Motion control system of slewing bearing based on Improved Adaptive Mutation Particle Swarm Optimization

Hua Wang[*], Yunke Han[**], Bitao Pang[**], Xuehai Gao[***]

[*]*School of Mechanical and Power Engineering, Nanjing Tech University, Nanjing, 211816, China*
（email: wanghua@njtech.edu.cn）
[**]*Luoyang LYC bearing Co., Ltd., Luoyang，471003，China*
[***]*Shanghai OujiKete Slewing bearing Co., Ltd. Shanghai 201906, China*

Abstract: The response performance and speed follow-up performance of the slewing bearing control system affect the whole equipment performance directly. The core of the system is PID controller, however, whether the optimal parameters could be found or not seriously affects the performance of the controller. In this paper, the mutation operation of genetic algorithm (GA) was introduced and the particle mutation part was taken to particle swarm optimization (PSO) to overcome the deficiency of falling into local optimal solution and eliminate the influence on system performance resulted from the improper selection of initial controller parameters, which was called adaptive mutation particle swarm optimization (AMPSO). And based on the AMPSO, a simplified particle mutation rule was presented and the mutation object was expanded to the entire population of the particles compared to AMPSO, which is called improved adaptive mutation particle swarm optimization (IAMPSO). The simulation show that the K_p, K_i parameters obtained by this algorithm is superior to that of the original PSO algorithm; and the experiment verified the effectiveness of this proposed control strategy.

Keywords: Slewing bearing; PI controller; PSO; Particle mutation; PID

1 INTRODUCTION

Slewing bearing is a type of mechanical parts widely used in engineering machinery, wind turbines, offshore platforms and other large mechanical structures. Its main function is to connect two structural parts and enable relative rotation of them, and the working loads are mostly pressured to it. It is similar to plain bearing, but it also has some different characteristics: gear transmission, heavy loads, low speed, various structure forms and poor working conditions (Amasorrain et al., 2003). Its operating status directly affects the whole equipment performance and determines the response characteristic of the device control unit, such as the wind steering system of wind turbine. Therefore, research on improving the slewing bearing motion control level has great significance for efficient operation of wind turbine and other equipments. The core of slewing bearing motion control system is PID controller, it is simple, mature and practical, but whether the optimal parameters seriously affect the performance of the controller (Xing et al., 2007). Slewing bearing motion control system requires fast response speed and anti-jamming capability, while PID controller tuned by traditional methods cannot meet these requirements well (Xing et al., 2007; Ji et al., 2010). So it is necessary to find an effective parameter tuning method of PID controller to improve the system performance.

In recent years, artificial intelligence (AI) algorithm provides a new way for PID controller parameters tuning. Particle swarm optimization (PSO) has been listed as a topic of discussion by IEEE international conferences on evolutionary computation (CEC) due to its simple algorithm structure, high efficiency and other characteristics (Tang, 2010). Many researchers applied PSO into PID controller parameters tuning (Gaing, 2004; Nasri et al., 2007). However, original PSO algorithm is easily falling into local optimum solution, which affects its optimization ability (Eberchart and Kennedy, 1995; Hou et al., 2014). (Kennedy, 2000) himself improved PSO algorithm firstly with cluster centers instead of particle best. This change improved the algorithm performance, but the cluster calculation made the algorithm more complex. (Shi and Eberchart, 2001) proposed a fuzzy-rules-based method of adjusting inertia weight ω dynamically. It is developing appropriate membership functions and fuzzy rules to determine the inertia weight increment by evaluating the current best performance and the current inertia weight. The disadvantage of this method is that the development of fuzzy rules requires expert knowledge which is poor and difficult to obtain before the complex system being optimized, so this method is difficult to achieve. Another way is to construct a new algorithm. Integrating the algorithm which has strong local search ability into PSO algorithm brings a new ground of new algorithms construction. Literatures (Lu and Hou, 2004; Yang et al., 2008) improved PSO algorithm based on this idea and achieved good results. However, literature (Lu and Hou, 2004) introduced too many parameters, which result in the extremely complex algorithm; literature (Yang et al., 2008) only performed mutation operation to the m particles which own best fitness value, which is lack of global optimization. This paper summarizes the experience of the above-described improved algorithms, presents an improved adaptive mutation particle swarm optimization (IAMPSO) algorithm based on the idea of algorithm fusion, thus the PI parameters in internal velocity loop were optimized to improve the

performance of slewing bearing motion control system.

2. EQUIVALENT MODEL OF MECHANICAL TRANSMISSION

As shown in Fig. 1, slewing bearing mechanical structure mainly includes slewing bearing, drive gear, servo reducer and motor. The outer ring of slewing bearing meshes with the drive gear and the transmission ratio of them is i_1; the drive gear connects to the output shaft of reducer and the ratio is i_2; the output shaft of motor connects to the input shaft of reducer; the slewing bearing is driven by the motor through the transmission parts mentioned above. During operation, the speed of slewing bearing is ω, the speed of the drive gear is ω_H same as the output shaft of reducer and the speed of the input shaft of reducer is ω_r same as the output shaft of motor.

