A NEURAL NETWORK REGULATOR FOR TURBO GENERATOR

PROJECT REPORT

Submitted by

K. PREMKUMAR
J. CHARLES AROKIARAJ
N. MAHESH KUMAR
S. VANI LAKSHMI

Guided by

Mr. R. HARIHARAN, M.E.,
M.I.S.T.E., Lecturer, EEE

Submitted in partial fulfilment of the requirements
for the award of the Degree of
BACHELOR OF ENGINEERING
IN ELECTRICAL AND ELECTRONICS ENGINEERING
of the Bharathiar University, Coimbatore.

1997 - 1998

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE - 641 006

Department of Electrical and Electronics Engineering

Kumaraguru College of Technology

Coimbatore - 641 006

CERTIFICATE

This is to certify that the report entitled

A NEURAL NETWORK REGULATOR FOR TURBO GENERATOR

has been submitted by

K. PremKumar, J. Charles Arokiaraj, N. Maheshkumar, [illegible]

Roll Number: 94EEE29, 94EEE08, 94EEE19, 94EEE[illegible]

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in Electrical and Electronics Engineering

Branch of Bharathiar University during the academic year 1997-98

[illegible] 6/4/98

Guide

[illegible]

Dr. K. A. PALANISWAMY

Professor and Head

Department of Electrical and Electronics Engg.

Kumaraguru College of Technology

Coimbatore 641 006

Professor and Head

Certified that the candidate with University register number ....

was examined in Project work Viva-Voce Examination held on ....

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

It is with deep sense of profound gratitude we express our heartful thanks to our beloved guide **Mr.R.HARIHARAN, M.E.,** Department of Electrical and Electronics Engineering, whose inspiration, unfailing enthusiasm, invaluable suggestion, the keen constant interest by him throughout the course of this project and endless support helped us to work in a dedicated and consistent manner and complete this project successfully.

We are highly indebted to **Dr. K.A. PALANISWAMY, B.E., MSc (Engg)., Ph.D., M.I.S.T.E., C.Eng.(I), FIE.,** Head of the Department, Electrical and Electronics Engineering for his encouragement which made the effective completion of this project a reality.

We thank our Principal , **Dr. S. SUBRAMANIAN B.E., MSc (Engg)., Ph.D., S.M .IEEE,** for his kind patronage and for his constant encouragement.

Our sincere thanks are due to all the Members of the **STAFF,** Department of Electrical and Electronics Engineering and all the friends for providing us immense help and kind co-operation to carry out this project.

[illegible]

SYNOPSIS

Ever increasing demand of electrical power has encouraged us to install large number of turbo generators with large capacity. These non linear, multivariable turbo generators are subjected to variety of operating conditions. Regulators are used to maintain the terminal voltage of the generator and speed of the set at almost constant values during steady operating conditions corresponding to specified reference levels.

Most turbo generator adaptive controllers which are in practice now are based on linear system models, however the industrial systems are non linear. And also these are subjected to the problems of unequal number system inputs and outputs. This complicates the design of adaptive controllers and degrades the practical robustness of designed controllers.

This project explores the links between the fields of control science and neural networks .Neural network have greatest promise in the realm of non linear control problems. This project presents a neural network based regulator for non linear, fast acting, multivariable turbo generator control.

CONTENTS

CHAPTER-I

INTRODUCTION

Recent development in power system, arising from advances in technology, changes in patterns of generation and load, and economic consideration have exposed various problems and requirements. Large power station have been constructed in locations remote from load centres, with consequent danger of dynamic instability under certain circumstances. Increasing demand has encouraged the power system engineers not only to increase the generation but also to increase the reliability of the power system. Turbo generator forms the heart of the power system . Successful operation of the power system entirely depends on turbo generator.

The turbo generator is highly non linear, fast acting multivariable system which operates over a wide range of operating conditions , from lagging to leading phase, and are subjected to different types of disturbances like

- Changes in terminal voltage
- Changes in tap positions of transformer
- Change in inputs to governor
- Change in inputs to excitation systems
- Short circuits on transmission line

Under these circumstances the behaviour of the system is highly non linear but the outputs have to be co-ordinated so as to satisfy the requirements of the power system.

Basic function of regulators is to maintain the terminal voltage of the generator and speed of the set at almost constant values during steady operating conditions corresponding to specified reference levels. Modern generators have high per unit values of reactances, and rely on increasingly complex control systems for their successful operation. However with the advent of advanced computer technology and the development of mathematical tools and the modern control theory, realisation of effective computer controllers for turbo generators has become possible.

Most turbo generator adaptive controllers which are in practice now are based on linear system models, however the industrial systems are non linear . And hence these are subjected to the problems of unequal number system inputs and outputs. This complicates the design of adaptive controllers and degrades the practical robustness of designed controllers.

Neural network have greatest promise in the realm of non linear control problems. Neural networks potentially provides a general frame work for modelling and control of non linear systems. This project presents a neural network based regulator for non linear, fast acting, multivariable turbo generator control. The hierarchical architecture of the regulator consists of a neural network mapper and a controller.

CHAPTER-II

CONVENTIONAL METHODS

The conventional automatic voltage regulators used in power plants with high fixed gains negative damping in an interconnected power system. To supplant the automatic voltage regulators increasingly attention is being focused on the adaptive controllers.

2.1 ADAPTIVE CONTROLLERS

Adaptive controllers which are in use now a days, involve considerable amount of on line computation and they require a large sampling period. Since the turbo generator is highly fast acting these controllers dose not produced the accurate control due to their large sampling period that is needed.

Turbo generators are highly non linear but while designing the adaptive controllers we neglect the non linearity and consider the turbo generators as linear system. However the industrial systems are non linear and consequently unmodeled dynamics and robustness problem arise in practical applications, and supervisory control is required.

Most turbo generator adaptive controllers are based on the linear model and are subjected to the problem of unequal number of system inputs and outputs. This complicates the design of adaptive controllers and degrades practical operation and robustness of the designed controllers. And hence we go in for Neural Network Regulator which overcomes these difficulties.

CHAPTER-III

TURBOGENERATOR MODEL

Modelling fulfils the vital role in investigations of the performances of electrical power systems and developments of new controllers for turbo generators. Numerous mathematical models has been developed to describe these large and complex systems, and representations have become increasingly detailed, keeping pace with modern technology and new problems in power system operation and control. These serve for validation of theoretical models under all operating conditions, and provide a valuable means of gaining a clear understanding of practical aspects.

The turbo generator consists of a synchronous generator, an exciter with an automatic voltage regulator, and a turbine and governor system. The turbo generator is connected to a large power system through a transformer and two parallel transmission lines. The model is not intended to represent a particular installation but rather to be typical of large modern sets.

3.1 GENERATOR

The generator model considered, has a laminated rotor. Due to greatly increased complexity and consequent cost in computing time the effect of magnetic saturation has been neglected. The synchronous generator is represented by five simultaneous, first order, non linear differential equations. The parameters which where considered to represent the generator are direct and quadrature axes components of stator flux, the rotor angle, slip, field current.

