

INVESTIGATION ON COMPLEX VARIABLE BASED BACKPROPAGATION ALGORITHM AND APPLICATIONS

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

by

A. Prashanth



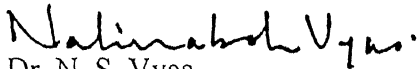
to the

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

March, 2003

CERTIFICATE

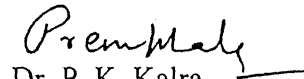
It is certified that the work contained in the thesis "INVESTIGATION ON COMPLEX VARIBALE BASED BACKPROPAGATION ALGORITHM AND APPLICATIONS", by A Prashanth, has been carried out under our supervision and that this work has not been submitted elsewhere for a degree.


Dr. N. S. Vyas

Professor

Department of Mechanical Engineering

I.I.T. Kanpur

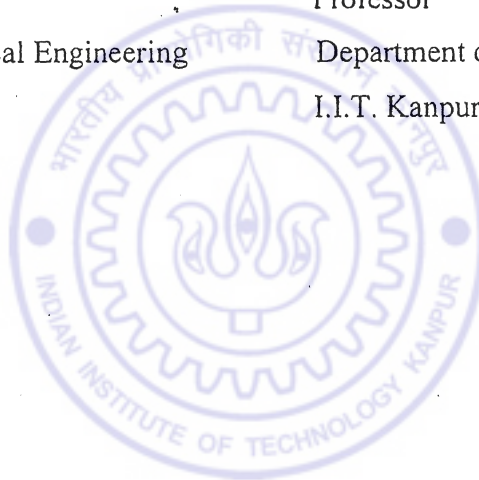

Dr. P. K. Kalra

Professor

Department of Electrical Engineering

I.I.T. Kanpur

March, 2003



26 OCT 2004

दुसरोत्तम काशीनाथ केलकर पुस्तकालय
भारतीय प्रौद्योगिकी संस्थान कानपुर
अवधि क्र० A...149338.....



Acknowledgement

I wish to record my grateful thanks to both my thesis advisors Dr. N. S. Vyas and Dr. P. K. Kalra for affording me freedom that my mind always needed. They always let me take my own direction and never imposed a thing on me. At times when my work went slow or I groped for too long to take decisions, they had always been there to give me suggestions and pass some interesting ideas on. I've realized after much experimentation with my own self that the task that one embarks on should be taken through to the very end clearing all the hurdles en route and shouldn't balk when the difficulty encountered appears arduous. After all, the hurdle that appeared leviathan might turn out trivial on a closer inspection. It's important that one develop a faculty to construct ideas and pulverize the hurdles as they crop up, make way thereby and surge ahead. I've come to realize that it's the successful ventures steered with an ideas based groundwork that makes an expert in the field.

There are many that I must thank for if it were not for them, I wouldn't have learnt as much. Foremost, I must thank Gopal Sharma for all the discussions I had with him over a spectrum of topics. We discussed Mathematics, Physics, Computer Science, Neural Networks, Astronomy, Chess among many others. I gained much from this interaction for it not only helped me improve my understanding but also upgrade my knowledge as I referred the web or textbooks as the discussions progressed. I shall always remember the last discussion we had on Huygens' principle not to mention the altercation the principle led us into.

I thank all my friends in the Mathematics department: Tony, Anjan, Srinivas, Chand, Challa, Sajid, Srinadh and Lamba. I gained much from the weekly discussions we organized and religiously followed for over two years. In the process, we covered many topics from Rudin's books, Algebraic Geometry and Analysis. Thanks to the many courses I audited in the Mathematics department, I've learnt how to think from a fundamental viewpoint. I've understood as the discussions progressed that a strong foundation in mathematics imparts enormous force to the argument one puts forth.

I thank Madhav for all the discussions I had with him. Although he was primarily into Fuzzy Logic and explored Path-Planning in Robotics, we had many topics that we could always cling on to and discuss for hours. There were instances when the discussion became an altercation but no hostility ever stood between Madhav and I. Some important points got clarified during these analyses. I recall the discussion on majorizing sequence and application of Cauchy-Schwartz inequality by considering a dummy sequence that prepares the platform for an interesting situation in a Path-Planning problem.

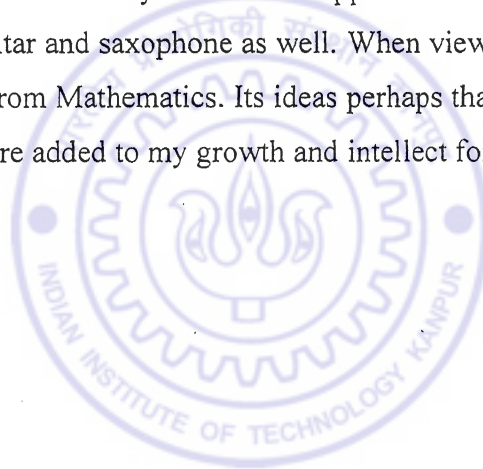
I also thank Sanat, Ashwani, HK, Animesh, GSK, Puneet, Raman, Sinha, kshitij, Deepak and Saurabh for making my stay at IITK memorable. I discussed topics in CAD with GSK, Sanat and Puneet while Animesh helped me with some fundamentals of Vibration and Dynamics. When I first began working on Complex-valued Neural Networks, Sidana had always been there to discuss various problems as they cropped up as we together coded varied algorithms in MATLAB. Later when Sreeram and Sushil joined us and I interacted with them, my outlook got widened as I closely followed the variety problems they worked on using MATLAB and JAVA based programs. I've also closely followed the work Padhy, Rao, Himanshu and Ankur did on the Fault Simulator running LabVIEW and MATLAB based panels. At one stage, Sunil Pandey and I embarked on a music related project of collecting signals from various musical instruments and designing neural networks to differentiate between their timbres.

Out of my own curiosity, I went ahead exploring some paths that often flashed on my mind, like FEM models for vibrating air-columns of musical instruments for instance. I always loved to perform Gedanken experiments on computers and network neighborhood. I discovered MAPLE as I went ahead with my excavations. I found MAPLE a truly wonderful Computer Algebra System. I explored the commands list of the package and with the aid of a book on how to model chaos problems using MAPLE, learnt the programming basics of the package.

I thank all my friends who have been with me at all times. I discussed several technical, practical and computational problems with a group of inmates who have been allotted place upstairs as there wasn't enough space for all the new computers in the office ground floor: Dinesh and Vipin worked tirelessly on a Power Systems Project, running MATLAB programs with whom I

discussed MATLAB related problems and there was Jaideep on the Indian Oil Project who furthered the Sequenced Data Model we proposed to solve a peculiar problem the company in question posed. I've often shared a lunch with Pushpendra and Sunil. Last but not the least, I am grateful to my parents who have always been with me and supported my ambitions.

I have always been an ardent fan of Sir Isaac Newton. His genius manifested in the Fundamental Laws of Motion that always inspired me. An ideas based study is the one that would always rise to the pinnacle and ideas substantiated by hard-work open up promising new avenues as one works on. This was the realization that accrued on me perhaps that urged me to adopt the approach in all the other topics I pursued in the last few years. I studied violin to some depth employing my own ideas and developing them, referring to the web when some puzzling questions crossed my path. I found my ideas based approach effective when I discovered that it worked well with flute, guitar and saxophone as well. When viewed from the ideas angle, Music didn't seem any different from Mathematics. Its ideas perhaps that binds all knowledge into one. All the interaction I had here added to my growth and intellect for I have become a refined and a better human being.



Contents

| | |
|--|----|
| 1. Introduction | |
| 1.1 The Neural Network | 1 |
| 1.2 Artificial Neural Networks and Complex valued Neural networks | 3 |
| 1.3 Updating Complex-valued weights | 5 |
| 1.4 A Few Remarks | 6 |
| 1.5 ANN versus CNN | 6 |
| 1.6 Organization of the Thesis | 7 |
| 2. Literature Survey | |
| 2.1 Research on ANN | 10 |
| 2.2 Complex variable based Neural Networks | 14 |
| 3. Investigation of Error Functions | |
| 3.1 Why vary Error Function | 16 |
| 3.2 Real Error Functions | 17 |
| 3.2.1 Absolute Error Function | 17 |
| 3.2.2 Andrew Error Function | 18 |
| 3.2.3 Bipolar Hyperbolic Squared Error Function | 18 |
| 3.2.4 Cauchy Error Function | 18 |
| 3.2.5 Fair Error Function | 18 |
| 3.2.6 Fourth Power Error Function | 19 |
| 3.2.7 Geman McClure Error Function | 19 |
| 3.2.8 Huber Error Function | 19 |
| 3.2.9 Hyperbolic Squared Error Function | 19 |
| 3.2.10 Log-Cosh Error Function | 20 |
| 3.2.11 Logarithmic Error Function | 20 |
| 3.2.12 Mean-Median Error Function | 20 |
| 3.2.13 Minkowski Error Function | 20 |
| 3.2.14 Quadratic Error Function | 20 |
| 3.2.15 Sinh Error Function | 21 |
| 3.2.16 Tukey Error Function | 21 |
| 3.2.17 Welsch Error Function | 21 |
| 3.3 Complex Error Function | |
| 3.3.1 Complex Absolute Error Function | 22 |
| 3.3.2 Complex Andrew Error Function | 22 |

| | | |
|---------|--|----|
| 3.3.3 | Complex Bipolar Hyperbolic Squared Error Function | 22 |
| 3.3.4 | Complex Cauchy Error Function | 23 |
| 3.3.5 | Complex Fair Error Function | 23 |
| 3.3.6 | Complex Fourth Power Error Function | 23 |
| 3.3.7 | Complex Geman McClure Error Function | 23 |
| 3.3.8 | Complex Huber Error Function | 23 |
| 3.3.9 | Complex Hyperbolic Squared Error Function | 24 |
| 3.3.10 | Complex Log-Cosh Error Function | 24 |
| 3.3.11 | Logarithmic Error Function | 24 |
| 3.3.12 | Complex Mean-Median Error Function | 25 |
| 3.3.13 | Complex Minkowski Error Function | 25 |
| 3.3.14 | Complex Quadratic Error Function | 25 |
| 3.3.15 | Complex Sinh Error Function | 25 |
| 3.3.16 | Complex Tukey Error Function | 25 |
| 3.3.17 | Complex Welsch Error Function | 26 |
| 3.4 | Properties of Error Functions | 35 |
| 3.5 | Benchmark Problems | 41 |
| 3.5.1 | The XOR Problem | 42 |
| 3.5.2 | The N-Parity Problem | 43 |
| 3.5.3 | The Coding-Decoding Problem | 53 |
| 3.5.4 | The Sin(x)Sin(y) Problem | 55 |
| 3.5.5 | The Two-Spirals Problem | 70 |
| 3.6 | Conclusion | 76 |
| 4. | Investigation of Activation Functions | 78 |
| 4.1 | The Complex Activation Function | 78 |
| 4.2 | Study of Complex Activation Functions | 82 |
| 4.2.1 | Liouville Theorem | 82 |
| 4.2.2 | Complex Activation Functions and their Derivatives | 82 |
| 4.2.2.1 | Nitta Activation and Derivatives | 83 |
| 4.2.2.2 | Haykin Activation Function | 84 |
| 4.2.2.3 | Haykin Activation and Singularities | 86 |
| 4.2.2.4 | Georgiou Activation | 87 |

| | | |
|----|---|-----|
| 5. | Application to Mapping Problems | 94 |
| | 5.1 Mapping Properties of Neural Networks | 94 |
| | 5.2 CNN Applied to Mapping Problems | 95 |
| | 5.2.1 Bilinear Transformation | 95 |
| | 5.2.2 The Polynomial Map | 101 |
| | 5.2.3 The Similarity Transformation | 116 |
| | 5.2.4 Complex Exponential Map | 122 |
| | 5.2.5 Exponential Map | 131 |
| | 5.2.6 Mapping $w = \sin(z_1)\sin(z_2)$ using CNN | 136 |
| | 5.3 Conclusion | 141 |
| 6. | Application to Surface Classification Problem | 143 |
| | 6.1 Introduction | 143 |
| | 6.2 Point Clouds in Practical Application | 143 |
| | 6.3 Training Surfaces | 148 |
| | 6.4 Network Testing | 159 |
| | 6.5 Classification Errors and Surface Signatures | 161 |
| | 6.6 Network Robustness | 166 |
| | 6.7 Order Determination of Algebraic Surfaces | 166 |
| | 6.8 BPA based on Different Error Functions | 169 |
| | 6.8.1 Classification result | 172 |
| | 6.9 CVBP based on Different Error Functions | 172 |
| | 6.9.1 The Complex variable based Algorithm | 172 |
| | 6.10 Some Remarks | 174 |
| | 6.11 Conclusion | 175 |
| 7 | Conclusion and Scope for Future Work | 176 |
| | 7.1 Summary | 176 |
| | 7.2 Scope for Future Work | 177 |
| | Bibliography | 181 |
| | Appendix | 189 |

List of Figures

- Fig 1.1 Structure of the Neuron showing dendrons, axon and the cell body
- Fig. 1.2 Comparing the actual response curve with the mathematical function.
- Fig. 1.3 Evolution of the CNN
- Fig. 3.1 (a) OR Gate showing a linear Decision Boundary (b) XOR Gate with a possible non-linear Decision Boundary
- Fig. 3.2 Error plotted against number of iterations while training a CNN for CoDec Problem
- Fig.3.3 Plots (a) and (b) are respectively, the real and imaginary parts of the complex map $w=\sin(z)$
- Fig. 3.4 (a) shows the input circles and (b) shows the image of the $w=\sin(z_1)\sin(z_2)$ map. All are normalized to within the unit square
- Fig. 3.5 The Problems of Two-Spirals requires telling apart the dotted spiral with the one shown in dots and dashes using a Back-Propagation trained Neural Networks
- Fig. 3.6 Error convergence characteristic with the two spirals problem with real EF based BPA
- Fig. 4.1 Nitta Activation Surfaces
- Fig. 4.2 Real Part of Haykin Activation
- Fig. 4.3 Singular points of Haykin Function
- Fig. 4.4 Real Part of Georgiou Function
- Fig. 4.5 Real Part of New Activation Function
- Fig. 5.1 Bilinear Transformation maps circles to circles
- Fig. 5.2 The polynomials of quadratic and Fourth Power are shown in (a) and (b) respectively. A comparison clearly reveals that the curves are a way different with Fourth Power having a slower rise than the Quadratic.
- Fig. 5.3 The Similarity Transformation. The outer circle is mapped radially onto the inner one which is concentric with the outer and half its radius
- Fig. 5.4 The Exponential Map trained with CNN based on Nitta and New Activation Functions
- Fig. 6.1 Points on the xy -Plane where all the surfaces were sampled

Fig. 6.2 Signatures of Training Surfaces

Fig. 6.3 Signatures of surfaces of Test Set 1

Fig. 6.4 Signatures of surfaces of Test Set 2

Fig. 6.5 Signatures of surfaces of Test Set 3

Fig. 6.6 Points for sampling the surfaces for order determination

Fig. 6.7 Planes, Quadratic and Cubic Surfaces for Order Determination

Fig. 6.8 Typical signatures of Planes, Quadratic surfaces and Cubic surfaces for order determination

List of Tables

Table 3.1 The Error Functions

Table 3.2 Complex Exclusive-OR Map

Table 3.3 Epochs for CXOR and XOR Problems solved using CNN, ANN

Table 3.4 The result with different CNNs tabulated for the CXOR problem. The simulated points shown in dots are close to the target points shown in asterisks.

Table 3.5 Complex 3-Parity Problem is Non-Associative

Table 3.6 Complex 3-Parity problem, composition as in eqn. (3.36)

Table 3.7 Complex 3-Parity problem, composition as in eqn. (3.37)

Table 3.8 Norm computation for the Complex 3-Parity Problem

Table 3.9 CoDec Problem using EF based ANN and CNN

Table 3.10 Result showing the number of epochs required for capturing $z = \sin(x)\sin(y)$. A target error of 0.0001 was set with a 1-5-1 architecture. Result was averaged across three runs.

Table 3.11 CNN Mapping the $\sin(x)\sin(y)$ surface in $[0, \pi/2]$. CNN outperformed ANN in this range of the map. Architecture was 1-5-1, Learning rate was 0.1.

Table 3.12 Real EF based ANNs mapping the $\sin(x)\sin(y)$ surface using a 1-5-1 architecture, in the range $[0, 2\pi]$ where ANN performed better than CNN.

Table 3.13 Approximating $\sin(z)$ using Taylor Series. The figures (a)-(f) show the convergence of the Real Part of $\sin(z)$ as the number of terms of the series

increase while figures (g)-(l) show how the imaginary part of the function approximates.

Table 3.14 Mapping $w=\sin(z_1)\sin(z_2)$ using EF based ANN and CNN

Table 3.15 Mapping $w = \sin(z_1)\sin(z_2)$ using EF based CNNs. The Learn rate was 0.1, architecture was 1-5-1.

Table 3.16 The simulation result with real EF based BPA addressing the Two-Spirals problem is shown in the table. The architecture was 1-5-1. Learning rate was 0.1.

Table 3.17 Comparing the performances of BPA and CVBP for Two-Spirals Problem. The average epochs for convergence are shown for the BPA while the average saturation value and the average epochs for saturation are tabulated for the CVBP. Learning rate was 0.1.

Table 3.18 CXOR with different architectures, using Nitta activation

Table 4.1 Architecture was 1-5-1, Learning Rate was 0.1, Target Error was 0.0001

Table 5.1 1-5-1 architecture, Nitta Activation Function. Learning rate 0.1, 1500 epochs

Table 5.2 1-5-1 architecture, New Activation Function. Learning rate 0.1, 1500 epochs

Table 5.3 Capturing Parabola using Nitta Activation, with a 1-5-1 architecture, learn rate of 0.1. Epochs were set to 2500.

Table 5.4 Capturing Parabola using New Activation, with a 1-5-1 architecture, learn rate of 0.1, 2500 epochs

Table 5.5 Capturing Fourth Power using Nitta Activation, with a 1-5-1 architecture, learn rate of 0.1, 2500 epochs

Table 5.6 Capturing Fourth Power using Nitta Activation, with a 1-10-1 architecture. learn rate of 0.1, 2500 epochs

Table 5.7 Capturing Fourth Power using New Activation, with a 1-5-1 architecture, learn rate of 0.1, 2500 epochs

Table 5.8 Capturing Fourth Power using New Activation, with a 1-10-1 architecture. learn rate of 0.1, 2500 epochs

Table 5.9 Capturing Similarity Transformation using Nitta Activation, with a 1-5-1 architecture, learn rate of 0.1, 1500 epochs

- Table 5.10 Capturing Similarity Transformation using New Activation, with a 1-5-1 architecture, learn rate of 0.1, 1500 epochs
- Table 5.11 $w=\exp(z)$ with 1-5-1 architecture, Nitta Activation, 4000 epochs, Learning rate = 0.1
- Table 5.12 $w=\exp(z)$ with 1-10-1 architecture, Nitta Activation, 4000 epochs, Learning rate = 0.1,
- Table 5.13 $w=\exp(z)$ with 1-10-1 architecture, New Activation, 4000 epochs, Learning rate = 0.1
- Table 5.14 $w=\exp(z)$ with 1-5-1 architecture, New Activation, 4000 epochs, Learning rate = 0.1.
- Table 5.15 $y=\exp(x)$ captured by 1-5-1 Nitta activation based CNNs. Learning rate was 0.1, 2500 epochs
- Table 5.16 $y = \exp(x)$ captured by 1-5-1, New activation based CNNs
- Table 5.17 $w=\sin(z_1)\sin(z_2)$ captured by 1-5-1, Nitta activation based CNNs. 1500 epochs.
- Table 5.18 $w=\sin(z_1)\sin(z_2)$ captured by 1-5-1, New activation based CNNs. 1500 epochs.
- Table 5.19 Errors with test data averaged across three runs. CExp is Complex Exponential Map, Exp is the Exponential Map, BTrans is Bilinear Transformation, Polynomial and $\sin(z_1)\sin(z_2)$ maps in that order.
- Table 6.1: List of Transcendental Surfaces in Practice
- Table 6.2 Algebraic and Transcendental Surfaces
- Table 6.3 Test Surfaces for the Surface Classification
- Table 6.4(a) Test performance of the 100-5-1 network on each test set ('1' indicates correct identification, '0' indicates incorrect identification)
- Table 6.4(b) Performance of various Back-Propagation Algorithms with test surfaces of Set 1.
- Table 6.5 Testing the Robustness of the Neural Network
- Table 6.6 Classification based on Error Function based BPA

Table 6.7 Test performance of complex network on each test set. a1 is 100-5-1 a2 is 100-10-1, a3 is 100-15-1, a4 is 100-20-1.('1' indicates correct identification, '0' indicates incorrect identification)

Table 6.8 Classification based on Error Function based CVBP (Nitta activation)

Table 6.9 Classification based on Error Function based CVBP (New activation)

List of Abbreviations

BPA: Back Propagation Algorithm

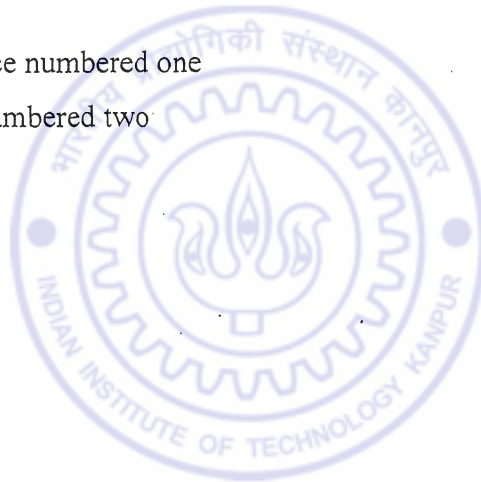
CNN: Complex-valued Neural Network

CVBP: Complex-Variable based Back Propagation algorithm

EF: Error Function

TrS1: Training Surface numbered one

Tst2: Test Surface numbered two



ABSTRACT

Complex-valued Neural Networks have been studied in the thesis from the viewpoint of Error Functions (EF). Practical data are prone to outliers that offset an optimization scheme by contributing greater cost to the standard Quadratic EF. Statistics literature pointed out other EFs that can effectively circumvent this problem and afford a better solution thereby. Seventeen such EFs have been gathered from different sources and Back-Propagation algorithm developed over them. The functions have been generalized to complex variables and Complex-valued Back-Propagation Algorithm (CVBP) developed over them. To validate the EF based networks, the following Benchmark Problems have been employed: Exclusive-OR, n-Parity, Coding-Decoding, Mapping $z=\sin(x)\sin(y)$ and Two-Spirals Problem. Each has been generalized to the complex-variables accordingly and the EF based CVBP have been used to solve them. An error criterion has been used to train the networks. The EF based algorithms have then been used to map the following standard problems: bilinear transformation, polynomials and real and complex-valued exponential function. To study the networks' ability to classify, they have been applied to a surface classification problem requiring sorting of algebraic and transcendental surfaces into different classes, that are later tested with three sets of test surfaces each with varying degrees of departure from the training set to quantify the performance of the trained networks.

SYNOPSIS

The thesis investigates the properties of Complex-variable based Neural Networks (CNN). CNN is a generalization of the Artificial Neural Network (ANN) to the complex domain where weights, biases and activation functions are complex numbers and functions. A recent investigation (Nitta, 1997) reports that the size of the CNN could be smaller than that of an ANN for the same problem (as each complex variable can take two real variables. Also the weights are complex which implies that they hold twice the information as real weights would). Literature review in the area revealed that many questions about the architecture of the CNN and activation functions employed (Leung and Haykin, 1991) have remained open as investigators have either not addressed them or have given partial information on these points of interest. It was discovered during the course of research that some reported results contradicted each other. For instance, Leung and Haykin (1991) claimed that the Complex Activation Function (CAF) given by the formula, $1/(1 + e^{-z})$, (where z is the net input to the complex Neuron), converged in their experiment while Nitta (1997) reported that the same CAF never converged in his experiments. Leung and Haykin (1991) also state that the singular points of the CAF could be circumvented by scaling the inputs to a region on the complex-plane but no procedure to implement this was described. The above facts clearly indicate the need for comprehensive investigation to establish the properties of the CNN.

Complex Activation Functions (CAF) and their characteristics depart from the traditional activation functions of the ANN. Firstly, these functions have real and imaginary parts each of which individually are functions of two variables that make them surfaces in three-dimensional space. Secondly, there exist additional constraints imposed by the complex plane in the form of Liouville Theorem (Ahlfors, 1979) that restrict the choice of functions that could be used as the CAF. As a result, tailoring new CAF by sewing pieces of surfaces along the common boundary would not be an acceptable proposition as the analyticity of the function developed this way should be established at each point on the boundary and later the construct must be verified to have cleared the constraint imposed by the Liouville Theorem. This restriction is unlike real value activation functions that could be easily tailored by joining differentiable functions and establishing

differentiability at finitely many points (where each piece joins up with the next). During the course of this study various CAFs reported by researchers have been investigated in the above context. A new CAF is also proposed. The number of inputs is dropped by as much as half in a CNN so also the sizes of the hidden layers. But the mechanism by which the phase (equivalently the imaginary part of the complex weights) compensates for the drop in the number of inputs is not addressed in the available literature. The thesis attempts to address the issue of size reduction by means of some computational experiments.

