POZNAN UNIVERSITY OF TECHNOLOGY ACADEMIC JOURNALS
No 100 Electrical Engineering 2019

DOI 10.21008/j.1897-0737.2019.100.0013

Rafał SZCZEPAŃSKI[*], Tomasz TARCZEWSKI[*]
Lech M. GRZESIAK[**]

EFFICIENCY ANALYSIS OF PARALLEL COMPUTING APPLIED TO AUTO-TUNING OF STATE FEEDBACK SPEED CONTROLLER FOR PMSM DRIVE

Nowadays the simulation is inseparable part of researcher's work. Its computation time may significantly exceed the experiment time. On the other hand, multi-core processors are common in personal computers. These processors can be used to reduce computation time by using parallel computing on multiple cores. The most popular software applied to simulate behavior of the plant is MATLAB/Simulink. A single simulation of Simulink model cannot be computed by multiple cores, but there are many engineering problems, that require a multiple simulation of the same model with different parameters. In these problems, the parallel computing can be employed to decrease the overall simulation time. In this paper the parallel computing is used to speed-up the auto-tuning process of state feedback speed controller for PMSM drive. In order to obtain the optimal coefficients of the controller, an Artificial Bee Colony optimization algorithm is employed.

KEYWORDS: parallel computing, Artificial Bee Colony, PMSM, state feedback controller, MATLAB/Simulink.

1. INTRODUCTION

Over the last few year multicore processors have become common in personal computers, but usage of them by programs is not trivial due to shared memory, scalability, synchronization, deadlock, race condition and difficult debugging process. There are many application programming interfaces (APIs) for parallel programming, the most known are: Open Multi-processing (OpenMP) [1], Message Passing Interface (MPI) [2], POSIX Threads (Pthreads). Each of these APIs have pros and cons, some of them are oriented for parallel computing and some are oriented for parallel tasking. One on the main disadvantage of these APIs is additional workload. Sometimes adding additional variable or condition in parallel program has consequences, that need to be taken

[*] Nicolaus Copernicus University
[**] Warsaw University of Technology

into account. Since the researcher's work requires a lot of modifications in model during tests, the most popular software used by researchers, which is MATLAB/Simulink offers Parallel Computing Toolbox [3]. It is based on parallel CPU computing using MPI and parallel GPU computing using CUDA. This toolbox allows to use parallel computing without directly programming in these APIs. It provides functions that allows to take advantage of multicore processor or graphics card e.g. asynchronously run a Simulink model, run *for* loop parallel, parallel multiply matrices. In this paper Parallel Computing Toolbox has been used for auto-tuning space-state feedback speed controller (SFC) for PMSM drive.

Permanent magnet synchronous motor (PMSM) are commonly used in industrial robots, CNC machines, electrical and hybrid vehicles, air conditioning application [4, 5]. The main reasons of its wide range of application are: compact construction, relatively high efficiency and high reliability [6]. The most commonly used control of PMSM drive is based on cascade of PI regulators [7], due to simple implementation and tuning process, while state-space feedback controller offers a better dynamical behavior and better disturbance compensation, but its tuning process is difficult [8]. Well-known methods of SFC tuning are: (i) pole-placement and (ii) linear-quadratic optimization. The first method requires expert knowledge for proper location of the poles, while the second method requires selection of the diagonal matrices **Q** and **R**. For complex control system both methods are not trivial and commonly requires trial-and-error approach, which is time consuming. Another approach is application of optimization algorithms to select coefficients of controller [9]. Recently Artificial Bee Colony algorithm [10], which is nature-inspired algorithm, has been used to select diagonal elements of matrices **Q** and **R** for linear-quadratic optimization.