3.2 EXCITER

The excitation for a synchronous generator includes a field winding placed in the slots of rotor in the form of coils. An automatic voltage regulator is used to feed a constant supply to the field winding from thyristor rectifier. The excitation system is described as a first order system, with a small time constant and limits on the positive and negative ceilings of the field voltage.

3.3 TRANSMISSION SYSTEM

The transmission system constitutes a transformer and two parallel transmission lines. The transmission system is represented as a lumped series resistance and reactance with R_T being the combined resistance of the lines and transformer and X_T including the transformer leakage and line reactance.

3.4 BOILER AND TURBINE

A three stage turbine with reheater is employed to drive the synchronous generator. The steam is produced by a conventional coal fired or oil fired boiler which is assumed to be a constant steam source.

The steam flow of the boiler is controlled by both main and interceptor valves. The steam pressure at the turbine inlet valves is assumed to be constant, and the steam mass flow taken to be proportional to the valve position. Each stage is represented by a first order transfer function and the outputs from each were weighed according to their relative contributions to the total shaft torque and summed. The reheater is described by a first order transfer function, as were the valve servomechanism limits were imposed on valve travel and rates of movement.

This model neglects loss and is based on recognition that the main effect of the boiler on the turbine is due to variation of pressure upstream of the main steam valve. The steam mass flow is a non linear function of the valve positions and this is usually corrected under steady state conditions by an electronic function generator. The interceptor valves may be continuously controlled in which case constrains are necessary on valve movement. However it is more usual to keep the interceptor valve fully open, except during emergency conditions. These effect are also included in the model.

This model is taken from the book "MODELLING OF DYNAMIC SYSTEMS" by H.Nicholson. The equations describing the turbo generator and the system parameters are listed in the Appendix-I. This model is simulated using a software called MATLAB, in order to provide the data for training NN Regulator. The simulation is dealt in the next section.

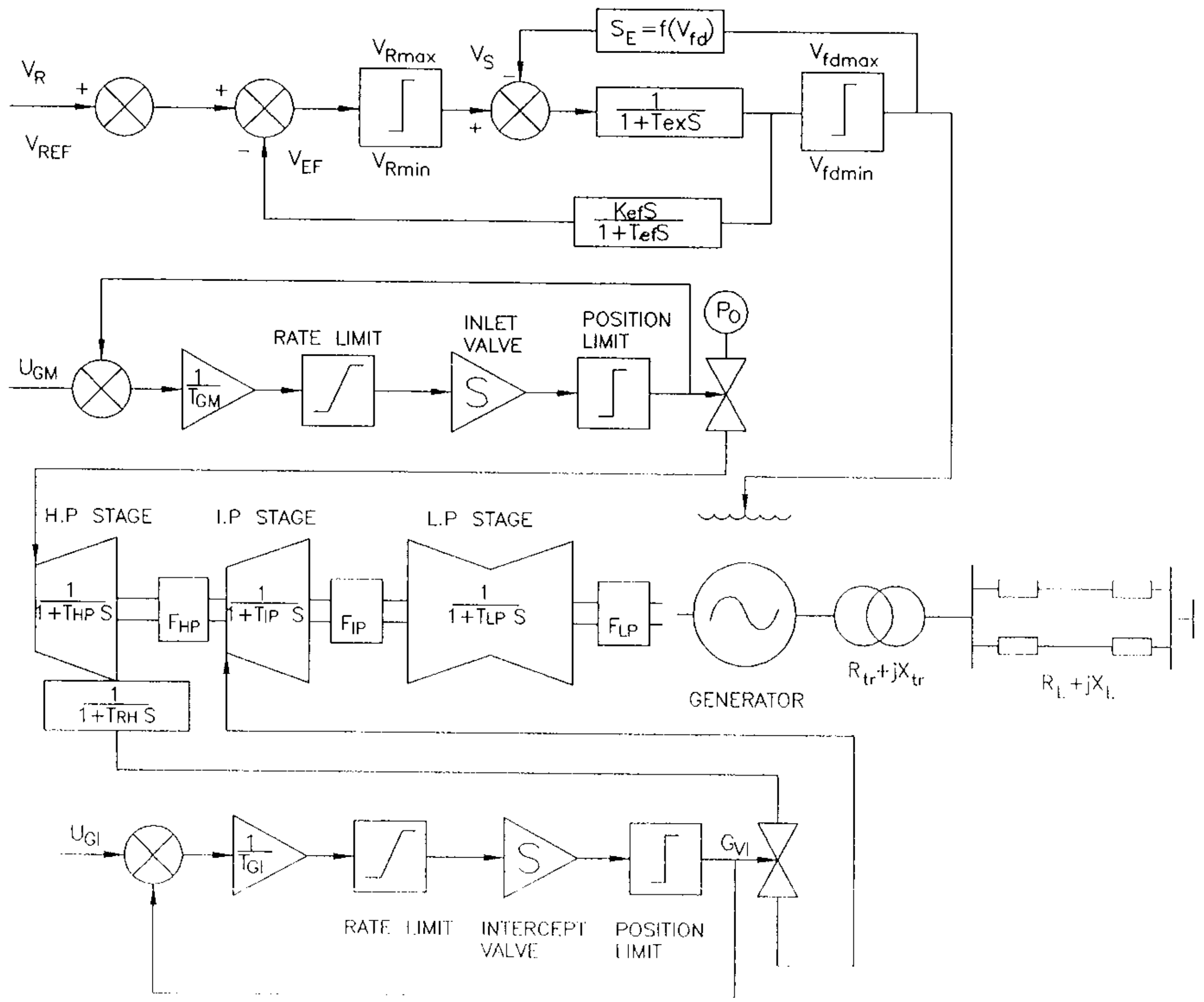

Fig (3.1) A TURBOGENERATOR UNIT CONNECTED TO A POWER SYSTEM

CHAPTER-IV

SIMULATION OF TURBOGENERATOR MODEL

The behaviour of the turbo generator system can be investigated by computer simulation, over a wide range of operating conditions. This includes the assessment of dynamic stability, which is done by solving the twelve non linear equations. These equations are solved using Numerical Integration technique. Few numerical algorithms which can tackle these equations are listed below

- Range kutta third order
- Range kutta fourth order
- Gears method
- Adams and gear method

Numerical Integration routine based on Gear's method which is able to cope with stiff system was selected from numerical algorithm library.

In order to simulate the exact response of the system, the main criteria is to specify the accurate initial conditions. Otherwise, the simulation results will not match with the actual system response. The initial conditions for all the twelve independent variables are found separately using the relations listed in the Appendix-II

The program is written in MATLAB, which is dealt in depth in MATLAB section later. This package provides us a very powerful tool for the simulation of non linear networks, called SIMULINK. The SIMULINK incorporates various network elements under six different categories. These elements are used to represent any non linear system. The main highlight of using SIMULINK is that the responses of any non linear system which has been simulated can easily be visualised as a function of time or any other variable. This visualisation can be made effective by choosing appropriate parameter settings such as horizontal and vertical scales.

In order to reduce the complexity, and to handle the system with ease the generator and boiler turbine are simulated separately and are integrated to obtain the complete response of the turbo generator system.

