IEEJ TRANSACTIONS ON ELECTRICAL AND ELECTRONIC ENGINEERING
IEEJ Trans 2014; **9**: 384–390
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI:10.1002/tee.21983

Paper

Self-Tuning PID Control of a Brushless DC Motor by Adaptive Interaction

Tayfun Gundogdu[a], Non-member
Guven Komurgoz, Non-member

In this paper, a self-tuning algorithm for proportional integral derivative (PID) control based on the adaptive interaction (AI) approach theory efficiently used in artificial neural networks (ANNs) is proposed. In this approach, a system is decomposed into interconnected subsystems, and adaptation occurs in the interaction weights among these subsystems. The principle behind the adaptation algorithm is mathematically equivalent to a gradient descent algorithm. The same adaptation as the well-known backpropagation algorithm (BPA) can be achieved without the need of a feedback network, which would propagate the errors, by applying adaptive interaction. Thereby, the ANN controller can be adapted directly without wasting calculation time in order to increase the frequency response of the controller. The velocity control of a brushless DC motor (BLDCM) under slowly and rapidly changing load conditions is simulated to demonstrate the effectiveness of the algorithm. The AI tuning algorithm was used to tune up the PID gains, and the simulation results with PID adaptation process are presented by comparing the obtained results with the adaptive PID controller based on BPNN and a conventional PID controller. © 2014 Institute of Electrical Engineers of Japan. Published by John Wiley & Sons, Inc.

Keywords: adaptive interaction, adaptive control, adaptive neural network, self-tuning, proportional integral derivative (PID) controller, brushless DC (BLDC) motor control

Received 20 March 2013; Revised 26 July 2013

1. Introduction

Artificial neural networks (ANNs) have found applications in many engineering fields, especially in control engineering, such as linear or nonlinear system identification [1–3] and also linear or nonlinear adaptive control [4–10]. Today, proportional integral derivative (PID) control still remains the most popular type of control employed in industries because of its simplicity, robustness, and widespread applicability. PID controllers are classified into two categories according to their tuning algorithm: initial "offline" tuning, and continuous "online" self-tuning. The second class will be the focus of this paper, because the objectives and hence requirements of the PID controller often change during the different stages of control and the plant to be controlled often changes from time to time.

For the brushless DC motor (BLDCM) control, the PID controller is one of the most popular methods, since its design is well known for simple systems and does not require detailed knowledge of the system dynamics [3]. However, when using PID structures, it can be difficult to determine the appropriate PID gains when various uncertainties and nonlinearities exist, such as payload variations, hysteresis, friction, and others. All these factors could degrade the control system performance. Several tuning methods and discussions have been given in Ref. [4]. In addition, numerous research papers that have focused on adaptive PID control [5], self-tuning PID control [6,7], self-tuning predictive PID control [8], and so on, can be found in the existing literature. In adaptive and self-tuning PID, the controller parameters were tuned in accordance with the changes of the process parameters automatically [5,7].

ANNs are used to modify the gains of PID controllers [9,10]. To adapt the ANNs, a learning algorithm with two essential categories, namely supervised learning and unsupervised learning, should be used [11]. For unsupervised learning applied to feed-forward networks, there is the learning matrix and counterpropagation. For the supervised learning applied to feed-forward networks, there are backpropagation, time delay NNs, and perceptions. However, the backpropagation algorithm (BPA), which is commonly used in adaptive control systems, cannot be applied directly to NN controllers. Because of BPA's reliance on a dedicated feedback network to propagate the error back, the system must consist of "pure" neurons. Nevertheless, instead of BPA, the adaptive interaction (AI) algorithm (AIA) can be used to the same effect as the BPA [12,13]. Additionally, AIA is mathematically equivalent to the BPA. An ANN controller (ANNC) can be adapted directly without approximating the plant by using the AIA. This not only eliminates the error in approximation but also significantly reduces the complexity of the design.

Using this theory, the BLDCM controlled by self-tuning PID is decomposed into four subsystems consisting of BLDCM and the proportional (P), integral (I), and derivative (D) controller. The parameters of the PID controller K_P, K_I, and K_D are viewed as the interactions between these four subsystems. A simple and effective adaptation algorithm developed in the theory of AI is applied to self-tune these coefficients. To be able to apply this self-tuning algorithm, the only information required about the plant is its Fréchet derivative, which can be easily replaced by a constant that is absorbed into the adaptation coefficient for linear systems [12].

In this paper, velocity control of a BLDCM under a rapidly changing load condition is analyzed by applying this AIA to tune up the PID gains online. The reference speed, actual speed, error, and PID gain curves for the speed control are presented after the simulation by using MATLAB Simulink®. On the other hand, in order to demonstrate the accuracy of the proposed method, the obtained results from the AIA-based PID (AIPID) controller have been compared with those of a conventional PID (CPID) and a neuron-based adaptive PID (ANNPID) controller that uses the BPA. There are many different studies on the self-tuning PID algorithms and control of a BLDCM in the existing literature. But, the modified AIA, which works very well in the linear systems and gives excellent results, has not been used in any of

[a] Correspondence to: Tayfun Gundogdu. E-mail: tgundogdu@itu.edu.tr

Department of Electrical Engineering, Istanbul Technical University, Faculty of Electrical-Electronics Engineering, 34469-Maslak/Istanbul, Turkey

the previous studies to tune up the PID gains, as is done in this paper.

2. Model of BLDCM

DC motors have been widely used in engineering because of their simple structure and ability to be easily integrated into control systems with relatively low cost. The traditional model of a BLDCM is a second-order linear one.

