ABSTRACT

Today automatic control systems have become an integrated part of our life. They appear in every things from simple electronic household products to air planes and spacecrafts. Automatic control systems can take highly different shapes but common to them all, is their function to manipulate a system so that it behaves in a desired fashion.

Control of nonlinear systems is a major application area for neural networks. The control design problem will be approached in two ways: direct design methods and indirect design methods, and the network must be trained as the controller according to some kind of relevant criterion.

In this thesis, nonlinear neuro-controller using neural network based actuator compensation scheme for nonlinear dynamical control system is presented.

The scheme that leads to stability, target following, tracking error and filtered error is proved . The tuning of artificial neural network weights for controller are derived and adjusted based on Lypanove function approach.

The verification of this scheme has been implemented using first order, 2-dymensional, nonlinear dynamical Pendulum problem and 1st order 3-dymensional nonlinear dynamical control system. The simulation can effectively compensate for the uncertain nonlinearity in the nonlinear uncertain dynamical control system.

Necessary mathematical concepts, comments, concluding remarks, future works, figures and graphers, have also been presented.

ACKNOWLEDGMENTS

Praise is to Allah, the cherisher and sustainer of the worlds. I would like to express my deep appreciation to my supervisor Dr. Radhi Ali Zboon for his patience and appreciable advice. I hope I will give him what he dissevers from gratitude.

Thanks are extended to the College of Science, Al-Nahrain University for giving me the chance to complete my postgraduate study.

Thanks are due to the head master of the department of mathematics and computer applications Dr. Akram M. Al-Abood, for his continuous help.

Also I am very thankful for all the staff members of the department of mathematics and computer applications.

Aleyaa

12/9/2006

APPENDIX (A)

Biological Neurons And Their Artificial Models

Ahuman brain consists of approximately 10^{11} computing elements called neurons. They communicate through a connection network of axons and synapses having adensity of approximately 10^{4} synapses perneuron. Our hypothesis regarding the modeling of natural nervous system is that neurons communicate with each other by means of electrical impulses [Arbib, 1987]. The neurons operate in a chemical environment that is even more important in terms of actual brain behavior.We thus can consider the brain to be adensely connected electrical switching network conditioned largely by the biochemical processes. The vast neural network has an claborate with very complex interconnections.

The input to the network is provided by sonsory receptors, receptors deliver stimulli both from within the body, as well as from sense organs when the stimuli originates in the external world. The stimuli are in the form of electrical impulses that convey the information into the network of neurons. As aresult of information processing in the

central nervous systems, the effectors are controlled and give human responses in the form of diverse actions. Wethus have a three stage system, consisting of receptors, neural network, and effecctors, in control of the organism and its actions.

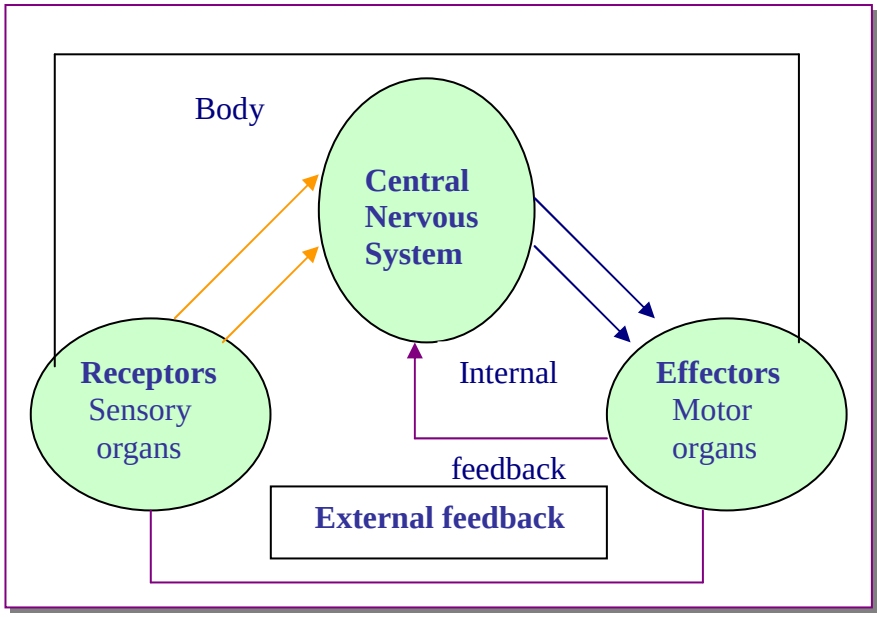


Figure (A.1)

Information flow in nervous system

A lucid, although rather approximate idea, about the information links the neurons system is shown above. As we can see from the (figure. a.1),the information is processed, evaluated and compared with the stored information in the central nervous system. When necessary command are generated there transmitted to the motor organs. Notic that motor organs are monitored to central nervous system by feedback link that verify their action. Both internal and external feedback control implementation of command as can be see the overall nervous system structure has many of the characteristics of a class loop control system.

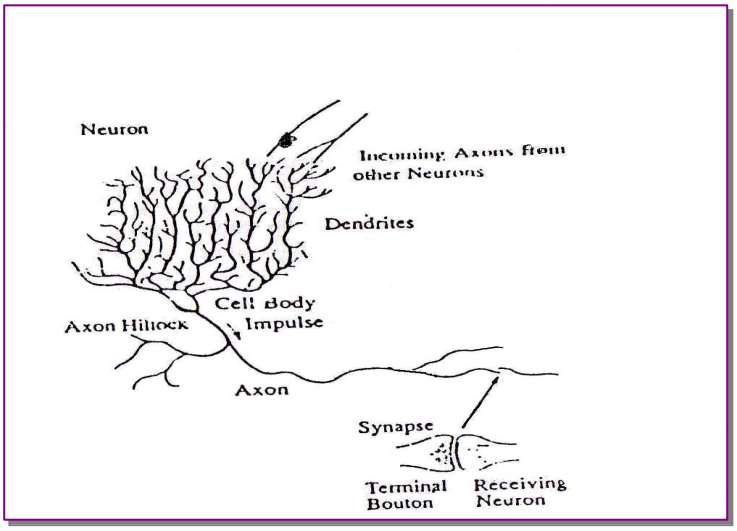
Living Neurons

To justify such a strong claim it is necessary to expand the argument a little. Living neurons are, in fact, composed of a cell body and numerous outgrowths. One of these, which may branch into several collaterals, is called the axon. It acts as the output line for the neurons. The other outgrowths are called the dendrites; they are often covered with

little 'spines', where the ends of the axons of other cells attach themselves.

The interior of the nerve cell is kept at a negative electric potential (usually about -60 mv) by means of active pumps in the cell wall which pump sodium ions outside and keep slightly fewer potassium ions inside. This electrical balance is espscially delicately assessed at the exit point of the axon.If the cell electrical potential becomes too positive, usually (+10 to +15 mv), then there will be a sudden reversal of the potential to about (+60 mv), and an almost as sudden return to the usual negative resting value ,all in about (2 to 3 ms).

his sequence of potential changes is called an action potential,which moves steadily down the axon and its branches (at about **1** to **10** ms^{-1}). It is action potential that is the signal sent from one cell to its neighbores. The generation of the signal by the neuron is achieved by the summation of the signals coming to the cell body from the dendrites, which themselves have been affected by action potentials coming to them from nearby cells. The strengths of the action potentials moving along the axons are all the same. It is by means of rescaling the effects of each action potential as it arrives at a synapse or junction from one cell to the next (by means of multiplication of the incoming activity of a nerve impulse by the appropriate connection weight mentioned earlier) that a differential effect is achieved for each cell on its neighbors.



Figuer (A.2)

Schematic diagram of a neuron and a sample

Of pulse train

APPENDIX (B)

Recurrent Network

A recurrent (feed back) network can be obtained from the feed forward network shown in figure (1.14) by connecting the neurons output to their inputs as illustrated in figure (B.1). Unlike the feed forward network, where there is an algebraic relationship between input and output, the recurrent architecture contains memory, i.e., it is a dynamic system. The recurrent network contains the feed forward network as a special case and obviously it therefore represents a more general class of architectures. The mathematical expression governing the network in figure (B.1) is given by

$$a(t/\theta) = g_i[\theta, p(t), t]$$
$$= F_i\left[\sum_{j=1}^{n_h} W_{ij} f_i(\bullet, t) + W_{i,0}\right] \qquad \text{(B.1)}$$
$$= F_i\left[\sum_{j=1}^{n_h} W_{ij} f_i\left(\sum_{l=1}^{n+m} w_{jl} f_l(t) + \sum_{l=1}^{n_h} w_{jl} f_l(\bullet, t-1) + w_{j0}\right) + W_{i0}\right]$$

Where F_i, f_i be the activation function of the hidden and input layer respectively, W_{ij}, w_{jl} be the weights of the hidden, input layer of the neural network respectively which be adapted later. w_{j0}, w_{i0} be the bias of the hidden, input layer respectively and $n_h, n+m, m$ be the number of layer on which the output is adapt on.

The recurrent can be implemented in many different ways. The example shown in figure (B.1) is just one example. If also the output of the output neurons are fed back, the network is often said to be **fully recurrent**.

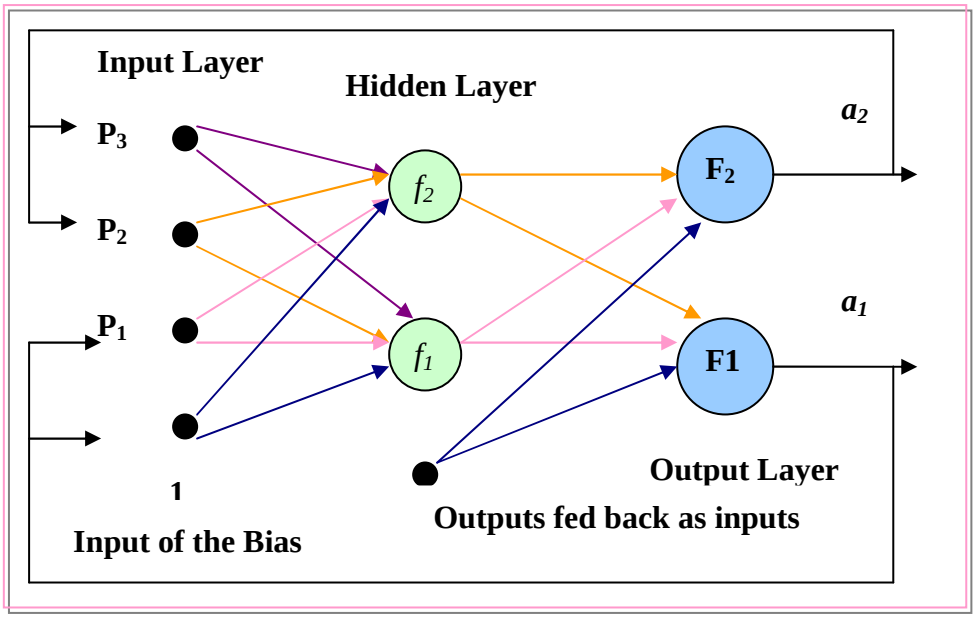


Figure (B.1)

A Simple Example of a Recurrent Network

Radial Basis Network

A network that has received a lot of attention recently in the field of neural networks is the radial basis function network. This can be described as

$$a = \sum_{i=1}^{nh} w_i \, g\left(\left\| p - c_i \right\|\right) \qquad \text{(B.2)}$$

with $p \in R^m$ the input vector and $a \in R$ the output (models with multiple outputs are also possible). The network consists of one hidden layer with **nh** hidden neurons. One of the basic differences with the multilayer perceptron is in the use of the activation function: in many cases one takes a Gaussian function for ***g***, which is radially symmetric with respect to the input argument. The output layer has output weights $w \in R^{n_h}$. The parameters for the hidden layer are the centers $c_i \in R^m$.

The networks (1.27) and (B.2) are feed forward neural networks, which means that there is a static nonlinear mapping from the input space to the output space: the layers are interconnected in a feed forward way to each other. In recurrent neural networks outputs of neurons are feed back to the network, resulting in a dynamical system. A simple example of a recurrent neural network is the Hopfield network, in discrete time looking as

$$p_{k+1} = \tanh\left(W p_k\right) \qquad \text{(B.3)}$$

with $p_k \in R^n$ the state vector and $W \in R^{n \times n}$ the synaptic matrix. The network consists of one layer of neurons.

APPENDIX (C)

Mathematical Foundations of Nuerocomputong

Neurocomputing makes use of parallel dynamical interactions of modifiable neuron-like elements. It is important to show, by mathematical treatments, the capabilities and limitations of information processing by various architectures of neural networks. This section, part tutorial and part review, tries to give mathematical foundations to neurocomputing. It considers the capabilities of transformations by layered networks, statistical neurodynamics, a general theory of neural learning, and self- organization of neural networks.

Statistical Analysis of Neural Transformation

A. One-layer Neural Network:

Let us consider one-layer network consisting of ***k*** of neural elements, which receive the same input signals $p = \lfloor p_1 \quad p_2 \quad \dots \quad p_m \rfloor$ in common, and emit respective outputs $a = \lfloor a_1 \quad a_2 \quad \dots \quad a_k \rfloor$. Let w_{ij} be the synaptic connection weight from the **ith** input component p_i to the **jth** neuron. The output a_j of the **jth** neuron is then written as

$$a_i = f\left(\sum_{i=1}^{m} w_{ij}\, p_i + b_i \right), \quad j = 1, 2, \dots, k \qquad \text{(C.1)}$$

Where f is a nonlinear output function and b_j is the threshold value (the bias).

Let us denote a bundle of output signals by a vector $a = \lfloor a_1 \quad a_2 \quad \dots \quad a_k \rfloor$. The network transforms a vector input signal p to a vector output signal a. We denote this transformation by

$$a = f_W\ p \qquad \text{(C.2)}$$

Where f_W is a nonlinear mapping defined by (C.1) and $w = w_{ij}$ is called connection matrix. A one-layer neural network thus defines a transformation or mapping from the input signal space $P = \{p\}$ to the output signal space $A = \{a\}$**.**

$$f_W : p \to a \qquad \text{(C.3)}$$

We use in this section a simple binary neuron model to demonstrate a mathematical method of approach, such that input and output signals p_j and a_i take on the binary values **+1** and **-1,** so that the function f is the sign function

$$\text{sgn}(u) = \begin{cases} 1, & u \geq 0 \\ -1, & u < 0 \end{cases} \qquad \text{(C.4)}$$

Moreover, we put $b_i = 0$ for the sake of simplicity so that

$$a = f_W\ p = \text{sgn}(w\ p) \qquad \text{(C.5)}$$

or

$$a_i = f_W\ p_i \qquad ,i = 1,2,\ldots,m \qquad \text{(C.6)}$$

There are a number of approaches to study characteristics of f_W . For example, one can define the capacity of the class of one-layer networks by the maximum number **m** of input-output pairs $(p_i, a_i) i = 1,2,\ldots,m$ as we defined in Eq., (C.6), such that for almost all such pairs there exists a network which realizes the input-output relation. It is known that the capacity is $m = 2n$ by a number of interesting but different methods [Cover, 1965],[Gardner, 1988],[Baum, 1989].

Remarks

In this section, we focus on the statistical method of approach. When a network is complex, the connection weights may be regarded as

if they are determined randomly. The statistical method is applicable to such networks. In the special case where the connection matrix $W = \{w_{ij}\}$ is determined randomly subject to a probability distribution, we can apply the statistical method to elucidate the characteristics of the mapping f_W. Obviously, the typical characteristics depend on the probability distribution of w_{ij}. We treat the two typical cases as examples:

1. Totally random networks in which all the components w_{ij} are independently and identically distributed.

2. Associative memory networks in which w_{ij} are not independent but are by a smaller number of random parameters.

Stability of Mapping Totally Random Networks

We first show properties of the mapping by a totally random network where w_{ij} are the realization of independent random variables subject to a **normal distribution N (ώ, σ^2_w),** with mean **ώ** and variance σ^2_w. Such a network is said to be totally random.

$$u_i = \sum_{j=1}^{m} w_{ij}\, p_j \qquad \text{(C.7)}$$

Which is the weighted sum of input stimuli, and the output is written as

$$a_i = \mathrm{sgn}(u_i) \qquad \text{(C.8)}$$

Since w_{ij} are the randomly determined, then given p then $u_i\,(i = 1,\ldots,k)$ and also randomly distributed. Moreover, they are independently and identically distributed. More precisely, u_i is a linear combination of w_{ij}, so that it is also normally distributed. Its mean is given by

$$\bar{u} = \mathrm{E}\left[\sum w_{ij}\; p_j\right] = \sum \bar{w}\; p_j = n\bar{w}A_p \qquad \text{(C.9)}$$

Where $\mathrm{E}[u]$ denotes the expectation of u and A_p is the mean activity of the input vector p defined by

$$A_p = \frac{1}{n}\sum p_j \tag{C.10}$$

The variance σ^2 of u_i

$$\sigma^2 = V\left[\sum_j w_{ij}\ p_j\right] = \sum_j p_j^2\, V\left[w_{ij}\right] = n\sigma_w^2$$

Where $V[u]$ denotes the variance of the u.

The probability P of $a_i = 1$, is given by

$$\mathrm{P} = prob\{a_i = 1\} prob\{u_i > 0\} = \Phi\left(\frac{\bar{u}}{\sigma}\right)$$

Where Φ is the error integral,

$$\Phi(u) = \int_{-\infty}^{u} \left(1/\sqrt{2\pi}\right)\exp\left\{-v^2/2\right\} dv \tag{C.11}$$

Since all the a_i are independent subject to the same probability distribution, the output activity

$$A_a = \frac{1}{k}\sum_{i=1}^{k} a_i$$

Converges in probability to

$$A_a = \mathrm{E}[a_i] = \Psi(\alpha A_p) \tag{C.12}$$

Where ***k*** is large, where we put

$$\alpha = \sqrt{n}\,\frac{\bar{w}}{\sigma_w} \tag{C.13}$$

And

$$\begin{aligned}\Psi(u) &= 2\Phi(u) - 1 \\ &= \int_{-u}^{u} \left(1/\sqrt{2\pi}\right)\exp\left\{-v^2/2\right\} dv\end{aligned} \tag{C.14}$$

When we partition the input and output signal space ***P*** and ***A***, according to activity, f_W maps input signals of activity $\boldsymbol{A_p}$ to output signals of activity $\boldsymbol{A_a}$ given by (C.12). This is a **macroscopic characteristic** of f_W. When $\overline{w} = 0$ we have α =0. So that the activity $\boldsymbol{A_a}$ is concentrated around **0.** We study more subtle **microscopic properties** of the mapping f_W. Let us assume that p is mapping to a

$$a = f_W\ p$$

It is then expected that signals p' belonging to a neighborhood of p are mapping to a neighborhood of a. To show this, we introduce the **normalized distance** between p and p' by

$$D_p(p, p') = \frac{1}{2n} \sum_{i=1}^{n} \left| p_i - p'_i \right| \qquad \text{(C.15)}$$

This is the **normalized Hamming distance**, satisfying $0 \le D_p \le 1$. The distance $D_p(p, p')$ between p and p' is defined similarly.

Let p' be a signal whose distance from p is $\boldsymbol{D_p}$. How far is the distance $\boldsymbol{D_a}$ between $a = f_W\ p$ and $p' = f_W\ p'$**.** See fig.(C.2) relation between D_p and D_a defines a stability or robustness of the mapping f_W, because when p' is regarded as a noisy version of p with noise rate $\boldsymbol{D_p}$, the noise rate of the output a' is given by $\boldsymbol{D_a}$. when $D_a < D_p$, the noise is reduced by the transformation, and if $D_p < D_a$, the noise is amplified.

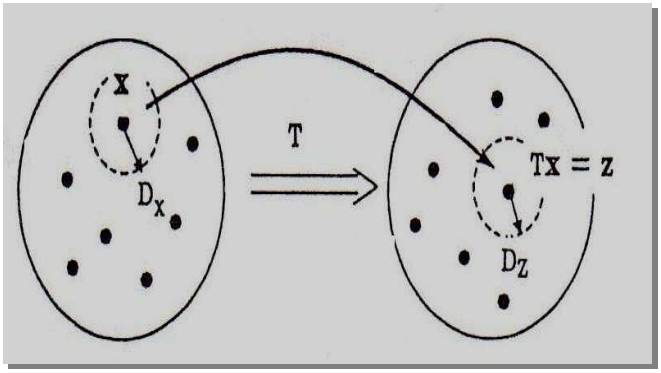


Figure (C.2)

Stability of transformation f_W

The next theorem shows that the noise is amplified by a totally random network [Amari, 1974]. This type of network enlarges small differences around p so that they are convenient for detecting differences among similar signals .

The following theorem holds for networks with $\overline{w} = 0$. Similar properties hold for any totally random networks.

Theorem 1

When the distance between p and p' is $D_p(p, p')$ **,** the distance $D_a(f_W\ p, f_W\ p')$ of their transforms is given by

$$D_a = \frac{2}{\pi} \sin^{-1} \sqrt{D_p} \qquad \text{(C.15)}$$

Proof: let us put

$$u_i = \frac{1}{\sqrt{n}} \sum w_{ij} p_j \qquad , v_i = \frac{1}{\sqrt{n}} \sum w_{ij} p'_j$$

When $u_i v_i > 0, a_j = a'_j$ and when $u_i v_i < 0, a_j = a'_j$. Therefore $\boldsymbol{D_a}$ is given by the ratio of the number of the components; satisfying $u_i v_i > 0$ to $\boldsymbol{k}$. Since (u_i, v_i) are a pair of normal random variables and are independent for different **i**, we have

$$D_a = prob\{u_i v_i < 0\}$$

When ***k*** The mean of $\boldsymbol{u_i}$ and $\boldsymbol{v_i}$ are zero, and the variances of $\boldsymbol{u_i}$ and $\boldsymbol{v_i}$ are $\sigma^2_{\boldsymbol{w}}$, their covariance is given by

$$\sigma_{uv}^2 = \mathrm{E}[u_i v_i] = \frac{1}{n}\sigma_w^2 \sum p_j p'_j = (1 - 2D_p)\sigma_w^2$$

Therefore,

$$D_a = \int_{uv<0} \frac{1}{2\pi} \exp\left\{-\frac{1}{2}\begin{pmatrix} u \\ v \end{pmatrix}' \sum{}^{-1} \begin{pmatrix} u \\ v \end{pmatrix}\right\} du\, dv,$$

Where

$$\sum = \begin{bmatrix} \sigma_w^2 & \sigma_{uv}^2 \\ \sigma_{uv}^2 & \sigma_w^2 \end{bmatrix}$$

This can easily be calculated, giving (C.15).

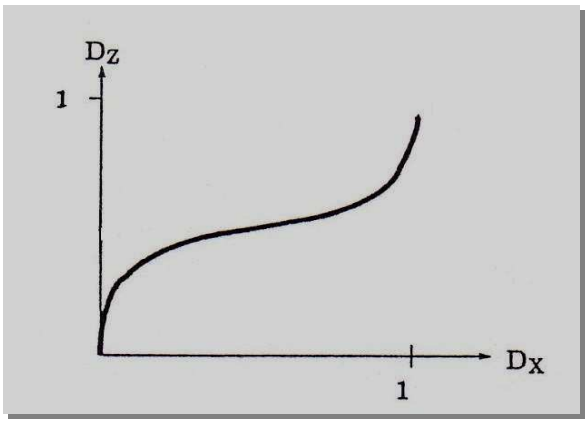


Figure (C.3)

Graph of C.15

Remark

Figure (C.3) gives the graph of relation (C.15), when $\boldsymbol{D_p}$ is small, we have approximately

$$D_a = \frac{2}{\pi}\sqrt{D_p}$$

The approximate derivative

$$\frac{d\,D_a}{d\,D_p} = \frac{1}{\pi}\frac{1}{\sqrt{D_p}}$$

Becomes infinitely large at $D_p = 0$. This implies that a small neighborhood of p is expanded and mapped to very large neighborhood of $a = f_W p$**.** Such a mapping is useful for recognizing differences among similar signals in a small neighborhood of p**,** because the differences are enlarged in the corresponding output signals. This is contrary the noise reduction property which reduces noise added to p in transforming it to a **.**

Statistical Neurodynamics

A. A fundamental problem of statistical neurodynamics:

We now treat a neural network with recurrent connection (Fig C.4). Let w_{ij} be the connection weight from the **jth** neuron to the **ith** neuron, and let $p_i(t)$ be the state or the output of the **ith** neuron at time ***t,*** taking values ***+1 or -1.***

When each neuron works synchronously at discrete time ***t =0, 1, 2…*** the behavior of the network is written as

$$p_i(t+1) = \mathrm{sgn}\left(\sum_{j=1}^{n} w_{ij}\ p_j(t) - h_i + \varepsilon_i \right) \qquad \text{(C.16)}$$

Where $\boldsymbol{h_i}$ is the threshold. ε_i is the weighted sum of stimuli coming to the **jth** neuron from the outside.

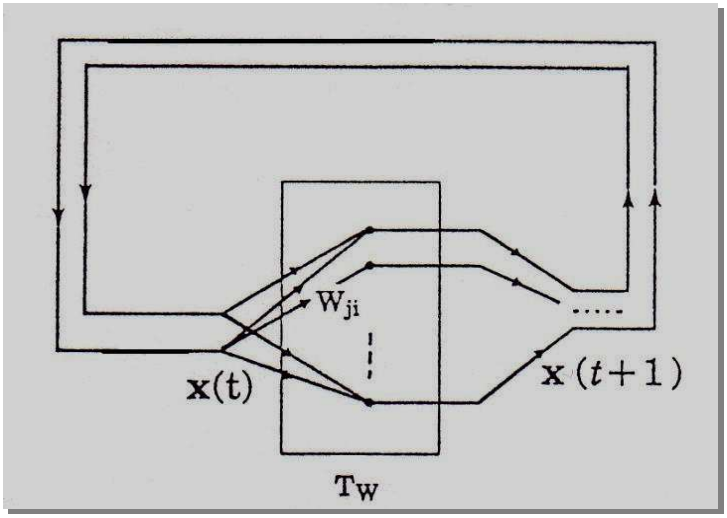


Figure (C.4)

Network with Recurrent Connections

For the sake of simplicity we let this ε_i be included in the term h_i, by putting $h_i - \varepsilon_i$ as the new h_i and neglect the term ε_i .

By using the nonlinear operator f_W

$$f_W\ p = \mathrm{sgn}(W\ p - h)$$

as before, eq.(C.16) is written as

$$p(t+1) = f_W\ p(t) \qquad \text{(C.17)}$$

The vector $p(t)$ is regarded as the state of the network at time t, and (C.17) is the state transition equation describing the dynamical behavior of the network.

The state space P consists of 2^n vectors p whose component are ±1 in the present case. The state transition f_W defines a mapping from p to itself, where $f_W\ p'$ is called the next state of p.

Remarks

1. The state transition graph is constructed in P by adding directed edges connecting tow nodes p and $f_W\ p$ in this order (Fig. C.5).

Each node (state) has one, and only one, edge starting from it and ending at its next state. The dynamic property of the net is fully represented by this graph.

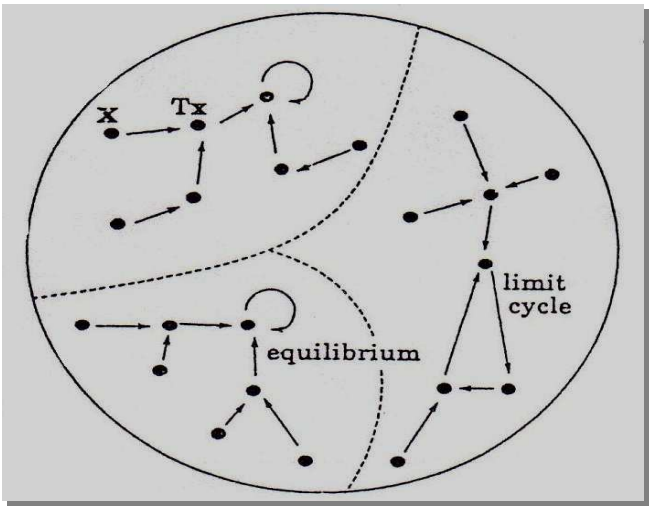


Figure (C.5)

.State transition graph of f_W.

- State p is said to be an equilibrium or a fixed point of P, when

$$p = f_W\ p$$

Holds, when and only when p has a self-closed edge, i.e. an edge starting p from and entering in p , it is a fixed point

- A sequence $\{p_1 \quad p_2 \quad \dots \quad p_k\}$ of nodes is said to be a cycle of period ***k,*** when

$$p_{t+1} = f_W\ p_t, \quad t = 1,\dots,k-1$$
$$p_1 = f_W\ p_k$$

Hold and all $\boldsymbol{x_t}$ are different. This is represented by a primitive loop of length ***k*** in the graph.

2. It is not easy to analyze the behavior of the dynamics ( ) of a nonlinear network. When the connection weight $\boldsymbol{w_{ij}}$ are randomly determined subject some probability distribution.

● There exist common dynamical properties that are shared by almost all randomly generated networks by the same probability law. Obviously such properties depend on the probability distribution.

B. Statistical neyrodynamics studies such properties by using macroscopic state variables:

A macroscopic state variable is a function of the (microscopic) state p summarizing some average features of the state. A simple example is the activity level.

$$\mathrm{A}(p)=\frac{1}{n}\sum_{i=1}^{n} p_i \qquad \text{(C.18)}$$

Of state p which shows the ratio of the excited neurons. The activity level $\mathbf{A}_t$ at time t is written as

$$\mathrm{A}_t = \mathrm{A}\{p(t)\} \qquad \text{(C.19)}$$

And it is expected that dynamical equation of the type

$$\mathrm{A}_{t+1} = F(\mathrm{A}_t) \qquad \text{(C.20)}$$

Holds for almost all randomly generated networks with a desired accuracy as n tends to infinity. Such a quantity is called ***a macroscopic state variable.***

It is not necessarily a scalar but may be a vector quantity. In order to elucidate common microscopic characteristics of the state transition graph, it is useful to define a macroscopic variable, which is a function of two or more microscopic states. For example, we may use the distance $D(p,y)$ between two states as a macroscopic variable and put

$$D_t = D\{p(t), y(t)\} \qquad \text{(C.21)}$$

Where $p(t) = f_w^t\, p(0)$ *and* $y(t) = f_w^t\, y(0)$ are the state transition sequences starting at $p(0)$ *and* $y(0)$ respectively. If we have such a dynamical relation as

$$D_{t+1} = G(D_t) \tag{C.22}$$

We can study how an initial difference in the state develops in the course of the state transition dynamics.

C. Macroscopic Dynamics of Activity:

When w_{ij} are independent subject to the same distribution of the average w^* and variable σ_w^2**,** say the normal distribution $N(h^*, \sigma_h^2)$ the dynamical equation of the activity is given by

$$\mathrm{A}_{t+1} = \psi(W^*\, \mathrm{A}_t + H^*) \tag{C.23}$$

Where

$$W^* = \frac{w^*}{\sigma}, \quad H^* = \frac{h^*}{\sigma}, \quad \sigma^2 = \sigma_w^2 + \sigma_{h^*}^2 \tag{C.24}$$

Totally random networks are classified into the three categories depending on the behaviors of the macroscopic equation or the parameters W^* *and* H^* [Amari, 1971].

1. Monostable, converging to the unique equilibrium macrostate A^* from whatever initial state it starts.

2. Bistable, converging to one of the two stable equilibrium macrostates depending on the initial state.

3. Oscillatory with periods 2. (Figure (C.5)) shows how the dynamic behavior depends on the macroparameters W^* *and* H^*. This is the catastrophe curve of the macroscopic dynamics. It should be noted that h_j *or* H^* can be controlled by stimulation from the outside.

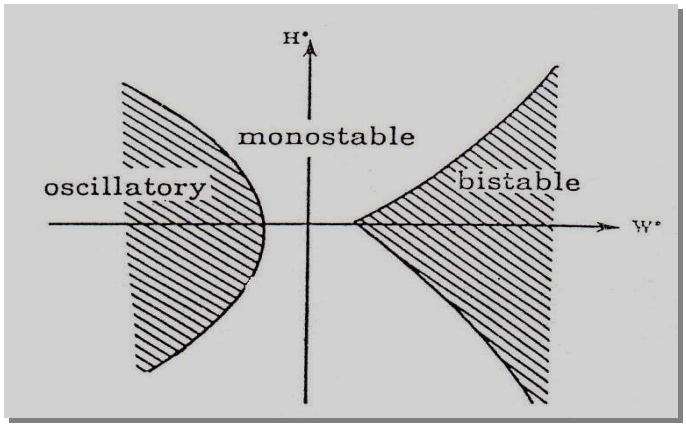


Figure C.5

Catastrophe of macro dynamics

Remarks

To study the dynamical behavior of especially in the case consisting of two different types of neurons:

Consider a random network consisting of several different types of neurons. Let $w_{ij}^{\alpha\beta}$ be the connection weight from the **jth** neuron of type β to the neuron of type α. It is a realization subject to the normal distribution with mean $w_{\alpha\beta}^{*}$ and variance $\sigma_{h\alpha}^{2}$. The threshold h_i^{α} of the **ith** neuron of type α is also assumed to be the normal distribution with mean h_{α}^{*} and variance $\sigma_{h\alpha}^{2}$. Let A_t^{α} be the activity of the neurons of type α at time ***t***,

$$\mathrm{A}_t^{\alpha} = \frac{1}{n_{\alpha}} \sum_i p_i^{\alpha}(t)$$

Where n_{α} is the number of the neurons of type α and $p_i^{\alpha}(t)$ is the state at time ***t*** of the **ith** neuron of type α. It is easy to show that the vector

$$\mathrm{A} = \left(\mathrm{A}^{\alpha}\right)$$

Is a macroscopic state satisfying the postulate [Cover, 1965], the macroscopic dynamics is given by

$$\mathrm{A}_{t+1}^{\alpha} = \psi\left(\sum_{\beta} W_{\alpha\beta}^{*} \mathrm{A}_{t}^{\beta} + H_{\alpha}^{*} \right) \qquad \text{(C.25)}$$

Where the macroparameters $W_{\alpha\beta}^{*}$ *and* H_{α}^{*} are given by

$$W_{\alpha\beta}^{*} = \frac{1}{\sigma_{\alpha}} w_{\alpha\beta}^{*}, \quad H_{\alpha}^{*} = \frac{1}{\sigma_{\alpha}} h_{\alpha}^{*}$$

$$\sigma_{\alpha}^{*} = \sum_{\beta} \sigma_{\alpha\beta}^{2} + \sigma_{h\alpha^{*}}^{2}$$

[Amari, 1971**]** studied the dynamical behavior of (C.25), especially in the case consisting of two different types of neurons. There exist at most **9** equilibrium macrostates in such a network [Harth, 1970]. When type **1** neurons are excitatory and type **2** neurons are inhibitory, it was shown that there exists a stable oscillatory solution with a long period. Similar results were found also by **Wilson** and **Cowan [**Wilson, 1972**].** Not only the oscillatory behavior but also its period can be controlled by stimuli from the outside, so that such a network is convenient for modeling temporal behaviors, [Amari, 1972**]** also studied similar behaviors in random nets of continuous-time analog neurons described by a set of differential equations.

D. Characteristics of Microstate Transition in Totally Random Networks:

We now study typical characteristics of the state transition graph of a totally random asymmetric network, where w_{ij} *and* w_{ji} are independent. On the other hand, it is also important to study characteristics of a network of symmetric connections, in which the symmetry condition $w_{ij} = w_{ji}$ is imposed. This is because a symmetric network can be used to solve the optimization problem under constraints [Hopfield, 1985], [Rumelhart, 1986], we consider a random symmetric network, too, where all the w_{ij} are independently, identically and normally distributed under

the symmetric condition $w_{ij} = w_{ji}$. As will be shown there are big differences in the characteristics of symmetric and a symmetric random networks.

1. Number of stable states: a state p is said to be equilibrium or stable when it's next state $f_w\, p$ is equal to p itself**.**

The expected number of states can be calculated by the statistical neurodynamical method.

Theorem The expected number of stable is equal to **1** for a symmetric random net, and is equal to $\mathbf{e^{0.199n}}$ for a symmetric random net.

Proof:

The proof for the asymmetric is easy [Amari, 1974]. The proof for the symmetric case was given by **Tanaka** and **Edwards [**Tanaka, 1980**]** in the connection with SK-model of spin glass. Here, we give a simple proof. We first calculate the probability that $p = (1,1,\ldots,1)$ is stable state. For the above ***x***, we put

$$u_i = \sum w_{ij} \cdot p_j = \sum_{j=1}^{n} wij$$

The state p is stable, if and only if $u_i > 0$ for all i. Therefore, the probability P that $f_w\, p = p$ is written as

$$\mathrm{P} = \mathrm{P}rob\{u_1 > 0, \ldots, u_n > 0\}$$

In the symmetric case all the u_i are independently and normally distributed with mean 0. Therefore, because of $\mathrm{P}rob\{u_1 > 0\} = 0.5$

$$\mathrm{P}\ asym = \prod_{i=1}^{n} prob\{u_i > 0\} = 2^{-n}$$

In the symmetric case, u_i are normally distributed with mean 0 and variance

$$\sigma^2 = V\left[\sum w_{ij}\right] = n\,\sigma_w^2$$

However, u_i *and* u_j are not independent because of $w_{ij} = w_{ji}$ and their covariance is given by

$$Cov\left[u_i, u_j\right] = Cov\left[w_{ij}, w_{ji}\right] = \sigma_w^2$$

These correlated u_i can be represented by using mutually independent normal random variables s_i *and* r subject to $N(0,1)$ as

$$u_i = \sigma_w\left(\sqrt{n-1}\cdot s_i - r\right)$$

The probability then because

$$\begin{aligned} \text{P } sym &= prob\left\{u_1 > 0, \ldots, u_n > 0\right\} \\ &= prob\left\{ s_1 > \frac{r}{\sqrt{n-1}}, \ldots, s_n > \frac{r}{\sqrt{n-1}}\right\} \end{aligned}$$

In order to calculate this, we first fix r, and calculate the probability. We then take expectation with respect to r **.** When r is fixed, the events

$$s_i > r/\sqrt{n-1}$$

are independent in the sense of the conditional probability. Therefore, because of $\text{P}rob\{s > c\} = 1 - \Phi(c)$.

$$\begin{aligned} p\ sym &= \text{E}_r\left[\prod_{i=1}^{n}\left(1 - \Phi\left(r/\sqrt{n-1}\right)\right)\right] \\ &= \frac{1}{\sqrt{2\cdot\pi}}\int \exp\left\{-\frac{r^2}{2}\right\} + n\cdot\log\left\{1 - \Phi\left(\frac{r}{\sqrt{n-1}}\right)\right\} dr \\ &= \sqrt{\frac{n-1}{2\pi}}\int \exp\left[-n\left\{\left(\frac{1}{2} + \frac{1}{2n}\right) y^2 - \log\left\{1 - \Phi(y)\right\}\right\}\right] dy \end{aligned}$$

By using the saddle-point approximation, we have

$$p_{sym} = \exp\left\{-n\left[\frac{1}{2}\,y_0^2 - \log\left\{1 - \Phi(y_0)\right\}\right]\right\},$$

where y_0 is the value of minimizing $(1/2)\,y^2 - \log\left\{1 - \Phi(y)\right\}$ Numerical calculation give

$$p\ sym = e^{-0.494n}$$

it is easy to show that $\mathrm{P}\,asym$ and $\mathrm{P}\,sym$ do not depend on specific p but are the same foe any p. since there are 2^n times the probability.

2. Stability of State Transition: for two states p_1 *and* p_2 whose distance is **D**, we search for the distance **D'** between their next states $f_w\,p_1$ *and* $f_w\,p_2$ [Amari, 1974].

Theorem When the distance between p_1 *and* p_2 is **D,** the distance **D'** between $f_w\,p_1$ *and* $f_w\,p_2$ is given by

$$D' = \frac{2}{\pi}\sin^{-1}\sqrt{D} \qquad \text{(C.26)}$$

in both cases of the asymmetric and symmetric random connections.

3. Potential function: In the case of symmetric connections, a potential function monotonically decreases as the state evolves by the network dynamics. This is a characteristic feature of asymmetric connection network. Let

$$\mathrm{E}(p) = -\frac{1}{2}\sum w_{ij}\,p_i\,p_j - \sum h_i\,p_i \qquad \text{(C.27)}$$

Where we assume $w_{ij} = 0, \forall i$, when each neuron changes its state (output) one by one in a non synchronized manner on the sign of the weighted sum of input stimuli, it is easy to show that $\mathrm{E}(p_t)$ is monotonically non increasing as the state transition takes place [Hopfield, 1982] [Hopfield, 1984]. This implies that the state of a network converges to one of the local minima of the potential function $\mathrm{E}(p)$.

This also proves that there is no oscillatory behavior in such a network see also [Cohen, 1983] [Hopfield, 1985], for the Lyapunov or Potential function (in the continuous-time case).

Back propagation of Recurrent Networks

The back propagation has been generalized to be applicable with recurrent connections by many researchers [Williams, 1989] [Doya, 1989]. We give a simple but example.

Let ***N*** be a network with n neurons, some of which are output neurons and some of which are input neurons. Let $z_i(t)$ be the output of the **ith** neuron at time ***t,*** which is determined by

$$z_i(t) = \Phi[u_i(t)] \tag{C.28}$$

$$u_i(t) = \sum w_{ij} z_j(t-1) + \sum s_{ik} x_k(t-1) \tag{C.29}$$

Here w_{ij} is the weight of the recurrent connection from the **jth** unit to the **ith** unit, and s_{ik} is the connection weight from the **kth** input x_k to the **ith** unit. When the **ith** unit is not an input neuron $s_{ik} = 0$. The input is a time sequence $\{x(t)\}$, $t = 1, 2, \ldots$, and the network is in the quiescent state at time 0. For each input time sequence $x(t)$ a sequence $y(t)$ of the desired outputs is given at ***t*** = 1, 2…, to the output neurons.

The parameters are (w_{ij}, s_{ik}) some of which may be fixed to **0** or to prescribed values. We denote by ***S*** only modifiable parameters. The squared error loss at time ***t*** is given by

$$\ddot{I}(x,t;S) = \frac{1}{2} \sum_{k \in O} | z_k(t) - y_k(t) |^2 \tag{C.30}$$

Where the summation is taken over the set ***O*** of the output neurons.

The connection weights are changed in the learning phase by

$$w_{ij}(t+1) = w_{ij}(t) - c\, \partial \ddot{I} / \partial w_{ij} \tag{C.31}$$

$$s_{ij}(t+1) = s_{ij}(t) - c\, \partial \ddot{I} / \partial s_{ij} \tag{C.32}$$

However, $\partial \ddot{I} / \partial w_{ij}$ *and* $\partial \ddot{I} / \partial s_{ij}$ are not simple, because of the recurrent connections. We have the following theorem.

Theorem The learning rule of a recurrent network is given by

$$w_{ij}(t+1) = w_{ij}(t) - c \sum_{k \in O} e_k \; r_{kij}(t) \tag{C.33}$$

$$s_{ij}(t+1) = s_{ij}(t) - c \sum_{k \in O} e_k \; q_{kij}(t) \tag{C.34}$$

Where

$$e_k = \sum \left| \; z_k(t) - y_k(t) \right|^2 \tag{C.35}$$

And r_{kij}, q_{kij} are calculated recurrently by

$$r_{kij} = \Phi'(u_k(t))\, z_j(t-1)\, \delta_{ki} + \sum_m w_{km} \; r_{mij}(t-1) \tag{C.36}$$

$$q_{kij} = \Phi'(u_k(t))\, x_j(t-1)\, \delta_{ki} + \sum_m w_{km} \; q_{mij}(t-1) \tag{C.37}$$

δ_{ki} being the ***Kronecker delta.***

Remark

When the dynamics of the neural network converges to $z = f(z;S)$ and if we want the final output z to be equal to the desired $y(x)$, we may use the equilibrium solution of (C.28) and (C.29) by solving the simultaneous equations. This method is proposed by **[**Pineda, 1987**]**.

1

1.1 History of Artificial Neural Systems [Zurada,1996]

Artificial neural systems development has an interesting history. Since it is not possible to cover this history in depth in a short introductory section, the historical summary below is not exhaustive; some milestones are omitted and some are mentioned only briefly.

The year **1943** is often considered as the initial year in the development of artificial neural systems, when **McCulloch and Pitts** in **(1943)** outlined the first formal model of an elementary computing neuron, **[**McCulloch, 1943]. The model included all necessary elements to perform logical operations, and thus it can operate as an arithmetic-logic computing element. The implementation of its compact electronic model, however, was not technologically feasible during the bulky vacuum tubes. The formal neuron model was widely adopted for the vacuum tube computing hardware description, and the model never became technically significant. However, the McCulloch and Pitts neuron model laid the groundwork for future developments. Influential researchers of that time suggested that research in design of brain-like processing might be interesting. To quote **John Von Neumann's (1958)** observations on the "**brain language**", **[**Von, 1958**]**.