4.1 SIMULATION OF GENERATOR

The six first order, simultaneous, non linear differential equations representing the turbo generator, exciter and transmission system is formed into a single network with the help of various elements provided in SIMULINK. This network presented at the end of this section comprises of six independent variables in direct and quadrature components of stator flux (Ψ_d, Ψ_q), rotor angle (δ), slip (w), field current (ifd), along with field voltage (efd). These equations are simulated by considering the input from the turbine as a step change. Parameter setting are as important as the initial conditions. Parameters are start time, stop time, maximum step size, minimum step size and tolerance.

The results obtained as a series of data are plotted against time and the plots are analysed.

4.2 SIMULATION OF BOILER AND TURBINE

The boiler and turbine are represented by a set of six first order, non linear, simultaneous differential equations. These equations are represented in SIMULINK with the help of transfer function block. This set of six equations consists of six independent variables in time constants of inlet and intercept valves, output of high pressure stage, reheater, intermediate pressure stage and low pressure stage. The output of this network is the total shaft torque (t_m). The contribution of the high pressure stage to the total shaft torque would be $F_H Y_H$,where F_H is the power fraction from the high pressure stage. By simulating this block with the help of GEAR'S algorithm the values of total shaft can be obtained as a series for different time instants. These values are plotted against time to study the behaviour of t_m as the time elapses. The network used for this simulation is presented at the end of this section.

4.3 SIMULATION OF TURBOGENERATOR

The above two simulation are integrated to get the complete response of the system. To achieve this the step input given for shaft torque is replaced by the output t_m of the boiler turbine model. This interconnected complex model would represent the complete turbo generator system comprising a turbo alternator, exciter, turbine and boiler system connected to a large power system through a transformer and two parallel transmission lines. The output values of all the twelve independent variables of the system are obtained as a series of data on simulating this network. The behaviour of various system variables are studied by plotting them against time. The values of excitation input (v_r), governor input (u_g), excitation voltage(efd),and slip are stored in a file. The data obtained from simulation is used to train the neural network regulator. Which is dealt separately in detail in the sections to come.

CHAPTER-V

NEURAL NETWORK

5.1 BIOLOGICAL NEURONS

The human brain is one of the most complicated things in the body. It contains approximately ten thousand million basic units, called neurons and they are connected to each other. Soma is the body of the neuron. Attached to the soma are long, irregularly shaped filaments, called dendrites.

These nerve processes are often less than a micron in diameter and has complex branching shapes. The dendrites act as the connections through which all the input to the neuron arrive. Another type of nerve process attached to the soma is called axon. This is electrically active , unlike the dendrite, and serves as a the output channel for the neuron. Axons always appear on output cells, but are often absent from inter neurons, which have both input and output on dendrites. The axon is a non linear threshold device, producing a voltage pulse called an action potential, that lasts about one millisecond when the resting potential within the soma rises above a certain critical threshold.

The axon terminates in a specialised contact called synapse that couples the axon with the dendrite of the another cell. There is no direct linkage across the junction, rather it is temporary chemical one. The synapse releases chemicals called neuro transmitters when its potential is raised sufficiently by the action potential. At the synaptic junction, the number of gates opened on the dendrite depends on the neuro transmitters released. It also appears that some synapse excite the dendrite they affect, whilst others serve to inhibit it. This corresponds to altering the local potential of the dendrite in a positive or negative direction. A single neuron will have many synaptic inputs on its dendrites, and may have many synaptic outputs connecting it to other cells.

5.2 ARTIFICIAL NEURONS

An Artificial Neural system that can do the function of actual biological neural system is called Neural Networks. In Artificial neural network each neuron is connected to each other or to itself through interconnections and these interconnections has their own synaptic strength or weights. Like a computer it does not have any memory location to store the information, but the knowledge is stored in the processing element.

There are many neural network models like Hopfield Net, Hamming Net, Carpenter Net, Guassian Classifier, Multi layer perceptron etc. Neural net models are specified by the Net Topology, Node characteristics, and training or learning rules. For the regulator application the sub class of multi layer network, namely, fully connected, feed forward layered network is selected.

The typical structure of this neural network is shown in fig (). The network is formed with three layers - input layer, hidden layer and output layer. The connections are given as a group. All neurons are connected to all other neurons thus forming a network.

The computation of neural network has two phases

- Learning phase
- Testing phase

5.2.1 LEARNING PHASE

Starting with one set of weight in the network and then modify some or all the weights, and then run the network with new set of weights. Repeat this process until some predetermined output is met. The process of changing the weight, or rather, updating the weight is called training or learning.

During learning phase, the network is separately presented with a set of input output patterns. Learning is accomplished by a general rule which dynamically modifies the weights of all the interconnections in an attempt to generate the output desired pattern for each input pattern.

The network can be subjected to supervised or unsupervised learning. The learning should be supervised if external criteria are used and matched by the network output and if not, the training is unsupervised. For the given input along with an expected response, the supervised neural network attempts to predict the future once the training is completed. The unsupervised network would act like a look up memory that is indexed by its contents. After the learning is completed, the network generates responses to the test patterns. Neural network computations are collectively performed by the entire network with the knowledge represented in the interconnection weights between neurons.

During learning the weights of the interconnections are appended, and the process of learning is known as Training the Network. For training, different algorithms exists. Among the different algorithms the algorithm that best suits this regulation problem is **BACK PROPAGATION ALGORITHM.**

5.3 TAXONOMY OF NEURAL NETWORKS

There are many neural network models like Hopfield net, Hamming net, Carpenter net, Gaussian classifier , single layer perceptron, and multi layer perceptron. The neural net models are specified by net topology, node characteristic and training or learning rules. the network is divided between nets with binary and continuous valued inputs. Below this, nets are divided between those trained with and without supervision. Nets are trained with or without supervision.

Nets trained with supervision such as Hopfield net and perceptron are used as associative memories or as classifiers. These nets are provided with side information that specify the correct class of new input patterns during training. Nets trained without supervision, such Kohonen's feature-map forming nets are used as vector quantizers or to form clusters. A further difference between nets, is whether adoptive training is supported. Although all nets shown can be trained adaptively, the Hopfield net and hamming net are generally used with fixed weights.

5.3.1 HOPFIELD NET

These nets are most appropriate when exact binary representation are possible as with black and white images where input elements are pixels values or with ASCII text where input elements could present bits in the eight ASCII representation of each character. These nets are less appropriate when input values are continuos, because a fundamental presentation problem must be addressed to analog quantities to binary values. This net can be used as an associative memory or to solve optimisation problems.

5.3.2 THE HAMMING NET

The Hopfield net is often tested on problems where inputs are selected an exemplar and reversing bit values randomly and independently with a given probability. This is a classic problem in communication theory that occurs when binary fixed length signals are sent through a memory less binary symmetric channel. The optimum minimum error classifier in this case calculates the hamming distance to the exemplar for each class and selects that class with the minimum hamming distance. The Hamming distance is the number of bits in the input which do not match a corresponding exemplar bits.

5.3.3 THE CARPENTER / GROSSBERG CLASSIFIER

Carpenter and Grossberg in the development of their adaptive resonance theory have designed a net, which forms the clusters and is trained without supervision. This net implements a clustering algorithm that is very similar to the simple sequential leader clustering algorithm. The leader algorithm selects the first input as the exemplar for the first cluster. The next input is compared to the first cluster exemplar. It follows the leader and is clustered with the first if the distance to the first is less than a threshold, otherwise it is the exemplar for the new cluster.