2. STATE FEEDBACK SPEED CONTROLLER FOR PMSM

Synthesis process of state feedback speed controller for PMSM requires knowledge of state-space description of PMSM fed by voltage-source inverter (VSI). Linearized model of PMSM has the following representation in *d-q* reference frame [11]:

$$\frac{d\mathbf{x_i}(t)}{dt} = \mathbf{A_i}\mathbf{x_i}(t) + \mathbf{B_i}\mathbf{u_i}(t) + \mathbf{F_i}r_i(t) \tag{1}$$

with:

$$\mathbf{F_i} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}, \mathbf{u_i}(t) = \begin{bmatrix} u_{ld}(t) \\ u_{lq}(t) \end{bmatrix}, r_i = \omega_{m\,ref}(t)$$

$$\mathbf{A_i}=\begin{bmatrix} -\frac{R_s}{L_s} & 0 & 0 & 0 \\ 0 & -\frac{R_s}{L_s} & 0 & 0 \\ 0 & -\frac{K_t}{J_m} & -\frac{B_m}{J_m} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \mathbf{x}(t)=\begin{bmatrix} i_d(t) \\ i_q(t) \\ \omega_m(t) \\ x_\omega(t) \end{bmatrix}, \mathbf{B_i}=\begin{bmatrix} \frac{K_p}{L_s} & 0 \\ 0 & \frac{K_p}{L_s} \\ 0 & 0 \\ 0 & 0 \end{bmatrix},$$

where: R_s, L_s – resistance and inductance of the PMSM stator, K_t – torque constant, J_m – moment of inertia B_m – viscous friction, $i_d(t)$, $i_q(t)$ – current space vector components, $\omega_m(t)$ – angular speed of the PMSM shaft, K_p – gain of VSI, $u_{ld}(t)$, $u_{lq}(t)$ – linear components of control voltages, $\omega_{m\,ref}(t)$ – reference value of angular speed. In order to ensure steady-state error-free operation for step changes of reference speed and load torque the $x_\omega(t)$ has been introduced and it has the following formula:

$$x_\omega(t)=\int_0^t\left[\omega_m(\tau)-\omega_{m\,ref}(\tau)\right]d\tau \tag{2}$$

A discrete state feedback speed controller for the considered drive has the following formula:

$$u_i(n)=-\mathbf{K}\mathbf{x_i}(n)=-\mathbf{K_x}\mathbf{x}(n)-\mathbf{K_\omega}\mathbf{x_\omega}(n) \tag{3}$$

with:

$$\mathbf{K}=\begin{bmatrix}\mathbf{K_x} & \mathbf{K_\omega}\end{bmatrix}=\begin{bmatrix} k_{x1} & k_{x2} & k_{x3} & k_{\omega1} \\ k_{x4} & k_{x5} & k_{x6} & k_{\omega2} \end{bmatrix}$$

where: n is a discrete sample time index. Tuning process of the SFC requires selection of all coefficients at the same time, which is difficult task. In the proposed approach linear-quadratic optimization (LQR) has been used to determine those coefficients. This method minimizes a discrete performance index for weighting matrices **Q** and **R**:

$$I_{LQR}=\sum_{n=0}^{\infty}\left[\mathbf{x_i}^{\mathrm{T}}(n)\mathbf{Q}\mathbf{x_i}(n)+\mathbf{u_{li}}^{\mathrm{T}}(n)\mathbf{R}\mathbf{u_{li}}(n)\right] \tag{4}$$

where:

$$\mathbf{Q}=\mathrm{diag}\left(\begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}\right), \mathbf{R}=\mathrm{diag}\left(\begin{bmatrix} r_1 & r_2 \end{bmatrix}\right)$$

3. ARTIFICIAL BEE COLONY ALGORITHM

The Artificial Bee Colony algorithm is a stochastic optimization algorithm proposed by Karaboga in 2005 [10]. The author got inspired by nature – the intelligent foraging behavior of honey bee swarm.

The algorithm is based on food sources. Each of food source is connected with three kinds of bees: employed bees, onlooker bees and scouts. The employed bee goes to its food source, but also randomly follow another bee. The onlooker bee is similar to employed bee, but it is smarter, because it follows the bee that bring the highest amount of nectar to the hive. The last kind of bees, scouts, are mostly not used. They fly away from the hive only if some of food source has been abandoned. The algorithm is divided into three phases: (i) employed bees phase, (ii) onlooker bees phase and (iii) scouts phase. Block diagram of the algorithm is shown in the Fig. 1. All phases are repeated for N times. In order to reduce the diversity of new food source produced in onlooker bees phase, the modification rate (MR) parameter is introduced. Parameter called *limit* is the number of failed attempts, that determine if food source is abandoned. Additional parameter called scout production period (SPP) is introduced in order to restrain spread of the colony. Algorithm is box-constrained in each dimension by lower bounds (LB) and upper bounds (UP). Due to ABC is employed to obtain **Q** and **R** weighting matrices, the solution dimensionality (DIM) is equal to 6. The parameters of Artificial Bee Colony algorithm used for auto-tuning process of SFC are listed in Table 1.

