PID – Neuron controller for ships heading with neuron identification

PHUNG HUNG NGUYEN [1], DUY ANH NGUYEN [2], HAI HONG VO [3]
[1]Faculty of Navigation, HCMC University of Transport, Ho Chi Minh City, Vietnam
[2]Faculty of Mechanical Engineering, HCMC University of Technology, Ho Chi Minh City, Vietnam
[3]Faculty of Navigation, Vietnam Maritime University, Hai Phong, Vietnam
Email: duyanhnguyen@hcmut.edu.vn, hungdktb@hotmail.com

Abstract: This paper presents new results of PID neural network controller for heading control of ship. The control system uses one neural network to drive parameters of a PID controller. Another neural network is used to predict the ship's heading so that it gives the predicted heading of k steps in advance. Output of the second neural network is sent to the input layer of the first one, that produces again the PID controller's parameters. We carry out computer-based simulations to see how the proposed control algorithm performs. It is shown by the simulation results that the controller can perform heading control task well employing the advantages of both neural networks and traditional PID.

Keywords: *Neural Network, PID, Heading Control System, Predictive*

Introduction:

Since 1990s theory and applications of artificial neural networks in automatic control has been proposed by many authors. The authors in [1], [2] introduced a neural network algorithm for controlling ship's heading and tracking. However, most of autopilot systems are traditional PID controllers. Some systems have been added with adaptive ability.

This paper introduces a PID controller, parameters of which are driven by a neural network. This combination of PID controller and neural network is called BPNN-PID controller in this paper. Another neural network is used for predicting ship's heading signal of some steps in advance to provide BPNN-PID controller predicted inputs.

The performance of the proposed BPNN-PID controller is then verified by computer simulations using MATLAB.

BPNN-PID Controller

Principal Configuraion

Configuration of the BPNN-PID controller based on back propagation neural network (BPNN) consists of two parts. 1) Traditional PID controller and 2) back propagation neural network. The control perfomance depends on setting of parameters Kp, Ki and Kd driven by BPNN. The BPNN adapts its weights using gradien decent method and ensures that the desired PID controller parameters are calculated.

In this research the combination of traditional PID controller and BPNN has shown the desired and stable control performance.

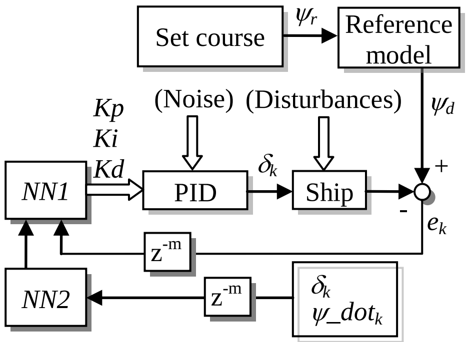

Figure 1: Configuration of the BPNN-PID heading control system with neuron identification.

Control Algorithm

PID controller

Control outputs (rudder angle) from PID controller is used as introduced in [5] as following:

$$\delta_{pid} = (Kp + Ki + Kd)e_k - (Kp + 2Kd)e_{k-1} + Kde_{k-2} \quad (1)$$

Where δ_{pid} is output of PID controller; Kp, Ki and Kd are propotional, integrative and derivative parameters respectively; $e(k)$ is input error determined as $e(k) = y(k) - r(k)$, with y is the actual output, r is the desired output of the system.

Training of NN

If NN have sufficient number of neuron, it can approximate any continuous function with only one hidden layer of neurons. Thus, we use NN with one input layer, one hidden layer and one output layer. This is BPNN configuration of which is shown in Figure 2.

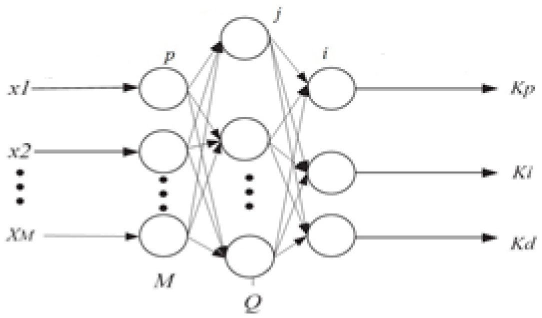

Figure 2: Configuration of the BPNN-PID heading control system with neuron identification (NN1)

a) Training of BPNN

Output of each input layer neuron is :

$$O_p = X_p \ \left(p = 1,2,3,...,M \right) \quad (2)$$

Where O_p is output of *p* neuron of input layer.

