Design and Implementation of a Closed Loop Sensor-less Position/Speed/Current Control of a DC Motor using Neural Network for Robotic Applications

N. A. Yehia #1, S. Rezeka #2, M. El-Habrouk #3
#1 Faculty of Engineering, Arab Academy for Science, Technology and Maritime Transport (AASTMT), +2010-68236274, nyehiaahmed@yahoo.com
#2 Faculty of Engineering, Arab Academy for Science, Technology and Maritime Transport (AASTMT), +2012-24611737, srezeka@yahoo.com
#3 Faculty of Engineering, Alexandria University, Egypt, +2010-3459772, eepgmme1@yahoo.co.uk

ABSTRACT

This paper presents a concise review of the control techniques of DC motors with the various control methods and optimisation techniques used in the literature. The paper presents a block diagram relating these control techniques. The paper then presents the closed loop control of a DC motor using Artificial Neural Networks (ANN). The closed loop control takes into account all three loops pertaining to current, speed and position control of the DC motor used for robotic applications. The proposed controller uses feedback data from an ANN estimator of the DC motor model, hence eliminating the need for mechanical feedback sensors, which is very advantageous in robotic applications. Simulation results are presented in the course of the paper for each of the control cases. The paper presents a performance comparison between the classically tuned PID controllers and the proposed ANN controller with and without the ANN DC motor estimator.

Key words: DC Motor, PID, ANN, Feedback Estimation, Closed Loop Control.

Corresponding Author: N. A. Yehia

INTRODUCTION

DC motors have always had the largest share in robotic and industrial control applications as the main workhorse actuators and the easiest to operate and control [1-4]. They have been used from the smallest ratings of a few Watts for the small robotic arms to the gigantic MegaWatt ratings in industrial automation systems [1, 2].

In low-power applications (ranging up to a few hundreds of Watts) [2-4], speed control applications are implemented without the use of current control loops since the currents are
already miniscule and are restricted by the large value resistances of the armature circuits. The sudden current increases are not likely to occur and hence do not represent any threat to the motor itself or the power semiconductor devices used to perform the control process [1, 4].

In high-power applications (above 10kWatts) the use of DC motors is completely shadowed by the presence of vector-controlled, direct-torque-controlled and direct-self-controlled squirrel-caged induction motors [2-4], which are much cheaper and require minimal maintenance as compared to DC machines [1-4]. The complexity of their control is shadowed by the DC motor frequent maintenance requirements in these power ratings [4]. In addition, the availability of high performance and high speed Digital Signal Processors (DSPs), which are required to perform the millions of floating point operations of the complex controllers, has made the induction motor precise control rather applicable in these power ranges [3, 4].

Medium-power applications (ranging from a few hundreds of Watts up to 10kWatts) [2], on the other hand, constitute a dilemma; since the playground is open for both DC and AC motors. The applications of AC motors mandate the highest technology of DSPs which would greatly increase the machine and controller cost and complexity [3, 4]. The accuracy of the DC machine is however stellar with minimal controller requirements [1, 2]. A simple controller with a non-expensive microcontroller renders the machine optimal for both speed and angle control processes [1].

Both the Medium and High power applications DC motors require the presence of current controllers in addition to the speed and/or position controllers. Most of the publications in the literature address the problem of the speed control which has been thoroughly discussed [5-9]. The problem of position control was also addressed in many publications [10-12] but they scarcely employ a three controller configuration system (Position/Speed/Current) [13]. The full system requiring three tuned controllers would be able to perform any angle motion with any speed profile without exceeding the pre-set current limits [14]. This is especially important in these power ranges since the armature resistances of the machines would be considerably low, hence allowing fast current variation due to transients. This is the main theme of this paper.

Another problem pertaining to the control process is the feedback. The system would require feedback devices for the motor variables (namely, voltage, current, speed and position) [7, 15, 16]. These are sometimes difficult to incorporate especially the mechanical sensors (speed and position) in harsh and tight environments [1, 3, 4, 17]. The introduction of the use of speed estimators is very popular in AC induction machines and brushless DC machines [7, 16, 18]. It is also used for DC machines [1, 2, 15, 19] using various techniques ranging from Artificial Neural Networks [20], Particle Swarm Optimisation [21, 22] as well as other techniques [19, 23]. Despite the accuracy degradation, which is expected with the use of speed (and hence position by integration) estimation, these systems are still acceptable in industry in the cases where an external additional/correcting feedback is available using for example vision sensors [1, 3, 17, 18, 24].
The system proposed in this paper is applicable to all these ratings taking into consideration the requirements of each of them. The proposed system uses a medium-power motor with the parameters outlined in the appendix at the end of this paper.