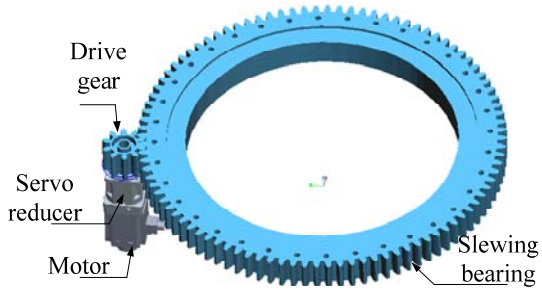


Fig. 1. Simplified model of slewing bearing mechanical structure.

In this study, the slewing bearing mechanical structure is divided into two sub-parts: gear train part and servo drive part. Without considering the stiffness, the moment of inertia in the mechanical transmission parts can be transferred to the motor shaft according to the superposition principle (Shi, 2007). The kinetic energy of gear train part is the sum of the kinetic energy of drive gear and outer ring of slewing bearing, and the servo drive part's is the sum of reducer's and motor's. The kinetic energy of planet carrier, ring gear, sun gear and three planet gears make up the kinetic energy of reducer. The moment of inertia of reducer and motor can be learned by buyer's guide, so the moment of inertia of each part can be obtained just by calculating the kinetic energy of gear train part. The kinetic energy of gear train part can be expressed as:

$$E_k = \frac{1}{2} J_d \omega_H^2 + \frac{1}{2} J_l \omega^2 \tag{1}$$

where J_d is the moment of inertia of drive gear; J_l is the moment of inertia of outer ring of slewing bearing. The equivalent kinetic energy of this part can be depicted as:

$$E_{eq} = \frac{1}{2} J_{eq} \omega_r^2$$
$$\omega = i_1 \omega_H = i_1 i_2 \omega_r \tag{2}$$

Hence, the equivalent moment of inertia of the gear train part is:

$$J_{eq} = i_2^2 J_d + (i_1 i_2)^2 \tag{3}$$

Take servo drive part into consideration, the moment of inertia of the mechanical structure can be all converted to the motor shaft as:

$$J = J_m + J_r + J_{eq} \tag{4}$$

where J is the moment of inertia of the whole equivalent, J_m is the moment of inertia of motor rotor and J_r is the moment of inertia of reducer. i_1 is calculated according to the number of teeth of drive gear and slewing bearing; i_2 is checked by reducer manual; the drive gear and slewing bearing are modeled in Pro-E by actual size, then export J_d, J_l. Take the amounts above into formula (3) and J_{eq} can be obtained. J_m and J_r are checked by relevant manuals; Therefore the moment of inertia of each part can be obtained and converted to the motor shaft as formula (4) with the final result of J=51.18×10^{-4} kg·m^2.

3. PI CONTROLLER OPTIMIZED by IAMPSO

3.1 Improvement of original PSO

When looking for food in a large field, the most simple and effective way for the bird flock is to search the surrounding area of the bird nearest to the food. Inspired by the way the birds looking for food, Kennedy and Eberhart proposed the theory of PSO algorithm in 1995 for the purpose of solving the problem of function extreme value optimization (Eberhart and Kennedy, 1995).

A brief description of PSO algorithm theory: there is a group of random 'particles' in D-dimensional search space, each particle is likely to be a potential solution to the problem. Each particle has its own position and velocity, the position of the ith particle in D-dimensional space can be written as $X_i=(x_{i1},x_{i2},\ldots,x_{iD})^T$, and the velocity is $V_i=(V_{i1},V_{i2},\ldots,V_{iD})^T$ similarly. The velocity of the particle moving in D-dimensional space determined the moving direction and distance with respect to original position. According to the objective function of the problem to be solved, the fitness value corresponding to each particle can be calculated, and the value decides the possibility of the particle to be the optimal solution. The particles are characterized by position, velocity and fitness value of these three parameters. The individual extreme value of a particle is written as $P_i=(P_{i1},P_{i2},\ldots,P_{iD})^T$ and the global extreme value is $P_g=(P_{g1},P_{g2},\ldots,P_{gD})^T$ similarly. In each iteration, the velocity of each particle adjusts dynamically to update their position according to these two extreme values (P_i and P_g), that is, moving closer to the optimal solution continuously till finding it.

