

國立交通大學  
電機與控制工程學系

碩士論文

基於類神經網路的演化策略

應用於一階動態系統



Neural Network-Based Evolution Strategies for  
Implementing First Order Dynamic Systems

研究生：陳思穎

指導教授：陳永平 教授

中華民國九十六年六月

基於類神經網路的演化策略應用於一階動態系統

Neural Network-Based Evolution Strategies for  
Implementing First Order Dynamic Systems

研究生：陳思穎

Student : Sze-Ying Chen

指導教授：陳永平

Advisor : Yon-Ping Chen

國立交通大學

電機與控制工程學系



A Thesis

Submitted to Department of Electrical and Control Engineering  
College of Electrical and Computer Engineering

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of Master

in

Electrical and Control engineering

June 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年六月

# 基於類神經網路的演化策略

## 應用於一階動態系統

學生：陳思穎

指導教授：陳永平 博士

國立交通大學電機與控制工程學系



本論文目的在於利用類神經網路學習一階動態系統，用以控制一迴授系統。論文 3.2 節中，為了學習一階動態系統，提出兩種簡單的類神經網路架構，一種是一般的類神經網路，另一種加入了參數影響，而此參數是根據一階差分方程而來的。另外，本論文不用常見的倒傳遞學習法則，而用一演化策略，因為倒傳遞學習法則必須知道反向動態系統，但反向動態系統並不容易得到，而該演化策略並無此缺點。雖然使用該演化策略的學習時間較長，但類神經網路經過演化策略的學習後，能表現的跟目標系統極度相似，即使將此類神經網路放入一有外來雜訊的迴授系統當控制器時，亦能控制系統穩定不受雜訊影響。

# Neural Network-Based Evolution Strategies for Implementing First Order Dynamic Systems

Student: Sze-Ying Chen

Advisor: Dr. Yon-Ping Chen

Department of Electrical and Control Engineering

National Chiao Tung University



## ABSTRACT

The objective of the thesis is training a neural network to perform as a first order LTI system, and then apply as a controller. Based on evolution strategies and the first order difference equation, two simple neural network structures are designed to implement the first order LTI systems. With the evolution strategies, it is not necessary to know the inverse dynamic system, which is required while using the backpropagation learning algorithm in general neural networks. From the simulation results, the proposed neural networks, called general structure (GS) and structure with sampling time (SST), may perform almost the same as the first order LTI system and are robust to an unexpected disturbance as a controller, even though the learning time is long.

# Acknowledgement

轉瞬間，碩士生涯就要結束了。回想這兩年，每當研究愈深入，愈覺自身之不足，愈覺學海之無涯，幸得指導老師 陳永平老師孜孜不倦的教導，讓論文能順利完成。老師除了在研究上給予建議與指導外，對於研究方法與學習態度也相當重視，更提供許多英文能力的訓練，讓我們除了專業上的深入外，有更多其他方面的成長，僅向老師至上最誠摯的謝意與感念。同時，謝謝口試委員 張浚林學長及 梁耀文老師寶貴的建議與指教，讓論文得以更加完善。

其次，也要感謝一路支持我的父母兄長，還有一路陪伴我的朋友麗中、欣宜、依文、君如、心韻，我能度過低潮，堅持到現在，都要歸功於您們。最後，要謝謝實驗室伙伴胤宏、士昌、子揚、坤佑的加油打氣，以及實驗室世宏學長、建峰學長和桓展學長的指教，還要謝謝這一年來相伴的學弟妹們，由於你們，讓平凡的研究生活更點綴了許多歡笑的回憶。

僅以此篇論文獻給所有關心、照顧我的人。

陳思穎 2007.07.03

# Contents

Chinese Abstract.....	i
English Abstract .....	ii
Acknowledgement .....	iii
Contents.....	iv
List of Figures .....	vi
List of Tables .....	viii
Notation .....	ix
Chapter 1 Introduction.....	1
Chapter 2 Intelligent Control.....	3
2.1 Introduction to Neural Network .....	3
2.2 Neural Network for Control .....	8
Chapter 3 Evolution Strategies .....	15
3.1 Modeling of First Order LTI System.....	15
3.2 Neural Network Structure .....	18
3.2.1 Structure for The First Order Difference Equation .....	18
3.2.2 General Structure .....	19
3.2.3 Structure with Sampling Time .....	21
3.3 Evolution Strategies.....	22
3.3.1 Initial Individuals Creation .....	24
3.3.2 Reproduction Process .....	25
3.3.3 Learning Process.....	28
3.3.4 Elite Process .....	30
3.3.5 Flow Chart .....	33

<b>Chapter 4 Simulation Results.....</b>	<b>34</b>
<b>4.1 Influence of The Sampling Time .....</b>	<b>35</b>
<b>4.2 Influence of The Initial Weights Setting.....</b>	<b>49</b>
<b>4.3 Implement as a Controller .....</b>	<b>55</b>
4.3.1 System with First Order LTI Plant .....	56
4.3.2 System with Second Order LTI Plant .....	58
<b>Chapter 5 Conclusion.....</b>	<b>62</b>
<b>References .....</b>	<b>64</b>



# List of Figures

Figure 2.1 An artificial neuron .....	4
Figure 2.2 Multilayer feedforward network .....	5
Figure 2.3 two-layer back-propagation network .....	6
Figure 2.4 the feedback system .....	8
Figure 2.5 to train neural network plant .....	9
Figure 2.6 to train the neural network controller.....	10
Figure 2.7 the fully recurrent neural network.....	11
Figure 3.1 the first order difference equation using one neuron .....	19
Figure 3.2 the GS for first order LTI system .....	20
Figure 3.3 the SST for first order LTI system .....	22
Figure 3.4 Illustration of finding children inward .....	26
Figure 3.5 Illustration of finding children outward .....	27
Figure 3.6 Random children creation .....	27
Figure 3.7 finding temporal individual of the next step .....	29
Figure 3.8 the flow chart of the evolution strategies .....	33
Figure 4.1.1 the learning result of the GS under sampling time 0.01.....	36
Figure 4.1.2 the change of the sum of the error of the GS under sampling time 0.01 .....	36
Figure 4.2.1 the learning result of the SST under sampling time 0.01 .....	37
Figure 4.2.2 the change of the sum of the error of the SST under sampling time 0.01.....	37
Figure 4.3.1 the learning result of the GS under sampling time 0.001.....	38
Figure 4.3.2 the change of the sum of the error of the GS under sampling time 0.001 .....	38
Figure 4.4.1 the learning result of the SST under sampling time 0.001 .....	39
Figure 4.4.2 the change of the sum of the error of the SST under sampling time 0.001.....	39
Figure 4.5.1 testing result when initial condition $y(0)$ is 0.5.....	42
Figure 4.5.2 testing result when initial condition $y(0)$ is 2.....	43
Figure 4.5.3 testing result when initial condition $y(0)$ is -2 .....	43
Figure 4.6.1 testing result when input function $u = 2p(t)$ .....	44
Figure 4.6.2 testing result when input function $u = -2p(t)$ .....	44
Figure 4.6.3 testing result when input function $u = \sin(2t)$ .....	45
Figure 4.7 the testing result with .....	46
Figure 4.8 the testing result with $\Delta T=0.001$ of the learned neural network with $\Delta T=0.01$ .....	46
Figure 4.9 the learning result of the SST with sampling time 0.01, 0.001, 0.0001 .....	47
Figure 4.10.1 the testing result under the sampling time smaller than trained sampling time .....	48
Figure 4.10.2 the testing result under the sampling time between trained sampling time .....	48
Figure 4.10.3 the testing result under the sampling time larger than trained sampling time .....	49



Figure 4.11.1 the learning result of the GS under sampling time 0.001 .....	51
Figure 4.11.2 the change of the sum of the error of the SST under sampling time 0.001 .....	51
Figure 4.12.1 testing result when initial condition $y(0)$ is 0.5.....	52
Figure 4.12.2 testing result when initial condition $y(0)$ is 2.....	53
Figure 4.12.3 testing result when initial condition $y(0)$ is -2 .....	53
Figure 4.13.1 testing result when input function $u = 2p(t)$ .....	54
Figure 4.13.2 testing result when input function $u = -2p(t)$ .....	54
Figure 4.13.3 testing result when input function $u = \sin(2t)$ .....	55
Figure 4.14 the feedback system .....	56
Figure 4.15 the learning result of a controller: $\dot{u}(t) + 6.5u(t) = 2.5e(t)$ .....	56
Figure 4.16 the testing result of the feedback system with a neural network controller.....	57
Figure 4.18 the testing result of the feedback system with unexpected disturbance.....	58
Figure 4.19 the feedback system .....	58
Figure 4.20 the testing result of the feedback system with a neural network controller.....	59
Figure 4.21 the feedback controller with disturbance .....	60
Figure 4.22 the testing result of the feedback system with unexpected disturbance.....	60



## List of Tables

Table 4.1 the results of learning with different sampling times .....	40
Table 4.2.1 the error and the accurate rate of learning with sampling time 0.01 .....	40
Table 4.2.2 the error and the accurate rate of learning with sampling time 0.001 .....	41
Table 4.3 learning results of GS with sampling time 0.001 .....	50



# Notation

GS : General structure

SST : Structure with sampling time

$W_k^g$  : The  $k^{\text{th}}$  individual of the  $g^{\text{th}}$  generation

$n^g$  : The number of the individuals in the  $g^{\text{th}}$  generation

$l$  : The multiple of the distance in the reproduction process

$\Omega_k^g(s)$  : The  $k^{\text{th}}$  child of the  $s^{\text{th}}$  step in the  $g^{\text{th}}$  generation

$\Omega_{\text{imp}}(s)$  : The temporal individual of the  $s^{\text{th}}$  step

$\delta\Omega(s)$  : The difference of the  $s^{\text{th}}$  step

$\delta\Omega^\perp(s)$  : The perpendicular vector of  $\delta\Omega(s)$

$\delta\Omega_{\text{imp}}(s)$  : The temporal difference of the  $s^{\text{th}}$  step



# Chapter 1

## Introduction

In recent years, many researchers have been devoted to developing intelligent theories, such as fuzzy logic [1], genetic algorithm [2-4], and neural network theory [5, 6]. These intelligent theories are used in more and more fields [7, 8] and show their great power as a problem solver.

Many investigators have developed intelligent machines, such as Kismet and Aryan. We also built an eye-robot to mimic the motion of human eyes, and try to control the robot. In the conventional control theory, the model of the plant should be known exactly before designing the controller. If the model has some error from the real system, the designed controller may not stable and control well. However, in reality, many plants are complex and the model could not be determined, thus that restricts the use of the conventional control theory. The controller could not be designed by the conventional control theory when the model of the plant is uncertain. Fortunately, the intelligent control needs no exact model of the plant to design a controller, and can learn to control the system gradually. Thus, the restriction of the conventional control theory is the reason why we use the intelligent theory to control our robot.

Basically, the artificial neural network based on the human neural network contains many neurons connected with synaptic weights. Thus, it can learn, recall, and generalize from training data like a human brain. Models of the neurons, models of the structure and the learning algorithm are the basic entities of the artificial neural network. They play important

roles in calculation of the output. Some learning algorithms focus on changing the way of the connections between neurons [11], some focus on adjusting the connecting weights [12], and some focus on the above two simultaneously [13]. On the other hand, the learning of the neural network theory could be divided into three main parts: supervised learning, unsupervised learning, and reinforcement learning [14]. The objective of the algorithm is to approach the optimal value of its error function. If the accurate data exist, the supervised learning is easier to approach the optimal value of the error function than the other two, because it doesn't view a wrong result as correct. In the thesis, we focus on adjusting the connecting weight by supervised learning in system control field. Although backpropagation learning algorithm is commonly used and easy to apply, it has a problem of local optimization [15]. Hence, evolution strategies are used to increase the search space and thus try to avoid local optimization.

The thesis contains five chapters. The introduction is described in this chapter. Chapter 2 describes the basic neural network theory and some application in system control. The learning algorithms for neural network, evolution strategies, are explained in detail in Chapter 3 and the simulation results are demonstrated in Chapter 4. Finally, the conclusion is given in Chapter 5.

# Chapter 2

## Intelligent Control

Intelligent control designed to simulate intelligent biological systems is a new control method which combines automatic control and artificial intelligence. The processes of the machines using intelligent control will be similar with the human thinking. Today automatic control systems have played an important role of our daily life, such as airplanes and spacecrafts. Although automatic control has works well, it is difficult to design a controller for a complex dynamic system. To solve the problem, recently investigators have paid their attentions to the intelligent algorithms, such as fuzzy theory and neural network, to achieve the system identification and controller design. It is known that neural networks are modeled after the physical architecture of the human brain; therefore, this chapter will focus on the control using neural network, which has been widely applied to intelligent systems.

### 2.1 Introduction to Neural Network

Neural networks demonstrate the ability to learn, recall, and generalize from training patterns or data.

In general, the biological neural network in human brain is constructed by a large amount of neurons, which contain somas, axons, dendrites, and synapses. The current excited by the impulse from the other neuron will change the strength of synapses until steady when the biological neural network is learning. Artificial neural networks (ANN) which are

modeled after the physical architecture of the human brain is proposed to simulate the learning function for intelligent machine. Therefore, ANN is highly interconnected by a large of processing elements, which are also called artificial neuron or neuron simply, and its connective behavior is like human brain

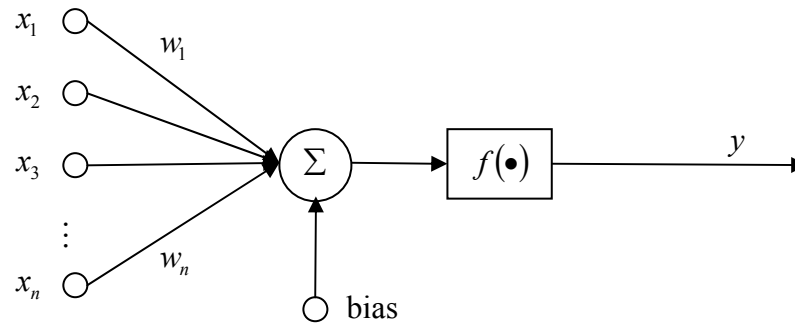


Figure 2.1 An artificial neuron

An artificial neuron, which imitates a biological neuron, is shown in Fig. 2.1 and expressed as

$$y = f\left(\sum_{k=1}^n x_k w_k + b\right) \quad (2.1)$$

where the output  $y$  is a function of the input  $x_k$ ,  $k=1,2,3,\dots$ . Note that the bias  $b$  and the weights  $w_k$ ,  $k=1,2,3,\dots$  are all constant. There are many types of activation function  $f$ , linear or nonlinear, and the hyperbolic tangent function described as

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.2)$$

is commonly used.

Because ANN is highly interconnected by a large of processing elements, the connection geometry among the processing elements is important to form a ANN. ANN can be constructed by artificial neurons in different modes, such as the commonest multilayer

feedforward network shown in Fig. 2.2, which possesses one input layer, one output layer and some hidden layers.

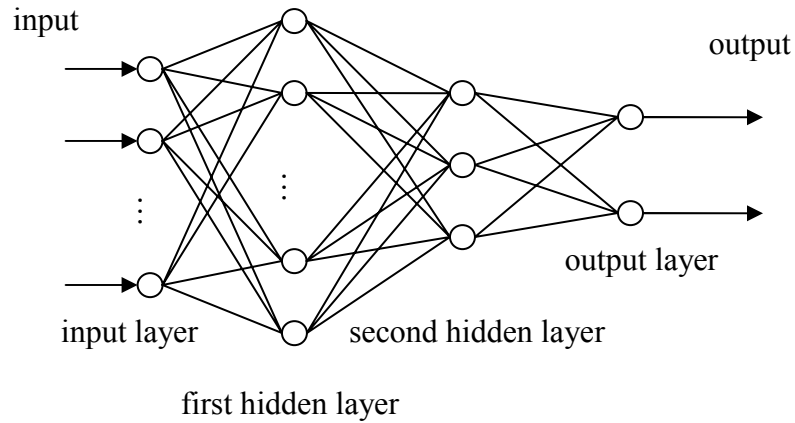


Figure 2.2 Multilayer feedforward network

The most important element of ANN is the learning rules, which are mainly classified into the parameter learning and the structure learning. The parameter learning is updating the weights in the neural network, while the structure learning is changing the network structure, such as the number of neurons and their connection. It is known that there are three types of parameter learning, including supervised learning, reinforcement learning, and unsupervised learning. In this thesis, the simulation uses a known plant and designed controller, so the input-output pairs of the controller can be gotten easily. The neural network learns the behavior of the controller by using these input-output pairs, and this kind of learning belongs to supervised learning. This thesis will focus on the supervised learning.

In supervised learning, the back propagation learning algorithm, based on the simple gradient algorithm for updating the weights, is commonly used in most applications. The back propagation learning is often executed by multilayer feedforward networks with elements containing differentiable activation functions. Such networks are also called the back-propagation networks and Fig. 2.3 shows a back-propagation network consisting of one



input layer with  $m$  neurons, one single hidden layer with  $l$  neurons, and one output layer with  $n$  neurons.

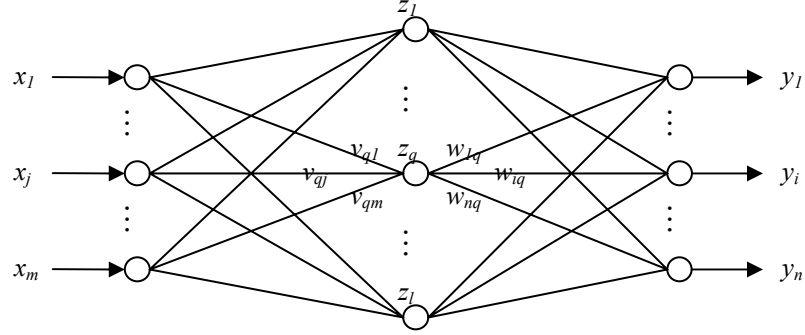


Figure 2.3 two-layer back-propagation network

For the back-propagation network in Fig. 2.3, let  $\mathbf{x} = [x_1 \ x_2 \ \cdots \ x_m]^T$ ,  $\mathbf{z} = [z_1 \ z_2 \ \cdots \ z_l]^T$ , and  $\mathbf{y} = [y_1 \ y_2 \ \cdots \ y_n]^T$  be the inputs of the network, the outputs of neurons in the hidden layer, and the outputs of the network, respectively. Let  $v_{qj}$  be the weight from the  $j$ -th neuron in the input layer to  $q$ -th neuron in the hidden layer and  $w_{iq}$  be the weight from the  $q$ -th neuron in the hidden layer to  $i$ -th neuron in the output layer. Then, the output of the  $q$ -th neuron in the hidden layer is described as

$$z_q = f_z \left( \sum_{j=1}^m v_{qj} x_j \right) \quad q = 1, 2, \dots, l \quad (2.3)$$

where  $f_z$  is the activation function, and the output of the  $i$ -th neuron in the output layer is described as

$$y_i = f_y \left( \sum_{q=1}^l w_{iq} z_q \right) \quad i = 1, 2, \dots, n \quad (2.4)$$

where  $f_y$  is the activation function.

In supervised learning with the given input-output training data  $(\mathbf{x}, \mathbf{d})$ , the cost function is defined as the following error function

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^n (d_i - y_i)^2 = \frac{1}{2} \sum_{i=1}^n \left[ d_i - f_y \left( \sum_{q=1}^l w_{iq} z_q \right) \right]^2 \quad (2.5)$$

where  $y_i$  is the output of the network and  $d_i$  is the desired output. Then, according to the gradient-descent method, the changes of the weights are determined as

$$\begin{aligned} \Delta w_{iq} &= -\eta \frac{\partial E}{\partial w_{iq}} = -\eta \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial w_{iq}} \\ &= \eta (d_i - y_i) \left[ f'_y \left( \sum_{q=1}^l w_{iq} z_q \right) \right] z_q \\ &= \eta \delta_{oi} z_q \end{aligned} \quad (2.6)$$

and

$$\begin{aligned} \Delta v_{qj} &= -\eta \frac{\partial E}{\partial v_{qj}} = -\eta \sum_{i=1}^n \left( \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial z_q} \frac{\partial z_q}{\partial v_{qj}} \right) \\ &= \eta \sum_{i=1}^n \left[ (d_i - y_i) f'_y \left( \sum_{q=1}^l w_{iq} z_q \right) w_{iq} \right] f'_z \left( \sum_{j=1}^m v_{qj} x_j \right) x_j \\ &= \eta \delta_{hq} x_j \end{aligned} \quad (2.7)$$

where  $\delta_{oi} = (d_i - y_i) \left[ f'_y \left( \sum_{q=1}^l w_{iq} z_q \right) \right]$  and  $\delta_{hq} = f'_z \left( \sum_{j=1}^m v_{qj} x_j \right) \sum_{i=1}^n (\delta_{oi} w_{iq})$ . Besides, the learning rate  $\eta$  is often given experimentally to reduce the computing time or increase the precision. Finally, the weights can be updated by

$$\begin{cases} \mathbf{w}(n+1) = \mathbf{w}(n) + \Delta \mathbf{w} \\ \mathbf{v}(n+1) = \mathbf{v}(n) + \Delta \mathbf{v} \end{cases} \quad (2.8)$$

It is the advantage that the back propagation algorithm can be used in the networks with nonlinear functions. Many researchers use the simple algorithm to learn classification and

input-output relationship. The objective of this thesis is to control the system using neural network, so the neural network for control will be introduced in the next section.

