Applied Soft Computing 11 (2011) 2556–2564

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Real-time particle swarm optimization based parameter identification applied to permanent magnet synchronous machine

Wenxin Liu[a,*], Li Liu[b], Il-Yop Chung[c], David A. Cartes[d]

[a] *Klipsch School of Electrical and Computer Engineering, New Mexico State University, Las Cruces, NM 88003, USA*
[b] *Controls-Common, Cummins Inc., 1460 N. National Rd., Columbus, IN 47201, USA*
[c] *School of Electrical Engineering, Kookmin University, 861-1 Jeongneung-dong, Seongbuk-gu, Seoul, 136-702, Korea*
[d] *Center for Advanced Power Systems, Florida State University, 2000 Levy Avenue, Tallahassee, FL 32310, USA*

ARTICLE INFO

Article history:
Received 16 October 2009
Received in revised form 22 June 2010
Accepted 4 August 2010
Available online 30 October 2010

Keywords:
Particle swarm optimization
Parameter identification
Real-time implementation
Permanent magnet synchronous motor

ABSTRACT

Particle swarm optimization (PSO) has been widely used in optimization problems. If an identification problem can be transformed into an optimization problem, PSO can be used to identify the unknown parameters in a nonlinear model that is used to describe a system. Currently, most PSO based identification or optimization solutions can only be implemented offline. The difficulties of online implementation mainly come from the unavoidable lengthy simulation time to evaluate a candidate solution. In this paper, a technique for faster than real-time simulation is introduced and implementation details of PSO based identification algorithm is presented. Performance of the proposed technique is demonstrated through application to parameters identification of permanent magnet synchronous machine control system. The algorithm is implemented in Matlab/Simulink with the most fundamental blocks and Embedded Matlab Functions. Thus the program can be compiled to C/C++ code through Real-time Workshop and be able to run on hardware controllers like dSPACE. The proposed techniques can also be applied to many other online identification and optimization problems.

1. Introduction

System parameters can be identified indirectly from measurements if it is hard or expensive to measure the parameters directly. Parameter identification is very important for state estimation, fault detection and diagnosis, controller design, etc. Many traditional parameter identification algorithms have been proposed in the past. However, accurate and efficient identification of complex nonlinear systems is still a challenging problem. Computational intelligence is the study of adaptive mechanisms to enable or facilitate intelligent behavior in complex and changing environments. It has been demonstrated that applications of computation intelligence techniques can bring about better performance or improved designs. Unlike traditional identification algorithms whose applications are limited by model structures, computational intelligence based algorithms do not have special requirements on model structures. As long as system model performs differently for different parameters, which is almost always true, computational intelligence will be able to identify the unknown parameters in the models. There are numerous researches on the applications of computational intelligence techniques to parameters identification and control system design, such as [1–10].

PSO is a population based stochastic search algorithm [11–13]. Because of its simplicity and computational efficiency, PSO has been widely used to solve a broad range of optimization problems. In additional to optimization problems, PSO can also be applied to identification problems provided that the identification problems can be transformed into optimization problems. There have been numerous applications that apply PSO to parameter tuning or identification problems, such as [14–16]. However, most of existing PSO based model parameter identifications were only implemented offline. During the optimization/identification process, to evaluate a candidate solution, system model has to be simulated with the candidate solutions/parameters under the same inputs and initial conditions as the measurement. PSO algorithm requires a number of iterations to obtain a satisfactory solution. For every iteration, the system model has to be simulated once. Thus, the model needs to be simulated a number of times to identify parameters from each measurement. Because of the unavoidable simulation time needed to evaluate the candidate solutions, unless the simulations can be finished faster than real-time, successful identification cannot be accomplished within the time used for measurement.

In this paper, the "real time" objective is to achieve good identification within the time used to measure system outputs. To implement PSO algorithm in real time, two possible solutions can

* Corresponding author. Tel.: +1 575 646 2570; fax: +1 575 646 1435.
E-mail address: wliu@nmsu.edu (W. Liu).

be explored. The first is to design a compact algorithm to save memory and improve computational efficiency, such as the compact genetic algorithm in [17,5]. The second solution is to realize faster than real-time simulation so as to speed up the performance evaluation process. This paper chooses to explore the possibility of the second method. By modifying the time constants of the original model, the modified model can run n-times faster. Besides, to minimize simulation time, only a small part of the system model that comprises the parameters to be identified is simulated. By doing that, the simulation time required for parameter evaluation can be tremendously reduced. The saved time can be used to complete predefined number of iterations. By properly adjusting the sample time, step size, and numbers of samples and iterations, it is possible to a predefined number of iterations within the same time used for measurement.

The real-time implementation of PSO based identification algorithm is much more difficult compared to the offline implementation. There are a lot of issues to be considered, such as the initialization, flow control, and simulation synchronization. This paper describes all the details of implementation using hardware controller. Since the dSPACE® controller can run Simulink®/Real-time Workshop® generated C code seamlessly, implementation is made easy. The implementation techniques with dSPACE controller can be directly applied to other hardware controllers with similar functions.

To evaluate its performance, the proposed technique is applied to identify two parameters in a Permanent Magnet Synchronous Motor (PMSM) control system, i.e. the stator resistance R_s and disturbed load torque T_{ld}. The implementation details will help the understanding of the proposed techniques. Simulation results demonstrate that the real-time PSO based technique is able to identify both time-invariant and time-varying parameters accurately. In addition to this PMSM application, the techniques described in this paper can also be applied to many other online identification and optimization problems.

The rest of this paper is organized as follows: Section 2 provides a brief introduction to the PSO based identification techniques. Section 3 introduces faster than real-time simulation. Section 4 describes the details of real-time implementation. Its application to parameter identification of PMSM control system is presented in Section 5 and concluding remarks are given in Section 6.

2. Background

2.1. Particle swarm optimization

Particle swarm optimization (PSO) is a population based stochastic search algorithm. It was first introduced by Kennedy and Eberhart [11]. Since then, it has been widely used to solve a broad range of optimization problems. The algorithm was presented as simulating animals' social activities, e.g. insects, birds, etc. It attempts to mimic the natural process of group communication to share individual knowledge when such swarms flock, migrate, or hunt. If one member sees a desirable path to go, the rest of this swarm will follow quickly [12,13]. In PSO, this behavior of animals is imitated by particles with certain positions and velocities in a searching space, wherein the population is called a swarm, and each member of the swarm is called a particle. Starting with a randomly initialized population, each particle in PSO flies through the searching space and remembers the best position it has seen. Members of a swarm communicate good positions to each other and dynamically adjust their own position and velocity based on these good positions. The velocity adjustment is based upon the historical behaviors of the particles themselves as well as their neighbors. In this way, the particles tend to fly towards better and better searching areas over the searching process [11]. The searching procedure based on this concept can be described by (1).

