Continental Automated Buildings Association

Information Series

IS 2005-42

Neural Networks for Self Tuning of PI- and PID-Controllers

CABA
www.caba.org

Neural Networks for Self Tuning of PI- and PID-Controllers

Reprint Date: December 2005

Neural Networks for Self Tuning of PI- and PID-Controllers

Mohsen Soleimani-Mohseni
Dept. of Building Services Engineering
Chalmers University of Technology
SE-402 96 Göteborg
Sweden
mohsen@chl.chalmers.se

Bertil Thomas
Dept. of Electrical and Computer Engineering.
Chalmers Lindholmen University College.
P.O.Box 8873
SE-402 72 Göteborg, Sweden
bertil@chl.chalmers.se

***Abstract* - In this paper, it is shown how neural networks can be used to estimate parameters of PID-controllers for different classes of dynamic processes. By measuring a number of points at the step-response or the impulse-response of a process and using them as input to a successfully trained neural network, the network can estimate the PID-parameters for the same process according to well-known tuning rules for PID-controllers. The estimation can be made with good accuracy for the three classes of dynamic processes studied in this paper and for the three tuning methods discussed. It is also shown that the result of the neural network training can be much improved by making different forms of pre-processing on the training data set. In the paper, we have used feed-forward neural networks with two layers of hidden neurons and the best networks were trained, using a two-step procedure based on Levenberg-Marquardts method.**

I. INTRODUCTION

In different kinds of industries there are many processes which must be automatically controlled. Two components, which are often used in industrial control applications are PI- and PID-controllers. In order to obtain the desired behaviour of a feedback control system, the system must be well designed and this requires that the PID-parameters are tuned with care. The most accurate methods today for tuning of PID-parameters require the frequency response of the process or a mathematical model from which the frequency response can be obtained. However, for many industrial processes, it is not realistic to put up these models, since it requires too much time and/or engineering skill.

In this paper, a new method to tune the PID-controllers is discussed. The method is based on neural networks and it is shown how such networks can be used to estimate parameters of PID-controllers for different classes of dynamic processes. The step responses of three different classes of processes are illustrated in Fig. 1. By measuring a number of points at the step-response or the impulse-response of a process and using them as input to a successfully trained neural network, as presented in Fig. 2, the network can estimate the PID-parameters for the same process according to different tuning rules for PID-controllers. It is shown that that the PID-parameters can be estimated with good accuracy by the neural network for the classes of dynamic processes studied and for the tuning methods discussed.

II. TYPES OF PROCESSES

The following three classes of processes have been studied in this paper, (1) processes with 3-6 real time constants, (2) processes with two complex poles and one real pole and (3) processes with dead-time and three time constants.

$$G(s) = \frac{1}{\prod_{i=1}^{n}(1+T_i \cdot s)}, \tag{1}$$

where the order n is a number between 3 and 6 and the time constants T_i are between 1and 100.

$$G(s) = \frac{1}{(1+2\cdot k\cdot s+s^2)\cdot(1+T\cdot s)}, \tag{2}$$

where $\begin{cases}0.1 \le k \le 0.5\\ 0.1 \le T \le 1\end{cases}$

$$G(s) = \frac{e^{d\cdot s}}{\prod_{i=1}^{n}(1+T_i \cdot s)}, \tag{3}$$

where the order n is a number between 3 and 6, the time constants T_i are between 1and 100, and the dead-time d is between 1 and 20.

In Fig. 1 three examples of step responses for the three different classes of processes are shown.

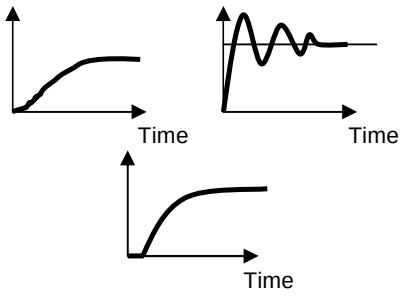

Fig. 1. Step responses for the three different classes of processes.

