A research about the predictive control of dynamic feedforward neural network based on particle swarm optimization

Lizheng Liu[1, 2*], Fangai Liu[2], Feng Yang[1]

[1]*Shandong University of Finance and Economics, Jinan, Shandong Province, China*

[2]*Shandong Normal University, Jinan, Shandong Province, China*

Received 1 July 2014, www.cmnt.lv

Abstract

The paper proposes the Dynamic Feedforward Neural Network based on Hidden Particle Swarm Optimization (HPSO-DFNN) to deal with the model predicative control problem of unknown nonlinear delay systems. It realizes quick, precise system modelling for controlled objects. Besides, the Smith predictive double controllers are designed to separate fixed set point control from external disturbance. The DFNN based on large-scale PSO is treated as an identifier and a predictor for the complex controlled objects with the purpose of increasing the robustness of the control system. Furthermore, aiming at the problem of constrained multi-input-multi-output (MIMO) model predictive control, rolling optimization is conducted to obtain controlled quantity through the PSO algorithm. After that, a combined neural network structure is put forward and applied to system modelling. Finally, the paper uses the typical nonlinear model to verify its effectiveness.

Keywords: nonlinear delay systems, dynamic feedforward neural network, particle swarm optimization

1 Introduction

Neural networks (NNs) are divided into two types in terms of network structure: recurrent neural network and feed forward neural network (FFNN). Compared with the former, the latter has advantages such as simple structure, fast calculation and distinctive layers. However, FFNN falls under the category of static networks. Moreover, it has poor generalization ability in dynamic systems. Among all learning algorithms, the paper chooses the general gradient descent algorithm for repeated reverse derivation. But it is easy to fall into local minima. Moreover, the adjustment formulas of parameters differ from each other [1]. A good deal of formula derivation is thus indispensable. Obviously, it requires excellent mathematics. The birth of PSO provides access to the problem. Nonetheless, the simple combination of particle swarms and NNs does not suffice to improve accuracy effectively. This is because the PSO has a "premature" problem. In the neural network training process, it often abandons searching ahead of schedule. Moreover, the ability of continuous approximation plays a more important role during the latter part of the process of NN identification. Therefore, it is of great significance to design a simple, feasible method, which can improve the efficiency of NN modelling with higher calculation speed and accuracy [2].

In addition, there is usually a lot of interference in practical control systems. The Smith predictive double controllers are designed to distinguish fixed set point control from disturbance control and enhance the robustness of the entire system. As for more complicated controlled objects, the FT-HPSO algorithm is chosen in the control system to optimize FFNN parameters and solve the large-scale optimization problem. The FT-HPSO-DFNN, as a predictor and an identifier, helps to simulate the predicted output value and the current output value respectively of the controlled object.

The above research is generally limited to unrestrained single/multiple input and single output control systems. In the paper, FT-HPSO-DFNN is used for unknown controlled object modelling, which conforms to the actual situation better and helps solve the constrained MIMO problem. Furthermore, rolling optimization is carried out and the controlled quantity can be figured out. Then, the paper puts forward the model predictive control structure based on combinatorial optimization. The effectiveness of the structure is then verified through a simulation experiment on the predictive control of the multivariable nonlinear controlled objects.

2 PSO algorithm design of neural network training

BP neural network technology is a mature neural network approach, which has been applied in various fields. However, there still exit some corresponding shortages, which mainly include that: easy to fall into local minima; unable to accurately determine the number of hidden layer nodes of the network; slow convergence and so on. In recent years, many scholars have proposed a variety of methods to solve the above problems while others dedicated to using other optimized algorithm to train

* ***Corresponding author*** e-mail: 20057789@sdufe.edu.cn

neural network. But in the existing algorithms, the network's hidden layer nodes are still difficult to be determined. Therefore, this paper attempts to propose a HPSO-NN with hidden layer nodes, aiming to set the number of nodes in hidden layer in BP network as an important optimization goal for particle swarm algorithm, and set the numbers of the hidden nodes and the each weight and threshold value of BP network jointly as an important optimization goal for particle swarm algorithm [3].

As a simple, effective stochastic search algorithm, PSO algorithm could be used to optimize the neural network. Although research in this regard is still in its infant stage, some research results have demonstrated that the PSO algorithm has a great potential in terms of optimization of neural networks.

The utilization of PSO algorithm as the study algorithm of neural network mainly due to fact that the search of PSO algorithm does not rely on gradient information and it simply requires the feasible solution of the function under the constraints as well as the ability of global search. Using the PSO algorithm could conduct a global research on the weight and threshold value of neural network as well as the hidden layer nodes until it meets the demand of accuracy [4]. For PSO algorithm is based on population, and it can get many extreme values from different nodes, this algorithm could not be easily to fall into local optimal solution, which can solve the problems of BP neural network effectively and improve the generalization ability of neural networks.

