國立交通大學

電機與控制工程研究所

碩士論文

動態類神經網路控制系統之設計及其應用

On the Design of Dynamical Neural Network Controller with Its Applications

研究生：黃薰毅

指導教授：王啟旭　教授

中華民國九十七年十二月

動態類神經網路控制系統之設計及其應用

On the Design of Dynamical Neural Network Controller with Its Applications

研 究 生：黃薰毅　　Student：Huang, Hsun-Yi

指導教授：王啟旭　教授　　Advisor：Prof. Chi-Hsu Wang

國 立 交 通 大 學

電 機 與 控 制 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Electrical and Control Engineering

College of Electrical Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Electrical and Control Engineering

December 2008

Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 七 年 十二 月

動態類神經網路控制系統之設計及其應用

研究生：黃薰毅　　指導教授：王啟旭 教授

國立交通大學電機學院電機與控制工程研究所

摘要

由於控制器的老化而造成控制系統的出錯是很普遍的，而這種情況發生，常常因為某些原因，原本的控制器很難被修復。本篇論文探討以動態類神經網路控制器取代原本控制器的可行性來設法解決上述問題。我們以霍普菲爾類神經網路控制器做為動態類神經網路控制器。先以最陡坡降演算法離線訓練霍普菲爾類神經網路的網路權重值使得霍普菲爾類神經網路的輸出能模仿原先的控制器。訓練完成之後再將該霍普菲爾類神經網路當作控制系統的即時控制器。我們以倒單擺系統及球桿系統來驗證該霍普菲爾類神經網路控制器的效果。模擬的結果顯示即使控制系統在和訓練時有不同的初始條件，該霍普菲爾類神經網路控制器依然可以模仿原先的控制器並達到令人滿意的效能。

On the Design of Dynamical Neural Network Controller with Its Applications

Student: Huang,Hsun-yi **Advisor: Prof. Chi-Hsu Wang**

Institute of Electrical and Control Engineering

College of Electrical Engineering

National Chiao Tung University

ABSTRACT

Faults due to the aging of a controller for a control system are very common; once they happen, the controller is quite difficult to be repaired for some reasons. To solve this problem, in this thesis, we discuss the feasibility of replacing the existing controller with a dynamical neural network (DNN) controller. A Hopfield neural network (HNN) controller is used as the DNN controller. The weightings of the HNN are first trained off line by the steepest descent algorithm to make the output of the HNN can mimic the existing controller. After the training is completed, the HNN is applied to the control system as a real-time controller. An inverted pendulum system (IPS) and a ball and beam system (BABS) are used to examine the effectiveness of the proposed HNN controller. The simulation results show that even with the initial condition different from that in the training data, the proposed HNN controller can mimic the existing controller and achieve favorable performance.

致謝

這篇論文的完成使得我在交通大學電機控制碩士班的研究生活畫上一個美好的句點。這段研究期間獲得很多人的幫助，首先我要感謝我的指導老師王啟旭教授指導我研究的方向，我要感謝林保村學長及林炳榮學長提供我研究的建議並和我討論研究的成果，我還要感謝陳品程學長提供我論文寫作的建議。另外我要感謝我的朋友黃庭偉等人，在我研究陷入瓶頸時陪我去散心並鼓勵我繼續堅持。最後我要感謝我的家人，尤其是爸媽的支持，讓我能很從容地完成研究。

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

Many tasks involving intelligence or pattern recognition are extremely difficult to automate, but appear to be performed very easily by animals. It stands to reason that computing systems that attempt similar tasks will profit enormously from understanding how animals perform these tasks and simulation these processes to the extent allowed by physical limitations [1]. Learning is a fundamental aspect of neural networks and a major feature that makes the neural approach so attractive for applications that have from the beginning been an elusive goal for artificial intelligence [2].Artificial neural networks (ANNs) are systems that are constructed to make use of some organizational principles resembling those of human brain. The collective behavior of an ANN, like a human brain, demonstrates the ability to learn, recall, and generalize from training patterns or data [3].In recent years, the research of artificial neural networks (ANNs) is more and more popular. In [4], one of the interesting characteristic of an ANN is that it can present its adaptivity by adjusting the connection strengths to new data or information. In [5], we can use the supervised training to adjust the weighting factors of the ANN. During the training, we can give the ANN a good reference, and let the ANN modifies its weighting factors of all the neurons of it by minimizing the "error" between the ANN and the reference. What is the "error" between the ANN and the reference? We can define many cost (error) functions to present it, and we had better using the functions satisfy the following conditions: the value of each error function must be always positive or zero, and if and only if ANN and reference are identical, the value of each cost (error) function is zero. The procedure of getting the proper weighting factors of all the neurons of ANN is called the training of ANN. It is generally understood that the selection of the ANN training algorithm plays an important role for most ANN's applications [6]. And the way of training of ANN most used is the steepest descent method [5, 7]. We will define the cost (error) function, and explain the steepest descent method in the latter chapters. ANNs consist of many interconnected simple nonlinear systems called neurons. Generally speaking, neuron models can be divided in two basic types, static and dynamic. A dynamic neuron is the one whose output is described by a differential equation, and the dynamic neural network is a neural network containing at least one dynamic neuron [8]. On the other hand, a recurrent neural network (RNN) usually belongs to a dynamical neural network (DNN). A recurrent neural network is a neural network with feedback connections and its techniques have been applied to a wide variety of problems [2]. In [9, 10], we know the Hopfield neural networks (HNNs) are

dynamical neural networks, and they have been extensively studied in many aspects and successfully applied to many fields [11]. We want to use HNN architecture controllers as mentioned in [12, 13]. The weights of the HNN architecture need to be updated using a dynamical learning algorithm during the control process. The learning algorithms adjust the weights of the network, so that the trained network will exhibit the required properties. The general algorithms used for HNN architecture are usually the steepest descent learning algorithms as mentioned before. By evaluating the gradient of a cost (error) function with respect to the weights of the networks, it is possible to iteratively adjust the value of the weights in the direction opposite to the gradient [14]. One of the famous nonlinear plants is the inverted pendulum system (IPS). Although the IPS is not a linear system, we want to know the ability of the HNN as to be a controller, so we try to use HNN controller to control the IPS. The problem is how we find the proper values of the parameters of HNN. First, we provided the architecture for IPS controlled well by a good reference controller and collecting the data of the control signal of the reference controller and the output of IPS. And by the data we collected, we trained the HNN controller to make the weighting factors between neurons of the HNN to be proper values. The process above is called the HNN in the training phase. In fact, the HNN controller after trained is used to mimic the well-designed controller of the IPS. After the training phase, the HNN controller is considered to be with proper values of parameters and it is used as a controller to control IPS in real time and called the HNN in the working phase. If the output of IPS controlled by the HNN controller is a good approximation of the output of IPS controlled by well-designed controller, then the HNN model for well-designed controller must be good [15]. After we succeed using HNN as a controller to control IPS in real time, we try to use it to control the ball and beam system (BABS). The BABS is a more complicated system, and the nonlinearity of BABS is very high, so it is a big challenge of the HNN controller to control BABS. Nevertheless, we can still get a nice result of the control of BABS by the HNN controller used the similar way to train the weighting factors between each two neurons of all neurons of the HNN. So, we think that the HNN architectures have the potential to be good controllers to control some nonlinear systems, and maybe some HNN controllers will be cheap controllers to control some complicated systems.

Chapter 2

Theoretical Foundation

The dynamical (recurrent) neural networks can be classified as being globally or partially dynamical (recurrent). Globally dynamical (recurrent) neural networks have arbitrary feedback connections, including neurons with self-feedback. On the other hand, partially dynamical (recurrent) neural networks have their main structure non-recurrent, or recurrent but with restrictive properties associated with the feedback connections [2].

We take the Hopfield neural network (HNN) as example. HNN belongs to the globally dynamical (recurrent) neural network, in fact, HNN is fully connected neural network, and we will explain the fully connected neural network later.

2.1 Structure of Hopfield Neural Network

.

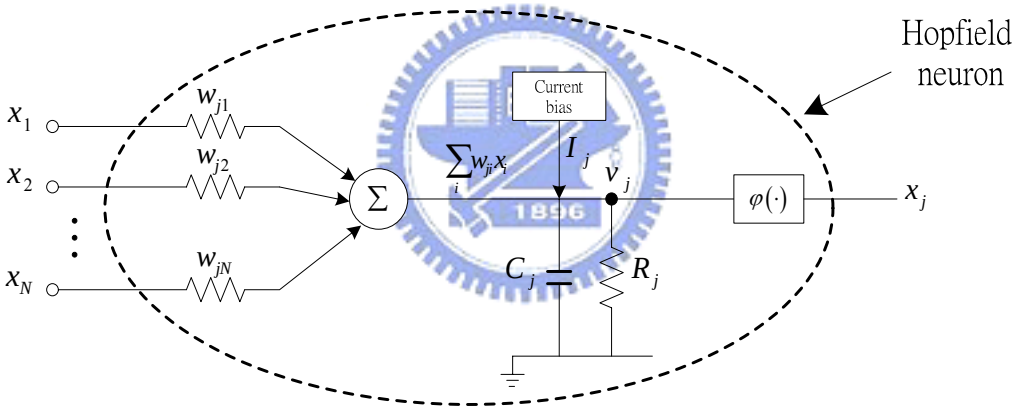

Fig-2.1. The circuit model of a Hopfield neuron

In [16], the circuits model of a Hopfield neuron figure 2-1 shows the input of the neuron is the output voltages $(x_1, x_2, ...x_N)$ of each Hopfield neurons, and the output each of voltages $(x_1, x_2, ...x_N)$ multiple each of the weighting factors, conductance $(w_{j1}, w_{j2}, ...w_{jN})$, become each of the currents $(w_{j1}x_1, w_{j2}x_2, ...w_{jN}x_N)$. Summation of the currents $(w_{j1}x_1, w_{j2}x_2, ...w_{jN}x_N)$ and adding the bias current I_j is the total current $I_{j-total}$, and the total current is the sum of the current I_{jC} passing through capacitor to the ground, and the current I_{jR} passing through the resistor to the

ground. We can express as the following equation:

$$I_{j-total} = \sum_{i=1}^{N} w_{ji} x_i + I_j = I_{jC} + I_{jR} \tag{2-1}$$

The voltage drop between the resistor and the ground equals the resistance R_j multiplying the current I_{jR}, and the voltage drop make the node voltage v_j, as the following equation:

$$v_j = R_j I_{jR} \tag{2-2}$$

And the current I_{jc} passing through the capacitor to ground equals the capacitance C_j multiplying the rate of the varying of the node voltage v_j, as the following equation:

$$I_{jC} = C_j \frac{dv_j}{dt} \tag{2-3}$$

The node voltage v_j passing to the voltage amplifier $\varphi(\bullet)$ will produce the output voltage x_j of the Hopfield neuron, and can be written as the following equation:

$$x_j = \varphi(v_j) = \tanh(v_j) = \frac{e^{v_j} - e^{-v_j}}{e^{v_j} + e^{-v_j}} \tag{2-4}$$

We note the input impedance of the voltage amplifier $\varphi(\bullet)$ is infinity, so no current passing through it and the output impedance of the voltage amplifier $\varphi(\bullet)$ is zero, so the output voltage x_j of the Hopfield neuron will not reduce, no matter how much the output current is. In practice, we often let the bias current I_j to be zero, so it means that we actually don't need the bias current I_j in the circuit model of a Hopfield neuron.

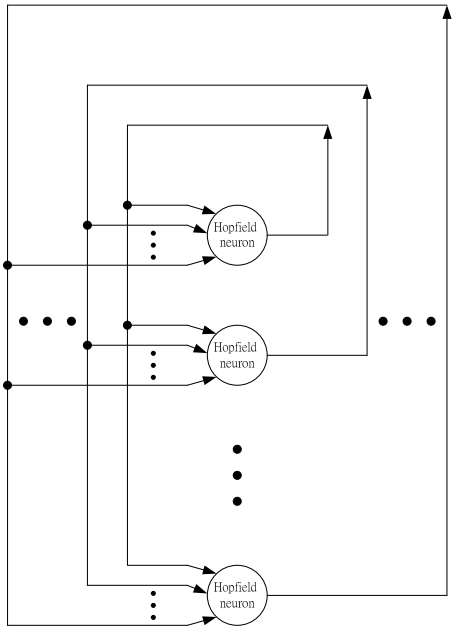

Fig-2.2. The architecture of a Hopfield neural network with Hopfield neurons

Fig-2.2. is also a full connected neural network, that is an artificial neural network architecture in which every node is connected to every node, and these connections may be either excitatory (positive weights), inhibitory (negative weights), or irrelevant (almost zero weights).This is the most general neural network architecture, and every other architecture can be seen to be its special case, obtained by setting some weights to zeros. [1].

The whole Hopfield neural network contains all the individual of Hopfield neurons. Each of the individual of Hopfield neurons can be modeled by the figure 2-1. So by figure 2-2, we have an idea that the input of one Hopfield neuron is the output of all Hopfield neurons of the whole Hopfield neural network, and the output of one Hopfield neuron will be the one of the input of the all Hopfield neurons of the whole Hopfield neural network.

2.2 Lyapunov Energy Function

For any Hopfield neural network with symmetric weights, we can define a suitable Lyapunov energy function as following equation:

$$E = -\frac{1}{2}\sum_{j=1}^{N}\sum_{i=1}^{N} w_{ji}x_j x_i + \sum_{j=1}^{N}\frac{1}{R_j}\int_0^{x_j}\phi^{-1}(x)dx - \sum_{j=1}^{N} I_j x_j \ , \qquad (2\text{-}5)$$

where $\phi^{-1}(x) = v$ is the inverse of the function $x = \varphi(v)$.To show that the equation (2-5) is a Lyapunov function for the Hopfield neural network with symmetric weights, we take the time derivative of equation (2-5):

$$\frac{dE}{dt} = -\sum_{j=1}^{N}[\sum_{i=1}^{N}\frac{1}{2}(w_{ji}x_i + w_{ij}x_i) + I_j - \frac{v_j}{R_j}]\frac{dx_j}{dt} \ . \qquad (2\text{-}6)$$

According to the symmetric weights $w_{ji} = w_{ij}$, so we can write the following equation from equation (2-6):

$$\frac{dE}{dt} = -\sum_{j=1}^{N}[\sum_{i=1}^{N} w_{ji}x_i + I_j - \frac{v_j}{R_j}]\frac{dx_j}{dt}. \tag{2-7}$$

Use the equations (2-1), (2-2), and (2-3), we can write the following equation from equation (2-7):

$$\frac{dE}{dt} = -\sum_{j=1}^{N} C_j \frac{dv_j}{dt}\frac{dx_j}{dt}. \tag{2-8}$$

By chain rule, we can write the following equation from equation (2-8):

$$\frac{dE}{dt} = -\sum_{j=1}^{N} C_j \frac{dv_j}{dt}\frac{dx_j}{dv_j}\frac{dv_j}{dt}. \tag{2-9}$$

Use the equation (2-4), we can write the following equation from equation (2-9):

$$\frac{dE}{dt} = -\sum_{j=1}^{N} C_j \{1-[\tanh(v_j)]^2\}(\frac{dv_j}{dt})^2, \tag{2-10}$$

or we can write the other form of equation (2-10):

$$\frac{dE}{dt} = -\sum_{j=1}^{N} C_j \frac{4}{(e^{v_j}+e^{-v_j})^2}(\frac{dv_j}{dt})^2. \tag{2-11}$$

Because $C_j > 0$, $\frac{4}{(e^{v_j}+e^{-v_j})^2} > 0$, $(\frac{dv_j}{dt})^2 \geq 0$, we get $\frac{dE}{dt} \leq 0$, and thus the Lyapunov energy function E must decrease as the system evolves. Hence, if E is bounded, the system will eventually reach a stable state where $\frac{dE}{dt} = 0$, and $\frac{dv_j}{dt} = 0$

But it is become increasing clear that the symmetric weights assumption has imposed serious constraints on both physical realizations and practical applications of the Hopfield neural networks [17], we will not use the symmetric weights condition of the Hopfield neural networks for our applications. But we still need the assumption that $\frac{dv_j}{dt} = 0$ for we training the weighting factors of HNN. In the latter chapter, we will use the other view to show that in the training of weighting factors of HNN , we can use the assumption $\frac{dv_j}{dt} = 0$.

2.3 Method of Steepest Descent

In [18], first, we set all weights to small values or zeros, and then we present the desired output $U = (u_1, u_2, ...u_k, ...u_n)$, and we calculate the network output $\hat{U} = (\hat{u}_1, \hat{u}_2, ...\hat{u}_k, ...\hat{u}_n)$. Because $\hat{U} = (\hat{u}_1, \hat{u}_2, ...\hat{u}_k, ...\hat{u}_n)$ is a function of weighting vector W and the error between the desired output and network output ,that is the error function J is a function of u_k and $\hat{u}_k$, so we know J is a function of weighting vector W , so we can write $J(W)$ to emphasize the error function is the function of weighting vector W .

In [7], we consider an error function $J(W)$ is a continuously differentiable function of some weighting vector W . We want to find an optimal solution W^* that satisfies the condition bellow:

$$J(W^*) \le J(W) \quad . \tag{2-12}$$

This means minimize the error function $J(W)$ with respect to the weighting vector W , so the necessary condition for (2-12) is

$$\nabla J(W^*) = 0 \quad . \tag{2-13}$$

So we can use the idea of local iteration descent by starting with an initial guess denoted by $W(0)$, generating the a sequence of weighting vectors $W(1)$, $W(2)$,…, $W(k)$, $W(k+1)$,…, such that the error function $J(W)$ being reduced at each iteration of the algorithm as shown by the following inequality:

$$J(W(k+1)) < J(W(k)) \tag{2-14}$$

Where $W(k)$ is the old value of the weighting vector and $W(k+1)$ is its updated value.

For convenience, we write the following equation:

$$g = \nabla J(W) \tag{2-15}$$

So the steepest descent algorithm can be written as the following equation:

$$W(k+1) = W(k) - \eta g(k) \tag{2-16}$$

Where η is a positive constant called learning rate parameter, and $g(k)$ is the gradient vector evaluated at the point $W(k)$ .From (2-16), we can write the following equation:

$$\Delta W(k) = W(k+1) - W(k) = -\eta g(k) \tag{2-17}$$

Now, we use the 1st order Taylor series expansion around $W(k)$ to approximate $J(W(k+1))$ as the following equation:

$$J(W(k+1)) \approx J(W(k)) + g^T(k)\Delta W(k) \tag{2-18}$$

Substituting equation (2-10) to equation (2-11), we can get the following equation:

$$J(W(k+1)) \approx J(W(k)) - \eta g^T(k) g(k) = J(W(k)) - \eta \left\| g(k) \right\|^2 \tag{2-19}$$

Because η is a positive learning rate parameter, so the error function is decreased as

the algorithm progresses from one iteration to the next.

2.4 Training More than One Epoch

As mentioned in the section 2.3, the desired output $U = (u_1, u_2, ...u_k, ...u_n)$ and we calculate the network output $\hat{U} = (\hat{u}_1, \hat{u}_2, ...\hat{u}_k, ...\hat{u}_n)$ can train the weighting vector W on the base of that the weighting vector W is the function of the network output $\hat{U} = (\hat{u}_1, \hat{u}_2, ...\hat{u}_k, ...\hat{u}_n)$ and the network output $\hat{U} = (\hat{u}_1, \hat{u}_2, ...\hat{u}_k, ...\hat{u}_n)$ is the function of the error function J. But after one whole epoch, the training of the weighting vector W may be not enough. One way to solve this problem is to add the number of the time sequence point of the desired output $U = (u_1, u_2, ...u_k, ...u_m)$, and calculate more number of time sequence point of the network output $\hat{U} = (\hat{u}_1, \hat{u}_2, ...\hat{u}_k, ...\hat{u}_m)$, $m > n$, but it is not a good method for take too many samples from a control system in the steady state. Even we just use the same total time with the higher sampling rate to get more points, it maybe produce big run-off error because of the difference of the two neighbor points maybe too small. To avoid the problems above, we can use more than one epoch. After one whole epoch training of the weighting vector W, we can take the final value of the weighting vector W of the first epoch as the initial value of the weighting vector W of the second epoch, and we will get the new value of the network output $\hat{U}(2) = (\hat{u}_1(2), \hat{u}_2(2), ...\hat{u}_k(2), ...\hat{u}_n(2))$, because of the weighting vector W is the function of the network output $\hat{U} = (\hat{u}_1, \hat{u}_2, ...\hat{u}_k, ...\hat{u}_n)$, and after the second epoch, we can take the final value of the weighting vector W of the second epoch as the initial value of the weighting vector W of the third epoch. And we can repeat this method till we finish the last epoch, and the training of the weighting vector W is good enough.

2.5 Fourth Order of Runge-Kutta Formula for Numerical Method

In [19], we use the fourth order of Runge-Kutta formula which provides a good approximation and efficient way for numerical method. Let an initial value problem as following equations:

$$\begin{aligned} y' &= f(t, y) \\ y(t_0) &= y_0 \end{aligned} \tag{2-20}$$

Then, the fourth order of Runge-Kutta formula for this problem can be written by the following equations:

$$
\begin{aligned}
y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\
t_{n+1} &= t_n + h \\
k_1 &= hf(t_n, y_n) \\
k_2 &= hf(t_n + \frac{h}{2}, y_n + \frac{1}{2}k_1) \\
k_3 &= hf(t_n + \frac{h}{2}, y_n + \frac{1}{2}k_2) \\
k_4 &= hf(t_n + h, y_n + k_3)
\end{aligned}
\tag{2-21}
$$

, where y_{n+1} is the fourth order of Runge-Kutta approximation of $y(t_{n+1}) = y(t_n + h)$.

Chapter 3

Control of Inverted Pendulum System (IPS)

3.1 Dynamics of Inverted Pendulum System (IPS)

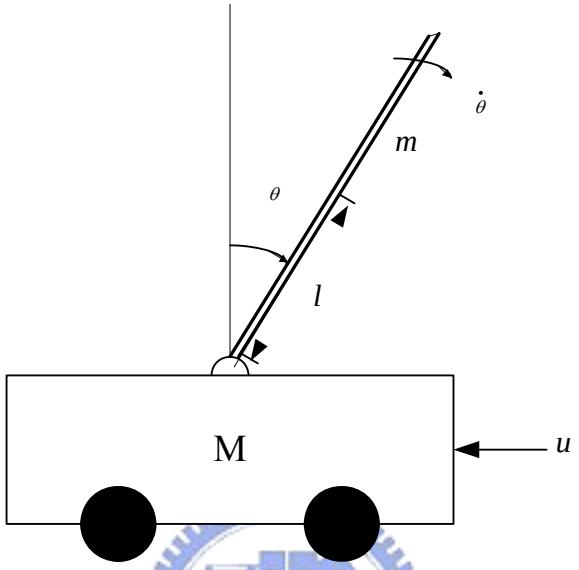

Fig-3.1. The inverted pendulum system (IPS)

In [16, 20], the half-axle length of the IPS l = 0.5 meter; the cart mass of the IPS M = 1 kg; the pendulum mass of the IPS m = 0.1 kg. And we can get the equation:

$$\begin{aligned}\dot{\theta}_1 &= \theta_2 \\ \dot{\theta}_2 &= \frac{g\sin\theta_1 - \frac{ml\theta_2^2\cos\theta_1\sin\theta_1}{M+m}}{l(\frac{4}{3}-\frac{m\cos^2\theta_1}{M+m})} + \frac{\frac{\cos\theta_1}{M+m}}{l(\frac{4}{3}-\frac{m\cos^2\theta_1}{M+m})}\times u\end{aligned} \tag{3-1}$$

, where θ_1 is the bias angle of the IPS; θ_2 is the angle velocity of the IPS; u is the control force to push the cart. We substitute the values above, and we can get the following equation:

$$\begin{aligned}\dot{\theta}_1 &= \theta_2 \\ \dot{\theta}_2 &= 215.6\times\sin(\theta_1) - (\theta_2)^2\times\cos(\theta_1)\times\sin(\theta_1) + \frac{20\times\cos(\theta_1)}{14.667-(\cos(\theta_1))^2}\times u\end{aligned} \tag{3-2}$$

3.2 Design of Reference Controller

Linearize the equation (3-1), by setting $\cos\theta_1 = 1$, $\sin\theta_1 = \theta_1$, $\theta_2^2 = 0$, and we

can get the following equation:

$$\begin{aligned}\dot{\theta}_1 &= \theta_2 \\ \dot{\theta}_2 &= \frac{g}{l(\frac{4}{3}-\frac{m}{M+m})}\times\theta_1 + \frac{\frac{1}{M+m}}{l(\frac{4}{3}-\frac{m}{M+m})}\times u\end{aligned} \tag{3-3}$$

Substitute the values ($M = 1, m = 0.1, l = 0.5, g = 9.8$), we can get the following equation:

$$\begin{aligned}\dot{\theta}_1 &= \theta_2 \\ \dot{\theta}_2 &= 15.8 \times \theta_1 + 1.46 \times u\end{aligned} \tag{3-4}$$

In [21], let $u = k_1\theta_1 + k_2\theta_2$, we can get the following equation by (3-4):

$$\begin{aligned}\dot{\theta}_1 &= \theta_2 \\ \dot{\theta}_2 &= (15.8 + 1.46k_1)\times\theta_1 + 1.46k_2 \times \theta_2\end{aligned} \tag{3-5}$$

And we present (3-5) by matrix form

$$\begin{pmatrix}\dot{\theta}_1 \\ \dot{\theta}_2\end{pmatrix} = \begin{pmatrix}0 & 1 \\ 15.8 + 1.46\times k_1 & 1.46\times k_2\end{pmatrix}\begin{pmatrix}\theta_1 \\ \theta_2\end{pmatrix} \tag{3-6}$$

Let $A = \begin{pmatrix}0 & 1 \\ 15.8 + 1.46\times k_1 & 1.46\times k_2\end{pmatrix}$, so $\lambda I - A = \begin{pmatrix}\lambda & -1 \\ -15.8 - 1.46\times k_1 & \lambda - 1.46\times k_2\end{pmatrix}$,

and if we ask the determinant of $\lambda I - A$ is zero ,we will get the following equation:

$$\det\begin{pmatrix}\lambda & -1 \\ -15.8 - 1.46\times k_1 & \lambda - 1.46\times k_2\end{pmatrix} = \lambda^2 - 1.46k_2\lambda - 15.8 - 1.46k_1 = 0 \tag{3-7}$$

(3-7) is the form of the second order linear system $s^2 + 2\xi\omega_n s + \omega_n^2 = 0$, so we get

$$\begin{aligned}\xi\times\omega_n &= -0.73\times k_2 \\ \omega_n^2 &= -15.8 - 1.46k_1\end{aligned} \tag{3-8}$$

Let $\xi = 0.4$, $\omega_n = 1.8$, we can get $k_1 = -13$, $k_2 = -0.99$, so we have

$$u = -13\theta_1 - 0.99\theta_2 = -13\theta_1 - 0.99\dot{\theta}_1 \tag{3-9}$$

So the equation of the reference controller of IPS is equation (3-9). Let the IPS initial angle=20 degree and initial angular speed=20 degree/sec, and the simulation time=10 sec. With the equations (3-2) and (3-9), we can get the results of the IPS controlled by the reference controller, and we show them in the following figures:

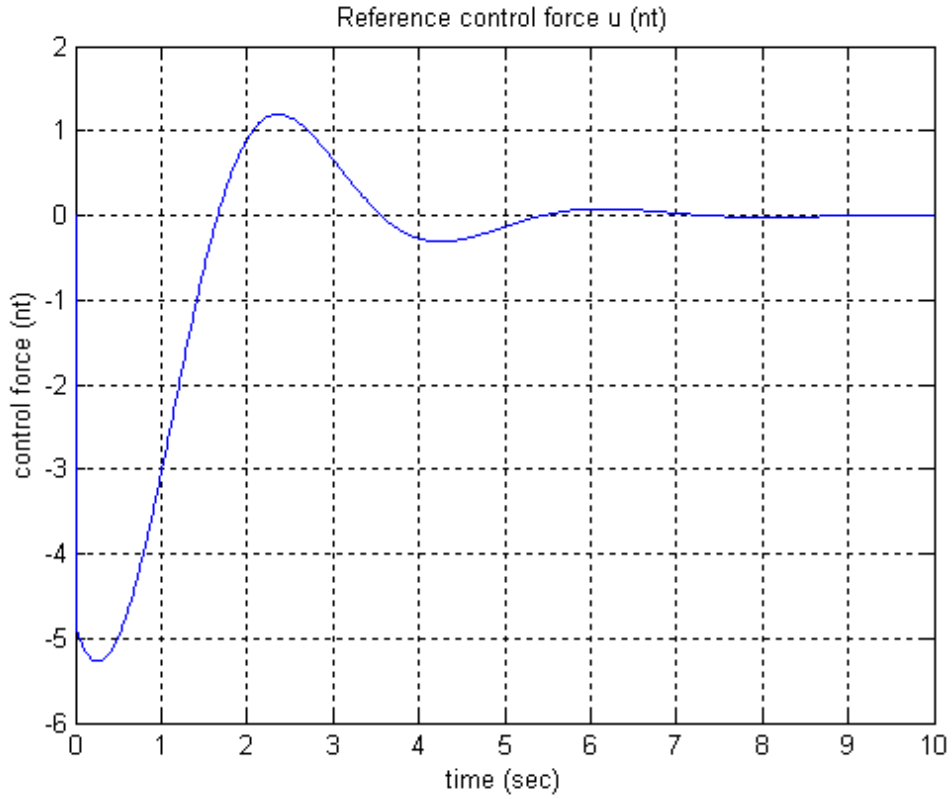

Fig-3.2. The control force of the reference controller

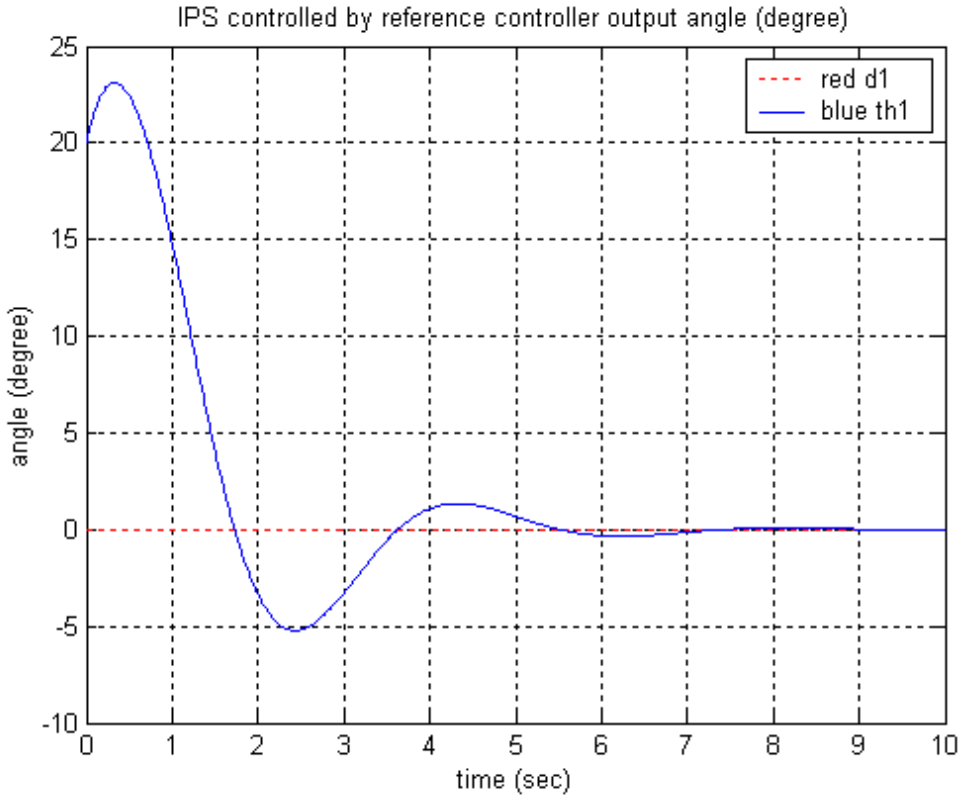

Fig-3.3. The output angle of IPS controlled by the reference controller

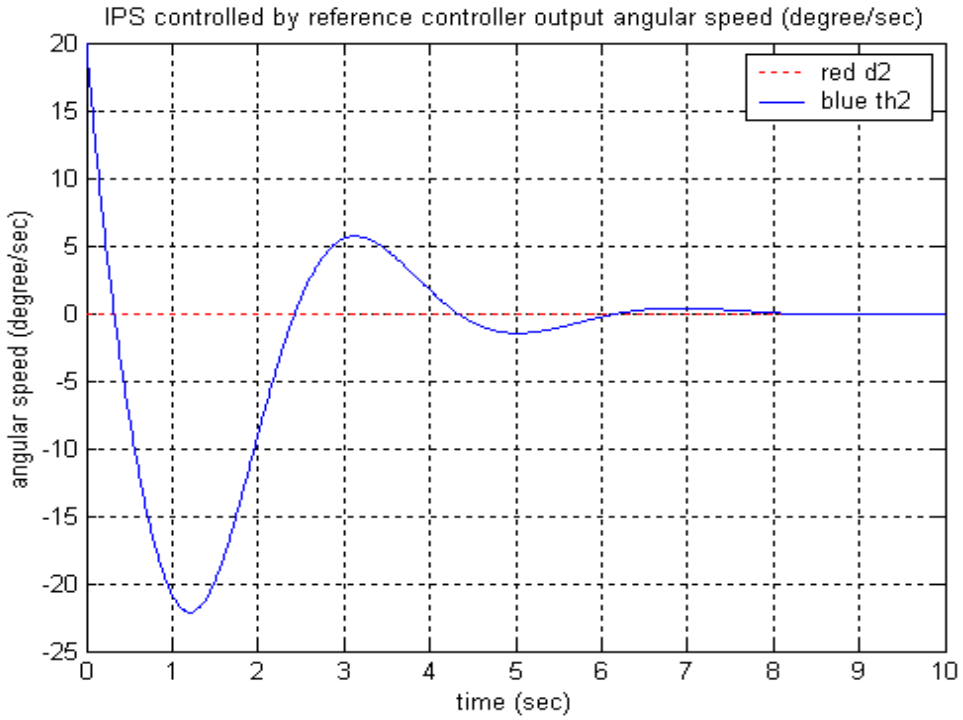

Fig-3.4. The output angular speed of IPS controlled by the reference controller

3.3 Architecture and Algorithm of Weighting Training

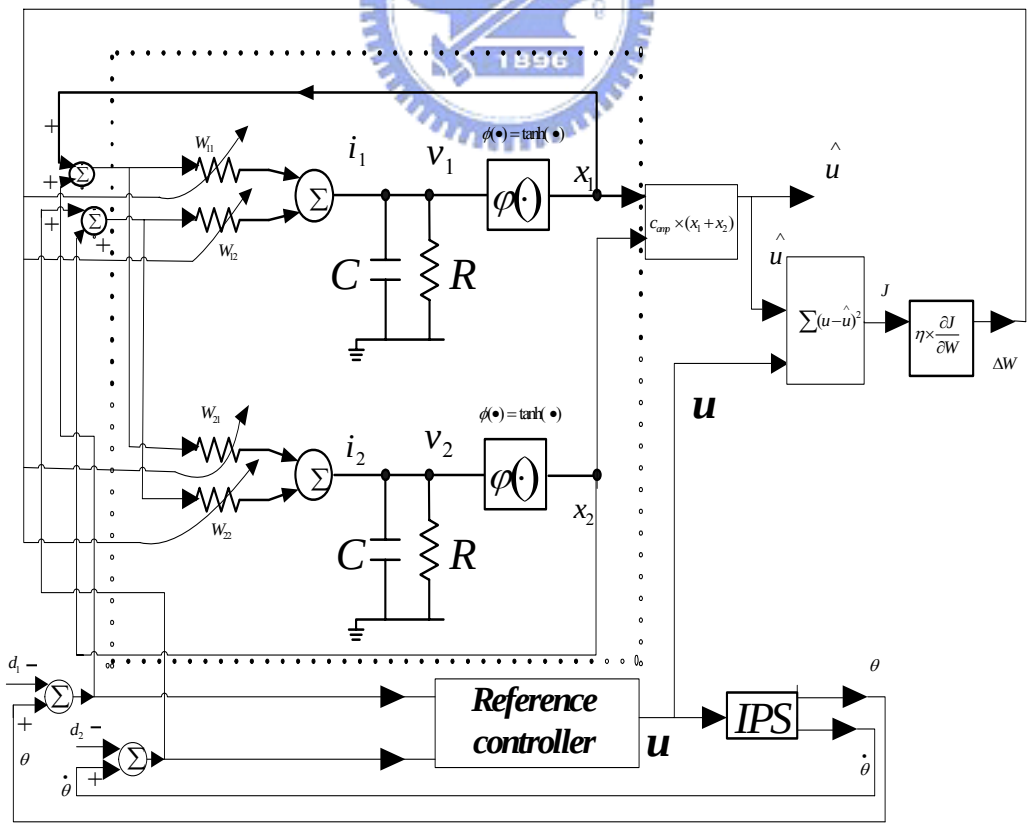

Fig-3.5. The architecture of the HNN controller in the training phase

We use the following parameters of HNN controller: the resistor=1(ohm), the capacitor=0.01(Farr), the amplification constant=-30, and the learning rate=0.001.
In the training phase, we note that the Hopfield neural network (HNN) doesn't control the IPS. So we can imagine that when train w_{11}, w_{12}, w_{21} and w_{22} by the set of $(u,\theta_1,d_1,\theta_2,d_2)$ (the value of the set is corresponding to this moment time t_1), the time of the reference controller and the IPS pauses until the HNN circuit is in the steady state. And we continue to train w_{11}, w_{12}, w_{21} and w_{22} by the next set of $(u,\theta_1,d_1,\theta_2,d_2)$ (the value of the "next set" is corresponding to the next moment time t_2).
So we can consider that in the training phase, the circuit is already in the steady state, so no current passes to both capacitors C (it is equal to both capacitors C are open, C=0), so we can get the equation bellow (by the circuit theorem: the voltage drop equals the multiplication of the resistance and the current):

$$\begin{aligned} v_1 &= R \times i_1 \\ v_2 &= R \times i_2 \end{aligned} \tag{3-10}$$

We have the equation bellow (by the circuit theorem: current equals the multiplication of the conductance and the voltage):

$$\begin{aligned} i_1 &= w_{11} \times (x_1 + \theta_1 - d_1) + w_{12} \times (x_2 + \theta_2 - d_2) \\ i_2 &= w_{21} \times (x_1 + \theta_1 - d_1) + w_{22} \times (x_2 + \theta_2 - d_2) \end{aligned} \tag{3-11}$$

The effect of $\varphi(\bullet)$ is as voltage amplifier, so we can write the equations bellow:

$$\begin{aligned} x_1 &= \varphi(v_1) = \tanh(v_1) \\ x_2 &= \varphi(v_2) = \tanh(v_2) \end{aligned} \tag{3-12}$$

The equation of the output of Hopfield neural network, the HNN controller's control signal $\hat{u}$ is bellow:

$$\hat{u} = c_{amp} \times (x_1 + x_2) \tag{3-13}$$

We define the error J as the measurement of the half squared distance between the reference controller's control signal u and the HNN controller's control signal $\hat{u}$. And we write the equation bellow:

$$J = \frac{1}{2} \times (u - \hat{u})^2 \tag{3-14}$$

Next, we have to find a good set of w_{11}, w_{12}, w_{21} and w_{22} to make J smaller, so the

half squared distance between the reference controller's control signal u and the HNN controller's control signal $\hat{u}$ will be smaller. How can we reach this task? We can deal it by training w_{11}, w_{12}, w_{21} and w_{22} with the steepest descent method as we introduced in the section 2.3. Let us take w_{11} for example to show how it to be trained. First, we can write the equation bellow to show how we get the better next value of w_{11}:

$$w_{11}(k+1) = w_{11}(k) - \eta \times \frac{\partial J}{\partial w_{11}}, \tag{3-15}$$

where η is the learning rate, and we should calculate the value of $\frac{\partial J}{\partial w_{11}}$. We can use the chain rule to write the equation bellow:

$$\frac{\partial J}{\partial w_{11}} = \frac{\partial J}{\partial \hat{u}} \times \frac{\partial \hat{u}}{\partial x_1} \times \frac{\partial x_1}{\partial v_1} \times \frac{\partial v_1}{\partial w_{11}} \tag{3-16}$$

