2009

Dynamic analysis of synchronous machine using neural network based characterization clustering and pattern recognition

Rashed Mazhar
University of Windsor

DYNAMIC ANALYSIS OF SYNCHRONOUS MACHINE USING NEURAL NETWORK BASED CHARACTERIZATION, CLUSTERING & PATTERN RECOGNITION

By

Rashed Mazhar

A Thesis
Submitted to the Faculty of Graduate Studies
Through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for the
Degree of Master of Applied Science at the
University of Windsor

Windsor, Ontario, Canada

2009

Your file Votre référence
ISBN: 978-0-494-82079-7
Our file Notre référence
ISBN: 978-0-494-82079-7

AUTHOR'S DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Synchronous generators form the principal source of electric energy in power systems. Dynamic analysis for transient condition of a synchronous machine is done under different fault conditions. Synchronous machine models are simulated numerically based on mathematical models where saturation on main flux was ignored in one model and taken into account in another. The developed models were compared and scrutinized for transient conditions under different kind of faults – loss of field (LOF), disturbance in torque (DIT) & short circuit (SC). The simulation was done for LOF and DIT for different levels of fault and time durations, whereas, for SC simulation was done for different time durations. The model is also scrutinized for stability stipulations.

Based on the synchronous machine model, a neural network model of synchronous machine is developed using neural network based characterization. The model is trained to approximate different transient conditions; such as – loss of field, disturbance in torque and short circuit conditions. In the case of multiple or mixture of different kinds of faults, neural network based clustering is used to distinguish and identify specific fault conditions by looking at the behaviour of the load angle. By observing the weight distribution pattern of the Self Organizing Map (SOM) space, specific kinds of faults is recognized. Neural network patter identification is used to identify and specify unknown fault patterns. Once the faults are identified neural network pattern identification is used to recognize and indicate the level or time duration of the fault.

DEDICATION

Dr. Rashida Akhter

Dr. Md. Mazharul Huque Khan

Mushfiq Mohammad Mazhar

Musabbir Mohammed Mazhar

Selina Akhter

ACKNOWLEDGEMENT

I wish to express my sincere gratitude to my advisor Dr. Narayan Kar for his assistance at every step of the way. His guidance has had an immense influence on my professional growth and without his technical expertise, reviews, and criticism it would not have been possible to shape this thesis. It was a rewarding experience working with him. I would also like to thank my committee members Dr. Lee and Dr. Khalid for their valuable suggestions and guidance in the completion of this work.

I would like to show my appreciation for my adorable brothers and my dear friends who made strenuous times seem easy and turned stressful days into fun. Their love and support will always be invaluable.

In the end I want to thank my fellow graduate students in the Electric Machines and Drives Research Lab for their support and encouragement. Working in their friendly company was a memorable experience.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

Generally symbols have been defined locally. The list of principle symbols is given below.

R_a	Stator winding resistance
R_{kd1}, R_{kq1}	d- and q-axis 1st damper resistances
R_{kd2}, R_{kq2}	d- and q-axis 2nd damper resistances
R_{fd}, R_{kq3}	Field and 3rd q-axis damper resistances
L_d, L_q	d- and q-axis synchronous Inductances
L_{kd1}, L_{kq1}	d- and q-axis 1st damper Inductances
L_{kd2}, L_{kq2}	d- and q-axis 2nd damper Inductances
L_{fd}, L_{kq3}	Field and q-axis 3rd damper Inductances
L_l	Stator leakage Inductances

1 INTRODUCTION

1.1 Background

The synchronous machine has long been the most important of the electromechanical power-conversion devices, playing a key role both in the production of electricity and in certain special drive applications. The history of the synchronous machine is now more than 100 years old. Within this span of time its power capacity has grown enormously, and it has established itself as a major player in the conversion of energy. Its beginnings are found in the closing decades of the 1800s, when innovatory engineers in several different countries showed courage, conviction and far-sightedness as they worked on its early development.

The beginning was in the 1880s. At first, stationary poles were used, with the poles surrounding a rotating ring armature. This was known as the external-pole type. An important milestone was the 'three-phase dynamo' derived from the direct-current machine with the Thomson-Houston armature. In 1887, the first three-phase synchronous generator shown in Figure 1.1 was built, which produced about 2.8 kW at 960 rev/min, corresponding to a frequency of 32 Hz.

Figure 1.1. The first three-phase synchronous machine built by Friedrich August Haselwander in 1887 (Photo: Deutsches Museum, Munich).

1891 was the year in which the three-phase synchronous machine passed its first big test and made its actual breakthrough. The scene was the Frankfurt Exposition, the event the great experiment whereby 300 hp was transmitted from the hydroelectric power plant at Lauffenam Neckar, 175km away, via three-phase current transmission. It was an event that drew worldwide attention and acclaim. The appearances of powerful steam turbines are at about the beginning of the 20th century. In 1901, the first actual turbogenerator was built by Charles E. Brown [1].

We have to thank the Americans for building the first hydrogen-cooled machines. They started in 1928 with a synchronous compensator, and in 1936 they put the first 3,600 rev/min hydrogen-cooled turbogenerator into commercial operation [1]. The first hydrogen-cooled turbine generator developed by the GE Company went into service in 1937, and hydrogen-cooled machines were able to satisfy the power output needs for many years. Between 1950 and 1960, manufacturers developed a broad range of direct cooling methods.

A milestone of the 1970s was the appearance of superconducting (SC) synchronous generator technology. A prototype of two-pole "utilitytype" generator was built during the early 1970s using low temperature superconducting (LTS) wires. A 5-MVA generator was developed and successfully tested in 1972. The purpose of these activities was to assess the technical feasibility of SC generators for long-term reliable operation on electric power systems. In 1979 a 20 MVA two-pole 3,600 r/min turbine generator for utility applications was designed, built, and load tested which was the largest SC generator to be fully load tested.

This LTS conductor technology was used in the design of an SC rotor for a synchronous turbo-generator. This rotor was essentially designed as a 250 MW machine with an active length of 2 m and an overall length of 3 m, but used a larger diameter of a 1200-MW machine (1.06 m) [2].

Great improvements of computers mark the 90's; powerful softwares were developed to design and analyze the synchronous generators. In 1995, K.W. Cowan presented advanced computational techniques involving computational fluid dynamics (CFD) and electromagnetic and thermal finite element analyses to predict the thermal

performance of prototype hydrogen cooled generator. During the last years of 1990s, the SuperGM project, which was launched by the Japan New Energy and Industrial Technology Development Organization in 1988, resulted in three models of superconducting rotors and a conventional stator. Between October 1998 and June 1999, this model machine was connected to a commercial power grid for the first time in the world to study basic performance in an actual electric power system.

Application of high temperature superconducting (HTS) materials in synchronous generators was a great milestone in this technology. In the mid–1990s, GE conducted design studies on HTS generators and built and tested an HTS prototype coil [2]. Last years of 1990s encountered the appearance of the powerformer technology. The idea of electrical generation in high voltages was proposed in the beginning of 1998 by Dr Mats Leijon from the ABB Corporate Research in Sweden. A new type of generator offered a possibility to build high voltage generators, which could be directly connected to the power transmission systems without any step-up transformer. In 1998, the first powerformer was installed in the Porjus power plant in the north of Sweden with the rating voltage of 45 kV and the rating power of 11 MVA [3].

Synchronous generators form the principal source of electric energy in power systems. Many large loads are driven by synchronous motors. Synchronous condensers are sometimes used as a means of providing reactive power compensation and controlling voltage. These devices operate on the same principle and are collectively referred to as synchronous machines. The power system stability problem is largely one of keeping interconnected synchronous machines in synchronism. Therefore, an understanding of their characteristics and accurate modeling of their dynamic performance are of fundamental importance to the study of power system stability.

The modeling and analysis of the synchronous machine has always been a challenge. The problem was worked on intensely in 1920s and 1930s [4]. [5] and has been the subject of several more recent investigations [5]. [7]. The theory and performance of synchronous machines have also been covered in a number of books [8], [9].

Synchronous machines while generating power are usually connected to a grid. As one of the prime requirements of synchronous machines is to run them in synchronous speed, as any distortion from synchronism can lead to instability of the grid i.e. the system. Synchronous machines while operating in generation mode are subjected to different kinds of faults or disturbances which can lead to potential speed distortion and ultimately instability of the system. To prevent this there are different kinds of precaution that have been taken. A lot of these precautions involve implementation of protection relays which depends on fault detection and analysis.

Nowadays, to connect a synchronous machine to a system for testing purposes are not so practical. Same goes for analyzing and investigating it for fault detection for the enormity and complexity of the machines as well as the complexity of the power system and the importance of its stability. With the rapid and vast development of computer based analysis tools the solution has to come as a package where the system is already analyzed the outcome is expected. In the field of fault detection there are different kinds of common occurrences in faults under which the machine behaviors should be analyzed and possible solution should be in effect. Of the faults very common occurrences are loss of field or excitation, disturbances in input torque, short circuit faults etc. For the sake of power system stability is absolutely nonnegotiable to have a proper understanding of how a machine going to behave under any of the faults and as well to identify what kind of fault is in incidence.

For the purpose synchronous machine computer aided analysis is done by simulating synchronous machine models and observing its dynamic behavior if different kinds of fault is initiated. It is also of utmost importance to detect what kind of fault is in occurrence by just looking at the machine activities. Under certain situation if there are multiple fault occurrences it is also essential to filter different kinds of faults to distinguish and identify them.

Up to the point different type of synchronous machine models are in effect, which are good approximations of the actual system. They are at most of the cases being simulated where the simulation is also a good approximation of the actual mathematical

machine model solution. Fault detection and distinguishing can be tricky under certain situations where fault specific behaviors are not very well known.

1.2 Research Objectives

The objective of this research is to understand and realize synchronous machine dynamic performances and to propose and design a better modeling of synchronous machine for the purpose; to understand behavior of synchronous machine performance under different kinds of fault and study system stability under these conditions; to identify and to be able to distinguish between different kinds of faults.

In this research work, artificial neural network has been used as a tool for the purpose:

- For approximation and characterization of synchronous machine dynamic behavior under different fault conditions
- For fault distinguishing and filtering under mixed or multiple fault occurrence
- For fault detection and identification to various details by looking at machine behavior

To achieve these purposes neural network based characterization, clustering and pattern recognition has been used.

1.3 Thesis Outline

This thesis is organized as follows:

Chapter 2: In this chapter, synchronous machine and its model details is being defined. Here synchronous machine is described from its operational point of view, constructional point of view and other theories related to it. Synchronous machine mathematical model is being depicted which is later used in chapter 4 for simulation purposes.

Chapter 3: Artificial neural network with its understanding and different aspects is focus of the chapter. Special types of artificial neural networks and there

attributes being scrutinized to comprehend there implicational and contextual properties.

Chapter 4: Synchronous machine model described in chapter 2 is simulated and dynamic analysis is performed. Simulation and result from the simulation is presented with detailed description and explanation.

Chapter 5: In this chapter, neural network characterization is used to approximate synchronous machine model using neural networks. The simulation of the approximation is presented under various dynamic conditions.

Chapter 6: In this chapter, neural network clustering is used to filter and distinguish between different kinds of faults in the case of multiple fault situations. The simulation results are presented.

Chapter 7: In this chapter, neural network pattern recognition technique is used to detect faults by looking at machine behaviors. Fault detection is done between different kinds and levels of faults. The simulations and findings are presented in end of the chapter.

Chapter 8: Findings of this research work is summarized in this chapter.

1.4 References

[1] G. Neidhofer, "The evolution of the synchronous machine," *Engineering Science and Education Journal*, pp. 239-248, October 1992.

[2] S. Kalsi, K. Weeber, H. Takesue, C. Lewis, "Development status of rotating machines employing superconducting field windings," *Proceeding of the IEEE*, vol. 92, no. 10, pp. 1688-1704, October 2004.

[3] M. Leijon, M. Dahlgren, L. Walfridsson, L. Ming and A. Jaksts, "A recent development in the electrical insulation systems of generators and transformers," *IEEE Electrical Insulation Magazine*, vol. 17, no. 3, pp. 10-15, May/June 2001.

[4] R.H. Park, "Two-Reaction Theory of Synchronous Machines - Generalized Method of Analysis - Part I," *AIEE Trans.*, vol. 48, pp. 716-727, 1929.

[5] R.H. Park, "Two-Reaction Theory of Synchronous Machines - Part II," *AIEE Trans.*, vol. 52, pp. 352-355, 1933.

[6] G. Shackshaft and P.B. Henser. "Model of Generator Saturation for Use in Power System Studies," *Proc. IEE (London)*, vol. 126, no. 8, pp. 759-763. 1979.

[7] EPRI Report EL-3359, "Improvement in Accuracy of Prediction of Electrical Machine Constants, and Generator Model for Subsynchronous Resonance Conditions," *Final Report of EPRI Projects RP 1288-1 and RP*. vols. 1, 2 and 3. (Prepared by General Electric Company), 1984.

[8] E.W Kimbark, *Power System Stability. Vol. III: Synchronous Machines*. John Wiley & Sons, 1956.

[9] A.E. Fitzgerald and C. Kingsley, *Electric Machinery, Second Edition*, McGraw-Hill, 1961.

2 SYNCHRONOUS MACHINE MODELING

2.1 Introduction

Synchronous machine is the most used machine in the purpose of electric power generation in the world. That is most of the energy conversion where mechanical power is converted into electrical power, large scale synchronous machine are in use. It's an AC machine where the rotor of the machine is in synchronism with the rotating stator magnetic field which refers its being in synchronism to the electrical frequency.

To understand the modeling of machine one has to understand a machine's construction, the fundamentals it operates on, mathematical model etc. To be able to analyze a machine one have to realize their underlying relationships. In the next section the key aspects of synchronous machine is portrayed and an effort was made to interrelate them. Also a synchronous machine mathematical model is described which is developed based on a standard IEEE model [1].

2.2 Theory and Modeling of Synchronous Machine

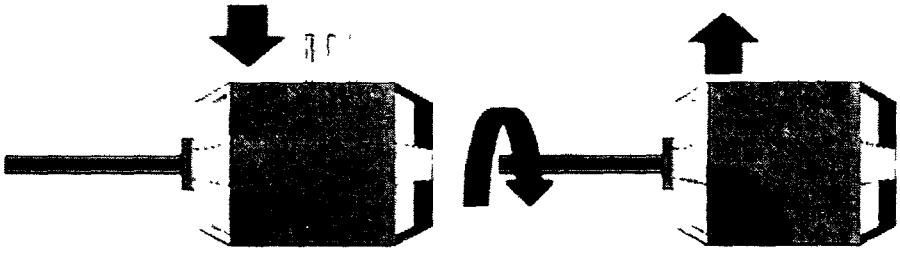
2.2.1 Constructional features

From mechanical point of view a synchronous machine has basically two parts: stator and rotor. The stator is the stationary part which has a three phase winding which is spatially distributed and either Y-connected or Δ-connected. Stator in a synchronous machine is the armature as the larger current flows through it. The rotor is the rotating part of the machine which has a DC winding. That is a DC power supply powers the rotor to make it act as an electromagnet. Hence, the rotor in a synchronous machine is the field [2].

2.2.2 Operating principles

Synchronous machine is an electromechanical energy conversion unit, which can convert mechanical energy to electrical and electrical energy to mechanical. When it converts electrical energy to mechanical energy it is said to be operating in motoring mode shown in Figure 2.1(a) and when it is converting mechanical energy to electrical it is called to be operating in generating mode shown in Figure 2.1(b). In most of the cases they are used as generators because of their high efficiency.

To understand the operating principles of synchronous machine it is assumed that the machine is operating in motoring mode. Once understood the motoring mode the generating mode works in the same way. except the direction of the operation is completely opposite. In motoring mode, a three phase AC power is supplied as in Figure 2.2. The three phase power supply creates a rotating magnetic field. The speed of the rotating magnetic field is synchronous to the frequency of the AC power supply and the speed depends on the number of poles in the rotor. As the electrical frequency and the number of poles in a synchronous machine are constant, the speed is as well [2]. The speed of the magnetic field can be calculated as,



(a) (b)

Figure 2.1. Synchronous machine operation. (a) Motoring mode (b) Generating mode.

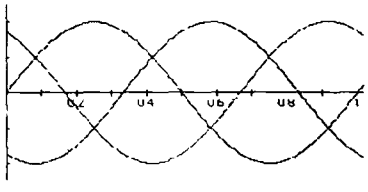


Figure 2.2. Three-phase AC signal.

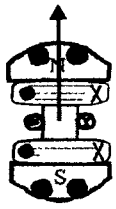


Figure 2.3. Field winding in the rotor.

—— (2.1)

Where,

is the electrical frequency in per second (Hz)

P is the number of poles

N is the synchronous speed in revolution per minute (rpm)

The DC power supply in the rotor winding as in Figure 2.3 makes the rotor act as an electromagnet; hence the magnetic field is created. The rotating magnetic field in the stator circuit cuts the magnetic field from that field winding of the stator; as a result they try to align with each other. As the rotating magnetic field continuous to rotate the rotor magnetic field follows, as a result the rotor starts rotating; hence the mechanical rotation.

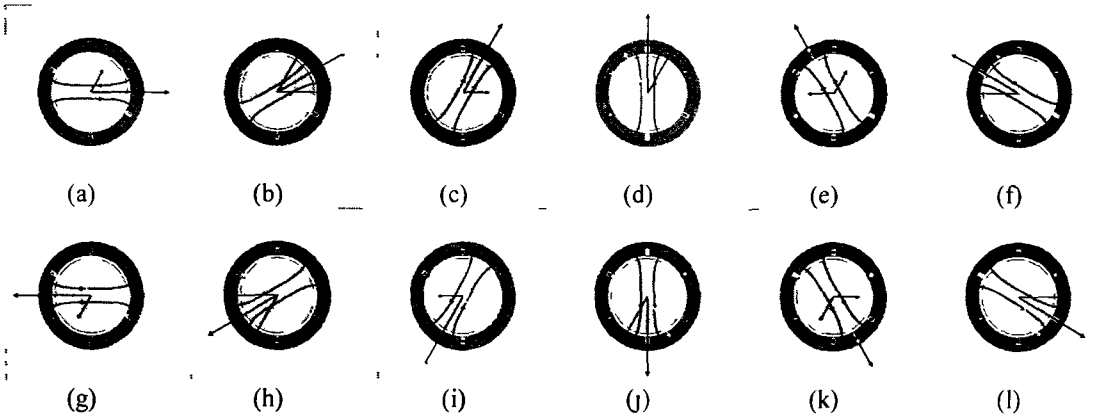


Figure 2.4. Rotating magnetic field of a synchronous machine.

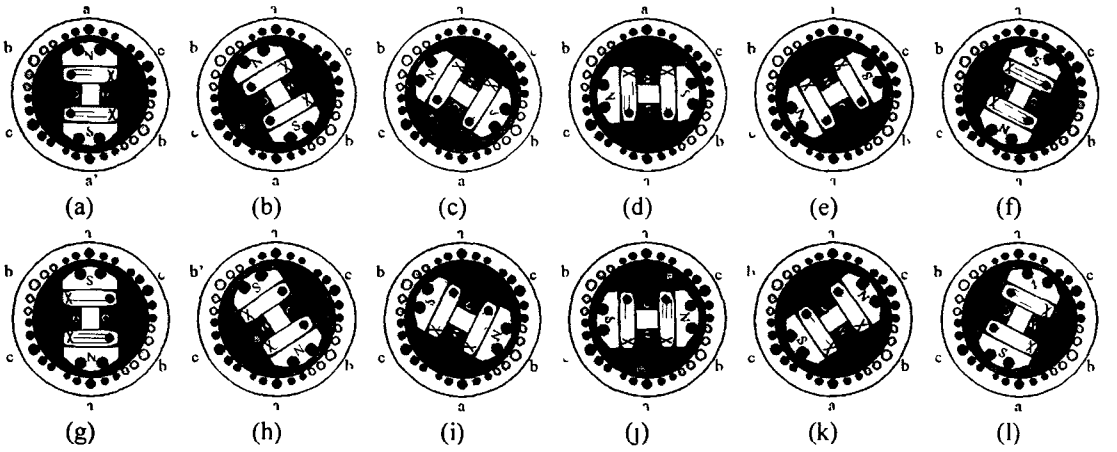


Figure 2.5. Synchronous machine rotation.

Rotation of the magnetic field in the stator circuit is shown in Figure 2.4(a)-(l). The rotation of the rotor because of the electromagnetic induction is shown in Figure 2.5(a)-(l). It is evident from the Figures 4 and 5 that the rotation of synchronous machine rotor is synchronous to the rotating magnetic field.

2.2.3 Reference frame theorem

The understanding of synchronous machine mathematical model one needs to have a proper understanding of reference frame theorem. Before getting in to the details of the reference frame theorem of a synchronous machine let us look at the machine equations from organizational point of view. From one point of view the mathematical model has two basic sets of equation describing the whole model – the electrical equations and the mechanical equations.

Now looking at the electrical part of the machine, the model as per machine structural construction has two distinct parts – the rotor and stator, thus, a set of equations that describes the stator part and another set of equations that describes the rotor part. Since, the rotor and stator of the machine are linked through magnetic flux while operating, the equations describing both stator and rotor are interconnected.

In describing the mathematical model problem arises as the stator is stationary and the rotor is rotating, and one has to inter-link the equation to make sense out of them; which calls for taking either stator or rotor as reference. In describing these equations whether the rotor or the stator or any other variable is taken as a reference is realized is expressed through reference frame theory.

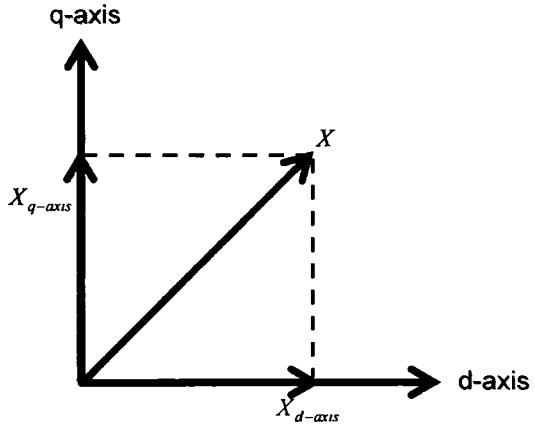


Figure 2.6. Reference frame.

Like many other coordinate system the reference frame theory is primarily defined by two axes as shown in Figure 2.6: direct and quadrature axes. All the vectors in the mathematical model of a synchronous machine are dissolved to these two axes. So, as in Figure 2.6, if an arbitrary vector X is assumed with an angle θ with respect to direct axis, it will have to be dissolved in two components – the d-axis component: $X_d = X\cos\theta$ and the q-axis component: $X_q = X\sin\theta$. All the vectors in the space which describe the machine operation are thus resolved into d and q axis components.

2.2.4 Per unit system

A per-unit system is the expression of system quantities as fractions of a defined base unit quantity. Calculations are simplified because quantities expressed as per-unit are the same regardless of the voltage level. Similar types of apparatus will have impedances, voltage drops and losses that are the same when expressed as a per-unit fraction of the equipment rating, even if the unit size varies widely. Conversion of per-unit quantities to volts, ohms, or amperes requires knowledge of the base that the per-unit quantities were referenced to.

A per-unit system provides units for power, voltage, current, impedance, and admittance. Only two of these are independent, usually power and voltage. All quantities are specified as multiples of selected base values. Per unit system is a way of normalizing machine parameters so that one can make a comparison between machines with different specification. In this research work all the values are calculated in per unit system [3].

$$Per\ unit\ value = \frac{Actual\ value}{Base\ value}$$

2.3 Mathematical Modeling

Mathematical model of a machine is realizing the machine in terms of a set of differential equation and polynomials. To understand a machine model and to relate and realize the relationship between the machine constructions, their broken down parts, operating principles and how the electrical and the mechanical vectors and variables in machine equations interact, different approaches are used; different way of telling the

same story using different point of views. This includes the circuit diagram, machine equations, phasor diagram etc. In this model synchronous reference frame is used to depict the machine equations.

2.3.1 d-axis mathematical modeling

The d-axis circuit diagram of the synchronous machine model is shown in Figure 2.7 which describes the d-axis electrical model [1], [2]. In this model, one field winding and two damper windings are considered in d-axis rotor circuit. The machine is assumed to be in a generation mode. All the currents in the machine should be assumed in an outward direction that is anti clockwise in the loops.

Looking at the circuit diagram to describe the relationship two sets of equations is being used. The first set are the voltage equations, which are differential equations relating voltage and flux. The second set of equations is flux equations which relates current and flux. The first equation in each set represents to the stator electrical model and the later three the rotor electrical model. The second equation in each set is representing the field circuit and the later two the damper circuit.

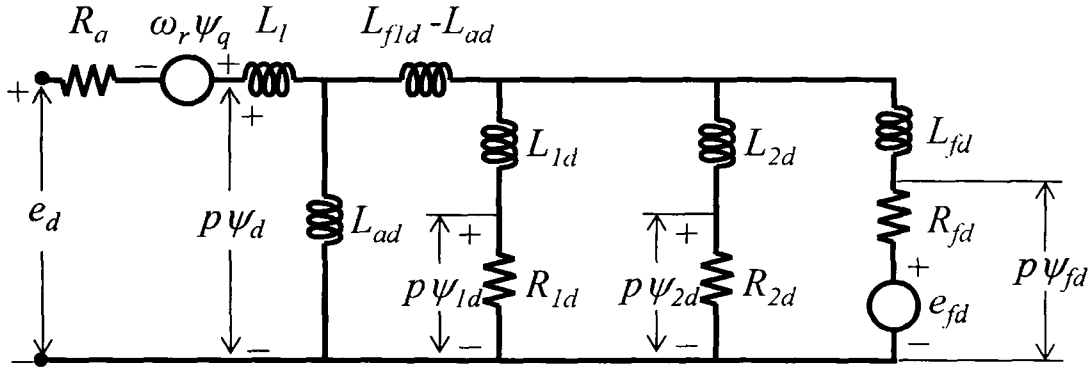


Figure 2.7. d-axis circuit diagram.

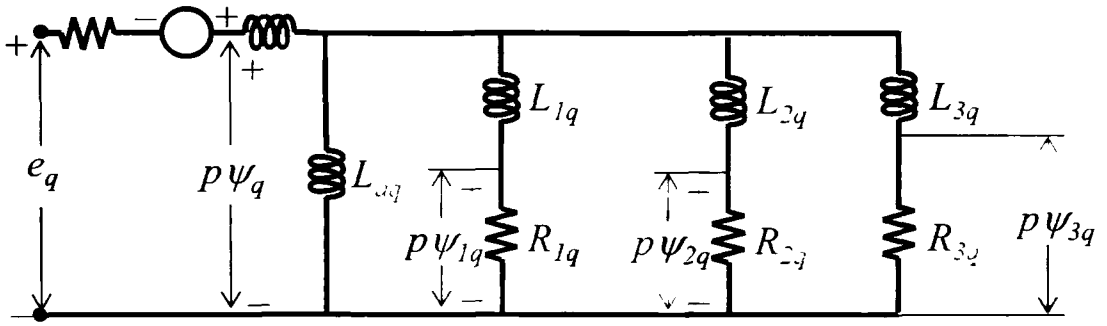


Figure 2.8. q-axis circuit diagram.

Voltage equations:

$$\left.\begin{aligned} e_d &= p\psi_d - \psi_q\omega_r - R_a i_a \\ e_{fd} &= R_{fd} i_{fd} + p\psi_{fd} \\ 0 &= R_{1d} i_{1d} + p\psi_{1d} \\ 0 &= R_{2d} i_{2d} + p\psi_{2d} \end{aligned}\right\} \quad (2.2)$$

Flux equations:

$$\left.\begin{aligned} \psi_d &= (L_{ad} + L_l) i_d - L_{ad} i_{fd} - L_{ad} i_{1d} \\ \psi_{fd} &= L_{fd} i_{fd} + L_{f1d} i_{1d} - L_{ad} i_d \\ \psi_{1d} &= L_{f1d} i_{fd} + L_{11d} i_{1d} - L_{ad} i_d \\ \psi_{2d} &= L_{f2d} i_{fd} + L_{22d} i_{2d} - L_{ad} i_d \end{aligned}\right\} \quad (2.3)$$

2.3.2 q-axis mathematical modeling

The q-axis circuit diagram of the synchronous machine model is shown in Figure 2.8 which describes the q-axis electrical model [1]. [2]. In this model, three damper windings are considered in q-axis rotor circuit. The machine is assumed to be in a generation mode. Looking at the circuit diagram to describe the relationship two sets of equations is being used. The first set are the voltage equations, which are differential equations relating voltage and flux. The second set of equations is flux equations which relates current and flux. The first equation in each set represents to the stator electrical model and the later three the rotor electrical model.

Voltage equations:

$$\left.\begin{aligned} e_q &= p\psi_q + \psi_d \omega_r - R_a i_q \\ 0 &= R_{1q} i_{1q} + p\psi_{1q} \\ 0 &= R_{2q} i_{2q} + p\psi_{2q} \\ 0 &= R_{3q} i_{3q} + p\psi_{3q} \end{aligned}\right\} \tag{2.4}$$

Flux equations:

$$\left.\begin{aligned} \psi_q &= (L_{aq} + L_l) i_q + L_{aq} i_{1q} + L_{aq} i_{2q} \\ \psi_{1q} &= L_{11q} i_{1q} + L_{aq}(i_{2q} + i_{3q}) - L_{aq} i_q \\ \psi_{2q} &= L_{aq}(i_{1q} + i_{3q}) + L_{22q} i_{2q} - L_{aq} i_q \\ \psi_{3q} &= L_{aq}(i_{1q} + i_{2q}) + L_{22q} i_{2q} - L_{aq} i \end{aligned}\right\} \tag{2.5}$$

2.3.3 Steady-state operation

The phasor diagram of a synchronous machine shows the relationship of synchronous machine voltage and current with the phase differences. It is necessarily useful for realizing the steady state condition of a synchronous machine, which is used as an initial condition of the machine simulation. From the voltage diagram in Figure 2.9, d- and q-axis terminal voltages can be found which later are being used to calculate the load angle [4], [5]. The load angle is used to calculate the initial conditions for machine operation.

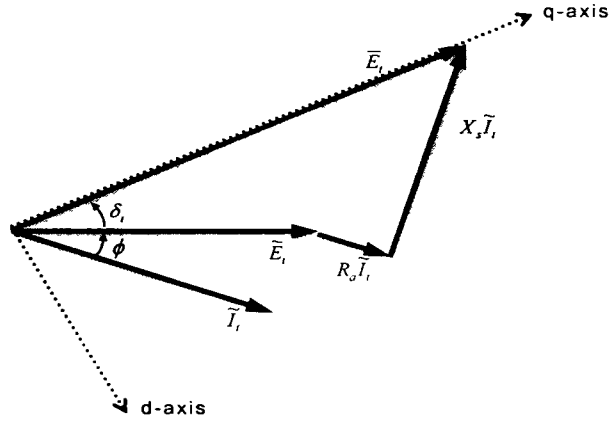


Figure 2.9. Phasor diagram for calculating initial conditions.

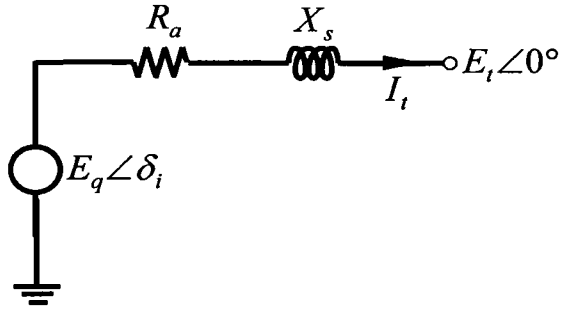


Figure 2.10. Simplified circuit diagram.

Calculation of load angle:

$$\left.\begin{aligned} &V_d = V_t \sin\delta,\ V_q = V_t \cos\delta \\ &\delta = \tan^{-1}\left[\frac{I_t.L_q.\cos\theta - I_t.R_a.\sin\theta}{V_t + I_t.L_q.\sin\theta + I_t.R_a.\cos\theta}\right] \end{aligned}\right\} \quad (2.6)$$

Where, $\omega_0 = 2\pi f$, $f = 60\,Hz$, $p = \frac{d}{dt}$

2.3.4 Mechanical equations

The mechanical part of the mathematical model describes the mechanical phenomenon of the machine as well as relates the mechanical effect with speed and load angle. As the electrical model of the machine depends on load angle and speed to calculate different parameters, the load angle and the speed are the relating factor between mechanical and electrical model.

$$\left.\begin{aligned} &\frac{d\delta}{dt} = \omega_0 \Delta\overline{\omega}_r \\ &T_e = \psi_d i_q + \psi_q i_d \\ &\frac{d\Delta\overline{\omega}_r}{dt} = \frac{1}{2H}\left(T_m - T_e\right) \end{aligned}\right\} \quad (2.7)$$

2.3.5 Current flux relationship in matrix form

Another way of looking at machine equations is a matrix form. Matrix form is just manipulation of the existing equations and representing in terms of matrix multiplication [6]. This is especially useful when programs are written in the purpose of numerical simulation. In matrix form the whole model is portrayed in three distinct matrixes; the current, flux and inductance matrix:

$$I = L^{-1}\psi \tag{2.8}$$

Where:

$$I = \left[i_d\ i_{kd1}\ i_{kd2}\ i_{fd}\ i_q\ i_{kq1}\ i_{kq2}\ i_{kq3}\right]^T$$

$$\psi = \left[\psi_d\ \psi_{kd1}\ \psi_{kd2}\ \psi_{fd}\ \psi_{kq1}\ \psi_{kq2}\ \psi_{kq3}\right]^T$$

$$L=\begin{bmatrix} -L_d & L_{md} & L_{md} & L_{md} & 0 & 0 & 0 & 0 \\ -L_{md} & L_{md}+L_{kd1} & L_{md} & L_{md} & 0 & 0 & 0 & 0 \\ -L_{md} & L_{md} & L_{md}+L_{kd2} & L_{md} & 0 & 0 & 0 & 0 \\ -L_{md} & L_{md} & L_{md} & L_{md}+L_{fd} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -L_q & L_{mq} & L_{mq} & L_{mq} \\ 0 & 0 & 0 & 0 & -L_{mq} & L_{mq}+L_{kq1} & X_{mq} & L_{mq} \\ 0 & 0 & 0 & 0 & -L_{mq} & L_{mq} & L_{mq}+L_{kq2} & L_{mq} \\ 0 & 0 & 0 & 0 & -L_{mq} & L_{mq} & L_{mq} & L_{mq}+L_{kq3} \end{bmatrix}$$

2.3.6 Internal control system

Synchronous generators are usually connected to the grid. This means that they have constant terminal voltage with a specific loading condition. While synchronous machine is operating under a grid it is usually generating power while running in synchronous speed. It is important to understand that synchronism in speed of a synchronous machine is a requirement, as any distortion in synchronism can lead to system instability. If a machine goes to a super synchronous speed for any type of

disturbance it usually gives away its extra kinetic energy as electrical energy to the grid and tends to comes back to synchronous speed. On the other hand, if it goes to a sub-synchronous speed, it absorbs some of the electrical energy and tends to speed up to go to the synchronous speed [2]. This tendency of synchronous machine to operate in synchronous speed is can be views as the internal control system which is shown in Figure 2.11. The control equations are:

$$p(P_m) = \frac{K_p}{\omega_0 K_I}(\omega_0 - \omega_m) + \frac{1}{K_I}(P_{m0} - P_m) \tag{2.9}$$

$$T_m = \frac{\omega_0 P_m}{\omega_m} \tag{2.10}$$

2.4 Saturation

Saturation is one of the most common occurrences in the nature; it is also true for electric machines with no exception in synchronous machines. It has been seen that taking saturation into account gives more accurate and realistic results [7], [8]. Two models of synchronous machine are developed in this research work. In the first model saturation is ignored and in the second model it's taken into account.

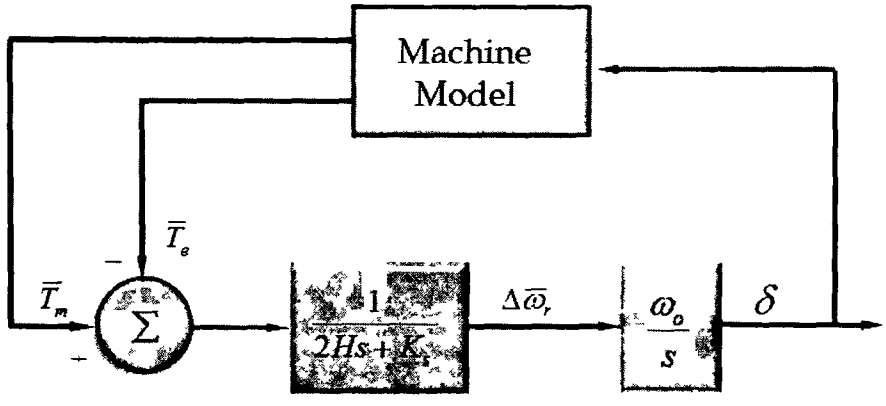


Figure 2.11. Internal control loop.

2.4.1 Unsaturated model

In the unsaturated mode the d-axis saturation and the q-axis saturation ignored, that is the d- and q-axis magnetizing reactances, X_{md} and X_{mq}, are assumed to be equal to their unsaturated values.

2.4.2 Saturated model

In electric machines, saturation is of basic two kinds: leakage flux saturation and main flux saturation.

2.4.2.1 Leakage flux saturation

Leakage flux saturation is defined by the saturation in the leakage flux of a machine. In this research work the leakage flux saturation is ignored. This is because it has negligible effect on the machine performance in comparison to the machine main flux saturation.

2.4.2.2 Main flux saturation

In this case, both d- and q-axis saturation are considered. The unsaturated d- and q-axis magnetizing reactances are replaced by their corresponding saturated values. These d- and q-axis saturated magnetizing reactances, X_{mds} and X_{mqs}, are obtained by modifying the corresponding unsaturated values, X_{mdu} and X_{mqu}, with two saturation factors calculated from the polynomials fitting the saturation curves. The d- and q-axis magnetizing ampere-turns (AT_d, AT_q) are used to locate the operating points on the d- and q-axis saturation characteristics respectively [9]-[11].

