PIDNN FOR MARINE DIESEL MAIN ENGINE SPEED CONTROL

PID neuronski regulator za upravljanje brzinom vrtnje brodskoga propulzijskog dizelskog motora

Petar Matić, dipl. ing.
Pomorski fakultet Sveučilišta u Splitu
E-mail: pmatic@pfst.hr

dr. sc. Nikola Račić
Pomorski fakultet Sveučilišta u Splitu
E-mail: nikola.racic@pfst.hr

dr. sc. Danko Kezić
Pomorski fakultet Sveučilišta u Splitu
E-mail: danko.kezic@pfst.hr

UDK 629.5.03 : 621.316

Abstract

This work deals with Marine Diesel Main Engine speed controller which uses ANN to optimize PID controller parameters thus obtaining better transient characteristics and better control properties. To validate this claim, PID Neural Network (PIDNN) controller model was designed using Matlab/Simulink, and implemented in numerical model of the ship propulsion diesel engine [5]. PIDNN controller uses one of the most common ANN structures (multi-layer perceptron) and training algorithm (back-propagation), also briefly described in this work. The effectiveness of the PIDNN controller is shown through simulation and experiment.

Key words: Artificial Neural Networks (ANN), PID Neural Network (PIDNN), back-propagation, speed control, Marine Diesel Main Engine, Matlab/Simulink.

Sažetak

Ovaj rad opisuje postupak u projektiranju neuronskog regulatora za upravljanje brzinom vrtnje brodskoga propulzijskog dizelskog motora. Umjetnom neuronskom mrežom se koristi kako bi se odredili optimalni parametri regulatora i postigla bolja upravljačka svojstva. Teza je dokazana simulacijski, u Matlabu. Izrađen je simulacijski model PID neuronskog regulatora i implementiran je numerički model brodskoga propulzijskog dizelskog motora [5], a učinkovitost PID neuronskog regulatora prikazana je grafički. PID neuronski regulator koristi se jednom od osnovnih neuronskih struktura, višeslojnim perceptorom, i jednim od najčešćih algoritama za učenje, algoritmom propagacije pogreške unatrag, također ukratko opisanima u ovom radu.

Ključne riječi: umjetne neuronske mreže (UNM), PID neuronski regulator, propagacija pogreške unatrag, regulacija brzine vrtnje, brodski propulzijski dizelski motor, Matlab/Simulink.

INTRODUCTION / *Uvod*

The power and usefulness of artificial neural networks have been demonstrated in several applications including speech synthesis, diagnostic problems, medicine, business and finance, robotic control, signal processing, computer vision and many other problems [6]. The purpose of this work is to show ANN can be used for process control such as Marin Diesel Main Engine speed control. Although satisfying results in

ISSN 0469-6255 (193-201)

speed control are already obtained via traditional PI and PID controllers, it will be shown that significant improvement can be made if using neural network to adjust the controller parameters. Furthermore, Marine Diesel Engine for ship's propulsion is a highly complex process that is not easy to describe mathematically, which makes reliable controller design very difficult. Because of the learning property, neural network can be used to adjust randomly selected controller parameters in order to reach optimal performance. This should find its purpose especially in the phase of construction and controller development.

PID NEURAL NETWORK – PIDNN / *PID neuronski regulator*

Neural networks have been applied very successfully in the identification and control of dynamic systems. The universal approximation capabilities of the multilayer perceptron have made it a popular choice for modeling nonlinear systems and for implementing general-purpose nonlinear controllers [2]. ANN can be used for control of various systems because the network can gain experience from training process and in this way adapt to the controlled system. PIDNN is a neural network witch combines advantages of PID control algorithm with neural network structure to control a system that can not be easily described via transfer function or suitable mathematical equations.

PID control algorithm / *PID upravljački algoritam*

Most controllers use PID control algorithm. It combines advantages of three correcting terms: proportional, integral and derivate, for the purpose of correcting the error e(t). The sum of these correcting terms represents the output of PID controller u(t), expressed with (1):

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(t)dt + K_d \cdot \frac{de(t)}{dt} \qquad (1)$$

where:

K_P = proportional gain,
$K_I = K_P/T_I$ = gain of the integral term,
T_I = integrate time constant,
$K_D = K_P \cdot T_D$ = gain of the derivative term,
T_D = derivate time constant.

An actuators physical constraints (valves have a limited operating range 0%-100%, pumps have limited power, motors have a maximum moment) can be regarded as non-linearity in the process and have to be considered in the application of the controller. Since many of these limitations appear at the input of the actuator (process), they are referred to as input limitations and are modeled with a non-linear element having a saturation characteristic. Furthermore, the integrator in a controller with integral action would produce an inaccurate and highly excessive value which would cause oscillation and slowing down of the transient response. In other words, the effect would be a large overshoot and a long settling time. This behavior is called the integrator windup. There are several anti-windup algorithms to avoid adverse effects of the integrator windup on the control system performance. The rate of anti-windup action is defined with the constant T_{AW} which can be explained as a time constant of this action. Åstrom and Hägglund (1995) calculated its value as follows: $T_{AW} = \sqrt{T_I T_D}$ [1]

The form of the PID controller, used in this work is given below:

$$U(s) = K_P \cdot \left[E(s) + \frac{1}{sT_I} \cdot E(s) - sT_D \cdot Y(s) \right] \qquad (2)$$

Regarding above mentioned the controller is described as in figure 1.