In each iteration, the velocity and position of the ith particle are expressed as:

$$V_{id}^{k+1} = \omega_i V_{id}^k + c_1 r_1 (P_{id}^k - X_{id}^k) + c_2 r_2 (P_{gd}^k - X_{id}^k) \tag{5}$$

$$X_{id}^{k+1} = X_{id}^k + V_{id}^{k+1} \tag{6}$$

where ω_i is inertia weight; c_1 and c_2 are acceleration factors; r_1 and r_2 are random numbers in [0,1]; i=1,2,…,n; d=1,2,…,D; k is the current iterations; X_{id} is particle position; V_{id} is particle velocity. In order to prevent the blind search, the velocity is generally limited in a range of [-V_{max},V_{max}].

PSO algorithm has some advantages such as simple operation, fast convergence and wide universality. However, there exists a serious shortcoming: easily falling into premature convergence. That is, in the later period of iteration, the particles getting higher and higher degree of similarity, and the algorithm may fall into partial optimum solution and hardly getting out, thus miss the global optimal solution (Zhao et al., 2013). By observing the running process of PSO, we found that many particles' position is beyond the set spatial extent in each cycle. Duo to the algorithm rules limit, these particles are placed at the edge of the search space after they out. More and more particles are stacked, causing particle with high similarity, what is not conducive to find the global optimal solution. If the "marginalization" of the particles were broken up and redistributed, the "particle accumulation" problem would be solved. The particle distribution changing process is shown in Fig. 2.

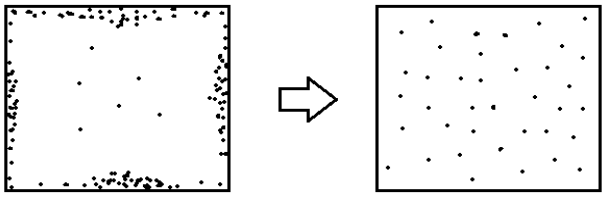


Fig. 2. Schematic Diagram of Particle distribution before and after the algorithm improvement.

To solve this problem, the mutation operation of GA was introduced to PSO to reset the particles in a certain probability, which is called adaptive mutation particle swarm optimization (AMPSO). However, the general AMPSO model has so many parameters resulting in the complex calculation. And because AMPSO just perform mutation operation to some particles instead of all particles, it is difficult to obtain the global optimized solution. Thus, based on the AMPSO, a simplified particle mutation rule was presented and the mutation object was expanded to the entire population of the particles compared to AMPSO. Not only does this operation reduce the similarity of the particles, but also it expands their search space which makes the particles jumping out of the current position and searching in a larger space, and increases the possibility of finding better value.

The specific implementation process is as follows: the position of the *i*th particle in *D*-dimensional space is X_{id}^{k} after k iterations. Set a mutation threshold p_{m} as mutation condition, if this particle meets the mutation condition, mutation occurs, if not, keeps the original position. The mutation rule is adding random perturbation to the particle position, described as:

$$X_{id}^{k}=\begin{cases} X_{id}^{k}*(1+0.5\eta) & r\geq p_m \\ X_{id}^{k} & r<p_m \end{cases} \tag{7}$$

where η is a random variable subjected to Gaussian distribution [0, 1]; r is random number in [0, 1]; and select a suitable value for p_{m} according to the actual situation.

After that, an improved adaptive mutation particle swarm optimization (IAMPSO) algorithm came into being. The particles of the new algorithm own the ability to self-reset and the "marginalization" particles can self-select a new position, which makes the particle similarity significantly decreased. Not only has the IAMPSO kept the advantages of PSO, it has also avoided the complex structure of AMPSO, expanded the search space of the particles, overcame the problem of the searching results easily falling into partial optimum solution and increased the probability of finding the global optimal solution.

3.2 Optimization performance test of IAMPSO

In order to verify the global optimization ability of IAMPSO, three typical functions are utilized to test the algorithm performance and comparison with PSO is given.

Test function 1: Rosenbrock. This function is a typical morbid secondary multimodal function, although the valley where the minimum value located is easily to be searched, there is a very narrow valley between local optimum and global minimum while general optimization algorithm easily fall into the trap of local optimum. Therefore, this function is commonly used in algorithm performance test, which is expressed as:

$$f(x_i)=\sum_{i=1}^{n}(100(x_i^{\,2}-x_i+1)^2+(x_i-1)^2) \qquad x_i\in[-100,100] \tag{8}$$

Test function 2: Griewangk. This function is continuous and has a plurality of minimum values which distributes with a certain regularity and is difficult to converge to global minimum, which is expressed as:

$$f(x_i)=\frac{1}{4000}\sum_{i=1}^{n}x_i^{\,2}+\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1 \qquad x_i\in[-600,600] \tag{9}$$

Test function 3: Rastrigin. This function is a typical nonlinear multimodal function whose peak shape is fluctuating, so it is difficult to find the global optimum, which is expressed as:

$$f(x_i)=100n+\sum_{i=1}^{n}\left[x_i^{\,2}-10\cos(2\pi x_i)\right] \qquad x_i\in[-10,10] \tag{10}$$

The optimization performance comparison experiment of PSO and IAMPSO was done with the three test functions mentioned above. Set the particle swarm m=30, the number of dimensions D=2, inertia weight ω_{i}=0.6, acceleration factors c_1=c_2=2, velocity range [-1, 1], maximum iterations 500, mutation threshold p_m=0.9, the global minimum as optimization goal. When the algorithm running greater than 500 iterations or the iterative error lesser than 10^{-5}, terminate the algorithm and output the results. The evaluation basis is the average value of 50 times experiments, and the results are shown in Table 1.

