T-4495

Neural Network Dynamics: A Classical Approach

by

John Ryan

A thesis submitted to the Faculty and Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Applied Mechanics).

Golden, Colorado

Date 4-5-95

Signed: John N. Ryan

Approved: Dr. Christopher G. Braun
Thesis Advisor

Golden, Colorado

Date 4-5-95

Dr. Joan P. Gosink
Professor and Head,
Department of Engineering

ABSTRACT

In this thesis, the author models the learning behavior of a single neuron and synapse trained with back propagation using the methods of classical dynamics. By determination of dynamic parameters of the neural system such as the generalized coordinate, velocity, mass, momentum, dissipation and time constant, the author demonstrates the connection between the dynamics of the neural system and physical systems such as those involving mechanical translation, rotation and LRC circuitry. Once the neural dynamic parameters and their meaning in terms of training are established, the author progressively augments the general equation describing first order weight trajectories to predict synaptic change during training. Specifically, the analysis proceeds from a linear time invariant (LTI) model, a single neuron and synapse trained with constant input and reference values, to a linear time varying (LTV) model of the same system excited by varying inputs and reference values. Time, in the context of training, is the number of training presentations, since the weights are updated at each computational time step. The analysis culminates with a non-linear, time varying (NLTV) model of the neural system with a sigmoid activation function. The accuracy of each of the models, measured by the difference between the actual and predicted weight values, is within the roundoff error of the computer.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

t - time

E_p - error with respect to a training pattern

w - weight value

w_{ij} - weight connecting the ith and jth layers

δ - error signal

δ_j - error signal at a node on the jth layer

σ - sigmoid activation function

σ' - derivative of sigmoid with respect to neuron input

w_{final} - final value of weight

R_k - Reference value at the kth node

R - Reference value

η - learning rate

o_j - output of the jth neuron

λ_n - scaling factor for the nth input

λ_{1-q} - product of scaling factors from patterns 1 to q

Ψ - sum of products of scaling factors and first weight values

m - mass

b - friction coefficient

D - Rayleigh's dissipation function

μ - dissipation coefficient

T - Kinetic Energy

V - potential energy

$L \equiv T - V$ - Lagrangian

H - Hamiltonian

τ - time constant

q - generalized coordinate

$\dot{q}$ - generalized velocity

p - generalized momentum

$\dot{p} = \frac{dp}{dt}$ - time derivative of generalized momentum

v - velocity

C - capacitance

λ - flux linkage

L - inductance

θ - angular position

ω - angular velocity

ACKNOWLEDGMENTS

There are many people who made have made my attendance at the Colorado School of Mines the most fulfilling period in my life. I would like to thank my advisor Dr. Chris Braun for going above and beyond the call of duty to provide assistance and support for my efforts. I am also indebted to the other members of my committee, Dr. Jerry Jones, Ravel Ammerman and Dr. Donald Mackison of the University of Colorado, Boulder. Without Dr. Mackison's friendship, I would have never undertaken this work or attended graduate school. I thank him for his enthusiastic pursuit of excellence in all endeavors, academic and humanitarian. I would also thank Dr. Mark Balas for his insight into the multi-faceted world of applied mathematics and the truly remarkable manifestations of advanced control systems. Dr. Lori Pratt has also served as an inspiration for excellence.

Paulette and George Whitcomb have provided help, guidance, and a great deal of support in all my endeavors. To them and many others, I am deeply grateful, thank you!

CHAPTER 1

INTRODUCTION

During the learning process, a neural network is a dynamic system; the weights evolve through time according to a learning algorithm. The present work investigates dynamic synaptic behavior resulting from the back-propagation algorithm with the weights updated after every training presentation. This a gradient descent algorithm which attempts to minimize the sum squared error over the training patterns. Through such a gradient descent algorithm, the network can "learn" complex input-output relationships due to the inherently non-linear nature of the activation functions. However, due to the mathematical complexity of nested, non-linear activation functions in the input output path of an interconnected neural network, calculation of the long term behavior of neural network activations and weight values over a large number of training patterns is prohibitive in terms of CPU time. The back-propagation algorithm is temporally local. The long-term behavior of the network is uncertain since the learning algorithm finds the steepest gradient at each time step only. Unforeseen multi-dimensional "plateaus" or "lakes" may lie in the path of steepest descent thereby halting the progression of the network toward the global minimum.

This thesis uses the energy of a single neuron and synapse to generate the dynamic parameters which are used to which predict long term and transient synaptic behavior. While others have viewed network function in terms of energy functions (Kanter 1992; Hertz, Krogh and Palmer 1991; Xu 1990; Cohen and Grossberg 1983), these have been in terms of stochastic energies, either in terms of statistical mechanics or thermodynamic models. While network interactions are very complex, stochastic models are not necessary. The present work involves the application of deterministic kinetic and dissipation functions. Once these energy functions are found, they are used in Lagrangian and Hamiltonian methods from classical mechanics. These dynamic formulations are primarily used as system identification tools, since the equations of motion through time require continuity of the energy functions (Wylie and Barrett 1982; Petrov 1968). The discrete version of Hamiltonain dynamics, via Schrödinger's equation (Kilmister 1963; Ter Haar 1961), or investigations into discrete dynamic energy functions (LaSalle and Artstien 1976) are beyond the scope of the present investigation.

A few have investigated the Hamiltonian approach to Neural Network dynamics (Ramacher 1993; De Wilde 1993; Kanter 1992; Ramacher and Nachbar 1991; Sandler 1991). However, these investigations have either applied an incorrect energy function, or have applied the Hamiltonian concept to a very restricted class of special neurons.

1.1 Objective

The objective of this work is the formulation of a mathematical model which accurately predicts the behavior of a single synapse with a sigmoid activation function, time-varying inputs and time-varying reference value.

1.2 Dynamics

In more traditional electrical or mechanical systems, differential equations of motion describe the long term and transient behavior of systems. While these equations are often non-linear and anholonomic, they can be formulated using energy methods such as Lagrangian and Hamiltonian dynamics. This presumes, however, that dynamic parameters such as the generalized coordinates, mass, momenta and dissipation are known with some degree of accuracy. While these concepts are intuitive in a mechanical context, they are unfamiliar in an electrical context. For example, the generalized coordinate of an electrical system may be flux linkage or charge. The generalized mass may be capacitance or inductance. Similar analogies exist for thermal systems as well. A neural network, however, is not a physical system. It resides entirely within the confines of a computer algorithm or is realized in electrical circuitry. However, generalized dynamic parameters apply to the neural system as well.

1.3 Approach

While neural networks are just that, networks of neurons, this work focuses on the training dynamics of a simple neural system, an input and output node connected by a single synapse or weight, since the form of the dynamic parameters are the same for each neuron and synapse in an arbitrarily connected network. The goal of this work is the accurate mathematical description of the transient and steady state behavior of a non-linear neural system: an input node connected to an output node with a sigmoid activation function excited by time varying reference and input values. The training presentation number represents time in this context. A brief introduction to neural networks provides insight into the function and methodology necessary for understanding of training dynamics. In the interest of forging the connection between neural dynamics and the dynamics of physical systems, a brief introduction to Lagrangian and Hamiltonian mechanics follows. Determination of the energy and dynamic parameters, such as generalized mass, momentum and dissipation, for a linear single input, single output (SISO) linear time invariant (LTI) neuron is the first step in the construction of a non-linear model. The results are then extended to a linear time varying (LTV) neuron and finally extended to the behavior of the non-linear time varying (NLTV) neural system with a sigmoid activation function, since the sigmoid commonly used in practice. Predicted weight or synaptic trajectories (values over time) are verified at each level of analysis.

CHAPTER 2

BACKGROUND

Neural networks present a novel approach to classification and other types of "learning" problems with unknown input-output (IO) relations. Two examples might be a network which learns (without software development other than the neural network and communications protocols) to read selected passages of text with input provided by a camera and output channeled to a voice synthesizer (Sejnowski and Rosenberg 1987) or a network that supplied inputs to a motor to control an inverted pendulum on a cart (Baker and Farrell 1990). Of course, with the proliferation of research in this area, there are multitudes of other such recognition, memory and control applications.

2.1 Neural Network Topology

For example, a graphical representation of a simple network which learns the XOR relation is shown in Figure 1. Table 1 provides the actual IO relations. Inputs are presented to the network at the two lower input nodes and are multiplied by weights, also called synapses, connecting to summing junctions called neurons or nodes. The output of an upper node is a function of the sum of the weighted inputs presented to the input layer. The XOR net of Figure 1 has three layers of neurons: input, hidden, and output, proceeding from bottom to top. These are designated as i, j, ,k, respectively. The weight which connect input to hidden layers have the subscript ij while those connecting the hidden to output layers have a jk subscript. In the figure, there are four weights connecting the input to hidden layer.

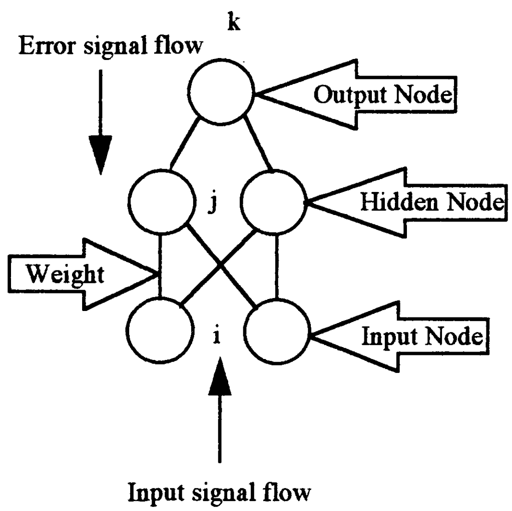


Figure 1 XOR Neural Network Schematic

Input 1	Input 2	Reference
1	0	1
0	0	0
0	1	1
1	1	0

Table 1 XOR Training Data

When a data point or training pattern is presented to the network, the input value is multiplied by the weights in the input layer which are then the inputs to the hidden nodes to which they connect. The output of these nodes is a non-linear function of the input. In a three layer net, the hidden layer connects to the output layer, whose input comes from the outputs of the hidden layer. A common nodal IO function is the sigmoid, shown in Figure 2, named for its S shape. Specifically, the function is

$$\sigma(in) = \frac{1}{1+e^{-in}} \tag{1}$$

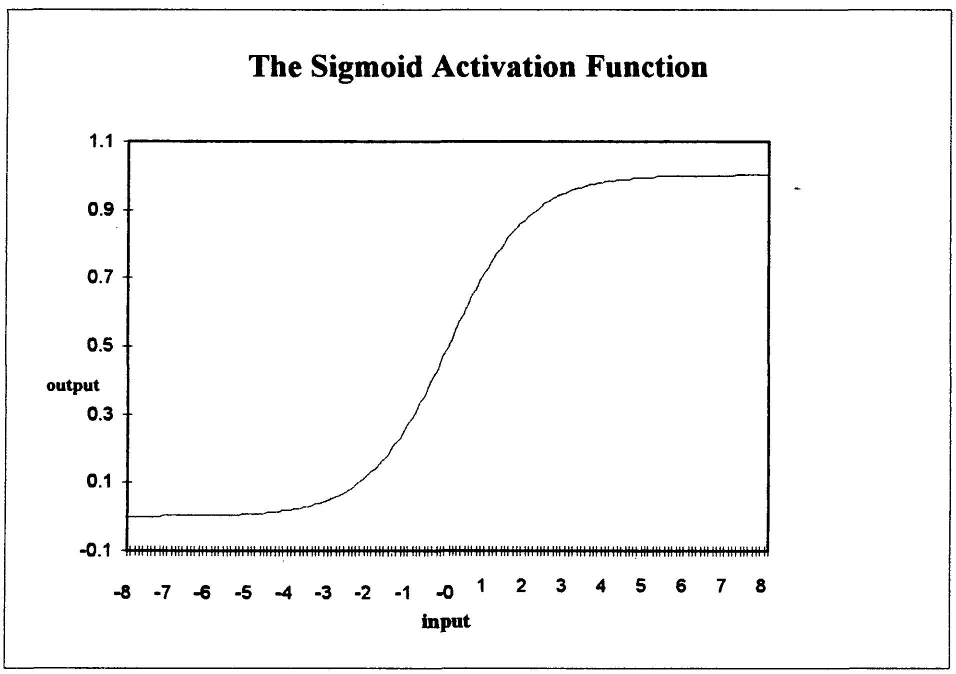


Figure 2 The Sigmoid Activation Function

The sigmoid is a common activation, (IO), function for nodes in a neural network. Where *in* is the input to the node. This function behaves like a switch; a large negative input will drive the output to zero while a large positive input will drive the output to one. If a large negative weight multiplies a large positive input from below, the hidden node will turn off . The converse is true for the on state. Complex and often competitive on and off patterns emerge in the network. Some weight input combinations will try to turn the connecting units on while other weights may turn them off. The activation states may change radically as different input patterns are presented. On one presentation, a weight may be excitatory for a connecting unit's activation. On the next presentation (training pattern), the same weight may serve to inhibit a connecting unit's activation. This non-linear switching capability makes a neural network extremely powerful in terms of being able to represent very complex input-output functions. This network topology with linear activation functions could not learn the XOR problem shown in Figure 1.

2.2 Training

Training is a two step process: a forward (static) pass from input to output and a backward (dynamic) from output to input. With the weights initialized to random values, the initial IO relations for the data points or training patterns are obviously erroneous. The forward path is static since the weights are not changed at this step. The error, the difference between the desired value (reference) and the actual value, is generated during the forward pass. Via a learning algorithm such as back-propagation the weights are adjusted so that each weight is assigned some blame for the output error and is adjusted accordingly. This adjustment occurs from output toward input, the backward pass through the net. In the same way the weights and inputs excite or inhibit the activation of nodes and ultimately constitute the net output in the forward pass, the weights multiply the error signal and induce a synaptic change in the backward pass. In other words, the error is "back-propagated" through the net. The weights are adjusted according to the gradient of the error as a function of weight.

The weights do not completely adjust in one epoch, a pass through all the IO relations in the training set. It may take many thousands or even hundreds of thousands of cycles to train a net. If the net is to adjust the weights along the steepest gradient of the error, progress must be slow. One training pattern may adjust the weights in one direction, while the next pattern may adjust the weights in the opposite direction. The two patterns compete. As long as training is slow and the weights do not make any large changes in one time step, the training patterns will often compromise and the net will converge to a global solution. In topologic terms, this convergence process is a descent down an error surface which may, depending upon the complexity and number of training patterns and the network configuration, be a very complex, multi-dimensional mountain range. If the path of steepest descent hits a valley in this error surface, which is not the global minimum for all training patterns, the training progression of the network may stop. The error measure may never go below a certain value. The weights may also wander aimlessly if there is a plateau on the error surface, leaving the competition between training patterns unresolved. The intricacies of this and other problems is beyond the scope of the present investigation.

2.2.1 Back-Propagation

The back-propagation algorithm, as its name indicates, propagates the network error backward through the network from the output toward the input adjusting the weights to minimize a cost index such as the total sum squared error (TSS) over the output patterns. A derivation of the algorithm is given in (Rumelhart, Hinton and Williams 1986).

$$TSS = \frac{1}{2}\sum(R_k - o_k)^2 \tag{2}$$

At this juncture it is prudent to mention that there are two types of back-propagation in common usage. One type updates the weights after each training pattern, thereby minimizing the TSS. Another variety in which error accumulates over the entire training epoch and adjusts the weights according to a slightly different algorithm. The former type, which minimizes the TSS, is the subject of this investigation since it is particularly conducive to dynamic analysis. The epoch update algorithm or other types of gradient decent methods could be adapted for dynamic analysis. However, the pattern update training is more simple in terms of the formulation of dynamic parameters. The application of the pattern update training algorithm proceeds down the steepest gradient of the error surface. Because the exact nature of the error surface is unknown, the algorithm is like a blind man feeling for the steepest path downward to some desired location. Since there are two layers in the example network, we will start with the update equation for the hidden-to output weights. The weights are adjusted proportionally to the negative gradient of the output error as a function of the weight which is

$$\Delta w_{ij} \alpha - \frac{\partial E_p}{\partial w_{ij}} \tag{3}$$

The update equation for the output layer weights takes the form

$$\Delta w_{jk} = \eta \delta_k o_j = \eta (R_k - o_k) o_j \tag{4}$$

The weight update equation for the non-output weights takes the form

$$\Delta w_{ij} = \eta \delta_j o_i = \eta (\sum \delta_k w_{jk}) o_i \tag{5}$$

Note that the error signal for the jth layer neurons is the error signal from above multiplied by the weights connecting from above. In intuitive terms, the error signal δ propagates down the weights in the reverse path from output toward input in the same way the input signal moves upward through the net in the forward path.

