

To the Graduate Council:

I am submitting herewith a dissertation written by Nazila H. Tehrani entitled "Predicting Astronaut Radiation Doses From Major Solar Particle Events Using Artificial Intelligence." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

Lawrence W. Townsend

Lawrence W. Townsend, Major Professor

We have read this dissertation
And recommend its acceptance:

J Wesley Morris
Arthur E. Dwyer

Marshall Pace

Accepted for the Council

Leon Mink

Associate Vice Chancellor and
Dean of The Graduate School

Predicting Astronaut Radiation Doses
From
Major Solar Particle Events
Using
Artificial Intelligence

A Dissertation
Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Nazila H. Tehrani
August 1998

Copyright © Nazila H. Tehrani, 1998

All rights reserved

DEDICATION

To my mother Afkham, my father Mohsen,
my husband Farshid,
my sons, Pedram and Parham,
and my daughter Pegah

تقدیم به پدرم محسن ، مادرم افخم

همسرم فرشید

پسرانم پدرام و پرهام

دخترم پگاه

Acknowledgments

The major driving intellectual and moral force behind this project has been my advisor Professor Lawrence W. Townsend. His skill and foresight provided the stimulation and courage required to navigate through the unclear waters of this work. He devoted continuous time and patience in the most gracious and helpful manner I have come to know. He has my gratitude forever.

I would like to thank my dissertation committee members, Professor Marshall O. Pace from the Electrical Engineering Department, and Professors J. Wesley Hines and Arthur E. Ruggles from the Nuclear Engineering Department. Their contributions to the present work have been most valuable. Many thanks are due to the Head of the Department of Nuclear Engineering, Professor Harold L. Dodds, for his valuable support throughout the period of my studies in the department.

So many thanks are due to my colleagues Mr. Paulo Brasko Ferreira and Mr. Xiao Xu, and the Nuclear Engineering staff, Ms. Kathy L. Phillips, Mr. Gary L. Graves and Mr. Richard J. Bailey.

I would like to thank the Basic Engineering Program for financial support through the course of this research.

My mom and dad have been the strongest support during these long years away from home. I have received the greatest encouragement and inspiration from them throughout my life and education.

The words of any language cannot express my love and gratitude to my husband Mr. Farshid Ghassemi who has always been with me, giving his love and reassurance. This is for you.

Last, but not least, I would like to thank the most valuable people in my life, my sons, Pedram and Parham, and my baby daughter, Pegah. Thanks for the joy and happiness that you give me everyday. Being your mom always gives me the power to get things done, and to make my dreams come true.

Abstract

Space radiation is an important issue for manned space flight. For long missions outside of the Earth's magnetosphere, there are two major sources of exposure. Large Solar Particle Events (SPEs) consisting of numerous energetic protons and other heavy ions emitted by the Sun, and the Galactic Cosmic Rays (GCRs) that constitute an isotropic radiation field of low flux and high energy.

In deep-space missions both SPEs and GCRs can be hazardous to the space crew. SPEs can provide an acute dose, which is a large dose over a short period of time. The acute doses from a large SPE that could be received by an astronaut with shielding as thick as a spacesuit may be as large as 500 cGy. GCRs will not provide acute doses, but may increase the lifetime risk of cancer from prolonged exposures in a range of 40-50 cSv/yr. Total doses received by astronauts during SPEs or from GCRs can be estimated using the measured intensity and energy dependence of the particle fluxes in combination with reliable, accurate space radiation shielding codes.

In this research, we are using artificial intelligence to model the dose-time profiles during a major solar particle event. Artificial neural networks are reliable approximators for nonlinear functions. Calculated dose-time profiles from past major solar particle events show a cumulative Weibull distribution shape. In this study we design a dynamic network. This network has the ability to update its dose predictions as new input dose data are received while the event is occurring. To accomplish this temporal behavior of the system we use an innovative Sliding Time-Delay Neural Network (STDNN). By using a STDNN one can predict doses received from large SPEs during the event. The parametric fits and actual calculated doses for the skin, eye and bone marrow are used. The parametric data set obtained by fitting the Weibull functional forms to the calculated dose points has been divided into two subsets. The STDNN has been trained using some of these parametric events. The other subset of parametric data and the actual doses are used for testing the trained network. This is done to show that the network can generalize properly. Results of this testing indicate that the STDNN is capable of predicting doses from events that it has not seen before. The percentage error for large events is as low as

0.22% for training and testing the smooth parametric data. The actual error is as low as $\pm 3.47\%$, for more than 95% of the event period. From the prediction results, we can see that, at the very beginning of the transient, some fluctuation around the target value exists, but after approximately 5% of the event is elapsed, the prediction almost overlaps the target dose value.

Nomenclature

A	Atomic Mass
d_i	Desired Value
D	Absorbed Energy per Unit Mass (Dose)
D_∞	Maximum Absorbed Dose
e_i	Error
E	Energy
E_j'	Recoil Energy
f_{ij}	Differential Cross Sections for Elastic and Inelastic Collision Processes
H	Dose Equivalent
I_t	Mean Excitation Energy
m	Mass
m_{jk}	Multiplicity of Ion j Produced from Interaction of Ion k with the Target
N	Number of Target Molecules per Unit Volume
Q	Quality Factor
$R_j(E)$	Range of Particle Type j with Energy E
$S_j(E)$	Stopping Power of the Medium for Particle Type j with Energy E
v	Velocity
w_{ij}	Synaptic Weights
x	Distance
Z	Atomic Number
Z_t	Number of Electrons per Target Atom
α	Fitting Parameter
β	Ratio of Particle Velocity to Light Speed
ϕ	Flux or Fluence as specified
γ	Fitting Parameter
λ	A Parameter
η	Coordinate Component
ν	Range Scaling Parameter

π	pi=3.14159265358979...
σ	Cross Section (Atomic or Nuclear as Specified)
ξ	Coordinate Component
Ψ	Wave Function
τ_{ij}	Synaptic Delay
∂	Partial Derivative
Σ	Summation
Ω	Direction Vector Taken as Being Normal to the Surface of a Sphere
1-D	One Dimensional

Abbreviations

c	10^{-2}
G	10^9
M	10^6
k	10^3
eV	Electron Volt
g	Gram
m	Meter
Gy	Gray
s	Second
sr	Steradian
Sv	Sievert
yr	Year
ANN	Artificial Neural Network
BFO	Blood Forming Organ (Bone Marrow)
BP	Back Propagation
BRYNTRN	Baryon Transport
CAM	Computerized Anatomical Man
CC	Correlation Coefficient
CME	Coronal Mass Ejection
EVA	Extravehicular activities
FIR	Finite Impulse Response
G	Gauss
GCR	Galactic Cosmic Rays
GOES	Geostationary Operational Environmental Satellites
GA	Genetic Algorithm
He	Helium
HZE	High Charge and Energy

HZETRN	A Free Space Ion and Nucleon Transport and Shielding Code
IC	Information Coefficient
ICRP	International Commission on Radiation Protection
LET	Linear Energy Transfer
LEO	Low Earth Orbit
MLP	Multilayer Perceptron
NASA	National Aeronautics and Space Administration
NLN	Neural Logic Network
NMSE	Normalized Mean Squared Error
RBE	Relative Biological Effectiveness
R_E	Earth's Radius
SPE	Solar Particle Event
STDNN	Sliding Time Delay Neural Network
TAR	Threshold Autoregressive
TDNN	Time Delay Neural Network

Table of Contents

Chapter	Page
1. Introduction	1
1.1 Objective of the work	3
1.2 Impact and originality of the work.....	3
1.3 Background.....	4
2. The space radiation environment.....	5
2.1 Van Allen belts.....	5
2.2 Galactic cosmic rays	6
2.3 Solar particle events	7
2.3.1 Solar radiation	7
2.4 Coronal mass ejections	7
2.5 Solar particle event composition	9
3. Dose calculations.....	11
3.1 Introduction	11
3.2 Particle transport	11
3.3 Calculation of dose and dose equivalent	16
3.4 Dose analyses for SPEs	19
4. Artificial neural networks	21
4.1 Introduction	21
4.2 Problem solving with neural networks	21
4.3 Neural network implementation	25
4.4 Different network architectures	26
4.5 Multilayer Perceptron	26
4.6 Time delayed neural network	27

4.7 Sliding time-delay neural network	28
4.8 Training the neural network	35
4.9 Steepest descent method	36
4.10 Gauss-Newton method	36
4.11 Levenberg-Marquardt method	37
5. Time series prediction of sunspot data.....	39
5.1 Introduction	39
5.2 Sunspot prediction	42
6. Methodology	50
6.1 Approach to research	50
6.2 Objective of the computations	50
6.3 Methods	51
6.4 Neural network architecture	51
6.5 Error analysis	60
6.6 Network training and validation	62
7. Results and discussion	70
7.1 STDNN training and validation	70
8. Conclusions and recommendations for future work	87
8.1 Conclusions	87
8.2 Recommendations for future work	87
References	89
Appendices	97
Appendix A	98
Appendix B	122
Appendix C	135

Appendix D	158
Appendix E	161
Appendix F	163
Appendix G	164
Appendix H	165
Appendix I	167
Vita	168

List of Figures

Figure	Page
1. Space Radiation Environment	2
2. A simplified cutaway of the Solar interior	8
3. Solar Particles moving down the interplanetary field lines	14
4. Illustration of geometric equivalence of a 3-D sphere with isotropically incident radiation and the 1-D with normally incident radiation	15
5. Quality factor versus Linear Energy Transfer	17
6. Illustration of six 1989, solar particle event integral fluences from GOES-7 data, protons only	20
7. Illustration of the pyramidal cell	22
8. Typical multilayer perceptron architecture	29
9. Neuron architecture	30
10. A typical time delayed neural network (TDNN)	31
11. Neuron model with each synapse consisting of a time-varying delay followed by a time- varying weight	32
12. Sliding time-delay neural network: (a) A typical sliding time delay neural network ($\tau = T/n$), T is the last dose sample time, n is the number of input neurons. (b) An example of an input set of arbitrary Weibull curves. Doses at time t , $t-\tau$, $t-2\tau$, $t-3\tau$, and $t-4\tau$ are depicted.	33
13. The 6 month ahead prediction of the monthly smoothed 10.7 cm solar flux using a 6-6-1 network trained by BP. A delay is evident in these predictions as the prediction curve lags behind the actual time series and specific features are not predicted at the correct time. The time shift graph shows the average delay over the prediction set	41

14. The region around 1990 solar maximum is plotted, showing the predictions obtained with the BP trained network and the predictions from the GA trained network	43
15. Sunspot data for years 1700 to 1979	44
16. Dose(cGy)-time(hr) profile using the parametric data and Weibull function (a) skin only, for the SPEs of Nov 87 ($D_{\infty}=0.072$), March 89 ($D_{\infty}=1.64$), April 89 ($D_{\infty}=2.41$), and May 89 ($D_{\infty}=0.11$) (b) for skin, eye, and bone marrow from Table 5	54
17. Sampling data to be used for STDNN. a) skin, b) skin, eye, and bone marrow	56
18. Event total dose (cGy) a) skin only, for November 87, March 89, April 89, and May 89 SPEs, b) skin, eye, and bone marrow from Table 5	58
19. Neural network results after training the network with parametric data. Skin dose is plotted versus the number of input/output pairs	64
20. March 90 event Dose (cGy) and Time (days)	65
21. March 91 event Dose (cGy) and Time (days)	66
22. Training the NN using the actual data	72
23. Train using parametric data, test using the parametric data (All events from Table 5)	73
24. Test with actual data with weights from training using parametric weights as initial values. (Skin dose only)	75
25. Test of parametric data for Oct 89, Nov 89, Mar 90, Apr 90, Mar 91, May 91, Aug 91 with weights from training with the same parametric data	77
26. Test of actual data for Oct 89, Nov 89, Mar 90, Apr 90, Mar 91, May 91, Aug 91 with weights from training with the parametric data for the same events	78
27. Test of parametric data for Mar 89, Apr 89, Aug 89, Sep 89, and June 91 with weights from training the parametric data for Oct 89, Nov 89, Mar 90, Apr 90, Mar 91, May 91, and Aug 91	79
28. Illustration of parametric data (solid line), asymptotic dose (+ symbol) and the prediction from STDNN (broken line) for event 6 (August 89) in Table 6	81

29. Illustration of parametric data (solid line), asymptotic dose (+ symbol) and the prediction from STDNN (broken line) for event 10 (June 91) in Table 6	82
30. Illustration of parametric data (solid line), asymptotic dose (+ symbol) and the prediction from STDNN (broken line) for event 14 (September 89) in Table 6	83
31. Illustration of actual data (dashed line), asymptotic dose (+ symbol) and the prediction from STDNN (solid line) for event 6 (March 91) in Table 7	84
32. Illustration of actual data (dashed line), asymptotic dose (+ symbol) and the prediction from STDNN (solid line) for event 7 in Table 7	85
A-1. Two events that are not used in training the network	98
A-2. Sampling two unused events of Figure A-1	99
A-3. Asymptotic doses for the two unused events.	100
A-4. Testing the two unused events with the trained network weights.	101
A-5. October 89 event Dose (cGy) and Time (days).	102
A-6. November 89 event Dose (cGy) and Time (days).	103
A-7. May 91 event Dose (cGy) and Time (days).	104
A-8. August 91 event Dose (cGy) and Time (days).	105
A-9. April 90 event Dose (cGy) and Time (days).	106
A-10. Train using actual data, test using the actual data (For skin only)	107
A-11. Test with actual doses with weights from parametric using weights from actual as initial values for events October 89, March 90, November 89, May 91, and August 91 (Skin only).	108
A-12. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 1 in Table 6	109
A-13. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 2 in Table 6	110
A-14. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 3 in Table 6	111
A-15. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 4 in Table 6	112

A-16. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 5 in Table 6	113
A-17. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 7 in Table 6	114
A-18. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 8 in Table 6	115
A-19. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 9 in Table 6	116
A-20. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 11 in Table 6	117
A-21. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 12 in Table 6	118
A-22. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 13 in Table 6	119
A-23. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 15 in Table 6	120
A-24. Illustration of actual data, asymptotic dose (+ symbol) and the prediction from STDNN for event 5 in Table 7	121

List of Tables

Table	Page
1. Normalized MSE for one-step ahead prediction on the sunspot data	45
2. Performance of fuzzy NLN using various input-output encoding schemes	47
3. Performance comparison of fuzzy NLN with the back-propagation Network and the threshold autoregressive model, in terms of normalized mean square error, on the sunspot data	48
4. Parametric events used in this research a) training, b) testing	53
5. Parametric values	67
6. Results of error calculation for 15 parametric events from Table 5 (single events only). The events are in ascending order of their D_{∞} values	68
7. Results of error calculation for 7 actual events (Table B-1) predicted By STDNN using parametric data for training (Table 4)	69
B.1 . Actual data for skin, eye, and bone marrow from the transport code calculations	122
B.2 . Error calculations for 15 parametric events.	126
B.3 . Weights and biases from training the network with parametric data for events October 89, November 89, March 90, April 90, March 91, May 91 and August 91 for all three organs	134

Chapter 1

Introduction

One very important consideration in planning for manned missions in space is to protect astronauts from exposure to space radiation. Sources of space radiation as a function of energy are depicted in Figure 1. For missions in low-Earth orbit (LEO), such as the Space Shuttle, the major concerns are particles from the Van Allen radiation belts. A secondary concern is very energetic protons and heavy ions from galactic cosmic rays (GCR) that are able to penetrate the Earth's magnetic field. For long-duration deep space missions, such as a Lunar base or mission to Mars, the two main sources of astronaut radiation exposure are GCR and large solar particle events (SPEs). The main concern with GCRs is chronic exposure to the high charge and energy component (HZE particles). Because of their high linear energy transfer (LET) characteristics and high penetrabilities, they can cause more damage than low LET radiations such as X-rays or protons.

Health concerns include increased risk of carcinogenesis and damage to nonregenerative tissues. Radiation exposures from GCRs during deep space missions can reach current career limits for LEO missions (there are no current limits for deep space missions). Solar particle events, however, may present an acute health risk since they can generate very large doses over relatively short time periods. Equivalent doses as large as 10 Sv may be received over a period of hours or days and at least 10 g/cm² of aluminum shielding may be needed to stay under current, short-term radiation limits for low-Earth orbit (Townsend 1992b). Although adequate protection from the large fluxes of energetic protons may be provided by spacecraft shielding, such is not the case for astronauts outside a spacecraft performing extravehicular activities (EVA) or outside a shielded habitat on the lunar surface. During EVA the only protection from SPE protons is the thinly-shielded (~1 g/cm² aluminum) space suit. Hence, methods for warning crews to seek adequate shelter once an SPE begins must be provided early in the occurrence of the event. At present no reliable method of projecting event total doses during the early

SPACE RADIATION ENVIRONMENT

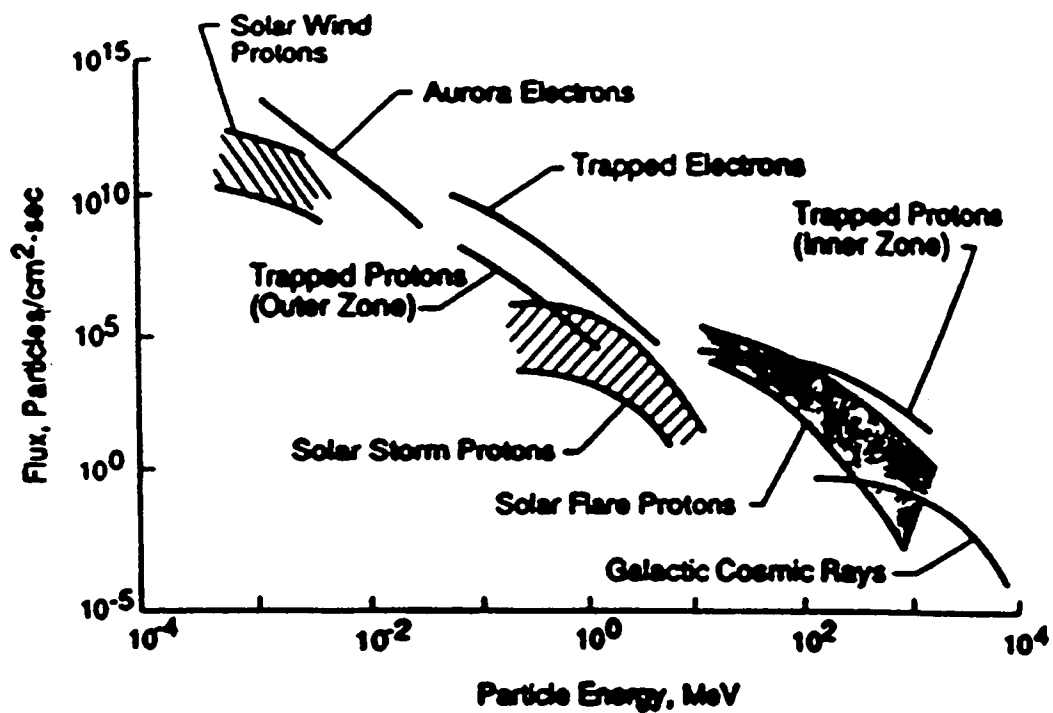


Figure 1. Space Radiation Environment

stages of an event is available, nor are there reliable methods of extrapolating real-time measurements to predict when dose limits might be exceeded. Neural networks and fuzzy neural networks may be useful, however, as tools to reliably perform these dose projections. In this work, one promising artificial neural network design will be developed, trained, tested and validated, and compared with data to determine its usefulness for this task.

1.1 Objective of the work

The purpose of this research is to develop and train an innovative neural network to generate and update dose-time curves for major SPEs from dosimeter readings obtained during the event. The designed network is trained with different organ doses of astronauts calculated from SPE particle fluences measured by space-borne platforms, such as the GOES and IMP satellites, during the past major SPEs.

1.2 Impact and originality of the work

This is an innovative problem in predicting solar particle event dose-time profiles using a neural network architecture. Training a neural network for this task is a relatively new idea that has not been explored in depth before.

Early attempts at using artificial neural networks to predict SPE parameters did not consider health risks; instead, they focused on predictions of solar properties such as sunspot numbers (Aso 1994) or high-energy particle fluxes (Jakimiec 1994), which are highly variable. Recently, Forde (1997) developed a simple ANN based only on smooth parameter fits to data and used it to predict cumulative doses from an SPE with varying degrees of success. Differences between predicted dose using the ANN and those obtained from detailed transport code calculations were as large as 50% for some events.

Here for the first time, an artificial neural network, capable of predicting dose-time profiles from actual dose measurements obtained early in an SPE are developed.

1.3 Background

Manned space missions to the Moon or Mars will be hazardous. The Sun and other stars emit energetic radiations into space. The Earth's magnetosphere can shield a spacecraft in low Earth orbit. But, once a spacecraft leaves the protection of the Earth's magnetosphere, it will be exposed to galactic cosmic radiation and also energetic solar particle events. An acute exposure to spacecraft crews from one SPE during a manned mission could be life-threatening or mission-threatening. Predicting radiation doses from an SPE is not a simple task. Dose-time curves have been plotted after the event has happened (Zapp, 1997). Predicting them before they occur, or even as they occur, has never been successfully accomplished. Prediction of maximum doses and the time build up of doses to astronauts during an SPE will be very important because astronauts working in space or on the Moon must decide if or when to seek shelter. Prediction of maximum doses to interplanetary astronauts during an SPE may be determined by using an artificial neural network (ANN).

Chapter 2

The space radiation environment

As depicted in Figure 1 the radiations in the space come from: 1) The Van Allen Belts, 2) Galactic Cosmic Rays, and 3) Particles emitted by the Sun.

These types of radiation have an energy range from keV to thousands of GeV. Radiation with energies as high as 10^{11} GeV have been detected (Smart and Shea, 1997). The high-energy component of the space radiation can produce secondaries and additional generations of particles. These are a major issue for crew safety. The energy lost in fragmentation reaction is much smaller than the incident energy of a higher energy particle, therefore a very thick shield is necessary to protect the crew.

2.1 Van Allen belts

James Van Allen discovered the Earth radiation belts in 1958 and found that the Earth's magnetic field could organize the particles intensities. He concluded that particles were charged and controlled by the Earth's geomagnetic field. Details about the Van Allen radiation belts were provided for the first time by the Explorer IV spacecraft. There are two different zones or belts, one is between the ionosphere and about $2R_E$ (R_E is Earth's radius) and the second one is between $3R_E$ and $4R_E$.

When cosmic rays penetrate the atmosphere they interact with the atmospheric components and produce secondaries such as neutrons. These neutrons can decay into protons, electrons, and anti-neutrinos, which are the primary sources of the energetic particles of the inner radiation belt.

The outer radiation belt has an outward extension to the Sun side about $10R_E$. On the hemisphere that is not facing the Sun the boundary is about $8-9 R_E$ (Parks, 1991). The energy range for these outer particles is from a few keV to hundreds of keV. The population of these particles increases from contributions by the solar wind and the ionosphere. Higher energy components are provided by accelerating solar wind and lower

energy components are provided by ionosphere. There is another source of high energy particles, that is the remnants of plasma sheet particles injected into the outer radiation belt during auroral disturbances.

2.2 Galactic cosmic rays

The galactic cosmic ray background is an isotropic field with relatively high energy composed of 98 percent protons and heavy ions, and 2 percent electrons, γ -rays and antiparticles. These particles are leftover gases from developing galaxies and the solar system. Of the different exposure sources for astronauts, the GCR has the highest energy. For missions to the Moon or to the Mars, we are concerned about the types of particles, their path lengths, and their energies. The universal hypothesis suggests that there might be 10^6 eVm^{-3} of GCR energy in the universe (Longair, 1994).

The γ -rays are from inverse Compton scattering of starlight and the bremsstrahlung of high energy electrons produce the lower energy spectrum $E < 100 \text{ MeV}$. The γ -rays from decay of neutral pions (from collisions between protons, nuclear components of high energy particles, and interstellar gases) make up the higher energy spectrum $E > 300 \text{ MeV}$. Wilson et al (1991) show that absorbed dose from GCR particles does not give an acute health hazard to astronauts. However, GCRs can provide a chronic dose to astronauts. For high-energy, heavy-ions, there are other biological effects not seen with x-rays or γ -rays. For biological protection, the elemental and isotopic composition of the field is very important.

The hazards of GCRs are not from the intensity of the radiation. The energy dependent fluxes, are in the range of $1 \text{ particle} / (\text{s cm}^2 \text{ sr})$. The major concern is that the average particle energy is high enough to produce fragmentation reactions within shielding material. These reactions become sources for secondaries within the shield, with a smaller thickness of shield to penetrate to reach the spacecraft. The dose received is not high enough to be considered an acute dose, but there is still an increased lifetime risk of carcinogenesis (NCRP, 1989), and increased damage to non-regenerative tissues (Grahn, 1973).

2.3 Solar particle events

2.3.1 Solar radiation

The Sun is a relatively small (radius of 700,000 km), cool, dynamic star with an approximate eleven year activity cycle. Its core looks like a fusion reactor operating at 1.5×10^7 K and at a particle density of 10^{32} m^{-3} (Priest, 1985). The Sun's atmosphere is made up of three regions: the photosphere, the chromosphere, and the corona. Figure 2 is a simplified cutaway of the Solar structure.

The Sun's light is produced in photosphere. This is a thin layer of 500 km thickness. In the chromosphere and the corona the density falls from 10^{23} to 10^{14} m^{-3} , but, temperature increases from 6000 K to 10,000 K.

The magnetic field strength in the photosphere varies between one and one thousand gauss { $1\text{G}=10^{-4}$ Tesla (T) }. However, in the chromosphere and the corona the magnetic field strength varies between one and one hundred gauss.

The particulate and electromagnetic radiation can be divided into two categories. One category is the steady-state, including the Solar wind, which does not change rapidly in time. The second category is the transient radiation, which includes Coronal Mass Ejections (CMEs), Solar energetic Particle Events (SPEs), and any other phenomena that shows changes over a small period of time. The second is the greater concern for protection of deep-space missions crews from the radiation exposure. SPE doses can exceed the annual dose limits for LEO missions and can reach levels that are mission threatening.

2.4 Coronal mass ejections

Coronal mass ejections are self contained structures of plasma that are embedded inside the magnetic field (Longair, 1994). When these are ejected into the space they can

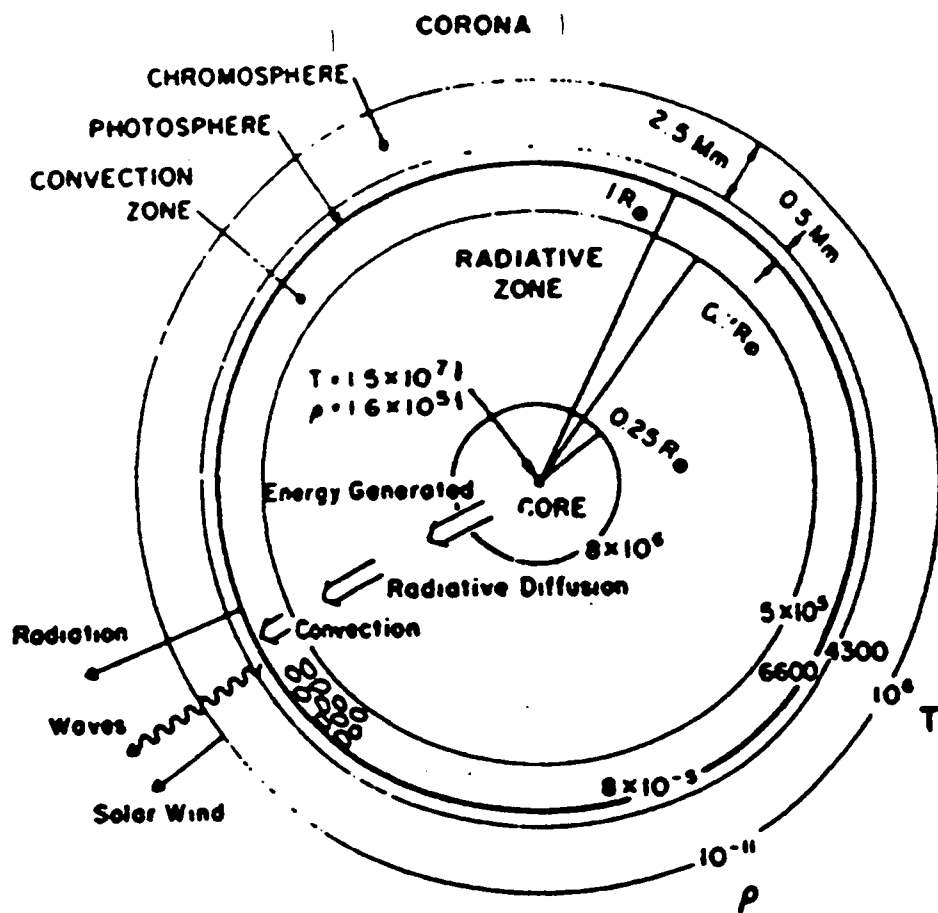


Figure 2. A simplified cutaway of the Solar interior (Priest, 1985)

perturb the planetary environment. These radiations are related to large scale changes in solar features such as disappearing filament and streamer blow out. Their frequency seems to be a function of the solar cycle. In the minimum part of the sunspot cycle the rate is about 0.25 per day and in the maximum part the rate is about 2.5 per day (Smart, Shea, 1997). The velocities of the leading edge of coronal mass ejections can range between 10 and 2100 Km s⁻¹. Therefore, their velocities are comparable to the solar wind. It is believed that these accelerated energetic particles are the actual particles that comprise an energetic SPE. SPEs are short-lived compared to GCRs, and as stated before they can give an acute exposure to interplanetary crews. The biological risks from an acute dose are mainly deterministic (they require a threshold value) rather than stochastic. The deterministic effects include fatigue, and mild radiation sickness at whole body doses of 50 cGy to 120 cGy. If the whole body dose exceeds 325 cGy within a day, it can cause bone marrow depression and even death. The major stochastic effect of concern is carcinogenesis.

One third of coronal mass ejections are fast enough to make interplanetary shock waves. The shock waves can propagate through the interplanetary medium and cause matter to be accelerated to high energies (Smart and Shea, 1997). This acceleration is caused by particles being deflected between the leading and trailing edges of the wave, until their energy is large enough so they can escape from the wave's internal potential. This is similar to cyclotron beam production. The profiles of intensity versus time of solar particles show that some acceleration due to shock waves does occur (Smart and Shea, 1997).

2.5 Solar particle event composition

There are two different categories of SPEs, "small" or "large", depending on their peak fluence, usually in energy range above 10 MeV. There is no constant composition for smaller events. There are two types of small events, 1) ³He rich, and 2) iron rich. Most of the particle fluxes in these events have relatively low energies or have low

enough fluxes at high energies that they are generally not a problem. Thus we are not concerned with them for safety issues.

Larger solar particle event composition is consistent with ion selection based on the first ionization potential of elements in the corona or solar wind. Because of their low ionization potential hydrogen and helium are the most abundant elements in the large solar events. Hence, protons and alphas dominate the composition of larger SPEs.

Because most particles of concern are protons, in this research only doses from proton fluxes are investigated.

Chapter 3

Dose calculations

3.1 Introduction

Historically, the effects of radiation on material is called a "dose" if accumulated over a period of time or a "dose rate" if the effect per unit of time is of interest (Chilton, 1984).

"One must examine more deeply the mechanism of the effect of radiation on matter in order to determine what properties of the radiation are best correlated with its hazard. It is apparent that one should be concerned not so much with the passage of particles or energy through a region of the material being subjected to the hazard, but with the creation of certain physical effects in that material. These effects may be the deposition of energy, ionization of the medium, induction of atomic displacements, production of molecular changes, or other phenomena." (Chilton, 1984).

In space high energy particles such as protons, neutrons, heavier ions and recoiling particles are sources of radiation quantity or dose. The old unit for dose was the rad which is equal to 10^{-2} J/kg. The S.I. unit for dose is the Gray (Gy) which is equal to 1J/kg.

3.2 Particle transport

The particle fluxes required for dose calculations are transported through the spacecraft's aluminum surface, the astronaut's suit, and astronaut's body tissue. The NASA Langley Research Center has developed the BRYNTRN (Wilson et.al., 1989) and HZETRN (Wilson et. al., 1995) transport computer codes to calculate the primary and secondary interactions of solar energetic particles and galactic cosmic rays. BRYNTRN is a coupled neutron-proton transport code and HZETRN is a free space ion and nucleon

transport and shielding code. In this research, we use BRYNTRN because we are only concerned with SPEs. In this code, primary and secondary protons, deuterons, tritons, ^3He , and alpha particles are propagated through the shielding material of interest (aluminum). The program propagates the incident particles and their reaction fragments through a slab of water to simulate the particles' interaction with soft human tissue. The code uses a 1-D Boltzmann transport equation for the high energies in these calculations

$$H_p^*(x, > E) = \sum_j \int_E^\infty \bar{E}'_j Q(E') \sigma_{jp}(E') \phi_p(x, E') dE' \quad (1)$$

for protons and heavy ions. For neutrons there is no stopping power ($S(E)=0$) because they are uncharged. Thus the Boltzmann equation for neutrons is

$$\left[\frac{\partial}{\partial x} + \sigma_n(x, E) \right] \phi_n(x, E) = \sum_j \int_E^\infty f_{nj}(E, E') \phi_j(x, E') dE' \quad (2)$$

where $\sigma_i(E)$ is the macroscopic nuclear absorption cross sections in cm^{-1} , $\partial\phi/\partial x$ represents the net convective particle transfer into or out of the material, $\partial S(E)\phi_p/\partial E$ represents energy losses from coulombic interactions, and $\sigma(E)\phi(x, E)$ represents nuclear attenuation. 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/nucleon.

The right hand side of equation represents the production of protons or neutrons from the interactions with target nuclei. By solving the Boltzmann equation using an iterative scheme, the solution obtained is

$$p(x+h, r_p) = \exp(-\sigma_p h) \Psi_p(x, r_p + h) + \sum_k m_{pk} \sigma_k \frac{v_p}{v_k} \left[\frac{\exp(-\sigma_k h) - \exp(-\sigma_p h)}{\sigma_k - \sigma_p} \right] \times \quad (3)$$

$$\Psi_k \left\{ x, r_k + \frac{v_p}{v_k} h \right\}$$

where

$$\Psi_p(x, r_p) = S_p(E) \phi_p(x, E) \quad (4)$$

and where $r_p = R_p(E)$, $R_p(E)$ is the proton range, h is the step size, and m_{pk} is the multiplicity of protons produced by ion k in a collision. The BRYNTRN algorithms have been validated to within one percent accuracy by direct comparison to an analytical benchmark solution for a continuous space proton spectrum (Wilson et al. 1991).

The equation can be reduced to one dimension due to the isotropy of the radiation created by its path down the interplanetary magnetic field. The isotropy assumption is very important because it allows a spatial de-coupling of the solution which simplifies the calculations. As the particles leave the Sun, they travel through the interplanetary medium. Permeating this medium is a magnetic field which streams away from the Sun in an Archimedean spiral pattern (Smart and Shea, 1997). When charged particles move down these field lines they follow a spiral path (Figure 3). Because of their different energies, individual particles have different radii. There is also overlap for a large number of these field lines. Since these particles are charged, their paths are perturbed after interaction with other particles. Therefore, a random direction for particles, or isotropic field (Smart and Shea, 1997) is assumed. In Figure 4 the equivalence of the calculations for the center of a 3-D spherical shield with isotropically incident radiation, and the one dimensional slab with normally incident radiation is depicted.

