# Cooperative-PSO-Based PID Neural Network Integral Control Strategy and Simulation Research with Asynchronous Motor Controller Design

PIAO HAIGUO, WANG ZHIXIN，ZHANG HUAQIANG
Department of Electrical Engineering
Shanghai Jiaotong University
No.800 Dong Chuan Road, Shanghai 200240,
CHINA
Email: phg0805@gmail.com

*Abstract:* This paper focuses on the design of robustness controller for asynchronous motor. a new PID neural network-integral (PIDNN-I) synthesis control strategy is proposed for the controller design, in which NARMA-L2, an approximated model of nonlinear auto regressive moving average(NARMA) model, is employed to represent the input-output behavior of the motor and gives out the expected control input. PID neurons network (PIDNN), as a kind of novel neural network model with dynamic characteristics, is adopted in NARMA-L2 to identify the motor. PIDNN integrates the advantages of PID with those of artificial neuron network. However, the conventional back-propagation (BP) algorithm, which easily gets trapped in local minimum and is being adopted in the current model, constrains the identifying ability of PIDNN so as to harm to the completion of the controller design. Particle swarm optimization (PSO) algorithm, a new population-based evolutionary global optimization method, is proposed to replace the BP algorithm to train the neurons model. Cooperative particle swarm optimization (CPSO), an improved version of cooperative random learning particle swarm optimization (CRPSO), is put forward to enhance the performances of the conventional PSO in the design. Due to the existence of the tracking error caused by approximate error between identifying and real system, integral (I) control is introduced into the design, namely adopting PIDNN control in large tracking error scale and PIDNN-I control in small tracking error scale. Compared with conventional PID control strategy, simulation results demonstrate that the CPSO-based PIDNN-I synthesis control strategy has improved the control performances of asynchronous motor in robustness and accuracy efficiently.

*Key-Words:-* NARMA-L2; PID neural network; integral; synthesis control; asynchronous motor; BP; PSO; Nonlinear; Cooperative particle swarm optimization;

## 1. Introduction

Asynchronous motors are most widely used in various fields of industry owing to its greatly advantageous nature, but are difficult to be controlled to get high control performances since they are a five-order, nonlinear and strongly coupling system.[1-3]. The nonlinear and parameter time-varying characteristics make asynchronous motor become a complex control object. These days, vector control technology, as a common method, is employed to control asynchronous motor. The controller's design based on the technology for asynchronous motor becomes as easy as DC motor does. However, the technology highly depends on the mathematical model of asynchronous motor so that the parameter-robustness and the resistant ability for the load-disturbance are weak. Moreover, the dynamic influences of non-modeling parts, such as the change of load torque, iron losses and magnetic saturation etc., are existent. All of these increases the difficulties of controlling asynchronous motor to get higher performances. Various controllers or modified control strategies are proposed to overcome the above shortcomings for getting good control performances[4-7].

Due to the nonlinear essence and excellent ability of function approximation, artificial neural networks (ANNs) is introduced into the control field for controller's design[8-10]. However, ANN-based controllers isn't used widely in control field. One of the reasons for such situation is that the neurons in most neural networks are only of static function, which are

not suitable for the control of actual dynamic system[11]. In view of the good dynamic performance of PID and merits of artificial neural network, SHU Huailin [12] firstly proposed P, I, D neurons and designed PID neural network (PIDNN). Instead of being a simple hybrid system of the PID and neural network, PIDNN is actually a new kind of dynamic neural network. Some PIDNN-based controllers had been designed to control linear or nonlinear systems and got better control performances[13-15]. The training algorithm for the above controllers' design is conventional BP algorithm, whose training results is greatly influenced by the initialized values and learning rate. The ability of global exploitation is not strong and easy to trap in local minimum [16,17]. BP algorithm constrains the wide application of PIDNN controller .

In view of the un-mathematics-based function approximation characteristic of ANN and the dynamic characteristic of PIDNN, this paper proposes to design a PIDNN-based controller for asynchronous motor for getting high control performances. The applied form of PIDNN, the overcoming of BP algorithm's defects and the offset of the error caused by function approximation make the design different from other ANN-based design methods.

NARMA model, which was first introduced in 1985 by Leontraritis and Billings [18], can accurately represent the input-output behavior of finite-dimensional nonlinear discrete-time dynamical systems in a neighborhood of the equilibrium state. Asynchronous motor, as an dynamical system, is suitable to be represented by the model. However, NARMA model is not convenient for the purpose of control using neural network due to its nonlinear dependence on the control input and its high demand in complicated computation. In reference [18], Narendra and Mukhopadhyay proposed an approximated version of NARMA named NARMA-L2, which has linear relationship with control input and can simplify the practical implementation of the neural controller with the theoretical analysis. In this paper, NARMA-L2 is employed for PIDNN to identify the asynchronous motor. And the designed PIDNN controller will be given out after the identification by an algebraic way. However, the designed controller could not work well since the BP algorithm used in PIDNN greatly influences the results of identification. A newly effective algorithm with stronger global and local search abilities needs to be employed for PIDNN to complete the design successfully.

PSO algorithm, a new population-based evolutionary global exploitation algorithm, was first introduced in 1995 by Eberhart and Kennedy [19]. The algorithm gets good performance in solving real-valued optimization problems, and can efficiently locate the basins of the optima. In fact, the training process of PIDNN is to find the optimum weight values in solution space, so PSO algorithm can be expected to get better training results than BP algorithm. However, the local search ability of PSO algorithm is still poor. In order to overcome the deficiencies of PSO, Liang Zhao & Yupu Yang [17] firstly proposed a modified version of PSO named CRPSO, which evolves multiple sub-swarms simultaneously and uses a randomly selected *gbest* (the best position of all particles found by their neighbors or itself) from all the sub-swarms to calculate the velocity and position of particle. The application of the algorithm in single neuron model for a variety of time series prediction problems validates the algorithm, however, the mechanism of randomly selected *gbest* can't get the most satisfying results for local search. In this Paper, cooperative particle swarm optimization (CPSO) algorithm, which is a modified version of CRPSO and proposed by the authors, is adopted for PIDNN to identify the system. The identifying results based on BP, PSO, CRPSO and CPSO algorithms demonstrate that the latter three algorithms highly enhance the identifying ability of PIDNN and only CPSO in the four algorithms can be used for the design's completion with a satisfactory result.