In almost all investigations and applications involving ANNs, the Back-Propagation Algorithm (BPA) applied is the one developed over a Quadratic Error Function. This Error Function (EF) may not perform satisfactorily for real life data with function approximation and classification. Literature points out many other functions that can take the place of the traditional Quadratic Function in data analysis but these EF based applications have been studied from a Statistics viewpoint and not from a Neural Network viewpoint. Incidentally, Rey (1983) pointed out that in Statistical Analysis replacing EF can yield better results. It must be stressed here that the EF over which the complex-variable based Back-Propagation Algorithm (CVBP) was built recently also is the Quadratic Function. However, a few researchers (Werbos and Titus (1978), Gill and Wright (1981), Fernandez (1991), Ooyen and Nienhaus (1992)) have used different EF with their BPA. While sufficient information exists about the EFs in literature, complex EFs have not been investigated in a systematic fashion. Different experiments have been carried out over proposed EFs in the complex domain. Replacing the EF assumes importance because practical data are prone to measurement errors and outliers. If the Quadratic EF were retained for analyzing data prone to outliers and other errors, the curve of best-fit would not be appropriate because the cost that accrues to the chosen EF would get enhanced due to the power term for far-off points (outliers). Instead, if the Quadratic EF were replaced with an Absolute Function for example, the curve-fitting scheme for noise and error-prone data would be more evenly placed because the cost accrued due to these data would be of the same order as the actual data points. This even-weighting results in a better curve-fit than the one obtained by Quadratic Error based approximation.

The approach to the investigation went along the following lines. The two important parameters that exist in the problem are: EF based CVBP and an associated CAF. For each problem tried, each of the seventeen EFs was combined with each of the CAFs by selecting the EF first and running the training by choosing the CAFs one after the other. Hence, the Nitta, Georgiou functions and New Activation function have been employed with each EF during the investigation.

Of the various parameters set to run the CNN, the CAF is most critical. In the complex domain, analyticity of the CAF must be verified before it can be used as an activation function to the CNN. This aspect is unlike ANN where no such constraint existed. The CAFs available from literature have been collected and analyzed. The functions are: Haykin Activation, Nitta Activation Georgiou Activation (Georgiou and Koutsougeras, 1992) that were comparatively studied. The newly proposed CAF satisfies all the constraints imposed by the theory of complex-variables. The singular points of the Haykin activation function which occur at the points $(0, (2n+1)\pi)$, (n an integer) were found to be responsible for disrupting the downstream convergence whenever at least one of the net inputs to neurons fell in their vicinity in the course of training.

Activation functions were studied on a comparative basis. The set of benchmarks used for comparative study of EF based algorithms are (Dagli, 1994): Two-Spirals Problem, Co-Dec Problem, Parity Problem, mapping the surface $z = \sin(x)\sin(y)$, Exclusive-OR Problem (XOR). These benchmarks were generalized to the complex domain and the CVBP based CNN was used to address them.

The Two-Spirals problem did not converge for all the EF based CVBPs. The non-linearity of the problem was preserved while generalizing, the outputs were chosen to be complex numbers for the two spirals. The extended version of the XOR problem, the complex XOR (CXOR, Nitta (1997)), was used to develop complex 3-Parity map. Owing to the non-commutative nature of the CXOR map, the complex 3-Parity turns out non-associative. Hence the two ways of evaluating the inputs for the complex 3-Parity mapping are $(c_1 \circ c_2) \circ c_3$ and $c_1 \circ (c_2 \circ c_3)$ where c_1, c_2, c_3 are the three inputs complex numbers. Higher dimensional complex Parity problems require evaluation of the inputs in several ways owing to the non-commutative CXOR map. It was observed that the CNN's were sensitive to the order of evaluation of inputs (hence to the non-associativity of the

map). A mechanism was proposed to explain this observation: the intermediate complex numbers (the bracketed ones in the three complex number composition) determine how the output would be and hence although it (the intermediate complex number) is close to the third argument (in the sense of the EF) the final output need not be. This happens more often with the second way of composing $(c_1 \circ (c_2 \circ c_3))$. The complex version of the Co-Dec was developed by assigning $(1+i)$ to the diagonal entries (instead of the usual unity along the diagonal). The surface map $w = \sin(z_1)\sin(z_2)$ was considered as this generalizes the $z = \sin(x)\sin(y)$ map to complex variables (Dagli, 1994). The mapping studied was the one obtained by setting z_1 and z_2 to curves on the respective planes and capturing the output curve on the w -plane. It turns out that the generalization is a plane-to-plane mapping unlike the real variable case where it was a surface in three dimensions. Different EFs used in literature were collected and BPA were developed over each of them. Explicitly the EFs are (Rey, 1983) – Absolute EF, Andrew EF, Bipolar Hyperbolic EF, Cauchy EF, Fair EF, Fourth Order EF, Geman-McClure EF, Huber EF, Hyperbolic Squared EF, Logarithmic EF, LogCosh EF, Mean-Median EF, Minkowski EF, Quadratic EF, Sinh EF, Tukey EF, Welsch EF. These functions were later generalized to the complex-variables by retaining the form in each case but extended to accommodate complex errors. The CVBP were constructed with these extended EFs. All these EF based algorithms have been applied to the following Benchmark problems listed above (Dagli, 1994) and studied comparatively. The generalized versions of the benchmarks have been used with the CVBP and the CAF. The Nitta Activation was employed for the training scheme with the CNN.

The CNN were next studied for their ability to map functions. The following maps were considered for the experiment: the polynomials, the Bilinear Transformation, Similarity Transformation, the Exponential Map (real and complex). The study was conducted with all the EF based CVBP. The New Activation Function was used to capture these and its performance is compared with that of the Nitta activation. The results indicate that the EF based CNN learns to capture these maps.

To study the CNN's ability to classify, twenty-five surfaces were considered (twelve algebraic and thirteen transcendental) to include rotational symmetry, planar symmetry and a data set was developed using their point clouds (sampled at hundred points on the

xy – plane). This problem of classification was solved by applying the EF based BPA and also EF based CNN. To test the network designs, three sets of test surfaces were constructed. In the first set, the test surfaces were the training surfaces with a mild variation in the parameters. In the second set the parameter variation is considerably large while the third set has new surfaces included. The results obtained were later compared. While modeling the problem, due weight was given to the geometric interpretation and hence the hundred by one input vector was not tailored into a fifty-by-one complex vector (as a result, the imaginary part of the inputs had to be assigned zero). The output however was chosen to be unity for algebraic surfaces and i for the transcendental surfaces. The weights and activation functions however were complex numbers (with non-trivial imaginary part) and functions.

In surface mapping problems, the CNN outperformed the ANN when the norms of the input vector were small while the ANN performed better for larger norms of the input vectors. All the Error Function based CNN's learned to map Polynomials, Bilinear Transformation, Exponential mapping (real and complex) to a relatively better degree of accuracy. The New Activation function proposed in the thesis outperformed the Nitta Activation function proposed in literature.

In short, with the Benchmark problems using CNN, the Nitta Activation was found to be the best across different architectures. The New Activation and Georgiou Activation were found to perform equally well but were inferior to the performance of Nitta Activation. The Haykin Activation showed poorest performance (owing to the existence of singular points). In Error Function study employing ANN and CNN, Cauchy Error Function, Log-Cosh Error Function, Absolute Error Function, Huber Error Function, Mean-Median Error Function have either equaled the Quadratic Error Function or have performed better in the Surface Classification problem (across three runs of training) employing Nitta Activation Function. In Error Function study with CNN addressing function mapping problems, Sinh Error Function, Tukey Error Function, Fair Error Function, Mean-Median Error Function performed on par with Quadratic function or have shown better performance (across three runs of training) using Nitta Activation and New Activation Function.

The main contributions of the thesis are

- It proposes an EF based approach to Neural Networks (both ANN and CNN); that the EF can be varied to advantage in practical applications that employ Neural Networks is established.
- The CVBP and the CAF are surveyed, studied and compared while a new CAF is proposed.
- The mapping properties and a classification problem were studied and the EF based networks were graded according to their performance.
- The properties of the CNN were investigated to determine appropriate CAF and EF for given set of problems relating to Classification and Function Mapping.



Chapter 1

Introduction

1.1 The Neural Network

The basic structure of the ANN evolved in due course from the basic study of the Neurons in the Brain. The anatomy of the human brain shows three prominent lobes – the big brain or the cerebrum, the middle brain or cerebellum and the hindbrain or medulla oblongata. The medulla synapses with the nerves that extend into the spinal column and divide and sub-divide as they spread out into the various parts of the body. The nerve-endings are fibrous and end in the sense organs. The set of nerves that carry impulses to the brain from the sense organs are called sensory nerves while the ones that take the signal from the brain to the muscles are called motor nerves. The nerve endings receive impulses and through the network of nerves pass them to the brain for interpretation. Studies revealed that different parts of the brain are responsible for various faculties. The cerebellum coordinates the limb movements; the cerebrum integrates information from all sense organs, controls emotions and holds memory and thought processes. The hypothalamus and pituitary gland control visceral functions, body temperature and behavioral responses.

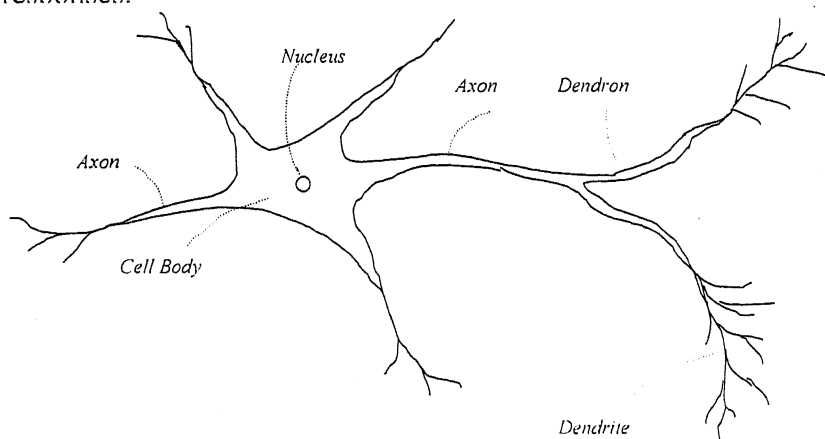


Fig 1.1 Structure of the Neuron showing dendrons, axon and the cell body

The brain is made up of a hundred billion cell, ten billion of which are 'Neurons' that are responsible for the different functions of the brain (Fig. 1.1). A typical Neuron has the central cell or the cell body, the axon, the dendrons that give rise to dendrites. The fibrous dendrites synapse with the cell body and axon of the neighboring Neurons. Typically each Neuron has ten thousand neurons in its vicinity to which it connects. The conduction of electrical impulse through the structure just described is the mechanism that results in the interpretation of the signal. Neurons respond to the input signal impinged onto them in a certain fashion that is referred to as the characteristic of the Neuron. A typical characteristic resembles the sigmoid (Fig. 1.2).

Much research went into how the Neurons operate in the unit, how they respond to an input impulse. A typical signal to the Neuron would be of the order of forty milli-volts, and the response of the Neuron depends on the threshold voltage it operates with. That is, the Neuron responds or 'fires' if the input voltage is greater than its threshold and doesn't otherwise. Research revealed that the Neurons respond according to their characteristic function, also called 'activation function' which is a general term to describe how a Neuron responds to a stimulus. Threshold function clearly is one kind of activation function.

1.2 Artificial Neural Networks and Complex Variable based Neural Networks

An Artificial Neural Network (ANN) is a model that mimics the real Neuron described. In the model description of the Neuron, the activation function is described by the sigmoid function that closely resembles the activation of the real Neuron (Fig. 1.2). The Artificial Neurons are shown connected with links going from one layer to the one immediately succeeding it and not to the layers that lie beyond the immediate successor (some applications of Neural Networks, however, have synapses that link the neurons of the present layer with the ones not only of the immediately succeeding layer but also to the neurons that lie further up in the line (Lang and Witbrock, 1988). The update rules for the new synaptic links have to be derived separately). Every such connection has strength associated with it quantified by a real number called 'weight.' A steep-ness factor was introduced to adjust the shape of the activation function and tailor it to a form that closely resembles the actual characteristic. The voltage level at the synapses in the brain is the

synaptic strength to which the real number called weight is associated. In the human brain the weight is actually the potential that controls the flow of electric impulses through the link.

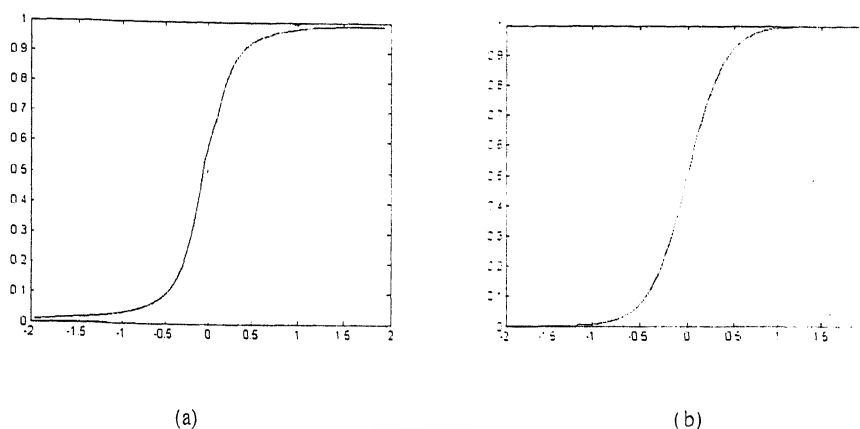


Fig. 1.2 Comparing the actual response curve with the mathematical function (a) The actual characteristic at the output to the neuron in the brain. (b) The sigmoid function approximates the characteristic.

Neural Network algorithms are applied widely today. Our understanding of the Networks improved over the years with the much light research in the direction threw, with the lasting contributions of McCulloch and Pitts (1943), Donald Hebb (1949), Minsky (1954), Rosenblatt (1958), Jon von Neumann (1958), Minsky and Papert (1969), Werbos (1974), Fukushima and Miyaka (1980), John Hopfield (1982), Nitta (1997) to name a few. Among the most recent developments in the area are the Complex variable based Neural Networks (CNN) that represent a second generation of architectures that scored over the standard Real variable based networks (ANN) in certain aspects (Nitta, 1997). The CNN is new because it operates in the complex variables setting and so the conventional Back-Propagation Algorithm (BPA) that trains the ANN is not designed to train the CNN. An extension of the BPA to the complex variables was reported by Leung and Haykin (1991), called the Complex Back-Propagation Algorithm (CVBP). A study of how the CNN's perform in comparison with ANN's would be interesting, that would be the theme of the rest of this report.

The following issues need attention from the viewpoint of the new algorithm. There exist additional constraints (Liouville Theorem) in a complex variable setup than there are in the real. It is hence not clear how the new algorithm would perform working with the

constraint when applied to various problems. The update rule is exactly same as the one employed while running the BPA to train an ANN (eqn. (1.1)), it must be noted that the complex number comes with the phase information embedded into it. This amounts to saying that the information that would have separately been input to the ANN while training (as is usually done while training ANNs) gets coupled resulting in a decrease of the number of inputs by as much as half (as two real numbers make one complex number) at the same time preserve the information in the form of phase. It is hence not clear how phase can effectively compensate for the decrease in the number of input parameters and yet solve the problem at hand - function mapping or classification. In most problems of practical interest that involve an optimization process, a Quadratic Error Function is chosen and subject to an optimization (in Neural Networks, the weights of the network are updated so the derivative of the Quadratic Error Function is as small as desired, eqn. (1.1)). The BPA as it is employed today, also uses a Quadratic Error Function. It was pointed out in literature (Werbos and Titus (1978), Gill and Wright (1981), Fernandez (1991)) that implementing the BPA employing a different Error Function can improve the performance of the Neural Network. The BPA minimizes a Quadratic Error Function by steering the weights along the direction of negative gradient (using the update rule). It is apparent that a Quadratic Error Function is used in the process even as the literature points out alternative functions that can effectively improve the performance of an optimization scheme. In fact in the *m*-Estimators approach to data analysis (Rey, 1983), a number of functions that can effectively serve as Error Functions have been listed. It was shown that the new Error Functions have the ability to suppress the ill-effects due to outliers and exhibit a robust performance to noise and outperform the standard Quadratic functions when applied to optimization problems involving data with a scatter of outliers (the fact was demonstrated using an Absolute Error Function to compute the cost instead of a Quadratic Error Function). Hence the question of how the BPA would perform when the Error Function is varied immediately comes up. The work presented in the thesis is an attempt to address these points. These points are addressed by applying the CNN to problems of varied sorts. The CNN's are just beginning to be understood as only a few investigators have reported work in this area (elucidated in Chapter 2).

Most practical problems that come from various fields (Medicine, Industry, Military, Aviation and so on) that involve modeling with Neural Networks employing BPA (or CVBP) can be modelled as Classification Problems or Function Approximation Problems. Hence to ascertain the performance of the CNN, they must at least be applied to problems of these two types. Therefore the issues raised were addressed by applying CNN to problems of Function approximation, Function mapping and Surface Classification. The application of complex variable based Neural Networks however becomes clear once they are applied to some standard problems.

It must be noted that a given data can be tailored in different ways to suit for input to the CNN. For instance, if in a certain problem there exist n inputs and m outputs, a variety of combinations naturally come up while grouping the m inputs into m_1 complex numbers and similarly n outputs also into n_1 complex numbers to construct the data patterns to design a CNN (while making sure the dimensions match while grouping the input variables into complex numbers). This is an additional feature of the CNN that a proper grouping can hasten up the training process and enhance the performance result (this was observed in a sequence of CNN designs for approximating the sine curve. It was observed in the experiment that the CNN which had input real but output purely imaginary took longer to train with a learning rate of 0.1 and an architecture 1-5-1 than did the network with same architecture and learning rate and weights initialized identically but the output set to be purely real). As complex numbers are dimension two with respect to the set of real numbers (Halmos, 1974), variety of CNN designs for a particular problem become available depending on the grouping and the architecture chosen. Needless to say that the interpretation of the simulated result should be done in the way in which the training data was modeled which means if the output was chosen to be purely complex while training, the imaginary part of the simulated data should be considered for interpreting the results.

1.3 Updating Complex-valued weights

As was pointed out in the previous section the CNNs operate in the complex domain. Hence, to understand the various mechanisms from the viewpoint of CNN, it becomes imperative that we study the basic formulation of the same by taking the properties of

complex variables into account. The weight update rule for the CNN is exactly same as the one used to training networks using the BPA

$$w_{ij}(n+1) = w_{ij}(n) + \eta \frac{\partial E}{\partial w_{ij}(n)} \quad (1.1)$$

where, w_{ij} are the weights that get updated as the algorithm runs, E is the Error Function that gets minimized in the process of weight update and η is the learning rate. The difference of course lies in the fact that the weights are complex numbers while the learning constant is a small positive real number. The functions of activation that the Neurons fire according to are all complex in nature. It is hence obvious that a study of complex variables and complex mappings is essential to comprehend the mechanism by which a CNN works. A list of basic definitions required for a systematic study has been given in Ahlfors (1979).

1.4 A Few Remarks

The complex plane is unlike Real line for it is dimension two with respect to real numbers and one dimensional with respect to the set of complex numbers (Halmos, 1974). A point on the plane can be viewed as a complex number with the x and y coordinates regarded as the real and imaginary parts of the number. The set of complex numbers is a Field and hence a perfect platform of operation but is devoid of order. The properties of the complex plane are different from those of the real line. The complex numbers have a magnitude associated with them and a phase that locates the complex number uniquely on the plane. It is hence clear that the complex CVBP that trains the CNN must not only obtain a convergence with respect to the magnitude but also with respect to the phase. This is equivalent to stating that the real as well as imaginary parts of the complex numbers must be separately captured by the CVBP.

1.5 ANN versus CNN

A brief survey into the history of the development of ANN points to the fact that the first idea of developing the architecture in a way as to mimic the neuronal arrangement in the

brain came from a study of the brain anatomy and analysis of the micro-structure of it (as pointed out in section 1.1). The characteristic that the typical neuron displayed when impinged with an input force (voltage) resembled the well-known sigmoid function that in a study down the line became accepted as the function of activation of the Neuron. The ANN from the incipient stages grew as more facts about the human brain surfaced with developments in Brain Research (Rose and Bynum, 1982). The results of this research that could be improvised, adapted and embedded into the ANN got accrued with time and enhanced its performance, brought some limitations to light (Kolmogorov, 1957; Werbos, 1974; Wang, 1992). On the other hand, heresy to the development of the ANN, the CNN came about as an extension of the ANN and not as a prototype of the neuronal arrangement in the brain (for the synaptic strengths in the brain are not complex valued but are voltages that could be represented by real numbers). The activation function too was an extension of what existed as the function of activation in the ANN but needed an improvisation to suit to the complex variable based ambience (and hence don't resemble the actual activation function of the brain's neurons). In the process, new constraints surfaced (in the form of Liouville Theorem) that had to be cleared for which a different search had to be carried out. In essence, the CNN is an extension of the ANN but doesn't draw anymore from the actual neurons and their arrangement in the brain. Fig. 1.3 portrays a diagrammatic representation of the view presented.

1.6 Organization of the Thesis

The objective of the present thesis is to study the CVBP from the viewpoint of Error Functions and compare the performance of these with different BPAs developed over Error Functions. A set of benchmark problems has been chosen for the study.

Literature Survey is presented in Chapter 2. The various Error Functions have been studied in Chapter 3. The BPAs were developed over these functions and applied to the benchmark problems. The Error Functions were extended to the Complex Variables setting and CVBP were developed over them and applied to the same benchmark problems to which the BPAs were applied and a comparison was drawn. The Complex Activation Functions (CAF) have been analyzed in Chapter 4. The functions put forth in

literature have been surveyed and comparative merits and demerits have been pointed out. The study also addressed the ramifications of the Liouville Theorem. The Benchmarks studied in Chapter 3 have been used with yet another CAF based CNN. A new CAF has been put forth and studied. It is shown that the new CAF performs on par with the existing ones. It is concluded that the singularities of the CAF are critical for the performance of the CVBP and the CNN. The benchmark problems earlier chosen have been addressed from a CAF viewpoint and results compared.

Chapter 5 deals with the Mapping Properties of the CNN. Some maps that were pointed out in literature (Nitta, 1997) have been studied from an Error Function viewpoint. Various other maps practically employed in various fields like Fluid Dynamics, Electromagnetic Theory, Stress Analysis, Aerofoil Design have been trained using the CNN.

A classification problem, one of identifying surfaces as Algebraic or Transcendental is considered in Chapter 6. The problem naturally arises in Reverse Engineering where the idea is to reconstruct the equation of the surface from a point cloud. All the available literature in the area describes procedures to approximate the cloud of points using polynomials while in the process, the actual form of the surface remains unknown. Hence the problem of identifying the kind of surface from which the point cloud was generated is a basic problem that the research in Reverse Engineering always overlooked. In problems where the constraint is extremely important (the tonal quality of the vibrating air column of a trumpet for instance), approximating the instrument's surface by a polynomial might offset the location of the nodes and anti-nodes of the air-column resulting in a poor sound quality (Askill, 1979). The Algebraic-Transcendental classification studied in Chapter 6 finds application to such problems where the approximating with algebraic polynomials may not yield accurate results. The problem is addressed in this thesis using the Error Functions based BPAs and the CVBPs.

Chapter 7 concludes the thesis giving a summary of the main contributions and presents directions for further research in the area.

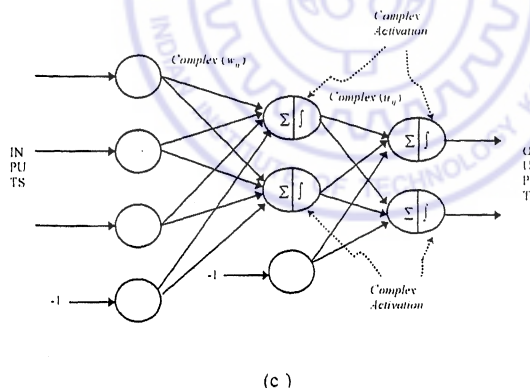
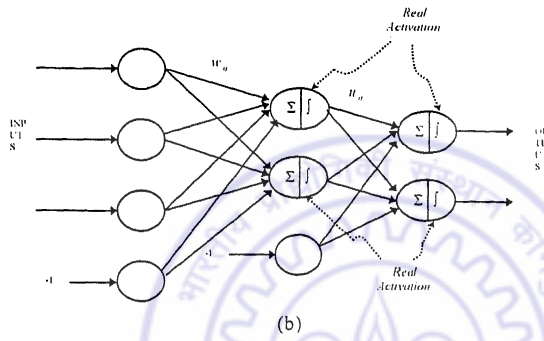
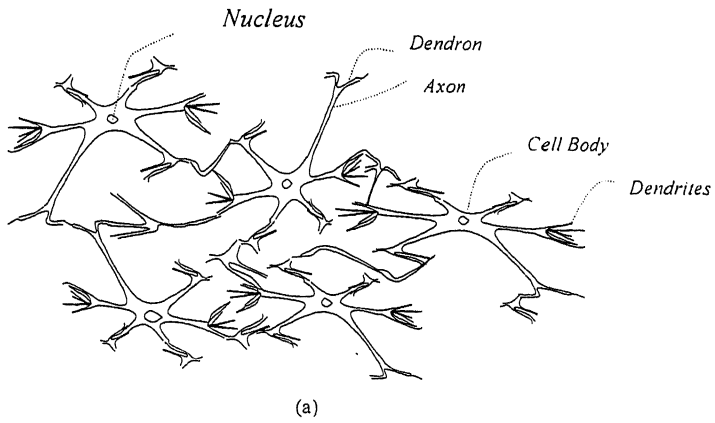


Fig. 1.3 The figure shows how the CNN evolved. (a) The neurons in the Brain (b) The ANN was developed by studying the neurons while (c) the CNN came into being as an extension of the ANN.

Chapter 2

Literature Survey

2.1 Research on ANN

The first mathematical result about the convergence of composition of functions was given by Kolmogorov (1957) who stated that any continuous real valued functions in n - variables defined on $[0,1]^n$ ($n>2$) can be represented in the form of composition of function as shown

$$f(x_1, x_2, x_3, \dots, x_n) = \sum_{j=1}^{2n+1} g_j \left[\sum_{i=1}^n \varphi_{ij}(x_i) \right] \quad (2.1)$$

When the result was announced, it wasn't clear how it might be applied and that left the investigators of the time puzzled. Many results followed the discovery of the ANN that stated how and why the Neural Network algorithms would converge. For instance, there were results on how to select activation functions to ensure the Neural Network scheme converged (Hassoun, 1995). Some effort describing how to select an architecture to ensure convergence (Mirchandani and Cao, 1989) was also reported. It should be however pointed out that most results in the area of Neural Networks are existential in nature; for instance, to state convergence obtained by an algorithm the theoretical results assure the existence of weights (Rudin, 1976) that can approximate the data with functions of the specific form but how the weights can be obtained is not explicitly mentioned.