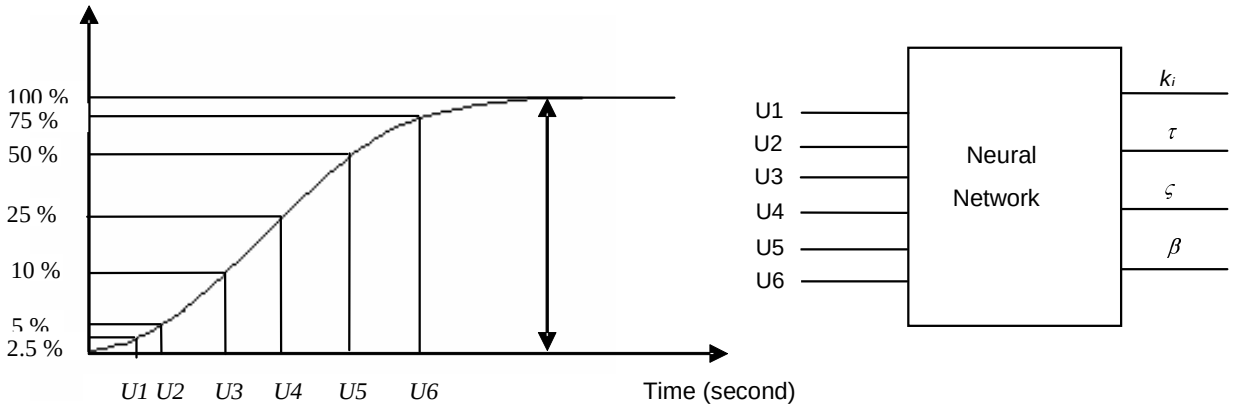

Fig. 2. The values 2.5 %, 5 %, 10 %, 25 %, 50 % and 75 % of the final level of the step response are chosen as input data for processes of type (1). The corresponding PID-parameters are the neural network output data.

III. NEURAL NETWORK TRAINING

First of all, a network must be *trained* and this means that the synaptic weights and thresholds are adjusted by iterative presentation of a set of examples called the *training set*. The training set is a set of input data and corresponding desired output data. In this work, a large number of transfer functions which belong to the three classes of transfer functions above have been created by random. Then, the step responses and PID-parameters for these processes have been calculated. As input data for the neural network, different points at the step responses have been chosen and as output data the PID-parameters are chosen. The purpose of the networks is to calculate PID-parameters for different processes, using only the step response as input data. For processes (1) with 3 to 6 real time constants, we have chosen the time when the step-response has reached 2.5 %, 5 %, 10 %, 25 %, 50 % and 75 % of the final level as input data (Fig 2). For other types of processes, the input data also includes other parameters such as overshoot and dead time.

In general, there are several factors which affect the accuracy of the trained neural network when estimating PID-parameters for different processes. Some examples of these factors are:

- *Network size*: How many layers of neurons and how many neurons are used in each layer? In the research behind this paper, we have used fully connected feed-forward neural networks with one or two layers of hidden neurons and with different number of neurons in each layer. The investigations show that two layers of hidden neurons are required in order to get a satisfactory training result. In this paper, we will present the result for networks with 10 and 21 neurons in the hidden layers of neurons, however networks of other sizes have been studied as well [2,4], but it seems that the sizes presented give a satisfactory training-result. The training time is about 4 times longer for the networks with 21 neurons in the hidden layers.

- *Type of threshold functions*: The type of threshold functions (such as hard-limit transfer function, log-sigmoid transfer function or tan-sigmoid transfer function) in each layer may affect the result of the neural network training. In this paper we will only present results using tan-sigmoid threshold functions.