5.3.4 SINGLE LAYER PERCEPTRON

This simple net can be used with both continuous valued and binary inputs. It can learn to recognise simple patterns. A perceptron can decide whether an input belong to one of two classes. The single node computes a weighted some of the input elements, subtracts a threshold and passes the result through a hard limiting non linearity such that the output is either +1 or -1.The decision rule is to respond class A if the output is +1 and class B if the output is -1.A useful technique for analysing the behaviour of the nets such as perceptron is to plot a map of the decision regions created in multidimensional space spanned by the input variables. This decision regions specify which input values result in a class A and which result in class B response. The perceptron forms two decision regions separated by a hyperplane. The equation of the boundary line depends on the connection weights and threshold. Connection weights and the threshold in a perceptron can be fixed or adapted using a number of different algorithms. The original perceptron convergence procedure for adjusting weights was developed by Rosenblatt.

5.3.5 MULTILAYER PERCEPTRON

Multilayer perceptrons are feed-forward nets with one or more layers of nodes between the input and output nodes. These additional layers contains hidden nodes that are directly connected to both input and output nodes. Multilayer perceptrons overcome many of the limitations of single layer perceptrons, but were generally not used in the past because of lack of effective training algorithms. Although it can be proven that these algorithms converge as with single layer perceptrons, they have been shown to be successful for many problems. The capabilities of multilayer perceptrons stem from the non linearities used with in nodes. The back propagation algorithm is used to train the network.

5.3.6 KOHENON'S SELF ORGANISING FEATURE MAPS

The important organising principle of sensory pathways in the train is that the placement of neurons is orderly and often reflects some physical characteristic of the external stimulus being sensed .For example, at each level of the auditory pathway, nerve cells and fibers are arranged automatically in relation to the frequency which elicits the greatest response in each neuron. This tonotopic organisation in the auditory pathway extends up to the auditory cortex. Although much of the low level organisation is genetically predetermined, it is likely that some of the organisation is at higher levels is created during learning by algorithms which promote self organisation. Kohonen's presents one of such algorithm which produces what he calls self organizing feature maps similar to those that occur in the brain.

Kohonen's algorithm creates a vector quantizer by adjusting weighs from common input nodes to output nodes arranged in a two dimensional grid. Output nodes are extensively inter connected with many local connections. Continuous valued inputs are presented sequentially in time without specifying the desired output. After enough input vector have been presented, weights will specify cluster or vector centres that sample the input space such that point density of function of the vector centres tends to approximate the probability density function of the input vector. In addition , the weights will be organised such that topologically closed nodes are sensitive to inputs that are physically similar. Output nodes will thus be ordered in natural manner. This may be important in complex systems with many layers of processing because it can reduce lengths of interlayer connections.

5.4 APPLICATIONS

- Hyphenation

  A network can be trained to predict the Hyphenation of randomly chosen long words.

- Nuclear binding energies

  The binding energy B of atomic of nuclei is a function of their proton and neutron numbers(Z & N).Since this function is almost linear over a wide range, network can able to predict the separation of energy of last neutron.

- Font Orientation

  The assembly of electronic circuits requires the determination of the correct orientation of its components, which may be achieved by detecting the orientation of the prints on them. A four-layer neural network was constructed to detect whether the symbols in the text were originated correctly or upside down.

- Robotics

  The control of trajectories of robotic manipulators operating in production plants or under hostile environmental conditions is a task for which neural networks appear to be well suited.

- Medicine

  Neural networks can classify patterns even in cases where it is difficult to formulate simple rules to distinguish the desired categories. This ability has led to a potential rewarding application in medical diagnosis. The system PAPNET introduced by Neuro medical systems Inc., employs a neural network to search for abnormal cells for the detection of certain types of cancer at an early stage.

- Neural networks finds in applications in the area of optimisation, image recognition, fault diagnosis, control system etc.,
- The commercial applications include prediction of stock commodity prices, signature verification on bank cheques and risk estimation in loan under writing.

CHAPTER-VI

BACK PROPAGATION ALGORITHM

6.1 BUILDING BLOCK

The figure (6.1) shows the neuron used as the fundamental building block for back propagation networks. A set of input is applied, either from outside or from a previous layer. Each of these is multiplied by a weight and products are summed. This summation of products is termed as net and must be calculated for each neuron in the network. After net is calculated, an activation function is applied to modify it, there by producing the signal out, and this function is called sigmoid or simply a squashing function.

6.2 TRAINING THE NETWORK

The objective of training the network is to adjust the weights so that application of a set inputs produces the desired set of outputs. Before the training process all the weights must be initialised to small random numbers.

Training the back propagation network requires the following steps:

I Apply the input to the network

II Calculate the output of the network

III Calculate the error between the network output and the desired output

IV Adjust the weights of the network in such a way that it minimises the error.

V Repeat the steps from I to IV until the error for the entire network is acceptably low.

To be brief, a back propagation network typically starts out with a random set of weights. The network adjusts its weighs each time it sees an input-output pair. Each pair requires three stages.

(I) Forward Pass

(II) Error Calculation

(III) Reverse Pass

The forward pass involves presenting a sample input to the network and allowing activations to flow until they reach the output layer. During the backward pass, the network's actual output(from the forward pass)is compared with the target output and the error estimates are computed for the output units.

The weights connected to the output units can be adjusted in order to reduce those errors. It can be used to estimate the error of the output units to derive error estimation for the units in the hidden layers. Finally, errors are propagated back to the connections stemming from the input units .

CHAPTER-VII

NEURAL NETWORK REGULATOR

Neural networks has greatest promise in the realm of non linear control problems and potentially provides a general frame work for modelling and control of non linear systems. Neural Network have a highly parallel structure which lends itself immediately to parallel implementation. such an implementation can be expected to achieve a higher degree of fault tolerance than other conventional schemes.

The proposed NEURAL NETWORK REGULATOR consist of , NEURAL NETWORK MAPPER and NEURAL NETWORK CONTROLLER. Neural Network Mapper is an optimal predictor which predicts the output of the plant while the controller produces control signals in order to keep the plant's output to the requirements. This hierarchical architecture presented shows good Performance and illustrates the potential of NEURAL NETWORK REGULATOR.

7.1 NEURAL NETWORK MAPPER

Neural Network Mapper is principally employed for mapping input output of the plant i.e. to predict the output response of the plant. Instead of using the plants output directly for the control purpose, a mapper is employed to foresee the plant's deviation from normal operating condition. The output is then fed to the controller for the purpose of regulation. This increases the speed of response of the NEURAL NETWORK REGULATOR.

The Mapper consist of twelve inputs and two outputs and the number of hidden layers would be chosen according to the ease of training. Neural Network Mapper is employed during the first time instant of a sample interval. Actually, four inputs are given to the mapper each at the three successive time instants. For example to predict the output of the plant at a time instant 't' the mapper is fed with the inputs at instants (t-1),(t-2),(t-3).

With the help of the data furnished from the simulation of the plant, in the previous section the Neural Network Mapper is trained, using backpropagation algorithm. The four inputs fed to network for training are deviations in Slip, Terminal voltage, Governor input and Excitation input.