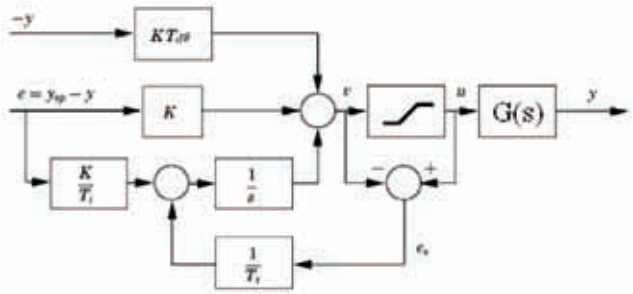

Figure 1. PID controller
Slika 1. PID regulator – blok-shema

Artificial Neural Network / *Umjetne neuronske mreže*

The branch of artificial intelligence called Artificial Neural Networks (ANN) dates back to the 1940s, when McCulloch and Pitts (1943) developed the first neural model. Inspired by the structure of the brain, the field has generated interest from researchers in such diverse areas as engineering, computer science, psychology, neuroscience, physics, and mathematics [6]. ANN is a network that consists of a group of neurons interconnected in such a manner to provide desired function, or solve given problem. Group of interconnected neurons form a system that resembles human brain, capable of acquiring experiential knowledge through process of learning and using it afterwards.

ISSN 0469-6255 (193-201)

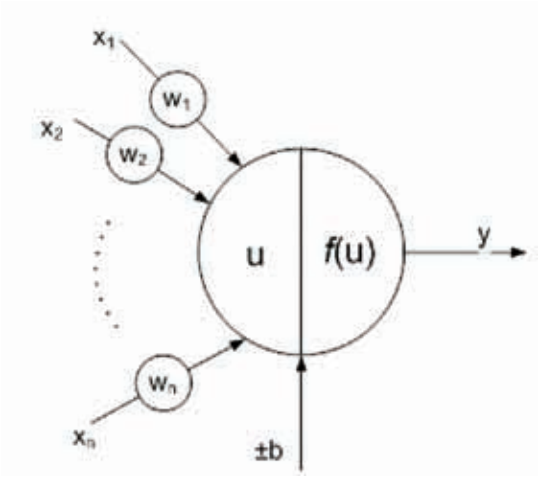

Figure 2. Artificial Neuron

Slika 2. Model umjetnog neurona

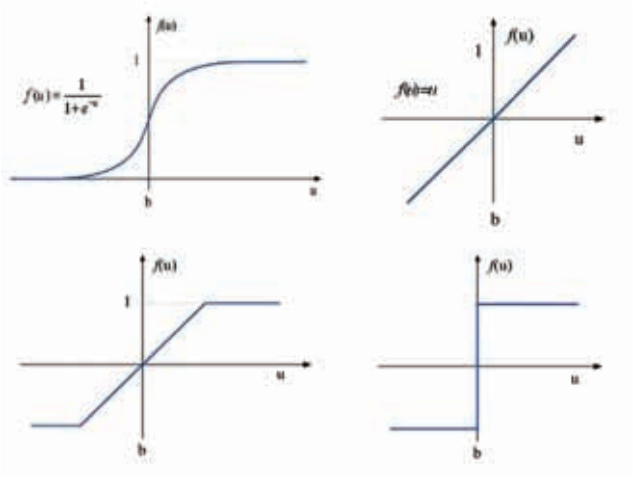

Figure 3. Commonly used activation functions

Slika 3. Uobičajeno korištene aktivacijske funkcije

The key element of that system, a neuron (figure 2), also called nod or a unit, performs input information processing (1) and calculates the output value (2).

$$u_j = \sum_{i=1}^{n} w_i \cdot x_i \tag{3}$$

$$y_j = f\left(\sum_{i=1}^{n} w_i \cdot x_i \pm b \right) \tag{4}$$

where:

u = weighted inputs

x_i = set of input values

wi = synaptic weight

y_j = output value of the j-th neuron

f = activation function (figure 3)

b = threshold

ANN is a structure formed from large number of interconnected neurons. Multi-layer perceptron[1], which is most commonly used structure (figure 4), consists of three layers of neurons: input, hidden and output layer. Input and output layers are in charge of data exchange with the environment, while the hidden layer 'does the thinking'. The outputs of input layer neurons are connected to next (hidden) layer inputs, and hidden layer outputs are connected to output layer inputs. The 'strength of connections' between neurons is defined by weight coefficients (synaptic weights) and is adjusted during the learning process to attain an objective and solve given problem. Knowledge that neural network can obtain through training process (and thus gain experience) is stored in synaptic weight values.

