POZNAN UNIVERSITY OF TECHNOLOGY ACADEMIC JOURNALS
No 85 Electrical Engineering 2016

Przemysław SIWEK*

MODELLING THE NEURAL SPEED REGULATION SYSTEM OF PMSM MOTOR IN MATLAB - SIMULINK

To implement speed regulation algorithms of PMSM motor on real object the exact model encompassing imperfections of digital measurement and AC-DC-AC converter is required. The aim of the following paper is to present this kind of simulation model of the neural speed regulation system of PMSM motor in Matlab – Simulink. In the beginning the direct-quadrature transform and mathematical model of permanent magnet synchronous motor in *dq* coordinate system are shown. The paper shows speed measurement algorithm which ensure digital character of the output data. In the second section the neural speed controller trained on-line by backpropagation algorithm and the method for calculating gradient of error function are described. At the end of the paper simulation results validating proper behaviour of neural speed regulation system of PMSM motor are shown.

KEYWORDS: artificial neural networks, speed control, PMSM, motor drive model, direct-quadrature, speed measurement, backpropagation algorithm

1. INTRODUCTION

In today's industry sector we encounter a huge demand for highly precise and dynamic speed regulation of electrical drives. A significant part of control systems still consist of PID regulators. Their main disadvantage is the constancy of parameters, which lowers the quality of control for objects subjected to changes. In recent years a substantial increase in usage and development of adaptive and nonlinear controllers such as artificial neural networks (ANN) has been observed. The ANN learning algorithms can be divided into two categories: offline and online. Offline algorithms do not change their weights after the learning phase. The main advantage of such approach is ANN being immune to overtraining. This phenomenon occurs when a desired function, given by a training vector, is reflected overly precisely which results in network losing the ability to generalize. On the other hand, an obvious disadvantage of this method is its vulnerability toward any changes of object parameters. In the industrial reality, the most changeable parameter for electrical motor speed

* Poznan University of Technology.

control is the moment of inertia. This can be observed for example in drives in industrial manipulators, drives in feeding mechanisms of lathes, drives of coiling and reeling machines, as well as drives in machines used in paper manufacturing. When working with objects of such kind, the best solution is to use online algorithms, which optimize ANN weights with each regulation cycle. Even in case of total lack of information about controlled object, after some time the regulators response will come close to the optimum. The only danger one may face in such case is the possibility of overtraining this kind of regulator, but this issue will not be discussed in this article.

In order to analyze the characteristics of neural controllers and their susceptibility to overtraining, a detailed microprocessor control system model which encompasses imperfections of current conversion is required. The model takes into account application of power inverter (described in [11]) and digital speed measurement in the drive system. In the following article a model of such kind was described using algorithms implemented in simulations performed in Matlab - Simulink.

2. REGULATION SYSTEM MODEL

2.1. PMSM model

For mathematical descriptions of AC machines a method of space vectors created by K.P. Kovacs and J. Racz [6] is generally used. The method involves a system analysis based not on transient currents flowing through two fictitious windings but rather based on dynamic sinusoidal spatial waves [5]. This kind of approach simplifies the model and allows for analysis with tools of linear algebra only.

Permanent magnet synchronous motors (PMSM) are machines with permanent magnets instead of windings in the rotor. They are manufactured in a such way that their magnetic flux distribution in the air gap is sinusoidal. In reality it is rather difficult to maintain an ideal symmetry of magnetic circuit. Taking into account the asymmetry of magnetic circuit results in creating a model which magnetic flux values are dependent on the angle of rotor placement against stator A phase [11]. In order to avoid that we use direct-quadrature transformation, known as *dq* transform, which was created based on works of Park and Clarke [9, 4]. This operation transforms the coordinate system of the model from a stationary system connected with the stator A phase into a rotary system, whose real axis coincides with the rotor and the system rotates with its angular velocity [11, 9]. This transform can be given by following formula (1a):

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \cos(\theta) & \cos\left(\theta - \frac{2\pi}{3}\right) & \cos\left(\theta + \frac{2\pi}{3}\right) \\ -\sin(\theta) & -\sin\left(\theta - \frac{2\pi}{3}\right) & -\sin\left(\theta + \frac{2\pi}{3}\right) \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{1a}$$

$$\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \cos\left(\theta - \frac{2\pi}{3}\right) & -\sin\left(\theta - \frac{2\pi}{3}\right) \\ \cos\left(\theta + \frac{2\pi}{3}\right) & -\sin\left(\theta + \frac{2\pi}{3}\right) \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} \tag{1b}$$

