Applications of Artificial Neural Networks For Regulation of Temperature in a Tank

SILVIYA KACHULKOVA
Faculty of Automatics, Department of Electrical Measurement
Technical University of Sofia
8, Kliment Ohridski Blvd. 1000 Sofia
BULGARIA
kachulkova@abv.bg

Abstract: - That paper describes the usage of Artificial Neural Networks (ANN) which gives us the advantage in control systems to solve and examine the problems with nonlinearities, complex plant modeling and prediction. One of the objectives of the current project is to develop an integrated control system, which consists of a plant (physical object, which should be controlled) – a liquid heating process, which temperature should be controlled via a real actuator.

Keywords: - Ziegler-Nichols models, Artificial Neural Networks, control the temperature of water in a tank, ANN plant predictor, PI controller, Accuracy of the Predictor

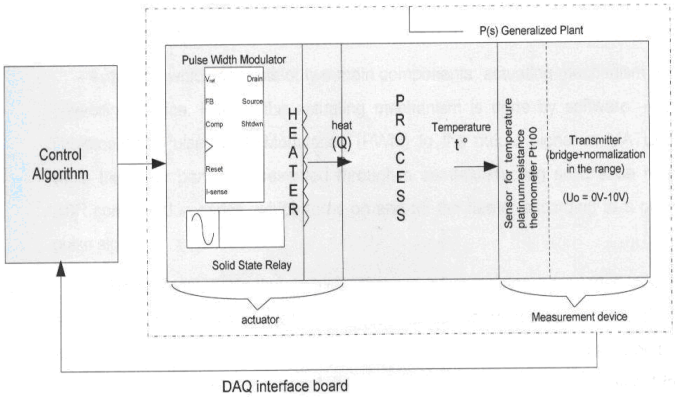
1 Introduction

In this research, the complex nonlinear plant-the process of heating liquid in a tank is investigated by real time experimentation with the help of MATLAB and SIMULINK. Thus the plant dynamic and static characteristics are examined and Ziegler-Nichols models derived, which parameters in different operating points differ. This material gives the grounds to develop algorithm for the control of the plant process using ANN for prediction of the plant output in order to compensate the plant inertia, time-delay and model parameter changes. One of the objectives of the current project is to develop an integrated control system, which consists of a plant (physical object, which should be controlled) – a liquid heating process, which temperature should be controlled via a real actuator.

A primary sensor – resistance thermometer Platinum 100 measures the temperature, then a temperature transmitter put it into voltage signal in the range 0-10V, which is sent to a Data Acquisition Board DAQ (interface board, produced by NI [National Instruments] with Analog-to-Digital Converter and Digital-to-Analog Converter). Thus the real physical signal is converted into binary code, which will be used by the computer in the software realization of the control algorithm.
The components of the integrated system can be divided into:

- Plant (to be controlled) represented by the heating process – the temperature of the water in the tank, which is regulated by the heat, generated from a heater [5].
- Actuator, which consists of two main components: actuating mechanism and controlling device. Part of the actuating mechanism is done by software – by implementing Pulse-Width Modulation (PWM) to the output signal in MATLAB, while the other part is represented through a semi-conducting solid state relay SSR connected in series, which turns on and off the heater according to a given pulse signal.
- Measurement device**,** which consists of a primary converter and a transmitter. The primary converter measures the current

The following scheme illustrates the structure of the system.



temperature of the plant, which is converted into voltage. The presence of the transmitter is required by the limited range of the electrical board. It is a bridge scheme with forming element for U_0 =0V-10V (U_0 – output voltage).

- Control algorithm – it is realized through MATLAB. Its operation is based on the change of the control variable – in this case the heat, generated from the heater by controlling the relative time for switching on of the electrical heater.
- DAQ interface board produced by NI with ADC (Analog-to-Digital Conversion) and DAC (Digital-to-Analog Conversion) for taking and converting the signals from and to the computer, as well as for synchronizing of the signals.

2 Determination of Plant Step Responses and Ziegler-Nichols Models

After the construction of the physical model of the plant and the realization of the actuating and measuring devices, it is necessary the SIMULINK application to be used for determination of the characteristics of the plant, which will allow the observation of the wanted parameters to be much easier.

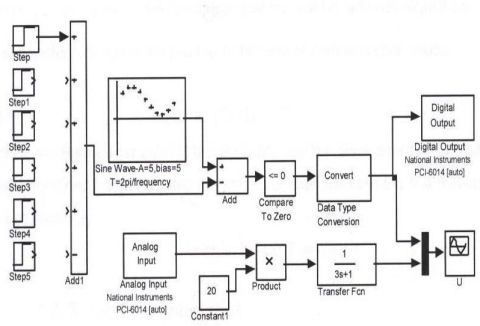


Fig. 1 Block diagram of the SIMULINK model used for the determination of plant step responses

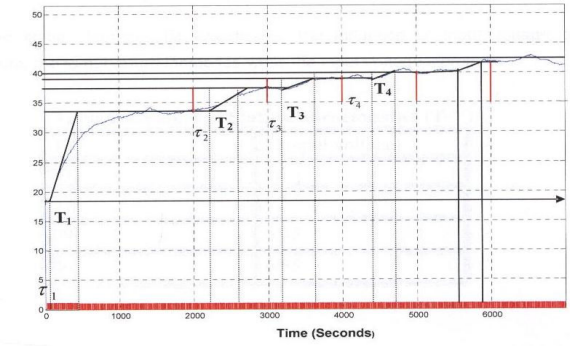


Fig. 2 Transient responses of the plant

The experiments led to the conclusion that the plant is with self-regulation.