2.3 Lagrangian Dynamics

Lagrangian and Hamiltonian dynamics formulations are based on the concept of energy rather than force in the more familiar Newtonian dynamics. The genesis of both formulations lies in the calculus of variations and its application to physics and is not germane to the present discussion (Petrov 1968). In essence, both methods use the energy of components in a system to generate differential equations of motion which describe the behavior of dynamic objects in the system. Dynamic objects are not necessarily those that involve physical motion. They may be components in an electrical system, thermal system or even as general as aircraft landing patterns (Sarti, Walsh and Sastry 1993). This generality can extend to neural networks as is shown in the subsequent analysis.

A simple example of a block sliding along a horizontal surface illustrates the use of Lagrange's equations of motion. The Lagrangian formulation results in a second order differential equation for each degree of freedom in the system. For a block sliding on a surface, the familiar equation of motion is

$$F = ma + bv = F\ddot{x} + b\dot{x} \tag{6}$$

where F is the force, m is a mass and b is a coefficient of friction. The dot denotes differentiation with respect to time in this one degree of freedom (x) system. Lagrange's equations of motion including Rayleigh's dissipation function, which accounts for dissipated energy, are given by

$$F_i = \frac{d}{dt}\frac{\partial(L)}{\partial\dot{x}_i} + \frac{\partial(L)}{\partial x_i} + \frac{\partial(D)}{\partial x_i} = \frac{d}{dt}\frac{\partial(T-V)}{\partial\dot{x}_i} + \frac{\partial(T-V)}{\partial x_i} + \frac{\partial(D)}{\partial x_i} \qquad (7)$$

where T is the kinetic energy of the system, V is the potential energy of the system, L is the Lagrangian defined as the difference between the kinetic and potential energy of the system. D is Rayleigh's dissipation function:

$$D \equiv \frac{1}{2}\mu\dot{x}^2 \qquad (8)$$

where μ is a coefficient of dissipation, such as a friction coefficient, and x is the velocity of the degree of freedom associated with the dissipation coefficient, i.e. the friction coefficient is associated with the linear velocity of the mass. The kinetic energy of the block is

$$T = \frac{1}{2}mx^2 \qquad (9)$$

The potential energy of the block zero, assuming the block is on a level surface. The dissipation function is

$$D = \frac{1}{2}b\dot{x}^2 \qquad (10)$$

Inserting these values into Lagrange's equations of motion yields

$$F_i = \frac{d}{dt}\frac{\partial(L)}{\partial\dot{x}_i} + \frac{\partial(L)}{\partial x_i} + \frac{\partial(D)}{\partial x_i} = \frac{d}{dt}\frac{\partial(\frac{1}{2}m\dot{x}^2)}{\partial\dot{x}_i} + \frac{\partial(\frac{1}{2}m\dot{x}^2)}{\partial x_i} + \frac{\partial(\frac{1}{2}b\dot{x}^2)}{\partial x_i} \qquad (11)$$

which results in Equation (6).

The example of a block sliding on a surface has only one energy term. Lagrangian mechanics is particularly powerful in that several different energy terms corresponding to different objects and degrees of freedom can be incorporated into the equations of motion without complicated free body diagrams and force interactions. The example of the equations of motion for an inverted pendulum on a cart, contained in Appendix A-1, demonstrates the modularity and the power of the Lagrangian approach.

2.4 Hamiltonian Dynamics

A related energy method with similar generality and some advantages in terms of system identification is Hamiltonian dynamics. Hamiltonian equations of motion can be derived quite simply from the Lagrangian equations. While both are energy methods, Hamilton's equations result in two coupled first order differential equations for each degree of freedom rather than one second order equation in the Lagrangian case. The equations describe the rate of change of the generalized coordinate with respect to time (generalized velocity) and the rate of change of the generalized momentum with respect to time. Hamilton's equations are

$$\dot{q} = \frac{\partial H}{\partial p} \tag{12}$$

$$\dot{p} = -\frac{\partial H}{\partial q} \tag{13}$$

where the Hamiltonian is the total energy of the system (T+V), p is the generalized momentum, and q is the generalized coordinate. The time derivative of momentum is force. For Equation (13), partial differentiation is taken with respect to the generalized coordinate only. Outside forces, therefore, do not influence the dynamic equations unless they are potential forces, such as gravity, which depend on the generalized coordinate. If the mass of the system is constant, the time derivative of the momentum is the product of mass and acceleration, or potential force. Hamiltonian dynamics, therefore, is not concerned with the interaction of the system with the outside world, excluding potential forces, but rather with the nature of the system itself.

In addition, if terms such as velocity and momentum are not intuitive, as is the case for a neuron, the Hamiltonian is a powerful identification tool. The kinetic energy expressed in terms of momentum is

$$T = \frac{1}{2}mv^2 = \frac{p^2}{2m} \tag{14}$$

A simple example involves a block sliding down a frictionless ramp. The kinetic energy of the block is given by Equation (14), above. The potential energy of the block is

$$V = mgy \tag{15}$$

The Hamiltonian is the therefore

$$H = T + V = \frac{p^2}{2m} + mgy \tag{16}$$

and Hamilton's equations give

$$\dot{q} = \frac{\partial H}{\partial p} = \frac{\partial(\frac{p^2}{2m} + mgy)}{\partial p} = \frac{p}{m} = \text{velocity} \tag{17}$$

2.3 State Functions for Various Dynamic Systems

Mechanical systems and the corresponding equations of motion are familiar due to

$$\dot{p} = -\frac{\partial H}{\partial q} = -\frac{\partial(\frac{p^2}{2m} + mgy)}{\partial y} = mg = \text{potential force} \tag{18}$$

their prevalence in engineering practice. To illustrate the generality of Lagrangian dynamics, Table 2 shows the generalized "building blocks", coordinates, velocities, energies, forces, and momenta, for most mechanical and electrical systems.

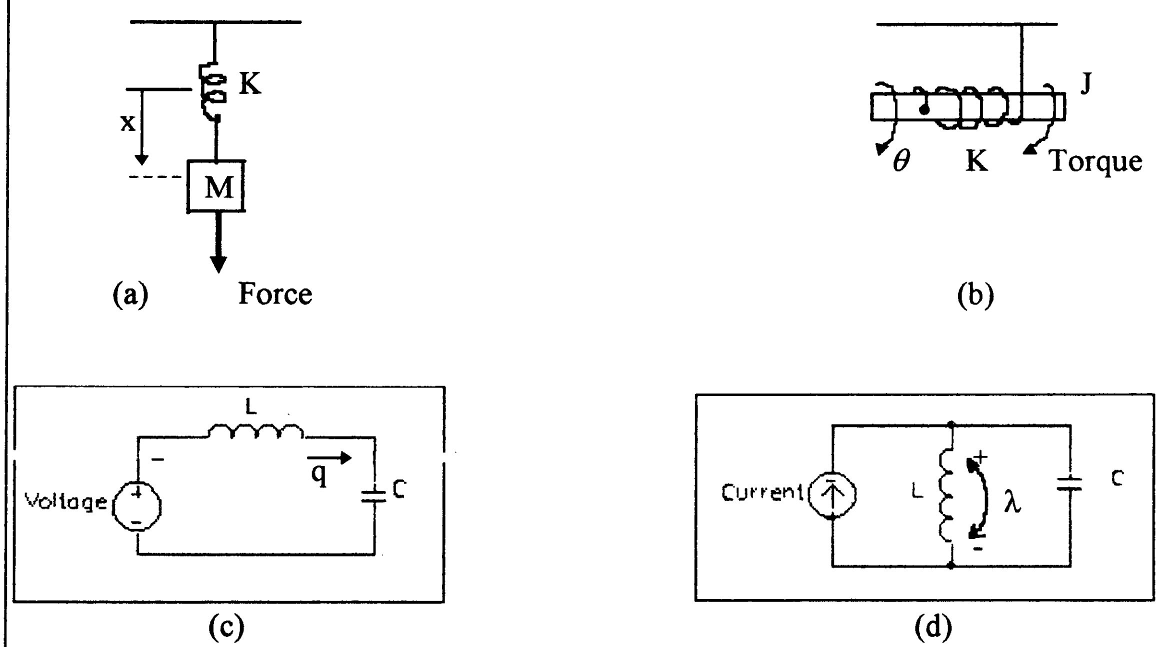


Figure 3 Simple Analog Electrical and Mechanical Circuits (a) Mechanical translational; (b) mechanical rotational; (c) electrical series; (d) electrical parallel.

Type of System	Generalized Coordinate q	Generalized Velocity $\dot{q}$	Kinetic Energy $T(q, \dot{q})$	Potential Energy $V(q)$	Generalized Momentum p	Generalized Force F
Mechanical translational	Position x	Velocity $\dot{x} = v$	$\frac{1}{2} M\dot{x}^2$	$\frac{1}{2} Kx^2$	Momentum $M\dot{x}$	Force
Mechanical rotational	Angular Position θ	Angular velocity $\dot{\theta} = \omega$	$\frac{1}{2} J\dot{\theta}^2$	$\frac{1}{2} K\theta^2$	Angular Momentum $J\dot{\theta}$	Torque
Electrical series	Charge q	Current $\dot{q} = i$	$\frac{1}{2} L\dot{q}^2$	$\frac{1}{2C} q^2$	Flux Linkage $L\dot{q}$	Voltage
Electrical parallel	Flux Linkage λ	Voltage $\dot{\lambda} = \upsilon$	$\frac{1}{2} C\dot{\lambda}^2$	$\frac{1}{2L} \lambda^2$	Charge $C\dot{\lambda}$	Current

Table 2 Generalized Building Blocks of Dynamic Systems (Schultz and Melsa 1967)

The terms are familiar in a physical (mechanical) context. However, the electrical counterparts may be somewhat less familiar, but still conform to the same physical principles described by Lagrangian and Hamiltonian Dynamics. The key to determination of the equation of motion for a given system is finding the energy, either in terms of mass and velocity for the Lagrangian case or momentum and velocity for the Hamiltonian case. The constituents of both forms of energy are mass and velocity. For a mechanical system, mass and velocity are familiar and intuitive concepts. However, when one considers other types of systems, thermal or electrical, for example, intuition fails. As a first step, the system has a generalized coordinate. It is generalized in the sense that the coordinates are different for each type of system; however, they all embody the same concept: a measure that changes or "moves" over time.

Not every dynamic measure is a candidate for a generalized coordinate; the choice must be made so that the system formulation is consistent. The resulting formulation for energy has the form and units of energy. For example, voltage in an electrical circuit changes over time as a function of system parameters like resistance and capacitance. It is not the generalized coordinate. Rather, it is a generalized velocity or the time integral of the generalized coordinate. The generalized coordinate, therefore, is the flux linkage. The energy is computed using the velocity and mass. What is the mass of a circuit? Again, the mass for each type of system is different, but the concept is the same for every system. Mass has familiar inertial properties.

To return the electrical system, the mass is the capacitance. An object with a large mass requires more power to accelerate. It tends to store energy. In a similar manner a system with a large capacitance requires more power to change the voltage (velocity). It tends to store energy. The formulation for kinetic energy is the same for each system:

$$KE = \frac{1}{2}mv^2 \tag{19}$$

For this example, the energy is

$$KE = \frac{1}{2}Cv^2 \tag{20}$$

The potential energy of a system depends only on the generalized coordinate. Compression of a spring or an object held some distance of the ground are both examples of potential energy: the object accelerates due to a forcing function that depends only upon position. The potential energy in the present example is

$$PE = \frac{1}{2L}\lambda^2 \tag{21}$$

where λ is the flux linkage and L is the inductance. The reciprocal of the inductance is analogous to a spring constant. The higher the spring constant, the stiffer the system and the more energy it stores for a given deflection. The larger the inductance, the stiffer the system , and the more energy it stores. The Lagrangian equations of motion are (Ogata 1978):

$$F_i = \frac{d}{dt}\frac{\partial(T-V)}{\partial x_i} + \frac{\partial(T-V)}{\partial x_i} = \frac{d}{dt}\frac{\partial(\frac{1}{2}Cv^2 - \frac{1}{2L}\lambda^2)}{\partial v} + \frac{\partial(\frac{1}{2}Cv^2 - \frac{1}{2L}\lambda^2)}{\partial \lambda} \tag{22}$$

$$I = Cv + \frac{\lambda}{L} \tag{23}$$

where I is the current or generalized force.

CHAPTER 3

NEURAL DYNAMICS

This chapter develops the theoretical results which are empirically verified and expanded in Chapter 4.

3.1 Quadratic Form of the Weight Update Equation

The weight update equation for the linear neuron can be written as a quadratic function of the current input.

$$\Delta w = \eta \delta o_i = \eta (R - o_j)\, o_i = R o_i - w_{ij} \eta o_i^2 \tag{24}$$

$$w_{ij} \eta o_i^2 - R o_i + \Delta w = 0 \tag{25}$$

The roots of this equation give the change in weight as a function of the current weight, reference and input. Since each weight value represents a velocity with a particular energy state, the smaller root represents a lower energy state that the larger root. In keeping with the intuitive nature of a dynamic system, the weight trajectory which corresponds to the smaller root of Equation (24) has lowest possible energy consistent with the weight update equation and is therefore the correct weight choice. After some algebraic manipulation, Equation (26) is written in the form

$$R = \frac{\Delta w}{\eta \, o_i} + w\, o_i \approx \frac{\dot{w}}{\eta \, o_i} + w\, o_i \tag{26}$$

Notice that this is the classical form of a first order dynamic system. This is analogous to the torque/ angular velocity behavior of a motor with the generalized mass being

$$mass = \frac{1}{\eta \, o_i} \tag{27}$$

and the generalized dissipation

$$dissipation = o_i \tag{28}$$

and the generalized force

$$Force = R$$

Many such analogies to mechanical and electrical systems can be drawn. Some dynamic parameters for a linear neuron are listed in Table 3.

Type of System	Generalized Coordinate	Generalized Velocity	Mass	Generalized Momentum	Kinetic Energy	Generalized Force
Neuron (LTI)	$\int w dt$	w	$\frac{1}{\eta o_i}$	$\frac{w}{\eta o_i}$	$\frac{w^2}{2\eta o_i}$	REF = $\delta + o_j$

Table 3 Dynamic Parameters for a Linear Neuron

3.1.1 Generalized Coordinate for a Neural System

Since the synaptic weight is the parameter which changes through time, it or its time integral are the most likely candidates for the coordinate. Recall that in an electrical circuit, with voltage as a time varying parameter, has the integral of the voltage over time, or the flux linkage, as the coordinate. Similarly and perhaps more familiarly, the angular velocity of the armature is the generalized velocity. The generalized coordinate, therefore, is the time integral of the velocity or the angular position. Both these first order systems are really second order systems in disguise. For example, the equation for the motor can be written

$$T = I\dot{\omega} + \mu\omega = I\ddot{\theta} + \mu\dot{\theta} \tag{29}$$

Since the linear neuron is ostensibly a first order system the logical choice for the generalized coordinate is the time integral of the weight value. It follows that the weight is really the generalized velocity of this system. It also follows that if the weight is the generalized velocity, the change in weight over one time step is the generalized acceleration. By looking at the describing differential equation, in consideration of the generalized velocity and acceleration, the generalized mass and dissipation terms are apparent. As a side note, since the neural system is discrete, the equation of motion is a difference equation rather than a differential equation. In order to draw a parallel between a physical system and the simple neural system, I first derive a first order difference equation for the neuron. Mass and dissipation terms are apparent from this Equation (26).

