

To the Graduate Council:

I am submitting herewith a thesis written by Garth Forde entitled "Application of Artificial Neural Networks in the Prediction of Maximum Absorbed Dose during a Major Solar Particle Event." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Nuclear Engineering.

Lawrence Townsend
Lawrence Townsend, Major Professor

We have read this thesis

and recommend its acceptance

Lawrence Z Miller
Wesley Atkins

Accepted for the Council:

Lawrence Mink
Associate Vice Chancellor and
Dean of The Graduate School

**APPLICATION OF ARTIFICIAL NEURAL NETWORKS IN THE
PREDICTION OF MAXIMUM ABSORBED DOSE DURING A
MAJOR SOLAR PARTICLE EVENT**

A Thesis

Presented for the

Master of Science Degree

The University of Tennessee, Knoxville

Garth Mark Forde

August 1997

Acknowledgments

There are many people to whom I am grateful for making my time at the University of Tennessee so interesting and rewarding. I have benefited greatly from knowing the faculty and graduate students in the Department of Nuclear Engineering over the last two years. I am especially grateful to my Thesis Committee, Dr. Lawrence Townsend, Dr. Wesley Hines, and Dr. Laurence Miller for their guidance, support, critique, and encouragement. My special thanks also to Dr. Robert Uhrig who gave his financial support for this work.

There are a number of other people whose assistance should also be recognized. I would like to thank Nigel Gomez, James Kuropatwinski, Rebecca Russel, and Donna Russel for their help in my personal and academic affairs, as well as their friendship.

Permission to use information from the following publishers was granted to this author: Kluger Academic Publishers, John Wiley and Sons Limited, and Prentice Hall.

Finally, the greatest debt is owed to my family. Without their support and encouragement I would never have striven to achieve.

Abstract

A major source of concern for planners involved in preparing for future human exploration of Earth's moon and Mars is the occurrence of random, large Solar Particle Events (SPEs) consisting of large numbers of energetic protons and heavy ions emitted by the Sun. During the events radiation exposures of astronauts performing extravehicular activities or extraterrestrial surface operations can reach levels which are mission-threatening over time periods ranging from hours to days. The total dose received by astronauts during one of these events can be estimated from the measured intensity and energy dependence of the particle fluxes. Typical profiles of dose versus time, calculated for past large SPEs display a Weibull-type mathematical form. Artificial Neural Networks are ideal approximators for this type of nonlinear function. In this work an innovative Sliding Time-Delay Neural Network (STDNN) is developed and applied to the problem of predicting maximum doses received by astronauts from large SPEs as they are occurring. The STDNN is trained and tested using astronaut organ doses calculated from actual data collected by satellites during previous large SPEs. Results of this testing indicate that the STDNN is able to predict total doses to within 5% of the organ dose calculations for SPEs which are not part of the training sets. This is the first time that Artificial Neural Networks have been used successfully to predict doses from these events.

Table of Contents

Chapter	Page
1. Introduction.....	1
1.1 Background.....	1
1.2 Statement of the Problem.....	2
2. The Space Environment.....	4
2.1 Introduction.....	4
2.2 Van Allen Belts.....	4
2.2.1 Inner and Outer Radiation Belts.....	4
2.2.2 The radiation sources of the Inner and Outer Radiation Belts.....	5
2.3 Galactic Cosmic Rays.....	6
2.4 Solar Particle Events.....	7
2.4.1 The Sun.....	7
2.4.2 Solar Flares.....	11
2.4.3 Coronal Mass Ejections.....	13
3. Dose Calculations.....	15
3.1 Introduction.....	15
3.2 Methods of Transport.....	15
3.3 Dose and Dose Equivalent Calculations.....	17
3.4 Dose Analysis for SPEs.....	20

4.	Artificial Neural Networks.....	21
4.1	Introduction.....	21
4.2	The Biological Model.....	21
4.3	The Basic Neuron Model.....	24
4.4	Network Architecture.....	28
4.4.1	Feedforward Multilayer Perceptron.....	28
4.4.2	Time-Delay Neural Networks.....	30
4.5	Training the Artificial Neural Network.....	31
4.5.1	Method of Steepest Descent.....	33
4.5.2	Gauss-Newton Method.....	34
4.5.3	Levenberg-Marquardt Method.....	36
4.5.4	Important Training Issues.....	37
5.	Methodology.....	41
5.1	Objective.....	41
5.2	The Data Set.....	41
5.3	Generating the Training and Test Sets.....	44
5.4	Training the Network.....	49
5.5	Testing the Network.....	50
6.	Results.....	52
6.1	Training and Testing the Network.....	52
6.2	Discussion.....	61

7.	Conclusions and Recommendations for Further Study.....	62
7.1	Conclusion.....	62
7.2	Recommendations for Further Study.....	63
	References.....	65
	Appendices.....	69
	Appendix 1.....	70
	Appendix 2.....	71
	Appendix 3.....	72
	Appendix 4.....	75
	Appendix 5.....	76
	Vita.....	79

List of Figures

Figure	Page
2.4.1 A simplified cutaway of the solar interior [8].....	8
2.4.2 An H α picture of the chromosphere [8].....	9
2.4.3 A photograph of a solar eclipse [8].....	10
2.4.4 The flaring loop model [9].....	12
2.4.5 An idealized solar-flare radiative energy intensity-time curve [9].....	13
3.3.1 Quality factor Vs Linear Energy Transfer [15].....	19
3.4.1 Six large solar flare integral fluences from GOES-7 data [7].....	20
4.2.1 (a):A schematic view of the biological neuron [16].....	22
(b):A simplified sketch of synapses [16].....	22
4.3.1 A schematic diagram of the basic neuron model [16].....	25
4.3.2 A plot of the symmetric sigmoid function $y_j = \tanh(u_j) = \frac{(1 - \exp(-2u_j))}{(1 + \exp(-2u_j))}$	26
4.3.3 A plot of the nonsymmetric sigmoid function $y_j = \frac{1}{(1 + \exp(-u_j))}$	27
4.4.1 A schematic diagram of a three layer MLP [17].....	29
4.4.2 Time Delay Neural Network.....	30
4.4.3 Sliding Time Delay Neural Network.....	31
4.5.1 The Steepest Descent Method with Line Search.....	34
4.5.2 Properly fitted data.....	38

4.5.3	Overfitted data.....	39
4.5.4	A plot of an error surface with both local and global minima.....	40
5.2.1	Weibull Fit of 9/89 SPE dose [26].....	42
5.2.2	A Weibull curve generated by the MATLAB script in Appendix 1.....	44
5.3.1	A Weibull curve with a sample along the interval T	45
5.3.2	The input set generated by the script from Appendix 2. Let dose at time t_0 be an arbitrary point on the Weibull curve, this corresponds to one nodal input. The other four input nodes are generated by obtaining dose at times $t_0 - \tau$, $t_0 - 2\tau$, $t_0 - 3\tau$, and $t_0 - 4\tau$	46
5.3.3	The output set generated by the script in Appendix 2. This set contains the D_∞ for a specific SPE.....	47
5.3.4	The concatenated input set.....	48
5.3.5	The concatenated output set.....	48
5.5.1	A flow chart of the Methodology process.....	51
6.1.1	The training input set.....	53
6.1.2	The training output set.....	54
6.1.3	Testing the training set.....	55
6.1.4	Test One.....	56
6.1.5	Test Two.....	57
6.1.6	Test Three.....	58
6.1.7	Test Four.....	59
6.1.8	Test Five.....	60

List of Figures

Figure	Page
2.4.1 A simplified cutaway of the solar interior [8].....	8
2.4.2 An H α picture of the chromosphere [8].....	9
2.4.3 A photograph of a solar eclipse [8].....	10
2.4.4 The flaring loop model [9].....	12
2.4.5 An idealized solar-flare radiative energy intensity-time curve [9].....	13
3.3.1 Quality factor Vs Linear Energy Transfer [15].....	19
3.4.1 Six large solar flare integral fluences from GOES-7 data [7].....	20
4.2.1 (a):A schematic view of the biological neuron [16].....	22
(b):A simplified sketch of synapses [16].....	22
4.3.1 A schematic diagram of the basic neuron model [16].....	25
4.3.2 A plot of the symmetric sigmoid function $y_j = \tanh(u_j) = \frac{(1 - \exp(-2u_j))}{(1 + \exp(-2u_j))}$	26
4.3.3 A plot of the nonsymmetric sigmoid function $y_j = \frac{1}{(1 + \exp(-u_j))}$	27
4.4.1 A schematic diagram of a three layer MLP [17].....	29
4.4.2 Time Delay Neural Network.....	30
4.4.3 Sliding Time Delay Neural Network.....	31
4.5.1 The Steepest Descent Method with Line Search.....	34
4.5.2 Properly fitted data.....	38

1. Introduction

1.1 Background

The Cold War was directly responsible for many advances in technology. The Space Race between the Union of Soviet Socialist Republics (USSR) and the United States of America (USA) was one of the outcomes. Early space missions by both powers involved simple satellites. These satellites would be launched into low Earth orbit where they would gather data and return it to Earth either by radio transmissions or by dropping the recorded information via parachute. These missions gradually progressed to the launching of small mammals into low Earth orbit. These missions provided planners with data vital to the design of technologies that would allow for humans to travel safely into low orbit. Both the USSR and the USA planned and executed successful manned missions to low Earth orbit. However, the USA went a step further. Between the years 1969 and 1972 they were able to safely send men to the Moon and back. However, even before the Cold War ended, manned space missions were once again relegated to the relative safety of the Earth's magnetosphere.

Manned space missions beyond the Earth's magnetosphere, e.g. trips to the Moon or to Mars, are an ambitious enterprise. The deep space environment is hazardous. The energetic radiation in this environment comes from the Sun and other stars. The Earth's magnetosphere protects spacecraft in low Earth orbit from much of this radiation. However, a spacecraft on a manned mission to Mars, once it leaves the Earth's magnetosphere, will encounter radiation from Galactic Cosmic Radiation (GCR), and

from energetic Solar Particle Events (SPEs). This project will ignore the radiation exposures that an interplanetary astronaut will receive from GCR and will focus instead on the doses from large SPE due to the fact that these doses can be very large and can vary significantly. Acute exposure to astronauts from one SPE during a manned mission could be mission threatening.

1.2 Statement of the Problem

The radiation doses associated with an SPE are not easily predicted. Dose versus time profiles of these events have only been calculated after such an event has occurred. A method of predicting the maximum dose to astronauts during an SPE will be useful during the planning and execution of a deep space manned mission e.g. an astronaut working on the Moon's surface may be able to decide whether to seek shelter or to continue working based on the prediction of the SPE maximum dose. The aim of this project is to determine the feasibility of using an artificial neural network (ANN) to predict the maximum dose to interplanetary astronauts when a major SPE occurs.

Artificial neural networks have many applications in modern day life. ANNs are used at nuclear power facilities for control and fault analysis. They are applied to solving the problems associated with complex control and signal processing in avionics, robotics, and fault diagnosis. ANNs are also widely used in pattern recognition. These patterns include images and sound. The ANN's pattern recognition ability is of key importance in this work.

The use of ANNs in SPE dose prediction is not a new application. Aso *et al* 1994 used the ANNs for short term prediction of solar flares [1]. The algorithm used sunspot characteristics in an if-then format as the basis for prediction. The results of this work indicated that a 99% prediction was possible provided the ANN used sunspot characteristics that were known. Work by Jakimiec *et al* 1994 incorporated an algorithm that used the fluxes of the high energy solar particles for prediction [2]. These works used parameters that were dependent on other highly variable parameters that had large uncertainties associated with them. This project uses a new yet very simple and novel approach in the application of ANNs for SPE prediction. The dose-time profile of an SPE can be modeled by a Weibull function fit [3]. Utilizing the ANNs for their ability to operate as function approximators, an ANN is trained and tested with SPE dose data that is in Weibull function form. Hence, for the first time, ANNs are used to predict accrued doses to interplanetary crews from solar particle events.

2. The Space Environment

2.1 Introduction

For a manned space mission to the Moon or to Mars, spacecraft designers must consider crew safety, and spacecraft shielding from radiation in the space environment. The radiations in the space environment fall into three categories: the Van Allen Belts, Galactic Cosmic Rays, and Solar Particle Events.

2.2 Van Allen Belts

In 1958 James Van Allen and his associates discovered that Earth had its own radiation belts. From the data it was observed that particle intensities could be organized by the Earth's magnetic field [4]. This was interpreted to mean that the particles were charged and being controlled by the Earth's geomagnetic field.

2.2.1 Inner and Outer Radiation Belts

Detailed information about the Van Allen radiation belts over an extended region of space was first provided by the Explorer IV spacecraft. From these data it was shown that there are two zones, one located between the ionosphere and about $2R_E$ (where R_E is the radius of the Earth) and the other located between $3R_E$ and $4R_E$. This double peak distribution led to the idea of the Earth having two radiation zones or belts. However,

measurements of lower-energy particles (i.e. > 40 keV) showed that the radiation was continuous from ionospheric heights to the magnetopause.

2.2.2 The Radiation Sources of the Inner and Outer Radiation Belts

When cosmic rays penetrate the atmosphere they interact with the atmospheric components and produce secondary particles. These secondary particles include neutrons. These neutrons decay into protons, electrons, and anti-neutrinos. These daughter particles are the primary sources of the energetic particle components of the inner radiation belt.

The outer radiation belt extends outward on the sunward side to about $10R_E$. It is bounded by a particle trapping boundary that is in the same region as the magnetopause position. On the hemisphere not facing the sun the boundary is $8-9R_E$ [4]. The particles in the outer radiation belt have energies that range from a few keV to hundreds of keV. Solar wind and the ionosphere contribute to this population of particles. The ionosphere provides the lower energy components. The higher energy components are provided by accelerating solar wind particles. Another source of high energy particles are the remnants of plasma sheet particles that were injected into the outer radiation belt during auroral disturbances.

2.3 Galactic Cosmic Rays

The true origins of galactic cosmic rays are unknown [5,6]. There is speculation that the particles are the remnants of supernova. Other ideas suggest that the particles are just leftover gases from developing solar systems and galaxies. For a mission to the Moon or to Mars one becomes concerned more with the types of particles, their path lengths, and their energies. The universal hypothesis indicates that there should be about 10^6 eV m^{-3} of cosmic ray energy everywhere in the universe [5]. The particles involved include protons, heavier ions, electrons, and γ -rays. The γ -rays are a result of inverse Compton scattering of starlight and the bremsstrahlung of high energy electrons make up the lower energy spectrum $E < 100 \text{ MeV}$. The γ -rays that are a result of the decay of neutral pions (themselves the result of collisions between protons, nuclear components of high energy particles, and interstellar gases) make up the higher energy spectrum $E > 300 \text{ MeV}$. Research done by Wilson *et al* [7] indicates that absorbed dose (where dose will be discussed in Chapter 3) from GCR particles does not pose an acute health hazard for humans. However the biological concern is that GCRs present a chronic radiation exposure to interplanetary crews, thus increasing the statistical likelihood of crew members developing cancers. In the case of the high-energy, heavy-ion component, there may be unique biological effects not observed with conventional x-rays or γ -rays.

2.4 Solar Particle Events

2.4.1 The Sun

The Sun is a sphere of plasma with a radius of 700,000 km. Its core is a fusion reactor operating at a temperature of 1.5×10^7 K and a particle density of 10^{32} m^{-3} [8]. The Sun's atmosphere consists of three regions: the photosphere, the chromosphere, and the corona. Figure 2.4.1 [8] shows a simplified cutaway of the solar interior. Most of the observable light comes from the photosphere, which is a thin surface layer only 500 km in depth. This region exhibits global oscillations with amplitudes of a few cm. Similar to seismology on Earth, these oscillations in the photosphere are used to probe and deduce the internal structure of the Sun. In the chromosphere and the corona the density falls from 10^{23} to 10^{14} however, the temperature behaves differently. Moving outward the temperature increases from 6000 K to 10,000 K. The temperature then makes a significant increase to 2×10^6 K through a narrow transition layer. In the photosphere the magnetic field strength varies between one and a few thousand gauss. In the chromosphere and the corona the magnetic field strength varies between one and a few hundred gauss(G) { $1 \text{ G} = 10^{-4} \text{ Tesla (T)}$ }. A turbulent convective zone can be found in the top 200,000 km of the Sun's interior. The turbulence is on discrete scales. At the surface it shows up as granulation, mesogranulation, and supergranulation and the giant cells are on scales of 1000 km, 10,000 km, 30,000 km, and 300,000 km respectively. For plasmas the particle collision mean-free path in the photosphere, chromosphere, and corona, are 10^{-7} m , 10^{-1} m , and $3 \times 10^5 \text{ m}$ respectively. Beyond a few solar radii the plasma becomes collisionless.