Table 1. Results Comparison of the Two Algorithms.

Test function	Algorithm	Global minimum	Error	Iterations
Rosenbrock	PSO	(1.0006,1.0013)	4.8982×10^{-7}	223
	IAMPSO	(1.0001,1.0002)	9.7693×10^{-9}	171
Griewangk	PSO	(102.25,94.8362)	7.396×10^{-3}	214
	IAMPSO	(100,100)	3.5826×10^{-6}	221
Rastrigin	PSO	(0,0)	1.6462×10^{-10}	178
	IAMPSO	(0,0)	1.4963×10^{-10}	138

As can be seen from the test results, to Rosenbrock and Rastrigin, PSO and IAMPSO can both get the global optimal solution, but the iteration times of IAMPSO is significantly less than PSO, it means that IAMPSO converges faster than PSO with the same solution accuracy; to Griewangk, PSO cannot get the global optimal solution and ultimately falls into local convergence, while IAMPSO can achieve the goal and prevent falling into local optimum under the premise of convergence rate.

Conclusions of the tests: the proposed IAMPSO has faster convergence rate, more accurate search results and better parameter optimization ability than PSO, which can be applied to PI controller parameters optimization to improve the slewing bearing motion control system performance.

3.3 PI controller optimization by IAMPSO

Generally PMSM is driven by servo amplifier. It adopts three-loop control mode, including position loop, velocity loop and current loop. Among them, current loop is the inner-loop of velocity loop; position loop is the outer-loop of the controller and in series with velocity loop. Current loop determines the dynamic performance of the control system, whose parameters are determined by system inherent characteristics which should not be changed artificially. Velocity loop controls the system by PI controller, mainly affects the response speed, positioning accuracy and anti-interference ability of the system. The performance of velocity loop directly determines the performance of position loop and affects the whole system performance significantly (Wang et al., 2012). So this study mainly aimed at optimizing the velocity loop PI parameters for the preferable control performance.

In computer control system, the form of PI controller is discrete case:

$$u(k) = K_p e(k) + K_i \sum_{j=0}^{k} e(j) \tag{11}$$

where $u(k)$ is the output value of the kth sampling time; $e(k)$ is the control error of the kth sampling time; K_p, K_i are the proportional and integral coefficients of the PI controller.

The performance of PI controller mainly depends on whether the optimization of K_p and K_i is reasonable. If it is, the control error will be reduced as much as possible until the requirement is satisfied.

To apply the algorithm proposed above to parameter optimization, it is necessary to establish the fitness function to evaluate the particle performance, that is, the performance evaluation indicator of PI controller. Generally we use overshoot, rise time and steady-state error to evaluate the controller performance. However, because of the interaction of the various parameters, it is almost impossible to make all indicators achieve the best, so we need to weigh the influence of the various parameters on the system performance. Meanwhile, there is strict requirement for positioning accuracy of slewing bearing in engineering applications, so we mainly focus on the changes of the systematic error over time. In this study, error property index *ITAE* is selected to evaluate the system response performance (Zheng et al., 2016), which is defined as:

$$F_{ITAE} = \int_{t=0}^{\infty} t\left|e(t)\right|dt \tag{12}$$

where $e(t)$ represents the deviation between the specified value and the actual value. This formula can be transformed as a discrete case:

$$F_k = F_{k-1} + t(k)\left|e(k)\right|\Delta t \tag{13}$$

where F_{k-1}, F_k represent the ITAE indexes at the (k-1)th and kth sampling point, $t(k)$ represents the time of kth sampling point, $e(k)$ represents the deviation between the specified value and the actual value, Δt represents the sampling interval.

Regard formula (13) as objective function, and then use IAMPSO to find the global optimal solution of the objective function, that is, the optimal K_p, K_i value. The process is shown in Fig. 3.

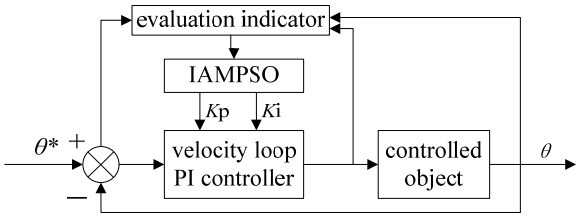


Fig. 3. IAMPSO-PI controller.