2.1 THE DESCRIPTION OF THE SOLVED PROBLEMS

Using PSO algorithm to train the neural network, for a given BP neural network shown in Figure 1, the parameters need to be optimized are: the number of the hidden layer nodes in network, the connection weights w_{ji} from the input layer, the threshold value b_{1j} in hidden layer, and the threshold value b_{2k} in output layer. For these parameters, using the corresponding encoding method to encode them could create a single particle, which is the vector space solution to solve the problem. The position of each particle is the corresponding solution, and its speed is the basis of the adjustment of the position. Many particles could form a particle groups. The iterative process of the algorithm requires a fitness function. By calculating the degree of adaptation, the quality of the particles could be evaluated to determine the optimal solution globally in each generation and the individual optimal solution. Each particle has its own speed and position [5]. Through a global optimal solution and the optimal solution as well as individual specific formula, the particle velocity and position could be adjusted to make it tend to the optimal solution. When the iteration error is less than a certain algebraic ε or reaches a given maximum number, the calculation could be stopped to find the optimal solution. Then the optimal solution could be turned into the weight and threshold value of BP neural network and a good optimized neural network could be obtained. This is the process to solve the problem [6].

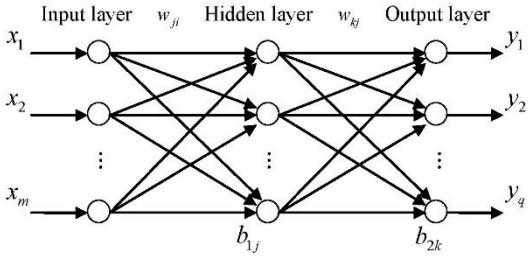

FIGURE 1 BP neural network

2.2 THE ENCODING METHOD OF THE PARTICLES

According to the neural network structure shown in Figure 1, the number of the network input and output layer nodes are m and q. Assuming that the maximum number of hidden nodes are n, then the number of connection weight w_{ji} from output layer to hidden layer are $m \times n$, the number of w_{kj} are $n \times q$, the threshold value b_{1j} are n, b_{2k} are q. So, the number of parameters wait to be optimized in BP neural network are total $1+m \times n+n \times q+n+q$, among which 1 refers to be the number of hidden layer nodes await to be optimized. In all, the PSO algorithm is to find the optimal number of hidden nodes and the corresponding weights and thresholds in this dimensional space.

Depending on the encoding objects, each particle is divided into two parts, the head and body, in which the body can be divided into four parts which could be expressed graphically in Figure 2:

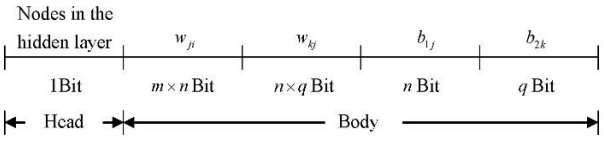

FIGURE 2 Structure of particles

There is only 1 part in the region of head, consisting of the information of the number of the hidden layer nodes in BP network. As the number of hidden layer nodes must be an integer, the head part could be taken the measure of a pseudo-integer encoding, that is, to use real number encoding method in the encoding process, and get the round when calculating the fitness of particles. Thus, an integer of information could be achieved. However, the position and velocity of a particle used in the equation is still a true real number [7].

The ranges of the real numbers in the head part of the particles are determined by the number of the hidden layer nodes in the scope of BP neural network. When conducting the initialization of the particles, the initialization of the real numbers in the head part could be determined by the possible scope of the hidden layer nodes. If the maximum

number of nodes in the hidden layer of BP neural network is $C_{\max}$, the minimum could be $C_{\min}$, thus the number of hidden layer nodes can take $C_{\max} - C_{\min} + 1$. So the ranges of the real number c in the head of particles are $\left[C_{\min} - 0.5, C_{\max} + 0.5\right)$. The reason for adding (minus) 0.5 on both digital range, as it is the only way to make the head portion of particles with equal probability indicates the number of hidden layer nodes. Real number of particles through the head store will be rounded to the nearest whole number obtained in the hidden layer nodes corresponding particle BP neural network.

For the convenience of the following description, the performance code is defined like this:

Performance Code: the real number obtained by integer rounding stored in the head portion is defined as the performance code of the particle. The performance code is the number of particles in the hidden layer node that it represents in BP neural network [8].

The performance code is very important. It shows the number of hidden layer nodes of the current particle under its corresponding BP neural network and plays a guidance role for further fitness calculation of particle and adjustment of particle position and velocity. The performance code is also an effective tool for the researchers to observe the number of the hidden layer nodes during the movement process of particle. Different particles have different performance code, along with different number of hidden layer nodes. Thus, it enables the particles to find different number of hidden layer nodes in space, as well as find out the most proper number of hidden layer nodes in constant search and adjustment of position and speed. By doing so, the structure of BP network could be determined to reach the optimization of the number of hidden layer nodes.

2.2.1 The part of head

Although there is only one real number in this region, it plays a big role. It is equivalent to the soul of particles, and all future operations on the particles depend on the special consideration of this part.

2.2.2 The part of body

Since all weight and threshold values are real numbers, the body part uses the method of real number encoding. Depending on the encoding object, the body part is divided into four smaller parts, representing the connection weight w_{ji} from input layer to hidden layer in BP neural network, the connection weight w_{kj} from hidden layer to output layer, the threshold value b_{1y} in hidden layer as well as b_{2k} in output layer.