The Neural Network Mapper is employed using error between the plant and the networks output, denoted by e_s the performance index,

$$J_m=1/2\ \Sigma_j\ r_j\,[y_j(t)-y_j(t)']^2$$

is used in the training, where $y_j(t)'$ is the j^{th} output of the sub network. The weight s in the Neural Network mapper are updated as follows,

$$w_m(t)= w_m(t-1)-\lambda\nabla E_m(w_m)$$

where the subscript m refers to the Neural Network Mapper and $w_m(t)$ is the weight vector consisting of all weights in the sub network at time t. The quantity $\nabla E_m(w_m)$ is the gradient vector, consisting of the derivatives of J_m with respect to each weight in the Neural Network Mapper and λ is the step length for weight updating. After proper training of the Neural Network Mapper, the error between y_j' and y_j tends to zero.

The training is accomplished by assigning the non linear transfer function in the hidden layer

$$y=(1-e^{-x})/ (1+e^{-x})$$

and the time delay k is chosen to be one.

Samples of the system outputs taken every 20 ms where used to train the Neural Network Mapper using backpropagation errors for weight adjustment with $\lambda=0.2$. The were initialised as random values in the range of + or - 0.1 and penalty coefficients applied on the voltage and speed were 20 and 1 respectively.

To start with the parameter values which has to be fixed in order to train the mapper are learning rate, momentum, increments and decrements in learning rate and error ratio.

With the learning rate set at 0.1 and the momentum at 0.95, the sum squared value of error starts at range of 1000 and finally converges at 8.6. At this trial, the learning rate is incremented by 1.05 and decremented by 0.85.

At the second trial, the learning rate is fixed at 0.15 and with the same momentum, the training is done. Hence the sum squared value of error starts at 1000 and finally converges at 7.3.

With various possible values for the learning rate, the training is done and the sum squared error value is reduced at each trail. Finally at a learning rate value of 0.2 and its increments and decrements set at 1.05 and 0.85 respectively, the sum squared error converges at 5.8.

The programs of the Neural Network Mapper are dealt in the programming section later.

7.2 NEURAL NETWORK CONTROLLER

The purpose of the Neural Network Controller is to compare the outputs of Mapper with the desired outputs, so as to send the control signals to the plant to achieve regulation.

The controller consist of six input layers and two output layers and the size of hidden layer would be chosen according to ease of training. The Neural Network Controller is employed during the second time instant in a sample interval. To control the major outputs of the turbo generator the excitation voltage and the slip are measured online and used as feed back variables to yield the excitation control signal and the governor input.

At the second time instant the weights in the Neural Network Controller are modified as adaptive parameters, according to errors e_d between the desired system outputs of the Neural Network Mapper. The performance index would be given as

$$J_c=1/2\ \Sigma_j\ r_j\,[y_j(t)-y_j(t)']^2$$

where $y_j(t)'$ is the jth output of the subnetwork. The weights of the Neural Network Controller are updated as follows

$$w_c(t) = w_c(t-1)-\lambda\nabla E_c(w_c)$$

where the subscript c refers to the Neural Network Controller and $w_c(t)$ is the weight vector consisting of all weights in the controller at the time t.

At this stage the Neural Network Mapper function as a channel for error backpropagation and its weights are unchanged. The output, excitation input and governor input from the controller sub network are sent to control the plant and are also taken as inputs to the Neural Network Mapper for the next mapping stage. The weights in the Neural Network Controller are updated to minimise the cost function y' used instead of y for two reasons. Future information is involved in the cost function and also the updating of the controller parameters with the gradient vector is directly related to the outputs of the Neural Network Mapper.

To achieve the control operation at a time t, the controller is fed with inputs at three successive time instants as (t-1),(t-2) and (t-3) instants. The step length λ used in training the Neural Network Controller was set at 0.1, half that used in Neural Network Mapper. This use of different step sizes in the two sub networks yielded larger backpropagated signals for weight updating in the Neural Network Controller, which speeds convergence and makes the regulator more robust.

The programs of the Neural Network Controller are dealt in the programming section later.