By applying the procedure described above, the transient performance of synchronous machines considering the saturation along the direct and quadrature axes can be calculated. However, in this case, an iterative technique has to be applied to determine the transient performance as the saturated d- and q-axis magnetizing reactances are a function of magnetizing current [12].

d-axis saturation

$$\psi_{ds} = f(AT_d) = -0.1501\,AT_d^3 + 0.0383\,AT_d^2 + 1.0283\,AT_d - 0.0007$$

$$X_{mds} = \frac{\psi_{ds}}{AT_d} \tag{2.11}$$

q-axis saturation

$$\psi_{qs} = f(AT_q) = -0.0155\,AT_q^{\,3} - 0.2246\,AT_q^{\,2} + 1.066\,AT_q - 0.0012$$

$$X_{mqs} = \frac{\psi_{qs}}{AT_q} \tag{2.12}$$

2.5 Rotor Angle

The electrical angular displacement of the rotor relative to its terminal is defined as the rotor angle. The rotor angle is the displacement of the rotor generally referenced to the maximum positive value of the fundamental component of the terminal voltage. Therefore, the rotor angle expressed in radian is,

$$\delta = \theta_r - \theta_e \tag{2.13}$$

Where,

θ_r is the rotor angle

θ_e is the angle of electrical magnetic field

Speed (ω) of a synchronous machine can be found by differentiating θ; hence any disturbance in the speed of the machine can be interpreted as change in δ. In steady state condition the speed of a synchronous machine is a constant. As a result δ is constant. Any change in speed in the machine thus can be interpreted from change in δ.

In this research work, in disturbance introduced in the machine is realized by looking at δ as an output. Regardless of what the disturbance is, how much the machine is affected and how much the stability of the machine is disturbed is analyzed and interpreted by looking at how δ behaves.

2.6 References

[1] *IEEE Guide for Synchronous Generator Modeling Practices in Stability Analyses*, Std. 1110-1991.

[2] P Kundur, *Power System Stability and Control*, McGraw Hill, 2004.

[3] A.E.Fitzgerald, C. Kingsley, and S.D. Umans, *Electric Machinery*. McGraw-Hill, 1991.

[4] L. Wang; J. Jatskevich, and H.W. Dommel, "Re-examination of synchronous machine modeling techniques for electromagnetic transient simulations," *IEEE Transactions on Power Systems*, vol. 22, no. 3, pp. 1221 - 1230, Aug. 2007.

[5] M. Kakiuchi, S. Nagano, D. Hiramatsu, K. Koyanagi, K. Hirayama, Y. Uemura, T. Satoh, and K. Nagasaka, "A study of synchronous machine modeling about synchronizing phenomena," *IEEE International Conference on Electric Machines and Drives*, pp. 890 - 895, 15-15 May 2005.

[6] C. Ellis, H. Nouri, R. Ciric, and B. Miedzinsky, "Overview of the development, simplification and numerical analysis of synchronous machine models for stability studies," *42nd International Universities Power Engineering Conference*, pp. 1019 - 1023, 4-6 Sept. 2007

[7] N.C. Kar and A.M. El-Serafi, "Effect of the main flux saturation on the transient short-circuit performance of synchronous machines," *IEEE Canadian Conference on Electrical and Computer Engineering,* pp.629 – 632, May 1-4, 2005.

[8] A.M. El-Serafi and A.S. Abdallah, "Saturated synchronous reactances of synchronous machines," *IEEE Transactions on Energy Conversion,* vol.7, no. 3, pp.570 – 579, Sept. 1992.

[9] F.P. Mello, "Representation of saturation in synchronous machines," *IEEE Trans. on Power Engineering Society,* vol. PWRS-1, no. 4, p.8, 1986.

[10] E. Levi, "Modeling of magnetic saturation in smooth air-gap synchronous machine," *IEEE Trans. on Energy conversion,* Vol. 12, no. 2, pp. 151-156, June 1997.

[11] S.D. Pekarek, E.A. Walters, and B.T. Kuhn, "An efficient method of

representing saturation in physical variable models of synchronous machines", *IEEE Trans. on energy conversion,* vol.14, no. 1, pp. 72-79, March 1999.

[12] D. Hiramatsu, K. Hirayama, T. Tokumasu, Y. Uemura, M. Takabatake, Y Ishikawa, and A. Iwai, "Analysis of damper saturation characteristic on synchronous machine transient condition," *IEEE Power Engineering Society General Meeting*, pp. 1501-07, 2003.

3 ARTIFICIAL NEURAL NETWORK (ANN)

3.1 Introduction

An artificial neural network (ANN) as shown in Figure 3.1, often just called a "neural network" (NN), is a mathematical model or computational model based on biological neural networks. It attempts to simulate the structure, interconnections and interactions of the nerve cells of a biological brain, while have the capability to update its knowledge from experience. It consists of an interconnected group of artificial neurons and processes information using a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase.

In more practical terms neural networks are non-linear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data.

Neural network as an idea comes from observing central nervous system and its construction. The neurons in central nervous system along with their axons, dendrites and synapses constitutes for the most sophisticated information processing entity. In a neural network model, replicating the central nervous system, simple nodes called "neurons", "neurodes", "PEs" ("processing elements") or "units" are connected together to form a network of nodes. Hence it is called "neural network" These neural networks of simple processing elements (neurons), can exhibit complex global behavior, whereas its complexity and capability is determined by the number of connections, connections paradigm between the processing elements, and element parameters. The practical use of a neural network comes with algorithms designed to alter the strength (weights) of the connections in the network to produce a desired signal flow [1]-[4].

Even though, in the crams of theoretical neuroscience neural networks models are designed with an intention to emulate that of a central nervous system (CNS), artificial neural network as a term in concurrency is a subject to utilization to design models in statistics, cognitive psychology and artificial intelligence.

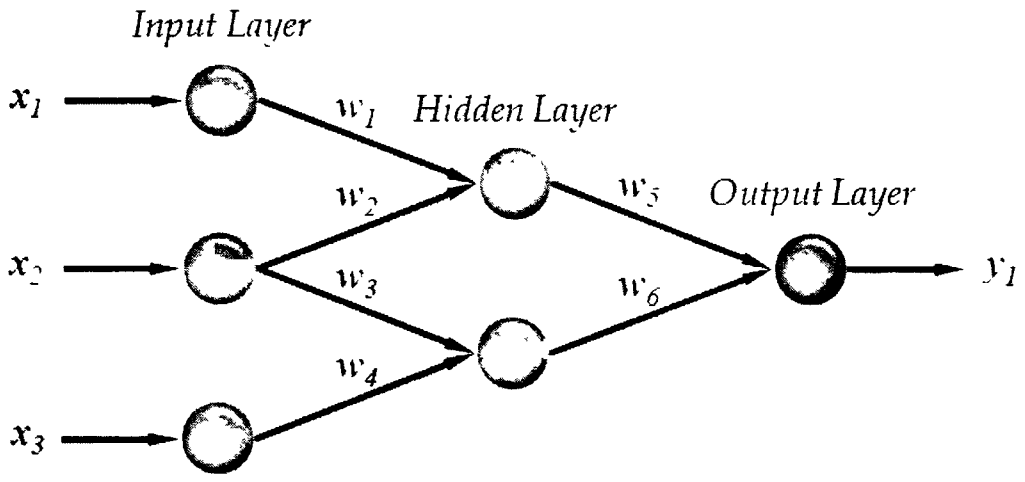


Figure 3.1. Artificial Neural Network (ANN).

Devoid of any qualm, biology has inspirited the invention of artificial neural network. In modern numerical implementation the approach is more or less discarded to fit the practical implicational needs based on signal processing and statistics. Both adaptive and non-adaptive elements are considered as used to realize large systems; though adaptive approach is more contextual in practical implementation which has a basis of non-linearity, distribution, parallelism, and local processing and adaptation.

3.2 Overview of ANN

3.2.1 Model

Artificial neural networks (ANNs) are in essence simple mathematical models defining a function, $f: X \rightarrow Y$ Any ANN model corresponds to a class of such functions.

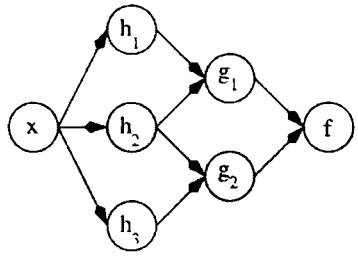


Figure 3.2. ANN dependency graph.

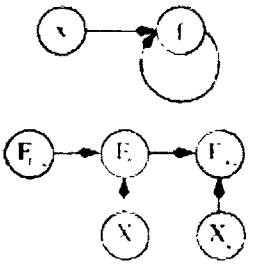


Figure 3.3. Recurrent ANN dependency graph.

Figure 3.2 illustrates essentials of a fundamental network structure with arrows depicting dependencies between variables, whereas $f(x)$ is defined as,

(3.1)

In this case $f(x)$ is a composite of a function $g_i(x)$ which can be represented as a simple vector,

It is a widely used type of composition known as the nonlinear weighted sum.

In similar fashion $g_i(x)$ can be shown as composite of other function depending on the network structure.

Interpretation of dependencies of the variables indicated by the arrows can be scrutinized in two ways, as in Figure 3.3 in case of function f.

Functional view: Input x is transformed into a 3-dimensional vector h, which is then transformed into a 2-dimensional vector g, which is finally transformed into f. This view is most commonly encountered in the context of optimization.

Probabilistic view: Random variable $F = f(G)$ depends upon the random variable $G = g(H)$, which depends upon $H = h(X)$, which depends upon the random variable X. This view is most commonly encountered in the context of graphical models.

Either of the views while implementation accord, has a naturally inhabited capability of enabling parallelism in some extent, which refers to the fact of them being independent of their inherited variables, hence, more or less equivalent quite to an extent; yet, there is some implied temporal dependencies.

The network with acyclic configuration in Figure 3.2 is usually known as feedforward neural network and the one with cyclic organization in Figure 3.3 is known as recurrent neural network [4]-[7].

3.2.2 Learning

No matter how interesting a neural network is with functions defining its structural paradigm, the most intriguing and captivating possibility lies in its competencies in learning ability [8]. Learning implementing a neural network optimally by observation means, given a specific task to solve a class of functions F, in order to find $f^* \epsilon F$.

This entails defining a cost function which is defined by $C: F \rightarrow \mathbb{R}$ such that, for the optimal solution f^*,

$$C(f^*) \leq C(f) \forall f \in F \tag{3.2}$$

That is, no solution has a cost less than the cost of the optimal solution.

Cost function C is an evaluation process through which it can determined to what extent a network is successful to learn a problem, in other words, how far the network is from the optimal solution of the problem dataset it is suppose to learn. The learning algorithm searches through the solution space with the intention of finding that of a smallest possible cost as appropriate.

For applications where the solution is dependent on some data, the cost must necessarily be a function of the observations; otherwise we would not be modeling anything related to the data. It is frequently defined as a statistic to which only approximations can be made. As a simple example considering the problem of finding the model f which minimizes C as, $C = E[(f(x) - y)^2]$ for data pairs (x,y) drawn from some distribution $\mathcal{D}$.

In practical situations, N samples from $\mathcal{D}$ will be available and thus, for the above example, we would only minimize, $\hat{C} = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2$ Thus, the cost is minimized over a sample of the data rather than the true data distribution.

For online learning parameter $N \to \infty$; as the learning progresses through time, the cost function is partially minimized with ingression of new data. Online learning is often used when $\mathcal{D}$ is fixed. In the case of finite dataset various customized versions of online learning are often being used [9], [10].

The use of problem specific cost functions is a frequent practice, although assigning ad hoc cost function in an arbitrary fashion can do the job. Obviously choosing problem specific cost function has its advantages in terms if addressing problem specific approximation; i.e. convexity in a model or probabilistic formulation the posterior probability of the model used as an inverse cost. In the end, choice of cost function is coherent to the task.

3.2.3 Learning paradigms

There are three major learning paradigms, for any given type of network architecture, each corresponding to a particular abstract learning task:

- Supervised learning
- Unsupervised learning
- Reinforcement learning.

3.2.3.1 Supervised learning

In supervised learning, a given set of example pair is (x, y), where $x \in X, y \in Y$ and the aim is to find a function $f: X \to Y$ in the allowed class of functions that matches the examples. In other words, it is inferred that the mapping is implied by the data; the cost function is related to the mismatch between the mapping and the data and it implicitly contains prior knowledge about the problem domain.

A commonly used cost is the mean-squared error which tries to minimize the average squared error between the network's output, $f(x)$, and the target value y over all the example pairs. When one tries to minimize this cost using gradient descent for the class of neural networks called Multi-Layer Perceptrons, one obtains the common and well-known back propagation algorithm for training neural networks[12], [13].

Pattern recognition (also known as classification) and regression (also known as function approximation) are the tasks which fall under the paradigm of supervised learning. The supervised learning paradigm is also applicable to sequential data (e.g., for speech and gesture recognition). This occurs in the form of a function that provides continuous feedback on the quality of solutions obtained up to that point [11].

3.2.3.2 Unsupervised learning

In unsupervised learning, some data x is given, and the cost function to be minimized can be any function of the data x and the network's output, f. The cost function is dependent on the task and as a priori assumption. For example, considering a model $f(x) = a$, where a is a constant and the cost $C = E[(x - f(x))^2]$. Minimizing this cost will give a value of that is equal to the mean of the data. The cost function can be in a form dependent on the application: In compression it could be related to the mutual information between x and y. In statistical modeling, it could be related to the posterior probability of the model given the data [12], [13].

Tasks that fall within the paradigm of unsupervised learning are in general estimation problems; the applications include clustering, the estimation of statistical distributions, compression and filtering [14].

3.2.3.3 Reinforcement learning

In reinforcement learning, data x is usually not given, but generated by an agent's interactions with the environment. At each point in time t, the agent performs an action y_t and the environment generates an observation x_t and an instantaneous cost C_t, according to some dynamics. The aim is to discover a policy for selecting actions that minimizes some measure of a long-term cost, i.e. the expected cumulative cost. The environment's dynamics and the long-term cost for each policy are usually unknown, but can be estimated.

More formally, the environment is modeled as a Markov decision process (MDP) with states $s_1, \ldots, s_n \in s$ and actions $a_1, \ldots, a_m \in a$ with the following probability distributions: the instantaneous cost distribution $P(c_t|s_t)$, the observation distribution $P(x_t|s_t)$ and the transition $P(s_t+1 \mid s_t, a_t)$, while a policy is defined as conditional

distribution over actions given the observations. Taken together, the two define a Markov chain (MC). The aim is to discover the policy that minimizes the cost, i.e. the MC for which the cost is minimal.

ANNs are frequently used in reinforcement learning as part of the overall algorithm. Tasks that fall within the paradigm of reinforcement learning are control problems, games and other sequential decision making tasks.

3.2.4 Learning algorithms

Most of the training algorithms can be scrutinized as a fundamental use of optimization theory statistical estimation. Presently there are numerous optimization algorithms are available for training a neural network, whereas choosing a model implies to selection of one from a set of allowed one, criteria being minimization of the cost function.

Gradient descent algorithm is a widespread tactics used when it comes to train an artificial neural network. In this method the derivative of the cost function with respect to the network parameters are considered and the change is done to those parameters in accordance with gradient-related direction. Among the other frequently used method evolutionary methods, simulated annealing and expectation-maximization and non-parametric methods are commonly used methods for training neural networks. Temporal perceptual learning relies on finding temporal relationships in sensory signal streams In an environment, statistically salient temporal correlations can be found by monitoring the arrival times of sensory signals. This is done by the perceptual networks [6], [9].

3.3 Real Life Applications

The utility of artificial neural network models lies in the fact that they can be used to infer a function from observations. This is particularly useful in applications where complexity of data or task makes design of such a function by hand impractical.

3.3.1 Applications of artificial neural networks

One of the most powerful applications of neural networks is function approximation, or regression analysis. Time series prediction and system modeling are typical examples of approximations or regressions. Classification is another popular neural network application paradigm. Pattern recognition, sequence recognition, novelty detection and sequential decision making are common type of classification example.

In the field of data processing neural networks are also used for various application processes. Typical data processing application are filtering, clustering, blind source separation and compression.

3.3.2 Application areas of artificial neural networks commonly spotted

Neural networks are applied in various fields to address different problems. Commonly spotted application areas of artificial neural networks are observed in system identification and control i.e. vehicle control, process control etc.; game-playing and decision making i.e. backgammon, chess, racing etc; pattern recognition i.e. radar systems, face identification, object recognition etc.; sequence recognition i.e. gesture, speech, handwritten text recognition etc.; medical diagnosis; financial applications i.e. automated trading systems; data mining i.e. knowledge discovery in databases ("KDD"); visualization; e-mail spam filtering; and many others.

3.4 Types of Neural Networks

3.4.1 Feedforward neural network

A feedforward neural network is an artificial neural network where connections between the units do not form a directed cycle. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network. In a feedforward network information always moves one direction; it never goes backwards.

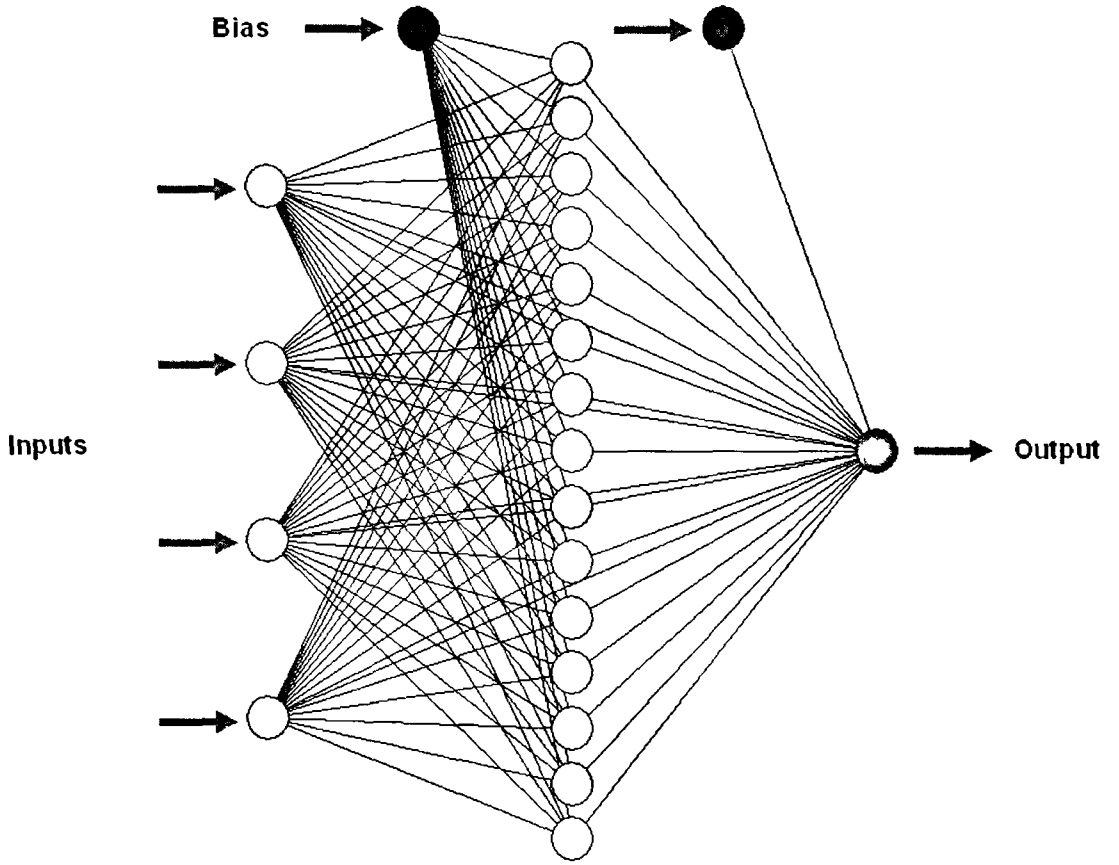


Figure 3.4. Feedforward neural network.

3.4.1.1 Single-layer perceptron

The earliest kind of neural network is a single-layer perceptron network, which consists of a single layer of output nodes; the inputs are fed directly to the outputs via a series of weights. In this way it can be considered the simplest kind of feedforward network. The sum of the products of the weights and the inputs is calculated in each node, and if the value is above some threshold (typically 0) the neuron fires and takes the activated value (typically 1); otherwise it takes the deactivated value (typically -1). Neurons with this kind of activation function are also called artificial neurons or linear threshold units. In the literature, the term perceptron often refers to networks consisting of just one of these units. A similar neuron was described by Warren McCulloch and Walter Pitts in the 1940s.

A perceptron can be created using any values for the activated and deactivated states as long as the threshold value lies between the two. Most perceptrons have outputs of 1 or -1 with a threshold of 0 and there is some evidence that such networks can be trained more quickly than networks created from nodes with different activation and deactivation values. Perceptrons can be trained by a simple learning algorithm that is usually called the delta rule. It calculates the errors between calculated output and sample output data, and uses this to create an adjustment to the weights, thus implementing a form of gradient descent.

Single-unit perceptrons are only capable of learning linearly separable patterns; in 1969 in a famous monograph entitled Perceptrons Marvin Minsky and Seymour Papert showed that it was impossible for a single-layer perceptron network to learn an XOR function. They conjectured (incorrectly) that a similar result would hold for a multi-layer perceptron network. Although a single threshold unit is quite limited in its computational power, it has been shown that networks of parallel threshold units can approximate any continuous function from a compact interval of the real numbers into the interval [-1,1].

A single-layer neural network can compute a continuous output instead of a step function. A common choice is the so-called logistic function:

$$y = \frac{1}{1+e^{-x}} \tag{3.3}$$

With this choice, the single-layer network is identical to the logistic regression model, widely used in statistical modeling. The logistic function is also known as the sigmoid function. It has a continuous derivative, which allows it to be used in backpropagation. This function is also preferred because its derivative is easily calculated: $y' = y(1 - y)$ (times df/dX, in general form, according to the Chain Rule)

3.4.1.2 Multi-layer perceptron

This class of networks consists of multiple layers of computational units, usually interconnected in a feedforward way. Each neuron in one layer has directed connections to the neurons of the subsequent layer. In many applications the units of these networks apply a sigmoid function as an activation function.

The universal approximation theorem for neural networks states that every continuous function that maps intervals of real numbers to some output interval of real numbers can be approximated arbitrarily closely by a multi-layer perceptron with just one hidden layer. This result holds only for restricted classes of activation functions, e.g. for the sigmoidal functions.

Multi-layer networks use a variety of learning techniques, the most popular being backpropagation. Here, the output values are compared with the correct answer to compute the value of some predefined error-function. By various techniques, the error is then fed back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function by some small amount. After repeating this process for a sufficiently large number of training cycles, the network will usually converge to some state where the error of the calculations is small. In this case, one would say that the network has learned a certain target function. To adjust weights properly, one applies a general method for non-linear optimization that is called gradient descent. For this, the derivative of the error function with respect to the network weights is calculated, and the weights are then changed such that the error decreases (thus going downhill on the surface of the error function). For this reason, backpropagation can only be applied on networks with differentiable activation functions.

In general, the problem of teaching a network to perform well, even on samples that were not used as training samples, is a quite subtle issue that requires additional techniques. This is especially important for cases where only very limited numbers of training samples are available. The danger is that the network over fits the training data and fails to capture the true statistical process generating the data. Computational learning theory is concerned with training classifiers on a limited amount of data. In the context of neural networks a simple heuristic, called early stopping, often ensures that the network will generalize well to examples not in the training set.

Other typical problems of the backpropagation algorithm are the speed of convergence and the possibility of ending up in a local minimum of the error function.

Today there are practical solutions that make backpropagation in multi-layer perceptrons the solution of choice for many machine learning tasks.

The numbers within the neurons represent each neuron's explicit threshold (which can be factored out so that all neurons have the same threshold, usually 1). The numbers that annotate arrows represent the weight of the inputs. This net assumes that if the threshold is not reached, zero (not -1) is output. The bottom layer of inputs is not always considered a real neural network layer.

3.4.1.3 ADALINE

ADALINE stands for Adaptive Linear Neuron or later called Adaptive Linear Element. It was developed by Professor Bernard Widrow and his graduate student Ted Hoff at Stanford University in 1960. It's based on the McCulloch-Pitts model. It consists of a weight, a bias and a summation function.

(3.4)

Its adaptation is defined through a cost function (error metric) of the residual,

(3.5)

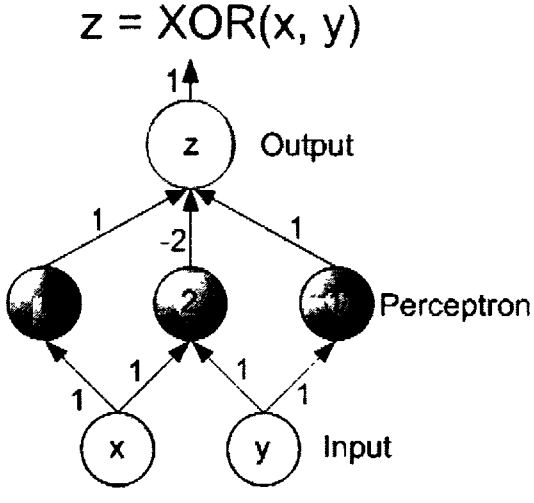


Figure 3.5. A two-layer neural network capable of calculating XOR.

Where, d_i is the desired input. With the MSE error metric,

$$E = \frac{1}{2N}\sum_i^N e_i^2 \tag{3.6}$$

The adapted weight and bias becomes,

$$w = \frac{\sum_i (x_i - \bar{x})(d_i - \bar{d})}{\sum_i (x_i - \bar{x})^2} \tag{3.7}$$

$$b = \frac{\sum_i x_i^2 \sum_i d_i - \sum_i x_i \sum_i x_i d_i}{N \sum_i (x_i - \bar{x})^2} \tag{3.8}$$

The ADALINE has practical applications in the controls area. Like the single-layer perceptron, ADALINE has a counterpart in statistical modelling. in this case least squares regression. There is an extension of the ADALINE. called the Multiple ADALINE (MADALINE) that consists of two or more ADALINEs serially connected.

3.4.2 Radial basis function (RBF) network

Radial Basis Functions are powerful techniques for interpolation in multidimensional space. A RBF is a function which has built into a distance criterion with respect to a centre. Radial basis functions have been applied in the area of neural networks where they may be used as a replacement for the sigmoidal hidden layer transfer characteristic in Multi-Layer Perceptrons.

RBF networks have two layers of processing: In the first, input is mapped onto each RBF in the 'hidden' layer. The RBF chosen is usually a Gaussian. In regression problems the output layer is then a linear combination of hidden layer values representing mean predicted output. The interpretation of this output layer value is the same as a regression model in statistics. In classification problems the output layer is typically a sigmoid function of a linear combination of hidden layer values, representing a posterior probability. Performance in both cases is often improved by shrinkage techniques, known as ridge regression in classical statistics and known to correspond to a prior belief in small parameter values (and therefore smooth output functions) in a Bayesian framework.

RBF networks have the advantage of not suffering from local minima in the same way as Multi-Layer Perceptrons. This is because the only parameters that are adjusted in

the learning process are the linear mapping from hidden layer to output layer. Linearity ensures that the error surface is quadratic and therefore has a single easily found minimum. In regression problems this can be found in one matrix operation. In classification problems the fixed non-linearity introduced by the sigmoid output function is most efficiently dealt with using iteratively re-weighted least squares.

RBF networks have the disadvantage of requiring good coverage of the input space by radial basis functions. RBF centers are determined with reference to the distribution of the input data, but without reference to the prediction task. As a result, representational resources may be wasted on areas of the input space that are irrelevant to the learning task. A common solution is to associate each data point with its own centre, although this can make the linear system to be solved in the final layer rather large, and requires shrinkage techniques to avoid overfitting.

Associating each input datum with an RBF leads naturally to kernel methods such as Support Vector Machines and Gaussian Processes (the RBF is the kernel function). All three approaches use a non-linear kernel function to project the input data into a space where the learning problem can be solved using a linear model. Like Gaussian Processes, and unlike SVMs, RBF networks are typically trained in a Maximum Likelihood framework by maximizing the probability (minimizing the error) of the data under the model. SVMs take a different approach to avoiding overfitting by maximizing instead a margin. RBF networks are outperformed in most classification applications by SVMs. In regression applications they can be competitive when the dimensionality of the input space is relatively small.

3.4.3 Kohonen self-organizing network

The self-organizing map (SOM) invented by Teuvo Kohonen performs a form of unsupervised learning. A set of artificial neurons learn to map points in an input space to coordinates in an output space. The input space can have different dimensions and topology from the output space and the SOM will attempt to preserve these.

3.4.4 Recurrent network

Recurrent neural networks (RNs) are models with bi-directional data flow. Unlike feedforward networks propagating data linearly from input to output, recurrent neural networks (RNs) propagates data from input to output as well as from later processing stages to earlier stages.

A simple recurrent network (SRN) is a variation on the Multi-Layer Perceptron, sometimes called an "Elman network" due to its invention by Jeff Elman. A three-layer network is used, with the addition of a set of "context units" in the input layer. There are connections from the middle (hidden) layer to these context units fixed with a weight of one. At each time step, the input is propagated in a standard feedforward fashion, and then a learning rule (usually backpropagation) is applied. The fixed back connections result in the context units always maintaining a copy of the previous values of the hidden units (since they propagate over the connections before the learning rule is applied). Thus the network can maintain a sort of state, allowing it to perform such tasks as sequence-prediction that is beyond the power of a standard Multi-Layer Perceptron.

The Hopfield network is a recurrent neural network in which all connections are symmetric. Invented by John Hopfield in 1982, this network guarantees that its dynamics will converge. If the connections are trained using Hebbian learning then the Hopfield network can perform as robust content-addressable (or associative) memory, resistant to connection alteration.

The echo state network (ESN) is a recurrent neural network with a sparsely connected random hidden layer. The weights of output neurons are the only part of the network that can change and be learned. ESN are good to (re)produce temporal patterns.

The Long short term memory is an artificial neural net structure that unlike traditional RNNs doesn't have the problem of vanishing gradients. It can therefore use long delays and can handle signals that have a mix of low and high frequency components.

3.4.5 Stochastic neural networks

A stochastic neural network differs from a typical neural network because it introduces random variations into the network. In a probabilistic view of neural networks, such random variations can be viewed as a form of statistical sampling, such as Monte Carlo sampling.

The Boltzmann machine can be thought of as a noisy Hopfield network. Invented by Geoff Hinton and Terry Sejnowski in 1985. the Boltzmann machine is important because it is one of the first neural networks to demonstrate learning of latent variables (hidden units). Boltzmann machine learning was at first slow to simulate, but the contrastive divergence algorithm of Geoff Hinton (circa 2000) allows models such as Boltzmann machines and products of experts to be trained much faster.

3.4.6 Modular neural networks

Biological studies have shown that the human brain functions not as a single massive network, but as a collection of small networks. This realization gave birth to the concept of modular neural networks, in which several small networks cooperate or compete to solve problems.

A committee of machines (CoM) is a collection of different neural networks that together "vote" on a given example. This generally gives a much better result compared to other neural network models. Because neural networks suffer from local minima. starting with the same architecture and training but using different initial random weights often gives vastly different networks. A CoM tends to stabilize the result. The CoM is similar to the general machine learning bagging method, except that the necessary variety of machines in the committee is obtained by training from different random starting weights rather than training on different randomly selected subsets of the training data.

The Associative neural network (ASNN) is an extension of the committee of machines that goes beyond a simple/weighted average of different models. ASNN represents a combination of an ensemble of feedforward neural networks and the *k*-nearest neighbor technique (kNN). It uses the correlation between ensemble responses as

a measure of distance amid the analyzed cases for the kNN. This corrects the bias of the neural network ensemble. An associative neural network has a memory that can coincide with the training set. If new data becomes available, the network instantly improves its predictive ability and provides data approximation (self-learn the data) without a need to retrain the ensemble. Another important feature of ASNN is the possibility to interpret NN results by analysis of correlations between data cases in the space of models.

3.4.7 Other types of networks

Holographic associative memory represents a family of analog, correlation-based, associative, stimulus-response memories, where information is mapped onto the phase orientation of complex numbers operating.

Instantaneously trained neural networks (ITNNs) were inspired by the phenomenon of short-term learning that seems to occur instantaneously. In these networks the weights of the hidden and the output layers are mapped directly from the training vector data. Ordinarily, they work on binary data, but versions for continuous data that require small additional processing are also available.

Spiking neural networks (SNNs) are models which explicitly take into account the timing of inputs. The network input and output are usually represented as series of spikes (delta function or more complex shapes). SNNs have an advantage of being able to process information in the time domain (signals that vary over time). They are often implemented as recurrent networks. SNNs are also a form of pulse computer.

Networks of spiking neurons — and the temporal correlations of neural assemblies in such networks — have been used to model figure/ground separation and region linking in the visual system.

Spiking neural networks with axonal conduction delays exhibit polychronization, and hence could have a potentially unlimited memory capacity. Dynamic neural networks not only deal with nonlinear multivariate behavior, but also include (learning of) time-dependent behavior such as various transient phenomena and delay effects.

Cascade-Correlation is architecture and supervised learning algorithm developed by Scott Fahlman and Christian Lebiere. Instead of just adjusting the weights in a network of fixed topology, Cascade-Correlation begins with a minimal network, then automatically trains and adds new hidden units one by one, creating a multi-layer structure. Once a new hidden unit has been added to the network, its input-side weights are frozen. This unit then becomes a permanent feature-detector in the network, available for producing outputs or for creating other, more complex feature detectors. The Cascade-Correlation architecture has several advantages over existing algorithms: it learns very quickly, the network determines its own size and topology, it retains the structures it has built even if the training set changes, and it requires no backpropagation of error signals through the connections of the network.

A neuro-fuzzy network is a fuzzy inference system in the body of an artificial neural network. Depending on the FIS type, there are several layers that simulate the processes involved in a fuzzy inference like fuzzification, inference, aggregation and defuzzification. Embedding an FIS in a general structure of an ANN has the benefit of using available ANN training methods to find the parameters of a fuzzy system.

Compositional pattern-producing networks (CPPNs) are a variation of ANNs which differ in their set of activation functions and how they are applied. While typical ANNs often contain only sigmoid functions and sometimes Gaussian functions, CPPNs can include both types of functions and many others. Furthermore, unlike typical ANNs, CPPNs are applied across the entire space of possible inputs so that they can represent a complete image. Since they are compositions of functions, CPPNs in effect encode images at infinite resolution and can be sampled for a particular display at whatever resolution is optimal.

This type of network can add new patterns without the need for re-training. It is done by creating a specific memory structure, which assigns each new pattern to an orthogonal plane using adjacently connected hierarchical arrays. The network offers real-time pattern recognition and high scalability; it however requires parallel processing and is thus best suited for platforms such as Wireless sensor networks (WSN), Grid computing, and GPGPUs.

3.5 Theoretical Properties

3.5.1 Computational power

The multi-layer perceptron (MLP) is a universal function approximator. as proven by the Cybenko theorem. However, the proof is not constructive regarding the number of neurons required or the settings of the weights.

Work by Hava Siegelmann and Eduardo D. Sontag has provided a proof that a specific recurrent architecture with rational valued weights (as opposed to the commonly used floating point approximations) has the full power of a Universal Turing Machine using a finite number of neurons and standard linear connections. They have further shown that the use of irrational values for weights results in a machine with trans-Turing power.

3.5.2 Capacity

Artificial neural network models have a property called 'capacity', which roughly corresponds to their ability to model any given function. It is related to the amount of information that can be stored in the network and to the notion of complexity.

3.5.3 Convergence

Nothing can be said in general about convergence since it depends on a number of factors. Firstly, there may exist many local minima. This depends on the cost function and the model. Secondly, the optimization method used might not be guaranteed to converge when far away from a local minimum. Thirdly, for a very large amount of data or parameters, some methods become impractical. In general, it has been found that theoretical guarantees regarding convergence are an unreliable guide to practical application.

3.5.4 Generalization and statistics

In applications where the goal is to create a system that generalizes well in unseen examples, the problem of overtraining has emerged. This arises in over complex or over specified systems when the capacity of the network significantly exceeds the needed free parameters. There are two schools of thought for avoiding this problem: The first is to use cross-validation and similar techniques to check for the presence of overtraining and optimally select hyper parameters such as to minimize the generalization error. The second is to use some form of regularization. This is a concept that emerges naturally in a probabilistic (Bayesian) framework, where the regularization can be performed by selecting a larger prior probability over simpler models; but also in statistical learning theory, where the goal is to minimize over two quantities: the 'empirical risk' and the 'structural risk', which roughly correspond to the error over the training set and the predicted error in unseen data due to overfitting.

Supervised neural networks that use an MSE cost function can use formal statistical methods to determine the confidence of the trained model. The MSE on a validation set can be used as an estimate for variance. This value can then be used to calculate the confidence interval of the output of the network, assuming a normal distribution. A confidence analysis made this way is statistically valid as long as the output probability distribution stays the same and the network is not modified.

By assigning a softmax activation function on the output layer of the neural network (or a softmax component in a component-based neural network) for categorical target variables, the outputs can be interpreted as posterior probabilities. This is very useful in classification as it gives a certainty measure on classifications.

The softmax activation function:

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{c} e^{x_j}} \tag{3.9}$$

3.5.5 Dynamic properties

Various techniques originally developed for studying disordered magnetic systems (i.e. the spin glass) have been successfully applied to simple neural network architectures, such as the Hopfield network. Influential work by E. Gardner and B. Derrida has revealed many interesting properties about perceptrons with real-valued synaptic weights, while later work by W. Krauth and M. Mezard has extended these principles to binary-valued synapses.

3.6 Corroboration

Variety and use of neural network for optimization of different types of problem set undoubtedly is fairly diverse; depending on the problem requirements, specifications, formulation and the trade-offs that has to be met the network as well as the algorithm that are used to approach any problem set can be delicate process. In this research work the problems asked to cover three very different but interrelated application targets.

- Approximation
- Clustering
- Pattern Recognition

3.6.1 Approximation

Neural network has its intrinsic ability to realize and map nonlinearity given that it is trained with a set of input-output datasets which represents the system [17]. In this case, considering different aspects, such as degree of non-linearity, complexity, accuracy, size of dataset etc., a backpropagation neural network is used for system approximation where Lavenberg-Marquardt algorithm is used as an optimization algorithm [18].

3.6.2 Clustering

Given a distributed data set where some kind of intrinsic formation exists, neural network is capable of grouping data with similar attributes; which is very effective in

filtering or perhaps distinguishing datasets blend or intermingle together. Thus neural network clustering is used to cluster similar type signals where more than one signal is mixed together [19], [20]. A self organizing map is used for the purpose where batch unsupervised weight with bias training is used.

3.6.3 Pattern recognition

Pattern recognition is a one of the very intriguing capabilities of neural network. This is different from pattern identification as, if the neural network is trained with a number of patterns with adequate number of dataset [21], [22]; it becomes capable of indentifying similar patterns even with significant noise associated with it. A backpropagation neural network is used for this where scaled conjugate gradient algorithm is used.

3.7 References

[1] Y. Bar-Yam, *Dynamics of Complex Systems*, Westview Press, 2003.

[2] Y. Bar-Yam, *Making Things Work*, Westview Press, 2005.

[3] G.V. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, Vol. 2, pp. 303-314, 1989.

