Proceedings of the 6th WSEAS Int. Conf. on NEURAL NETWORKS, Lisbon, Portugal, June 16-18, 2005 (pp1-4)

Generalized Predictive Control with Adaptive Model Based on Neural Networks

PETR PIVOŇKA, PETR NEPEVNÝ
Department of Control and Instrumentation
Faculty of Electrical Engineering and Communication, Brno University of Technology
Kolejní 4, 61200 Brno
CZECH REPUBLIC
http://www.uamt.feec.vutbr.cz

Abstract: - Generalized Predictive Control (GPC) is well known control algorithm. If we put together predictive strategy of GPC and Neural Network model, which is adaptive, then we obtain new controller with many advantages. Neural model is able to observe system changes and adapt itself, therefore regulator based on this model is adaptive. Algorithm was implemented in MATLAB-Simulink with aspect of future implementation to Programmable Logic Controller (PLC) B&R. It was tested on mathematical and physical models in soft-real-time realization. Predictive controller in comparison with classical PSD controller and it's advantages and disadvantages are shown.

Key-Words: - GPC, MPC, predictive, control, adaptive model, Neural Networks

1 Introduction

The beginning of predictive regulators was in late 60s, when we can find terms as optimal system, optimal control. In 70s, after the first optimal controllers like LQR (Linear Quadratic Regulator), the branch of predictive controllers has been separated. Since that a lot of predictive algorithms as MPHC (Model Predictive Heuristic Control), DMC (Dynamic Matrix Control), GPC (Generalized Predictive Control) were published. All predictive methods have the same base, but difference is in used model of system and disturbance and used cost function.

The core of predictive regulator is plant model, which is use to predict a future value of system output. If we know predicted output, we can compute future error (if we know future reference trajectory) and this knowledge is used to compute optimal future action. Control action is optimal in accordance to the cost function, which is used. A number of predicted steps is known as prediction horizon. Prediction horizon is shifted in the time every sampling period (receding horizon).

2 Generalized Predictive Control

GPC is one of the most favourite predictive control method. This method is popular not only in industry, but also at universities. It was first published in 1987. The authors wanted to find one universal method to control different systems. Generalized predictive control is applicable to the systems with non-minimal phase, unstable systems in open loop, systems with unknown or varying dead time and systems with unknown order.

2.1 Unconstrained GPC

We assume Auto Regressive Moving Average (ARMA) model, whose output depends on past systems inputs and outputs. MPC algorithm computes future action increments vector by minimize the cost function *J*.

$$J = E\left\{ \sum_{j=1+d}^{P+d} \left((\hat{y}(t+j\,|\,t) - w(t+j\,|\,t) \right)^2 + \lambda \sum_{j=0}^{M-1} \left(\Delta u(t+j\,|\,t) \right)^2 \right\} \quad (1)$$

where $\hat{y}(t+j\,|\,t)$ is predicted output of system in *j*-th prediction step in discrete time *t*, $w(t+j\,|\,t)$ is reference trajectory in coincidence points, $\Delta u(t+j\,|\,t)$ is *j*-th increment of control action, *P* is prediction horizon, *M* is control horizon, λ is cost constant and *d* is dead time. Variable *t* expresses discrete time – the step of control algorithm. Requirement $M \le P$ is always hold valid.

The aim of predictive control algorithm is to compute future control actions so that the systems output tracks the reference trajectory with minimal error. The cost function also contains a cost of control action increment (it limits actuator damage).
Criterion (1) can be rewritten to matrix form:

$$\mathbf{J} = \frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{H}\mathbf{u} + \mathbf{b}^{\mathrm{T}}\mathbf{u} + \mathbf{f_0} \quad (2)$$

where

$$\mathbf{H} = 2\left(\mathbf{G}^{\mathrm{T}}\mathbf{G} + \lambda\mathbf{I}\right)$$

$$\mathbf{b}^{\mathrm{T}} = 2\left(\mathbf{f} - \mathbf{w}\right)^{\mathrm{T}}\mathbf{G}$$