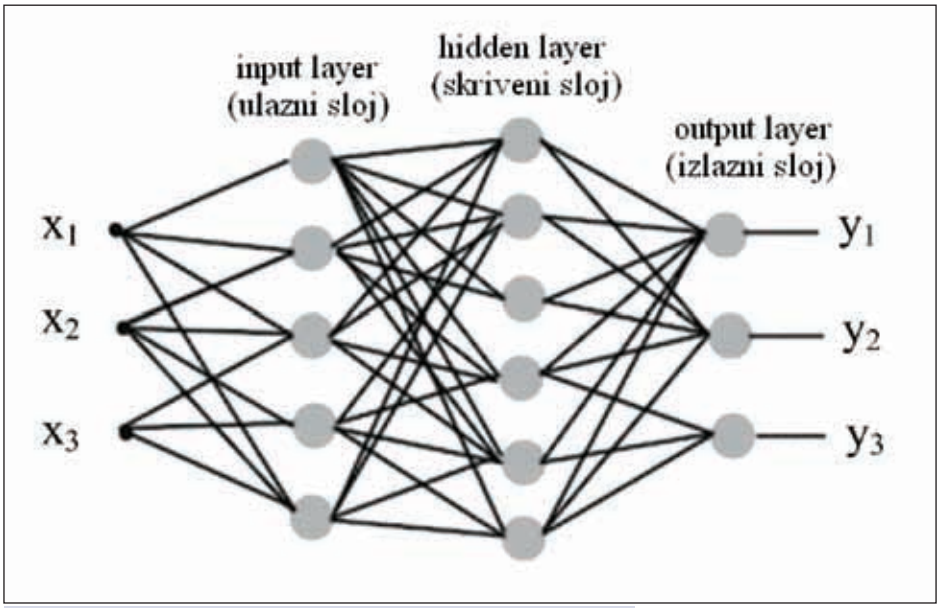

Figure 4. Multi-layer perceptron

Slika 4. Višeslojni perceptron

[1] Perceptron is first neural network with learning ability developed by F. Rosenblatt (1958.) and described in 'Principles of Neurodynamics', Washington D.C., Spartan Press, 1961. It was followed by Multi-Layer-Perceptron first introduced by M. Minsky and S. Papert in 1969. [3]

ISSN 0469-6255 (193-201)

According to Haykin (1994), a neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1. Knowledge is acquired by the network through a learning process.
2. Interneuron connection strengths known as synaptic weights are used to store the knowledge.[4]

Although defined by its structure (number of neurons and layers), ANN takes a final form only after learning process is over. So, in order to make ANN useful it has to be trained. Most popular training method, also used in this work, is Back-propagation[2] (back-propagation algorithm).

Back-propagation learning algorithm / *Algoritam za učenje propagacijom pogreške unatrag* [4]

In the case of multi-layer perceptron that uses back-propagation learning algorithm as a training method, signal travels through the network in two ways. One way is a forward propagation which starts at input layer. Signal travels forward to the output and network computes the output signal according to the expressions (3) and (4), while synaptic weights stay intact.

$$J = E_{av} = \frac{1}{2}\sum_{j=1}^{m} e_j^2(n) \tag{6}$$

Necessary adjustment of the synaptic weight between *i*-th and *j*-th neuron, during *n*-th iteration, can be calculated from the expression:

$$\Delta w_{ji}(n) = -\eta \cdot \frac{\partial J(n)}{\partial w_{ji}(n)} \tag{7}$$

where factor η defines synaptic weight adjustment speed i.e. learning speed, and partial derivation of coast function is:

$$\frac{\partial J(n)}{\partial w_{ji}(n)} = \frac{\partial J(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial u_j(n)} \cdot \frac{\partial u_j(n)}{\partial w_{ji}(n)} \tag{8}$$

Based on the expressions (3), (4), (5), (6), (7) and (8), follows the expression known as delta rule:

$$\Delta w_{ji}(n) = \eta \cdot \delta_j(n) \cdot y_i(n) \tag{9}$$

where $\delta_j(n)$ stands for local gradient:

$$\delta_j(n) = e_j(n) \cdot f_j'\left(u_j(n)\right) \tag{10}$$

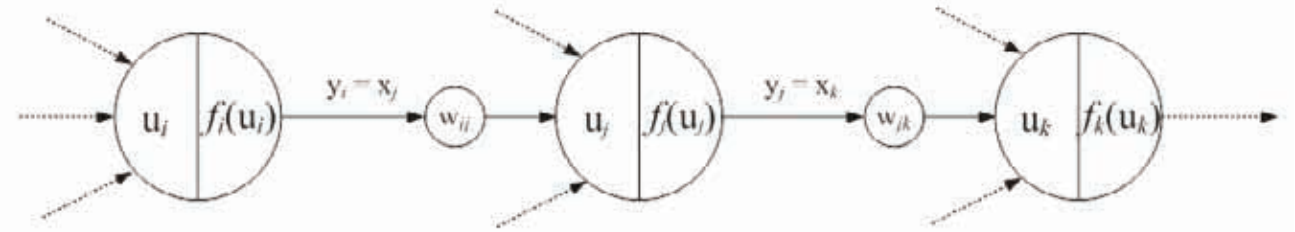

Figure 5. Forward-propagation of the function signal
Slika 5. *Rasprostiranje signala kroz mrežu unaprijed*

Back-propagation starts at output layer by computing the error signal for every neuron of the output layer:

$$e_j(n) = d_j(n) - y_j(n) \tag{5}$$

where:

d_j = desired value of the *j*-th neuron

y_j = actual output value of the *j*-th neuron

Learning process (training) mission is to adapt synaptic weights in order to gain minimum coast function (Average Square Error):

This local gradient would be the key element representing a neuron during back-propagation of the error signal and calculating the synaptic weight adjustments.

