Piotr KOZIERSKI, Marcin LIS, Karol ONOSZKO, Adam GULCZYŃSKI

Poznan University of Technology, Faculty of Electrical Engineering

The use of Particle Filter and Neurocontroller for Current Converter Control

Abstract. *This paper presents the use of particle filter and neurocontroller to the power converter current control task. The article shows the neural network learning method, the impact of particle filter to the quality of the reference signal tracking and the accuracy of the state variables estimation by changing particle filter parameters. A method of the noise variance auto-tuning in particle filter was proposed in order to improve the quality of estimation.*

Streszczenie. *W pracy zaprezentowano użycie filtru cząsteczkowego i neurokontrolera do zadania sterowania prądem przekształtnika energoelektronicznego. Pokazano metodę uczenia sieci neuronowej, wpływ wykorzystania filtru cząsteczkowego na jakość nadążania za sygnałem referencyjnym oraz dokładność estymacji stanu przez zmianę parametrów filtru cząsteczkowego. Zaproponowano także sposób autostrojenia wariancji szumów w filtrze cząsteczkowym w celu polepszenia jakości estymacji. (**Wykorzystanie filtru cząsteczkowego i neurokontrolera do zadania sterowania prądem przekształtnika**).*

Keywords: particle filter, neurocontroller, learning method, the noise variance autotuning.
Słowa kluczowe: filtr cząsteczkowy, neurokontroler, metoda uczenia, autostrojenie wariancji szumów.

doi:10.12915/pe.2014.02.50

Introduction

H-bridge circuits can be used as an independent voltage and current sources, which find use in various fields (e.g., in the generation of the reference signal, compensating harmonics of non-linear loads or medicine) [1]. In the systems where the dynamics is high, the influence of the inverter in the control circuit cannot be neglected. It can not be treated as unity gain. This hinders the selection of the optimal control by means of analytical methods, as the mathematical model of the inverter is hard to obtain, mostly it is approximated (e.g. using a delay element, or a first order transfer function with delay [2]). Using a neural network for this task can avoid many disadvantages associated with the unknown part of the object and maintain a satisfactory control performance.

In this paper, a method to control the H-bridge inverter by using a neural network in the presence of high measurements noise is presented. In order to eliminate noises, a particle filter is used, which acts as a state observer.

Object

The test object is a single-phase power electronics H-bridge inverter, current control type. Its scheme is shown in Figure 1. It consists of the following parts: a transistor bridge, LC filter, almost purely inductive load, voltage and current meters (not included in Fig. 1). Object parameters:

- $L_1 = 250\ \mu H$,
- $C = 15\ \mu F$,
- $R_0 = 0.1\ \Omega$,
- $L_0 = 1\ mH$,
- $U_{in} = 100\ V$,
- $f_{sw} = 12.5\ kHz$,

where: f_{sw} means switching frequency of transistors (transistors are switched bipolar – pairs $G1$ or $G2$).

The assumption has been made that the system is operating in a very noisy environment. Therefore, all measurements are subject to errors, which can be described by the formula:

$$y = x + n \tag{1}$$

where: y – measured value, x – true value (state variable), n – measurement noise, with normal distribution: $N \sim (0, (5/6)^2)$.

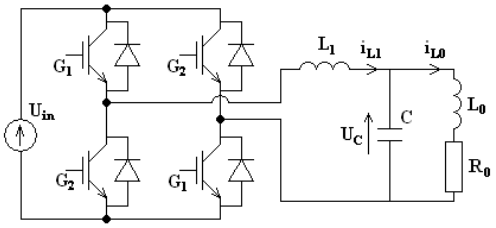

Fig.1. Scheme of the inverter with LC filter and load RL

Control scheme

For the research, two control block diagrams have been used which are shown in Figures 2 and 3. These diagrams comprise the object (shown in Figure 1), a neural network (neurocontroller) and the particle filter. The last two are described in the following sections. The main purpose is to generate the control signal u, so that the object can keep up with the reference signal r_k with the least static and dynamic error.

The generated signal u is modulated by the PWM with a constant carrier frequency, and then fed to the transistors in the form of a rectangular waveform signal ($G1$ and $G2$).

