ACSE 05 Conference, 19-21 December 2005, CICC, Cairo, Egypt

www.icgst.com

Decoupled Temperature Control System Based on PID Neural Network

Huailin Shu* Youguo Pi**

*Department of Information and Control Engineering
Guangzhou University, 510091, P.R. China
hlshu@163.com
** Institute of Automation Control,
Huanan University of Science and Technology, 510000, P.R. China

Abstract

In this paper, the author analyzes the characteristics of the temperature control systems in industry, which have long delay time, large time constants and strong couple affects. The author then introduces a Proportional, Integral and Derivative neural networks (PIDNN) and lets PIDNN be a multivariable controller. The simulation results for a three-stage heater in a plastic injection machine are shown. It is proved that the PID neural network has perfect decoupling and self-learning control performances in the coupled temperature system.

Key words: *Multivariable system; temperature control; PID neural network; decoupling control.*

1. Introduction

There are many temperature control systems in process but it is difficult to control the coupled multi-point temperature systems. For example, in the process of jetting-moulding, the jetting heater is commonly consists of three to five stages, in which every stage has a independent temperature controller, and the stages will influence each other because of the couple properties. Besides the couple properties, the long delay time and the large time constants will also increase the difficulties of the system control.

Neural networks give us more desires but they haven't be widely used in control areas until now. The reasons may be given as following: (1) Most neural networks have much long training time and can't meet the quick performance of the control system. (2) The prime connective weights of the neural networks are random selected in general and the systems may be unstable in the prime situation . (3) The neurons in most neural networks only have stated function which can't be suitable for actual dynastic systems.

PIDNN, in which proportional(P) neuron, integral(I) neuron and derivative(D) neuron are defined, has created a new concept and a new tool for control. It is not the simple hybrid system of the PID controller and the neural network but a new kind of dynastic neural networks. [1]~[8]

In this paper, the temperature system of the plastic injecting-moulding machine is displayed and is analyzed in Section II. Then, the proportional(P) neuron, integral(I) neuron and derivative(D) neuron are defined in Section III. Their forms, input-output transfer function are introduced too. Based on the P, I, D neurons, the PIDNN are constructed in Section IV. The back-propagation algorithms of the PIDNN are introduced. Finally, the PIDNN is used to control the temperature of a plastic jetting-moulding machine and the results are shown in Section 5.

II. Temperature system in plastic injecting- moulding machine

The temperature of the plastic in injecting-moulding machine decides the quality of the plastic products. But, there are more difficulties to control the temperature because there are three to five heating stages in which every stage has its own heater, and they will influence each other in the injecting process as shown in Fig.1. Generally, the heaters are controlled by some 1-input and 1-output controllers but the performance will be limited because of the coupling property existed.

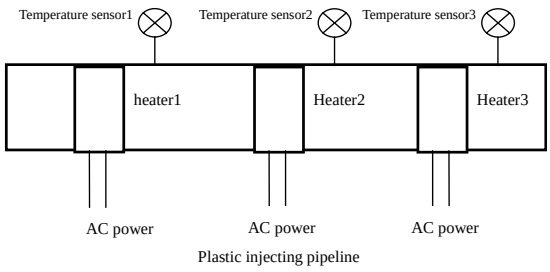


Fig.1 Temperature system in plastic injecting-moulding machine

The transfer function of a three stage injecting-moulding pipeline can be described as follows:

$$G(S)=\frac{Y(S)}{V(S)}$$
$$=\begin{bmatrix} \frac{K_{11}}{1+T_{11}S}e^{-\tau_{11}S} & \frac{K_{12}}{1+T_{12}S}e^{-\tau_{12}S} & \frac{K_{13}}{1+T_{13}S}e^{-\tau_{13}S} \\ \frac{K_{21}}{1+T_{21}S}e^{-\tau_{21}S} & \frac{K_{22}}{1+T_{22}S}e^{-\tau_{22}S} & \frac{K_{23}}{1+T_{23}S}e^{-\tau_{23}S} \\ \frac{K_{31}}{1+T_{31}S}e^{-\tau_{31}S} & \frac{K_{32}}{1+T_{32}S}e^{-\tau_{32}S} & \frac{K_{33}}{1+T_{33}S}e^{-\tau_{33}S} \end{bmatrix}$$

where

$T_{11}=15$, $T_{12}=25$, $T_{13}=45$,
$T_{21}=45$, $T_{22}=15$, $T_{23}=25$,
$T_{31}=65$, $T_{32}=45$, $T_{31}=15$;
$K_{11}=2$, $K_{12}=1$, $K_{13}=0.5$,
$K_{21}=0.6$, $K_{22}=2$, $K_{23}=1$,
$K_{31}=0.3$, $K_{32}=0.6$, $K_{33}=2$;
$\tau_{11}=3$, $\tau_{12}=12$, $\tau_{13}=18$,
$\tau_{21}=15$, $\tau_{22}=3$, $\tau_{23}=12$,
$\tau_{31}=24$, $\tau_{32}=15$, $\tau_{33}=3$;

Obviously, the function is a strong coupling object, in which every input v_i has more strong influences to every output y_i and has long time-delay. So, if only using three convention controllers to control above object, it should have more problems to get good performances.