## 2.2 Neural Network for Control

Since traditional control theory is based on the mathematic model of the plant, it fails when the mathematic model is unknown or not accurate. The intelligent control theory based on the abilities of thinking and learning of human is not restricted to the mathematic model. It has more abilities to solve the control problem than the conventional control theory.

The investigators have proposed many control method based on neural network, such as model reference control. Besides, some neural network structures are proposed for system identification, such as recurrent neural network. The model reference control based on neural network and recurrent neural network will be introduced in this section.

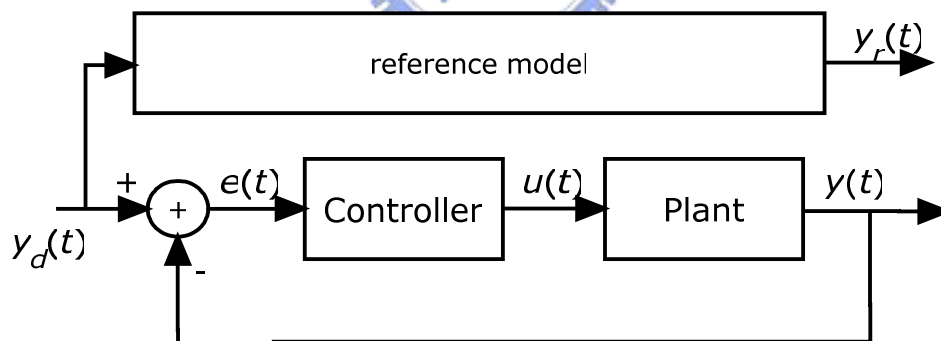


Figure 2.4 the feedback system

The model reference control based on neural network is introduced first. When the plant is given, the feedback system is shown as Fig. 2.4 and the controller is the learning objective system [16]. The reference model is designed according to the specification of the problem. Basically, the learning algorithm is based on the gradient method, so the learning process needs two neural networks, one to be a controller and the other to be a plant. This learning can

use backpropagation learning algorithm introduced in the last section. To back propagate the error to the neural network, the inverse of the plant should be known. Unfortunately, not every inverse dynamic of the plant exists, or can be known even if it exists. Hence, the neural network plant should be trained first for back propagating the system error to the neural network controller, shown as Fig 2.5. However, it causes that the neural network plant should be retrained whenever the condition of the plant changed a little.

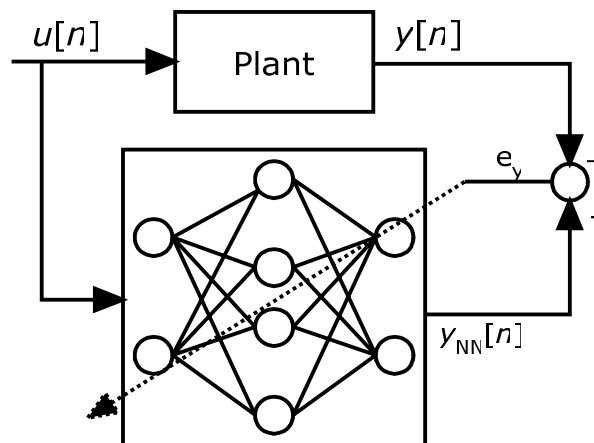


Figure 2.5 to train neural network plant

It is known that the learning algorithm alters the connecting weights depending on the error between the plant output and the neural network plant output. If the neural network plant is trained well, it will replace the original plant. It is worthy noticing that the connecting weights of the neural network plant do not change when the neural network controller is trained. Fig. 2.6 shows the learning system for training the neural network controller. Compared with the reference model, the neural network controller does not learn as the real controller. If the plant in  $s$ -domain is defined as  $H(s)$  and the controller is defined as  $G(s)$ , the system is described as

$$\frac{Y(s)}{U(s)} = \frac{G(s)H(s)}{1+G(s)H(s)} \quad (2.9)$$

where  $Y(s)$  is the system output and  $U(s)$  is the system input. However, the neural network

system is described as

$$\frac{Y_{NN}(s)}{U(s)} = G_{NN}(s)H_{NN}(s) \quad (2.10)$$

where  $G_{NN}(s)$  denotes the neural network controller and  $H_{NN}(s)$  denotes the neural network plant. Because the neural network plant is trained as the real plant,  $H_{NN}(s)$  is assumed identical to  $H(s)$ . Thus, the neural network controller does not learn as the real controller, and expressed as

$$G_{NN}(s) = \frac{G(s)}{1 + G(s)H(s)} \quad (2.11)$$

which does not equal to  $G(s)$ .

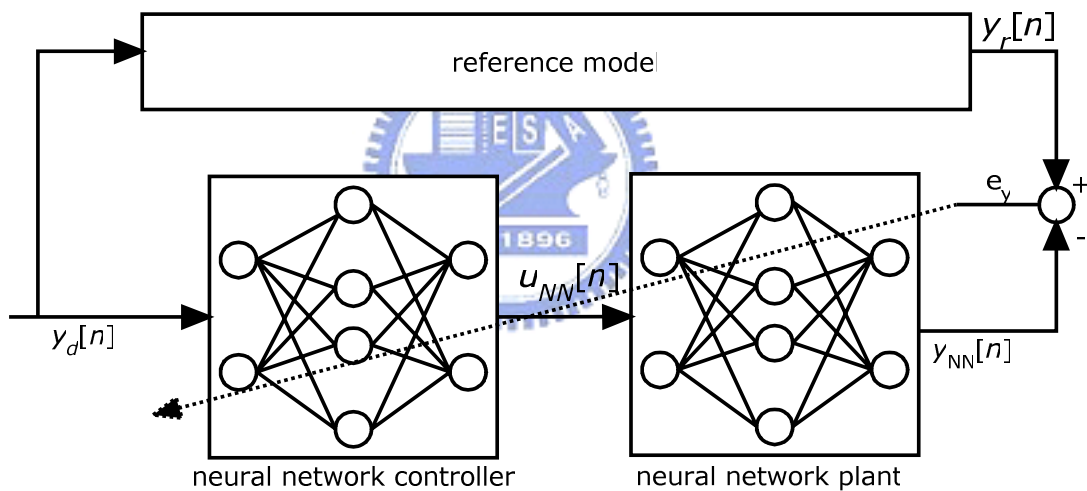


Figure 2.6 to train the neural network controller

The model reference control based on neural network is a simple way to control the system depending on the specification of the problem. However, the neural network plant should be trained first, and that leads errors while the plant changes, such as the input-output relationship and the sampling time. No matter what the plant changes, the neural network plant should be trained again for back propagation precisely. Besides, it is known that the neural

network controller is not the original controller. These restrict the use of the model reference control based on the neural network.

Because of the restriction of the inverse dynamic, some investigators proposed other control method without learning the neural network plant [17-19]. For simplicity, the neural network controller could be learned as system identification while the controller is designed, but it can not be guarantee the control ability of the neural network controller. Thus, a fully recurrent neural network, shown as Fig 2.7, is proposed with capability of dynamics and the ability to store the information for later use. In the next paragraph, a leaning algorithm for fully recurrent neural network, real- time recurrent learning (RTRL), will be introduced.

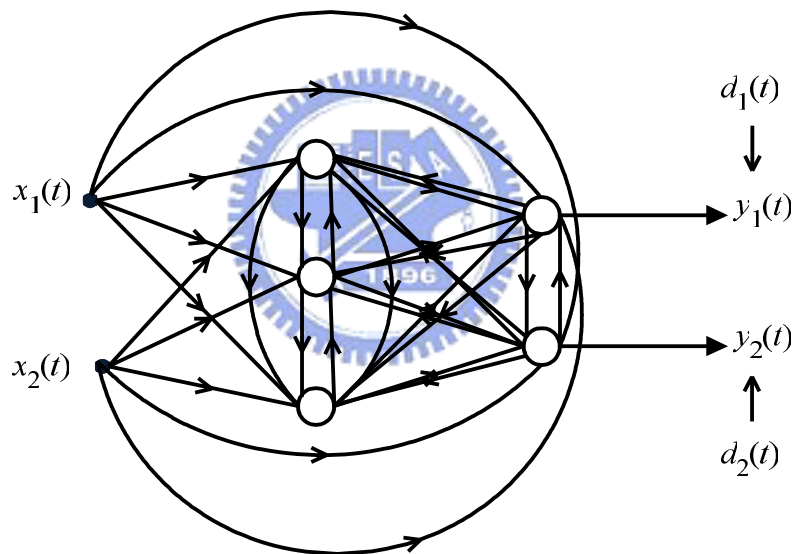


Figure 2.7 the fully recurrent neural network

RTRL has been proposed by many investigators independently, and the most used is by Williams and Zipser [20], which is introduced here. The learning algorithms will be described in detail as below.

While the network have  $n$  neurons, with  $m$  external inputs, the outputs of neurons are denoted as  $y_{kn}(t)$ ,  $kn = 1,2,\dots,n$ , and the external inputs are denoted as  $x_{ki}(t)$ ,  $ki = 1,2,\dots,m$ ,

where  $t$  is the time. These values are concatenated to form the  $(m+n)$ -tuple  $\mathbf{z}(t)$ , which is expressed as

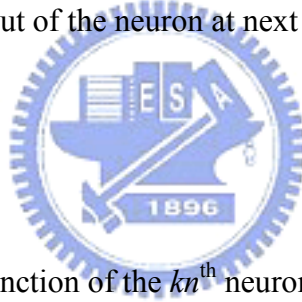
$$z_k(t) = \begin{cases} y_{kn}(t) & \text{if } k = kn \\ x_{ki}(t) & \text{if } k = n + ki \end{cases} \quad (2.12)$$

which are the outputs of every neurons and input neurons. Since the network is fully connected, the weight matrix  $\mathbf{W}$  in a  $n \times (m+n)$  matrix to obtain  $n$  outputs of neurons by  $m$  external inputs and  $n$  inputs of neurons. Thus, the net input of the  $kn^{\text{th}}$  neurons at time  $t$  can be calculated as

$$s_{kn}(t) = \sum_l w_{knl} z_l(t) \quad (2.13)$$

where  $l = 1, 2, \dots, m+n$ . The output of the neuron at next time step is determined as

$$y_{kn}(t+1) = f_{kn}(s_{kn}(t)) \quad (2.14)$$



where  $f_{kn}(\cdot)$  is the activation function of the  $kn^{\text{th}}$  neuron. Let the first  $o$  neurons exist specified target value  $\mathbf{d}$ , so the  $n$ -tuple error  $\mathbf{e}$  is described as

$$e_{kn}(t) = \begin{cases} d_{kn}(t) - y_{kn}(t) & \text{if } kn \leq o \\ 0 & \text{otherwise} \end{cases} \quad (2.15)$$

Define the overall network error at time  $t$  as

$$J(t) = \frac{1}{2} \sum_{kn} [e_{kn}(t)]^2, \quad (2.16)$$

thus, let

$$J_{\text{total}}(t_0, t_1) = \sum_{t=t_0+1}^{t_1} J(t) \quad (2.17)$$

denotes the network error running from time  $t_0$  to the time  $t_1$ .  $\mathbf{W}$  is adjusted along the negative of  $\nabla \mathbf{W} J_{\text{total}}(t_0, t_1)$ . Thus, the weight change can be written as

$$\Delta w_{ij} = \sum_{t=t_0+1}^{t_1} \Delta w_{ij}(t) \quad (2.18)$$

where

$$\Delta w_{ij}(t) = -\alpha \frac{\partial J(t)}{\partial w_{ij}} = \alpha \sum_{kn} e_{kn}(t) \frac{\partial y_{kn}(t)}{\partial w_{ij}} \quad (2.19)$$

and  $\alpha$  is the fixed learning rate. The value of  $\frac{\partial y_{kn}(t)}{\partial w_{ij}}$  can be determined as

$$\frac{\partial y_{kn}(t+1)}{\partial w_{ij}} = f'_{kn}(s_{kn}(t)) \left[ \sum_{l=1}^n w_{knl} \frac{\partial y_l(t)}{\partial w_{ij}} + \delta_{ikn} z_j(t) \right] \quad (2.20)$$

where  $\delta_{ik}$  denotes the Kronecher delta. It is known that the initial states of the network is independent of the weights, so it can be given as

$$\frac{\partial y_{kn}(t_0)}{\partial w_{ij}} = 0. \quad (2.21)$$

One variable  $p_{ij}^{kn}(t)$ , defined as

$$p_{ij}^{kn}(t+1) = f'_{kn}(s_{kn}(t)) \left[ \sum_{l=1}^n w_{knl} p_{ij}^l(t) + \delta_{ikn} z_j(t) \right] \quad (2.22)$$

where

$$p_{ij}^{kn}(t_0) = 0 \quad (2.23)$$

is created to denote  $\frac{\partial y_{kn}(t)}{\partial w_{ij}}$ . Finally, the weight change at time  $t$  can be determined by

$$\Delta w_{ij}(t) = \alpha \sum_{kn} e_{kn}(t) p_{ij}^{kn}(t+1) \quad (2.24)$$



and the overall weight change can be also determined.

The RTRL is broadly used such as in classification and learning finite state systems, and the author show many simulation results to demonstrate its ability. Unfortunately, the calculation of this learning algorithm needs many previous data, so the algorithm needs a lot of memory to store information. Besides, it is computational expensive, because the weight matrix is large and the previous state information is much.

There are still many other methods to learn dynamic systems or controllers, but they will not be explained in detail in the thesis. In the next section, evolution strategies for neural network will be introduced using simple neural network structures to learn first order LTI systems.



# Chapter 3

## Evolution Strategies

Recently, the concept of the biological evolution is used in the intelligent theory, such as genetic algorithms, to reproduce the species generation by generation, then learn and survive based on the nature selection. Several investigators have proposed many evolution strategies to solve problems in diverse fields.

### 3.1 Modeling of First Order LTI System

In general, a linear time invariant system is described by an ordinary differential equation, expressed as

$$y^{(n)}(t) + a_1 y^{(n-1)}(t) + a_2 y^{(n-2)}(t) + \dots + a_{n-1} y(t) = bu(t) \quad (3.1)$$

where  $y(t)$  is the system output at time  $t$ ,  $u(t)$  is the system input at time  $t$ , and  $a_i$  are the constant parameters of the system,  $i=1,2,3,\dots,n-1$ . When the initial conditions,  $y^{(n)}(0)$ ,  $y^{(n-1)}(0)$ , ...,  $\dot{y}(0)$ , and  $y(0)$ , are all zero, the LTI system could be rewritten into the transfer function as

$$\frac{Y(s)}{U(s)} = \frac{b}{s^n + a_1 s^{n-1} + \dots + a_{n-2} s + a_{n-1}} \quad (3.2)$$

which is commonly used to clarify the characteristics of the LTI system. This thesis will focus

on the modeling of the simplest first order LTI system, expressed as

$$\dot{y}(t) + ay(t) = bu(t), \quad (3.3)$$

by intelligent structures and algorithms. For simplicity, the first LTI system is commonly rewritten into the transfer function as

$$\frac{Y(s)}{U(s)} = \frac{b}{s+a} \quad (3.4)$$

which has been widely used in controller design.

The thesis focus on implementing the first order LTI systems using neural networks, but it is known that the neural networks belong to discrete time systems, not continuous time system. Therefore, the error does exist between the NN system and the objective system, the first order LTI systems. A simple method of DT system has been proposed to approximate the first order LTI system [21], and will be introduced next.

The discrete-time system obtained from (3.3) under the sampling time  $\Delta T$  is described as

$$y[n+1] = (1 - a \Delta T)y[n] + (b \Delta T)u[n] \quad (3.5)$$

where  $y[n] = y(n\Delta T)$  and  $u[n] = u(n\Delta T)$  and which is so called the first order difference equation.

To find the error between the first order LTI system (3.3) and its corresponding difference equation (3.5) when the system input is a step function, the solution of (3.3) should be determined first as

$$y(t) = c + de^{-at} \quad \text{and} \quad \dot{y}(t) = -ade^{-at} \quad (3.6)$$

where  $c$  and  $d$  are related to the input amplitude and the system initial conditions  $y(0)$  and  $\dot{y}(0)$ . If the step input is given with amplitude  $A$  and the system is initially idled, the solution

is obtained as

$$y(t) = \frac{bA}{a} - \frac{bA}{a} e^{-at}. \quad (3.7)$$

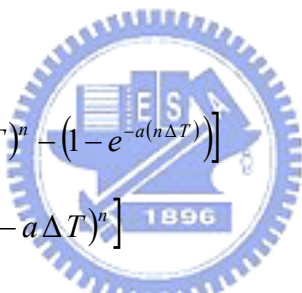
Thus, the exact solution at  $t = n\Delta T$  is expressed as

$$y(n\Delta T) = \frac{bA}{a} - \frac{bA}{a} e^{-a(n\Delta T)} = \frac{bA}{a} [1 - e^{-a(n\Delta T)}]. \quad (3.8)$$

As for the solution of (3.5) with the same input and initial conditions of (3.3), its solution corresponding sampling time  $\Delta T$  can be found as

$$y[n] = \frac{bA}{a} [1 - (1 - a\Delta T)^n]. \quad (3.9)$$

Compared to (3.8), the error at time  $t = n\Delta T$  is

$$\begin{aligned} \Delta y[n] &= y[n] - y(n\Delta T) \\ &= \frac{bA}{a} [1 - (1 - a\Delta T)^n - (1 - e^{-a(n\Delta T)})] \\ &= \frac{bA}{a} [e^{-a(n\Delta T)} - (1 - a\Delta T)^n] \end{aligned} \quad (3.10)$$


and the sum of error is

$$\begin{aligned} \sum_{k=0}^n \Delta y[k] &= \frac{bA}{a} \left[ \frac{1 - e^{-a(n+1)\Delta T}}{1 - e^{-a\Delta T}} - \frac{1 - (1 - a\Delta T)^{n+1}}{1 - (1 - a\Delta T)} \right] \\ &= \frac{bA}{a} (1 + e^{-a\Delta T} + e^{-2a\Delta T} + \dots + e^{-an\Delta T}) \\ &\quad - \frac{bA}{a} (1 + (1 - a\Delta T) + (1 - a\Delta T)^2 + \dots + (1 - a\Delta T)^n) \\ &= \frac{bA}{a} \left\{ \frac{(-a\Delta T)^2}{2} + \left[ (1 - a\Delta T)(-a\Delta T)^2 + \left( \frac{(-a\Delta T)^2}{2} \right)^2 \right] + \dots \right\} \\ &= (\Delta T)^2 \times \frac{bA}{a} \left\{ \frac{(-a)^2}{2} + \left[ (1 - a\Delta T)(-a)^2 + \frac{(-a)^2}{2} \frac{(-a\Delta T)^2}{2} \right] + \dots \right\} \end{aligned} \quad (3.11)$$

where  $\Delta T$  is assumed to be small and the sum of error is then proportional to  $\Delta T^2$ . It implies that the sum of error is reduced while the sampling time decreases.

A close look at (3.11) will reveal that the fitness is large while the sampling time is small. The problem is how to increase the fitness except decreasing the sampling time. Here, neural network theory is introduced to learn the first order LTI system for solving the problem. Next, two neural network structures will be introduced in order to learn the first order LTI system.

## 3.2 Neural Network Structure

In the thesis, the learning result after evolution strategies is determined from the fitness function which is defined as the negative sum of the errors between the outputs of the first order LTI system and the NN system. This thesis is intended as an investigation of whether a neural network structure could learn the first order LTI system as a controller. In the last section, the first order difference equation is proposed as a simple way to approach the first order LTI system. Before introducing our neural network structures, the structure for first order difference equation will be introduced first. It shows that the first order LTI system could be implementing based on neural networks.

### 3.2.1 Structure for The First Order Difference Equation

It can be found in (3.5) that the first order difference equation needs two parameters,  $(1-a\Delta T)$  and  $(b\Delta T)$ , for the input and the output at last time step to approach the first order LTI system. In neural network, a general neuron with  $n$  inputs contains  $n$  connecting weights,  $(n-1)$  operators, and one activation function to produce an output. Here, the system input and output at last time are both viewed as inputs of a neural network. Now that the first order difference equation could be implemented by just one neuron with the activation function whose slope is one, shown as Fig. 3.1. Thus, it is concerned that whether the neural network with more

neurons is possible to let fitness larger than first order difference equation. Intuitively, it is possible to do that. It will be shown in Chapter 4.