Table 1. Parameters of Artificial Bee Colony algorithm.

Parameter	Symbol	Value
Total number of iterations	N	25
Solution dimensionality	DIM	6
Colony size	CS	40
Number of food sources	NFS	CS / 2
Trial limit	*limit*	NFS*DIM
Scout production period	SPP	NFS*DIM
Modification rate	MR	0.8
Lower bounds in each dimension	LB	10^{-3}
Upper bounds in each dimension	UB	10^{4}

In order to apply an optimization algorithm to PMSM drive, the performance index, that will fulfill respective objectives, has to be determined. In a considered case, the main objective is to obtain a satisfactory dynamical behavior of the angular velocity and to ensure steady-state error-free operation in case of step changes of the reference value and of the load torque. Additionally, in order to maximize efficiency of the PMSM, the d-axis current should be zero and chattering in control signals shouldn't occur. The performance index has the following form [11]:

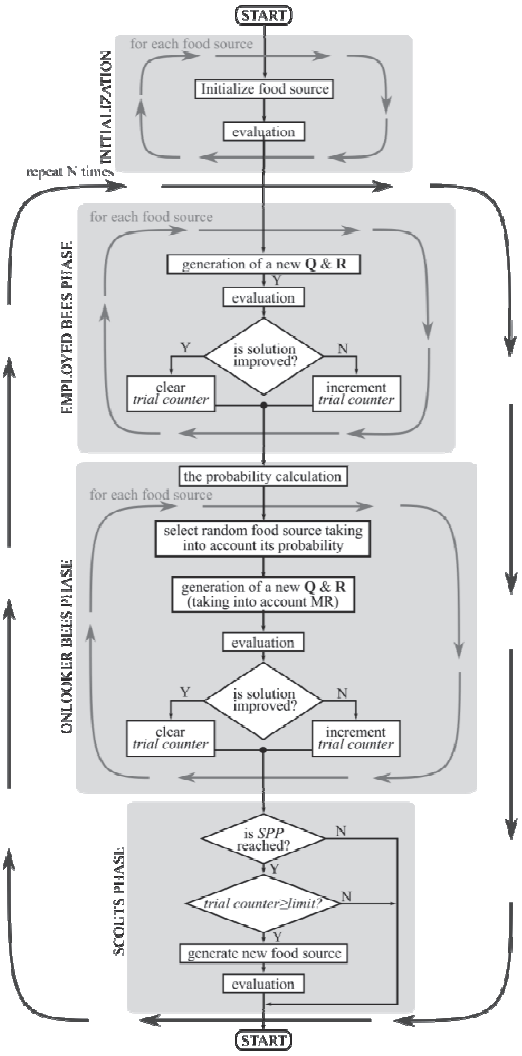

Fig. 1. Block diagram of Artificial Bee Colony algorithm

$$f(x)=\sum_{n=0}^{N}\left[e_{\omega}^{2}(x,n)nT_s+e_{id}^{2}(x,n)nT_s+\alpha\Delta u_{sq}^{2}(x,n)\right] \tag{5}$$

with:

$$e_{\omega}(x,n)=\omega_m(x,n)-\omega_{m\,ref}(x,n)$$
$$e_{id}(x,n)=i_d(x,n)-i_{d\,ref}(x,n)$$
$$\Delta u_{sq}(x,n)=\left[u_{sq}(x,n)-u_{sq}(x,n-1)\right]/T_s$$

where: α is manually selected coefficient.

Since SFC in its basic form operates without state-space variable constraints, respective method should be added to optimization. In order to achieve currents in proper range, the Deb's rules method has been applied [12]. The Deb's rules method is a simple constraint-handling method, which use an additional variable during optimization called *violation*. This variable, in proposed problem, is defined as:

$$violation = \sum_{n=0}^{N} MAX\left(0, \frac{\left|i_q(n)\right|}{i_{q\,max}} - 1\right) \tag{6}$$

The *violation* defines if solution is feasible (*violation* == 0) or infeasible (*violation* > 0). The Deb's rules method extends process of comparison two solutions as follows:

– a feasible solution is always better than infeasible solution,
– a better objective function is selected, when both solutions are feasible,
– a smaller *violation* is selected, when both solutions are infeasible.