[4] M. Egmont-Petersen, D. de Ridder, H. Handels, "Image processing with neural networks - a review," *Pattern Recognition,* vol. 35 (10), pp 2279–2301, 2002.

[5] K. Gurney, *An Introduction to Neural Networks*, London: Routledge, 1997.

[6] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 1999.

[7] J. Hertz, R.G. Palmer, A.S Krogh, *Introduction to the Theory of Neural Computation*, Perseus Books, 1990.

[8] J. Lawrence, *Introduction to Neural Networks*, California Scientific Software, 1994.

[9] T. Masters, *Signal and Image Processing with Neural Networks*, John Wiley & Sons, 1994.

[10] H.T. Siegelmann, and E.D. Sontag, "Analog computation via neural networks," *Theoretical Computer Science*, v. 131, no. 2, pp. 331-360, 1994.

[11] S.A. Danziger, S.J. Swamidass, J. Zeng, L.R. Dearth, Q. Lu, J.H. Chen, J. Cheng, V.P. Hoang, H. Saigo, R. Luo, P. Baldi, R.K. Brachmann, and R.H. Lathrop, "Functional census of mutation sequence spaces: the example of p53 cancer rescue mutants", *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 3, pp. 114-125, 2006.

[12] S.A. Danziger, J. Zeng, Y. Wang, R.K. Brachmann, and R.H. Lathrop, "Choosing where to look next in a mutation sequence space: Active learning of informative p53 cancer rescue mutants," *Bioinformatics*. 23(13), pp. 104-114, 2007.

[13] S. Kotsiantis, "Supervised machine learning: A review of classification techniques," *Informatica Journal,* vol. 31, pp. 249-268, 2007.

[14] G. Hinton and J.T. Sejnowski, *Unsupervised Learning: Foundations of Neural Computation*, MIT Press, 1999.

[15] S. Kotsiantis and P. Pintelas, "Recent advances in clustering: A brief survey," *WSEAS Transactions on Information Science and Applications*. vol. 1, no. 1, pp. 73-81, 2004.

[16] R.O. Duda, P.E. Hart, and D.G. Stork, *Unsupervised Learning and Clustering, Ch. 10 in Pattern classification (2nd edition)*, pp. 571, Wiley. New York, 2001.

[17] R.D. Jones, Y.C. Lee, C.W. Barnes, G.W. Flake, K. Lee, P.S. Lewis, and S. Qian, "Function approximation and time series prediction with neural networks," *Proceedings of the International Joint Conference on Neural Networks*, June 17-21, p. I-649, 1990.

[18] J. R. Davies, S. V. Coggeshall, R. D. Jones, and D. Schutzer. *Intelligent Security Systems, in Freedman, Artificial Intelligence in the Capital Markets*, Chicago, 1995.

[19] L.B. Bourque, C.A. Virginia, *Processing Data: The Survey Example (Quantitative Applications in the Social Sciences)*, Sage Publications Inc., December 14, 2006.

[20] C.M. Walt, *Data Measures that Characterise Classification Problems*, Master's dissertation, Department of Electrical, Electronic and Computer Engineering, University of Pretoria, South Africa, February 2008.

[21] B.B. Nasution and A.I. Khan, "A hierarchical graph neuron scheme for real-time pattern recognition", *IEEE Transactions on Neural Networks*, vol 19(2), 212-229, Feb. 2008.

[22] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification (2nd edition)*, Wiley, 2001.

4 SYNCHRONOUS MACHINE SIMULATION

4.1 Introduction

To investigate the transient performance of synchronous machine the synchronous machine model described in chapter 2 is used for simulation. The main objective of the simulation primarily is to realize how the machine behaves under different conditions. Two models the saturated and the unsaturated models are developed. To perform the numerical simulation of the machine model Matlab is used.

In a synchronous machine field excitation voltage and input torque are the usual inputs. Under steady state condition all of these input values are constant. In order to simulate a transient condition usually some kind of fault is introduced. The faults can be of various sources, but in most common cases they are either loss of excitation in the field, disturbance in torque or at times short circuit occurrence in the terminal circuit. In this research work, the machine model was simulated under all of these conditions. A brief description of these three commonly occurred faults or commotions in a synchronous machine is described in the following section.

Response of synchronous machine to any of the disturbances is realized by looking at the output of the machine. As output of the machine it is the load angle δ, which is usually looking into.

In a synchronous machine load angle δ is the usual output. Once a disturbance is introduced in any certain point in time, it is cleared after a certain amount of time. Initially a machine is running in steady state; once the machine is exposed to a fault and it is cleared δ gets distorted at first. If the machine doesn't become unstable, δ will settle down to its original steady state with the passage of time. If the machine goes to instability, δ goes out of control. The behavior of δ from the time of distortion to its stalling down time and its behavior during this time gives us the transient behavior of the machine under the fault or commotion.

A tenth order model is used to run the simulation. The saturation is taken into account to in the second model. The behavior of the model is compared with that of the unsaturated model [1]. Runge-Kutta method was used to solve the differential equations.

The fault occurred at 0.1 s and was cleared after different time periods to analyze the stability. The level of fault was varied as well. Machine parameters, operating conditions, simulation process flow, etc. follow in the next coming sections.

4.2 Synchronous Machine Commotions

4.2.1 Loss of excitation/field (LOF)

Loss of excitation or field in a synchronous machine is one of most common occurrences of faults in synchronous machine. The loss of excitation is defined by partial or full short circuit in the field circuit of a synchronous machine [1]-[3]. Another way of looking at it is the field or excitation voltage of a synchronous machine rotor winding is subjected to a partial loss or goes to complete zero in case of total loss in field [3]-[6].

In this research work, the loss of excitation condition is scrutinized under two conditions. In the first case, the duration of the fault is kept constant. That is the fault is kept persistent for 0.1 s. It is initiated at 0.1 s and cleared at 0.2 s. For 0.1 s the fault is introduced for 25%, 50%, 75% & 100% loss in field. In this case, the fault level is kept constant to 100% loss of field. The duration of the fault is varied, that is, the fault continues for 0.1 s, 0.2 s, 0.5 s or 1.0 s.

4.2.2 Disturbance in Torque (DIT)

Disturbance in torque is defined by disturbance in mechanical torque [7] input to the machine. The source of input torque can be hydro, steam, coal, gas, etc. The torque provided by the prime mover is desired to be constant to provide the machine with a constant torque. But at times there can be disruption in the input torque as a result the prime mover can speed up or slow down as a result synchronous machine can go into sub-synchronous or super-synchronous speed [8].

In this research work, the disturbance in torque condition is scrutinized under two conditions. In the first case, the duration of the fault is kept constant. That is the fault is kept persistent for 0.1 s. It is initiated at 0.1 s and cleared at 0.2 s. For 0.1 s the fault is

introduced for 50% loss, 100% loss, 50% gain and 100% gain in input torque. In the second case, the fault level is kept constant to 100% loss in input torque. In this case, the duration of the fault is varied, that is, the fault is kept persistent for 0.1 s, 0.2 s, 0.5 s or 1.0 s.

4.2.3 Short circuit (SC)

The terminal of a synchronous machine is subjected to constant voltage as it is connected to the grid. At times under certain conditions, the terminals of the machine might get shorted which is known as the short circuit fault [9], [10].

In this research work, the short circuit condition is scrutinized. In this case, the duration of the fault is varied, that is, the fault is kept persistent for 0.075 s, 0.15 s, 0.212 s and 0.213 s. It will be demonstrated later that at 0.212 s the machine becomes marginally stable. The machine becomes unstable if the fault persists more than 0.212 s.

4.3 System Deliberates

4.3.1 Machine parameters

The machine simulated here is a 3-phase Y-connected 900 MVA synchronous generator. The machine parameters are presented in Table 4.1. All the parameters are in per unit. The model is simulated in per unit system [11]. For unsaturated model L_d and L_q are kept constant as in Table 4.1. For the saturated model they are calculated using the saturation characteristics.

Table 4.1. Machine parameters.

Parameters	Values (per unit)	Parameters	Values (per unit)
R_a	0.0018	L_l	0.172
R_{kd1}	0.1142	R_{kq1}	0.00538
R_{kd2}	0.00592	R_{kq2}	0.1081
R_{fd}	0.00094	R_{kq3}	0.0188
L_d	2.152	L_q	2.057
L_{kd1}	2.732	L_{kq1}	1.657
L_{kd2}	0.00753	L_{kq2}	0.1193
L_{fd}	0.0155	L_{kq3}	0.4513

Table 4.2. Operating conditions.

Parameter	Value
Terminal voltage	1.0 pu
Terminal apparent power	1.0 pu
Power Factor	0.9
Speed control Gain K_p	20
Speed control Integral Time T_i	2 s

4.3.2 Operating conditions

As the simulation is done in per unit system, all values including the loading conditions are taken as per unit system. Table 4.2 illustrates the operating characteristics used in the machine model simulation.

4.3.3 Process flow

The process flow diagram is basically the flow chart describing the simulation procedure. It gives us the basic understanding of the simulation steps under different conditions. In this section, the basic procedures used to perform the initial value

calculation and transient simulations by the developed models are explained. For each of the conditions both the saturated and the unsaturated models are shown.

4.3.3.1 Steady-state condition

Figure 4.1 shows flowchart to calculate the initial values, where, terminal voltage, apparent power and power factor are given as input. Then, the load angle (δ) of the machine can be calculated. The d- and q-axis components of the stator voltage and current, field voltage and current are determined.

An additional loop has been considered for taking saturation into account. By calculating the d- and q-axis magnetizing currents and then using the saturation characteristics in Figure 4.2, the saturated d- and q-axis magnetizing reactances (X_{mds} and X_{mqs}) can be obtained [12], [13]. These new values of the magnetizing reactance result new values of stator and rotor currents and load angle. And if all this current values are less than $\varepsilon = 10^{-6}$, the saturation condition is met for initial conditions. Then fluxes and developed electromagnetic torque can be calculated.

4.3.3.2 Transient condition

To obtain transient performance of the synchronous generator under LOF fault, we have to solve the system differential equations. In general, there are two methods for the integration of differential equations in power system simulation: one is an explicit method, such as the 4th-order Runge-Kutta method, and the other is an implicit one, such as the trapezoidal rule. In this research work explicit method has been used. 4th-order Runge-Kutta method was used. Figure 4.3 shows the flowchart to calculate transient condition for unsaturated model and Figure 4.4 shows the flowchart for saturated model.

The flowchart also illustrates an iteration process within each time step to determine saturated magnetizing reactance in both direct and quadrature axes. Basically within each time step after numerically solving differential equations and obtaining currents, saturated X_d and X_q are needed to be determined [14], [15]. This involves an iteration loop and after the currents converge then the process can proceed to the next time step.

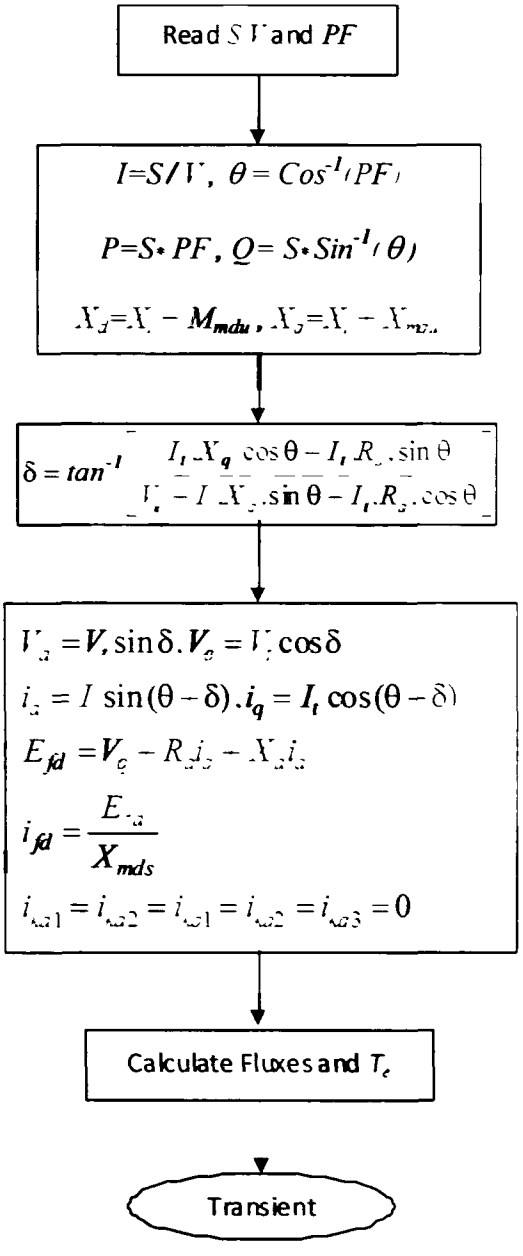


Figure 4.1. Initial value calculation flowchart for unsaturated model.

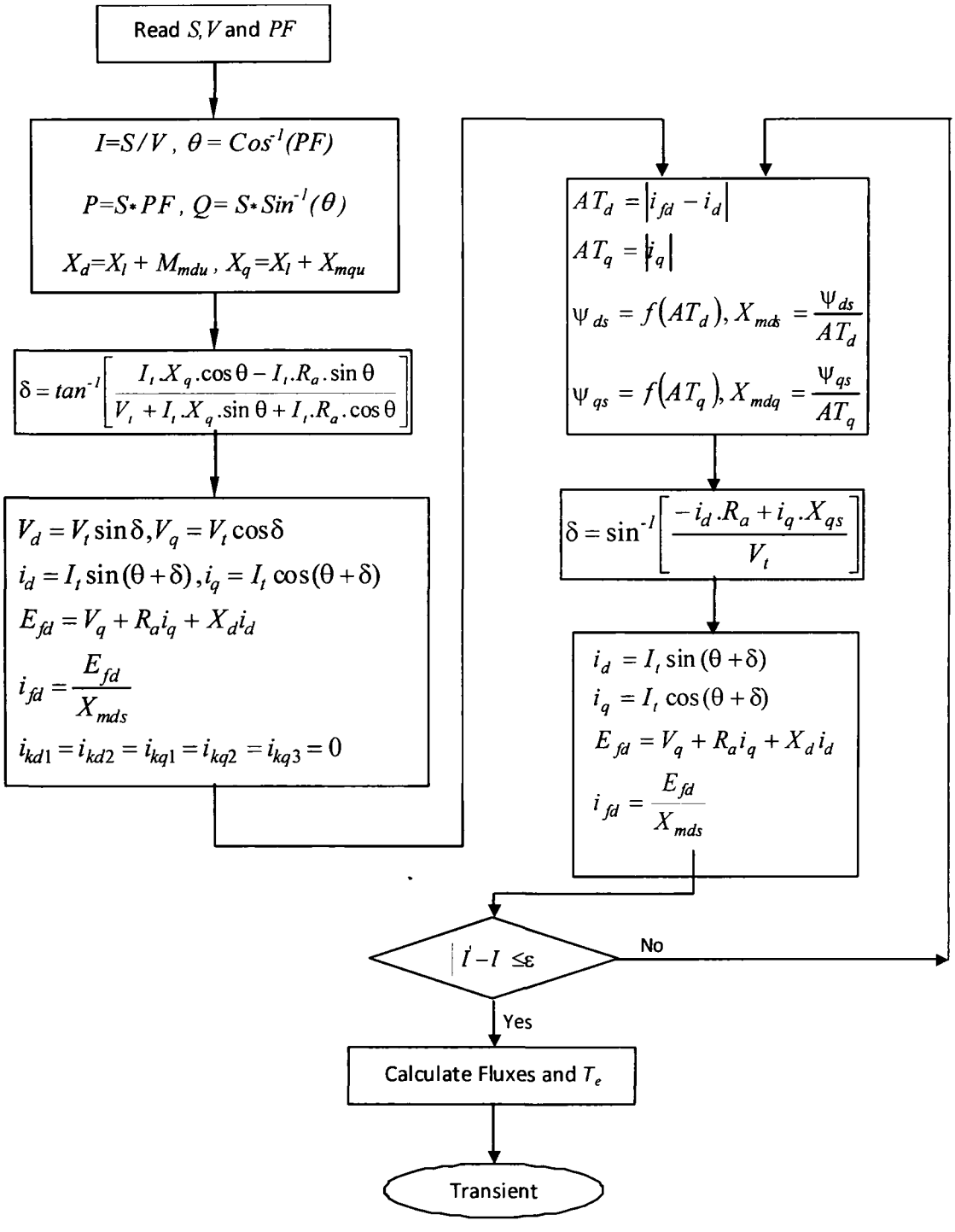


Figure 4.2. Initial value calculation flowchart for saturated model.

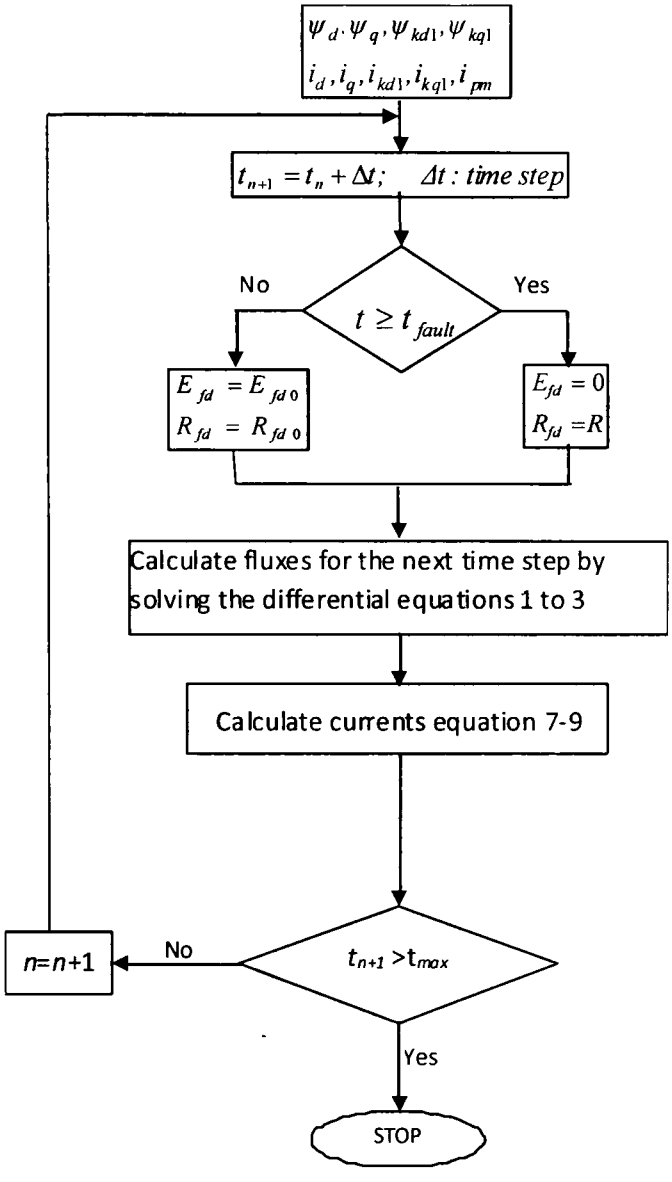


Figure 4.3. Calculation of transient values after LOF fault for unsaturated model.

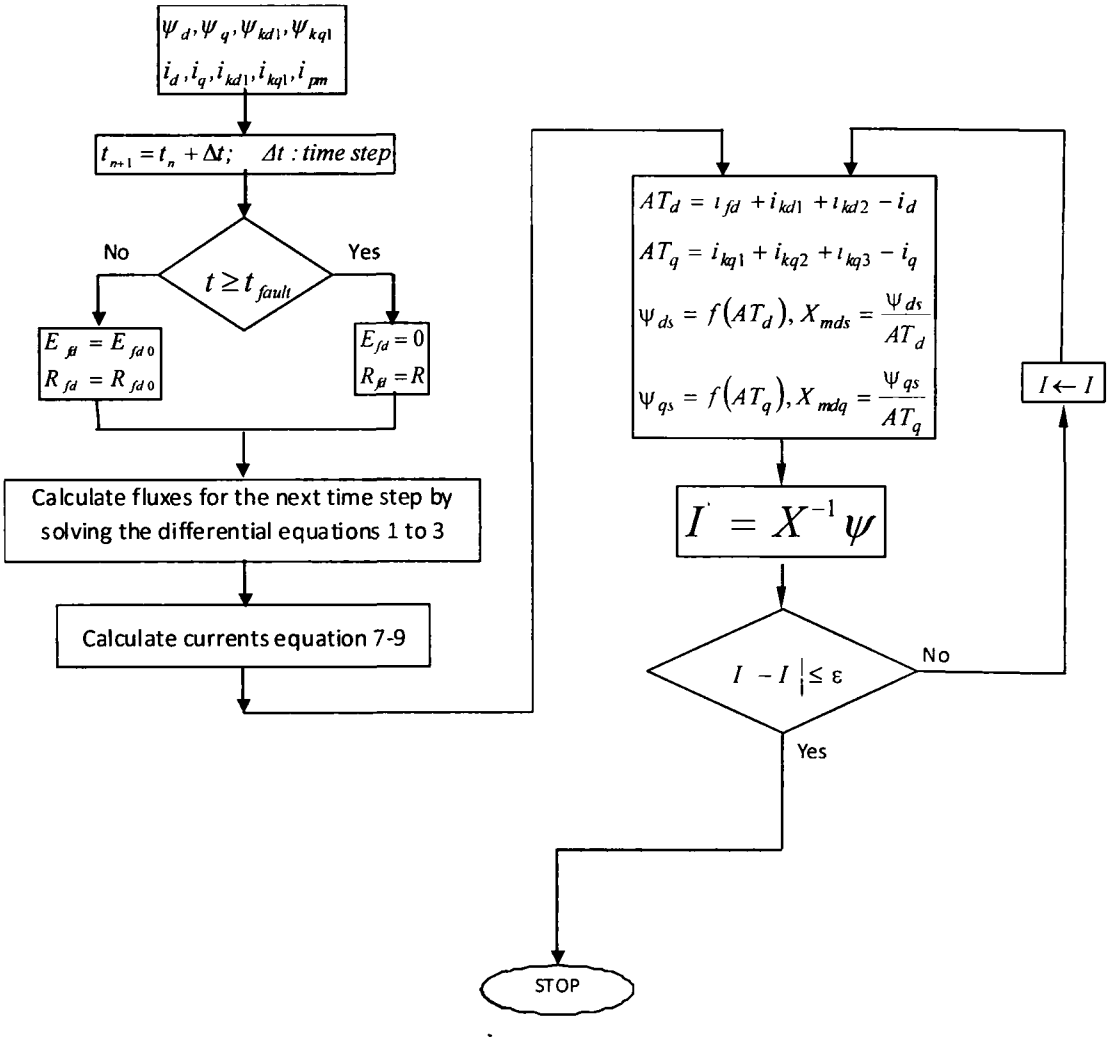


Figure 4.4. Calculation of transient values after LOF fault for saturated model.

4.4 Simulation and Results

4.4.1 Loss of excitation/field (LOF)

In this section, the machine model is simulated under loss of excitation or field. The operating condition under which the machine is run is specified in Table 4.2. The machine is simulated for both the saturated and unsaturated models and the results are shown in the same graph for comparison. Observation tells us that taking saturation into account makes a big difference in the results, hence, a more accurate one.

In Figure 4.5, the fault is persistent for 0.1 s. It is introduced at 0.1 s and cleared at 0.2 s. The level of fault is 25% LOF, which means that the excitation has been reduced to 25% of the rated value. It is seen that the maximum overshoot is 44.35 degrees for unsaturated model and 44.7 degrees for saturated model. The settling time in both cases is roughly 9 to 10 s. Figure 4.6 shows the fault persistence for 0.1 s. It is introduced at 0.1 s and cleared at 0.2 s. The level of fault this time is kept at 50% LOF, which means that the excitation has been reduced to 50% of the rated value. It is seen that the maximum overshoot is 45.35 degrees for unsaturated model and 45.9 degrees for saturated model. The settling time in both cases is about 8 s to 9 s.

Again, in Figure 4.7, the fault is invariable for 0.1 s. It is introduced at 0.1 s and cleared at 0.2 s. In this case, the excitation has been reduced to 75% of the rated value. It is seen that the maximum swing is 46.2 degrees for unsaturated model and 47.2 degrees for saturated model. The settling time in both models is roughly 7 to 8 s. In Figure 4.8, the fault is also persistent for 0.1 s. It is introduced at 0.1 s and cleared at 0.2 s. The excitation in this case has been reduced to 0% of the rated value. It is seen that the maximum swing is 47 degrees for unsaturated model and 48.5 degrees for saturated model. The settling time in both models is around 6 to 7 s.

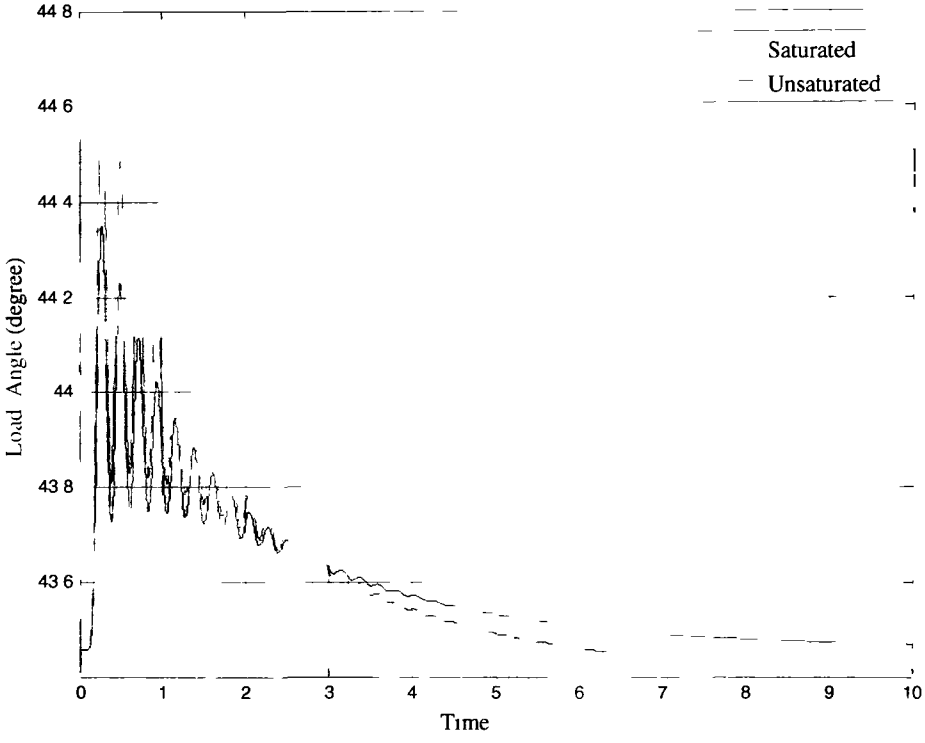


Figure 4.5. 25% LOF for 0.1 s.

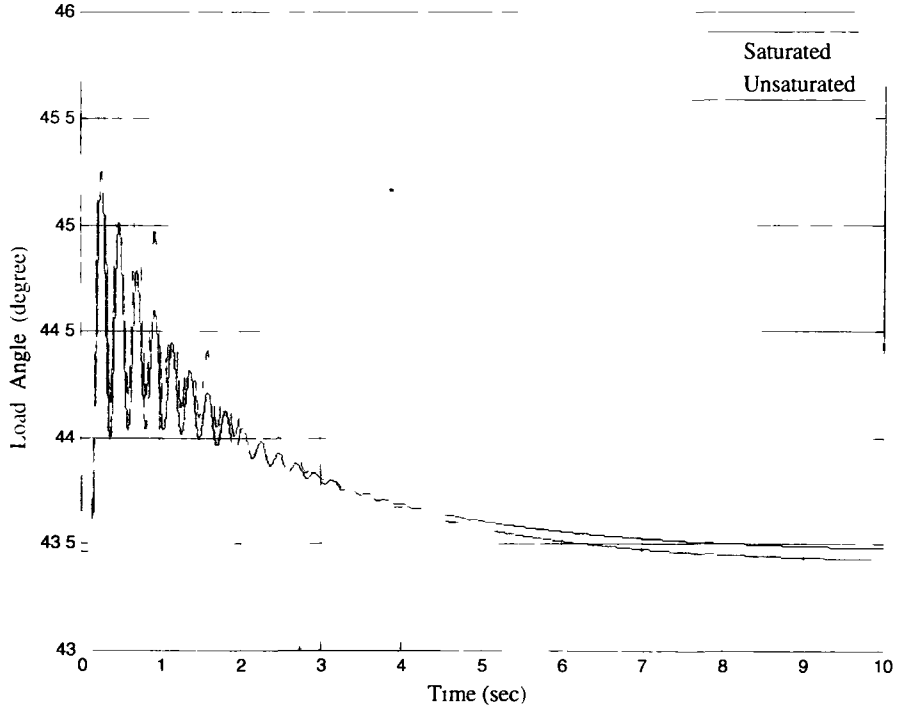


Figure 4.6. 50% LOF for 0.1 s.

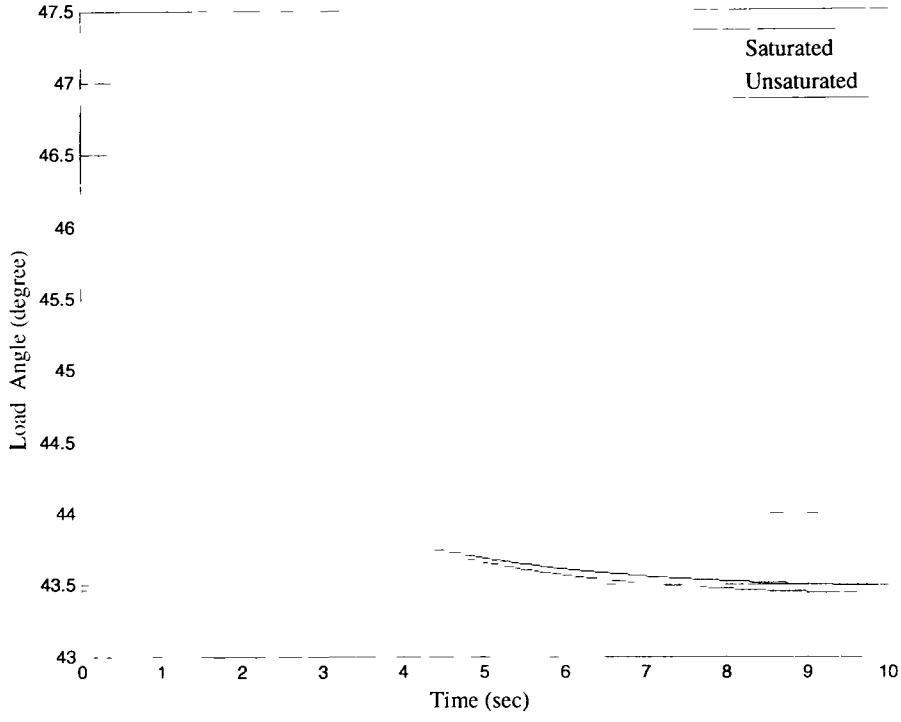


Figure 4.7. 75% LOF for 0.1 s.

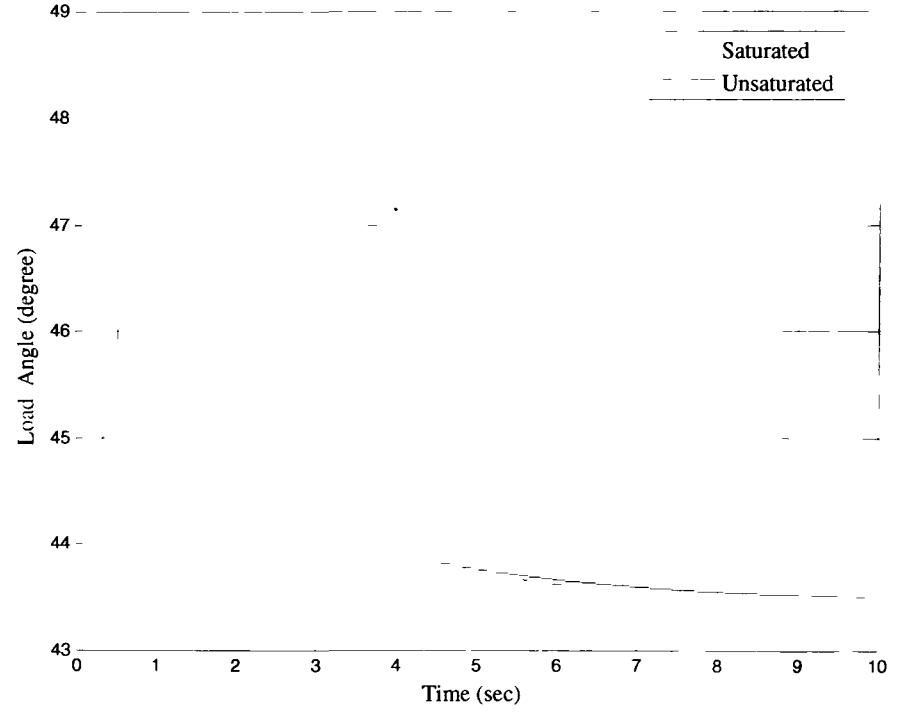


Figure 4.8. 100% LOF for 0.1 s.

In Figure 4.9, the fault is taking place for 0.2 s while the fault is kept at the same level where the excitation has been reduced to 0% of the rated value. The fault is introduced at 0.1 s and cleared at 0.3 s. It is seen that the maximum overshoot is 48.5 degrees for unsaturated model and 50.2 degrees for saturated model. The settling time in both cases is around 8 to 9 s. In Figure 4.10, the fault persists for 0.5 s again for full loss of excitation. It is introduced at 0.1 s and cleared at 0.6 s. It is seen that the maximum overshoot is 55 degrees for unsaturated model and 59 degrees for saturated model. The settling time in both cases is approximately 9 to 10 s.

In Figure 4.11, the fault is introduced at 0.1 s and cleared at 1.1 s for 100% LOF. It is seen that the maximum overshoot is 47 degrees for unsaturated model and 48.5 degrees for saturated model. The settling time in both cases is almost same as the previous case that is around 9 to 10 s.

From these observations, one can see that as the level of faults goes up or as the time duration of the fault persistence goes up, the change in the load angle increases. Another observation is that as level of fault went up the settling time decreased but as the duration of fault went up the settling time increased.

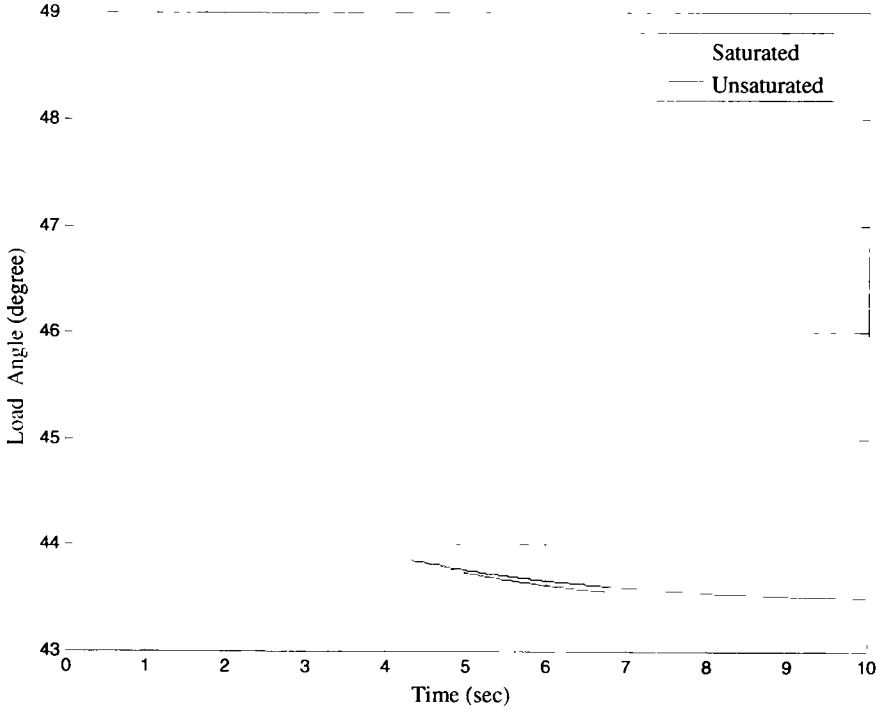


Figure 4.9. 100% LOF for 0.2 s.

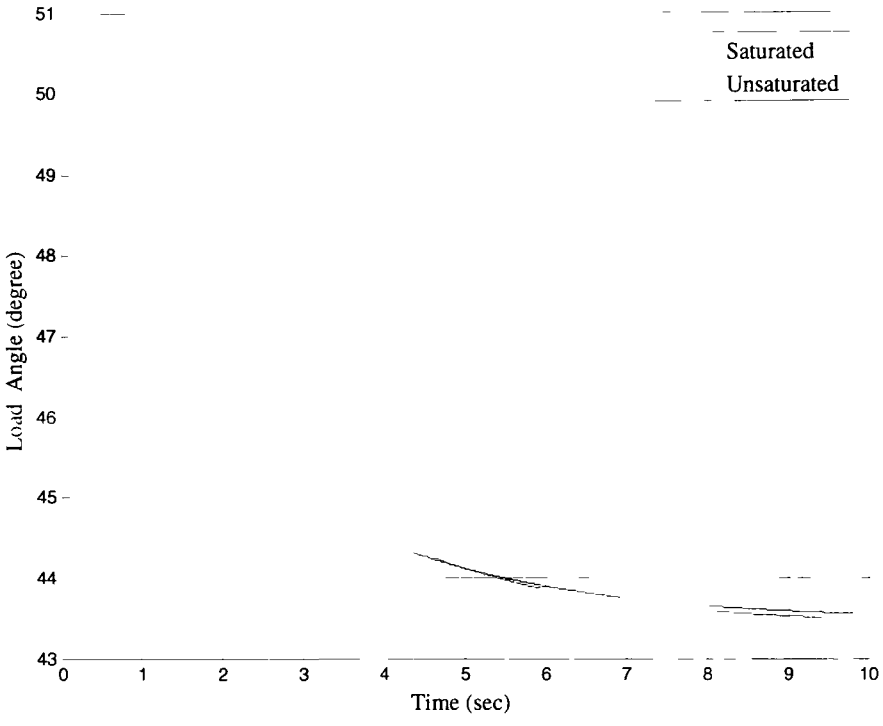


Figure 4.10. 100% LOF for 0.5 s.