In order to offset the tracking error caused by approximate error between identified and real system, integral control is introduced into the design and a new design method named PIDNN-I synthesis control strategy is proposed, namely adopts PIDNN control in large error scale and PIDNN-I control in small error scale. Compared with conventional PID control strategy, the simulation results indicate that CPSO-based PIDNN-I control strategy greatly improves the control performances of asynchronous motor in accuracy and robustness and can also be used in other dynamic nonlinear system.

The rest of the paper is organized as follows: the NARMA-L2 method and the basic PIDNN neurons model with BP algorithm is introduced in Section 2. The standard PSO, CRPSO and proposed CPSO algorithms are provided in details in Section 3. Section 4 gives out the design process of PIDNN-I controller. In Section5, the simulation experiments and analysis with three different situations are done. And the conclusions are drawn in Section 6.

## 2 NARMA-L2 Method with PIDNN Neural Model

In this section, the NARMA , NARMA-L2, forward propagation of PIDNN and cost function with BP algorithm used in PIDNN will be introduced in

details, which is the theoretical basic for the controller's design.

## 2.1 NARMA Model

A nonlinear dynamical system can be represented by state equations as follows:

$$\sum: \quad \begin{cases} x(k+1) = f[x(k), u(k)] \\ y(k) = h[x(k)] \end{cases} \tag{1}$$

where, $\{u(k)\}, \{x(k)\}, and \{y(k)\}$ are discrete time sequences; $f(\cdot)$ and $h(\cdot)$ are mapping functions. After multi-steps iterated computation, from the eq.(1), we can have:

$$Y_n(k) = \psi[x(k), U_{n-1}(k)] \tag{2}$$

where, $Y_n(k)$ denotes the sequence $y(k), y(k+1), \cdots, y(k+n-1)$ and $U_{n-1}(k)$ denotes $u(k), u(k+1), \cdots, u(k+n-2)$ ; $\psi(\cdot)$ is mapping function. It can be know from the eq.(1) that the state $x(k+n)$ depends only on the state $x(k)$ and the input sequence $U_n(k)=[u(k),u(k+1),\cdots,u(k+n-1)]$, thus, from the eq.(2) based on the implicit function theorem, $x(k+n)$ can be expressed as:

$$\begin{aligned} x(k+n) &= \bar{g}[Y_n(k), U_n(k)] \\ &= \bar{g}[y(k), y(k+1), \cdots, y(k+n-1), \\ &\quad u(k), u(k+1), \cdots, u(k+n-1)] \end{aligned} \tag{3}$$

Since $y(k+n) = h[x(k+n)]$ according to the eq.(1), this leads to the results that NARMA model can be expressed as:

$$\begin{aligned} y(k+1) = F[&y(k), y(k-1), \cdots, y(k-n+1), \\ &u(k), u(k-1), \cdots, u(k-n+1)] \end{aligned} \tag{4}$$

If a relative degree *d* is taken into account, then the eq.(4) changes as follows:

$$\begin{aligned} y(k+d) = F[&y(k), y(k-1), \cdots, y(k-n+1), \\ &u(k), u(k-1), \cdots, u(k-n+1)] \end{aligned} \tag{5}$$

Based on the eq.(5), the model identified by using neural network can be expressed as:

$$\begin{aligned} \hat{y}(k+d) = N[&y(k), y(k-1), \cdots, y(k-n+1), \\ &u(k), u(k-1), \cdots, u(k-n+1)] \end{aligned} \tag{6}$$

Where $\hat{y}(k+d)$ is the estimate of $y(k+d)$ and *N(•)* is network mapping. In view of control problem, if the expected outputs $y^*(k+d)$ is given out, then the eq.(6) based on the implicit function theorem should become as follows:

$$\begin{aligned} u^*(k) = G[&y(k), y(k-1), \cdots, y(k-n+1), \\ &y^*(k+d), u(k-1), \cdots, u(k-n+1)] \end{aligned} \tag{7}$$

Where $u^*(k)$ is the expected control output at instant *k*; *G(•)* is network mapping. Compared the eq.(6) and eq.(7), the changing from *N(•)* to *G(•)* needs another neural network to finish the controller design, which makes the controller complicated and increases the computing task.

## 2.2 NARMA -L2 Model

NARMA-L2 is the Taylor expansion of NARMA around the scalar $u(k)$ , the remainder *R* is negligible when input *u* is small enough, namely, NARMA-L2 can be made as accurate as desired by decreasing the amplitude of the input *u*. The model can be expressed as follows:

$$\begin{aligned} y(k+d) = &f_0[y(k), y(k-1), \cdots, y(k-n+1), u(k-1), \cdots, u(k-n+1)] \\ &+ g_0[y(k), y(k-1), \cdots, y(k-n+1), u(k-1), \cdots, u(k-n+1)] \cdot u(k) \end{aligned} \tag{8}$$

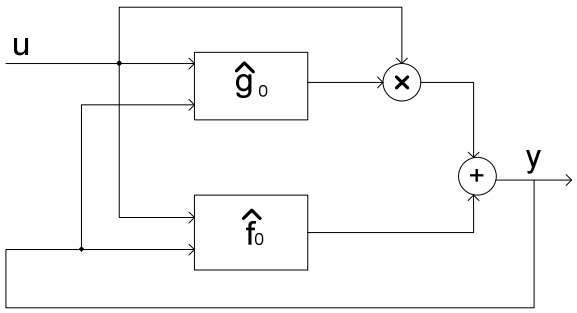
So the identification model using neural network can be expressed as:

$$\begin{aligned} \hat{y}(k+d) = &\hat{f}_0[y(k), y(k-1), \cdots, y(k-n+1), u(k-1), \cdots, u(k-n+1)] \\ &+ \hat{g}_0[y(k), y(k-1), \cdots, y(k-n+1), u(k-1), \cdots, u(k-n+1)] \cdot u(k) \end{aligned} \tag{9}$$

