




Research on Active Disturbance Rejection Control With Parameter Autotune Mechanism for Induction Motors Based on Adaptive Particle Swarm Optimization Algorithm With Dynamic Inertia Weight

Chao Du, *Student Member, IEEE*, Zhonggang Yin , *Member, IEEE*, Yanping Zhang , Jing Liu ,
Xiangdong Sun, and Yanru Zhong

Abstract—An active disturbance rejection control (ADRC) of induction motor based on an adaptive particle swarm optimization (APSO) algorithm is proposed in this paper, in order to realize the precise decoupling of induction motor and the disturbance compensation. The novel control method employs APSO as the automatic tune mechanism for ADRC controller. According to the feedback information of induction motor, an optimal solution can be achieved via the optimization mechanism and self-learning ability of APSO, so the reliance of ADRC controller on parameters can be reduced. In order to obtain the better optimization solution more efficient, the aggregation degree and the evolution speed are introduced into the APSO to dynamically modify the inertia weight based on the practical optimization process. Experimental results indicate that the robustness of the proposed optimal design method for ADRC is better than the conventional ADRC when the disturbances occur, and the method is feasible and effective.

Index Terms—Active disturbance rejection control (ADRC), adaptive particle swarm, induction motor (IM), parameter optimization, robustness.

I. INTRODUCTION

INDUCTION MOTORS (IMs) can be controlled similarly as dc motors using the field-oriented control (FOC) approach, and the control performance index for the controlled IMs based on FOC is comparable to those of the dc motors. IM drives employing FOC and conventional proportional-integral regulators have been commercialized. However, the control performance index for an IM drive still remains a challenging problem due

to the variation of the parameters, the highly nonlinear nature of the drive, and the characteristics such as time lag and complexity. With the rapid development of a control theory in recent years, several advanced algorithms have been proposed to improve the control performance index for the IM control system. For example, sliding model control [1]–[4], adaptive control [5]–[7], internal model control [8]–[10], active disturbance rejection control (ADRC), and other advanced control technologies have been applied to achieve better control performance index.

ADRC method is a nonlinear control for uncertain systems. It can estimate and compensate the external disturbances, and it is independent on the model of the object and insensitive on the variation of the system parameters. Therefore, it is the best method for solving the control problems of the nonlinear system in the control field. As one of the robust control methods to deal with the uncertainty, ADRC is applied in many technical fields [11]–[13], particularly in the field of the motor control [14]–[18]. Many ADRC-based research works have been done to improve the robustness of the IMs. The ADRC for the IM drives has the advantage of good robustness to the external disturbance. In [14], a linear active disturbance rejection controller is applied for sensor-less control of internal permanent magnet synchronous motors to address the phase delay and speed chattering. In [15], a nonlinear autodisturbance rejection controller (ADRC) is utilized to ensure the good robustness and adaptability when the modeling uncertainty and external disturbance occur in the IM control system. In [18], a fractional-order $PI^\lambda D^\mu$ and an active disturbance rejection controller are combined to regulate the speed of the system. Most of the aforementioned works based on the conventional ADRC are limited to the applied technical fields and the experimental results. These existing ADRC methods do not illustrate how to tune the parameters of ADRC controller. Regardless of the structure of the controller, its parameters must be carefully tuned in order to guarantee to deliver an appropriate control signal. Therefore, tuning parameters is an important step in the system design because it has a direct and significant impact on the system response control performance

Manuscript received February 12, 2017; revised April 28, 2018; accepted May 17, 2018. Date of publication May 28, 2018; date of current version February 5, 2019. This work was supported in part by the National Natural Science Foundation of China (51677150), in part by the Specialized Research Fund of Shaan Xi Province (2015KJXX-29), and in part by the State Key Laboratory of Large Electric Drive System and Equipment Technology (SKLLDJ012016006). Recommended for publication by Associate Editor I. Slama-Belkhdja. (Corresponding author: Zhonggang Yin.)

The authors are with the Xi'an University of Technology, Xi'an 710048, China (e-mail: duchaoworkhard@163.com; zhgyin@xaut.edu.cn; zhangyan-pingmao@126.com; jingliu@xaut.edu.cn; sxd1030@163.com; zhonggyr@xaut.edu.cn).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TPEL.2018.2841869

index. Generally, the designer of the ADRC controller adopts a trial-and-error tune method. However, the process of manually tuning ADRC parameters is time consuming and requires a significant design effort, and it is difficult to obtain satisfactory control performance. Furthermore, according to the different controlled objects, the range of some parameters is completely different. If the parameters are not tuned properly, the control performance index of ADRC controller will be affected and even limit the wide application of an ADRC method. Thus, it is necessary to find out a method to tune the parameters because tuning the ADRC parameters is not practical and convenient manually.

In order to simple the design and improve control performance, a particle swarm optimization algorithm (PSO) is usually utilized to solve the predicaments [19]–[21]. PSO is often trapped into local optimum, while dealing with complex and practical design problems so that the optimal result is not the best. The main reason why the local optimum appears is that the inertia weight is an important parameter of PSO, and it is the tradeoff between the global and local search ability. Meanwhile, the inertia weight constitutes a forgetting factor related to the previous results. Therefore, if the improper value of inertia weight is selected, PSO usually prefers the local search resulting in the local optimum eventually. In order to obtain the proper value of inertia weight, the linearly decreasing inertia weight (LDW) with increasing iteration is commonly used to avoid the premature convergence and local optimum [21], [22]–[25]. However, the practical optimization problems are nonlinear and complicated, and the optimization process is more nonlinear when the optimization problems are optimized by PSO. LDW cannot adapt to the complex and nonlinear characteristics when PSO is operated. As a result, it results in the inefficient search and the local optimal for the complex optimization problems, especially for the complicated IM control system in [26], Ant colony optimization algorithm (ACO) is more effective and more efficient than PSO based on the comparison results. However, as to the computational efficiency, ACO is still an inefficient algorithm during the parallel calculation, although the amount of calculation has reduced. Meanwhile, the constant pheromone evaporation of ACO goes against the local precise search and the fast convergence so that the convergence speed is still influenced, and the efficiency of ACO is degraded. Therefore, it is necessary to dynamically modify the inertia weight based on the practical operation process of PSO. The evolution speed and the aggregation degree indicate the optimization process of PSO among the particles, and they reveal the effectiveness of the particles movement and the variety of particles during the optimization, respectively. In this paper, an adaptive mechanism of the inertia weight is designed and introduced into PSO, and it is named by active disturbance rejection control (APSO). The adaptive inertia weight is dynamically modified based on the practical evolution speed and the aggregation degree to obtain an ideal inertia weight in each iteration time. Meanwhile, with the dynamic modification of the inertia weight, APSO has faster convergence speed and faster optimization efficiency than ACO and PSO.

A novel ADRC method which employs APSO is proposed in this paper, and it is named as APSO-ADRC. With the

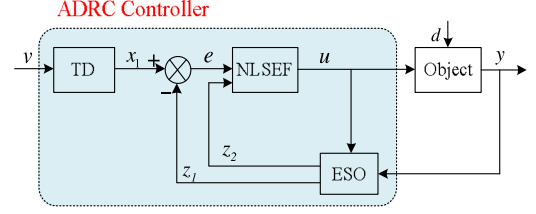


Fig. 1. Structure of ADRC controller.

optimization mechanism of APSO, the key parameters of ADRC can be optimized by APSO automatically so that APSO-ADRC has less reliance on the parameters of ADRC than the conventional ADRC. In order to avoid the premature convergence of the local optimum solutions, an adaptive inertia weight of APSO-ADRC is designed and can be dynamically modified. Based on the aggregation degree and the evolution speed of particles, the novel adaptive inertia weight can make the optimal solution more reasonable and more efficient. The experimental results show the feasibility and effectiveness of the proposed method.

II. ACTIVE DISTURBANCE REJECTION CONTROL OF INDUCTION MOTORS

The state equations of IMs based on d - q coordinates are given in (1). With the nonlinear strategy, ADRC transforms a nonlinear object which has unknown disturbance into a series-connection integral object, and treats the internal disturbance and the external disturbance as the total system disturbance. The total system disturbance is estimated, and it is compensated for the estimated model. Based on the principles of ADRC, load disturbance, coupling parts, and modeling, uncertainties existing in the state equations can be treated as the total system disturbance and compensated by ADRC

$$\begin{cases} \dot{\omega}_r = \frac{n_p}{J} \frac{L_m}{L_r} \psi_r i_{sq} - \frac{n_p}{J} T_L \\ \dot{i}_{sq} = -\frac{L_m}{\sigma L_s L_r} \omega_r \psi_r - \frac{R_s L_r^2 + R_r L_m^2}{\sigma L_s L_r^2} i_{sq} - \omega_s i_{sd} + \frac{u_{sq}}{\sigma L_s} \\ \dot{i}_{sd} = \frac{L_m R_r}{\sigma L_s L_r^2} \psi_r - \frac{R_s L_r^2 + R_r L_m^2}{\sigma L_s L_r^2} i_{sd} + \omega_s i_{sq} + \frac{u_{sd}}{\sigma L_s} \\ \dot{\psi}_r = -\frac{R_r}{L_r} \psi_r + \frac{L_m R_r}{L_r} i_{sd} \end{cases} \quad (1)$$

where ω_r is the rotor angular velocity, ω_s is the synchronous angular velocity, u_{sd} is the direct component of stator voltage, u_{sq} is the quadrature component of stator voltage, i_{sd} is the direct component of stator current, and i_{sq} is the quadrature component of stator current.

ADRC controller is composed of three parts: tracking differentiator (TD), extended state observer (ESO), and nonlinear state error feedback control law (NLSEF), and Fig. 1 shows the structure of ADRC controller. In Fig. 1, v is the input signal of the controller, u is the output signal of the controller, y is the output signal of the object, and d is the system disturbance.