3.1.2 Generalized Neural Force

Unlike physical systems there are two distinct parts to the forcing function in Equation(26). The first term is simply the weight update equation

$$\Delta w = \delta_e \eta\, o_i \tag{30}$$

or

$$\delta_e = \frac{\Delta w}{\eta\, o_i} \tag{31}$$

This kinetic portion of the describing differential equation is analogous to

$$F = m\ddot{x} \tag{32}$$

The error signal forces the weight change in inverse proportion to the learning rate and output of the ith neuron. For a constant force or error in the present case, and if the mass remains constant, the generalized acceleration or the change in the weight will be constant. The weight update equation is only part of the picture, however. The forcing function for the dissipative part of the system is

$$o_j = o_i w_{ij} \tag{33}$$

which is the output of the upper (jth) unit. The total forcing function, which conveniently lends itself into dissipative and kinetic decomposition, is therefore

$$o_j + \delta_e = \frac{\Delta w}{\eta\, o_i} + o_i w_{ij} = R \tag{34}$$

which is the sum of the output of the jth unit and the error signal impinging upon it. Since the error is the difference between the reference and the output, the sum of the error and output is simply the reference value or the total force.

3.1.3 Generalized Neural Mass

Mass has inertial properties, such as a large mass requires greater force for a given acceleration than a small mass. For example, the lower the learning rate and input to the ith layer, the more force (higher reference value) is required to produce a given acceleration (weight change). In addition, the weight will remain constant if the error signal is zero. Another way of viewing this is "an object in motion tends to remain in motion at a constant velocity and direction until acted upon by an outside force".

3.1.4 Neural Dissipation

As more energy is dissipated, the kinetic forcing function, or error, decreases and the dissipative forcing function, the output of the jth unit, increases. The converse is also true. Of course, this presumes a constant total forcing function or reference value. The speed curve of a motor, for example, rises to some final velocity where the amount of energy dissipated equals the energy put into the system. The amount of energy dissipated is some coefficient of friction (dissipation) multiplies by the angular velocity of the armature. The weight in the neural system also rises toward some final value. If the equations describing the motor and the synaptic weight change are structurally the same, some factor multiplies the generalized velocity (weight value) to produce dissipated energy. In light of Equation (34), the coefficient of dissipation must be the output of the neuron on the ith layer which is the input in our neural system.

3.2 Neural Time Constants

Once the mass and dissipation are known, the time constant follows simply mass/dissipation or

$$\tau_{theoretical} = \frac{1}{\eta\, o_i^2} \tag{35}$$

Notice that this relation is approximate for the neuron. The gradient decent algorithm is not a continuous function; mass and dissipation are not acting at the same instant in time as is the case for physical systems. Thus, the time constant takes the form

$$\tau_{actual} = \frac{-1}{\ln(1-\eta\, o_i^2)} \tag{36}$$

3.3 Neural Lagrangian

The Lagrangian equations of motion do not give the equations of the synapse through time unless the input is constant. The formulation (based in the calculus of variations) does not permit discontinuities in the energy function. A neural network is a discrete and therefore the energy functions are discontinuous. As an input changes, the mass an dissipation change instantaneously, which violates the foundations of Lagrangian dynamics. The energy of the neuron is piecewise continuous. The Lagrangian equation of motion is piecewise applicable.

The Lagrangian is the difference between the kinetic and potential energies of the neural system.

$$L = T - V \tag{37}$$

The kinetic energy is

$$T = \frac{w_{ij}^2}{2\eta o_i} = \frac{1}{2} m\dot{x}^2 \tag{38}$$

The potential energy is zero.

$$V = 0 \tag{39}$$

The dissipation function is

$$D = \frac{1}{2}\frac{\partial\left(o_i w_{ij}^2\right)}{\partial w_{ij}} \tag{40}$$

Inserting into the Lagrangian equations of motion gives

$$F = \frac{\partial\left(\frac{w_{ij}^2}{2\eta o_i}\right)}{\partial w_{ij}} - 0 + \frac{\partial\left(o_i w_{ij}^2\right)}{\partial w_{ij}} \tag{42}$$

$$\begin{aligned} F &= \frac{1}{\eta o_i}\dot{w} + o_i w \approx \frac{1}{\eta o_i}\Delta w + o_i w \\ &= \delta + o_j = \text{Error} + \text{Output} = \text{Reference} \end{aligned} \tag{42}$$

which is the same result derived in Section 3.1 using the quadratic formula.

3.4 Neural Hamiltonian

The momentum of a neuron is

$$p = \frac{\dot{w}}{\eta o} \tag{43}$$

and the kinetic energy is

$$T = \frac{p^2}{2m} = \frac{(\frac{\dot{w}}{\eta o})^2 \eta o}{2} = \frac{\dot{w}^2}{2\eta o} \tag{44}$$

As we have defined the parameters of the synapse, there is no potential energy and the Lagrangian and Hamiltonian are the same. Hamilton's equations give

$$\dot{q} = \frac{\partial H}{\partial p} = \frac{\partial\left(\frac{\left(\frac{\dot{w}}{\eta o}\right)^2 \eta o}{2}\right)}{\partial p} = \dot{w}$$
$$\dot{p} = -\frac{\partial H}{\partial q} = -\frac{\partial\left(\frac{\left(\frac{\dot{w}}{\eta o}\right)^2 \eta o}{2}\right)}{\partial q} = 0 \qquad (45)$$

Note that the time derivative of momentum is force. This is only potential force since the partial derivative of the Hamiltonian is taken with respect to the generalized coordinate and forces that depend only on the generalized coordinate are potential forces. The potential force is zero in the neural system because the potential energy is zero. Both the Lagrangian and Hamiltonian formulations are consistent with the choices for mass and dissipation.

3.5 The First Order Neural System

Through the quadratic form of the weight equation and the supporting Lagrangian and Hamiltonian equations of motion, the system is found to be first order. As such, the weight trajectories exhibit familiar "rising" and "falling" curves similar to other first order systems such as a motor or charging/ discharging capacitor. Some example trajectories are shown in Figure 4.

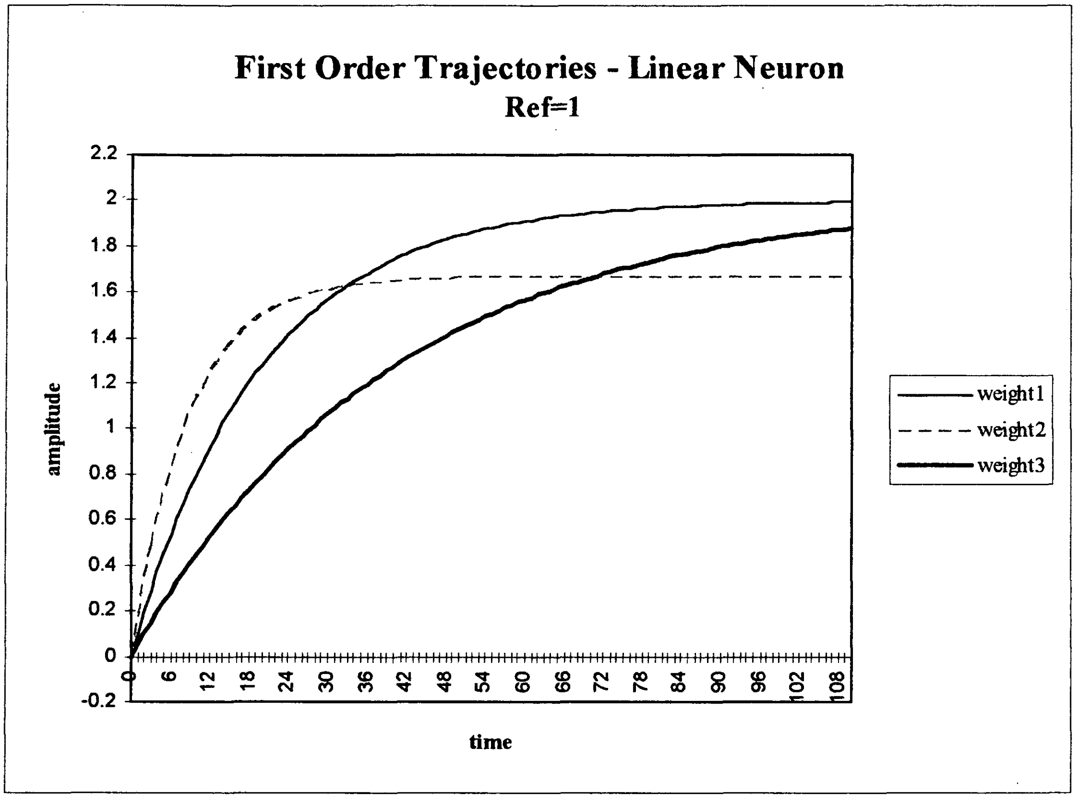


Figure 4 Rising first order weight trajectories for the LTI neural system

Of course, the weight also exhibit first order falling behavior for negative inputs or reference value. As a first order system, the weight rise or fall to a steady state value. This steady state (final) weight value or stationary point for the LTI neural system is simply

$$w_{final} = \frac{R}{o_i} \tag{46}$$

The equation describing the evolution of the weight through time is

$$w = w_{\text{final}}\left(1 - e^{-\frac{t}{\tau}}\right) \tag{47}$$

where time is the number of training cycles that take place along the trajectory.

3.5.1 Non-Zero Initial Conditions

Since neural networks are initialized with non-zero random values, the first step toward the modeling of a non-linear neuron is the accommodation of non-zero initial conditions. Non-zero initial conditions also have ramifications for the time varying neural systems we shall examine subsequently. If the initial weight value is non-zero, the exponential rise or fall time still applies, but the weight is merely closer to or farther from its final value. Another way to consider the non-zero initial weight starting at time zero is a zero initial weight starting with some non-zero time. Figure 5 illustrates this concept.

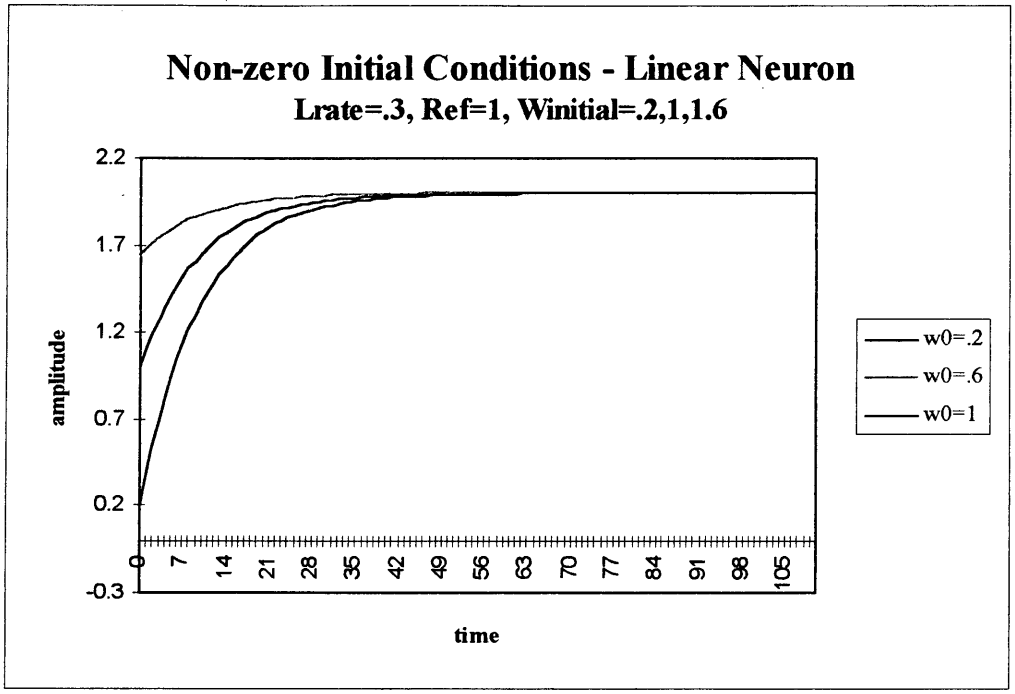


Figure 5 Non-Zero Initial Conditions for a Linear Neuron The training parameters for all these LTI neural weight trajectories are the same except the initial conditions on the weights.

The time corresponding to the initial weight is calculated from

$$t = -\tau \ln\left(1 - \frac{w}{w_{final}}\right) \tag{48}$$

For a changing training pattern, the time constant and final weight value will change with each presentation. The initial condition for each weight trajectory is the weight value from the end of the previous trajectory corresponding the previous presentation. If the value of time in Equation (48) is used in the exponential weight formula an interesting pattern emerges.

3.5.2 First Weight Values and Scaling Factors

The weight value after the first presentation of a training run (starting with zero initial conditions) is given by

$$w_{first} \equiv w_{final}\left(1 - e^{-\frac{1}{\tau}}\right) \tag{49}$$

For a constant input, the change in weight decreases for each presentation. The system decelerates or loses energy. For a constant input, the value of the weight at the next time step is:

$$w = w_{final}\left(1 - e^{-\frac{2}{\tau}}\right) = w_{final}\left(1 - e^{-\frac{1}{\tau}}\right) + w_{final}\left(1 - e^{-\frac{1}{\tau}}\right)e^{\frac{-1}{\tau}} \tag{50}$$

As is shown above, the trajectory at the second time step can be decomposed into First Weight values and Scaling Factors. The First Weight value is defined by Equation (49) above. The scaling factor is the exponential term multiplying the first weight value in the second term of Equation (50). If the inputs are not constant, the relation still holds. For example, if a training epoch contains three patterns the weight value at the second time step is given by

$$w_2 = w_{final2}\left(1 - e^{-\frac{1}{\tau 2}}\right) + w_{final1}\left(1 - e^{-\frac{1}{\tau 1}}\right)e^{-\frac{1}{\tau_2}} \tag{51}$$

The pattern continues. For example, the weight value for the third time step is given by

$$w_3 = w_{f3}\left(1-e^{-\frac{1}{\tau 3}}\right) + w_{f2}\left(1-e^{-\frac{1}{\tau 2}}\right)e^{-\frac{1}{\tau_3}} + w_{f1}\left(1-e^{-\frac{1}{\tau 1}}\right)e^{-\frac{1}{\tau_2}}e^{-\frac{1}{\tau 3}} \qquad (52)$$

The weight value at and given time step is given by progressively scaled first weight values. Beyond the first epoch, the general value for subsequent weight values is

$$w_n = w_{\text{final n}}\left(1-e^{-\frac{1}{\tau_n}}\right) + w_{n-1}e^{-\frac{1}{\tau_n}} \qquad (53)$$

In words, Equation (53) states that the value for the weight at the current time step is the value of the first weight for that time step, as defined above, added to the previous weight value multiplied by the scaling factor for the current time step. This relation holds for both for linear and non-linear neural systems.