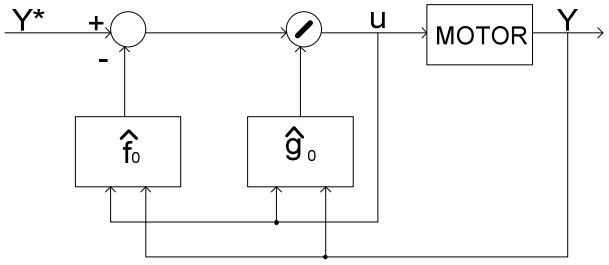
Where $\hat{y}(k+d)$ is the estimate of $y(k+d)$ ; $\hat{f}_0(\cdot)$ and $\hat{g}_0(\cdot)$ are network mapping. From the eq.(9), we can get the control input $u(k)$ after the expected outputs $y^*(k+d)$ is given out :

$$u(k) = \frac{y^*(k+d) - \hat{f}_0[y(k), y(k-1), \cdots, y(k-n+1), u(k-1), \cdots, u(k-n+1)]}{\hat{g}_0[y(k), y(k-1), \cdots, y(k-n+1), u(k-1), \cdots, u(k-n+1)]} \tag{10}$$

Compared with the eq.(7), the eq.(10) indicates that the design of controller becomes easy, which can give out the expected $u(k)$ only by algebraic way. The sketch map of identification and control structure is shown in Fig. 1.




**(a) identification model**




**(b) control model**

**Fig. 1 the sketch map of identification and control structure based on NARMA-L2**

## 2.3 PIDNN Neural Model

The basic structure of PIDNN consists of three layers, i.e., the input layer, the hidden layer and the output layer, respectively. The input layer is made up of two proportional neurons used for the inputs of set value and feedback value. The output layer has one proportional neuron, and exports the control value of object system. The hidden layer, which is the core of the PIDNN structure, is composed of three neurons: proportional (P) neuron, integral (I) neuron and derivative (D) neuron. The basic structure of PIDNN predicted model is shown as Fig. 2.

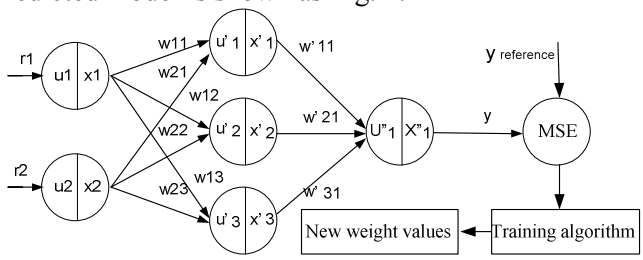



**Fig. 2. The basic structure of PIDNN predicted model**

### 2.3.1 Forward Propagation Algorithm of PIDNN

In Fig. 2, the forward propagation algorithm of the model is also composed of three parts.

In the input layer, the input-output functions of the two P-neurons are defined as follows:

$$x_i(k)=\begin{cases} q' & u_i(k)\ge q \\ u_i(k) & -q<u_i(k)<q \\ -q' & u_i(k)\le -q \end{cases} \quad (11)$$

where, $i$=1,2, is the input number and $k$ is the sample time. $q$ is the maximum limitation value of the input layer and $q'$ is the actual output value after the input value exceeds $q$. $u_i(k)$ and $x_i(k)$ are the respective input and output values of the $i$ neuron at the $k^{th}$ sample time.

In the hidden layer, the neurons' inputs are defined as:

$$u'_j=\sum_{i=1}^{2} w_{ij}\,x_j(k) \quad (12)$$

where, $j$=1,2,3, is the neuron number of hidden layer and $w_{ij}$ are the connective weight values between the input and hidden layers. The input-output functions of the P, I, D neurons in the hidden layer are different from each other and are defined as follows:

$$x'_1(k)=\begin{cases} q' & u'_1(k)\ge q \\ u'_1(k) & -q<u'_1(k)<q \\ -q' & u'_1(k)\le -q \end{cases} \quad (13)$$

$$x'_2(k)=\begin{cases} q' & u'_2(k)\ge q \\ x'_2(k-1)+u'_2(k) & -q<u'_2(k)<q \\ -q' & u'_2(k)\le -q \end{cases} \quad (14)$$

$$x'_3(k)=\begin{cases} q' & u'_3(k)\ge q \\ u'_3(k)+u'_3(k-1) & -q<u'_3(k)<q \\ -q' & u'_3(k)\le -q \end{cases} \quad (15)$$

where Eqs. (13-15) are the function of P-neuron, I-neuron and D-neuron, respectively.

In the output layer, the input of the only P-neuron is defined as:

$$u''_h=\sum_{j=1}^{3} w_{jh}\,x'_j(k) \quad (16)$$

where $h$=1, is the output number and $w_{jh}$ are the connective values between the hidden and output layers. The input-output function is defined as:

$$x''_h(k)=\begin{cases} q' & u''_h(k)\ge q \\ u''_h(k) & -q<u''_h(k)<q \\ -q' & u''_h(k)\le -q \end{cases} \quad (17)$$

### 2.3.2 Cost Function and BP Algorithm

In order to make the model functional, a cost function is necessary. The cost function used in Ref. [12] is named MSE (mean square error), which is a network performance function and employed to measure the network's performance according to the mean of squared errors. The MSE used in basic PIDNN is defined as follows:

$$J=E_h=\frac{1}{m}\sum_{k=1}^{m}[y_{reference}(k)-y(k)]^2 \quad (18)$$

where, $m$ is the number of sampling points; $y_{reference}(k)$ and $y(k)$ are the reference and output values of the predicted model, respectively.