$$\mathbf{f_0} = \left(\mathbf{f} - \mathbf{w}\right)^{\mathrm{T}}\left(\mathbf{f} - \mathbf{w}\right)$$

u …vector of action increments

$$\mathbf{u} = \begin{pmatrix} \Delta u(t) & \Delta u(t+1) & \dots & \Delta u(t+P-1) \end{pmatrix}$$

w…vector of reference trajectory in coincidence points

$$\mathbf{w} = \begin{pmatrix} w(t+1) & w(t+2) & \dots & w(t+P) \end{pmatrix}$$

f …free response of model

$$\mathbf{f} = \begin{pmatrix} f(t+1) & f(t+2) & \dots & f(t+P) \end{pmatrix}$$

G … matrix of dynamics

$$\mathbf{G} = \begin{bmatrix} g_0 & 0 & 0 & \cdots & 0 \\ g_1 & g_0 & 0 & \cdots & 0 \\ g_2 & g_1 & g_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{M-1} & g_{M-2} & g_{M-3} & \cdots & g_0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{P-1} & g_{P-2} & g_{P-3} & \cdots & g_{P-M} \end{bmatrix}$$

where element g_j is j-th coefficient of model's step response.

Minimum of the cost function (2) is obtained by making the gradient of **J** equal to zero. The result is equation (3) for computation the future control action increments vector.

$$\mathbf{u} = -\mathbf{H}^{-1}\mathbf{b} = \left(\mathbf{G}^{\mathrm{T}}\mathbf{G} + \lambda\mathbf{I}\right)^{-1}\mathbf{G}^{\mathrm{T}}(\mathbf{w}-\mathbf{f}) = \mathbf{K}(\mathbf{w}-\mathbf{f}) \quad (3)$$

Only the first element of vector **u** is used for control. In next sampling period, the new control sequence is computed. Basic control loop with GPC is shown in Fig. 1.

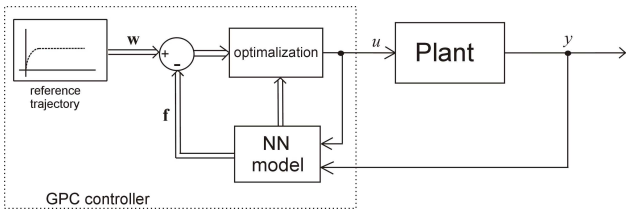

Fig 1:.Control loop with predictive controller (GPC)

2.2 Constrains implementation

Real process output and input are constrained. We can compute unconstrained GPC and limit only algorithm output $\Delta u(t)$ and $u(t)$. It is possible, but it is not optimal solution of constrain GPC.

It is impossible to found solution of GPC with constrains analytically by equation (3), but it should be solved numerically every sampling period. This problem can be solved by a quadratic programming. We have quadratic cost function (2), where f_0 is constant vector. The constants don't have an effect for quadratic programming result, then we can use cost function in form (4). Constrains are defined by equation (5).

$$\mathbf{J} = \frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{H}\mathbf{u} + \mathbf{b}^{\mathrm{T}}\mathbf{u} \quad (4)$$

$$\mathbf{R}\mathbf{u} \le \mathbf{c} \quad (5)$$

Equation (5) can be rewritten to the form:

$$\begin{bmatrix} \mathbf{R_{du}} \\ -\mathbf{R_{du}} \\ \mathbf{R_{du}} \\ -\mathbf{R_{du}} \end{bmatrix} u \le \begin{bmatrix} \mathbf{c_{duMax}} \\ \mathbf{c_{duMin}} \\ \mathbf{c_{uMax}} \\ \mathbf{c_{uMin}} \end{bmatrix} \quad (6)$$

where constrain of control action increment $\Delta u_{\min} \le \Delta u(t+j \mid t) \le \Delta u_{\max}$ is realized by:

$$\mathbf{R_{du}} = \begin{bmatrix} 1 & 0 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ 0 & 0 & 1 & \\ \vdots & \vdots & & \ddots \end{bmatrix} \text{ and } \mathbf{c_{duMax}} = \begin{bmatrix} \Delta u_{\max} \\ \Delta u_{\max} \\ \Delta u_{\max} \\ \cdots \end{bmatrix} \text{ and } \mathbf{c_{duMin}} = \begin{bmatrix} \Delta u_{\min} \\ \Delta u_{\min} \\ \Delta u_{\min} \\ \cdots \end{bmatrix}$$

and constrain of control action $u_{\min} \le u(t+j \mid t) \le u_{\max}$ is realized by:

$$\mathbf{R_u} = \begin{bmatrix} 1 & 0 & 0 & \cdots \\ 1 & 1 & 0 & \cdots \\ 1 & 1 & 1 & \\ \vdots & \vdots & & \ddots \end{bmatrix} \text{ and } \mathbf{c_{uMax}} = \begin{bmatrix} u_{\max} \\ u_{\max} \\ u_{\max} \\ \cdots \end{bmatrix} \text{ and } \mathbf{c_{uMin}} = \begin{bmatrix} u_{\min} \\ u_{\min} \\ u_{\min} \\ \cdots \end{bmatrix}$$

Size of matrices $\mathbf{R_{du}}$ and $\mathbf{R_u}$ is $P \times P$ and length of all vectors **c** is P.

Problem of quadratic programming solves MATLAB's function QUADPROG, which can be used in form **u** = QUADPROG (**H**, **b**, **R**, **c**). The result is vector **u** (future control action increments). In current time, we used only the first element of vector **u**.

2.3 Model Based on Neural Network

Neural Network (NN) (feed-forward perceptron) is used for realizing the third order ARMA model. Inputs of Neural networks are process input and output. Neural network dynamic is represented by step delays z^{-1} (Fig. 2).

