Fractional-order Learning Algorithm for PID Neural Network Decoupling Control Based on Sparrow Search Algorithm

Min Wu
Nanjing University of Posts and Telecommunications

Jie Ding (dingjie@njupt.edu.cn)
Nanjing University of Posts and Telecommunications
https://orcid.org/0000-0002-9910-6280

Tingting Yuan
Nanjing University of Posts and Telecommunications

Min Xiao
Nanjing University of Posts and Telecommunications

Research Article

Keywords: Decoupling control, Fractional-order algorithm, PID neural network, Sparrow search algorithm

Posted Date: October 22nd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-1004130/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License
Fractional-order Learning Algorithm for PID Neural Network Decoupling Control Based on Sparrow Search Algorithm

Min Wu · Jie Ding · Tingting Yuan · Min Xiao

Received: date / Accepted: date

Abstract Research on control of multi-variable system with strong coupling has been a significant issue in industry. To accurately eliminate the coupling between system variables and improve the control effect, decoupling control techniques are investigated. In this paper, a decoupling control scheme based on fractional-order proportion integration differentiation neural network and sparrow search algorithm (SSA-FPIDNN) is proposed, where sparrow search
algorithm is employed to derive the optimal initial weights, preventing the weights from falling into the local optimum, while the fractional-order algorithm is used to correct its connection weights to improve control accuracy. Compared with traditional PIDNN, the proposed SSA-FPIDNN has better decoupling control performance, and the tracking time can be reduced significantly. Numerical simulation and engineering examples verified its effectiveness.

**Keywords** Decoupling control · Fractional-order algorithm · PID neural network · Sparrow search algorithm

1 Introduction

The decoupling and tracking control of multi-variable systems with strong coupling in modern industry has been a research hotspot in recent years [1–4]. Traditional decoupling controllers are designed for linear systems [5] or for weak coupling systems [6].

Neural network based decoupling controllers provide excellent performance since they are suitable for nonlinear multi-variable systems with strong coupling. Shu proposed proportion integration differentiation neural network (PIDNN) [7], which upgraded hidden layer neurons with PID characteristics so the neurons have a memory function, that turns the static neural network into a dynamic one. The PIDNN controller integrates both the advantages of neural network and PID controller in decoupling control [8,9]. However, two limiting factors in achieving higher control performance by PIDNN are random initial
weights of PIDNN and undesirable accuracy of the gradient descent method [10–12]. Carefully selected initial weights may help PIDNN find global optimum, otherwise, only local optimal results can be derived for control systems [13–15], namely, optimal initial weights are primary to improve the control performance. Shu proposed a network weight correction strategy with additional momentum that can generate a disturbance to avoid the local minimum [16], but the controller is unstable. Hao proposed a PIDNN controller based on an adaptive scaling factor, which can adaptively change the scaling factor during the training process [17]. This method improves the control performance of PIDNN, but the selection of the initial scaling factor is constrained. The gradient descent method in PIDNN is integer-order and ignores the influence of the past information when calculating the weights of the current sampling time, and leads to weak calculation accuracy [18].

In order to eliminate the coupling between system variables and improve control accuracy, this paper presents a modified PIDNN controller for nonlinear multi-variable systems with strong coupling. Sparrow search algorithm (SSA) is employed to find optimal initial weights of PIDNN, while a fractional-order weight correction algorithm is used to modify the gradient descent method and improve system control. The proposed algorithm takes the advantages of optimal search algorithm and fractional-order characteristics, which has a better performance in tracking speed, stability, and global optimum. The main contributions can be described as follows:
The SSA is introduced to PIDNN to solve the local minimum problem and explore optimal initial weights, thereby enhancing the response speed of the PIDNN.

A fractional-order algorithm is employed to modify the weights of PIDNN which is in integer-order form. Past information is included when updating the weights, and the decoupling control performance of PIDNN can be improved.

The rest of the paper is organized as follows: In Section 2, a detailed introduction of PIDNN is stated. In Section 3, the SSA is introduced to PIDNN to find optimal initial weights. In Section 4, a fractional-order algorithm is employed to modify the weights of PIDNN. In Section 5, two examples are presented, which verified the effectiveness of the proposed algorithm. Finally, some conclusions are given in Section 6.

