PORTABLE IMPLEMENTATION OF A NEURAL NETWORK BASED SEMI-INVERSE CONTROLLER

Michal SCHMIDT, Doctoral Degree Programme (1)
Dept. of Control and Instrumentation, FEEC, BUT
E-mail: xschmi00@stud.feec.vutbr.cz

Supervised by: Prof. Petr Pivoňka

ABSTRACT

The semi-inverse controller avoids some of the problems of normal inverse controllers. Unlike the inverse controller it is based on a forward model of the plant. The model is then used as a controller by reconfiguring its inputs. For maximum portability of the algorithm between the development environment and the final implementation in PLC an abstract object-oriented interface is used.

1 INTRODUCTION

Ever since neural networks began to be used in controller algorithms, the designs of the algorithms were often based on the ability of a neural network to approximate the inverse dynamics of the controlled plant. Serial connection of this inverse model with the plant then theoretically suffices to respond well to the desired value. Unfortunately the inversion approach has several problems. The inverse model can be hard or even impossible to obtain. Other problems arise from the fact that the inverse controller is a dead-beat controller. This leads to high sensitivity to the precise learning of the model. When working with short sampling periods, it also produces extreme control actions. These problems are highlighted when noise is present. They can be mitigated with filtration, but it worsens the dynamics of the control system. The semi-inverse controller [1] attempts to avoid the problem of learning of the inverse model by basing itself on a forward model instead.

2 THE SEMI-INVERSE CONTROLLER

2.1 THE INVERSE CONTROLLER IN A CLOSED LOOP

Let's suppose the plant is approximated by an ARMA model:

$$F_{\mathrm{M}}(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_m z^{-m}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}} \tag{1}$$

An inverse controller would have a transfer function:

$$F_C(z) = \frac{1 + a_1 z^{-1} + a_2 z^{-2} + ... + a_n z^{-n}}{b_0 + b_1 z^{-1} + b_2 z^{-2} + ... + b_m z^{-m}} \tag{2}$$

The inverse controller is normally used in an open loop, without a real feedback. This is undesirable because it doesn't eliminate disturbance. Therefore when used in a closed control loop, it computes the action as:

$$u_k = \frac{e_k + a_1 e_{k-1} + a_2 e_{k-2} ... + a_n e_{k-n} - b_1 u_{k-1} - b_2 u_{k-2} - ... - b_m u_{k-m}}{b_0} \tag{3}$$

where: k .... discrete time step
u_k ...... control action in step k
e_k ...... control error in step k

The big sensitivity of the controller on the model accuracy comes mainly from the parameter b_0, especially when its value is close to zero.

After the finish of a transient response, the control action will be:

$$u_S = \frac{e_S(1 + a_1 + a_2 + ... + a_n) - u_S(b_1 + b_2 + ... + b_m)}{b_0} \tag{4}$$

It is clear that in the steady state (indicated by the index S) the control error e_S would be non-zero. Let's introduce a modified control error:

$$e_{M,k} = w_k + e_k = 2w_k - y_k \tag{5}$$

where: $e_{M,k}$ .... modified control error in step k
w_k ..... desired value in step k
y_k ...... system output in step k

This is implemented by a filter $F_k(z)$. If the controller operates on the modified control error, it will remain non-zero in the steady state but the real error will be able to reach zero. In order for this to work as expected, the gain of the open loop must be 1.

2.2 MODIFICATION FOR A SEMI-INVERSE CONTROLLER

Let's start from the inverse controller operating on the modified error. In a steady state, its action will be:

$$u_S = \frac{e_{M,S}(1 + a_1 + a_2 + ... + a_n) - u_S(b_1 + b_2 + ... + b_m)}{b_0} \tag{6}$$

In order to simplify the situation, let's assume that the gain of the plant model is $A_M = 1$. We'll generalize this assumption away later. For now let's also assume that the desired value is $w_k = 1$. Under these simplifying conditions we can write:

$$u_S = y_S = e_{M,S} = w_S = 1 \tag{7}$$

Therefore some of the members in the equation (6) can be cancelled, resulting in:

$$1 = \frac{(1 + a_1 + a_2 + ... + a_n) - (b_1 + b_2 + ... + b_m)}{b_0} \tag{8}$$

