A MODEL FOR THE MANAGEMENT OF NEURODEGENERATIVE DISEASES: THE CASE OF HUNTINGTON'S DISEASE

By

AJIBOLA, OLAWALE OLANIYI EMMANUEL

B.Sc (Maths), M.Sc (Engr. Analysis)

A THESIS SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES, UNIVERSITY OF LAGOS, LAGOS NIGERIA

FOR THE AWARD OF Ph.D IN SYSTEMS ENGINEERING

AUGUST 2008

SCHOOL OF POSTGRADUATE STUDIES
UNIVERSITY OF LAGOS

CERTIFICATION

This is to certify that the Thesis:

"A MODEL FOR THE MANAGEMENT OF NEURODEGENERATIVE DISEASES: THE CASE OF HUNTINGTON'S DISEASE"

Submitted to the
School of Postgraduate Studies
University of Lagos

For the award of the degree of
DOCTOR OF PHILOSOPHY (Ph. D)
is a record of original research carried out

By

AJIBOLA OLAWALE OLANIYI EMMANUEL
in the Department of Systems Engineering

AJIBOLA, O.O.E.		27-11-2008
AUTHOR'S NAME	**SIGNATURE**	**DATE**
Professor O. Ibidapo-Obe		27-11-2008
1ST SUPERVISOR'S NAME	**SIGNATURE**	**DATE**
Prof V.O.S. Olunloyo		27-11-2008
2ND SUPERVISOR'S NAME	**SIGNATURE**	**DATE**
Prof A.B. Sofoluwe	AB Sofoluwe.	27/11/2008
1ST INTERNAL EXAMINER	**SIGNATURE**	**DATE**
Dr. T.A. Fashanu		27/11/2008
2ND INTERNAL EXAMINER	**SIGNATURE**	**DATE**
Prof James Katende Prof M.A. Adewumi		27-11-2008 27-11-2008
EXTERNAL EXAMINER	**SIGNATURE**	**DATE**
Dr. S.R. Akinola		27/11/2008
SPGS REPRESENTATIVE	**SIGNATURE**	**DATE**

DECLARATION

I declare that this thesis is a record of the research work carried out by me. I also certify that neither this nor the original work contained therein has been accepted in any previous application for a degree.

All sources of information are specifically acknowledged by means of references.

.... 27-11-2008

Ajibola, O.O.E Date

DEDICATION

This thesis is dedicated to Jesus Christ, The Lion of the Tribe of Judah

ACKNOWLEDGEMENTS

In the most humble being of myself I want to express my sincere gratitude to my Supervisors: Professor O. Ibidapo-Obe, and the Distinguished Professor V.O.S. Olunloyo, for their constructive criticism, direction, guidance and patience while this work lingered. I also wish to express my sincere appreciation to Professor G.O.S. Ekhaguere whose contributions to my growth from my undergraduate days at the University of Ibadan has being my driving force.

I appreciate God in my brother, Pastor E.K.O. Otegbeye and her amiable wife, Pastor (Mrs.) G.O. Otegbeye. I also thank God for His anointing upon my brother, Pastor Joel Ajikobi. Special thanks go to my Systems Engineering family. To my students and others too numerous to mention, I thank you all.

The writing of this thesis has provided another opportunity for me to say a BIG thank you to my mother: Mrs. Dorcas Iyabode Ajibola for bringing me into this world and nurturing me into maturity in the best way possible, and to my late father: Pa. Joseph Oyekunle Ajibola whose guiding philosophy has projected me into the nonce and most certainty, the futurity. To my sister, Mrs. Oyewumi Felicia Olafimihan, I appreciate your most overwhelming contributions towards my academic pursuit. Of notable mention is the emotional, physical and spiritual support of God's gift to me, my wife, Princess Sade Oluwakemi Ajibola. I sincerely appreciate you.

Above all I give all the glory to Jehovah-Adonai who in the beauty of His holiness made this work possible.

AJIBOLA, Olawale Olaniyi Emmanuel
Lagos, Nigeria

TABLE OF CONTENTS

CHAPTER 3: METHODOLOGY

CHAPTER 4: RESULTS AND DISCUSSION

CHAPTER 5: CONCLUSION

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF NOTATIONS

Notations	
T	Absolute Temperature
$\frac{d^2x}{dt^2}$	Acceleration in Horizontal Axis of the Cartesian Plane
$\frac{d^2y}{dt^2}$	Acceleration in Vertical Axis of the Cartesian Plane
$\underline{a}$	Acceleration Vector
ω	Angular Velocity
η	Arbitrary real number
N_A	Avogadro's constant
k	Boltzmann' constant
ΔH	Change in Enthalpy
ΔG	Change in Gibbs Free Energy
C	Concentration of the Neurotransmitter
ΔS	Change in the State of the Neurotransmitter
μ_C	Chemical potential
$\boldsymbol{F}$	Collection of Open Sets
C_i	Concentration of Neurotransmitter Inside the Neuron
C_0	Concentration of Neurotransmitter Outside the Neuron
c_1	Concentration of Neurotransmitter at State 1
c_2	Concentration of Neurotransmitter at State 2
ψ	Constant of Proportionality
n	Charge on the ion in question

$\delta(\ )$	Delta Function
$s(t)$	Distance at Time t
$\zeta(\ )$	Dummy function to simplify an equation
E	Electromotive Force
H	Enthalpy
F	Faraday Constant
$\frac{d\underline{a}}{dt}$	First Derivative of Acceleration
$\frac{d\underline{x}}{dt}$	First Derivative of Displacement
$\frac{d\underline{v}}{dt}$	First Derivative of Velocity
$\forall$	For All Elements of a Set
R	Gas Constant
G	Gibbs Free Energy
C	Hand Acceleration in the Minimum Jerk Model of Flash and Hogan
Δt	Infinitesimal Change in Time
E_0	Initial Electromotive Force before Jerk
$\frac{d^3x}{dt^3}$	Jerk in Horizontal Axis of the Cartesian Plane
$\frac{d^3x}{dt^3}$	Jerk in Vertical Axis of the Cartesian Plane
J	Jerk Scalar
$\underline{j}$	Jerk Vector
α	Learning constant of the Artificial Neural network (ANN)

$\frac{d^2s}{dt^2}$	Magnitude of Acceleration
$\frac{d^3s}{dt^3}$	Magnitude of Jerk
$\frac{ds}{dt}$	Magnitude of Velocity
m	Mass of mechanical Structure
μ	Mean value of a distribution
F^*	Mechanical Force
$\lvert f(\ )\rvert$	Modulus of function f
β	Momentum parameter for the ANN
ε	Negligible Real Number
$\left\Vert \frac{d^3s}{dt^3} \right\Vert$	Norm of jerk
$\boldsymbol{G}$	Open Covering of a Set
$\boldsymbol{I}$	Open Interval
$\underline{x}$	Position Vector
C	Quantity of Electricity
$f(x)$	Real Valued Function of variable x
$\in$	Set Inclusion
σ	Standard deviation
S	State of Neurotransmitter
S_1	State 1 of Neurotransmitter
S_2	State 2 of Neurotransmitter

∍	Such That
t_f	Time needed to reach the final position in a Jerk
t_k	Time taken for *k-th* jerk to take place
Ω	Number of states available to a molecule
$\frac{dx}{dt}$	Velocity in Horizontal Axis of the Cartesian Plane
$\frac{dy}{dt}$	Velocity in Vertical Axis of the Cartesian Plane
$\underline{v}$	Velocity Vector
V	Volume of the Neurotransmitter
$\alpha(\ )$	Weighted Function

ABSTRACT

Mutant huntingtin gene can bind to transcription factors, resulting in reduced levels of acetylated histones. One consequence of this appears to be a decreased expression of genes which may play critical roles in neuronal survival. Early motor signs of Huntington's disease (HD) typically include the gradual onset of clumsiness, balance difficulties, and brief, random, fidgeting movement. Many HD patients develop a distinctive manner of walking (gait) that may be unsteady, disjointed, or lurching. In this work an attempt has been made to promote a better understanding of the physiological chorea associated with the human health hazard of HD origin by proposing a crisp model which describes the arm gait of a Huntington's disease patient. However, the mathematical solution proffered to the proposed model by Frobenius method, failed to capture the staccato nature of the jerk for which the model has been proposed. To this end, we have carried out an artificial neural network (ANN) simulation of the arm gait of a HD patient, based on the same set of data as the crisp model. We therefore carried-out an expository analysis of the arm gait of HD patients using ANN techniques upon which we have designed our proposed model for the management of neurodegenerative diseases based on the physiological presentation (i.e. chorea) of Huntington's disease. It is believed that this work will form a basis for biomedical engineering device for the management of chorea in HD.

Keywords: chorea, choreiform, choreathosis, GABAergic, dopaminergic, mutant alleles, polyglutamine, autosomal,

CHAPTER ONE

INTRODUCTION

1.0 Background to the Study

Huntington's disease (HD), earlier known as Chorea or Hereditary Chorea, due to writhe, twist, constant uncontrollable dance-like motion of various parts of the body of the affected person, is a progressive neural disorder that causes untold suffering for thousands of families. The history of HD dated back to at least the middle ages, Lanska, D.J. (2000). The name Huntington's disease was coined out of the name of an American physician George Huntington who wrote about the illness way back in 1872, describing it as "an heirloom from generations away back in the dim past", Huntington, G. (1872). HD results from genetically programmed degeneration of nerve cells, called *neurons*, in certain areas of the brain, Kopp, P. et al (1998). This degeneration causes, Guyton, A.C. et al (1996):

- Uncontrolled movements (physiological)
- Loss of intellectual faculties (psychiatric), and
- Emotional disturbance (psychological).

Specifically affected are cells of the *basal ganglia*, structures deep within the brain that perform many important functions, including coordinating movement. Within the basal ganglia, HD especially targets neurons of the *striatum*, particularly those in the *caudate nuclei* and the *pallidum*. Also affected is the *cortex* (the brain's outer surface) which controls thought, perception, and memory. HD is found in every country of the world. It every country of the world. It is a familiar disease, passed from parent to child through a *mutation* or misspelling in the normal *gene*, Shi-Hua, L. et al (2004).

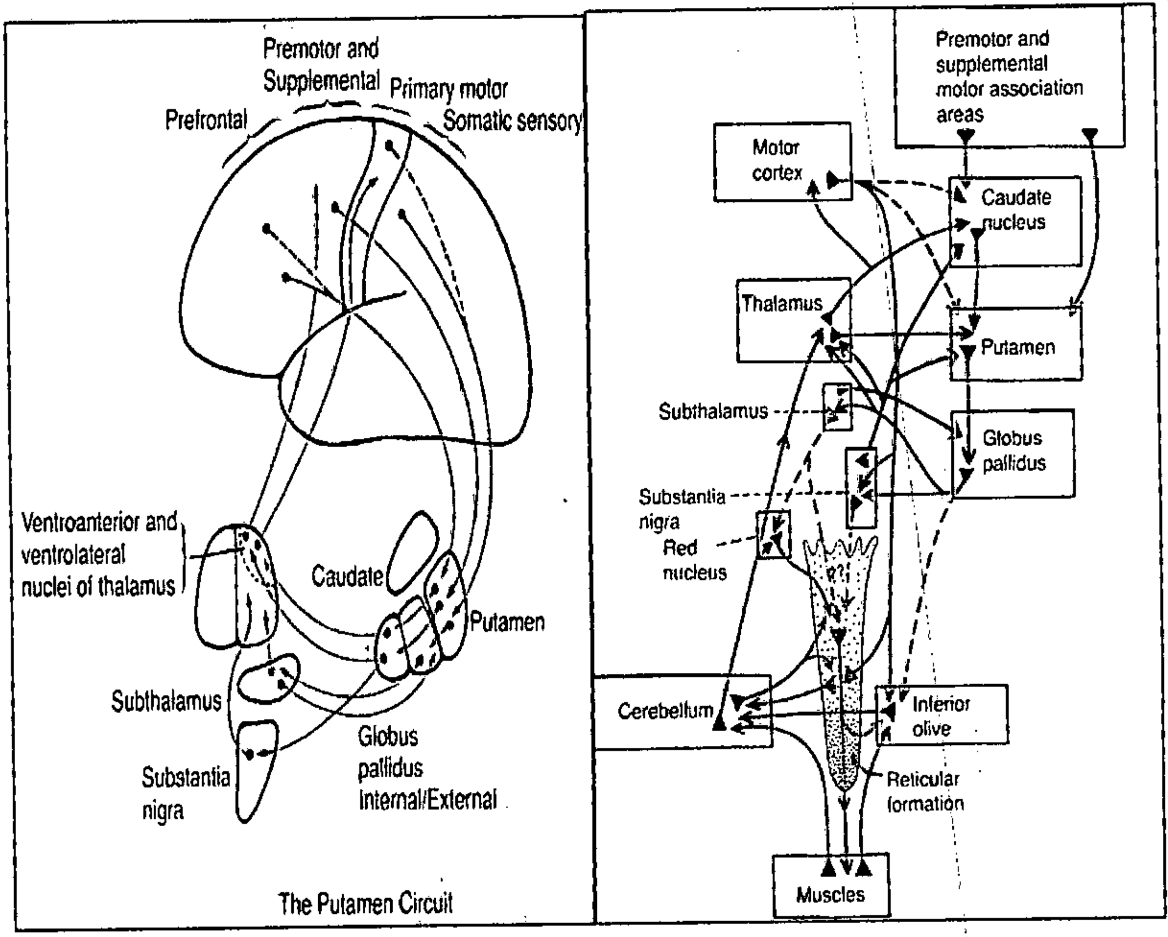

Figure 1: Putamen circuit for subconscious execution of learned pattern of movement (LEFT); and the relation of the basal ganglia circuitry to the corticospinal cerebellar system for movement control (RIGHT)

A single abnormal gene, the basic biological units of heredity, produces HD. Genes are composed of *deoxyribonucleic acid (DNA),* a molecule shaped like a spiral ladder. Each rung of this ladder is composed of two paired chemicals called *bases*. There are four types of bases; adenine, thymine, cytosine, and guanine, each abbreviated by the first letter of its name: A, T, C, and G. Certain bases always "pair" together, and different combinations of base pairs join to form coded messages. A gene is a long string of this DNA in various combinations of A, T, C, and G. These unique combinations determine the gene's function, much like letters join together to form words. Each person has about 30,000 genes (a billion base pairs of DNA or bits of information repeated in the nuclei of human cells, which determine individual characteristics or *traits).*

Genes are arranged in precise locations along 23 rod-like pairs of *chromosomes*. One chromosome from each pair comes from an individual's mother, the other from the father. Each half of a chromosome pair is similar to the other, except for one pair, which determines the sex of the individual. This pair has two X chromosomes in females and one X and one Y chromosome in males. The gene that produces HD lies on chromosome 4, one of the 22 non-sex-linked, or "autosomal" pairs of chromosomes, placing men and women at equal risk of acquiring the disease. The impact of a gene depends partly on whether it is *dominant* or *recessive*. If a gene is dominant, then only one of the paired chromosomes is required to produce its called-for effect. If the gene is recessive, both parents must provide chromosomal copies for the trait to be present. HD is called an *autosomal dominant disorder* because only one copy of the defective gene, inherited from one parent, is sufficient to produce the disease.

The genetic defect responsible for HD is a small sequence of DNA on chromosome 4p16.3 in which several base pairs are repeated many, many times. The normal gene has three DNA bases, composed of the sequence CAG. In people with HD, the sequence abnormally repeats itself dozens of times. Over time and with each successive generation, the number of CAG repeats may expand further.

Each parent has two copies of every chromosome but gives only one copy to each child. Each child of an HD parent has a 50-50 chance of inheriting the HD gene. If a child does not inherit the HD gene, he or she will not develop the disease and cannot pass it to subsequent generations. A person who inherits the HD gene, and survives long enough, will sooner or later develop the disease. In some families, all the children may inherit the HD gene; in others, none does. Whether one child inherits the gene has no bearing on whether others will or will not share the same fate. However, a small number of cases of HD are sporadic, that is, they occur even though there is no family history of the disorder. These cases are thought to be caused by a new genetic mutation-an alteration in the gene that occurs during sperm development and that brings the number of CAG repeats into the range that causes disease.

Of interest to us is the modeling of the management of choreiform movement in HD. The question of control of the ailment does not arise since the etiology of the disease is not properly understood now. To achieve the stated aim, we have arranged our work in the following order. The first chapter introduces the subject matter. Chapter two gives various definitions, and a detailed survey of previous work. The third chapter discusses the problem formulation, assumptions and proffered solutions. The chapter also outlines how HD affects the neural motor circuitry system in the body and the existing medical practices as regards the treatment of HD with drugs. A short preview of an existing practice in medicine involving applications of electroconvulsive devices in Parkinson's diseases; and the mathematical considerations for computer simulation for the arm gait in HD are also contained in chapter three. Chapter four contains summary of results. The conclusions, contributions to knowledge and the recommendations for further research are itemized in the fifth and indeed the last chapter.

1.1 Statement of the Problem

A concise statement of the problem requires the identification of possible factors that are responsible for the chorea in the arm of Huntington's disease patient and the development of an adequate model for the arm gait which can serve as basis for a Neuro-therapy. Such a model can be either crisp or one based on Artificial Neural Network (ANN) simulation.

Neurodegenerative diseases are ailments which have devastating effect on both the patients and the caretakers alike. The first intervention in mild chorea is discontinuous use of drugs that have potential to exacerbate symptom of HD. Treatment therefore begins with a trial and error session whereby drugs are administered and the patient is examined to determine if the drugs will exacerbate hyperactivity in the patient. If the drugs aggravate hyperactivity in the patient they are stopped promptly. However, if they don't, other after-effects are tested for before such drugs are recommended for the patient's usage. Major setback to this practice includes the possibility of introducing variety of latent devastating conditions with symptoms that may remain concealed until much later and at considerable risk to the patient. Moreover, present clinical practice where drugs are administered on patients as a panacea to the disease is cumbersome because, in addition to their devastating side effect, such drugs also degrade the ability of patients to function. This study is therefore aimed at developing a mathematical model that will promote a better understanding of the physiological features of the disease in order to provide a framework upon which a viable Neurotherapy for the management of such ailment could be built.

1.2 Objectives of the Study

The objectives of this study are:

1. To formulate a representative model of the writhe, twist, uncontrollable dance-like motion of the arm of neuropath suffering from the Huntington's disease (HD).
2. To provide a mathematical solution to the proposed model and relate to the arm gait of such a neuropath.
3. To establish the basis for a control mechanism by illustrating our research findings with an interactive computer simulation based on Artificial Neural Networks.

1.3 Scope and Limitations of the Study

Merriam Webster (1996) defines disease as an impairment of the normal state of the living animal or plant body or one of its parts that interrupts or modifies the performance of the vital functions and is a response to environmental factors (as malnutrition, industrial hazards or climate), to specific infective agents (as worms, bacteria, or virus), to inherent defects of the organisms (as genetic anomalies), or to combination of these factors. Several diseases are curable by drugs but in some cases such drugs have been more of hazard than of help to the patients. Even when they cure the disease they are invariably administered in overdose thereby degrading the ability of the patient to function. However, electroconvulsive therapy only administers adequate volume of the drugs required to curb the ailment at programmed intervals. In view of the foregoing, modeling for management of neuro-degenerative disease and the result of this study should no doubt have wide applications in neurology and various fields of medicine where drug therapy can produce devastating side effects on the patients. It will also find wide applications in any inter-modal system structures involving transmission of fluid.

1.4 Significance of the Study

Research efforts are ongoing on the etiology of Huntington's disease. Control of the ailment will be a mirage without an indebt understanding of the etiology of the disease. In other to assist the sufferers live a normal life, research efforts have been geared towards the management of the various conditions associated with HD. One of such conditions is the choreiform movements that constitute a major physiological symptom of the disease. However, interventions through drug administration have proved to be dismal failures. Nonetheless, Several research efforts have been made in the area of application of electroconvulsive therapy to manage chorea associated with patients of Huntington's disease. In this regards, Metrode incorporation, U.K, has manufactured a deep brain stimulation nanorobot of the size of the head of a push pin which can be introduced into human system with the sole aim of managing the physiological considerations as may, for example be prevalent in a Parkinson's disease patient. It should as a management strategy be possible for an electrode to be implanted in a patient's brain, and made capable of sending out electrical impulses at programmed interval(s) with the sole aim of neutralizing the excitatory post synaptic potential that aggravates the HD condition. It is imperative however to note here that there have been major achievements in area of application of deep brain technology in the management of similar conditions such as treatment of depression, tremors, slowness of movement and rigidity to mention but a few. A major achievement of Metrode incorporation's device is the ability of the electrode to deliver electrical stimulation to the brain which essentially rewires the brain by restoring contacts between neurons.

There are various techniques used in solving real life problems in engineering and sciences. Some of these techniques include empirical, analytical and artificial intelligence methods. Empirical method involves performance of experiments, collection of data arising from the experiments, data analysis in that order to mention but a few; but in most cases empirical analysis often lead to modeling, and analytical procedures are used to determine the future behavioral pattern of such physical systems. On the other hand, analytical method requires the identification of useful parameters each of which represents an attribute of the physical system under consideration. A close study of combinations of these attributes results in a governing rule called an equation (or a model) of the physical system which constitutes a class of problems. In artificial intelligence (AI), efforts are geared towards the simulation of the behaviour of the physical system by generating a set of data that may be used to train the AI system to emulate the activities of the physical system.

In this study, we have employed both the analytical method involving modeling, and the artificial intelligence method involving simulation of an artificial neural network (ANN). The analytical method gave birth to a crisp model which defines both the choreoathetosis and the choreiform associated with HD. However, the solution gave a smooth parabolic curve which is in total agreement with the experimental evidence of Flash and Hogan (1985) and the Nernst equation for the state of equilibrium of arm at resting potential as contained in the work of Susan Greenfield (1999). Even though the crisp model agrees with the pioneering work of Flash and Hogan its solution fails to capture the staccato nature of the jerky motion exhibited by the arm of HD patients. It is clear from our analysis that the ANN model did not fail in this respect. These two methods have been carefully analyzed in chapter three.

This particular study is deemed necessary to aid the development of fully intelligent system that is aimed at:

- Providing a rational basis for the design of an artificial intelligent mechanism that ultimately controls the unwanted motion.
- Providing a more efficient basis for determining level of physiological information (signals) needed to control hyperkinetic reactions in HD, and
- Providing a more efficient and targeted management of similar ailments such as Parkinson's disease, Alzheimer's disease, disease of the cerebellum, epilepsy and contractions during childbirth.

Science has now reached a stage in the field of nanomedicine where in the nearest future nanorobots can be introduced into the body system of a patient which will release drug at programmed intervals to curb an ailment. The purpose of our study therefore is to propose a platform for this mechanism. The world is getting more sophisticated each passing moment with the development of varieties of self driven control devices which may be applied in the field of medicine to tackle diverse motor neural disorders which makes the research both justifiable and desirable. The practical significant of the results for an Huntington's disease patient is that, since the unwanted motion (in form of periodic jerk) experienced in the arm of the HD patient constitutes a noise to all the activities of the patient, imposing a control on the vibration will bring about a relieve to the patient. The essence of this work is to provide a basis for the design of an artificial intelligence device which ultimately controls the unwanted movement.

1.5 Research Questions

In an attempt to channel our research effort toward a rewarding goal, we shall endeavour to provide adequate answers to the following research questions; viz:

- Which engineering tools can we employ for modeling the arm gait of Huntington's disease patient?
- What method of solution can we employ to solve the model above?
- Which of the engineering tools employed for modeling the arm gait of Huntington's disease is most adequate?

1.6 Operational Definition of Terms

The following table contains the definitions of the frequently used terms in the body of the thesis.

Acceleration	The rate of change of velocity with time
Jerk	The rate of change of acceleration with time

Velocity	The rate of chance of displacement with time
Electromotive Force	Maximum electrical energy generated by a chemical process
Mechanical Force	The force that moves a particle from one point to another
Absolute Temperature	Temperature measured on Kelvin scale
Neurotransmitter	A substance that transmits nerve impulses across a synapse
Real Valued Function	A function which domain is a subset of the real number
Gibbs Free Energy	The energy that drives a chemical process
Charge on the ion	Valency of molecule in solution or molten state
Mean value	Average of numerical values
Chemical potential	Electromotive force
Gait	A manner of moving resemblance of how a horse moves
Dysphagia	Difficulty in swallowing
Kinematics	A branch of physics that deals with aspects of motion apart from consideration of mass and force
ANN	Artificial neural network
Artificial	Humanly contrived often on a natural model
Neural cell	One of the cells that constitute the nervous tissue
Dominant trait	A dominant genetic factor
Recessive trait	A character that produces little or no phenomenon
DNA	Any of various nucleic acids that are usually the molecular basis of heredity
Crisp	Based on traditional mathematical formulation
Neuropath	An individual subject to nervous disorders
Trinucleotide	Codon
Glutamine	Crystalline amino acid that is found both free and in protein

	in animals
Electroconvulsive	Relating to a convulsive response to a shock from electricity
Disorder	An abnormal physical or mental condition
Neurology	The scientific study of the nervous system
Acetylcholine	A neurotransmitter that is active in the transmission of nerve impulse
Electrochemical	Involving relationship of electricity to chemical changes and interconversion of chemical and electrical energy
Fluid element	Neurotransmitter
Parabolic smooth curve	A curve of a shape of parabola which does not have discontinuity at any point
Axon	The pathway of neurotransmitter from the nerve cell body to the synapse of a post synaptic neuron
Codon	A specific sequence of three consecutive nucleotides that is part of the genetic code and that specifies a particular amino acid in a protein or starts or stops protein synthesis
Gene	The functional unit of inheritance controlling the transmission and expression of one or more traits.
Simulation	The deliberate making of a certain condition that could exist in reality
IPSP	Inhibitory post synaptic potential (neurotransmitter that inhibits motion
Hyperkinetic	Uncontrollable muscular movements
Model	An abstraction of real life situation
GABA	Gamma-aminobutyric acid
Pathogenesis	The origination and development of a disease
Chorea	Spasmodic movement of the limbs and facial muscles and by incoordination

Choreiform	Chorea
Neurodegenerative	Gradual degrading of the brain tissues
Mutation	A relatively permanent change in hereditary material
Huntingtin	A protein that assists in body building
Autosomal	Of body
Athetosis	A nervous disorder that is marked by continual slow movement
EPSP	Excitatory post synaptic potential (neurotransmitter) such as Acetylcholine, is a neurotransmitter which excites postsynaptic neurons

CHAPTER TWO

REVIEW OF LITERATURE

2.1 Preamble

For clarity we have organized this chapter in three sections. Section 2.1 contains a systematic review of works on the pathogenesis of Huntington's disease and relevant hypotheses and theories that promotes the understanding of HD. In section 2.2 we have considered the review of related works that culminated in our formulation of the classical model of jerk motion exacerbated by the excitatory post-synaptic potential in neuronal circuit of the arm of HD patient while section 2.3 delved into the artificial neural networks analytic approach to the jerky motion of arm of HD sufferers.

2.2 The Huntington's Disease

In a publication of NINDS (2000) titled, Huntington's disease: Hope through research, the origin of expository study of the disease was systematically outlined. Margolis and Ross (2003) defined Huntington's disease as a rare, progressive, and fatal autosomal neurodegenerative disorder, typically of adult onset, that has captured the imagination of the scientific and medical community far in excess of its direct impact on public health. Guyton and Hall (1996) defined Huntington's disease as a hereditary disorder that usually begins to cause symptoms in the fourth or fifth decade of life. According to Guyton and Hall, HD is characterized at first by flicking movements at individual joints and then progressive severe distortional movements of the entire body. In their own work, Shi-Hua and Xiao-Jiang (2004) simply described the disease as the most common genetic disease that is caused by an expansion of a polyglutamine (polyQ) tract in the associated disease protein. Features common to these definitions revealed the fact that HD is a common hereditary neurodegenerative disorder that has no regards for gender or race. Huntington's disease (HD) was coined out of the name of an American physician George Huntington who wrote about the illness way back in 1872, Lanska, D.J. (2000). It is a complex neural disorder that causes untold suffering for thousands of families, Kopp, P. and Jameson, J.L. (1998). HD results from genetically programmed degeneration of nerve cells in certain areas of the brain. The genetic defect responsible for HD is a small sequence of DNA on chromosome 4 in which several base pairs are repeated many, many times, Gardian, G. et al (2004); Peterson, S.P. (2006); Squitierri, F. et al (2003); Tassicer, R. et al (2003).

2.2.1 The Pathogenesis of Huntington's Disease

Marcolis, R.L. et al (2001); Shi-Hua, L. et al (2004) proclaimed that Huntington's disease was the first gene mapped to a chromosomal locus by use of anonymous markers that provided the molecular tools for predictive genetic testing by linkage analysis. The molecular basis of heredity is one of the various nuclei acids called the DNA and it is also responsible for the storage of any information required for structural formation in the body, Luthi-Carter, R. (2000). A normal gene is a specific sequence of nucleotides in DNA that is located in the germ plasma on a chromosome and it has three DNA bases, composed of the sequence of CAG called alleles, Frazin, N. et al (2004); Guesella, J.F. et al (1983). A sequence of three consecutive nucleotides that is part of the genetic code that also specifies a particular amino acid in a protein, or, start or stops protein synthesis is called a codon (or a triplet), Ji-Yeon, S. et al (2005); Margolis, R.L. et al (2003).

Discovered in 1994, the function of HD protein huntingtin (htt) is still not fully understood. Human htt is a large protein containing 3144 amino acids, Tassicker, R. et al (2003). The polyglutamine (polyQ) domain, which begins at the 18th amino acid position usually contains 11-34 glutamine residue in unaffected individuals and expands to more than 37 glutamines in HD patients Kopp, P. et al (1998), Shi-Hua, L. et al (2004); Squitierri, F. et al (2003). In people with HD, mutation occurs causing the sequence of the three DNA bases CAG to repeats itself abnormally dozens of times (in biology and medical sciences, mutation refers to changes in the genetic materials). Over time and with each successive generation, the number of CAG repeats may expand further. HD is neither racial nor gender biased since it originates from the short arm of chromosome 4, a non-sex chromosome, each child of an HD parent has a 50-50 chance of inheriting the HD gene. However, if a child does not inherit the HD gene, he or she will not develop the disease and cannot pass it to subsequent generations. A person who inherits the HD gene, and survives long enough, will sooner or later develop the disease. In some families, all the children may inherit the HD gene; in others, none do. Whether one child inherits the gene has no bearing on whether others will or will not share the same fate. Nevertheless, a small number of cases of HD are sporadic, that is, they occur even though there is no family history of the disorder. These cases are thought to be caused by a new genetic mutation (i.e. an alteration in the gene that occurs during sperm development and that brings the number of CAG repeats into the range that causes disease).

2.2.2 Pathology of Huntington's Disease

According to literature, gross pathology of HD is limited to the brain, George Huntington (1872), Jen-Zen, C. et al (2002); Ross, C.A. et al (1997); resulting in atrophy of the caudate, putamen, and cerebral cortex. The abnormal movements in Huntington's disease are believed to be caused by loss of most of the cell bodies of the GABA-secreting neurons in the caudate nucleus and putamen and acetylcholine-secreting neurons in many parts of the brain. The axon terminals of the GABA neurons normally cause inhibition in the globus pallidus and substantia nigra. This loss of inhibition is believed to allow spontaneous outbursts globus pallidus and substantia nigra activity that cause the distortional movements.