4. PARALLELISATION OF AUTO-TUNING PROCESS

The Parallel Computing Toolbox for MATLAB/Simulink allows to take advantage of using both: multi-core processor (CPU) and graphical processing unit (GPU). It allows to create parallel pool, which is set of MATLAB workers on a compute cluster or desktop. The parallel pool can be used by functions e.g. *parfor* – parallel "for" loop, *parfeval* – asynchronous function computation, *parsim* – asynchronous Simulink model simulation. The script, which uses one of above mentioned function, is executing in fork-join model. In fork-join model the master worker forks into multiple workers (parallel computation) only for parts of the program, and then, when all workers finish their job, they join to a single worker again. It is worth to point out, that creating and deleting of the parallel pool are operations, which together least for several dozen seconds. An example for asynchronous simulation of a Simulink model is listed below:

```
N = 4;  % Number of Workers
% Creating parallel pool
parpool(N);
% Initialize Simulink objects
in(1:N) = Simulink.SimulationInput('Model');
% Set variable value for every worker
for i = 1:N
   in(i) = in(i).setVariable('x', i*0.1);
end
% Simulate asynchronous
out = parsim(in);
% Removing parallel pool
```

```
13.  delete(gcp('nocreate'));
14.  % Plot the results
15.  figure; hold on;
16.  for i = 1:N
17.     plot( out(i).t, out(i).y );
18.  end
```

Auto-tuning process of SFC takes a several-tens of minutes. The most time takes simulation of considered plant (i.e. PMSM fed by VSI) in Simulink environment. In order to reduce computation time, the ABC algorithm has been adapted to fork-join model. All phases have been divided into three parts: (i) generate all new food sources, (ii) evaluating new food sources and (iii) comparing actual food sources with a new one. The first and third part is evaluating on single thread, because they are not very time consuming, while the second part is executing with parallel pool of workers (*parsim* function). The division has been presented on the Fig. 2 as block diagram.

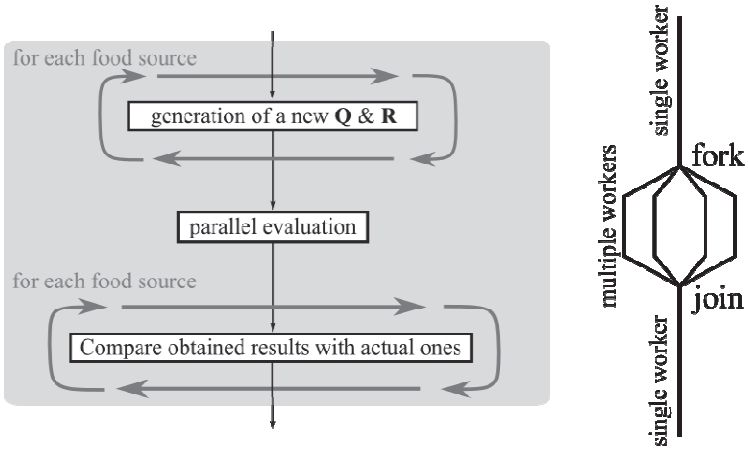

Fig. 2. Block diagram of parallel computation of ABC phase

5. RESULTS AND EFFICIENCY ANALYSIS

The most important simulation parameters are: simulation time (equal to 2 s in this particular case), sample time of SFC (T_s = 0.1 ms), fixed-step solver is used with step size set to T_s /10, evaluation examine step changed of the reference value and of the load torque. Used parameters of PMSM drive are presented in [11]. The progress of performance index, angular speed, currents and control signals obtained coefficients are presented in the Fig. 3, while coefficients obtained from ABC algorithm are presented below:

$$\mathbf{Q} = \mathrm{diag}\left(\begin{bmatrix} 1\times10^{-3} & 0.0706 & 0.0376 & 9.6011 \end{bmatrix}\right) \tag{7}$$

$$\mathbf{R} = \mathrm{diag}\left(\begin{bmatrix} 2.1518 & 1.7272 \end{bmatrix}\right), \tag{8}$$

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{\mathbf{x}} & \mathbf{K}_{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} 2.2\times10^{-5} & 0 & 0 & 0 \\ 0 & 0.1903 & 0.1578 & 2.1163 \end{bmatrix} \tag{9}$$

Obtained coefficients ensure: (i) a satisfactory dynamical behavior of the angular velocity, (ii) steady-state error-free operation in case of step changes of the reference value and of the load torque and (iii) proper current limitation.