2 PID neural network

The neural network has been proved to be a powerful control tool for multi-variable and nonlinear systems [19]. However, the structure and function of traditional neural network are difficult to relate to the dynamic indicators in the control system. Although the multi-layer forward neural network is widely used, its input and output characteristics are static, and dynamic components must be added when designing the controller. PIDNN integrates proportional, integral, and derivative functions into hidden layer neurons to have dynamic characteristics. Therefore, PIDNN is more suitable for system control [20].
Fig. 1 depicts the structure of a PIDNN [7], where the controlled object is a coupling system with $m$-input and $n$-output. The PIDNN is a forward neural network with three layers composed of several parallel sub-networks, $r_s$ ($s = 1, 2, \cdots, n$) represent the desired signal of variables, $y_s$ represent the outputs of controlled object, $v_o$ ($o = 1, 2, \cdots, m$) represent the neuron states.

![PID neural network](image)

**Fig. 1** PID neural network

The input layer of each sub-network contains two neurons, which process the desired signal and output respectively, the hidden layer contains three neurons, specifically unit proportional, integral, and differential element, the output layer carries only one unit proportional neuron to complete the synthesis and output of the entire PIDNN control law. The output of PIDNN is the input of the controlled object.

(1) Input layer
The inputs of a single layer of neurons are:

\[ X_{s1}(p) = r(p) \]  
\[ X_{s2}(p) = y(p) \]

where \( p \) is the sampling time.

The input layer of PIDNN includes \( 2n \) identical neurons, and their state functions are unit proportional functions, so the neurons’ states are equal to the inputs,

\[ v_{si}(p) = X_{si}(p) \]  

Then, the outputs of the input layer are:

\[
x_{si}(p) = \begin{cases} 
1, & v_{si}(p) > 1 \\
 v_{si}(p), & -1 \leq v_{si}(p) \leq 1 \\
-1, & v_{si}(p) < -1
\end{cases}
\]

where \( i \) is the serial number of the sub-network input layer \((i = 1, 2)\), \( s \) is the sequence number of the sub-network \((s = 1, 2, \ldots, n)\).

(2) Hidden layer

The hidden layer contains \( n \) neurons each for proportional, integral, and differential neurons, their states are updated by equation (5)–(7) respectively,

\[ v'_{s1}(p) = X'_{s1}(p) \] 
\[ v'_{s2}(p) = v'_{s2}(p - 1) + X'_{s2}(p) \] 
\[ v'_{s3}(p) = X'_{s3}(p) - X'_{s3}(p - 1) \]
The inputs of the hidden layer are:

$$X'_{sh}(p) = \sum_{i=1}^{2} \omega_{sih}(p)x_{si}(p)$$  \hspace{1cm} (8)

Then, the outputs of the hidden layer are:

$$x'_{sh}(p) = \begin{cases} 
1, & v'_{sh}(p) > 1 \\
 v'_{sh}(p), & -1 \leq v'_{sh}(p) \leq 1 \\
-1, & v'_{sh}(p) < -1 
\end{cases}$$  \hspace{1cm} (9)

where $\omega_{sih}$ is the connection weight from input layer to the hidden layer and $h$ is the number of the sub-network hidden layer ($h = 1, 2, 3$).

3) Output layer

The output layer contains $m$ neurons, and the output is an $m$-dimensional vector. The input of each output neuron is the weighted sum of the output of all neurons in the hidden layer.

$$X''_o(p) = \sum_{s=1}^{n} \sum_{h=1}^{3} \omega_{sho}(p)x'_{sh}(p)$$  \hspace{1cm} (10)

The states of the output neurons are:

$$v''_o(p) = X''_o(p)$$  \hspace{1cm} (11)

Then, the outputs of the output layer are:

$$x''_o(p) = \begin{cases} 
1, & v''_o(p) > 1 \\
v''_o(p), & -1 \leq v''_o(p) \leq 1 \\
-1, & v''_o(p) < -1 
\end{cases}$$  \hspace{1cm} (12)

The output of the output layer is the same as the output of PIDNN, that is:

$$y_o(p) = x''_o(p)$$  \hspace{1cm} (13)
where \(\omega_{sho}\) is the weight from hidden layer to output layer and \(o\) is the output layer neuron number \((o = 1, 2, \cdots, m)\).

