ISTANBUL TECHNICAL UNIVERSITY * INSTITUTE OF NUCLEAR ENERGY

STATE ESTIMATION IN A NUCLEAR REACTOR WITH THE HELP OF ARTIFICIAL NEURAL NETWORKS

M.S.THESIS

Bektaş Ali KÖKSAL, B.S.

Department : NUCLEAR SCIENCES

Program : NUCLEAR ENERGY

ISTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF NUCLEAR ENERGY

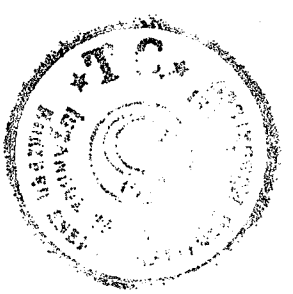
STATE ESTIMATION IN A NUCLEAR REACTOR WITH THE HELP OF ARTIFICIAL NEURAL NETWORKS

M.S.THESIS

Bektaş Ali KÖKSAL, B.S.

Date of Submission : 16 January 1995
Date of Approval : 30 January 1995
Examination Comittee
Supervisor : Prof. Dr. Melih GEÇKİNLİ
Member : Prof. Dr. Şarman GENÇAY
Member : Assoc. Prof. Dr. Erdinç EDGÜ

JANUARY 1995



PREFACE

This work is a thesis submitted by the author to the Institude of Nuclear Energy of Istanbul Technical University in partial fulfillment of the requirements for the degree of Master of Science in Nuclear Energy. It is mainly an application of the state of the art artificial neural networks methodology to the solution of a nuclear engineering problem of practical value.

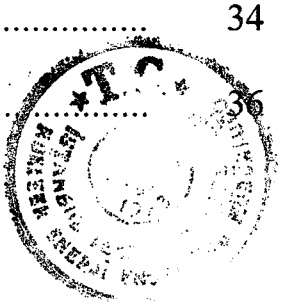
I wish to express my deep appreciation to Prof. Dr. Melih Melih GEÇKİNLİ for his guidence and help.

Also I would like to thank İbrahim Engin BAYKAL for his help in entering the manuscripts to computer environment. This work is partially sponsored by the ITU Research Fund.

January, 1995

Bektaş Ali Köksal
B.S. Control and Computer Eng.

TABLE OF CONTENTS



LIST OF SYMBOLS

C_i (t) : Density of the delayed neutron precursor of the i*th* type
c : Capacitance
D(t) : Delayed neutron kernel of the inverse kinetic equation
H : Average convective heat transfer coefficient (W/K)
M : Thermal capacity of the core (W.s/K)
R : Resistance
W,z : Synaptic weights
X,x : Input vector to a neuron
Y,y : Nonlinear output from a neuron
S,s : Linear output from a neuron
S(t) : Neutron source
T_f(t) : Average fuel temperature in the core
T_w(t) : Pool water temperature
T_0 : Reference temperature for fuel elements ($T_0 = T_w$)
α : Training parameter; prompt temperature coefficent of reactivity ($\Delta k/k/K$)
β : Total delayed neutron fraction $\beta = \sum_i \beta_i$
β_i : Delayed neutron fraction of the i*th* type
δ : Sensitivity of the output error to changes in the linear output of a neuron
ε : Output error; sensitivity of output error to changes in the nonlinear output of a neuron
γ_i : Relaxation coefficient of a recurrent neural node
η : Momentum parameter
λ_i : Decay constant of the precursor of type i (s^{-1})
Λ : Neutron generation time (s)
μ : Neural network training parameter
μn : Thermal power released in the fuel elements (W)
ρ(t) : Net reactivity ($\rho(t) = \rho_{ext} + \rho_{fb}$)
σ(t) : Sigmoid transfer function for a neuron

LIST OF FIGURES

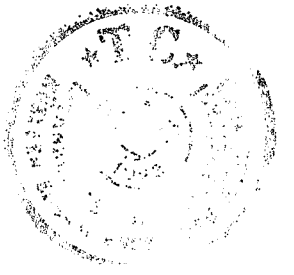
LIST OF TABLES

SUMMARY

In a nuclear reactor the instantaneous estimate of hidden state variables, which might be exploited for control purposes, is a difficult task most of the time. For the estimation to be dependable, the past dynamical behavior of the system need to be known from the given initial state on. In practice, for a rough estimation, without compromising critical safety issues, one can just do with data for a short period of time prior to the demand. From another point of view, there may exist a number of unknown correlations or functional relationships between the observable and unobservable state variables under the imposed operating procedures. Such correlations are hard to establish analytically by conventional methods, yet artificial neural networks can be trained with the help of some analytical models so that they can learn to estimate the state in a real situation. For this purpose, we investigated the performance of multilayer feed forward networks trainable via back propogation algorithm. A successfully trained network is expected to estimate the state with a reasonable accuracy at a short glance of the pattern of variation of the observable state variables.

Out of three possible schemes trained and tested, the only multilayer feed-forward network, which proved to be successful, estimates the net reactivity of the core when presented with a set of delayed sequences of sampled reactor power, average fuel temperature, and the reactivity worth of the control rod. A recurrent type neural network which is recommended to solve the same estimation problem will surely provide an economy in the number of input variables and the number of synaptic weights to be adjusted.

This study is a sequel to the TRIGA Dynamics Projects which have been undertaken at the Institute for over a period of time, and partially sponsered by the ITU Research Fund.



ÖZET

BİR NÜKLEER REAKTÖRDE YAPAY SİNİR AĞLARI YARDIMIYLA DURUM DEĞİŞKENLERİNİN KESTİRİLMESİ

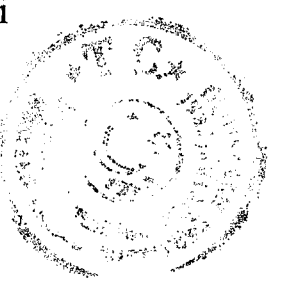
Sinir ağı uygulamaları ya da alışıla gelen ismi ile Yapay Sinir Ağları birbirleri ile çok yoğun bir şekilde bağlantılandırılmış basit hesaplama birimlerinden oluşurlar. Yapay sinir mimarisi yapacağı göreve göre seçilir. Genelde nöronlar "LEGO" modülleri gibi eklenerek daha büyük yapılar oluştururlar. Paralel ve dağılmış işlemci yapısı sinir ağlarına "spread sheet" gibi bir özellik kazandırır.

Sinir ağları statik örüntü tanımada, fonksiyon kestiriminde, sistem dinamiği benzeşiminde, içerik adresli bellek oluşturmada, istatistik gruplamada kullanılabilirler.

Uygulama olarak, el yazısı okuma ve seslendirme, proses kontrolu, optik algılama, konuşma algılama, finans mühendisliğinde borsa trendi belirleme gibi, doğrudan nümerik yöntemlerle çözümlenmesi zor olan veya algoritması bilinmeyen popüler problemler gösterilebilir.

Bellek kapasiteleri geniş olan ağlar verilen örnek problemleri öğrenip genellemek yerine ezberlemek yoluna gidebilirler. Bu durumun olup olmadığını anlamak için egğitim esnasında yeni test problemleri ile sistemin performansı sürekli olarak sınanır.

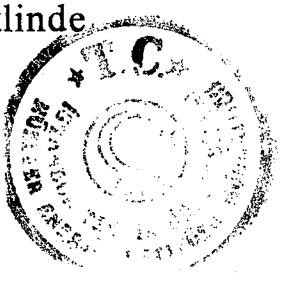
Sinir ağlarında kullanılan sigmoid ve Gaussian transfer fonksiyonları, bu sistemlerle bulanık mantık sistemleri arasında organik bir bağlantı sağlamaktadır.



Yapay sinir ağlarının insan beynindeki sinir hücrelerini model olarak almaları güçlü, hataya bağışık, gerçek zamanda cevap verme süreleri çok kısa olan sistemlerin gerçeklenmelerini olası kılmıştır. Şekil tanıma ve durum kestirme uygulamalarında, giriş bilgilerinde kesintilerin, gürültülerin yoğun olduğu durumlarda bile yüksek başarılar gösterebilmektedirler. Yapay sinir ağlarıyla kurulan sistemlerin performanslarında olabilecek düşmelerin doğrudan doğruya ağdaki hücrelerin sayısındaki düşmeyle orantılı olması, bu tür ağlarla hata bağışık sistemlerin kurulabilmesinin ana nedenini oluşturmaktadır.

Uygulamalarda yapay sinir ağının tanımasının veya sınıflandırmasının istendiği bilgiler, bir şekilde ölçümlerle veya modeller yardımıyla elde edilmekte ve önceden tanımlanmış kriterler sağlanıncaya kadar yani öğrenme işlemi gerçekleninceye kadar ağa beslenmektedir. Bu işlemler aslında sistemin çeşitli durumlarını tanımlayan bilgilerin ve bu bilgilerdeki değişimlerin karşılık düştüğü çeşitli durumların ve durum geçişlerinin bir birleriyle ilişkileri içerisinde sistemin sürecini tanımlayacak şekilde yapay sinir ağınca öğrenilmesine karşılık düşmektedir.

Bu öğrenme ya da çalışma şekli çözümde kullanılan yapay sinir hücresi modeline göre değişmekte ve öğretme algoritması buna göre belirlenmektedir. Öğrenme işlemine göre yapay sinir ağları, öğretme işleminin ; "gösterilerek" yapılması yani sistemin ürettiği cevabın doğrusu ile karşılaştırılması (supervised training) ya da öğrenilmenin "kendini düzenliyerekten" (self organized mapping) kendiliğinden gerçekleşmesi, yani girdi olarak verilen işaretin içinde gizli olan bir düzenin dışarıdan bir müdahale olmadan bazı kurallara uygun olarak bulu öğrenmesine veya çıkışın değerlendirilmesinin "doğru" ya da "yanlış" şeklinde



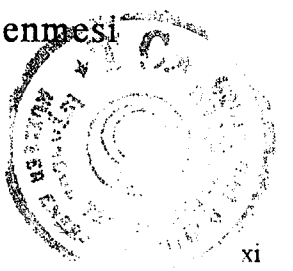
(reinforcement training) yapılmasına göre sınıflandırılmaktadır.

Cevap gösterilerek (supervised) yapılan öğretme işleminde ana fikir sisteme belirli bir anda verilen giriş ile sistemin cevap olarak ürettiği çıkış ile üretilmesi istenilen bilgi arasındaki farkın öğretme işleminde ilerlendikçe sıfıra gitmesinin istenmesi şeklindedir.

"Gösterilmeden" (non supervised) ya da "kendini düzenleyerekten" (self organizing) gerçekleşen öğrenme işleminde kurulan yapay sinir ağının, kendisini besleyen bilgi içerisindeki benzerliklerin çıkarılması, temel bileşenin bulunması,özelliklerin, karşılılıkların, tek düzeliklerin bir şekilde kendi kendine ayırt edebilir hale gelmesi düşüncesi yatmaktadır.

Nükleer endüstride yapay sinir ağları teşhis koyma sistemlerinin kurulmasında, sistem durum geçiş çözümleme işlemlerinde, güç üretim tesislerinde çalışma rejimindeki değişiklerin tanımlanmasında, basınçlı su reaktörlerinin devreye alınmasında, tesisin tamamının izlenmesinde, tesiste kullanılan duyargaların sağlıklı çalıştığının onaylanmasında, reaktör yakıt yükleme organizasyon işlemlerinin gerçeklenmesinde, titreşim çözümleme işlemlerinin yapılmasında, rulman arızalarının belirlenmesinde ve raporlama veri tabanının taranması işlemleri alanlarında uygulamalar bulabilmektedir. Bu uygulamaların çoğunda yapay sinir ağları, uzman sistemler, bulanık mantık ve genetik algoritma teknileri bir arada iç içe kullanılmaktadırlar.

Bu çalışmada İTÜ Triga reaktörü için durum değişkenlerinin kümülatif olarak oluşturduğu net reaktivitenin konsoldan yapay sinir ağları kullanılarak sürekli gözlenmesi



amaçlanmıştır. Yapı olarak, çok tabakalı perseptron modeli seçilmiş ve eğim düşümü yöntemiyle eğitilmiştir. Eğitmek için gereken veriler daha önce geliştirilen ve Triga dinamiğini simüle eden bir program yardımıyla türetilmiştir. Üç tür giriş veri konfügürasyonu denemiştir. Bunlardan giriş nötron populasyonu, yakıt sıcaklığı ve kontrol çubuğu konumu verileri ile beslenen ağ başarılı sonuçlar vermiştir. Diğer iki modelde giriş sadece nötron populasyonu olarak seçilmiştir.

Çalışmanın sonunda, girişi kontrol çubuğunun konumu, çıkışı ise net reaktivite olan bir "recurrent", tersine tekrarlayan ağ önerilmiş, ancak eğitilmemiştir. Bu ağın darbe cevabı sonsuzdur ve gizli durum değişkenleri, gene gizli düğüm noktalarında oluşmaktadır.

Çalışma için FORTRAN dilinde yazılmış olan eğitim programı ekte verilmiştir. Program, ağırlıkları bir vektöre taşımaktadır. Böylece, bilgisayarın kısıtlı belleği üç indeksli fakat seyrek elemanlı bir ağırlık matrisi ile zorlanmamaktadır ve ağ yapısı çarşaf şeklinde iki yönde kolaylıkla açılabilmektedir.