- *Training algorithms*: The mathematical algorithm which is used for training of a network will also affect how successful the trained network is. In the research behind this paper, the networks have been trained using many different algorithms, such as the Levenberg-Marquardt algorithm, the scaled conjugate gradient algorithm and resilient backpropagation [2,3,4]. The best results, as well as the fastest convergence were in general obtained using the Levenberg-Marquardt algorithm [3,5,6] and only the results using this algorithm will be presented here. For some networks and processes, we also used a two-step training procedure based on Levenberg-Marquardts method. This method is further discussed below.

- *Number of training examples*: In this thesis, 1000 examples have been used for training of the networks and 1000 examples have been used as validation examples.

- *Number of training epochs*: In general, 400 epochs of training was used in all learning experiments. This number of epochs was chosen because in a number of trials, it was shown that the result for test-data could not be further improved by using more than 400 epochs.

The following three methods for tuning of PID-controllers (or PI-controllers) and the following structures of PID-controllers have been used in this report:

- For processes with 3-6 real time constants (1) the Kristiansson-Lennartsson's method (KL) has been used. For a description of the KL-method see [7]. When using the KL tuning method the parameters in the PID-structure (4) is estimated by the neural network.

- For processes with two complex poles and one real pole (2) the Ziegler-Nichols (ZN) tuning method has been used. For a description of Ziegler Nichols method, see [8]. When using the ZN tuning method the parameters in the structure (5) is estimated by the neural network.

- For processes with dead-time and three time-constants we have used PI-controllers and a method which guarantees a phase-margin of 45^{O}. When using this tuning method the parameters in the structure (6) are estimated by the neural network.

$$G_{PID} = k_i \cdot \frac{1 + 2 \cdot \varsigma \cdot \tau \cdot s + (\tau \cdot s)^2}{s \cdot (1 + s \cdot \tau / \beta)} \tag{4}$$

$$G_{PID} = K \cdot (1 + \frac{1}{T_I \cdot s} + \frac{T_D}{1 + T_F \cdot s}) \tag{5}$$

$$G_{PI} = k_i \frac{1 + \tau \cdot s}{s} \tag{6}$$

Table I summarizes all investigated cases, where three different types of processes with three different tuning methods and controllers have been used.

For processes of the first type (1), we have studied the training result, using raw input data to the network as well as normalized (scaled) data. The training result was clearly better when using normalized input data. When normalizing, all time-parameters in the input and output data matrices were divided by the time for the different step responses to reach 50% of their final values. All frequency parameters were multiplied by the same amount. The normalization will scale the inputs and outputs so that they fall in a smaller range. This will, in general, improve the neural network training.

For some networks and processes, we also present the results using a two-step training procedure based on Levenberg-Marquardts method. The first step is an ordinary training procedure using normalized input data and Levenberg-Marquardts method. Then a second step is performed, where an extended training set is used for the network. In the extended training data set, all processes for which the network had a relative error above 1% for any parameter in the first step, were included three times in the extended training set.

IV. PERFORMANCE CRITERIA

When simulating the network with training data or test data an error matrix is obtained. This error matrix is the difference between the desired and the simulated output data matrix. If the network has three output signals and if 1000 examples are simulated, the error matrix has 3 rows and 1000 columns (7).

$$\begin{bmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,1000} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,1000} \\ e_{3,1} & e_{3,2} & \cdots & e_{3,1000} \end{bmatrix} \tag{7}$$

In order to compare different networks, the following performance-measures have been used:

$e_{average}$: The average relative error in percent for each row (i.e. for each estimated parameter).
The average relative error is defined according to equations (8-10) for processes of the first type (1).

$$(\sum_{j=1}^{1000} \frac{|e_{1,j}| \cdot 100}{k_{i(j)}})/1000 \qquad \text{For the first row} \tag{8}$$

$$(\sum_{j=1}^{1000} \frac{|e_{2,j}| \cdot 100}{\tau_j})/1000 \qquad \text{For the second row} \tag{9}$$

$$(\sum_{j=1}^{1000} \frac{|e_{3,j}| \cdot 100}{\beta_j})/1000 \qquad \text{For the third row} \tag{10}$$

e_{max} The maximum relative error for each parameter.