In this paper, a DC motor is controlled via the input voltage. The control design and theory for controlling a BLDCM via current is nearly the same. For simplicity, a constant value as a reference signal is injected to the system to obtain a desired position. The transfer function from the input voltage $V_a(s)$ to the angular velocity ω is

$$\frac{\omega(s)}{V_a(s)} = \frac{K_t - (R_a + L_a s)T_L}{(R_a + L_a s)(Js + B) + K_t[K_e - (R_a + L_a s)T_L]} \quad (1)$$

where R_a is the armature resistance, L_a is the armature inductance, J is the moment of inertial of the motor rotor and load, B is the damping ratio of the mechanical system, K_t is the torque and K_e is the back EMF constants, respectively, and T_L is the load (disturbance) torque.

3. Adaptive PID Control Algorithm Based on BPNN

The algorithm of the basic PID controller that is commonly known in the literature is given as in (2), [14].

$$u(k) = K_p e(k) + K_I \sum_{j=1}^{k} e(j) + K_D[e(k) - e(k-1)] \quad (2)$$

where K_P, K_I, K_D are the proportional, integral, and differential coefficients, respectively, and are the input/output sampling sequence of the control system, $y(k)$ is the controller output, $e(k)$ is the system error which is equal to $r(k) - y(k)$, and k is the iteration number. The general structure of the backpropagation neural network (BPNN) is shown in Fig. 1. Three inputs of the BPNN are as given as in (3).

$$\begin{aligned} x_1(k) &= e(k) - e(k-1) \\ x_2(k) &= e(k) \\ x_3(k) &= e(k) - 2e(k-1) + e(k-2) \end{aligned} \quad (3)$$

The learning algorithm of the BPNN is described as follows: The network input is $x(j)$, and the input and output of the hidden layer are given in (4), where j is the neuron number at input layer, and w_{ij}^{in} is the weight value of the input layer. The activation function of the hidden layer adopts a symmetrical sigmoid (tan-sigmoid) function given in (5), [15].

$$O_h(k) = \sum_{j=1}^{k} w_{ij}^{in} x_j, v_{ij}(k) = f(O_h(k)) \quad (4)$$

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (5)$$

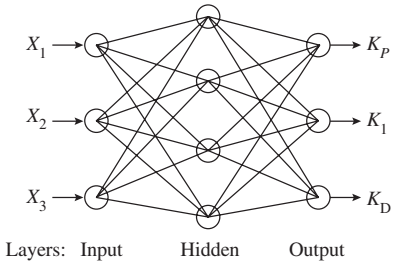


Fig. 1. BPNN structure

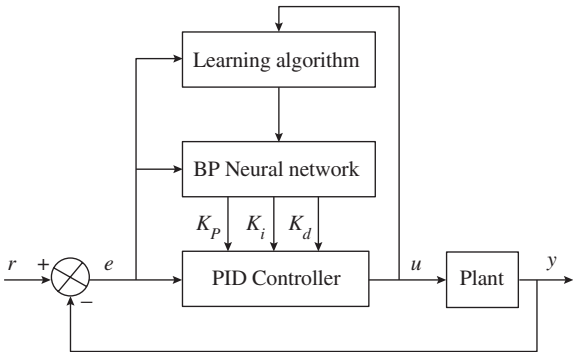


Fig. 2. PID control system structure

Therefore, the controller output can be obtained from (6).

$$\begin{aligned} u(k) = {} & f\left(v_{ij}(k) w_{ij}^h(k)\right) e(k) + f\left(v_{ij}(k) w_{ij}^h(k)\right) \sum_{j=1}^{k} e(j) \\ & + f(v_{ij}(k) w_{ij}^h(k))[e(k) - e(k-1)] \end{aligned} \quad (6)$$

where v_{ij} is the output of hidden neurons and w_{ij}^h is the weights of the hidden layers. In a manner similar to the least mean squares (LMS) algorithm, the BPA applies a correction to the synaptic weight w_{ij}, which is proportional to the partial derivative $\partial E/\partial w_{ij}$ where E is the system average square error. According to chain rule, the adjustment of the K_P, K_I, K_D parameters adopts the gradient as in (7). The adaptive PID control system structure based on BPNN is shown in Fig. 2. The variable parameters of the PID controller are determined according to (7) from BPNN.

$$\begin{aligned} \Delta K_P &= -\eta \frac{\partial E}{\partial K_P} = -\eta \frac{\partial E}{\partial y} \frac{\partial y}{\partial u} \frac{\partial u}{\partial K_P} = \eta e(k) \frac{\partial y}{\partial u} x_1(k) \\ \Delta K_I &= -\eta \frac{\partial E}{\partial K_I} = -\eta \frac{\partial E}{\partial y} \frac{\partial y}{\partial u} \frac{\partial u}{\partial K_I} = \eta e(k) \frac{\partial y}{\partial u} x_2(k) \\ \Delta K_D &= -\eta \frac{\partial E}{\partial K_D} = -\eta \frac{\partial E}{\partial y} \frac{\partial y}{\partial u} \frac{\partial u}{\partial K_D} = \eta e(k) \frac{\partial y}{\partial u} x_3(k) \end{aligned} \quad (7)$$

where $\partial y/\partial u$ is the Jacobian information of the controlled plant, which can also be obtained from the derivative of the plant's transfer function if available, and η is the learning rate.

4. Theory and Background of Interactive Adaptation

Depending on the application and configuration of the algorithm, the adjusted coefficients can be the ANN weights, PID gains, or transfer function coefficients. The theory of AI considers N subsystems called devices. Each device (indexed by $n \epsilon N = \{1, 2, \ldots, N\}$) has an integrable output signal y_n and an integrable input signal x_n. The dynamics of each device is described as a causal functional, as given in (8).

$$F_n = X_n \rightarrow Y_n, n \epsilon N \quad (8)$$

where X_n and Y_n are the input and output spaces, respectively. Therefore, the relation between the input and output of the nth device is given by (9).

$$y_n(t) = (F_n \circ x_n)(t) = F_n[x_n(t)], n \epsilon N \quad (9)$$

where $\circ$ denotes the functional composition.