BP algorithm is a common method and used widely in neural network. The standard BP algorithm adopted by Shu Huailin in PIDNN control system [12] is based on the steepest descent gradient approach. The Eq. of the learning rules for P-I-D predicted model is given as follows:

$$\Delta w_{ij} = -\eta \frac{dJ}{dw_{ij}} = \begin{cases} \frac{\eta}{m}\sum_{k=1}^{m} 2\cdot(y_{ref}-y)\cdot w_{jh}\cdot 1\cdot x_i & P_{neuron} \\ \frac{\eta}{m}\sum_{k=1}^{m} 2\cdot(y_{ref}-y)\cdot w_{jh}\cdot u'_j\cdot x_i & I_{neuron} \\ \frac{\eta}{m}\sum_{k=1}^{m} 2\cdot(y_{ref}-y)\cdot w_{jh}\cdot dx'_j/du'_j\cdot x_i & D_{neuron} \end{cases} \quad (19)$$

$$\Delta w_{jh} = -\eta \frac{dJ}{dw_{jh}} = \frac{\eta}{m}\sum_{k=1}^{m} 2\cdot(y_{ref}-y)\cdot x'_j \quad (20)$$

where $\eta$ is the learning rate and $dx'_j/du'_j$ is replaced by $sign(\frac{x'_j(k)-x'_j(k-1)}{u'_j(k)-u'_j(k-1)})$. As a common factor in Eqs. (19), the positive or negative of $dx'_j/du'_j$ decides the changing directions, while its numerical value only influences the changing rate, which can be offset by modulating the learning rate $\eta$. Thus, the replace is permissive. After $n_0$ steps, the modified formulas of weight can be expressed as;

$$\begin{cases} w_{ij}(n_0+1) = w_{ij}(n_0) + \Delta w_{ij} \\ w_{jh}(n_0+1) = w_{jh}(n_0) + \Delta w_{jh} \end{cases} \quad (21)$$

# 3 CPSO Learning Algorithm

This section will introduce the standard PSO, CRPSO and proposed CPSO algorithms used for replacing BP. The updating formulas and sharing mechanism will be given and discussed in details, which is beneficial to understanding the essences of each algorithm and knowing why CPSO algorithm is important for the controller's design.

## 3.1 The Standard PSO Algorithm

PSO is a novel evolutionary algorithm which comes from the imitation of birds flocking or fish schooling looking for food [20,21]. Each particle of PSO has a position and a velocity used to represent the solution of the optimization problem and the search direction in the solution space, respectively. Each particle of swarm adjusts the velocity and position of themselves according to the best experiences called *pbest* found by itself and *gbest* found by either all its neighbors or itself. The updating formulas of the velocity and position about the particle are defined as follows:

$$\begin{cases} v(k+1) = \omega v(k) + c_1\cdot r_1\cdot[p_{best}-x(k)] + c_2\cdot r_2\cdot[g_{best}-x(k)] \\ x(k+1) = x(k) + v(k+1) \end{cases} \quad (22)$$

where $v$ and $x$ are the velocity and position of the particles and $n$ is the updating times. $\omega$ is the inertia weight introduced to balance global and local exploitation abilities of algorithm. $c_1$ and $c_2$ are positive constants named acceleration constants whereas $r_1$ and $r_2$ are random numbers between 0 and 1. $p_{best}$ is the best position of each particle found by themselves while $g_{best}$ is the best position of all particles found by their neighbors or itself.

In terms of optimization, PSO algorithm can locate the basins of the global optimization quickly. However, initial PSO suffers from a serious problem that is often unable to explore the basin quickly. To tackle this problem, Eberhart and Shi introduced a time decreasing inertia factor to balance the ability of global exploitation and local exploitation [22,23].

## 3.2 The CRPSO Learning Algorithm

In order to overcome the shortcomings of PSO, under the cooperative search framework, Liang Zhao & Yupu Yang[17] introduces an improved PSO algorithm named cooperative random learning PSO (CRPSO). This algorithm employs a cooperative random learning mechanism to get the balance between global and local search. The aim of the algorithm was to search different portions of the solution space effectively by using multiple sub-swarms.

In CRPSO algorithm, the particles in each sub-swarm learn the *gbest* randomly among the *gbest*s found by all sub-swarms when updating their velocity and position. The velocity updating formula of CRPSO is defined as:

$$\begin{cases} v_i(k+1) = \omega\cdot v_i(k) + c_1\cdot r_1\cdot[p_{best}-x_i(k)] \\ \qquad\qquad + c_2\cdot r_2\cdot[g_{best}(r)-x_i(k)] \\ x(k+1) = x(k) + v(k+1) \end{cases} \quad (23)$$

where $i=1,\ldots,m$, is the number of sub-warms and $r$ is a random integer from 1 to $m$ which is used to randomly select the *gbest* between different sub-swarms so as to share the information and enhance the ability of global and local search.

## 3.3 The CPSO Learning Algorithm

Although CRPSO algorithm improves the ability of global search, the mechanism of randomly selecting

the *gbest* excessively depends on the other sub-swarms' *gbest* while ignores the *gbest*'s use of sub-swarm itself. This results in CRPSO not being able to effectively enhance the ability of local search. To avoid the defects of CRPSO, as a modified version, cooperative particle swarm optimization is proposed by the authors. In CPSO algorithm, the particles in each sub-swarm simultaneously use the *gbest* found by all the other sub-swarms randomly and by the sub-swarm itself to update their velocity and position. The updating formulas of CPSO algorithm are defined as:

$$\begin{cases} v_i(k+1) = \omega \cdot v_i(k) + 0.5 \cdot c_1 \cdot r_1 \cdot [p_{best} - x_i(k)] \\ \qquad\qquad +0.5 \cdot c_2 \cdot r_2 \cdot [g_{best} - x_i(k)] \\ \qquad\qquad +0.5 \cdot c_2 \cdot r_3 \cdot [g_{best}(r) - x_i(k)] \\ x(k+1) = x(k) + v(k+1) \end{cases} \quad (24)$$

where $r_1$, $r_2$ and $r_3$ are random numbers between 0 and 1, and *0.5* is used to balance the role of *pbest*, *gbest* and *gbest(r)*.