Çalışmanın sunulması aşağıdaki şekilde gerçeklenmiştir: Birinci bölümde reaktörlerde reaktör durumunun izlemesine yönelik bazı çalışmalardan bahsedilmekte ve uygulamalara ilişkin referanslar verilmektedir.Bu bölümün ikinci kısmında ise çalışmanın hedefleri konulmakta ve kısa bir giriş yapılmaktadır.

İkinci bölümde yapay sinir ağlarının nükleer endüstrideki uygulamalarından ve çeşitli potansiyel uygulama alanlarından bahsedilmekte; elde edilen çeşitli sonuçlar özet olarak değerlendirilmektedir.

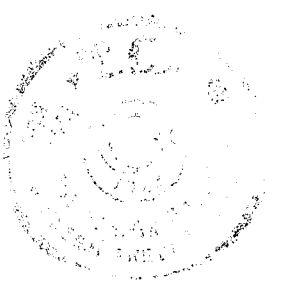
Üçüncü bölümün birinci kısmında bir yapay sinir ağının oluşturulmasında kullanılan birim işlemci eleman ele alınıp tanımlanmaktadır. Bu bölümün ikinci kısmında her tabakasında üç nöronun bulunduğu üç tabakalı bir yapay sinir ağı şekli üzerinde hatanın sondan başa yayılmasına ilişkin

delta formülasyonu denklemlerinin türetilmesi aşamalarıyla verilmiştir.

Bu bölümün üçüncü kısmında ise kendine geri beslemeli (recurrent) tersine tekarlamalı, yapay sinir ağları konu edilmiş ve bu yapının tanımlamaları verilmiştir.

Çalışmanın dördüncü bölümünün birinci ve ikinci kısımlarında bir nükleer reaktörde reaktivitenin tahmin edilmesine ilişkin olarak problem fomüle edilmekte ve reaktivite geri beslemesi durumunda reaktör dinamiği ele alınmaktadır.Bu bölümün üçüncü kısmında ise kullanılan yapay sinir ağı teknik olarak tanımlanmaktadır. Deneysel sonuçlar bu bölümün dördüncü kısmında sunulmuştur.

Çalışma değerlendirme ve önerilerin yer aldığı bir bölümle sonlandırılmış, deneylerde kullanılan program ve bu programa ait giriş ve çıkış dosyaları ekler kısmında verilmiştir.

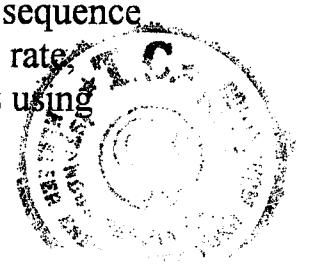


CHAPTER 1.

INTRODUCTION

1.1 PREVIOUS WORK

In the literature , there have been numerous attepts tocontinuously monitor the reactivity of a reactor, from the operating console, on line with a host computer. One of the simplest methods of monitoring and diagnosing the state of a fast reactor is carrying out an instantaneous reactivity balance. This is especially necessary to log the events prior to melt-down of a fuel assembly due to blockage of sodium flow. To achieve this goal, the sodium coolant flow rate, inlet temperature, power, control rod positions are continuously monitored on BOR-60 reactor, [6]. Some implementations of the experimental reactivity meter on research reactors is given in Refs.[3] and [5]. The availability of a reactivity monitor on the console is good practice of added safety measure. From the pedagogical point of view of operator training, this extra information on the state of the reactor helps the trainee understand some of the safety measures already built into the electronics of the console, such as trip and scram mechanisims, and avoids some misconceptions. Also, he or she can gain a clear insight of the dynamic behavior of the reactor, such as internal feedback effects [4]. The operator gets a feeling of the effectiveness of control mechanisms when manouvering the power. External disturbances, such as caused by irradiation samples are also monitored and reported with due importance. In Ref. [4], an ANN implementation of continuous reactivity surveillance in a fast breeder test reactor is given. The input pattern consists of a sequence of core inlet temperature, power, burnup, and coolant flow rate, sampled daily. The ANN is trained by estmating the weights using



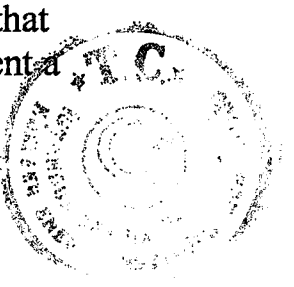
the maximum likelihood method, to predict the steady state control rod position. When a systematic error is observed, the network is re-adapted.

The reactivity estimation algorithms are based on the inverse kinetic equation, [2], [10]. Formally, an instantaneous reactivity estimation is impossible, because the full state of the reactor at any instant of time has to be known exactly, which is not the case. In this work an attempt has been made to put artificial neural networks to use for the solution of this problem. The the network trained has a multilayer perceptron type architectutre. For a better working estimator, a recurrent neural network with dynamical capabilities is recommended.

1.2 OBJECTIVES

ITU TRIGA reactor dynamics is modeled by seven point kinetic equations, and an equation for the energy balance between the core and the constant temperature coolant water, which gives us a total of eight state variables. The system is externally driven by reactivity insertion affected by a control rod. An internal feedback loop on the reactivity due to negative temperature reactivity coefficient of the core, determines the net amount of reactivity operative at any instant of time. The reactivity balance can be set up by simply using the inverse kinetic equation, which requires the past history of the reactor power. If we want to instantaneously determine the net reactivity of the core, then we should know the exact neutronic state. On the other hand six out of the seven neutronic state variables are unobservable. Therefore the problem is mathematically intrackable, and this brings us into the territory of artificial neural networks (ANN). The scope of the work is limited to feedforward multilayer neural networks.

In Chapter 2., some potential and realized ANN applications in nuclear engineering problems are reviewed. In Chapter 3. ANN and its backpropagation training algorithm are introduced. In Chapter 4. three schemes have been tried to solve the reactivity estimation problem. In Chapter 5 it is concluded that multilayer feedforward networks are not adequate to present a



satisfactory solution for the tracking of hidden neutronic state variables of the core. Use of recurrent neural networks is recommended. In Appendix A, a Fortran listing of the code written for supervised training of feedforward neural networks is given.

CHAPTER 2.

NEURAL NETWORK APPLICATIONS IN THE NUCLEAR INDUSTRY

Nuclear industry used to cover up its deeply rooted feelings of insecurity and hypocrisy by acting increasingly overconservative. Those cut and set nuclear rules , regulations, and procedures left no room for experimentation heuristic developement and, innovation that situation went on up to a point of self victimization. However with the wisdom and lessons learned from the past experience, a more positive and progressive attitude is setting in among the new generation of nuclear engineers. Some modern techniques are finding their way into applications to the safe and reliable operation of nuclear power plants. One such method is the artificial neural network whose basic element is a neuron-like processor. Now the ANN models, algorithms and applications are considered under the discipline of computational intelligence,(CI). A recent and extensive review on the potential applications of neural networks to the operation of nuclear power plants was given by Uhrig [21].

Neural network applications as they are named in convention "Artificial Neural Networks " are structures which involve highly interconnected simple processing units.The model they are based on are human brains.This property makes them robust and fault tolerant systems when they are used for real time response purposes when there is a need of patern recognition or state estimation for complex systems.This functionality of recognition or state estimation is succesfully achieved in cases even though the input signals are noisy, sparse, or incomplete. System performance reductions in cases of

network demage are said to be proprotional to the amount of network demaged.

In nuclear industry artificial neural networks had found applications in items such as diagnostics,system transient identification,identification of non-linear dynamics, steam generator transients,detection of change of mode of operation of power plant, neuro-control of PWR start-up,sensor validation, plant wide monitoring, design of nuclear fuel cycle reload,vibration analysis, detection of bearing failures and review of licensee event report database .

Artificial neural networks are used in these fields alone or in conjunction with other various technologies as expert systems, such as fuzzy logic and/or genetic algorithms.

Applications aiming diagnostics of a nuclear plant involves tracking of a vast number of instruments readings taking part in a control room.A collection of all of the readings at a definite time defines the state of the plant at that instant.A transition to a new state is resambled by a new set of readings which may be normal or abnormal.This new set of readings may be used in identifying the new state.This operation corresponds to a patern recognition process.

In an application conducted in University of Tennessee "Autoadaptive " stochastic learning technique based on "dynamic node structure " was used to process data supplied by Watts Bar Nuclear Power Plant. In this application to achieve "near optimization of the interneural connections" or training, nodes were added to the 3 hidden layers of fully interconnected neural network until performance of the network aproaches a limiting value. The degree of higher learning performance was stated as training and genaralizing more rapidly.

The optimization of the fully interconnected neetwork was done via nodal bias, gain, the output activation constant, and connection weights.The network was trained with a Monte Carlo training procedure to identify the following seven different nuclear power plant transitions:

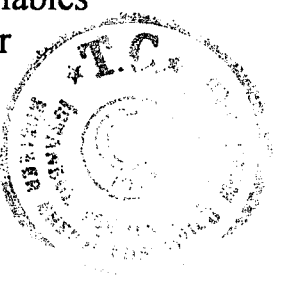
- Loss of coolant in the hot leg of the Reactor Coolant System (RCS)
- Loss of coolant in the cold leg of the RCS.
- Main steam line break in containment.
- Total loss of offsite power.
- Control rod ejection.
- Steam generator tube leakage.
- Main feedwater line break in containment

The Monte Carlo training procedure was also affective in teaching the steady-state of the plant besides the transitions which were expected to be distinguished.An extension to the work was made by using genetic algorithms and multiple neural networks to optimize the neural network configuration. Tracking of more than 500 variables which constituted the input vector of the network was used to train the neural network as an operator.

The aim of the work conducted was to detect accident conditions before the plant is "tripped" so precautions can be taken .

Hybrid systems making use of expert, fuzzy, and artificial neural network systems are also investigated.One of these systems developed at the University of Tennessee used model referenced aproach to classify the inputs.The expert system performs the basic interpretation and model data processing.The outputs of the artificial neural network are membership functions of the model data fed into network. The outputs are used as inputs of the rule based identification (expert system).

In a similar work diagnostic methodology was applied to patterns describing a depressurization in a coolant circuit of a PWR reactor during a loss of coolant accident. The neural network used in this diagnostic process was trained to distinguish four different sizes of breaks. The expected results from the neural network was as identification of these breaks in a noisy and errorenous data supplying environment. The underlying idea in this application was a disturbance in the system will result transients in a group of variables of the system such as changes of levels temperature, pressure or



radiation and the neural network can be trained to identify the disturbances.

In a work at Texas A&M University a recurrent artificial neural network was used to identify the dynamics of a nonlinear system by correlating the inputs and outputs of the system.The work included modeling of a boiler with the help of a multiple layer perceptrons with crossover links inbetween nodes of the same layer including itself (self feed-back or recurrent).

A "neural diagnostician" project proposed at Texas A&M University was aiming to identify power-plant component faults by adapting to drifting plant components in time and reconfiguring the parameters through learning.The project was to include a conventional expert system and two types of neural networks one for optimization and control, a recurrent dynamic network, and one for measurement and logistics an "adaptive sparse distributed memory".

Adaptive control concepts were said to be used in identifying the dynamic behavior of the system.The results of this work states that responses of a nonlinear system can be predicted with limited number of nodes and limited amount of off-line training time.Also subsequent work conducted on online training with the use of a modified backpropogation learning algorithm has been stated to be robust in emprical modelling of the plant.

In a work conducted at the University of Tennessee a network with three outputs 12 artificial neurons in the hidden layer and 40 inputs of four traced variables (ten samples of each traced variable are fed into the network) has been trained with backpropogation method to identify 6 transients that may occur in a U-tube steam generator.The network after 1500 training cycles designated the 6 transients with its 3 bit coded outputs in cases where the input data were corrupted with very large levels of 90% (noise to signal ratio) uniformly distributed noise.

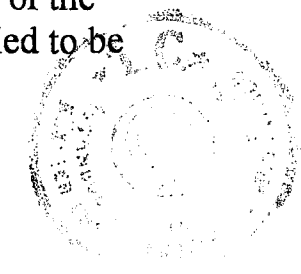
A project on detection of the change of mode in nuclear power plants was conducted in the Unversity fo Tennessee to identify un forseen changes in the operation of a nuclear plant in the most

possible earlier state of operation.The deviation wanted to be discriminated was increase or decrease of power .The problem wanted to be solved was discrimination of data of steady state conditions and data of transient conditions.Two techniques used were polynomial discriminant method and backpropogation network methods. Polynomial discrimanat functions basicly used in patern recognition were making use of Bayes decision rule to form nonlinear decision bounderies for multiple catagory problems.Useful results of the method were rapid matching, high extrapolating ability and aplicability to complex problems.

In the second method mentioned a multilayer fully connected heteroassociative neural network was trained to update its weights between the neurons by least squares method to minimize the error at the outputs.The inputs to the network were data supplied from sensors measuring power level presurizer pressure, RCS (reactor coolent system) hot leg temperature, steam generator main feed water flow and steam generator pressure.