The transfer function of the plant is in the form of Ziegler-Nichols model:

$$P(s) = K.(Ts+1)^{-1}.e^{-\varpi} \quad (1)$$

where the time constant (T), the Gain (K) and the time delay (τ) of the plant are determined from each step response, for example from the first response they are respectively:

T_1=250s; K_1=60; τ_1 =70s.

We can directly construct from steady states in Fig.2 the static characteristics of the plant – Plant Output versus Plant Input in steady state, shown in Fig.3, Fig.4 and Fig.5 with the values correspondingly in Table 1, Table 2 and Table 3.

The static characteristic investigates the influence of input voltage on temperature; the temperature increases while the input voltage increases.

Table 1 Values of Static Characteristics (T-V)

T°C	V [input voltage]
33	0.25
37	0.35
39	0.45
40	0.55

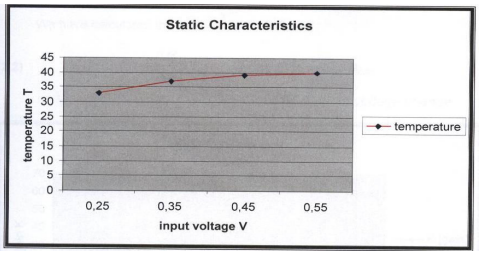


Fig. 3 Static characteristics of the plant

The second characteristic investigates the influence of input voltage U,V on plant model gain K; gain decreases with the increase of the input voltage.

Table 2 Values of Characteristic (K-U)

Gain(K) $k = \frac{\Delta\theta}{\Delta u}$	U,V [input voltage]
60	0.25
40	0.35
20	0.45
10	0.55

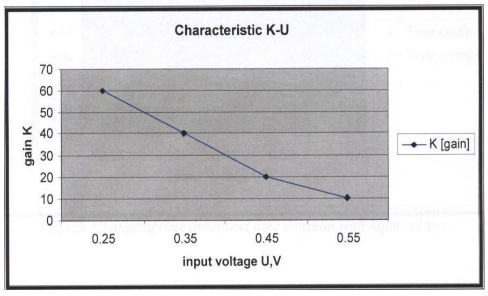


Fig. 4 Change of plant gain with input voltage

We have calculated the gain, using the formula:

$$k = \frac{\Delta\theta}{\Delta u} \qquad (2)$$

where $\Delta\theta=\theta_{final}-\theta_{initial}$

Δu = input voltage change

The third characteristic investigates the influence of the input voltage on the time delay τ and the time constant T.

Table 3 Values of Characteristics (τ,T-U)

Time delay	V [input voltage]	Time constant
70	0.25	250
260	0.35	350
200	0.45	600
450	0.55	500

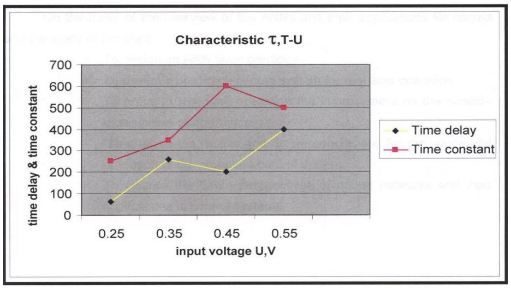


Fig. 5 Change of time delay and time constant with input voltage

As a conclusion, after such a non-linear behavior of the plant, as seen from the static characteristics for the control of the plant, and changes of plant model parameters in the different operation points (different input voltage) more sophisticated controller should be used. That is why the next section is based on Neural Network Control Algorithms, especially ANN Predictive Controller.

The aim of the present project is to control the temperature of water in a tank by electrical heater, which is a nonlinear inertial plant with time delay and disturbances on coolant; using ANN predictor to improve control system performance.

3 Design of an ANN Plant Predictor

The Simulink Model of the closed loop PI control System, used for gathering training data for the developing of the predictor, is shown in Fig.6.

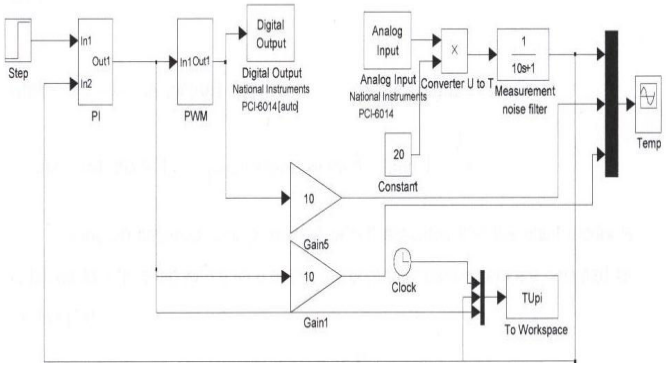


Fig. 6 Step Response of Closed Loop System with PI Controller without Predictor

In Fig. 7 the relation between the measured temperature and the voltage as functions of time is presented.