Since the actual time constant is logarithmic, the first weight value for each time step is the reciprocal of the theoretical time constant . For example. the actual logarithmic time constant in Equation (36) gives the First Weight, as defined above, for the nth input value to a linear neuron as

$$w_{first(n)} = \eta\, o_{i_n}^2 \qquad (54)$$

and the Scaling factor for the nth input value is

$$\lambda_n = (1-\eta\, o_{i_n}^2) \qquad (55)$$

These relations yield a particularly simple form for the weights as a function of time. The weight value at the end the k<u>th</u> epoch for q input patterns is

$$w_{k\underline{th}} = (w_{first(q)} + w_{first(q-1)}\lambda_q + w_{first(q-2)}\ \lambda_q\lambda_{q-1} + \ldots)(1 + \lambda_{1-q} + \lambda_{1-q}^2 + \ldots \lambda_{1-q}^{n-1})$$

$$w_{k\underline{th}} = \Psi(1 + \lambda_{1-q} + \lambda_{1-q}^2 + \ldots \lambda_{1-q}^{n-1}) \tag{56}$$

where $\lambda_{1\text{-}q}$ is the product of the scaling factors for all input patterns and ψ is the product of the first weights and the remaining scaling factors as indicated in Equation (56). The second term on the right hand side is also a convergent series as the number of training epochs approaches infinity, provided

$$(\lambda_{1-q}) \in (-1,1) \tag{57}$$

Therefore, we can predict the final steady state value when the weights have ceased to change for their particular input patterns. A note of clarification is in order at this juncture. Since the present system is linear, and if the particular reference values do not permit a single solution, i.e. a single steady state value for all inputs, the weights will undergo steady state oscillation or a limit cycle. The example below illustrates this. For an infinite number of training epochs, Equation (56) above becomes

$$w_{k\underline{th}} = \Psi \frac{1}{1 - \lambda_{1-q}} \tag{58}$$

For example, for a single linear neuron with two input patterns o_{i1}=.1, o_{i2}=.5, learning rate (η) =.1, and a reference value of 1 the weight value at the end of each epoch after the system reaches a limit cycle is found from:

$$\lambda_{1-2} = (1-\eta o_1^2)(1-\eta o_2^2) = (1-.5.1^2)(1-.5.5^2) = .870625$$
$$\Psi = w_{first(2)} + w_{first(1)}\lambda_2 = R_2\eta o_2 + R_1\eta o_1(1-\eta o_2^2) = .29375 \tag{59}$$
$$w_{\underline{nth}} = .29375\frac{1}{1-.870625} \approx 2.27053$$

Where Table 4 shows the data for the SISO neural system.

Epoch	Training Presentation	Training Input	SISO Weight
85	170	0.1	2.270526
85	171	0.5	2.309174
86	172	0.1	2.270527
86	173	0.5	2.309174
87	174	0.1	2.270527
87	175	0.5	2.309175
88	176	0.1	2.270528
88	177	0.5	2.309175
89	178	0.1	2.270528
89	179	0.5	2.309176
90	180	0.1	2.270529
90	181	0.5	2.309176
91	182	0.1	2.270529
91	183	0.5	2.309177
92	184	0.1	2.270529
92	185	0.5	2.309177
93	186	0.1	2.270530

Table 4 Training Data and Weights Values for the LTV System. The weight value for the steady state limit cycle is given by Equation (59).

3.6 Transition to Non-linear Behavior

A neuron with a non-linear activation function is similar to a neuron with a linear activation function in a structural sense. By structural I mean the components which comprise the system: an input value, an output value, a weight value, and an activation function which relates the output and to the input and weight values. It is reasonable to assume that the similarity should extend to dynamic parameters such as the generalized coordinate, mass, dissipation and forcing function. If we leave the strictly algorithmic realm for a moment, and consider the systems from the perspective of a free body diagram, the generalized force, or reference value for the present case, is an external input. In terms of a mechanical analogy, it pushes, or forces the weight (velocity) to another value. Since the external inputs to the non-linear and linear neuron are the same, it is reasonable to assume that the forcing function for both systems is the same. If this is the case, the resulting describing 1st order difference equation is similar.

3.6.1 Changes in the Learning Rule

The update rule for the non-linear system is similar to that of the linear system, but

$$\Delta w_{non-linear} = \eta \delta_e o_i = \eta \sigma' (\mathrm{R} - o_j)\, o_i \qquad (60)$$

there are some differences. In the update rule for the non-linear system, the derivative of the sigmoid activation function is incorporated into the error signal. In terms of the linear system, this is simply a constant multiplying the error value.

$$\Delta w_{linear w/const.} = \eta \delta_e o_i = \eta\, c\, (\mathrm{R} - o_j)\, o_i \tag{61}$$

This constant does affect the time constant of the system, as one would expect. For the non-linear neuron, c in Equation (61) is variable with neuron input, since it represents the derivative of the sigmoid at the given input level. Since the error signal is affected and the error signal is the forcing function for the kinetic part of the system (as opposed to the dissipative part), the mass term is changed. The non-linear mass term is simply

$$mass_{nl} = \frac{1}{c \eta o_i} \tag{62}$$

and resulting time-varying time constant is therefore

$$\frac{1}{\eta c o_i^2} \approx \tau_{linear} = \frac{-1}{\ln(1 - \eta c o_i^2)} \tag{63}$$

3.6.2 Activation Offset

The next step in construction the non-linear model is accommodation of the offset in the output function. Figure 6, below shows the sigmoid function

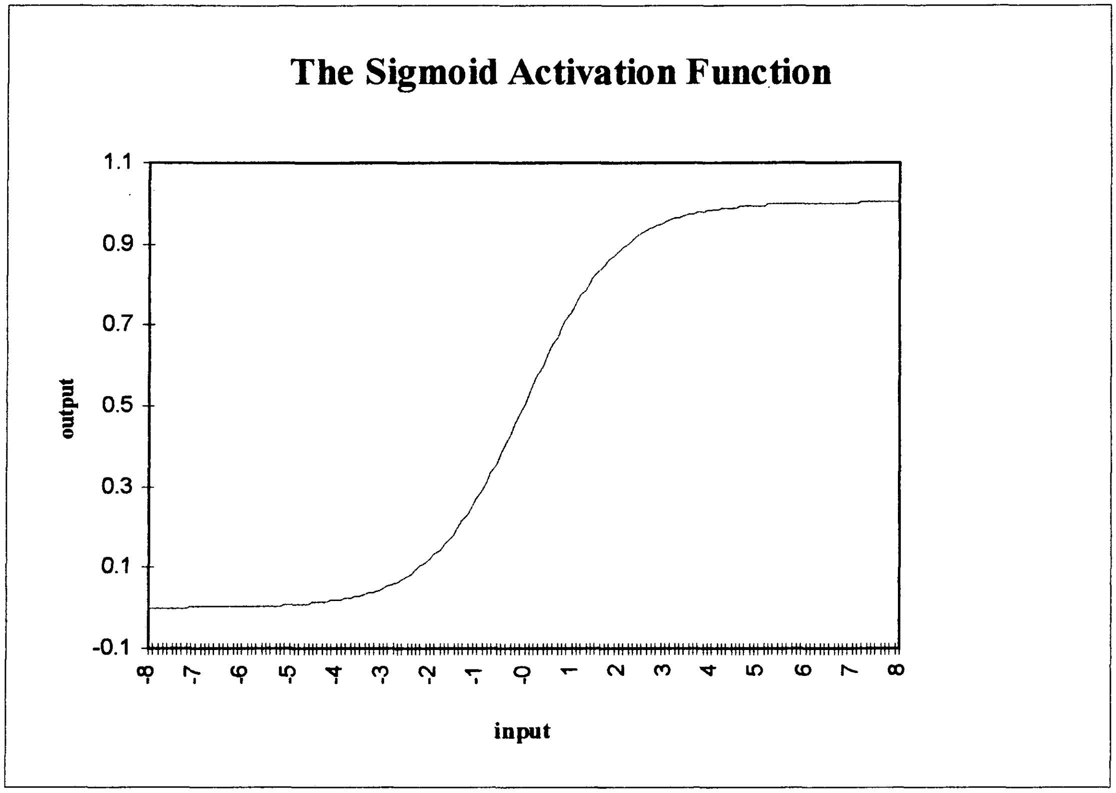


Figure 6 The Sigmoid Activation Function

The sigmoid activation function has a non-zero output, .5 in this case, for a zero input. There are two ways to view the addition of .5 to the output node in terms of the time constant formulated thus far. Either w_{final} remains the same as in the previous case (when examining changes to the mass term) and the time constant varies by a constant value, (the dissipation constant increases), or the time constant is unaffected and the value for w_{final} changes. While both viewpoints are theoretically valid, the latter view, a change in w_{final}, represents actual neuron behavior. Figure 7 shows a comparison between the weights and output values for offset and non-offset activation functions. Note that the time constant of the two trajectories appears to be the same, but the value of w_{final} is changed.

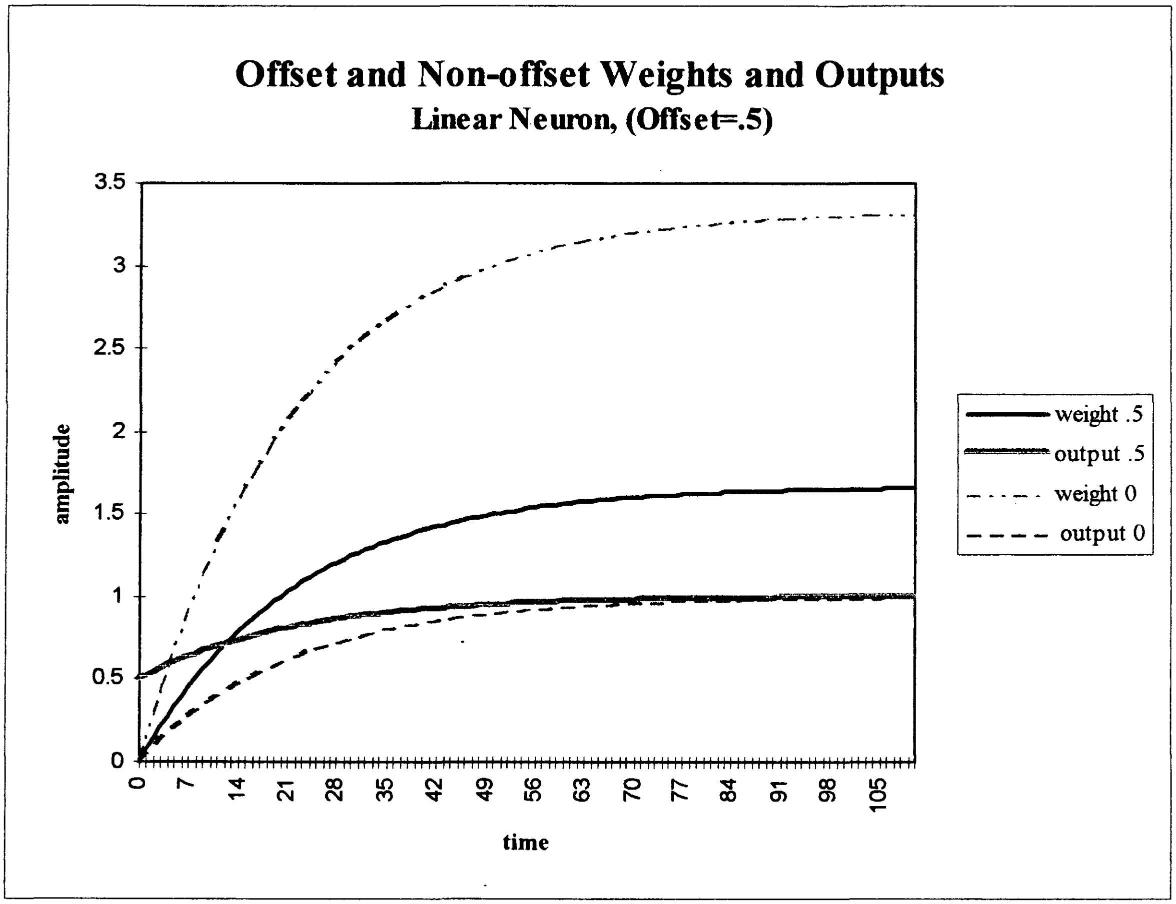


Figure 7 Weight and Neuron Output Values - Offset and Non-offset. The offset is .5 to roughly simulate a sigmoid function.

CHAPTER 4

RESULTS

The neural system is extremely unusual in that this research is performed entirely on a computer without the usual sources of experimental uncertainty such as electrical noise, instrument calibration, foreign particles, inconsistent experimental technique, etc.. In addition, the local behavior of the training algorithm is known. All the external inputs to this rather simple neural system are controllable and known to the accuracy of the computer and the spreadsheet. There is no sensor noise or other stochastic complications. This simple neural system provides a unique opportunity for exact system identification. By exact, I mean discrepancy between mathematical models and the neural system's behavior are only due to roundoff error in the computer. The roundoff error of the spreadsheet appears to be 10^{-14} or less, depending upon how many steps in calculation a particular value involves. The computer has a 64 bit mantissa, which corresponds to a minimum roundoff error of 10^{-16} for each floating point operation. Calculations pertaining to this investigation have roundoff error is in the 10^{-14} to 10^{-17} range. Results with errors in this range are termed "exact". The following chapter shows the results of the largely theoretical constructs of the previous chapter.

4.1 General Considerations

The objective of this investigation is the formulation of an accurate dynamic mathematical model of a single neuron with a sigmoid activation function . This objective is attained by progressive modification of a simple, linear neural system - a single weight and two abutting neurons, as is indicated in the previous sections. The investigation proceeds from LTI (linear time invariant) neural systems, to LTV (linear time varying) neural systems, to the final NLTV (non-linear time varying) system. The distinction between LTV and NLTV is subtle. While both systems are time-varying in with arbitrary sequences of input and/or reference values, and can be interpreted as non-linear by virtue of this randomness, I define a linear system as one that is completely mathematically predictable. This is predictable in the sense that the entire weight trajectory can be calculated by closed form equations. The fully non-linear activation function does not have this property. The weight at the next time step is dependent upon the weight at the current time step.

The theoretical justification for the particular model augmentations is developed in the previous pages. The empirical ramifications of each step are contained in the following results. The specific model progression is as follows. The LTI neuron models are a linear system, an input node and an output node connected by a single synapse, with constant training inputs and reference. The first model has the same configuration with non-zero initial weight conditions. The next model has a constant offset in the activation function. The LTV models are: a linear neuron excited by step inputs and a constant reference, oscillating inputs with a constant reference, ramp inputs with a constant reference and zero initial weights, ramp inputs with non-zero initial weights, a neuron with time varying offset on the activation function and the same system with time varying activation scaling. The final simulation involves the non-linear system with a training presentation - repeated training epochs of a 4 value input output relation. The following results will clarify the preceding theoretical analysis and organizational introduction.