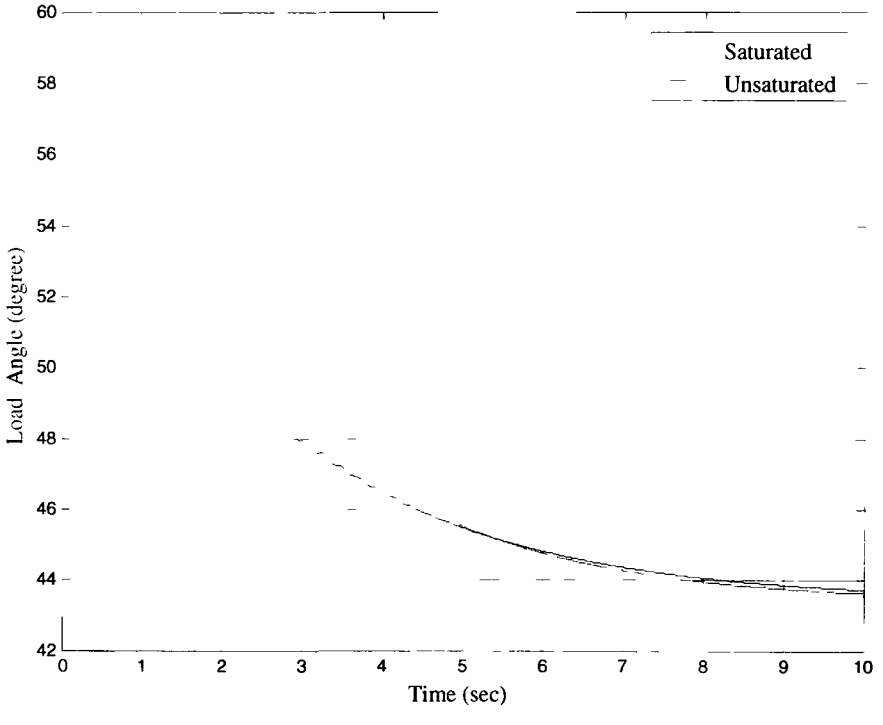


Figure 4.11. 100% LOF for 1 s.

4.4.2 Disturbance in torque (DIT)

In this section, the machine model is simulated under disturbance in torque. The operating conditions under which the machine is run are presented in 4.2. The machine is simulated for both the saturated and unsaturated models and the results are shown in the same graph for assessment purposes. Observation tells us that taking saturation into account makes a difference in the results even though not very significant one.

In Figure 4.12, the fault is continual for 0.1 s. It is introduced at 0.1 s and cleared at 0.2 s. The level of fault is 50% loss in DIT. which means that the torque has lost 50% of its rated value. It is seen that the maximum disruption is 14 degrees for both unsaturated and saturated models. The settling time in both cases is around 6 to 7 s. In Figure 4.13, the fault is persistent for 0.1 s. It is introduced at 0.1 s and cleared at 0.2 s. The level of fault is 100% loss in DIT. which means that the torque 0% of its rated value. It is seen that the maximum disruption is 30 degrees for both unsaturated and saturated models. The settling time in both cases is around 6 to 7 s.

In Figure 4.14, the fault occurrs for 0.1 s. It is introduced at 0.1 s and cleared at 0.2 s. The torque input is increased to 150% of its rated value. It is seen that the maximum disruption is 12 degrees for both unsaturated and saturated models. The settling time in both cases is around 6 to 7 s. In Figure 4.15, the torque input is increased to 200% of its rated value. It is seen that the maximum disruption is 21 degrees for both unsaturated and saturated models. The settling time in both cases is around 7 to 8 s.

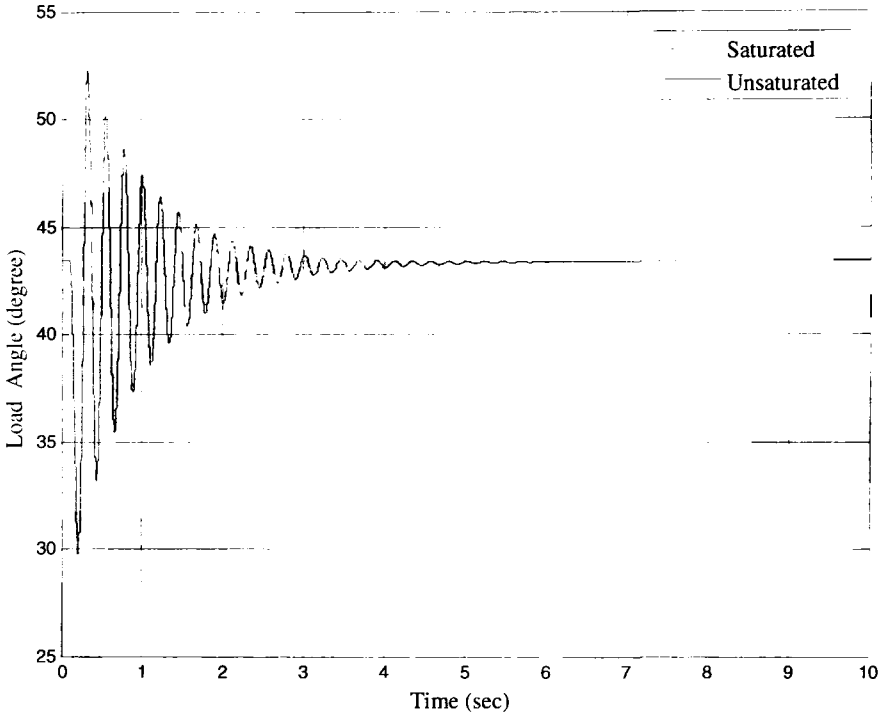


Figure 4.12. 50% loss of DIT for 0.1 s.

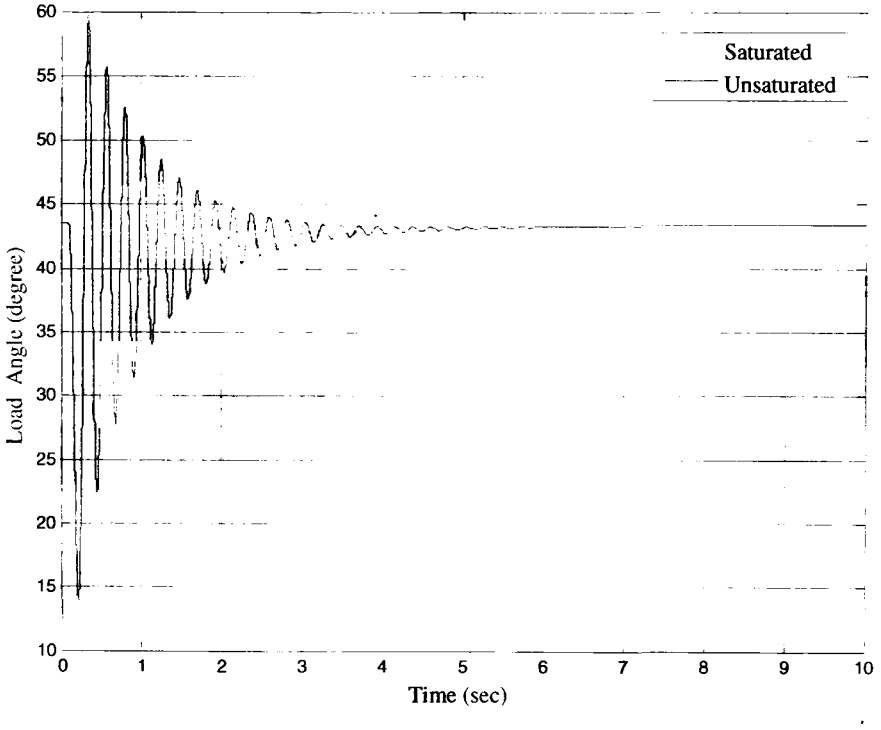


Figure 4.13. 100% loss of DIT for 0.1 s.

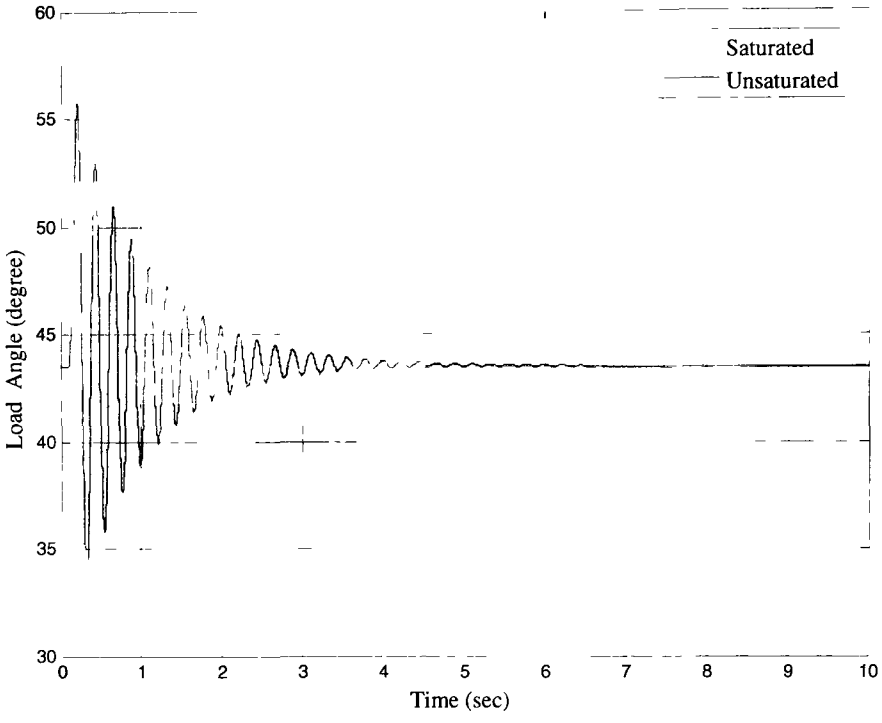


Figure 4.14. 50% over-excitation of DIT for 0.1 s.

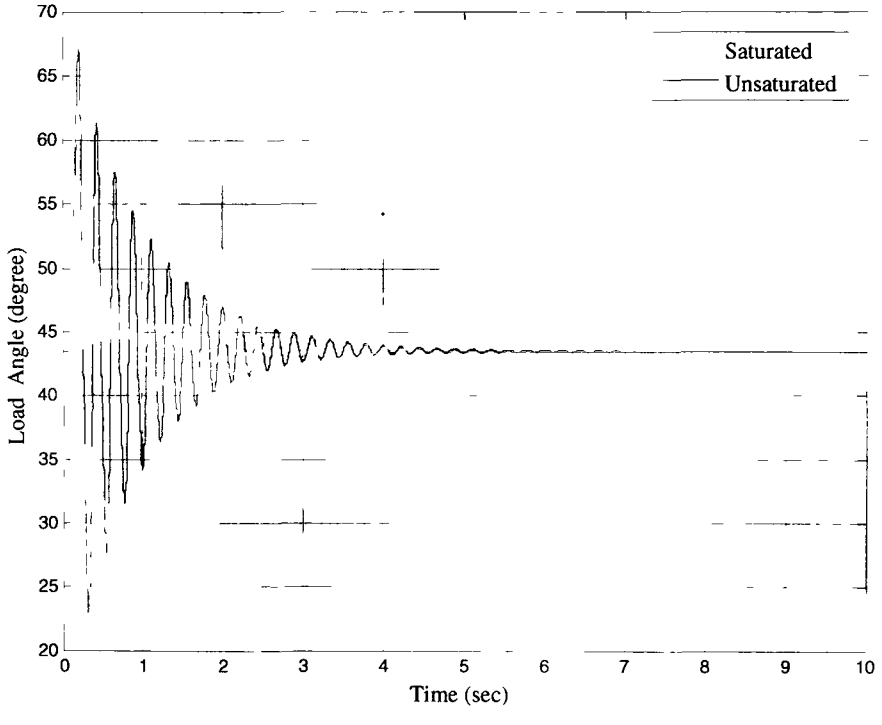


Figure 4.15. 100% over-excitation of DIT for 0.1 s.

Here, Figure 4.16 shows fault continues for 0.2 s. It is introduced at 0.1 s and cleared at 0.3 s. The level of fault is 100% loss in torque input. It is seen that the maximum disruption is 34 degrees for both unsaturated and saturated models. The settling time in both cases is around 8 to 9 s. In Figure 4.17 the fault is introduced at 0.1 s and cleared at 0.6 s. It is seen that the maximum disruption is 34 degrees for both unsaturated and saturated model with extra overshooting element. The settling time in both cases is approximately in the range of 9 to 10 s. In Figure 4.18, the fault is persistent for 1 s. It is seen that the maximum disruption is 37 degrees for both unsaturated and saturated models with extra overshooting element. The settling time in both cases is about 9 to 10 s.

From observation one can see that as the level of faults goes up or as the time duration of the fault persistence goes up the degree to which δ's maximum distortion occurrence increases. In the case of loss in torque, the overshoot is in one direction and, in the case of torque increase over its rated value, the overshoot is in the opposite direction. Another observation is as level of fault went up the settling time remained fairly constant but as the duration of fault went up the settling time increased.

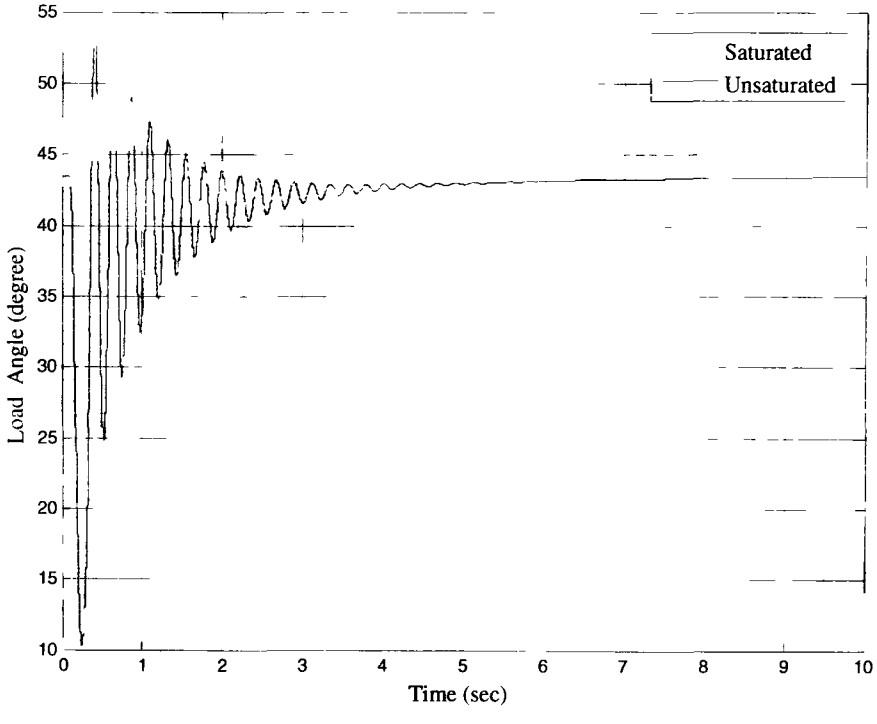


Figure 4.16. 100% loss of DIT for 0.2 s.

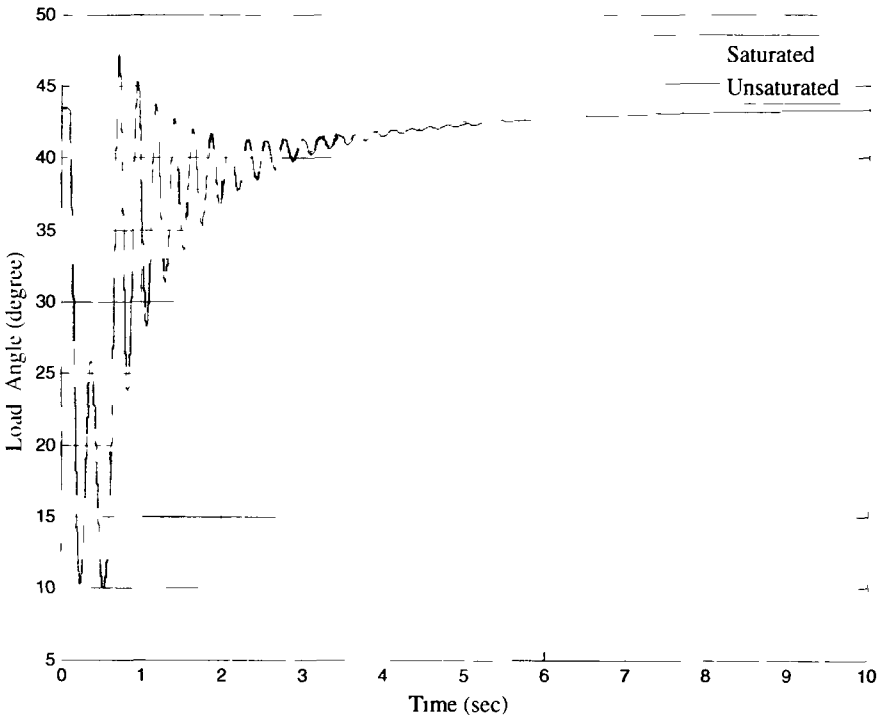


Figure 4.17. 100% loss of DIT for 0.5 s.

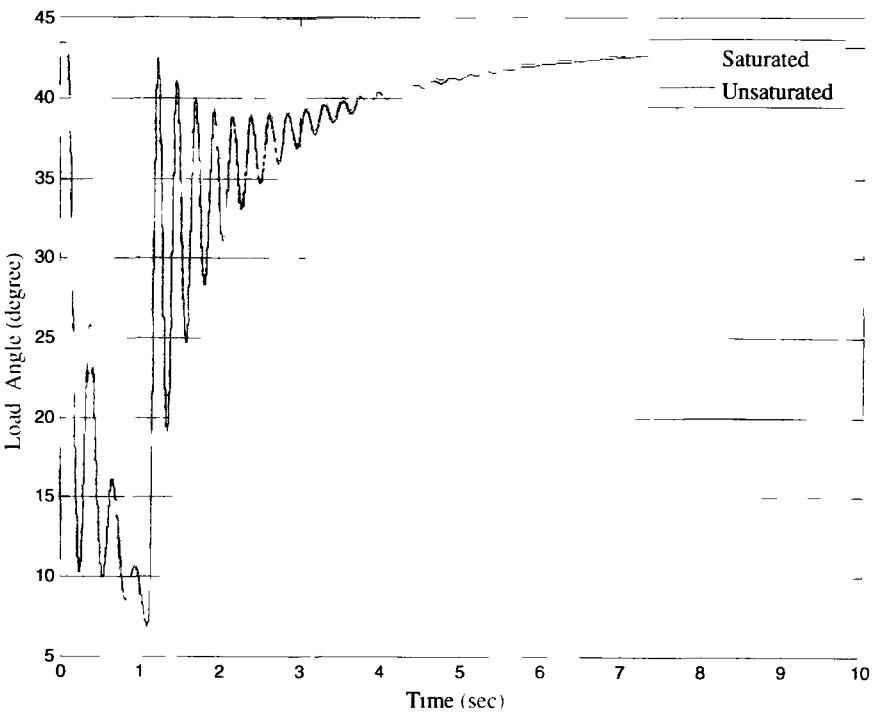


Figure 4.18. 100% loss of DIT for 1 s.

4.4.3 Short circuit (SC)

In this section, the machine model is simulated under short circuit condition at the machine terminals. The operating condition under which the machine is run is specified in Table 4.2. The machine is simulated for both the saturated and unsaturated models and the results are shown in the same graph for comparison. In Figure 4.23, all the graphs for different time periods are put together for stability analysis. Observation tells us that taking saturation into account makes a difference in the results even though not very significant one.

Figure 4.19 is for SC persistence for 0.075 s. It is introduced at 0.1 s and cleared at 0.175 s. It is seen that the maximum disruption is 39 degrees for both unsaturated and saturated model. The settling time in both cases is around 5 to 6 s. In Figure 4.20, SC is introduced at 0.1 s and cleared at 0.25 s. It is seen that the maximum disruption is 58 degrees for both unsaturated and saturated models. The settling time in both cases is around 6 to 7 s.

Figure 4.21 shows the results for the case where the SC is introduced at 0.1 s and cleared at 0.312 s. It is seen that the maximum disruption is 126 degrees for both unsaturated and saturated models. The settling time in both cases is around 6 to 7 s. At 0.212 s the machine becomes marginally stable. In Figure 4.22, the SC is introduced at 0.1 s and cleared at 0.313 s. The machine becomes unstable for both unsaturated and saturated models.

In Figure 4.23 all the graphs for different fault durations for saturated cases are compared. It can be seen that at time 0.212 s the system is marginally unstable and at time 0.213 s the system becomes unstable.

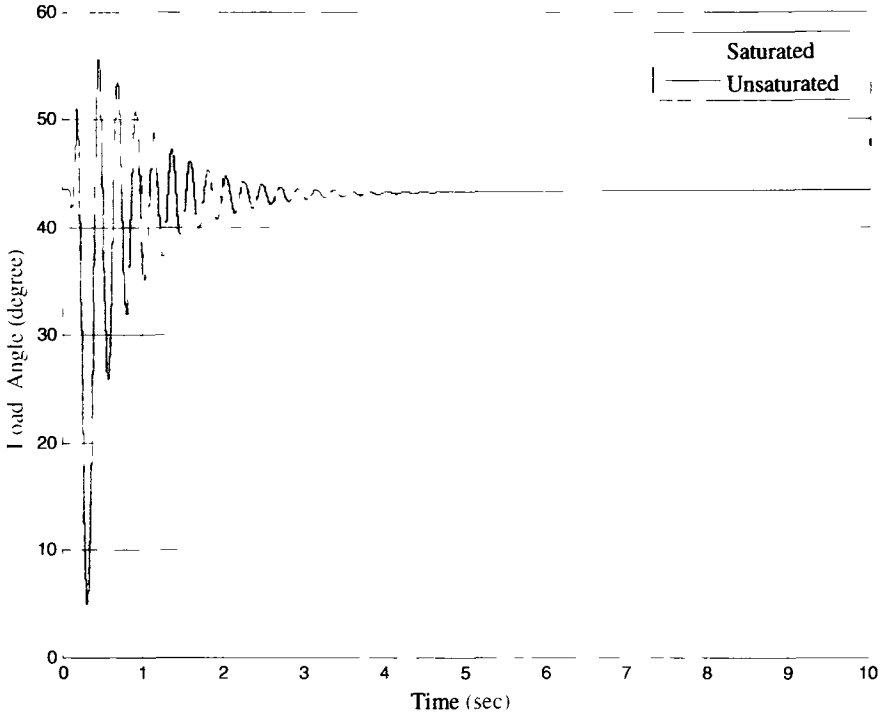


Figure 4.19. SC for 0.075 s.

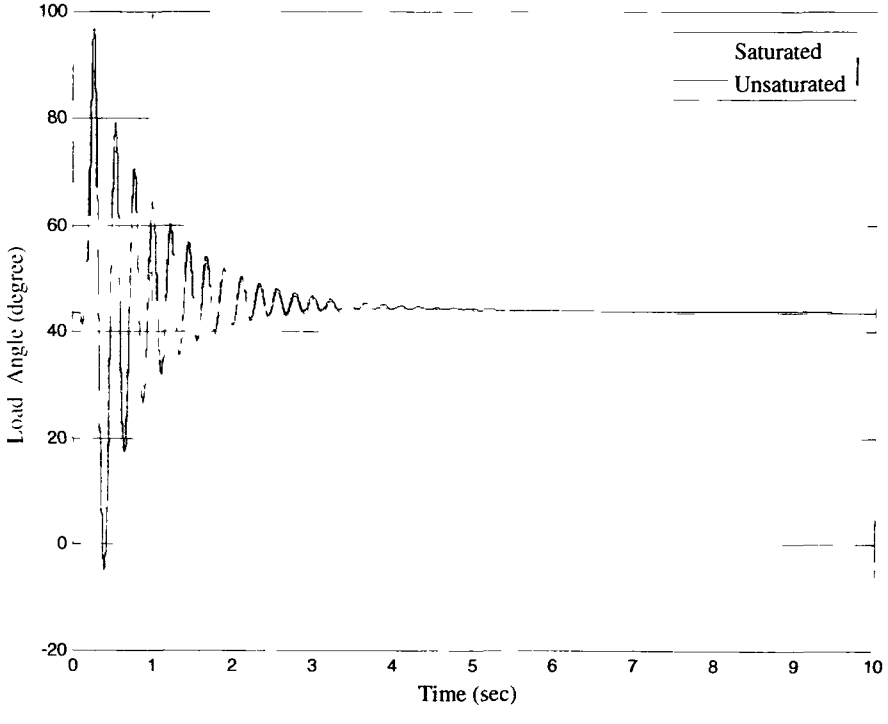


Figure 4.20. SC for 0.150 s.

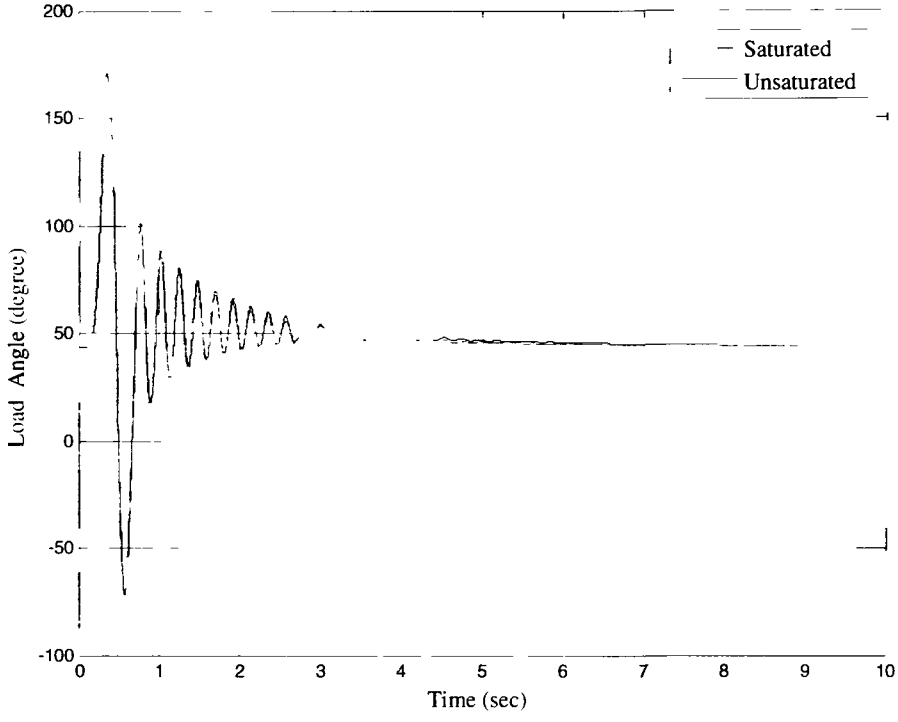


Figure 4.21. SC for 0.212 s (Marginally Stable).

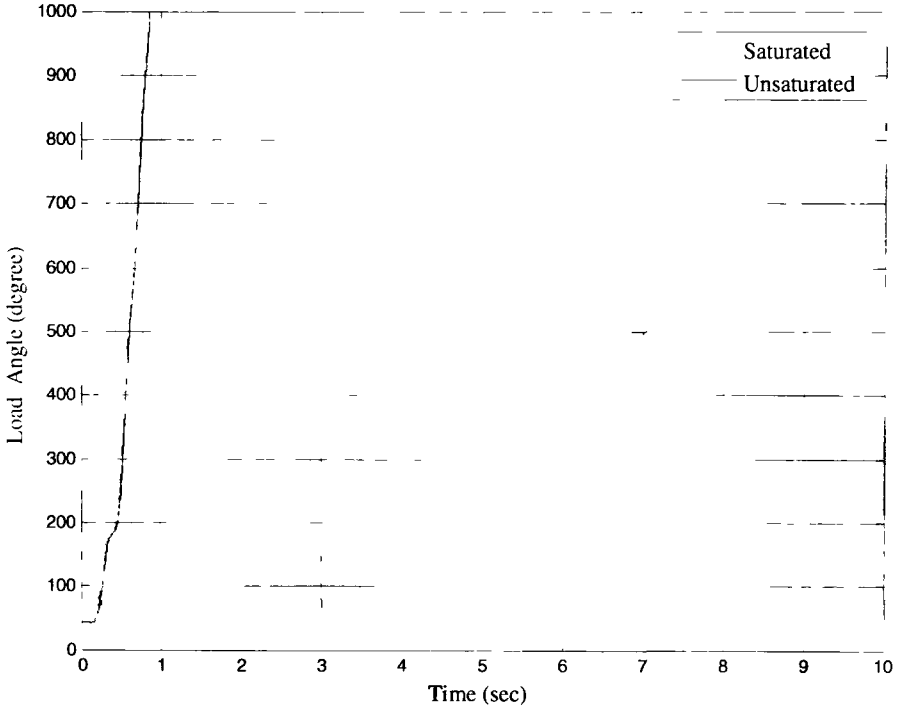


Figure 4.22. SC for 0.213sec (Unstable).

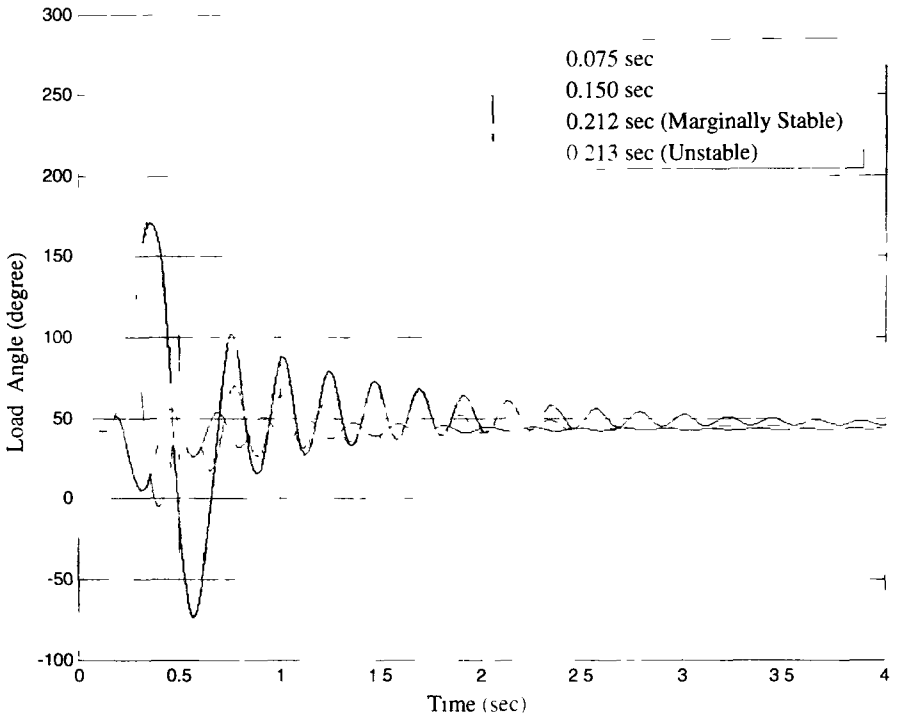


Figure 4.23. Comparison of SC for 0.075 s, 0.150 s, 0.212 s (marginally stable) & 0.213sec (unstable).

4.5 References

[1] A Murdoch, G.E. Boukarim, B.E. Gott, M.J. D'Antonio, R.A. Lawson, "Generator over excitation capability and excitation system limiters", *IEEE Power Engineering Society Winter Meeting,* vol. 1, 28 Jan.-1 Feb., pp.215 – 220, 2001.

[2] L. Lin; C. Sun and D. Mou; "Study on the excitation protection and control of synchronous generator based on the δ and s". *IEEE on Transmission and Distribution Conference and Exhibition, Asia and Pacific,* vol.15-18, pp. 1 – 4, Aug. 2005.

[3] O. Rodriguez and A. Medina, "Stability analysis of the synchronous machine under unbalance and loss of excitation conditions" *IEEE Conf. on Power Engineering Society General Meeting,* vol. 3, pp. 1508-1511, July 13-17, 2003.

[4] L. Tao, Z. Qian, W. Xiangheng, S. Pengsheng, and W. Weijian, "Dynamic performance for turbo generator under low excitation and loss of field,"

Proceedings of the Fifth International Conference on Electrical Machines and Systems, vol. 1, pp. 436-439, 2001.

[5] H. Tashakori, *Synchronous Generator Transient Behavior and Protection under Loss of Excitation fault*, M.A.Sc. Thesis, Department of Electrical and Computer Engineering, University of Windsor, 2007.

[6] D. Hiramatsu, K. Hirayama, T. Tokumasu. Y. Uemura, M. Takabatake, Y. Ishikawa, and A. Iwai, "Analysis of damper saturation characteristic on synchronous machine transient condition," *IEEE Power Engineering Society General Meeting*, pp. 1501-07, 2003.

[7] C. Sihler and A.M. Miri, "A stabilizer for oscillating torques in synchronous machines," *IEEE Transactions on Industry Applications*, vol. 41, Issue 3, pp. 748 - 755, May-June, 2005.

[8] J.H. Dymond, B. Mistry, and R. Ong, "Acceleration tests to determine salient pole synchronous motor inrush currents and torques," *IEEE Industry Applications Magazine*, vol. 8, Issue 4, pp. 44 50, July-Aug, 2002.

[9] H.H. Hwang, "Transient Analysis of Simultaneous Unbalanced Short Circuits of Synchronous Machines," *IEEE Transactions on Power Apparatus and Systems*, PAS-90, Issue 4, pp. 1815 1821, July 1971.

[10] H. H. Hwang, "Mathematical analysis of double-line-to-ground short circuit of an alternator," *IEEE Trans. Power Apparatus and Systems*, PAS-86, pp. 1254, October 1967.

[11] N.C. Kar and A.M. El-Serafi, "Effect of the main flux saturation on the transient short-circuit performance of synchronous machines," *IEEE Canadian Conference on Electrical and Computer Engineering,* pp.629 – 632, May 1-4, 2005.

[12] *IEEE Guide for synchronous generator modeling practices in stability analyses*, Std. 1110-1991.

[13] A.M. El-Serafi and A.S. Abdallah, "Saturated synchronous reactances of synchronous machines," *IEEE Transactions on Energy Conversion,* vol.7, Issue 3, pp.570 – 579, Sept. 1992.

[14] S.D. Pekarek, E.A. Walters, and B.T. Kuhn, "An efficient method of

representing saturation in physical variable models of synchronous machines," *IEEE Trans. on Energy Conversion,* vol.14. No. 1, pp. 72-79. March 1999.

[15] J. Tamura and I. Takeda, "A new model of saturated synchronous machines for power system transient stability simulations," *IEEE Trans. on Energy Conversion,* vol. 10, Issue 2, pp.218 224, June 1995.

[16] J.R. Marti and K.W Louie, "A phase-domain synchronous generator model including saturation effects," *IEEE Transactions on Power Systems,* vol. 12, Issue 1. pp.222 – 229, Feb. 1997

5 NEURAL NETWORK CHARACTERIZATION

5.1 Introduction

A function approximation problem asks to select a function among a well-defined class that closely matches ("approximates") a target function in a task-specific way. In mathematics, approximation theory is concerned with how functions can best be approximated with simpler functions, and with quantitatively characterizing the errors introduced thereby. What is meant by best and simpler will depend on the application.

5.2 Overview of Function Approximation

A closely related topic is the approximation of functions by generalized Fourier series, that is, approximations based upon summation of a series of terms based upon orthogonal polynomials.

One problem of particular interest is that of approximating a function in a computer mathematical library, using operations that can be performed on the computer or calculator (e.g. addition and multiplication), such that the result is as close to the actual function as possible. This is typically done with polynomial or rational (ratio of polynomials) approximations [1], [2].

The objective is to make the approximation as close as possible to the actual function, typically with accuracy close to that of the underlying computer's floating point arithmetic. This is accomplished by using a polynomial of high degree, and/or narrowing the domain over which the polynomial has to approximate the function. Narrowing the domain can often be done though the use of various addition or scaling formulas for the function is being approximated. Modern mathematical libraries often reduce the domain into many tiny segments and use a low-degree polynomial for each segment [2]-[5].

Once the domain and degree of the polynomial are chosen, the polynomial itself is chosen in such a way as to minimize the worst-case error. That is, the goal is to minimize the maximum value of $|P(x) - f(x)|$, where $P(x)$ is the approximating polynomial and $f(x)$ is the actual function. For well-behaved functions, the optimum Nth degree polynomial will lead to an error curve that oscillates back and forth between + ε and − ε a

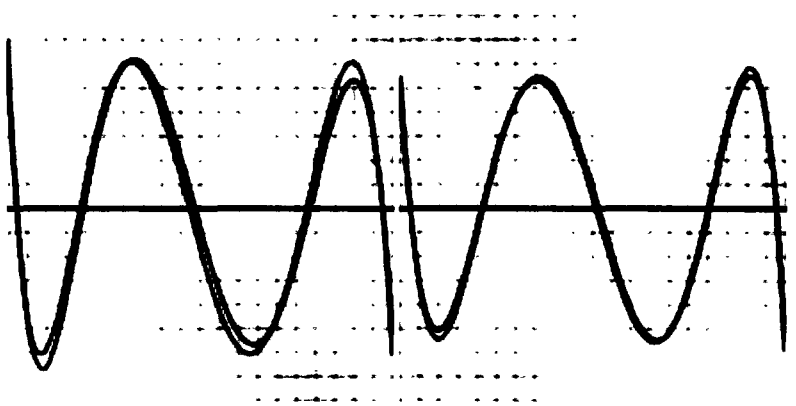
total of N+2 times, giving a worst-case error of ε. (It is possible to make contrived functions $f(x)$ for which this property does not hold, but in practice it is generally true.) Example graphs, for N=4, showing the error in approximating *log(x)* and *exp(x)*, are shown below [3], [6]. Of different kinds of function approximation problems two major classes worth mentioning: known target function approximation and unknown target function approximation.

5.2.1 Known target function approximation

Known target functions approximation theory is the branch of numerical analysis that investigates how certain known functions (for example, special functions) can be approximated by a specific class of functions (for example, polynomials or rational functions) that often have desirable properties (inexpensive computation, continuity, integral and limit values, etc.).

5.2.2 Unknown target function approximation

Second, the target function, call it g, may be unknown; instead of an explicit formula, only a set of points of the form $(x, g(x))$ is provided. Depending on the structure of the domain and co-domain of g, several techniques for approximating g may be applicable. For example, if g is an operation on the real numbers, techniques of interpolation, extrapolation, regression analysis, and curve fitting can be used [6]-[8].



(a) **(b)**

Figure 5.1. Approximation in blue and actual signal in red (a) log(*x*) (b) exp(*x*).

In this research work, function approximation used falls in the class of known target function approximation paradigm. Synchronous machine model can be realized by its input output relationship. For the mathematical model developed in chapter 2, simulation results were produced chapter 4. From the simulation results produced in chapter 4 synchronous machine can be realized by its input output relationship. This input output relationship is used in this research work to approximate the synchronous machine model using neural network. The implementation details are described in the next section.