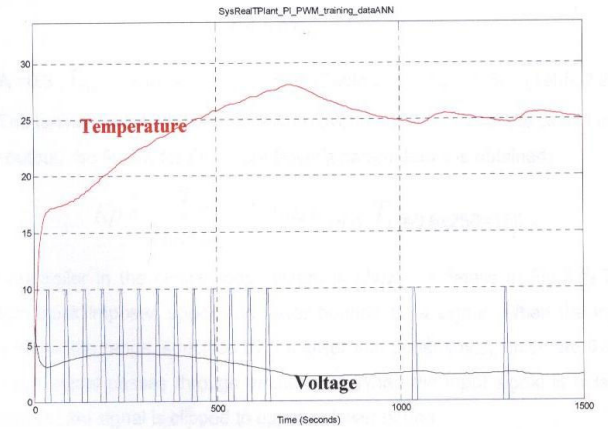


Fig. 7 Temperature (red line) and Voltage (black line)

The PI controller control the plant, using the error as the difference between the desired set output and the measured output and the integral of the error. The PI controller has the following transfer-function:

$$C_{PI}(s) = K_P\left(1+\frac{1}{T_i s}\right) \qquad (3)$$

where K_P – is the controller gain and T_i –I s the integral action time. In SIMULINK the PI block parameters are P= K_P and I= K_P/ T_i.

Using an empirical tuning method, which assumes that the plant model is obtained as a time lag with a time delay, the controller parameters are obtained as follows [6]:

$$K_p = \frac{AT_{\min}}{k_{\max}\tau_{\max}}, \qquad T_i = 0.6T_{\min} \tag{4}$$

Where A=0.3, T_{min}=250s, τ_{max}=450s (Table 3), k_{max}=60 (Table 2).

The tuning criterion is taken to ensure overshoot σ=20% of the closed loop system output. For the PI controller's parameters it is obtained: K_P=0.0028 and T_i=150s.

The PI controller in the closed loop system is shown in details in Fig. 8.

The Saturation block imposes upper and lower bounds on a signal. When the input signal is within the range, specified by the lower and upper limits, which are 0÷10V, the input signal passes through unchanged. When the input signal is outside these bounds, the signal is clipped to upper or lower bound.

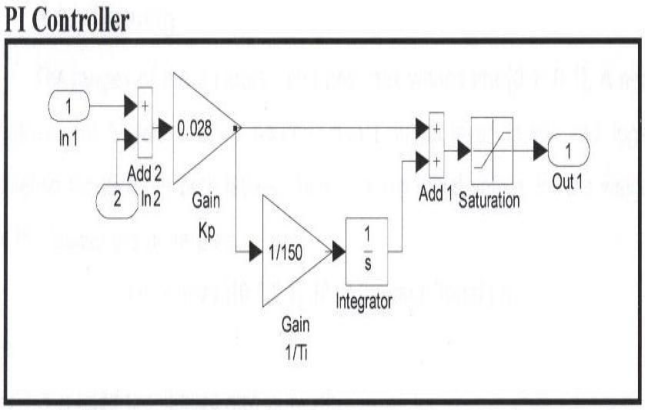


Fig. 8 PI controller Simulink model

Pulses at the PWM output appear with magnitude of 1V and width, dependent on the analogue voltage. The duty ratio is the duration of pulses/period of pulses with period T, tuned by the frequency f of sine waves

$$T = \frac{2\Pi}{f} \tag{5}$$

The block diagram of the Pulse-Width Modulator in the closed loop system in Fig.6 is shown in detail in Fig. 9.

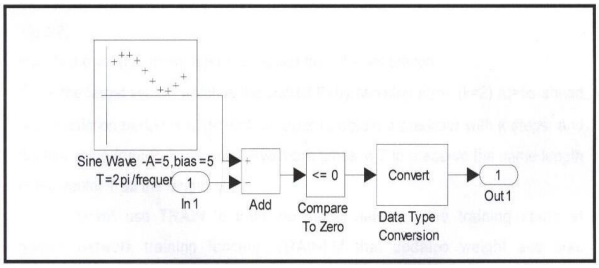


Fig. 9 Pulse-Width Modulator Simulink Model

The ranges of the 2 inputs – min and max values are [0 1;0 1]. A net is initialized with 8 hidden layers neurons and 1 output layer neuron and 'logsig' activation functions in both layers. Also, random initial values for the weights and the biases are generated.

net = newff([0 1;0 1],[8 1],{'logsig' 'logsig'});

where newff – is used to initialize new network.

The net training parameters are assigned for the number of epochs and the accuracy goal.

net.trainParam.epochs = 20000;

net.trainParam.goal = $1.e^{-10}$;

In the m file which trains the neural network, are used the following variables and functions:

P – is the training input vector, which consists of P_1 and P_2;

P_1 – is the normalized temperature (plant output), collected from the simulation (Fig. 7).

P_2 – is the voltage (plant input), collected from the simulation;

T – is the target vector, which is the shifted P_1 by two step sizes (k=2) Δt=5s ahead (the prediction period is k. Δt=10s), in order to obtain a predictor with k steps; and the last value from P_1 is repeated two more times in T to preserve the same length of the vector T as the length of P_1.

Use is TRAIN to train the model network. The training starts at default network training function TRAINLM that updates weight and bias values according to Levenberg-Marquardt optimization (a modification for speeding up the steepest descent method). The default criterion is MSE (Mean Squared Error).

net = train(net, P, T);

3.1 Network training

The Levenberg-Marquardt network training function (trainlm.m) is used to update weight and bias values and to obtain a solution in shorter time. The network is trained for up to 20 000 epochs, displaying progress of every epoch. The epochs

represent the number of iterations of plant training to be performed.

Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares (as is typical in training feed forward networks), then the Hessian matrix can be approximated as

$$H = J^T J \quad (6)$$

and the gradient can be computed as

$$g = J^T e \quad (7)$$

where J is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases, and e is a vector of network error. The Jacobian matrix can be computed through a standard backpropagation technique that is much less complex than computing the Hessian matrix.

The Levenberg-Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton-like update:

$$X_{k+1} = X_k - \left[J^T J + \mu I\right]^{-1} J^T e \quad (8)$$

When the scalar μ is zero, this is just Newton's method, using the approximate Hessian matrix.

When μ is large, this becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift toward Newton's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function is always reduced at each iteration of the algorithm.

This algorithm appears to be the fastest method for training moderate-sized feed forward neural networks (up to several hundred weights). It also has a very efficient MATLAB implementation, because the solution of the matrix equation is a built-in function, so its attributes become even more pronounced in a MATLAB setting.

The main drawback of the Levenberg-Marquardt algorithm is that it requires the storage of some matrices that can be quite large for certain problems. The size of the Jacobian matrix is Q x n, where Q is the number of training sets and n is the number of weights and biases in the network. It turns out that this matrix does not have to be computed and stored as a whole. For example, if you were to divide the Jacobian into two equal submatrices you could compute the approximate Hessian matrix as follows:

$$H = J^T J = \left[J_1^T J_2^T\right]\begin{bmatrix} J_1 \\ J_2 \end{bmatrix} = J_1^T J_1 + J_2^T J_2 \quad (9)$$

Therefore, the full Jacobian does not have to exist at one time. You can compute the approximate Hessian by summing a series of subterms. Once one subterm has been computed, the corresponding submatrix of the Jacobian can be cleared.

When you use the training function trainlm, the parameter mem_reduc determines how many rows of the Jacobian are to be computed in each submatrix. If mem_reduc is set to 1, then the full Jacobian is computed, and no memory reduction is achieved. If mem_reduc is set to 2, then only half of the Jacobian is computed at one time. This saves half the memory used by the calculation of the full Jacobian.

There is a drawback to using memory reduction. A significant computational overhead is associated with computing the Jacobian in submatrices.

If you have enough memory available, then it is better to set mem_reduc to 1 and to compute the full Jacobian. If you have a large training set, and you are running out of memory, then you should set mem_reduc to 2 and try again. If you still run out of memory, continue to increase mem_reduc.

Even if you use memory reduction, the Levenberg-Marquardt algorithm will always compute the approximate Hessian matrix, which has dimensions n x n . If your network is very large, then you might run out of memory. If this is the case, try trainscg, trainrp, or one of the conjugate gradient algorithms.

When the training is over, the net Simulink block for the trained net with sample time Δt = 5 is generated by using gensim(net,5), "gensim" means generation of Simulink block for the trained network.

For the predictor the error is shown in Fig. 10

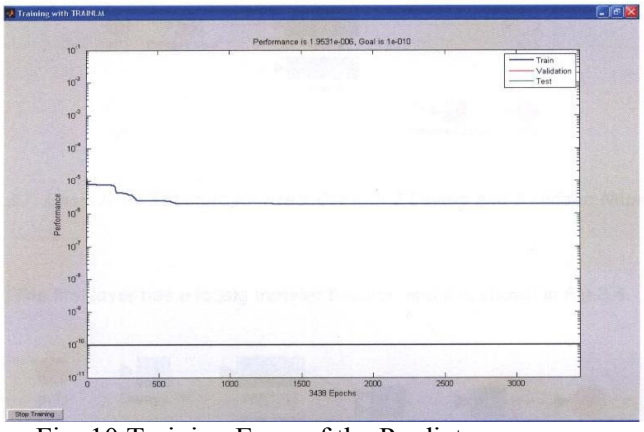


Fig. 10 Training Error of the Predictor

After we finished the training, the goal is not reached but the accuracy is high enough. As a result the Simulink Block is generated, as shown in Fig.11.

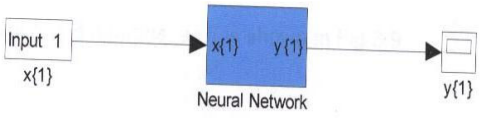


Fig. 11 Neural Network Predictor Simulink Block

The two layers of the network are presented in Fig.12.

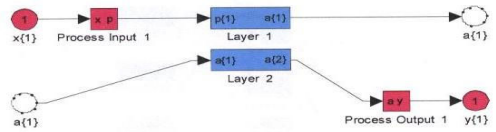


Fig. 12 Basic ANN Structure for Predictor with 2 layers and 8 hidden neurons

The first layer has a logsig transfer function and has 8 hidden neurons. The second layer has one neuron and logsig function. The values for the weights and the biases for the two layers of the predictor are given below.

Weight and Bias Matrix of First Layer of ANN:

$$W^1 = \begin{vmatrix} 9.10 & 13.05 \\ 9.37 & 11.86 \\ -34.82 & -217.85 \\ 36.97 & -2.103 \\ 69.74 & 126.75 \\ -34.38 & -214.15 \\ -0.612 & 9.250 \\ 2.177 & -15.71 \end{vmatrix}$$

b^1=[-18.71,-18.76,21.71,-17.121,-43.00,21.40,-1.66,14.57]

Weight and Bias Matrix of Second Layer of ANN:

$$W^2 = \begin{vmatrix} -0.791 & 1.195 \\ 12.03 & 0.258 \\ 0.266 & -12.52 \\ -6.176 & 3.804 \end{vmatrix} \qquad b^2 = [-2.963]$$

After receiving the ANN Predictor as a result from our training in Matlab, we construct a Simulink Model of closed loop system with PI Controller and ANN Predictor for testing the accuracy of the designed predictor.