The first successful attempt to develop the ANN architecture was made by McCulloch and Pitts (1943), hence from the incipient stages since 1943 with the discovery of the perceptron the theory of Neural Networks reached a stage where all algorithms developed here find application to every field studied today. The first successful model of Neuron didn't appear until the ADALINE (Adaptive LINEar combiner) entered the scene in the

11

1960's and the Widrow-Hoff learning rule that trained the ADALINE based networks. The ADALINE or the Adaptive Linear Network is the basic element of the Neural Network that adds the inputs incident onto it. The MADALINES or Many ADALINES performed the linear combination with many ADALINES; the Widrow-Hoff rule minimized a sum-squared error in a pattern classification problem as it trained the MADALINES based Neural Network. Computational power available at the time was insufficient to support the load due to training algorithms and as a whole the research in the area slowed down drastically due to the impediment. Nilsson (1965) put forth the idea of multi-layered networks but the concept didn't receive much attention as the research went slow at the time. Minsky and Papert (1969) published a book that further put Neural Networks in jeopardy as it questioned the potential of Neural Networks as computational tools and exposed the limitations of the perceptron. Research almost stopped since the publication of the work of Minsky and Papert and not much research happened for about twenty years between 1965 and 1984. But the few who pursued Neural Networks during the period made lasting contributions: mathematical theory of Neural Networks was developed by Sun-Ichi Amari (1972, 1977), Fukushima (1980) developed the Neocognitron, Associative Memory was developed by Tuevo Kohonen (1977, 1980).

In fact it was much after Kolmogorov published the first theoretical result that a consistent method (that worked universally) to obtain the weights was worked out by Werbos (1974), employing the idea of gradient descent. Although Werbos' discovery of the BPA was an important milestone in the history of Neural Networks, the method isn't fully free from bottlenecks. The problem of local minima while training using the BPA needs a special mention. Depending on the initial condition the network was set to, BPA might steer the Neural Network into a local minimum at which the training process gets stuck (the weights do not get updated but stay frozen even as the epochs run). To circumvent the problem, some algorithms for minimization without the use of derivatives have also been reported (Brent, 1973). These involve the ideas of random search techniques. In addition to this, methods to prune synaptic links that are least sensitive to the training process have been recently proposed (Karnin, 1990). The redundant synaptic links are severed from the network in this approach. This process improves network

12

generalization by decreasing the number of weights, results in a reduced network complexity, and decreases the needed computation. As applied to Differential Equations the networks have been used to study chaos in dynamical systems (Aihara *et. al.* (1990)). Complex chaotic neural networks were studied by Hirose (1992). Differential Equations were modeled using Neural Networks and solutions of equations were studied by inputting chaotic initial conditions to the network. Along the same line but for small architecture Neural Networks with delay, Francois and Chauvet (1992) reported dynamics of Neural Networks classifying the regimes as stable, unstable or oscillatory. The chaotic Neural Networks have also been applied to information processing (Ishi *et. al.*, 1996).

Fletcher and Reeves (1964) reported a version of the Conjugate Gradient Algorithm. Beale (1972) developed the idea of conjugate gradient further by computing the gradient using a different formula. Battiti (1992) reported an efficient method to compute the conjugate gradient. Charalambous (1992) furthered the step by developing a Conjugate Gradient based BPA incorporating efficient methods discovered. The scaled conjugate gradient algorithm was put forth by Moller (1993) in which a scaling based on the position in the weights space was used in conjunction with a conjugate direction based update of weights. A highly efficient way of training using gradient descent, embedding the good features of the second order algorithms (that involve computing the Hessian) by using an approximation to the Hessian and bypassing the actual computation of it is the Lavenberg-Marquardt Algorithm put forth by Hagan and Menhaj (1994). The Neural Network is initialized with a set of weights and the performance of the final Network depends on these initial weights. The effect of adding noise during BPA training and the final network's performance were studied by An (1995) while the generalization performance based on the weight initialization was studied by Amir *et. al.* (1997). In the same year, Dai and MacBeth reported their observations on learning parameters and how they influence a BPA based training. The size of the architecture that optimally suits a problem remains an open problem in Neural Networks to this day. Huang and Babri (1998) reported a result that yielded an upper bound to the number of hidden neurons for Neural Networks with arbitrary non-linear activation functions. Guarneri and Piazza

(1999) reported an adaptive spline-type activation function for standard Neural Network based training. Among the new types of architectures proposed are the Cascade Correlation Networks by Fahlman and Lebiere (1990) and the SOPNN (Sum Of Product Neural Networks) by Chun *et. al.* (2000) that found application to multi-variable functions. The Neural Network in this application determined if there were obstacles on the road based on a processed vector that was the resultant of the sensor data and an image processing algorithm.

Rey (1983) pointed out that by varying the Error Function in an optimization scheme, the result could be improved substantially. The statement was backed by demonstrating (Rey, 1983) that an Absolute Error Function based optimization solved a curve-fitting problem more efficiently than the standard Quadratic Error Function based optimization. Earlier Werbos and Titus (1978), Gill and Wright (1981) also discussed the idea of changing the Error Function in an optimization scheme. Fernandez (1991) implemented some new Error Functions that were designed to counter the ill-effects local minima by weighting the errors according to their magnitudes. Matsuoka (1991) reported BPA based on logarithmic Error Function and elimination of local minima. Ooyen and Nienhaus (1992) used an entropy type Error Function and showed that it performs better than the Quadratic Error Function based BPA for function approximation problems. Among the various other applications are the control problems from the aerospace industry where Neural Network algorithms have been used to determine satellite orbit motion (Sinha *et. al.*, 2000, 2000a) employing a Compensatory Neuron model, proposed in the paper. This novel model incorporates a product term inside the neuron (in addition to the usual weighted sum of signals from the previous layers) as a compensation scheme and obtains an accelerated convergence. Motion planning problems encountered in robotics to which neuro-fuzzy models of solution have been proposed (Krishna and Kalra, 2000). The Neural Networks used in the application determined an appropriate set of parameters for the membership functions that supported a rule-base for a fuzzy logic model that implemented the control. The automobile industry developed an ALVINN (Autonomous Land Vehicle In Neural Networks; Jochem *et. al.*, 1995) that steered a land motor vehicle without a driver. Image-Processing techniques were coupled with Neural Network

algorithms to detect and circumvent obstacles as the land vehicle navigated the set path with obstacles introduced purposely to test the effectiveness of the Neural Network scheme.

2.2 Complex Variable based Neural Networks

The research in the area took a different turn in the early 1990's with the publication of Complex Back-Propagation Algorithm (CVBP). A first attempt to generalize logic gates to complex-valued functions was made by Aizenberg *et. al.* (1971). The complex version of the BPA, however, made its first appearance when Widrow, McCool and Ball (1975) announced their Complex Least Mean Squares (LMS) algorithm. Kim and Guest (1990) published a complex valued learning algorithm for Signal Processing applications. The necessity came in the form of capturing the phase information in Signal Processing applications where complex numbers naturally enter the study and must be retained all through the problem as they should be later interpreted. Lueng and Haykin (1991) published the CVBP in which the activation used was an extended version of the sigmoid function. Georgiou and Koutsougeras (1992) published another version of the CVBP incorporating a different activation function. The dynamics of complex valued networks was studied by Hirose (1992) which was later applied to the problem of reconstructing vectors lying on the unit circle. Benvenuto and Piazza (1992) developed a variant of the CVBP by choosing a different activation function. A complex-valued Recurrent Neural Network was proposed by Wang (1992) that solved complex valued linear equations. Deville (1993) implemented a complex activation function for digital VLSI Neural Networks that required lesser hardware than the conventional real Neural Network would need. Smith and Hui (1997) used the CVBP to data extrapolation. It was shown that the traditional method of Fourier Transform to reconstruct the data was inferior to the CVBP based Neural Network for short data sets. An extensive study of the CVBP was reported by Nitta (1997) in which the algorithm along with certain problems that the standard BPA fails but the CVBP manages to solve were pointed out. The paper concludes saying that the set of problems to which the CVBP can be successfully applied is a research theme in the light of the problems that the CVBP solves but the BPA fails to do so. Weber and Casasent (1998) used Neural Networks with complex weights to perform a piecewise

approximation of plane curves. Recently, some applications of CNN to optics have also been reported (Takeda and Kishigami (1992), Hirose and Miller (1996)).

It can be seen that the CVBP is steadily gaining prominence although the algorithm as yet is in an embryonic stage. The avenues for the BPA further open up as the survey indicates but once established the CVBP can compete with the BPA in problems where both could be applied. Needless to state that the CVBP would be preferred over BPA in applications that demand the real and imaginary parts of complex numbers and functions be retained and no modeling involving a tailoring with these quantities may be allowed. Such applications require that the physical significance of the complex numbers be kept intact. Typically, Signal Processing, satellite channel equalization are two areas where such requirements exist.



Investigation of Error Functions

The chapter deals with Error Functions in detail. The CNNs are studied by varying the EF. The Error Function approach to data analysis is emphasized from a CNN viewpoint. Practical data is prone to outliers resulting from observational or experimental errors. If the Quadratic EF were used to analyze data containing outlier points, the optimization scheme would result in estimates that depend significantly on the outliers as the Quadratic EF assigns greater cost to far off points. A different EF, an Absolute EF can perform better optimization than Quadratic as the cost that accrues to the Absolute EF due to outliers would be lesser than that with the Quadratic EF resulting thereby in a better estimate. The present chapter surveys some EFs and studies BPA and the CVBP from an EF viewpoint.

3.1 Why Vary Error Function?

The BPA as it is applied today employs a Quadratic Error Function (QEF). In fact, most problems that involve minimization employ a QEF; for example, the problem of regression in statistics employs the QEF to develop normal equations that are later solved to obtain the curve of best-fit. A typical procedure involves assuming a form to the data set (a cubic polynomial with four coefficients) and assuming a Quadratic Error between outputs of the assumed equation and the actual output. By subjecting this error function to minima by equating its derivatives with respect to the coefficients to zero (that are referred to as normal equations) the best-fit curve is obtained in the Least-Squares sense. It is known that the data set obtained while practically experimenting is prone to system noise, process noise and measurement errors (like parallax). The outlier points contribute to the offset in the solution to the curve-fitting problem (Rey, 1983). There exist two approaches to tackle the undesirable affects of spurious data points. The first approach demands that these points be eliminated completely (by some data processing technique) and later after weeding out these points, subject the data to a QEF based optimization scheme and obtain a solution, which can be termed ideal approach. The second approach, as explained in Rey (1983) requires incorporating a modified Error Function that would

by the nature of design and construction have useful properties to bypass the ill-effects of the spurious points in the data and obtain a better fit of curve to the data set than the Quadratic Error Function. Among the many alternative Error Functions proposed in the m -Estimators approach to data analysis are the Absolute Error Function, Fourth Power Error Function and so on. For instance, the Absolute Error Function is defined by the equation

$$y = |x| \quad (3.1)$$

which when applied to data prone to outliers and analysis obtains a better curve of fit than the QEF. This is due to the fact that the cost that accrues to the Error Function due to a far off outlier point would get enhanced in the presence of a power term in the definition. Hence, if the error measure were computed by a QEF, the cost accrued would be higher than would be when the Error Function was the Absolute Error Function. The strategy of adopting an Absolute Error Function results is a better estimate to the curve of best fit than the one obtained by adopting the QEF as was demonstrated in Rey (1983).

Taking this as the point of start, the following questions need attention from the viewpoint of the BPA and the CVBP: how the BPA and the CVBP would perform when the Error Function is varied, validate the performances by applying them to some well-known benchmarks. The following Error Functions were collected from various sources.

3.2 Real Error Functions

The Error Functions for the ANN are listed in this section. With usual notation, $e_i = T_i - O_i$. A reference to all Error Functions listed here is Rey (1983).

3.2.1 Absolute Error Function

The Absolute Error is given by

$$E = \sum_n |e_i| \quad (3.2)$$

where n denotes the number of outputs. Absolute Error is one of several robust functions that displays less skewing of error due to outliers. A small number of outliers are less likely to affect the total error and so they do not affect the learning algorithm as severely as the Quadratic Error.

3.2.2 Andrew Error Function

The Andrew Error Function is given by

$$E = \sum_n \begin{cases} 1/\pi^2 * \cos(\pi * e_i); \text{if } |e_i| \leq 1 \\ 0; \text{else} \end{cases} \quad (3.3)$$

where n is the number of outputs.

3.2.3 Bipolar Hyperbolic Squared Error Function

The Bipolar Hyperbolic Squared error is given by

$$E = \sum_n \ln \left(\frac{2 - e_i^2}{2 + e_i^2} \right) \quad (3.4)$$

where, n is the number of outputs.

3.2.4 Cauchy Error Function

The cauchy's error function is given by

$$E = \sum_n \frac{c^2}{2} \ln \left(1 + (e_i/c)^2 \right) \quad (3.5)$$

where n is the number of outputs while c is the tuning constant. Cauchy Error Function, also known as the Lorentzian function, is one of the most robust functions of statistics. The tuning constant for the EF is 2.3849.

3.2.5 Fair Error Function

The Fair function is given by

$$E = \sum_n c^2 \left[(|e_i/c|) - \ln(1 + (|e_i/c|)) \right] \quad (3.6)$$

where n is the number of outputs and c is the tuning constant. This function has everywhere defined continuous derivatives except at the origin. The tuning constant, c is usually set as 1.3998.

3.2.6 Fourth Power Error Function

The Fourth Power Error is given by

$$E = \sum_n e_i^4 \quad (3.7)$$

where n is the number of outputs. This would be useful when dealing with data known to be free from outliers, or in cases where it is important to minimize the worst-case error, rather than the average error (Rey, 1983).

3.2.7 Geman-McClure Error Function

The Geman-McClure Error Function is given by

$$E = \sum_n \frac{e_i^2/2}{1+e_i^2} \quad (3.8)$$

where n is the number of outputs. The function tries to reduce the effect of large errors.

3.2.8 Huber Error Function

The Huber Error function is given by

$$E = \sum_n \begin{cases} e_i^2/2, & \text{if } |e_i| < c \\ c(|e_i| - c/2), & \text{if } |e_i| \geq c \end{cases} \quad (3.9)$$

(Huber, 1981) where n is the number of outputs and c is the tuning constant. A typical value for c is 1.345. When dealing with noisy data, the training values may contain outliers with unusual deviation from the true underlying function. Huber function can be used to ignore these outliers, or at least reduce the ill effect they have on learning. The function hence has good effects of both Quadratic and Absolute Error Functions.

3.2.9 Hyperbolic Squared Error Function

The Hyperbolic Squared error is given by

$$E = \sum_n \ln \left(\frac{1+e_i^2}{1-e_i^2} \right) \quad (3.10)$$

where n is the number of outputs. Hyperbolic Squared Error needs a normalization while running a training with one of the BPA or variants.

3.2.10 Log-Cosh Error Function

The Log-Cosh Error Function is given by

$$E = \sum_n \ln(\cosh(e_i^2)) \quad (3.11)$$

where n is the number of outputs.

3.2.11 Logarithmic Error Function

The Logarithmic Error Function is given by

$$E = \sum \left[(1 + y_d) \ln \left\{ \frac{(1 + y_d)}{(1 + y_i)} \right\} \right] + \sum \left[(1 - y_d) \ln \left\{ \frac{(1 - y_d)}{(1 - y_i)} \right\} \right] \quad (3.12)$$

where y_i is the calculated output using current weights and y_d is the desired output

3.2.12 Mean-Median Error Function

The Mean-Median Error Function is given by

$$E = \sum_n 2 * \left(\sqrt{(1 + e_i^2)/2} - 1 \right) \quad (3.13)$$

where n is the number of outputs. This takes the advantage of both the Mean Error function and the Median Error function. Hence, reduces the influence of large errors but at the same time retains its convexity.

3.2.13 Minkowski Error Function

The Minkowski Error Function is given by

$$E = \sum_n |e_i|^r \quad (3.14)$$

where n is the number of outputs and the typical value of r is chosen is 4.0.

3.2.14 Quadratic Error Function

The Mean Squared Error is given by

$$E = \sum_n e_i^2 \quad (3.15)$$

where n is the number of outputs. This is the standard error function (Bose and Liang, 1996).

3.2.15 Sinh Error Function

The Sine-Hyperbolic Error Function is given by

$$E = \sum_n (\text{Sinh}(e_i)) \quad (3.16)$$

where n is the number of outputs. The function is steeper than the Quadratic Error Function. Moreover the function is symmetric about the origin and hence the update involves two parts, the first is the gradient in the first quadrant while the second is gradient in the third quadrant. In both cases, the gradient is directed towards the origin.

3.2.16 Tukey Biweight Error Function

The Tukey Biweight function is given by

$$E = \sum_n \begin{cases} c^2/6 \left(1 - \left[1 - (e_i/c)^2 \right]^3 \right) & \text{if } |e_i| \leq c \\ c^2/6 & \text{if } |e_i| > c \end{cases} \quad (3.17)$$

Where n is the number of outputs and c is the tuning constant. The typical value of c is 4.6851. Tukey Biweight function reduces the effect of large errors and suppresses the outliers. The contribution of an outlier to this EF hence is smaller.

3.2.17 Welsch Error Function

The Welsch Error Function is given by

$$E = \sum_n \frac{c^2}{2} \left[1 - \exp\left(- (e_i/c)^2\right) \right] \quad (3.18)$$

where n is the number of outputs. This function reduces the influence of large errors. The typical value of the tuning constant $c = 2.9846$.

3.3 Complex Error Function

The following error functions were considered for the present study. The derivatives of these functions were computed to implement the basic update rule for training the Neural Network (eqn. (1.1)). In literature, these error functions were proposed to implement an m -estimators approach for bypassing or reducing the ill-effects of the outliers. The functions in each case have been generalized in a way as to retain the form and yet be operational in the complex variable setting. The surface plots of all the definitions are tabulated in Table 3.1.

3.3.1 Complex Absolute Error Function

The Complex Absolute Error is defined to be

$$E = \sum_n abs(\varepsilon_i) = \sum_n \sqrt{\varepsilon_i \varepsilon_i^*} \quad (3.19)$$

where n denotes the number of outputs.

3.3.2 Complex Andrew Error Function

The Complex Andrew Error Function is defined to be

$$E = \sum_n \begin{cases} 1/\pi^2 * \cos(\pi * abs(\varepsilon_i)); & \text{if } abs(\varepsilon_i) \leq 1 \\ 0; & \text{else} \end{cases} \quad (3.20)$$

and n is the number of outputs.

3.3.3 Complex Bipolar Hyperbolic Squared Error Function

The Complex Bipolar Hyperbolic Squared Error is defined to be

$$E = \sum_n \ln \left(\frac{2 - \varepsilon_i \varepsilon_i^*}{2 + \varepsilon_i \varepsilon_i^*} \right) \quad (3.21)$$

where, n is the number of outputs.

3.3.4 Complex Cauchy Error Function

The Complex Cauchy Error Function is defined to be

$$E = \sum_n \frac{c^2}{2} \ln \left(1 + \left(\varepsilon_i \varepsilon_i^* / c^2 \right) \right) \quad (3.22)$$

where n is the number of outputs and c is the tuning constant which was set to the Cauchy constant. The tuning constant for the EF is set equal to 2.3849.

3.3.5 Complex Fair Error Function

The Complex Fair Function is defined to be

$$E = \sum_n c^2 \left[\left(\text{abs}(\varepsilon_i) / c \right) - \ln \left(1 + \left(\text{abs}(\varepsilon_i) / c \right) \right) \right] \quad (3.23)$$

where n is the number of outputs and c is the tuning constant which was set to the Fair constant. The tuning constant, c is set to 1.3998.

3.3.6 Complex Fourth Power Error Function

The Complex Mean Fourth Power Error is defined to be

$$E = \sum_n \frac{1}{2} (\varepsilon_i \varepsilon_i^*)^2 \quad (3.24)$$

n is the number of outputs.

3.3.7 Complex Geman-McClure Error Function

The Complex Geman-McClure Error Function is defined to be

$$E = \sum_n \frac{\varepsilon_i \varepsilon_i^* / 2}{1 + \varepsilon_i \varepsilon_i^*} \quad (3.25)$$

where n is the number of outputs.

3.3.8 Complex Huber Error Function

The Complex Huber Error function is defined to be

$$E = \sum_n \begin{cases} \varepsilon_i \varepsilon_i^* / 2 & ; \text{if } |\varepsilon_i| < c \\ c(\text{abs}(\varepsilon_i) - c/2) & ; \text{if } |\varepsilon_i| \geq c \end{cases} \quad (3.26)$$

(Huber, 1981) where n is the number of outputs and c is the tuning constant. A typical value for c is 1.345.

3.3.9 Complex Hyperbolic Squared Error Function

The Complex Hyperbolic Squared Error is defined to be

$$E = \sum_n \ln \left(\frac{1 - \varepsilon_i \varepsilon_i^*}{1 + \varepsilon_i \varepsilon_i^*} \right) \quad (3.27)$$

where n is the number of outputs. Hyperbolic Squared Error is similar in shape to the Bipolar Hyperbolic Squared Error function.

3.3.10 Complex Logarithmic Error Function

The Complex Logarithmic Error Function is defined to be

$$E = \sum \left[(1 + y_d) \ln \left(1 + \frac{\text{abs}(\varepsilon_i)}{(1 + y_i)} \right) \right] + \sum \left[(1 - y_d) \ln \left(1 - \frac{\text{abs}(\varepsilon_i)}{(1 - y_i)} \right) \right] \quad (3.28)$$

where y_d is the desired output.

3.3.11 Complex Log-Cosh Error Function

The complex Log-Cosh Error Function is defined to be

$$E = \sum_n \ln(\cosh(\varepsilon_i \varepsilon_i^*)) \quad (3.29)$$

where i is the number of outputs.

3.3.12 Complex Mean-Median Error Function

The Complex Mean-Median Error Function is defined to be

$$E = \sum_n 2 * \left(\sqrt{(1 + \varepsilon_i \varepsilon_i^* / 2)} - 1 \right) \quad (3.30)$$

where n is the number of outputs.

3.3.13 Complex Minkowski Error Function

The Complex Minkowski Error Function is defined to be

$$E = \sum_n \text{abs}(\varepsilon_i)^r \quad (3.31)$$

where n is the number of outputs and the typical value of r was chosen to be 4.

3.3.14 Complex Quadratic Error Function

The Complex Quadratic Error is defined to be

$$E = \sum_n \frac{1}{2} \varepsilon_i \varepsilon_i^* \quad (3.32)$$

where n is the number of outputs. This is the standard error function.

3.3.15 Complex Sinh Error Function

The Complex Sine-Hyperbolic Error Function is defined to be

$$E = \sum_n (\text{Sinh}(\text{abs}(\varepsilon_i))) \quad (3.33)$$

where n is the number of outputs.

3.3.16 Complex Tukey Biweight Error Function

The Complex Tukey Biweight function is defined to be

$$E = \sum_n \begin{cases} c^2/6 \left(1 - \left[1 - (\varepsilon_i \varepsilon_i^* / c^2) \right]^6 \right); & \text{if } \text{abs}(\varepsilon_i) \leq c \\ c^2/6; & \text{if } \text{abs}(\varepsilon_i) > c \end{cases} \quad (3.34)$$

Where n is the number of outputs and c is the tuning constant. The constant c was set to the Tukey constant, 4.6851.