The encoding of the body region is designed according to the maximum number of hidden layer nodes (n), to fully reflect all the hidden layer nodes in the head part, in case that there is no enough digits for the body part to save the weight and threshold values. As the performance code is different, it is impossible to reflect the maximum possible number of the hidden layer nodes by performance code, which will inevitably lead to the situation that when the performance code is not the maximum possible number of hidden layer nodes, some rear part of the body could not be used and the calculation in respect of the bit is invalid [9].

3 Particle swarm neural network algorithm with optimized hidden layer nodes

3.1 THE INITIALIZATION OF PARTICLE SWARM

According the way of encoding, before the beginning of the algorithm, you need to initialize the particle swarm work. In this process, you need to form a population based on population size and the numbers of random individuals (particles), in which different individuals represent different neural network structure and different set of weight and threshold value. The speed of each particle and g_{best} and several p_{best} also need to be initialized.

3.2 DESIGN OF NEURAL NETWORK TRAINING AND FITNESS FUNCTION

The optimization of network weight value is an iterative process. Normally in order to ensure that the training neural network has strong generalization ability, in the process of training the network, the given sample space is divided into two parts: one part is the training sample, called a training set; another part is a test sample, called a test set. With the training set to train the network, using the test set to evaluate the accuracy and generalization ability of the network [10].

In order to determine the individual (particle) the merits of the degree in the population, a uniform standard to measure is needed. The following definitions of particle's fitness are given like this:

Fitness: a standard population used to assess the extent of the pros and cons of each particle. Fitness particles usually calculated by the fitness function. Depending on the specific problem, the design of the fitness function is different.

The design of the fitness function is to assess the extent of the major merits of each particle, which is the driving force of evolution. The degree of the merits of the particles is that it represents the accuracy of BP neural network. The smaller the network error on the smaller sample of learning is, the better the fitness of particles is. Hence, the fitness function could be defined like this:

$$Fitness = \frac{1}{q}\sum_{p=1}^{N}\sum_{k=1}^{q}\left(y_{pk} - o_{pk}\right)^2, \quad (1)$$

where N is the number of training samples, q is the number of output layer nodes in the BP neural network, y_{pk}, o_{pk} are actual output and the target output of the p samples at the k output node. As it can be seen from the fitness function,

the smaller the fitness value of particles is, the smaller of the error of BP network is, and the smaller the particles are.

BP neural network training process is: calculated based on particle swarm in the first part of the performance of each individual code to determine the number of hidden layer neural network nodes, and then they are part of the body is mapped to the network weights and thresholds, which constitutes a neural network [11]. Corresponding to each individual neural network input training samples for training. When finished entering all of the samples to calculate the fitness of particles, provide the basis for adjusting the speed and position of the particles, the particles closer to the global final solution to achieve the training of the network.

3.3 THE TERMINATION CONDITION OF THE ALGORITHM OF THE PARTICLE VELOCITY AND THE ADJUSTMENT OF POSITION

The adjustment of the particle velocity is mainly based on the population of the globally optimal solution and its own individual optimal solution. It is conventional to adjust the position of the superimposed position and its speed. And the adjustment of the position of the particle velocity at time $t + 1$ is shown in the following formula [12]:

$$v(t+1) = wv(t) + c_1 r_1 \left(p_{best}(t) - x(t) \right) + c_2 r_2 \left(g_{best}(t) - x(t) \right), \quad (2)$$

$$x(t+1) = x(t) + v(t+1), \quad (3)$$

where w is the inertia factor, c_1 and c_2 are acceleration coefficients, r_1 and r_2 are called to the interval [0,1] of two independent uniformly distributed random numbers. Parameter w, c_1 and c_2 values depend on the specific issues.

The conditions for the termination of algorithm:

Condition 1: when the fitness of the particle is less than the given $\varepsilon(\varepsilon \to 0)$, the algorithm is terminated.

Condition 2: when the revolution of the population reaches to some given upper limit T, the algorithm is terminated.

Condition 3: when the gap of the optimized fitness of the population is less than ε, the algorithm is terminated.

Conditions 1 and 2 is the original PSO algorithm to optimize the conditions for BP neural network that already exists on the basis of this paper, in order to overcome the network. Over-fitting, phenomenon proposed a third termination condition. When the N consecutive optimal fitness groups are less than ε, a majority of the particles has evolved well. If this algorithm continues, although the network will reduce the error, the generalization capability of the network will gradually decrease.

When the algorithm terminates, put out the global optimal solution. The first part of the calculated performance code could be as BP neural network hidden layer nodes, and it will be mapped as part of the body weights and thresholds of the network. Finally we get the optimized BP neural network [13].