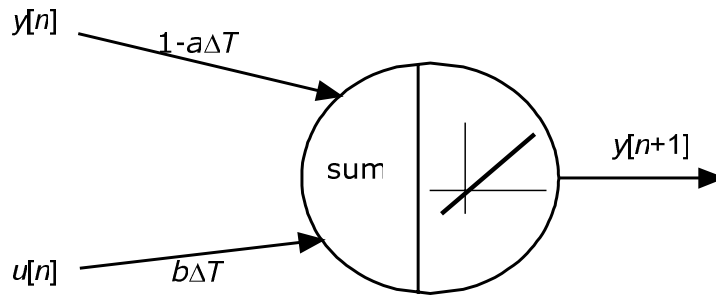


Figure 3.1 the first order difference equation using one neuron

### 3.2.2 General Structure

According to the first order difference equation, a general structure of neural network contains an input layer with two neurons which represent the system input and output at last time, an output layer with one neuron which represents the system output at this time, and some hidden layers. It is known that MLP can process more problems than single layer, so one hidden layer is used in the general structure for simplicity. Although amount of the neurons in the hidden layer will increase the possibility of good performance, they will increase the computation time. To give consideration to the possibility of good performance and the computation time, five neurons is chosen to put in the hidden layer, shown as Fig. 3.2. One may notice that the activation functions of this structure are all described as

$$f(\cdot)=1, \tag{3.12}$$

and there is no threshold term in the structure. In this two-layered neural network structure, called ‘GS’ for short, the synaptic weight connecting the neuron  $i$  to the neuron  $j$  is symbolized as  $w_{ji}^{(l)}$  where  $l$  means that the synaptic weight is between  $(l-1)^{\text{th}}$  layer and  $l^{\text{th}}$

layer,  $l = 1, 2$ . Namely, there are two weight matrices, which are described as

$$\mathbf{W}^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \\ w_{31}^{(1)} & w_{32}^{(1)} \\ w_{41}^{(1)} & w_{42}^{(1)} \\ w_{51}^{(1)} & w_{52}^{(1)} \end{bmatrix} \quad \text{and} \quad \mathbf{W}^{(2)} = [w_{11}^{(2)} \quad w_{12}^{(2)} \quad w_{13}^{(2)} \quad w_{14}^{(2)} \quad w_{15}^{(2)}] \quad (3.13)$$

in the GS, and the output of the GS can be simply determined as

$$y[n+1] = \mathbf{W}^{(2)} \mathbf{W}^{(1)} \begin{bmatrix} y[n] \\ u[n] \end{bmatrix} \quad (3.14)$$

that is a discrete time equation.

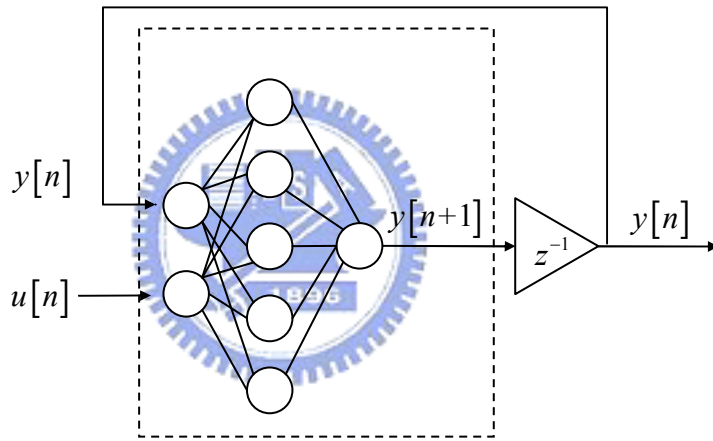


Figure 3.2 the GS for first order LTI system

Compare with the first order difference equation in the last section, the output of the GS will be the same if the weight matrices are given as

$$\mathbf{W}^{(1)} = \begin{bmatrix} 1 - a\Delta T & b\Delta T \\ 1 - a\Delta T & b\Delta T \\ 1 - a\Delta T & b\Delta T \\ 1 - a\Delta T & b\Delta T \\ 1 - a\Delta T & b\Delta T \end{bmatrix} \quad \text{and} \quad \mathbf{W}^{(2)} = [0.2 \quad 0.2 \quad 0.2 \quad 0.2 \quad 0.2] \quad (3.15)$$

where  $a$  and  $b$  are the parameters of first order LTI system  $\frac{b}{s+a}$  and  $\Delta T$  means the sampling time. As long as the GS with weight matrices described as (3.15) will be equivalent to the first

order difference equation, the fitness will be the same.

Whether any other weight matrices will lead to larger fitness is a question. It can be said that the GS is better than the first order difference equation if fitness of the GS is larger with the same sampling time. There remains a second question: how much does the sampling time affect the fitness of the GS. It is known that the fitness will increase as the sampling time decrease in the first order difference equation, so to reduce the sampling time is necessary. Thus, the question about whether the sampling time has the same influence on the GS with first order difference equation is taken up in the next chapter.

### 3.2.3 Structure with Sampling Time

Since the GS is a discrete time system without coefficients related to sampling time, it learns under the fixed sampling time of the training data. As a result, the GS is only suitable for the fixed sampling time, which restricts the use of the GS. One problem is raised: Does any neural network structure exist for arbitrary sampling times after learning, just like the first order difference equation suitable for a first order LTI system? To solve the problem, a structure with sampling time, called ‘SST’ in short, is designed to adapt a wide range of sampling times, as shown in Fig. 3.3, which adopts the parameters of the first order difference equation and is described as

$$y[n+1] = \mathbf{W}^{(2)} \mathbf{W}^{(1)} \begin{bmatrix} (1 - a\Delta T)y[n] \\ (b\Delta T)u[n] \end{bmatrix} \quad (3.16)$$

where the gains of the inputs is the main difference with the GS. By setting the weight matrices as

$$\mathbf{W}^{(1)} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \text{ and } \mathbf{W}^{(2)} = [0.2 \quad 0.2 \quad 0.2 \quad 0.2 \quad 0.2], \quad (3.17)$$



the SST obtains the output same as the first order difference equation. In this case, the SST is available for a wide range of sampling times and subject to an error due to the use of the sampling time. To further increase the fitness, it is required to find different weight matrices based on the evolution strategies, which will be introduced in the next section.

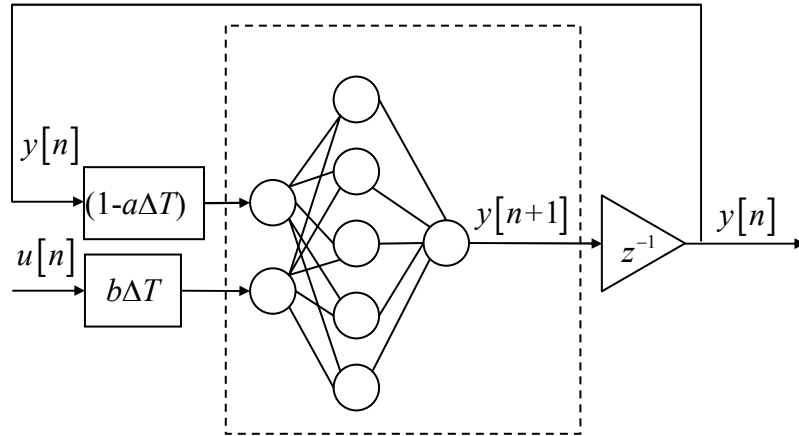


Figure 3.3 the SST for first order LTI system

In this section, except the structure for the first order difference equation, two types of the structures of the neural network are designed to learn the first order LTI system, and the most difference between these two types is how the sampling time effects. Before simulations, the learning algorithm of these structures will be introduced in the next section.

### 3.3 Evolution Strategies

In neural network theory, it is an important issue to find the neural network whose weights lead to the global minimum of an error function. Unfortunately, it is difficult to know the minimum is global or not, even for systems without uncertainties. Therefore, instead of global minimum, investigators often develop evolution strategies to search the optimal minimum of fitness function with largest fitness locally, not globally.

The backpropagation learning algorithm adjusts the synaptic weights using chain rule

depending on the gradient descent method as (2.6) and (2.7), so the synaptic weights could not be updated using the backpropagation learning algorithm while the inverse dynamic system is unknown. However, the evolution strategies we proposed adjust weights using evolution, but the chain rule according to the fitness function, so it avoid the disadvantage of the backpropagation learning algorithm. It can be said that that the neural network can learn easily using the evolution strategies even if the inverse dynamic system is unknown.

According to the origin of species by Darwin [22], individuals less suited to the environment are less likely to survive and to reproduce. Under the limit of the environment, much of the species reproduce sexually, which leads no two individuals are identical generally, and thus the individuals more suited to the environment are more likely to keep their inheritable characteristics to future generations. That is so called nature selection, the most widely used by biologists to represent the scientific model of how species evolve.

Here, evolution strategies, depending on nature selection, are proposed for the learning algorithm of the neural network. In the evolution strategies, the given problem is viewed as the environment and every set of weight matrices is viewed as an individual [23]. Basically, the biological reproduction is divided into two groups: sexual and asexual. Individual is different with their parents by sexual reproduction, but it is just identical copy of its parents by asexual reproduction except for mutation. Different with the nature world where the mutation happens with the reproduction unpredictably, it does not happen in the evolution strategies. In general, the species do not reproduce sexually and asexually at the same time, even if hydras and earthworms which can reproduce either sexually or asexually. In the evolution strategies, the reproduction happen both sexually and asexually at the same time.

Four points is helpful in sketch out the evolution strategies: the initial individuals creation, the reproduction process, the learning process, and the elite process. Since the generation inherits from the last generation, the initial individuals affect the future offspring;

it means that not arbitrary initial individual after learning will behave as the objective system. By various reproduction processes and the learning processes, the individuals of every generation will be different even if the initial individuals are the same. Further, the elite process chooses the individuals which are more suitable to the problem. In the thesis, the evolution leads to a lot of results depending on its initial population, reproduction process, learning process, and elite process, and any above terms probably fails to learn. These four influences will be discussed next.

### 3.3.1 Initial Individuals Creation

From (3.13), the structures both contain two weight matrices  $W^{(1)}$  and  $W^{(2)}$ . In the evolution strategies, the matrices are combined as an individual  $W_k^g$ , defined as

$$W_k^g = \left[ w_{11}^{(1)} \quad w_{12}^{(1)} \quad w_{21}^{(1)} \quad w_{22}^{(1)} \quad w_{31}^{(1)} \quad w_{32}^{(1)} \quad w_{41}^{(1)} \quad w_{42}^{(1)} \quad w_{51}^{(1)} \quad w_{52}^{(1)} \quad w_{11}^{(2)} \quad w_{12}^{(2)} \quad w_{13}^{(2)} \quad w_{14}^{(2)} \quad w_{15}^{(2)} \right]$$

(3.18)

which is the  $k^{\text{th}}$  individual of the  $g^{\text{th}}$  generation. Let  $g=0, 1, 2, \dots$ , and  $k=1, 2, 3, \dots, n^g$  where  $n^g$  is the population size of the  $g^{\text{th}}$  generation. First, it is known that the initial individuals  $W_k^0$  affect the offspring  $W_k^g$  where  $g \neq 0$  and not arbitrary  $W_k^0$  lead a successful learning. No exact way can decide how these initial individuals are before the evolution. The learning is expected to success even if the initial individuals are given randomly. Unfortunately, it is difficult because of the issue of the local optimal. Note that (3.15) and (3.17) could be thought as good individual of GS and SST, respectively. If the parameters are used to be an initial individual, the probability of success increases. Chapter 4 will show the influences of the initial individuals.

### 3.3.2 Reproduction Process

The reproduction process is used for increasing the searching space of the individuals. For the human beings, the offspring combines the half chromosomes of each parent when the sexual reproduction happens. The child contains the half genes from the mother and the other half genes from the father. Not all the reproduction processes of the living things are same to the human begin, such as hydras whose offspring can be produced asexually. Therefore, how to create the offspring is concerned.

In the evolution strategies, the individuals  $W_k^g$  are called the parents and  $\Omega_k^g$  are called the children after the reproduction process. In the beginning of the reproduction process,  $n^0$  initial individuals have been created using method of Section 3.3.1. Of course, there are many approaches to create children and they can be produced from not only two parents. In the thesis, the reproduction process is divided into two methods: inward method and outward method. The inward method satisfied by

$$|\Omega_k^g - W_c^g| \leq \max_k |W_k^g - W_c^g| \quad (3.19)$$

is introduced first where  $W_c^g$  expressed as

$$W_c^g = \frac{1}{n^g} (W_1^g + W_2^g + \dots + W_{n^g}^g) \quad (3.20)$$

is the center of the parents . For example, a child can be created as

$$\Omega_k^g = \begin{cases} \frac{1}{2} (W_{n^g}^g + W_1^g), & \text{if } k = n^g \\ \frac{1}{2} (W_k^g + W_{k+1}^g), & \text{otherwise} \end{cases}, \quad (3.21)$$

shown as Fig. 3.4(a), or be created as

$$\Omega_k^g = \frac{1}{n^g} (W_1^g + W_2^g + \dots + W_{n^g}^g), \quad (3.22)$$

shown as Fig. 3.4(b). Besides, the outward method could be used. The children can be created outward as

$$\begin{cases} \Omega_k^g - W_{n^g}^g = l(W_1^g - W_{n^g}^g), & \text{if } k = n^g \\ \Omega_k^g - W_k^g = l(W_{k+1}^g - W_k^g), & \text{otherwise} \end{cases} \quad (3.23)$$

$$\Rightarrow \Omega_k^g = \begin{cases} l(W_1^g - W_{n^g}^g) + W_{n^g}^g, & \text{if } k = n^g \\ l(W_{k+1}^g - W_k^g) + W_k^g, & \text{otherwise} \end{cases}$$

shown in Fig. 3.5(a), or

$$\begin{aligned} \Omega_k^g - W_c^g &= l(W_k^g - W_c^g) \\ \Rightarrow \Omega_k^g &= l(W_k^g - W_c^g) + W_c^g \end{aligned} \quad (3.24)$$

where  $l$  is larger than one, shown in Fig. 3.5(b). In (3.23) and (3.24), the multiple of the distance  $l$  affects the children, thus it affects the probability of finding an individual which contains the best optimal value, called the best individual. However, it is unknown what the best individual is, and the effect degree could not be predicted because the evolution strategies not only use the reproduction process. Since the effect of  $l$  is unknown, a variable  $l$  seems a better choice than the fixed.

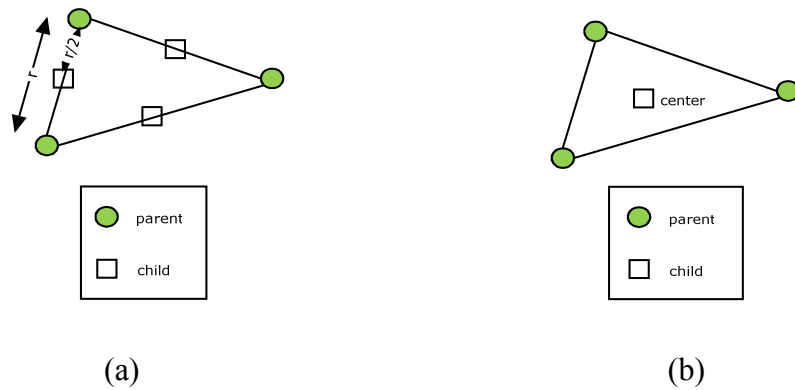


Figure 3.4 Illustration of finding children inward

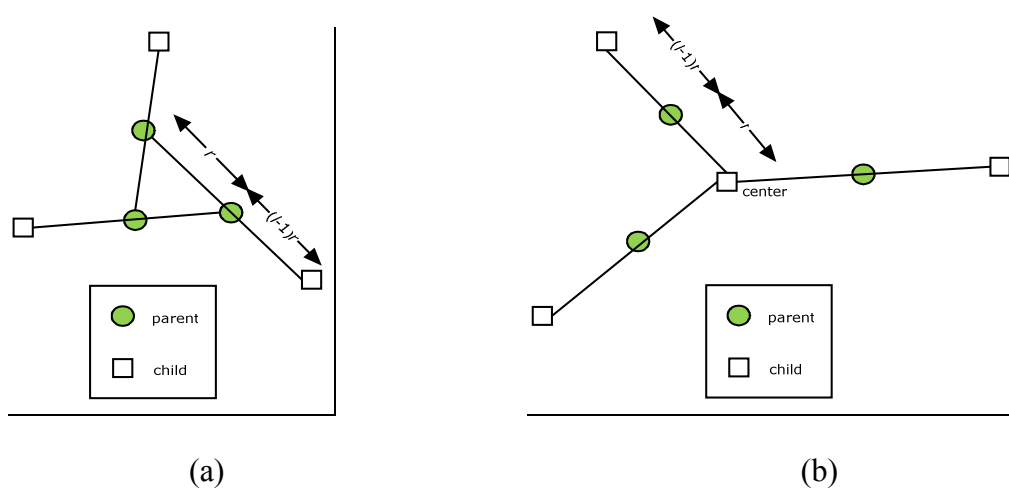


Figure 3.5 Illustration of finding children outward

It is mentioned that the variable multiple  $l$  can increase the probability of finding the best individual, and be given 0.95 of  $l$  in the last generation while the variation of fitness is not more than 5 generations in the evolution strategies. However, when  $l$  decreases to smaller than 1, the method is inward, not outward. In this case, a random reproduction process is adopted to replace the original outward process described as (3.24). The new children are reproduced randomly with larger distance than the maximum distance  $\max_k |W_k^g - W_c^g|$ , shown as Fig. 3.6. The purpose of this replacement is to increase the possibility of finding the best individual.

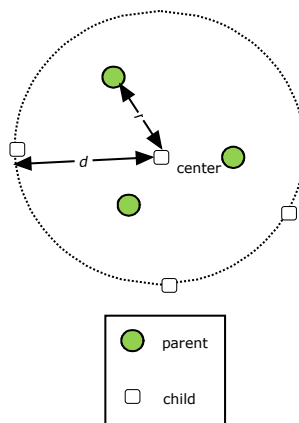


Figure 3.6 Random children creation

### 3.3.3 Learning Process

After the reproduction process, there are  $(2n+1)$  children when there are  $n$  parents. In the learning process, every child learns independently, and the learning process will be introduced in this section.

The objective of the learning process is to find an individual whose fitness approaches the optimal value, called optimal individual. Traditionally, the backpropagation learning algorithm uses the gradient of the error function to reach the objective. In the evolution strategies, it doesn't use the gradient method to decide how the synaptic weights alter. Every step of the learning process tries to find a new individual whose fitness is larger than last step until the process reaches one of the stop conditions.

The learning process starts with randomly creating a small enough difference  $\delta \Omega_{\text{tmp}}(1)$  around the first individual  $\Omega_k^g(1)$  and then obtain the temporal individual

$$\Omega_{\text{tmp}}(2) = \Omega_k^g(1) + \delta \Omega_{\text{tmp}}(1). \quad (3.25)$$

If  $\Omega_{\text{tmp}}(2)$  does not lead to a fitness larger than  $\Omega_k^g(1)$ , give it up and further find a new temporal individual. Once the fitness created by  $\Omega_{\text{tmp}}(2)$  is larger than  $\Omega_k^g(1)$ , choose  $\delta \Omega_{\text{tmp}}(1)$  and  $\Omega_{\text{tmp}}(2)$  as the desired difference and individual, that is,  $\delta \Omega(1) = \delta \Omega_{\text{tmp}}(1)$ , and  $\Omega_k^g(2) = \Omega_{\text{tmp}}(2)$ .

After the difference  $\delta \Omega(1)$  is determined, the desired individual is updated as the procedure depicted in Fig. 3.7 and described as

$$\Omega_{\text{tmp}}(s+1) = \Omega_k^g(s) + \delta \Omega_{\text{tmp}}(s) \quad (3.26)$$

where

$$\begin{aligned}\delta\Omega_{\text{tmp}}(s) &= a\delta\Omega(s-1) + b\delta\Omega^\perp(s-1) \\ b &= \sqrt{1-a^2}\end{aligned}\tag{3.27}$$

and  $\delta\Omega^\perp(s-1)$  is the perpendicular vector of  $\delta\Omega(s)$  and  $a$  is chosen randomly between 0 and 1 to avoid the direction of  $\delta\Omega_{\text{tmp}}(s)$  is opposite to  $\delta\Omega(s-1)$ .

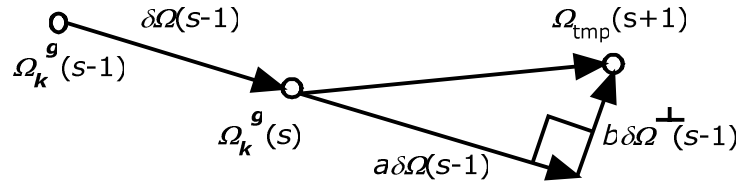


Figure 3.7 finding temporal individual of the next step

To find a perpendicular vectors, for example, one method is to choose two entries indexed  $i$  and  $j$  in the original vector, which is described as

$$\mathbf{v} = [v_1 \ v_2 \ \dots \ v_i \ \dots \ v_j \ \dots \ v_n]\tag{3.28}$$

and obtain the perpendicular vector as

$$\mathbf{v}^\perp = [0 \ 0 \ \dots \ v_j \ \dots \ -v_i \ \dots \ 0].\tag{3.29}$$

Although there are other simple ways to find perpendicular vectors, a more complex way which contains more variety is used in the thesis. The origin vector is described as (3.28), and every entry of the perpendicular vector is given randomly first except the index  $i$ , determined as

$$\mathbf{v}^\perp = [r_1 \ r_2 \ \dots \ r_{i-1} \ c \ r_{i+1} \ \dots \ r_n]\tag{3.30}$$



where  $c$  is given as

$$c = -\frac{v_1 r_1 + v_2 r_2 + \dots + v_{i-1} r_{i-1} + v_{i+1} r_{i+1} + \dots + v_n r_n}{v_i} \quad (3.31)$$

such that  $\mathbf{v} \cdot \mathbf{v}^\perp = 0$ . Since the perpendicular vector is decided,  $\delta \Omega_{\text{tmp}}(s)$  could be calculated. The temporal individual of  $(s+1)^{\text{th}}$  step is determined as (3.26). If  $\Omega_{\text{tmp}}(s+1)$  leads larger fitness than  $\Omega_k^g(s)$ , keep  $\delta \Omega(s)$  as  $\delta \Omega_{\text{tmp}}(s)$  and let  $\Omega_k^g(s+1)$  be equal to  $\Omega_{\text{tmp}}(s+1)$ , otherwise, choose  $a$  randomly and find a new perpendicular vector to create a new  $\delta \Omega_{\text{tmp}}(s)$  again. Then, use the same way to get individual of every step until the learning process reaches the stop conditions. If the fitness of the final step does not reach the desired fitness, suppose it as the optimal fitness and labeled as  $\Omega_{k,\text{opt}}^g$ . Every child uses the learning process to find its own optimal fitness.