From the equation (3-14), we have the equation bellow:

$$\frac{\partial J}{\partial \hat{u}} = (\hat{u} - u) \tag{3-17}$$

From the equation (3-13), we have the equation bellow:

$$\frac{\partial \hat{u}}{\partial x_1} = c_{amp} \tag{3-18}$$

From the equation (3-12), we have the equation bellow:

$$\frac{\partial x_1}{\partial v_1} = 1 - [\tanh(v_1)]^2 \tag{3-19}$$

From the equation (3-10), (3-11), we have the equation bellow:

$$\frac{\partial v_1}{\partial w_{11}} = R \times (x_1 + \theta_1 - d_1) \tag{3-20}$$

So, from the equations (3-16), (3-17), (3-18), (3-19) and (3-20), we have equation bellow:

$$\frac{\partial J}{\partial w_{11}} = (\hat{u} - u) \times c_{amp} \times \{1 - [\tanh(v_1)]^2\} \times R \times (x_1 + \theta_1 - d_1) \tag{3-21}$$

Substitute equations (3-21) to the equations (3-15), we have equation bellow:

$$w_{11}(k+1) = w_{11}(k) - \eta \times (\hat{u} - u) \times c_{amp} \times \{1 - [\tanh(v_1)]^2\} \times R \times (x_1 + \theta_1 - d_1) \tag{3-22}$$

Similarly, for training w_{12}, w_{21} and w_{22}, we can write the equations bellow:

$$w_{12}(k+1) = w_{12}(k) - \eta \times (\hat{u} - u) \times c_{amp} \times \{1 - [\tanh(v_1)]^2\} \times R \times (x_2 + \theta_2 - d_2) \tag{3-23}$$

$$w_{21}(k+1) = w_{21}(k) - \eta \times (\hat{u} - u) \times c_{amp} \times \{1 - [\tanh(v_2)]^2\} \times R \times (x_1 + \theta_1 - d_1) \quad (3\text{-}24)$$

$$w_{22}(k+1) = w_{22}(k) - \eta \times (\hat{u} - u) \times c_{amp} \times \{1 - [\tanh(v_2)]^2\} \times R \times (x_2 + \theta_2 - d_2) \quad (3\text{-}25)$$

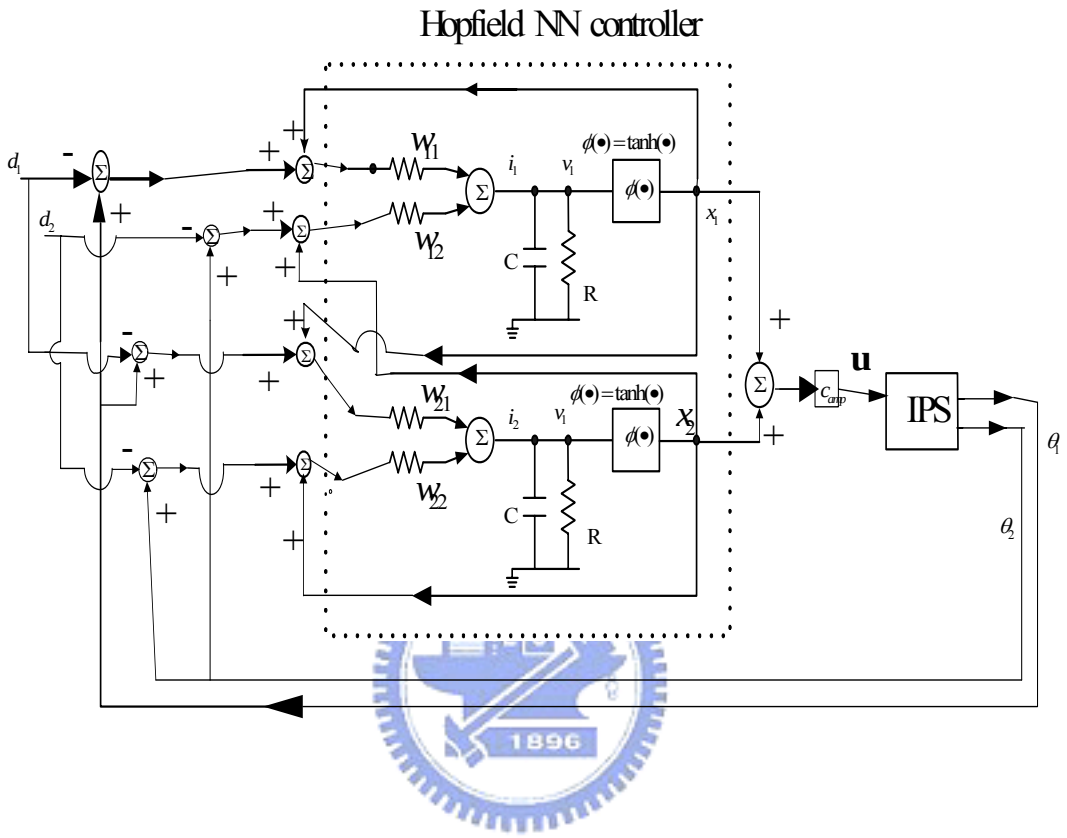

Fig-3.6. The architecture of the IPS controlled by the HNN controller

In the working phase, the HNN is a real time controller, responding to control the IPS. The value of the set of w_{11}, w_{12}, w_{21} and w_{22} is fixed. And as a real time controller, the circuit is dynamic, so we cannot ask the circuit always in the steady state, so we should know that the current passing the both capacitors C is not always zero. Actually, it is very complicated to calculate the output of IPS. In the working phase, the architecture is the recurrent neural network. The output of IPS is $(\theta_1(t), \theta_2(t))$, and $(\theta_1(t), \theta_2(t))$ will affect the values of the current $(i_1(t), i_2(t))$, and the current $(i_1(t), i_2(t))$ will affect the values of the voltage $(v_1(t), v_2(t))$, and the time-varying current passing the capacitors will affect the values of the voltage $(v_1(t), v_2(t))$, too. The voltage $(v_1(t), v_2(t))$ will be amplified by $\varphi(\bullet)$ to

get the values of $(x_1(t), x_2(t))$, and the values of $x_1(t)$ and $x_2(t)$ will affect $i_1(t)$ and $i_2(t)$ respectively, and $(x_1(t)+x_2(t))\times c_{amp}$ will get the control signal $u(t)$. Of course, the control signal $u(t)$ will affect the output of the IPS $(\theta_1(t), \theta_2(t))$ by the IPS equation (3-2). So this is a complicated recurrent control system. So we don't calculate the analytic solution of this system. Instead, we try to use the numerical method to simulate this system.

3.4 Simulation Result

3.4.1 Seeking for Weighting Factors

First, we should try to seek a good set of w_{11}, w_{12}, w_{21} and w_{22}. With the equations (3-22), (3-23), (3-34) and (3-25), we may use many training epochs for get better training results. How can we use training epochs to get better results? First, we define a new parameter: the total error during the whole time J_t as the following equation:

$$J_t = \sqrt{\frac{1}{m}\sum_{k=1}^{m} J(k)} \tag{3-27}$$

Where the parameter $J(k)$ is just the detailed presentation of J, with $J(k) = \frac{1}{2}\times(u(k)-\hat{u}(k))^2$, to emphasize J is the function of the parameter k (the discrete time point). And to emphasize the inference of the epoch number to the total error during the whole time we can write the equation bellow:

$$J_t(epo) = \sqrt{\frac{1}{m}\sum_{k=1}^{m} J(k, epo)} \tag{3-28}$$

Because J_t is the function of the parameter epo (the epoch number), and generally speaking, J_t will decrease as epo increase. So we can use a big value of epo to let J_t become small enough. We can show this in the following figure, Fig-3.7, and we can find that using 6 epochs is good enough for simulation.

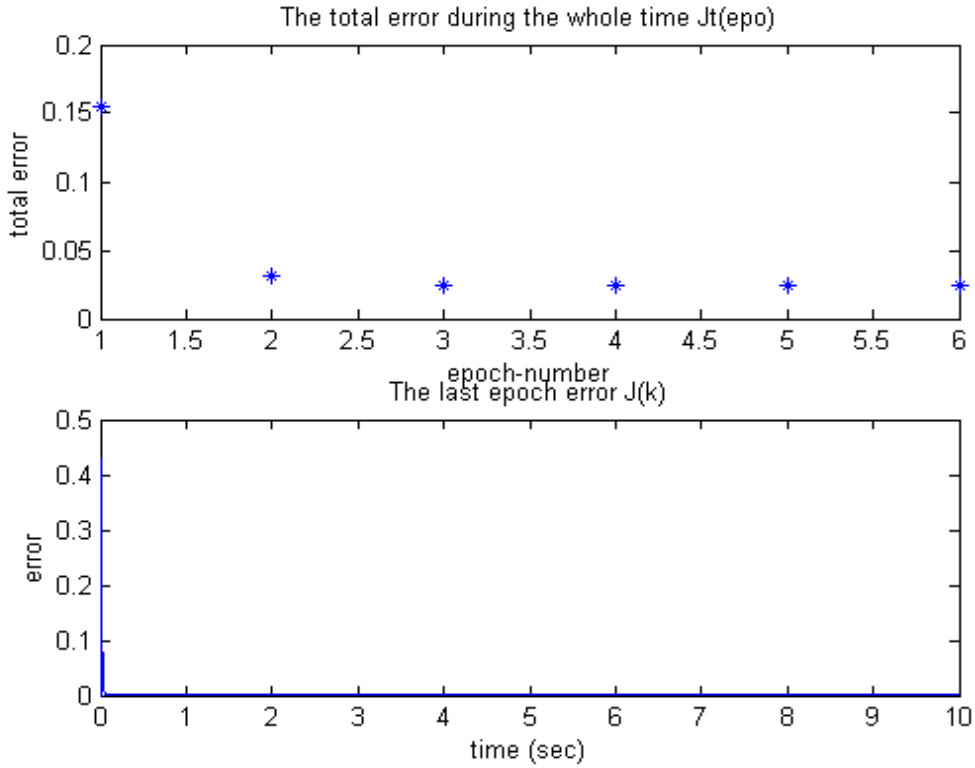

Fig-3.7. The total error $J_t(epo)$ and the last epoch error $J(k,6)$

By the simulation, we can find a set of w_{11}, w_{12}, w_{21} and w_{22}, and we write it down as $(w_{11}, w_{12}, w_{21}, w_{22}) = (0.1755, 0.01367, 0.1755, 0.01367)$.

In fact, the seeking of a good set of w_{11}, w_{12}, w_{21} and w_{22} is the process of the training iteration in the whole time and every epoch, and we can show the training iteration process (in the last epoch) in the whole time by the following figure.

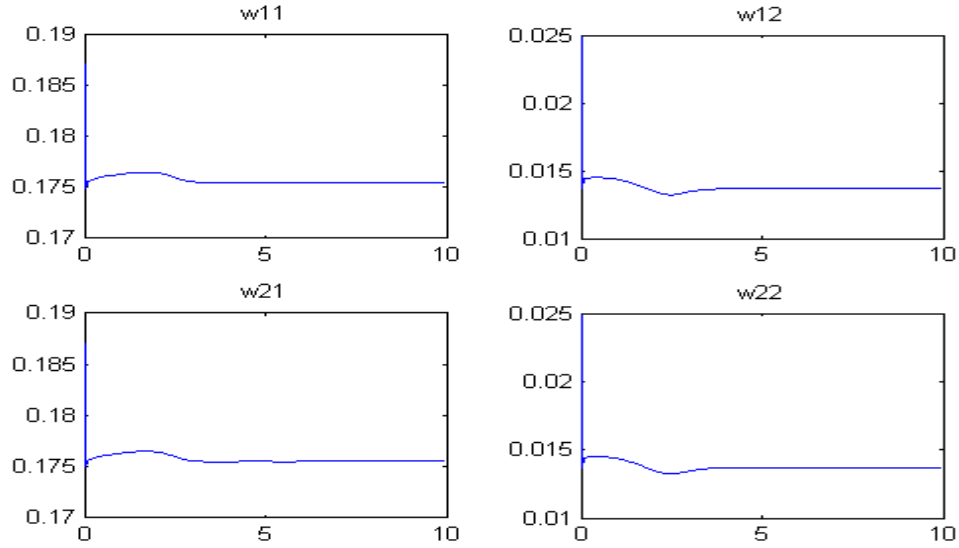

Fig-3.8. The last epoch training iteration process of w_{11}, w_{12}, w_{21} and w_{22}

3.4.2 IPS Controlled by HNN Controller Trained with the Same Initial State

After we seek the values of w_{11}, w_{12}, w_{21} and w_{22}, we can begin to run the simulation of the IPS controlled by the HNN controller in real time. We use the Matlab with the 4^{th} order of Runge-Kutta formula to simulate IPS controlled by HNN controller. By the simulation, we can get the following figures as the results.

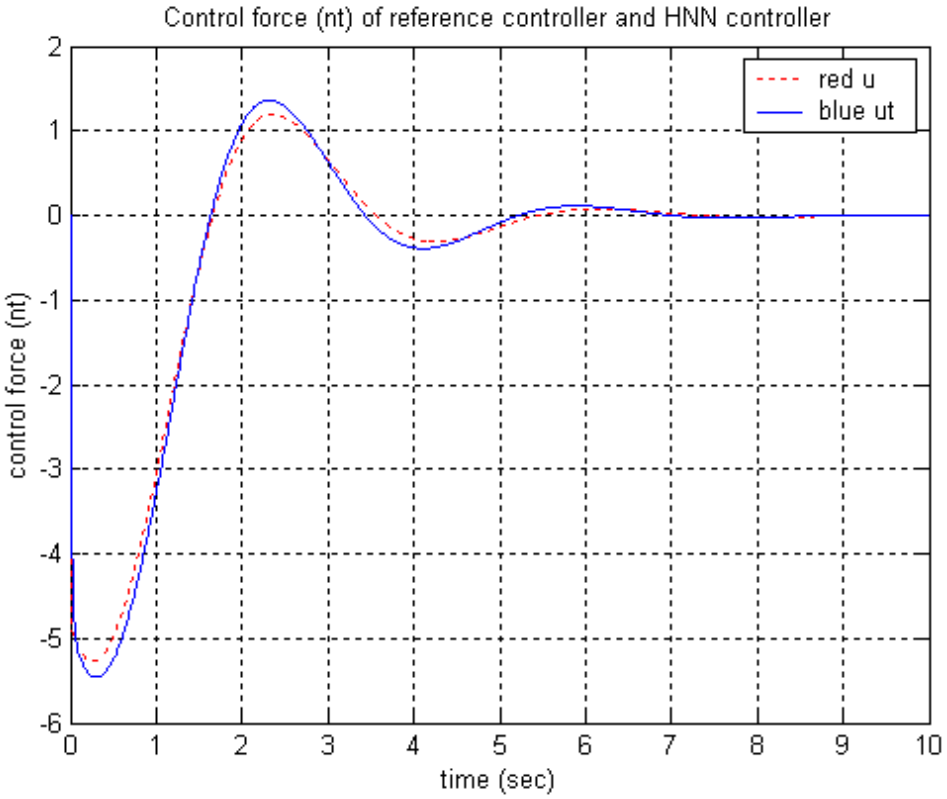

Fig-3.9. The control forces of IPS: reference controller (dash line) and HNN controller (solid line)

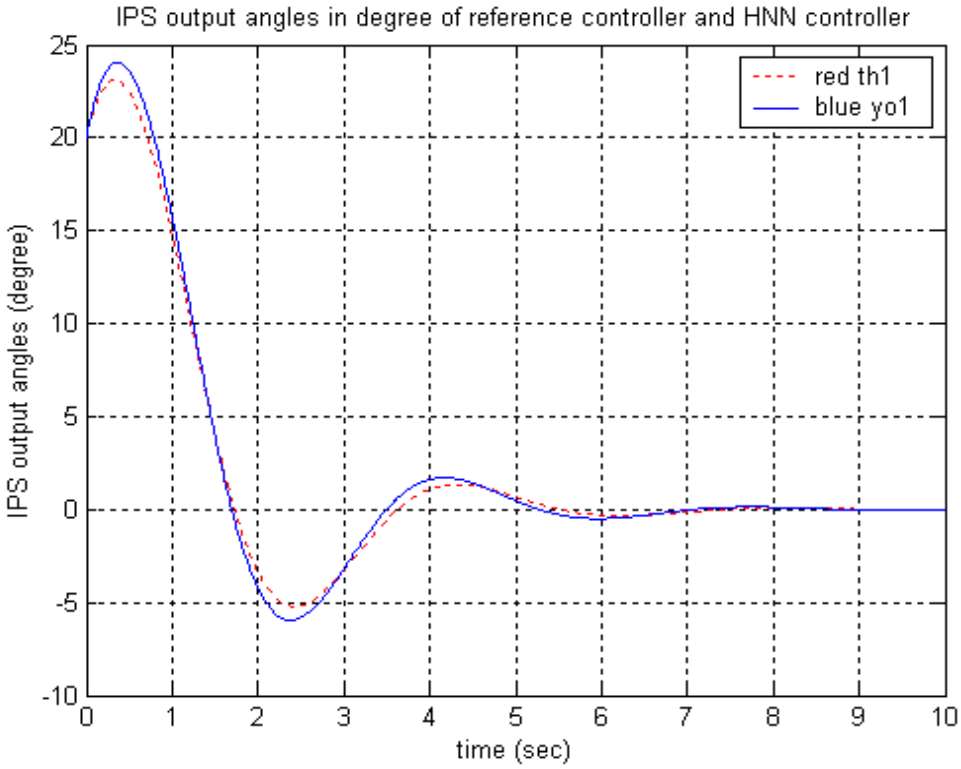

Fig-3.10. The output angles of IPS: reference controller (dash line) and HNN controller (solid line)

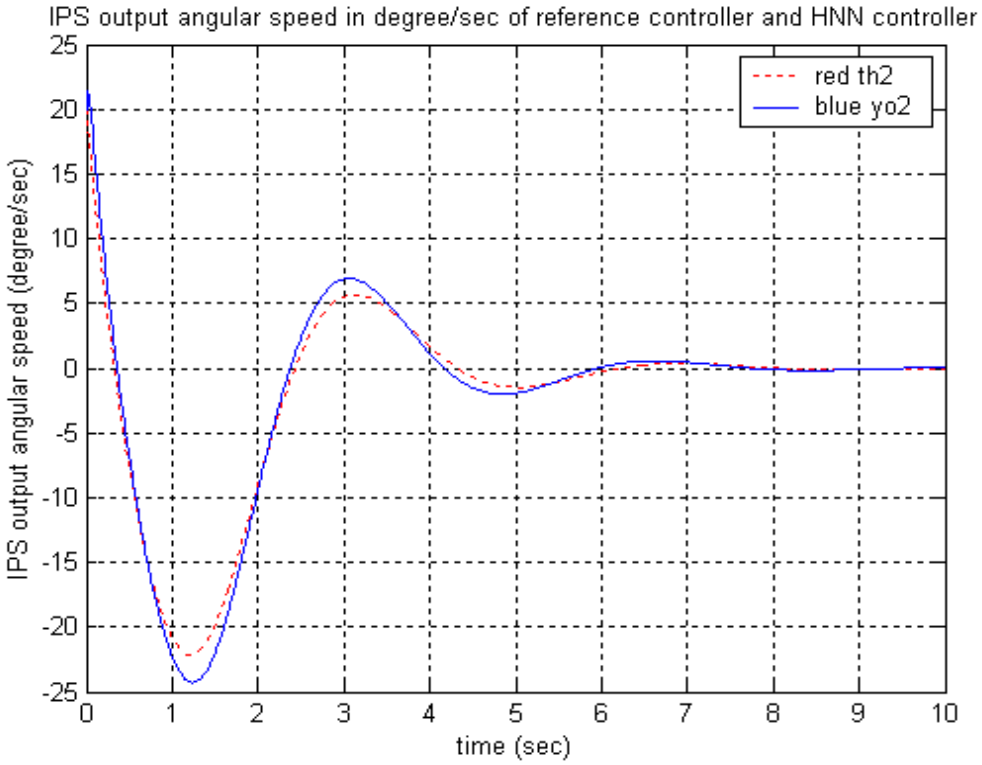

Fig-3.11. The output angular speeds of IPS: reference controller (dash line) and HNN controller (solid line)

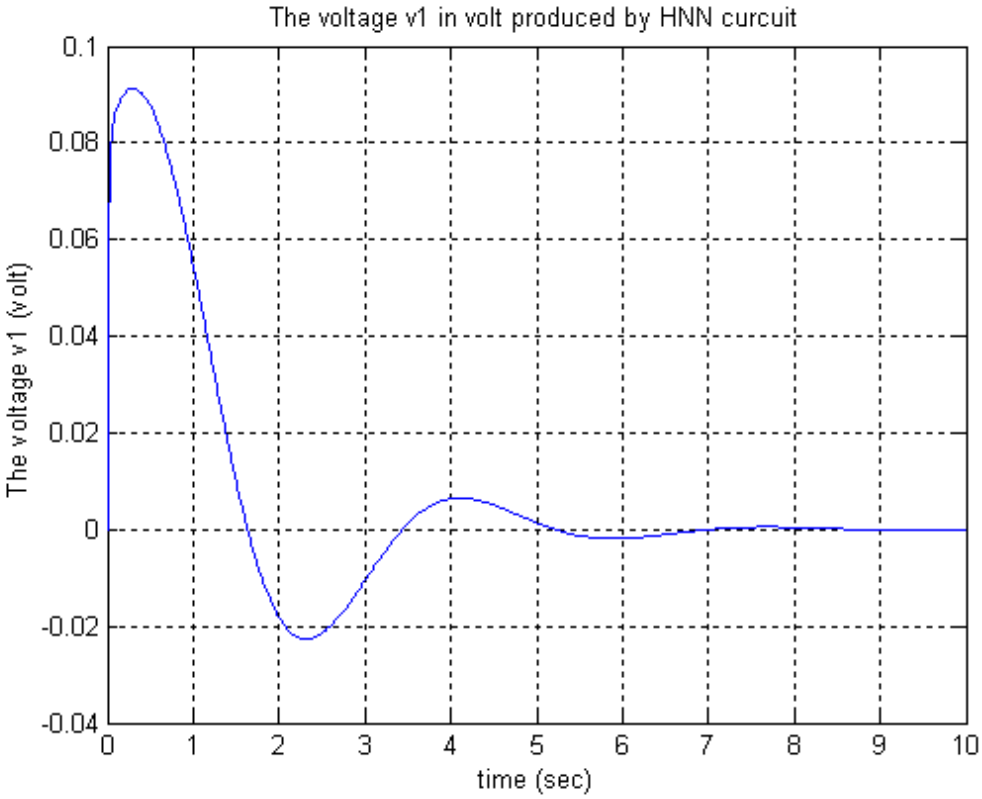

Fig-3.12. The node 1 voltage v_1 of the HNN circuit

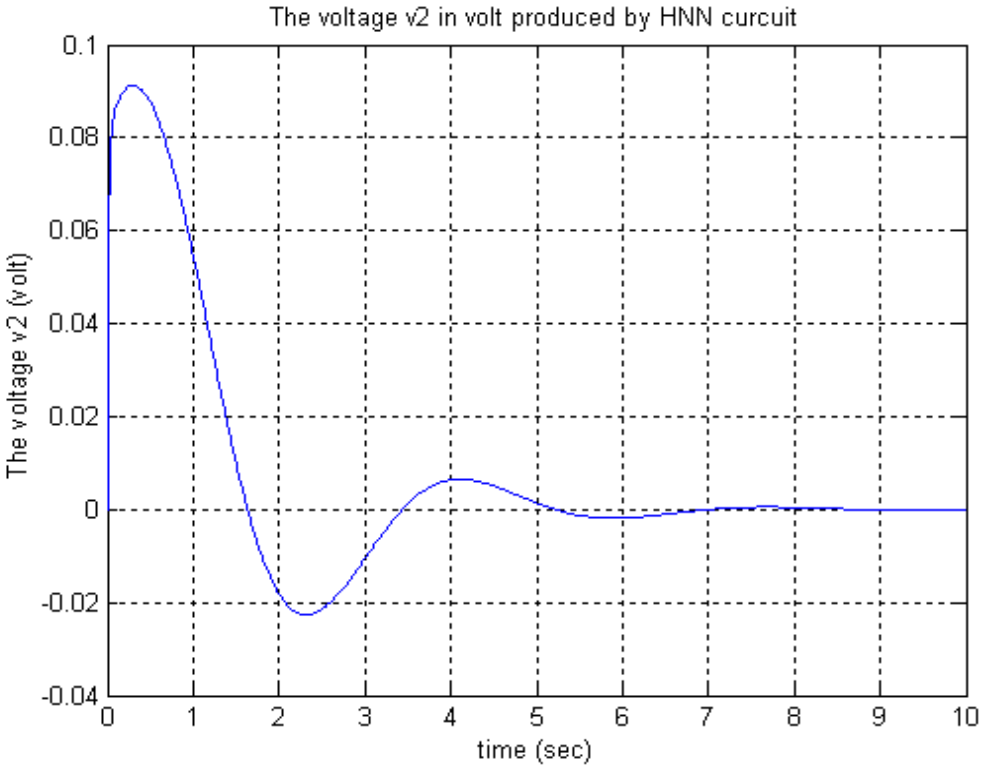

Fig-3.13. The node 2 voltage v_2 of the HNN circuit

3.4.3 IPS Controlled by HNN Controller Trained with Different Initial State

We use the following examples to examine the abilities of HNN controllers to control IPS with initial state different from the initial state of training the HNN controller. We can let the values of w_{11}, w_{12}, w_{21} and w_{22} be the same with section 3.4.1, so $(w_{11}, w_{12}, w_{21}, w_{22}) = (0.1755, 0.01367, 0.1755, 0.01367)$ is fixed for the initial state of IPS: the initial angle=20 (degree) and initial angular speed=20 (degree/sec) in the training phase. And then, we will examine the following pairs (initial angle (degree), initial angular speed (degree/sec)) of the initial state of IPS: (30, 20), (20, 30), (25, 25), and (10, 10) in the working phase as the following figures:

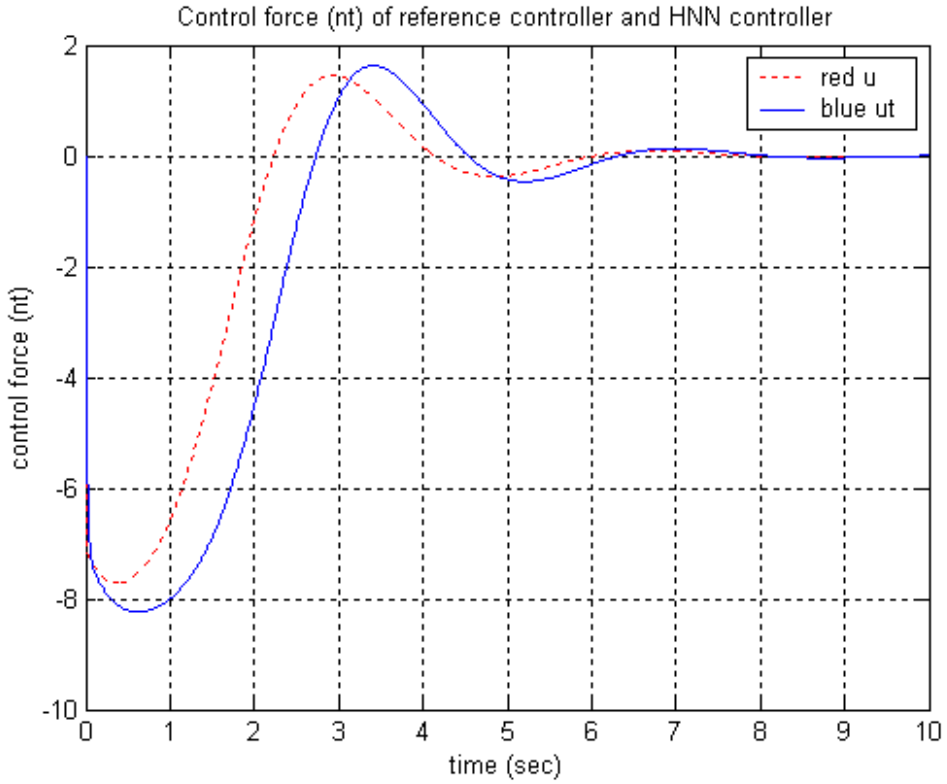

Fig-3.14. The control forces of IPS with initial angle=30 degree, initial angular speed=20 degree/sec: reference controller (dash line) and HNN controller (solid line)

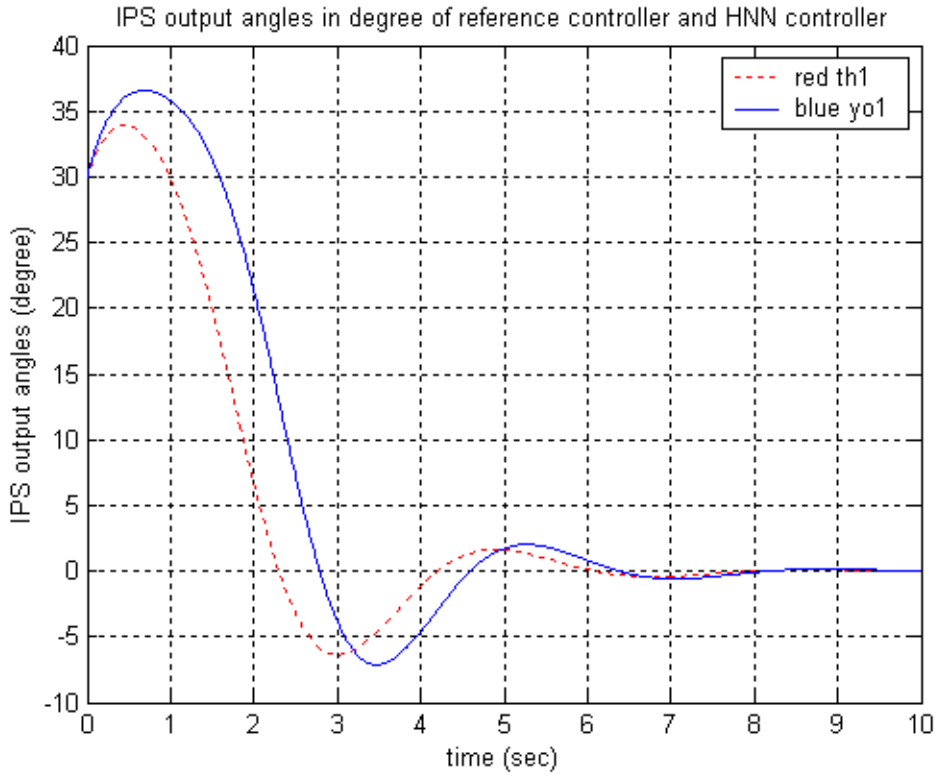

Fig-3.15. The output angles of IPS with initial angle=30 degree, initial angular speed=20 degree/sec: reference controller (dash line) and HNN controller (solid line)

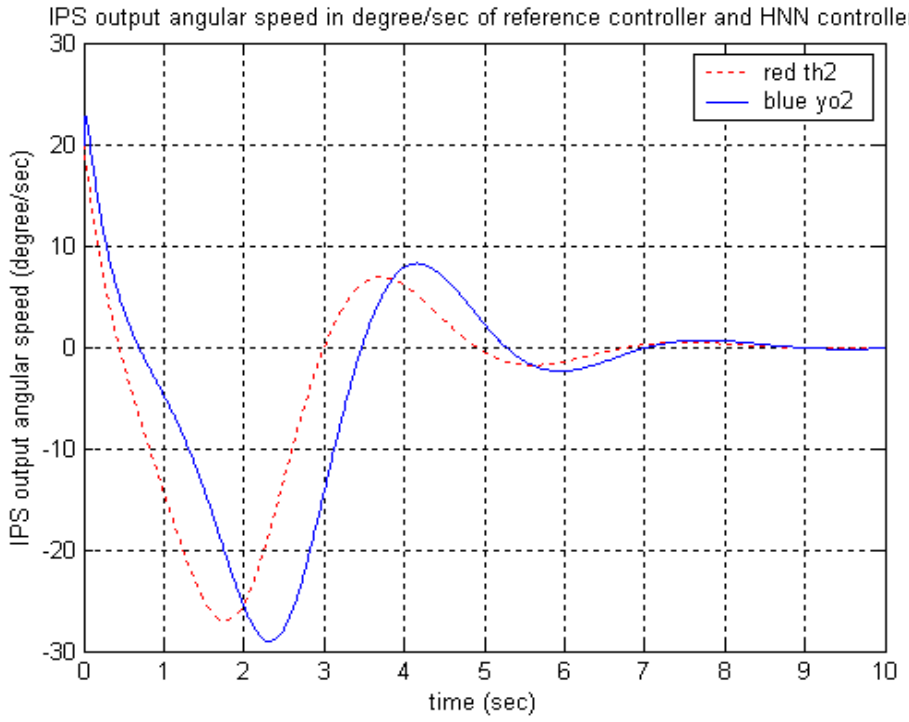

Fig-3.16. The output angular speeds of IPS with initial angle=30 degree, initial angular speed=20 degree/sec: reference controller (dash line) and HNN controller (solid line)

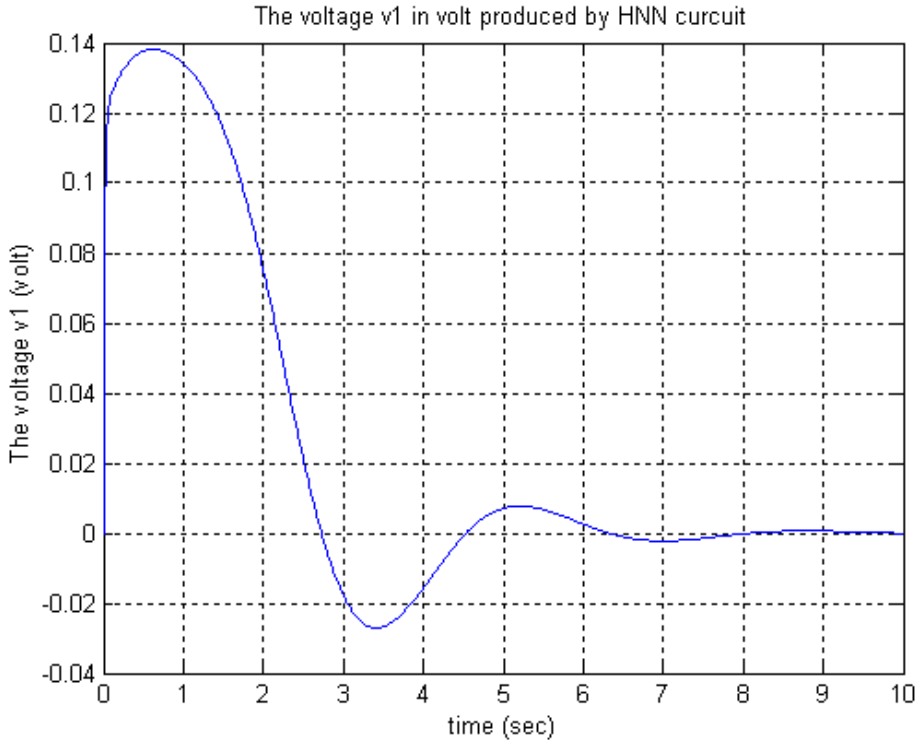

Fig-3.17. The node 1 (2) voltage v_1 (v_2) of the HNN circuit with IPS initial angle=30 degree, initial angular speed=20 degree/sec

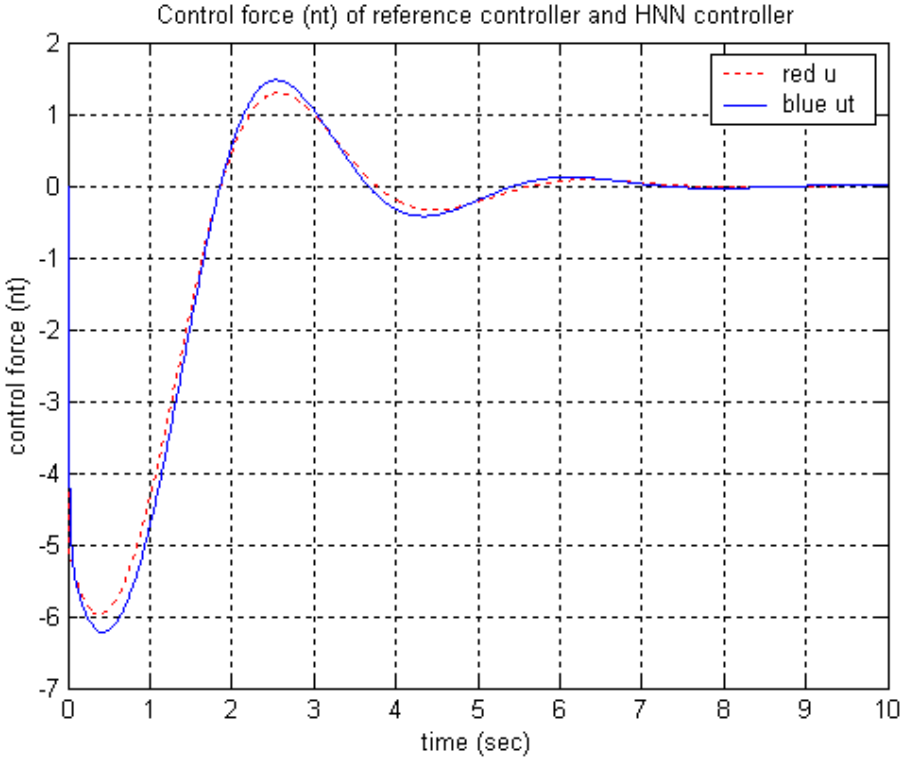

Fig-3.18. The control forces of IPS with initial angle=20 degree, initial angular speed=30 degree/sec: reference controller (dash line) and HNN controller (solid line)

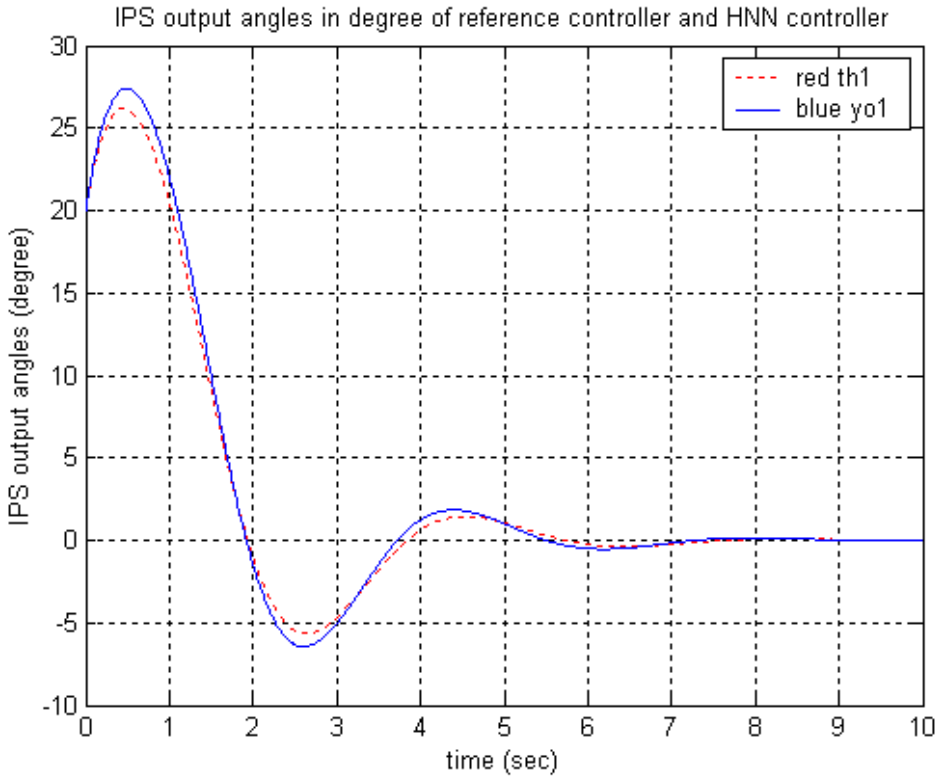

Fig-3.19. The output angles of IPS with initial angle=20 degree, initial angular speed=30 degree/sec: reference controller (dash line) and HNN controller (solid line)