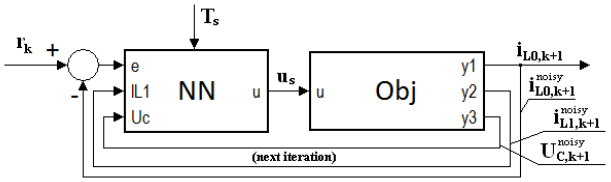

Fig.2. Block diagram of control scheme with neural network

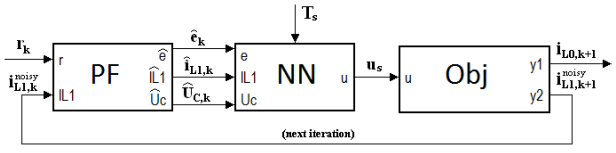

Fig.3. Block diagram of control scheme with neural network and particle filter

Neurocontroller

A neurocontroller (neural network – NN) is a feed-forward type [3]. It consists of 6 neurons in the hidden layer and one neuron in the output layer. Hyperbolic tangent is used as an activation function (tansig), due to its

monotonicity and a smooth transition saturation. In the hidden layer it has a form of $\mathrm{tansig}(x/\alpha)$ and in the output layer $\mathrm{tansig}(x/\beta)$, where: x – tansig input, α and β – correction coefficients (initial value equal to unity).

One of the most important parameters of the network is T_s. This is the time at which the network takes the inputs, performs its calculations and updates the output. It is the same as in the case of switching frequency and it is equal to $80\ \mu s$. It is not allowed to change network output more frequently as the switching frequency of the transistors f_{sw} could vary.

The implementation of structure and input signals to the network are shown in Figure 4. The network takes a total of nine signals, of which six are delayed (by T_s or $2 \cdot T_s$). Delayed signals are: error between the reference value r and current i_{L0}, coil current i_{L1} and the capacitor voltage U_C.

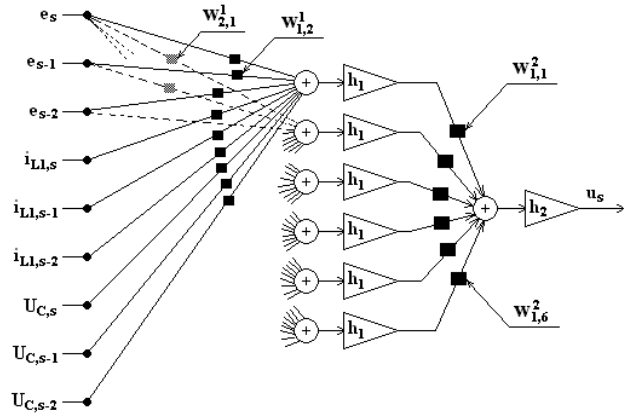

Fig.4. The structure of the neurocontroller; $w^a_{b,c}$, where w – weight value, a – number of layer, b – neuron number in layer a, c – number of weight at that neuron, h – activation function

Learning method

The initial value of network weights (there are 60 of them) were randomly selected from range $(-1, 1)$. Network learning has been conducted using Adaptive Interaction method [4, 5]. This method is based on the idea of approximation of object using a constant (simulation assumed that the value to be 1) and the selection of weights to minimize error e shown in Figure 3.

The learning process was as follows:

- as reference signal has been set square wave with an amplitude of $50\ A$ and frequency of $50\ Hz$,
- adaptation was performed until non-fading oscillation were achieved, with an average value close to the amplitude of the reference signal, as shown in Figure 5 (input signals were noisy),
- calculations have been performed in the hidden layer in order to search for the greatest value of neurons before the activation function and then assigned to the parameter α ($\alpha = 7000$),
- in the case of the output layer, it has been noticed that the sum of signals from the hidden layer neurons is, in the most cases, saturating between the value of $+1$ and -1, or oscillating around some mean value, as shown in Figure 6. From the behavior of the output signal, one can get the impression that the neurocontroller is an ON/OFF regulator. To counteract this phenomenon, parameter β has been changed ($\beta = 0.1$),
- for such selected network parameters further studies have been performed (results in a separate chapter).