3.3.17 Complex Welsch Error Function

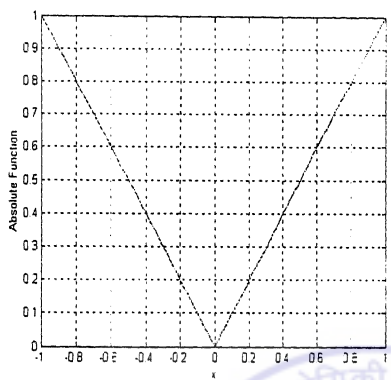
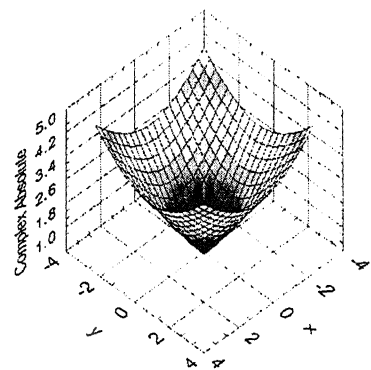
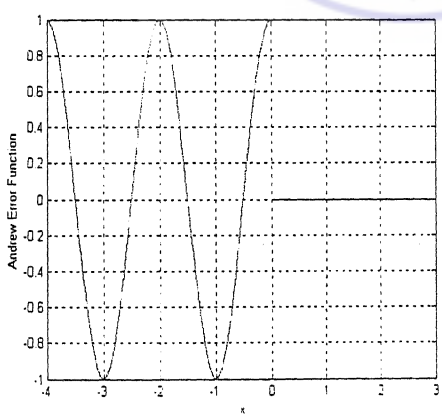
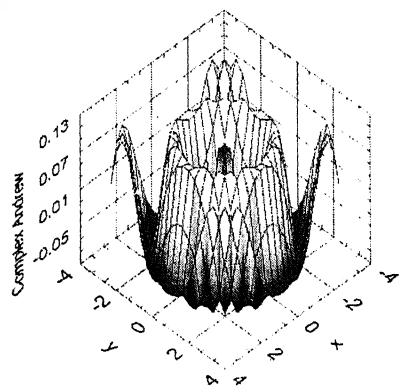
The Complex Welsch Error Function is defined to be

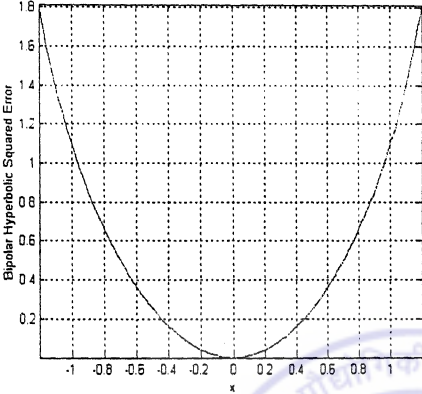
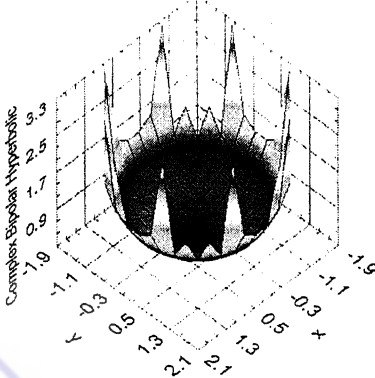
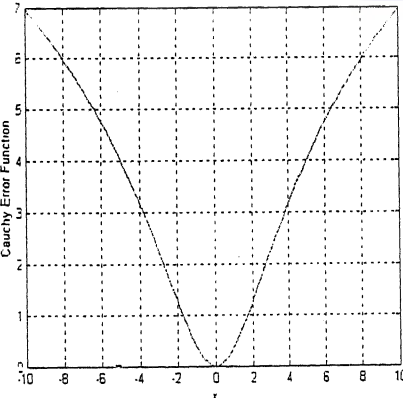
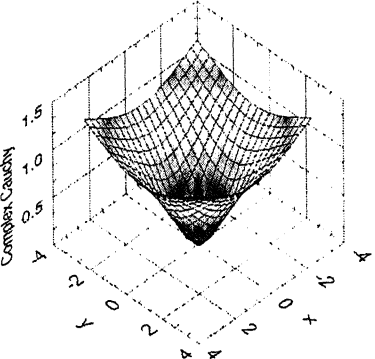
$$E = \sum_n \frac{c^2}{2} \left[1 - \exp\left(-(\varepsilon_i \varepsilon_i^* / c^2)\right) \right] \quad (3.35)$$

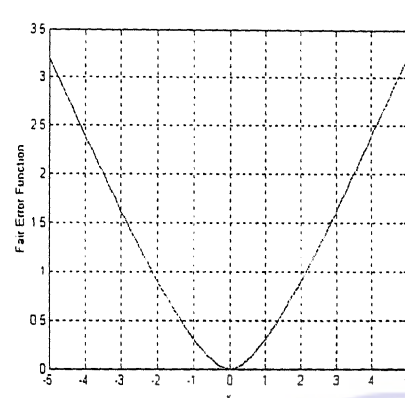
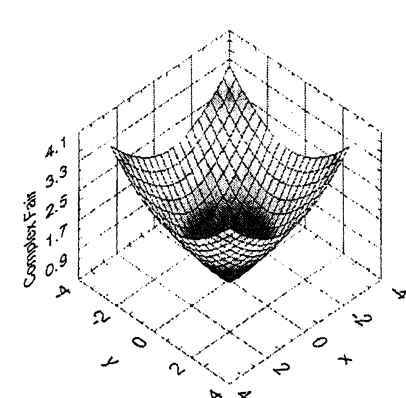
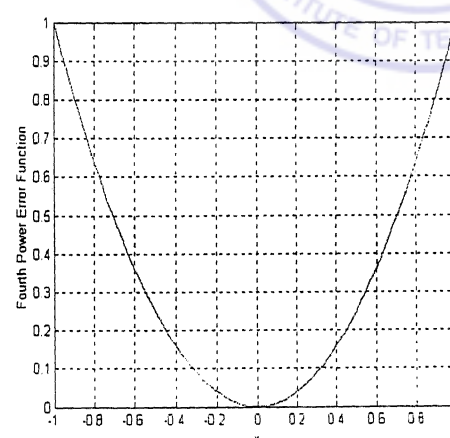
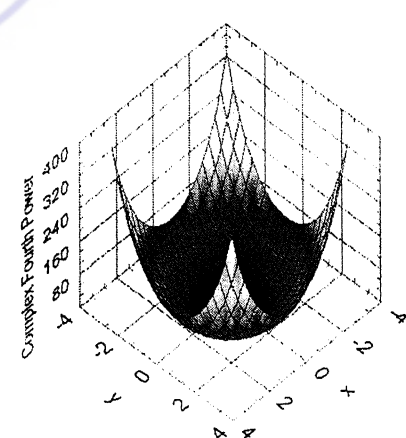
where n is the number of outputs. The tuning constant c was set to the Welsch constant, 2.9846.

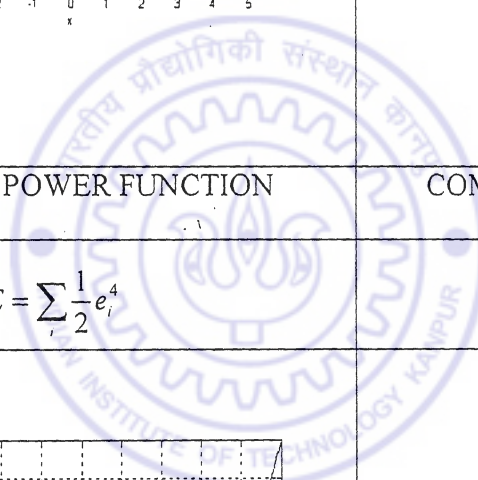
In these definitions, the function's form has been retained while extending to the complex domain. This was done to make sure that the error computed kept the same formula even while operating in the complex domain. This also makes sure that the surface plot of the function is close to the plane plot of the same. As a ramification of the fact, the additional factor that enters the weight update rule in the BPA and the CVBP has the same form. Table 1 lists the Error Functions, their definition and plane and surface graphs displaying their overall shape. It may be seen from the graphs that retaining the form of the function while generalizing preserves the shape of the plane curve.

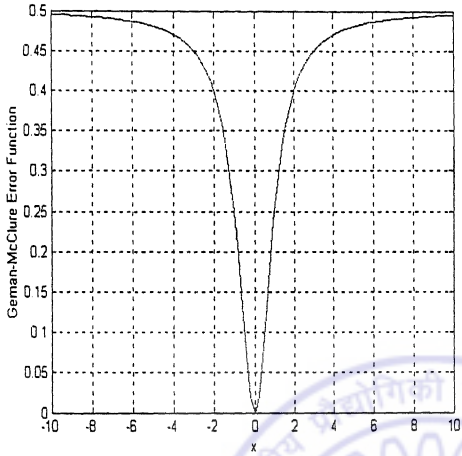
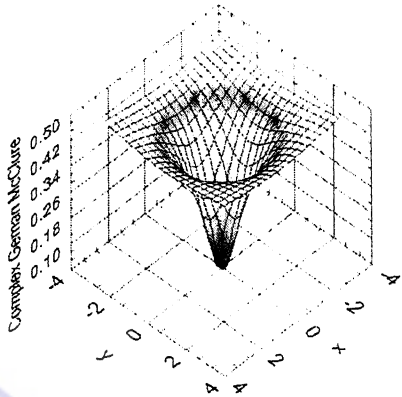
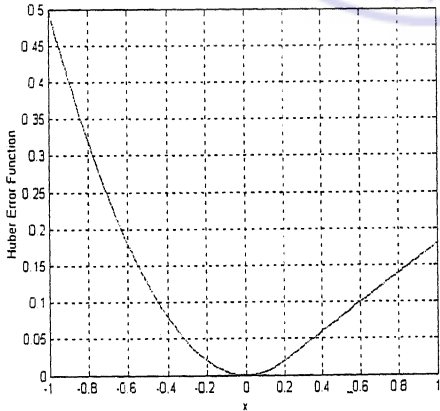
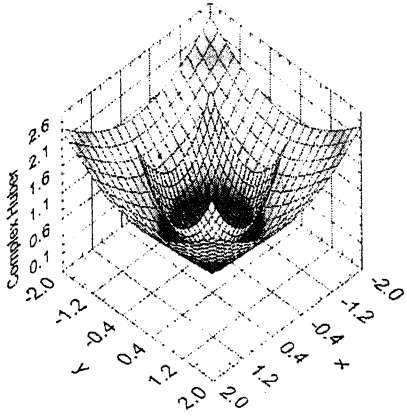
In sinh EF, the extended version doesn't keep the odd function property. The complex function developed from the old error function is an even function in the argument. The gradient in both cases set directing towards the origin but the same requires two steps in the real function case. In Table 3.1, the x and y labeled in the surface plots (complex EF) must be interpreted as the real and imaginary parts of ε_i , which in turn is $T_i - O_i$ (the target value minus the output of the network.)

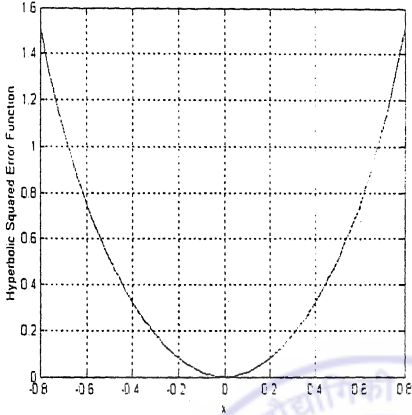
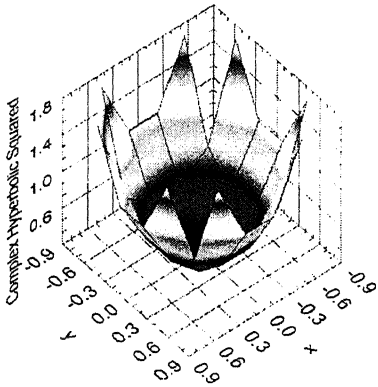
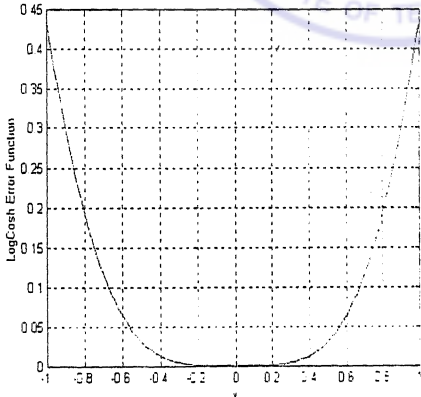
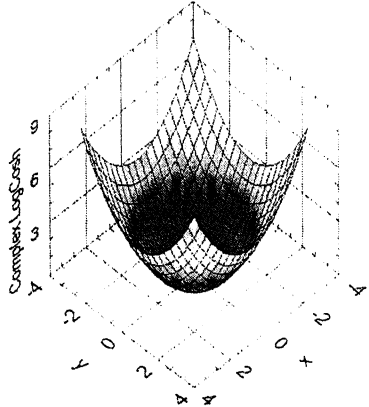
| | ABSOLUTE ERROR FUNCTION | COMPLEX ABSOLUTE ERROR FUNCTION |
|-------|---|---|
| Def | $E = \sum e_i $ | $E = \sum abs(\epsilon_i) = \sum \sqrt{\epsilon_i \epsilon_i^*}$ |
| Plots |  <p>A 2D line plot of the Absolute Error Function. The x-axis ranges from -1 to 1 with increments of 0.2. The y-axis, labeled 'Absolute F. function', ranges from 0 to 1 with increments of 0.1. The plot shows a V-shaped curve that starts at (-1, 1), goes down to (0, 0), and goes back up to (1, 1).</p> |  <p>A 3D surface plot of the Complex Absolute Error Function. The x and y axes both range from -4 to 4 with increments of 2. The z-axis, labeled 'Complex Absolute', ranges from 0 to 1.5 with increments of 0.2. The surface is a smooth, bowl-shaped curve that reaches its minimum value of 0 at the origin (0,0,0) and rises to approximately 1.5 at the corners of the plot.</p> |
| | ANDREW ERROR FUNCTION | COMPLEX ANDREW ERROR FUNCTION |
| Def | $E = \sum_i \begin{cases} 1/\pi^2 * \cos(\pi * e_i); & \text{if } e_i \leq 1 \\ 0; & \text{else} \end{cases}$ | $E = \sum_i \begin{cases} 1/\pi^2 * \cos(\pi * abs(\epsilon_i)); & \text{if } abs(\epsilon_i) \leq 1 \\ 0; & \text{else} \end{cases}$ |
| Plots |  <p>A 2D line plot of the Andrew Error Function. The x-axis ranges from -4 to 3 with increments of 1. The y-axis, labeled 'Andrew Error Function', ranges from -1 to 1 with increments of 0.2. The plot shows a periodic wave that oscillates between -1 and 1 for x values between -1 and 1, and is zero elsewhere.</p> |  <p>A 3D surface plot of the Complex Andrew Error Function. The x and y axes both range from -4 to 4 with increments of 2. The z-axis, labeled 'Complex Andrew', ranges from -0.05 to 0.13 with increments of 0.02. The surface shows a complex oscillating pattern with a central peak and trough, and smaller oscillations towards the edges.</p> |

| | BIPOLAR HYPERBOLIC ERROR SQUARED FUNCTION | COMPLEX BIPOLAR HYPERBOLIC SQUARED FUNCTION |
|-------|---|---|
| Def | $E = \sum_i \ln \left(\frac{2 - e_i^2}{2 + e_i^2} \right)$ | $E = \sum_i \ln \left(\frac{2 - \varepsilon_i \varepsilon_i^*}{2 + \varepsilon_i \varepsilon_i^*} \right)$ |
| Plots |  <p>A 2D line plot showing the Bipolar Hyperbolic Squared Error. The x-axis ranges from -1 to 1 with increments of 0.2. The y-axis ranges from 0 to 1.8 with increments of 0.2. The curve is symmetric about the y-axis, starting at approximately 1.75 at x = -1, reaching a minimum of 0 at x = 0, and returning to approximately 1.75 at x = 1.</p> |  <p>A 3D surface plot of the Complex Bipolar Hyperbolic Squared Error. The x and y axes range from -2.1 to 2.1 with increments of 0.5. The z-axis ranges from 0 to 3.3 with increments of 0.8. The surface is a bowl-shaped paraboloid opening upwards, with its minimum at the origin (0,0,0) and rising to approximately 3.3 at the corners of the x-y plane.</p> |
| | CAUCHY ERROR FUNCTION | COMPLEX CAUCHY ERROR FUNCTION |
| Def | $E = \sum_i \frac{c^2}{2} \ln \left(1 + (e_i / c)^2 \right)$ | $E = \sum_i \frac{c^2}{2} \ln \left(1 + (\varepsilon_i \varepsilon_i^* / c^2) \right)$ |
| Plots |  <p>A 2D line plot of the Cauchy Error Function. The x-axis ranges from -10 to 10 with increments of 2. The y-axis ranges from 0 to 7 with increments of 1. The curve is symmetric about the y-axis, starting at approximately 6.5 at x = -10, reaching a minimum of 0 at x = 0, and rising to approximately 6.5 at x = 10.</p> |  <p>A 3D surface plot of the Complex Cauchy Error Function. The x and y axes range from -4 to 4 with increments of 2. The z-axis ranges from 0 to 1.5 with increments of 0.5. The surface is a bowl-shaped paraboloid opening upwards, with its minimum at the origin (0,0,0) and rising to approximately 1.5 at the corners of the x-y plane.</p> |

| | FAIR ERROR FUNCTION | COMPLEX FAIR ERROR FUNCTION |
|-------|--|--|
| Def | $E = \sum_i c^2 \left[\left(\frac{e_i}{c} \right) - \ln \left(1 + \left(\frac{e_i}{c} \right) \right) \right]$ | $E = \sum_i c^2 \left[\left(\frac{abs(\varepsilon_i)}{c} \right) - \ln \left(1 + \left(\frac{abs(\varepsilon_i)}{c} \right) \right) \right]$ |
| Plots |  |  |
| | FOURTH POWER FUNCTION | COMPLEX FOURTH POWER FUNCTION |
| Def | $E = \sum_i \frac{1}{2} e_i^4$ | $E = \sum_i \frac{1}{2} (\varepsilon_i \varepsilon_i^*)^2$ |
| Plots |  |  |



| | GEMAN-McCLURE ERROR FUNCTION | COMPLEX GEMAN-McCLURE ERROR FUNCTION |
|-------|---|---|
| Def | $E = \sum_i \frac{e_i^2/2}{1+e_i^2}$ | $E = \sum_i \frac{\varepsilon_i \varepsilon_i^* / 2}{1 + \varepsilon_i \varepsilon_i^*}$ |
| Plots |  <p>A 2D line plot showing the Geman-McClure Error Function. The x-axis ranges from -10 to 10 with major ticks every 2 units. The y-axis ranges from 0 to 0.5 with major ticks every 0.05 units. The curve is symmetric about the y-axis, starting at 0.5 for x = -10, dipping to a minimum of 0 at x = 0, and rising back to 0.5 at x = 10. A grid is overlaid on the plot.</p> |  <p>A 3D surface plot of the Complex Geman-McClure Error Function. The vertical z-axis is labeled 'Complex Geman-McClure' and ranges from 0 to 0.50 with ticks at 0.10, 0.18, 0.26, 0.34, 0.42, and 0.50. The horizontal x and y axes both range from -4 to 4 with ticks at -4, -2, 0, 2, 4. The surface is a smooth, symmetric bowl shape centered at the origin (0,0,0), with a maximum value of 0.50 at the center. A grid is visible on the base of the plot.</p> |
| | HUBER ERROR FUNCTION | COMPLEX HUBER ERROR FUNCTION |
| Def | $E = \sum_i \begin{cases} e_i^2/2 & : \text{if } e_i < c \\ c(e_i - c/2) & : \text{if } e_i \geq c \end{cases}$ | $E = \sum_i \begin{cases} \varepsilon_i \varepsilon_i^* / 2 & : \text{if } \varepsilon_i < c \\ c(\text{abs}(\varepsilon_i) - c/2) & : \text{if } \varepsilon_i \geq c \end{cases}$ |
| Plots |  <p>A 2D line plot of the Huber Error Function. The x-axis ranges from -1 to 1 with ticks every 0.2 units. The y-axis ranges from 0 to 0.5 with ticks every 0.05 units. The curve is symmetric about the y-axis, starting at approximately 0.48 at x = -1, reaching a minimum of 0 at x = 0, and ending at approximately 0.18 at x = 1. A grid is overlaid on the plot.</p> |  <p>A 3D surface plot of the Complex Huber Error Function. The vertical z-axis is labeled 'Complex Huber' and ranges from 0 to 2.6 with ticks at 0.1, 0.6, 1.1, 1.6, 2.1, and 2.6. The horizontal x and y axes both range from -2.0 to 2.0 with ticks at -2.0, -1.2, -0.4, 0.4, 1.2, and 2.0. The surface is a smooth, symmetric bowl shape centered at the origin (0,0,0), with a maximum value of 2.6 at the center. A grid is visible on the base of the plot.</p> |

| | HYPERBOLIC SQUARED ERROR FUNCTION | COMPLEX HYPERBOLIC SQUARED ERROR |
|-------|--|--|
| Def | $E = \sum_i \ln \left(\frac{1 - e_i^2}{1 + e_i^2} \right)$ | $E = \sum_i \ln \left(\frac{1 - \varepsilon_i \varepsilon_i^*}{1 + \varepsilon_i \varepsilon_i^*} \right)$ |
| Plots |  <p>A 2D line plot showing the Hyperbolic Squared Error Function. The x-axis ranges from -0.8 to 0.8 with increments of 0.2. The y-axis ranges from 0 to 1.6 with increments of 0.2. The curve is symmetric about the y-axis, starting at approximately 1.55 at x = -0.8, reaching a minimum of 0 at x = 0, and rising back to approximately 1.55 at x = 0.8.</p> |  <p>A 3D surface plot of the Complex Hyperbolic Squared Error function. The x and y axes both range from -0.9 to 0.9 with increments of 0.3. The z-axis, labeled 'Complex Hyperbolic Squared', ranges from 0 to 1.8 with increments of 0.2. The surface is a symmetric, bowl-like shape with a central dip to 0 at (0,0) and rising to a peak of approximately 1.7 at the corners of the plot.</p> |
| | LOG COSH ERROR FUNCTION | COMPLEX LOG COSH ERROR FUNCTION |
| Def | $E = \sum_i \ln(\cosh(e_i^2))$ | $E = \sum_i \ln(\cosh(\varepsilon_i \varepsilon_i^*))$ |
| Plots |  <p>A 2D line plot of the LogCosh Error Function. The x-axis ranges from -1 to 1 with increments of 0.2. The y-axis ranges from 0 to 0.45 with increments of 0.05. The curve is symmetric about the y-axis, starting at approximately 0.4 at x = -1, reaching a minimum of 0 at x = 0, and rising to approximately 0.4 at x = 1.</p> |  <p>A 3D surface plot of the Complex LogCosh Error function. The x and y axes both range from -4 to 4 with increments of 2. The z-axis, labeled 'Complex LogCosh', ranges from 0 to 9 with increments of 3. The surface is a symmetric, bowl-like shape with a central dip to 0 at (0,0) and rising to a peak of approximately 8.5 at the corners of the plot.</p> |

| | LOGARITHMIC ERROR FUNCTION | COMPLEX LOGARITHMIC ERROR FUNCTION |
|-------|---|---|
| Def | $E = \sum \left[(1+y_i) \ln \left(\frac{1+y_i}{1+y_i} \right) \right] + \sum \left[(1-y_i) \ln \left(\frac{1-y_i}{1-y_i} \right) \right]$ | $E = \sum \left[(1+y_i) \ln \left(1 + \frac{abs(\epsilon_i)}{(1+y_i)} \right) \right] + \sum \left[(1-y_i) \ln \left(1 - \frac{abs(\epsilon_i)}{(1-y_i)} \right) \right]$ |
| Plots | | |
| | MEAN-MEDIAN ERROR FUNCTION | COMPLEX MEAN-MEDIAN ERROR FUNCTION |
| Def | $E = \sum_i 2 * \left(\sqrt{1 + \epsilon_i^2 / 2} - 1 \right)$ | $E = \sum_i 2 * \left(\sqrt{1 + \epsilon_i \epsilon_i^* / 2} - 1 \right)$ |
| Plots | | |

MINKOWSKI ERROR FUNCTION

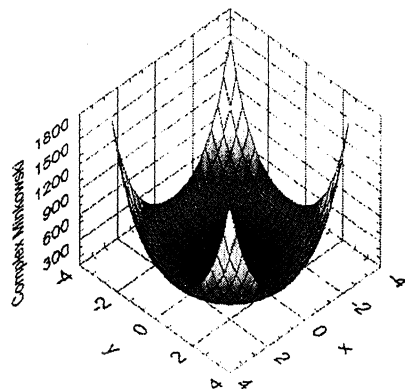
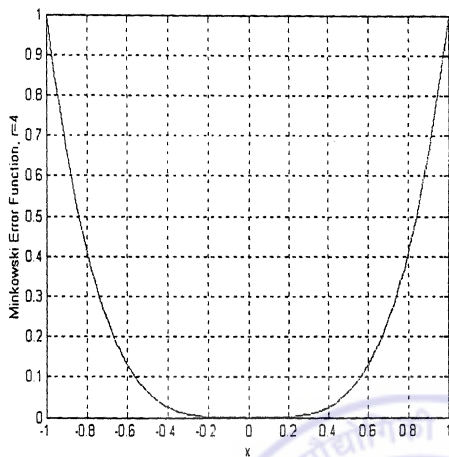
COMPLEX MINKOWSKI ERROR FUNCTION

Def

$$E = \sum_i |e_i|^r$$

$$E = \sum_i \text{abs}(\epsilon_i)^r$$

Plots



QUADRATIC ERROR FUNCTION

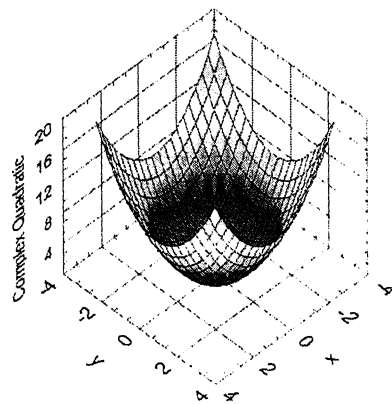
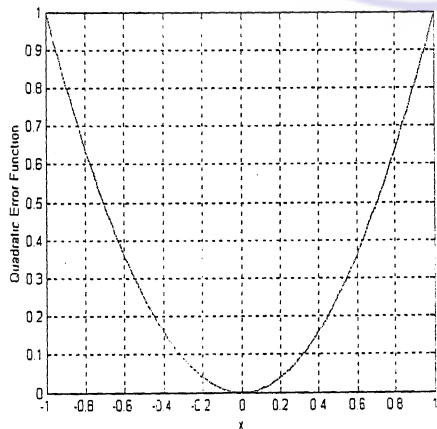
COMPLEX QUADRATIC ERROR FUNCTION

Def

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

$$E = \sum_i \frac{1}{2} \epsilon_i \epsilon_i^*$$

Plots



SINH ERROR FUNCTION

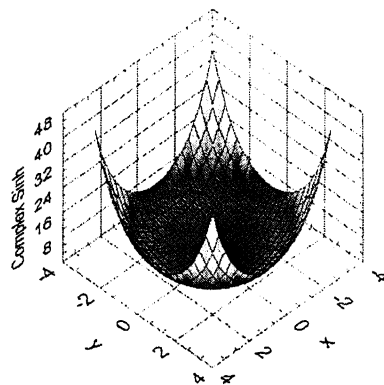
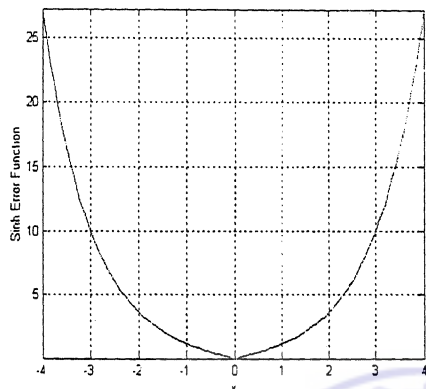
COMPLEX SINH ERROR FUNCTION

Def

$$E = \sum_i (\text{Sinh}(e_i))$$

$$E = \sum_i (\text{Sinh}(\text{abs}(e_i)))$$

Plots



TUKEY ERROR FUNCTION

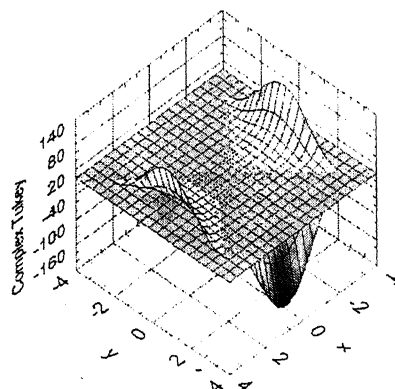
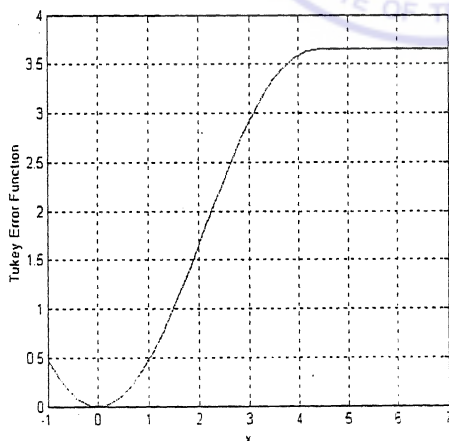
COMPLEX TUKEY ERROR FUNCTION

Def

$$E = \sum_i \begin{cases} c^2/6 \left(1 - \left[1 - \left(\frac{e_i}{c} \right)^2 \right]^3 \right) & \text{if } |e_i| \leq c \\ c^2/6 & \text{if } |e_i| > c \end{cases}$$

$$E = \sum_i \begin{cases} c^2/6 \left(1 - \left[1 - \left(\frac{e_i}{c} \right)^2 \right]^3 \right) & \text{if } \text{abs}(e_i) \leq c \\ c^2/6 & \text{if } \text{abs}(e_i) > c \end{cases}$$

Plots



| | WELSCH ERROR FUNCTION | COMPLEX WELSCH ERROR FUNCTION |
|-------|---|--|
| Def | $E = \sum_i \frac{c^2}{2} [1 - \exp(-(e_i/c)^2)]$ | $E = \sum_i \frac{c^2}{2} [1 - \exp(-(\varepsilon_i \varepsilon_i^* / c^2))] $ |
| Plots | | |

3.4 Properties of Error Functions

The properties of the various Error Functions are summarized here. The seventeen Error Functions employed in statistical analysis have been used for developing the BPA and the CVBP. The update rule of the BPA demands the EF be at least once differentiable from which it becomes clear that the functions be at least C^1 . Finitely many discontinuities or countably many of them can always be bypassed by defining the update rule accordingly by breaking the real line into finitely many or countably many intervals and developing a form of the update rule in each of the intervals separately. Of the many parameters that should be set for running the algorithm, the weights, biases, architecture, must be kept fixed to study the influence of the Error Function while EF based training algorithms run. Each of the EFs has its unique properties that statistical methods cash on while implementing them. These extended error functions were used to develop the CVBP for training CNN. A comparative study was later done to see how the ANN and CNN performed when EF was varied.

