

# Research on load simulator control strategy based on BP neural network and PID method

Yong Zhou, Yubo Zhang\*, and Tianhao Yang

School of Aeronautics, Northwestern Polytechnical University, Xi'an, China

**Abstract.** In the research of load simulator control method, PID control is the most widely used control strategy, but PID controller's three parameters is difficult to set. This paper proposes a BP neural network feedforward PID controller system which uses BP neural network for setting these parameters, and in order to make the network learning speed up the convergence speed and not fall into local minimum, the adaptive vector method is adopted to improve the algorithm. The simulation and experimental results show that this method is good at avoiding the primeval shock and the sine tracking performance of the system has also been improved.

## 1 Introduction

The flight control system is an important system of modern aircraft, and its performance plays a decisive role in the performance and safety of the aircraft. In actual engineering, the load simulator is often used to test its performance. With the rapid development of aerospace technology, the performance requirements of flight control systems are getting higher and higher in the industry. How to improve the performance of load simulators become a difficult and important point in this field.

At present, researches have been done much on this issue at home and abroad. In general, it can be divided into two aspects: improving hardware or optimizing control strategies. For example, Jacazio G and Balossini G propose a spring correction method [1] in which the loading part and the measured part are connected by elastic elements to buffer the influence of the positional disturbance of the measured part. Baghestan K propose a robust control method [2], that is selecting an appropriate correction network to reduce the system sensitivity.

Due to the high cost of improving hardware, this paper proposes a feedforward PID controller system which uses BP neural network for setting parameters and adopt an adaptive vector method to improve the algorithm.

The rest of this paper is organized as follows: The feedforward PID control method is introduced in Section 2, Section 3 provides a brief introduction of back-propagation neural network (BPNN) and proposed a BPNN-based feedforward PID control method. In Section 4 and Section 5, simulation and experiment show that this method has good performance.

## 2 Feedforward PID controller

### 2.1. PID and extra force

The PID controller consists of three correction links: proportional link P, integral link I, and differential link D. The roles of each correction link are as follows:(1)P: The output of the regulator is proportional to the input deviation, as long as the deviation occurs, it can play a proportional adjustment effect in time.(2)I: It can eliminate the steady state error of the system and improve the control accuracy.(3)D: It can predict the trend of error changes in advance and overcome the adverse effects caused by the error changes in advance.

The system under test is connected to the loading system through a rigid structure. When the system under test moves actively, the loading system will move together and generate interference forces, which are called excess forces [3].

When only the traditional PID control is used, the output of the system will have a certain degree of lag relative to the input, and the excess force generated during this period cannot be eliminated in time, which has a great influence on the system performance.

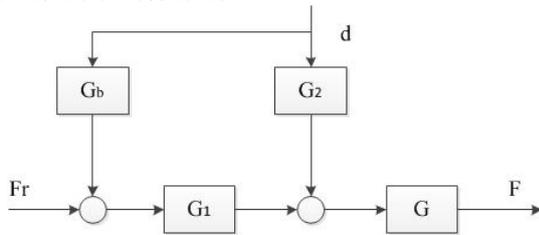
### 2.2. Feedforward controller

The feedforward compensation link is a controller designed to overcome excess forces. The role of the feedforward controller is to compensate the system before the active motion of the steering gear interferes, thus overcoming

the effects of excess force. The control schematic diagram is shown in Fig. 1. In the figure,  $G_b$  is the transfer function of the feedback link,  $G$ ,  $G_1$ ,  $G_2$  are the

\* Corresponding author: 1030869628@qq.com

transfer function of the load simulator,  $F_r$  is the target command signal,  $F$  is the output feedback force signal, and  $d$  is the excess force.