Interactions among devices are achieved by connections which are denoted by 'C' as seen in Fig. 3. Devices whose output is conveyed by connection 'C' is denoted by pre_{cn}, and a device

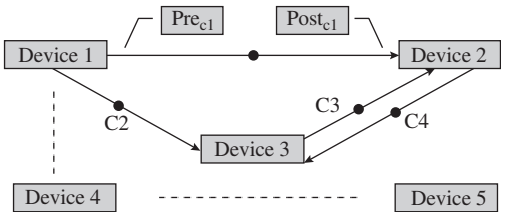


Fig. 3. Devices and their connections

whose input depends on the signal conveyed by 'C' is denoted by $post_{cn}$.

$I_n = \{c{:}pre_c = n\}$ is the set of input interactions for the nth device, $O_n = \{c{:}post_c = n\}$ is the set of output interactions for the nth device as shown in Fig. 3. Assuming linear interaction among devices and the external signal $u_n(t)$

$$x_n(t) = u_n(t) + \sum_{c\epsilon I_n} \alpha_c y_{pre_c}(t), n\epsilon N \tag{10}$$

where α_c are the connection weights. With this linear interaction, the dynamics of the system is described by (11).

$$y_n(t) = F_n\left[u_n(t) + \sum_{c\epsilon I_n} \alpha_c y_{pre_c}(t)\right], n\epsilon N \tag{11}$$

The goal of the adaptation algorithm is to adapt the connection weights α_c so the performance index $E(y_1, \ldots, y_n, u_1, \ldots, u_n)$ as a function of the inputs and outputs will be minimized by using the Fréchet derivative [16]. The system with dynamics can expressed as in (12).

$$y_n = F_n\left[u_n + \sum_{c\epsilon I_n} \alpha_c y_{\mathrm{pre}_c}\right], n\epsilon N \tag{12}$$

$$\dot{\alpha}_c = F'_{\mathrm{post}_c}[x_{\mathrm{post}_c}] \circ \frac{y_{\mathrm{per}_c}}{y_{\mathrm{post}_c}} \sum_{s\epsilon O_{\mathrm{post}_c}} \alpha_s \dot{\alpha}_s - \gamma F'_{\mathrm{post}_c}[x_{\mathrm{post}_c}] y_{\mathrm{per}_c} \frac{\partial E}{\partial y_{\mathrm{post}_c}} \tag{13}$$

If the connection weights α_c are adapted according to simplified adaptation algorithm as given in (13), [17], where c is an element of C and (13) has a unique solution, it leads to monotonic decrease in the performance index E with time. In fact, (13) is always satisfied, which is the unique solution to (14), where $\gamma > 0$ is the adaptation coefficient [18].

$$\dot{\alpha}_c = -\gamma \frac{dE}{d\alpha_c}, c\epsilon C \tag{14}$$

Using this algorithm, an ANN can adapt without the need of a feedback network to backpropagate the errors. The algorithm hence provides a biologically plausible mechanism for adaptation in biological neurons.

5. ANN Controller With Tuning Algorithm

An ANN is decomposed into multiple devices, which are taken in the system as neurons, as described in Fig. 3 and shown in Fig. 4(a). The ANN can be described as in (15) by ignoring the dynamics.