After using it, state equations of PMSM [11] can be written as (2):

$$u_d = R_s i_d + \frac{d\psi_d}{dt} - p\omega\psi_q \tag{2a}$$

$$u_q = R_s i_d + \frac{d\psi_q}{dt} - p\omega\psi_d \tag{2b}$$

$$\psi_d = L_d i_d + \psi_f \tag{2c}$$

$$\psi_q = L_q i_q \tag{2d}$$

$$m = p\frac{2}{3}(L_d i_q - \psi_q i_d) \tag{2e}$$

where: R_s – stator resistance, $L_{d,q}$ – stator inductance in *dq* coordinate system, $\psi_{d,q}$ – stator flux in *dq* coordinate system, *p* – number of magnetic poles, ψ_f – permanent magnets flux, $i_{d,q}$ – stator current in *dq* coordinate system, $u_{d,q}$ – stator voltage in *dq* coordinate system, ω – rotor rotational speed, *m* – electromagnetic torque.

This model uses stator flux linkages in space vector, which can be removed by replacing them with stator currents. That allows to transform formulas from (2) into (3):

$$\frac{\mathrm{d}i_d}{\mathrm{d}t} = -\frac{R_s}{L_d} i_d + \frac{L_q}{L_d} p\omega i_q + \frac{1}{L_d} u_d \tag{3a}$$

$$\frac{\mathrm{d}i_q}{\mathrm{d}t} = -\frac{R_s}{L_q} i_q - \frac{L_d}{L_q} p\omega i_d - p\omega\psi_f + \frac{1}{L_q} u_q \tag{3b}$$

$$m = \frac{3}{2} p\left[\psi_f i_q + \left(L_d - L_q\right) i_d i_q\right] \tag{3c}$$

As long as a description using equations (2) or (3) can be useful for analytics, in order to perform simulations a signal model is required. To create such kind of model we need to transform the state equations in such a way, that the highest derivative is on the left side of the equation. Then the operations from state equations maybe presented as a block diagram. Thanks to this fact, one can calculate the unknown derivative by adding together components of the equation. The model, depicted in Figure 1, can be effectively implemented in Matlab – Simulink simulation environment.

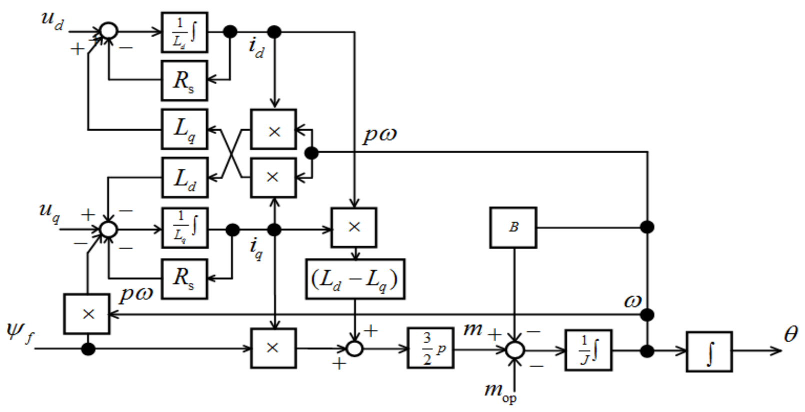

Fig. 1. Schemat blokowy PMSM w układzie współrzędnych *dq*

The object presented in Figure 1 can be used in PMSM simulation in a rotary coordinate system of a rotor. It should be mentioned here that voltage u_d and u_q have characteristics of a direct voltage. In order to achieve the possibility of supplying and measuring the object in stator coordinate system, in a way that the supply voltage and measured voltage would have a sinusoidal alternating character, one needs to use the transform (1a) before the motor model and a reverse transform (1b) after it.