The specific steps of PI controller optimization by IAMPSO can be expressed as:

1) Particle swarm initialization: define the search space of K_p, K_i parameters and generate initial swarm X in the space;

2) Input each set of particles $X_i=(x_{i1},x_{i2})^T$ into the control system and run, calculate the fitness value F_j corresponding to each set according to formula (13);

3) If $F_j<F_{j-1}$, P_i updated to X_i; if $F_j=min[F]$, P_g updated to X_i;

4) Update the velocity and position of each particle according to formula (5) and (6);

5) Mutation: mutation operation on the particles according to formula (7), if mutation occurs, new particle position obtained;

6) If the maximum number of iterations reached or system performance requirements met, take P_g as the final result of K_p, K_i parameters, if not, return to Step 2 and continue the iteration.

4. SYSTEM SIMULATIONS AND EXPERIMENT

4.1 Simulation results

The system simulation model was established based on the mathematical model of slewing bearing motion control system. In the MATLAB environment, the simulation model was established by *simulink* based on the equivalent model of mechanical transmission, mathematical model of motor and PI controller model; the control algorithm in the form of *m file* is written based on the theory of PI controller optimization by IAMPSO. Through joint debugging of simulation model and control algorithm, the influence on slewing bearing motion control system by the velocity loop PI controller optimized by IAMPSO can be observed.

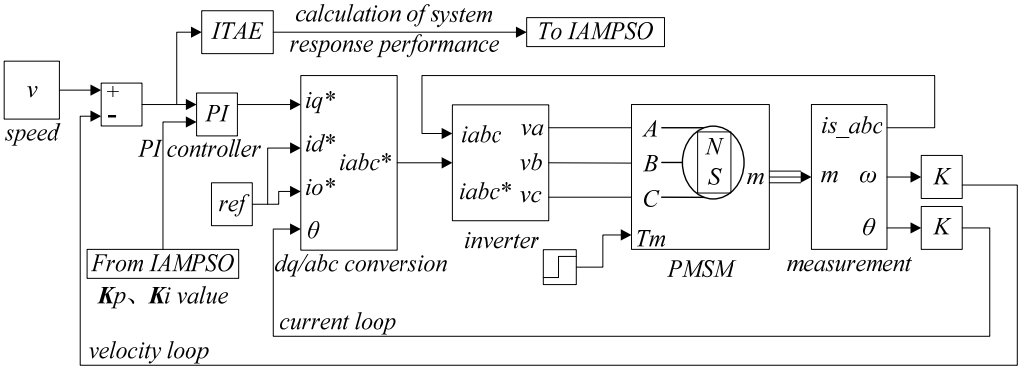


Fig. 4. Simulation model of the system.

Table 2. Control system parameters.

Parameter	Unit	Value	Parameter	Unit	Value
Rated current I	A	11	Stator resistance R_s	Ω	1.5
Rated torque T_e	$N{\cdot}m$	9.55	Rotor flux ψ_f	Wb	0.243
Rated speed n	r/min	2000	Pole pairs P_n		3
Inductance L_d、L_q	mH	5.7	Moment of inertia J	$kg{\cdot}m^2$	51.18×10^{-4}

As shown in Fig. 4, the simulation model mainly includes PI controller, system response performance calculation module, *dq*/*abc* conversion, inverter, PMSM and measurement module. Among them, *dq*/*abc* conversion, inverter and PMSM make up the current loop; the current loop and PI controller make up the velocity loop; the position loop is not added in the simulation model because it belongs to the outer-loop of the system and has no effect on the inner-loop; the system response performance calculation module established by formula (12) connects the simulation model and control algorithm; in addition, *dq*/*abc* conversion, inverter and measurement module are the fixed connection module, so they are added to the system directly but not modeled here. According to the chosen optional types of the system, the relevant parameters are determined and input into the simulation model. The chosen parameters are depicted in Table 2.

After that, the control algorithm parameters were determined. In IAMPSO, the particle swarm m=100, the number of dimensions D=2, inertia weight ω_i=0.6, acceleration factors $c_1=c_2$=2, velocity range [-1, 1], mutation threshold p_m=0.9, PI controller parameters K_p=[0, 10], K_i=[0, 1], maximum iterations 100 and minimum fitness value 1 were all set. When the iteration times of the algorithm is more than 100 or the fitness value is lesser than 1, the algorithm terminates and the results are obtained. The control algorithm was written based on the settings described above.