4.2 LTI System - Constant Inputs

The weight trajectory for the linear system exhibits behavior of a first order system. Figure 8 shows three different weight trajectories for a SISO neuron corresponding to the three different input and learning rates in Table 5. Each neuron is trained for 110 iterations, which corresponds to the time axis of the graph. Of course the

Weight	Input	Learning Rate	Wfinal
1	0.5	0.2	2.00
2	0.3	0.6	3.33
3	0.5	0.1	2.00

Table 5 Training Parameters for the LTI System - Rising Trajectories

training inputs are constant for the entire trajectory. Note that weights 1 and 3 have the same training input value and are converging to the same final value. The only difference

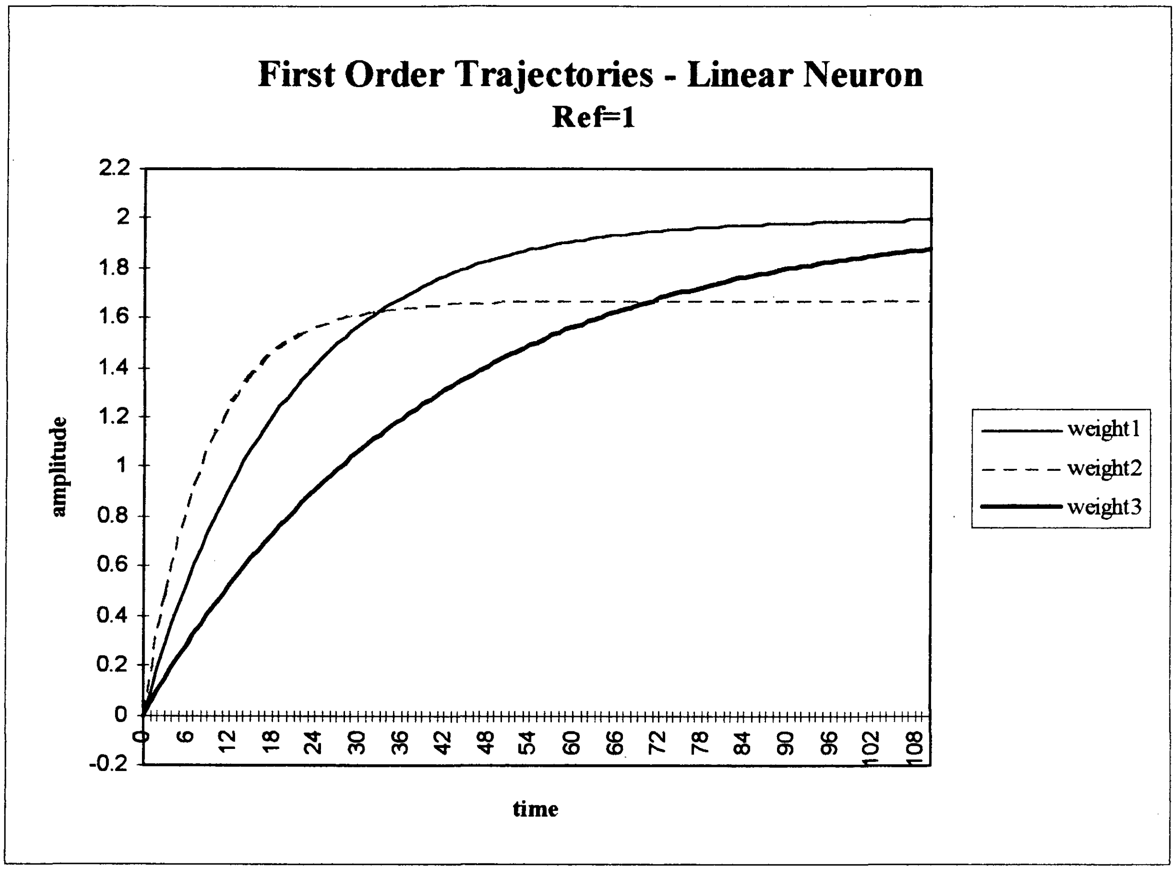


Figure 8 First Order Weight Trajectories. Parameters for this training are shown in Table 5.

between these two trajectories is the learning rate. The lower learning rate gives weight 3 a slower convergence resulting from a higher mass. System 2 (weight 2) has different dissipation as well as mass from the other two systems. These weights appear to be first order from graphical trends, but numerical data is conclusive. The measured time constant τ_i is computed from

$$\tau = \frac{-t}{\ln(1 - \frac{w}{w_{final}})} \tag{64}$$

where t is time (or training presentation number), w is the weight value at the time of data presentation (before weights are updated in training), is exactly constant. Figure 9 shows three different "falling" weight trajectories for a SISO neuron corresponding to the values in Table 7.

Weight	Input	Learning Rate	Wfinal
1	-0.5	0.2	-2.00
2	-0.3	0.6	-3.33
3	-0.5	0.1	-2.00

Table 6 Training Parameters for the LTI Neural System - Falling Trajectories

The same observations hold for the falling weights as the rising weights. Note that the inputs and w_{final} have the same absolute values as their counterparts in Table 5 but are negative.

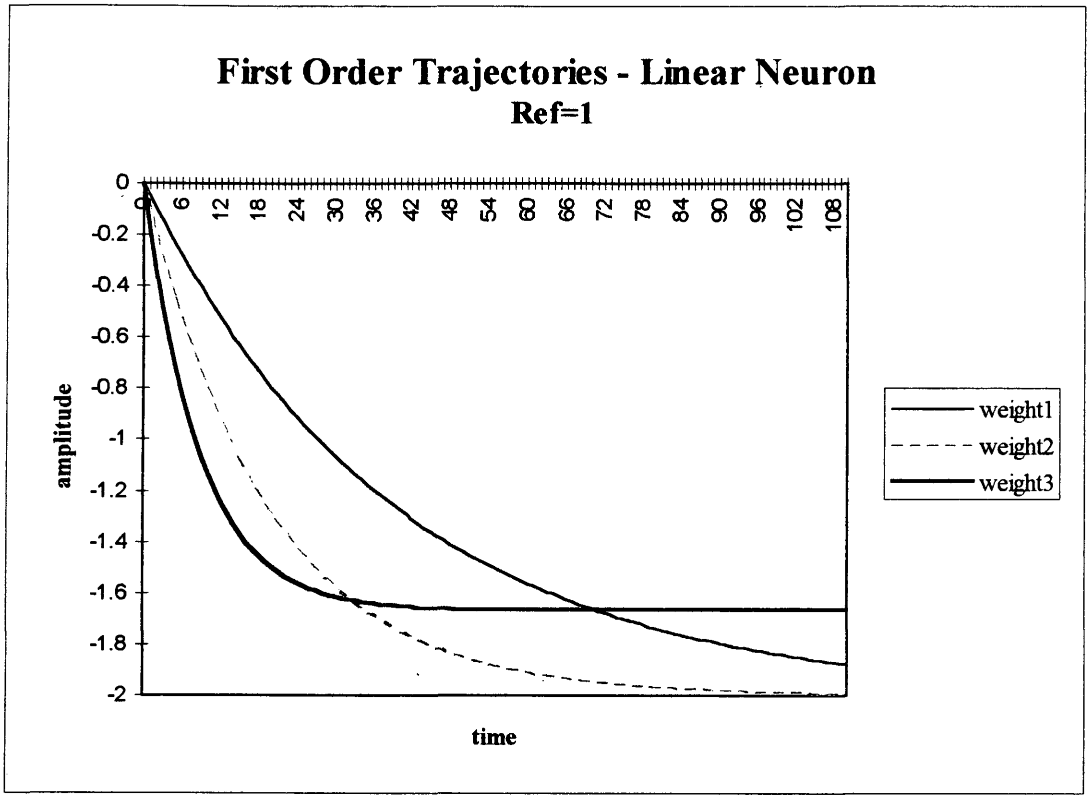


Figure 9 First Order Weight Trajectories - Falling

4.2.1 LTI System - Non-Zero Initial Conditions

In practice, weights are initialized with small, random, non-zero values. To accurately predict the weight trajectory the possibility of non-zero initial conditions must be incorporated into the model. There are two ways to approach this. The least complicated in terms of analysis is simply to add an initial value to the first order equation (47). However, the weight value at the present time step would be dependent upon the starting value at the previous time step. If this is the case, a model is almost useless, since it supplies trajectory information for one time step into the future. The other alternative is the addition of time to Equation (47). As is stated previously, the initial weight value corresponds to an initial time in the exponential equation

$$w = w_{final}\left(1 - e^{-\frac{t + tinitial}{\tau}}\right)$$

Rearranging and solving for the time gives

$$t = -\tau \ln\left(1 - \frac{w}{w_{final}}\right)$$

Using this compensated time constant the calculated weight trajectories exactly match the actual trajectories. If this concept is not clear, it is easy to imagine the weights staring from zero if the time axis is moved backward. The moving of the axis is the same as adding time to the exponential term. The trajectories predicted using the initial time compensation exactly match the actual weight values.

4.2.2 LTI System - Activation Offset

As is indicated in Chapter 2, the next step toward a non-linear model, is compensation for any offset in the activation function. Figure 10 shows the effect of offsetting, adding a constant to, the neuron's output. Two sets of trajectories are shown: a weight and output for an offset of zero and an offset of .5 Adding a constant offset to the output of the linear system does affect any of the parameters in the time constant, but it does appear to change the time constant. However it moves to a different w_{final}. With a positive offset, the system will reach equilibrium at a lower weight value than the same system without the corresponding offset. A more general form of w_{final} is

$$w_{final} = \frac{(\text{Ref - offset})}{o_i} \tag{65}$$

When using this value for w_{final}, the time constant is the same for the offset and non-offset systems.

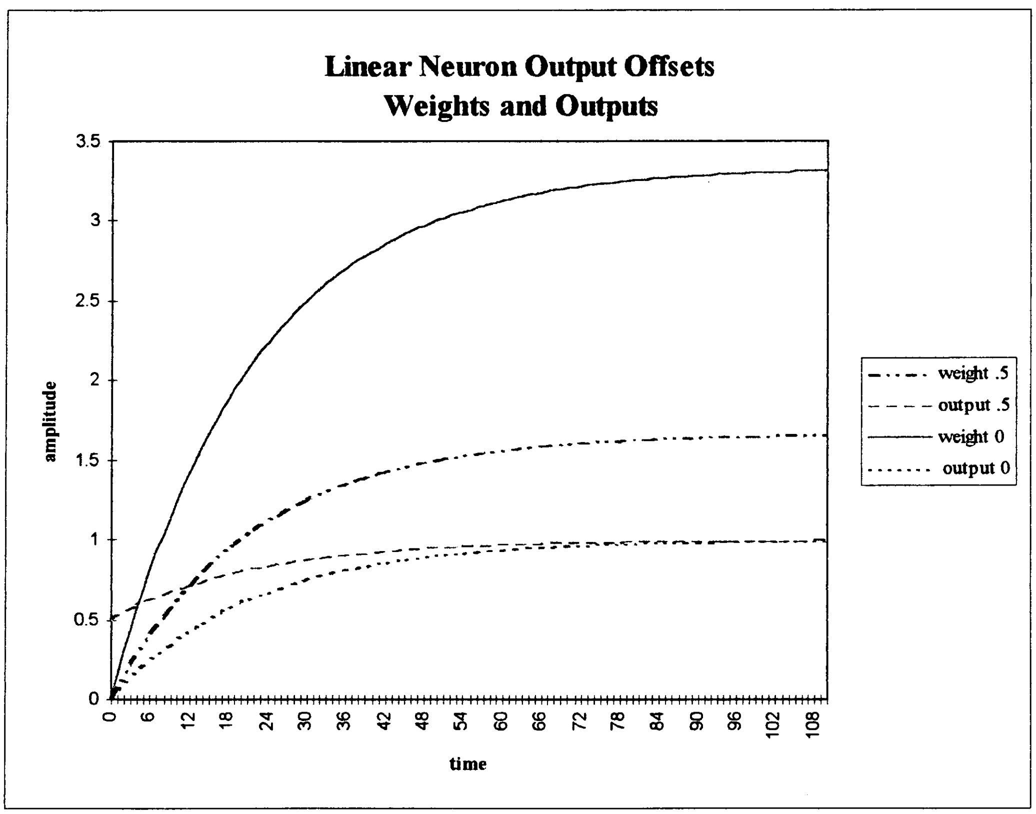


Figure 10 Offset and Non-offset Weights and Outputs

4.3 LTV System - Step Inputs

Once the parameters of the LTI system are verified, the next step is the formulation of an LTV model. If the system is excited by step inputs, the first order behavior still applies; each input has its own time constant with initial conditions from the weight value at the end of the previous input value. Figure 11 illustrates this. Each different input is heading toward some final value, depending only upon the value of the reference and the current input. Only on the third branch of the trajectory in the figure, where the weight curve is flat, does the weight attain its final value. This is expected, since this is the branch with the highest input, and the smallest time constant. For more usual training scenarios, a given input is only one time step in duration, and a time constant for each input is obscured. In addition, the time constant must be compensated for the initial conditions at the beginning of the training input: the weight value at the end of the last input. The only other difference between the present LTV and the previous LTI cases is a change in the final weight value for each new input. The final value is computed in the same manner as the previous LTI case, so the describing Equation (47) does not require modification. In terms of verification, again, the predicted values match the actual values exactly.

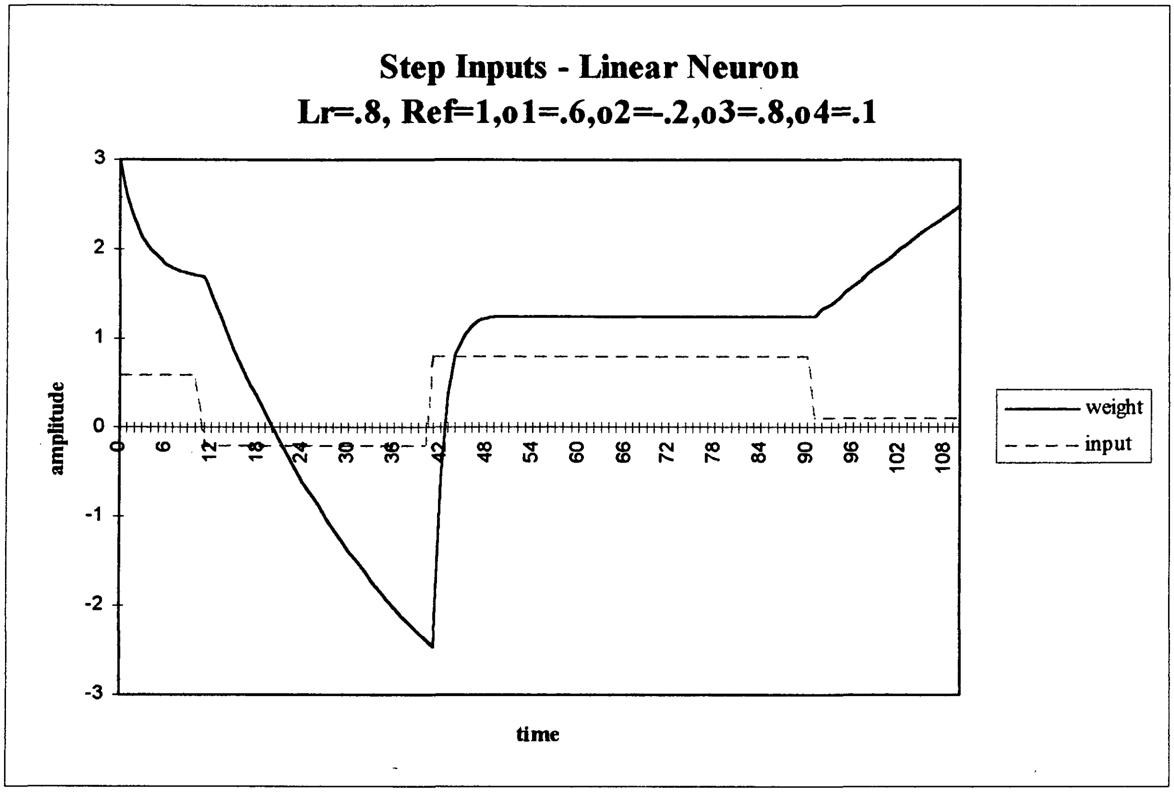


Figure 11 LTV System Step Inputs and Weight Trajectories

4.3.1 LTV System - Oscillating Inputs

The next step in the simulation process is the introduction of oscillating training inputs. This simulates actual training conditions. However, for the sake of simplicity and of verification of derivations two examples are given with the number of training patterns per epoch is limited to 2 and 4, respectively. Figure 12 shows a weight trajectory for two inputs, .1 and .5 presented in succession: Ref=1. Note that a single weight value cannot drive the error to zero for two inputs and a single reference in a linear system. Therefore, the system has two stationary points, one for each input value, which drive the error to zero for that input. Table 7 shows the numerical values of the weights from iterations 70 to 80, where it converges to its final value. The reader may note that the low input (.1) is along the same row with the low weight and vice versa. If the weights were listed after the update phase of training, the value would all move down one row, and the higher input would correspond to the lower weight and vice versa, as one might intuitively expect for a single reference value. As is seen form the table, the .1 presentation increases the weight, while the .5 presentation decreases the weight. The two patterns are competitive and neither attains its final value.