Thus it holds that:

$$1 = b_0 - a_1 - a_2 - ... - a_n + b_1 + b_2 + ... + b_m \tag{9}$$

At this point we'll modify the inverse controller and rewrite the action of the semi-inverse controller in the steady state as:

$$\begin{aligned} 1 = u_\mathrm{S} = e_{\mathrm{M,S}}(b_0 - a_1 - a_2 - ... - a_n) + u_\mathrm{S}(b_1 + b_2 + ... + b_m) = \\ = 1 \cdot (b_0 - a_1 - a_2 - ... - a_n) + 1 \cdot (b_1 + b_2 + ... + b_m) \end{aligned} \tag{10}$$

The idea behind this modification is that the parameter b_0 is transferred instead of constant 1 in the inverse controller. In the non-steady case the action is computed as:

$$\begin{aligned} u_k = b_0 e_{\mathrm{M},k} - a_1 e_{\mathrm{M},k-1} - a_2 e_{\mathrm{M},k-2} - ... - a_n e_{\mathrm{M},k-n} + \\ + b_1 u_{k-1} + b_2 u_{k-2} + ... + b_m u_{k-m} \end{aligned} \tag{11}$$

The corresponding transfer function of the semi-inverse (SI) controller is:

$$F_{\mathrm{SI}}(z) = \frac{b_0 - a_1 z^{-1} - ... - a_n z^{-n}}{1 - b_1 z^{-1} - ... - b_m z^{-m}} \tag{12}$$

2.3 NORMALIZATION OF GAIN

The total gain of the serial connection of the semi-inverse controller and the plant is:

$$A = A_{\mathrm{SI}} A_\mathrm{M} = \frac{b_0 - a_1 - a_2 ... - a_n}{1 - b_1 - b_2 - ... - b_m} \cdot \frac{b_0 + b_1 + b_2 + ... + b_m}{1 + a_1 + a_2 + ... + a_n} \tag{13}$$

To fulfil the requirement that the gain of the open loop is equal to 1, it's necessary to add a multiplication by $\frac{1}{A}$ to the open loop.