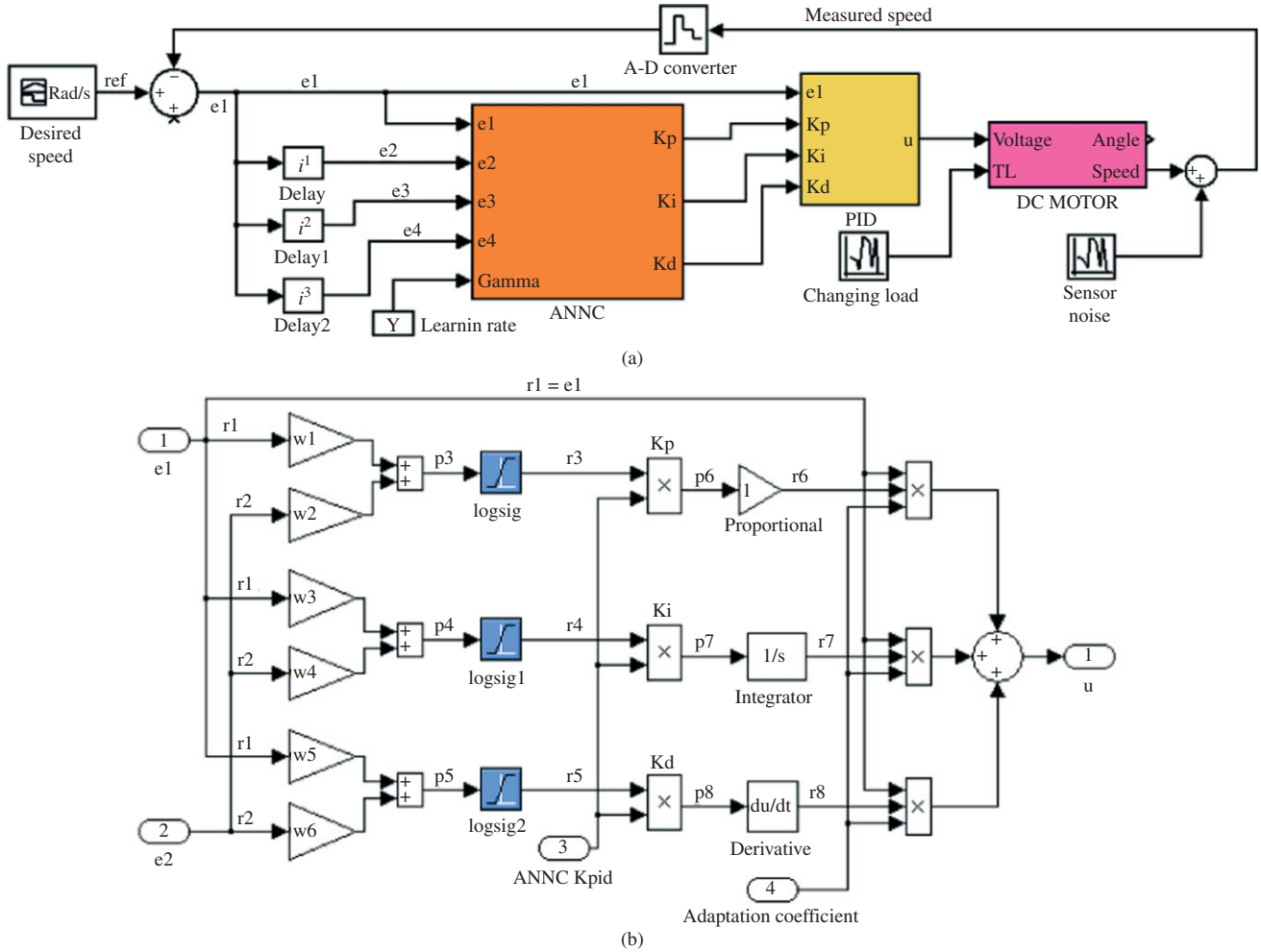


Fig. 4. NN-based control system. (a) Self-Tuning PID Controller. (b) Simplified schema of the controller

$$v_i = g(h_i) = g\left(\sum_{j\epsilon N} w_{ij} v_j + \xi_i\right), i\epsilon N \tag{15}$$

where $g(h_i)$ is the sigmoidal function, v_i is the output of the neuron i, h_i is the input of the neuron i, ξ_i is the external input of neuron i, and w_{ij} is the weight of the connection from neuron j to neuron i. For output neurons, t_i is the desired output of neuron i. The goal is to minimize the error as in (16).

$$E = \frac{1}{2}\sum_{i\epsilon N} e_i^2, e_i = \begin{cases} v_i - t_i, & \text{if } i \text{ is an output neuron} \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

Applying the adaptation algorithm to ANN, the following substitutions are obtained.

$$\alpha_c \to w_{ij} \qquad F'_{\text{post}_c}[x_{\text{post}_c}] \to g'(h_i) y_{\text{post}_c} \to v_i$$
$$\alpha_s \to w_{ki} \qquad \frac{\partial E}{\partial y_{\text{post}_c}} \to \frac{\partial E}{\partial v_i} = e_i$$

Therefore, the adaptation algorithm is given as in (17).

$$\dot{w}_{ij} = g'(h_i)\frac{v_j}{v_i}\sum_{k\in N} w_{ki}\dot{w}_{ki} - \gamma g'(h_i) v_j e_i \tag{17}$$

Algorithm (17) is mathematically equivalent to the BPA. However, it does not require a feedback network to propagate the errors. By eliminating the feedback network, the algorithm allows a much simpler implementation than the BPA. Using the AIA, the adaptation mechanism can be built within each neuron to make the neuron trainable. A trainable neuron can be built as a standard unit. In Fig. 4(b), w_n are the weights which are adjusted by error, and log-sig is the sigmoidal function which it is equal to (18).

$$g(x) = \sigma_x = (1 + e^{-x})^{-1} \tag{18}$$

Mathematically, the NN and adaptation algorithm are described as follows like (15).

$$p_n = \sum_{s\in D_n} w_s r_{pre_s} \to \phi_n = \frac{1}{2}\frac{d}{dt}\sum_{s\in D_n} w_s^2 = \sum_{s\in D_n} w_s \dot{w}_s \tag{19}$$

By applying the adaptation law in (13), the weight adaptation becomes as in (20).

$$\dot{w}_s = r_{pre_s}(\phi_{post_s}\sigma(-p_{post_s}) + \gamma f_{post_s}) \tag{20}$$

where n is the label for a particular neuron, s is the label for a particular synapse, D_n is the set of input synapses of neuron n, pre_s and $post_s$ are the presynaptic and postsynaptic neuron corresponding to synapse s, respectively, w_s is the weight of synapse s, p_n is the membrane potential, and r_n is the firing rate.

Equations (19) and (20) describe AIA for adaptation in NNs. As shown in Ref. [18], it is equivalent to the BPA but requires no feedback network to backpropagate the error. Figure 4(b) is obtained through this algorithm. The ANN controller (ANNC) has five inputs: e_1, e_2, e_3, e_4, and γ, as shown in Fig. 4(a). e_1 is the error between the set point and the output, and e_2, e_3, e_4 are delayed signals based on e_1. γ is the learning rate. Delayed signals are introduced in order to control the output to depend not only on the current input but also on past inputs, since the network controller is itself a memory-less device. Mathematically, the input–output relations of neurons are as presented for just e_1 and e_2 as follows (mathematical expression of Fig. 4(b)):

$$r_1 = e_1 \text{ and } r_2 = e_2$$
$$p_3 = w_1 r_1 + w_2 r_2, p_4 = w_3 r_1 + w_4 r_2, p_5 = w_5 r_1 + w_6 r_2$$
$$r_3 = \sigma(p_3), r_4 = \sigma(p_4), r_5 = \sigma(p_5)$$
$$p_6 = w_7 r_3, p_7 = w_8 r_4, p_8 = w_9 r_5$$
$$r_6 = p_6, r_7 = s^{-1} p_7, r_8 = s p_8$$
$$p_9 = r_6 + r_7 + r_8$$