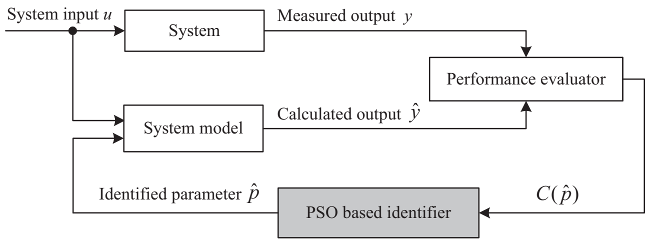

Fig. 1. Block diagram of the PSO based identification approach.

$$\begin{aligned} v_i &= w \cdot v_i + c_1 \cdot rand_1 \cdot (x_p - x_i) + c_2 \cdot rand_2 \cdot (x_g - x_i) \\ x_i &= x_i + v_i \end{aligned} \tag{1}$$

where w, c_1, and c_2 are the inertia weight, cognitive acceleration and social acceleration constants respectively; $rand_1$ and $rand_2$ are two random numbers; x_i represents the location of the i^{th} particle; x_p represents the best solution (fitness) the particle has achieved so far (*pbest*); x_g represents the overall best location obtained so far by all particles in the population (*gbest*); v_i represents the velocity of the particle with $v_i^{\min} \le v_i \le v_i^{\max}$.

$v_i^{\max}$ determines the resolution, or fitness, with which regions between the present position and target position are searched. The constants c_1 and c_2 represent the weighting of the stochastic acceleration terms that pull each particle toward *pbest* and *gbest* positions. According to past experience, $v_i^{\max}$ is often set at 10–20% of the dynamic range of the variable on each dimension, and w, c_1, and c_2 are often set to 0.8, 2, and 2.

From the updating rules and flow chart that can be found in many PSO related papers, it can be seen that PSO is very simple in concept and easy in realization. Thus, PSO has gained popularity in recent years and many researchers from different fields have attempted to improve the performance of the original PSO. Newer versions and the choices of proper values of the parameters to improve the performances of PSO are discussed in [18–26]. Both performance and execution time of PSO are improved in these modifications. Recently, a simple and efficient way of tuning the PSO parameters was presented by Pedersen et al. [27,28]. The best performing PSO parameters were found to be contrary to guidelines in the literature and often yield satisfactory optimization performance for simple PSO variants. Recent survey of PSO can be found in [29–33], where [32,33] focused on power system applications.

2.2. PSO based parameter identification

The block diagram of the PSO based parameter identification approach can be illustrated with Fig. 1.

First, system response under input u is measured for identification. Then, system model with tentative parameters ($\hat{p}$) is simulated under the same initial condition and inputs as the system. The outputs of the simulated model ($\hat{y}$) and measurement (y) of system are input to a performance evaluator for comparison. The performance evaluator calculates the fitness $C(\hat{p})$ of the tentative solution according to a cost/fitness function. The commonly used cost function can be defined as a weighted quadratic function as in [5,6]. Once all of the candidate solutions (particles) have been evaluated, the PSO based identifier can update candidate solutions according to the updating rules to provide a better set of candidate solutions. The new candidate solutions with updated $\hat{p}$ will be used to update the system model for next iteration of optimization till a preset number of iterations has been accomplished or the preset maximum allowable cost/fitness $C(\hat{p})$ has been met. More details on the above identification process can be found in related literature, such as in [5,6].

3. Faster than real-time simulation

The requirement or definition of "real-time" is different for different applications. In this paper, the objective is to achieve good identification within the time used to measure system outputs. For example, if 1000 samples are measured at a sample time of 10^{-4} s, which means it takes 0.1 s to measure the dataset of 1000 samples, then the PSO based algorithm is required to identify system parameters within 0.1 s.

According to Section 2, PSO usually needs a number of iterations to identify parameters according to a measured dataset. For every iteration, the system model needs to be simulated once for each candidate solution. If there are n particles, which means n candidate solutions, the system model needs to be simulated n times for each iteration. Even if all the particles can be simulated in parallel, the optimization has to be done iteration after iteration. Thus, the major problem for real-time implementation is to complete predefined number of simulations in a limited time. To realize the "real-time" objective described above, faster than real-time simulation is necessary. Instead of using hardware with powerful computing capability, the simulation time can be minimized by modifying original system model. This idea can be illustrated using the following equations.

$$\dot{X}_1 = F(X_1) + G(X_1, U_1) \tag{2}$$

$$\dot{X}_2 = n * [F(X_2) + G(X_2, U_2)] \tag{3}$$

Eq. (2) describes the dynamics of the original system and (3) describes the modified system. In (3), n is a positive number that is larger than 1 and the definitions of *F(.)* and *G(.)* in (2) and (3) are the same. Since the time constant of (3) is $1/n$ that of (2), the system response of (3) is n-times faster than (2). Given the same initial conditions and n-times faster input, (3) can provide n-times faster response compared with (2). If both systems are running in real-time and the simulated system (3) is measured n-times faster, only $1/n$ of the time for measurement is needed to obtain the same set of samples.

The idea can be illustrated using a practical example as shown in Fig. 2. The only difference between the two systems is the time constants. Since time constant of System 2 is 10 times smaller, System 2 responds 10 times faster than System 1. To obtain the same number of samples for System 1, System 2 only needs to be simulated for 0.1 s.

Now, let's consider another practical example illustrated in Fig. 3. If 1000 samples are measured from the original system at a sample time of 10^{-4} s, it will take 0.1 s to obtain the 1000 samples. If we modify the system model by making it running 10 times faster and measure the modified system at a sample time of 10^{-5} s, it will only take 0.01 s to obtain the 1000 samples for the modified system. Even though their time stamps are different, the quantities of the two 1000-samples will be the same. According to the PSO based identification algorithm, 5 particles in a swarm can be simulated in parallel. Thus, we can nearly finish 10 iterations within 0.1 s. Since it takes time to update the parameters according to PSO algorithm, we can choose to run the optimization for 5 times/iterations in our implementation. In doing so, we can realize faster identification, which means the interval for updating identified parameters (just a little longer than 0.05 s if considering the updating process of PSO) is less than the measurement time (0.1 s).

It should be noted that the above-mentioned examples assume that the modified system can be simulated in real-time. During implementation, the actual simulation time is mainly determined by the capability of the controller's hardware and the complexity of the model. If the modified model can be simulated in real-time or even faster, the real-time implementation of the PSO based identification will have no problem. In the following discussion, we will consider the situation when the modified model cannot be simulated in real-time. What can we do if we have to overcome the problem by tuning the algorithm rather than upgrading the controller hardware?

