Generalized Predictive Control with a Non-linear Autoregressive Model

ING. HYNEK VYCHODIL, ING. MICHAL SCHMIDT,
ING. PETR NEPEVNÝ, PROF. ING. PETR PIVOŇKA, CSC.
Department of Control and Instrumentation
FEEC VUT Brno
Kolejní 4, 61200 Brno
CZECH REPUBLIC

Abstract: – This paper presents a solution to computation of predictive control using non-linear auto-regressive models. For the non-linear model a neural network is used as a perspective tool for modelling of dynamic systems. However, the described approach is applicable to any type of auto-regressive model. The model is not linearized in the operating point, but in each control optimization step the model's derivative is computed (linearization) for all points in the prediction horizon. The method can be used in real-time control. This is verified by porting the algorithm directly to the PLC.

Key–Words: – Neural network, Modelling, Non-linear, Predictive control

1 Control Algorithm

1.1 Generalized predictive control of a non-linear system

Let's have a discrete causal non-linear dynamic system defined by the function:

$$y_k = f(k, \mathbf{u}, \mathbf{x}_0) \tag{1}$$

where:
y_k .... system output in step k,
f .... non-linear function of the system,
$\mathbf{u}$ .... control action vector $(u_{k-1}, u_{k-2}, \dots u_0)^{\mathrm{T}}$ and
$\mathbf{x}_0$ ... initial state.

and its model

$$\hat{y}_k = \hat{f}(k, \mathbf{u}, \mathbf{x}_0) \tag{2}$$

where:
$\hat{y}_k$ .... estimated system output in step k and
$\hat{f}$ .... non-linear function of the model.

Let's have a cost function:

$$E = \frac{1}{2}\sum_{i=1}^{n}\left[(\hat{y}_i - w_i)^2 + \alpha(u_{i-1} - u_{i-2})^2\right] \tag{3}$$

where:
E .... cost function,
n .... prediction horizon,
$\hat{y}_i$ .... system output in step i,
w_i .... set value in step i,
α .... cost of action,
u_i .... control action in step i.

The cost function can be represented in a vector form as

$$E = \frac{1}{2}\left[\|\mathbf{e}\|^2 + \alpha\,\|\mathbf{h}\|^2\right] \tag{4}$$

where:
$\mathbf{e}$ .... vector of differences between the predicted system output and the desired value $(\hat{y}_n - w_n, \hat{y}_{n-1} - w_{n-1}, \dots, \hat{y}_1 - w_1)^{\mathrm{T}}$ and
$\mathbf{h}$ .... vector of action increments $(u_{n-1} - u_{n-2}, u_{n-2} - u_{n-3}, \dots u_0 - u_{-1})^{\mathrm{T}}$.

For both vectors $\mathbf{e}$ and $\mathbf{h}$ we can introduce their variation by the control vector $\mathbf{u}$ using variation matrices $\mathbf{G}$ resp. $\mathbf{H}$ and estimate their values in the iteration process $\mathbf{u}_{j+1} = \mathbf{u}_j + \Delta\mathbf{u}$ with a first-order Taylor polynomial:

$$\hat{\mathbf{e}}_{j+1} = \mathbf{e}_j + \mathbf{G}\Delta\mathbf{u} \tag{5}$$
$$\text{and} \quad \hat{\mathbf{h}}_{j+1} = \mathbf{h}_j + \mathbf{H}\Delta\mathbf{u} \tag{6}$$

By substitution of (5) and (6) into (4) we get an estimation of the cost function:

$$\hat{E}_{j+1} = \frac{1}{2}\Big[\, \|\mathbf{e}_j + \mathbf{G}\Delta\mathbf{u}\|^2 + \alpha\, \|\mathbf{h}_j + \mathbf{H}\Delta\mathbf{u}\|^2 \Big] \quad (7)$$

By minimizing this estimation we get:

$$\begin{aligned}\frac{\partial \hat{E}}{\partial \Delta\mathbf{u}} &= \mathbf{G}^{\mathrm{T}}\left(\mathbf{e}_j + \mathbf{G}\Delta\mathbf{u}\right) + \\ &\quad + \alpha\mathbf{H}^{\mathrm{T}}\left(\mathbf{h}_j + \mathbf{H}\Delta\mathbf{u}\right) = \mathbf{0} \end{aligned} \quad (8)$$

The corresponding new control is then:

$$\begin{aligned}\mathbf{u}_{j+1} = \;& \mathbf{u}_j - \left(\mathbf{G}^{\mathrm{T}}\mathbf{G} + \alpha\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1} \cdot \\ & \cdot \left(\mathbf{G}^{\mathrm{T}}\mathbf{e}_j + \alpha\mathbf{H}^{\mathrm{T}}\mathbf{h}_j\right)\end{aligned} \quad (9)$$