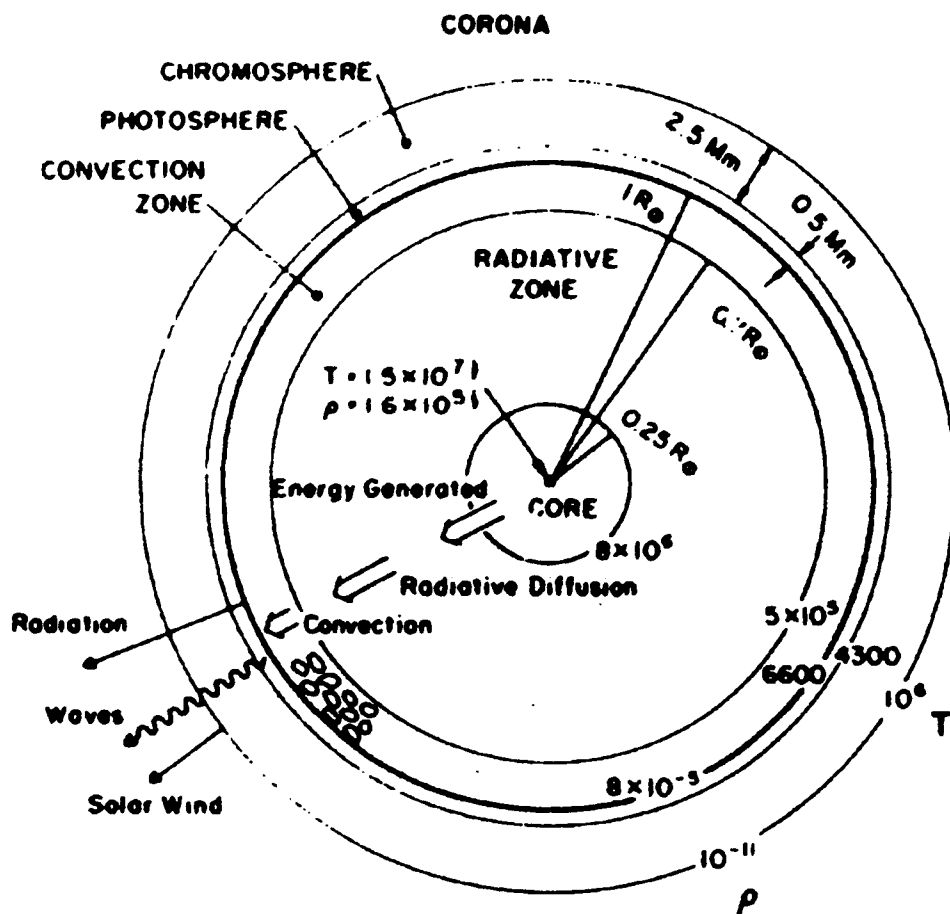


Figure 2.4.1 A simplified cutaway of the solar interior [8]

Reprinted with kind permission from Kluwer Academic Publishers

When magnetic flux tubes break through the Sun's surface in one location and then reenter in another location the results are sunspots. These can be seen in a picture of the photosphere. Sunspots are usually found in pairs. Figure 2.4.2 [8] is an $H\alpha$ photograph of the chromosphere. It shows that around a sunspot region there is a bright active region where a solar flare can occasionally take place. In the photograph, structures called filaments are visible. There are two types of filaments: quiescent filaments are long and thick and can be found far from active regions, and plage filaments are much smaller and can usually be found inside active regions.

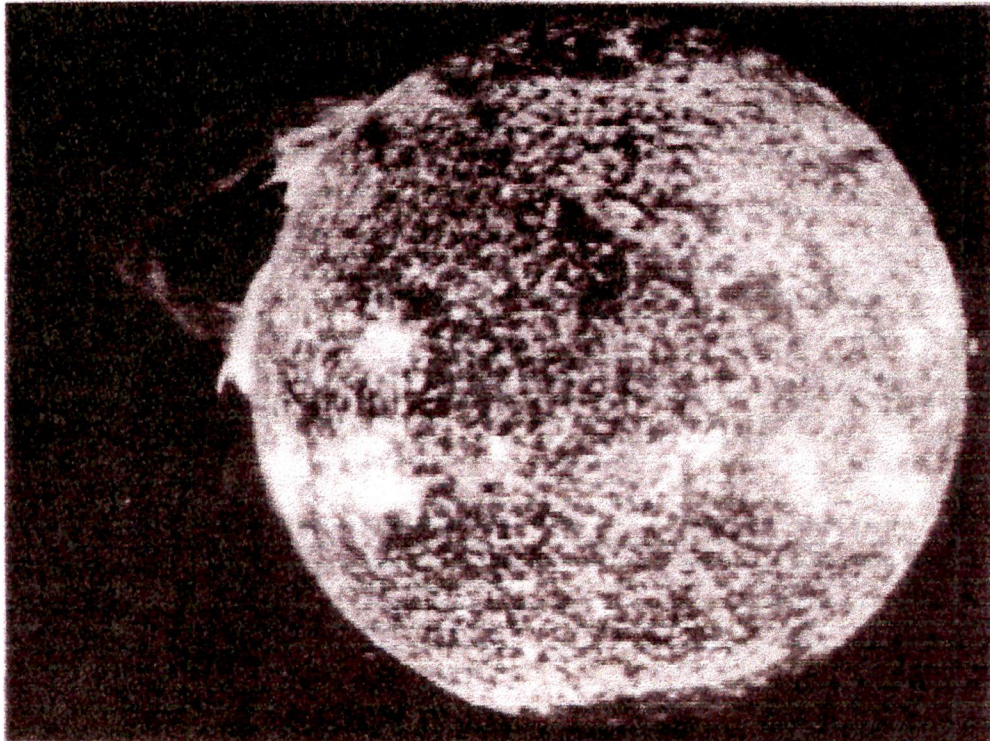


Figure 2.4.2 An $H\alpha$ picture of the chromosphere [8]

Reprinted with kind permission from Kluwer Academic Publishers

Figure 2.4.3 [8] is a photograph of a solar eclipse. The corona is revealed with the magnetic field being outlined by the plasma structures. There are some regions, called closed regions, where the magnetic field is strong enough to contain the plasma, there are also other regions called open magnetic regions where plasma streams outward from the Sun as solar wind.

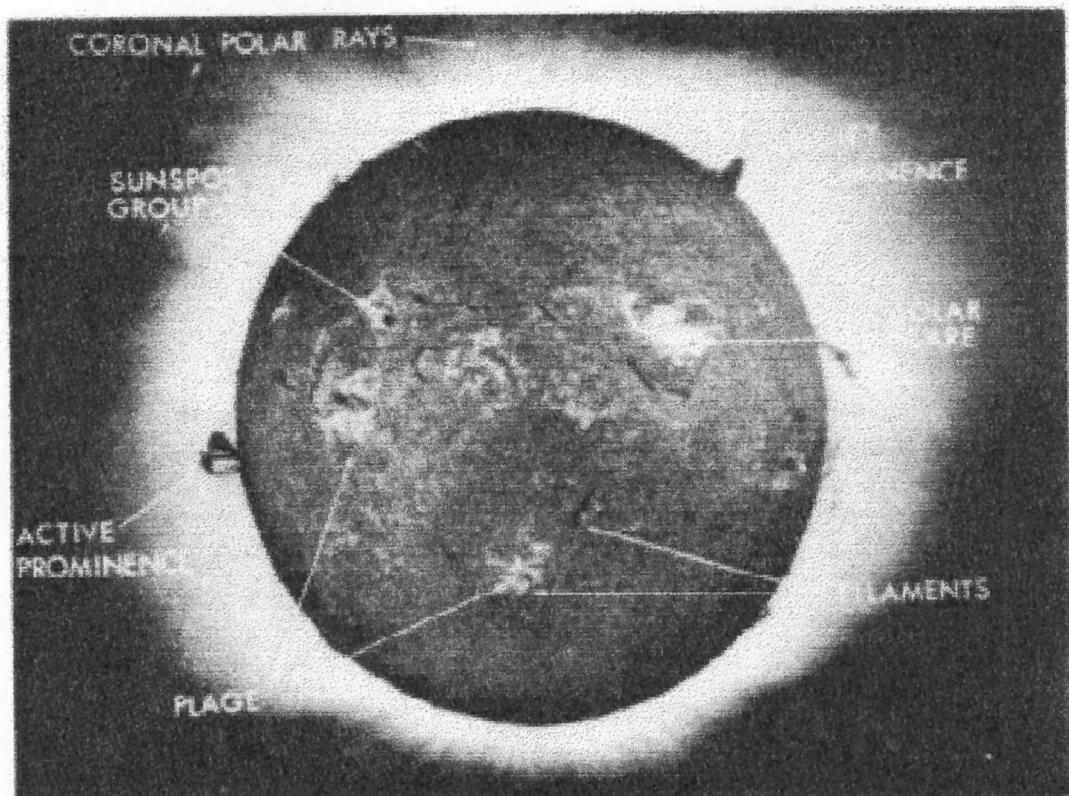


Figure 2.4.3 A photograph of a solar eclipse [8]

Reprinted with kind permission from Kluwer Academic Publishers

2.4.2 Solar Flares

The mechanics of a solar flare are still unknown. Theories on the process also differ. Figure 2.4.4 [9] shows the flaring loop model developed by NASA from the Solar Maximum Mission. The concepts on the process outlined by Smart *et al* [9] involve both an impulsive phase and a gradual phase. This theory suggests that a disruption in the magnetic stress on a loop results in the solar flare. During the impulsive phase magnetic reconnection produces large quantities of electrons and protons in the upper solar atmosphere. This follows a model known as the thick target model. These particles move away from the reconnection site, the corona, toward the lower, more dense part of the sun's atmosphere where they deposit their energy in the plasma. Electrons which interact with the denser material generate bremsstrahlung radiation. They also generate gyrosynchrotron radiation as they gyrate along the magnetic field lines. Energetic ions are involved in nuclear reactions. They knock neutrons out of target nuclei and generate gamma rays. During this energy deposition process the plasma is also heated, the heated plasma then emits ultraviolet and soft x-ray radiation, at this point the gradual phase begins. As the heating decreases the plasma cools by radiation. Figure 2.4.5 [9] shows an idealization of a solar-flare radiative energy intensity-time curve.

During the gradual phase explosive evaporation carries large quantities of high-temperature plasma into the magnetic flux tubes. The energy put into the corona by the flare is gradually moved by thermal conduction to the chromosphere where it is radiated away by hydrogen and helium emission lines. The energy associated with a solar flare is of the order of 10^{32} ergs.

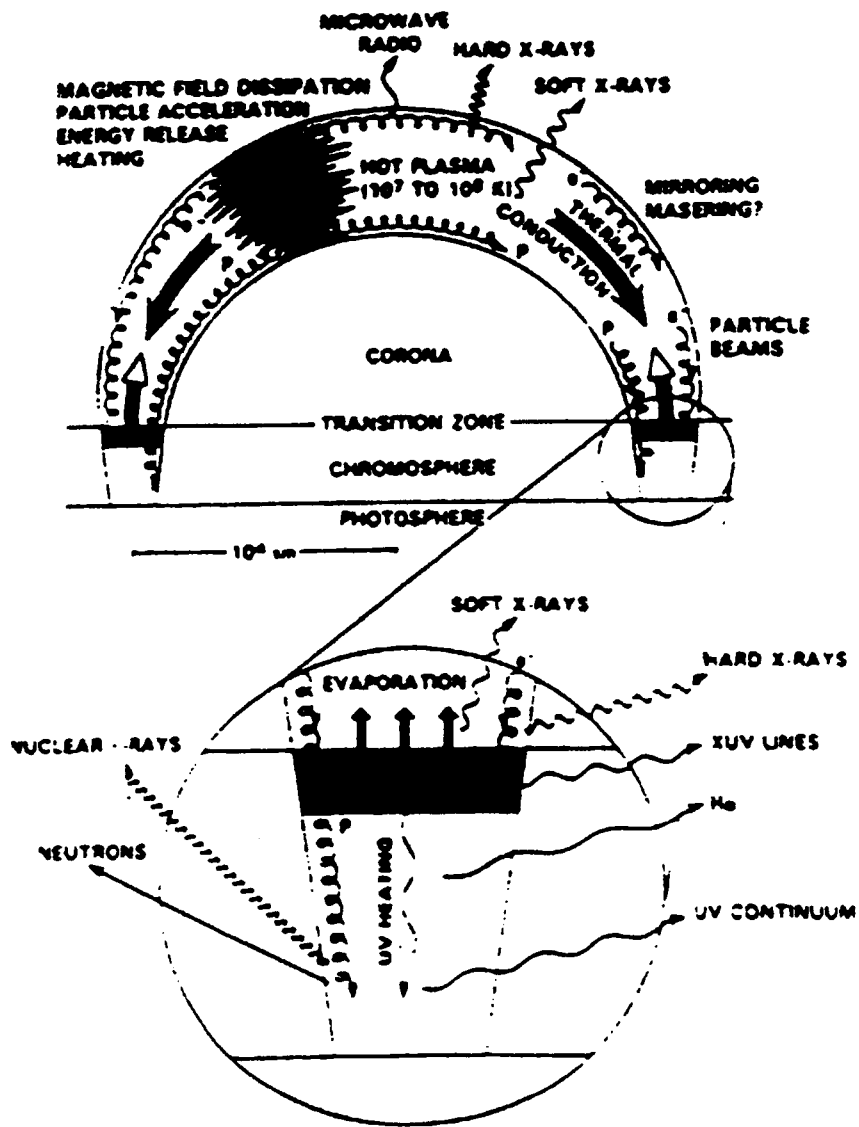


Figure 2.4.4 The flaring loop model [9]

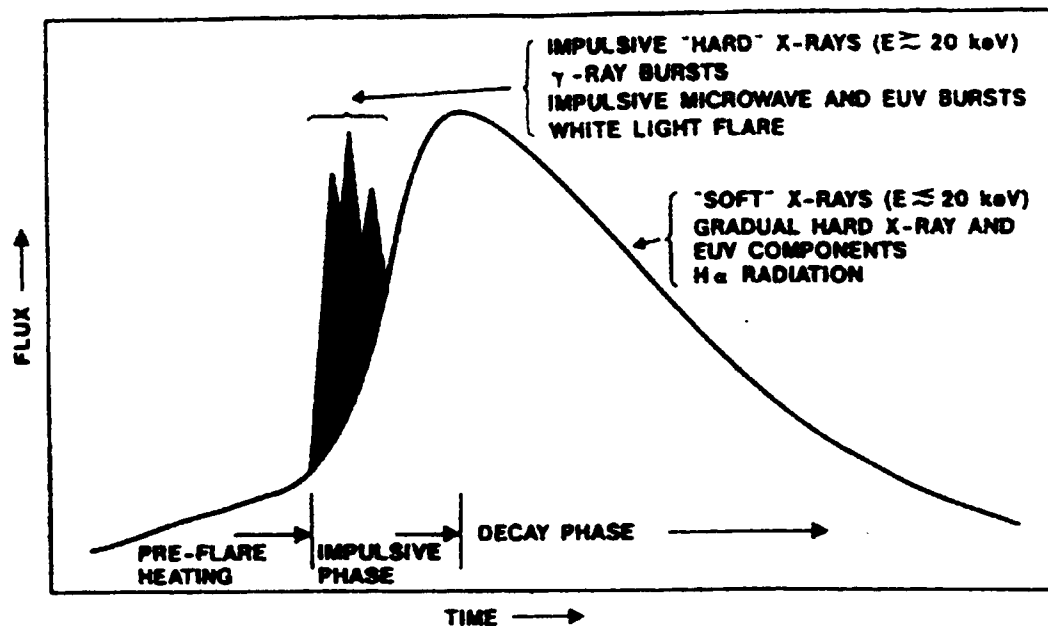


Figure 2.4.5 An idealized solar-flare radiative energy intensity-time curve [9]

2.4.3 Coronal Mass Ejections

Coronal mass ejections are self contained structures of plasma that are embedded in the magnetic field [5]. When ejected into space they perturb the planetary medium. Rather than an explosive energy release, these emissions are believed to be related to large scale changes in solar features such as disappearing filament and streamer blow out. The frequency of the coronal mass ejections also seems to be related to the solar cycle.

During the minimum part of the sunspot cycle the rate is 0.25 per day and at the maximum the rate is 2.5 per day [9]. The total energy involved in a large mass ejection is equal to and may sometimes exceed that of a large solar flare. The velocities of the leading edge of coronal mass ejections can range between 10 and 2100 km s⁻¹, as observed by the Solar Maximum Mission satellite. Thus their velocities are comparable to that of the solar wind. Only about one third of coronal mass ejections travel fast enough to generate interplanetary shock waves. These shock waves impart energy into the interplanetary medium to accelerate electrons and ions to higher energy. It is believed that these accelerated energetic particles are the actual particles that comprise an energetic SPE. Compared to GCRs, SPEs are short-lived, thus radiation exposure to interplanetary crews may be acute. The biological hazards from acute exposures are deterministic (i.e. deterministic effects require a threshold value) rather than stochastic. Typical deterministic effects include fatigue, and mild radiation sickness at whole body doses between 50 cGy and 120 cGy. For whole body doses above 325 cGy the effects are more severe, they include bone marrow depression and even death. The main stochastic effects of concern are carcinogenesis.

3. Dose Calculations

3.1 Introduction

In the space environment high energy particles such as protons, neutrons, and recoiling particles are a source for the radiation quantity known as dose. Dose is defined as the quotient of $\frac{d\bar{\epsilon}}{dm}$ where $d\bar{\epsilon}$ is the mean energy absorbed by a material of mass dm from the incident ionizing radiation [10, 11,12]. The old unit for dose was the rad.

$$1 \text{ rad} = 10^{-2} \text{ J/kg}$$

The S.I. unit for dose is the gray (Gy) where

$$1 \text{ Gy} = 1 \text{ J/Kg}$$

3.2 Methods of Transport

The radiation dose that an astronaut receives inside his spacecraft will be different from that at some arbitrary point outside his spacecraft. The particle fluxes necessary for dose calculations must be transported through the spacecraft's aluminum surface, the astronaut's suit, and then his or her body tissue. The NASA Langley Research Center has developed the BRYNTRN [13] and HZETRN [14] space radiation transport computer codes to characterize the primary and secondary interactions of solar particles and galactic cosmic rays. BRYNTRN is a coupled neutron-proton transport program and

HZETRN is a free space ion and nucleon transport and shielding program. Since this work is primarily concerned with SPEs the discussion is limited to BRYNTRN. This code propagates protons, neutrons, deuterons, tritons, He-3, and alpha particles through the shield material of interest. In this case, the material is aluminum. The nuclear fragments produced by interactions inside the shield material are also taken into account. The code then propagates the incident particles and their fragments through a slab of water. This simulates the particles' interaction with soft human tissue. The energies involved in these calculations are high, thus, a straight ahead approximation is introduced into the 1-D Boltzmann transport equation to give

$$\left[\frac{\partial}{\partial x} - \frac{\partial}{\partial E} S(E) + \sigma_p(E) \right] \phi_p(x, E) = \sum_j \int_E^\infty f_{pj}(E, E') \phi_j(x, E') dE' \quad (3.1)$$

for protons and heavier ions. Since neutrons are uncharged, there is no stopping power contribution $S(E)$. Hence the equation for neutrons is

$$\left[\frac{\partial}{\partial x} + \sigma_n(E) \right] \phi_n(x, E) = \sum_j \int_E^\infty f_{nj}(E, E') \phi_j(x, E') dE' \quad (3.2)$$

where on the left hand side $\sigma_i(E)$ are the macroscopic nuclear absorption cross sections in

cm^{-1} , $\frac{\partial \phi}{\partial x}$ represents the net convective transfer of particles into or out of the material,

$\frac{\partial S(E) \phi_p}{\partial E}$ represents the energy losses due to coulombic interactions, and $\sigma(E) \phi(x, E)$

represents nuclear attenuation. On the right hand side the $f_{ij}(E, E')$ are the differential cross sections for elastic and inelastic collision processes and the $\phi_i(x, E)$ are the fluxes of i type ions at position x moving in direction x with energy E in MeV. These right hand

side terms represent the production of protons or neutrons from the interactions with the target molecules.