The same problem has been solved with number of workers in parallel pool from 1 to 8, but also on three different personal computers: (i) Intel Core i3-3120M 2.5GHz (2 cores, 4 threads) supported by 4GB RAM, (ii) Intel Core i5-7500 3.4GHz (4 cores, 4 threads) supported by 16GB RAM and (iii) Intel Core i7-7700HQ 2.8GHz (4 cores, 8 threads) supported by 8GB RAM. The operating system used on each computer was Windows 10. In order to compare computation time, *speedup* term is used, which is defined as sequential program computation time divided by parallel program computation time:

$$speedup = \frac{t_1}{t_N} \tag{10}$$

where: N is a number of workers in parallel pool. The results of computation time reduction for presented optimization problem are shown in Table 2 and its graphical representation in the Fig. 4.

The results confirm that parallel computing allows decreasing computation time of auto-tuning of state feedback speed controller for PMSM drive. Speedup obtained for i5-7500 and i7-7700HQ are similar, due to the same number of physical cores. On the other hand, the maximum speedups for these processors are obtained with different number of workers, which is caused by different number of logical threads. The optimal number of workers depends on number of physical cores and logical threads of the processor. Increasing the number of workers above optimal value does not improve speedup and even decrease it, which is caused by race condition. It is worth to point out, the operating system has processes, that have higher priority than Matlab application. For this reason, Matlab is not able to use all physical cores during the whole simulation time.