5.3 Implementation of Function Approximation

5.3.1 Neural network

A feedforward neural network is used to approximate the input output relationship of the synchronous machine transient conditions. A Lavenberg-Marquardt backpropagation algorithm is used which is a special type of backprobagation algorithm. The Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix. The performance function has the form of a sum of squares as is typical in training feedforward networks. It is a gradient descent algorithm. The network basics are described in Table 5.1.

5.3.2 Neural network specifications

Choosing an appropriate network with appropriate size. algorithm for training a dataset can sometimes be tricky. It depends on complexity of data pattern, size of data set, accuracy and speed one want to train a network, at times experience of the trainer or even intuition. Depending on size of dataset and complexity of the problem a neural network with a single input, single output and single hidden layer is used.

The number of neuron in the input and output layer is 1, and the number of neuron in the hidden layer is 50. The size of the data points is 5000. Of this dataset 70% of the dataset that is 3500 points were used as training data. Of the rest 15% of the data that is 750 were used as validation data, which were used to validate how the network is performing while training the network using training dataset. Once the training was done

the rest 15% that is 750 data points were used to test the performance of the network. The network specifications are shown in Table 5.2.

Table 5.1. Neural network.

Algorithm	**Backpropagation**
Training	Lavenberg-Marquardt
Type	Gradient Descent
Performance	Mean Squared Error (MSE)
Data Division	Random

Table 5.2. Neural network specification.

Number of Neuron (Input Layer)	1
Number of Neuron (Output Layer)	1
Number of Neuron (Hidden Layer)	50
Number of Hidden Layers	1
Sample Size	5000 (100%)
Training Sample	3500 (70%)
Validation Sample	750 (15%)
Testing Sample	750 (15%)

5.3.3 Neural network training conditions

An epoch in neural network is defined by one round of training using all the dataset once. After training the network for one epoch, the error function is used to calculate the error value once. Until the error value reaches a certain minimum threshold, the epochs are continued; that is the training process is carried on again and again. The error curve as the number of epochs increases are also known as performance curve or simply performance. The neural network training conditions are shown in Table 5.3.

Table 5.3. Neural network training conditions.

Epoch	828
Time	0:19:41
Initial Performance	95.5
Final Performance	1.31e-07
Best Validation Performance	1.4246e-007
Best Validation Performance Epoch	822
Initial Gradient	1.0
Final Gradient	1.0076e-05
Best Gradient	1.0076e-05
Best Gradient Epoch	828
Training MSE	1.30986e-7
Validation MSE	1.42455e-7
Testing MSE	1.11054e-7

5.4 Simulation and Results

5.4.1 Loss of excitation/field (LOF)

In this section, the model approximated for machine model under loss of excitation or field. The machine is simulated for saturated model and the results are used to approximate the neural network to emulate the machine model.

In Figure 5.2, the approximation is done for 100% LOF condition for fault being constant for 0.1 s. It can be seen from this figure that the approximation is quite accurate since the training dataset, the validation dataset and the testing dataset fits into the curve almost perfectly. Figure 5.3 shows the performance curve as it decreases with the number of epochs. The number of epoch needed for training is 828 and the best validation performance is at 1.4246×10^{-7} at 822 epochs. Figure 5.4 shows the gradient, change in gradient and validation check performance throughout the training process. Figure 5.5, the approximation is done for 25% LOF condition (field excitation reduced to 25% of the rated value) for fault being persistent for 0.1 s. The Figure shows successful learning. Figure 5.6 shows the performance curve as it decreases with the number of epochs. The number of epoch needed for training is 772 and the best validation performance is at 7.782×10^{-9} at 766 epochs.

In Figure 5.7, the approximation is done for 50% LOF condition (field excitation reduced to 50% of the rated value) for fault being persistent for 0.1 s. Approximation is quite precise as can be seen from the Figure. Figure 5.8 shows the performance curve as it decreases with the number of epochs. The number of epoch needed for training is 198 and the best validation performance is at 1.4246×10^{-8} at 192 epochs. In Figure 5.9, the approximation is done for 75% LOF condition for fault being persistent for 0.1 s. It can be seen from this figure that the accuracy of the approximation is fairly perfect. Figure 5.10 shows the performance curve as it decreases with the number of epochs. The number of epoch needed for training is 235 and the best validation performance is at 1.5725×10^{-7} at 229 epochs.

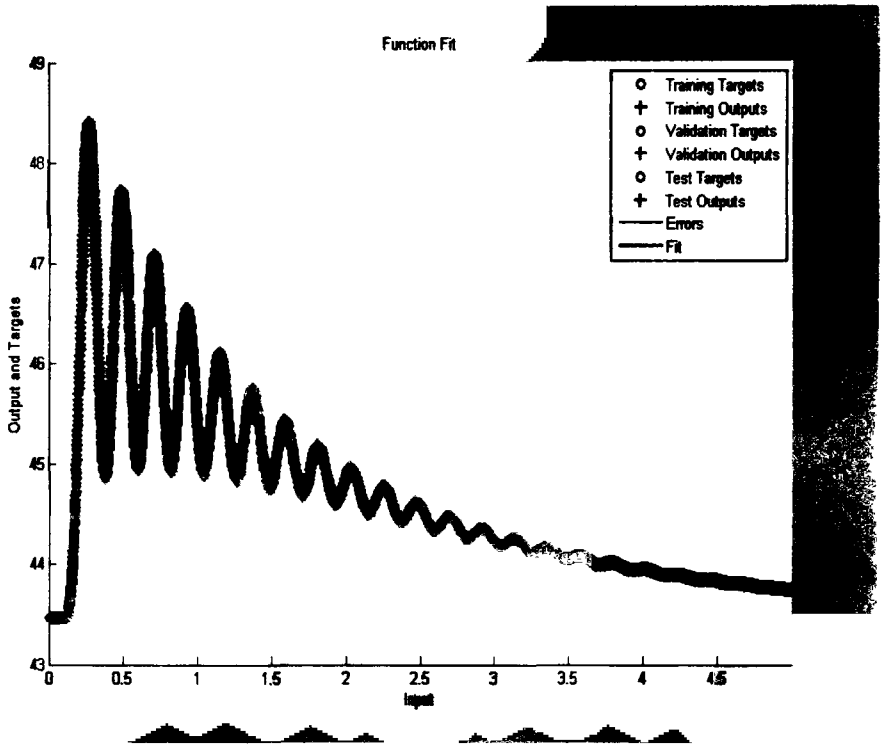


Figure 5.2. Approximation for 100% LOF (0.1 s).

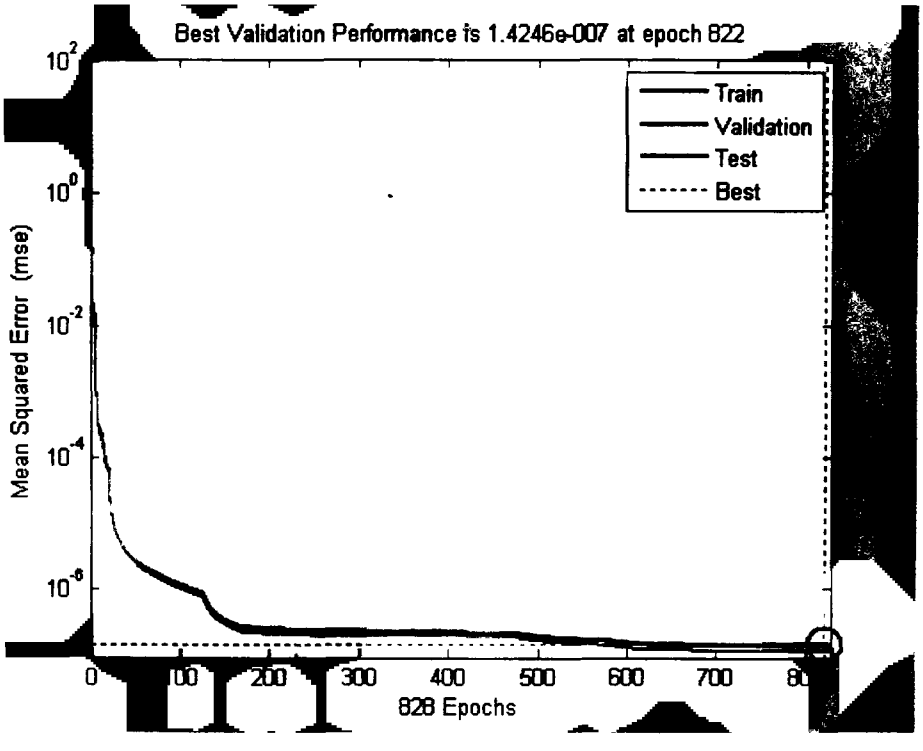


Figure 5.3. Error curve for 100% LOF (0.1 s).

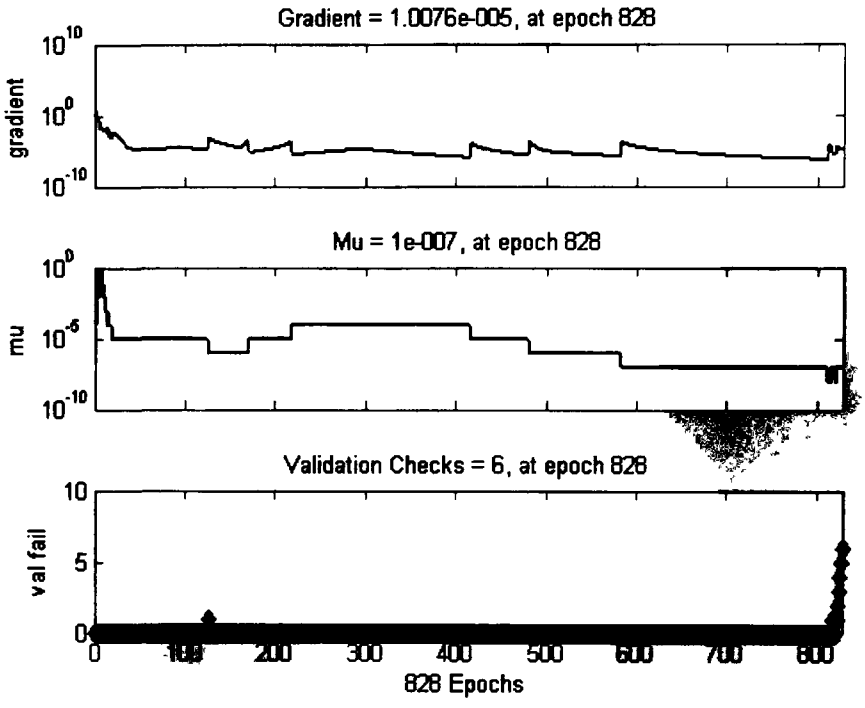


Figure 5.4. Performance graph for 100% LOF (0.1 s).

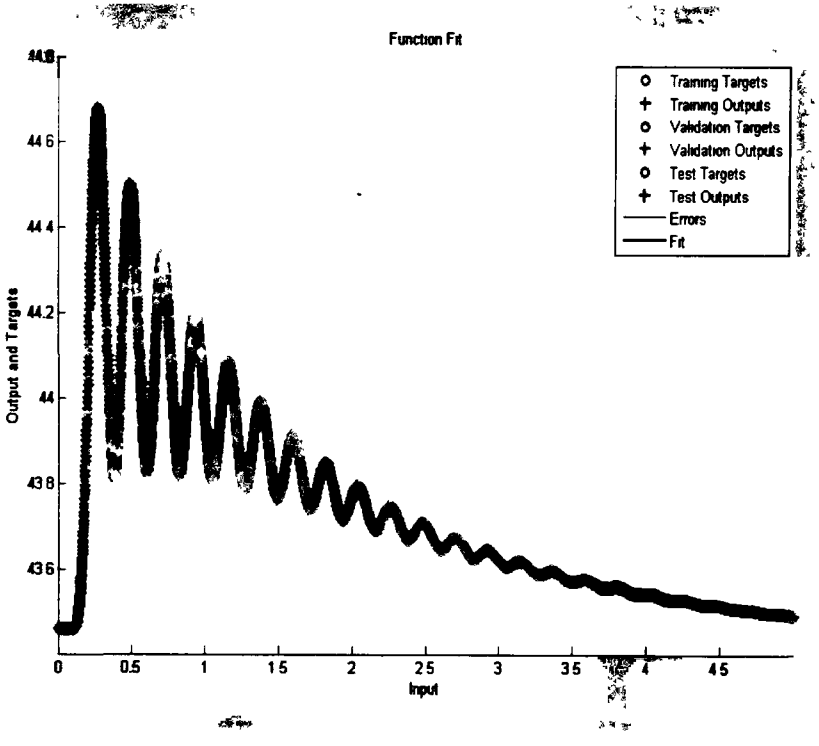


Figure 5.5. Approximation for 25% LOF (0.1 s).

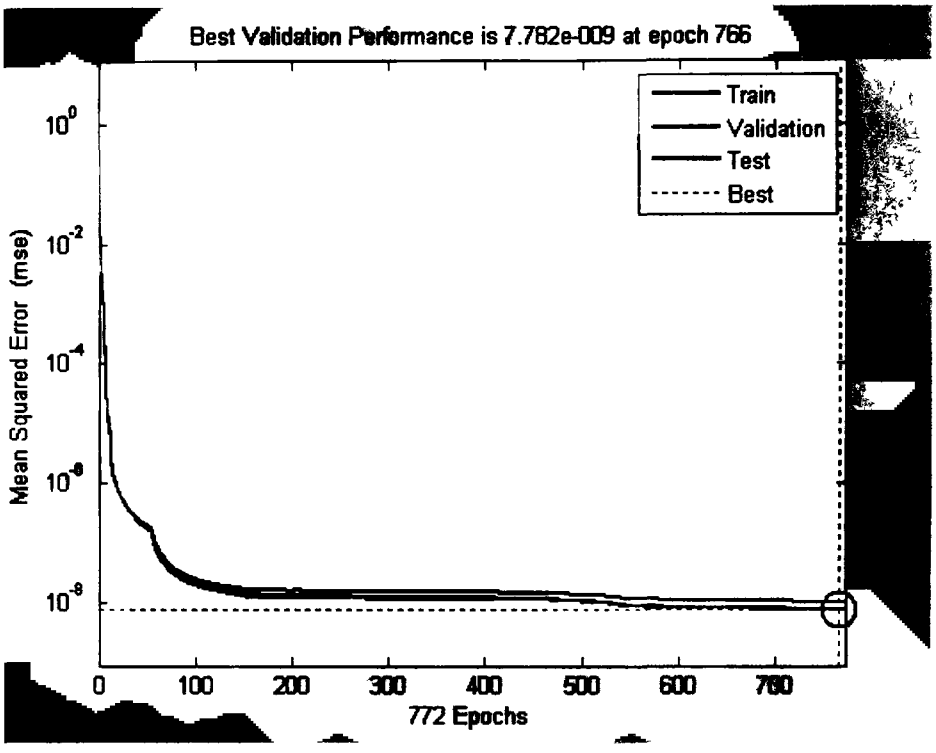


Figure 5.6. Error curve for 25% LOF (0.1 s).

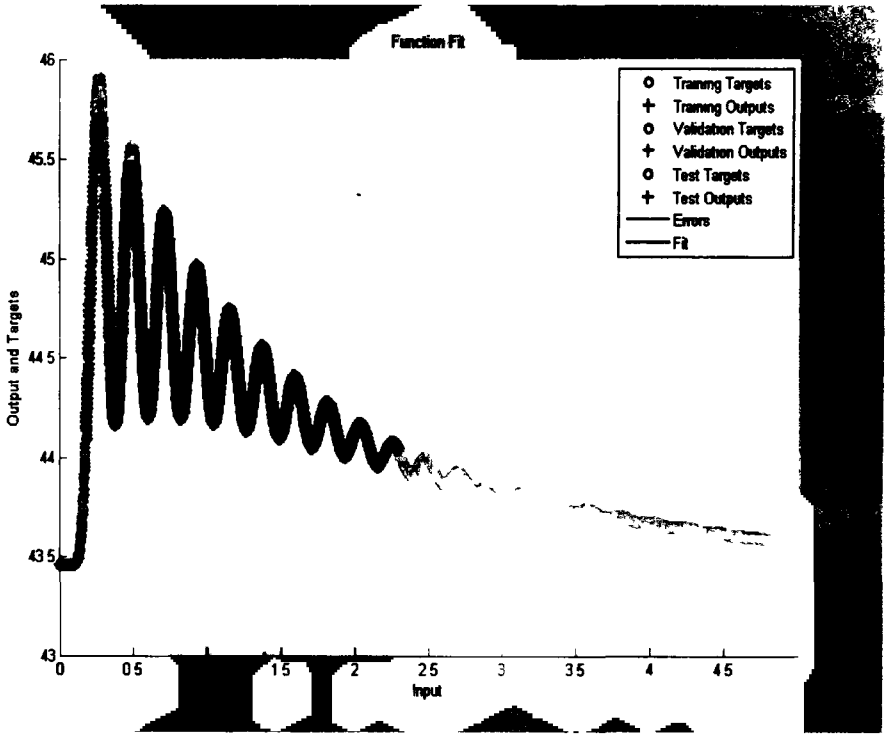


Figure 5.7. Approximation for 50% LOF (0.1 s).

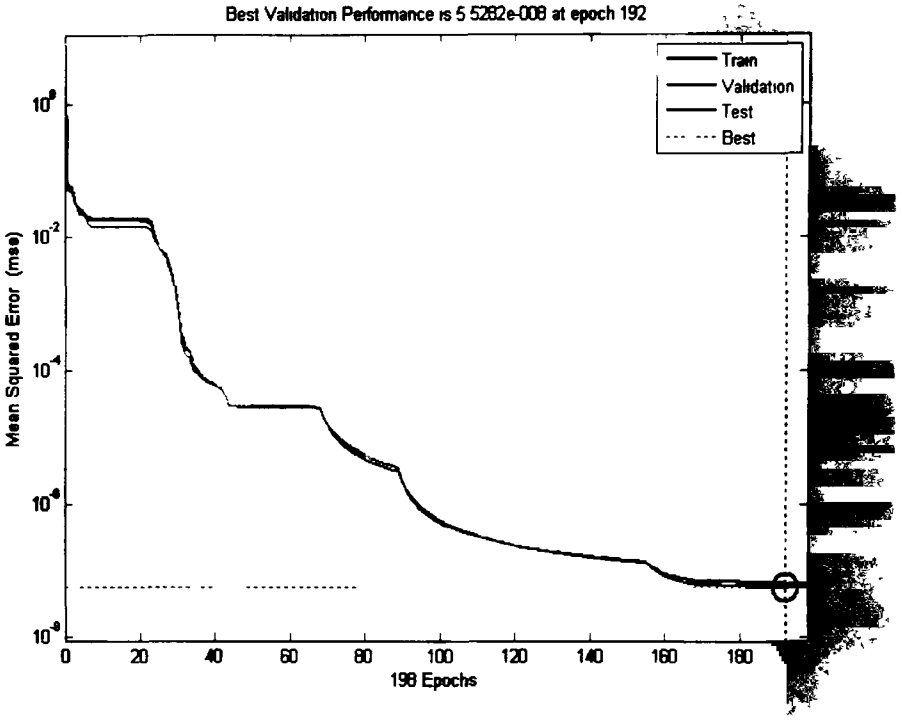


Figure 5.8. Error curve for 50% LOF (0.1 s).

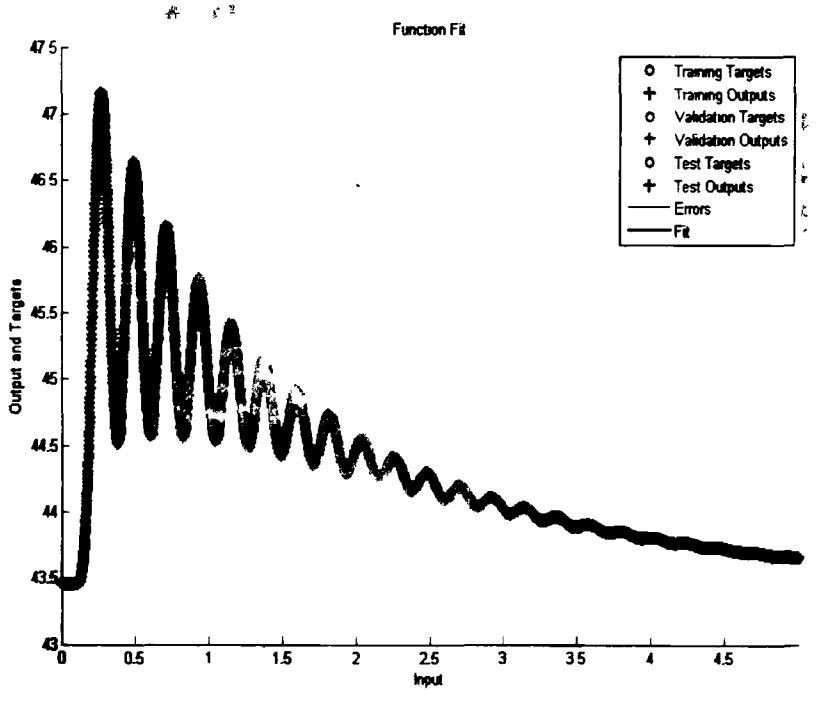


Figure 5.9. Approximation for 75% LOF (0.1 s).

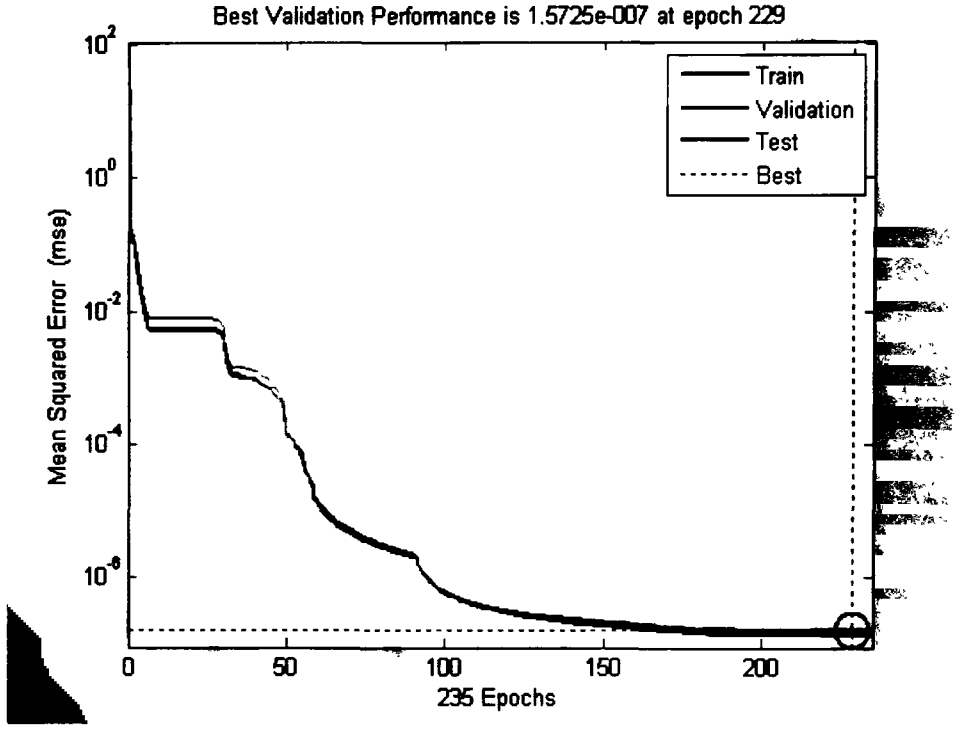


Figure 5.10. Error curve for 75% LOF (0.1 s).

In Figure 5.11, the approximation is done for 100% LOF condition (zero excitation) for fault being persistent for 0.2 s. It can be seen from this figure that the approximation is quite accurate since the training dataset, the validation dataset and the testing dataset fits into the curve almost perfectly. Figure 5.12 shows the performance curve as it decreases with the number of epochs. The number of epoch needed for training is 1000 and the best validation performance is at 3.1754×10^{-7} at 1000 epochs. In Figure 5.13, the fault stays for 0.5 s. Figure 5.14 shows the performance curve as it decreases with the number of epochs. The number of epoch needed for training is 366 and the best validation performance is at 1.8487×10^{-5} at 360 epochs. In Figure 5.15, the fault is persistent for 1 s. It is shown in the figure that, here as well the curve fitting is almost perfect Figure 5.16 shows the performance curve as it decreases with the number of epochs. The number of epoch needed for training is 1000 and the best validation performance is at 3.6538×10^{-6} at 1000 epochs.

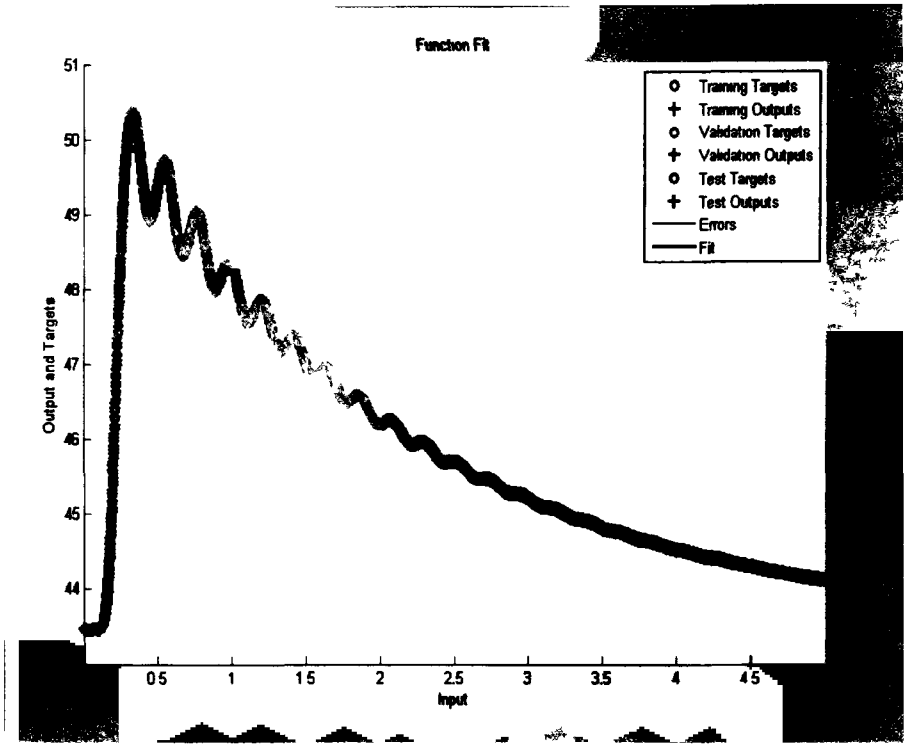


Figure 5.11. Approximation for 100% LOF (0.2 s).

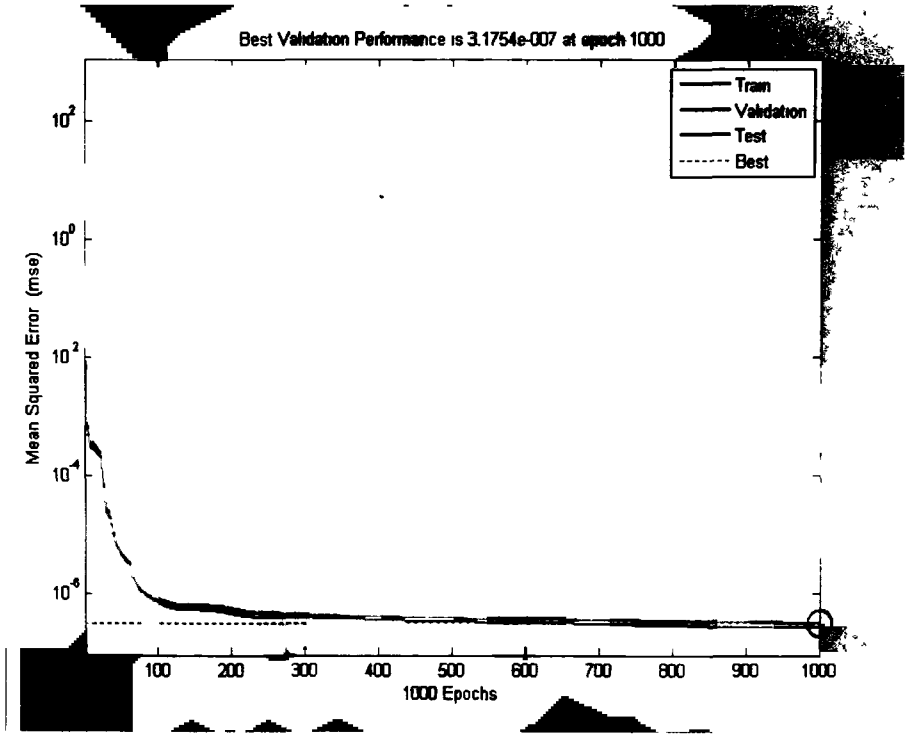


Figure 5.12. Error curve for 100% LOF (0.2 s).

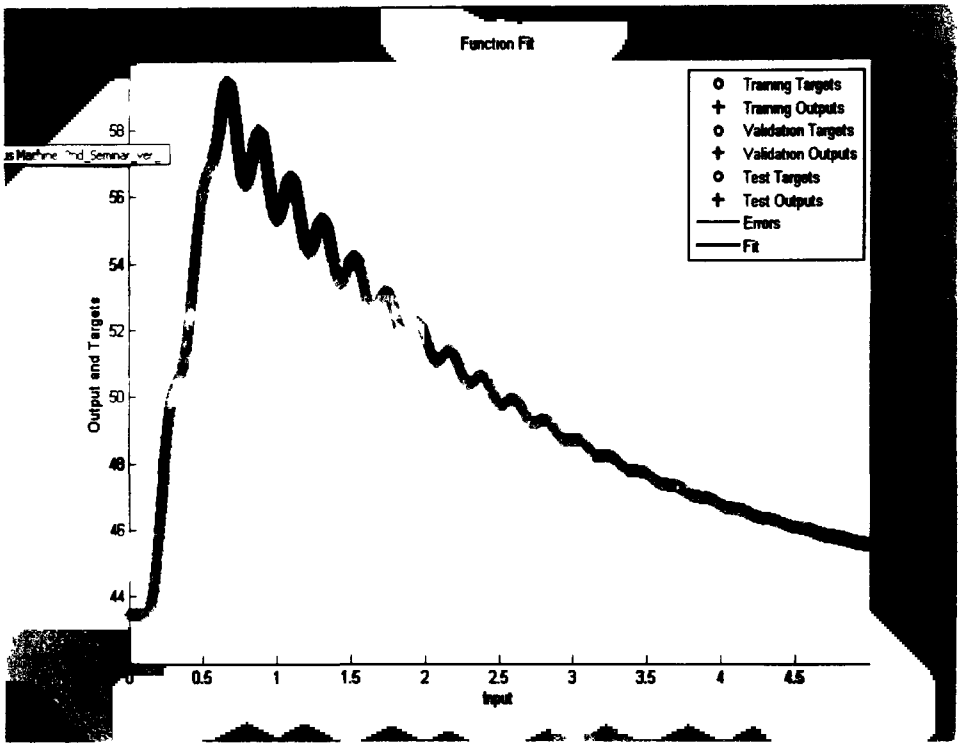


Figure 5.13. Approximation for 100% LOF (0.5 s).

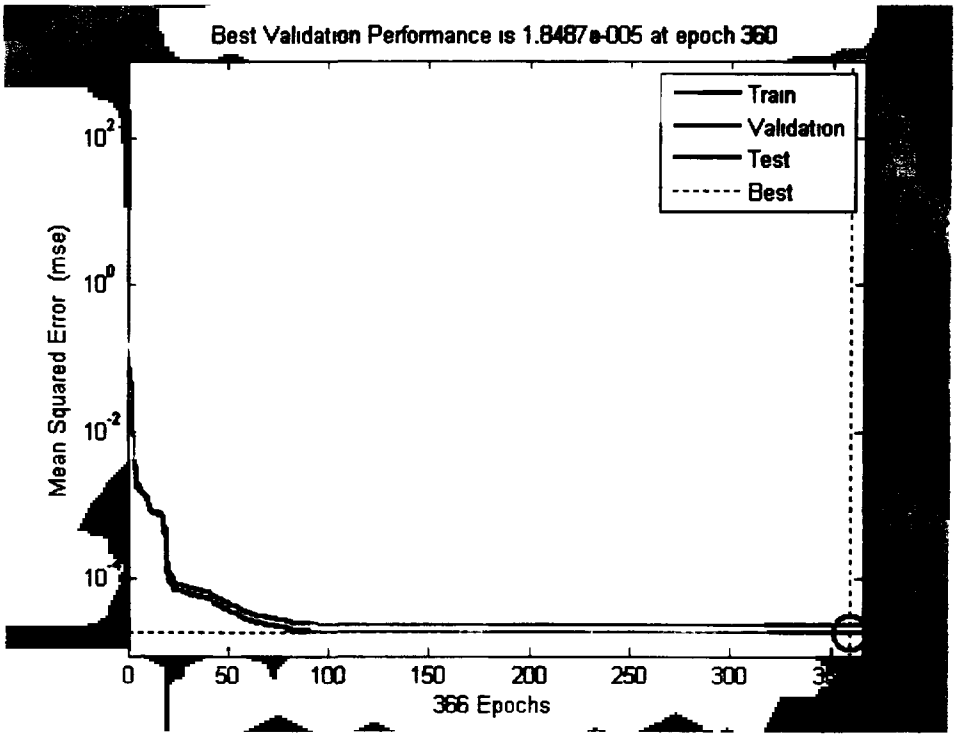


Figure 5.14. Error curve for 100% LOF (0.5 s).

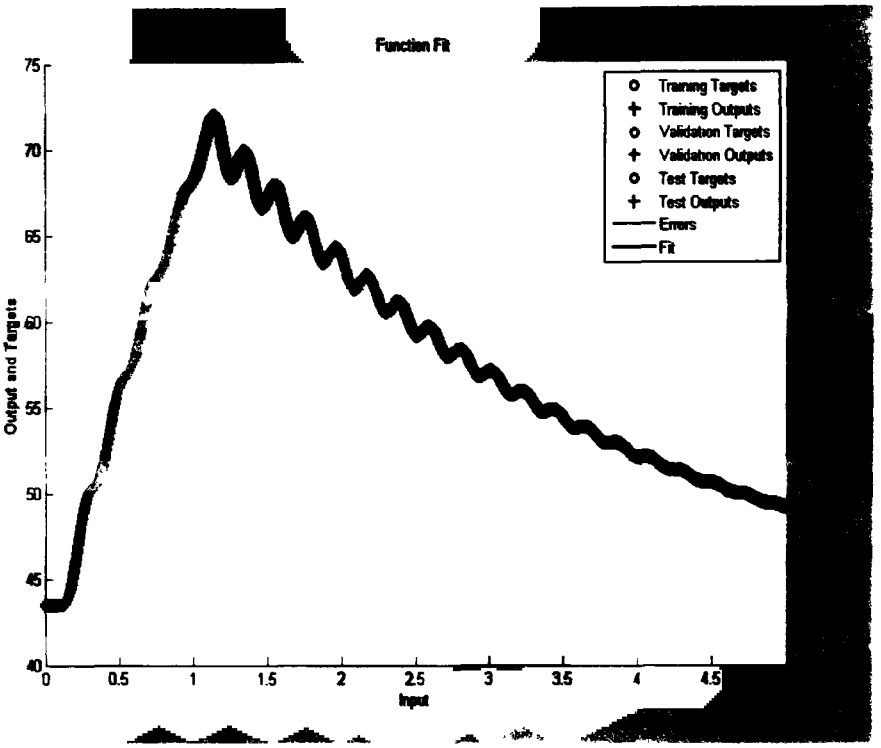


Figure 5.15. Approximation for 100% LOF (1 s).

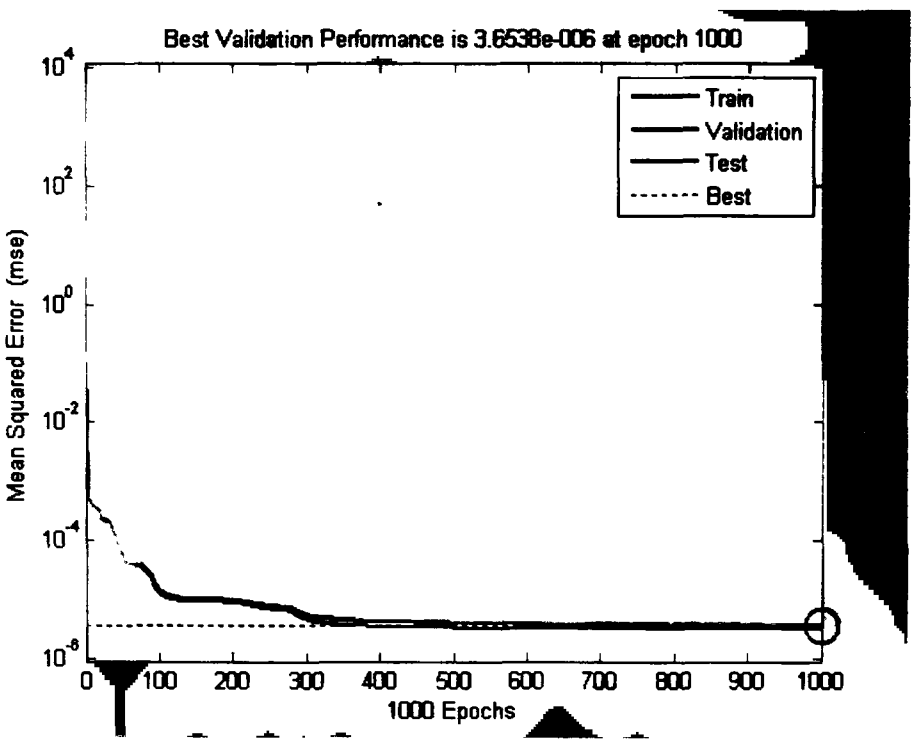


Figure 5.16. Error curve for 100% LOF (1 s).

5.4.2 Disturbance in torque (DIT)

In this section, the machine model approximated for machine model under disturbance in torque. The machine is simulated for saturated model and the results are used to approximate the neural network to emulate the machine model.

In Figure 5.17, the approximation is done for 50% loss in torque input for fault being persistent for 0.1 s. It can be seen from this figure that the approximation is quite accurate since the training dataset, the validation dataset and the testing dataset fits into the curve almost perfectly. Figure 5.18 shows the performance curve as it decreases with the number of epochs. The number of epoch needed for training is 1000 and the best validation performance is at 1.0897×10^{-4} at 1000 epochs. In Figure 5.19, the fault is persistent for 0.1 s. It can be seen from the Figure that the approximation is quite accurate in this case as well. Figure 5.20 shows the performance curve as it decreases with the number of epochs. The number of epoch needed for training is 1000 and the best validation performance is at 1.9652×10^{-3} at 1000 epochs.