In such situations, the following two methods can be adopted one after another. The first method is to decrease sampling frequency, the number of samples, and number of iterations. If this

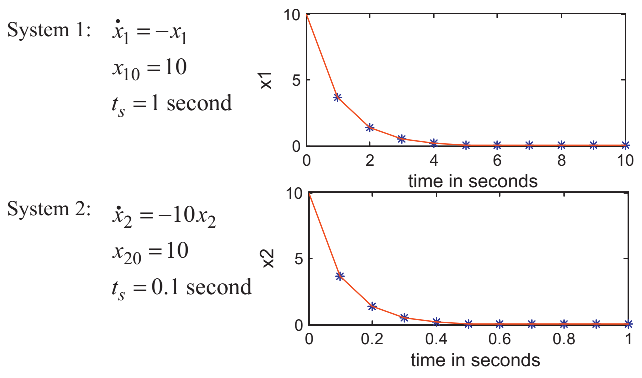

Fig. 2. Comparison of system response with different time constants.

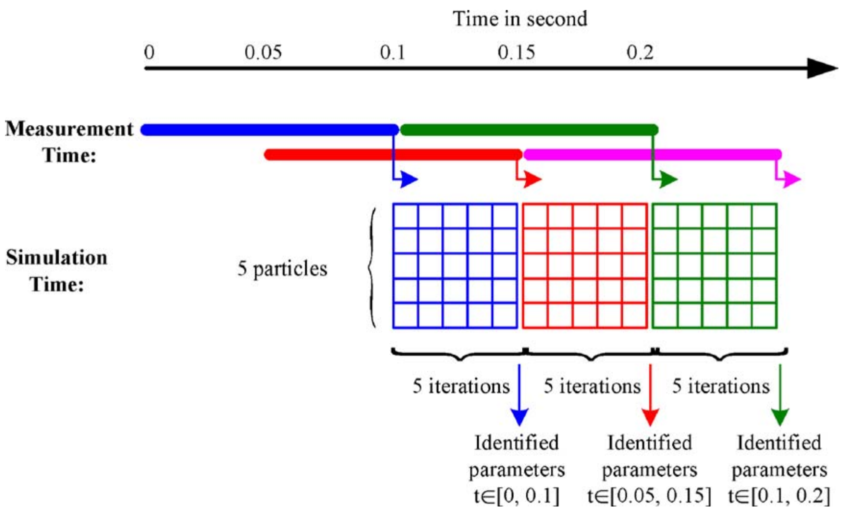

Fig. 3. Running of the PSO based identification algorithm.

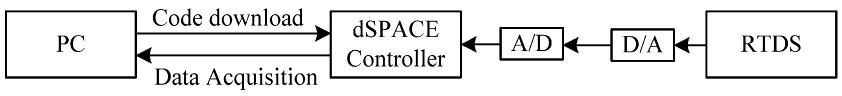

Fig. 4. Block diagram of real-time hardware in the loop simulation.

method works, there is no impact on the performance and the "real-time" objective. If the first method does not work, we can try the second method. The second method is to decrease n, which will definitely make it work. However, the expectation on performance has to be lowered. Detailed analyses are given below.

By decreasing sampling frequency, the simulation can be focused on less fixed data points. In this way, the computational burden on controller hardware and I/O module can be alleviated. However, according to Shannon sampling theorem, the sample time cannot be too large. Otherwise, the measurement will miss some important details in measurement. If decreasing sampling frequency start to result some identification error, we need to keep the minimum working sampling frequency and try other methods. The second method is to use fewer samples, which mean simulating the system for a shorter time. As mentioned before, the PSO based identification algorithm need different signature for different parameters. If the samples are not enough to demonstrate enough difference, it will make identification very difficult. Again, if using fewer samples does not work, we can try the last method that will not impact the "real-time" performance, which is to reduce the number of iterations for each measured dataset. Less iteration means more simulation time for each iteration provided the total time for identification is fixed. In this way, a more complicated simulation can be finished. However, PSO may need several iterations to find a good solution. Thus, the method of decreasing number of iterations may not work for complex problem. In this worse case, if all of the above mentioned three methods fail, we have to resort to the last method that is to decrease n.

Decreasing n will result slower simulation, then the predefined number of iterations may not be finished within the time for measurement. To provide accurate identification, the predefined number of iterations has to be maintained. Thus, the interval for updating identified parameters has to be increased. This will limit the applications to problems whose parameters do not change severely. This situation may be unavoidable and is constrained by the capability of controller hardware. However, by adjusting n and the above-mentioned related parameters, we can achieve the best possible performance out of an existing hardware controller. Anyway, this problem does not hinder the applications of the proposed techniques to systems where their parameters of system do not change rapidly. With the development of computer techniques, the improved hardware will extend the real-time implementation techniques to a wider range of applications. In addition to the above-mentioned possible problems, there are many other issues to be considered during implementation. For example, the PSO generated solutions may make the model running unstable, thus the program should be able to terminate unstable simulations and regenerate a new stable one. Details of the real-time implementation of the PSO based identification algorithm are provided in Section 4.

4. Details on real-time implementation of PSO

Compared to offline implementation, real-time implementation is much more difficult. There are a lot of issues to be considered. For realtime controller hardware in the loop simulation, we choose to use dSPACE® controller and Real-Time Digital Simulator® (RTDS). dSPACE controller enables us to implement our real-time PSO