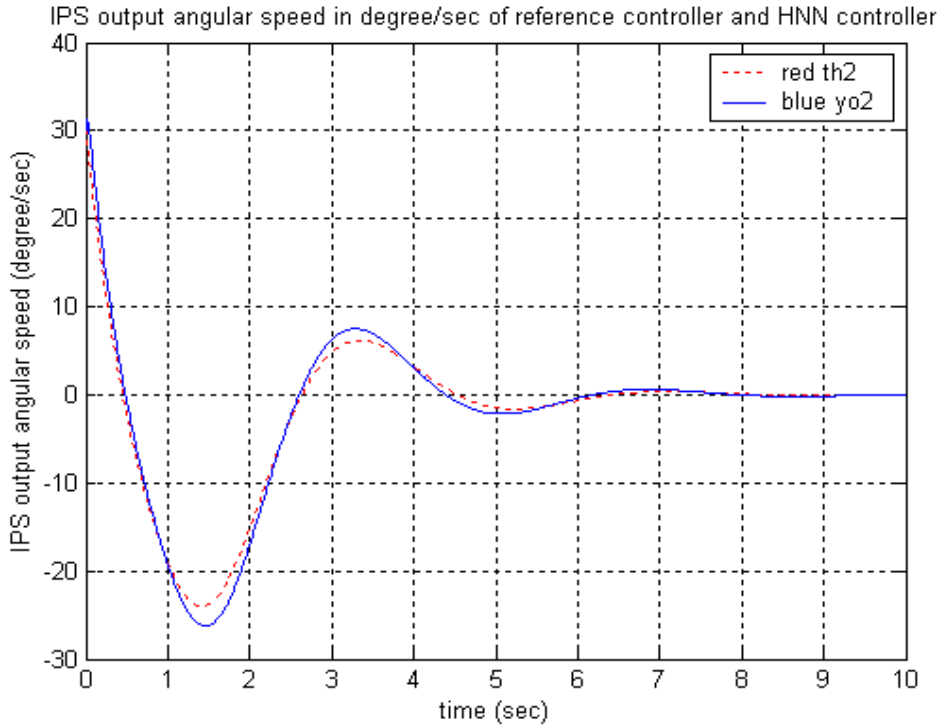

Fig-3.20. The output angular speeds of IPS with initial angle=20 degree, initial angular speed=30 degree/sec: reference controller (dash line) and HNN controller (solid line)

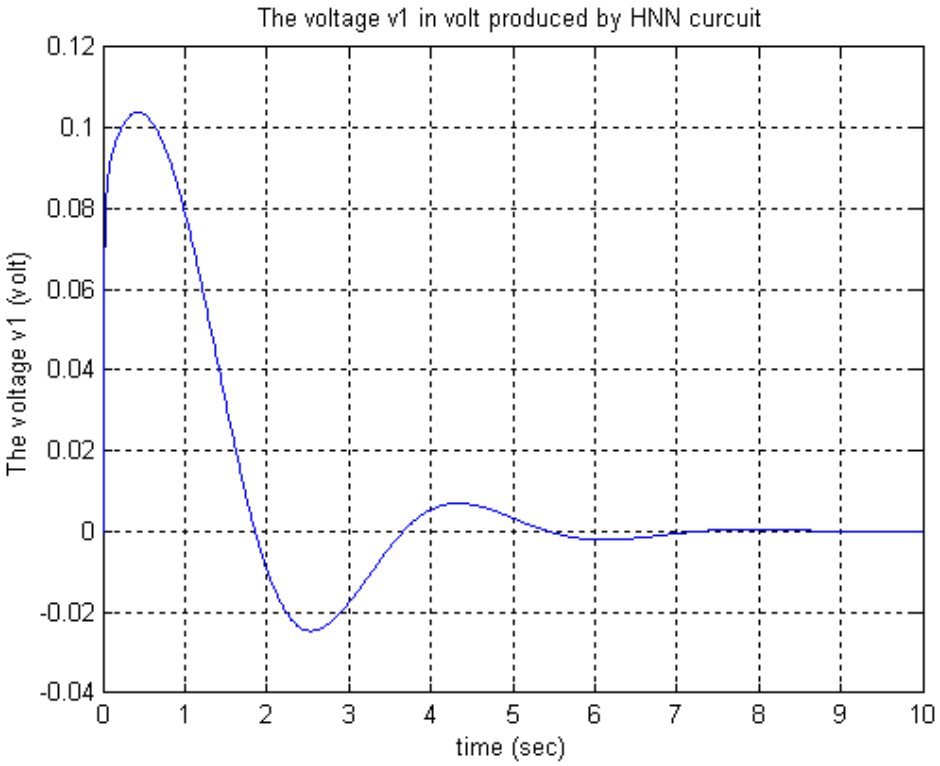

Fig-3.21. The node 1 (2) voltage v_1 (v_2) of the HNN circuit with IPS initial angle=20 degree, initial angular speed=30 degree/sec

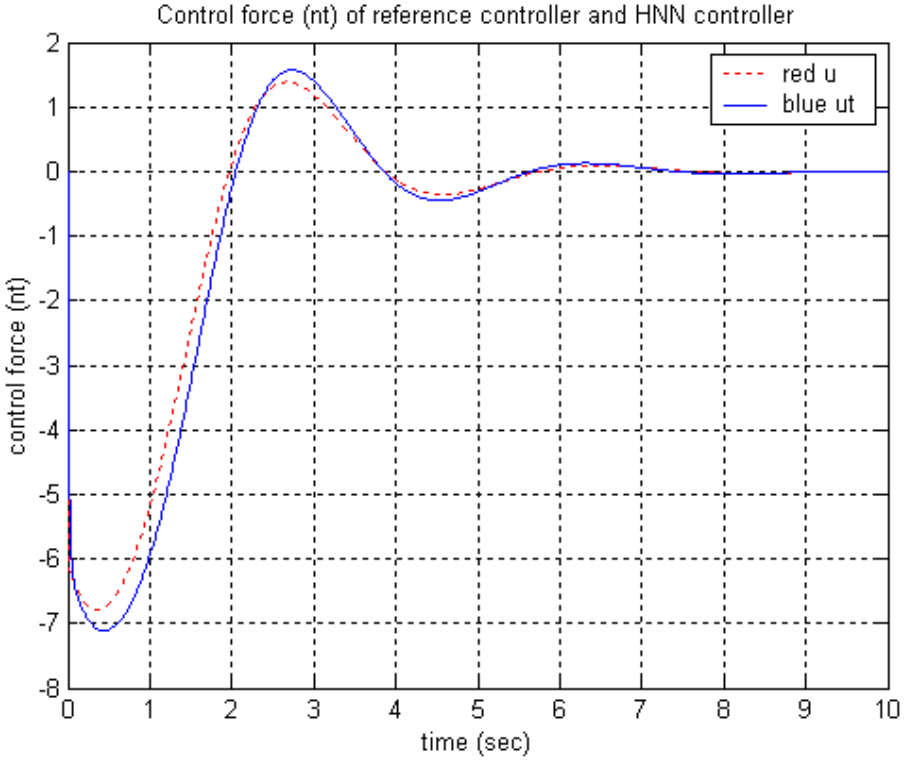

Fig-3.22. The control forces of IPS with initial angle=25 degree, initial angular speed=25 degree/sec: reference controller (dash line) and HNN controller (solid line)

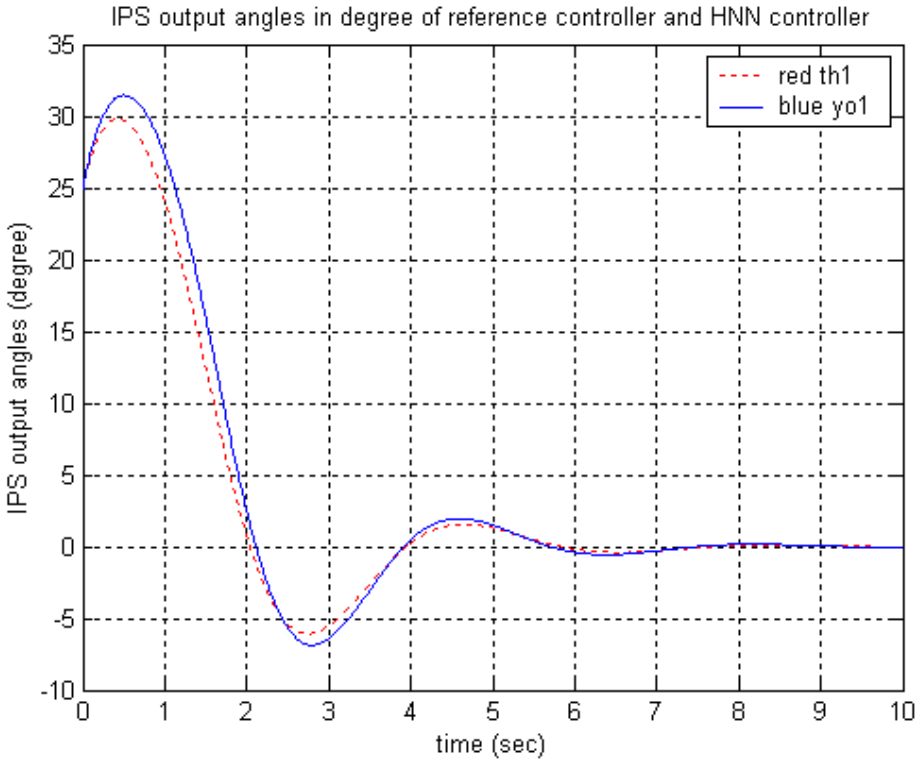

Fig-3.23. The output angles of IPS with initial angle=25 degree, initial angular speed=25 degree/sec: reference controller (dash line) and HNN controller (solid line)

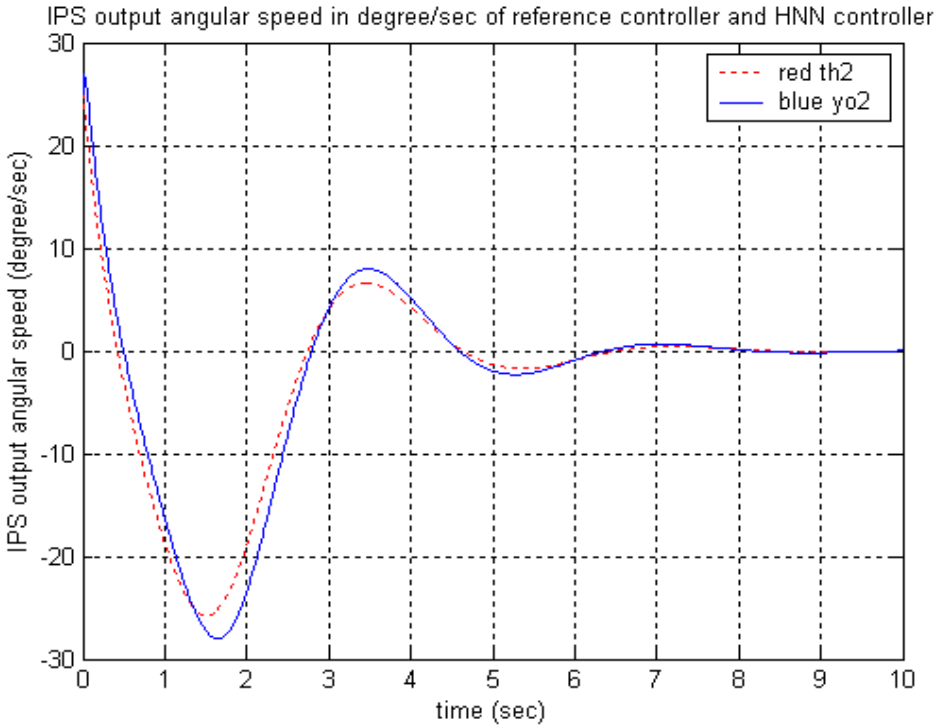

Fig-3.24. The output angular speeds of IPS with initial angle=25 degree, initial angular speed=25 degree/sec: reference controller (dash line) and HNN controller (solid line)

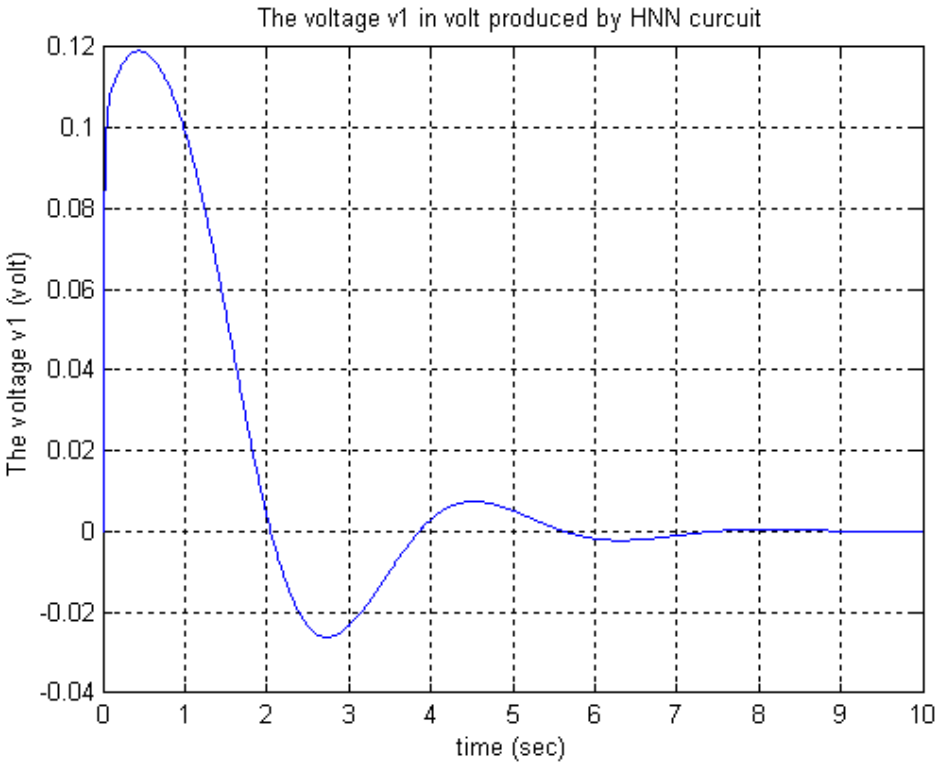

Fig-3.25. The node 1 (2) voltage v_1 (v_2) of the HNN circuit with IPS initial angle=25 degree, initial angular speed=25 degree/sec

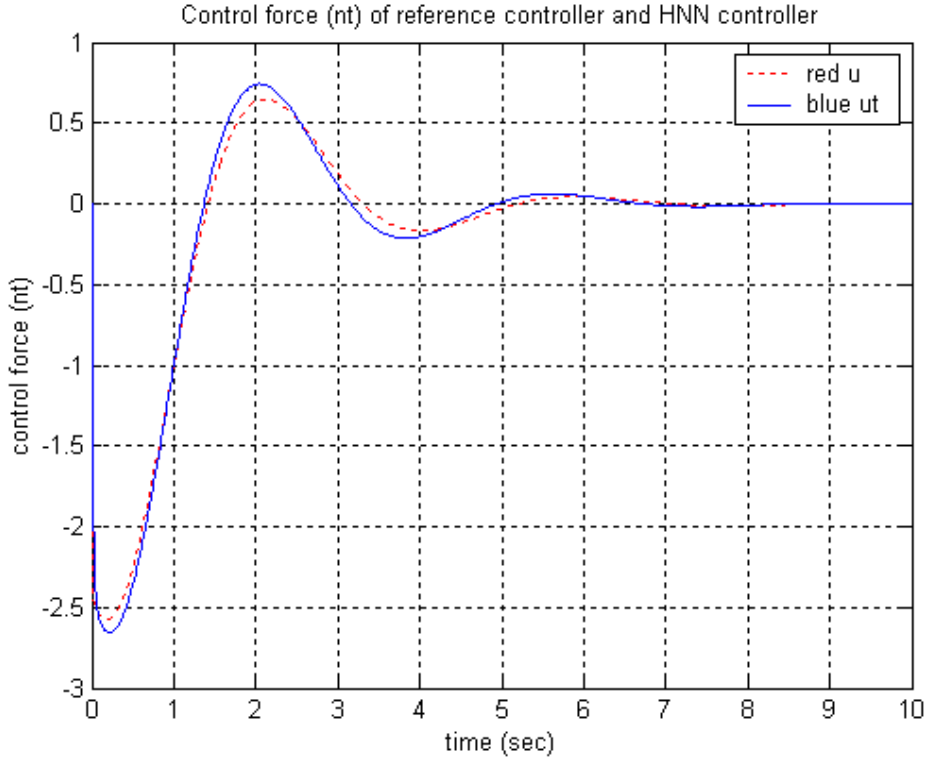

Fig-3.26. The control forces of IPS with initial angle=10 degree, initial angular speed=10 degree/sec: reference controller (dash line) and HNN controller (solid line)

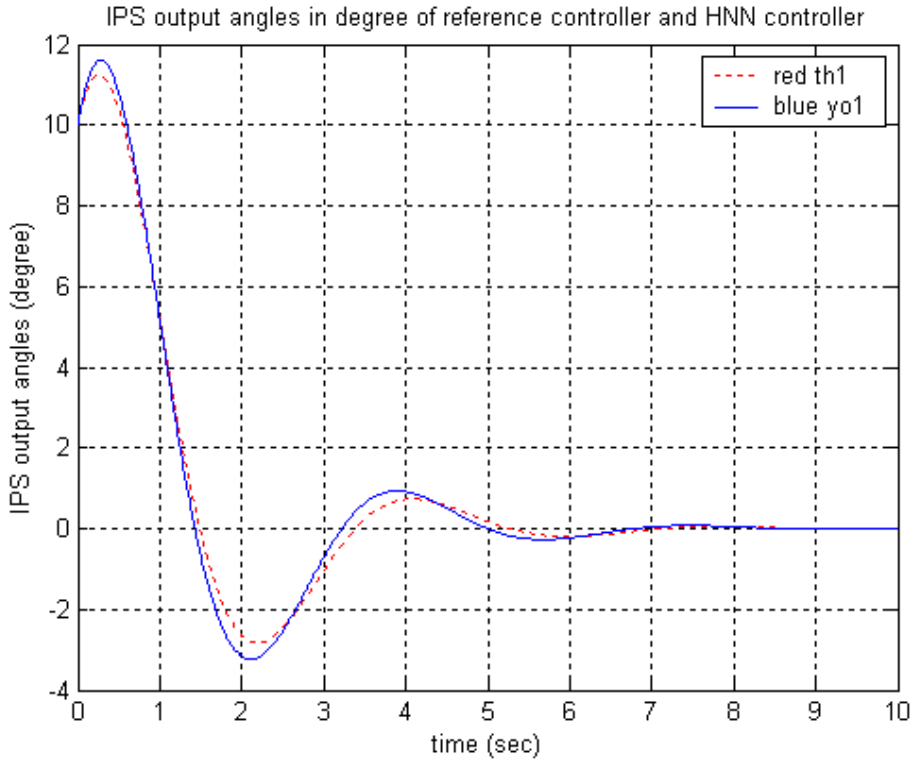

Fig-3.27. The output angles of IPS with initial angle=10 degree, initial angular speed=10 degree/sec: reference controller (dash line) and HNN controller (solid line)

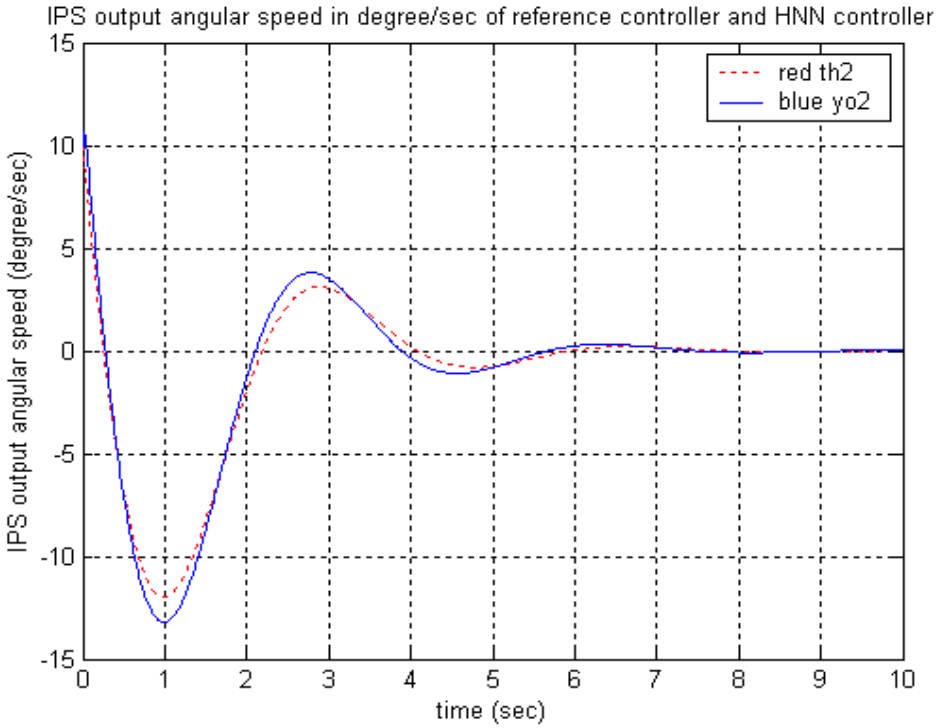

Fig-3.28. The output angular speeds of IPS with initial angle=10 degree, initial angular speed=10 degree/sec: reference controller (dash line) and HNN controller (solid line)

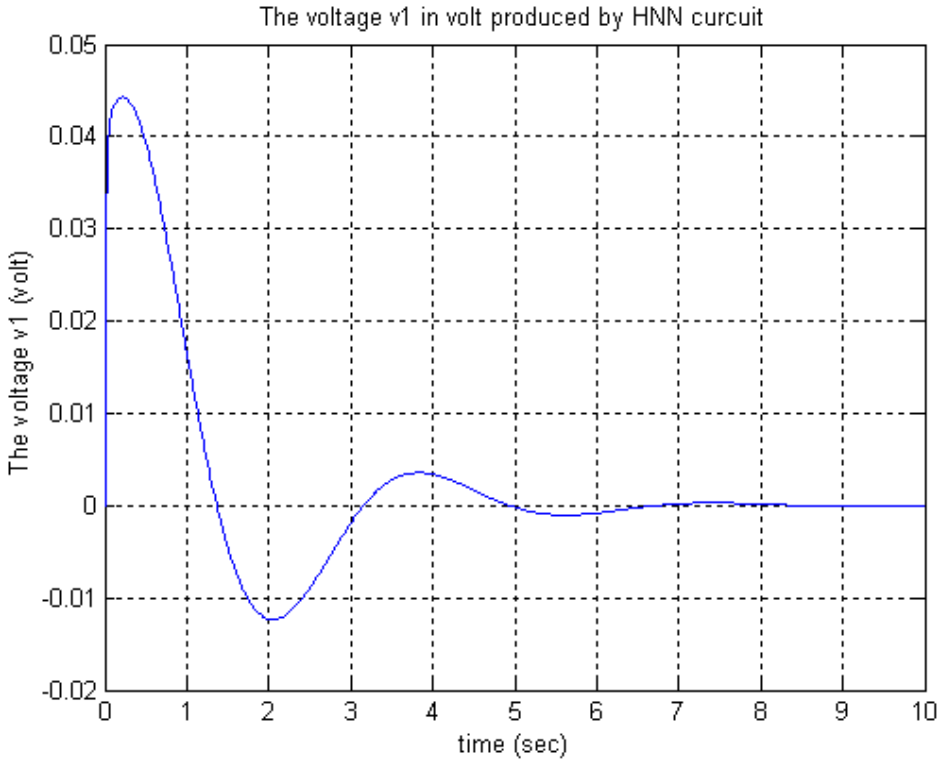

Fig-3.29. The node 1 (2) voltage v_1 (v_2) of the HNN circuit with IPS initial angle=10 degree, initial angular speed=10 degree/sec

3.4.4 IPS Controlled by HNN Controller Trained by Nonlinear Reference Controller with the Same Initial State

To examine the ability of HNN controller to mimic the nonlinear reference controller, we use the following equation as the nonlinear reference controller of IPS:

$$u = \frac{\dfrac{70}{(1+e^{\theta_1})(1+e^{\theta_2})} - \dfrac{70}{(1+e^{-\theta_1})(1+e^{-\theta_2})}}{\dfrac{1}{(1+e^{\theta_1})(1+e^{\theta_2})} + \dfrac{1}{(1+e^{\theta_1})(1+e^{-\theta_2})} + \dfrac{1}{(1+e^{-\theta_1})(1+e^{\theta_2})} + \dfrac{1}{(1+e^{-\theta_1})(1+e^{-\theta_2})}} \quad (3\text{-}29)$$

Let the IPS initial angle=20 degree and initial angular speed=20 degree/sec, and the simulation time=20 sec. We can get the following three figures of IPS controlled by the nonlinear reference controller:

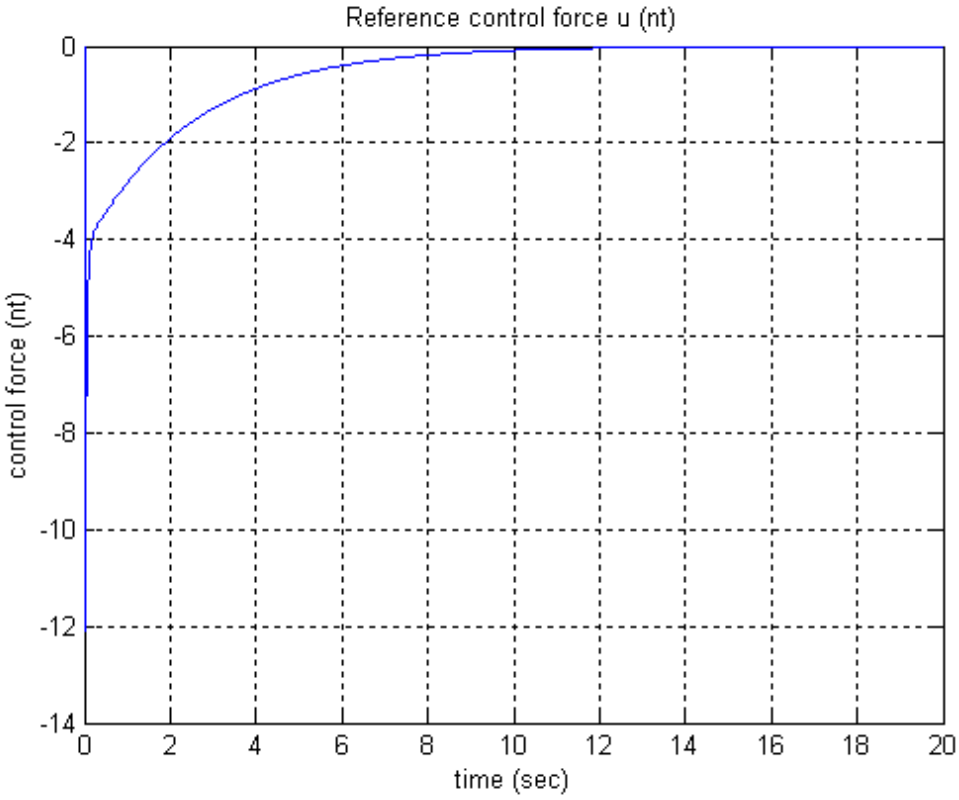

Fig-3.30. The control force of the nonlinear reference controller

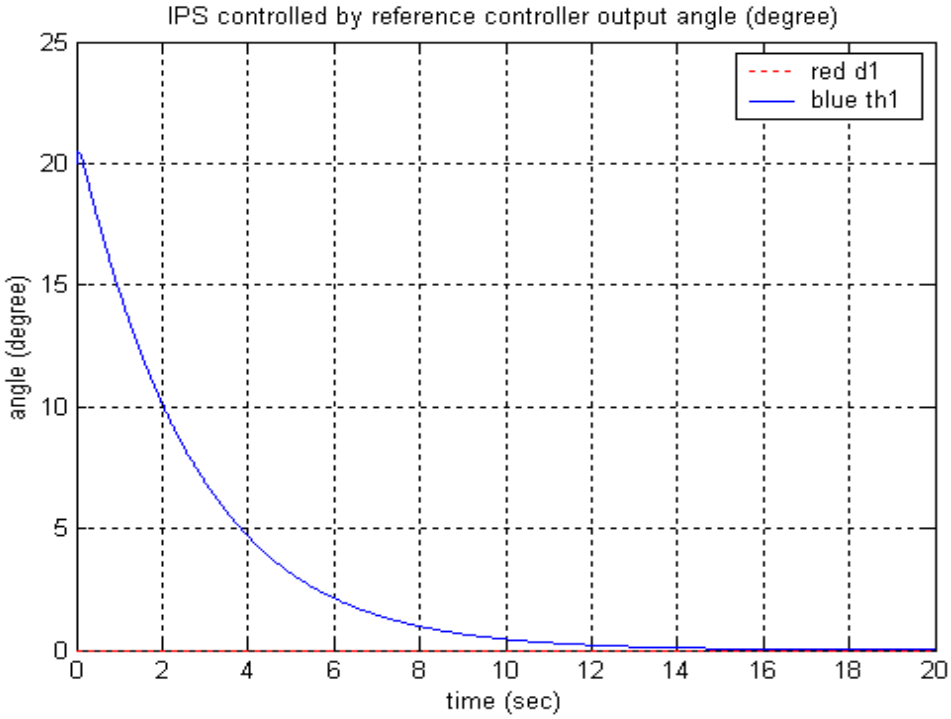

Fig-3.31. The output angle of IPS controlled by the nonlinear reference controller

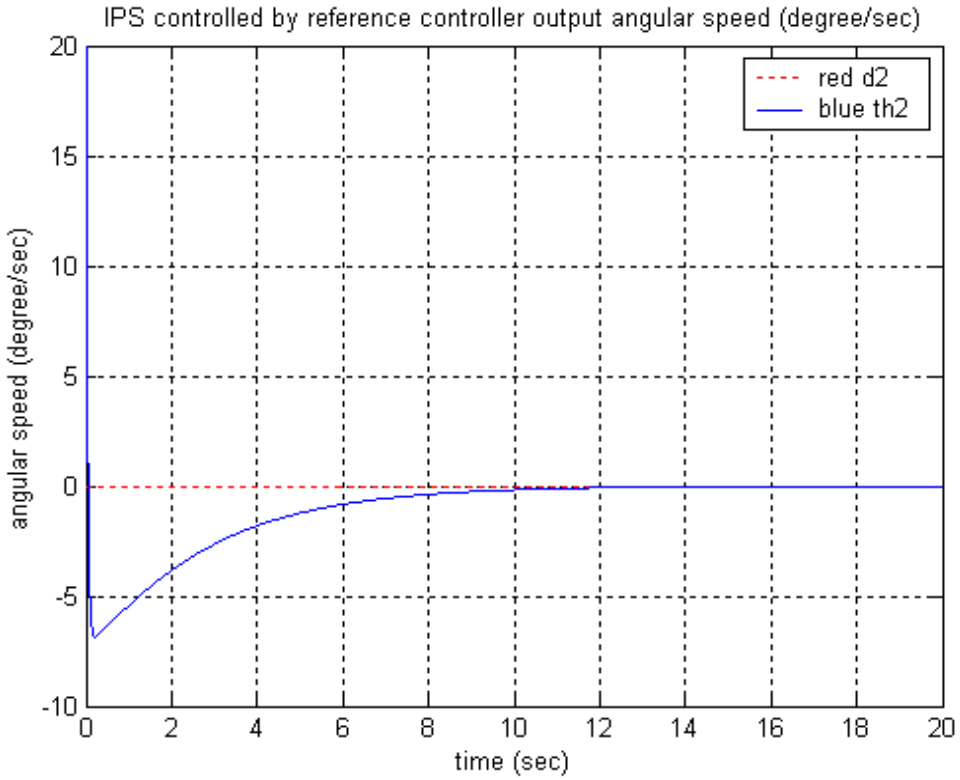

Fig-3.32. The output angular speed of IPS controlled by the nonlinear reference controller

We use the same parameters of HNN controller: the resistor=1(ohm), the capacitor=0.01(Farr), the amplification constant=-30, the learning rate=0.001, and 10 epochs for training, and we show the training process as the following two figures:

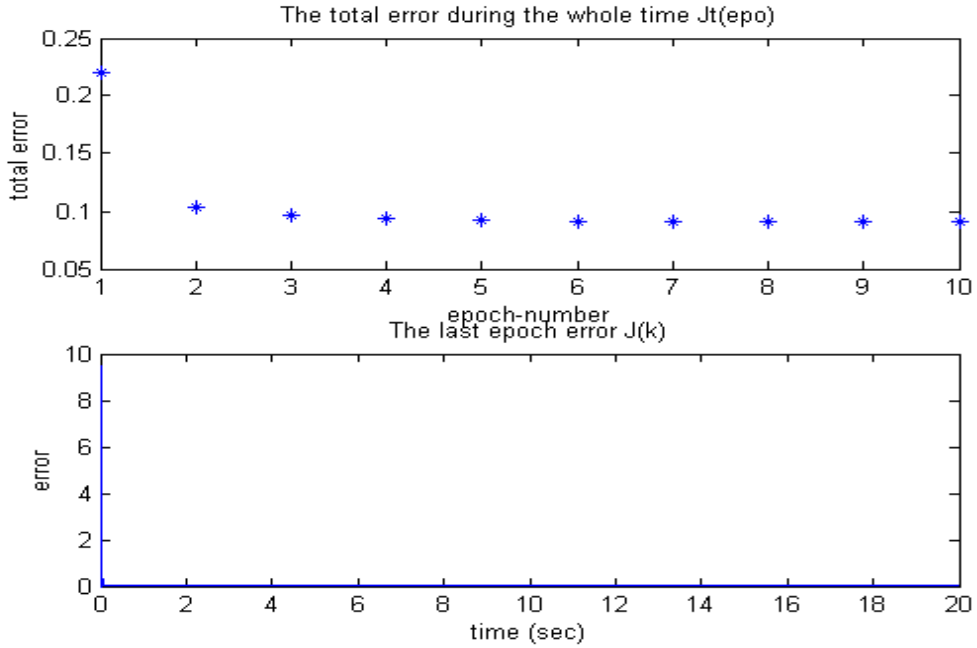

Fig-3.33. The total error $J_t(epo)$ and the last epoch error $J(k,10)$

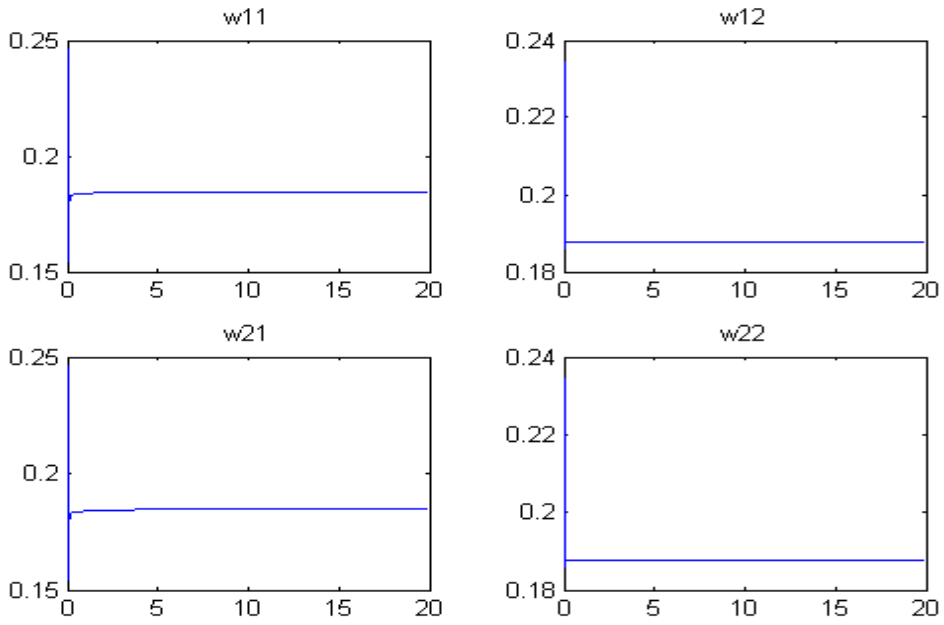

Fig-3.34. The last epoch training iteration process of w_{11}, w_{12}, w_{21} and w_{22}

By the simulation, we find the values of w_{11}, w_{12}, w_{21} and w_{22}, and we write it down as $(w_{11}, w_{12}, w_{21}, w_{22}) = (0.1846, 0.1876, 0.1846, 0.1876)$. Then, we show the following four figures as simulation result of IPS controlled by trained HNN controller:

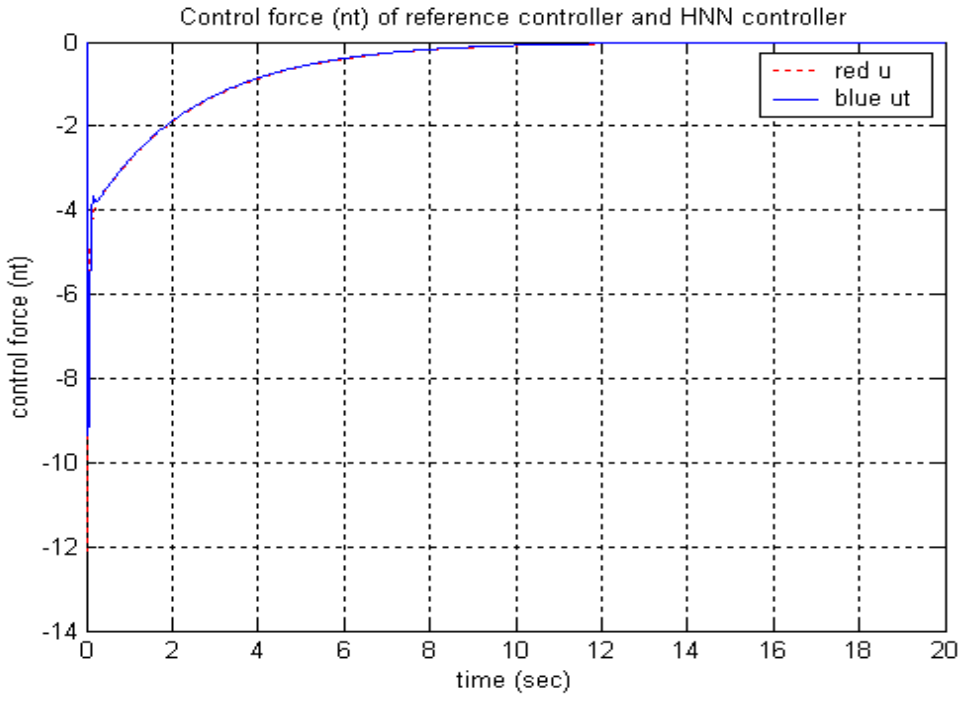

Fig-3.35. The control forces of IPS: nonlinear reference controller (dash line) and HNN controller (solid line)