(4) Weight adjustment algorithm

PIDNN weight adjustment adopts the gradient descent method, and its cost function is defined as:

\[
J = \sum_{o=1}^{m} E_o = \frac{1}{l} \sum_{o=1}^{m} \sum_{p=1}^{l} [r_o(p) - y_o(p)]^2
\]  

(14)

where \(l\) is the number of sampling points. Define the learning rate as \(\eta\), the connection weights of each layer can be updated by:

\[
\omega(p + 1) = \omega(p) - \eta \frac{\partial J}{\partial \omega(p)}
\]

(15)

PIDNN embeds PID rules into the neural network, which makes the neural network have dynamic characteristics so that it can deal with coupling systems. However, because of the random convergence direction of the PIDNN, a heavy computational budget will be required to find the right direction, which will slow down the convergence speed. In addition, it will increase the probability of the weights falling into a local optimum [20]. To solve this problem, SSA is introduced into PIDNN to optimize initial weights and improve the control of the PIDNN.

3 PID neural network based on sparrow search algorithm

SSA is a novel evolutionary search algorithm with the advantages of fast convergence, robustness, and sturdy global searchability. In SSA, sparrows can be
divided into producers and scroungers. Producers provide foraging directions for the rest sparrows while finding food. Therefore, producers with higher fitness can get food first and have a large search range. In the search range, every sparrow can turn into a producer if it finds food. Scroungers find food by making use of the producer. Furthermore, they may compete for the food with discoverers [21]. The algorithm flow is shown in Fig. 2.

![Flowchart](image)

**Fig. 2** The flow chart of sparrow search algorithm

The positions of the producers are updated by:
\[ P_{t+1}^{\mu,\rho} = \begin{cases} 
  P_t^{\mu,\rho} \cdot \exp \left( \frac{-\mu}{\text{iter}_{\text{max}}} \right) & \text{if } R_2 < ST \\
  P_t^{\mu,\rho} + Q \cdot L & \text{if } R_2 \geq ST 
 \end{cases} \]  

(16)

where \( t \) represents the number of iterations, \( P_{t+1}^{\mu,\rho} \) is the position of the \( \mu \)th sparrow in \( \rho \)th dimension at iteration \( t \), \( a \in (0, 1] \) and \( Q \) obeys normal distribution are numbers that are generated arbitrarily, \( \text{iter}_{\text{max}} \) is maximum number of iterations, \( R_2 \in [0, 1] \) and \( ST \in [0.5, 1.0] \) denote early warning value and safety critical value respectively, if \( R_2 < ST \), it indicates that the surrounding environment is safe, otherwise, dangers occur, and the sparrow must fly to safe places, \( L \) is a matrix with each element inside 1.

The positions of the scroungers are updated by:

\[ P_{t+1}^{\mu,\rho} = \begin{cases} 
  Q \cdot \exp \left( \frac{P_{\text{worst}}^t - P_{t+1}^{\mu,\rho}}{z} \right) & \text{if } \mu > z/2 \\
  P_{t+1}^{\mu,\rho} \cdot |P_{t+1}^{\mu,\rho} - P_{t}^{\mu,\rho}| \cdot C^\dagger \cdot L & \text{otherwise} 
 \end{cases} \]  

(17)

where \( z \) is the number of sparrows, \( P_B \) represents the best position founded by the producer, \( P_{\text{worst}} \) denotes the global worst position at iteration \( t \), \( C \) is a matrix with each entry randomly assigned 1 or \(-1\), and \( C^\dagger = C^T (CC^T)^{-1} \).