3.4 THE ALGORITHM FLOW CHART

The main purpose of hidden layer nodes with the purpose of optimizing the particle swarm neural network algorithm is to seek the number of nodes and weight and threshold value that most closely matches the initial neural network hidden layer. For different initialization, it will come to different hidden layer junction point number. Firstly, we should set the required parameters, and then according to the number and scope of the position and velocity of the particle particles population, we could randomly generate particle population. For each generation of particles, the particles must first be calculated on performance code that represents the number of hidden layer nodes in BP neural network, and then map into the particle body parts neural network weight and threshold value, and also calculate the fitness of particles degrees, and further determine the global optimum and individual optimal solution. We finally generate a new generation of particles continue iteration. Until the fitness particles are less than a given number of iterations or meet the requirements, the algorithm terminates.

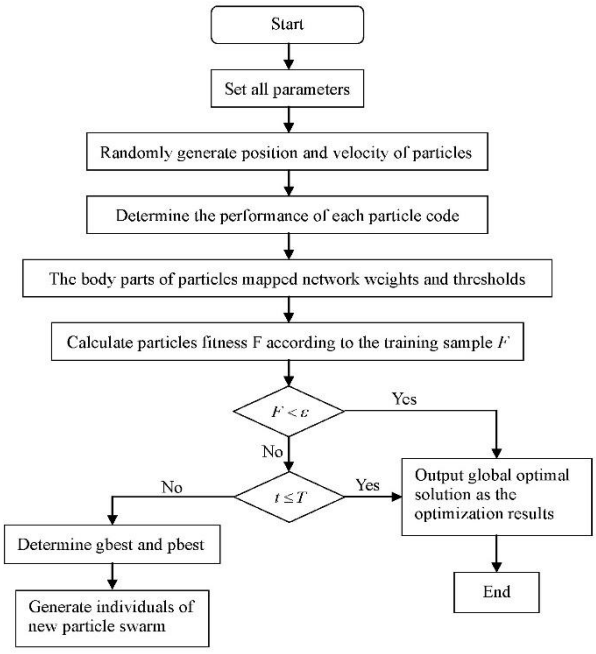

FIGURE 3 Flow chart of the algorithm

4 Smith predictive double controllers based on cooperative large-scale PSO

The paper first uses FT-HPSO to optimize the proposed DFNN parameters and thus fulfil the FT-HPSO-DFNN model. The FT-HPSO large-scale optimization algorithm is adopted in the face of more complicated controlled objects. It is necessary to choose the NN structure that

contains more hidden layers so as to increase network information and approximation capability. Take the proposed "2×20×20×1" DFNN for example. The dimension of the parameter to be optimized in the model is 561. Ordinary PSO algorithms cannot realize excellent optimization. The FT- HPSO-DFNN model plays an important role in Smith double controllers in two ways. As an identifier, it can predict the output of the controlled object and correct modelling errors. As a predictor, it is able to predict the output of the deferred object. After removing the dynamic delay operators in DFNN output layer, the delay time in the system can be eliminated and the change in controlled objects can be predicted ahead of time. Furthermore, the proposed double controllers can separate fixed set point control from interference rejection and enhance the system robustness.

4.1 DESIGN OF SMITH PREDICTIVE DOUBLE CONTROLLERS

The Smith predictive double controllers based on FT-HPSO-DFNN has the predictive control structure as shown in Figure 4:

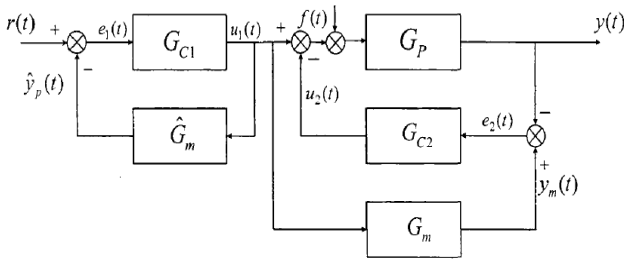

FIGURE 4 Structure of Smith predictive double controllers system

As shown in the above figure, the system structure includes two controllers, G_{C1} and G_{C2}, which give rise to two controlled quantity, $u_1(t)$ and $u_2(t)$. The former plays a part in controlled object G_p, identifier G_m, and predictor $\hat{G}_m$. System error $e_1(t)$ is the difference between set value $r(t)$ and predicated output $\hat{y}_p(t)$. $e_2(t)$ is the difference between the output of identifier G_m and controlled object G_p. In an ideal condition when G_m=G_p, $e_2(t)$=0, and the output of controlled object G_p is determined by the output $u_1(t)$ of controller G_{C1}. The system is a typical Smith predictive control structure. On account of the interference of $f(t)$ or the error in modelling process, $G_m \neq G$ and deviation inevitably occurs between G_p and $r(t)$ under the control of G_{C1}. The model error compensation controller G_{C2} is introduced to get rid of the situations such as $e_2(t) \neq 0$. The proposed Smith predicative double controllers have better robustness owning to a full consideration of the error in modelling process as well as the influence of external disturbance.