Synaptic weight value in next iteration can bi calculated from:

$$w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n) \tag{11}$$

If the *j*-th neuron belongs to an output layer, it is easy to calculate the local gradient from (10). However, if the local gradient of a hidden layer neuron needs to be determined, it is not possible to use expression (10), because the error of this neuron is impossible to define since its desired output value is unknown.

[2] Neural network that uses Back-propagation was first introduced by G.E. Hinton, E. Rumelhart and R.J. Williams in 1986 and is one of the most powerful neural net types. It has the same structure as the Multi-Layer-Perceptron and uses the back-propagation learning algorithm, used by neural nets with supervised learning, in fact special form of the delta learning rule [3].

ISSN 0469-6255 (193-201)

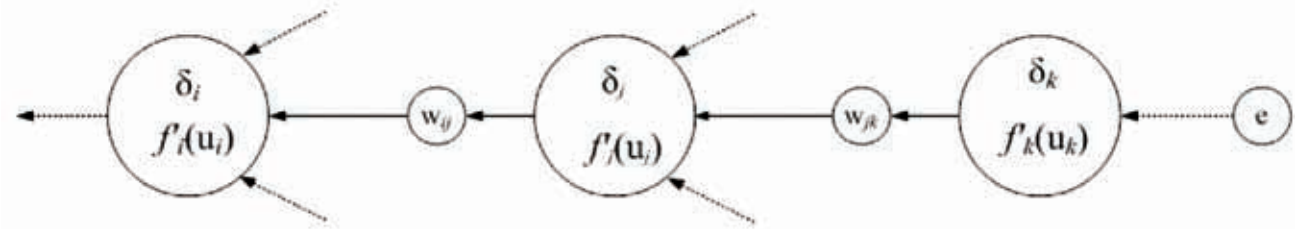

Figure 6. Back-propagation of the error signal

Slika 6. Rasprostiranje signala kroz mrežu unatrag

Solution of this problem is the benefit of the back-propagation method where calculated error signal travels backwards and in that way reduces the unknown variables. Local gradient of the hidden layer can be calculated from:

$$\delta_j(n) = f_j'\left(u_j(n)\right)\sum_k w_{kj}(n)\cdot\delta_k(n) \tag{12}$$

$$\delta_k(n) = e_k(n)\cdot f_k'\left(u_k(n)\right) \tag{13}$$

Design of PID controller using ANN / *Projektiranje PID neuronskog regulatora*

ANN designed for process control based on PID control algorithm (figure 7) is named PID Neural Network (PIDNN) controller. It has a multi-layer perceptron structure. This means it consists from a three layers of neurons: input, hidden and output layer. Input layer presents two P-neurons that are assigned of receiving two input signals: desired (r) and actual (y) value of the controlled parameter. Hidden layer consists of three neurons: P-neuron, I-neuron and D-neuron. These neurons are supposed to ensure three correcting terms of the controlled parameter error: proportional, integral and derivate [7], [8].

The output values of the P-neurons are calculated from:

$$x_j(n) = \begin{cases} a\ , & u_j(n) > a \\ u_j(n)\ , & -a \le u_j(n) \le a \\ -a\ , & u_j(n) < -a \end{cases} \tag{14}$$

The output value of the I-neuron is calculated from:

$$x_j(n) = \begin{cases} a\ , & x_j(n) > a \\ x_j(n-1) + u_j(n), & x_j(n) \in [-a, a] \\ -a\ , & x_j(n) < -a \end{cases} \tag{15}$$

The output value of the D-neuron is calculated from:

$$x_j(n) = \begin{cases} a\ , & x_j(n) > a \\ u_j(n) - u_j(n-1), & x_j(n) \in [-a, a] \\ -a\ , & x_j(n) < -a \end{cases} \tag{16}$$

From these expressions (14), (15), (16) it is obvious that P-neuron has a saturation transient characteristic, while I-neuron besides that has an integral function and