A. Tracking Differentiator

TD arranges the transition process and smooths the sharp changes in the input signal. The discrete mathematical model of

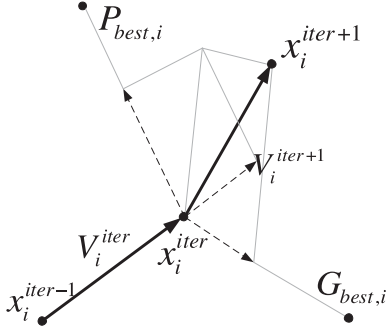


Fig. 3. Optimization movement of particles in the search space.

B. Adaptive Particle Swarm Algorithm

1) *Basic Principles of PSO*: PSO is inspired by the social behavior of bird flocking, and it is a kind of algorithm based on population. Each particle has a position X and a velocity V in the search space. During the optimization process, the position and the velocity of each particle are modified based on the present velocity, the present position, the individual optimal position obtained by the particle itself, and the global optimal position obtained by the whole particle swarm. Therefore, each particle has the tendency toward the optimal position. The modified process of each particle is shown in Fig. 3. The velocity information and the position information are updated according to

$$V_i^{iter+1} = w \cdot V_i^{iter} + c_1 \cdot r_1 \cdot (P_{best,i} - X_i^{iter}) + c_2 \cdot r_2 \cdot (G_{best} - X_i^{iter}) \quad (7)$$

$$X_i^{iter+1} = X_i^{iter} + V_i^{iter+1} \quad (8)$$

where i denotes the i th particle in the swarm, $iter$ denotes the present iteration times, $P_{best,i}$ denotes the individual best position obtained by the i th particle, G_{best} denotes the global best position among the swarm, c_1 and c_2 are the cognitive factor and social factor, respectively, and they have the influence of $P_{best,i}$ and G_{best} on the updated position, r_1 and r_2 are random numbers between zero and 1, and w is the inertia weight.

The inertia weight w is the tradeoff between the global and local search ability of particles. A bigger value of inertia weight improves the global search ability of PSO, and a smaller value of inertia weight improves the local search ability of PSO. An LDW with increasing iteration is commonly used to avoid the premature convergence of algorithm and local optimum, and w is dynamically updated and calculated by

$$w = w_{ini} - \frac{iter}{iter_{max}} \cdot (w_{ini} - w_{fin}) \quad (9)$$

where $iter_{max}$ denotes the maximum number of iterations. $iter$ denotes the present number of iteration. w_{ini} denotes the initial value of inertia weight, and its value is 1. w_{fin} denotes the final value of inertia weight, and its value is 0.5. In the LDW function, the linearly updated inertia weight is closely associated with the increasing iteration times, and LDW is only effective to solve some optimization problems. However, as to the different optimization problems, the ideal relations between the inertia weight and the iteration times are not always the same. The

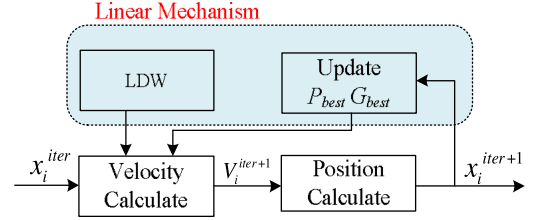


Fig. 4. Block diagram of PSO.

search process is always nonlinear and highly complicated for the practical optimization problems, and LDW cannot adapt to the complex and nonlinear characteristics. As a result, it results in the inefficient search and the local optimum for the complex optimization problems, especially for the complicated IM control system in this paper. The block diagram of a PSO is shown in Fig. 4.

In order to solve the nonlinear optimization problems effectively, an adaptive mechanism of the inertia weight is introduced into the updated process of the inertia weight so that the adaptive inertia weight is dynamically modified to keep the good balance between the global and the local search ability of particles. The adaptive inertia weight is decided by the evolution speed and the aggregation degree. The evolution speed and the aggregation degree have great influence on the optimization performances of algorithm, and they are further explained in Section III-B2 and III-B3.

2) *Evolution Speed*: The global optimal depends on the variation of the local optimum, and it also shows the effectiveness of the optimization movement among all the particles.

In the optimization process of iteration, the global optimal value $F(G_{best}^T)$ of the present iteration times T is always better than or equal to the global optimal value $F(G_{best}^{T-1})$ of the previous iteration times $T - 1$. In details, there are two cases to be discussed and defined the evolution speed.

1) If the maximum value of the optimization problem is searched, $F(G_{best}^{T-1}) \leq F(G_{best}^T)$. The evolution speed h is defined as

$$h = \frac{F(G_{best}^{T-1})}{F(G_{best}^T)}. \quad (10)$$

2) If the minimum value of the optimization problem is searched, $F(G_{best}^{T-1}) \geq F(G_{best}^T)$. The evolution speed h is defined as

$$h = \frac{F(G_{best}^T)}{F(G_{best}^{T-1})}. \quad (11)$$

Based on the two different cases above, the evolution speed h can be defined in summary as

$$h = \frac{\min(F(G_{best}^{T-1}), F(G_{best}^T))}{\max(F(G_{best}^{T-1}), F(G_{best}^T))}. \quad (12)$$

The history data of the global optimal are utilized to analyze the evolution speed of particles. A smaller value of evolution speed indicates the evolution speed is faster. The value of evo-

lution speed gradually comes to converge after a certain number of iteration times until the optimal solution is found.

3) *Degree of Aggregation*: At the present iteration times T , the global optimal value $F(G_{best}^T)$ is always better than the average fitness value F_T of all the particles. F_T can be calculated based on

$$F_T = \frac{1}{n} \sum_{i=1}^n F(X_T[i]) \quad (13)$$

where $X_T[i]$ denotes the position of the i th particle at the iteration times T , and n is the number of the particles in a swarm. In details, there are two cases to be discussed and defined the aggregation degree.

1) If the maximum value of the optimization problem is searched, $F(G_{best}^T) \geq F_T$. At this time, the aggregation degree s is defined as

$$s = \frac{F_T}{F(G_{best}^T)}. \quad (14)$$

2) If the minimum value of the optimization problem is searched, $F(G_{best}^T) \leq F_T$. At this time, the aggregation degree s is defined as

$$s = \frac{F(G_{best}^T)}{F_T}. \quad (15)$$

Based on the two different cases above, the aggregation degree s can be defined in summary as

$$s = \frac{\min(F(G_{best}^T), F_T)}{\max(F(G_{best}^T), F_T)}. \quad (16)$$

The aggregation degree reflects the aggregation degree and the variety of all the particles. A bigger value of the aggregation degree indicates the aggregation of particles and the variety of particles is higher and less, respectively. All the particles have the same identity when s is equal to 1 in particular. At this time, it is easy to be involved into local optimum because all the particles are converged to one position.

4) *Adaptive Inertia Weight*: LDW is often trapped into local optimum while dealing with complex and practical design problems, because the improper value of the inertia weight is selected. Therefore, it is necessary to dynamically modify the inertia weight based on the practical operation process. The evolution speed and the aggregation degree indicate the optimization process among the particles, and they reveal the effectiveness of the particles movement and the variety of particles during the optimization, respectively. Therefore, an adaptive dynamic inertia weight is introduced into PSO, and it is named after APSO. In APSO, the adaptive inertia weight is dynamically modified based on the evolution speed and the aggregation degree to obtain an ideal inertia weight in each iteration time. The block diagram of APSO is shown in Fig. 5, and it forms the feedback control so that it is more reliable and more stable than PSO.

The evolution speed and the aggregation degree show the practical movement process and characteristics of particles. Therefore, the adaptive inertia weight should be varied based on the evolution speed and the aggregation degree, and its function

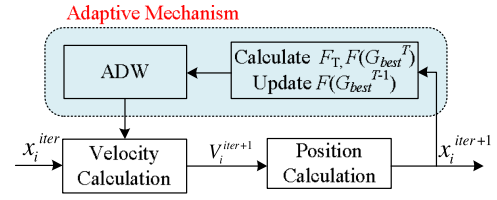


Fig. 5. Block diagram of APSO.

is given by

$$w = f(h, s). \quad (17)$$

As to the evolution speed, if the evolution speed is faster, particles have the stronger ability to continuously search the optimal solution in a larger range of search space. When the evolution speed slows down, the inertia weight should be decreased to make the particles search in a smaller range of search space so that the optimal result is found as soon as possible. As to the degree of aggregation, if the aggregation degree is dispersive, it is not easy to be involved into the local optimum. With the improvement on the aggregation degree, particles are gradually easy to be involved into local optimum. At this time, the range of search space should be enlarged to improve the global search ability of particles. In conclusion, the adaptive inertia weight should be decreased with the decrease of the evolution speed, and it should be increased with the increase of the aggregation degree. The improved adaptive inertia weight is given by

$$w = w_{ini} - h \cdot w_h + s \cdot w_s \quad (18)$$

where w_{ini} denotes the initial value of inertia weight, w_h is the evolution speed, and w_s is the aggregation degree. Based on the ranges of the evolution speed and the aggregation degree, the range of adaptive inertia weight is between $w_{ini} - w_h$ and $w_{ini} + w_s$. The best values of w_{ini} , w_h , and w_s are recommended as 1, 0.5, and 0.05, respectively.

C. Implementation of APSO Algorithm to ADRC

In this paper, APSO is implemented to optimize the parameters of ADRC controller in the IM control system. In details, the key parameters $\beta_1 = \text{diag}\{\beta_{1s}, \beta_{1q}, \beta_{1d}\}$ and $\beta_2 = \text{diag}\{\beta_{2s}, \beta_{2q}, \beta_{2d}\}$ in (4) are optimized via APSO under the consideration of the stability and accuracy of the system. Thus, an objective (fitness) function should be carefully selected, because it is the basis for updating the velocity and the position of particles.

