2016 International Conference on Mathematical, Computational and Statistical Sciences and Engineering (MCSSE 2016)
ISBN: 978-1-60595-396-0

Application of PID Neural Network Control System Based on Virtual Instrument

Hua SHU

Institute of Mechanical and Electrical Engineering, Guangzhou University, Guangzhou, China

Keywords: Virtual instrument, LabVIEW, PID, Neural network.

Abstract. The PID neural network controller can make both neural networks and PID control into an organic whole, which has the merit of any PID controller for its simple construction and definite physical meaning of parameters, and also has the self-study and adaptive functions of neural network. The approach to realize neural network control in virtual instrument is researched. Combining virtual instrument technology and PID neural network together organically and adopting LabVIEW to design the module of PID neural network, which will improve performance of the control system and development efficiency.

Introduction

PID neural network (PIDNN) integrates the PID control rule with the neural network. Composed with proportion (P), integral (I), and differential (D) the neurons, it is one kind of multilayer neural network that has the merit of PID control and the neural network. Regarding the multivariable close coupling time-dependent system, the PID neural network may adjust the connection weight of the system according to the influence of output performance by target parameter when it changes, and changes proportion, integral and differential action strength in network through on-line study, to enable the system with better dynamic and static performance and achieve the purpose of decoupling control.

At present, American National Instruments Corporation's LabVIEW is the only graphical programming language (G language) with translation ability based on the data stream in the world. Because LabVIEW can rapidly construct the graphical user interface of control system, closely combines measurements and automation hardware, and has completed solution of data acquisition, signal analyzing and information display, the application of PID neural network based on LabVIEW must have bright future in industrial control field [1].

PID Neural Network Control System Structure

The PID neural network (Fig. 1) is a 3-layers forward network cross parallel by the several PID sub-nets. Each sub-net has 2 neurons within each input layer to separately accept modulated value y and given value r, while the hidden layer has 3 neurons with input and output functions being proportion (P), integral (I), differential (D) function. There is one neuron in the output layer to give control value the output object needed [2,3]. In PID neural network, the number of neurons within each layer, the connection method and the connection weight is determined by basic principal of PID control rule and previous experience, making sure the stability and fast convergence of system. PID neural network does not take advantage of traditional PID controller, but also has the parallel structure, the ability of learning and remembering and the ability of approaching any function with multiple network of neural network.

PID Neural Network Forward Algorithm

If controlled object has *m* inputs and *l* outputs, the PID neural network will be composed with *l* sub networks with *m* output ports. The forward function of this PID neural network at sampling time *k* is as follows [4,5]:

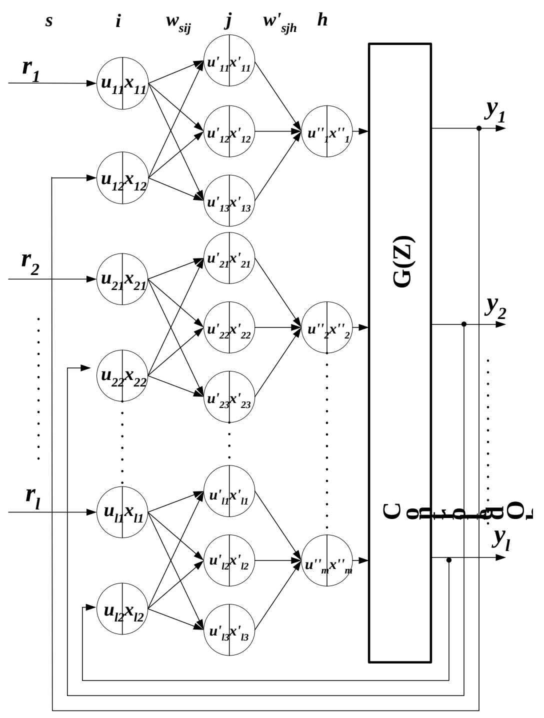

Figure 1. Structure of PID neural network control system.

Input Layer

The input layer has 2*l* same neurons, its input-output relationship is

$$x_{si}(k) = u_{si}(k) \, . \tag{1}$$

In the formula u_i(i=1, 2) is the input value of neurons in input layer , x_i(i=1, 2) is the output value of neurons in input layer, s is the serial number of the parallel sub-nets, and is also the serial number of the controlled variables, s=1, 2, …, l.