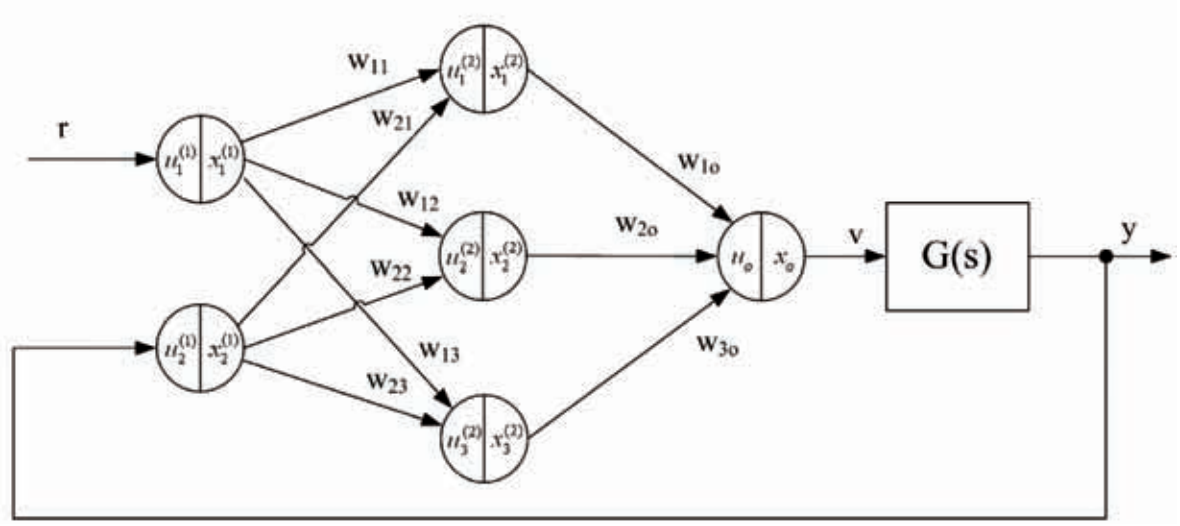

Figure 7. PIDNN controller in closed loop control system

Slika 7. PID neuronski regulator u regulacijskoj petlji

ISSN 0469-6255 (193-201)

D-neuron has derivate function. Parameter '*a*' depends on characteristics of the controlled process. The best way to determine its value is through an experiment.

To satisfy PID control algorithm (2), initial values of the synaptic weights are:

$$w_{11} = w_{12} = 1,\ w_{13} = 0 \tag{17}$$

$$w_{21} = w_{22} = w_{23} = -1 \tag{18}$$

Every neuron of the hidden layer has input signal:

$$u_1^{(2)}(n) = w_{11} \cdot x_1^{(2)}(n) + w_{21} \cdot x_2^{(2)}(n) = r(n) - y(n) \tag{19}$$

$$u_2^{(2)}(n) = w_{12} \cdot x_1^{(1)}(n) + w_{22} \cdot x_2^{(1)}(n) = r(n) - y(n) \tag{20}$$

$$u_3^{(2)}(n) = w_{23} \cdot x_2^{(1)}(n) = -y(n) \tag{21}$$

The output layer is presented by a single P-neuron which sums these three correcting terms into a unique control signal according to the PID control algorithm (1):

$$u_o(n) = w_{1o} \cdot x_1^{(2)}(n) + w_{2o} \cdot x_2^{(2)}(n) + w_{3o} \cdot x_3^{(2)}(n) \tag{22}$$

This means that synaptic weights between hidden and output layer are actually PID controller parameters that need to be set during a training process ($w_{1o} = K_P$, $w_{2o} = K_I$, $w_{3o} = K_D$).

According to the back-propagation learning algorithm (earlier explained in section 2.2.1.), synaptic weight values from hidden to output layer, in next iteration, can be calculated from:

$$w_{jo}(n+1) = w_{jo}(n) + \Delta w_{jo}(n) = w_{jo}(n) + \eta_{j1} \cdot \delta_o(n) \cdot x_j^{(2)}(n) \tag{23}$$

where:

$$\delta_o(n) = \left[r(n) - y(n)\right] \cdot f'\left(v(n)\right) \tag{24}$$

Since transfer function of the highly complex controlled process is unknown, the element *f'(v(n))* of the expression (24) is not possible to determine. So, following approximation will be used in order to determine output local gradient:

$$f'\left(v(n)\right) \approx \frac{\Delta y}{\Delta x_o} = \frac{y(n) - y(n-1)}{x_o(n-1) - x_o(n-2)} \tag{25}$$

From the back-propagation of the error signal, it is possible to determine local gradient of the hidden layer:

$$\delta_j(n) = \delta_o(n) \cdot w_{jo}(n) \cdot f_j'\left(u_j(n)\right) \tag{26}$$

Again, we use the approximation:

$$f_j'\left(u_j(n)\right) \approx \frac{\Delta y_j}{\Delta x_j} = \frac{x_j(n) - x_j(n-1)}{u_j(n) - u_j(n-1)} \tag{27}$$

Adjustments and next iteration synaptic weight values between input and hidden layer are:

$$w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n) = w_{ij}(n) + \eta_{j2} \cdot \delta_j(n) \cdot x_i^{(1)}(n) \tag{28}$$

Simulation model of ship propulsion system / *Simulacijski model brodskoga propulzijskog sustava* [5]

The model is control volume type, which treats multi-cylinder ship propulsion turbocharged two-stroke marine diesel engine as a series of thermodynamic control volumes interconnected by connections for mass and energy transfer.

The computer-simulation model in this work was developed in program code MATLAB – SIMULINK. A model was made according to mathematical zero-dimensional model, described by the system of nonlinear differential equations, added with empirical and correlation equations which describe components of the system and boundary conditions. Using the time of simulation, the change of angle and real time has been determined, so that all calculated parameters can be followed in the time and angle.

The figure 8 presents the block scheme of model made in Simulink program code. Model is composed of eight basic elements, connected in one unique unit. Basic elements are: regulator, and fuel pump, cylinder, exhaust gas receiver, scavenging air receiver, turbine, compressor, turbocharger and engine dynamic. The model is used to simulate a ship propulsion system on one chemical tanker, consist of slow-speed two-stroke turbocharged diesel engine type MAN B&W 6S50MC and fixed pitch propeller. It is possible to set any desired two-stroke diesel engine propulsion system combination, using the same model. The necessary data for the set build up one in the form of input data file. Before start of calculations of system transient, it is necessary to match calculated and measured data for the system steady operation point. Adequate match between calculated and measured data is the basic precondition for the reliable calculation of engine transient.