Let

$$E = e_1^2 = (r - y)^2 \to \frac{\partial E}{\partial y} = -2(r - y) = -2e_1 \tag{21}$$

Weights are calculated as follows by using (13) and (14):

$$\dot{w}_1 = r_1(\phi_3\sigma(-p_3) + \gamma_0) = e_1\phi_3\sigma(-p_3)$$
$$\dot{w}_2 = r_2(\phi_3\sigma(-p_3) + \gamma_0) = e_2\phi_3\sigma(-p_3)$$
$$\dot{w}_3 = r_1(\phi_4\sigma(-p_4) + \gamma_0) = e_1\phi_4\sigma(-p_4)$$
$$\dot{w}_4 = r_2(\phi_4\sigma(-p_4) + \gamma_0) = e_2\phi_4\sigma(-p_4)$$
$$\dot{w}_5 = r_1(\phi_5\sigma(-p_5) + \gamma_0) = e_1\phi_5\sigma(-p_5)$$
$$\dot{w}_6 = r_2(\phi_5\sigma(-p_5) + \gamma_0) = e_2\phi_5\sigma(-p_5)$$

where

$$\phi_3 = w_7\dot{w}_7, \phi_4 = w_8\dot{w}_8, \phi_5 = w_9\dot{w}_9 \tag{22}$$

The adaptation law for w_7, w_8, and w_9 is more complicated as it is linked to the plant to be controlled. By using (13), O_{post_c} is the empty adapted weight is given in (23).

$$\dot{w}_7 = -\gamma F'_{\text{post}_c}[u] r_3(-2e_1) \tag{23}$$

If the Fréchet derivative is approximated by a constant that will be absorbed in γ, then the above expression is approximated by (24).

$$\dot{w}_7 = \gamma r_6 e_1, \dot{w}_8 = \gamma r_7 e_1, \dot{w}_9 = \gamma r_8 e_1 \tag{24}$$

The constant γ, which is varied to analyze the rate adaptation of the ANNC, is considered as the learning rate. Therefore, decomposed subsystems (four devices) are as follows: Device 1 is the proportional part with transfer function 1; Device 2 is the integral part with transfer function s^{-1}; Device 3 is the derivative part with transfer function s; and Device 4 is the BLDCM's transfer function given in (1). In any case, there are three adaptive connections: $\dot{a}_c = K_P, K_I, K_D$. K_P, K_I and K_D may be rewritten as below by using (25).

$$K'_p = \gamma r_3 e_1, \quad K'_I = \gamma r_4 e_1, \quad K'_D = \gamma r_5 e_1 \tag{25}$$

6. Simulation Results

In order to simulate the behavior of the prepared self-tuning PID, MATLAB Simulink model of the proposed method is performed. The reference speed, actual speed, and error curves for the speed control of the DC motor, whose transfer function is given

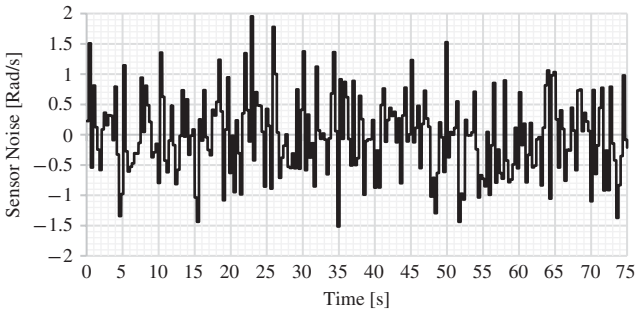


Fig. 5. Sensor noise (noise power: 0.12; sample time: 0.3; seed: [23341])

