BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 60, No. 3, 2012
DOI: 10.2478/v10175-012-0059-9

VARIA

Particle swarm optimization of artificial-neural-network-based on-line trained speed controller for battery electric vehicle

B. UFNALSKI* and L.M. GRZESIAK

Institute of Control and Industrial Electronics, Warsaw University of Technology, 75 Koszykowa St., 00-662 Warsaw, Poland

Abstract. The paper presents implementation of PSO (Particle Swarm Optimization) to ANN-based speed controller tuning. Selected learning parameters are optimized according to the control objective function. A battery electric vehicle is considered as a potential plant for an adaptive speed controller. The need for adaptivity in the control algorithm is justified by variations of a total weight of the vehicle. A sizable section of the paper deals with selection of a combined objective function able to effectively evaluate the quality of a solution.

Key words: electric vehicle, speed control, adaptive ANN controller, particle swarm optimization.

1. Introduction

The Computational Intelligence (CI) based approach to control and optimization problems in power electronics and drives has already proved to provide successful solutions for nonlinear or time-variant systems. We usually tap into CI if LTI (linear time-invariant) approximation is no longer feasible for the system under consideration. As far as a speed controller tuning is considered, in most cases, an electric drive with decoupled flux and torque control can be regarded as a first-order inertia or several first-order inertias connected in series if e.g. delays introduced by the digital control system or communication bus (if present in feedback path) should be taken into account due to their non-negligible contribution to the overall delay. In some continuous-time domain based approaches to control system synthesis more accurate approximation of the delay is used than truncating its Taylor series, namely the Padé approximant [1]. However, the truncation of a Taylor series after two initial terms is the most common procedure, because it produces relatively simple plant representation ready for well-established PID controller tuning procedures like Kessler's criteria or predictive schemes [2, 3]. On the other hand, many torque-controlled drives cannot be simplified to an LTI subsystem for the purpose of speed controller design task. One quite common reason is that they are significantly time-variant as a consequence of the resultant moment of inertia variations. This is for example the case if speed control loop has to be closed for an electric vehicle. The resultant moment of inertia seen by the control system varies significantly with number and weight of passengers and their luggage. The difference between the kerb weight and the gross one can be at the level of 50% in respect to the kerb one. Such a difference can be observed for a light A- or B-segment car as well as for a city bus, e.g. [4]. Moreover, speed control loop has to be stable for sudden change in inertia by a factor of one order of magnitude observed when a drive wheel loses traction.

It should be noticed that not all drive systems for electric vehicles are equipped with speed closed-loop controlled inverter. Some solutions incorporate only torque control loop and tend to mimic response of an internal combustion engine to a throttle pedal. Advantages and disadvantages of both approaches (i.e. speed plus torque control vs torque control) are widely discussed in the topical literature and are out of the scope of this paper.

In the case of this study, a drivetrain with one electric motor and a mechanical differential is assumed. Such a topology is quite popular in commercialized electric and hybrid passenger cars and city busses. The paper presents tuning procedure for an adaptive neural speed controller incorporated into this scheme. An evolutionary optimization algorithm was chosen to determine some key parameters of neural adaptive speed controller.

2. Nonlinear and adaptive speed controllers

Several techniques have been proposed to cope with the problem of inertia variations present in many commercial applications of speed-controlled drives. They can be divided into two main groups. One set of solutions assumes introduction of an inertia estimator into the system – this enables us to vary controller gains (e.g. of PI-type) according to the estimated moment of inertia (direct method) [5]. The second set of solutions takes advantage of introducing nonlinearity into the controller with intention to achieve some degree of insensitivity to variation of selected parameters. We sometimes refer to these methods as indirect ones, because the inertia is newer calculated explicitly – no inertia estimator is present in these schemes. Fuzzy logic (FL), artificial neural networks (ANN) and their combinations are among common tools widely used for digital implementation of PI-like nonlinear controllers. This nonlinearity can be static, i.e. determined during an off-line optimization procedure, or can be tuned continuously during normal operation of the drive in on-line mode. Examples of off-line and on-line trained controllers can be found in [6–11].

The ANN-based controllers offer straightforward capability of adaptation. A learning algorithm (a tuning procedure)

*e-mail: bartlomiej.ufnalski@ee.pw.edu.pl

can be kept active during regular operation of the drive [10, 11]. This in turn means that weights of the ANN will be changed automatically if the dynamics of a drive will change. Two crucial design decisions have to be made at this point: type of a learning algorithm and a method of determining parameters for this algorithm. It was already identified that the Rprop (resilient back-propagation) algorithm possesses properties that are of paramount importance as far as real-time code execution is needed. The algorithm takes into account only sign of the gradient. As a result, the training process is less sensitive to noise. Moreover, in many tests Rprop outperforms other first-order learning algorithms in terms of speed of convergence [12]. However, to maintain its high performance in a particular control system four training parameters have to be set, namely minimal and maximal step sizes, decrease and increase factors. Although there are some rough rules that can be followed, fine tuning is usually done by guessing and checking. Our idea is to eliminate this guess and check stage and replace it with a particle swarm optimization. Note that this does not mean that no expert knowledge will be needed. As it will be shown later on, common candidates for the objective function that do not require setting of any parameter (penalty factor), do not produce in this particular case good results. Nevertheless, a combined objective function with one or two penalty coefficients turns out to give satisfactory results.