Generally, the control performance index of IM includes dynamic and stable performances. The dynamic performance of IM mainly includes speed overshoot, rise time, and settling time, and the stable performance mainly includes stable error. Therefore, the dynamic and stable control performance indices can be used as constraint conditions to emphasize the desired response and to penalize undesired response, in order to obtain the ideal control performance. In the automatic control theory, each control performance index has a standard nomenclature, such as the speed overshoot is σ , the rise time is t_r , the settling time is t_s , and the stable error is e_{ss} . In this paper, it is considered as a

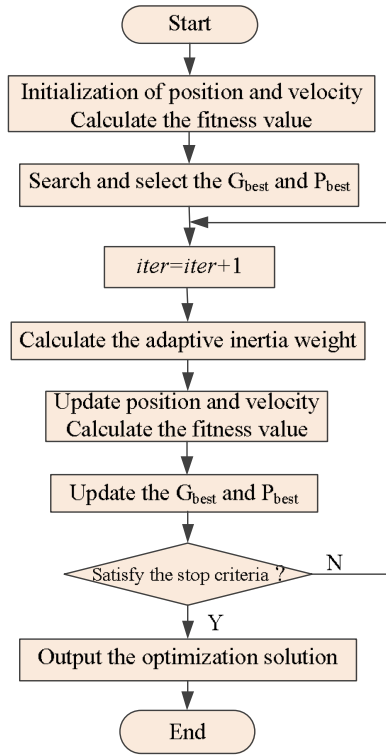


Fig. 6. Flowchart of adaptive particle swarm algorithm.

desired solution (parameters) if the speed overshoot σ is smaller than 1.5%, the rise time t_r is shorter than 0.2 s, the settling time t_s is shorter than 0.2 s, and the stable error e_{ss} is smaller than 0.01%. Therefore, the desired solution will be obtained eventually when the optimization finishes, and the objective (fitness) function Y can be composed of the weighted sum of the dynamic and stable control performance index as follows:

$$Y = a \cdot \sigma + b \cdot t_r + c \cdot t_s + d \cdot e_{ss} \quad (19)$$

where a , b , c , and d are the weights. Among the dynamic and stable control performance indices, they have a same characteristic that the smaller their value is, the better control performance index of the IM will be achieved. Thus, the ideal parameters can make the system has the best control performance index, and make the object function Y reach the minimum. In each iteration time, the optimization process of APSO is the process of changing the position of each particle toward the optimal position. In the experiment, the process of optimization in this paper is summarized as follows, and its flowchart is shown in Fig. 6.

Step 1: Initialize a population of particles with random positions and velocities in the 6-D search space using a random probability distribution function. The initial value of six dimensions is corresponding to the initial value of β_{1s} , β_{2s} , β_{1q} , β_{2q} , β_{1d} , and β_{2d} .

Step 2: Operate the IM under the control of the parameters from step 1. Collect and calculate the control performance index in the host computer when the IM operates, and they are the speed overshoot σ , the rise time is t_r , the settling time is t_s , and the stable error is e_{ss} . According to the collected control

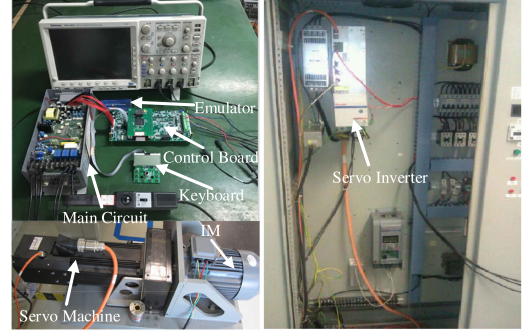


Fig. 7. Photograph of experimental setup.

performance index, evaluate the fitness value of each particle based on the objective function.

Step 3: Compare the position (parameter) of each particle with its previous P_{best} , and choose the better one as the present P_{best} . Among the present P_{best} of each particle, choose the best one to compare with the previous G_{best} , and the better one is selected as the present G_{best} .

Step 4: Calculate the evolution speed, the aggregation degree, and the adaptive inertia weight. The updated adaptive inertia weight is utilized to update the velocity information and the position information in step 5.

Step 5: Update the velocity and position information for each particle according to (7) and (8), respectively.

Step 6: Check the stop criterion. A maximum number of iteration times $iter_{max}$ or a desired fitness is selected as the stop criterion. If either the iteration times or the desired fitness is not matched with the stop criterion, go to step 3, otherwise output the parameters β_{1s} , β_{2s} , β_{1q} , β_{2q} , β_{1d} , and β_{2d} , and stop the algorithm.

IV. EXPERIMENTAL RESULTS

The photograph of experimental setup is shown in Fig. 7. The experimental setup consists of the main circuit and the control board. The main circuit and the control board act as an inverter to control the IM. The step load torque from the servo machine is controlled by the servo inverter. ADRC algorithm is written by the C language on the TMS320F28335 DSP which is in the control board. APSO is applied to tune the parameters offline in the host computer. The feedback information of the system is delivered to the host computer under the environment of VC++, and it makes use of Windows application program interface to realize USB interface communication method.

In order to verify the effectiveness of the proposed control method, the robustness of the proposed method has been investigated under three cases: 1) start up; 2) load disturbance; and 3) parameter variation. The parameters of IM are given in the Appendix.

A. Effectiveness Verification of ADRC

Fig. 8 shows that the IM without load torque operates to 30 r/min at 0 s, to 300 r/min at 10 s, to 900 r/min at 20 s, to 1500 r/min at 30 s, to 1200 r/min at 40 s, to 600 r/min at 50 s,

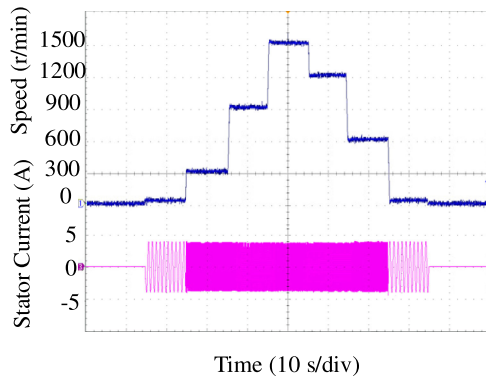


Fig. 8. Experimental results based on ADRC without load.

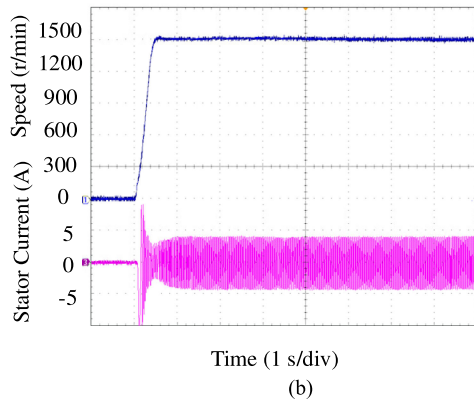
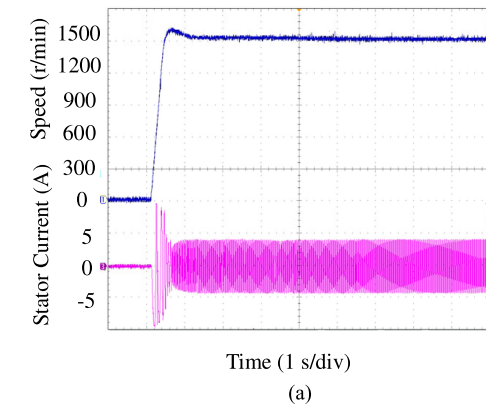


Fig. 9. Comparative results based on APSO-ADRC when IM starts up at 1500 r/min. (a) 25th iteration of APSO. (b) 52nd iteration of APSO.

to 30 r/min at 60 s, and stops operating at 70 s. ADRC has its superiority that it is effective in all over the speed range and load condition, and it does not need several parameter sets to obtain the desired speed control performance in different speed ranges.

B. Convergence Verification of APSO-ADRC

In the experiment based on APSO-ADRC, the 25th iteration optimization results and the 52nd iteration optimization results are selected to be compared to prove the convergence of APSO in Figs. 9–17. (Based on the experimental results as shown in

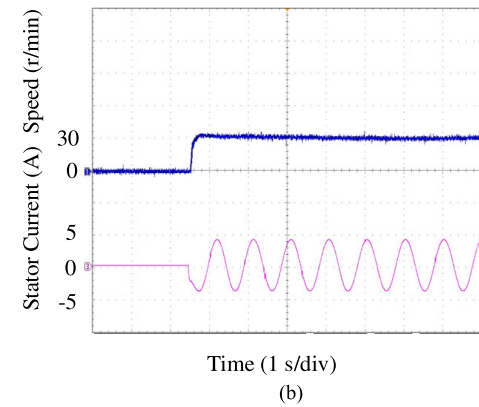
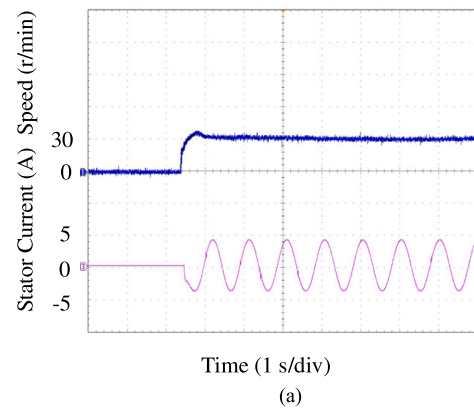
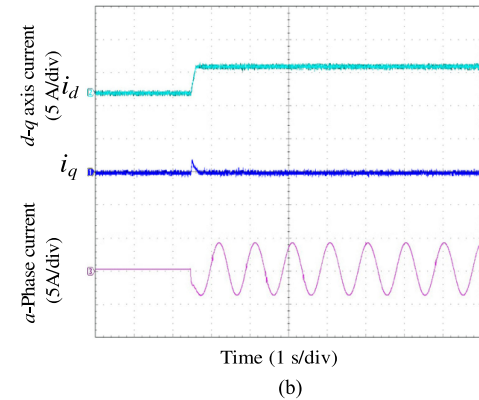
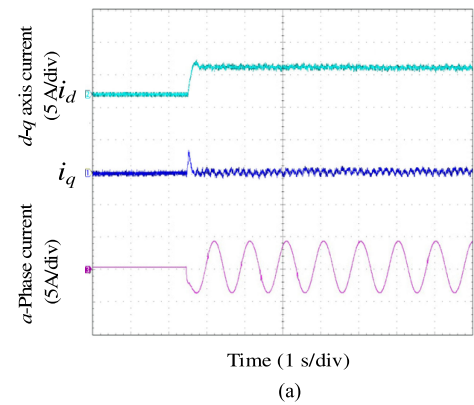