**Fig. 1.** Feedforward compensation controller schematic

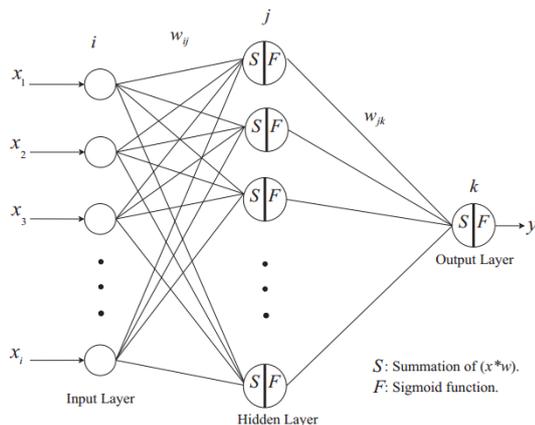
The control effect of the feedforward compensator is related to the input signal. The specific analysis is as follows:

The input of the feedforward controller selects the actual displacement of the servo to be tested: because the correction of the phase and gain of the system output needs to be based on the actual motion of the steering gear, and only the actual displacement of the steering gear is collected by the grating scale. The operation is convenient and the compensation effect is good. Therefore, the actual displacement of the steering gear is used as the input of the feedforward compensator.

### 3 PID controller based on BPNN

#### 3.1 Back-propagation neural network (BPNN)

The BP neural network (BPNN) is also known as the backpropagation neural network, which was proposed by the research team headed by Rumelhart and McClland in 1986. BP neural network is a multi-layer feedforward network trained according to the error back propagation algorithm. It is one of the most widely used neural network models in practical applications. Fig. 2 illustrates the BP architecture in details. It consists of an input layer, one or more hidden layer, and an output layer. Layers are connected sequentially starting from the input layer through the hidden layers to the output layer. Where the connections between layers contain weights and each layer includes one or more neurons [4].



**Fig. 2.** Back-propagation architecture.

#### 3.2 PID control algorithm based on BPNN

The formula of the traditional incremental digital PID control algorithm is as follows.

$$\begin{aligned} \Delta u(k) &= k_p[e(k) - e(k-1)] + k_i e(k) \\ &\quad + k_D[e(k) - 2e(k-1) + e(k-2)] \quad (1) \\ &= k_p \Delta e(k) + k_i e(k) \\ &\quad + k_D[\Delta e(k) - \Delta e(k-1)] \end{aligned}$$

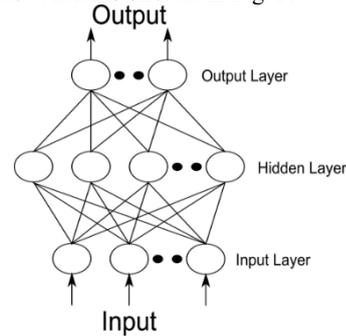
Where:

$k_p$  means proportional coefficient.

$k_i$  means integral coefficient.

$k_D$  means differential coefficient.

And taking a three-layer BP neural network structure as an example which is shown in Fig. 3.



**Fig. 3.** Three-layer BP neural network.

##### 3.2.1 Forward algorithm

The title is set in bold 16-point Arial, justified. The first letter of the title should be capitalised with the rest in lower case.

The input to the input layer is:

$$O_j = x(j) \quad (j = 1, 2, \dots, M) \quad (2)$$

In the formula,  $M$  is determined by the complexity of the controlled system. Here  $M$  is assumed to be 4.

$$x(2) = e(k) \quad (3)$$

$$x(3) = e(k) - 2e(k-1) + e(k-2) \quad (4)$$

$$x(4) = u(k-1) \quad (5)$$

The input and output expressions of the hidden layer neurons of the network are as follows:

$$net_i^{(2)}(k) = \sum_{j=0}^4 w_{ij}^{(2)} O_j^{(1)} \quad (6)$$

$$O_i^{(2)}(k) = f(net_i^{(2)}(k)) \quad (i = 1, 2, 3, 4, 5) \quad (7)$$

Where  $w_{ij}^{(2)}$  is the connection weight between the hidden layer and the input layer; the superscripts (1), (2), and (3) represent the input layer, the hidden layer and the output layer.