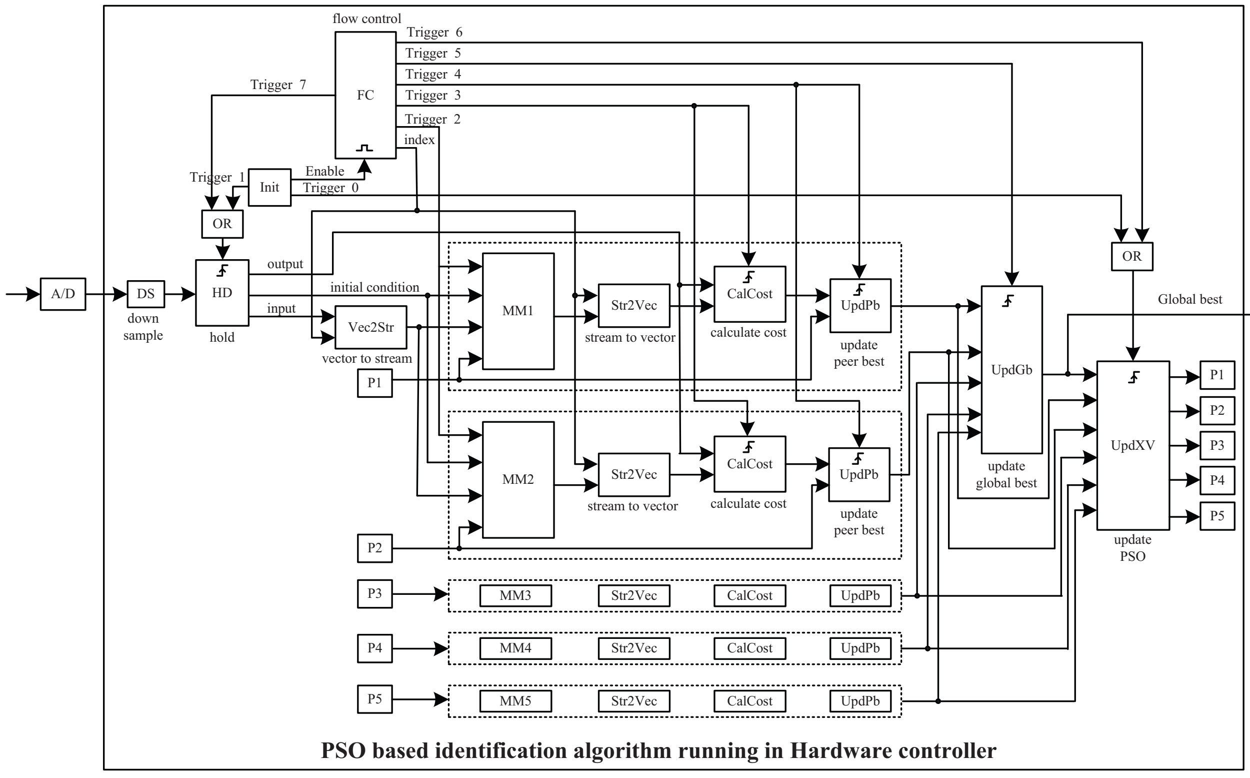

Fig. 5. Block diagram of the real-time PSO algorithm running in dSPACE controller.

Table 1
Functions of the modules in Fig. 5.

Name	Function
Init	Initialize simulation model before “FC” takes over the flow control
FC	Control the flowing of data by generating triggering signals
DS	Down sample the measured data by n-times
HD	Hold the output of DS (measured dataset) for identification
Vec2Str	Convert vector (measured dataset) to streams of control inputs
MM1∼5	Simulate the modified model
Str2Vec	Save the outputs of MM1∼5 (samples) to vector for comparison
CalCost	Evaluate particles by comparing measured and simulated datasets
UpdPb	Update the peer best particles
UpdGb	Update the global best particle
UpdXV	Update the position and velocity vectors
P1∼5	Store particles in memory for read/write

based identification algorithms using in Simulink®. The Simulink code can be converted to C code through Real-time Workshop (RTW). Thus, the time-consuming C programming can be avoided. RTDS allow us to test our hardware controller in real-time and under customizable simulation environment. Through controller Hardware-In-the-Loop simulation, the proposed algorithm can be fully tested just like through experimental studies. The experimental setup is shown in Fig. 4.

The basic idea of the implementation is to use the same step size and sampling time for both measurement and simulations. In this way, the identification interval (the time used to identify parameters form a measured dataset) is fully under control. Since the modified model is supposed to simulate n times faster than the original system, the measurement needs to be down sampled by n times to match the simulated data as illustrated in Fig. 2.

The block diagram of the PSO based identification algorithm running in dSPACE controller is shown in Fig. 5. In the figure, five particles are used to identify parameters from measurements. The simulations of the five particles are conducted in parallel and synchronized by the flow control module. The difficulties of this implementation include reproducing measured control signal, synchronizing the simulations of modified models, and implementing the PSO algorithms step by step. Definitions of modules are provided in Table 1.

The “Init” module initializes the identification algorithm before the “FC” takes over the flow control. The “Init” block has three outputs. The first output is the “enable” signal, which will enable the “FC” block after a preset delay. The delay is used to restrain the identification algorithm from running during start up transient. The second output ‘Trigger 0’ initializes particles (P1–P5) before using them to simulate the modified models. The third output ‘Trigger 1’ is the triggering signal, which can trigger the “HD” block once the very first m samples have been measured after startup transient.

The “FC” module ensures desired operating sequence of the PSO based identification algorithm. The only input signal is from “Init” module. The “FC” module will be enabled once initialization process is done. The first output is ‘index’, which is input to both “Vec2Str” and “Str2Vec” modules. In this way, the outputs in response to the corresponding inputs can be measured. Among the 6 triggering signals, Trigger 2 is used to initialize the modified system model in MM1∼5. Trigger 3 is used to trigger ‘CalCost’ module after all simulated data have been measured. Triggers 4–6 are used to ensure a right PSO updating sequence. The triggering process of corresponding modules is repeated for preset number of iterations. After the identification of one measured dataset has been completed, the “FC” module will trigger “HD” module with ‘trigger 7’ to hold a new set of measurement for identification.

The data measured from RTDS is input to “DS” block for down sampling. The measurement is down-sampled by n times continuously and only the most recent m down-sampled data are recorded for future usage. Once the “HD” block is triggered, the output of the down sampling model is held fixed for the PSO algorithm to identify from which. Besides holding the input and output vectors, the “HD” block also initiates the simulation of “MM1∼5” modules and provides measured data to “CalCost” module to compare with simulated data.

The “Vec2Str” module is used to reproduce the sampled control signals one after another based on index received from the “FC” module. The control signal is then applied to “MM1∼5”. The same index is also input to the “Str2Vec” block, which will record simulation results for evaluation. Since the “index” inputs of “Vec2Str” and “Str2Vec” are the same, the measured and simulated data can be synchronized. Once all the m samples have been obtained from all of the “MM1∼5”, the “FC” module will generate a triggering signal for “CalCost” module. Then, the “CalCost” module will evaluate the performance of the particles according to the defined cost/fitness function.

The “MM1∼5” modules take four inputs. The first input signal is the initial condition provided by the “HD” module. The second input is the control signals generated from the “Vec2Str” module. These two inputs are necessary to simulate the same operating condition as the system. The third input is the reset signal (trigger 5 generated by “FC”), which will initialize the modified simulation model before a new round of simulation. Once triggered, all the internal states of the MM modules will be set to the same initial condition. And the fourth signal is the particles (parameters) to be evaluated. The output of the “MM1∼5” modules will be measured and compared with system measurements. During simulation, the whole control system model should try to be avoided to minimize simulation time. This can be done by simulating only the smallest independent module where inputs and internal states can be measured.

After all samples have been measured for “MM1∼5”, the “FC” block will trigger the “CalCost” block, which will calculate cost/fitness of the corresponding particle. Based on the cost/fitness, the peer best solutions and the global best solution will be updated in “UpdPb” and “UpdGb” modules respectively. After that, the “UpdXV” module will update particle information and the above evaluation process will be repeated until the terminating conditions have been met.

The “global best” is updated periodically and two sets of values can be displayed, one is the identified parameters, and the other is the corresponding cost. By observing the cost, we can track the running of the PSO algorithm and get an idea of how good the identification is.