Donald Hebb in **(1949)** first proposed a learning scheme for updating neuron's connections that we now refer to as the ***Hibbian* leaning rule** [Hebb, 1949] . He started that the information can be stored in connections, and postulated the learning technique that had a profound

impact developments in this field. Hebb's learning rule. Made primary contributions to neural networks theory.

During the **1950,** the first **neurocomputers** were built and tested **Minsky (1954)** [Minsky, 1954]. They adapted connections automatically.

During this stage, the neuron-like element called a ***perceptron*** was invented by **Frank Rosenblatt** in **1958**, it was a trainable machine of learning to classify certain patterns by modifying connections to the threshold elements. The idea caught the imagination of engineers and scientists and laid the groundwork for the basic machine learning algorithms that we still use today [Rosenblatt, 1958]. In the early **1960** a device called **ADALINE** (for **ADAptive LINEar** combiner) was introduced, and a new powerful learning rule called the ***Widrow-Hoff* learning rule** was developed by **Bernard Widrow and Hoff** in **(1960)** [Widrow, 1960]. **1962** the rule minimized the summed square error during training involving pattern classification. Fatly application of **ADALINE** and its extensions to **MADALINE** (for many ADALINES) include pattern recognition. The monograph on learning machines by **Nile Nilsson** in **(1965)** clearly summarized many of the developments of that time, [Nilsson, 1965].

Despite the successes and enthusiasm of the early and **mid-1960**, the existing machine learning theorems of that time were too weak to support more complex computational problems. Although the bottlenecks were exactly identified in Nilsson's work and the neural network architectures called **layered networks** were also known. Neural network research entered into the stagnation phase.

Another reason that contributed to this research slowdown at that time was relatively modest computational resources available then. The final episode of this area was the publication of a book by **Marvin Minsky and Papert** in **(1969)** that gave more doubt as to the layered

learning networks' potential, **[**Minsky, 1969**]**. The stated limitations of the **perceptron-class** networks were made public; however, the challenge was not answered until the **mid-1980**. The discovery of successful extensions of neural network knowledge had to wait until **1986**. Meanwhile, the mainstream of research activity in the neural network field, called at that time ***cybernetic*** had sharply decreased. The artificial intelligence area emerged as dominant and promising research field, which took over, among others, many of the tasks that neural networks of that day could not solve.

During the period from **1965 to 1984**, further pioneering work was accomplished by a handful of researchers. The study of learning in networks of threshold elements and of the mathematical theory of neural networks was pursued by **Sun-Ichi Amari (1972, 1977) [**Amari, 1972] [Amari, 1977]. Also in Japan, Kunihiko Fuukushima developed a class of neural network architectures known as ***neocognitrons*** Fukushima and Miyaka (1980) [Fukushima, 1980]. The necognitron is a model for visual pattern recognition and concerned with biological plausibility. The network emulates the retinal image and processes them using two-dimensional layers of neurons.

Associative memory research has been pursued by, among others, **Tuevo Kohonen** in Finland **(1977, 1982, 1984, 1988)** [Kohonen, 1977,1982,1984 and 1988] and **James A.Anderson** [Anderson, 1977] Unsupervised learning were developed for feature mapping into regular arrays of neurons [Kohonen, 1982]. **Stephen Grossberg and Gail Carpenter** have introduced a number of neural architectures and theories and developed the theory of adaptive resonance networks [Grossberg, 1977, and 1982].

During the period from 1982 until 1986, several seminal publications were published that significantly furthered the potential of

neural networks. The aera of renaissance started with **John Hopfield (1982, 1984)** [Hopfield, 1984] introducing ***a recurrent neural network*** architecture for associative memories. His papers formulated computational properties of a fully connected network of units.

Another revitalization of the field came from the publication in **1986** of two volumes on parallel distributed processing, edited by **James McClelland and David Rumelhart (1986)** [McClelland, 1986] **[**Rumelhart, 1986]. The new learning rules and other concepts introduced in this work have removed one of the most essential network training barriers that grounded the mainstream efforts of the mid-1960. the publication by McClelland and Rumelhart opened a new era for the once-underestimated computing potential of layered networks. The function approximator, EEG spike detector and autonomous driver discussed in the previous section provide examples facilitated by the new learning rules.

The field of function approximation has led to the important **'universal approximation theorem' [**Hornik, 1989]**.** This theorem states that (any suitably smooth function can be approximated arbitrarily closely by a neural network with only one hidden layer). The number of nodes required for such an approximation would be expected to increase without bound as the approximation was increasingly better. The result is of the utmost importance to those who wish to apply neural networks to a particular problem (it states that a suitable network can always be found). This is also true for trajectories of patterns [Funahashi, 1993]**.**

There is a similar, but more extended result, for the learning of conditional probability distribution [Allen, 1994**],** where now the universal network has to have at least two layers to be able to have a smooth limit when the stochastic series being modeled becomes noise-free.

Although the **mathematical framework** for the new training scheme of layered networks (**Backprpagation algorithm**) was discovered in **1974** by **Paul Werbos [**Werbos, 1974] (and studied by **Parker (1985)** and **LeCun (1985))**, it went largely unnoticed at that time [Werbos, 1974]. According to the most recent statement [Dreyfus, 1990], the first authors of the **optimization approach for multilayer feedforward systems** were **Bryson** [Bryson, 1969] and **Kelley** who obtained a gradient solution for multistage network training. In **1962**, **Dreyfus** used a simple, new recursive derivation based on the chain-rule of differentiation to prove the Bryson-Kelley results and dealt explicitly with the optimal control problem in its discrete-stage form [Dreyfus, 1962]. Their work, however, has not been carried to maturity and adopted for neural network learning algorithms.

Beginning in **1986-87**, many new neural networks research programs were initiated. The intensity of research in the **neurocomputing** discipline can be measured by a quickly growing number of conference and journals devoted to the field. In addition to many edited volumes that contain collections of papers, several books have already appeared. The list of application that can be solved by neural networks has expanded from small test-size examples to large practical tasks. Very-large- scale integrated neural network chips have been fabricated. At the time of this writing, educational offering have been established to explore the **artificial neural systems science**. Although **neurocomputing** has an interesting history, the field is still in its early stages of development. In 1988 Austin discussed Rapid learning with a hybrid neural network, Bolt, Austin study the assessing the reliability of artificial neural networks, Morgan study safety critical neural networks in 1995, Yan, Austin discussed the mathematical foundations of statistical parallelism in 1997, Hodge, Austin study the

evaluation of standard retrieval algorithms and a binary neural approach in 2000, in 2003 Hodge, Austin study the evaluation of standard spell checking algorithm and a binary neural network. in 2004 Pears, Crook-Dawkins study the robust dependable model-based visual location for mobile robots.

For more Details on Biological Neurons: see appendix **A**.

1.2 Some Neural Networks Concepts

The following are some Artificial Neural Network definitions that are needed in the following section [Zurada, 1996] [Paual, 1997] :

1. An artificial neural network*:* is a computational structure that is inspired by observed processes in natural networks of biological neurons in the brain. It consists of simple computational units, called ***neurons,*** which are highly interconnected. Each interconnection has a strength that is expressed by a number referred to as a ***weight****.*

2. Neuron: is the basic processing element of a neural network. Includes weights and bias, a summing function and an output transfer function.

3. Bias: is a neuron parameter that is summed with the neuron's weighted inputs and passed through the neuron's transfer function to generate the neuron's output.

4. Bias vector: a column vector of bias values for a layer of neurons.

5. Connection: a one-way link between neurons in a network.

6. Weight matrix: is matrix connection strengths from a layer's inputs to its neurons. The element w_{ij} of a weight matrix W refers to the connection strength from input j to neuron i.

7. Layer: is a group of neurons having connections to the same inputs and sending outputs to the same destinations.

8. Input layer: a layer of neurons receiving inputs directly from outside the network.

9. Layer weight: the weights connecting layers to other layers.

10. Hidden layer: a layer of network that is not connected to the network output.

11. Output layer: a layer whose output is passed to the world outside the network.

12. Input space: the range of all possible input vectors.

13. Input vector: a vector presented to the network.

14. Input weights: the weights connecting network inputs to layers.

15. Input weight vector: the row vector of weighs going to a neuron.

16. Weight input vector: the result of applying a weight to a layer's input, whether it is a network input or the output of another layer.

17. Output vector: the output of a neural network. Each element of the output vector is the output of the neuron.

18. Output weight vector: the column vector of weights coming from a neuron or input.

19. Target vector: the desired output vector for a given input vector.

20. Error vector: the difference between a network's output vector in response to an input vector and an associated target output vector.

21. Transfer function: the function that maps a neurons (or layers) net output **n** to its actual output.

22. Architecture: a description of the number of the layers in a neural network, each layer's transfer function, the number of neurons per layer, and the connections between layers.

23. Learning: the process by which weights and biases are adjusted to achieve some desired network behavior.

24. Training: a procedure whereby a network is adjusted to do a particular job.

25. Learning rate: a training parameter that controls the size of weight and bias changes during learning.

26. Learning rules: methods of deriving the next changes that might be made in a network OR a procedure for modifying the weights and biases of a network.

27. Update: make a change in weights and biases. The update can occur after presentation of a single input vector or after accumulating changes over several input vectors.

28. Supervised Learning: a learning process in which changes in a network's weights and biases are due to the intervention of any external teacher. The teacher typically provides output targets.

29. Unsupervised Learning: a learning process in which changes in a network's weights and biases are not due to the intervention of any external teacher. Commonly changes are a function of the current network input vectors, output vectors, and previous weights and biases.

1.3 Why Neural Network ?

Why have neural networks attracted particular attention compared with alternative techniques? For a given application it is of course difficult to say that one identification technique will outperform another before they have both been evaluated. Nevertheless, it is desirable to consider only one technique for all applications rather than having to evaluate several candidates on each new application. Partly because it simplifies the modeling process itself, and also because it will enable implementation of generic tools for control system design.

When searching for a single technique that in most cases of practical interest performs reasonable well, certain types of neural

network appear to be an excellent choice. In particular the multilayer perceptron network has gained an immense popularity. From numerous practical applications published over the past decade there seems to be substantial evidence that multilayer perceptorns indeed possess an impressive ability. Lately, there have also been some theoretical results that attempt to explain the reasons for this success, [Barron, 1993] [Judisky 1995].

1.3.1 Remarks

1. Artificial neural network solutions can now be implemented on special-chips and boards which offer considerably more throughput per dollar and more portability than conventional computers or super computers [Werbos, 1997].

2. Because the brain itself is made up of neural networks, artificial neural network designs seem like a natural way to try to replicate brain-like intelligence in artificial systems.(Advantages (1) and (2) follow directly from [Werbos, 1997]).

3. Designs are often much easier to use than the non-neural equivalents-especially when the conventional alternatives require first-principles models which are not well developed.

4. Various universal approximation theorems suggest that artificial neural network's can usually approximate what can be done with other methods and that the approximation can be as good as desired, if one can offered the computational cost of the accuracy required.

5. Artificial neural network designs usually offer solutions based on "learning" which can be far cheaper and faster than the traditional approach of elaborate prior research followed by tweaking applications until they work.

6. The artificial neural network literature includes designs to solve a variety of specific tasks-like function approximation, pattern

recognition, clustering, feature extraction, and a variety of novel control-related capabilities of familiar linear methods (Advantage (5) and (6) are not unique to artificial neural networks; most of the algorithms used to adapt artificial neural networks for specific tasks can also be used to adapt other nonlinear structures, such as fuzzy logic systems or physical models based on first principles or econometric model. For example, backpropagation-the most popular artificial neural network algorithm-was originally formulated in 1974 as a general algorithm, for use across a wide variety of nonlinear systems, of which artificial neural networks were discussed only as a special case [Werbos, 1994].

1.3.2 Remarks (General Advantages and disadvantages)

A. The general advantages of artificial neural networks including:

1. Access to existing sixth-generation computer hardware with hung price-performance advantages.

2. Links to brain-like intelligence.

3. Easy to use.

4. Superior approximation of nonlinear function.

5. Advantages of learning over tweaking including learning off-line to be adaptive on-line (in control).

6. Availability of many specific designs providing nonlinear generalizations of many familiar algorithms (among the algorithms and applications are those for image and speech preprocessing, function maximization or minimization, feature extraction, pattern classification, function approximation, identification and control of dynamical systems, data compression, and so on).

***B*. Disadvantages:**

1. Not exact.

2. Large complexity of the network structure.

1.4 What is Neural Network?

1.4.1 Neuron Model: [Haward, 1996]

The neuron or node unit (as we defined in section three), as it is also called, is a processing element that takes a number of inputs, weights them, sums them up, and uses the result as the argument for a singular valued function, the activation function (transfer function).

The neuron model and the architecture of a neural network describe how network transforms its input into an output. Those transformation can be viewed as a computation. The model and the architecture each place limitations on what a particular neural network can compute. The way a network computes its output must be understood before training methods for the network can be explained.

1.4.1.1 Simple Neuron:

A neuron with a single scalar input and no bias is shown on the figure (1.1a). The scalar input ***P*** is transmitted through a connection that multiplies its strength by the weight ***w***, to form the product ***wp,*** again a scalar. Here, the weighted input ***wp*** is the only argument of the transfer function ***f***, which produces the scalar output ***a***.

$$a = f(n) \equiv f(w\,p) \qquad (1.1)$$

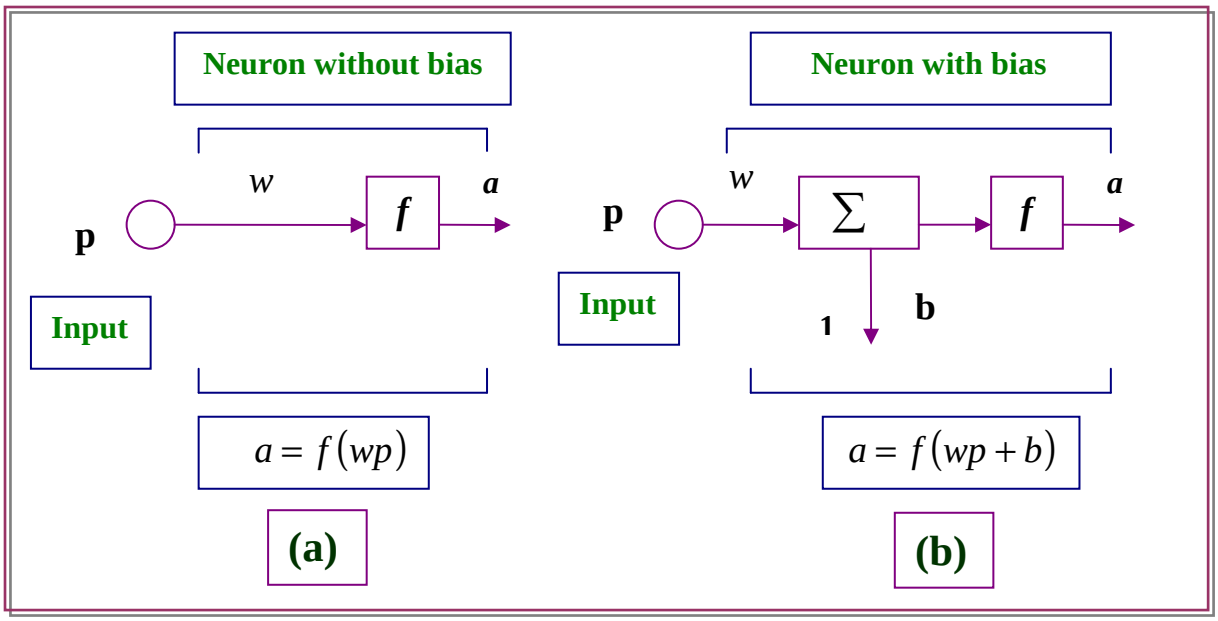


Figure (1.1)

Simple Neuron

And we can see the general neuron symbol in the figure (1.2) below

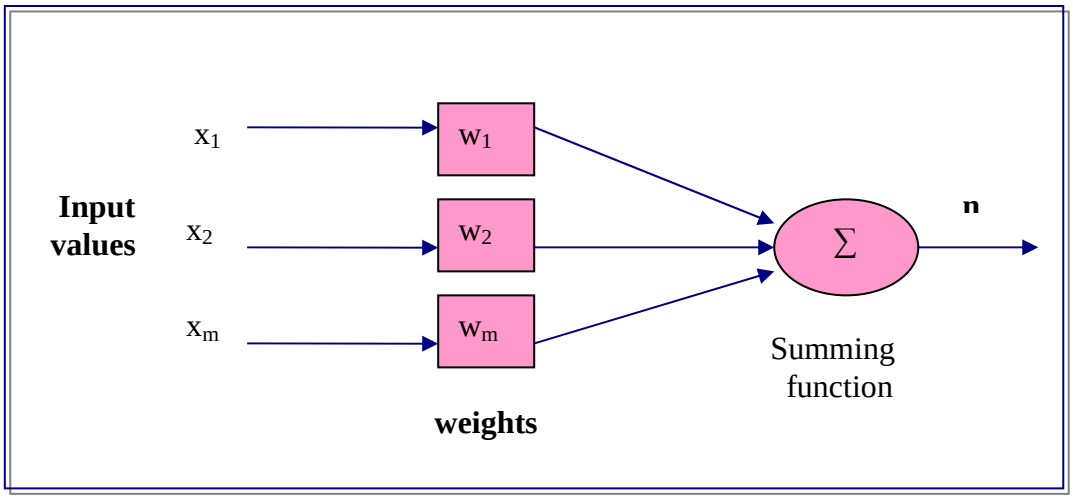


Figure (1.2)

General Neuron Symbol

The neuron on figure (1.1b) has a scalar bias, ***b***. One may view the bias as simply being added to the product ***wp*** as shown by the summing function or as shifting the function ***f*** to the left by an amount ***b.*** the bias is much like a weight, except that it has a constant input of value **1**. the

transfer function net input ***n***, again a scalar, is the sum of the weighted input ***wp*** and the bias ***b***. this sum is the argument of the transfer function (activation function) ***f***. Examples of various transfer functions are given in the next section. Note that ***w*** and ***b*** are both adjustable scalar parameters of the neuron.

$$a = f(n) \equiv f(w\,p + b) \qquad (1.2)$$

As shown below in figure (1.3), the bias ***b*** is an adjustable (scalar) parameter of the neuron. It is not an input. However, the constant ***1*** that drives the bias is an input and must be treated as such when considering the linear dependence of input vectors.

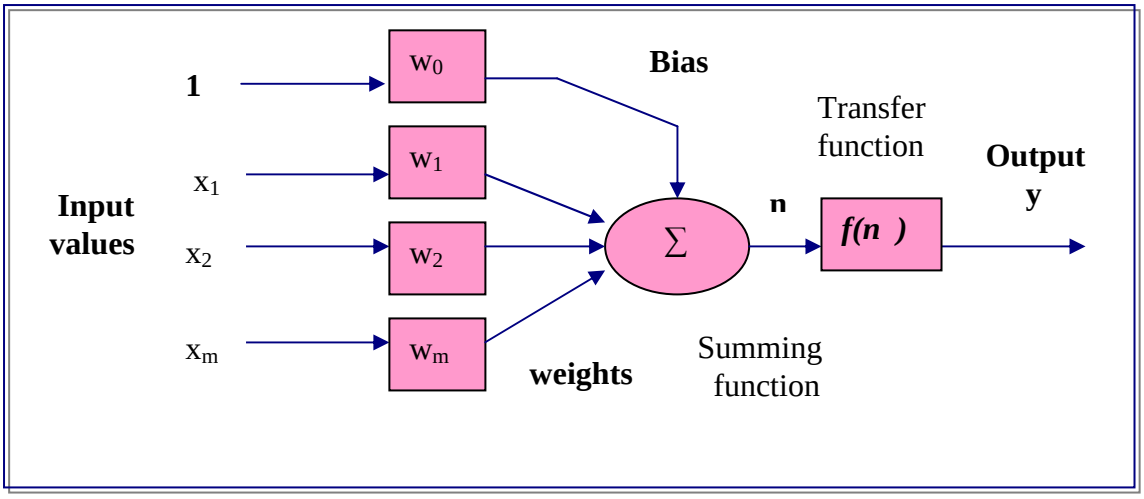


Figure (1.3)

Network with Bias Unit

1.4.1.1 Remark

The central idea of neural networks is that such parameters can be adjusted so that the network exhibits some desired or interesting behavior. Thus, we can train the network to do a particular job by adjusting the weight or bias parameters, or perhaps the network itself will adjust these parameters to achieve some desired end.

1.4.1.2 Transfer Function (Activation function):

As we defined the transfer function (activation function) in section three, the choice of it determines the neuron model, such that, there are different kinds of the transfer function, for examples:

1. Hyperbolic tangent sigmoid transfer function: a squashing function of the form shown below that maps the input to the interval (1,-1)

$$f\ (n) = 2/\left(1+\exp\left(-2*n\ \right)\right)-1 \qquad (1.3)$$

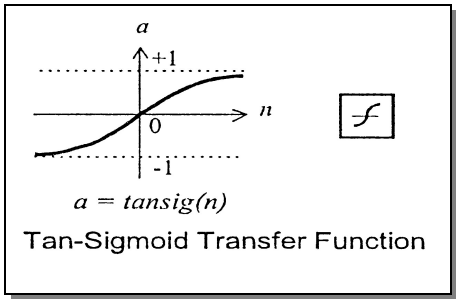


Figure (1.4)

Tangent-Sigmoid Transfer Function

2. Symmetric saturation linear transfer function: produces the input as its output as long as the input i in the range -1 to 1. Outside that range the output is -1 and +1 respectively.

3. Symmetric hard limit transfer function: a transfer that maps inputs greater-than or equal-to 0 to +1, and all other values to -1.

$$f\ (n) = \begin{cases} 1 & if\ \ n \geq 0 \\ -1 & otherwise \end{cases} \qquad (1.4)$$

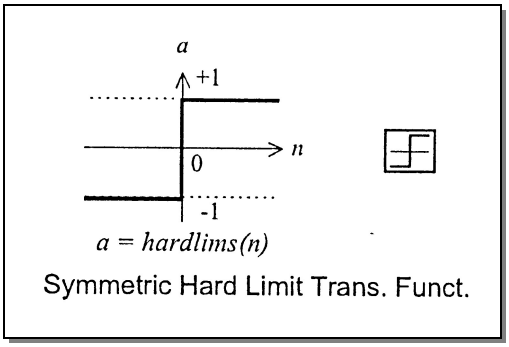


Figure (1.5)

Symmetric Hard Limit Transfer function

4. Saturating linear transfer function: a function that is linear in the interval (-1,+1) and saturates outside this interval to -1 or +1.

$$f\ (n)=\begin{cases}0 & if\ \ n\le 0\\ n & if\ \ 0\le n\le 1\\ 1 & if\ \ 1\le n\end{cases} \qquad (1.5)$$

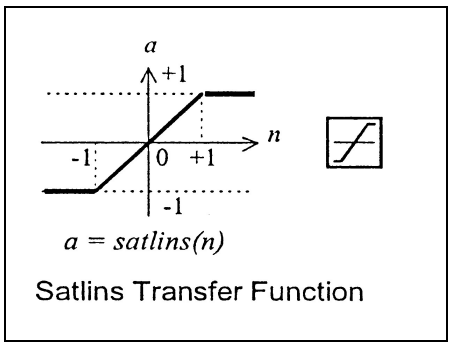


Figure (1.6)

Saturating linear transfer function

5. Radial basis transfer function: the transfer function for a radial basis neuron is:

$$radbas\ (n)=\exp^{-n^2} \qquad (1.6)$$

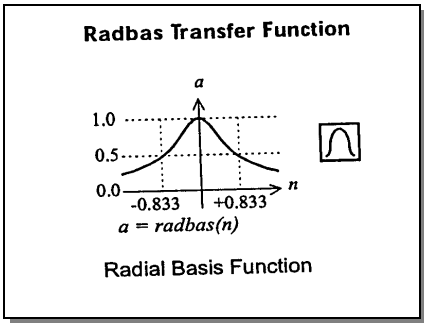


Figure (1.7)

Radial Basis Transfer Function

6. Positive linear transfer function: a transfer function that produces an output of zero for negative inputs and an output equal to the input for positive input.

$$f\ (n) = \begin{cases} n & if\ \ n \geq 0 \\ 0 & if\ \ n \leq 0 \end{cases} \qquad (1.7)$$

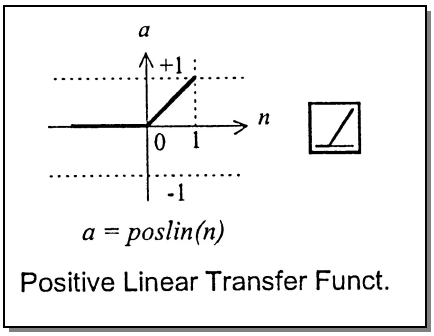


Figure (1.7)

Positive Linear Transfer Function

7. Log-sigmoid transfer function: a squashing function of the form shown below that maps the input to the interval (0, 1).

$$f\ (n) = \frac{1}{1+\exp^{-n}} \qquad (1.8)$$

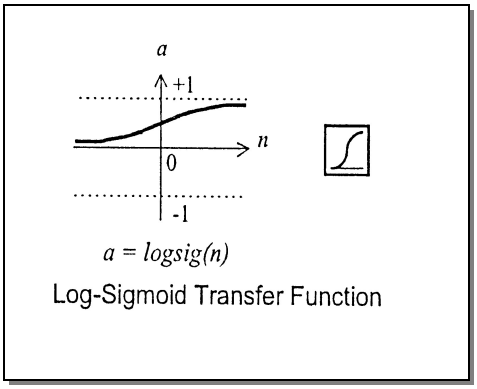


Figure (1.8)

Log-sigmoid Transfer function

8. Linear transfer function: a transfer function that produces its input as its output.

$$f\ (n) = n \quad \forall n \tag{1.9}$$

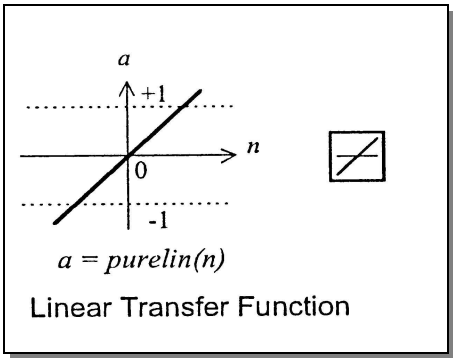


Figure (1.9)

Linear Transfer function

9. Hard limit transfer function: a transfer that maps inputs greater-than or equal-to 0 to 1, and all other values to 0.

$$f\ (n) = \begin{cases} 1 & if\ \ n \geq 0 \\ 0 & otherwise \end{cases} \tag{1.10}$$

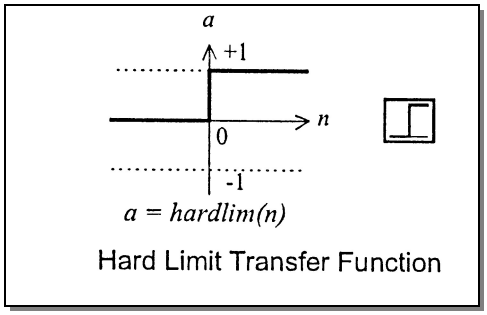


Figure (1.10)

Hard Limit Transfer Function

10. Competitive transfer function: accepts a net input vector for a layer and returns neuron outputs of 0 for all neurons except for the "winner", the neuron associated with the most positive element of the net input ***n***.

11. Triangular basis transfer function: a function calculate its output with according to

$$\begin{aligned} tribas(n) &= 1 - abs(n) \quad , if\ -1 \le n \le 1; \\ &= 0 \qquad\qquad\ \ , otherwise \end{aligned} \tag{1.11}$$

12. Class of sigmoid transfer function:

$$f(n) = \frac{1}{1+\exp(-\lambda \cdot n)} \tag{1.12}$$

Where λ is a positive constant (so-called steepness), whose value specifies a particular sigmoid function in this class.

13. Sigmoid transfer function :(with z, x, y parameters)

$$f(n) = z + \frac{1}{1+\exp(-xn+y)} \tag{1.13}$$

14. Gaussian transfer function:

$$f(n)=\frac{1}{\sqrt{2\pi\sigma}}\exp\left[-\frac{1}{2}\left[\frac{n-\mu}{\sigma}\right]^2\right] \qquad (1.14)$$

1.4.1.2 Remarks:

1. The hard limit transfer function is a special sigmoid function, obtained as $\lambda\rightarrow\infty$

2. The sigmoid function (***Hyperbolic tangent sigmoid and Log-sigmoid***) has historically been a very popular choice [Rumelhart, 1986].

1.4.1.3 Neuron with Vector Input:

A neuron with a single R-element input vector is shown below. Here the individual element inputs

$$P=\left[p_1,p_2,\ldots,p_R\right]^T \qquad (1.15)$$

Are multiplied by weights

$$W=\left[w_{1,1}\quad w_{1,2}\quad \cdots\quad w_{1,R}\right]^T \qquad (1.16)$$

and the weighted values are fed to the summing function. Their sum is simply **WP,** the dot product of the matrix **W** and the vector **P.**

$$a=f\left(W^T P+b\right) \qquad (1.17)$$

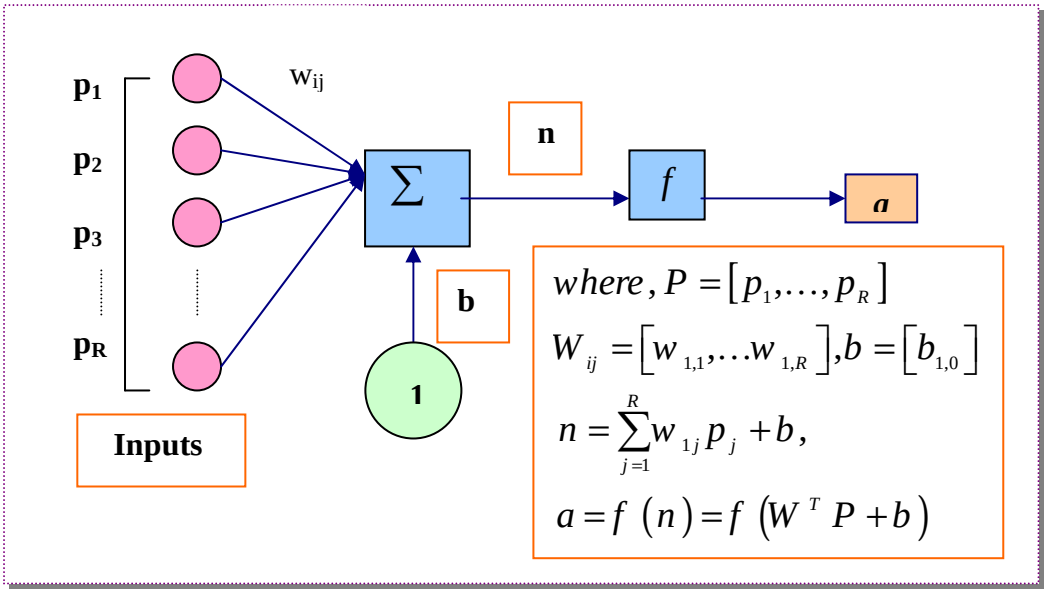


Figure (1.11a)
Neuron with Vector Input

The neuron has a bias ***b***, which is summed with the weighted inputs to form the net input **n** . This sum, **n**, is the argument of the transfer function ***f***.

$$n = \sum_{j=1}^{R} w_{1j}\, p_j + b \qquad (1.18)$$

this expression can rewritten as :

$$n = W * P + b \qquad (1.19)$$

We have re-down the neuron with a single R-element input vector network in figure (1.11a) shown above, in a abbreviated form below in figure (1.11b)

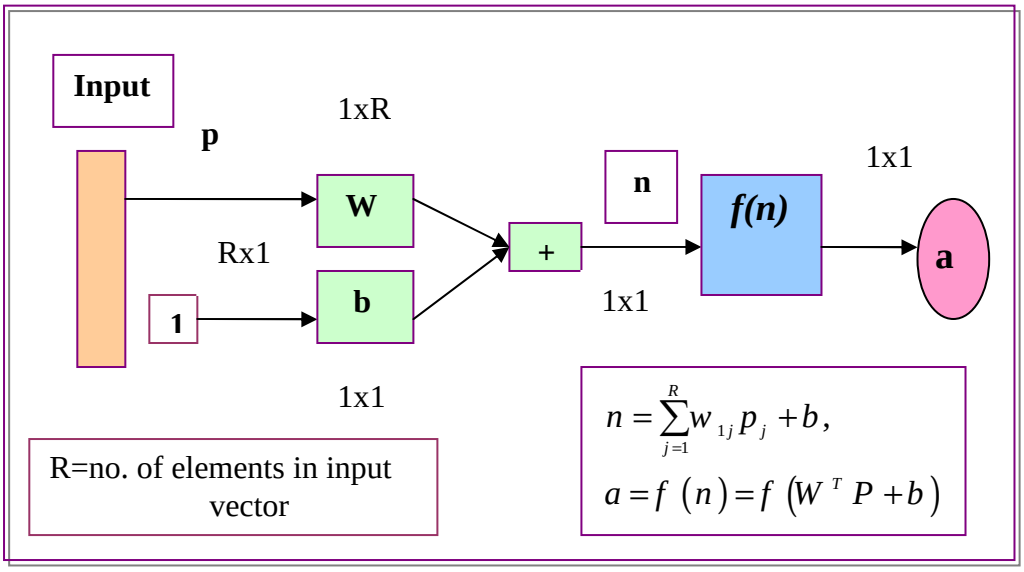


Figure (1.11b)
Neuron with Vector Input

Here the input vector **p** is represented by the solid dark vertical bar at the left. The dimensions of **p** are shown below the symbol **p** in the figure as Rx1.

P is a vector of R input elements. These inputs post multiply the single row, R column matrix **W**. As before, a constant 1 enters the neuron as an input and is multiplied by a scalar bias ***b***. The net input to the transfer function ***f*** is **n**, the sum of the bias ***b*** and the product **Wp.** This sum is passed to the transfer function ***f*** to get the neuron's output **a**, which in this case is scalar. Note that if we had more than one neuron, the network output would be a vector [Zurada, 1996].

$$a_i = f\left(W_i^T P\right) \qquad for\ i = 1, 2, \ldots, S \tag{1.20}$$

where weight vector w_i contains weights leading toward the **ith** output node, and is defined as follows

$$W_i = \left[w_{i1} \quad w_{i2} \quad \cdots \quad w_{iR}\right]^T \tag{1.21}$$

1.4.2 Network Architectures

1.4.2.1 Remarks

Before the training can be performed, some issues need special attention. Unfortunately, not all questions are easily answered:

1. What type of relationships can be learned with a multilayer network.

2. How many hidden layers should the network have and how many neurons should be included in each layer?

3. How should the transfer functions be chosen?.

In **[Cybenko, 1989]** it is shown that all continuous functions can be approximated to any desired accuracy, in terms of the uniform norm, with a network of one hidden layer of sigmoidal hidden neurons and a layer of linear output neurons. But it dose not explains how many neurons to include in the hidden layer. This issue is addressed in [Barron, 1993] and a significant result is derived about the approximation capabilities of two-layer networks when the function to be approximated exhibits certain smoothness. Unfortunately, the result is difficult to apply in practice for selecting the number of hidden neurons.

Due to the above mentioned results one might think that there is no need for using more than one hidden layer and/or mixing different types of activation functions (transfer function). This not quite true as it may occur that accuracy can be improved using more sophisticated network

architecture. In particular when the complexity of the mapping to be learned is high, it is likely that the performance can be improved. However, since implementation, training, and statistical analysis of the network become more complicated, it is customary to apply only a single hidden layer of similar activation functions (transfer functions) and an output layer of linear neurons, [Norgaard, 2000].

Now consider a single layer of neurons with details.

1.4.2.2 Neurons Layers:

A one layer network with *R* input elements and *S* neurons is shown below in the following figure (1.12a).

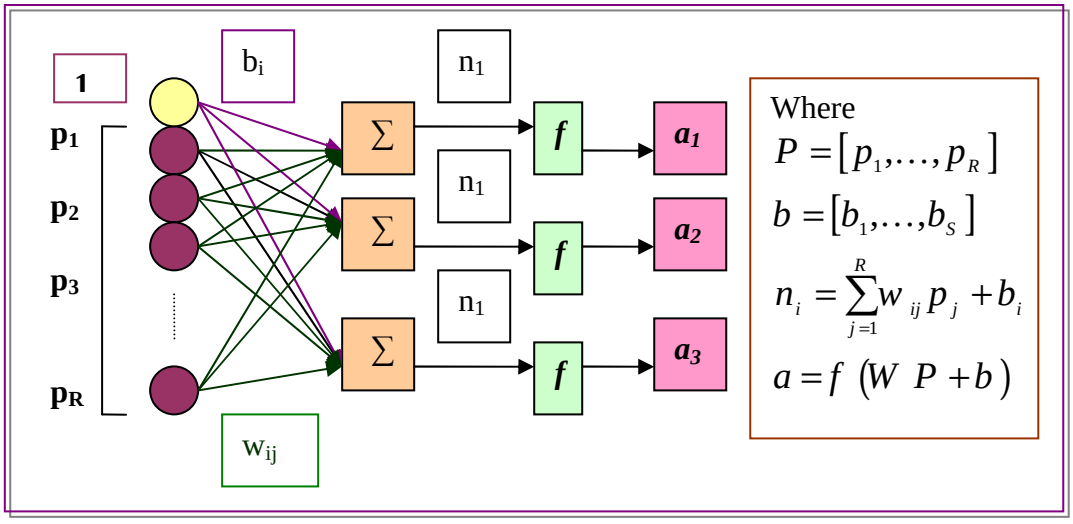


Figure (1.12a)
One Layer Neural Network

The S neuron R input one layer network also can be drown in abbreviated notation.

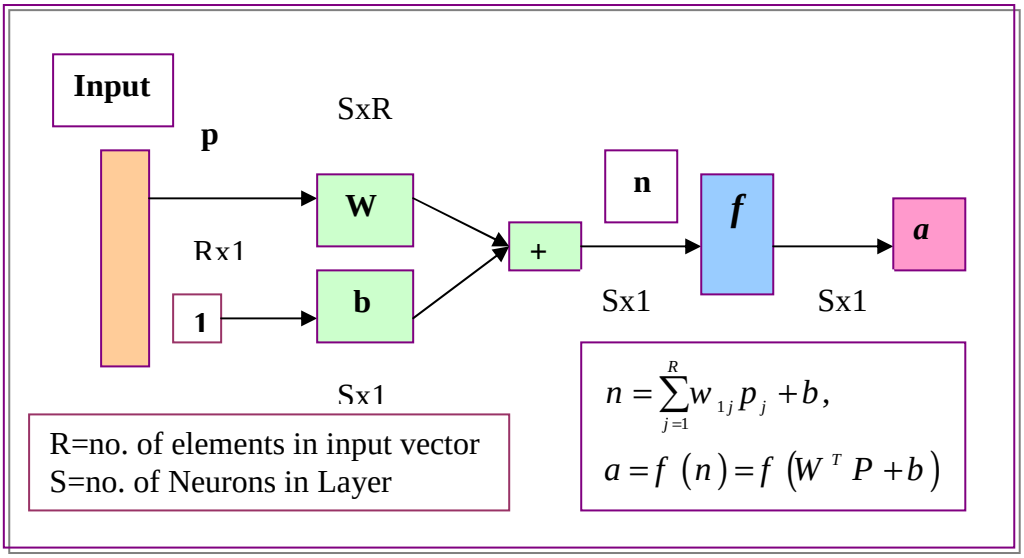


Figure (1.12b)

One Layer Neural Network

In this network, each element of the input vector $P = \begin{bmatrix} p_1 & p_2 & \dots & p_R \end{bmatrix}$ is connected to each neuron input through the weight matrix **W.** The ***i*th** neuron has a summand that gathers its weighted inputs and bias to form its own scalar output $\mathbf{n}_i$**.** The various $\mathbf{n}_i$ taken together form an *S*-element net input vector **n**. Finally, the neuron layer outputs from a column vector $a = \begin{bmatrix} a_1 & a_2 & \dots & a_S \end{bmatrix}^T$. We show the expression for **a** at the bottom of the figure (1.12b).

The input vector elements enter the network through the weight matrix **W.**

$$W = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{pmatrix} \qquad (1.22)$$

Note that the row indices on the elements of matrix **W** indicate the destination neuron of the weight and the column indicates which source is the input for the weight. Thus, the indices in $\mathbf{w}_{1,2}$ say that the strength of

the signal from the second input element to the first (and only) neuron is $\mathbf{w}_{1,2}$. Such that

$$n_i = \sum_{j=1}^{R} w_{ij}\, p_j \qquad , for\, i = 1,2,\ldots,S \tag{1.23}$$

$$a_i = f\left(\sum_{j=1}^{R} w_{ij}\, p_j \right) \qquad for\, i = 1,2,\ldots,S \tag{1.24}$$

1.4.2.2 Remarks

The following notations are used

1. weight matrices connected to inputs, ***input weight***.

2. weight matrices coming from layer outputs, ***layer weight***.

3. we will use superscripts to identify the source (**second index**) and the destination (**first index**) for the various weights and other elements of the network. Further, we have re-drawn the one layer multiple input network shown above (Figure 1.11a, 1.11b) in abbreviated form below.

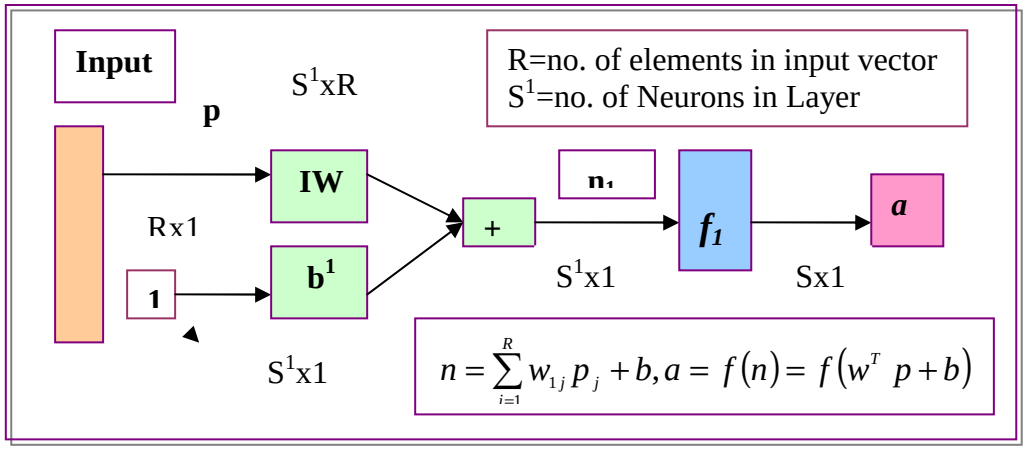


Figure (1.12c)

One Layer Neural Network

1.4.2.3 Multiple Layer of Neurons

The following is the architectures of multilayered neural network, figure (1.13) shows the notations of 3-layer neural network where the network can have several layers. Each layer has a weight matrix **W**, a bias vector ***b***, and an output vector **a**. To distinguish between the weight matrices, output vectors, etc., for each of those layers in our figures, we will append the number of the layer as a superscript to the variable of interest.