The Simulink block diagram with the plant predictor is shown in Fig. 13. The sample time is 5s, reference change is from 20 to 25 deg C. According to this Simulink Model the ANN predictor output is delayed by 10s to fit and to allow comparison with the real plant output.

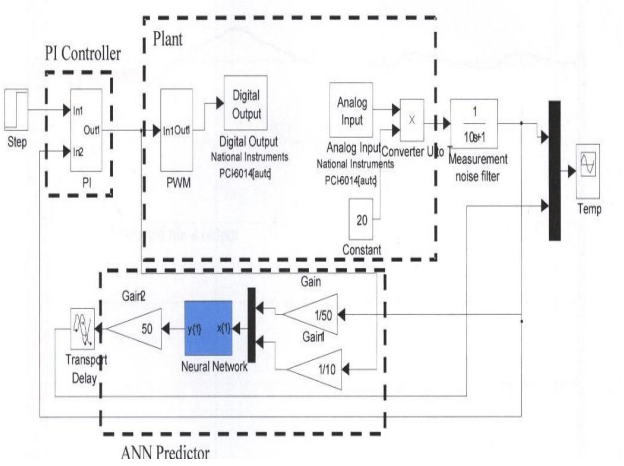


Fig. 13 Simulink Model for Testing the Accuracy of the Predictor

In Fig. 14 are shown the step responses of the plant in the closed loop system and the predictor which input signals are taken from the plant input and output in real time.

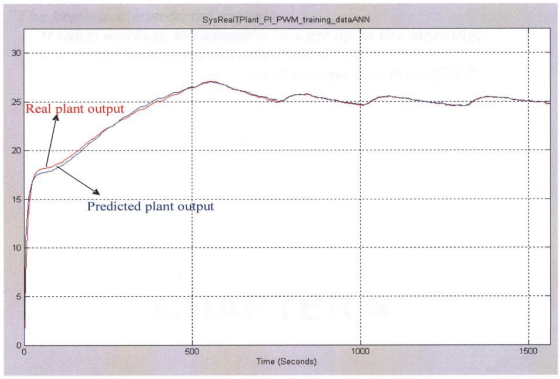


Fig. 14 Predictor and Plant step responses for the whole simulation period

The same input signals for the ANN are used both for the test of the predictor's accuracy and its training. The step size Δt=5s is also the same. The general behavior is estimated as good in the training

points and the accuracy reached is high. As a result the ANN behaves as an accurate predictor.

4 Design of Predictive Control System for Real Time Operation

The design of the Predictive Controller (PC) is based on the tuned PI controller and the designed ANN Plant Predictor. The block diagram of the predictive control system is represented in Fig. 15. The ANN predictor compensates the plant delay by supplying the PI controller with advanced information on the plant output.

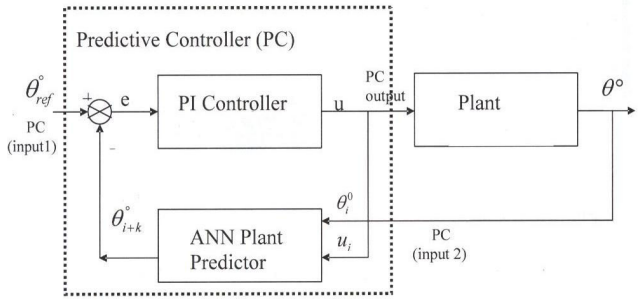


Fig. 15 Functional Block Diagram of Predictive Control System

The PI controller compares the measured process output θ° with a reference set-point value $\theta°_{ref}$. The difference e is then processed to calculate a new process input u_i, which will keep the output process variable close to the desired set-point $\theta°_{ref}$.

The PI controller is capable of manipulating the process outputs, using the advanced information from the ANN predictor.

The Simulink model of closed loop predictive control is shown in Fig. 16.

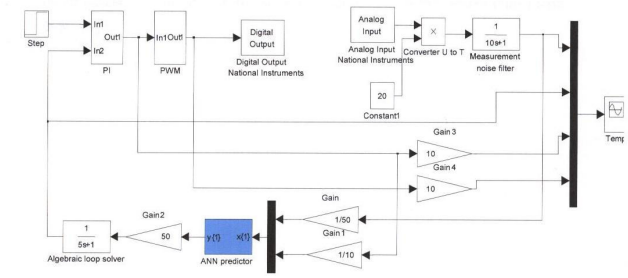


Fig.16 Simulink Model of Closed Loop Predictive Control System in Real Time

The design of the Predictive Control System requires:

- To design the PI Controller and tune its parameters according to the formula (3);
- To design the ANN predictor;
- To design the configuration of the predictive control system according to Fig. 15 and Fig. 16.

The new Predictor Controller (Fig. 15) has two inputs – the reference temperature $\theta°_{ref}$ and the measured temperature (plant output) at current time moment $\theta°_{1}$, and output – the control action u.

A filter "Algebraic loop solver" in Fig. 16, in the form of time-lag with very small time-constant T=Δt=5s solves Simulink loop problems.