The Absolute Error Function is continuous through out the real line and is differentiable at all points on the line except the origin. As the real line is partitioned into two disconnected sets by the origin (the only point where the function is not differentiable),

the update rule has a two-step definition – when the error is positive and when the error is negative. The absence of an index (the power, unlike the Quadratic EF) is a distinguishing feature of this EF as this enables smoothing out the ill-effects of the outlier points that would otherwise have offset the best-fit of the optimisation scheme. The contribution to the EF from the outlier points would be on the same scale as the actual data points of the problem and hence the ill-effects due to spurious points are nullified to a great extent. On the other hand if the data were normalised to a specific region so that all the entries in the data set are small real numbers lying in $[-1,1]$, the contribution from the outliers is once again on the same scale as the actual data points. The gradient for both parts in the definition in the update rule is directed towards the origin. The complex Absolute Error Function is a generalisation of the Absolute Function defined to compute errors using the absolute function yet retaining the functional form. The definition is a surface as both real and imaginary parts are involved in it. It can also be noted that the function form in fact is the quadric, cone. It has all the complex weights of the CNN in its definition. The update rule for the CNN steers the real part and the imaginary part of the weights to the minima separately. The problem of local minima that existed in the ANN repeats while studying the CNN in general and this EF based algorithm in particular. As is clear the initial weight and the learning parameter decide how the training should progress. The dynamics of the real part depends not only on the real parts of the weights but also on the imaginary parts as the updates of the real and imaginary parts are dependent on each other (are coupled. Appendix 1). The complex EF is not differentiable at the origin as the function inside the radical is always positive while the function as a whole is not differentiable at zero.

The Andrew Error Function is characterized by a discontinuity at the origin and two continuous functions on either side of the discontinuity. The design of the function is basically to nullify the effect due to one kind of data - here in the definition, the data suppressed by the function are those that are positive valued. The definition also shows a periodic function on the other half characterized by several maxima and minima. Hence to implement the function, the network was set to drive the training in the direction of the negative gradient towards one of the several minima. It can be easily shown that the points of maxima of the sine function are in fact points of unstable equilibrium. Hence the training process steers the weights in a way as to reach one of the minima points. The point beyond which the function is suppressed can be chosen as desired while the

dynamics of update operate according to a sinusoid by definition. The extension of the Andrew Function to the complex variables was made retaining the functional form but extending the definition to take complex values. The surface plot of the complex definition reveals that the function is rotationally symmetric about the z -axis. While implementing the function, the training was directed towards the minima that are in the present form, concentric circles lying on the plane $z = \frac{1}{\pi^2}$.

The Bipolar Hyperbolic Squared Error Function was so defined because the error as computed from the network can take values in the interval $[-1,1]$, and hence for this error function, a bipolar sigmoid activation function can be employed (as it assumes values in the positive and negative quadrants). The function is characterized by unique maxima and hence the training process should steer the network so as to attain the maxima of the function. While implementing, the sign of the function is reversed so the update runs in accordance with accepted conventions. The complex function was defined by retaining the form of the function in the real setting but extending to accommodate complex errors. The surface is characterized by a unique maximum and rotational symmetry. The surface is differentiable with respect to real and imaginary parts of the complex variable that now appear as argument of the function.

The Cauchy Error Function has one minimum point at the origin. The function is symmetric about the y -axis. The training steers the weights so as to reach the minimum of the function. The function is defined through out the real line, its continuous everywhere and differentiable all through. The function changes convexity as x increases. As a ramification of this fact, the update from the slope function based on this error function at larger values of x would be smaller in comparison with that of the Quadratic Derror Function. The complex Cauchy Error Function was defined to perform the Cauchy function for the complex variables. The surface plot reveals a unique point of global minimum. The surface is differentiable through out the real plane. The surface is also characterized by changing convexity as the radius vector increases. It is rotationally symmetric about the z – axis.

The Fair Error Function has an absolute function in the definition, which makes the Fair Function continuous at all points on the real line. It is differentiable at all points except

the origin. The function remains convex with respect to the x – axis thorough out the real line. The parameter c in the definition only alters the shape of the bell by making it wider or narrower as it varies. The complex Fair Error Function is rotationally symmetric, has one global minimum (as a function of the real and imaginary parts of the complex argument). The convexity with respect to the xy – plane is maintained all through.

The Fourth Power Error Function is smooth over the whole real line. Unlike Quadratic, the function rises rapidly and is more convex with respect to the x – axis than the Quadratic Error Function. The weight update is more rapid for error values greater than unity and the rate of training is diminished for fractional errors, lying in the interval $[0,1]$. The contribution from the cube term that results from the form of the error function is hence a parameter that enhances the update if the error is greater than unity and suppresses it if then error is fractional. The complex Fourth Power Error Function keeps the form of the real error function but is defined so as to be able to operate with complex errors. The error function unlike the complex Quadratic Function increases more rapidly for errors more than unity. The surface is rotationally symmetric about the z – axis. The surface is smooth for the derivatives of all orders exist.

The Geman-McClure Error Function was defined to suppress large errors on either side of the origin. The asymptotes of the function suppress the outliers. For smaller values of the error the function approximates to the Quadratic Function as the denominator can be approximated as unity. The function is symmetric about the y –axis. The extension to the complex domain retains the form of the function. The principle of the function hence is carried over to the complex domain – for smaller values of the complex errors, the function is just the Quadratic Function, while for the large values the denominator comes into play and the function deviates from being Quadratic.

The Huber Error Function is defined piece-wise. The characteristic feature of the functions is it involves the Quadratic Error on the one hand and an Absolute Error on the other. The parameter, c in the definition is the point of demarcation to assign a domain of operation for each error function. The definition has both features – of the Quadratic Error and the Absolute Error. The function enables one to optimally choose error functions. If the data were prone to outliers and if their scatter is biased to one side, an obvious choice would be to suppress the influence of these spurious points by assigning

an Absolute Error Function to the side and set Quadratic Function to operate on the other side. It was found that in statistical analysis such choice indeed bettered the results as a judicious assignment was proven to be effective. The complex Huber function was defined to generalize the real Huber Function to the complex variables retaining the form of the function. The principle of operation of the function remains the same as the choice of the parameter c assigns the domains of operation of the Error Function. The Quadratic and the Absolute functions exist in the definition and the assignment will accordingly suppress the ill-effects of the outliers and other spurious data by evaluating the assigned function accordingly. It must be noted that the extended function is a paraboloid of revolution for the part of the definition that was Quadratic and for the part that was Absolute function, the extended version is a cone.

The Hyperbolic Squared Error Function has normalized errors for otherwise the argument of the function becomes negative and the function outputs complex-values. The function is symmetric about the y -axis and has a unique maxima point (at the origin). The training procedure was developed to steer the weights so as to enable the network to reach the tip of the function. The extension of the Hyperbolic Squared Error Function to the complex variables results in the retains the functional form but was extended to take complex numbers. The functional form is retained but the argument is modified to operate with complex numbers. The surface is characterized by a unique maximum at the origin to which the training process should steer the network error to. While implementing the function a negative was prefixed to the error surface and the usual gradient descent was developed for the function.

The logarithmic cosine error function is just the Quadratic Error Function for large values of the error as the hyperbolic cosine can be approximated by the exponential function for large values of the argument. For small errors near the origin however the function rises slower than the Quadratic. The function finds application at places where small errors should be suppressed while large errors should be treated with a quadratic function. The function is convex through out the real line and always lies below the Quadratic function. The complex version of the Logarithmic Hyperbolic Cosine function has one global minimum in the definition. The function is convex with respect to the xy - plane. For smaller values of complex error the training would be at a lower rate than the conventional Quadratic function owing to the low slope of the error function in the

vicinity of the origin. For larger error values the function behaves like the Quadratic Error Function.

The Logarithmic Error Function shows a unique minimum. Its convex with respect to the x – axis. The function maintains its curvature through out the real line. The complex version of the function has a unique maximum. The training algorithm was developed over the negative of this function, steering the network weights to the maximum point.

The Mean-Median Error Function behaves like Absolute Function for large values of the error. The function is convex through out the x – axis. For smaller error values the function behaves like Quadratic Error Function. The function finds best application for data that are prone to an outlier scatter that should be treated by assigning a function to nullify the ill effects due to them. Since the Mean-Median Error Function behaves like Absolute function that allots less cost that the conventional Quadratic function would have, the solution obtained by this function would be better than the one obtained by employing Quadratic function. The complex version of the Mean-Median Error function carries the above-described features forward to the complex plane. The extended function hence, behaves like the Quadratic Error Function for smaller values of the complex error. For complex errors large, the function is remnant of the Absolute Error.

The Minkowski Error Function is characterized by a parameter that appears as the index in the definition. The additional feature of the function makes it wider than it actually is. in that the other standard functions studied thus far (like Absolute Error, Fourth Power Function, Quadratic Error) can be obtained as particular cases of this function (by accordingly setting the index). For even indices the Minkowski Function behaves like the Quadratic function typically (for all others in this class have similar properties like convexity towards x – axis, symmetry about the y – axis). The odd indices generate functions that need a piece-wise defined update rule for the function would be symmetric about the origin in this case. The complex Minkowski Error Function keeps the form of the actual real valued function. The Sinh Error Function is symmetric about the origin. The computation of slope should be directed towards the origin in the first and the third quadrants separately, for which the update rule must be defined in these quadrants accordingly. The function is smooth and maintains convexity through out the x – axis. The slope computed from this EF varies according to $\cosh(x)$.

The complex Sinh EF extends the Hyperbolic Sine function to the complex domain. A complex conjugation was employed in the argument to the function that makes the extended complex function an even function. This is rotationally symmetric about the z – axis. The surface maintains convexity with respect to xy – plane. The steep-ness of the slope increases as the index rises.

The Tukey EF is continuous through out the x – axis. The function suppresses the affect of outliers that lie beyond a fixed value. c . On the other half, Tukey is a polynomial to the power six. It maintains the convexity with respect to the x – axis but gets concave at the point $(c, 0)$. The complex Tukey has a similar definition that retains the structure of the function but extends the domain of operation to the complex variables. The surface is convex with respect to the xy – plane for small values of the error and changes convexity once before restoring back to convex. The weight update hence depends on the part of the surface at which the error vector lies. The rest of the surface is the plane. The construction retains all the properties of the real Tukey EF.

The Welsch EF is designed to suppress large errors and give a Quadratic function like performance in the vicinity of the origin. The function is convex with respect to x – axis near the origin and has the asymptote $y = \frac{c^2}{2}$. The complex version of the function retains the form but operates with complex errors. Convexity prevails near the origin. the plane $z = \frac{c^2}{2}$ is an asymptote to the surface. The function is designed to suppress large errors and give an Quadratic function like performance for small errors.

3.5 Benchmark Problems

The Error Function based Neural Networks were applied to the Benchmark Problems (Dagli, 1994). In the following, a brief statement of these problems and their significance is presented.

The problems and their data come from different fields. As a consequence, nothing about the data is known, in fact the order of differential equations governing the system, the parameters and their effective ranges are varied and it becomes extremely difficult to

compare the performance of Neural Networks and their training algorithms. To estimate the strength of the Neural Network and to benchmark the effectiveness of Neural Networks to solving different types of problems arising from various fields, researchers have identified Benchmark Problems (Dagli, 1994). They represent problems that offered a challenge to Neural Network algorithms, thereby contributed to the development of the field. Any new algorithm in the area of CNN must hence be directed toward the Benchmarks to study how it performs as compared with the already existing ones.

3.5.1 The XOR Problem

In 1969, Minsky and Papert performed a series of experiments to ascertain the Neural Network's ability to capture input-output maps. Rumelhart, Hinton, Williams (1986) studied the question of learning representation in multi-layered Neural Networks. Mapping problems are one class of application of Neural Networks (as opposed to problems of classification where the Neural Network is trained to capture many-to-one maps). The study of applying the Neural Network algorithms for training nets to map the logical gates is important from the viewpoint of hardware development. It is well known that the NAND and NOR gates are universal approximators and if the Neural Networks could be made to perform these gates, any logic could be built and implemented using a Neural Network based hardware design. The study of Minsky and Papert led them to obtain Neural Network designs for the standard problems of AND, OR, NAND, NOR. The results showed that a linear decision boundary is all that was required to perform these mappings. It was also observed in the study that the XOR problem was significant as the solution involved a non-linear region to separate out the outputs. The mapping of an XOR problem is considered as an example in the present study. The question of whether Networks capture a given logic (posed as a mapping problem by developing a truth table of the logic) can hence be equivalently stated in terms of the Network's ability to capture NAND and NOR gates as any logic can be expressed in terms of these. The following diagrams indicate how the mapping of the OR can be solved by a linear decision boundary. Fig (b) shows the XOR mapping that requires a non-linear boundary to demarcate regions as belonging to different classes. The extension of XOR as was presented in Nitta (1997), is

- (i) The real part of the output is unity if the first input is equal to the second input else it is zero

- (ii) The imaginary part of the output is unity if the second input is equal to unity else it is zero.

Taking the extended definition, the truth table for the Complex XOR (CXOR) taken as the training set is the first set of eight patterns in Table 3.2. The rest of the combinations make the test set but the simulation was made over the whole sixteen patterns. The designed network however was tested with all the sixteen patterns. A 1-5-1 CNN was considered with the CXOR problem. The same map was also solved by using a 2-9-2 EF based ANN. In addition, the XOR map was solved using the EF based ANN as well. The learning rate was set equal to 0.1, the target error was set to 0.0001. The experiments were run thrice and averaged values of epochs for convergence have been tabulated. The mapping performance of EF based CNN is shown in Table 3.4. As can be observed from Table 3.3, the CNN solved the CXOR problem more efficiently requiring lesser epochs than the ANN. On the other hand, the standard ANN needed more epochs to solve XOR than CNN required to solve CXOR.

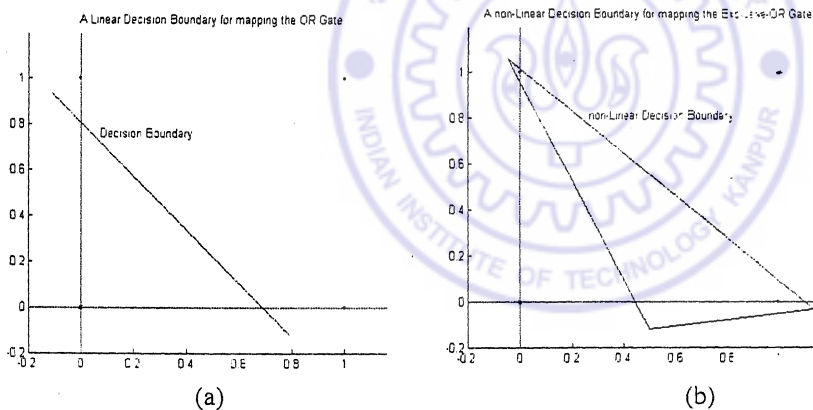


Fig. 3.1 (a) OR Gate showing a linear Decision Boundary (b) XOR Gate with a possible non-linear Decision Boundary

Table 3.2 Complex Exclusive-OR

| Serial Number | Input 1 | Input 2 | Output |
|---------------|---------|---------|--------|
| 1 | 0 | 0 | 1 |
| 2 | 0 | i | i |
| 3 | i | 0 | 0 |
| 4 | i | i | $1-i$ |
| 5 | i | 1 | i |
| 6 | 1 | 1 | $1-i$ |
| 7 | $1-i$ | i | i |
| 8 | $1-i$ | $1+i$ | 1 |
| 9 | 0 | 1 | i |
| 10 | 0 | $1+i$ | 0 |
| 11 | i | $1-i$ | 0 |
| 12 | 1 | 0 | 0 |
| 13 | 1 | $0+i$ | i |
| 14 | 1 | $1+i$ | 0 |
| 15 | $1-i$ | 0 | 0 |
| 16 | $1+i$ | 1 | i |

3.5.2 The N-Parity Problem

The problem is to train the Neural Network to differentiate between odd and even parity in the n – bit binary number. The problem was first reported by Minsky and Papert (1969) and further work was done by Rumelhart (1986), Fahlman (1988), Tesauro and Jansens (1988). The problem involves capturing the input-output map is assigned to words with even parity and a different output assigned to words with odd parity. In the complex variable setting, the outputs assigned were unity for even parity and the imaginary unit, i for odd parity.

Table 3.3 Epochs for CXOR and XOR Problems solved using CNN, ANN

| Average epochs with Real and Complex Error Functions. Learning rate was 0.1. Target Error was 0.0001 | | | |
|--|--|----------------------------|------------------------------------|
| Error Function | CXOR with Real BPA (architecture: 4-9-2) | CXOR (architecture: 2-5-1) | XOR with BPA (architecture: 2-5-1) |
| Absolute Error | 21500 | 12100 | 22500 |
| Andrew Error | 29500 | 20300 | 33300 |
| Cauchy Error | 18300 | 12900 | 32000 |
| Error Fourth Order | 25100 | 11000 | 33000 |
| Fair Error | 20100 | 15100 | 20000 |
| Geman-McClure Error | 27500 | 21000 | 29000 |
| Huber Error | 25100 | 10000 | 15000 |
| Hyperbolic Squared | 28100 | 23500 | 20600 |
| Bipolar Hyperbolic | 27200 | 20200 | 32500 |
| LogCosh Error | 15100 | 12000 | 23300 |
| Logarithmic Error | 24900 | 20000 | 31000 |
| Mean Median Error | 15200 | 12000 | 16000 |
| Minkowski Error | 16000 | 12100 | 33000 |
| Quadratic Error | 29500 | 10000 | 17600 |
| Sinh Error | 17100 | 14200 | 20000 |
| Tukey Error | 21900 | 18100 | 25000 |
| Welsch Error | 25000 | 21000 | 33000 |

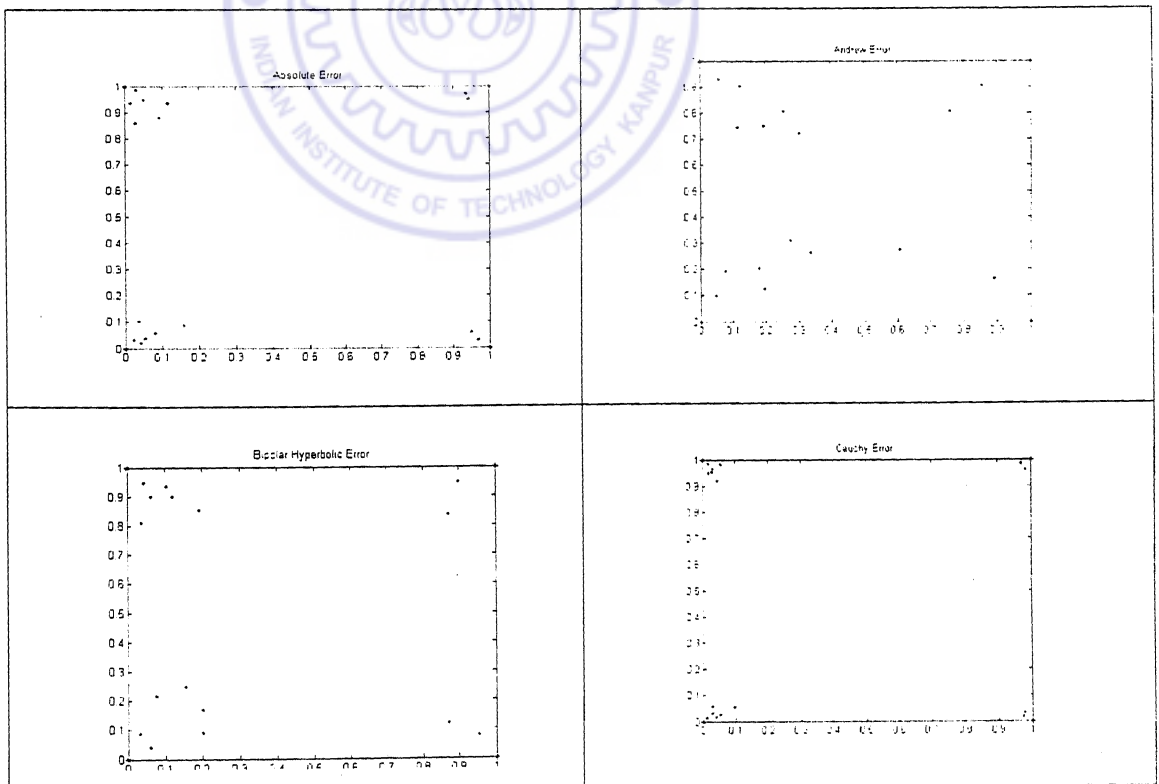
Table 3.4 The result with different CNNs tabulated for the CXOR problem. The simulated points shown in dots are close to the target points shown in asterisks. In all frames, the asterisks are positioned on the vertices of the rectangles.