Input and output of neuron in hidden layer respectively are:

$$net_j\left(k\right) = \sum_{p=1}^{M} \omega_{jp} O_p \quad (3)$$

$$O_j\left(k\right) = f\left(net_j\left(k\right)\right) \left(j = 1,2,3,...,Q \right) \quad (4)$$

With net_j is input of *jth* neuron; ω_{jp} is weight, $f\left(x\right)$ is activation function in hidden layer; $f\left(x\right)$ is a sigmoidal function.

$$f\left(x\right) = \tanh\left(x\right) = \left(e^x - e^{-x}\right) \Big/ \left(e^x + e^{-x}\right) \quad (5)$$

Inputs and outputs of the output layer are:

$$net_i\left(k\right) = \sum_{j=1}^{Q} \omega_{ij} O_j \quad (6)$$

$$O_i\left(k\right) = g\left(net_i\left(k\right)\right) \left(i = 1,2,3\right) \quad (7)$$

$$\begin{cases} K_P\left(k\right) = O_1\left(k\right) \\ K_I\left(k\right) = O_2\left(k\right) \\ K_D\left(k\right) = O_3\left(k\right) \end{cases} \quad (8)$$

Where w_{ij} are weights of neurons in output layer; outputs of the output layer are *Kp*, *Ki* and *Kd*; *g(x)* is activation function of output neurons and this is a sigmoidal one.

$$g\left(x\right) = \frac{1}{2}.\left[1 + \tanh\left(x\right)\right] = e^x \Big/ \left(e^x + e^{-x}\right) \quad (9)$$

BPNN adjuds PID parameters automatically and reduces time for designing controller. Online training scheme is used for updating the NN weights. This method can make the controller effectively cope with uncertain and nonlinearities in ship model [1], [2].

b) Error backpropagation and weights updating

Cost function of the controller is:

$$E\left(k\right) = \frac{1}{2}\left(rin\left(k\right) - yout\left(k\right)\right)^2 \quad (10)$$

Where *rin* is desired output, *yout* is actual output. Generally, hidden and output layer weights are apdated as following:

$$\Delta\omega_{ij}\left(k\right) = -\eta \frac{\partial E\left(k\right)}{\partial \omega_{ij}} \quad (11)$$

A momentumn is added as:

$$\Delta\omega_{ij}\left(k\right) = -\eta \frac{\partial E\left(k\right)}{\partial \omega_{ij}} + \alpha \Delta\omega_{ij}\left(k-1\right) \quad (12)$$

Where η is learning rate, α is momentumn factor. Note that:

$$\frac{\partial E\left(k\right)}{\partial \omega_{ij}\left(k\right)} = \frac{\partial E\left(k\right)}{\partial y\left(k\right)} \cdot \frac{\partial y\left(k\right)}{\partial u\left(k\right)} \cdot \frac{\partial u\left(k\right)}{\partial O_i\left(k\right)} \cdot \frac{\partial O_i\left(k\right)}{\partial net_i\left(k\right)} \cdot \frac{\partial net_i\left(k\right)}{\partial \omega_{ij}\left(k\right)} \quad (13)$$

$$\frac{\partial net_i\left(k\right)}{\partial \omega_{ij}\left(k\right)} = O_i\left(k\right) \quad (14)$$

And based on (10) we can obtain:

$$\frac{\partial u\left(k\right)}{\partial O_1\left(k\right)} = e_y\left(k\right) - e_y\left(k-1\right) \quad (15)$$

$$\frac{\partial u\left(k\right)}{\partial O_2\left(k\right)} = e_y\left(k\right) \quad (16)$$

$$\frac{\partial u\left(k\right)}{\partial O_3\left(k\right)} = e_y\left(k\right) - 2e_y\left(k-1\right) + e_y\left(k-2\right) \quad (17)$$

Then updating algorithm of output weights is:

$$\omega_{ij}\left(k+1\right) = \omega_{ij}\left(k\right) + \Delta\omega_{ij}\left(k\right) \quad (18)$$

$$\Delta\omega_{ij}\left(k\right) = \alpha\Delta\omega_{ij}\left(k-1\right) + \eta\delta_i O_j\left(k\right) \quad (19)$$

Where, δ_i is hidden error function and it is used for updating weights between input and hidden layers. δ_i is shown as:

$$\delta_i = e_y\left(k\right). \frac{\partial y\left(k\right)}{\partial u\left(k\right)} . \frac{\partial u\left(k\right)}{\partial O_i\left(k\right)} . \dot{g}\left(net_i\left(k\right)\right) \quad (20)$$