The paper starts by discussing the available DC motor control techniques used in the industry. This is followed by the quick review of PID controller requirements as well as Artificial Neural Networks (ANN). The implementation of an ANN approximation of the various PI controllers of the motor as well as an all-controller which replaces all three motor controllers is then presented in details. The implementation of a speed estimator is also presented in this paper as well as its integration with the previous all-neural controller. The simulated performance is presented for each of these cases coroneted by the conclusion of the paper.

**DC MOTOR CONTROL TECHNIQUES**

The block diagram of Fig.1 represents the different control techniques implemented for DC motors. The techniques vary between conventional and modern control techniques in addition to the heuristic techniques which have evolved in the recent publications [1, 2, 17]. Several conventional controllers use the PID controller with its various tuning techniques [25-33] as well as the pole placement techniques [34, 35]. PID controllers can come in two flavors, namely Full-Order-PID (simply PID) [26, 31, 33, 36, 37] or Fractional-Order-PID (FOPID) [29, 36, 38]. Modern controllers have been applying modern control theory with adaptive/optimal and sliding mode controllers [12, 39-44]. These normally follow [21, 45, 46] the Least or Mean Square error (LSE or MSE) or the Nonlinear Auto-Regressive with Moving Average (NARMA) models. The more advanced techniques employing heuristic control methods such as Artificial Neural Networks (ANN) [1, 11, 24, 47-50], Fuzzy Logic Controllers (FLC) [1, 30, 51-55] and Neuro-Fuzzy Systems (NFS) [2, 6, 56] have evolved in the past few years presenting better performance and improved transient as well as steady state characteristics [47, 56, 57]. They however require either offline- or online-training, which is quite computationally demanding [1, 2, 58]. On the other hand the implementation of such systems with conventional and cheap microcontrollers are rather easy [1].

The improvements introduced by the plethora of modern controllers, such as self-tuning, adaptive and sliding mode, are numerous [8, 12, 28, 39, 40, 59, 60]. They include system order reduction, exact control for both transient and steady state applications as well as increased robustness with respect to parametric uncertainties and disturbances [12, 39, 40, 61, 62]. Their major drawback, in addition to noise sensitivity [12, 39, 40, 60], is that they are accompanied by an exponential increase in computational complexity in conjunction with an increasing demand for better and more complex Digital Signal Processing (DSP) hardware implementations [2-4, 12]. The hardware complexity of DSPs introduced herein is of course accompanied by a corresponding steep price increase [2, 4, 12, 28, 59, 62]. The question would of course remain on the viability of the complex DSP controllers mandated by the control algorithms and the increased computational complexities as compared to tuned conventional PID systems [2, 28, 40, 59].
The heuristic techniques on the other hand have been infiltrating system controllers yielding rather acceptable performance dependent on their training process [58]. The Artificial Neural Networks (ANN), which has an abundant research share [5, 6, 8-11, 14, 15, 17, 20, 24, 31, 47-50, 52, 56, 58, 63-67], requires a considerable amount of off-line training in order to reach the required optimal weights and hence a good performance. The training process of ANN systems is rather tedious and lengthy. It however possesses the ability to generate control effort output relevant to the input even though the input was not among the training sequence [68-70]. This is performed using their ability to extrapolate functions [58, 65, 68-70].

Fuzzy Logic Controllers (FLC) [1, 6, 30, 45, 51-55, 70, 72-74] on the other hand do not require the same amount of training. They simply need the definition of the proper number, shapes and parameters of the membership functions. They also require a good understanding of the system operation in order to fill-in the required rule-base [6, 51]. Both ANN and FLC require more off-line fine tuning and simulations before being fully operational [1, 5, 15, 30, 51, 54]. They both do not require a large run-time computational load burden. A simple multiply-accumulate process can be implemented for the ANN, while the FLC requires simple computations in addition to logic function comparisons [1, 6, 30, 51, 53, 65, 75].