The iteration process described by the equation (9) does not necessarily converge for all non-linear functions f from the equation (1) nor for all initial conditions $\mathbf{u}_0$. For a class of non-linear continuous smooth functions and initial vectors $\mathbf{u}_0$ the convergence can be expected. The probability of convergence for a given pair of function $f(k, \mathbf{u}, \mathbf{x}_0)$ and an initial vector $\mathbf{u}_0$ can be increased using the Marquardt–Levenberg's modification of the Newton's method for searching extremes of a multidimensional function.

The result is the well-known algorithm, which is also used for learning of neural networks:

$$\begin{aligned}\mathbf{u}_{j+1} = \mathbf{u}_j - \left(\mathbf{G}^{\mathrm{T}}\mathbf{G} + \alpha\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda\mathbf{I}\right)^{-1} \cdot \\ \cdot \left(\mathbf{G}^{\mathrm{T}}\mathbf{e}_j + \alpha\mathbf{H}^{\mathrm{T}}\mathbf{h}_j\right)\end{aligned} \quad (10)$$

where:
λ .... non-negative scalar which allows smooth transition between Newton's method and gradient descent method.

For $\lambda \to 0$ we get equation (9).
For $\lambda \to \infty$ we get

$$\mathbf{u}_{j+1} = \mathbf{u}_j - \left(\mathbf{G}^{\mathrm{T}}\mathbf{e}_j + \alpha\mathbf{H}^{\mathrm{T}}\mathbf{h}_j\right)/\lambda \quad (11)$$

which is a gradient descent method. The usually chosen starting value of λ is 0.1. In each step a new control vector is computed, the cost function E is evaluated and compared with the value in the previous step. If it is less than in the previous step, then λ is decreased and the next iteration step starts. Otherwise λ is increased, the control vector is assigned its previous value and the current step is repeated with the new value of λ.

1.2 The variation matrix of a non-linear autoregressive model

For the equation (10) the knowledge of variance matrices $\mathbf{H}$ and $\mathbf{G}$ is necessary. The matrix $\mathbf{H}$ is:

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & & \cdots & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & & & 0 \\ \vdots & & \ddots & \ddots & & \vdots \\ 0 & & & -1 & 1 & 0 \\ 0 & 0 & \cdots & 0 & -1 & 1 \end{pmatrix} \quad (12)$$

The matrix $\mathbf{G}$ has a form:

$$\mathbf{G} = \begin{pmatrix} \left.\frac{\partial \hat{f}}{\partial u_0}\right|_{k=1} & \left.\frac{\partial \hat{f}}{\partial u_1}\right|_{k=1} & \cdots & \left.\frac{\partial \hat{f}}{\partial u_{n-1}}\right|_{k=1} \\ \left.\frac{\partial \hat{f}}{\partial u_0}\right|_{k=2} & \left.\frac{\partial \hat{f}}{\partial u_1}\right|_{k=2} & \cdots & \left.\frac{\partial \hat{f}}{\partial u_{n-1}}\right|_{k=2} \\ \vdots & \vdots & \ddots & \vdots \\ \left.\frac{\partial \hat{f}}{\partial u_0}\right|_{k=n} & \left.\frac{\partial \hat{f}}{\partial u_1}\right|_{k=n} & \cdots & \left.\frac{\partial \hat{f}}{\partial u_{n-1}}\right|_{k=n} \end{pmatrix} \quad (13)$$

To simplify we will write $\frac{\partial \hat{y}_p}{\partial u_r}$ instead of $\left.\frac{\partial \hat{f}}{\partial u_r}\right|_{k=p}$. For causal systems the matrix $\mathbf{G}$ is lower triangular:

$$\mathbf{G} = \begin{pmatrix} \frac{\partial \hat{y}_1}{\partial u_0} & 0 & \cdots & 0 \\ \frac{\partial \hat{y}_2}{\partial u_0} & \frac{\partial \hat{y}_2}{\partial u_1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}_n}{\partial u_0} & \frac{\partial \hat{y}_n}{\partial u_1} & \cdots & \frac{\partial \hat{y}_n}{\partial u_{n-1}} \end{pmatrix} \quad (14)$$

Let's define a time-invariant non-linear autoregressive model of the system (1) in the form:

$$\begin{aligned}\hat{y}_{k+1} = F(&\hat{y}_k, \hat{y}_{k-1}, \ldots \hat{y}_{k-m}, \\ & u_k, u_{k-1}, \ldots, u_{k-l})\end{aligned} \quad (15)$$

where:
F .... non-linear function (can be realized by a neural network) and
l, m .. define the order of the model. Often $l = m$.