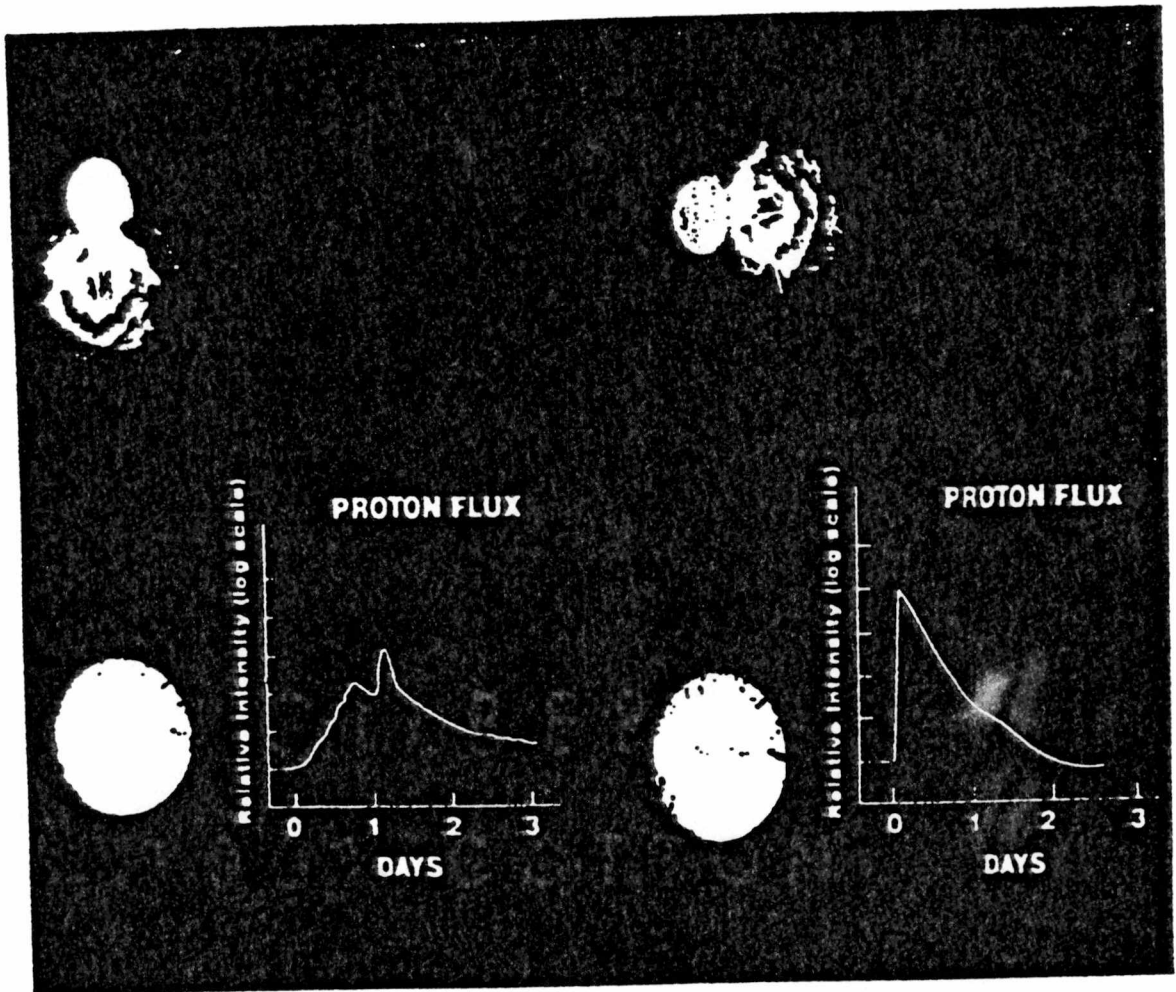


Figure 3. Solar particles moving down the interplanetary field lines. (Smart and Shea, 1997)

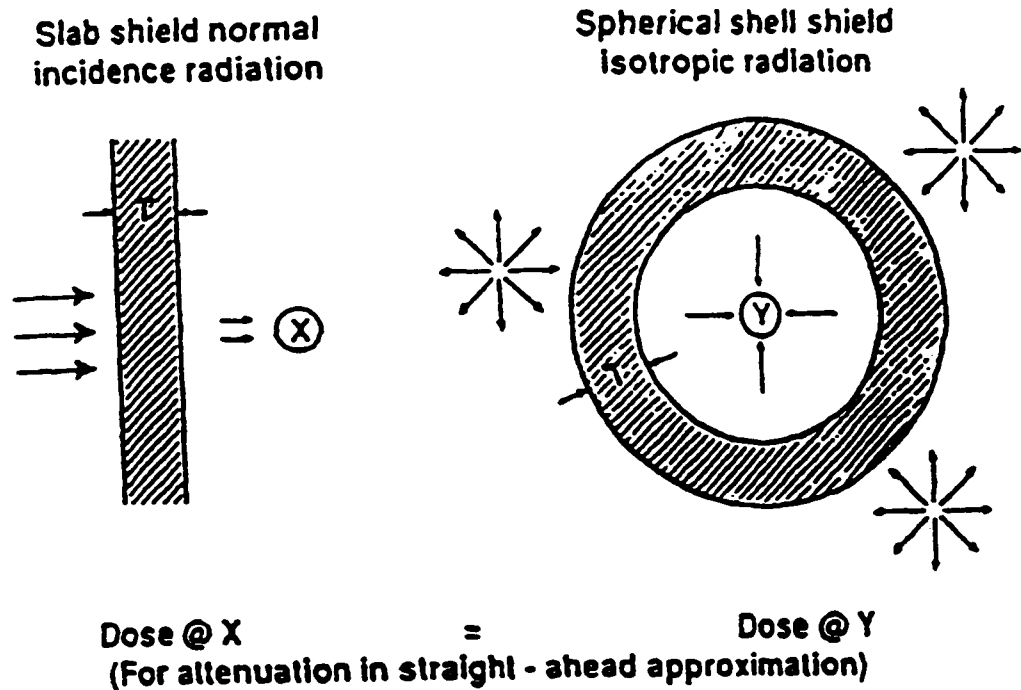


Figure 4. Illustration of geometric equivalence of a 3-D sphere with isotropically incident radiation and the 1-D with normally incident radiation

3.3 Calculation of dose and dose equivalent

Organ doses can not be measured, but must be calculated. For energetic particles in deep space the following equation is used

$$D_i(x, > E) = A_i \int_E^{\infty} S_i(E') \phi_i(x, E') dE' \quad (5)$$

where the stopping power is

$$S(E) = \frac{4\pi N Z_p^2 Z_t e^4}{mv^2} \left(\ln \left[\frac{2mv^2}{(1-\beta)I_t} \right] - \beta^2 - \frac{C}{Z_t} \right) \quad (6)$$

where N denotes 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 (Wilson et al., 1991).

As stated before the biological effects of radiation are dependent on dose and the type of radiation. In radiation protection there is quantity related to the dose and is called the dose equivalent (H). The SI unit for the dose equivalent is joules per kilogram or sievert (Sv) (ICRU Report 51, 1993; Hall, 1994, and Khan 1994). When dose is in rad, the dose equivalent is in rem. The following equation is used to calculate the dose equivalent

$$H_i(x, > E) = A_i \int_E^{\infty} Q(E') S_i(E') \phi_i(x, E') dE' \quad (7)$$

Q is the quality factor which relates absorbed dose to the risk. Figure 5 represents a graph of Quality Factor versus Linear Energy Transfer (LET) (ICRP Publication 60, 1991). For

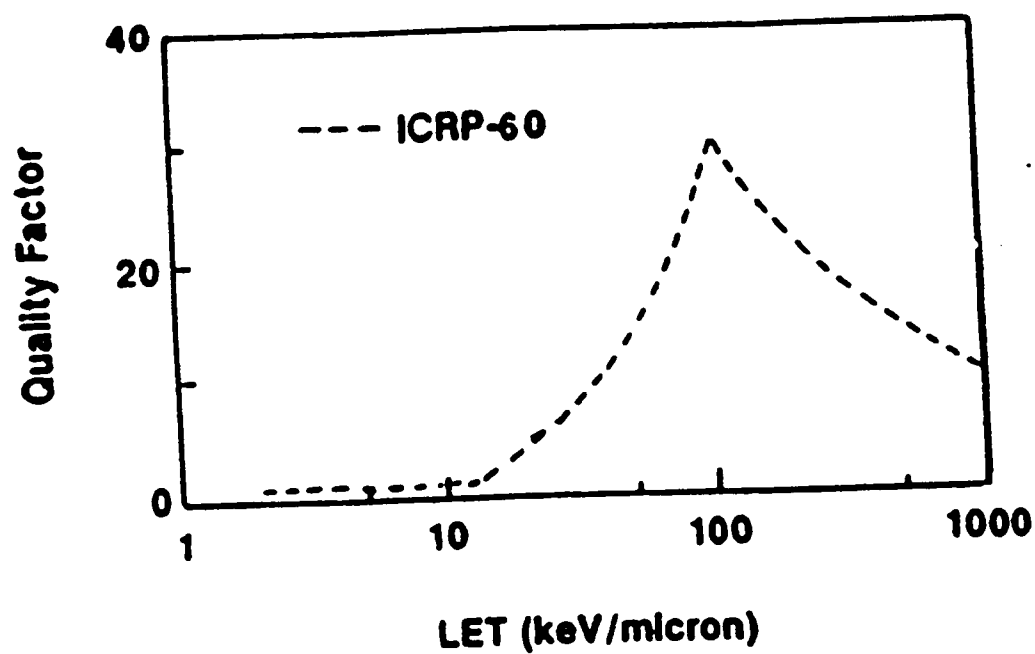


Figure 5. Quality factor versus Linear Energy Transfer (ICRP Publication 60, 1991)

nuclear recoils and target fragments produced by incident protons the equations for dose equivalent are as following

$$D_p^*(x, > E) = \sum_j \int_E^\infty \bar{E}'_j \sigma_{jp}(E') \phi_p(x, E') dE' \quad (8)$$

and

$$H_p^*(x, > E) = \sum_j \int_E^\infty \bar{E}'_j Q(E') \sigma_{jp}(E') \phi_p(x, E') dE' \quad (9)$$

where $\bar{E}'_j$ represents the average recoil energy of the jth fragment.

For calculating human body organ doses and dose equivalents, the incident proton spectra and other resulting products are transported through 2 g/cm^2 of Aluminum shield to represent a spacesuit or thin spacecraft and then through an additional 100 cm of water (tissue-equivalent). In this fashion BRYNTRN produces a depth versus dose distribution for the tissue-equivalent material. These doses and dose equivalents are then folded with the human body organ self-shielding distribution to produce the estimated values of dose and dose-equivalent for a particular organ. Self-shielding distributions are taken from a standing, human computer model called the Computerized Anatomical Man (CAM) (Billings, 1973). The body self-shielding distribution is obtained using a 512-ray distribution over the entire 4π solid angle about a specific body point. In the CAM model used in this work (Atwell, 1990) only one site is used for the ocular lens because it is small and localized organ. For the skin and bone marrow, which are not localized, but rather are distributed all over the body, distributions for 33 different sites for each organ are averaged to get the actual organ distribution used in the dose calculations. The one dimensional form of the Boltzmann transport equation can be solved at a specific thickness into the medium by setting up the proton range and incrementing the solution from $x=0$ in steps of h to the desired thickness.

3.4 Dose analyses for SPEs

SPE fluences used in this research were measured by instruments on Geostationary Operational Environmental Satellites (GOES). The resulting data are dose vs. time since the start of the event. Protons are the only incident particles used, primary alpha particles and other heavy ions are neglected because of a lack of information of their spectra. Their spectra are at best poorly known and too unreliable for precise calculations. Figure 6 illustrates six large solar particle events fluences from GOES-7 measurements (Wilson et al, 1991).

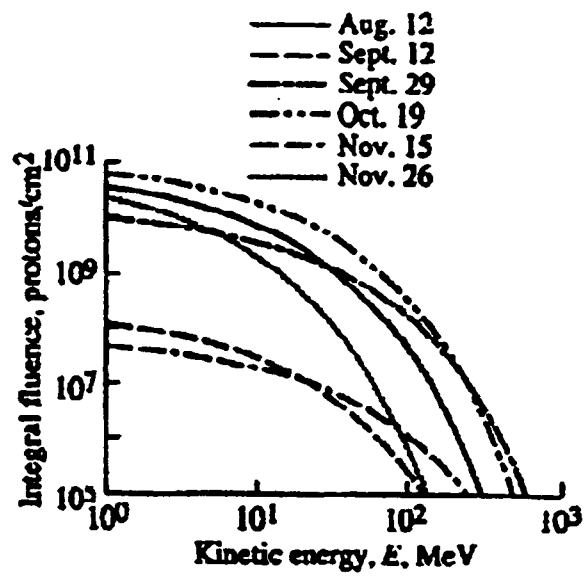


Figure 6. Illustration of six 1989 solar particle event integral fluences from GOES-7 data, protons only (Wilson et al, 1991)

Chapter 4

Artificial neural networks

4.1 Introduction

Development of artificial neural networks has been motivated by their similarities to the human brain, which can problem solve in a different manner compared to conventional computers. The brain is a nonlinear, complex, and parallel information processing system. The brain is capable of perceptual recognition tasks. The brain can build up its own rules from experience over years of lifetime.

The brain is made of nerve cells called neurons with interconnections between them called synapses. Synapses mediate the interactions between neurons. In a human brain there are two different learning mechanisms, the creation of new synaptic connections between neurons, and the modification of existing synapses. There are a variety of shapes and sizes of neurons in the brain. Figure 7 shows the shape of a pyramidal cell. The pyramidal cell can receive 10,000 synaptic contacts and project them onto thousands of cells.

Neural networks are made up of artificial neurons or processing units. A neural network is defined as parallel distributed processor with a natural propensity for storage of knowledge from its experience and making it usable to do certain tasks. The similarities of neural networks and brain are 1) They both learn from experience, and 2) They both use synaptic weights to store knowledge that they learned.

4.2 Problem solving with neural networks

Neural networks are powerful computing tools because of their parallel distributed structure and their learning process and generalization capabilities.

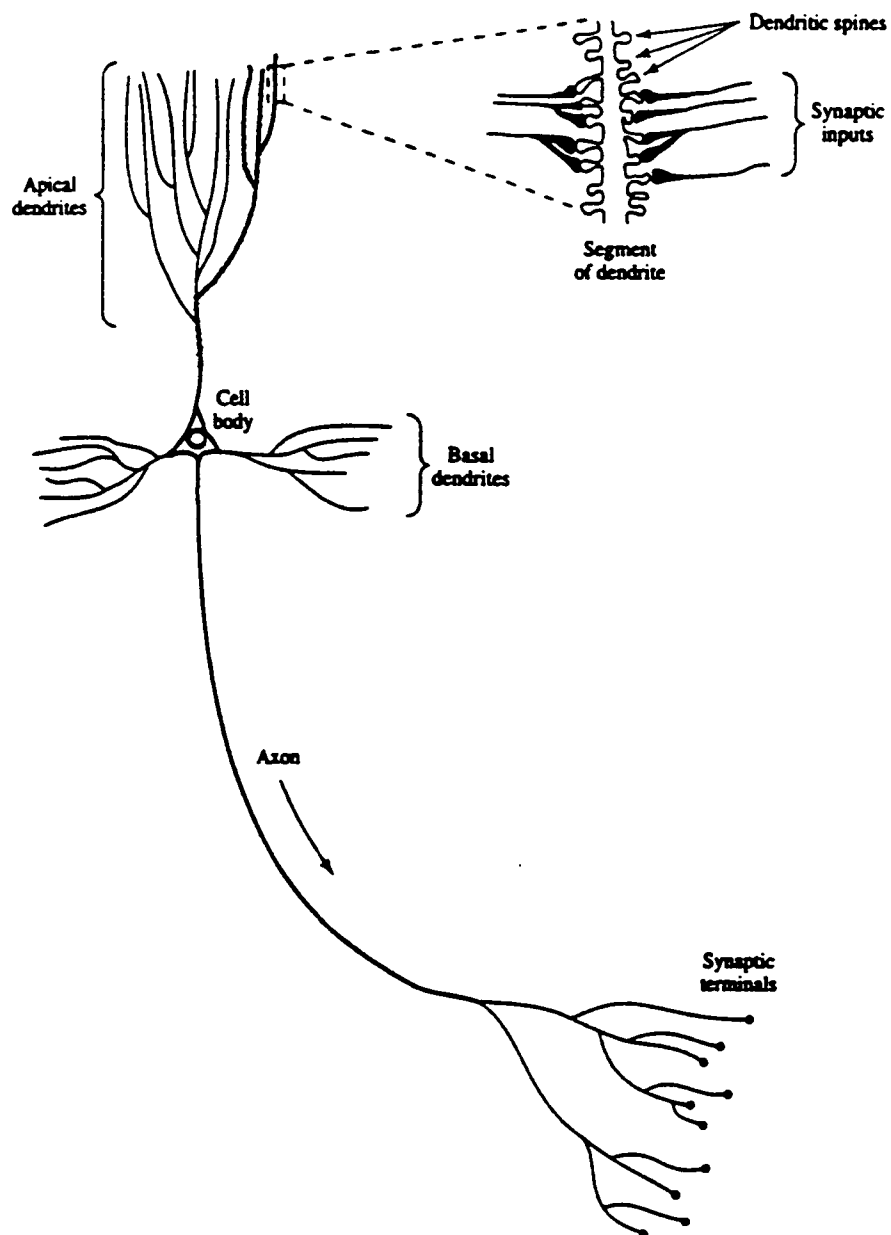


Figure 7. Illustration of the pyramidal cell (Haykin, 1994)

Generalizing refers to the ability to create correct output for inputs not seen during learning. Therefore a properly trained neural network can solve very complex problems.

Artificial neural networks are powerful tools to use for pattern recognition, process and control, time-series prediction, generalization, and system identification. ANN's are very effective in the above applications because of the following characteristics (Haykin 1996): 1) ANN's are distributed nonlinear devices 2) ANN's are fault tolerant 3) ANN's are parallel processors 4) ANN's are able to adapt their free parameters in case of environment change 5) ANN's can approach the nonlinear modeling with nonparametric solutions 6) ANN's are universal approximators 7) ANN's provide evidential response, and 8) ANN's have contextual information.

A neuron can be a nonlinear device. Therefore, an ANN built up from non-linear neurons is also nonlinear. The nonlinearity distributed through the network can model nonlinear input output relationships. Fault tolerant means that if a neuron or its connecting links are damaged, the overall response of the network will not be degraded seriously. Of course the recall of stored information will be impaired in quality, but the damage must be great to lose the overall response. Therefore, neural networks provide graceful degradation in results, instead of catastrophic failure. Adaptivity of an ANN means that there is a built-in capability in neural networks to adapt their weights and biases to changes in the environment. A neural network trained to operate in a particular environment can be used again in that environment after some minor changes are applied to that environment. A neural network used for signal processing, pattern classification, and control applications, which has adaptivity, is an ideal tool in a nonstationary environment, giving robust output.

Neural networks are information processors and can perform universality. The same notation is being used in all the different domains involving neural network applications. ANNs are universal approximators. The Universal Approximation theorem says that the standard Feedforward Multilayer Perceptron (MLP) with one hidden layer is able to estimate any continuous function to any degree of accuracy (Cybenko 1989, Hornik 1989, Funahashi 1989). Neurons always represent a common ingredient to all

neural networks. Thus, in different applications of neural networks, one can share theories and algorithms.

By training neural networks to update their weights and biases in a supervised manner, one can minimize the error between the output and the target values. This is also called input-output mapping. Each example consists of a unique input signal and the corresponding desired output. The training of the network is performed for several cases in the training set until the network reaches steady state, where synaptic weights are not changing anymore. ANNs provide evidential response for pattern classification. They not only give information about which pattern to select, but also they give the reliability of this decision making. This will improve the performance of the network. ANNs can also handle the contextual information naturally, because each neuron in the network is affected by the global activity of every other neuron.

In contrast to classical statistics, neural networks can deal with nonlinearities, non-Gaussian distributions, and non-stationary processes. One can characterize a neural network by specifying its transfer functions embedded in the neurons, the specific architecture, and the training procedure to update the weights and biases. In this research, the task is to model and predict dose versus time profiles by training an appropriate neural network architecture.

Since SPE dose-time profiles usually follow a cumulative Weibull distribution pattern, we characterize them as a nonlinear system. For this purpose the standard Feedforward Multilayer Perceptron and the radial basis function neural network may give the best fit to the cumulative Weibull function. As mentioned earlier, the standard Feedforward Multilayer Perceptron (MLP) with one hidden layer is able to estimate any continuous function to any degree of accuracy (Cybenko 1989, Hornick 1989, Funahashi 1989). To approximate a Weibull function, one has to map domain x (time) to a range y (dose). Neural networks are able to model input-output pairs without an analytic model of the relationships between the pairs.

4.3 Neural network implementation

There are two different classes of neural network architectures that will be explored to simulate the dose-time profiles: 1) Multilayer Perceptron, and 2) Time Delayed Neural Network. For these two architectures and any hybrid architectures that might be developed during this research, the specific architecture, training procedure, and input data must be selected. The results from all of the different neural network systems will be compared to optimize the training and operational performance abilities.

The specific architecture has an impact on the neural network performance. An architecture with a small number of the neurons or hidden layers lacks the free parameters to do the necessary mapping. Networks having a large number of neurons or hidden layers may overfit the data and learn the system noise. Therefore, a robust architecture is required to model the problem and to generalize without overfitting. One can use different techniques to avoid the overfitting problem (Hines et al, 1996b). We can use one or two hidden layers. This will avoid overtraining of the network and overfitting of the data.

There are two types of training algorithms: 1) on-line and 2) off-line. In this research we use off-line training where, during network training, free parameters can be modified and updated. Because we have some information about the doses received at the beginning of the event, we use those to train the network. Later on, as we have more data measured, we can update the network on-line using the new data. One can use the off-line training in constructing a network before putting it into operation or for batch training.

The major task in input selection is to find the most relevant information to model dose-time profiles. Some examples of useful input data are dose, counts per time of different types of particles, time since the event started, and the derivatives of these quantities. We can determine the importance of different inputs using genetic algorithms (Genetic Algorithms in Search, Optimization and Machine Learning", 1989 or NeuralWare's Predict manual, 1995). In this research we chose the actual dose data from the transport code calculations, and time since the actual events started. We also choose the fitting parameters to be used in the cumulative Weibull equation.

4.4 Different network architectures

There are two different classifications of neural networks 1) feedforward networks and 2) recurrent networks. In feedforward networks, signals can only flow in the forward direction. In recurrent networks however, signals can feed back from the output to the input which helps to model the system dynamics. In this section three different neural network architectures that can be used for modeling nonlinear systems are described. They are multilayer perceptron, time delayed, and sliding time-delay neural networks.

4.5 Multilayer Perceptron

The Multilayer Perceptron (MLP) is a feedforward network (Figure 8) which can be trained with a back propagation gradient descent algorithm (Rummelhart and McClelland, 1986), it can also be trained with Levenberg-Marquardt method (Rummelhart and McClelland, 1986). A MLP is built of several different layers of neurons or processing elements. Figure 9 presents a neuron which calculates the summation of incoming signals after these signals have been multiplied by specific weights. Then the neuron operates on the summation result with a monotonically increasing function, which can be linear, sigmoidal, etc.

A MLP can be applied to solve difficult and diverse problems and performs a static nonlinear mapping. An MLP structure is described by 1) number of inputs, 2) number of outputs, 3) number of hidden layers or hidden neurons, 4) activation function in the hidden and output neurons, 5) number of neurons in each layer.

The weights that connect the different layers determine the input-output relationship (Haykin, 1994). One should optimize the values of these weights when training the network to perform the appropriate input-output mapping. To optimize the weights, a supervised algorithm called the error back-propagation process is used. This method uses the error-correction learning rule. There are two passes in this process: 1)

forward pass and 2) backward pass. In the forward pass, an input vector is input to a network and its effects propagated layer by layer through the network. The output is produced as the actual results of the network. The forward pass uses fixed synaptic weights. In backward pass, the synaptic weights are adjusted according to the error-correction rule. The desired value minus the actual output is the error signal. This signal is propagated backward through the network, and the synaptic weights are adjusted in a way to make the actual output get closer to the target.

4.6 Time delayed neural network

To provide a dynamic behavior for the network, memory is needed. To fulfill the memory requirement and provide a dynamic behavior, one can use both current and delayed signals as inputs. This type of network is a Time Delayed Neural Network (Lang and Hinton, 1988) and is able to perform spatial modeling. A typical TDNN is shown in Figure 10 (Haykin, 1994). In part (a) of this figure a single hidden-layer TDNN is illustrated. This time delay ANN consists of 192 (16 by 12) sensory nodes encoding the spectrogram, the hidden layer has 10 copies of 8 hidden neurons, and the output layer has 6 copies of 4 output neurons. The various replicas of a hidden neuron use the same synaptic weights set to narrow windows of the spectrogram. In the same manner, the various replicas of an output neuron use the same synaptic weights set to narrow windows of the pseudospectrogram computed by the hidden layer. In part (b) a time-delay interpretation of the replicated neural network of part (a) is illustrated. This network has a total of 544 synaptic weights. Lang and Hinton (1988) used the TDNN for pattern recognition.

Day and Davenport (1993) and Lin et al. (1992, 1993) developed generalizations of temporal back-propagation that allow the synaptic time delays to be adapted in a continuous way. In Figure 11 the continuous-time filter is shown, which connects neuron i to j . In this figure the synaptic delay ($\tau_{ij}(t)$) and the synaptic weight ($w_{ij}(t)$) are connected in a cascade form.

One should use large numbers of delayed inputs because the order of the system is ambiguous. However, too many inputs result in very sluggish training. Another network called Finite Impulse Response (FIR) Multilayer Perceptron (WAN, 1990), is more general than the TDNN. One can use FIR for chaotic time series modeling (Haykin 1994).

4.7 Sliding time-delay neural network

In this type of time- delay neural network there is a variation in the time delay (see Figure 12.a). Standard time-delay neural networks always have constant delays. In this research the Sliding Time-Delay Neural Network (STDNN) is used. The STDNN has a variable time delay. The input data to the NN are from the time dependent function $f(t)$, and the delay is τ . There is the last dose sample time T and $T=n\tau$, where n =the number of input neurons. Training is performed with a standard BP algorithm since there is no feedback in sliding time-delay neural networks.

In Figure 12.b an example of an input set is depicted. Five arbitrary Weibull curves are plotted. Dose at time t corresponds to one nodal input. The other four nodes are generated by obtaining dose at times $t-\tau$, $t-2\tau$, $t-3\tau$, and $t-4\tau$. Usually n corresponds to the number of neurons in the input layer and τ is the sliding time delay of the interval. As can be seen in Figure 12.b these newly converted input/output data sets each represent a specific SPE. In sliding time delay neural network instead of looking at one cumulative Weibull curve, we break that into say five different Weibull curves. This way, we are using dose information from past and from the present dose measurements. Using these information all together gives a temporal behavior to the neural network.

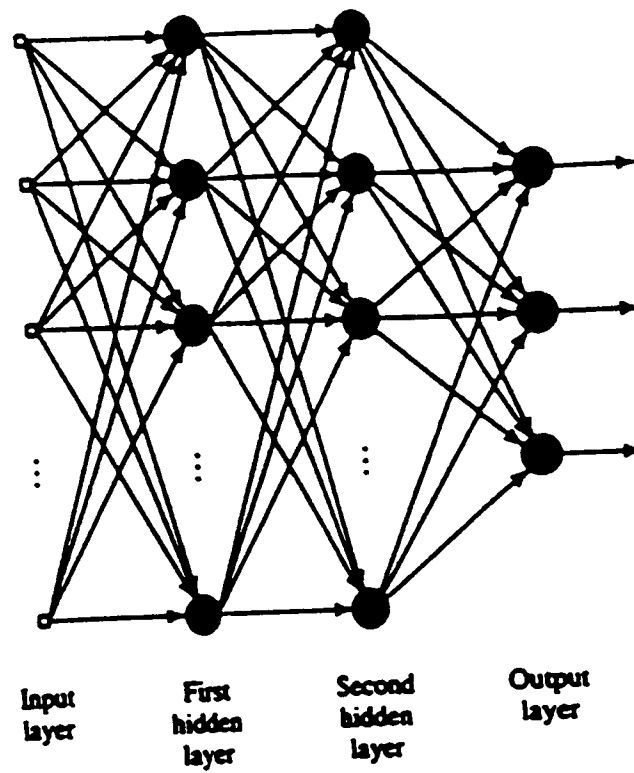


Figure 8. Typical multilayer perceptron architecture (Haykin, 1994)

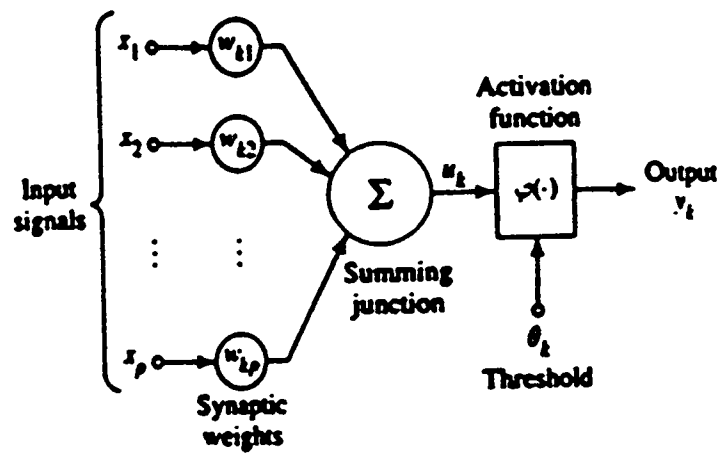


Figure 9. Neuron architecture (Haykin, 1994)

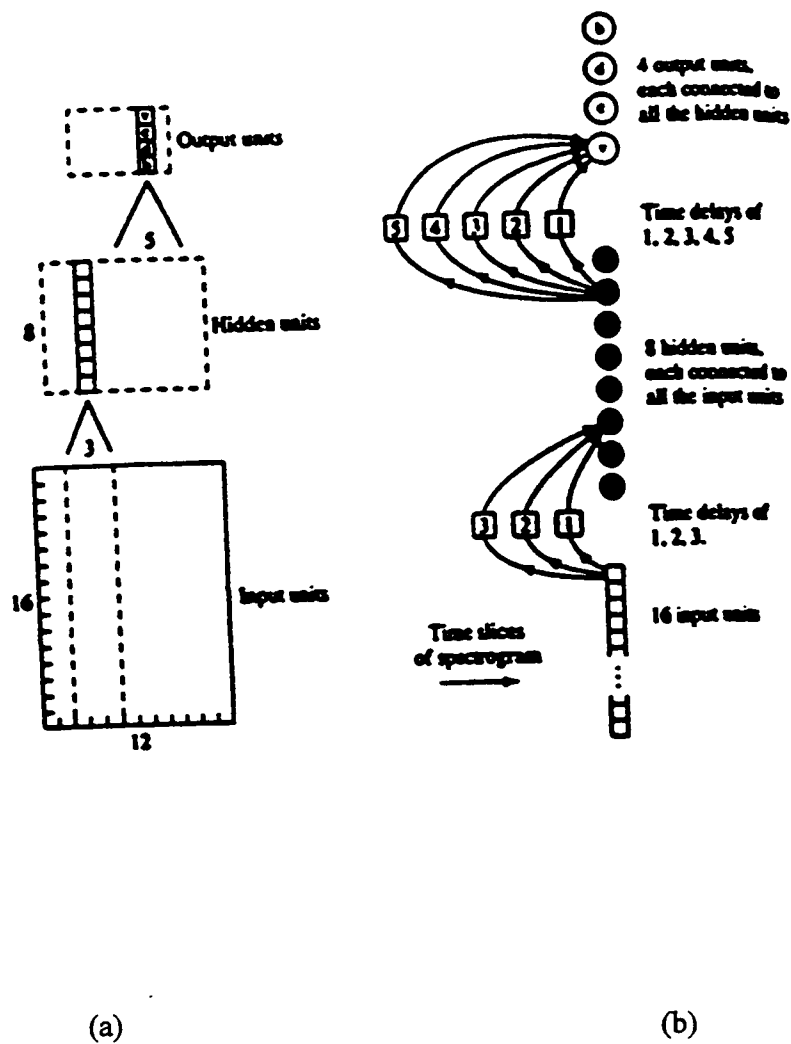


Figure 10. A typical time delayed neural network (TDNN). (Haykin, 1994)

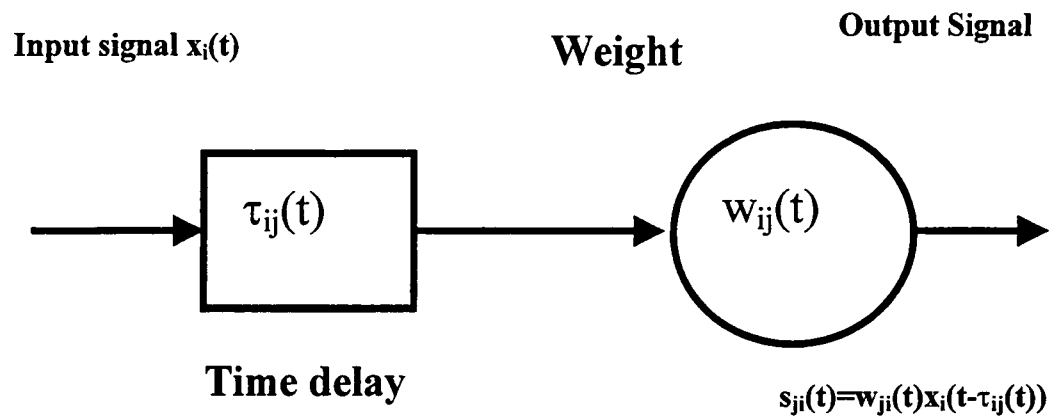
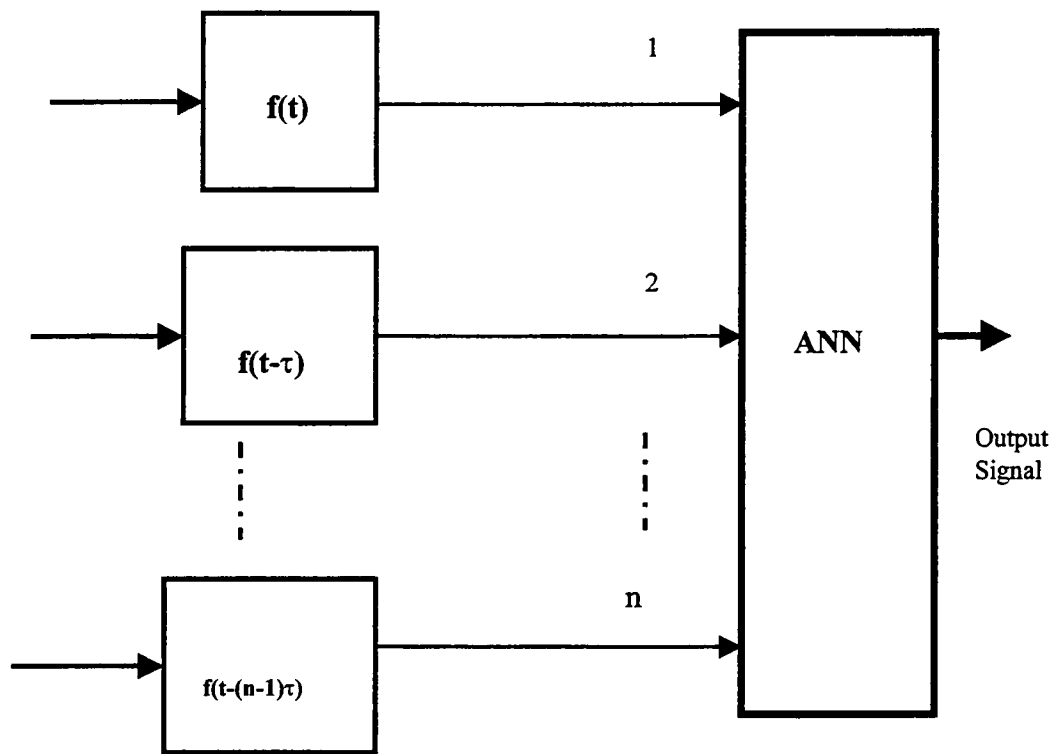


Figure 11. Neuron model with each synapse consisting of a time-varying delay followed by a time-varying weight.

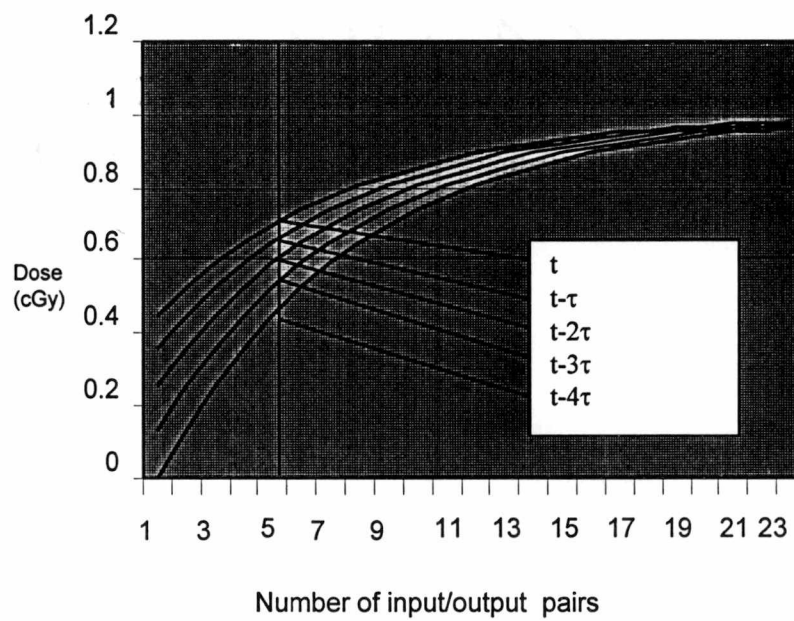
Input Signals



Sliding Time Delay

(a)

Figure 12. Sliding time-delay neural network: (a) A typical sliding time delay neural network ($\tau = T/n$), T is the last dose sample time, n is the number of input neurons. (b) An example of an input set of arbitrary Weibull curves. Dose at time t , $t-\tau$, $t-2\tau$, $t-3\tau$, and $t-4\tau$ are depicted.



(b)

Figure 12. (continued)

4.8 Training the neural network

The goal in training the network is to minimize the sum squared error E . We know that the sum squared error is a function of the neural network's weights and biases. If we assume that x represents the network weights vector and biases, and Δx represents the differences needed to decrease the sum squared error, then we can use the Taylor series expansion for vectors to find the minimum error using gradient methods.

$$f(x+\Delta x) \approx f(x) + g^T \Delta x + .5 \Delta x^T H \Delta x + \dots \quad (10)$$

where g represents the Jacobian gradient vector, and the transpose of the vector g represents the gradient vector of a row vector of partial derivatives of the function f .

$$\nabla f = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \dots \quad \frac{\partial f}{\partial x_n} \right] \quad (11)$$

$$g^T = \nabla f \quad (12)$$

The Hessian matrix is:

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \dots & \dots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} \quad (13)$$

The ANN must have continuous and differentiable activation functions, therefore the first and second order partial derivatives must exist. We can conclude that at local and global minima, g is a null matrix, and the Hessian Matrix is positive definite.

4.9 Steepest descent method

This is a first order method to reduce the error (E) using the Jacobian gradient. We define the error variation (ΔE) as:

$$\Delta E = g^T \Delta x \quad (14)$$

which is a first order approximation. To get a maximum reduction in E in the g direction, we perform a line search. Hence, $\Delta x = \lambda \phi$ is obtained, where ϕ is the unit vector in the $-g$ direction. To find λ , it is increased until the error in the ϕ direction is minimized. These line searches are computationally intensive, because E must be evaluated several times to reach the minimum in the ϕ direction. The advantage of this method is that it has a reliable convergence on the error surface, when far from the minimum. The disadvantage of this method is the slow convergence near a minimum.

4.10 Gauss-Newton method

In this method the error (E) is the sum of squares of individual errors from the output of the training vector:

$$E = \sum_{k=1}^m [f_k(x)]^2 \quad (15)$$

where $f^T = [f_1(x) \ f_2(x) \dots f_n(x)]$, and $E = f^T f$

We use an approximation of the Hessian matrix to find second order information about the error surface. Using the Newton method (Cichocki, 1994) to find Δx we have

$$\Delta x = -H^{-1}g \quad (16)$$

The advantage of the Gauss-Newton method is the rapid convergence when close to the minimum. Also, we use the sum-squared error because it tends to favor error reduction for the large dose events, which are most important, over the small dose events, which are less important as discussed earlier. The disadvantage of this method is that it uses an approximation for the Hessian matrix and the results are not exact.

4.11 Levenberg-Marquardt method

In this study the Levenberg-Marquardt method (Fletcher, 1987 ; Levenberg, 1944; Marquardt, 1963) is used to train the artificial neural network. The Levenberg-Marquardt method combines the reliable convergence of the steepest descent method with the quick convergence of the Gauss-Newton method (Cichocki, 1994). To implement this method, we use a parameter λ in the following fashion:

$$\Delta x = -[\lambda I + H]^{-1}g \quad (17)$$

where I is the identity matrix and

$$\Delta x \cong -[\lambda I + A^T A]^{-1} A^T f \quad (18)$$

When $\lambda \rightarrow \infty$, Δx becomes $-A^T f$ (Steepest Decent Method)

When $\lambda \rightarrow 0$, Δx becomes $[A^T A]^{-1} A^T f$ (Gauss-Newton Method)

The goal is to minimize the error (E). One way to accomplish that is to change the parameter λ (Levenberg, 1944). If we multiply λ by 10, we approach the Steepest Descent Method until the error is reduced and incrementing. Then in this method λ is divided by ten to find out if the new increment Δx has decreased the error. If not, λ has to be multiplied by ten again until the error is reduced. If the Δx is decreasing the error, we are approaching the Gauss-Newton method. If Δx is not reducing the error, we are approaching the steepest descent method. The advantage of the Levenberg-Marquardt method (Fletcher, 1987; Levenberg, 1944; and Marquardt, 1963) over the standard Backpropagation (Cichocki, 1994; Haykin, 1994; and Gallant, 1993) in training the network is its rapid convergence.