Solving eq(3.1) the following is obtained

$$\begin{aligned} \psi_p(x+h, r_p) = & \exp(-\sigma_p h) \psi_p(x, r_p + h) \\ & + \sum_k m_{pk} \sigma_k \frac{v_p}{v_k} \left[\frac{\exp(-\sigma_p h) - \exp(-\sigma_k h)}{\sigma_k - \sigma_p} \right] \psi_k \left(x, r_k + \frac{v_p}{v_k} h \right) \end{aligned} \quad (3.3)$$

This is the basis of the BRYNTRN code where

$$\psi_p(x, r_p) = S_p(E) \phi_p(x, E) \quad (3.4)$$

and $r_p = R_p(E)$, where $R_p(E)$ is the range of protons of energy E , h is the step size, and m_{pk} is the multiplicity of proton p produced by ion k in a collision. Equation (3.2) is solved in a similar manner. The BRYNTRN algorithms have been verified to within one percent accuracy by direct comparison to an exact analytical benchmark solution for a continuous space proton input spectrum [12].

3.3 Dose and Dose Equivalent Calculations

Dose is a quantity that cannot be measured but must be calculated. There are various methods for calculating doses. For the case of energetic particles in deep space dose is calculated using the following equation

$$D_i(x, >E) = A_i \int_E^\infty S_i(E') \phi_i(x, E') dE' \quad (3.5)$$

The equation represents the dose to a point x from particles of energy greater than E

Stopping power is described by the equation:

$$S(E) = \frac{4\pi N Z_p^2 Z_t e^4}{mv^2} \left\{ \ln \left[\frac{2mv^2}{(1-\beta^2)I_t} \right] - \beta^2 - \frac{C}{Z_t} \right\} \quad (3.6)$$

where N is the number of target molecules per unit volume, m is the electron mass, v is the projectile velocity, β is the ratio of the velocity to the speed of light, Z_p is the charge of the projectile, Z_t is the number of electrons per target atom, I_t is the mean excitation energy, and C is a velocity-dependent shell correction term.

The biological effects of radiation are dependent on both the dose and the type of radiation. The quantity relevant to the issue of radiation protection is dose equivalent (H). Dose equivalent has S.I. units of joules per kilogram, similar to dose; however, the name given is sievert (Sv) [10, 11, 12]. If dose is expressed in rad, then the unit for dose equivalent is the rem. To calculate dose equivalent the following equation is used

$$H_i(x, > E) = A_i \int_E^\infty Q(E') S_i(E') \phi_i(x, E') dE' \quad (3.7)$$

where Q is the quality factor that relates absorbed dose to risk. Figure 3.3.1 shows a plot of Quality factor Vs Linear Energy Transfer (LET) [15]. For target fragments and nuclear recoils produced by incident protons the equations for dose and dose equivalent are rewritten as

$$D_p^*(x, > E) = \sum_j \int_j^\infty \overline{E_j} \sigma_{jp}(E') \phi_p(x, E') dE' \quad (3.8)$$

and

$$H_p^*(x, > E) = \sum_j \int_j^{\infty} \overline{E'_j} Q(E') \sigma_{sp}(E') \phi_p(x, E') dE' \quad (3.9)$$

where $\overline{E'_j}$ is the average recoil energy of the j^{th} fragment.

For propagating neutrons the dose and dose equivalent contributions that result from target nuclei recoils or fragments are computed using similar expressions.

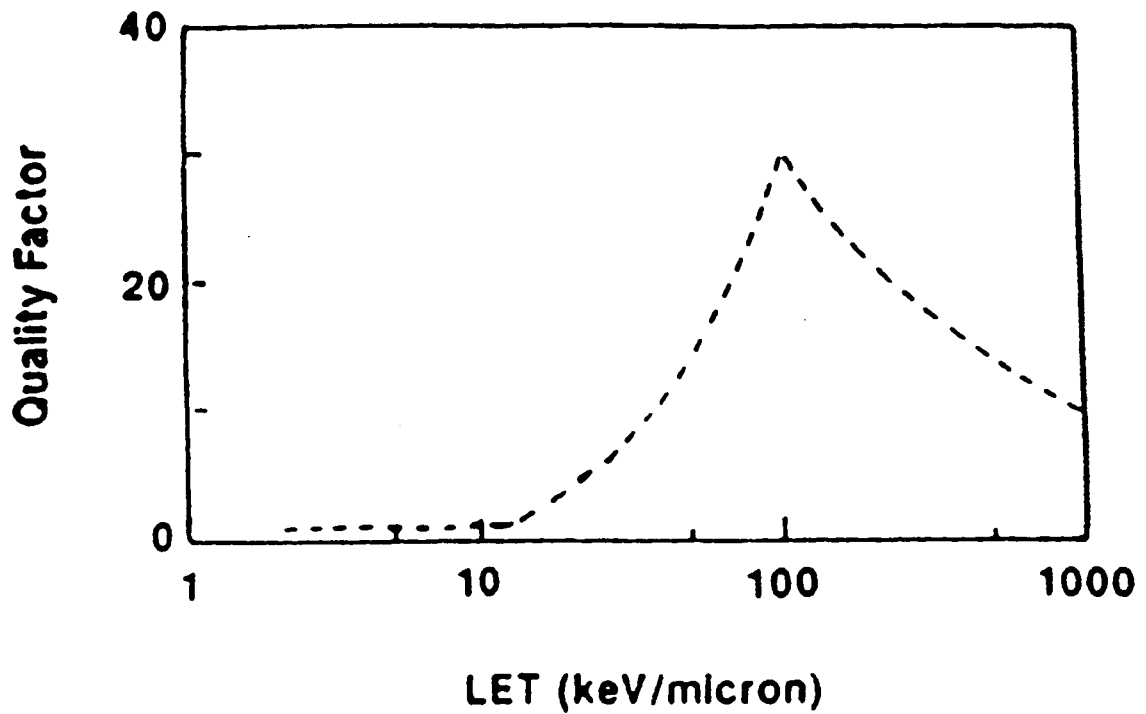


Figure 3.3.1 Quality factor Vs Linear Energy Transfer [15]

3.4 Dose Analyses for SPEs

For this research SPE fluence data were measured by instruments on Geostationary Operational Environmental Satellites (GOES). These SPE spectra at various times since the start of the event are transported using BRYNTRN through 2 g/cm^2 of aluminum shielding, and then through water to simulate the human body's self shielding. The results are dose versus time since the beginning of the event. For dose analysis, protons only are currently used. Alpha particles and heavy ions are neglected because their spectra are not adequately measured. Figure 3.4.1 shows plots of six large solar flare integral fluences from GOES-7 data [7].

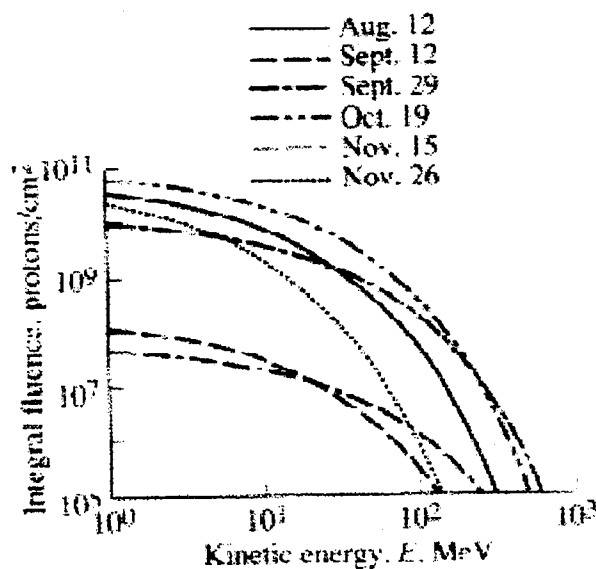


Figure 3.4.1 Six large solar flare integral fluences from GOES-7 data [7]

4. Artificial Neural Networks

4.1 Introduction

An ANN is developed to predict the maximum doses from an event in the space environment. Although nowhere near as complex as the human brain, a neural network is quite capable of performing complex tasks. In order to understand the neurons that make up an artificial neural network, an understanding of the biological model is helpful.

4.2 The Biological Model

In the human brain there is a complex interconnected network of approximately 10^{11} biological neurons [16]. Figure 4.2.1(a) [16] shows a schematic view of a biological neuron. Figure 4.2.1(b) [16] shows a simplified sketch of synapses.

1. The soma is the central structure in which logical functions are performed. The soma also contains the nucleus of the cell and the protein synthesis machinery, thus making the structure also responsible for keeping the entire cell alive.
2. The axon is a highly branched nerve fiber attached to the soma. It serves as the output channel for the neuron. The axon hillock is the initial segment of the axon.

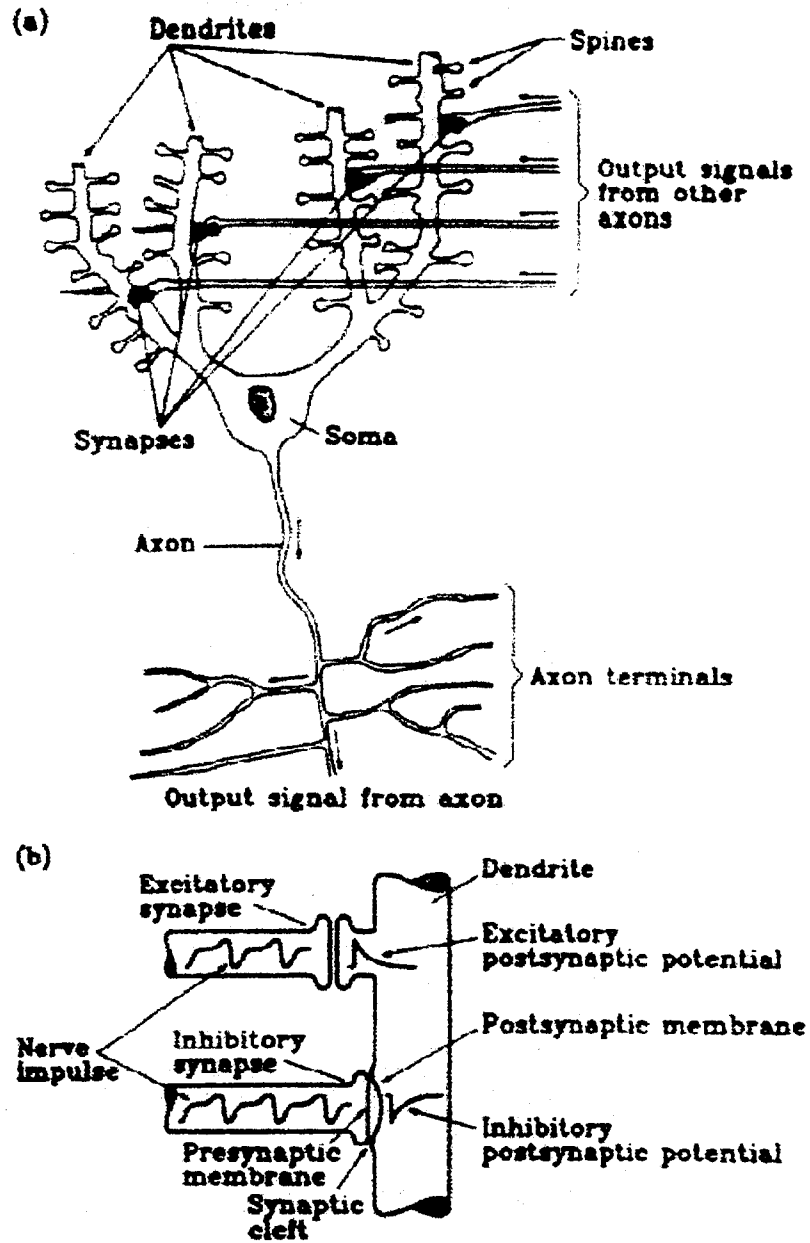


Figure 4.2.1: (a) A schematic view of the biological neuron [16]

(b) A simplified sketch of synapses [16]

Reprinted with kind permission from John Wiley & Sons Limited

At this structure, the signals are converted into nerve pulses and propagated along the axon to the target cells such as other neurons, receptors, and muscles.

3. The dendrites are branchlike structures that connect one neuron to other neurons. A neuron usually has between 10^3 and 10^4 dendrites connected to it at the soma. Dendrites receive their inputs from other neurons via contacts that are called synapses. Dendrites are the media that carry the input signals to the neuron and that carry the conducting signals from the soma.
4. Synapses act as interfaces between axons and dendrites. They can change a dendrite's local potential from positive to negative. Synapses can work either as excitors or inhibitors, increasing or dampening neuron excitation. Information is supposedly stored in a neuron in the neuron's synaptic connections. In humans, synapses are primarily chemical in nature while in less advanced animals, synapses are based on electrical transmission.

The soma receives inputs from other neurons via adaptive synaptic connections to dendrites. Output signals from a cell are transmitted along a branching axon to synapses of other neurons [16]. The output pulse rate is dependent on the input signal's strength and the strength (weight) of the corresponding synaptic connection. The maximum rate of fire is about 1000 pulses per second. The output signal is proportional to a linear combination of the input signals. Neurons operate in a combination of analog and digital form. Information transmitted between neurons is in the digital form and the encoded information is in the form of an analog signal. Pulses that arrive at the synapse generate an analog internal potential that is proportional to the synaptic strength which is either

positive or negative depending on whether the synapse is exciting or inhibiting the neuron. The potentials are then summed in a spatial-temporal way. Should the total potential exceed some total value or threshold then a series of pulses is generated which travel along the neuron's axon [16]. Artificial neurons process signals using the same fundamental methods as the biological model, thus making them useful for function approximation.

4.3 The Basic Neuron Model

There are several models of artificial neurons. For the sake of simplicity the discussion is limited to the model used in this work. The basic neuron is a multi-input nonlinear device with weighted interconnections w_{ji} also referred to as synaptic weights. Representing the cell body is the activation function $\psi(u_j)$. The mathematical model of the basic neuron is

$$y_j = \psi \left[\sum_{i=1}^n w_{ji} x_i + \Theta_j \right] \quad (4.1)$$

where Θ_j is the external threshold or bias, w_{ji} are weights, x_i are inputs ($i = 1, 2, \dots, n$), n is the number of inputs, and y_j is the output. Figure 4.3.1 [16] is a schematic diagram of the basic neuron model. In very early models a binary, hard-limiting activation function was used. The weighted sum of all the inputs was compared to a threshold, Θ_j . If the sum exceeded the threshold, then the output was set to the high value or logic 1; otherwise, the output was set to the low value or logic 0.

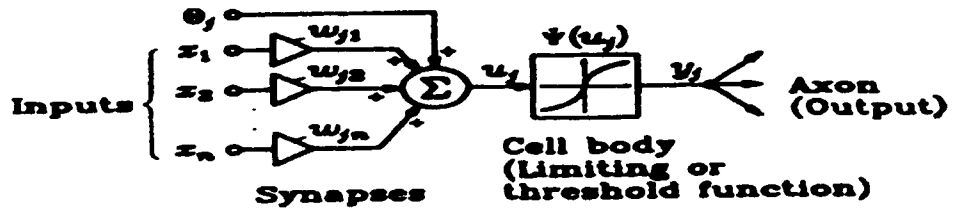


Figure 4.3.1 A schematic diagram of the basic neuron model [16]

Reprinted with kind permission from John Wiley & Sons Limited

Usually the threshold function is a more general nonlinear function so that the output of the neuron can either be a discrete or continuous value between -1 and 1. In this case the output signal is determined by an increasing sigmoid function of a weighted sum of the input signal. This sigmoid function can either be symmetrical around zero

$$y_j = \tanh(\gamma u_j) = \frac{(1 - \exp(-2\gamma u_j))}{(1 + \exp(-2\gamma u_j))} \quad (4.2)$$

or non symmetrical

$$y_j = \frac{1}{(1 + \exp(-\gamma u_j))} \quad (4.3)$$

where $u_j = \sum_{i=0}^n w_{ji} x_i$ and γ is a positive constant that determines the steepness of the sigmoid function's slope. For this work γ has a value of 1 and the symmetric sigmoid function is used. Figures 4.3.2 and 4.3.3 are plots of the symmetrical and non symmetrical sigmoid functions respectively.

4.4 Network Architecture

4.4.1 Feedforward Multilayer Perceptron

Multilayer Perceptrons (MLP) are feedforward neural networks in which the neurons are arranged in layers. Neurons from one layer are connected to neurons in adjacent layers via unidirectional links. The weights represent these links and they act as signal multipliers. Figure 4.4.1 [17] shows a schematic of a three layer perceptron. The neurons in the input layer do not perform any calculations, but feed signals to neurons in the first hidden layer, which feed the second hidden layer, which then feed the neurons in the output layer. Layers between the input and output layers are called hidden layers. According to the Universal Approximation Theorem [17, 18, 19, 20, 21], a single hidden layer is sufficient for a multilayer perceptron to perform computations for a given training set represented by a set of inputs and a desired output function [17]. There is no theoretical limit to how many hidden layers are used in a network architecture, however, in practice, one or two layers are used. This is done to reduce the chance of overfitting. In a feedforward MLP each neuron has one output and multiple inputs. Connections from higher layers to lower layers are not permitted, neither are connections between neurons in the same layer.

In the MLP the individual neuron's output is computed like that of the basic neuron model:

$$y_j = \psi [\sum w_{ji} x_i + \Theta_j] \quad (4.1)$$

where ψ is the activation function, Θ_j is the external threshold or bias, w_{ji} are weights, x_i are inputs ($i = 1, 2, \dots, n$), n is the number of inputs, and y_j is the output.