Fig. 10. Comparative results based on APSO-ADRC when IM starts up at 30 r/min. (a) 25th iteration of APSO. (b) 52nd iteration of APSO.

Fig. 11. d - q axis current decoupling based on APSO-ADRC at 30 r/min. (a) 25th iteration of APSO. (b) 52nd iteration of APSO.

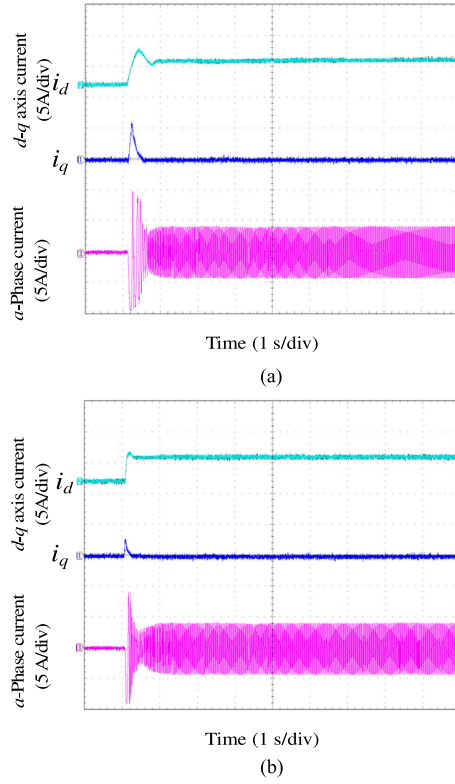


Fig. 12. d - q axis current decoupling based on APSO-ADRC at 1500 r/min. (a) 25th iteration of APSO. (b) 52nd iteration of APSO.

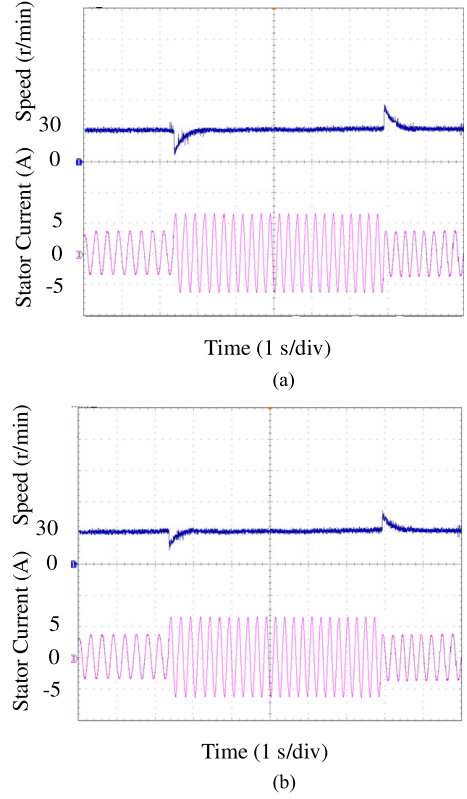


Fig. 14. Comparative experimental results based on APSO-ADRC with load disturbance at 30 r/min. (a) 25th iteration of APSO. (b) 52nd iteration of APSO.

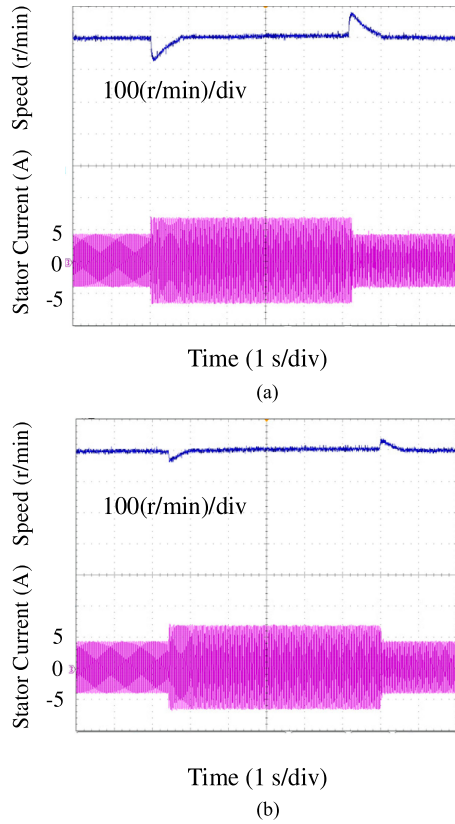


Fig. 13. Comparative experimental results based on APSO-ADRC with load disturbance at 1500 r/min. (a) 25th iteration of APSO. (b) 52nd iteration of APSO.

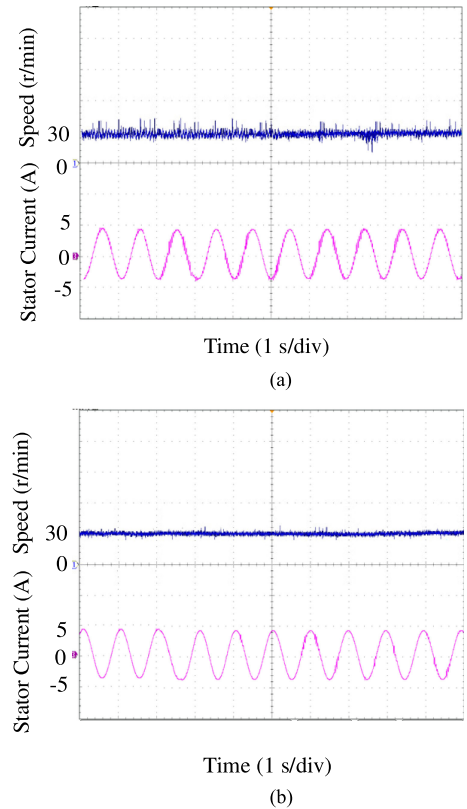


Fig. 15. Comparative experimental results based on APSO-ADRC with R_s variation at 30 r/min. (a) 25th iteration of APSO. (b) 52nd iteration of APSO.

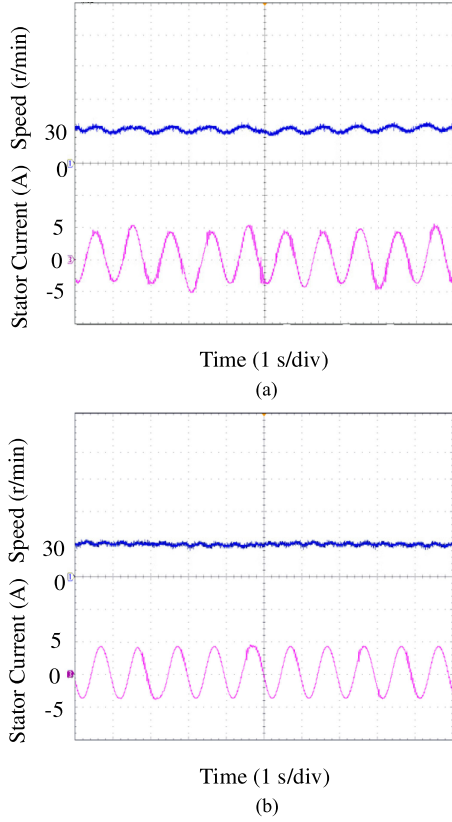


Fig. 16. Comparative experimental results based on APSO-ADRC with R_r variation at 30 r/min. (a) 25th iteration of APSO. (b) 52nd iteration of APSO.

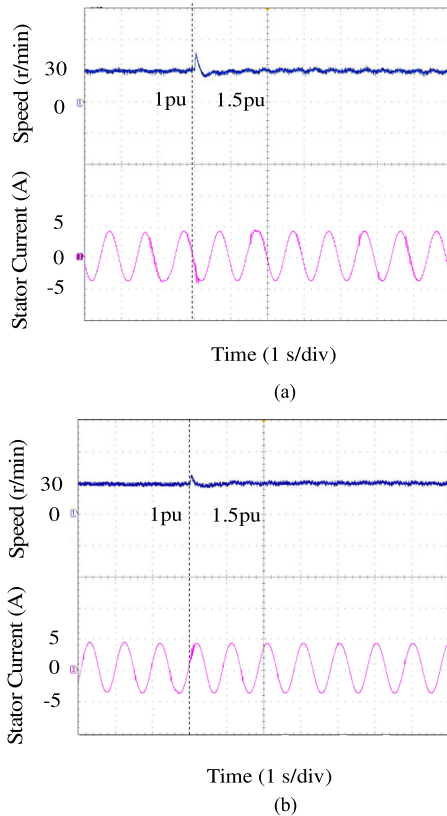


Fig. 17. Comparative experimental results based on APSO-ADRC with L_m variation at 30 r/min. (a) 25th iteration of APSO. (b) 52nd iteration of APSO.