2.2.3 Diagnosis of Huntington's Disease

Diagnosis of HD is based on a thorough personal and family medical history, physical examination (which may include neurological examination), and a series of laboratory tests. To aid diagnosis, the physician may require that patients supply comprehensive information regarding recent changes in patients' recent intellectual and/or emotional function, which constitute signs of Huntington's disease, Margolis, R.L. et al (2003).

At the onset, HD manifests with motor symptoms, and most times it is the initial complaint of clumsiness with attendant tremor, balance trouble or jerkiness that makes patients to seek medical attention. The earliest symptoms include chorea or choreoathetosis, continuous and irregular writhing and jerking movement, most prominently of the limbs and the trunk, Ross, C.A. et al (1997). Other symptoms may include respiratory, oral and nasal musculature to mention but a few. However, as the disease progresses, the dementia becomes more global. It is worthy of mention that as many as 80% of HD patients develop some form of non-cognitive psychiatric disorder within 10 – 15 years of the onset of the disease, George Huntington, (2003). Personality changes often manifest as irritability and apathy in HD patient. But the psychological manifestations of HD patients are often responsive to treatment.

2.3 Modeling

A model is an abstraction of real life action. It is a simple description of a system, used for explaining, calculating, projecting, planning or evaluating the system, Hoppensteadt, F.C. et al (2002). Chi-Tsong, C. (1984) discussed the technicalities involved in solving real-life problems using analytical methods. He highlighted for important components of analytical methods to include; modeling, development of mathematical equation description, analysis, and design. He further claimed that the distinction between physical systems and models are basic in engineering. Modeling is a very important problem since the success of a design depends upon whether the physical system is properly modeled or not. However, to develop a suitable model of a physical system, a thorough understanding of the physical system and its behavior is essential, Nakano, E. et al (1999). It should be noted here that every mathematical equation representing the behavioral pattern of a physical system is also covered by our definition of a model and generally in literature, models are used synonymously as mathematical equations. In furtherance of analyses, another type of model is also considered in this study. This is the artificial neural networks (ANN) model, which is of great importance to engineering analysis where simulation of a physical system is involved, in an area of study called Artificial Intelligence, Williamson, M.M. et al (1998); Xu, Z-B et al (2004); Zhao, H. (2004); Zhou, J. et al (2004).

Physical systems may be studied by empirical methods whereby various signals are applied to the physical system under consideration and its responses are measured, Zehr, E.P. et al (2003). Based on experience, if the performance is not satisfactory, we adjust some of its parameters or connect to it some compensator to improve its performance. It is an established fact that this approach to problem solving has undoubtedly succeeded in designing many useful physical systems. More than often, the odds against empirical methods may be too overwhelming thus giving analytical methods an edge over the empirical techniques. For instance, empirical methods may become unsatisfactory if

- The specification on the physical systems become very precise and stringent
- The physical systems become very complicated
- The physical systems become too expensive
- The physical systems become too dangerous

to be experimented, to mention but a few. In such cases, analytical methods take precedence over the empirical considerations.

2.3.1 Modeling in Huntington's Disease

In the above analysis, we have categorized the symptoms of Huntington's disease under the following broad classes:

- Physiological
- Psychological and
- Psychiatric

To model therefore, we considered the three broad classes above, and using their features, we concluded that it is the physiological features that can be modeled. The reason for our decision is not far fetched. The attraction to these features is the choreiform movements. The rapid jerky motion of the entire body (the arm inclusive) of the patients of Huntington's disease (HD) is an outward symptom of HD exhibited by persons who suffer from the disease. The basis of our analysis is the theory of motor control.

2.3.2 Motor Control

Brown, L.E. et al (2001) described motor control as the ability of biological and artificial systems to plan, initiate, maintain, monitor and correct movements to attain physically realizable goals. The nature of the action to be carried out is usually not fully determined by specification of goal alone. One challenge for motor control research is to explain how one movement is chosen from the plethora that is possible; since there may be an infinite number of movements that lead to the achievement of the set goal. This leads to the problem of redundancy which is present at all levels of the motor control system. The redundancy problem can be approached by considering:

- Body positions while ignoring the forces behind them
- Body positions with the forces involved.

We have chosen the former in our study.

Other challenges in motor control include:

- The learning problem as regards how to model relations between movements and their effects.
- Perceptual motor integration which involves the understanding of how feedback is used to correct errors, and also how feed forward is used to prevent errors.

CHAPTER THREE

METHODOLOGY

3.0 Preamble

This chapter contains an expository consideration of fundamental principles and theories upon which the analytical build-up of models and their solutions contained therein are based. In order to present a detailed analysis of the problem, we have divided the chapter into three broad sections. Section A is a clear exposition of the basic physiological considerations of the arm gait in Huntington's disease (HD) patients. It also contains techniques for problem formulation and crisp modeling. Section B is based on the seemingly short-coming of section A. It discusses the Artificial Neural Network (ANN) modeling of the arm gait in Huntington's disease patient as a viable representation of the condition by back propagation using the Sigmund function, while Section C, and the last section of the chapter deals with the modeling of the management of the aforementioned physiological condition in HD and other related conditions in patients suffering from other Neurodegenerative disease with similar traits.

A. PROBLEM FORMULATION AND MODELLING

Towards the end of eighteenth century, Luigi Galvani, a professor of Anatomy at Bologna University in Italy, published a book describing how a freshly dissected frog's leg could be thrown into muscular convulsions, simply by connecting the foot and the exposed nerves through a length of copper and iron wire. The recorded muscular convulsion is as a result of movements of ions in the chemical materials (i.e.

neurotransmitters) in the nerve cells of the fresh frog's leg in accordance with the polarity of the ions, with the ions being attracted to various metals in accordance with their positions in the electrochemical series. This discovery was what led to the invention of the standard electrolytic cell, which culminated in the manufacture of a battery.

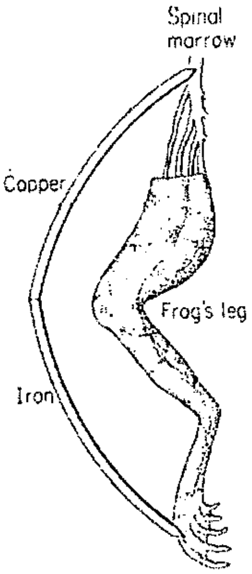

Fig. 3.1 Galvani's experiment with a frog's leg

According to elementary science, the driving force in electrolytic cell has been attributed to the potential difference between two poles of the cell as demonstrated in the Voltaic cell. Basically a cell converts chemical energy to electrical energy with the resultant effect that the chemical materials get gradually used up while the cell is in action. A cell has two unlike metal plates or poles, with chemicals between them. One pole, termed positive is at higher electrical potential than the other, called negative pole, so that when connected by a wire there is flow of current from the positive to the negative pole.

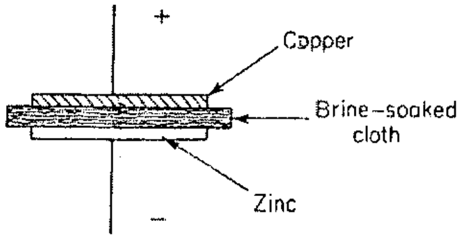

Fig.3.2 A Simple Cell

3.1 The Equation of a Human Arm

It is well known that the first derivative of **position** (symbol ***x***) with respect to time is **velocity** (symbol ***v***), while the second is **acceleration** (symbol ***a***). It is however less well known that the third derivative of position vector is known as **jerk** (symbol ***j***). Jerk is a vector but it may be used loosely as a scalar quantity because there is no separate term for the magnitude of jerk analogous to speed for velocity. Jerk is therefore the rate of change of acceleration with respect to time i.e. it is the first derivative of acceleration. Hence,

$$\underline{v} = \frac{d\underline{x}}{dt} \qquad (3.1)$$

$$\underline{a} = \frac{d\underline{v}}{dt} \qquad (3.2)$$

$$\underline{j} = \frac{d\underline{a}}{dt} \qquad (3.3)$$

Moreover, if a particle travels a distance *s(t)* as a function in circle (*s* may be thought of as the arc length of the curve traced out by the particle). The speed is given by:

$$\frac{ds}{dt} = \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2} \qquad (3.4)$$

But acceleration, the second derivative of position *s(t)* may be derived from the quotient rule as follows:

$$\frac{d^3s}{dt^3}=\frac{\left[\left(\frac{d^2x}{dt^2}\right)^2+\left(\frac{d^2y}{dt^2}\right)^2+\frac{dx}{dt}\frac{d^3x}{dt^3}+\frac{dy}{dt}\frac{d^3y}{dt^3}\right]\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]^{1/2}}{\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2}$$

$$-\frac{\left(\frac{dx}{dt}\frac{d^2x}{dt^2}+\frac{dy}{dt}\frac{d^2y}{dt^2}\right)\left(\frac{dx}{dt}\frac{d^2x}{dt^2}+\frac{dy}{dt}\frac{d^2y}{dt^2}\right)\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]^{-\frac{1}{2}}}{\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2} \qquad (3.5)$$

$\Rightarrow$

$$\frac{d^3s}{dt^3}=\frac{\left[\left(\frac{d^2x}{dt^2}\right)^2+\left(\frac{d^2y}{dt^2}\right)^2+\frac{dx}{dt}\frac{d^3x}{dt^3}+\frac{dy}{dt}\frac{d^3y}{dt^3}\right]\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]-\left(\frac{dx}{dt}\frac{d^2x}{dt^3}+\frac{dy}{dt}\frac{d^2y}{dt^3}\right)^2}{\sqrt{\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]^3}}$$

$$\frac{d^3s}{dt^3}=\frac{\left[\left(\frac{d^2x}{dt^2}\right)^2+\left(\frac{d^2y}{dt^2}\right)^2+\frac{dx}{dt}\frac{d^3x}{dt^3}+\frac{dy}{dt}\frac{d^3y}{dt^3}\right]\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]^2}{\sqrt{\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]^3}}$$

$$-\frac{\left[\left(\frac{d^2x}{dt^2}\right)^2+\left(\frac{d^2y}{dt^2}\right)^2\right]\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]^2-2\frac{dx}{dt}\frac{dy}{dt}\frac{d^2x}{dt^2}\frac{d^2y}{dt^2}+\frac{dx}{dt}\frac{d^2y}{dt^2}+\frac{dy}{dt}\frac{d^2x}{dt^2}}{\sqrt{\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]^3}}$$

$$\frac{d^3s}{dt^3}=\frac{\left[\left(\frac{d^2x}{dt^2}\right)^2+\left(\frac{d^2y}{dt^2}\right)^2+\frac{dx}{dt}\frac{d^3x}{dt^3}+\frac{dy}{dt}\frac{d^3y}{dt^3}\right]-\left[\left(\frac{d^2x}{dt^2}\right)^2+\left(\frac{d^2y}{dt^2}\right)^2\right]}{\sqrt{\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]}}$$

$$+\frac{2\frac{dx}{dt}\frac{dy}{dt}\frac{d^2x}{dt^2}\frac{d^2y}{dt^2}-\frac{dx}{dt}\frac{d^2y}{dt^2}-\frac{dy}{dt}\frac{d^2x}{dt^2}}{\sqrt{\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]^3}}$$

$$\frac{d^3s}{dt^3}=\frac{\frac{dx}{dt}\frac{d^3x}{dt^3}+\frac{dy}{dt}\frac{d^3y}{dt^3}}{\sqrt{\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]}}+\frac{2\frac{dx}{dt}\frac{dy}{dt}\frac{d^2x}{dt^2}\frac{d^2y}{dt^2}}{\sqrt{\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]^3}}-\frac{\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]\left[\left(\frac{d^2x}{dt^2}\right)^2+\left(\frac{d^2y}{dt^2}\right)^2\right]}{\sqrt{\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]^3}}$$

$$+\frac{\left(\frac{dx}{dt}\right)^2\left(\frac{d^2x}{dt^2}\right)^2+\left(\frac{dy}{dt}\right)^2\left(\frac{d^2y}{dt^2}\right)^2}{\sqrt{\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]^3}}$$

$$\frac{d^3s}{dt^3}=\frac{\frac{dx}{dt}\frac{d^3x}{dt^3}+\frac{dy}{dt}\frac{d^3y}{dt^3}}{\sqrt{\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]}}+\frac{\frac{dx}{dt}\frac{d^2x}{dt^2}+\frac{dy}{dt}\frac{d^2y}{dt^2}}{\sqrt{\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]^3}}+\frac{2\frac{dx}{dt}\frac{dy}{dt}\frac{d^2x}{dt^2}\frac{d^2y}{dt^2}}{\sqrt{\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]^3}}$$

$$-\frac{\left(\frac{d^2x}{dt^2}\right)^2+\left(\frac{d^2y}{dt^2}\right)^2}{\sqrt{\left[\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2\right]}}$$

We therefore claim that jerk is defined as:

$$\left\|\frac{d^3s}{dt^3}\right\|\approx\sqrt{\left(\frac{d^3x}{dt^3}\right)^2+\left(\frac{d^3y}{dt^3}\right)^2} \qquad (3.6)$$

Hence $$j=\sqrt{\left(\frac{d^3x}{dt^3}\right)^2+\left(\frac{d^3y}{dt^3}\right)^2} \qquad (3.7)$$

Since the jerk in the arm of a HD patient results not from an electromotive force, but from the difference in electrical potentials, then it must have resulted from the derivative of the electromotive force. Our analysis is therefore based on a function of the potential difference which results from the electromotive force produced.

Susan Greenfield, (1999) offered an explanation in respect of distribution of ions on either side of the membrane. At resting potential, the ions will flow one way or the other as a spontaneous process stemming from two forces:

- The tendency to equalize concentrations (a chemical or "diffusional" force); and
- The tendency to be attracted to an opposite, negative charge inside the neuron (an electrical force).

Because there is finite number of ions, this movement will not continue indefinitely: there is a stage when the concentration and charge of each particular ion is balanced on either side of the membrane, and as a result, no net flux of ions occurs. There would be a state of dynamic equilibrium, like two individuals of equal weight on a seesaw that was perfectly motionless. The potential difference corresponding to this equal distribution is given by the Nernst equation:

$$E = \left(\frac{RT}{nF}\right) \ln \frac{C_0}{\mathrm{C_i}} \qquad (3.8)$$

The critical parameters in determining this value are: the concentration of ions both inside C_i and outside C_o the neuron; and the absolute temperature, T. In addition, three further, non-changeable factors have to be taken into account: the charge on the ion in question, n; the Faraday constant, F, the magnitude of the charge per mole of electrons; and the universal gas constant R.

Flash and Hogan (1984) proposed a mathematical model, *the minimum jerk model*, that simulates unconstrained point-to-point movements of the arm through a third specified point. The model is based on the minimization of the rate of change of hand acceleration in a fixed Cartesian coordinate system. The integration of the square of the derivative of hand acceleration gives a function denoted by C which has to be minimized, as:

$$C = \frac{1}{2}\int_0^{t_f}\left[\left(\frac{d^3x}{dt^3}\right)^2 + \left(\frac{d^3y}{dt^3}\right)^2\right]dt \qquad (3.9)$$

where t_f is the time needed to reach the final position.

Equation (3.9) matches observed human planar two-joint arm movements and implies that trajectories are invariant under translation, rotation, time and amplitude scaling.

The basis for using optimization theory in the derivation of the mathematical model is its ability to describe an assumed goal of the class of the movements in a relatively simple formula; they thereafter derived a detailed prediction of the kinematics of a large number of specific movements from the formula. Such a mathematical model also succeeds in accounting for the majority of the kinematical features of planar horizontal arm movements. The dynamic optimization technique requires:

- The definition of a criterion function which describes the objective of the movement.
- Formulation of a set of differential equations, which describes the response of the system to its inputs.

- The application of the methods of variation calculus and optimal control theory to find the trajectory which minimizes the criterion function subject to dynamic constraints imposed by the system of differential equations and the algebraic constraints imposed at the end points, or during the motion.

3.2 The Equation of the Arm Gait of a HD Patient

As brain cells become depleted in a Huntington's disease (HD) patient, problems may develop in the following three areas:

- Motor control (movement)
- Cognition (thinking), and
- Behavior

Chorea problems arise when the centers of motor or cognitive control are affected that cause muscle weakness or discoordination in speech and swallowing, and problems with memory, sequencing, new learning ability, reasoning, and problem solving. Huntington's disease is an autosomal dominantly inherited progressive neurodegenerative disorder. The mutant gene has been localised to chromosome 4p16.3. The gene product huntingtin is widely distributed in both neurons and extra-neuronal tissues. The mutation in Huntington's disease involves the expansion of a trinucleotide (CAG) repeat encoding glutamine. The etiology of Huntington's is yet unknown but increasing evidence suggests important role of altered gene transcription, mitochondrial dysfunction and excitotoxicity. The expanded polyglutamine stretch leads to a conformational change and abnormal protein-protein interactions. Mutant huntingtin can bind to transcription factors, resulting in reduced levels of acetylated histones. One consequence of this appears to be a decreased expression of genes which may play critical roles in neuronal survival.

Early motor signs of HD typically include the gradual onset of clumsiness, balance difficulties, and brief, random, fidgeting movement. At first, chorea, - a movement disorder characterized by frequent, irregular, purposeless, jerky motions - may be incorporated into intentional actions, potentially masking symptoms and delaying recognition of the condition. Early during the course of HD, chorea may be limited to the fingers and toes. However, these movements become more noticeable over time and may extend to the arms, legs, face and trunk. Under certain circumstances, such as stress or a highly emotional state, choreic movements may become widespread or generalized. Movements essentially blend or flow into one another, causing them to appear relatively slow and writhing in nature (athetosis). In addition, involuntary movements may develop a dystonic quality in which there may be unusual twisting motions and alternating or fixed postures resulting from sustained muscle contractions.

Many HD patients develop a distinctive manner of walking (gait) that may be unsteady, disjointed, or lurching. The gait has also been described as dance-like in nature. As the disease progresses, other findings may include:

- Clumsy fine motor movements
- Postural instability
- Inability to sustain certain voluntary movements
- Poor control of tongue and diaphragm

- Difficulty swallowing (dysphagia)
- A strained, hoarse or inappropriately loud voice

Of concern in this work is the analysis and management of arm gait of HD patient. It is in particular assumed that the jerky arm movement here is a point action exacerbated by the pulse resulting from the action potential attained by the neurons in the arm. Usually, a point action is described by Dirac delta function given by:

$$\int_{a-\varepsilon}^{a+\varepsilon} \delta(x-a)dx = 1 \qquad \forall\, x \ni x \in (a,\varepsilon), 0 < \varepsilon << 1 \tag{3.10}$$

And from literature, it is possible to define the integral of the unit impulse function and any continuous and bounded function *f*, thus

$$\int_{-\infty}^{\infty} \delta(x-a)f(x)dx = \lim_{\varepsilon \to 0} \frac{1}{2\varepsilon} \int_{a-\varepsilon}^{a+\varepsilon} f(x)dx \tag{3.11}$$

By the mean value theorem of the integral calculus, there exists a real number $\eta \in [a,b]$ such that;

$$\int_{a}^{b} f(x)dx = f(\eta)(b-a) \tag{3.12}$$

Combining (3.11) with (3.12), we have that;

$$\int_{-\infty}^{\infty} \delta(x-a)f(x)dx = \lim_{\varepsilon \to 0} \frac{1}{2\varepsilon} f(\eta)(2\varepsilon) \tag{3.13}$$

Hence,

$$\int_{-\infty}^{\infty} \delta(x-a)f(x)dx = f(a) \tag{3.14}$$

However, chorea syndrome in HD is a coupled process. The coupling is between electrical and mechanical systems resulting in an electromechanical system where an electromotive force *E* drives a mechanical system with an inertia (or mass). According to the Newton's law of motion, the force F^* required by the mechanical system to achieve motion is given by:

$$F^* = ma \tag{3.15}$$

However, the force F^* is not a mechanical force but rather an electromotive force:

$$E = ma \tag{3.16}$$

Nevertheless, the motion that results is not a product of the electromotive force E but the change in potential E, thus:

$$\frac{dE}{dt} = \frac{d}{dt}(ma)$$

But m is constant for the period of motion, therefore:

$$\frac{dE}{dt} = m\frac{da}{dt} \tag{3.17}$$

Hence $$E = m\int\frac{da}{dt}\,dt \tag{3.18}$$

Adopting the dynamic optimization theory based *minimum jerk model* of Flash and Hogan as represented in equation (3.9), the chorea experienced by a HD patient in the arm from point to point could be described by the model:

$$E = \tfrac{1}{2}m\int_{\Delta t}\left[\left(\frac{d^3x}{dt^3}\right)^2 + \left(\frac{d^3y}{dt^3}\right)^2\right]dt \tag{3.19}$$

In equation (3.16), m is the mass of the mechanical structure driven by E at time t, while Δt stands for the infinitesimal time interval within which a jerk occurs in the arm of an HD patient. It is pertinent to note that the motion under consideration had been created by the difference of potential aggravated by the energy of chemical reaction which was converted into electrical energy in the neuronal circuitry system of the body. The energy that drives a chemical process is called Gibb's free energy. It will be recalled from chemical engineering processes that for a chemical process to be feasible the Gibbs free energy that drives the chemical process must be negative. It will be recalled further from literature that the Gibb's free energy is given by:

$$G = nFE \tag{3.20}$$

where E is the electromotive force due to the excitatory neurotransmitter, while n and F are as earlier defined. We now relate equation (3.8) with equation (3.20) using entropy and Gibbs free energy. To achieve this aim we first express all the quantities involved per molecule so that Boltzmann's constant k and the electron charge e are used in place of the gas constant R and the Faraday constant F. By definition, the entropy of a molecule is given by:

$$S = k \ln \Omega \tag{3.21}$$

where Ω is the number of states available to the molecule. The number of states must vary linearly with the volume V of the system, which is inversely proportional to the concentration c, so we can also write the entropy as:

$$S = k \ln (\text{constant } t \times V) = - k \ln (\text{constant } t \times c) \tag{3.22}$$

The change in entropy from some state 1 to another state 2 is therefore:

$$\Delta S = S_2 - S_1 = -k \ln c_2 - (-k \ln c_1) = -k \ln \frac{c_2}{c_1} \tag{3.23}$$

In electrochemical cell, the cell potential E is the chemical potential available from redox reaction:

$$E = \mu_c / e \tag{3.24}$$

where μ_c is the chemical potential, e is the electron charge and E is related to Gibbs free energy change ΔG only by a constant:

$$\Delta G = -neE \tag{3.25}$$

where n is the number of electrons transferred.

NOTE: there is a negative sign because a spontaneous reaction has a negative ΔG and a positive E.

The Gibbs free energy is related to the entropy by:

$$G = H - TS \tag{3.26}$$

where H is the enthalpy and T is the absolute temperature of the system.

Using these relations, we can now express change in Gibbs free energy as follows:

$$\Delta G = \Delta H - T\Delta S = \Delta G_0 - kT\ln\frac{c_2}{c_1} \tag{3.27}$$

And the cell potential becomes:

$$E = E_0 - \frac{kT}{ne}\ln\frac{c_2}{c_1} \tag{3.28}$$

To convert molar quantities, we simply multiply the Boltzmann's constant k and the electron charge e by Avogadro's number N_A, thus:

$$R = kN_A \text{ and } F = eN_A \tag{3.29}$$

Hence:

$$E = E_0 - \frac{RT}{nF}\ln\frac{c_2}{c_1} \tag{3.30}$$

And at equilibrium $E = 0$ wherefrom:

$$E_0 = \frac{RT}{nF}\ln\frac{c_2}{c_1} \tag{3.31}$$

To establish a relationship between E and the jerk j, it is important to note that the magnitude of jerk produced is a function of the electromotive force producing it in accordance with equation (3.18). However, it is essential to establish the relationship between E and the jerk produced.

According to Byrne, J.H. et al (1994), Dale and his colleagues in 1936 found that electrical stimulation of motor axons led to an increase in the concentration of acetylcholine (ACh). As a result of a nerve action potential that invades the presynaptic terminal, ACh is released into the synaptic cleft. This acetylcholine diffuses across the synaptic cleft and combines with receptors on the postjunctional membrane. The resultant increase in Na^+ and K^+ permeabilities depolarizes the postsynaptic membrane thus triggering an action potential in the muscle cell. As noted by Byrne, J.H. et al (1984), it is also possible to achieve the same result by Ca^{2+} hydrolysis as proposed by Katz and his colleagues. The action potential produced in the muscle cell ultimately leads to an impulse in the form of muscular contraction. The jerk produced depends on the electromotive force produced by the EPSP. However, change of electromotive force is a function of the magnitude of jerk produced. Choosing the jerk element as the incremental factor therefore, we can generate the governing equation for the electrical impulse E. Clearly, E is maximum at maximum impact of jerk and tends to zero as jerk varnishes, therefore given that a is the jerk element:

$$\frac{dE}{da} = \psi E \qquad (3.32))$$

We can set $\psi = 1$ without loss of generality, hence:

$$\frac{dE}{da} = E \qquad (3.33)$$

$$\Rightarrow \qquad \frac{dE}{E} = da \qquad (3.34)$$

$$\ln|E| = a(t) + C \tag{3.35}$$

$$E = e^{[a(t)+C]}$$

$$E = Ae^{a(t)} \tag{3.36}$$

We define $A = E_0 = \frac{RT}{nF}(\ln C_0 C_i)$ and a is the jerk element. Substituting A in equation (3.36) above:

$$E = \frac{RT}{nF}\ln\left(\frac{C_O}{C_i}\right)\exp[a(t)] \tag{3.37}$$

Of primary interest is the part of our analysis that generates the jerk element, we hereafter reverts to the jerk element in equation (3.9) to obtain our proposed equation, viz:

$$E = \frac{RT}{nF}\left(\ln\frac{C_o}{C_i}\right)\exp\left\{\tfrac{1}{2}\int_C\left[\left(\frac{d^3x}{dt^3}\right)^2 + \left(\frac{d^3y}{dt^3}\right)^2\right]dt\right\} \tag{3.38}$$

where all the parameters are as defined above in (3.8) capture the motion during resting potential as well as during the irregular dance-like, jerky motion of the arm of an HD patient. For instance, we recover E_0 from equation (3.38) when jerk equals zero.

To justify equation (3.38) it is important to note the following facts:

- That the exponential function for E results from equation (3.33) and as ultimately expressed in equation (3.36);
- That E is maximum translates to maximum jerk at $t = t_0$;
- That $E = E_0$ (equilibrium potential) at final time t_f when jerk is zero.

Thus, when $a(t_f) = 0$, $E = E_0$ but $E = E(t_0)$ when $a(t)$ is maximum. Now suppose:

$$a(t) = \tfrac{1}{2}\int_{\Delta t}\left[\left(\frac{d^3x}{dt^3}\right)^2 + \left(\frac{d^3y}{dt^3}\right)^2\right]dt\Bigg|_t \qquad (3.39)$$

It follows that:

$$E = \frac{RT}{nF}\ln\left(\frac{C_0}{C_i}\right)\exp[a(t)]\Big|_{t_0}^{t_f} \qquad (3.40)$$

$$\Rightarrow \qquad E = \frac{RT}{nF}\ln\left(\frac{C_0}{C_i}\right)\exp[a(t)]\Big|^{t_f} - \frac{RT}{nF}\ln\left(\frac{C_0}{C_i}\right)\exp[a(t)]\Big|^{t_0} \qquad (3.41)$$

$$E = \frac{RT}{nF}\ln\left(\frac{C_0}{C_i}\right) - \frac{RT}{nF}\ln\left(\frac{C_0}{C_i}\right)\exp[a(t)]\Big|^{t_0} \qquad (3.42)$$

Or

$$E = E_0 - \frac{RT}{nF}\ln\left(\frac{C_0}{C_i}\right)\exp[a(t_0)] \qquad (3.43)$$

Since $a(t_f) > 0$, then $\exp[a(t_f)]$ accounts for the ΔE that is responsible for the jerk. We can therefore generalize equation (3.43) as follows:

$$E = E_0 - \frac{RT}{nF}\ln\left(\frac{C_0}{C_i}\right) \qquad (3.44)$$

Notice that equation (3.44) is the general form of Nernst equation.

Georgopoulos, A.P. *et al* (1982) showed in their work that studies of two-joint arm movements revealed that the variability in hand trajectories is reduced exponentially with time as a result of practice. Since with learning and practice movements tend to be performed more smoothly and gracefully, this may indicate an underlying objective of achieving the smoothest movement which carries the hand from one equilibrium position to another, and this is what we hope to achieve at the end of this research effort. In our own case we have a recurrence of such movements, which could be assume to occur at equal interval due to the reasons adduced in our analysis above.

3.3 Solving thc Equation of the Arm Gait of HD Patient

Equation (3.38) is an integral equation which is revertible into an ordinary differential equation as follows:

$$E = \frac{RT}{nF}\left(\ln\frac{C_o}{C_i}\right)\exp\left\{\tfrac{1}{2}\int_C\left[\left(\frac{d^3x}{dt^3}\right)^2+\left(\frac{d^3y}{dt^3}\right)^2\right]dt\right\}$$

and $$y(t)=\alpha(t)x(t) \tag{3.45}$$

where $\alpha(t)$ is a constant.

Let t_k be the time taken for *k-th* jerk to take place, then $T=\{t_1,t_2,\cdots,t_{n-1},t_n,t_{n+1},\cdots\}$ forms an infinite sequence of time taken for successive jerks to make complete revolution; there exists a corresponding real interval $I_k=(0,t_k)$ for each t_k. And by the following definition of covering for a set:

Definition 3.1: Let ***F*** be a subset of ***R***, a collection ***G*** of subsets of ***R*** is said to be a covering for ***F*** (or ***G*** is said to cover *F*) if $F\subset\left(\bigcup_i\{G_i : G_i\in G\}\right)$. ***G*** is called an open covering for ***F*** in case each $G_i\in G$ is an open set. *G* is said to be a finite covering if ***G*** contains only a finite number of sets, say, *G₁*, *G₂*, *G₃*, ... *Gₙ*. (Note that in the last sentence we do not imply that the sets *Gₖ* are finite sets, i.e. we do not imply that *Gₖ* consists only of a finite number of points).

And by the Heine-Borel theorem:

Theorem 3.1 (Heine-Borel Theorem): Let ***F*** be a closed and bounded subset of ***R***. then every open covering of ***F*** has a finite sub-covering.