3. Computational model

The discussed system is sketched schematically in Fig. 1. A more detailed structure of the adaptive ANN-based controller is shown in Fig. 2.

An Rprop modification known as the Rprop with weight-backtracking [12] was chosen in this study. For the sake of clarity its pseudocode is as follows:

```
*if* $\left(\frac{\partial E}{\partial w_{ij}}(k-1)\cdot\frac{\partial E}{\partial w_{ij}}(k)>0\right)$ *then* {
    $\delta_{ij}(k)=\min(\delta_{ij}(k-1)\cdot\eta^{+},\delta_{\max})$
    $\Delta w_{ij}(k)=-\mathrm{sgn}\left(\frac{\partial E}{\partial w_{ij}}(k)\right)\cdot\delta_{ij}(k)$
    $w_{ij}(k+1)=w_{ij}(k)+\Delta w_{ij}(k)$
}
*else if* $\left(\frac{\partial E}{\partial w_{ij}}(k-1)\cdot\frac{\partial E}{\partial w_{ij}}(k)<0\right)$ *then* {
    $\delta_{ij}(k)=\max(\delta_{ij}(k-1)\cdot\eta^{-},\delta_{\min})$
    $w_{ij}(k+1)=w_{ij}(k)-\Delta w_{ij}(k)$
    $\frac{\partial E}{\partial w_{ij}}(k)=0$
}
*else if* $\left(\frac{\partial E}{\partial w_{ij}}(k-1)\cdot\frac{\partial E}{\partial w_{ij}}(k)=0\right)$ *then* {
    $\Delta w_{ij}(k)=-\mathrm{sgn}\left(\frac{\partial E}{\partial w_{ij}}(k)\right)\cdot\delta_{ij}(k)$
    $w_{ij}(k+1)=w_{ij}(k)+\Delta w_{ij}(k)$
}
}
```

where E – objective function (cost function), w_{ij} – weight, $\delta_{\min}$, $\delta_{\max}$ – minimal and maximal allowed change of weight (absolute values), δ_{ij} – current weight change (absolute value), η^{-}, η^{+} – decrease and increase factors for δ_{ij}.

The cost function for ANN constitutes control task:

$$E_{ANN}(k)=\frac{1}{2}\left(\omega_m^{ref}(k)-\omega_m(k)\right)^2. \tag{1}$$

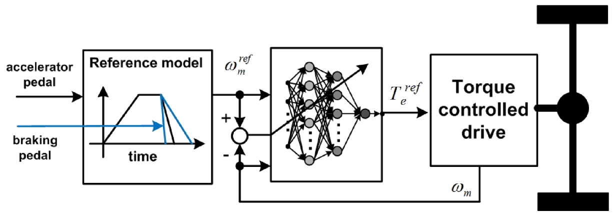

Fig. 1. Blok diagram of the control system

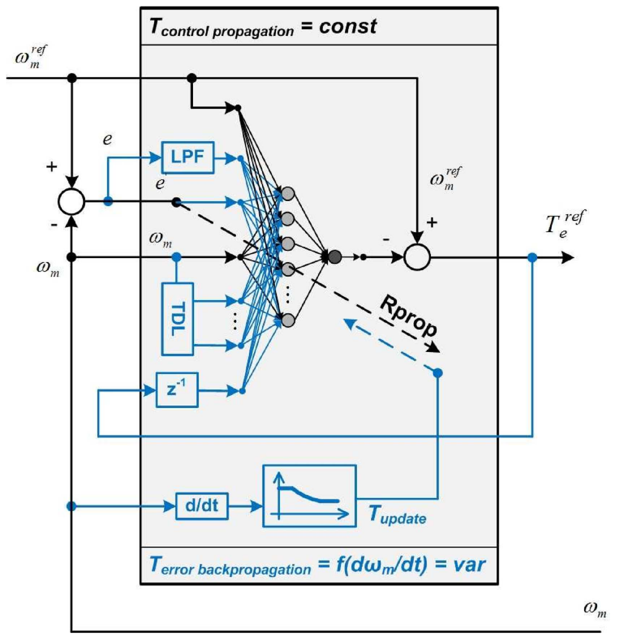

Fig. 2. Topology of the adaptive controller – basic realization shown in black, additional elements shown in blue

It was tested that the most crucial settings are $\delta_{\max}$, η^{-} and η^{+}. Other parameters like number of neurons, length of tapped delay lines (TDL) or $\delta_{\min}$ (usually set to 0 or very small positive) are easy to tune by guessing and checking. There are some tips on potentially working settings for η^{-}, η^{+}, e.g. $\eta^{+}=1.2$, $\eta^{-}=0.5$ and $\delta_{\max}=50$ are reported to deliver good results in many off-line benchmarks [12]. It was found out that in the case of an on-line trained controller for the discussed drive these settings are far from optimal. At the same time the performance of the drive is highly sensitive to these parameters and guessing a fairly good set of settings is usually time consuming. The idea is to turn this problem into numerical optimization problem.