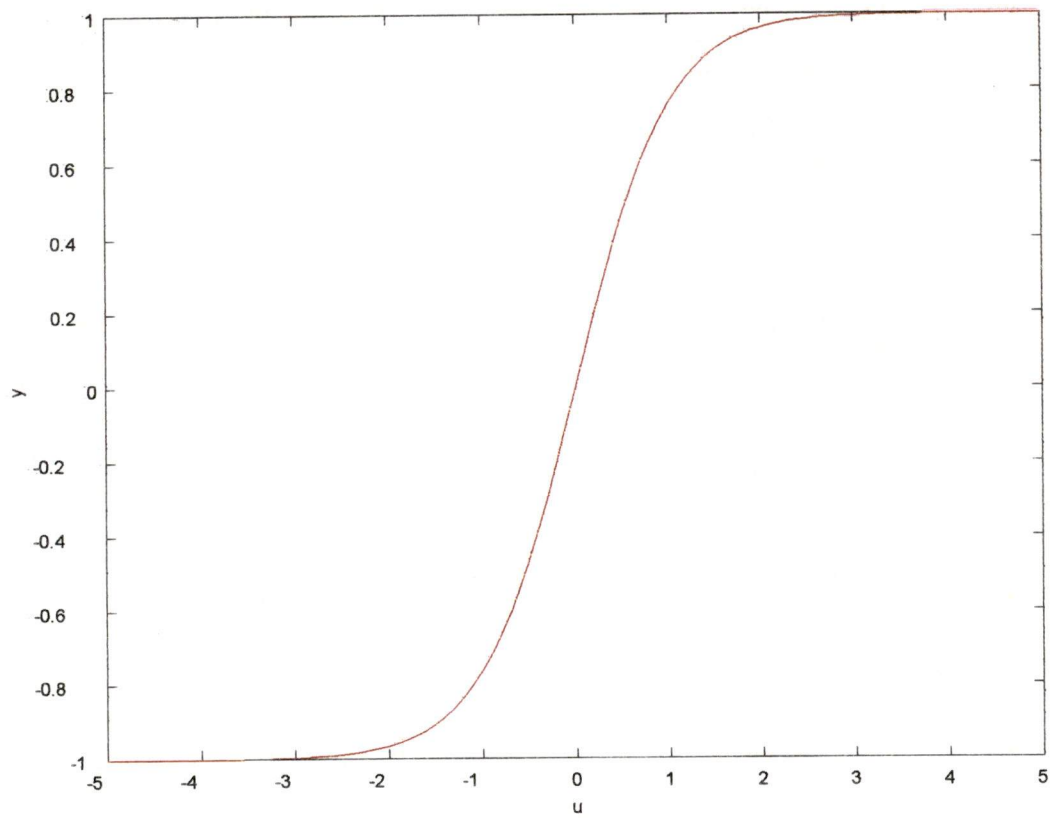


Figure 4.3.2 A plot of the symmetric sigmoid function $y_j = \tanh(u_j) = \frac{(1 - \exp(-2u_j))}{(1 + \exp(-2u_j))}$

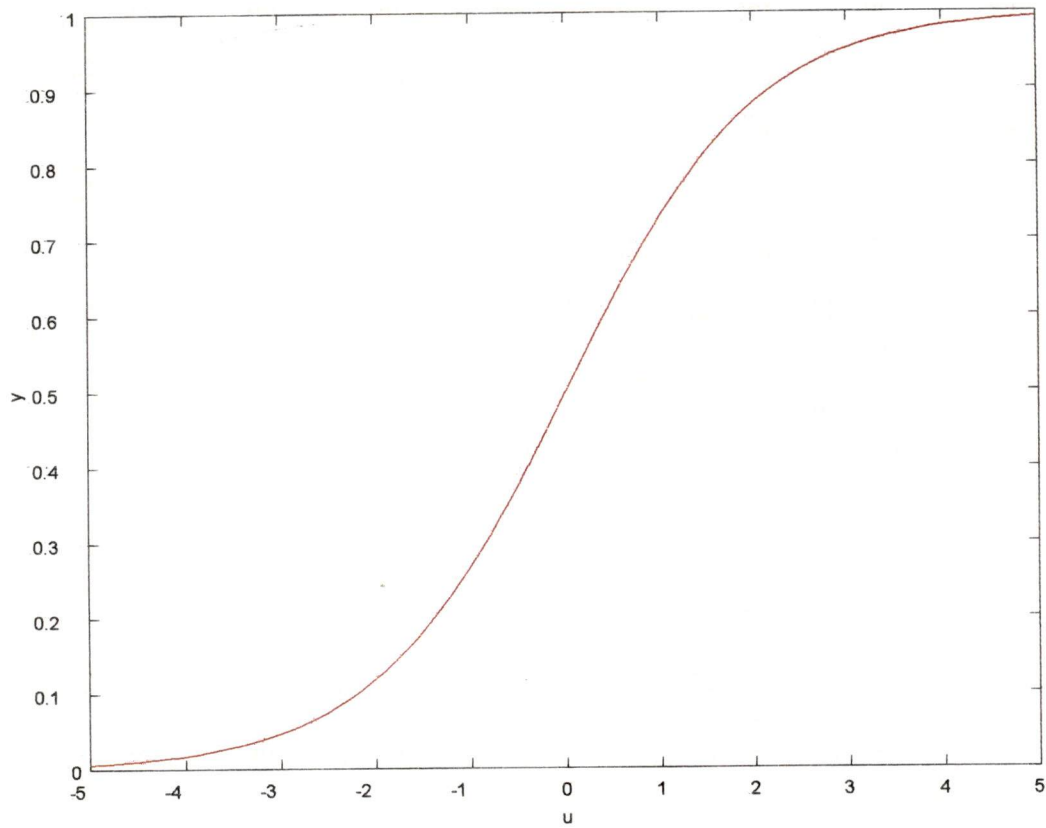


Figure 4.3.3 A plot of the nonsymmetric sigmoid function $y_j = \frac{1}{1 + \exp(-\gamma u_j)}$

The values of the weights determine network behavior and its ability to process input signals. These values are kept constant during computation. To obtain the desired network response, the optimal values of the weights must be properly calculated. This requires that the MLP must be trained. This process is called the learning or training process. During this process information is back-propagated through the network in order to update the weights in successive layers. Thus weights in the output layer are updated first, followed by the weights in second hidden layer and then the weights in the first hidden layer.

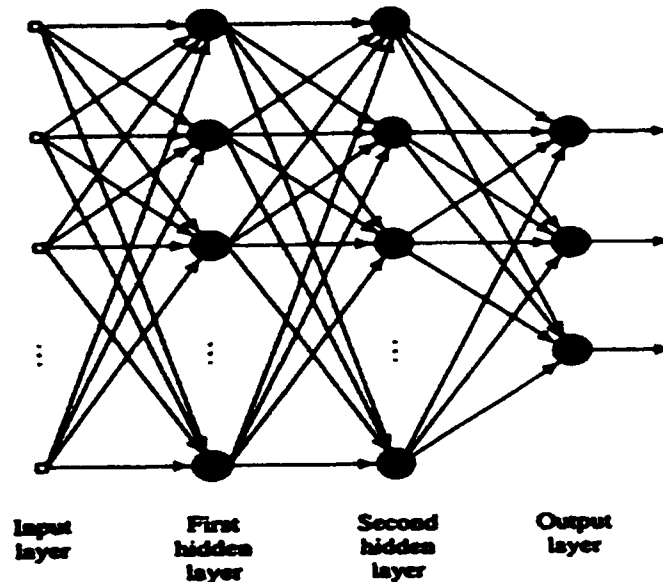


Figure 4.4.1 A schematic diagram of a three layer MLP [17]

Haykin, Simon, NEURAL NETWORKS: A Comprehensive Foundation, Copyright 1994, p.141. Reprinted by permission of Prentice Hall, Upper Saddle River, New Jersey.

4.4.2 Time Delay Neural Networks

Standard MLPs process a static mapping. In this research the dose profiles that the network is trying to approximate are temporal in nature. The final dose may be estimated by using doses sampled from several different times. A brief discussion on ANNs that do temporal processing is necessary, specifically, Time-Delay Neural Networks. This type of network, shown in Figure 4.4.2, was originally designed for speech recognition. Time-Delay Neural Networks are multilayer feedforward networks in which the inputs are both past and present signals. By using both present and past inputs temporal behavior is learned [22]. For this application, the measured signals are delayed and used as input to the network.

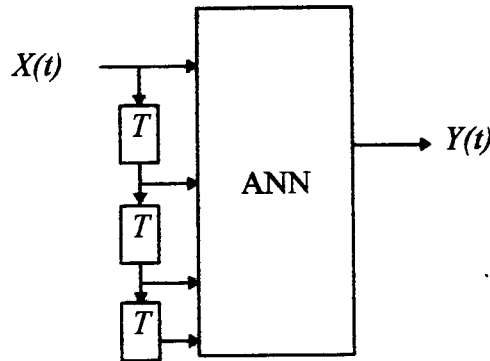


Figure 4.4.2 Time Delay Neural Network

The network type used in this research is the Sliding Time-Delay Network (STDNN), as shown in Figure 4.4.3. This is a variation on the TDNN. In the case of the STDNN the time delays vary. In the standard TDNN the delays are constant. The inputs are the values of the time dependent function $f(t)$, with τ being the

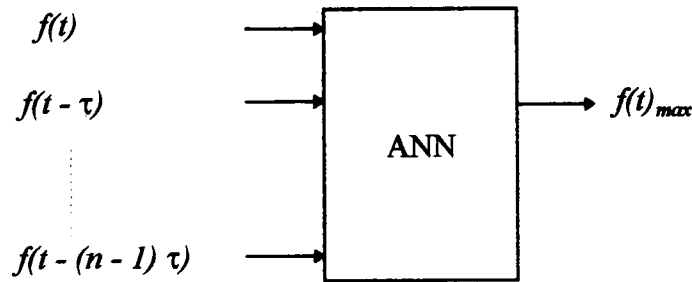


Figure 4.4.3 Sliding Time Delay Neural Network

fractional size of an arbitrary time interval T , where $T = n\tau$, and n is the number of input neurons. There are no feedback terms in an STDNN so training is accomplished using the standard algorithms.

4.5 Training the Artificial Neural Network

The disparity between the function that the network is trying to approximate and the actual function the network generated is represented by the value of the sum-squared error. The error generated by an arbitrary j th neuron is represented by the following expression:

$$e_j(n) = d_j(n) - o_j(n) \quad (4.4)$$

where $e_j(n)$ is error, $d_j(n)$ is desired output, and $o_j(n)$ is actual output of the j th neuron(n). For this work there is only one output neuron. The value of the sum-squared error is expressed as follows

$$E = \sum_j e_j^2(n) \quad (4.5)$$

where E is the sum of squared errors. Thus if the training set had eighty training patterns, and an error of $\sqrt{2}$ was tolerated for each training pattern, then the sum-squared error is 160. A target value of sum squared error is called a training parameter.

Training a neural network can be viewed as a mathematical optimization problem. The minimization of the sum squared error E , which is also a function of the network's weights and biases, is the desired objective. Letting $\mathbf{x}$ represent the vector of weights and biases, and $\Delta\mathbf{x}$ represent the change in weights and biases to reduce E , the minimum error E_{min} is written as follows

$$E_{min} = f(x_1, x_2, x_3, \dots, x_n) = f(\mathbf{x}) \quad (4.6)$$

Gradient methods for solving eq(4.6) utilize the Taylor Series expansion in vector form:

$$f(\mathbf{x} + \Delta\mathbf{x}) \approx f(\mathbf{x}) + \mathbf{g}^T \Delta\mathbf{x} + \frac{1}{2} \Delta\mathbf{x}^T \mathbf{H} \Delta\mathbf{x} + \dots \quad (4.7)$$

where $\mathbf{g}$ is the Jacobian gradient vector, ∇f is the gradient vector of a row vector of partial derivatives of the function f :

$$\nabla f = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \dots \quad \frac{\partial f}{\partial x_n} \right] \quad (4.8)$$

and the transpose of the gradient vector is the Jacobian gradient vector

$$\mathbf{g}^T = \nabla f \quad (4.9)$$

and $\mathbf{H}$ is the Hessian Matrix

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & & & \vdots \\ \vdots & & & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \dots & \dots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} = E''(\mathbf{x}) \quad (4.10)$$

The activation functions of an ANN must be continuous and differentiable, which means that the first and second order partial derivatives exist. Thus at local and global minima, $\mathbf{g}$ is a null matrix, and $\mathbf{H}$ is positive definite.

4.5.1 Method of Steepest Descent

Steepest descent is a first order method of optimization. In this method the Jacobian gradient is used to determine the direction movement necessary to reduce the error. The change in error (ΔE) is defined as follows:

$$\Delta E = \mathbf{g}^T \Delta \mathbf{x} \quad (4.11)$$

This is a first order approximation. In order to obtain a maximum reduction in E in the direction $\mathbf{g}$, a line search is performed. From this line search $\Delta \mathbf{x} = \lambda \boldsymbol{\phi}$ is determined, where $\boldsymbol{\phi}$ is a unit vector in the direction of $-\mathbf{g}$. To determine a value for λ , it is raised until the error is a minimum in the direction $\boldsymbol{\phi}$. Line searches are computationally intensive. This is due to the fact that E must be evaluated several times to reach the minimum in the direction $\boldsymbol{\phi}$. The rationale for repeating these evaluations is that the

global minimum may not be in the direction of the initial negative gradient. Figure 4.5.1 shows the steepest descent method with line search. The pathway to the minimum is very random and near the minimum learning is slow. In Figure 4.4.1 each new line search is perpendicular to the previous

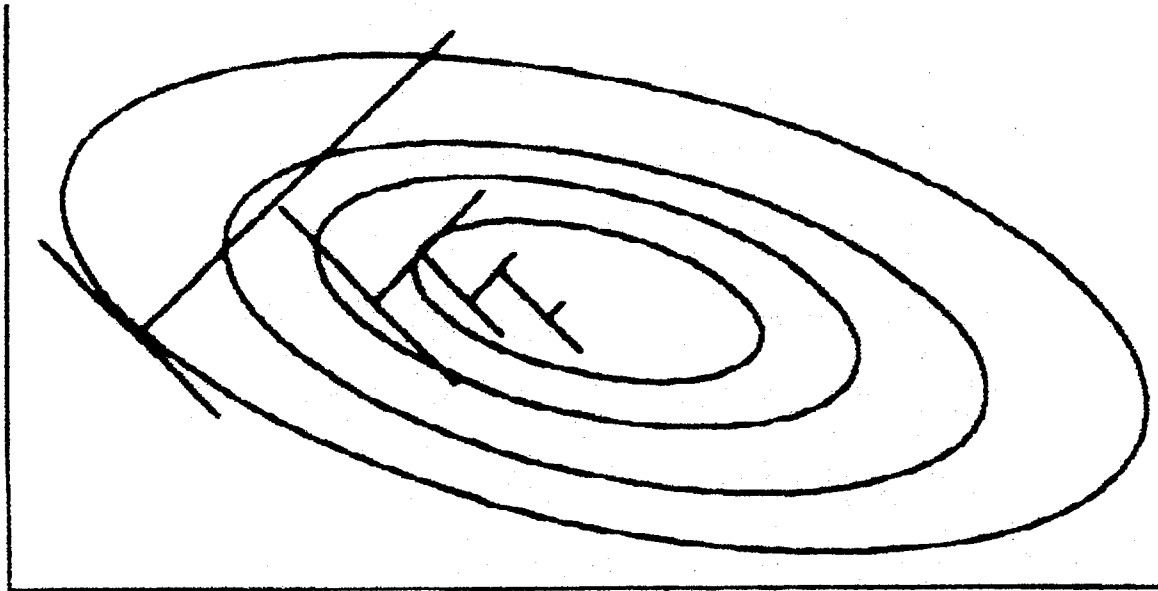


Figure 4.5.1 The Steepest Descent Method with Line Search

and the length of movement gets smaller in the vicinity of the minimum. This means that convergence near the minimum is slow and that is a disadvantage of this method.

4.5.2 Gauss-Newton Method

This Gauss-Newton Method uses an approximation of the Hessian matrix in order to obtain second order information about the error surface. The error function E is the sum of squares of each error generated by the corresponding output for each training vector

$$E = \sum_{k=1}^m [f_k(x)]^2 \quad (4.12)$$

Defining $\mathbf{f}$ as follows

$$\mathbf{f}^T = [f_1(x) f_2(x) f_3(x) \dots f_m(x)] \quad (4.13)$$

we obtain

$$E = \mathbf{f}^T \mathbf{f} \quad (4.14)$$

The next step is to define a matrix $\mathbf{A}$. This matrix comprises variations of each m the term in the error function due to variations of each parameter n : the weights and biases.

$$\mathbf{A} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \quad (4.15)$$

The first partial derivative of the error equation eq(4.12) with respect to each parameter n yields the Jacobian vector in terms of $\mathbf{A}$ and $\mathbf{f}$:

$$\mathbf{g} = 2 \mathbf{A}^T \mathbf{f} \quad (4.16)$$

The second partial derivative with respect to each parameter yields

$$\mathbf{H} = 2\mathbf{A}^T \mathbf{A} + \text{a second order term} \quad (4.17)$$

Now using the Newton method [16] we solve for any movement $\Delta \mathbf{x}$ towards $\mathbf{x}_{\min}$ as follows

$$\Delta \mathbf{x} = -\mathbf{H}^{-1} \mathbf{g} \quad (4.18)$$

By substitution of eq(4.16) and eq(4.17) into eq(4.18) and ignoring the second order term in eq(4.17) the following is obtained

$$\Delta \mathbf{x} = -[\mathbf{A}^T \mathbf{A}]^{-1} \mathbf{A}^T \mathbf{f} \quad (4.19)$$

4.5.3 Levenberg-Marquardt Method

There are many methods used to train an artificial neural network. The method of interest in this work is the Levenberg-Marquardt method [23, 24, 25]. The Levenberg-Marquardt method makes use of both the reliable convergence of the steepest descent method on the error surface when far from the minimum, and the rapid convergence of the Gauss-Newton method [16] when close to the minimum. This is accomplished by the use of a variable metric λ

$$\Delta \mathbf{x} = - [\lambda \mathbf{I} + \mathbf{H}]^{-1} \mathbf{g} \quad (4.20)$$

where $\mathbf{I}$ is the identity matrix.