As seen from the Eq.(22)、Eq. (23), the updating formulas are similar and the velocities are determined by three parts. The first part is $\omega \cdot v(n)$, which, as a momentum term, fills the role of preventing excessive oscillations in the search direction. The second part, $c_1 \cdot r_1 \cdot [p_{best} - x(n)]$, is the individual part of a particle, which is employed to explore the optimal position towards the best position found by the particle itself. The last one, $c_2 \cdot r_2 \cdot [g_{best} - x(n)]$ for PSO and $c_2 \cdot r_2 \cdot [g_{best}(r) - x_i(n)]$ for CRPSO, is the social part of a particle, which is employed to explore the optimal position towards the best position located by all particles. In fact, the third part plays the role of sharing the social information. However, from their differences, it can be known that the sharing strategy of PSO is done only in one swarm, whereas the sharing strategy of CRPSO is done in different swarms through selecting the *gbest* randomly. The initialization and evolving process of each sub-swarm in CRPSO is performed independently, in which the particles don't always move toward a single *gbest* position. Therefore, the sharing mechanism of CRPSO is novel, which can maintain the diversity of swarm effectively and find more feasible solutions due to the enlarged search space. However, according to the knowledge of probability, the sharing mechanism of CRPSO weakens the role of the *gbest* found by the swarm itself. That is to say, the sharing mechanism excessively emphasizes the role of the *gbest*s found by the other swarms. The bigger the number of swarms is, the heavier the role of the other swarms is emphasized. Thus, in actual evolving process, the *gbest* of each swarm itself holds the possibility of not being employed. Furthermore, the *gbest*s found by the other swarms don't indicate that it is more beneficial for the updating of particle than the *gbest* found by the swarm itself. Therefore, although CRPSO increase the probability finding the global optimal, the local search ability isn't always better than PSO, i.e., the sharing mechanism of CRPSO doesn't significantly improve the local search ability.

In order to exert the information sharing mechanisms of CRPSO and PSO effectively, in CPSO, in addition to the *gbest* found by the swarm itself that is fixed in the updating formula as PSO does, the *gbests* found by the other swarms are randomly introduced into the updating formula simultaneously. As seen in the Eq.(24), the velocity of a particle is determined by four parts. It is obvious that CPSO integrates the merits of PSO and CRPSO and forms a new information sharing strategy. This new sharing strategy makes the CPSO have both strong global and local search abilities. Since more useful information is used in the evolving process of a particle, the convergent speed of CPSO is faster and it is also easier to find the optimal values, which enhances the robustness of the algorithm. The schematic diagram of CPSO is shown in Fig. 3 where each circle represents a sub-swarm with its *gbest*. It requires two steps for a particle in each circle to update its velocity, i.e., the first one randomly selects a *gbest* from the connective circles, and the second uses the selected *gbest* with the *gbest* itself to complete the updating process. As can be seen, Fig. 3 expresses the idea of balancing the role of *gbest* between sub-swarm itself and the other sub-swarms. From the experiments in section 5, it will be known that the search abilitys of the algorithms for the optimal values become a crucial point for the design's completion with a satisfactory result.

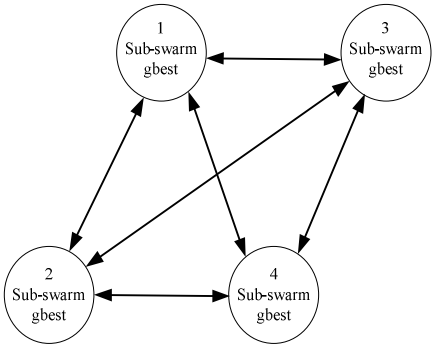



**Fig. 3 The schematic diagram of the information sharing of CPSO**

### 3. 4 The Training Process of PIDNN with CPSO

The training process of CPSO based PIDNN can be simply summarized as the following steps:

Step 1. Initialize all parameters of PIDNN and CPSO and define the dimensions and range of positions and velocities.

Step 2. Initialize the positions and velocities of the particles in all sub-swarms.

Step 3. Calculate the first step's cost function value of each particle, namely the fitness of all particles, and initialize the *pbest* and *gbest* of each sub-swarm and save the minimum of *gbest*s as *Gbest*.

Step 4. Generate a stochastic integer *r* between 1 and the number of the sub-swarms.

Step 5. Update the velocity and position of particle *i* in sub-swarm according to Eq. (24).

Step 6. if $f(x_i)<f(p_{best})$, update the *pbest* of the sub-swarm.

Step 7. if $f(p_{best})<f(g_{best})$, update the *gbest* of the sub-swarm.

Step 8. Save the minimum of *gbest*s gained by sub-swarms to *Gbest.*

Step 9. Repeat Step 4~8 until meeting the setting maximum of cycle.

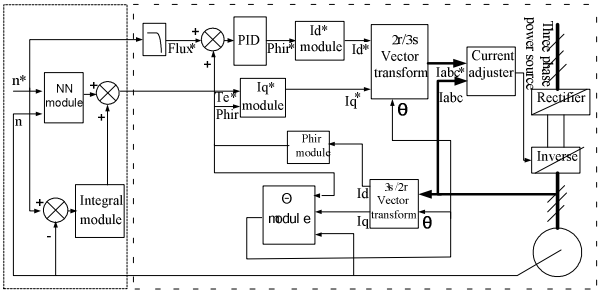
Where $f(x_i)$, $f(p_{best})$, $f(g_{best})$ represent the fitness of particle *i*, *pbest* and *gbest,* respectively.

## 4 PIDNN-I Synthesis Controller's Design

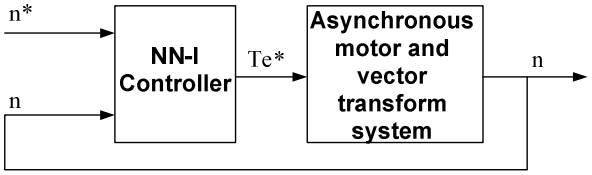
In this section, based on the model and algorithms mentioned in section 2 and 3, the designing process of the CPSO-based PIDNN-I synthesis controller will be presented.