Fig 1: DC Motor Control Techniques
A different approach found in the literature revolves around optimising the weights of the ANN controller or the parameters of the FLC, which is normally performed off-line [1, 6, 72]. In this case the system can enjoy the lower computational burden of the ANN or the FLC controllers. Neuro-fuzzy systems (NFS) [53, 75-78] refer to combinations of artificial neural networks and fuzzy logic. NFS incorporates the human-like reasoning style of fuzzy systems through the use of fuzzy sets and a set of linguistic variables with a set of “IF-THEN” fuzzy rules. Such systems show two distinct behaviours [6, 72]: Separately, each one of these techniques possess advantages and disadvantages that, when mixed together, their cooperation provides better results than the ones achieved with the use of each isolated technique [1, 6, 72].

The optimisation of the PID weights or the ANN, FLC and NFS parameters is the state of the art technique, which is found throughout a large amount of publications. This optimisation process is largely based on heuristic optimisation techniques, such as Particle Swarm Optimisation (PSO) [21, 32, 37, 38, 41-45, 74, 77-88] (and binary PSO [21, 22, 89]), Ant Colony Systems (ACS) [90], Bee Colony Systems (BCS) [84, 91], Cuckoo Search (CS) techniques [80, 92], Genetic Algorithms (GA) [13, 37, 52-55, 72, 73, 75, 76, 79, 80, 82, 93-96], Gravitational Search Algorithms (GSA) [27] and other techniques [55, 80, 83, 85, 97], are quite abundant in the literature. They produce the optimal or sub-optimal control efforts (of the PID or the FLC) required by the system in order to implement the control function. They are used online in order to change the parameters of the controller in order to improve the transient and steady state performance of the system [2].

ANN systems do however have an advantage. They can be used for signal estimation which is not the case of any of the other techniques [1, 3, 20] without recurrence to elaborate signal estimation and system models. The control process, using either the conventional PID or any other control technique, requires the presence of signal feedback from the motor. The feedback required is normally the voltage and current waveforms in addition to the mechanical speed for speed control systems and the mechanical speed and position for position control systems [2, 20]. The use of the ANN would be useful in estimating the mechanical speed of the machine knowing the values of the voltages and currents [20]. This is a rather important aspect since the speed and/or position angle sensing would require a shaft encoder which is normally expensive and not easily installed in motors which do not possess a built-in version of the encoder. Another problem relating to shaft encoders is their use in harsh environments (due to mechanical fixation problems) and in tight areas (due to the lack of space) [2, 3]. This is mostly the case for Robotic applications where the confined space can only fit the motor and the use of an external shaft encoder would be problematic. The signal estimation in conjunction with the implementation of the ANN controller has mainly been addressed for other types of motors (AC induction machines and brushless DC machines) in the literature [1-4].

The following two sections describe the PID controller implementation and tuning using the simplest trial and error method in comparison with the Artificial Neural Network (ANN) techniques, which is presented in the latter section. As mentioned above, the ANN implementation involves only computations of multiply-accumulate processes which are
relatively simple and do not require the presence of an elaborate DSP. A mere simple microcontroller can be used for their implementation.

CLASSICAL CLOSED LOOP PID IMPLEMENTATION OF DC MOTOR controllers

Fig.2 presents the basic block diagram of a PWM controlled DC motor with current, speed and position loops. The system uses PID controllers in order to operate the motor according to the system position requirements without violating the maximum currents and speed limitations imposed by the overall system [26, 31, 37].

![PID Control System Diagram](image)

**Fig 2: Current/Speed/Rotation-Angle PID Control System**

PID Control (Proportional-Integral-Differential) is by far the widest type of automatic control used in industry [2]. A proportional–integral–derivative controller (PID controller) is a generic control loop feedback mechanism widely used in industrial control systems. A PID controller will correct the error $e(t)$ between the feedback from the output and the desired input or set point by calculating and giving a control effort $e_c(t)$ for correction that will adjust the process accordingly. A PID controller has the general form [83],

$$e_c(t) = K_p e(t) + K_i \int_{0}^{t} e(\tau) d\tau + K_d \frac{de}{dt}$$  \hspace{1cm} (1)$$

where, $K_p$ is the proportional gain, $K_i$ is the integral gain, and $K_d$ is the derivative gain. The values of the three gain variables ($K_p$, $K_i$ and $K_d$) determine the appropriate controller design capable of yielding best performance for any system inputs and reaching these values is called the tuning process of the PID controllers [33].

No controller has ever enjoyed the fame of the PID controller despite its drawbacks and difficulties in tuning. It is by far the most widely used control technique. Over 85% of all dynamic controllers are of the PID variety [2]. There is actually a great variety of types and design methods for the PID controller tuning and design [80] as discussed briefly in the previous section.