4. Particle Swarm Optimization (PSO)

PSO is a stochastic optimization algorithm that shares many similarities with evolutionary computation techniques. The system is initialized with a set (here called swarm) of random candidate solutions (here called particles). Particles fly through the problem space. Their current speed vector is a result of global best solution and particle best solution found so far. Solutions are rated according to an objective function. A search path for each particle is not deterministic. There is a random factor that modifies in each iteration individual and social behaviour coefficients for each particle. The core of the basic algorithm is as follows [13]:

$$\begin{aligned} v_i(k+1) = c_1 \cdot v_i(k) + c_2 \cdot \text{rand}() \cdot \left(x_i^{best} - x_i(k)\right) \\ +c_3 \cdot \text{rand}() \cdot \left(x_{global}^{best} - x_i(k)\right), \end{aligned} \tag{2}$$

$$x_i(k+1) = x_i(k) + v_i(k+1), \tag{3}$$

where i – particle identification number, v_i – speed of the i-th particle, x_i – position of the i-th particle, x_i^{best} – best solution proposed so far by i-th particle, x_{global}^{best} – best solution found so far by the swarm, c_1 – explorative factor (inertia weight), c_2 – cognitive factor (individual influence), c_3 – social factor, rand() – uniform random number between 0 and 1.

There are numerous mutations of basic PSO [14]. However, there is no ultimate velocity-update rule. A set of modifications should be carefully chosen to suit the particular optimization problem. If rule (2) manifests unsatisfactory convergence speed or limited ability to find the optimum for the particular problem, some refinements can be added to the velocity-update rule or significant modification regarding information flow can be made, e.g.:

- inertia weight (explorative factor) can decrease with the number of iterations or can be the function of particle performance [15],
- constriction factor can be introduced (canonical PSO) as constant one or variable one [16],
- x_{global}^{best} can be redefined to represent best solution found so far by particle's neighbourhood limited by the distance (the radius of the neighbourhood is infinite in the rule as presented in (2)) or by the fixed number of neighbours (particles closest in distance) [17],
- particle can be attracted by every other particles in its neighbourhood (i.e. knowledge of x_i^{best} is distributed among all particles) [18],
- boundary conditions can be introduced with the help of absorbing, reflecting or invisible walls [19].

In our study the constricted PSO algorithm with equal cognitive and social coefficients is employed. The radius of the neighbourhood is assumed to be infinite. Three parameters of the Rprop are taken into account: $\delta_{\max}$, η^- and η^+. They are positive numbers where $\eta^- < 1$ and $\eta^+ > 1$. These boundary conditions are introduced to the search algorithm as absorbing walls. This gives the following velocity-update rule:

$$\begin{aligned} v_i(k+1) = \chi[v_i(k) + \frac{\varphi}{2} \cdot \text{rand}() \cdot \left(x_i^{best} - x_i(k)\right) \\ + \frac{\varphi}{2} \cdot \text{rand}() \cdot \left(x_{global}^{best} - x_i(k)\right)] \end{aligned} \tag{4}$$

where φ is chosen to be equal to 4.1 and the constriction factor is calculated using the formula:

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}. \tag{5}$$

It should be noted that (4) is a special case of (2). The constricted PSO is not a modification of the original algorithm. It only gives recipe on how to set c_1, c_2 and c_3 in (2). In our case each particle is a vector

$$x_i = \left[\eta^-, \eta^+, \delta_{\max}\right] \tag{6}$$

of candidate settings for the ANN-based adaptive controller. Absorbing walls are defined according to

$$\begin{cases} 0 < \eta^- < 1 \\ \eta^+ > 1 \\ \delta_{\max} > 0 \end{cases} \tag{7}$$

Other search directions are left unconstricted. Due to unknown upper limits for two directions (no clear physical limits) no maximal particle velocity was assumed in these directions. It is common practice to set velocity clamping $v_{\max} = x_{\max}$ if search space is constricted. Such clamping could be set for the first variable in our case. However, the constricted PSO with a constriction factor defined as constant (5) in many benchmark optimization problems does not require velocity clamping to deliver satisfactory speed of convergence. One can alternatively use modified strategies for the constriction factor, including time-dependent strategies and random effects [16].