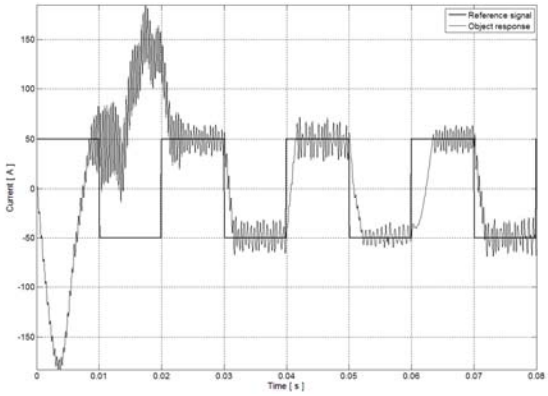

Fig.5. On-line learning of neural network

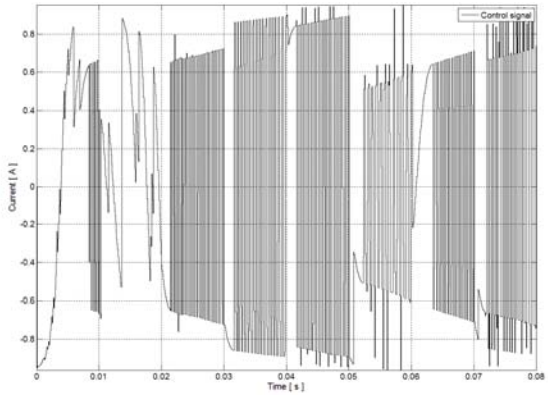

Fig.6. Shape of the control signal u before changing parameters α and β

Particle filter

Particle filter (PF) is based on the Bayes Filter, whose task is to estimate the probability density function (PDF). The principle of operation can be represented by the expression

$$\underbrace{p(x_k \mid Y_k)}_{\text{Posterior}} = \frac{\overbrace{p(y_k \mid x_k)}^{\text{Likelihood}}\overbrace{p(x_k \mid Y_{k-1})}^{\text{Prior}}}{\underbrace{p(y_k \mid Y_{k-1})}_{\text{Evidence}}} \tag{2}$$

where x_k is a state variable at the k-th time step, Y_k is a set of observations (measurements) at time steps between 1 and k. It can be assumed that the object is a Hidden Markov Model, which means that the values of the state variables depend only on the values at the previous time step

$$\begin{aligned} x_k &= f_{k-1}(x_{k-1}, v_{k-1}, u_{k-1}) \\ y_k &= g_k(x_k, n_k) \end{aligned} \tag{3}$$

Each state variable x_k is hidden, and the only information about it can be derived from the measured values y_k and the knowledge about the functions g_k and f_k.

Derivation of the equation (2), and a more extensive discussion about the Bayes Filter can be found in [6, 7].

Particle Filter (PF) is one of the possibilities of Bayes Filter implementation, wherein the set of randomly selected particles (drawn from this probability function) approximates the continuous posterior $p(x_k|Y_k)$ [8]. In this approach, each particle has a certain value x^i and the weight q^i, which can be written as [9]

$$\hat{p}(x_k|Y_k) = \sum_{i=1}^{N} q_k^i \cdot \delta(x - x_k^i) \tag{4}$$

where δ is the Dirac delta, and k is the time step number.

Based on the strong law of large numbers it can be stated that the number of particles N tend to infinity imply that posterior is equal to the expression (4)

$$p(x_k|Y_k) \overset{N\to\infty}{=} \hat{p}(x_k|Y_k) = \sum_{i=1}^{N} q_k^i \cdot \delta(x - x_k^i) \tag{5}$$

Through the use of particles, relatively complex calculations based on the probability density functions can be represented by simple operations, which, however, must be calculated N times. Therefore, it is possible to perform parallel processing, which causes the algorithm implementation that occurs more and more likely on FPGA – they allow algorithm to improve operating speed up to a million times [10]. This is especially important for multidimensional objects, in which the computational complexity grows exponentially with the dimension of the task [11].

However, despite of the large computational requirements, particle filter can be used to strongly nonlinear systems [12] which models are not differentiable and also in cases where there is a density of other distributions than Gaussian (Kalman filter and its extensions – EKF and UKF – may only be used in Gaussian distribution models) [13].

Thanks to its versatility, PF may be used in a variety of problems, such as the parametric estimation problems [14], video object tracking [15] and the robot localization problems [16].

The first PF algorithm together with a necessary step – resampling – was proposed in [17] by Gordon and is called Bootstrap Filter (BF). In many sources it is also called the SIR algorithm (Sequential Importance Resampling) as it relates to Importance Sampling method (it is described in detail in [18]), but it should be noted that the BF is a special case of the general SIR method.

BF method can be represented by algorithm:

1. Initialization. Set an initial values x^i_1 and weights $q^i_1 = {}^1/_N$; $k=1$.
2. Prediction. Draw N particles based on importance density $p(x_k|x^i_{k-1})$.
3. Update. Calculate particle weights according to $q^i_k \propto p(y_k|x^i_k)$.
4. Normalization. Scale weights, so that their sum are equal to 1.
5. Resampling. Draw N particles from resulting posterior.
6. Estimation. Calculate parameter based on particle values.
7. End of iteration. Increase time step $k=k+1$, go to step 2.