Conventional decouple controller should be used to solve above problems, but it will cause new problems, such as complicated structure of controller, unknown transfer function of object, and difficulty to apply, etc.

III. PID neurons

Neuron form is shown in Fig.2. Every neuron has an input *u* and an output *x*. The properties of a neuron is decided by the input-output function. The P-neuron, I-neuron and D-neuron are different from each other by proportional (P) function, integral (I) function and derivative (D) function. Their definitions are

P-neuron

If a P-neuron is the *j*th neuron in a network and it has $n-1$ inputs, at any *k* time the input of a neuron is

$$u_j(k)=\sum_{i=1}^{n-1} w_{ij}\cdot x_i(k), \qquad (1)$$

where $x_i(k)$ are the outputs of $n-1$ connected neurons in foregoing layer and w_{ij} are the connected weights.

The input-output function of a P-neuron is the proportional function, as shown in Formula (2).

$$x_j(t)=u_j(t) \quad . \qquad (2)$$

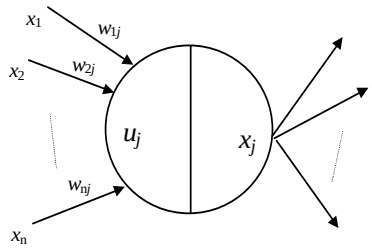


Fig.2 neuron form

And its numeral form is

$$x_j(k)=u_j(k), \qquad (3)$$

I-neuron

If a I-neuron is the *j*th neuron in a network. The input of an I-neuron is the same as that of the P-neuron. See Formula (1). The input-output function of an I-neuron is the integral function, as shown in following formula.

$$x_j(t)=\int_0^t u_j(t)dt\,. \qquad (4)$$

And its numeral form is

$$x_j(k)=x_j(k+1)+u_j(k), \qquad (5)$$

where $x_j(k)$ is the neuron output.

D-neuron

If a D-neuron is the *j*th neuron in a network The input function of a D-neuron is the same as that of the P-neuron. See Formula (1). The input-output function of the D-neuron is the derivative function, as shown by following formula.

$$x_j=\frac{du_j(t)}{dt}. \qquad (6)$$

And its numeral form is

$$x_j(k)=u_j(k)-u_j(k-1)\,. \qquad (7)$$

where $x_j(k)$ is the neuron output.

IV PID neural network (PIDNN)

1. Basic PIDNN

A basic PIDNN has 2 inputs and 1 output and has three layers which are input layer, hidden layer and output layer. The input layer has two neurons and the output layer has one and their neurons are P-neurons. The hidden layer has three neurons and they are P-neuron, I-neuron and D-neuron respectively. The basic PIDNN is shown in Fig.3.

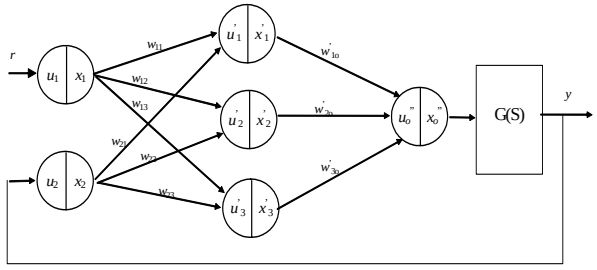


Fig.3 Basic PIDNN

The structure of the basic PIDNN is special. When to choose suitable connective weights, a PIDNN is equal to a PID controller. Generally, let

$w_{1j} = +1$, $w_{2j} = -1$, $w'_{1o} = K_P$, $w'_{2o} = K_I$, $w'_{3o} = K_D$,

then,

$$u'_1 = w_{11}x_1 + w_{21}x_2 = r - y = e,$$
$$u'_2 = w_{12}x_1 + w_{22}x_2 = r - y = e,$$
$$u'_3 = w_{13}x_1 + w_{23}x_2 = r - y = e,$$

and,

$$x'_1 = u'_1 = e,$$
$$x'_2 = \int_0^t u'_2 dt = \int_0^t e dt,$$
$$x'_3 = \frac{du'_3}{dt} = \frac{de}{dt},$$

then,

$$x''_o = u''_o = \sum_{j=1}^{3} w'_{jo}x'_j = w'_{1o}x'_1 + w'_{2o}x'_2 + w'_{3o}x'_3$$
$$= K_P e + K_I \int_0^t e dt + K_D \frac{de}{dt}$$

This speciality is very important and PIDNN can be practically used by the speciality. As we know, most neural networks can't be practically used in system control because the initial connective weights of the neural networks are random selected and the neural controller can't satisfy the stability of the systems.