The most common used stop condition is to check whether the fitness reach the desired fitness or not. Besides, the number of the steps could be restricted by experience or the learning could be stopped when the fitness has unchanged for several steps. In the thesis, the stop conditions of the learning process are checking whether the fitness is smaller than the desired and the weights have unchanged for several steps. If the learning process reaches any of the stop conditions, it stops.

### 3.3.4 Elite Process

It is known that the environment limits the population size of the species, so the number of individuals is restricted no matter the individuals belongs to which generation. Since the meaning of this restriction is to control the population size within limits, the generation length,

the population size of every generation, or the population size of all generations could be chosen as a restriction. For simplicity, the generation length and the population size of every generation are set to be the restriction. Unfortunately, the restriction of the generation length may stop the learning before learning well, and the restriction of the population size may increase the learning time. To be accurate, the restriction of population size of every generation is the first choice of the evolution strategies and the generation length is set to be very large.

It deserves to be mentioned that there are  $(2n+1)$  children with their own optimal minimum after the learning process if there are  $n$  parents. It tells the number of the individuals is larger than the last generation. Generation by generation, the number of the individuals will become very large and that will increase the learning computation time seriously. Therefore, the population size of every generation should be restricted. The elite process is introduced to solve the problem by choosing some child to be the parent of the next generation. It is an issue which individuals should be kept for the next generation. They can be chosen randomly from the  $(2n+1)$  children, but, in this thesis, they are chosen depending on their own optimal fitness. To find  $n$  individuals for the next generation, the children are sorted by their fitness. Then, the first  $n$  children which contain the largest fitness will be kept. In other words, it is expressed as

$$W_k^{g+1} = \Omega_{k,\text{opt}}^g \quad (3.32)$$

where  $\Omega_{k,\text{opt}}^g$  contains the  $k^{\text{th}}$  largest fitness.

According to the above methods, the evolution strategies start from creating  $n$  initial individuals. Then do the reproduction process to produce the  $(2n+1)$  children. For the  $(2n+1)$

children, use the learning process to find their own optimal fitness. At this time, the elite process is used to reduce the number of the individual to  $n$ , and it keeps the number of the individuals unchanged generation by generation. The reproduction process, the learning process and the elite process repeats again and again until the largest fitness reach the desired goal or is unchanged for several generations. The flow chart of the evolution strategies is shown in the next section.



### 3.3.5 Flow Chart

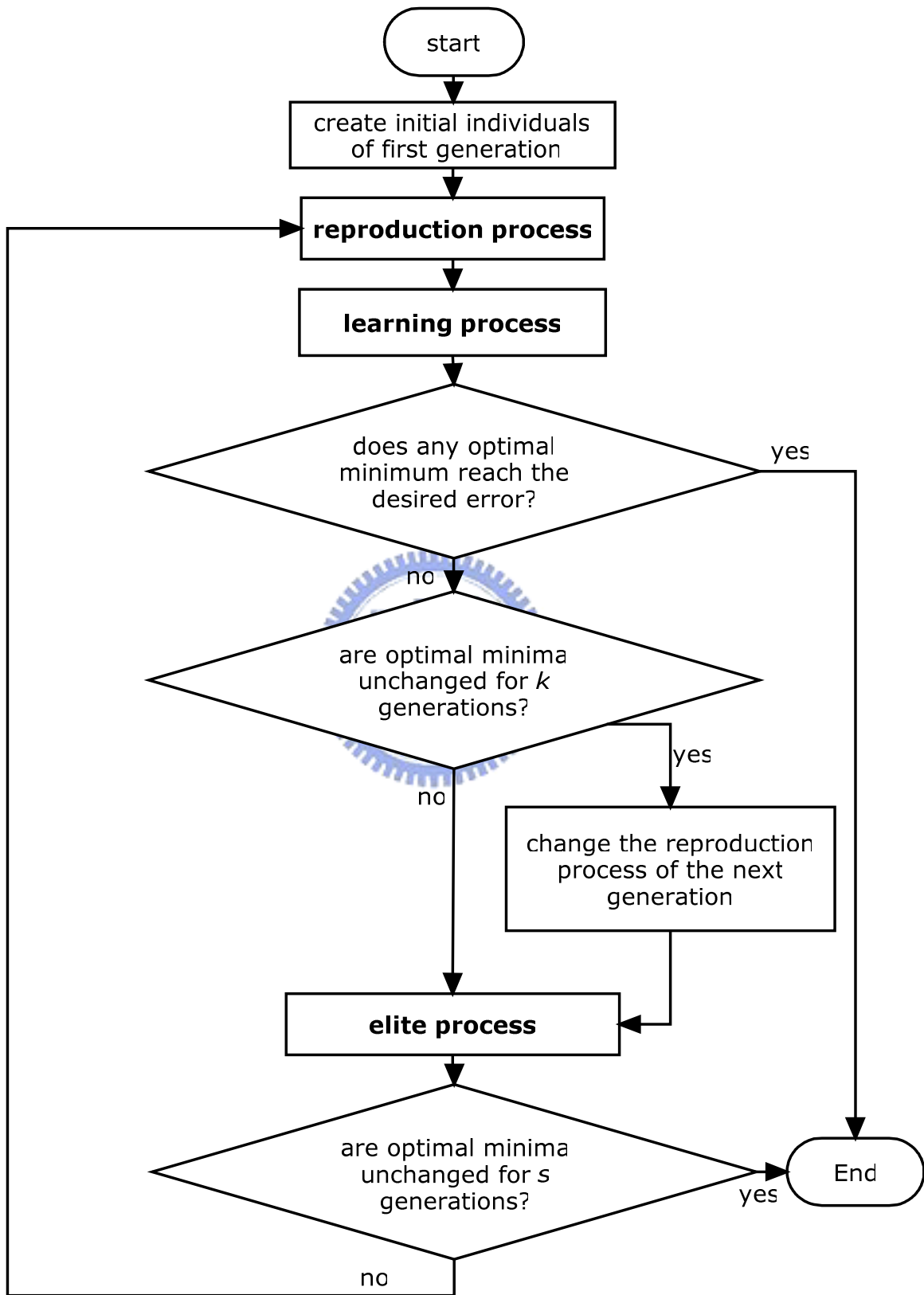


Figure 3.8 the flow chart of the evolution strategies

# Chapter 4

## Simulation Results

In this chapter, the objective system of the neural network is a first order LTI system, described as

$$\dot{y}(t) + y(t) = u(t) \quad (4.1)$$

for simplicity. According to the system, the training data are captured with given sampling times when the input function is a step function whose amplitude is equal to 1 and the system initial condition are idled. The fitness function is defined as the negative sum of the errors between the outputs of the first order LTI system and the NN system. The learning procedure indicated in the last chapter was implemented by a Matlab program.

In last chapter, it is mentioned that there are some settings which will affect the performance of the evolution strategies, such as the initial individuals and sampling times. Here, the influence of these settings will be discussed, and then the learned neural network using the evolution strategies will be implemented as a controller. The influence of the sampling times will be discussed in Section 4.1, the influence of the initial individuals creation will be discussed in Section 4.2, and then Section 4.3 will show the results of the neural network trained as a controller.

## 4.1 Influence of The Sampling Time

The influence resolves itself into the following two points: one is the influence of the sampling time to learning result; the other is the abilities of two neural network structures to adapt different sampling times. The first point will be discussed in the following paragraph.

In the first order difference equation, it showed that the fitness increases as the sampling time decreases. It is concerned whether the fitness of the two neural network structures increase as the sampling time decrease like the first order difference equation. There remains a second question about whether the sampling time affects the success rate of the learning result or not. Here, the GS and SST are trained under the sampling time 0.01 and 0.001. Fig. 4.1.1 shows the learning result of the GS when the training data are under sampling time 0.01 at 1<sup>st</sup>, 3<sup>rd</sup>, and 5<sup>th</sup> time. Fig 4.1.2 shows the variation of negative fitness, sum of the error, during the learning process of the GS when the training data are under sampling time 0.01 at 1<sup>st</sup>, 3<sup>rd</sup>, and 5<sup>th</sup> time. Similarly, the objective of Fig. 4.2.1 and Fig 4.2.2 is the SST under the sampling time 0.01, the objective of Fig. 4.3.1 and Fig 4.3.2 is the GS under the sampling time 0.001, and the objective of Fig. 4.4.1 and Fig 4.4.2 is the SST under the sampling time 0.001. Table 4.1 presents the learning results whose initial individual are all given randomly. We define the learning as success learning while the average of the error is smaller than 0.05, and show the result in the Table 4.2.1 and Table 4.2.2. Every case is learned by starting with three different initial random individuals.

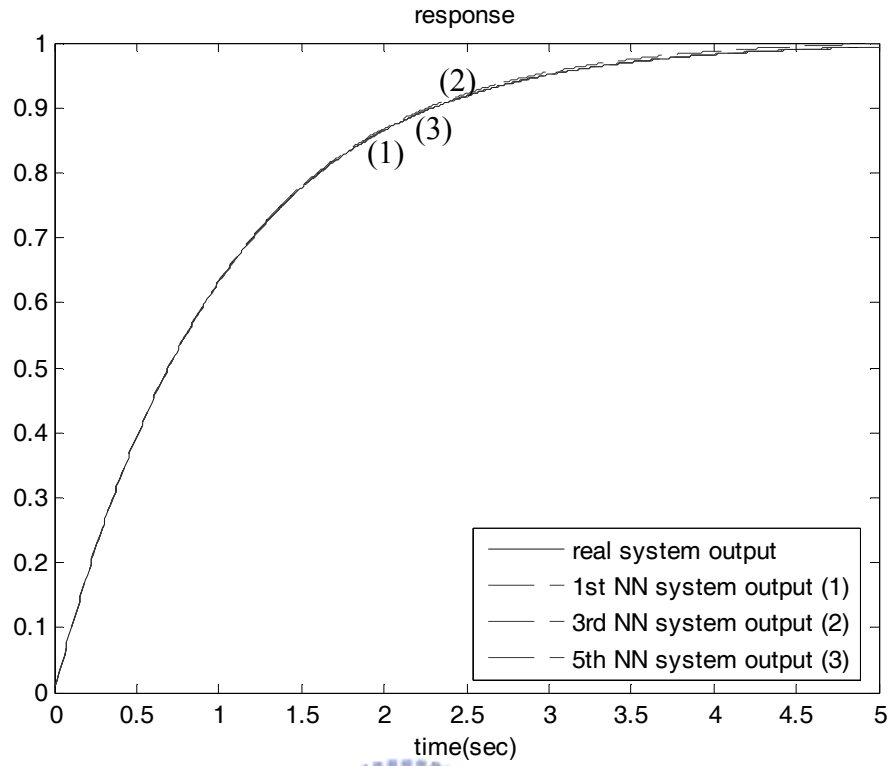


Figure 4.1.1 the learning result of the GS under sampling time 0.01

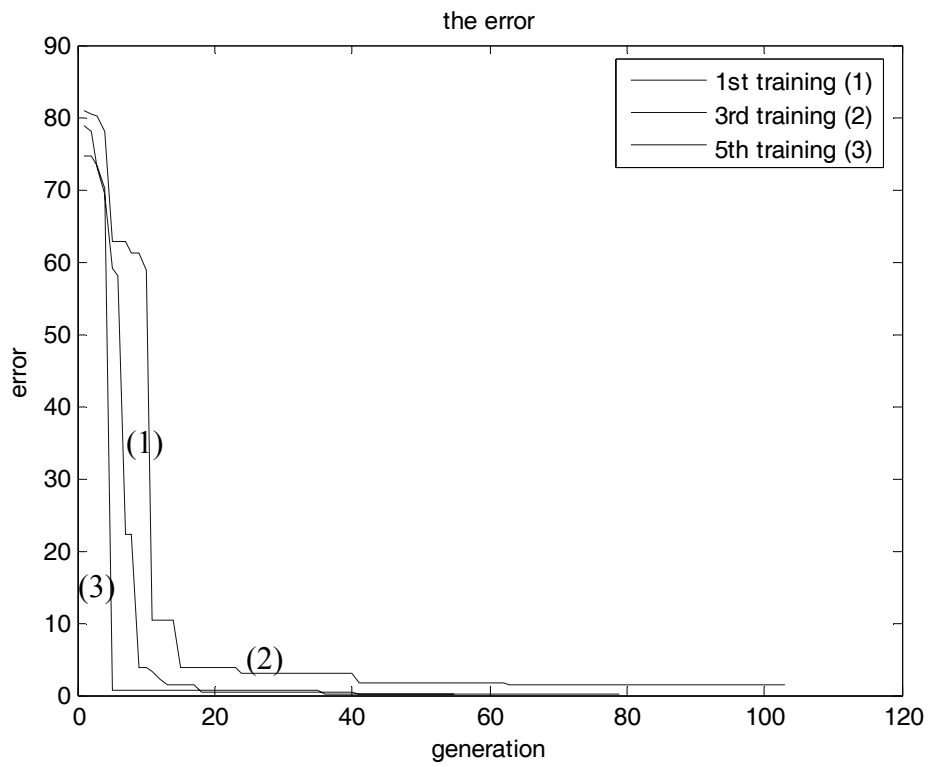


Figure 4.1.2 the change of the sum of the error of the GS under sampling time 0.01

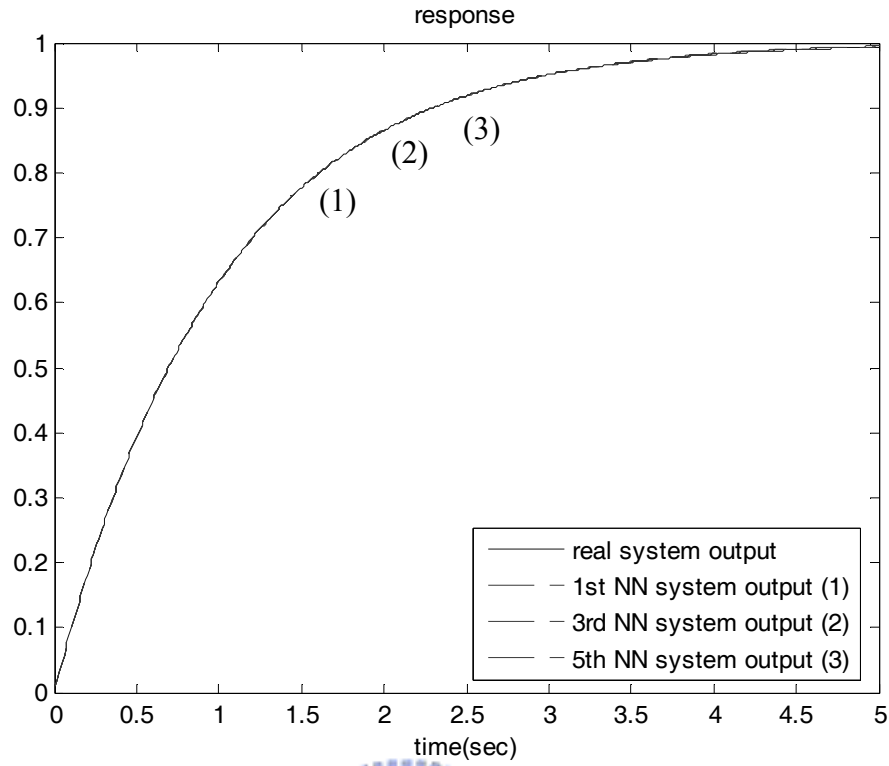


Figure 4.2.1 the learning result of the SST under sampling time 0.01

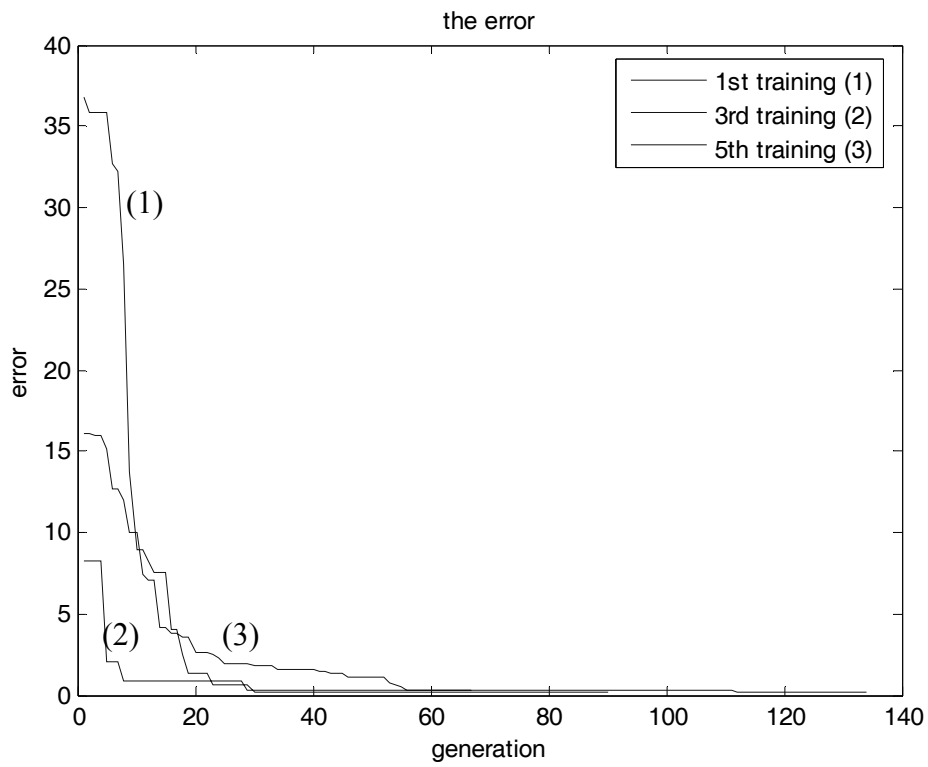


Figure 4.2.2 the change of the sum of the error of the SST under sampling time 0.01



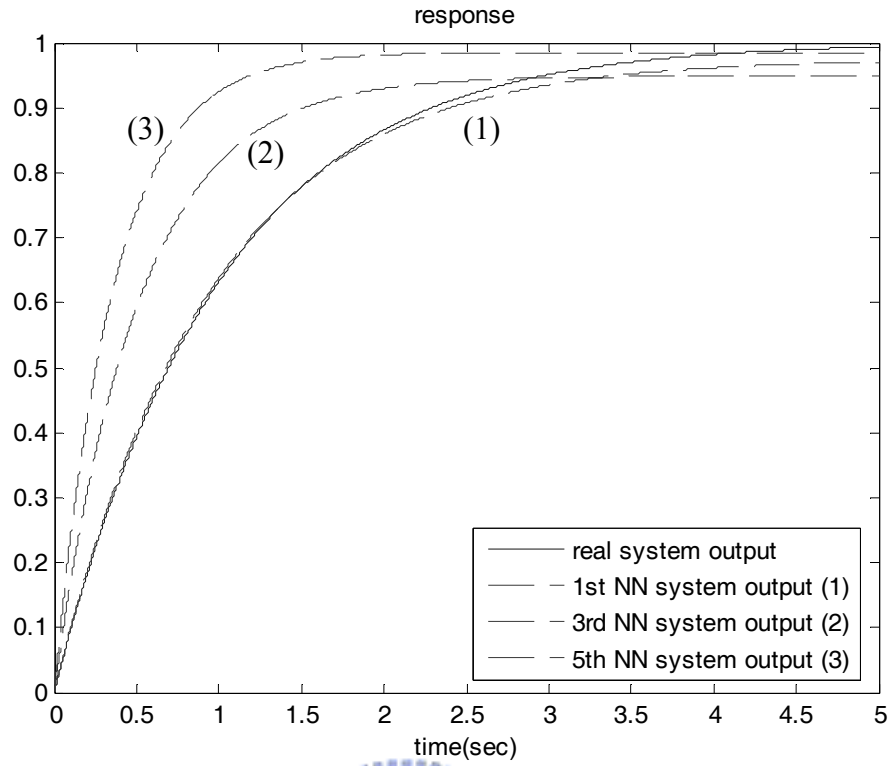


Figure 4.3.1 the learning result of the GS under sampling time 0.001

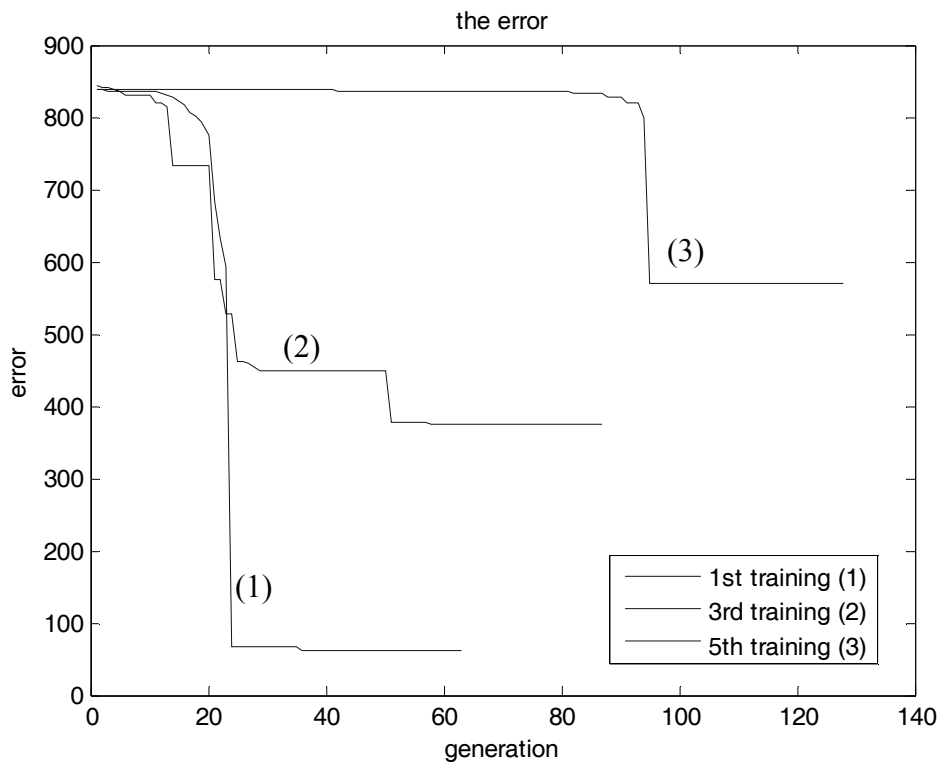


Figure 4.3.2 the change of the sum of the error of the GS under sampling time 0.001