Simulation starts after the steps above is completed. The specific connection mode between simulation model and control algorithm can be described as follows: input the K_p, K_i parameters generated by control algorithm into the PI controller of simulation model; run the model and obtain system response performance indicator; input the indicator into control algorithm and run the algorithm. The joint

debugging is achieved and the iteration keeps ongoing in such a cycle. In operation, the PMSM adopts the control mode of i_d=0, given speed 500r/min and the load is the moment of inertia converted to the motor shaft.

In order to illustrate the optimization capability of the algorithm proposed in this research, original PSO and IAMPSO to the optimization of velocity loop PI parameters are applied separately. The fitness value evolutionary process and the K_p, K_i parameters optimization process for the two methods are shown in Fig. 5 and Fig. 6; and the control effect is shown in Table 3.

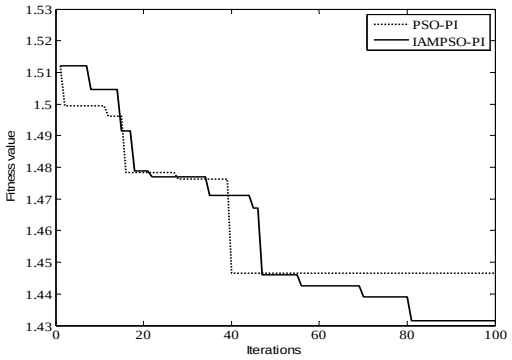
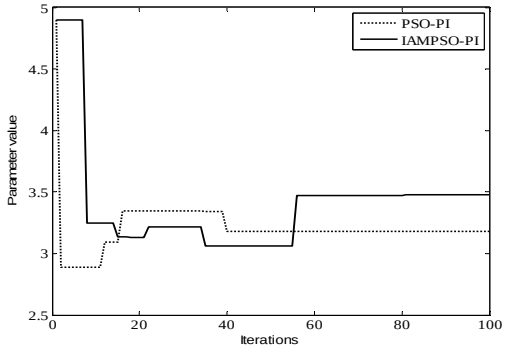
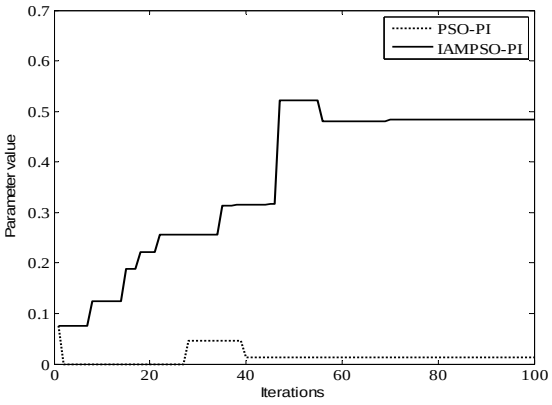


Fig. 5. Fitness value evolutionary process.



(a)K_p optimization process



(b) K_i optimization process

Fig. 6. K_p, K_i optimization process

From Fig. 5, two optimization methods both make the fitness value gradually decreasing along with the iteration, that is to say, they both improve the system response performance gradually. The difference is that the fitness no longer changes after the *40*th iteration of PSO. Thus it can be seen that the algorithm has been trapped into partial optimum solution and is not able to improve the system performance any more. However, IAMPSO skipped this area and continued to search for better solution. The number of effective iterations of IAMPSO compared to PSO is up to more than 40 times. The same iteration law can be found by observing K_p, K_i optimization process in Fig. 6. It is worth mentioning that PSO searches the range of [0, 0.1] approximately in the K_i optimization process, while the IAMPSO's is [0, 0.6] approximately much larger than the PSO's. It shows that IAMPSO is able to search in a larger space than PSO and increase the probability of finding optimal parameters.

Table 3. Results comparison of the two control strategies.

	K_p	K_i	t_r/ms	t_s/ms	$\sigma/\%$	F	ΔF
PSO-PI	3.1772	0.0138	37.7640	66.5969	4.09	1.4466	0.0654
IAMPSO-PI	3.4763	0.4837	29.0924	49.5418	1.57	1.4315	0.0805

The parameters and system performance indicators obtained are listed in Table 3. The PI controller parameters optimized by PSO is [3.1772, 0.0138], while by IAMPSO is [3.4763, 0.4837]. The parameters optimized by IAMPSO are greater than PSO's. Compared to the system using the parameters optimized by PSO, the rise time t_r of the system using the parameters optimized by IAMPSO is shortened by 12.6716ms, the adjustment time t_s is shortened by 17.0551ms and the overshoot σ is reduced by 2.52%. Furthermore, by comparing the fitness values reduced by the two control strategies, it can be seen that IAMPSO improved the system performance by 18.7% more than PSO. All the simulation described above illustrate that the IAMPSO proposed in this paper has better optimization capability than original PSO.