time (iteration)	Weight	Input (oi)	Wfinal
70	2.23294	0.1	10
71	2.31061	0.5	2
72	2.23296	0.1	10
73	2.31063	0.5	2
74	2.23297	0.1	10
75	2.31064	0.5	2
76	2.23298	0.1	10
77	2.31065	0.5	2
78	2.23299	0.1	10
79	2.31066	0.5	2
80	2.23299	0.1	10

Table 7 Weight Values and Input Data for Oscillating Inputs

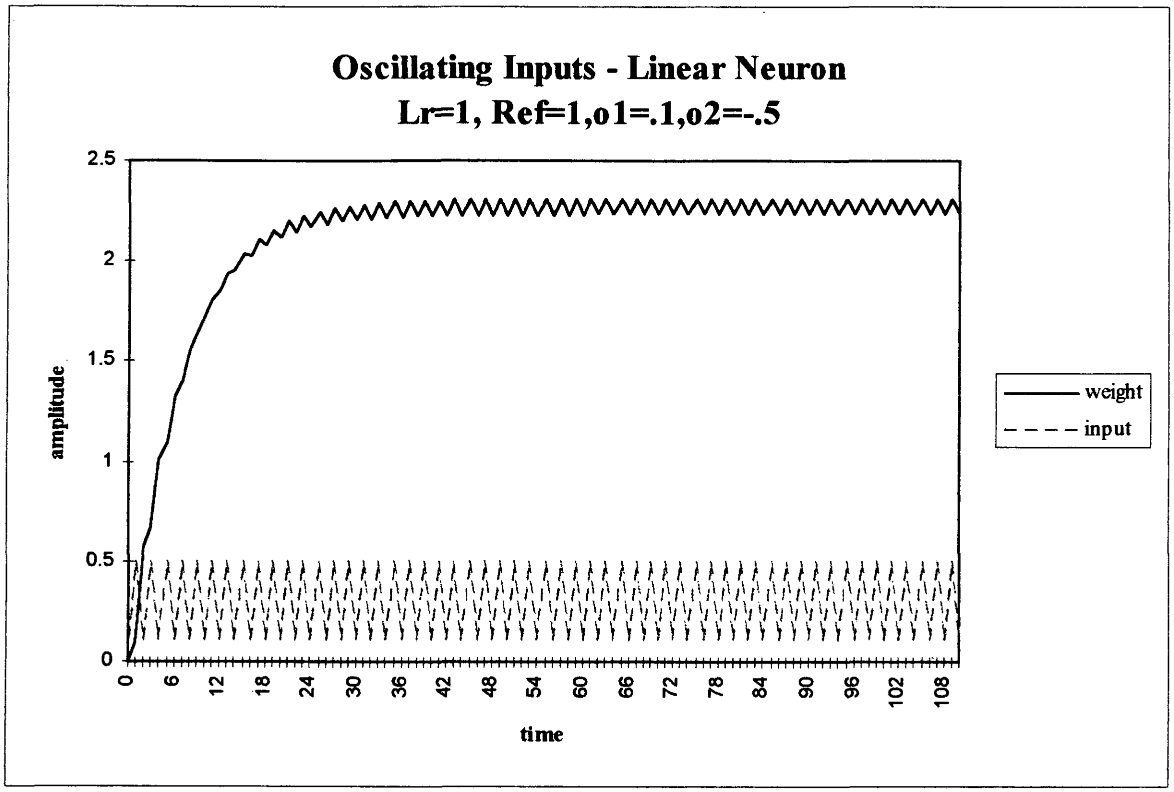


Figure 12 LTV System - Ramp Inputs

4.3.2 LTV System - Ramp Inputs

To magnify the weight oscillations and look at the effect of multiple input patterns, another training run was completed with inputs of .1,.2,.3,.4. In other words, this is a small ramp function $.1t$ where t is the time of the presentation inside the epoch. Figure 12 shows these results. Note that the peaks of the wave form are truncated near the end of the training session. Table 8 shows the numerical weight values, inputs and final weight

Reference = 1		Lrate=.8	
time	Input (oi)	Weight	Wfinal
107	0.4	3.3180	2.5
108	0.1	3.2394	10
109	0.2	3.2800	5
110	0.3	3.3213	3.3333

Table 8 Training Data for Ramp Input

values. None of the weights have reached their final values. However, they have reached a quasi-equilibrium where one input perturbs the weight slightly in one direction, while the others may move it in either the positive direction if $w_{final} > w_{current}$ or the negative direction if $w_{final} < w_{current}$. This is the classic local minimum problem. The distance a weight moves in one iteration, as can be seen from the first order rise or fall behavior, depends not only upon the learning rate and input, but also upon how far the current weight is from its final value. Note that the weights listed are at the beginning of the training phase before they are updated. Referring to Table 8, the .3213 weight value at the .4 presentation decreases during the weight update because the current value of the weight is higher than w_{final} at that time.

As the learning rate is pushed higher, each weight moves closer to its particular final value. In view of the competitive aspects of learning, some weights get closer while some are pushed farther away from their respective final values. For example, the weight in the .4 input row decreases steadily as the learning rate moves higher since $w_{final} < w_{current}$. The weight on the .3 input row also decreases slightly as the learning rate is pushed farther higher. While $w_{final} > w_{current}$, the starting value at the beginning of the presentation is lower, so the weight has farther to travel. In other words, the higher learning rate aids the .4 input in the learning competition. Figure 13 shows the weight trajectories for the data in Table 9.

Reference = 1		Lrate=.8	Lrate=1.8	Lrate=2.8	
time	input	weight1	weight2	weight3	wfinal
107	0.4	3.3180	3.3566	3.3684	2.5
108	0.1	3.2394	3.1099	2.9794	10
109	0.2	3.2800	3.2339	3.1759	5
110	0.3	3.3213	3.3611	3.3802	3.3333

Table 9 Training Data for Ramp Input with Different Learning Rates

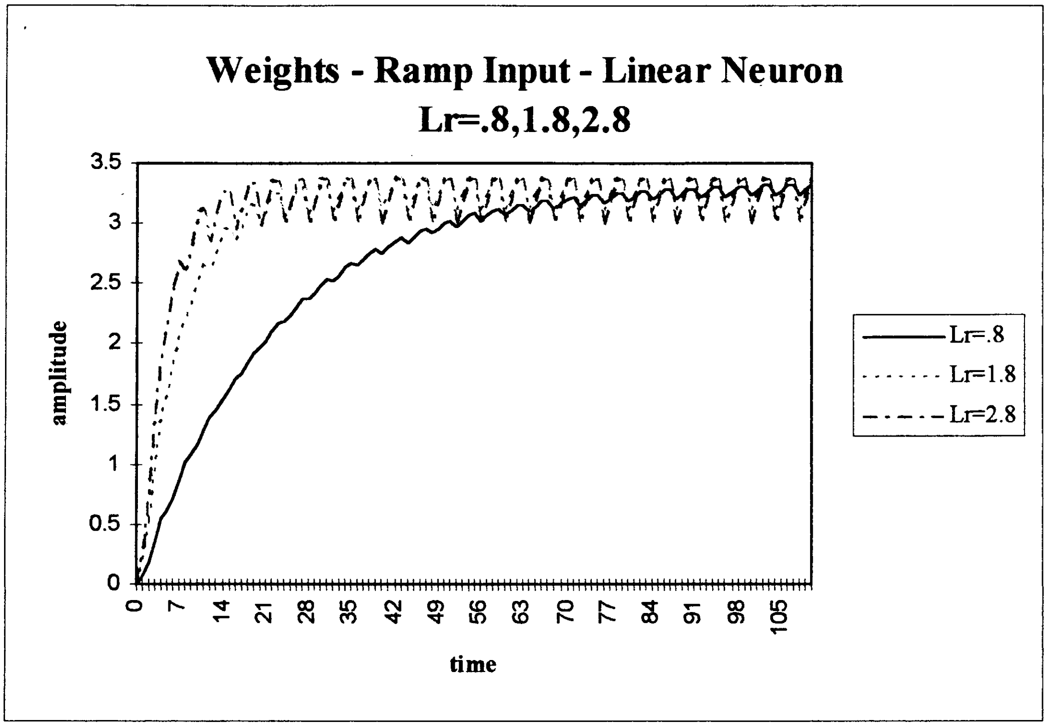


Figure 13 Ramp Input with Different Learning Rates. The learning rate affects the final values in the limit cycle.

Table 10, below, shows the effects of the initial condition on the final values of the weights for the same ramp input. Evidently, the final value is insensitive to the weight's initial condition. This is expected form the dynamic nature of the system.

		Lrate=1.8			
Reference = 1		winit.=3.3333	winit=4	winit=7	
time	input	weight1	weight2	weight3	wfinal
107	0.4	3.3566	3.3566	3.3566	2.5
108	0.1	3.1099	3.1099	3.1099	10
109	0.2	3.2339	3.2339	3.2339	5
110	0.3	3.3611	3.3611	3.3611	3.3333

Table 10 Training Data - Ramp Input Various Initial Conditions The final value of the weights is insensitive to initial conditions imposed on the weights.

The time constant does not depend upon the initial condition of the weight. Therefore, the energy of the weight at each training presentation varies is constant proportion to the other trajectories within the training epoch. For example, the .4 input in the table above corresponds to a certain mass and dissipation. Note also that the generalized force, the reference value, is constant over all the input presentations. Since the ratio of energies is the same regardless of the starting position, the weights will come to the same value *for a particular learning rate and order of training presentation.*

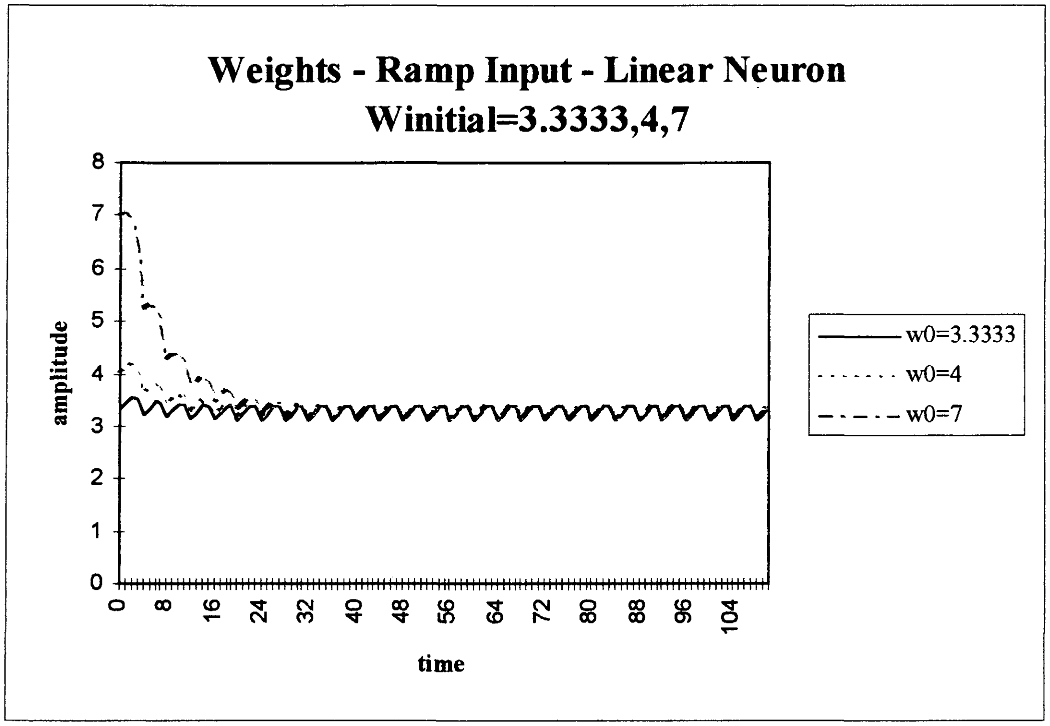


Figure 14 Ramp Input with Various Initial Conditions The steady state value is independent of the initial conditions on the weight.

If the learning rate changes, the ratio of energies will change, since the time constant is not a linear function of learning rate and input. If the order of presentation changes, the ratio of energies of the weights will change relative to each other and the energy for the entire epoch will change. The weights will attain a different "compromise". Table 11 shows this relationship.

Reference = 1		Lrate=.8	Lrate=1.8	Lrate=2.8	
time	input	weight1	weight2	weight3	wfinal
107	0.4	3.3180	3.3566	3.3684	2.5
108	0.1	3.2394	3.1099	2.9794	10
109	0.2	3.2800	3.2339	3.1759	5
110	0.3	3.3213	3.3611	3.3802	3.3333

Table 11 Ramp Inputs Various Learning Rates

Since the concept of input to a neural system as components of mass and dissipation is not intuitive, thinking of a block which suddenly and repetitively changes mass and coefficients of friction while being pushed along a surface may be helpful. If the order of the mass/dissipation change is constant, the block will always reach the same velocity after some period of time, independent of the starting velocity of the block. If the order of the mass/ dissipation changes, the block with reach a different final velocity. Of course, neither the mass nor the dissipation may be zero. In terms of equations of motion., the final weight value in Equation (46) does not depend on initial conditions of the weight.

The predicted value for the weight at the end of an infinite number of epochs for the ramp input pattern is given by Equation (58). Note that this equation is only valid for

$$(\eta o_i^2) \in (-1,1)$$

So the infinite epoch prediction can only be carried out for the (η=.8) line in Table 11 above. The weight value for the .1 input after an infinite number of training epochs is calculated to be 3.2398. The tabular value is 3.2394. There is either some error or the weights have not quite converged to the final values after 27 epochs. After Training for 36 epochs, the calculated and actual weight values match. To further examine the effect of presentation order on the final weight value, a training run was carried out with a .1, .3, .4, .2 presentation sequence. Clearly, the order of presentation does affect the final weight values, as is the hypothesis. Table 12 shows these data.

Reference = 1		Lrate=.8	Lrate=1.8	Lrate=2.8	
time	input	weight1	weight2	weight3	wfinal
107	0.2	3.2405	3.1183	2.9894	5
108	0.1	3.2968	3.2538	3.2146	10
109	0.3	3.3505	3.3753	3.4046	3.3
110	0.4	3.3492	3.3685	3.3866	2.5

Table 12 Weight Values for .1,.3,.4,.2 Presentation

4.3.3 LTV System - Time Varying Offset

While the sigmoid does not have a time-varying offset, in the interest of rigorous step by step verification, weight values for a time varying offset are calculated and compared with the actual system. Figure 15, below, shows how the offset, the value of w_{final}., and the weight trajectory interact. The specific training parameters for this run are η=.8 and o_i=.3. The offset is ramped by the function .5+.001t, where t is the iteration number. Some additional noisy values are included to test the accuracy of the predicted values. Again, the prediction is found to be exact.