$e_{>2\%}$ Number of relative errors above 2 % for each parameter (in percent of the total number of examples)

The same performance measures have also been used for processes of type (2) and (3) and are defined in a similar way.

V. RESULTS

The result of the neural network training using Levenberg-Marquardts method is dependent on the initial random values for the synaptic weights in the network. Therefore, the result will in general not be the same in two different trials even if the same training examples have been used. In this report, we will only present the best result obtained after a number of trials with the same input-output-data for each model. In general, we have made between 5 and 10 different trials for each network. As the best result, the one that gave the smallest average relative error for test data was chosen.

TABLE I
A SUMMARY OF ALL INVESTIGATED CASES IN THIS PAPER

Process type	Tuning method	Type of controller	Input data	Output data
Process type 1 equation (1)	The KL-method	PID equation (4)	2.5, 5, 10, 25, 50, 75 % of the final level of the step response	k_i, τ and β [3]
Process type 2 equation (2)	The ZN-Method	PID equation (5)	10, 50, 100 % of the final level of the step response and M^1, $T_M{}^1$	$k_0{}^2$ and $T_0{}^2$
Process type 3 equation (3)	Phase margin 45^O	PI equation (6)	2.5, 5, 10, 25, 50, 75 % of final level of step response and d^1	k_i and τ

[1] M, T_M and d denote overshoot, the time for overshoot and dead-time respectively.
[2] k_0 and T_0 are the critical amplitude and the period time where the phase shift is -180°. From these two frequency data and using ZN- method, the PID-parameters i.e. K, T_D, T_I and T_F are obtained.
[3] Note that ζ is not included in the output matrix because the value of ζ is constant and equal to 0.75 according to the KL-method.

A. Networks for Processes of Type 1

In tables II-VI, the result is presented for the neural networks for processes of type (1).

TABLE II
$e_{average}$, e_{max} and $e_{>2\%}$ defined as above for a network of type *6-21-21-3* for both training data and test data.

	Training data			Test data		
	k_i	τ	β	k_i	τ	β
$e_{average}$ (%)	0.51	0.12	0.19	0.69	0.15	0.24
e_{max} (%)	6.34	2.28	2.51	21.10	6.51	9.19
$e_{>2\%}$ (%)	1.8	0.2	0.4	3.3	0.70	0.8

TABLE III
$e_{average}$, e_{max} and $e_{>2\%}$ defined as above for a network of type *6-10-10-3* for both training data and test data.

	Training data			Test data		
	k_i	τ	β	k_i	τ	β
$e_{average}$ (%)	3.33	1.09	1.13	4.70	1.26	1.38
e_{max} (%)	40.32	61.82	17.02	471.87	54.25	30.00
$e_{>2\%}$ (%)	61.3	12.50	13.10	62.20	12.80	17.00

TABLE IV
$e_{average}$, e_{max} and $e_{>2\%}$ defined as above for a network of type *6-21-21-3* for both training data and test data *after normalization*.

	Training data			Test data		
	k_i	τ	β	k_i	τ	β
$e_{average}$ (%)	0.064	0.046	0.050	0.073	0.056	0.055
e_{max} (%)	0.50	1.17	0.36	2.53	4.37	0.70
$e_{>2\%}$ (%)	0	0	0	0.1	0.3	0

TABLE V
$e_{average}$, e_{max} and $e_{>2\%}$ defined as above for a network of type *6-21-21-3* for both training data and test data *after normalization and two step training*.

	Training data			Test data		
	k_i	τ	β	k_i	τ	β
$e_{average}$ (%)	0.053	0.044	0.035	0.061	0.046	0.043
e_{max} (%)	0.22	0.49	0.29	1.42	0.99	1.078
$e_{>2\%}$ (%)	0	0	0	0	0	0

TABLE VI
$e_{average}$, e_{max} and $e_{>2\%}$ defined as above for a network of type *6-10-10-3* for both training data and test data *after normalization*.