The positions of sparrows that are perceived the danger in the sparrow population are updated by:

\[ P_{t+1}^{\mu,\rho} = \begin{cases} 
  P_{\text{best}}^t + \beta \cdot |P_{t}^{\mu,\rho} - P_{\text{best}}^t| & \text{if } f_\mu > f_g \\
  P_{t}^{\mu,\rho} + G \cdot \left( \frac{|P_{t}^{\mu,\rho} - P_{\text{worst}}^t|}{(f_g - f_w) + \epsilon} \right) & \text{if } f_\mu = f_g 
 \end{cases} \]  

(18)

where \( \beta \) represents step size, \( P_{\text{best}} \) is the global best position at iteration \( t \), \( G \in [-1, 1] \) is a number that is generated arbitrarily, \( \epsilon \) denotes a small
constant, \( f_\mu \) represents the fitness of the sparrow, \( f_g \) and \( f_w \) represent the global optimal and worst fitness at iteration \( t \), respectively.

In SSA, the position of each sparrow represents the candidate solution of the optimization system, set the number of iterations to \( \text{length1} \) and define the fitness function as:

\[
F = e^{0.01 \cdot o} \cdot \left( \sum_{o=1}^{\text{length1}} |r_o - y_o| \right), o = 1, 2, \ldots, \text{length1} \tag{19}
\]

SSA takes efforts in finding the positions with the best fitness, which are the optimal initial weights of PIDNN.

SSA-PIDNN conquers the adverse effects caused by the random initial weights. However, the gradient descent method in (15) ignores the impact of the past moment weights on the current calculation. To improve the control accuracy and tracking response, a fractional-order weight correction technique is introduced to cope with the problem.

4 SSA-PIDNN by the fractional-order learning algorithm

A valuable property of fractional calculus is that it considers the influence of past information on the current variables, so more accurate results can be obtained. One of the most widely used fractional calculus definitions is the discrete fractional differential operator proposed by Grünwald-Letnikov:

\[
D^\alpha f(t) = \lim_{p_1 \to 0} \frac{1}{p_1^\alpha} \sum_{q=0}^{\infty} (-1)^q \binom{\alpha}{q} f(t - qp_1) \tag{20}
\]
where $D$ is a differential operator, $p_1$ is the sampling time, $\alpha$ is the fractional order, $f(\cdot)$ is the function of time $t$, \( \binom{\alpha}{q} \) is Newton’s binomial:

\[
\binom{\alpha}{q} = \frac{\Gamma(\alpha+1)}{\Gamma(q+1)\Gamma(\alpha-q+1)}
\]

where $\Gamma(\cdot)$ is Gamma function:

\[
\Gamma(\alpha) = \int_0^\infty e^{-\delta^\alpha-1} \, d\delta
\]

Let $c_j^\alpha = (-1)^j \binom{\alpha}{q}$, which is the binomial coefficient, and the recursive formula is denoted by:

\[
c_j^\alpha = \begin{cases} 
1 & j = 0 \\
(1 - \frac{\alpha+1}{j}) c_{j-1}^\alpha & j = 1, 2, 3, \ldots 
\end{cases}
\]

From (20), the weights in (15) can be modified by fractional order gradient descent algorithm as follows:

\[
\Delta \omega = \omega(p+1) - \omega(p) = -\eta \frac{\partial J}{\partial \omega(p)} p_1^{\alpha} - \sum_{j=0}^{p} c_j^\alpha \omega(p)
\]

The structure of the fractional-order PIDNN is the same as that of the integer-order PIDNN, but the adjustment of the weights of FPIDNN adopts the fractional-order gradient descent algorithm. The following calculation methods determine the weights between each layer:

(1) Hidden layer to output layer
The connection weights from the hidden layer to the output layer are updated by:

\[
\omega_{sho}(p + 1) = \omega_{sho}(p) - \eta_1 \frac{\partial J}{\partial \omega_{sho}(p)} p^s_1 - \sum_{j=0}^{p} c^s_j \omega_{sho}(p) \quad (25)
\]

where \(\eta_1\) is the learning rate from the hidden layer to the output layer, and

\[
\frac{\partial J}{\partial \omega_{sho}(p)} = -\frac{2}{l} \sum_{o=1}^{m} \sum_{p=1}^{l} [r_o(p) - y_o(p)] x'_{sh}(p) \quad (26)
\]