5. Objective function

Choosing the appropriate control objectives for PSO is the most crucial decision to be made. There is a set of commonly used objective functions in control problems. Each objective function puts stress on (favours or penalizes) slightly different behaviour of the system. Most of them are meant to favour fast and asymptotic tracking of reference input. It usually takes expert knowledge supported by trial and error iterations to determine satisfactory objective function for a given control task. We have started with well-known integral performance indices of a general form [20]:

$$I_{p,q} = \int_0^\infty t^p \left|e(t)\right|^q dt, \tag{8}$$

where p and q are fixed numbers, and $e(t)$ denotes the control error. At first the $I_{0,1}$, $I_{0,2}$, $I_{1,1}$, $I_{1,2}$ and $I_{2,2}$ candidates were tested. They are known as the integral of absolute error (IAE), the integral of squared error (ISE), the integral of time multiplied absolute error (ITAE), the integral of time multiplied squared error (ITSE) and the integral of squared time multiplied error (ISTE), respectively. In our system none of the above indices have produced expected results. These indices favour fast response at the cost of speed overshoot and underdamped oscillations of a reference torque (Fig. 3).

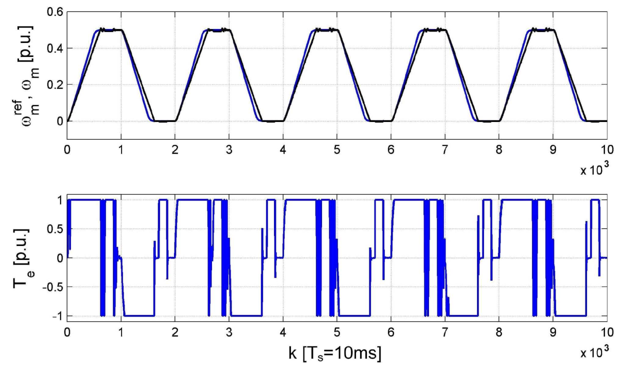

Fig. 3. System tuned using the ISE performance index

It should be noticed that these indices are examples of "non-parametric" ones. Once selected they do not require any parameter (e.g. penalty factors) tuning. This is important advantage, because one of our goals is to reduce number of steps that involve guessing and checking. This does not mean that experimenting with combined indices that require subjective choice of the weight(s) is unjustified. Even if number of weights in objective function is higher than original number of parameters to be found by optimization process (in our case three parameters shown in (6)), guessing a satisfactory set of weights could be far easier than direct guessing a good suboptimal solution. Four more weighted indices were tested [21]: the generalized integral of squared error

$$I_{GISE} = \int_0^\infty \left[e^2(t) + \alpha \dot{e}^2(t)\right] dt, \tag{9}$$

the integral of squared error and control effort

$$I_{ISECE} = \int_0^\infty \left[e^2(t) + \beta\left(u(t) - u_\infty\right)^2\right] dt, \tag{10}$$

the integral of squared error and derivative of control effort

$$I_{ISEDCE} = \int_0^\infty \left[e^2(t) + \gamma \dot{u}^2(t)\right] dt, \tag{11}$$

and the integral of squared error and squared overshoot

$$I_{ISEO} = \int_0^\infty \left[e^2(t) + \lambda e^2_{overshoot}(t)\right] dt, \tag{12}$$

where selection of α, β, γ and λ is subjective.

These experiments let us gather some expert knowledge on building the objective function (OF) for this particular problem. We have decided to combine (11) with (12), and to switch to $I_{2,2}$ (ISTE). The value of the OF is determined by running the model and gathering necessary data – no analytical evaluation of OF is involved, i.e. the plant can be black-boxed. This gives important flexibility in an OF formulation. Due to discrete nature of the controller and finite simulation time, integral is replaced with summation and infinity is replaced with simulation stop time (t_{STOP}). Positive scaling factor (sampling time) is omitted before sum symbol, because it does not affect optimization problem definition. This gives

$$I_{PSO} = \sum_{k=0}^{kT_s = t_{\mathrm{STOP}}} \left((kT_s)^2 e^2[k] + \gamma \dot{u}^2[k] + \lambda e^2_{overshoot}[k] \right), \tag{13}$$

where T_s stands for sampling time for the ANN-based controller. Further modifications of (13) have been undertaken to assess ANN learning process more effectively. First of all, the test reference speed and load torque vary periodically. A time resetting function is incorporated into I_{PSO} and two logical functions are introduced to suppress to zero selected (or all) terms in I_{PSO} for a given time intervals being the function of reference speed and test load torque. This can be written as:

$$I_{PSO} = \sum_{k=0}^{kT_s = t_{\mathrm{STOP}}} f_{START}[k]\left(\left(f_{RESET}(kT_s)\right)^2 e^2[k] \right. \tag{14}$$
$$\left. + f_\gamma(\omega_m^{ref}[k], T_{LOAD}[k], k)\gamma \dot{u}^2[k] + \lambda e^2_{overshoot}[k]\right),$$