2.2. Speed measurement

In speed control systems, an information about temporal (current) speed value of an object is required. Nowadays in drive systems to measure the mentioned value optical encoders are usually applied. They can be divided into incremental and absolute ones. This type of encoders do not differ significantly in terms of simulations, because one of their main characteristics is digital speed measurement. This means that the measurement is quantized and sampled. It is also characterized by a noise connected with fast and step changes of measured value. This is the feature which needs to be reflected in order to perform an accurate control system simulation.

In the work [3] a modification of M/T speed measurement method [1] was proposed. This new method considerably improved the quality of a measurement, ensures a constant sampling period and manages to work with overflowing buffers in case of low and high speed. It can be effectively used in microprocessor systems.

Measurement algorithm uses two types of counters. First one counts the number of encoder impulses in a particular, constant time T_g. The other one counts the number of clock impulses between the encoder impulses. After each measurement the value C(*k*) of a difference between closest encoder impulse and end of measurement period T_s is saved, along with the C(*k*-1) value being the same difference but in the measurement preformed for previous sample. Having this data one can calculate the speed using the following formula (4):

$$\omega_m = \frac{2\pi M}{N(T_s - [C(k) - C(k-1)]T_g)} \tag{4}$$

A pseudocode in Matlab notation, simulating the microprocessor speed measurement performed using abovementioned method is presented below.

Algorithm 1 Modified M/T speed measurement method

```
1: Inputs:
     $\mathrm{in\_args} = (\theta_m, t)$
2: for $t = 0 : T_{global} : T_{sim}$
3:   if $(\mathrm{mod}(t, T_g) == 0)$
4:     $C(k) = C(k-1) + 1$
5:   end
6:   if $(\mathrm{mod}(t, T_s) == 0)$
7:     if $(((T_s - [C(k) - C(k-1)]T_g) \sim= 0) \& \& (M \sim= 0))$
8:       $\omega_m = \frac{2\pi M}{N(T_s - [C(k) - C(k-1)]T_g)}$
9:     end
10:    $M = 0$
11:    $C(k-1) = C(k)$
12:  end
13:  if $(\theta_m \sim= \theta_{old})$
14:    if $(\theta_m > \theta_{old})$
15:      $M = M + 1$
16:    else
17:      $M = M - 1$
18:    end
19:    $C(k) = 0$
20:    $\theta_{old} = \theta_m$
21:  end
22: end

23: Outputs:
     $\mathrm{out\_args} = \omega_m$
```

3. SPEED CONTROL USING ANN TRAINDED ONLINE

Presented in the article neural controller can be used instead of traditional speed controllers. Its main advantage is online adaptation, which allows for adapting to any controlled object. It is possible because the algorithm training the network online does not use the learning vectors, i.e. values for optimal controller outputs, for which the objects achieves the desired state. Instead of this the controller aims to minimalize the error function given by the formula (5) through the learning algorithm. Algorithms of this type are called unsupervised learning algorithms. Because the function (5) achieves high values under step changes of reference signal, there is a need to feed the control system with a variable reference signal during the initial learning phase.

$$E = \frac{1}{2}\left(\omega_{\mathrm{ref}} - \omega_{\mathrm{m}}\right)^2 \tag{5}$$

3.1. Control system block diagram

In the control system, a field-oriented control is used, under assumption that the reference current in *q* axis is given by the speed controller and the current in *d* axis should be equal to zero. It enables the drive to maintain a constant power angle, which allows for it to achieve maximal value of torque even in case of stopped shaft. This, along with high dynamics, makes it possible for the drive to be used as a servomechanism. In this method a double transform between *dq* coordinate system and ABC is required. Reverse transform is used for measuring motor currents, and the simple transform transforms voltages given by current controller and feeds them in an ABC coordinate system to a Pulse Width Modulation (PWM) block. Using two transforms enables the application of linear PI current controllers, which parameters were chosen based on autotuning, which result was arbitrarily corrected in accordance with expert knowledge.