5. PMSM parameter identification

5.1. Problem description

To aid advanced controller design for PMSM, it is very important to obtain an appropriate model of the motor. A good model should not only be an accurate representation of system dynamics but also facilitate the application of existing control techniques. Among a variety of models presented in literature since the introduction of PMSM, the two axis dq-model is the most widely used in variable speed PMSM drive control applications. This model is considered in this work. In general, PMSM’s dynamics can be partitioned into two subsystems, i.e. the electrical system and the mechanical system. Both subsystems can be described by nonlinear differential equations. In Park’s dq-axis synchronous rotating reference frame, the unsaturated electrical model of a sinusoidal PMSM is expressed as

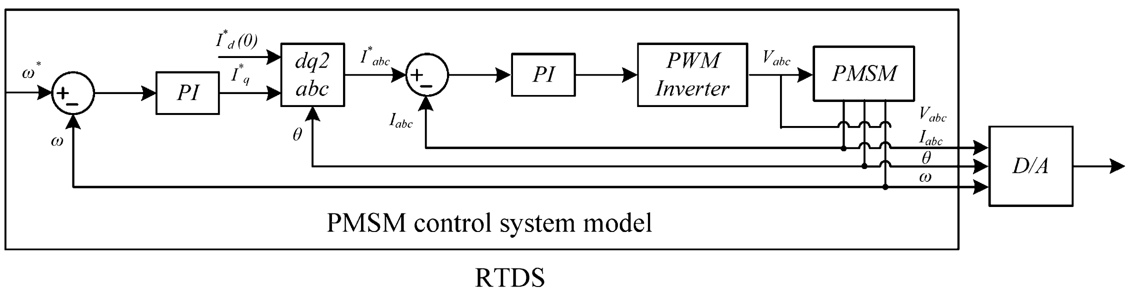

Fig. 6. Control system model running in RTDS.

follows [34,35].

$$\frac{di_d^r}{dt} = \frac{1}{L_d}[v_d^r - R_s i_d^r + \omega_r L_q i_q^r]$$
$$\frac{di_q^r}{dt} = \frac{1}{L_q}[v_q^r - R_s i_q^r - \omega_r L_d i_d^r - \omega_r \psi_{mag}] \tag{4}$$

In (4), i_d^r, i_q^r, v_d^r and v_q^r are the dq-components of the stator currents and voltages in synchronously rotating rotor reference frame; $(_r$ is the rotor electrical angular speed; parameters R_s, L_q, L_d and $(_{mag}$ are the stator resistance, d-axis and q-axis inductance, and the permanent magnet flux linkage, respectively.

The mechanical system of PMSM is composed of the rotor and its bearings of PMSM. Mechanical model is built upon the dynamics of these motional components. The derivative of the rotor electrical speed $(_r$ is expressed by the following motion equation:

$$\frac{d\omega_r}{dt} = n_p \frac{1}{J}\left[T_e - B\frac{\omega_r}{n_p} - T_L\right] \tag{5}$$

In (5), J denotes the inertia of the rotor; n_p is the number of pole pairs of the machine; B is the viscous friction coefficient, and T_L is the load torque. T_e is the electromechanical torque developed by the machine. This torque can be calculated as:

$$T_e = \frac{3}{2} n_p [i_q^r \psi_{mag} + (L_d - L_q) i_q^r i_d^r] \tag{6}$$

Eqs. (4)–(6) complete the description of system dynamics for PMSM. Taking i_d^r, i_q^r and $(_r$ as the state variables, and v_d^r and v_q^r as control signals (inputs), the model is a highly coupled nonlinear 2-input 3-output system in dq-axis reference frame.

In many applications, PMSM operates under various operating conditions, where various disturbances and parameter variations are unavoidable and unmeasurable. Often times, datasheet parameters only provide us a rough starting point for simulation or practical design purposes. They are not accurate enough for the entire PMSM operational range. For instance, PMSM stator resistance, R_s, is directly related to motor operating temperature. Its value tends to fluctuate up to twice of its nominal value [15]. Likely, the rotor inertia, J, and the frictional coefficient, B, may vary as they are coupled with load torques of the motor. Accurate knowledge of these parameters is important to control system performance. Therefore, it is of our interest to investigate an efficient approach to achieve precise parameter identification.

The PSO algorithm is utilized here to track PMSM parameters based on the traditional dq-model (4)–(6). The model with these estimated parameters should provide improved accuracy in modeling PMSM dynamics when maintaining its simplicity for controller designs. Specifically, this work focuses on the identification of the stator resistance R_s and disturbed load torque T_{Ld} that is defined in (7). It is a summation of the actual load torque T_L, and the disturbances caused by inertia and frictional coefficient variations.

$$T_{Ld} = T_L + \tilde{J}\frac{1}{n_p}\frac{d\omega_r}{dt} + \tilde{B}\frac{1}{n_p}\omega_r \tag{7}$$

In (7), we define $\tilde{J} = J - J_0$ and $\tilde{B} = B - B_0$. J_0 and B_0 denote the nominal values of motor inertia and viscous friction coefficient listed in PMSM datasheet.

Based on the output of the PMSM model, the cost function for identification performance evaluation of the PSO algorithm is shown in (8).

$$C(R_s, T_{Ld}) = \sum_{k=1}^{m}[(i_a(k) - \hat{i}_a(k))^2 + (i_b(k) - \hat{i}_b(k))^2 + (i_c(k) - \hat{i}_c(k))^2 + w(\omega_r(k) - \hat{\omega}_r(k))^2] \tag{8}$$

In (8), the phase currents i_{abc} and the rotor speed ω_r are the measurable system outputs, $\hat{i}_{abc}$ and $\hat{\omega}_r$ are the estimated values of i_{abc} and ω_r, which are calculated from the system model, R_s and T_L are the unknown parameters to be identified, n is the number of sampled data, and the weighting factor w is defined as $w = \left|i_{abc}\right| / |\omega_r|$.

The PMSM control system model running on RTDS is illustrated in Fig. 6. In this diagram, the motor is applied to a variable frequency drive system with cascaded PI-controllers. This design is widely used in industrial motor drive applications. Nominal parameters of the simulated PMSM are listed in Table 2. During simulation, several parameters, i.e. the stator resistance, R_s, motor inertia, J, and frictional coefficient, B, are defined as variables of time in order to demonstrate PSO method's capability to track multiple time-varying parameters. Since the time-varying inertia and viscous frictional coefficient can be interpreted as load torque disturbance, the objective of identification is to identify the disturbed load torque and stator resistance $[R_s, T_{Ld}]$.

The contents and dimensions of the inputs and outputs of the functional blocks are listed in Table 3.

Table 2
PMSM specification.