### 4.1 PIDNN-I Control System

The control sketch map of PIDNN-I control system is shown as fig.4-a. In spite of the defects introduced in section 1, vector control technology is still a common and useful method for the decoupling control. In view of the advantages, vector control is remained in this paper. PID controller is enough for the control of flux to ensure the control accuracy. The disadvantageous effect of flux to the speed control can be seen as an uncertain disturbance and be offset by PIDNN-I speed controller. Therefore, the section in dotted line can be denoted by a whole module. The simplified control sketch map is shown as Fig. 4-b.




**(a) NN-I sketch map with vector transform and motor**




**(b) simplified sketch map**

**Fig.4 Sketch map of NN-I Control System**

### 4.2 PIDNN-I Controller Design

PIDNN-I controller design comprises of two different sections as seen in fig.4-a: namely PIDNN controller design and I controller design. The two controllers give out the expected control input together.

PIDNN controller design is based on the NARMA-L2 model, which means that two stages need to be completed. During the identification stage, PIDNN is employed to identify the mapping functions $f_0(\cdot)$ and $g_0(\cdot)$ of NARMA-L2 model. The identified results are known as the network mappings named $\hat{f}_0(\cdot)$ and $\hat{g}_0(\cdot)$. The sketch map of identified structure based on PIDNN is shown as Fig.5-a. Owing to the strong search ability of CPSO algorithm and dynamical characteristic of PIDNN, the identifying accuracy of the model is very high, which is beneficial to the controller design and will be seen in section 5. According to the principle of NARMA-L2 model, the design of PIDNN controller becomes simple and easy by changing the form of identification structure after the identifying stage, which is only a algebraic transform. The sketch map of control structure is shown in fig.5-b. Where, the *s* in weight value is the number of basic PIDNN and the number equals to 1 and 2 in $\hat{f}_0(\cdot)$ and $\hat{g}_0(\cdot)$ mapping function, respectively; $K_1, K_2, K_3,$ and $K_4$ are ratio factors.

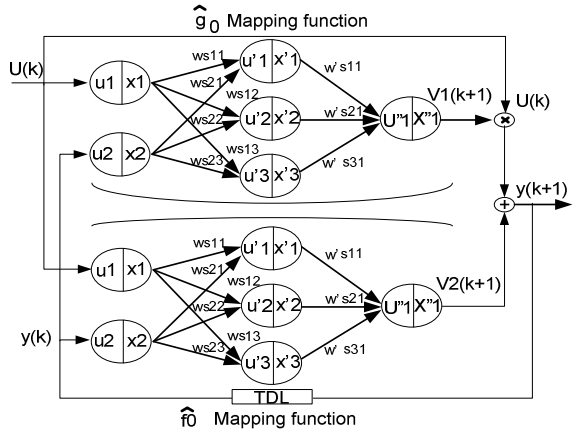
As a dynamic neural network, CPSO-based PIDNN can decrease the identified error and approximate the real system effectively. However, the error between identified and real model still exists, which leads to the departure of the control input signal from the expected. In view of the fact above and in order to further enhance the control robustness and accuracy, this paper first proposed PIDNN and integral synthesis

control strategy, which is adopting PIDNN control in large tracking-error scale and PIDNN-I control in small tracking-error scale. To avoid integral saturation, the scale of speed error $e$ shouldn't be too large; however, the error can't be too small to be integral enough. The integral domain selected in this paper is [-15,15], and the integral factor is 2000 to quicken the integral process.

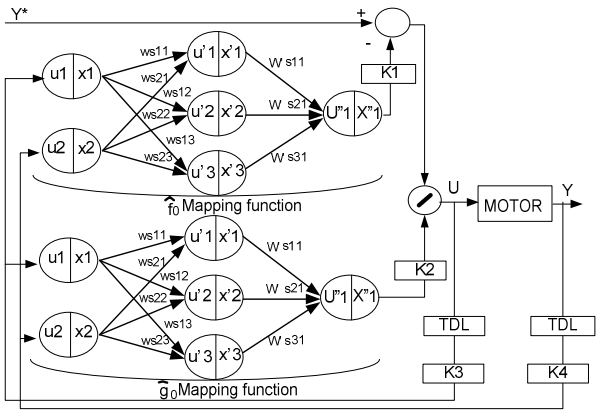
Actual input and output need to take scale transform. The former is transformed from actual scale to required domain. The latter is opposite. The formula of scale transform can be expressed as:

$$\begin{aligned} x^* &= 2\cdot(x-x_{\min})/(x_{\max}-x_{\min})-1 \\ x &= (x^*+1)\cdot(x_{\max}-x_{\min})/2\cdot(x-x_{\min}) \end{aligned} \quad (25)$$

Where, $x_{\max}$ and $x_{\min}$ is the up and down limitation of input or output; $x$ is the actual input or output value and $x^*$ is the transformed value for input or output. In Fig.6-b, $K_1=K_2$, is used for output ratio transform and $K_3=K_4$, is used for the input ratio transform.




**(a) identification structure based on PIDNN**




**(b) control structure based on PIDNN**

**Fig. 5 Structure sketch map of NN controller**

## 5. RESULTS AND ANALYSIS

Some experiments will be done in this section to validate the proposed CPSO-based PIDNN-I synthesis control strategy. Three kind of situations will be considered in experiments and the analysis according to the results of experiments will be given. The simulation tools used in this paper for the experiments is MATLAB™.

### 5.1 The Parameters of Motor, PIDNN and CPSO

To validate the control strategy proposed in this paper, a 5kW asynchronous motor is selected. The whole parameters of the motor can be obtained from Table 1.