Gains at ANN predictor input are used to normalize the ANN input signals in the ranges 0 ÷ 1. The ANN predictor output is in the same range 0 ÷ 1 and is denormalized by the gain at the ANN predictor output in the temperature range 0 ÷ 50 deg C.

5 Investigation of the Closed Loop System in Real Time

5.1 Real Time Investigations of Closed Loop System with PI Controller

The aim is to control the nonlinear plant with time delay and variable model parameters, using PI controller and predictive PI controller. First, experiments are carried out on the ordinary closed system with PI controller and after that 0 on the closed loop system with predictive PI controller (using ANN plant predictor)[9], applying different sample period and parameters of the noise filters and the algebraic loop solver. At the end the step responses of the two systems to different reference temperature changes are compared.

In Fig. 17 is shown the Simulink block diagram of the control system. First the step input (reference change) is from 25 to 30 deg C, the sample time is Δt=0.5s, and the noise filter's time-constant is T_n=3s. The PI Controller parameters are K_p=0.0028 and T_i=150s and he stabilizes the plant output – the measured temperature.

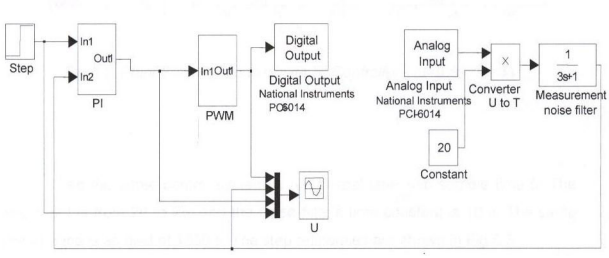


Fig. 17 Simulink Block Diagram with PI Controller

The plant output response is shown in Fig. 18. A disturbance – addition of cold water in the tank, is

applied at 3000s in order to study the response of the closed loop system and the reaction of the controller at disturbances at the plant output.

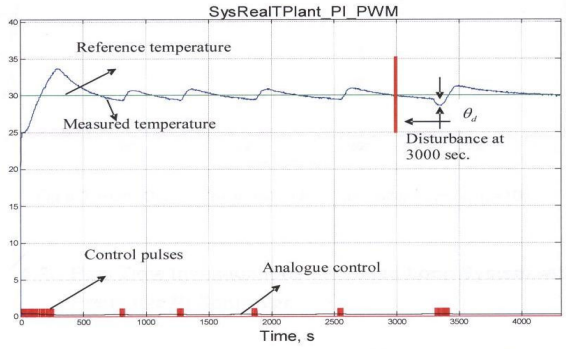


Fig. 18 Plant Step Responses with PI Controller Δt=0.5s, T_n=3s

Then the same control system is run in real time with sample time 5. The step input is from 20 to 25 , and the noise filter's time constant is 10s. The same disturbance is applied at 1500s. The step responses are shown in Fig. 19.

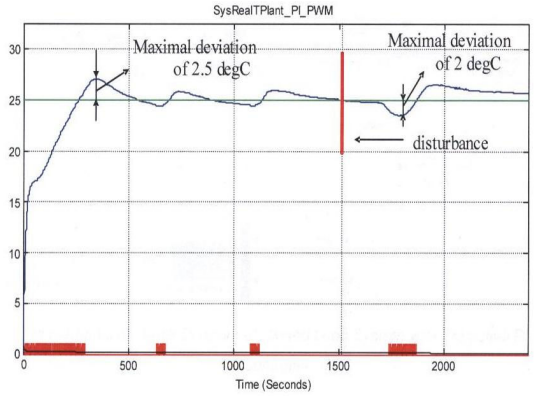


Fig. 19 Plant Output Responses with PI Controller Δt=5s, T_n=10s

5.2 Real Time Investigations of Closed Loop System with Predictive PI Controller

The next experiments are related to control of the plant using Predictive PI Control System. The Simulink Block diagram of the closed loop system with the Predictive PI Controller is shown in Fig. 20. The input is step with initial value of 20 and final 25 deg C. The sample time is Δt=0.5s, the noise filter's time-constant is T_n=10s and the algebraic loop solver time constant is T_a=5s. Gain blocks are also used, since the input of the Neural Network is normalized in training. The system outputs to be recorded and analyzed are the plant output, the Predictor output, the Analogue control, the control pulses and the reference and their step responses are shown in Fig. 21.

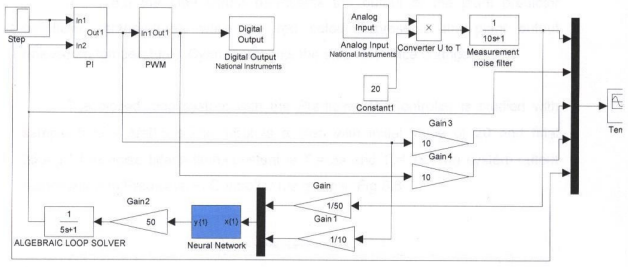


Fig. 20 Simulink Block Diagram of Closed Loop System with Predictive PI Controller

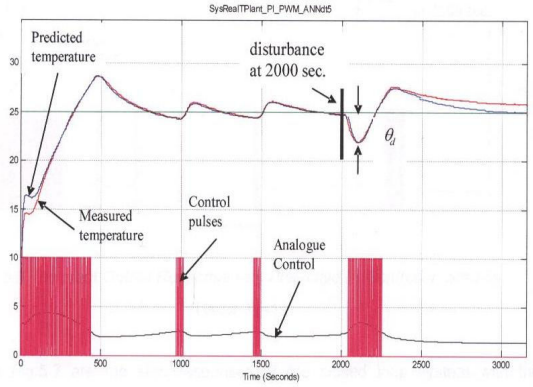


Fig. 21 The System Output Response with Predictive PI Controller Δt=0.5s, T_n=10s, T_a=5s

In Fig. 21 the blue colour represents the output of the plant predictor (predicted temperature), while the red colour represents the plant output (measured temperature). Cyan represents the step reference change.