Table 3.4 Continued

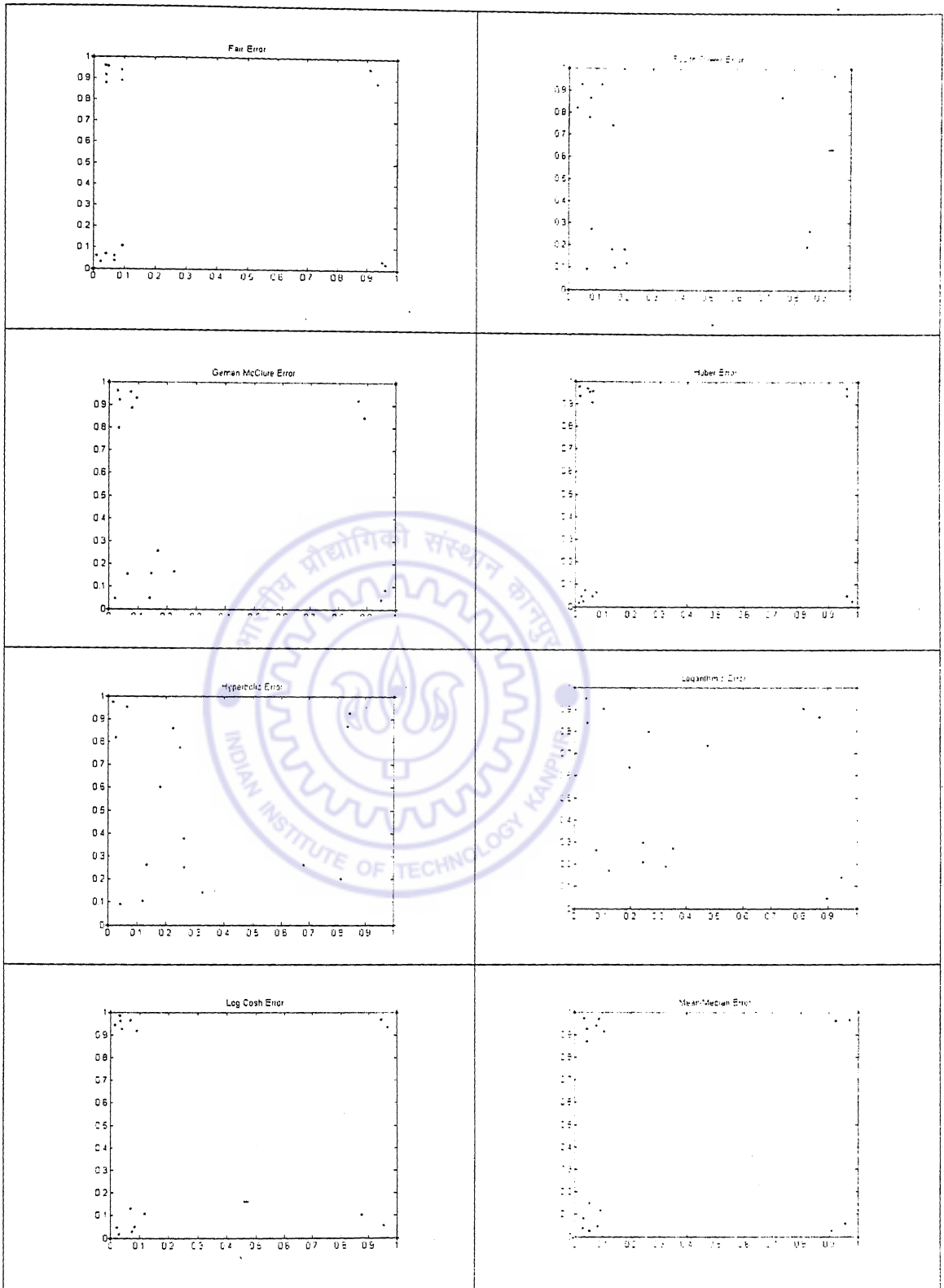
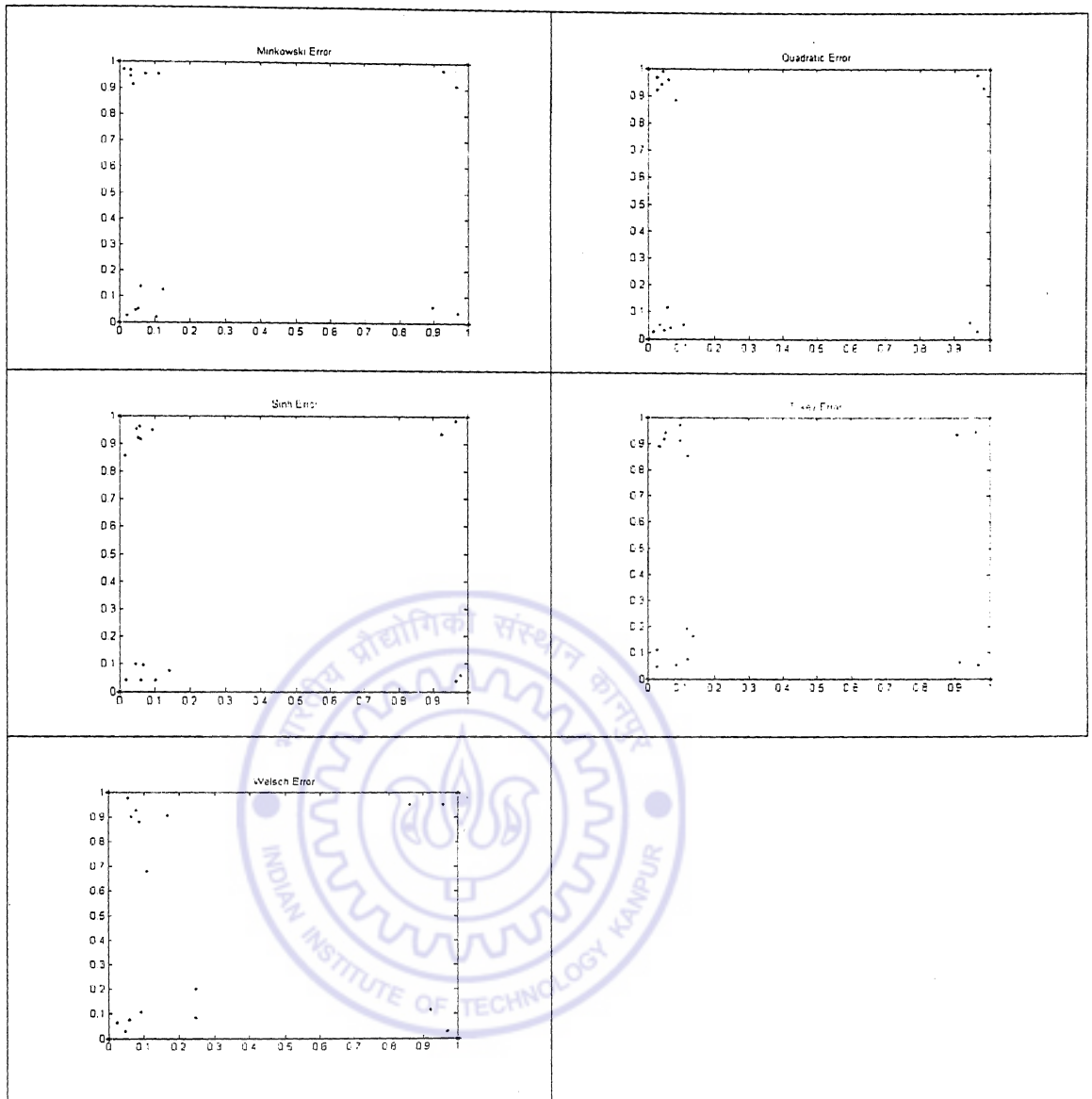


Table 3.4 Continued



The n -Parity was worked out for n equaling four, eight and sixteen all of which produced good result with both ANN and CNN. The n - parity problem was used to compare different training algorithms' by comparing the epochs needed to produce a perfect result. A further extension of the n -Parity problem involves developing the map in a generalized form by incorporating the definition of the Complex XOR to compose the arguments instead of the traditional real variable based XOR. The CXOR map as displayed in Table 2 can be seen to be a non-commutative map. Owing to this property, the 3-Parity complex map turns out non-associative. The two orders of evaluation of the inputs are

$$(c_1 \circ c_2) \circ c_3 \quad (3.36)$$

$$c_1 \circ (c_2 \circ c_3) \quad (3.37)$$

where, c_i are the three inputs. In the first combination, the first and the second are composed and the result later composed with the third. In the second combination, the first is composed with the result of the second and third. The data matrix for this problem is displayed in Table 3. In each row, the two ways of composing the arguments are shown for the same input arguments. It is by no means clear whether the CNN learns both maps defined by the compositions or what the network's behavior would be for non-associative transformations. It was observed during the runs that the first way of composing (eqn. (3.36)) defines a map that the CNN captures while the second sequence of compositions (eqn. (3.37)) didn't get captured even after running the algorithms for 15000 epochs. The results are displayed in Tables 3.6, 3.7. The reason for the same is attributed to the fact that the distance between the intermediate complex number as they are composed associatively changes in accordance with the sequence of operations for computing the maps. As explained for the n-Parity problem (Dagli, 1994), the problem was pronounced Benchmark because the input patterns that are relatively close are required to be mapped into different outputs which is what makes the problem a difficult one for the algorithms of the Back-Propagation family. A similar phenomenon is observed in the complex setting extension as displayed in Table 3.8 where the distances are computed in the following sequence. Referring to eqn. (3.36), the first computation was distance between the vectors $(c_1 \circ c_2)$ and c_3 and a second distance between the vectors $(c_1 \circ c_2)$ and $(c_1 \circ c_2) \circ c_3$. As shown labelled in Table 3.8, the vectors of distances have been plotted. these reveal how the intermediate complex numbers are hidden in the final mapping of the two compositions shown in Table 3.5. As a net effect of the interplay between the peaks of cells (c) and (d) of the Table 3.8 the mapping was captured. There are effectively more peaks in cells (a) and (b) of Table 3.8, which disrupt the convergence process, contrary to the peaks of cell (d) which are more flat.

Table 3.5 Complex 3-Parity Problem is Non-Associative

| c_1 | c_2 | c_3 | $c_1 \circ (c_2 \circ c_3)$ | c_1 | c_2 | c_3 | $(c_1 \circ c_2) \circ c_3$ |
|-------|-------|-------|-----------------------------|-------|-------|-------|-----------------------------|
| 0 | 0 | 0 | 0+i | 0 | 0 | 0 | 0 |
| 0 | 0 | 0+i | 0+i | 0 | 0 | 0-i | 0-i |
| 0 | 0 | 1 | 0+i | 0 | 0 | 1 | 1-i |
| 0 | 0 | 1+i | 1 | 0 | 0 | 1-i | 0 |
| 0 | 0+i | 0 | 1 | 0 | 0-i | 0 | 0 |
| 0 | 0+i | 0+i | 0 | 0 | 0+i | 0+i | 1-i |
| 0 | 0+i | 1 | 0+i | 0 | 0+i | 1 | 0-i |
| 0 | 0+i | 1+i | 1 | 0 | 0+i | 1-i | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0+i | 0+i | 0 | 1 | 0+i | 1-i |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0-i |
| 0 | 1 | 1+i | 1 | 0 | 1 | 1-i | 0 |
| 0 | 1+i | 0 | 1 | 0 | 1+i | 0 | 1 |
| 0 | 1+i | 0+i | 0+i | 0 | 1+i | 0-i | 0-i |
| 0 | 1+i | 1 | 0+i | 0 | 1+i | 1 | 0-i |
| 0 | 1+i | 1+i | 0+i | 0 | 1+i | 1-i | 0 |
| 0+i | 0 | 0 | 0+i | 0+i | 0 | 0 | 1 |
| 0+i | 0 | 0+i | 1+i | 0+i | 0 | 0+i | 0-i |
| 0+i | 0 | 1 | 1+i | 0+i | 0 | 1 | 0-i |
| 0+i | 0 | 1+i | 0 | 0+i | 0 | 1+i | 0 |
| 0+i | 0+i | 0 | 0 | 0+i | 0+i | 0 | 0 |
| 0+i | 0+i | 0+i | 0 | 0+i | 0+i | 0+i | 0-i |
| 0+i | 0+i | 1 | 1+i | 0+i | 0+i | 1 | 0-i |
| 0+i | 0+i | 1+i | 0 | 0+i | 0+i | 1-i | 1 |
| 0+i | 1 | 0 | 0 | 0+i | 1 | 0 | 0 |
| 0+i | 1 | 0+i | 1+i | 0+i | 1 | 0-i | 1-i |
| 0+i | 1 | 1 | 0 | 0+i | 1 | 1 | 0-i |
| 0+i | 1 | 1+i | 0 | 0+i | 1 | 1-i | 0 |
| 0+i | 1+i | 0 | 0 | 0+i | 1+i | 0 | 1 |
| 0-i | 1+i | 0+i | 1+i | 0+i | 1+i | 0-i | 0-i |
| 0-i | 1+i | 1 | 1+i | 0+i | 1+i | 1 | 0-i |
| 0-i | 1+i | 1+i | 0+i | 0-i | 1+i | 1-i | 0 |
| 1 | 0 | 0 | 1+i | 1 | 0 | 0 | 1 |
| 1 | 0 | 0+i | 0+i | 1 | 0 | 0-i | 0-i |
| 1 | 0 | 1 | 0+i | 1 | 0 | 1 | 0-i |
| 1 | 0 | 1+i | 0 | 1 | 0 | 1-i | 0 |
| 1 | 0+i | 0 | 0 | 1 | 0+i | 0 | 0 |
| 1 | 0+i | 0+i | 0 | 1 | 0+i | 0+i | 1-i |
| 1 | 0+i | 1 | 0+i | 1 | 0+i | 1 | 0-i |
| 1 | 0+i | 1+i | 0 | 1 | 0+i | 1+i | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0+i | 0+i | 1 | 1 | 0+i | 0-i |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0-i |
| 1 | 1 | 1+i | 0 | 1 | 1 | 1+i | 1 |
| 1 | 1+i | 0 | 0 | 1 | 1+i | 0 | 1 |
| 1 | 1+i | 0+i | 0+i | 1 | 1+i | 0-i | 0-i |
| 1 | 1+i | 1 | 0+i | 1 | 1+i | 1 | 0-i |
| 1 | 1+i | 1+i | 1-i | 1 | 1+i | 1-i | 0 |
| 1+i | 0 | 0 | 0+i | 1+i | 0 | 0 | 1 |
| 1+i | 0 | 0+i | 0+i | 1+i | 0 | 0+i | 0-i |
| 1+i | 0 | 1 | 0+i | 1+i | 0 | 1 | 0-i |
| 1+i | 0 | 1+i | 0 | 1+i | 0 | 1+i | 0 |
| 1+i | 0+i | 0 | 0 | 1+i | 0+i | 0 | 0 |
| 1+i | 0+i | 0+i | 1 | 1+i | 0+i | 0-i | 1-i |
| 1+i | 0+i | 1 | 0+i | 1+i | 0+i | 1 | 0-i |
| 1+i | 0+i | 1+i | 0 | 1+i | 0+i | 1-i | 0 |
| 1-i | 1 | 0 | 0 | 1+i | 1 | 0 | 0 |
| 1-i | 1 | 0+i | 0-i | 1+i | 1 | 0-i | 1-i |
| 1-i | 1 | 1 | 1 | 1+i | 1 | 1 | 0-i |
| 1+i | 1 | 1+i | 0 | 1+i | 1 | 1-i | 0 |
| 1+i | 1+i | 0 | 0 | 1+i | 1+i | 0 | 0 |
| 1+i | 1+i | 0+i | 0+i | 1+i | 1+i | 0-i | 0-i |
| 1-i | 1+i | 1 | 0+i | 1+i | 1+i | 1 | 1-i |
| 1+i | 1+i | 1+i | 0-i | 1+i | 1-i | 1+i | 0 |

... eqn. (3.50). The table displays non-associative mapping captured by EF based CNN. The data for the map is as in Table 3.5.

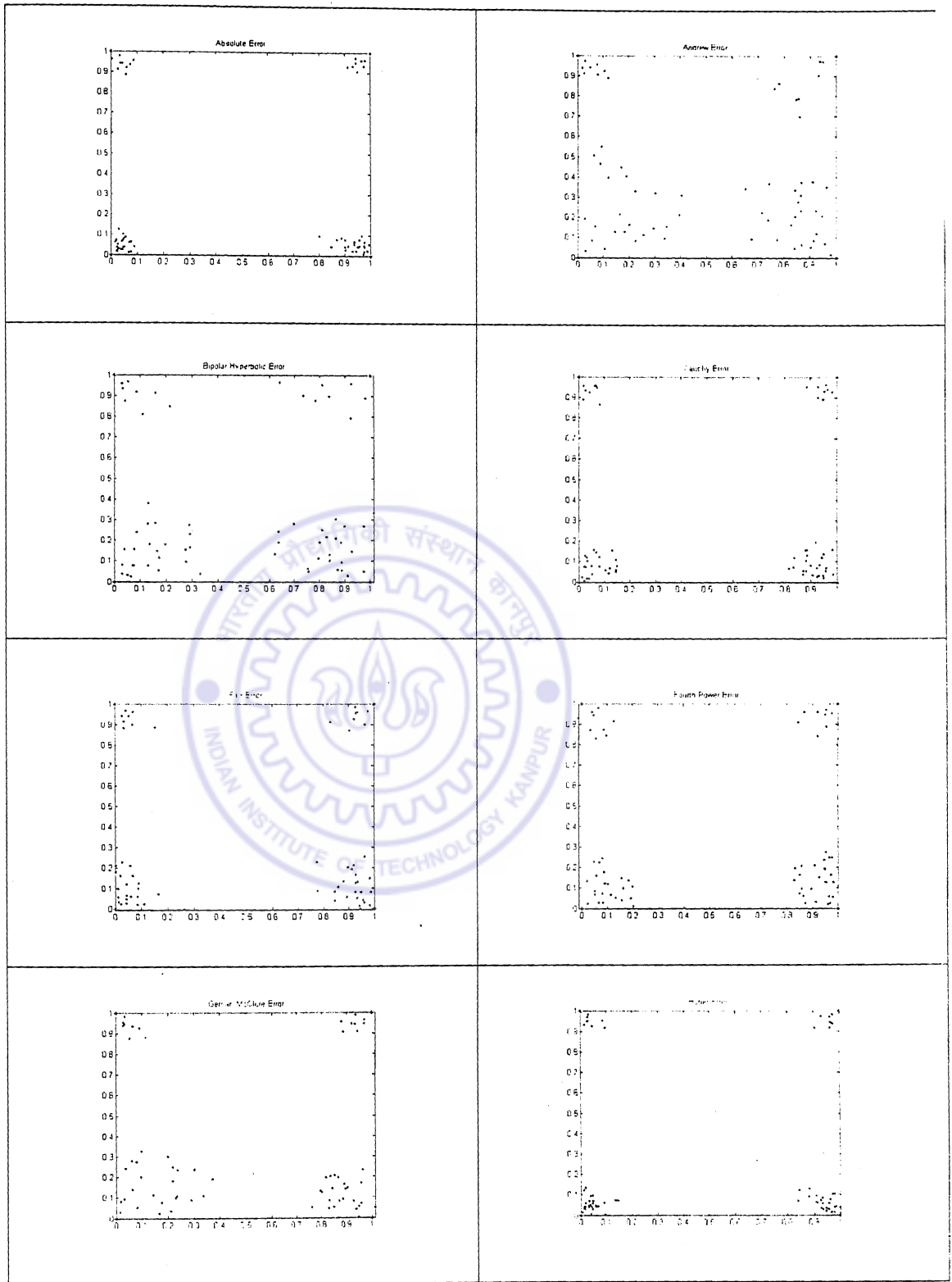


Table 3.6 Continued

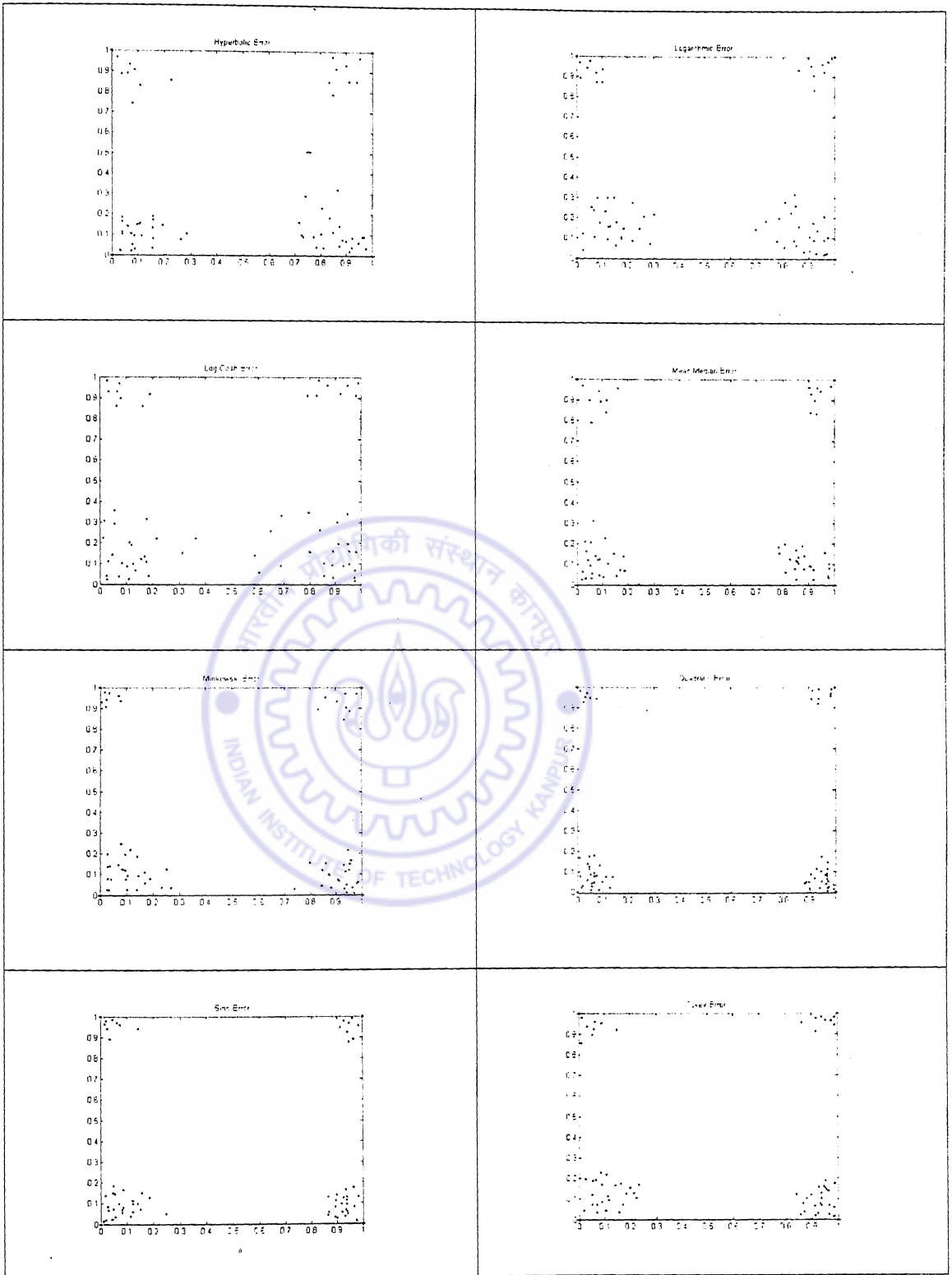


Table 3.6 Continued

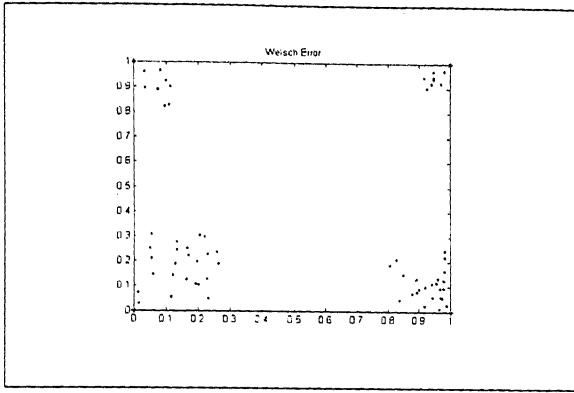


Table 3.7 Complex 3-Parity problem, composition as in eqn. (3.37). The table displays non-associative mapping. The data for the map is as in Table 3.5. This map didn't produce satisfactory convergence even after 15000 epochs

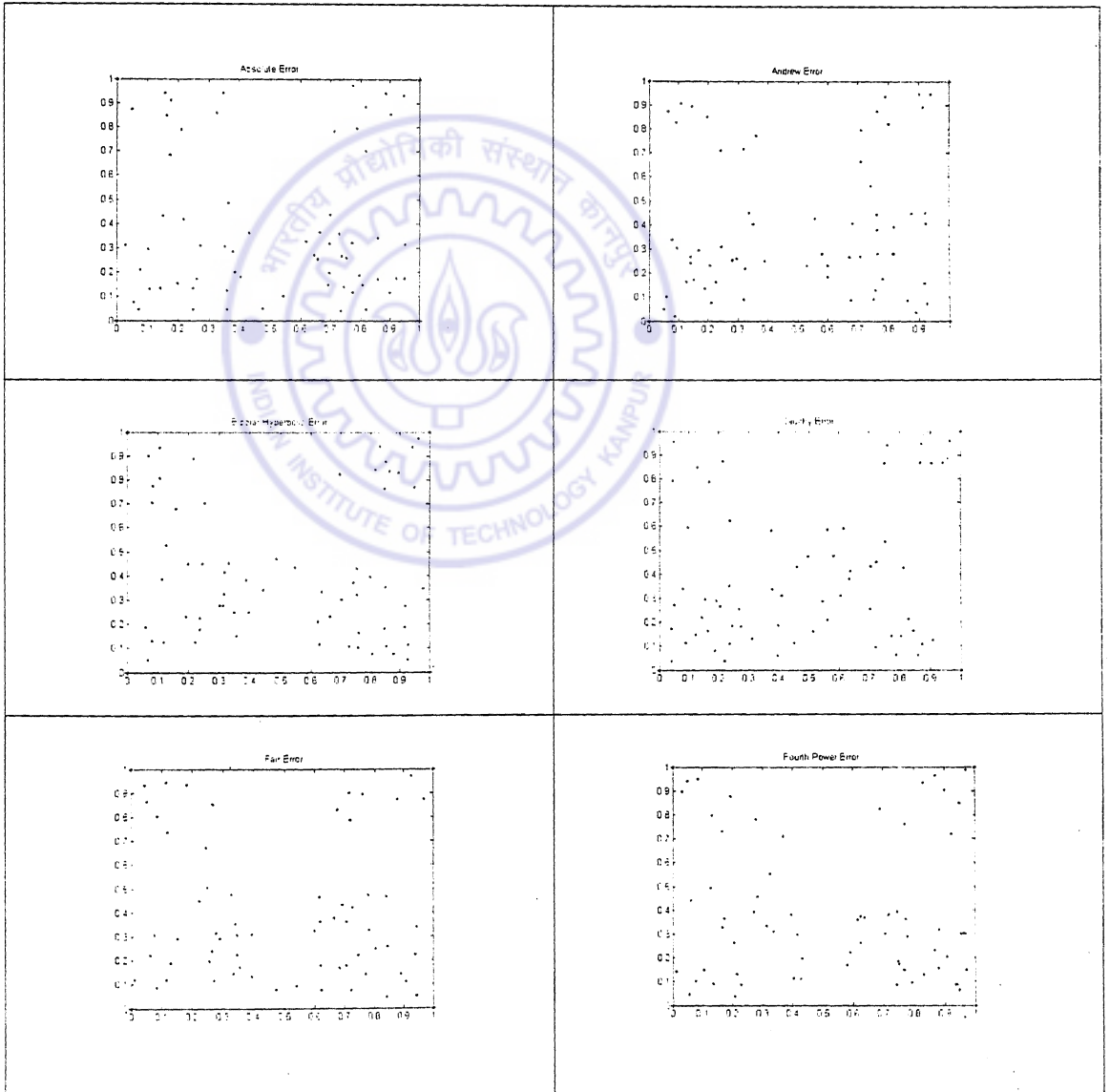


Table 3.7 Continued

मुद्रास्तम्भ प्रौद्योगिकी संस्थान कानपुर पुस्तकालय
 भारतीय प्रौद्योगिकी संस्थान कानपुर
 अवाप्ति क्र० A. 149338.....