In Figure 5.21, the approximation is done for 150% of rated torque condition for fault being persistent for 0.1 s. Figure 5.22 shows the performance curve as it decreases with the number of epochs. The number of epoch needed for training is 1000 and the best validation performance is at 9.7553×10^{-5} at 1000 epochs. In Figure 5.23, the torque is doubled for 0.1 s. Figure 5.24 shows the performance curve as it decreases with the number of epochs. The number of epoch needed for training is 928 and the best validation performance is at 3.0472×10^{-6} at 928 epochs.

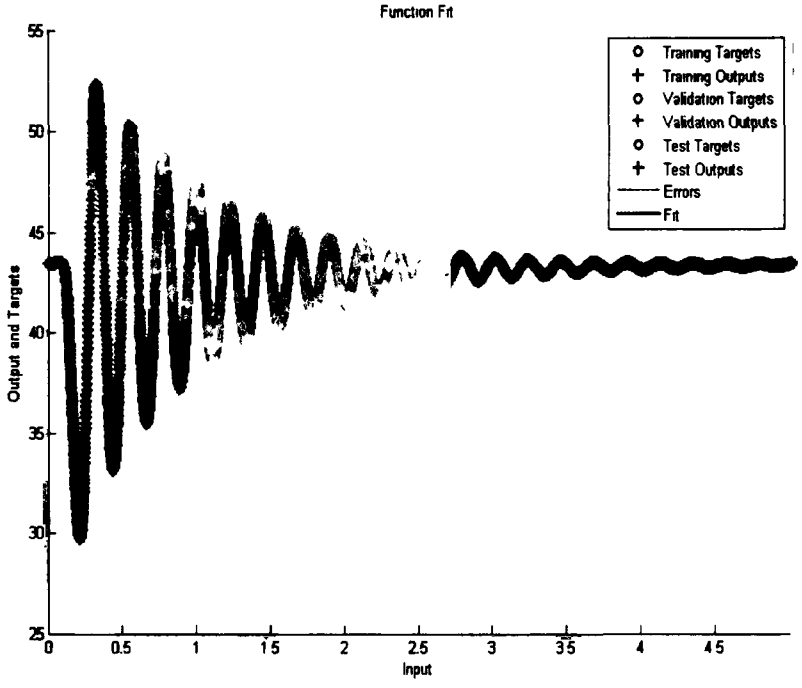


Figure 5.17. Approximation for 50% loss in DIT (0.1 s).

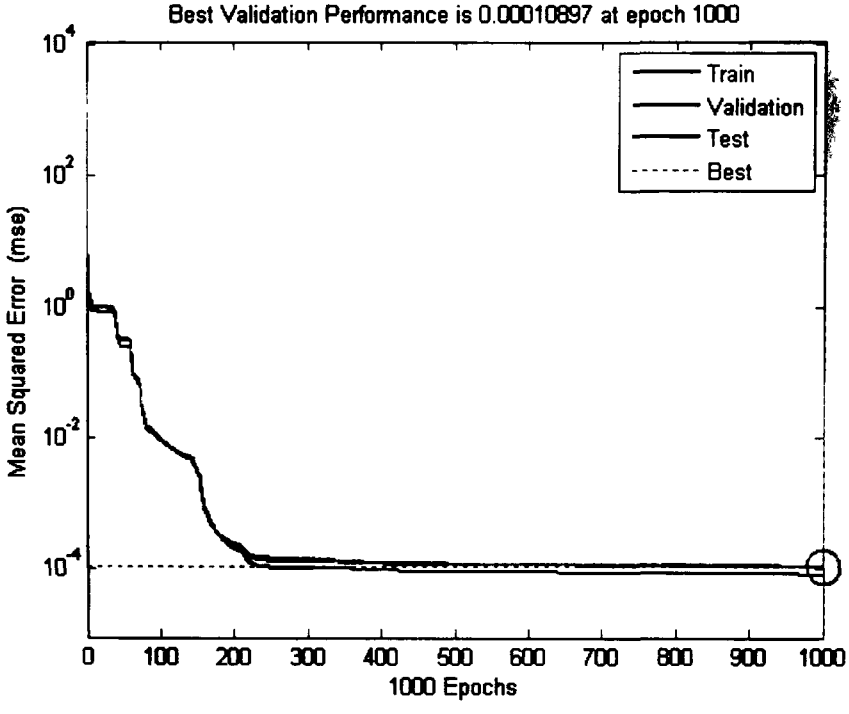


Figure 5.18. Error curve for 50% loss in DIT (0.1 s).

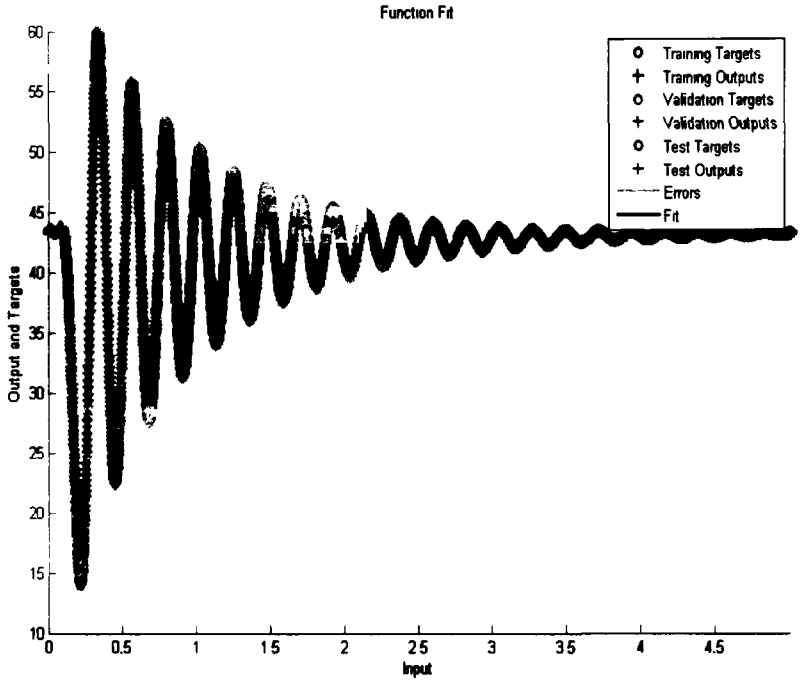


Figure 5.19. Approximation for 100% loss in DIT (0.1 s).

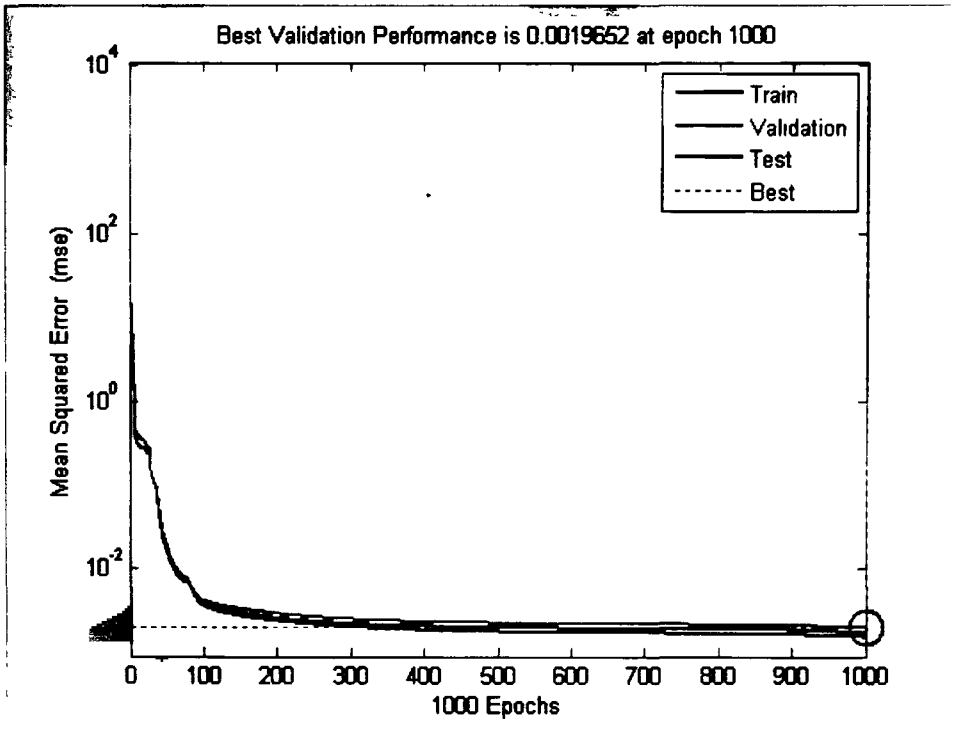


Figure 5.20. Error curve for 100% loss in DIT (0.1 s).

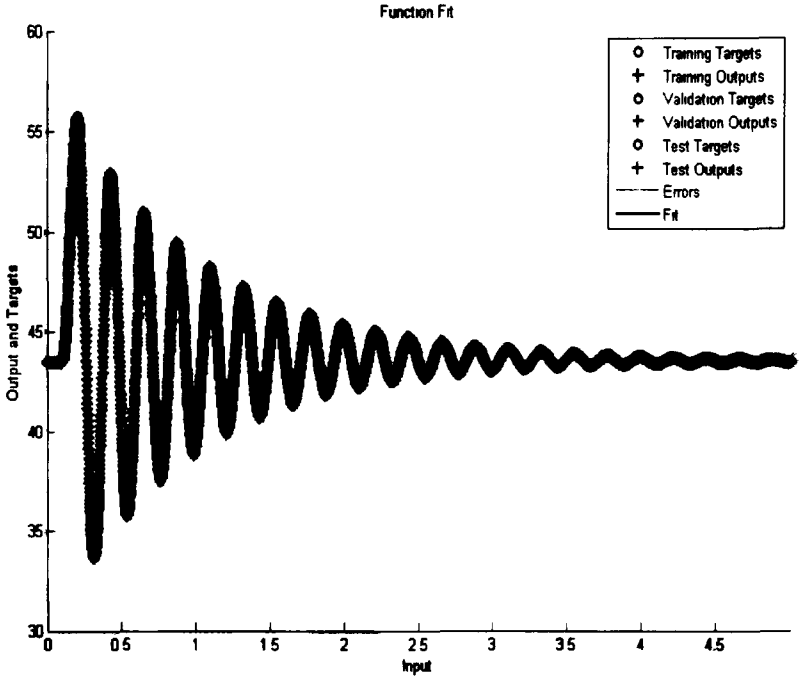


Figure 5.21. Approximation for 50% over-excitation in DIT (0.1 s).

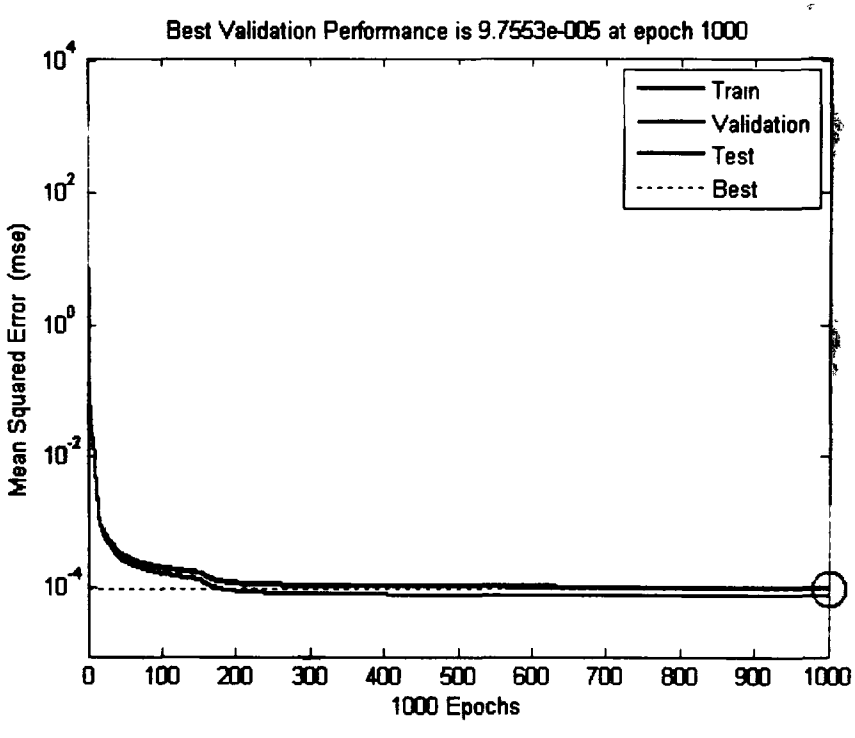


Figure 5.22. Error curve for 50% over-excitation in DIT (0.1 s).

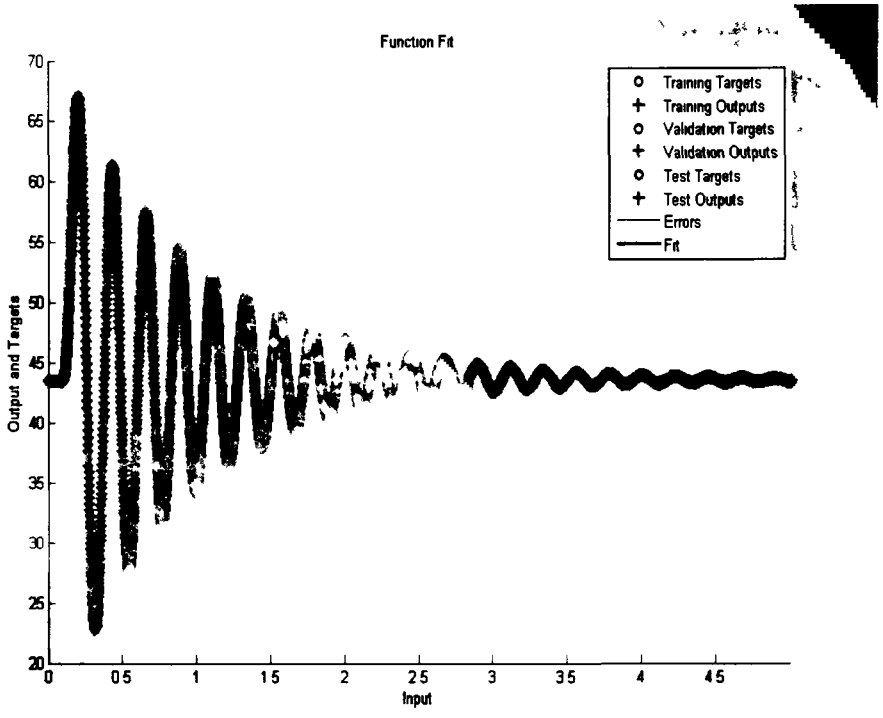


Figure 5.23. Approximation for 100% over-excitation in DIT (0.1 s).

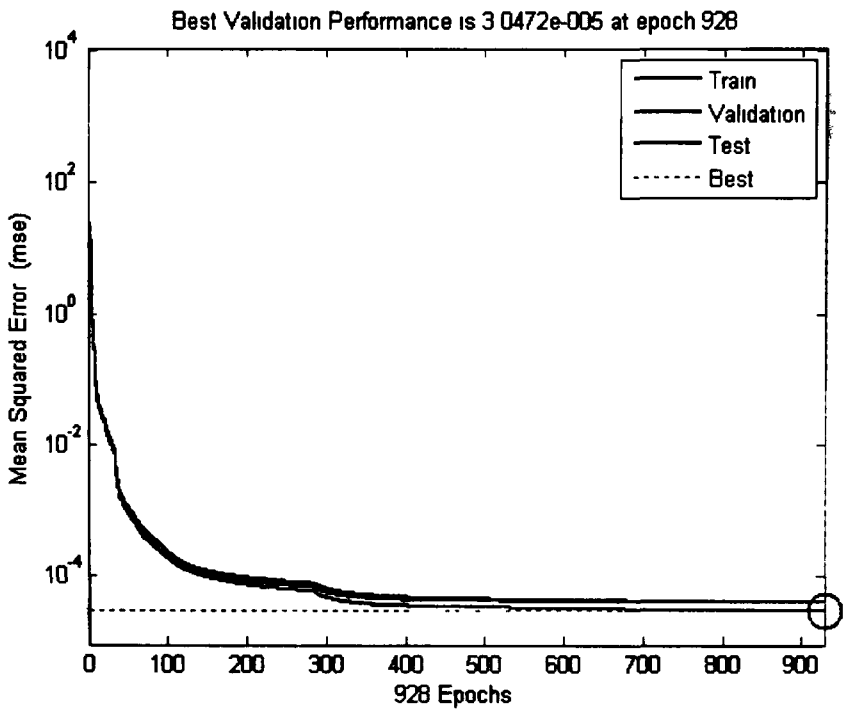


Figure 5.24. Error curve for 100% over-excitation in DIT (0.1 s).

In Figure 5.25, the approximation is done for 100% loss in input torque and it stays for 0.2 s. It can be seen from the Figure that the approximation is quite accurate since the training dataset, the validation dataset and the testing dataset fits into the curve almost perfectly. Figure 5.26 shows the performance curve as it decreases with the number of epochs. The number of epoch needed for training is 611 and the best validation performance is at 1.3398×10^{-5} at 605 epochs. In Figure 5.27, the approximation is done for 100% loss in input torque condition for fault being persistent for 0.5 s. Figure 5.28 shows the performance curve as it decreases with the number of epochs. The number of epoch needed for training is 270 and the best validation performance is at 1.7829×10^{-3} at 264 epochs. In Figure 5.29. the approximation is done for 100% loss in torque for 1 s. Figure 5.30 shows the performance curve as it decreases with the number of epochs. The number of epoch needed for training is 1000 and the best validation performance is at 1.8159×10^{-3} at 1000 epochs.

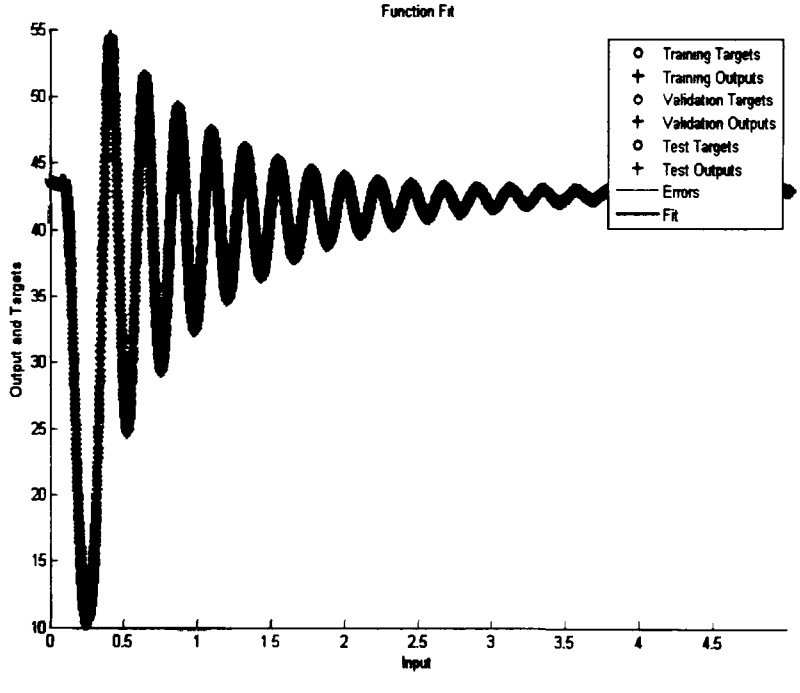


Figure 5.25. Approximation for 100% loss in torque (0.2 s).

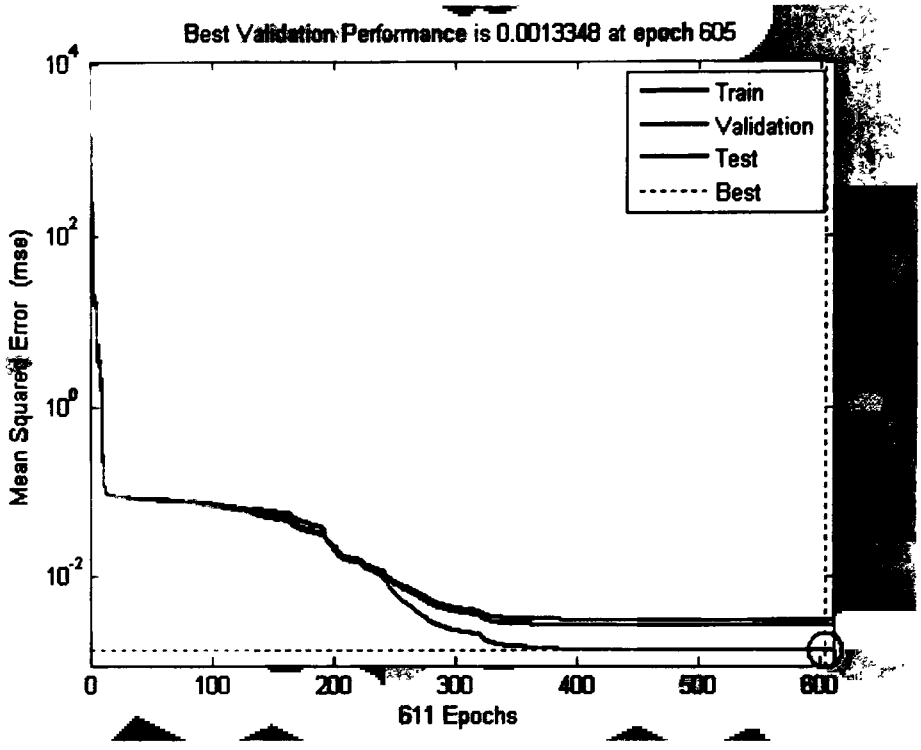


Figure 5.26. Error curve for 100% loss in torque (0.2 s).

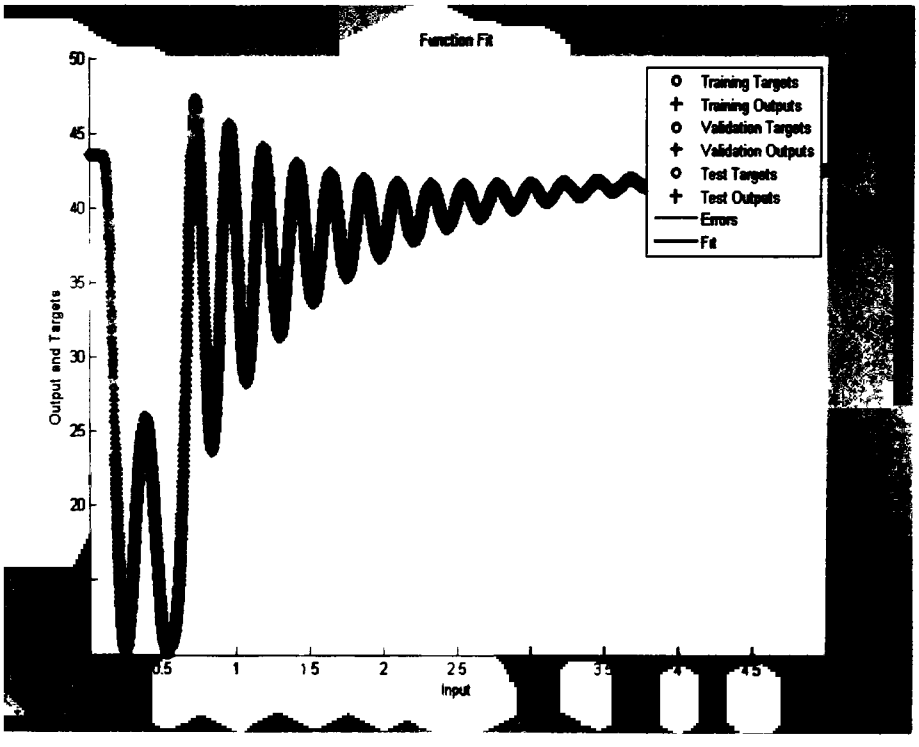


Figure 5.27. Approximation for 100% loss in torque (0.5 s).

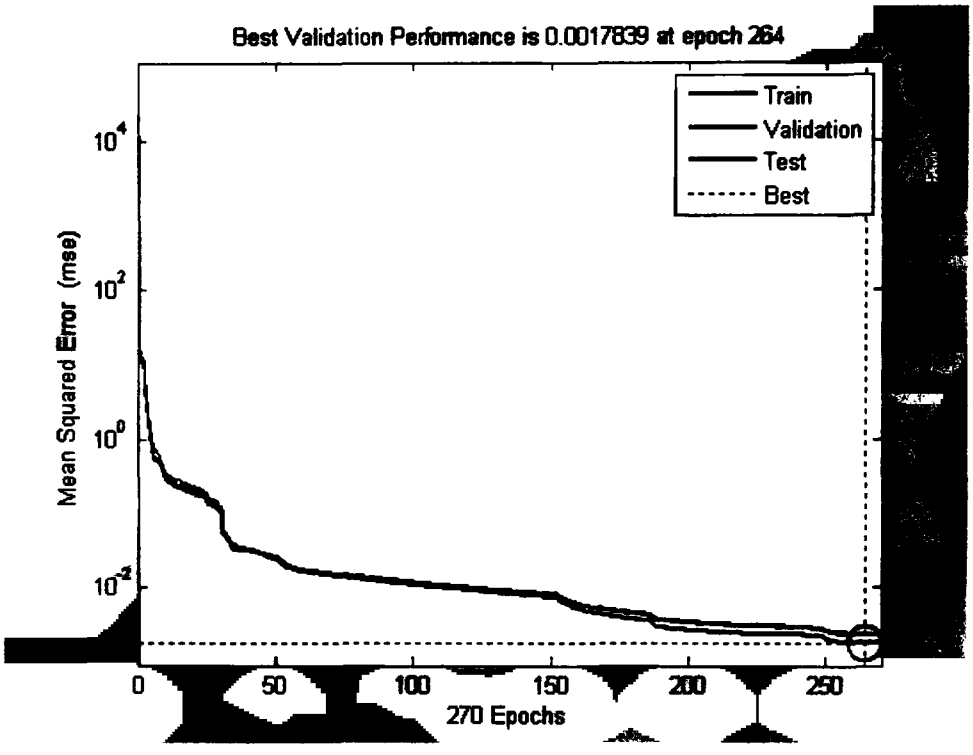


Figure 5.28. Error curve for 100% loss in torque (0.5 s).

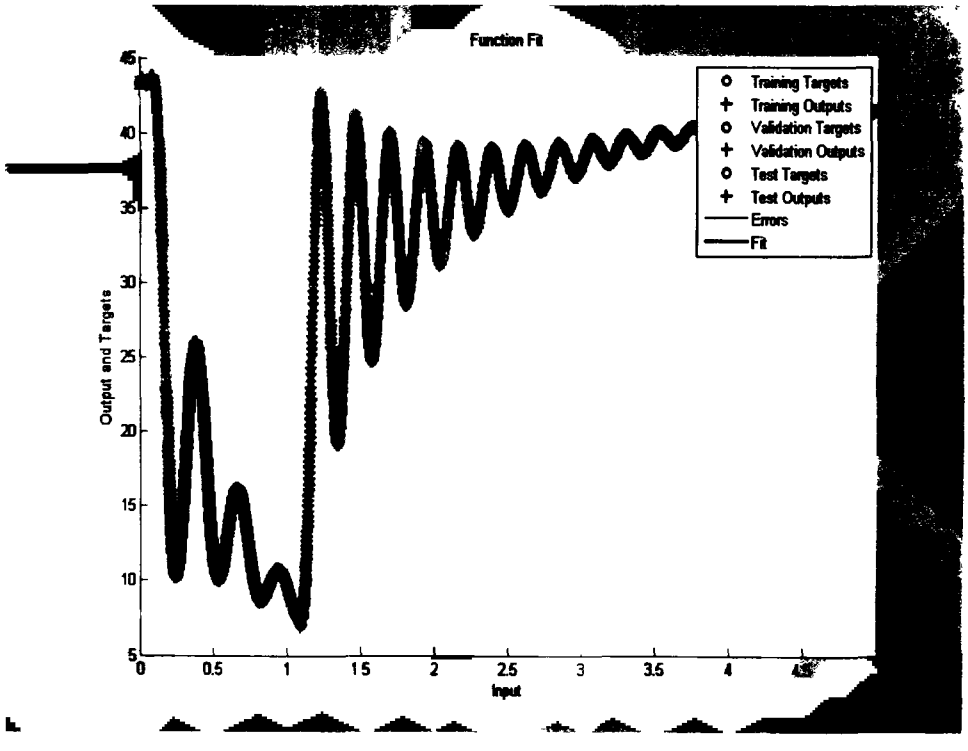


Figure 5.29. Approximation for 100% loss in torque (1 s).

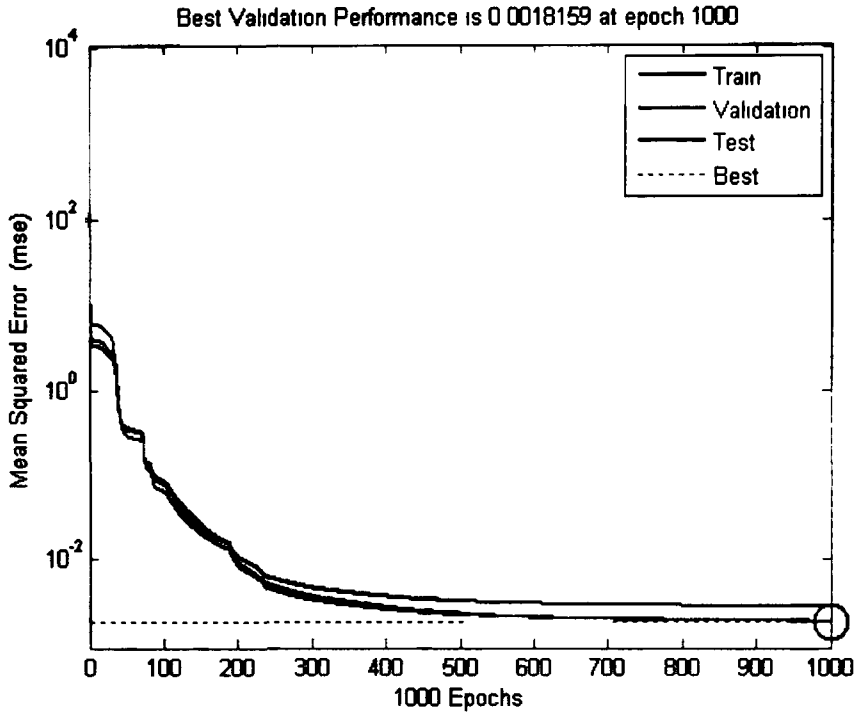


Figure 5.30. Error curve for 100% loss in torque (1 s).

5.4.3 Short circuit (SC)

In this section, the machine model approximated for machine model under short circuit. The machine is simulated for saturated model and the results are used to approximate the neural network to emulate the machine model.

In Figure 5.31, the approximation is done for short circuit condition for fault being persistent for 0.075 s. It can be seen from the Figure that the approximation is quite accurate. Figure 5.32 shows the performance curve as it decreases with the number of epochs. The number of epoch needed for training is 503 and the best validation performance is at 9.5679×10^{3} at 497 epochs. In Figure 5.33, the approximation is done for short circuit condition for fault being persistent for 0.15 s. Figure 5.34 shows the performance curve as it decreases with the number of epochs. The number of epoch needed for training is 1000 and the best validation performance is at 3.138×10^{-3} at 1000 epochs.

In Figure 5.35, the approximation is done for short circuit condition for fault being persistent for 0.212 s. It can be seen from this figure that the approximation is very precise. Figure 5.36 shows the performance curve as it decreases with the number of epochs. The number of epoch needed for training is 678 and the best validation performance is at 1.3769×10^{-2} at 678 epochs.

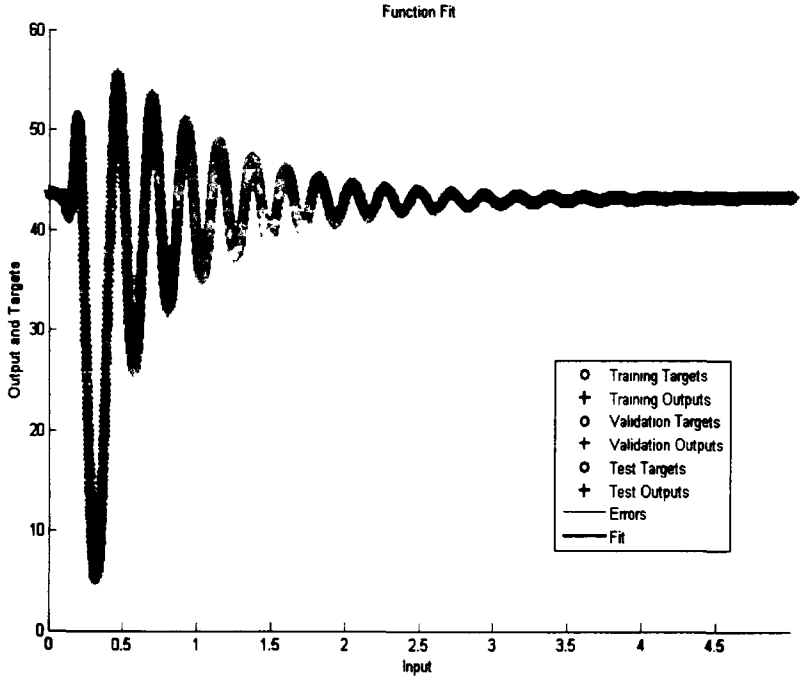


Figure 5.31. Approximation for SC for 0.075 s.

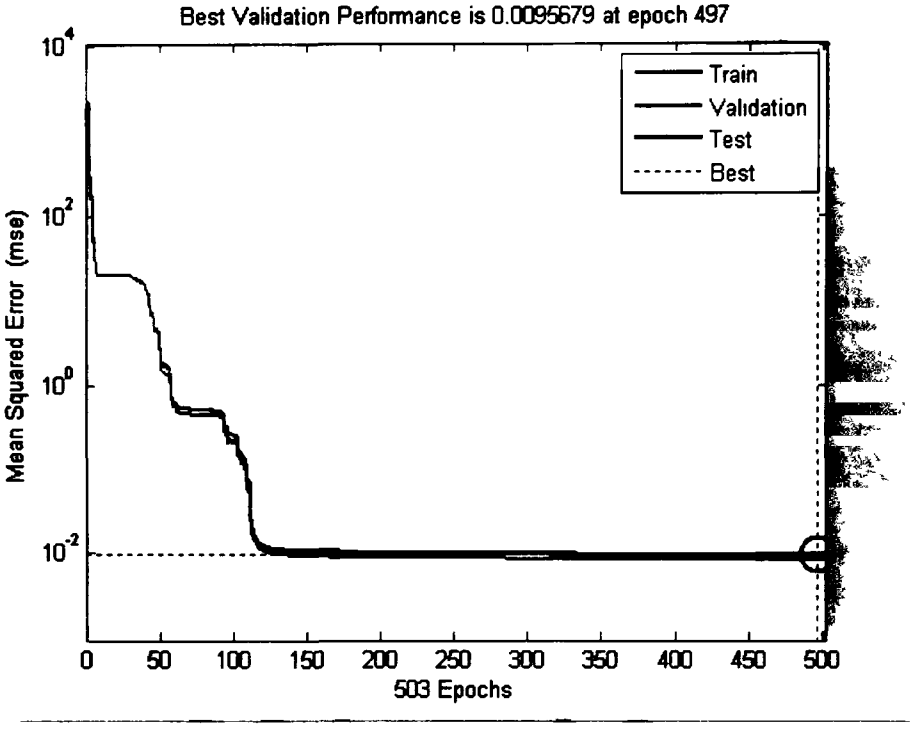


Figure 5.32. Error curve for SC for 0.075 s.

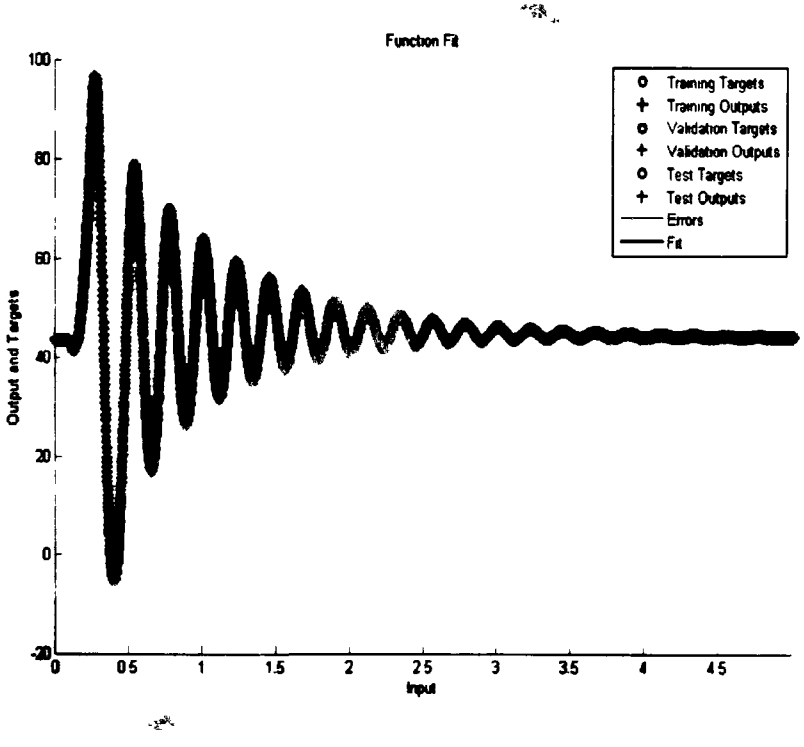


Figure 5.33. Approximation for SC for 0.150 s.