The closed loop system with the Predictive PI Controller is studied with sample time - Δt=0.5s. The input is a step with initial value of 20 and final 25 deg C. The noise filter's time-constant is T_n=3s, T_a=1s. The system output responses with Predictive PI Controller are given in Fig. 22.

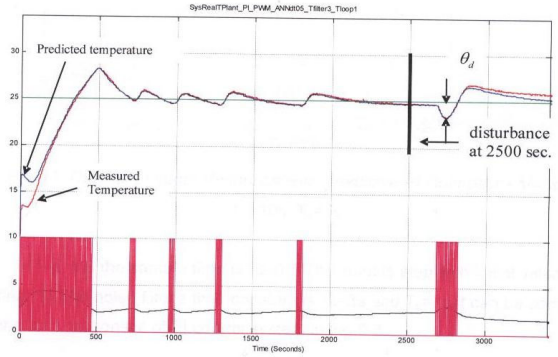


Fig. 22 The plant Output Response with Predictive PI Controller Δt=0.5s, T_n=3s, T_a=1s

In Fig. 23 are the step responses of the closed loop system with the predictive PI controller with sample time - Δt=5s, noise filter time-constant - T_n=10s and T_a=5s in order to compare with the responses of the system with PI controller, shown in Fig. 19. The blue color represents the predicted temperature and the red color represents the measured temperature.

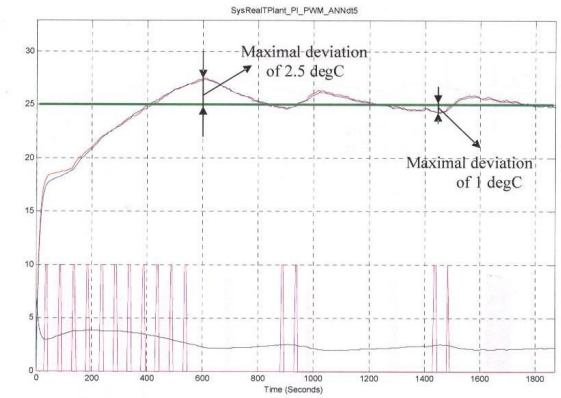


Fig.23 The plant Output Response with Predictive PI Controller Δt=5s, T_n=10s, T_a=5s

In Fig. 24 the sample time is Δt=0.5s. The input is step with initial value of 25 and final 30. The noise filter's time-constant is T_n=3s and T_a=1s. It can be compared with the step responses of PI control system in Fig. 18.

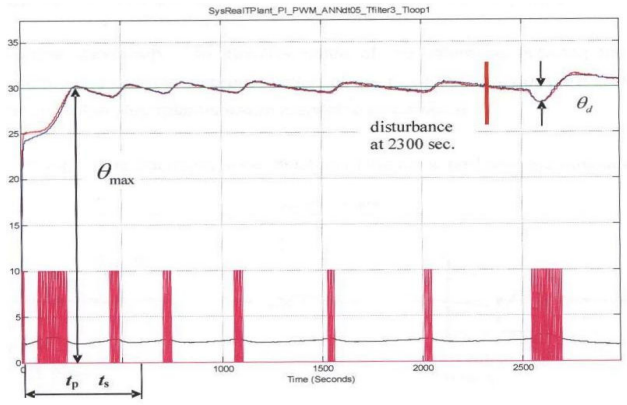


Fig. 24 The plant Output Response with Predictive PI Controller Δt=0.5s, T_n=3s, T_a=1s

5.3 Assessment of Performance measures

The following performance measures are considered.

- Peak time t_p – time required to reach the first peak of the response over the reference.
- Overshoot percent – the relative with respect to reference amount the response overshoots the final (steady) state value:

$$\sigma = \frac{\theta_{\max} - \theta_{ref}}{\theta_{ref}} * 100\% \qquad (10)$$

- Settling time – the time t_s, for which the output reaches steady state.
- Maximal deviation – the absolute value of the difference between the first extremum value and the reference value of the plant output Δθ; here it will be estimated for the step response with respect to disturbance.

In Fig. 25 is illustrated the evaluation of the basic performance measures.

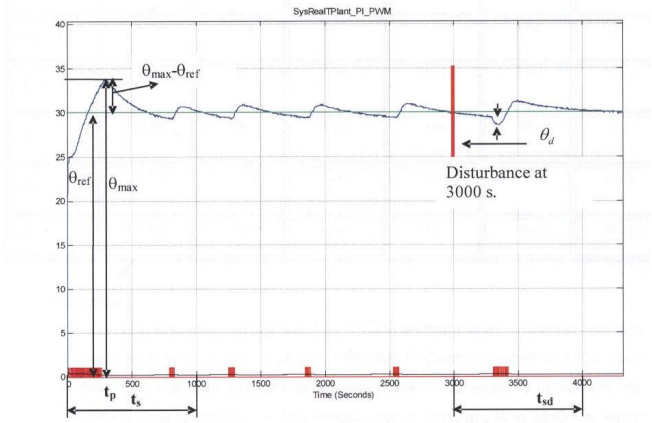


Fig. 25 Evaluation of Control System in Fig. 18 Overshoot, Settling Time, Peak Time and Maximal Deviation with Respect to Disturbance

Table 4 is filled with the assessed performance measures of the two investigated systems from the step responses in the given figures.