During start up of a PWR reactor control of temperature and pressure of reactor vessel represents a problem similar to that of "Inverted Broom" problem.Aself teaching neural architecture developed in the University of Illinois was used in heat and temperature control of a PWR reaktor.The task was as increasing the reactor temperature from 66 °C to 288°C with a temperature change rate of 56°C per hour while keeping the pressure with in its operating range.The functionality of the network was as assigning a control vector to every state the reactor may be in.The states and the controls were represented as two dimensional vectors, the measured and normalized control values of temperature and pressure.The network is said to be trained in 150 trials.

The concern of an another neural network application conducted at the University of Tennessee was validation of sensor signals placed into the reactor.In this application the network had 21 nodes in the hidden layers and 150 input paterns were used in 3000 iterations to train the network."Adaptive thresholding" was used as a technique to accelerate the back propogation training.Some of the inputs to the network were the actual signals which were tried to be



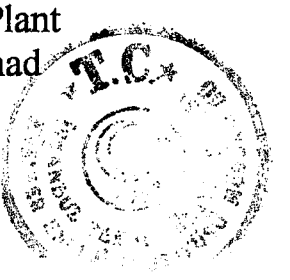
estimated at the outputs.After the training process the ouputs were with the signals fed into the network and any mismatch was considered as sensor failure.

This work for sensor validation was extended to plant wide monitoring. The network used in this project was autoassociative where the inputs and the outputs are the same variables.An optimization was made in the number of neurons taking part to to maximize generalization ability and minimize the training time. The learning algorithm included backpropogation.After training any difference inbetween the estimated value and the tracked variable value was considered as a kind of error.

Feedwater flow into the steam generators of a PWR power plant is measured with venturi flowmeters.The readings taken from these sensors are highly related to the thermal power and its calculations.Fouling of the flow meter or a loosening of the crud may couse the system to derate or exceed the licensed power level which needs to be reported.Sensor validation methods mentioned above are said to be sufficient to monitor fouling.

Check valve failures faced in the near past had atracted attention to monitoring of these values.During failures the feed water system was under water hammering produced by the failures.The failures that may be seen on check valves covered a wide range of reasons.Most of the failure reasons were also reasons of sound, improper movement of valve disk or improper through movements. Various sensors (acoustic, ultrasonic, magnetic) replaced on the valve supplies information on events going on while fluid is transmitted through valve.At the University of Tennessee a program to analyse the acousti cal data taken from check valves with the help of neural networks been started under support of EPRI (Electric Power Research Institute funded by Nuclear Industry Check Valve Group).

A program aiming to generate a model of the thermodynamic process going on in the plant has been conducted at the university of tennessee.In this work 45 measurements of 26 variables of TVA's (Tennesse Valley Authority) Sequoyah Unit 1 Nuclear Power Plant logged in a week were fed into a neural network.The network had



two parts. The first part was a Kohonen network that was used to cluster data into 22 clusters that is 22 neurons and the second part consisted of a back propogation network where the outputs of the first part or the clustered data are fed into. The model after training is said to have a system error of 0.5% . The model of the plant gained by training was concluded to be superior to to results calculated with the aid of models which included assumptions and approximations. The model produced was also used in sensitivity analysis of the plant to changes of various input variables. The partial derivatives of the outputs with respect to different input variables were considered for terms of plant efficiency.

CHAPTER 3.

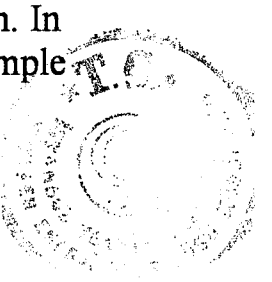
ARTIFICIAL NEURAL NETWORKS

3.1. MICROSTRUCTURE OF ARTIFICIAL NEURAL NETWORKS

The concept of artificial neural network is a paradigm which helped organize and systematize optimization algorithms used in various disciplines under a single title, much as chaos clarified and unified some physically uninterpretable solutions of nonlinear system equations. The computer revolution has been highly conducive in this process. Bringing researches from several fields together stimulated an explosive advancement of the topic. Such a dense synaptic interconnection between the researchers is the main agent for the state of affairs enjoyed by ANN. The popularity of neural networks can be described concisely by the following proverbial quote: "Neural networks are the second best way of doing just about anything".

As the name implies artificial neural networks are inspired by the biological processors, and its terminology is borrowed from neurophysiology. The basic building block of an ANN is a neuron, the function of which is explained in Figure 3.1 .

A neural element can perform a very limited and simple task. It is only when the neural elements are cascaded that the real capacity of a neural structure emerges. To perform its job, a neural network is to be trained. Fundamentally, training procedures can be divided into three categories : supervised training, reinforcement training, and self-organization. In supervised training the network is supplied with a set of "example



problem and solution sets". In reinforcement training there is an example set of "problems" but the answers are missing; instead a teacher grades the performance of the network. Self-organizing networks learn mappings from one space to another all by themselves using some competitive rules as guidelines.

In the next section supervised training by error backpropagation is explained in detail.

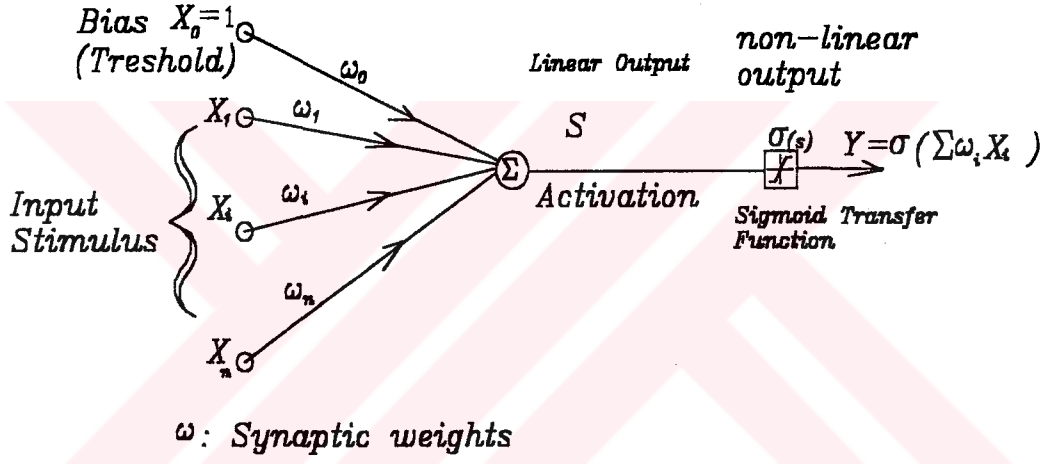


Figure 3.1 A neural element (Sigmoid Adaline).

3.2. THE BACK PROPOGATION ALGORITHM

Back propogation method used in updating the weights of a multi-element neural network is a consequence of the steepest descent rule.It makes use of a technique in which the weghts are adjusted " in a direction opposite the instentanious error gradient".

Here the term instentanous error gradient refers to gradient in the *kth* step where :

$$\overline{\nabla}_k = \frac{\partial \varepsilon^2}{\partial \mathbf{W}_k} = \begin{bmatrix} \frac{\partial \varepsilon^2}{\partial w_{1k}} \\ \frac{\partial \varepsilon^2}{\partial w_{2k}} \\ . \\ . \\ . \\ \frac{\partial \varepsilon^2}{\partial w_{mk}} \end{bmatrix} \qquad (3.2.1)$$

The sum squred error ε_k^2 corresponds to sums of squares of all errors at the outputs of the network. If there are to be N_y outputs, then

$$\varepsilon_k^2 = \sum_{i=1}^{Ny} \varepsilon_{ki}^2 \qquad (3.2.2)$$

For a network with three outputs y_1, y_2, y_3 ,Fig.3.2. with respective desired responses of d_1, d_2 and d_3 then

$$\varepsilon_k^2 = (y_{1k} - d_{1k})^2 + (y_{2k} - d_{2k})^2 + (y_{3k} - d_{3k})^2 \qquad (3.2.3)$$

In the back propogation method the network produces an output for a training input vector. Then the error found between the desired output and the actual output of the network at that k*th* time step is used in generating a "square error derivative" δ for each adaline in the network. Then this δ is used in updating the weights of the adaline. By definition the δ that will be associated with the j_{th} Adaline of the l_{th} layer, is given as

$$\delta_j^{(l)} \stackrel{\Delta}{=} -\frac{1}{2}\frac{\partial \varepsilon^2}{\partial S_j^{(l)}} \qquad (3.2.4)$$

$S_j^{(l)}$ here corresponds to the linear output of the j_{th} Adaline of the l_{th} layer. $\delta_j^{(l)}$ can be considered as the sensitivity of the sum square

error to the change in $S_j^{(l)}$. For the case of three outputs the sum square errors term is ;

$$\varepsilon^2 = (y_1 - d_1)^2 + (y_2 - d_2)^2 + (y_3 - d_3)^2 \qquad (3.2.5)$$

If $y_1 = x_1^{(3)}$ the output of first adaline of layer-2 than the partial derivative of the sum square error ε^2 with respect to this output $y_1 = x_1^{(3)}$ may be stated as

$$\begin{aligned} &\frac{\partial \varepsilon^2}{\partial y_1} = \frac{\partial \varepsilon^2}{\partial x_1^{(3)}} = \frac{\partial}{\partial y_1}\left((d_1 - y_1)^2 + (d_2 - y_2)^2 + (d_3 - y_3)^2\right) \\ &= -2(d_1 - y_1) = -2\varepsilon_1 \ , \\ &\Rightarrow \ \varepsilon_1 = -\tfrac{1}{2}\frac{\partial \varepsilon^2}{\partial y_1} = -\tfrac{1}{2}\frac{\partial \varepsilon^2}{\partial x_1^{(3)}} \end{aligned} \qquad (3.2.6)$$

and this is simply the difference between the output and the desired response.

The partial derivative of sum square error with respect to the linear output of the Adaline may be written as :

$$\begin{aligned} &y_1 = \text{tgh}(S_1^{(3)}) \quad \Rightarrow \\ &\frac{\partial y_1}{\partial S_1^{(3)}} = 1 + \text{tgh}^2(S_1^{(3)}) \\ &= 1 + y_1^2, \\ &\frac{\partial \varepsilon^2}{\partial S_1^{(3)}} = \frac{\partial \varepsilon^2}{\partial y_1}\frac{\partial y_1}{\partial S_1^{(3)}} = -2(d_1 - y_1)(1 - y_1^2), \\ &\Rightarrow \ \delta_1^{(3)} \overset{\Delta}{=} -\tfrac{1}{2}\frac{\partial \varepsilon^2}{\partial S_1^{(3)}} \\ &\delta_1^{(3)} = \varepsilon_1(1 - y_1)^2 = \varepsilon_1^{(3)} \cdot \text{Sgm}'(S_1^{(3)}) \\ &\varepsilon_1 \equiv \varepsilon_1^{(3)} \end{aligned} \qquad (3.2.7)$$

In the back propogation technique for the derivative of aquare error sum, the linear output of the related adaline is required. For the three layer example considered, this derivative term is defined by Widrow as [22] :

$$\delta_1^{(2)} \stackrel{\Delta}{=} -\frac{1}{2}\frac{\partial \varepsilon^2}{\partial S_1^{(2)}} \tag{3.2.8}$$

The term $\varepsilon_1^{(2)}$ can be obtained by using the chain rule :

$$\varepsilon_1^{(2)} \stackrel{\Delta}{=} -\frac{1}{2}\frac{\partial \varepsilon^2}{\partial X_1^{(2)}} = -\frac{1}{2}\left(\frac{\partial \varepsilon^2}{\partial S_1^{(3)}}\frac{\partial S_1^{(3)}}{\partial X_1^{(2)}} + \frac{\partial \varepsilon^2}{\partial S_2^{(3)}}\frac{\partial S_2^{(3)}}{\partial X_1^{(2)}} + \frac{\partial \varepsilon^2}{\partial S_3^{(3)}}\frac{\partial S_3^{(3)}}{\partial X_1^{(2)}}\right) \tag{3.2.9}$$

when the definitions for $\delta_1^{(3)}$, $\delta_2^{(3)}$ and $\delta_3^{(3)}$ are used that is ;

$$\delta_1^{(3)} \stackrel{\Delta}{=} -\frac{1}{2}\frac{\partial \varepsilon^2}{\partial S_1^{(3)}} \quad , \quad \delta_2^{(3)} \stackrel{\Delta}{=} -\frac{1}{2}\frac{\partial \varepsilon^2}{\partial S_2^{(3)}} \text{ and } \delta_3^{(3)} \stackrel{\Delta}{=} -\frac{1}{2}\frac{\partial \varepsilon^2}{\partial S_3^{(3)}} ;$$

$$\text{the term } \varepsilon_1^{(2)} \stackrel{\Delta}{=} \delta_1^{(3)}\frac{\partial S_1^{(3)}}{\partial X_1^{(2)}} + \delta_2^{(3)}\frac{\partial S_2^{(3)}}{\partial X_1^{(2)}} + \delta_3^{(3)}\frac{\partial S_3^{(3)}}{\partial X_1^{(2)}} . \tag{3.2.10}$$

For the three layer sample (Figure 3.2.)