where f_{START} is set to 0 during the first period of test (othervise 1), f_{RESET} resets time after each period of reference speed and f_γ neglects derivative of control effort (reference torque) in transient states of the electromagnetic torque initiated by load change or reference speed change. It is assumed that f_γ is set to 0 for a period of time equal to electromagnetic torque rise time. This cost function may appear complex, but in fact is very intuitive. We do not rate ANN performance at the beginning because the controller is initialized with random weights. The derivative of control effort is introduced into I_{PSO} in order to distinguish "good" solutions from solutions with chatter. However, punishing derivative of control signal during specific time interval after a change in load or reference speed is undesirable. It should be noticed that mentioned controller output behaviour cannot be effectively assessed using only $e[k]$ because of big time constant of the mechanical part of a vehicle.

6. Numerical results

A numerical verification of the proposed tuning procedure has been carried out using a hypothetical drive with inputs and outputs normalized to [-1,1] range. This reflects a common approach to ANN learning, which assumes normalization of all signals. Selected parameters related to ANN controller and its tuning are shown in Table 1.

Table 1
Selected parameters of the model

Parameter	Value
Sampling time for outer loop (speed loop)	10ms
Inertia increase for fully loaded vehicle	50%
δ_{min}	0.001
Number of particles	27
Number of iterations	75
Boundary conditions implementation	absorbing walls

Functions f_{START}, f_{RESET} and f_γ used in (14) are shown along with reference speed and disturbance torque in Fig. 4. Initial position of all particles is depicted in Fig. 5. The swarm after 25 iterations is shown in Fig. 6. The search is arbitrary stopped after 75 iterations (Fig. 7). In ideal case all particles should gather in one point. In our case this does not happen because of process and measurement noise included in the model. It could also happen that the area around found maximum of $-I_{PSO}$ is more like plateau than a sharp peak. In such a case the swarm may stay in constant move. From practical point of view these movements are unimportant if best solution found so far is satisfactory. Nevertheless, it is advisable to monitor variances of parameters being optimized (components of (6)). This will clearly indicate sensitivity of the objective function I_{PSO} to $\delta_{\max}$, η^- and η^+. The evolution of its variances in this particular case is shown in Figs. 8–10.