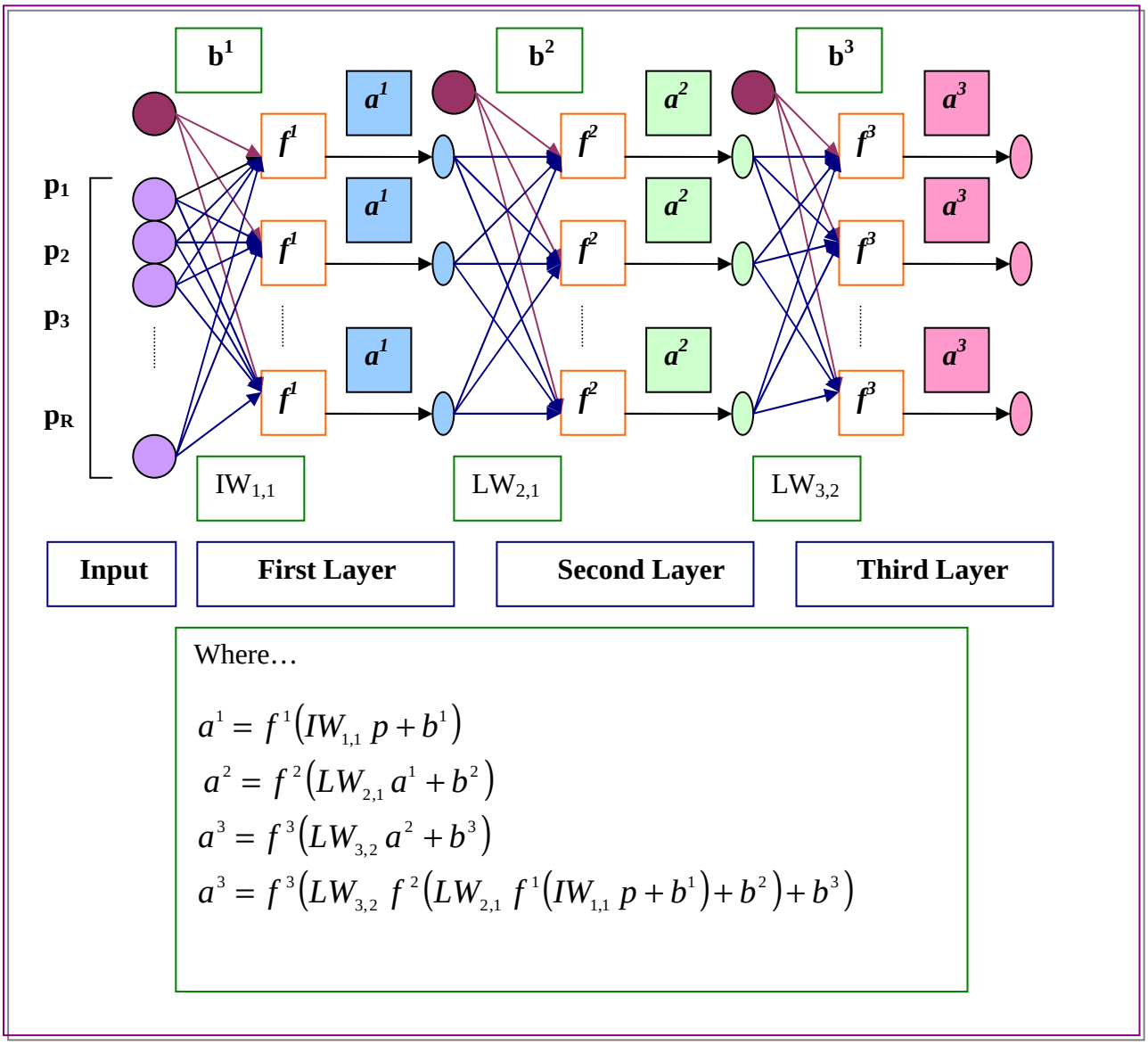


Figure (1.13a)

Three Layer Neural Network

The same three layer network discussed previously also can be drowning using our abbreviated notation.

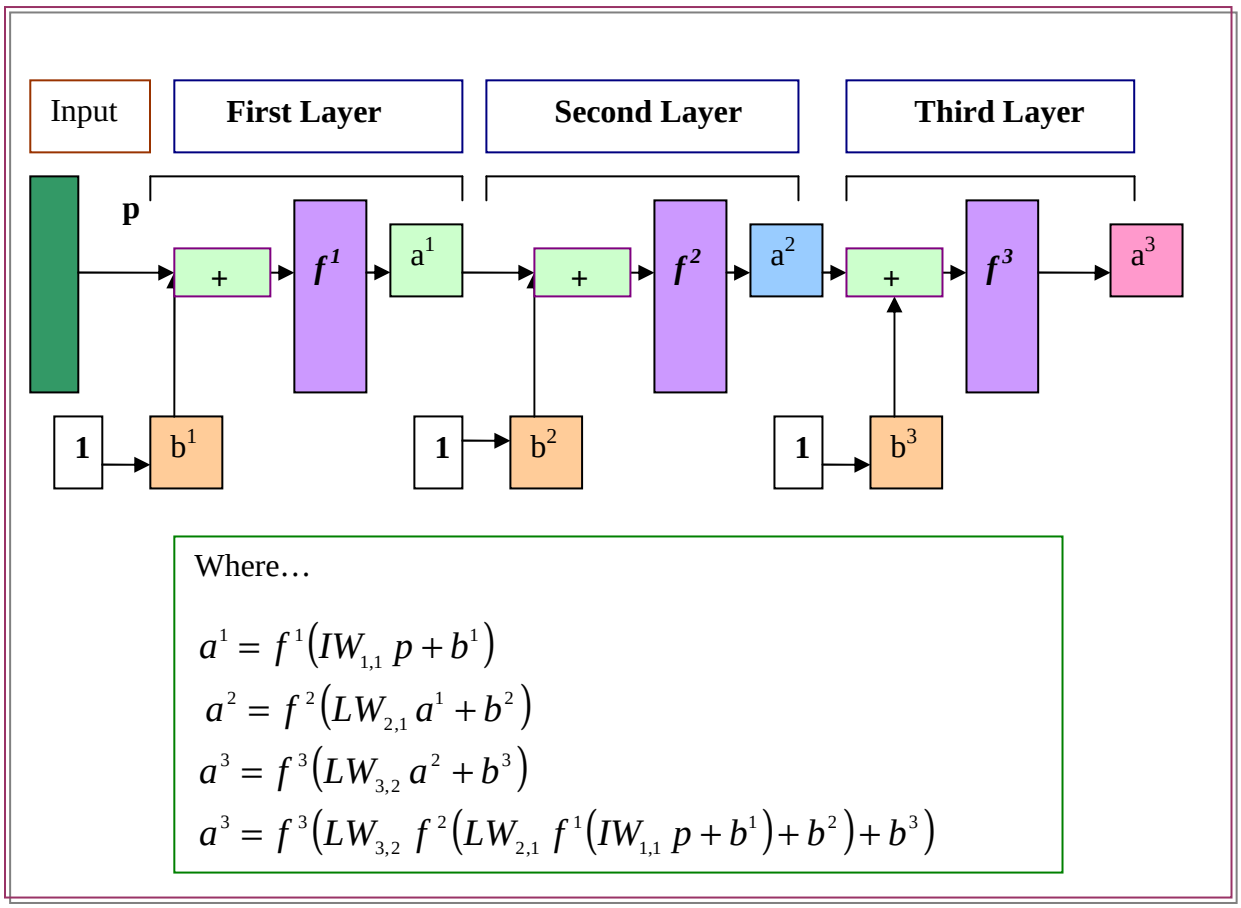


Figure (1.13b)

Three Layer Neural Network

In figure (1.13a, 1.13b), the first layer is called the ***Input layer*** which connected with the inputs of the network, the third layer is called the ***output layer*** referring to the fact that it produces the output of the network, and the second layer is known as the ***hidden layer*** since it is in some sense hidden between the external inputs $[\,p_1, p_2, p_3\,]$ and the output layer.

Note that the outputs of each intermediate layer are the inputs to the following layer. Thus $\boldsymbol{a^1}$ is the output of the ***input layer*** (first layer)

$$a^1 = f^1\left(\mathrm{IW}_{1,1}\, p + b_1\right) \quad \textbf{(}1.25\text{a}\textbf{)}$$

so that a^1 can be written as

$$a_j^1 = f_j^1\left(\sum_{l=1}^{R^1} W_{jl}\, p_l + b_j\right) \quad , \textit{for } j = 1, 2, \ldots, S^1 \qquad (1.25\text{b})$$

where a^1 will be the input to the ***hidden layer*** (second layer), and a^2 will be the output of the hidden layer

$$a^2 = f^2\left(LW_{2,1}\, a^1 + b_2\right) \qquad (1.26\text{a})$$

which can be written as

$$a_i^2 = f_i^2\left(\sum_{j=1}^{S^1} W_{ij}\, a_j^1 + b_i\right) \quad , \textit{for } i = 1, 2, \ldots, S^2 \qquad (1.26\text{b})$$

where a^2 will be the input of the ***output layer*** (third layer), and a^3 will be the output of the network

$$a^3 = f^3\left(LW_{3,2}\, a^2 + b_3\right) \qquad (1.27\text{a})$$

which can be written as

$$a_k^3 = f_k^3\left(\sum_{i=1}^{S^2} W_{ki}\, a_i^2 + b_k\right) \quad , \textit{for } k = 1, 2, \ldots, S^3 \qquad (1.27\text{b})$$

such that the mathematical formula expressing that is going on in the three-layer network takes the form

$$a^3 = f^3\left(LW_{3,2}\, f^2\left(LW_{2,1}\, f^1\left(IW_{1,1}\, p + b_1\right) + b_2\right) + b_3\right) \qquad (1.28\text{a})$$

which can be written as

$$a_k = f_k\left(\sum_{i=1}^{S^2} W_{ki}\, f_i\left(\sum_{j=1}^{S^1} W_{ij}\, f_j\left(\sum_{l=1}^{R^1} W_{jl}\, p_l + b_j\right) + b_i\right) + b_k\right) \qquad (1.28\text{b})$$

where $j = 1,2,\ldots S^1$, $i = 1,2,\ldots, S^2$, $k = 1,2,\ldots, S^3$

The network shown above has R^1 inputs, S^1 neurons in the first layer, S^2 neurons in the second layer, etc., it is common for different layers to have different numbers of neurons. A constant input value **1** is fed to the biases for each neuron. The three-layer neural network is a vector-matrix form as shown below

$$a^3 = f^3\left(LW_{3,2}\, f^2\left(LW_{2,1}\, f^1\left(IW_{1,1}\, p + b_1\right) + b_2\right) + b_3\right) = y \quad (1.29)$$

1.4.2.3.1 Remarks

1.Multiple layer networks are quite powerful. For instance, a network of two layers, where the first layer is sigmoid and the second layer is linear, can be used to approximate any function (with a finite number of discontinuities) arbitrarily well [Zurada, 1996].

2.The styles of approach of the two extremes are some what different. The subject of artificial neural computing is based on networks. There are two extremes of the architectures of the networks: **Feed forward networks** (input streams steadily through the network from a set of input neurons to a set of output ones) and **recurrent networks (**where there is constant feedback from the neurons of the network to each other) [Hophield, 1982]. This is mirrored in the differences between the topologies such networks posses: one is the line, and the other the circle, which cannot be topologically deformed into each other. As is to be expected, there are two extreme styles of computation in these networks. In the feed forward case the input moves through the network to become the output; in the recurrent network the activities in the network develop over time until it settles into some asymptotic value which is used as the output of the network. The network thus relaxes into this asymptotic state.

There are several kind of networks (see appendix **B**) , so we will discussed the feed forward networks because we use it in our work.

1.4.2.4 Basic Neural Network Architectures

1. Feed forward Neural Network: The best known neural network architecture is the multilayer **feed forward neural network** (**multilayer perceptron**). It is a static network that consists of a number of layers: ***input layer***, ***output layer*** and or more ***hidden layers*** connected in a feed forward way[Zurada, 1996]**.** Each layer consists of a number of **McCulloch-pitts** neurons. One single neuron makes the simple operation of a ***weighted*** sum of the incoming singles and a ***bias*** term (or threshold), fed through a ***transfer function*** (activation function) ***f*** and resulting in the output value of the neuron. A network with one hidden layer is described in matrix-vector notation as

$$a = W\ f(V\ p + b) \tag{1.30}$$

or in element wise notation:

$$a_i = \sum_{r=1}^{nh} w_{ir}\ f\left(\sum_{j=1}^{m} v_{rj}\ p_j + b_r \right) \qquad i = 1, \ldots, l \tag{1.31}$$

Here $p \in R^m$ is the input and $a \in R^l$ the output of the network and the nonlinear operation ***f*** is taken element wise. The interconnection matrices are $W \in R^{l \times nh}$ for the output layer $V \in R^{nh \times m}$ for the hidden layer, $b \in R^{nh}$ is the bias vector (thresholds of hidden neurons) with nh the number of hidden neurons.

For a network with two hidden layers one has

$$a = W\ f\left(V_2\ f\left(V_1\ p + b_1 \right) + b_2 \right) \tag{1.32}$$

or

$$a_i = \sum_{r=1}^{nh2} w_{ir}\ f\left(\sum_{s=1}^{nh1} v_{rs}^{(2)} f\left(\sum_{j=1}^{m} v_{sj}^{(1)} p_j + b_s^{(1)} \right) + b_r^{(2)} \right) \qquad i = 1, \ldots, l \tag{1.33}$$

The interconnection matrices are $W \in R^{l \times nh2}$ for the output layer, $V_2 \in R^{nh2 \times nh1}$ for the second hidden layer and $V_1 \in R^{nh1 \times m}$ for the first hidden layer. The bias vectors are $b_2 \in R^{nh2}, b_1 \in R^{nh1}$ for the second and first hidden layer respectively. In order to describe a network with L layers (L-1 hidden layers, because the input layer is a 'dummy' layer), the following notation will be used in the sequel

$$p_i^l = f\left(\xi_i^l\right), \qquad \xi_i^l = \sum_{j=1}^{N_1} w_{ij}^1 \, p_j^{l-1} \tag{1.34}$$

Where $l = 1, \ldots, L$ is the layer index, N_l denotes the number of neurons in layer l and p_i^l is the output of the neurons at layer l. The thresholds are considered here to be part of the interconnection matrix, by defining additional constant inputs.

The choice of the transfer function $\boldsymbol{f}$ may depends on the application are. Typical transfer functions are shown in section four. For applications in modeling and control the hyperbolic tangent function

$$\tanh(p) = (1 - \exp(-2p))/(1 + \exp(-2p)) \tag{1.35}$$

is normally used. In case of a 'tanh' the derivative of the transfer function is $\dot{f} = 1 - f^2$.The neurons of the input layer have a linear transfer function.

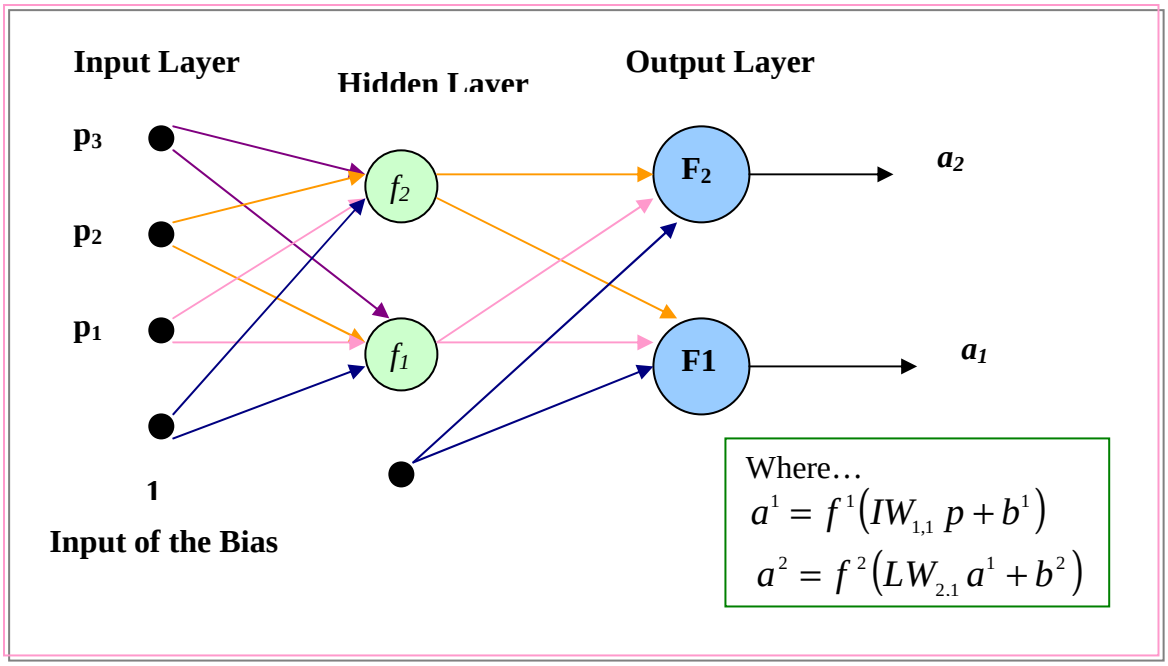


Figure (1.14)

A fully Connected Two Layer Feed forward Network with Three Inputs, Two Hidden Neurons and Two Outputs

For more information about different networks such as (***Radial Basis function network, Recurrent neural networks)*** see appendix **B.**

1.4.3 Mathematical Theory of Neural Learning

We defined a ***learning rule*** in section three as a procedure for modifying the weights and biases of a network. (This procedure may also be referred to as a **training algorithm**). The learning rule is applied to train the network to perform some particular task. Learning rulers fall into two broad categories: **supervised learning** and **unsupervised learning**.

In ***supervised learning***, the learning rule is provided with a set of examples (the *training set*) of proper network behavior:

$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$

Where $\mathbf{p}_q$ is an input to the network, and $\mathbf{t}_q$ is the corresponding correct (*target*) output. As the inputs are applied to the network, the network

outputs are compared to the targets. The learning rule is then used to adjust the weights and biases of the network in order to move the network outputs closer to the targets.

In ***unsupervised learning***, the weights and biases are modified in response to network inputs only. There are no target outputs available. Most of these algorithms perform clustering operations. They categorize the input patterns into a finite number of classes. This is useful in such applications as vector quantization [Rosenblatt, 1961].

Equivalently the ***supervised learning*** algorithm is guided by specifying, for each training input pattern, the class to which the pattern is supposed to belong. That is, the desired response of the network of each training input pattern and its comparison with the actual output of the network are used in the learning algorithm for appropriate adjustments of the weights. These adjustments, whose purpose is to minimize the difference between the desired and actual outputs, are made incrementally. That is, small adjustments in the weights are made in the desired direction for each training pair. This is essential for facilitating a convergence to a solution (specific values of the weights) in which patterns in the training set are recognized with high fidelity. Once a network converges to a solution, it is then capable of classifying each unknown input pattern with other patterns that are close to it in terms of the same distinguishing features. While in an ***unsupervised learning*** algorithm, the network forms its own classification of patterns. The classification is based on commonalities in certain features of input patterns. This requires that a neural network implementing an unsupervised learning algorithm be able to identify common features across the range of input patterns [Paual, 1997].

1.4.3.1 Remark

Some theoretical results concerning the artificial neural networks and its derivates, necessary mathematical backgrounds and its generalizations, etc, can be found with some details in appendix **C** and for simplicity the information's have omitted from here.

1.4.3.1 General Learning Equation:

A neuron has ability to modify its connection weights $w = \left[w_1, w_2, \ldots, w_n \right]$, depending on the input signals $p = \left[p_1, p_2, \ldots p_n \right]$ which it receives and the associated teacher signals or error signals. The teacher or error signal is not provided in some cases, where a neuron modifies its weights depending only on its state and input signal. This is the case of **non supervised learning**, and such a learning scheme is sometimes called **self-organization**. In order to build a general theory of neural learning, we consider the following situation [Amari, 1977]: A neuron receives input signals p from an information source I, to which it is to adapt. A set of training signals $p_\alpha, \alpha = 1, 2, \ldots, k$, may be regarded as a set of examples from the information source (Figure 1.15).

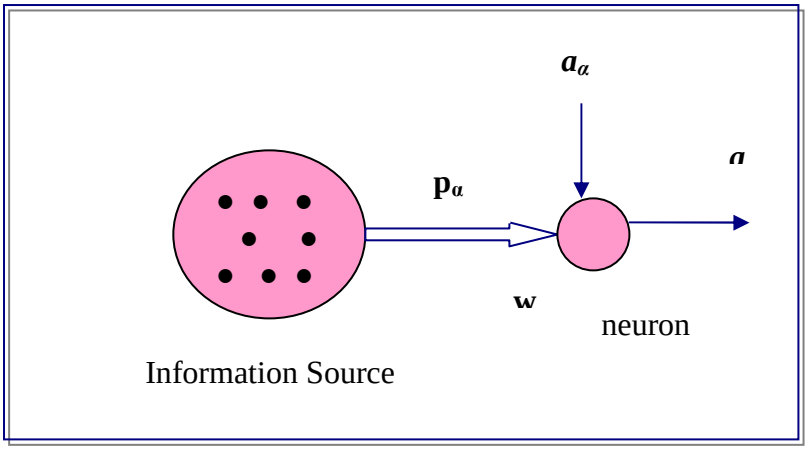


Figure (1.15)

Learning Scheme

The probability or relative frequency of p_α is denoted by $\overline{\mathrm{P}}_\alpha, \sum \overline{\mathrm{P}}_\alpha = 1$. Let a_α be the teacher signal associated with input p_α. One may use the error signal e_α instead of a_α .

The environmental information source I is, henceforth, specified by a probability structure of signal pairs.

$$\mathrm{I} = \left\{ \left(p_\alpha, a_\alpha, \overline{\mathrm{P}}_\alpha \right), \alpha = 1, \ldots, k \right\} \tag{1.36}$$

or more generally by

$$\mathrm{I} = \left\{ \left(p, a, \overline{\mathrm{P}}(p,a) \right) \right\} \tag{1.37}$$

Where $\overline{\mathrm{P}}(p,a)$ represents the probability distribution of a pair (p,a) of an input p and the associated a . In some cases a is missing, so that

$$\mathrm{I} = \left\{ \left(p, \overline{\mathrm{P}}(p) \right) \right\} \tag{1.38}$$

1.4.3.2 Learning Neural Networks:

Learning behaviors of a single neuron are treated in the previous part, where we fix the environmental information source I. When a neural network modifies its behavior cooperatively, each neuron changes its connection weights according to the learning equation. The environmental information source I of each neuron, however, is not fixed but changes as the other neurons modify their connection weights, because the information source I of one neuron is given by the behaviors of other neurons in the network. Hence, we need to solve a set of the mutually coupled learning equations.

We give an example of learning networks.

1.4.3.2.1 Backpropagation and its Generalization:

1. General Learning Scheme:

Let us consider a neural network ***N,*** which receive a vector input signal p, processes it, and emits a vector output a. Let ***S*** be the set of modifiable parameters (connection weights and thresholds). Which

specify the network. The output is determined as a function of input p and the parameter values $\boldsymbol{S}$ as

$$a = f(p,S) \tag{1.39}$$

depending on the network architecture.

Let I be the information source of the network, which emits signal p with probability $\overline{\mathrm{P}}(p)$**,** for each signal p there is an associated desired output $a = d(p)$**.** Amari in 1967 formulated a general learning scheme of neural networks in the following manner.

Let $\mathrm{I}(p,S)$ be a loss when input p is processed by a network whose parameter value are $\boldsymbol{S}$. A simple example of the loss is the squared error

$$\mathrm{I}(p;S) = \frac{1}{2}\left|f(p;S) - d(p)\right|^2 \tag{1.40}$$

Which is used in the **back propagation learning rule**. However, there are many other types of reasonable loss. The expected loss is given by

$$L(S) = \langle \mathrm{I}(p;S) \rangle = \int \overline{\mathrm{P}}(p)\, \mathrm{I}(p;S)\, dp \tag{1.41}$$

The parameters which minimize $L(S)$ give the best network, which satisfies

$$\frac{\partial L(S)}{\partial S} = 0 \tag{1.42}$$

The best parameters may be obtained by the **gradient method**.

The learning rule is given by the gradient method as

$$S_{t+1} = S_t - c_t \frac{\partial\, \mathrm{I}(p_t;S_t)}{\partial S} \tag{1.43}$$

Where $\boldsymbol{S_t}$ are the values of $\boldsymbol{S}$ at time $\boldsymbol{t}$ and p_t is the input at $\boldsymbol{t}$. The parameter $\boldsymbol{S_t}$ approaches one of the local minima of $L(S)$. Amari in 1967 proposed, more than 39 years ago, this type of general learning and

studied the trade-off between the speed and the accuracy of convergence. He also applied the method to layered neural networks (Figure 1.16), and gave a general learning rule including that of the so-called **hidden units**.

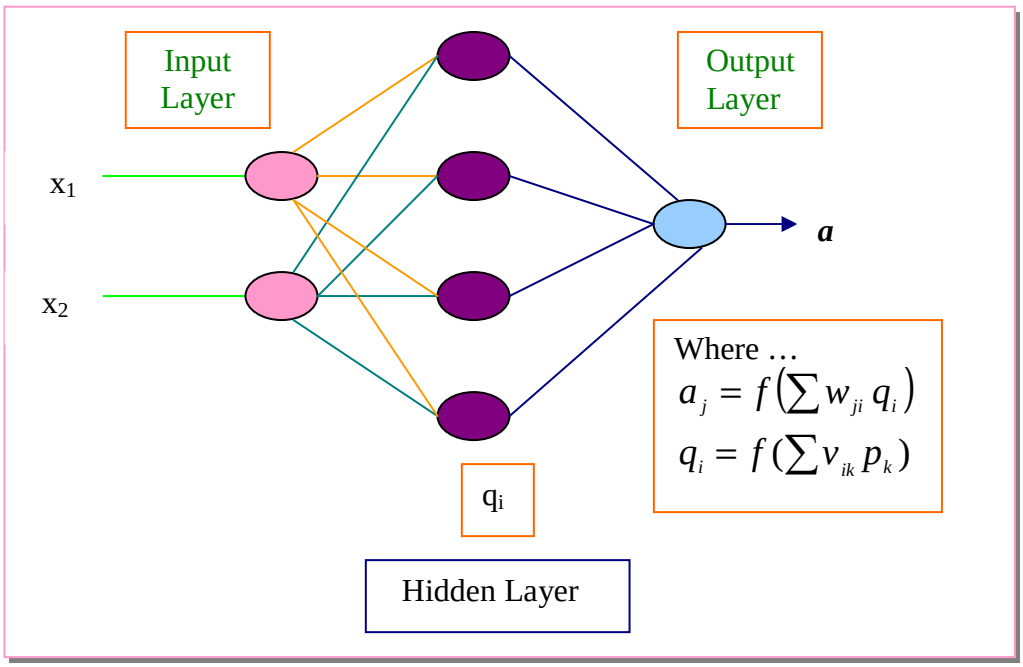


Figure (1.16)

Layered Neural Network

2. Back propagation of Layered Networks:

We show the famous **back propagation rule** as an example of the general scheme. Here we use three layer networks (Figure 1.17),

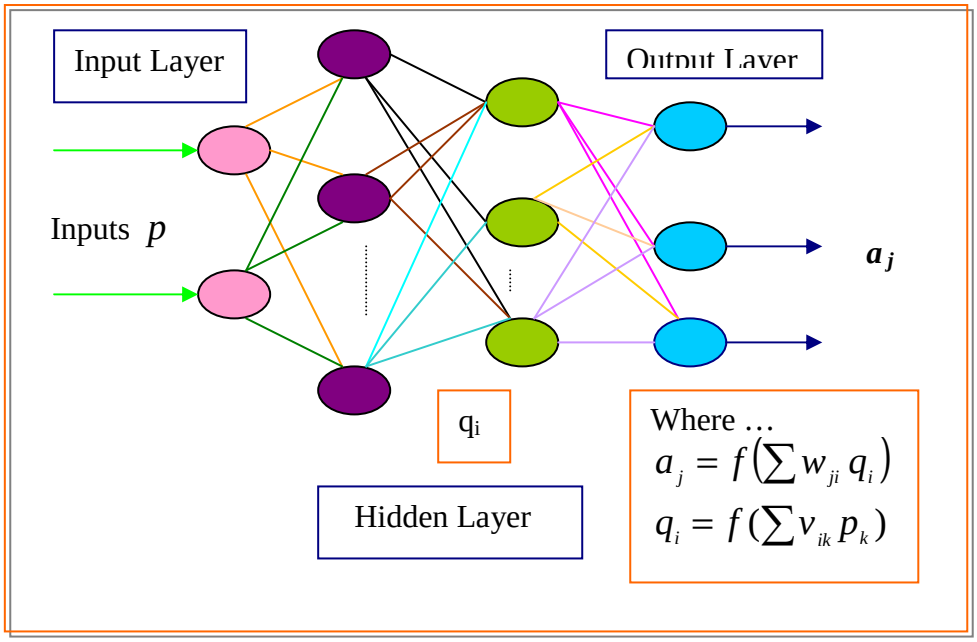


Figure (1.17)

Back propagation learning

But the same rule holds for more general layered networks. For input P, the **ith** element of the hidden unite emits $q_i = f(\sum w_{ik}\ p_k)$ and the **jth** element of the output unit emits

$$a_j = f\left(\sum v_{ji}\, q_i\right) = f\left(\sum_i v_{ji}\ f\left(\sum_k w_{ik}\ p_k\right)\right) \qquad (1.44)$$

Where w_{ik} and v_{ji} are the connection weights from the input layer to the hidden layer, and from the hidden layer to the output layer. Threshold values may be included in the above by adding a unit which emits a constant value.

The modifiable parameters are $S = \left(v_{ji}, w_{ik}\right)$. For the error loss $\mathrm{I} = 0.5\, e^2, e = |a - d|$,

$$\frac{\partial \mathrm{I}(p_t; S)}{\partial v_{ji}} = r_j q_i \qquad (1.45)$$

where

$$\frac{\partial \mathrm{I}(p_t; S)}{\partial w_{ik}} = r_i^* p_k \tag{1.46}$$

Where

$$r_j = e\, f'\left(\sum v_{ji} q_i\right) \tag{1.47}$$

$$r_i^* = \sum_j w_{ji} f'\left(\sum_k w_{ik} p_k\right) r_j \tag{1.48}$$

are the learning signals of **jth** and **ith** elements of the output and hidden units, respectively. Since the learning signals r_j^* are determined by **back propagation** the error signals, this is called the **back propagation method [**D. E. Rumlhart, G. E. Hinton, and R. J. Williams, 1986] , although there were many predecessors (e.g.,[S. Amari, 1967], [P. Werbos, 1974], [D. B. Parker, 1982]).

3. Details on Back propagation of Recurrent Networks: see appendix **C** .

3

3.1 Introduction

Based on the theoretical results obtained in chapter two, the following illustrations and computational algorithm have been developed. Two illustrations concerning the nonlinear Pendulum system and 3-dimentional nonlinear system have also been simulated. The numerical control results are shown in Tables and Figures, supported by some useful comments.

3.2 Computational Algorithm

Based on the proof of the main problem (theorem 2.7.1, theorem 2.7.2) of chapter two, the following step-by-step computational algorithm have been proposed.

Consider the general problem

$$\dot{x} = \mathrm{A}x + \mathrm{B}f(x)x + \mathrm{B}g(x)u(t) \qquad (3.1)$$

where $x \in R^n$, and the smooth functions $f \in R^{m\times n}$, $g \in R^{m\times m}$ such that $g(x) \neq 0, \forall x$, and $g_{ij}(x) \neq 0, \quad \forall i = 1,\ldots,m; \forall j = 1,\ldots,m$, $u \in R^m$ is the control, $\mathrm{A} \in R^{nxn}$, $\mathrm{B} \in R^{nxm}$ are constant matrices.

To design a nonlinear neuro-controller (a nonlinear controller) so that the nonlinear uncertain dynamic control system is stabilized, the following step-by-step procedure have been adapted.

Step (1): To test the controllability of (3.1), the pair (A,B) should be controllable and have $\rho(\mathrm{A},\mathrm{B}) \equiv [\mathrm{B}, \quad \mathrm{AB}, \quad \ldots, \quad \mathrm{A}^{n-1}\mathrm{B}]$ is of rank n.

Step (2): check the boundnees of the given function $g(x)$ such that $g(x) \neq 0, \forall x, g_{ij}(x) \neq 0, \ \forall i = 1,\ldots,m; \forall j = 1,\ldots,m$, And

$$\|g(x)\| > \varepsilon, \text{ where } \varepsilon > 0, \varepsilon \in R$$

Step (3): the uncertain function $f(x)$ where its estimator $\hat{f}(x)$, and a functional error estimation $\tilde{f}(x)$, such that

$$\left\| f(x) - \hat{f}(x) \right\| = \left\| \tilde{f}(x) \right\| \le f_m(x)$$

For some known function bound $f_m(x)$

Step(4): set that the desired state vector $y_d(t) \in R^n$, $y_d(t) = [y_d, \ \dot{y}_d, \ \ldots, \ y_d^{(n-1)}]^T$, and to check that $y_d(t)$ is bounded and continuous.

Step (5): using the result of step (1), to transform the system in (3.1) into block companion form by assuming $z = T\,x$, where $z = [z_1 \ \ \ldots \ \ z_n]^T$, Such that T be any **nxn** invertible transformation matrix where $T\,\mathrm{B} = \begin{bmatrix} 0 \\ \mathrm{I}_m \end{bmatrix}$, I_m is a unit matrix of dimension m, T can be chosen such that $T^{-1} = [L : \mathrm{B}]$, where L is selected such that the inverse exists (see remark (2.6.2.1), remark (2.6.2.2) and example (2.6.2.1)), so that the system in (3.1) can be rewritten as

$$\dot{z} = \overline{\mathrm{A}}\, z + \overline{\mathrm{B}}\, h(z) + \overline{\mathrm{B}}\, g\, u \qquad (3.2)$$

Where $h(z) = f\left(T^{-1} z\right)$ be a known functions, $g \in R^{m \times m}$ be a known function and $g(z) \neq 0, \forall z$, $u(t) \in R^m$ be the control, and assume that

$$\overline{\mathrm{A}} = T\,\mathrm{A}\,T^{-1} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_n \end{bmatrix}, \quad \overline{\mathrm{B}} = T\,\mathrm{B} = [\mathrm{O}_m \quad \mathrm{O}_m \quad \cdots \quad \mathrm{I}_m]^T$$

where $\overline{\mathrm{A}}$ be **nxn** matrix, $\overline{\mathrm{B}}$ be **nxm** matrix and $\mathrm{O}_m, \mathrm{I}_m$, be the **Zero** matrix and **Unit** matrix of dimension **m** respectively, $\alpha_i, i = 1, \ldots, n$ are constants. So that the system (3.2) can be rewritten as

$$\Rightarrow \quad \dot{z} = T\,\mathrm{A}\,T^{-1} z + T\,\mathrm{B}\,h(z) + T\,\mathrm{B}\,g\,u(t)$$

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \vdots \\ \dot{z}_{n-1} \\ \dot{z}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-1} \\ z_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ h_m(z) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ g_m u(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \alpha_1 z_1 + \alpha_2 z_2 + \ldots + \alpha_n z_n \end{bmatrix}$$

where $g_m = \begin{bmatrix} g_{m1} & g_{m2} & \cdots & g_{mm} \end{bmatrix}$, $u = \begin{bmatrix} u_1 & u_2 & \cdots & u_m \end{bmatrix}^T$.

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \vdots \\ \dot{z}_{n-1} \\ \dot{z}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-1} \\ z_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ h_m(z) + \alpha_1 z_1 + \alpha_2 z_2 + \ldots + \alpha_n z_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ g_m u(t) \end{bmatrix}$$

Let

$$F(z) = h_m(z) + \alpha_1 z_1 + \alpha_2 z_2 + \ldots + \alpha_n z_n \tag{3.3}$$

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \vdots \\ \dot{z}_{n-1} \\ \dot{z}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-1} \\ z_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ F(z) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ g_m u(t) \end{bmatrix}$$

Hence

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= z_3 \\ &\vdots \\ \dot{z}_n &= g_m u(t) + F(z) \\ y &= z_1 \end{aligned} \tag{3.4}$$

Step (6): Define the filtered tracking error $r(t)$ as follows

$$r_z = K^T e_z$$

Where $e_z = z - y_d$, and Chose vector $K = \begin{bmatrix} k_1 & k_2 & \ldots & k_{n-1} & 1 \end{bmatrix}^T$.

And find norm r_z as

$$\|r_z\| = \sqrt{\sum |r_z(t)|^2}$$

Such that the derivative of the filtered tracking error

$$\dot{r}_z = K^T \dot{e}_z \; and \; from \, \dot{e}_z = \dot{z} - \dot{y}_d$$

We have that

$$\dot{r} = F(z) + g_m u(t) + \overline{Y}_d, \quad where \, \overline{Y}_d = -y_{dn} + \sum_{i=1}^{n-1} k_i e_{i+1}$$

Step (7): Define the control of the system (3.4) $u(t)$ as follows

$$u(t) = u_1(t) + u_2(t)$$

Where $u_1(t)$be the control input of the system (3.4), such that

$$u_1 = W - \hat{u}_2$$

Where W is the tracking control law and $\hat{u}_2$ is the approximate function of the $u_2(t)$ whose is the nonlinear control of the system (3.4) which be approximated by neural network.

Step (8): To find the tracking control law W using the derivative of the filtered tracking error $r(t)$ and substituting $u(t) = u_i (u_1(t) + u_2(t))$ where $u_i = \frac{1}{g_{mi}}, \forall i = 1, \ldots, m$ *such that* $u_i \neq 0, \forall i = 1,2, \ldots, m$ and $u_1 = W - \hat{u}_2$ we will get the following

$$W = \frac{1}{g_m u_i} \left[- \hat{F}(z) - Y_d - k_v r + \alpha \right]$$

Where we must chose

1. where $\hat{F}$ be the fixed approximation of the functional F.
2. k_v is the feedback gain.
3. that α be the robust term chosen for the disturbance rejection which can be defined as

$$\alpha(t) = -f_m(z)\,sign(r)$$

where

$$sign(r) = \begin{cases} 1 & r > 0 \\ 0 & r \le 0 \end{cases}$$

where $sign(\bullet)$ is the standard sign function.

Step (9) (Neural Network): Consider the two-layered feed forward neural network have an **2n** input $x_{NN} = \left[y_d, e\right]^T$, whose value are real number and the hidden layer has **m** neurons, the output layer has **L** neurons.

so that the first layer feed forward neural network have

1. An input $x_{NN} = \left[y_d, e\right]^T$.

2. The bias b we will chose it randomly and have an input value $x_0 = -1$.

3. The weights matrix V which we chose it randomly too.

4. We chosing the **Log-sigmoid** as the transfer function (activation function) f to the first layer and to the hidden layer too (which be explained in chapter one) because it is differentiable and commonly used in many methods such as Back propagation method

$$f(\lambda) = \frac{1}{1+\exp(-\lambda)},\ where\ \lambda_i = \sum_{j=1}^{2n} V_{ij}\, x_j,\ i = 1,2,\ldots,m$$

5. The output of the first layer q can be found by

$$q_i(t) = f\left(\sum_{j=1}^{2n} V_{ij}(t)\, x_j(t) \right) + b_i \quad where\ i = 1,\ldots,m$$

And the hidden layer (second layer) have

1. The output q of the input layer (first layer) will be the input to the hidden layer.

2. The ideal neural network weight matrix of the hidden layer w we chose it randomly and it will be bounded by

$\|w\| \le w_m$

With w_m known bounds.

3. The estimate neural network weight of the ideal weight w can be provided by the neural network tuning algorithm

$$\dot{\hat{w}} = S\, f\left(V^T x_{NN}\right) r\, g_m(z) - kS\|r\|\hat{w}$$

Where S : be any constant representing the learning rates of the neural network.

k : be a small positive parameter.

4. The neural network weights approximation error as follows

$$\tilde{w} = w - \hat{w}$$

Where the neural network weights approximation error $\tilde{w}$ and the filtered tracking error are bounded by the proof of theorem (1) of chapter two as follows

$$\|\tilde{w}\|_F \ge \left(\frac{\sqrt{\frac{k}{4} w_m^2 + \varepsilon_n} + \frac{1}{2} w_m}{k} \right)^{\frac{1}{2}} \quad , where \ \|\tilde{w}\|_F = \sqrt{\sum_{ij} w_{ij}^2}$$

And

$$\|r\| \ge \frac{\frac{k}{4} w_m^2 + \varepsilon_n}{k_{v\min}}$$

5. The output of the hidden layer can be written as

$$y_k(t) = f\left(\sum_{i=1}^{m} \tilde{w}_{ki}(t)\, q_i(t) \right), \quad where\ k = 1, \ldots L$$

such that the output of the neural network y_k is the approximation function $\hat{u}_2(t)$ of the nonlinear control $u_2(t)$ of the system (3.4), hence

$$\hat{u}_2 \equiv y_k$$

Step (10): Now we can find the control input $u_1(t)$ by

$$u_1(t) = W - \hat{u}_2(t)$$

such that the control inputs to the system (3.4) and all steps from (step (9)) will be repeated in close loop until the nonlinear system in (3.4) approaches zero (until the nonlinear system in (3.4) will be stable).

The solution of system (3.1), (3.4) and the neural network tuning weights algorithm $\left(\dot{\hat{w}}(t)\right)$ can be obtained using any suitable numerical method. In this thesis a Rung-Kutta method of order 4, using MATLAB version 6.5 and personal computer's (PIT 4), have been adapted. The simulation have implemented step-by-step on time interval $t \in [0, \tau]$ and step size $h = \frac{\tau}{N_0}$, N_0 is suitable positive natural number. The random number of w, v are initially selected to belong to $(-1, 1)$. The initial conditions of $\hat{w}$ are randomly selected, the initial selection of dynamical system are suitable selected or given. The adjusting of the weights $\tilde{w}$ are adapted for each step size h. The artificial neural network are designed for each discrete system (for each step time) and collected as a whole problem for all time interval $t \in [0, \tau]$ and the number of inputs, outputs, layer, notes are dependent on number of divided time interval.

3.3.1 Application (Nonlinear System of Pendulum Type):

For solving nonlinear system of " Pendulum Type " and to verify the effectiveness of the neural network we will show down some successive steps for this purpose.

Consider the nonlinear system of Pendulum type as

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -5x_1^3 - 2x_2 + u \qquad (3.5)$$
$$y = x_1$$

Where u is the control of the system, which we defined it in the algorithm above step (8) as

$$u = u_1 + u_2$$

where $u_1 \equiv$ is the control input to the system above and

$u_2 \equiv$ is the nonlinear control which we fined it by the general neural network approximation property.

Then we have

$$f(x) = -5x_1^3 - 2x_2 \quad , g(x) = 1. \qquad (3.6)$$

Step (1): since the system in (3.5) is already in companion form, so set $T = \mathrm{I}$ and go to the next step.

Step (2): The numerical solution haven been obtained using 4^{th} order Rung-Kutta explicit, for the time interval $t = [0, 100]$ and step size $h = \frac{100}{N_0}$ for $N_0 = 20$ on MATLAB version (6.5), and the following symbols have been used

$$x_1(t) = x_1(t = ih) \equiv x_1(ih) \equiv x_1(i)$$
$$x_2(t) = x_2(t = ih) \equiv x_2(ih) \equiv x_2(i)$$

where

$i \equiv$ stands for number of dividing time interval.