4.2 Experimental verification

The slewing bearing motion control system was designed and the physical system platform was built as Fig. 7 according to the research needs. The system was mainly consisted of controller (PC monitoring software and lower machine PLC), servo drive system (servo amplifier and PMSM) and controlled object (slewing bearing). Among them, PC monitoring software acted as human-machine interface, PLC acted as command transmission and data processing unit, servo drive system acted as actuator, and the system adopted dual closed-loop control mode. In actual operation, the PC monitoring software sent commands to PLC according to the control requirements which contained specified pulse

frequency and number, where the frequency controlled the speed of slewing bearing and the number located slewing bearing; PLC received commands from the host and sent pulse signal to servo amplifier; servo amplifier output voltage signal generated by the internal three-loop controller and electronic gear to the PMSM, and positioning was detected by encoder; in order to avoid the PMSM running at low speed stage and reduce system interference, servo reducer was added to the system because of the PMSM instability at low speed stage; the PMSM connected with slewing bearing and controlled it through servo reducer and drive gear. PC monitoring software monitored the whole system in real-time by reading the PLC operating status information and inner information of servo amplifier. The main function of the software was to monitor the consumed number of pulses, speed, torque, operating voltage and current of PMSM and other information.

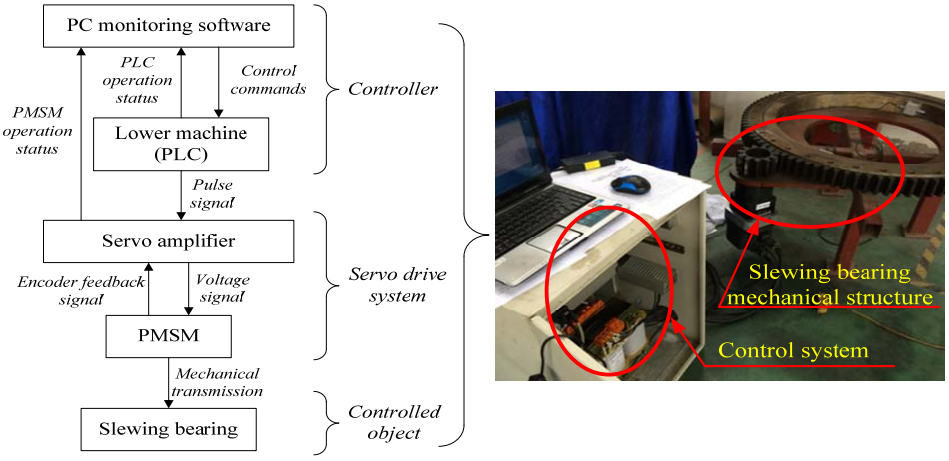


Fig. 7. Slewing bearing motion control system.

The PI controller parameters corresponding to two control strategies were obtained by simulation. The ultimate goal of this study is to improve the response performance of slewing bearing motion control system, so two sets of parameters on the system need to be verified, the experimental equipment was the established physical system platform, which is shown in Fig. 7 above. In the experiment, the parameters were input into servo amplifier and the system operate, thus the real-time speed of PMSM through PC monitoring software was got. Set the motor speed in the experiment same as the simulation for the purpose of improving the comparability. Set the program running process as the motor accelerates from locked state to 500r/min and operates at constant-speed after acceleration, and the sampling period is 1ms. Figure 8 describes the PMSM operation speed curve and the system performance indicators are shown in Table 4.

The measurement error is unavoidable because of the interference of external environment to the system, meanwhile, due to the low sampling precision and graphical display precision, the speed curve in Fig. 8 is a little different with the actual situation, but it reflects the change trend of motor speed substantially. It can be seen from Fig. 8 that the motor started from zero-speed at the zero-position and finally stabilized at 500r/min in accordance with the program commands. On the other hand, by comparing the experimental results in Table 4 with the simulation results in Table 3, it shows that the experimental results are a little bit worse, that is to say, the rise time of the experiment is greater than the corresponding rise time of the simulation under the same control strategy, adjustment time and overshoot as well. This is mainly caused by the precision of the experimental equipment itself. Even the existence of such a situation, it still has research value to do comparative analysis based on the two groups of experiments.

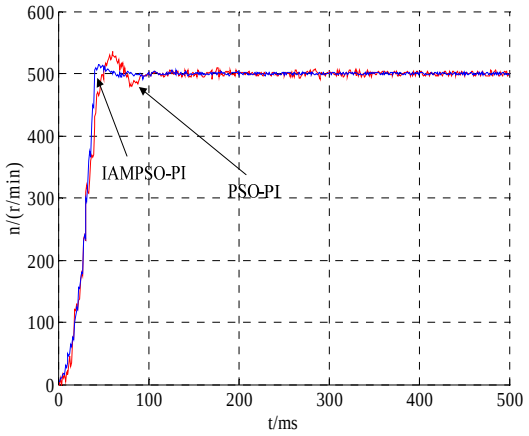


Fig. 8. PMSM operation speed curve.