ISSN 0469-6255 (193-201)

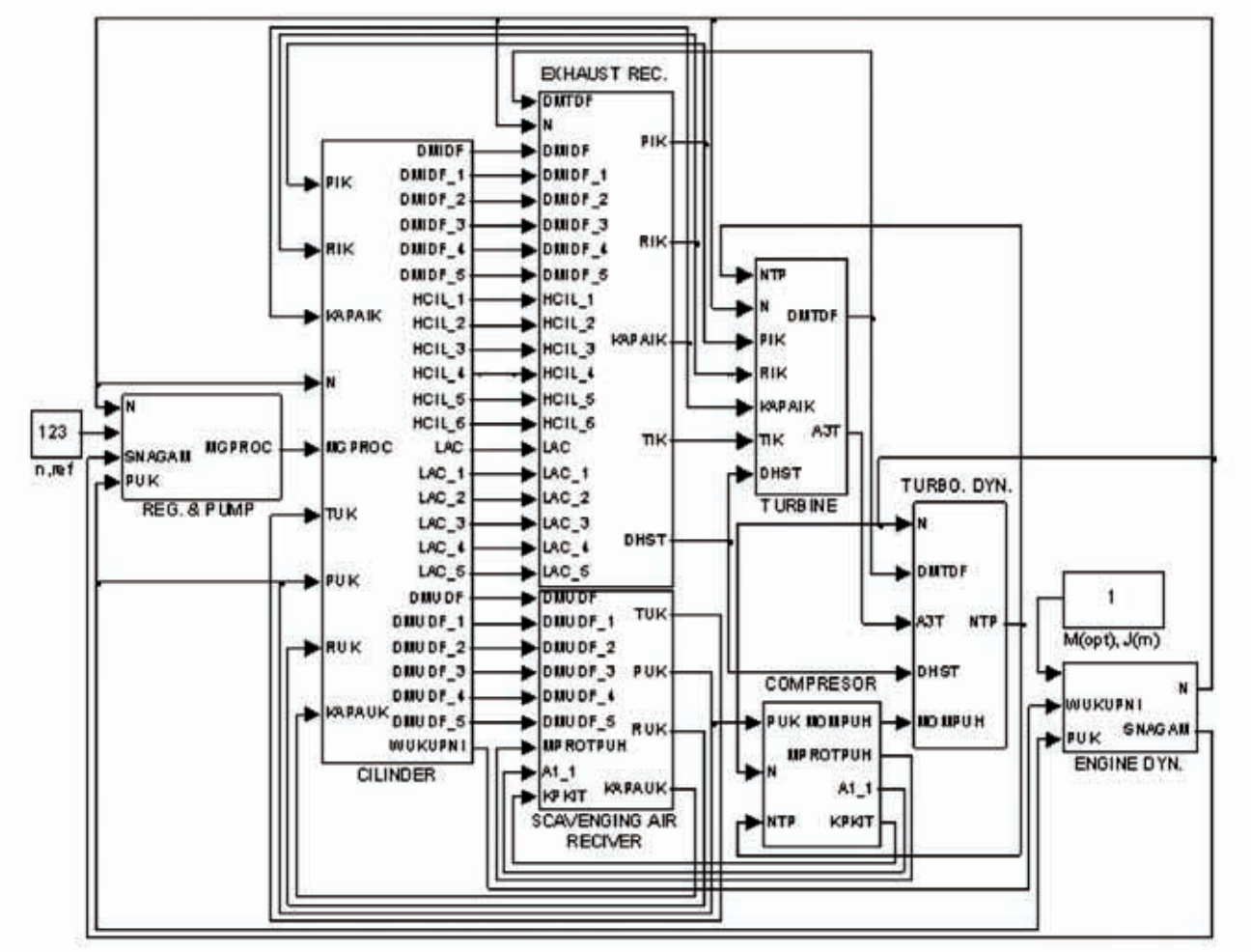

Figure 8. Simulation model of ship propulsion system in program code MATLAB –SIMULINK

Slika 8. Simulacijski model brodskoga propulzijskog sustava realiziran u Matlab/Simulink programskom jeziku

SIMULATION AND EXPERIMENT / *Simulacija i eksperiment*

Based on mathematical model, equations from (1) to (28), simulation model of the PIDNN controller with back-propagation training algorithm was built, using Matlab/Simulink. The model was implemented, trained and tested on Marine diesel main engine model, also designed in Matlab/Simulink, for the control of Main Engine (ME) speed.

Simulation and experiment – Training process / *Simulacija i eksperiment – process učenja*

For the training purpose, following parameters were used:

- Initial controller parameter values ($K_p = K_i = K_d = 1$) were selected randomly, but also having in mind not to make the system unstable.
- Learning speed values established thru an experiment: $\eta_{Kp} = -10^{-4}$, $\eta_{Ki} = -10^{-7}$, $\eta_{Kd} = -10^{-2}$
- initial speed setpoint: 121.4 rpm
- input signal, speed change signal: pulse (Amplitude = ±25,4 , Period = 50000)

Training process should be stopped after reaching satisfying results, in this case after approximately 65000 seconds (figure 9). System response improves during training, until some point which is a signal that it should be stopped. These improvements manifest as overshoot decrease, and oscillations termination.