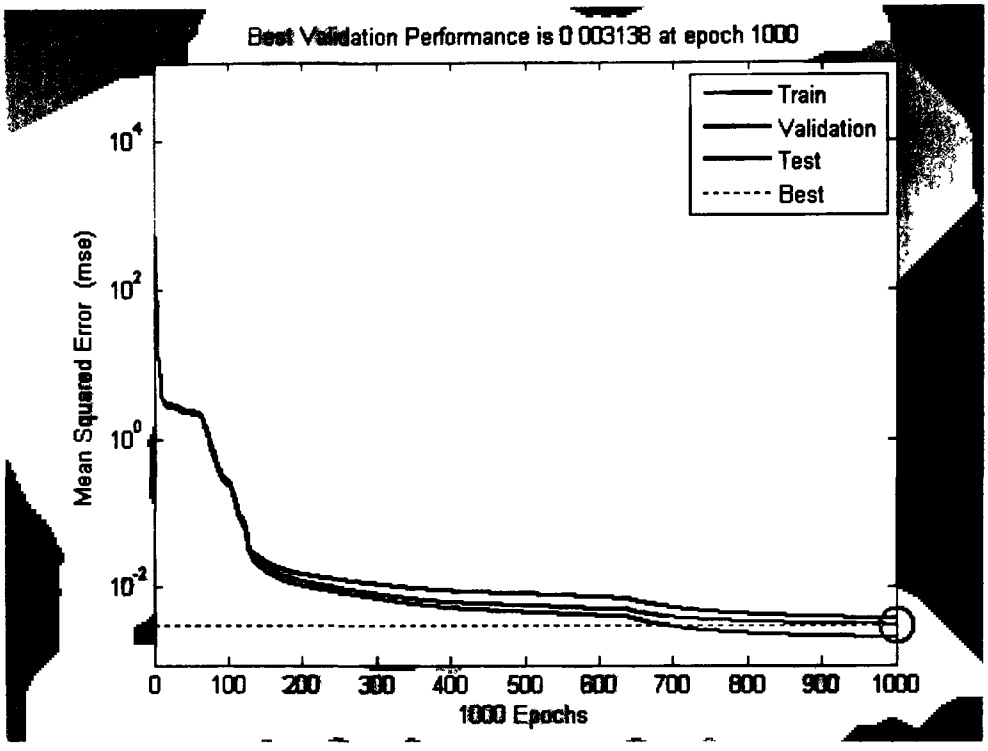


Figure 5.34. Error curve for SC for 0.150 s.

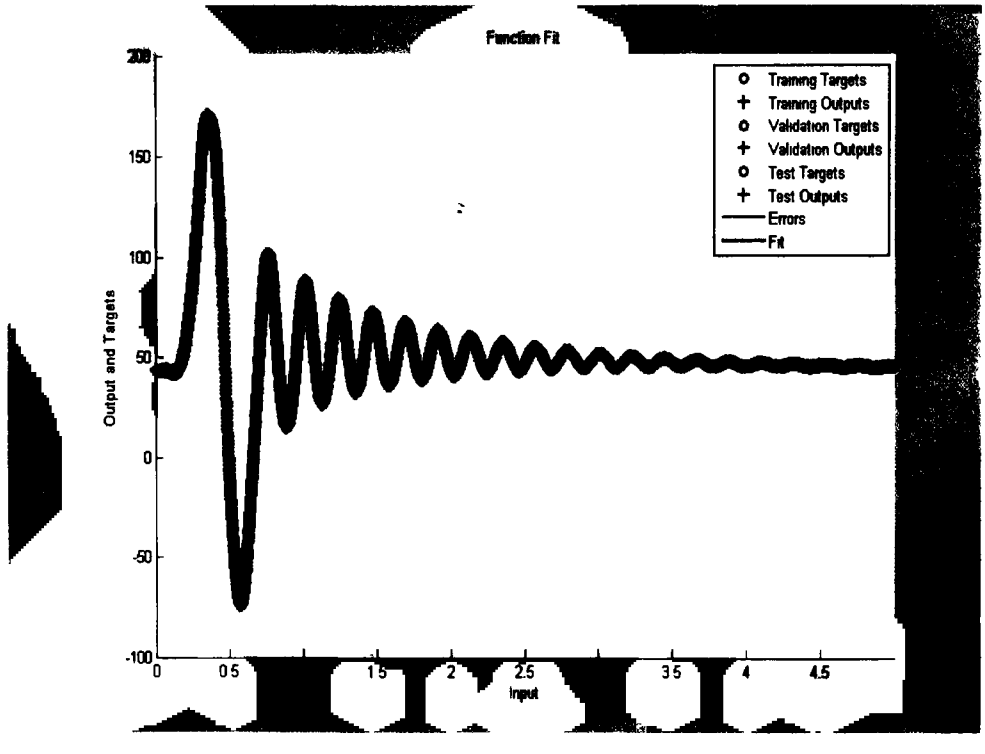


Figure 5.35. Approximation for SC for 0.212 s (Marginally Stable).

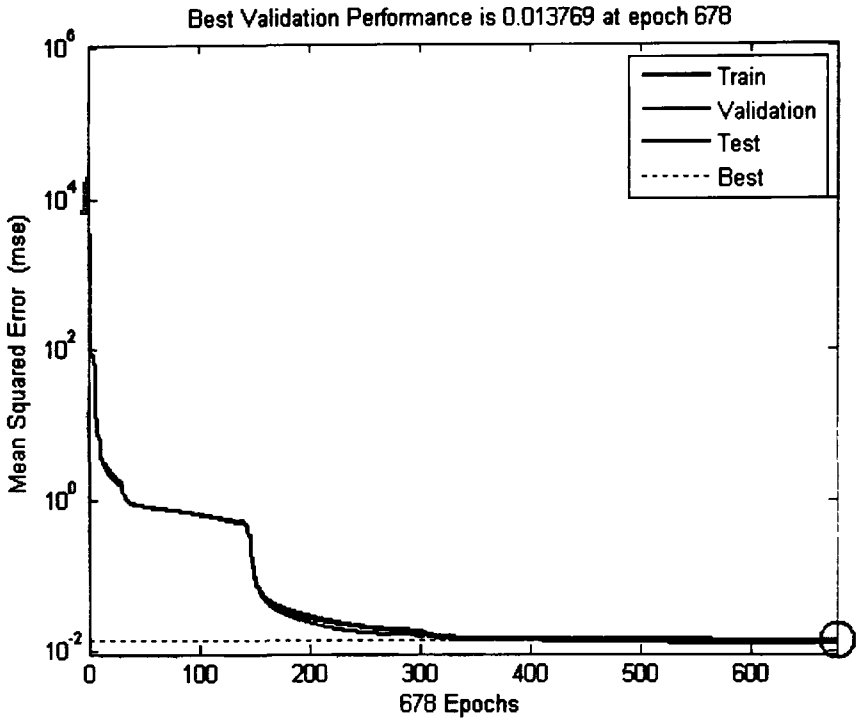


Figure 5.36. Error curve for SC for 0.212 s (Marginally Stable).

5.5 References

[1] A J. Moody and C. J. Darken, "Fast learning in networks of locally tuned processing units," *Neural Computation*, 1989.

[2] T. Poggio and F Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol. 78(9), pp. 1484-1487, 1990.

[3] R.D. Jones, Y.C. Lee, C.W. Barnes, G.W. Flake, K. Lee, P.S. Lewis, and S. Qian, "Function approximation and time series prediction with neural networks," *Proceedings of the International Joint Conference on Neural Networks*, June 17-21, p. I-649, 1990.

[4] M.D. Buhmann and M.J. Ablowitz, *Radial Basis Functions : Theory and Implementations*, Cambridge University, 2003.

[5] P.V. Yee, and S. Haykin, *Regularized Radial Basis Function Networks: Theory and Applications*, John Wiley, 2001.

[6] J.R. Davies, S.V Coggeshall, R.D. Jones, and D. Schutzer, *Intelligent Security*

Systems, Artificial Intelligence in the Capital Markets, Chicago, 1995.

[7] S. Haykin, *Neural Networks: A Comprehensive Foundation (2nd edition)*, Upper Saddle River, NJ: Prentice Hall, 1999.

[8] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks", *IEEE Transactions on Neural Networks*, vol 2, no. 2 March 1991.

6 NEURAL NETWORK CLUSTERING

6.1 Introduction

Clustering in general terms means grouping of the same or similar elements gathered or occurring closely together. It can be considered as a subtopic of computational data processing.

In data processing, data are defined as numbers or characters that represent measurements from observable phenomena. A single datum is a single measurement from observable phenomena. Measured information is then algorithmically derived and/or logically deduced and/or statistically calculated from multiple data. Information is defined as either a meaningful answer to a query or a meaningful stimulus that can cascade into further queries [1].

A neural network based clustering can be considered as an algorithm exploring the similarity between patterns and places where similar patterns are put in clusters. Best known applications include data compression and data mining.

6.2 Overview of Clustering

Data clustering deals with the problem of classifying a set of N objects into groups so that objects within the same group are more similar than objects belonging to different groups. Each object is identified by a number m of measurable features, consequently, i^{th} object can be represented as a point $i \times \hat{I}_{Rm}$, $i = 1, 2,..., N$. Data clustering aims at identifying clusters as more densely populated regions in the space Rm [2]-[4].

This is a traditional problem of unsupervised pattern recognition. A lot of approaches to solve this problem were suggested. The general strategy is as follows: first, somehow or other one finds the optimal partition of the points into K classes, and then changes the value of the parameter K from N to 1. Here, the main interest is the way how small classes (relating to big values of K) are combined into bigger classes (relating to small values of K). These transformations allow us to get some idea about the structure of empirical data. They indicate mutual location of compact groups of points in many-

dimensional space. They also indicate which of these groups are close and which are far from each other. Interpretation of the obtained classes in substantial terms, and the details of their mutual location allow the researcher to construct meaningful models of the phenomenon under consideration [2]-[4].

Different methods of data clustering differ from each other by the way of finding of the optimal partition of the points into K classes. It is literally to say that almost all of them own the same feature: the result of partition into K classes depends on arbitrary chosen initial conditions, which have to be specified to start the partition procedure.

Consequently, to obtain the optimal partition, it is necessary to repeat the procedure many times, each time starting from new initial conditions. Here the situation is close to the one, which we face when founding the global minimum of multiextremal functional. The problems of such a kind exhibit a tendency to become *NP*-complete. This means that for large N a local optimal partition can be found.

Thus, almost all clustering methods based on the local partitioning of objects can be distributed into K classes. Among them there is the well-known and most simple K-means approach, mathematically advanced Super-Paramagnetic Clustering and Maximum Likelihood Clustering, popular in Russia the FOREL-type algorithms and prevailing in the West different variants of Hierarchical Clustering [5].

The general scheme of the FOREL-algorithm is as follows:

1. Specify a value T that is the radius of m-dimensional sphere, which in what follows is used as a threshold for interaction radius between points;
2. Place the center of the sphere with the radius T at an arbitrary input point;
3. Find coordinates of the center of gravity of points that find themselves inside the sphere;
4. Transfer the center of the sphere in the center of gravity and go back to item 3;
5. Far as when going from one to the next iterating the sphere remains in the same place, we suppose that the points inside it constitute a class; we move them away from the set and go back to the item 2.

It is clear that after finite number of steps we obtain a partition of the points into some classes. In each class the distances between points are less than $2T$. However, the result of partition depends on the starting point, where the center of the sphere is situated. Since the step 2 is repeated again and again.

The Hierarchical Clustering is based on a very simple idea too. Given some partition into K classes, it merges the two closest classes into a single one. So, starting from the partition into $K = N$ classes. the algorithm generates a sequence of partitions as K varies from N to 1. The sequence of partitions and their hierarchy can be represented by a dendrogram [4].

In this research work, neural network clustering is used to distinguish or filter different kinds of fault in the case of multiple or mixture in faults. In a synchronous machine there are different kinds of faults. In certain cases more than one fault can occur at the same time. In that case synchronous machines behavior is in response of multiple faults. The output of the machine load angle δ is in response to multiple faults; hence, the behavior of δ can become apparently gibberish as a mixture of more than one response signal. Using numerical simulation or by human observation it is almost impossible to make any sense out of it.

Neural network cluster in this case comes handy in clustering similar kinds of data patterns together to distinguish between different kinds of faults. Once clustered, the pattern can tell us how many signals has been mixed, that is the number of faults overlapping each other and by looking at the clustering weight density pattern map one can even identify the type of faults.

6.3 Implementation of Clustering

6.3.1 Neural network

In this research work a self-organizing network is implemented for clustering the mixed dataset. Self-organizing in networks can learn to detect regularities and correlations in their input and adapt their future responses to that input accordingly. A self-organizing feature maps (SOFMs) algorithm is used. SOFMs learn to classify input vectors according to how they are grouped in the input space. They differ from

competitive layers in that neighboring neurons in the self-organizing map learn to recognize neighboring sections of the input space. Thus, self-organizing maps learn both the distribution (as do competitive layers) and topology of the input vectors they are trained on. Batch training is used. The batch training algorithm is generally much faster than the incremental algorithm, and it is the default algorithm for SOFM training.

6.3.2 Neural network specifications

Choosing an appropriate network with appropriate size, algorithm for training a data set can sometimes be tricky. It depends on complexity of data pattern, size of data set, accuracy and speed one want to train a network, at times experience of the trainer or even intuition. The SOM network in the clustering process has 9 inputs with 20 hidden layers with 20 neurons in each layer. So the weight is mapped in a 20 by 20 weight space. It has to learn 3 patterns with 3 samples in each pattern with 10001 elements in each sample to learn.

Table 6.1. Neural network

Algorithm	Self-organizing network
Training	Self-organizing feature maps (SOFM)
Type	Batch training algorithm

Table 6.2. Neural network specification

Number of Neuron (Input Layer)	9
Number of Neuron (Hidden Layer)	20
Number of Hidden Layers	20
Number of pattern	3
Number of samples in each pattern	3
Number of samples	9
Sample Size	10001

Table 6.3. Neural network specification

Epoch	200
Time	0:20:37

6.3.3 Neural network training conditions

An epoch in neural network is defined by one round of training using all the data set once. In the case of clustering the training process is carried on again and again until the predefined number of epochs is completed. In this case unlike other feed foreward neural network there is no performance curve for this specific algorithm. The neural network training conditions for the first simulation are shown in Table 6.3.

6.4 Simulation and Results

In this section, clustering of mixed signals is presented. The weight map consists of a space of 20 by 20 neurons; that is 400 neurons in total. In the 2D space of 20 by 20 the 9 mixed signals of three distinct patterns are mapped. In Figure 6.1, the mapping for 9 samples are shown in SOFM space. 3 samples from each LOF, DIT and SC are chosen. As it can be seen from the SOFM map in Figure 1 that inputs 2, 3 & 7 has similar weight patterns. On the other hand inputs 1, 5 & 6 has similar weight patterns, and inputs 4, 8 & 9 has similar weight patterns. The first one corresponds to LOF, the second one to DIT and the third one to SC.

SOM neighboring weight distance graph in Figure 6.2 shows three distinct areas – one in the top right, one diagonally in the middle and another in bottom-left each corresponding to each of the patterns. SOM weight positions in Figure 6.3 shows all the weights in their density positioning. This shows visualization of the weight in multiple dimensions. Figure 6.4 shows the total number of hits in the weight by the samples. This shows us the weight density in the weight space.

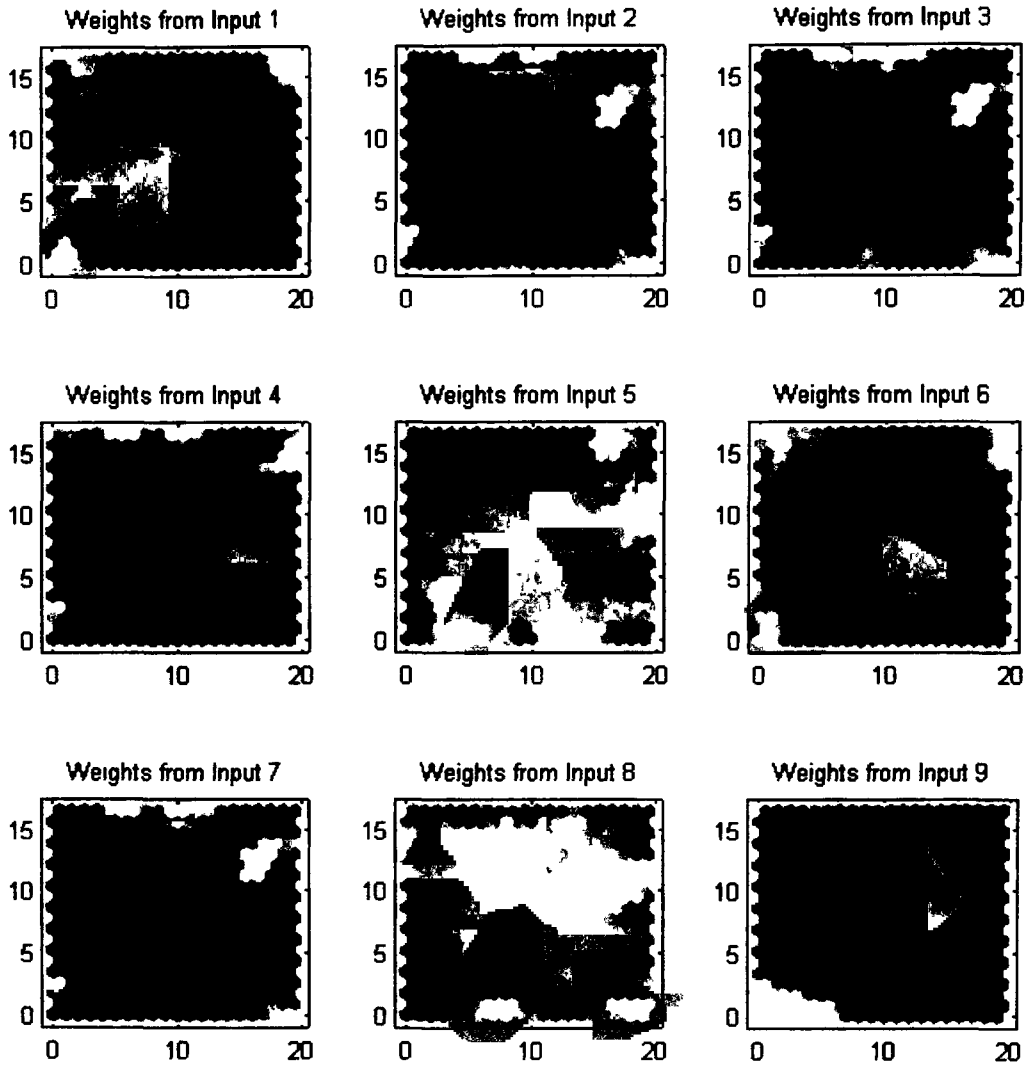


Figure 6.1. SOM weight plane.

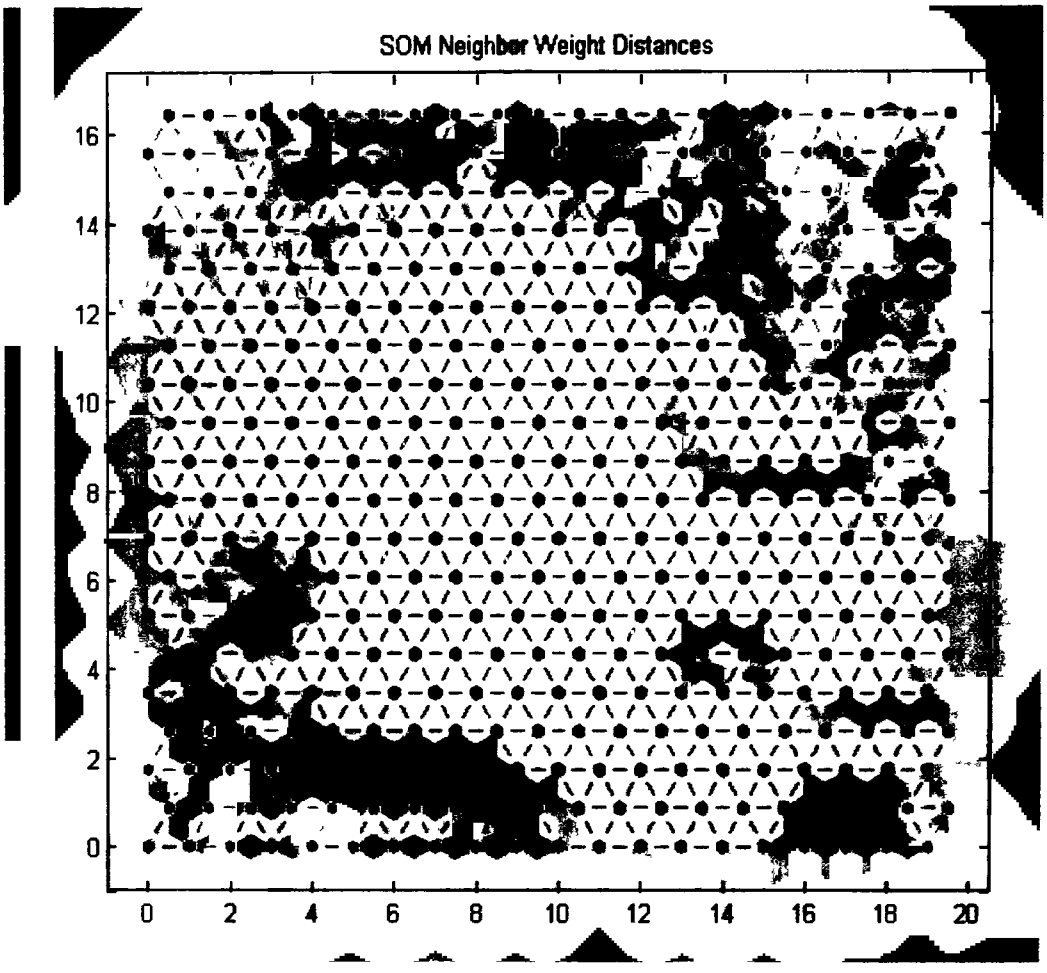


Figure 6.2. SOM neighbor weight distances.

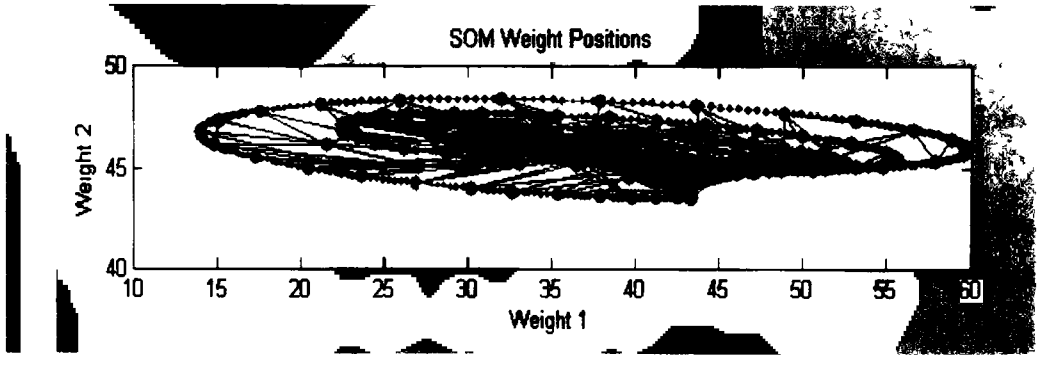


Figure 6.3. SOM weight positions.

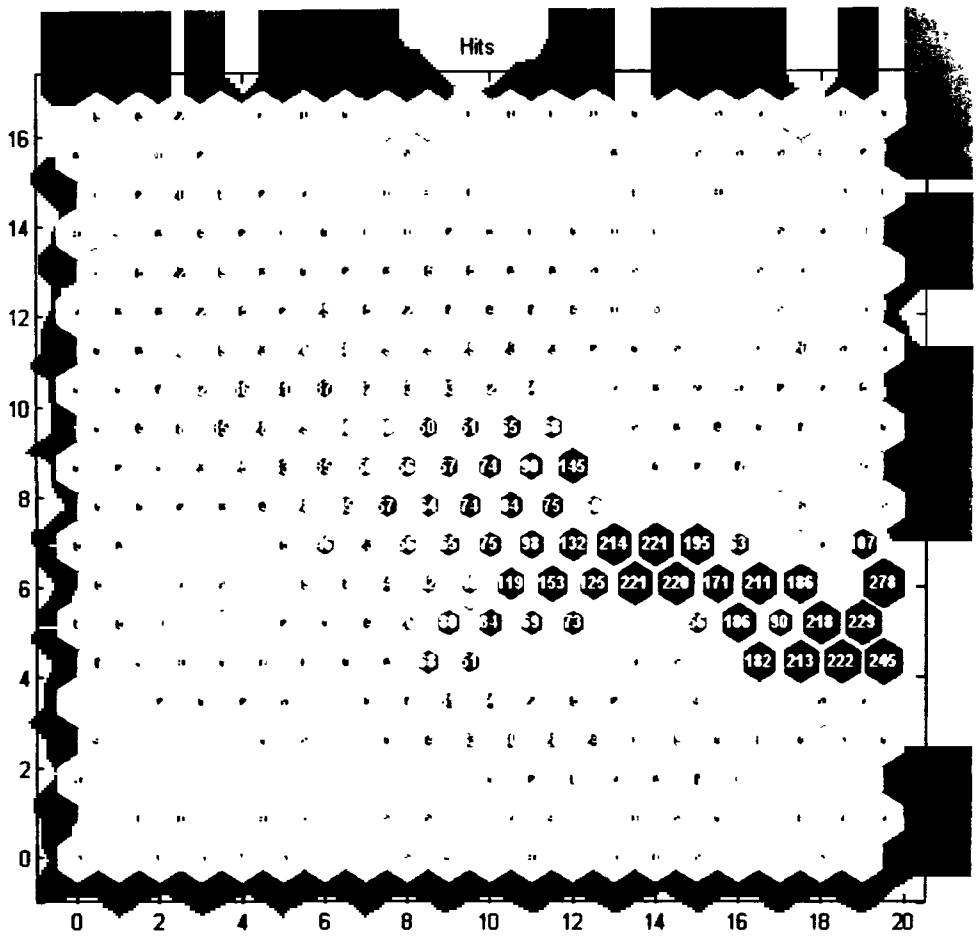


Figure 6.4. SOM weight hits.

6.5 References

[1] B.B. Linda and C.A. Virginia, "Processing data: The survey example (Quantitative applications in the social sciences)" *Sage Publications Inc.*, December 14, 2006.

[2] C.M. van der Walt and E. Barnard, "Data characteristics that determine classifier performance," *SAIEE Africa Research Journal*, vol 98 (3), pp 87-93, September 2007.

[3] C.M. van der Walt, *Data Measures that Characterise Classification Problems*. Master's dissertation, Department of Electrical, Electronic and Computer Engineering, University of Pretoria, South Africa, February 2008.

[4] Z. Zhang, H. Cheng, and S. Zhang, "Approach to SOM based correlation clustering," *CCDC 2008*, pp. 2485 - 2489, 2-4 July, 2008.

[5] M.H. Wang and H.C. Chang, "Novel clustering method for coherency identification using an artificial neural network," *IEEE Transactions on Power Systems*, vol. 9, Issue 4, pp. 2056 2062, Nov. 1994.

7 NEURAL NETWORK PATTERN RECOGNITION

7.1 Introduction

Pattern recognition means classification of data (patterns) based either on a priori knowledge or on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations, defining points in an appropriate multidimensional space. In the case of pattern recognition the pattern is not rigidly specific distinguishing it from pattern matching [1], [2].

Pattern recognition is a sub-topic of machine learning. The idea can be described as taking in raw data and taking an action based on the category of the data. Most research in pattern recognition is associated with methods for supervised learning and unsupervised learning.

7.2 Overview of Pattern Recognition

One of the prime concerns of a pattern recognition system is to identify the data pattern which is to be learned. This is where observations to be classified or described, a feature extraction mechanism that computes numeric or symbolic information from the observations, and a classification or description scheme that does the actual job of classifying or describing observations, relying on the extracted features.

The classification or description scheme is usually based on the availability of a set of patterns that have already been classified or described. This set of patterns is termed the training set, and the resulting learning strategy is characterized as supervised learning. Learning can also be unsupervised, in the sense that the system is not given an a priori labeling of patterns, instead it itself establishes the classes based on the statistical regularities of the patterns.

The classification or description scheme usually uses one of the following approaches: statistical (or decision theoretic) or syntactic (or structural). Statistical pattern recognition is based on statistical characterizations of patterns, assuming that the patterns are generated by a probabilistic system. Syntactical or structural pattern

recognition is based on the structural interrelationships of features. A wide range of algorithms can be applied for pattern recognition, from very simple Bayesian classifiers to much more powerful neural networks [3], [4].

An intriguing problem in pattern recognition is the relationship between the problem to be solved implied by the data to be classified and the performance of various pattern recognition algorithms also known as classifiers. Van der Walt and Barnard investigated very specific artificial data sets to determine conditions under which certain classifiers perform better and worse than others.

Holographic associative memory is another type of pattern matching scheme where a target small patterns can be searched from a large set of learned patterns based on cognitive meta-weight [5]. Pattern recognition is studied in many fields, including psychology, ethnology, cognitive science and computer science. Within medical science, pattern recognition is the basis for computer-aided diagnosis (CAD) systems. CAD describes a procedure that supports the doctor's interpretations and findings. Typical applications are automatic speech recognition, classification of text into several categories e.g. spam/non-spam email messages, the automatic recognition of handwritten postal codes on postal envelopes, or the automatic recognition of images of human faces.

In this research work, pattern recognition has been used to distinguish between different kinds of faults. A synchronous machine operating under a grid can be subjected to different kinds of faults. In this research work three different kinds of faults being discussed. They are loss of field, disturbance in torque and short circuit condition. These are the most common types of faults for synchronous machines. Any of these faults can be of different levels and different time durations.

When a fault occurs in synchronous machine it can be realized by looking at the way the load angle δ is behaving. Under different kinds of faults, δ shows different patterns of behavior. Neural network pattern recognition is used in this research work to distinguish between different kinds of faults. As any of these faults can be of different levels or time duration, once the type of fault is detected synchronous machine pattern recognition is used to identify the level and time duration of the fault.

7.3 Implementation of Pattern Recognition

7.3.1 Pattern recognition between loss of excitation, disturbance in torque and short circuit

In this section, a network was designed to distinguish between loss of field, disturbance in torque and short circuit. This is done by just looking at the pattern of the load angle δ. Details of the implementation are depicted in the next sections.

7.3.1.1 Neural network

A feedforward neural network is used to approximate the input output relationship of the synchronous machine transient conditions. A Scaled Conjugate Gradient backpropagation algorithm is used which is an especial type of backprobagation algorithm. Conjugate gradient algorithms require a line search at each iteration. This line search is computationally expensive, because it requires that the network response to all training inputs be computed several times for each search.

The scaled conjugate gradient algorithm (SCG), developed by Moller, and was designed to avoid the time-consuming line search. This algorithm combines the model-trust region approach (used in the Levenberg-Marquardt algorithm. The routine can require more iteration to converge than the other conjugate gradient algorithms, but the number of computations in each iteration is significantly reduced because no line search is performed. The network basics are described in Table 7.1.

Table 7.1. Neural network.

Algorithm	Backpropagation
Training	Scaled Conjugate Gradient
Type	Gradient Descent
Performance	Mean Squared Error (MSE)
Data Division	Random

Table 7.2. Neural network specifications.

Number of Neuron (Input Layer)	5001
Number of Neuron (Output Layer)	3
Number of Neuron (Hidden Layer)	200
Number of Hidden Layers	1
Sample Size	300 (100%)
Training Sample	270 (90%)
Validation Sample	15 (5%)
Testing Sample	15 (5%)

7.3.1.2 Neural network specifications

Choosing an appropriate network with appropriate size, algorithm for training a dataset can sometimes be tricky. It depends on complexity of data pattern, size of dataset; accuracy and speed one want to train a network, at times experience of the trainer or even intuition. Depending on size of dataset and complexity of the problem a neural network with a single input, single output and single hidden layer is used.

The number of neuron in the input is 5001, output layer is 3 and the number of neuron in the hidden layer is 200. The size of the data points is 300. Of this dataset 90% of the dataset that is 270 points were used as training data. Of the rest 5% of the data that is 15 were used as validation data, which were used to validate how the network is performing while training the network using training dataset. Once the training was done the rest 5% that is 15 data points were used to test the performance of the network. The network specifications are shown in Table 7.2.

7.3.1.3 Neural network training conditions

An epoch in neural network is defined by one round of training using all the dataset once. After training the network for one epoch, the error function is used to calculate the error value once. Until the error value reaches a certain minimum threshold,

the epochs are continued; that is the training process is carried on again and again. The error curves as the number of epoch's increases are also known as performance curve or simply performance. The neural network training conditions for the first simulation are shown in Table 7.3.

Table 7.3. Neural network training conditions.

Epoch	71
Time	0:08:34
Initial Performance	0.464
Final Performance	1 2155e-08
Best Validation Performance	1.2155e-08
Best Validation Performance Epoch	71
Initial Gradient	1.0
Final Gradient	9.7096e-7
Best Gradient	9.7096e-7
Best Gradient Epoch	71
Training MSE	1.21546e-7
Validation MSE	5.66630e-9
Testing MSE	1.11054e-7

7.3.2 Pattern recognition between 20%, 40%, 60%, 80% & 100% loss of excitation

In this section, a network was designed to distinguish between different levels of loss of field. Five different patterns are defined. First pattern is between 0% and 20% LOF, second pattern is between 20% and 40% LOF, third pattern is between 40% and 60% LOF, fourth pattern is between 60% and 80% LOF and the fifth pattern is between 80% and 100% LOF This is done by just looking at the pattern of the load angle. Details of the implementation are depicted in the next sections.

7.3.2.1 Neural network

A feedforward neural network is used to approximate the input output relationship. A Scaled Conjugate Gradient backpropagation algorithm is used The network basics are described in Table 7.4.

7.3.2.2 Neural network specifications

Depending on size of dataset and complexity of the problem a neural network with a single input, single output and single hidden layer is used. The number of neuron in the input 5001, the output layer is 5 and the number of neuron in the hidden layer is 200. The size of the data points is 100. Of this dataset 90% of the dataset that is 90 points were used as training data. Of the rest 5% of the data that is 5 were used as validation data, which were used to validate how the network is performing while training the network using training dataset. Once the training was done the rest 5% that is 5 data points were used to test the performance of the network. The network specifications are shown in Table 7.5 and the training conditions for the simulation are shown in Table 7.6.

Table 7.4. Neural network.

Algorithm	Backpropagation
Training	Scaled Conjugate Gradient
Type	Gradient Descent
Performance	Mean Squared Error (MSE)
Data Division	Random

Table 7.5. Neural network specifications.

Number of Neuron (Input Layer)	5001
Number of Neuron (Output Layer)	5
Number of Neuron (Hidden Layer)	200
Number of Hidden Layers	1
Sample Size	100 (100%)
Training Sample	90 (90%)
Validation Sample	5 (5%)
Testing Sample	5 (5%)

Table 7.6. Neural network training conditions.

Epoch	151
Time	0:11:04
Initial Performance	0.410
Final Performance	0.00898
Best Validation Performance	0.0012871
Best Validation Performance Epoch	145
Initial Gradient	1.0
Final Gradient	0.0129
Best Gradient	0.012934
Best Gradient Epoch	151
Training MSE	8.97618e-3
Validation MSE	1.28705e-3
Testing MSE	1.02214e-2

7.3.3 Pattern recognition between 20%, 40%, 60%, 80% & 100% loss in torque

In this section, a network was designed to distinguish between different levels of disturbance in torque. Five different patterns are defined. First pattern is between 0% and 20% DIT, second pattern is between 20% and 40% DIT, third pattern is between 40% and 60% DIT, fourth pattern is between 60% and 80% DIT and the fifth pattern is between 80% and 100% DIT. This is done by just looking at the pattern of the load angle. Details of the implementation are depicted in the next sections.

7.3.3.1 Neural network

A feedforward neural network is used to approximate the input output relationship. A Scaled Conjugate Gradient backpropagation algorithm is used. The network basics are described in Table 7.7

7.3.3.2 Neural network specifications

Depending on size of dataset and complexity of the problem a neural network with a single input, single output and single hidden layer is used. The number of neuron in the input 5001, the output layer is 5 and the number of neuron in the hidden layer is 300. The size of the data points is 100. Of this dataset 90% of the dataset that is 90 points were used as training data. Of the rest 5% of the data that is 5 were used as validation data, which were used to validate how the network is performing while training the network using training dataset. Once the training was done the rest 5% that is 5 data points were used to test the performance of the network. The network specifications are shown in Table 7.8 and the training conditions for the simulation are shown in Table 7.9.

Table 7.7. Neural network.

Algorithm	Backpropagation
Training	Scaled Conjugate Gradient
Type	Gradient Descent
Performance	Mean Squared Error (MSE)
Data Division	Random

Table 7.8. Neural network specifications.

Number of Neuron (Input Layer)	5001
Number of Neuron (Output Layer)	5
Number of Neuron (Hidden Layer)	300
Number of Hidden Layers	1
Sample Size	100 (100%)
Training Sample	90 (90%)
Validation Sample	5 (5%)
Testing Sample	5 (5%)

Table 7.9. Neural network training conditions.

Epoch	93
Time	0:06:49
Initial Performance	0.253
Final Performance	1.7241e-9
Best Validation Performance	1.7241e-9
Best Validation Performance Epoch	93
Initial Gradient	1.0
Final Gradient	9.1239e-7
Best Gradient	9.1239e-7
Best Gradient Epoch	93
Training MSE	3.01629e-8
Validation MSE	1.7407e-9
Testing MSE	3.02911e-2

7.3.4 Pattern recognition between 0.05 s, 0.10 s, 0.15 s, 0.20 s & 0.212 s SC

In this section, a network was designed to distinguish between different durations of short circuit. Five different patterns are defined. First pattern is between 0.05 s and 0.05 s of SC, second pattern is between 0.05 s and 0.05 s of SC, third pattern is between 0.05 s and 0.05 s of SC, fourth pattern is between 0.05 s and 0.05 s of SC, and the fifth pattern is between 0.05 s and 0.05 s of SC. This is done by just looking at the pattern of the load angle. Details of the implementation are depicted in the next sections.

7.3.4.1 Neural network

A feedforward neural network is used to approximate the input output relationship. A Scaled Conjugate Gradient backpropagation algorithm is used. The network basics are described in Table 7.10.

7.3.4.2 Neural network specifications

Depending on size of dataset and complexity of the problem a neural network with a single input, single output and single hidden layer is used. The number of neuron in the input 5001, the output layer is 5 and the number of neuron in the hidden layer is 300. The size of the data points is 100. Of this dataset 90% of the dataset that is 90 points were used as training data. Of the rest 5% of the data that is 5 were used as validation data, which were used to validate how the network is performing while training the network using training dataset. Once the training was done the rest 5% that is 5 data points were used to test the performance of the network. The network specifications are shown in Table 7.11 and training conditions for the simulation are shown in Table 7.12

Table 7.10. Neural network.

Algorithm	Backpropagation
Training	Scaled Conjugate Gradient
Type	Gradient Descent
Performance	Mean Squared Error (MSE)
Data Division	Random

Table 7.11. Neural network specifications.

Number of Neuron (Input Layer)	5001
Number of Neuron (Output Layer)	5
Number of Neuron (Hidden Layer)	300
Number of Hidden Layers	1
Sample Size	100 (100%)
Training Sample	90 (90%)
Validation Sample	5 (5%)
Testing Sample	5 (5%)

Table 7.12. Neural network training conditions.