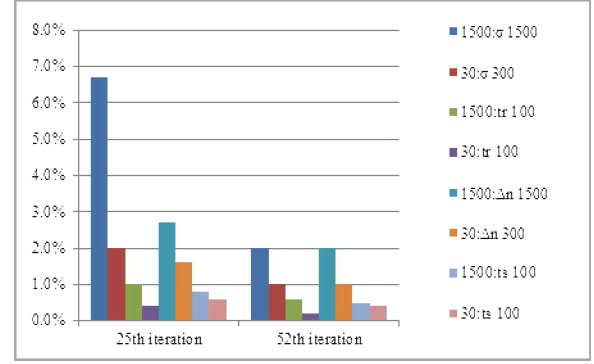


Fig. 18. Comparative histogram of experimental data between the results of 25th iteration and the results of 52nd iteration.

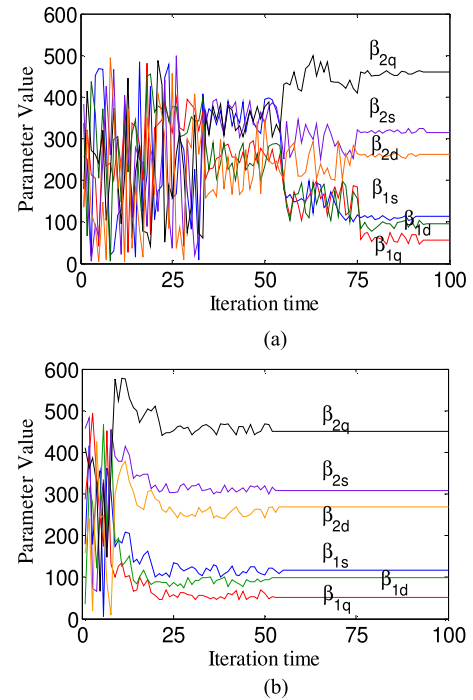


Fig. 19. Comparison of parameter convergence between PSO and APSO. (a) PSO. (b) APSO.

Fig. 19, the 25th iteration is the middle stage of optimization, and the 52nd iteration is the final iteration to find optimal results.)

1) *Start Up*: Figs. 9 and 10 show the comparative optimization results between the 25th iteration and the 52nd iteration, and the speed reference is 1500 r/min in Fig. 9 and 30 r/min in Fig. 10, respectively. The 52nd iteration optimization results have less overshoot and faster dynamic process than the 25th iteration optimization results in Fig. 9. Although the optimization results of the 25th iteration and the 52nd iteration are nearly the same in Fig. 10, the stator current of the 52nd iteration is more stable than that of the 25th iteration.

Figs. 11 and 12 show the d - q axis current decoupling of IM between the 25th iteration and the 52nd iteration, and the speed reference is 30 r/min in Fig. 11 and 1500 r/min in Fig. 12, respectively. The d - q axis current decoupling waveforms of IM have a larger oscillation in Fig. 11(a) and a bigger overshoot

in Fig. 12(a). Therefore, the optimization results of the 52nd iteration have better d - q axis current decoupling performance of IM than those of the 25th iteration.

As mentioned in Section II, ADRC can smooth the sudden change of the input signal in order to decrease the overshoot of the output response during the transient state. It makes ADRC get a good balance between the fast transient response and the small overshoot. However, the optimization results of 25th iteration still has bigger overshoot because the optimized parameters do not match with the controller plant well. With the optimization of APSO-ADRC, smaller overshoot and more rapidly dynamic process can be obtained.

2) *Robustness to Step Load*: The comparative experimental results between the 25th iteration and the 52nd iteration are selected to be compared in Figs. 13 and 14. The step load steps up from no load to rated load, and then steps down from rated load to no load. The speed reference is 1500 r/min in Fig. 13, and the speed reference is 30 r/min in Fig. 14. It can be seen that the 52nd iteration optimization results have shorter regulation time and smaller speed fluctuation amplitude than the 25th iteration optimization results when the load torque is varied suddenly.

As mentioned in Section II, ADRC is robust to the system model uncertainty. The conclusion can be drawn that the control performance index of the system has indeed been improved via the ESO and the real-time disturbance estimation and compensation. With the optimization of APSO-ADRC, better robustness and more rapidly dynamic process is obtained when the step load occurs.

3) *Robustness to Motor Parameter Variation*: Figs. 15 and 16 show the comparative experimental results between the 25th iteration and the 52nd iteration, and the speed reference is 30 r/min. The stator resistance R_s varies from 1 to 1.5 p.u. in Fig. 15, and the rotor resistance R_r varies from 1 to 1.5 p.u. in Fig. 16. It can be seen that the experimental results of 52nd iteration have smaller speed fluctuation than that of 25th iteration when R_s and R_r are changed, respectively.

Fig. 17 shows the comparative experimental results between the 25th iteration and the 52nd iteration, and the mutual inductance L_m varies from 1 to 1.5 p.u. in Fig. 17. It can be seen that the experimental results of 52nd iteration has smaller overshoot and smaller fluctuation than that of 25th iteration when L_m is changed, respectively.

The histogram of the comparative results between the 25th iteration and the 52nd iteration is given to compare the experimental data in Fig. 18. In Fig. 18, σ is the speed overshoot, t_r is the rise time, t_s is the settling time, and e_{ss} is the stable error. Δn is the maximum speed decrease when step load in this paper. The values of control performance index are divided by different denominators, in order to show the difference between the 25th iteration optimization results and the 52nd iteration optimization results more clearly. In Fig. 18, each control performance index of 52nd iteration is better than those of 25th iteration. Especially, no matter the speed reference is 1500 or 30 r/min, σ , t_r , and Δn are decreased more obviously. In summary, the dynamic performance of APSO-ADRC is improved greatly with the optimization going on.

TABLE I
FINAL BEST VALUE OF THE OPTIMIZED PARAMETER BETWEEN PSO AND APSO

	β_{1s}	β_{1q}	β_{1d}	β_{2s}	β_{2q}	β_{2d}
PSO-ADRC	112.2	55.2	95	315.2	459.5	262.2
APSO-ADRC	116.3	50.1	97.8	308.1	450.2	269.5

As the comparative experimental results are shown in Figs. 9–17, smaller overshoot, faster dynamic process, and better robustness can be obtained in the experimental results. With the mechanism of optimization, the control performance index of latter iteration is better than the former iteration. The conclusion can be drawn that the convergence performance of APSO-ADRC is verified, and the desired control performance index can be obtained by the optimization of APSO.

C. Convergence Performance Comparison Between PSO and APSO

In order to verify the advantages of APSO and PSO is selected to be compared with APSO. It is difficult to distinguish the differences between PSO and APSO based on the experimental results merely, because PSO and APSO are both effective to optimize the parameters of ADRC. As the important and specific characteristics of intelligent algorithm, the convergence speed, convergence performance, and efficiency are considered to compare the differences between PSO and APSO further. During the optimization, the experimental data shown in Fig. 18 are recorded and saved in the host computer, and these important data show the real complex process of optimization.

1) *Comparison of Parameter Convergence*: As to the convergence of parameter, the convergence process of the optimized parameter is shown in Fig. 19, and the final best value of each parameter is listed in Table I.

In details, the specific convergence process of every optimized parameter is shown in Fig. 20. As shown in Fig. 20, each point represents the global best position (value) among all the particles in every iteration time, and it corresponds to a fitness value calculated by the objective function. During the optimization, the parameter value gradually converges to one point, and the point represents the final best value of the parameter. In Fig. 21, APSO needs less iteration times to converge to reach the best value than PSO.

Based on these experimental data above, it can be concluded that although PSO and APSO are both effective to optimize the parameters of ADRC, however, as to the process of optimization, APSO has the faster convergence speed to reach the final value and makes the fitness value Y reach the minimum.

2) *Comparison of Convergence Performance*: During the optimization in the IM control system, it is a complex optimization problem, and the convergence performance of APSO is important to ensure whether APSO can be implemented in the IM control system and combined with ADRC successfully. As to the convergence performance, the variation of inertia weight w , evolution speed h , aggregation s , and fitness Y indicate the convergence performance, and they are selected to