ISSN 0469-6255 (193-201)

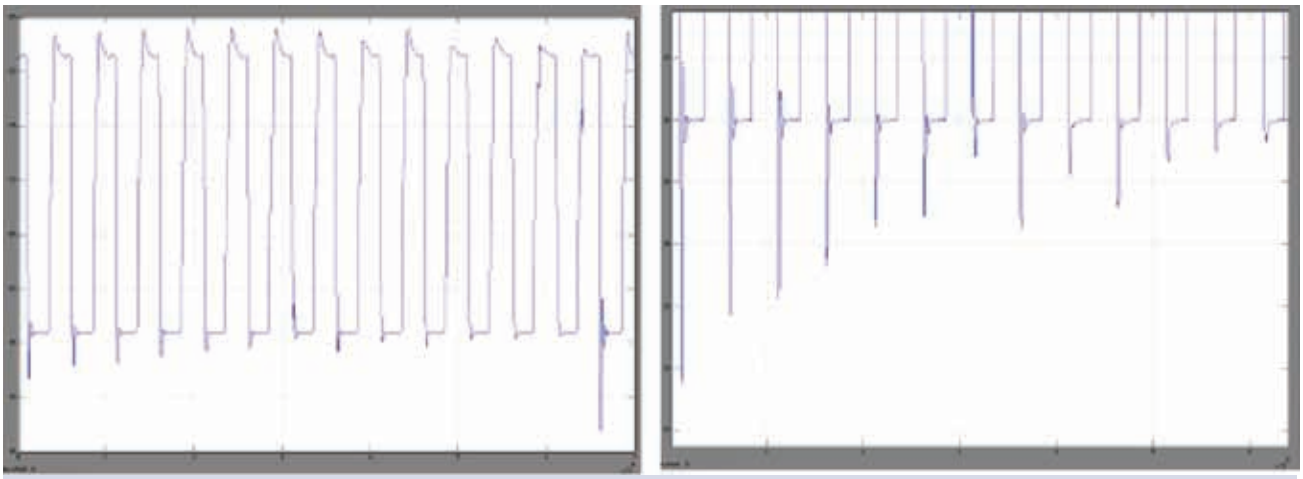

Figure 9. Simulation results of speed control - training process

Slika 9. Simulacijski rezultati regulacije brzine vrtnje za vrijeme procesa učenja/treniranja mreže)

During the training process back-propagation algorithm changes synaptic weights (controller parameters values) according to the delta rule, figure 10.

The results of the training process are controller parameters:

Kp = 0,86 ; Ki = 0,62 ; Kd = 1,01

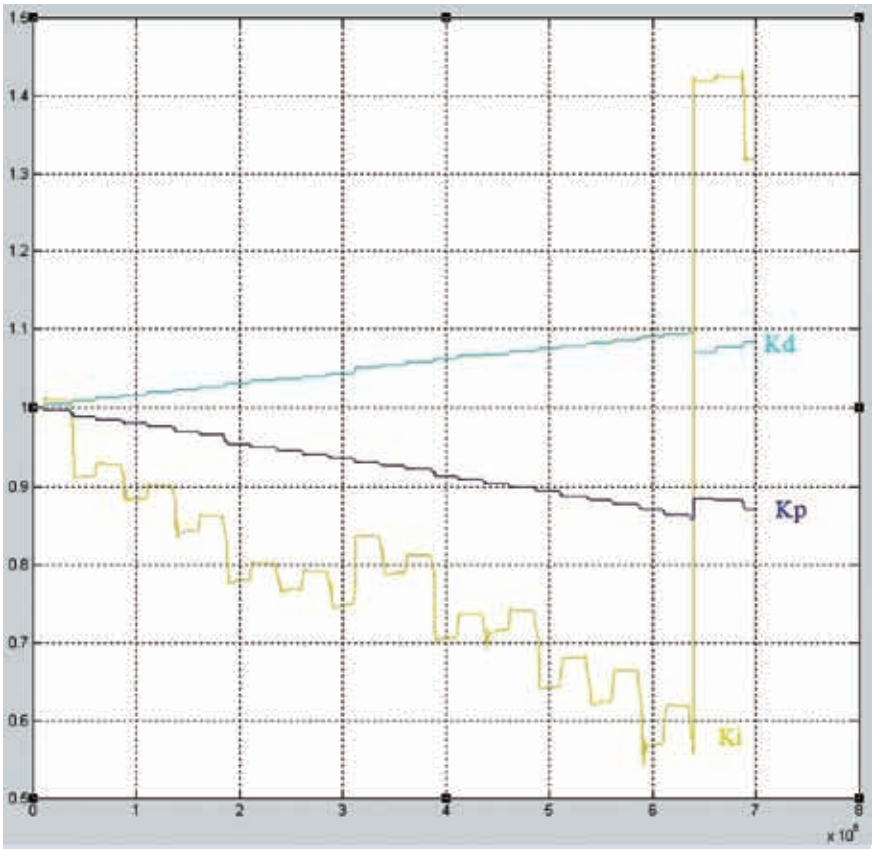

Figure 10. The change of the controller parameters during training

Slika 10. Promjena parametara regulatora za vrijeme procesa učenja/ treniranja mreže

To show that these values of the controller parameters provide good control ability thus prove that ANN can be used for the controller parameters optimization, it should be tested. Furthermore, during the testing process a comparison of the conventional PI and PIDNN controller will be made to show that it is possible to obtain significantly better transient characteristics and better control properties by using PIDNN for ship propulsion control.