Suppose the activation function  $f$  is a Sigmoid function, and the expression is as follows:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (8)$$

The input and output expressions of the output layer of the neuron network are as follows:

$$net_i^{(3)}(k) = \sum_{i=0}^5 w_{ii}^{(3)} O_i^{(2)}(k) \quad (9)$$

$$O_i^{(3)}(k) = g(net_i^{(3)}(k)) \quad (10)$$

The output values of the output layer correspond to the scale factor, integral coefficient, and differential coefficient, respectively. The formula is as follows:

$$O_1^{(3)}(k) = K_p \quad (11)$$

$$O_2^{(3)}(k) = K_I \quad (12)$$

$$O_3^{(3)}(k) = K_D \quad (13)$$

It is also assumed that the activation function of the output layer neuron is a sigmoid function.

$$g(x) = \frac{1}{1 + e^{-x}} \quad (14)$$

### 3.2.2 Backward algorithm

Suppose the performance indicator function is:

$$E(k) = \frac{1}{2} (r(k) - y(k))^2 \quad (15)$$

The gradient descent method is used to correct the weight of the neural network, that is, the search is adjusted along the direction in which E(k) has the largest negative gradient of the weight, and a global minimum inertia term that can quickly converge the search process is attached.

$$\Delta w_{ii}^{(3)}(k) = -\mu \frac{\partial E(k)}{\partial w_{ii}^{(3)}} + \alpha w_{ii}^{(3)}(k-1) \quad (16)$$

Where  $\mu$  is the learning rate and  $\alpha$  is the inertia coefficient.

$$\begin{aligned} \frac{\partial E(k)}{\partial w_{ii}^{(3)}} &= \frac{\partial E(k)}{\partial y(k)} * \frac{\partial y(k)}{\partial \Delta u(k)} * \frac{\partial \Delta u(k)}{\partial O_i^{(3)}(k)} \\ &* \frac{\partial O_i^{(3)}(k)}{\partial net_i^{(3)}(k)} * \frac{\partial net_i^{(3)}(k)}{\partial w_{ii}^{(3)}} \end{aligned} \quad (17)$$

Where:

$$\frac{\partial E(k)}{\partial y(k)} = e(k) \quad (18)$$

$$\text{sgn}(k) = \frac{\partial y(k)}{\partial \Delta u(k)} \quad (19)$$

$$\frac{\partial \Delta u(k)}{\partial O_1^{(3)}(k)} = \Delta e(k) \quad (20)$$

$$\frac{\partial \Delta u(k)}{\partial O_2^{(3)}(k)} = e(k) \quad (21)$$

$$\frac{\partial \Delta u(k)}{\partial O_3^{(3)}(k)} = \Delta e(k) - \Delta e(k-1) \quad (22)$$

$$\frac{\partial O_i^{(3)}(k)}{\partial net_i^{(3)}(k)} = g'(net_i^{(3)}(k)) \quad (23)$$

$$\frac{\partial net_i^{(3)}(k)}{\partial w_{ii}^{(3)}} = O_i^{(2)}(k) \quad (24)$$

After substituting Equation (40~46) into Equation 39, the connection weights of the output layer and the hidden layer can be obtained as follows:

$$\Delta w_{ii}^{(3)}(k) = -\mu \varphi_i^{(3)} O_i^{(2)}(k) \quad (25)$$

Where:

$$\varphi_i^{(3)} = e(k) \text{sgn}(k) \frac{\partial \Delta u(k)}{\partial O_i^{(3)}(k)} g'(net_i^{(3)}(k)) \quad (26)$$

Similarly, the connection weight between the hidden layer and the input layer can be derived as follows:

$$\Delta w_{ii}^{(2)}(k) = \mu \varphi_i^{(2)} O_i^{(2)}(k) \quad (27)$$

Where:

$$\varphi_i^{(2)} = f'(net_i^{(2)}(k)) \sum_{l=1}^3 \varphi_l^{(3)} w_{li}^{(3)}(k) \quad (28)$$

The specific steps of the BP neural network PID controller algorithm are as follows:

(1) Determine the structure of the BP neural network, that is, determine the number of neurons in the input layer and the hidden layer, find the appropriate initial weight, and choose the learning rate and the inertia coefficient.

(2) The input value  $r(t)$  and the output value  $y(t)$  of the system at time  $k$  are obtained by sampling, and the error at the moment is calculated as  $e(k)=r(t)-y(t)$ .

(3) According to the forward algorithm, the input and output of each layer of neurons are obtained. At this time, the output values of the output layer correspond to the parameters of the PID controller: proportional coefficient, integral coefficient, differential coefficient

(4) Adjust the connection weight according to the gradient descent method, and realize the adaptive adjustment of the three parameters of the PID controller through neural network self-learning.