Substituting eq(4.16) and eq(4.17) into eq (4.20) we obtain

$$\Delta \mathbf{x} \cong - [\lambda \mathbf{I} + \mathbf{A}^T \mathbf{A}]^{-1} \mathbf{A}^T \mathbf{f} \quad (4.21)$$

When $\lambda \rightarrow \infty$, $\Delta \mathbf{x}$ approaches $-\mathbf{A}^T \mathbf{f}$ (Steepest Decent Method)

When $\lambda \rightarrow 0$, $\Delta \mathbf{x}$ approaches $[\mathbf{A}^T \mathbf{A}]^{-1} \mathbf{A}^T \mathbf{f}$ (Gauss-Newton Method)

To minimize E , Marquardt [24] proposed a method for varying the metric λ . The metric λ is multiplied by ten, moving towards the Steepest Descent Method, until the error is reduced and incrementing. The method then divides λ by ten checking to see whether the new increment $\Delta \mathbf{x}$ has reduced the error. Should the error not be reduced λ will be multiplied again until the error is reduced. When $\Delta \mathbf{x}$ is reducing the error, the method will move towards Gauss-Newton. When $\Delta \mathbf{x}$ is not reducing the error, the method will move towards steepest descent.

Levenberg-Marquardt [23, 24, 25] is chosen over standard Backpropagation [16, 17, 19] as the training algorithm. Levenberg-Marquardt has been shown to converge much more rapidly than Backpropagation for normal sized networks and training sets.

4.5.4 Important Training Issues

After a network is trained it is tested with data that are not used for training. A network's ability to give a correct output for an input that is not used during the training process, is called its ability to generalize [17, 18]. These input and output patterns are referred to as test data or test sets and must originate from the same or similar population that generated the training set.

Training a neural network is considered to be a curve fitting problem, as shown in Fig 4.5.2, where the data are properly fitted. Generalization is a result of good non-linear interpolation of the input set [17]. Multilayer Perceptrons (MLPs) with continuous activation functions are useful for generalization because they yield output functions which are also continuous. MLP networks that have good generalization properties produce the desired input to output relationship even if the test input set differs slightly from the training input set.

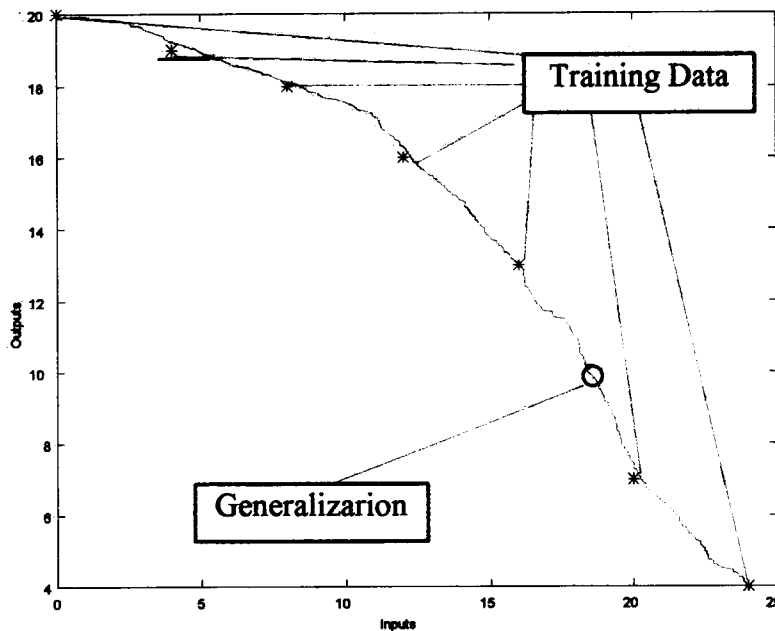


Figure 4.5.2 Properly fitted data

A network that has learned much more input/output relationships than necessary is said to have been overtrained. The network memorizes only the training data, including any electronic noise in the data, thereby reducing its ability to generalize between similar input/output patterns. This can occur when there are too many hidden neurons or hidden layers. This can also occur when training is allowed to continue to unreasonable error goals. The result is that the network overfits the data. An example of overfitting is illustrated in Fig 4.5.3.

To avoid the chance of overfitting the following methods are recommended [21]:

1. Reduce the number of hidden neurons. This reduces the number of degrees of freedom in the network and reduces the chance of overfitting.

2. Train the network with a realistic error goal.
3. Increase the size of the training set. In larger training sets, the noise tends to be averaged out. Thus, the properties of the training function and not its noise are learned.
4. Use cross validation training to identify when overfitting occurs. In cross validation training two data sets are used during training. The first set is used for training the network and the second set is used to check for overfitting. Due to the fact that overfitting occurs when the network learns the training data better than it would for other data, the checking data is used during the training process to check for behavior [21].

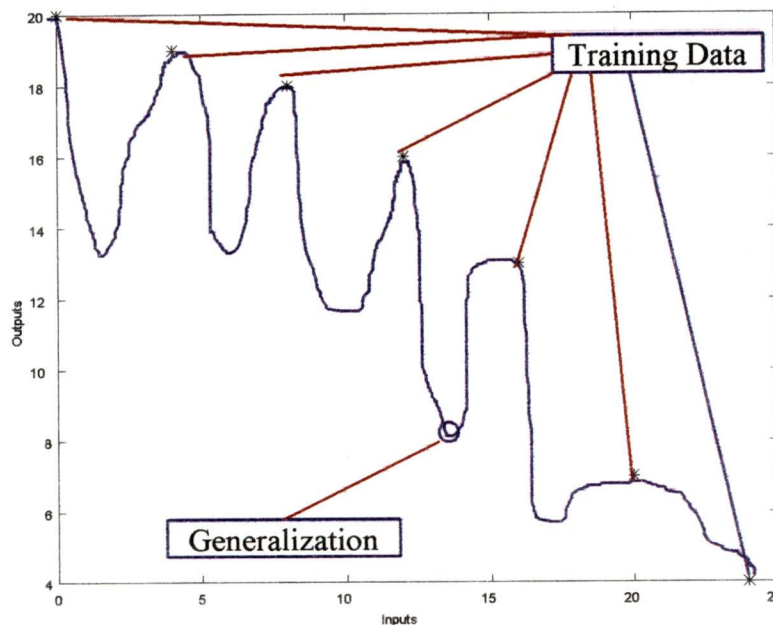


Fig 4.5.3 Overfitted data

The other training issue of relevance to this research is the issue of local and global minima. During the training process the network's objective is to reach the global minimum of the error function. Despite the convergence properties of gradient descent training algorithms, there is no guarantee of reaching the global minimum when non-linear activation functions are used. Following the negative of the derivative of the error function with respect to weights and biases can sometimes cause the network to become stuck in a local minima instead of reaching the global minima. This is a stochastic process. Figure 4.5.4 shows a sketch of an arbitrary error function with local and global minima. To remedy the problem of local minima, training is restarted using new random weights and biases, in the hope that a local minimum will be avoided.

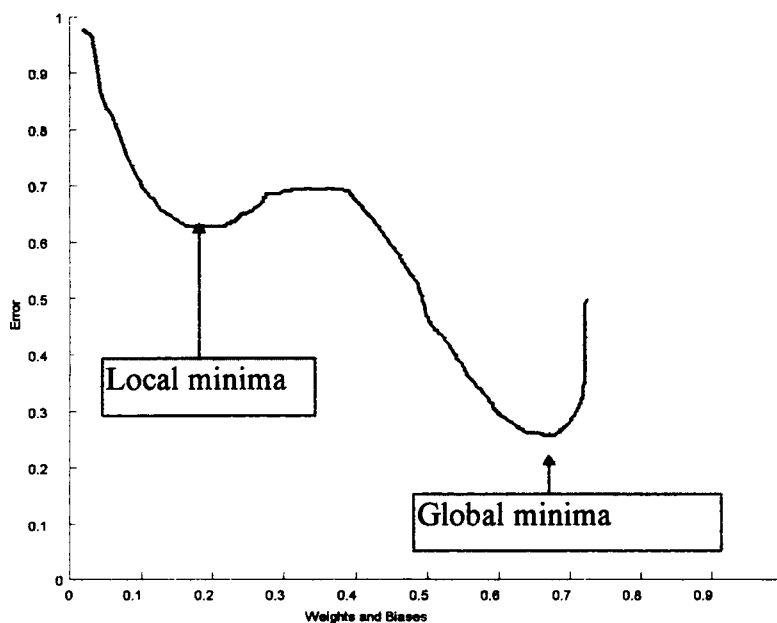


Figure 4.5.4 A plot of an error surface with both local and global minima

5. Methodology

5.1 Objective

The objective of this research is to determine the feasibility of using an artificial neural network to predict the maximum dose to astronauts early on during a major Solar Particle Event. Details of how this is accomplished are presented in this chapter.

5.2 The Data Set

Calculations of skin dose versus time are used in the training of the neural networks. These data originate from particle fluence data collected during a series of solar particle events dating between 1989 and 1994. The SPE fluence data were measured by instruments on GOES-6 and GOES-7. To obtain the dose-time data, the SPE integral fluence spectra over time is transported through 2 g/cm² of aluminum shielding. This represented an astronaut's space suit. These modified spectra are then transported through water to simulate the human body's self shielding. Work done by Badhwar [3] at NASA Johnson Space Center indicated that the data could be modeled by a Weibull function

$$D(t) = D_{\infty} (1 - \exp(-\alpha t^{\beta})) \quad (5.1)$$

where $D(t)$ is the dose at time t , D_{∞} is the maximum absorbed dose, and α and γ are fitting parameters. Figure 5.2.1 [26] shows a Weibull function fit for the SPE that occurred in September 1989. The filled squares are the calculated skin doses, the open

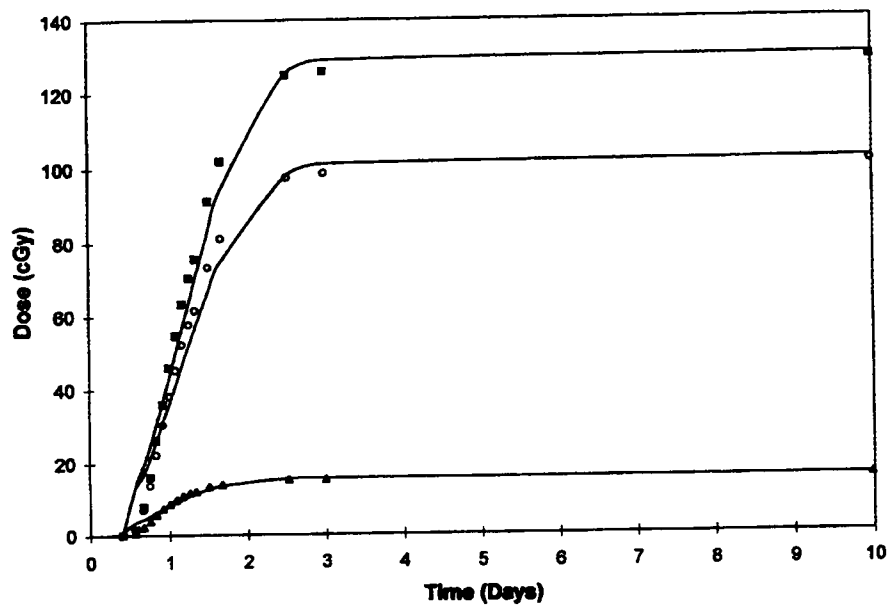


Figure 5.2.1 Weibull Fit of 9/89 SPE dose [26]

circles are calculated ocular lens doses, and the filled triangles are calculated bone marrow doses. The solid lines are the fits obtained using least squares regression techniques. Table 5.2 shows the fitted D_{∞} 's, α 's, and γ 's for selected events between 1989 and 1994. From the table there were fifteen D_{∞} 's each for eyes, skin, and the blood

forming organs. Thus there are forty-five D_{∞} 's available for use. Once in this functional form, the data training and test sets are generated.

MATLAB [Copyright 1984 - 1994 The MathWorks Inc.] software is used to train and test the artificial neural network. A MATLAB script [Appendix 1] was written which converts the data in Table 5.2 into the corresponding Weibull functions. Thus a total of forty-five such models are generated. The Weibull functions generated are stored in a 1 x 100 matrix. This is done by letting time (t) range from zero to 4400 minutes in time steps of forty-four minutes. Figure 5.2.2 shows a typical Weibull curve generated by the Weibull generating MATLAB script [Appendix 1]

Table 5.2 Fitted D_{∞} 's (labeled Skin, Eye and BFO), α 's, and γ 's for selected events between 1989 and 1994

YEAR	SOLAR PARTICLE EVENT	LOCATION	SKIN	EYE	BFO	ALPHA	GAMMA
1989	MAR 17/1855 TO MAR 27/0000	N83/W60	2.726392	1.023816	0.025778	1.00E-04	1.95
1989	AUG 12/1630 TO AUG 22/0000	S18/E37	185.4835	88.5949	8.003866	1.00E-10	3.2
1989	SEPT 12/1800 TO SEP 22/0000	S18/W79	0.24118	0.147437	0.023507	1.00E-06	1.9
1989	SEPT 29/1205 TO OCT 9/0000	S28/W90	146.1551	107.3608	18.44648	1.00E-06	1.95
1989	OCT 19/1305 TO OCT 29/0000	S27/E10	457.5358	305.8578	47.19525	1.00E-06	1.66
1989	NOV 15/0735 TO NOV 25/0000	N11/W28	0.599286	0.493477	0.10098	3.80E-03	0.9
1989	NOV 30/1345 TO DEC 6/0000	N28/W59	14.34295	6.003807	0.237253	1.00E-11	3.45
1991	MAR 23/0820 TO MAR 28/0000	S29/E28	190.5048	94.05822	6.084993	1.00E-10	3.17
1991	JUNE 11/0230 TO JUNE 15/0000	N30/E70	40.26891	26.01857	3.55424	1.00E-07	2.35
1991	JUNE 15/0825 TO JUNE 21/0000	N33/W69	27.58589	17.5841	2.480967	1.00E-04	1.32
1991	JULY 7/0455 TO JULY 17/0000	N28/E03	0.995784	0.403118	0.024873	1.00E-08	1.87
1992	MAY 9/1005 TO MAY 18/0000	S28/E08	1.45738	0.623851	0.045546	1.00E-05	1.5
1992	JUNE 25/2045 TO JUNE 30/0000	N09/W67	5.158972	2.975845	0.380155	8.00E-05	1.38
1992	OCT 30/1920 TO NOV 9/0000	S22/W61	79.28634	46.81602	5.42332	1.00E-06	1.8
1994	FEB 20/0300 TO FEB 25/0000	N09/02	0.474763	0.229816	0.024878	1.00E-05	1.8

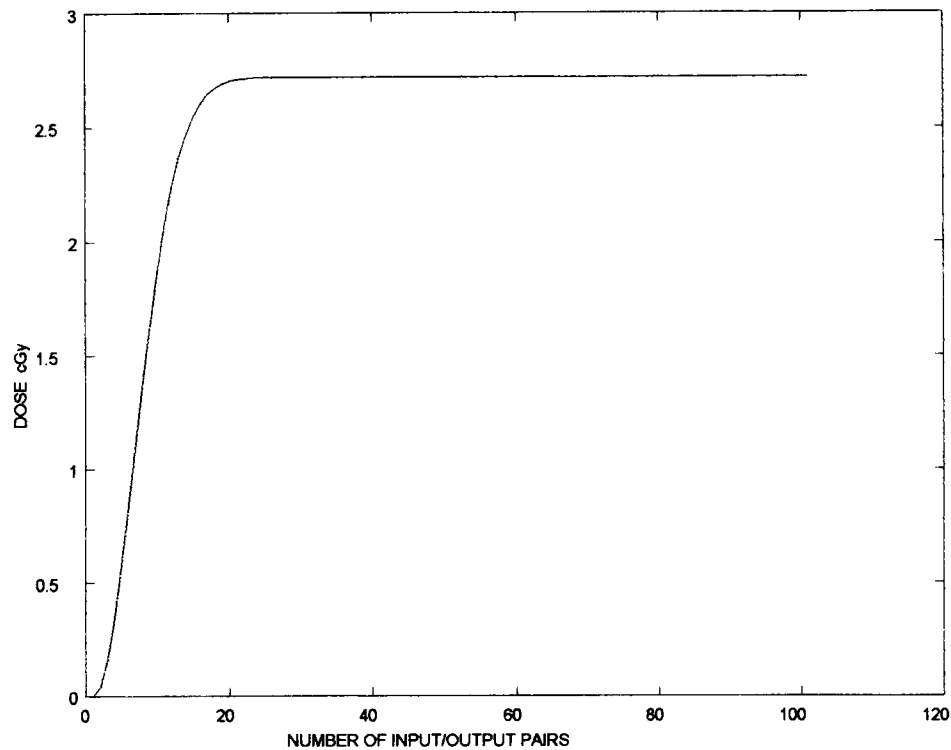


Figure 5.2.2 A Weibull curve generated by the MATLAB script in Appendix 1

5.3 Generating the Training and Test Sets

As stated in Chapter 4 the type of neural network used in this research is a Sliding Time-Delay Neural Network (STDNN). Therefore the data sets must be in a form that the STDNN can process. A second MATLAB script [Appendix 2] is written which further converts the data sets into a form that can be used for training and testing purposes. The Weibull functions generated by the previous script formed the input set. The second script [Appendix 2] samples the input set at a certain time interval T , as shown in Figure

5.3.1. For this research, the training inputs to the network are equally spaced samples from the initiation of an SPE to an arbitrary time t . At this time t the output would be the D_{∞} (maximum dose) at the end of the event. Therefore at any time t the network could approximate the maximum dose the astronaut receives. Using the relationship $T = n\tau$, where n , number of samples acquired along the curve of the specific Weibull function,

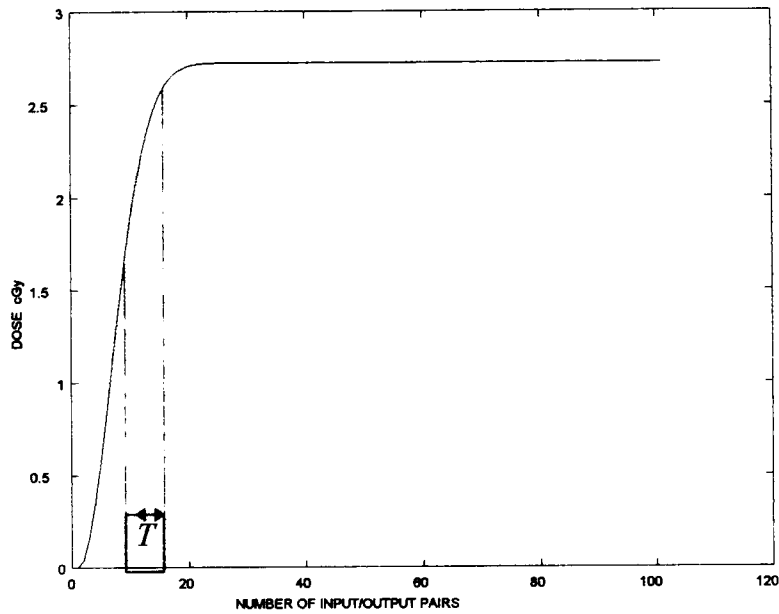


Figure 5.3.1 A Weibull curve with a sample along the interval T

correspond to the number of neurons in the input layer, and τ is the sliding time delay of the interval, dose values at arbitrary time $t, t - \tau, \dots, t - (n - 1)\tau$ are obtained. The script generated input and output training sets. The number of samples in an interval, performed along the Weibull curve corresponds to the number of training inputs to the

STDNN, e.g. five points along the Weibull curve corresponds to five input neurons which in turn are represented by a five column matrix in the input set. The output set, however, remains a single column matrix that contains the D_{∞} for the specific SPE. Figures 5.3.2, and 5.3.3 show the converted input, and output data sets respectively, generated by the second MATLAB script [Appendix 2].