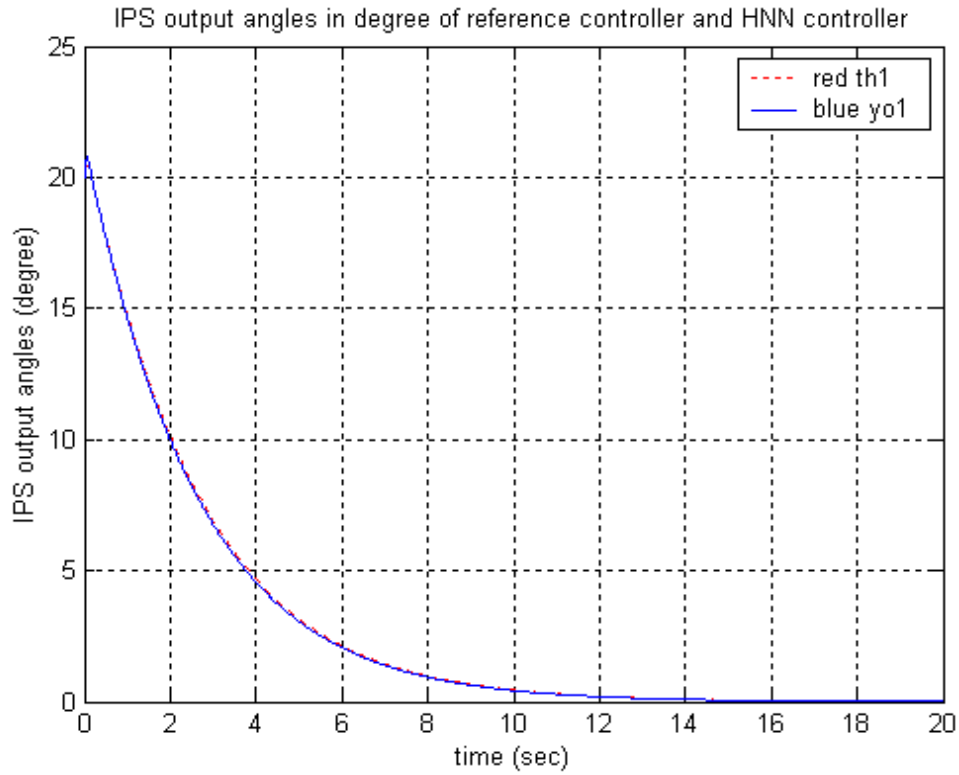

Fig-3.36. The output angles of IPS: reference controller (dash line) and HNN controller (solid line)

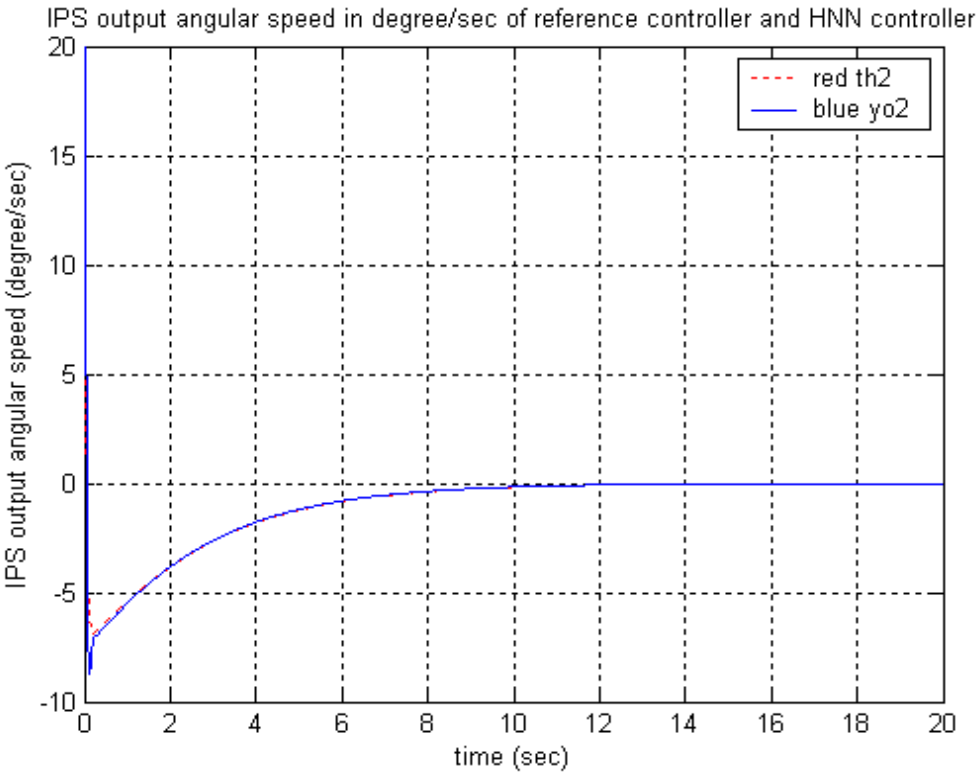

Fig-3.37. The output angular speeds of IPS: reference controller (dash line) and HNN controller (solid line)

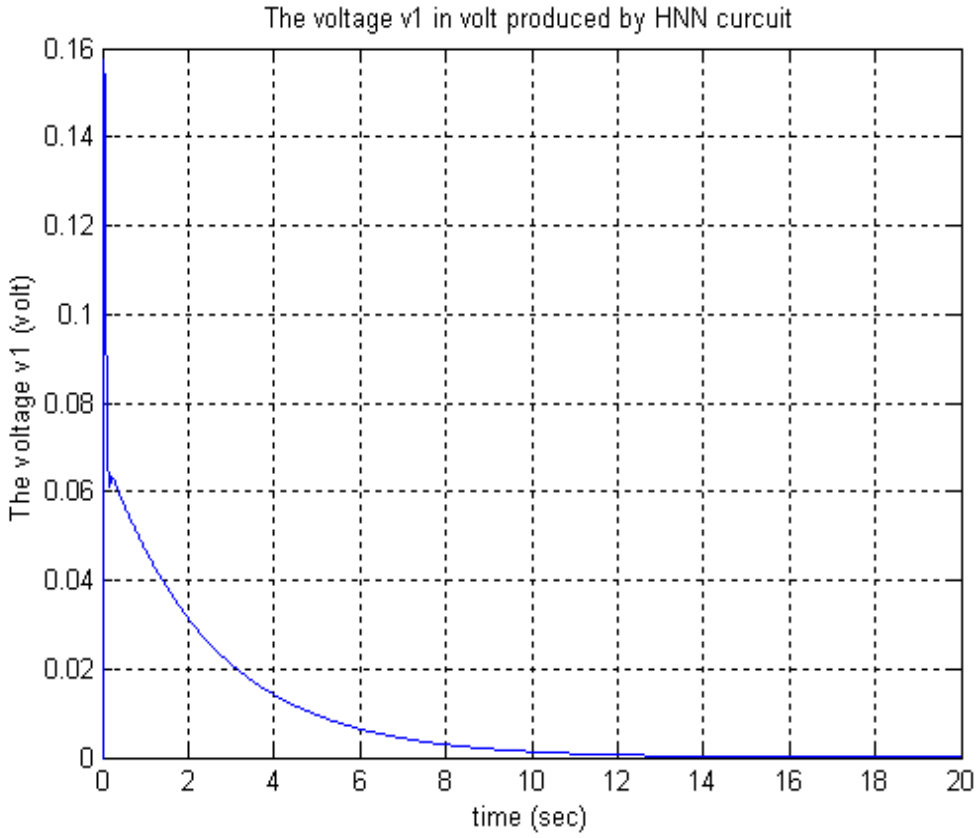

Fig-3.38. The node 1 (2) voltage v_1 (v_2) of the HNN circuit

3.4.5 IPS Controlled by HNN Controller Trained by Nonlinear Reference Controller with Different Initial State

We use the following examples to examine the abilities of HNN controllers to control IPS with initial state different from the initial state of training the HNN controller by the nonlinear reference controller. We let the values of w_{11}, w_{12}, w_{21} and w_{22} be the same with section 3.4.4, so $(w_{11}, w_{12}, w_{21}, w_{22}) = (0.1846, 0.1876, 0.1846, 0.1876)$ is fixed for the initial state of IPS: the initial angle=20 (degree) and initial angular speed=20 (degree/sec) in the training phase. And then, we will examine the following pairs (initial angle (degree), initial angular speed (degree/sec)) of the initial state of IPS: (30, 30), and (10, 10) in the working phase as the following figures:

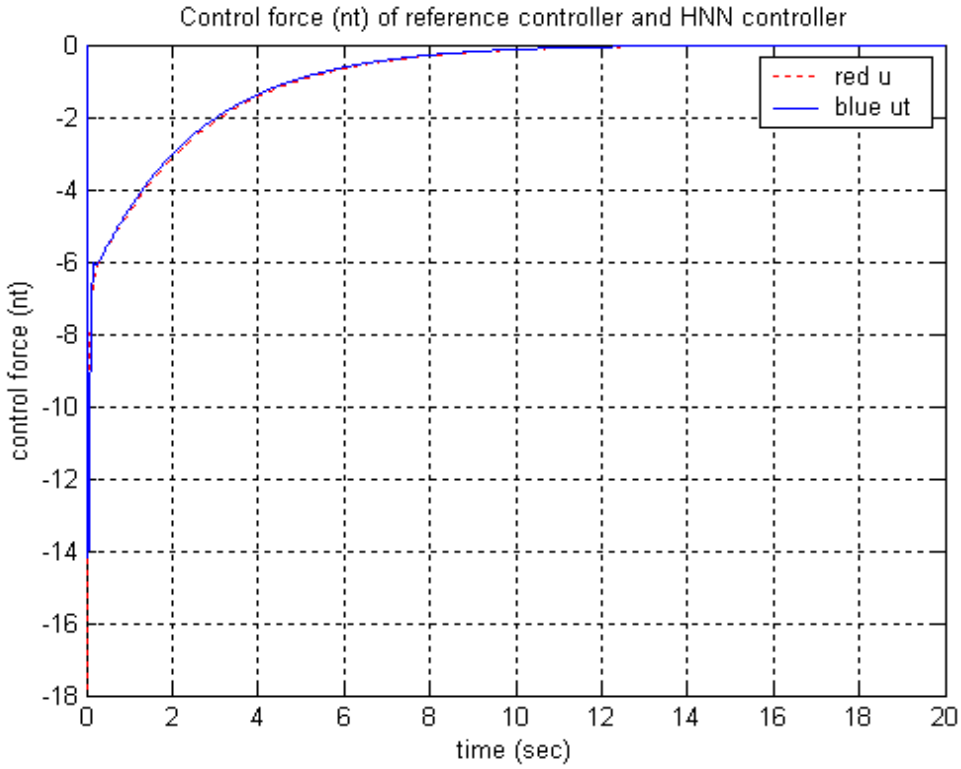

Fig-3.39. The control forces of IPS with initial angle=30 degree, initial angular speed=30 degree/sec: the nonlinear reference controller (dash line) and HNN controller (solid line)

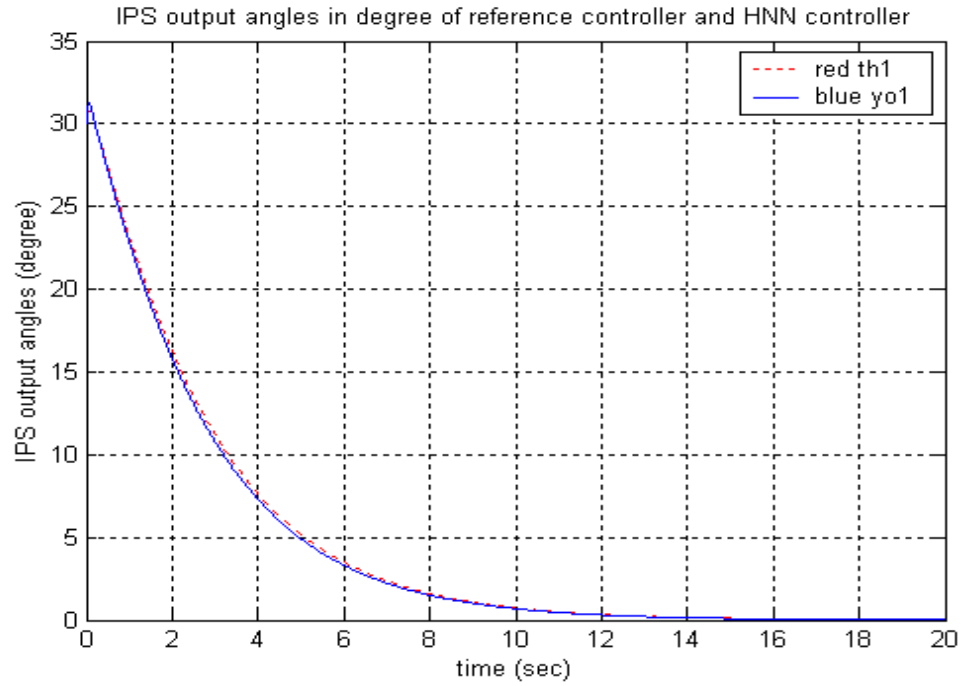

Fig-3.40. The output angles of IPS with initial angle=30 degree, initial angular speed=30 degree/sec: the nonlinear reference controller (dash line) and HNN controller (solid line)

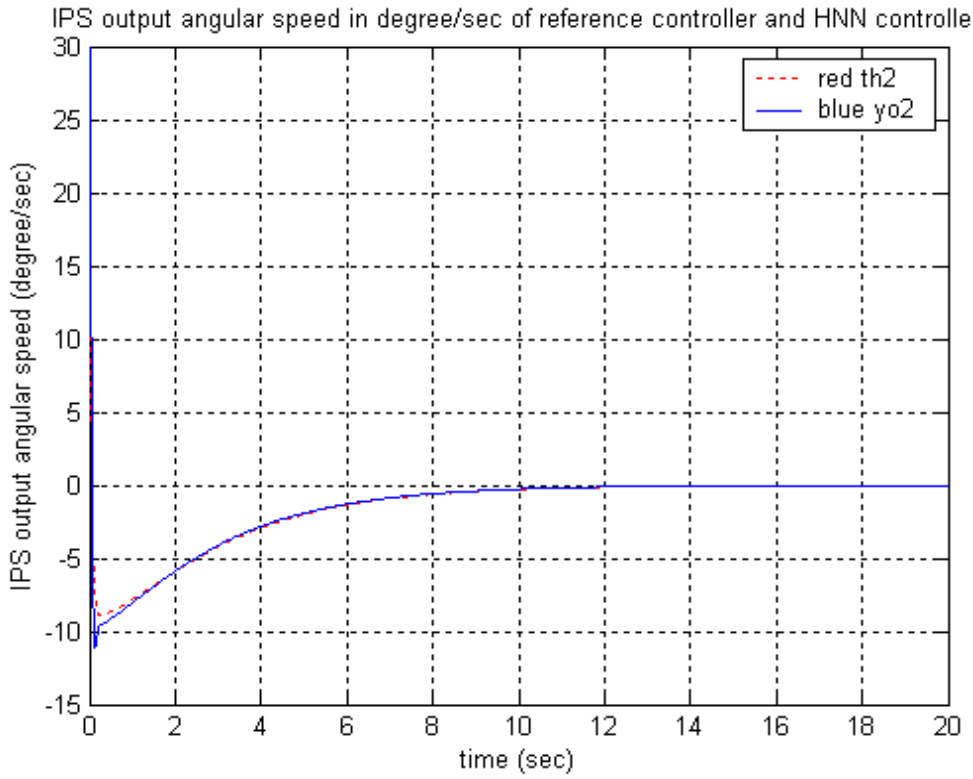

Fig-3.41. The output angular speeds of IPS with initial angle=30 degree, initial angular speed=30 degree/sec: the nonlinear reference controller (dash line) and HNN controller (solid line)

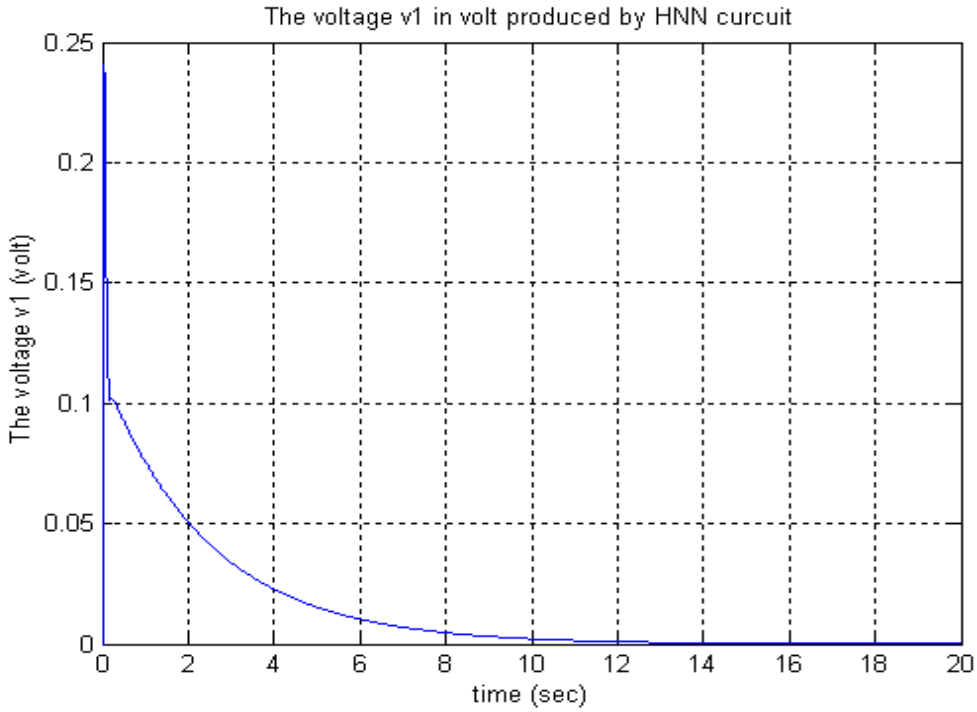

Fig-3.42. The node 1 (2) voltage v_1 (v_2) of the HNN circuit with IPS initial angle=30 degree, initial angular speed=30 degree/sec

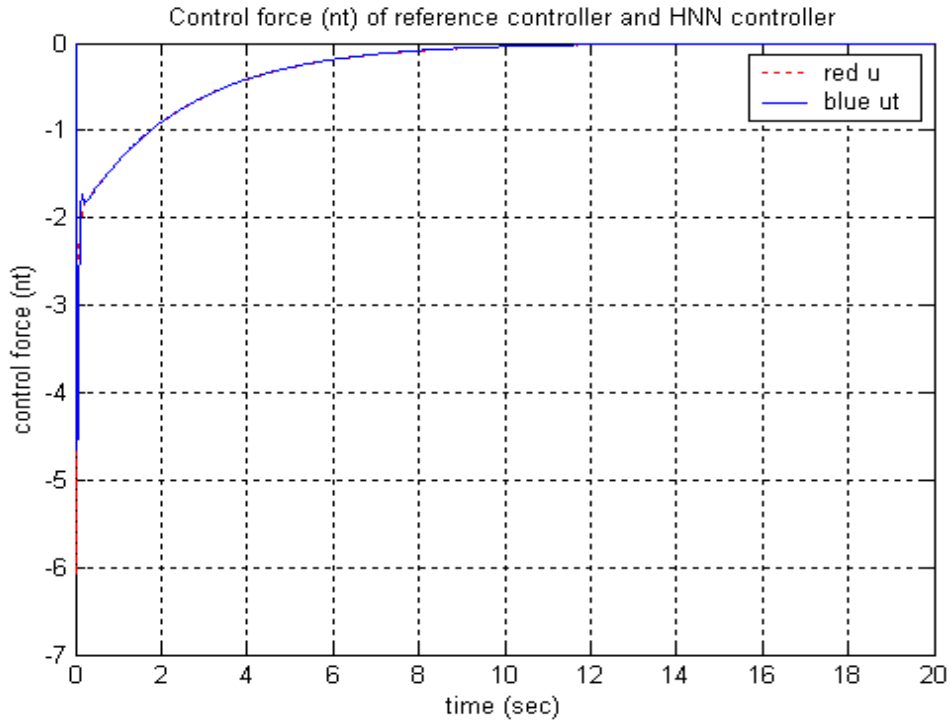

Fig-3.43. The control forces of IPS with initial angle=10 degree, initial angular speed=10 degree/sec: the nonlinear reference controller (dash line) and HNN controller (solid line)

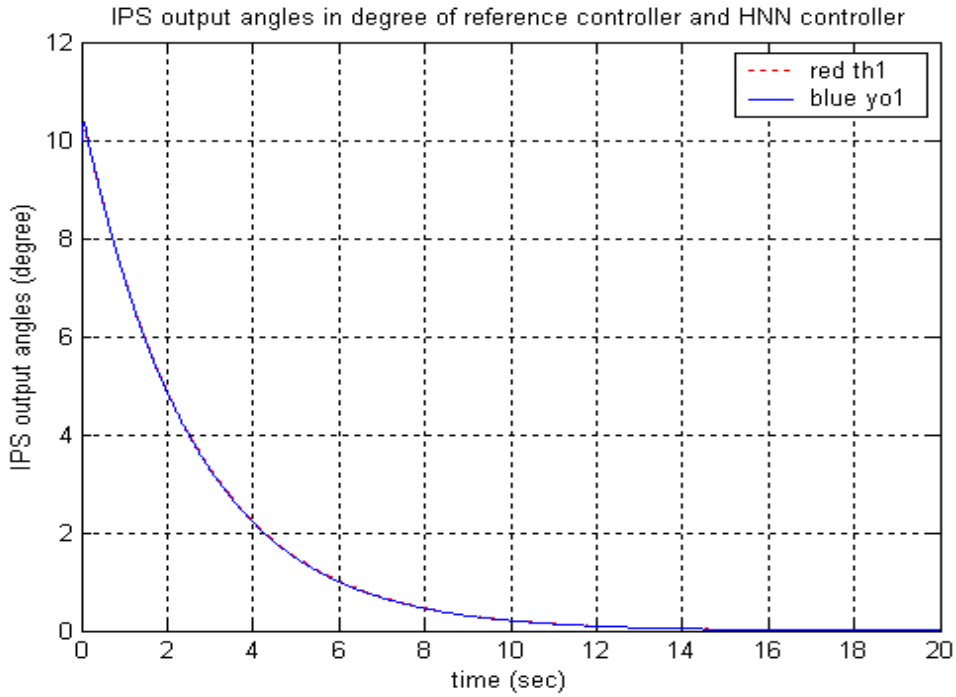

Fig-3.44. The output angles of IPS with initial angle=10 degree, initial angular speed=10 degree/sec: the nonlinear reference controller (dash line) and HNN controller (solid line)

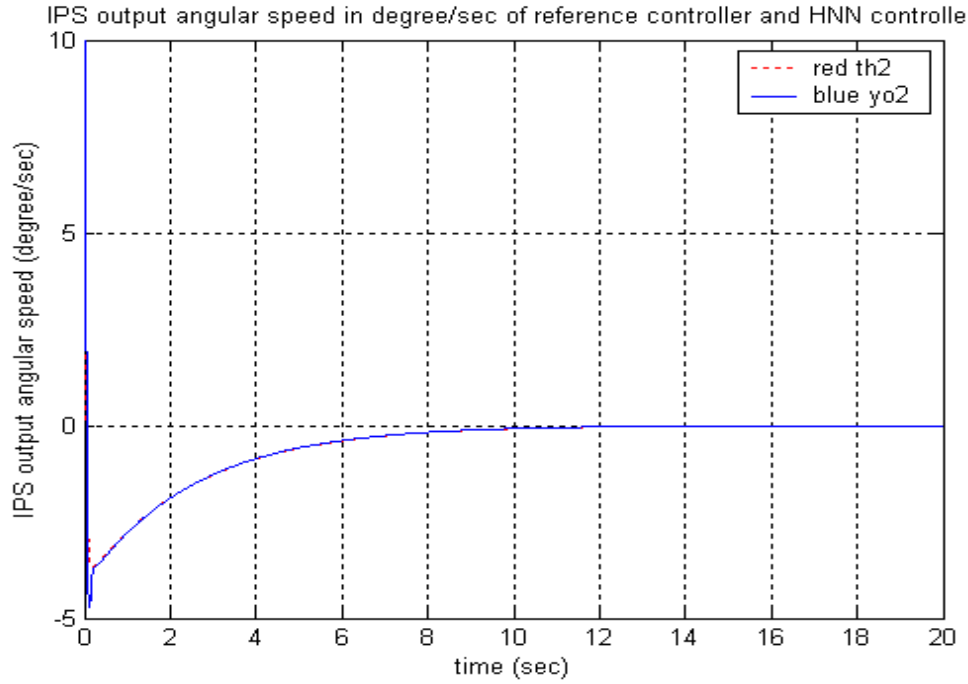

Fig-3.45. The output angular speeds of IPS with initial angle=10 degree, initial angular speed=10 degree/sec: the nonlinear reference controller (dash line) and HNN controller (solid line)

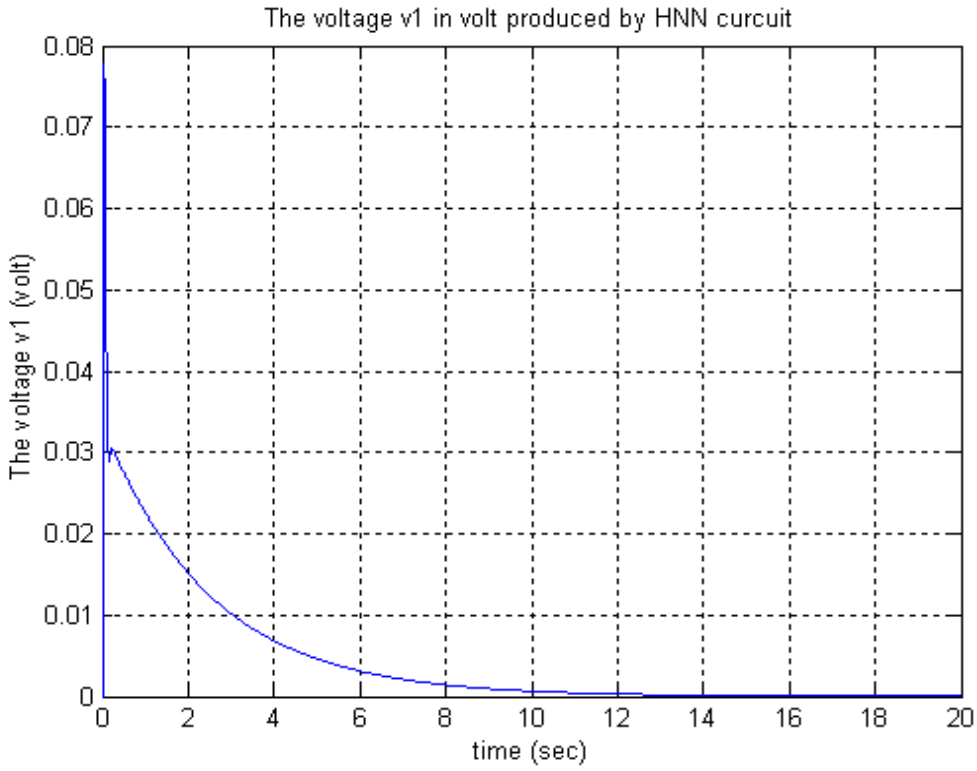

Fig-3.46. The node 1 (2) voltage v_1 (v_2) of the HNN circuit with IPS initial angle=10 degree, initial angular speed=10 degree/sec

Chapter 4

Control of Ball and Beam System (BABS)

4.1 Dynamics of Ball and Beam System (BABS)

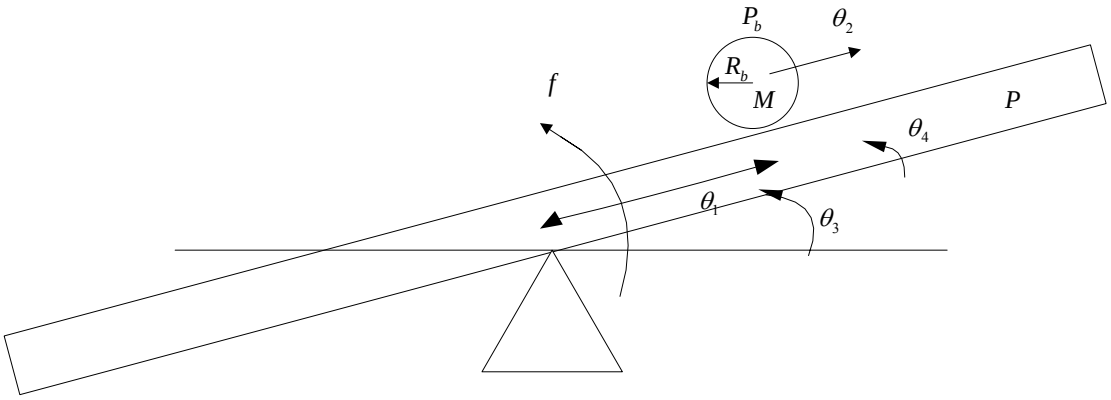

Fig-4.1. The ball and beam system (BABS)

In [22], the moment of inertia of the beam is P=0.02 (kg-m^2); the mass of the ball is M=0.05 (kg); the moment of inertia of the ball is $P_b = 2\times10^{-6}$ (kg-m^2); the radius of the ball is R=0.01(m); the gravity acceleration is G=9.8(m/s^2); the torque applied to the beam is f (nt-m). With the ball position θ_1 , the ball velocity θ_2 , the beam angle θ_3 , and the beam angular speed θ_4 , we can get the equations bellow:

$$(\frac{P_b}{R^2}+M)\times\dot{\theta}_2+M\times G\times\sin(\theta_3)-M\times\theta_1\times(\theta_4)^2=0 \quad (4\text{-}1)$$

$$[M\times(\theta_1)^2+P+P_b]\times\dot{\theta}_4+2\times M\times\theta_1\times\theta_2\times\theta_4+M\times G\times\theta_1\times\cos(\theta_3)=f \quad (4\text{-}2)$$

And we define $B=\frac{M}{M+\frac{P_b}{R^2}}$, so $B=\frac{5}{7}=0.7143$, and we can get the equations bellow:

$$\dot{\theta}_1=\theta_2 \quad (4\text{-}3)$$

$$\dot{\theta}_2=B\times[\theta_1\times(\theta_4)^2-G\times\sin(\theta_3)] \quad (4\text{-}4)$$

$$\dot{\theta}_3=\theta_4 \quad (4\text{-}5)$$

$$\dot{\theta}_4=\frac{-2\times M\times\theta_1\times\theta_2\times\theta_4-M\times G\times\theta_1\times\cos(\theta_3)+f}{M\times(\theta_1)^2+P+P_b} \quad (4\text{-}6)$$

We substitute the values of B, G, M, P, P_b above, and we can get the following equations:

$$\begin{aligned}
\dot{\theta}_1 &= \theta_2 \\
\dot{\theta}_2 &= \frac{5}{7} \times [\theta_1 \times (\theta_4)^2 - 9.8 \times \sin(\theta_3)] \\
\dot{\theta}_3 &= \theta_4 \\
\dot{\theta}_4 &= \frac{-2 \times 0.05 \times \theta_1 \times \theta_2 \times \theta_4 - 0.05 \times 9.8 \times \theta_1 \times \cos(\theta_3) + f}{0.05 \times (\theta_1)^2 + 0.02 + 0.000002}
\end{aligned} \tag{4-7}$$

4.2 Simulation of BABS Controlled by Reference Controller

We use the following equation as reference controller of BABS:

$$f = 0.7 \times \theta_1 + 0.7 \times \theta_2 - 3 \times \theta_3 - 1.5 \times \theta_4 , \tag{4-8}$$

where f is the control torque. Let the BABS initial position (ball's position) =0.2 meter, initial velocity (ball's velocity) =0 meter/sec, initial angle (beam's angle) =10 degree and initial angular speed (beam's angular speed) =0 degree/sec, and the simulation time=15sec. With equation (4-7) and (4-8), we can get the results of the BABS controlled by the reference controller, and we can show them in the following figures.

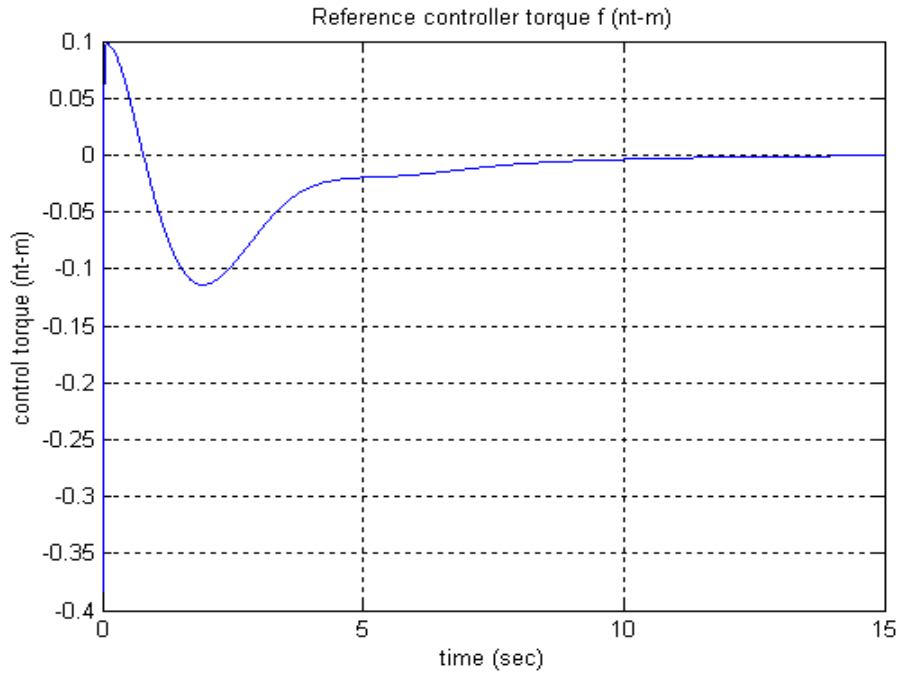

Fig-4.2. The control torque of the reference controller

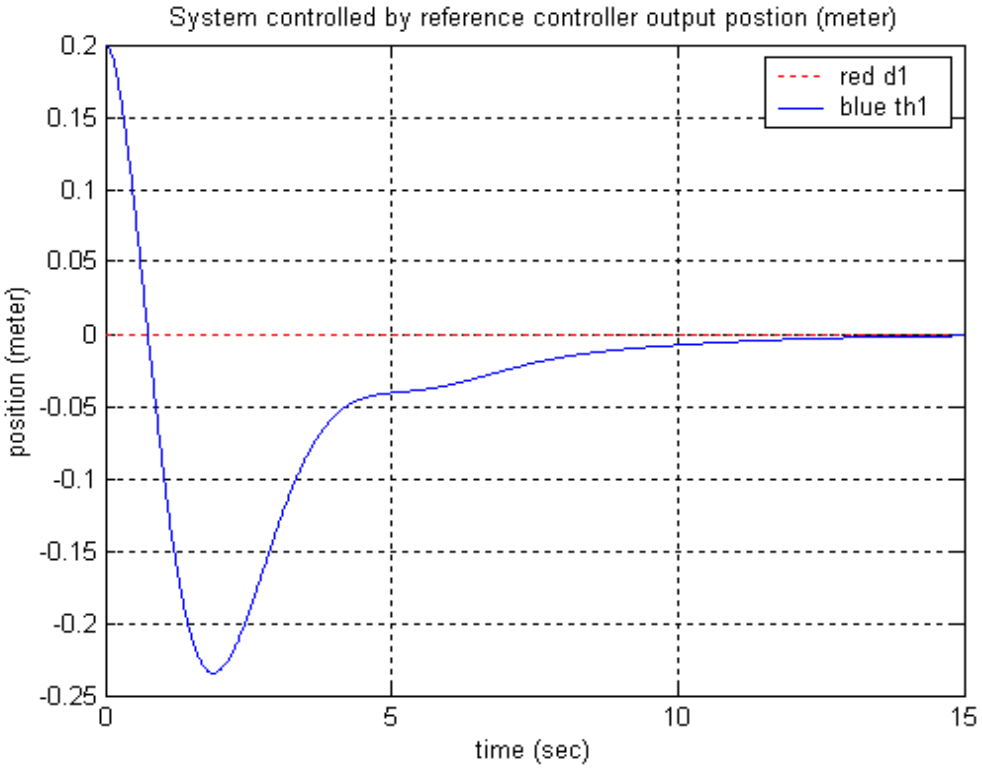

Fig-4.3. The ball's position of BABS controlled by the reference controller

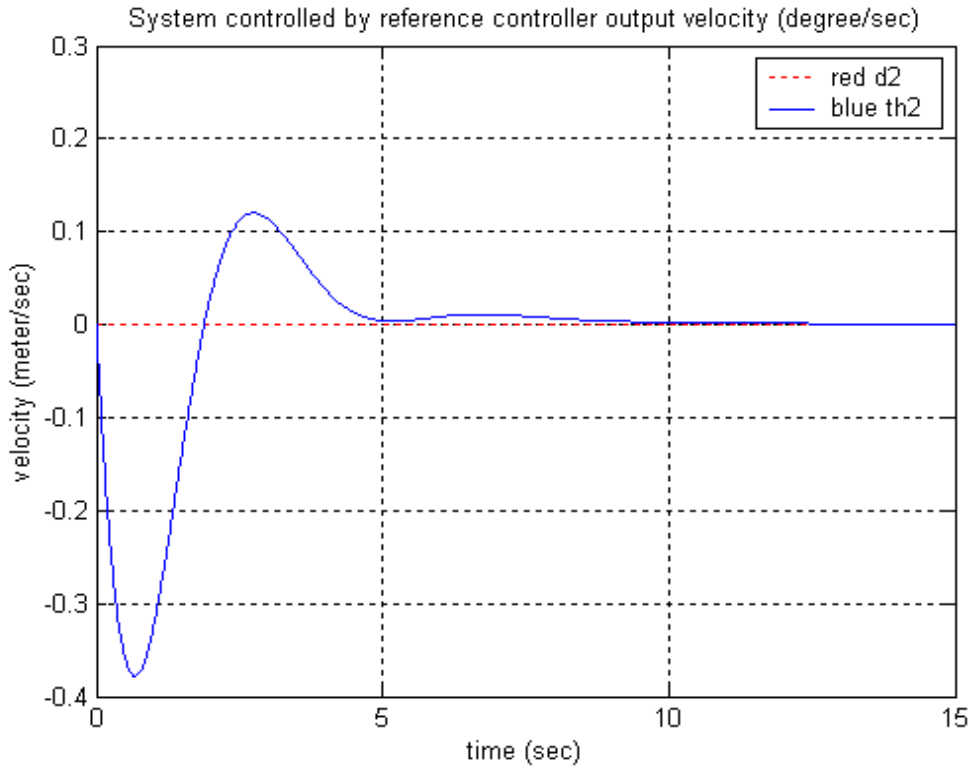

Fig-4.4. The ball's velocity of BABS controlled by the reference controller

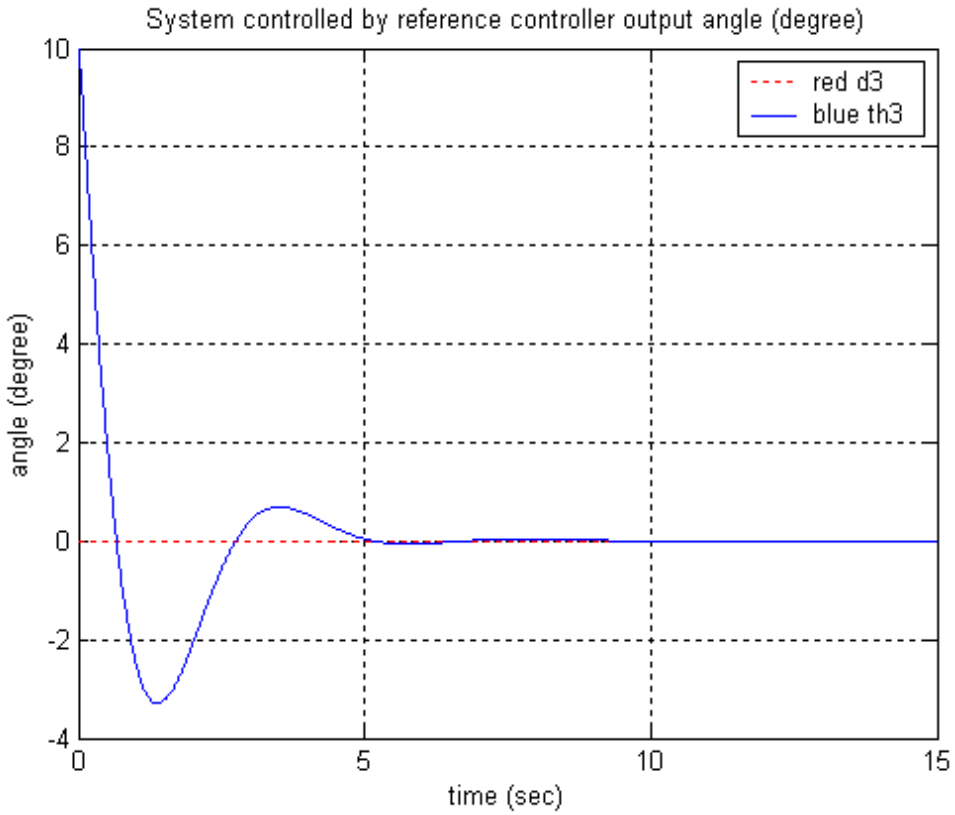

Fig-4.5. The beam's angle of BABS controlled by the reference controller

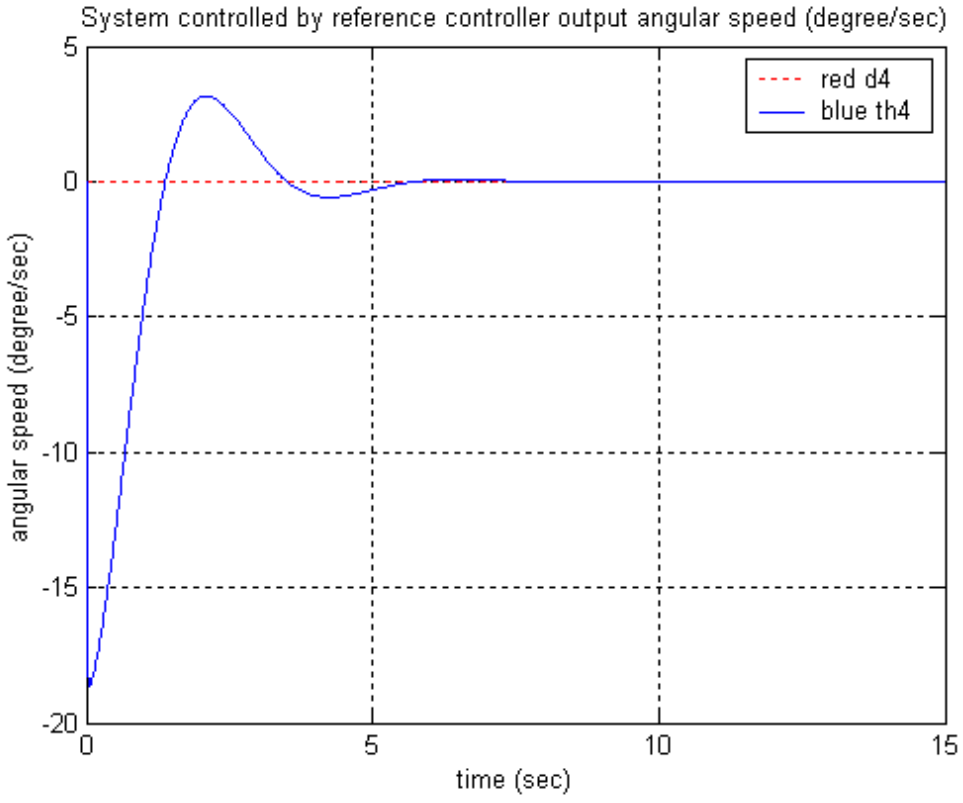

Fig-4.6. The beam's angular speed of BABS controlled by the reference controller

4.3 Architecture and Algorithm of Weighting Training

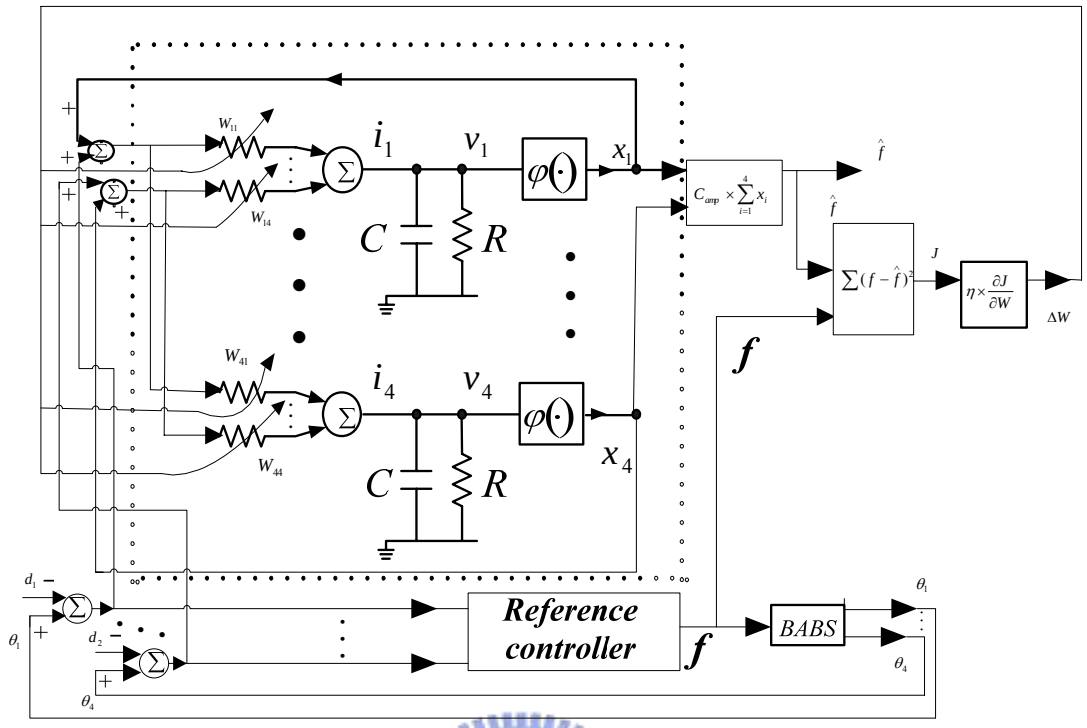

Fig-4.7. The architecture of the BABS controlled by the HNN controller in the training phase

We use the following parameters of HNN controller: the resistor=1(ohm), the capacitor=0.01(Farr), the amplification constant=-30, and the learning rate=0.001.