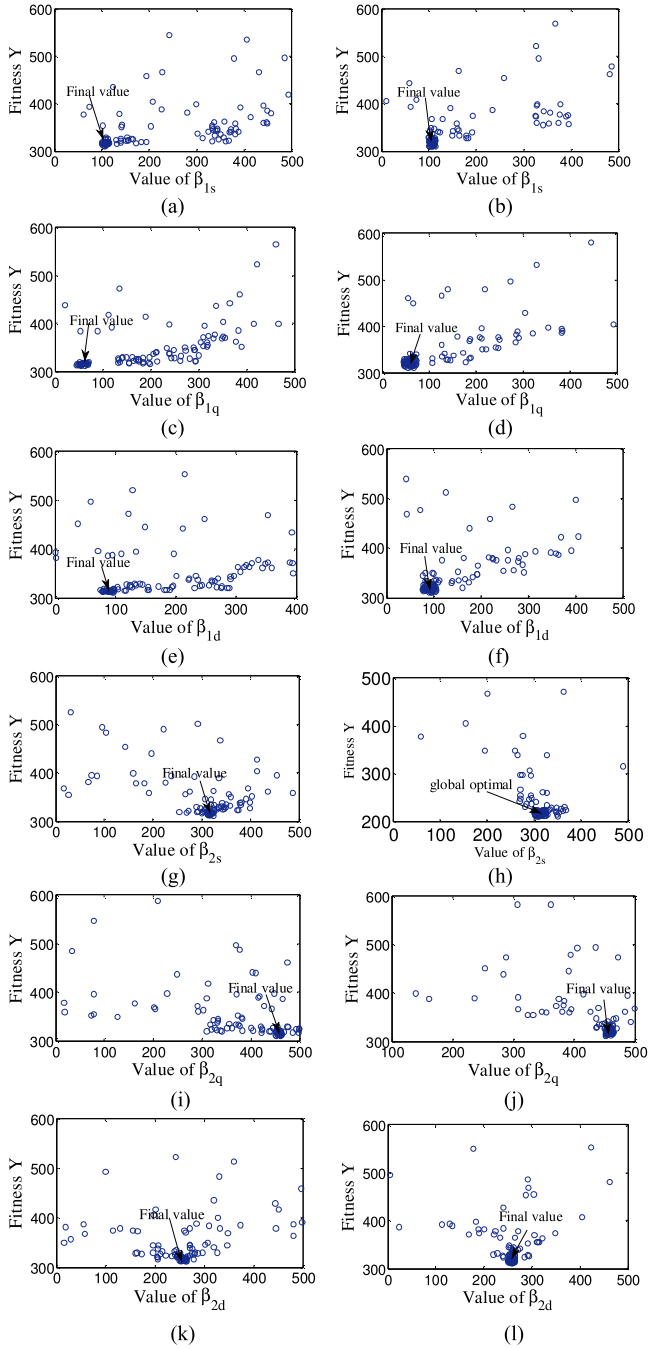


Fig. 20. Optimization process of each parameter optimized by PSO and APSO. (a) β_{1s} optimized by PSO. (b) β_{1s} optimized by APSO. (c) β_{1q} optimized by PSO. (d) β_{1q} optimized by APSO. (e) β_{1d} optimized by PSO. (f) β_{1d} optimized by APSO. (g) β_{2s} optimized by PSO. (h) β_{2s} optimized by APSO. (i) β_{2q} optimized by PSO. (j) β_{2q} optimized by APSO. (k) β_{2d} optimized by PSO. (l) β_{2d} optimized by APSO.

compare the convergence performance between PSO and APSO in Fig. 21.

In Fig. 21(a), the convergence process of inertia weight is shown. The inertia weight of PSO is decreased linearly with the iteration times. The inertia weight of APSO is changed based on the evolution speed and the aggregation degree shown in Fig. 21(b) and (c), respectively.

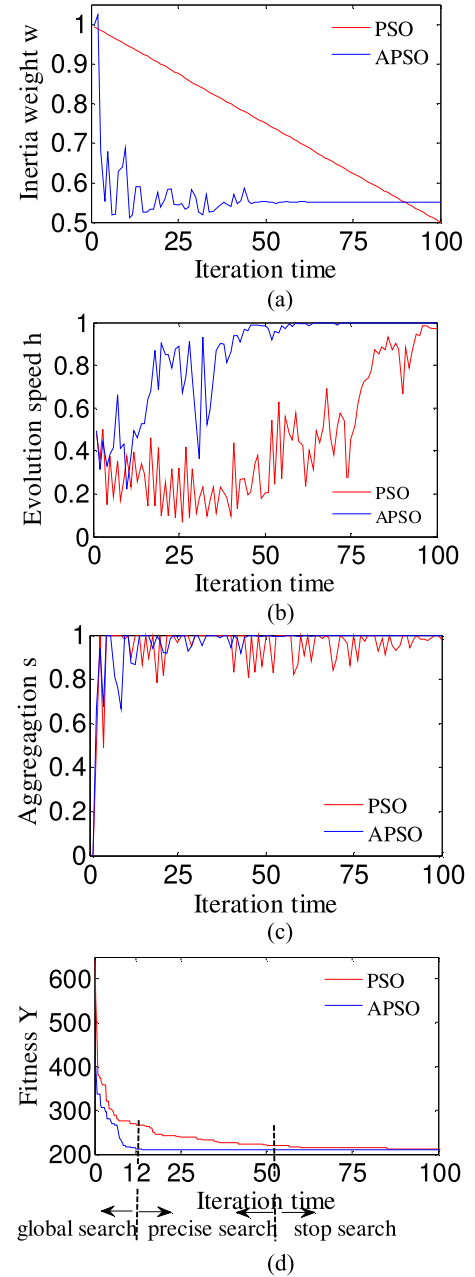


Fig. 21. Comparison of convergence performance between PSO and APSO. (a) Convergence of inertia weight w . (b) Convergence of evolution speed h . (c) Convergence of aggregation s . (d) Convergence of fitness Y .

a) At the Initial of Optimization: At the initial of optimization, the aggregation degree of both PSO and APSO are almost coincidence and close to one until the fourth iteration times in Fig. 21(c), and the evolution speed of both PSO and APSO are faster until the fifth iteration times in Fig. 21(b).

Based on the important information revealing the practical optimization process above, it can be concluded that a better position (value) is discovered and the particles have the tendency to converge; therefore, the inertia weight should be reduced to improve the ability of precise (local) search so that the best position (value) is discovered as soon as possible. However,

TABLE II
OPTIMIZATION PERFORMANCE COMPARISON BETWEEN
PSO-ADRC AND APSO-ADRC

	Y	$iter$	T_{loc}	T_{glb}
PSO-ADRC	333	97	16	84
APSO-ADRC	310	52	1	99

although the inertia weight of PSO is linearly decreasing, it is still bigger and unreasonable at this time so that it goes against the precise search and slows down the convergence speed. The inertia weight of APSO is nonlinearly reduced based on evolution speed and aggregation degree, it is smaller and more reasonable at this time so that APSO has stronger ability of precise search and faster convergence speed.

b) At the Middle of Optimization: At the middle of optimization, the aggregation degree of both PSO and APSO are very close to one after 25th iteration times in Fig. 21(c), and the evolution speed of both PSO and APSO gets bigger and bigger after the 29th iteration times [see Fig. 21(b)].

Based on the important information revealing the practical optimization process above, it can be inferred that the difference between different iteration times gets smaller and smaller so that the minimum value of inertia weight should be adopted to make the ability of precise search strongest. However, although the inertia weight of PSO is linearly decreasing, it is much bigger than that of APSO and results in more iteration times to converge. The inertia weight of APSO is reduced to the minimum value of inertia weight based on evolution speed and aggregation degree as soon as possible so that APSO has strongest ability of precise search and fastest convergence speed.

c) At the End of Optimization: To some extent, a faster convergence speed may result in being involved into local optimum. Therefore, it is essential to compare the final fitness. At the end of optimization, PSO needs 97 iteration times until the fitness Y changes no longer in Fig. 21(d), and APSO needs 52 iteration times until the fitness Y changes no longer. In the other words, the optimal solution is achieved, and the set of parameter is the optimal solution to the IM control system. The final fitness of PSO and APSO are 333 and 310, respectively. The fitness of APSO is smaller than that of PSO. Moreover, other differences between PSO and APSO are listed to compare their convergence performance in Table II. $iter$ is the iteration time to reach the final fitness. T_{loc} is the times that the final solution is the local optimum. T_{glb} is the times that the final solution is the global optimal. (In details, if the final fitness is smaller than 340, the final solution is regarded as the global optimal; otherwise, the final solution is regarded as the local optimum.) The experiment is carried out 100 times, T_{loc} and T_{glb} are counted, respectively. Based on the data in Table II, not only APSO has faster convergence speed than those of PSO, but also the final fitness of APSO is better and more effective than PSO.

According to the experimental data above, the adaptive inertia weight is decreased with the decrease of the evolution speed, and it is increased with the increase of the aggregation degree.

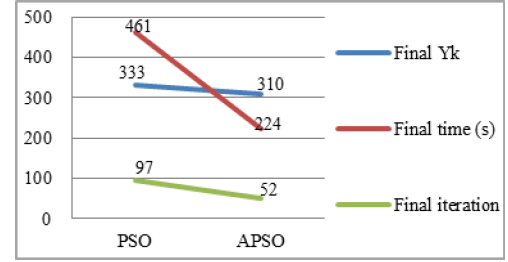


Fig. 22. Comparison between PSO and APSO.

It can be concluded that it is reasonable to modify the adaptive inertia weight based on the practical evolution speed and the aggregation degree so that APSO has faster convergence speed and better convergence performance than PSO. The proposed method is feasible and effective in the experiment.

3) Comparison of Efficiency: As to the same optimization problem, the length of time consumed shows the efficiency of algorithms, and the minimum fitness shows the optimization level of algorithms. As to the optimization level of algorithms, APSO has the minimum final Y_k than PSO in Fig. 22. The label meaning of Y -axis in Fig. 17 represents the value of final Y_k , the final time, and the final iteration, respectively. The final Y_k does not have unit, and it is a per-unit value. The unit of the final time is second. The unit of the final iteration is how many times (how often). As to the length of time consumed, the final time of PSO and APSO is 461 and 224 s to reach the final Y_k , respectively. It can be concluded that not only APSO has the minimum final Y_k , but also ACO has the faster convergence speed. Therefore, according to the analysis about the length of time consumed and the minimum value of object function Y , APSO is more efficient and effective than PSO.

D. Comparisons Between ACO and APSO

During the optimization, the experimental data shown in Fig. 23 are recorded and saved in the host computer, and these important data show the real complex process of optimization. The specific convergence process of every optimized parameter is shown in Fig. 23. As shown in Fig. 23, it can be concluded that ACO and APSO are both effective to optimize the parameters of ADRC and have fast speed to converge. However, it is difficult to distinguish them based on the whole optimization process so that the comparisons between ACO and APSO are mainly focused on the efficiency of the algorithm.