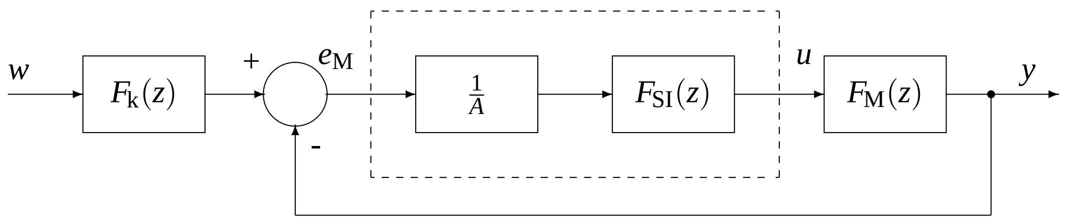

Figure 1: Control loop with a semi-inverse controller

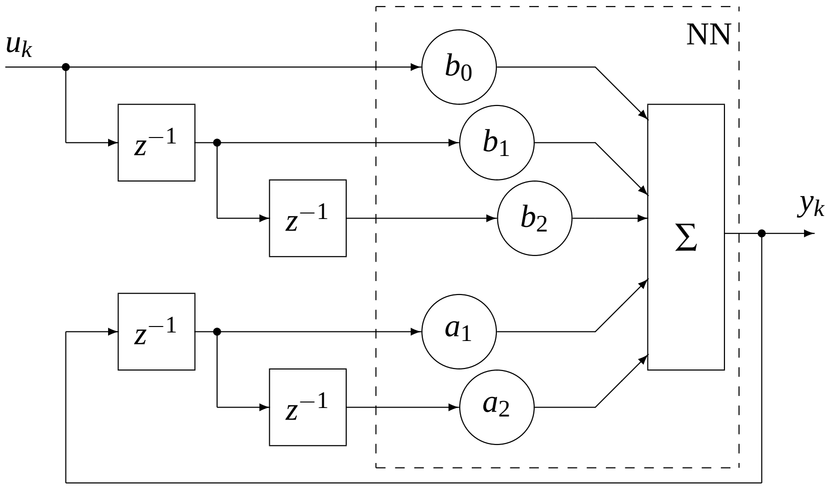

Figure 2: The model can be linear ARMA or non-linear neural network (NN) model

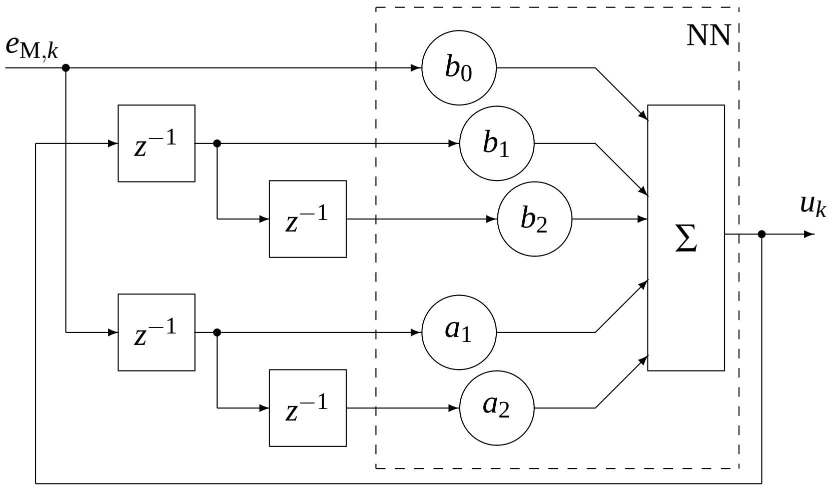

Figure 3: The semi-inverse controller is based on the model, only its inputs are reconfigured

3 IMPLEMENTATION

The algorithm was originally implemented in ANSI C as an S-function in Matlab Simulink. Thanks to the availability of ANSI C development environment in the B&R PLC the final porting was simplified. To make porting completely seamless, an abstract object-oriented interface was developed for control algorithms. It allows to transfer the source code of control algorithms between Matlab S-functions and B&R PLCs. Only the skeleton parts of the interface had to be implemented separately on both platforms. Not having to change the source code of the algorithm decreases the possibilities of introducing new bugs.

4 TESTING THE CONTROLLER

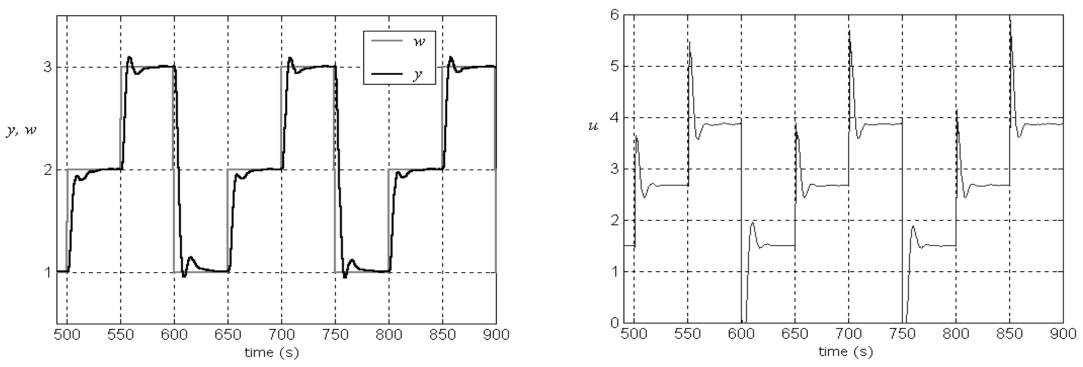

Figure 4: Testing the semi-inverse controller on a physical model with approximate transfer function $F_{\mathrm{M}}(s) = \frac{1}{(5s+1)(s+1)^2}$

5 CONCLUSION

The semi-inverse controller was implemented and tested in simulations and on physical models. It is computationally simple and suitable for real-time control. It is highly adaptable and it works with short sampling periods. The abstract interface for control algorithms proved useful in porting the semi-inverse controller to the PLC. Using this interface, the source code of the algorithm is identical in the Matlab S-function and in the PLC.

Acknowledgement: This paper has been prepared with partial support of the project FRVŠ:F1/2902/2005.

REFERENCES

[1] Krupanský, P.: The Possibilities of Using Neural Networks in Control, (Ph.D. work, in Czech), ÚAMT, FEKT VUT, Brno, 2003.

[2] Švancara, K., Pivoňka, P.: Adaptive Optimal Controller Direct Implemented from MATLAB into PLC B&R. The 13th International Conference DAAAM, Wien, 2002.