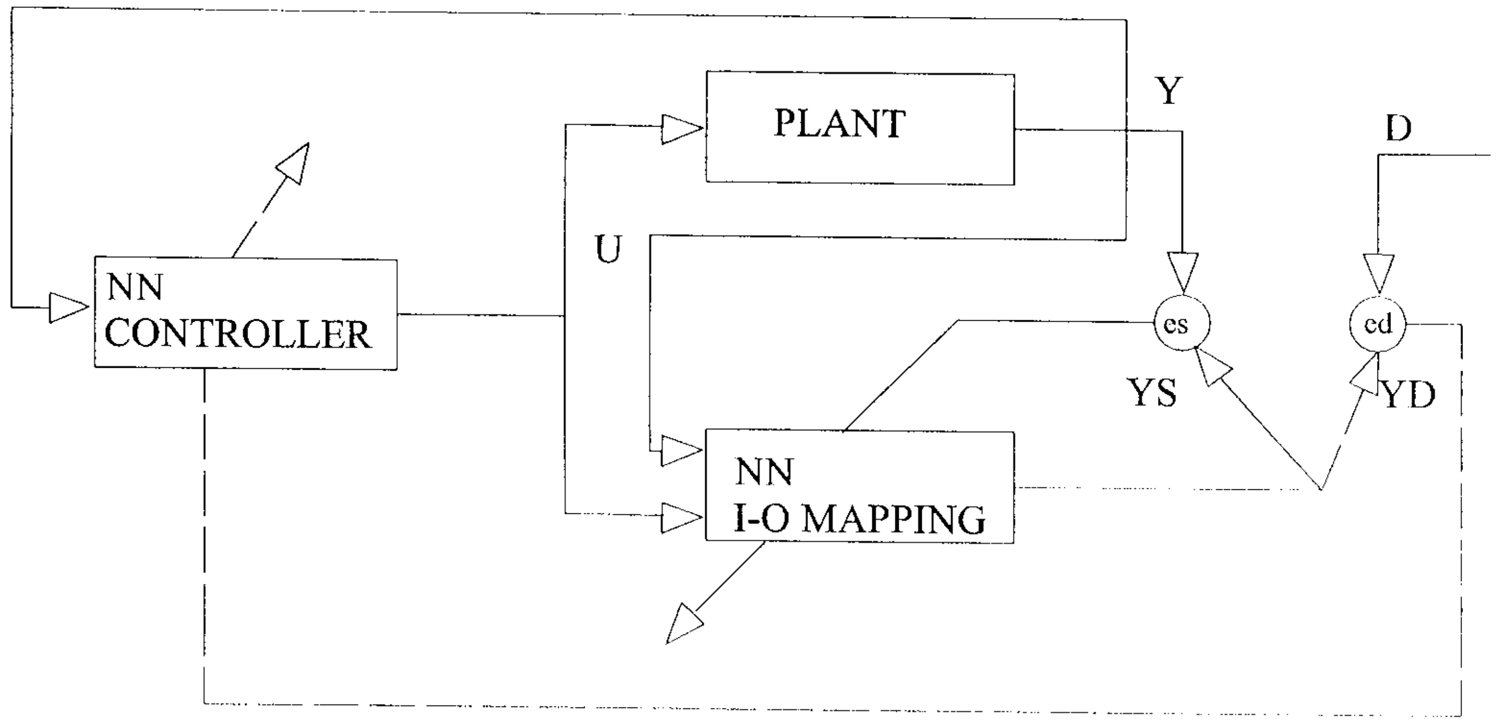

fig (7.1) A NN regulator for adaptive control

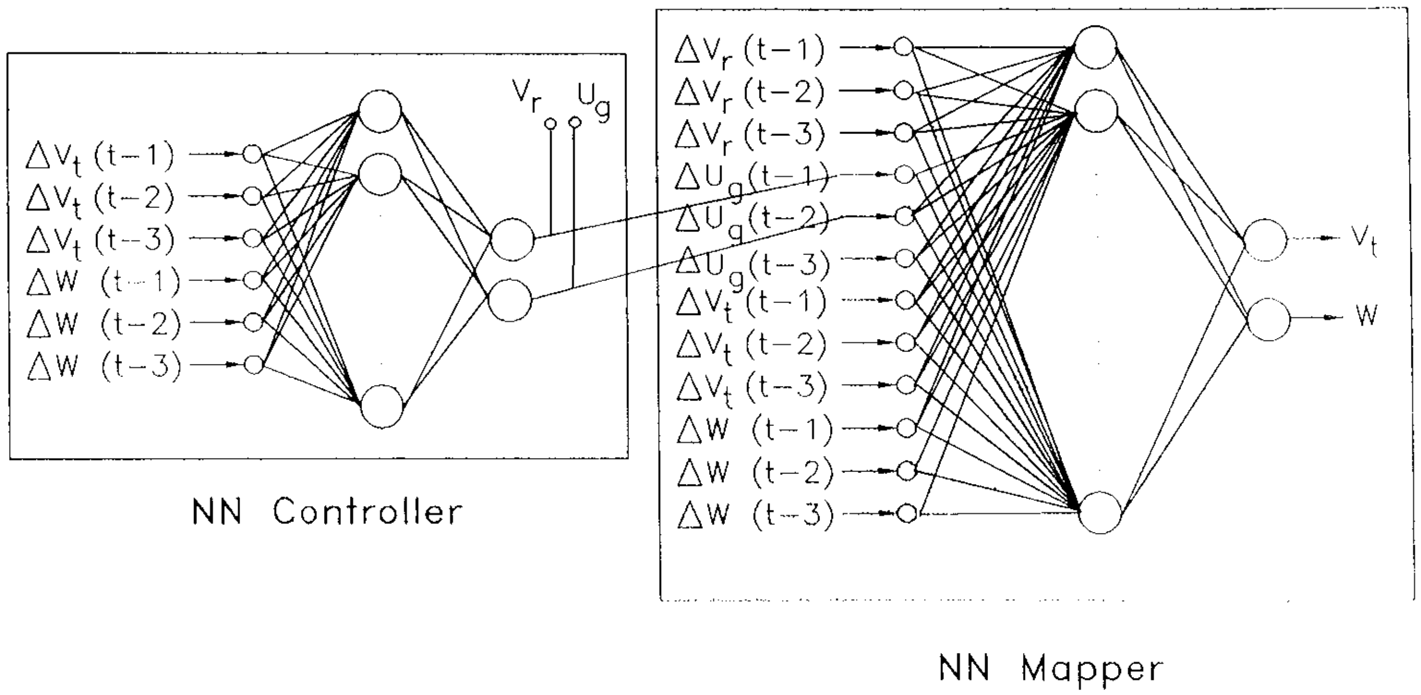

Fig (7.2) NEURAL NETWORK REGULATOR

CHAPTER-VIII

INTRODUCTION TO MATLAB

MATLAB is a an interactive program which help in numerical computation and data visualisation. MATLAB is built upon a foundation of sophisticated matrix software for analysing linear systems of equations. The springing from these numerical foundations have proven to be extraordinarily versatile and capable in their ability to solve problems in Physics, Chemistry, Engineering, Finance - almost any application area that deals with complex numerical calculations.

SIMULINK is a tool for modelling, analysing and simulating an extraordinary wide variety of physical and mathematical systems, including those with non-linear elements and those which make use of continuous and discrete time. As an extension of MATLAB ,SIMULINK adds many features specific to dynamic systems while retaining all of MATLAB's general purpose functionality. Using SIMULINK ,a system can be modelled graphically, side stepping much of the nuisance associated with conventional programming.

Toolboxes are specified collection of M-files (MATLAB language programs) built specifically for solving particular classes of problems. Toolboxes are more than just collections of useful functions, though. They represent the efforts of some of the world's top researchers in field such as controls, signal processing, neural network ,system identification, and others.

Neural Network Toolbox is a collection of MATLAB functions for designing and simulating Neural Networks. Here algorithms are already available which are to moulded to our requirements. Few algorithms that are available are Backpropagation, ADLINE, Kohenon's self learning algorithm and etc.,

CHAPTER-IX

DESCRIPTION OF THE PROGRAM

The programs are written in a software called MATLAB. The programs are divided into modules in order to provide better understanding, also to provide flexibility of modifying the program to the requirements and split the work. Four program are written in order to implement the Neural Network Regulator. Four separate M-file has been created for this purpose.

The data obtained from the simulation are to be transformed into appropriate form in order to feed this as the input to the Neural Network Mapper. The Program I stated in the Appendix (III) converts the result of the simulation into a required form. As stated in the section(***) the input matrix must contain the change in the values of slip, terminal voltage, regulator input, governor input. The above values at the instances (t-1),(t-2),(t-3) are to considered to formulate the input matrix and the desired matrix must contain the values of terminal voltage, slip at the instant (t). Two matrixes p, t are the input and desired matrixes are formulated in this program. Algorithm is stated below

Step 1 Find the change in the terminal voltage at instants .

Step 2 Find the change in the slip at instants .

Step 3 Find the change in the regulator input at instants .

Step 4 Find the change in the governor input at instants .

Step 5 First to third row of the input matrix is assigned with the matrixes obtained from the step 1 .

Step 6 Fourth to sixth row of the input matrix is assigned with the matrixes obtained from the step 2 .

Step 7 Seventh to ninth row of the input matrix is assigned with the matrixes obtained from the step 3.

Step 8 Ninth to twelth row of the input matrix is assigned with the matrixes obtained from the step 4.

Step 9 Form the desired matrix with the help of terminal voltage, slip.

Once the program is run the matrixes is available in the 'Command Window' of MATLAB . These matrixes are used in implementing the Neural Network Mapper.

The program-II trains and implements the Neural Network Mapper. This program implements the feed forward network and trains the network. Neural Network Mapper stated in the chapter (***) is implemented in this program. This program has flexibility of changing the parameters like hidden neurons, learning rate and etc.,. Algorithm for implementing the Neural Network Mapper is listed below

Step 1. State the number of hidden units.

Step 2. Initialise the feed forward network.

Step 3. Set the display frequency, error goal, learning rate, increase in learning rate, decrease in learning rate, momentum and error ratio.

Step 4. Train the network using the backpropagation algorithm with momentum and adaptive learning rate.