Simulation and experiment – Testing process / *Simulacija i eksperiment – ispitivanje rada*

From the simulation results (figure 11) it is obvious that the process of speed control is carried out much better by the PIDNN controller because of the smaller oscillations, significantly smaller overshot and faster reach of speed set point.

ISSN 0469-6255 (193-201)

CONCLUSION / *Zaključak*

Marine diesel main engine for ship's propulsion is a highly complex system that is not easy to model, which makes reliable controller design rather difficult. This work shows that it is possible to find optimal controller parameters through simulation and experiment by using ANN. Based on the simulation results comparison between conventional PI and new PIDNN controller it is easy to make a conclusion about effectiveness of the proposed control method. It has been proven that it is possible to obtain better transient characteristics and better control properties by using PIDNN controller for the purpose of controlling Marine diesel main engine speed. Further improvements are yet to be made, especially in training phase, where optimal learning speed values as well as optimal set of input wave forms needs to be found experimentally. These simulation results, although very useful, are yet to be tested in real working environment before making final conclusion.

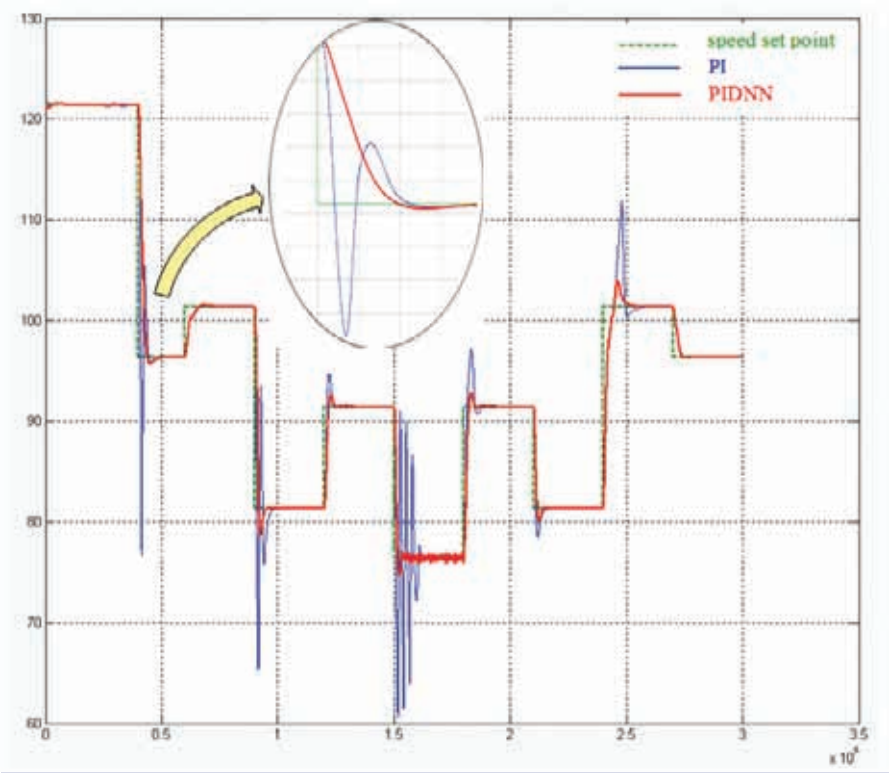

Figure 11. Simulation results comparison of speed control - the transient characteristics

Slika 11. Prikaz rezultata simulacije - usporedba odziva sustava vođenoga standardnim PI regulatorom i odziva sustava vođenoga PID neuronskom regulatorom

REFERENCES / *Literatura*

[1] K. J. Åstrom,. and T. Hägglund : *PID controllers: Theory, Design and Tuning*, Instrument Society of America, North Carolina, USA, 1995

[2] Martin T. Hagan, Howard B. Demuth: *Neural Networks for Control*, Proceedings of the 1999 American Control Conference, San Diego, CA, 1642-1656 (1999)

[3] Jochen Fröhlich: *Neural Net Components in an Object Oriented Class Structure,* Fachhochschule Regensburg, Department of Computer Science, 1996/'97

[4] S. Haykin: *Neural Networks: A Comprehensive Foundation*, NY: Macmillan, 1994.

[5] N. Racic: *Simulation of performance of the ship propulsion system with slow speed diesel engine in aggravated conditions*, Doctor Science thesis, University of Rijeka Faculty of engineering, Croatia, 2008

[6] Ingrid F. Russell: *Neural Networks*, Printed with permission from the Journal of Undergraduate Mathematics and its Applications, Vol 14, No 1

[7] Huailin Shu, Youguo Pi: *PID neural networks for time-delay systems*, 2000 Elsevier Science Ltd.

[8] Huailin Shu, Youguo Pi (2) : *Decoupled Temperature Control System Based on PID Neural Network*, ACSE 05 Conference, 19-21 December 2005, CICC, Cairo, Egypt

Rukopis primljen: 21. 7. 2009.

ISSN 0469-6255 (193-201)