The parameters of PIDNN with CPSO are shown in Table 2. The initialization formulas of position and velocity about each sub-swarm in CPSO algorithm can be expressed as:

$$\begin{cases} pos = X\min+(X\max-X\min)\cdot rand \\ vel = V\min+2\cdot V\max\cdot rand \end{cases} \quad (26)$$

Where $pos$ and $vel$ are the random initialization value of position and velocity; $rand$ is a random value between 0 and 1; $X\max$ and $X\min$ are the up and down limitation of weight value in PIDNN. $V\max$ and $V\min$ are the up and down limitation of velocity in CPSO. It is known from Section 2 that the weight values can be divided into two parts according to the layer's difference. Thus, the search scale and search velocity can also be set with different values based on the layer's difference. The limitation values of velocity based on the limitation values of position can be expressed as:

$$\begin{cases} V\max_{sij} = -V\min_{sij} = 0.1\cdot(X\max_{sij} - X\min_{sij}) \\ V\max_{sjh} = -V\min_{sjh} = 0.1\cdot(X\max_{sjh} - X\min_{sjh}) \end{cases} \quad (27)$$

Where the meanings of $s$, $i$, $j$ and $h$ is as before. setting $Xmax_{s\,i\,j} = -Xmin_{s\,i\,j}=1$, $Xmax_{s\,j\,h} = -Xmin_{s\,j\,h}=2$ ,then the maximum and minimum of velocity used in simulation are: $Vmax_{s\,i\,j} = -Vmin_{s\,i\,j}=0.2$; $Vmax_{s\,j\,h} = -Vmin_{s\,j\,h}=0.4$, just as seen in Table 2.

**Table 1**
**Parameters of yawing synchronous motor**

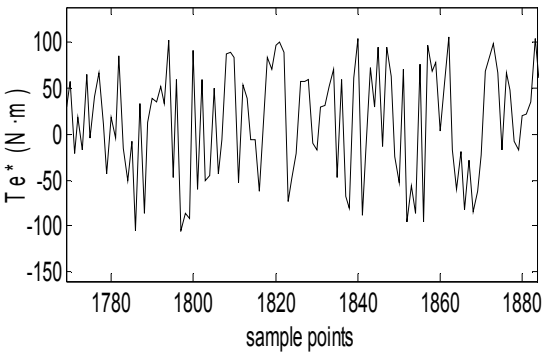
| Parameters | value | unit |
|---|---|---|
| $P_N$ | 5 | kW |
| $U_N$ | 380 | V |
| Polar pairs | 2 | |
| Ls | 74.3 | mh |
| Rs | 0.45 | Ω |
| Lm | 70.3 | mh |
| Rr | 0.85 | Ω |
| Lr | 74.3 | mh |
| J | 0.11 | Kg·m$^2$ |

**Table 2**
**Parameters of PIDNN and CPSO**

## 5.2 System Identification

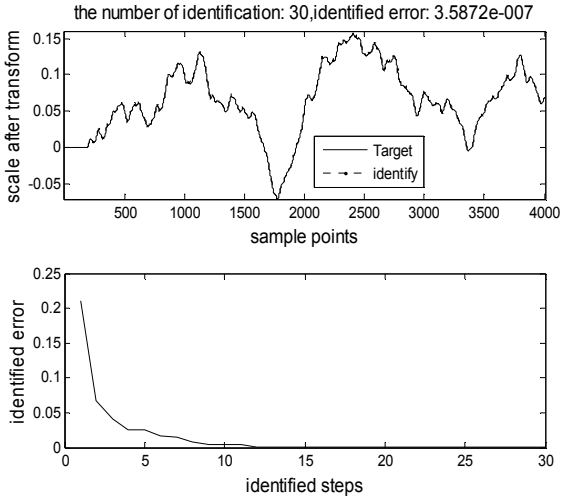
The data for identification come from the random input and corresponding output. The data curves are shown in Fig. 6. Where, the Fig.6-a shows the random input curve of electromagnetic torque. And in Fig.6-b, the corresponding output curve with the identified curve and the error curve are shown. As is observed from Fig.6 that the output and identification curves are almost coincident completely while the identified steps only need 30 steps and identified accuracy is very high, i.e. the identified error is 3.5872e-7, which indicates that the CPSO-based PIDNN can identify the asynchronous motor effectively since CPSO algorithm with strong search ability easily find out the global optimum solution in solution space.

To test the search ability of CPSO, a contrasting experiment has been done, and the error results of experiment are given out in Table 3. There are twelve times trainings for each algorithm. The parameters of PSO and CRPSO are the same as CPSO except that the particle number of PSO is ninety. The data in the table 3 indicates that the error values of BP are biggest, which indicates that the search ability of BP algorithm is the poorest and isn't suitable for PIDNN to identify the dynamic system and complete the design of PIDNN-I controller. Compared with BP algorithm, PSO, CRPSO and CPSO algorithms have stronger search ability. However, the error values of PSO algorithm are still bigger and not suitable for PIDNN to identify the object. Though CRPSO, as a modified version of PSO, has made a progress in search ability, the error values in Table 3 indicate that CRPSO algorithm can't enhance the ability of local search effectively, so the control performances of PIDNN controller based on CRPSO aren't always better than the PIDNN controller with PSO. If 1e-6 is used as a threshold value, the successful number of CPSO is absolutely the biggest, which indicates clearly that CPSO algorithm has the strongest search ability in four algorithm and greatly improves the identified performance of PIDNN.