(2) Input layer to hidden layer

The connection weights from the input layer to the hidden layer are updated by:

\[
\omega_{sih}(p + 1) = \omega_{sih}(p) - \eta_2 \frac{\partial J}{\partial \omega_{sih}(p)} p^s_1 - \sum_{j=0}^{p} c^s_j \omega_{sih}(p) \quad (27)
\]

where \(\eta_2\) is the learning rate from the input layer to the hidden layer, and

\[
\frac{\partial J}{\partial \omega_{sih}(p)} = -\frac{2}{l} \sum_{o=1}^{m} \sum_{p=1}^{l} [r_o(p) - y_o(p)] \omega_{sho} \frac{v'_{sh}(p) - v'_{sh}(p-1)}{X_{sh}(p) - X'_{sh}(p-1)} x_{si}(p) \quad (28)
\]

SSA-FPIDNN uses the optimal position vector obtained by SSA as the initial weights of PIDNN and modifies the weights by the fractional-order gradation descent method. The specific calculation steps are described by Algorithm 1, where \(PD\) and \(SD\) are the numbers of producers and the sparrows that are aware of the danger, respectively.
Algorithm 1 FPIDNN based on SSA

1: Initialization: $PD = 25$, $SD = 10$, $ST = 0.7$, $z = 50$, $l = 200$, $length_1 = 200$

2: function $SSA(SD, PD, R, z, length_1, t)$

3:   while $(t < length_1)$ do

4:       Find the current optimal position and worst position

5:       for $\mu = 1 : PD$ do

6:           Update producer’s position by (16)

7:       for $\mu = (PD + 1) : z$ do

8:           Update scrounger’s position by (17)

9:       for $\mu = 1 : SD$ do

10:          Update sparrow’s position by (18)

11:         $t = t + 1$

12:        Update $P_{sset}$

13: Output the best individual position vector $P_{sset}$, that is the optimal initial weights

14: for $p = 1 : l$ do

15:     Calculate $x_{si}(p)$ by (4)

16:     Update $\omega_{sh}(p)$ by (27)

17:     Compute $x'_{sh}(p)$ by (9)

18:     Form $\omega_{sho}(p)$ according to (25)

19:     Estimate $x''_{o}(p)$ from (12)

5 Examples

Example 1: Consider a coupling system with 3-input and 3-output as the controlled object of the PIDNN, which is as follows:

$$
\begin{align*}
        y_1(p) &= 0.4y_1(p-1) + v_1(p-1)/ [1 + v_1^2(p-1)] + \\
              & 0.2v_1^3(p-1) + 0.5v_2(p-1) + 0.3y_2(p-1) \\

        y_2(p) &= 0.2y_2(p-1) + v_2(p-1)/ [1 + v_2^2(p-1)] + \\
              & 0.4v_2^3(p-1) + 0.2v_1(p-1) + 0.3y_3(p-1) \\

        y_3(p) &= 0.3y_3(p-1) + v_3(p-1)/ [1 + v_3^2(p-1)] + \\
              & 0.4v_3^3(p-1) + 0.4v_2(p-1) + 0.3y_1(p-1) \\
\end{align*}
$$

(29)
where \( \mathbf{y} = [y_1, y_2, y_3]^T \) and \( \mathbf{v} = [v_1, v_2, v_3]^T \) are output and input vectors of the system, respectively. The desired signal is set as \( \mathbf{r} = [0.7, 0.4, 0.6]^T \), \( \eta_1 = 0.03 \), \( \eta_2 = 0.006 \), the fractional order \( \alpha = 0.998 \). Comparing SSA-FPIDNN with PIDNN and SSA-PIDNN respectively, the output and errors of various methods are shown in Figs. 3–5.