In the training phase, we note that the Hopfield neural network (HNN) doesn't control the ball and beam system (BABS). So we can imagine that when train w_{11}, w_{12}, w_{13}, w_{14}, w_{21}, w_{22}, w_{23}, w_{24}, w_{31}, w_{32}, w_{33}, w_{34}, w_{41}, w_{42}, w_{43} and w_{44} by the set of $(u,\theta_1,d_1,\theta_2,d_2,\theta_3,d_3,\theta_4,d_4)$(the value of the set is corresponding to this moment time t_1), the time of the reference controller and the BABS pauses until the HNN circuit is in the steady state. And we continue to train w_{11}, w_{12}, w_{13}, w_{14}, w_{21}, w_{22}, w_{23}, w_{24}, w_{31}, w_{32}, w_{33}, w_{34}, w_{41}, w_{42}, w_{43} and w_{44} by the next set of $(u,\theta_1,d_1,\theta_2,d_2,\theta_3,d_3,\theta_4,d_4)$ (the value of the "next set" is corresponding to the next moment time t_2).

So we can consider that in the training phase, the circuit is already in the steady state, so no current passes to the four capacitors C (it equals that the four capacitors C are open, C=0), so we can get the equations bellow (by the circuit theorem: the voltage drop equals the multiplication of the resistance and the current):

$$\begin{aligned} v_1 &= R \times i_1 \\ v_2 &= R \times i_2 \\ v_3 &= R \times i_3 \\ v_4 &= R \times i_4 \end{aligned} \tag{4-9}$$

And we have the equation bellow (by the circuit theorem: current equals the multiplication of the conductance and the voltage):

$$\begin{aligned} i_1 &= w_{11}(x_1+\theta_1-d_1)+w_{12}(x_2+\theta_2-d_2)+w_{13}(x_3+\theta_3-d_3)+w_{14}(x_4+\theta_4-d_4) \\ i_2 &= w_{21}(x_1+\theta_1-d_1)+w_{22}(x_2+\theta_2-d_2)+w_{23}(x_3+\theta_3-d_3)+w_{24}(x_4+\theta_4-d_4) \\ i_3 &= w_{31}(x_1+\theta_1-d_1)+w_{32}(x_2+\theta_2-d_2)+w_{33}(x_3+\theta_3-d_3)+w_{34}(x_4+\theta_4-d_4) \\ i_2 &= w_{41}(x_1+\theta_1-d_1)+w_{42}(x_2+\theta_2-d_2)+w_{43}(x_3+\theta_3-d_3)+w_{44}(x_4+\theta_4-d_4) \end{aligned} \tag{4-10}$$

And the effect of $\varphi(\bullet)$ is as voltage amplifier, so we can write the equations bellow:

$$\begin{aligned} x_1 &= \varphi(v_1) = \tanh(v_1) \\ x_2 &= \varphi(v_2) = \tanh(v_2) \\ x_3 &= \varphi(v_3) = \tanh(v_3) \\ x_4 &= \varphi(v_4) = \tanh(v_4) \end{aligned} \tag{4-11}$$

And the equation of the output of Hopfield neural network, the HNN controller's control signal $\hat{f}$ is bellow:

$$\hat{f} = c_{amp} \times (x_1 + x_2 + x_3 + x_4) \tag{4-12}$$

We define the error J as the measurement of the half squared distance between the reference controller's control signal f and the HNN controller's control signal $\hat{f}$. And we write the equation bellow:

$$J = \frac{1}{2} \times (f - \hat{f})^2 \tag{4-13}$$

Next, we have to find a good set of w_{11}, w_{12}, w_{21} and w_{22} to make J smaller, so the half squared distance between the reference controller's control signal f and the HNN controller's control signal $\hat{f}$ will be smaller. How can we reach this task? We can deal it by training the w_{11}, w_{12}, w_{13}, w_{14}, w_{21}, w_{22}, w_{23}, w_{24}, w_{31}, w_{32}, w_{33}, w_{34}, w_{41}, w_{42}, w_{43} and w_{44} with the steepest descent method. Let us take w_{11} for example to show how it to be trained. First, we can write the equation bellow to show how we get the better next value of w_{11}:

$$w_{11}(k+1) = w_{11}(k) - \eta \times \frac{\partial J}{\partial w_{11}} \tag{4-14}$$

Where η is the learning rate, and we should calculate the value of $\frac{\partial J}{\partial w_{11}}$. We can use the chain rule to write the equation bellow:

$$\frac{\partial J}{\partial w_{11}} = \frac{\partial J}{\partial \hat{f}} \times \frac{\partial \hat{f}}{\partial x_1} \times \frac{\partial x_1}{\partial v_1} \times \frac{\partial v_1}{\partial w_{11}} \tag{4-15}$$

From the equation (4-13), we have the equation bellow:

$$\frac{\partial J}{\partial \hat{f}} = (\hat{f} - f) \tag{4-16}$$

From the equation (4-12), we have the equation bellow:

$$\frac{\partial \hat{f}}{\partial x_1} = c_{amp} \tag{4-17}$$

From the equation (4-11), we have the equation bellow:

$$\frac{\partial x_1}{\partial v_1} = 1 - [\tanh(v_1)]^2 \tag{4-18}$$

From the equation (4-9), (4-10), we have the equation bellow:

$$\frac{\partial v_1}{\partial w_{11}} = R \times (x_1 + \theta_1 - d_1) \tag{4-19}$$

So, from the equations (4-15), (4-16), (4-17), (4-18) and (4-19), we have equation bellow:

$$\frac{\partial J}{\partial w_{11}} = (\hat{f} - f) \times c_{amp} \times \{1 - [\tanh(v_1)]^2\} \times R \times (x_1 + \theta_1 - d_1) \tag{4-20}$$

Substitute equations (4-20) to the equations (4-14), we have equation bellow:

$$w_{11}(k+1) = w_{11}(k) - \eta \times (\hat{f} - f) \times c_{amp} \times \{1 - [\tanh(v_1)]^2\} \times R \times (x_1 + \theta_1 - d_1) \tag{4-21}$$

Similarly, for training w_{12}, w_{13}, w_{14}, w_{21}, w_{22}, w_{23}, w_{24}, w_{31}, w_{32}, w_{33}, w_{34}, w_{41}, w_{42}, w_{43} and w_{44}, we can write the equations bellow:

$$w_{12}(k+1) = w_{12}(k) - \eta \times (\hat{f} - f) \times c_{amp} \times \{1 - [\tanh(v_1)]^2\} \times R \times (x_2 + \theta_2 - d_2) \tag{4-22}$$

$$w_{13}(k+1) = w_{13}(k) - \eta \times (\hat{f} - f) \times c_{amp} \times \{1 - [\tanh(v_1)]^2\} \times R \times (x_3 + \theta_3 - d_3) \tag{4-23}$$

$$w_{14}(k+1) = w_{14}(k) - \eta \times (\hat{f} - f) \times c_{amp} \times \{1 - [\tanh(v_1)]^2\} \times R \times (x_4 + \theta_4 - d_4) \tag{4-24}$$

$$w_{21}(k+1) = w_{21}(k) - \eta \times (\hat{f} - f) \times c_{amp} \times \{1 - [\tanh(v_2)]^2\} \times R \times (x_1 + \theta_1 - d_1) \tag{4-25}$$

$$w_{22}(k+1) = w_{22}(k) - \eta \times (\hat{f} - f) \times c_{amp} \times \{1 - [\tanh(v_2)]^2\} \times R \times (x_2 + \theta_2 - d_2) \quad (4\text{-}26)$$

$$w_{23}(k+1) = w_{23}(k) - \eta \times (\hat{f} - f) \times c_{amp} \times \{1 - [\tanh(v_2)]^2\} \times R \times (x_3 + \theta_3 - d_3) \quad (4\text{-}27)$$

$$w_{24}(k+1) = w_{24}(k) - \eta \times (\hat{f} - f) \times c_{amp} \times \{1 - [\tanh(v_2)]^2\} \times R \times (x_4 + \theta_4 - d_4) \quad (4\text{-}28)$$

$$w_{31}(k+1) = w_{31}(k) - \eta \times (\hat{f} - f) \times c_{amp} \times \{1 - [\tanh(v_3)]^2\} \times R \times (x_1 + \theta_1 - d_1) \quad (4\text{-}29)$$

$$w_{32}(k+1) = w_{32}(k) - \eta \times (\hat{f} - f) \times c_{amp} \times \{1 - [\tanh(v_3)]^2\} \times R \times (x_2 + \theta_2 - d_2) \quad (4\text{-}30)$$

$$w_{33}(k+1) = w_{33}(k) - \eta \times (\hat{f} - f) \times c_{amp} \times \{1 - [\tanh(v_3)]^2\} \times R \times (x_3 + \theta_3 - d_3) \quad (4\text{-}31)$$

$$w_{34}(k+1) = w_{34}(k) - \eta \times (\hat{f} - f) \times c_{amp} \times \{1 - [\tanh(v_3)]^2\} \times R \times (x_4 + \theta_4 - d_4) \quad (4\text{-}32)$$

$$w_{41}(k+1) = w_{41}(k) - \eta \times (\hat{f} - f) \times c_{amp} \times \{1 - [\tanh(v_4)]^2\} \times R \times (x_1 + \theta_1 - d_1) \quad (4\text{-}33)$$

$$w_{42}(k+1) = w_{42}(k) - \eta \times (\hat{f} - f) \times c_{amp} \times \{1 - [\tanh(v_4)]^2\} \times R \times (x_2 + \theta_2 - d_2) \quad (4\text{-}34)$$

$$w_{43}(k+1) = w_{43}(k) - \eta \times (\hat{f} - f) \times c_{amp} \times \{1 - [\tanh(v_4)]^2\} \times R \times (x_3 + \theta_3 - d_3) \quad (4\text{-}35)$$

$$w_{44}(k+1) = w_{44}(k) - \eta \times (\hat{f} - f) \times c_{amp} \times \{1 - [\tanh(v_4)]^2\} \times R \times (x_4 + \theta_4 - d_4) \quad (4\text{-}36)$$

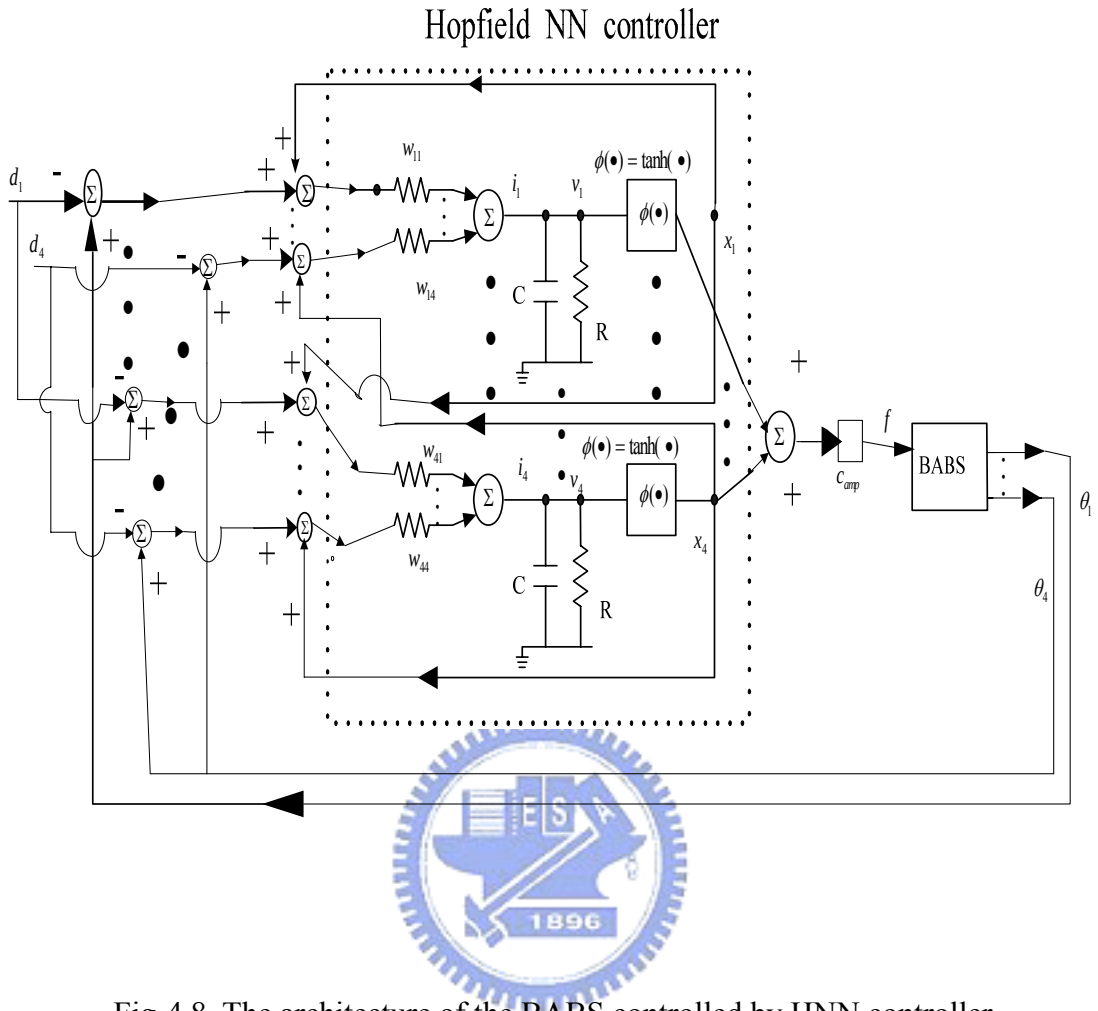

Fig-4.8. The architecture of the BABS controlled by HNN controller

In the working phase, the HNN is a real time controller, responding to control the BABS. The value of the set of w_{11}, w_{12}, w_{13}, w_{14}, w_{21}, w_{22}, w_{23}, w_{24}, w_{31}, w_{32}, w_{33}, w_{34}, w_{41}, w_{42}, w_{43} and w_{44} is fixed. And as a real time controller, the circuit is dynamic, so we cannot ask the circuit always in the steady state, so we should know that the current passing the both capacitors C is not always zero. Actually, it is very complicated to calculate the output of BABS. In the working phase, the architecture is the recurrent neural network. The output of BABS is $(\theta_1(t),\theta_2(t),\theta_3(t),\theta_4(t))$, and $(\theta_1(t),\theta_2(t),\theta_3(t),\theta_4(t))$ will affect the values of the current $(i_1(t),i_2(t),i_3(t),i_4(t))$, and the current $(i_1(t),i_2(t),i_3(t),i_4(t))$ will affect the values of the voltage $(v_1(t),v_2(t),v_3(t),v_4(t))$, and the time-varying current passing the capacitors will affect the values of the voltage

$(v_1(t), v_2(t), v_3(t), v_4(t))$, too. The voltage $(v_1(t), v_2(t), v_3(t), v_4(t))$ will be amplified by $\varphi(\bullet)$ to get the values of $(x_1(t), x_2(t), x_3(t), x_4(t))$, and the values of $x_1(t), x_2(t), x_3(t)$ and $x_4(t)$ will affect $i_1(t), i_2(t), i_3(t)$, and $i_4(t)$ respectively, and $(x_1(t) + x_2(t) + x_3(t) + x_4(t)) \times c_{amp}$ will get the control signal $f(t)$. Of course, the control signal $f(t)$ will affect the output of the BABS $(\theta_1(t), \theta_2(t), \theta_3(t), \theta_4(t))$ by the BABS equation (4-7). So this is a complicated recurrent control system. So we don't calculate the analytic solution of this system. Instead, we try to use the numerical method to simulate this system.

4.4 Simulation Result

4.4.1 Seeking for Weighting Factors

First, we should try to seek a good set of w_{11}, w_{12}, w_{13}, w_{14}, w_{21}, w_{22}, w_{23}, w_{24}, w_{31}, w_{32}, w_{33}, w_{34}, w_{41}, w_{42}, w_{43} and w_{44}. With the equations (4-21), (4-22), (4-23), (4-24), (4-25), (4-26), (4-27), (4-28), (4-29), (4-30), (4-31), (4-32), (4-33), (4-34), (4-35)and (4-36), and we may use many training epochs for get better training results. And we can show that the total error during the whole time is the function of the epoch number and the error is the function of the time(last epoch for example) in the following figure, Fig-4.9, and we can find that using 20 epochs is good enough for simulation.

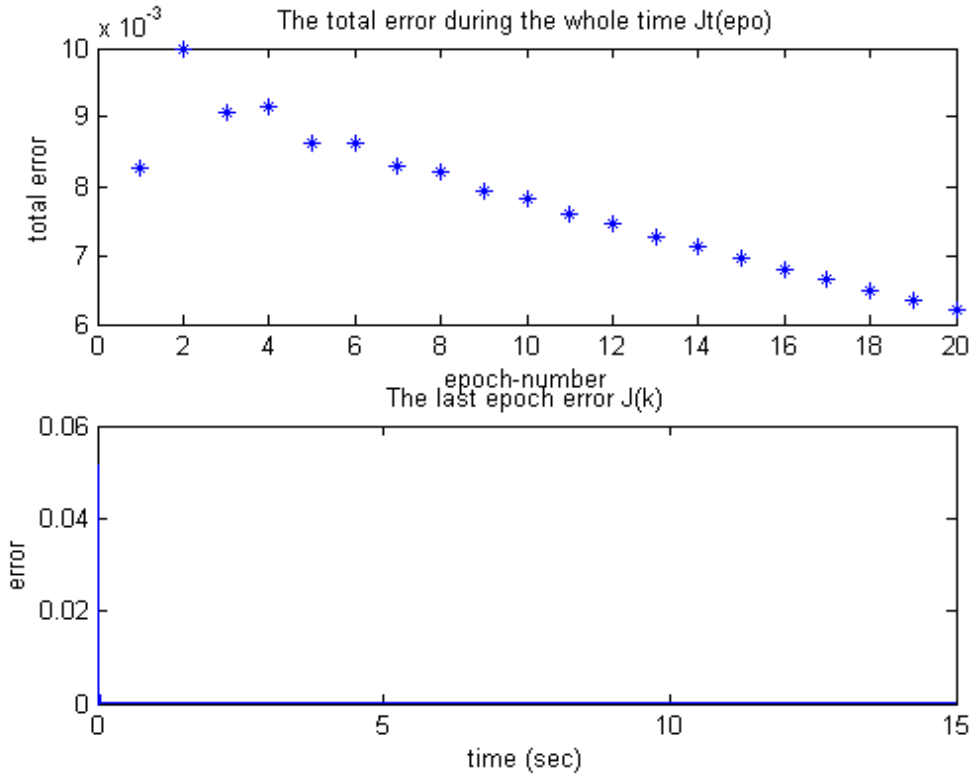

Fig-4.9. The total error $J_t(epo)$ and the last epoch error $J(k,20)$

By the simulation, we can find a good set of w_{11}, w_{12}, w_{13}, w_{14}, w_{21}, w_{22}, w_{23}, w_{24}, w_{31}, w_{32}, w_{33}, w_{34}, w_{41}, w_{42}, w_{43} and w_{44}, and we write it down as following equation in the matrix form:

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{bmatrix} = \begin{bmatrix} -0.0048 & -0.0023 & 0.0088 & 0.0047 \\ -0.0048 & -0.0023 & 0.0088 & 0.0047 \\ -0.0048 & -0.0023 & 0.0088 & 0.0047 \\ -0.0048 & -0.0023 & 0.0088 & 0.0047 \end{bmatrix} \quad (4\text{-}37)$$

In fact, the seeking of a good set of w_{11}, w_{12}, w_{13}, w_{14}, w_{21}, w_{22}, w_{23}, w_{24}, w_{31}, w_{32}, w_{33}, w_{34}, w_{41}, w_{42}, w_{43} and w_{44} is the process of the training iteration in the whole time and every epoch, and we can show the training iteration process (in the last epoch) in the whole time by the following figures:

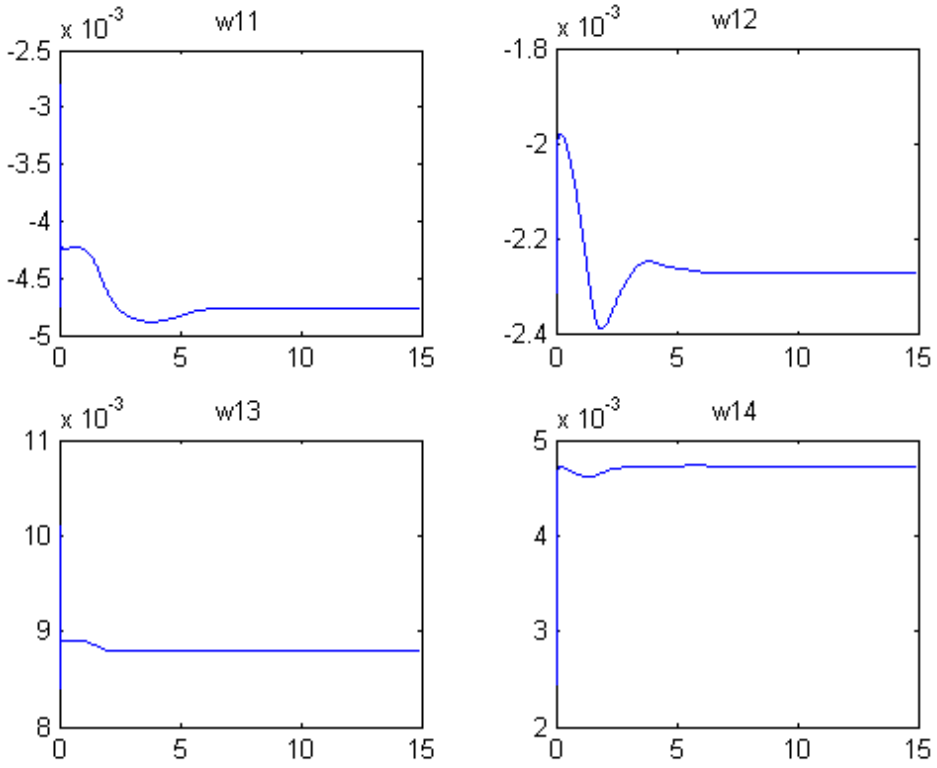

Fig-4.10. The last epoch training iteration process of w_{11}, w_{12}, w_{13} and w_{14}

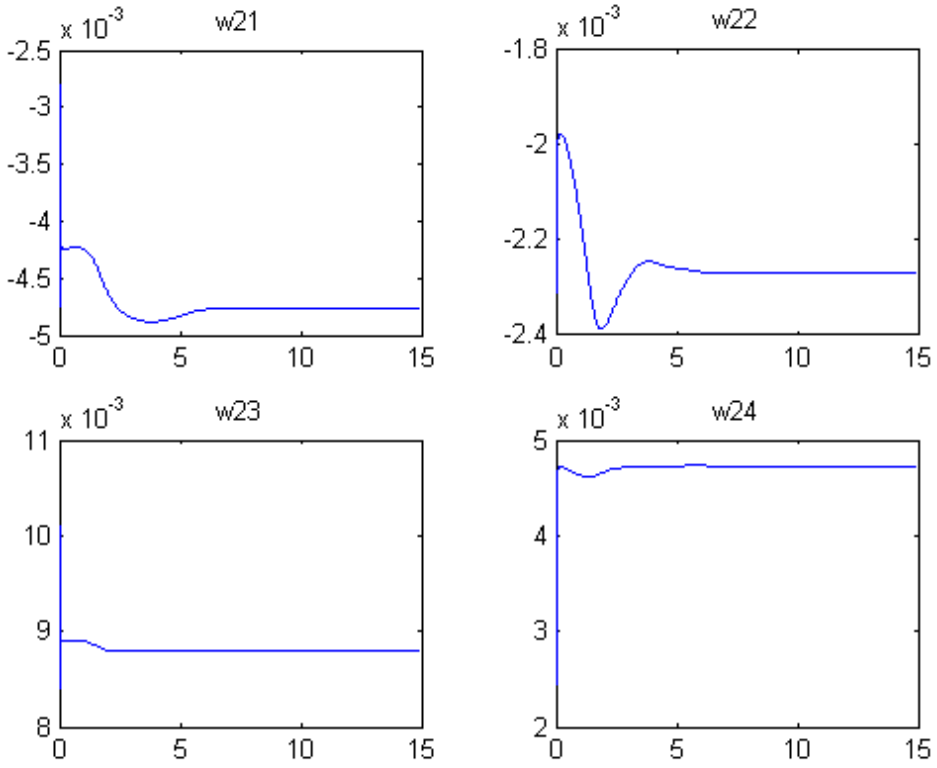

Fig-4.11. The last epoch training iteration process of w_{21}, w_{22}, w_{23} and w_{24}

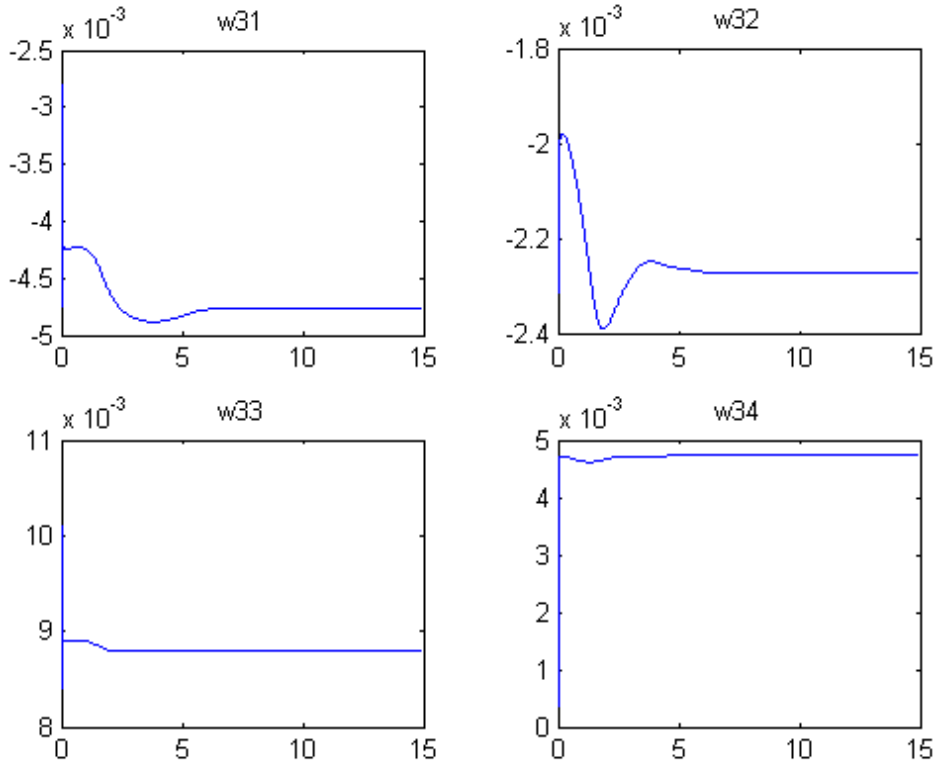

Fig-4.12. The last epoch training iteration process of w_{31}, w_{32}, w_{33} and w_{34}

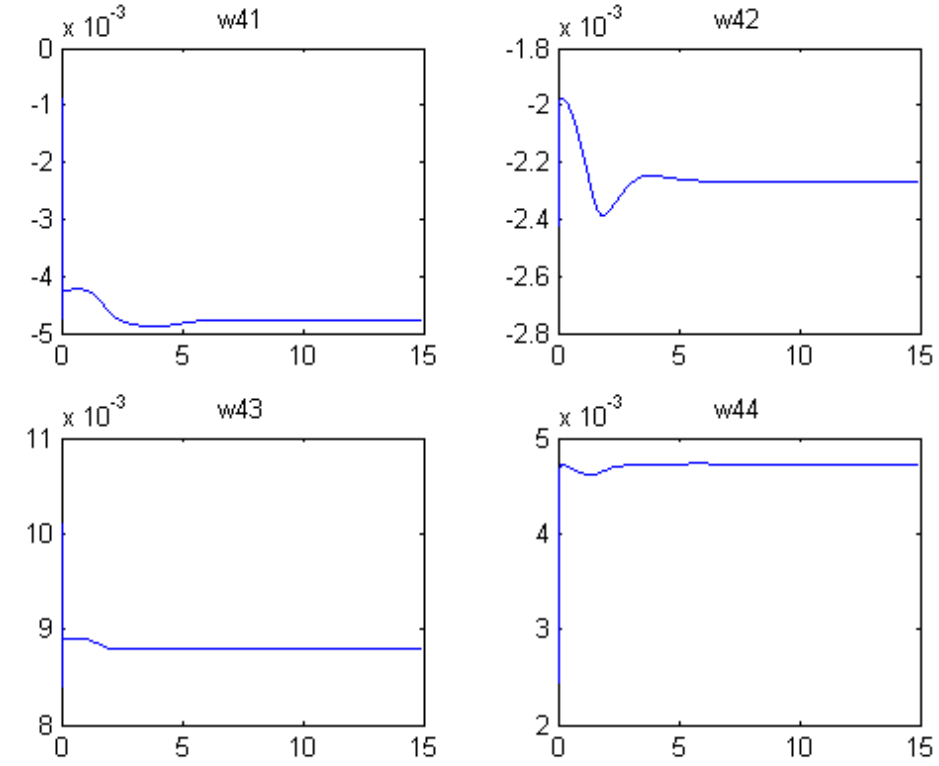

Fig-4.13. The last epoch training iteration process of w_{41}, w_{42}, w_{43} and w_{44}

4.4.2 BABS Controlled by HNN Controller Trained with the Same Initial State

After we have the values of the good set of w_{11}, w_{12}, w_{13}, w_{14}, w_{21}, w_{22}, w_{23}, w_{24}, w_{31}, w_{32}, w_{33}, w_{34}, w_{41}, w_{42}, w_{43} and w_{44}, we can begin to run the simulation of the BABS controlled by the HNN controller in real time. We use the Matlab with the 4^{th} order of Runge-Kutta formula to simulate it. Because its nonlinearity is very high, so we use the 4^{th} order of Runge-Kutta formula to simulate BABS controlled by HNN controller. By the simulation, we can get the following figures as the results:

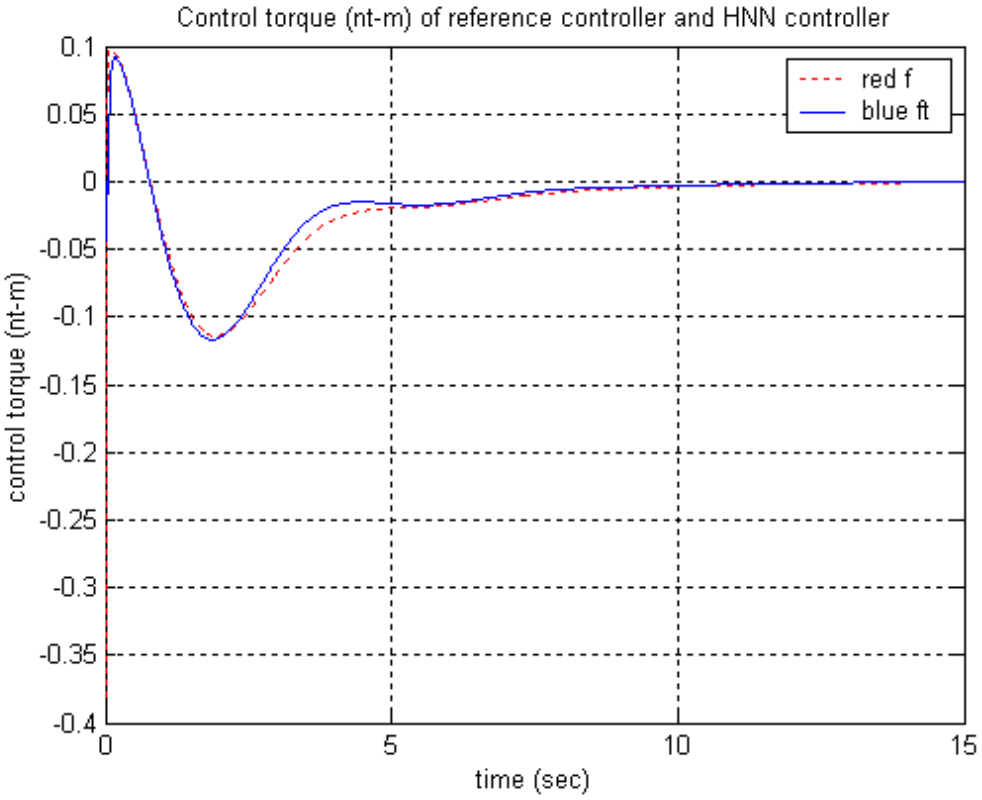

Fig-4.14. The control torques of BABS: reference controller (dash line) and HNN controller (solid line)

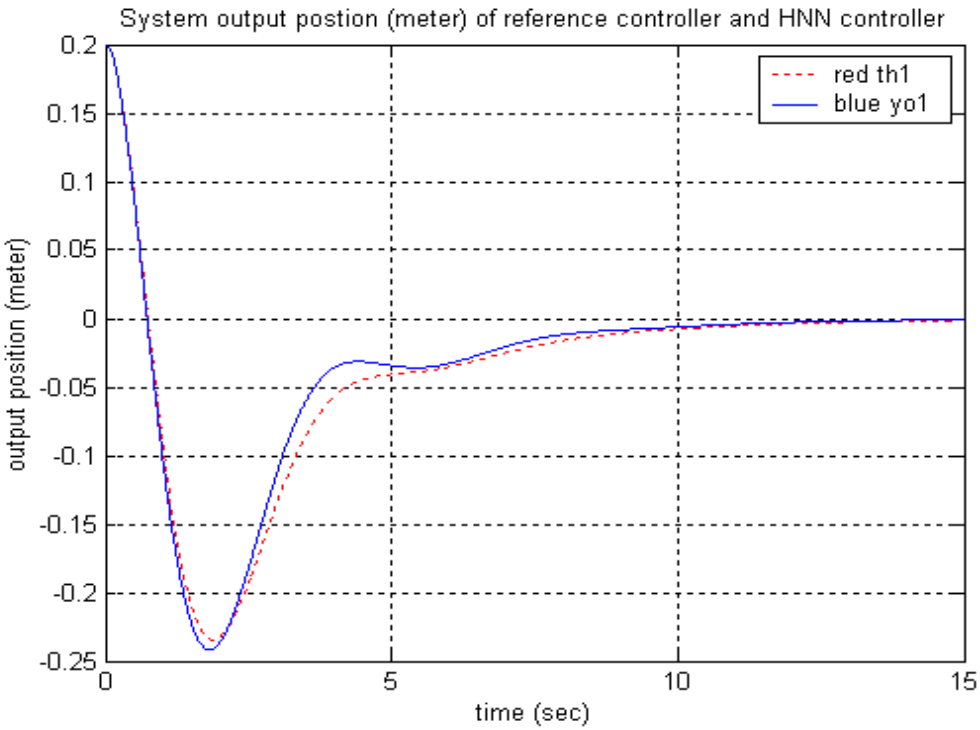

Fig-4.15. The ball's positions of BABS: reference controller (dash line) and HNN controller (solid line)