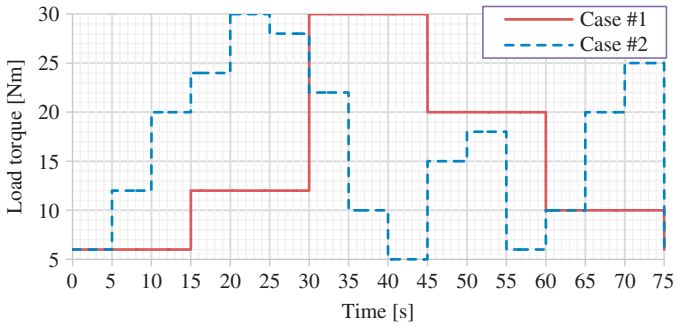


Fig. 6. Load torques (T_L) according to cases

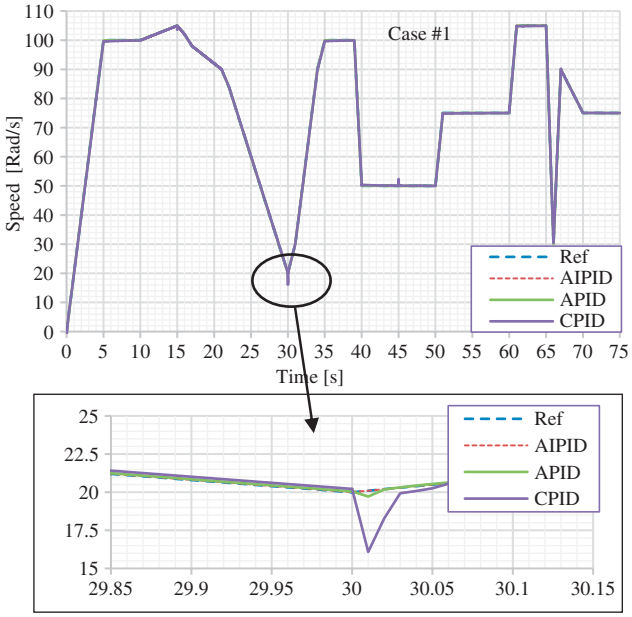


Fig. 7. Obtained speed responses for Case #1

in (1), are determined using the Simulink models under the steady-state condition. The motor speed is controlled so as to follow the reference speed, which is designed in MATLAB via a signal generator. In this study, a brushless BLDCM of 140 V, 2.536 kW, 24.4 A, 30 Nm, 1000 rpm (104.72 rad/s) and having the following parameters is used: $R_a = 0.314\Omega$, $L_a = 1.97$ mH, $J = 0.0241$ kgm^2, $B = 0.3$ Nm/rad/s, $K_e = K_t = 1.22$ Nm/A [19]. And a random load (T_L) whose sample time is variable and changes between 5 and 30 Nm is applied to the BLDCM to observe the adaptation. In order to model the system more realistically, sensor noise, whose waveform is given in Fig. 5, and the effect of the A/D converter signal are taken into account as seen in Fig. 4(a).

The modified tuning algorithm is applied to a BLDCM with a constantly changing load. Simulation consists of two cases as summarized below and shown in Fig. 6.

Case #1. When sample time of load is 15 s.

Case #2. When sample time of load is 5 s.

To be able to realize the accuracy of the proposed AIA and sensitivity of the controller designed by using AIA, an adaptive PID controller based on BPNN and conventional PID controller have been designed for the speed control of the same BLDCM.

BPNN-based adaptive PID controller is proposed according BP algorithm described in Section 3. Constant gains of the conventional PID controller are determined as $K_P = 20.5$, $K_I = 2.14$, $K_D = 0.412$ by using Ziegler–Nichols method.

Figures 7 and 8 show the results of the simulation for Case #1 and #2 with the voltage input being a velocity reference signal,

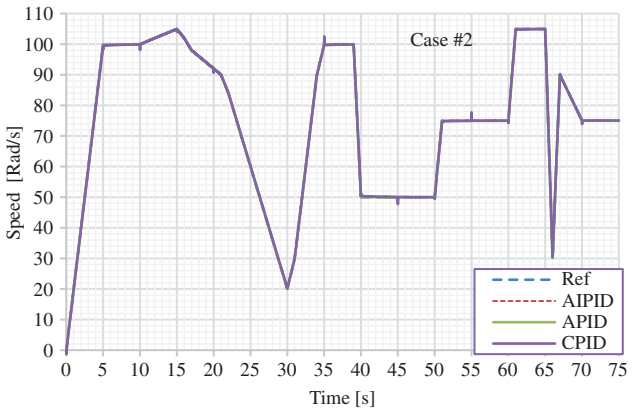


Fig. 8. Obtained speed responses for Case #2

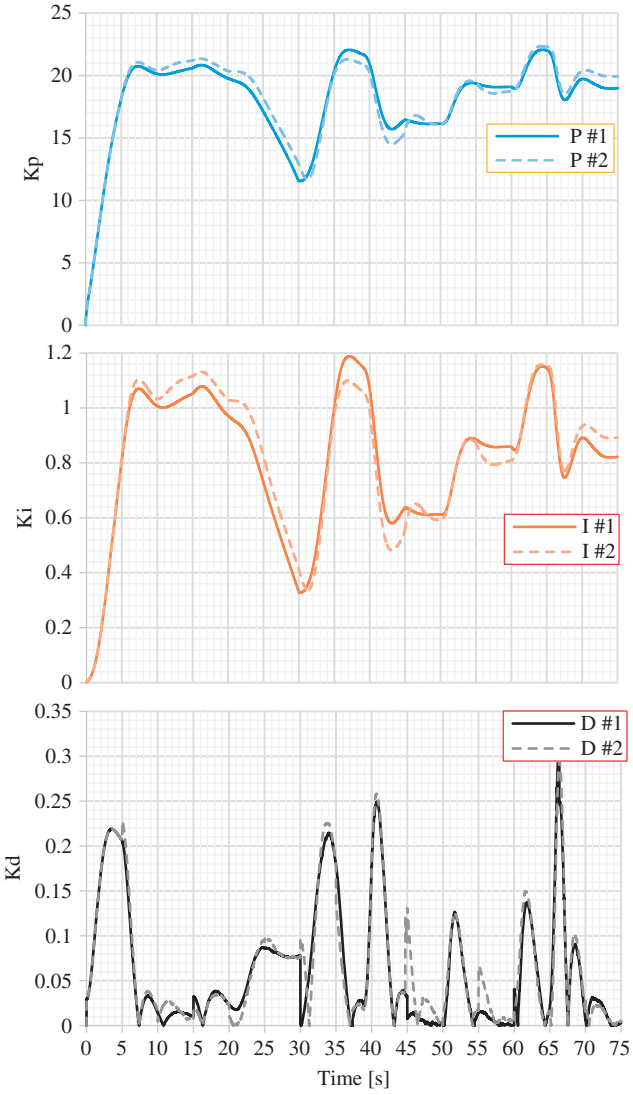


Fig. 9. Variation of the PID gains of AIPID controller

changing between 0 to 105 rad/s. The adaptation coefficient of AIPID is determined as $\gamma = 50$, and also the learning rate of the ANNPID is determined as $\eta = 0.3$. As seen in the Figs 7 and 8, control responses of the controllers are quite good for both cases.