{That is, whenever ***G*** is a collection of open sets such that $F \subset \left(\bigcup_i \{G_i : G_i \in G\} \right)$, there exists a finite collection $\{G_i\}_{i=1}^n$ of sets in ***G*** such that $F \subset \left(\bigcup_{i=1}^n G_i \right)$.}

We can choose for all time t_k, an open covering $I = \bigcup_k I_k = (0,t)$ such that $I_k \subset I$ $\forall$ $k = 1,2,\cdots$, where $t = \sup_k t_k$ or simply put, $t = \max_k t_k$. Thus defining equation (3.38) over *I*:

$$V = \frac{RT}{nF}(\ln C_o C_i)\exp\left\{ \tfrac{1}{2} \int_C \left[\left(\frac{d^3x}{dt^3} \right)^2 + \left(\frac{d^3y}{dt^3} \right)^2 \right] dt \right\}$$

$$\Rightarrow \qquad \frac{nVF}{RT \ln C_0 C_i} = \exp\left\{ \tfrac{1}{2} \int_I \left[\left(\frac{d^3x}{ds^3} \right)^2 + \left(\frac{d^3y}{ds^3} \right)^2 \right] ds \right\} \qquad (3.46)$$

$$\Rightarrow \qquad \ln \xi(t) = \int_I \left[\left(\frac{d^3x}{ds^3} \right)^2 + \left(\frac{d^3y}{ds^3} \right)^2 ds \right] \qquad (3.47)$$

where $\xi(t) = \left(\frac{nVF}{RT\ln C_0 C_i}\right)^2$ and $J(t) = \left(\frac{d^3x}{ds^3}\right)^2 + \left(\frac{d^3y}{ds^3}\right)^2$ is the jerk that emanated from left hand side (L.H.S.) of (3.38). Clearly $\exists$ an open sphere $\wp(t,x,y)$ such that a class C(t,x,y) of solution of (3.38) exists in the open sphere $\wp(t,x,y)$.

However $\exists$ two possible natures the R.H.S. of (3.38) could take, namely when:

- Jerk $J(t) = \left(\frac{d^3x}{ds^3}\right)^2 + \left(\frac{d^3y}{ds^3}\right)^2 = b$, where b is a constant, and
- Jerk $J(t) = \left(\frac{d^3x}{ds^3}\right)^2 + \left(\frac{d^3y}{ds^3}\right)^2 = z(t)\cos\beta t$, i.e. L.H.S. is not a constant.

Case One: when $J(t) = \left(\frac{d^3x}{ds^3}\right)^2 + \left(\frac{d^3y}{ds^3}\right)^2 = b$, is a constant? Equation (3.47) then becomes:

$$\ln\xi(t) = \int_{t_0}^{t} b\, ds \tag{3.48}$$

i.e.
$$\ln\xi(t) = \omega b(t - t_0) \tag{3.49}$$

hence,
$$V = \frac{RT}{nF}(\ln C_0 C_i)\exp\{\tfrac{1}{2}\omega b(t - t_0)\} \tag{3.50}$$

where ω is the angular velocity of the fluid element (i.e. the transmitter).

At $t = t_0$, let $\frac{d^k x}{dt^k} = x_0^{(k)}, \frac{d^k y}{dt^k} = y_0^{(k)}, \mathrm{k} = 1, 2, 3.$ thus:

$$x(t) = x_0 + x_0^{(1)}t + x_0^{(2)}t^2 + x_0^{(3)}t^3 + \cdots \tag{3.51}$$

and
$$y(t) = y_0 + y_0^{(1)}t + y_0^{(2)}t^2 + y_0^{(3)}t^3 + \cdots \tag{3.52}$$

Case Two: when $J(t) = \left(\frac{d^3x}{ds^3}\right)^2 + \left(\frac{d^3y}{ds^3}\right)^2$ is not a constant? Equation (3.47) becomes:

$$\zeta(t) = \int_t z(s)\cos\beta s \, \mathrm{ds} \qquad (3.53)$$

Where $\zeta(t) = \ln\xi(t)$ and $J(t) = \left(\frac{d^3x}{dt^3}\right)^2 + \left(\frac{d^3y}{dt^3}\right)^2 = z(t)\cos\beta t$. The onus on us at this juncture is to find $z(t)$. We introduce Fourier transforms techniques to find *z(t)*, in accordance with the following theorem:

Theorem 3.2: *Given that f(x) is piecewise continuous in every finite interval I, and has a right hand and a left hand derivative at every point $x \in I$, and if the integral*

$$\int_{-\infty}^{\infty} |f(x)| dx \qquad (3.54)$$

exists, then f(x) can be represented by a Fourier integral. At a point of discontinuity of f(x) on I the value of the Fourier integral is equal to the average of the left- and right-hand limits of f(x) at that point.

Clearly *J(t)* satisfies all the requirements in theorem (3.2) above. Let the L.H.S. of equation (3.53) be:

$$\zeta(t) = x(t)\sqrt{\frac{\pi}{2}} \qquad (3.55)$$

Then
$$x(t) = \sqrt{\frac{2}{\pi}} \int_t [z_1(\tau)\sin\omega\tau + z_2(\tau)\cos\omega\tau] \mathrm{d}\tau \qquad (3.56)$$

Applying Fourier Transform techniques,

$$x(\omega) = \sqrt{\frac{2}{\pi}} \int_0^\infty z(\omega) \cos \omega t \, \mathrm{dt} \tag{3.57}$$

Hence
$$z(t) = \sqrt{\frac{2}{\pi}} \int_0^\infty \gamma(\omega) \cos \omega t \, d\omega \tag{3.58}$$

Since $\xi(t)$ and (3.51) defines $z(t)$, we can express $x(t)$ as a power series:

$$x(t) = \sum_{i=0}^{\infty} a_i t^i = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + \cdots \tag{3.59}$$

where $a_k = \frac{1}{k!} \frac{d^k z}{dt^k}$ at $t = 0$. And by (3.45);

$$y(t) = \sum_{i=0}^{\infty} b_i t^i = b_0 + b_1 t + b_2 t^2 + b_3 t^3 + \cdots \tag{3.60}$$

To obtain an expression for a_k, $k = 0,1,2,\cdots$, it will be recalled that the entire trajectory depends on the initial excitation brought about by the action potential aggravated by the potential difference, E, as generated by the difference in the concentration C_i and C_o of the fluid element respectively for the inside and outside the cell membrane. Therefore,

$$\varsigma(t) = 2 \ln \left(\frac{nVF}{RT \ln C_0 C_i} \right) \tag{3.61}$$

is constant, ω is maximum at $t = 0$ and (3.61) is zero at $t = t_f$. As such, when time t in $x(t)$ is normalized (3.59) may be expressed as Taylor series of both sine and cosine functions as:

$$z(t) = \varsigma(0)\left\{ \sum_{n=0}^{\infty} \frac{1}{n!}\left[\int_t d\omega \left(\frac{d^n}{ds^n}(\cos \omega s)\bigg|_{s=0} \right)\right]\right\} \tag{3.62}$$

Since the integral and summation operators are both linear (3.62) may be express thus;

$$z(t) = \varsigma(0)\left\{ \int_t \left(\sum_{n=0}^{\infty} \frac{s^n}{n!}\frac{d^n}{ds^n}(\cos \omega s)\bigg|_{s=0} \right) d\omega \right\} \tag{3.63}$$

Hence,

$$z(t) = \varsigma(0)\left[\int_\omega ds \left(1 - \frac{(ts)^2}{2!} + \frac{(ts)^4}{4!} - \frac{(ts)^6}{6!} + \cdots \right)\right] \tag{3.64}$$

$$z(t) = \varsigma(0)\left[\omega - \frac{\omega^3 t^2}{3!} + \frac{\omega^5 t^4}{5!} - \frac{\omega^7 t^6}{7!} + \cdots \right] \tag{3.65}$$

And,

$$x(t) = \sqrt{\frac{2}{\pi}} \int_0^\infty d\tau \, z(\omega) \cos \omega t$$

$\Rightarrow$

$$x(t) = \sqrt{\frac{2}{\pi}} \int_0^\infty dt \, \varsigma(0)\left[\omega - \frac{\omega^3 t^2}{3!} + \frac{\omega^5 t^4}{5!} - \frac{\omega^7 t^6}{7!} + \frac{\omega^9 t^8}{9!} - \cdots \right]\cos \omega t \tag{3.66}$$

Integrating by parts, we obtain the following coefficients $a_k \; \forall \; k \in N$ in (3.59). Hence,

$$a_0 = \varsigma(t)\left(1 - \frac{2!}{3!} + \frac{4!}{5!} - \frac{6!}{7!} + \cdots \right)\sin \omega t$$

$$a_1 = \varsigma(t)\left(\frac{2!}{1!3!} - \frac{4!}{1!5!} + \frac{6!}{1!7!} - \frac{8!}{1!9!} + \cdots \right)\omega \cos \omega t$$

$$a_2 = \varsigma(t)\left(\frac{2!}{2!3!} - \frac{4!}{2!5!} + \frac{6!}{2!7!} - \frac{8!}{2!9!} + \cdots\right)\omega^2 \sin \omega t$$

$$a_3 = \varsigma(t)\left(\frac{4!}{3!5!} - \frac{6!}{3!7!} + \frac{8!}{3!9!} - \frac{10!}{3!11!} + \cdots\right)\omega^3 \cos \omega t$$

$$a_4 = \varsigma(t)\left(\frac{4!}{4!5!} - \frac{6!}{4!7!} + \frac{8!}{4!9!} - \frac{10!}{4!11!} + \cdots\right)\omega^4 \sin \omega t$$

$$a_5 = \varsigma(t)\left(\frac{6!}{5!7!} - \frac{8!}{5!9!} + \frac{10!}{5!11!} - \frac{12!}{5!13!} + \cdots\right)\omega^5 \cos \omega t$$

$$\vdots$$

Therefore, for *n* even,

$$a_n(t) = \omega^n \sin \omega t \sum_{k=0}^{\infty} (-1)^k \frac{(n+2k)!}{n!(n+2k+1)!} \tag{3.67}$$

And for *n* odd,

$$a_n = \omega^n \cos \omega t \sum_{k=0}^{\infty} (-1)^k \frac{(n+2k+1)!}{n![n+2(k+1)]!} \tag{3.68}$$

At this juncture, *x(t)* is evaluated at various values of *t*, *ω* given the values of *V*, *C_i*, and *C₀*. The solution to the crisp model is represented graphically as follows:

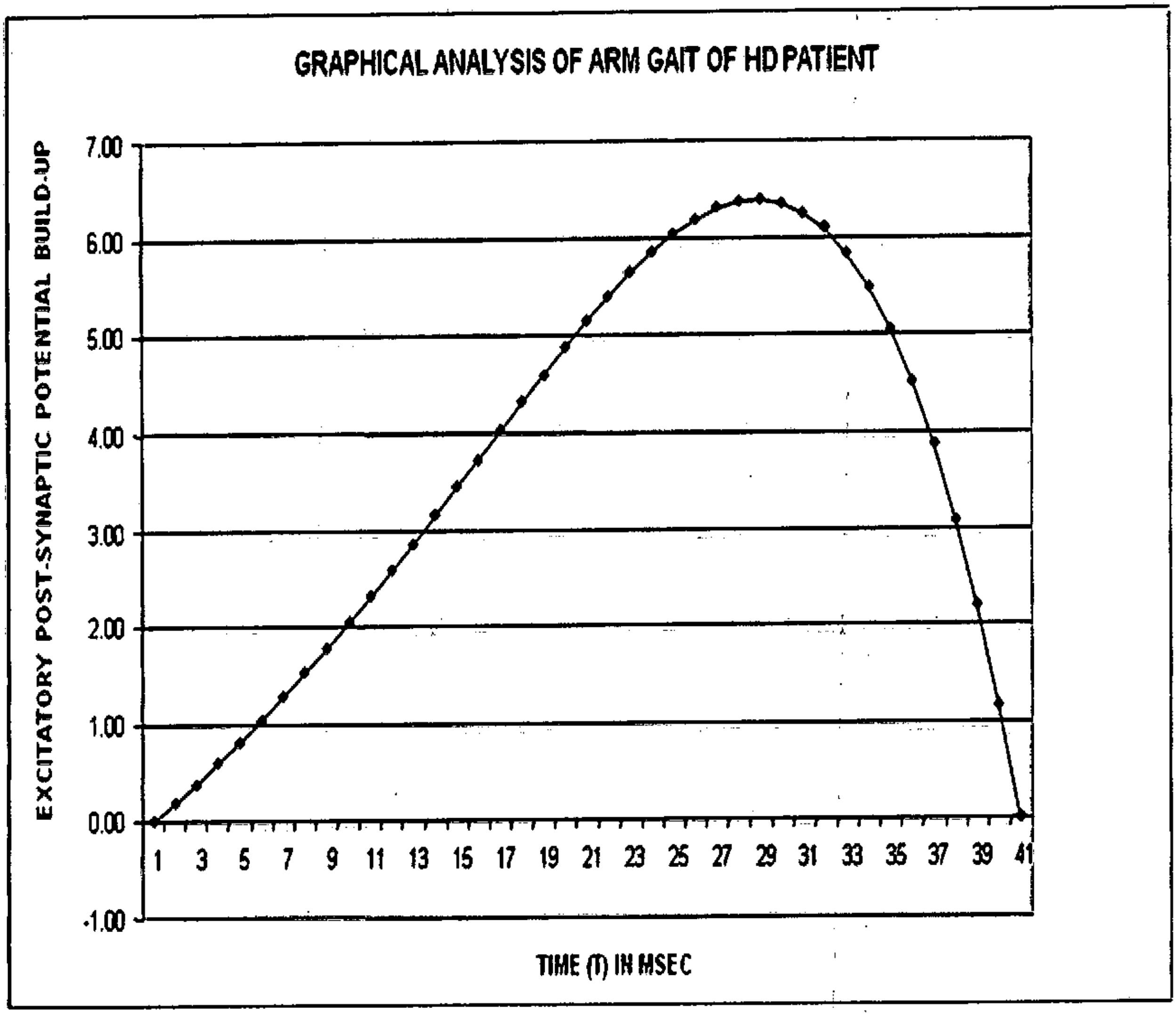

Figure 3.3: The graph of ESPS against time for the Crisp Model

Details containing the table of values and enlarged version of the graph are as contained in Appendix A-1.

B. ANN MODELLING OF THE ARM GAIT IN HD

Theoretical studies of motor control have proposed that the brain generates motor commands as a consequence of computations that resemble control policies and internal models. Control policies allow the brain to select goals and plans actions, while internal models computes motor commands that are appropriate for those plans and monitor sensory feedback in order to update the plans. For instance, when the goals is to reach a target, the motor system may evaluate the current state of the limb with respect to the goal and use a control policy to plan a small change in hand position. It may use an internal model of limb's inverse dynamics called an inverse model, to convert the plan into motor commands or the internal model of limb's forward dynamics called a forward model, to predict the sensory consequences of the motor commands and compare this prediction with sensory feedback to re-estimate current hand position with respect to the goal and update the motor plan by issuing an error-dependent response aimed at correcting the ongoing movement.

3.4 The Arm Gait Mechanisms

According to Smith, M.A. et al (2005), two different compensatory mechanisms are engaged when the nervous system senses errors during a reaching movement. The simple control method therefore would be to generate corrective responses proportional to the sensed error. The significant delays that exist in the sensorimotor loop would require the gain of these responses to be quite small to maintain stability of the arm. These delays may be effectively compensated if the motion state of the arm can be predicted at the time point when compensatory motor command would take effect. Such a predictor is termed a forward model of dynamics. And it can be computationally implemented using delayed sensory feedback, knowledge of the recent history of the motor output and knowledge of how the arm is likely to respond to this motor output. A good forward model can reduce delay driven instability allowing high feedback gains and powerful corrective responses based on simple linear responses to error, but these responses may not maximize the smoothness or efficiency of movement.

Huntington's disease patients become somewhat clumsy and may have trouble with fine motor tasks such as tying shoelaces, buttoning clothing, or performing needlework. This is largely due to a disturbance in error feedback control: errors in the early part of the movement were poorly compensated by the motor commands in the remainder of the movement. This suggested that one of the many computational mechanisms that are involved in error feedback control was affected by damage to the basal ganglia. Error in a given movement not only requires a motor response during the same movement, it also requires a response in the subsequent movement: the error changes the inverse model that is thought to be used by the brain to compute the motor commands that initiate the subsequent movement.

3.5 Comparative Analyses of Crisp and ANN Models for the Arm Gait of Huntington's disease Patient

In section A of this chapter, we proposed a crisp model for the arm gait of HD patients and we provided an analytical solution which solves the proposed model. However, the crisp model failed to capture the jerk in the arm of an HD patient, hence the need to propose a new model. Specifically, the new model is proposed for the following reasons:

- The Frobenius method is plagued by jump discontinuities.
- The solution is a parabolic smooth curve which misses out the staccato nature (jerk) of the arm gait of an HD patient; and

- There is a need to look for a more realistic representation (model) of the choreiform movement.

This takes us to the terrain of Artificial Neural Network (ANN) analysis of the arm gait of HD patient based on critical parameters in equation (3.8) above.

3.6 Methodology in Artificial Neural Network

In this section, we have discussed the procedural considerations for the neural networks techniques for the prediction of the arm gait of the HD patient under the following subheadings:

- Tuning the Network
- Data Preprocessing
- Training of the Network
- Data Deprocessing

The steps are considered in their order of application in the ANN algorithm.

3.6.1 Tuning Parameters for the Artificial Neural Network

This is the first step in this neural network analysis. It is essential to identify the set up parameter for the network. Some parameters considered are:

- the number of hidden layers
- the size of hidden layers
- the learning constant, β
- the momentum parameter, α
- the range, format and bias of data presented to the network

- the form of the activation function (sigmoid is used here)

The output layer has a single unit, which is the expected change for the prediction problem. One middle layer is used. It is best to choose the smallest number of neurons possible for a given problem to allow for generalization. A major set back associated with the use of too many neurons in exercise is problem of memorization of patterns that may in turn exacerbate inefficiency of the neural network to effectively carry out accurate predictions outside the data in the training set.

3.6.2 Processing the ANN Data for the Arm Gait in HD

For the neural network to function the acquired raw data must be preprocessed. The sigmoid activation function is used for data preprocessing in this thesis. The steps involved in preprocessing of the raw data are:

(a) Presenting a data which is a second derivative of the data set

$$d_i = P_{i+1} - P_i \tag{3.69}$$

where P_k is the row matrix representing the data on level k of the Neural Net.

(b) The next step is to normalize the data, viz:

$$t_i = \frac{d_i - \mu}{\sigma} \tag{3.70}$$

where $\mu = mean$ and $\sigma = s\tan dard\ deviation$

(c) In image processing edge can be detected by accenting change with the function $\left(\frac{a-b}{a+b}\right)$, where a and b are adjacent pixel values.

This enables feature detection and will be used to accent change in the data:

$$\dot{s}_i = \frac{P_{i+1} - P_i}{P_{i+1} + P_i} \tag{3.71}$$

Therefore, all columns from the last (feature detection) procedure are appended with the columns from the previous (squashing) procedure. This doubles the number of columns.

3.6.3 Training the ANN for the Arm Gait in HD

The network is trained using the back propagation algorithm. The weights are initialized with random floating point numbers in the range $[-1, 1]$ and the error function used is the mean square error defined as:

$$\frac{1}{T}\sqrt{\sum_{i=1}^{T}(\sigma_i - t_i)^2} \tag{3.72}$$

where T is the number of output units, σ is the network output and t is the desired target output. This error will be propagated backward for each training pattern and for each epoch. The Back Propagation Algorithm used is as given above:

3.6.4 The Back Propagation Algorithm for the ANN Simulation

Step 1: Read first input pattern and associated output pattern

$$\text{CONVERGE} = \text{TRUE} \tag{3.73}$$

Step 2: For input layer – assign as net input to each unit in its corresponding element in the input vector. The output for each unit is the net input.

Step 3: For the first hidden layer units – calculate the net input and output

$$\text{net}_j = W_o + \sum_{i=1}^{n} x_i W_{ij} \tag{3.74}$$

$$\sigma_j = \frac{1}{1+\exp(-\mathrm{net}_j)} \tag{3.75}$$

W_o = initial weight values, x_i = input vector and $W = \{W_{ij}\}$is the weight matrix.

Step 4: For the output layer units – calculate the net input and output.

$$\mathrm{net}_j = W_o + \sum_{i=1}^{n} x_i W_{ij} \tag{3.77}$$

$$\dot{\sigma}_i = \frac{1}{1+\exp(-\mathrm{net}_j)} \tag{3.78}$$

Step 5: Is the difference between target and output pattern within tolerance? If No,

$$\text{THEN CONVERGE = FALSE} \tag{3.79}$$

Step 6: For each output unit calculate its error,

$$\sigma_j = (t_j - \sigma_j)\dot{\sigma}_j (1 - \sigma_j) \tag{3.80}$$

Step 7: For last hidden layer calculate error each unit

$$\sigma_k = \sigma_j (1 - \sigma_j) \sum_k \delta_k W_{kj} \tag{3.81}$$

Repeat step 7 for all subsequent hidden layers.

Step 8: For all layers, update weights for each unit,

$$\Delta W_{ij}(n+1) = \beta(\delta_j \sigma_j) + \alpha \Delta W_{ij}(n) \tag{3.82}$$

(last pattern is presented) CONVERGE – TRUE

STOP

Read next input pattern and associated output pattern and GOTO step 2.

3.6.5 Training Data for the Artificial Neural Network

We have used the same data, in Appendix A-1, for the kinematic model as well as the artificial neural network model so as to have leverage between the analytical model and our ANN model.

The data set used for training is obtain for various angular velocities namely π/2, π/3, π/8,π/10,π/12,π/14 (210 data samples); and these data can be found in Appendix B. For quick referencing figure A-2 in Appendix A contains the table of epoch values and the graph of crisp model versus ANN model.

The data set used for testing and prediction is π/4, π/6 (35 data sets for each).

The deprocessing is done by repeating the steps of the processing backwards.

The parameters used for this prediction are for pi/4 prediction,

Learning Rate: 0.0004 Error Tolerance: 0.0002

Number of cycles: 500 Architecture: 17 18 1

The parameters used for this prediction are for pi/6 prediction,

Learning Rate: 0.0002 Error Tolerance: 0.0002

Number of cycles: 500 Architecture: 17 18 1

The artificial neural network analysis was done based on the computer C++ programming language. The C++ program from which the results were obtained was based on the source data for the ANN as contained in the appendices attached. The primary data used for the artificial neural analysis are the same set of data used for the Frobenius solution of the crisp model.

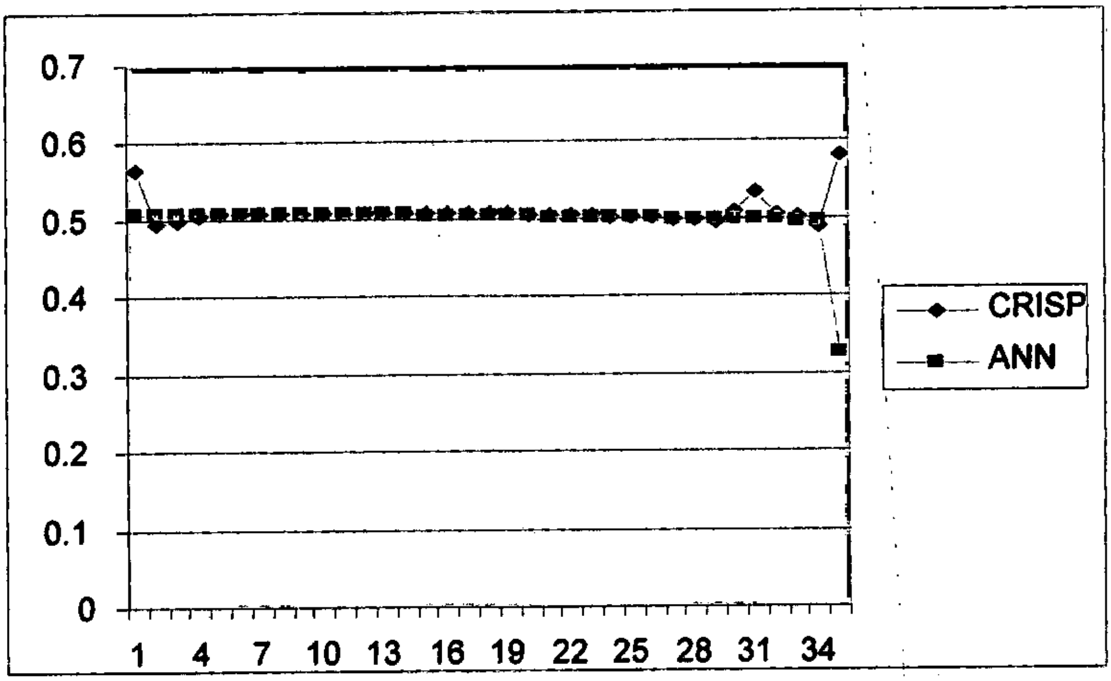

Figure 3.4: The graph of ANN Analysis of Arm gait of HD Patient using sigmoid function for $\alpha = \pi/6$

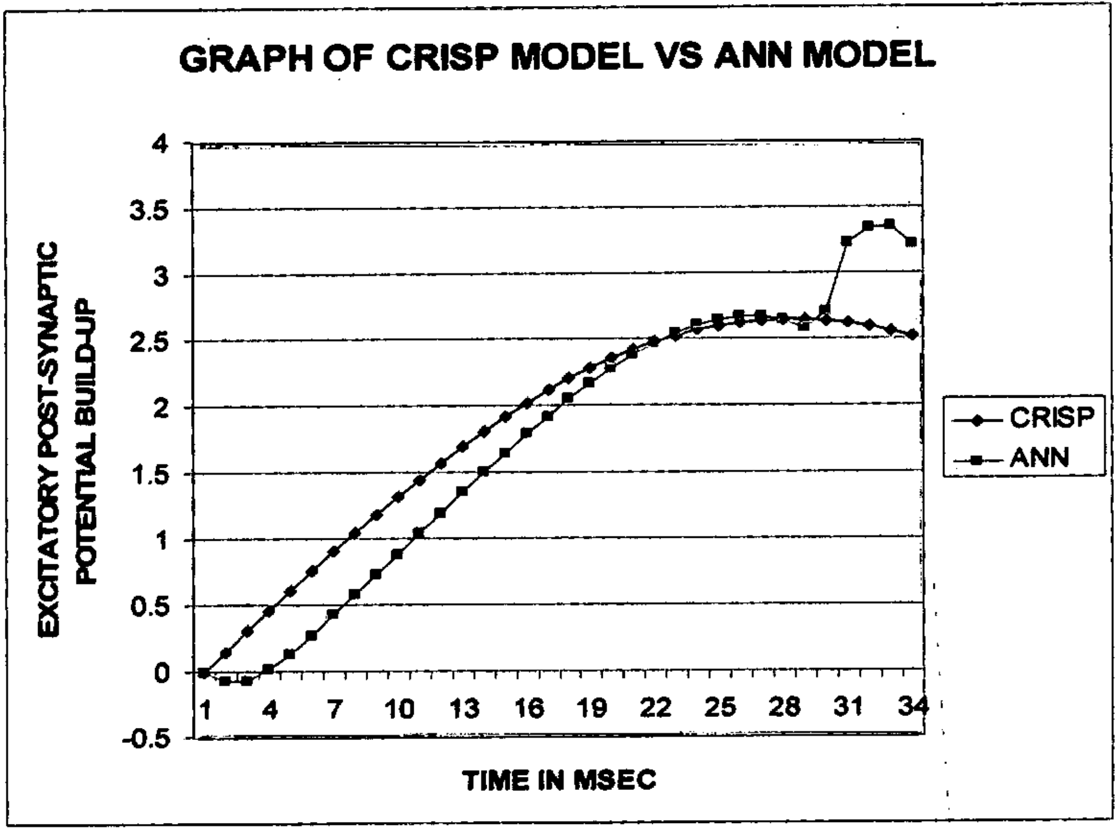

Figure 3.5: The graph of deprocessed NN Analysis of Arm gait of HD Patient using sigmoid function for $\alpha = \pi/6$

3.6.6 ANN Analysis of the Arm Gait of Huntington's Disease Patient

Having established the power and suitability of the Artificial Neural Network to describe the arm gait of HD patient since it captures the staccato nature of the jerk, we now go ahead and train an Artificial Neural Network for the arm gait of HD patient relating the sum/mean square errors of the training data vis-à-vis the test data using Stuttgart Neural Networks Simulator (SNNS).

3.6.7 Summary of Data for the Training/Testing of the ANN

The following statistics have been obtained from the data used for the training of the artificial neural network as well as the testing data. However a list of the sum square error and the corresponding means square error at various epochs are as tabulated for a training session with its graph.

	TRAINING DATA			TEST DATA		
EPOCH	SSE	MSE	SSE/O-UNITS	SSE	MSE	SSE/O-UNITS
10	20.27844	0.00378	20.27844	20.35651	0.00380	20.35651
20	21.32877	0.00398	21.32877	21.09393	0.00393	21.09393
30	22.94093	0.00428	22.94093	22.58194	0.00421	22.58194
40	25.16621	0.00469	25.16621	23.79496	0.00444	23.79496
50	27.43690	0.00512	27.43690	26.10738	0.00487	26.10738
60	30.67787	0.00572	30.67787	29.28762	0.00546	29.28762
70	35.23222	0.00657	35.23222	33.97247	0.00634	33.97247
80	43.22889	0.00806	43.22889	41.92018	0.00782	41.92018
90	61.74262	0.01152	61.74262	62.60379	0.01168	62.60379

Table 3.6: Comparative analysis of Sum Square Error, Mean Square Error of Training Versus Test Data at various epoch values during training session for the ANN using SNNS

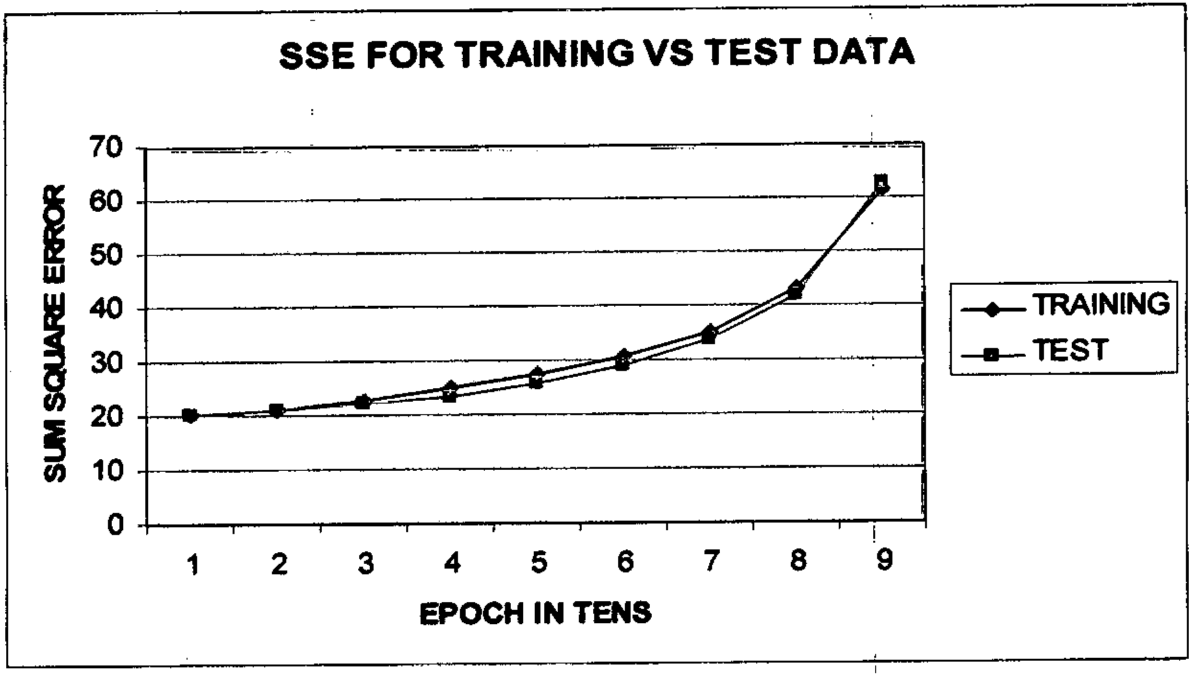

Figure 3.6: Graphical analysis of Sum Square Error, Mean Square Error of Training Versus Test Data at various epoch values during the first training session for the ANN using SNNS.