Hidden Layer

The hidden layer of PID neural network is consisted with 3×l neurons, which are l proportion neurons, l integral neurons and l differential neurons respectively. The formula of them is the same, which is

$$u'_{sj}(k) = \sum_{i=1}^{2} w_{sij} x_{si}(k) \, . \tag{2}$$

There are three kinds of input output function in hidden layer, the proportion of which is

$$x'_{s1}(k) = u'_{s1}(k) \, . \tag{3}$$

The output of integral neurons is

$$x'_{s2}(k) = x'_{s2}(k-1) + u'_{s2}(k) \, . \tag{4}$$

The output of differential neurons is

$$x'_{s3}(k) = u'_{s3}(k) - u'_{s3}(k-1) \, . \tag{5}$$

In the above functions, s (=1, 2, …, l) stands for parallel sub-nets' serial number, j (=1, 2, 3) stands for neuron serial number in hidden layer, w_{sij} stands for connection weight between input layer and hidden layer of each sub-net. The superscript 'stands for variables in hidden layer.

Output Layer

The output layer of PID neural network has *m* neurons, which form *m* outputs. There is no direct relationship between output neuron serial number and sub-net serial number, and the input of each output neuron is the weighting summation of all neurons' output in hidden layer as follows:

$$u''_h(k) = \sum_{s=1}^{l}\sum_{j=1}^{3} w'_{sjh} x'_{sj}(k) \quad . \tag{6}$$

The neuron output in output layer is

$$x''_h(k) = u''_h(k) \quad . \tag{7}$$

These output values are the control input $v_h(k)$ for object. In the formulas h(=1, 2, …, m) stands for serial number of neurons in output layer, w'_{sjh} stands for connection weight between hidden layer and output layer. The superscript ″ stands for variables in output layer.

Back Propagation Algorithm of PID Neuron Network

The learning goal of PID neural network is to minimize the average value of system output square error

$$J = \sum_{s=1}^{l} E_s = \frac{1}{n}\sum_{s=1}^{l}\sum_{k=1}^{n}\left[r_s(k) - y_s(k)\right]^2 \frac{1}{n}\sum_{s=1}^{l}\sum_{k=1}^{n} e_s^2(k) \quad . \tag{8}$$

In the above formula, *n* stands for number of sampling points at a time, *m* stands for number of controlled variables, and *k* stands for sampling point. The weight can be determined after n_0 steps of training and learning as following formulas [6]:

Weight between Hidden Layer and Output Layer

The iteration formula between hidden layer and output layer is:

$$w'_{sjh}(n_0+1) = w'_{sjh}(n_0) - \eta'_{sjh}\frac{\partial J}{\partial w'_{sjh}} \quad . \tag{9}$$

In the above formula,

$$\frac{\partial J}{\partial w'_{sjh}} = \frac{\partial J}{\partial y_s}\frac{\partial y_s}{\partial x''_h}\frac{\partial x''_h}{\partial u''_h}\frac{\partial u''_h}{\partial w'_{sjh}} = -\frac{1}{n}\sum_{k=1}^{n}\left[r_s(k+1) - y_s(k+1)\right]\cdot \operatorname{sgn}\left[\frac{y_s(k+1) - y_s(k)}{x''_h(k) - x''_h(k-1)}\right]\cdot x'_{sj}(k) \quad . \tag{10}$$

Here the partial differential of object output to object input takes the approximation of relative variation in sign function.

Weight between Input Layer and Hidden Layer

The iteration formula between input layer and hidden layer is

$$w_{sij}(n_0+1) = w_{sij}(n_0) - \eta_{sij}\frac{\partial J}{\partial w_{sij}} \quad . \tag{11}$$

where

$$\frac{\partial J}{\partial w_{sij}} = \sum_{h=1}^{m}\frac{\partial J}{\partial y_s}\frac{\partial y_s}{\partial x''_h}\frac{\partial x''_h}{\partial u''_h}\frac{\partial u''_h}{\partial x'_{sj}}\frac{\partial x'_{sj}}{\partial u'_{sj}}\frac{\partial u'_{sj}}{\partial w_{sij}}$$

$$=-\frac{1}{n}\sum_{k=1}^{n}\sum_{h=1}^{m}\left[r_s(k+1)-y_s(k+1)\right]\cdot\operatorname{sgn}\left[\frac{y_s(k+1)-y_s(k)}{x''_h(k)-x''_h(k-1)}\right]\cdot w'_{sjh}(n_0)\cdot\operatorname{sgn}\left[\frac{x'_{sj}(k)-x'_{sj}(k-1)}{u'_{sj}(k)-u'_{sj}(k-1)}\right]\cdot x_{si}(k)\quad. \tag{12}$$