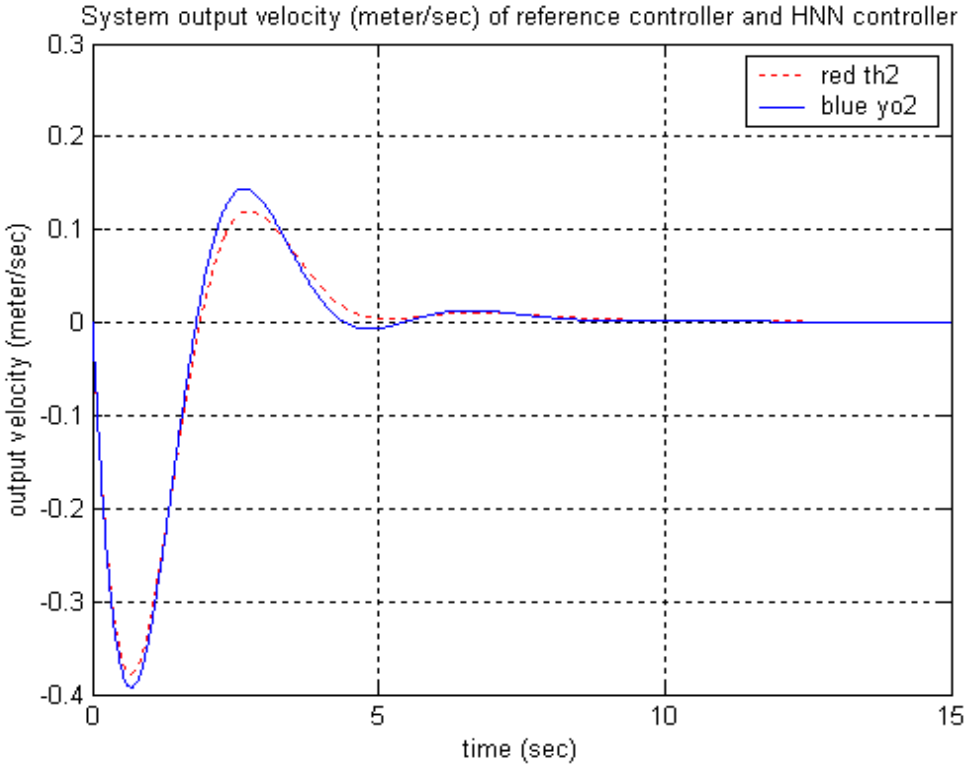

Fig-4.16. The ball's velocities of BABS: reference controller (dash line) and HNN controller (solid line)

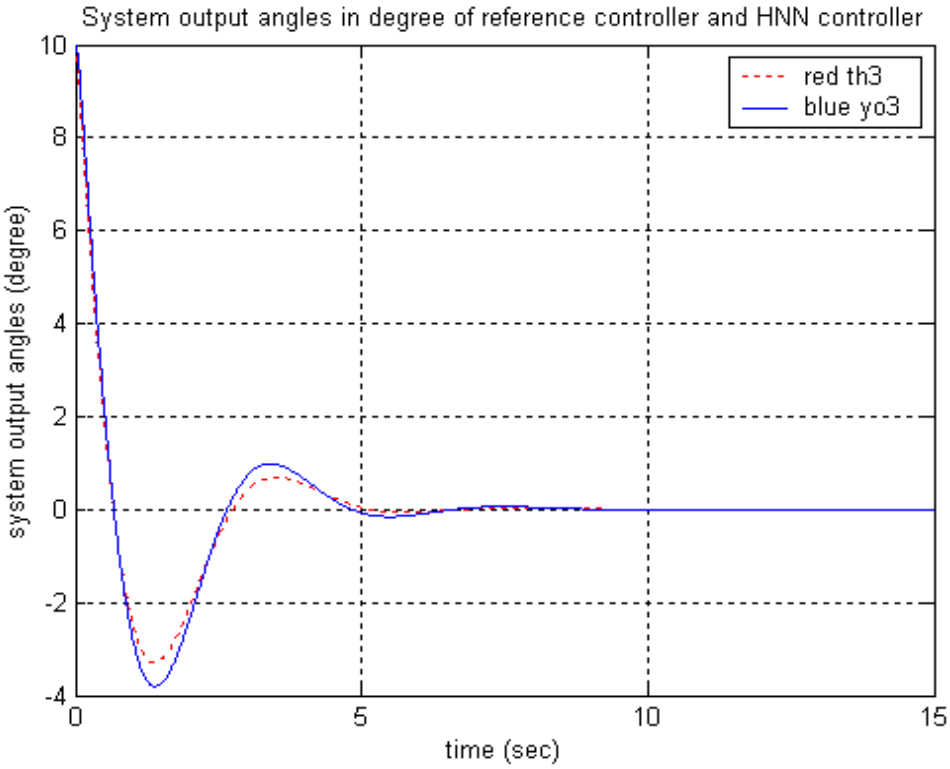

Fig-4.17. The beam's angles of BABS: reference controller (dash line) and HNN controller (solid line)

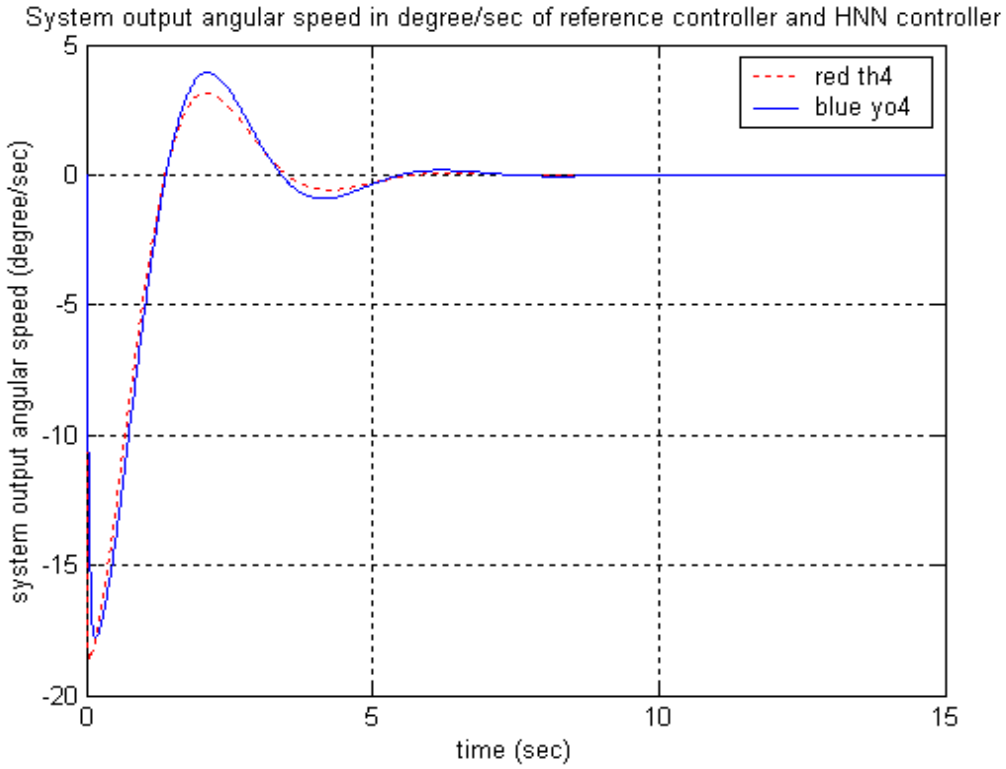

Fig-4.18. The beam's angular speeds of BABS: reference controller (dash line) and HNN controller (solid line)

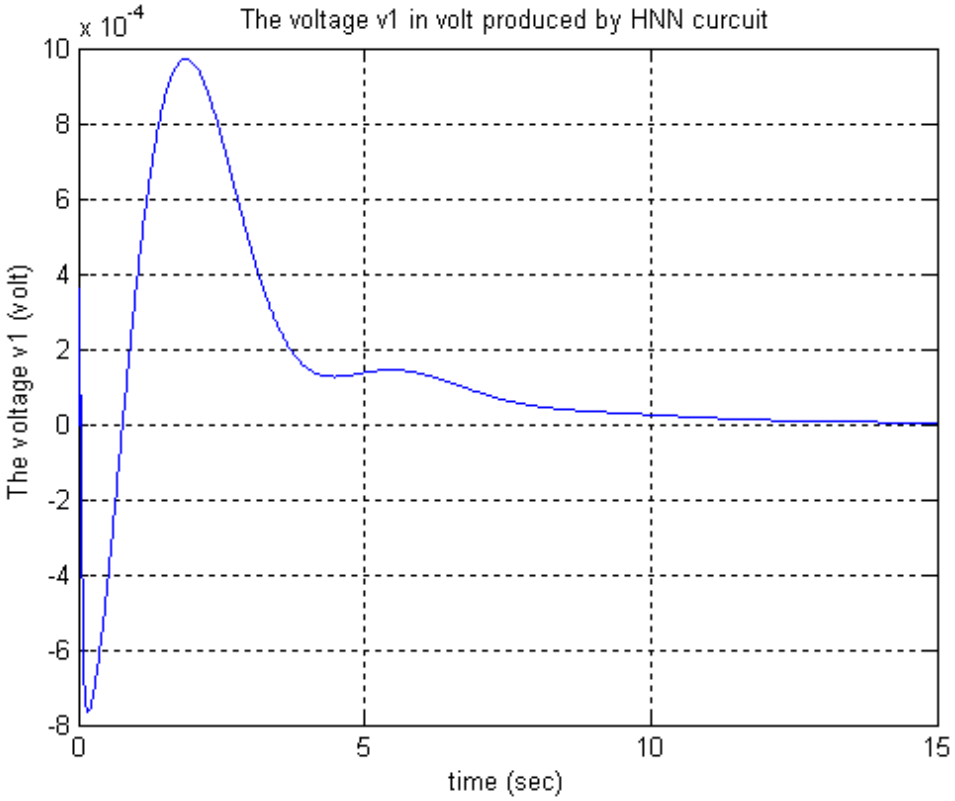

Fig-4.19. The node 1 voltage v_1 of the HNN circuit

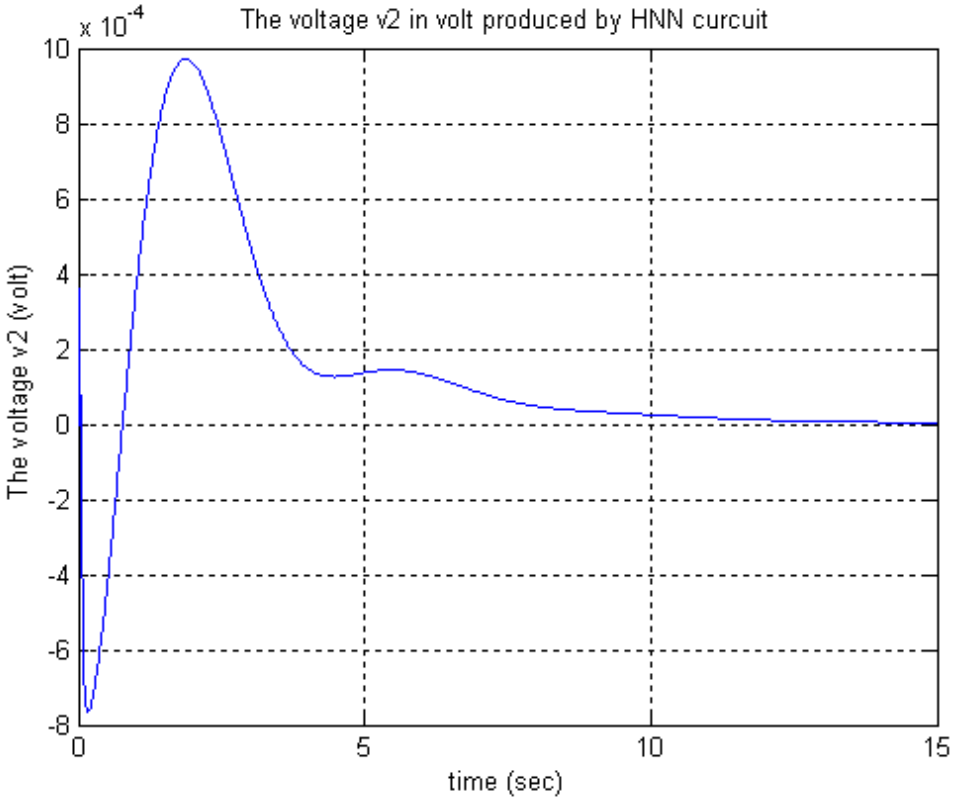

Fig-4.20. The node 2 voltage v_2 of the HNN circuit

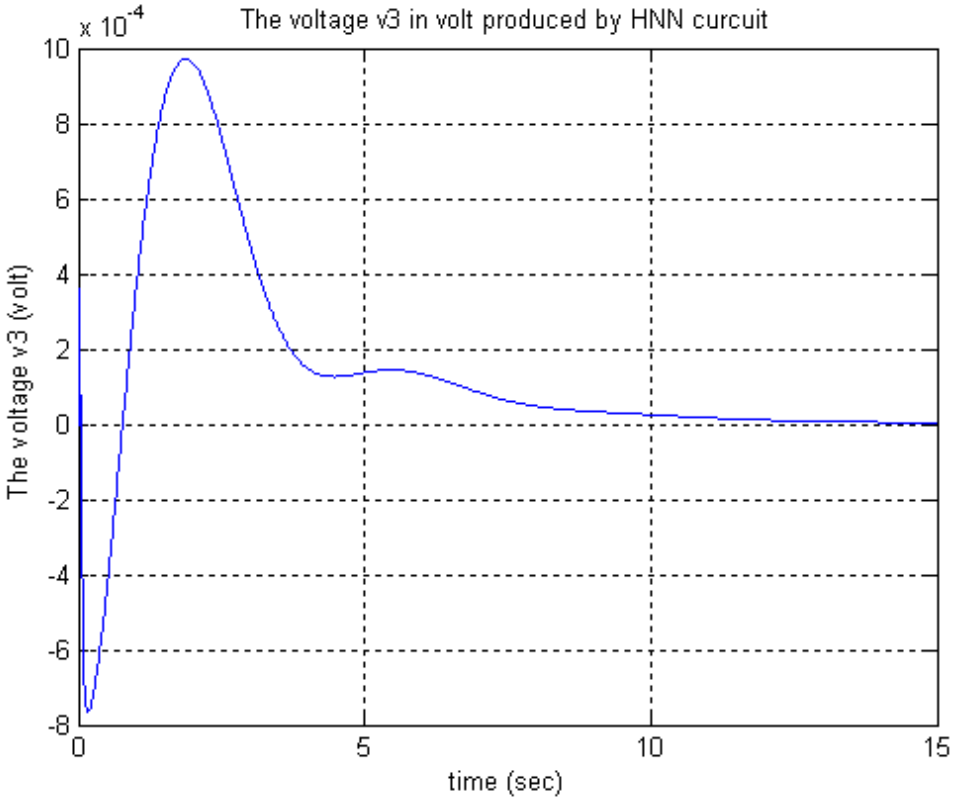

Fig-4.21. The node 3 voltage v_3 of the HNN circuit

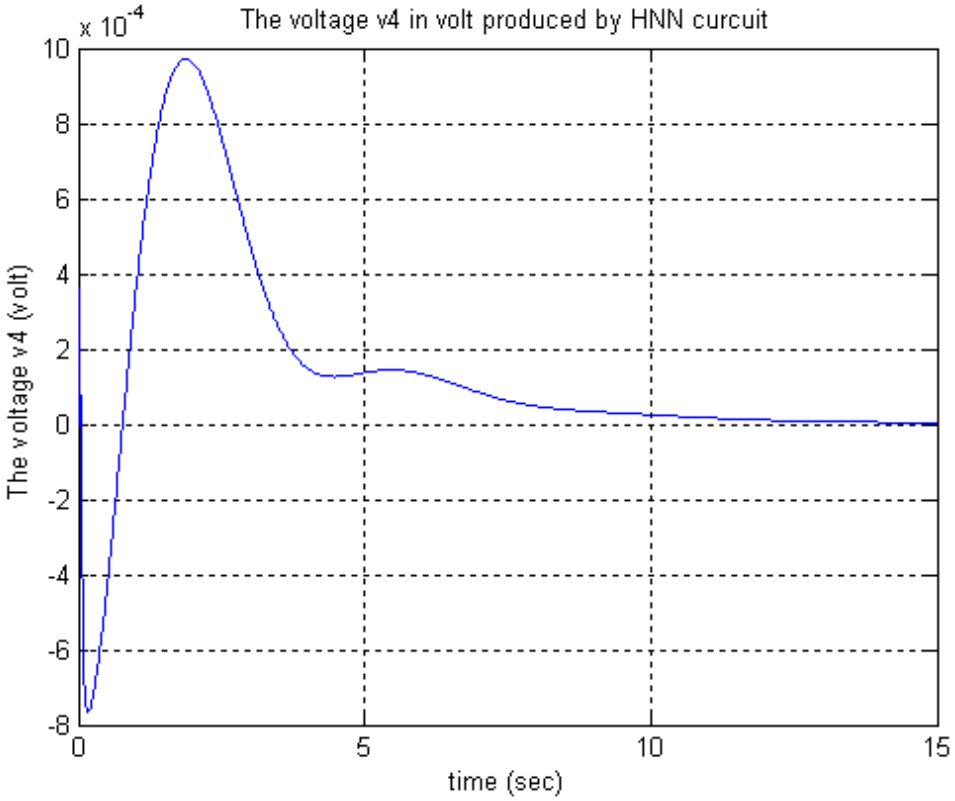

Fig-4.22. The node 4 voltage v_4 of the HNN circuit

4.4.3 BABS Controlled by HNN Controller after Trained with Different Initial State

We use the following examples to examine the abilities of HNN controllers to control BABS with initial state different from the initial state of training the HNN controller. We can let the values of w_{11}, w_{12}, w_{13}, w_{14}, w_{21}, w_{22}, w_{23}, w_{24}, w_{31}, w_{32}, w_{33}, w_{34}, w_{41}, w_{42}, w_{43} and w_{44} be the same with section 4.4.1 as the equation (4-37):

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{bmatrix} = \begin{bmatrix} -0.0048 & -0.0023 & 0.0088 & 0.0047 \\ -0.0048 & -0.0023 & 0.0088 & 0.0047 \\ -0.0048 & -0.0023 & 0.0088 & 0.0047 \\ -0.0048 & -0.0023 & 0.0088 & 0.0047 \end{bmatrix}$$

is fixed for the initial state of BABS: the initial position of the ball=0.2 (meter),the initial velocity of the ball=0 (meter/sec),the initial angle of the beam=10 (degree), and the initial angular speed of the beam of the BABS=0 (degree/sec) in the training phase. And then, we will examine the following sets (initial position (meter), initial velocity (meter/sec), initial angle (degree), initial angular speed (degree/sec)) of the initial state of BABS: (0.1, 0, 5, 0) and (0.4, 0, 20, 0) in the working phase as the following figures:

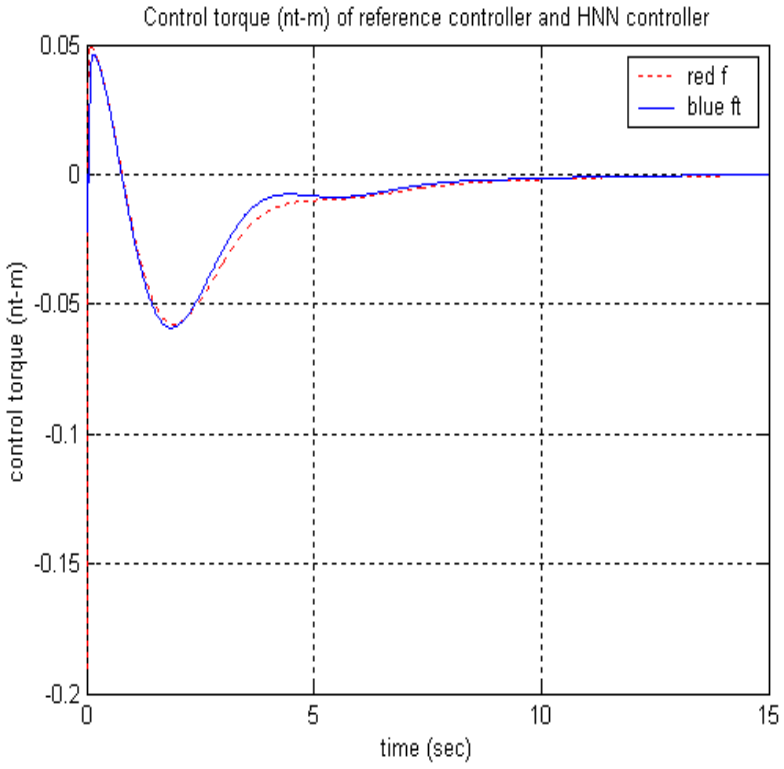

Fig-4.23. The control torques of BABS with initial ball's position=0.1 meter, initial beam's angle=5 degree: reference controller (dash line) and HNN controller (solid line)

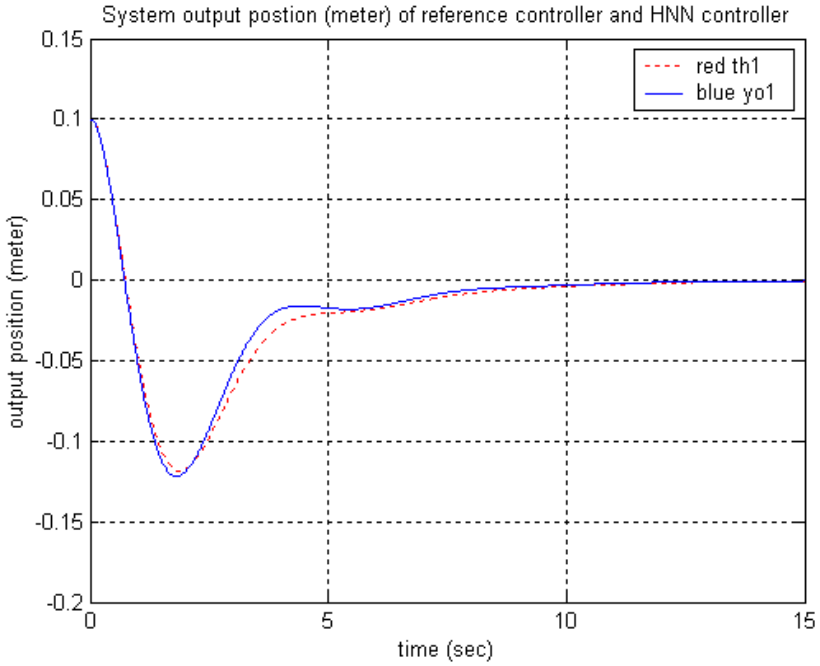

Fig-4.24. The ball's positions of BABS with initial ball's position=0.1 meter, initial beam's angle=5 degree: reference controller (dash line) and HNN controller (solid line)

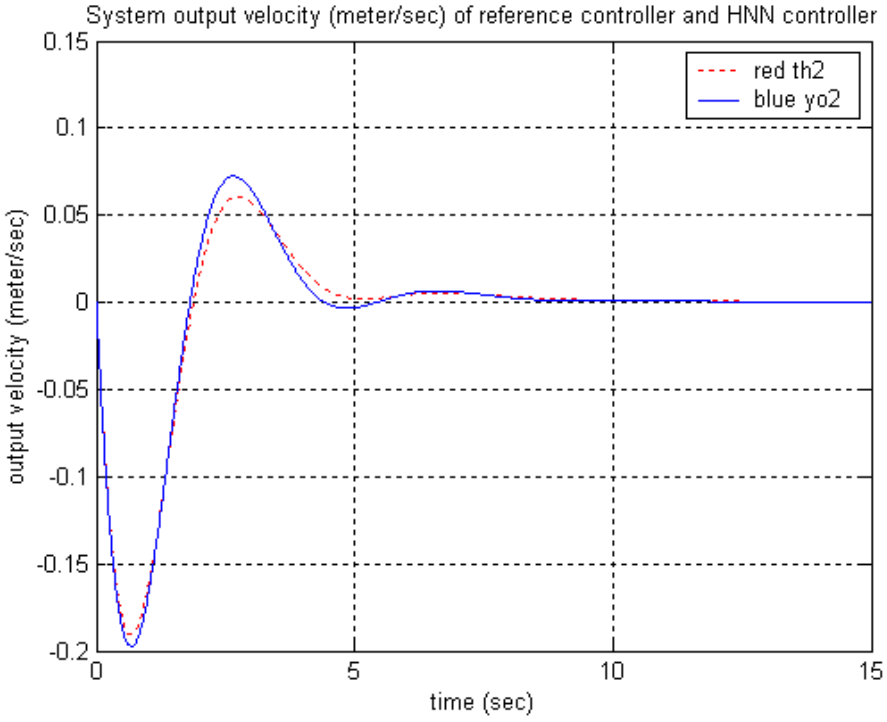

Fig-4.25. The ball's velocities of BABS with initial ball's position=0.1 meter, initial beam's angle=5 degree: reference controller (dash line) and HNN controller (solid line)

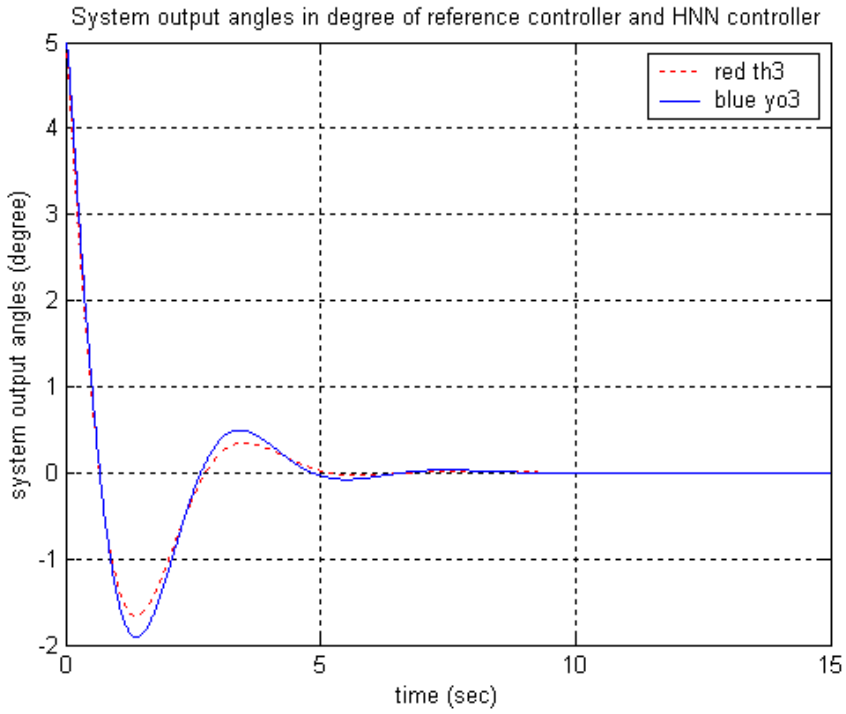

Fig-4.26. The beam's angles of BABS with initial ball's position=0.1 meter, initial beam's angle=5 degree: reference controller (dash line) and HNN controller (solid line)

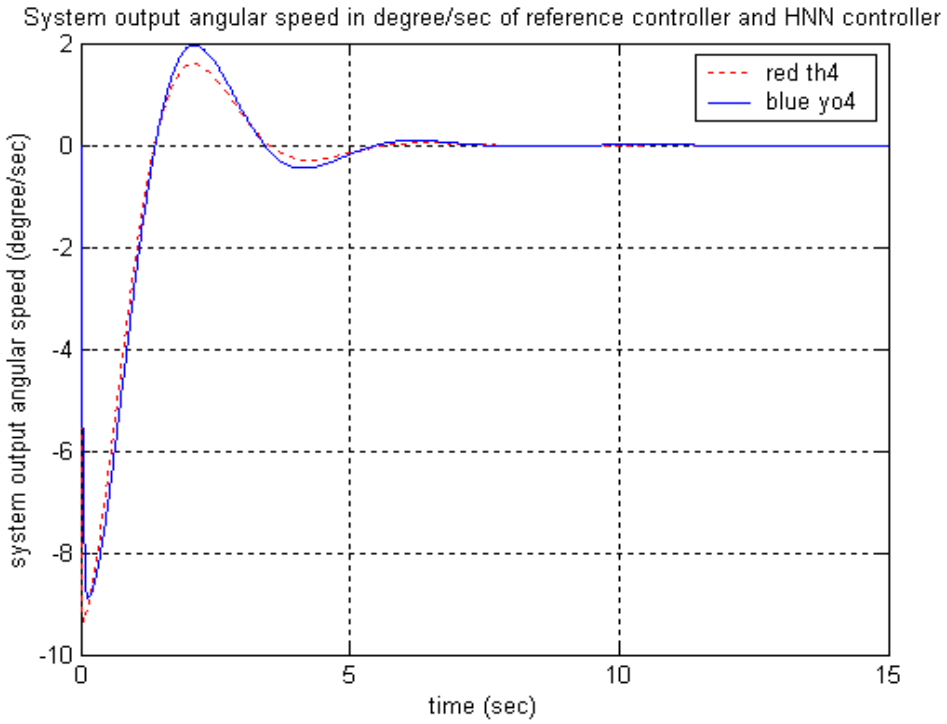

Fig-4.27. The beam's angular speeds of BABS with initial ball's position=0.1 meter, initial beam's angle=5 degree: reference controller (dash line) and HNN controller (solid line)

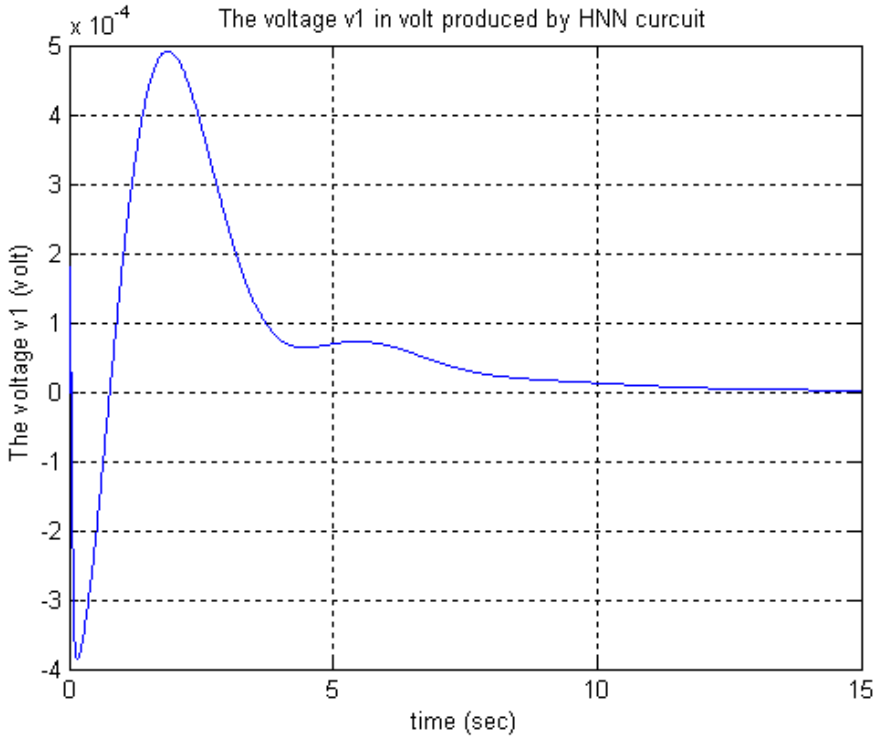

Fig-4.28. The node 1 (2) (3) (4) voltage v_1 (v_2) (v_3) (v_4) of the HNN circuit with BABS initial ball's position=0.1 meter, initial beam's angle=5 degree

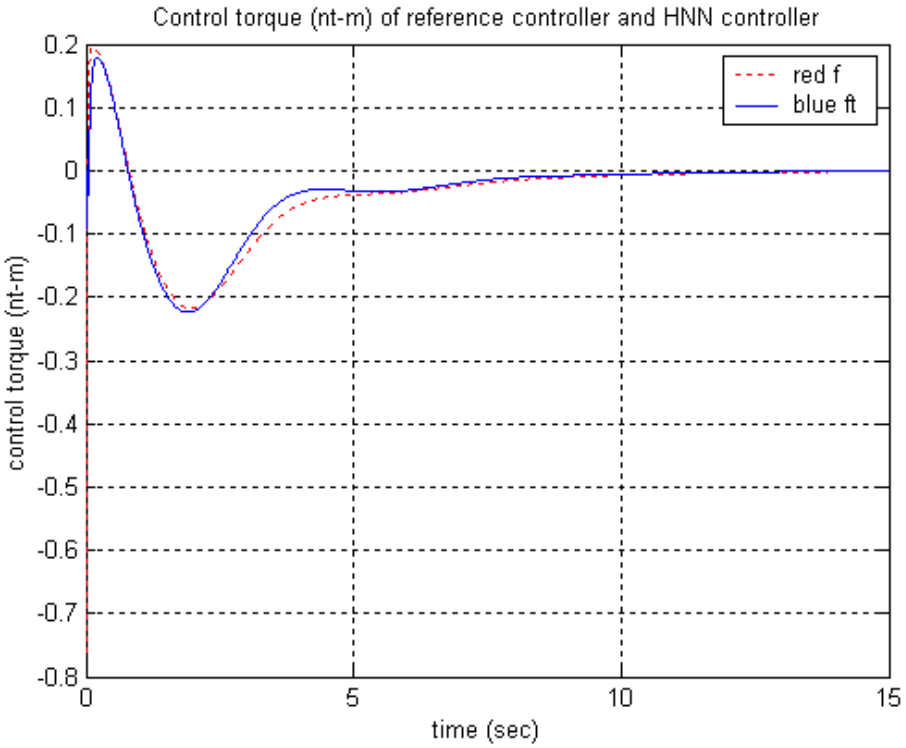

Fig-4.29. The control torques of BABS with initial ball's position=0.4 meter, initial beam's angle=20 degree: reference controller (dash line) and HNN controller (solid line)

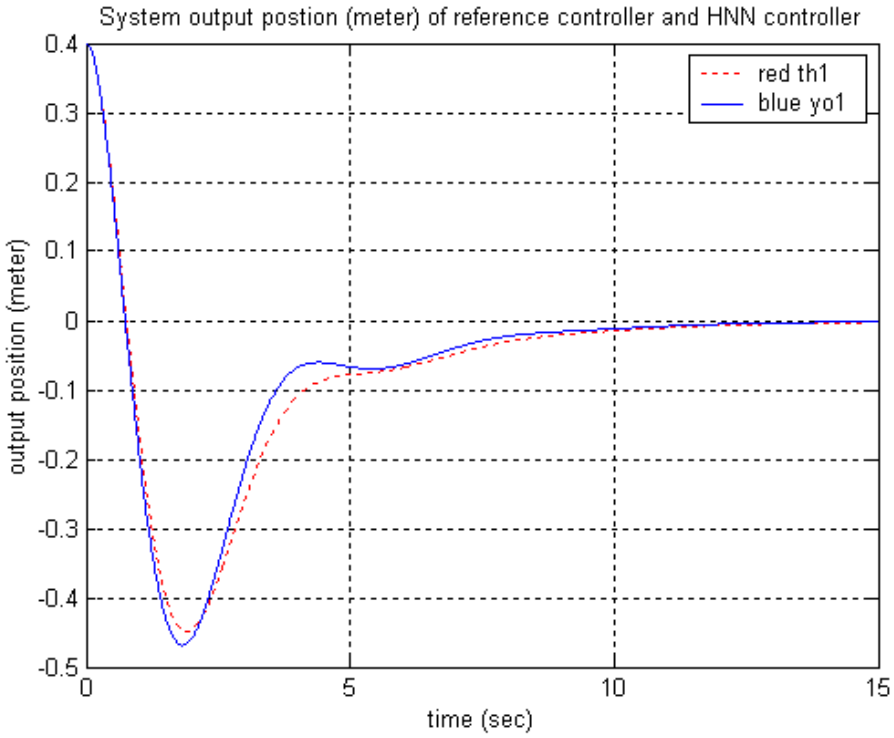

Fig-4.30. The ball's positions of BABS with initial ball's position=0.4 meter, initial beam's angle=20 degree: reference controller (dash line) and HNN controller (solid line)

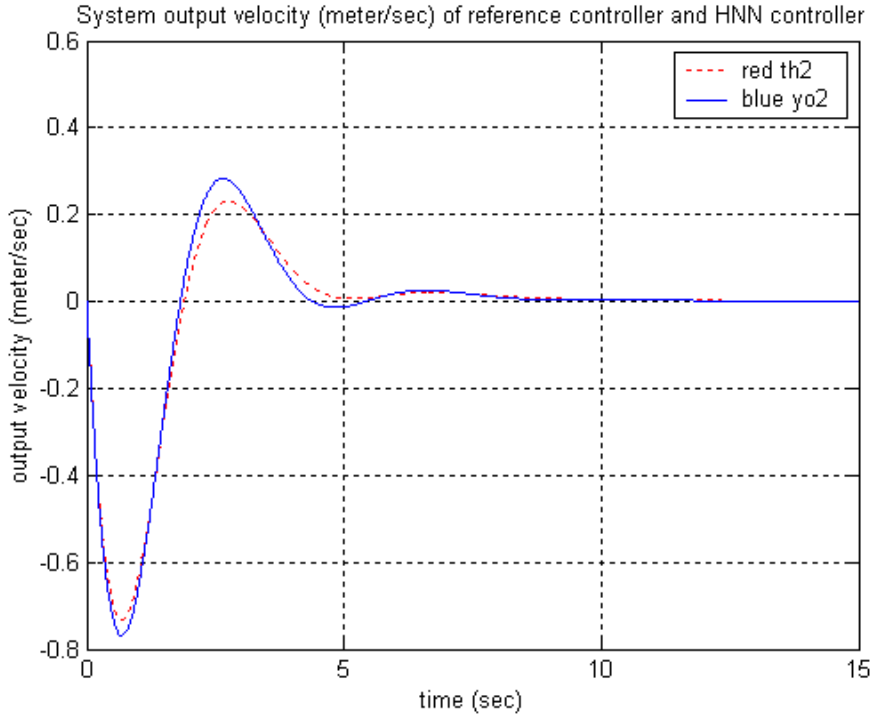

Fig-4.31. The ball's velocities of BABS with initial ball's position=0.4 meter, initial beam's angle=20 degree: reference controller (dash line) and HNN controller (solid line)

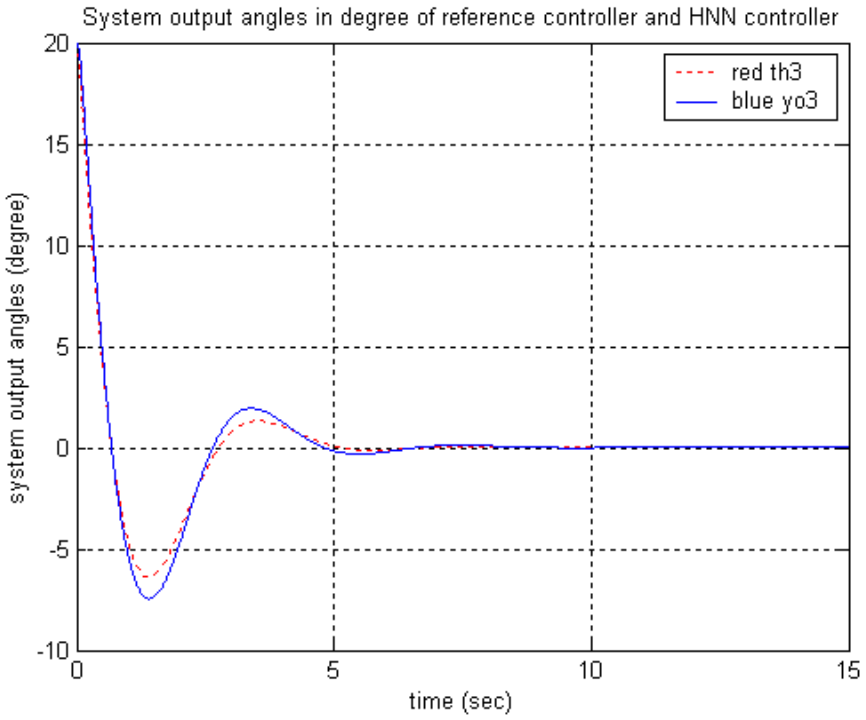

Fig-4.32. The beam's angles of BABS with initial ball's position=0.4 meter, initial beam's angle=20 degree: reference controller (dash line) and HNN controller (solid line)