3.6.8 Comments on Figures 3.6

For figure 3.6, we have used a 5-3-6-1 architecture with one input layer having five nodes, two hidden layers having three and six nodes respectively and, one and only one output node. The graphical illustrations show that the discrepancies in all cases are within the set error margin of 0.05. The training is best with Figure 3.6 we shall show in the succeeding chapter that efficacy reduces as we continue with the training. This reveals that further training of the ANN may result in overtraining. A training session of 400 epochs is shown in Appendix A-3 with the enlarged version of the graph of the ANN for the comparative analysis of the Frobenius solution and the ANN method.

C. MANAGEMENT OF HUNTINGTON'S DISEASE

In this section we have consolidated on the gains of our analyses in the previous chapters to arrive at our ultimate goal; the model for the management of Huntington's disease. However, to give an impetus to our proposed model, we shall first enumerate the existing medical practices as regards neurodegenerative disease with clear emphasis on HD. To manage Huntington's disease, two broad approaches are used in practice. These are:

- Administration of drugs such as Tetrabenazine and Levetiracetam; and
- Electroconvulsive Therapy

3.7 Drug Administration in Huntington's Disease

Evidence supporting the pharmacological management of chorea and the psychiatric manifestations of Huntington's disease is summarized below. However, adjuvant psychotherapy, physiotherapy and speech therapy should be applied to provide optimal management, Bevan, E. (2006).

3.7.1 Chorea

Choreiform movements occur in approximately 90 percent of patients. Chorea is most prominent in the early stages of diagnosis of the disease. These become more prominent as the disease progresses. The first intervention in mild chorea should always be to discontinue drugs that have the potential to exacerbate symptoms. These drugs include piracetam and dopamine agonists, such as:

- Levodopa
- Amantadine, and
- Cabergoline

There are no published data pertaining to psychotropic drugs that can increase dopaminergic neurotransmission, such as aripiprazole and venlafaxine. These drugs should be considered as potential causes of exacerbations in dyskinetic movements and their use is probably best avoided, at least as first-line treatments. Other movement disorders include:

- Parkinson's disease
- Alzheimer's disease
- Disease of Cerebellum
- Epilepsy
- Progressive Sclerosis
- Contractions during child birth

to mention but a few.

Choreiform movements are often more distressing for carers and health care-professionals than they are for patients and it should not be assumed that intervention is always in a patient's best interest. If choreiform movements are problematic, the use of a small dose of typical antipsychotic such as haloperidol is established clinical ,practice. There is limited information available about the use of atypical antipsychotic for chorea. Two open pilot studies used olanzapine 5mg/day without success but a third open pilot study reported significantly improved motor function. There are also anecdotal case reports to suggest risperidone and quetiapine may be helpful. Several case reports suggest that moderate doses of risperidone (6mg) are needed to have a significant effect on motor disability. However, other case reports support lower doses (1mg twice daily) of risperidone in the treatment of chorea. It is therefore, unclear if higher doses of atypical antipsychotics may be required to achieve an optimal response in chorea, but these should be considered if lower doses produce a suboptimal response.

Tetrabenazine, a dopamine-depleting drug, is effective in treating various stage of the condition from moderate to severe choreiform movements. Efficacy is supported by double-blind placebo-controlled crossover trials. However, up to 80 percent of patients experience adverse effects, including sedation, insomnia, pseudo-Parkinsonism, depression, anxiety and akathisia. According to literature, serious side effects may occur in the course of the disease. These include: neuroleptic malignant syndrome and dysphagia. And they may lead to death from aspiration pneumonia, having also been reported. The decision to treat chorea with tetrabenazine must be balanced against the added risk of developing Parkinsonism and depression, both of which are already common in HD. Levetiracetam also records some benefit reducing choreiform movements in a small short-term study. Hypo kinetic rigidity (decrease motor function leading to stiffness) can occur independently of antipsychotic medication in patients with HD. Treatment strategies are similar to those used in Parkinson's disease although patients with HD usually respond less well.

3.7.2 Psychosis

It is estimated that about 23 percent of patients with HD will develop psychotic syndrome during the course of their illness. These tend to present early in the course of the illness and ameliorate as cognitive function deteriorates. Early neuropathological changes include atrophy of the medial caudate. Neurotransmitter changes are complex but include a reduction in gamma-aminobutyric acid and acetylcholine; and an increase in glutamatergic activity. The net result appears to be a hyperdopaminergic state. It follows that antipsychotic drugs are likely to be effective. Case reports and case series show the benefit of individual agents but no randomized controlled trials have conducted.

The use of antipsychotic drugs in HD psychosis is complicated by the risk of exacerbating the underlying movement disorder. Some evidence supports the efficacy of typical antipsychotics, particularly haloperidol, when the HD is mild to moderate. As the disease progresses, typical antipsychotics tend to be poorly tolerated due to dystonia and Parkinsonism. Atypical antipsychotics tend to be used at this point although the evidence to support their efficacy and tolerability is also limited to case reports and series. Meco *et al* compared risperidone with haloperidol in three patients with HD and found that risperidone was comparable with haloperidol in two patients (and superior to haloperidol in the other patient) in reducing both dyskinesia and psychotic symptoms. Additional case reports support the efficacy of risperidone, quetiapine and aminsulpride, although extra pyramidal side effects can be problematic with all these drugs.

3.7.3 Depression

Depression is common in HD. Estimates of the point prevalence range from 9 to 63 percent but the true rate is probably between 40 and 50 percent. The suicide rate is four to six times higher than in people without HD. Suicide among patients diagnosed with HD tends to occur early in the course of illness. It has been suggested that this reflects the occurrence of suicide before motor skills decline to the point where the person is no longer physically able to take his or her own life.

There are two randomized controlled trials to guide treatment choice, Bevan, E. et al (2006). Case reports of successful treatment with tricyclic antidepressants (TCAs), monoamine oxidase inhibitors (MAOIs), mirtazepine, and selective serotomin reuptake inhibitors (SSRIs) have been published. Patients with HD seem to be particularly prone to the side effects that we commonly associated with the TCAs, namely sedation, falls and anticholinergic induced cognitive impairment. MAOIs are also potentially problematic because they can worsen neurotransmission. There has been almost no new primary literature in this area over the past 20 years. The use of SSRIs tends to be favoured because these drugs may also reduce the irritability and apathy that are commonly seen in HD. The choice of SSRIs is not affected by the patient having HD.

3.7.4 Dementia

Almost all patients with HD develop sub-cortical dementia. Patients in the later stage of the disease tend to have profound dementia. No robust data could be found on the use of cholinesterase inhibitors to treat dementia in HD. There are however, some data to suggest that galantamine can be used to regulate mood and behaviour, thus improving some of the psychotic features associated with HD. It is thought that this occurs through allosteric modulation of nicotinic acetylcholine receptors. There is no reason to suspect that the efficacy and tolerability of cholinesterase inhibitors would be any different in HD patients than in those with Alzheimer's disease, Bevan, E. et al (2006).

Patients suffering from psychosis, depression and dementia are likely to be referred to psychiatrist for advice and management. According to literature, some psychiatrists see enough cases to build up expertise in this area.

In summary, the literature consists entirely of case reports and case series. Most are old and treatment is largely empirical. According to Bevan and Paton, there is poor evidence on which to base decisions for the management of psychiatric symptoms in patients with HD. However, with the exception of tetrabenazine which is used to treat choreiform movements, no placebo controlled or randomized controlled trials were identified. Systematic studies are required before any definite conclusions can be drawn as to the efficacy of various approaches. However, this is unlikely to happen owing to the small number of patients diagnosed with HD. Clinicians who treat patients with HD should be encouraged to publish reports of both positive and negative outcomes to increase the primary literature base in this neglected area of care.

3.8 The Way Out

Electroconvulsive therapy seems to be well tolerated in HD patients. A notable example of this application is the **deep brain simulation device** by Metrode Incorporation, UK. Metrode Incorporation, a modest company with six employees made the device in 2006. It consists of an electrode that can be implanted in a patient's brain, sending out electrical impulses at programmed intervals to neutralize the unwanted excitatory post synaptic potential which may precipitate the tremor that constitute a major physical symptom of

Parkinson's disease. Several of such devices have been made in the US and other parts of the developed world.

3.9 The Design and Building of a Model for the Management Device

Hereafter, we considered the build-up of the model for the management of HD. This is done in two stages. The first stage enumerates the mathematical consideration of the physiological presentations in HD while the last delve in the model.

3.9.1 The Design

According to literature, Huntington's disease is known to be associated with writhe, twist, constant uncontrollable dance-like motion of various parts of the affected persons. In our model, the phrase "constant uncontrollable dance-like motion" implies that the jerk occurs periodically.

A function $f(x)$ is said to be periodic if it is defined for all real numbers x and if there is some positive number p such that

$$f(x+p)=f(x) \tag{3.83}$$

The real number p is then called the period of $f(x)$. The graph of such functions is obtained by periodic repetition of its graph in any interval p, Kreyzig, E. (1988). And by extension, we have that:

$$f(x)=f(x+p)=f[(x+p)+p]=f(x+2p)=\cdots=f(x+np) \tag{3.84}$$

where n is a positive integer.

In an HD free situation where both the excitatory and the inhibitory memories are functional, the law of electroneutrality is obeyed where macroscopic (or bulk) portion of physiological information in the neural circuit contains equal number opposite charges as a result of the damping effect of the inhibitory post synaptic potential (IPSP) on the precipitated excitatory post synaptic potential (EPSP) in the system. However, in HD patient two possibilities may have arisen:

1. it is not impossible that the memory of inhibitory activities is deleted, and/or
2. since HD results from mutant allele causing the CAG sequence to repeat itself abnormally, the excessive polyglutamine generated by such a mutation may have overwhelmed the IPSP produced resulting in the chorea condition

This condition is prevalent given the following reasons:

- CAG codes for glutamine
- Glutamine is a crystalline amino acid with the molecular formula $C_5H_{10}N_2O_3$
- Glutamine yields glutamic acid and ammonia on hydrolysis in accordance with the chemical equation $C_5H_{10}N_2O_3+H_2O \rightarrow C_5H_9NO_4 + NH_3$
- $C_5H_9NO_4$ is a neurotransmitter which excites postsynaptic neurons

In his work, Schaneggenburger, R. et al (2000) declared that action potential can be attained as many times as possible in a second. It is therefore logical that if a patient is confronted by any of the two (or both) situations as in equations (3.83) and (3.84) above, choreiform movements may occur. Moreover, since the physiological condition is constant, it clearly implies that the function $f(t)$ that describes the motion is periodic, thus:

$$f(t) = f(t + nT) \qquad \forall \quad n \in \mathrm{N} \tag{3.85}$$

where N is the set of natural numbers.

Let us consider a time interval $[0, I]$. Suppose the incidence of jerk within the time interval $[0, I]$ is λ, and then the time taken for single jerk to occur is $T = {}^{I}\!/_{\lambda}$. It is pertinent to state at this juncture that even though the literature says that the jerk is constant, it is silent on the uniformity of the jerk. It is therefore important that we assume the general case where:

- the jerk occurs at constant intervals, but
- the jerk is NOT uniform

and design our model in such a way as to accommodate the two scenarios. Hence, if V is the volume of drugs (or better still the quantum of electrical impulse) required to dampen the jerk then V must be a function of the Gibbs free energy required to drive the chemical process and by extension the electromotive force that results from the potential difference created by the gap that was born out of the marriage between the excess EPSP and the shortfall in the supply of the IPSP, Margeta-Mitrovic (2000). This translates to the following model.

3.9.2 The Model

Every model is aimed at achieving a set goal. The aim of our model is to neutralize the effect of excitatory post synaptic potentials (EPSP) that precipitate the chorea. In other words, our objective is to create an equilibrium position between the EPSP and the IPSP thereby causing the law of electro-neutrality to hold, Byrne, J.H. (2000). To this end, it is enough to create an artificial IPSP just before threshold. To achieve this goal, we therefore propose the model:

$$f(t)=\begin{cases}0 & \text{if} \quad 0\le t<T-\Delta t\\ V & \text{if} \quad T-\Delta t\le t\le T\end{cases} \tag{3.86}$$

Where $V=V_0\,E/E_0$; E is the potential for the jerk, E_0 is the maximum possible potential for the ailment, and V_0 is the constant volume of drug (or electrical influx) required to neutralize the EPSP that culminated in the jerk.

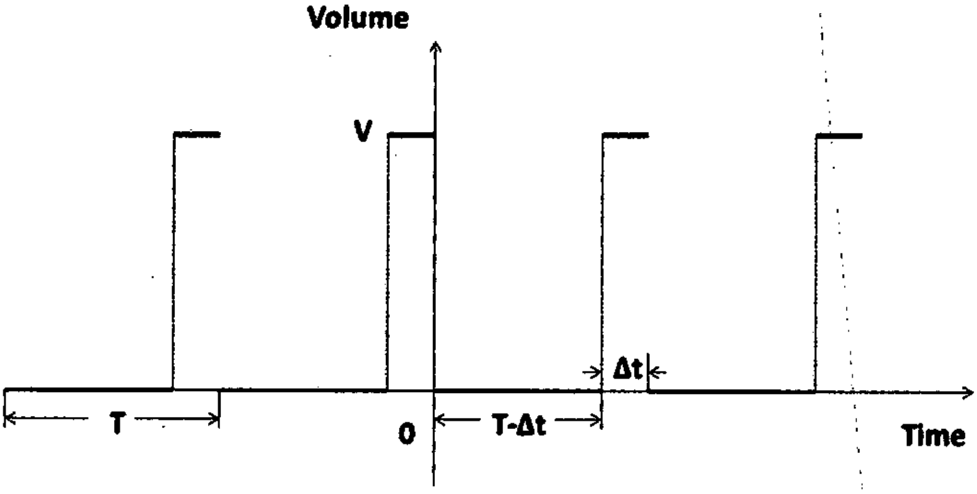

Figure 3.7: The graph of the Management Model

And the schematic diagram showing the flow of signals as predicted by the proposed model in equation (3.86) is as follows:

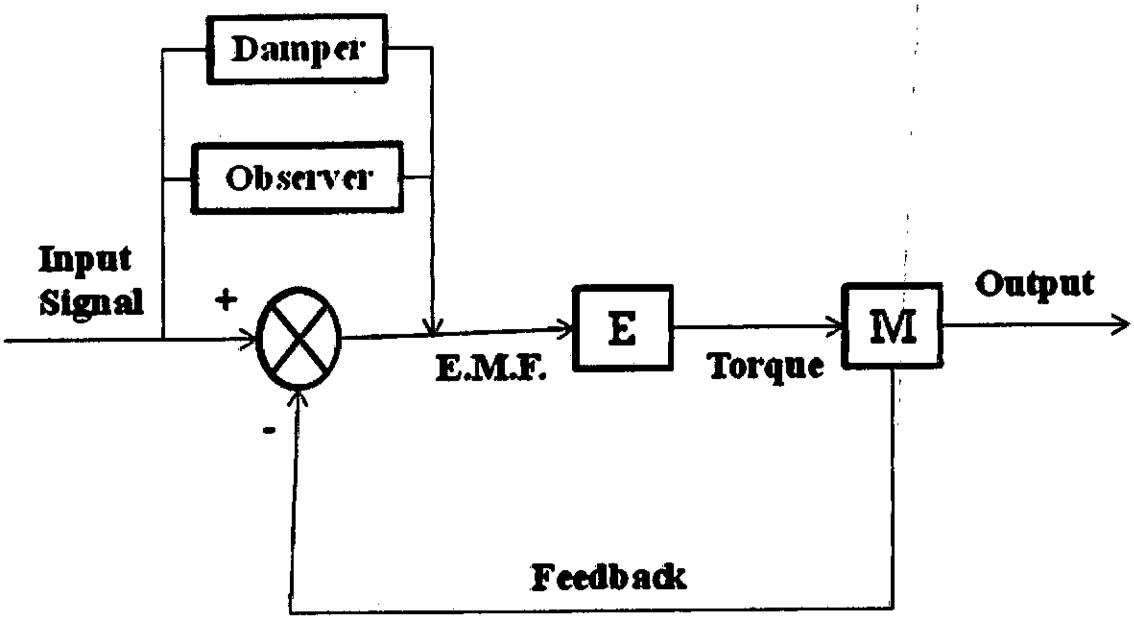

Figure 3.8: The Signal Flow diagram

Figure 3.8 represents a single electromechanical system consisting of electrical module (E) and mechanical module (M). However, for the purpose of our analysis the system has been decoupled in accordance with our earlier explanation in section 3.2. Since our objective is to enable HD patients leave a life devoid of chorea, our management model works based on a pattern classifier (i.e. an ANN simulator) where the interactive classifier monitors the build-up of the excitatory physiological information and kills the build-up process prior to threshold. Our management technique is therefore a predictive management as opposed to the conventional reactive management technique.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.0 Preamble

In this chapter, the results of both the crisp model and the Artificial Neural Network (ANN) simulation highlighted in Chapter three are presented, analyzed and discussed. Tables containing data which size cannot be accommodated within a page are presented in Appendix A for the perusal of user of this thesis. **Section 4.1** contains the result of the analysis of the crisp model presented in graphical form while **Section 4.2** contains the result of the artificial neural network simulation also in graphical form. In order to promote a better understanding of the two methodical analyses, we have done a comparative analysis of both the graph of the crisp model and that of the Artificial Neural Network in a single Cartesian plane but in different colours so as to remove any ambiguity in our presentation. In **Section 4.3** we have shown a sum/mean square error analysis between the training and the test data. We do not have the luxury of presenting the training data in full neither do we for the test data because each of the two will gulp between 500 and 750 pages. However, we have presented various values of both the training and the test data at distinct epoch values and graphed them. And we concluded every Section with a concise but clear discussion of the results.

4.1 The Results for the Crisp Modeling

It will be recalled that the crisp model for the arm gait of Huntington's disease patient was defined as follows:

$$\dot{E} = \frac{RT}{nF}\left(\ln\frac{C_o}{C_i}\right)\exp\left\{\tfrac{1}{2}\int_C\left[\left(\frac{d^3x}{dt^3}\right)^2+\left(\frac{d^3y}{dt^3}\right)^2\right]dt\right\} \tag{4.1}$$

where the critical parameters in determining this value are: the concentration of ions both inside C_i and outside C_o the neuron; the absolute temperature, T; the charge on the ion in question, n; the Faraday constant, F, the magnitude of the charge per mole of electrons; and the universal gas constant R.

And the solution of the model expressed in power series was given as:

$$x(t) = \sum_{i=0}^{\infty} a_i t^i = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + \cdots \tag{4.2}$$

where $a_k = \frac{1}{k!}\frac{d^k z}{dt^k}$ at $t = 0$.

Henceforth:

$$x(t) = \sqrt{\frac{2}{\pi}}\int_0^{\infty} dt\,\varsigma(0)\left[\omega - \frac{\omega^3 t^2}{3!} + \frac{\omega^5 t^4}{5!} - \frac{\omega^7 t^6}{7!} + \frac{\omega^9 t^8}{9!} - \cdots\right]\cos\omega t \tag{4.3}$$

which gave rise to the following coefficients $a_k \ \forall\ k \in N$ when integrated by parts:

$$a_n(t) = \omega^n \sin\omega t \sum_{k=0}^{\infty}(-1)^k \frac{(n+2k)!}{n!(n+2k+1)!} \tag{4.4}$$

for n even and

$$a_n = \omega^n \cos\omega t \sum_{k=0}^{\infty}(-1)^k \frac{(n+2k+1)!}{n![n+2(k+1)]!} \tag{4.5}$$

when n is odd.

The result was interpreted graphically and the graph is as displayed below:

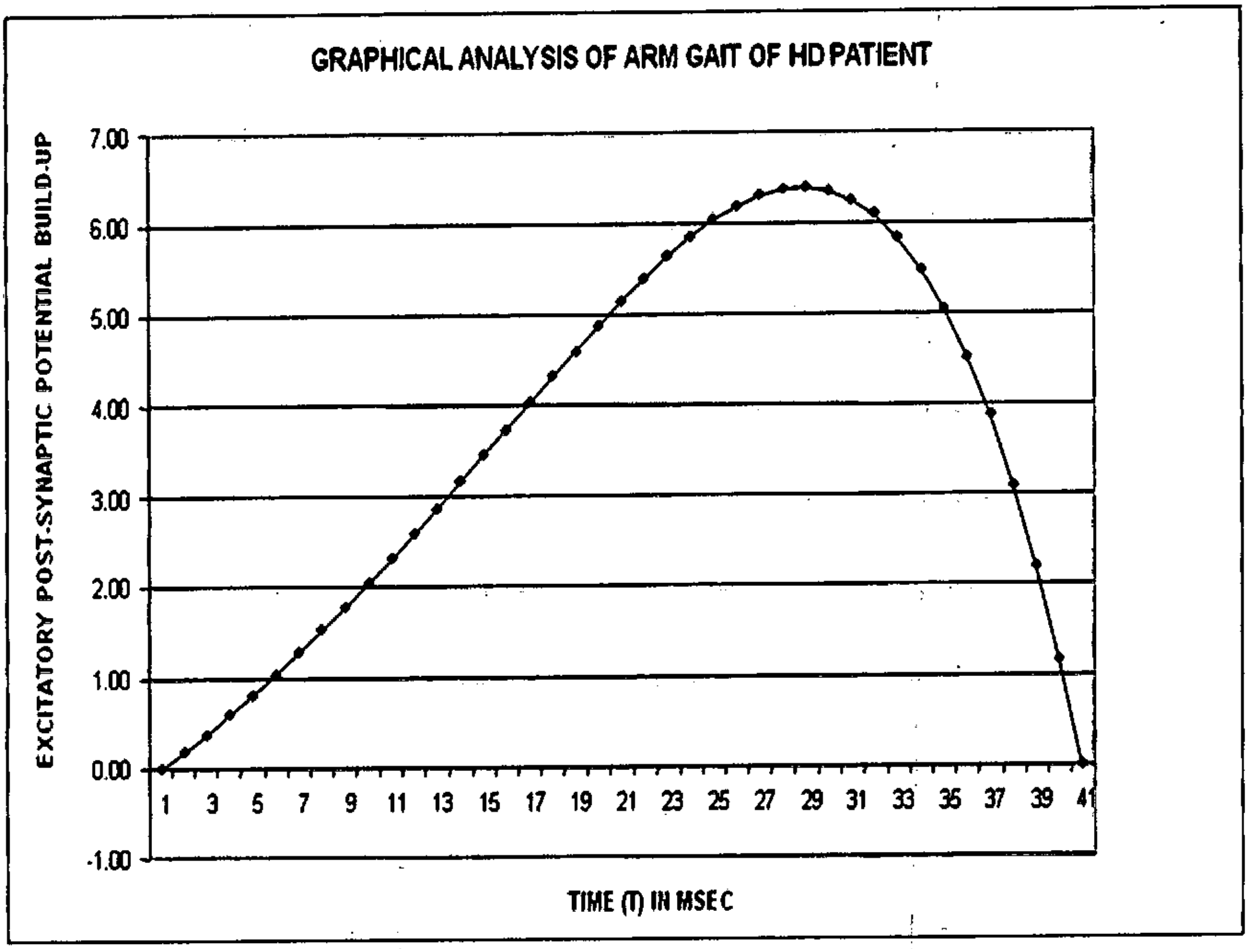

Figure 4.1: *The graph of EPSP against time for the Crisp Model*

4.1.1 Discussion

Figure 4.1 is the solution of the crisp model. The figure clearly reveals that the Frobenius solution of the crisp model is a parabolic smooth curve which agrees in totality with the existing scholarly works especially that of Flash and Hogan (1985) experimental evidence for *the minimum jerk model*, that simulates unconstrained point-to-point movements of the arm through a third specified point. The model which was based on the minimization of the rate of change of hand acceleration in a fixed Cartesian coordinate system also has a parabolic smooth curve (that does not reflect a jerk motion) as solution.

However, it is a known fact that Science has now reached a stage in the field of nanomedicine where in the near future nanorobots can be introduced into the body system of a patient which will release drug at programmed intervals to curb an ailment. In line with the purpose of our study therefore we require firm and impeccable platform for this mechanism. This has been found to have been provided by ANN simulation of the arm gait of an HD patient since, for our purposes we require a solution which will capture a point action in total agreement with the physiological presentation of the condition in question if we must indeed propose a workable model that will adequately arrest the choreiform movement that impedes day to day activities of an HD patient. It is in light of this fact that we forged ahead to propose an ANN simulation model and the results are as discussed in the subsequent section.

4.2 The Results of the ANN Simulation

In order that we may justify the necessity for the use of ANN simulation we have done a worthwhile comparison between the Frobenius solution and ANN simulation. To start with, we have generated the source data for the Artificial Neural Network simulation from the data set used for the crisp model, and for obvious reasons, ANN behaves well with large volume of data. To this end, we have obtained the expanded source data by merging the same data set with its derived data through the algorithm presented in section 3.6. And for easy inference, we have embedded the graph of the ANN simulation on the graph of the crisp model as shown in the following figure.

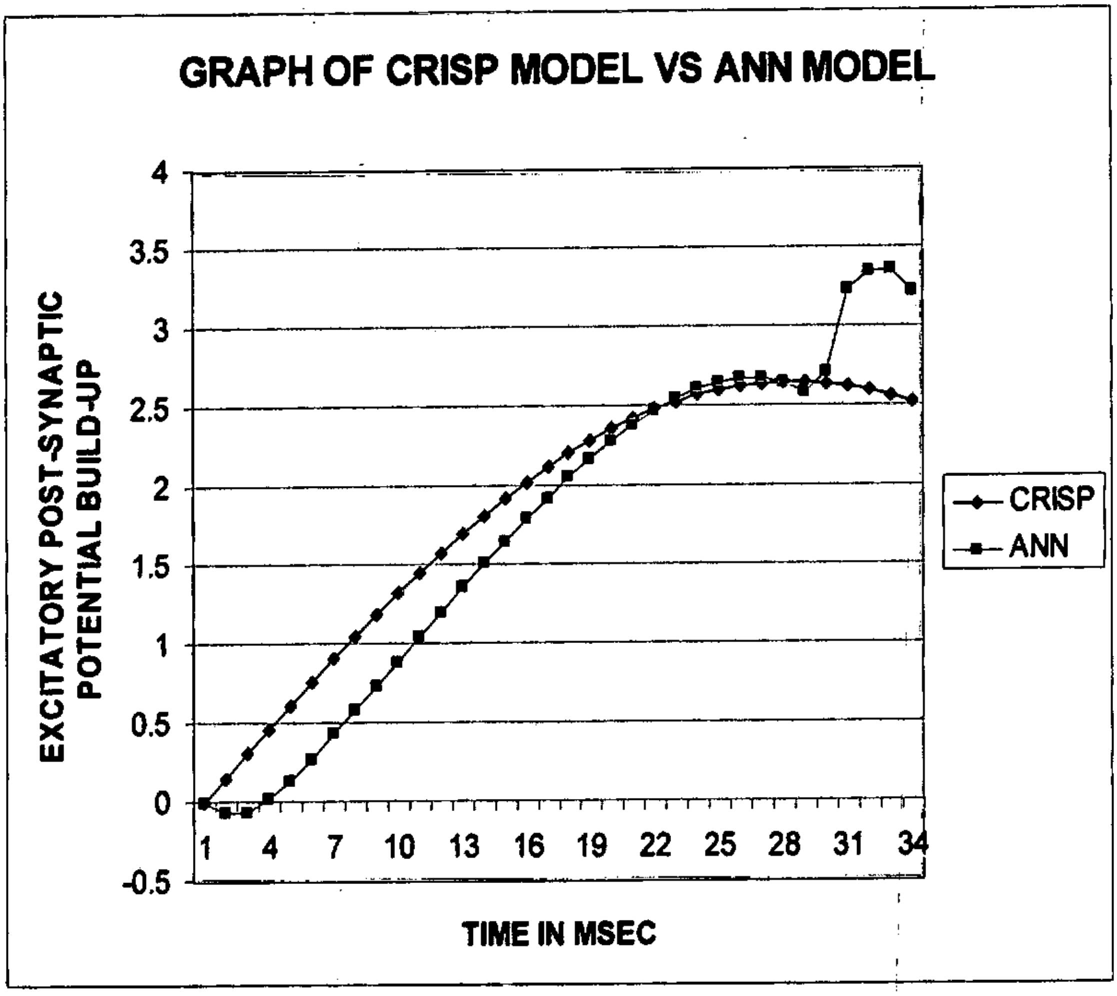

Figure 4. 2: The graph of deprocessed NN Analysis of Arm gait of HD Patient using sigmoid function for $\alpha = \pi/6$

4.2.1 Discussion

In diagram 4.2, we have graphed the crisp model versus the ANN model in order that we may carry-out a good comparative analysis of the two. The Frobenius solution for the crisp model is outlined in blue while the pink line represents the ANN model. From the figure it is clear that while the blue line is part of a parabolic smooth curve in figure 4.1, the pink graph clearly brings out the staccato nature of the jerk. It is therefore obvious that the ANN graph is a better representation of the jerk of the arm of an HD patient.

Since the crisp model gave rise to a parabolic smooth graph, it clearly misses out the staccato nature (jerk) of the arm gait of an HD patient and therefore failed to capture the jerk in the arm of an HD patient, hence the need to adopt the ANN model for the arm gait in HD patient because it is a more realistic representation (model) of the actual movement

4.3 Analyzing the ANN Simulation of the Arm Gait of HD Patient

Having established the power and suitability of the Artificial Neural Network to describe the arm gait of HD patient since it captures the staccato nature of the jerk, we now go ahead and discuss the training/testing sessions of our Artificial Neural Network relating the sum/mean square errors of the training data vis-à-vis the test data using Stuttgart Neural Networks Simulator (SNNS).

4.3.1 Detailed Functions and Data Samples used for the Training of the ANN

The following statistics have been obtained from the data used for the training of the artificial neural network as well as the testing data. However a list of the sum square error and the corresponding means square error at various epochs are as tabulated for distinct training sessions with their graphs.