Transition model $p(x_k|x^i_{k-1})$ and the adjustment model $p(y_k|x^i_k)$ (also called measurement model), based on the knowledge of the object structure, the knowledge of the transition and adjustment noises PDF (also called system and measurement noises), are given.

The presented algorithm is the easiest to implement, however, there have been proposed many modifications so far, such as Particle Filtering with Elite Particles Mean Shift [19] or Gaussian Particle Filter [20].

Point 5 of the algorithm still requires an explanation, namely, the resampling, which is a very important part of the particle filter. SIS algorithm is deprived of this step, which causes the degeneration of the particles (one particle has a weight close to unity, and the rest of the particles – have the weights close to zero) [6].

Resampling is a random selection of N new particles based on posterior (the particle values drawn in step 2 of the algorithm and the weights calculated and normalized in steps 3-4), and giving them new weights – equal to ${}^1/_N$ for each particle.

There have been invented a few different resampling methods. The multinomial algorithm with a careless implementation may have a very high computational complexity. One can also find a residual resampling [21] or stratified and rejection resamplings [22]. The authors of this article recommend a systematic resampling.

This method is as easy to implement as multinomial resampling, and its biggest advantage is linear computational complexity [23]. It is based on dividing the entire range $(0;1)$, from which the draw occurs, into N equal intervals when j-th particle is drawn from the range

$$\left(\frac{j-1}{N}; \frac{j}{N}\right) \tag{6}$$

Systematic resampling algorithm is shown below [24]

1. „Prepare discrete cumulative distribution function $S_k^{1:N}$ based on particle weights, so that $S_k^1 = q^1{}_k$ and $S_k^N = 1$.
2. Set initial variable value $j=1$.
3. For $i=1,...,N$ perform steps 4-6.
   4. Draw a random value from partial uniform distribution $d \sim U({}^{i-1}/_N; {}^i/_N)$.
   5. As long as $S_k^j < d$ increase variable $j=j+1$.
   6. Remember the drawn value ${}^*x^i{}_k = x^j{}_k$.
7. Replace the old set of particles by saving values $x_k = {}^*x_k$, set new weights for $i=1,...,N$: $q^i{}_k = {}^1/_N$."

(7-8) shows the particle filter equations.

$$\mathbf{x}_k = \begin{bmatrix} r_{k-1} - x_{4,k-1} \\ x_{2,k-1} + v_{k-1} \\ x_{2,k-1} n_C + x_{5,k-1} n_D - x_{3,k-1} d_A - x_{6,k-1} d_B \\ x_{2,k-1} n_A + x_{5,k-1} n_B - x_{4,k-1} d_A - x_{7,k-1} d_B \\ x_{2,k-1} \\ x_{3,k-1} \\ x_{4,k-1} \end{bmatrix} \tag{7}$$

$$y_k = x_{2,k} + n_k \tag{8}$$

where $\mathbf{x}_k$ is the state vector, $x_{1,k}$ is the error value at the k-th time step, $x_{2,k}$ is the current value i_{L1}, $x_{3,k}$ is the voltage value U_C, $x_{4,k}$ is the current value i_{L0}. $x_{5,k}$, $x_{6,k}$ and $x_{7,k}$ are the first 3 state variable values delayed by one sample. This has been done in such a way that the particle filter describes a hidden Markov process, so all the values of the state variables depend only on the value of the previous time step (3).

The values of n_A, n_B, n_C, n_D, d_A and d_B are calculated on the basis of the connections and the parameters of individual circuit elements (using zero-order hold method). It should be noted that the estimated values of the error e and voltage U_C are calculated (from the object structure given by constants n_A, n_B, n_C, n_D, d_A and d_B) based on estimated current value i_{L1}. v_k is the value of the transition noise, and n_k is the value of adjustment noise (both are Gaussian). It is very difficult to analytical implementation of H-bridge,

because of the transition noise which has such a value that the particles are randomly drawn from the transition model (step 2 algorithm PF) in order to "search" the correct value of current i_{L1}. Based on the maximum possible dynamics of the system (the control signal u=1), the variance has been designated and its value has been set to $\sigma^2_v = {}^1/_{121}$. In the case of adjustment noise the variance was chosen according to measurement noise (simulation parameter) $\sigma^2_n = {}^{25}/_{36}$.

Simulation results of NN (with and without PF)

The simulation experiment assumes that the neural network has been trained to the noisy signals as it was previously shown. Figures 7-9 present the response of the object on the reference square-wave signal.