$$S_1^{(3)} = w_{10}^{(3)} + \sum_{i=1}^{3} w_{1i}^{(3)} \cdot X_i^{(2)} \quad ,$$

$$S_2^{(3)} = w_{20}^{(3)} + \sum_{i=1}^{3} w_{2i}^{(3)} \cdot X_i^{(2)} \quad ,$$

$$\text{and} \quad S_3^{(3)} = w_{30}^{(3)} + \sum_{i=1}^{3} w_{3i}^{(3)} \cdot X_i^{(2)} \tag{3.2.11}$$

The derivative term $\frac{\partial \text{Sgm}(S_i^{(2)})}{\partial S_j^{(2)}} = 0$ for all $i \neq j$

$$\varepsilon_1^{(2)} \stackrel{\Delta}{=} \delta_1^{(3)} w_{11}^{(3)} + \delta_2^{(3)} w_{21}^{(3)} + \delta_3^{(3)} w_{31}^{(3)} \tag{3.2.12}$$

$$\delta_1^{(2)} = \varepsilon_1^{(2)} \text{Sgm}'(S_1^{(2)})$$

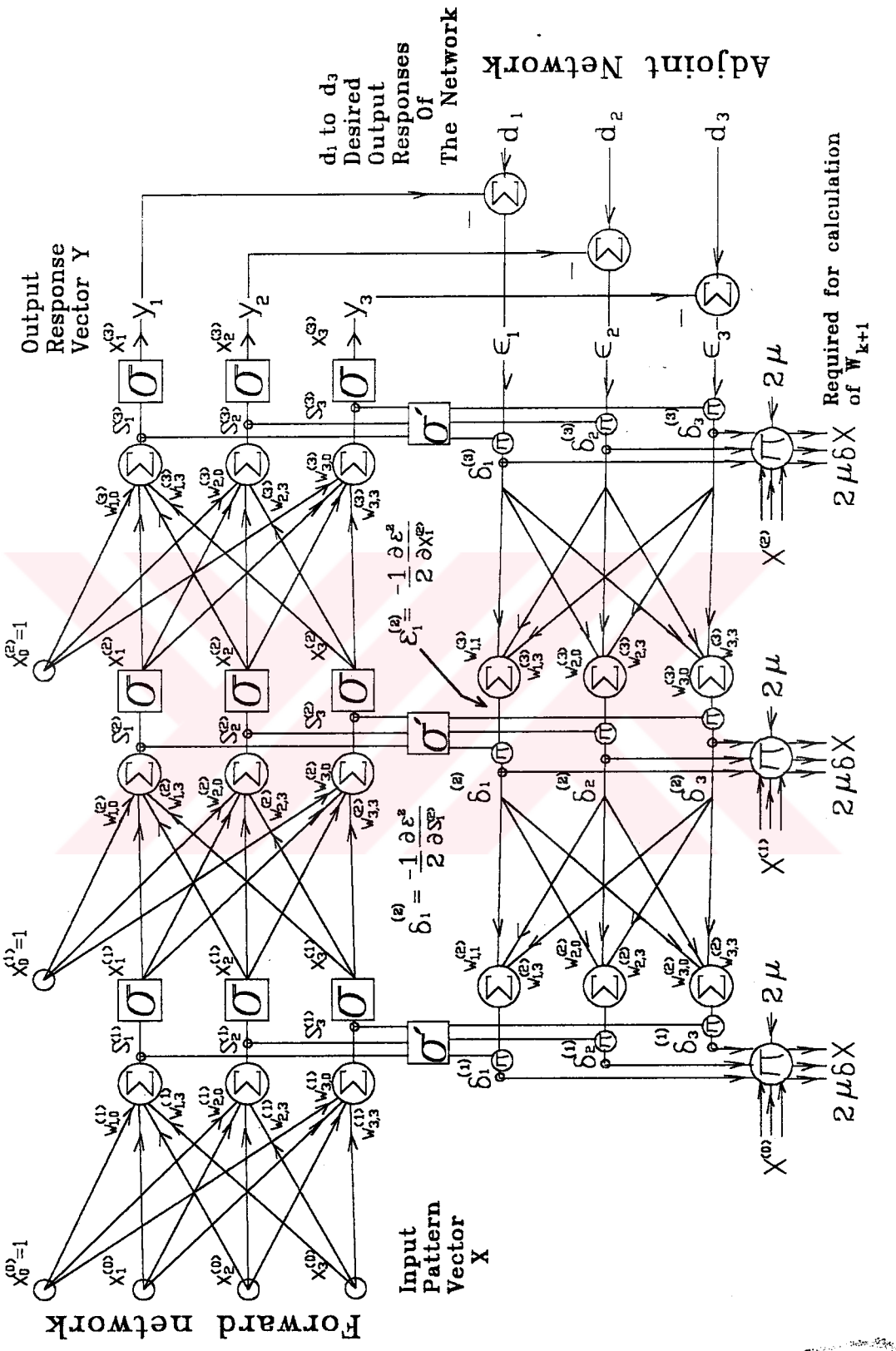


Figure 3.2 Three layer representation for Back-Propogation synthesized from Fig.25. of [22] and Fig.1. of [14]

The calculation of $\delta^{(l)}$ for an adaline in the *lth* layer involves a procedure in which ;all the $\delta^{(l+1)}$ of the following layer are multiplied by the weights connecting the given adaline in the lth layer to the *(l+1)*th layer .Therefore a weighted sum error of $\varepsilon^{(l)}$ is formed for the given adaline in the *lth* layer.Then afterwards $\varepsilon^{(l)}$ is multiplied by the derivative of sigmoid function taking the linear sum $S^{(l)}$ as argument.The δs produced for each adaline element in the network are then used for the calculation of instantaneous gradient $\overline{\nabla}_k$ here the k index refers to the *kth* instant in learning lifetime of an adaline.

$$\overline{\nabla}_k = \frac{\partial \varepsilon_k^2}{\partial \mathbf{W}_k} = \frac{\partial \varepsilon_k^2}{\partial S_k}\frac{\partial S_k}{\partial \mathbf{W}_k} \tag{3.2.13}$$

here the adaline is represented by its linear sum *Sk* where

$$S_k = W_k^T X_k \quad ; \tag{3.2.14}$$

Xk the input vector to the network at the *kth* instant. The input vector *Xk* and W_k are independent therefore

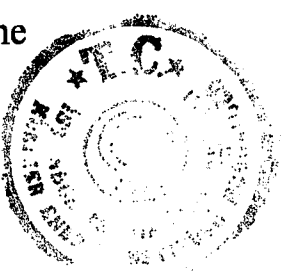
$$\begin{aligned}
\overline{\nabla}_k &= \frac{\partial \varepsilon_k^2}{\partial \mathbf{W}_k} = \frac{\partial \varepsilon_k^2}{\partial S_k}\frac{\partial S_k}{\partial \mathbf{W}_k} \\
&= \frac{\partial \varepsilon_k^2}{\partial S_k}\frac{\partial W_k^T X_k}{\partial W_k} \\
&= \frac{\partial \varepsilon_k^2}{\partial S_k} X_k \quad ; \quad \delta_k = -\tfrac{1}{2}\frac{\partial \varepsilon_k^2}{\partial S_k} \\
&\Rightarrow \overline{\nabla}_k = -2\delta_k X_k
\end{aligned} \tag{3.2.15}$$

(3.2.16)

therefore the steepest decent rule using the backpropogation of all "square error derivatives" to change the weights of the network is defined as

$$W_{k+1} = W_k + \mu(-\overline{\nabla}_k) = W_k + 2\mu\delta_k X_k \tag{3.2.17}$$

A momentum term is also included in the equation to reduce the effects of gradient noise in the weights.The inclusion of the



momentum term make use of the previous change of the weight at the *(k-1)th* step by a factor η, less then 1 as stated in the following equation.

$$\Delta W_k = 2\mu(1-\eta)\delta_k X_k + \eta\Delta W_{k-1}$$
$$W_{k+1} = W_k + \Delta W_k \qquad (3.2.18)$$

In this technique the momentum parameter η affects the proccess as low pass filtering of weight updates and therefore prevents erratic weight changes; in other words, oscillations in the learning curve are dampened, and approach to global minimum is smooth.Inclusion of the factor $(1-\eta)$ makes the DC gain of the filter 1 , so that η does not interfere with the choice of μ.

3.3 RECURRENT NEURAL NETWORKS.

A recurrent neural network transforms an input sequence into an output sequence, while using the contextual information.

The internal state of RNN can represent the context information, that is the information from the past inputs necessary to calculate the desired outputs. The interval for which the information is retained is not fixed. Static networks with delay-line inputs, have a finite impulse response, therefore can not store information for an indefinite time.

RNN can be trained in two different ways depending on the performance targeted [19], [20], namely adapting for a fixed point is a static problem, hereas adapting for a given trajectory is a dynamic problem.

The dynamics of a node of an RNN is given by an ordinary differential equation [11],[20],[23] ;

$$C_i \frac{dX_i}{dt} + \frac{X_i}{\rho_i} + \sum_j \frac{(X_i - Y_j)}{R_{ij}} \qquad i = 1, 2, \ldots N$$

$$\dot{X}_i = -\gamma_i X_i + \sum_j^N w_{ij} Y_j + I_i \quad ,$$

$$\text{where} \quad \gamma_i = \frac{1}{C_i}\left[\frac{1}{\rho_i} + \sum_j \frac{1}{R_{ij}}\right], \qquad (3.3.1)$$

$$w_{ij} = \frac{1}{C_i R_{ij}} \quad ,$$

$$Y_i = \sigma(X_i)$$

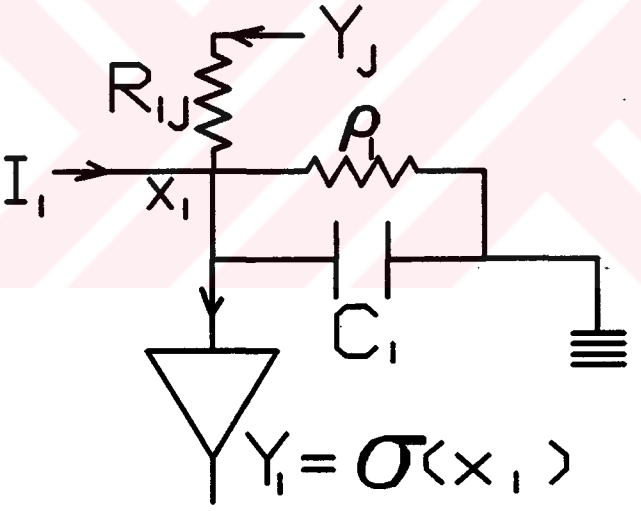


Figure 3.3. Physical realization of a RNN node.

where y_i is the activity or state of the ith neuron; I_i is the external input of the *ith* neuron;w_{ij} is the connection strength or weight from the *jth* neuron,

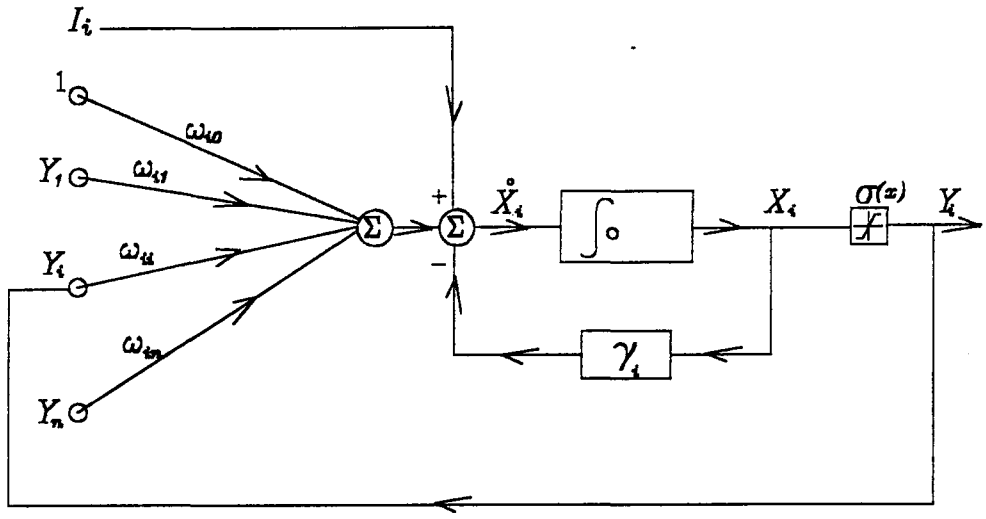


Figure 3.4. Typical processing element in a RNN (operational diagram) .

Fixed point $x_i$of the system is determined by solving the following set of algebraic equations :

$$\gamma x_i = \sum w_{ij}\sigma(x_j) + I_i \qquad i = 1,2,3\ldots N \tag{3.3.2}$$

Training the RNN is outside the scope of this work.

CHAPTER 4.