![Fig. 3 PIDNN control performance for numerical system](image)

![Fig. 4 SSA-PIDNN control performance for numerical system](image)

**Example 2**: Consider a complex greenhouse system with strong coupling. The key factors of the greenhouse system control are humidity and temperature, which are mainly related to heating, humidification, and ventilation. Therefore,
the control object of the greenhouse system can be regarded as a system with 3-input and 2-output. In this paper, the simplified greenhouse system model is the same as that in [22]:

\[
\begin{align*}
    y_1(p) &= 0.879y_1(p - 1) + 0.202v_1(p) - 0.371v_2(p) + 0.25v_3(p) \\
    y_2(p) &= 0.4y_2(p - 1) + 0.25v_1(p) + 0.5v_2(p) - 0.25v_3(p)[y_2(p - 1) - 0.45]
\end{align*}
\]  

(30)

where \( y = [y_1, y_2]^T \) and \( v = [v_1, v_2, v_3]^T \) are output and input vectors of the system, respectively. The desired signal is set as \( r = [0.5, 0.7]^T \), \( \eta_1 = 0.006 \), \( \eta_2 = 0.002 \), the fractional order \( \alpha = 0.9899 \). Comparing SSA-FPIDNN with PIDNN and SSA-PIDNN respectively, the output and errors of various methods are shown in Figs. 6–8.

From Table 1 and Fig. 3-8, one can find that:

1. As shown in Table 1, the algorithms can perform decoupling control effectively, but the proposed SSA-FPIDNN can track the target outputs in the shortest time.
(2) The traditional PIDNN in Fig. 3 and Fig. 6 have the decoupling capability.

However, due to the local optimization and integer-order algorithm accu-
Table 1 The minimum tracking time of different methods

| Example | Example 1 | Example 2 |
|---------|-----------|-----------|
| Method  | PIDNN SSA-PIDNN SSA-FPIDNN | PIDNN SSA-PIDNN SSA-FPIDNN |
| Tracking time | $T_y(s)$ | $T_y(s)$ |
| $y_1$ | 0.097 0.017 0.010 | 5.850 1.950 1.050 |
| $y_2$ | 0.044 0.016 0.011 | 2.550 0.900 0.400 |
| $y_3$ | 0.053 0.016 0.009 | |

racy problems, the output has a certain error to the tracking signal, and the response is relatively slow.

(3) The SSA-PIDNN in Fig. 4 and Fig. 7 have a faster response speed than PIDNN. It solves the problem of PIDNN initial weights selection and achieves better tracking performance.

(4) Compared with PIDNN and SSA-PIDNN, the SSA-FPIDNN in Fig. 5 and Fig. 8 combines the advantages of SSA and fractional-order algorithm, which has a strong decoupling control capability. So the output of the controlled object can track the ideal signal quickly with a small steady-state error.

6 Conclusions

In this study, an SSA-FPIDNN is proposed for decoupling control of multivariable and strong coupling systems. It exploits SSA to obtain the global optimal initial weights of FPIDNN, which can solve difficulties caused by random initial weights and then decouples the variables with the help of FPIDNN.
The proposed method can implement more reliable operations than PIDNN and SSA-PIDNN because the fractional-order gradient descent algorithm has a long memory function. Combined with SSA in the weight adjustment process can further enhance the control performance. This work provides a reliable method for the control of multi-variable systems with strong coupling in the industry.

Conflict of interest

The authors declare that they have no conflict of interest.