Where the derivative of $g\left(x\right)$ is:

$$\dot{g}\left(x\right) = g\left(x\right)\left(1 - g\left(x\right)\right) \quad (21)$$

Next, using similar method with error in hidden layer δ_j we can get:

$$\omega_{jp}\left(k+1\right) = \omega_{jp}\left(k\right) + \Delta\omega_{jp}\left(k\right) \quad (22)$$

$$\Delta\omega_{jp}\left(k\right) = \alpha\Delta\omega_{jp}\left(k-1\right) + \eta\delta_j O_p\left(k\right) \quad (23)$$

$$\delta_j = \dot{f}\left(net_j\left(k\right)\right). \sum_{i=1}^{3} \delta_i \omega_{ij}\left(k\right) \quad (24)$$

The derivative of $f\left(x\right)$ is:

$$\dot{f}\left(x\right) = \frac{\left(1 - f^2\left(x\right)\right)}{2} \quad (25)$$

In this paper, the “intensive” learning method as shown in [1] is used to improve online learning ability of the NNs.

NN ship model for identification

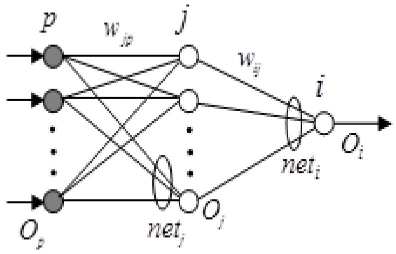

Figure 3: Configuration of the NN2 for identifying ship’s rate of turn.

BPNN using for predicting ship’s rate of turn (ψ_dot_k) is also a three layers NN. Input layer neurons are rate of turn and rudder angle signals in *k-1, k-2, k-3* steps. The predicted heading of ship is then obtained from the identified rate of turn and sent to input layer of NN1.

Simulation results

Simulation Setup

Ship Model

The mathematical ship model is used for simulating and testing the performance of the controller in this paper. The simulations are carried out for a Mariner Class Vessel, the nonlinear model of which can be found in GNC Toolbox for MATLAB [5]. The planar motion mechanism tests and full-scale steering and maneuvering predictions for this Mariner Class Vessel were performed by the hydro- aerodynamics laboratory in Lyngby, Denmark.

Controller Parameters

The NN1 is a multi-layer feed-forward neural network with three layers (Figure2). Number of neuron in input, hidden, and output layers is 6, 9, and 3 respectively. Input layer includes linear activation function neurons, hidden layer includes sigmoidal activation function neurons, and output neuron is a tangent sigmoidal activation function one. The NN2 is a similar neural network (Figure3) with number of neuron in input, hidden, and output layers is 6, 9, and 1 respectively.

Heading Control Simulation Results

The initial heading is 000° and desired heading is 025°. Nominal ship speed is 15 knots (or 7.7175 m/s).

Firstly, we simulate heading control of ship using BPNN-PID controller without influence of wind. Secondly, wind effect is added.