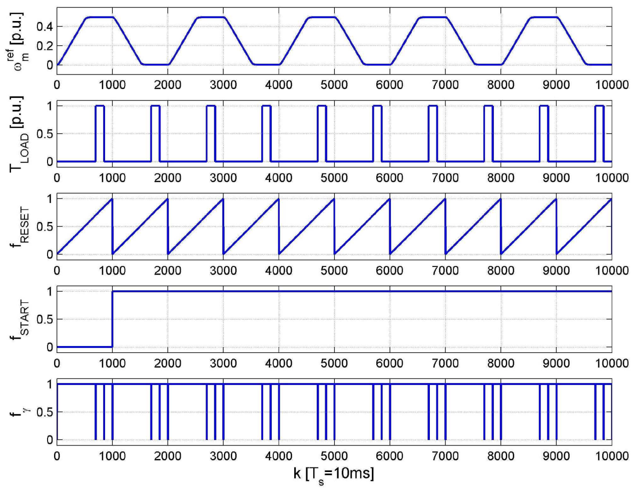

Fig. 4. Correlation between auxiliary switching functions and input signals

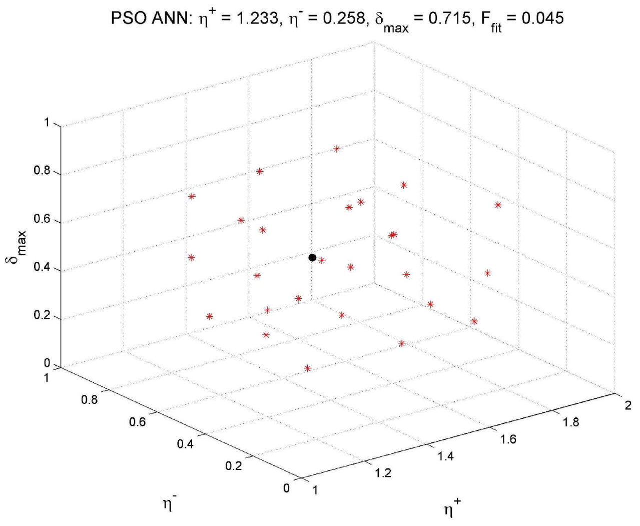

Fig. 5. Initial position of the swarm

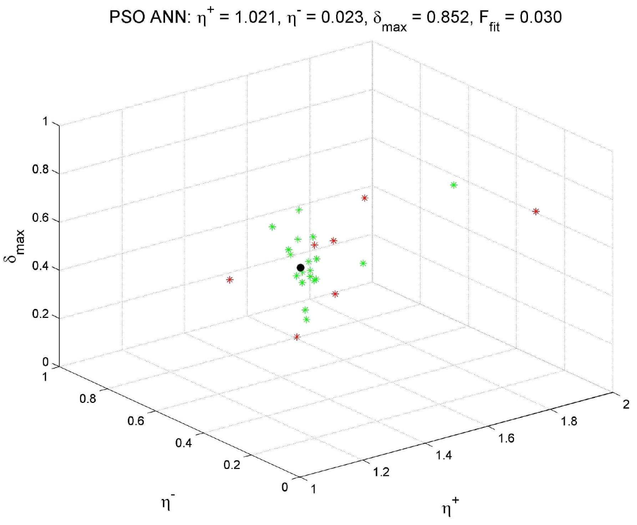

Fig. 6. The swarm after 25 iterations

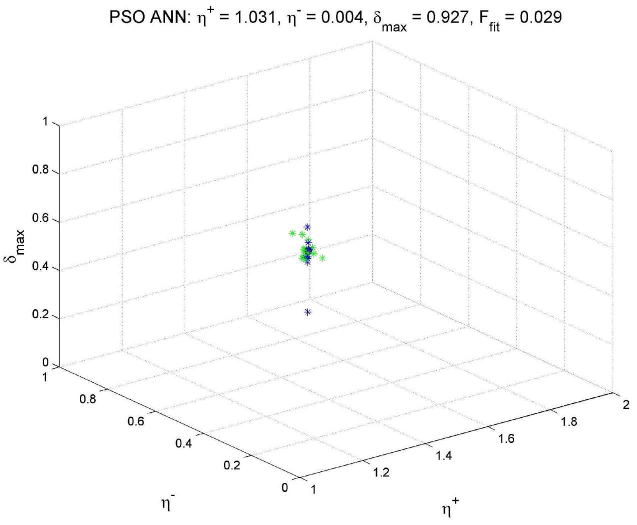

Fig. 7. The swarm after 75 iterations

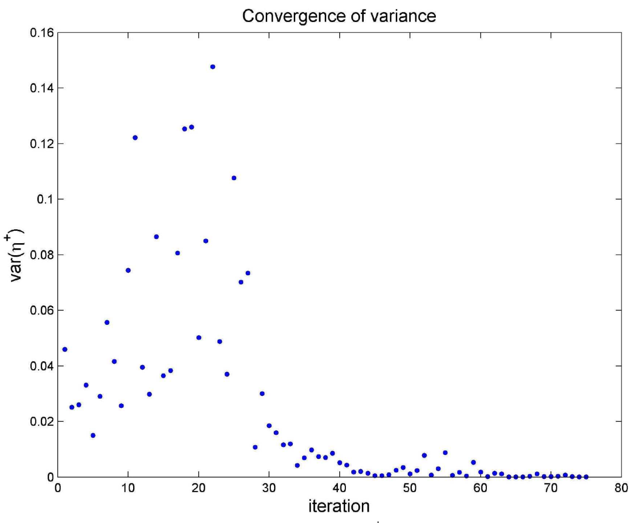

Fig. 8. Evolution of η^+ variance

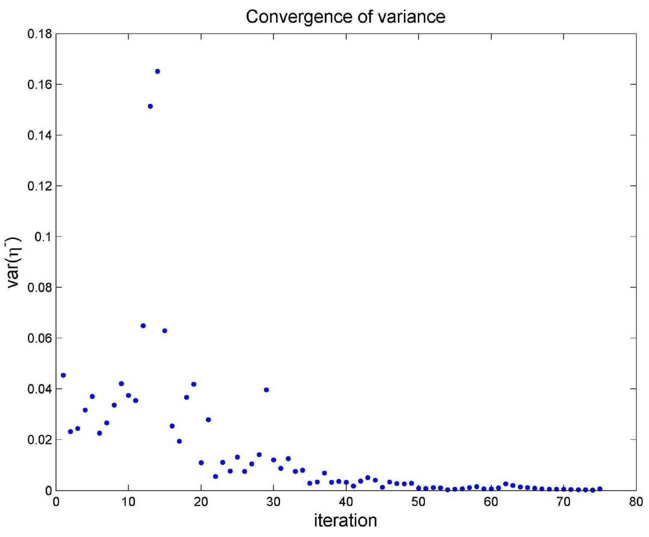

Fig. 9. Evolution of η^- variance

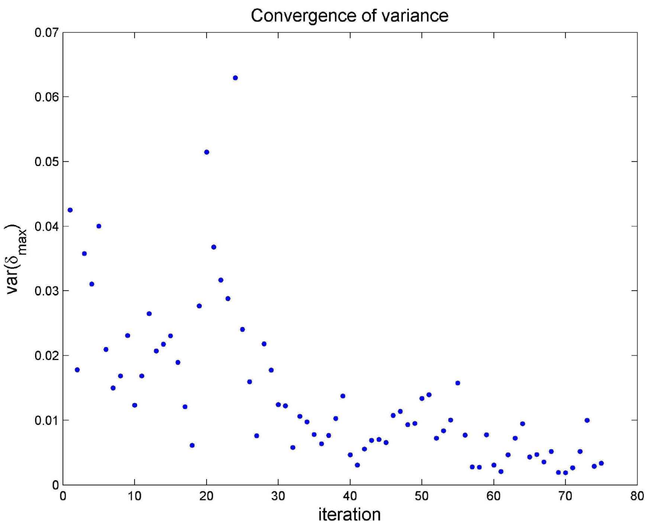

Fig. 10. Evolution of $\delta_{\max}$ variance

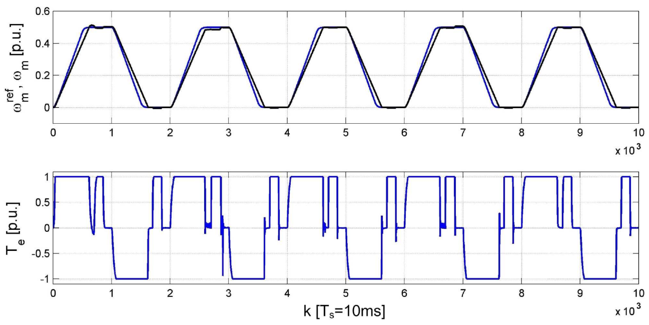

Fig. 11. Performance of the drive tuned according to the I_{PSO} index

It can be concluded that $\delta_{\max}$ can be set more freely than η^-or η^+. Performance of the drive tuned by the PSO with ranking according to the performance index (14) is illustrated in Fig. 11. An increase in moment of inertia at the level of 50% is shown in Fig. 12. For comparison purposes the performance of the system if η^- and η^+ are set according to the recommendations made for off-line approximation problems (i.e. $\eta^+ = 1.2$ and $\eta^- = 0.5$) is presented in Fig. 13.