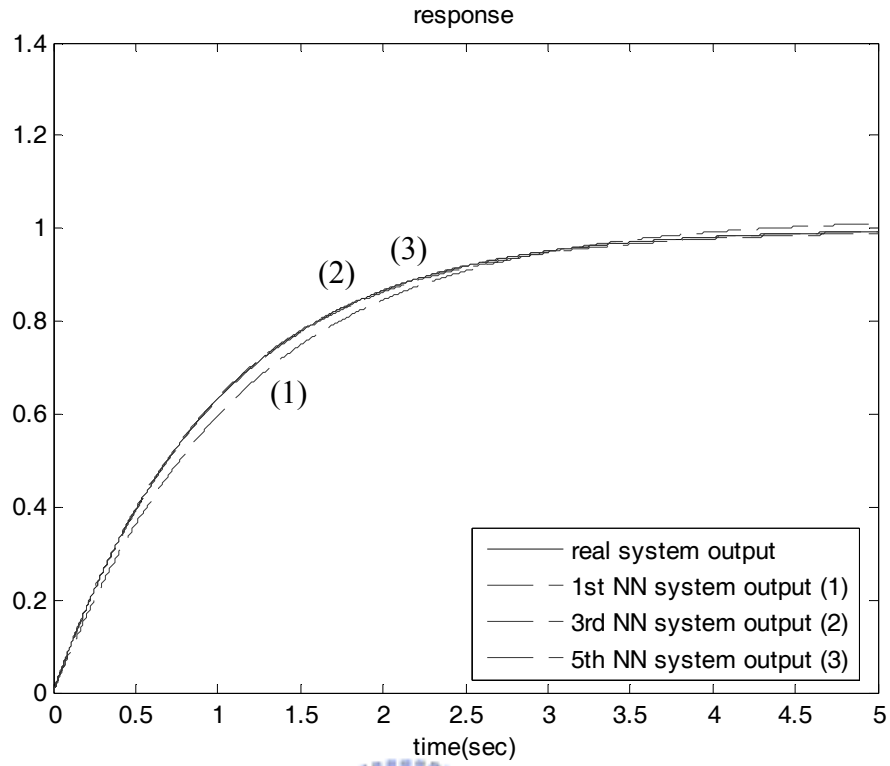


Figure 4.4.1 the learning result of the SST under sampling time 0.001

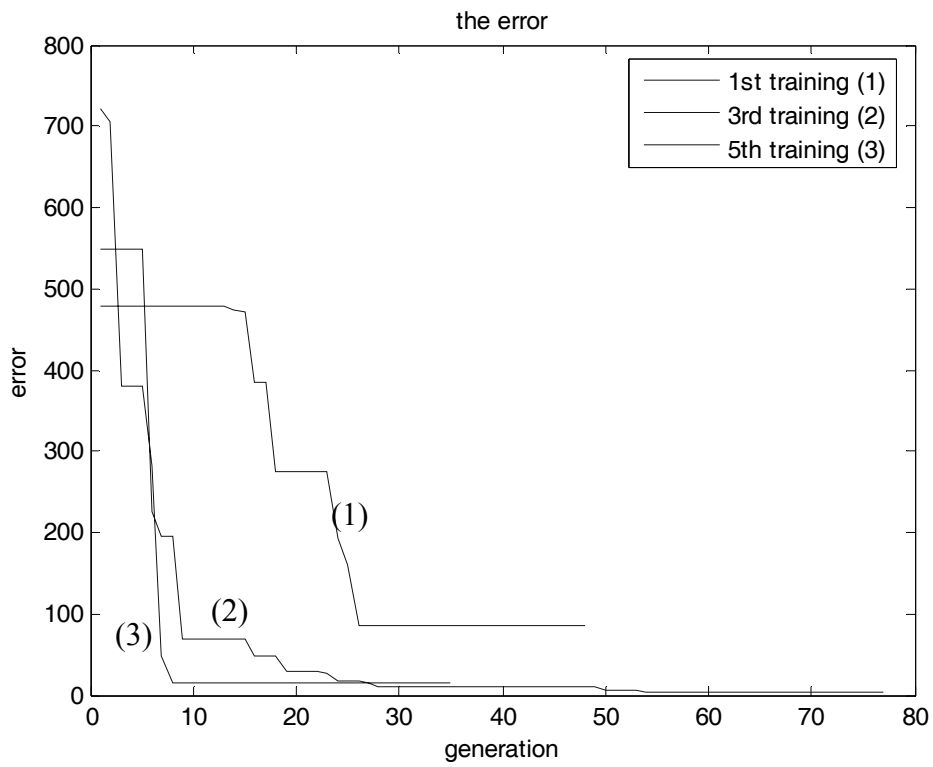


Figure 4.4.2 the change of the sum of the error of the SST under sampling time 0.001

Table 4.1 the results of learning with different sampling times

	$\Delta T = 0.01$				$\Delta T = 0.001$			
	generation length		learning time		generation length		learning time	
	GS	SST	GS	SST	GS	SST	GS	SST
1 <sup>st</sup>	78	89	01:34:55	00:08:13	62	47	01:21:54	00:33:07
2 <sup>nd</sup>	36	88	00:50:07	00:09:01	485	27	15:27:51	00:16:57
3 <sup>rd</sup>	102	66	01:50:32	00:09:27	86	76	01:12:17	00:55:54
4 <sup>th</sup>	81	38	01:32:53	00:12:34	94	53	01:43:28	00:39:49
5 <sup>th</sup>	54	133	01:34:51	00:07:21	127	34	04:42:18	00:28:33
average	70	83	01:28:40	00:09:19	171	47	04:53:34	00:24:52

Table 4.2.1 the error and the accurate rate of learning with sampling time 0.01

	$\Delta T = 0.01$					
	Sum of error		average of error ( $10^{-2}$ )		Success/Fail	
	GS	SST	GS	SST	GS	SST
1 <sup>st</sup>	0.0740	0.1784	0.01480	0.03568	S	S
2 <sup>nd</sup>	0.2939	0.1538	0.05878	0.03076	S	S
3 <sup>rd</sup>	1.5743	0.2525	0.31486	0.05050	S	S
4 <sup>th</sup>	1.8689	0.4242	0.37378	0.08484	S	S
5 <sup>th</sup>	0.1782	0.1291	0.03564	0.02582	S	S
average	0.7979	0.2276	0.15957	0.04552	Success rate(%)	100

Table 4.2.2 the error and the accurate rate of learning with sampling time 0.001

	$\Delta T = 0.001$						
	Sum of error		average of error ( $10^{-2}$ )		Success/Fail		
	GS	SST	GS	SST	GS	SST	
1 <sup>st</sup>	61.3104	85.9001	1.22621	1.71800	S	S	
2 <sup>nd</sup>	144.4808	118.1299	2.88962	2.36260	S	S	
3 <sup>rd</sup>	374.9141	4.1978	7.49828	0.08396	F	S	
4 <sup>th</sup>	131.3523	32.7474	2.62705	0.65495	S	S	
5 <sup>th</sup>	570.8313	15.3980	11.41663	0.30550	F	S	
average	256.5779	51.2746	5.13156	1.02500	Success rate(%)	60	100

The Fig 4.1.1 and Fig. 4.2.1 show that the learning results are almost the same with the system response when the sampling time is 0.01. Fig. 4.3.1 and Fig. 4.4.1 show that the learning results have some error from the first order LTI system response when the sampling time is 0.001, although the direction of the trend is the same. It means the fitness of the proposed neural network structures increases as the sampling time decrease. Besides, Table 4.1, Table 4.2.1, and Table 4.2.2 tell that the error has no relationship with generation length, and the learning time is independent with the value of the generation length. The Table 4.1 also shows that the learning time increases as the sampling time decrease in the same neural network structure, and the average learning time of the SST is shorter than the GS. Table 4.2.1 and Table 4.2.2 indicate that the larger sampling time produce smaller fitness. It is similar with the first order difference equation. Besides, compared to the success rate, it is clear that the success rate of the GS decreases as the sampling time decreases.

Since the simulation results show errors are small, i.e. fitness are large, take the best individual with  $\Delta T=0.01$  to test the dynamic of the NN system. Here, two significant conditions for a system are variable for validation: the one is the initial condition  $y(0)$  of the system, and the other is the input function. Let the initial condition  $y(0)$  be 0.5, 2, and -2, and the input function be  $2p(t)$ ,  $-2 p(t)$ , and sine function where  $p(t)$  is a unit step function. The testing results of the initial conditions are shown in Fig. 4.5.1 to Fig. 4.5.3, and the testing results of the input functions are shown in Fig. 4.6.1 to Fig. 4.6.3.