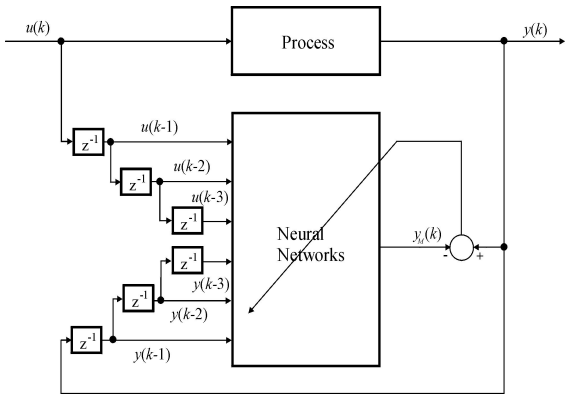

Fig 2:.Adaptive model based on Neural Network

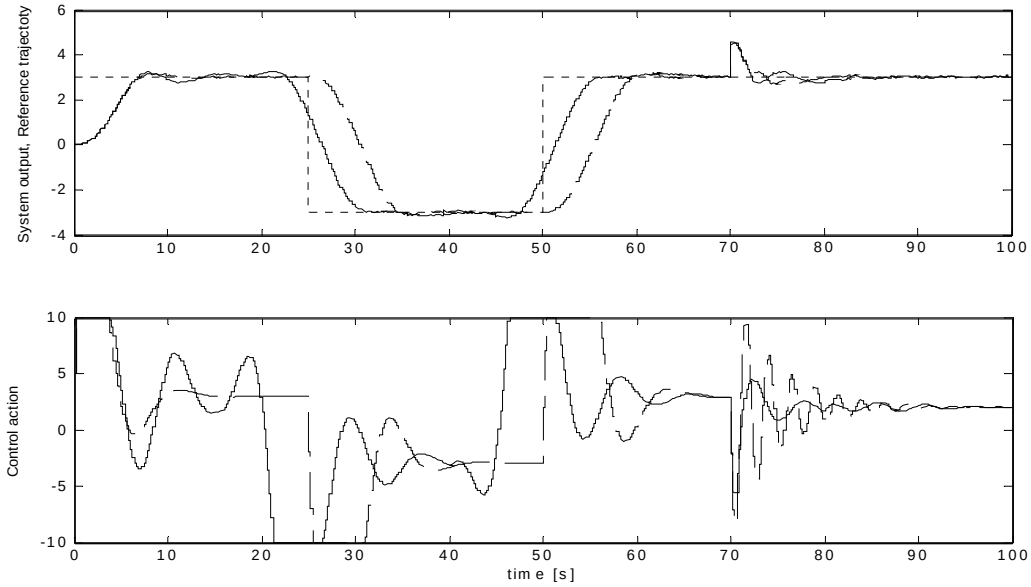

Fig. 3: Comparison of unconstrained GPCBNN with prediction (solid line) and PSD controller (dashed line)

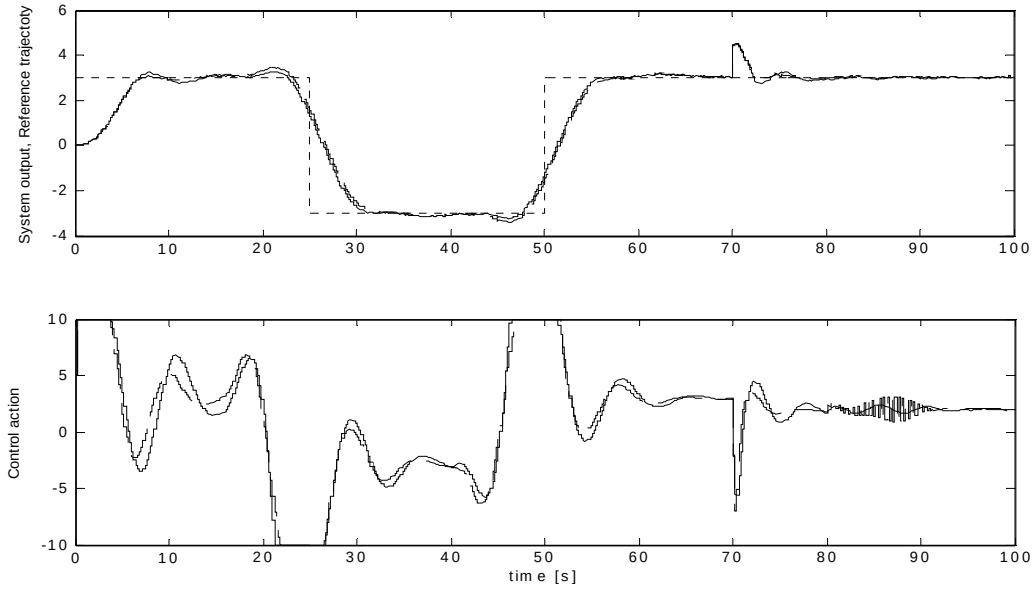

Fig. 4: Comparison of unconstrained (solid line) and constrained (dashed line) GPCBNN with prediction

	constrain GPCBNN without prediction		constrain GPCBNN with prediction		unconstrain GPCBNN without prediction		unconstrain GPCBNN with prediction		PSD		average
	abs.val.	rel.val.	abs.val.	rel.val.	abs.val.	rel.val.	abs.val.	rel.val.	abs.val.	rel.val.	
c_D	350	*0,46*	**298**	*0,39*	1452	*1,92*	1248	*1,65*	425	*0,56*	755,3
c_E	**10830**	*0,95*	11350	*0,99*	11600	*1,01*	11920	*1,04*	11460	*1,00*	11432,8
c_Q	1708	*1,40*	525	*0,43*	1701	*1,39*	**479**	*0,39*	1706	*1,39*	1224,4
Σ	-	2,81	-	**1,82**	-	4,33	-	3,09	-	2,96	-