PMSM parameters	Nominal values (unit)
Power rating P_r (kw)	19.8
Rated speed ω_r (rpm)	1700
Current at rated speed I_r (Amps RMS)	41.56
Torque at rated speed T_r (N m)	67.27
Voltage constant k_e (V s/rad)	1.33
Torque constant kt (N m/Amp)	1.629
Max bus voltage V_{DC} (Volt)	560
Pole pairs n_p	4
Stator resistance R_s (Ω)	0.17
q-Axis inductance L_q (mH)	1.9
d-Axis inductance L_d (mH)	1.9
Static friction T_f (N m)	0.1483
Damping coefficient B (N m s/rad)	0.00115
Moment of inertia J (kg m^2)	0.008

Table 3
Input output definitions of function modules in Fig. 5.

Name	Inputs		Outputs	
	Contents	Dimension	Content	Dimension
Init			Enable	1
			Trigger0	1
			Trigger1	1
FC	Enable	1	Triggers2–7	(1)*6
			Index	1
DS	Measurement ($V_a, V_b, V_c, I_a, I_b, I_c, \omega, \theta$)	8*1	Samples ($V_A, V_B, V_C, I_A, I_B, I_C, \Omega, \Theta$)	8*1000
HD	Samples ($V_A, V_B, V_C, I_A, I_B, I_C, \Omega, \Theta$)	8*1000	Initial condition (I_{d0}, I_{q0}, θ_0)	3*1
			Inputs (V_A, V_B, V_C)	3*1000
			Outputs1 (I_A, I_B, I_C, Ω)	4*1000
Vec2Str	Inputs (V_A, V_B, V_C)	3*1000	Input (V_a, V_b, V_c)	3*1
	Index	1		
MM1∼5	Trigger2	1	Output (I_a, I_b, I_c, ω)	4*1
	Initial condition (I_{d0}, I_{q0}, θ_0)	3*1		
	Input (V_a, V_b, V_c)	3*1		
	p1∼5	(2*1)*5		
Str2Vec	Output (I_a, I_b, I_c, ω)	4*1	Outputs2 (I_A, I_B, I_C, Ω)	4*1000
	Index	1		
Calcost	Outputs1 (I_A, I_B, I_C, Ω)	4*1000	Cost/fitness	1
	Outputs2 (I_A, I_B, I_C, Ω)	4*1000		
UpdPb	p1∼5	(2*1)*5	Pbests1∼5	(2*1)*5
UpdGb	Pbests1∼5	2*1	Gbest	2*1
UpdXV	Pbests1∼5	(2*1)*5	p1∼5	(2*1)*5
	Gbest	2*1		

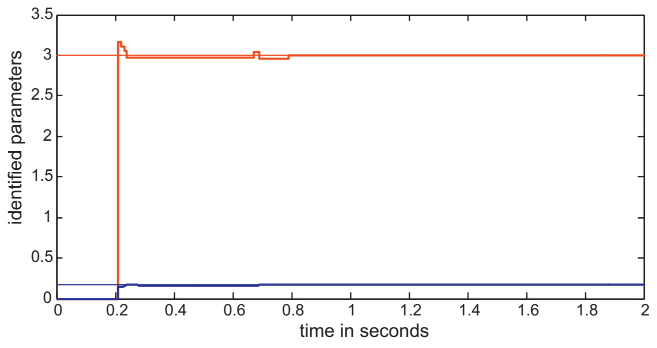

Fig. 7. Parameter identification at sample time of 10 μs.

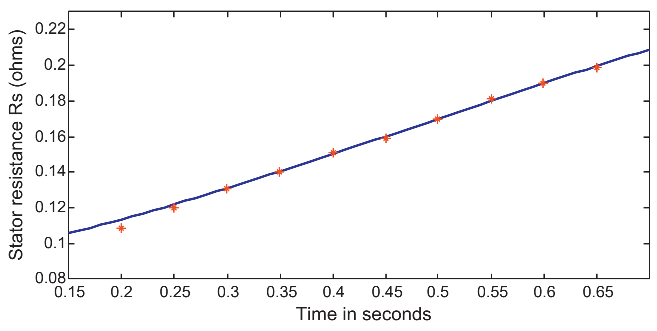

Fig. 9. Identification of the stator resistance R_s.

5.2. Simulation results

The real-time PSO based identification algorithm is evaluated in two types of operating conditions, i.e. identification of time-invariant parameters and identification of time-varying parameters. Results and discussion are given in following sessions.

5.2.1. Identification of time-invariant parameters

Figs. 7 and 8 show simulation results under a sample time of 10 μs. For the 10 μs sample time, the identified parameters will be updated every 0.05 s (1000 samples × 10 μs sample time × 5 iterations). If 100 μs sample time is used, the PSO algorithm will update the identified parameters for every 0.5 s. The realizable updating frequency is decided by the capability of controller hardware and the complexity of system model.

From Fig. 7, it can be seen that the PSO algorithm is not running during the first 0.2 s to avoid the startup transient. Once it is applied, the algorithm is able to provide good estimation of the unknown parameters $[R_s, T_{Ld}] = [0.17, 3]$ accurately. Since PSO initialize particles with randomly generated values, there is some estimation error at the beginning. The estimation accuracy will improve through time. To avoid the initial estimation error, the initial estimated parameters can be discarded.

To demonstrate the performance of the identification algorithm, a comparison between measurement and simulation data is provided in Fig. 8. In Fig. 8, the measured data are plotted using red solid lines and the simulated data are plotted using blue dash-dot lines. (For interpretation of the references to color in this text, the reader is referred to the web version of the article.) We can see that the two plots match very well, which means accurate identification.

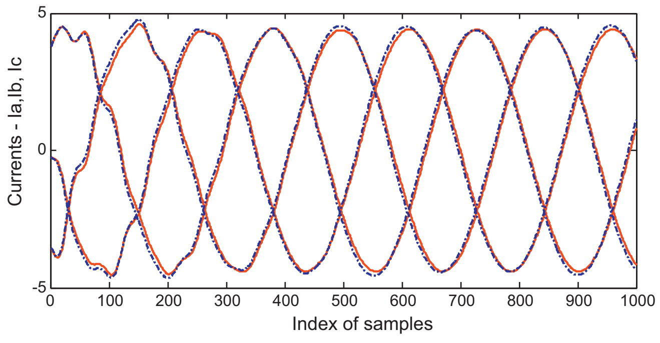

Fig. 8. Comparison between measured data and simulated data.

5.2.2. Identification of time-varying parameters

Figs. 9 and 10 show the identification of time-varying parameters. From these figures, it can be seen that the proposed technique can identify time-varying parameters successfully. Fig. 11 shows the entire optimization process of the PSO algorithm. For each run

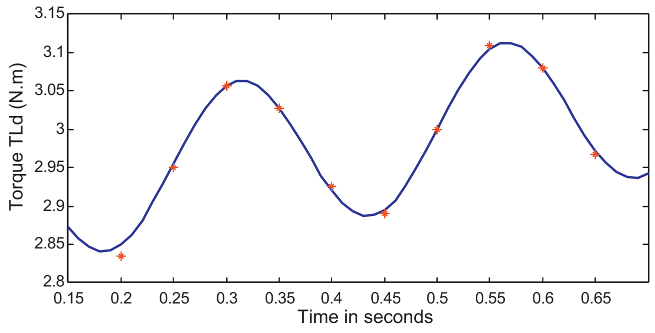

Fig. 10. Identification of the load torque T_{Ld}.