Tack the initial condition of the system in (3.5) as $x_1(0) = 0$, $x_2(0) = 1$. We get the following

Results of uncontrolled system $(u \equiv 0)$ (3.25) without using neural network at time $t \in [0,100]$ and step size $h = 5$

t	i	$x_1(i)$	$x_2(i)$
0	1	0	1.000
5	2	0.1831	-0.0191
10	3	0.1312	-0.0061
15	4	0.1087	-0.0034
20	5	0.0950	-0.0022
25	6	0.0856	-0.0016
30	7	0.0785	-0.0012
35	8	0.0730	-0.0010
40	9	0.0685	-0.0008
45	10	0.0647	-0.0007
50	11	0.0615	-0.0006
55	12	0.0588	-0.0005
60	13	0.0564	-0.0005
65	14	0.0542	-0.0004
70	15	0.0523	-0.0004
75	16	0.0506	-0.0003
80	17	0.0490	-0.0003
85	18	0.0476	-0.0003
90	19	0.0463	-0.0003
95	20	0.0451	-0.0002
100	21	0.0440	-0.0002

Table (3.1)

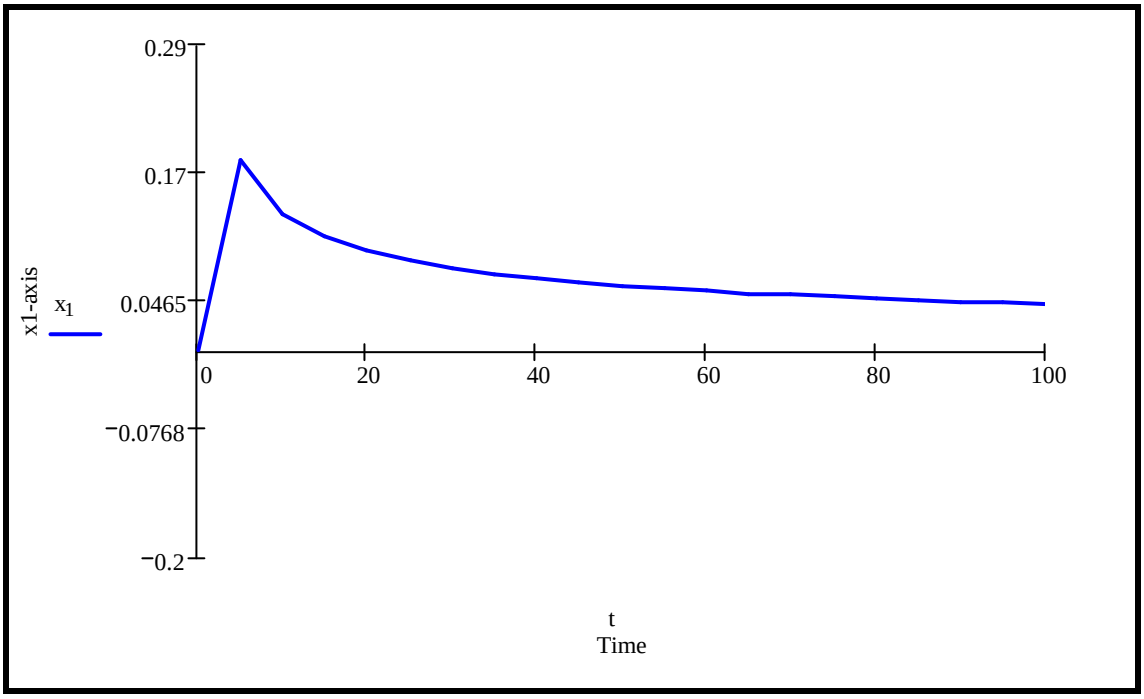


Figure (3.1)

The solution of the system (3.5) ($x_1(t), i \in [0,100]$) without using neural network

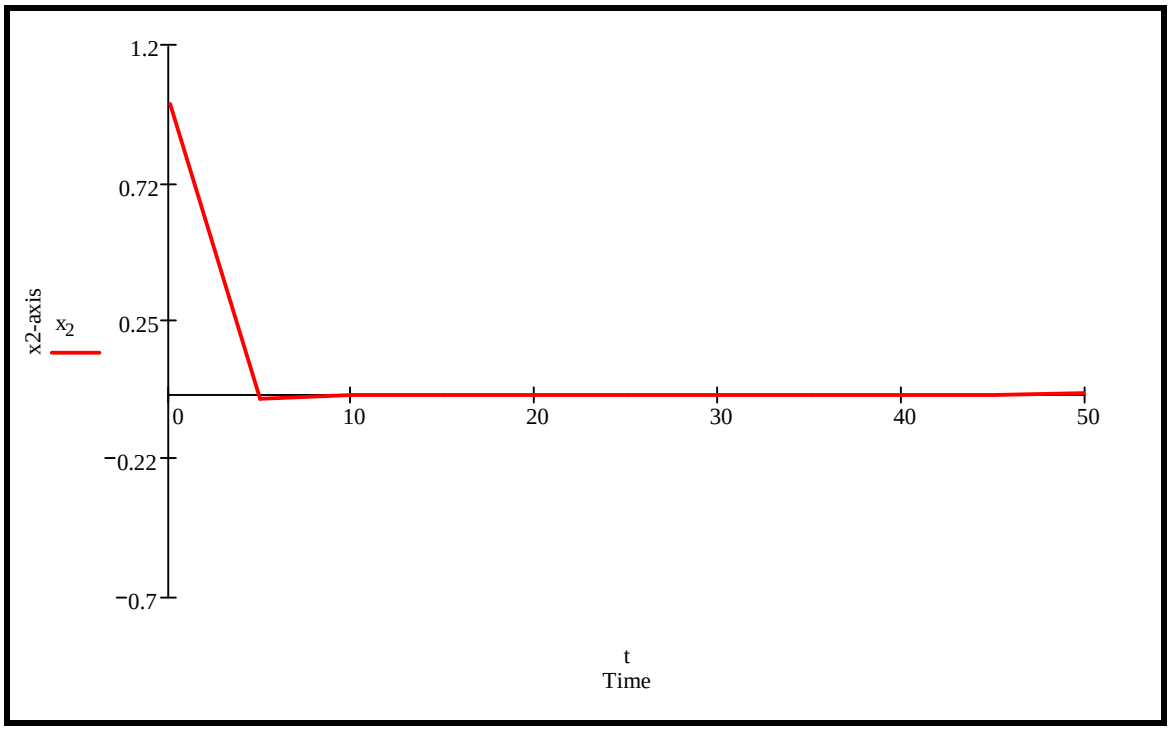


Figure (3.2)

Solution of the system (3.5) $x_2(t), t \in [0,100]$ without using neural network

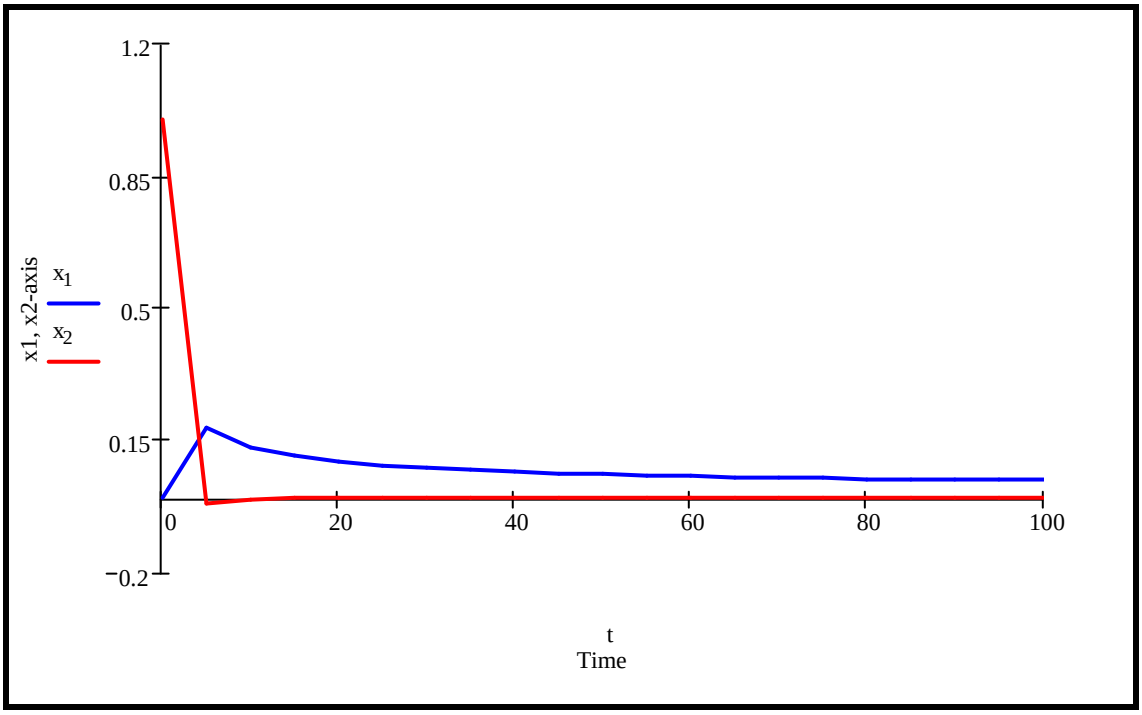


Figure (3.3)

$x_1(t), x_2(t), t \in (0,100)$ without using neural network

Step(3):Consider the desired trajectory are given by

$$x_{d1}(t) = \sin(t) = \sin(t = ih) = \sin(ih)$$

$$x_{d2}(t) = \cos(t) = \cos(t = ih) = \cos(ih)$$

where $y_d = (x_{d1}(t), x_{d2}(t))$, with time $t \in [0,100]$, and step size $h = 5$.

Desired Trajectory with $t \in [0,100]$ and step size $h = 5$

t	i	$x_{d1}(i) = \sin(ih)$	$x_{d2}(i) = \cos(ih)$
0	1	0	1.0000
5	2	-0.9589	0.2837
10	3	-0.5440	-0.8391
15	4	0.6503	-0.7597
20	5	0.9129	0.4081
25	6	-0.1324	0.9912
30	7	-0.9880	0.1543
35	8	-0.4282	-0.9037
40	9	0.7451	-0.6669
45	10	0.8509	0.5253
50	11	-0.2624	0.9650
55	12	-0.9998	0.0221
60	13	-0.3048	-0.9524
65	14	0.8268	-0.5625
70	15	0.7739	0.6333
75	16	-0.3878	0.9218
80	17	-0.9939	-0.1104
85	18	-0.1761	-0.9844
90	19	0.8940	-0.4481
95	20	0.6833	0.7302
100	21	-0.5064	0.8623

Table (3.2)

Step (4): Find the tracking errors $e_1(t), e_2(t)$ at time interval $t=[0,100]$such that

$$e_1(t)=x_1(t)-\sin(t) \quad \Rightarrow e_1(t)=e_1(t=ih)\equiv e_1(ih)\equiv e_1(i)$$
$$\Rightarrow e_1(i)=x_1(i)-\sin(ih)$$

$$e_2(t)=x_2(t)-\cos(t) \quad \Rightarrow e_2(t)=e_2(t=ih)\equiv e_2(ih)\equiv e_2(i)$$
$$\Rightarrow e_2(i)=x_2(i)-\cos(ih)$$

The evaluation of the tracking error $(e_1(i), e_2(i))$ at time interval $t=[0,100]$,and step size $h=5$ are as follows

Errors $(e_1(t), e_2(t))$ of the System (3.5) Without Using Neural Network

t	i	$e_1(i)$	$e_2(i)$
0	1	0	0
5	2	1.1420	-0.3028
10	3	0.6752	0.8330
15	4	-0.5416	0.7563
20	5	-0.8179	-0.4103
25	6	0.2179	-0.9928
30	7	1.0665	-0.1555
35	8	0.5011	0.9027
40	9	-0.6767	0.6661
45	10	-0.7862	-0.5260
50	11	0.3239	-0.9656
55	12	1.0585	-0.0226
60	13	0.3612	0.9520
65	14	-0.7726	0.5621
70	15	-0.7216	-0.6337
75	16	0.4384	-0.9221
80	17	1.0429	0.1101
85	18	0.2237	0.9841
90	19	-0.8477	0.4478
95	20	-0.6382	-0.7304
100	21	0.5504	-0.8625

Table (3.3)

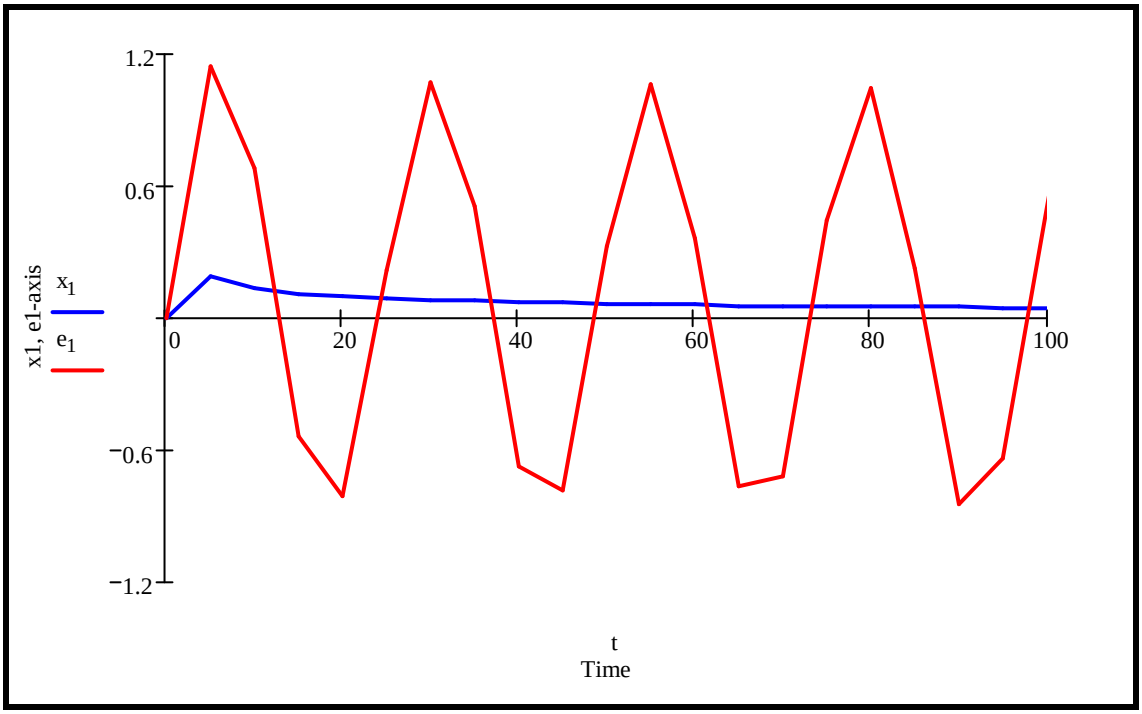


Figure (3.4)

The output of the system (3.5) and the error without using neural network

Step (5): Find the filtered tracking error $r(t)$ as $r(t)=K\,e(t),\ K=[2,1]$ and $e(t)=\lfloor e_1(t) \quad e_2(t)\rfloor^T$, such that $r(t)$ at time interval $t=[0,100]$,and step size $h=5$, $r(t)=r(t=ih)\equiv r(i)=2(x_1(i)+\sin(ih))+(x_2(i)-\cos(ih))$

Filtered Tracking error with $t \in [0,100]$, $h = 5$

t	i	$r(i)$
0	1	0
5	2	1.9811
10	3	2.1834
15	4	-0.3269
20	5	-2.0462
25	6	-0.5570
30	7	1.9776
35	8	1.9050
40	9	-0.6872
45	10	-2.0984
50	11	-0.3178
55	12	2.0944
60	13	1.6743
65	14	-0.9832
70	15	-2.0768
75	16	-0.0453
80	17	2.1959
85	18	1.4315
90	19	-1.2476
95	20	-2.0067
100	21	0.2382

Table (3.4)

The norm of $r(t)$ is approximated using the numerical results which is depends on the simulation such that the norm $r(t) = \|r(t)\| = 7.1271$.

Step (6): Chose the tracking control law of the system (3.5) by using the derivative of the filter tracking error and by substituting that $u_1(t)=W(t)-\hat{u}_2(t)$ so that we can chose W as follows

$$W=\frac{1}{g(x)}\left(-\hat{F}(x)-\overline{\mathrm{Y}}_d+\alpha+k_v r\right)$$

Where we can find $\overline{\mathrm{Y}}_d$ from the algorithm step (6)above as

$$\overline{\mathrm{Y}}_d=\sin(t)-2(\cos(t))+2x_2(t)$$

suppose $\hat{F}(x)=-2x_1^2-2x_2$, $k_v=1$. And Consider the robust term have the form $\alpha=5x_1^3-2x_1^2$.

so that the tracking control law $W(t)=W(t=ih)=W(ih)\equiv W(i)$ where $i\equiv$ stands for number of dividing time interval. Where the behavior of nonlinear part of tracking control law W is shown in the following figure too.

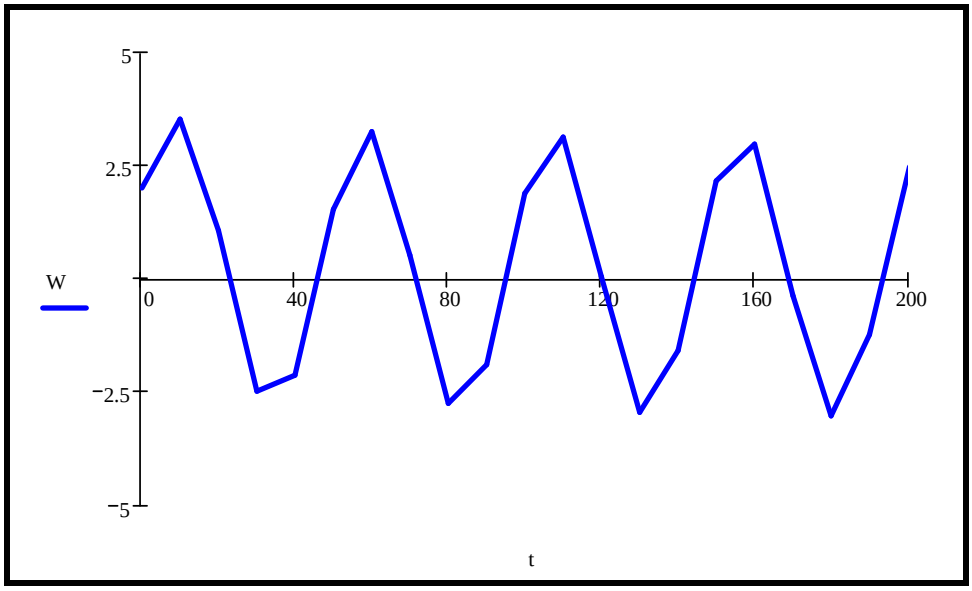


Figure (3.5)

The behavior of the tracking control law W

And the simulated on time interval $t\in[0,100]$ for step size $h=5$ as follows

The Tracking Control Law (W) results with $t \in [0,100]$ for $h=5$

t	i	Tracking control law $(W(i))$
0	1	2.0000
5	2	3.5381
10	3	1.0605
15	4	-2.4902
20	5	-2.1387
25	6	1.5609
30	7	3.2765
35	8	0.5277
40	9	-2.7646
45	10	-1.8973
50	11	1.8757
55	12	3.1394
60	13	0.0752
65	14	-2.9341
70	15	-1.5834
75	16	2.1866
80	17	2.9696
85	18	-0.3607
90	19	-3.0372
95	20	-1.2292
100	21	2.4696

Table (3.5)

Step (7): Since the control input as we defined above is

$$u_1(t) = W(t) - \hat{u}_2(t)$$

Where $W(t) \equiv$ be the tracking control law

$\hat{u}_2(t) \equiv$ be the approximate function for the nonlinear control $u_2(t)$.

such that we will going to find $\hat{u}_2(t)$ by using the general neural network approximation property as follows.

Step (8) (Neural Network): Consider the two-Layered feed forward neural network have four, ten, one neurons at the input, hidden and output layers respectively.

1. The inputs for the first layer are $x_{NN} = \begin{bmatrix} x_{d1} & x_{d2} & e_1 & e_2 \end{bmatrix}$.

For $t \in [0,100]$ using step size $h = 5$.

Remark:

Since the simulation are implemented step-by-step depending on the divided time interval, the number of input to artificial neural network as well as the number of outputs are depending on the number of divided time interval and this is the very important problem appearing in continuous neural network.

Such that the inputs of the network have the following results

$$x_{NN} = \left[\sin(t = ih),\quad \cos(t = ih),\quad e_1(t = ih),\quad e_2(t = ih)\right]$$

$$\Rightarrow \quad = \left[\sin(ih),\quad \cos(ih),\quad e_1(i)\quad e_2(i)\right]$$

Such that, the simulation of the neural network inputs will be shown in the table (3.6) with time interval $t \in [0,100]$ using step size $h = 5$.

Inputs to the First Layer Neural Network x_{NN} with $t \in [0,100]$

and step size $h = 5$

t	i	$x_{d1}(i)$	$x_{d2}(i)$	$e_1(i)$	$e_2(i)$
0	1	0	1.0000	0	0
5	2	-0.9589	0.2837	1.1420	-0.3028
10	3	-0.5440	-0.8391	0.6752	0.8330
15	4	0.6503	-0.7597	-0.5416	0.7563
20	5	0.9129	0.4081	-0.8179	-0.4103
25	6	-0.1324	0.9912	0.2179	-0.9928
30	7	-0.9880	0.1543	1.0665	-0.1555
35	8	-0.4282	-0.9037	0.5011	0.9027
40	9	0.7451	-0.6669	-0.6767	0.6661
45	10	0.8509	0.5253	-0.7862	-0.5260
50	11	-0.2624	0.9650	0.3239	-0.9656
55	12	-0.9998	0.0221	1.0585	-0.0226
60	13	-0.3048	-0.9524	0.3612	0.9520
65	14	0.8268	-0.5625	-0.7726	0.5621
70	15	0.7739	0.6333	-0.7216	-0.6337
75	16	-0.3878	0.9218	0.4384	-0.9221
80	17	-0.9939	-0.1104	1.0429	0.1101
85	18	-0.1761	-0.9844	0.2237	0.9841
90	19	0.8940	-0.4481	-0.8477	0.4478
95	20	0.6833	0.7302	-0.6382	-0.7304
100	21	-0.5064	0.8623	0.5504	-0.8625

Table (3.6)

2. Let the bias b are selected randomly distributed between **-15** and **+1**5, where $b(j),\ j=1,2,\ldots,10$, $j\equiv$ depends on the number of the neuron in the hidden layer $(j=10)$in this application, the bias will be chosen as follows:

The bias b to the First Layer Feed forward Neural Network

Bias b	١	٠	-1	٠	٠	٣	-4	-5	١	٨

Table (3.7)

with input to the bias has the value $x_0=-1$.

3. Consider the first layer feed forward weights $V_{ij}\ where i=1,\ldots 10,\ j=1,\ldots 4$,(in this application) where $i,j\equiv$ representing the number of neurons on the hidden layer and the input layer respectively. Such that V_{ij} will be chosen randomly distributed between -1 and +1 as

The First Layer Feed forward Neural Network Weights V

i	The first layer neural network V_{ij}			
1	-1	1	0	1
2	1	-1	-1	1
3	1	0	0	-1
4	-1	-1	-1	0
5	-1	-1	0	-1
6	0	-1	0	0
7	-1	0	0	0
8	0	1	0	1
9	-1	1	-1	0
10	1	1	1	-1

Table (3.8)

4. We chose the **Log-sigmoid** transfer function f (activation function) (which be explained in chapter one) because it is differentiable and commonly used in many methods such as Back propagation method

$$f(\lambda)=\frac{1}{1+\exp(-\lambda)}, \text{ where } \lambda_i = \sum_{j=1}^{4} V_{ij}\, x_j, i=1,\ldots,10$$

noted that $,\lambda_i(t)\equiv\lambda_i(t_j)\equiv\lambda_i(jh)\equiv\lambda_i(j),$ $where\, i=1,2,\ldots,10$, $i\equiv$ stand for number of evaluates in value $\lambda(t)$, $j\equiv$ stand for number of divided time interval. Such that the result of λ on the time interval $t=[0,100]$ with step size $h=5$, the evaluation are as shown in table (3.9)

The values of the sigmoid function $f(\lambda)$ at $t \in [0,100]$ and step size $h = 5$

t	j	$\lambda_1(j)$	$\lambda_2(j)$	$\lambda_3(j)$	$\lambda_4(j)$	$\lambda_5(j)$	$\lambda_6(j)$	$\lambda_7(j)$	$\lambda_8(j)$	$\lambda_9(j)$	$\lambda_{10}(j)$
0	1	3.00	4.00	2.00	3.00	3.00	-6.00	2.00	6.00	-6.00	-4.00
5	2	0.596	4.413	1.041	4.67	4.372	-6.23	3.55	3.182	-6.302	-3.403
10	3	2.613	6.672	1.456	5.381	6.216	-8.540	5.347	2.941	-5.167	-1.198
15	4	4.948	6.516	2.650	4.109	4.865	-8.407	3.974	5.432	-5.243	-1.375
20	5	4.320	4.181	2.912	2.679	2.268	-6.086	1.363	7.139	-6.410	-3.723
25	6	1.656	3.016	1.867	3.141	2.148	-4.930	1.233	5.640	-6.992	-4.898
30	7	0.789	4.690	1.012	4.833	4.678	-6.611	3.756	3.099	-6.155	-3.231
35	8	2.973	6.806	1.571	5.331	6.234	-8.733	5.307	3.167	-5.097	-1.1206
40	9	5.087	6.331	2.745	3.921	4.587	-8.264	3.656	5.754	-5.339	-1.598
45	10	4.111	3.948	2.850	2.623	2.097	-5.884	1.162	7.162	-6.526	-3.986
50	11	1.448	3.069	1.737	3.297	2.331	-5.008	1.394	5.378	-6.965	-4.869
55	12	0.919	4.955	1.002	4.977	4.955	-6.896	4.013	2.963	-6.022	-2.980
60	13	3.286	6.904	1.695	5.257	6.209	-8.848	5.265	3.381	-5.048	-1.039
65	14	5.161	6.124	2.826	3.735	4.297	-8.070	3.351	6.037	-5.437	-1.821
70	15	3.861	3.733	2.773	2.592	1.959	5.680	1.011	7.128	-6.633	-4.421
75	16	1.251	3.156	1.612	3.466	2.544	-5.105	1.594	5.095	-6.922	-4.793
80	17	1.073	5.220	1.006	5.104	5.214	-7.171	4.263	2.852	-5.889	-2.7305
85	18	3.584	6.968	1.823	5.160	6.144	-8.920	5.192	3.615	-5.015	-0.983
90	19	5.189	5.895	2.894	3.554	4.001	-7.849	3.048	6.293	-5.552	-2.057
95	20	3.591	3.539	2.683	2.586	1.856	-5.493	0.901	7.051	-6.730	-4.415
100	21	1.080	3.275	1.493	3.644	2.781	-5.231	1.825	4.805	-6.8625	-4.6809

Table (3.9)

5. the output of the first layer feed forward neural network is given by

$$q_i(t) = f\left(\sum_{j=1}^{4} V_{ij}(t)\, x_j(t) \right) + b_i \quad where\ i = 1,\ldots,10$$

The result is as follows in table (3.10). $t \in [0,100]$ and step size $h = 5$.

t	j	$q_1(j)$	$q_2(j)$	$q_3(j)$	$q_4(j)$	$q_5(j)$	$q_6(j)$	$q_7(j)$	$q_8(j)$	$q_9(j)$	$q_{10}(j)$
0	1	0.952	0.982	0.880	0.952	0.952	0.002	0.880	0.997	0.002	0.0180
5	2	0.644	0.988	0.739	0.990	0.965	0.002	0.9722	0.960	0.001	0.0322
10	3	0.931	0.998	0.810	0.995	0.998	0.002	0.995	0.949	0.005	0.2320
15	4	0.993	0.998	0.934	00.983	0.992	0.002	0.981	0.995	0.005	0.201
20	5	0.986	0.985	0.948	0.935	0.966	0.002	0.796	0.992	0.001	0.0023
25	6	0.839	0.9533	0.866	0.958	0.895	0.007	0.774	0.996	0.009	0.0074
30	7	0.687	0.990	0.733	0.992	0.890	0.001	0.9772	0.956	0.002	0.038
35	8	0.951	0.998	0.8280	0.995	0.980	0.002	0.995	0.959	0.0061	0.2459
40	9	0.993	0.9982	0.936	0.9806	0.0899	0.900	0.9748	0.0968	0.1048	01682
45	10	0.983	0.981	0.945	0.932	0.890	0.002	0.761	0.999	0.001	0.018
50	11	0.809	0.955	0.850	0.964	0.911	0.006	0.801	0.995	0.009	0.007
55	12	0.714	0.993	0.731	0.993	0.993	0.001	0.982	0.950	0.002	0.048
60	13	0.963	0.999	0.844	0.994	0.998	0.000	0.994	0.967	0.006	0.2613
65	14	0.994	0.997	0.944	0.976	0.986	0.000	0.966	0.997	0.004	0.139
70	15	0.979	0.976	0.941	0.930	0.876	0.003	0.733	0.992	0.001	0.0146
75	16	0.777	0.959	0.833	0.969	0.927	0.006	0.831	0.993	0.001	0.008
80	17	0.745	0.994	0.732	0.9940	0.9946	0.00	0.9861	0.9455	0.002	0.061
85	18	0.973	0.999	0.861	0.994	0.997	0.001	0.994	0.973	0.006	0.272
90	19	0.994	0.997	0.947	0.972	0.982	0.000	0.954	0.998	0.003	0.113
95	20	0.973	0.971	0.936	0.930	0.864	0.004	0.711	0.991	0.001	0.011
100	21	0.746	0.963	0.816	0.974	0.941	0.005	0.861	0.991	0.001	0.009

Table (3.10)

Now we must produce the hidden layer to the feed forward neural network which have the following:

1. the hidden layer (second layer) for the neural network have an input whose already the output of the first layer (input layer) $\langle q_i, where\, i = 1,\ldots,10 \rangle$

2.The neural network weights to the hidden layer (second layer) is $w(t) = \lfloor w_1(t) \quad w_2(t) \quad w_3(t) \quad \ldots \quad w_{10}(t) \rceil$ which we are chose it randomly between **-1** and **+1** as follows such that $w_i(t) \equiv w_i(t_j) \equiv w_i(jh) \equiv w_i(j), i = 1,2,\ldots,10$, $i \equiv$ stand for number of evaluates in value $w(t)$, $j \equiv$ stand for number of divided time interval. Such that the result of w on the time interval $t = [0,100]$ with step size $h = 5$, the evaluation are as shown in table (3.11).

The hidden layer feed forward neural network weights at $t=[0,100]$ with step size $h=5$

t	j	$w_1(j)$	$w_2(j)$	$w_3(j)$	$w_4(j)$	$w_5(j)$	$w_6(j)$	$w_7(j)$	$w_8(j)$	$w_9(j)$	$w_{10}(j)$
0	1	0	0	-1	0	0	-1	0	-1	-1	٠
5	2	0	0	0	0	0	0	0	0	-1	0
10	3	0	0	0	0	0	-1	0	0	-1	0
15	4	0	0	0	0	0	0	0	0	0	0
20	5	0	0	-1	0	0	0	-1	0	-1	0
25	6	0	0	-1	0	0	0	0	-1	-1	0
30	7	-1	0	0	0	-1	0	0	-1	0	0
35	8	0	0	0	0	-1	-1	0	-1	0	0
40	9	0	0	0	0	-1	-1	0	0	0	0
45	10	-1	0	0	0	-1	-1	0	0	0	0
50	11	0	0	0	0	-1	0	-1	0	0	-1
55	12	0	0	0	0	0	0	0	0	-1	0
60	13	0	0	-1	0	0	0	0	-1	-1	-1
65	14	0	0	-1	-1	-1	-1	0	-1	-1	0
70	15	0	0	-1	0	0	0	0	0	0	-1
75	16	-1	0	0	0	-1	0	0	0	-1	-1
80	17	0	-1	-1	0	0	-1	-1	0	0	-1
85	18	-1	-1	0	0	0	0	0	0	-1	0
90	19	0	0	0	0	0	0	-1	-1	-1	0
95	20	0	0	0	0	0	0	0	0	0	0
100	21	0	0	-1	0	0	-1	-1	0	0	-1

Table (3.11)

Where the norm of $w(t)$ is bounded by the value w_m $(\|w(t)\| \le w_m)$ (the norm approximated by matrix norm nested of continuous norm) suppose that $w_m = 7$, such that $\|w(t)\| = 4.6944$.

3.Consider the hidden layer estimated neural network weights can be found by solving the neural network tuning algorithm

$$\dot{\hat{w}} = S\, f\left(V^T x_{NN}\right) r\, g(x) - k\, S \|r\| \hat{w}$$

And suppose that $S = 5, k = 1.$ with initial conditions $\hat{w}_i(0), i = 1,2,\ldots 10$,

$i \equiv$ stand for number of evaluates in value $\hat{w}(t)$(number of the neurons in the hidden layer).

The initial conditions to the neural network tuning algorithm equation

$\hat{w}_1(0)$	$\hat{w}_2(0)$	$\hat{w}_3(0)$	$\hat{w}_4(0)$	$\hat{w}_5(0)$	$\hat{w}_6(0)$	$\hat{w}_7(0)$	$\hat{w}_8(0)$	$\hat{w}_9(0)$	$\hat{w}_{10}(0)$
0	0	0	0	0	0	0	0	0	0

Table (3.12)

The solution of the neural network estimated weights $\hat{w}(t) \equiv \hat{w}_i(t_j) \equiv \hat{w}_i(jh) \equiv \hat{w}_i(j), i = 1,2,\ldots,10$, $i \equiv$ stand for number of evaluates in value $\hat{w}(t)$, $j \equiv$ stand for number of divided time interval. Such that the result of $\hat{w}$ on the time interval $t \in [0,100]$ with step size $h = 5$, the solution are as shown in table (3.13).

Results of the neural network estimated weights $\hat{w}(t)\ t \in [0,100]$ with step size $h=5$

t	j	$\hat{w}_1(j)$	$\hat{w}_2(j)$	$\hat{w}_3(j)$	$\hat{w}_4(j)$	$\hat{w}_5(j)$	$\hat{w}_6(j)$	$\hat{w}_7(j)$	$\hat{w}_8(j)$	$\hat{w}_9(j)$	$\hat{w}_{10}(j)$
0	1	0	0	0	0	0	0	0	0	0	0
5	2	-0.10	-0.48	-0.08	-0.54	-0.59	0.002	-0.75	-0.37	-0.005	-0.220
10	3	-0.10	-0.48	-0.08	-0.54	-0.59	0.002	-0.75	-0.37	-0.005	-0.220
15	4	-0.10	-0.48	-0.08	-0.54	-0.59	0.002	-0.75	-0.37	-0.005	-0.220
20	5	-0.10	-0.48	-0.08	-0.54	-0.60	0.002	-0.75	-0.37	-0.005	-0.220
25	6	-0.10	-0.48	-0.08	-0.54	-0.59	0.002	-0.75	-0.36	-0.005	-0.220
30	7	-0.10	-0.48	-0.08	-0.54	-0.60	0.002	-0.76	-0.37	-0.005	-0.220
35	8	-0.10	-0.48	-0.08	-0.54	-0.60	0.002	-0.76	-0.37	-0.005	-0.220
40	9	-0.10	-0.48	-0.08	-0.54	-0.59	0.002	-0.75	-0.37	-0.005	-0.220
45	10	-0.10	-0.48	-0.08	-0.54	-0.60	0.002	-0.75	-0.37	-0.005	-0.220
50	11	-0.10	-0.48	-0.08	-0.54	-0.59	0.002	-0.75	-0.37	-0.005	-0.220
55	12	-0.10	-0.48	-0.08	-0.54	-0.59	0.002	-0.75	-0.37	-0.005	-0.220
60	13	-0.10	-0.48	-0.08	-0.54	-0.59	0.002	-0.75	-0.37	-0.005	-0.220
65	14	-0.10	-0.48	-0.08	-0.54	-0.59	0.002	-0.75	-0.37	-0.005	-0.220
70	15	-0.10	-0.48	-0.08	-0.54	-0.59	0.002	-0.75	-0.37	-0.005	-0.220
75	16	-0.10	-0.48	-0.08	-0.54	-0.60	0.002	-0.75	-0.37	-0.005	-0.220
80	17	-0.10	-0.48	-0.08	-0.54	-0.59	0.002	-0.75	-0.37	-0.005	-0.220
85	18	-0.10	-0.48	-0.08	-0.54	-0.60	0.002	-0.75	-0.37	-0.005	-0.220
90	19	-0.10	-0.48	-0.08	-0.54	-0.59	0.002	-0.75	-0.36	-0.005	-0.220
95	20	-0.10	-0.48	-0.08	-0.54	-0.60	0.002	-0.76	-0.37	-0.005	-0.220
100	21	-0.10	-0.48	-0.08	-0.54	-0.59	0.002	-0.75	-0.37	-0.005	-0.220

Table (3.13)

4. we can find the neural network weights approximation error $\tilde{w}(t)$ by the following $\tilde{w} = w(t) - \hat{w}(t), \tilde{w} \in R^{10} \Rightarrow\ \tilde{w}_i(t = jh) \equiv \tilde{w}_i(jh) \equiv \tilde{w}(j), i = 1,2,\ldots,10$

$i \equiv$ stand for number of evaluates in value $\tilde{w}(t)$, $j \equiv$ stand for number of divided time interval. Such that the result of $\tilde{w}$ on the time interval $t \in [0,100]$ with step size $h = 5$, the solution are as shown in table (3.14).

The neural network weights approximation error $\tilde{w}(t)$ $t \in [0,100]$, $h = 5$

t	j	$\hat{w}_1(j)$	$\hat{w}_2(j)$	$\hat{w}_3(j)$	$\hat{w}_4(j)$	$\hat{w}_5(j)$	$\hat{w}_6(j)$	$\hat{w}_7(j)$	$\hat{w}_8(j)$	$\hat{w}_9(j)$	$\hat{w}_{10}(j)$
0	1	0	0	-1.00	0	0	-1.00	0	-1.00	-1.00	0
5	2	0.104	0.481	0.088	0.549	0.599	-0.00	0.759	0.370	-0.99	0.220
10	3	0.104	0.481	0.089	0.541	0.599	-1.00	0.759	0.370	-0.99	0.220
15	4	0.104	0.480	0.088	0.540	0.599	-0.00	0.758	0.370	0.005	0.220
20	5	0.104	0.481	-0.91	0.541	0.600	-0.00	-0.24	0.370	-0.99	0.220
25	6	0.104	0.480	-0.91	0.540	0.599	-0.00	0.758	-0.630	-0.99	0.220
30	7	-0.89	0.481	0.089	0.541	-0.39	-0.00	0.760	-0.629	0.005	0.220
35	8	0.104	0.480	0.088	0.540	-0.40	-1.00	0.758	-0.630	0.005	0.220
40	9	0.104	0.481	0.089	0.541	-0.39	0.100	0.760	0.370	0.005	0.220
45	10	0.895	0.480	0.088	0.540	-0.40	-1.00	0.758	0.370	0.005	0.220
50	11	0.101	0.481	0.089	0.541	-0.40	-0.00	-0.24	0.370	0.005	-0.77
55	12	0.194	0.480	0.088	0.540	0.599	-0.00	0.759	0.370	-0.99	0.220
60	13	0.101	0.481	-0.91	0.540	0.599	-0.00	0.759	-0.629	-0.99	-0.77
65	14	0.104	0.481	-0.91	-0.45	-0.40	-1.00	0.759	-0.629	-0.99	0.220
70	15	-0.09	0.480	0.088	0.540	-0.40	-0.00	0.758	0.370	-0.99	0.220
75	16	0.104	0.481	-0.91	0.541	0.600	-0.00	0.759	0.3707	0.005	-0.77
80	17	-0.88	0.480	0.088	0.540	-0.40	-0.00	0.758	0.3699	-0.99	-0.77
85	18	0.104	-0.51	-0.91	0.541	0.600	-1.00	-0.23	0.370	0.005	-0.77
90	19	-0.89	-0.51	0.088	0.540	0.599	-0.00	0.758	0.369	-0.99	0.220
95	20	0.104	0.481	0.089	0.541	0.600	-0.00	-0.24	-0.629	-0.99	0.220
100	21	0.104	0.481	-0.91	0.541	0.599	-1.00	-0.24	0.370	0.005	-0.79

Table (3.14)

The adjusting of the neural network depending on the approximated weights $\tilde{w}(t)$ by the equality

$$\|\tilde{w}\|_F \geq \left(\frac{\sqrt{\frac{k}{4} w_m^2 + \varepsilon_n} + \frac{1}{2} w_m}{k} \right)^{\frac{1}{2}} \quad , where \ \|\tilde{w}\|_F = \sqrt{\sum_{ij} w_{ij}^2}$$

Where $k = 1, \varepsilon_n = 0.0001,$ and $w_m = 7$ as we defined it above. [if the above equality false we will restart the computations of the network from the begging (i.e. go back to neural network steps, step (2). And whenever the equality true we will stop the training of the network and compute the output of the network $y(t)$]. So that norm of $\tilde{w}(t) = 8.1828 > 2.6458$.

5. we will tack the Log-sigmoid transfer function (activation function) .

$$f(\lambda) = \frac{1}{1 + \exp(-\lambda)}$$

6. the output of the neural network is given by

$$y_k(t) = f\left(\sum_{i=1}^{10} \tilde{w}_{ki}(t)\, q_i(t) \right), \quad where \ k = 1, \ i = 1, \ldots 10.$$

Suppose that

$y_1(t) \equiv y_1(t_j) \equiv y_1(jh) \equiv y_1(j)$, $j \equiv$ stand for number of divided time interval. Such that the result of $y(t)$ on the time interval $t \in [0, 100]$ with step size $h = 5$, the solution are as shown in table (3.15).

The output of the neural network $t \in [0,100]$ with step size $h = 5$

t	j	$y_1(j)$
0	1	٠,٣٧٧٥
5	2	٠,٦٣١٨
10	3	٠,٦٦٨٥
15	4	٠,٢٨٥٣
20	5	٠,٦٨٨٢
25	6	٠,٤٨٧١
30	7	٠,٦٠٠٦
35	8	٠,٤٦٣٦
40	9	٠,٦٨٩٠
45	10	٠,٦٠١٨
50	11	٠,٧٦٨٣
55	12	٠,٤١١٨
60	13	٠,٦٥٤٦
65	14	٠,٢٨٢٩
70	15	٠,٢٣٦٥
75	16	٠,٥٦٨٦
80	17	٠,٧٣٠٥
85	18	٠,٥٣٤٢
90	19	٠,٢٦٨٥
95	20	٠,٥١٤٦
100	21	٠,٦١١٠

Table (3.15)

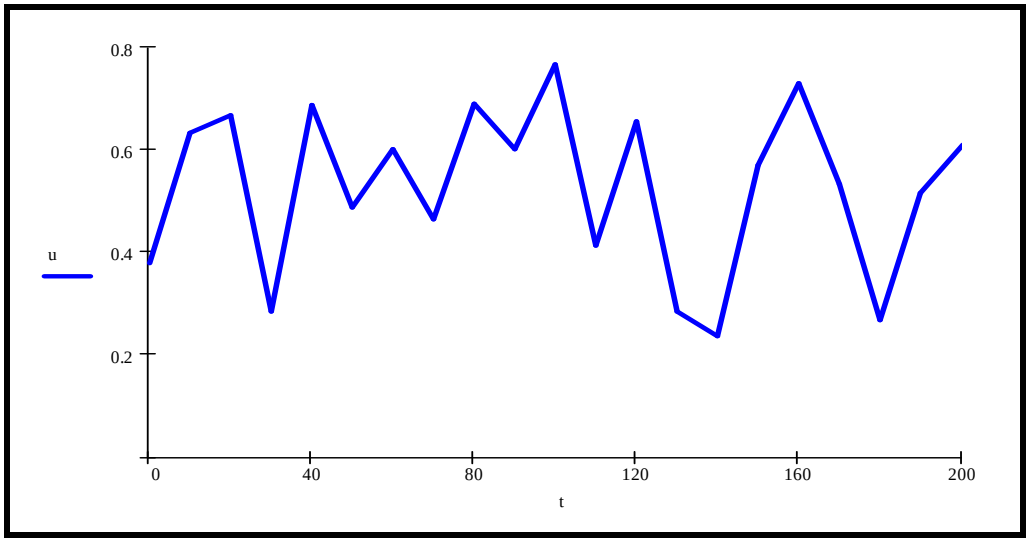


Figure (3.6)

The output of the neural network $\hat{u}_2$

Step(9): We can now find the control input of the system (3.5) with using $\hat{u}_2(t) = y_k(t)$,and we defined $u_1(t)$ in step (7) as $u_1(t) \equiv u_1(t_j) \equiv u_1(jh) \equiv u_1(j)$, $j \equiv$ stand for number of divided time interval. Such that the result of $u_1(t)$ on the time interval $t \in [0, 100]$ with step size $h = 5$, the solution are as shown in table (3.16). is given by

The control input of the system (3.5) $t \in [0,100]$ with step size $h = 5$

t	j	$u_1(j)$
0	1	1.6225
5	2	2.9036
10	3	0.3920
15	4	-2.7754
20	5	-2.8268
25	6	1.0738
30	7	2.6760
35	8	0.0641
40	9	-3.4536
45	10	-2.4991
50	11	1.1074
55	12	2.7277
60	13	-0.5794
65	14	-3.2170
70	15	-1.8198
75	16	1.6180
80	17	2.2392
85	18	-0.8949
90	19	-3.3057
95	20	-1.7438
100	21	١,٨٥٨٦

Table (3.16)

Step (10): display the control input $u_1(t)$ in the system (3.5) (step(1)) and find new the results and the error of the nonlinear system with using neural network.