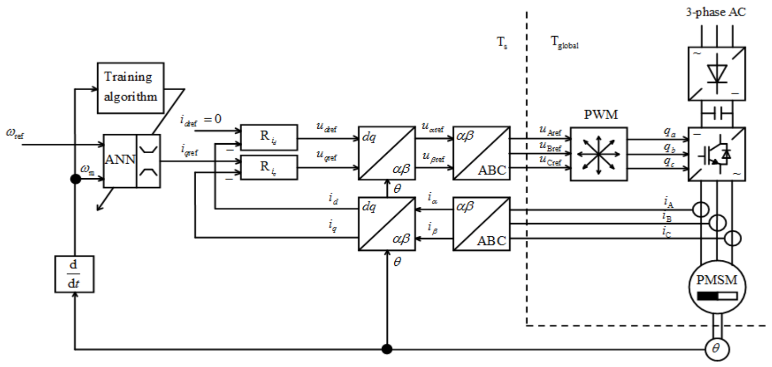

Fig. 2. Block diagram of a neural speed control of PMSM

Blocks described above are presented in Figure 2. In the article two periods of simulation were used. First, T_{global}, is used to simulate continuous elements and the second, T_s, is used to simulate the control system which operate with PWM frequency. These periods should differ at least by one order of magnitude. It is important for the means of calculating the gradient of the ANN error function.

3.2. ANN topology

In artificial neural networks there is not any generally defined optimal net topology for the given problem. Microprocessor control systems, however, introduce restrictions for the number of neurons that can be used by stating a requirement for calculating the network output in real time. The topology presented in this article was taken from the works of [8] which was based on the research conducted by [2]. The mentioned topology consist of 3 neurons in the hidden layer with an hyperbolic tangent activation function and one output neuron with a linear activation function. As an input vector, the neural control network in being fed with reference speed $\omega_{ref}(k)$, control error $e(k)$, measured speed $\omega_m(k)$ and measured speed $\omega_m(k–1)$ delayed by one sample.

$$\text{in} = [\omega_{ref}(k)\ e(k)\ \omega_m(k)\ \omega_m(k-1)]^T \tag{6}$$

The controller neural network described above is depicted in the Figure 3.

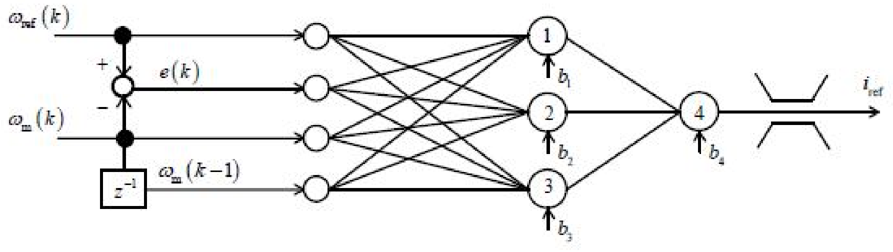

Fig. 3. Neural speed controller topology

3.3. Learning algorithm and calculating the gradient

In the controller, a standard backpropagation algorithm was used. The main obstacle for using the mentioned algorithm in online systems is calculating the exact value for gradient of the error function for ANN training. To determinate it, there is a requirement of knowing the desired output of ANN for given reference input signal. Usually, in the offline approach for backpropagation, the gradient calculation is realised by feeding the network with given input vector, determining the network outputs, subtracting the achieved result from the desired value and propagating the error gained by this method backward.

However, in real systems without observers and models of controlled object there is no information about required inputs. Thus, there exist no possibility of instant adaptation (during one sample) of network weights to the optimal values for controlling the given object. However, there is a possibility for the controller to gradually bring its weights values to the global optimum with each ensuing period of work. A rough draft of the algorithm presented in this article was given in [7]. The main idea behind it is to wait, each time, for the output of an object fed by input vector. The value of the gradient calculated this way is a value taken from the previous control period, although when faced with premise that the object dynamics is much lower than the dynamics of regulation system, this poses no problem for the backpropagation algorithm. The whole algorithm for training the ANN online is presented below in form of a pseudocode:

Algorithm 2 Backpropagation learning of ANN

```
 1: Inputs:
       $\overline{\mathrm{in\_args}} = \left(\omega_{\mathrm{ref}}, e, \omega_{\mathrm{m}}(k), \omega_{\mathrm{m}}(k-1)\right)$
 2: Initialize:
       $k = 0;\ \overline{\mathrm{W}_1} = \mathrm{randn}(3,4);\ \overline{w_2} = \mathrm{randn}(1,3);\ \overline{b_1} = \mathrm{randn}(3,1);\ b_2 = \mathrm{randn};\ \overline{\mathrm{in\_args}_{\mathrm{old}}} = \mathrm{zeros}(4,1)$
 3: for $t = 0 : \mathrm{T_s} : \mathrm{T_{sim}}$
 4:    if $(k == 0)$
 5:       $i_{\mathrm{ref}} = \overline{w_2} \cdot \tanh\left(\overline{\mathrm{W}_1} \cdot \overline{\mathrm{in\_args}}^{\mathrm{T}} + \overline{b_1}\right) + b_2$
 6:    end
 7:    if $(k == 1)$
 8:       if $(|i_{\mathrm{ref}} \le i_{\max}|)$
 9:          $\delta_{\mathrm{ANN}} = \omega_{\mathrm{ref}} - \omega_{\mathrm{m}}(k)$
10:          $\delta_{\mathrm{hid}} = \overline{w_2}^{\mathrm{T}} \delta_{\mathrm{ANN}} \left(1 - \tanh^2\left(\overline{\mathrm{W}_1}\,\overline{\mathrm{in\_args}_{\mathrm{old}}} + \overline{b_1}\right)\right)$
11:          $\overline{w_2} = \overline{w_2} + \eta \delta_{\mathrm{ANN}} \left(\tanh\left(\overline{\mathrm{W}_1}\,\overline{\mathrm{in\_args}_{\mathrm{old}}} + \overline{b_1}\right)\right)^{\mathrm{T}}$
12:          $b_2 = b_2 + \eta \delta_{\mathrm{ANN}}$
13:          $\overline{\mathrm{W}_1} = \overline{\mathrm{W}_1} + \eta \delta_{\mathrm{hid}} \overline{\mathrm{in\_args}_{\mathrm{old}}}^{\mathrm{T}}$
14:          $\overline{b_1} = \overline{b_1} + \eta \delta_{\mathrm{hid}}$
15:       end
16:       $i_{\mathrm{ref}} = \overline{w_2} \cdot \tanh\left(\overline{\mathrm{W}_1} \cdot \overline{\mathrm{in\_args}}^{\mathrm{T}} + \overline{b_1}\right) + b_2$
17:    end
18:    $\overline{\mathrm{in\_args}_{\mathrm{old}}} = \overline{\mathrm{in\_args}}^{\mathrm{T}}$
19:    $k = 1$
20: end
21: Outputs:
       $\mathrm{out\_args} = i_{\mathrm{ref}}$
```

4. SIMULATION RESULTS

In the following chapter the simulation results for the abovementioned model for PMSM neural speed control system are presented. The speed measurement system, learning algorithm, responses to reference speed step and load step were analysed. The results aim to show the correctness of the model.

Figure 4 shows speed measurement system response to a sinusoidal speed change in range of 0 to π for three different encoder resolutions. Algorithm output is characterized by step changes and noise, which is typical for a real digital measurement. Yet, standard deviation of the response is not a high value. While measuring a constant speed of 20 rad/s for a 12 bit resolution the standard deviation achieves the value of 0.054 rad/s; for a 16 bit resolution it achieves 0.046 rad/s and for 20 bit - 0.025 rad/s.

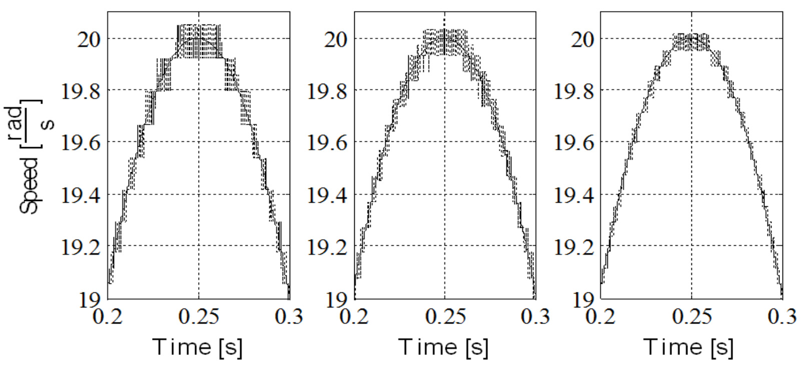

Fig. 4. Speed measured for different encoder resolution values for modified M/T method (from left: 12 bit, 16 bit, 20 bit)

Figure 5 presents the influence of the learning constant value on the neural network training. The system was fed by a periodic signal ranging from -20 to 20 rad/s. With the increase of learning constant η increases the oscillatory of controller response and the time of adaptation is getting shorter, which is natural for ANN.