Based on the important data recorded in the host computer in Fig. 23, these important data show the real complex process of optimization from the macroscopic perspective, more importantly, they are closely related to the efficiency of ACO and APSO from the microscopic perspective. In Fig. 24, it is analyzed about the efficiency of ACO and APSO. The vertical axis denotes time in Fig. 24, and its unit is seconds. Y denotes the fitness, and the fitness is decreased as time goes on.

1) Efficiency of Global Optimization Process: In Fig. 24(a), the value of fitness Y is from 325 to 400, and it can be regarded as the global optimization process of ACO and APSO. At the initial

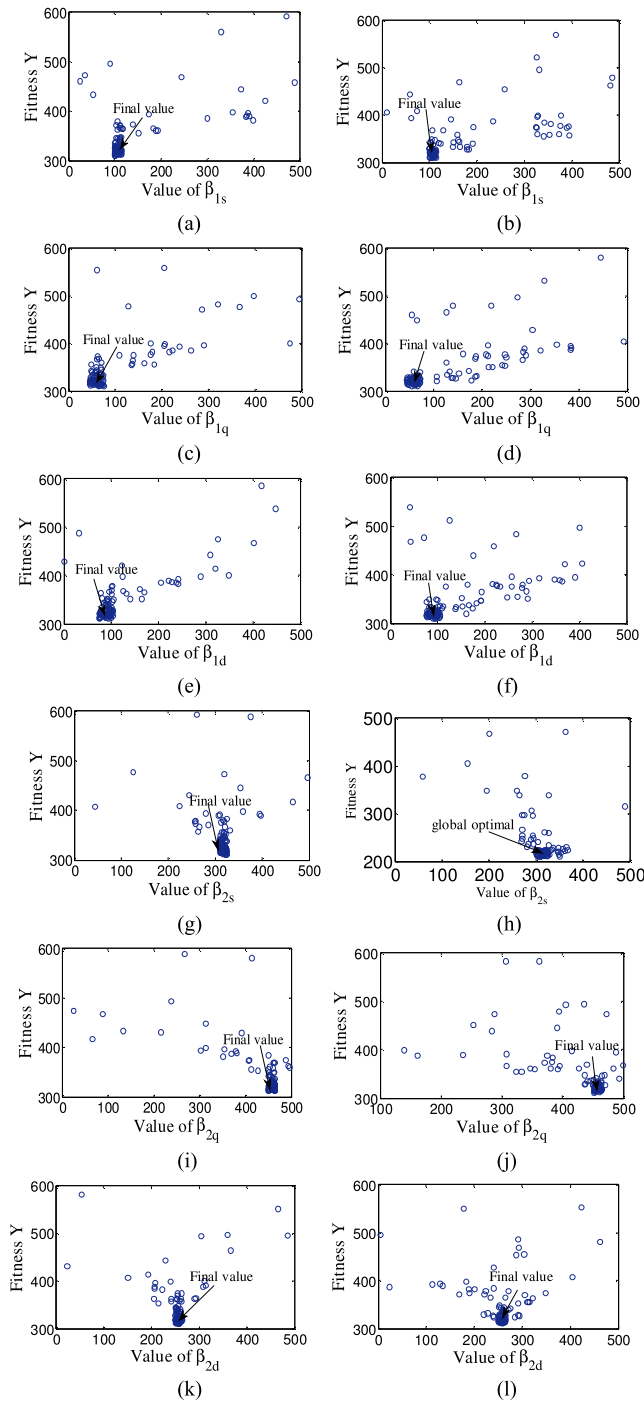


Fig. 23. Optimization process of each parameter optimized by ACO and APSO. (a) β_{1s} optimized by ACO. (b) β_{1s} optimized by APSO. (c) β_{1q} optimized by ACO. (d) β_{1q} optimized by APSO. (e) β_{1d} optimized by ACO. (f) β_{1d} optimized by APSO. (g) β_{2s} optimized by ACO. (h) β_{2s} optimized by APSO. (i) β_{2q} optimized by ACO. (j) β_{2q} optimized by APSO. (k) β_{2d} optimized by ACO. (l) β_{2d} optimized by APSO.

of the global optimization process, ACO takes 19 s to reach the fitness value 400, while APSO needs 27 s so that ACO has faster convergence speed than APSO. With the optimization going by, at the middle of the global optimization process, ACO takes 33 and 45 s to reach the fitness value 375 and 350, respectively,

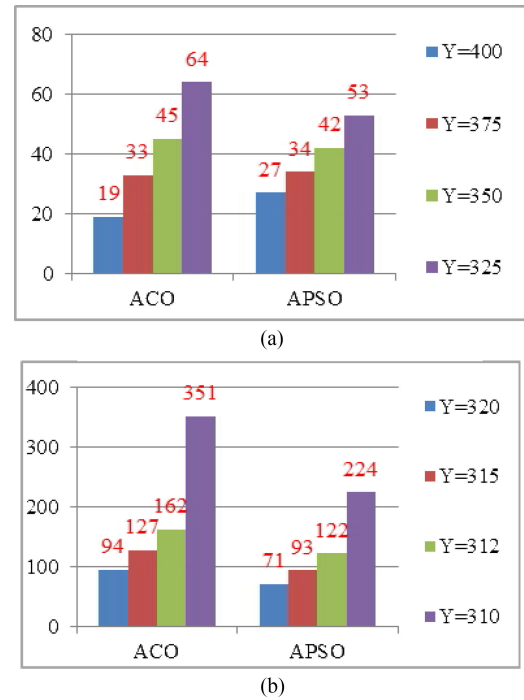


Fig. 24. Efficiency comparison of different sections. (a) Global optimization of ACO and APSO. (b) Precise optimization of ACO and APSO.

while APSO needs 34 and 42 s so that they almost have the same convergence speed. Meanwhile, benefit from the adaptive mechanism, APSO speeds up its convergence speed from this process. At the end of the global optimization process, ACO takes 64 s to reach the fitness value 325, while APSO needs 53 s so that APSO has faster convergence speed than ACO.

2) Efficiency of Precise Optimization Process: In Fig. 24(b), the value of fitness Y is from 320 to 310, and it can be regarded as the precise optimization process of ACO and APSO. At the initial of the global optimization process, ACO takes 94 s to reach the fitness value 320, while APSO needs 71 s. With the optimization going by, at the middle of the precise optimization process, ACO takes 127 and 162 s to reach the fitness value 315 and 312, respectively, while APSO needs 93 and 122 s. At the end of the precise optimization process, the time ACO consumed increases significant (351 s) to reach the fitness value 310, while APSO needs 224 s. Therefore, benefit from the adaptive mechanism, APSO has faster convergence speed than ACO at the whole of the precise optimization process.

It can be concluded that ACO has faster convergence speed than APSO at the initial of the global optimization process, ACO has almost the same convergence speed with APSO at the middle of the global optimization process, and ACO has slower convergence speed than APSO from the end of the global optimization process to the whole precise optimization process. The comparisons of the convergence speed between ACO and APSO directly express their comparisons of optimization efficiency, especially their comparisons of optimization time, which is shown in Fig. 25. As the pheromone evaporation of ACO is a constant, and it is same with the inertia weight of PSO and is the

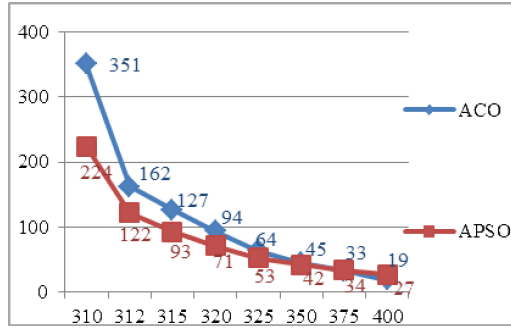


Fig. 25. Optimization time comparison between ACO and APSO.

tradeoff between the global and local search ability. To some extent, ACO avoids being trapped into the premature convergence owing to all the history information is used to update the pheromone. It is the reason why ACO has faster convergence speed than APSO at the initial. However, when the optimization results of ADRC gets close to the optimum, the convergence speed of ACO always gets slower because the constant pheromone evaporation of ACO goes against the local precise search, and the history information also goes against the convergence as soon as possible. Therefore, the efficiency of ACO is degraded. The adaptive inertia weight of APSO is dynamically modified based on the practical operation process of PSO (evolution speed and the aggregation degree) to obtain an ideal inertia weight. Meanwhile, with the dynamic modification of the inertia weight, APSO has faster convergence speed and better optimization efficiency than ACO.

V. CONCLUSIONS

- 1) A novel APSO-ADRC method is proposed to be applied in the IM control system. The novel control method employs APSO as the automatic tune mechanism for ADRC controller. According to the feedback information from IM, the ADRC controller parameters can be optimized by the optimization mechanism and self-learning ability of APSO, so the reliance of ADRC controller on parameters can be reduced. Experimental results indicate that the control performance index of the novel control method has smaller overshoot, shorter regulation time, and smaller speed fluctuation amplitude than the conventional one in this paper. Therefore, the robustness of the proposed optimal design method for ADRC is better than the conventional ADRC, and the method is feasible and effective.
- 2) The evolution speed and the aggregation degree reflect the practical optimization process, and they are used to dynamically modify the inertia weight of APSO so that the adaptive inertia weight is reasonable updated to make the control performance better. It uses performance criterion tailoring to the application so that the optimal solution is statistically valuable. It can avoid trapping into the local optimum and premature convergence effectively. Therefore, APSO is a more efficient algorithm than PSO and ACO in finding optimal solutions for difficult combinatorial problems.