Where

$$x_1(t) \equiv x_1(t = jh) \equiv x_1(jh) \equiv x_1(j)$$
$$\Rightarrow e_1(t) = \sin(t) \equiv \sin(t = jh) \equiv \sin(jh)$$

And

$$x_2(t) \equiv x_2(t = jh) \equiv x_2(jh) \equiv x_2(j)$$
$$\Rightarrow e_2(t) = \cos(t) \equiv \cos(t = jh) \equiv \cos(jh)$$

$j \equiv$ stand for number of divided time interval. Such that the result of $u_1(t)$ on the time interval $t \in [0,100]$ with step size $h = 5$, the solution are shown in table (3.17) as follows

The Results and the error of the system (3.5)with using neural network $t \in [0,100]$ with step size $h = 5$

t	j	$x_1(t)$	$e_1(t)$
0	1	0	0
5	2	1	0.0907
10	3	1.6225	1.4814
15	4	-1.0937	-0.3369
20	5	-19.3412	-18.3823
25	6	0.0003	0.0003
30	7	3.6154	3.6153
35	8	-0.0000	-0.0000
40	9	-2.3629	-2.3629
45	10	0.0000	0.0000
50	11	6.5962	6.5962
55	12	0.0000	0.0000
60	13	-1.4350	-1.4350
65	14	0.0000	0.0000
70	15	0.0000	0.0000
75	16	0.0000	0.0000
80	17	0.0000	0.0000
85	18	0.0000	0.0000
90	19	0.0000	0.0000
95	20	0.0000	0.0000
100	21	0.0000	0.0000

Table (3.17)

Now, the figures down shows the effectiveness of using neural network and who the system (3.5) will be stable as follows

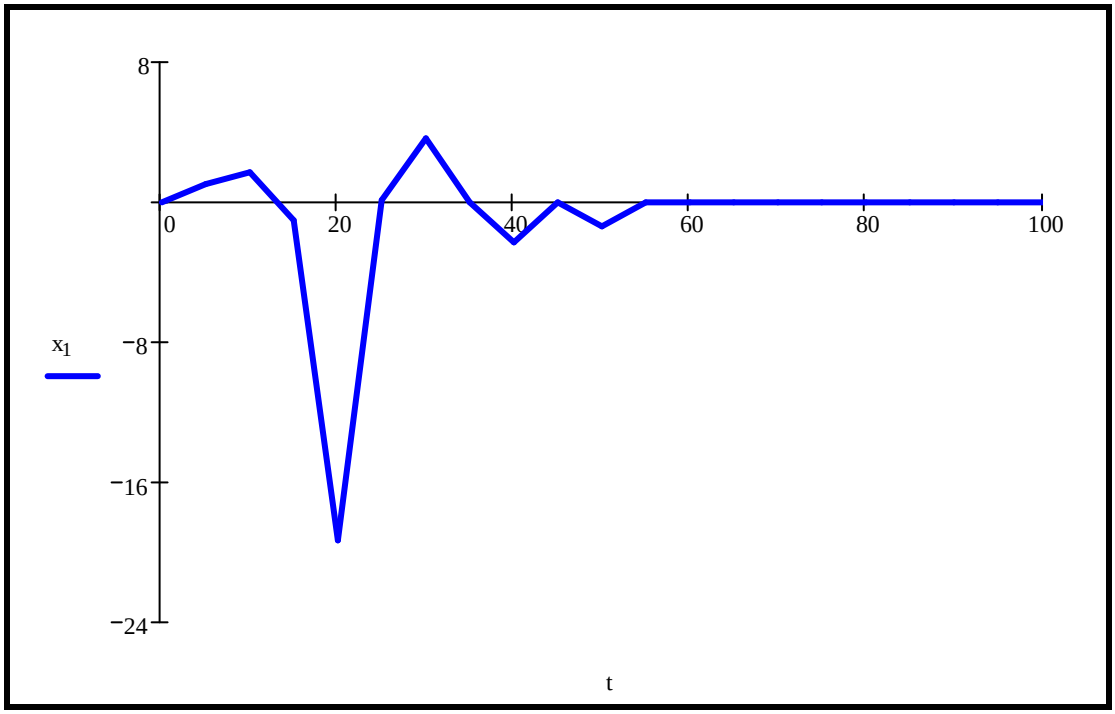


Figure (3.7)

The solution of the system (3.5)($x_1(t), t \in (0,100)$) with using neural network

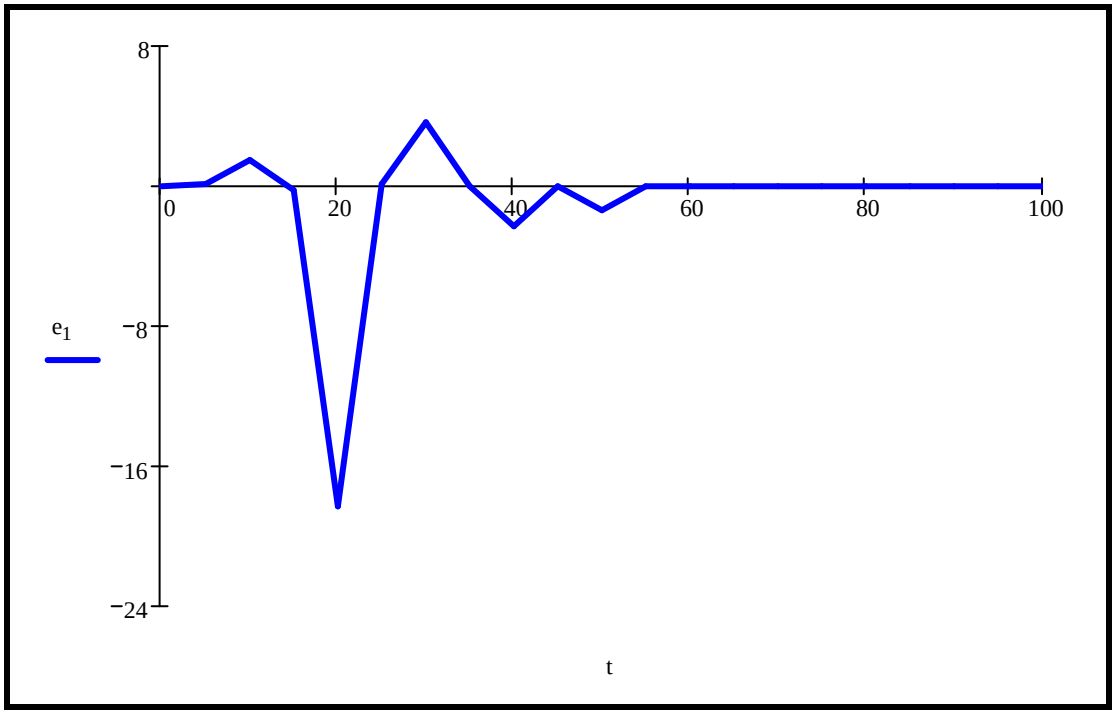
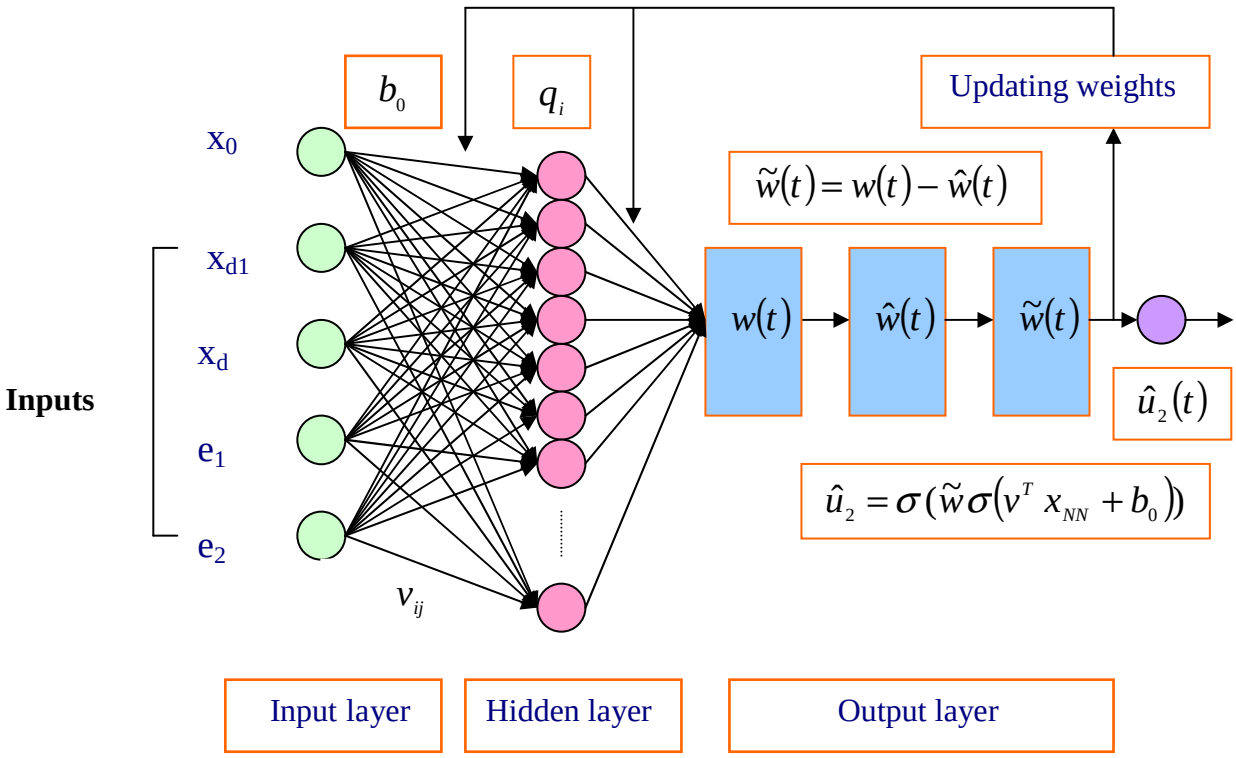


Figure (3.8)

The error of the system (3.5) with using neural network



where

$(the\,bias \equiv b_0 = [b_{1,0}, \ldots, b_{10,0}]),\ (the\,input\,to\,the\,neural\,network\,are \equiv x = [x_{d1}, x_{d2}, e_1, e_2])$

$(the\,weights\,in\,the\,1st\,layer)\, v_{ij}, i = 1, \ldots 10, (i \equiv the\,no.of\ neurons\,in\,the\,hidden\,layer)$

$j = 1, \ldots 4, (j \equiv the\,no.of\ neurons\,in\,the\,input\,layer)$

$$q_i = \sigma\left(\sum_{j=1}^{4} v_{ij}\ x_j\right),\ (q \equiv the\,output\,of\ the\ first\,layer)$$

$(the\,weights\,of\ the\,hidden\,layer) w_{ki}, where$

$k = 1, (k \equiv the\,no.of\ the\,neurons\,in\,the\,output\,layer),\ and \quad i = 1, \ldots, 10$

$(the\,estimated\,weights\ \hat{w}\ which\ found\,by\,solving \Rightarrow \dot{\hat{w}} = sqg - ks\|r\|\hat{w})$

$(the\,erorr\,weights\ \tilde{w} = w - \hat{w})$

$$\left(the\,output\,of\ the\,neural\,network\ \hat{u}_{2k} = \sigma\left(\sum_{i=1}^{10} \tilde{w}_{ki}\ q_i\right)\right)$$

Figure (3.9)

Architectural graph of two-layered neural network of application

(3.2.1)

3.2.2 APPLICATION:

Consider the nonlinear system

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{bmatrix}=\begin{bmatrix}-0.428 & -0.339 & 0\\ -2.939 & -1.011 & 0\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}+\begin{bmatrix}0.15\\ -1.011\\ 1\end{bmatrix}u(t)+\begin{bmatrix}0.15\\ -1.011\\ 0\end{bmatrix}*\left(-5x_1^3-2x_2-x_3\right) \quad (3.8)$$

where n=3, m=1, such that

$$x\in R^3 \Rightarrow x=\begin{bmatrix}x_1 & x_2 & x_3\end{bmatrix}^T$$

$$\mathrm{A}\in R^{3\times 3}, \mathrm{A}=\begin{bmatrix}-0.428 & -0.339 & 0\\ -2.939 & -1.011 & 0\\ 0 & 1 & 0\end{bmatrix}$$

$$\mathrm{B}\in R^{3\times 1}, \mathrm{B}=\begin{bmatrix}0.15\\ -1.011\\ 1\end{bmatrix}$$

$$g(x)\in R^{1\times 1} \Rightarrow g(x)=1$$

$$f(x)\in R^{1\times 1} \Rightarrow f(x)=-5x_1^3-2x_2-x_3$$

Step (1): we need to translate the system by define

$$z=T\,x, \text{ where } z=\begin{bmatrix}z_1 & z_2 & z_3\end{bmatrix}^T$$

Such that we have a new system

$$\dot{z}=T\mathrm{A}T^{-1}z+T\,\mathrm{B}\,g\,u(t)+T\,\mathrm{B}\,f$$

Where we consider that T is as follows

$$T=\begin{bmatrix}-3.1728 & -0.4707 & -1.1447\\ 2.7414 & 0.4067 & 0\\ -2.3688 & -1.3406 & 0\end{bmatrix}, T^{-1}=\begin{bmatrix}0.000 & 0.4944 & 0.15\\ 0 & -0.8735 & -1.011\\ -1.8736 & -1.011 & 1\end{bmatrix}$$

and we have that $T\,\mathrm{A}\,T^{-1}=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & 0.5636 & -1.4390\end{bmatrix}, T\,\mathrm{B}=\begin{bmatrix}0\\ 0\\ 1\end{bmatrix}$

The following transform system is derived.

$$\begin{aligned}\dot{z}_1 &= z_2\\ \dot{z}_2 &= z_3\\ \dot{z}_3 &= -5z_1^3 - 1.2264z_2 - 1.439z_3 + u(t)\\ y &= z_1\end{aligned} \tag{3.9}$$

Step (2) From the system above (3.9)we have

$g(z)=1$, and we have that $f(z)=-5z_1^3-1.2264z_2-1.439z_3$.

Step (3): The numerical solution haven been obtained using Rung-Kutta of order 4 , for time interval $t=[0,100]$ and step size $h=\dfrac{100}{N_0}$ for $N_0=20$ on MATLAB version (6.5), then the following symbols have been used

$$z_1(i)=z_1(t=ih)\equiv z_1(ih)\equiv z_1(i)$$
$$z_2(i)=z_2(t=ih)\equiv z_2(ih)\equiv z_2(i)$$
$$z_3(i)=z_3(t=ih)\equiv z_3(ih)\equiv z_3(i)$$

Where

$i\equiv$ stands for number of dividing time interval.

Tack the initial condition of the system in (3.9) as $z_1(0)=0,\ z_2(0)=0.2, z_3(0)=0.5$. We get the following evaluation

The solution of non controller $u = 0$ system (3.9) without using neural network $t \in [0, 100]$ with step size $h = 5$

t	i	$z_1(i)$	$z_2(i)$	$z_3(i)$
0	1	0	0.2000	0.5000
5	2	-0.2707	-0.1324	0.2075
10	3	-0.0282	0.000	-0.0178
15	4	-0.0423	0.0008	-0.0001
20	5	-0.0403	0.0003	0.000
25	6	-0.0390	0.0002	-0.000
30	7	-0.0378	0.0002	-0.000
35	8	-0.0368	0.0002	-0.000
40	9	-0.0358	0.0002	-0.000
45	10	-0.0349	0.0002	-0.000
50	11	-0.0340	0.0002	-0.000
55	12	-0.0332	0.0002	-0.000
60	13	-0.0325	0.0002	-0.000
65	14	-0.0318	0.0001	-0.000
70	15	-0.0312	0.0001	-0.000
75	16	-0.0305	0.0001	-0.000
80	17	-0.0300	0.0001	-0.000
85	18	-0.0294	0.0001	-0.000
90	19	-0.0289	0.0001	-0.000
95	20	-0.0284	0.0001	-0.000
100	21	-0.0280	0.0001	-0.000

Table (3.18)

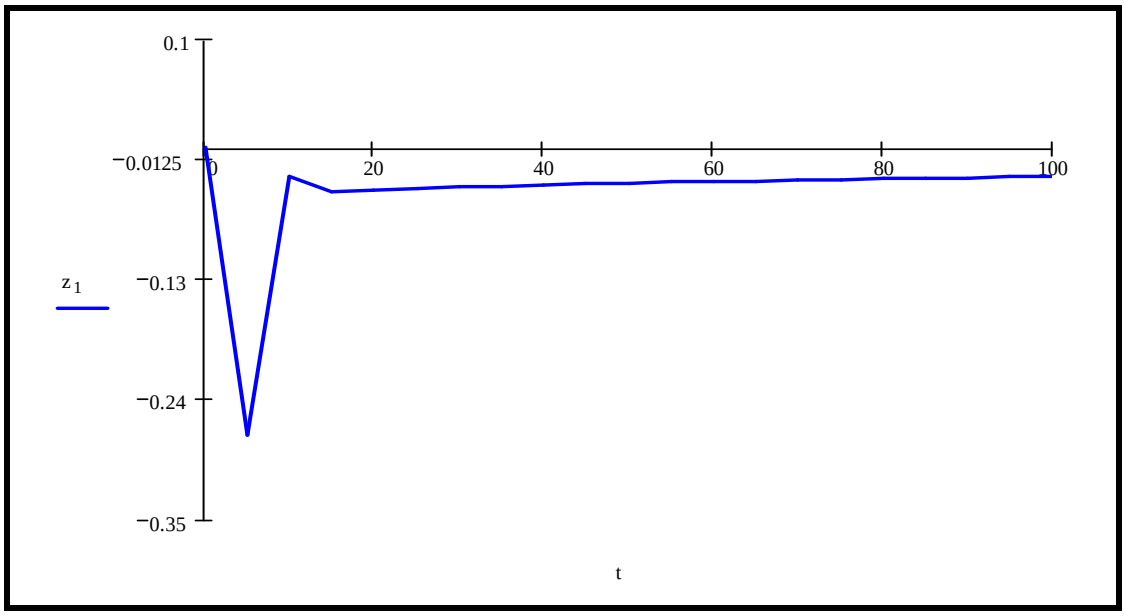


Figure (3.10)

The solution of the system (3.9) ($z_1(t), t \in (0,100)$) without using neural network

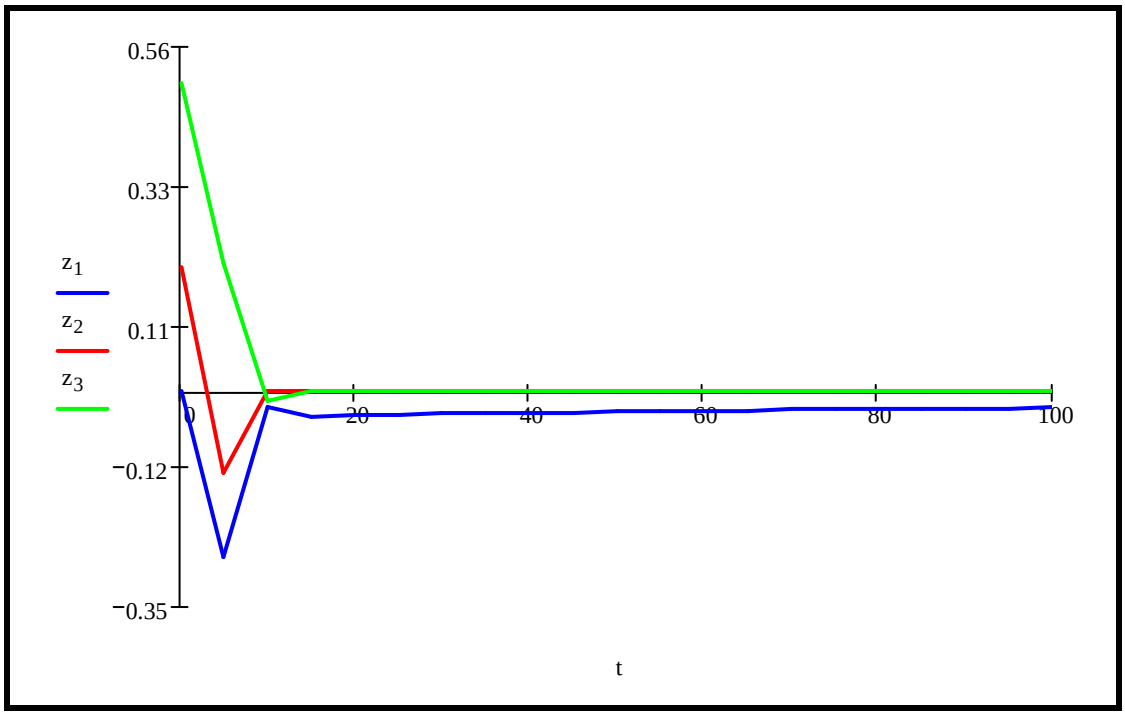


Figure (3.11)

the solution of the system (3.9) ($z_1(t), z_2(t), z_3(t), t \in (0,100)$) without using neural network

Step (4): Consider the desired trajectory are given by

$$z_{d1}(t)=\sin(t) \quad \Rightarrow z_{d1}(t)=z_{d1}(t=ih)\equiv z_{d1}(ih)\equiv \sin(ih)$$

$$z_{d2}(t)=\cos(t) \quad \Rightarrow z_{d2}(t)=z_{d2}(t=ih)\equiv z_{d2}(ih)\equiv \cos(ih)$$

$$z_{d3}(t)=e^{-5t} \quad \Rightarrow z_{d3}(t)=z_{d3}(t=ih)\equiv z_{d3}(ih)\equiv e^{-5ih}$$

where $y_d=(z_{d1}(t), z_{d2}(t), z_{d3}(t))$,such that $i\equiv$ stands for number of dividing time interval, and with step size $h=5$.

The desired trajectory $z_{d1}(t), z_{d2}(t), z_{d3}(t)$ with $t \in [0,100]$, and step size $h = 5$

t	i	$z_{d1}(i) = \sin(ih)$	$z_{d2}(i) = \cos(ih)$	$z_{d3}(i) = e^{-5ih}$
0	1	0	1.0000	1.000
5	2	-0.9589	0.2837	0.000
10	3	-0.5440	-0.8391	0.000
15	4	0.6503	-0.7597	0.000
20	5	0.9129	0.4081	0.000
25	6	-0.1324	0.9912	0.000
30	7	-0.9880	0.1543	0.000
35	8	-0.4282	-0.9037	0.000
40	9	0.7451	-0.6669	0.000
45	10	0.8509	0.5253	0.000
50	11	-0.2624	0.9650	0.000
55	12	-0.9998	0.0221	0.000
60	13	-0.3048	-0.9524	0.000
65	14	0.8268	-0.5625	0.000
70	15	0.7739	0.6333	0.000
75	16	-0.3878	0.9218	0.000
80	17	-0.9939	-0.1104	0.000
85	18	-0.1761	-0.9844	0.000
90	19	0.8940	-0.4481	0.000
95	20	0.6833	0.7302	0.000
100	21	-0.5064	0.8623	0.000

Table (3.19)

Step (5): Find the tracking errors $e_1(t), e_2(t), e_3(t)$ at time interval $t = [0, 100]$ and step size $h = 5$

$$e_1(t) = z_1(t) - \sin(t) \quad \Rightarrow e_1(t) = e_1(t = i) \equiv e_1(ih) \equiv e_1(i)$$
$$\Rightarrow e_1(i) = z_1(i) - \sin(ih)$$
$$e_2(t) = z_2(t) - \cos(t) \quad \Rightarrow e_2(t) = e_2(t = i) \equiv e_2(ih) \equiv e_2(i)$$
$$\Rightarrow e_2(i) = z_2(i) - \cos(ih)$$
$$e_3(t) = z_3(t) - e^{-5t} \quad \Rightarrow e_3(t) = e_3(t = i) \equiv e_3(ih) \equiv e_3(i)$$
$$\Rightarrow e_3(i) = z_3(i) - e^{-5ih}$$

Where $i \equiv$ stands for number of dividing time interval, so that the simulation are shown as below in table (3.20).

Errors $(e_1(t), e_2(t), e_3(t))$ of the System (3.9) Without

t	i	$e_1(i) = \sin(ih)$	$e_1(i) = \cos(ih)$	$e_3(i) = e^{-5ih}$
0	1	0	-0.8000	-0.5000
5	2	0.6882	-0.4161	0.2075
10	3	0.5159	0.8391	-0.0178
15	4	-0.6926	0.7605	-0.0001
20	5	-0.9533	-0.4078	0.000
25	6	0.0933	-0.9910	-0.000
30	7	0.9502	-0.1540	-0.000
35	8	0.3914	0.9039	-0.000
40	9	-0.7809	0.6671	-0.000
45	10	-0.8858	-0.5251	-0.000
50	11	0.2284	-0.9648	-0.000
55	12	0.9665	-0.0220	-0.000
60	13	0.2723	0.9526	-0.000
65	14	-0.8586	0.5626	-0.000
70	15	-0.8050	-0.6332	-0.000
75	16	0.3572	-0.9216	-0.000
80	17	0.9639	0.1105	-0.000
85	18	0.1466	0.9844	-0.000
90	19	-0.9229	0.4482	-0.000
95	20	-0.7117	-0.7301	-0.000
100	21	0.4784	-0.8622	-0.000

Table (3.20)

Using Neural Network with time interval $t \in [0, 100]$, step size $h = 5$

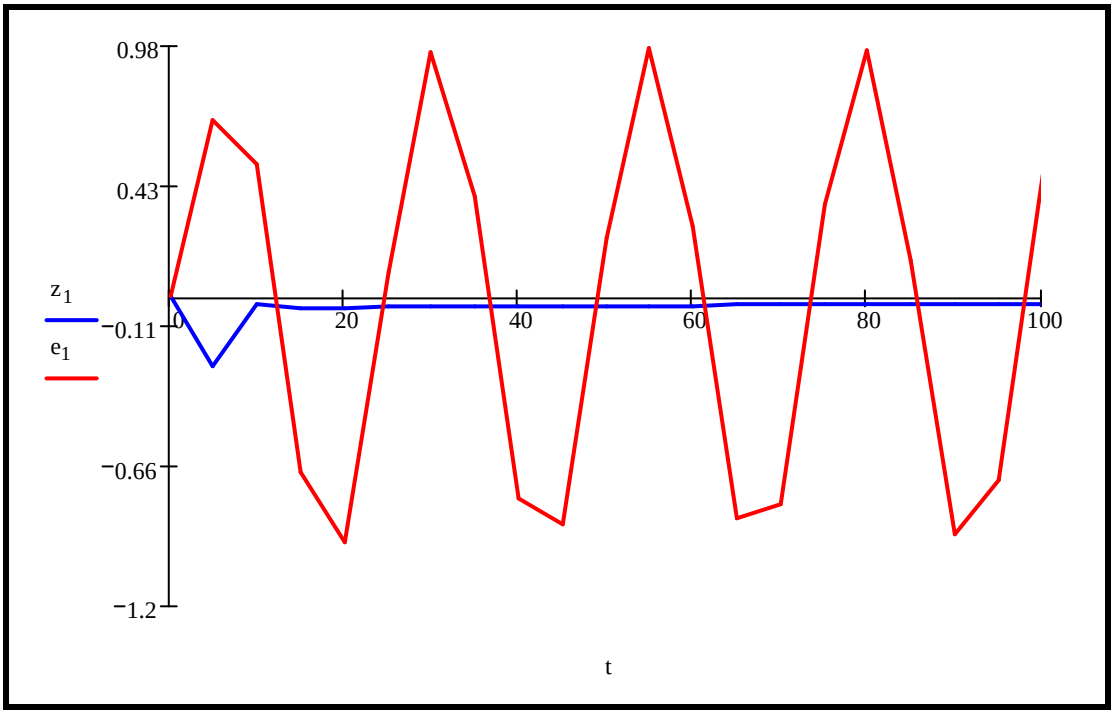


Figure (3.12)

The output of the system (3.9) and the error of the system without using neural network

Step (6): Find the filtered tracking error $r(t)$ as $r(t)=K\,e(t)$, where $K=[2,1,1]$, $e(t)=\left[\ e_1(t),\ e_2(t),\ e_3(t)\ \right]^T$

$$\Rightarrow r(t)=2(z_1(t)+\sin(t))+(z_2(t)-\cos(t))+(z_3(t)-e^{-5t})$$

Such that

$r(t)=r(t=ih)\equiv r(ih)\equiv r(i)$ where $i\equiv$ stands for number of dividing time interval such that $t=[0,100]$ and step size $h=5$.

The simulation are shown as below

Filtered tracking error at time interval $t=[0,100]$ and step size $h=5$. $r(t)$

t	i	$r(i)$
0	1	-1.3000
5	2	1.1678
10	3	1.8530
15	4	-0.6248
20	5	-2.3143
25	6	-0.8040
30	7	1.7464
35	8	1.6867
40	9	-0.8946
45	10	-2.2967
50	11	-0.5081
55	12	1.9111
60	13	1.4972
65	14	-1.1547
70	15	-2.2433
75	16	-0.2072
80	17	2.0383
85	18	1.2778
90	19	-1.3977
95	20	-2.1535
100	21	0.0946

Table (3.21)

The norm of $r(t)$ is approximated using the numerical results which is depends on the simulation such that the norm $r(t)=\|r(t)\|=7.0568$.

Step (7): chose the tracking control law of the system (3.9) by using the derivative of the filter tracking error and by substituting that $u_1(t)=W(t)-\hat{u}_2(t)$ so that we can chose W as follows

$$W=\frac{1}{g(z)}\left(-\hat{F}(z)-\overline{\mathrm{Y}}_d+\alpha+k_v r\right)$$

Where we can find $\overline{\mathrm{Y}}_d$ from the algorithm step (6)above are defined as

$$\overline{\mathrm{Y}}_d=-5e^{-5t}+\sin(t)+2z_2(t)-2\cos(t)+z_3(t)$$

And suppose $\hat{F}(z)=-2z_1^2-1.2264z_2-1.439z_3$, $\alpha=5z_1^3-2z_1^2$, $k_v=1.$ so that the tracking control law $W(t)=W(t=i)\equiv W(ih)\equiv W(i)$, where $i\equiv$ stands for number of dividing time interval such that $t\in[0,100]$ and step size $h=5$.The simulation are shown as below in table (2.22):

The Tracking Control Law (W) results $t \in [0,100]$ for $h = 5$

t	i	Tracking control law $(W(i))$
0	1	5.7648
5	2	2.7886
10	3	0.719
15	4	-2.7955
20	5	-2.4116
25	6	1.310
30	7	3.0424
35	8	0.3071
40	9	-2.9740
45	10	-2.0973
50	11	1.6839
55	12	2.9548
60	13	-0.1031
65	14	-3.1067
70	15	-1.7508
75	16	2.0239
80	17	2.8112
85	18	-0.5151
90	19	-3.1880
95	20	-1.3766
100	21	2.3254

Table (3.22)

The behavior of nonlinear part of tracking control law W is shown in the following figure.

Step (8): Since the control input as we defined above is

$$u_1(t) = W - \hat{u}_2$$

Where $W \equiv$ be the tracking control law

$\hat{u}_2(t) \equiv$ be the approximate function for the nonlinear control $u_2(t)$

which we will going to find it by using the general neural network approximation property.

Step (9) (Neural Network): Consider the two-Layered feed forward neural network have six, ten and one neurons at the input, hidden and output layers respectively.

1.The inputs for the first layer are.

$$x_{NN} = \left[z_{d1}(t),\ \ z_{d2}(t),\ \ z_{d3}(t),\ \ e_1(t),\ \ e_2(t),\ \ e_3(t) \right]$$

For $t \in [0, 100]$ with using step size $h = 5$ .where

$$z_{d1}(t) = z_{d1}(ih) \equiv \sin(ih)$$
$$z_{d2}(t) = z_{d2}(ih) \equiv \cos(ih)$$
$$z_{d3}(t) = z_{d3}(ih) \equiv e^{-5ih}$$
$$e_1(t) = e_1(ih) \equiv e_1(i)$$
$$e_2(t) = e_2(ih) \equiv e_2(i)$$
$$e_3(t) = e_3(ih) \equiv e_3(i)$$

Inputs to the First Layer Neural Network x_{NN} For $t \in [0,100]$ using step size $h = 5$

t	i	$z_{d1}(i)$	$z_{d2}(i)$	$z_{d3}(i)$	$e_1(i)$	$e_2(i)$	$e_3(i)$
0	1	0	1.0000	1.000	0	-0.8000	-0.5000
5	2	-0.9589	0.2837	0.000	0.6882	-0.4161	0.2075
10	3	-0.5440	-0.8391	0.000	0.5159	0.8391	-0.0178
15	4	0.6503	-0.7597	0.000	-0.6926	0.7605	-0.0001
20	5	0.9129	0.4081	0.000	-0.9533	-0.4078	0.000
25	6	-0.1324	0.9912	0.000	0.0933	-0.9910	-0.000
30	7	-0.9880	0.1543	0.000	0.9502	-0.1540	-0.000
35	8	-0.4282	-0.9037	0.000	0.3914	0.9039	-0.000
40	9	0.7451	-0.6669	0.000	-0.7809	0.6671	-0.000
45	10	0.8509	0.5253	0.000	-0.8858	-0.5251	-0.000
50	11	-0.2624	0.9650	0.000	0.2284	-0.9648	-0.000
55	12	-0.9998	0.0221	0.000	0.9665	-0.0220	-0.000
60	13	-0.3048	-0.9524	0.000	0.2723	0.9526	-0.000
65	14	0.8268	-0.5625	0.000	-0.8586	0.5626	-0.000
70	15	0.7739	0.6333	0.000	-0.8050	-0.6332	-0.000
75	16	-0.3878	0.9218	0.000	0.3572	-0.9216	-0.000
80	17	-0.9939	-0.1104	0.000	0.9639	0.1105	-0.000
85	18	-0.1761	-0.9844	0.000	0.1466	0.9844	-0.000
90	19	0.8940	-0.4481	0.000	-0.9229	0.4482	-0.000
95	20	0.6833	0.7302	0.000	-0.7117	-0.7301	-0.000
100	21	-0.5064	0.8623	0.000	0.4784	-0.8622	-0.000

Table (3.23)

2. Let the bias b are selected randomly distributed between **-15** and **+1**5 as where $b(j),\ j=1,2,\ldots,10$, $j\equiv$ depends on the number of the neuron in the hidden layer $(j=10)$in this application, the bias will be chosen as follows:

The Bias b to the First Layer Feed forward Neural Network

Bias b	٩	-2	٤	٢	٨	٦	١	-5	٧	١

Table (3.24)

with input to the bias has the value $x_0=-1$.

3.Consider the first layer feed forward weights $V_{ij}\ where i=1,\ldots 10,\ j=1,\ldots 6$, where i, j representing the number of neurons on the hidden layer and the input layer respectively. Such that V_{ij} will be chosen randomly distributed between -1 and +1 as

The First Layer Feed forward Neural Network Weights V

i	The first layer weight V_{ij}					
1	-1	٠	-1	١	٠	-1
2	-1	٠	٠	-1	-1	-1
3	0	-1	-1	-1	-1	١
4	-1	١	٠	١	١	٠
5	-1	١	-1	-1	١	-1
6	1	٠	١	-1	١	٠
7	0	٠	٠	-1	١	٠
8	1	٠	١	-1	٠	-1
9	١	١	-1	٠	٠	-1
10	-1	٠	-1	٠	٠	-1

Table (3.25)

4. We chose the **Log-sigmoid** transfer function f (activation function) (which be explained in chapter one) because it is differentiable and commonly used in many methods such as Back propagation method

$$f(\lambda)=\frac{1}{1+\exp(-\lambda)}, \text{ where } \lambda_i = \sum_{j=1}^{6} V_{ij}\, x_j, i=1,\ldots,10$$

noted that $,\lambda_i(t)\equiv\lambda_i(t=jh)\equiv\lambda_i(jh)\equiv\lambda_i(j), \quad where\ i=1,2,\ldots,10$, $i\equiv$ stand for number of evaluates in value $\lambda(t)$, $j\equiv$ stand for number of divided time interval. Such that the result of λ on the time interval $t=[0,100]$ with step size $h=5$, the evaluation are as shown in table (3.26)

The values of the sigmoid function $f(\lambda)$ with $t=[0,100]$, $h=5$

t	j	$\lambda_1(j)$	$\lambda_2(j)$	$\lambda_3(j)$	$\lambda_4(j)$	$\lambda_5(j)$	$\lambda_6(j)$	$\lambda_7(j)$	$\lambda_8(j)$	$\lambda_9(j)$	$\lambda_{10}(j)$
0	1	-4.50	2.300	-10.7	-2.80	-9.30	-1.80	-7.80	3.500	2.500	-8.500
5	2	-2.56	1.479	-9.34	-1.48	-9.06	-4.063	-8.10	0.145	1.1172	-7.248
10	3	-2.92	0.206	-9.53	-1.94	-8.95	-2.220	-6.67	0.958	0.634	-7.438
15	4	-5.34	0.281	-8.30	-4.32	-8.95	0.103	-5.54	3.343	1.890	-8.650
20	5	-5.86	1.448	-8.04	-4.86	-8.95	-0.541	-6.45	3.866	3.321	-8.913
25	6	-3.77	2.030	-9.09	-2.77	-8.96	-3.216	-8.08	1.774	2.858	-7.868
30	7	-2.06	1.191	-9.95	-1.06	-8.96	-4.092	-8.10	0.061	1.166	-7.012
35	8	-3.18	0.132	-9.39	-2.18	-8.96	-1.915	-6.48	1.180	0.668	-7.571
40	9	-5.52	0.368	-8.21	-4.52	-8.99	0.152	-5.52	3.526	2.078	-8.745
45	10	-5.73	1.560	-8.11	-4.73	-8.96	-0.788	-6.63	3.736	3.376	-8.850
50	11	-3.50	1.998	-9.22	-2.50	-8.96	-3.455	-8.19	1.509	2.702	-7.737
55	12	-2.03	1.055	-9.96	-1.03	-8.96	-3.988	-7.98	0.033	1.022	-7.000
60	13	-3.42	0.079	-9.27	-2.42	-8.96	-1.624	-6.31	1.422	0.742	-7.695
65	14	-5.68	0.469	-8.14	-4.68	-8.96	0.248	-5.57	3.685	2.2644	-8.856
70	15	-5.57	1.664	-8.19	-4.57	-8.96	-1.054	-6.82	3.578	3.107	-8.773
75	16	-3.25	1.952	-9.35	-2.25	-8.96	-3.666	-8.27	1.255	2.534	-7.612
80	17	-2.04	0.919	-9.96	-1.04	-8.96	-3.847	-7.85	0.042	0.895	-7.006
85	18	-3.77	0.045	-9.14	-2.67	-8.97	-1.338	-6.16	1.677	0.839	-7.823
90	19	-5.81	0.580	-8.07	-4.81	-8.97	0.265	-5.62	3.816	2.445	-8.894
95	20	-5.39	1.758	-8.28	-4.39	-8.97	-1.335	-7.01	3.395	3.413	-8.683
100	21	-3.01	1.890	-9.47	-2.01	-8.97	-3.847	-8.34	1.015	2.356	-7.493

Table (3.26)

5. the output of the first layer feed forward neural network is given by

$$q_i(t)=f\left(\sum_{j=1}^{6} V_{ij}(t)\,x_j(t)\right)+b_i \quad where\ i=1,\ldots,10$$

The result is as follows The output of the first layer neural network with $t=[0,100]$ with step size $h=5$

t	j	$q_1(j)$	$q_2(j)$	$q_3(j)$	$q_4(j)$	$q_5(j)$	$q_6(j)$	$q_7(j)$	$q_8(j)$	$q_9(j)$	$q_{10}(j)$
0	1	0.011	0.390	0.00	0.073	0.000	0.141	0.000	0.970	0.924	0.0002
5	2	0.071	0.814	0.000	0.184	0.000	0.016	0.000	0.536	0.753	0.000
10	3	0.051	0.551	0.000	0.125	0.000	0.097	0.001	0.722	0.653	0.000
15	4	0.004	0.570	0.000	0.012	0.000	0.525	0.003	0.965	0.868	0.000
20	5	0.002	0.809	0.000	0.007	0.000	0.367	0.001	0.979	0.965	0.000
25	6	0.022	0.883	0.000	0.058	0.000	0.038	0.000	0.855	0.945	0.0004
30	7	0.112	0.767	0.000	0.257	0.000	0.0164	0.000	0.515	0.762	0.0009
35	8	0.039	0.533	0.000	0.101	0.000	0.128	0.001	0.765	0.661	0.0005
40	9	0.004	0.591	0.000	0.010	0.000	0.548	0.003	0.971	0.888	0.0002
45	10	0.003	0.826	0.000	0.008	0.000	0.312	0.001	0.976	0.967	0.0001
50	11	0.029	0.880	0.000	0.075	0.000	0.030	0.000	0.849	0.937	0.0004
55	12	0.115	0.741	0.000	0.262	0.000	0.018	0.000	0.508	0.735	0.0009
60	13	0.031	0.520	0.000	0.081	0.000	0.164	0.001	0.805	0.677	0.0005
65	14	0.003	0.615	0.000	0.009	0.000	0.561	0.0038	0.975	0.905	0.0001
70	15	0.003	0.840	0.000	0.010	0.000	0.258	0.001	0.972	0.967	0.0002
75	16	0.037	0.875	0.000	0.0949	0.000	0.024	0.000	0.778	0.926	0.0005
80	17	0.114	0.714	0.000	0.260	0.000	0.020	0.000	0.510	0.710	0.0009
85	18	0.024	0.511	0.000	0.0643	0.000	0.207	0.002	0.842	0.698	0.0004
90	19	0.003	0.641	0.000	0.008	0.000	0.565	0.003	0.978	0.920	0.0001
95	20	0.004	0.853	0.000	0.012	0.000	0.208	0.000	0.967	0.968	0.0002
100	21	0.046	0.868	0.000	0.117	0.000	0.020	0.000	0.734	0.913	0.0006

Table (3.27)

Now we must produce the hidden layer to the feed forward neural network which have the following

1. the hidden layer (second layer) for the neural network have an input whose already the output of the first layer (input layer) $\left(q_i, where\, i = 1,\ldots,10\right)$

2.The neural network weights to the hidden layer (second layer) is $w(t) = \left[\, w_1(t) \;\; w_2(t) \;\; w_3(t) \;\; \ldots \;\; w_{10}(t) \,\right]$ which we are chose it randomly between **-1** and **+1** as follows ,such that $w_i(t) \equiv w_i(t = jh) \equiv w_i(jh) \equiv w_i(j), i = 1,2,\ldots,10$, $i \equiv$ stand for number of evaluates in value $w(t)$, $j \equiv$ stand for number of divided time interval. Such that the result of w on the time interval $t = [0,100]$ with step size $h = 5$, the evaluation are as shown in table (3.28).

The hidden layer weights $w(t)$ $t=[0,100]$ with step size $h=5$

t	j	$w_1(j)$	$w_2(j)$	$w_3(j)$	$w_4(j)$	$w_5(j)$	$w_6(j)$	$w_7(j)$	$w_8(j)$	$w_9(j)$	$w_{10}(j)$
0	1	0	0	-1	0	0	-1	0	-1	-1	•
5	2	0	0	0	0	0	0	0	0	-1	0
10	3	0	0	0	0	0	-1	0	0	-1	0
15	4	0	0	0	0	0	0	0	0	0	0
20	5	0	0	-1	0	0	0	-1	0	-1	0
25	6	0	0	-1	0	0	0	0	-1	-1	0
30	7	-1	0	0	0	-1	0	0	-1	0	0
35	8	0	0	0	0	-1	-1	0	-1	0	0
40	9	0	0	0	0	-1	-1	0	0	0	0
45	10	-1	0	0	0	-1	-1	0	0	0	0
50	11	0	0	0	0	-1	0	-1	0	0	-1
55	12	0	0	0	0	0	0	0	0	-1	0
60	13	0	0	-1	0	0	0	0	-1	-1	-1
65	14	0	0	-1	-1	-1	-1	0	-1	-1	0
70	15	0	0	-1	0	0	0	0	0	0	-1
75	16	-1	0	0	0	-1	0	0	0	-1	-1
80	17	0	-1	-1	0	0	-1	-1	0	0	-1
85	18	-1	-1	0	0	0	0	0	0	-1	0
90	19	0	0	0	0	0	0	-1	-1	-1	0
95	20	0	0	0	0	0	0	0	0	0	0
100	21	0	0	-1	0	0	-1	-1	0	0	-1

Figure (3.28)

Where the norm of $w(t)$ is bounded by the value w_m $(\|w(t)\| \le w_m)$ (the norm approximated by matrix norm nested of continuous norm) suppose that $w_m = 7$, such that $\|w(t)\| = 4.6944$.

3.Consider the hidden layer estimated neural network weights can be found by solving the neural network tuning algorithm

$$\dot{\hat{w}} = S\, f\left(V^T x_{NN}\right) r\, g(x) - k\, S \|r\| \hat{w}$$

And suppose that $S = 5, k = 1.$ with initial conditions

The initial conditions to the neural network tuning algorithm equation

$\hat{w}_1(0)$	$\hat{w}_2(0)$	$\hat{w}_3(0)$	$\hat{w}_4(0)$	$\hat{w}_5(0)$	$\hat{w}_6(0)$	$\hat{w}_7(0)$	$\hat{w}_8(0)$	$\hat{w}_9(0)$	$\hat{w}_{10}(0)$
0	0	0	0	0	0	0	0	0	0

Table (3.29)

The solution of the neural network estimated weights $\hat{w}(t) \equiv \hat{w}_i(t = jh) \equiv \hat{w}_i(jh) \equiv \hat{w}_i(j), i = 1,2,\ldots,10,$ $i \equiv$ stand for number of evaluates in value $\hat{w}(t)$, $j \equiv$ stand for number of divided time interval. Such that the result of $\hat{w}$ on the time interval $t \in [0,100]$ with step size $h = 5$, the solution are as shown in table (3.30).