	Training data			Test data		
	k_i	τ	β	k_i	τ	β
$e_{average}$ (%)	0.13	0.14	0.075	0.15	0.13	0.083
e_{max} (%)	1.021	3.59	1.13	4.62	5.65	2.48
$e_{>2\%}$ (%)	0	0.4	0	0.5	0.3	0.1

B. Networks for Processes of Type 2

In tables VII-VIII, the result is presented for the neural networks for processes of type (1).

TABLE VII
$e_{average}$, e_{max} and $e_{>2\%}$ defined as above for a network of type *5-21-21-2* for both training data and test data.

	Training data		Test data	
	k_0	T_0	k_0	T_0
$e_{average}$ (%)	0.028	0.010	0.032	0.013
e_{max} (%)	0.20	0.070	0.20	0.22
$e_{>2\%}$ (%)	0	0	0	0

TABLE VIII
$e_{average}$, e_{max} and $e_{>2\%}$ defined as above for a network of type *5-10-10-2* for both training data and test data.

	Training data		Test data	
	k_0	T_0	k_0	T_0
$e_{average}$ (%)	0.051	0.016	0.055	0.018
e_{max} (%)	0.41	0.098	0.43	0.12
$e_{>2\%}$ (%)	0	0	0	0

C. Networks for Processes of Type 3

In tables IX-X, the result is presented for the neural networks for processes of type (1).

TABLE IX
$e_{average}$, e_{max} and $e_{>2\%}$ defined as above for a network of type *7-21-21-2* for both training data and test data.

	Training data		Test data	
	k_i	τ	k_i	τ
$e_{average}$ (%)	0.022	0.023	0.045	0.032
e_{max} (%)	0.21	0.47	3.47	1.25
$e_{>2\%}$ (%)	0	0	0.2	0

TABLE X
$e_{average}$, e_{max} and $e_{>2\%}$ defined as above for a network of type *7-10-10-2* for both training data and test data.

	Training data		Test data	
	k_i	τ	k_i	τ
$e_{average}$ (%)	0.023	0.024	0.044	0.17
e_{max} (%)	0.30	0.30	4.77	84.57
$e_{>2\%}$ (%)	0	0	0.4	0.4

VI. CONCLUSIONS

From the results presented above, the following conclusions can be drawn:

- The neural network training result for processes of type (2) is very good. For processes of this type, the maximum error is only 0.43% for test data when using a 5-10-10-3-network and it is 0.22% for a 5-21-21-2-network.

- For processes of type (3), the result is also satisfactory. The maximum error is 3.47% for a 7-21-21-2-network and only 0.2% of the processes have an error above 2% for any parameter. The result shows that the networks perform well for this type of processes.

- For processes of type (1) the training result is not satisfactory when using raw training data. The maximum error is 21.1% for a 6-21-21-3-network and 3.3% of the test processes have an error above 2%. When using normalized input data, the performance is clearly better. Here, the maximum error is 4.37% and 0.3% of the processes have an error above 2%.

- The best networks for processes of type (1) were obtained when using the two-step procedure for training of the neural networks. In particular, it was possible to reduce the maximum error and the amount of processes with an error above 2%. For the best network. The maximum error was 1.4 % and no processes had an error above 2 %.

VII. DISCUSSION AND FUTURE WORK

In this paper, it was shown that neural networks can be used to estimate PID-regulator parameters for different classes of dynamic processes. By measuring a number of points at the step-response of a process and using them as input to a successfully trained neural network, the network can estimate the PID-parameters for the same process according to well-known tuning rules for PID-controllers. The estimation could be done with good accuracy for the classes of dynamic processes studied and for the tuning methods discussed.