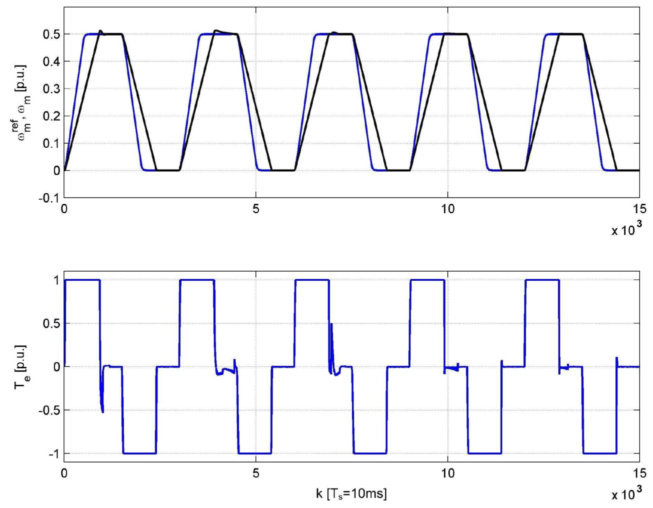

Fig. 12. Performance of the drive tuned according to the I_{PSO} index after 50% increase in moment of inertia

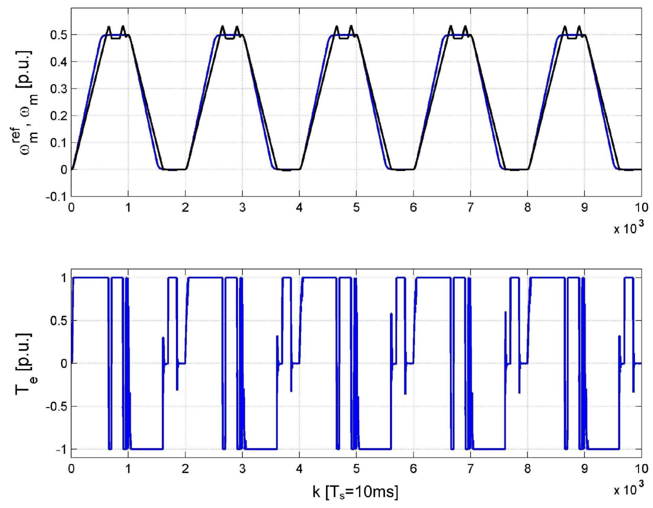

Fig. 13. Performance of the drive tuned according to the general recommendation for off-line Rprop parameter settings

7. Conclusions

The on-line trained ANN has been used as a speed controller for an electric vehicle to introduce better insensitivity to variations in vehicle mass. The Rprop learning algorithm has been selected by virtue of its speed and good ability to cope with noise. To fully benefit from this solution at least three learning parameters have to be properly tuned. There is no universal direct tuning procedure for this kind of problem. It was decided to rearrange this problem into more intuitive one. The objective function for the control task has been formulated and the PSO with absorbing walls has been implemented to solve the multivariable optimization problem. Due to the nature of the ANN learning process special measures have been taken to effectively assess performance of the controller. Obtained results indicate usefulness of this approach.