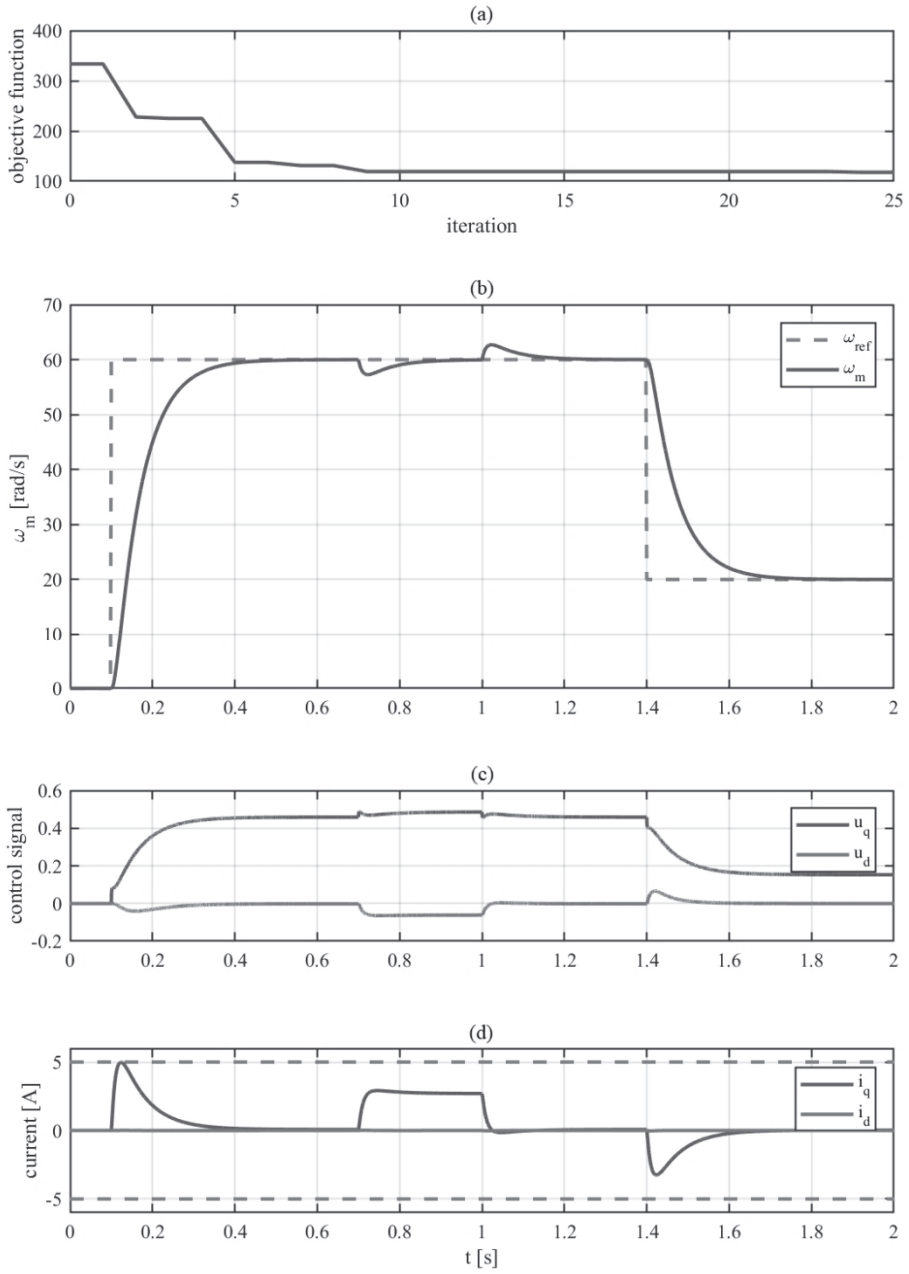

Fig. 3. (a) Objective function progress during optimization process, (b) Angular speed, (c) Control signal and, (d) Currents for obtained SFC coefficients

Table 2. Computation time and speedup for different number of workers.

	Intel Core i3-3120M 2.5GHz 2 cores, 4 threads		Intel Core i5-7500 3.40GHz 4 cores, 4 threads		Intel Core i7-7700HQ 2.8GHz 4 cores, 8 threads	
no. workers	time [min]	speedup	time [min]	speedup	time [min]	speedup
1	168.05	1.00	74.87	1.00	73.47	1.00
2	110.31	1.52	37.36	2.00	41.39	1.78
3	94.97	1.77	28.83	2.60	31.83	2.31
4	95.02	1.77	25.36	2.95	24.96	2.94
5	97.13	1.73	26.17	2.86	23.11	3.18
6	99.69	1.69	26.23	2.85	24.35	3.02
7	98.54	1.71	27.30	2.74	22.71	3.23
8	102.12	1.65	26.45	2.83	25.07	2.93