REACTIVITY ESTIMATION PROBLEM

4.1 FORMULATION OF THE PROBLEM

In the literature , there have been numerous attepts to continuously monitor the reactivity of a reactor, from the operating console, on line with a host computer [3],[5],[6],[7]. The estimation algorithms are based on the inverse kinetic equation[2],[10]. Formally, an instantaneous reactivity estimation is impossible, because the full state of the reactor at any instant of time has to be known exactly, which is not the case. In this work an attempt has been made to put artificial neural networks to use for the solution of this problem. The the network trained has a multilayer perceptron type architectutre. For a better working estimator, a recurrent neural network with dynamical capabilities is recommended. For the derivation of the inverse kinetic equation, we begin with the point kinetic equations [10].

$$\frac{dn(t)}{dt} = \frac{\rho(t) - \beta}{\Lambda} + \sum_{i}^{N} \lambda_i C_i(t) \qquad (4.1.1)$$

$$\frac{dC_i(t)}{dt} = \frac{\beta_i}{\Lambda} - \lambda_i C_i(t) \quad (i = 1,2,\ldots N) \qquad (4.1.2)$$

where

$n(t)$ is the neutron density,

$C_i(t)$ density of thedelayed neutron precursor of type i

λ_i dècay constant of the of the precursor of type i

β_i delayed neutron fraction of the ith precursor

β total delayed neutron fraction, $\beta = \sum_i \beta_i$

ρ(t) net reactivity (external-feedback)

Λ neutron generation time

Since $C_i(t)$ are unobservable state variables, we need a formula which directly relates ρ(t) to n(t). Integrate Eq. (4.1.2) with a set of suitable initial condition given at t=t to get :

$$C_i(t) = C_i(t_0)e^{-\lambda i(t-t_0)} + \int_{t_0}^{t} \frac{\beta_i}{\Lambda} n(t')e^{-\lambda i(t-t')}dt' \qquad (4.1.3)$$

Eq.(4.1.3), together with Eq.(4.1.1) will yield, with some rearrangement:

$$\rho(t) = \frac{\Lambda}{n(t)}\frac{dn(t)}{dt} + \beta - \frac{S_0(t)\Lambda}{n(t)} - \frac{S_d(t)\Lambda}{n(t)} \qquad (4.1.4)$$

where,

$$S_0(t) = \sum_{i=1}^{N} \lambda_i C_i(t_0)e^{-\lambda_i(t-t_0)} \qquad t - t_0 = \Delta t \qquad (4.1.5)$$

is the initial delayed source, which decays with time, and

$$S_d(t) = \frac{1}{\Lambda}\int_{t_0}^{t}\sum_{i=1}^{N} \lambda_i \beta_i n(t')e^{-\lambda_i(t-t')}dt' \qquad (4.1.6)$$

is the delayed source produced along the course of time. Using these sources we can strike a reactivity balance:

TABLE I.
Reactivity Balance, (4.1.4)

$\frac{\Lambda}{n(t)}\frac{dn(t)}{dt}$	the contribution of prompt neutrons,to reactivity
β	the contribution of delayed neutrons as if they where acting promptly
$-\frac{S_0(t)\Lambda}{n(t)}$	β is discounted due to the delayed source S_0 (t)
$-\frac{S_d(t)\Lambda}{n(t)}$	β is discounted due to the delayed source S_d (t)
$+$	$+$
$\rho(t)$	net reactivity

for further simplification, we make the usual assumption:

$$\lim_{t_0 \to -\infty} e^{\lambda_i t_0} C_i(t_0) = 0 \tag{4.1.7}$$

Therefore,

$$\rho(t) = \beta + \Lambda \frac{d \ln n(t)}{dt} - \beta \int_{-\infty}^{t} \frac{n(t')}{n(t)} D(t - t') dt' \tag{4.1.8}$$

where

$$D(t) \equiv \sum_{i=1}^{N} \frac{\beta_i}{\beta} \lambda_i e^{-\lambda_i t} \tag{4.1.9}$$

Where $D(t)dt$ is interpreted as the probability that a delayed neutron will be emitted in dt about t, following a fission event at t=0. Replacing the dummy integration variable t' by $u = t - t'$, Eq.(4.1.8) becomes

$$\rho(t) = \beta + \Lambda \frac{d \ln n(t)}{dt} - \beta \int_0^{\infty} \frac{n(t-u)}{n(t)} D(u) du \qquad (4.1.10)$$

This equation is known as the "inverse kinetic equation" for obvious reasons. It can be easily implemented to monitor the reactivity of a reactor on a continuous basis [3],[5],[6],[7]. This equation has an integro differential form, which requires the complete history of *n(t)* on the semiinfinite interval from t= -∞ to the present. If we inspect the graphical results presented by the previous workers, we can see that in the early stages of reactivity estimation, noise which results from the uncertainty in the past history of *n(t)*, is quite noticable. Now we take a twist and propose way to make use of Eq.(4.1.4) in place of the usual Eq.(4.1.8) for the estimation of the reactivity on a short order. At the moment when a sudden demand is made, the complete past history of *n(t)* might not be available. In Eq.(4.1.4), if we let $t = t_0$, we recover Eq.(4.1.1) again. Another possibility is to keep $t\text{-}t_0$ finite. Eq.(4.1.4) can be rearranged as:

$$\rho(t_0+\Delta t) - \beta - \frac{\Lambda}{n(t)} \frac{dn(t)}{dt}\bigg|_{t=t0+\Delta t} - \int_{t0}^{t0+\Delta t} \beta \frac{n(t')}{n(t_0+\Delta t)} D(t_0+\Delta t - t')dt' = -\Lambda \sum_{i=1}^{N} \frac{\lambda_i C_i(t_0) e^{-\lambda_i \Delta t}}{n(t_0+\Delta t)}$$

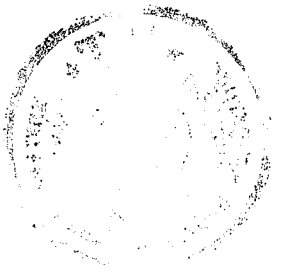
(4.1.11)

where Δt is the time allowed for the estimation to take place.

To summarize the results, we give the following table for three different choices of Δt:

TABLE II.
Choice of Observation Time

$\Delta t = t\text{-}t_0$	**Equation**
0	Point kinetic, Eq. (4.1.1-2).
∞	Inverse kinetic, Eq. (4.1.8)
finite	Eq. (4.1.11).



From the point of training an artificial neural network the above equation may offer a good starting point. A small but finite observation/estimation interval is advantageous over Eq.(4.1.1) because, firstly it helps filter out noise to some degree, and secondly it deemphasizes or reduces the the contribution of the initial state variables $C_i\,(t)$ by a factor $e^{-\lambda_i \Delta t}$.

4.2 REACTOR DYNAMICS WITH REACTIVITY FEEDBACK

The point kinetic equations of a nuclear reactor are supplemented with thermal response in order to give a dynamic picture of the process. The thermal response mainly contributes to point kinetics through a feedback effect in the reactivity, Figure 4.1.

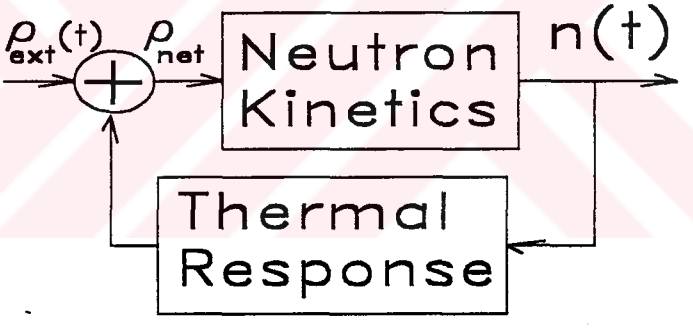


Figure 4.1. Inherent reactivity feedback loop of the reactor.

In general the temperature reactivity feedback is mainly caused by changes in neutron slowing down , resonance escape, and the dimensions of the core. In a TRIGA core the contribution of the former is prompt and dominant arising from the special composition of the fuel elements, 20% enriched U-235 and zirconium hydride, which makes it suitable for pulsing,[10].

Neglecting the changes in the water temperature, the thermal response can be accounted for in terms of an additional state variable, namely $T_f(t)$, fuel temperature:

$$M\frac{dT_f}{dt} = \mu n(t) - H(T_f - T_w) \tag{4.2.1}$$

and the reactivity feedback is:

$$\rho_{fb}(t) = \alpha(T_f - T_0) \tag{4.2.2}$$

where ,

M : thermal capacity of the core, (W.s/K)
H : average convective heat transfer coefficient, (W/K)
α : prompt temperature coefficient of reactivity ,(∇k/k /K)
μ n (t) : thermal power released in the fuel elements, (W)
T_w : pool water temperature,assumed constant (K)
T_0 : referece fuel temperature.

Thermal feedback model parameters are determined emprically.

We observe that, the inclusion of the thermal response of the reactor only modifies a parameter of the point kinetic equations. Point kinetic equations remain to be valid with the replacement of $\rho(t)$ by $\rho_{ext}+\rho_{fb}$. We expect the trained network to reproduce on demand the net reactivity at time t, when it is presented with a set of sampled values of n(t) in the next Δt. In Figure 4.3, a delay line supplies N delayed values of n(t), with a sampling interval of h. Another possibility is that, the left hand side of Eq. (4.2.11) can be calculated from the given time history of n(t), Figure 4.4. In order to improve the signal to noise ratio of the derivative, a higher order difference formula is recommended, for example a six point backward difference formula is given as Ref.[1], p.914:

$$\left.\frac{dn}{dt}\right|_{t=t_0+5h} = \frac{1}{120h}\left[-24n(t_0)+150n(t_0+h)-400n(t_0+2h)+600n(t_0+3h)-600n(t_0+4h)+274n(t_0+5h)\right] \tag{4.2.3}$$

The remaining integral term can also be carried out numerically, on line.

A more straightforward approach is to feed *n(t)* to a delay line at the input of a multilayer perceptron and expect the network to learn to estimate directly the reactivity in a short sampling interval However, this task puts too much burden on the feedforward netrwork, which is good at static pattern learning problems. Our problem is of dynamic nature, and besides, the system it is supposed to learn and mimic is stiff.

4.3 TECHNICAL ASPECTS OF THE NEURAL NETWORK

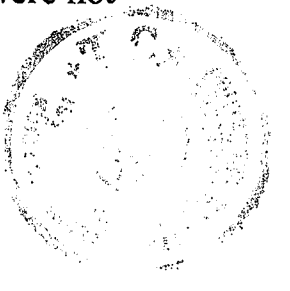
The network, which is actually trained, consists of 3 types of layer, namely, an input, hidden inner, and an output.

Training is affected by the error backpropagation algorithm, as described in Chapter II. A listing of FORTRAN code ANN is given in the Appendix A. A generic structure of the ANN with a description of its input and output files is given in Figure 4.2.

4.4. EXPERIMENTAL RESULTS

The training and testing data sets are produced by a program which simulates TRIGA dynamics with reasonable accuracy [7]. A control rod is actuated from the keyboard by an operator, and the resulting responses including, reactor power, fuel temperature, water temperature, feedback reactivity, net reactivity and the delayed neutron precursor concentrations are generated and stored in a file for future use. Testing data set is used to generate learning curves to see how well the neural network is able to generalize what it has been already taught. We proposed and trained three different schemes as shown in Figures 4.3. to 4.5 . The sampling interval is h=0.1 sec.

Scheme 1. Delayed samples of power are fed in, and the desired response is the delayed source term. This scheme turned out to be a hard learning problem, and satisfactory results were not obtained after reasonable training periods (Figure 4.3.).



Scheme 2. Delayed samples of power are fed in, and the desired response is the net reactivity. This scheme also turned out to be a hard learning problem (Figure 4.4.).