The transfer function of the entire system is:

$$Y(s) = H_r(s)R(s) + H_f(s)F(s), \tag{4}$$

where $H_r(s)$ and $H_f(s)$ are the transfer functions of fixed set point control and load disturbance, respectively. Below are their expressions:

$$H_r(s) = \frac{G_{C1}(s)P(s)e^{-ds}}{\left[1+G_{C1}(s)\hat{G}_m(s)\right]} \frac{\left[1+G_{C2}(s)\hat{G}_m(s)e^{-d^*s}\right]}{\left[1+G_{C2}(s)P(s)e^{-ds}\right]}, \tag{5}$$

$$H_f(s) = \frac{G_P(s)e^{-ds}}{1+G_{C2}(s)G_P(s)e^{-ds}}. \tag{6}$$

4.2 CONTROL STRUCTURE & PERFORMANCE ANALYSIS

The normal PI controller is adopted as shown in Equation (7) for better performance analysis of the controller.

$$G_C(s) = K_C\left(1+\frac{1}{T_C s}\right). \tag{7}$$

Describe the controlled object G_P by a first-order inertial element in Equation (8). So the identification model G_m is as shown in Equation (9).

$$G_P(s) = P(s)e^{-ds} = \frac{K_P}{T_P s+1}e^{-ds}, \tag{8}$$

$$G_m = \hat{G}_m e^{-d^*s} = K_P^*\left(1+\frac{1}{T_P^* s}\right)e^{-d^*s}, \tag{9}$$

where d is delay time and d^* is delay time parameter after identification. Put them into the equation, and the results are:

$$H_r(s) = \frac{K_{C1}K_P(T_{C1}s+1)e^{-ds}}{T_{C1}s(T_P^*s+1)+K_{C1}K_P^*(T_{C1}s+1)} \cdot \frac{T_{C2}s(T_P^*s+1)+K_{C2}K_P^*(T_{C2}s+1)e^{-d^*s}}{T_{C2}s(T_Ps+1)+K_{C2}K_P(T_{C2}s+1)e^{-ds}}, \tag{10}$$

$$H_f(s) = \frac{K_P T_{C2}se^{-ds}}{T_{C2}s(T_Ps+1)+K_{C2}K_P(T_{C2}s+1)^{-ds}}. \tag{11}$$

In ideal conditions, $G_p(s)$=$G_m(s)$ and fixed set point control $H_r(s) = \frac{G_{C1}(s)G_P(s)e^{-ds}}{1+G_{C1}(s)G_P(s)}$. It is only determined by fixed set point controller $G_{C1}(s)$, while $H_f(s)$ is only determined by $G_{C2}(s)$. Since $\lim_{s\to 0} H_r(s) = 1$ and $\lim_{s\to 0} H_f(s) = 0$, there are no steady-state errors in the system. Output $y(t)$ can also follow the change of set value $r(t)$ even in dual-ring, contact-free structures. The control signal:

$$U_1(s)=\frac{G_{C1}(s)}{1+G_{C1}(s)\hat{G}_m(s)}R(s)=\frac{K_{C1}(T_P^*s+1)(T_{C1}s+1)R(s)}{T_{C1}s(T_P^*s+1)+K_{C1}K_P^*((T_{C1}s+1)}, \quad (12)$$

So $\lim_{s\to 0}U_1(s)=\frac{1}{K_P}R$. Similarly, $\lim_{s\to 0}U_2(s)=F$. The total controlled quantity of the control system is:

$$\lim_{s\to 0}U(s)=\lim_{s\to 0}(U_1(s)-U_2(s))=\frac{1}{K_P}R-F.$$

In Smith control structure, model precision produces a great influence on control effect. The insensitivity of model mismatch can be analysed when $K_P^*\neq K_P, T_P^*\neq T_P$ and $d^*\neq d$. The controlled quantity $\lim_{s\to 0}U_1(s)=\frac{1}{K_P^*}R$, while $\lim_{s\to 0}U_2(s)=F+(\frac{1}{K_P^*}-\frac{1}{K_P})R$. Thus, the total controlled quantity remains the same:

$$\lim_{s\to 0}U(s)=\lim_{s\to 0}(U_1(s)-U_2(s))=\frac{1}{K_P}R-F. \quad (13)$$

From above we can draw the following conclusions. Model mismatch has no influence on control signals but enhances system robustness. $G_{C1}(s)$ and $G_{C2}(s)$ are mainly influenced by fixed set point controller and barrage jamming. The deviation resulted from model mismatch can be eliminated through $G_{C1}(s)$ and $G_{C2}(s)$.