Chapter 5

Time series prediction of sunspot data

5.1 Introduction

In the last decade, neural networks have emerged as excellent models for analyzing and predicting the dynamic behavior of complex and nonlinear systems. In time series prediction studies, neural networks can provide accurate predictions of chaotic behavior (Lapedes and Farber, 1987).

Various studies of the long observed time series variations in sunspot numbers, which follow an approximately 11 year cycle between periods of maximum sunspot activity, show that the sunspot data is chaotic in nature (Mundt, et al., 1991). There have been several investigations into the power of neural networks to predict sunspot numbers. (Calvo, et al. 1995, Weigend, et al. 1992, Williams, 1991).

Conway et al (1998) have investigated neural network applications in predicting several sets of time series data, sunspot number, geomagnetic indices and solar 10.7 cm flux. The results of these studies were highly promising as tested by the root mean square error (RMS)¹ for accuracy, and it seems to be superior where compared to those generated by the McNish and Lincoln methods (McNish and Lincoln, 1949). This technique has been used as the basis for prediction in solar activity models by both European Space Agency and NASA.

Generally, assessments of accuracy of prediction using neural networks have been based on RMS or χ^2 error measurements (Calvo et al., 1995, Williams, 1991, Macpherson et al., 1995). These error measurements only give the overall goodness-of-fit. But the solar activity prediction has a dynamic nature and it is very important to

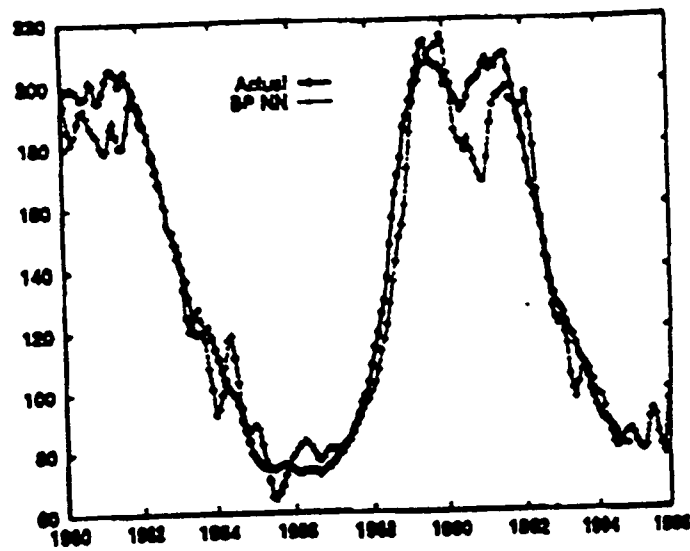
¹ The RMS error E is given by

$$E = \sqrt{\left(\frac{1}{N} \sum_i (P_i - O_i)^2\right)}$$

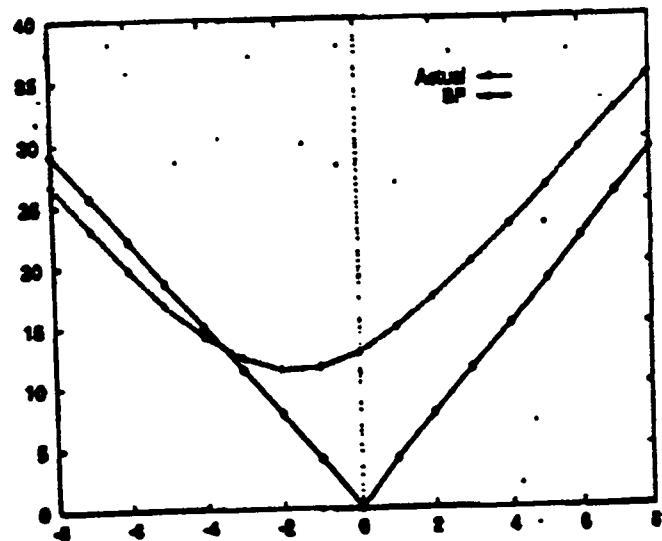
where P_i is the predicted value and O_i is the observed or actual value, and N number of points in the test set

check these features in correct time. It is also very important to note that extremes of high solar activity can have serious consequences for the Earth, such as rapid decay of satellite orbits, geomagnetically induced currents in power systems (with subsequent disruptions on the power distribution grid) and enhanced numbers of incident high energy solar particles that can interfere with communication satellite operations and transmissions (Gorney, 1990).

One important criteria is to judge a network's performance for time series prediction not only with the magnitude of the error, but also with its ability to predict features at the correct time. Conway et al. (1998) showed that the neural networks will not necessarily satisfy both concepts at the same time. If a network provides low RMS prediction accuracy this is at the expense of not being able to predict features in the right time frame. For example, when maxima in a time series occur. Conway et al. called this effect, where a network predicts features late, as the "delay effect". In Figure 13 this delay effect is demonstrated for the smoothed 10.7 cm solar flux, where the 1989 solar maximum prediction is 5 months delayed. To avoid the occurrence of delayed predictions, Conway et al. (1998) adapted the neural network training scheme so that the error measurements considered the RMS error and a penalty for delayed prediction. For producing non-delayed predictions, they had to sacrifice RMS accuracy for on-time prediction. A more flexible approach to do this is offered by the use of a Genetic Algorithm (GA) (Goldberg, 1989). Conway et al. (1998) took the 6-6-1 network (6 input neurons, 6 hidden neurons, and 1 output neuron) used earlier, and trained it using a GA scheme. They succeeded in training a network that predicts solar flux with no delay. In Figure 14 the actual predictions are compared with the actual data using both the BP network's predictions and the GA trained network's predictions are shown. The elimination of the delay is demonstrated clearly in Figure 13b, where the GA trained network's predictions clearly show no delay on the time shift plot. Of course, the GA trained network does have a higher RMS error than the network trained purely by back propagation. This is consistent with the foregoing arguments that the error produced through delayed predictions can be lower than error achieved by on-time prediction.



(a)



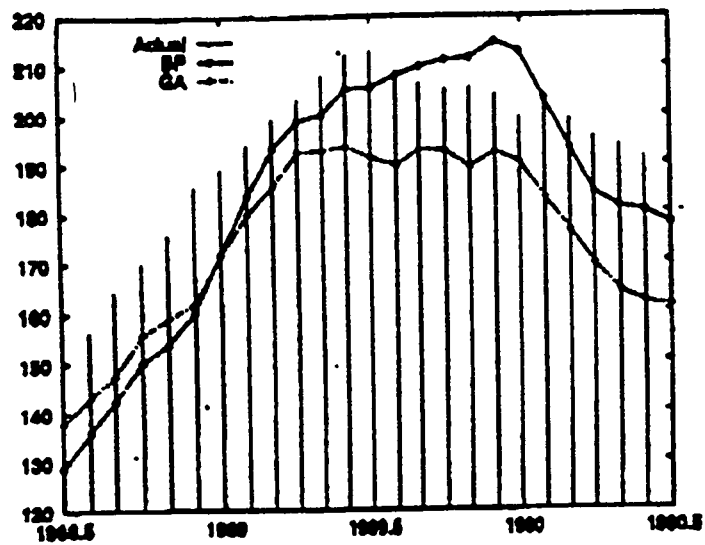
(b)

Figure 13. The 6 month ahead prediction of the monthly smoothed 10.7 cm solar flux using a 6-6-1 network trained by BP. A delay is evident in these predictions as the prediction curve lags behind the actual time series and specific features are not predicted at the correct time. The time shift graph shows the average delay over the prediction set. (Conway et al. 1998)

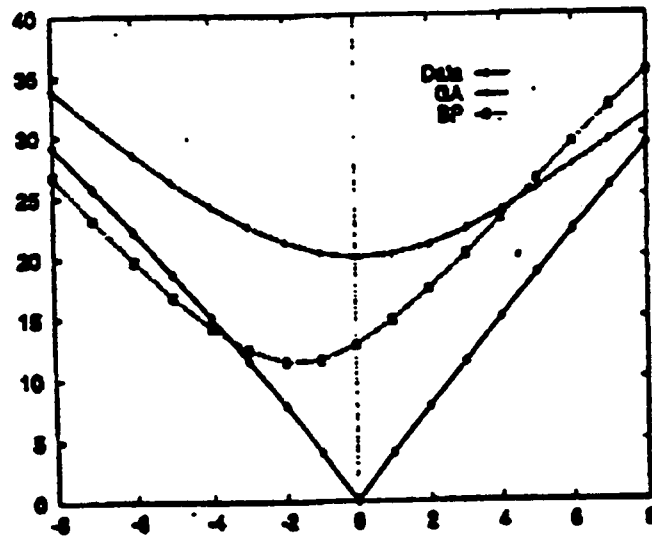
5.2 Sunspot prediction

Sunspots are dark blotches on the Sun and yearly averages have been recorded since 1700. The series is shown in Figure 15 and has served as a benchmark for time series prediction problems. The sunspot series consists of yearly averages of sunspots starting from the year 1700 to the year 1979 (Tan and Teow, 1995). The number of sunspots is used to track solar activity through a solar cycle. For the sunspot problem a prediction of the sunspot number for the following year is made using the information from consecutive sunspot data over the past few years. In the literature, data from the year 1700 to 1920 were applied to train and data from 1921 to 1979 were used for testing the prediction (Tong and Lim, 1980; Weigend et. al., 1990). Also, the prediction set was divided into two groups, the 1921-1955 set was used for validation, and the 1956-1979 set was used for testing. Tan and Teow (1995) used the same training set. They also used the entire prediction set for testing. The neural network architecture used in this study is Neural Logic Network (NLN).

In the study, Weigend et al. (1990) trained a time delay neural network with an embedded dimension of 12 with eight hidden neurons and pruned the weights by adding a complexity term to the cost function. They were able to reduce the network from eight hidden nodes to three. The embedding dimension remained the same. Svarer et al. (1992) pruned the weights using a second-order sensitivity measure. Lin et al. (1997) pruned the orders of networks directly and use the weight decay technique to optimize the nonlinear mapping function. Lin et al. (1997) treat the data in the same way as Weigend (1990) and partitioned it into a training set from 1700 through 1920 and a testing set from 1921 to 1979. The data set was scaled to be in the range of $[0,1.0]$. Various architectures and various number of hidden nodes and different embedding dimensions were used. The networks were trained as a one-step ahead predictor at the learning rate of 0.1 and weight decay of 0.001. After pruning the taps, the networks were retrained. Training was stopped when the normalized mean squared error was less than 0.1. These results compared to Weigend (1990) results are listed in Table 1 (Lin et al. 1997).



(a)



(b)

Figure 14. The region around the 1990 solar maximum is plotted showing the predictions obtained with the BP trained network and the predictions from the GA trained network. (Conway et al, 1998)

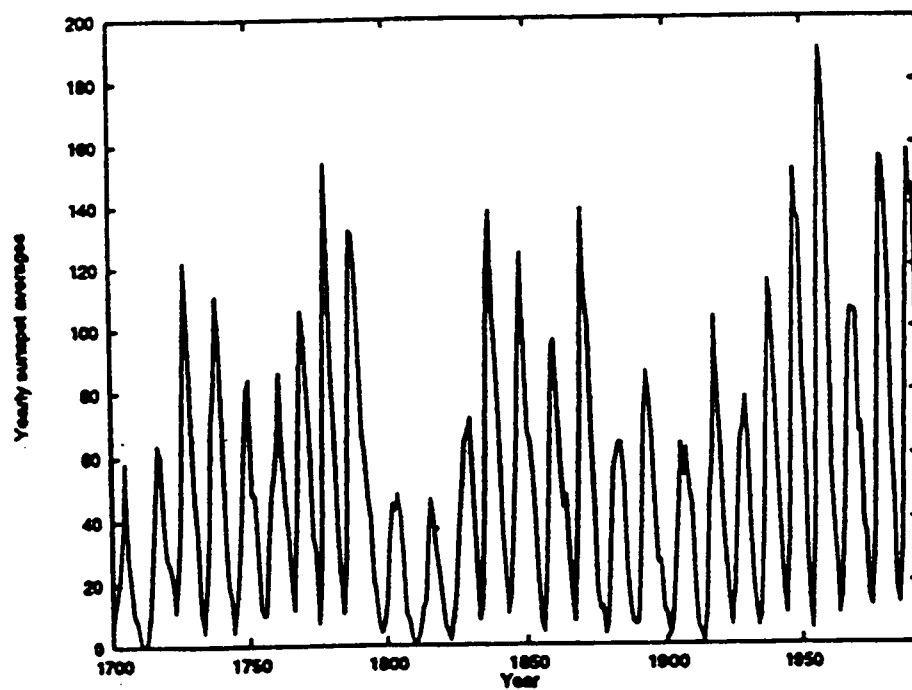


Figure 15. Sunspot data for years 1700 to 1979. (Lin et al., 1997)

Table 1. Normalized MSE for one-step ahead prediction on the sunspot data. (Lin et al, 1997)

Network	NMSE			No. of parameters
	Train (1700-1920)	Test (1921-1955)	Test (1956-1979)	
A (Weigend <i>et al.</i>)	0.082	0.088	0.35	43
B	0.094	0.1065	0.449	69
C	0.0966	0.1013	0.4033	31
D (Svarer <i>et al.</i>)	0.090	0.082	0.35	14

This neural network model is capable of both logical inference and pattern processing (Teh; 1993 and 1994). Although Neural Logic Network (NLN) is presented by nodes and links, an NLN functions like a logical system. A Neural Logic Network performs massively parallel pattern learning and recognition.

The performance of each input-output encoding configuration is measured in terms of normalized mean square error (MSE), correlation coefficient, information coefficient, and trend (up-down) prediction accuracy (Tan and Teow, 1995). Normalized mean squared error (NMSE) indicates the closeness of the prediction to the target. Because of the normalization, this is independent of both the size and the dynamic range of the data set. A correlation coefficient (CC) measures how well the predicted values correlate with the target values. It roughly corresponds to the inverse of the NMSE. The information coefficient (IC) measures the performance with respect to a trivial predictor that uses the current value as the prediction. An information coefficient of 1 means that the performance is only as good as that of the trivial predictor; a value less than 1, means better results. Trend prediction accuracy (Trend) gives the percentage of rise and falls of the predicted values matching with the corresponding target values. It actually, measures how well the prediction follows the trend of the actual series.

Tan and Teow (1995) experimented with six encoding configurations: (A) that uses the actual values in the forecast; (D) that uses the consecutive differences between adjacent values; and (R) that takes the difference between each forecast value and the current value. In Table 2, performances of fuzzy NLN using various input-output encoding schemes are listed. The best entry for each performance measure is highlighted with boldface. Fuzzy NLN gave the best overall performance when the actual forecast values were used to approximate the difference between the current and forecast values (i.e. configuration AD).

Experiments on sunspot data have been conducted using the back-propagation network (Weigend, 1990) and the threshold autoregressive (TAR) model (Tong and Lim, 1980), both of which refer to the normalized mean squared error as average relative variance. Table 3 compares a fuzzy NLN with these two models based on this measure. This comparison is only for the results for the test set (1956-1979), since the performance

Table 2. Performance of fuzzy NLN using various input-output encoding schemes (Tan and Teow, 1995).

Encoding Scheme		No. Nodes	Performance Measures			
Input	Output		NMSE	CC	IC	Trend (%)
A	A	38	0.334	0.860	0.912	89.8
D	A	24	0.349	0.873	0.931	86.4
R	A	26	0.425	0.820	1.028	89.8
A	D	39	0.165	0.914	0.640	96.6
D	D	44	0.199	0.897	0.703	93.2
R	D	28	0.210	0.892	0.721	91.5

Table 3. Performance comparison of fuzzy NLN with the back-propagation network and the threshold autoregressive model, in terms of normalized mean square error, on the sunspot data. (Tan and Teow, 1995)

Model	Training (1700 - 1920)	Validation (1921 - 1955)	Testing (1956 - 1979)
Back-propagation	0.082	0.086	0.35
TAR	0.097	0.097	0.28
Fuzzy NLN (AA)	0.250	0.286	0.42
Fuzzy NLN (AD)	0.179	0.140	0.21

for the validation set (1921-1955) has been optimized during training in both the back-propagation network and the TAR model. As we can see in Table 1 the mean square error when testing (1956-1979) set is 0.35 for backpropagation model, but using the fuzzy NLN (AD) model gives a mean square error of 0.21. Hence, results are much better when using fuzzy NLN (AD) model. The advantage of an NLN architecture is its speed. The network weights typically converge in only a few iterations. Another important advantage of the NLN is the ease of interpreting learned knowledge.

Chapter 6

Methodology

6.1 Approach to research

To evaluate the dose-time profiles using an ANN, a computational investigation will be conducted. In this section, the objectives of the computation, the computational approach, and other investigations are described.

6.2 Objective of the computations

The purpose of this work is to develop a new prototype neural-network-based model to estimate dose-time profiles from dose measurements at the early stages of a solar particle event. Steps in this process are:

- 1) Obtain SPE dose data from calculations of actual, historic events
- 2) Determine the input data from these dose calculations
- 3) Optimize the data preprocessing method
- 4) Train a STDNN to model the dose-time profile
- 5) Test the STDNN model
- 6) Develop a hybrid STDNN based architecture
- 7) Perform additional training & testing with other parameters
- 8) Validate trained neural networks with “real-time simulations” of several actual SPEs using appropriate dose-time inputs.

6.3 Methods

The data used in this study originate from particle fluence data measured for a series of solar particle events from 1989 to 1994. The SPE data were collected using instruments on GOES-6 and GOES-7 satellites. These fluences are parameterized and then input the BRYNTRN transport code to be transported through 2 g/cm^2 of aluminum shield (representing an astronaut's space suit) and through 100 cm of water to simulate the human body's self shielding. These dose data can be expressed by a Weibull equation (Badhwar, 1995)

$$D(t)=D_{\infty}(1-\exp(-(\alpha t)^{\gamma})) \quad (19)$$

where

$D(t)$: dose at time t since start of the event

D_{∞} : maximum absorbed dose (asymptotic dose)

α and γ : fitting parameters

6.4 Neural network architecture

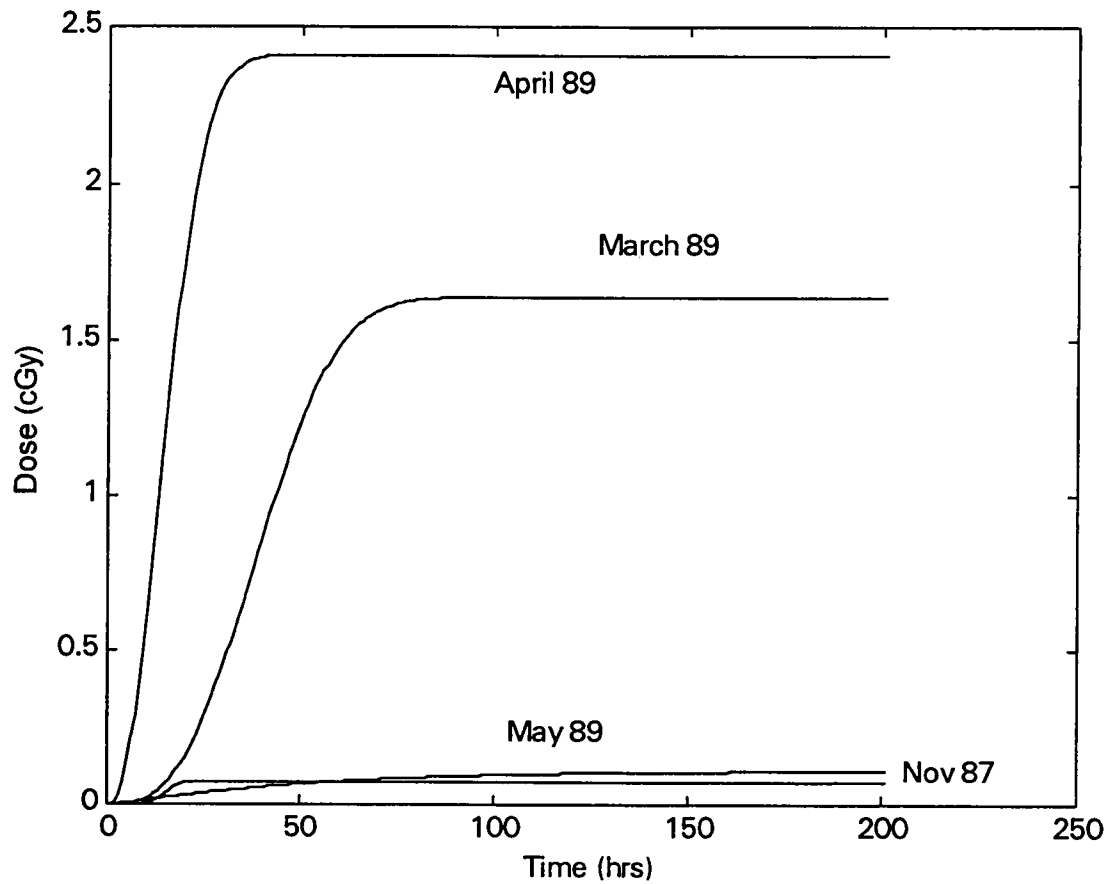
We use the Sliding Time Delay Neural Network (STDNN) described in chapter 4 to predict the dose-time profiles. The sliding time delay neural network architecture used in this study consists of 5 input neurons, 1 hidden layer with 10 hidden neurons and 1 output neuron. Of course it is possible to change the number of input neurons and train different architectures. The neurons in the input layer do not perform any calculations, but feed signals to neurons in the hidden layer, which feed the output layer. 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. We used one hidden layer to reduce the chance of overfitting. We are using a feedforward MLP described in chapter 4, therefore each neuron has one output and multiple inputs (5 inputs in our case). We used 10 hidden neurons because it gives the smallest sum-squared error without overfitting.

The time domain is in days and varies for different events. In the first step we create the training data. Table 4 shows the values for parameters D_{∞} , α and γ for the Weibull equation taken from Zapp (1997). Figure 16.a displays skin doses for several different events using the Weibull function and Figure 16.b shows skin, eye, and bone marrow doses for several events using the Weibull function. In this study we choose a five neuron input architecture for the STDNN. Thus, we need to sample the data in a way that we have five inputs equally spaced in time (Figure 17). In this sampling process we use the dose data from early stages of the event with the dose data from later stages of the event. This gives a dynamic nature to the ANN. Each Weibull curve is broken into five smaller Weibull curves. To get these smaller curves, we use dose data at selected time intervals. This means that instead of plotting the dose versus time at first hour and then second hour, we skip, we use dose say at first hour of the event, then instead of the second hour of event, we use the dose at 4th hour of the event and so on. The D_{∞} or the maximum value of dose is the target value.

Now we use a STDNN to train these data. We try different numbers of hidden neurons, a tansigmoid hidden layer, and a linear output layer. There is no theoretical limit to how many hidden layers are used in this network architecture, however, we found it simple to use one layer. We want to make the ANN as simple as possible. We only use more hidden layers if the desired accuracy has not been reached with one layer. Also, if we use too many hidden layers there is a good chance for overfitting. We used one hidden layer to reduce the chance of overfitting. Using one hidden layer gave us a good dose prediction for all of the important large events and most of the less-important smaller events. We will linearly scale the inputs because in different events data we have very large events, medium events, small events, and even very small events. When training the neural network we want the resulting weights and biases to come from all of the events. If we do not linearly scale the data then the neural network will ignore the small events and is trained based on the large events only. To get over this loss of data, we linearly scale input data so ANN treats all inputs equally. Figure 18 represents event total dose (D_{∞}). The MATLAB script for training the data called LMTRAINP.m is enclosed in Appendix D.

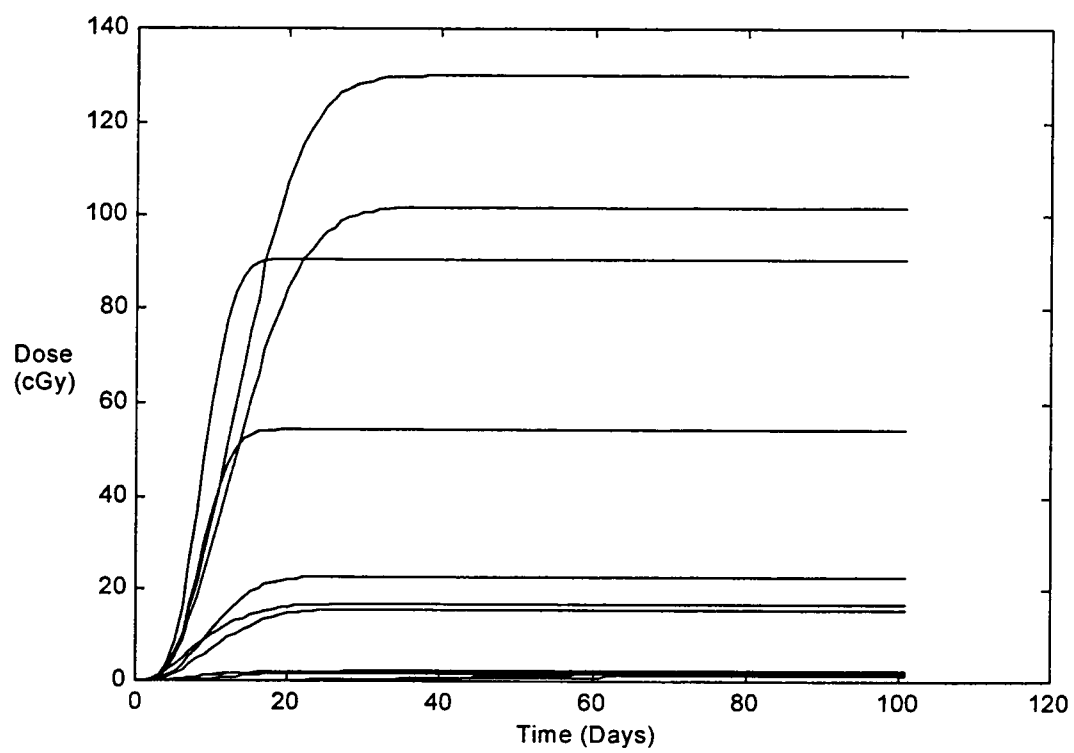
Table 4. Parametric values (Zapp 1997)

Event Date	α , 1/days			γ			X_{∞} , cGy		
	Skin	Eye	BFO	Skin	Eye	BFO	Skin	Eye	BFO
November-87	0.66	0.66	0.71	5.79	5.24	2.1	0.072	0.053	0.0055
March-89	0.23	0.23	0.26	2.79	3.18	3.07	1.64	1.09	0.066
April-89	0.61	0.62	0.49	2.06	2.03	2.66	2.41	0.151	0.0082
	0.36	0.37	0.17	1.94	1.29	1.65	0.051	0.061	0.009
May-89	0.19	0.204	0.16	1.13	1.25	1.13	0.11	0.086	0.019
August-89	1.14	1.16	1.32	2.92	2.9	2.5	90.8	54	1.73
	0.7	0.81	1.08	1.84	1.22	1.02	33.5	26.9	2.6
September-89	0.67	0.68	0.66	2.3	2.2	2.03	130	102	15.75
October-89	0.95	0.56	0.63	6.06	5.55	4.65	218	166	22.1
	0.6	0.61	0.66	4.07	4.15	5.88	108	78.2	9.6
	0.91	0.98	0.88	1.24	1.2	1.19	54.4	45.3	6.8
November-89	0.99	1.01	1.13	3.37	3.27	2.84	8.9	4.7	0.073
March-90	1.13	1.1	0.97	2.92	2.68	1.61	1.21	0.69	0.017
April-90	0.26	0.26	0.27	16.2	15.3	2.44	0.321	0.211	0.012
	1.31	0.65	0.7	3.03	0.82	1.57	0.031	0.031	0.0031
January-91	1.05	0.97	0.55	2.47	2.23	1.17	0.0855	0.062	0.0052
March-91	0.95	0.95	0.92	5.65	5.77	5.62	143.6	82.7	2.31
May-91	2.52	2.5	3.95	1.46	1.32	0.87	1.53	1.09	0.099
	0.53	0.69	0.1	1.94	2.12	0.04	0.121	0.08	0.011
June-91	0.94	1.06	1.13	2.35	1.67	2.21	22.6	16.75	1.95
August-91	0.18	0.18	0.22	7.14	14.4	3.61	0.21	0.145	0.0131



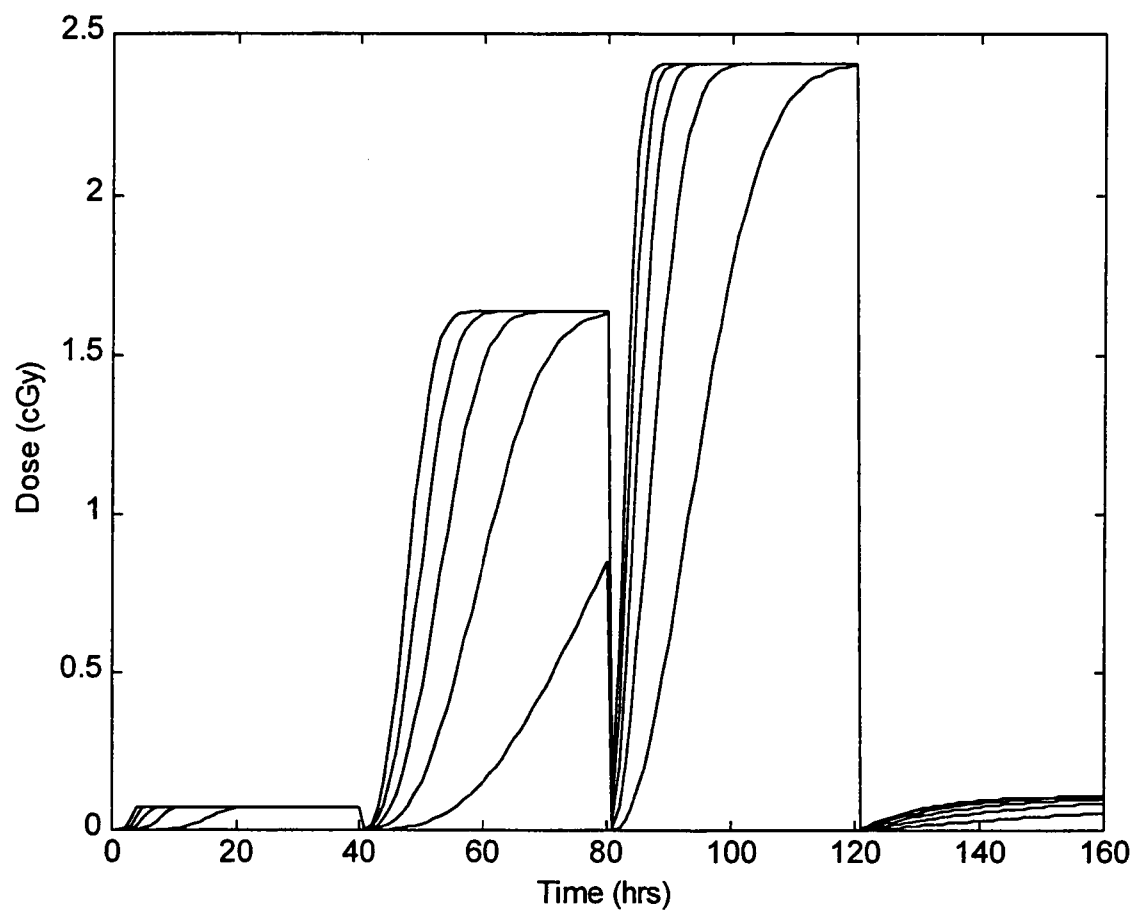
(a)

Figure 16. Dose(cGy)-time(hr) profile using the parametric data and Weibull function (a) skin only, for the SPEs of November 87 ($D_{\infty}=0.072$), March 89 ($D_{\infty}=1.64$), April 89 ($D_{\infty}=2.41$), and May 89 ($D_{\infty}=0.11$), (b) for skin, eye, and bone marrow from Table 4



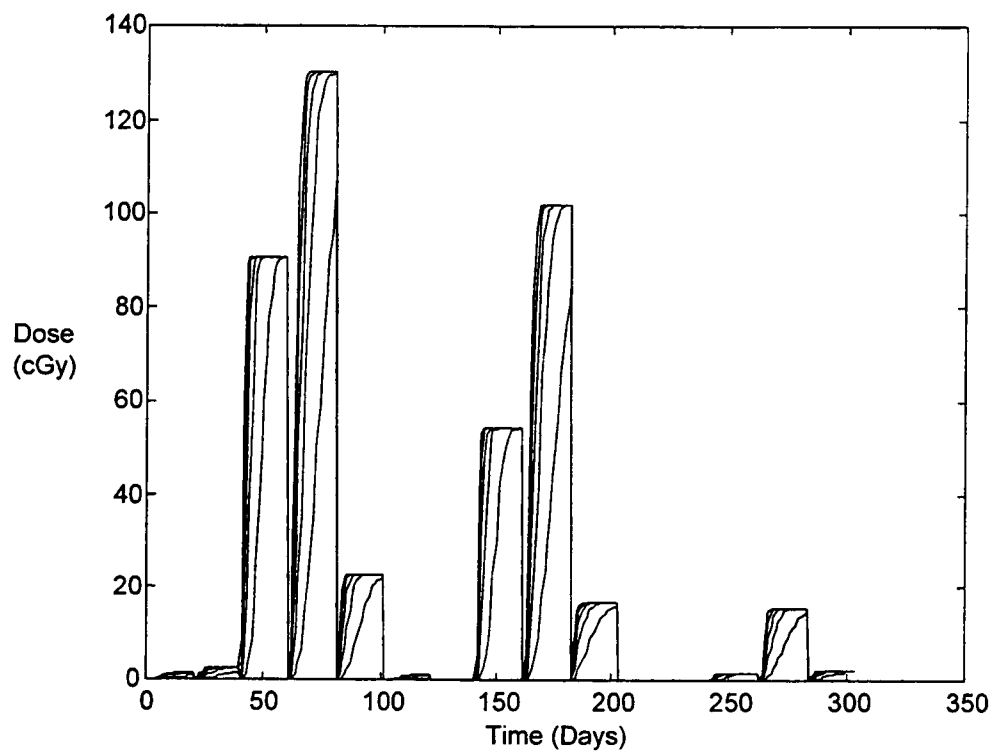
(b)

Figure 16. (continued)



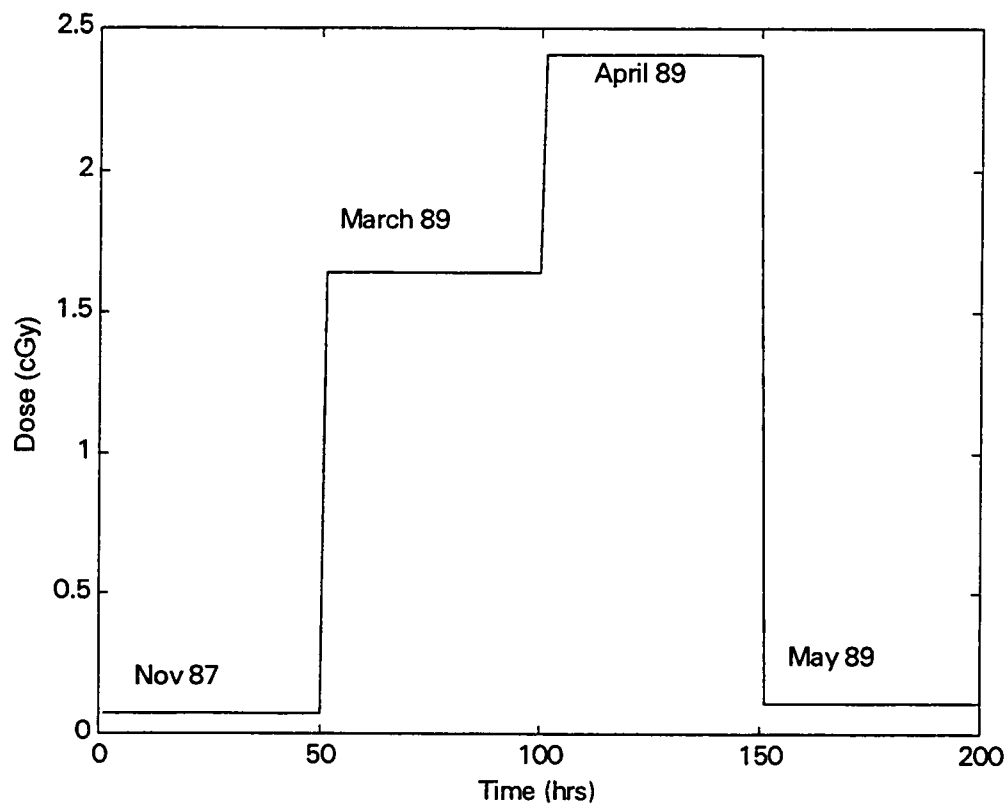
(a)

Figure 17. Sampling data to be used for STDNN. a) skin, b) skin, eye, and bone marrow



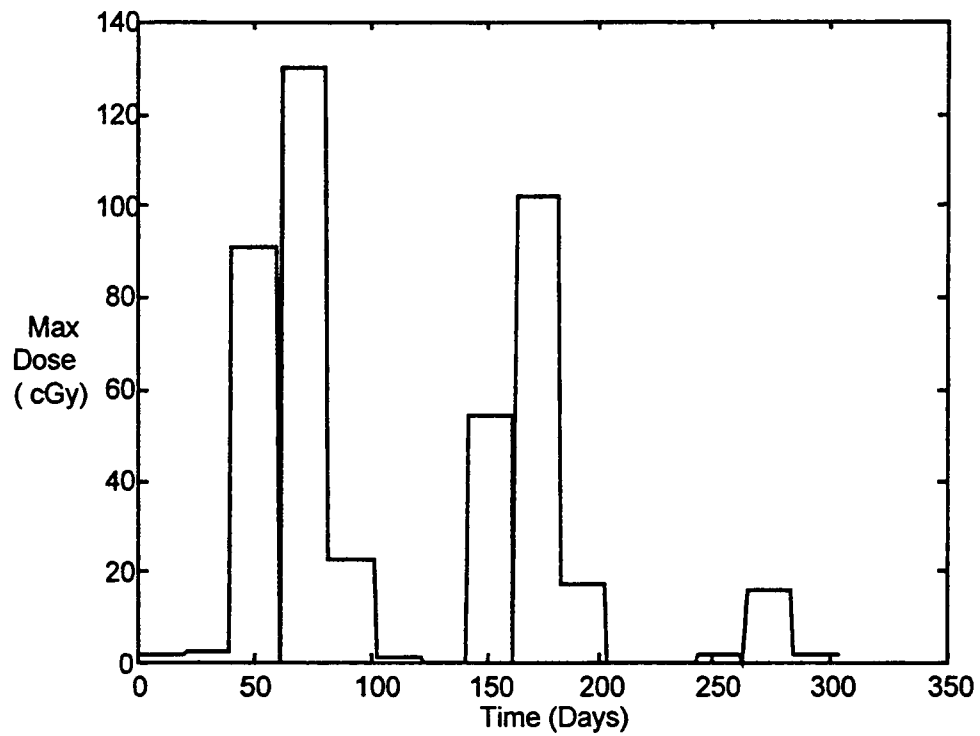
(b)

Figure 17. (continued)



(a)

Figure 18. Event total dose (cGy) a) skin only, for November 87, March 89, April 89, and May 89 SPEs, b) skin, eye, and bone marrow from Table 4



(b)

Figure 18. (continued)

To observe the neural network performance, we plot the target values and the actual outputs in Figure 19. In Figure 19 the STDNN prediction is plotted with the asymptotic dose on the same graph in order to compare the reliability of the prediction. We expect that the STDNN gives the asymptotic dose value for each event or something very close to it. From Figure 19 we see that the neural network is doing a good job of predicting the dose-time profiles. This is because the prediction overlaps the actual value. As shown in Figure 19 predictions for concatenated data sets are very close to the target values for four different events.