Results of the neural network estimated weights $\hat{w}(t)$

t	j	$\hat{w}_1(j)$	$\hat{w}_2(j)$	$\hat{w}_3(j)$	$\hat{w}_4(j)$	$\hat{w}_5(j)$	$\hat{w}_6(j)$	$\hat{w}_7(j)$	$\hat{w}_8(j)$	$\hat{w}_9(j)$	$\hat{w}_{10}(j)$
0	1	0	0	0	0	0	0	0	0	0	0
5	2	-0.12	0.554	0.000	-0.28	0.000	0.575	0.002	0.957	0.798	-0.00
10	3	-0.12	0.554	0.000	-0.28	0.000	0.5752	0.002	0.957	0.798	-0.00
15	4	-0.12	0.554	0.000	-0.28	0.000	0.579	0.002	0.957	0.798	-0.00
20	5	-0.12	0.553	0.000	-0.28	0.000	0.574	0.002	0.956	0.797	-0.00
25	6	-0.12	0.553	0.000	-0.28	0.000	0.5741	0.002	0.955	0.796	-0.00
30	7	-0.12	0553	0.000	-0.28	0.000	0.573	0.002	0.955	0.796	-0.00
35	8	-0.12	0.553	0.000	-0.28	0.000	0.573	0.002	0.955	0.796	-0.00
40	9	-0.12	0.553	0.000	-0.28	0.000	0.574	0.002	0.955	0.797	-0.00
45	10	-0.12	0.553	0.000	-0.28	0.000	0.574	0.002	0.956	0.797	-0.00
50	11	-0.12	0.554	0.000	-0.28	0.000	0.575	0.002	0.957	0.798	-0.00
55	12	-0.12	0.554	0.000	-0.28	0.000	0.575	0.002	0.975	0.798	-0.00
60	13	-0.12	0.554	0.000	-0.28	0.000	0.571	0.002	0.957	0.797	-0.00
65	14	-0.12	0.553	0.000	-0.28	0.000	0.574	0.002	0.956	0.797	-0.00
70	15	-0.12	0.553	0.000	-0.28	0.000	0.574	0.002	0.955	0.797	-0.00
75	16	-0.12	0.553	0.000	-0.28	0.000	0.573	0.002	0.955	0.796	-0.00
80	17	-0.12	0.553	0.000	-0.28	0.000	0.573	0.002	0.955	0.796	-0.00
85	18	-0.12	0.553	0.000	-0.28	0.000	0.574	0.002	0.955	0.796	-0.00
90	19	-0.12	0.553	0.000	-0.28	0.000	0.574	0.002	0.956	0.797	-0.00
95	20	-0.12	0.554	0.000	-0.28	0.000	0.574	0.002	0.956	0.797	-0.00
100	21	-0.12	0.553	0.000	-0.28	0.000	0.574	0.002	0.956	0.797	-0.00

Table (3.30)

4. we can find the neural network weights approximation error $\tilde{w}(t)$ by the following

$$\tilde{w} = w(t) - \hat{w}(t), \tilde{w} \in R^{10} \Rightarrow \tilde{w}_i(t) \equiv \tilde{w}_i(t = jh) \equiv \tilde{w}_i(jh) \equiv \tilde{w}_i(j), i = 1,2,\ldots,10,$$

$i \equiv$ stand for number of evaluates in value $\tilde{w}(t)$, $j \equiv$ stand for number of divided time interval. Such that the result of $\tilde{w}$ on the time interval $t \in [0,100]$ with step size $h = 5$, the solution are as shown in table (3.31).

The neural network weights approximation error $\tilde{w}(t)\, t \in [0,100],\ h = 5$

t	j	$\tilde{w}_1(j)$	$\tilde{w}_2(j)$	$\tilde{w}_3(j)$	$\tilde{w}_4(j)$	$\tilde{w}_5(j)$	$\tilde{w}_6(j)$	$\tilde{w}_7(j)$	$\tilde{w}_8(j)$	$\tilde{w}_9(j)$	$\tilde{w}_{10}(j)$
0	1	-1.00	0	0	-1.00	0	0	-1.00	0	0	-1.00
5	2	0.124	-0.55	-0.00	0.285	-0.00	-1.57	-0.00	-1.95	-1.79	0.00
10	3	-0.87	-0.55	-0.00	0.285	-1.00	-0.57	-0.00	1.957	-0.79	-0.99
15	4	0.124	-1.55	-1.00	0.285	-0.00	-0.57	-0.00	-0.95	-0.79	0.00
20	5	0.124	-0.55	-1.00	0.285	-0.00	-1.57	-0.00	-1.95	-0.79	0.00
25	6	0.124	-0.55	-0.00	-0.71	-0.00	-0.57	-0.00	-0.95	-1.79	-0.99
30	7	-0.87	-0.55	-0.00	-0.71	-0.00	-1.57	-0.00	-1.95	-1.79	0.00
35	8	0.124	-0.55	-1.00	0.284	-1.00	-1.57	-1.00	-0.95	-0.79	0.00
40	9	0.124	-0.55	-1.00	0.284	-0.00	-0.57	-0.00	-1.95	-0.79	-0.99
45	10	-0.87	-0.55	-1.00	-0.71	-1.00	-0.57	-0.00	-1.95	-0.79	0.00
50	11	-0.87	-1.55	-0.00	0.285	-0.00	-0.57	-0.00	-0.95	-0.79	0.00
55	12	0.124	-0.55	-0.00	0.285	-0.00	-0.57	-0.00	-0.95	-0.79	0.00
60	13	-0.87	-0.55	-0.00	0.285	-0.00	-0.57	-1.00	-0.95	-0.79	-0.99
65	14	-0.87	-1.55	-1.00	0.251	-1.00	-0.57	-0.00	-0.95	-0.79	0.00
70	15	0.124	-0.55	-1.00	0.28	-1.00	-0.57	-0.00	-0.95	-1.79	-0.99
75	16	0.124	-0.55	-0.00	-0.71	-0.00	-0.57	-1.00	-0.95	-0.79	0.00
80	17	0.124	-1.55	-0.00	0.284	-0.00	-0.57	-1.00	-0.95	-1.79	-0.99
85	18	0.124	1.553	-1.00	0.284	-0.00	-1.57	-1.00	-1.95	-0.79	0.00
90	19	-0.87	-1.55	-0.00	0.284	-0.00	-0.57	-1.00	-1.95	-0.79	-0.99
95	20	0.124	-0.55	-1.00	0.285	-0.00	-0.57	-0.00	-0.95	-0.79	-0.99
100	21	-0.87	-0.55	-1.00	-0.71	-1.00	-0.57	-0.00	-1.95	-0.79	-0.99

Table (3.31)

The adjusting of the neural network depending on the approximated weights $\tilde{w}(t)$ by the equality

$$\|\tilde{w}\|_F \geq \left(\frac{\sqrt{\frac{k}{4} w_m^2 + \varepsilon_n} + \frac{1}{2} w_m}{k} \right)^{\frac{1}{2}} \quad , where \ \|\tilde{w}\|_F = \sqrt{\sum_{ij} w_{ij}^2}$$

Where $k = 1, \varepsilon_n = 0.0001,$ and $w_m = 7$ as we defined it above. [if the above equality false we will restart the computations of the network from the begging (i.e. go back to neural network steps, step (2). And whenever the equality true we will stop the training of the network and compute the output of the network $y(t)$]. So that norm of $\tilde{w}(t) = 8.1828 > 2.6458$.

5.we will tack the Log-sigmoid transfer function (activation function) .

$$f(\lambda) = \frac{1}{1 + \exp(-\lambda)}$$

6.the output of the neural network is given by

$$y_k(t) = f\left(\sum_{i=1}^{10} \tilde{w}_{ki}(t) q_i(t) \right), \quad where \ k = 1, \ i = 1, \ldots 10.$$

Suppose that $y_1(t) \equiv y_1(t = jh) \equiv y_1(jh) \equiv y_1(j), \quad j \equiv$ stand for number of divided time interval. Such that the result of $y(t)$ on the time interval $t \in [0, 100]$ with step size $h = 5$, the solution are as shown in table (3.32).

The output of the neural network $t \in [0,100]$ with step size $h = 5$

t	j	$y_1(j)$
0	1	٠,٤٨١٠
5	2	٠,٣٦٢٣
10	3	٠,١٩٦٩
15	4	٠,٤٠١٥
20	5	٠,٣٦٧٦
25	6	٠,٣٩٢٠
30	7	٠,٢٠١٥
35	8	٠,٣٦٥٢
40	9	٠,٣٧٢٨
45	10	٠,٣٥٨١
50	11	٠,٣٩٤٣
55	12	٠,٢٠٢٥
60	13	٠,٤٠٤٢
65	14	٠,١٨٤٣
70	15	٠,٣٦٤٤
75	16	٠,٣٦٤٠
80	17	٠,١٧٦٩
85	18	٠,٣٤٩٣
90	19	٠,٣٧١٢
95	20	٠,١٧٧٢
100	21	٠,٤٨١٠

Table (3.32)

Step(9): We can now find the control input of the system (3.9), with using $\hat{u}_2(t) = y_k(t)$ and we defined it in step (7) as $u_1(t) \equiv u_1(t = jh) \equiv u_1(jh) \equiv u_1(j)$, $j \equiv$ stand for number of divided time interval. Such that the result of $u_1(t)$ on the time interval $t \in [0, 100]$ with step size $h = 5$, the solution are as shown in table (3.33). is given by

The control input of the system (3.9) $t \in [0,100]$ with step size $h = 5$

j	j	$u_1(j)$
0	1	5.2837
5	2	2.4263
10	3	0.5140
15	4	-3.1970
20	5	-2.7792
25	6	0.9180
30	7	2.8410
35	8	-0.0581
40	9	-3.3468
45	10	-2.4554
50	11	1.2897
55	12	2.7523
60	13	-0.5073
65	14	-3.2909
70	15	2.1152
75	16	1.6599
80	17	٢,٦٣٤٣
85	18	-0.8645
90	19	-3.5592
95	20	-1.5538
100	21	٢,١٣٠٨

Table (3.33)

Step (10): display the control input $u_1(t)$ in the system (3.9) (step(1)) and find new the results and the error of the nonlinear system with using neural network.

Where

$$z_1(t) \equiv z_1(t=j) \equiv z_1(jh) \equiv z_1(j) \Rightarrow e_1(t) = \sin(t) \equiv \sin(t=j) \equiv \sin(jh)$$

$$z_2(t) \equiv z_2(t=j) \equiv z_2(jh) \equiv z_2(j) \Rightarrow e_2(t) = \cos(t) \equiv \cos(t=j) \equiv \cos(jh)$$

$$z_3(t) \equiv z_3(t=j) \equiv z_3(jh) \equiv z_3(j) \Rightarrow e_3(t) = e^{-5(t)} \equiv e^{-5(t=j)} \equiv e^{-5\,jh}$$

$j \equiv$ stand for number of divided time interval. Such that the result of $u_1(t)$ on the time interval $t \in [0,100]$ with step size $h = 5$, the solution are as shown in table (3.34). is given by

The results of the system (3.9) with using neural network on the time interval $t \in [0,100]$ with step size $h = 5$

t	j	$z_1(j)$	$z_2(j)$	$z_3(j)$	$e_1(j)$	$e_2(j)$	$e_3(j)$
0	1	0	0.2	0.5	0	-0.8	-0.5
5	2	0.2	0.7	4.8140	-0.7093	1.1161	4.8184
10	3	0.9	5.5190	-0.5877	0.7584	6.5089	-0.5877
15	4	6.4190	4.9312	-9.6414	7.1758	5.5849	-9.6414
20	5	0.0114	-0.0047	-1.3274	0.0123	-0.0050	-1.3274
25	6	0.0066	-1.3321	-6.7253	0.0064	-1.331	-6.7253
30	7	-1.3255	-8.0574	3.1233	-1.3261	-8.0582	3.1233
35	8	-0.000	-0.000	1.1644	-0.000	-0.000	1.1644
40	9	-0.000	-0.0116	4.1252	-0.000	0.10116	4.1252
45	10	0.0012	0.4137	1.2848	0.0012	0.4137	1.2848
50	11	0.000	0.000	-7.8930	0.000	0.000	-7.8931
55	12	0.000	-0.000	-3.5698	0.000	0.000	-3.598
60	13	-0.000	-0.0357	-4.7038	-0.000	-0.0357	-4.7038
65	14	-0.000	-0.000	2.4587	-0.000	-0.000	2.4587
70	15	-0.000	0.000	2.2740	-0.000	0.000	2.2740
75	16	0.000	0.000	5.4442	0.000	0.000	5.4442
80	17	0.0000	0.000	-7.4319	0.0000	0.000	-7.4319
85	18	0.000	-7.4319	0.000	0.000	-7.4319	0.000
90	19	-7.4319	0.000	0.000	-7.4319	0.000	0.000
95	20	0.0000	0.0000	0.000	0.000	0.000	0.000
100	21	0.000	0.000	0.000	0.000	0.000	0.000

Table (3.34)

The effectiveness of using neural network shown in the figures down and who the system in (3.9) will be stable

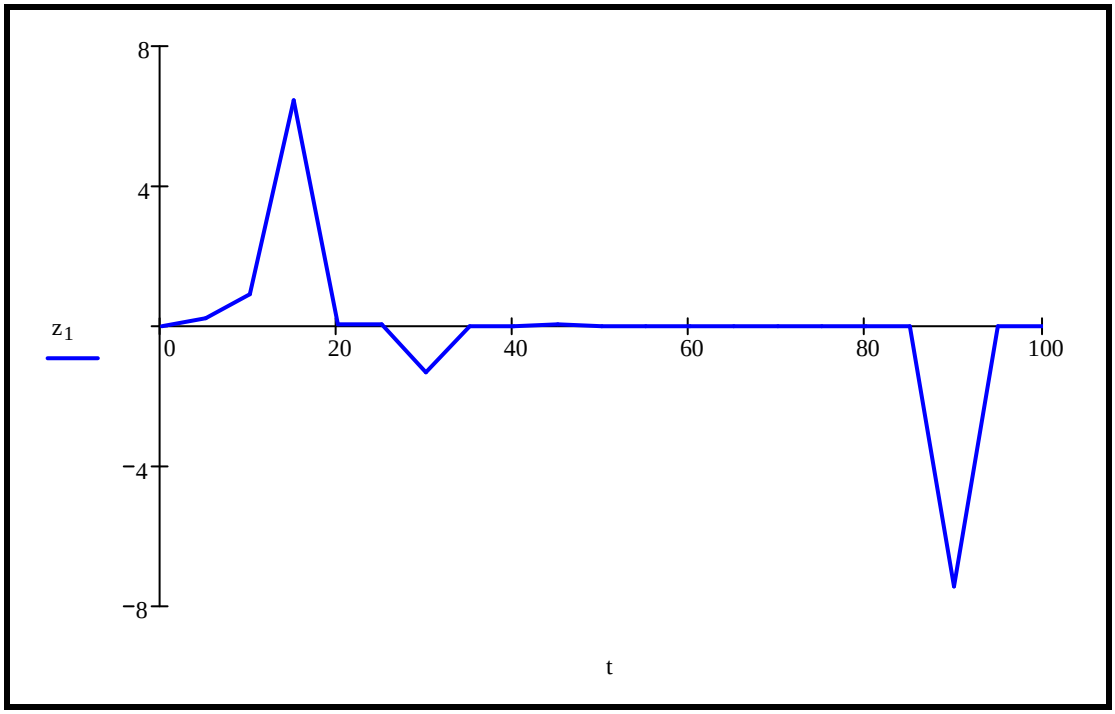


Figure (3.13)

The output of the system (3.9) ($z_1(t), t \in (0,100)$) with using neural network

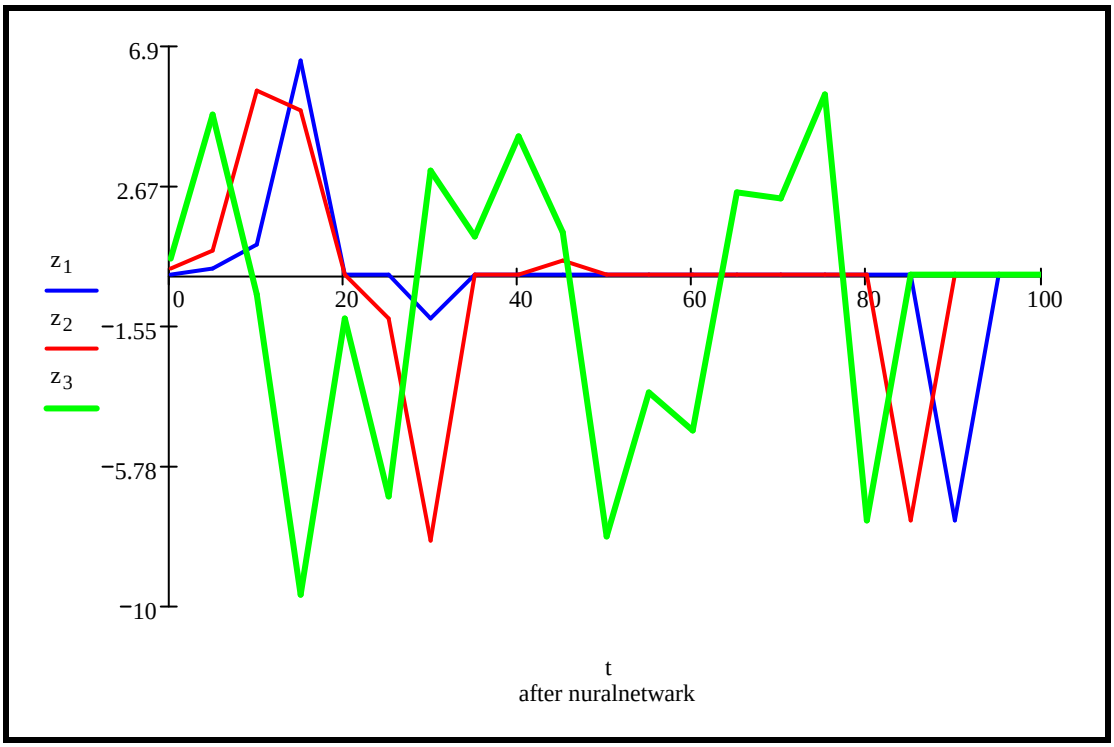


Figure (3.14)

The solution of the system (3.9) ($z_1(t), z_2(t), z_3(t), t \in (0,100)$) with using neural network

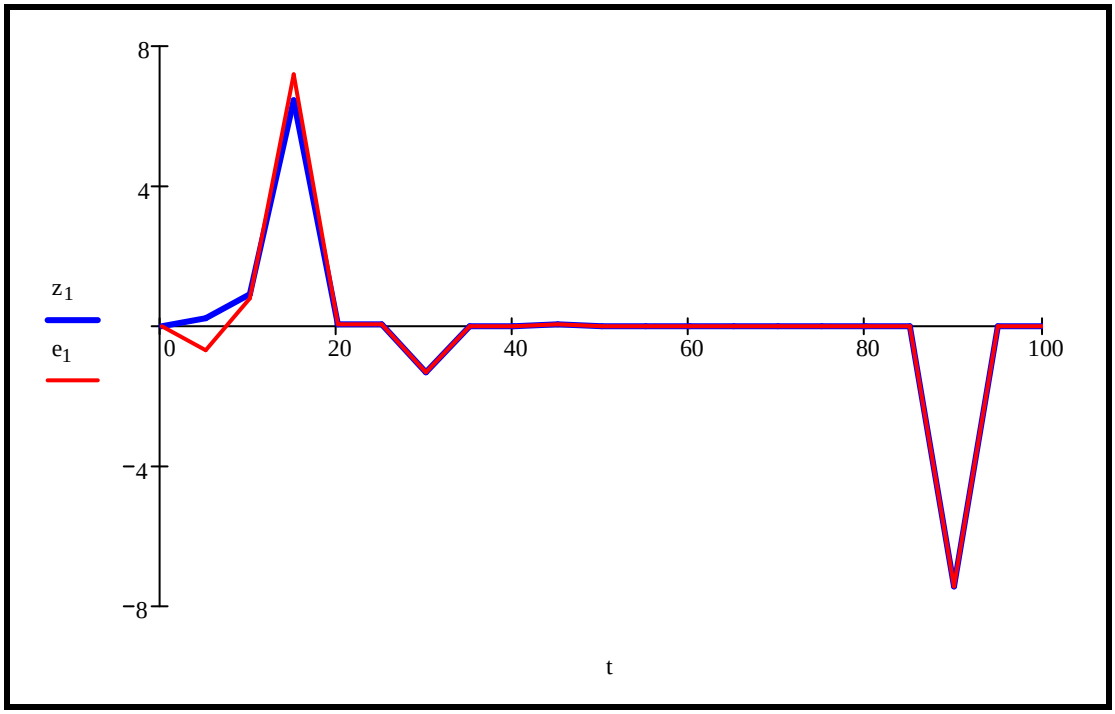


Figure (3.15)

The output of the system (3.9) ($z_1(t), t \in (0,100)$) and the error of the system with using neural network

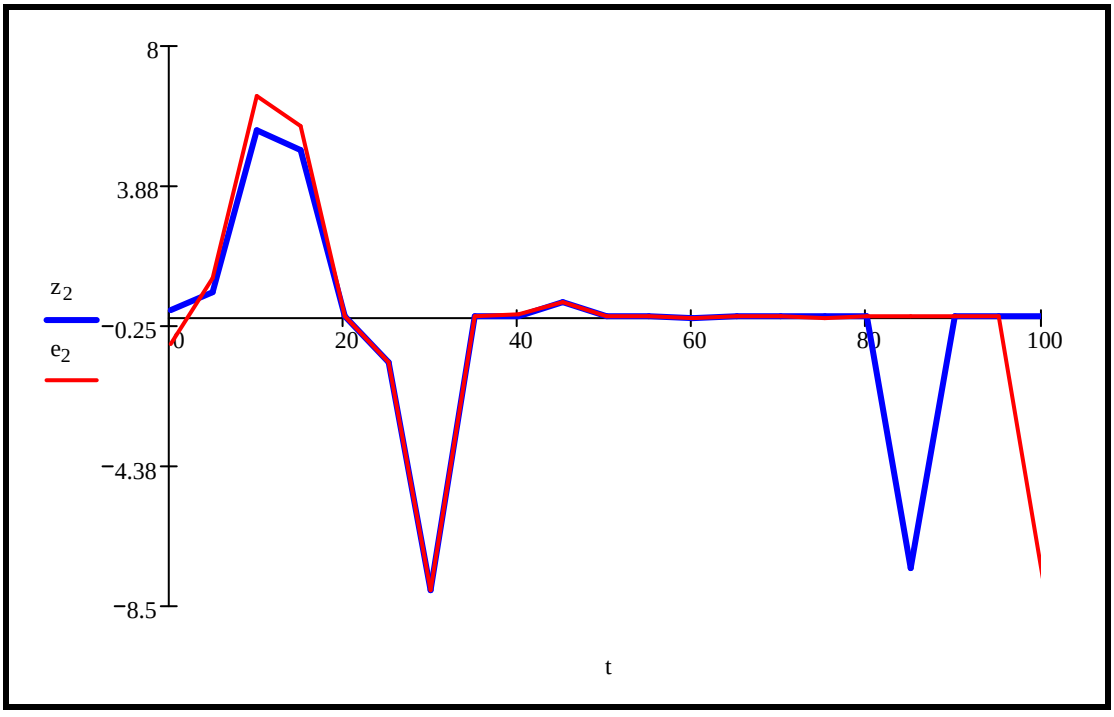


Figure (3.16)

The solution of the system (3.9) ($z_2(t), t \in (0,100)$) and the error of the system with using neural network

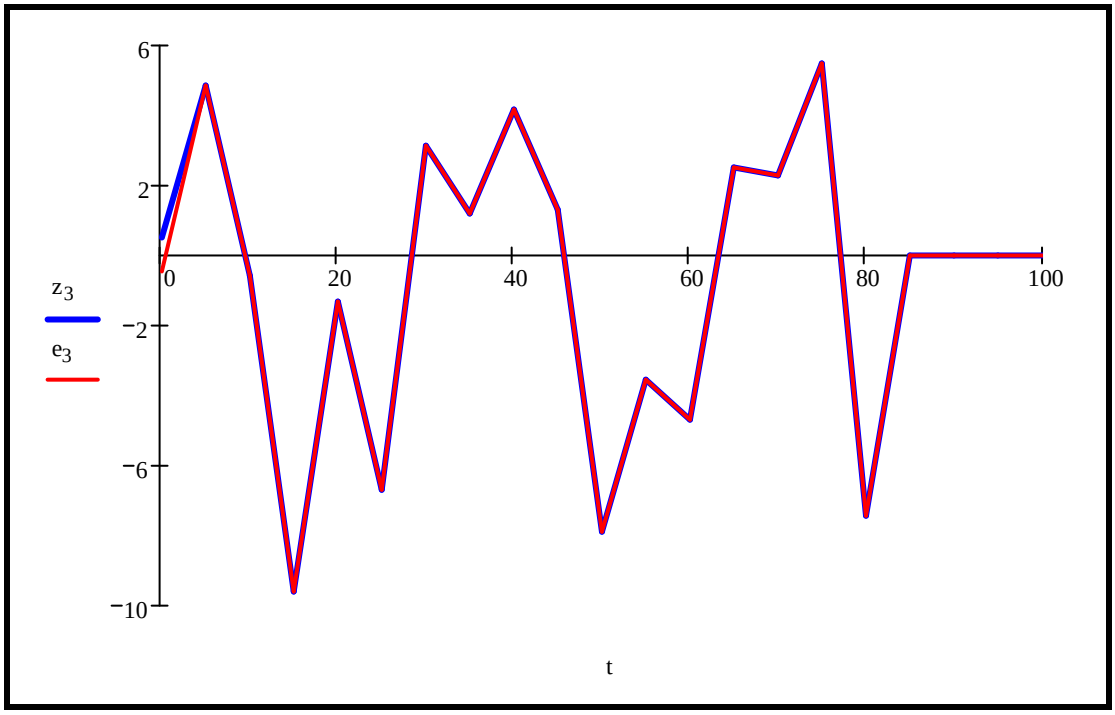
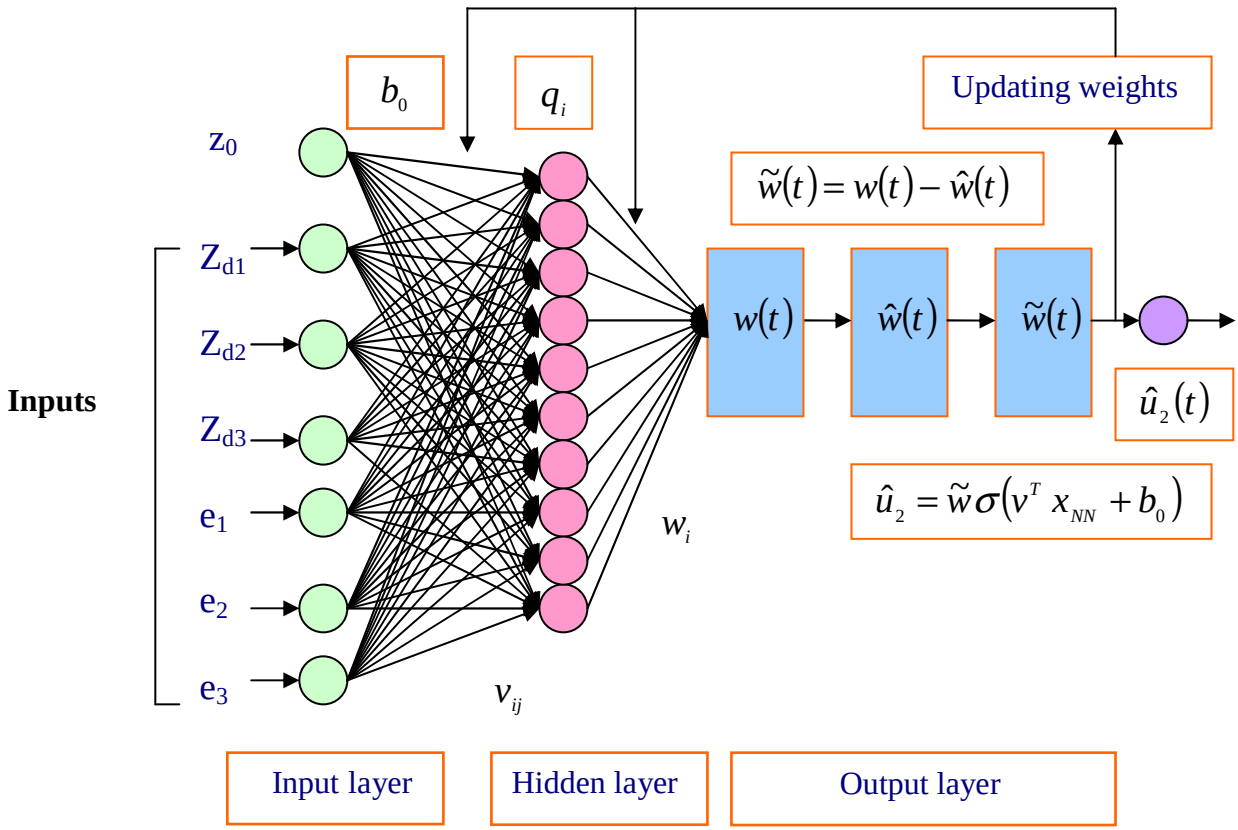


Figure (3.17)

The solution of the system (3.9) ($z_3(t), t \in (0,100)$) and the error of the system with using neural network



where

$(the\ bias \equiv b_0 = [b_{1,0}, \ldots, b_{10,0}])$

$(the\ input\ to\ the\ neural\ network\ are \equiv z = [z_{d1}, z_{d2}, z_{d3}, e_1, e_2, e_3])$

$(the\ weights\ in\ the\ 1st\ layer)\ v_{ij}, i = 1, \ldots 10, (i \equiv the\ no.\ of\ neurons\ in\ the\ hidden\ layer)$

$j = 1, \ldots 6, (j \equiv the\ no.\ of\ neurons\ in\ the\ input\ layer)$

$$q_i = \sigma\left(\sum_{j=1}^{6} v_{ij}\ z_j\right), (q \equiv the\ output\ of\ the\ first\ layer)$$

$(the\ weights\ of\ the\ hidden\ layer) w_{ki}, k = 1, (k \equiv the\ no.\ of\ the\ neurons\ in\ the\ output\ layer),$

$i = 1, \ldots, 10$

$(the\ estimated\ weights\ \hat{w}\ which\ found\ by\ solving \Rightarrow \dot{\hat{w}} = sqg - ks\|r\|\hat{w})$

$(the\ error\ weights\ \tilde{w} = w - \hat{w})$

$$\left(the\ output\ of\ the\ neural\ network\ \hat{u}_{2k} = \sigma\left(\sum_{i=1}^{10} \tilde{w}_{ki}\ q_i\right)\right)$$

Figure (3.18)

Architectural graph of two-layer of neural networks of application(3.2.2)

2

Introduction

Today automatic control systems have become an integrated part of our life. They appear in every thing from simple electronic household products to air planes and spacecrafts. Automatic control systems can take highly different shapes but common to them all is their function to manipulate a system so that it behaves in a desired fashion. When designing a controller for a particular system, it is obvious that a vital intermediate step is to acquire some knowledge about how the system will respond when it is manipulated in various ways. Not until such knowledge is available, can one plan how the system should be controlled to exhibit a certain behavior.

Control of nonlinear systems is a major application area for neural networks. The control design problem will be approached in two ways: direct design methods and indirect design methods. "Direct design" mean that a neural network directly implements the controller. Therefore, a network must be trained as the controller according to some kind of relevant criterion. The indirect methods represent a more conventional approach, where the design is based on a neural network model of the system to be controlled. In this case the controller is not itself a neural network, [Zurada, 1996].

2.1 Remarks and Comments

To control a system is to make it behave in a desired manner. How to express this "desired behavior" depends primarily on the task to be solved, but the dynamics of the system, the actuators, the measurement equipment, the available computational power, etc., influence the formulation of the desired behavior as well. Although the desired behavior obviously is very dependent of the application, the need to rephrase it in mathematical terms suited for

practical design of control systems seriously limits the means of expression. At the higher level it is customary to distinguish two basic types of problems:

Regulation problems. The fundamental desired behavior is to keep the output of the system at a constant level regardless of the disturbances acting on the system (e.g., controlling the temperature in a room).

Servo problems. The fundamental desired behavior is to make the output follow a reference trajectory closely (e.g., controlling a robot), [Hertz, 1991].

2.1.1 Why Using Neural Networks in Control System?

How are neural networks useful for control system design? It is practical to distinguish between the following categories of controllers:

1. **Highly specialized controllers** that are relevant when the system to be controlled is in some sense difficult to stabilize or when the performance is extremely important.

2. General purpose controllers where the same controller structure can be used on a wide class of practical systems. The controllers are characterize by being simple to tune so that a satisfactory performance can be achieved with a modest effort.

Basically, neural networks are relevant in both cases, but they probably have the biggest potential within general purpose control. It is believed that their ability to model a wide class of systems in many applications can reduce time spent on development and offer a better performance than can be obtained with conventional techniques like auto-tuned PID-controllers, [Zurada, 1996].

2.2 Remarks and Comments

1. The multilayer perceptron (MLP) network (which discussed in chapter one/section four) is straightforward to employ for discrete-time

modeling of dynamic systems for which there is a nonlinear relationship between the system's input and output:

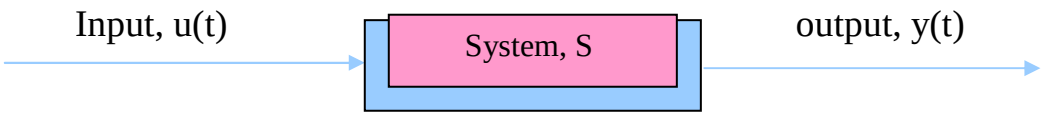


Figure (2.1)

A dynamic system with one input u(t) and one output y(t)

Let t count the multiple of sampling periods so that $y(t)$ specifies the present output while $y(t-1)$ signifies the output observed at the previous sampling instant, etc, if it is assumed that the output of the dynamic system at discrete time instances can be described as a function of a number of past inputs and outputs

$$y(t) = S[y(t-1), \ldots, y(t-n), u(t-1), \ldots, u(t-m)] \qquad (2.1)$$

an multilayer perceptron network can be used for approximating ***S*** if the inputs to the network $[\, p_1, p_2, \ldots \,]$ are chosen as the n past outputs and the m past inputs:

$$\hat{y}(t \setminus \theta) = g[\theta, p(t)] = \sum_{j=1}^{nh} W_j f_j \left[\sum_{l=1}^{n+m} w_{j,l}\ p_l(t) + w_{j,0} \right] + W_{i,0} \qquad (2.2)$$

Where $W_j, f_j, p_l(t), W_{i,0}$ are the weights matrix, the transfer function (activation function), the inputs and the bias of the multilayer perceptron network respectively, and $nh, n+m$ are the number of the neuron in the hidden layer, the number of neuron in the first layer of the multilayer perceptron network, respectively, [Hertz.,1991].

2. The history of universal approximation by neural networks started in fact in 1900. When Hilbert formulated a list of 23 challenging problems for

the century to come. The famous 13-th problem is the following conjecture [Funahashi, 1989].

Conjecture [Hilbert,1900]. There are analytic functions of three variables which cannot be represented as a finite superposition of continuous functions of only two variables.

This conjecture was refused by Kolmogorov and Arnold in 1957, and they proved the following Theorem:

3. Theorem: Any continuous function $\boldsymbol{f}(\boldsymbol{x_1}, \ldots, \boldsymbol{x_n})$ of several variables defined on the cube $[0,1]^n$ ($n \geq 2$) can be represented in the form

$$f(x) = \sum_{j=1}^{2n+1} \chi_j \left(\sum_{i=1}^{n} \psi_{ij}(x_i) \right)$$

where χ_j, ψ_{ij} are continuous functions of one variable and ψ_{ij} are monotone functions which are not dependent on $\boldsymbol{f}$.

This Theorem was refined by Sprecher as follows:

4. Theorem: For each integer **n** ≥ 2, there exists a real, monotone increasing function $\psi(\boldsymbol{x})$, $\psi([0,1]) = [0,1]$, depending on **n** and having the property: for each preassigned number $\delta > 0$ there is a rational number ε, $0 < \varepsilon < \delta$, such that every real continuous function of **n** variables $\boldsymbol{f}(\boldsymbol{x})$, defined on $[0,1]^n$, can be represented as

$$f(x) = \sum_{j=1}^{2n+1} \chi \left[\sum_{i=1}^{n} \lambda^i \psi\left(x_i + \varepsilon(j-1)\right) + j - 1 \right]$$

where the function χ is real and continuous and λ is a constant independent of $\boldsymbol{f}$.

5. A link between the Sprecher Theorem and neural networks was first revealed by Hecht-Nielsen in 1987. He pointed out that the Sprecher Theorem means that: Any continuous mapping $f : x \in [0,1]^n \rightarrow \left(f_1(x), \ldots, f_m(x)\right) \in R^m$ is represented by a form of a two hidden layer neural network with hidden units whose output functions are ψ, χ_i (i= 1, …, m), where ψ is used for the first

hidden layer, χ_i are used for the second hidden layer and is given by the Sprecher Theorem for $\boldsymbol{f}_i(\boldsymbol{x})$. However the fact that the functions ψ_{ij} are highly nonsmooth and the functions χ_i depend on the specific function f and are not representable in a parameterized form, was part of the criticism made by Girosi & Poggio, who claimed that Kolmogorov's representation theorem is irrelevant for neural networks [Girosi, 1989]. In 1991 this was refuted by Kurkova, who proved that Kolmogorov's Theorem is indeed relevant: by using staircase-like functions of the form $\sum_I a_i \sigma(b_i x + c_i)$ where $\sigma(\ )$ is a sigmoidal function, for χ_i and ψ_i in the two hidden layer network it is indeed possible to approximate any continuous function arbitrarily well [Kurkova (1991), (1992)].

6. On the other hand in 1991 it was also shown independently by Hornik, Funahashi, and Cybenko, that a multilayer feed forward neural network with one or more hidden layers is sufficient in order to approximate any continuous nonlinear function arbitrarily well on a compact interval, provided sufficient hidden neurons are available. In contrast to Kurkova, they make use of advanced theorem from functional analysis. We will focus here on the results presented in [Horink, 1989]. In order to understand the following Theorem some preliminary definitions have to be introduced. Let C^r denote the set of continuous functions $R^r \to R$, ρ a metric to measure the distance between $f, g \in C^r$, A^r the set of all affine functios from

$$R^r \text{ to } R: \left\{ \mathrm{A}(x): R^r \to R: \mathrm{A}(x) = w^T x + b; x \in R^r \right\}.$$

so called $\sum$ networks and $\sum\Pi$ networks are then defined as

$$\sum^r(G) = \left\{ f: R^r \to R: f(x) = \sum_{j=1}^{q} \beta_j \, G\left(A_j(x) \right); x \in R^r, \beta_j \in R, A_j \in A^r \right\}$$

$$\sum\Pi^r(G) = \left\{ f: R^r \to R: f(x) = \sum_{j=1}^{q} \beta_j \prod_{k=1}^{l_j} G\left(A_{jk}(x) \right); x \in R^r, \beta_j \in R, A_{jk} \in A^r \right\}$$

where $G: R \to R$ is a Borel measurable function. A function ψ: **R**→ [0,1] is a

squashing function if ψ is nondecreasing, $\lim_{\lambda\to-\infty}\psi(\lambda)=0$ and $\lim_{\lambda\to\infty}\psi(\lambda)=1$ and has a conuntable number of discontinuities. Remarks that a $\sum^r(G)$network reduces to a stnandard classical multilayer feed forward neural network with one hidden layer and activation function ψ if $G=\psi\left(\sum^r(\psi)network\right)$.

The following definitions are necessary to what follows:

2.2.1 Definition:

A **s**ubset ***S*** of a metric space (X, ρ) is ρ- dense in a subset T if $\forall t\in T,\exists s\in S \;such\,that\; \rho(t,s)<\varepsilon$.

2.2.2 Definition:

$S\subset C^r$ is uniformly dense on compact in C^r if for all compact subsets $K\subset R^r$ the subset ***S*** is $\rho\kappa$-dense in C^r with $\rho\kappa(f,g)=\sup_{x\in K}|f(x)-g(x)|$. A sequence$\{f_n\}$ converges to f uniformly on compact if for all $K\subset R^r:\rho\kappa(f_n,f)\to 0\; as\; n\to\infty$.

In our case T corresponds to C^r and ***S*** to $\sum^r(G)$ or $\sum\Pi^r(G)$. The following theorem then holds:

7. Theorem Let G be any continuous nonconstant function from **R** to **R**. Then $\sum\Pi^r(G)$ is uniformly dense on compact in C^r.

Hence ∑Π(G) feed forward networks are capable of arbitrarily accurate approximation to any real-valued continuous function over a compact set and G may be any continuous nonconstant function here.

Now let μ be a probability measure defined on (R^r,B^r) with $B^r\subset R^r$ a Borel σ-field and M^r the set of all Borel measurable functions from $R^r\; to\; R$. Functions $f,g\in M^r$ are called μ-equivalent if $\mu\{x\in R^r:f(x)=g(x)\}=1$. Then the metric $\rho_\mu:M^r*M^r\to R^+$ is defined by $\rho_\mu(f,g)=\inf\{\varepsilon>0:\mu\{x:|f(x)-g(x)|>\varepsilon\}<\varepsilon\}$. Hence f

and g are close in the metric ρ_μ if and only if there is only a small probability that they differ significantly and f and g are μ-equivalent if $\rho_\mu(f,g)=0$ [Hornik, 1989].

The following Theorem then holds

8. Theorem For every squashing function ψ and every probability measure μ on $(R^r, B^r), \sum^r(\psi)$ is uniformly dense on compact in C^r and ρ_μ-dense in M^r [Hornik, 1989].

2.2.1 Remarks

1.The above theorem means that regardless of the dimension **r** of the input space and for any squashing function ψ, a feed forward neural network with one hidden layer can approximate any continuous function arbitrarily well in the ρ_μ metric. In the proof of Hornik's Theorems a central role is played by the Stone-Weierstrass theorem.

2. In addition to the previous Theorems, more refined Theorems were formulated by [Hornik, 1991]. More recently [Leshno, 1993] showed that a standard multilayer feed forward network with a locally bounded piecewise continuous activation function can approximate any continuous function to any degree of accuracy if and only if the network's activation function is not a polynomial.

3. The previous Theorems are existence theorems and are not constructive in the sense that no learning algorithms are presented and they give no or little information about the number of hidden unites to be used for the approximation. Furthermore, no comparision is made between neural networks and other universal approximators such as polynomial expansions in terms of network complexity.

4. The latter problem is addressed by [Barron, 1993]. It has been shown that the parsimony of a neural network parameterization is surprisingly advantageous in high-dimensional settings. Feed forward networks with one

hidden layer of sigmoidal activation function achieve an integrated squared error of order ***O*** (1/*n*), independent of the dimension of the input space, where *n* denotes the number of hidden neurons. The underlying function to be approximated is assumed to have a bound on the first moment of the magnitude distribution of the Fourier transform (smoothness property). On the other hand for a series expansion with *n* terms, in which only the parameters of a linear combination are adjusted (such as traditional polynomial, spline and trigonometric expansions), the integrated square error cannot be made smaller than $\boldsymbol{O}(1/n^{2/d})$ where *d* is the dimension of the input, for functions satisfying the same smoothness assumption. In order to formulate Barron's theorem, let $f(x)$ denote the class of functions on R^d for which there is a Fourier representation of the form

$$f(x)=\int_{R^d} \exp(iwx)\,\tilde{f}(w)dw.$$

for some complex-valued function $\tilde{f}(w)$ for which $w\tilde{f}(w)$ is integrable, and define

$$C_f=\int_{R^d} |w|\left|\tilde{f}(w)\right| dw.$$

Let $f_n=\sum_{k=1}^{n} c_k\,\psi(a_k x+b_k)+c_0$ with $a_k\in R^d, b_k, c_k\in R$ denote a linear combination of sigmoidal functions with squashing function ψ. Furthermore let μ denote a probability measure on the ball $B_r=\{x:|x|\le r\}$ with radius r > 0. then the following theorem could be proved.

5. Theorem: For every function ***f*** with C_f finite, and every $n\ge 1$, there exists a linear combination of sigmoidal functions $\boldsymbol{f}_n(\boldsymbol{x})$, such that

$$\int_{B_r} (f(x)-f_n(x))^2\,\mu(dx)\le\frac{k_f}{n}$$

where $k_f=\left(2rC_f\right)^2$. Hence the effects of the curse of dimensionality are avoided in terms of the accuracy of approximation [Barron, 1993].

6. Stability plays a very important role in control theory. It is a necessary condition for feasibility of the control system that the closed-loop system, consisting of a controller and the system to be controlled, is stable. Additionally, the controller must of course be designed in such a way that the behavior of the closed-loop system satisfies various requirements, e.g., with respect to speed and damping.

Sometimes stability of the solution is not an important issue, what is important to get a bounded output if the input is bounded. Examples of norms used in these cases are

$$\|x(t)\|_{\ell\infty} = \sup \|x(t)\| \tag{2.3}$$

$$\|x(t)\|_{\ell 2} = \left(\sum_{t=0}^{\infty} x^T(t)\, x(t) \right)^{\frac{1}{2}} \tag{2.4}$$

$$\|x(t)\|_{\ell p} = \left(\sum_{t=0}^{\infty} \|x(t)\|^p \right)^{\frac{1}{p}} \tag{2.5}$$

2.3 Feedback Linearization

In the area of nonlinear control theory, feedback linearization is a principle which has drawn much attention. The application is restricted to certain classes of systems, but these are actually not uncommon in practice. **The advantage of feedback linearization** is that the design can be used generally, in the sense that the same principle can be used on all systems of the right type. Moreover, extensions have been developed to take into account possible model inaccuracies, design and associated stability analysis is based on quite well established theory. See for example [Slotine, 1991], [Isidori, 1995], [Khalil, 1996] and [Chen, 1991].