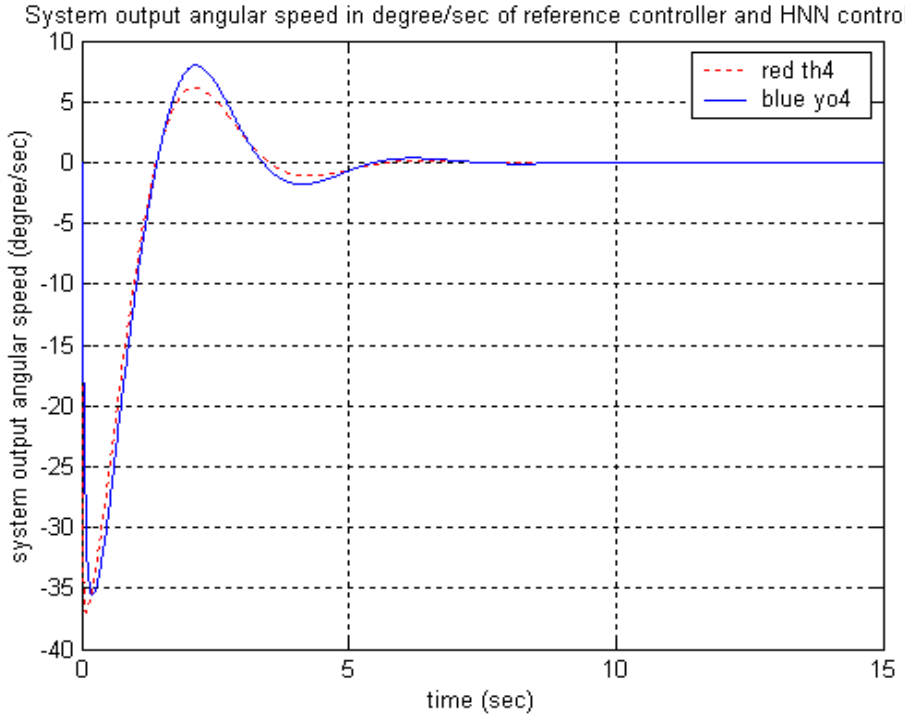

Fig-4.33. The beam's angular speeds of BABS with initial ball's position=0.4meter, initial beam's angle=20 degree: reference controller (dash line) and HNN controller (solid line)

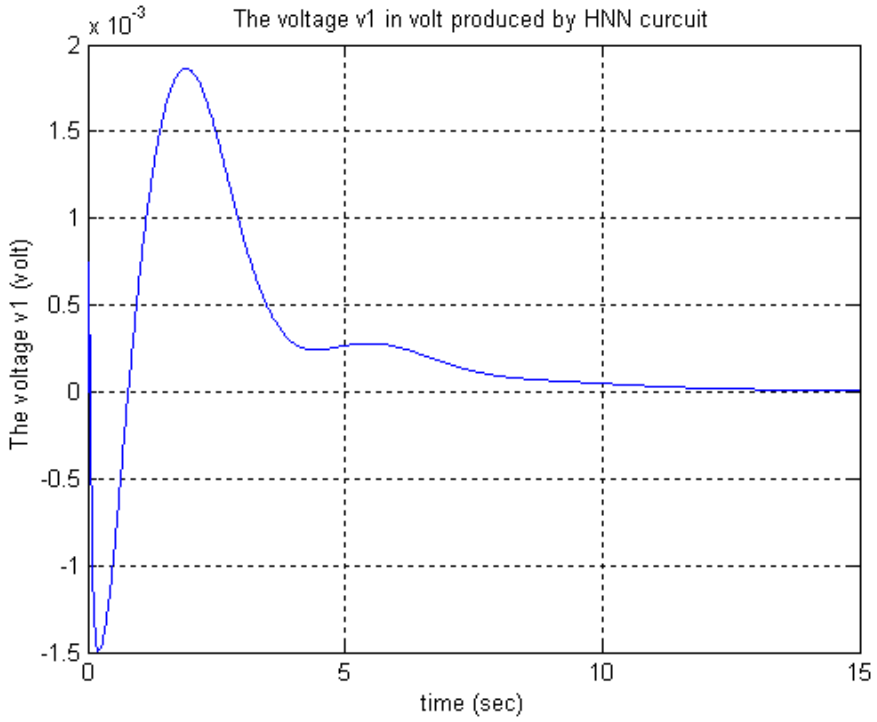

Fig-4.34. The node 1 (2) (3) (4) voltage v_1 (v_2) (v_3) (v_4) of the HNN circuit with BABS initial ball's position=0.4 meter, initial beam's angle=20 degree

4.4.4 BABS Controlled by HNN Controller Trained by Nonlinear Reference Controller with the Same Initial State

To examine the ability of HNN controller to mimic the nonlinear reference controller, we try to find the nonlinear reference controller of BABS. First, we use the functions f_{ball} and f_{beam} similar to 3.4.4 as the following equations.

$$f_{\text{ball}} = \frac{\frac{-1}{(1+e^{2.3\theta_1})(1+e^{2.3\theta_2})} + \frac{1}{(1+e^{-2.3\theta_1})(1+e^{-2.3\theta_2})}}{\frac{1}{(1+e^{2.3\theta_1})(1+e^{2.3\theta_2})} + \frac{1}{(1+e^{2.3\theta_1})(1+e^{-2.3\theta_2})} + \frac{1}{(1+e^{-2.3\theta_1})(1+e^{2.3\theta_2})} + \frac{1}{(1+e^{-2.3\theta_1})(1+e^{-2.3\theta_2})}} \quad (4\text{-}38)$$

$$f_{\text{beam}} = \frac{\frac{1}{(1+e^{2.3\theta_3})(1+e^{2.3\theta_4})} - \frac{1}{(1+e^{-2.3\theta_3})(1+e^{-2.3\theta_4})}}{\frac{1}{(1+e^{2.3\theta_3})(1+e^{2.3\theta_4})} + \frac{1}{(1+e^{2.3\theta_3})(1+e^{-2.3\theta_4})} + \frac{1}{(1+e^{-2.3\theta_3})(1+e^{2.3\theta_4})} + \frac{1}{(1+e^{-2.3\theta_3})(1+e^{-2.3\theta_4})}} \quad (4\text{-}39)$$

Equations (4-38) and (4-39) represent the necessary control torques of ball and beam, respectively. So the total necessary control torque is the sum of (4-38) and (4-39).

$$f = f_{\text{ball}} + f_{\text{beam}} \quad (4\text{-}40)$$

We use the equation (4-40) (with equations (4-38) and (4-39)) as the nonlinear

reference controller of BABS.

Let the BABS initial position (ball's position) =0.2 meter, initial velocity (ball's velocity) =0 meter/sec, initial angle (beam's angle) =10 degree, initial angular speed (beam's angular speed) =0 degree/sec, and the simulation time=15sec. We can get the following five figures of BABS controlled by the nonlinear reference controller:

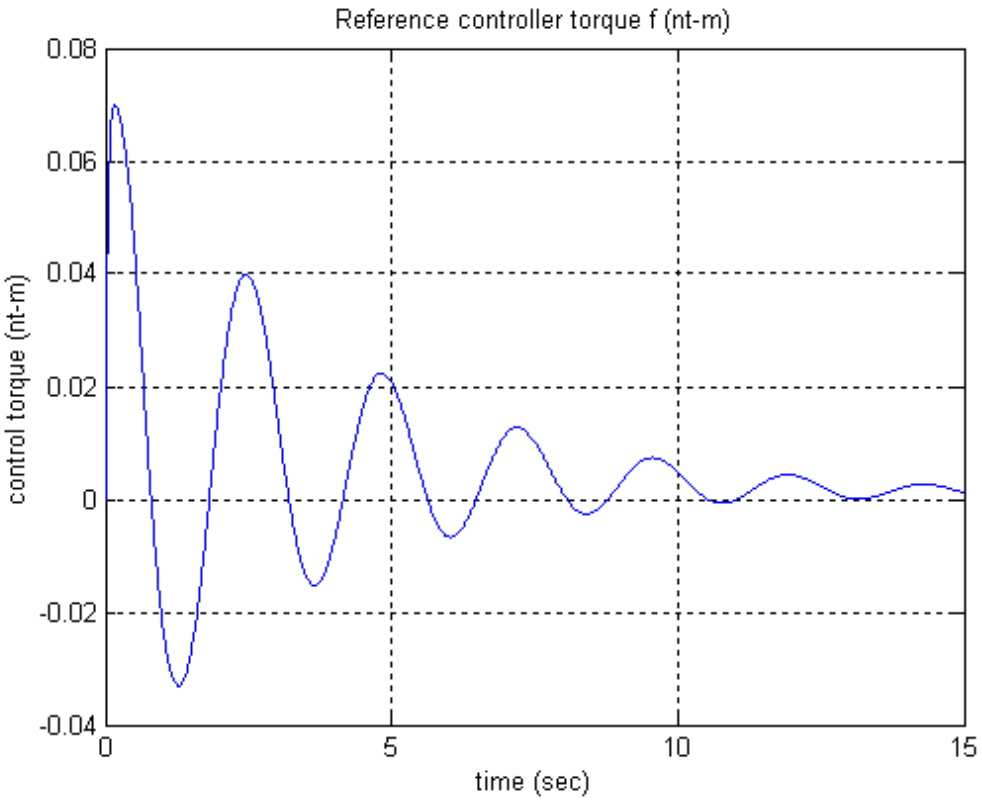

Fig-4.35. The control torque of the nonlinear reference controller

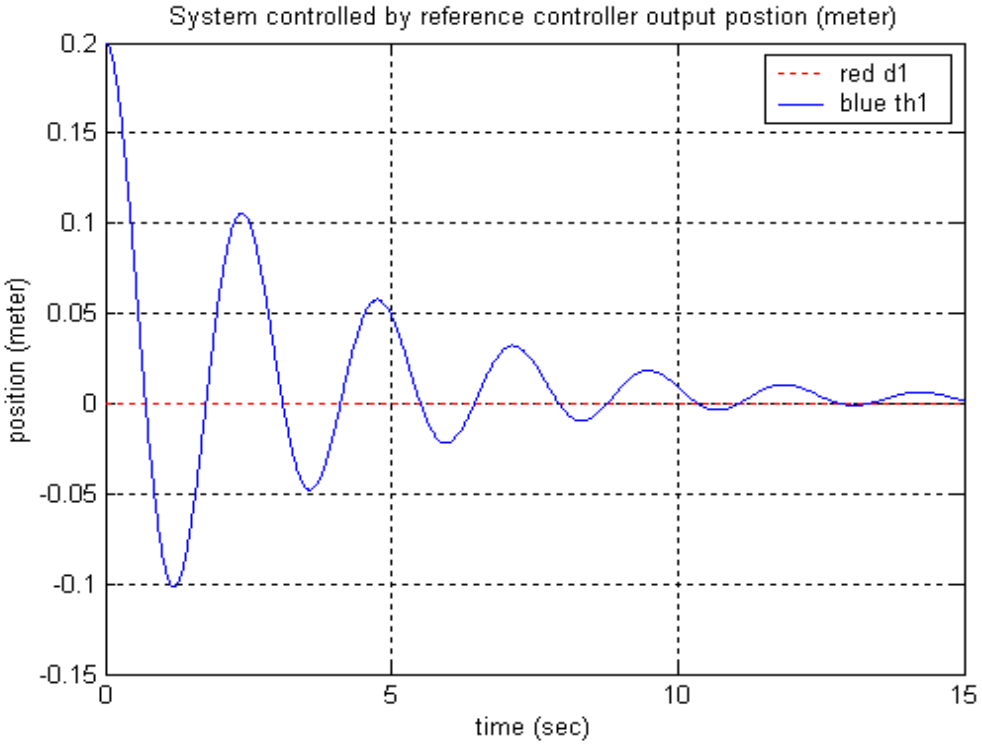

Fig-4.36. The ball's position of BABS controlled by the nonlinear reference controller

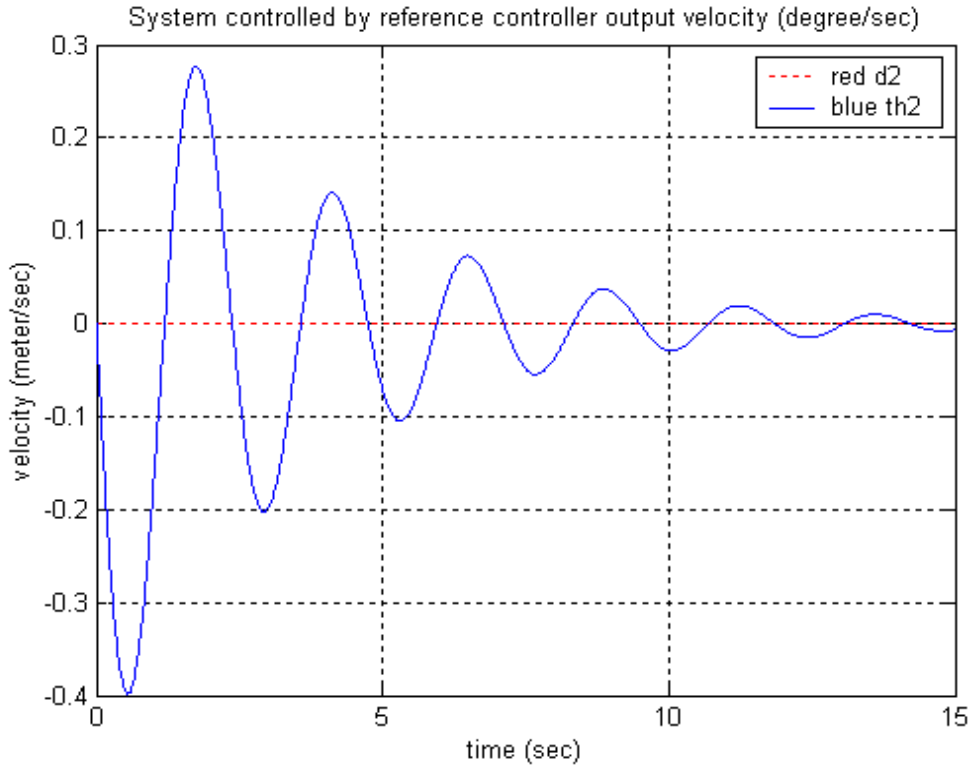

Fig-4.37. The ball's velocity of BABS controlled by the nonlinear reference controller

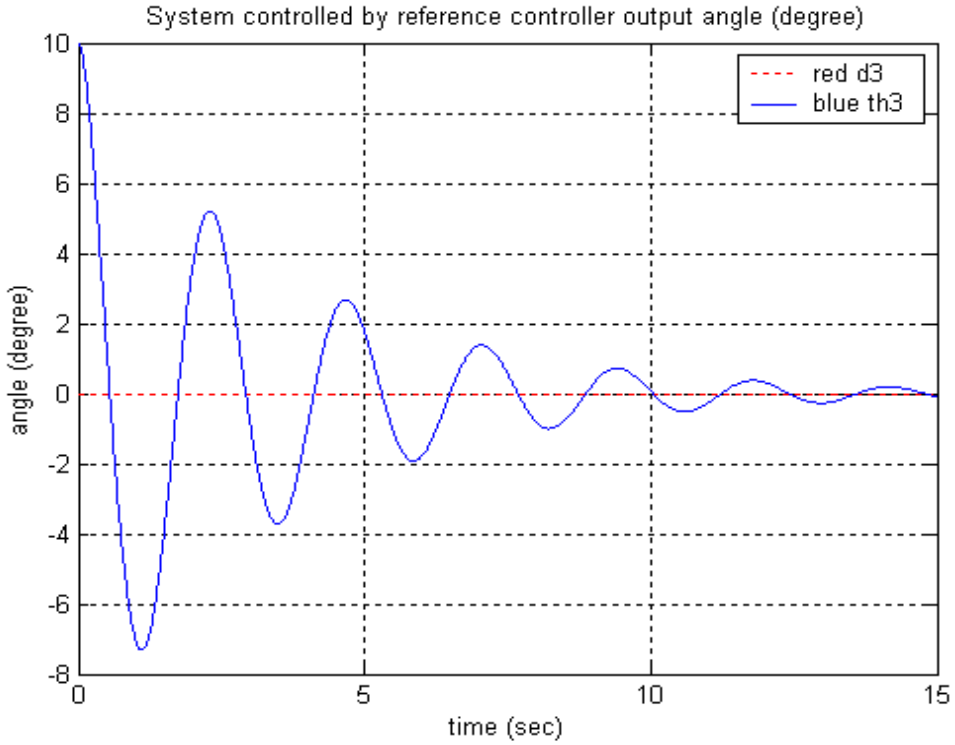

Fig-4.38. The beam's angle of BABS controlled by the nonlinear reference controller

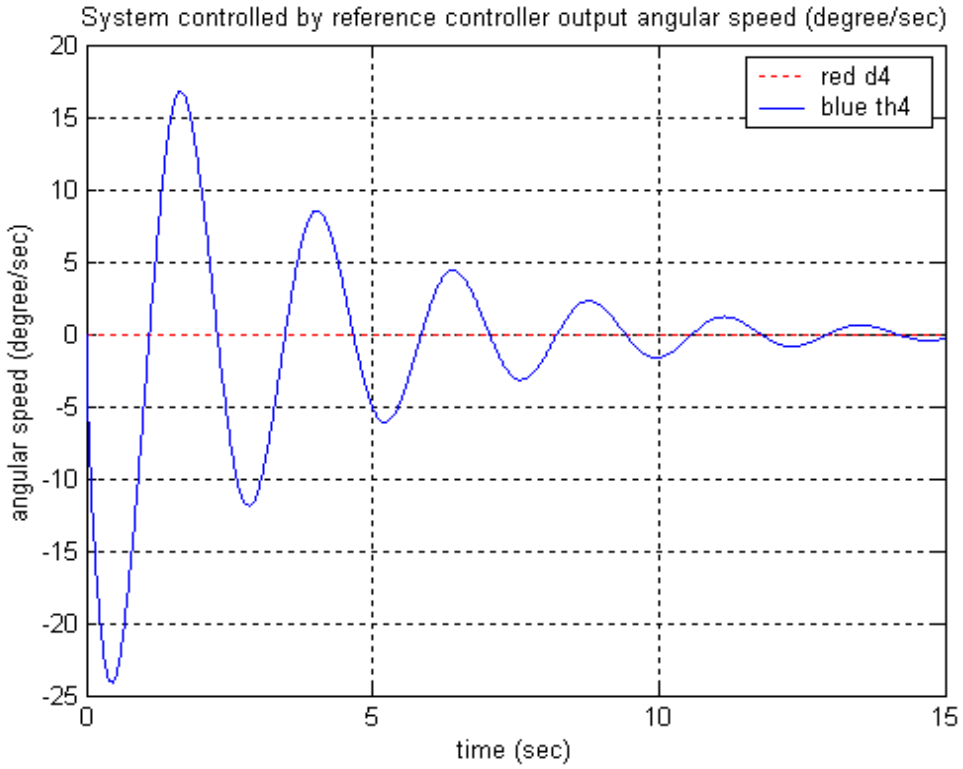

Fig-4.39. The beam's angular speed of BABS controlled by the nonlinear reference controller

We use the same parameters of HNN controller: the resistor=1(ohm), the capacitor=0.01(Farr), the amplification constant=-30, the learning rate=0.001, and 200 epochs for training, and we show the training process as the following two figures:

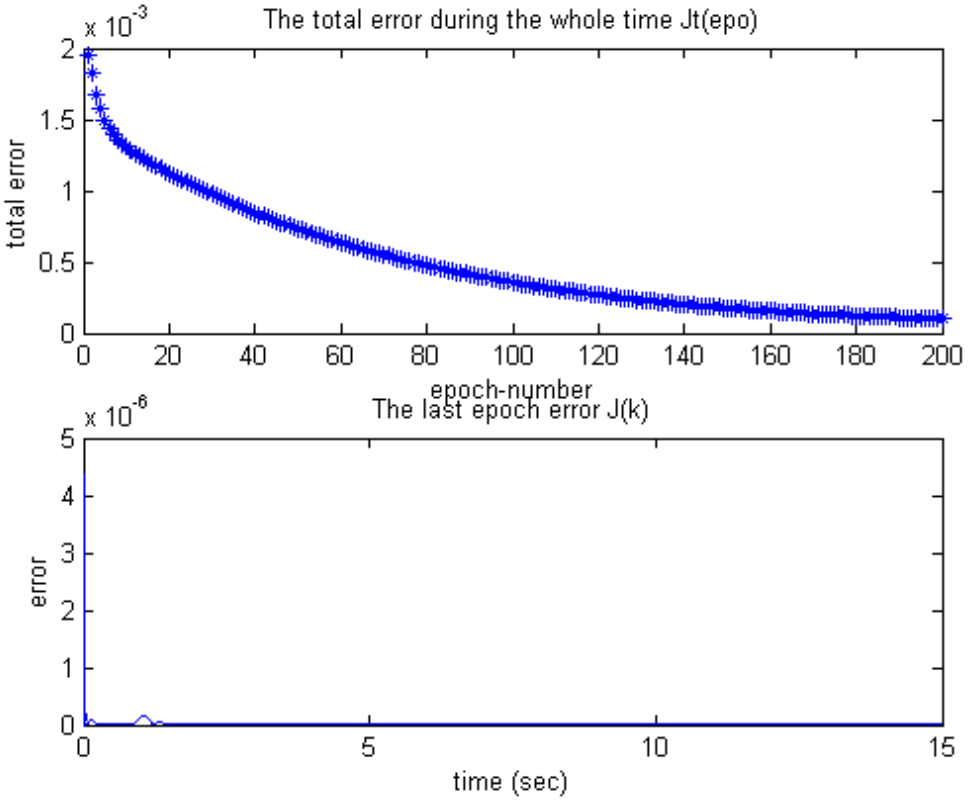

Fig-4.40. The total error $J_t(epo)$ and the last epoch error $J(k,200)$

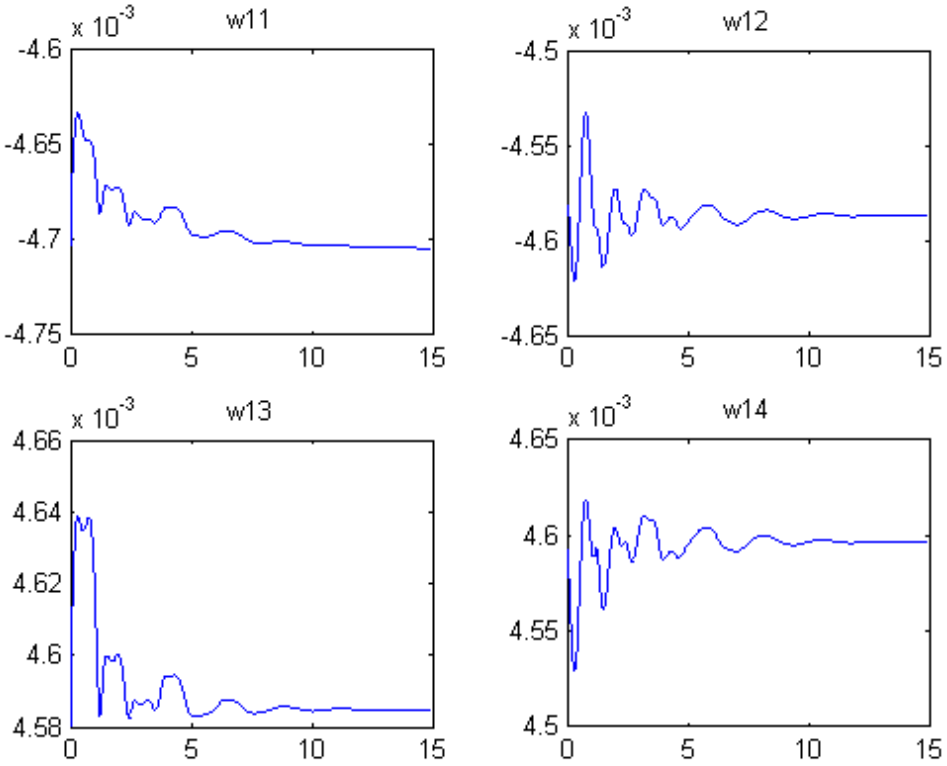

Fig-4.41. The last epoch training iteration process of $w_{11}(w_{21}, w_{31}, w_{41}), w_{12}(w_{22}, w_{32}, w_{42}), w_{13}(w_{23}, w_{33}, w_{43})$ and $w_{14}(w_{24}, w_{34}, w_{44})$

So, we find the values of w_{11}, w_{12}, w_{13}, w_{14}, w_{21}, w_{22}, w_{23}, w_{24}, w_{31}, w_{32}, w_{33}, w_{34}, w_{41}, w_{42}, w_{43} and w_{44}, and we write it down as following equation in the matrix form:

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{bmatrix} = \begin{bmatrix} -0.00470 & -0.00459 & 0.00458 & 0.00460 \\ -0.00470 & -0.00459 & 0.00458 & 0.00460 \\ -0.00470 & -0.00459 & 0.00458 & 0.00460 \\ -0.00470 & -0.00459 & 0.00458 & 0.00460 \end{bmatrix} \quad (4\text{-}41)$$

Then, we show the following six figures as simulation result of BABS controlled by trained HNN controller:

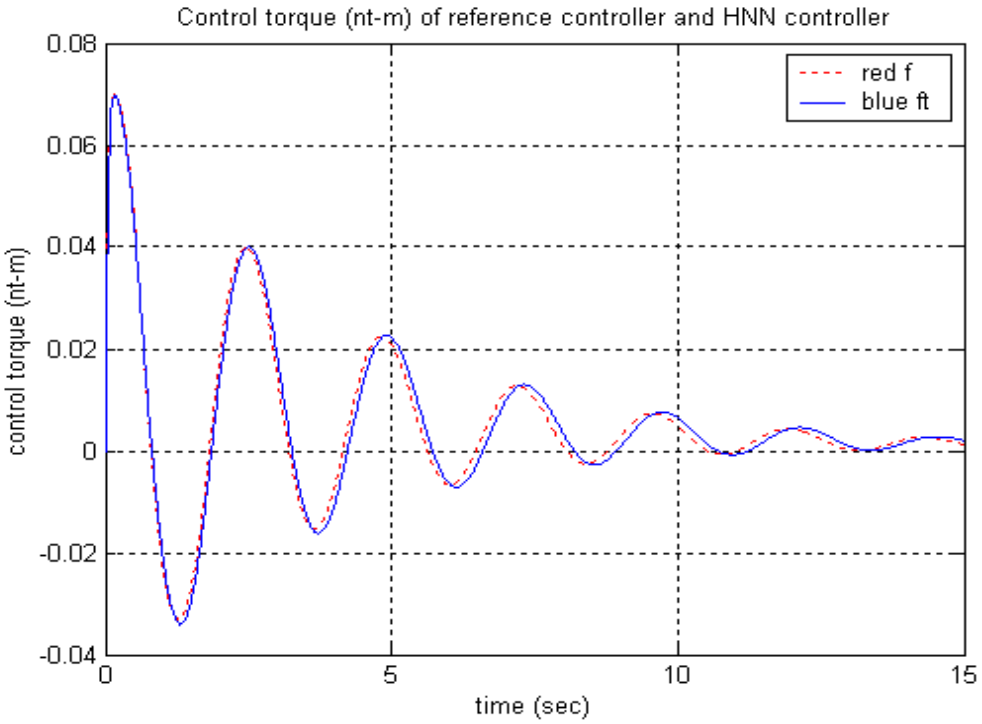

Fig-4.42. The control torques of BABS: the nonlinear reference controller (dash line) and HNN controller (solid line)

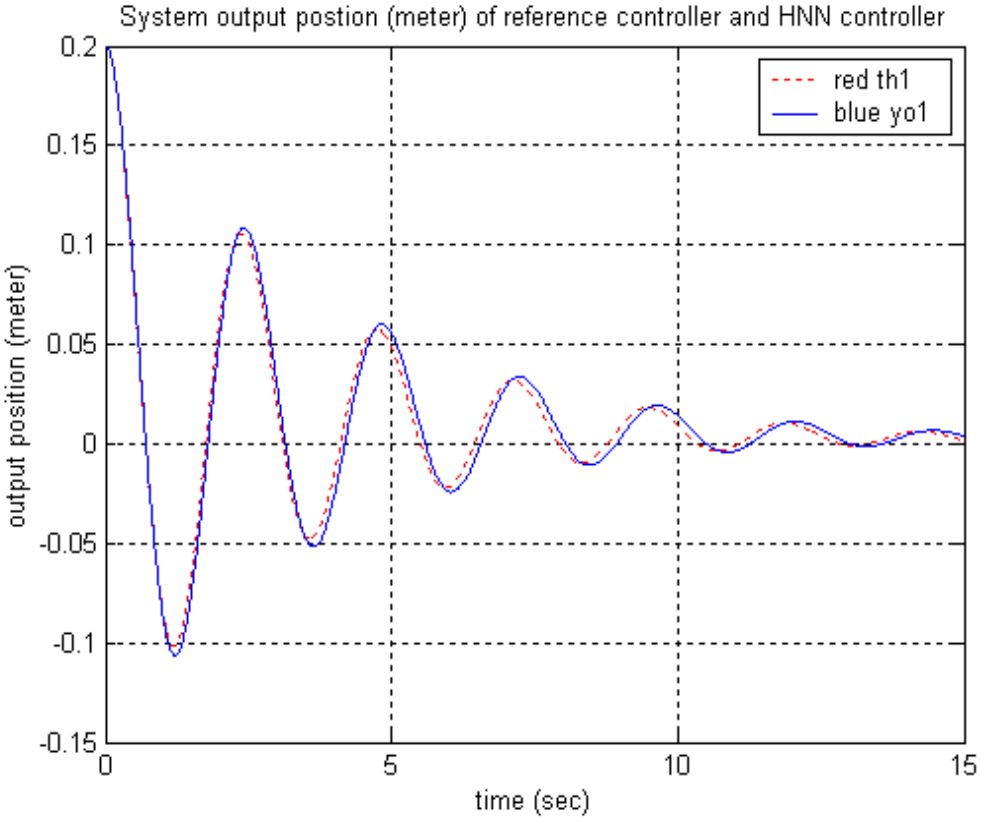

Fig-4.43. The ball's positions of BABS: the nonlinear reference controller (dash line) and HNN controller (solid line)

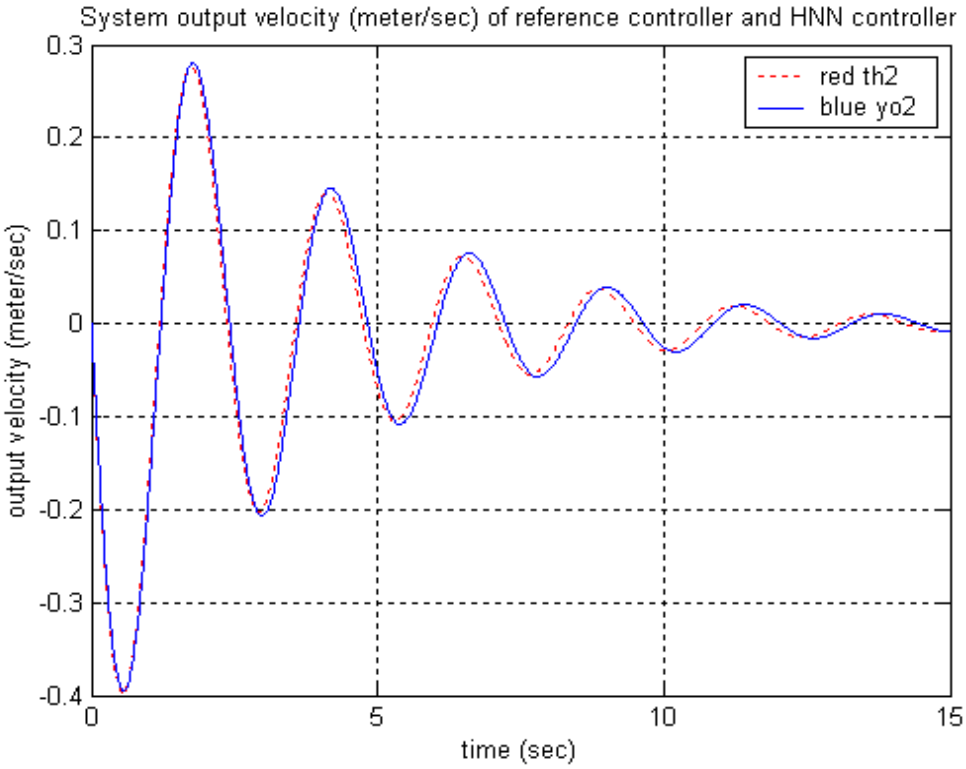

Fig-4.44. The ball's velocities of BABS: the nonlinear reference controller (dash line) and HNN controller (solid line)

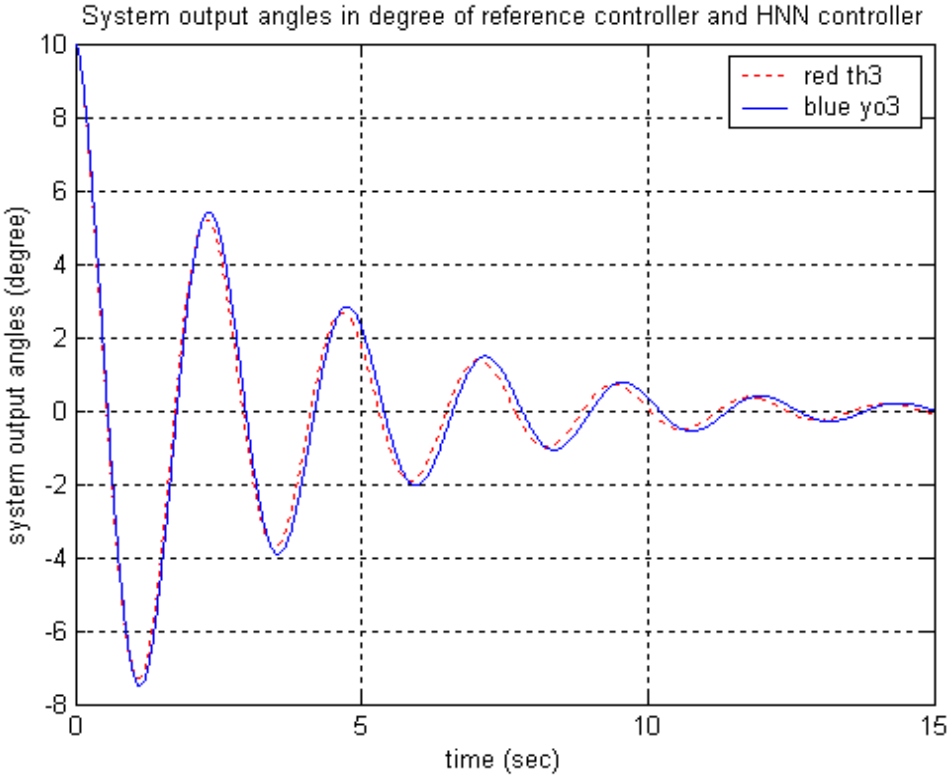

Fig-4.45. The beam's angles of BABS: the nonlinear reference controller (dash line) and HNN controller (solid line)

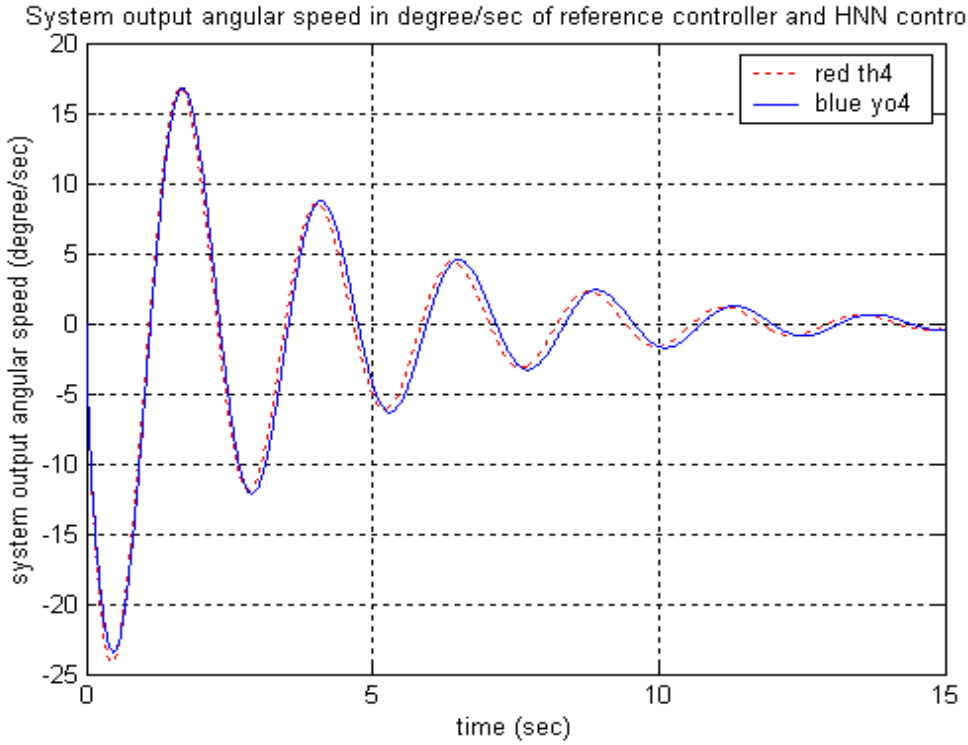

Fig-4.46. The beam's angular speeds of BABS: the nonlinear reference controller (dash line) and HNN controller (solid line)

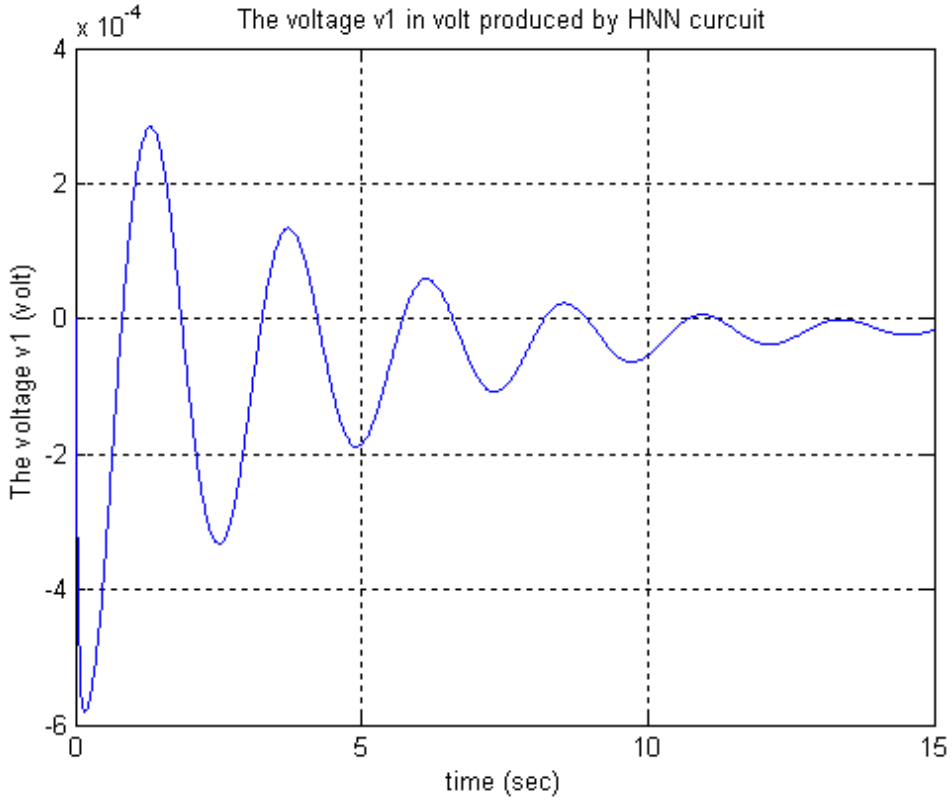

Fig-4.47. The node 1 (2) (3) (4) voltage v_1 (v_2) (v_3) (v_4) of the HNN circuit