6.5 Error analysis

The results thus far are shown based on the number of input/output pairs. But we can also show the results in terms of time. Also we would like to find the error. We find the error using

$$\Delta D_{\infty} = (D_{\infty})_{actual} - (D_{\infty})_{pred} \quad (20)$$

or the percentage error using

$$\% \Delta D_{\infty} = \frac{(D_{\infty})_{actual} - (D_{\infty})_{pred}}{(D_{\infty})_{actual}} \quad (21)$$

and the weighted average error is calculated from

$$wt_average = \frac{\sum_{i=1}^{\#events} (D_{\infty})_{actual} \Delta D_{\infty}}{\sum_{i=1}^{\#events} (D_{\infty})_{actual}} \quad (22)$$

We can calculate the actual error from

$$E_a = D_{\infty} \times \% \Delta D_{\infty} \quad (23)$$

The percentage error ($\% \Delta D_{\infty}$) and the actual error (E_a) for parametric data are listed in Table B-2. These results are summarized and rearranged in ascending order for the 15 events in Table 6. Data used for training and testing the neural network are listed in Table 5. The percentage error $\% \Delta D_{\infty}$ and the actual error (E_a) for prediction of actual data (Table B-1) using parametric data (Table 5) are summarized in ascending order in Table 7.

The illustration of three curves, parametric data, target or the dose infinity, and dose prediction from the STDNN are on pages 86 and 87 and in Figures A-12 to A-23. In Figures 34, 35, and A-24 we can see the actual data, dose infinity, and dose prediction from the STDNN for actual data using Table 7. The MATLAB scripts for these processes are called testfff.m (Appendix G) for parametric data, and testf.m (Appendix H) for actual data. The difference between the function to be approximated by the NN and the actual value for the function is called the error. We can define the error in, for example, the i th neuron as follows:

$$e_i(n) = d_i(n) - O_i(n) \quad (24)$$

where $e_i(n)$ represents the error, $d_i(n)$ is the desired value, and $O_i(n)$ is the actual value of the function at the i th neuron.

Since the network has only one output neuron, the sum-squared error

$$E = \sum_i e_i^2(n) \quad (25)$$

is the only value needed to be calculated.

Note that using the sum-squared error has a significant advantage compared to other error constraints such as percentage error ($\% \Delta D_{\infty}$). It consists of the absolute error for each point rather than the percentage error ($\% \Delta D_{\infty}$). Thus, for example, if the dose for an event is 100 cGy and the error is 10 cGy, then the sum-squared error is 100 and the percentage error ($\% \Delta D_{\infty}$) is 10%. If, on the other hand, the dose for an event is 10 cGy and the error is 10 cGy, then the sum-squared error is the same in both cases but the percentage error is much less for the large event. Hence, training the network using an absolute error bound, such as the sum-squared error, rather than an average relative error will tend to favor error reduction for the large events, which are most important, over the small events, which are less important.

6.6 Network training and validation

MATLAB (The Math Works, 1994) software is used for training and validation. All MATLAB scripts for this research are enclosed as Appendices. In the MATLAB file called `pall.m` (Appendix E) the parametric data for the skin, eye, and bone marrow are used to plot the Weibull functions for each organ. In Figure 16 these profiles for the skin are illustrated. In Figure 27, the parametric data for events at March 89, April 89, August 89, September 89, and June 91 for all three organs are illustrated using the MATLAB script `pall.m` (Appendix E). To obtain sampled data to be used by the STDNN a MATLAB file called `process1.m` has been developed and is included in Appendix F. Examples of sampled data are illustrated in Figure 19 for skin dose only, and in Figure 17.b for all three organs.

The strategy used in this research is to use parametric and actual data for the same events. In Table 5 parametric events used for training and testing the STDNN are listed. Dose data for seven actual events are listed in Table B-1. From Table 4 we have 15 single parametric events. In this research we would like to examine the performance of the neural network using the smooth parametric data and also explore the performance of the same neural network with actual data from the transport code calculation. For different cases explored in this research we used several combination of these events for training

the network and testing the network. Table 5 shows an example of training the neural network with one subset of events and testing the network with another subset of events. We can train the network with any data set that we choose. There is no obligation to choose which set or subset for training or which specific data set for testing. One should note that 1) use a data set for training that is as large as possible 2) use the weights and biases from network training to test the network with a different data set. Even though these two rules are the most important rules in neural network techniques, but sometimes we may use some special condition for a specific problem. In our case we tried to choose, for example, the parametric events and actual events for the same dates for training and testing process. STDNN predicted better when we used the data for the same events. That is because we are in the same domain or in another words we stay in a specific range. The parametric data used for training are for the events of October 89, November 89, March 90, April 90, March 91, May 91 and August 91 for all three organs. Then the weights and biases for these events are used to test the actual data for the same events. Also there is another validation set, using the parametric data for the March 89, April 89, August 89, September 89, and June 91 events for all three organs that was not used for training the network. The actual data for events October 89, November 89, March 90, April 90, March 91, May 91 and August 91 for all three organs are shown in Table B-1. The actual data from Table B-1 are regenerated using MATLAB script actf.m (Appendix C) and plotted in Figures 20, 21 and A-5 to A-9.

In Figure 25 we tested the data for October 89, November 89, March 90, April 90, March 91, May 91 and August 91 for all three organs with the weights from training the same parametric data. The weights and biases from this training is shown in Table B-3 in Appendix B.

In Figure 26 the actual data for October 89, November 89, March 90, April 90, March 91, May 91 and August 91 for all three organs have been tested with the weights and biases from Table B-3. In Figure 27 the parametric data for March 89, April 89, August 89, September 89, and June 91 for all three organs are tested with the weights and biases from Table B-3. From Figure 27 we can see that the trained STDNN is able to generalize, because it does well predicting events that were not used for training.

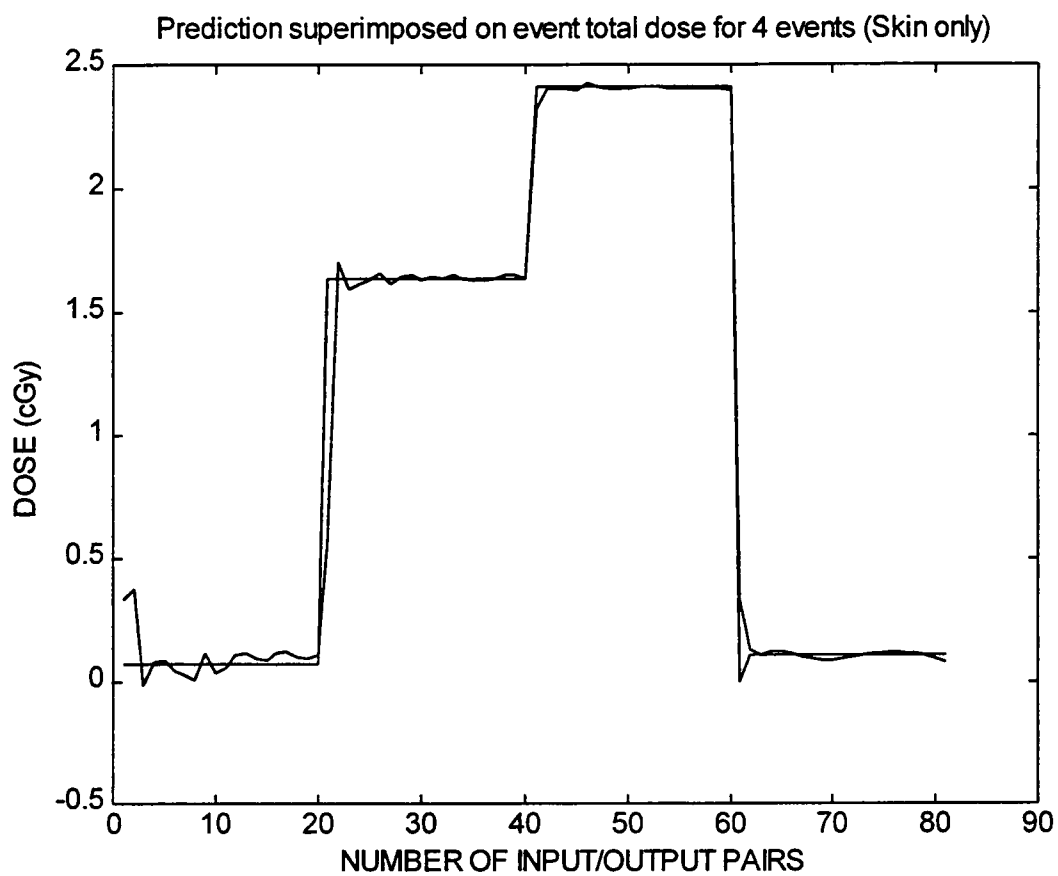


Figure 19. Neural network results after training the network with parametric data. Skin dose is plotted versus the number of input/output pairs

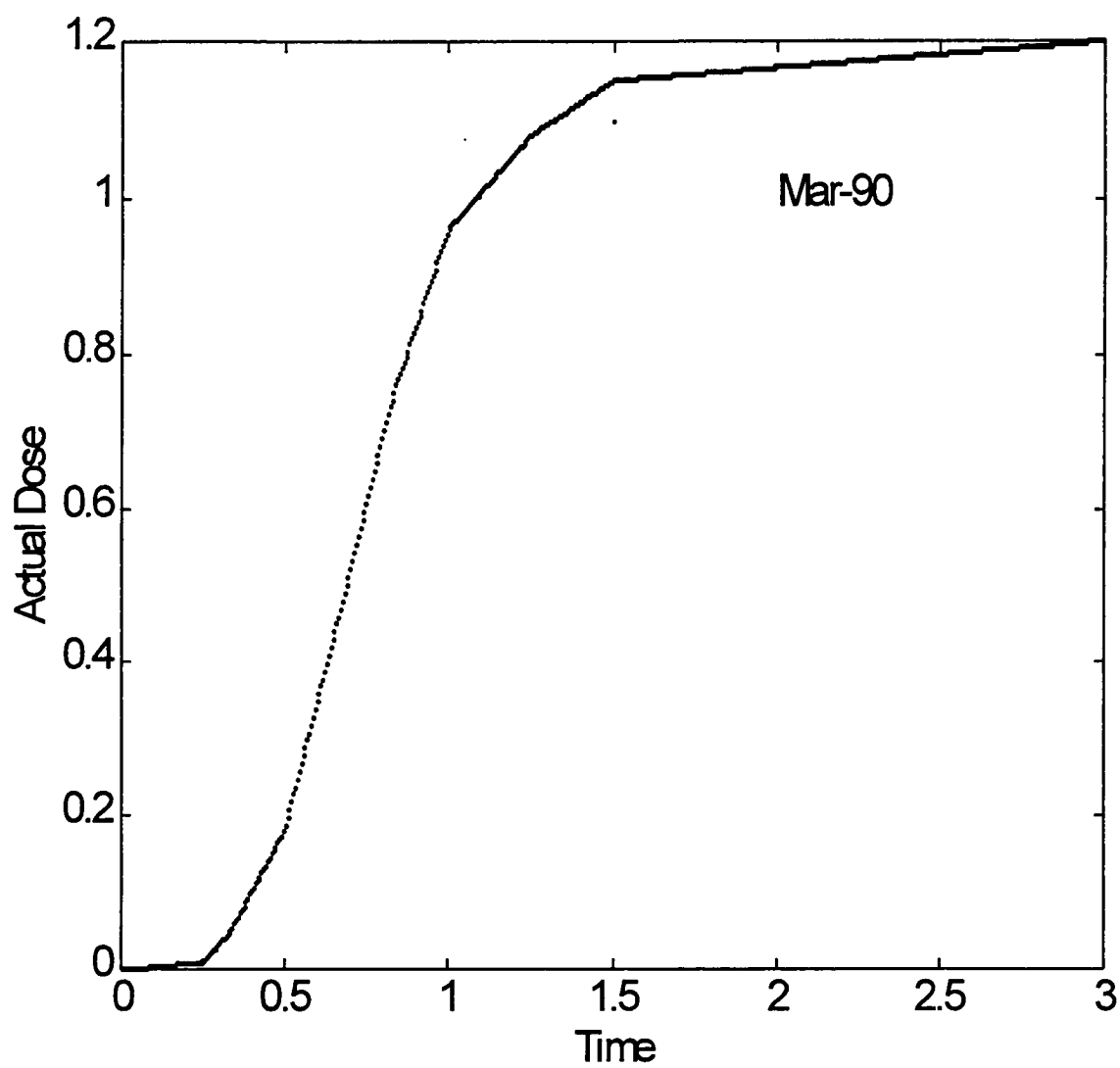


Figure 20. March 90 event Dose (cGy) and Time (days).

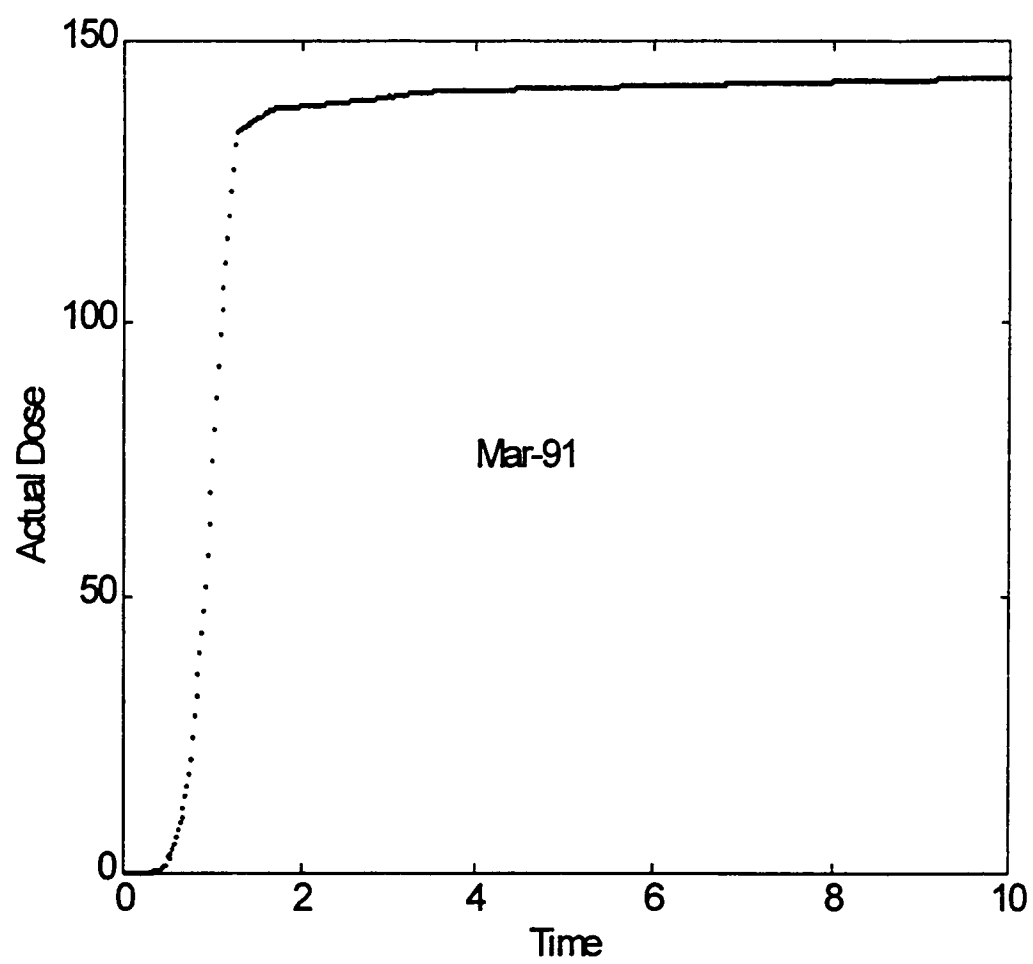


Figure 21. March 91 event Dose (cGy) and Time (days).

Table 5. Parametric events used in this research a) training, b) testing

Event Date	α , 1/days			γ			D_{∞} (cGy)		
	Skin	Eye	BFO	Skin	Eye	BFO	Skin	Eye	BFO
October 89	0.95	0.56	0.63	6.06	5.55	4.65	218	166	22.1
November 89	0.99	1.01	1.13	3.37	3.27	2.84	8.9	4.7	0.073
March 90	1.13	1.1	0.97	2.92	2.68	1.61	1.21	0.69	0.017
April 90	0.26	0.26	0.27	16.2	15.3	2.44	0.321	0.211	0.012
March 91	0.95	0.95	0.92	5.65	5.77	5.62	143.6	82.7	2.31
May 91	2.52	2.5	3.95	1.46	1.32	0.87	1.53	1.09	0.099
August 91	0.18	0.18	0.22	7.14	14.4	3.61	0.21	0.145	0.0131

(a)

Event Date	α , 1/days			γ			D_{∞} (cGy)		
	Skin	Eye	BFO	Skin	Eye	BFO	Skin	Eye	BFO
March 89	0.23	0.23	0.26	2.79	3.18	3.07	1.64	1.09	0.066
April 89	0.61	0.62	0.49	2.06	2.03	2.66	2.41	0.151	0.0082
August 89	1.14	1.16	1.32	2.92	2.9	2.5	90.8	54	1.73
September 89	0.67	0.68	0.86	2.3	2.2	2.03	130	102	15.75
June 91	0.94	1.06	1.13	2.35	1.67	2.21	22.6	16.75	1.95

(b)

Table 6. Results of error calculation for 15 parametric events from Table 4 (single events only). The events are in ascending order of their D_{∞} values.

Event	Dose Infinity	Percentage Error	Actual Error
	D_{∞}	$\% \Delta D_{\infty}$	$\% \Delta D_{\infty} \times D_{\infty}$
1	0.0082	-273.3085	-2.2411
2	0.066	-33.4913	-2.2104
3	0.151	-14.7016	-2.2199
4	1.09	-1.3799	-1.5041
5	1.64	-0.7831	-1.2843
6	1.73	-1.149	-1.9878
7	1.95	-1.0028	-1.9554
8	2.41	-0.783	-1.887
9	15.75	-0.0022	-0.0347
10	16.75	0.0054	0.0913
11	22.6	0.0368	0.8326
12	54	-0.0549	-2.9653
13	90.8	-0.0032	-0.2904
14	102	0.0025	0.2557
15	130	0.0084	1.0925

Table 7. Results of error calculation for 7 actual events (Table B-1) predicted by STDNN using parametric data for training (Table 5)

	Error	Dose Infinity	Actual Error
Event	ΔD_{max}	D_{max}	$\Delta D_{max} * D_{max}$
1	-175.413	0.013	-2.2804
2	-140.67	0.016	-2.2507
3	-124.725	0.018	-2.245
4	-31.1119	0.0712	-2.2152
5	-20.2	0.11	-2.222
6	-0.8394	2.3	-1.9305
7	-1.3335	38.426	-51.24

Chapter 7

Results and discussion

7.1 STDNN training and validation

In this chapter we present results for the processes of training and generalizing using the STDNN. When a network generalizes, it means that the input-output relationship computed by the network is correct for test data that were not used in training the network, assuming that the test data are drawn from the same population used to generate the training data. Training of a neural network may be viewed as a curve fitting problem. Then generalization may be looked on as the effect of a good nonlinear interpolation of the input data. The results of this work are divided for discussion purposes into the following cases:

Case 1 - Training the network with a small data set, testing the network with a small data set:

First we present results using parametric data for training and for testing. In Figure 16.a the parametric data, from Table 4, for skin are illustrated. In Figure 16.b the parametric data for skin, eye, and bone marrow are shown. Results in Figure 16.a are for the November 87, March 89, April 89, and May 89 events for skin doses only. As explained in chapter 6, we performed the sampling procedure on these data to train the sliding time delay neural network. In Figure 17 each event in Figure 16 is divided to five events equally spaced in time. In Figure 18 the maximum dose (D_{∞}) for the same events are plotted. The training algorithm LMTRAINP.m (Appendix D) is used to perform the network training. The results of the STDNN prediction using the training data are shown in Figure 19 (skin doses only). The network is doing very well predicting the dose values. At the end of each transient there is overlap of the predicted values and the target values. This is to be expected here since the network is being tested against the data that were used to train it. In Figure A-1 to A-4 we choose two other different events that the

network was not trained with. We plot the parametric data for these two events. We perform the preprocessing routine. Then we check the generalization capability of this STDNN. The two events that have not been used in training process are being tested with weights and biases determined using our training set. From these results, the STDNN performance is fair at best due to the lack of data points. To generate these results we used only four or five events to train the network with. We also used only two events for testing the network. In our next attempt we will use more data to do the same procedure.

Case 2 - Test the actual data with case 1 weights and biases

For the next study we use actual data from the transport code calculations. These are actual data compared to the parametric data, which are very smooth. In Figures 20, 21, and A-5 to A-9 these actual data have been regenerated and plotted using a MATLAB script very similar to actf.m but with fewer data points (Appendix C). Comparing these curves with the curves from Figure 16 we see that there is scatter in the data. Before we start over with a larger data set we first try another attempt to test these actual data with our STDNN weights and biases. Figure 22 shows the target values and predictions from the STDNN for these actual data. The agreement is very good for the large events. Some events however have doses smaller than the minimum doses used in training set and as we can see from Figure 22 their predictions are not in as good agreement.

Case 3 - Training the network with a large data set, testing the network with a large data set:

Now we will use more data points for training. In Figure 23 we use all of the parametric data from Table 4 regardless of the date of the event or the type of the organ. We follow the same procedure explained above and the same data set with the resulting weights and biases. Figure 23 displays the results of testing all of the parametric dose values with the weights and biases from training the same data set. The network is doing an excellent job in predicting the doses.

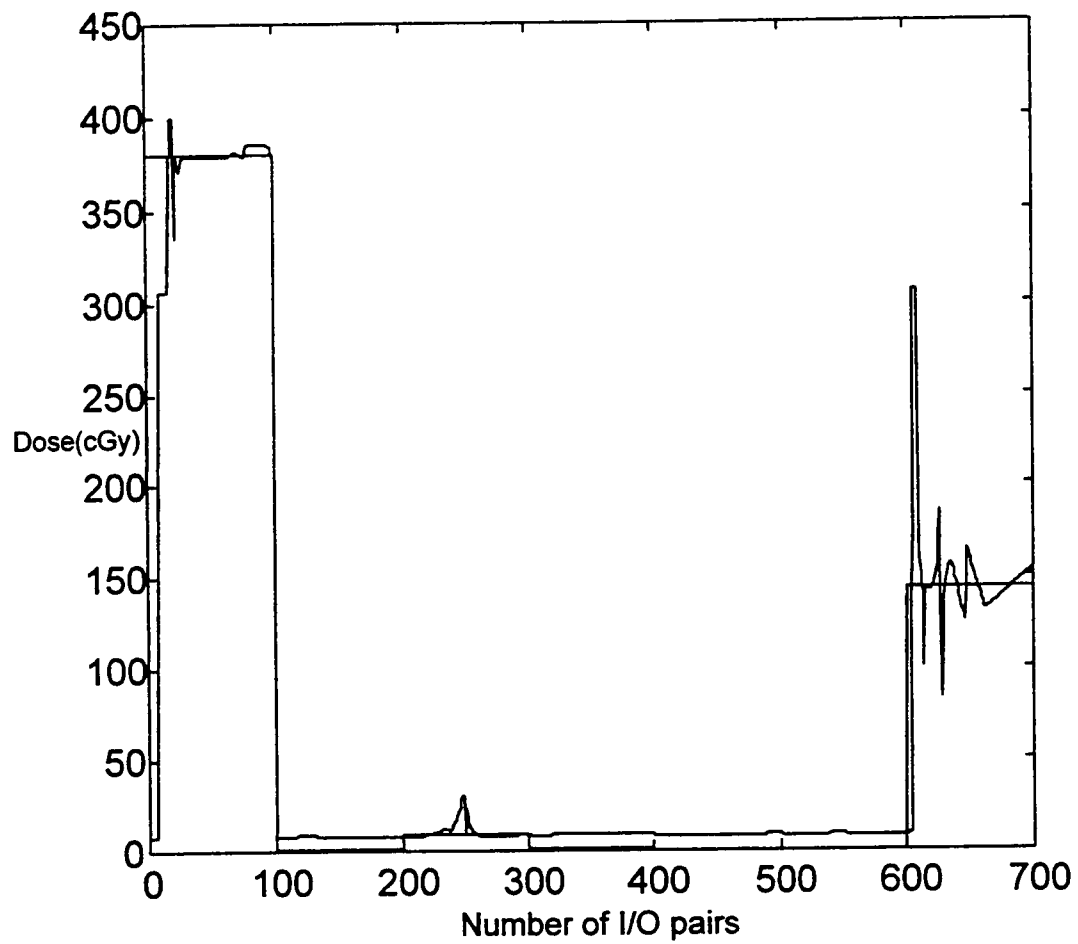


Figure 22. Training the NN using the actual data.

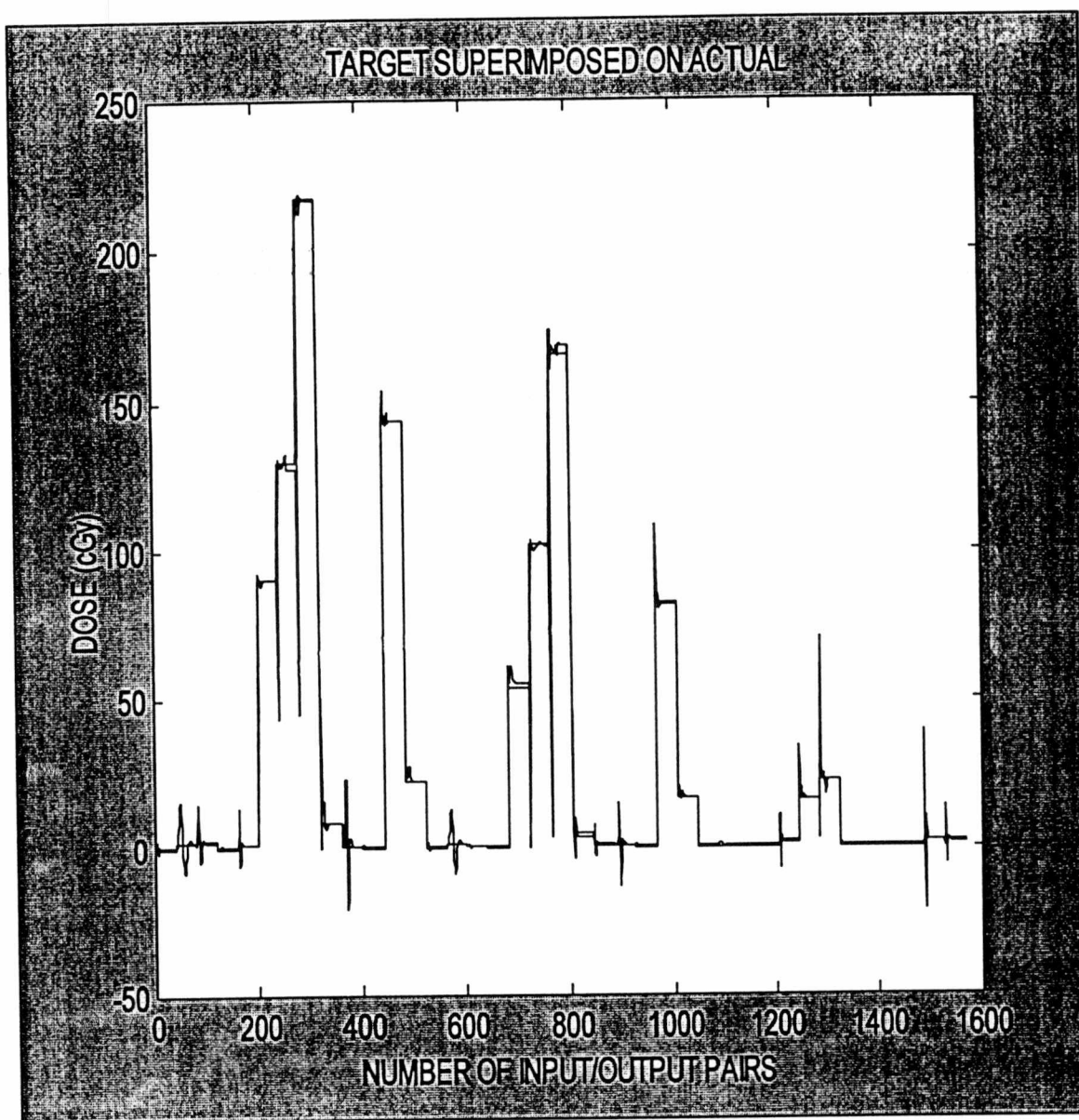


Figure 23. Train using parametric data, test using parametric data (All events from Table 4)

Case 4- Training the network with actual data, testing the network with actual data

In this research we have two different types of data; actual and parametric. It turns out that it does not matter what type of data we use to train the neural network. In the appendix A there is a case that illustrates what happens when actual data for skin are used to train the network and then tested using the same set with the resulting weights and biases (A-10). The resulting prediction is good. Better results might be obtained if more data for training were available.

Case 5 - Hybrid training

There are several alternative methods to train a network. For example, we can do hybrid training (Figure 24). For this case, we first use all of the parametric data to train the neural network. Then we use the resulting weights and biases as initial values to retrain the network using the actual dose data. Now the network has information about the parametric data as well as actual data. In Figure 24 we use weights from parametric data as initial values to train the network for skin doses only and then test the predictions against the same actual data with these hybrid weights and biases. The agreement is very good but we have not introduced a new data set since we are testing with the same actual data used to train the network. This shows that we are not limited to initializing the weights of a network only with some random numbers. We can use the weights from training the parametric data as initial value for weights when retraining with the actual data. This hybrid training gives information about parametric data as well as actual data to the neural network. Finally, we have more freedom for testing this kind of hybrid network with either set. In the appendix other hybrid network results are shown. We trained the network with actual data first. Then we used the resulting weights and biases as initial values to retrain the network with parametric data. The resulting weights and biases from this hybrid network are used to test the actual data. From results in appendix A we can see that prediction does not match the target values in some events, this is

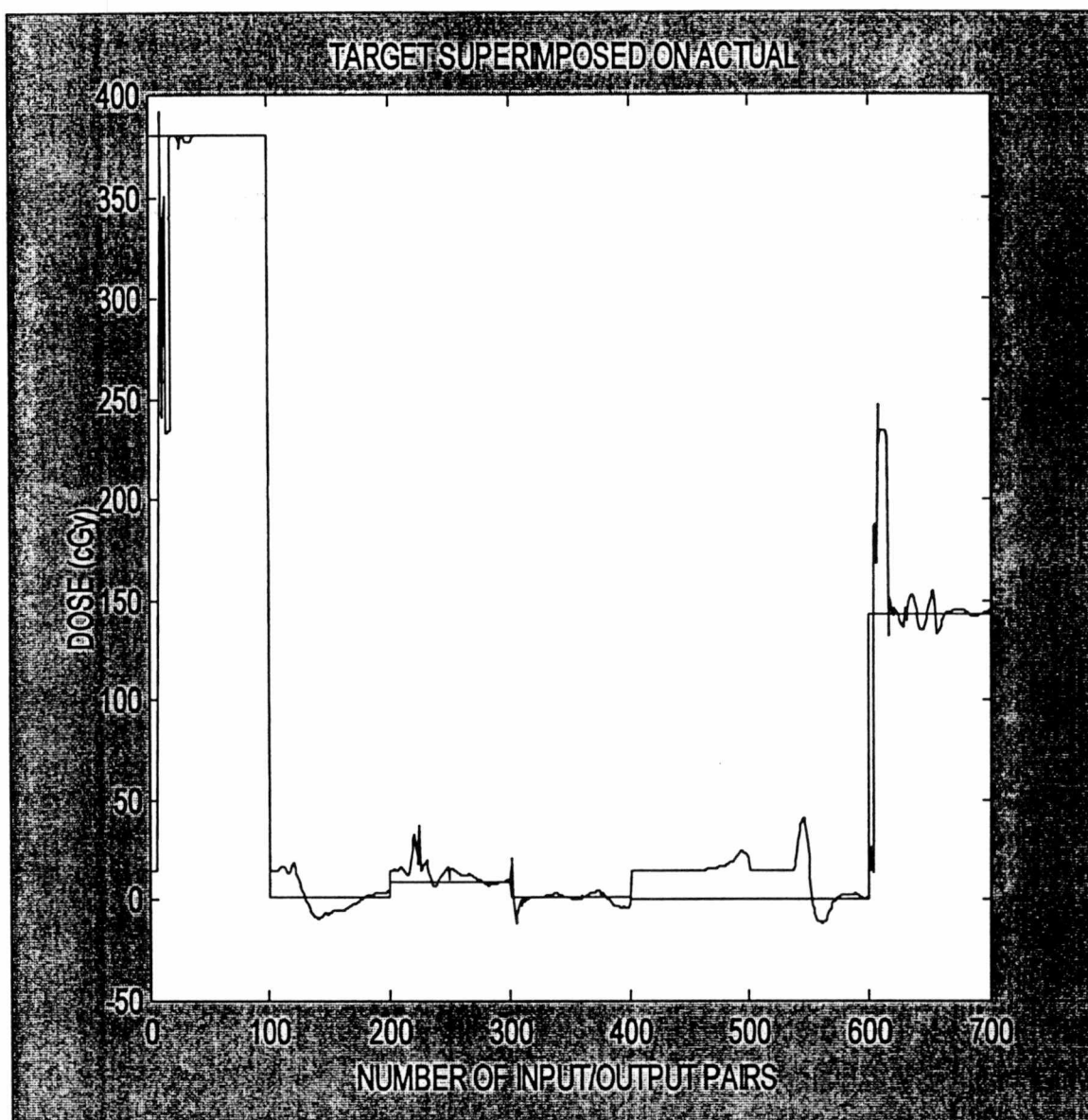


Figure 24. Test with actual data with weights from training using parametric weights as initial values. (Skin dose only)

because the testing set is outside of the training set range (A-11).

Case 6 - Train with parametric data, test with actual data for the same events

Next, we select parametric and actual or actual data that happened at the same time for training and testing. In Figure 25 the data for events in October 89, November 89, March 90, April 90, March 91, May 91, and August 91 are selected to train the network. Then we tested these same events with the resulting weights and biases. The predictions look good as can be seen in Figure 25. Then we use the weights and biases from training the parametric data, and test with the actual data for the same events (Figure 26). Even though the STDNN has not previously seen any actual data, the predictions are in good agreement with the actual data. In two events, however, the neural network performed poorly. This is because the network has not been trained with any actual data, and network has always seen smooth data from the parameterization.

Case 7 - Generalization of STDNN

In this section we explore the generalization capability of the STDNN. We start by dividing the parametric data from Table 4 into two subsets. We train the network with data from October 89, November 89, March 90, April 90, March 91, May 91, and August 91 events. We call these the first data subset. Then we use the rest of parametric data (second subset) as the validation set. The validation set is used for testing the neural network. Testing the neural network is done to see if the network can generalize. In Figure 27 we plot the data from the second subset. These are events in March 89, April 89, August 89, September 89, and June 91. We perform the preprocessing routine on this data subset as seen in Figure 17.b. Figure 18.b shows the target values for second subset events. In Figure 27 we illustrate the generalization ability of the neural network. We test the second data subset events with the weights and biases of the first data subset.