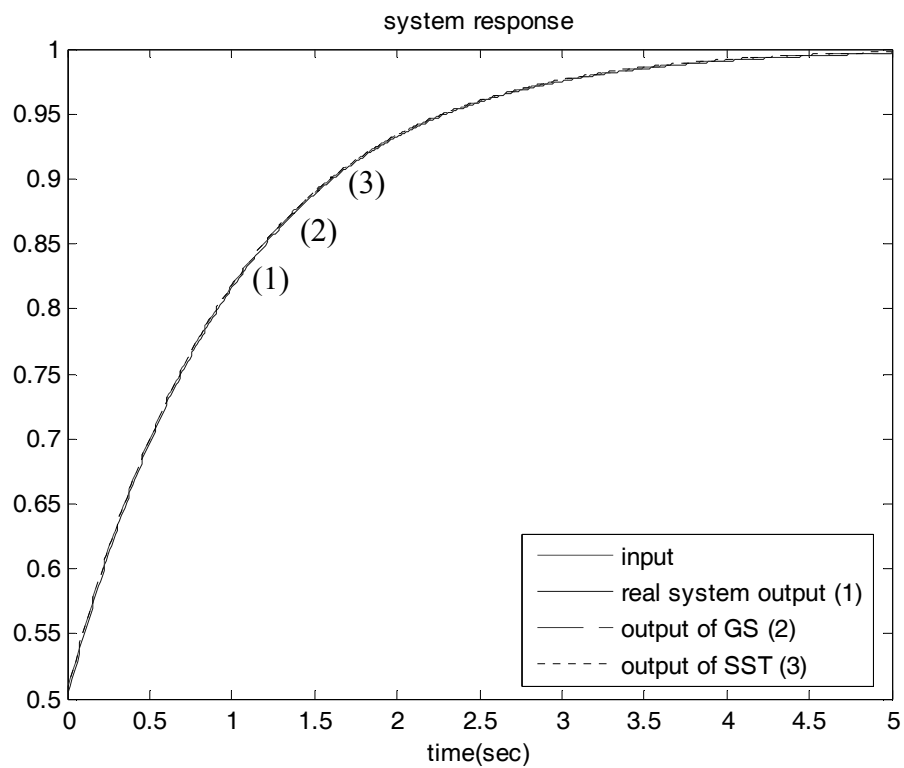


Figure 4.5.1 testing result when initial condition  $y(0)$  is 0.5

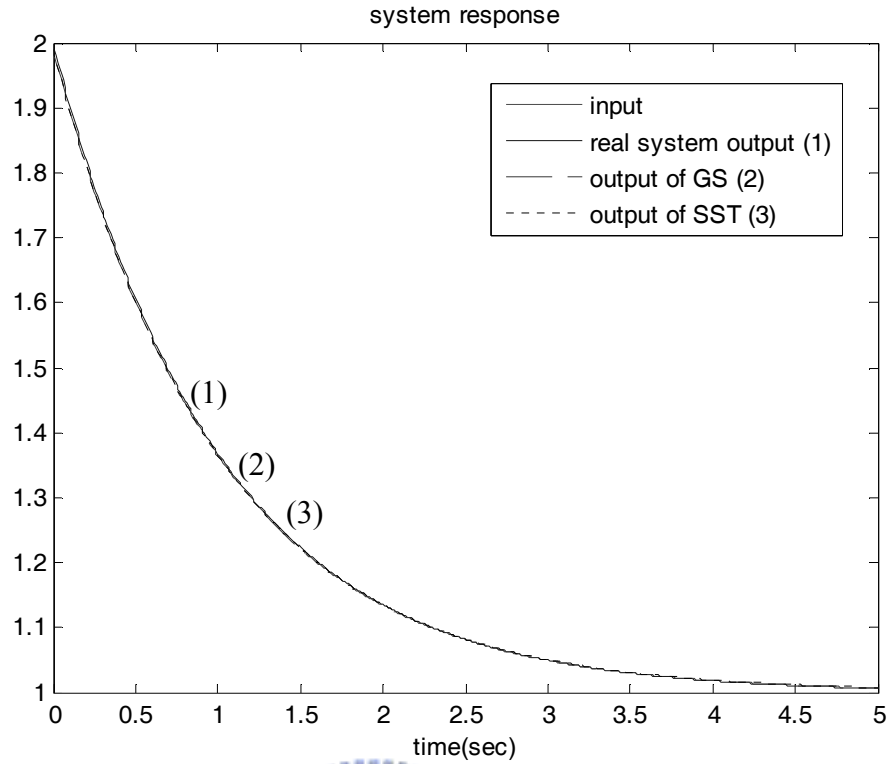


Figure 4.5.2 testing result when initial condition  $y(0)$  is 2

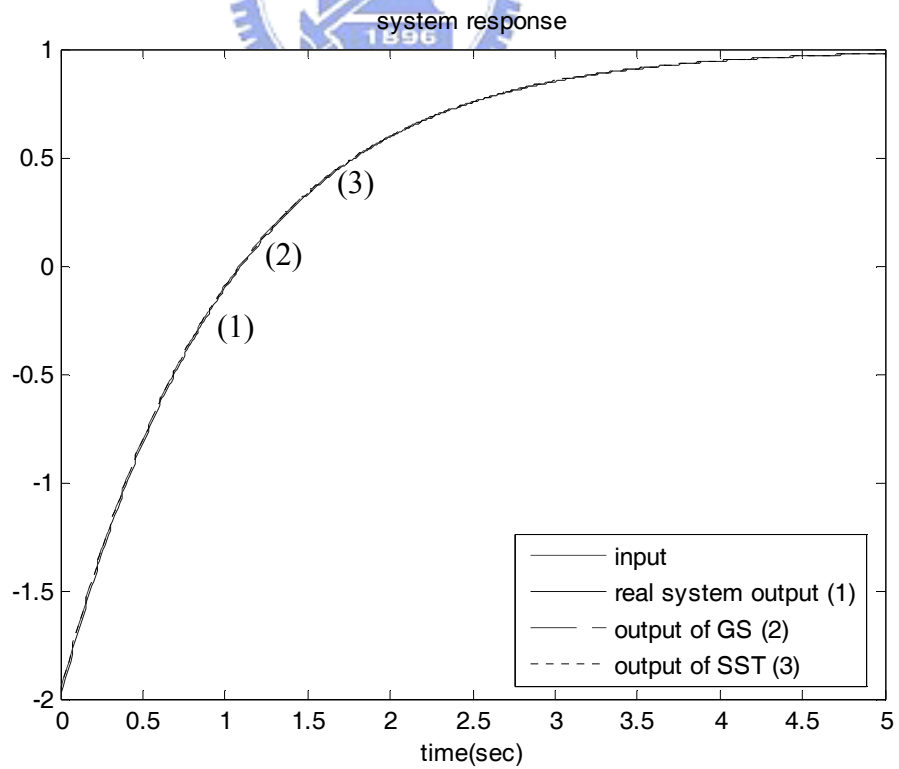


Figure 4.5.3 testing result when initial condition  $y(0)$  is -2

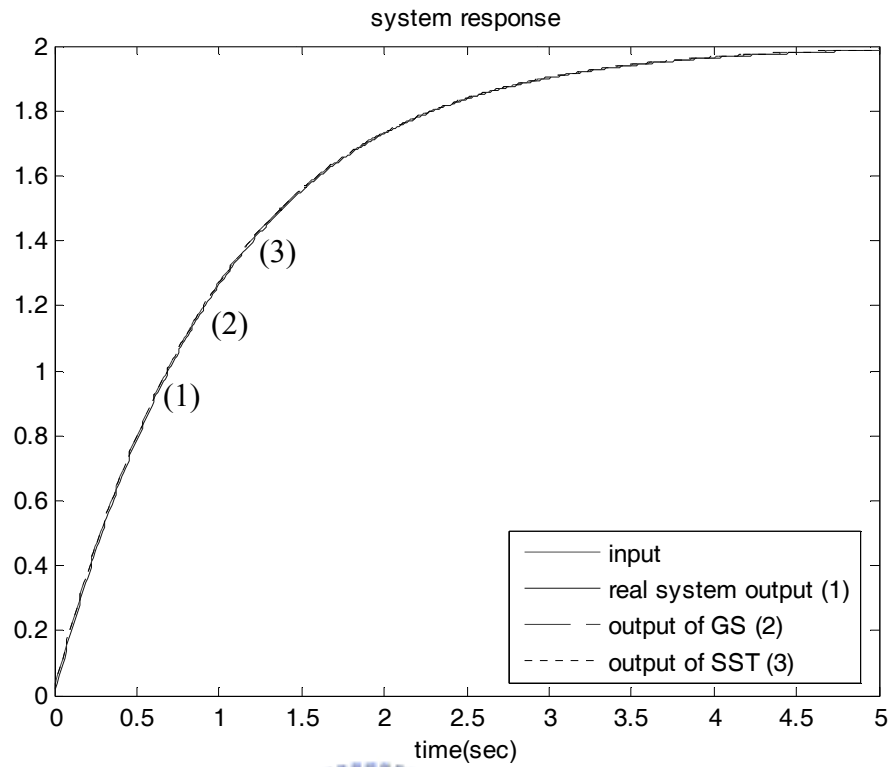


Figure 4.6.1 testing result when input function  $u = 2p(t)$

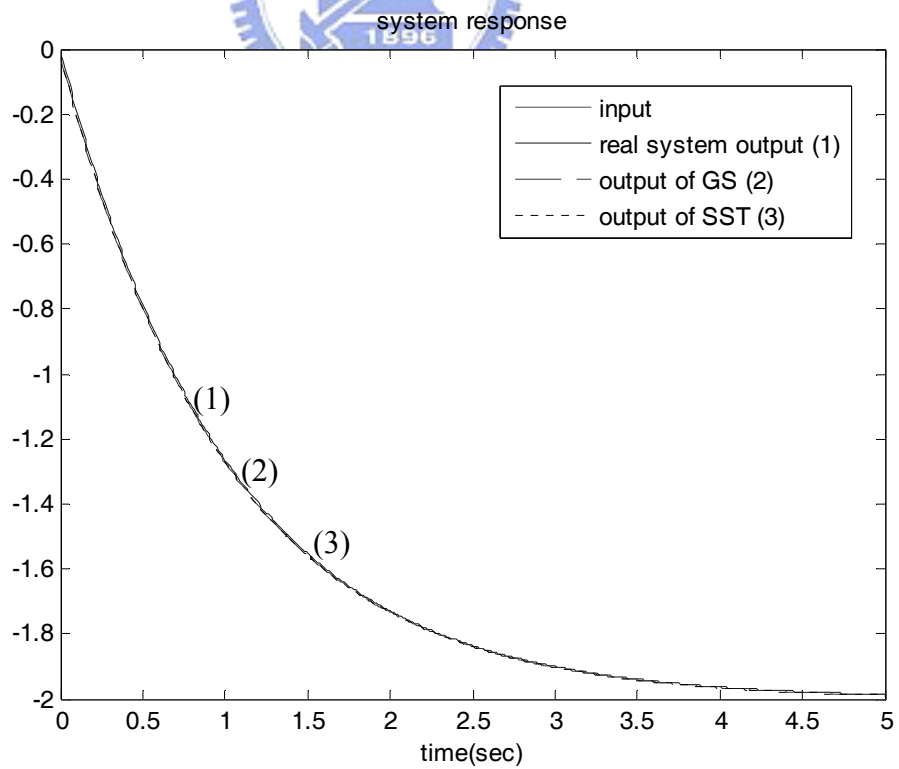


Figure 4.6.2 testing result when input function  $u = -2p(t)$

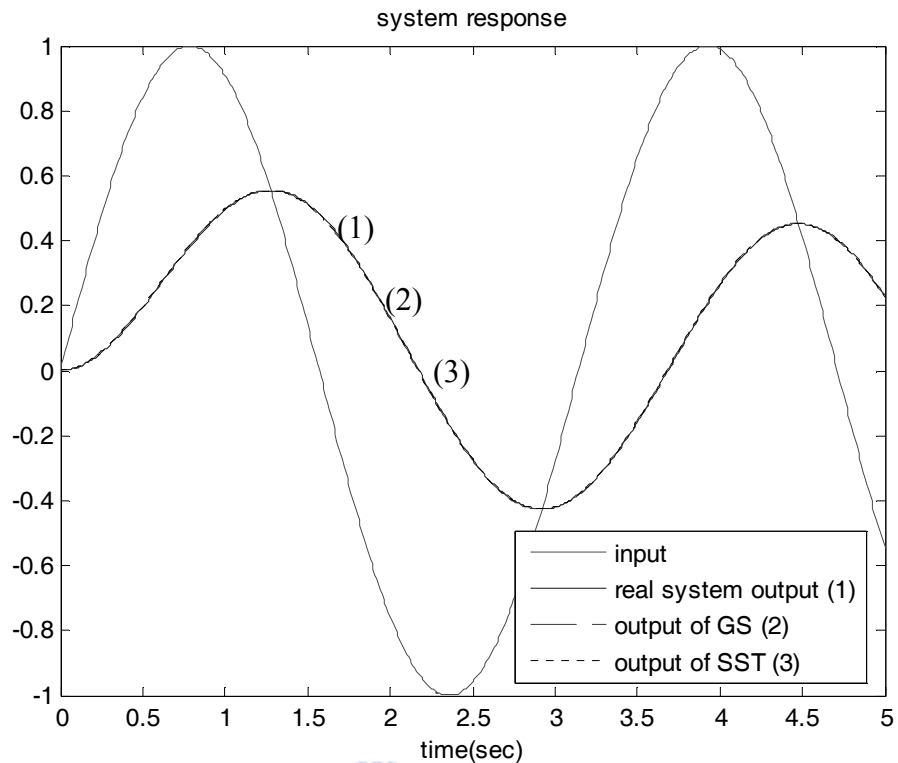


Figure 4.6.3 testing result when input function  $u = \sin(2t)$

Fig 4.5.1 to Fig 4.6.3 indicate the neural networks using evolution strategies don't just learn as a specific curve, but the input-output relationship of the objective system. In another words, the learned neural network system behaves similar with the first order LTI system no matter what the initial condition or the input function is. Thus, these results lead to the conclusion that that the neural network using evolution strategies could learn well as a first order LTI system.

Since the influence of the sampling time to the fitness does exist, the abilities of two structures to adapt different sampling times must be recalled here. To verify the abilities, take the best learning result under the sampling time 0.01, to test the system under the smaller sampling time 0.005 whose result is shown as Fig. 4.7, and to test under the much smaller sampling time 0.001 whose result is shown as Fig. 4.8.



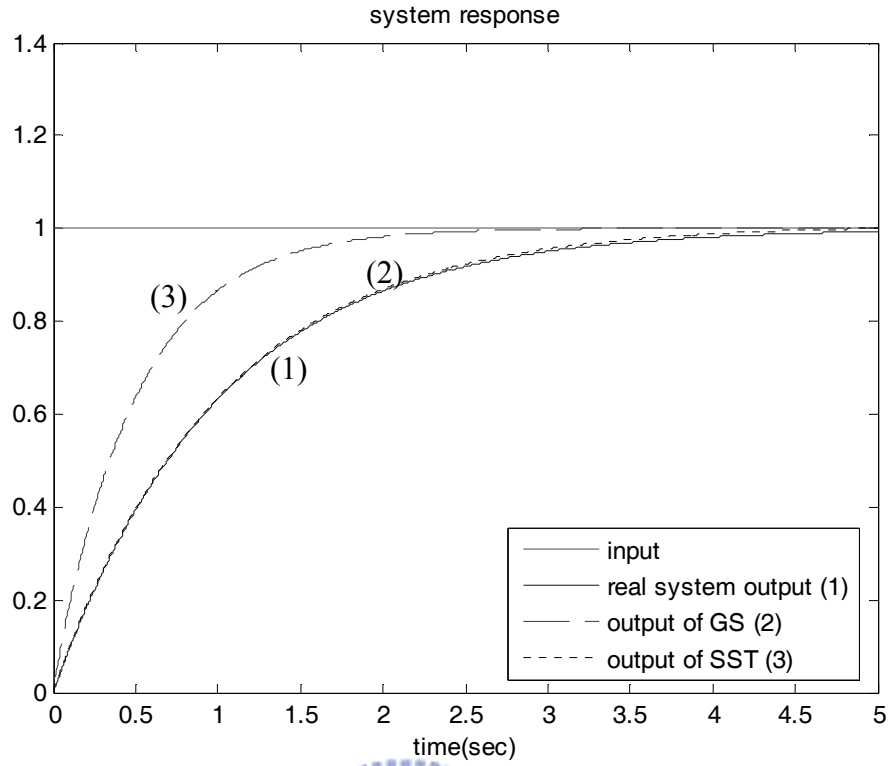


Figure 4.7 the testing result with  $\Delta T=0.005$  of the learned neural network with  $\Delta T=0.01$

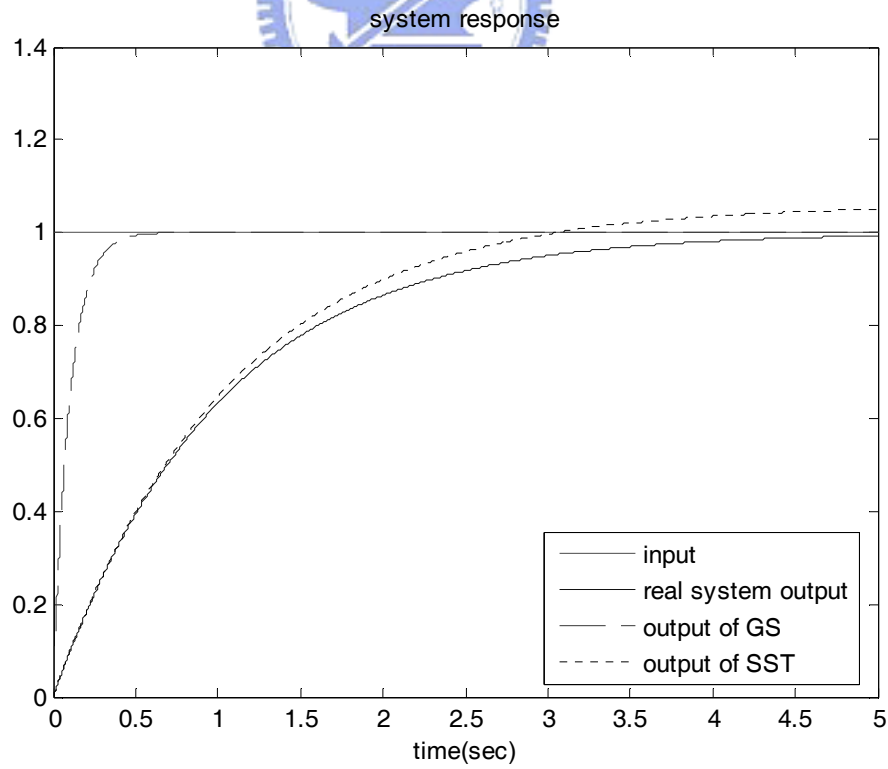


Figure 4.8 the testing result with  $\Delta T=0.001$  of the learned neural network with  $\Delta T=0.01$

The Fig 4.7 tells that the GS fails, but the SST successes. Fig. 4.7 and Fig. 4.8 indicate that the GS can only be used under a fixed sampling time, but the SST can be used under larger range near the sampling time of the training data. Therefore, the SST is trained with different sampling times, 0.01, 0.001, and 0.0001 using evolution strategies, and then the results will show in Fig. 4.9 and Fig 4.10.1 to Fig. 4.10.3. It demonstrates that the SST can adapt larger range of the sampling time when it is trained under larger range. It concludes that the SST is better than the GS concerning about the sampling time for learning the first order LTI system.

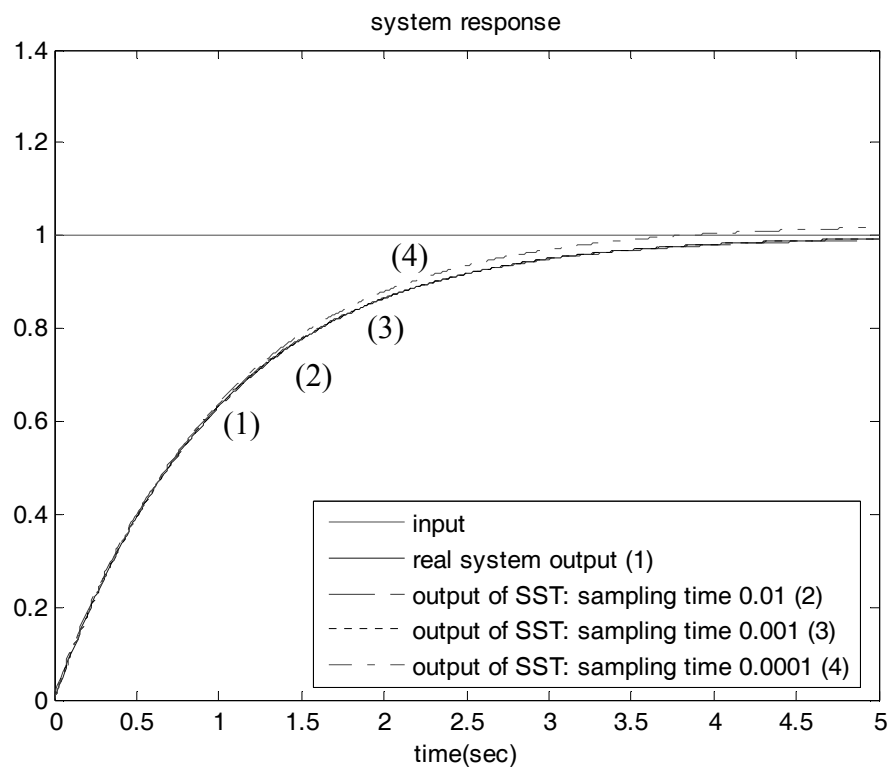


Figure 4.9 the learning result of the SST with sampling time 0.01, 0.001, 0.0001

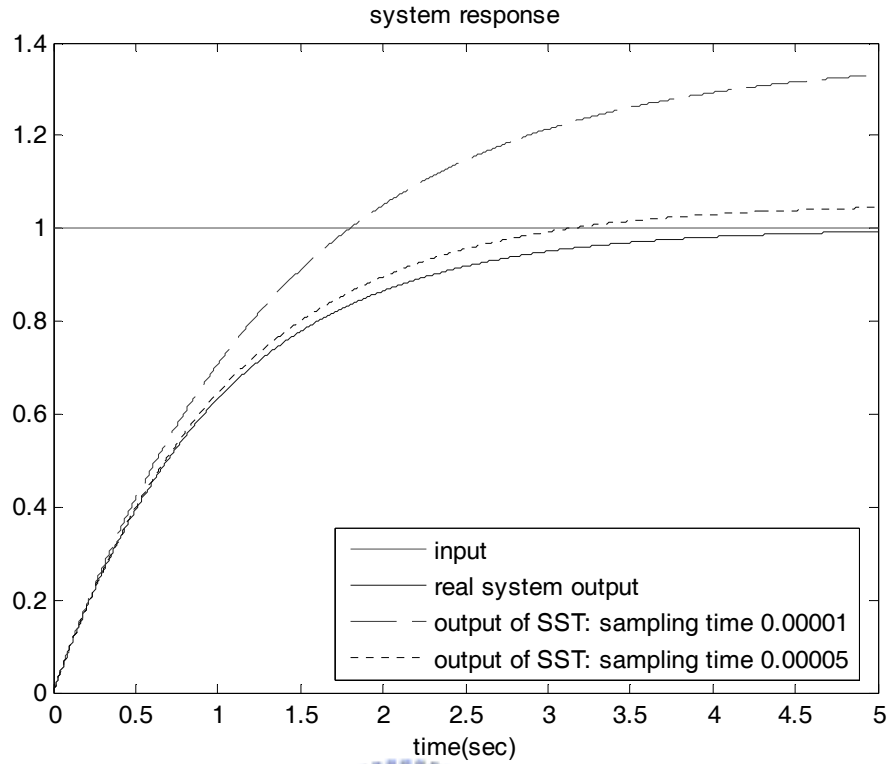


Figure 4.10.1 the testing result under the sampling time smaller than trained sampling time

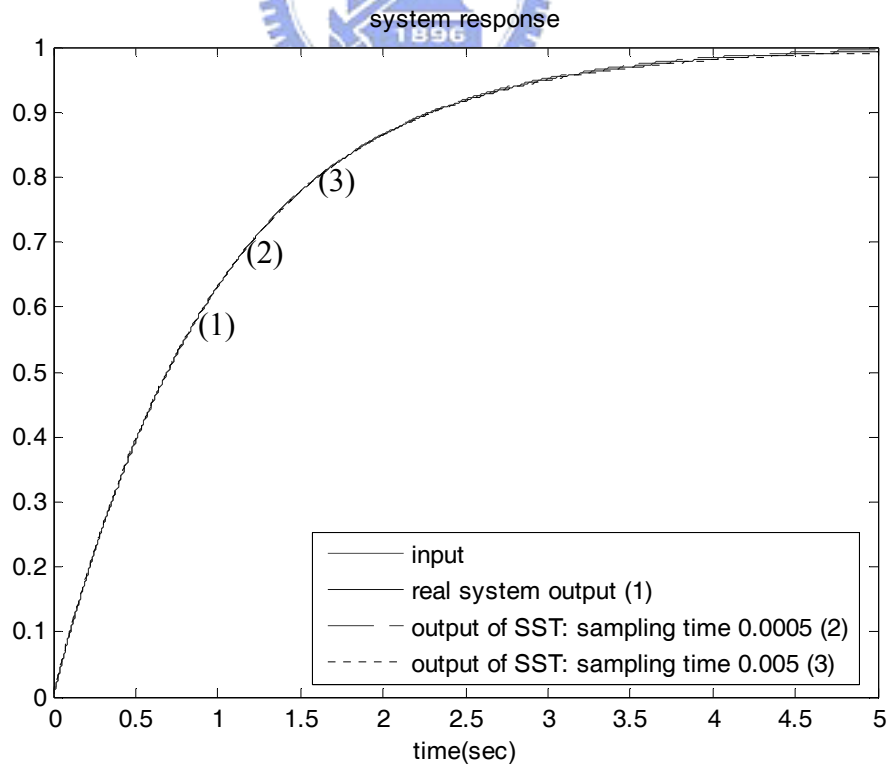


Figure 4.10.2 the testing result under the sampling time between trained sampling time

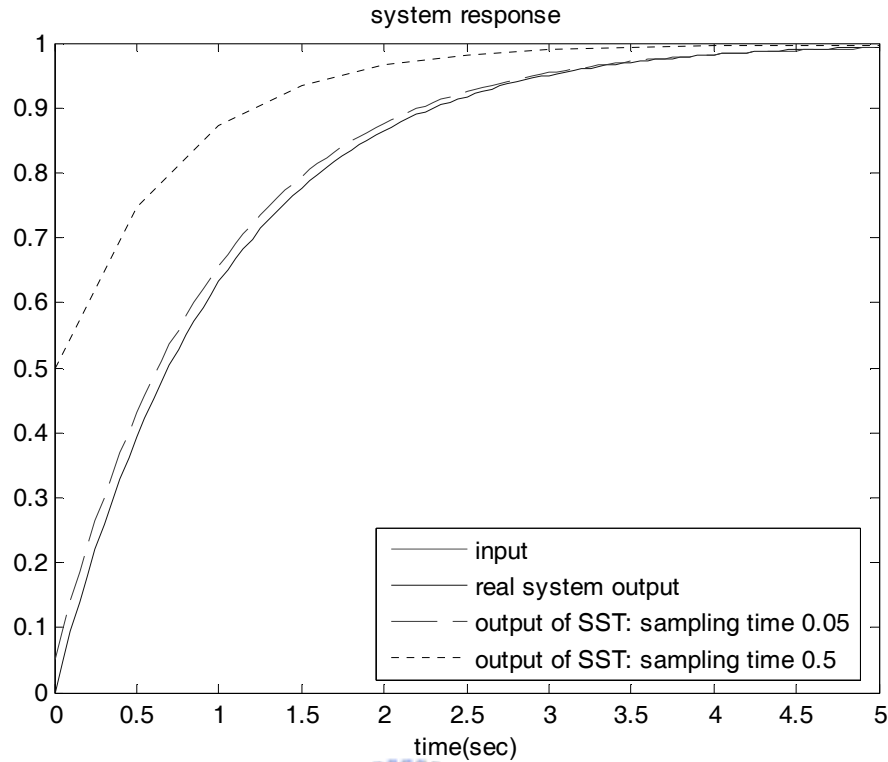


Figure 4.10.3 the testing result under the sampling time larger than trained sampling time



## 4.2 Influence of The Initial Weights Setting

In the last section, it is said that the success rate decreases and the learning time increases as the sampling time decreases. How to let the error reach the global minimum is an important issue for neural network investigators because it is easy to consider the local minimum as the global minimum. Thus, the local minimum may be the main reason for failure learning. However, it is worthy noticing that the first order difference equation provides a set of adequate parameters to approach the first order LTI system. In the following, one of the initial individual is given depending on the parameters of the first order difference equation and compare with the random initial individual. The general structure using the initial individual given by the first order difference equation is called  $GS_i$ . Since the SST has used the parameters of the first order difference equation in the network, SST is not discussed in this

section. The learning objective is the first order LTI system with sampling time 0.001, and one of the initial individual is given by the first order difference equation with sampling time 0.01. The learning result is shown in Figure 4.11.1 to Fig. 4.11.2, and Table 4.3.