4.3 DESIGN OF NEURAL NETWORK CONTROLLER

As for controller $G_{C1}(s)$, a three-layer BP neural network is adopted. The controlled quantity is the NN output, which is shown as follows:

$$u_1(t)=\sum_{i=1}^{n}\omega_2(t)\mathrm{sig}\left(\sum_{j=1}^{m}\omega_1(t)e_1(t)\right), \quad (14)$$

where $e_1(t)=r(t)-y_p(t)$; ω_1 and ω_2 are respectively the connection weight of the network; m and n represent the neuron numbers at the input level and the hidden level. The objective function is defined as $J_1(t)=\frac{1}{2}e_1^2(t)$ and then the weight can be updated as the following expression shows:

$$\omega(t+1)=\omega(t)-\eta_1\frac{\partial J(t)}{\partial\omega(t)}, \quad (15)$$

where η_1 is the learning rate of controller $G_{C1}(s)$ and $\frac{\partial J(t)}{\partial\omega(t)}=-e_1(t)\frac{\partial y_p(t)}{\partial\omega(t)}$. So Equation (15) becomes:

$$\omega(t+1)=\omega(t)-\eta_1e_1(t)\frac{\partial y_p(t)}{\partial\omega(t)}. \quad (16)$$

Similarly, the control law of controller $G_{C1}(s)$ is as Equation (17) shows:

$$u_2(t+1)=u_2(t)+\eta_2e_2(t)\frac{\partial y(t)}{\partial u_2(t)}, \quad (17)$$

where η_2 is the learning rate of controller $G_{C2}(s)$.

5 Model predictive control of unknown nonlinear dynamic systems

The above model predictive control helps obtain the future output on the basis of controlled object equation. But it is necessary to use NN for system identification and multistep prediction when the controlled object is a model unknown nonlinear dynamic system. A combined NN model based on FT-HPSO-DFNN is used for system identification. The model parameter built in accordance with the NN after system modelling is as follows:

$$y(k)=g(u(k-d),u(k-d-1),...,u(k-d-m),y(k-1),y(k-2),...,y(k-n)), \quad (18)$$

where $g(\cdot,\cdot)$ stands for a nonlinear function. d, u, and y represent system delay, input and output of the controlled object, respectively. The DFNN structure mentioned is then adopted. After off-line system identification, the dynamic delay operators between output layer and hidden layer are viewed as the input, while the variables in the hidden layer are expressed as the output. Definitions:

$$\varphi_m(k)=[u(k-d),u(k-d-1),...,u(k-d-m)]^{\mathrm{T}}$$

and

$$\varphi_n=[y(k-1),y(k-0),...,y(k-n)]^{\mathrm{T}}.$$

The single-step predicated output of the NN is:

$$\hat{y}(k|\theta(k))=F\left(\sum_{j=1}^{n_h}\omega_{i,j}f_j(a)+b_{i,0}\right), \quad (19)$$

$$a=\sum_{l_m=1}^{n_m}\omega_{j,l_m}\varphi_{l_m}(k)+\sum_{l_n=1}^{n_n}\omega_{j,l_n}\varphi_{l_n}(k-1)+b_{j,0}, \quad (20)$$

where j is the neural network number in the hidden layer; ω_{j,l_m}, ω_{j,l_n} and $\omega_{i,j}$ are the threshold values of the hidden layer and the output layer; $b_{j,0}$ and $b_{i,0}$ mean the bias of the hidden layer and the output layer; f_j and F_i are the excitation functions of the hidden layer and the output layer; $\theta(k)$ represents all adjustable parameters at the moment k.

So multistep output prediction of the controlled object is realized the combined neural network model based on

FT-HPSO-DFNN. Multistep prediction is carried out by dint of the neural network. It is divided into two categories: recursive multistep prediction and non-recursive multistep prediction. The former uses one-step prediction and obtains η-step ahead prediction through repeated equation solving. The latter predicts the $k+\eta$ step by directly using the input and output data at and before the moment *k*. Thus, the two have different nonlinear functions because their predicted horizons are not the same. Consequently, calculation becomes more complicated. Therefore, the paper adopts the recursive multistep prediction method. Compared with Equation (18), the η-step ahead prediction is figured out:

$$\begin{aligned}&\hat{y}(k+\eta|k)=\\&g(u(k+\eta-d),u(k+\eta-d-1),...,u(k+\eta-d-m)\\&\quad y(k-1),y(k-2),...,y(k-\max(n-\eta,0)),\\&\quad\hat{y}(k+\eta-1),\hat{y}(k+\eta-\min(\eta,n)))\end{aligned}\tag{21}$$

$\hat{y}(k+\eta|k)$ is the output of controlled object at the moment $k+\eta$, which is predicated at the moment *k*. Thus, the multistep predicated output of the NN is:

$$\hat{y}(k+\eta|k)=F_i(\sum_{j=1}^{n_h}\omega_{i,j}f_j(\tilde{a}(\eta,j))+b_{i,0}),\tag{22}$$

$$\begin{aligned}\tilde{a}(\eta,j)=&\sum_{\lambda=1}^{\min(\eta-d,n)}\omega_{j,\lambda}\hat{y}(k+\eta-\lambda)+\\&\sum_{\lambda=\min(\eta,n)+1}^{n}\omega_{j,\lambda}y(k+\eta-\lambda)+\\&\sum_{\lambda=0}^{m}\omega_{j,n+1+\lambda}u(k+\eta-d-\lambda)+b_{j,0}.\end{aligned}\tag{23}$$

So the model predictive control problem is based on the proposed PSO so as to carry out constrained optimization and get the optimal controlled quantity.