	TRAINING DATA			TEST DATA		
EPOCH	SSE	MSE	SSE/O-UNITS	SSE	MSE	SSE/O-UNITS
10	20.27844	0.00378	20.27844	20.35651	0.00380	20.35651
20	21.32877	0.00398	21.32877	21.09393	0.00393	21.09393
30	22.94093	0.00428	22.94093	22.58194	0.00421	22.58194
40	25.16621	0.00469	25.16621	23.79496	0.00444	23.79496
50	27.43690	0.00512	27.43690	26.10738	0.00487	26.10738
60	30.67787	0.00572	30.67787	29.28762	0.00546	29.28762
70	35.23222	0.00657	35.23222	33.97247	0.00634	33.97247
80	43.22889	0.00806	43.22889	41.92018	0.00782	41.92018
90	61.74262	0.01152	61.74262	62.60379	0.01168	62.60379

Table 4.3: Comparative analysis of Sum Square Error, Mean Square Error of Training Versus Test Data at various epoch values during training session for the ANN using SNNS

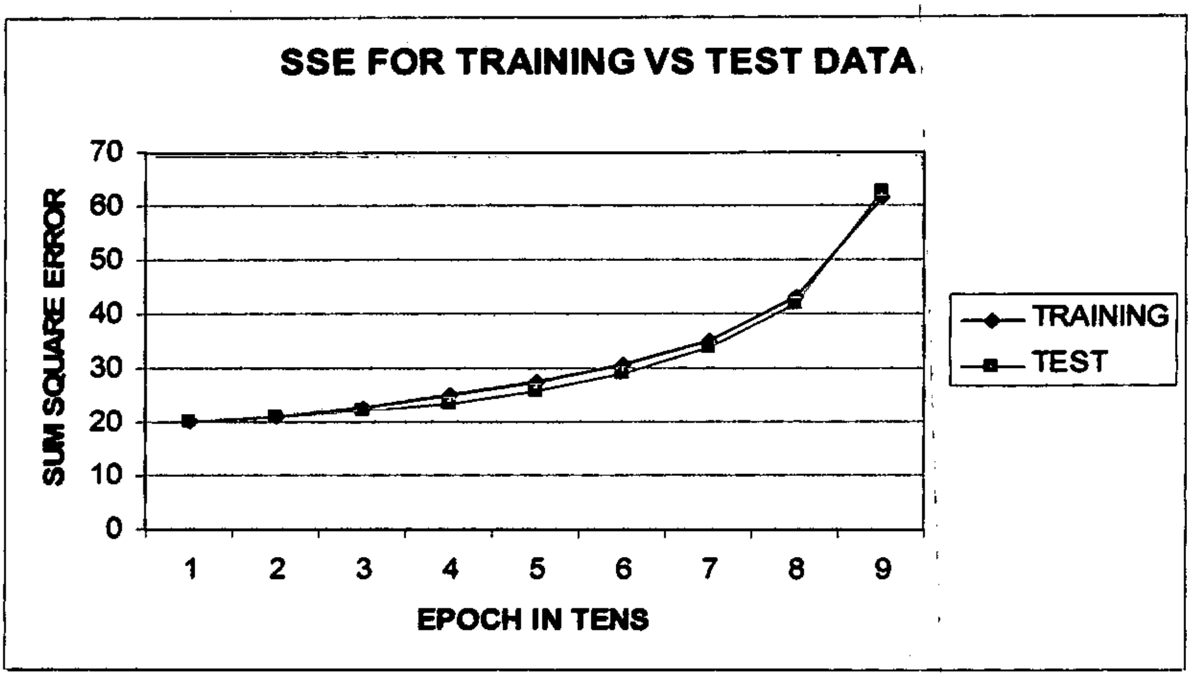

Figure 4.3: Graphical analysis of Sum Square Error, Mean Square Error of Training Versus Test Data at various epoch values during the first training session for the ANN using SNNS.

```
Learning func:  Std_Backpropagation
Update func:  Topological_Order
Init. func:  Randomize_Weights
Remap. func:  None
Learning all patterns:
  epochs   : 90
  parameter:  0.05000
  #o-units : 1
  #patterns: 5361 (total: 5361)
```

	TRAINING DATA			TEST DATA		
EPOCH	SSE	MSE	SSE/O-UNITS	SSE	MSE	SSE/O-UNITS
10	13.77107	0.00257	13.77107	13.32772	0.00249	13.32772
20	13.25124	0.00247	13.25124	14.56657	0.00272	14.56657
30	14.55394	0.00271	14.55394	13.80332	0.00257	13.80332
40	15.01160	0.00280	15.01160	14.25410	0.00266	14.2541
50	15.58682	0.00291	15.58682	15.3598	0.00287	15.3598
60	16.04093	0.00299	16.04093	15.55503	0.0029	15.55503
70	17.02671	0.00318	17.02671	16.01175	0.00299	16.01175
80	17.61792	0.00329	17.61792	16.56122	0.00309	16.56122
90	18.38665	0.00343	18.38665	17.33072	0.00323	17.33072
100	18.90159	0.00353	18.90159	19.68319	0.00367	19.68319

Table 4.4: Comparative analysis of Sum Square Error, Mean Square Error of Training Versus Test Data at various epoch values during the second training session for the ANN using SNNS.

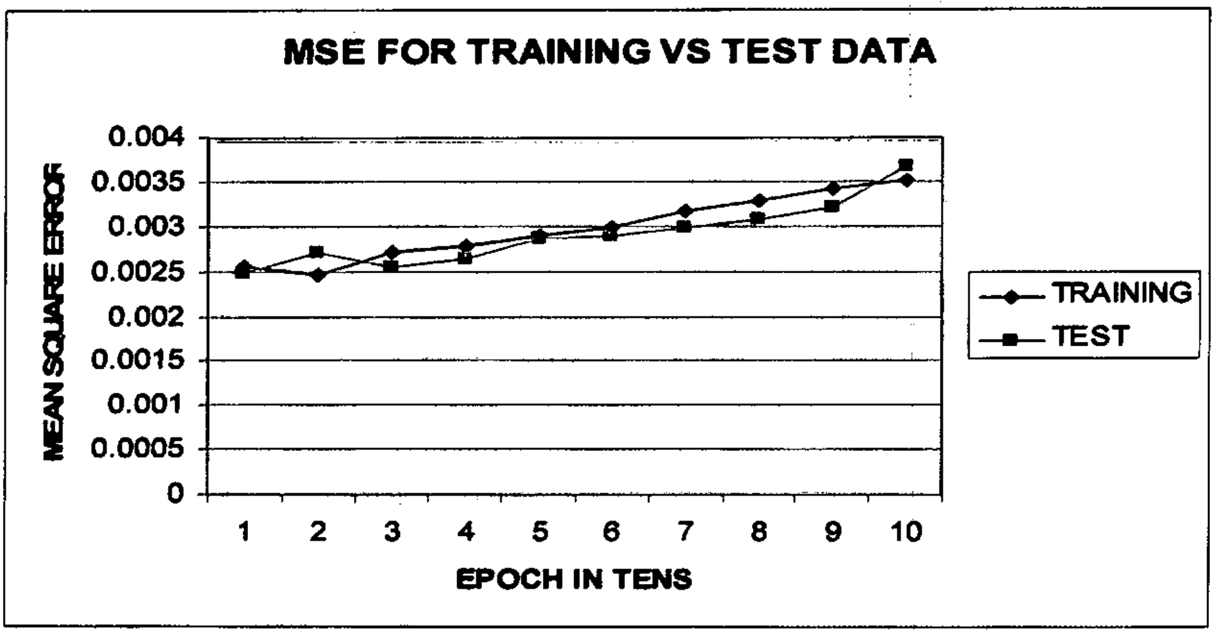

Figure 4.4: Graphical analysis of Sum square error, Mean Square Error of Training Versus Test Data at various epoch values during the second training session for the ANN using SNNS.

```
Learning func:  Std_Backpropagation
Update func:  Topological_Order
Init. func:  Randomize_Weights
Remap. func:  None
Learning all patterns:
  epochs   : 100
  parameter:  0.05000
  #o-units : 1
  #patterns: 5361 (total: 5361)
```

	TRAINING DATA			TEST DATA		
EPOCH	SSE	MSE	SSE/O-UNITS	SSE	MSE	SSE/O-UNITS
10	11.19287	0.00209	11.19287	13.33786	0.00249	13.33786
20	11.44877	0.00214	11.44877	12.21618	0.00228	12.21618
30	11.94615	0.00223	11.94615	10.75449	0.00201	10.75449
40	12.15728	0.00227	12.15728	11.06835	0.00206	11.06835
50	12.05269	0.00225	12.05269	13.93886	0.0026	13.93886
60	12.27877	0.00229	12.27877	12.20879	0.00228	12.20879
70	12.75445	0.00238	12.75445	11.8591	0.00221	11.8591
80	12.25608	0.00229	12.25608	12.16397	0.00227	12.16397
90	13.26925	0.00248	13.26925	12.35164	0.0023	12.35164
100	13.79329	0.00257	13.79329	12.62475	0.00235	12.62475

Table 4.5: Comparative analysis of Sum square error, Mean Square Error of Training Versus Test Data at various epoch values during the third training session for the ANN using SNNS.

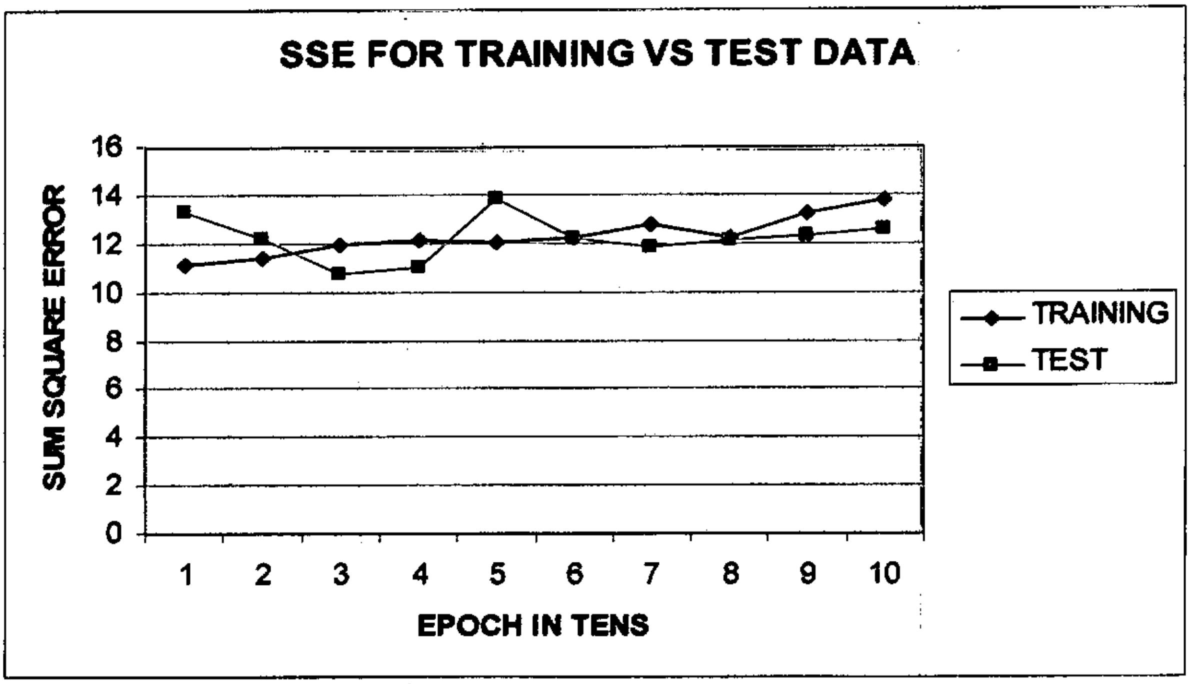

Figure 4.5: Graphical analysis of Sum Square Error, Mean Square Error of Training Versus Test Data at various epoch values during the third training session for the ANN using SNNS.

```
Learning func:  Std_Backpropagation
Update func:  Topological_Order
Init. func:  Randomize_Weights
Remap. func:  None
Learning all patterns:
  epochs    : 100
  Parameter:  0.05000
  #o-units : 1
  #patterns: 5361 (total: 5361)
```

	TRAINING DATA			TEST DATA		
EPOCH	SSE	MSE	SSE/O-UNITS	SSE	MSE	SSE/O-UNITS
10	9.68830	0.00181	9.68830	8.84850	0.00165	8.84850
20	9.62162	0.00179	9.62162	9.15772	0.00171	9.15772
30	9.73212	0.00182	9.73212	9.91758	0.00185	9.91758
40	10.26644	0.00192	10.26644	9.38853	0.00175	9.38853
50	10.37286	0.00193	10.37286	9.41039	0.00176	9.41039
60	10.22872	0.00191	10.22872	9.62538	0.00180	9.62538
70	10.31495	0.00192	10.31495	9.86351	0.00184	9.86351
80	10.72408	0.00200	10.72408	10.46962	0.00195	10.46962
90	10.85954	0.00203	10.85954	10.45353	0.00195	10.45353
100	11.03667	0.00206	11.03667	10.39531	0.00194	10.39531

Table 4.6: Comparative analysis of Sum Square Error, Mean Square Error of Training Versus Test Data at various epoch values during the fourth training session for the ANN using SNNS.

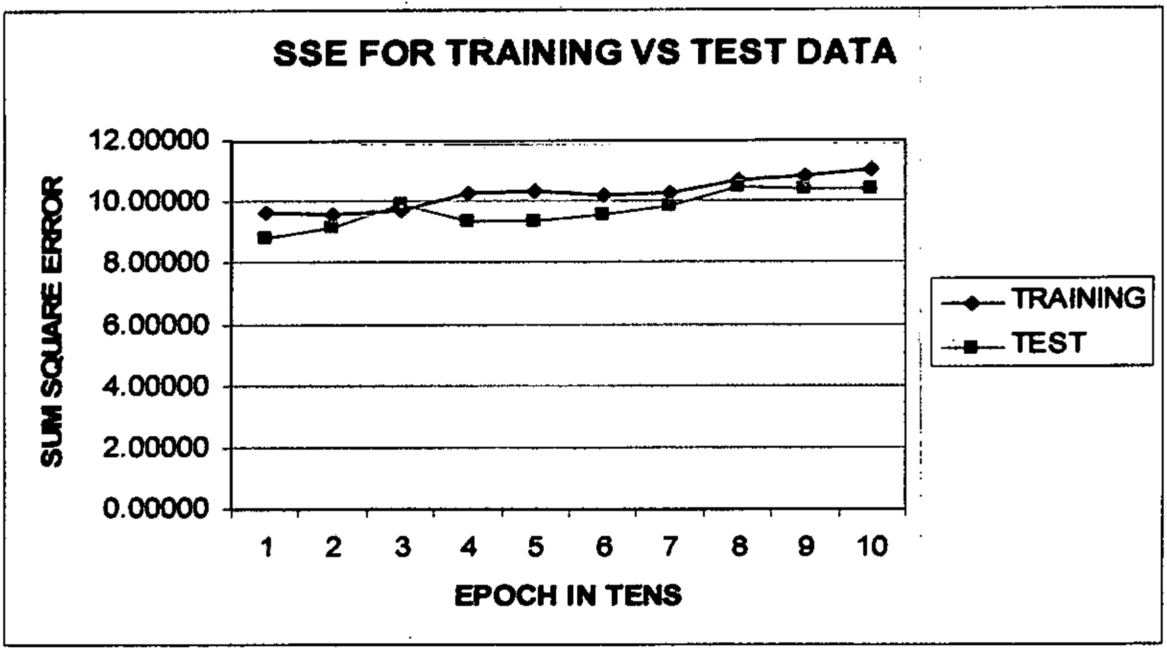

Figure 4.6: Graphical analysis of Sum Square Error, Mean Square Error of Training Versus Test Data at various epoch values during the fourth training session for the ANN using SNNS.

```
Learning func:  Std_Backpropagation
Update func:  Topological_Order
Init. func:  Randomize_Weights
Remap. func:  None
Learning all patterns:
  epochs   : 100
  parameter:  0.05000
  #o-units : 1
  #patterns: 5361 (total: 5361)
```

4.3.2 Comments Based on Training/Test Figures

For figures 4.3 to 4.6, we have used a 5-3-6-1 architecture with one input layer having five inputs, two hidden layers having three and six inputs respectively. There is one and only one output layer. The graphical illustrations show that the discrepancies in all cases are within the set error margin of 0.05. The training is best with Figure 4.3 while the efficacy reduces as we move from Figure 4.3 to Figure 4.6. Further training of the ANN may result in overtraining. A training session of 400 epochs is shown in Appendix A-3 with the enlarged version of the graph of the ANN for the comparative analysis of the Frobenius solution and the ANN method.

CHAPTER FIVE

CONCLUSION

In this work, a major headway had been made concerning the modeling and management of neurodegenerative diseases. Efforts have been geared towards managing one of the major physiological problems associated with such diseases; precisely the chorea in Huntington's disease (HD). To achieve this aim, we have made effort to capture the staccato nature of jerk by simulating same with an Artificial Neural Network so that we could proffer a workable model for the management of Huntington's disease, and by extension neurodegenerative diseases in general.

In trying to model the arm gait of HD patients, we have considered a kinematic analysis of same by constructing an electro-mechanical model, arriving at series solution to our crisp model. We based our deductions on the fact that arms move in circles about the joints among others. The graphical illustrations of our kinematic model revealed that the entire analysis gave a parabolic smooth curve, an indication that the Frobenius method of solution for the crisp model is not a viable representation of the choreiform movement that characterizes HD. As a result of the inadequacy of the aforementioned we proceeded to propose a more realistic representation of the jerk in question. The new solution technique was ably demonstrated through an interactive Artificial Neural Network (ANN) simulation.

The contributions to knowledge include a mathematical model which describes the arm gait of an HD patient with a view to form a strong basis for the development of an effective and efficient management mechanism for the ailment.

5.1 Summary of Findings and Contributions to Knowledge

In this section an attempt has been made to present the summary of our findings during the course this study, and to highlight the areas of its possible contribution to knowledge in a concise and precise manner. These are as highlighted in sections 5.1.1 and 5.1.2 above:

5.1.1 Summary of Findings

The following are the highlights of our findings during the course of our study:

- The Nernst equation and the minimum jerk model can be married to develop an electro-mechanical model for the arm gait of an HD patient.
- The mathematical solution to the model based on Frobenius method failed to capture the arm gait of HD
- The involuntary, dance-like movement of the arm of an HD patient can be captured with ANN simulation.

5.1.2 Contributions to Knowledge

A number of new grounds have been broken in this research effort. Above is the summary of our humble contributions to knowledge and engineering practice: the study has indeed:

1. brought to fore a new way of looking at HD
2. established Artificial Neural Network (ANN) modeling as a viable representation of motor function in HD
3. provided the mechanism for coupling the management of HD with ANN modeling

5.1.3 Contributions to Medical Practice

Possible application of the study can readily be found in medical practices the world over. This includes the application of the study to the management and treatment of conditions with similar manifestation such as:

- Parkinson's disease
- Alzheimer's disease
- Disease of Cerebellum
- Epilepsy
- Childbirth conditions such as contractions during labour

To mention but a few

5.2 Conclusion of Research

With the conclusion of this study, a major problem in the field of medicine has been identified, studied and analyzed. This is the chorea in the arm of a Huntington's disease patient for which we have proposed a model for its management based on an interactive artificial neural network (ANN) simulation of the arm gait of the arm of Huntington's disease (HD) patient.

5.3 Recommendations for Further Work

The following are our recommendations for further research effort in this area of study:

- Fuzzy analysis of arm gait of HD patient (to enable us capture the progressive nature/developmental stage of the onset of the disease)
- Identification and modeling of stochastic parameters in choreiform in HD
- Applications of the above to other diseases with similar manifestation as HD

REFERENCES

Abend, W., Bizzi, E., Morasso, P. (1982), Human arm trajectory formation. *Brain* 105: 331-348.

Adeli, H., and Hung, S. (1995); Machine Learning: Neural networks, genetic algorithm and fuzzy systems. *New York: Wiley & Sons Inc., USA.*

Ahissar, E., Sosnik R., and Haidarliu, S. (2000); Transformation from temporal to rate coding in a somatosensory thalamocortical pathway. *Nature* 406: 302- 306.

Akazawa, K. (1994); Modulation and adaptation of mechanical properties of mammalian skeletal muscle.. In: *Clinical Biomechanics and Realted Research,* Hirasawa, Y., Sledge, C.B., and Woo, S.L.Y. (eds.) 217-227.

Atkeson, C.G., and Hollerbach, J.M. (1985); Kinematics features of unrestrained vertical arm movements.. *J. Neurosci.* 5: 2318-2330.

Ayaso, O., and Massaquoi, S.G. (2006); Modelling sensorimotor cortical control of movement. *Artificial Intelligence Laboratory, Massachussetts Institute of Technology* Cambridge, Massachussetts 02139.

Back, T. (1996); Evolutionary algorithms in theory and practice. *Oxford University Press, USA.*

Bashaw, G.T., Kidd, T., Murray, D., Pawson, T., and Goodman, C.S. (2003); Repulsive axon guidance: Abelson and enable play opposing roles downstream of the roundabout receptor. *Cell* 101: 703-715.

Benes, F.M. (2000); Huntington's disease: Hope through research. *National Institute of Neurological disorders and stroke (NINDS)* NY.

Bevan, E. and Paton, C. (2006); What evidence there is for the drug treatment of Huntington's disease. The Pharmaceutical Journal; www.pjonline.com 277: 641-642.

Bleier, N. (2005); Understanding jerk control. *Siemens' MMS Online.*

Bosman, R.J.C., van Leeuwan, W.A., and Wemmenhove, B. (2004); Combining Hebbian and reinforcement learning in aminibrain model. *Neural Networks* 17: 29-36.

Bright, P., Miller, M.R., Franklyn, J.A., and Sheppard, M.C. (1998); The use of a neural network to detect upper airway obstruction caused by goiter. *American Journal of Respiratory and Critical care Medicine* 157(6):1885-1891.

Brinkmeier, H., Aulkemeyer, P., Wollinsky, K.H., and Rudel, R. (2000); An endogenous pentapeptide acting as a sodium channel blocker in inflammatory autoimmune disorder of the central nervous system. *Nat. med.* 6: 808-811.

Brown, L.E. and Rosenbaum, D.A. (2007); Motor control: Models. *Advanced article. Pennsylvania State University, University Park, Pennsylvania, USA.*

Brundin, P., Pogarell, O., Hagell, P., Piccini, P., Widner, H., Schrag, A., Kupsch, A., Crabb, L., Odin, P., Gustavii, B., *et al.* (2000); Bilateral caudate and putamen grafts of embryonic mesencephalic tissue treated with lazaroids in Parkinson's disease. *Brain* 123: 1380-1390.

Bulle, F., Chiannilkulchai, N., Pawlak, A., weissenbach, J., Gyapay, G., and Guellaen, G., (1997); Identification and chromosomal localization of Human genes containing CAG/CTG repeats expressed in testis and brain. *Genome Research.* Vol. 7, No. 7, pp705-715.

Bullock, D., Cisek, P.E., and Grossberg, S. (1995); Cortical networks for control of voluntary arm movements under variable force conditions. *Technical Report* Boston University Center for adaptrive Systems and Department of Cognitive and Neural Systems. CAS/CNS TR-95-019.

Byrne, J.H and Schultz, S.G, (1994); An introduction to membrane transport and bioelectricity (Foundation of General Physiology and Electrochemical Signalling), Raven Press, New York, USA.

Chen., M., Ona, V.O., Li, M., Ferrante, R.J., Fink, K.B. Zhu, S., Bian, J., Guo, L., Farrell, L.A., Hersch, S.M., and Hobbs, W. (2000); Minocycline inhibits caspase-1 and caspase-3 expression and delays mortality in a transgenic mouse of Hunmtington disease. Nat. Med: 6: 797-801.

Clerke, G., Collins, R.A., Leavitt, B.R., Andrew, D.F., Hayden, M.R., Lumsden, C.J., and Mclnnes, R.R. (2000); A one-htt model of cell death in inherited neuronal degenerations. *Nature* 406: 195-199.

Criscimagna-Hemminger, S.E., Donchin, O., Gazzaniga, S., and Shadmehr, R. (2003); Learned dynamics of reaching movements generalize from dominant to nondominant arm. *Journal of Neurophysiology*, The American Physiological Society 89:168-176.

De Zazzo, J., Sandstron, D., de Belle, S., Velinzon, K., Smith, P., Grady, L., Vecchio, M., Ramaswani, M., and Tully, T. (2000); nalyot, a mutation of the Drosophila Myb-related *Adf1* transcription factor, disrupts synapse formation and olfactory memory. *Neuron* 27: 145-158.

Dingwell, J.B., Mah, C.D., and Mussa-Ivaldi (2004); Experimentally confirmed mathematical model for human control of a non-rigid object. *Journal of Neurophysiology* 91: 1158-1170.

Duke University (2002); Researchers discover "doorways" into brain cells. *ScienceDaily.*

Flash, T. (1987); The control of hand equilibrium trajectories in multi-joint arm movements. *Biol. Cybernetics.* 5: 257-274.

Flash, T. (1991); The organization of human arm trajectory control. In: *Multiple Muscle Systems.* Winter, J.M., Woo, S.L.-Y. New York: Springer-Veerlag: 282-301.

Flash, T., and Hogan, N. (1985); The coordinate of arm movements: an experimentally confirmed mathematical model. *Journal of Neuroscience* 5: 1688-1703.

Flash, T., and Sejnowski, T.J. (2001); Computational approaches to motor control. *Current Opinion in Neurobiology* 11:655-662.

Frazin, N. and Clipper, S., (2004); Study using robotic microscope shows how mutant Huntington's disease protein affects neurons. *NIH News, U.S. Department of Health and Human Services.*

Georgopoulos, A.P., Kalaska, J.F., Caminiti, R., and Massey, J.T. (1982); On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. *Journal of Neurophysiology* 60: 1874-1895.

Gardian, G., and Vecsei, L. (2004); Huntington's disease: pathomechanism and therapeutic perspectives. *Journal of Neural transmission* 111(10-11): 1485-1494.

Gray, J.M., Young, A.W, Barker, W.A, Curtis, A. and Gibson, D. (1997); Impaired recognition of disgust in Huntington's disease gene carrier. *Brain* 120:2029-2038.

Greenfield, S. (1999); The private life of the brain. *Penguin Books*

Gusella, J.F., Wexler, N.S., Conneally, P.M., Naylor, S.L., Anderson, M.A., Tanzi, R.E. et al (1983); A polymorphic DNA marker genetically linked to Huntington's disease. *Nature.* Vol. 306, pp234-238.

Guyton, A.C and Hall, J.E (1996); Textbook of medical physiology. *W.B.Saunders Company (A division of Harcourt Brace and Company),* USA.

Haggard, P., and Richardson, J. (1996); Spatial patterns in the control of human arm movement. *Journal of Experimental Psychology* 22: 42-62.

Hollerbach, J.M. (1990); Planning of arm movement. In: *Visual Cognition and Action,* Osherson, D.N., Kosslyn, S.M., and Hollerbach, J.M. (eds.). Cambridge, MA: MIT Press: 183-211.

Huntington's disease Collaborative research Group (1993); A novel gene containing a trinucleotide repeat that is expanded and unstable on Huntington's disease chromosomes. *Cell.* Vol. 72, pp 971-983.

Huntington, G. (1872); On chorea. *The Medical and Surgical Reporter: A weekly Journal* (Philadelphia, S.W. Butler) Vol. 26, No. 15: pp 117-132.

.Jasmine, L., (2005); Fine motor control. *Medline Plus Medical encyclopedia.*

Jean-Christophe, N. (2000); Realistic collision avoidance of upper limbs based on neuroscience models. *EUROGRAPHICS '2000* 19: Number.

Jen-Zen, C., Hui, Z., Shi-Hua, L., Xiao-Jiang, l., and Ching-Hwa, S. (2002);'Characterization of a brain-enriched chaperone, MRL, that inhibits huntingtin aggregation and toxicity independently. *Journal of Biological chemistry* 277(22): 19831-19838.

Ji-Yeon, S., Zhi-Hui, F., Zhao-Xue, Y., Chuan-En, W., Shi-Hua, L., and Xiao-Jang, L. (2005); Expression of mutant huntingtin in glial cells contributes to neuronal excitoxicity. *The Rockfeller University press.* 171(8): 1001-1012.

Katayama, M., and Kawato, M. (1993); Virtual trajectory and stiffness ellipse during multijoint arm movement predicted by neural inverse models. *Biol. Cybernetics* 69: 353-362.

Kawato, M. (1996); Optimization and Learning in neural networks for formation and control of coordinated movement. In: *Attention and Performance XIV*, Meyer D.E., Kornblum, S. Cambridge, MA: MIT Press, 821-849.

Koiko, Y., and Kawato, M. (1995); Estimation of dynamic joint torques and trajectory formation from surface electromyography signals using a neural network model. *Boil. Cybernetics* 73: 291-300.

Kopp, P., and Jameson, J.L. (1998); Transmission of disease. *Principles of MolecularMedicine.* (J.L. Jameson. ed.) Humana Press Inc., Totowa, NJ.

Kreyzig, Edwin (1988); Advanced Engineering Mathematics. *John Wiley & Sons Incorporation.*

Luthi-Carter, R., Strand, A., Peters, N.L., Solano, S.M., Hollingsworth, Z.R., Menon, A.S., Frey, A.S., Spektor, B.S., Penney, E.B., Schilling, G., Ross, C.A., Borchelt, D.R., Tapscott, S.J., Young, A.B., Cha, J-H.J., and Olson, J.M. (2000); Decreased expression of striatal signaling genes in a mouse model of Huntington's disease. *Human Molecular Genetics* 9(9): 1259-1271.

Mantell, C.L. (1960); Electrochemical engineering. McGraw-Hill Book Company, New York, USA.

Margolis, R.L., and O'Hearn E, Rosenblatt, A., Willour, V., Holmes S.E., Franz, M.L., et al (2001); A disorder similar to Huntington's disease is associated with a novel CAG repeat expansion. *Ann neurology.* Vol. 50, pp 373-380.

Margolis, R.L., and Ross, C.A. (2003); Diagnosis of Huntington disease. *Clinical Chemistry.* Vol. 49, No. 10, pp 1726-1732.

Margeta-Mitrovic, M., and Jan, Y.N. (2000); A trafficking checkpoint controls GABA(B) receptor heterodimerization. *Neuron* 27: 97-106.

Maschke, M., Tuite, P.J., Pickett, K., Wachter, T., and Konczak, J. (2005); The effect of subthalamic nucleus stimulation on kinaesthesia in Parkinson's disease. *Journal of Neurology, Neurosurgeon and Psychiatry* 76:569-571.

Merriam-Webster, (1996); Merriam-Webster's medical desk dictionary. *Merriam-Webster incorporate, USA..*

Michalewicz, Z. (1996); Evolutionary computation: Practical issues. In *Proceedings 1996 International Conference on Evolutionary Computation*: 30-39.

Miyoshi, S., Yanai, H.-F., and Okada, M. (2004); Associated memory by recurrent neural networks with delay elements. *Neural networks* 17: 55-64.

Morasso, P. (1981 Spatial control of arm movements. *Experimentation Brain Research* 42: 223-227.

Nakano E, Hiroshi I., Rieko, O., Yoji, U., Hiropaki, G., Toshinori, Y., and Mitsuo, K. (1999); Quantitative examinations of internal representations for arm trajectory planning: minimum commanded torque change model. *The Journal of Neurology*, Vol. 81 No.5, pp:2140-21565.