PID controllers have been widely used in industry and there are much more experiences to choose P, I, D parameters in order to suit the stability. So, if a PIDNN equals to a PID controller, it has the useful foundation of the use. Then, via train and study, PIDNN can get better control performances.

2 Multi PIDNN control system

A multi PIDNN consists of several basic PIDNNs in which every basic PIDNN is a sub-net . A PIDNN multivariable control system is shown in Fig.4.

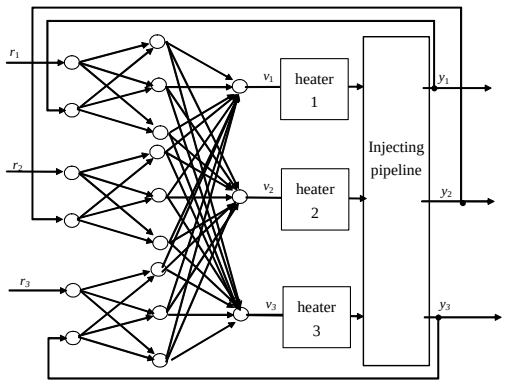


Fig.4 PIDNN multivariable control system

3. Back-propagation algorithms

In PIDNN multivariable control system,the aim of the PIDNN algorithms is to minimize

$$J = \sum_{h=1}^{n} E_h = \frac{1}{m}\sum_{h=1}^{n}\sum_{k=1}^{m}\left[r_h(k) - y_h(k)\right]^2 \qquad (8)$$

where $r_h(k)$ is the inputs and $y_h(k)$ is the outputs of the system as shown in Fig.3. h (= 1, 2, ... , n) is the serial number. The weights of PIDNN is changed by gradient algorithms in on-line training process. After n training steps, the weights from hidden layer to output layer are

$$w_{sjh}(n+1) = w_{sjh}(n) - \eta' \frac{\partial J}{\partial w_{sjh}}, \qquad (9)$$

where,

$$\frac{\partial J}{\partial w_{sjh}} = \frac{\partial J}{\partial E_h}\frac{\partial E_h}{\partial y_h}\frac{\partial y_h}{\partial v_h}\frac{\partial v_h}{\partial x''_h}\frac{\partial x''_h}{\partial u''_h}\frac{\partial u''_h}{\partial w_{sjh}}$$

$$= -\frac{1}{m}\sum_{k=1}^{m}\delta'_h(k)x'_{sj}(k) \qquad (10)$$

where,

$$\delta'_h = 2\left[r'_h(k) - y_h(k)\right]\cdot\frac{y(k+1)\text{-}y(k)}{v(k) - v(k-1)}$$

where η' is the training step, h (= 1, 2, ... , n) is the output neurons' serial number, j (= 1, 2, 3) is the hidden neurons' serial number in every sub-net, s = (1, 2, ... , n) is the sub-nets' serial number, k is training step sequence number and m is the sampling points in every step. x'_{sj} is the output of the hidden layer's neuron, u''_h and x''_h are input and output of the output layer's neuron respectively.

The weights from input layer to hidden layer are

$$w_{sij}(n+1) = w_{sij}(n) - \eta\frac{\partial J}{\partial w_{sij}}, \qquad (11)$$

where

$$\frac{\partial J}{\partial w_{sij}} = \sum_{h=1}^{n} \frac{\partial J}{\partial E_h} \frac{\partial E_h}{\partial y_h} \frac{\partial y_h}{\partial v_h} \frac{\partial v_h}{\partial x_h^{''}} \frac{\partial x_h^{''}}{\partial u_h^{''}} \frac{\partial u_h^{''}}{\partial x_{sj}} \frac{\partial x_{sj}}{\partial u_{sj}} \frac{\partial u_{sj}}{\partial w_{sij}}$$

$$= -\frac{1}{m} \sum_{k=1}^{m} \delta_{sj}(k) x_{si}(k) \qquad (12)$$

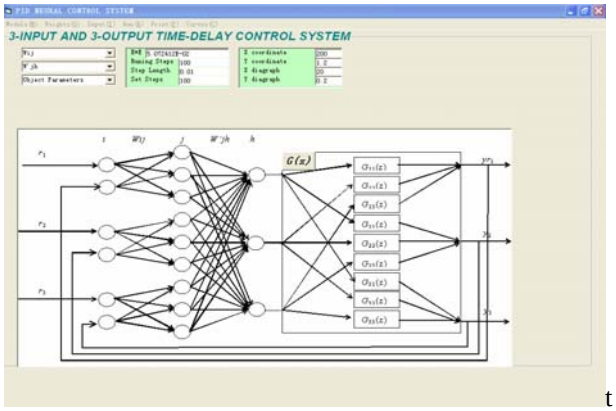
where

$$\delta_{sj} = \sum_{h=1}^{n} \delta_h^{'}(k) w_{sjh}^{'} \frac{x_{sj}^{'}(k) - x_{sj}^{'}(k-1)}{u_{sj}^{'}(k) - u_{sj}^{'}(k-1)}$$

where η is training step, i (= 1, 2) is hidden neurons' sequence number in every sub-net, $x_{si}^{'}$ is outputs of input layer's neurons.