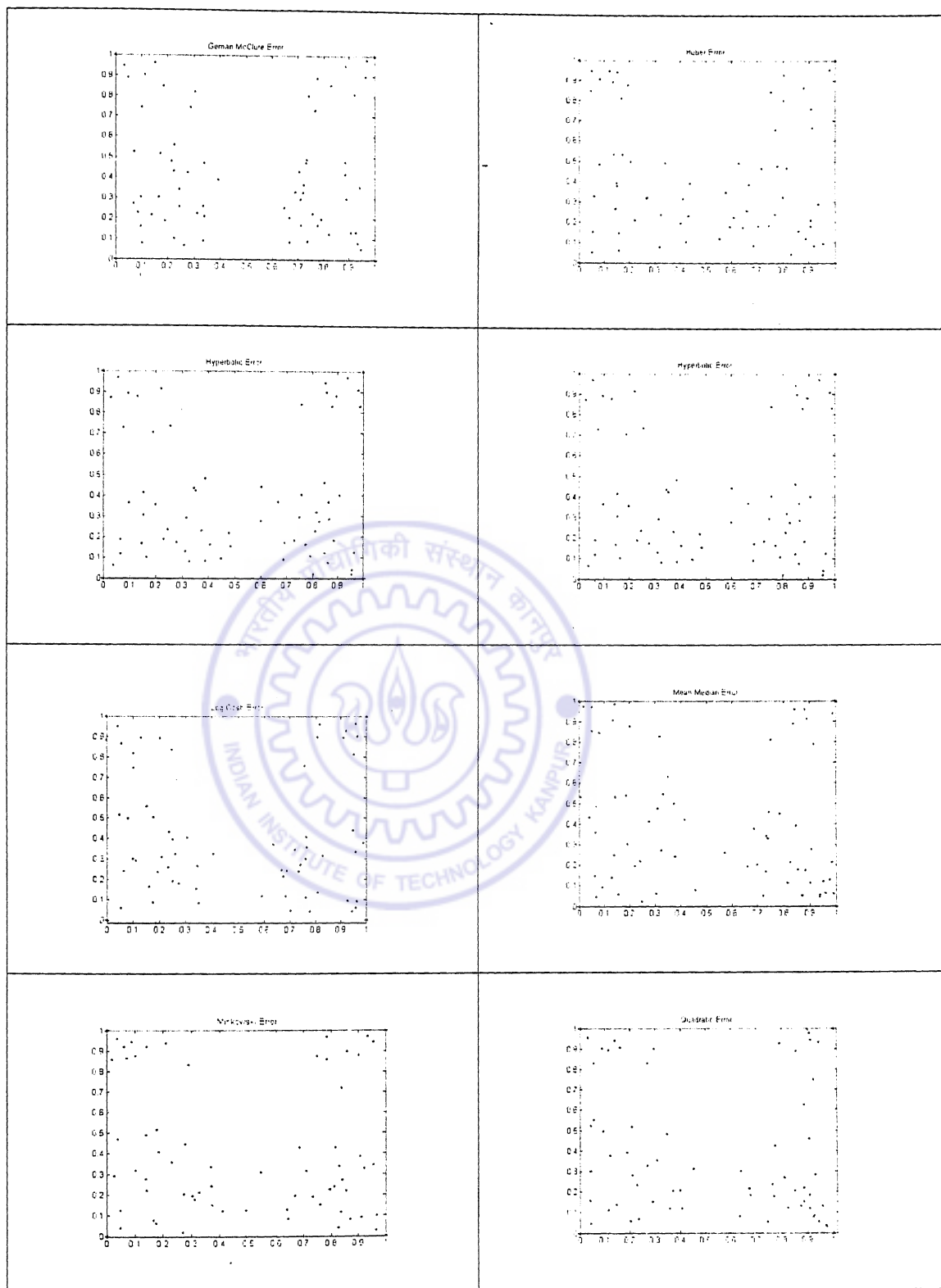
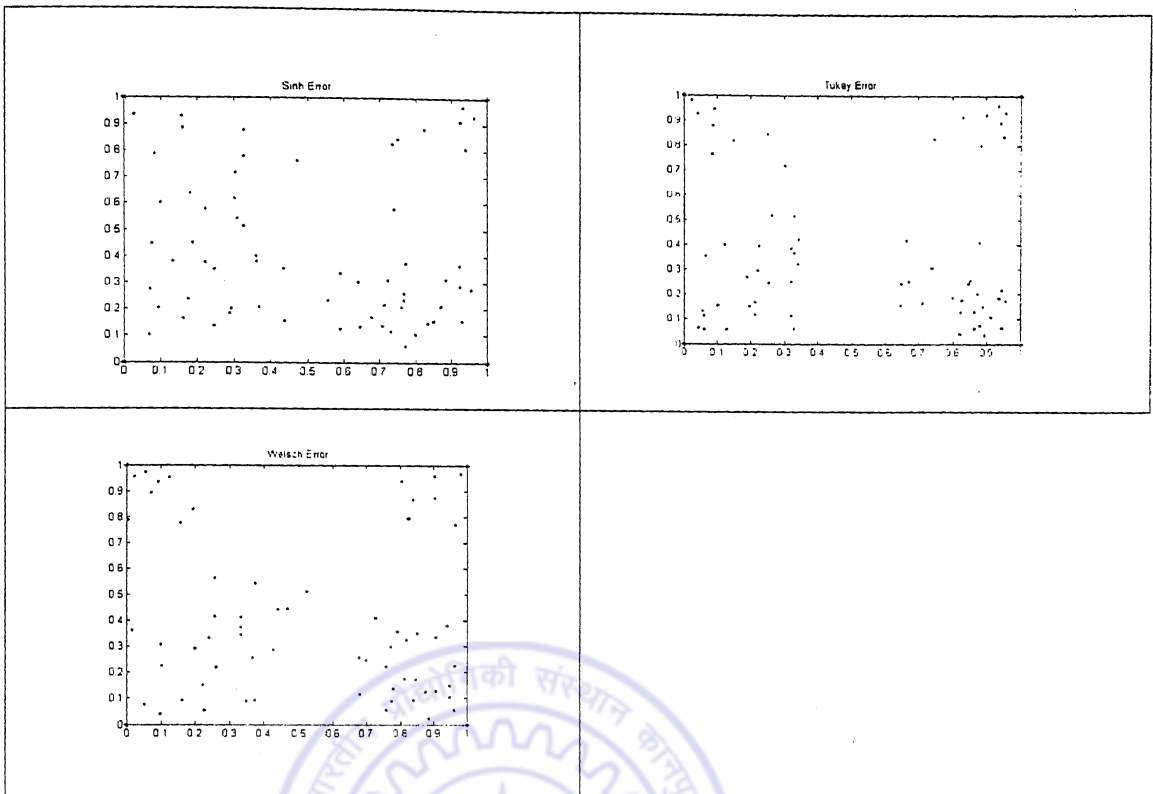


Table 3.7 Continued



3.5.3 The Coding-Decoding Problem

Ackley, Hinton, Sejnowski (1986) posed a problem for internal representation testing in which a set of N_I orthogonal input patterns get mapped to a set of N_O orthogonal output patterns. Internal representation is a study that involves deciphering how the input patterns get transformed when input to a Neural Network. A suitable number of hidden layer neurons N_h need to be selected to capture the input-output map. The mapping obtained performs a coding of the N_I bit input pattern onto an N_h bit pattern and an N_h bit pattern onto the N_O bit output pattern. The first half of the mapping from input to the hidden layer is the coding part while the rest of the map from hidden layer to the output layer is the decoding part of the problem. The Benchmark problem is important to evaluate the network storage capacity and has application to data compression and transmission. For the particular case of $N_O = N_h = N$, N_h was shown to be $\log_2(N)$ (Zurada, 1997). In the extended version to the complex domain, the problem should be raised to serve as a coding-decoding problem for the CNN's. The problem in the generalized form reads mapping the matrix with diagonal entries $(1+i)$ to itself and CNN trained to capture this. The number of hidden layer neurons was chosen to be two as the input-output matrix was of order 16×16 .

Table 3.8 Norm computation for the Complex 3-Parity Problem. (a) and (b) depict respectively how the norms between (c2oc3) and c1 and (c2oc3) and c1o(c2oc3) vary with vector index. (c) and (d) are the corresponding plots for the second composition sequence. The titles of each plot describe how they are computed.

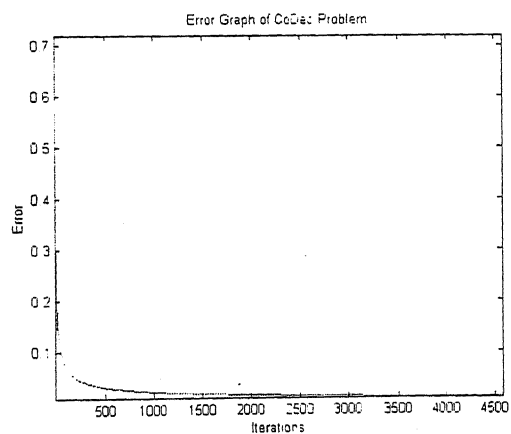
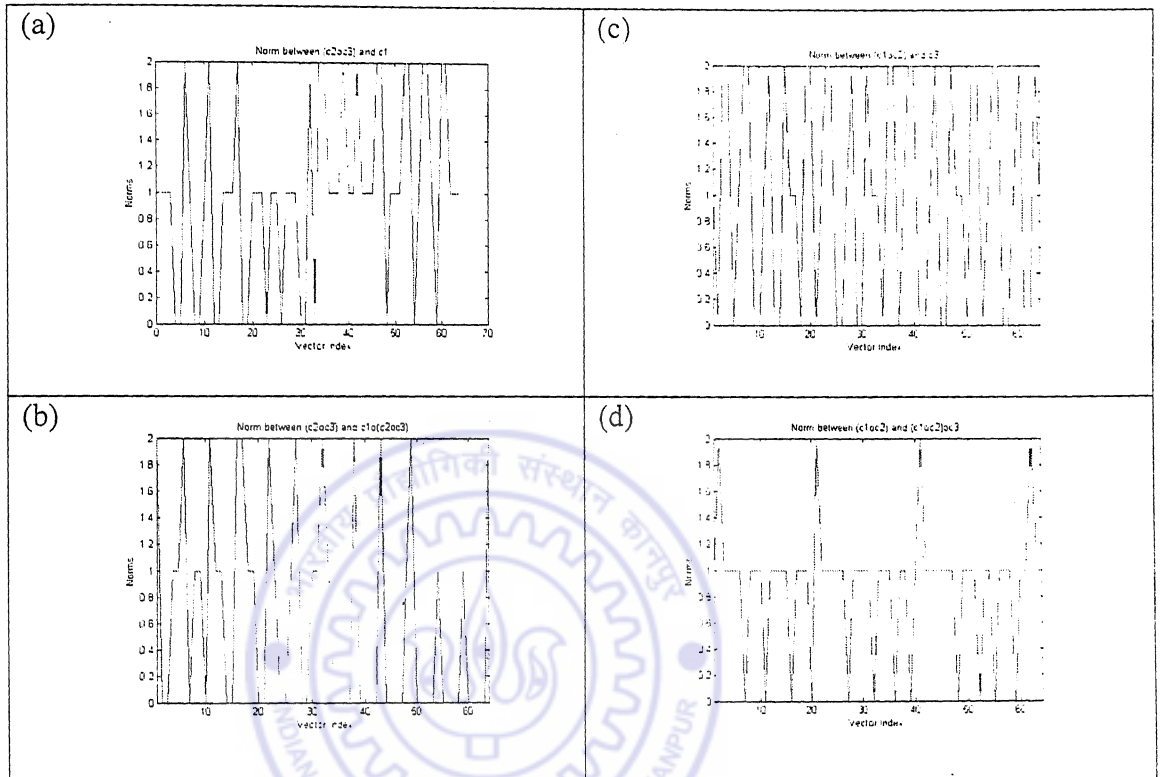


Fig. 3.2 Error plotted against number of iterations while training a CNN for CoDec Problem

Table 3.9 CoDec Problem using EF based ANN and CNN

| Average Result with Real and Complex Error Functions (16-4-16 Architecture) Learning Rate was 0.1 Target Error was 0.0001 | | |
|---|-----------------|-------------------|
| Error Function | Real EF, Epochs | Complex EF Epochs |
| Absolute Error | 2500 | 1600 |
| Andrew Error | 17000 | 22000 |
| Cauchy Error | 3500 | 2000 |
| Error Fourth Order | 12200 | 8600 |
| Fair Error | 5000 | 10000 |
| Geman-McClure Error | 2500 | 4000 |
| Huber Error | 1700 | 1700 |
| Hyperbolic Squared | 21000 | 17000 |
| Bipolar Hyperbolic | 20000 | 19000 |
| LogCosh Error | 1900 | 7700 |
| Logarithmic Error | 12500 | 19000 |
| Mean Median Error | 2000 | 7000 |
| Minkowski Error | 10000 | 5000 |
| Quadratic Error | 1700 | 3500 |
| Sinh Error | 1600 | 2000 |
| Tukey Error | 2100 | 2500 |
| Welsch Error | 20000 | 13000 |

3.5.4 The Sin(x)Sin(y) Problem

In general researchers have worked with several function-mapping and approximation problems to test the Neural Networks' learning capabilities and the efficiency of their training algorithms. A most frequently used problem is capturing the surface governed by the equation

$$z = \sin(x) \sin(y) \quad (3.38)$$

using Neural Networks. The surface is periodic as it is made up of functions that repeat at regular intervals. The mapping becomes more complex as the norm of the input vector grows (Dagli, 1994). The following explanation shows why approximating the surface for large values of the input vector makes the training process more complicated.

To approximate the sine curve, in the vicinity of the origin a single term in the Taylor series expansion suffices. The following sandwich inequality that connects the sine, the linear and the tangent functions in an inequality is an extension of the fact just stated.

$$\sin(x) < x < \tan(x) \quad (3.39)$$

As the argument to the sine function increases, more terms in the Taylor expansion need be considered. The Taylor expansion approximates the sine function in the manner shown in the diagram as the number of terms considered in the expansion are increased.

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \dots \quad (3.40)$$

The sine function is bounded and the approximation given by Taylor series is valid over the whole real line. It must be noted that the Taylor expansion is an infinite polynomial that approximates the function over the whole line (a finite polynomial would go unbounded over the real line but the sine function is bounded over the line). A sufficient number of terms of the expansion must be considered for approximating the function for large input arguments. The sine function is bounded over the whole real line and hence

Table 3.10 Result showing the number of epochs required for capturing $z=\sin(x)\sin(y)$. A target error of 0.0001 was set with a 1-5-1 architecture. Result was averaged across three runs.

| Average Result with Real and Complex Error Functions (1-5-1 Architecture) | | | | |
|---|-------------------------|---------|------------------------|---------|
| Error Function | Surface in $[0, \pi/2]$ | | Surface in $[0, 2\pi]$ | |
| | Real | Complex | Real | Complex |
| Absolute Error | 1500 | 1000 | 2500 | 6200 |
| Andrew Error | 6500 | 8000 | 9000 | 11000 |
| Cauchy Error | 1100 | 900 | 2000 | 4000 |
| Error Fourth Order | 1200 | 1500 | 3000 | 5500 |
| Fair Error | 1400 | 2000 | 2500 | 5000 |
| Geman-McClure Error | 1550 | 2500 | 4000 | 3500 |
| Huber Error | 1400 | 1100 | 2500 | 3000 |
| Hyperbolic Squared | 1600 | 1900 | 6000 | 13000 |
| Bipolar Hyperbolic | 1550 | 2200 | 5500 | 15000 |
| LogCosh Error | 1100 | 1600 | 3000 | 3000 |
| Logarithmic Error | 3550 | 5200 | 8000 | 11000 |
| Mean Median Error | 1450 | 1100 | 2500 | 4500 |
| Minkowski Error | 1300 | 900 | 4500 | 6000 |
| Quadratic Error | 1200 | 900 | 2500 | 3500 |
| Sinh Error | 1550 | 1000 | 3500 | 5000 |
| Tukey Error | 1450 | 2200 | 4000 | 7000 |
| Welsch Error | 1500 | 2000 | 4000 | 6000 |

truncating the Taylor Series to finitely many terms would imply that the approximation is valid over a finite length on either sides of the axis of abscissa and invalid outside the range for the expression starts to rise and go unbounded. The approximation by the Neural Network is restricted to the region in which the problem is considered, as beyond

this the polynomial is invalid because it doesn't approximate the sine function anymore (and goes unbounded).

As the surface in question is made up of two sine functions, two such approximations of the polynomial must be considered for each of the sine functions that makes up the equation. For approximating the surface in the interval $[0, 2\pi]$, the series should be expanded up to the ninth term (by actually expanding and comparing terms). The surface has an x -series and a y -series each considered up to the ninth term (assuming that the region of interest on the xy -plane is $[0, 2\pi] \times [0, 2\pi]$) thus making a polynomial surface of order eighteen, rendering the training process complicated. If the region of interest extends beyond the interval, more terms of the Taylor expansion need be considered to approximate the sine function and hence the problem's complexity increases. It follows from the explanation that as the norm of the input vector grows, the problem becomes more complex. The surface to be trained as the input vector grows is shown in the following diagrams for different input norms. The above reason extends to the complex variable based networks as well. The series expansion of the sine function for complex inputs is given by the Taylor expansion of sine for complex arguments

$$\sin(z) = z - \frac{z^3}{3!} + \frac{z^5}{5!} - \frac{z^7}{7!} + \frac{z^9}{9!} - \dots \quad (3.41)$$

The argument for the real function extends to the complex sine function also. The number of terms required for approximating the function increases as the norm of the input vector grows (now in the complex domain) and hence for the complex function

$$w = \sin(z_1) \sin(z_2) \quad (3.42)$$

with two complex input arguments, the order of the approximating polynomial increases as the norm of the vector grows making the approximation scheme complicated for large norms of the argument. The analysis reveals that the constraint about the complexity of training that existed with real-variable based network for large norms of the input extends even to the complex domain.

Table 3.11 CNN Mapping the $\sin(x)\sin(y)$ surface in $[0, \pi/2]$. CNN outperformed ANN in this range of the map. Architecture was 1-5-1, Learning rate was 0.1. The dots are the network outputs, grid lines are the map of the surface from the function.

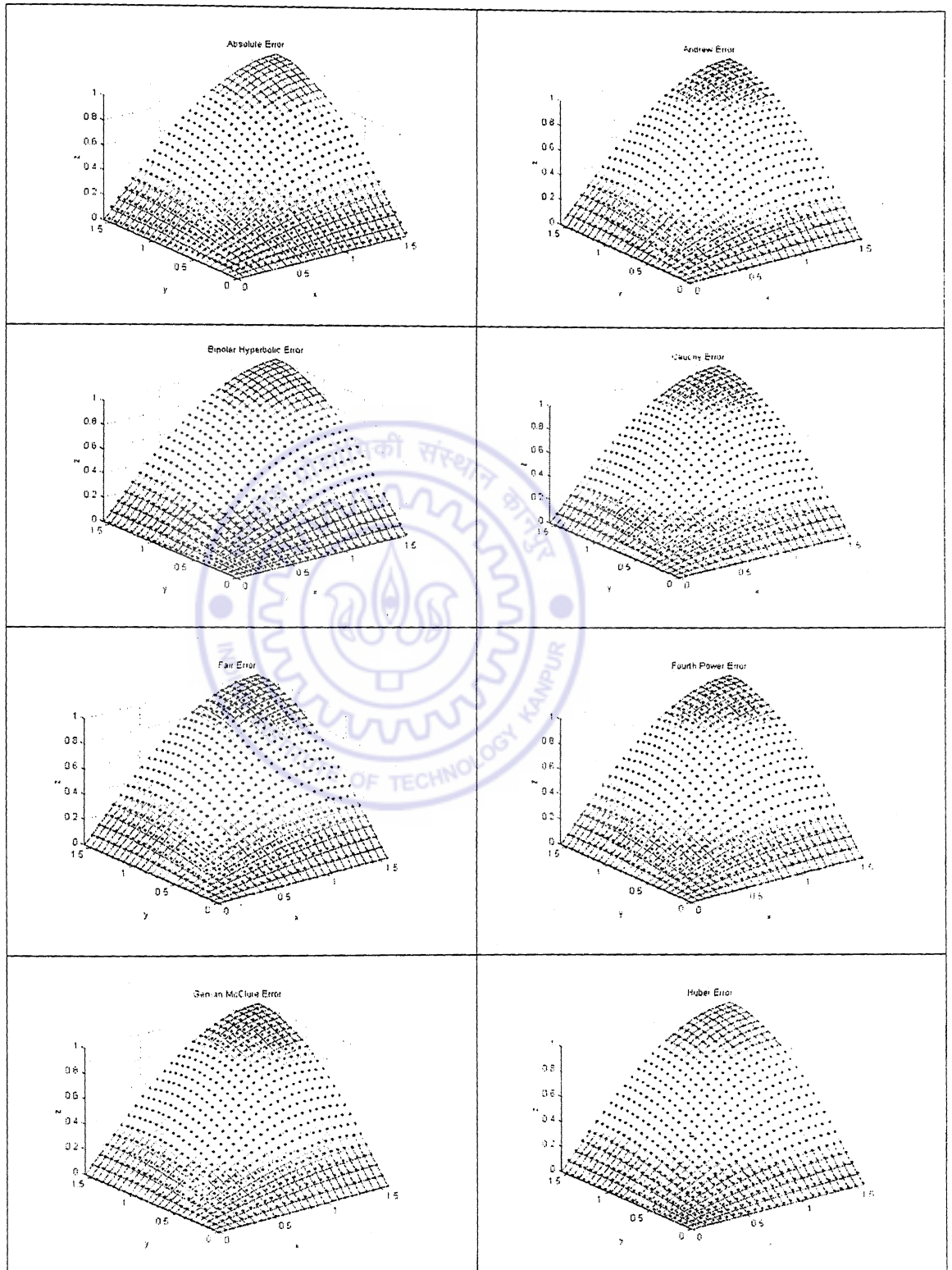


Table 3.11 Continued

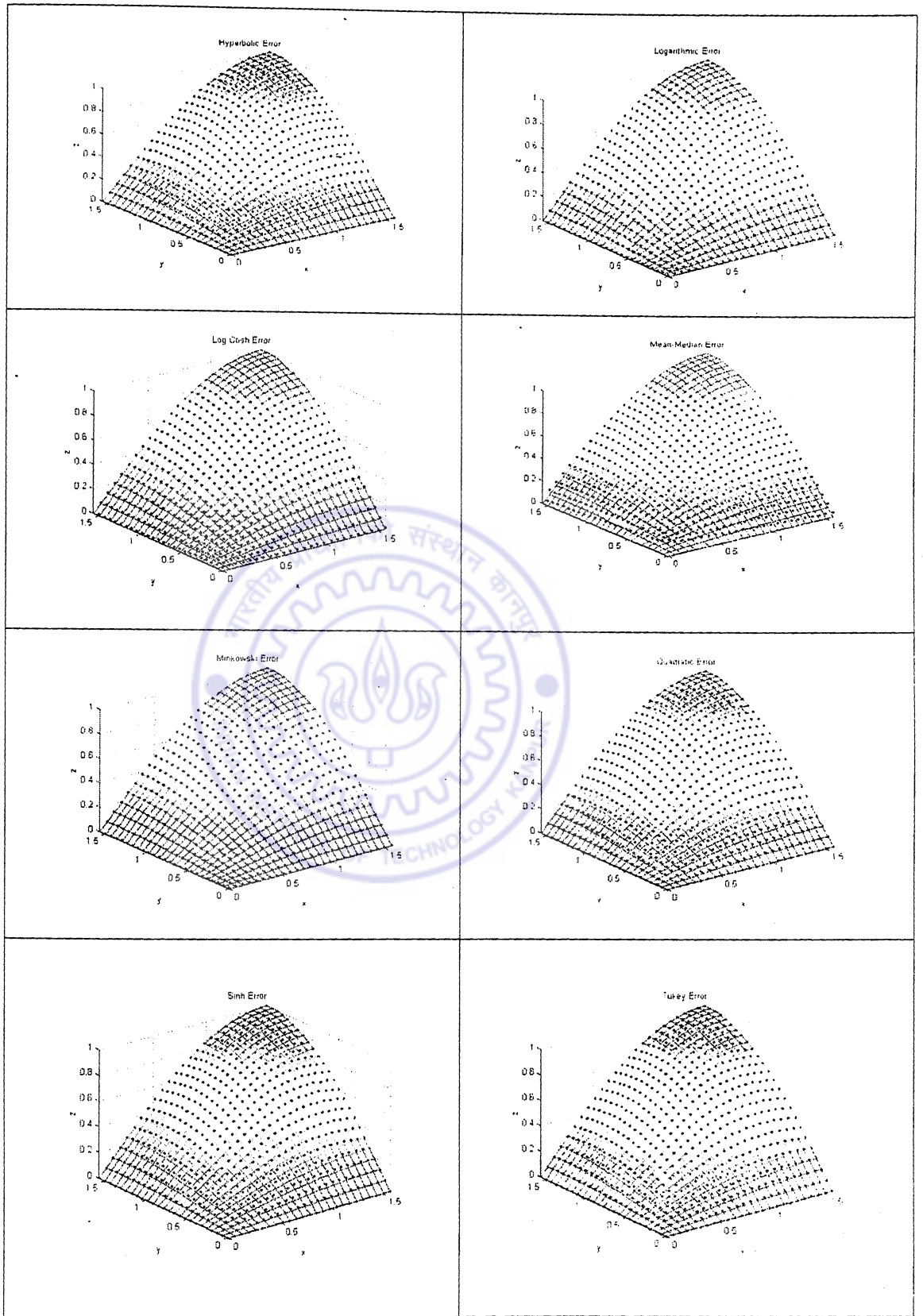


Table 3.11 Continued

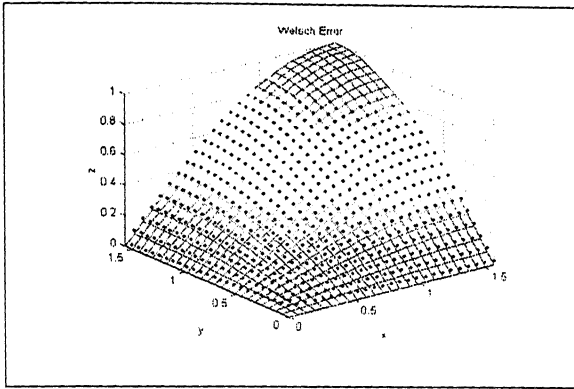


Table 3.12 Real EF based ANNs mapping the $\sin(x)\sin(y)$ surface using a 1-5-1 architecture. in the range $[0.2*\pi]$ where ANN performed better than CNN. Dots are the network outputs, grid lines show the surface map from the function.