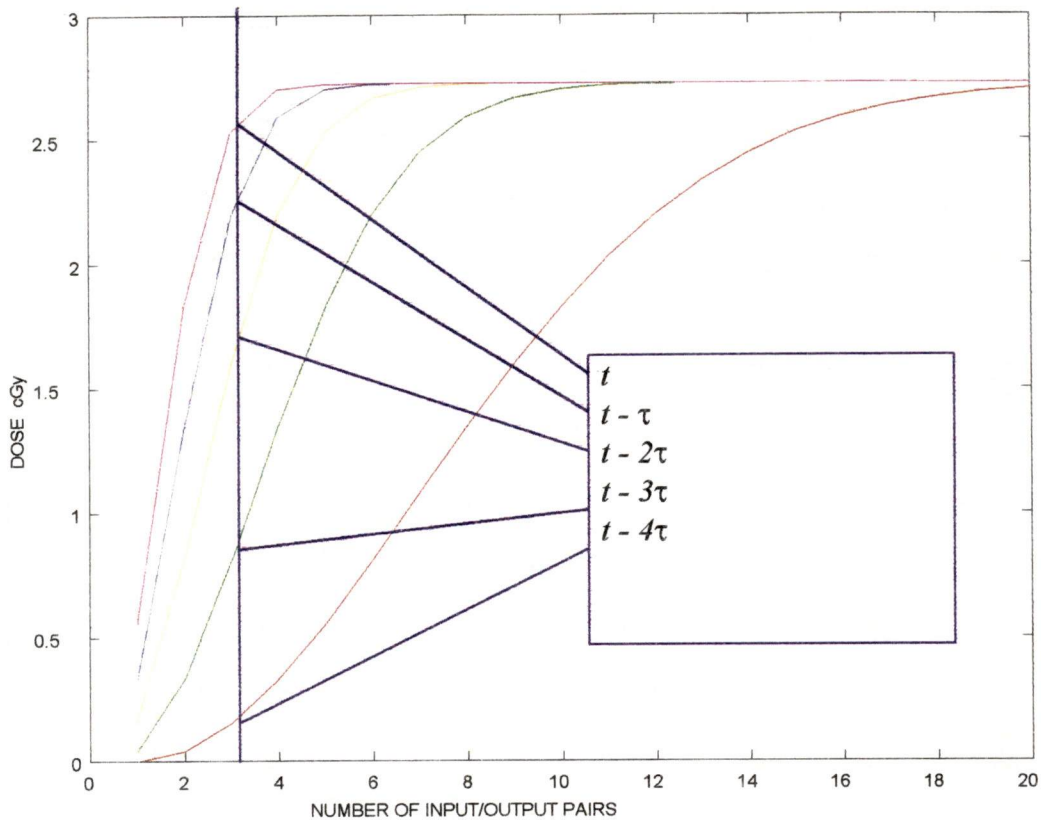


Figure 5.3.2 The input set generated by the script from Appendix 2. Let dose at time t be an arbitrary point on the Weibull curve, this corresponds to one nodal input. The other four input nodes are generated by obtaining dose at times $t - \tau$, $t - 2\tau$, $t - 3\tau$, and $t - 4\tau$.

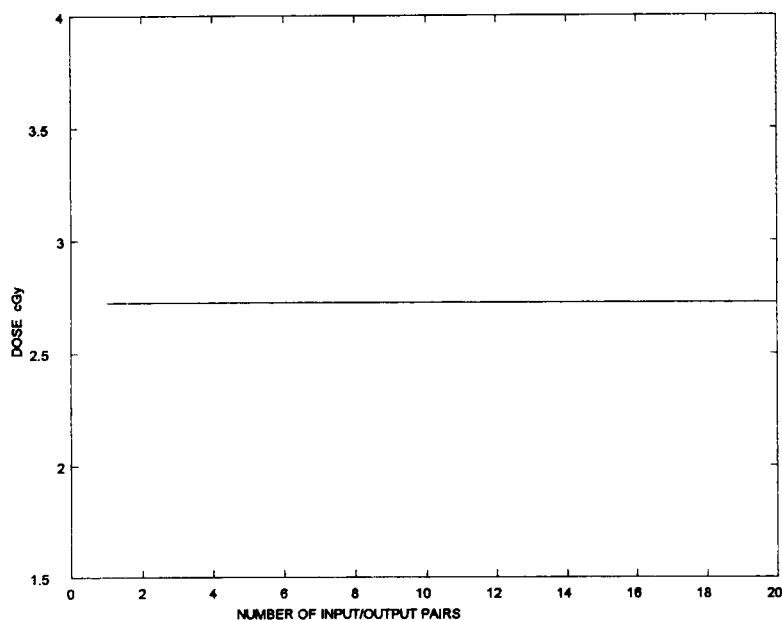


Figure 5.3.3 The output set generated by the script in Appendix 2. This set contains the D_{∞} for a specific SPE

These newly converted input/output data sets each represent a specific SPE. By combining three to five different SPE input/output data sets, the network is given a large range of values of D_{∞} . Figures 5.3.4, and 5.3.5 show the concatenated input, and output data sets respectively. For this work D_{∞} ranged from 0 to 450 cGy, thus facilitating both training and testing. The concatenated data sets are split into two groups, a training group, and a test group. The training group contains training sets that are used to train the STDNN on the functional relationship between the training Weibull function and its

D_{∞} . The test group contains the test sets that test the network's ability to predict the test

D_{∞} .

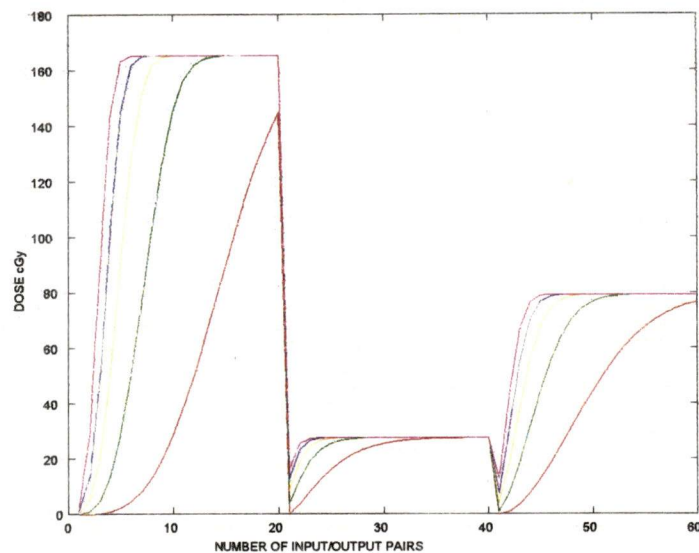


Figure 5.3.4 The concatenated input set

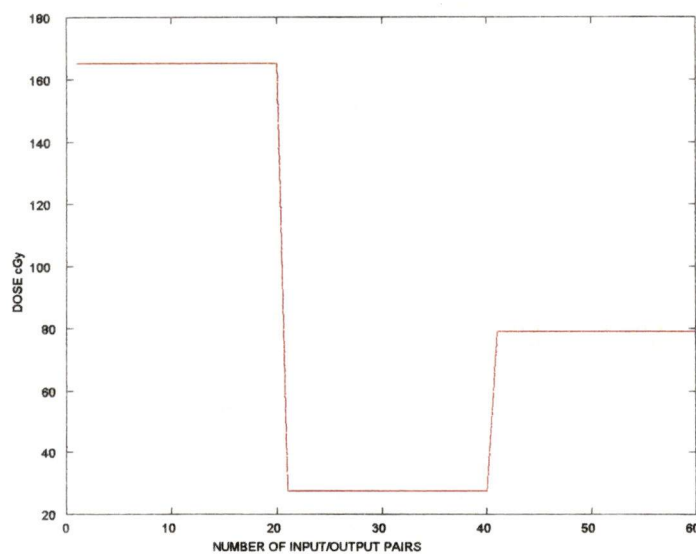


Figure 5.3.5 The concatenated output set

5.4 Training the Network

The network is trained using the Levenberg-Marquadt training algorithm [19] described in Chapter 4. A MATLAB script [Appendix 3] was written to perform training. This script uses a function from the Neural Networks Toolbox [MathWorks Inc.] which trains the neural network using Levenberg-Marquadt.

The Weibull function is a nonlinear function. In order for the network to perform optimally as a function approximator the activation function in the first hidden layer must be nonlinear and continuous. As a result, the tansigmoid (equation 4.2) activation function is chosen. The corresponding activation function in the output layer is the linear activation function since the output can attain any value.

The input data set is scaled using mean centered unit variance. This is done to prevent premature saturation of the hidden neurons, thus slowing training. Since the output data set is linear, there is no need to perform scaling of that set.

The network is trained until the target value of the sum-squared errors is obtained. Using eq(4.5) the target value of the sum-squared error is 160 cGy^2 for training sets with four SPEs, and 200 cGy^2 for training sets that contained five SPEs. Once the network trains to within $\pm 5\%$ of the sum-squared error goal, the values of the corresponding weights and biases are saved.

5.5 Testing the Network

Another MATLAB script [Appendix 4] was written specifically for testing the network. This script uses SIMUFF [Copyright 1984 - 1994 MathWorks INC. Neural Networks Toolbox] which simulates the feed-forward network, using the same network architecture, corresponding activation functions, and the weights and biases obtained from the training script. The test sets are used as the input/output sets. The network's performance is judged visually by comparing the superimposed target output set with the actual output set the network generated. If the difference between the actual output and the target output is greater than 5% the performance is judged to be poor. Poor performance may be a result of overfitting or an incomplete training set. Therefore the following steps are taken:

1. If overfitting is responsible for the network's poor performance, the network is retrained using a smaller network architecture as well as new weights and biases.
2. If the poor performance is the result of an incomplete training set, the network was retrained using a different training set, as well as new weights and biases.

If performance is good (difference of less than 5%) the training and testing process is sometimes repeated to obtain the smallest value of the sum-squared error using the largest permissible data set. Figure 5.5.1 shows a flow chart of the methodology process.

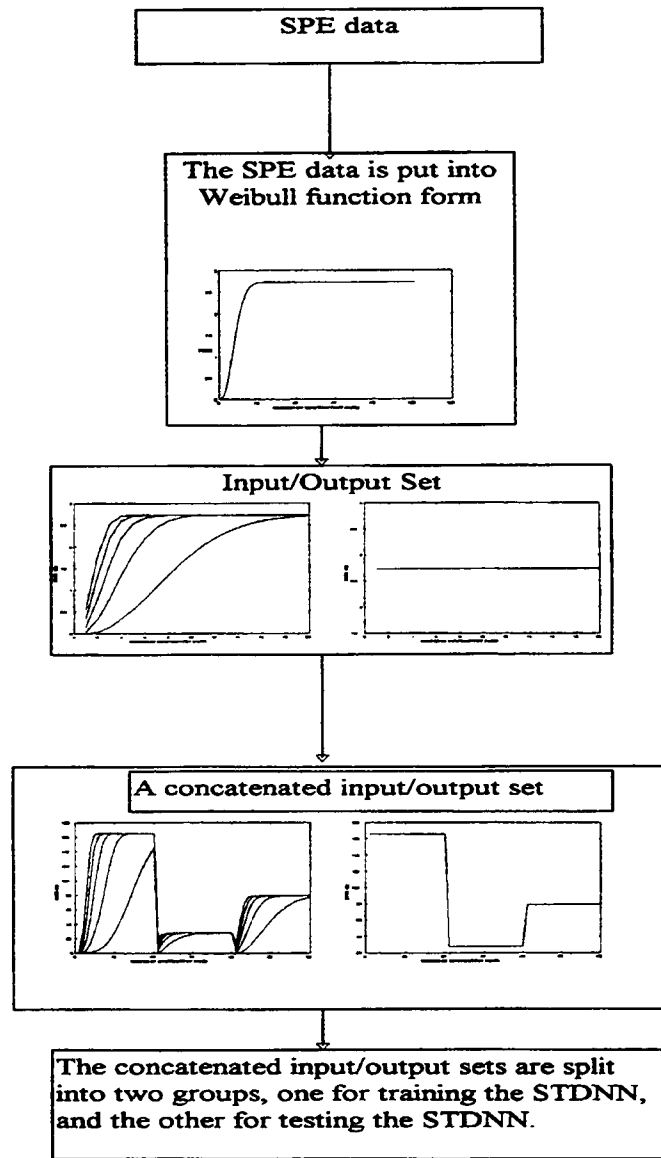


Figure 5.5.1 A flow chart of the Methodology process

6. Results

6.1 Training and Testing the Network

The training process was initially tedious on two fronts. The training algorithm used is the Levenberg-Marquardt [24] algorithm. As with all training algorithms there is the possibility of the algorithm software getting stuck in a local minima. In a local minima the sum-squared error remains virtually constant as the training epochs progress. Should this occur, the training process must be terminated and restarted. The second problem involves the size of the training set. The network is initially trained with data sets that have twenty input/output pairs for each SPE. Thus for a data set that contains seven D_{∞} 's, the data set has 140 input/output pairs. In this case the target sum-squared error was 280 cGy^2 . For such large training sets the D_{∞} has to be within an order of magnitude for the network to train. This is unacceptable since the range of D_{∞} used in the research is between zero and 457.5 cGy. There is also the increased occurrence of the network getting caught in local minima. The problem of the size of the training set is resolved by the reducing the number of input/output pairs from twenty to ten for each SPE. This reduces training difficulties.

For a neural network to perform the task of prediction the training data has to cover the data space where it is expected to operate. Figure 6.1.1 and 6.1.2 respectively show the input and output sets that are used for training the network. This training data set contains both the maximum (457.5 cGy) and minimum (0.5 cGy) values of D_{∞} .

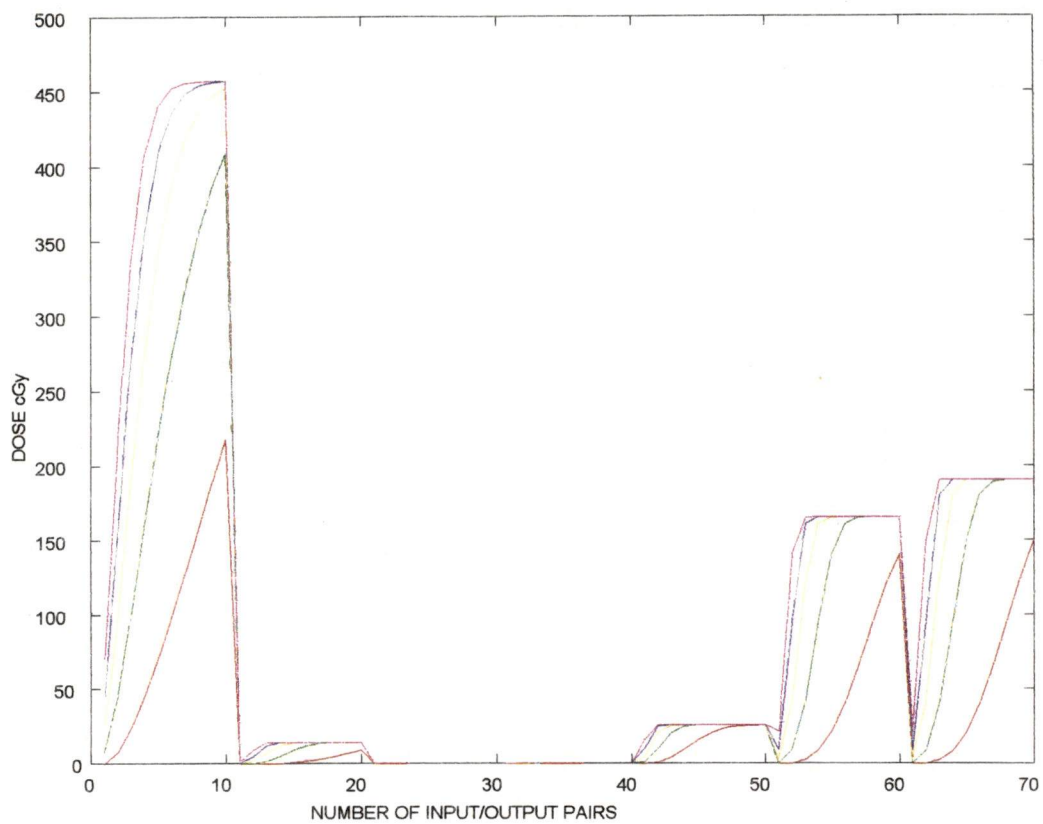


Figure 6.1.1 The training input set

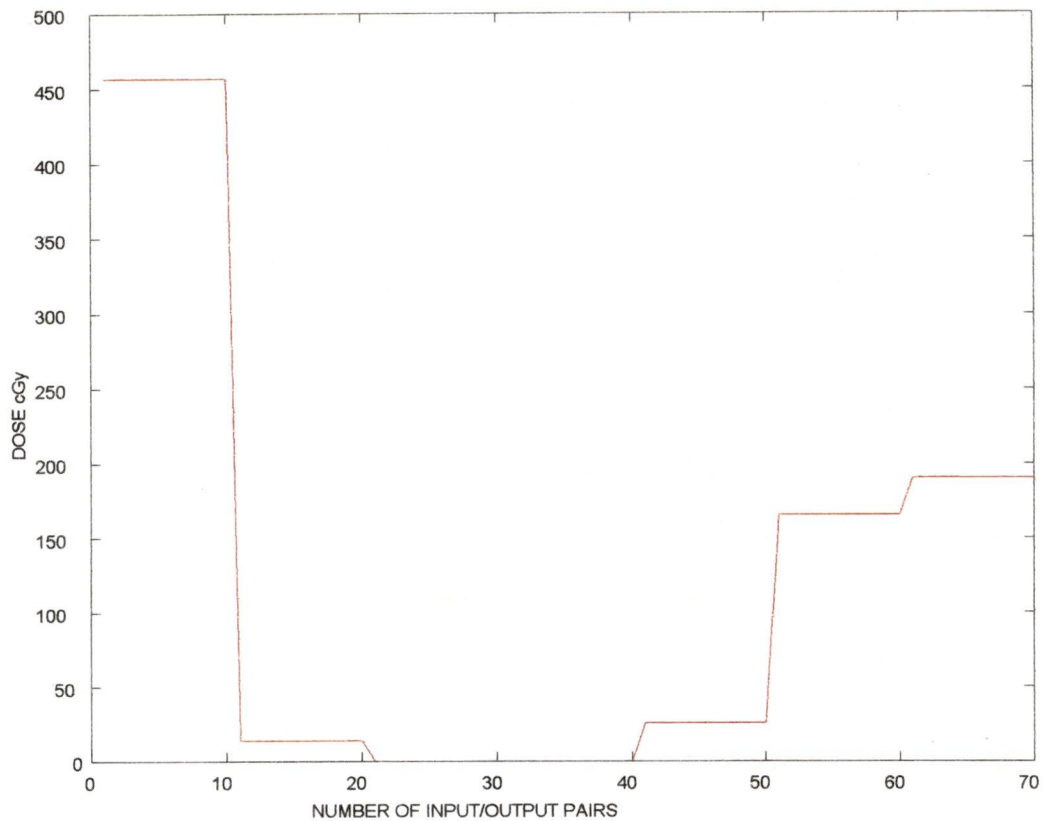


Figure 6.1.2 The training output set

A network architecture with five input neurons, sixteen neurons in the hidden layer, and one neuron in the output layer is used. The network trains to a sum-squared error of 79.7 cGy^2 . This value is lower than the target sum-squared error of 140 cGy^2 . The target error for each neuron is 0.31%, the actual error from each neuron is 0.24%. The first test conducted on the network was to test how well the network generalizes the training data. This is Phase I. Figure 6.1.3 shows the network's approximation of this training data.