The tuning of the PID controllers takes several forms [80]. The most commonly used ones are the manual tuning method which requires experienced personnel and requires no math,
while the second method is the famous Ziegler-Nichols method which incorporates some trial and errors and is a real proven method [79].

The transfer function of a PID controller is found by taking the Laplace transform of Equation (1) and yielding:

\[ K_p + \frac{K_i}{s} + K_d s = \frac{K_ds^2 + K_ps + K_i}{s} \] (2)

The standard PID control configuration is as shown in Fig.3. In this configuration, the control signal \( e_c(t) \) is the sum of three terms. Each of these terms is a function of the tracking error \( e(t) \) and requires a saturation at the integrator and at the output of the PID block, which prevents the output from reaching very large values and causing arithmetic overflow (in digital implementations of PID controllers) [3, 4]. Moreover, the Integral part of the controller will not be able to come out of saturation in proper time to cope with the system performance [2].

The addition of this integral term makes the open-loop forward path of Type 1. Thus, the system, if stable and is guaranteed to have zero steady-state error to a step input [26]. If \( e(t) \) is non-zero for any length of time, the control signal gets larger and large as time goes on. It thus forces the plant to react in the event that the plant output starts to drift [33].

It is the required of the PID controller to obtain optimal behaviour. This implies that the system should be able to offer proper regulation (disturbance rejection) as well as command tracking (implementing set-point changes) [26, 33]. These refer to how well the controlled variable tracks the desired value. Specific criteria for command tracking include rise time and settling time, as shown in Fig.4. Some processes must not allow an overshoot of the process variable beyond the set-point [2].

![PID Controller Configuration](image)

**Fig 3: PID Controller Configuration**

![Time Performance Characteristics](image)

**Fig.4: Time Performance Characteristics**
Fig. 4 shows the response in the process output variable with three different controller gains due to a step change of the setpoint. The response shown with minimum overshoot and fast response (shown in thick line) in Fig. 4 is representative for acceptable stability [33].

ARTIFICIAL NEURAL NETWORKS

Recently, ANNs have been found to be an important technique for classification and optimisation problem. They have emerged as a powerful learning technique to perform complex tasks in highly nonlinear dynamic environments. Some of the prime advantages of using ANN models are their ability to learn based on optimisation of an appropriate error function and their excellent performance for approximation of nonlinear function. The ANN is capable of performing nonlinear mapping between the input and output space due to its large parallel interconnection between different layers and the nonlinear processing characteristics [68-70].

A neural network’s ability to perform computations is based on the hope that one can reproduce some of the flexibility and power of the human brain by artificial means [10]. Network computation is performed by a dense mesh of computing nodes and connections. They operate collectively and simultaneously on most or all data and inputs. The basic processing elements of neural networks are called artificial neurons, or simply neurons. Often they are simply called “nodes”. They are often organized in layers, and feedback connections both within the layer and toward adjacent layers are allowed. Each connection strength is expressed by a numerical value called a weight, which can be modified. The terms described above are shown in Fig. 5.

As shown in Fig. 5, each neuron is associated with three parameters whose learning can be adjusted; these are the connecting weights, the bias and the slope of the nonlinear function \( f(a) \), where “a” is the sum of the products of the inputs and weights [68].

![Neuron Implementation](image)

**Fig. 5: Neuron Implementation**

Neural networks usually use a learning mode, in which an input is presented to the network along with the desired output and the weights are adjusted so that the network attempts to
mimic the desired output. This process is called training. Random initial “weights” are transformed into meaningful information after training [68].

A multilayer feed-forward neural network consists of an input layer, one or more hidden layers, and one output layer of neurons, as shown in Fig.6. A neural network that has no hidden layers is called a Perceptron. However, a perceptron can only represent linear functions, so it is not powerful enough for the applications under question [5, 10, 11]. On the other hand, a multilayer feed-forward neural network can represent a very broad set of nonlinear functions, rendering it very useful.

The main problem in Multi-Layered-Neural-Networks is to determine the number of hidden layers, which requires experimentation to determine the best number of hidden layers [68, 69]. On one hand, too few hidden layers would prevent the network from being able to learn the required function (if it is non-linear in nature), as it will have too few degrees of freedom. On the other hand, too many hidden layers may cause the network to tend to over-fit the training data, thus reducing generalization accuracy. The use of too many hidden layers can also significantly increase the training time [68, 69].