Table 4.3 learning results of GS with sampling time 0.001

	GS with $\Delta T = 0.001$					
	generation length	learning time	sum of error	average of error ( $10^{-2}$ )	success/fail	
1 <sup>st</sup>	70	03:02:29	3.9376	0.07875	S	
2 <sup>nd</sup>	49	02:40:50	25.6901	0.51380	S	
3 <sup>rd</sup>	850	06:34:49	1.1476	0.02295	S	
4 <sup>th</sup>	235	14:45:59	22.7393	0.45479	S	
5 <sup>th</sup>	117	05:01:20	93.8409	1.87682	S	
average	264	06:25:05	29.4711	0.58942	success rate (%)	100

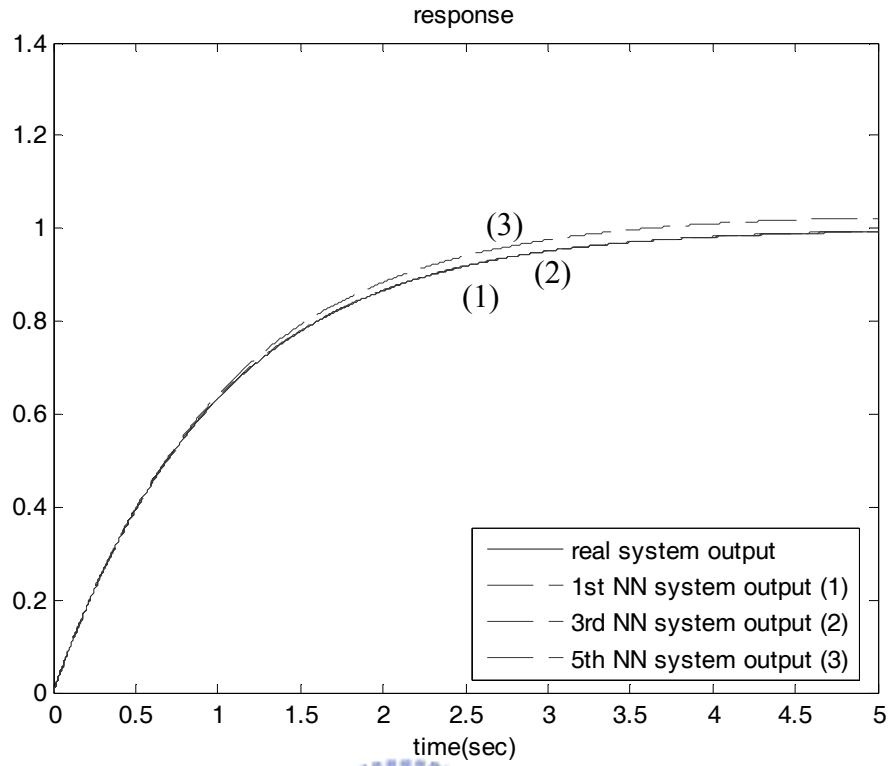


Figure 4.11.1 the learning result of the GS under sampling time 0.001

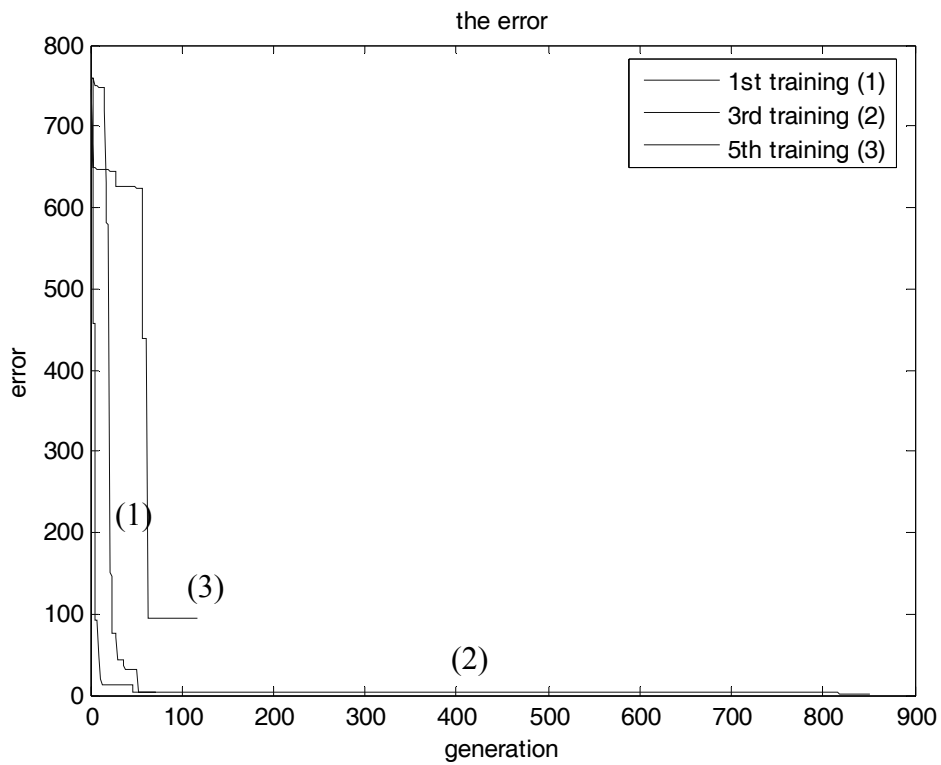


Figure 4.11.2 the change of the sum of the error of the SST under sampling time 0.001

Compared with the Table 4.1 to Table 4.2.2, Table 4.3 indicates that the success rate of  $GS_i$  is larger than GS. It is worthy noticing that the sum of error of 3<sup>rd</sup> learning is smaller than the first order difference equation, 2.0302. It means the GS could perform better than the first order difference equation under the sampling time 0.001. However, the average of the learning time is larger than the last section. It reminds that the learning time is independent of the success rate. Similar to Section 4.1, two significant conditions for a system are also variable for validation: the one is the initial condition of the system, and the other is the input function. The influences of these two conditions are shown in Fig. 4.12.1 to Fig. 4.13.3 using the 3<sup>rd</sup> learning result. Thus, as the figures indicate, no matter what the initial condition and the input functions are, the learned neural network behaves corresponding to the first order LTI system.

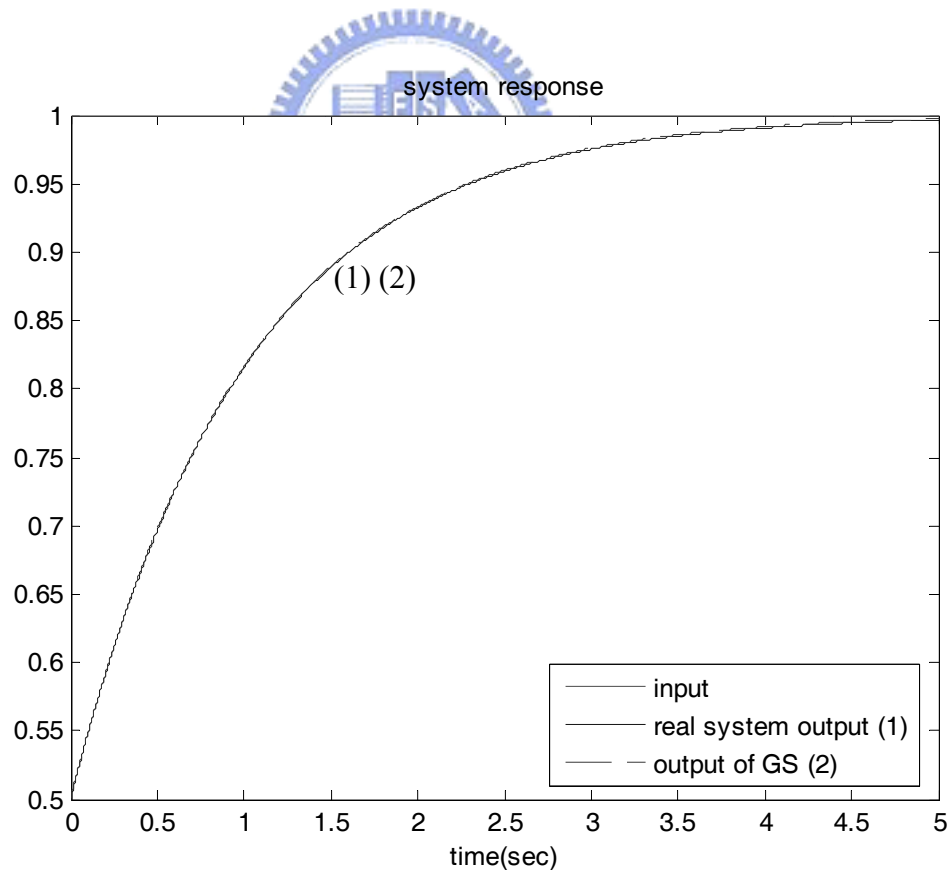


Figure 4.12.1 testing result when initial condition  $y(0)$  is 0.5

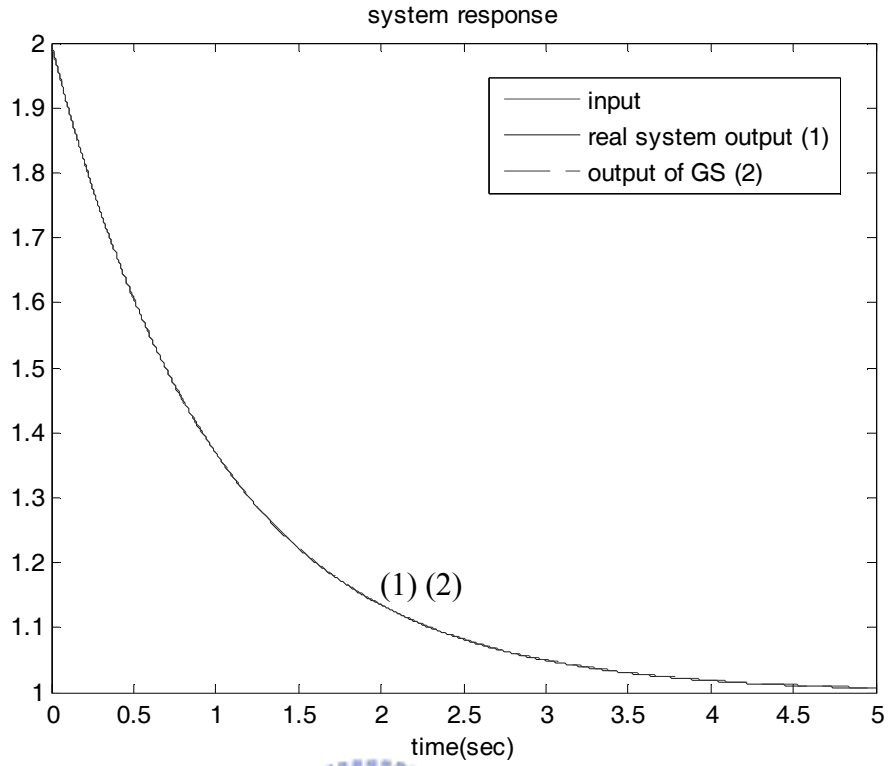


Figure 4.12.2 testing result when initial condition  $y(0)$  is 2

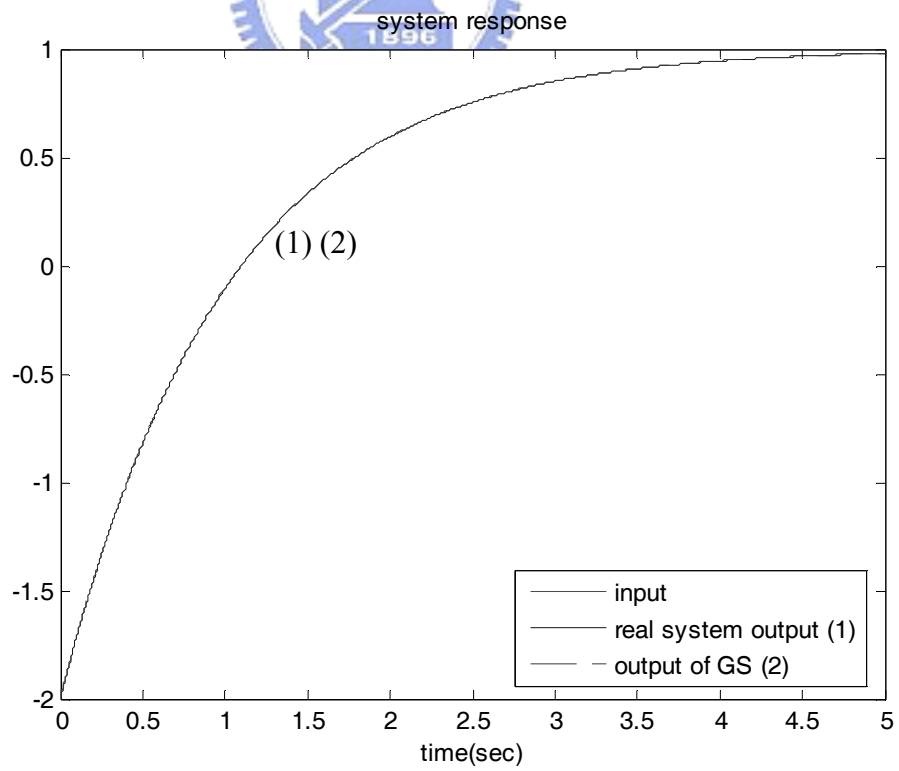


Figure 4.12.3 testing result when initial condition  $y(0)$  is -2



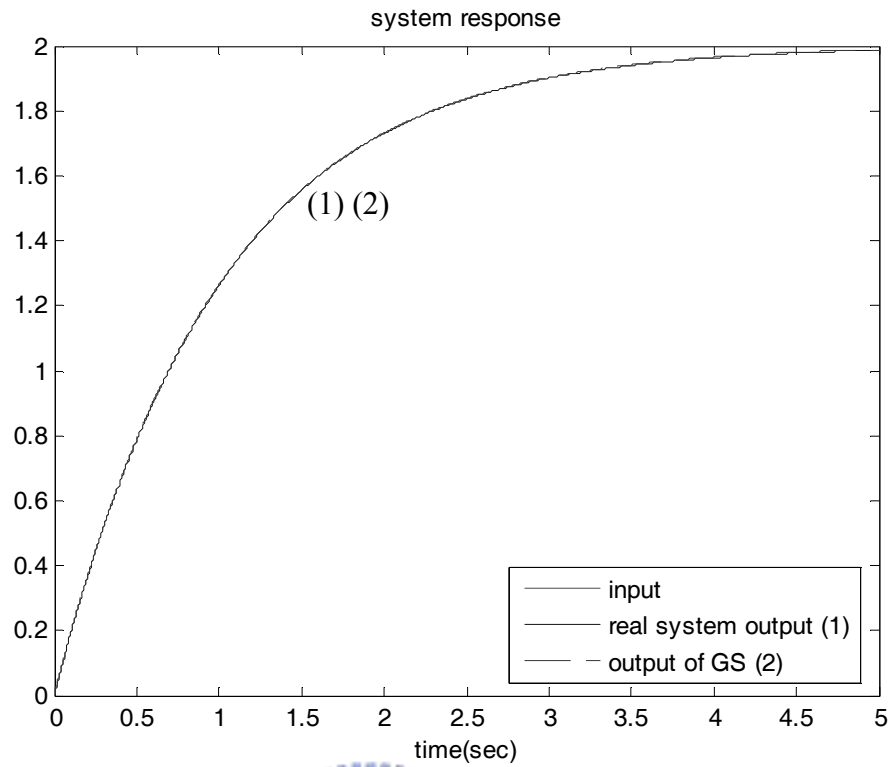


Figure 4.13.1 testing result when input function  $u = 2p(t)$

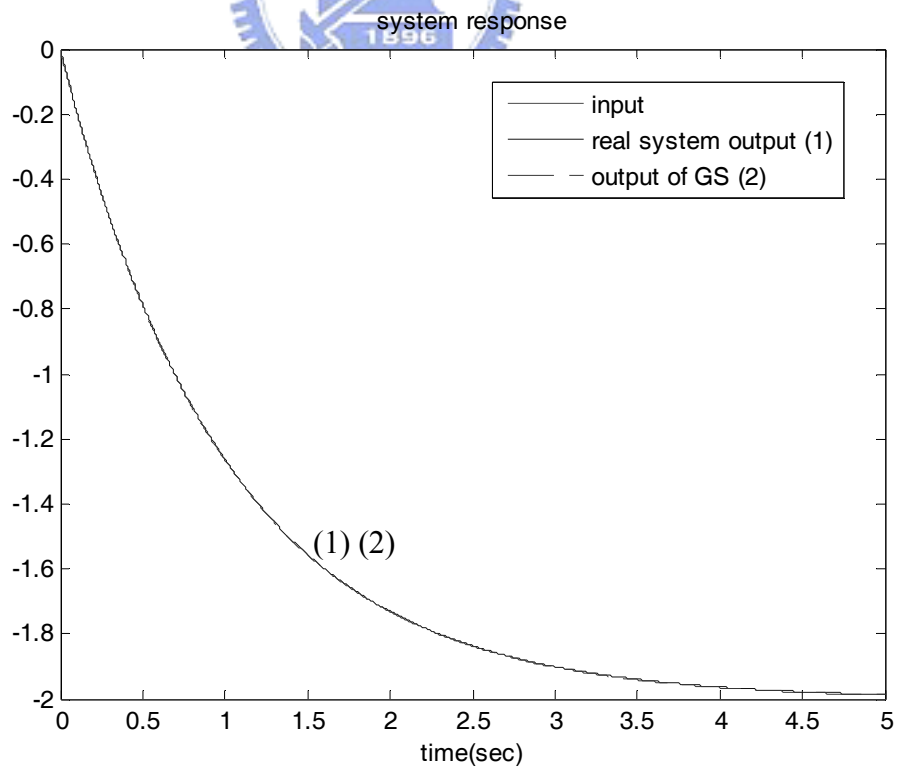


Figure 4.13.2 testing result when input function  $u = -2p(t)$

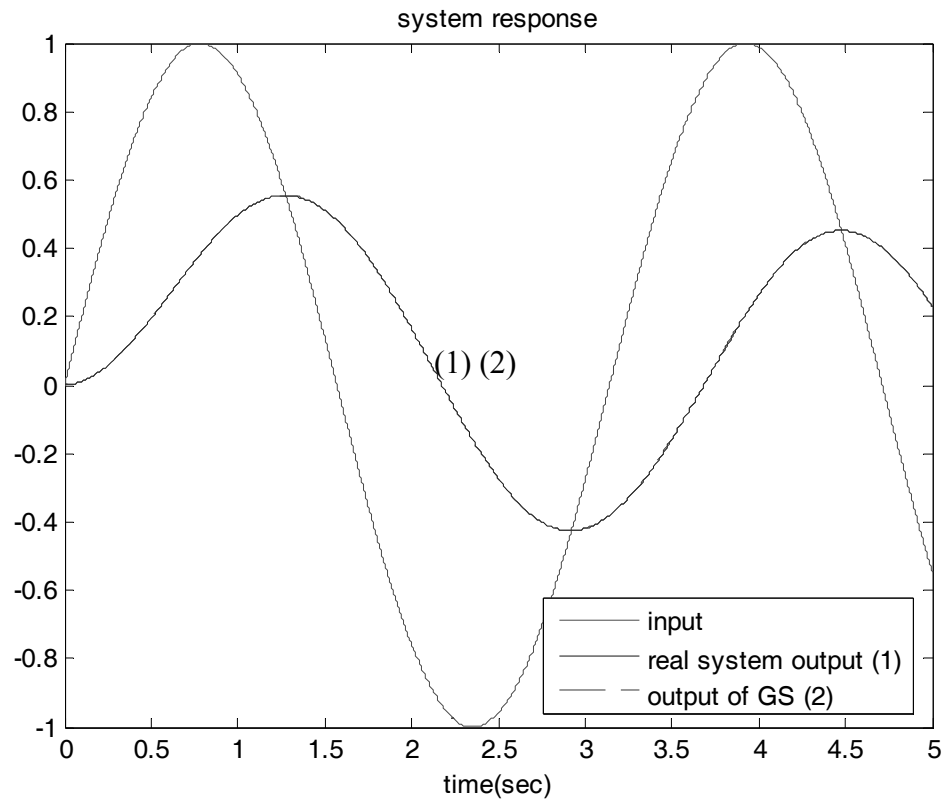


Figure 4.13.3 testing result when input function  $u = \sin(2t)$



### 4.3 Implement as a Controller

According to the results of the last two sections, it concludes that GS and SST using the evolution strategies have good performance on learning a first order LTI system. Further, implement it as a first order LTI controller to control the given plant to verify its ability. Here, a first order and a second order LTI system are the objective plant of the learned controller. Section 4.3.1 use the first order LTI plant to test the learned neural network structures, and Section 4.3.2 use the second order LTI plant to test.

### 4.3.1 System with First Order LTI Plant

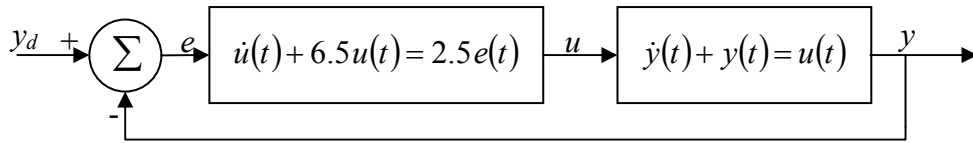


Figure 4.14 the feedback system

In the beginning, a first order LTI system, described as

$$\dot{y}(t) + y(t) = u(t) \quad (4.2)$$

where  $y$  is the system output and  $u$  is the input of the plant, is given as a plant of a feedback system, which is shown as Fig. 4.14. In the system, the neural network is trained as a first order LTI system and replaces the original controller, which is designed as

$$\dot{u}(t) + 6.5u(t) = 2.5e(t) \quad (4.3)$$

where the  $e$  is the error between the desired output and the system output.