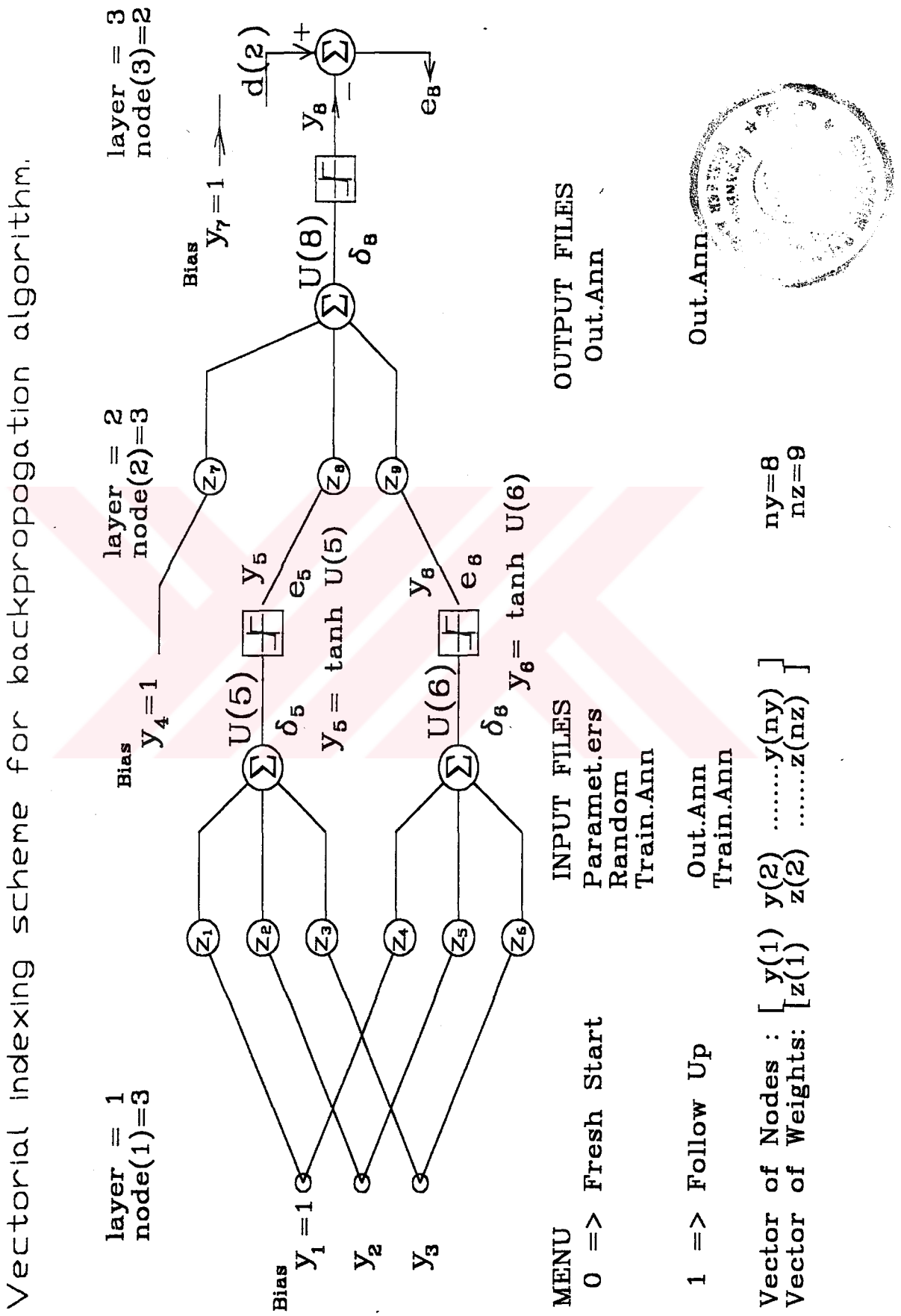


Figure 4.2. Vectorial indexing scheme for ANN.FOR
's for the

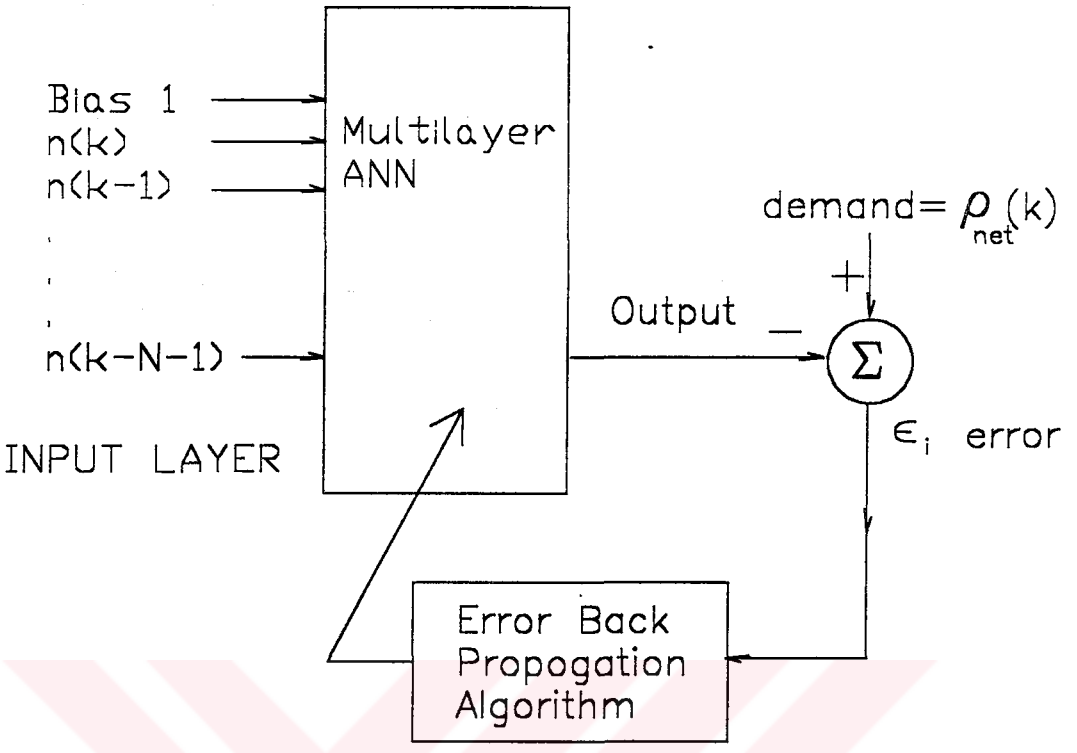


Figure 4.3. Proposed scheme #1. Hard learning problem.

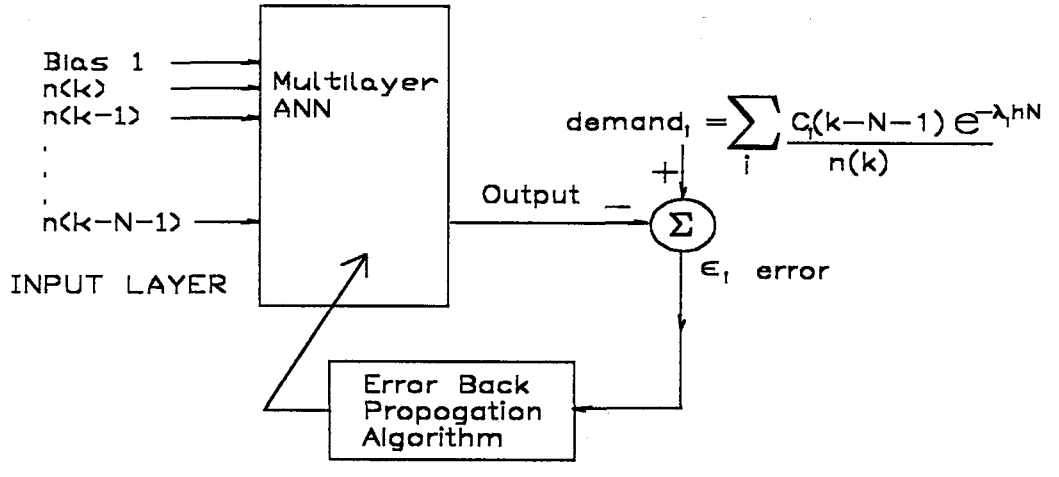
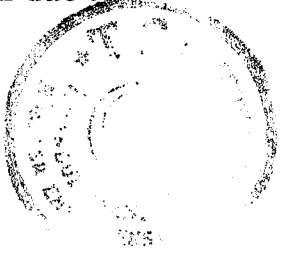


Figure 4.4. Proposed scheme #2.Estimation of delayed term due to initial neutron precursor concentrations.



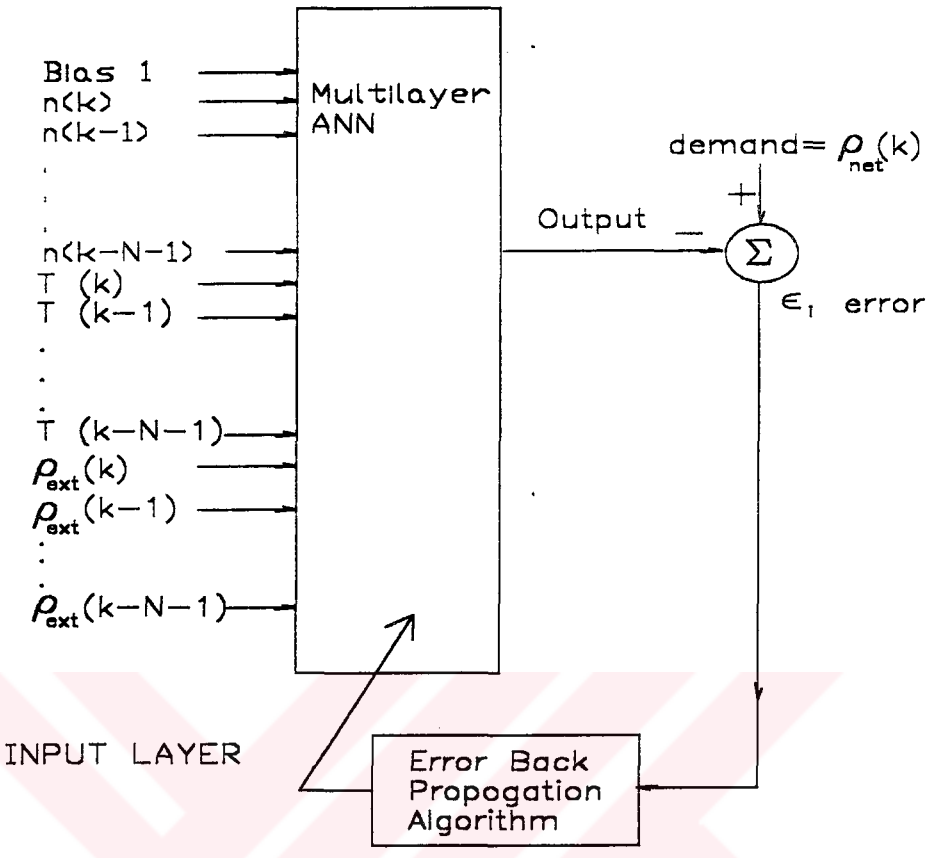


Figure 4.5. Proposed scheme #3.Easy learning problem.

Scheme 3. Delayed samples of power, fuel temperature, and external reactivity insertion are fed in. The desired response is the net reactivity. This scheme produced satisfactory results within a reasonable training period. The results are plotted in Figure 4.6.

For all of the topologies given in figures 4.3 to 4.5 the networks with the following parameters are trained :

N=9 , t=0.15 , node(1)=28 , node(2)=70 , node(3)=2 .

For nomenclature refer to figure 4.2.

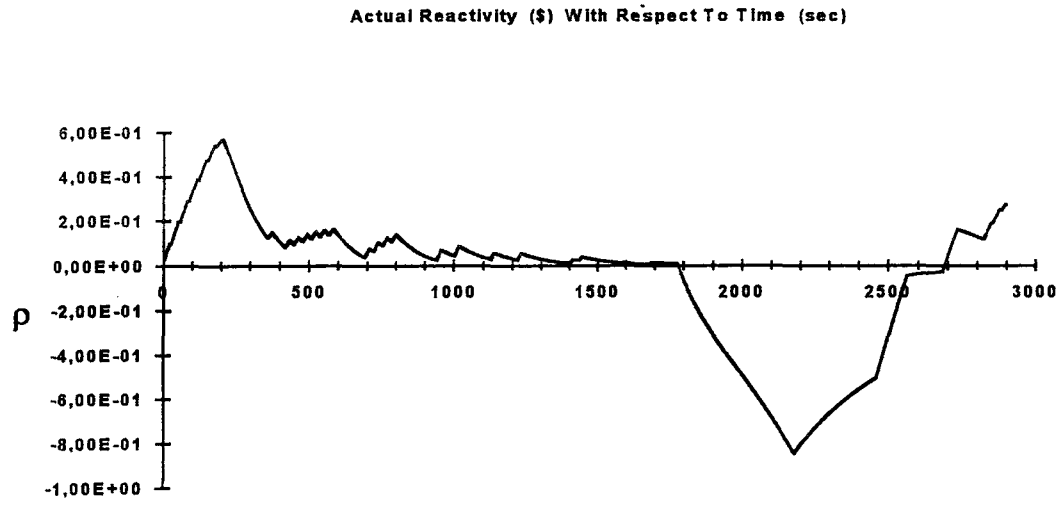


Figure 4.6.a. The desired response : Variation of net reactivity as a function of time during the course of a particular run. Data set #1.

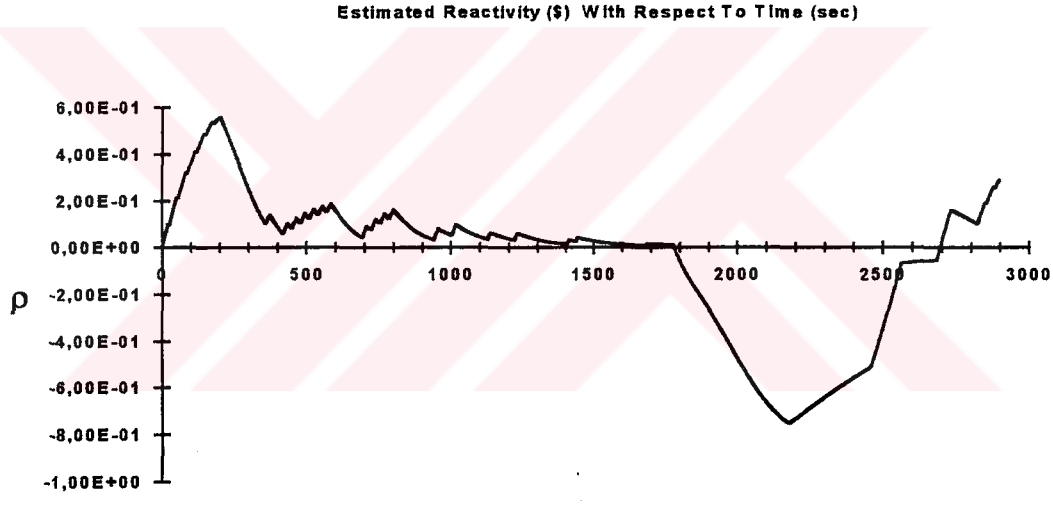


Figure 4.6.b. The net reactivity estimated by an already trained neural network for the same run. Data set #1.