Data availability statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. Shen, G. H., Xia, Y. Q., Zhang, J. H., Cui, B.: Adaptive Fixed-time Trajectory Tracking Control for Mars Entry Vehicle. Nonlinear Dynamics. 102, 2687-2698 (2020)
2. Chen, Y. M., He, Y. L., Zhou, M. F.: Decentralized PID Neural Network Control for a Quadrotor Helicopter Subjected to Wind Disturbance. Journal of Central South University. 22, 168-179 (2015)
3. Zhang, X., Liu, J., Gao, Q., Ju, Z.: Adaptive Robust Decoupling Control of Multi-arm Space Robots Using Time-delay Estimation Technique. Nonlinear Dynamics. 100, 2449-2467 (2020)
4. Xue, X. C., Xiang, W. G., Lu, J. H.: Modeling and Control of Industrial Fischer Tropsch Synthesis Slurry Reactor Using Artificial Neural Networks. Journal of Chemical Engineering of Japan. 47, 887-892 (2014)
5. Hao, J., Zhang, G. S., Liu, W. Q., Zheng, Y. Q.: Data-Driven Tracking Control Based on LM and PID Neural Network with Relay Feedback for Discrete Nonlinear Systems. IEEE Transactions on Industrial Electronics. 99, 1-1 (2020)
6. Han, J. Q.: From Pill Technique to Active Disturbances Rejective Control Technique. Control Engineering of China. 9, 13-18 (2002)
7. Shu, H. L., Pi, Y.: PID Neural Networks for Time-Delay Systems. Computers and Chemical Engineering. 24, 859-862 (2000)
8. Lee, T., Kim, I., Baek, Y. S.: Design of a 2DoF Ankle Exoskeleton with a Polycentric Structure and a Bi-Directional Tendon-Driven Actuator Controlled Using a PID Neural Network. Actuators. 10, 9-26 (2021)
9. Zhang, H., Qiu, Y., Liang, H., Li, Y., Yuan, X.: Research on Remote Intelligent Control Technology of Throttling and Back Pressure in Managed Pressure Drilling. Petroleum. 7, 222-229 (2020)
10. Zhang, X., Xu, T., Zhao, L., Fan, H., Zang, J.: Research on Flatness Intelligent Control Via GA-PIDNN. Journal of Intelligent Manufacturing. 26, 359-367 (2015)
11. Zhang, P. Y., Wu, X. Q., Yuan, H. Q.: Research on Decoupling Control by MPSO-PID Neural Network in MIMO System. Industrial Control Computer 25, 45-47 (2012)
12. Zhou, J. X., Zhao, L. I., Song, D. L., Shi, L.: PID Neural Network Decoupling Control Based on Improved Particle Swarm Optimization. Machine Tool and Hydraulica. 46, 74-79 (2018)
13. Lee, C., Chen, R.: Optimal Self-Tuning PID Controller Based on Low Power Consumption for a Server Fan Cooling System. Sensors. 15, 11685-11700 (2015)
14. Li, C. D., Zhang, Q., Liang, Y.: GA-Based PID Neural Network Control For Magnetic Bearing Systems. Chinese Journal of Mechanical Engineering. 2, 56-59 (2007)
15. Feng, D. Q., Chao-Yang, M. A., Liu, Y. H.: Nonlinear Control System of PID Neural Network Based on Artificial Fish Swarm Algorithm. Computer Simulation. 29, 247-269 (2012)
16. Shu, H., Xu, Y. K.: Application of Additional Momentum in PID Neural Network. IEEE International Conference on Information Science and Technology. 172-175 (2014)

17. Hao, J., Zhang, G. S.: Data-Driven Tracking Control for a Class of Unknown Non-linear Time-Varying Systems Using Improved PID Neural Network and Cohen-Coon Approach. IEEE International Data Driven Control and Learning Systems Conference. 619-624 (2021)

18. Zou, J. F., Zhang, R. D., Zhang, F. R., Xue, A. K.: Fractional Order MPC Design Using Improved State Space Model. IFAC - Papers Online. 50, 7535-7504 (2017)

19. Gao, H., Li, X., Gao, C., Wu J.: Neural Network Supervision Control Strategy for Inverted Pendulum Tracking Control. Discrete Dynamics in Nature and Society. 6, 1-14 (2021)

20. Shu, H. L.: PID Neural Network and Its Control System, National Defense Industry Press, China (2006)

21. Xue, J. K., Bo, S.: A Novel Swarm Intelligence Optimization Approach: Sparrow Search Algorithm. Systems Science and Control Engineering, 8, 22-34 (2020)

22. Wang, P., Gong, R. K.: Decoupling Control of Greenhouse System Based on Particle Swarm Optimization PIDNN. Jiangsu Agricultural Sciences. 46, 241-244 (2018)