Step 5. Once the network is trained the weights and biases are stored in a mat file.

Once the Mapper is trained the input matrix for the controller is formed. This is achieved with the help of the program-III. The input to the controller as stated in the chapter(***) are the change in the terminal voltage, slip. The desired vector consist of the governor input and the regulator input. The desired matrix and the input matrix is formed using the program-III shown in the Appendix-III. The algorithm for the program-III is stated below

Step 1. With the help of the simulation results the difference in the terminal voltage for three sampling instants are obtained.

Step 2. With the help of the simulation results the difference in the Slip for three sampling instants are obtained.

Step 3. The first three rows of the input matrix is substituted by the matrix obtained from step 1.

Step 4. The fourth to sixth rows of the input matrix is substituted by the matrix obtained from step 2.

Step 5. The desired matrix the first row is formed using the values of the regulator input obtained from the simulation.

Step 6. The desired matrix the second row is formed using the values of the governor input obtained from the simulation.

Program-IV deal with the training of Neural Network Controller. This program is similar to that of the Program-II. Algorithm fcr Program-IV is stated below

Step 1. State the number of hidden units.

Step 2. Initialise the feed forward network.

Step 3. Set the display frequency, error goal, learning rate, increase in learning rate, decrease in learning rate, momentum and error ratio.

Step 4. Train the network using the backpropagation algorithm with momentum and adaptive learning rate.

Step 5. Once the network is trained the weights and biases are stored in a mat file.

Program-V deal with the Neural Network Regulator simulation. The details of the NN Regulator is described in the Chapter (***). This program uses the data from the Program-II and Program-IV that is the weights & biases of the Neural Network Mapper (NNM) and Neural Network Controller (NNC). The algorithm is explained below

Step 1. Weights of the NNM which is stored in the file is read.

Step 2. Input matrix is formed using the simulation results.

Step 3. This is given as the input to the NNM.

Step 4. The weights of the NNC is read from the file

Step 5. The input matrix for the NNC is formed from this results

Step 6. The controller output is graphically represented

CHAPTER-X

CONCLUSION

The project presents an NN Regulator for adaptive multivariable control of a turbo generator. This consist of two sub networks for modelling and adaptive control respectively. The controller avoids use of sign of plant errors during backpropagation, dose not require reference model or inverse system model and avoids the use of probing signal.

The NN regulator has a compact structure, which can easily be extended to cater for more complex dynamic systems or additional control loops. The modelling adaptive control performance have been evaluated by simulation studies on detailed non linear model of a turbo generator system. With various possible values for the learning rate, the mapper is trained and the sum squared error value is reduced at each trail. Finally at a learning rate value of 0.2 and its increments and decrements set at 1.05 and 0.85 respectively, the sum squared error converges at 5.8.

SCOPE FOR FURTHER WORK

This project lays the base for the usage of Neural Networks for complex control operation. Even though, only the software part of the controller is realised in this project the better performance of the simulation results shows that this controller can be used meritoriously for real time applications. This can be made possible by the hardware implementation which can be done with the programmable chips. Once this has been achieved the neural network controllers could outdate the conventional controllers implied in various control operations with their enormously increased accuracy and speed of control operation.

The other main areas related to this project is that this project ensures that the neural networks can be successfully implemented for the vector control method of speed control of induction motor ,which could be a real boon for industrial applications

REFERENCES

1. B.W. Hogg , "Representation and control of Turbo generators in electrical power system," in Modelling of Dynamic Systems, vol.2, H.Nicholoson, Ed. London : Peter Peregrinus , 1981 , ch.5.
2. Q.H. Wu and B.W. Hogg, "Adaptive controller for a Turbo Generator system," Proc.Inst.Elec.Eng., vol. 135,pt. D, no. 1, pp35-42,1998.
3. Derrick H. Nguyen and Bernard Widrow, "Neural Network for Self-Learning Control System", IEEE Control System Mag., pp. 18-23, April 1990.
4. K. J. Hunt, D. Sbarbaro, R. Zbikowski and P. J. Gawthrop, "Neural Network For Control Systems-A survey," Automatica, vol. 28, no.6,pp. 1083-1112, 1992, Pergamon Press Ltd., Oxford, England.
5. Howard Demuth, Mark Beale, "Neural Network Toolbox" Release Notes Version 2.0a , The Math Works Inc,.
6. K. R. Padiyar, " Power System Dynamics,"1992, Interline Publications Ltd., Bangalore.

APPENDIX-I

Equation representing the Turbogenerator

Equations representing the Generator

$P\delta = \omega_b\, (S_m - S_{m0})$

$p\, S_m = [-d\, (S_m - S_{m0}) + T_m - T_e]/2H$

$p\, E_q' = [-E_q' + (X_d - X_d')i_d + E_{fd}]/ T_{d0}'$

$P\, E_d' = [-E_d' + (X_q - X_q')i_q]$

$T_e = E_d'\, i_d + E_q'\, i_q + (X_d - X_q')i_q\, i_d$

$(X_d' + Z_l)\, i_d - (R_a + Z_r)\, i_q \quad = f_1(\delta) - E_q'$

$-(R_a + Z_r)\, i_d - (X_q' + Z_l)i_q \quad = f_2(\delta) - E_d'$

where

$f_1(\delta) = h_1 E_b \cos(\delta) + h_2 E_b \sin(\delta)$

$f_2(\delta) = h_2 E_b \cos(\delta) - h_1 E_b \sin(\delta)$

Equations representing the Boiler and Turbine

$pY_h = (G_m P_o - Y_h)/\zeta_h$

$pY_r = (Y_h - Y_r)/\zeta_r$

$pY_i = (G_i Y_r - Y_i)/\zeta_i$

$pY_l = (Y_i - Y_l)/\zeta_l$

$pG_m = (U_m - G_m)/\zeta_{mg}$

$pG_i = (U_i - G_i)/\zeta_{ig}$

$T_m = Y_h F_h + Y_i F_i + Y_l F_l$

SYSTEM PARAMETERS

Values are in per unit

GENERATOR :

R_a =0.00327,

X_q=1.5845,

X_q'=1.04,

X_d=1.7572,

X_d'=0.4245,

T_{d0}'=6.66,

T_{q0}'=0.44,

f=50Hz,

H=3.542.

TRANSFORMER :

R_L=0.0,

X_L=0.1364.

TRANSMISSION LINE :

R_l=0.08593,

X_l=0.8125,

B_c=0.1184.

EXCITATION SYSTEM :

K_A=400,

T_A=0.025,

E_{fdmin}= -6.0,

E_{fdmax}= 6.0.