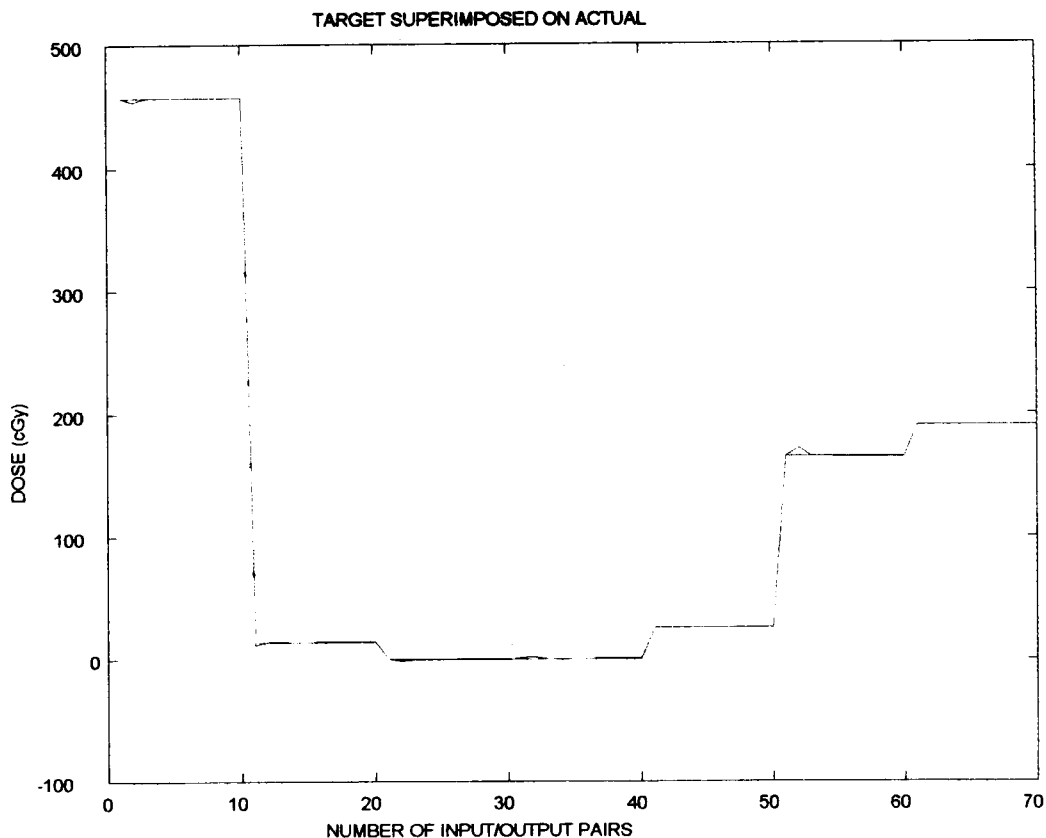


Figure 6.1.3 Testing the training set

Once the network is trained satisfactorily, it is tested with SPEs not used in training. This attempts to verify that the network learned the relationships between the sampled doses and the final dose. To perform this testing, five sets of SPEs are used. The architecture, activation functions, weights, and biases [Appendix 5] saved from the training and initial test phase are used. The new dose profiles originate from the same population as those used for training, but differ from the training set. Figure 6.1.4 is the plot of the first four SPEs tested. In this plot the network's response is good (within 5%)

for the first and fourth profiles but extremely poor for the second and third. The possible reasons for this poor performance are discussed in later.

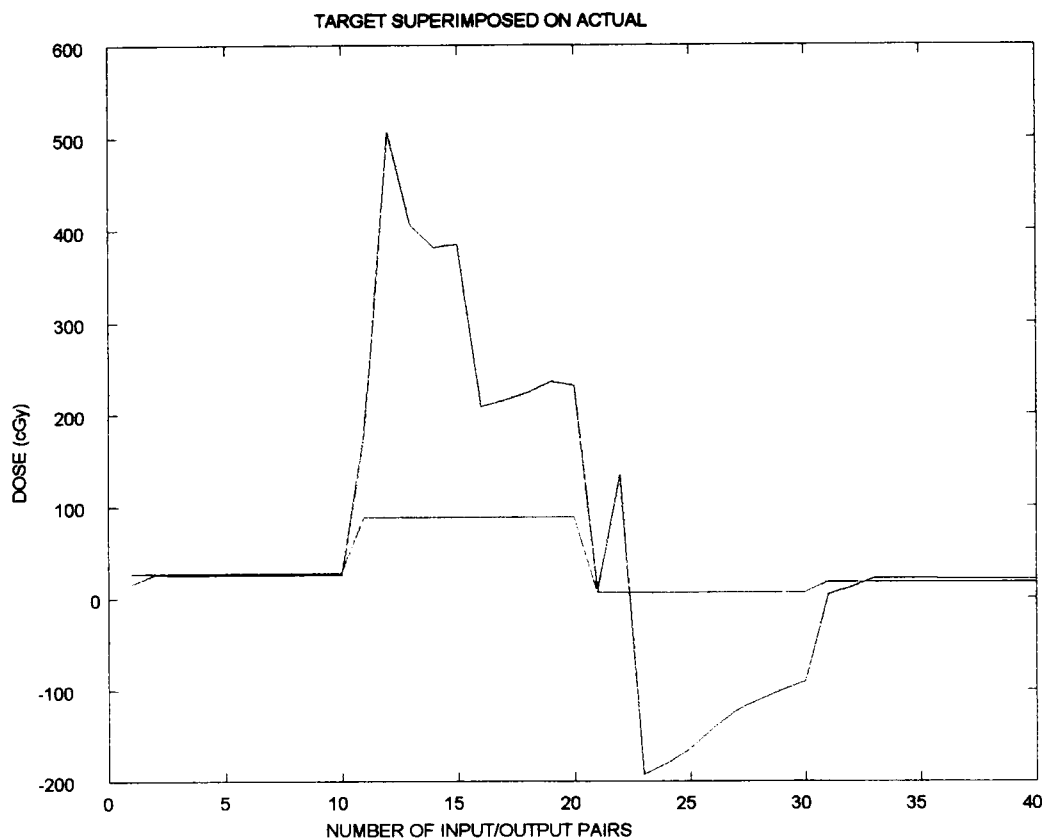


Figure 6.1.4 Test One

The second set of test cases also uses four profiles. This is illustrated in Figure 6.1.5. In the first profile the network gives an overestimate of the maximum but as more information is obtained the approximation, despite being an underestimate, is improved. In the second and fourth profile the network initially has a gross underestimate but as more information is obtained the estimate improves to the desired value. Poor

performance during the initial samples of the event can be tolerated since future predictors using more information can override the previous poor performance. In the third profile the network gives a poor estimate for the entire profile.

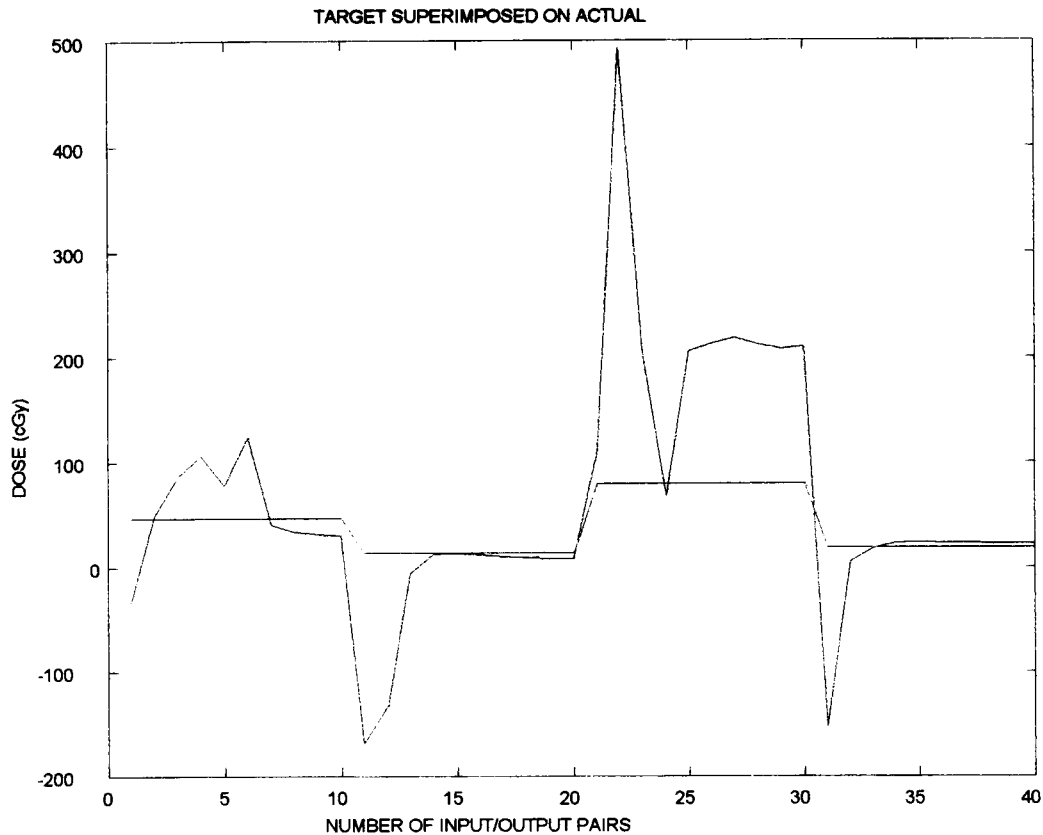


Figure 6.1.5 Test Two

The third set of test cases investigated the network's response for low doses. Three dose profiles are used. Figure 6.1.6 illustrates this third test. In the first profile the network initially overestimates the D_{∞} by 13%, then underestimates it by 50%. Although this underestimation is poor, it is noted that the doses used in this test are small

and, that relative to the data set population, a difference of 0.8 cGy is negligible. In the second and third profiles, the network respectively overestimates and underestimates the D_{∞} . In these cases the percentage differences are approximately 10% and 15%.

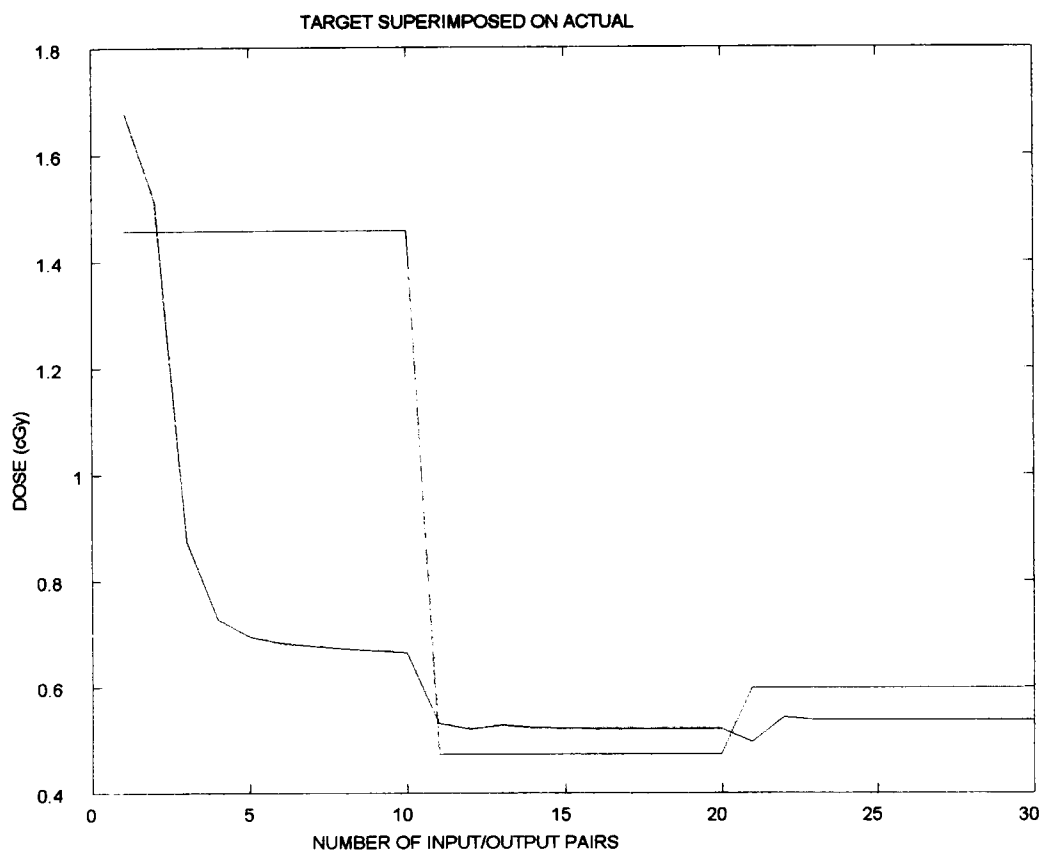


Figure 6.1.6 Test Three

The fourth set of test cases, as illustrated in Figure 6.1.7, involves doses between zero and 10 cGy. In this test seven different profiles are used. Again there is an initial instability but as more information is acquired the network's approximation improves. For the first four profiles the network approximates the D_{∞} well; however, in the fifth

and sixth profiles this is not the case. In the fifth profile there is a gross over-approximation, as more information about the profile is acquired there is improvement. In the sixth profile the network initially underestimates, but the estimate is then instantaneously improved. In the seventh profile the network overestimates the D_{∞} by 10%.

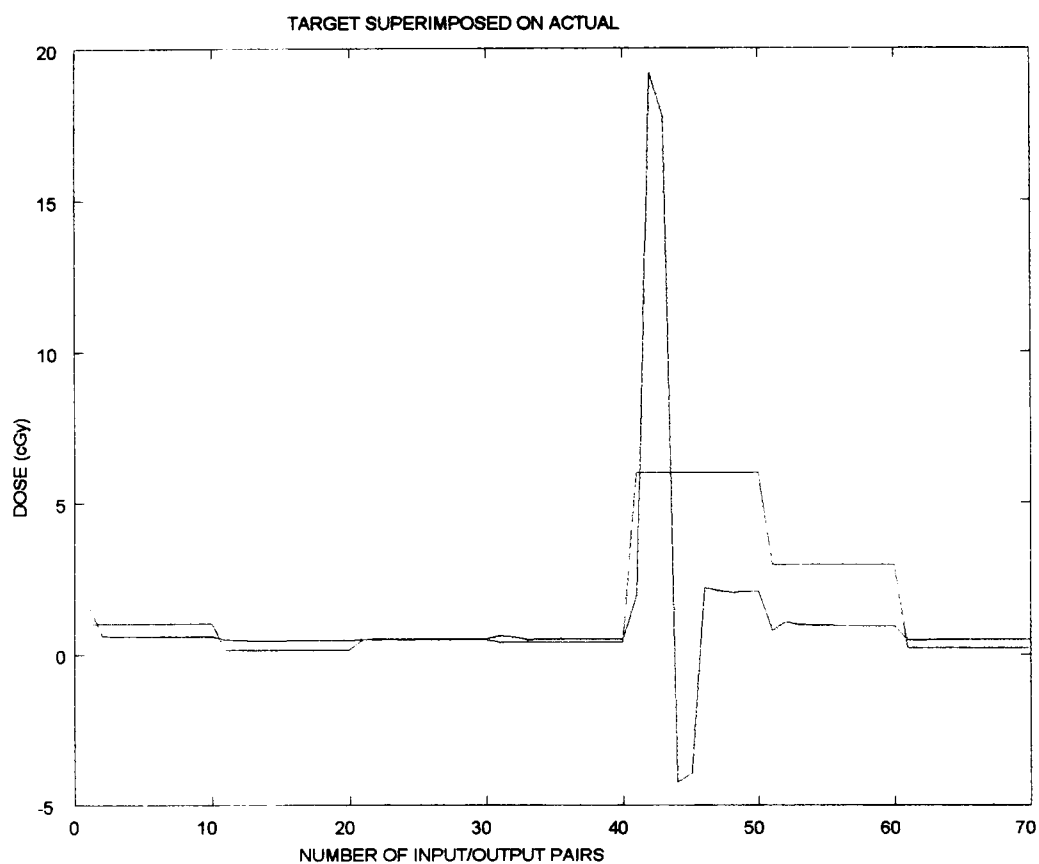


Figure 6.1.7 Test Four

The fifth set of test cases, illustrated in Figure 6.1.8, uses four dose profiles. This test also tested network response at low doses. In the first two profiles the network initially overestimates and then gives more accurate approximations for D_{∞} . In the third profile the network's approximation is improved to within 5%. However, in the fourth profile the network performs poorly at approximating the D_{∞} .