Here the partial differential of neuron output to neuron input in hidden layer also takes the approximation of relative variation in sign function. Such approximation only influences the learning speed of network, but could greatly simplify computing.

In the above formulas, i(=1, 2) stands for the neuron serial number of sub-net in input layer, j(=1, 2, 3) stands for neuron serial number of sub-net in hidden layer, h(=1, 2, ..., m) stands for neuron serial number of sub-net in output layer, and s(=1, 2, ..., l) stands for serial number of sub-net and serial number of controlled variable.

The Realization of PID Neural Network Control Algorithm

LabVIEW, the virtual instrument software development tool, needs no text programming code. It simplifies the complex language programming into graphics mode and connects various graphics by wires. According to the partition of PID neuron network control algorithm function, several modules in software design is shown in Fig. 2.

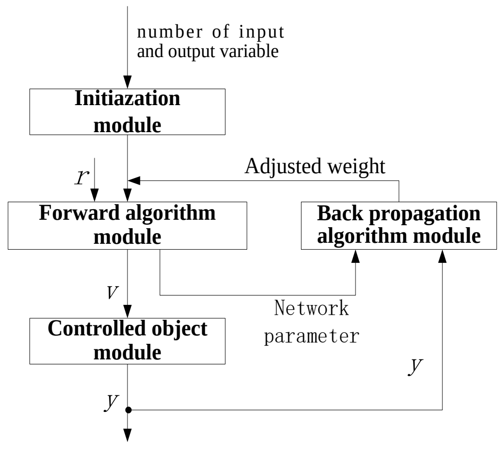

Figure 2. PID neural network control algorithm module.

Initialization Module

The initialization module is used for determining the number of input variable and output variable of neural network. Meanwhile, the initial value selection of network connection weight is according to PID control rule.

The initial value of weight between input layer and hidden layer follows the (r, y)→e mapping requirement:

r⟶ hidden layer $w_{s1j}(0)=+1$

y⟶ hidden layer $w_{s2j}(0)=-1$

The initial weight between hidden layer and output layer takes a small positive number given no priori experience, which takes w_{sj1}=+0.1 here. Meanwhile, to keep the system adjustment without steady-state error, the weight of integral nod form input layer to hidden layer satisfies the non-distortion characteristic of (r, y)→e. Therefore, the absolute values of integral nod weight w_{s12}, w_{s22} are equal during the whole process while signs are the opposite.

Forward Algorithm Module

Given pre-determined value r and controlled object output y, forward algorithm module computes values in hidden layer and output layer of network, according to Eq. 1 ~ Eq. 7.

The output functions of hidden layer are different with each other which include proportion function, integral function and differential function. Considering the controller output must be limited in actual control system, amplitude limiting of network output is used here. Meanwhile, in order to prevent supersaturating, amplitude is also used on integral nods.

Back Propagation Algorithm Module

Back propagation module records each output value in network hidden layer and output layer computed by each forward algorithm of *n* sample point, and computes the average output square error *J* of the system in this training step according to Eq. 8. When *J* is bigger than permissible value of error, the network weight is adjusted according to Eq. 9 ~ Eq. 12.

Controlled Object Module

The controlled object module is a series of sub module that is separately designed according to characteristic and transfer function of different object model. The controlled module includes first and multi-order object, single and multi-variable, as well as linear and non-linear object. The main function of controlled object is to respond the input of PID neural network and give corresponding output according to the characteristic of given object system.