Table 4 Performance Measures Comparison of the Systems with the two controllers

	PI Control System		Predictive PI Control System			
	Fig.18	Fig.19	Fig.21	Fig.22	Fig.23	Fig.24
Performance Measures	Δt=0.5s T_n=3s	Δt=5s T_n=10s	Δt=0.5s T_n=10s T_a=5s	Δt=0.5s T_n=3s T_a=1s	Δt=5s T_n=10s T_a=1s	Δt=0.5s T_n=3s T_a=1s
σ, %	13%	12%	16%	12%	12%	0%
t_p, s	300s	350s	450s	500s	600s	270s
t_s, s	1000s	900s	900s	900s	1200s	700s
t_s, dist.	1000s	1200s	1200s	900s	-	1000s
Δθ at dist.	1.5degC	2degC	4degC	2degC	1degC	2degC

The comparison of Fig.18 with Fig. 21 shows that the PI Controller has less overshooting (σ = 13%) than the Predictive PI Control System at 450s peak time. The maximal deviation of the Predictive PI Controller is higher and the settling time is smaller, so this means that at constant time Δt=0.5s, when T_n is increasing, overshooting (σ) and the settling time

t_s are also increasing. In this case, the predictor does not perform well.
The comparison of Fig.18 with Fig. 22 shows that the Predictive PI Controller has less overshooting σ = 12% at 500s, when the only difference is the algebraic loop server time constant T_a=1s. The settling time of the system with PI Control system t_s = 900s is smaller while the maximal deviation is only 0.5degC greater.
The comparison of Fig.18 with Fig. 24 shows that the Predictive PI Control System has no overshooting at all and the settling time is less than the settling time of the PI Control System. The maximal deviation of the Predictive Control System is 0.5degC greater. In this case, the predictor improves and responds better.
The comparison of Fig.19 with Fig. 23 shows that there is no change in overshooting at the two controllers. The settling time of PI Control System is smaller but Predictive PI Controller has 1degC less maximal deviation at the disturbance.
Since we have used two different controllers, such that PI Controller and Predictive PI Controller to control liquid temperature in a tank, we can conclude that as seen from Fig. 18 – Fig. 24 the closed loop system with Predictive PI Controller when the sample time is Δt=0.5s, noise filter's time constant is T_n=3s and the time constant of algebraic loop server is T_a=1s, gives one of the best performance – it settles fast without overshoot.

6 Conclusions

The PI controller controls the plant, using the error as the difference between the desired set output and the measured output and the integral of the error. The same input signals for the ANN are used both for the test of the predictor's accuracy and its training. The step size Δt=5s is also the same. The general behavior is estimated as good in the training points and the accuracy reached is high.

As a result the ANN behaves as an accurate predictor. The ANN plant predictor is good only when its inputs are within the range, for which it is trained. If some additional influences, not reflected in training, appear – the system with the ANN predictive controller may lose even stability.

So, the training data should include all cases of set-point changes and disturbances and should be rich in magnitudes and frequencies. It should be long and carefully collected, considering the real time peculiarities (measurement noise, ambient influences, disturbances, sample period impact). The ANN plant predictor should be tested for generalization with smaller Δt than the used in training, and in real time.

References:

[1] Smith. J. *Applications of neural networks*, New York, Prentice Hall, 2008.
[2] Applications of Artificial Neural Networks, 20 March 2007 http://www.gm.fh-koeln.de/~west/ANN_app.htm
[3]Peziylo Alexander, *Multi-Layer Neural Networks and Learning Algorithms*, in German, PDF, 22 December 2003.
[4] Nelson Marlilyn McCord, W.T. Illingworth, *Neural Nets-A Practical Guide*, Texas Instruments, Adison Wesley Publishing Company, Inc., ISBN:0-201-56309-6, August 1990.
[5] National Instruments, PCI 6014 product data sheet, http://ni.com
[6] T. Neshkov, S.Yordanova, I. Topalova, *Process Control and Production Automation*, Technical University – Sofia, 2005.
[7] Wasserman P., *Neural Computing – Theory and Practice*, ANZA Research, Inc., Van Nostrand Reihold, New York ISBN 0-442-207433.
[8] Suykens, J.A.K., Vandewalle, J.P.L., and De Moor, B.L.R., *Artificial Neural Networks for Modelling and Control of Non-Linear Systems*, New York, Kluwer Academic, 1996.
[9] Fausett, L., *Fundamentals of Neural Networks*, Prentice-Hall, Englewood Cliffs, NJ, 1994, p.461, ISBN 0-13-334186-0
[10] Haykin, S.,*Neural Networks: A Comprehensive Foundation*, Prentice Hall, Upper Saddle River, NJ, 1999.
[11] Yao, X., *A review of evolutionary artificial neural networks*, International Journal of Intelligent Systems, 1995, 4-203-222.