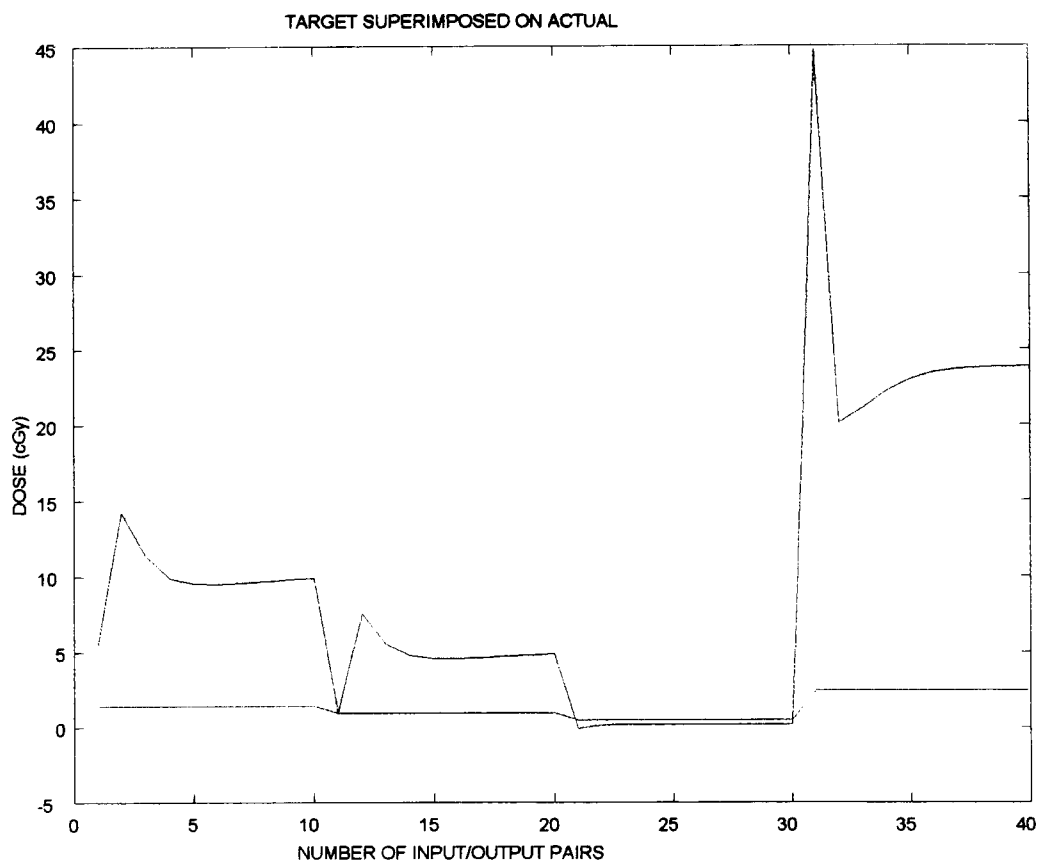


Figure 6.1.8 Test Five

6.2 Discussion

Seven dose profiles were used to train the network and twenty-two other profiles used to test the network. Of the twenty-two test SPEs, five dose profiles were poorly estimated. Clearly the network did not learn the characteristics of these five dose profiles. It is probable that the network's poor performance is due to an inadequate representation of dose profiles in the training. Seven profiles covering zero to 457.5 cGy, with three parameters α , γ , and D_∞ are used for training. All of these parameters affect the shape of the Weibull curve in both axes. Initially, the shape of the Weibull curve was not a consideration in the construction of both the training and test sets. The choice of dose profiles for training was based only on D_∞ . This made the training set too narrow. Hence, in the testing phase, the network would responded poorly to dose profiles that were shaped differently than those used in the training set.

7. Conclusions and Recommendations for Future Study

7.1 Conclusions

In this work an artificial neural network is trained and tested using data sets that were modeled to Weibull functions. The major problems that occurred during training included overfitting and local minima. The network also demonstrates an inability to generalize outside of the training set. There is also poor generalization in cases where the shape of the Weibull is significantly different from that of the training curves.

The optimal network architecture in this research is a five neuron input, sixteen hidden neurons, and one output neuron. Training sets contained seven Weibull functions representing seven different SPEs. The testing is divided into two phases, the first phase tested how well the network generalizes the training set. The second phase tests how well the network generalizes the test sets. It is noted in the second test phase, that the network's initial estimate of the D_{∞} 's is sometimes incorrect but as more information is acquired about the dose profile, the estimate improved. This evidence supports the conclusion that the ANN demonstrates its ability to predict the maximum value of the dose-time profiles of interest.

Artificial neural networks generalize poorly if the range of the test set is outside of that of the training set. More training data needs to be made available, so that the test set always has a range smaller than that of the training set. This presents an interesting dilemma. If this proposed system is implemented, an upper boundary value of D_{∞} is

required. This value should be for a worst case scenario SPE. The issue as to what value should be used is still a scientific unknown.

7.2 Recommendations for Future Study

In this final section recommendations for future study in the use of artificial neural networks for the prediction of major solar particle events are made.

As stated earlier the network performs poorly with Weibull functions that are shaped differently from those of the training set. The parameters that affect the shape of the curves are D_{∞} , α , and γ . The original criteria for the training and test sets was based solely on the values of D_{∞} and not on the other parameters α and γ . In future work, the range of these parameters should be taken into account when determining the training and test sets. This is a way of increasing the knowledge base of the network, thus improving the network's response for all shapes of Weibull curves.

In this research a Sliding Time-Delay Neural Network (STDNN) was used. There are other types of networks that are also capable of processing temporal information. The use of Recurrent Neural Networks, and other networks capable of processing temporal information, for this application should be explored.

In this work a Weibull model of the dose-time profile is used for training and testing the network. This is an intermediate step in the use of ANNs for SPE dose

prediction. The final stage of research should use the actual dose-time data for training and testing of the artificial neural network.

REFERENCES

1. Aso, Takehiko.; Ogawa, Tadahko.; Abe, Minoru.; "Application of Back-Propagation Neural Computing for the Short-Term Prediction of Solar Flares"; J. Geomag. Geoelectr 46, pp 663-668
2. Jakimiec, Maria.; Bartkowiak, Anna.; "Short-term Solar Flare Predictions by Distance-based Regression I. Bearaler Regions in 1988 and 1989 - Continuous Predictors"; Acta Astronomica 44 (1994) pp. 115-140
3. Badhwar, G.D.; "User needs in the area of human presence. Naval Research Laboratory Space Weather Workshop, Arlington, Virginia 1995
4. Parks, George K.; "Physics of Space Plasmas"; Addison Wesley Publishing Company 1991
5. Longair, Malcolm S.; "High Energy Astrophysics"; Cambridge University Press 1994
6. Setti, Giancarlo.; Spada, Gianfranco.; Wolfendale, Arnold W.; "Origin of Cosmic Rays"; D. Reidel Publishing Company 1981
7. Wilson, John W.; Townsend, Lawrence W.; Schimmerling, Walter.; Khandelwal, Govind S.; Khan, Ferdous.; Nealy, John E.; Cucinotta, Francis A.; Simonsen, Lisa C.; Shinn, Judy L.; Norbury, John W.; "Transport Methods and Interactions for Space Radiations" NASA Reference Publication 1257, December 1991
8. Priest, E. R.; "Solar System Magnetic Fields"; D. Reidel Publishing Company 1985
9. Smart, D. F.; Shea, M. A.; "Solar Radiation "; Encyclopedia of Applied Physics, Vol 18, VCH Publishers Inc. 1997

10. ICRU Report 51.; "Quantities and Units in Radiation Protection Dosimetry"; 1993
International Commission on Radiation Units and Measurements, 7910 Woodmont
Avenue, Bethesda, Maryland 20814 USA
11. Hall, Eric J.; "Radiobiology for the Radiobiologist Fourth Edition"; J.B. Lipincott
Company 1994.
12. Khan, Faiz M.; "The Physics of Radiation Therapy, Second Edition"; Williams &
Wilkins 1994
13. Wilson, John W.; Townsend, Lawrence W.; Nealy, John E.; Chun, S. Y.; Hong, B.
S.; Buck, W. W.; Lamkin, S. L.; Ganapol, B. D.; Khan, F.; Cucinotta, F. A.;
"BRYNTRN: A Baryon Transport Model." NASA Technical Paper #2887. 1989
14. Wilson, John W.; Badavi, Francis F.; Cucinotta, Francis. A.; Shinn, Judy L.;
Badhwar, Gautam D.; Silberberg, R.; Tsao, C. H.; Townsend, Lawrence W.; Tripathi,
Ram K.; "HZETRN: Description of a Free-Space Ion and Nucleon Transport and
Shielding Computer Program." NASA Technical Paper #3495. 1995
15. ICRP Publication 60; "1990 Recommendations of the International Commission on
Radiological Protection"; 1991 Pergamon Press Inc., Maxwell House, Fairview Park,
Elmsford, New York 10523, USA
16. Cichocki, A.; Unbehauen, R.; "Neural Networks for Optimization and Signal
Processing"; John Wiley and Sons Ltd & B. G. Teubner, Stuttgart 1994
17. Haykin, Simon.; "NEURAL NETWORKS: A Comprehensive Foundation"; Prentice
Hall, Upper Saddle River, New Jersey Copyright 1994

18. Funahashi, K.; "The Approximate Realization of Continuous Mappings by Neural Networks"; Neural Networks, 2, pp 183-192 1989
19. Gallant, Stephen I.; "Neural Network Learning and Expert Systems"; Massachusetts Institute of Technology 1993
20. Cybenko, G.; "Approximation by Superpositions of a Sigmoid Function"; Mathematics of Control, Signal, and Systems, 2, pp. 303-314 1989
21. Hornick, K., Stinchcombe, M., White, H.; "Multilayer Feedforward Networks are Universal Approximators"; Neural Networks, 2, pp 359-366
22. Hines, J. Wesley.; "MATLAB Supplement to Fuzzy and Neural Approaches in Engineering"; John Wiley & Sons, Inc. 1997
23. Fletcher, R.; "Practical Methods of Optimization"; John Wiley and Sons Ltd 1987
24. Levenberg, K.; "A Method for the Solution of Certain Non-linear Problems in Least Squares"; Quarterly Journal of Applied Mathematics, 2 (1944), pp.164-168
25. Marquardt, D. W.; "An Algorithm for Least Squares Estimation of Non-linear Parameters"; J.SIAM, 11 (1963), pp. 431-441
26. Zapp, E. N.; Ramsey, C. R; Townsend, L. W.; Badhwar, G. D.; "Solar Particle Event Dose Distributions: Parameterization of Dose-Time Profiles." 12th Man in Space Symposium, Washington, D.C., June 8-13, 1997

APPENDICES

Appendix 1

```
% WEIBULL.M

% Written by Garth Forde

% Creates a Weibull Curve

t = [0:88:4400];

% The event lasts 3 days

m = input('enter m parameters: ');
a = input('enter a parameters: ');
g = input('enter g parameters: ');

y1 = m.*(1 - exp((-a).*(t.^g)));

d = y1';

choice = input('Do you want to save these Weibull functions?[y,n]: ','s');

if choice == 'y'

save Weibull y1
end
```

Appendix 2

```
% MKDAT5.M

% Written by Garth Forde to sample points on Weibull Curve

% It will create a five input STDNN

l=length(d);
num=floor(l/5);
x=zeros(num,5);

for i=1:num
    x(i,1)=d(i);
    x(i,2)=d(2*i);
    x(i,3)=d(3*i);
    x(i,4)=d(4*i);
    x(i,5)=d(5*i);
    y=max(max(x))*ones(length(x),1);
end
```

Appendix 3

% LMTRAIN.M

% Written by J. Wesley Hines modified by Garth Forde

% Trains a neural network using LM(trainlm)

```
vecfile = input('enter filename of input/output data vector: ','s');
numhid = input('how many neurons in the hidden layer?');
sel1 = input('sigmoid, tansigmoid, or linear hidden layer?[s,t,l]: ','s');
sel2 = input('sigmoid, tansigmoid, or linear output layer?[s,t,l]: ','s');
sc = input('input and output scaling type MCVU=m,linear=l,none=n ?[m,l,n]: ','s');
```

%%%%%%%% set up default parameters %%%%%%%%%

```
fprintf('\n\nthe default variables are:\n\n');
fprintf('output error tolerance (sse)=200\n');          tolerance = 200;
fprintf('errors plotted in progress every 25 epoch.\n'); epoch_disp = 25; plotflag = 25;
fprintf('max number of training epochs = 1000.\n');    maxepochs = 1000;
```

%%% allow user to change default parameters %%%

```
choice = input('are you satisfied with these selections?[y,n]: ','s');
```

```
if choice == 'n'
epoch_disp = input('how often to display errors(# of epochs): ');
tolerance = input('output error tolerance (sse): ');
maxepochs = input('max number of epochs to train: ');
end
```

%%% load training data %%%%%%%%%

```
eval(['load ',vecfile]);
[ nov1,numin] = size(x);
[ nov2,numout] = size(y);
```

```
if nov1 ~= nov2
    error('the number of input vectors and targets vectors has to be the same');
```

```

end

fprintf('\nthis network has:\n\n');
fprintf('%0.f input neurons\n',numin);
fprintf('%0.f neurons in the hidden layer\n',numhid);
fprintf('%0.f output neurons\n',numout);
fprintf('there are %0.f input/output pairs in this training set\n\n',nov1);

%%% scale inputs %%%

if sc == 'm'
    [xn,xm,xs] = makenice(x);
elseif sc == 'l'
    [xn,xm,xs] = scale(x);
else
    xn = x;
end

%%% initialize weight vectors to small values %%%

[R,Q] = size(xn');
S1 = numhid;
[S2,Q] = size(y');

[W1,B1] = rands(S1,R);
[W2,B2] = rands(S2,S1);
W2 = W2*.5;
W1 = W1*.5;
B2 = B2*.5;
B1 = B1*.5;

%%% set training parameters for levenberg-marquardt %%%

TP = [epoch_disp,maxepochs,tolerance,.0001,.001,10,.1,1e10];

if sel1 == 'l'           % first layer selection
    F1 = 'purelin';
elseif sel1 == 't'
    F1 = 'tansig';
else
    F1 = 'logsig';
end

```

```

if sel2 == 'l'           % second layer selection
    F2 = 'purelin';
elseif sel2 == 't'
    F2 = 'tansig';
else
    F2 = 'logsig';
end

%%% train the network using LM %%%%

[W1,B1,W2,B2,epochs,TRLM] = trainlm(W1,B1,F1,W2,B2,F2,xn',y',TP);

%% LM tolerance criteria not met %%%

if epochs == maxepochs
    fprintf('\n\n*** LM training complete, no answer found!');
    fprintf('\n*** SSE = %e \n\n',min(TRLM))
else

    %%%% LM tolerance criteria met %%%%

    fprintf('\n\n*** LM training complete after %0.f epochs! ***',epochs);
    fprintf('\n*** SSE = %e \n\n',min(TRLM));
end

save weights W1 B1 W2 B2 xm xs F1 F2 sc

```

Appendix 4

```
% TEST.M

% Written by Garth Forde to test the network

data = input('Enter filename of input/output data: ','s');
eval(['load ',data]);
vecfile = input('Enter filename of weight vectors: ','s');
eval(['load ',vecfile]);

X = makenice(x,xm,xs);
Y = simuff(X', W1, B1, F1, W2, B2, F2);

plot([Y' y]);

title('TARGET SUPERIMPOSED ON ACTUAL');
xlabel('NUMBER OF INPUT/OUTPUT PAIRS');
ylabel('DOSE (cGy)');

end
```

Appendix 5

The weights and biases used in this research:

W1 =

1.0e+003 *

-0.0054	-0.0112	-0.0137	-0.0112	-0.0071
0.0447	-0.3736	0.3359	1.2840	-1.6173
-0.0030	0.0012	0.0043	-0.0122	-0.0490
0.0256	0.0053	-0.0008	0.0154	0.0532
0.0197	0.0026	-0.0025	0.0141	0.0499
0.0058	0.0071	0.0160	0.0225	0.0182
-0.0126	0.0350	0.0048	-0.0059	-0.0147
-0.0044	-0.0107	-0.0109	0.0036	0.0332
-0.0213	-0.0042	0.0014	-0.0122	-0.0446
0.0099	0.0358	0.0393	0.0238	-0.0067
0.0000	0.0000	-0.0002	0.0236	-0.0398
0.0111	0.0216	0.0266	0.0153	-0.0122
-0.0104	-0.0130	-0.0065	-0.0235	0.0954
-0.0008	-0.0029	-0.0142	-0.0239	-0.0309
-0.0118	-0.0212	-0.0259	-0.0148	0.0146
-0.0001	0.0014	-0.0236	0.1697	-0.2313

W2 =

Columns 1 through 7

-133.4339 -6.9438 -178.0688 -39.2678 -55.9464 -119.0362 45.0858

Columns 8 through 14

118.2105 32.4916 -144.0439 -146.0700 73.3327 -76.4966 32.7059

Columns 15 through 16

-70.6856 -120.0399

B1 =

-0.3460

-251.8162

-20.9282

14.3202

20.2976

16.3748

-41.3259

21.5820

-18.1125

-5.6104

9.6419

-8.5485

21.3634

-13.9475

6.5316

-47.8359

B2 =

155.8096

Vita

Garth Forde was born in Diego Martin, Trinidad on April 25, 1972. He attended Bryn Mawr School at the primary school level and Queen's Royal College at the secondary school level. He graduated Queen's Royal College in June of 1991. He entered Morehouse College, Atlanta, Georgia during August of 1991 where in May, 1995 he received his Bachelor of Science in Physics. He then entered the Master's Program in Nuclear Engineering at the University of Tennessee, Knoxville in August of 1995.