NINDS (2000); Huntington's disease: Hope through research. *National Institute of Neurological Disorders and Stroke.*

Olubunmo, A.O. (1978); Real Analysis. *Ibadan University Press Limited, Ibadan, Nigeria.*

Peterson, S.P. (2006); Mutation. *Wikipedia, the free encuclopedia* Wikimedia Foundation, Inc. 81.227.135.26.

Piazzi, A. (2000); Glogal minimum-jerk trajectory planning of robot manipulators. *IEEE Transactions on Industrial Electronics.* Vol. 47, No. 1: 140-149.

Ramsey, M. (2004); GA Optimizer. *Artificial Intelligence Lab, University of Arizona, USA.*

Roberts, M., Whitley, L.D., Howe, A.E., Barbulescu, L. (2005); Random walks and neibourhood bias in oversubscribed scheduling. *Air Force Office of Science research, Air Force Materiel Command, USAF,* under grant number F49620-03-1-0233.

Richardson, M.J.E., and Flash, T. (2002); Comparing smooth arm movements with two thirds power law and the related segmented-control hypothesis. *The journal of Neuroscience* 22(18) 8201-8211.

Roberts, J.S., Cupples, A., Relkin, N. R., Whitehouse, P.J., and Green, R.C. (2005); genetic risk assessment for adult children of people with alzheimer's disease: The risk evaluation and education for alzheimer's disease (REVEAL) study. *Sage Publications.*

Ross, C.A., Becher, M.W., Colomer, V, Engelender, S., Wood, J.D., Sharp, A.H. (1997); huntington's disease and dentatorubral pallidoluysian atrophy; proteins, pathogenesis, and pathology. *Brain pathology.* Vol. 7, pp 1003-1016.

Schaneggenburger, R., Neher, E. (2000); Intercellular calcium dependence of transmitter release rates at a fast central synapse. *Nature* 406: 889-893.

Scheidt, R.A., Reinkensmeyer, D.J., and Conditt, M.A. (2000); Persistence of motor adaptation during constrained, multi-joint, arm movement. *Journal of Neurophysiology* 84: 853-862.

Shadmehr, R., and Mussa-Ivaldi, F.A. (1994); Adaptive representation of dynamics during learning of a motor task. *Journal of Neuroscience* 14: 3208-3224.

Shadmehr, R., and Moussavi, Z.M.K. (2000); Spatial generalization from learning dynamics of reaching movements. *The Journal of Neuroscience* 20(20): 7807-7815.

Shi-Hua, L., and Xiao-Juang, L. (2004); Huntingtin and its role in neuronal degeneration. *The Neuroscientist* 10:467-475.

Shi-Hua, L., and Xiao-Juang, L. (2004); Huntingtin-protein interactions and the pathogenesis of Huntington's disease. *TRENDS in Genetics, ELSEVIER Vol.*20 No.3.

Smith, M.A., and Shadmehr, R., (2005); Inability to learn internal models of arm dynamics in Huntington's disease but not cerebellar degeneration. *Journal of Neurophysiology* 93: 2809-2821.

Squitierri, F., Gellera, C., Canella, M., Mariotti, C., Cislaghi, G., Rubinsztein, C.D., Almqvist, W.E., Turner, D., Bachoud-Levi, A., Simpson, A.S., Delatycki, M., Maglione, V., Hayden, R.M., and Donato, D.S. (2003); Homozygosity for CAG mutation in Huntington disease is associated with a more severe clinical course. *Brain* 126(4): 946-955.

Stonier, R., and Mohammadian, M. (1996); Evolutioary learning in fuzzy logic control systems. In *Complexity International* 3.

Tassicker, R. Savulescu, J. *et al.* (2003); Prenatal diagnosis requests for Huntington's disease when the father is at risk and does not want to know his gene status: clinical, legal and ethical viewpoints. *Embryo Journal* 22(30): 355-361.

University of Texas Southwestern Medical Center At Dallas (2005); Drug treatment promising for halting Huntington's-related nerve death. *ScienceDaily*. Retrieved February 7, 2008, from http://www.sciencedaily.com/release/2005/01/050131223723.htm.

Uno, Y., Imamizu, H., Gomi, H., and Kawato, M. (1995); Space where arm trajectory is planned: evidence from experiments in altered dynamics. In: *Fourth Annual Meeting of IBRO World Congress of Neuroscience Society.* D6.33, 343.

Uno, Y., and Kawato, M. (1996); The trajectory of human arm movements depend on movement duration. In: *Eleventh Symposium on Biological and Physiological Engineering* (in Japanese)., Osaka, Japan: SICE, 329-332.

Weisstein, E.W. (1999-2005); "Acceleration". *Mathworld. A Wolfram Web resource.* Wolfram Research, Inc.

Wikipedia, the free encyclopedia (2005); Jerk. *GNU Free Documentation Licence.*

Williamson, M.M. (1998); Neural control of rhythmic arm movements. *MIT AI Lab,* Cambridge MA 02139.

Whitley, D., and Rowe, J., (2005); Gray, binary and real valued encodings: quad search and locality proofs.

Worpert, D.M., Ghahramani, Z., and Jordan, M. I. (1994); Perceptual distortion contributes to the curvature of human reaching movements. *Exp. Brain research* 98: 153-156.

Xu, Z.-B., Qiao, H., Peng, J., and Zhang, B. (2004); A comparative study of two modeling approaches in neural networks. *Neural Networks* 17: 73-86.

Ye B., Liao, D., Zhang, X, Zhang, P., Dong, H., and Huganir, R.L. (2000); GRASP-1: a heuronal RasGEF associated with the AMPA receptor/GRIP complex. *Neuron* 3: 603-617.

Zehr, E.P., Collins, D.F., Frigon A., and Hongenboom, N. (2003); Neural control of rhythmic human arm movement: phase dependence and task modulation of Hoffmann reflexes in forearm muscles. *J. Neurophysiology* 89: 12-21.

Zhao, H. (2004); Global asymptotic stability of Hopfield neural network involving distributed delays. *Neural Networks* 17: 47-54.

Zhou, J., Liu, G., and Chen, G. (2004); Dynamics of periodic delayed neural networks. *Neural Networks* 17: 87-102.

APPENDICES

APPENDIX A

TABLE A-1: The plot of displacement against time for the Crisp Model

Time (t)	ς(t)	sin(πt/4)	A(0)	A(1)	A(2)	A(3)	A(4)	A(5)	X(t)
0.0	3.32143	0.0000000	0.7238	0.00000	0.000000	0.0000000	0.00000000	0.000000000	0.00000000
0.1	3.32143	0.078459279	0.7238	0.01651	0.000825	0.0000129	0.00000032	0.000000004	0.19314101
0.2	3.32143	0.156434828	0.7238	0.03302	0.003300	0.0001032	0.00000512	0.000000128	0.39500507
0.3	3.32143	0.2334459	0.7238	0.04953	0.007425	0.0003483	0.00002592	0.000000972	0.60566821
0.4	3.32143	0.309017693	0.7238	0.06604	0.013200	0.0008256	0.00008192	0.000004096	0.82516037
0.5	3.32143	0.382684281	0.7238	0.08255	0.020625	0.0016125	0.00020000	0.000012500	1.05345374
0.6	3.32143	0.453991482	0.7238	0.09906	0.029700	0.0027864	0.00041472	0.000031104	1.29044989
0.7	3.32143	0.5224997	0.7238	0.11557	0.040425	0.0044247	0.00076832	0.000067228	1.53596563
0.8	3.32143	0.587786441	0.7238	0.13208	0.052800	0.0066048	0.00131072	0.000131072	1.78971755
0.9	3.32143	0.649449305	0.7238	0.14859	0.066825	0.0094041	0.00209952	0.000236196	2.05130502
1.0	3.32143	0.70710808	0.7238	0.16510	0.082500	0.0129000	0.00320000	0.000400000	2.32019181
1.1	3.32143	0.760407278	0.7238	0.18161	0.099825	0.0171699	0.00468512	0.000644204	2.59568620
1.2	3.32143	0.80901829	0.7238	0.19812	0.118800	0.0222912	0.00663552	0.000995328	2.87691970
1.3	3.32143	0.852641412	0.7238	0.21463	0.139425	0.0283413	0.00913952	0.001485172	3.16282450
1.4	3.32143	0.891007692	0.7238	0.23114	0.161700	0.0353976	0.01229312	0.002151296	3.45210983
1.5	3.32143	0.923880587	0.7238	0.24765	0.185625	0.0435375	0.01620000	0.003037500	3.74323744
1.6	3.32143	0.951057424	0.7238	0.26416	0.211200	0.0528384	0.02097152	0.004194304	4.03439660
1.7	3.32143	0.972370649	0.7238	0.28067	0.238425	0.0633777	0.02672672	0.005679428	4.32347893
1.8	3.32143	0.987688858	0.7238	0.29718	0.267300	0.0752328	0.03359232	0.007558272	4.60805361
1.9	3.32143	0.996917608	0.7238	0.31369	0.297825	0.0884811	0.04170272	0.009904396	4.88534340
2.0	3.32143	1.000000	0.7238	0.33020	0.330000	0.1032000	0.05120000	0.012800000	5.15220222
2.1	3.32143	0.996917031	0.7238	0.34671	0.363825	0.1194669	0.06223392	0.016336404	5.40509480
2.2	3.32143	0.987687709	0.7238	0.36322	0.399300	0.1373592	0.07496192	0.020614528	5.64007933
2.3	3.32143	0.972368934	0.7238	0.37973	0.436425	0.1569543	0.08954912	0.025745372	5.85279367
2.4	3.32143	0.951055154	0.7238	0.39624	0.475200	0.1783296	0.10616832	0.031850496	6.03844615
2.5	3.32143	0.923877775	0.7238	0.41275	0.515625	0.2015625	0.12500000	0.039062500	6.19181172
2.6	3.32143	0.891004356	0.7238	0.42926	0.557700	0.2267304	0.14623232	0.047525504	6.30723432
2.7	3.32143	0.852637573	0.7238	0.44577	0.601425	0.2539107	0.17006112	0.057395628	6.37863643
2.8	3.32143	0.809013972	0.7238	0.46228	0.646800	0.2831808	0.19668992	0.068841472	6.39953655

Time (t)	**ς(t)**	**sin(πt/4)**	**A(0)**	**A(1)**	**A(2)**	**A(3)**	**A(4)**	**A(5)**	**X(t)**
2.9	3.32143	0.760402507	0.7238	0.47879	0.693825	0.3146181	0.22632992	0.082044596	6.36307558
3.0	3.32143	0.707102885	0.7238	0.49530	0.742500	0.3483000	0.2592000	0.097200000	6.26205281
3.1	3.32143	0.649443719	0.7238	0.51181	0.792825	0.3843039	0.29552672	0.114516604	6.08897231
3.2	3.32143	0.587780498	0.7238	0.52832	0.844800	0.4227072	0.33554432	0.134217728	5.83610026
3.3	3.32143	0.522493397	0.7238	0.54483	0.898425	0.4635873	0.37949472	0.156541572	5.49553397
3.4	3.32143	0.453984936	0.7238	0.56134	0.953700	0.5070216	0.42762752	0.181741696	5.05928271
3.5	3.32143	0.382677494	0.7238	0.57785	1.010625	0.5530875	0.48020000	0.210087500	4.51936096
3.6	3.32143	0.309010706	0.7238	0.59436	1.069200	0.6018624	0.53747712	0.241864704	3.86789389
3.7	3.32143	0.233438756	0.7238	0.61087	1.129425	0.6534237	0.59973152	0.277375828	3.09723526
3.8	3.32143	0.156427572	0.7238	0.62738	1.191300	0.7078488	0.66724352	0.316940672	2.20009725
3.9	3.32143	0.078451955	0.7238	0.64389	1.254825	0.7652151	0.74030112	0.360896796	1.16969199
4.0	3.32143	-7.3464E-06	0.7238	0.66040	1.320000	0.8256000	0.81920000	0.409600000	-0.00011611

Table A-1 continued

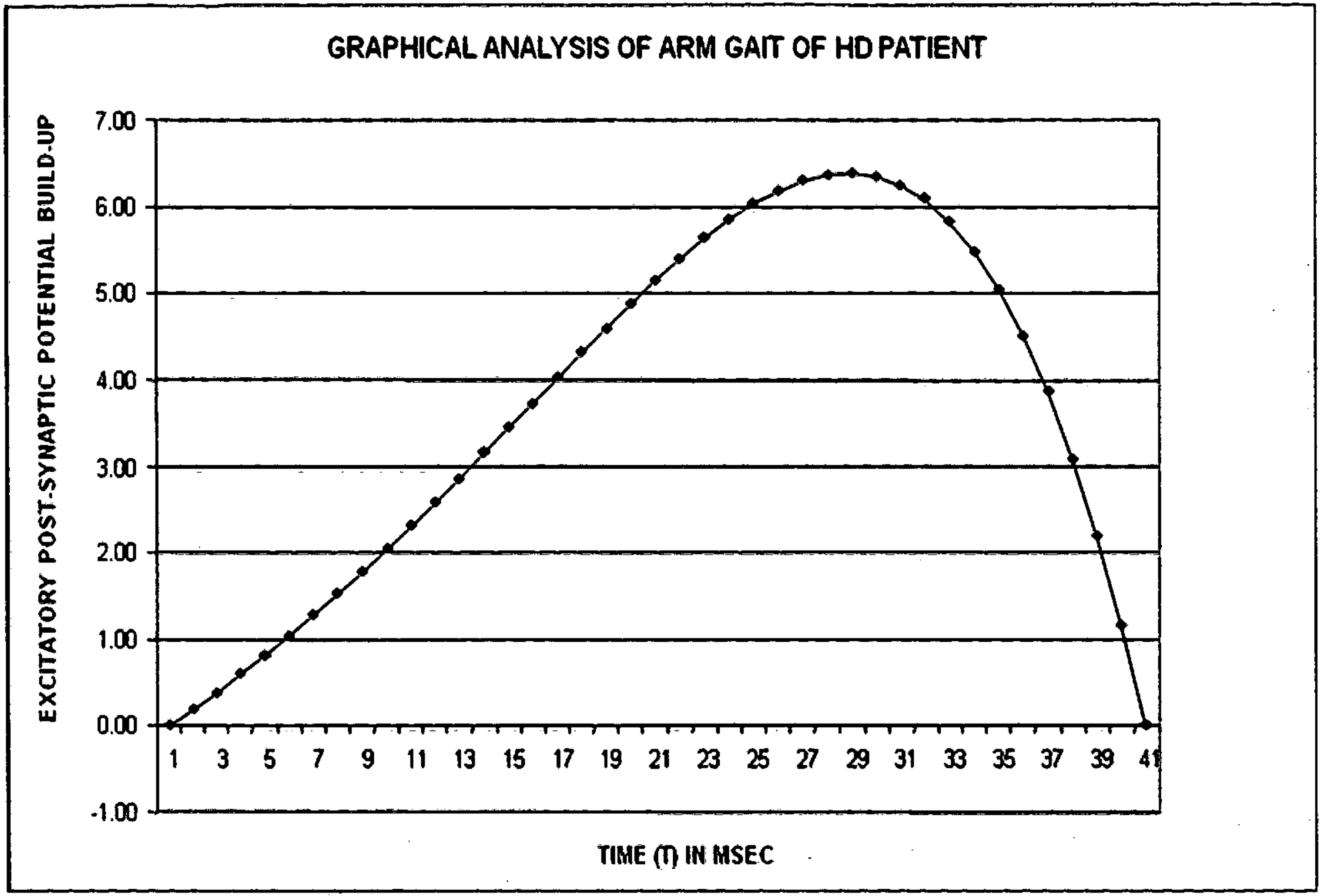

Figure A-1: The graph of displacement against time for the Crisp Model

APPENDIX A

EPOCH	CRISP	ANN
0	0	0
10	0.154582	-0.06304
20	0.308486	-0.05758
30	0.461202	0.021569
40	0.612217	0.136378
50	0.761006	0.275615
60	0.907089	0.435387
70	1.049966	0.584056
80	1.189172	0.735779
90	1.324243	0.891097
100	1.45474	1.047544
110	1.580243	1.204307
120	1.700331	1.357654
130	1.814628	1.508385
140	1.922773	1.654963
150	2.024421	1.795944
160	2.119241	1.930563
170	2.206961	2.058157
180	2.287297	2.177237
190	2.360008	2.287022
200	2.424899	2.386293
210	2.481789	2.473638
220	2.530528	2.548048
230	2.571012	2.607585
240	2.603179	2.650445
250	2.626985	2.67428
260	2.64243	2.67565
270	2.649559	2.649545
280	2.648462	2.590686
290	2.639275	2.709225
300	2.622163	3.239446
310	2.597351	3.34505
320	2.565126	3.363036
330	2.525803	3.223094

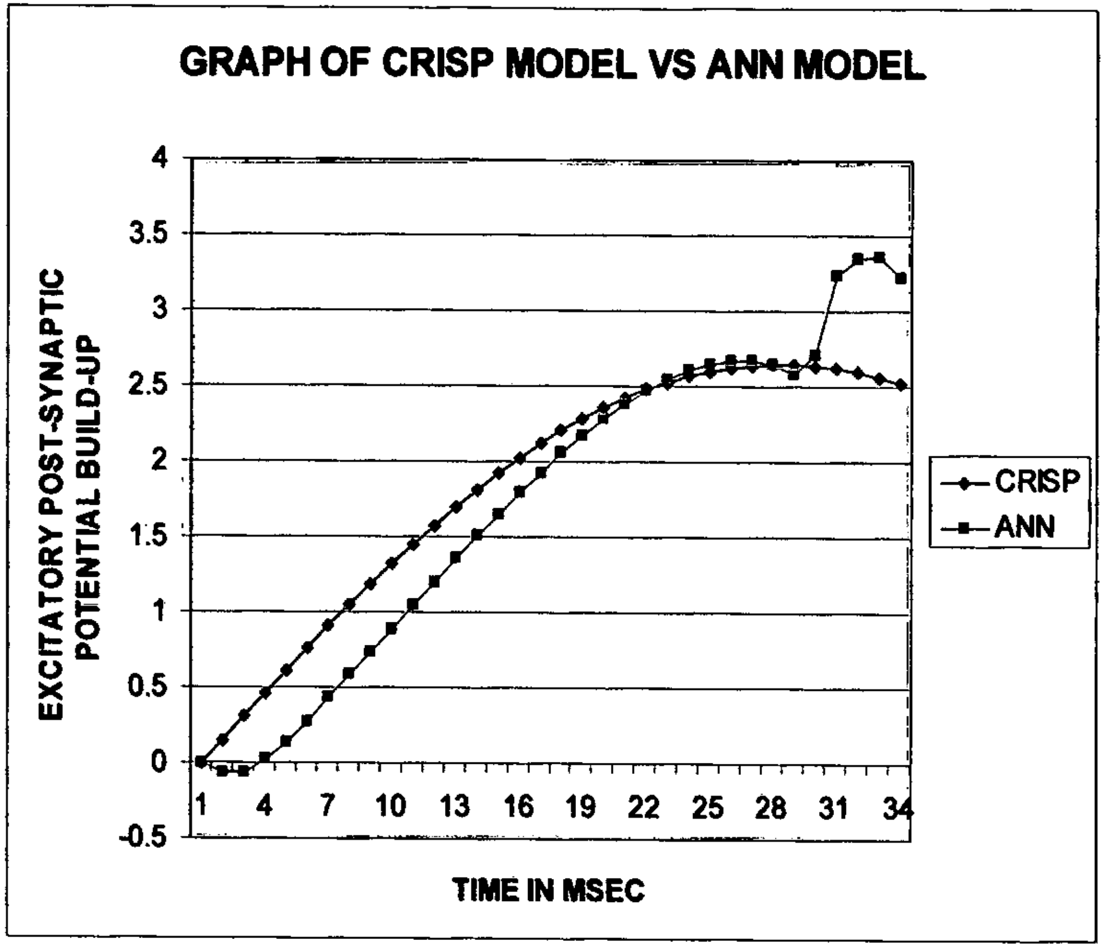

FIGURE A-2: Comparative studies of Crisp versus ANN models in the representation of arm gait of HD patient

EPOCH	TRAINING DATA			TEST DATA		
	SSE	MSE	SSE/O-UNITS	SSE	MSE	SSE/O-UNITS
10	9.68830	0.00181	9.68830	8.84850	0.00165	8.84850
20	9.62162	0.00179	9.62162	9.15772	0.00171	9.15772
30	9.73212	0.00182	9.73212	9.91758	0.00185	9.91758
40	10.26644	0.00192	10.26644	9.38853	0.00175	9.38853
50	10.37286	0.00193	10.37286	9.41039	0.00176	9.41039
60	10.22872	0.00191	10.22872	9.62538	0.00180	9.62538
70	10.31495	0.00192	10.31495	9.86351	0.00184	9.86351
80	10.72408	0.00200	10.72408	10.46962	0.00195	10.46962
90	10.85954	0.00203	10.85954	10.45353	0.00195	10.45353
100	11.03667	0.00206	11.03667	10.39531	0.00194	10.39531
110	11.19287	0.00209	11.19287	13.33786	0.00249	13.33786
120	11.44877	0.00214	11.44877	12.21618	0.00228	12.21618
130	11.94615	0.00223	11.94615	10.75449	0.00201	10.75449
140	12.15728	0.00227	12.15728	11.06835	0.00206	11.06835
150	12.05269	0.00225	12.05269	13.93886	0.0026	13.93886
160	12.27877	0.00229	12.27877	12.20879	0.00228	12.20879
170	12.75445	0.00238	12.75445	11.8591	0.00221	11.8591
180	12.25608	0.00229	12.25608	12.16397	0.00227	12.16397
190	13.26925	0.00248	13.26925	12.35164	0.0023	12.35164
200	13.79329	0.00257	13.79329	12.62475	0.00235	12.62475
210	13.77107	0.00257	13.77107	13.32772	0.00249	13.32772
220	13.25124	0.00247	13.25124	14.56657	0.00272	14.56657
230	14.55394	0.00271	14.55394	13.80332	0.00257	13.80332
240	15.01160	0.00280	15.01160	14.25410	0.00266	14.2541
250	15.58682	0.00291	15.58682	15.3598	0.00287	15.3598
260	16.04093	0.00299	16.04093	15.55503	0.0029	15.55503
270	17.02671	0.00318	17.02671	16.01175	0.00299	16.01175
280	17.61792	0.00329	17.61792	16.56122	0.00309	16.56122
290	18.38665	0.00343	18.38665	17.33072	0.00323	17.33072
300	18.90159	0.00353	18.90159	19.68319	0.00367	19.68319
310	20.27844	0.00378	20.27844	20.35651	0.00380	20.35651
320	21.32877	0.00398	21.32877	21.09393	0.00393	21.09393
330	22.94093	0.00428	22.94093	22.58194	0.00421	22.58194
340	25.16621	0.00469	25.16621	23.79496	0.00444	23.79496
350	27.43690	0.00512	27.43690	26.10738	0.00487	26.10738
360	30.67787	0.00572	30.67787	29.28762	0.00546	29.28762
370	35.23222	0.00657	35.23222	33.97247	0.00634	33.97247
380	43.22889	0.00806	43.22889	41.92018	0.00782	41.92018
390	61.74262	0.01152	61.74262	62.60379	0.01168	62.60379

TABLE A-3: Comparative analysis of Sum square error, Mean Square Error of Training data versus Test Data at various epoch values for the training of the ANN using Stuttgart Neural Networks Simulator (SNNS).

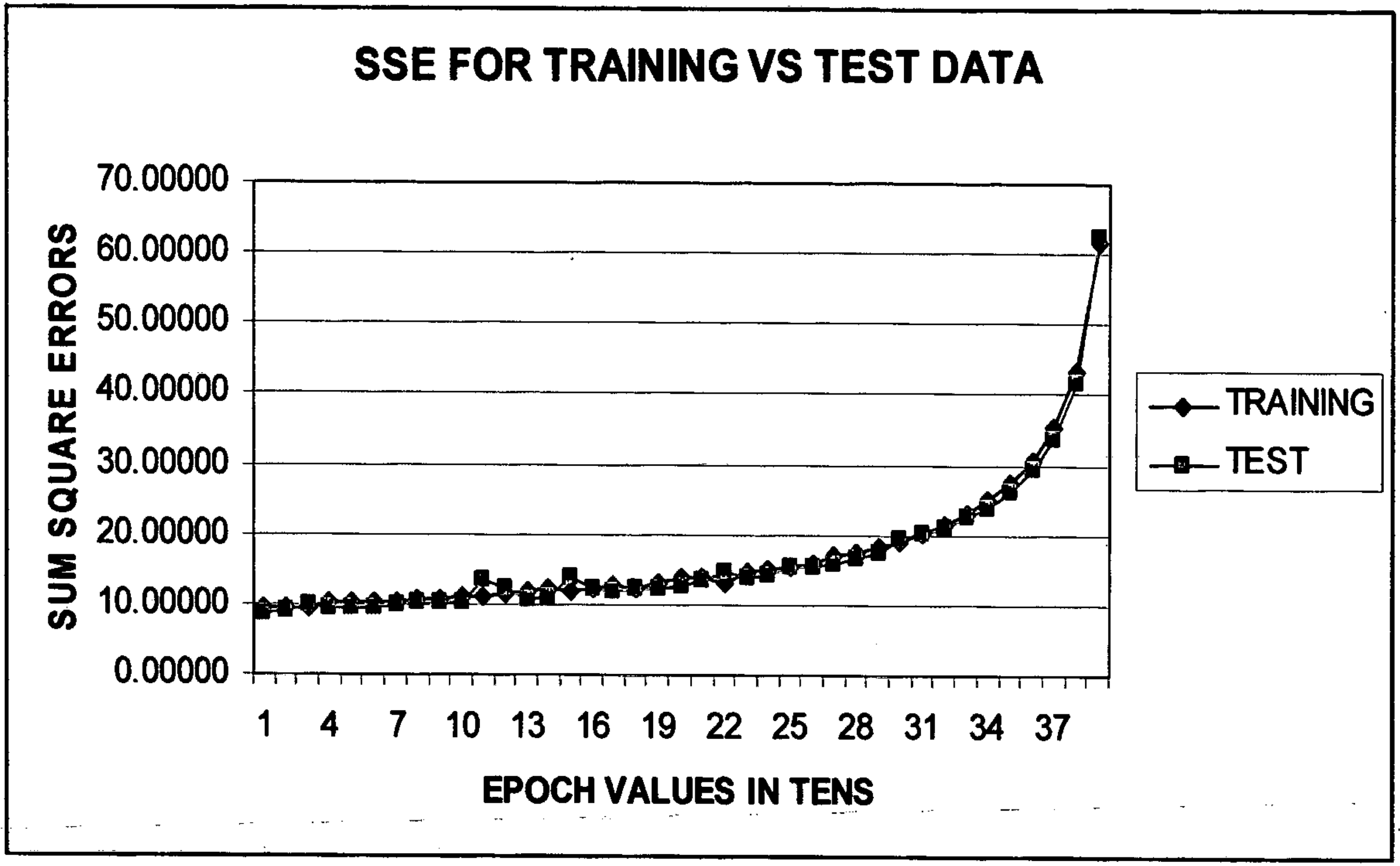

Figure A-3: Graphical analysis of Sum Square Error of Training data versus Test Data at various epoch values for the training session for the ANN using SNNS.