As discussed above, multilayer feed-forward Artificial Neural Network is an important class of networks. The input signal propagates through the network in a layer by layer fashion in a forward direction. This multilayer feed-forward Artificial Neural Network is trained in supervised manner with a highly popular algorithm known as error back propagation algorithm. This algorithm is based on error correction learning rule. Basically, back-propagation learning consists of two passes through the difference layers of the network; feed forward pass and backward pass [8].

The total error in the performance of the network with a particular set of weights can be computed by comparing the actual and the desired output vectors for each presentation of an input vector [8]. Learning comprises changing weights so as to minimize the error function and to minimize mean square error (MSE) by any gradient descent method.
PI CONTROLLER IMPLEMENTATIONS USING ANN

A PI controlled DC motor Position/Speed/Current controller drive was implemented on Matlab/Simulink as shown in Fig.7-a. The DC motor and system parameters are given in the Appendix. The block labelled “power circuit” includes the DC motor coupled with a DC chopper. The DC drive uses a chopper of 25kHz in order to control the PWM output fed to the DC machine. The figure shows the implementation of the system with feedback from the current and speed signals. The angle feedback is effectuated using integration of the angular speed signal. The system employed herein uses only PI controller for the tuning which was sufficient for the performance achievement. The three PI controllers were tuned using the conventional trial and error techniques.

The system response is presented in Fig.7-b, where the tuned response of the system shows a rate of change of the rotation angle of around 500 radians. This value is equivalent to around 79 complete turns of the motor shaft. On the other side of the gearbox (of ratio 1:80), this would imply that the system would have rotated only one turn. This in effect is very practical in the case of robotic arm applications, where the 500 radians are performed in around 0.7 sec and a complete turn from -500 to +500 radians is effectuated in 1.45 sec. This is considered as a very good response of the system since the time taken is the minimum possible due to the inertia of the motor and the gearbox. The oscillations of the system are also minimal. Further improvements to the PI controllers (or the inclusion of a derivative term) could have been effectuated but the main aim in this research is to enable the ANN to implement the same characteristics of the PI controllers. The curves, shown in Fig.7-b in order from top to bottom, are for: “Current in Ampere, Speed in rpm, Field Current in Ampere, Electromagnetic Torque, Applied DC Voltage, and Rotational Angle”.

Fig.7-a: DC motor Matlab/Simulink implementation for Position/Speed/Current Controllers
The implementation of the ANN PI controllers is performed on 4 steps according to the following:

- Current-Controller-only ANN implementation (Fig.8-a and Fig.9-a).
- Speed-Controller-only ANN implementation (Fig.8-b and Fig.9-b).
- Angle-Controller-only ANN implementation (Fig.8-c and Fig.9-c).
- All three Controller implementations into one large ANN (Fig.8-d and Fig.9-d).

The Matlab/Simulink implementations of those four cases are shown in Fig.8-a, Fig.8-b, Fig.8-c and Fig.8-d, respectively.

The simulation results of the implemented systems of Fig.8 are shown respectively in Fig.9-a to Fig.9-d. The simulations of Fig.9 show the results of the ANN implementation (purple trace) as compared to the original PI controller (yellow trace) overall system performance. The results shown in this case represent the trained neural network model of each case. The optimized ANN configuration for each of the cases was obtained by attempting to generate the best achievable result for the speed and position curves obtained from the corresponding ANN controllers as compared to the original tuned PI case obtained above and shown on the same graphs. Several oscillations exist in the different cases however they do not affect the rotational angle curve.
Fig. 8-a: Matlab/Simulink Implementation of ANN based Current Controller

Fig. 8-b: Matlab/Simulink Implementation of ANN based Speed Controller

Fig. 8-c: Matlab/Simulink Implementation of ANN based Position Controller
The all-neural controller has been implemented by entering the three feedback signals. The angular position is the integration of the angular speed similar to all the above cases. A unit delay is needed in the all-neural controller in order to separate the current step from the previous step in the Matlab/Simulink. The controller exhibits a delay of angle control behind the actual one implemented using the three PI controllers. The summary of all four ANN implementations is presented in Table 1. It is to be noted that the all-neural controller has three hidden layers and one output layer as compared to the three cases of only one ANN controller for angle, speed or current, which all have one hidden layer and one output layer.
Fig. 9-b: Matlab/Simulink Implementation of ANN based Speed Controller