V. Performances of the PIDNN temperature control system

PIDNN can decouple and control the strong-coupled multivariable system and has better performances. When PIDNN is used to control the



t

Fig.5 Interface of the simulation program

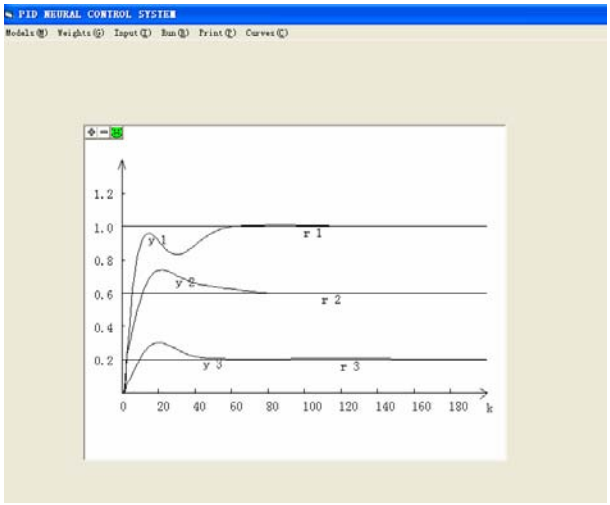
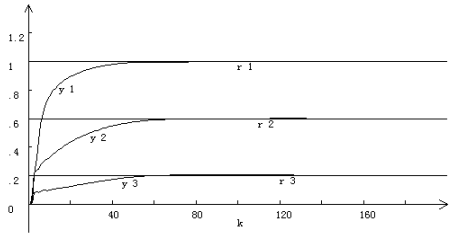


Fig.6 Training process of the PIDNN

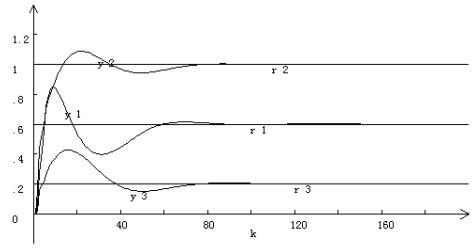
temperature of the injecting-moulding machine, the results can be taken from following simulation. The simulation environment is Microsoft Visual Basic 6.0. The interface of the program is shown in fig.5 and the training process is shown in fig.6.

A 3-output and 6-input PIDNN is used to control the temperature system as shown in Fig.4. To chose the sampling period $T = 3$ s; the weights $w_{s1j} = 0.1$, $w_{s2j} = -0.1$; chose the weights $w_{sjh}^{'}$ as random and smaller than 0.1; the training step $\eta_{sjh}^{'} = 0.01$ and $\eta_{sij} = 0.1$. The temperature range is 0~300 which corresponding to 0~1 scale in the figures.

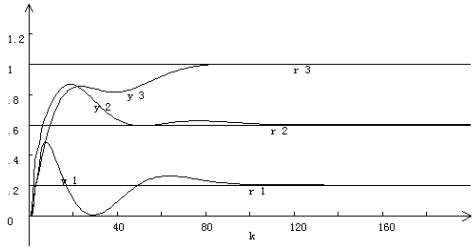
First, choosing the step as the input of the system, $r_1 = 1$, $r_2 = 0.6$, $r_3 = 0.2$, and choosing 100 training steps. Then, to chose $r_1 = 0.6$, $r_2 = 1$, $r_3 = 0.2$ and 100 training steps, too. Finally, choosing $r_1 = 0.2$, $r_2 = 0.6$, $r_3 = 1$ and 100 training steps. The number of sampling points in every step is 200.



(a)



(b)



(c)

Fig.7 responses of the system to different step

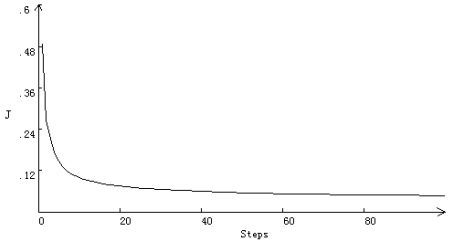


Fig.8 Training target J

The responses of the system after the training process above are shown in Fig. 7. (a), (b), (c). These