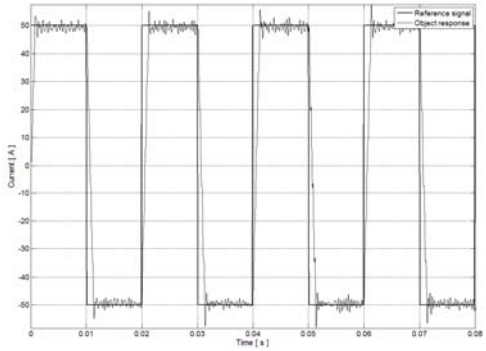

Fig.7. Operation of the control system shown in Figure 2

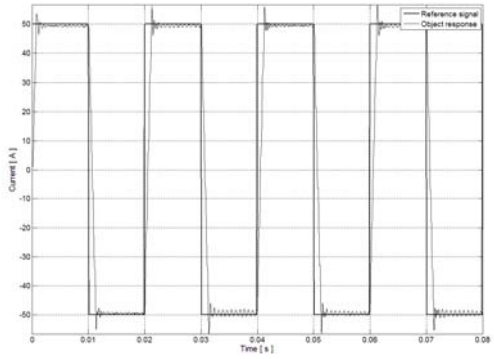

Fig.8. Operation of the control system shown in Figure 3

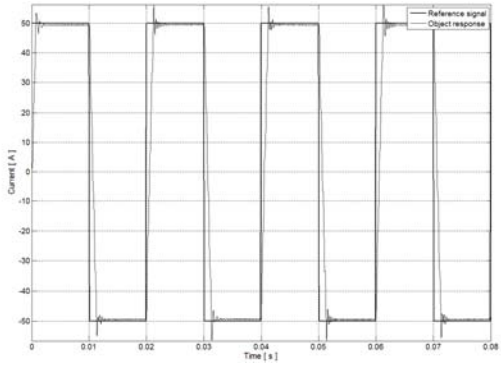

Fig.9. Operation of the control system with ideal case (the signals collected by the neural network are noise free)

Initially, in order to compare the quality of responses MSE index have been used:

$$R_1 = \frac{1}{k} \cdot \sum_k \left(r_k - i_{L0,k}\right)^2 \tag{9}$$

However, it is not possible to use it to compare the dynamic states that have occurred in response. Introduced complement to the quality index:

$$R = R_1 + \frac{C_0}{k} \cdot \sum_k \left| i_{L0,k} - i_{L0,k-1} \right| \tag{10}$$

where: C_0 – normalizing constant with a value of $1 \cdot e^{-8}$. The results of such a constructed quality index are shown in Table 1.

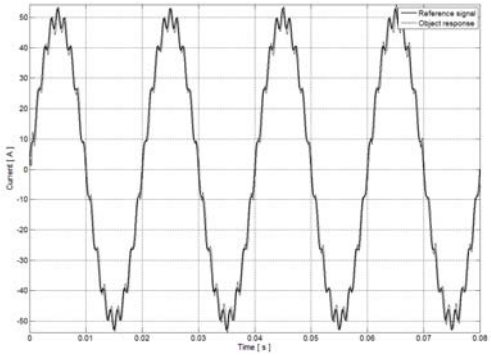

Fig.10. The system response to the reference signal of sine with 17-th harmonic for circuit with particle filter

Table 1. Values of the quality index R (10)

r_k	NN with noise	NN with noise and PF	NN without noise
Square-wave	525.70	483.56	479.74
Sine-wave with 17-th harmonic	146.74	126.65	125.86

Simulations results of PF and its modifications

The first set of simulations was performed to show the quality of tracking each of the state variables based on the number of particles in the filter.