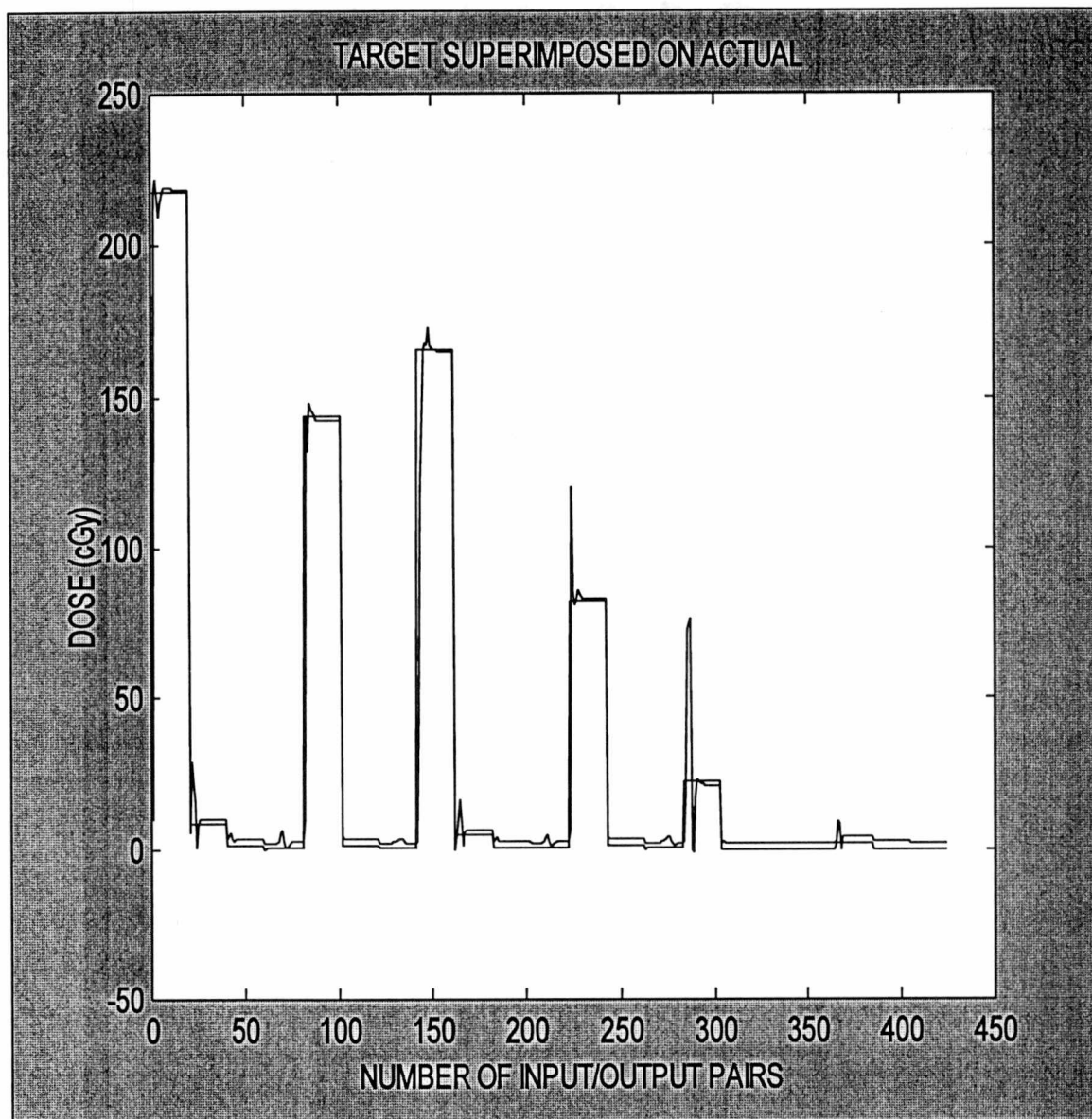


Figure 25. Test of parametric data for Oct 89, Nov 89, Mar 90, Apr 90, Mar 91, May 91, Aug 91 with weights from training using the same parametric data

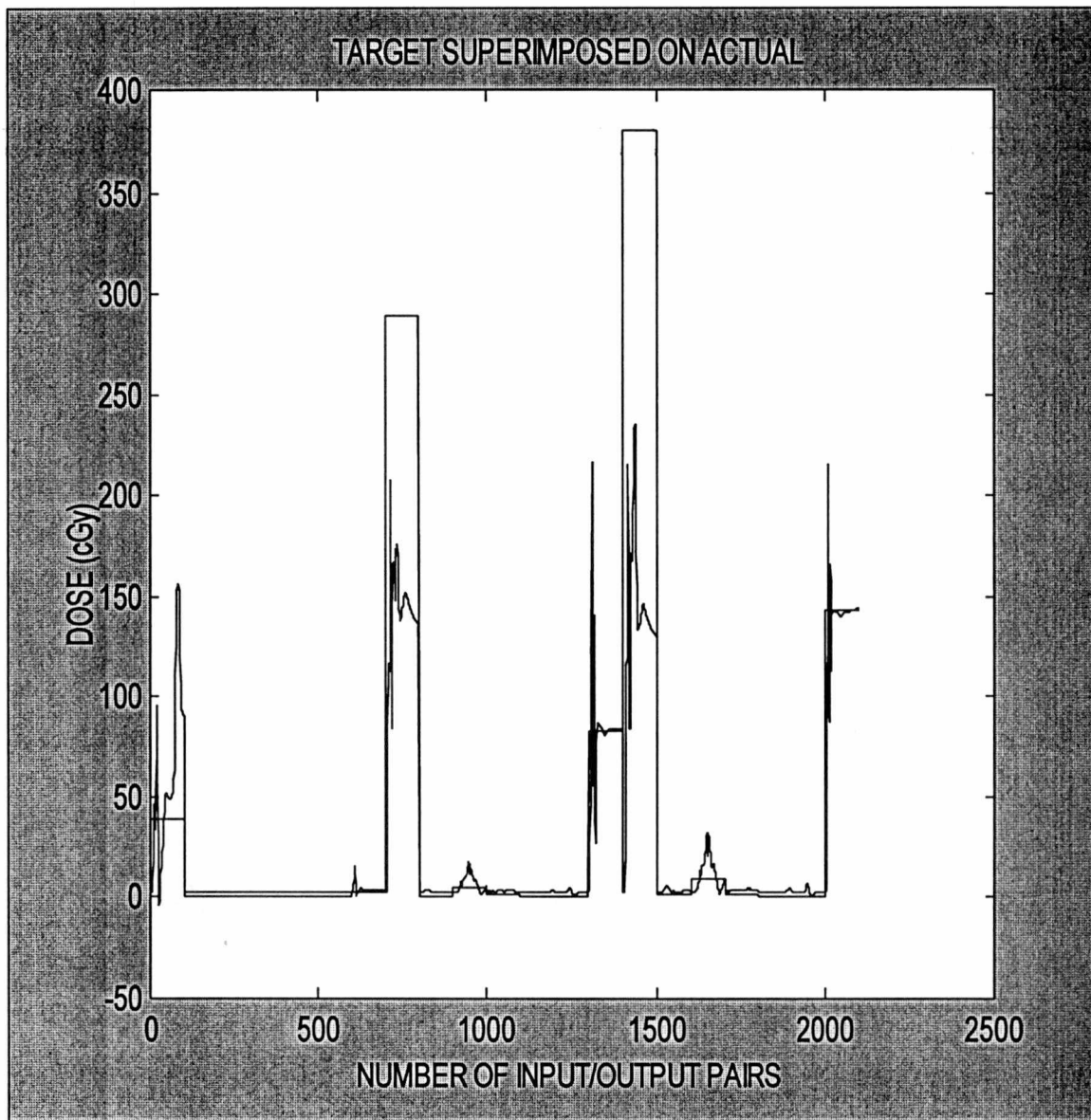


Figure 26. Test of actual data for Oct 89, Nov 89, Mar 90, Apr 90, Mar 91, May 91, Aug 91 with weights from training with the parametric data for the same events

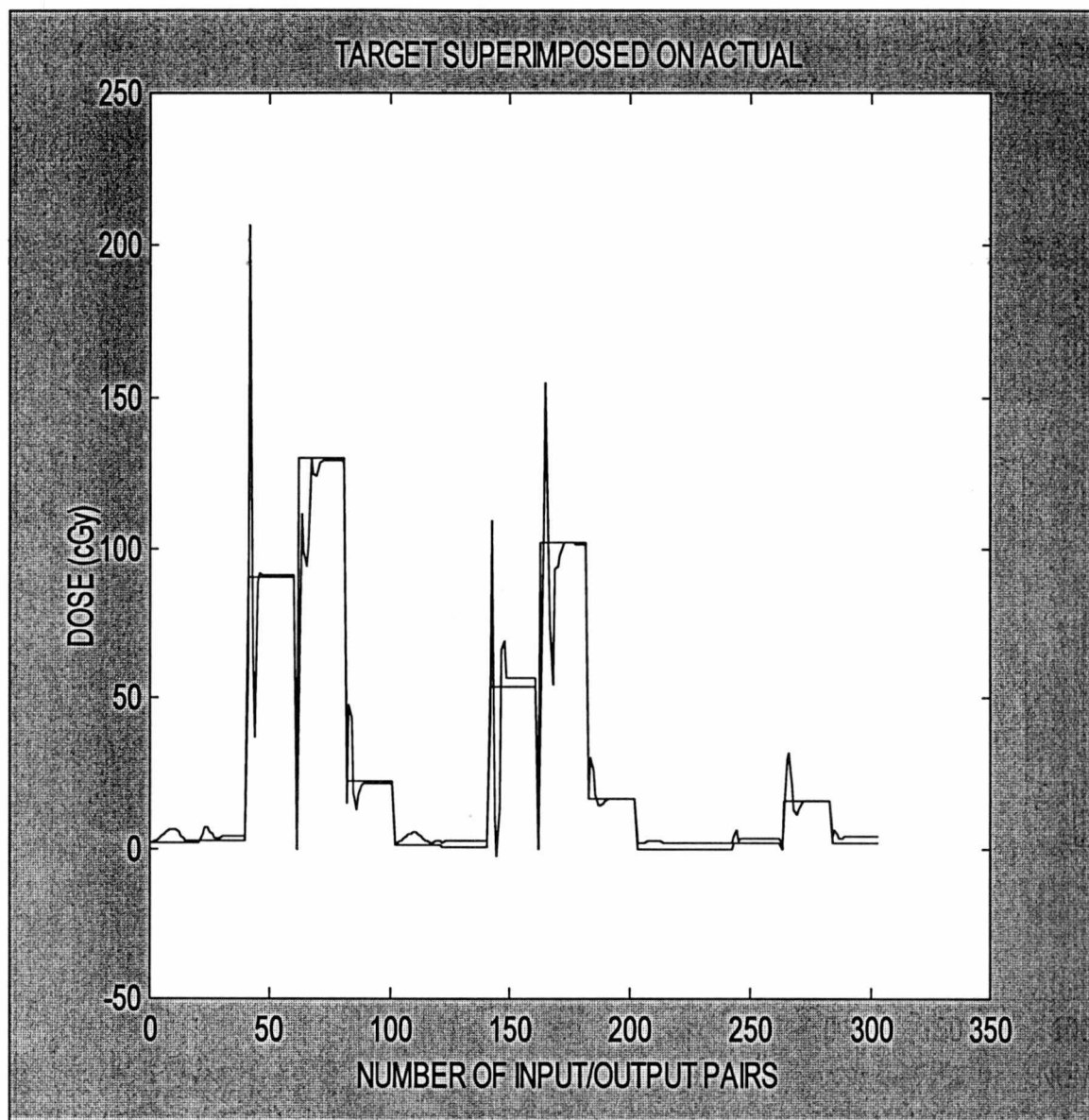


Figure 27. Test of parametric data for Mar 89, Apr 89, Aug 89, Sep 89, and June 91 with weights from training with the parametric data for Oct 89, Nov 89, Mar 90, Apr 90, Mar 91, May 91, and Aug 91

The STDNN does a very good job of predicting doses for events that it has not been trained with before.

Case 8 - Plotting the STDNN predictions in dose-time frame for parametric data

For most results presented thus far, such as Figure 27, we express the prediction results using the number of input/output pairs, which come from the preprocessing process where we had to make the data equally spaced in time. We performed this procedure using `process1.m` in Appendix F. It would be useful if we could see all of these dose predictions as a function of time since event onset. The MATLAB script `testfff.m` in Appendix G is used to plot the parametric data, target value, and the prediction from STDNN on one single figure. On pages 86 and 87 and on Figures A-12 to A-23 illustrate these three plots for 15 different parametric events for skin, eye, and bone marrow. These figures are plotted using data from Table B-2. The plots are arranged in order from small dose events to large dose events. In the appendix we have shown cases with very small maximum dose (plus signs show the maximum dose). From the scale of the dose axis the error is noted to be very small. As we move on to larger events we see that the STDNN performs better. This has also been discussed in the methodology chapter. Figures 28 and 30 show good prediction results for larger events using the STDNN.

Case 9 - Plotting the STDNN predictions in dose-time frame for actual data

In this section we follow the procedure explained in the previous section using actual dose data from the transport calculations. The MATLAB script to do this task is in Appendix H and is called `testf.m`. It plots the actual or actual data, the predictions of them, and the target values altogether on the same graph. Figures 31 and 32 show these results. The target value is shown using the `+` symbol.

In carrying out the previously-discussed optimization procedure one must aware of overfitting or overtraining problems, where the networks learn only to recreate the

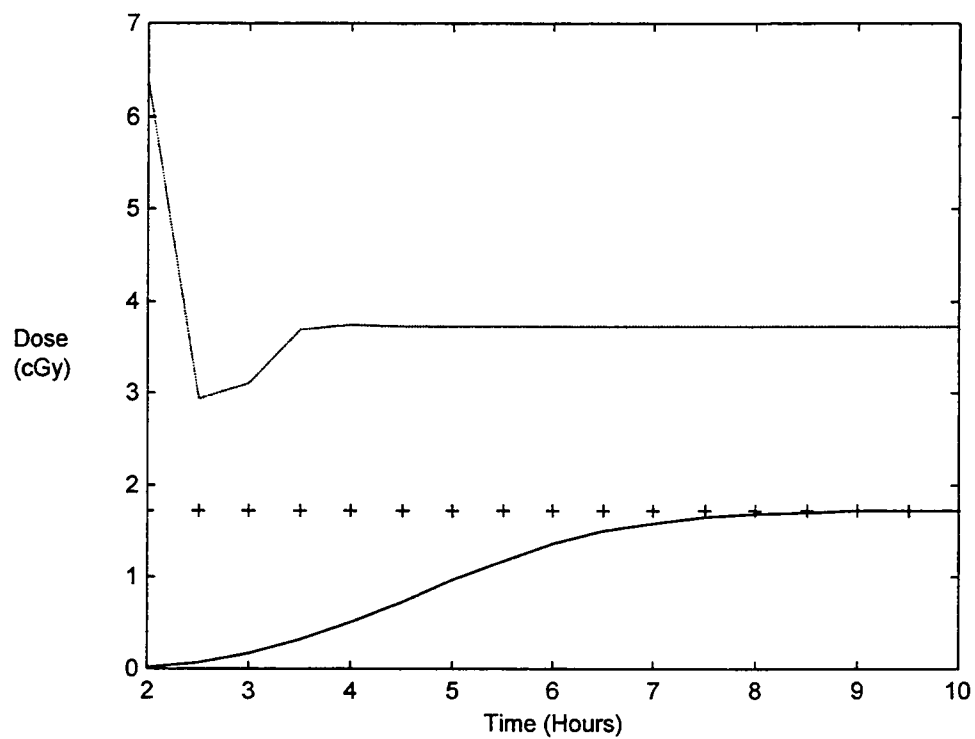


Figure 28. Illustration of parametric data (solid line), asymptotic dose (+ symbol) and the prediction from STDNN (broken line) for event 6 (August 89) in Table 6

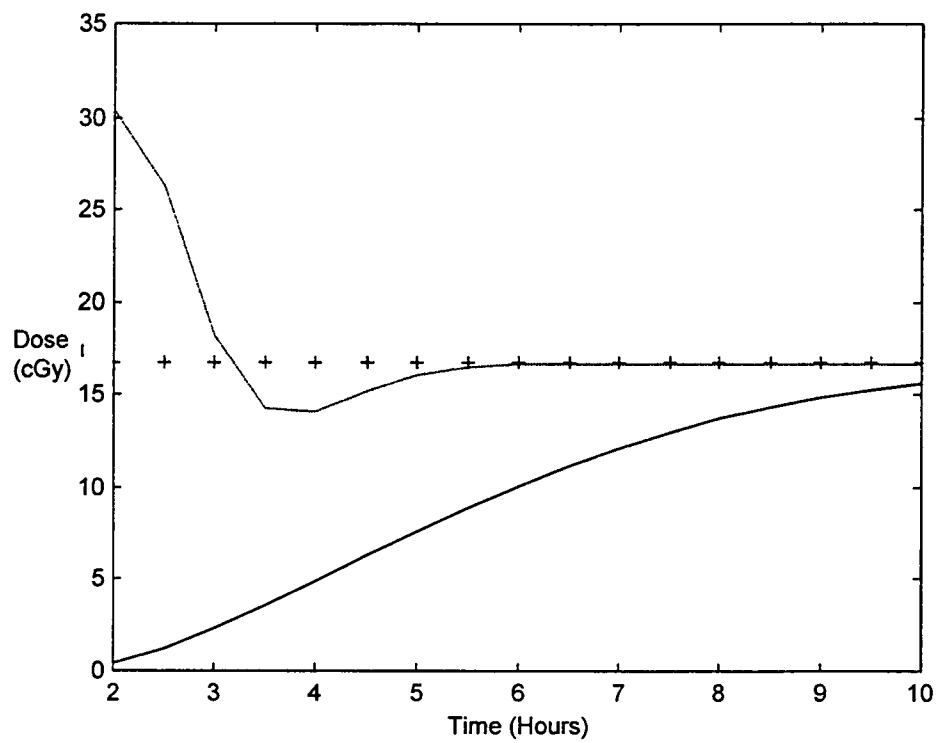


Figure 29. Illustration of parametric data (solid line), asymptotic dose (+ symbol) and the prediction from STDNN (broken line) for event 10 (June 91) in Table 6

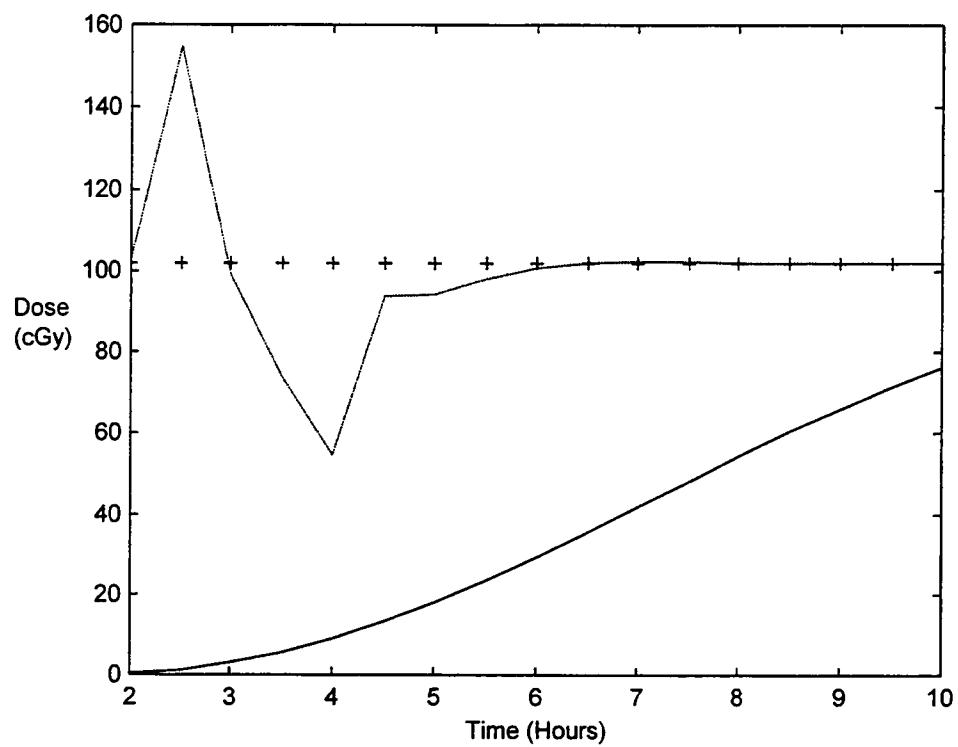


Figure 30. Illustration of parametric data (solid line), asymptotic dose (+ symbol) and the prediction from STDNN (broken line) for event 14 (September 89) in Table 6

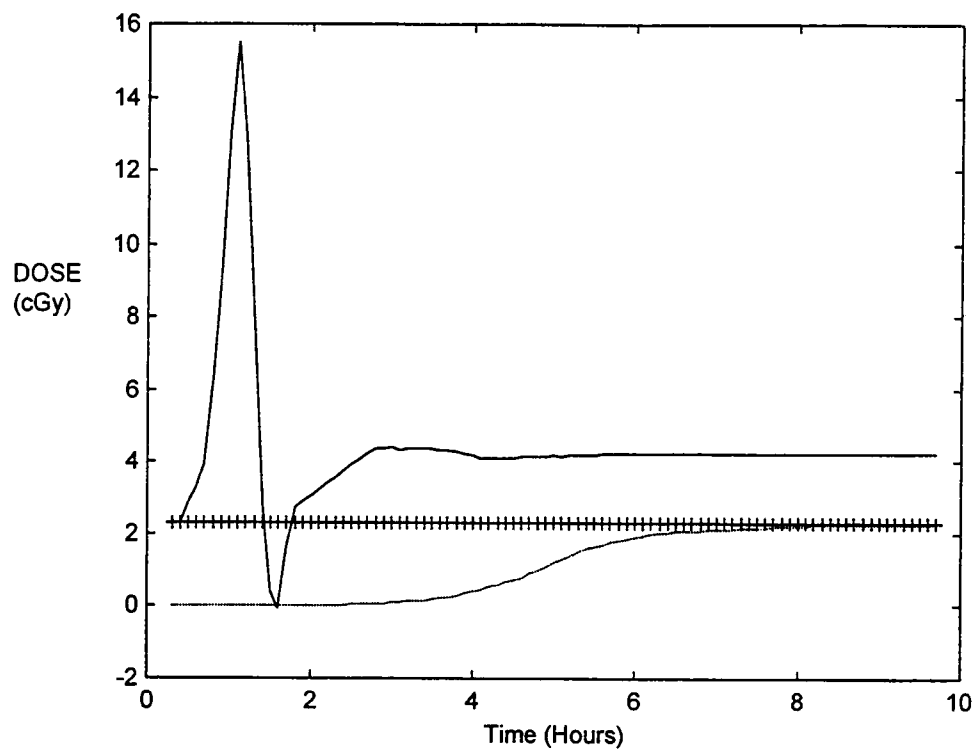


Figure 31. Illustration of actual data (dashed line), asymptotic dose (+ symbol) and the prediction from STDNN (solid line) for event 6 (March 91) in Table 7

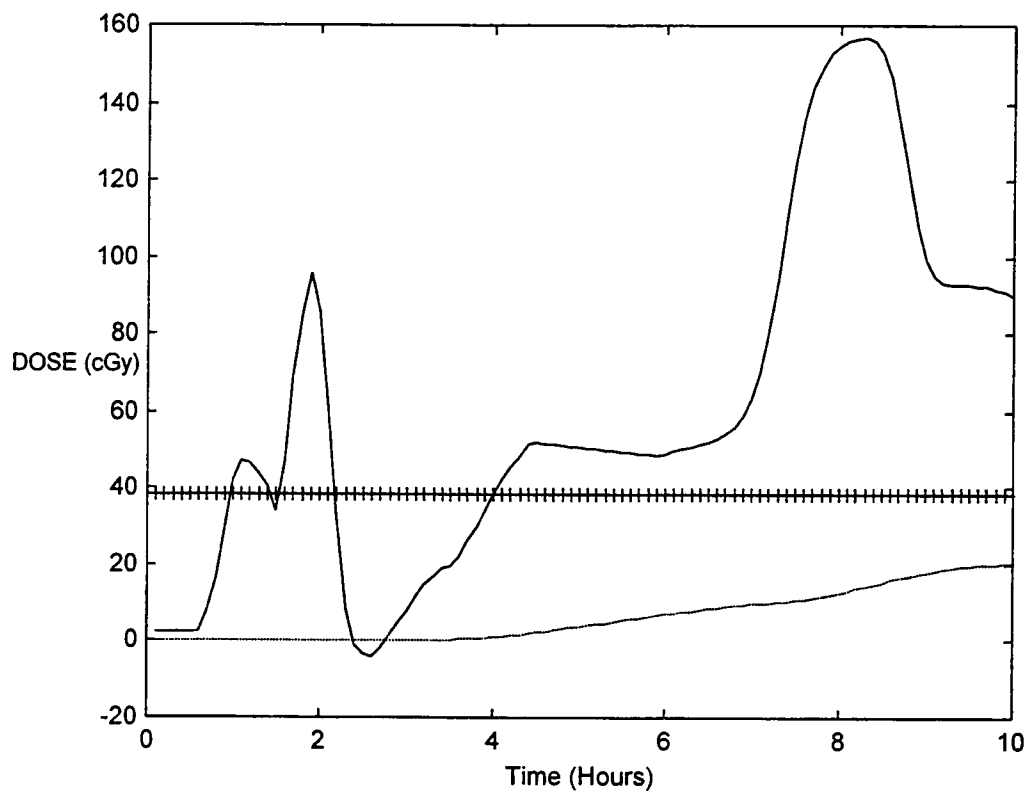


Figure 32. Illustration of actual data (dashed line), asymptotic dose (+ symbol) and the prediction from STDNN (solid line) for event 7 (October 89) in Table 7

noise present in the input data and lose their ability to generalize to previously unseen data. This loss of generalization ability, is avoided in this work by splitting the data into two subsets, using one larger subset to train the network while another smaller "test set" (sometimes called a 'validation set') is used to monitor the prediction accuracy on data for which the network is not trained. The errors on both subsets are monitored and if the error on the test-set is increasing while the error on the training set is still decreasing then the network is overfitting the training data and losing its ability to generalize.

Chapter 8

Conclusions and recommendations for future work

8.1 Conclusions

In this research on solar particle event dose predictions a sliding time delay artificial neural network is trained and validated using parametric and actual data sets. These data sets follow the form of a cumulative Weibull function. Several different methods for training and testing are carried out. The network is also tested with new data sets to check the generalization.

The network architecture used in this research consisted of five input neurons, ten hidden neurons, and one output neuron. A variety of training sets are used. The sliding time delay neural network performed well in validation and generalization in nearly all cases. We conclude that a sliding time delay neural network has the capability to predict the astronauts doses from large solar particle events in a temporal fashion. Error analysis has been done, and the result is that for larger, more important events ANN performs much better than smaller, less important events. The percentage error for large events is as low as 0.22% for training and testing the smooth parametric data. The actual error is as low as $\pm 3.47\%$, after approximately 5% of the event time elapsed. From the prediction results, we can see that, at the very beginning of the transients some fluctuations from the target values exists, but after 5% of the event time has elapsed, the predictions nearly overlap the target dose values.

8.2 Recommendations for future work

In this final section recommendations for future work for predicting astronauts doses from large solar particle events using artificial intelligence are presented:

1. Develop methods for multiple components SPEs using sums of Weibull curves. In this research only single events are used. But, from Table 4 there are some

multiple component SPEs that can be used to explore for the ANN prediction ability (The October 1989 SPE is an example).

2. The larger the training set the better the result of validation and generalization. One possible improvement is to use larger training sets than were used herein for extra error reduction. Also in this research we used skin, eye, and bone marrow doses. In the future other organs and also whole body doses may be attempted as they become available.

3. Different network architectures, such as a recurrent neural network which is suitable for temporal estimation can also be used. Different sizes of sliding time delay neural networks can be use for the sake of comparison or optimization of training time. Each of these should be investigated.

4. Other innovative hybrid STDNNs can be tried. Also other fitting functions can be tried. The cumulative Weibull function worked out well for this research but other mathematical models should be explored.

References

Atwell, W., 1990, "Astronaut Exposure to Space Radiation: Space Shuttle Experience", Society of Automotive Engineering paper no. SAE-901342.

Badhwar, G.D., 1995, "User needs in the area of human presence". Naval Research Laboratory Space Weather Workshop, Arlington, Virginia.

Billings, M.P. and Yucker, W.R., 1973;" The Computerized Anatomical Man (CAM) Model", NASA contractor report no. CR-134043.

Calvo, R.A., H.A. Coccatto, R.D. Piacentini, 1995, "Neural network prediction of solar activity, *Astrophys. J.*, 444, pp. 916-921.

Chilton, A.B.; Shultis, J.K., 1984;"Principles of Radiation Shielding"; Prentice-Hall, Inc., Englewood Cliffs, NJ 07632

Cichocki, A.; Unbehauen, R., 1994 ; "Neural Networks for Optimization and Signal Processing"; John Wiley and Sons Ltd & B.G. Teubner, Stuttgart .

Conway, A.J., K.P. Macpherson, J.C. Brown, 1998, "Delayed time series predictions with neural networks", *Neurocomputing*, 18, pp. 81-89.

Cybenko, G., 1989, "Approximation by Superpositions of a Sigmoidal Function", *Mathematics of Control, Signals, and Systems*, 2, pp. 303-314.

Day, S.P., and M.R. Davenport, 1993. "Continuous-time temporal back-propagation with adaptive time delays." *IEEE Transactions on Neural Networks*, 4, 348-354.

Demuth, H., 1994, "Neural Network Toolbox", The Math Works Inc.

Fletcher, R., 1987; "Practical Methods of Optimization"; John Wiley and Sons Ltd .

Forde, G, 1997; "Application of Artificial Neural Networks in the Prediction of Maximum Absorbed Dose During a Major Solar Particle Event"; Master of Science Thesis, The University of Tennessee, Knoxville.

Funahashi, K., 1989, "On the approximate Realization of Continuous Mappings by Neural Networks", Neural Networks, 2, pp. 183-192.

Gallant, Stephen I., 1993; "Neural Network Learning and Expert Systems"; Massachusetts Institute of Technology .

Goldberg, D., 1989, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison Wesley Publishing Company.

Gorney, D.J. , 1990, "Solar cycle effects on the near-Earth space environment", Rev. Geophys., 28, pp. 315.

Grahn, D., editor; 1973, "HZE Particle Effects in Manned Spaceflight". National Academy Press. Washington, D.C.

Hall, Eric J.; 1994, "Radiobiology for the Radiobiologist" Fourth Edition; J.B. Lipincott Company .

Haykin, S., 1994, Neural Networks, A Comprehensive Foundation, MacMillan, New York.

Haykin, S., 1996, "Neural Networks Expand Signal Processing Horizons", IEEE Signal Processing Magazine, March, pp. 24-49.

Hines, J.W., 1996a, "Logarithmic Neural Networks for Function Approximation", International Conference on Neural Networks, Washington, DC.

Hines, J.W., Uhrig, R.E., Black, C., Wrest, D., and Xu, X, 1996b, "Instrument Surveillance and Calibration Verification System", Sandia National Laboratory Contract AQ-6982 Final Report.

Hornick, K., Stinchcombe, M., and White, H., 1989, "Multilayer Feedforward Networks are Universal Approximators", Neural Networks, 2, pp. 359-366.

ICRP Publication 60 , 1991; "1990 Recommendations of the International Commission on Radiological Protection"; Pergamon Press Inc., Maxwell House, Fairview Park, Elmsford, New York 10523, USA.

ICRU Report 51 , 1993; "Quantities and Units in Radiation Protection Dosimetry"; International Commission on Radiation Units and Measurements, 7910 Woodmont Avenue, Bethesda, Maryland 20814 USA.

Jakimiec, Maria; Bartkowiak, Anna; 1994, " Short-term Solar Flare Predictions by Distance-based Regression I. Bearaler Regions in 1988 and 1989 - Continuous Predictors"; Acta Astronomica, 44 pp. 115-140.

Jordan, M.I., 1986, "Attractor Dynamics and Parallelism in a Connectionist Sequential Machine", Proceeding of the 8th Annual Conference of the Cognitive Science Society, pp. 531-546.

Khan, Faiz M.; 1994, "The Physics of Radiation Therapy, Second Edition"; Williams & Wilkins .

Lang, K.J. and Hinton, G.E., 1988, "The Development of the Time-Delay Neural Network Architecture for Speech Recognition", Technical Report CMU-CS-88-152, Carnegie-Mellon University, Pittsburgh, PA.

Lapedes, A and Farber, R , 1987; "Nonlinear signal processing using neural networks: prediction and system modeling, Technical Report LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, NM.

Letaw, J.R., Silberberg, R., Tsao, C.H., 1989, "Radiation Hazards on Space Mission Outside the Magnetosphere", Adv. Sp. Res., 9, (10) 285-(10) 291.

Levenberg, K.; 1944, "A Method for the Solution of Certain Non-linear Problems in Least Squares"; Quarterly Journal of Applied Mathematics, 2 , pp.164-168.

Lin, D.-T, J.E. Dayhoff, and P.A. Ligomenides, 1992, "Adaptive time-delay neural network for temporal correlation and prediction." In SPIE Intelligent Robots and Computer Vision XI:Biological, Neural Net, and 3-D Methods, Vol.1826, pp.170-181. Boston, MA .

Lin, D.-T, P.A.Ligomenides, and J.E. Dayhoff, 1993, "Learning spatiotemporal topology using an adaptive time-delay neural network". World Congress on Neural Networks, Vol. 1,pp.291-294, Portland, OR.

Lin, Tsung-Nan, Lee Giles, Bill Horne and Sun_Yuan Kung, November 1997 , "A delay damage model selection algorithm for NARX neural networks", IEEE Transactions on Signal Processing, Vol 45, No. 11.

Longair, Malcolm S., 1994, "High Energy Astrophysics"; Cambridge University Press .

Macpherson, K.P., A.J. Conway, J.C. Brown, 1995, "Prediction of solar and geomagnetic activity data using neural networks, J. Geophys. Res. 100 (A11), 21, pp. 735-744.

Marquardt, D. W.; 1963, "An Algorithm for Least Squares Estimation of Non-linear Parameters"; J.SIAM, 11 , pp. 431-441.

McNish, A. and J. Lincoln, 1949, "Prediction of Sunspot Numbers", Trans. Amer. Geophys. J., 30, pp.673.

Mundt, M.D.; Maguire II, W.B.; R.R.P. Chase, 1991, "Chaos in the Sunspot Cycle: Analysis and Prediction, J.Geophys. Res., 96 (A2) ,pp 1705-1716.

NCRP, 1989; "Guidance on Radiation Received in Space Activities". Report no 98, National Council on Radiation Protection and Measurements. Bethesda, MD.

NeuralWare PREDICT, 1995, NeuralWare Inc., Pittsburgh Pennsylvania, pp. 8.1-8.5, Rumelhart, D., E., eds., 1986, Parallel distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, MIT Press, Cambridge, MA.

Parks, George K., 1991, "Physics of Space Plasmas", Addison Wesley Publishing Company .

Priest, E.R.; 1985, "Solar System Magnetic Fields"; D. Reidel Publishing Company.

Rumelhart, D.E., and J.L. McClelland, eds., 1986, "Parallel Distributed Processing: Explorations in the Microstructure of Cognition", Vol. 1. Cambridge, MA: MIT Press.

Smart, D.F. and Shea, M.A., 1997, "Solar Radiation", Encyclopedia of Applied Physics. The American Institute of Physics, New York, N.Y. Vol 18, VCH Publishers Inc.

Svarer, C., L. K. Hansen, and J. Larsen, 1992, "On design and evaluation of tapped-delay neural architectures," in Proc. IEEE Int. Conf. Neural Networks. pp. 46-51.

Tan, Ah-Hwee and Loo-Nin Teow , 1995, "Learning by Supervised Clustering and Matching", IEEE-International-Conference on Neural Networks Conference Proceedings, Vol. 1, pp 242-246, RWCP Neuro ISS Laboratory.

Teh; H. H. , 1993, "A new class of neural networks called neural logic networks: First technical report," Technical Report TR-93004, Real World Computing Partnership.

Teh; H. H., 1994, "A new class of neural networks called neural logic networks: Second technical report," Technical Report TR-94013, Real World Computing Partnership.

Tong, H. and K. S. Lim, 1980, "Threshold autoregression, limit cycles and cyclical data," J. R. Stat. Soc. B, Vol. 42, pp. 245.

Townsend, L. W., Cucinotta, F.A. and Wilson, J.W., 1992a, "Interplanetary Crew Exposure Estimates for Galactic Cosmic Rays", Rad. Res., 129, pp. 48-52.

Townsend, L.W., Wilson, J.W., Shinn, J.L. and Curtis, S.B., 1992b, "Human Exposure to Large Solar Particle Events in Space", Adv. Sp. Res., 12, pp. (2) 339-(2) 348.

Wan, E.A., 1990, "Temporal Backpropagation for FIR Neural Networks", IEEE International Joint Conference on Neural Networks", 1, pp. 575-580, San Diego, CA.

Williams, K.E., 1991, "Prediction of solar activity with a neural network and its effect on orbit prediction", Technical Digest 12 (4), John Hopkins A.P.L., MD.

Weigend, A.S., B. A. Huberman, and D.E. Rumelhart, 1990, "Predicting the future: A connectionist approach," International Journal of Neural Systems, Vol. 1, no. 3, pp. 193-209.

Weigend, A.S., B. A. Huberman, and D.E. Rumelhart, 1992, "Predicting sunspots and exchange rates with connectionist networks", in: M. Casdagli, S. Eubank (Eds.), Proc. Workshop on Nonlinear Modeling and Forecasting, SFI Studies in the Sciences of Complexity, Addison-Wesley, Reading, MA , pp 395-432.

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.; 1989, "BRYNTRN: A Baryon Transport Model". NASA Technical Paper 2887.

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. , December 1991; "Transport Methods and Interactions for Space Radiations" NASA Reference Publication 1257.

Wilson, John W.; Badavi, Francis F.; Cucinotta, Francis. A; Shinn, Judy L.; Badhwar, Gautam D.; Siberberg, R.; Tsao, C. H.; Townsend, Lawrence W.; Tripathi, Ram K. , 1995; "HZETRN: Description of a Free-Space Ion and Nucleon Transport and Shielding Computer Program". NASA Technical Paper 3495.

Zapp, E.N. , 1997, "Solar Particle Event Dose Distributions: Parameterization of Dose-Time Profiles, with Subsequent Dose-Rate Analysis", Master of Science Thesis, The University of Tennessee, Knoxville.