If we denote $k+1$ as the operating point and partial derivations by one of the parameters x of the function F in this point is $\left.\frac{\partial F}{\partial x}\right|_{k+1}$, we'll denote this derivation as $\frac{\partial F_{k+1}}{\partial x}$. Then we can calculate each element of the matrix $\mathbf{G}$ as:

$$\frac{\partial \hat{y}_p}{\partial u_r} = \frac{\partial F_p}{\partial u_{k-p+r+1}} + \sum_{i=0}^{m} \frac{\partial F_p}{\partial \hat{y}_{k-i}} \cdot \frac{\partial \hat{y}_{p-i-1}}{\partial u_r} \quad (16)$$

$$
\begin{array}{c|cccccccccc}
\text{row}\backslash\text{column} & & p-l-1 & p-l & & p-2 & p-1 & p & p+1 & \\
\hline
 & & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \\
 & & + & + & & + & + & + & + & \\
 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots \\
 & & \cdot & \cdot & & \cdot & \cdot & \cdot & \cdot & \\
p-m-2 & \cdots & \frac{\partial \hat{y}_{p-m-2}}{\partial u_{p-l-2}} & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots \\
 & & + & + & & + & + & + & + & \\
 & \cdots & \frac{\partial F_p}{\partial \hat{y}_{k-m}} & \frac{\partial F_p}{\partial \hat{y}_{k-m}} & \cdots & \frac{\partial F_p}{\partial \hat{y}_{k-m}} & \frac{\partial F_p}{\partial \hat{y}_{k-m}} & \frac{\partial F_p}{\partial \hat{y}_{k-m}} & \frac{\partial F_p}{\partial \hat{y}_{k-m}} & \cdots \\
 & & \cdot & \cdot & & \cdot & \cdot & \cdot & \cdot & \\
p-m-1 & \cdots & \frac{\partial \hat{y}_{p-m-1}}{\partial u_{p-l-2}} & \frac{\partial \hat{y}_{p-m-1}}{\partial u_{p-l-1}} & \cdots & 0 & 0 & 0 & 0 & \cdots \\
 & & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \\
 & & + & + & & + & + & + & + & \\
 & \cdots & \frac{\partial F_p}{\partial \hat{y}_{k-1}} & \frac{\partial F_p}{\partial \hat{y}_{k-1}} & \cdots & \frac{\partial F_p}{\partial \hat{y}_{k-1}} & \frac{\partial F_p}{\partial \hat{y}_{k-1}} & \frac{\partial F_p}{\partial \hat{y}_{k-1}} & \frac{\partial F_p}{\partial \hat{y}_{k-1}} & \cdots \\
 & & \cdot & \cdot & & \cdot & \cdot & \cdot & \cdot & \\
p-2 & \cdots & \frac{\partial \hat{y}_{p-2}}{\partial u_{p-l-2}} & \frac{\partial \hat{y}_{p-2}}{\partial u_{p-l-1}} & \cdots & \frac{\partial \hat{y}_{p-2}}{\partial u_{p-3}} & 0 & 0 & 0 & \cdots \\
 & & + & + & & + & + & + & + & \\
 & \cdots & \frac{\partial F_p}{\partial \hat{y}_{k}} & \frac{\partial F_p}{\partial \hat{y}_{k}} & \cdots & \frac{\partial F_p}{\partial \hat{y}_{k}} & \frac{\partial F_p}{\partial \hat{y}_{k}} & \frac{\partial F_p}{\partial \hat{y}_{k}} & \frac{\partial F_p}{\partial \hat{y}_{k}} & \cdots \\
 & & \cdot & \cdot & & \cdot & \cdot & \cdot & \cdot & \\
p-1 & \cdots & \frac{\partial \hat{y}_{p-1}}{\partial u_{p-l-2}} & \frac{\partial \hat{y}_{p-1}}{\partial u_{p-l-1}} & \cdots & \frac{\partial \hat{y}_{p-1}}{\partial u_{p-3}} & \frac{\partial \hat{y}_{p-1}}{\partial u_{p-2}} & 0 & 0 & \cdots \\
 & & + & + & & + & + & + & + & \\
 & \cdots & 0 & \frac{\partial F_p}{\partial u_{k-l}} & \cdots & \frac{\partial F_p}{\partial u_{k-2}} & \frac{\partial F_p}{\partial u_{k-1}} & \frac{\partial F_p}{\partial u_{k}} & 0 & \cdots \\
 & & \Downarrow & \Downarrow & & \Downarrow & \Downarrow & \Downarrow & \Downarrow & \\
p & \cdots & \frac{\partial \hat{y}_{p}}{\partial u_{p-l-2}} & \frac{\partial \hat{y}_{p}}{\partial u_{p-l-1}} & \cdots & \frac{\partial \hat{y}_{p}}{\partial u_{p-3}} & \frac{\partial \hat{y}_{p}}{\partial u_{p-2}} & \frac{\partial \hat{y}_{p}}{\partial u_{p-1}} & 0 & \cdots
\end{array}
$$