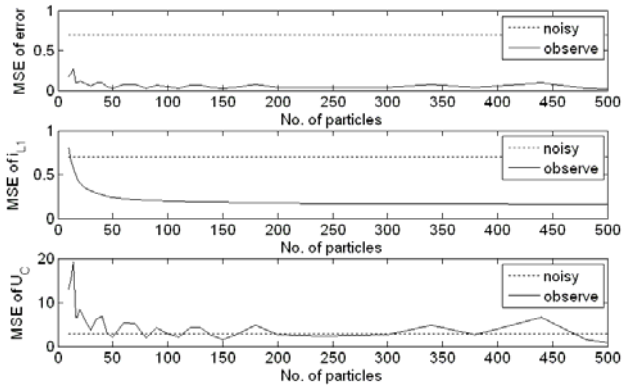

Fig.11. Quality of state variables observability

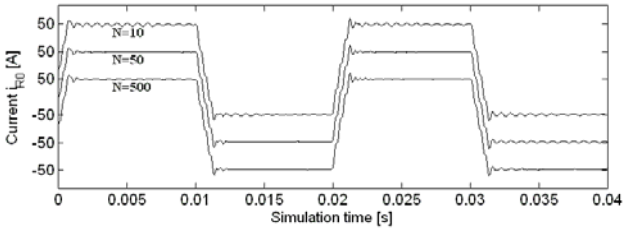

Fig.12. Chart of output system signal for 3 cases: N=10, N=50 and N=500

Each simulation is related to system from Figure 3. The reference signal is assumed to be a square wave with an amplitude of 50 A and a frequency of 50 Hz, the simulation time T_{sim}=0.04 s, and the simulation step T_k=1e-7 s.

Figure 11 shows the quality of the observation signals passed to neurocontroller, and Figure 12 presents three waveforms for different values of the particles number N.

It can be seen that while the observation of the error and the current i_{L1} are satisfactory and in most cases better than the noisy signal, therefore U_C voltage signal quality differs significantly from the other two. This is related to a delay of current i_{L1} observation towards the true values (see Fig. 13).

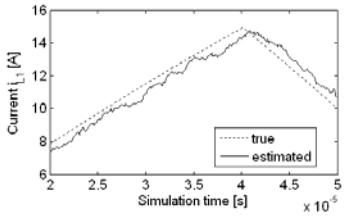

Fig.13. The true and the estimated values of current i_{L1} – part of the waveform

To improve the tracking quality, algorithm has to be modified in the adjustment noise variance – the arbitrarily selected value was too big.

Therefore the simulations have been performed in which the variance σ^2_n was scaled. The number of particles assumed to N=50. Simulation time and the sample period remain unchanged. The simulation results are shown in Figure 14.

The results have proven that the best value of the scaling factor is about 0.1. In addition, Figure 15 shows a comparison of the tracking quality, between the case with and without scaled adjustment noise variance.

It can be concluded that by scaling the variance σ^2_n one can significantly reduce the number of particles while maintaining a good tracking quality.

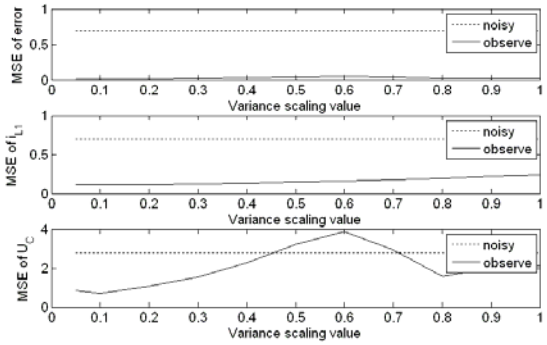

Fig.14. Quality of state variables observability with varying adjustment noise variance

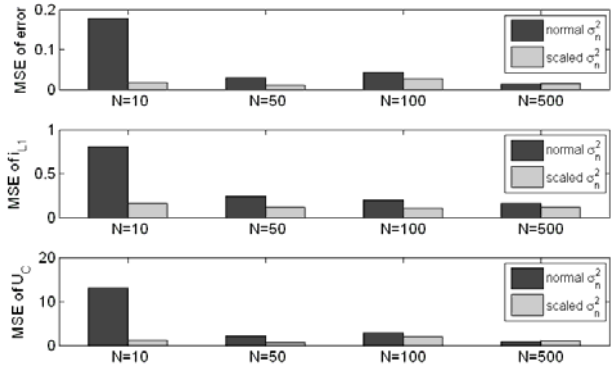

Fig.15. Comparison between case with and without scaled variance of adjustment noise

Auto-tuning of PF noises

The proposed method which improves the obtained results by simple scaling of the variance of the adjustment noise (Fig. 14-15) works well, however this approach, on the one hand, is inconvenient (need to determine for which value the best set of the state variables can be gained), on the other, is impossible to do in case of on-line control.

Therefore, an approach have been proposed, in which the noise variances (in PF model) could adapt itself by depending on the obtained results during the operation. The idea of auto-tuning particle filter is not new, but usually it refers only to the change in the particles number N [25, 26].

Rules of changing variances are shown in Table 2.

Table 2. Rules changes in the variance σ^2_v and σ^2_n.