Appendices

Appendix A

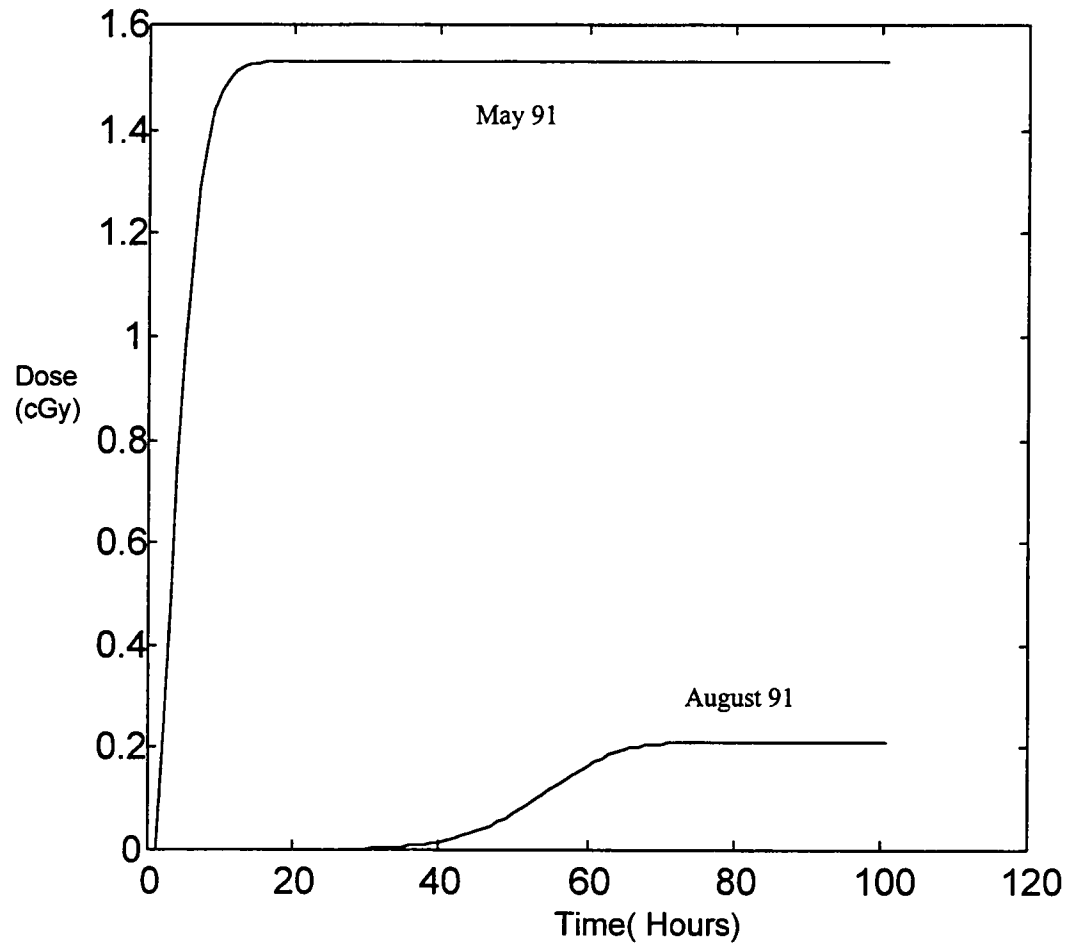


Figure A-1. Two events that are not used in training the network

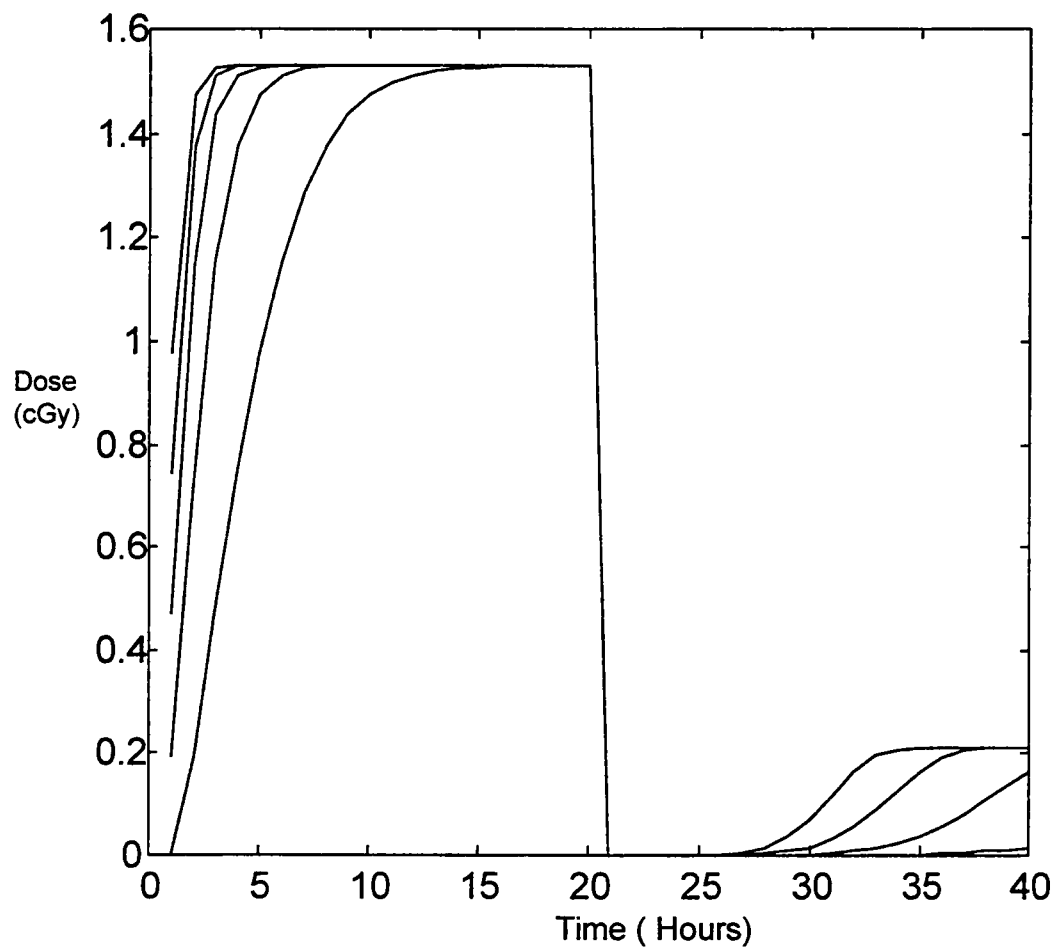


Figure A-2. Sampling the two unused events of Figure A-1

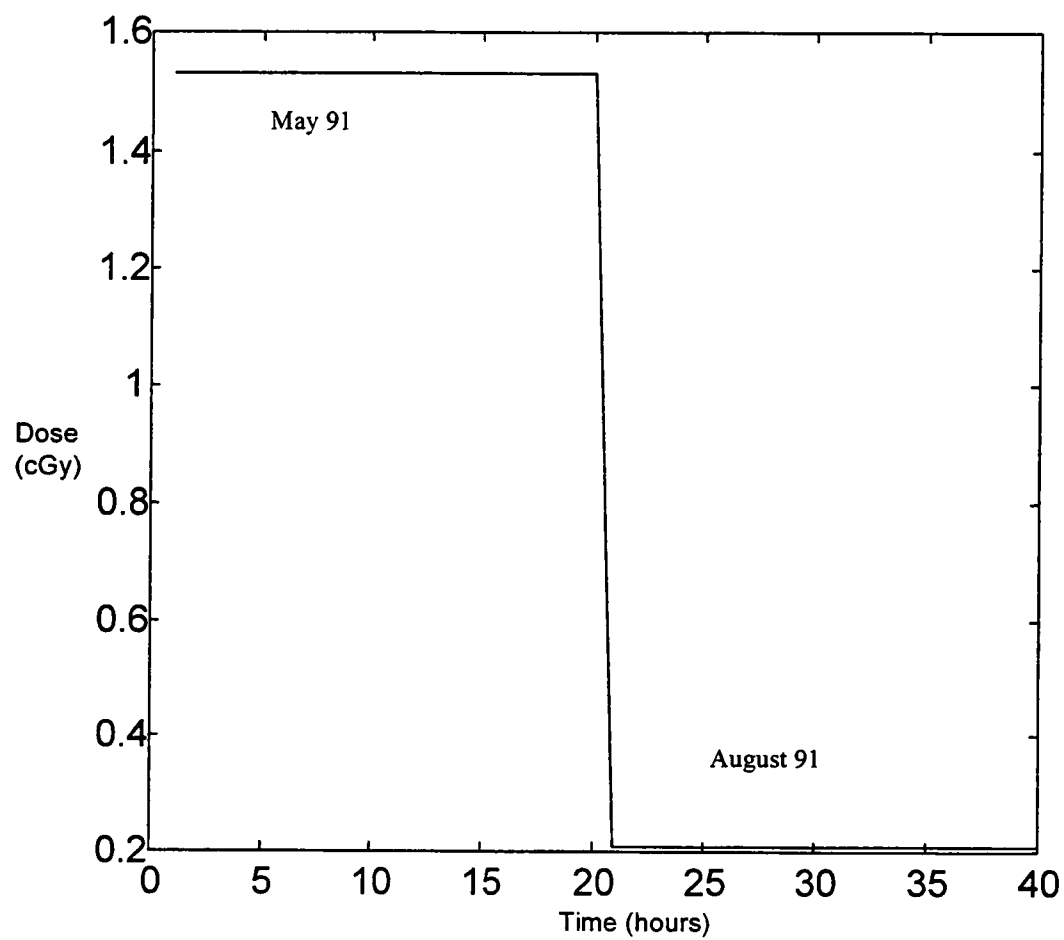


Figure A-3. Asymptotic doses for the two unused events.

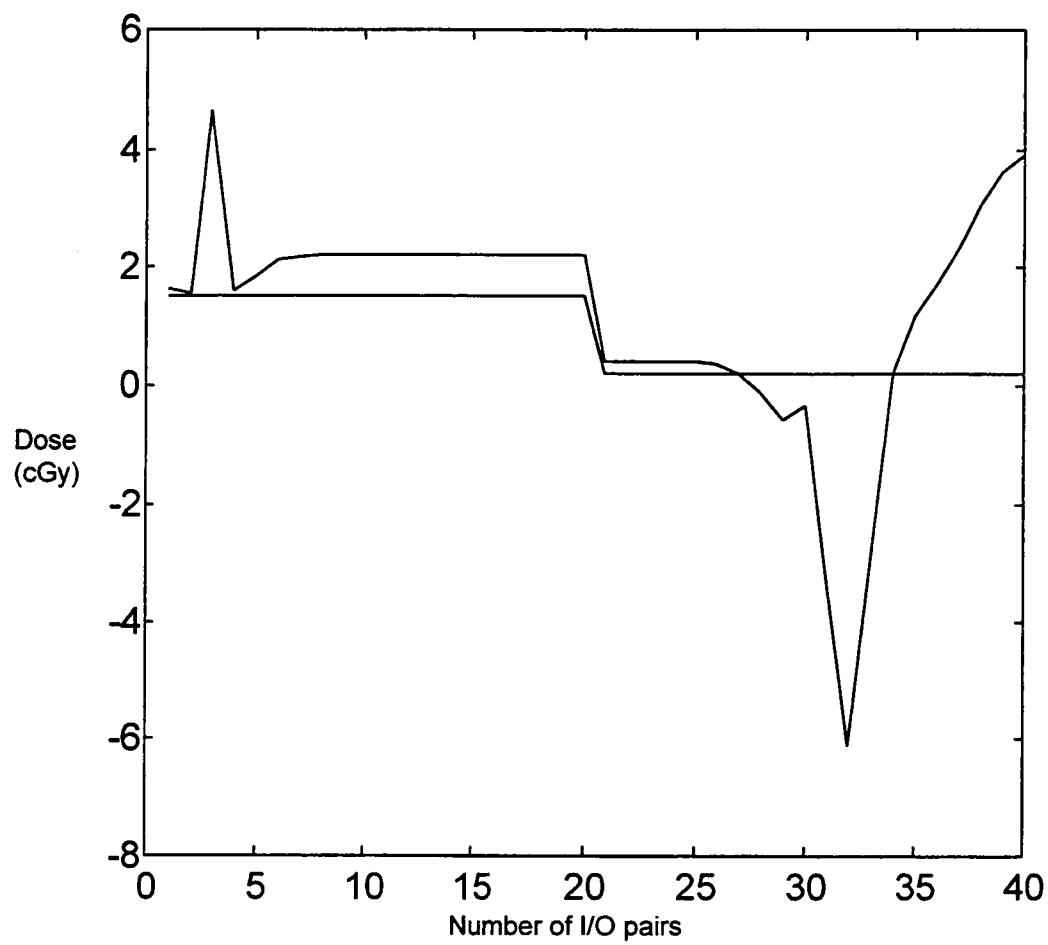


Figure A-4. Testing the two unused events with the trained network weights.

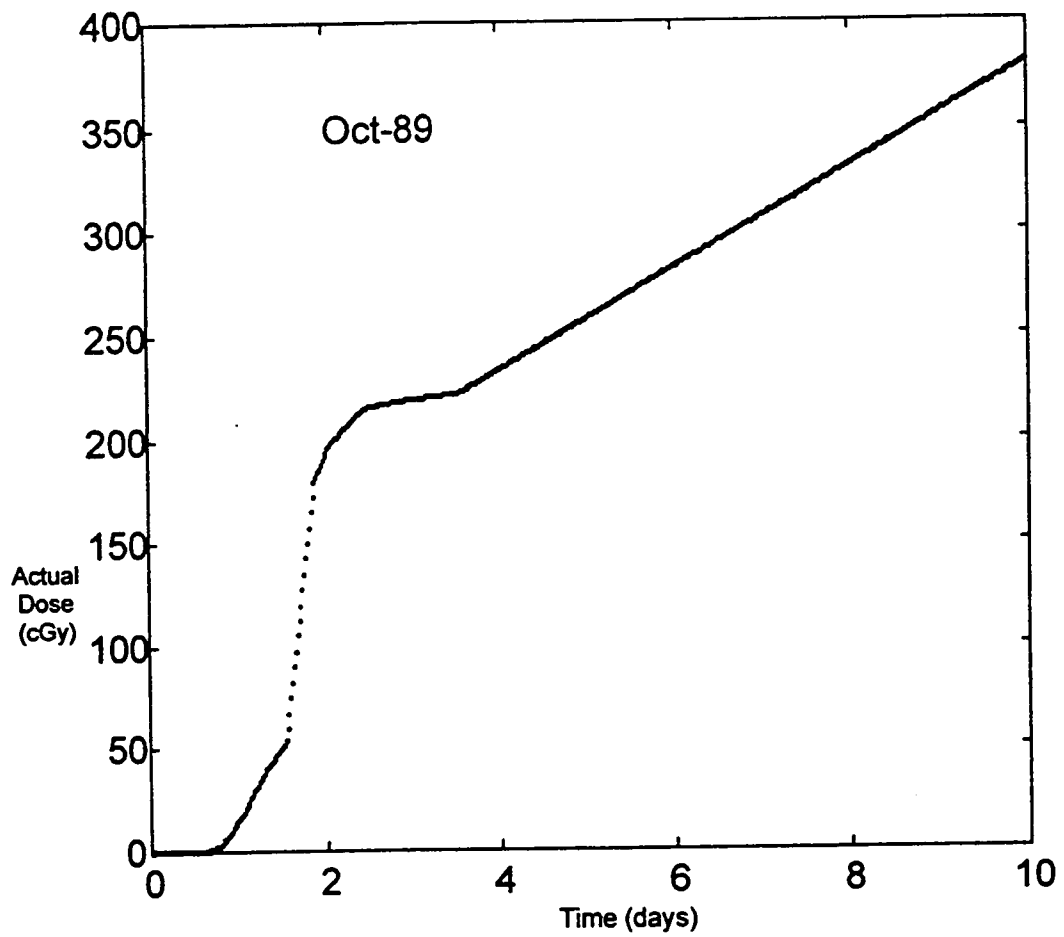


Figure A-5. October 89 event Dose (cGy) vs. Time (days) .

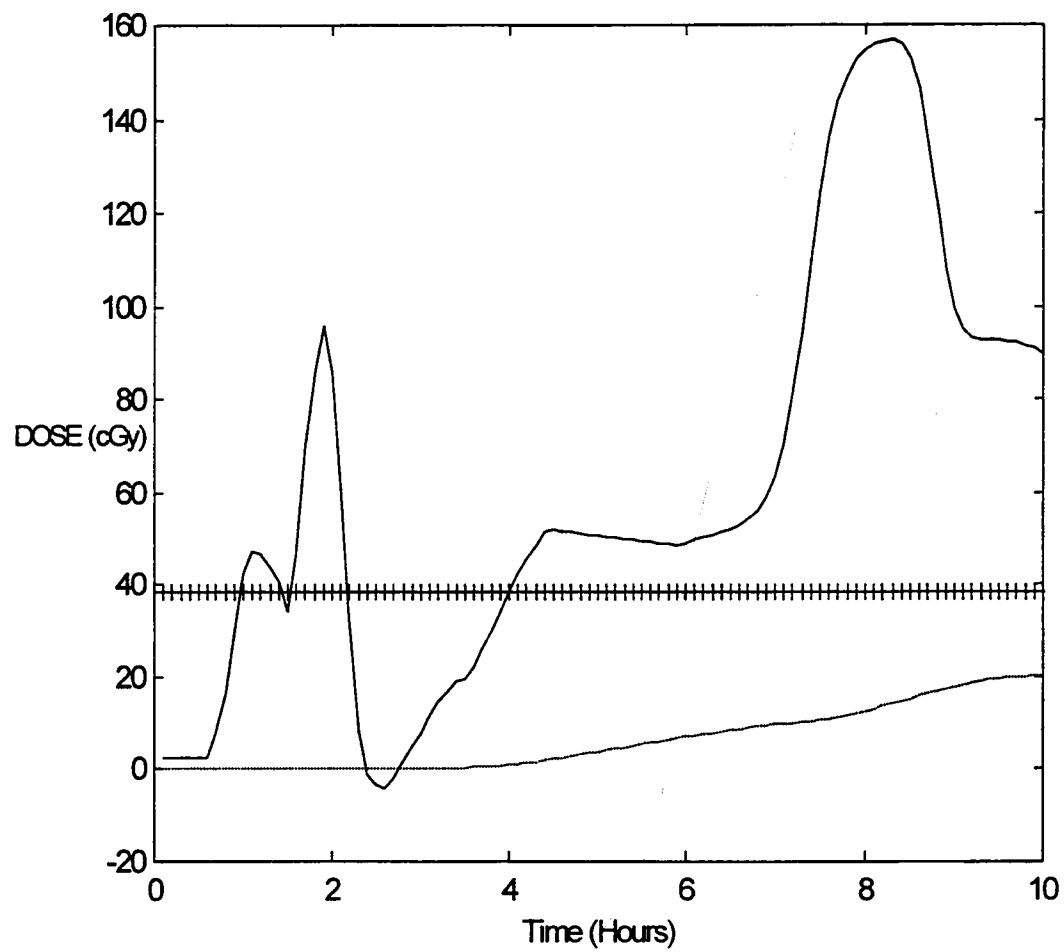


Figure A-6. November 89 event Dose (cGy) and Time (days).

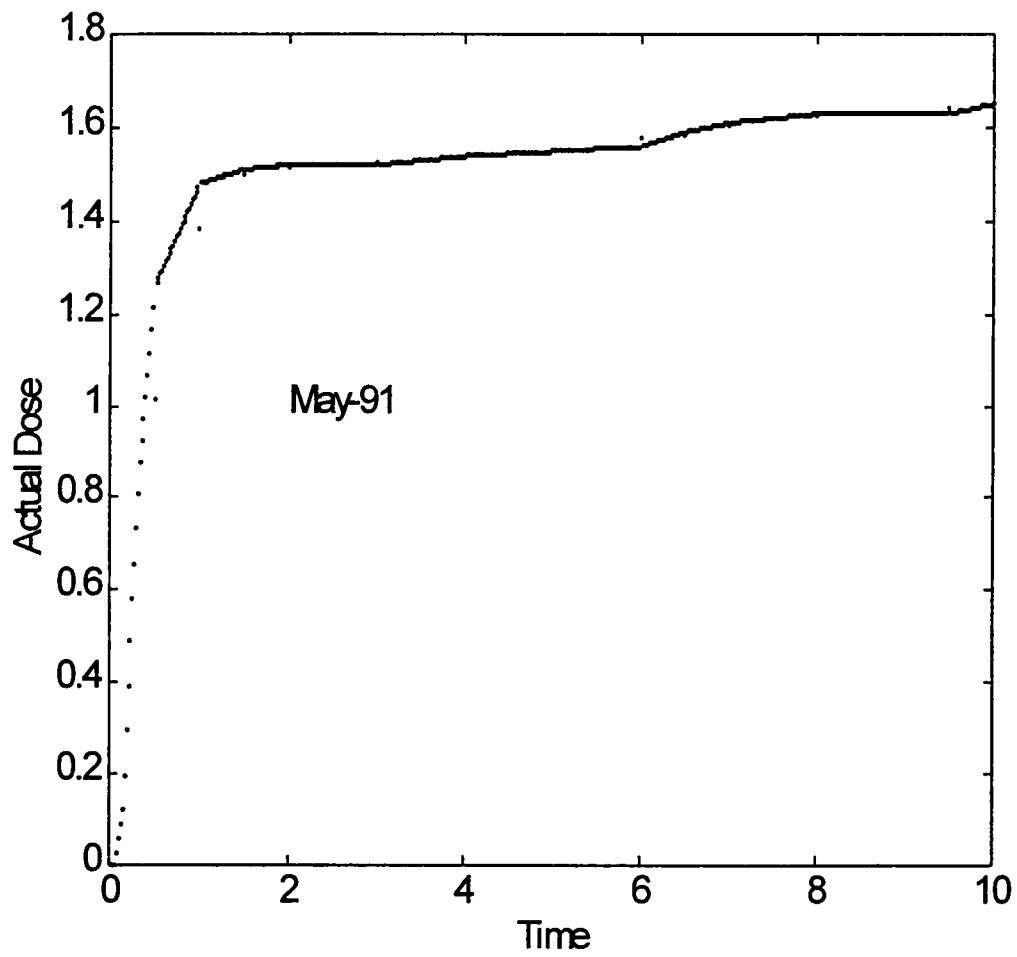


Figure A-7. May 91 event Dose (cGy) and Time (days).

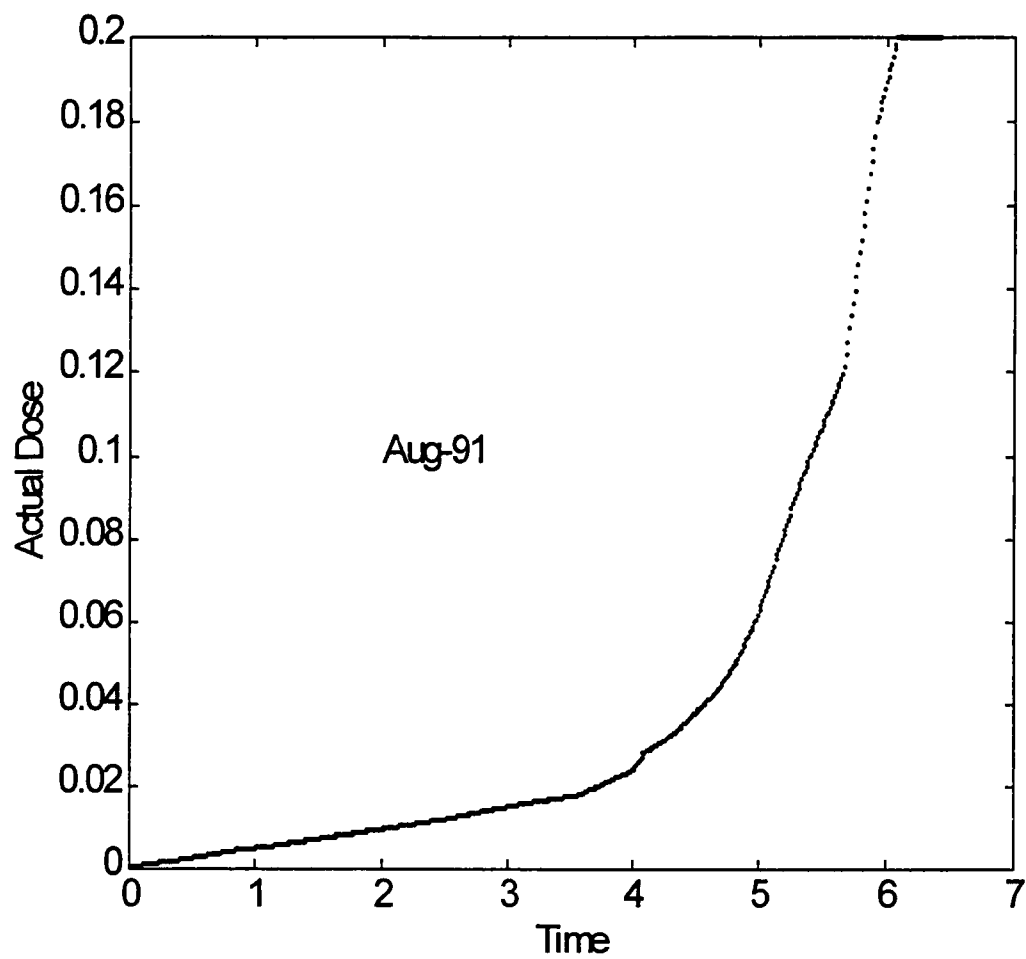


Figure A-8. August 91 event Dose (cGy) and Time (days).

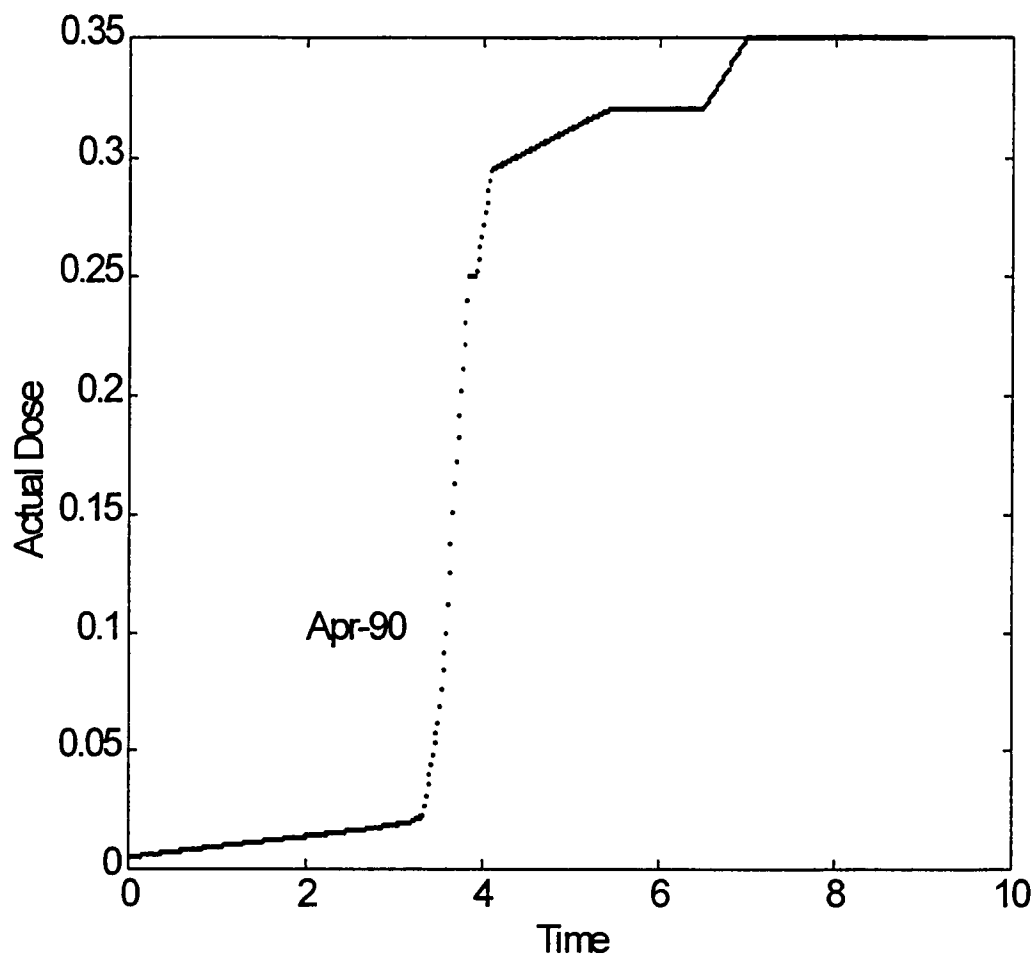


Figure A-9. April 90 event Dose (cGy) and Time (days).

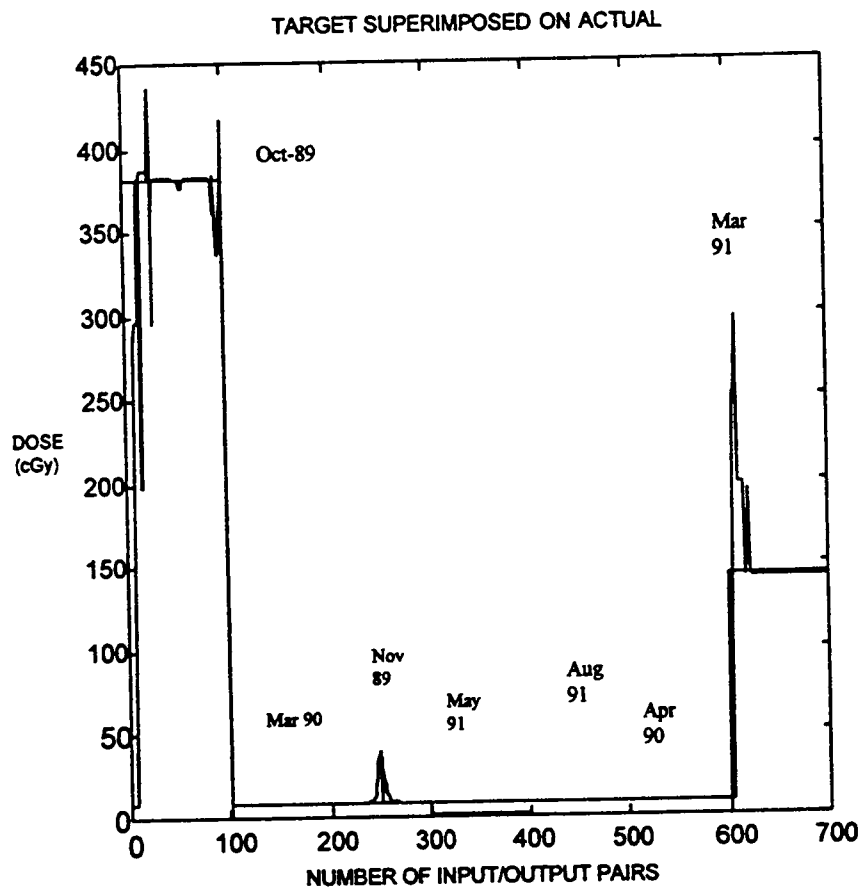


Figure A-10. Train using actual data, test using the actual data (For skin only)

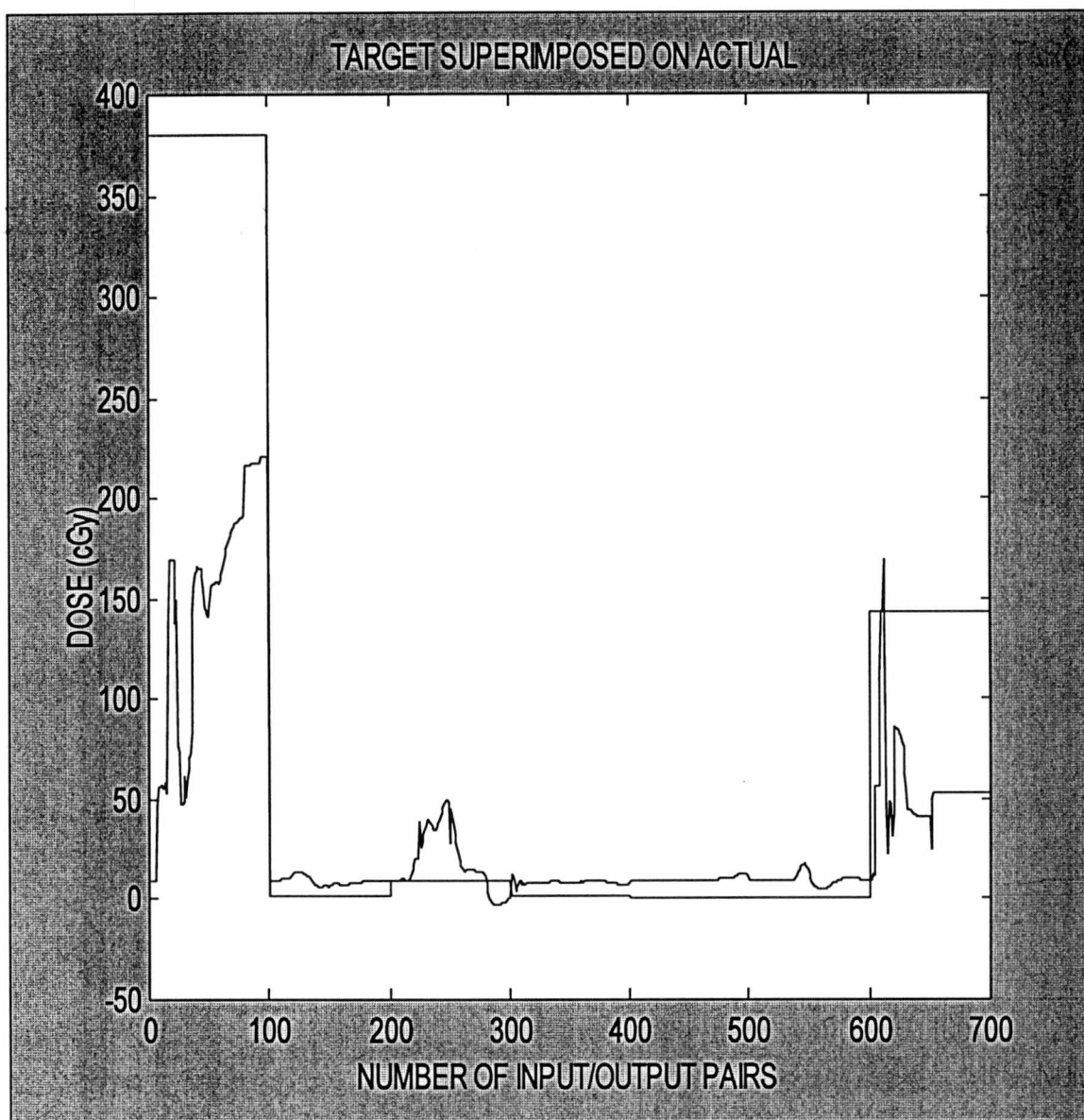


Figure A-11. Test with actual doses with weights from parametric using weights from actual as initial values for events October 89, March 90, November 89, May 91, and August 91 (Skin only).