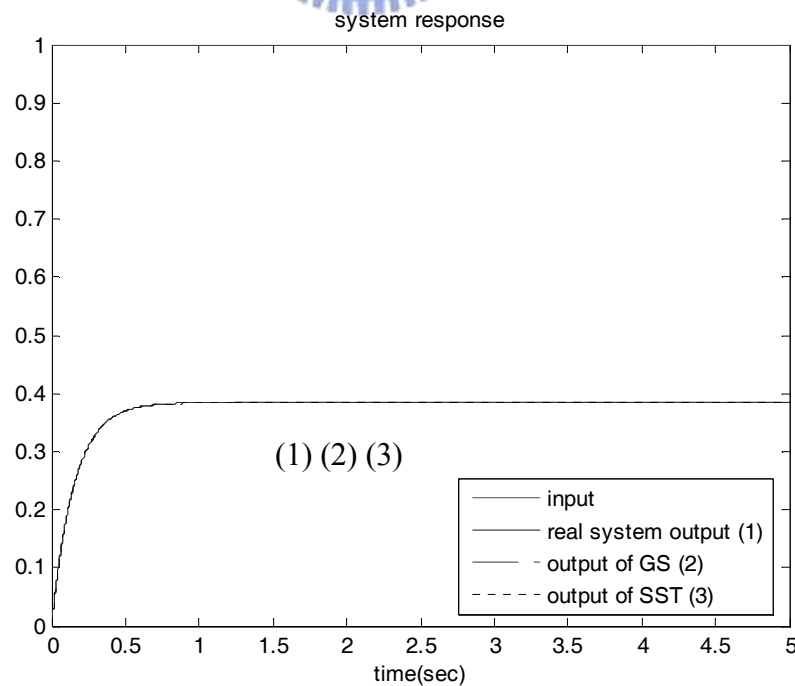


Figure 4.15 the learning result of a controller:  $\dot{u}(t) + 6.5u(t) = 2.5e(t)$

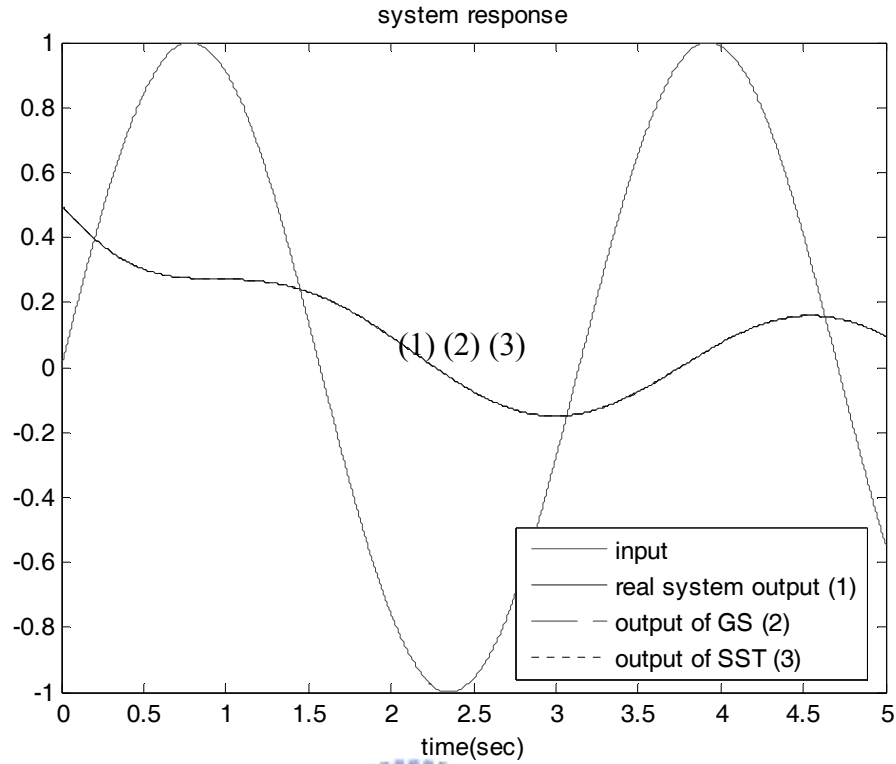


Figure 4.16 the testing result of the feedback system with a neural network controller

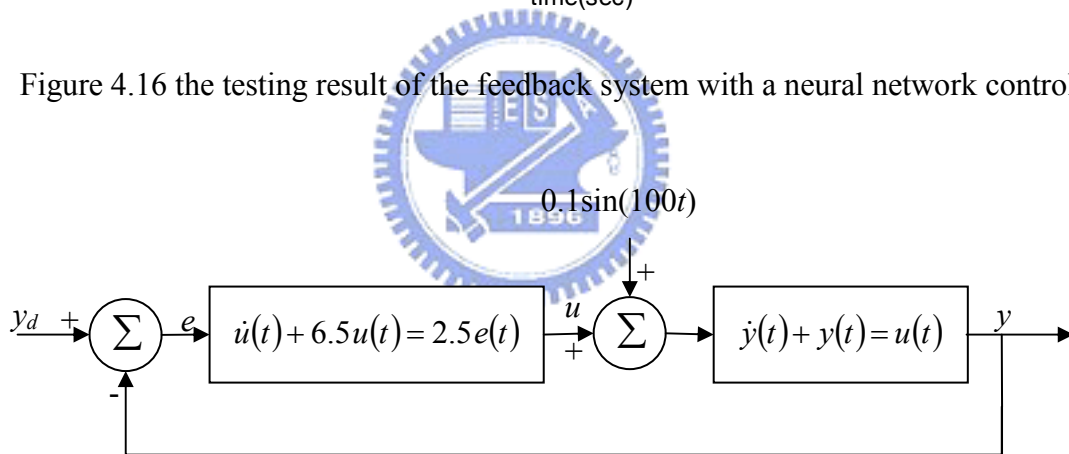


Figure 4.17 the feedback controller with disturbance

The neural networks, SST and  $GS_i$ , are trained as (4.3) with sampling time 0.001, the input function is a unit step function. The Fig. 4.15 shows the training result of two structures when they use the same input function and initial condition. It shows that the output of neural network controller is very close to the original controller, thus take it to replace the original one. Fig. 4.16 shows the system response with neural network controller when the  $y(0)$  is 0.5 and the input function  $u$  is  $\sin(2t)$ . It was observed that the controller control the system well even if the initial condition or the input function change.

In reality, the feedback system may contain some unexpected disturbances, so we add a sin function with high frequency 100 rad/sec and small amplitude 0.1 as the unexpected disturbance to the feedback system, shown as Fig. 4.17. Fig 4.18 shows the testing result of the feedback system with disturbance. It tells that the neural network controller performs similar to the original one, and can reject the influence of the disturbance.

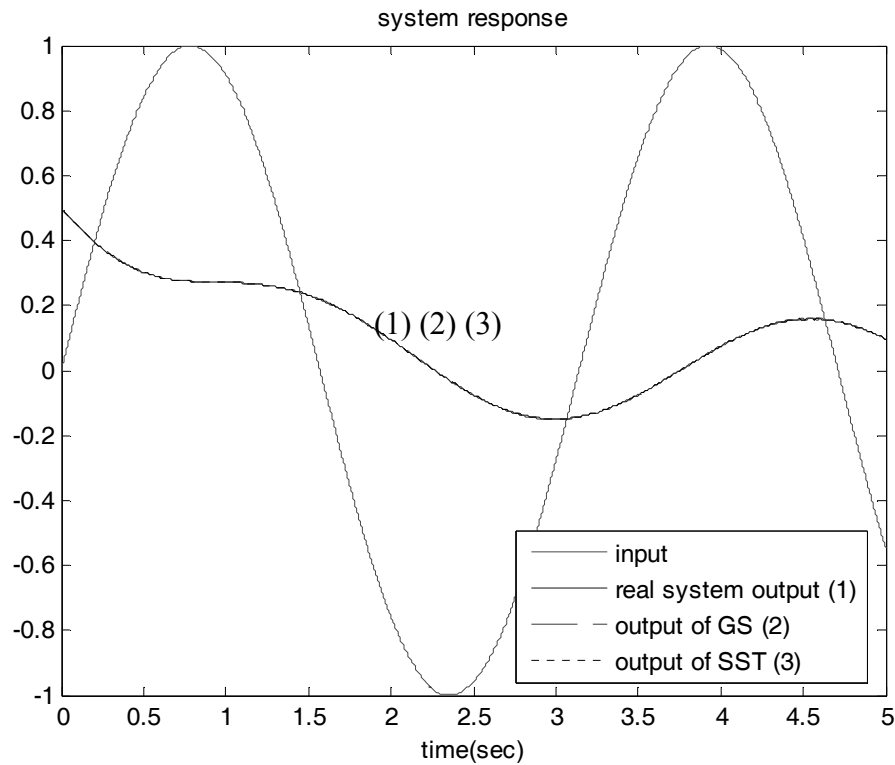


Figure 4.18 the testing result of the feedback system with unexpected disturbance

### 4.3.2 System with Second Order LTI Plant

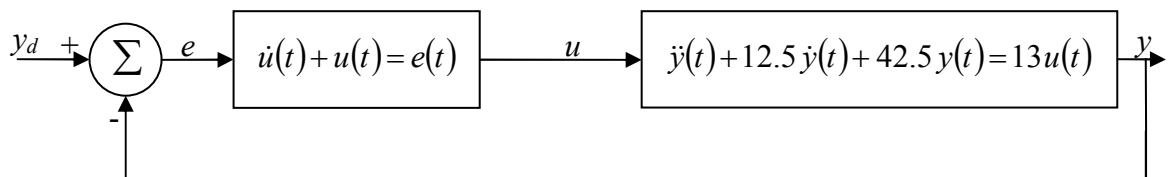


Figure 4.19 the feedback system

Similar to the last section, a second order LTI plant, described as

$$\ddot{y}(t) + 12.5\dot{y}(t) + 42.5y(t) = 13u(t) \quad (4.4)$$

in the feedback system, which is shown as Fig. 4.19. In the feedback system, the neural network is trained as a first order LTI system and replaces the original controller, which is designed as

$$\dot{u}(t) + u(t) = e(t). \quad (4.5)$$

Since the original controller is designed the same with (4.1), let the best learning result of the above simulations be the neural network controller here.

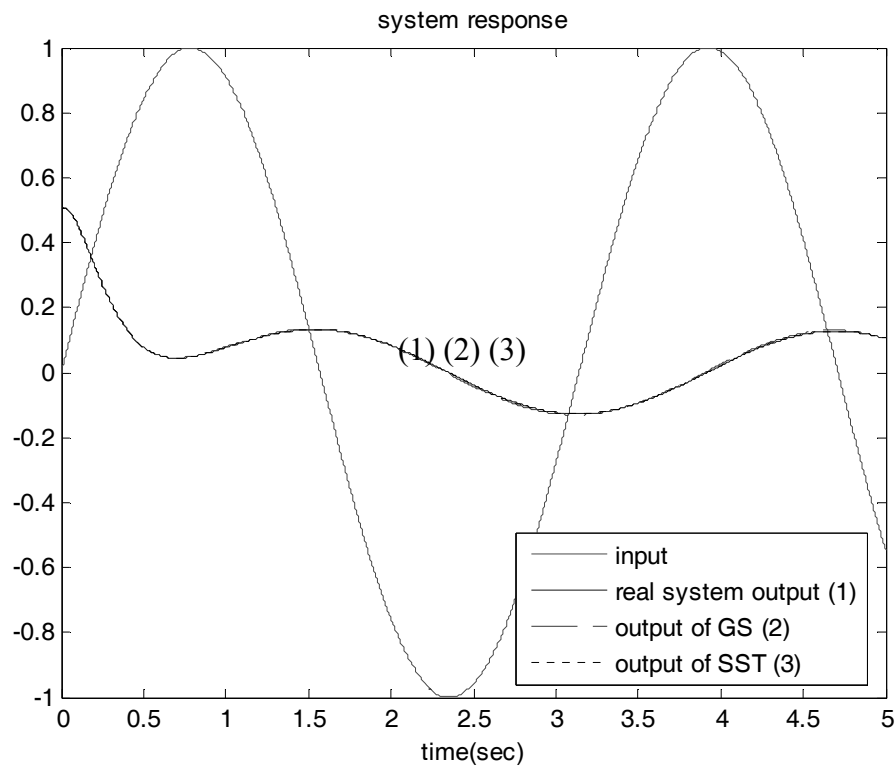


Figure 4.20 the testing result of the feedback system with a neural network controller

Fig. 4.20 shows the system response with neural network controller when  $y(0)$  is 0.5, and the input function  $u$  is  $\sin(2t)$ . It was observed that the controller control the second order system stable. Then, we also add an unexpected disturbance corresponding to the last section

to the feedback system. The system is shown as Fig. 4.21. Fig 4.22 shows the testing result of the feedback system with disturbance, and it shows that the neural network controller can reject the influence of the disturbance. Fig. 4.23(a) and Fig. 4.23(b) show the testing results of the plant input and plant output respectively while the unexpected disturbance is given as a random function. According to the above two simulations by different plants, it can be said that the neural network can learn well as a first order LTI controller by the evolution strategies. As long as the original controller can do, the neural network can also do.

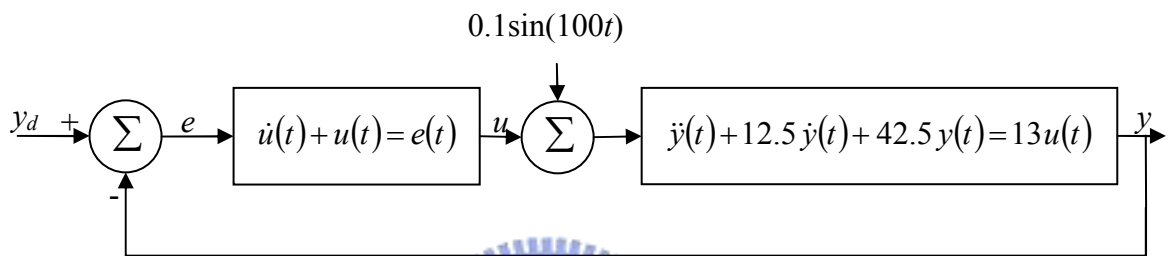


Figure 4.21 the feedback controller with disturbance

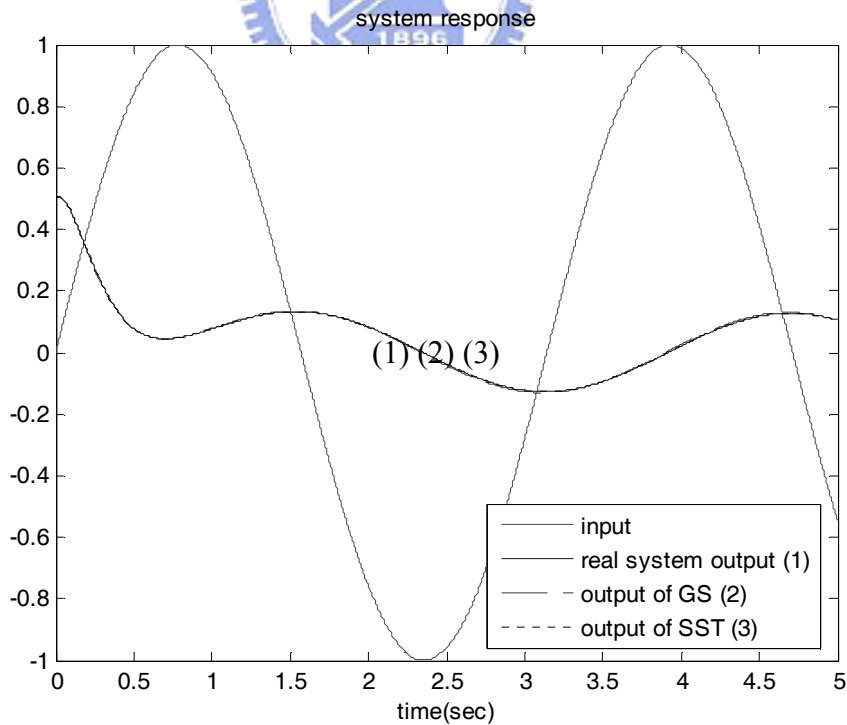
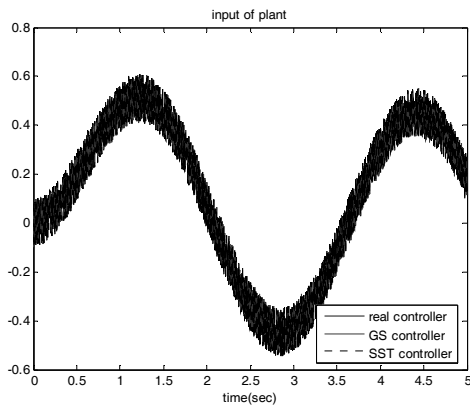
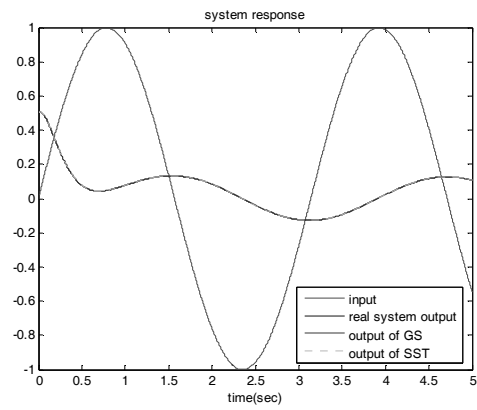


Figure 4.22 the testing result of the feedback system with unexpected disturbance



(a)



(b)

Figure 4.23 the testing result of the feedback system with unexpected random disturbance





# Chapter 5

## Conclusion

In this thesis, we provide two neural network structures, GS and SST, to learn a first order LTI system. These two structures are both simple structures using only one output delay. The GS can be used under a fixed sampling time, so it should be trained under sampling time small enough to increase its workable range. The SST contains the parameters of the first order difference equation in the network, so the parameters help SST adapt larger range of sampling time around that of the training data. Besides, we provide evolution strategies to help the neural network learn. It increases the searching space and tries to avoid local minima. For the purpose, the initial individual can be given by a set of known parameters from the first order difference equation. In the evolution strategies, the objective of the learning process is to find the individuals which lead to larger fitness. In the learning process, the change of the individuals of last step is important for creating the individuals of the next step. In the reproduction process, we use two ways to generate children such that the chance of finding a better individual is increased.

Originally, it is expected that the cost of the computation time is low because the neural network structures are simple, but not as expected due to the use of the evolution strategies. The evolution strategies are time consumption since they increase the searching space and require too many generations and steps for a better individual. It is noted that even though the number of generations and steps is sometimes large, the learning result is not guaranteed to be good.

No matter how long the learning costs, the neural network structures using the evolution

strategies can learn the first order LTI system, and behave as well as the objective system. No matter how the input functions and the initial conditions change, the output of the neural network is very similar to the objective system. It can replace the original first order LTI controller and engage as a feedback control. Even if the system has some unexpected disturbance, the neural network controller could also control the system successfully.

In the thesis, the fitness function is the sum of the errors between the outputs of the objective system and the neural network, so the neural network learns from a given system. Further, the fitness function can be changed as the sum of the error between the desired output and the outputs of the feedback system with neural network controller. Then, it is not necessary to know the controller designed by the conventional control theory and the neural network could learn to control the feedback system if the system could be controlled by a first order LTI system. In other words, the neural network could be trained to be a first order LTI controller by the evolution strategies even if the plant and the objective controller are unknown. Besides, the neural network structure can add some items for more order dynamic system, such as  $\Delta T^2$  and  $u[n] \times y[n]$ . These kinds of items may help increase the accuracy of the learning, and objective system is not restricted to a first order LTI system. In the future, both the above concepts, changing the fitness function and adding extra suitable items,  $\Delta T^2$  and  $u[n] \times y[n]$ , could be tried for improving the proposed method.

# References

- [1] L. A. Zadeh, "Knowledge representation in fuzzy logic," *IEEE Transaction on Knowledge Data Engineering*, vol. 1, pp. 89-100, 1989.
- [2] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley, 1989.
- [3] K. F. Man, K. S. Tang and S. Kwong, "Genetic algorithms: concepts and applications [in engineering design]," *IEEE Transactions on Industrial Electronics*, vol. 43, pp. 519-534, 1996.
- [4] M. Mitchell, *An Introduction to Genetic Algorithms*, Cambridge, MA, USA: MIT Press, 1996.
- [5] S. Haykin, *Neural Networks. A Comprehensive Foundation*, New York: Macmillan College Publishing, 1994.
- [6] C. G. Looney, *Pattern Recognition using Neural Networks: Theory and Algorithms for Engineers and Scientists*, New York, NY, USA: Oxford University Press, Inc, 1997.
- [7] A. Hiramatsu, "ATM communications network control by neural networks," *IEEE Transactions on Neural Networks*, vol. 1, pp. 122-130, 1990.
- [8] W. T. Miller III, R. S. Sutton and P. J. Werbos, *Neural Networks for Control*, Cambridge, MA, USA: MIT Press, 1990.
- [9] C. Breazeal, *Sociable machines: Expressive social exchange between humans and robots*, Ph.D. dissertation, Department Electrical Engineering and Computer Science Massachusetts Institute Technology, Cambridge, 2000.
- [10] H. Mobahi, *Building an Interactive Robot Face from Scratch*, project report, Azad University, South of Tehran campus Tehran, Iran, May 2003
- [11] T. Lee, *Structure Level Adaptation for Artificial Neural Networks*, Norwell, MA, USA: Kluwer Academic Publishers, 1991.

- [12] P. A. Mastorocostas and J. B. Theocharis, "A stable learning algorithm for block-diagonal recurrent neural networks: application to the analysis of lung sounds," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 36, pp. 242-254, 2006.
- [13] S. J. Han and S. B. Cho, "Evolutionary neural networks for anomaly detection based on the behavior of a program," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 36, pp. 559-570, 2006.
- [14] C. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*, Upper Saddle River, NJ, USA: Prentice-Hall, Inc, 1996.
- [15] M. Gori and A. Tesi, "On the problem of local minima in backpropagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 76-86, 1992.
- [16] 林俊良，智慧型控制分析與設計，全華科技圖書股份有限公司，2005。
- [17] G. Lightbody and G. W. Irwin, "Direct neural model reference adaptive control," *IEE Proceedings of Control Theory and Applications*, vol. 142, pp. 31-43, 1995.
- [18] D. Nguyen and B. Widrow, "The truck backer-upper: an example of self-learning in neural networks," *In Proceedings of the International Joint Conference on Neural Networks*, Vol. 2, pp. 357-363, 1989.
- [19] J. Zhang, S. S. Ge and T. H. Lee, "Output feedback control of a class of discrete MIMO nonlinear systems with triangular form inputs," *IEEE Transactions on Neural Networks*, vol. 16, pp. 1491-1503, 2005.
- [20] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270-280, 1989.
- [21] C. Chen, *System and Signal Analysis*, 2nd ed. Orlando, Florida, USA: Saunders College Publishing, 1994.
- [22] C. Darwin, *On the Origin of Species by Means of Natural Selection*, London: John Murray, 1859.
- [23] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing (Natural Computing Series)*, Springer, 2003.