Fig. 1: Calculation of the p-th row of matrix $\mathbf{G}$

It is clear that the matrix $\mathbf{G}$ can be filled during the prediction. In p-th step of prediction the p-th row is calculated using m previous rows multiplied by the corresponding coefficients of the sensitivity derivation of the non-linear model (neural network). An illustration of the calculation is shown in Fig. 1.

1.3 Model Error Correction

An inaccuracy of the model can cause a steady error in the system output. To prevent future errors the knowledge of past differences between the model and the system can be used. Average difference is calculated over several past time steps and this value is assumed to be constant in the future. This average value is added to the model output over the whole prediction horizon. This approach is described in [8].

1.4 The non-linear model

For modelling of the non-linear system (1) a non-linear auto-regressive moving-average (NARMA) model (15) is used, where the function F is realized by a neural network. The neural network is a feed-forward multi-layered perceptron with one hidden layer with two non-linear neurons. For learning, the Marquardt-Levenberg's method is used. The size of the training set is fixed. The set is updated on-line with new samples. The selection of old samples to replace is controlled by the samples' criterion of "credibility". This criterion penalizes samples which are too similar to each other, therefore it ensures sufficient diversity of the training set [7].

2 Implementation

2.1 Development of the control algorithm

The control algorithm is developed in three phases as shown in Fig. 2. First, the algorithm is tested with a simulation tool e. g. MATLAB/Simulink. Second, the algorithm is connected to the real world. At this point it is still running on a PC and communicates in real-time with a PLC which connects it to the process. We used a B&R PLC which communicated with MATLAB through Process Vizualization Interface (PVI) over Ethernet. Finally, the algorithm is ported directly to the PLC. The B&R PLC is programmable in ANSI C, therefore the porting fairly straightforward. We developed a portable C library *sfmat* which supports matrix operations, neural networks, system modelling and predictive control. The library works in the same way on the PC and the B&R PLC.

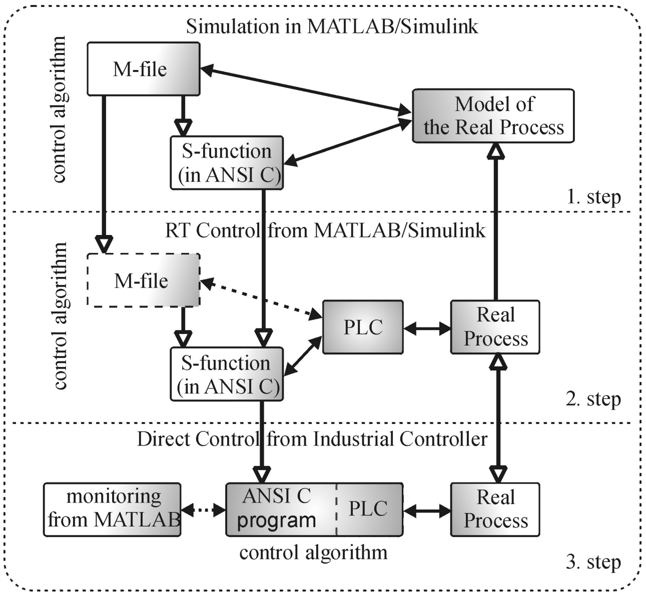

Fig. 2: The scheme of three steps of direct implementation from MATLAB/Simulink into PLC.

2.2 Simulation results

We compared the neural predictive controller with a classical well known and used PID (PI-D variant) controller. For the comparison, analog physical model was used. It is a 3-rd order system with non-linearities, offset, and noise. Simulation results are shown in Fig. 3. The predictive controller was intentionally set so that it didn't take into account the information about future desired value. The response to the step in desired value of predictive controller is faster and with smaller overshoot than with the PID. Response to the disturbance in the system is also faster if we use predictive controller. When we used a predictive controller with knowledge of future desired value, it reacted to the change of reference trajectory before the change was done. This can be seen on Fig. 4.