Epoch	30
Time	0:03:11
Initial Performance	0.443
Final Performance	0.0116
Best Validation Performance	0.0012871
Best Validation Performance Epoch	24
Initial Gradient	1.0
Final Gradient	0.0751
Best Gradient	0.075124
Best Gradient Epoch	30
Training MSE	0.0125
Validation MSE	0.0159
Testing MSE	0.0211

7.4 Simulation and Results

7.4.1 Pattern recognition between loss of excitation, disturbance in torque and short circuit

In Figure 7.1, there are four matrixes representing the confusion of the neural network training process. In the matrixes red represents failure, green represents success, grey represents total samples in each pattern and the blue the cumulative total. The first one on the top-left is the training matrix, top-right is the validation, bottom-left is the test matrix and the bottom-right is the cumulative matrix. As there are three distinct patterns, diagonally there are three element; one for each pattern.

From Figure 7.1 we can see that in all of the cases the network learned the patterns and identified them with 100% success rate. Figure 7.2 is the performance matrix where the best validation performance was recorded at epoch 71 with minimum error 1.2155×10^{-7} Figure 7.3 shows the gradient and the validation check throughout the training process. The total number of epochs needed to train the network is 71.

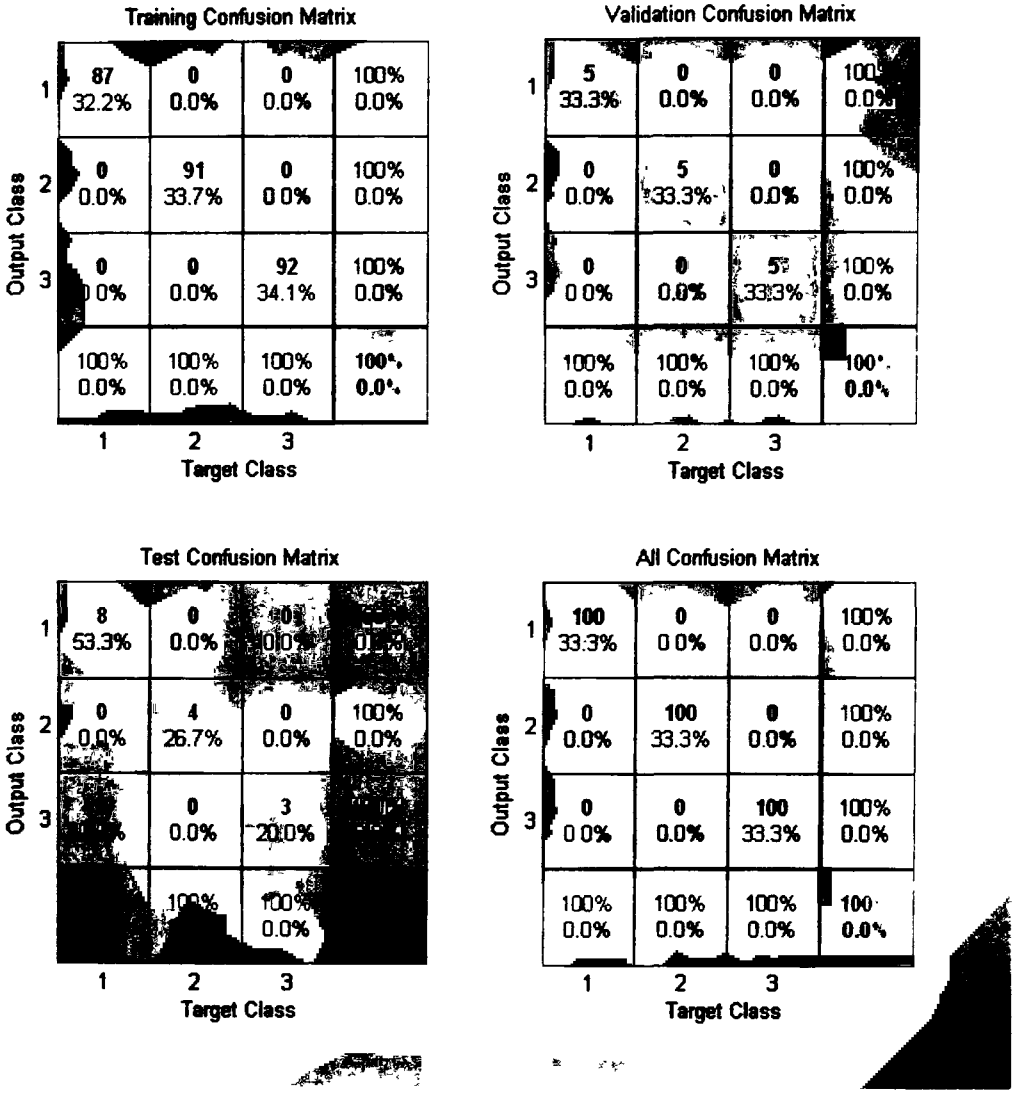


Figure 7.1. Confusion matrix.

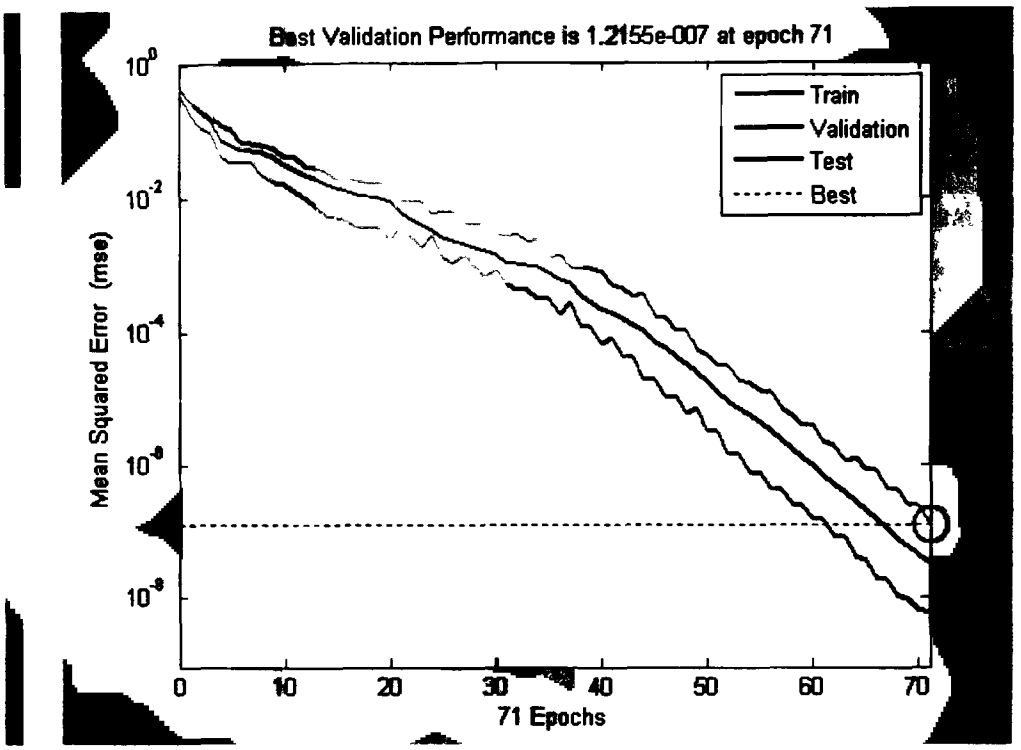


Figure 7.2. Error curve.

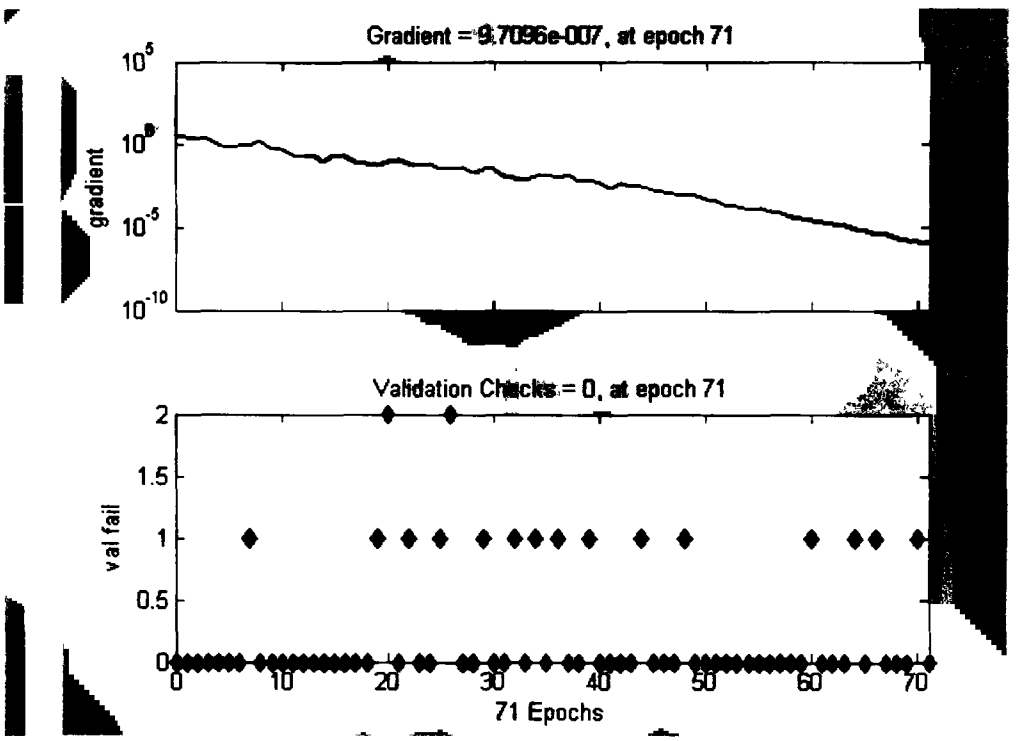


Figure 7.3. Performance graph.

7.4.2 Pattern recognition between 20%, 40%, 60%, 80% & 100% loss of excitation

In Figure 7.4, there are four matrixes representing the confutation of the neural network training process. In the matrixes red represents failure, green represents success, grey represents total samples in each pattern and the blue the cumulative total. The first one on the top-left is the training matrix, top-right is the validation, bottom-left is the test matrix and the bottom-right is the cumulative matrix. As there are five distinct patterns, diagonally there are five element; one for each pattern.

From Figure 7.4, we can see that in all of the cases the network learned the patterns and identified them with 100% success rate. Figure 7.5 is the performance matrix where the best validation performance was recorded at epoch 145 with minimum error 1.2871×10^{-7} Figure 7.6 shows the gradient and the validation check throughout the training process. The total number of epochs needed to train the network is 151.

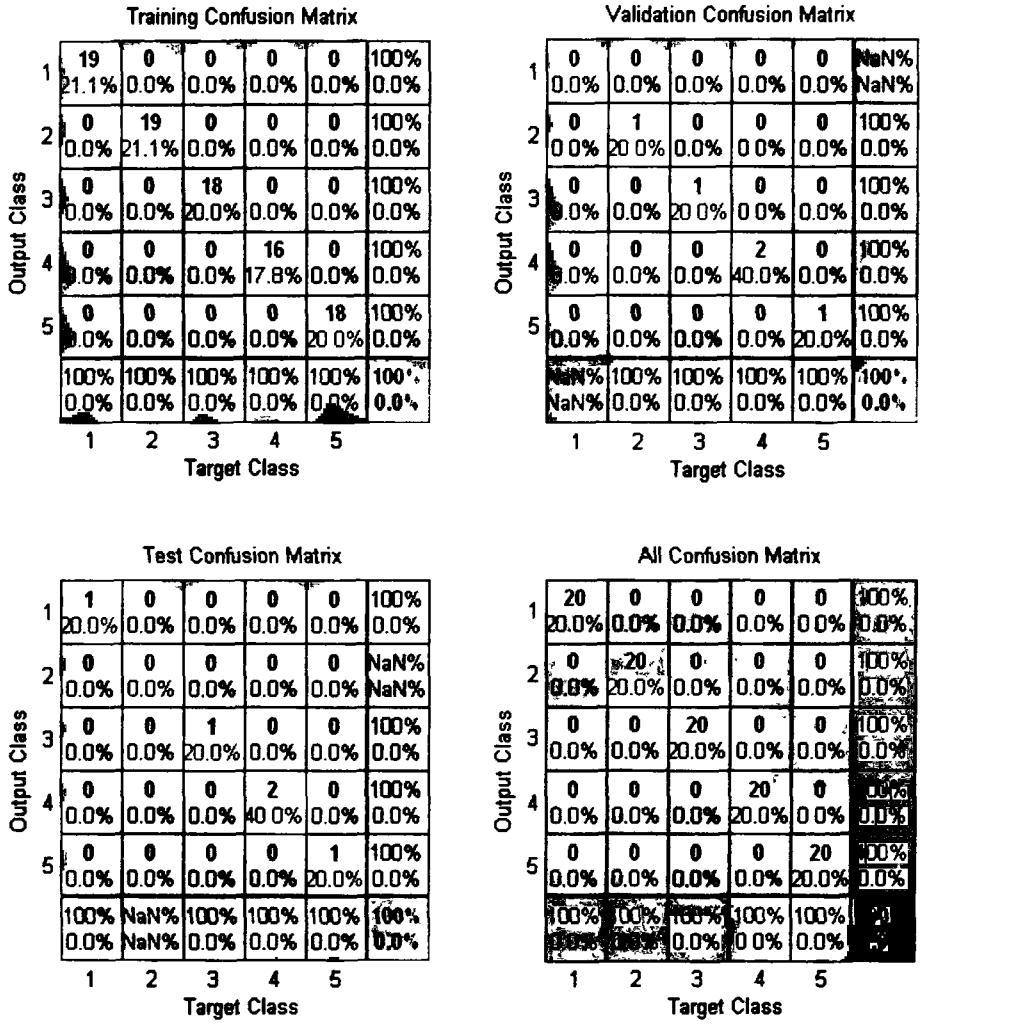


Figure 7.4. Confusion matrix.

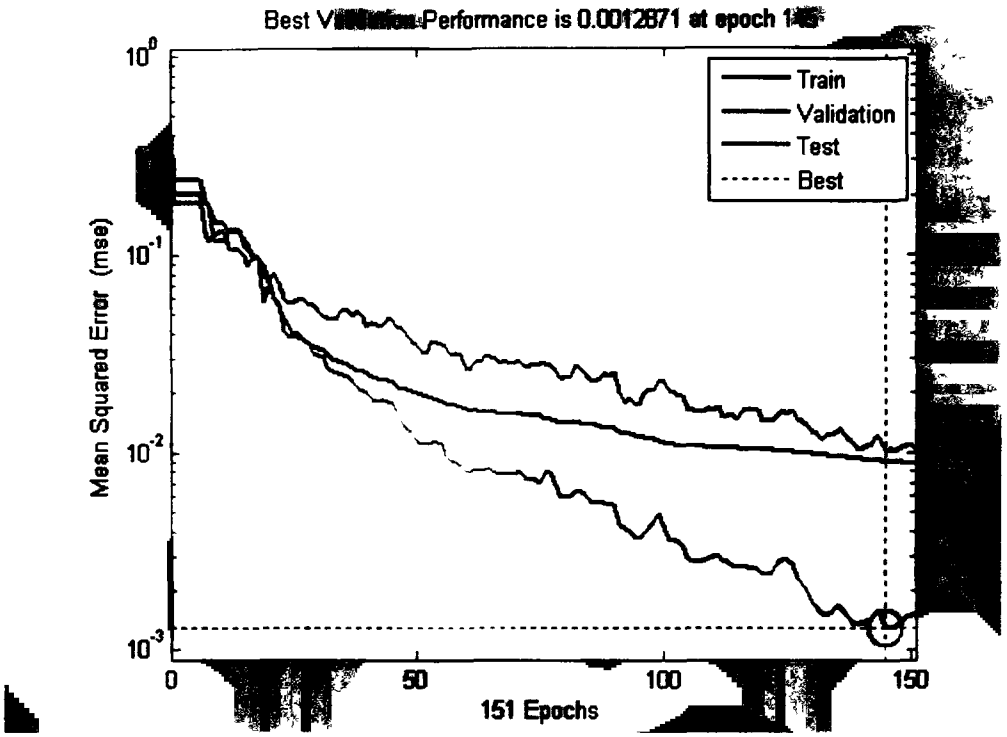


Figure 7.5. Error curve.

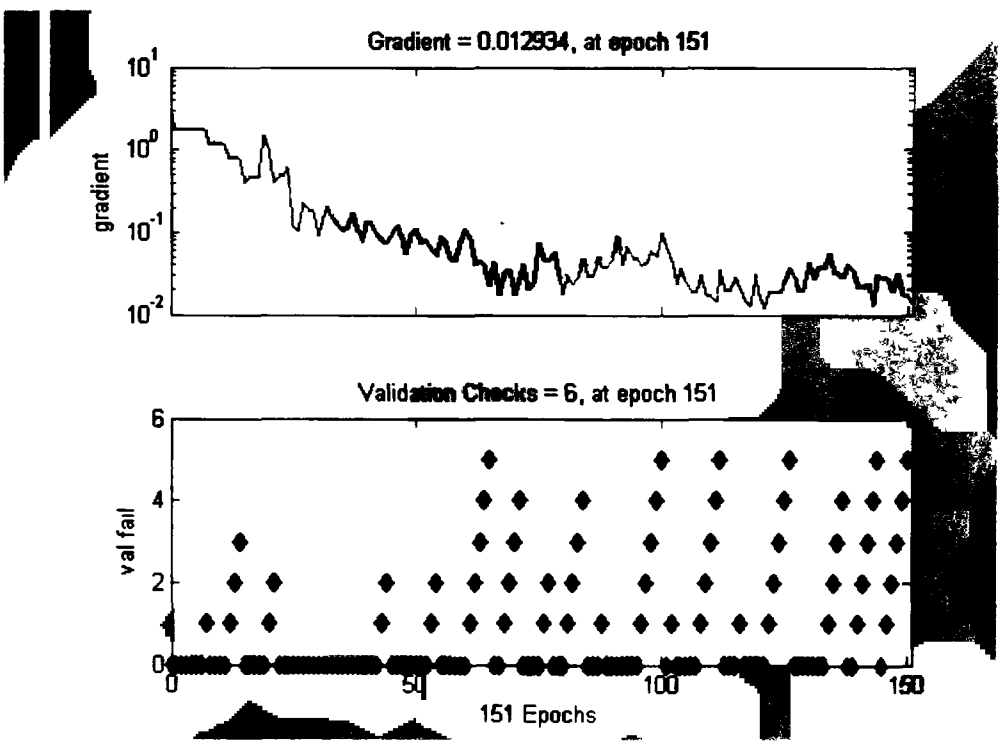


Figure 7.6. Performance graph.

7.4.3 Pattern recognition between 20%, 40%, 60%, 80% & 100% loss in torque

In Figure 7.7, there are four matrixes representing the confutation of the neural network training process. In the matrixes red represents failure, green represents success, grey represents total samples in each pattern and the blue the cumulative total. The first one on the top-left is the training matrix, top-right is the validation, bottom-left is the test matrix and the bottom-right is the cumulative matrix. As there are five distinct patterns, diagonally there are five element; one for each pattern.

From Figure 7.7 we can see that in all of the cases the network learned the patterns and identified them with 100% success rate. Figure 7.8 is the performance matrix where the best validation performance was recorded at epoch 24 with minimum error 4.895×10^{-7} Figure 7.9 shows the gradient and the validation check throughout the training process. The total number of epochs needed to train the network is 30.

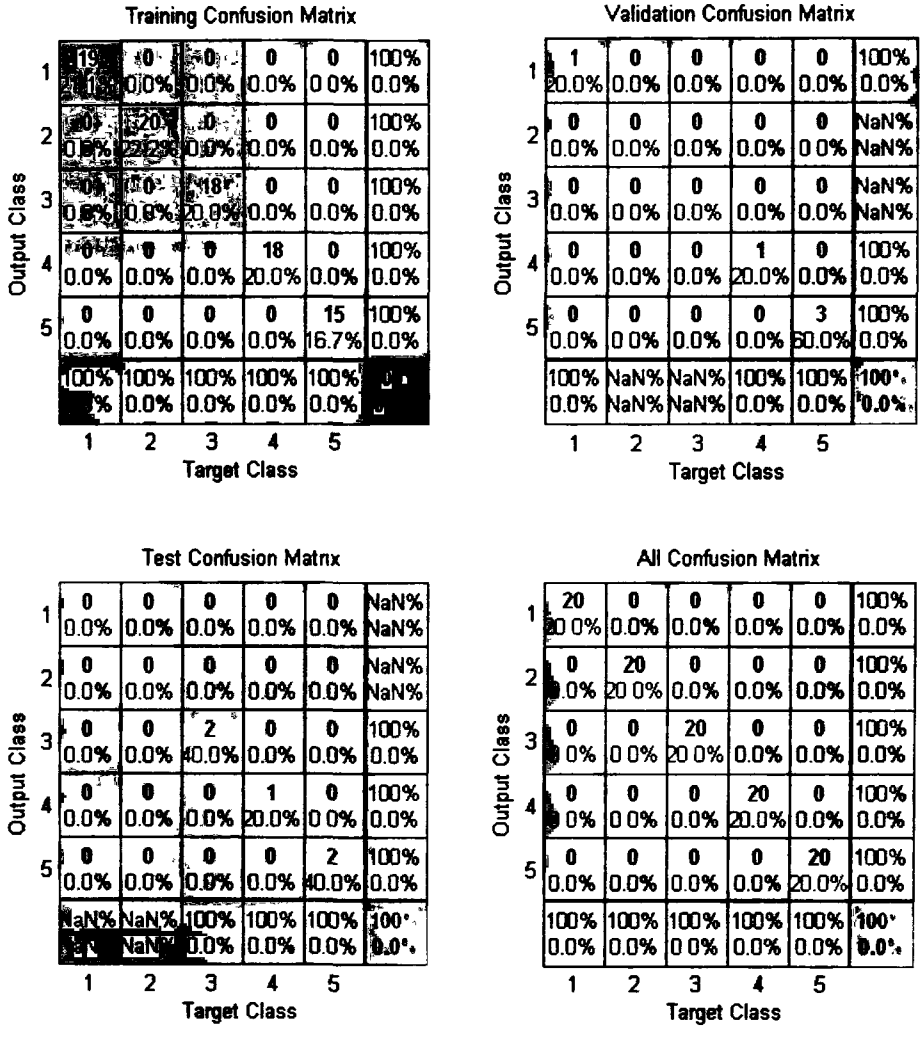


Figure 7.7. Confusion matrix.

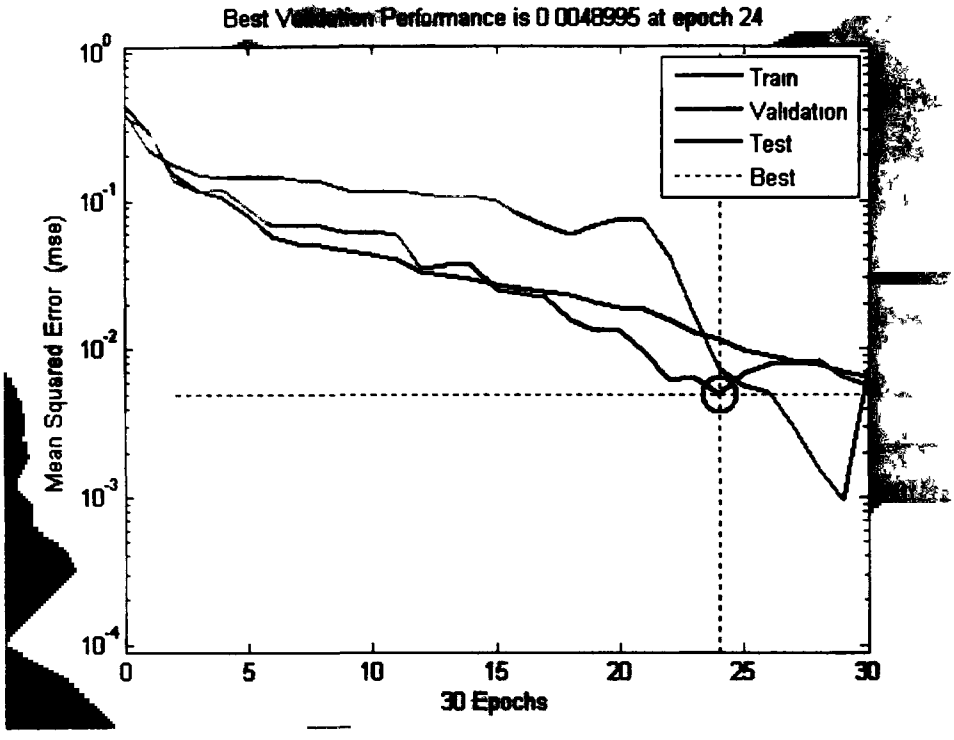


Figure 7.8. Error curve.

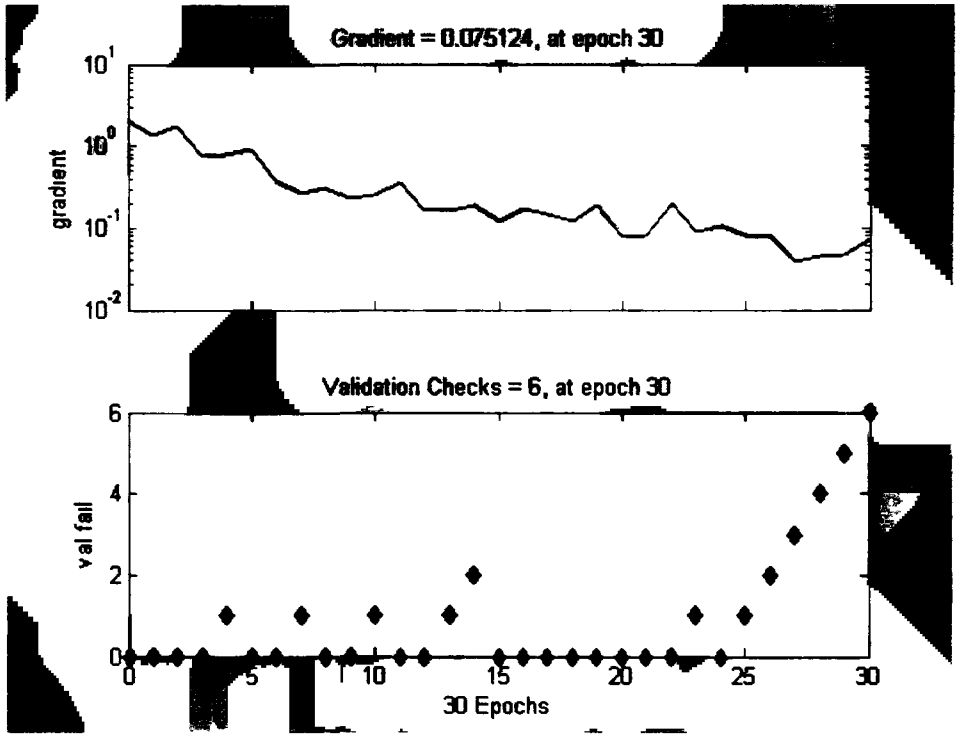


Figure 7.9. Performance graph.

7.4.4 Pattern recognition between 0.05 s, 0.10 s, 0.15 s, 0.20 s & 0.212 s SC

In Figure 7.10, there are four matrixes representing the confutation of the neural network training process. In the matrixes red represents failure, green represents success, grey represents total samples in each pattern and the blue the cumulative total. The first one on the top-left is the training matrix, top-right is the validation, bottom-left is the test matrix and the bottom-right is the cumulative matrix. As there are five distinct patterns, diagonally there are five element; one for each pattern.

From Figure 7.11 we can see that in all of the cases the network learned the patterns and identified them with 100% success rate. Figure 7.12 is the performance matrix where the best validation performance was recorded at epoch 93 with minimum error 1.7241×10^{-7} Figure 7.13 shows the gradient and the validation check throughout the training process. The total number of epochs needed to train the network is 93.

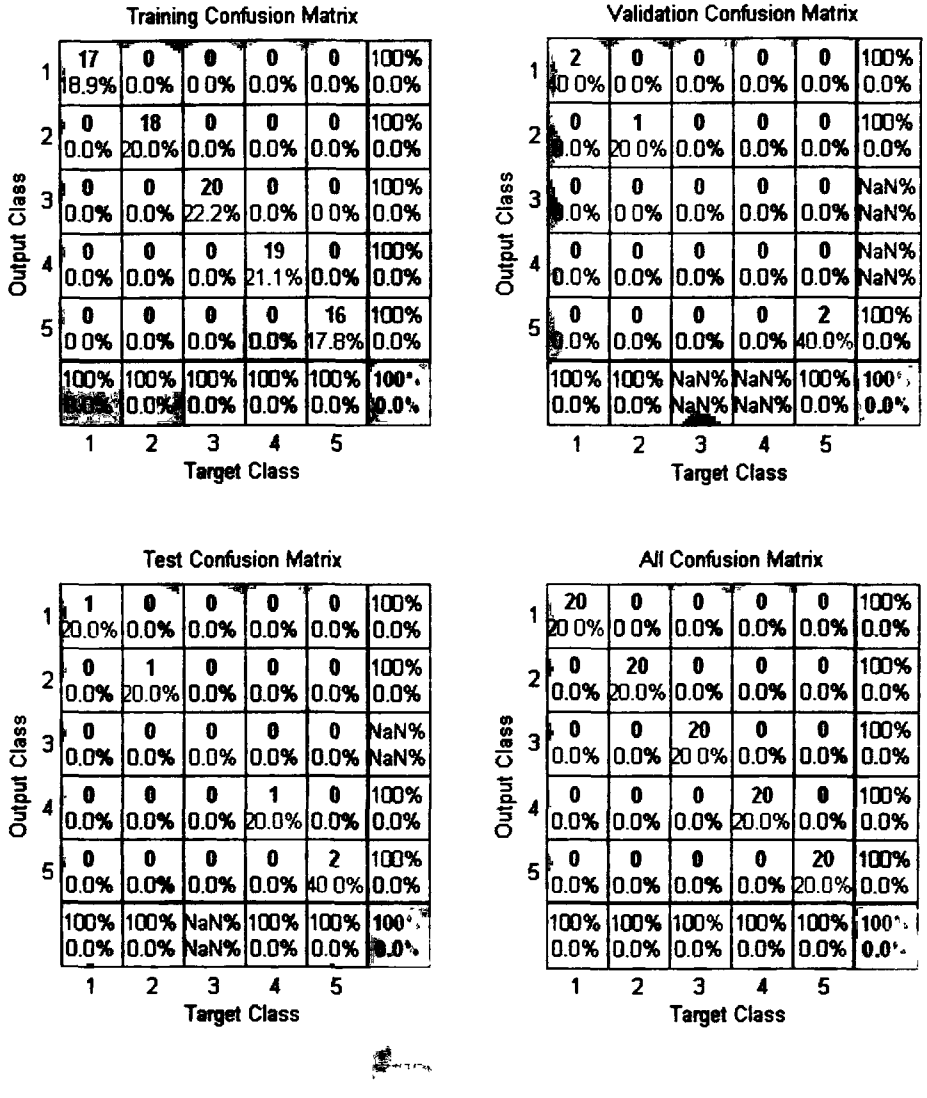


Figure 7.10. Confusion matrix.

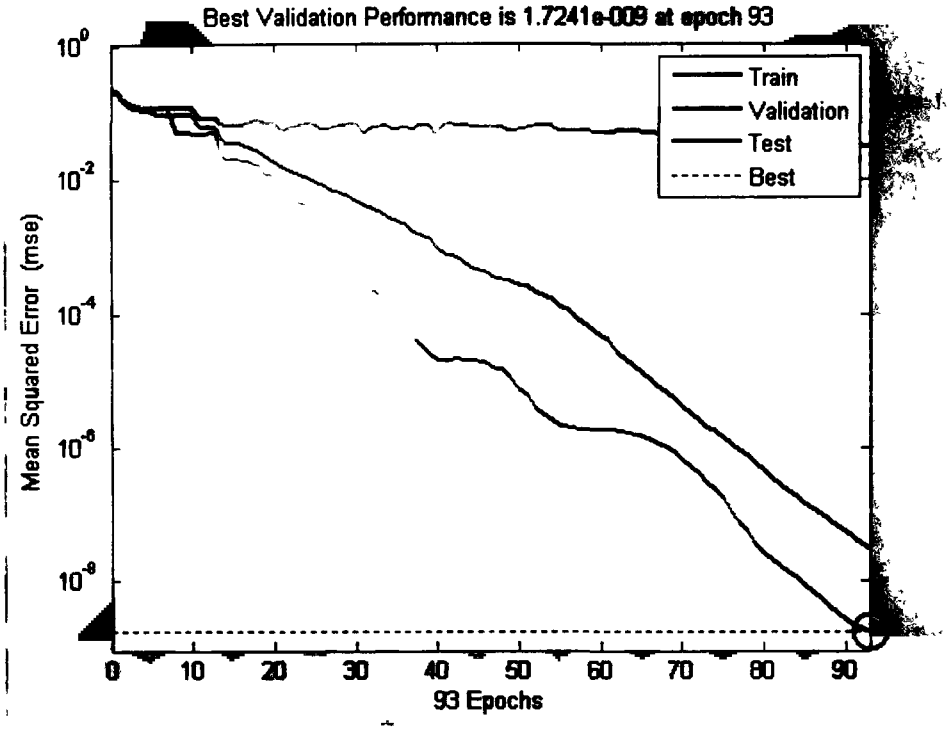


Figure 7.11. Error curve.

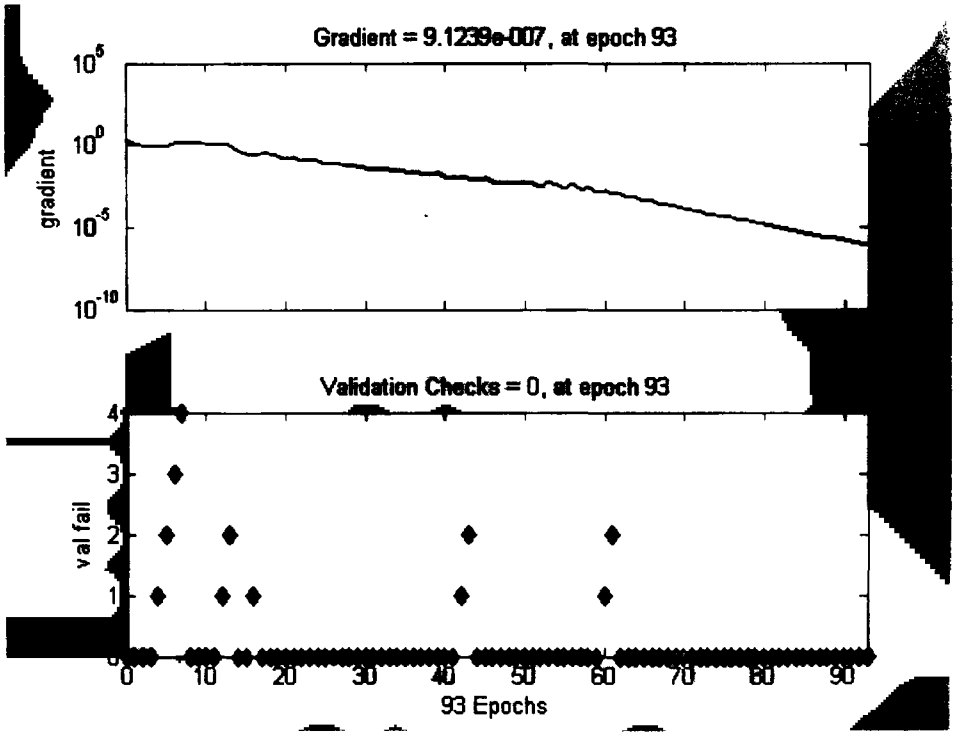


Figure 7.12. Performance graph.

7.5 References

[1] B.B. Nasution and A.I. Khan, "A Hhierarchical graph neuron scheme for real-time pattern recognition," *IEEE Transactions on Neural Networks*, vol. 19(2), 212-229, Feb. 2008.

[2] P.M. Bhagat, *Pattern Recognition in Industry*, Elsevier, 2005.

[3] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford: Oxford University Press, 1995.

[4] R.O. Duda, P.E. Hart, D.G. Stork, *Pattern classification (2^{nd} edition)*, Wiley, 2001.

[5] B.D. Ripley, *Pattern Recognition and Neural Networks*, Cambridge, 1996.

8 CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

In this research work, dynamic analysis of synchronous machine was done using neural network based characterization, clustering and pattern recognition. A synchronous machine model was simulated numerically based on a mathematical model. The developed model was scrutinized for transient conditions under different kind of faults – LOF, DIT & SC. The model was also scrutinized for stability stipulations. Based on the model a neural network model of synchronous machine was developed using neural network based characterization. In the case of multiple or mixture of different kinds of faults, neural network based clustering was used to distinguish and identify specific fault conditions. In the case of unknown fault responses, neural network based pattern recognition was used to identify different kinds of faults and their fault level. In the end, the finding shows that:

- Neural network based characterization can be used to simulate a synchronous machine model given its input output relationships.
- The model can be trained to approximate different transient conditions; such as – loss of field, disturbance in torque and short circuit conditions.
- Neural network clustering can be used to filter and distinguish between different kinds of faults by looking at the behaviour of the load angle.
- By observing the weight distribution pattern of the SOM space similar kinds of faults can be identified.
- Neural network pattern identification can be used to identify and specify unknown fault patterns.
- Once the faults are identified, neural network pattern identification can be used to recognize and indicate the level or time duration of the fault.

8.2 Future Work

Neural network is a very powerful tool with immense potential and diverse strengths. Use of neural network in synchronous machine dynamic analysis can go a long way. Some of the potential research areas with possibilities are:

- Characterization, clustering and pattern recognition of synchronous machine using different network topologies and algorithm; their performance comparisons.
- Neural network based future fault prediction of synchronous machine under different disturbances.
- Neural network based protection system for synchronous machine stability control.
- Online fault filtering of synchronous machine using neural network.
- Neural fuzzy controller based protection system for synchronous machine stability control.

LIST OF PUBLICATION

[1] R. Mazhar, H. Tashakori, and N. Kar, "Investigation of performance analysis of a synchronous generator under loss of excitation," *IREE Journal*, April, 2008.

VITA AUCTORIS

Name	Rashed Mohammed Mazhar
Place of Birth	Dhaka, Bangladesh
Year of Birth	1985
Education	*University of Windsor, Windsor, Ontario* 2007 -2009 M.A.Sc.
	University of Windsor, Windsor, Ontario 2003 2006 B.A.Sc.