The control system response under the load of 0.5 Nm to a single reference step causing motor reversion was also analyzed (Figure 6). In the picture we can see the phase currents and the magnification of a single phase current, which shows that the voltage inverter operate correctly. The object reaches a positive reference speed without overshooting and a negative reference speed with an overshoot of 6.4%. This kind of behaviour is related with a lack of integrator in the controller, which was discussed in [8]. Still it does not influence its application in ANN in form of a controller due to the learning algorithm minimizing the error function in formula (5).

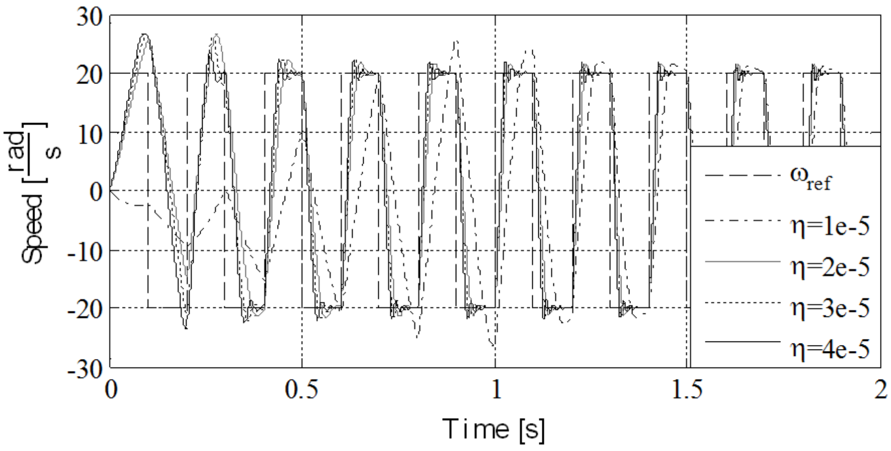

Fig. 5. The training of ANN, speed measured for different learning constants

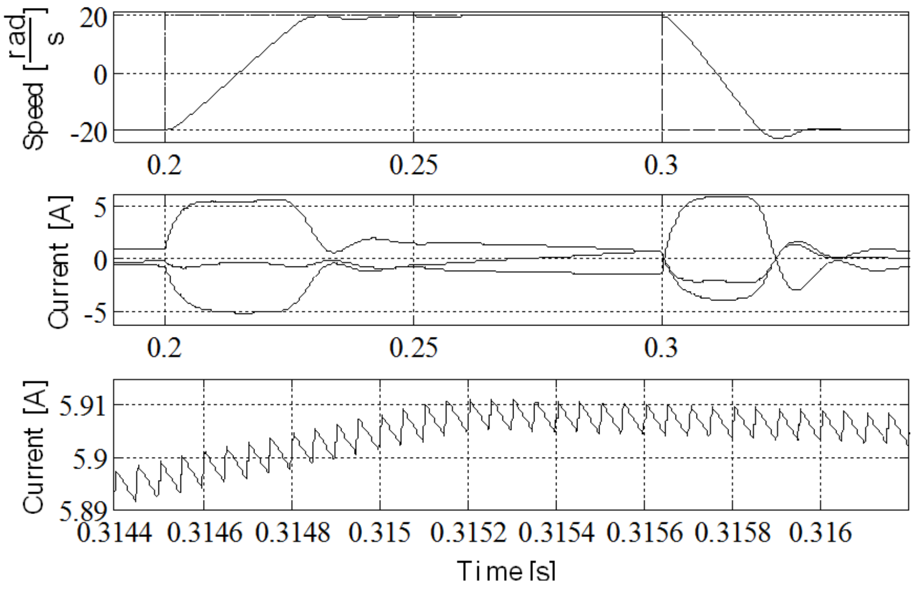

Fig. 6. Response of a system with load of 0.5 Nm to a step of reference speed given with i_{AB} currents and magnified single current depicting the inverter work

As presented (in magnification) in Figure 7, the response does not contain a steady state error, characteristic for traditional PD controllers. The picture depicts control system responses to a single reference speed step and load step from 0 to 0.5 Nm for four different values of learning constant.