2.3.1 The Basic Principle of Feedback Linearization

Feedback linearization is commonly discussed in a continuous-time framework. The fundamental assumption made about the system is that the model of the considered system can be written in the canonical form

$$\dot{x}(t)=\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\\ \vdots\\ \dot{x}_n(t)\end{bmatrix}=\begin{bmatrix}x_2(t)\\ x_3(t)\\ \vdots\\ \tilde{f}[x(t)]+\tilde{g}[x(t)]u(t)\end{bmatrix}, \tag{2.6}$$

with $\tilde{f}$ and $\tilde{g}$ being nonlinear functions of the states. t is now the actual time and not a multiple of the sampling period. If this form is not obtained directly when modeling the system, it must be possible to derive it through an appropriate diffeomorphic transformation [Slotine, 1991].

The system can be linearized by introducing of the following control redefinition (it is assumed that $\tilde{g}[x(t)]\neq 0$), [Khalil, 1991].

$$u(t)=\frac{w(t)-\tilde{f}[x(t)]}{\tilde{g}[x(t)]} \tag{2.7}$$

If complete knowledge about the states is available; either from measurements or from an observer, a pole placement type design is easily accomplished. Selecting the **virtual control input**, ***w***, as the reference plus a linear combination of the states results in a closed-loop system specified by

$$\dot{x}(t)=\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\\ \vdots\\ \dot{x}_n(t)\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & \cdots & 0\\ 0 & 0 & 1 & \cdots & 0\\ \vdots & & & \ddots & \vdots\\ -a_0 & -a_1 & \cdots & & -a_{n-1}\end{bmatrix}x(t)+\begin{bmatrix}0\\ 0\\ \vdots\\ 0\\ 1\end{bmatrix}r(t) \tag{2.8}$$

$$y(t)=\begin{bmatrix}1 & 0 & \cdots & 0\end{bmatrix}x(t),$$

corresponding to the transfer function model

$$H_{cl}(s)=\frac{1}{s^n+a_{n-1}s^{n-1}+\cdots a_0} \tag{2.9}$$

The coefficients $\{a_i\}_{i=0}^{n-1}$ will specify the characteristic polynomial; i.e., the poles of the linear time invariant closed-loop system. The connection to pole placement control is thus obvious.

Discretization of nonlinear systems is a quite involved action unless it is done by crude approximations. Here a somewhat pragmatic approach will be taken, similar to the one suggested in Chen and Khalil (1991). It is assumed that the system can be modeled as

$$\begin{aligned} y(t) &= f[y(t-1),\ldots,y(t-n),u(t-2),\ldots,u(t-m)] \\ &\quad + g[y(t-1),\ldots,y(t-n),u(t-2),\ldots,u(t-m)]u(t-1) \end{aligned} \tag{2.10}$$

or equivalently

$$\begin{aligned} x(t+1) &= \begin{bmatrix} x_1(t+1) \\ x_2(t+1) \\ \vdots \\ x_n(t+1) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ x_3(t) \\ \vdots \\ f[.]+g[.]u(t) \end{bmatrix} \\ y(t) &= x_n(t), \end{aligned} \tag{2.11}$$

with the state vector being defined by

$$x(t) = [y(t-n+1),\ldots,y(t-1),y(t)]^T \tag{2.12}$$

Assuming the functions f and g are known. Introduction of the following control redefinition will linearize the system at the sampling instants:

$$u(t) = \frac{w(t) - f[y(t),\ldots,y(t-n+1),u(t-1),\ldots,u(t-m+1)]}{g[y(t),\ldots,y(t-n+1),u(t-1),\ldots,u(t-m+1)]} \tag{2.13}$$

Selecting the virtual control input, w, as the reference plus an appropriate linear combination of past outputs again allows for an arbitrary assignment of the closed-loop poles. The control design can be regarded as a nonlinear counterpart to pole placement with full zero cancellation.

2.3.2 Feedback Linearization Using Neural Network Models

In case that the system is unknown, a model can be induced from data by letting two separate neural networks approximate the functions f and g

$$\hat{y}(t/\theta) = \hat{f}\left[y(t-1), \ldots, y(t-n), u(t-2), \ldots, u(t-m), \theta_f \right] + \hat{g}\left[y(t-1), \ldots, y(t-n), u(t-2), \ldots, u(t-m), \theta_g \right] u(t-1) \tag{2.14}$$

Derivation of a training method for determination of the weights in the two networks used for approximating f and g is straightforward. The prediction error approach requires knowledge of the derivative of the model output with respect to the weights. In order to calculate this derivative, the derivative of each network output with respect to the weights in the respective network must be determined first

$$\Psi_f\left(t, \theta_f\right) = \frac{\partial \hat{f}}{\partial \theta_f} \qquad \Psi_g\left(t, \theta_g\right) = \frac{\partial \hat{g}}{\partial \theta_g} \tag{2.15}$$

The derivative of the model output with respect to the weights is then composed of the derivatives of each network in the following manner

$$\Psi(t,\theta) = \frac{\partial \hat{y}(t/\theta)}{\partial \theta} = \begin{bmatrix} \Psi_f\left(t, \theta_f\right) \\ \psi_g\left(t, \theta_g\right) u(t-1) \end{bmatrix} \tag{2.16}$$

With this derivative in hand, any training methods can be used without further modification [Chen, 1991].

2.3.1 Remarks

The controller is related to the model-reference controller discussed previously: a nonlinear controller is designed to make the closed-loop system behave linearly according to a specified transfer function model.

Advantages and disadvantages of the neural-network-based feedback linearization method are briefly listed below.

Advantages:

1. Implementation is simple.
2. Only a model of the system to be controlled is required.
3. Tuning of the closed-loop response can be made without retraining of the model. An outer feedback can be introduced containing a linear pole placement controller.

Disadvantages:

1. Restricted to a particular class of systems. Difficult to resolve whether an unknown system actually belongs to this class.
2. Model structure selection is very complicated because two neural network architectures must be chosen.
3. Lack of design parameters for tuning of the controller.

2.3.3 Feedback control action

1. The main reason for using feedback control is to stabilize unstable systems and to reduce the influence from possible disturbances and model inaccuracies.

2. Using feedback to ensure that the system rapidly follows changes in the reference is not always good practice.

3. A rapid reference tracking obtained with feedback generally has the side effect that the controller becomes highly sensitive to noise.

Feedback linearization was proposed as a method for designing pole placement type controllers for a particular class of nonlinear systems. The neural network used for modeling the system have a specific structure in order to implement the controller. But we use another method (**Coordinate Transformation**) to translate the systems in the main theorems (**Theorem 2.7.1, Theorem 2.7.2)** of this chapter, which we will be discuss it below, but first we will show some remarks that we need it later.

2.4 Controllability of Dynamical System

System analysis generally consists of two part; quantitative and qualitative.

In quantitative study, we are interested in the exact response of the system to certain input and initial conditions. The quantitative study is concerned with the general properties of a system. In qualitative properties of linear dynamical equations is introduced here: controllability, which is very important in the study of control.

2.4.1 Definition (Controllable system):

A system is said to be controllable at time t_0 if it is possible by means of an unconstrained control vector to transfer the system from any initial state $x(t_0)$ to any other state in a finite time interval, [Jayc, 1968].

2.4.2 Definition (controllability):

A system $\dot{x} = f(x,u,t)$ is completely controllable if any initial state $x(t_0)$ can be transferred to any final state $x(t_1)$ by means of some control $u(t)$ over a finite interval $t_0 \le t \le t_1$ [Kolman, 1984].

2.4.1 Theorem

Consider the linear time-invariant system:

$$\dot{x} = \mathrm{A}\,x(t) + \mathrm{B}\,u(t) \qquad (2.17)$$

Where $x \in R^n$ is the state vector, $u \in R^m$ is the control, $\mathrm{A} \in R^{nxn}$ *and* $\mathrm{B} \in R^{nxm}$ are constant matrices.

The necessary and sufficient condition for the complete controllability of the system (2.17) is the $n \times nm$ matrix $\rho(\mathrm{A},\mathrm{B}) \equiv \left[\mathrm{B}, \mathrm{AB}, \ldots, \mathrm{A}^{n-1}\mathrm{B}\right]$ has rank n, [Chen, 1984].

2.4.2 Theorem

A continuous time system described by (2.17) is completely state controllable if and only if the composite $n \times nm$ matrix $\rho(\mathrm{A}, \mathrm{B}) \equiv \left[\mathrm{B}, \mathrm{AB}, \ldots, \mathrm{A}^{n-1}\mathrm{B} \right]$ is of rank n [Ogata, 1996].

2.4.3 Definition:

Let A be an $n \times n$ matrix and B be an $n \times m$ matrix, then we say that the pair (A, B) is completely state controllable if the system:

$$\dot{x} = \mathrm{A}\, x(t) + \mathrm{B}\, u(t)$$

Is completely state controllable, [Ogata, 1967].

2.5 Lyapunov Stability:

We present here the Lyapunov methods of stability analysis (the first method and the second method) which are applicable to both linear and nonlinear system. Our attention will be devoted to the second of Lyapunov method, which provides stability information on linear and non-linear differential equations without solving them, hence the second method is called the direct method of Lyapunov, the direct method is most useful for investigating stability of non-linear systems. It gives sufficient conditions for asymptotic stability of equlilibrium states of non-linear systems and gives necessary and sufficient conditions for asymptotic stability of equilibrium states of time-invariant systems, [Jayc, 1968].

2.5.1 Definition (Equilibrium States):

Consider the dynamical system $\dot{x} = f(x,t)$, a state x_e, where $f(x_e, t) = 0, \forall t$ is called an equilibrium state of the system, [Ogata, 1967].

2.5.2 Definition (Lyapunov Stability):

An equilibrium state x_e of the dynamical system $\dot{x} = f(x,t)$ is stable (or stable in the sense of Lyapunov) if for every $\varepsilon > 0$, there exists $\delta > 0(\delta(\varepsilon, t_0))$ such that

$$\|x_0 - x_e\| \le \delta \ implies \|x(t;x_0) - x_e\| \le \varepsilon, \ for\ all\ t \ge t_0 \qquad (2.18)$$

Where $\|\cdot\|$ denotes the Euclidean norm of a vector, [Jayc, 1968].

2.5.3 Definition (Asymptotic Stability):

An equilibrium state x_e of the system $\dot{x} = f(x,t)$ is asymptotically stable if:

1. It is stable in the sense of Lyapunov.

2. For all t_0, there exists a $\rho(t_0) > 0$ (possibly depending on t_0) such that

$$\|x_0 - x_e\| < \rho \ implies that \ \|x(t,x_0) - x_e\| \to 0 \ as\ t \to \infty \qquad (2.19)$$

[Jayc, 1968].

2.5.4 Definition (Asymptotic Stability in the Large):

The nominal solution $x_0(t)$ of the system $\dot{x}(t) = f(x(t),t)$ is asymptotically stable in the large if :

1. It is stable in the sense of Lyapunov.

2. For any $x_0(t)$ and any $t_0, \|x(t) - x_0(t)\| \to 0\ as\ t \to \infty$.

A solution that is asymptotically stable in the large has the property that all other solutions eventually approach it, [Jayc, 1968].

2.5.1 Theorem (Stability of Time Invariant System):

The time-invariant linear system:

$$\dot{x}(t) = \mathrm{A}\, x(t)$$

is stable in the sense of Lyapunov if and only if :

a. All of the characteristic values of A has non-positive real parts, and,

b. To any characteristic value on the imaginary axis with multiplicity m, there correspond exactly m characteristic vectors of the matrix A, [Huibert, 1972].

2.5.2 Theorem:

The time-invariant system:

$$\dot{x}(t) = \mathrm{A}\, x(t)$$

Is asymptotically stable if and only if all of the characteristic values (eigenvalues) of A have strictly negative real part [Huibert, 1972].

2.5.3 The Direct Method of Lyapunov:

The second method of Lyapunov attempts to give information on the stability of equilibrium state of linear and non-linear systems without any prior knowledge of their solutions.

The essence of the second method of Lyapunov is given in the following theorem:

2.5.3.1 Theorem (Lyapunov Main Stability Theorem):

Consider the system:

$$\dot{x}(t) = f(x(t),t)$$

and suppose that $f(0,t) = 0, \forall\, t$.

Suppose also, that there exists a scalar function $V(x,t)$ which has continuous first partial derivatives. If $V(x,t)$ satisfies the following conditions:

1. $V(x,t)$ is positive definite , namely $V(0,t)$, and $V(x,t) \geq \alpha(\|x\|) \geq 0,\ for\ all\ x \neq 0, and\ all\ t, where\ \alpha$ is a continuous, non-decreasing scalar function, such that $\alpha(0) = 0$.

2. The total derivative $\dot{V}$ is negative for all $x \neq 0$, and all t or $\dot{V}(x,t) \leq -\gamma\|x\| < 0,\ for\ all\ x \neq 0\ and\ all\ t,\ where\ \gamma$ is a continuous, non-decreasing scalar function such that $\gamma(0) = 0$.

3. There exists a continuous non-decreasing function such that $\beta(0) = 0\ for\ all\ t, V(x,t) \leq \beta\|x\|$.

4. $\alpha(\|x\|)$ approaches infinity as $\|x\|$ increases indefinitely, or

$$\alpha(\|x\|) \to \infty \ as \ \|x\| \to \infty$$

Then, the origin of the system $x = 0$ is uniformly asymptotically stable in the large, [Ogata, 1996].

2.5.3.2 Theorem:

If there exists a scalar function $V(x,t)$ with contiuous first partial derivatives satisfying the following conditions:

(1)
$$\begin{aligned} &V(x,t) > 0, \ for \ all \ x \neq 0 \ in \ \Omega \ and \ all \ t \\ &V(0,t) = 0, \ for \ all \ t \end{aligned} \qquad (2.20)$$

(2)
$$\begin{aligned} &\dot{V}(x,t) < 0, \ for \ all \ x \neq 0 \ \ in \ \ \Omega \ and \ all \ t \\ &\dot{V}(0,t) = 0, \forall t \end{aligned} \qquad (2.21)$$

where Ω is the region (can be the entire state space), which includes the origin. Then, the origin of the system $\dot{x}(t) = f(x(t),t)$ is uniformly asymptotically stable, [Ogata, 1967].

2.6 Mathematical Preliminaries

2.6.1 Definition:

A norm is a function which assign to every vector x in a given vector space a real number denoted by $\|x\|$, such that:

1. $\|x\| \geq 0 \, and \, \|x\| = 0 \ \ if \ and \ only \ if \ \ x = 0$ (2.22)

2. $\|\alpha x\| = |\alpha| \ \|x\|, \alpha \in R$ is a scalar and $|\alpha|$ is the absolute value of α (2.23)

3. $\|x_1 + x_2\| \leq \|x_1\| + \|x_2\|, \forall \ x_1, x_2$ (2.24)

4. $|\langle x_1, x_2 \rangle| \leq \|x_1\| \|x_2\|$ (2.25)

The Euclidean norm of a vector $x \in R^n$ is defined as:

$$\|x\| = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2} \qquad (2.26)$$

(a) The Euclidean norm of $n \times n$ matrix is defined as:

$$\|\mathrm{A}\| = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2} \qquad (2.26a)$$

Where $|a_{ij}|$ is the absolute value of a_{ij}

The Euclidean norm of $n \times n$ matrix is also defined as:

$$\|\mathrm{A}\| = \left(\lambda_{\max}\left(\mathrm{A}^T\ \mathrm{A}\right)\right)^{1/2} \tag{2.26b}$$

Where A^T is stands for the transpose of A, $\lambda_{\max}$ is standing for the maximum eigenvalue of $\left(\mathrm{A}\ \mathrm{A}^T\right)$, [Chen, 1984].

2.6.2 Preliminary Remarks and Definitions:

Let R denote real number, R^n denote the real n vector, and $R^{m\times n}$ denote the real $m \times n$ matrices. Let S be a compact simple connected set of R^n. With map $f : S \to R^m$, define $C(S)$ as the space such that f is continuous. The initial condition is $x_0 \equiv x(t_0)$, let the equilibrium point x_e, *and* U_{x_e} be the neighborhood of x_e.

1. Definition (Vector and Matrix Norms):

By $\|\ \|$ is denoted any suitable vector norm. when it is required to be specific, we denote the p-norm by $\|\ \|_p$. The supremum norm of norm of $f(x)$, over S, is defined as

$$\sup_{x\in S} \|f\ (x)\|, f\ : S \to R^m, x \in X \tag{2.27}$$

Given $\mathrm{A} = [a_{ij}], \mathrm{B} \in \Re^{m\times n}$ the Frobenius norm is defined by

$$\|\mathrm{A}\|_F^2 = tr\left(\mathrm{A}^T\ \mathrm{A}\right) = \sum_{i,j} a_{ij}^2 \tag{2.28}$$

The Frobenius norm is compatible with the 2-norm so that

$$\|\mathrm{A}\,x\|_2 \le \|\mathrm{A}\|_F\ \|x\|_2 .$$

The associated inner product is $\langle \mathrm{A}, \mathrm{B} \rangle_F = tr\left(\mathrm{A}^T, \mathrm{B}\right)$. Suppose A is positive definite, then for any $\mathrm{B} \in R^{m\times n}$

$$tr\left(\mathrm{B}\,\mathrm{A}\,\mathrm{B}^T\right) \ge 0, \tag{2.29}$$

$$tr\left[\tilde{\mathrm{A}}^T\left(\mathrm{A} - \tilde{\mathrm{A}}\right)\right] \le \|\tilde{\mathrm{A}}\|_F\ \|\mathrm{A}\|_F - \|\tilde{\mathrm{A}}\|_F^2 \tag{2.30}$$

and

$$\frac{d}{dt}\{tr(\mathrm{A}(x))\} = tr\left(\frac{d\mathrm{A}(x)}{dt}\right) \tag{2.31}$$

[Lewis,1993].

2. Definition(Uniformly Ultimate Boundness)(UUB)):

Consider the non-linear system

$$\dot{x} = g(x,t) \tag{2.32}$$

With state $x(t) \in \Re^n$. The equilibrium point x_e is said to be uniformly ultimately bounded if there exists a compact set $S \subset \Re^n$, so that for all $x_0 \in S$ there exists an $\varepsilon > 0$, and a number $T(\varepsilon, x_0)$ such that $\|x(t) - x_e\| \le \varepsilon$ *for all* $t \ge t_0 + T$. That is, after a transition period T, the state $x(t)$ remains within the ball of radius ε around x_e [Lewis, 1999].

2.6.2.1 Remark

Leangs Shich and Yates et al in 1983. proposed a method to transform system into block companion form as follows:

Consider the linear time-invariant system:

$$\dot{x} = \mathrm{A}x + \mathrm{B}u \tag{2.33}$$

where $x \in R^n, u \in R^m, \mathrm{A} \in R^{nxn}, \mathrm{B} \in R^{nxm}$.

If the rank of the block controllability test matrix $\rho(\mathrm{A}, \mathrm{B}) \equiv \left[\mathrm{B}, \mathrm{AB}, \ldots, \mathrm{A}^{n-1}\mathrm{B}\right]$ is n, then the system (2.33) is completely block controllable companion form:

$$\dot{x}_C = \mathrm{A}_C x_C + \mathrm{B}_C u$$

where

$$A_C = T_C A T_C^{-1} = \begin{bmatrix} O_m & I_m & O_m & \cdots & O_m & O_m \\ O_m & O_m & I_m & \cdots & O_m & O_m \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ O_m & O_m & O_m & \cdots & O_m & I_m \\ -A_n & -A_{n-1} & -A_{n-2} & \cdots & -A_2 & -A_1 \end{bmatrix}$$

Where $-A_i, i = 1,\ldots n$ are constant values.

$$B_C = T_C B = [O_m, \quad O_m, \quad \ldots, \quad O_m, \quad I_m]^T$$

$$x_C = T_C x$$

The similarity transformation matrix T_C is given by **[**Leangs, 1983]:

$$T_C = \left[T_{c1} \quad T_{c1}A \quad T_{c1}A^2 \quad \ldots \quad T_{c1}A^{n-1} \right]^T$$

where

$$T_{c1} = B_c^T \ \rho^{-1}(A, B)$$

2.6.2.2 Remark (Coordinate Transformation)

It is recalled that the state model for a system is not unique, but depends on the choice of a set of state variables. To simplify analysis and design for a system (2.33) can be written as:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

it is often beneficial to define a new state variable z by a coordinate transformation:

$$x = Tz \qquad (2.34)$$

where T is a non-singular matrix.

The state model corresponding to these new state variable is found by substituting (2.34) into (2.33).

$$T\,\dot{z}(t) = AT\,z(t) + Bu$$

$$\dot{z}(t) = \overline{A}\ z(t) + \overline{B}\ u \qquad (2.35)$$

where $\overline{A} = T^{-1}AT, \quad \overline{B} = T^{-1}B$

The definition of a new set of internal state variables should evidently not affect the eigenvalues or input-output behavior, this may verified by evaluating the characteristic equation of the transformed system [John, 1990]:

$$\left|s\mathrm{I}-\overline{\mathrm{A}}\right| = \left|s\mathrm{I}-T^{-1}\mathrm{A}T\right| = \left|T^{-1}(s\mathrm{I}-\mathrm{A})T\right| = \left|T^{-1}\right| \left|s\mathrm{I}-\mathrm{A}\right| \left|T\right|$$
$$= \left|s\mathrm{I}-\mathrm{A}\right| \left|T^{-1}\right| \left|T\right| = \left|s\mathrm{I}-\mathrm{A}\right|$$

2.6.2.1 Example:

Choosing the $n\times n$ invertable transformation matrix T such that $T\,\mathrm{B} = \begin{bmatrix} 0 \\ \mathrm{I}_m \end{bmatrix}$, I_m is a unit matrix of dimension m, can be proposed by ([Leangs, 1986] [Huibert, 1972] [Ian, 1986]).

It can be chosen such that $T^{-1} = [L:\mathrm{B}]$, *where* L is selected such that the inverse exists.

To show this consider the linear system

$$\dot{x} = \mathrm{A}x + \mathrm{B}\,u \qquad (2.36)$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -0.428 & -0.339 & 0 \\ -2.939 & -1.011 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0.15 \\ -1.011 \\ 1 \end{bmatrix} u(t) \qquad (2.37)$$

where n=3, m=1, such that

$$x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$$

$$\mathrm{A} = \begin{bmatrix} -0.428 & -0.339 & 0 \\ -2.939 & -1.011 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad \mathrm{B} = \begin{bmatrix} 0.15 \\ -1.011 \\ 1 \end{bmatrix}$$

suppose

$$z = T\,x$$

where $z = [z_1 \quad z_2 \quad z_3]^T$

such that we have a new system

$$\dot{z} = TAT^{-1} z + T\,B\,u(t)$$

Choose an $n \times n$ invertable matrix T such that $T\,B = \begin{bmatrix} 0 \\ I_m \end{bmatrix}$

The choice can be such that $T^{-1} = [L:B]$, *where* L is selected such that the inverse exists. T is found to be:

$$T = \begin{bmatrix} -3.1728 & -0.4707 & -1.1447 \\ 2.7414 & 0.4067 & 0 \\ -2.3688 & -1.3406 & 0 \end{bmatrix}$$

and

$$T^{-1} = \begin{bmatrix} 0.000 & 0.4944 & 0.15 \\ 0 & -0.8735 & -1.011 \\ -1.8736 & -1.011 & 1 \end{bmatrix}$$

hence we get

$$TAT^{-1} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0.5636 & -1.4390 \end{bmatrix}, \quad TB = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Finally, the new system will be as follows

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= z_3 \\ \dot{z}_3 &= 0.5636 z_2 - 1.439 z_3 + u(t) \\ y &= z_1 \end{aligned} \tag{2.38}$$

So far, we have discussed some necessary requirements that are needed in what follows, and hence we are in position to present and develop the main results of this work. The representation of the main theorem are discussed as follows:

2.7 Main Problem

2.7.1 Theorem

Consider the non-linear system

$$\dot{x} = \mathrm{A}x + \mathrm{B}g(x)u + \mathrm{B}f(x) \tag{2.39}$$

where $x \in R^n$, and the smooth functions $f \in R^m, g \in R^m$, $u \in R^m$ is the control, $\mathrm{A} \in R^{nxn}, \mathrm{B} \in R^{nxm}$ are constant matrices. Assuming that

1. The pair (A,B) is controllable matrix.
2. The function $g(x) = \lfloor g_1(x), \ g_2(x), \ \ldots, \ g_m(x) \rfloor$ is known such that $g(x) \neq 0, \forall x$, and $\|g(x)\| > \varepsilon$ where $\varepsilon \in R^+$ and $g_i(x) \neq 0, \forall x, i = 1,\ldots,m$.
3. Let the desired state be defined by $y_d(t) = \left[y_d \quad \dot{y}_d \quad \cdots \quad y_d^{(n-1)} \right]$, such that $\|y_d(t)\| \leq Q$, for some scalar Q.
4. The uncertain nonlinear function $f(x)$ may be estimated by $\hat{f}(x)$ such that $\|f(x) - \hat{f}(x)\| \equiv \|\tilde{f}(x)\| \leq f_m(x)$ for some bounds function $f_m(x)$.
5. Let the filtered tracking error can be defined as

$$r = K^T e \tag{2.40}$$

Where $e(t) = x(t) - y_d(t)$ be the tracking error of the system in (2.39), and $K = \lfloor k_1, k_2, \ldots, k_{n-1}, 1 \rfloor$ is approximate chosen coefficient vector.

6. Set the nonlinear neuro-controller by

$$u(t) = u_1 + u_2 \tag{2.41}$$

Where

$$u_1(x(t)) = W(x(t)) - \hat{u}_2(x(t)) \tag{2.42}$$

Where $W(t)$ is chosen as the control law of the system (2.39), and u_2 is the

nonlinear control function which we will approximate by the general neural network property.

7. Let the following structure of artificial neural network have been adapter where

- Consider two-layer neural network, consisting of two layers of tunable weights. The hidden layer has L neurons, and the output layer has m neurons.
- The first layer neural network have the input which is chosed as $x_{NN} = \lfloor y_d, e \rfloor^T, v = \lfloor v_{ji} \rfloor, \quad j = 1,2,\ldots,n; \quad i = 1,2,\ldots,L$ as the weights of the first layer neural network which we will chose it randomly, and $b_0 = [b_{01}, b_{02}, \ldots, b_{0L}]^T$ as the bias of the first layer neural network which we will chose it randomly too.
- Choose $\sigma(.)$ as any continuous sigmiodal function which be the activation function of the network in the first layer and in the hidden layer (second layer).
- the output of the first layer will be defined as following

$$q_i = \sigma\left(v^T x_{NN} + b_0\right) \quad i = 1,2,\ldots,L \tag{2.43}$$

- the input to the hidden layer (second layer) of the neural network will be the output of the first layer (q_i).
- w is neural network adjusted weight (hidden layer neural network weights), and it is assumed that they are bounded so that $\|w\| \le w_m$, with w_m known bounds. And $\hat{w}$ are the estimated neural network weights which be provided by the neural network tuning algorithm as

$$\dot{\hat{w}} = s\sigma\left(v^T x_{NN}\right) r \, \mathrm{B}\, g - Ks\|r\|\hat{w} \tag{2.44}$$

where $s = s^T > 0$, any constant matrices representing the learning rates of the neural network.

K, small scalar positive design parameter.

Hence, the neural network weights approximation error is

$$\tilde{w} = w - \hat{w} \tag{2.45}$$

- Neural network universal approximation property defines that any continuous function can be approximated arbitrarily well using a linear combination of sigmoidal functions, such that the output of the neural network $u_2(x)$ can be defined as

$$u_2(x) = w^T \sigma\left(v^T x_{NN} + b_0\right) + \varepsilon(x) \tag{2.46}$$

Where the $\varepsilon(x)$ is the neural network approximation error.

Implementer neural network $\hat{u}_2$, is actually an approximation of the ideal neural network (2.49) and is given by

$$\hat{u}_2 = \hat{w}^T \sigma\left(v^T x_{NN} + b_0\right) \tag{2.47}$$

If the following conditions are satisfied

$$\|r\| \geq \frac{\frac{K}{4} w_m^2 + \varepsilon_n}{k_{v\min}} \tag{2.48}$$

or

$$\|\tilde{w}\|_F \geq \left[\frac{\sqrt{\frac{K}{4} w_m^2 + \varepsilon_n} + \frac{1}{2} w_m}{K} \right]^{\frac{1}{2}} \tag{2.49}$$

Then the filtered error $r(t)$ and the neural network weights $\hat{w}$ are (Uniformly Ultimate Boundness).

Proof:

Let

$$z = T x$$

where $z \in R^n$, and T is any suitable invertible matrix (The choice can be such that $T^{-1} = [L : \mathrm{B}]$, *where* L is selected such that the inverse exists).

.

$$\begin{aligned}\dot{z} &= T\dot{x} = T\left(\mathrm{A}\,x + \mathrm{B}\,g\,u + \mathrm{B}f\right)\\ &= T\left(\mathrm{A}T^{-1}z + \mathrm{B}\,g\,u + \mathrm{B}\,f\right)\\ &= T\,\mathrm{A}T^{-1}z + T\,\mathrm{B}\,g\,u + T\,\mathrm{B}\,f\end{aligned} \tag{2.50}$$

Since (A,B) is controllable by theorem (2.4.1), we have

$$T\,\mathrm{A}T^{-1} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0\\ 0 & 0 & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & 1\\ \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_n \end{bmatrix} \qquad T\,\mathrm{B} = \begin{bmatrix} \mathrm{O}_m\\ \mathrm{O}_m\\ \vdots\\ \mathrm{I}_m \end{bmatrix}$$

Where $\alpha_1, \alpha_2, \ldots, \alpha_n$ are sutable constant.

Hence

$$\begin{bmatrix} \dot{z}_1\\ \dot{z}_2\\ \vdots\\ \dot{z}_{n-1}\\ \dot{z}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0\\ 0 & 0 & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & 1\\ \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_n \end{bmatrix} \begin{bmatrix} z_1\\ z_2\\ \vdots\\ z_{n-1}\\ z_n \end{bmatrix} + \begin{bmatrix} \mathrm{O}_m\\ \mathrm{O}_m\\ \vdots\\ \mathrm{O}_{m-1}\\ \mathrm{I}_m \end{bmatrix} \begin{bmatrix} g_1\\ g_2\\ \vdots\\ g_{m-1}\\ g_m \end{bmatrix} u(t) + \begin{bmatrix} \mathrm{O}_m\\ \mathrm{O}_m\\ \vdots\\ \mathrm{O}_{m-1}\\ \mathrm{I}_m \end{bmatrix} \begin{bmatrix} f_1\\ f_2\\ \vdots\\ f_{m-1}\\ f_m \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \dot{z}_1\\ \dot{z}_2\\ \vdots\\ \dot{z}_{n-1}\\ \dot{z}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0\\ 0 & 0 & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & 1\\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} z_1\\ z_2\\ \vdots\\ z_{n-1}\\ z_n \end{bmatrix} + \begin{bmatrix} 0\\ 0\\ \vdots\\ 0\\ g_m \end{bmatrix} u + \begin{bmatrix} 0\\ 0\\ \vdots\\ 0\\ f_m \end{bmatrix} + \begin{bmatrix} 0\\ 0\\ \vdots\\ 0\\ \alpha_1 z_1 + \alpha_2 z_2 + \ldots + \alpha_n z_n \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \dot{z}_1\\ \dot{z}_2\\ \vdots\\ \dot{z}_{n-1}\\ \dot{z}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0\\ 0 & 0 & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \ddots & 1\\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_1\\ z_2\\ \vdots\\ z_{n-1}\\ z_n \end{bmatrix} + \begin{bmatrix} 0\\ 0\\ \vdots\\ 0\\ g_m\,u(t) \end{bmatrix} + \begin{bmatrix} 0\\ 0\\ \vdots\\ 0\\ f_m + \alpha_1 z_1 + \alpha_2 z_2 + \ldots + \alpha_n z_n \end{bmatrix}$$

Let $F(z) = f_m + \alpha_1 z_1 + \alpha_2 z_2 + \ldots + \alpha_n z_n$ (2.51)

$$\Rightarrow \begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \vdots \\ \dot{z}_{n-1} \\ \dot{z}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-1} \\ z_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ g_m u(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ F(z) \end{bmatrix}$$

and hence, we have the following system

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= z_3 \\ &\vdots \\ \dot{z}_n &= g_m\, u(t) + F(z) \\ y &= z_1 \end{aligned} \tag{2.52}$$

with $z = \lfloor z_1, z_2, \ldots, z_n \rfloor_{nx1}^T$, $F : R^n \to R$, an unknown smooth function; $g_m : R^n \to R$, a known smooth function; $u(t)$ the control.

Define the state tracking error $e_z(t)$ as

$$e_z = z - y_d \tag{2.53}$$

where $y_d(t)$ is the desired state vector which we can defined as

$$y_d(t) = \begin{bmatrix} y_d & \dot{y}_d & \cdots & y_d^{(n-1)} \end{bmatrix}$$

hence $\Rightarrow e_z = z - y_d \equiv T\,x - y_d$

Differentiation yields

$$\dot{e}_z = T\,\dot{x} - y_d \tag{2.54}$$

Using equation (2.39) we get

$$\begin{aligned} \dot{e}_z &= T(\mathrm{A}\,x + \mathrm{B}\,g\,u + \mathrm{B}f) - \dot{y}_d \\ &\equiv T\left(\mathrm{A}\left(e_x - y_d\right) + \mathrm{B}\,g\,u + \mathrm{B}f\right) - \dot{y}_d \\ &\equiv T\,\mathrm{A}\,e_x - T\mathrm{A}\,y_d + T\,\mathrm{B}\,g\,u + T\,\mathrm{B}\,f - \dot{y}_d \end{aligned}$$

where $e_x = x - y_d$ as we defined it above, and since

$$T\,\mathrm{B}=\begin{bmatrix}\mathrm{O}_m\\ \mathrm{O}_m\\ \vdots\\ \mathrm{I}_m\end{bmatrix}\Rightarrow T\,\mathrm{B}\,g=\begin{bmatrix}\mathrm{O}_m\\ \mathrm{O}_m\\ \vdots\\ \mathrm{I}_m\end{bmatrix}\begin{bmatrix}g_1\\ g_2\\ \vdots\\ g_m\end{bmatrix}=\begin{bmatrix}0\\ 0\\ \vdots\\ g_m\end{bmatrix},\quad T\,\mathrm{B}\,f=\begin{bmatrix}\mathrm{O}_m\\ \mathrm{O}_m\\ \vdots\\ \mathrm{I}_m\end{bmatrix}\begin{bmatrix}f_1\\ f_2\\ \vdots\\ f_m\end{bmatrix}=\begin{bmatrix}0\\ 0\\ \vdots\\ f_m\end{bmatrix}$$

And if we suppose that

$$u=\frac{1}{g_m}\left(u_1+u_2\right) \tag{2.55}$$

Where by condition (2) above $g_i(x)\neq 0,\forall\, x$.

The system (2.53) and the action (2.55) leads to the following

$$\dot{e}_z=T\,\mathrm{A}e_x+T\,y_d+\left[u_1+u_2\right]+T\,\mathrm{B}f-\dot{y}_d \tag{2.56}$$

and, from equation (2.42) we choose that

$$W=-T\,\mathrm{B}\,f-T\,y_d+\dot{y}_d+T\,K\,\mathrm{B}e_x-T\,K\,\mathrm{B}e_x \tag{2.57}$$

Such that equation (2.56) can be modified as

$$\dot{e}_z=T\left(\mathrm{A}+K\,\mathrm{B}\right)e_x \tag{2.58}$$

Since (A,B) is a controllable pair by condition (1). To make the convergent faster define the filtered tracking error of the system (2.52) and by equation (2.40) we have the following

$$r_z=K^T e_z \tag{2.59}$$

where $K=\left[k_1,k_2,\ldots,k_{n-1},1\right]$ is approximate chosen (filter coefficients vector), so that $e\to 0\quad as\ r\to 0.$ Then the time derivative of the filtered error can be written as

$$\dot{r}=g_m\,u(t)+F(z)+Y_d \tag{2.60}$$

where $Y_d=-y_d^{(n)}+\sum_{i=1}^{n-1}k_i\,e_{i+1}$, and $F(z)$ is defined in equation (2.51)

Similarly in terms of the filtered tracking error, the above system dynamics (by using equation (2.41)), can be described as follows

$$\dot{r} = g_m \left[u_1 + u_2 \right] + F(z) + Y_d \tag{2.61}$$

Using condition (4) above and equation (2.60), the tracking control law W which be chosen in equation (2.57) can be written as

$$W = \frac{1}{g_m}\left[-\hat{F}(z) - Y_d - k_v r + \alpha \right] \tag{2.62}$$

where $\hat{F}$ is the fixed approximation of the functional F, k_v is the feedback gain, and assuming that α be the robust term chosen for the disturbance rejection which can be defined as

$$\alpha(t) = -f_m(z)\, sign(r) \tag{2.63}$$

where

$$sign(r) = \begin{cases} 1 & r > 0 \\ 0 & r \le 0 \end{cases} \tag{2.64}$$

$sign(\bullet)$ is the standard sign function. Using control law in equations (2.62), (2.42), (2.45), and substituting into equation (2.61) we get

$$\dot{r} = \tilde{F}(z) + T\,\mathrm{B}\, g\left\{ \tilde{w}^T \sigma\left(v^T x_{NN} + b_0 \right) \right\} + \alpha - k_v r + \varepsilon \tag{2.65}$$

Setting that Lyapunov function candidate as

$$V = \frac{1}{2} r^2 + \frac{1}{2} tr\left(\tilde{w}^T s^{-1} \dot{\tilde{w}} \right) \tag{2.66}$$

where r is the filtered tracking error and $\tilde{w}$ is the neural network approximation error which is defined in equation (2.45), such that

$$\dot{\tilde{w}} = \dot{w} - \dot{\hat{w}} \tag{2.67}$$

Where $\dot{w}$ is the neural network ideal weights, and $\dot{\tilde{w}}$ is the estimated neural network weights which provided by the neural network tuning algorithm as defined in equation (2.44), where

$$\dot{V} = \partial V\left(r(t) \right)/\partial t + \partial V\left(\tilde{w}(t) \right)/\partial t = \frac{\partial V}{\partial r} \cdot \frac{dr}{dt} + \frac{\partial V}{\partial \tilde{w}} \cdot \frac{d\tilde{w}}{dt}$$

$$\Rightarrow \quad \dot{V} = r\,\dot{r} + tr\left(\tilde{w}^T s^{-1} \dot{\tilde{w}} \right) \tag{2.68}$$

Hence substitution equation (2.65) into equation (2.68) yields

$$\begin{aligned}\dot{V} &= -k_v r^2 + r\tilde{F} + rT\,\mathrm{B}\,g\,\tilde{w}^T\,\sigma\left(v^T\,x_{NN}\right) + r\alpha + r\varepsilon + tr\left(\tilde{w}\,s^{-1}\,\dot{\tilde{w}}\right) \\ &= -k_v r^2 + r\left\{\tilde{F} + \alpha + \varepsilon\right\} + tr\left(\tilde{w}^T\left\{s^{-1}\dot{\tilde{w}} + \sigma\left(v^T\,x_{NN}\right)rT\,\mathrm{B}\,g\right\}\right)\end{aligned} \tag{2.69}$$

Applying the neural network tuning rules (equation (2.44)), then the derivative of Lyapunov function is simplified to

$$\dot{V} = -k_v r^2 + r\left\{\tilde{F} + \alpha + \varepsilon\right\} + K\|r\|tr\left(\tilde{w}^T\,\hat{w}\right) \tag{2.70}$$

Using equation (2.63) one has the following

$$\dot{V} \le -k_{v\min}\|r\|^2 + K\|r\|tr\left(\tilde{w}^T\left(w - \tilde{w}\right)\right) - \|r\|\,f_m + \|r\|\tilde{F} + \|r\|\varepsilon_n \tag{2.71}$$

Using the inequality

$$tr\left[\tilde{x}^T\left(x - \tilde{x}\right)\right] \le \|\tilde{x}\|_F\,\|x\|_F - \|\tilde{x}\|_F^2 \tag{2.72}$$

Using the inequality (2.72), such that the inequality (2.71) can be written as

$$\dot{V} \le -k_{v\min}\|r\|^2 + K\|r\|\left(\|\tilde{w}\|_F\,\|w\|_F - \|\tilde{w}\|_F^2\right) + \|r\|\varepsilon_n \tag{2.73}$$

(From using $\|w\|_F \le w_m$), and from inequality (2.73)we have that

$$\begin{aligned}\dot{V} &\le \|r\|\left\{-k_{v\min}\|r\| + K\left(\|\tilde{w}\|_F\,w_m - \|\tilde{w}\|_F^2\right) + \varepsilon_n\right\} \\ &= \|r\|\left\{-k_{v\min}\|r\| - K\left(\|\tilde{w}\|_F - \frac{1}{2}w_m\right)^2 + \frac{1}{4}Kw_m^2 + \varepsilon_n\right\}\end{aligned} \tag{2.74}$$

which is guaranteed to remain negative as long as (conditions (2.48) (2.49) above satisfied as following

$$\|r\| \ge \frac{\frac{K}{4}w_m^2 + \varepsilon_n}{k_{v\min}}$$

Or

$$\|\tilde{w}\|_F \geq \left[\frac{\sqrt{\frac{K}{4} w_m^2 + \varepsilon_n} + \frac{1}{2} w_m}{K} \right]^{\frac{1}{2}}$$

Such that, since $V(0)=0, V(r)\geq 0 \; and \; \dot{V}(r)<0$, which means that the stable of the system for tracking filter is guarantied and if $r \to 0, as\ t \to \infty$. Therefore $z \to 0, as\ t \to \infty$ and hence $x \to 0, as\ t \to \infty$ so that the stability of the system (2.39), with using neural network controller is proven.

The advanced vision of system (2.39) is know proposed. The more related theorem based on theorem (2.7.1), is discussed, the root of this theorem also been submitted, the following requirement and representation are shown as below, the generalizing of vector valued function $f \in R^m, g \in R^m$ into $f \in R^{m \times n} \; and \; g \in R^{m \times m}$ respectively, are of the main developed in this following theorem.

2.7.2 Theorem

Consider the nonlinear system

$$\dot{x} = \mathrm{A}\, x + \mathrm{B}\, f(x) x + \mathrm{B}\, g(x) u(t) \qquad (2.75)$$

where $x \in R^n$, and the smooth functions $f \in R^{m \times n}, g \in R^{m \times m}$, $u \in R^m$ is the control, $\mathrm{A} \in R^{nxn}, \mathrm{B} \in R^{nxm}$ are constant matrices. Assuming that

1. The pair (A, B) is controllable matrix.

2. The Function $g(x) \in R^{m \times m}$ is known such that $g(x) \neq 0, \forall x$, and $\|g(x)\| > \varepsilon$ where $\varepsilon \in R^+$ and $g_{ij}(x) \neq 0, \forall i = 1, \ldots, m; \forall j = 1, \ldots, m$.

3. Let the desired state be defined by $y_d(t) = \left[y_d \quad \dot{y}_d \quad \cdots \quad y_d^{(n-1)} \right]$, such that $\|y_d(t)\| \leq Q$, for some scalar Q.

4. The uncertain nonlinear function $f(x)$ may be estimated by $\hat{f}(x)$ such that $\left\| f(x) - \hat{f}(x) \right\| \equiv \left\| \tilde{f}(x) \right\| \le f_m(x)$ for some bounds function $f_m(x)$.

5. Let the filtered tracking error can be defined as

$$r = K^T e \tag{2.76}$$

Where $e(t) = x(t) - y_d(t)$ be the tracking error of the system in (2.75), and $K = \lfloor k_1, k_2, \ldots, k_{n-1}, 1 \rfloor$ is approximate chosen coefficient vector.

6. Set the nonlinear neuro-controller by

$$u(t) = u_1 + u_2 \tag{2.77}$$

Where

$$u_1(x(t)) = W(x(t)) - \hat{u}_2(x(t)) \tag{2.78}$$

Where $W(t)$ be chosen as the control law of the system (2.78). And u_2 be the nonlinear control function which we will approximate by the general neural network property.

7. let the following structure of artificial neural network have been adapter where

- Consider two-layer neural network, consisting of two layers of tunable weights. The hidden layer has L neurons, and the output layer has m neurons.
- The first layer neural network have the input which be chosen as $x_{NN} = \lfloor y_d, e \rfloor^T, v = \lfloor v_{ji} \rfloor, \quad j = 1,2,\ldots,n; \quad i = 1,2,\ldots,L$ as the weights of the first layer neural network which we will chose it randomly, and $b_0 = [b_{01}, b_{02}, \ldots, b_{0L}]^T$ as the bias of the first layer neural network which we will chose it randomly too.
- Choose $\sigma(.)$ as any continuous sigmiodal function which be the activation function of the network in the first layer and in the hidden layer (second layer).
- the output of the first layer will be defined as following

$$q_i = \sigma\left(v^T\ x_{NN} + b_0\right) \quad i = 1,2,\ldots,L \tag{2.79}$$

- the input to the hidden layer (second layer) of the neural network will be the output of the first layer (q_i).