Fig. 9-c: Matlab/Simulink Implementation of ANN based Position Controller

Fig. 9-d: Matlab/Simulink Implementation of ANN based Position/Speed/Current Controllers
<table>
<thead>
<tr>
<th>ANN Controller</th>
<th>Layer 1 Neurons</th>
<th>Layer 2 Neurons</th>
<th>Layer 3 Neurons</th>
<th>Layer 4 Neurons</th>
<th>Minimum Training LS Error</th>
<th>Training Epochs Needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Controller</td>
<td>5</td>
<td>1</td>
<td>---</td>
<td>---</td>
<td>0.002615</td>
<td>32</td>
</tr>
<tr>
<td>Speed Controller</td>
<td>30</td>
<td>1</td>
<td>---</td>
<td>---</td>
<td>0.002572</td>
<td>535</td>
</tr>
<tr>
<td>Angle Controller</td>
<td>30</td>
<td>1</td>
<td>---</td>
<td>---</td>
<td>0.005046</td>
<td>1000</td>
</tr>
<tr>
<td>Combined Controller</td>
<td>40</td>
<td>80</td>
<td>30</td>
<td>1</td>
<td>0.03935</td>
<td>200</td>
</tr>
</tbody>
</table>

**DC MOTOR ANN BASED STATE ESTIMATOR**

In order to further exploit the use of the ANN, the feedback speed signal was also estimated from the current and smoothed voltage data signal feedbacks. The voltage signal being the output of a DC to DC chopper circuit needs to be transformed into an averaged signal using a Butterworth 8th order low pass filter with cut-off frequency at 100Hz. The Matlab/Simulink block diagram of the system implemented with an estimator is given in Fig.10-a. The ANN used in this case uses 90 Neurons in the single hidden layer and 2 Neurons in the output layer in order to estimate the rotational speed (which requires another Butterworth low-pass 8th order filtering at a frequency of 100Hz) as well as the electromagnetic torque of the machine, which is not required in the case outlined in this research. Fig.10-b shows the simulation results of the motor output signals using the trained ANN estimator for the speed signal implemented in Fig.10-a.
The combination of the state estimator and the neural controller for the Position/Speed/Current was performed in this section where the Simulink block diagram file is shown in Fig.11-a with the simulation scope results shown in Fig.11-b. In this manner, the resulting system is composed of the DC motor and its drive electronics with only voltage and current feedbacks. The estimation of the speed is performed using the trained ANN of the last section. The controller used in this case is the same one obtained from the previous section, which exhibited the best performance so far. The resulting combination of the two above controllers (without further training and tuning) is presented here in Fig.11. The implementation and training of both controllers into one ANN is not performed in this paper and is the subject of a separate publication.

The second graph of Fig.11-b shows the actual motor speed curve with the corresponding desired PI controller curve. An error exists around the low angle values. This error is due mainly to the error in the estimation as compared to the small values of angle considered in these specific points. The current at the same instants of time spikes to large values. The treatment of such errors requires the retraining of both ANNs which is a very lengthy and complicated process since a different type of ANN would be required (a recurrent neural network).
The error in the actual motor speed is not largely manifested in the angle curve which is the last one of Fig. 11-b. The error in this case does not exceed 10% of the value of the desired curve given by the reference PI controller.

CONCLUSION

This paper presented the use of artificial neural networks (ANN) in order to replace the conventional PID controllers of a conventional angular-position/angular-speed/current controlled DC machine. The proposed system also incorporated the estimation of the motor states from its inputs in order to enable the sensor-less operation of the system without the
recurrence to speed feedback encoders which might be difficult in robotic applications. Both ANNs were used together in the same system to perform the control as well as the estimation concurrently. The concept presented in this paper can be applied to perform the sensor-less control of robotic actuators with the proper tuning of the ANNs used in the control and estimation processes. The combined tuning of both control/estimation ANN in the same process can be implemented and trained with other types of motors which would be the subject of further publications.

REFERENCES


[91] Dongshan Geng Kunyi Chen ; Yongchao Yang ; Shaowu Li ; Qing Ai, “A research of DC motor dual close-loop PID speed-tuning system on the basis of ABC Algorithm”, The 26th Chinese Control and Decision Conference (2014 CCDC), May 31 2014-June 2 2014, Changsha, China, pp. 3450-3454