**(a) input data curve**

| Parameter of PIDNN | value | Parameter of CPSO | value |
|---|---|---|---|
| s (the number of inputs) | 2 | sub-swarm | 3 |
| h (the number of outputs) | 2 | particle number of each sub-swarm | 30 |
| q | 1 | range of position $x_{sij}$ | [-1 1] |
| q' | 1 | range of position $x_{sjh}$ | [-2 2] |
| samples | 4008 | the dimension of the space | 18 |
| max-echo step | 30 | range of velocity $v_{sij}$ | [-0.2 0.2] |
| | | range of velocity $v_{sjh}$ | [-0.4 0.4] |




**(b) identified and error curve with output curve**

**Fig.6 the curves of input, output, identification and error**

**Table 3**
**Error values of four algorithms in 12 times training**

| algorithm | error value (1e-007) | | | | | |
|---|---|---|---|---|---|---|
| BP | 65541 | 55412 | 79654 | 33221 | 55562 | 22391 |
| | 22390 | 22390 | 21334 | 21334 | 21533 | 21334 |
| PSO | 265.3 | 689.6 | 429.0 | 411.5 | 589.1 | 149.0 |
| | 184.1 | 84.97 | 308.2 | 9101 | 442.2 | 10297 |
| CRPSO | 30.28 | 83.63 | 142.3 | 3155 | 7.901 | 37.34 |
| | 226.7 | 683.3 | 522.0 | 17.79 | 981.0 | 2642 |
| CPSO | 2.733 | 22.87 | 3.587 | 2.436 | 24.69 | 115.9 |
| | 32.33 | 1.765 | 24.42 | 3.587 | 223.4 | 2.337 |

## 5.3 Experiment 1: Load Fluctuation with Fixed Speed Value

The beginning conditions: reference speed value $n^*$=400 n/min; Load torque $T_L$=30 N·m. And the different loads are loaded at 0.16s, 0.24s and 0.34s for validating the control performances of PIDNN-I control system. As a contrasting reference, the control results based on the PID controller are also given out. The speed and torque simulation curves with fluctuating load are shown as Fig.7.

(1) In the stage of speed run-up , the PIDNN controller will work alone until speed error gets to 15 r/min, this moment, the integral controller is available and gives out the expected control input with PIDNN controller. From the Fig.7, it is observed that PID controller causes the steady-state error and the big modulating oscillation of speed and torque, the

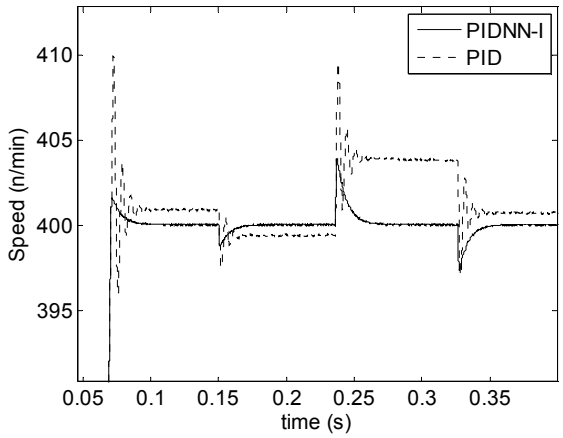
maximum overshoot exceeds about 2.5 percent. On the contrary, PIDNN-I controller is no oscillation and the steady-error is zero. The maximum overshoot is only 0.375 percent.

(2) At 0.16s, The load $T_L$ changes from 30 N•m to 60 N • m. It can be seen from the Fig.7-a that the steady-error of PID controller also changes, which changes from positive value to negative value. Compared with PID controller, PIDNN-I controller has a better performance behavior, which doesn't lead to the modulating oscillation and the steady-state error is still of zero value. .

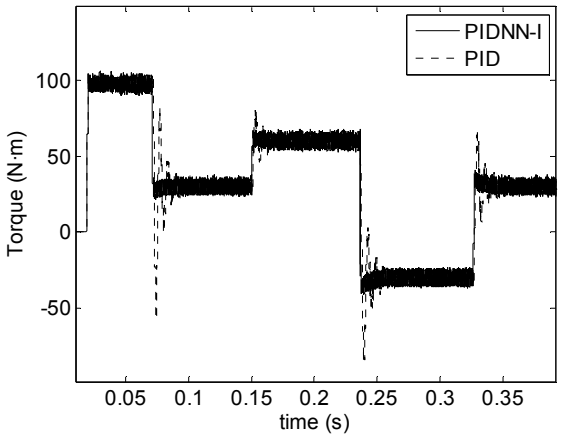
(3) At 0.24s, load suddenly decreases from 60N•m to -30N•m. The speed of motor rises fast due to the change of load. PID and PIDNN-I controller both responds quickly to make the electromagnetic torque Te* become negative value so as to restrain the changing of speed. Contrasting the responding curves, it can be conclude that the responding speed of PIDNN-I is a little fast, which makes the increasing speed error begin to decrease before getting to 4 n/min. And no modulating oscillation occurs, the steady-state error still remains at zero value. However, the situation of PID controller is different, stronger modulating oscillations of speed and torque occur and the steady-state error becomes bigger.

(4) At 0.34s, load changes from -30 N•m to 30 N•m, the respond and analysis is similar with 0.16s.

According to the results above, facing the fluctuation of load, it can be known that PIDNN-I controller has much better responding curves than PID controller; namely it decreases the overshoot and cancels the oscillation as well as the steady-state error of system, which equips the system with much stronger stability and robustness. All of these are significant for asynchronous motors in terms of enhancing the machine life and increasing the safety factor.




**(a) speed responded curves**




**(b) torque respond curves**

**Fig.7 responded curves of speed and torque with load torque fluctuating**

## 5.4 Experiment 2: Speed fluctuation with fixed torque value

The variable speed control of asynchronous motors is often required in the application of industry. In order to validate the control performance of PIDNN-I controller with changing reference speeds, some corresponding simulation researches have be done. The simulation curves are shown in Fig.8.

In order to see the respond of system clearly, the selected reference speed is 400 n/min, 350 n/min, 450 n/min and 400 n/min, respectively. And the load also changes from 30 N•m to -30 N•m with the changing of last reference speed. From the Fig.8-a and-b, it can be seen that the speed and torque oscillation of PID controller is intense with the every changing of reference speed. And the maximum overshoot of speed also increases. Compared with PID controller, facing the changing of speed, PIDNN-I controller has much stronger robustness since its dynamic respondent process is faster. The speed and torque don't have intensely modulating process. Moreover, the static tracking error is zero. All of the curves indicate again that the control performances of PIDNN-I controller is better than PID controller, which is useful for enhancing the machine life and protecting the operating safety of motor. The fig.8-c is a parts zoom of speed and torque at 350 n/min, which is beneficial to seeing the respondent curves more clearly and can give out more details about the respondent process.