(5) Set  $k=k+1$  and return to step (2)

### 3.3 Controller optimization

The choice of the learning rate has a great impact on the system. If the learning rate is very large, it will cause system fluctuations and even the system will collapse; but if the learning rate is small, the training time will be extended. In the actual design, the fixed learning rate does not satisfy the whole learning process very well. In order to make the system speed up the convergence without falling into the local minimum, the adaptive learning rate method is used here, which means the rate can be automatically adjusted according to the change in the error value throughout the learning process.

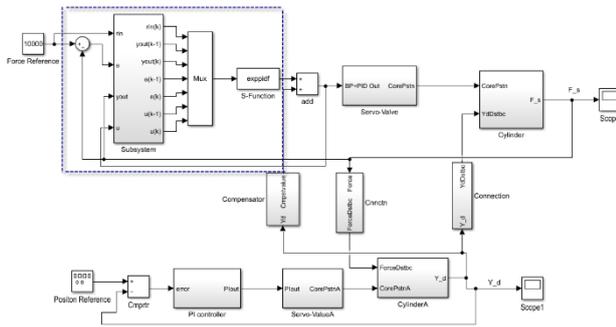
The algorithm of adaptive learning rate is as follows:

$$w^{k+1} = w^k - \mu g(w^k) \quad (29)$$

Where  $w^k$  is the training weight at time  $k$ , and  $g(w^k)$  is the gradient vector.

## 4 Controller design and simulation

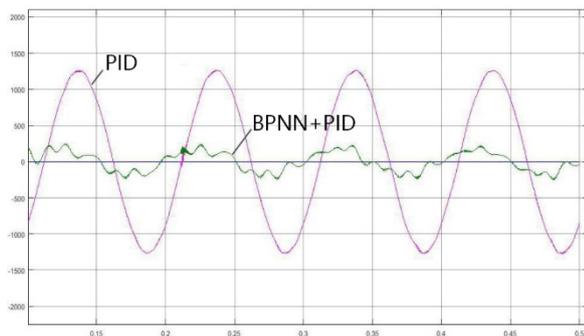
The block diagram of the load simulator control system of the feedforward PID controller based on BP neural network is established in Matlab/simulink, as shown in Fig. 4. The part in the dotted line is the BP neural network PID controller.



**Fig. 4.** Three-layer BP neural network.

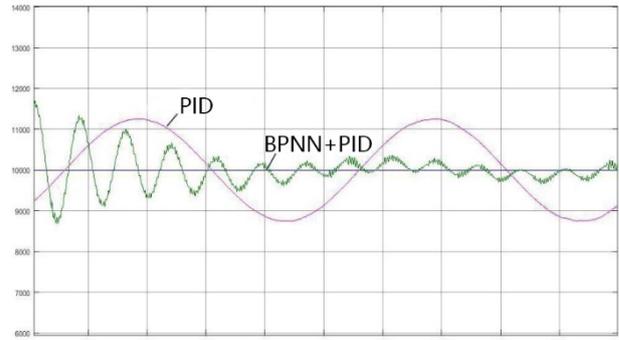
The system is affected by the extra force at work, and good control strategy can suppress the excess force. Therefore, the strength of the control strategy is measured by the amount of excess force received by the system during the test.

Under no-load conditions, the system is sinusoidal with an amplitude of 2 mm and a frequency of 10 Hz, the actual loading force is shown in Fig. 5. Compared with the traditional PID controller, the BP neural network feedforward PID controller reduced the excess force from 1250N to 230N, which means the excess force was reduced by 82% based on the traditional PID controller.



**Fig. 5.** 0N,2mm,10Hz loading force.

At this point, a constant load of 10kN is added to the system. The actual loading force curve is shown in Fig. 6. Obviously, the loading force of the BP neural network feedforward PID controller tends to be more predictive. At 0.18s, the loading force becomes stable, and the error between it and the load command is 230N, which means that the error rate is only 2.3%.