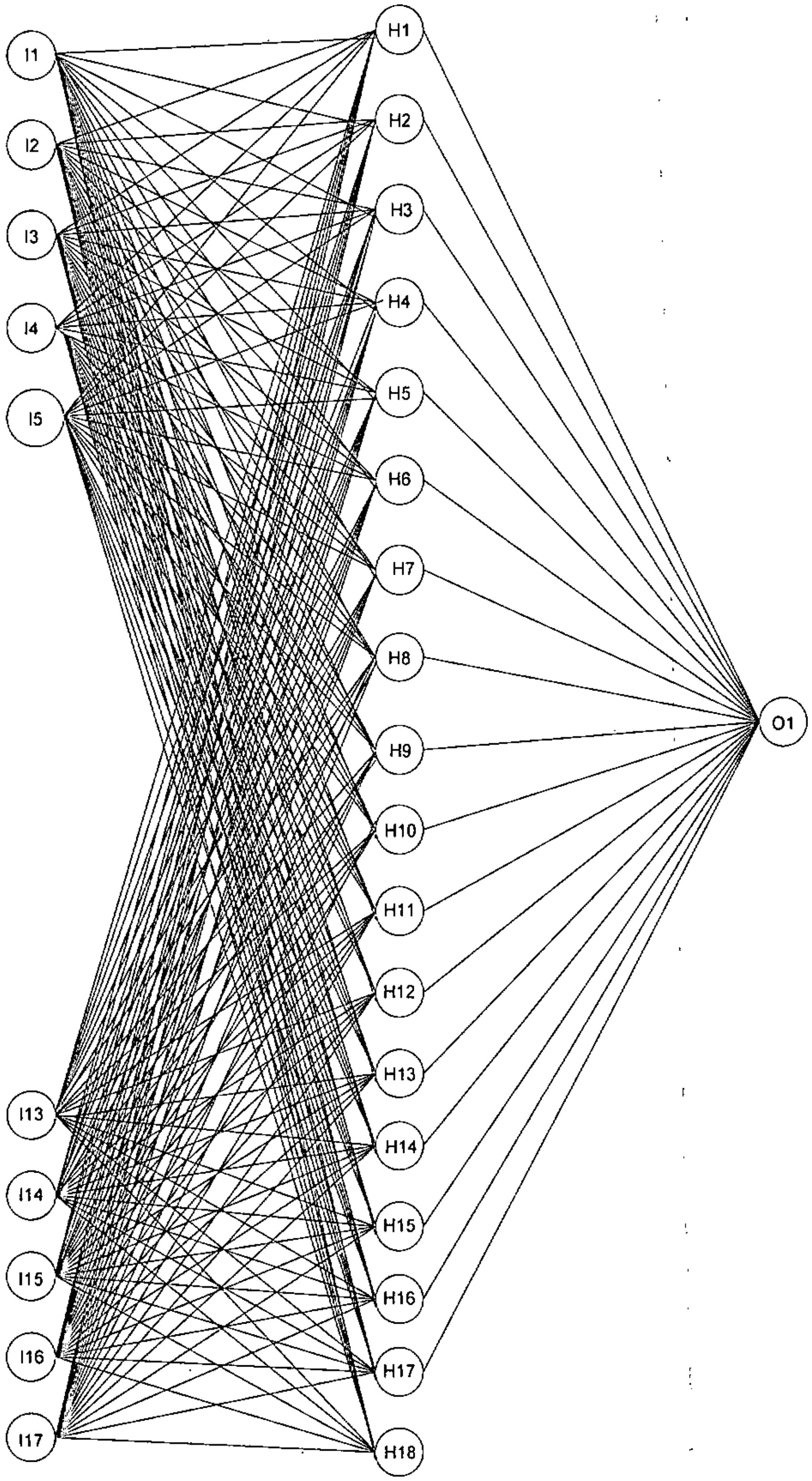

Figure A-4: The architecture for the Artificial Neural Network simulation

Appendix A

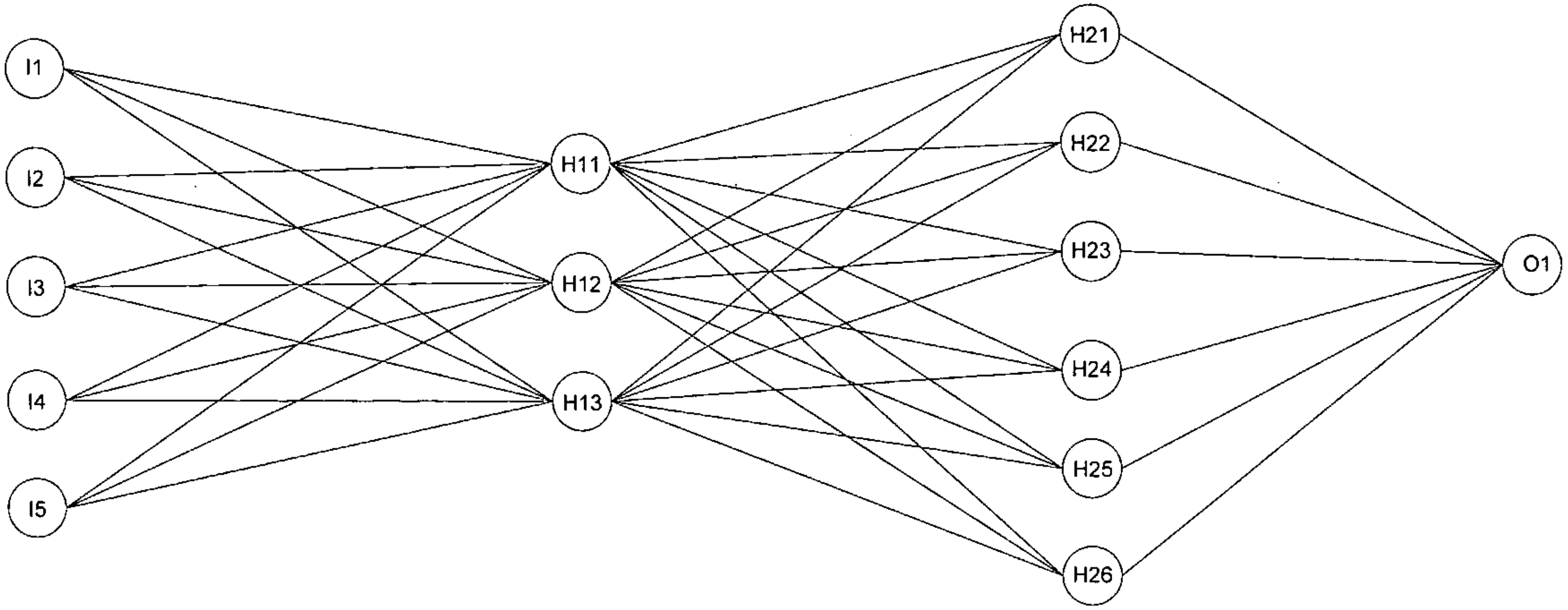

Figure A-5: The Artificial Neural Network for the SSE for Training versus Test Data

APPENDIX B

0.0000000	1.0000000	0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000000
0.1564348	0.9876883	0.3760773	0.08508	0.01661	0.00016	3.2E-07	0.00000	0.47793297
0.3090177	0.9510563	0.7428943	0.16384	0.06564	0.00126	5.12E-06	0.00000	0.97364847
0.4539915	0.8910060	1.0914187	0.23025	0.14465	0.00400	2.592E-05	0.00003	1.47036547
0.5877864	0.8090161	1.4130686	0.27875	0.24970	0.00860	8.192E-05	0.00013	1.95032757
0.7071081	0.7071055	1.6999239	0.30454	0.37549	0.01468	0.00020000	0.00040	2.39523045
0.8090183	0.5877835	1.9449213	0.30378	0.51553	0.02108	0.0004147	0.00099	2.78671638
0.8910077	0.4539882	2.142028	0.27374	0.66240	0.02586	0.0007683	0.00214	3.10693070
0.9510574	0.3090142	2.2863906	0.21294	0.80805	0.02627	0.0013107	0.00416	3.33913110
0.9876889	0.1564312	2.3744544	0.12127	0.94407	0.01894	0.0020995	0.00750	3.46833611
1	-0.0000037	2.404051	0.00000	1.06204	0.00000	0.00320000	0.01271	3.48199479
0.9876877	-0.1564385	2.3744517	-0.14823	1.15386	-0.03458	0.0046851	0.02046	3.37065592
0.9510552	-0.3090212	2.2863851	-0.31942	1.21207	-0.08868	0.0066355	0.03162	3.12861153
0.8910044	-0.4539948	2.1420199	-0.50838	1.23017	-0.16564	0.0091395	0.04717	2.75448724
0.809014	-0.5877894	1.9449109	-0.70883	1.20289	-0.26784	0.0122931	0.06833	2.25175038
0.7071029	-0.7071107	1.6999114	-0.91363	1.12646	-0.39631	0.01620000	0.09648	1.62910666
0.5877805	-0.8090204	1.4130543	-1.11499	0.99880	-0.55030	0.0209715	0.13323	0.90075740
0.4539849	-0.8910094	1.091403	-1.30474	0.81966	-0.72695	0.0267267	0.18040	0.08649142
0.3090107	-0.9510586	0.7428775	-1.47460	0.59073	-0.92109	0.0335923	0.24008	-0.78840900
0.1564276	-0.9876894	0.3760599	-1.61647	0.31565	-1.12502	0.0417027	0.31460	-1.69347007
-0.000007	-1.0000000	-1.766E-05	-1.72275	-0.00002	-1.32852	0.05120000	0.40657	-2.59353064
-0.1564421	-0.9876871	-0.3760948	-1.78662	-0.34891	-1.51899	0.0622339	0.51890	-3.44947671
-0.3090247	-0.9510540	-0.7429111	-1.80227	-0.72203	-1.68171	0.0749619	0.65479	-4.21917466
-0.453998	-0.8910027	-1.0914344	-1.76522	-1.10898	-1.80028	0.0895491	0.81777	-4.85860365
-0.5877924	-0.8090118	-1.4130829	-1.67247	-1.49822	-1.85723	0.1061683	1.01169	-5.32315801
-0.7071133	-0.7071003	-1.6999364	-1.52270	-1.87746	-1.83476	0.12500000	1.24077	-5.56908738
-0.8090226	-0.5877775	-1.9449316	-1.31637	-2.23396	-1.71558	0.1462323	1.50958	-5.55503298
-0.891011	-0.4539817	-2.142036	-1.05583	-2.55499	-1.48391	0.1700611	1.82309	-5.24361121
-0.9510597	-0.3090072	-2.286396	-0.74528	-2.82818	-1.12647	0.1966899	2.18665	-4.60298902
-0.98769	-0.1564239	-2.3744572	-0.39075	-3.04201	-0.63354	0.2263299	2.60603	-3.60839146
-1	0.0000110	-2.404051	0.00003	-3.18613	0.00005	0.2592000	3.08742	-2.24347859
-0.9876866	0.1564457	-2.3744489	0.41775	-3.25179	0.77397	0.2955267	3.63746	-0.50152876
-0.9510529	0.3090282	-2.2863797	0.85181	-3.23219	1.68161	0.3355443	4.26324	1.61363286
-0.891001	0.4540013	-2.1420119	1.29052	-3.12273	2.70942	0.3794947	4.97232	4.08701758
-0.8090097	0.5877954	-1.9449005	1.72146	-2.92129	3.83655	0.4276275	5.77277	6.89221833

APPENDIX B

0.0000000	1.0000000	0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000000
0.0784593	0.9969173	0.1886201	0.04294	0.00104	0.00002	3.2E-07	0.00000	0.23261878
0.1564348	0.9876883	0.3760773	0.08508	0.00415	0.00016	5.12E-06	0.00000	0.46547724
0.2334459	0.9723698	0.5612159	0.12564	0.00930	0.00054	2.592E-05	0.00000	0.69672156
0.3090177	0.9510563	0.7428943	0.16384	0.01641	0.00126	8.192E-05	0.00000	0.92449680
0.3826843	0.9238792	0.9199925	0.19895	0.02540	0.00240	0.00020000	0.00001	1.14695585
0.4539915	0.8910060	1.0914187	0.23025	0.03616	0.00400	0.0004147	0.00003	1.36226871
0.5224997	0.8526395	1.2561159	0.25706	0.04856	0.00607	0.0007683	0.00007	1.56863222
0.5877864	0.8090161	1.4130686	0.27875	0.06243	0.00860	0.0013107	0.00013	1.76428019
0.6494493	0.7604049	1.5613093	0.29475	0.07760	0.01151	0.0020995	0.00023	1.94749396
0.7071081	0.7071055	1.6999239	0.30454	0.09387	0.01468	0.00320000	0.00040	2.11661350
0.7604073	0.6494465	1.8280579	0.30768	0.11104	0.01794	0.0046851	0.00064	2.27004883
0.8090183	0.5877835	1.9449213	0.30378	0.12888	0.02108	0.0066355	0.00099	2.40629202
0.8526414	0.5224965	2.0497935	0.29254	0.14715	0.02383	0.0091395	0.00147	2.52392943
0.8910077	0.4539882	2.142028	0.27374	0.16560	0.02586	0.0122931	0.00214	2.62165447
0.9238806	0.3826809	2.2210561	0.24722	0.18398	0.02681	0.01620000	0.00302	2.69828050
0.9510574	0.3090142	2.2863906	0.21294	0.20201	0.02627	0.0209715	0.00416	2.75275409
0.9723706	0.2334423	2.3376287	0.17092	0.21945	0.02381	0.0267267	0.00564	2.78416830
0.9876889	0.1564312	2.3744544	0.12127	0.23602	0.01894	0.0335923	0.00750	2.79177609
0.9969176	0.0784556	2.3966408	0.06420	0.25146	0.01117	0.0417027	0.00983	2.77500359
1.000000	-0.0000037	2.404051	0.00000	0.26551	0.00000	0.05120000	0.01271	2.73346322
0.996917	-0.0784629	2.3966394	-0.07097	0.27793	-0.01508	0.0622339	0.01622	2.66696639
0.9876877	-0.1564385	2.3744517	-0.14823	0.28847	-0.03458	0.0749619	0.02046	2.57553583
0.9723689	-0.2334495	2.3376245	-0.23125	0.29690	-0.05896	0.0895491	0.02556	2.45941717
0.9510552	-0.3090212	2.2863851	-0.31942	0.30302	-0.08868	0.1061683	0.03162	2.31908978
0.9238778	-0.3826877	2.2210493	-0.41205	0.30662	-0.12412	0.12500000	0.03877	2.15527664
0.8910044	-0.4539948	2.1420199	-0.50838	0.30754	-0.16564	0.1462323	0.04717	1.96895301
0.8526376	-0.5225028	2.0497842	-0.60760	0.30562	-0.21349	0.1700611	0.05697	1.76135390
0.809014	-0.5877894	1.9449109	-0.70883	0.30072	-0.26784	0.1966899	0.06833	1.53397994
0.7604025	-0.6494521	1.8280464	-0.81116	0.29275	-0.32880	0.2263299	0.08144	1.28860167
0.7071029	-0.7071107	1.6999114	-0.91363	0.28161	-0.39631	0.2592000	0.09648	1.02726201
0.6494437	-0.7604097	1.5612958	-1.01525	0.26727	-0.47024	0.2955267	0.11367	0.75227677
0.5877805	-0.8090204	1.4130543	-1.11499	0.24970	-0.55030	0.3355443	0.13323	0.46623304
0.5224934	-0.8526433	1.2561008	-1.21184	0.22890	-0.63606	0.3794947	0.15539	0.17198537
0.4539849	-0.8910094	1.091403	-1.30474	0.20491	-0.72695	0.4276275	0.18040	-0.12735044
0.0000000	1.0000000	0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000000
0.0523361	0.9986295	0.1258186	0.02867	0.00021	0.00001	3.2E-07	0.00000	0.15470413
0.1045287	0.9945219	0.2512923	0.05711	0.00082	0.00005	5.12E-06	0.00000	0.30927927
0.1564348	0.9876883	0.3760773	0.08508	0.00185	0.00016	2.592E-05	0.00000	0.46319054
0.2079122	0.9781475	0.4998315	0.11234	0.00327	0.00039	8.192E-05	0.00000	0.61591069

APPENDIX B

0.2588196	0.9659257	0.6222156	0.13867	0.00509	0.00074	0.00020000	0.00000	0.76692115
0.3090177	0.9510563	0.7428943	0.16384	0.00729	0.00126	0.0004147	0.00000	0.91571322
0.3583687	0.9335801	0.8615368	0.18764	0.00987	0.00197	0.0007683	0.00001	1.06178915
0.4067375	0.9135451	0.9778178	0.20984	0.01280	0.00288	0.0013107	0.00002	1.20466335
0.4539915	0.8910060	1.0914187	0.23025	0.01607	0.00400	0.0020995	0.00003	1.34386359
0.5000011	0.8660248	1.2020281	0.24866	0.01967	0.00533	0.00320000	0.00005	1.47893220
0.5446402	0.8386698	1.3093428	0.26488	0.02357	0.00687	0.0046851	0.00008	1.60942732
0.5877864	0.8090161	1.4130686	0.27875	0.02774	0.00860	0.0066355	0.00013	1.73492421
0.6293216	0.7771450	1.5129213	0.29008	0.03218	0.01050	0.0091395	0.00019	1.85501653
0.6691319	0.7431437	1.6086272	0.29873	0.03685	0.01254	0.0122931	0.00028	1.96931768
0.7071081	0.7071055	1.6999239	0.30454	0.04172	0.01468	0.01620000	0.00040	2.07746221
0.7431461	0.6691292	1.7865612	0.30740	0.04677	0.01686	0.0209715	0.00055	2.17910716
0.7771473	0.6293188	1.8683017	0.30718	0.05197	0.01902	0.0267267	0.00074	2.27393357
0.8090183	0.5877835	1.9449213	0.30378	0.05728	0.02108	0.0335923	0.00099	2.36164786
0.8386718	0.5446371	2.0162099	0.29712	0.06268	0.02298	0.0417027	0.00129	2.44198338
0.866027	0.4999979	2.0819722	0.28712	0.06813	0.02460	0.05120000	0.00167	2.51470188
0.8910077	0.4539882	2.142028	0.27374	0.07360	0.02586	0.0622339	0.00214	2.57959508
0.9135466	0.4067342	2.1962125	0.25692	0.07906	0.02664	0.0749619	0.00269	2.63648616
0.9335814	0.3583653	2.2443774	0.23666	0.08446	0.02682	0.0895491	0.00337	2.68523140
0.9510574	0.3090142	2.2863906	0.21294	0.08978	0.02627	0.1061683	0.00416	2.72572166
0.9659266	0.2588161	2.3221369	0.18578	0.09499	0.02487	0.12500000	0.00511	2.75788405
0.9781483	0.2079086	2.3515183	0.15521	0.10004	0.02248	0.1462323	0.00621	2.78168343
0.9876889	0.1564312	2.3744544	0.12127	0.10490	0.01894	0.1700611	0.00750	2.79712400
0.9945223	0.1045251	2.3908823	0.08403	0.10953	0.01411	0.1966899	0.00900	2.80425085
0.9986297	0.0523324	2.4007568	0.04358	0.11391	0.00785	0.2263299	0.01072	2.80315149
1	-0.0000037	2.404051	0.00000	0.11800	0.00000	0.2592000	0.01271	2.79395737
0.9986293	-0.0523397	2.4007559	-0.04659	0.12177	-0.00959	0.2955267	0.01497	2.77684530
0.9945215	-0.1045324	2.3908804	-0.09604	0.12518	-0.02107	0.3355443	0.01754	2.75203890
0.9876877	-0.1564385	2.3744517	-0.14823	0.12821	-0.03458	0.3794947	0.02046	2.71980995
0.9781467	-0.2079158	2.3515147	-0.20297	0.13082	-0.05026	0.4276275	0.02376	2.68047972
0.0000000	1.0000000	0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000000
0.0392599	0.9992290	0.0943828	0.02152	0.00007	0.00000	3.2E-07	0.00000	0.11596870
0.0784593	0.9969173	0.1886201	0.04294	0.00026	0.00002	5.12E-06	0.00000	0.23184239
0.1175377	0.9930684	0.2825666	0.06416	0.00059	0.00007	2.592E-05	0.00000	0.34740267
0.1564348	0.9876883	0.3760773	0.08508	0.00104	0.00016	8.192E-05	0.00000	0.46243891
0.1950908	0.9807852	0.4690082	0.10560	0.00162	0.00032	0.00020000	0.00000	0.57674856
0.2334459	0.9723698	0.5612159	0.12564	0.00232	0.00054	0.0004147	0.00000	0.69013735
0.2714411	0.9624551	0.6525582	0.14508	0.00315	0.00086	0.0007683	0.00000	0.80241960
0.3090177	0.9510563	0.7428943	0.16384	0.00410	0.00126	0.0013107	0.00000	0.91341848
0.3461178	0.9381910	0.8320849	0.18183	0.00517	0.00177	0.0020995	0.00001	1.02296626
0.3826843	0.9238792	0.9199925	0.19895	0.00635	0.00240	0.00320000	0.00001	1.13090459
0.4186607	0.9081428	1.0064816	0.21512	0.00764	0.00314	0.0046851	0.00002	1.23708479
0.4539915	0.8910060	1.0914187	0.23025	0.00904	0.00400	0.0066355	0.00003	1.34136812
0.4886223	0.8724954	1.1746729	0.24425	0.01054	0.00497	0.0091395	0.00005	1.44362606

APPENDIX B

0.5224997	0.8526395	1.2561159	0.25706	0.01214	0.00607	0.0122931	0.00007	1.54374062
0.5555714	0.8314688	1.3356219	0.26858	0.01383	0.00728	0.01620000	0.00009	1.64160460
0.5877864	0.8090161	1.4130686	0.27875	0.01561	0.00860	0.0209715	0.00013	1.73712194
0.6190952	0.7853160	1.4883364	0.28749	0.01746	0.01001	0.0267267	0.00018	1.83020796
0.6494493	0.7604049	1.5613093	0.29475	0.01940	0.01151	0.0335923	0.00023	1.92078974
0.678802	0.7343213	1.6318747	0.30045	0.02140	0.01307	0.0417027	0.00031	2.00880635
0.707108	0.7071055	1.6999239	0.30454	0.02347	0.01468	0.05120000	0.00040	2.09420926
0.7343238	0.6787993	1.7653519	0.30697	0.02559	0.01631	0.0622339	0.00051	2.17696259
0.7604073	0.6494465	1.8280579	0.30768	0.02776	0.01794	0.0749619	0.00064	2.25704348
0.7853182	0.6190923	1.8879451	0.30663	0.02997	0.01955	0.0895491	0.00080	2.33444240
0.8090183	0.5877835	1.9449213	0.30378	0.03222	0.02108	0.1061683	0.00099	2.40916352
0.8314709	0.5555683	1.9988984	0.29910	0.03449	0.02252	0.12500000	0.00121	2.48122501
0.8526414	0.5224965	2.0497935	0.29254	0.03679	0.02383	0.1462323	0.00147	2.55065941
0.8724972	0.4886191	2.0975278	0.28410	0.03909	0.02496	0.1700611	0.00178	2.61751397
0.8910077	0.4539882	2.142028	0.27374	0.04140	0.02586	0.1966899	0.00214	2.68185101
0.9081443	0.4186573	2.1832252	0.26145	0.04370	0.02649	0.2263299	0.00254	2.74374826
0.9238806	0.3826809	2.2210561	0.24722	0.04599	0.02681	0.2592000	0.00302	2.80329924
0.9381923	0.3461144	2.2554622	0.23105	0.04826	0.02675	0.2955267	0.00355	2.86061359
0.9510574	0.3090142	2.2863906	0.21294	0.05050	0.02627	0.3355443	0.00416	2.91581743
0.9624561	0.2714375	2.3137935	0.19289	0.05271	0.02531	0.3794947	0.00486	2.96905376
0.9723706	0.2334423	2.3376287	0.17092	0.05486	0.02381	0.4276275	0.00564	3.02048276
0.0000000	1.0000000	0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000000
0.0314108	0.9995066	0.0755132	0.01722	0.00003	0.00000	3.2E-07	0.00000	0.09276061
0.0627907	0.9980267	0.150952	0.03439	0.00011	0.00001	5.12E-06	0.00000	0.18546147
0.0941085	0.9955619	0.2262417	0.05145	0.00024	0.00004	2.592E-05	0.00000	0.27799646
0.1253335	0.9921147	0.3013082	0.06837	0.00043	0.00008	8.192E-05	0.00000	0.37026721
0.1564348	0.9876883	0.3760773	0.08508	0.00066	0.00016	0.00020000	0.00000	0.46218318
0.1873817	0.9822872	0.4504753	0.10153	0.00096	0.00028	0.0004147	0.00000	0.55366173
0.2181437	0.9759166	0.5244287	0.11769	0.00130	0.00044	0.0007683	0.00000	0.64462825
0.2486905	0.9685830	0.5978645	0.13349	0.00169	0.00066	0.0013107	0.00000	0.73501616
0.2789917	0.9602935	0.6707104	0.14889	0.00213	0.00093	0.0020995	0.00000	0.82476709
0.3090177	0.9510563	0.7428943	0.16384	0.00263	0.00126	0.00320000	0.00000	0.91383091
0.3387387	0.9408805	0.8143451	0.17830	0.00317	0.00166	0.0046851	0.00001	1.00216584
0.3681254	0.9297762	0.8849922	0.19221	0.00375	0.00213	0.0066355	0.00001	1.08973853
0.3971488	0.9177542	0.9547659	0.20554	0.00439	0.00268	0.0091395	0.00002	1.17652415
0.4257802	0.9048266	1.0235974	0.21823	0.00506	0.00330	0.0122931	0.00002	1.26250649
0.4539915	0.8910060	1.0914187	0.23025	0.00579	0.00400	0.01620000	0.00003	1.34767803
0.4817547	0.8763061	1.1581629	0.24155	0.00655	0.00477	0.0209715	0.00004	1.43204007
0.5090425	0.8607414	1.2237641	0.25208	0.00735	0.00562	0.0267267	0.00006	1.51560278
0.5358279	0.8443272	1.2881576	0.26182	0.00819	0.00654	0.0335923	0.00008	1.59838534
0.5620845	0.8270798	1.3512799	0.27072	0.00907	0.00754	0.0417027	0.00010	1.68041598
0.587786	0.8090161	1.4130686	0.27875	0.00999	0.00860	0.05120000	0.00013	1.76173213
0.6129083	0.7901541	1.4734628	0.28586	0.01094	0.00972	0.0622339	0.00017	1.84238051
0.6374252	0.7705122	1.5324028	0.29203	0.01191	0.01090	0.0749619	0.00021	1.92241720
0.6613131	0.7501100	1.5898305	0.29722	0.01292	0.01212	0.0895491	0.00026	2.00190776
0.6845484	0.7289674	1.6456893	0.30140	0.01396	0.01339	0.1061683	0.00032	2.08092733

APPENDIX B

0.7071081	0.7071055	1.6999239	0.30454	0.01502	0.01468	0.12500000	0.00040	2.15956075
0.7289699	0.6845457	1.7524809	0.30662	0.01610	0.01598	0.1462323	0.00048	2.23790263
0.7501124	0.6613104	1.8033084	0.30760	0.01721	0.01729	0.1700611	0.00058	2.31605750
0.7705146	0.6374224	1.8523563	0.30747	0.01833	0.01859	0.1966899	0.00070	2.39413987
0.7901563	0.6129054	1.8995761	0.30621	0.01947	0.01986	0.2263299	0.00083	2.47227437
0.8090183	0.5877835	1.9449213	0.30378	0.02062	0.02108	0.2592000	0.00099	2.55059585
0.8270819	0.5620815	1.988347	0.30018	0.02178	0.02225	0.2955267	0.00116	2.62924949
0.8443292	0.5358248	2.0298105	0.29539	0.02296	0.02333	0.3355443	0.00136	2.70839091
0.8607433	0.5090393	2.0692707	0.28939	0.02413	0.02430	0.3794947	0.00159	2.78818629
0.8763079	0.4817515	2.1066889	0.28218	0.02531	0.02516	0.4276275	0.00185	2.86881246
0.0000000	1.0000000	0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000000
0.026177	0.9996573	0.0629309	0.01435	0.00001	0.00000	3.2E-07	0.00000	0.07729618
0.0523361	0.9986295	0.1258186	0.02867	0.00005	0.00001	5.12E-06	0.00000	0.15455454
0.0784593	0.9969173	0.1886201	0.04294	0.00012	0.00002	2.592E-05	0.00000	0.23171852
0.1045287	0.9945219	0.2512923	0.05711	0.00021	0.00005	8.192E-05	0.00000	0.30873932
0.1305265	0.9914448	0.3137924	0.07117	0.00032	0.00010	0.00020000	0.00000	0.38557586
0.1564348	0.9876883	0.3760773	0.08508	0.00046	0.00016	0.0004147	0.00000	0.46219484
0.1822359	0.9832548	0.4381045	0.09881	0.00063	0.00026	0.0007683	0.00000	0.53857077
0.2079122	0.9781475	0.4998315	0.11234	0.00082	0.00039	0.0013107	0.00000	0.61468602
0.2334459	0.9723698	0.5612159	0.12564	0.00103	0.00054	0.0020995	0.00000	0.69053085
0.2588196	0.9659257	0.6222156	0.13867	0.00127	0.00074	0.00320000	0.00000	0.76610341
0.284016	0.9588195	0.6827889	0.15142	0.00154	0.00098	0.0046851	0.00000	0.84140982
0.3090177	0.9510563	0.7428943	0.16384	0.00182	0.00126	0.0066355	0.00000	0.91646419
0.3338076	0.9426412	0.8024905	0.17593	0.00213	0.00159	0.0091395	0.00001	0.99128863
0.3583687	0.9335801	0.8615368	0.18764	0.00247	0.00197	0.0122931	0.00001	1.06591334
0.3826843	0.9238792	0.9199925	0.19895	0.00282	0.00240	0.01620000	0.00001	1.14037658
0.4067375	0.9135451	0.9778178	0.20984	0.00320	0.00288	0.0209715	0.00002	1.21472476
0.430512	0.9025848	1.0349729	0.22028	0.00360	0.00341	0.0267267	0.00002	1.28901246
0.4539915	0.8910060	1.0914187	0.23025	0.00402	0.00400	0.0335923	0.00003	1.36330244
0.4771598	0.8788166	1.1471165	0.23971	0.00446	0.00463	0.0417027	0.00004	1.43766572
0.500001	0.8660248	1.2020281	0.24866	0.00492	0.00533	0.05120000	0.00005	1.51218158
0.5224997	0.8526395	1.2561159	0.25706	0.00540	0.00607	0.0622339	0.00007	1.58693764
0.5446402	0.8386698	1.3093428	0.26488	0.00589	0.00687	0.0749619	0.00008	1.66202985
0.5664074	0.8241254	1.3616723	0.27212	0.00641	0.00771	0.0895491	0.00011	1.73756256
0.5877864	0.8090161	1.4130686	0.27875	0.00694	0.00860	0.1061683	0.00013	1.81364855
0.6087626	0.7933524	1.4634965	0.28474	0.00748	0.00953	0.12500000	0.00016	1.89040907
0.6293216	0.7771450	1.5129213	0.29008	0.00805	0.01050	0.1462323	0.00019	1.96797389
0.6494493	0.7604049	1.5613093	0.29475	0.00862	0.01151	0.1700611	0.00023	2.04648131
0.6691319	0.7431437	1.6086272	0.29873	0.00921	0.01254	0.1966899	0.00028	2.12607824
0.6883559	0.7253731	1.6548426	0.30200	0.00982	0.01360	0.2263299	0.00034	2.20692021
0.7071081	0.7071055	1.6999239	0.30454	0.01043	0.01468	0.2592000	0.00040	2.28917143
0.7253757	0.6883532	1.7438401	0.30635	0.01106	0.01577	0.2955267	0.00047	2.37300483
0.7431461	0.6691292	1.7865612	0.30740	0.01169	0.01686	0.3355443	0.00055	2.45860208
0.7604073	0.6494465	1.8280579	0.30768	0.01234	0.01794	0.3794947	0.00064	2.54615365
0.7771473	0.6293188	1.8683017	0.30718	0.01299	0.01902	0.4276275	0.00074	2.63585889
0.0000000	1.0000000	0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000000

APPENDIX B

0.0196337	0.9998072	0.0472005	0.01077	0.00000	0.00000	3.2E-07	0.00000	0.05797036
0.0392599	0.9992290	0.0943828	0.02152	0.00002	0.00000	5.12E-06	0.00000	0.11592463
0.0588709	0.9982656	0.1415287	0.03225	0.00004	0.00001	2.592E-05	0.00000	0.17384564
0.0784593	0.9969173	0.1886201	0.04294	0.00007	0.00002	8.192E-05	0.00000	0.23172389
0.0980174	0.9951847	0.2356388	0.05358	0.00010	0.00004	0.00020000	0.00000	0.28955758
0.1175377	0.9930684	0.2825666	0.06416	0.00015	0.00007	0.0004147	0.00000	0.34735261
0.1370127	0.9905693	0.3293854	0.07466	0.00020	0.00011	0.0007683	0.00000	0.40512261
0.1564348	0.9876883	0.3760773	0.08508	0.00026	0.00016	0.0013107	0.00000	0.46288893
0.1757967	0.9844265	0.4226242	0.09540	0.00033	0.00023	0.0020995	0.00000	0.52068063
0.1950908	0.9807852	0.4690082	0.10560	0.00040	0.00032	0.00320000	0.00000	0.57853453
0.2143096	0.9767658	0.5152113	0.11569	0.00049	0.00042	0.0046851	0.00000	0.63649518
0.2334459	0.9723698	0.5612159	0.12564	0.00058	0.00054	0.0066355	0.00000	0.69461490
0.2524922	0.9675989	0.607004	0.13544	0.00068	0.00069	0.0091395	0.00000	0.75295375
0.2714411	0.9624551	0.6525582	0.14508	0.00079	0.00086	0.0122931	0.00000	0.81157959
0.2902853	0.9569401	0.6978608	0.15455	0.00090	0.00105	0.01620000	0.00000	0.87056804
0.3090177	0.9510563	0.7428943	0.16384	0.00103	0.00126	0.0209715	0.00000	0.93000251
0.3276309	0.9448058	0.7876414	0.17294	0.00116	0.00151	0.0267267	0.00001	0.98997419
0.3461178	0.9381910	0.8320849	0.18183	0.00129	0.00177	0.0335923	0.00001	1.05058212
0.3644713	0.9312146	0.8762076	0.19051	0.00144	0.00207	0.0417027	0.00001	1.11193310
0.382684	0.9238792	0.9199925	0.19895	0.00159	0.00240	0.05120000	0.00001	1.17414178
0.4007497	0.9161876	0.9634228	0.20716	0.00175	0.00275	0.0622339	0.00002	1.23733063
0.4186607	0.9081428	1.0064816	0.21512	0.00191	0.00314	0.0749619	0.00002	1.30162996
0.4364102	0.8997478	1.0491524	0.22282	0.00208	0.00355	0.0895491	0.00002	1.36717794
0.4539915	0.8910060	1.0914187	0.23025	0.00226	0.00400	0.1061683	0.00003	1.43412057
0.4713977	0.8819207	1.1332642	0.23740	0.00244	0.00447	0.12500000	0.00004	1.50261173
0.4886223	0.8724954	1.1746729	0.24425	0.00264	0.00497	0.1462323	0.00005	1.57281317
0.5056584	0.8627338	1.2156287	0.25081	0.00283	0.00551	0.1700611	0.00006	1.64489453
0.5224997	0.8526395	1.2561159	0.25706	0.00303	0.00607	0.1966899	0.00007	1.71903332
0.5391394	0.8422165	1.2961187	0.26298	0.00324	0.00666	0.2263299	0.00008	1.79541496
0.5555714	0.8314688	1.3356219	0.26858	0.00346	0.00728	0.2592000	0.00009	1.87423280
0.5717891	0.8204006	1.3746102	0.27384	0.00368	0.00793	0.2955267	0.00011	1.95568808
0.5877864	0.8090161	1.4130686	0.27875	0.00390	0.00860	0.3355443	0.00013	2.03998998
0.6035572	0.7973197	1.4509822	0.28330	0.00413	0.00929	0.3794947	0.00015	2.12735561
0.6190952	0.7853160	1.4883364	0.28749	0.00437	0.01001	0.4276275	0.00018	2.21801003