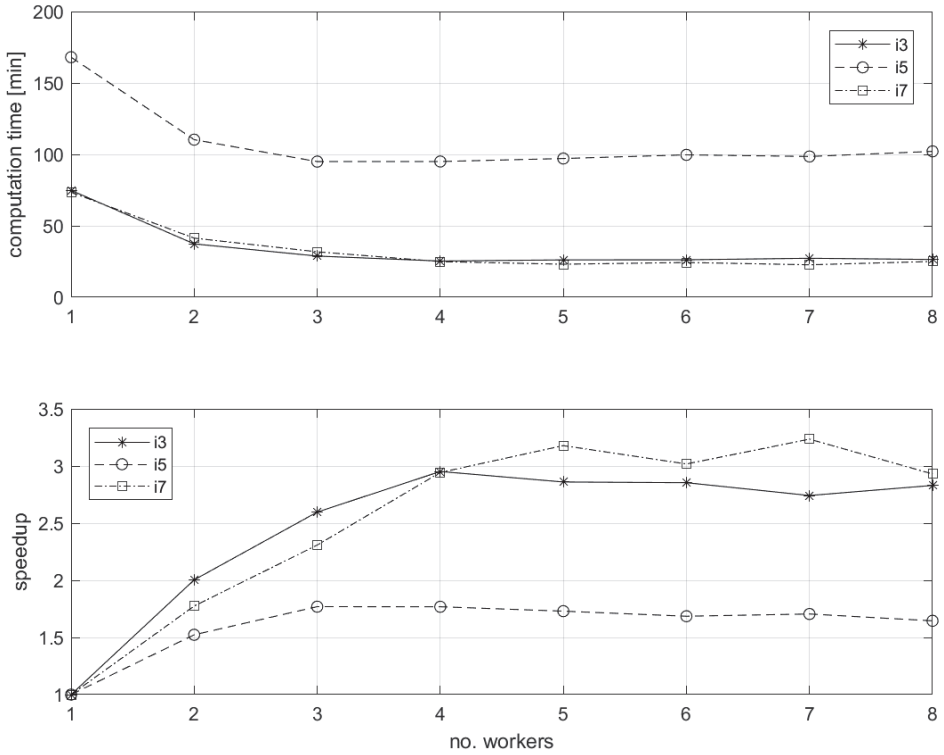

Fig. 4. Graphical representation of computation time and speedup for different number of workers.

6. CONCLUSION

In this paper, efficiency analysis of parallel computing applied to auto-tuning of state feedback speed controller for PMSM drive has been presented. The Artificial Bee Colony has been applied to auto-tuning process of state feedback speed controlled for PMSM drive fed by VSI. The Artificial Bee Colony algorithm has been also adapted to fork-join model. The computation time of solving described optimization problem has been examined on various parallel pool size and on three different personal computers. The results confirm a positive effect of parallel computing on the computation time. The maximum obtained speedup is equal to 1.77 with i3-3120M, 2.95 with i5-7500 and, 3.23 with i7-7700HQ. It is worth to point that the difference in computation time observed for i5 and i7 is very small. Similar situation is observed for speedup. The main reason is related to the same number of physical cores in considered processors. It should be noted, that by increasing the number of logical threats the management possibilities of operating system increases but the computational capability does not change.

REFERENCES

[1] Open Multi-processing homepage (www.openmp.org).
[2] Message Passing Interface homepage (www.mpi-forum.org).
[3] MathWorks, Parallel Computing Toolbox homepage (www.mathworks.com/products/parallel-computing.html).
[4] Liu, X., Chen, H., Zhao, X., Belahcen, A., Research on the Performances and Parameters of Interior PMSM Used for Electric Vehicles, IEEE Transactions on Industrial Electronics, vol. 63, no. 6, pp. 3533–3545, 2016.
[5] Faa-Jeng Lin, Hsin-Jang Shieh, Po-Huang Shieh, Po-Hung Shen, An adaptive recurrent-neural-network motion controller for X-Y table in CNC Machine, in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 36, no. 2, pp. 286–299, 2006.
[6] Zhong L., Rahman M. F., Hu W. Y., Lim K. W., Analysis of direct torque control in permanent magnet synchronous motor drives, in IEEE Transactions on Power Electronics, vol. 12, no. 3, pp. 528–536, 1997.
[7] Jarzebowicz L., Error analysis of calculating average dq current components using regular sampling and Park transformation in FOC drives, in Proc. IEEE Intern. Conf. EPE 2014, pp. 901–905, 2014.
[8] Franklin G. F., Powell J. D., Emami-Naeini, A., Powell, J. D., (1994), Feedback control of dynamic systems, vol. 3, Reading, MA: Addison-Wesley.
[9] Kamiński M., Application of the BAT algorithm in optimization of adaptive state space controller used for two-mass system, Przegląd Elektrotechniczny, vol. 93, no.1, pp. 300–304, 2017, (in Polish).

[10] Karaboga D., Basturk B., Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems. In: Melin P., Castillo O., Aguilar L.T., Kacprzyk J., Pedrycz W. (eds) Foundations of Fuzzy Logic and Soft Computing, IFSA 2007, Lecture Notes in Computer Science, vol. 4529. Springer, Berlin, Heidelberg.
[11] Tarczewski T., Grzesiak L. M., An Application of Novel Nature-Inspired Optimization Algorithms to Auto-Tuning State Feedback Speed Controller for PMSM, IEEE Transactions on Industry Applications, vol. 54, no. 3, pp. 2913–2925, 2018.
[12] Deb K., An efficient constraint handling method for genetic algorithms. Comput. Meth. Appl. Mech. Eng., vol. 186, pp. 311–338, 2000.

(Received: 24.01.2019, revised: 09.03.2019)