Acknowledgements. The research work has been supported by the Electrical Drive Division statutory funds for 2012.

REFERENCES

[1] R. McCann, A. Le, and D. Traore, "Model predictive control for time-delay compensation of a switched reluctance motor drive in smart building applications", *Proc. IEEE IAS Industry Applications Society Annual Meeting* 1, 1–4 (2008).

[2] V. Blasko, V. Kaura, and W. Niewiadomski, "Sampling of discontinuous voltage and current signals in electrical drives: a system approach", *IEEE Trans. on Industry Applications* 34 (5), 1123–1130 (1998).

[3] T. Nussbaumer, M. Heldwein, G. Gong, and J. Kolar, "Prediction techniques compensating delay times caused by digital control of a three-phase buck-type PWM rectifier system", *Proc. IEEE IAS Industry Applications Conf. -+ Fortieth IAS Annual Meeting* 2, 923–927 (2005).

[4] *Solaris Vehicles Technical Data*, www.solarisbus.pl (2012).

[5] J.-W. Choi, S.-C. Lee, and H.-G. Kim, "Inertia identification algorithm for high-performance speed control of electric motors", *IEE Proc. -+ Electric Power Applications* 153 (3), 379–386 (2006).

[6] T. Pajchrowski and K. Zawirski, "Application of artificial neural network to robust speed control of servodrive", *IEEE Trans. on Industrial Electronics* 54 (1), 200–207 (2007).

[7] T. Pajchrowski and K. Zawirski, "Robust speed and position control based on neural and fuzzy techniques", *Proc. Eur. Conf. on Power Electronics and Applications* 1, 1–10 (2007).

[8] T. Pajchrowski and K. Zawirski, "Application of fuzzy logic techniques to robust speed control of PMSM", *Proc. EPE-PEMC Power Electronics and Motion Control Conf.* 1, 1198–1203 (2008).

[9] T. Orlowska-Kowalska, K. Szabat, and M. Dybkowski, "Neuro-fuzzy adaptive control of the IM drive with elastic coupling", *Proc. EPE-PEMC Power Electronics and Motion Control Conf.*, 2211–2218 (2008).

[10] L.M. Grzesiak, V. Meganck, J. Sobolewski, and B. Ufnalski, "On-line trained neural speed controller with variable weight update period for direct-torque-controlled AC drive", *Proc. 12th Int. EPE-PEMC Power Electronics and Motion Control Conf.* 1, 1127–1132 (2006).

[11] L. Grzesiak, V. Meganck, J. Sobolewski, and B. Ufnalski, "Genetic algorithm for parameters optimization of ANN-based speed controller", *Proc. IEEE EUROCON Int. Conf. on Computer as a Tool* 1, 1700–1705 (2007).

[12] C. Igel and M. Hsken, "Empirical evaluation of the improved Rprop learning algorithms", *Neurocomputing* 50, 105–123 (2003).

[13] J. Kennedy and R. Eberhart, "Particle swarm optimization", *Proc. IEEE Int. Conf. on Neural Networks* 4, 1942–1948 (1995).

[14] F. Van den Bergh, "An analysis of particle swarm optimizers", *Ph.D. Dissertation*, the University of Pretoria, Pretoria, 2002.

[15] M. Senthil Arumugam and M.V.C. Rao, "On the performance of the particle swarm optimization algorithm with various inertia weight variants for computing optimal control of a class of hybrid systems", *Discrete Dynamics in Nature and Society* 1, 1–17 (2006).

[16] L. Bui, O. Soliman, and H. Abbass, "A modified strategy for the constriction factor in particle swarm optimization", *Progress in Artificial Life* 1, 333–344 (2007).

[17] J. Pugh, L. Segapelli, and A. Martinoli, "Applying aspects of multi-robot search to particle swarm optimization", *Proc. 5th Int. Conf. on Ant Colony Optimization and Swarm Intelligence (ANTS'06)* 1, CD-ROM (2006).

[18] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better", *IEEE Trans. on Evolutionary Computation* 8 (3), 204–210 (2004).

[19] J. Robinson and Y. Rahmat-Samii, "Particle swarm optimization in electromagnetics", *IEEE Trans. on Antennas and Propagation* 52 (2), 397–407 (2004).

[20] M.S. Tavazoei, "Notes on integral performance indices in fractional-order control systems", *IFAC J. Process Control* 20 (3), 285–291 (2010).

[21] H. Unbehauen, "Controller design in time-domain", *Control Systems, Robotics and Automation*, in *Encyclopedia of Life Support Systems (EOLSS)*, EOLSS Publishers Ltd, London, 2002.