APPENDIX C

```
#define TRAINING_FILE "C:\\training.dat"

#define WEIGHTS_FILE "C:\\weights.dat"
#define OUTPUT_FILE "C:\\output.dat"
#define TEST_FILE   "C:\\test.dat"
#include "conti.cpp"

 void main()
 {
         float error_tolerance=0.1;
         float total_error=0.0;
         float avg_error_per_cycle=0.0;
         float error_last_cycle=0.0;
         float avgerr_per_pattern=0.0;
         float error_last_pattern=0.0;
         float learning_parameter=0.02;
         unsigned temp,startup;
         long int vectors_in_buffer;
   long int max_cycles;
         long int patterns_per_cycle=0;

         long int total_cycles, total_patterns;


         int i;
         network backp;

         FILE * training_file_ptr,*weights_file_ptr,* output_file_ptr;
         FILE * test_file_ptr,* data_file_ptr;

         if ((output_file_ptr=fopen(OUTPUT_FILE,"w"))==NULL)
         {

                 cout<<"problem opening output file \n";
                 exit(1);
         }


   cout<<"     ------------- NEURAL NETWORK PREDICTION---------------------\n";
         cout<<"   ---------Emmanuel Olawale Olaniyi AJIBOLA-----------------\n";
         cout<<"       ----------------------------------------------\n";
         cout<<"         C++ Neural Networks and fuzzy logic \n";
         cout<<"            backpropagation simulator\n";
         cout<<"         -------------------------------------\n";
         cout<<"   Please enter 1 for TRAINING on, or 0 for off:\n\n";
         cin>>temp;
```

```
backp.set_training(temp);
if (backp.get_training_value()==1)
{
        cout<<"     --> Training mode is *ON*,weights will be  saved\n";
        cout<<"     in the file weights.dat at the end of the\n";
        cout<<"     current set of input (training) data\n";
}
else
{
        cout<<"   --->Training mode is *OFF*.weights will be\
 loaded\n";
        cout<<"   from the file weights.dat and the current\n";
        cout<<"     (test) data set will be used. For the test\n";
         cout<<"     data set, the test.data file should contain\n";
         cout<<"       only inputs, and no expected outputs.\n";
                }


if (backp.get_training_value()==1)
{
        //Read in values fore the error tolerance and
        //the learning rate parameter
        cout<<"         Please enter in the error_tolerance\n";
        cout<<"       ---between 0.001 and 100.0,try 0.1 to start\
\n";
        cout<<"\n";
        cout<<"       and the learning _parameter,beta\n";
        cout<<"     -----between 0.01 to 1.0,try 0.5 to start ---\
 \n\n";
        cout<<"         separate entries by a space\n";
        cout<<"     example:0.1 0.5 sets defaults mentioned :\n\n";
        cin>>error_tolerance>> learning_parameter;
        //------------------------------------------
        // open training file for reading
        //------------------------------------------
if((training_file_ptr=fopen(TRAINING_FILE,"r"))==NULL)
 {
        cout<<"     problem opening training file\n";
        exit(1);
 }

 data_file_ptr=training_file_ptr; //training on

  //read in the maximum number of cycles
 cout<<"     Please enter the maximum cycles for the simulation\n";
 cout<<"     A cycle is one pass through the data set.\n";
 cout<<"        Try a value of 10 to start with\n";
 cin>>max_cycles;


}
else
{
        if ((test_file_ptr=fopen(TEST_FILE,"r"))==NULL)
        {
                cout<<"     problem opening test file\n";
                exit(1);
        }

        data_file_ptr=test_file_ptr;

}
        //training and non training mode.

        //initialize counters
        total_cycles=0;
        total_patterns=0;


        //get layer information
        backp.get_layer_info();
        //set up the network
        backp.set_up_network();

        //initialize the weights
        if (backp.get_training_value()==1)
        {
                //randomize weights
                //open weights file for writing

                if((weights_file_ptr=fopen(WEIGHTS_FILE,"w"))
                        ==NULL)
                {
                        cout<<"problem openiing weights file\n";
                        exit(1);
                }

                backp.randomize_weights();
        }
        else{
                if((weights_file_ptr=fopen(WEIGHTS_FILE,"r"))
                        ==NULL)
                {
                        cout<<"problem opening weights file\n";
                        exit(1);
                }
                backp.read_weights(weights_file_ptr);
        }

        //main loop
        startup=1;
        vectors_in_buffer=MAX_VECTORS;//startup condition
        total_error =0;

        while(    ((backp.get_training_value()==1)
                    &&(avgerr_per_pattern
                                   >error_tolerance)
                                        &&(total_cycles<max_cycles)
                                        &&(vectors_in_buffer!=0))
                                        ||((backp.get_training_value()==0)
                                        &&(total_cycles<1))
                                        ||((backp.get_training_value()==1)
                                        &&(startup==1))
                                        )
        {
                startup=0;
                error_last_cycle=0;//reset for each cycle
                patterns_per_cycle=0;
                //process all the vectors in the datafile
                //going through one buffer at a time
                //pattern by pattern

                while((vectors_in_buffer==MAX_VECTORS))
                {
                        vectors_in_buffer= backp.fill_IObuffer(data_file_ptr);//fill
buffer
                        if(vectors_in_buffer<0)
                        {
                                cout<<"error in reading in vectors,aborting\n";
                                cout<<"check that there are no extra   linefeeds\n";
                                cout<<"in your data file,and that the number\n";
                                cout<<"of layers and size of layers match the\n";
                                cout<<"the parameters provided.\n";
                                exit(1);
                        }

                        //process vectors
                        for(i=0;i<vectors_in_buffer;i++)
```

```
                                {
                                        //get next pattern
                                        backp.set_up_pattern(i);
                                        total_patterns++;
                                        patterns_per_cycle++;
                                        //forward propagate
                                        backp.forward_prop();

                                        if (backp.get_training_value()==0)
                                                backp.write_outputs(output_file_ptr);
                                        //back_propagate,if appropriate
                                        if(backp.get_training_value()==1)
                                        {
                                                backp.backward_prop(error_last_pattern);
                                                error_last_cycle+=error_last_pattern *
error_last_pattern;


//if((avgerr_per_pattern>error_tolerance)&&(total_cycles+1 <max_cycles))
                                                backp.update_weights(learning_parameter);
                                               // backp.list_weights();
                                        }
                                }
                                error_last_pattern=0;
                        }


avgerr_per_pattern=((float)sqrt((double)error_last_cycle))/patterns_per_cycle;
                        total_error+=error_last_cycle;
                        total_cycles++;

                        cout<<"\n\n";
                        cout<<total_cycles<<"\t"<<avgerr_per_pattern<<"\n";
                        fseek(data_file_ptr,0L,SEEK_SET);//reset the file pointer to the
beginning of the file
                        vectors_in_buffer=MAX_VECTORS;//reset
                }//end main loop
                cout<<"\n\n\n\n\n\n\n\n\n\n\n";
                cout<<"------------------------------------------------\n";
                cout<<"done:results in file output.dat\n";
                cout<<"     training: last vector only\n";
                cout<<"      not training: full cycle\n\n";
                if (backp.get_training_value()==1)
                {
                        backp.write_weights(weights_file_ptr);
                        backp.write_outputs(output_file_ptr);
```

```
                              avg_error_per_cycle=(float)sqrt((double)total_error)/ total_cycles;
                              error_last_cycle=(float)sqrt((double)error_last_cycle);


                              cout<<"    weights saved in file weights.dat\n";
                              cout<<"\n";
                              cout<<"------>average error per
cycle="<<avg_error_per_cycle<<"<-----\n";
                              cout<< "-->error last cycle="<<error_last_cycle<<"<-----\n";
                              cout<<"-->error last cycle per
pattern="<<avgerr_per_pattern<<"<----\n";
                    }

                    cout<<"-------------------->total cycles="<<total_cycles<<
                              " <-----\n";
                    cout<<"-------------------->total patterns="<< total_patterns<<
                              " <------\n";
                    cout<<"--------------------------------------------\n";
                    //close all files
                    fclose(data_file_ptr);
                    fclose(weights_file_ptr);
                    fclose(output_file_ptr);

}
```

APPENDIX D

```
#include<stdio.h>
#include<iostream.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>


#define MAX_LAYERS 5
#define MAX_VECTORS 100


  class network;

  class layer
  {
  protected:
          int num_inputs;
          int num_outputs;
          float *outputs; // pointer to array of outputs
          float *inputs;//pointer to array of inputs which are outputs of some other layer

          friend  network;
  public:
          virtual void  calc_out()=0;
  };


  class input_layer:public layer
  {
  private:
  public:
          input_layer(int,int);
          ~input_layer();
          virtual void calc_out();
  };

    class middle_layer;
          class output_layer:public layer
          {
            protected:
         float * weights;
                         float * output_errors;// array of errors at output
                         float * back_errors;//array of errors backpropagated
                         float * expected_values; // to inputs
                         friend network;
    public:
        output_layer(int,int);
        ~output_layer();
        virtual void calc_out();
        void calc_error(float &);
        void randomize_weights();
        void update_weights(const float);
        void list_weights();
        void write_weights(int,FILE *);
        void read_weights(int,FILE *);
        void list_errors();
        void list_outputs();
};
class middle_layer:public output_layer
{
private:
public:
    middle_layer(int,int);
    ~middle_layer();
     void calc_error();
};

 class network
    {
    private:
        layer *layer_ptr[MAX_LAYERS];
        int number_of_layers;
        int layer_size[MAX_LAYERS];
        float *buffer;
        fpos_t position;
        unsigned training;

    public:
        network();
        ~network();
        void set_training(const unsigned &);
        unsigned get_training_value();
        void get_layer_info();
        void set_up_network();
        void randomize_weights();
        void update_weights(const float);
        void write_weights(FILE *);
        void read_weights(FILE *);
        void list_weights();
        void write_outputs(FILE *);
        void list_outputs();
```

```
                    void list_errors();
                    void forward_prop();
                    void backward_prop(float &);
                    int fill_IObuffer(FILE *);
                    void set_up_pattern(int);

                };
                    //layer.cpp


                    inline float squash(float input)
        //squashing function
                    {
                            if (input<-50)
                                    return 0.0;
                            else if (input>50)
                                    return 1.0;
                            else return (float)(1/(1+exp(-(double)input)));
                    }
                    inline float randomweight(unsigned init)
                    {
                            int num;
                            if (init==1)
                                    srand((unsigned)time(NULL));
                            num=rand() % 100;

                            return  2*(float(num/100.00))-1;
                    }
        static void force_fpf()
                    {
                            float x, *y;
                    y=&x;
                    x=*y;
                    }


        // input layer
                    input_layer::input_layer(int i,int o)
                    {
                            num_inputs=i;
                            num_outputs=o;
                            outputs= new float[num_outputs];
                            if (outputs==0)
                            {
                                    cout <<"not enough memory\n";
                                    cout<<"choose a smaller architecture\n";
```

```
                        exit(1);
                }

}

input_layer::~input_layer()
{
        delete[num_outputs] outputs;
}

void input_layer::calc_out()
{
        //nothing to do, yet
}

 //--------------------------------------
 //    output layer
 //---------------------------------------

output_layer::output_layer(int i,int o)
{
        num_inputs=i;
        num_outputs=o;
        weights=new float[num_inputs*num_outputs];
        output_errors=new float[num_outputs];
        back_errors=new float[num_inputs];
        outputs=new float[num_outputs];
        expected_values=new float[num_outputs];
        if((weights==0)||(output_errors==0)||(back_errors==0)
                ||(outputs==0)||(expected_values==0))
        {
                cout<<"not enough memory\n";
                cout<<"choose a smaller architecture\n";
                exit(1);
        }

}

output_layer::~output_layer()
{
        //some compilers may require the array
        //size in the delete statement;those
        //conforming to ansi C++ will not
        delete [num_outputs*num_inputs] weights;
        delete [num_outputs] output_errors;
        delete [num_inputs] back_errors;
```

```
                  delete [num_outputs] outputs;
  }

          void output_layer::calc_out()
          {
                  int i,j,k;
                  float accumulator=0.0;

                  for (j=0;j<num_outputs;j++)
                  {

                          for (i=0;i<num_inputs;i++)
                          {
                                  k=i*num_outputs;
          if(weights[k+j]*weights[k+j]>1000000.0)
                                  {
                                          cout<<"weights are blowing up\n";
                                          cout<<"try a smaller learning constant\n";
                                          cout<<"e.g. beta=0.02  aborting...\n";
                                          exit(1);

                                  }
                                  outputs[j]=weights[k+j]*(*(inputs+i));
                                  accumulator+=outputs[j];
                          }
                          //use the sigmoid activation function
                          outputs[j]=squash(accumulator);
                          accumulator=0;
                  }
          }


          void output_layer::calc_error(float & error)
          {
int i,j,k;
           float accumulator=0;
           float total_error=0;
            for (j=0;j<num_outputs;j++)
            {
                    output_errors[j]=expected_values[j]-outputs[j];
                    total_error+=output_errors[j];
            }

            error=total_error;
            for (i=0; i<num_inputs;i++)
            {
```

```
           k=i*num_outputs;
           for(j=0;j<num_outputs;j++)
           {
                   back_errors[i]=weights[k+j]*output_errors[j];
                   accumulator+=back_errors[i];
           }

            back_errors[i]=accumulator;
                 accumulator=0;
                 //now multiply by derivative of
                 //sigmoid squashing function ,which is
                 // just the input*(1-input)
                 back_errors[i]*=(*(inputs+i))*(1-(*(inputs+i)));
  }

}

void output_layer::randomize_weights()
{
        int i,j,k;
        const unsigned first_time=1;
        const unsigned not_first_time=0;
        float discard;
        discard =randomweight(first_time);
        for (i=0;i<num_inputs;i++)
        {
                k=i*num_outputs;
                for(j=0;j<num_outputs;j++)
                weights[k+j]=randomweight(not_first_time);
        }

}

void output_layer::update_weights(const float beta)
{
        int i,j,k;
        //learning law: weight_change =
        //   beta*output_error*input

        for(i=0;i<num_inputs;i++)
        {
                k=i*num_outputs;
                for (j=0;j<num_outputs;j++)
                        weights[k+j]+=beta*output_errors[j]*(*(inputs+i));
        }
}
```

```
        void output_layer::list_weights()
        {
                int i,j,k;
                for (i=0;i<num_inputs;i++)
                {
                        k=i*num_outputs;
                        for(j=0;j<num_outputs;j++)
                                cout<<"weight["<<i<<","<<j<<"]
is:"<<weights[k+j];
                }
        }
        void output_layer::list_errors()
        {
                int i,j;
                for (i=0;i<num_inputs;i++)
                        cout<<"backerror["<<i<<      "] is
:"<<back_errors[i]<<"\n";

                for (j=0;j<num_outputs;j++)
                        cout<<"outputerrors["<<j<<"]
is:"<<output_errors[j]<<"\n";
        }

        void output_layer::write_weights(int layer_no, FILE * weights_file_ptr)
        {
                int i,j,k;
                //assume file is already open and ready for
                //writing
                //prepend the layer _no to all lines of data
                //format;
                //  layer_no weight[0,0] weight[0,1] ....
                // layer_no weight[1,0] weight[1.1].....
                for(i=0;i<num_inputs;i++)
                {
                        fprintf(weights_file_ptr,"%i ",layer_no);
                        k=i*num_outputs;
                        for(j=0;j<num_outputs;j++)
                        {
                                fprintf(weights_file_ptr,"%f",weights[k+j]);
                        }
                        fprintf(weights_file_ptr,"\n");
                }
        }
        void output_layer::read_weights(int layer_no,FILE * weights_file_ptr)
```

```
{
        int i,j,k;
        //assume file is already open and ready for reading
        //look for the prepended layer _no
        while (1)
        {

                 fscanf(weights_file_ptr,"%i",&j);
                if ((j==layer_no)||(feof(weights_file_ptr)))
                        break;
                else
                {
                        while (fgetc(weights_file_ptr) !='\n')
                        {;}// get rest of line
                }
        }
        if(!(feof(weights_file_ptr)))
        {
                //continue getting first line
                i=0;
        for (j=0;j<num_outputs;j++)
                {
fscanf(weights_file_ptr,"%f",&weights[j]);//i*num_outputs=0
        }

        fscanf(weights_file_ptr,"\n");
        //now get the other lines
        for(i=1;i<num_inputs;i++)
        {
                fscanf(weights_file_ptr,"%i",&layer_no);
                k=i*num_outputs;
                for(j=0;j<num_outputs;j++)
                {
                        fscanf(weights_file_ptr,"%f",&weights[k+j]);
                }
        }
        fscanf(weights_file_ptr,"\n");
        }
        else cout<<"end of file reached\n";
}
void output_layer::list_outputs()
{
        int j;
        for(j=0;j<num_outputs;j++)
        {
                cout<<"outputs["<<j  <<"]is: "<<outputs[j]<<"\n";
            }
    }
    //------------------------------------
    //      middle layer
    //------------------------------------
    middle_layer::middle_layer(int i,int o):
    output_layer(i,o)
    {

    }
    middle_layer::~middle_layer()
    {
            delete [num_outputs*num_inputs] weights;
delete [num_outputs] output_errors;
            delete [num_inputs] back_errors;
            delete [num_outputs] outputs;
    }
    void middle_layer::calc_error()
    {
            int i,j,k;
            float accumulator=0;
            for(i=0;i<num_inputs;i++)
            {
                k=i*num_outputs;
                for(j=0;j<num_outputs;j++)
                {
                    back_errors[i]=weights[k+j]*(*(output_errors+j));
                    accumulator+=back_errors[i];
                }
                back_errors[i]=accumulator;
                accumulator=0;
                //multiply by derivative of
                //sigmoid squashing function-input*(1-input)
                back_errors[i]*=(*(inputs+i))*(1-(*(inputs+i)));
            }
    }
    network::network()
    {
            position=0L;
    }
    network::~network()
    {
    int i,j,k;
    i=layer_ptr[0]->num_outputs;//inputs
    j=layer_ptr[number_of_layers-1]->num_outputs;//outputs
```

```
                k=MAX_VECTORS;


                delete [(i+j)*k]buffer;
                }
                void network::set_training(const unsigned & value)
                {
                        training=value;
                }
                unsigned network::get_training_value()
                {
                        return training;
                }

                void network ::get_layer_info()
                {
                        int i;
                        //-------------------------------------------
                        //
                        //  Get layer sizes for the network
                        //
                        //-------------------------------------------

                        cout<<"      Please enter  in the number of layers for your
network.\n";
                                cout<<" you can have a minimum of 3 to a maximum of
5.\n";
                        cout<< " 3 implies i hidden layer;5 implies 3 hidden layers:\n\n";
                                cin>>number_of_layers;
                        cout<<" Enter in the layer sizes separated by spaces.\n";
                        cout<<"For a network with 3 neurons in the input  layer,\n";
                        cout<<"2 neurons in a hidden layer and 4 neurons in the \n";
                        cout<<"output layer ,you would enter :3 2 4.\n";
                                cout<<" You can have up to 3 hidden layers,for five
maximum entries :\n\n";

                                for (i=0;i<number_of_layers;i++)
                                {
                                        cin>>layer_size[i];
                                }


                                //--------------------------------------------
                                // size of layer:
                                //--------------------------------------------
```

```
        }
        void network::set_up_network()
        {
                int i,j,k;

                //-----------------------------------------
                //construct the layers
                //-----------------------------------------

                layer_ptr[0]=new input_layer(0,layer_size[0]);

                for (i=0;i<(number_of_layers-1);i++)
                {
                        layer_ptr[i+1]=          new
middle_layer(layer_size[i],layer_size[i+1]);
                }

                layer_ptr[number_of_layers-1]=new
output_layer(layer_size[number_of_layers-2],layer_size[number_of_layers-1]);
                for (i=0;i<(number_of_layers-1);i++)
                {
                        if(layer_ptr[i]==0)
                        {
                                cout<<"insufficient memory\n";
                                cout<<"use a smaller architecture\n";
                                exit(1);
                        }
                }


                //-----------------------------------------
                // connect the layers
                //-----------------------------------------
                for (i=1;i<number_of_layers;i++)
                        layer_ptr[i]->inputs= layer_ptr[i-1]->outputs;
                //for backpropagation ,set the output_errors to next layer
                // back_errors for all layers except the output
                // layer and input layer

                for (i=1;i<number_of_layers-1;i++)
                ((output_layer *)layer_ptr[i])->output_errors=((output_layer
*)layer_ptr[i+1])->back_errors;

                //define the iobuffer that caches data from the datafile
                i=layer_ptr[0]->num_outputs;//inputs
                j=layer_ptr[number_of_layers-1]->num_outputs;//outputs
```

```
                        k=MAX_VECTORS;
                        buffer=new float[(i+j)*k];
                        if (buffer==0)
                                cout<<"insufficient memory for buffer\n";
                }

                void network::randomize_weights()
                {
                        int i;
                        for (i=1;i<number_of_layers;i++)
                                ((output_layer *)layer_ptr[i])->randomize_weights();
                }

                void network::update_weights(const float beta)
                {
                        int i;
                        for(i=1;i<number_of_layers;i++)
                                ((output_layer *)layer_ptr[i])->update_weights(beta);
                }

                void network::write_weights(FILE * weights_file_ptr)
                {
                        int i;
                        for (i=1;i<number_of_layers;i++)
                                ((output_layer *)layer_ptr[i])-
>write_weights(i,weights_file_ptr);
                }

                void network::read_weights(FILE * weights_file_ptr)
                {
                        int i;
                        for (i=1;i<number_of_layers;i++)
                                ((output_layer *)layer_ptr[i])-
>read_weights(i,weights_file_ptr);
                }

                void network::list_weights()
                {
                        int i;
                        for(i=1;i<number_of_layers;i++)
                        {
                                cout<<"layer number :"<<i<<"\n";
        ((output_layer *)layer_ptr[i]) ->list_weights();
                        }
                }
                void network::list_outputs()
                {
                        int i;
                        for(i=1;i<number_of_layers;i++)
                        {
                                cout<<"layer number:"<<i<<"\n";
                                ((output_layer *)layer_ptr[i])->list_outputs();
                        }
                }
                void network::write_outputs(FILE *outfile)
                {
                        int i,ins,outs;
                        ins=layer_ptr[0]->num_outputs;
                        outs=layer_ptr[number_of_layers-1]->num_outputs;
                        float temp;
                //      fprintf(outfile,"for input vector:\n");
                        //for (i=0;i<ins;i++)
                //      {
                        //      temp=layer_ptr[0]->outputs[i];

        //              fprintf(outfile,"%f  ",temp);
                //      }

                        //fprintf(outfile,"\noutput vector is :\n");
                        for (i=0;i<outs;i++)
                        {
                                temp=layer_ptr[number_of_layers-1]-> outputs[i]    ;
                                fprintf(outfile,"%f  ",temp);

                        }

                        if (training==1)
                        {
                                fprintf(outfile,"expected output vector is:\n");
                                for (i=0;i<outs;i++)
                                {
                                        temp=((output_layer
*)(layer_ptr[number_of_layers-1]))->expected_values[i];
                                        fprintf(outfile,"%f   ",temp);
                                }
                        }

                fprintf(outfile,"\n");
                }

                void network::list_errors()
                {
```

```
        int i;
        for(i=1;i<number_of_layers;i++)
        {
                cout<<"layer number :"<<i<< "\n";
                ((output_layer *)layer_ptr[i])
                        ->list_errors();
        }
}


int network::fill_IObuffer(FILE * inputfile)
{
        int i,k,count,veclength;
        int ins,outs;
        ins=layer_ptr[0]->num_outputs;
        outs=layer_ptr[number_of_layers-1]->num_outputs;
        if(training==1)
                veclength=ins+outs;
        else
                veclength=ins;
        count=0;
        while ((count<MAX_VECTORS)&&(!feof(inputfile)))
        {
                k=count*(veclength);
                for (i=0;i<veclength;i++)
                {
                        fscanf(inputfile,"%f",&buffer[k+i]);
                }
                fscanf(inputfile,"\n");
                count++;
        }


        if (!(ferror(inputfile)))
                return count;
        else return -1;
}


void network::set_up_pattern(int buffer_index)
{
        int i,k;
        int ins,outs;
        ins=layer_ptr[0]->num_outputs;
        outs=layer_ptr[number_of_layers-1]->num_outputs;
        if (training==1)
```

```
                k=buffer_index*(ins+outs);
        else
                k=buffer_index*ins;
        for(i=0;i<ins;i++)
                layer_ptr[0]->outputs[i]=buffer[k+i];
        if(training==1)
        {
                for(i=0;i<outs;i++)
                        ((output_layer *)layer_ptr[number_of_layers-1])->
                        expected_values[i]=buffer[k+i+ins];
        }
}
void network::forward_prop()
{
        int i;
        for(i=0;i<number_of_layers;i++)
        {
                layer_ptr[i]->calc_out();
        }
}

void network::backward_prop(float & toterror)
{
        int i;
        ((output_layer*)layer_ptr[number_of_layers-1])->
                calc_error(toterror);
        for(i=number_of_layers-2;i>0;i--)
        {
                ((middle_layer*)layer_ptr[i])->
                        calc_error();
        }
}
```

APPENDIX E

```
#include<stdio.h>
#include<iostream.h>
#include<stdlib.h>
#include<math.h>


    void main ()
    {
        FILE * outfile,*infile;
  float div[245][9];
        float a[245][245];
        float p[245][9];
        int num_fields=12;
        int i;
        int j;
  float mean[9];
   float sdev[9];
   float d[9];
        int k;
     float n;
        n=0;
        float c[9];



        infile=fopen("C:\\day.dat","r");
        outfile=fopen("C:\\arrayy.dat", "w");
        if ((infile==NULL) || (outfile ==NULL))
        {
            cout<< " cant open a file\n";
            exit(1);
        }

        while(!(feof(infile)))
        {
            for (i=0;i<245;i++)
                for(j=0;j<9;j++)
            {
                fscanf(infile,"%f ",&p[i][j]);

            }


        }
```

```
                //cout << p[1][4]-p[0][4];



  float diff[245][9];

        for(i=0;i<243;i++)
                for(j=0;j<9;j++)
                {
        diff[i][j]=(p[i+1][j]-p[i][j]);
                }


                //finding the mean

        for(j=0;j<9;j++)
                {
                for(i=0;i<243;i++)
                {


                 n+=diff[i][j];

                        }
        c[j]=n;


        n=0;

        }


                //computes the mean
 for( j=0;j<9;j++)
 {
        mean[j]=c[j]/243;

 }



    float sd=0;
//computes the standard deviation
for(j=0;j<9;j++)
{
       for(i=0;i<243;i++)
```

```
         {
                //diff[i][j]=(p[i+1][j]-p[i][j]);
         sd+=pow((diff[i][j]-mean[j]),2);

         }

         d[j]=sd;

         sd=0;

         }


         for(j=0;j<9;j++)
         {

                 sdev[j]=sqrt(d[j]/243);
         //      cout<<sdev[j]<<"  ";

         }

  //cout<<"  "<<sdev[4];
 cout<<"  "<<mean[8]<<"  "<<sdev[8];

   //normalising the range
     float t[245][9];
                 for(i=0;i<243;i++)
                 {
                         for(j=0;j<9;j++)
                         {
                                 //if(sdev[j]==0)
                                 //{sdev[j]=0.1;}

                         t[i][j]=(diff[i][j]-mean[j])/sdev[j];

                 }

                 }

        //squashing funnction
 float h[245][9];

 for(  i=0;i<243;i++)
         { for(j=0;j<9 ;j++)
         {
         if(t[i][j]< -20000)
```

```
                        {


                                t[i][j]=0;
                                h[i][j]=t[i][j];}

                       // cout<<h[i][j]<<"  ";}
                        else if(t[i][j]>20000)
                        {
                                t[i][j]=1.0;
                                h[i][j]=t[i][j];}

                       // cout<<h[i][j]<<"  ";}
                        else  {

                                h[i][j]= (float)(1/(1+exp(-t[i][j])));}


                }
        }


//      cout<<"     accenting the change"<<"\n\n";

 //accenting the ch8ange(image processing technique)
 float s[245][9];
 for(i=0;i<243;i++)
 {
        for(j=0;j<9;j++)
        {

                s[i][j]=(p[i+1][j]-p[i][j])/(p[i+1][j]+p[i][j]);
                //cout<<s[i][j]<<"  ";
        }
        // cout<<"\n\n";
 }


 //cout<<" Time shifting and final data set"<<"\n\n";
for(i=0;i<243;i++)
        {
        for(j=0;j<18;j++)
        {
                if(j>8)
                {
                        if(j==9)
                        { h[i][j-9]=h[i+1][j-9];
```

```
                                fprintf(outfile,"%f\t",h[i][j-9]);}
                        else{
                                fprintf(outfile,"%f\t", h[i][j-9]) ;}
                }

  else
                {fprintf(outfile,"%f\t",s[i][j]);}
        }
        fprintf(outfile,"\n");
 }




}
```

APPENDIX F

```
#include<stdio.h>
#include<iostream.h>
#include<stdlib.h>
#include<math.h>


        void main ()
        {
   FILE * outfile,*infile;
    float p[34];
         float d,e ,f[34];
         int j;
         float sdev=3.75946;
         float mean=0.000707759;
         f[0]=0;


                infile=fopen("C:\\afile.dat","r");
                outfile=fopen("C:\\bfile.dat", "w");
                if ((infile==NULL) || (outfile ==NULL))
                {
                        cout<< " cant open a file\n";
                        exit(1);
                }


    while(!(feof(infile)))
                {

                                for(j=0;j<34;j++)
                        {
                                fscanf(infile,"%f ",&p[j]);
         fprintf(infile,"\n");

                        }


               }

        for (j=1;j<34;j++)
        {
                d=-(log((1/p[j])-1));
                e=((d*sdev)+mean);
                f[j]=e+f[j-1];
        }
```

```
  for (j=0;j<34;j++)
          {
   fprintf(outfile,"%f\t",f[j]);

          fprintf(outfile,"\n");
          }
         }
```