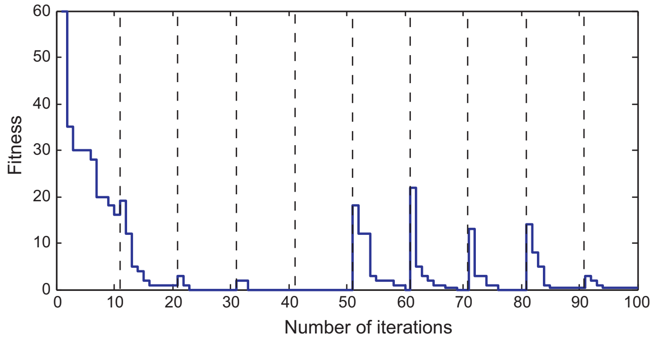

Fig. 11. Optimization process of the PSO based identification algorithm.

of the PSO based algorithm, the previously optimized parameters are used as the initial value. In doing so, the identification process for online applications tends to converge faster in terms of less number of iterations. This is especially true for systems with slowly time-varying parameters. From Fig. 11, it can be seen that the algorithm usually converges within 5 iterations because of a good initial guess. But from the figure we can also see that the initial fitness for the runs of 5, 6, 7, and 8 are not as good as others. This is because the fitness's sensitivity to parameter changes is different at different operating points. Sometimes, even a small change in the parameters will result in a large change in the fitness.

6. Conclusion

In this paper, PSO based identification algorithm is implemented online through faster than real-time simulation and advanced programming. The algorithm is implemented using the most fundamental Simulink blocks and Embedded Matlab Functions and the program can be compiled to C code through Real-time Workshop. The generated C code can run in most modern hardware controllers such as the dSPACE controller. The proposed technique is applied to concurrently identify two parameters in PMSM control system. Simulation results show that the algorithm can successfully identify both time-invariant and time-varying parameters.

Since PSO is a random search based optimization algorithm, there is no guarantee that the algorithm can find the optimal solution within a given number of iterations, even for offline implementation. However, similar problem exists for most existing optimization algorithms. Since PSO based identification does not require that system model to satisfy some particular conditions, PSO can solve many problems that are difficult for traditional algorithms to handle. If the parameters to identify do not change severely, previously identified parameters can be used as a good initial guess for next round of identification. By doing this, the convergence speed of the optimization process can be significantly increased. This property is very similar to the idea of continuously online trained neural network based identifier. Simulation results in Section 5 show that PSO based identification algorithm can provide good identifications consistently. The proposed techniques can be implemented with most modern popular hardware controllers and can be extended to a wide range of online identification and optimization problems.

There are two frequently asked questions about papers related to PSO applications. One is that why PSO is used not one of many other computational intelligence algorithms like genetic algorithm, ant colony, memetic algorithm, differential evolution, etc. The other is the request to compare performance between different algorithms. The comparative study between PSO and other type of algorithms is not conducted in this paper for the following reason. First, every popular type of algorithm has many new versions and improvements, which becomes available every day. Even for some specific version, different parameter setting and implementation will lead to different performance. If a comparative study is conducted, the conclusion can only be that one specific version of *A algorithm* performs better than another specific version of *B algorithm* under specific conditions. It will be problematic and inconvincible to claim one algorithm is better than the other algorithm. Since it is impossible to cover all recent developments of one type of algorithm, quantification study may not be as good as qualitative analysis. Some very good comparative studies can be found in [36–39]. In this paper, the authors chose to constrain the scope of the paper to the real time implementation of one algorithm (PSO) of the basic version. The implementation details can be easily modified according to different versions of PSO and can also help the real time implementation of other computational intelligence algorithms.

References

[1] A. Caponio, G.L. Cascella, F. Neri, N. Salvatore, M. Sumner, A fast adaptive memetic algorithm for off-line and on-line control design of PMSM drives, IEEE Transactions on Systems, Man and Cybernetics – Part B, Special Issue on Memetic Algorithms 37 (1) (2007) 28–41.
[2] F. Neri, X. del Toro Garcia, G.L. Cascella, N. Salvatore, Surrogate assisted local search on PMSM drive design, International Journal for Computation and Mathematics in Electrical and Electronic Engineering 27 (3) (2008) 573–592.
[3] A. Caponio, F. Neri, V. Tirronen, Super-fit control adaptation in memetic differential evolution frameworks, soft computing-a fusion of foundations, Methodologies and Applications, Springer 13 (8) (2009) 811–831.
[4] Y. Fujishima, S. Wakao, A. Yamashita, T. Katsuta, Design optimization of a permanent magnet synchronous motor by the response surface methodology, Journal of Applied Physics 91 (10) (2002) 8305–8307.
[5] F. Cupertino, F. Mininno, D. Naso, B. Turchiano, L. Salvatore, On-line genetic design of anti-windup unstructured controllers for electric drives with variable load, IEEE Transactions on Evolutionary Computation 8 (4) (2004) 347–364.
[6] D.C. Aliprantis, S.D. Sudhoff, B.T. Kuhn, Genetic algorithm-based parameter identification of a hysteretic brushless exciter model, IEEE Transactions on Energy Conversion 21 (1) (2006) 148–154.
[7] B.N. Cassimere, S.D. Sudhoff, Population-based design of surface-mounted permanent magnet synchronous machines, IEEE Transactions on Energy Conversion 24 (2) (2009) 338–346.
[8] J. Cale, S.D. Sudhoff, R.R. Chan, Ferrimagnetic inductor design using population-based design algorithms, IEEE Transactions on Magnetics 45 (2) (2009) 726–734.
[9] R.R. Chan, Y. Lee, S.D. Sudhoff, E.L. Zivi, Evolutionary optimization of power electronics based power systems, IEEE Transactions on Power Electronics 23 (4) (2008) 1907–1917.
[10] C. Kwon, S.D. Sudhoff, Genetic algorithm-based induction machine characterization procedure with application to maximum torque per amp control, IEEE Transactions on Energy Conversion 21 (2) (2006) 405–415.
[11] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, vol. 4, Perth, Australia, 1995, pp. 1942–1948.
[12] Y. Shi, Particle Swarm Optimization, Feature article, Electronic Data Systems, Inc., IEEE Neural Networks Society, 2004.
[13] J. Kennedy, R.C. Eberhart, Y. Shi, Swarm Intelligence, Morgan Kaufmann Publishers, San Francisco, 2001.
[14] Q. Lewei, L. Liu, D.A. Cartes, A reconfigurable and flexible experimental footprint for control validation in power electronics and power systems research, in: IEEE Power Electronics Specialists Conference, 17–21, 2007, pp. 1995–2000.