TURBINE AND GOVERNOR :

P_o=1.2p.u

ζ_h=0.3s

ζ_r=10.0s

ζ_i=0.3s

ζ_l=0.72s

ζ_{mg}=0.1s

ζ_{ig}=0.01s

F_h=0.24

F_i=0.34

F_l=0.42

where

δ	- rotor angle
ω_b	- angular frequency of infinite bus bar
S_m	- slip
S_{m0}	- initial slip
i_q	- q-axis component of armature current
E_q	- transformed armature voltage in q axis
E_q'	- transient transformed armature voltage in q axis
X_q	- synchronous reactance in q-axis
X_q'	- transient reactance in q-axis
X_d	- synchronous reactance in d-axis
X_d'	- transient reactance in d-axis
i_d	- d-axis component of armature current
E_d	- transformed armature voltage in d axis
E_d'	- transient transformed armature voltage in d axis
E_{fd}	- field voltage
T_m	- generator shaft toque
T_e	- air gap torque
$h_1+j\,h_2$	- the voltage gain at the generator terminal with armature open
E_b	- Thevenin's voltage source viewed from generator terminals
R_a	- armature resistance
$Z_i + jZ_r$	- the input impedance of the external network viewed from the generator

H	- inertia constant
Y_h	- output of the high pressure stage per unit
Y_r	- output of the reheater stage per unit
Y_i	- output of the intermediate pressure stage per unit
Y_l	- output of the low pressure stage per unit
P_o	- internal boiler steam pressure
G_m	- position of governor on inlet valve
G_i	- position of governor on intercept valve
U_m	- actuating signal to governor on inlet valve
U_i	- actuating signal to governor on intercept valve
ζ_h	- time constant associated with high pressure stage
ζ_r	- turbine reheat time constant
ζ_i	- time associated with intermediate pressure stage
ζ_l	- time associated with low pressure stage
ζ_{mg}	- time associated with inlet valve
ζ_{ig}	- time associated with interrcept valve
F_h	- power fraction from high pressure stage
F_i	- power fraction from intermediate pressure stage
F_l	- power fraction from low pressure stage

APPENDIX-II

EQUATION FOR THE CALCULATION OF INITIAL CONDITIONS

1. Compute I_{ao} from

   $I_{ao}=(P_{to}- Q_{to})/ V_{to}\angle\theta_{to}$

2. Compute E_{qo} and δ_o from

   $E_{qo}\angle\delta = V_{to}\angle\theta_o+(R_a+j X_q) I_{ao}\angle\phi_o$

3. Compute

   $I_{do}= -I_{ao}\sin(\delta_o-\phi_o)$

   $I_{qo}= I_{ao}\cos(\delta_o-\phi_o)$

   $V_{do}= -V_{to}\sin(\delta_o-\theta_o)$

   $V_{qo}=V_{to}\cos(\delta_o-\theta_o)$

4. Compute

   $E_{fdo}=E_{qo}-(X_d-X_q) I_{do}$

   $E_{qo}'= E_{fdo}+(X_d-X_d') I_{do}$

   $E_{do}'= -(X_q-X_q') I_{qo}$

   $T_{eo}= E_{qo}' I_{qo} + E_{do}' I_{do}+(X_d-X_q') I_{do} I_{qo}$

APPENDIX-III

PROGRAM-I

```
% This is program to form the input matrix for NNM
% Store the TERMINAL VOLTAGE in the command window as vt
        p(:,1)=vt(5:999,1)-vt(4:998,1);
        p(:,2)=vt(4:998,1)-vt(3:997,1);
        p(:,3)=vt(3:997,1)-vt(2:996,1);
% store the change in SPEED  in the command window as slip
        p(:,4)=slip(5:999,1)-slip(4:998,1);
        p(:,5)=slip(4:998,1)-slip(3:997,1);
        p(:,6)=slip(3:997,1)-slip(2:996,1);
% store the FIELD VOLTAGE in the command window as efd
        p(:,7)=efd(5:999,1)-efd(4:998,1);
        p(:,8)=efd(4:998,1)-efd(3:997,1);
        p(:,9)=efd(3:997,1)-efd(2:996,1);
% store the GOVERNOR INPUT in the command window as ug
        p(:,10)=ug(5:999,1)-ug(4:998,1);
        p(:,11)=ug(4:998,1)-ug(3:997,1);
        p(:,12)=ug(3:997,1)-ug(2:996,1);
% input matrix is complete
        p=p';
% This is to form the desired matrix
        t(:,1)=vt(6:1000,1);
        t(:,2)=slip(6:1000,1);
% desired matrix is complete
        t=t';
```

Program-II

```
% this is to train the NNM

p;      % input vector

t;      % desired vector

s1=14;        % hidden neurons

[wm1,bm1,wm2,bm2]=initff(p,s1,'tranfun',t,'tranfun');
                        % Initialisation of weights
tp(1)=10;               % display frequency
tp(2)=100000;           % maximum no. of epochs
tp(3)=0.1;              % error goal
tp(4)=0.2;              % learning rate
tp(5)=1.05;             % increase in learning rate
tp(6)=0,75;             % decrease in learning rate
tp(7)=0.95;             % momentum constant
tp(8)=1.01;             % error ratio
[wm1,bm1,wm2,bm2,te,tr]=trainbpx(wm1,bm1,'tranfun',wm2,bm2,'
                                tranfun',p,t,tp); % train the network

save wm1,bm1,wm2,bm2;       % save weights and biases
```

PROGRAM-III

```
% This is program to form the input matrix for NNC
% Store the TERMINAL VOLTAGE in the command window as vt
        p(:,1)=vt(5:999,1)-vt(4:998,1);
        p(:,2)=vt(4:998,1)-vt(3:997,1);
        p(:,3)=vt(3:997,1)-vt(2:996,1);
% store the change in SPEED  in the command window as slip
        p(:,4) = slip(5:999,1)-slip(4:998,1);
        p(:,5) = slip(4:998,1)-slip(3:997,1);
        p(:,6) = slip(3:997,1)-slip(2:996,1);
% input matrix is complete
        p = p';
% This is to form the desired matrix
% store the FIELD VOLTAGE in the command window as efc
        t(:,1) = efd(5:999,1);
% store the GOVERNOR INPUT in the command window as ug
        t(:,2) = ug(5:999,1);
% desired matrix is complete
        t = t';
% You can now proceed for the training
```

Program-IV

```
% this is to train the NNC
        p;      % input vector
        t;      % desired vector
        s1=8; % hidden neurons
        [wc1,bc1,wc2,bc2]=initff(p,s1,'tranfun',t,'tranfun');
                                        %initialisation of weights
        tp(1)=10;           % display frequency
        tp(2)=100000;       % maximum no. of epochs
        tp(3)=0.1;          % error goal
        tp(4)=0.2;          % learning rate
        tp(5)=1.05;         % increase in learning rate
        tp(6)=0,75;         % decrease in learning rate
        tp(7)=0.95;         % momentum constant
        tp(8)=1.01;         % error ratio

        [wc1,bc1,wc2,bc2,te,tr]=trainbpx(wc1,bc1,'tranfun',wc2,bc2,'tranfu
                                       n',p,t,tp);    % train the network

        save wc1,bc1,wc2,bc2;    % save the weights and biases
```

Program -V

```
% Simulation of NNM & NNC
load   save wm1,bm1,wm2,bm2;        % load the weights
p;
[vt,slip] = simuff( p,wm1,bm1, 'tranfun',wm2,bm2, 'tranfun');
                                     % output of the mapper
q;
load   save wc1,bc1,wc2,bc2;
[efd,ug]= simuff( p,wc1,bc1, 'tranfun',wc2,bc2, 'tranfun');
                                          % output of the controller
plot ( vt );
plot ( slip );
```