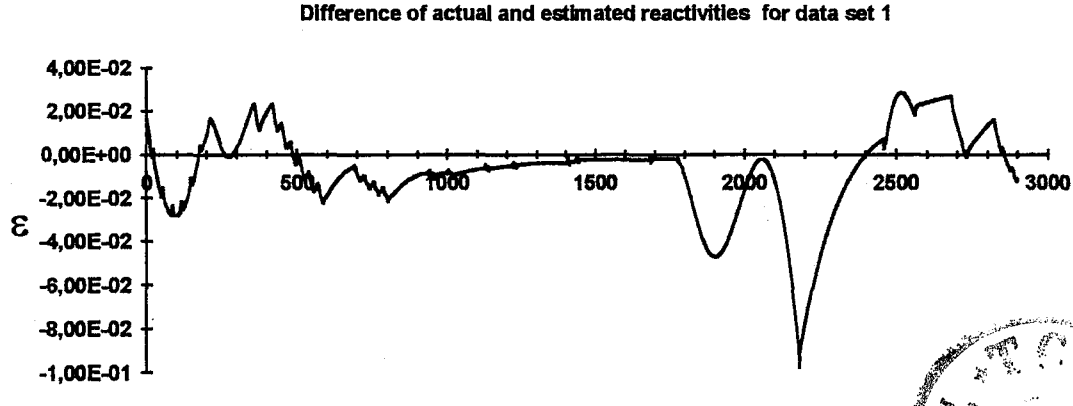


Figure 4.6.c. The performance check of the trained neural network. Data set #1 .

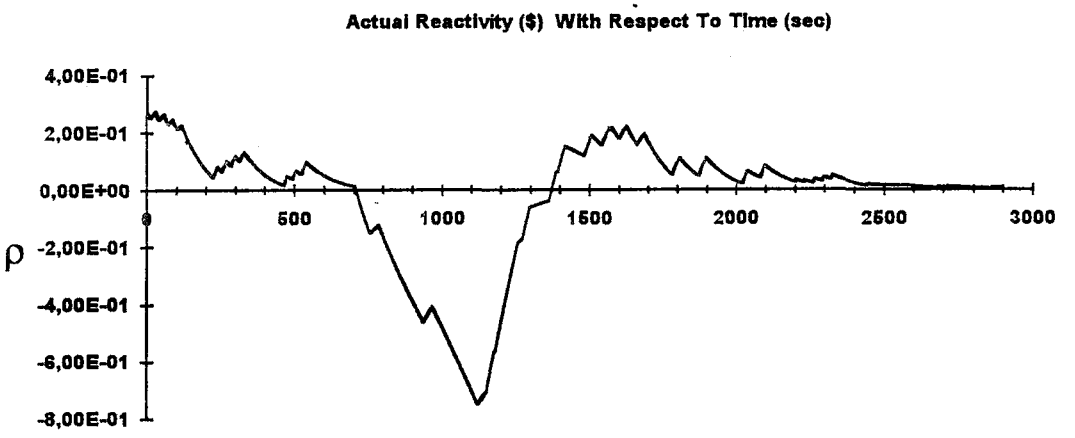


Figure 4.6.d. The desired response : Variation of net reactivity as a function of time during the course of a particular run. Data set #2.

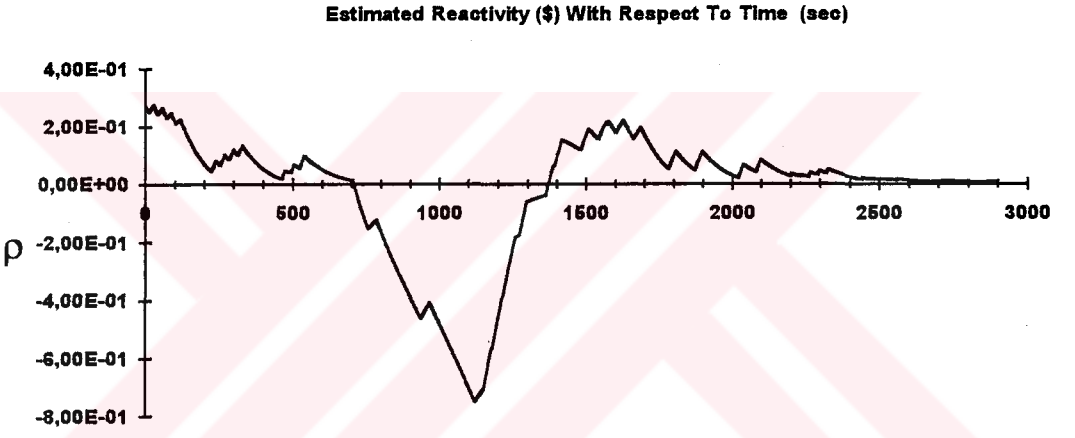


Figure 4.6.e. The net reactivity estimated by an already trained neural network for the same run. Data set #2.

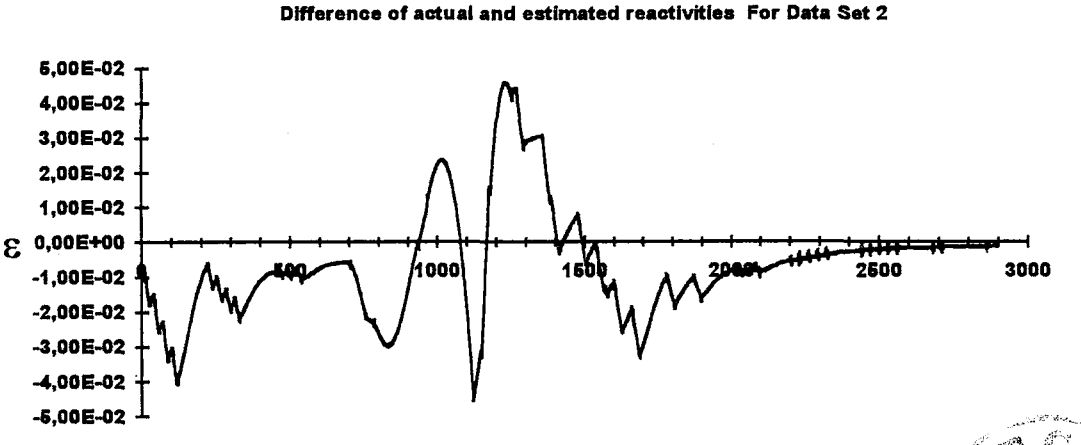
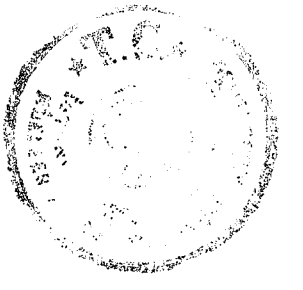


Figure 4.6.f. The performance check of the trained neural network. Data set #2.



CHAPTER 5.

DISCUSSION OF THE RESULTS AND CONCLUSIONS

An attempt has been made to estimate on-line the net reactivity of a reactor core, making use of artificial neural networks, specifically, feedforward multilayer structure built with perceptron type processing elements, which give way to backpropagation training. The sampled data needed for training and testing is generated from a simple dynamic simulation of TRIGA core.

A computer code for training of multilayer feed forward neural networks by back propagation of the square of the target error is written based on the algorithm of Ref. [22], with a slight modification to use the available memory efficiently; all of the nodal values, including the input and fixed bias nodes are mapped into a single vector, and all of the synaptic weights are mapped into another vector, so that one can spread the size of the structure nodewise on the layers as well as layerwise across the network, without overtaxing the memory The training session may initiate with a random selection of synaptic weight, the structure of the ANN being read in from a file, or it may follow-up with the synaptic weights of an already, but judged to be incompletely trained ANN.

Out of several ANN schemes proposed, only one was able to learn its task. As shown on Fig. 4.5 the input of the ANN consists of 9 samples from each of the three observable reactor parameters, namely reactor power, average fuel temperature, and control rod position, Sampling interval is 0.1 sec. The input data is delayed with respect to the estimation instant of the reactivity.

The results obtained with an set of data independent from that used for training, are given in Fig 4.6. The maximum absolute error in the estimated reactivity is less than 2 cents. This scheme turned out to be an easy learning problem, just because there is a simple static relation between the net reactivity, fuel temerature and control rod position. Furthermore, the input data is

redundant: but this feature is expected to make the estimator robust against noise, and also immune to complete failure of one of the channels for a short period of time.

As reported in the earlier literature, [8], [19], state estimation or dynamic simulation problems require recurrent network architecture, especially when all of the state variables are non-observable, for several reasons; each node represents an ordinary differential equation coupled with other nodes, or to put it another way, the recurrent connections provide an inherent memory to hold an infinite number of delayed samples of the input signals. Feedforward structures are best at capturing static relations. To conclude, an RNN topology is suggested in Fig.5.1 for an economic and short cut and neat solution of the problem.

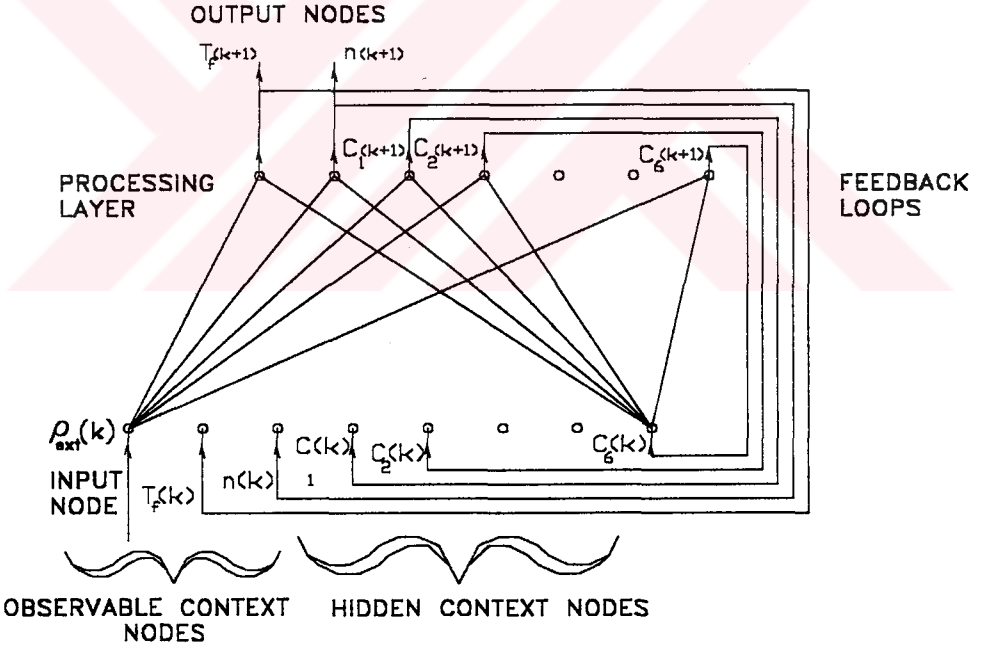


Figure 5.1. A proposed RNN architecture for our problem after references [8] and [18] .

REFERENCES

[1]. Abromowitz, M., and Stegun, I.A., eds., "Handbook of Mathematical Functions", Dover (1965).

[2]. Akçasu, A.Z. , Lellouche,G.S., and Shotkin, L.M.,"Mathematical Methods in Nuclear Reactor Dynamics", Associated Press, (1971).

[3]. Ansari, S. A., "Development of On-Line Reactivity Meter for Nuclear Reactors", IEEE Trans. Nuclear Science, Vol. 38, 946, (1991).

[4]. Arul, A.J., "Reactivity Surveillance in a Nuclear Reactor by Using a Layered Neural Network", Nuclear Science and Engineering, Vol.117, pp. 186-193 (1994).

[5]. Çiftçioğlu, O., and Geçkinli, M., " A CAMAC Based Reactivity Meter for Nuclear Reactors", Nuclear Instruments and Methods, Vol. 177,pp. 321-326 (1980).

[6]. Efimov, V.N., Eschenko, S.N., Kachalin,V.A., Nikol, A.S., and Pridachin, V.N., "System of Computation of the Reactivity Balance for the BOR-60 Reactor", Atomnaya Energiya, Vol. 59, pp 379-381, (1985).

[7]. Geçkinli, M., Hızal, N.A., Gencay, Ş., Çiftçioğlu, O., Can, B., and Güngördü,E., " Nonlinear Dynamics of İTU TRIGA Reactor", İTU Research Projet Final Report (July 1989).

[8]. Hecht-Nielsen, R., "Neurocomputing", Addison-Wesley (1989).

[9]. Hertz, J., Krogh, A., and Palmer, R.G., "Introduction to the Theory of Neural Computation",Leture Notes Colume i. Addison-Wesley (1991)

[10]. Hetrick, D. L. , "Dynamics of Nuclear Reactors" , The University of Chicago Press (1971).

[11]. Hopfield, J.J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", Proc. Natl. Acad. Sci. USA, Vol. 79, pp. 2554-2558 (1982).