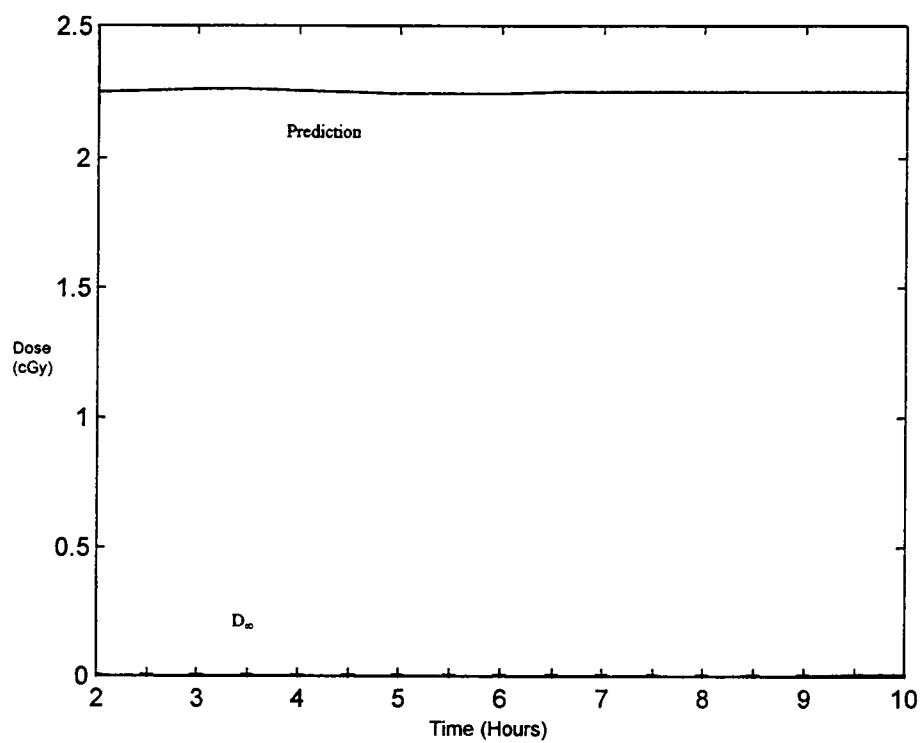


Figure A-12. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 1 (April 89) in Table 6

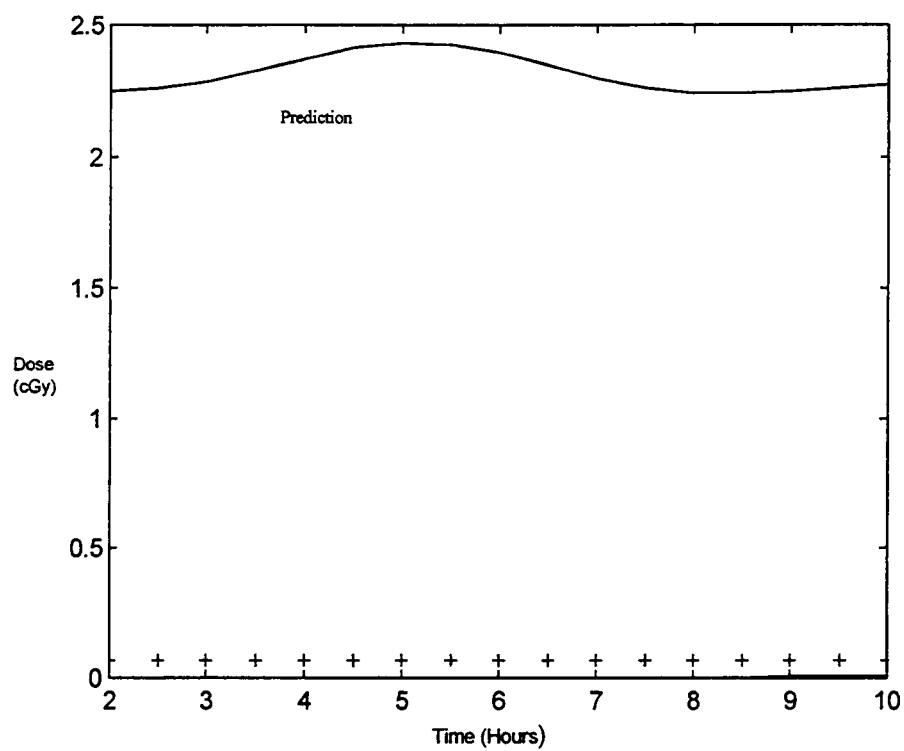


Figure A-13. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 2 (March 89) in Table 6

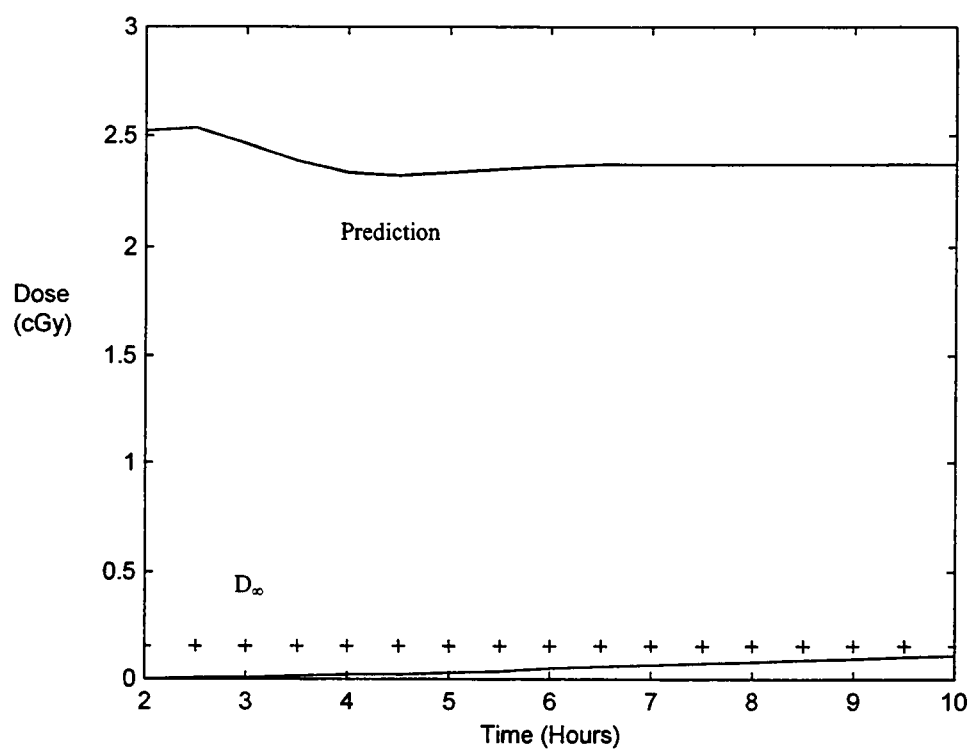


Figure A-14. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 3 (April 89) in Table 6

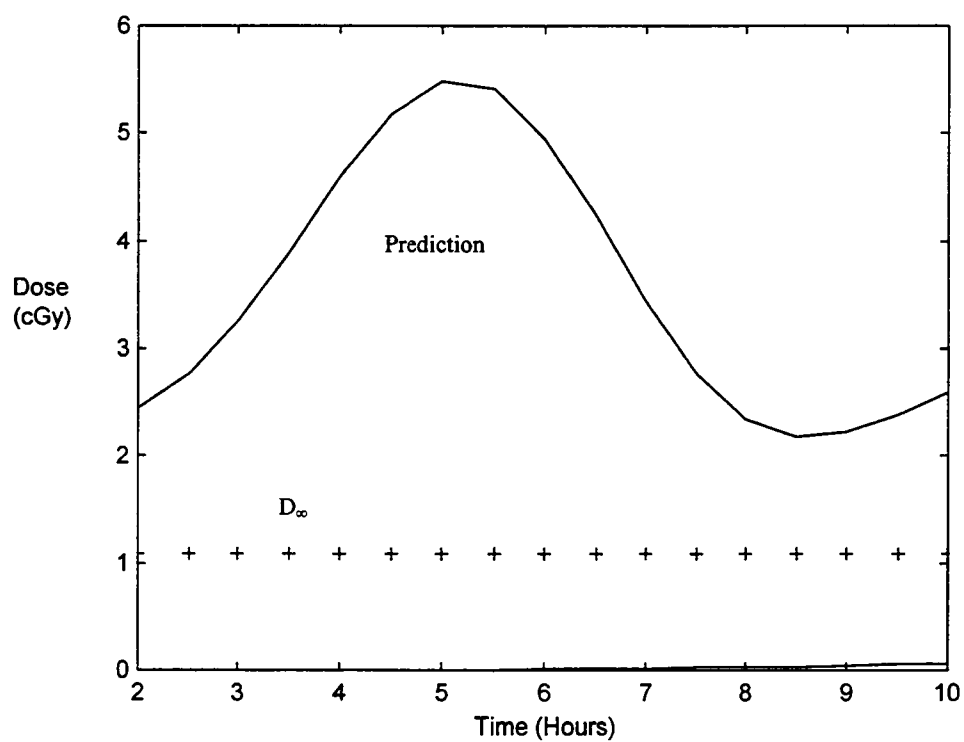


Figure A-15. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 4 (March 89) in Table 6

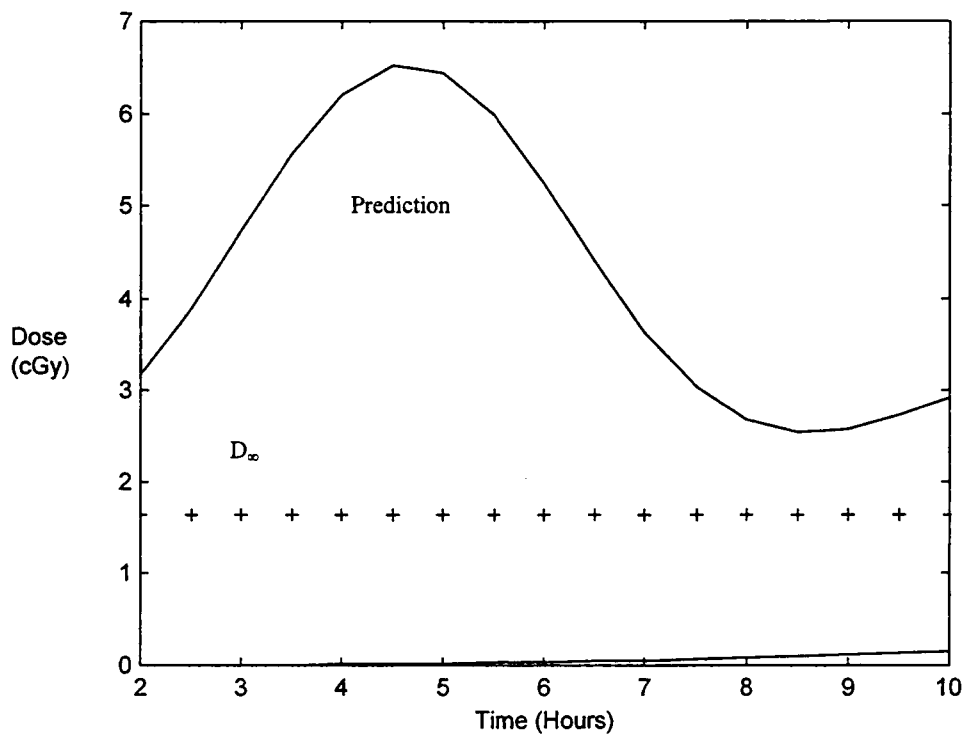


Figure A-16. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 5 (March 89) in Table 6

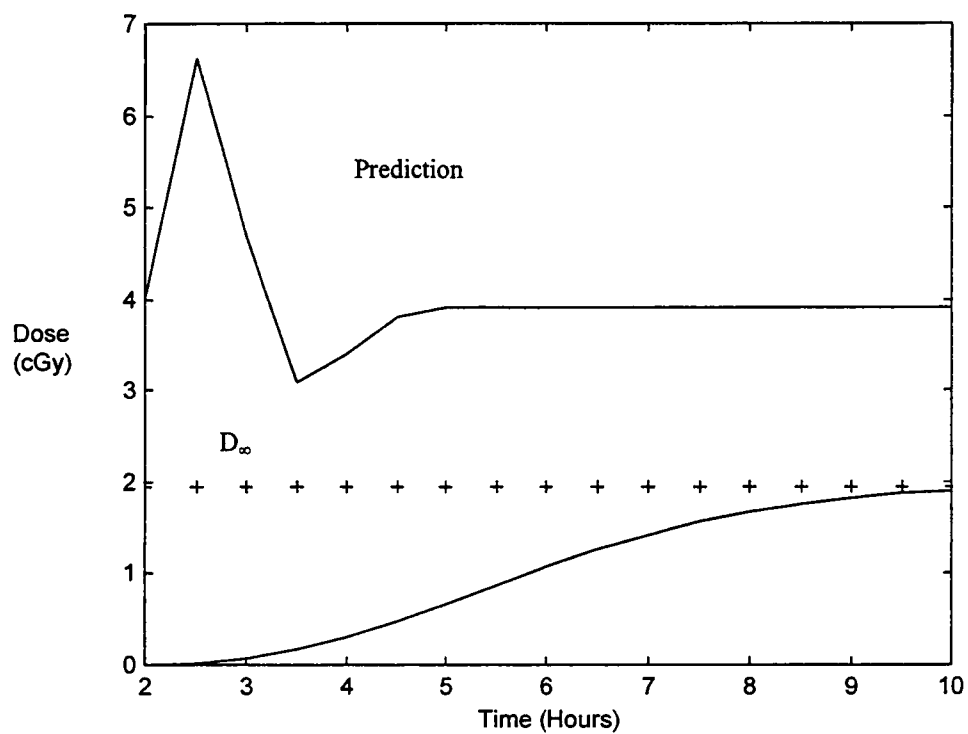


Figure A-17. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 7 (June 91) in Table 6

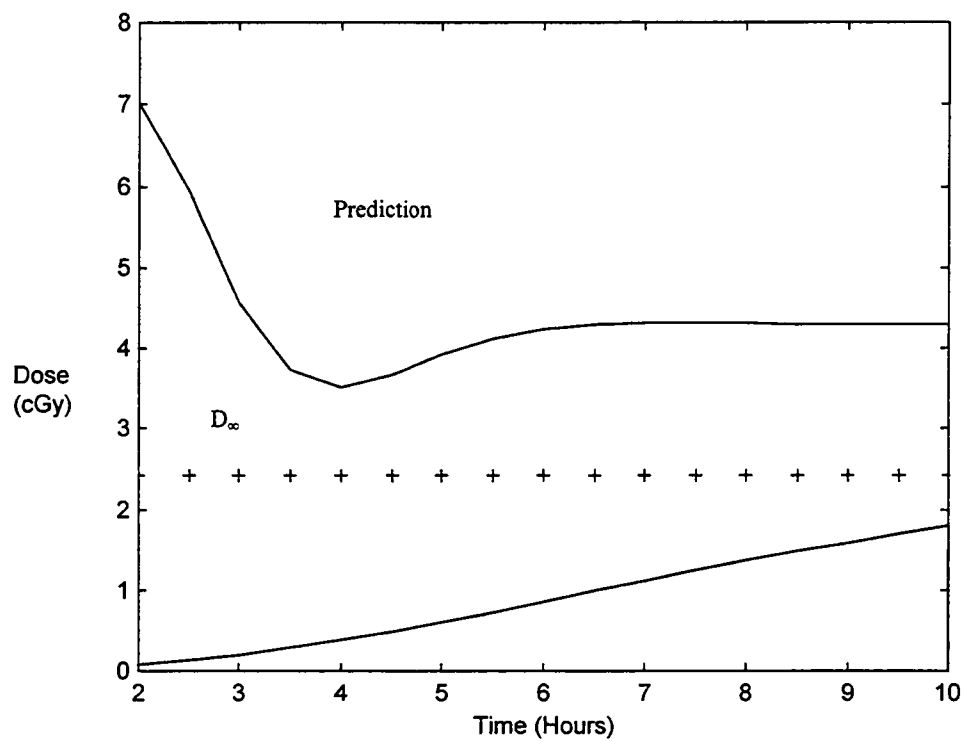


Figure A-18. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 8 (April 89) in Table 6

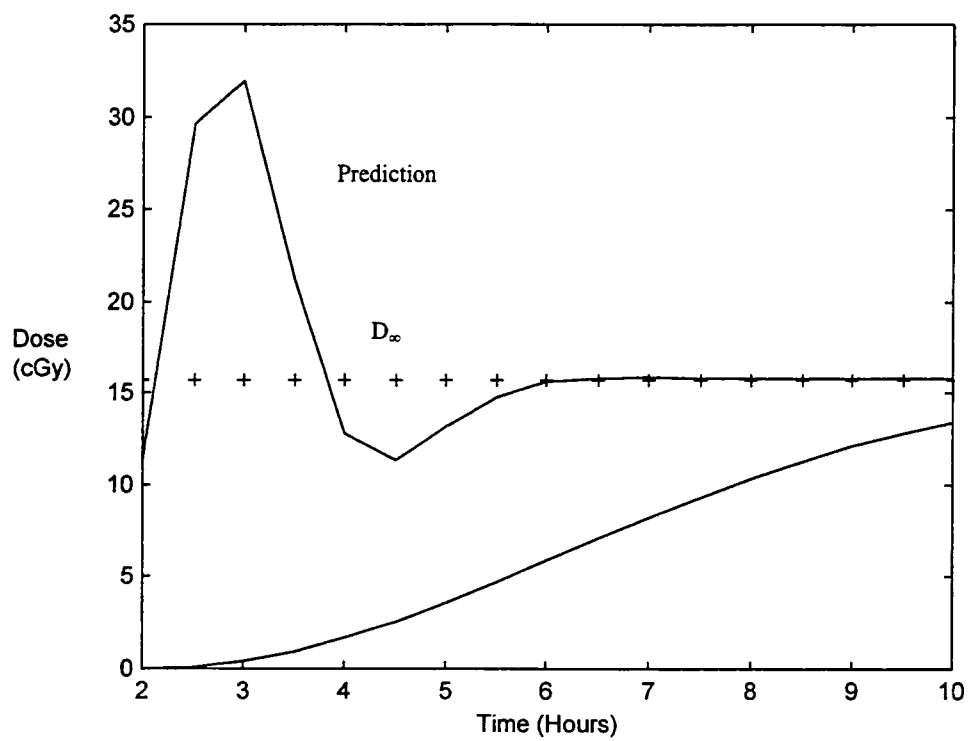


Figure A-19. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 9 (September 89) in Table 6

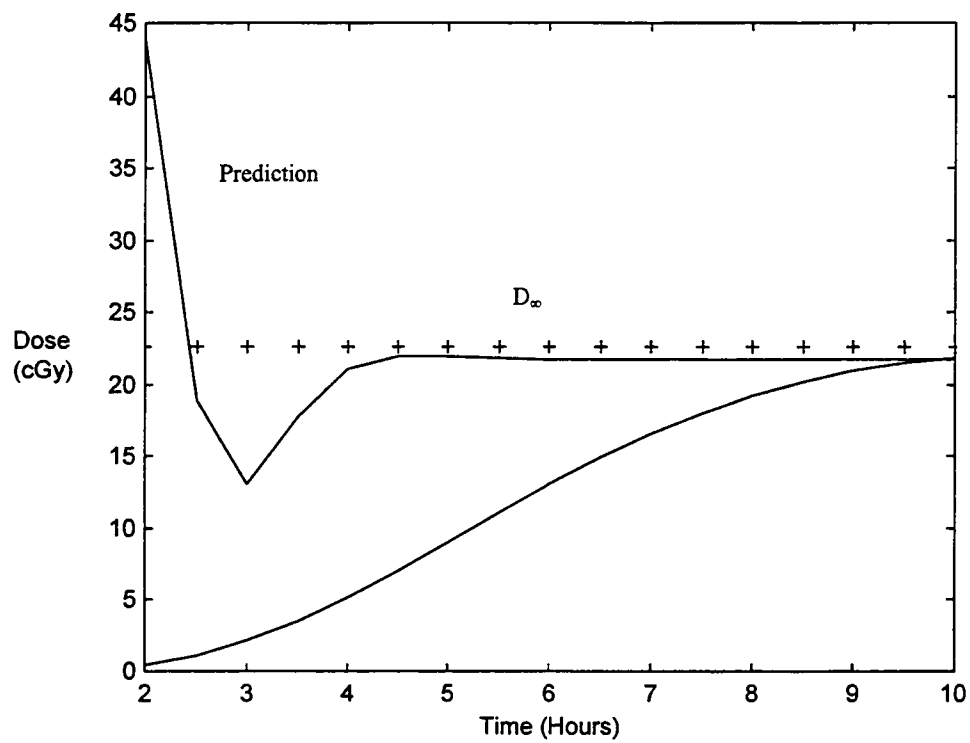


Figure A-20. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 11 (June 91) in Table 6

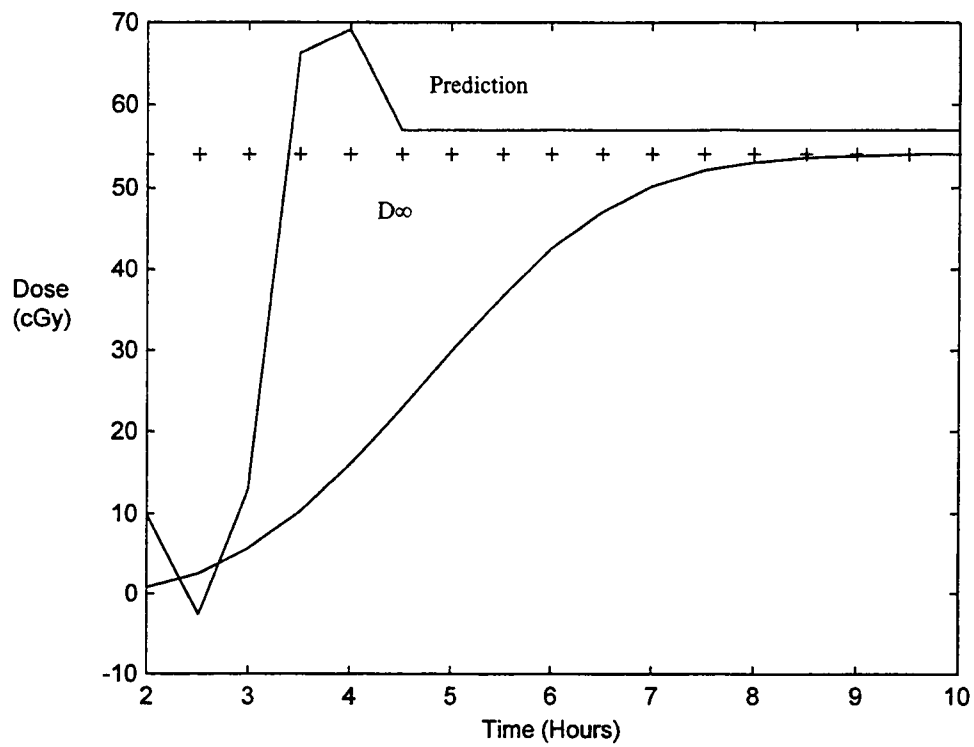


Figure A-21. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 12 (August 89) in Table 6

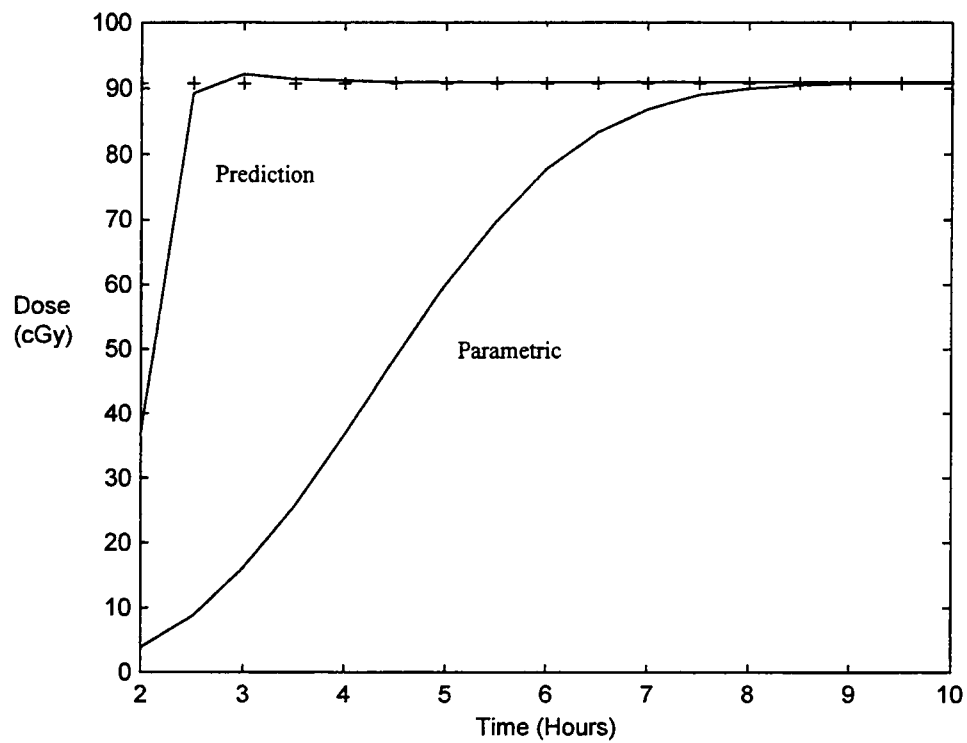


Figure A-22. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 13 (August 89) in Table 6

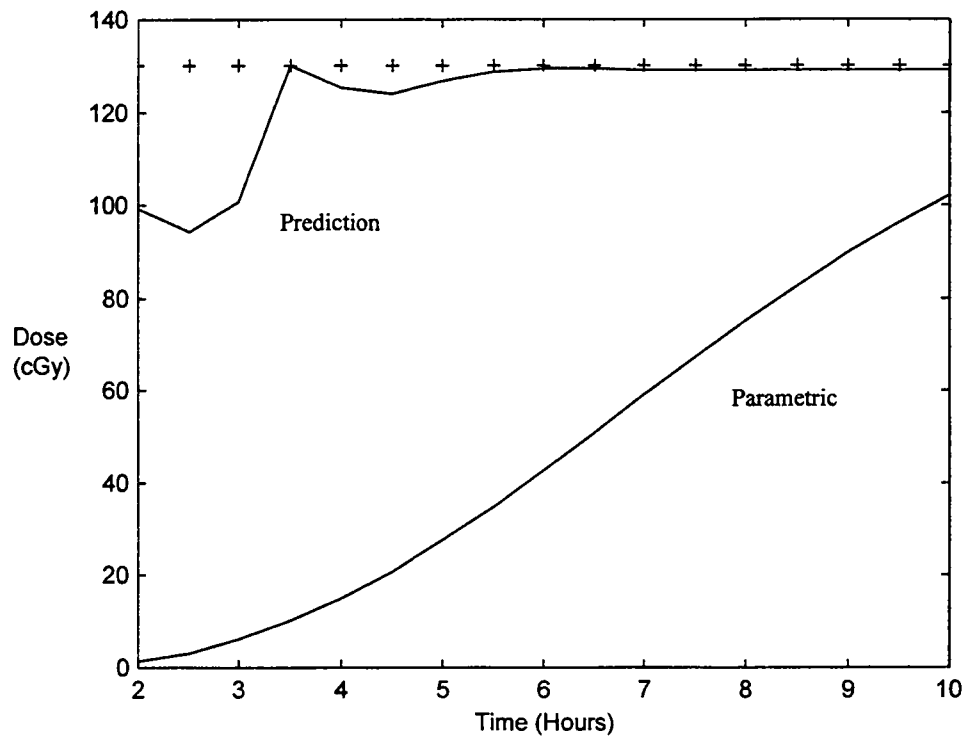


Figure A-23. Illustration of parametric data, asymptotic dose (+ symbol) and the prediction from STDNN for event 15 (September 89) in Table 6

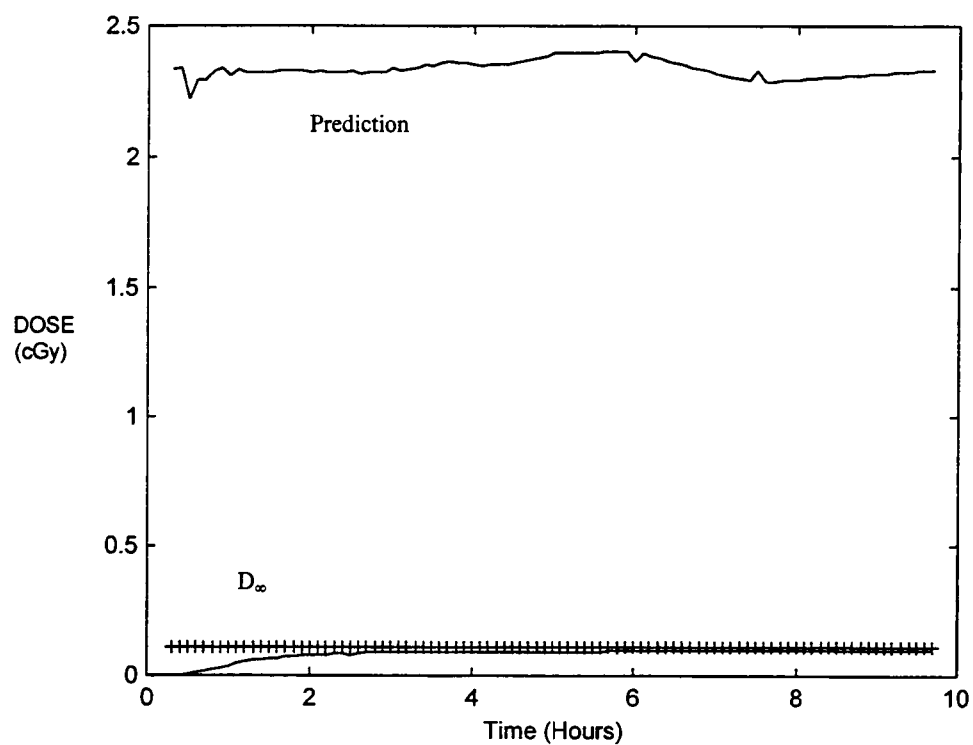


Figure A-24. Illustration of actual data, asymptotic dose (+ symbol) and the prediction from STDNN for event 5 (October 89) in Table 7

Appendix B

Oct-89	Days	Skin Dose	Days	Eye Dose	Days	BFO Dose
	0.627083	0.108133	0.627083	0.100781	0.627083	0.0496
	0.70417	1.01208	0.70417	0.90705	0.70417	0.335538
	0.79375	3.20229	0.79375	2.819334	0.79375	0.924668
	0.877083	7.081388	0.877083	6.183382	0.877083	1.920596
	1.04375	17.86957	1.04375	15.47196	1.04375	4.547567
	1.127083	24.42482	1.127083	20.97496	1.127083	5.862883
	1.210417	31.37366	1.210417	26.79229	1.210417	7.207263
	1.377083	43.50455	1.377083	36.86057	1.377083	9.432728
	1.54375	53.61958	1.54375	45.11669	1.54375	11.05142
	1.710417	116.5384	1.710417	92.32921	1.710417	15.6162
	1.877083	179.2399	1.877083	137.7565	1.877083	19.44064
	2.04375	196.5419	2.04375	150.3971	2.04375	20.68174
	2.29375	208.2598	2.29375	158.7104	2.29375	21.2992
	2.460417	215.032	2.460417	163.7184	2.460417	21.85006
	2.79375	217.9803	2.79375	165.8122	2.79375	22.01067
	3.54375	222.0246	3.54375	168.6765	3.54375	22.22476
	10	380.387	10	289.3685	10	38.42597

Days	Skin	Fit	Eye	Fit	BFO	Fit	Mar-90
0.083333	0.0016	0.001214	0.0014	0.001138	0.00047	2.93620E	
						-	
0.25	0.0084	0.029698	0.0067	0.021396	0.0011	0.001651	
0.333333	0.047	0.067699	0.032	0.045472	0.0023	0.002548	
0.5	0.18	0.207515	0.11	0.126426	0.0043	0.004557	
0.666666	0.46	0.427326	0.26	0.244736	0.0068	0.006647	
0.833333	0.75	0.684987	0.42	0.379100	0.009	0.008646	
1	0.96	0.917039	0.53	0.502029	0.011	0.010445	
1.25	1.08	1.127725	0.59	0.625291	0.012	0.012659	
1.5	1.15	1.193804	0.64	0.675472	0.013	0.014275	
3	1.2	1.205	0.68	0.69	0.016	0.016936	

Table B-1. Actual data for skin, eye, and bone marrow from the transport code calculations

Nov-89						
Hrs	Days	Skin Dose	Days	Eye Dose	Days	BFO Dose
2	0.083333	0.002080	0.083333	0.001389	0.083333	8.66346E -
4	0.166666	0.013355	0.166666	0.007766	0.166666	2.19843E -
8	0.333333	0.172119	0.333333	0.094090	0.333333	0.001865
12	0.5	0.900427	0.5	0.523327	0.5	0.014761
18	0.75	2.907275	0.75	1.611573	0.75	0.034631
24	1	5.994321	1	3.204753	1	0.055790
32	1.333333	8.206882	1.333333	4.316268	1.333333	0.068221
48	2	8.838538	2	4.62749	2	7.12E-02

May-91	Days	skin	fit	eye	fit	bfo	fit
	0.083333	5.50E-04	0	4.88E-04	0	1.75E-04	0
	0.166666	0.13	0.375576	0.11	0.303047	0.025	0.049587
	0.25	0.54	0.610526	0.42	0.465751	0.058	0.062180
	0.333333	0.86	0.825212	0.63	0.607968	0.072	0.071213
	0.5	1.26	1.153212	0.91	0.820144	0.088	0.082772
	1	1.48	1.497649	1.05	1.056972	0.094	0.095371
	1.5	1.51	1.528565	1.07	1.087250	0.096	0.098104
	2	1.52	1.529962	1.08	1.089828	0.097	0.098765
	2.5	1.52	1.529999	1.08	1.089991	0.097	0.098935
	3	1.52	1.529999	1.08	1.089999	0.097	0.098981
	3.5	1.53	1.53	1.09	1.089999	0.099	0.098994
	4.083333	1.54	1.53	1.1	1.09	0.1	0.098998
	6	1.56	1.560691	1.12	1.119143	0.11	0.109999
	6.5	1.59	1.587368	1.14	1.142607	0.11	0.109999
	7	1.61	1.611649	1.16	1.158892	0.11	0.109999
	7.5	1.62	1.629594	1.16	1.166644	0.11	0.11
	8	1.63	1.640740	1.17	1.169250	0.11	0.11
	8.5	1.63	1.646661	1.17	1.169877	0.11	0.11
	9	1.63	1.649379	1.17	1.169985	0.11	0.11
	9.5	1.63	1.650464	1.17	1.169998	0.11	0.11
	10	1.65	1.650843	1.17	1.169999	0.11	0.11

Table B-1. (continued)

Aug-91	Days	skin	fit	eye	fit	bfo	fit
	0	0.00039	0	0.00036	0	0.00015	0
	0.083333	0.00079	1.94E-14	0.00072	1.34E-14	0.00029	1.21E-15
	0.25	0.0015	4.95E-11	0.0014	3.42E-11	0.00058	3.09E-12
	0.333333	0.0018	3.86E-10	0.0017	2.67E-10	0.00074	2.41E-11
	0.5	0.0026	6.98E-09	0.0024	4.82E-09	0.001	4.36E-10
	0.583333	0.003	2.1E-08	0.0028	1.45E-08	0.0012	1.31E-09
	0.666667	0.0035	5.45E-08	0.0032	3.76E-08	0.0014	3.4E-09
	0.833333	0.0043	2.68E-07	0.0039	1.85E-07	0.0016	1.67E-08
	1.166667	0.0056	2.96E-06	0.0051	2.05E-06	0.0021	1.85E-07
	1.25	0.006	4.85E-06	0.0054	3.35E-06	0.0022	3.02E-07
	2.5	0.012	0.000683	0.011	0.000472	0.0046	4.26E-05
	3	0.015	0.0025	0.014	0.001726	0.0056	0.000156
	3.583333	0.018	0.008756	0.016	0.006046	0.0066	0.000546
	4	0.024	0.018729	0.022	0.012932	0.0074	0.001168
	4.083333	0.028	0.021542	0.025	0.014874	0.0078	0.001344
	4.333333	0.033	0.03202	0.029	0.022109	0.008	0.001997
	4.5	0.038	0.040898	0.032	0.028239	0.0083	0.002551
	4.666667	0.043	0.051418	0.036	0.035503	0.0086	0.003207
	4.833333	0.051	0.063605	0.042	0.043918	0.009	0.003968
	5	0.062	0.07738	0.05	0.053429	0.0094	0.004827
	5.333333	0.094	0.108666	0.073	0.075031	0.011	0.006779
	5.666667	0.12	0.141715	0.093	0.097851	0.011	0.00884
	5.916667	0.18	0.164485	0.13	0.113573	0.012	0.010261
	6.083333	0.2	0.177457	0.14	0.12253	0.013	0.01107
	6.416667	0.2	0.19629	0.14	0.135534	0.013	0.012245

Table B-1. (continued)

Apr-90	days	skin	fit	eye	fit	bfo	fit
	0	0.005	0	0.005	0	0	0
	2.666667	0.0168	0.000939	0.015	0.000916	0.005	0.004221
	3.166667	0.0196	0.014808	0.017	0.012373	0.006	0.005793
	3.25	0.021	0.022268	0.018	0.018148	0.0061	0.006055
	3.333333	0.023	0.032944	0.02	0.026186	0.0063	0.006316
	3.5	0.064	0.068095	0.048	0.051474	0.0067	0.006828
	3.583333	0.1	0.094541	0.073	0.069693	0.0069	0.007079
	3.666667	0.16	0.127477	0.11	0.091678	0.007	0.007326
	3.833333	0.25	0.207394	0.16	0.142548	0.008	0.007804
	3.916667	0.251	0.247261	0.165	0.166866	0.0081	0.008035
	4.083333	0.295	0.303097	0.19	0.200079	0.0086	0.008479
	5.416667	0.32	0.321	0.2	0.211	0.01	0.010958
	5.5	0.32	0.321	0.21	0.211	0.01	0.01105
	6	0.32	0.321	0.21	0.211	0.011	0.011479
	6.5	0.32	0.328561	0.22	0.221016	0.012	0.013211
	6.666667	0.33	0.336122	0.22	0.223079	0.013	0.013826
	7	0.35	0.348856	0.23	0.226409	0.015	0.015077
	7.5	0.35	0.351988	0.23	0.230074	0.016	0.016688
	8	0.35	0.352	0.23	0.232706	0.016	0.017803
	8.5	0.35	0.352	0.24	0.234653	0.017	0.018465
	9	0.35	0.352	0.24	0.236122	0.018	0.018814

Mar-91	Days	skin	Fit	eye	Fit	bfo	Fit
	0.083333	2.60E-03	8.38E-05	2.10E-03	3.56E-05	4.30E-04	1.29E-06
	0.166667	4.40E-03	0.004219	0.0037	0.001941	0.00089	6.33E-05
	0.25	0.0066	0.041753	0.0054	0.020126	0.0012	0.000617
	0.416667	0.41	0.747973	0.25	0.382488	0.0097	0.010835
	0.5	1.774	2.086956	1.05	1.090269	0.032	0.030036
	0.583333	5.6	4.938669	3.23	2.628014	0.087	0.070742
	0.666667	10.74	10.30363	6.11	5.57325	0.15	0.147178
	0.75	18.79	19.37193	10.69	10.63452	0.27	0.276626
	0.916667	50.75	52.09513	29.18	29.33396	0.77	0.751627
	1.083333	98.48	98.52962	57.22	56.46962	1.61	1.465005
	1.25	133.6	132.9601	76.83	76.68934	2.04	2.065587
	1.666667	137.87	143.5997	80.1	82.69991	2.26	2.309971
	3	139.9	143.6	81.5	82.7	2.28	2.31
	3.333333	140.8	143.6	81.8	82.7	2.29	2.31
	3.583333	141.1	143.6	82	82.7	2.29	2.31
	10	143.5	143.6	82.6	82.7	2.3	2.31

Table B-1. (continued)

	Dose Infinity	Error	Actual Error
Event	Dmax	ΔD_{max}	$D_{max} \cdot \Delta D_{max}$
1	0.0082	-273.1069	-2.2395
1	0.0082	-274.1002	-2.2476
1	0.0082	-274.9083	-2.2542
1	0.0082	-274.8769	-2.254
1	0.0082	-274.0583	-2.2473
1	0.0082	-273.1466	-2.2398
1	0.0082	-272.6998	-2.2361
1	0.0082	-272.7264	-2.2364
1	0.0082	-272.9475	-2.2382
1	0.0082	-273.1479	-2.2398
1	0.0082	-273.263	-2.2408
1	0.0082	-273.3102	-2.2411
1	0.0082	-273.3218	-2.2412
1	0.0082	-273.32	-2.2412
1	0.0082	-273.3153	-2.2412
1	0.0082	-273.3112	-2.2412
1	0.0082	-273.3085	-2.2411

	Dose Infinity	Error	Actual Error
Event	Dmax	ΔD_{max}	$D_{max} \cdot \Delta D_{max}$
2	0.066	-33.0565	-2.1817
2	0.066	-33.2737	-2.1961
2	0.066	-33.6841	-2.2232
2	0.066	-34.2775	-2.2623
2	0.066	-34.9576	-2.3072
2	0.066	-35.55	-2.3463
2	0.066	-35.8591	-2.3667
2	0.066	-35.7585	-2.3601
2	0.066	-35.2642	-2.3274
2	0.066	-34.5379	-2.2795
2	0.066	-33.8098	-2.2314
2	0.066	-33.2709	-2.1959
2	0.066	-33.0031	-2.1782
2	0.066	-32.9789	-2.1766
2	0.066	-33.1104	-2.1853
2	0.066	-33.3036	-2.198
2	0.066	-33.4913	-2.2104

Table B-2. Error calculations for 15 parametric events

	Dose Infinity	Error	Actual Error
Event	Dmax	Δ Dmax	Dmax* Δ Dmax
3	0.151	-15.7323	-2.3756
3	0.151	-15.7996	-2.3857
3	0.151	-15.3407	-2.3164
3	0.151	-14.7901	-2.2333
3	0.151	-14.4581	-2.1832
3	0.151	-14.3826	-2.1718
3	0.151	-14.452	-2.1822
3	0.151	-14.5538	-2.1976
3	0.151	-14.6323	-2.2095
3	0.151	-14.6773	-2.2163
3	0.151	-14.6977	-2.2193
3	0.151	-14.7045	-2.2204
3	0.151	-14.7053	-2.2205
3	0.151	-14.7043	-2.2204
3	0.151	-14.7031	-2.2202
3	0.151	-14.7021	-2.22
3	0.151	-14.7016	-2.2199

	Dose Infinity	Error	Actual Error
Event	Dmax	Δ Dmax	Dmax* Δ Dmax
4	1.09	-1.2464	-1.3586
4	1.09	-1.5316	-1.6695
4	1.09	-1.9832	-2.1617
4	1.09	-2.5731	-2.8046
4	1.09	-3.21	-3.4989
4	1.09	-3.75	-4.0875
4	1.09	-4.0377	-4.4011
4	1.09	-3.9692	-4.3264
4	1.09	-3.5445	-3.8635
4	1.09	-2.8773	-3.1362
4	1.09	-2.1497	-2.3432
4	1.09	-1.5374	-1.6758
4	1.09	-1.1469	-1.2501
4	1.09	-0.9962	-1.0859
4	1.09	-1.036	-1.1292
4	1.09	-1.1881	-1.295
4	1.09	-1.3799	-1.5041