4.4.5 BABS Controlled by HNN Controller Trained by Nonlinear Reference Controller with Different Initial State

We use the following examples to examine the abilities of HNN controllers to control IPS with initial state different from the initial state of training the HNN controller by the nonlinear reference controller. We let the values of w_{11}, w_{12}, w_{13}, w_{14}, w_{21}, w_{22}, w_{23}, w_{24}, w_{31}, w_{32}, w_{33}, w_{34}, w_{41}, w_{42}, w_{43} and w_{44} be the same with section 4.4.3 as the equation (4-41):

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{bmatrix} = \begin{bmatrix} -0.00470 & -0.00459 & 0.00458 & 0.00460 \\ -0.00470 & -0.00459 & 0.00458 & 0.00460 \\ -0.00470 & -0.00459 & 0.00458 & 0.00460 \\ -0.00470 & -0.00459 & 0.00458 & 0.00460 \end{bmatrix}$$

is fixed for the initial state of BABS: the initial position of the ball=0.2 (meter),the initial velocity of the ball=0 (meter/sec),the initial angle of the beam=10 (degree), and the initial angular speed of the beam of the BABS=0 (degree/sec) in the training phase. And then, we will examine the following sets (initial position (meter), initial velocity (meter/sec), initial angle (degree), initial angular speed (degree/sec)) of the initial state of BABS: (0.1, 0, 5, 0) and (0.4, 0, 20, 0) in the working phase as the following figures:

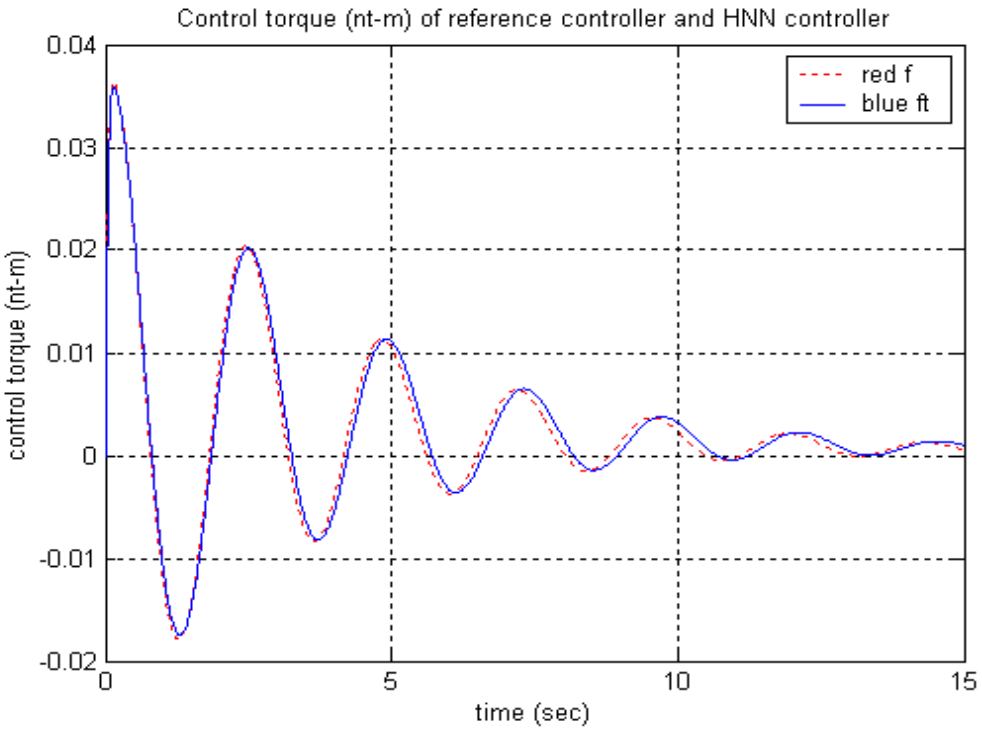

Fig-4.48. The control torques of BABS with initial ball's position=0.1 meter, initial beam's angle=5 degree: the nonlinear reference controller (dash line) and HNN controller (solid line)

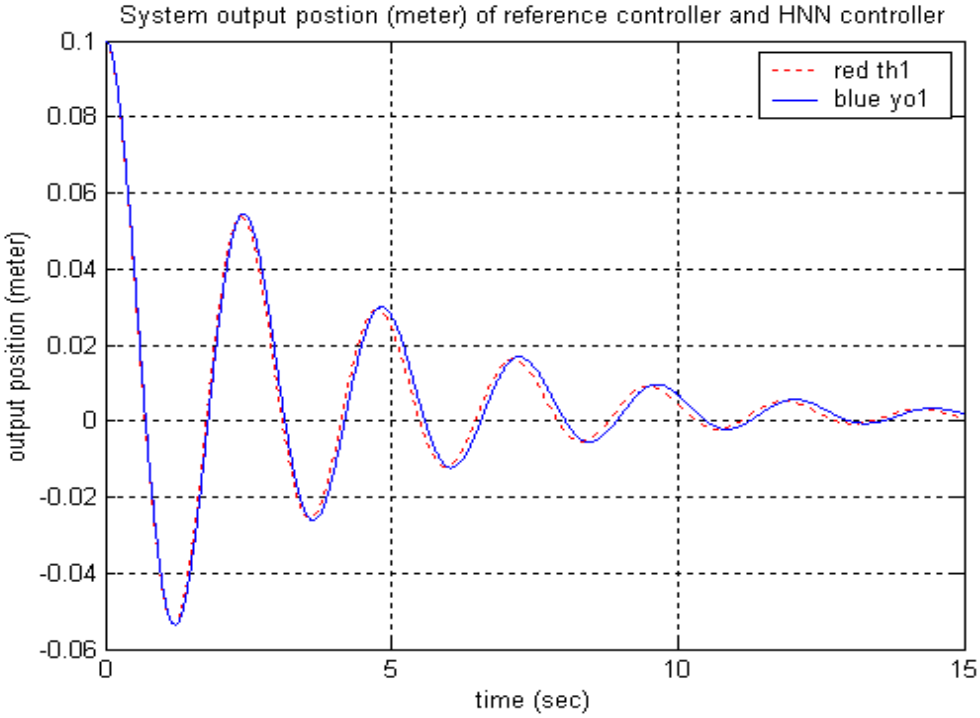

Fig-4.49. The ball's positions of BABS with initial ball's position=0.1 meter, initial beam's angle=5 degree: the nonlinear reference controller (dash line) and HNN controller (solid line)

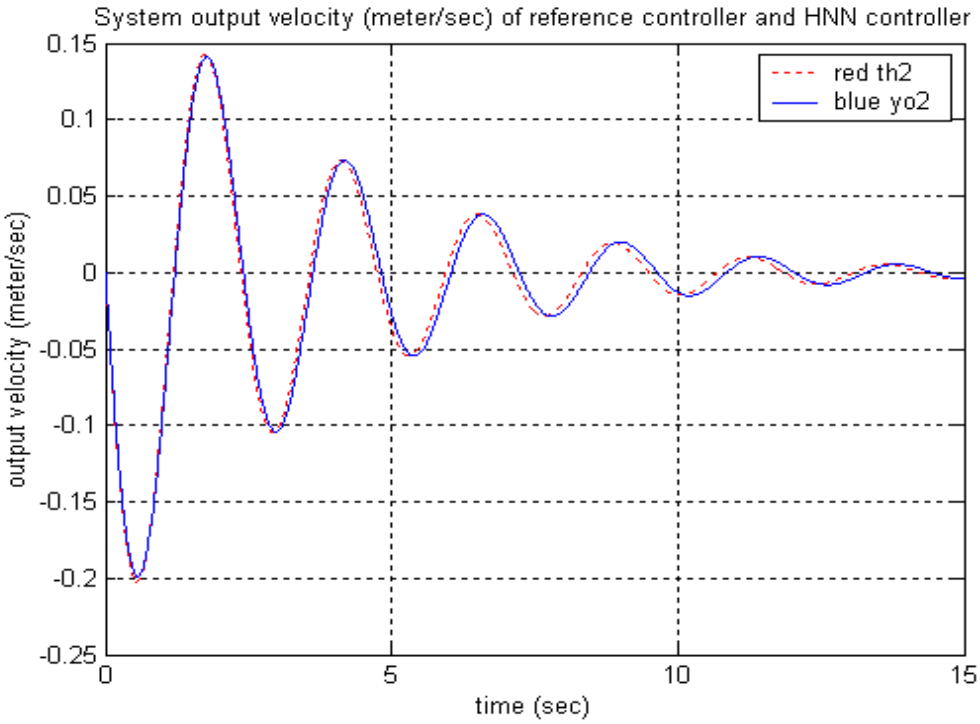

Fig-4.50. The ball's velocities of BABS with initial ball's position=0.1 meter, initial beam's angle=5 degree: the nonlinear reference controller (dash line) and HNN controller (solid line)

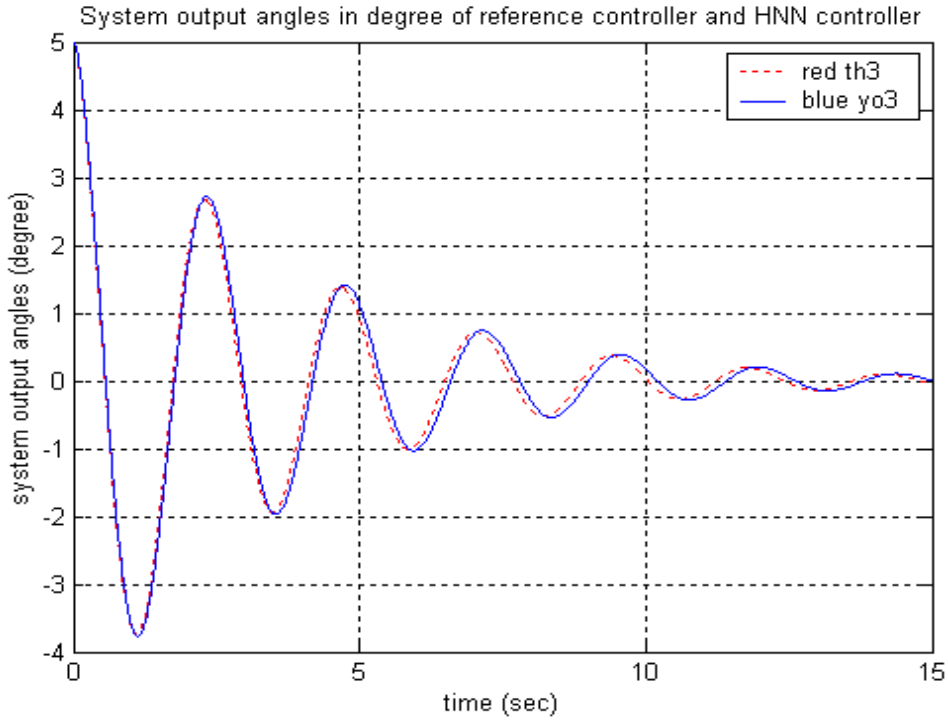

Fig-4.51. The beam's angles of BABS with initial ball's position=0.1 meter, initial beam's angle=5 degree: the nonlinear reference controller (dash line) and HNN controller (solid line)

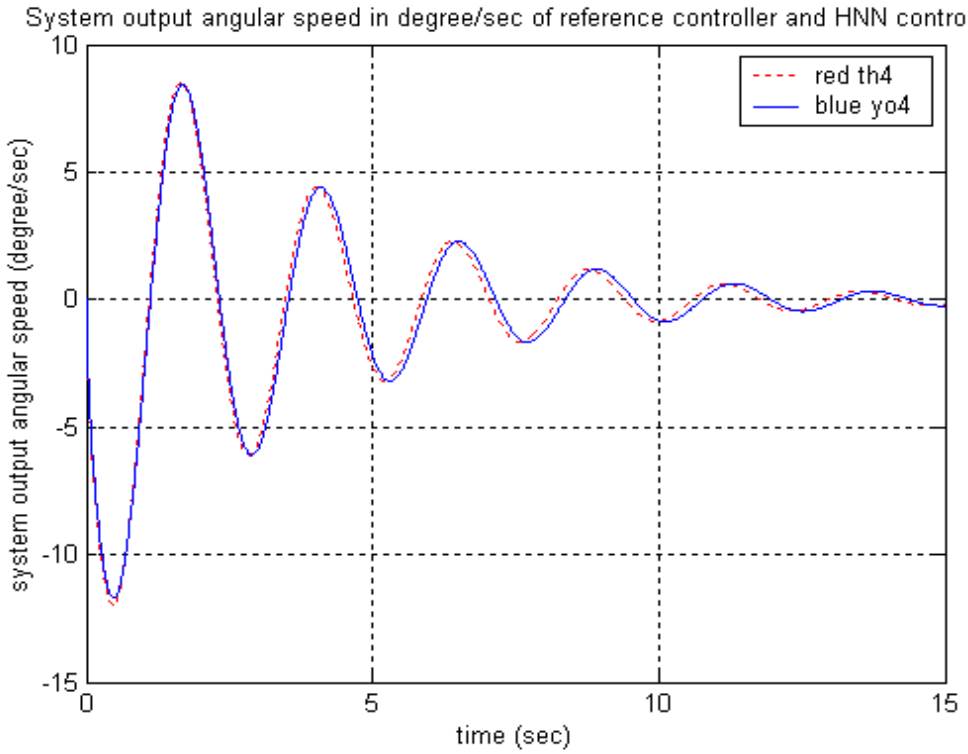

Fig-4.52. The beam's angular speeds of BABS with initial ball's position=0.1 meter, initial beam's angle=5 degree: the nonlinear reference controller (dash line) and HNN controller (solid line)

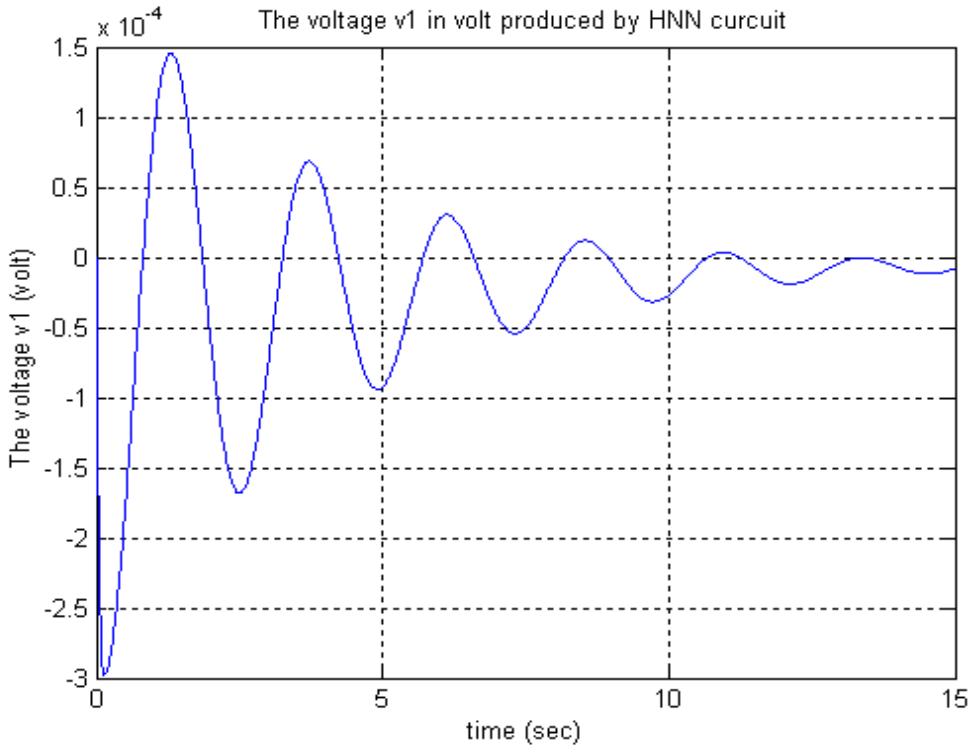

Fig-4.53. The node 1 (2) (3) (4) voltage v_1 (v_2) (v_3) (v_4) of the HNN circuit

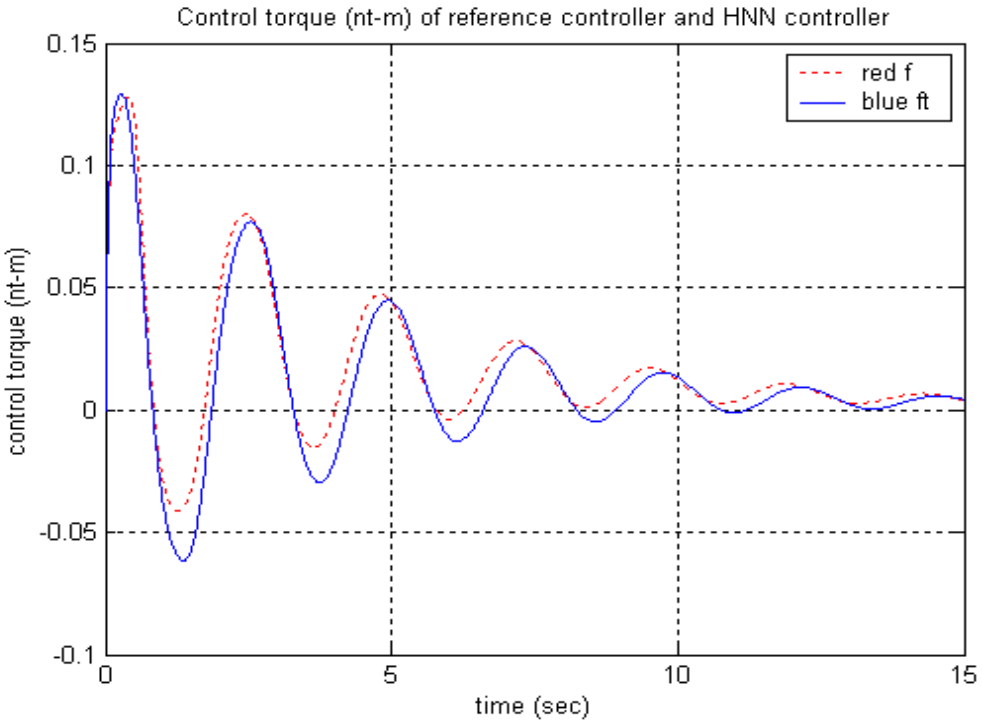

Fig-4.54. The control torques of BABS with initial ball's position=0.4 meter, initial beam's angle=20 degree: the nonlinear reference controller (dash line) and HNN controller (solid line)

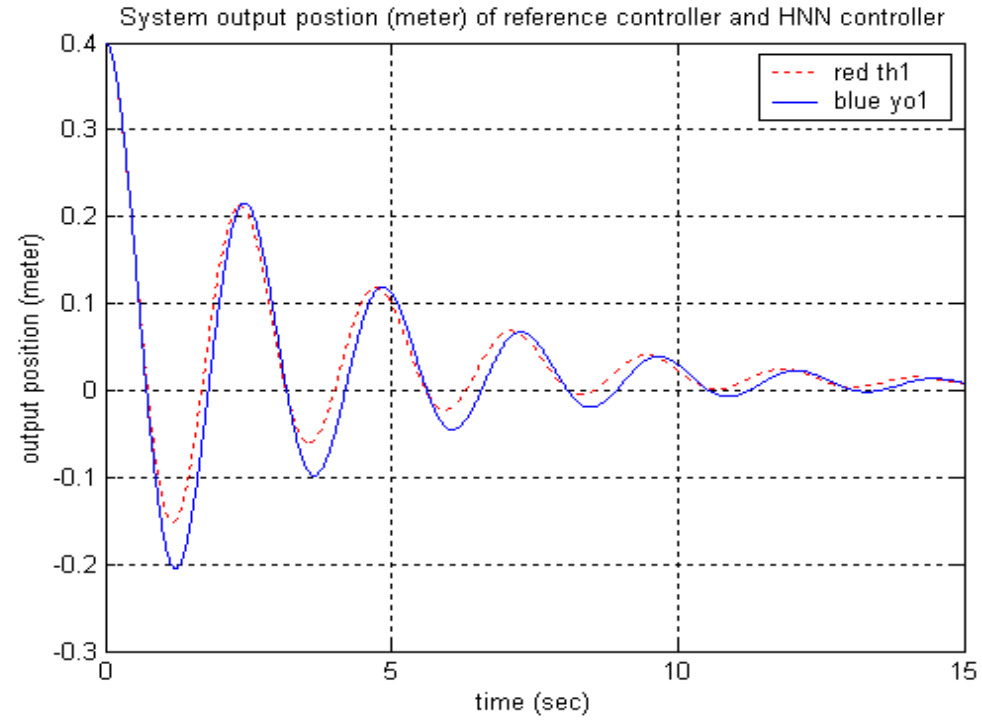

Fig-4.55. The ball's positions of BABS with initial ball's position=0.4 meter, initial beam's angle=20 degree: the nonlinear reference controller (dash line) and HNN controller (solid line)

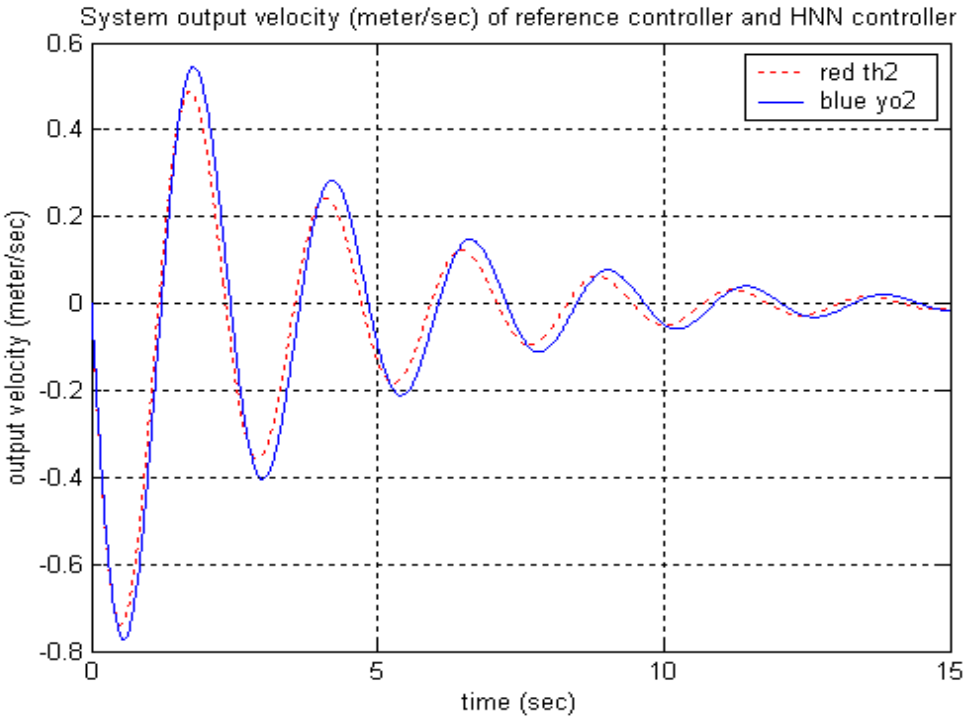

Fig-4.56. The ball's velocities of BABS with initial ball's position=0.4 meter, initial beam's angle=20 degree: the nonlinear reference controller (dash line) and HNN controller (solid line)

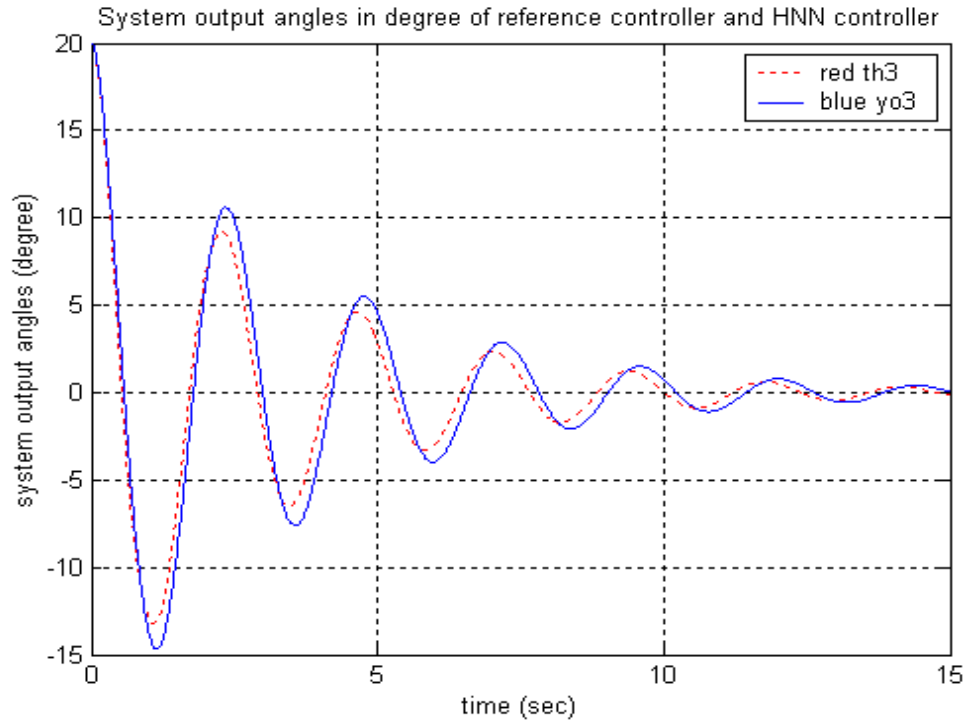

Fig-4.57. The beam's angles of BABS with initial ball's position=0.4 meter, initial beam's angle=20 degree: the nonlinear reference controller (dash line) and HNN controller (solid line)

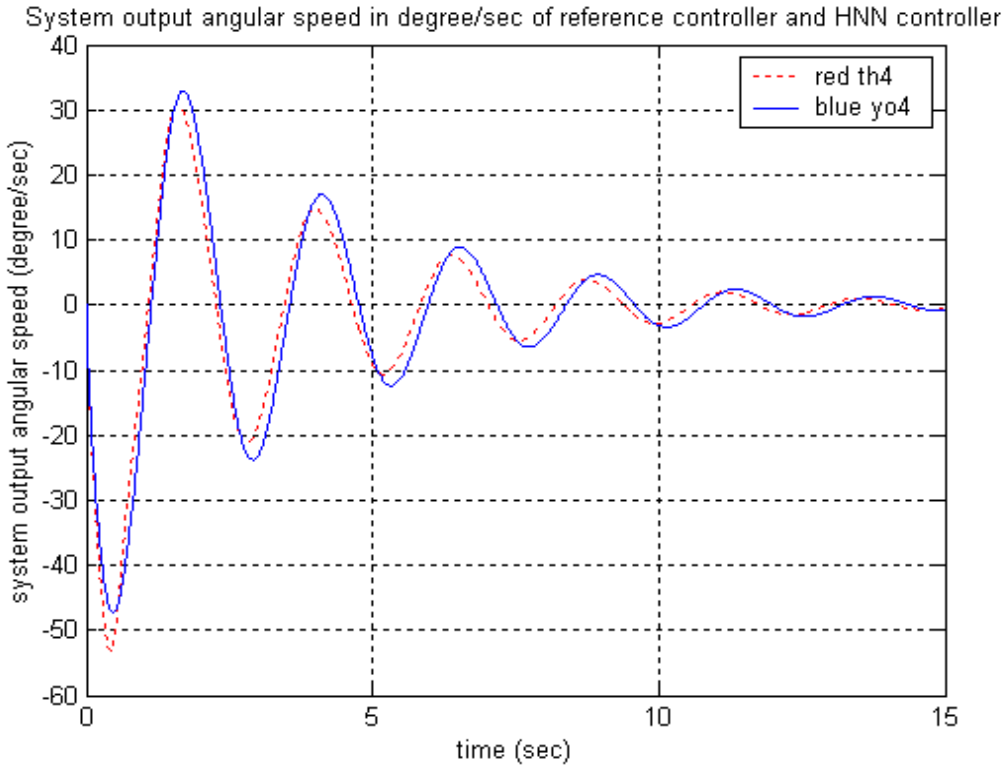

Fig-4.58. The beam's angular speeds of BABS with initial ball's position=0.4 meter, initial beam's angle=20 degree: the nonlinear reference controller (dash line) and HNN controller (solid line)

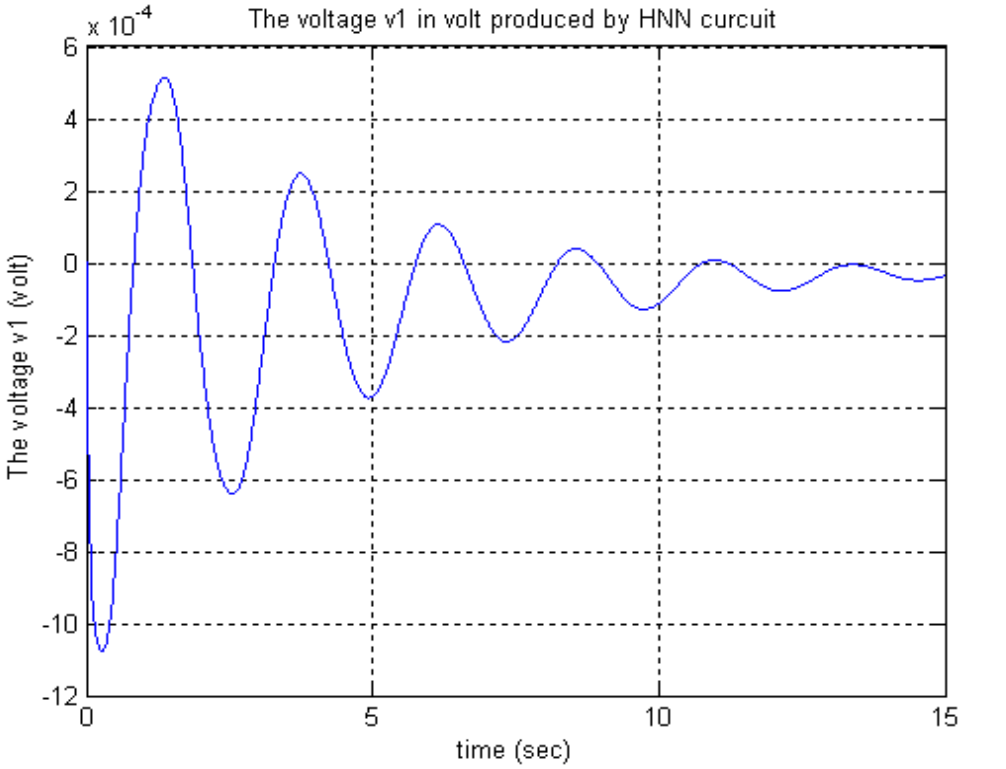

Fig-4.59. The node 1 (2) (3) (4) voltage v_1 (v_2) (v_3) (v_4) of the HNN circuit

Chapter 5

Discussion and Conclusion

5.1 Discussion of Parameters Setting

The resistance R of the HNN controller by the way that larger resistance causes larger node voltage v. Because a large voltage is not preferred, therefore a small R should be chosen. The time constant τ can be express as

$$\tau = RC \quad , \tag{5-1}$$

which is the product of the resistance R and the capacitance C. τ cannot be chosen too large because it leads to slow response. The amplification constant c_{amp} is also an important parameter of the HNN controller. According to the voltage amplifier $\varphi(\bullet) = \tanh(\bullet)$, the output of a neuron of the HNN is limited between the values -1 to +1, we express as the following inequality.

$$-1 \le x_j = \tanh(v_j) \le 1 \tag{5-2}$$

From the inequality (5-2), we can show that the output control signal of HNN controller is limited as

$$-nc_{amp} \le \hat{u} = c_{amp} \sum_{j=1}^{n} x_j \le nc_{amp} \quad , \tag{5-3}$$

where n is the number of neurons in the HNN. That is the absolute value of the output control signal of the HNN controller is limited by the product of the amplification constant c_{amp} and the number of neurons n.

The learning rate η must be a positive as we discussed in the section 2.3. The large η is not preferred because that will contradict (2-14) [7]. However η cannot be chosen too small or the weightings convergence will be too slow, so we use a proper value of the learning rate η to let the simulation process well.

The simulation time is chosen long enough so that the regulation state of the controlled system can be near the desired point.

The time interval is set to be a small enough value to get the required accuracy of the differential equations of the system controlled by the HNN controller.

In many cases one epoch is enough to achieve favorable training performance. More training epoch is unnecessary unless one epoch training cannot achieve the preferred performance.

If we do not have prior knowledge of the proper weighting vector, we can just

simple initialize W from zero vector. Notice that the values of the elements of W are the same in the same column. This can be briefly explained as following. We note the equations (3-22) and (3-24), they are very similar. We write them down in the simple form to show the difference. For (3-22), the equation is $w_{11}(k+1) = w_{11}(k) - ... \times \{1 - [\tanh(v_1)]^2\}$, while for (3-24), the equation is $w_{21}(k+1) = w_{21}(k) - ... \times \{1 - [\tanh(v_2)]^2\}$. And we note the equations (3-10) and (3-11), we find if $w_{11} = w_{21}$ and $w_{12} = w_{22}$, then we can get $i_1 = i_2$ by equation (3-11), and furthermore, we can get $v_1 = v_2$ by equation (3-10). So it is interesting that because we set the initial values of all the weighting factors zeros, so $w_{11}(0) = w_{21}(0) = 0$ and $w_{12}(0) = w_{22}(0) = 0$, so $i_1(0) = i_2(0)$ and $v_1(0) = v_2(0)$. According to the fact $w_{11}(0) = w_{21}(0)$ and $v_1(0) = v_2(0)$, we can find $w_{11}(1) = w_{21}(1)$ by the equations (3-22) and (3-24), and $w_{12}(1) = w_{22}(1)$ by the equations (3-23) and (3-25). Thus, we have $i_1(1) = i_2(1)$ and $v_1(1) = v_2(1)$, and so on. At last, the important facts are obtained as following:

$$\begin{aligned} & i_1(k) = i_2(k) = ... = i_n(k) \\ & v_1(k) = v_2(k) = ... = v_n(k) \\ & w_{1j}(k) = w_{2j}(k) = ... = w_{nj}(k),\ j = 1,2,...,n \end{aligned} \tag{5-4}$$

It should keep in mind that the equation (5-4) is satisfied on the premise of the following:

$$\begin{aligned} & R_1 = R_2 = ... = R_n \\ & w_{1j}(0) = w_{2j}(0) = ... = w_{nj}(0),\ j = 1,2,...,n \end{aligned} \tag{5-5}$$

5.2 Conclusion

After training the weighting factors of HNN controller, we find that the output (control signal) of HNN controller is approximated to the output (control signal) of the well-designed controller and the outputs of the plant controlled by the HNN controller are approximated to the outputs of the plant controlled by the well-designed controller. So, the trained HNN controller can be a good model of the well-designed controller with the controlled plant with the same initial state with HNN trained. Furthermore, although the weighting factors of HNN controller are trained by well-designed controller with the plant with initial state different from the initial state of the plant controlled by the HNN controller, the HNN controller can still control the plant well, and the output (control signal) of HNN controller is approximated to the output (control signal) of the well-designed controller and the outputs of the plant controlled by the HNN controller are still approximated to the outputs of the plant controlled by the well-designed controller. So, the trained HNN controller not only can “memorize” the output (control signal) of the well-designed

controller with the controlled plant in the same initial state, but it can also “simulate” the output (control signal) of the well-designed controller with the controlled plant in the different initial state. So, the HNN controller has ability more than just to memorize the training data, and this property of HNN controller is important for applications.

Faults due to the aging of a controller for a control system are very common; once they happen, the controller is quite difficult to be repaired for some reasons. We proposed an HNN controller for a control system to solve this problem. After discussing the two examples of the nonlinear systems controlled by the HNN controllers, we understand that the HNN has the potential to be a good controller. The key point of the HNN controller is the parameters, especially the weighting factors between each two neurons of one HNN. To design an HNN controller for some specified nonlinear system is still a challenge. In this thesis, we trained the weighting factors of the HNN controller to mimic the existing controller. Then, the trained HNN controller is used to replace the existing controller. Can we control the system by an HNN controller trained online without a reference controller? We will focus our research interests on exploring the potential of this interesting question.

References

[1] K. Mehrotra, C. K. Mohan, and S. Ranka, "Elements of Artificial Neural Networks," *The MIT Press Cambridge*,1996.

[2] L. R. Medsker and L. C. Jain, "Recurrent Neural networks: Design and Applications," *CRC Press LLC,* 2000.

[3] C. T. Lin, and C. S. G. Lee, "Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems," *Prentice Hall P T R*,1996.

[4] S. K. Pal and S. Mitra, "Neuro-Fuzzy Pattern Recognition: Method in Sift Computing" Networks: A Comprehensive Foundation," *John Wiley & Sons, INC.,* 1999.

[5] M. Friedman and A. Kandel, "Introduction to Pattern Recognition: Statistical, Structural, Neural and Fuzzy Logic Approaches," *Imperial College Press, 1999*.

[6] T. W. S. Chow, X. D. Li, and Y. Fang, "A Real-Time Learning Control Approach for Nonlinear Continuous-Time System Using Recurrent Neural Networks," *IEEE Transactions on Industrial Electronics,* vol. 47, pp 478-486, 2000.

[7] S. Haykin, "Neural Networks: A Comprehensive Foundation," *Prentice-Hall,* second edition, 1999.

[8] A. Delgado, C. Karnbhampati, and K. Warwick, "Dynamic recurrent neural network for system identification and control," *IEE proceedings. Control theory and applications,* vol.142, pp307-314, 1995.

[9] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two states neurons," *Proceedings of National Academy of sciences, USA,* vol.81, pp3088-3092, 1984.

[10] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of National Academy of sciences, USA,* vol.79, pp2554-2558, 1982.

[11] H. Jing and N. Zhao, "Study on the Global Asymptotic Stability of Hopfield Neural Networks," *IEEE International Conference on Control and Automation,* pp2780-2784, 2007.

[12] L. Wang, Y. S. Xiao, G. Zhou, and Q. Wu, "Further Discussion of Hopfield Neural Network based DC Drive System Identification and Control," *IEEE. Proceedings of the 4''' World Congress on Intelligent Control and Automation,* pp1990-1993, 2002.

[13] T. P. Troudet and S. M. Walters, "Neural Network Architecture for Crossbar Switch Control," *IEEE Transactions on Circuit and Systems,* vol. 38, pp42-56,

1991.

[14] T. W. S. Chow and Y. Fang, "A Recurrent Neural-Network-Based Real-Time Learning Control Strategy Applying to Nonlinear Systems with Unknown Dynamics," *IEEE Transactions on Industrial Electronics,* vol. 45, pp151-161, 1998.

[15] R. Craddock and C. Kambhampati, "Trained Hopfield Neural Networks Need Not Be Black-boxes," *Proceedings of the American Control Conference,* pp368-372, 1999.

[16]N. C. Kan, "Design of Hopfield Neural Network Controller with Its applications," *MS Thesis, Department of Electrical and Control Engineering, NCTU, Hsin-Chu, Taiwan,* 2006.

[17] Z. B. Xu and C. P. Kwong, "Global Convergence and Asymptotic Stability of Asymmetric Hopfield Neural Networks," *Journal of Mathematical Analysis and Applications,* vol. 191, pp405-427, 1995.

[18] L. X. Wang, "Adaptive Fuzzy Systems and Control: Design and Stability Analysis," *PTR Prentice Hall,*1994.

[19] Y. L. Li, "Coupled Derivatives Compact Schemes for One-Dimensional KDV Equation," *MS Thesis, Department of Applied Mathematics, NCTU, Hsin-Chu, Taiwan,* 2007.

[20] Y. F. Tung, "Multi-Degrees of Freedom H∞ Controller Design for the Inverted Pendulum to Fix Position," *MS Thesis, Institute of Mechanical Engineering, NCTU, Hsin-Chu, Taiwan,* 2004.

[21] R. C. Dorf and Robert H. Bishop, "Modern Control Systems," *Addison-Wesley,* eighth edition, 1998.

[22] J. Hauser, S. Sastry, and P. Kokotovic, "Nonlinear Control Via Approximate Input-Output Linearization: The Ball and Beam Example," *IEEE Transactions on Automatic Control,* vol. 37, pp392-398, 1992.