[12]. Ku., C.-C., Lee., K.Y., and Edwards, R.M., "Improved Nuclear Reactor Temperature Control Using Diagonal Recurrent Neural Networks", IEEE Transactions on Nuclear Sciece, Vol.39, 2298 (1992).

[13]. Ku, C.C.,K.Y. Lee, Edwards, R.M., "D iagonal recurrent neural networks for nuclear power plant control", The 8th power plant dynamics, control and testing symposium, Vol.II, pp.61.01-14 Knoxville May. 1992.

[14]. Narendra, K.S., and Parthasarathy, K.,"Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks", IEEE Trans. Neural Networks, Vol.2, No.2, 252 (1991).

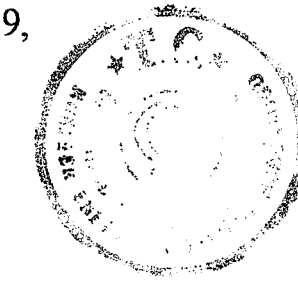
[15]. Nerrand, O., Roussel-Ragot, P., Urbani, D., Personnaz, L. and Dreyfus, G.,"Training Recurrent Neural Networks: Why and How? An Illustration in Dynamic Modeling", IEEE Transactions on Neural Networks, Vol. 5, 178 (1994).

[16]. Ott, K.O., and Neuhold, R.J., "Introductory Nuclear Reactor Dynamics", ANS (1985).

[17]. Parlos, G. Alexander, Chong and Atiya, " Emprical Model Development and Validation With Dynamic Learning in the Recurrent Multilayer Perceptron",Nucl. Tech., Vol. 105, p.271, (1994).

[18] Pham, D.T., and Liu, X., "Identification of linear and nonlinear dynamic systems using recurrent neural networks", AI in Engineering, Vol. 8, pp. 67-75 (1993)

[19]. Pineda, F.J., "Generalization of Back-Propagation to Recurrent Neural Networks", Phys. Rev. Lett., Vol. 59, 2229, (1987).

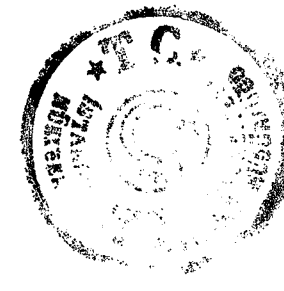


[20]. Sato, M., " A Real Time Learning Algorithm for recurrent Anolog Neural Networks ", Biol. Cybern., Vol. 62, 237-241 (1990).

[21]. Uhrig, R. E., "Potential Applications of Neural Networks to the Operation of Nuclear Plants", Nuclear Safety, Vol. 322, p.68 (1991).

[22]. Widrow, and Lehr, B., M.A., "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation", Proceedings of the IEEE, Vol. 78, 1415 (1990).

[23]. You, Y., and Nikolau, M., " Dynamic Modeling with Recurrent Neural Networks", AIChE J., Vol. 39, 1654 (1993).

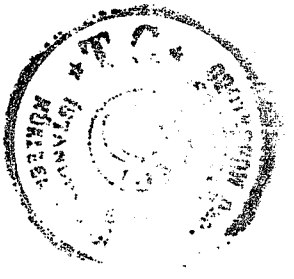


APPENDIX

```
      program ann

c                  revised 94-11-16
c                  *VECTOR VERSION*
c
c           modular programs for nonlinear/linear network training by
c     backward propagation  method. the  nomenclature used  is in
c     accordance with B.WIDROW & LEHR Proc. IEEE, Vol. 78,
c     p.1433,
c     Fig.25. main program prepares  the network for a  pattern
c     training
c     session then calls subroutine nfw to evaluate the response  vector
c     of the network and subroutine  nbw to backpropagate the error
c     for  adapting all of the weights of the network.
c     z(k):weight; u(i):linear output of  node i; y(i):nonlinear output.
c     file #1  PARAMET.ERS  : input data for fresh start-up
c     file #2  TRAIN.ANN    : input file includes training data
c     file #3  OUT.ANN      : network parameters+weights for future
c     use
c     file #4  RANDOM       : a set of random numbers on [-0.5,+0.5]
      dimension ydat(4,5000), dmdat(4,500)
      common  y(999),u(999),node(10),dm(10),z0(9999),z(9999),
     *layers,alpha,epsl(999),dlt(999),ny,nz,nsize(10),xmu,wmu
c     file paramet.ers includes the network parameters
c     alpha,mu,layers,node(i)
c     opening menu
      write(*,*) 'STATUS:training   fresh start/follow-up          '
      write(*,*) 'ENTER :...          0       /   else '
      write(*,*)
      read(*,*) status
      if(status.eq.0) then
c     *****input parameters of the network are read in    **********
      open(1,file='paramet.ers',status='old')
      do 190 i=1,6
190   read(1,*)
      read(1,1) alpha
```

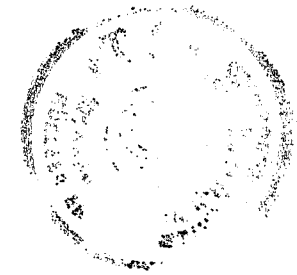
```
      read(1,1) wmu
      read(1,2) layers
      write(*,*)'          layer          # of nodes'
      do 200 i=1,layers
      read(1,2) node(i)
  200 write(*,*)i,node(i)
      close(1,status='keep')
      call sizing(layers,ny,nz,node,nsize)
    1 format(15x,d19.12)
    2 format(15x,i5)
c     initialize weight vector to begin a pattern training session
      open(unit=4,file='random',status='OLD')
      do 240 k=1,nz
      read(4,*)z(k)
  240 z0(k)=z(k)
      close(4,status='keep')
      else
      write(*,*)'enter training rate'
      read(*,*) alpha
      write(*,*)'enter momentum parameter'
      read(*,*) wmu
      open(3,file='out.ann',status='OLD')
      read(3,*) layers
      write(*,*)'          layer          # of nodes'
      do 250 i=1,layers
      read(3,*) node(i)
  250 write(*,*)i,node(i)
      call sizing(layers,ny,nz,node,nsize)
      do 260 k=1,nz
      read(3,*)mk,z(k)
  260 z0(k)=z(k)
      close(3,status='keep')
      endif
c     all initial nodal values are set to 1, so that some of them will
c     remain unchanged and act as  bias.
      do 270 j=1,ny
  270 y(j)=1.
      write(*,*)'# of cycles'
      write(*,*)
```



```
      read(*,*)icycle
      write(*,*)'# of iterations in every cycle '
      write(*,*)
      read(*,*)iters
c     initialize input vector to begin a pattern training session******
      open(unit=2,file='train.ann',status='OLD')
c     input and demand data is read in
      do 525 k=1,iters
      read(2,*) ik,(ydat(j,k),j=2,node(1)),(dmdat(i,k),i=2,
     *node(layers))
525 continue
      close(2,status='keep')
      do 526 k=1,iters
      write(*,*)  k,(ydat(j,k),j=2,node(1)),(dmdat(i,k),i=2,
     *node(layers))
526 continue
      xmu=1.0-wmu
c     training begins
*********************************************
      itally=0
      do 300 ic=1,icycle
      do 310 it=1,iters
c     data is assigned to y(j) on the input layer, from file 'train.ann'
      do 545 j=2,node(1)
545 y(j)=ydat(j,it)
c     data is assigned to demand vector dm(j)
      do 555 j=2,node(layers)
555 dm(j)=dmdat(j,it)
      call nfw
c     the following write statement is meant for this example
      write(*,*)ic,it ,y(2),y(3),y(ny),dm(2)
      call nbw
310 CONTINUE
      if(itally.eq.4) then
c     output file is updatedat every  fifth iteration,
c     for retraining runs, and against mishaps.
      open(3,file='out.ann',status='UNKNOWN')
      rewind 3
      write(3,*) layers
```

```
      do 340 i=1,layers
340 write(3,*) node(i)
      do 350 k=1,nz
350 write(3,*)k,z(k)
      close(3,status='keep')
      itally=0
      else
      itally=itally+1
      endif
300 continue
      open(3,file='out.ann',status='UNKNOWN')
      rewind 3
      write(3,*) layers
      do 360 i=1,layers
360 write(3,*) node(i)
      do 370 k=1,nz
370 write(3,*)k,z(k)
      close(3,status='keep')
      stop
      end

      subroutine nfw
c     forward evaluation
      common  y(999),u(999),node(10),dm(10),z0(9999),z(9999),
     *layers,alpha,epsl(999),dlt(999),ny,nz,nsize(10),xmu,wmu,
     *tstdat(4,5000),ndelay,ntest
      kw=0
      do 100 l=2,layers
      kx1=0
      do 20 m=1,l-2
 20 kx1=kx1+node(m)
      do 110 k=2,node(l)
           sum=0.0
            do 120 j=1,node(l-1)
               kw=kw+1
               kx=kx1+j
120          sum=sum+z(kw)*y(kx)
      ks=kx1+node(l-1)+k
      u(ks)=sum
```



```
      y(ks)=tanh(sum)
110     continue
100 continue

      return
      end
```

subroutine nbw

```
c     w weights are updated during backward pass
      common  y(999),u(999),node(10),dm(10),z0(9999),z(9999),
     *layers,alpha,epsl(999),dlt(999),ny,nz,nsize(10),xmu,wmu,
     *tstdat(4,5000),ndelay,ntest
c     now determine the error vector @ the output layer.
      kd=1
      nn=ny-(node(layers)-2)
      do 6 i=nn,ny
      kd=kd+1
      epsl(i)=dm(kd)-y(i)
  6 dlt(i)=epsl(i)*(1-y(i)**2)
      do 120 l=layers-1,1,-1
      nw1=0
      do 15 m=2,l
 15 nw1=nw1+nsize(m)
      nx1=0
      do 20 m=1,(l-1)
 20 nx1=nx1+node(m)
      nd1=nx1+node(l)
      do 140 j=2,node(l)
      sum=0.0
      nw2=nw1+j
      nx=nx1+j
      do 160 k=2,node(l+1)
      nw=nw2+(k-2)*(node(l))
      nd=nd1+k
160 sum=sum+dlt(nd)*z(nw)
      epsl(nx)=sum
140 dlt(nx)=epsl(nx)*(1-y(nx)**2)
120 continue
      nw=0
```

```
        do 200 l=2,layers
         nx1=0
          do 40 m=1,(l-2)
 40         nx1=nx1+node(m)
        do 200 k=2,node(l)
        sum=0.
        do 200 j=1,node(l-1)
     nw=nw+1
     nx=nx1+j
     nd=nx1+node(l-1)+k
     z2=z(nw)+alpha*(xmu)*y(nx)*dlt(nd)+wmu*(z(nw)-z0(nw))
     z0(nw)=z(nw)
200 z(nw)=z2
     return
     end

     subroutine sizing  (layers,ny,nz,node,nsize)
c    determines  the dimensions of the network vectors y and z
c    for a given network structure.
     dimension node(1),nsize(1)
c    ny : dimension of 'y'
     ny=0
     do 10  l=1,layers
 10  ny=ny+node(l)
     do  20 l=2,layers
 20  nsize(l)=(node(l)-1)*node(l-1)
c    nz : dimension of 'z' vector
     nz=0
     do  30 l=2,layers
30   nz=nz+nsize(l)
     do  40 l=2,layers
 40  write(*,*)'l,nsize(l)',l,nsize(l)
     write(*,*)'total # of nodes including bias nodes,ny',ny
     write(*,*)'total # of weights .....,nz',nz
     return
     end
```

The Inputs to The Network are read from the file :TRAIN.RNN

Input Sequence	Input X_1	Input X_2	Output Y
1	-0.25	0.75	0.5
2	-0.5	0.5	0.0
3	-0.1	0.0	-0.1
4	0.0	0.1	0.1
5	0.1	0.2	0.3

The Output of the program (Output file is OUT.RNN).

Value In File	Meaning
3	layers
3	node(1)
3	node(2)
2	node(3)
1 0.444570	*i* and *z(i)*
2 -0.202955	
3 0.138376	
4 -0.332015	
5 0.329726	
6 -0.513570	
7 -0.108218	
8 0.366010	
9 -0.202895	

The following parameters are read from the file PARAMET.ERS

VARIABLE NAME	ASSIGNED VALUE		DESCRIPTION
examples:	+3.1 255	d+01	format (15x,d19.12) format (15x,i5)
alpha	1.00	d-01	training rate
mu	1.00	d-01	momentum factor
layers	3		# of layers
node(1)	3		# of input nodes
node(2)	3		# of nodes in the hidden layer
node(3)	2		# of nodes in the output layer

The menu of the program ANN.FOR

STATUS: training fresh start / follow up
ENTER : 0 / else
1
enter training rate 0.1

enter momentum parameter 0.1

layer	# of nodes
1	3
2	3
3	2

l, nsize(l) 2 6
l, nsize(l) 3 3

total # of nodes including bias nodes ..... n_y
total # of weights ..... n_z 9

# of cycles
100
# of iterations in every cycle
0

BIOGRAPHY

B.Ali KÖKSAL was born in Tarsus in 1961. He was graduated from Tarsus American Lycee in 1979. He was attended to Control and Computer Department of Istanbul Technical University in 1982. He received his B.Sc. degree in Control and Computer Engineering in 1988. He is currently working in Saykon Digital Systems, one of the firms taking part in ITU-SMIDO Istanbul Technology Development Center.