Simulation Example

Supposes the transfer function of controlled object in 2 input - 2 output system is as follows:

$$G(Z)=\frac{Y(Z)}{V(Z)}=\begin{bmatrix} \dfrac{a_{11}z^{-1}}{1-b_{11}z^{-1}} & \dfrac{a_{12}z^{-1}}{1-b_{12}z^{-1}} \\[2ex] \dfrac{a_{21}z^{-1}}{1-b_{21}z^{-1}} & \dfrac{a_{22}z^{-1}}{1-b_{22}z^{-1}} \end{bmatrix} \tag{13}$$

In Eq. 13, a_{11}=0.787, a_{12}=0.1903, a_{21}=0.1535, a_{22}=0.6636, b_{11}=0.6065, b_{12}=0.9048, b_{21}=0.8465, b_{22}=0.7788. Given step input *r1*=1, *r2*=0, we use a 2 input PID neural network to control the above object and take advantage of block learning method, with learning step η=0.1 and each step with *n*=200 sampling points.

Figure 3(a), 3(b), 3(c) are control system responding curve after 0, 10 and 100 steps of training. Fig. 4 is the average value of system output square error in the first 100 steps training. According to the graph, the overshoot of the system is very small with relatively fast learning speed and the target function curve monotonic decreases and decays rapidly.

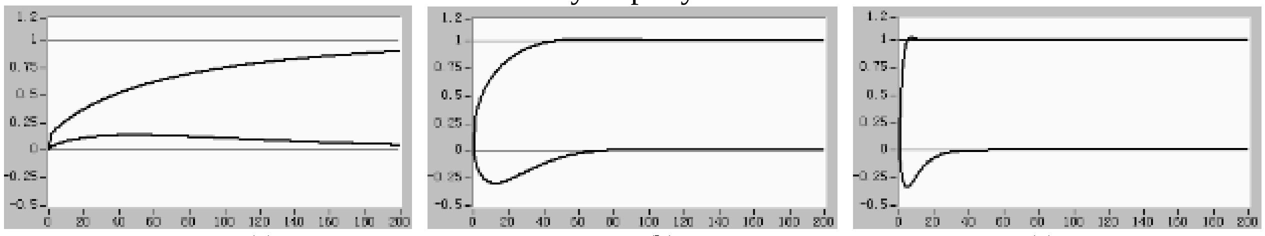
(a) (b) (c)

Figure 3. Control system responding curve.

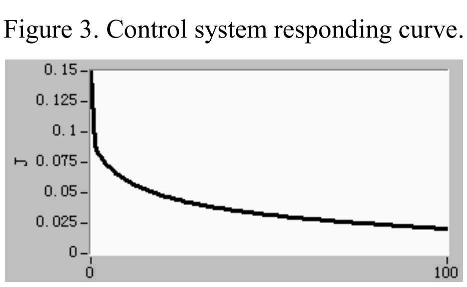

Figure 4. The average value of system output square error.

Conclusions

The research utilizes the theory of graphical programming with virtual instrument, and has realized the simulation of PID neural network control algorithm. It has proved that the PID neural network has good control performance by simulation experiment. This research also demonstrates that graphical programming of LabVIEW can effectively simulate complex algorithm and have the attribute of visualize, easy application and understanding in compared with traditional text programming language.

Acknowledgements

This work was financially supported by Guangzhou Education Bureau under Grant 1201430721.

References

[1] Su Jun and Wu Tuo, Ye Bang-Yan, Zhao Xue-Zhi. Simulation of BP Neural Network Study Algorithm Based on Virtual Instrument Program. Journal of Guangdong Industry Technical College, 2005, 4(1), pp. 5-8.

[2] Huailin Shu. PID neural network for decoupling control of strong coupling multivariable time-delay systems, Control Theory and Application, 1998,15(6), pp.920-924.

[3] Huailin Shu. Analysis of PID neural network multivariable control systems, Acta Automatica Sinaca, 1999,25(1), pp.105-111.

[4] Wang Jun and Tong Wei, Shi Lei, Ren Qing-chang. The application of PID neural network decoupling control technology in the VAV air-conditioning system. Journal of Northwest University(Natural Science Edition), 2002, 32(3), pp.217-23.

[5] Huailin Shu & Youguo Pi. PID neural networks for time-delay systems, Proceedings of 7th Inter-national Symposium on Process Systems Engineering, Elsevier House. 2000, 24, pp.2-7.

[6] Huai-lin Shu, PID neural network and its control system. Beijing: National Defense Industry Press, China, 2006.