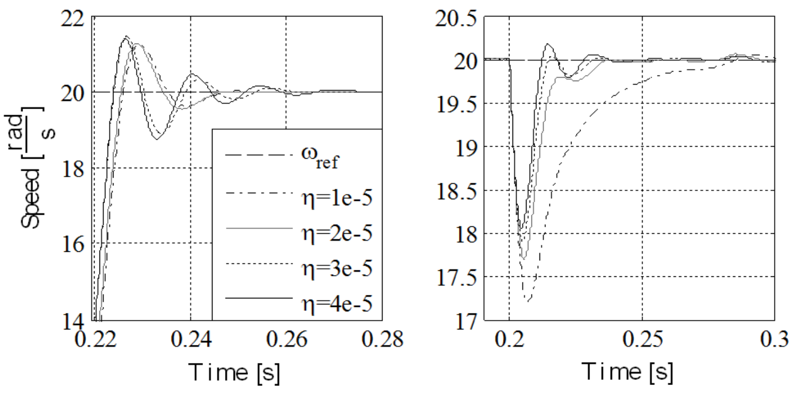

Fig. 7. Response to a step of reference speed from -20 to 20 rad/s for 4 values of learning constant (on the left), response to a step of load from 0 to 1 Nm for 4 values of learning constant (on the right)

5. CONCLUSIONS

In this paper, elements required to create a simulation model for PMSM neural speed control system trained online were presented. An analytic and signal motor model were described, along with transforms transforming coordinate systems from stationary to rotary ones. Also, an algorithm correctly modelling the real speed measurement and the backpropagation algorithm were given, the latter calculating precise value of error gradient for neural network training and modifying its weights online.

The achieved simulation results suggest that the model is correct. Measurement and current control work in a way similar to reality, which is reflected in the signals. Neural network learns faster with higher learning constant η. Yet, it is important to mention that increasing the value of η leads to an oscillating character of system response and increasing of the overtraining risk for ANN. An element that minimizes the error in researched system is the learning algorithm. Its main advantage is the ability of auto-adaptation of the controller. That enables the presented controller to be used even in case of lack of knowledge about the parameters of the controlled object.

REFERENCES

[1] Bonert R. Digital Tachometer with Fast Dynamic Response Implemented by a Microprocessor, IEEE Trans. Ind. Applic., 1983.
[2] Bose B. K., "Neural network applications in power electronics and motor drives-An introduction and perspective," IEEE Trans. Ind. Electron., Feb. 2007.

[3] Brock S., Zawirski K. Speed Measurement Method for Digital Control System, 9th EPE, 2001.

[4] Duesterhoeft W. C., Schulz M. W., Clarke E. Determination of Instantaneous Currents and Voltages by Means of Alpha, Beta and Zero Components, Transactions of the American Institute of Electrical Engineers, 1951.

[5] Ertan H. B., Uctug M. Y., Colyer R, Consoli A. Modern Electrical Drives, Springer, 2000.

[6] Kovacs K. P., Racz J. Transiente Vorgange in Wechselstrommachinen, Budapest, Hungary, Akad.Kiado, 1959.

[7] Noriega J. R., Wang H. A direct adaptive neural network control for unknown nonlinear systems and its application, American Control Conference, 1995.

[8] Pajchrowski T., Zawirski K., Nowopolski K. Neural Speed Controller Trained On-Line by Means of Modified RPROP Algorithm, IEEE Trans. Ind. Inf., 2015.

[9] Park R. H. Two Reaction Theory of Synchronous Machines, AIEE, N. Y., 1929.

[10] Rutkowski L., “Computational Intelligence Methodts and Techniques,“ Springer-Verlag, 2008.

[11] Zawirski K., Deskur J., Kaczmarek T. Automatyka Napędu Elektrycznego, Wydawnictwo Politechniki Poznańskiej, 2012, (in Polish).

(Received: 16. 02. 2016, revised: 5. 03. 2016)