[15] W. Liu, J. Sarangapani, G.K. Venayagamoorthy, L. Liu, D.C. Wunsch, M.L. Crow, D.A. Cartes, Decentralized neural network based excitation control of large scale power systems, International Journal of Control, Automation, and Systems 5 (5) (2007) 526–538.
[16] L. Liu, W. Liu, D.A. Cartes, Particle swarm optimization based parameter identification applied to permanent magnetic synchronous machine, Engineering Applications of Artificial Intelligence 21 (7) (2008) 1092–1100.
[17] G. Harik, F.G. Lobo, D.E. Goldberg, The compact genetic algorithm, IEEE Transactions on Evolutionary Computation 3 (1999) 287–297.
[18] Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, Proceedings of Evolutionary Programming VII (EP98) (1998) 591–600.
[19] R.C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, Proceedings of the Congress on Evolutionary Computation 1 (2000) 84–88.
[20] G. Carlisle, Dozier, An off-the-shelf PSO, in: Proceedings of the Particle Swarm Optimization Workshop, 2001, pp. 1–6.
[21] F. van den Bergh, An analysis of particle swarm optimizers, PhD Dissertation, University of Pretoria, 2001.
[22] M. Clerc, J. Kennedy, The particle swarm – explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (1) (2002) 58–73.
[23] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters 85 (6) (2003) 317–325.
[24] J. Kennedy, R. Mendes, Neighborhood topologies in fully informed and best-of-neighborhood particle swarms, IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews 36 (4) (2006) 515–519.
[25] J. Kennedy, In search of the essential particle swarm, in: Proceedings of 2006 IEEE Congress on Evolutionary Computations, Vancouver, BC, Canada, July 16–21, 2006.
[26] D. Bratton, T. Blackwell, A simplified recombinant PSO, Journal of Artificial Evolution and Applications 2008 (1) (2008) 1–10.
[27] M.E.H. Pedersen, Tuning and Simplifying Heuristical Optimization, PhD Dissertation, University of Southampton, 2010.
[28] M.E.H. Pedersen, A.J. Chipperfield, Simplifying particle swarm optimization, Applied Soft Computing 10 (2) (2010) 618–628.
[29] A. Banks, J. Vincent, C. Anyakoha, A review of particle swarm optimization. Part I. Background and development, Natural Computing 6 (4) (2007) 467–484.
[30] A. Banks, J. Vincent, C. Anyakoha, A review of particle swarm optimization. Part II. Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications, Natural Computing 7 (1) (2008) 109–124.
[31] K. Kameyama, Particle swarm optimization – a survey, IEICE Transactions on Information and Systems E92-D (7) (2009) 1354–1361.
[32] Y. del Valle, G.K. Venayagamoorthy, S. Mohagheghi, J. Hernandez, R.G. Harley, Particle swarm optimization: basic concepts, variants and applications in power systems, IEEE Transactions on Evolutionary Computation 12 (2) (2008) 171–195.
[33] M.R. AlRashidi, M.E. El-Hawary, A survey of particle swarm optimization applications in electric power systems, IEEE Transactions on Evolutionary Computation 13 (4) (2009) 913–918.
[34] R. Krishnan, Electric Motor Drives Modelling, Analysis and Control, Prentice Hall, Englewood Cliffs, NJ, 2001.
[35] S.E. Lyshevski, Electromechanical Systems, Electric Machines, and Applied Mechatronics, CRC Press LLC, 2000.
[36] S. Das, A. Abraham, A. Konar, Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives, Studies in Computational Intelligence 116 (2008) 1–38.
[37] R. Hassan, B. Cohanim, O. de Weck, A comparison of particle swarm optimization and the genetic algorithm, in: 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Austin, Texas, April 18–21, 2005.
[38] N. Chakraborti, S. Das, R. Jayakanth, R. Pekoz, S. Erkoc, Genetic algorithms applied to Li+ ions contained in carbon nanotubes: an investigation using particle swarm optimization and differential evolution along with molecular dynamics, Materials and Manufacturing Processes 22 (5-6) (2007) 562–569.
[39] N. Chakraborti, R. Jayakanth, S. Das, E.D. Calisir, S. Erkoc, Evolutionary and genetic algorithms applied to Li+–C system: Calculations using differential evolution and particle swarm algorithm, Journal of Phase Equilibria and Diffusion 28 (2) (2007) 140–149.

Wenxin Liu received his Ph.D. degree in Electrical Engineering from the Missouri University of Science and Technology (formerly University of Missouri – Rolla) in 2005 and B.S. and M.S. degrees from Northeastern University (China) in 1996 and 2000 respectively. He was an Assistant Scholar Scientist with the Center for Advanced Power Systems, Florida State University, from 2005 to 2009. Currently, he is an Assistant Professor with the Klipsch School of Electrical and Computer Engineering, New Mexico State University. His current research interests include intelligent control of power systems, control and optimization of microgrids, computational intelligence, and fault detection and diagnosis.

Li Liu received her Ph.D. degree in Mechanical Engineering from Florida State University (FSU) in 2006. She got the BS in Mechanical Engineering from Tong Ji University, Shanghai, China in 1997; and the M.S. in the same field from Shanghai Jiao Tong University, Shanghai, China in 2000. She then joined General Motors (Shanghai) as a Product Engineer and worked there for 2 years before pursuing her Ph.D. degree. She is now a Senior Control Engineer at Cummins Inc. Her research interests are condition based maintenance, fault detection, diagnosis, nonlinear system control and identification for electric machines.

Il-Yop Chung received his B.S., M.S., and Ph.D. degrees in electrical engineering from Seoul National University, Seoul, Korea, in 1999, 2001, and 2005, respectively. He was a Postdoctoral Associate at Virginia Polytechnic Institute and State University, Blacksburg, VA, from 2005 to 2007. He was also a Visiting Researcher at ABB US Corporate Research Center, Raleigh, NC in 2007. From 2007 to 2010, he worked for the Center for Advanced Power Systems (CAPS) at Florida State University, Tallahassee, FL as a PostDoc Associate and an Assistant Scholar Scientist. Currently, he is a Fulltime Lecturer at Kookmin University, Seoul, Korea. His research interests are power system control, distributed energy resources, renewable energy, and shipboard power systems.

David A. Cartes received the Ph.D. degree in engineering science from Dartmouth College, Hanover, NH. He is an Associate Professor of the Department of the Mechanical Engineering at Florida State University (FSU), Tallahassee, FL from January 2001. He directs the Institute for Energy Systems, Economics and Sustainability (IESES) at FSU. His research interests include energy management systems, energy policy and energy business practices, having published over 100 peer reviewed papers. In 1994, he completed a 20-year U.S. Navy career with experience in operation, conversion, overhaul, and repair of complex marine propulsion systems.