Table 1: Comparing of various GPCs and PSD controller

For learning the Back-Propagation method is used. There is used batch learning procedure with fixed-sized buffer of training patterns. Patterns in the training set are choose by filtering, which means storing pattern every *n*-th sampling time. Elements in buffer are shifted in time. Using of time-filter is suitable, if we have short sampling period, because it makes the training set smooth and more reliable. It is better to use off-line measured data for NN weights initialization. The synaptic weights *h* are recomputed in each sampling time by equation:

$$h_{ji}^{l}(n+1) = h_{ji}^{l}(n) + \alpha\left[h_{ji}^{l}(n) - h_{ji}^{l}(n-1)\right] + \\ + \eta\delta_j(n)y_i^{(l-1)}(n) \tag{7}$$

where *n* iteration step, α is the momentum constant, η is the learning-rate parameter, δ is local gradient and finally, $y_i^{(l-1)}$ is the function signal of neuron *i* in the previous layer (*l*-1) at iteration *n*. Parameters α and η represent algorithm's adjusting parameters with influence on the rate of convergence. Parameters should be limited to interval (0,1).

3 Application

The base of adaptive predictive controller is neural network model, which substitute third order ARMA model. We can call it Generalized Predictive Control Based on Neural Networks (GPCBNN). During control process the training set is updated and neural model is learned by Back-Propagation algorithm. Then the neural network model is able to tracks the plants changes – GPCBNN is adaptive.

The coefficients of the model are used for calculation of future action vector. At first, the unconstrained GPCBNN action is computed using equation (3). If we want to compute optimal constrained GPCBNN, we can use unconstrained GPCBNN control vector as a starting point for quadratic programming.

Simulations were realized with mathematical and physical models of a plant**.** Control and identification algorithms were implemented into S-function in MATLAB and control loop was created in MALAB-Simulink. The physical model control was realized by connection PC and physical model via PLC B&R. For connection the Ethernet was used. Sampling period for communication and control was 0.2 s.

For relevant comparison of GPCBNN and classical PSD controller, we find PSD parameters by the process of minimize the cost function, which is similar to (1). This problem can be solved using MATLAB's function FMINS. Simulations results are shown in Fig. 3, where is the comparison of unconstrain GPC with prediction and PSD controller. In time 70 s was change of plant.

Comparison of unconstrained GPCBNN (constrained after optimization) and constrained GPC (optimization with constrains) is in Fig. 4. It is shown, that there is only little difference.

For evaluation of results the criterions were used:

$$c_D = \sum_t \left(\Delta u(t)\right)^2 \tag{8}$$

$$c_E = \sum_t \left(u(t)\right)^2 \tag{9}$$

$$c_Q = \sum_t \left(y(t) - w(t)\right)^2 \tag{10}$$

where

c_D…criterion of actuator Damage

c_E…criterion of Energy

c_Q…criterion of Quadratic error

Table 1 presents comparison of various GPCBNN and PSD controller. If we know future desired value (reference trajectory), we call it GPCBNN with prediction. If we don't have an information about future desired value, we have only GPCBNN without prediction. Absolute value (abs. val.) means real computed value of criterion and relative value (rel. val.) means absolute value divided by average absolute value. Main criterion for controller evaluation is sum of relative values of all criterions.

4 Conclusion

In this paper the application of constrained predictive controller with neural network model is shown. Table 1 demonstrates, that (in accordance to criterion) the constrain GPCBNN with prediction is the best. If the classic PSD controller is set by similar criterion as GPC, then the results are only little different. Main advantages of predictive controller are three. It reacts faster (with small overshoots) to changes of desired value then PSD controller (if it has a knowledge about future reference trajectory). The constrains can be implemented in control algorithm and it is adaptable to plant changes. Using of adaptive model based on NN is useful if we have short sampling period. This new predictive algorithm GPCBNN is going to be full implemented in PLC for real-time control application.

Acknowledgements:

The paper has been prepared as a part of the solution of GAČR project No. 102/03/H116 and with the partially support of the research plan FRVŠ: F1/2902/2005.

References:

[1] Camacho, E. F., Bordos, C., *Model Predictive Control*, Springer, 1999

[2] Cichocki, A., Unbehauen R. *Neural networks for optimization and signal processing*, Wiley, 1994