As shown in Figs 7 and 8 the rise times and overshoots are excellent except the CPID. Variations of the PID gains of the adaptive controllers are given in Figs 9 and 10 for both cases. Legends of the graphs indicate the case numbers: e.g., I #1:

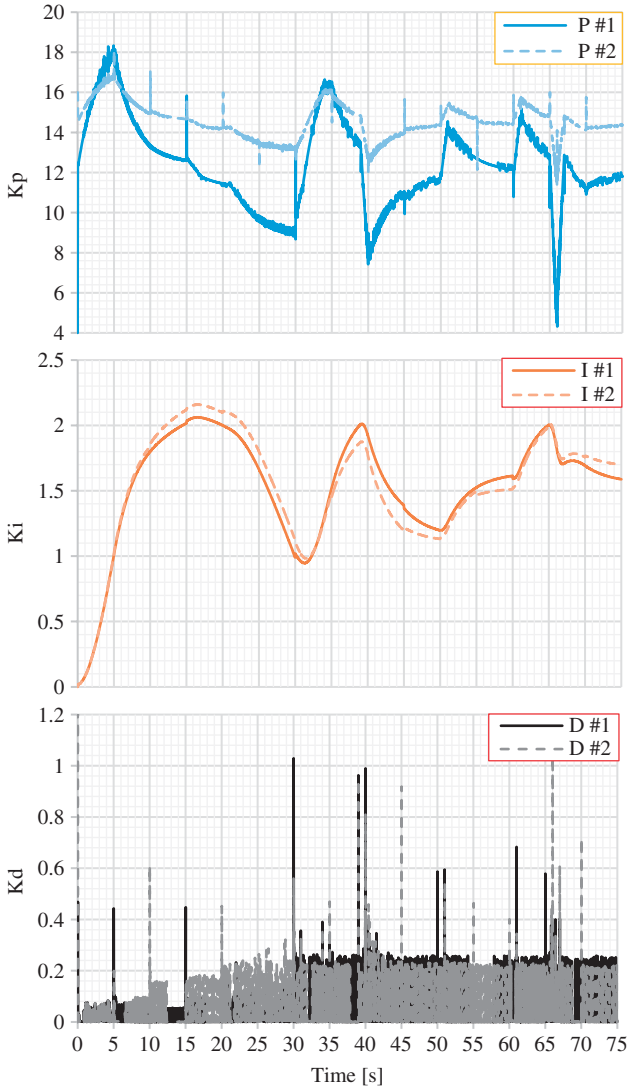


Fig. 10. Variation of the PID gains of ANNIPID controller

Table I. Speed errors according to cases[a]

	CPID		ANNPID		AIPID	
	SSE	SAE	SSE	SAE	SSE	SAE
Case #1	0.0106	9.70776	0.00335	1.91771	0.00045	0.62722
Case #2	0.0076	9.77866	0.00205	1.89957	0.00057	0.66396

[a](ST:0.01). SSE, steady-state error; SAE, sum of absolute errors.

Integral gain for Case #1. As seen in the figures, the PID gain magnitudes of all controllers are almost within the same limits. As seen in Figs 9 and 10, all PID gains show greater change according to the current speed and load torque.

The K_P and K_I gains of the ANNPID controller show sharp changes at every moments of change in load torque, simultaneously. On the other hand, gains of the AIPID controller show smooth changes even if the load torque shows very sharp and large changes.

Steady-state error (SSE) and sum of absolute errors (SAE) are calculated from the recorded data during the simulation in order to realize the robustness of the controllers. Sample time (ST) of the simulation is chosen as 0.01 s, which means there are 7500 data measurement points during the 75-s simulation time. At the end

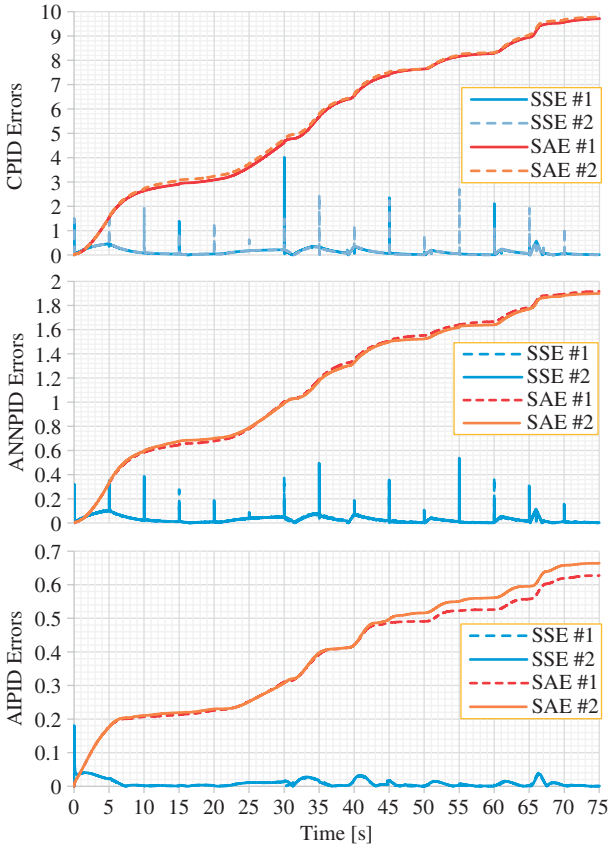


Fig. 11. Variation of the SSEs and SAEs according to cases

Table II. Memory usage and calculation time of the simulations

	CPID	ANNPID	AIPID
RAM (%)	27	35	29
Cal. time (s)	5	15	10

of the simulation, the obtained results are summarized as seen in Table I.