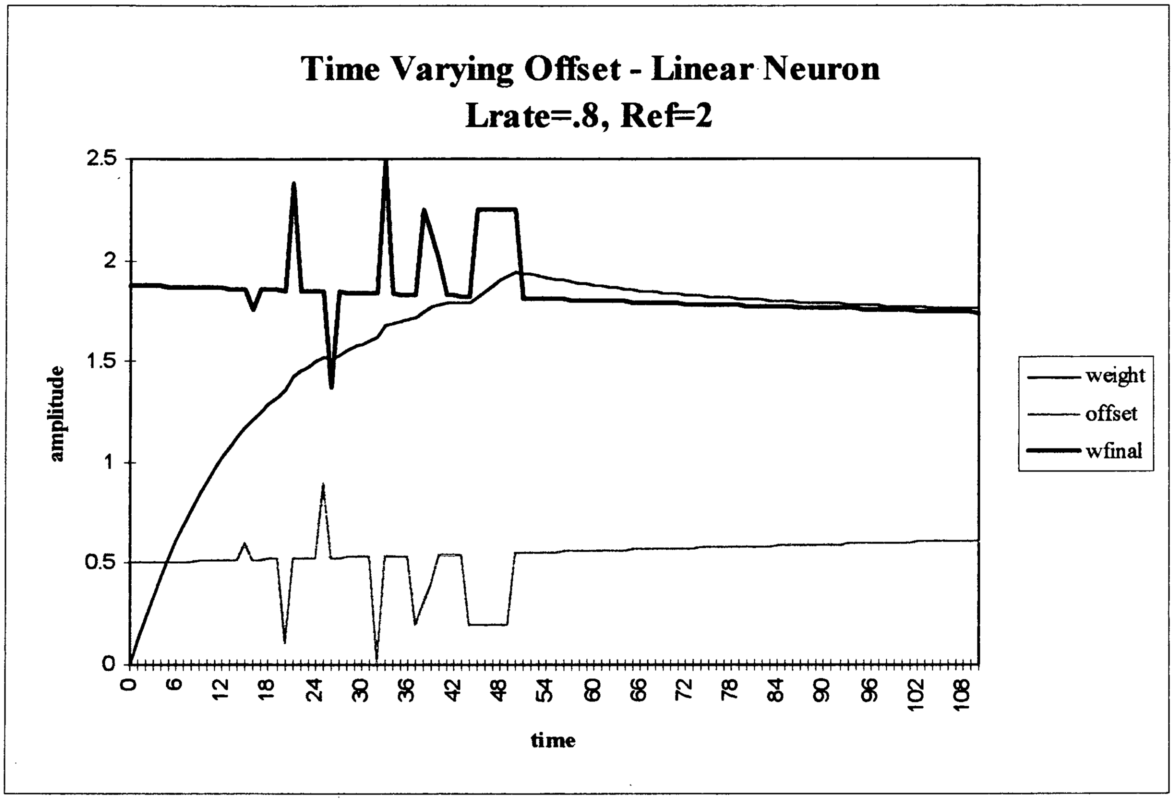


Figure 15 LTV Neural System Time Varying Offset. The offset is increased as a function of time in addition to some purely noisy values. w_{final} varies inversely with the offset.

4.4 Complete Model

At this point all the adjustments to the original LTI model are complete. The following three graphs depict 1) time varying input 2) time varying input and time varying reference 3) time varying input, reference and σ'. The σ' value is analogous to the constant which multiples the error value in the back propagation. While this neuron does not have a sigmoid activation function, the sigma constant has the same role as in the back-propagation algorithm. In essence, there is no difference between the back propagation algorithm and the training under investigation. The following figure depicts the weight trajectory of the system with varying inputs.

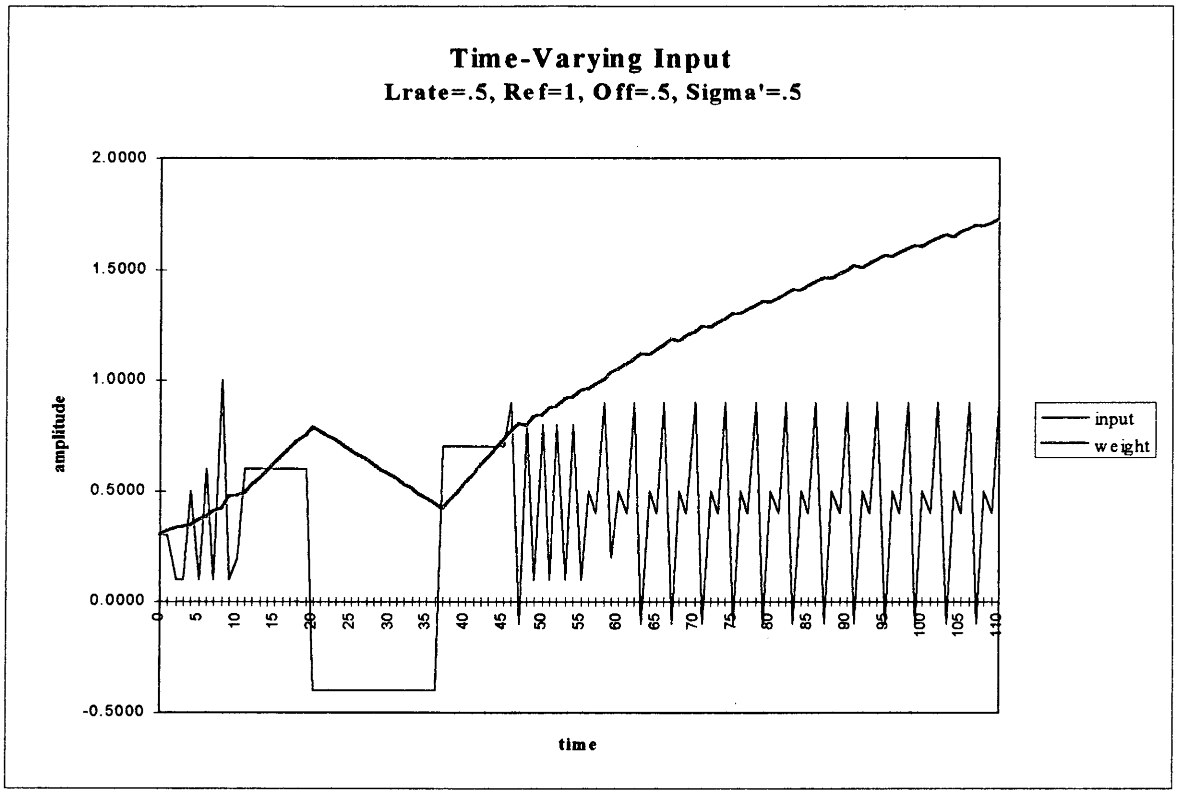


Figure 16 A SISO Neural System with Varying Inputs. The offset, reference and the σ' value are held constant for this training run. The input varies as shown.

The next figure depicts a time varying reference and the corresponding weight trajectory. Even though they are not shown, the input values are the same as shown in Figure 16.

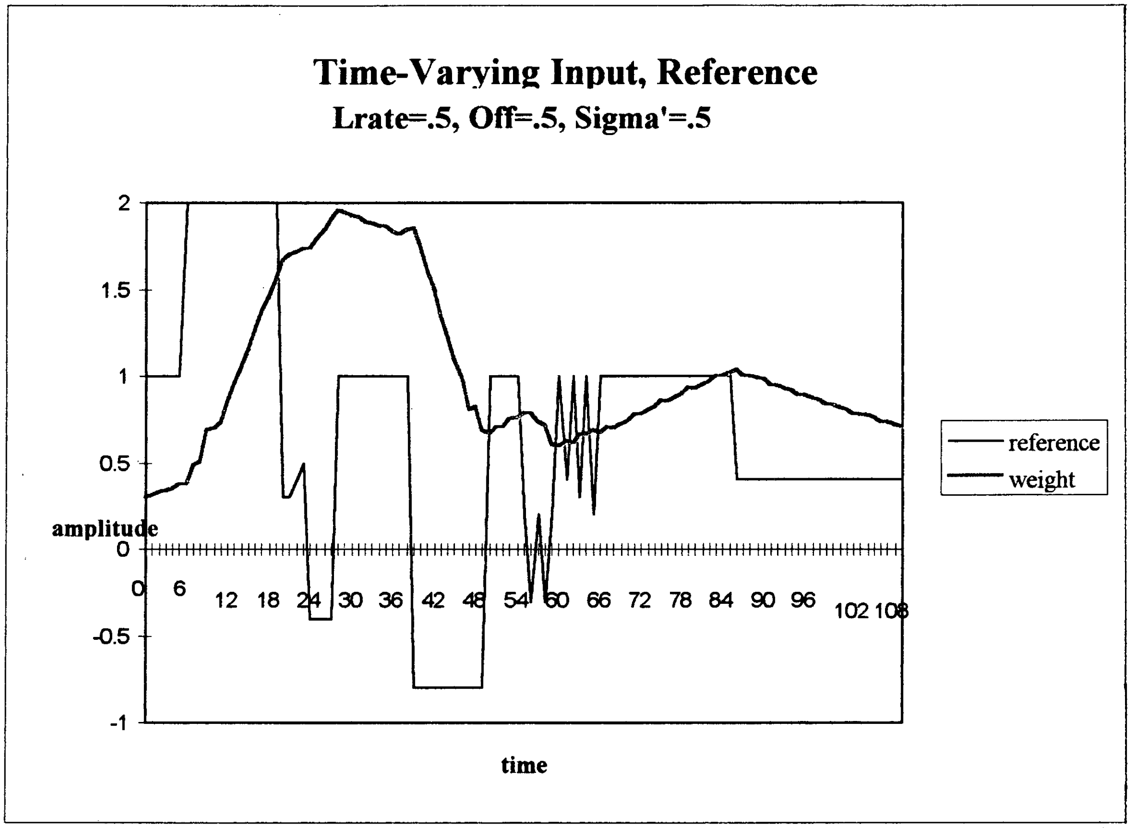


Figure 17 A SISO Neural System with Varying Inputs and Reference. The offset, reference and the σ′ value are held constant for this training run. The reference varies as shown. The input is the same as depicted in Figure 16.

Figure 18 shows the variance in the σ′ value. Both the input and reference are as shown in Figures 16 and 17 respectively. The learning algorithm is the same as the back-propagation algorithm except that the input, reference and σ′ are random.

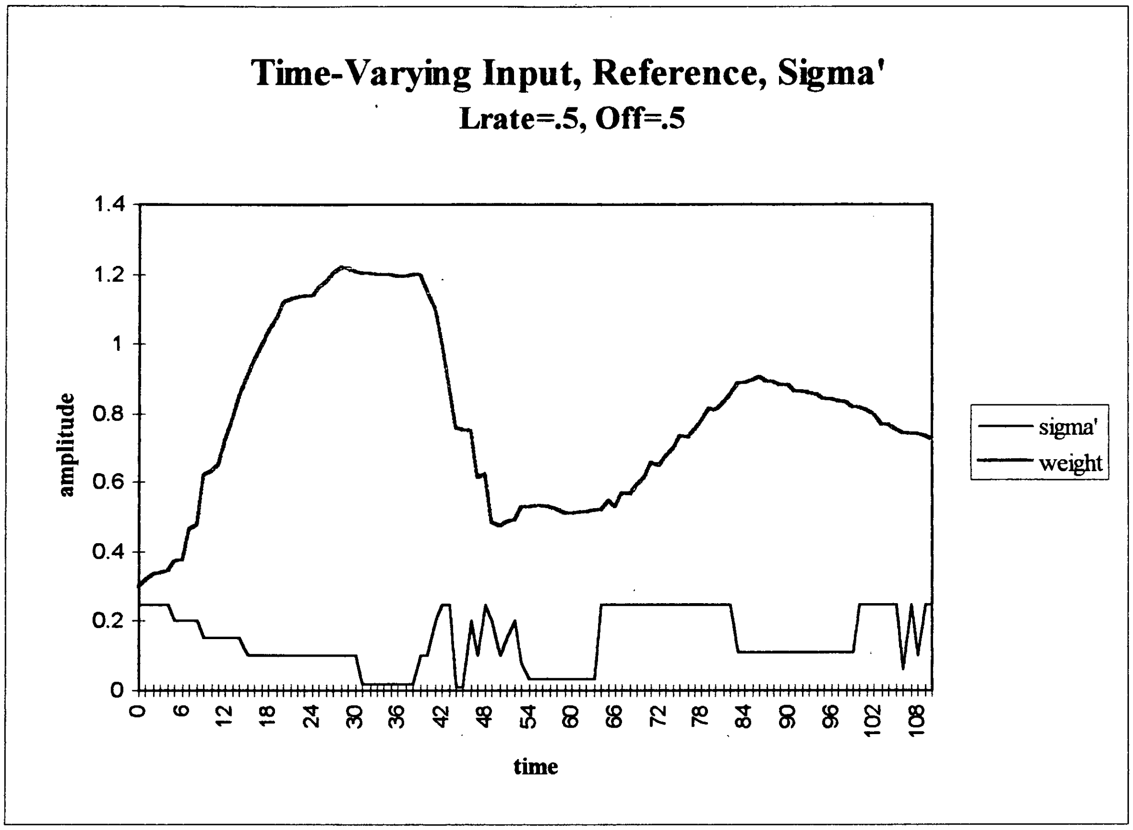


Figure 18 Varying Sigma. The input and reference are also varying as shown in Figures 16 and 17.

In all these cases, the difference between the actual weight and predicted weight is the range termed "exact" at the beginning of the results section. However, the roundoff error does warrant some mention at this point. The weight prediction Equation (53) is repeated below.

$$w_n = w_{\text{final}\,n}\left(1 - e^{-\frac{1}{\tau_n}}\right) + w_{n-1}e^{-\frac{1}{\tau_n}}$$

The weight value at the end of the present update cycle depends on the previous weight. Since there is some roundoff error in the computer, the error becomes compounded as time progresses. In this simulation, the compounded error is still not significant, but these training runs a very short compared to what would be involved in an actual problem. If training were allowed to proceed for several thousand iterations, the error might become significant. At present this is an open problem. In the interest of quantifying the error in the present 110 presentation training run, Figure 19 is a plot of two error measures. One trace is the error if the prediction equations are completely separate from the actual training; the weights at the previous time step are mathematical predictions only. The next trace is the error if the weight at the previous time step, the right hand term in Equation (53), is supplied from the synapse as it trains.

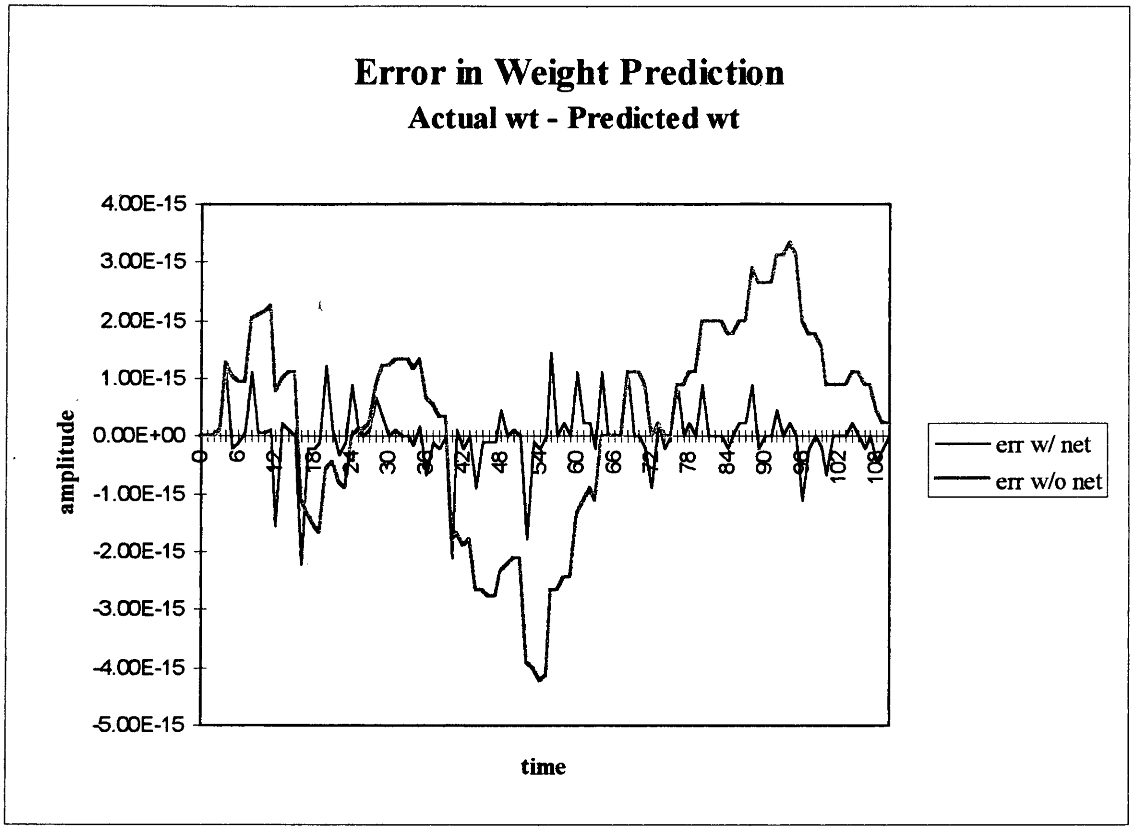


Figure 19 Errors in Weight Values. The weight value from the previous time step is used to predict the weight at the next time step. The figure shows errors if purely mathematical prediction is performed and if weights actual weights are used to help predict weights at the next time step.