- w is neural network adjusted weight (hidden layer neural network weights), and it is assumed that they are bounded so that $\|w\| \le w_m$, with w_m known bounds. And $\hat{w}$ are the estimated neural network weights which be provided by the neural network tuning algorithm as

$$\dot{\hat{w}} = s\,\sigma\left(v^T\ x_{NN}\right) r\ \mathrm{B}\, g - Ks\|r\|\hat{w} \tag{2.80}$$

where $s = s^T > 0$, any constant matrices representing the learning rates of the neural network.

K, small scalar positive design parameter.

Hence, the neural network weights approximation error is

$$\tilde{w} = w - \hat{w} \tag{2.81}$$

- Neural network universal approximation property defines that any continuous function can be approximated arbitrarily well using a linear combination of sigmoidal functions, such that the output of the neural network $u_2(x)$ can be defined as

$$u_2(x) = w^T\ \sigma\left(v^T x_{NN} + b_0\right) + \varepsilon(x) \tag{2.82}$$

Where the $\varepsilon(x)$ is the neural network approximation error.

Implementer neural network $\hat{u}_2$, is actually an approximation of the ideal neural network (2.49) and is given by

$$\hat{u}_2 = \hat{w}^T\ \sigma\left(v^T\ x_{NN} + b_0\right) \tag{2.83}$$

If the conditions down satisfied

$$\|r\| \ge \frac{\frac{K}{4} w_m^2 + \varepsilon_n}{k_{v\min}} \tag{2.84}$$

Or

$$\|\tilde{w}\|_F \geq \left[\frac{\sqrt{\frac{K}{4} w_m^2 + \varepsilon_n} + \frac{1}{2} w_m}{K} \right]^{\frac{1}{2}} \tag{2.85}$$

Then the filtered error $r(t)$ and the neural network weights $\hat{w}$ are (Uniformly Ultimate Boundness).

Proof:

Let

$$z = T\,x \tag{2.86}$$

and $z = \left[z_1,\ \ z_2,\ \ \cdots,\ \ z_n\right]^T$, where T is any invertible matrix, (The choice can be such that $T^{-1} = [L:\mathrm{B}]$, *where* L is selected such that the inverse exists).

$$\begin{aligned} \dot{z} = T\,\dot{x} &= T[\mathrm{A}\,x + \mathrm{B}\,f(x)x + \mathrm{B}\,g(x)u(t)] \\ &= T\left[\mathrm{A}\left(T^{-1}z\right) + \mathrm{B}\,f(z)\left(T^{-1}z\right) + \mathrm{B}\,g(z)u(t)\right] \end{aligned}$$

Let

$$h(z) = f(z)\left(T^{-1}z\right) \tag{2.87}$$

$$\Rightarrow \quad \dot{z} = T\,\mathrm{A}T^{-1}z + T\,\mathrm{B}\,h(z) + T\,\mathrm{B}\,g\,u(t) \tag{2.88}$$

Since (A,B) is controllable by theorem (2.4.1), we have

$$T\,\mathrm{A}\,T^{-1} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_n \end{bmatrix}, \qquad T\mathrm{B} = \begin{bmatrix} \mathrm{O}_m \\ \mathrm{O}_m \\ \vdots \\ \mathrm{O}_m \\ \mathrm{I}_m \end{bmatrix}$$

where $\alpha_1, \alpha_2, \ldots, \alpha_n$ are constant. And I_m standard for identity of order m. hence

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \vdots \\ \dot{z}_{n-1} \\ \dot{z}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-1} \\ z_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ h_m(z) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ g_m u(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \alpha_1 z_1 + \alpha_2 z_2 + \ldots + \alpha_n z_n \end{bmatrix}$$

where $g_m = \begin{bmatrix} g_{m1} & g_{m2} & \cdots & g_{mm} \end{bmatrix}$, $u = \begin{bmatrix} u_1 & u_2 & \cdots & u_m \end{bmatrix}^T$.

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \vdots \\ \dot{z}_{n-1} \\ \dot{z}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-1} \\ z_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ h_m(z) + \alpha_1 z_1 + \alpha_2 z_2 + \ldots + \alpha_n z_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ g_m u(t) \end{bmatrix}$$

Let

$$F(z) = h_m(z) + \alpha_1 z_1 + \alpha_2 z_2 + \ldots + \alpha_n z_n \tag{2.89}$$

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \vdots \\ \dot{z}_{n-1} \\ \dot{z}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-1} \\ z_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ F(z) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ g_m u(t) \end{bmatrix}$$

and hence, we have the following system

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= z_3 \\ &\vdots \\ \dot{z}_n &= F(z) + g_m u(t) \\ y &= z_1 \end{aligned} \tag{2.90}$$

with $z = \begin{bmatrix} z_1, z_2, \ldots, z_n \end{bmatrix}^T_{nx1}$, $F : R^n \to R$, an unknown smooth function; $g_m : R^n \to R$, a known smooth function; $u(t)$ the control.

Define the state tracking error $e_z(t)$ as

$$e_z = z - y_d \tag{2.91}$$

where $y_d(t)$ be the desired state vector which we can defined it by

$$y_d(t) = \begin{bmatrix} y_d & \dot{y}_d & \cdots & y_d^{(n-1)} \end{bmatrix}$$

$$\Rightarrow e_z = z - y_d \equiv T\,x - y_d$$

Differentiation yields

$$\dot{e}_z = T\,\dot{x} - \dot{y}_d \tag{2.92}$$

Using equation (2.75) we get

$$\begin{aligned} \dot{e}_z &= T(\mathrm{A}\,x + \mathrm{B}\,g\,u + \mathrm{B}\,f\,x) - \dot{y}_d \\ &\equiv T\left(\mathrm{A}\left(e_x - y_d\right) + \mathrm{B}\,g\,u + \mathrm{B}\,h(z)\right) - \dot{y}_d \\ &\equiv T\,\mathrm{A}\,e_x - T\mathrm{A}\,y_d + T\,\mathrm{B}\,g\,u + T\,\mathrm{B}\,h(z) - \dot{y}_d \end{aligned}$$

where $e_x = x - y_d$ as we defined it above, and since

$$T\,\mathrm{B} = \begin{bmatrix} \mathrm{O}_m \\ \mathrm{O}_m \\ \vdots \\ \mathrm{I}_m \end{bmatrix} \Rightarrow T\,\mathrm{B}\,g\,u = \begin{bmatrix} \mathrm{O}_m \\ \mathrm{O}_m \\ \vdots \\ \mathrm{I}_m \end{bmatrix} \begin{bmatrix} g_{11}u_1 + \ldots + g_{1m}u_m \\ g_{21}u_1 + \ldots + g_{2m}u_m \\ \vdots \\ g_{m1}u_1 + \ldots + g_{mm}u_m \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ g_m u(t) \end{bmatrix}$$

where

$$g_m = [g_{m1},\ \ g_{m2},\ \ \ldots,\ \ g_{mm}],\ \ u(t) = [u_1,\ \ u_2,\ \ \ldots\ \ u_m]^T.$$

and

$$T\,\mathrm{B}\,h(z) = \begin{bmatrix} \mathrm{O}_m \\ \mathrm{O}_m \\ \vdots \\ \mathrm{I}_m \end{bmatrix} \begin{bmatrix} h_1(z) \\ h_2(z) \\ \vdots \\ h_m(z) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ h_m(z) \end{bmatrix}$$

suppose that

$$u = \lfloor u_1 \quad u_2 \quad \dots \quad u_m \rfloor^T (u_1 + u_2)$$
$$= \left[\frac{1}{g_{mi}}, \quad \frac{1}{g_{m2}}, \quad \dots, \quad \frac{1}{g_{mm}} \right]^T (u_1 + u_2) \tag{2.93}$$

where by condition (2) above $g_{ij}(x) \neq 0, \forall\, x; \forall i = 1,\dots,m; \forall j = 1,\dots,m$, and

$u_i = \frac{1}{g_{mi}}$, $\forall i = 1,\dots,m$ *such that* $u_i \neq 0, \forall i = 1,2,\dots,m$.

The system (equation (2.91)) and the action (equation (2.93)) leads to the following

$$\dot{e}_z = T\,\mathrm{A}\,e_x - T\,\mathrm{A}\,y_d + \lfloor u_1 + u_2 \rfloor + T\,\mathrm{B}\,h(z) - \dot{y}_d \tag{2.94}$$

and, from equation (2.78) we choose that

$$W = -T\,\mathrm{B}\,h(z) + T\,\mathrm{A}\,y_d + \dot{y}_d + T\,K\,\mathrm{B}\,e_x - T\,K\,\mathrm{B}\,e_x \tag{2.95}$$

Such that equation (2.94) can be modified as

$$\dot{e}_z = T(\mathrm{A} + K\,\mathrm{B})e_x \tag{2.96}$$

Since (A,B) is a controllable pair by condition (1). To make the convergent faster define the filtered tracking error of the system (2.90) and by equation (2.76) we have the following

$$r_z = K^T e_z \tag{2.97}$$

where $K = \left[k_1, k_2, \dots, k_{n-1}, 1 \right]$ is approximate chosen (filter coefficients vector), so that $e \to 0$ *as* $r \to 0$. Then the time derivative of the filtered error can be written as

$$\dot{r} = g_m\, u(t) + F(z) + Y_d \tag{2.98}$$

where $Y_d = -y_d^{(n)} + \sum_{i=1}^{n-1} k_i\, e_{i+1}$, and $F(z)$ be defined in equation (2.89)

Similarly in terms of the filtered tracking error above system dynamics (by using equation (2.77)), can be described as follows

$$\dot{r} = g_m u_i \left[u_1 + u_2\right] + F(z) + Y_d \tag{2.99}$$

Using condition (4) above and equation (2.98), the tracking control law W which be chosen in equation (2.95) can be written as

$$W = \frac{1}{g_m u_i}\left[-\hat{F}(z) - Y_d - k_v r + \alpha\right] \tag{2.100}$$

where $\hat{F}$ be the fixed approximation of the functional F, k_v is the feedback gain, and assuming that α be the robust term chosen for the disturbance rejection which can be defined as

$$\alpha(t) = -f_m(z)\,sign(r) \tag{2.101}$$

where

$$sign(r) = \begin{cases} 1 & r > 0 \\ 0 & r \le 0 \end{cases} \tag{2.102}$$

$sign(\bullet)$ is the standard sign function. Using control law in equations (2.100), (2.78), (2.81), and substituting into equation (2.99) we get

$$\dot{r} = \tilde{F}(z) + T\,\mathrm{B}\,g\left\{\tilde{w}^T\,\sigma\left(v^T\,x_{NN} + b_0\right)\right\} + \alpha - k_v\,r + \varepsilon \tag{2.103}$$

Setting the Lyapunov function candidate as

$$V = \frac{1}{2}r^2 + \frac{1}{2}tr\left(\tilde{w}^T\,s^{-1}\,\dot{\tilde{w}}\right) \tag{2.104}$$

Where r be the filtered tracking error and $\tilde{w}$ be the neural network approximation error which we defined in equation (2.81), such that

$$\dot{\tilde{w}} = \dot{w} - \dot{\hat{w}} \tag{2.105}$$

Where $\dot{w}$ be the neural network ideal weights, and $\dot{\hat{w}}$ be the estimated neural network weights which provided by the neural network tuning algorithm as defined in equation (2.80).such that the differentiating yields

$$\dot{V} = \partial V\left(r(t)\right)/\partial t + \partial V\left(\tilde{w}(t)\right)/\partial t = \frac{\partial V}{\partial r}\cdot\frac{dr}{dt} + \frac{\partial V}{\partial \tilde{w}}\cdot\frac{d\tilde{w}}{dt}$$

$$\Rightarrow \quad \dot{V} = r\,\dot{r} + tr\left(\tilde{w}^T\,s^{-1}\,\dot{\tilde{w}}\right) \tag{2.106}$$

Hence substitution equation (2.103) in equation (2.106) yields

$$\begin{aligned}\dot{V} &= -k_v r^2 + r\tilde{F} + rT\,\mathrm{B}\,g\,\tilde{w}^T\,\sigma\left(v^T\,x_{NN}\right) + r\alpha + r\varepsilon + tr\left(\tilde{w}\,s^{-1}\,\dot{\tilde{w}}\right) \\ &= -k_v r^2 + r\left\{\tilde{F} + \alpha + \varepsilon\right\} + tr\left(\tilde{w}^T\left\{s^{-1}\dot{\tilde{w}} + \sigma\left(v^T\,x_{NN}\right) rT\,\mathrm{B}\,g\right\}\right)\end{aligned} \tag{2.107}$$

Applying the neural network tuning rules (equation (2.80)), the derivative of Lyapunov function is simplified to

$$\dot{V} = -k_v r^2 + r\left\{\tilde{F} + \alpha + \varepsilon\right\} + K\|r\|tr\left(\tilde{w}^T\,\hat{w}\right) \tag{2.108}$$

Using equation (2.101) one has the following

$$\dot{V} \le -k_{v\min}\|r\|^2 + K\|r\|tr\left(\tilde{w}^T\left(w - \tilde{w}\right)\right) - \|r\|\,f_m + \|r\|\tilde{F} + \|r\|\varepsilon_n \tag{2.109}$$

Using the inequality

$$tr\left[\tilde{x}^T\left(x - \tilde{x}\right)\right] \le \|\tilde{x}\|_F\,\|x\|_F - \|\tilde{x}\|_F^2 \tag{2.110}$$

Using the inequality (2.110) in the inequality (2.109) can be written as

$$\dot{V} \le -k_{v\min}\|r\|^2 + K\|r\|\left(\|\tilde{w}\|_F\,\|w\|_F - \|\tilde{w}\|_F^2\right) + \|r\|\varepsilon_n \tag{2.111}$$

(From using $\|w\|_F \le w_m$), and from (2.111)we have that

$$\begin{aligned}\dot{V} &\le \|r\|\left\{-k_{v\min}\|r\| + K\left(\|\tilde{w}\|_F\,w_m - \|\tilde{w}\|_F^2\right) + \varepsilon_n\right\} \\ &= \|r\|\left\{-k_{v\min}\|r\| - K\left(\|\tilde{w}\|_F - \frac{1}{2}w_m\right)^2 + \frac{1}{4}Kw_m^2 + \varepsilon_n\right\}\end{aligned} \tag{2.112}$$

which is guaranteed to remain negative as long as (conditions (2.84) (2.85) above satisfied as following

$$\|r\| \ge \frac{\frac{K}{4}w_m^2 + \varepsilon_n}{k_{v\min}}$$

Or

$$\|\tilde{w}\|_F \geq \left[\frac{\sqrt{\frac{K}{4} w_m^2 + \varepsilon_n} + \frac{1}{2} w_m}{K} \right]^{\frac{1}{2}}$$

Such that, since $V(0) = 0, V(r) \geq 0 \; and \; \dot{V}(r) < 0 \Rightarrow$ the stable of the system for tracking filter is guarantied and if $r \to 0, as\, t \to \infty \Rightarrow z \to 0, as\, t \to \infty$ and hence $x \to 0, as\, t \to \infty$ so that the stability of the system (2.75), with using neural network controller is proven.

2.7.1 Remark

The following are structure of the neuro-controller

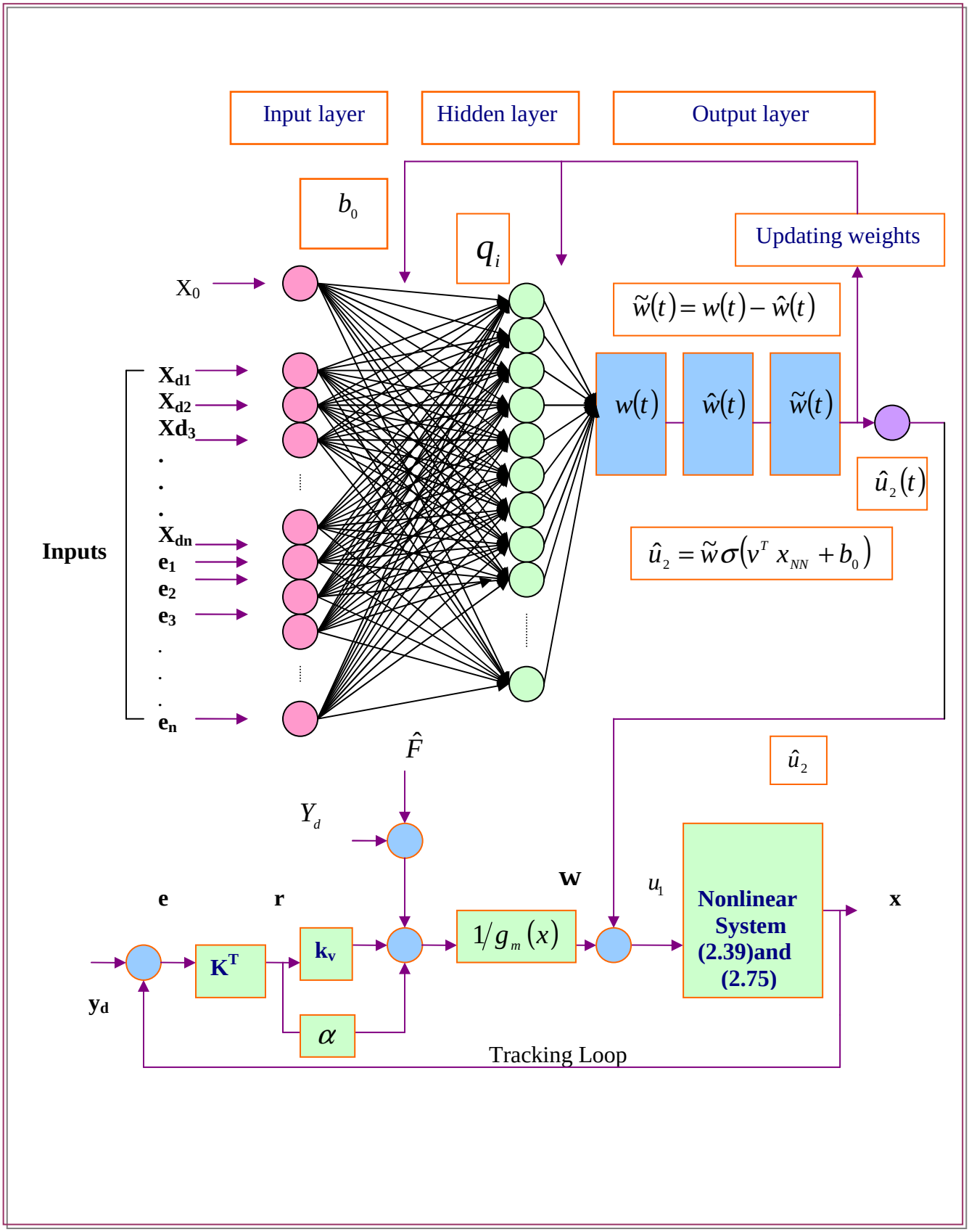


Figure (2.2)

General Nonlinear System and Neural Network

CONCLUDING REMARKS

1. The class loop optimal control using neural network has been defined and named as neurocontroller.

2. Since the weights are function of time as well as the solution and the control of the nonlinear dynamical control system, some difficulties have been faced for design neurocontroller. To over come this difficulties, in this work, some divided difference scheme has been adapted and then successive tuning weight and controller are implemented.

3. The neurocontroller depends on the nature of artificial neural network structure, number of inputs layers, number of hidden layers, number of output layers as well as the way for which the weights are tuned. If the number of hidden layers are increased , the numerical future becomes more accurate.

4. From the simulation, it is clear that the proposed scheme can effectively compensate the uncertain nonlinearity in a class of nonlinear control systems. Simulation results show that the proposed uncertain compensation techniques for the illustrations can be effective for a feedback-linearizable class of nonlinear systems so that the stability is reach faster.

5. neural network is trained by the filtered tracking error, trying to minimize the filtered tracking error.

CONTENTS

CHAPTER TWO: NONLINEAR NEURO CONTROLLER

DEDICATION

To…

My dear father and mother whose, I thank God every day to make me their daughter
With my love and respect

My brothers and sister whose are the secret of happiness in my life

My friend for their support

With my love

Aleyaa

FUTURE WORK

- Neurocontroller for uncertain nonlinear Robotic dynamical control systems may be developed.

- Neurocontroller plus neuro solution of some neural adjusting control system may be developed.

- Identification of uncertain dynamical control system based control system in artificial neural network base on our work must be developed.

- Fuzzy neurocontroller for uncertain of nonlinearity some may nonlinear also dynamical control system be considered.

Neural networks constitute a very large research field, and it is difficult to obtain a clear overview of the entire field. Several motives originally lead researchers to study neural networks.

When designing a controller for a particular system, it is obvious that a vital intermediate step is to acquire some knowledge about how the system will respond when it is manipulated in various ways. Not until such knowledge is available, can one plan how the system should be controlled to exhibit a certain behavior.

A common and practically oriented approach to control system design is to use physical insights about the system supplemented with a series of practical closed-loop tests. In the tests, different parameters are treated until a working controller is obtained. Another often-use approach is based on conducting a simple experiment with the system to provoke a particular response, [Khalil, 2002].

Control of nonlinear systems is a major application area for neural networks. The control design problem will be approached in two ways: direct design methods and indirect design methods. "Direct design" mean that a neural network directly implements the controller. Therefore, a network must be trained as the controller according to some kind of relevant criterion. The indirect methods represent a more conventional approach, where the design is based on a neural network model of the system to be controlled. In this case the controller is not itself a neural network, [Ogata, 1996], [John, 1990].

To control a system is to make it behave in a desired manner. How to express this "desired behavior" depends primarily on the task to be solved, but the dynamics of the system, the actuators, the measurement equipment, the available computational power, etc., influence the formulation of the desired behavior as well. Although the desired behavior obviously is very dependent on the application, the need to rephrase it in mathematical terms

suited for practical design of control systems seriously limits the means of expression.

The work in this thesis is divided into three chapters; the first chapter entitled "Introduction to Artificial Neural Network" gives the introductory material that is necessary to understand the Artificial Neural Network subject by giving historical background, some remarks and definitions on the neuron model and the architecture of a neural network which describe how a network transforms its input into an output. This transformation can be viewed as a computation. The model and the architecture each place limitations on what a particular neural network can compute. The way a network computes its output must be understood before training methods for the network can be explained. And we will discuss the learning rule of neural networks which fall into two broad categories: **supervised learning** which can be the training method are commonly used, and **unsupervised learning** and an example of learning networks (**Back propagation algorithm**) had been given.

In chapter two we have discussed some necessary requirements that are needed in the main theorems of the work of this thesis and proposed an artificial neural network-based scheme for control a class of nonlinear systems, which can be transformed to the canonical form. Neural network weights are tuned on-line, and the overall system performance is guaranteed using Lyaounov function approach. The convergence of the neural network learning process and the boundness of the neural network weights estimation error are all rigorously proven.

Chapter three can be considered as an extension to the work of chapter two. Where we give a general computational algorithm for our work and we discussed two simulation examples: "Pendulum type" nonlinear system and a proposed nonlinear 3-dimention concerning system, the numerical result are shown in tables and figures.

Numerical simulation are obtained using MATLAB version 6.5 and using the personal computers PIT 4.

INTRODUCTION

Artificial neural networks form a class of systems that is inspired by biological neural networks. They usually consist of a number of simple processing elements, called neurons, that are interconnected to each other. In most cases one or more layers of neurons are considered that are connected in a feed forward or recurrent way. The strength of the interconnections is quantified by means of interconnection weights. Basic features of neural architectures are that they work massively parallel, the weights have to be learned from a set of examples and can be adapted. Although artificial neural networks can perform human brain-like tasks such as object and pattern recognition, speech recognition or associative memory, there is still a huge gap between biological and artificial neural nets. Nevertheless, although we are still far away from mimicking the human brain, from an engineering point of view, it is certainly step to let us inspire by biology. Indeed artificial neural networks have provided good solutions to many problems in various fields: example include classification problems, vision, speech, signal processing, time series prediction, modeling and control, robotics, optimization, expert systems and financial applications,[Zurada, 1996].

Many of the abilities one possesses as a human have been learned from examples. Thus, it is only natural to try to carry this "didactic principle" over to a computer program to make it learn how to output the desired answer for a given input. In a sense the artificial neural network is one such computer program; it is a mathematical formula with several adjustable parameters, which are tuned from a set of examples. These examples represent what the network should output when it is shown a particular input.

REFERENCES

[Adam, 1990] Adam Hilger, R. Beal and T. Jackson, " Neural Computing ", 1990.

[Alexander, 1989] Alexander, I., " Neural Computing Architectures-The Design of Brain Link Machines ", ed. Cambridge, Mass.: MTT Press.1989.

[Amari, 1967] S. Amari, " Theory of adaptive Pattern Classifiers ", IEEE Trans. Electron. Computers, vol. EC-16, pp. 299-307, June 1967.

[Amari, 1971] S. Amari, " Characteristics of randomly Connected threshold element networks and network systems ", Proc. IEEE., vol. 59, pp. 35-47, Jan., 1971.

[Amari, 1972] S. Amari, " Characteristics of random nets of analog neuron-like elements ", IEEE Trans. Syst., Man, Cybernetics, vol. SMC-2, PP. 643-657, Nov. 1972.

[Amari, 1972] S. Amari, " Learning Patterns and Pattern Sequences by Self-organizing nets of threshold elements ", IEEE Trans. Comput., vol. C-21, pp. 1197-1206, Nov. 1972.

[Amari, 1974] S. Amari, " A method of Statistical neurodynamics ", Kybernetik, vol. 14, pp. 201-215, Apr. 1974.

[Amari, 1977] Amari, S. I., " Neural Theory of Association and Concept Formation ", Cybern, vol. 26, pp. 175-185, 1977.

[Amari, 1977] S. Amari, " Neural theory of association and concept-formation ", Biol. Cybernetics, vol. 26, pp. 175-185, 1977.

[Amari, 1977] S. Amari, K. Yoshida and K. Kanatani, " A mathematical Foundation for Statistical neurodynamics ", SIAMJ. APPI. Math., vol. 33, pp. 95-126, 1977.

[Amari, 1990] Amari, S. I., " Mathematical Foundations of Neurocomuting ", IEEE. Prod: vol. 78, No. 9, pp. 1443-1463, 1990.

[Allen, 1994] Allen, D. W. and Taylor, J.G., ED. M. Marinaro and P. Morasso**,** " Learning time series by Neural Networks ", Proc. INT. Conf. on Artificial Neural Networks (Sorrento, Italy, Berlin: Springer), PP 529-32, 1994.

[Arbib, 1987] Arbib, M.A., " Brains. Machines and Mathematics ", 2nd ed. New York, Springer Verlag., 1987.

[Anderson, 1972] J.A. Anderson, " A simple neural network generating interactive memory ", Math. Biosciences. vol. 14, pp. 197-220, 1972.

[Anderson, 1977] Anderson, J. A., J. W. Silverstein, S. A. Rite, and R. S. Jones., " Distinctive Features, Categorical Perception. and Probability Learning, Some Application of a Neural Model. ", Psrch Rev. vol.84, pp. 413-451, 1977.

[Barron, 1993] Barron, A. R., " Universal approximation bounds for Super Positions of a Sigmoid function ", IEEE. Transactions on Information Theory, vol. 3a, pp. 930-945, 1993.

[Barron, 1993] Barron A. R., " When do neural nets avoid the curse of dimensionality ? ", NATO advanced study institute- From Statistics to neural networks, Les Arcs, France, June 21-July 2, 1993.

[Baum, 1989] E. B. Baum and D. Haussler, " What size net gives valid generalization ? ", Neural Computation, vol. 1, pp. 151-160, 1989.

[Cover, 1965] T. Cover , " Geometrical and Statistical Properties of Systems of linear inequalities with applications to pattern recognition ", IEEE Trans. Electron. Computers, vol., 14, pp. 326-334, 1965.

[Cybenko, 1989] Cybenko, G., " Approximation by Superpositions of a sigmoidal function ", Mathematics of Control, Signals and systems, vol. 2, No. 4, pp. 303-314. 1989.

[Cohen, 1983] M. Cohen and S. Grossberg, " Absolute stability of global pattern formation and parallel memory stage ", by competitive neural networks IEEE Trans. Syst., Man, Cybernetics, vol. SMC-13, pp.

[Chen, 1984] Chen, Chi-Tsony, " Linear System theory and design ", HRW Series in Electrical and Computer Engineering., 1984.

[Chen, 1991] Chen, F. and Khalil, H. K., " Adaptive Control of nonlinear system using neural networks-a dead-zone approach ", In proceedings of the American Control Conference, Boston, MA, pp. 667-672, 1991.

[Dayhoff,1990] Dayhoff. J., " Neural Network Architectures-An Introduction ", New York. Van Nostrand Reinhold, 1990.

[Dreyfus, 1962] Dreyfus, S., " The Numerical Solution of Variational Problems ", Math. Appl., vol. 5, No.1, pp. 30-45, 1962.

[Dreyfus,1990] Dreyfus. S. E., J. Guridance., " Artificial Neural Networks Back propagation and the Kelley-Bryson Gradient procedure, Control Dynamic. 1990.

[Fukushima,1980] Fukushima, K. and S. Miyaka., "Neocognition: A self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position.", Biol. Cybern, Vol. 36, No.4, pp. 193-202, 1980.

[Funahashi, 1989] Funahashi K. I., " On the Approximate Realization of Continuous Mapping by Neural Networks ", vol. 2, pp. 183-192. 1989.

[Funahashi, 1993] Funahashi, K. and Nakamura, Y., " Approximation of Dynamical Systems by Continuous time recurrent neural networks ", Neural Networks, vol. 6, pp. 801-806, 1993.

[Girosi, 1989] Girosi F., Paggio T., " Representation Properties of networks, Kolmogorov's theprem is irrelevant ", Neural Computation, vol. l, pp. 1465-469, 1989.

[Grossberg,1982] E. Gardner, J. Phys, Grossberg. S., " Studies of Mind and Brain, Learning Perception. Development. Cognition, and Motor, 1982

[Grossberg,1977] Grossberg, S., "Classical and Instrumental Learning by Neural Networks in Progress in Theoretical Biology ", New York: Academic Press, vol. 3, pp. 51-141, 1977.

[Gardner, 1988] Gardener, A., " The Space of interactions in neural network models", vol. 21, pp. 257-270, 1988.

[Helb,1984] Helb, D. O., " The Organization of Behavior, A Neuropsychological

Theorem ", New York, John Wiley, 1949.

[Hopfield, 1982] Hopfield, J., " Neural Networks and Physical System with emergent collective computational properties ", Proc. Natl. Acad. Sci., USA, vol. 81, pp. 3088-3092, vol. 79, pp. 2554-2558, 1982.

[Hopfield, 1984] Hopfield, J. J., " Neurons With Graded Response Have Collective Computational Properties Like Those of two state Neurons ", Proe. Nod Accd. Sci., vol. 81, pp. 3088-3092, 1984.

[Hebb, 1949] Hebb, D. O., " The Organization of Behavior a Newopsychological Theory ", New York, John Wiley, 1949.

[Hertz, 1991] Hertz, J., A. Krogh. And R. G. Palmer., " Introduction to Theory of Neural Computation", Redwood city, Calif., Addison-Wes leg Publishing Co., 1991.

[Hecht, 1987] Hecht-Nielsen, R., " Kolmogorov,s mapping neural network existence theorem ", Proc. Int. Cof. On Neural Networks III, New York, IEEE, PP 11-13, 1987.

[Hornik, 1989] Hornik, K., Stinch combe, M. and White, H., " Multi-larger feed forward networks are Universal approximators Neural Networks ", vol. 2, pp. 359-366, 1989.

[Hornik, 1991] Hornik K., " Approximation Capabilities of multilayer feed forward networks ", Neural Networks, vol. 4, pp. 251-257, 1991.

[Hagan, 1996] Hagan, M.T., and M. Menhaj, " Traing feed forward network with the Marquued algorithm ", IEEE Transaction son Neural Network, vol. 5, no. 6, pp. 989-993, 1994.

[Haward, 1996] Haward Demuth and Mark Beal, " Neural Network Toolbox ", Published by PWS publishing Company. 1996.

[Huibert, 1972] Huibert K. W. and Raphael S., " Linear Optimal Control Systems ", John Wiley and Sons, Inc., 1972.

[Ian, 1986] Ian, Peterson, and Khristopher Hout, " Equation approach to the Stabilization of uncertain linear Systems ", a Riccati, Automatica, vol. 22, No. 4, 1986.

[Isidori, 1995] Isidori, A.," Nonlinear Control System ", Springer-Verlag, London, UK, 3rd edition. 1995.

[Judistisky, 1995] Judisky, A., Hjalmarson, H., Benveniste, A., Delyon, B., Ljung, L., Sjoberg, J., and Zang, Q., " Nonlinear black-box models in system Identification", : Mathematical foundations. Automatica, vol. 31no. 12, pp. 1725-1750, 1995.

[John, 1990] John Van Deregate, " Feedback Control Systems ", Prentice Hall International, Inc., 1990.

[Jayc, 1968] Jacy H. and U.M., " Modern Control Principle and applications ", New York, Mc Graw-Hill, 1968.

[Johan, 1996] Johan A.K. Suykens, Joos P.L.Vandewalle and Bart L.R. De Moor., " Artificial Neural Networks for Modelling and Control of Non-Linear Systems ", Published by Kluwer Academic Publishers, 1996.

[Kohonen, 1972] T. Kohonen, " Correlation matrix memories ", IEE Trans. Comput., vol. C-21,pp. 353-359, 1972.

[Kohonen,1977] Kohonen, T., "Associative Memory: A system-Theoretical Approach, " Berlin: Springer-Verlag. 1977.

[Kohonen, 1982] Kohonen, T., ed. S. Amari, M. Arbib., " A simple Paradigm for the self-organized Formation of Structured Feature Maps, in Compeition and Cooperation in Neural Nets ", Berlin: Springer-verlag, vol. 45 1982.

[Kohonen,1984] Kohonen, T., "Self- Organization and Associative Memory", Berlin: Springer Verlag. 1984.

[Kohonen, 1988] Kohonen, T., " The Neural Phonetic Typewriter ", IEEE Computer, Vol. 27, no.3, pp. 11-22, 1988.

[Kurkova, 1991] Kurkova V., " Kolmogorov's theorem is relevant ", Neural Computation, vol. 3, pp. 617-622, 1991.

[Kurkova, 1992] Kukova V., " Kolmogorov's theorem and multilayer neural Networks ", Neural Networks, vol. 5, pp. 501-506, 1992.

[Kac, 1959] M. Kac, " Probability and Related Topics in Physical Sciences ",. New York: Wiley Interscience, 1959.

[Khalil, 1996] Khalil, H. K., " Nonlinear Systems ", Prentic Hal, 1996.

[Khalil, 2002] H. k. Khalil, " Nonlinear System ", Prentice Hall, Upper Saddle River, NS, 3rd Ed., 2002.

[Kolman, 1984] Kolman B., " Introductory linear algebra with applications ", 1984.

[Leshno, 1993] Leshno M., Lin V. Y., Pinkus A., Schockens., " Multilayer feed forward networks with a nonpolyomial activation function can approximate any function ", Neural Networks, vol. 6, pp. 861-867, 1993.

[Leag, 1983] Leag S. Shieh, Yiht, Tsay and Robert Yates, " State feedback decomposition of multivariable system via block-pole placemant ", IEEE, vol. AC-28, No. 8,. 1983.

[Lewis, 1993] F. L. Lewis, C. T. Abdallah, and Dawson, Macmillan, " Control of Robot Manipulators ", New York, 1993.

[Lewis, 1999] F. L. Lewis, S. Jagannathan, and A. Yesildirek, " Neural Network Control of Robot Manipulators and Systems ", Taylor and Francis Philadelphia, PA, 1999.

[McCulloch, 1943] McCulloch, W. S. and W. H. Pitts, "A logical calculus of the Ideas Imminent in Nervous Activity, " Bull. Math. Biophy., vol. 5, pp. 115-133, 1943.

[McClelland, 1986] McClelland, T.L., D. E. Rumelhart., "Parallel Distributed Group. Processing ", Combridge: MIT Press and the PDP Research, 1986.

[Minsky, 1954] Minsky, M., " Neural Nets and the Brain ", Doctoral Dissertation, Princeton University. NJ., 1954.

[Minsky, 1969] Minsky, M. and S. Papert, "Perceptrons. Combridge, Mass.",MIT Press. 1969.

[Nakano, 1972] K. Nakano, " Association-A model of associative memory ", IEEE Trans. Syst., Man, Cybernetics, vol. SMC-2, pp. 318-388, 1972.

[Nilsson, 1965] Nilsson, N. J, "Learning Machines: Foundations of Trainable Pattern Classifiers ", New York: Mc Graw Hill; also republished as the Mathematical Foundations of Learning Machines, Morgan-Kaufmann Publishers, San Mateo. Calif., 1965.

[Norgaard, 2000] M. Norgaard, O. Raven, N.K. Poulsen and L.K. Hansen, " Neural Networks for Modeling and Control of Dynamic Systems ", Springer-Verlag London, 2000.

[Ogata, 1967] Ogata, K., " State Space analysis of control systems ", Prentice Hall, Englewood, Cliffs, USA, 1967.

[Ogata, 1996] Ogata Kat Suhiko, " Modern Control Engineering ", Prentice-Hall of India, Second Edition, 1996.

[Pineda, 1987] F. J. Pineda, " Generalization of back-propagation to recurrent neural networks ", Phys. Rev. Lett., vol. 59, pp. 2229-2232, 1987.

[Poggio, 1990] Poggio, T., and F. Girosi., " Networks for Approximation and Learning ", Prod. IEEE, vol. 78, no. 9, pp.1481-1497, 1990.

[Paual, 1997] Paual, J. Werbos., " Handbook of Neural Computation"., IOP Publishing Ltd and Oxford University Press., 1997.

[Parker, 1982] D. B. Parker, " Learning Logic ", TECH. Rep. TGR-47, M.I.T. Sloan School of Management, Cambridge, MA, 1982.

[Parker, 1985] Parker, D. B., " Learning logic Technical Report ", TR-47 CENTER FOR Computational Research in Econnomics and Management Science, Massachusetts Institute of Technology, Cambridge, MA. 1985.

[Rozoner, 1969] L. I. Rozonoer, " Random logical nets: I ", Automat, Telemek., vol. 5, pp. 137-147, 1969.

[Rosenblatt, 1958] Rosenblatt, F., " The Perceptron, A Probabilistic Model for Information Storage and Organization in the Brain," Psych. Rev., Vol. 65, pp. 386-408, 1958.

[Rosenblatt, 1961] Rosenblatt, F., " Principle of Neurodynamics ", Washington, D.C.: Spartan press, 1961.

[Rumelhart, 1986] D. E. Rumelhart, G.E. Hinton, and R.J. Williams, " Learning internal representations by error propagation ", in parallel Distributed processing, D. E. Rumelhart, and J.L. McCelland, eds. Cambridge : M.I.T. Press, vol. I, pp. 318-362, 1986.

[Salle, 1961] Salle, J. L. and Lefschetc, S., " Stability by Lypunov's Direct Press. Method ", Mathematics in Science and Engineering. Academic 1961.

[Slotine, 1991] Slotine, J.- J. E. and Li, W., " Applied Nonlinear Control ", Prentice-Hall Englewood Cliffs, N. J. 1991.

[Tanakard, 1980] F. Tanakard, S.F. Edwards, J. phys. F., " Analytic theory of the ground state properties of a Spain glass ", vol. 10, pp. 2769-2778, 1980.

[Von, 1958] Von Neumann, J., " The computer and the Brain ", New Haven, Conn.: Yale University Press, vol. 87, 1958.

[Widrow, 1960] Widrow, B. and M.E. Hoff, Jr., " Adaptive Switching Circuits, " IRE Western Electric sow and Convention Record, pare 4 (Aug. 23), pp. 96-104, 1960.

[Widrow, 1962] Widrow. B., ed. M.C. Jovitz. G.T. Jacobi, G. Goldstein. Washington, D.C, " Generalization and Information Storage in Networks of Adaline 'Neurons', inself-organizing system ", Spartan Books, pp. 433-461, 1962.

[Werbos, 1974] P.Werbos, " New Tool for Prediction and Analysis in the Harvand Behavioral Sciences ", Beyond Regression, ph. D. thesis, Univ., 1974.

[Werbos, 1974] Werbos. P.J., " Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences ", Doctoral Dissertation Appli. Math., Harvard University. Mass. 1974.

[Werbos, 1993] Werbos, P., " Elastic fuzzy logic ", a better fill to neurocontrol and trueintelligence J. Int. Fuzzy Syst. Vol. 1, pp. 365-377, 1993.

[Werbos, 1994] Werbos, P., " The Roots of Backpropagation : From ordered (New Derivatives to Neural Networks and political Forecasting ", York: Wiely, 1994.

[White, 1992] White, D. A. and Sofge, D.A., " Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches ", (eds), (New York: Van Nostrand). 1992.

[Werisbach, 1972] G. Werisbach, J. Phys. Lett., " Scaling laws for the attractors of Hopfield networks ", vol. 46, pp. 623-630, 172.

[Wilson, 1972] H. R. Wilson and J. D. Cowan, Biophys. J., " Excitatory and vol. inhibitory interactions in localized populations of model neurons ", vol. 12, pp. 1-24, 1972.

[Zurada, 1996] J. M. Zurada, " Artificial Neural System ", West Publishing Co., 1969.

SUPERVISORS CERTIFICATION

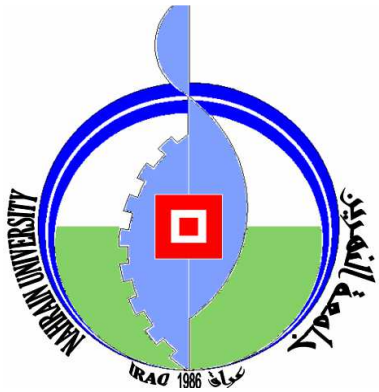
I certify that this thesis was prepared under my supervision at the department mathematics and computer application in the College of Science, Al-Nahrain University as a partial fulfillment of the requirements for the degree of Master of Science in Applied Mathematics.

Signature:
Name: Dr. Radhi Ali Zboon
Lecturer
Date: /9/2006

In view of the available recommendations; I forward this thesis for debate by the examining committee.

Signature:
Name: Asist. Prof. Dr. Akram M. Al-Abood
Head of the department mathematics and computer applications
Date: /9/2006

Ministry of Higher Education
and Scientific Research
Al-Nahrain University
College of Science



Nero-Controller of a Class of Uncertain Nonlinear Dynamical Control System

A Thesis

Submitted to the Department of Mathematics, College of Science, Al-Nahrain University, as a Partial Fulfillment of the Requirements for the Degree of Master of Science in Applied Mathematics

By

Aleyaa Hussein Naser Al-Janabi

(B.Sc., 2003)

Under Supervision of

Dr. Radhi Ali Zboon

September 2006 *Shaaban 1427*

وزارة التعليم العالي والبحث العلمي
جامعة النهرين
كلية العلوم

مسيطر عصبي غير خطي لحزمه من الأنظمة الديناميكية الا خطيه غير المحددة

رسالة

مقدمه إلى كلية العلوم في جامعة النهرين وهي جزء من متطلبات نيل
درجة ماجستير علوم في الرياضيات

من قبل الباحثه

علياء حسين ناصر الجنابي

(بكالوريوس علوم، ٢٠٠٣)

بإشراف

د. راضي علي زبون

أيلول ٢٠٠٦

رجب ١٤٢٧

المستخلص

لقد أصبحت أنظمة السيطرة في يومنا هذا جزء مهم ومكمل لحياتنا حيث أنها تدخل في معظم الأشياء ابتدأ من الأجهزة المنزلية الكهربائية الصناعية إلى صناعة الطائرات والمركبات الفضائية .
تشترك أنظمة السيطرة الاوتوماتيكية بصفات موحدة رغم الاختلافات في الإشكال لكي تجعل من النظام قيد الدراسة يتصرف بسلوك مقبول .
يعتبر نظام السيطرة الغير الخطي من أهم التطبيقات الشبكية العصبية الأصطناعية (Artificial Neural Network)
في هذه الرسالة لقد تم تطوير مسيطر عصبي غير خطي (Neuro-Controller) يعتمد على نظام الشبكية العصبي يهدف إلى التعويض والتصحيح والموازنة في بعض أنظمة السيطرة الديناميكية غير الخطية.
يضمن مشروع العمل إلى الاستقرارية ،تقليص الخطأ ،فلترة النتائج،
لقد تم أشتقاق و تنظيم وتدريب أوزان الشبكة العصبية (Neural Network Weights) المستعملة بالأعتماد على أسلوبية دالة ليابانوف.
(Lyapunov Function approach)
لتحقق من فاعلية عملنا نفذنا على الحاسبة الالكترونية بعض أنظمة السيطرة الدينامية غير الخطية وكانت النتائج جيدة بما فية الكفاية لتعويض والسيطرة
على الجزء غير الخطي غير المحدد في نظام (Uncertain nonlinear function) السيطرة غير المحدد .
وأخيرا لقد تم عرض بعض المفاهيم الرياضية الضرورية والاستنتاجات المهمة والعمل المستقبلي مدعوم بأشكال ورسومات لإكمال طرح وافي للعمل.