APPENDIX

Parameter	Value	Unit
Rated Power	2.2	kW
Rated Voltage	380	V
Rated Current	5.0	A
Rated Frequency	50	Hz
Rated Speed	1410	r/min
Pole Pairs	2	-
Stator Resistance	6.03	Ω
Rotor Resistance	6.77	Ω
Stator Inductance	28.787	mH
Rotor Inductance	28.787	mH
Mutual Inductance	28.312	mH
Moment Inertia	0.02	kg·m ²

REFERENCES

- [1] C. Lascu, S. Jafarzadeh, M. S. Fadali, and F. Blaabjerg, "Direct torque control with feedback linearization for induction motor drives," *IEEE Trans. Power Electron.*, vol. 32, no. 3, pp. 2072–2080, Mar. 2017.
- [2] G. Zhang, G. Wang, D. Xu, and N. Zhao, "ADALINE-network-based PLL for position sensor-less interior permanent magnet synchronous motor drives," *IEEE Trans. Power Electron.*, vol. 31, no. 2, pp. 1450–1460, Feb. 2016.
- [3] Y. Fan, L. Zhang, and M. Cheng, "Sensor-less SVPWM-FADTC of a new flux-modulated permanent-magnet wheel motor based on a wide-speed sliding mode observer," *IEEE Trans. Ind. Electron.*, vol. 62, no. 5, pp. 3143–3151, May 2015.
- [4] X. Zhang *et al.*, "Nonlinear speed control for PMSM system using sliding-model control and disturbance compensation techniques," *IEEE Trans. Power Electron.*, vol. 3, no. 28, pp. 1358–1365, Mar. 2013.
- [5] W. Sun, Y. Zhang, Y. Huang, H. Gao, and O. Kaynak, "Transient-performance-guaranteed robust adaptive control and its application to precision motion control systems," *IEEE Trans. Ind. Electron.*, vol. 63, no. 10, pp. 6510–6518, Oct. 2016.
- [6] G. Luo, R. Zhang, Z. Chen, W. Tu, S. Zhang, and R. Kennel, "A novel nonlinear modeling method for permanent-magnet synchronous motors," *IEEE Trans. Ind. Electron.*, vol. 63, no. 10, pp. 6490–6498, Oct. 2016.
- [7] J.-T. Huang, T. Van Hung, and M.-L. Tseng, "Smooth switching robust adaptive control for omnidirectional mobile robots," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 5, pp. 1986–1993, Sep. 2015.
- [8] Q. Zhu, Z. Yin, Y. Zhang, J. Niu, Y. Li, and Y. Zhong, "Research on two-degree-of-freedom internal model control strategy for induction motor based on immune algorithm," *IEEE Trans. Ind. Electron.*, vol. 63, no. 3, pp. 1981–1992, Mar. 2016.
- [9] X. Sun, Z. Shi, L. Chen, and Z. Yang, "Internal model control for a bearingless permanent magnet synchronous motor based on inverse system method," *IEEE Trans. Energy Convers.*, vol. 31, no. 4, pp. 1539–1548, Dec. 2016.
- [10] L. C. Sun, H. Jiang, Z. Yang, J. Chen, and W. Zhang, "High-performance control for a bearingless permanent-magnet synchronous motor using neural network inverse scheme plus internal model controllers," *IEEE Trans. Ind. Electron.*, vol. 63, no. 6, pp. 3479–3488, Jun. 2016.
- [11] L. Zhao, Y. Yang, and Y. Xia, "Active disturbance rejection position control for a magnetic rodless pneumatic cylinder," *IEEE Trans. Ind. Electron.*, vol. 62, no. 9, pp. 5838–5846, Sep. 2015.
- [12] W. Xue, W. Bai, S. Yang, K. Song, Y. Huang, and H. Xie, "ADRC with adaptive extended state observer and its application to air–fuel ratio control in gasoline engines," *IEEE Trans. Ind. Electron.*, vol. 62, no. 9, pp. 5847–5857, Sep. 2015.
- [13] X. Chang, Y. Li, and W. Zhang, "Active disturbance rejection control for a flywheel energy storage system," *IEEE Trans. Ind. Electron.*, vol. 62, no. 2, pp. 991–1001, Feb. 2015.

- [14] B. Du, S. Wu, S. Han, and S. Cui, "Application of linear active disturbance rejection controller for sensor-less control of internal permanent-magnet synchronous motor," *IEEE Trans. Ind. Electron.*, vol. 63, no. 5, pp. 3019–3027, May 2016.
- [15] G. Wang, Y. Wang, J. Xu, N. Zhao, and D. Xu, "Weight-transducerless rollback mitigation adopting enhanced MPC with extended state observer for direct-drive elevators," *IEEE Trans. Power Electron.*, vol. 31, no. 6, pp. 4440–4451, Jun. 2016.
- [16] G. Feng, Y.-F. Liu, and L. Huang, "A new robust algorithm to improve the dynamic performance on the speed control of induction motor drive," *IEEE Trans. Power Electron.*, vol. 19, no. 6, pp. 1614–1627, Nov. 2004.
- [17] J. Li, H.-P. Ren, and Y.-R. Zhong, "Robust speed control of induction motor drives using first-order auto-disturbance rejection controllers," *IEEE Trans. Ind. Appl.*, vol. 51, no. 1, pp. 712–720, Jan./Feb. 2015.
- [18] K. Erenturk, "Fractional-order and active disturbance rejection control of nonlinear two-mass drive system," *IEEE Trans. Ind. Electron.*, vol. 60, no. 9, pp. 3806–3813, Sep. 2013.
- [19] J. Prasanth Ram and N. Rajasekar, "A novel flower pollination based global maximum power point method for solar maximum power point tracking," *IEEE Trans. Power Electron.*, vol. 32, no. 11, pp. 8486–8499, Nov. 2017.
- [20] S. S. Sebtahmadi, H. B. Azad, S. H. A. Kaboli, M. D. Islam, and S. Mekhilef, "A PSO-DQ current control scheme for performance enhancement of Z-source matrix converter to drive IM fed by abnormal voltage," *IEEE Trans. Power Electron.*, vol. 33, no. 2, pp. 1666–1681, Feb. 2018.
- [21] C. Zhang, Z. Chen, Q. Mei, and J. Duan, "Application of particle swarm optimization combined with response surface methodology to transverse flux permanent magnet motor optimization," *IEEE Trans. Magn.*, vol. 53, no. 12, Dec. 2017, Art. no. 8113107.
- [22] M. Hu, T. Wu, and J. D. Weir, "An adaptive particle swarm optimization with multiple adaptive methods," *IEEE Trans. Evol. Comput.*, vol. 17, no. 5, pp. 705–720, Oct. 2013.
- [23] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1362–1381, Dec. 2009.
- [24] J. Liu, Y. Mei, and X. Li, "An analysis of the inertia weight parameter for binary particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 666–681, Oct. 2016.
- [25] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 240–255, Jun. 2004.
- [26] Z. Yin, C. Du, and J. Liu, "Research on autodisturbance-rejection control of induction motors based on ant colony optimization algorithm," *IEEE Trans. Ind. Electron.*, vol. 65, no. 4, pp. 3077–3094, Apr. 2018.



Chao Du (S'17) was born in Shannxi, China, in 1991. He received the B.S. and M.S. degrees in electrical engineering from Xi'an University of Technology, Xi'an, China, in 2013 and 2016, respectively, where he has been working toward the Ph.D. degree since 2016.

His research interests include high-performance ac drive systems and its optimization of efficiency and parameters.



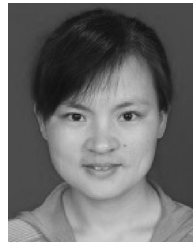
Zhonggang Yin (M'13) was born in Shandong, China, in 1982. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from Xi'an University of Technology, Shaanxi, China, in 2003, 2006, and 2009, respectively.

In 2009, he joined the Electrical Engineering Department, Xi'an University of Technology, where he is currently a Professor. His research interests include high-performance control of ac motor and digital control of power converters.



Yanping Zhang was born in Shaanxi, China, in 1989. He received the B.S. degree in electrical engineering from Xi'an Polytechnic University, Xi'an, China, in 2013, and the M.S. degree in power electronics and electrical drives in electrical engineering from Xi'an University of Technology, Xi'an, in 2017, where he has been working toward the Ph.D. degree since 2017.

His research focuses on high-performance sensor-less control of PMSM.



Jing Liu was born in Anhui, China, in 1982. She received the B.S., M.S., and Ph.D. degrees in electronic engineering from Xi'an University of Technology, Xi'an, Shaanxi, China, in 2003, 2006, and 2009, respectively.

In 2009, she joined the Electronic Engineering Department, Xi'an University of Technology, where she is currently an Associate Professor. Her research interests include power semiconductor devices and their application to power electronic devices.



Xiangdong Sun was born in Shenyang, China, in 1971. He received the Ph.D. degree in electrical engineering from Xi'an University of Technology, Xi'an, China, in 2003.

He was a Postdoctoral Researcher with Tokyo Polytechnic University supported by the government scholarship of Japan during 2006–2008. He is currently a Professor with the Department of Electrical Engineering, Xi'an University of Technology. His research interests include motor control, power electronics, and renewable energy system.



Yanru Zhong was born in Xi'an, China, in 1950. He received the B.S. degree in electrical engineering from Xi'an Jiaotong University, Xi'an, in 1975, and the M.S. degree in electrical engineering from Xi'an University of Technology, Xi'an, in 1983.

In 1983, he joined Xi'an University of Technology. In 1987, he was a Visiting Scholar with the Electrical Engineering Department, Sophia University, Japan. Since 1993, he has been a Professor with Xi'an University of Technology. His research focuses on power electronics, especially inverter and ac drive systems.