Table B-2. (continued)

	Dose Infinity	Error	Actual Error
Event	Dmax	ΔD_{max}	$D_{max} * \Delta D_{max}$
5	1.64	-0.941	-1.5432
5	1.64	-1.3771	-2.2585
5	1.64	-1.8878	-3.0961
5	1.64	-2.3896	-3.9189
5	1.64	-2.7821	-4.5627
5	1.64	-2.9773	-4.8827
5	1.64	-2.9281	-4.8021
5	1.64	-2.6466	-4.3404
5	1.64	-2.2001	-3.6082
5	1.64	-1.6892	-2.7703
5	1.64	-1.2147	-1.9921
5	1.64	-0.8503	-1.3945
5	1.64	-0.6293	-1.032
5	1.64	-0.5464	-0.896
5	1.64	-0.5709	-0.9363
5	1.64	-0.6624	-1.0864
5	1.64	-0.7831	-1.2843

	Dose Infinity	Error	Actual Error
Event	Dmax	ΔD_{max}	$D_{max} * \Delta D_{max}$
6	1.73	-2.7084	-4.6854
6	1.73	-0.7001	-1.2113
6	1.73	-0.7981	-1.3808
6	1.73	-1.1315	-1.9574
6	1.73	-1.1608	-2.0081
6	1.73	-1.1512	-1.9915
6	1.73	-1.1478	-1.9857
6	1.73	-1.1476	-1.9853
6	1.73	-1.148	-1.986
6	1.73	-1.1483	-1.9866
6	1.73	-1.1486	-1.9871
6	1.73	-1.1488	-1.9874
6	1.73	-1.1489	-1.9876
6	1.73	-1.149	-1.9877
6	1.73	-1.149	-1.9877
6	1.73	-1.149	-1.9878
6	1.73	-1.149	-1.9878

Table B-2. (continued)

	Dose Infinity	Error	Actual Error
Event	Dmax	ΔD_{max}	$D_{max} * \Delta D_{max}$
7	1.95	-1.0446	-2.0369
7	1.95	-2.3997	-4.6793
7	1.95	-1.4045	-2.7389
7	1.95	-0.582	-1.1348
7	1.95	-0.7449	-1.4525
7	1.95	-0.9531	-1.8585
7	1.95	-1.0069	-1.9635
7	1.95	-1.0081	-1.9657
7	1.95	-1.004	-1.9578
7	1.95	-1.0022	-1.9543
7	1.95	-1.0019	-1.9536
7	1.95	-1.002	-1.9539
7	1.95	-1.0022	-1.9543
7	1.95	-1.0024	-1.9547
7	1.95	-1.0026	-1.955
7	1.95	-1.0027	-1.9553
7	1.95	-1.0028	-1.9554

	Dose Infinity	Error	Actual Error
Event	Dmax	ΔD_{max}	$D_{max} * \Delta D_{max}$
8	2.41	-1.911	-4.6055
8	2.41	-1.4602	-3.5191
8	2.41	-0.8953	-2.1576
8	2.41	-0.544	-1.3112
8	2.41	-0.4572	-1.1019
8	2.41	-0.5246	-1.2643
8	2.41	-0.6292	-1.5163
8	2.41	-0.711	-1.7135
8	2.41	-0.7581	-1.8271
8	2.41	-0.7795	-1.8785
8	2.41	-0.7865	-1.8955
8	2.41	-0.7874	-1.8976
8	2.41	-0.7863	-1.8949
8	2.41	-0.7849	-1.8917
8	2.41	-0.7839	-1.8893
8	2.41	-0.7833	-1.8878
8	2.41	-0.783	-1.887

Table B-2. (continued)

	Dose Infinity	Error	Actual Error
Event	Dmax	ΔD_{max}	$D_{max} \cdot \Delta D_{max}$
9	15.75	0.2784	4.3848
9	15.75	-0.8786	-13.8385
9	15.75	-1.0294	-16.2132
9	15.75	-0.3426	-5.3964
9	15.75	0.1884	2.9676
9	15.75	0.2775	4.3701
9	15.75	0.1661	2.6168
9	15.75	0.0604	0.9509
9	15.75	0.01	0.1575
9	15.75	-0.0048	-0.0754
9	15.75	-0.0061	-0.0968
9	15.75	-0.0045	-0.0716
9	15.75	-0.0032	-0.0496
9	15.75	-0.0024	-0.0381
9	15.75	-0.0022	-0.0339
9	15.75	-0.0021	-0.0335
9	15.75	-0.0022	-0.0347

	Dose Infinity	Error	Actual Error
Event	Dmax	ΔD_{max}	$D_{max} \cdot \Delta D_{max}$
10	16.75	-0.8129	-13.6163
10	16.75	-0.5703	-9.5526
10	16.75	-0.0836	-1.3999
10	16.75	0.1491	2.4975
10	16.75	0.158	2.6462
10	16.75	0.0951	1.5928
10	16.75	0.0429	0.7181
10	16.75	0.0164	0.2748
10	16.75	0.0068	0.1134
10	16.75	0.0045	0.0753
10	16.75	0.0046	0.0763
10	16.75	0.005	0.0839
10	16.75	0.0053	0.0894
10	16.75	0.0055	0.0919
10	16.75	0.0055	0.0925
10	16.75	0.0055	0.0921
10	16.75	0.0054	0.0913

Table B-2. (continued)

	Dose Infinity	Error	Actual Error
Event	Dmax	ΔD_{max}	$D_{max} \cdot \Delta D_{max}$
11	22.6	-0.9421	-21.292
11	22.6	0.1638	3.7022
11	22.6	0.424	9.5827
11	22.6	0.2153	4.8654
11	22.6	0.0668	1.5093
11	22.6	0.0301	0.6795
11	22.6	0.0304	0.6875
11	22.6	0.0347	0.7833
11	22.6	0.0369	0.8335
11	22.6	0.0376	0.8493
11	22.6	0.0376	0.8504
11	22.6	0.0375	0.8469
11	22.6	0.0373	0.8428
11	22.6	0.0371	0.8391
11	22.6	0.037	0.8363
11	22.6	0.0369	0.8342
11	22.6	0.0368	0.8326

	Dose Infinity	Error	Actual Error
Event	Dmax	ΔD_{max}	$D_{max} \cdot \Delta D_{max}$
12	54	0.8154	44.0305
12	54	1.0462	56.4932
12	54	0.7583	40.9488
12	54	-0.2283	-12.33
12	54	-0.2831	-15.2877
12	54	-0.0546	-2.9504
12	54	-0.054	-2.9134
12	54	-0.0541	-2.9234
12	54	-0.0544	-2.937
12	54	-0.0546	-2.948
12	54	-0.0547	-2.9556
12	54	-0.0548	-2.9604
12	54	-0.0549	-2.9631
12	54	-0.0549	-2.9644
12	54	-0.0549	-2.965
12	54	-0.0549	-2.9652
12	54	-0.0549	-2.9653

Table B-2. (continued)

	Dose Infinity	Error	Actual Error
Event	Dmax	ΔD_{max}	$D_{max} \cdot \Delta D_{max}$
13	90.8	0.5946	53.9877
13	90.8	0.0161	1.4628
13	90.8	-0.0163	-1.4779
13	90.8	-0.0075	-0.6839
13	90.8	-0.0037	-0.3348
13	90.8	-0.0029	-0.2673
13	90.8	-0.0029	-0.2665
13	90.8	-0.003	-0.2736
13	90.8	-0.0031	-0.2799
13	90.8	-0.0031	-0.2844
13	90.8	-0.0032	-0.2873
13	90.8	-0.0032	-0.289
13	90.8	-0.0032	-0.2898
13	90.8	-0.0032	-0.2902
13	90.8	-0.0032	-0.2903
13	90.8	-0.0032	-0.2904
13	90.8	-0.0032	-0.2904

	Dose Infinity	Error	Actual Error
Event	Dmax	ΔD_{max}	$D_{max} \cdot \Delta D_{max}$
14	102	-0.0059	-0.6067
14	102	-0.5196	-53.0016
14	102	0.0288	2.9382
14	102	0.2786	28.4209
14	102	0.4625	47.1777
14	102	0.0831	8.4762
14	102	0.0768	7.838
14	102	0.0407	4.1504
14	102	0.0129	1.3107
14	102	0.0012	0.1239
14	102	-0.0013	-0.1304
14	102	-0.0006	-0.0597
14	102	0.0006	0.0628
14	102	0.0015	0.1561
14	102	0.0021	0.2122
14	102	0.0024	0.2418
14	102	0.0025	0.2557

Table B-2. (continued)

	Dose Infinity	Error	Actual Error
Event	Dmax	Δ Dmax	Dmax* Δ Dmax
15	130	0.235	30.5501
15	130	0.276	35.8769
15	130	0.2248	29.2269
15	130	-0.0001	-0.0127
15	130	0.0369	4.7992
15	130	0.0466	6.0521
15	130	0.0259	3.3611
15	130	0.0102	1.3241
15	130	0.005	0.6515
15	130	0.0048	0.6229
15	130	0.0059	0.7638
15	130	0.0069	0.8973
15	130	0.0076	0.9865
15	130	0.008	1.0386
15	130	0.0082	1.0673
15	130	0.0083	1.0832
15	130	0.0084	1.0925

Table B-2. (continued)

Weights

-0.0022	0.0965	-3.1946	10.0225	-6.9305
1.0886	-24.4740	-21.3766	-11.8763	-61.8130
37.7739	20.0551	62.3124	67.1041	-98.1198
-15.0653	-14.1033	-22.1166	-71.0725	11.5018
-34.6632	-20.9745	-3.1371	15.2156	-76.0191
-0.0022	0.0920	-3.0736	9.5935	-6.5684
-13.3283	-15.0105	2.5406	34.5562	26.0338
-2.4483	-31.0513	-9.7760	-2.9477	-62.1082
28.9387	78.6177	66.6708	51.7380	79.9603
0.0374	17.0459	6.0069	12.4305	16.8878

Biases

-0.3942
19.0809
-25.5074
5.9320
32.1310
-0.4962
30.1737
4.2151
-142.4059
-6.2329

Table B-3. Weights and biases from training the network with parametric data for events October 89, November 89, March 90, April 90, March 91, May 91 and August 91 for all three organs

Appendix C

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   actf.m
%   Use actual data for skin, eye and bone marrow,
%   interpolate them to the number of points
%   desired and save them in a data file to be used
%   for testing
%
%   Nazila Tehrani
%   copyright 1998
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% These data are from transport code calculations
% Enter the doses for skin bone marrow
a1=[0.627083
0.70417
0.79375
0.877083
1.04375
1.127083
1.210417
1.377083
1.54375
1.710417
1.877083
2.04375
2.29375
2.460417
2.79375
3.54375
10];

b1=[0.0496
0.335538
0.924668
1.920596
4.547567
5.862883
7.207263
9.432728
11.05142
15.6162
```

19.44064
20.68174
21.2992
21.85006
22.01067
22.22476
38.42597];

a2=[0.083333333
0.25
0.333333333
0.5
0.666666667
0.833333333
1
1.25
1.5
3];

b2=[0.00047
0.0011
0.0023
0.0043
0.0068
0.009
0.011
0.012
0.013
0.016];

a3=[0.083333333
0.166666667
0.333333333
0.5
0.75
1
1.333333333
2];

b3=[8.66346E-05
0.000219843
0.001865148
0.014761231
0.034631152
0.055790734
0.068221889
7.12E-02];

a4=[0.083333333
0.166666667

0.25
0.333333333
0.5
1
1.5
2
2.5
3
3.5
4.083333333
6
6.5
7
7.5
8
8.5
9
9.5
10];
b4=[1.75E-04
0.025
0.058
0.072
0.088
0.094
0.096
0.097
0.097
0.097
0.099
0.1
0.11
0.11
0.11
0.11
0.11
0.11
0.11
0.11
0.11
0.11];
a5=[0
0.083333333
0.25
0.333333333
0.5
0.583333333

0.66666667
0.83333333
1.16666667
1.25
2.5
3
3.58333333
4
4.08333333
4.33333333
4.5
4.66666667
4.83333333
5
5.33333333
5.66666667
5.91666667
6.08333333
6.41666667];
b5=[0.00015
0.00029
0.00058
0.00074
0.001
0.0012
0.0014
0.0016
0.0021
0.0022
0.0046
0.0056
0.0066
0.0074
0.0078
0.008
0.0083
0.0086
0.009
0.0094
0.011
0.011
0.012
0.013
0.013];
a6=[0
2.66666667

3.16666667
3.25
3.33333333
3.5
3.58333333
3.66666667
3.83333333
3.91666667
4.08333333
5.41666667
5.5
6
6.5
6.666667
7
7.5
8
8.5
9];
b6=[0
0.005
0.006
0.0061
0.0063
0.0067
0.0069
0.007
0.008
0.0081
0.0086
0.01
0.01
0.011
0.012
0.013
0.015
0.016
0.016
0.017
0.018];
a7=[0.08333333
0.16666667
0.25
0.41666667
0.5
0.58333333

0.66666667

0.75

0.91666667

1.08333333

1.25

1.66666667

3

3.33333333

3.58333333

10];

b7=[4.30E-04

0.00089

0.0012

0.0097

0.032

0.087

0.15

0.27

0.77

1.61

2.04

2.26

2.28

2.29

2.29

2.3];

u1=[b1';a1'];

u2=[b2';a2'];

u3=[b3';a3'];

u4=[b4';a4'];

u5=[b5';a5'];

u6=[b6';a6'];

u7=[b7';a7'];

% We interpolate above data to have nn points

nn=500;

% process.m is the interpolating function

[p1,times1]=process(u1,nn);

% process1 is a m-file to sample the data

% to be used by sliding time delay NN

sy1=process1(p1');

p1max=p1(length(p1));

yx1=ones(length(p1)/5,1)*p1max;

```
% Regenerate the transport code plots
```

```
plot(times1,p1,'.');
```

```
xlabel('Time (Days)');
```

```
ylabel('Dose (cGy)');
```

```
Title('Dose-Time curves from actual data')
```

```
for i=1:nn/5
```

```
tt1(i)=times1(i*5);
```

```
end
```

```
% Do the above for all events for skin
```

```
figure(2)
```

```
[p2,times2]=process(u2,nn);
```

```
sy2=process1(p2');
```

```
p2max=p2(length(p2));
```

```
yx2=ones(length(p1)/5,1)*p2max;
```

```
plot(times2,p2,'.');
```

```
xlabel('Time (Days)');
```

```
ylabel('Dose (cGy)');
```

```
Title('Dose-Time curves from actual data for skin')
```

```
for i=1:nn/5
```

```
tt2(i)=times2(i*5);
```

```
end
```

```
figure(3)
```

```
[p3,times3]=process(u3,nn);
```

```
sy3=process1(p3');
```

```
p3max=p3(length(p3));
```

```
yx3=ones(length(p1)/5,1)*p3max;
```

```
plot(times3,p3,'.');
```

```
xlabel('Time (Days)');
```

```
ylabel('Dose (cGy)');
```

```
Title('Dose-Time curves from actual data for skin')
```

```
for i=1:nn/5
```

```
tt3(i)=times3(i*5);
```

```
end
```

```
figure(4)
```

```
[p4,times4]=process(u4,nn);
```

```
sy4=process1(p4');
```

```

p4max=p4(length(p4));
yx4=ones(length(p1)/5,1)*p4max;
plot(times4,p4,'.');
xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data for skin')
for i=1:nn/5
tt4(i)=times4(i*5);
end
figure(5)
[p5,times5]=process(u5,nn);
sy5=process1(p5);
p5max=p5(length(p5));
yx5=ones(length(p1)/5,1)*p5max;
plot(times5,p5,'.');
xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data for skin')
for i=1:nn/5
tt5(i)=times5(i*5);
end
figure(6)
[p6,times6]=process(u6,nn);
sy6=process1(p6);
p6max=p6(length(p6));
yx6=ones(length(p1)/5,1)*p6max;
plot(times6,p6,'.');
xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data for skin')
for i=1:nn/5
tt6(i)=times6(i*5);
end
figure(7)
[p7,times7]=process(u7,nn);
sy7=process1(p7);
p7max=p7(length(p7));
yx7=ones(length(p1)/5,1)*p7max;
plot(times7,p7,'.');
xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data for skin')
for i=1:nn/5
tt7(i)=times7(i*5);
end

```

```

x1=[sy1;sy2;sy3;sy4;sy5;sy6;sy7];
y1=[yx1;yx2;yx3;yx4;yx5;yx6;yx7];
tz1=[tt1';tt2';tt3';tt4';tt5';tt6';tt7'];

```

```

% Enter actual data for eye

```

```

a1=[0.627083
0.70417
0.79375
0.877083
1.04375
1.127083
1.210417
1.377083
1.54375
1.710417
1.877083
2.04375
2.29375
2.460417
2.79375
3.54375
10];

```

```

b1=[0.100781
0.90705
2.819334
6.183382
15.47196
20.97496
26.79229
36.86057
45.11669
92.32921
137.7565
150.3971
158.7104
163.7184
165.8122
168.6765
289.3685];

```

```

a2=[0.083333333
0.25
0.333333333
0.5

```

0.66666667
0.83333333
1
1.25
1.5
3];
b2=[0.0014
0.0067
0.032
0.11
0.26
0.42
0.53
0.59
0.64
0.68];
a3=[0.08333333
0.16666667
0.33333333
0.5
0.75
1
1.33333333
2];
b3=[0.001389728
0.007766993
0.09409079
0.5233271
1.611573
3.204753
4.316268
4.62749];
a4=[0.08333333
0.16666667
0.25
0.33333333
0.5
1
1.5
2
2.5
3
3.5
4.08333333
6
6.5

7
7.5
8
8.5
9
9.5
10];
b4=[4.88E-04
0.11
0.42
0.63
0.91
1.05
1.07
1.08
1.08
1.08
1.09
1.1
1.12
1.14
1.16
1.16
1.17
1.17
1.17
1.17
1.17];
a5=[0
0.083333333
0.25
0.333333333
0.5
0.583333333
0.666666667
0.833333333
1.166666667
1.25
2.5
3
3.583333333
4
4.083333333
4.333333333
4.5
4.666666667

4.83333333
5
5.33333333
5.66666667
5.91666667
6.08333333
6.41666667];
b5=[0.00036
0.00072
0.0014
0.0017
0.0024
0.0028
0.0032
0.0039
0.0051
0.0054
0.011
0.014
0.016
0.022
0.025
0.029
0.032
0.036
0.042
0.05
0.073
0.093
0.13
0.14
0.14];
a6=[0
2.66666667
3.16666667
3.25
3.33333333
3.5
3.58333333
3.66666667
3.83333333
3.91666667
4.08333333
5.41666667
5.5
6

6.5
6.6666667
7
7.5
8
8.5
9];
b6=[0.005
0.015
0.017
0.018
0.02
0.048
0.073
0.11
0.16
0.165
0.19
0.2
0.21
0.21
0.22
0.22
0.23
0.23
0.23
0.24
0.24];
a7=[0.083333333
0.166666667
0.25
0.416666667
0.5
0.583333333
0.666666667
0.75
0.916666667
1.083333333
1.25
1.666666667
3
3.333333333
3.583333333
10];
b7=[2.10E-03
0.0037

```

0.0054
0.25
1.05
3.23
6.11
10.69
29.18
57.22
76.83
80.1
81.5
81.8
82
82.6];
u1=[b1';a1'];
u2=[b2';a2'];
u3=[b3';a3'];
u4=[b4';a4'];
u5=[b5';a5'];
u6=[b6';a6'];
u7=[b7';a7'];

% perform interpolation like we did
% for skin

nn=500;
[p1,times1]=process(u1,nn);
sy1=process1(p1');
p1max=p1(length(p1));
yx1=ones(length(p1)/5,1)*p1max;
plot(times1,p1,'.');
xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data')
for i=1:nn/5
    tt1(i)=times1(i*5);
end
figure(2)
[p2,times2]=process(u2,nn);
sy2=process1(p2');
p2max=p2(length(p2));
yx2=ones(length(p1)/5,1)*p2max;
plot(times2,p2,'.');
xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data')

```

```

for i=1:nn/5
tt2(i)=times2(i*5);
end
figure(3)
[p3,times3]=process(u3,nn);
sy3=process1(p3');
p3max=p3(length(p3));
yx3=ones(length(p1)/5,1)*p3max;
plot(times3,p3,'.');
xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data')
for i=1:nn/5
tt3(i)=times3(i*5);
end
figure(4)
[p4,times4]=process(u4,nn);
sy4=process1(p4');

p4max=p4(length(p4));
yx4=ones(length(p1)/5,1)*p4max;
plot(times4,p4,'.');
xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data')
for i=1:nn/5
tt4(i)=times4(i*5);
end
figure(5)
[p5,times5]=process(u5,nn);
sy5=process1(p5');
p5max=p5(length(p5));
yx5=ones(length(p1)/5,1)*p5max;
plot(times5,p5,'.');
xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data')
for i=1:nn/5
tt5(i)=times5(i*5);
end
figure(6)
[p6,times6]=process(u6,nn);
sy6=process1(p6');
p6max=p6(length(p6));
yx6=ones(length(p1)/5,1)*p6max;
plot(times6,p6,'.');
```

```

xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data')
for i=1:nn/5
    tt6(i)=times6(i*5);
end
figure(7)
[p7,times7]=process(u7,nn);
sy7=process1(p7);
p7max=p7(length(p7));
yx7=ones(length(p1)/5,1)*p7max;
plot(times7,p7,'.');
xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data')
for i=1:nn/5
    tt7(i)=times7(i*5);
end
xx=[sy1;sy2;sy3;sy4;sy5;sy6;sy7];
yy=[yx1;yx2;yx3;yx4;yx5;yx6;yx7];
ttz=[tt1';tt2';tt3';tt4';tt5';tt6';tt7'];
xxx=[x1;xx];
yyy=[y1;yy];
ttt=[tz1;ttz];

```

% Enter data for ~~bone marrow~~ skin

```

a1=[0.627083
0.70417
0.79375
0.877083
1.04375
1.127083
1.210417
1.377083
1.54375
1.710417
1.877083
2.04375
2.29375
2.460417
2.79375
3.54375
10];

```

```

b1=[0.108133

```

1.01208
3.20229
7.081388
17.86957
24.42482
31.37366
43.50455
53.61958
116.5384
179.2399
196.5419
208.2598
215.032
217.9803
222.0246
380.387];

a2=[0.083333333
0.25
0.333333333
0.5
0.666666667
0.833333333
1
1.25
1.5
3];

b2=[0.0016
0.0084
0.047
0.18
0.46
0.75
0.96
1.08
1.15
1.2];

a3=[0.083333333
0.166666667
0.333333333
0.5
0.75
1
1.333333333
2];
b3=[0.002080648

0.013355603
0.1721198
0.9004272
2.907275
5.994321
8.206882
8.838538];
a4=[0.083333333
0.166666667
0.25
0.333333333
0.5
1
1.5
2
2.5
3
3.5
4.083333333
6
6.5
7
7.5
8
8.5
9
9.5
10];
b4=[5.50E-04
0.13
0.54
0.86
1.26
1.48
1.51
1.52
1.52
1.52
1.53
1.54
1.56
1.59
1.61
1.62
1.63
1.63

1.63
1.63
1.65];
a5=[0
0.083333333
0.25
0.333333333
0.5
0.583333333
0.666666667
0.833333333
1.166666667
1.25
2.5
3
3.583333333
4
4.083333333
4.333333333
4.5
4.666666667
4.833333333
5
5.333333333
5.666666667
5.916666667
6.083333333
6.416666667];
b5=[0.00039
0.00079
0.0015
0.0018
0.0026
0.003
0.0035
0.0043
0.0056
0.006
0.012
0.015
0.018
0.024
0.028
0.033
0.038
0.043

0.051
0.062
0.094
0.12
0.18
0.2
0.2];
a6=[0
2.666666667
3.166666667
3.25
3.333333333
3.5
3.583333333
3.666666667
3.833333333
3.916666667
4.083333333
5.416666667
5.5
6
6.5
6.6666667
7
7.5
8
8.5
9];
b6=[0.005
0.0168
0.0196
0.021
0.023
0.064
0.1
0.16
0.25
0.251
0.295
0.32
0.32
0.32
0.33
0.35
0.35

```

0.35
0.35
0.35];
a7=[0.083333333
0.166666667
0.25
0.416666667
0.5
0.583333333
0.666666667
0.75
0.916666667
1.083333333
1.25
1.666666667
3
3.333333333
3.583333333
10];
b7=[2.60E-03
4.40E-03
0.0066
0.41
1.774
5.6
10.74
18.79
50.75
98.48
133.6
137.87
139.9
140.8
141.1
143.5];
u1=[b1';a1'];
u2=[b2';a2'];
u3=[b3';a3'];
u4=[b4';a4'];
u5=[b5';a5'];
u6=[b6';a6'];
u7=[b7';a7'];

% perform the interpolation for bone marrow doses
nn=500;
[p1,times1]=process(u1,nn);

```

```

sy1=process1(p1');
p1max=p1(length(p1));
yx1=ones(length(p1)/5,1)*p1max;
plot(times1,p1,'.');
xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data')
for i=1:nn/5
tt1(i)=times1(i*5);
end
figure(2)
[p2,times2]=process(u2,nn);
sy2=process1(p2');
p2max=p2(length(p2));
yx2=ones(length(p1)/5,1)*p2max;
plot(times2,p2,'.');
xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data')
for i=1:nn/5
tt2(i)=times2(i*5);
end
figure(3)
[p3,times3]=process(u3,nn);
sy3=process1(p3');
p3max=p3(length(p3));
yx3=ones(length(p1)/5,1)*p3max;
plot(times3,p3,'.');
xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data')
for i=1:nn/5
tt3(i)=times3(i*5);
end
figure(4)
[p4,times4]=process(u4,nn);
sy4=process1(p4');

p4max=p4(length(p4));
yx4=ones(length(p1)/5,1)*p4max;
plot(times4,p4,'.');
xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data')
for i=1:nn/5
tt4(i)=times4(i*5);

```

```

end
figure(5)
[p5,times5]=process(u5,nn);
sy5=process1(p5);
p5max=p5(length(p5));
yx5=ones(length(p1)/5,1)*p5max;
plot(times5,p5,'.');
xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data')
for i=1:nn/5
tt5(i)=times5(i*5);
end
figure(6)
[p6,times6]=process(u6,nn);
sy6=process1(p6);
p6max=p6(length(p6));
yx6=ones(length(p1)/5,1)*p6max;
plot(times6,p6,'.');
xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data')
for i=1:nn/5
tt6(i)=times6(i*5);
end
figure(7)
[p7,times7]=process(u7,nn);
sy7=process1(p7);
p7max=p7(length(p7));
yx7=ones(length(p1)/5,1)*p7max;
plot(times7,p7,'.');
xlabel('Time (Days)');
ylabel('Dose (cGy)');
Title('Dose-Time curves from actual data')
for i=1:nn/5
tt7(i)=times7(i*5);
end
xx1=[sy1;sy2;sy3;sy4;sy5;sy6;sy7];
yy1=[yx1;yx2;yx3;yx4;yx5;yx6;yx7];
ttzz=[tt1';tt2';tt3';tt4';tt5';tt6';tt7'];
x=[x1;xx;xx1];
y=[y1;yy;yy1];
t=[tz1;tz;ttzz];
% Generate a data file w/ the name aall.mat
% put all doses and max dose and time in aall
save aall x y t

```

Appendix D

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%      LMTRAINP:
%          Retrain a neural network using LM (trainlm).
%          Using weights from training parametric data.
%          Wes Hines
%      Modified by: Nazila Tehrani
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

vecfile = input('Enter filename of input/output data vectors: ','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) = .01\n');          tolerance = .01;
fprintf('Errors plotted in progress every 10 epochs.\n'); epoch_disp = 10 ;plotflag = 1;
fprintf('Maximum number of training epochs = 300.\n');    maxepochs=300;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      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('Maximum 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 target 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',nov1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      Scale inputs.                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load WEIGHTP
if sc == 'm'
    xn = makenice(x,xm,xs);
elseif sc == 'l'
    xn = scale(x,xm,xs);
else
    xn=x;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      SET TRAINING PARAMETERS FOR LEVENBERG-MARQUARDT      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

TP=[epoch_disp,maxepochs,tolerance,.0001,.001,10,0.1,1e10];

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

if sel2 == 't'                                % Second layer selection
    F2 = 'tansig';
elseif sel2 == 'l'
    F2 = 'purelin';
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*** LM Training complete, no answer found!')

 fprintf('\n*** SSE = %e \n\n',min(TRLM))

else

%%

% LM Tolerance criteria met. %

%%

 fprintf('\n*** LM Training complete after %0.f epochs! ***',epochs)

 fprintf('\n*** SSE = %e \n\n',min(TRLM))

end

save WEIGHTFP W1 B1 W2 B2 xm xs F1 F2 sc

Appendix E

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% pall.m
%
% Use the parametric data and sample the doses
% to be used by sliding time delay neural network
% Nazila Tehrani
% copyright 1998
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Enter the time

t=[0:1:10];

% Enter # of events

n=21;
y=zeros(length(t),n);

yx=ones(length(t)/5,n);

% Enter dose infinity

m=[218 8.9 1.21 .321 143.6 1.53 .21 166 4.7 .69 .211 82.7,...
1.09 .145 22.1 .073 .017 .012 2.31 .099 .0131];

% Enter alpha values

a=[.95 .99 1.13 .26 .95 2.52 .18 .56 1.01 1.1 .26 .95,...
2.5 .18 .63 1.13 .97 .27 .92 3.95 .22];

% Enter gamma values

g=[6.06 3.37 2.92 16.2 5.65 1.46 7.14,...
5.55 3.27 2.68 15.3 5.77 1.32 14.4 4.65 2.84 1.61 2.44 5.62 .87 3.61];

% Use the Wiebull equation to find doses

for i=1:n
yx(:,i)=m(i).*yx(:,i);
y(:,i) = m(i).*(1 - exp(-(a(i).*t).^g(i))));
end

```

```

% Generate the dose-time profiles

plot(y)
xlabel('Time (Days)')
ylabel('Dose (cGy)')
title('Dose vs. Time using parametric data')
pause
% This file will sample the data sets so that we
% have five inputs equally separated in time

% Use function process1 to sample data
% to be used by sliding time delay NN

for i=0:n-1
j=i*length(t)/5;
yy(j+1:j+(length(t)/5),1:5)=process1(y(:,i+1));
end

% Generate the sampled data plots

figure(2);
plot(yy)
xlabel('Time (Days)')
ylabel('Dose (cGy)')
title('Dose vs. Time after sampling')

pause
for i=0:n-1
j=i*length(t)/5;
maxy(j+1:j+(length(t)/5),1)=yx(:,i+1);
end

% Plot the target or max values

figure(3);
plot(maxy)
xlabel('Time (Days)')
ylabel('Max Dose (cGy)')
title('Target values')
pause
x=yy;
y=maxy;
% save results in data file pall to be used
% for training
save pall x y

```

Appendix F

```
#####
%
%       process1.m is sampling the data to be used
%       in sliding time delay neural network
#####

% To use the data for STDNN the data should be equally spaced
% In this research we convert the data to 5 equally spaced sets
function y=process1(x)

len=length(x);
y=zeros(len/5,5);
for i=5:5:len
    y(i/5,1)=x(i/5);
    y(i/5,2)=x(i/5*2);
    y(i/5,3)=x(i/5*3);
    y(i/5,4)=x(i/5*4);
    y(i/5,5)=x(i);
end
```

Appendix G

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% testfff.m
% use wieghts from 1st parametric subset
% test the 2nd parametric subset
% Plot the dose-time curves
%
% Nazila Tehrani
% copyright 1998
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load ppall

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);

%t=0:.1:10;
yy=Y';

ttt=2:.5:10;
t1=ttt';
for i=0:14
    ii=i+1;
    fprintf('Event, %g ,\n\n',ii)
    %figure(i+1)
    j=i*20+4;
    jj=(i+1)*20;
    q=yy(j:jj,1);
    p=y(j:jj,1);
    z=x(j:jj,1);
    error=(p-q)./p;
    errora=error .* p;
    disp(p)
    disp(error)
    disp(errora)
    %fprintf('          Error          Maximum Dose          Actual Er          \n\n')
    %fprintf('          %g          %g          %g,\n\n',error,p,errora)
    %plot(t1,q,t1,p,'+',t1,z,'-')
    %title('Testing para data w/ weights from training NN w/ different para events');
    %xlabel('Time (Days)');
    %ylabel('Dose (cGy)');
end

```

Appendix H

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   testf.m
%       This matlab file does the testing and plot the actual dose
%   and prediction for actual dose vs. time
%
%   Nazila Tehrani
%   copyright 1998
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   data = input('Enter filename of input/output data: ','s');
%   load aall
%   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);

yy=Y';
p1=t(1:100);
q1=yy(1:100);
yyy1=y(1:100);
xxx1=x(1:100);
error1=(yyy1-q1)./yyy1;
erroral=error1 .* yyy1;

plot(p1,q1,p1,yyy1,'+',p1,xxx1,'-')

title('Prediction SUPERIMPOSED ON ACTUAL');
xlabel('Time (Days)');
ylabel('DOSE (cGy)');
%axis([0 10 0 500]);
figure(2)
p2=t(101:200);
q2=yy(101:200);
yyy2=y(101:200);
xxx2=x(101:200);
error2=(yyy2-q2)./yyy2;
errora2=error2 .* yyy2;

plot(p2,q2,p2,yyy2,'+',p2,xxx2,'-')

title('Prediction SUPERIMPOSED ON ACTUAL');
xlabel('Time (Days)');
ylabel('DOSE (cGy)');
%axis([0 10 0 50]);
figure(3)
p3=t(203:297);
q3=yy(203:297);
yyy3=y(203:297);
xxx3=x(203:297);
error3=(yyy3-q3)./yyy3;
errora3=error3 .* yyy3;

plot(p3,q3,p3,yyy3,'+',p3,xxx3,'-')

title('Prediction SUPERIMPOSED ON ACTUAL');
xlabel('Time (Days)');
ylabel('DOSE (cGy)');
%axis([0 10 0 50]);
figure(4)
p4=t(303:397);
q4=yy(303:397);

```

```

yyy4=y(303:397);
xxx4=x(303:397);
error4=(yyy4-q4)./yyy4;
errora4=error4 .* yyy4;

plot(p4,q4,p4,yyy4,'+',p4,xxx4,'-')

title('Prediction SUPERIMPOSED ON ACTUAL');
xlabel('Time (Days)');
ylabel('DOSE (cGy)');
%axis([0 10 0 50])
figure(5)
p5=t(403:497);
q5=yy(403:497);
yyy5=y(403:497);
xxx5=x(403:497);
error5=(yyy5-q5)./yyy5;
errora5=error5 .* yyy5;

plot(p5,q5,p5,yyy5,'+',p5,xxx5,'-')

title('Prediction SUPERIMPOSED ON ACTUAL');
xlabel('Time (Days)');
ylabel('DOSE (cGy)');
%axis([0 10 0 50])
figure(6)
p6=t(503:597);
q6=yy(503:597);
yyy6=y(503:597);
xxx6=x(503:597);
error6=(yyy6-q6)./yyy6;
errora6=error6 .* yyy6;

plot(p6,q6,p6,yyy6,'+',p6,xxx6,'-')

title('Prediction SUPERIMPOSED ON ACTUAL');
xlabel('Time (Days)');
ylabel('DOSE (cGy)');
%axis([0 10 0 50])
figure(7)
p7=t(603:697);
q7=yy(603:697);
yyy7=y(603:697);
xxx7=x(603:697);
error7=(yyy7-q7)./yyy7;
errora7=error7 .* yyy7;

plot(p7,q7,p7,yyy7,'+',p7,xxx7,'-')

title('Prediction SUPERIMPOSED ON ACTUAL');
xlabel('Time (Days)');
ylabel('DOSE (cGy)');
%axis([0 10 0 50])

```

Appendix I

```
function [y,times] = process(data,num);
%PROCESS      Preprocess data for a Sliding Time Delayed Neural Network
%      [y] = process(FS,level)
%
%      Takes data and interpolates num equally spaced data points
%      y      : num equally spaced data points
%      Takes data and interpolates num equally spaced data points
%      y      : num equally spaced data points
%      data   : matrix consisting of two row vectors
%              1. data
%              2. respective sample times (start with 0)
%      num    : number of data points to return
%
% Wes Hines
% The University of Tennessee
% Copyright 1998

[row col]=size(data);

if row ~=2
    error('Data matrix needs to have 2 rows.');
```

```
end

% Find num equally spaced times.
times=zeros(1,num);
tmax=data(row,col);

for i=1:num
    times(i)=tmax/num*i;
end

% Now we need to interpolate and find the data points at those times.
y=zeros(1,num);
for i=1:num-1
    inda=max(find((data(2,:)-times(i))<0));
    indb=min(find((data(2,:)-times(i))>0));
    a=data(1,inda);
    b=data(1,indb);
    c=data(2,inda);
    d=data(2,indb);
    if isempty(a)
        a=0;
    end
    y(i)=a+(b-a)*(times(i)-c)/(d-c);
end
y(num)=data(1,col);
```

VITA

Nazila H. Tehrani was born in Tehran, Iran in 1966. She completed her High School education in 1983. She was first honors among all of the High Schools in her zone. She took the National Entrance Examination and ranked 158 out of 600,000 participants. She graduated from The University of Tehran, Iran with a Bachelor of Science in Chemical Engineering, minor in Petrochemical Engineering in May 1988. After 9 months working in a Polymer Research Center she came to the US to pursue her Master 's degree. She graduated in May 1993 with a Master of Science in Chemical Engineering from The University of Tennessee, Knoxville. Her thesis topic was, "Simulation of Pipe Extrusion: Cooling and Annealing." After being hired as a quality assurance manager in a plastic pipe manufacturing company, she decided to pursue her doctoral degree.

She joined the Nuclear Engineering department at the University of Tennessee, Knoxville in August 1994. She is an active member of the Women's Engineer Society, American Institute of Chemical Engineers (AIChE), American Nuclear Society (ANS) and the Health Physics Society.

She has been involved in teaching in the Basic Engineering Curriculum since 1990. She taught programming in the Basic Engineering Computer Laboratory. She was also an instructor for a "Numerical Methods for Engineers" course in the Basic Engineering curriculum.