3 Conclusion

This paper presents a computation method for non-linear predictive control. The linearization in each pre-

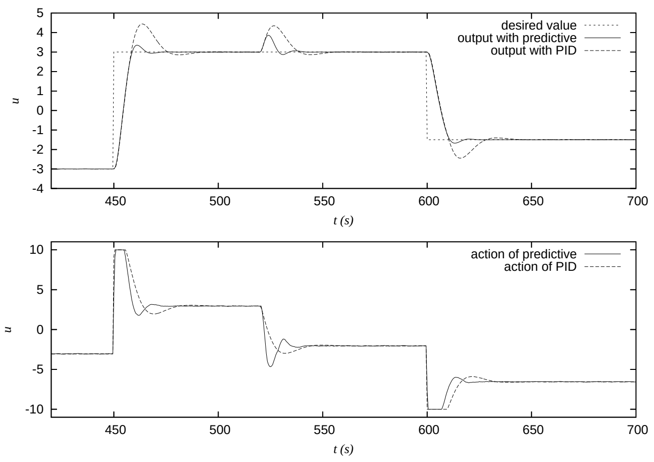

Fig. 3: Comparison of control process with the predictive controller and PID controller. The physical model's transfer function is $F(s) \approx \frac{1}{(s+1)} \frac{1}{(10s+1)} \frac{1}{(s+1)}$. A constant disturbance of size 3 is entering into the middle integrator at time $t = 520$s.

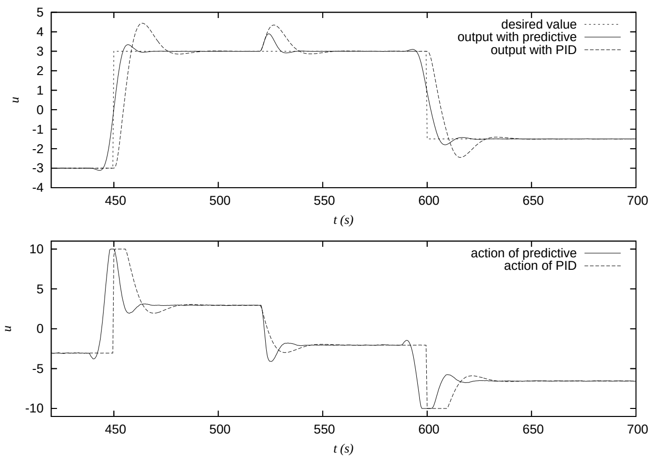

Fig. 4: Comparison of the control process with the predictive controller and PID controller. The controlled system is the same as in Fig. 3. The predictive controller is aware of future reference trajectory.

diction step brings the advantage of more optimized control of non-linear systems as opposed to linearization of the model in only one operating point. The computation is still effective enough to implement the control algorithm in real-time control.

The feasibility of the control agorithm for real-time control was verified when the algorithm was connected through PVI to the physical model and later ported directly into the PLC. Testing showed that the algorithm compares favourably to the PID controller.

The full power of the control algorithm is fully unleashed, when the reference trajectory is known in advance. In such cases the predictive controller can react to the change ahead of time.

Acknowledgement: The paper has been prepared with the support of the research plan PhD grant Talent 102/03/H116.

References

[1] Omatu, S., Khalid, M., Yusof, R., *Neuro-Control and its Application*, Springer–Verlang, London Limited, 1996

[2] Alexander, I., Morton, H., *An Introduction to Neural Computing*, Chapman and Hall, 1990

[3] Hassoun, M. H., *Fundamentals of Artificial Neural Networks*, The MIT Press, 1995

[4] Hunt, K. J., et al., Neural Networks for Control Systems – A Survey, *Automatica*, Vol.28, No.6, 1992, pp. 1083-1112.

[5] Najvárek, J., *Neural networks in predictive control. Dissertation work* (in Czech), ÚAMT, FEI VUT Brno, 1998

[6] Pivoňka, P., Neural controllers (in Czech), *Automatizace*, Vol.38, No.2, 1995, pp. 39-43.

[7] Vychodil, H., Pivoňka, P., Krupanský, P., The Choice of Patterns in Training Set for Neural On-line Identification, *In Advances in Scientific Computing, Computational Intelligence and Applications*, (Eds. Mastorakis V.) Electrical and Computer Engineering Series -A Series of Reference Books and Textbooks, WSES Press, Greece, 2002, pp. 250-254.

[8] Richalet, J.A., A. Rault, J.D. Testud, J. Papon, Model Predictive Heuristic Control: Applications to Industrial Processes. *Automatica*, 14, 1978, pp. 413-428.