The entire predictive control process of the PSO-based neural network is as follows:

Step 1: Make *k*=1, predicated horizon be *N*, control horizon be N_u, and weight matrixes be *Q* and *R*;

Step 2: Generate input and output data of the controlled object, and conduct off-line training for FT- HPSO-DFNN;

Step 3: After the NN model stabilizes, calculate predicated the output value $\hat{y}(k)$ of one-step controlled object using the well-trained NN at the sampling time *k*;

Step 4: At the time $k+N_u-1$, obtain a series of control input $u(k+N_u-1|k)$ with the help of the MPC control strategy. And then get the predicated value $\hat{y}(k+\eta|k)$ of controlled object through the recursive model;

Step 5: Minimize the objective function using the PSO algorithm and work out the control increment $\Delta u(k)$;

Step 6: Figure out the control input $\Delta u(k)=\Delta u(k|k)+u(k-1)$;

Step 7: If *k*=*k*+1, return to Step 4.

6 Simulation analysis of model predictive control of multivariable nonlinear systems

The paper uses the multivariable nonlinear ball and plate system to verify the proposed research on model predictive control based on PSO and neural network system identification. The ball and plate system, as a multivariable nonlinear controlled object, is the two-dimensional extension of the ball and beam system. Its controlled object is a plate with two rings pivoted on axes at right angles to each other. The visual feedback enables the ball which rolls freely to maintain balance at any given position on the plate or roll along a particular track.

The state variable is defined as:

$$X=(x_1,x_2,...,x_8)^{\mathrm{T}}=(x,\dot{x},\theta_x,\dot{\theta}_x,y,\dot{y},\theta_y,\dot{\theta}_y)^{\mathrm{T}},\tag{24}$$

where *x* and *y* are the positions of the ball. θ_x and θ_y are the rotor angles on the *X* axis and the *Y* axis. Conduct dynamic modelling for the controlled object of two-degree of freedom according to Lagrange differential equation. The state equation of the ball and plate system is:

$$\begin{bmatrix}\dot{x}_1\\\dot{x}_2\\\dot{x}_3\\\dot{x}_4\\\dot{x}_5\\\dot{x}_6\\\dot{x}_7\\\dot{x}_8\end{bmatrix}=\begin{bmatrix}x_2\\B(x_1x_4^2+x_4x_5x_8-g(\sin x_3-\mu\cos x_3))\\x_4\\0\\x_3\\B(x_5x_8^2+x_1x_4x_8-g(\sin x_7-\mu\cos x_7))\\x_8\\0\end{bmatrix}+\begin{bmatrix}0&0\\0&0\\0&0\\1&0\\0&0\\0&0\\0&0\\0&1\end{bmatrix}\begin{bmatrix}u_x\\u_y\end{bmatrix},\tag{25}$$

where parameter B is defined as $B = \dfrac{m}{m + \dfrac{J_b}{R^2}}$. The ball mass m=0.1kg, and the ball radius R=0.02m. The rotational inertia $J_b = 1.76\text{e}^{-5}\text{Kg.m}^2$. The input-output equation of the entire ball and plate system is expressed as:

$$[x, y]^{\mathrm{T}} = g(\theta_x, \theta_y). \tag{26}$$

Hence, the ball and plate system is a two-input, two-output nonlinear dynamic system that has eight state variables. Firstly, FT-HPSO-DFNN model is used for off-line system modelling. After that, the paper analyses the result of multistep prediction. Control the ball and plate system by dint of the PID controller so as to obtain 2,000 sets of input and output data. The former 1,000 sets of data are used for NN training and the rest for testing. Table 1 contains the predictive results of different NN model tests.

TABLE 1 The comparison of different NN structures predictive results

NN Structure	1-Step Prediction	5-Step Prediction	10-Step Prediction	100-Step Prediction
FT-HPSO-DFNN	1.33E-4	7.01E-4	7.12E-4	7.94E-4
ELM	5.40E-3	9.10E-3	1.43E-2	1.21E-1
ESN	5.61E-3	1.56E-2	2.29E-2	3.07E-2

The paper chooses three NN models: FT-HPSO-DFNN, ESN, and ELM to conduct multistep prediction and simulation. The result shows that precision declines as the predicated step length increases. As the FT-HPSO-DFNN model has high precision in the above 1-Step Prediction, the precision decreases slowly as the predicated step length increases. But the ELM algorithm leads to a quick rate of descent. Therefore, the proposed NN model can be well applied to unknown model predictive control of controlled objects. The precision remains quite high even in the situation of multistep prediction. Figure 5 is the fitted curve of 5-Step Prediction.

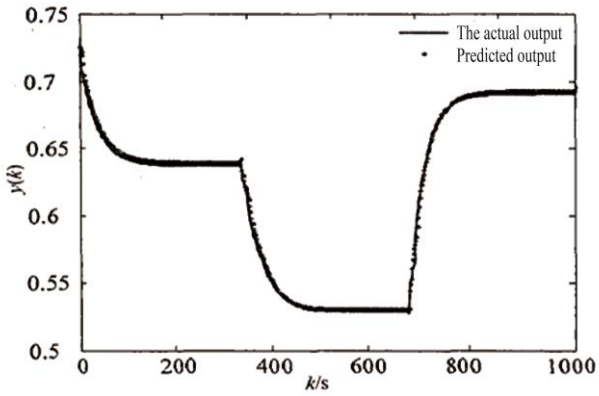

FIGURE 5 The 5 step predictive results of the proposed NN model

As shown in the above figure, the proposed FT-HPSO-DFNN model can realize multistep prediction well in the case of unknown models.

