-I-007-d) from Iceland, Liechtenstein and Norway through the EEA Financial Mechanism and the Norwegian Financial Mechanism. This project is also co-financed from the state budget of the Slovak Republic.

REFERENCES

Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, E.N. (2002). The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1), 3–20.

Kvasnica, M. and Fikar, M. (2010). Performance-lossless complexity reduction in explicit MPC. In *Proceedings of the 49th IEEE Conference on Decision and Control*, 5270–5275. Atlanta, USA.

Kvasnica, M., Grieder, P., and Baotić, M. (2004). Multi-Parametric Toolbox (MPT). Available from http://control.ee.ethz.ch/~mpt/.

Kvasnica, M., Löfberg, J., and Fikar, M. (2011). Stabilizing polynomial approximation of explicit MPC. *Automatica*. (accepted).

Maciejowski, J.M. (2002). *Predictive Control with Constraints*. Prentice Hall.

Siemens (2008). *S7-200 Programable Controller System Manual*.

Siemens (2010). *Betriebsanleitung Nr.:C98130-A7560-A2-6419*.

Svetíková, M., Annus, J., Čirka, Ľ., and Fikar, M. (2003). Real time control of a laboratory fan heater using dspace tools. In *Proceedings of the 14th International Conference Process Control '03*. Štrbské Pleso, High Tatras (Slovakia).