Premise	Conclusion
Many particles have a weight close to the maximum	Too high σ^2_n
Significant weight values only in a small number of particles	Too low σ^2_n
Small weights of outermost particles	Too high σ^2_v
Large weights of outermost particles	Too low σ^2_v

This change of the PF parameters causes settling of the both values to (approximately) constant level, as shown in Figure 16.

Figure 17 shows a comparison of the basic method (without scaling the variance), the method with reduced adjustment variance and the method using auto-tuning. As expected, manually set a specific value of variance gives the best results.

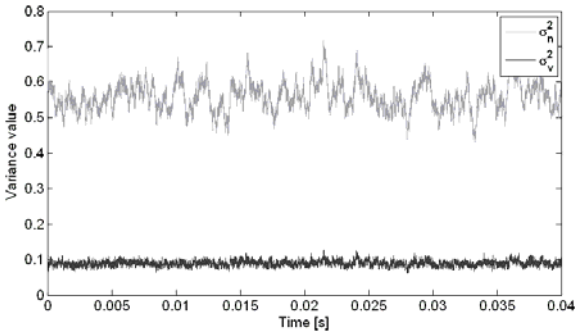

Fig.16. Change of variance values during simulation. σ^2_n starts from about 0.694; σ^2_v starts from about 0.008

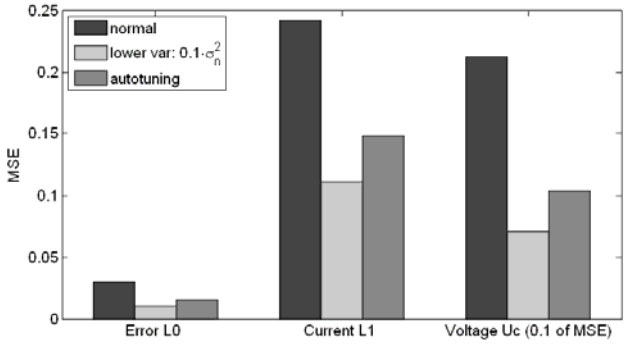

Fig.17. Compare the performance of PF for different cases of changes in variances

Summary

This paper realized highly nonlinear control dynamic object, which is the power electronics converter in the H-bridge configuration. Neural network controller allows to treat the object as a linear (frequency range up to 2 kHz) in the context of tracking reference signal with an acceptable dynamic and static error (maximum to 2%).

The following behavior has been observed while using neural network. If the network learns to operate with rectangular reference signal it is going to be suitable to control different reference signal shapes. Static and

dynamic errors are almost identical to those of the rectangular signal. Besides, the working point (the amplitude of the reference signal) is virtually irrelevant in terms of static errors. If the amplitude of $50\ A$ has static error of 2%, for $150\ A$ it will also be 2%.

Simulations have been also carried out on the network learned in the absence of noise (the idea of learning is as described earlier). The quality index for the respective systems differs from one another in a noisy environment tests only 1%. It follows that the network has well-learned behavior of the object.

Based on the simulations, it can be concluded that the tracking quality of the state variables affect the accuracy of the control (Fig. 12). It is also presented that the number of particles has an impact on the obtained results, but the number needs not to be very high (Fig. 11).

Proposed approach, in which both variances in PF evolved each iteration, depended on the values and weights of the particles. It was thus obtained worse efficiency than the manual setting case, nevertheless this algorithm can work on-line.

REFERENCES

[1] Porada R., Lis M., Sterowanie energoelektronicznym źródłem prądu z zastosowaniem sieci neuronowych, *Poznan University of Technology Academic Journals. Electrical Engineering*, Issue 76 (2013), 63-70

[2] Porada R., Control of independent power electronic voltage sources with application of Kalman filter, *Przegląd Elektrotechniczny*, No 7 (2009), 154-158

[3] Osowski S., Sieci Neuronowe do Przetwarzania Informacji, *Oficyna Wyd. Politechniki Warszawskiej*, Warszawa (2006)

[4] Brandt R. D., Feng L., Adaptive interaction and its application to neural networks, *Information Sciences*, Vol. 121, Issues 3–4 (1999), 201-215

[5] Lis M., Wpływ czasu odpowiedzi neurosterownika na jakość regulacji, *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska*, No. 3 (2012), 15-18

[6] Kozierski P., Lis M., Filtr cząsteczkowy w problemie śledzenia – wprowadzenie, *Studia z Automatyki i Informatyki*, Vol. 37 (2012), 79-94