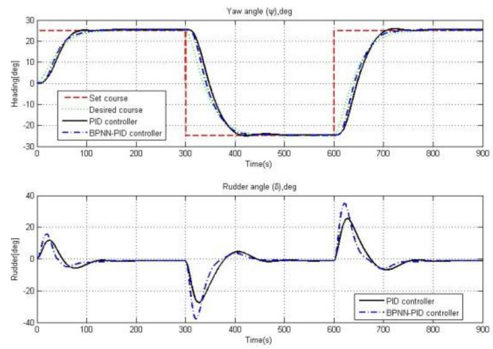

Figure 4.1a: Ship heading and rudder without wind

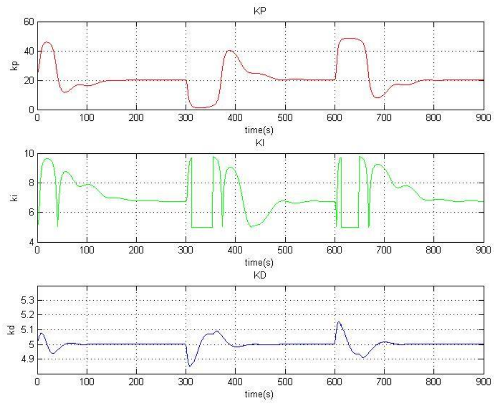

Figure 4.1b: PID parameters without wind

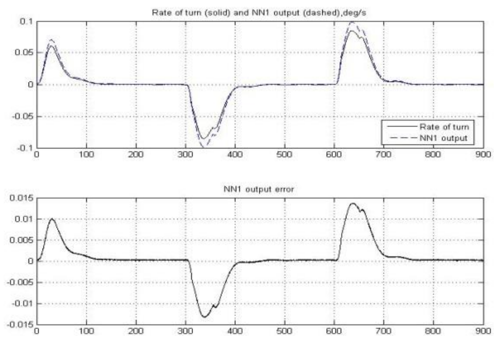

Figure 4.1c: NN2 output without wind

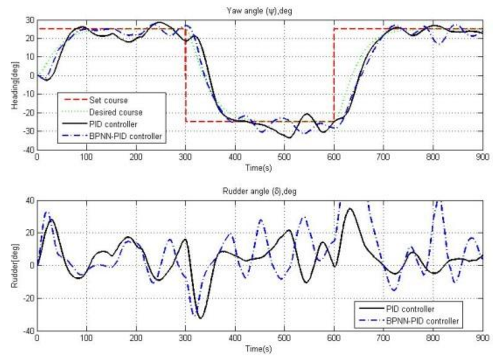

Figure 5.1a: Ship heading and rudder in wind

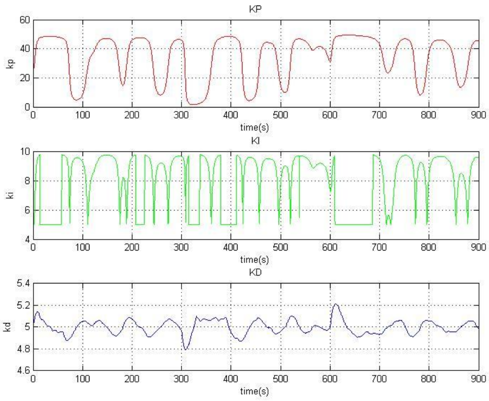

Figure 5.1b: PID parameters in wind

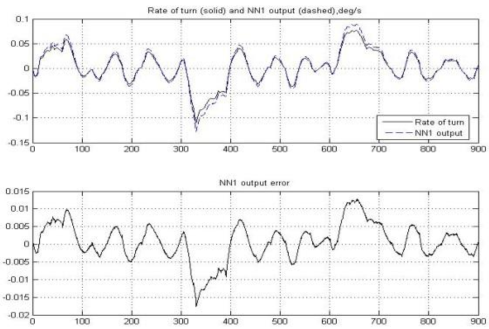

Figure 5.1c: NN2 output in wind

In the simulations, the perfomance of BPNN-PID controller is more active than that of PID controller. This is because BPNN-PID controller parameters are adjusted adaptively with the aids from NN1 and NN2.

Conclusion

In this paper we introduce a PID controller, parameters of which are driven by a neural network. Another neural network is introduced for predicting ship's heading signal of some steps in advance to provide BPNN-PID controller predicted inputs.

The performance of the proposed BPNN-PID controller is then verified by computer simulations using MATLAB. The BPNN-PID controller is more active than PID controller because its parameters are adjusted adaptively with the aids of NN1 and NN2.

In future studies we intend to investigate more about the role of the above NN1, and NN2 in combining with traditional PID controller. The purpose of those studies is to find the method of tuning the PID controller adaptively employing the abilities of NNs.

References:

[1] P.H. Nguyen and Y.C. Jung, "An adaptive autopilot for course-keeping and track-keeping control of ships using adaptive neural network (Part I: Theoretical Study)", International Journal of Navigation and Port Research (KINPR), Vol.29, No.9 pp. 771~776, ISSN-1598-5725, 2005.

[2] P.H. Nguyen and Y.C. Jung (2006a), "An adaptive autopilot for course-keeping and track-keeping control of ships using adaptive neural network (Part II: Simulation Study)", International Journal of Navigation and Port Research (KINPR), Vol.30, No.2 pp. 119~124, ISSN-1598-5725, 2006.

[3] P.H. Nguyen, "A study on the automatic ship control based on adaptive neural networks", PhD thesis, Graduate school of Korea Maritime University, 2007.

[4] Phung-Hung Nguyen, Hong-Hai Vo, Duy-Anh Nguyen, "PID-Neural networks controller for ship autopilot system", Proceedings of 2015 Conference on Transport, HCM city, 2015/May.

[5] Fossen, T.I. (2002). Marine Control Systems: Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles. Marine Cybernetics, Trondheim, Norway. ISBN 82-92356-00-2.