As seen in Table 4, the rise time t_r, adjustment time t_s and overshoot σ of the system using PSO-PI controller are 48ms, 91ms and 7.32% respectively, while the corresponding indicators of the system using IAMPSO-PI controller are 37ms, 60ms and 3.18%, respectively. The results show that the overshoot of the system using IAMPSO-PI controller is significantly smaller than the overshoot of the system using PSO-PI controller, besides, the rise time and adjustment time are shortened and the speed stability is better in constant-speed-running period. In summary, the dynamic characteristic of the system is significantly improved by IAMPSO. The reason could be concluded by analyzing the parameters optimized by the two control strategies. As we know, the K_p, K_i values optimized by IAMPSO are greater than PSO's. When system overshoot occurs, the greater the

K_p value is, the faster the system response speed is and the better the ability to suppress overshoot is, therefore, the system using IAMPSO-PI controller has shorter acceleration time and smaller overshoot; on the other hand, the greater the K_i value is, the lesser the steady-state error is, so the system using IAMPSO-PI controller has less speed fluctuation.

Table 4. Results comparison of the two controllers.

	t_r/ms	t_s/ms	$\sigma/\%$
PSO-PI	48	91	7.32
IAMPSO-PI	37	60	3.18

5. CONCLUSIONS

To improve the response performance of slewing bearing motion control system, the IAMPSO algorithm is proposed to optimize the internal velocity loop PI controller parameters of the system. The simulation shows that the K_p, K_i parameters obtained by this algorithm is superior to that of the original PSO algorithm; and the experiment verified the effectiveness of this proposed control strategy. It can be seen that the control strategy based on IAMPSO algorithm optimizes the slewing bearing motion control system indeed. By this study, a novel way to improve the motion control level of slewing bearing is provided.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support provided by the National Natural Science Foundation of China (51105191, 51375222), the Shanghai Sailing Program (16YF1408500) and China Postdoctoral Science Foundation (Project No.2015M580632).

REFERENCES

Amasorrain, J.I., Sagartzazu, X., and et. al. (2003). Load distribution in a four contact-point slewing bearing. *Mechanism and Machine Theory*, 38(6), 479-496.

Kennedy, J., and Eberhart, R.C. (1995). Particle swarm optimization. *IEEE International Conference on Neural Networks*, 1942-1948.

Gaing, Z.L. (2004). A particle swarm optimization approach for optimum design of PID controller in AVR system. *IEEE Transactions on Energy Conversion*, 19(2), 384-391.

Hou, R.M., Liu, R.Z., and et. al. (2014). Application of Particle Swarm Optimization Wavelet Neural Network on AC servo system. *Journal of System Simulation*, 26(4), 881-885.

Ji, Q.C., and Sun, Y.G., (2010). Research of position servo system control algorithm. *Electric Drive*, 40(5), 60-62.

Kennedy, J. (2000). Stereotyping: Improving Particle Swarm Performance with Cluster Analysis. In: *IEEE Congress on Evolutionary Computation*, 2, 1507-1512.

Lu, Z.S., and Hou, Z.R. (2004). Particle swarm optimization with adaptive mutation. *Acta Electronica Sinica*, 1(1), 99-104.

Nasri, M., Nezamabadi-Pour, H., and et. al. (2007). A PSO-based optimum design of PID controller for a linear brushless DC motor. *World Academy of Science, Engineering and Technology*, 26(40), 211-215.

Shi, D.W. (2007). *Dynamics of machinery*. China Electric Power Press, Beijing.

Shi, Y. and Eberchart, R.C. (2001). Fuzzy adaptive particle swarm optimization. In: *IEEE Congress on evolutionary computation*, 1(12)，101-106.

Tang, J. (2010). Principle and application of PSO Algorithm. *Computer Technology and Development*, 20(2), 213-216.

Wang, X.L., Zhang, H.Y., and et. al. (2012). Design of three loop control system in NC cam grinding process. *Journal of Jilin University (Information Science Edition)*, 30(1), 40-46.

Xing, Z.X., Zheng, Q.L., and et. al. (2007). PID control in adjustable-pitch wind turbine system based on BP neural network. *Journal of Shenyang University of Technology*, 28(6), 681-686.

Yang, C.H., Gu, L.S., and et. al. (2008). Particle swarm optimization algorithm with adaptive mutation. *Computer Engineering*, 34(16), 188-190.

Zhao, H., Yu, J.L. and et. al. (2013). An improved particle swarm optimization algorithm with invasive weed. *Advanced Materials Research*, 621, 356-359.

Zheng, W.J., and Pi, Y.G. (2016). Study of the fractional order proportional integral controller for the permanent magnet synchronous motor based on the differential evolution algorithm. *ISA Transactions*, 63,387-393.