However, even if the method has proved to work well for the processes studied, the networks obtained are not yet developed to take care of all possible types of processes and they have not yet been tested using data from real processes. In this section, the following topics will be discussed:

- How to make the PID-tuning-networks better when the input data is disturbed by a lot of measurement noise? How to improve the robustness and accuracy of the networks?

- How to construct PID-networks for broader classes of processes, such as processes with integration and non-minimum phase processes?

In this report, we have studied the ability of neural networks to estimate PID-parameters from time domain data for a number of simulated processes. The results show that the networks are well suited for this problem. However, in real applications, we have to deal with processes having a lot of noise (measurement noise and process noise). It is easily shown that the estimations using the PID-networks are rather sensitive to disturbances. That is, a small change in one or more of the input parameters may lead to great changes in the PID-parameter estimations.

There are several methods to reduce the sensitivity of the estimations. One method is to reduce the noise level at the input data by measuring several step responses and then calculate the average values of the input parameters. Another way may be to build networks, which uses additional information as input data. Probably the training-result and the accuracy of the network can be improved by using additional input data to the network. Instead of only using points at the step response, we can also use points at the impulse response or points at other transient responses. In this report, we have used the time when the step response has reached 2.5, 5, 10, 25, 50 and 75% of the final value as input information for processes of type (1). The reason, using smaller time-intervals at the beginning of the step-response, is that the start of the transient gives more accurate information about the high-frequency characteristics of a process. However, this information may also be obtained by measuring different points at the impulse response.

The networks discussed in this report have used training-data from processes with three to six time-constants, as well as, processes with dead time and complex poles. However, it would be desirable to have networks, which can estimate PID-parameters for broader classes of processes such as processes with integration and non-minimum phase processes. When estimating PID-parameters for these types of processes, however, the network has to be more complex and it probably has to use different input data for different types of processes.

For a process with overshoot, it is natural to use the overshoot and period time as input data to the network, for processes with integration, it is natural to use the slope of the step response and so on.

In short, networks for broader classes of processes can work according to two principles. The simplest approach is that the user makes the classification of the processes and then uses different networks for different classes of processes. Some important classes of processes are:

1. Processes with s-formed step response, but no overshoot.
2. Processes with integration.
3. Processes with overshoot.
4. Non minimum phase processes.
5. Instable processes.

The other approach is to use a separate network for classification of the process and then automatically chose one of different networks for different classes of processes. When this approach is used, the network needs more input data, not only the six or seven points at the step response. In order to make good estimations, the network perhaps needs the complete step response and other input information as well.

VIII. REFERENCES

[1] Åström, K. J. and Hägglund, T., *PID Controllers: Theory, Design and Tuning*, Instrument Society of America, 1995.

[2] Andersson, R. and Persson, A., *Uppskattning av regulatorparametrar med artificiella neurala nätverk*. Bachelor thesis, Chalmers Lindholmen University College, Göteborg, 2003.

[3] Demuth, H. & Beale, M., *Neural Network Toolbox for use with Matlab*, The Math Works Inc, User Guide, Natick, USA, 2000.

[4] Hammarström, R. and Larsson, J., *Dimensionering av PID-parametrar med hjälp av neurala nätverk*, Bachelor thesis, Chalmers Lindholmen University College, Göteborg, 2003.

[5] Hagan, M. T., and M. Menhaj, *Training feedforward networks with the Marquardt algorithm*, IEEE Transactions on Neural Networks, vol. 5, no 6, pp. 989-993, 1994.

[6] Hagan, M. T., H. B. Demuth, and M. H. Beale, *Neural Network Design*, Boston, MA: PWS Publishing, 1996.

[7] Kristiansson, B., *PID Controllers, Design and Evaluation*, Technical Report No 453, Control and Automation Laboratory, Chalmers University of Technology, 2003.

[8] Ziegler, J. G. and Nichols, N. B., Optimum settings for automatic controllers, Trans ASME, vol 64, No 11, 1942.