7 Conclusion

To integrate with practical control process better, the paper proposes a nonlinear model predictive control method based on cooperative large-scale PSO aiming at the problem of constrained MIMO model predictive control. Using the FT-HPSO-DFNN model for system modeling, the research realizes multi-step prediction for controlled objects. Furthermore, it adopts the PSO algorithm to optimize constrained objective functions and acquire the optimal control law. Based on simulation experiments of the ball and plate system, the paper proves the proposed model has good control effect and then provides some effective solutions.

Acknowledgments

This paper is supported by National Natural Science Foundation of China (NSFC) Grant (No.71172046) and Shandong Province Higher Educational Science and Technology Program (No.J11LG11).

References

[1] Kuo T J, Hung S Y, Cheng W C 2014 Application of an optimization artificial immune network and particle swarm optimization-based fuzzy neural network to an RFID-based positioning system *Information Sciences* **262** 78-98
[2] Coban C 2014 Power level control of the TRIGA Mark-II research reactor using the multifeedback layer neural network and the particle swarm optimization *Annals of Nuclear Energy* **69** 260-6
[3] Hajihassani M, Jahed Armaghani D, Sohaei H, Tonnizam Mohamad E, Marto A 2014 Prediction of airblast-overpressure induced by blasting using a hybrid artificial neural network and particle swarm optimization, *Applied Acoustics* **80** 57-67
[4] Khajeh M, Kaykhaii M, Hossein Hashemi S, Shakeri M 2014 Particle swarm optimization–artificial neural network modeling and optimization of leachable zinc from flour samples by miniaturized homogenous liquid–liquid microextraction *Journal of Food Composition and Analysis* **33**(1) 32-8
[5] de Mingo López L F, Gómez Blas N, Arteta A 2012 The optimal combination: Grammatical swarm, particle swarm optimization and neural networks *Journal of Computational Science* **3**(1-2) 46-55
[6] Green II R C, Wang L, Alam M 2012 Training neural networks using Central Force Optimization and Particle Swarm Optimization: Insights and comparisons *Expert Systems with Applications* **39**(1) 555-63
[7] Tsekouras G E, Tsimikas J 2013 On training RBF neural networks using input–output fuzzy clustering and particle swarm optimization *Fuzzy Sets and Systems* **221** 65-89
[8] Mirjalili S, Mohd Hashim S Z, Moradian Sardroudi H 2012 Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm *Applied Mathematics and Computation* **218**(22) 11125-37
[9] Valdez F, Melin P, Castillo O 2014 Modular Neural Networks architecture optimization with a new nature inspired method using a fuzzy combination of Particle Swarm Optimization and Genetic Algorithms *Information Sciences* **270** 143-53
[10] Ren C, An N, Wang K, Li K, Hu B, Shang D 2014 Optimal parameters selection for BP neural network based on particle swarm optimization: A case study of wind speed forecasting *Knowledge-Based Systems* **56** 226-39

[11] Chen H S, Leou J J 2012 Saliency-directed color image interpolation using artificial neural network and particle swarm optimization *Journal of Visual Communication and Image Representation* **23**(2) 343-58

[12] Lazzús J A 2013 Neural network-particle swarm modeling to predict thermal properties *Mathematical and Computer Modelling* **57**(9-10 2408-18

[13] Dehuri S, Roy R, Cho S-B, Ghosh A 2012 An improved swarm optimized functional link artificial neural network (ISO-FLANN) for classification *Journal of Systems and Software* **85**(6) 1333-45

Authors

Lizheng Liu, born in February, 1980, Jinan, China

Current position, grades: lecturer in Shandong University of Finance and Economics.
University studies: M.Sc. Degree in Computer Science (2005, Shandong University).
Scientific interest: O2O (online-to-offline) commerce, business intelligence and mobile e-commerce technology.
Publications: 2 books, 2 scientific papers.

Fangai Liu, born in 1962, Qingdao, China

Current position, grades: professor in Shandong Normal University.
University studies: Ph.D in Computer Science (2002, Chinese Academy of Sciences).
Scientific interests: parallel processing, network optimization, e-commerce and grid computing.
Publications: 8 books, over 80 refereed articles and papers in scientific journals.
Experience: head of 3 National Natural Science Funds projects and 5 scientific and technological projects of Shandong province.

Feng Yang, born in August, 1972 in Jinan, China

Current position, grades: associate professor in Shandong University of Finance and Economics.
University studies: Ph.D in Computer Science (2005, Beijing institute of technology). Post doctor in High Computing Lab of Tsinghua University.
Scientific interests: O2O (online-to-offline) commerce, mobile e-commerce technology and mobile cloud computing.
Publications: 15 scientific papers, 9 patents.