Waveforms of the SSE and SAE for both cases are given in Fig. 11. As seen in Table I, SSE and SAE of the AIPID controller are lower than those of the other controllers because, while the AIPID controller absorbs the load change impact very smoothly, the other controllers cannot, as seen in Fig. 11. Furthermore, because of the low sampling time of the Case #2, errors are higher than in Case #1, as expected.

Simulations were performed on a desktop PC with an Intel Core i5@3.20 GHz and 6 GB RAM. Memory use and calculation time for the simulations are given in Table II. As seen in this table, the simulation performance of the AIPID is better than that of the other controllers.

7. Conclusion

The application of the theory of AI to ANN control results in a direct adaptation algorithm that works very well. The PID tuning algorithm proposed in this paper has many advantages like a low error rate, low calculation time, low memory usage, and high frequency response in BLDCM control applications. The simulation results shows that it performs very well even in constantly changing load conditions. It is possible to reduce the considerably the small error still further. This new approach does not require the transformation of the continuous time domain

plant into its NN equivalent. Another benefit of applying the proposed algorithm is that it does not require a separate feedback network to backpropagate the error. The adaptation algorithm is mathematically isomorphic to the BPA.

It is realized that this self-tuning PID controller with proposed AIA can be preferred in sensitive speed control application areas because of its very low speed control error rate compared to conventional and BP-based ANN controllers.

References

(1) Huang X, Xu J, Wang S. Nonlinear system identification with continuous piecewise linear neural network. *Neurocomputing* 2012; **77–1**:167–177.

(2) Peng J, Dubay R. Identification and adaptive neural network control of a DC motor system with dead-zone characteristics. *ISA Transactions* 2011; **50–4**:588–598.

(3) Bennett S. Development of the PID controller. *IEEE Control Systems Magazine* 1993; **13**:28–38.

(4) Cominos P, Munro N. PID controllers: recent tuning methods and design to specification. IEE Proceedings. Part D, Control Theory and Applications 2002; **149(1)**:46–53.

(5) Howell MN, Gordon TJ, Best MC. The application of continuous action reinforcement learning automata to adaptive PID tuning. IEE Colloquium on Sources and Training of knowledge Engineers 1990 (Digiset No: 1990/69); 5–8.

(6) Cameron F, Seborg DE. A self-tuning controller with a PID structure. International Journal of Control 1983; **38(2)**:401–417.

(7) Hsieh SP, Hwang TS. Dynamic modeling and neural network self-tuning PID control design for a linear motor driving platform. IEEJ Transactions on Electrical and Electronic Engineering 2010; **5**:701–707.

(8) Vega P, Prada C, Aleixander V. Self-tuning predictive PID controller. *IEE Proceedings—Control Theory and Applications* 1991; **138(3)**:303–311.

(9) Martins FG, Man C. Application of feed-forward artificial neural to improve process control of PID-based control algorithms. *Computers and Chemical Engineering* 2000; **24**:853–858.

(10) Chen J, Huang TC. Applying neural networks to on-line updated PID controllers for nonlinear process control. *Journal of Process Control* 2004; **14(2)**:211–230.

(11) Narendra KS, Parthasarathy K. Gradient methods for optimization of dynamical systems containing neural networks. *IEEE Transaction on Neural Network* 1991; **2**:252–262.

(12) Lin F, Brandt RD, Saikalis G. Self-tuning of PID controllers by adaptive interaction. Proceedings of the American Control Conference 2000, Chicago, Illinois; 3676–3681.

(13) Brandt RD, Lin F. Adaptive interaction and its application to neural networks. *Information Sciences* 1999; **121–3(4)**:201–215.

(14) Guo B, Liu H, Luo Z, Wang F. Adaptive PID controller based on BP neural network. IEEE International Joint Conference on Artificial Intelligence 2009, Hainan Island; 148–150.

(15) Han W, Han J, Lee C. Development of a self-tuning PID controller based on neural network for nonlinear systems. Proceedings of the 7th Mediterranean Conference on Control and Automation (MED99) 1999, Haifa, Israel; 979–988.

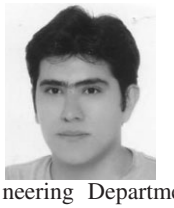
(16) Slotine JJE, Weiping L. Applied Nonlinear Control. Prentice Hall: Englewood Cliffs, New Jersey; 1989.

(17) Brandt RD, Lin F. Can supervised learning be achieved without explicit error back-propagation? Proceedings of the International Conference on Neural Networks 1996, Washington, DC; 300–305.

(18) Brandt RD, Lin F. Supervised learning in neural networks without feedback network. IEEE International Symposium on Intelligent Control 1996, Dearborn, MI; 86–90.

(19) AC Servo Motors and Servo Rated Gearheads for the automation industry. http://www.baldor.com/pdf/literature/BR1202-E.pdf. Accessed June 8, 2013.

Tayfun Gundogdu (Non-member) was born in Çorum, Turkey, in 1985. He obtained the B.S. degree from Gazi University in 2009 and the M.S. degree from Istanbul Technical University (ITU) in 2012, both in electrical engineering. He is presently pursuing the Ph.D. degree and also is working as a Research Assistant in the Electrical Engineering Department, ITU. His research intewrests include the design, analaysis and control of electrical machines, power electronics, magnetism and magnetic materials, artificial neural networks, fuzzy logic, microcontrollers, harmonics, and power system analysis.



Guven Komurgoz (Non-member) was born in Malatya, Turkey. She received the B.S., M.S., and Ph.D. degrees, all in electrical engineering, from the Istanbul Technical University (ITU), Turkey, in 1991, 1995, and 2002, respectively. She is currently an Assistant Professor of electrical engineering at ITU. Her research interests include heat transfer, numerical methods, and the design of transformers and rotating electrical machines.