4.5 NLTV System - Simulation of Training Using a Non-Linear Neuron

Figure 20 is the input and weight values for 110 iterations of a training run with the non-linear neuron with a sigmoid activation function. There are 4 input output pairs as shown in Table 13. Table 13 also shows the other relevant network parameters including the error in the prediction. Again it is within the magnitude of computer roundoff error. The weight prediction is purely mathematical; the prediction algorithm does not make use of any actual weight values, as is the case for one of the error trajectories in the previous example. The values for the derivative of the sigmoid are supplied from the training data of the actual system. The learning rate for this training is .5.

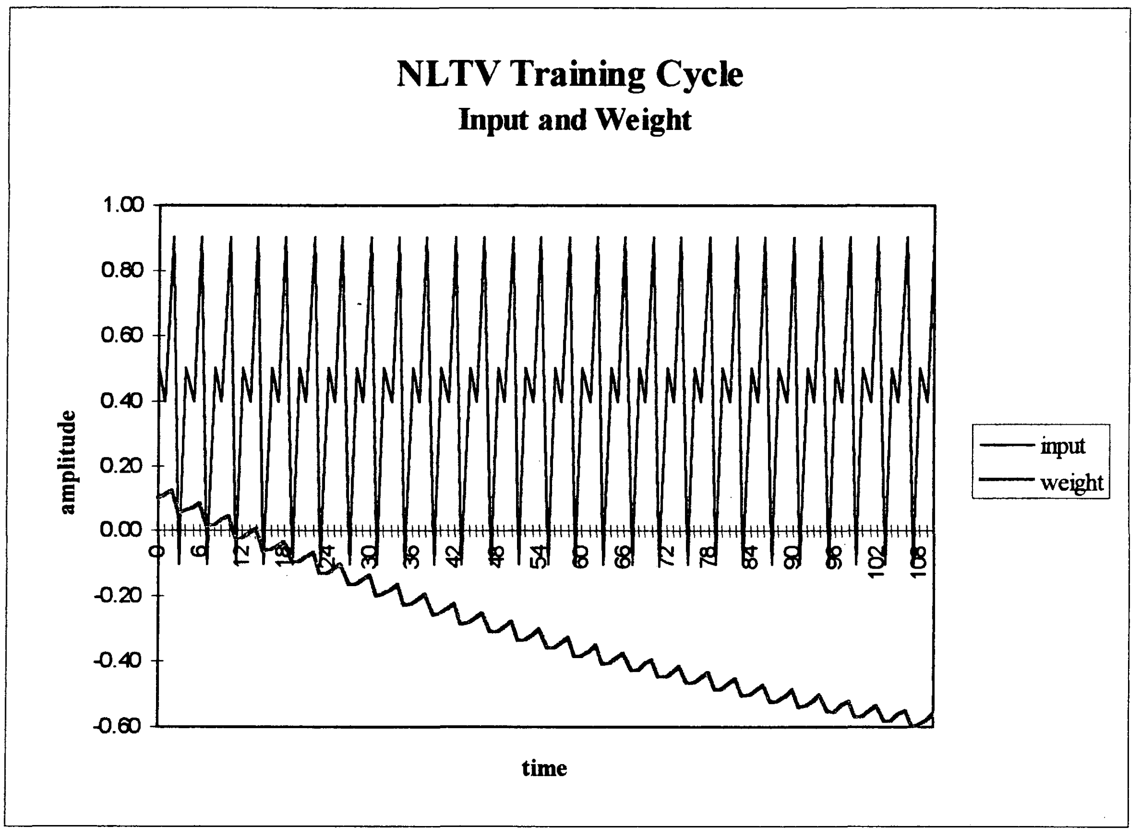


Figure 20 Input and Weight Trajectories - Non-linear Neuron

time	Input	Weight	Reference	sigma'	Error
0	0.5	0.1000	0.7	0.2498	0
1	0.4	0.1117	0.8	0.2499	-3E-17
2	0.9	0.1261	-0.1	0.2492	8.3E-17
3	-0.1	0.0557	0.3	0.2500	9.7E-17
4	0.5	0.0582	0.7	0.2499	9.7E-17
5	0.4	0.0702	0.8	0.2500	-3E-17
6	0.9	0.0849	-0.1	0.2496	1.4E-17
7	-0.1	0.0153	0.3	0.2500	-1E-16
8	0.5	0.0178	0.7	0.2500	-7E-18
9	0.4	0.0302	0.8	0.2500	7.3E-17
10	0.9	0.0450	-0.1	0.2499	-1E-16
11	-0.1	-0.0236	0.3	0.2500	7.6E-17
12	0.5	-0.0211	0.7	0.2500	-3E-16
13	0.4	-0.0084	0.8	0.2500	5.4E-17
14	0.9	0.0066	-0.1	0.2500	-8E-17
15	-0.1	-0.0610	0.3	0.2500	4.2E-17
16	0.5	-0.0585	0.7	0.2499	-4E-16
17	0.4	-0.0456	0.8	0.2500	2.1E-17
18	0.9	-0.0303	-0.1	0.2500	-2E-16
19	-0.1	-0.0971	0.3	0.2500	9.7E-17
20	0.5	-0.0945	0.7	0.2499	-2E-16
21	0.4	-0.0813	0.8	0.2499	-7E-17
22	0.9	-0.0659	-0.1	0.2498	1.4E-16
23	-0.1	-0.1317	0.3	0.2500	1.1E-16
24	0.5	-0.1291	0.7	0.2497	3.9E-16
25	0.4	-0.1156	0.8	0.2499	2.8E-17
26	0.9	-0.1001	-0.1	0.2495	-2E-16
27	-0.1	-0.1649	0.3	0.2500	1.7E-16
28	0.5	-0.1624	0.7	0.2496	3.1E-16
29	0.4	-0.1486	0.8	0.2498	8.3E-17
30	0.9	-0.1329	-0.1	0.2491	2.8E-17

Table 13 Training Data from the NLTV Neural System

CHAPTER 5

CONCLUSIONS

In this thesis, the author demonstrates that the learning dynamics of a single neuron and synapse trained using the back propagation algorithm are analogous to the dynamic behavior of physical systems such as those involving mechanical translation, rotation or LRC circuitry. By determination of dynamic parameters such as the generalized coordinate, velocity, mass, momentum, dissipation, time constant, and force of a simple neural system, an input and an output node connected by a single synapse, the author generates accurate mathematical models of several permutations of the neural system mentioned above. Specifically, the analysis proceeds from a linear time invariant (LTI) model, a single neuron and synapse trained with constant input and reference values, to a linear time varying (LTV) model of the same system excited by varying inputs and reference values. Time, in the context of training, is the number of training presentations, since the weights are updated at each computational time step. The analysis culminates with a non-linear, time varying (NLTV) model of the neural system with a sigmoid activation function. The energy functions demonstrated here are deterministic, as contrasted with stochastic or thermodynamic models used by some others (Kanter 1992;

Hertz, Krogh and Palmer 1991; Xu 1990; Cohen and Grossberg 1983). The Lagrangian and Hamiltonian dynamic formulations are also deterministic. However, the Lagrangian and Hamiltonian equations of motion are not valid for more than one time step due to the discrete nature of a neural network and the strict continuity requirements imposed on the Lagrangian and Hamiltonian energy functions. Even though the equations of motion cannot be integrated through time, the fact that they are accurate for one time step makes this approach particularly illuminating in terms of developing an intuitive understanding of neural dynamics.

5.1 Further Work

Since the neural system examined in this thesis is particularly simple and not realistic is terms of an actual training paradigm, this work could naturally be extended to multiple neuron networks and actual training data. Since this work quantifies the behavior of complex dynamic interactions in a relatively simple system, it is conceivable that this work could be extended to complete determination of the dynamic interactions to the extent that training time could be greatly reduced or perhaps eliminated. The determination of the dynamic parameters for a larger system is a prelude to neural network design from a deterministic standpoint, much as mechanical and electrical systems are presently designed. Of course, what sort of constraints or design parameters the dynamic parameters imply is the subject of further investigation.

REFERENCES CITED

Baker, Walter and Farrell, J. A. 1990. Connectionist Learning Systems for Control. *SPIE Intelligent Robotics and Computer Vision IX: Neural, Biological and 3D Methods* 1382: 181-188.

Cohen, M. A. and Grossberg, S. 1983. Absolute Stability of Global Pattern Formulation and Parallel Memory Storage by Competitive Neural Networks. *IEEE Transactions on Systems, Man and Cybernetics* 13: 815-826.

De Wilde, Phillippe. 1993. Class of Hamiltonian Neural Networks. *Physical Review E* 47 (2) : 1392-1396.

Dietterich, T., and Shavlik, J., 1990. Readings in Machine Learning. San Mateo, Ca.: Morgan Kauffman Publishers, Inc.

Hertz, J., Krough, A., and Palmer, R. 1991. Introduction to the Theory of Neural Computation. Redwood City, Ca.: Addison-Wesley Publishing Company.

Kanter, Ido. 1992. Hidden Information in Hamiltonian Systems with Discrete Weights. *Physical Review A* 45 (8) : 6051-6055.

Kilmister, C. W. 1964. Hamiltonian Dynamics. Mathematical Physics Series New York: Wiley Publishers

LaSalle, J. P. and Artstien, Z. 1976. The Stability of Dynamical Systems. *Regional Conference Series in Applied Mathematics*. Brown University

Ogata, K. 1978. System Dynamics. Englewood Cliffs, NJ : Prentice-Hall, Inc.

Petrov, I. 1968. Variational Methods in Optimum Control Theory. New York: McGraw-Hill, Inc.

Ramacher, Ulrich 1993. Hamiltonian Dynamics of Neural Networks. *Neural Networks* 6 : 547-557

Ramacher, U. and Nachbar, P. 1992. The Hamiltonian Approach to Neural Network Dynamics. *IEEE Transactions on Neural Networks* --- : 1930-1936.

Rumelhart, D., Hinton, G., and Williams, R. 1986. Learning Representations by Back-Propagating Errors. *Nature* 323 : 533-536.

Sandler, Yu 1991. Neural Networks with Color Neurons and Hidden Units: Hamiltonian and General Properties. *Physics Letters A* 157 (2,3): 119-123.

Sarti, A., Walsh, G. and Sastry, S. 1993. Steering Left-Invariant Control Systems on Matrix Lie Groups. *1993 Proceedings of the 32nd IEEE Conference on Decision Control*: 3117-3121.

Schultz, D. and Melsa, J. 1967. State Functions and Linear Control Systems. New York : Academic Press.

Sejnowski, T. and Rosenberg, C. 1987. Parallel Networks which Learn to Pronounce English Text. *Complex Systems* 1 : 145-168.

Ter Haar, D. 1961. Elements of Hamiltonian Dynamics. Amsterdam : North-Holland Publishing Company.

Wylie, C. Ray and Barrett, Louis. 1982. Advanced Engineering Mathematics. New York: McGraw-Hill, Inc.

Xu, Xiaoshu 1990. Phd. Thesis: Acceleration of the Learning of Artificial Neural Networks Utilizing a Thermodynamic and Kinetic Model with Application in Sensor Data Processing. **Colorado School of Mines** : (T-3997)

APPENDIX A-1

The following is the Lagrangian formulation for an inverted pendulum on a cart.

<u>Cart Kinetic Energy:</u>

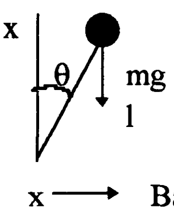
$$\left(\frac{1}{2}\right)Mv_c^{\,2} = \left(\frac{1}{2}\right)M\dot{x}_c^{\,2} \qquad \text{(M = Mass of cart)}$$

<u>Pendulum Kinetic Energy:</u>

$$\left(\frac{1}{2}\right)mv_p^{\,2} \qquad \text{(m = mass of pendulum)}$$

<u>Pendulum</u>

$$\mathrm{x}_{\mathrm{pend.}} = \mathrm{l}\sin\theta + \mathrm{x}$$



Base of pendulum is moving:

$$\dot{x}_{pend} = l\cos\theta\dot{\theta} + \dot{x}$$

$$y_{pend} = l\cos\theta$$

$$\dot{y}_{pend} = -l\sin\theta\dot{\theta}$$

Velocity of Pendulum:

$$v_p^{\,2} = \left(\dot{x}_p^{\,2} + \dot{y}_p^{\,2}\right) = \left(\dot{x} + l\cos\theta\dot{\theta}\right)^2 + \left(-l\sin\theta\dot{\theta}\right)^2$$

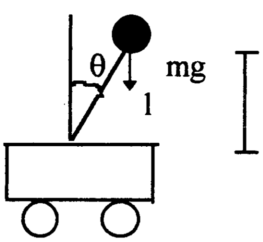
$$v_p^{\,2} = \dot{x}_p^{\,2} + l^2\cos^2\theta\dot{\theta}^2 + 2\dot{x}l\cos\theta\dot{\theta} + l^2\sin^2\theta\dot{\theta}^2 \qquad (\sin^2\theta + \cos^2\theta = 1)$$

$$v_p^{\,2} = \dot{x}_p^{\,2} + l^2\dot{\theta}^2 + 2\dot{x}l\cos\theta\dot{\theta}$$

Kinetic Energy of Pendulum:

$$\boxed{\frac{1}{2}m\left[\dot{x}^2 + l^2\dot{\theta}^2 + 2\dot{x}l\cos\theta\dot{\theta}\right]}$$

Potential Energy of Pendulum:



V = mgh

Note: h = lcosθ

$$\boxed{mgl\cos\theta = V_p}$$

Lagrangian is:

L = T - V

$$\boxed{L = \frac{1}{2}M_c\dot{x}_c^{\,2} + \frac{1}{2}m\left[\dot{x}_c^{\,2} + l^2\dot{\theta}^2 + 2\dot{x}l\cos\theta\dot{\theta}\right] - mgl\cos\theta}$$

For the x degree of freedom:

$$F_x = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = \frac{d}{dt}\left(M_c\dot{x} + m\dot{x} + ml\cos\theta\dot{\theta}\right) - 0$$
$$= M_c\ddot{x} + m\ddot{x} + ml\left[\ddot{\theta}\cos\theta - \sin\theta\dot{\theta}\dot{\theta}\right]$$
$$= M_c\ddot{x} + m\ddot{x} + ml\left[\ddot{\theta}\cos\theta - \sin\theta\dot{\theta}^2\right]$$

$$\boxed{0 = (M+m)\ddot{x} + ml\cos\theta\ddot{\theta} - ml\sin\theta\dot{\theta}^2}$$

For the θ degree of freedom:

$$F_\theta = 0 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta}$$
$$= \frac{d}{dt}\left(ml^2\dot{\theta} + m\dot{x}l\cos\theta\right) + m\dot{x}l\sin\theta\dot{\theta} + mgl\sin\theta$$

$$0 = ml^2\ddot{\theta} + ml\ddot{x}\cos\theta - ml\dot{x}\sin\theta\dot{\theta} + m\dot{x}l\sin\theta\dot{\theta} + mgl\sin\theta$$
$$0 = \left[l\ddot{\theta} + \ddot{x}\cos\theta + g\sin\theta\right]ml$$

$$\boxed{= \ddot{\theta} + \frac{\ddot{x}}{l}\cos\theta + \frac{g}{l}\sin\theta}$$