[7] Candy J.V., Bayesian signal processing, *WILEY*, New Jersey (2009), 36-44

[8] Thrun S., Burgard W., Fox D., Probabilistic robotics, *MIT Press*, Cambridge, MA (2005), 67-90

[9] Doucet A., Godsill S., Andrieu C., On sequential Monte Carlo sampling methods for Bayesian filtering, *Statistics and Computing*, No. 10 (2000), 197-208

[10] Mountney J., Obeid I., Silage D., Modular Particle Filtering FPGA Hardware Architecture for Brain Machine Interfaces, *Conf Proc IEEE Eng Med Biol Soc.* (2011), 4617-4620

[11] Sutharsan S., Kirubarajan T., Lang T., McDonald M., An Optimization-Based Parallel Particle Filter for Multitarget Tracking, *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 48, No. 2 (2012), 1601-1618

[12] Simon D., Optimal State Estimation, *WILEY--INTERSCIENCE*, New Jersey (2006), 461-484

[13] Chen H., Liu X., She C., Yao C., Power System Dynamic State Estimation Based on a New Particle Filter, *Procedia Environmental Sciences*, Vol. 11, Part B (2011), 655-661

[14] Doucet, A., Tadić, V. B., Parameter estimation in general state-space models using particle methods, *Annals of the institute of Statistical Mathematics*, Vol. 52, No. 2 (2003), 409-422

[15] Chang C., Ansari R., Khokhar A., Multiple Object Tracking with Kernel Particle Filter, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (June 2005), Vol. 1, 566-573

[16] Woo J., Kim Y-J., Lee J., Lim M-T., Localization of Mobile Robot using Particle Filter, *SICE-ICASE International Joint Conference* (2006), 3031-3034

[17] Gordon N.J., Salmond N.J., Smith A.F.M., Novel approach to nonlinear/non-Gaussian Bayesian state estimation, *IEE Proceedings-F*, Vol. 140, No. 2 (1993), 107-113

[18] Doucet A., Johansen A.M., A Tutorial on Particle Filtering and Smoothing: Fifteen years later, *handbook of Nonlinear Filtering,* No. 12 (2009), 656-704

[19] Zhong S., Hao F., Hand tracking by particle filtering with elite particles mean shift, *Proceedings on Japan-China Joint Workshop on Frontier of Computer Science and Technology*, (2008), 163-167

[20] Kotecha J.H., Djurić P.M., Gaussian Particle Filtering, *IEEE Trans Signal Processing*, Vol. 51, No. 10 (2003), 2592-2601

[21] Launay T., Philippe A., Lamarche S., On particle filters applied to electricity load forecasting, *arXiv* preprint, arXiv: 1210.0770 (2012)

[22] Murray L., Lee A., Jacob P., Rethinking resampling in the particle filter on graphics processing units, *arXiv* preprint, arXiv: 1301.4019 (2013)

[23] Arulampalam S., Maskell S., Gordon N., Clapp T., A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking, *IEEE Proceedings on Signal Processing*, Vol. 50, No. 2 (2002), 174-188

[24] Kozierski P., Lis M., Królikowski A., Gulczyński A., Resampling – essence of particle filter, Nowe trendy w naukach inżynieryjnych 3, *CREATIVETIME*, Vol. 1 (June 2013), 174-185.

[25] Straka O., Simandl M., Particle Filter with Adaptive Sample Size, Kybernetika, Vol.47, No. 3 (2011), 385-400

[26] Soto A., Self Adaptive Particle Filter, In IJCAI (July 2005), 1398-1406

***Authors**: M.Sc. Eng. Piotr Kozierski, Poznan University of Technology, Faculty of Electrical Engineering, Institute of Control and Information Engineering, ul. Piotrowo 3a, 60-965 Poznań, E-mail: piotr.kozierski@gmail.com; M.Sc. Eng. Marcin Lis, Faculty of Electrical Engineering, Institute of Electrical Engineering and Electronics, ul. Piotrowo 3a, 60-965 Poznań, E-mail: mail.dla.studenta@gmail.com; M.Sc. Eng. Karol Onoszko, Poznan University of Technology, Faculty of Electrical Engineering, Institute of Control and Information Engineering, ul. Piotrowo 3a, 60-965 Poznań, E-mail: karol.onoszko@doctorate.put.poznan.pl; M.Sc. Eng. Adam Gulczyński, Faculty of Electrical Engineering, Institute of Electrical Engineering and Electronics, ul. Piotrowo 3a, 60-965 Poznań, E-mail: adam.gulczynski@put.poznan.pl.*