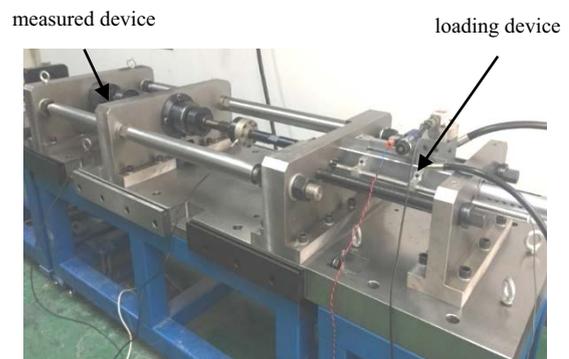


**Fig. 6.** 10kN,2mm,10Hz loading force.

The above simulation comparison test shows that BP neural network feedforward PID control strategy does have excellent control effects.

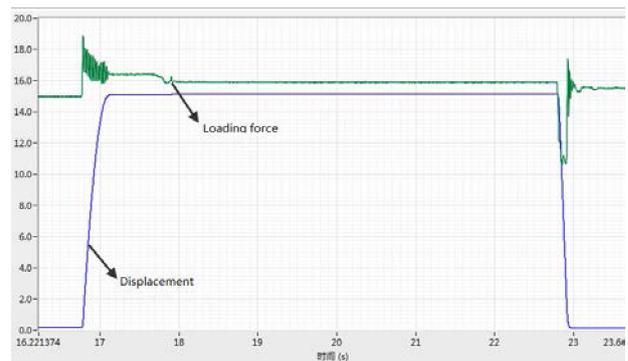
## 5 Experiments

The actual test bench is shown in Fig.7. The right side is the loading device and the left side is the measured device. During the test, the controller parameters are input to the controller through the input command. After being processed, the device under test is driven to move actively, and the hydraulic power is supplied by controlling the servo valve opening to achieve loading.

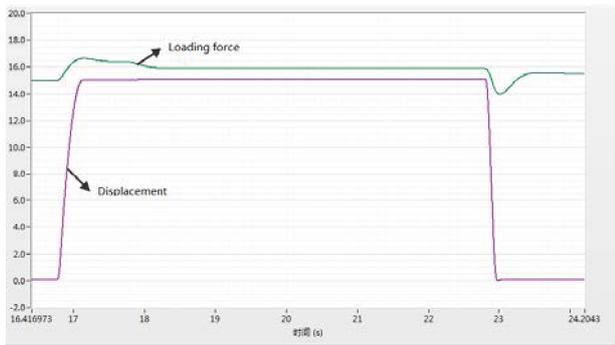


**Fig. 7.** Test bench

The displacement input is a 15mm step command, and the load command is 16kN. The test results are shown in Fig. 8 and Fig. 9.



**Fig. 8.** Traditional PID control method



**Fig. 9.** BPNN feedforward PID control method

It can be seen from the above Fig. 8 and Fig. 9 that the loading system using the traditional PID control method will generate excess force of up to 5.5kN, while the loading system using the BP neural network feedforward PID control method will generate only 2kN of excess power, which is 63.6% lower than the traditional PID method. It can be seen from the above test results that under the action of constant force, the feedforward PID control method based on BP neural network can effectively suppress the extra force.

## 6 Conclusions

The load simulator is mainly used to test the flight control system, but it is often affected by excess force during operation. Traditional PID control is the most widely used control method in the industry, but its suppression effect on excess force is poor.

In this study, a BP neural network feedforward PID control method has been developed. This method adds feedforward control based on PID control and uses BP

neural network to automatically find three parameters of PID. Simulation and experiments show that this method has better control effect than traditional PID control, and can suppress the excess force generated during the experiment. All in all, this method has a strong engineering application value.

This research is supported by “the Fundamental Research Funds for the Central Universities”, No. 31020190504003 and No. 3102016ZY002.

## References

1. Jacazio G, Balossini G. Real-time loading actuator control for an advanced aerospace test rig. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 221(2): 199-210 (2007)
2. Baghestan K, Rezaei S M, Talebi H A, Zareinejad M. Robust force control in a novel electro-hydraulic structure using polytopic uncertainty representation. *Isa Trans*, 53(6):1873-1880 (2014)
3. Han S , Jiao Z , Yao J , et al. Compound Velocity Synchronizing Control Strategy for Electro-Hydraulic Load Simulator and Its Engineering Application. *Journal of Dynamic Systems, Measurement, and Control*, 136(5):051002 (2014)
4. Hameed A A, Karlik B, Salman M S. Back-propagation algorithm with variable adaptive momentum. *Knowledge-Based Systems*, 114:79-87 (2016)