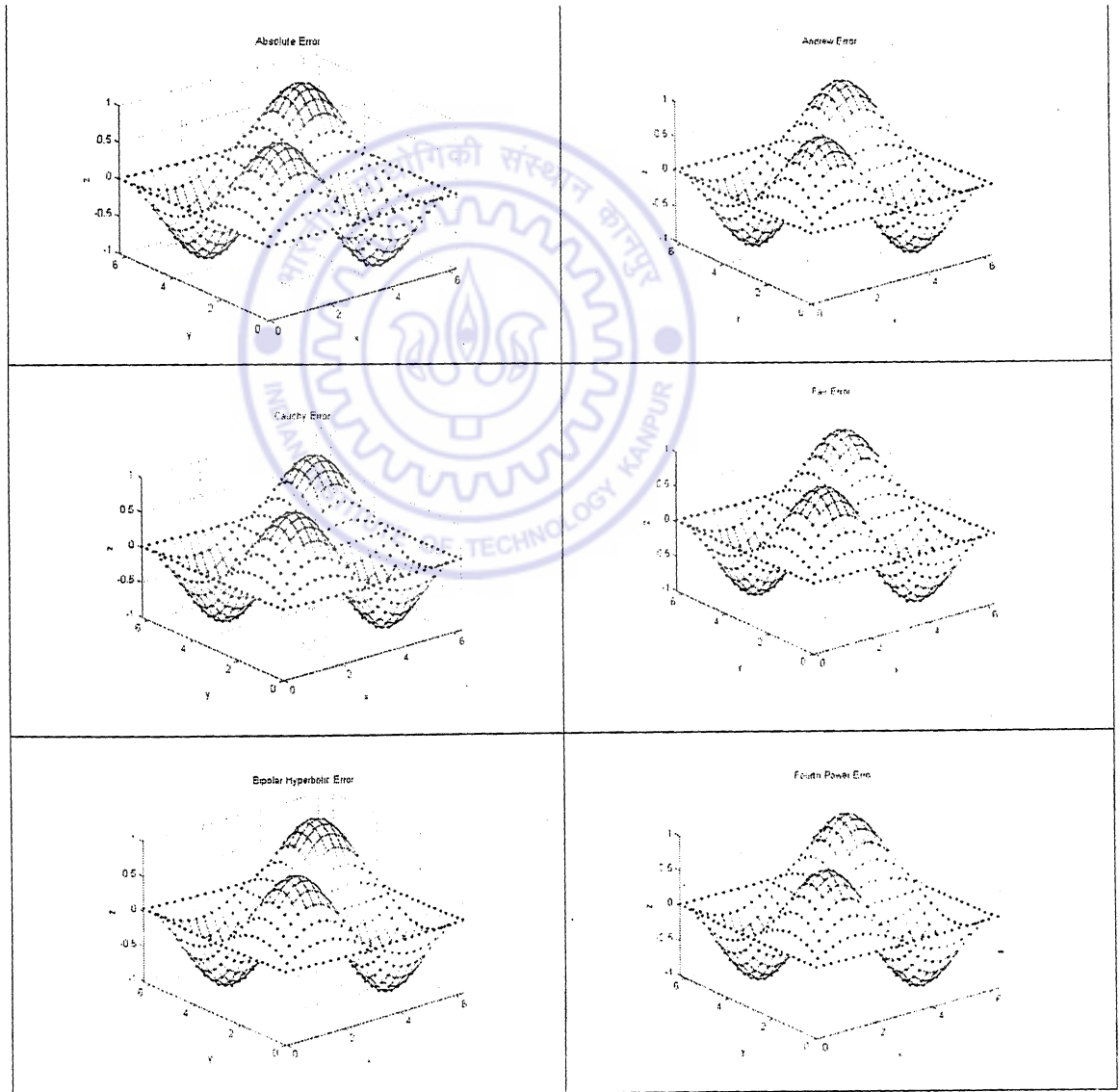


Table 3.12 Continued

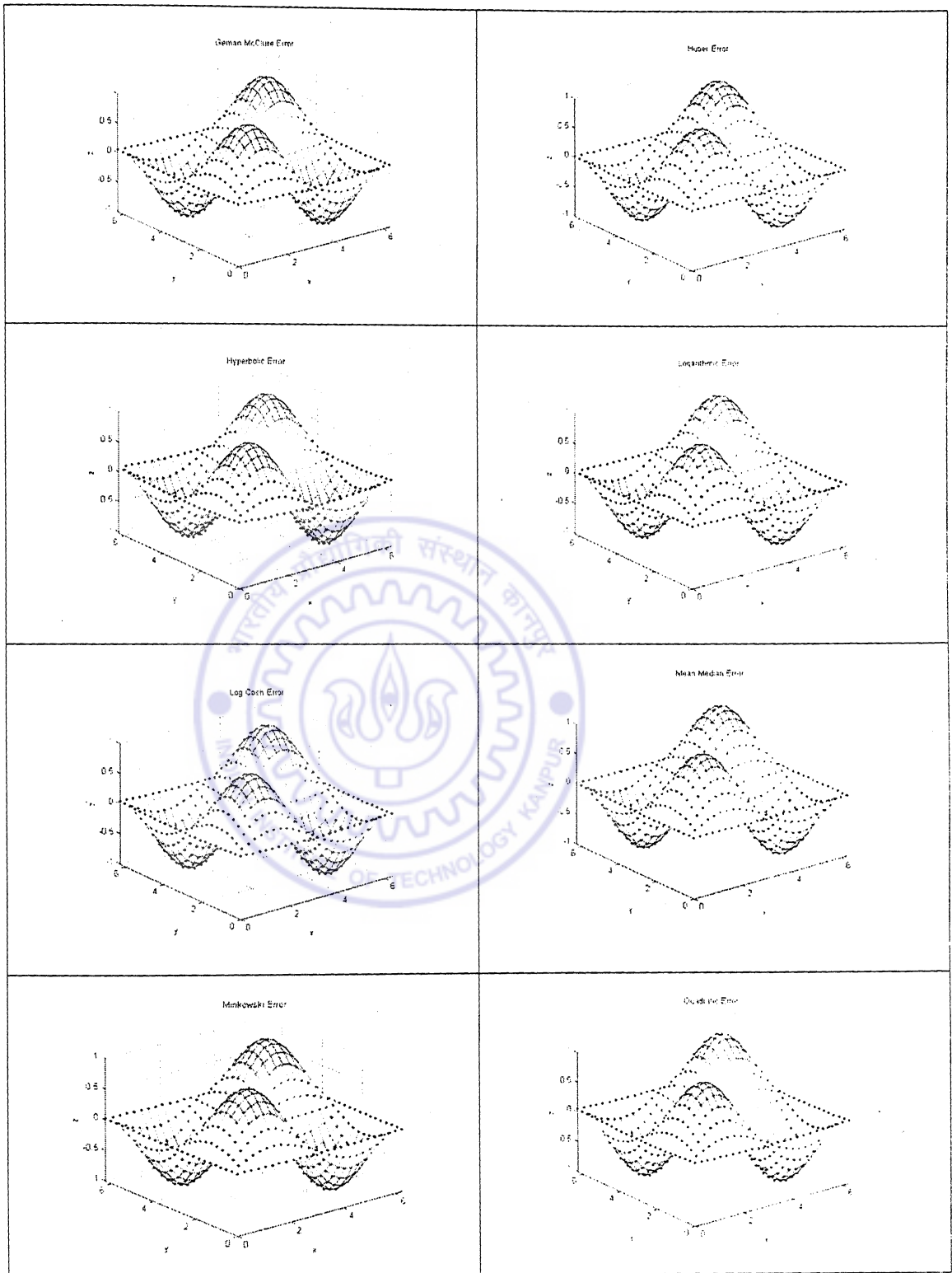
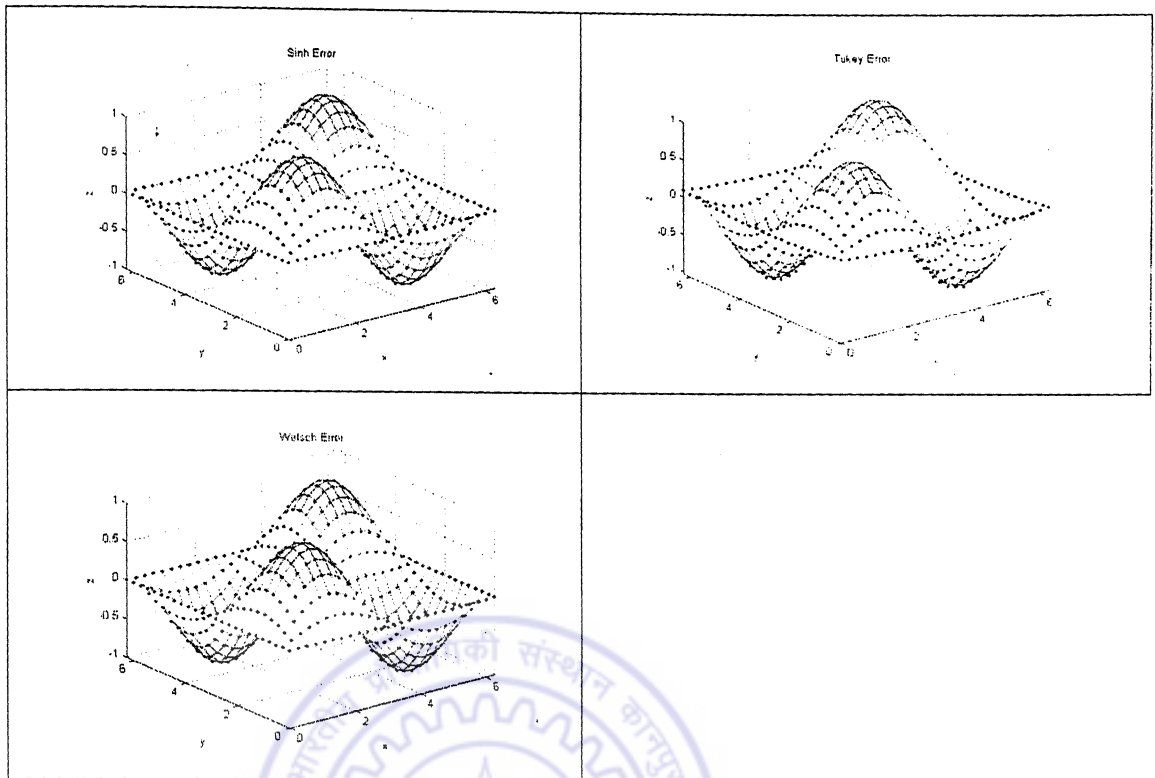


Table 3.12 Continued



To study the complex input-output map, the extended version should be modeled to include the above-mentioned constraint. Moreover, the problem of normalization is pronounced in the complex domain. Owing to these constraints, the map was normalized with absolute value lying in the interval $[0,1]$, taking care to see that the input always lay in the effective region of all activation functions. The constraint must be respected because the comparison scheme can be effective only when the constraints are efficiently met with for otherwise, a certain algorithm might produce inferior results owing to the improper development of the data set than the actual convergence characteristic of the algorithm. It must be observed that the arguments to the extended version of the surface map are both complex-valued. To interpret the surface, one requires two different complex planes, one for each argument. The first argument is interpreted to lie on the first complex plane and the second argument on the second plane. The image of the function as computed by the definition is interpreted to lie on a third complex-plane. No graphical description can exist for the present map for it's a function in four independent real variables and two dependent real variables. In the present problem, the first variable was chosen to lie on a circle in the first complex-plane (represented by the variable z_1 in the equation (3.43)) and the second variable also on a circle in the second complex-plane (represented by the variable z_2) each curve normalized to suit to the constraints imposed

by all the EF based algorithms. It must be observed here that unlike the real case where a whole region on the plane was mapped according to the function and shown plotted (in three-dimensional space), the complex-variable problem shows the mapping between curves chosen on the two planes and the output of the function, interpreted to lie on the third plane. The plots shown depict how the CNN captures the mapping on the output plane.

To see the complexity of the approximation as the number of terms in the series increase, the real and imaginary parts of the surface are displayed separately in a sequence. The real part of the following term is

$$\begin{aligned} & \left((x + iy) - \frac{(x + iy)^3}{3!} \right) \left((x1 + iy1) - \frac{(x1 + iy1)^3}{3!} \right) \\ & \left(x - \frac{1}{6}x^3 + \frac{1}{2}xy^2 \right) \left(x1 - \frac{1}{6}x1^3 + \frac{1}{2}x1y1^2 \right) \\ & - \left(y - \frac{1}{2}x^2y + \frac{1}{6}y^3 \right) \left(y1 - \frac{1}{2}x1^2y1 + \frac{1}{6}y1^3 \right) \end{aligned} \quad (3.43)$$

and the imaginary part is

$$\begin{aligned} & \left(y - \frac{1}{2}x^2y + \frac{1}{6}y^3 \right) \left(x1 - \frac{1}{6}x1^3 + \frac{1}{2}x1y1^2 \right) \\ & + \left(x - \frac{1}{6}x^3 + \frac{1}{2}xy^2 \right) \left(y1 - \frac{1}{2}x1^2y1 + \frac{1}{6}y1^3 \right) \end{aligned} \quad (3.44)$$

The real and imaginary parts of the expression when the number of the terms in the series is three is

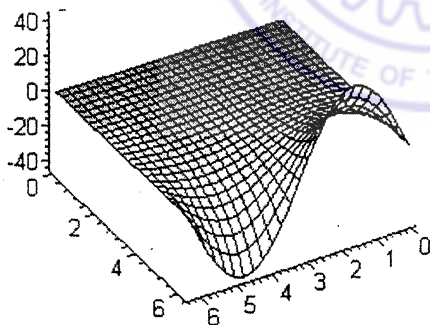
$$\begin{aligned} & \left(x - \frac{1}{6}x^3 + \frac{1}{2}xy^2 + \frac{1}{120}x^5 - \frac{1}{12}x^3y^2 + \frac{1}{24}xy^4 \right) \left(x1 - \frac{1}{6}x1^3 + \frac{1}{2}x1y1^2 \right) \\ & - \left(y - \frac{1}{2}x^2y + \frac{1}{6}y^3 + \frac{1}{24}x^4y - \frac{1}{12}x^2y^3 + \frac{1}{120}y^5 \right) \left(y1 - \frac{1}{2}x1^2y1 + \frac{1}{6}y1^3 \right) \\ & + \frac{1}{120}x1^5 - \frac{1}{12}x1^3y1^2 + \frac{1}{24}x1y1^4 \end{aligned} \quad (3.45)$$

The imaginary part of the expression is

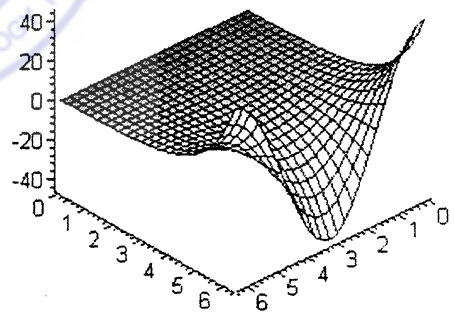
$$\begin{aligned}
& \left(y - \frac{1}{2} x^2 y + \frac{1}{6} y^3 + \frac{1}{24} x^4 y - \frac{1}{12} x^2 y^3 + \frac{1}{120} y^5 \right) \left(x l - \frac{1}{6} x l^3 + \frac{1}{2} x l y l^2 \right) \\
& + \left(x - \frac{1}{6} x^3 + \frac{1}{2} x y^2 + \frac{1}{120} x^5 - \frac{1}{12} x^3 y^2 + \frac{1}{24} x y^4 \right) \left(y l - \frac{1}{2} x l^2 y l + \frac{1}{6} y l^3 \right) \\
& + \frac{1}{24} x l^4 y l - \frac{1}{12} x l^2 y l^3 + \frac{1}{120} y l^5
\end{aligned} \tag{3.46}$$

To see how the series approximates progressively, the real and imaginary parts the Taylor polynomial are shown separately with plots as approximating the $\sin(x)\sin(y)$ surface.

The image of the map $w = \sin(z)$ as approximated by the Taylor series is displayed in the following graphs. The Taylor Series approximation to this function with increasing series terms are shown in the following sequence of figures (Table 3.13). The imaginary part of the Taylor series approximates the imaginary part of the function according to the sequence of diagrams shown there.



(a)



(b)

Fig.3.3 Plots (a) and (b) are respectively, the real and imaginary parts of the complex map $w = \sin(z)$. The plots show how Taylor series approximates $\sin(z_1)\sin(z_2)$. This is to show that as the argument increases, the order of polynomial causes the function to be more complex rendering the training process difficult.

Approximating $\sin(z)$ using Taylor Series. The figures (a)-(f) show the convergence of the Real Part of $\sin(z)$ as the number of terms of the series increase while figures (g)-(l) show how the imaginary part of the function approximates. These plots throw light on how the complexity of the $w=\sin(z_1)\sin(z_2)$ map varies as the number of terms of approximation vary.

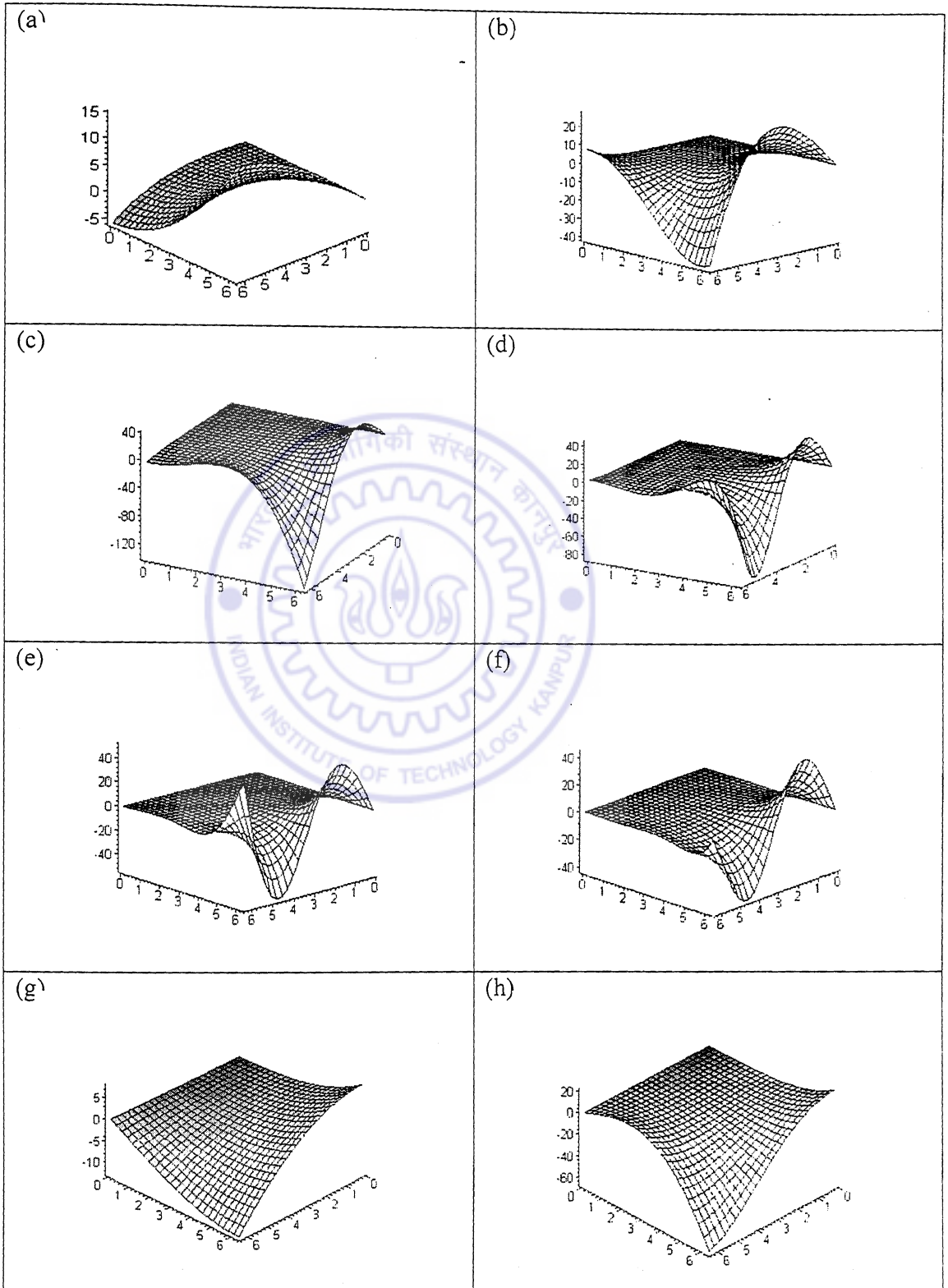


Table 3.13 Continued

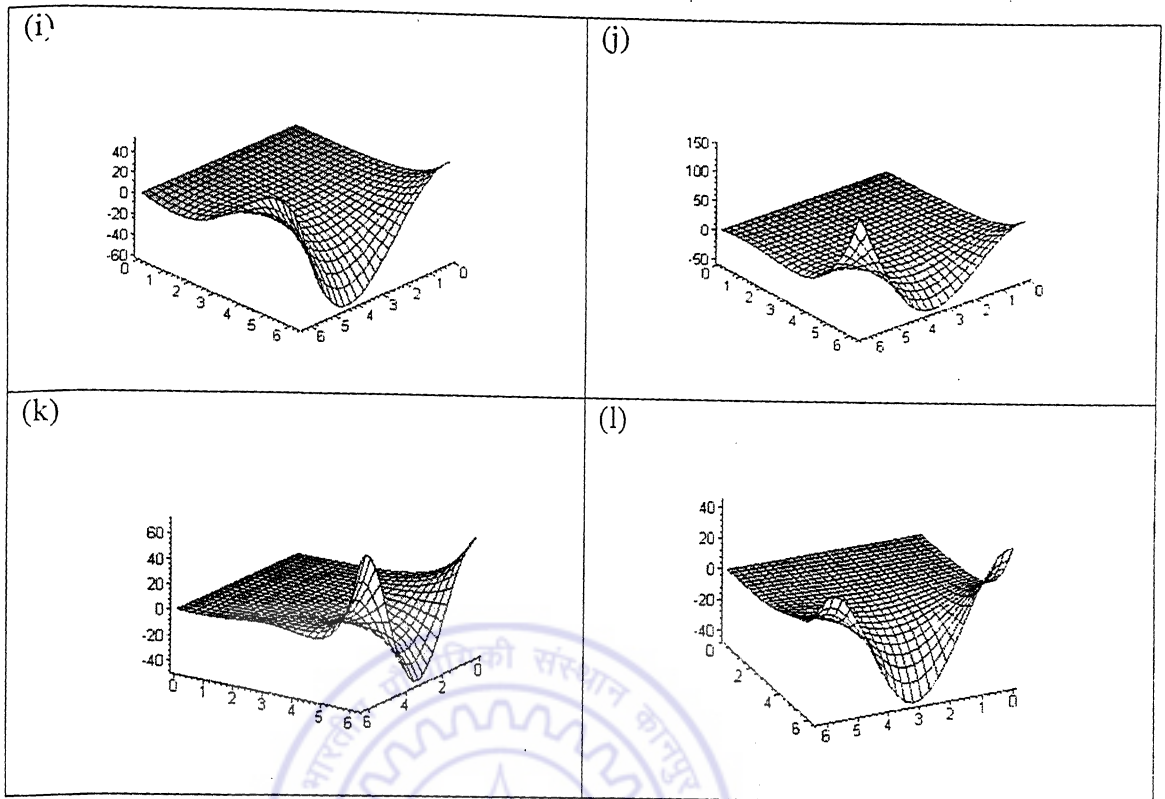


Table 3.14 Mapping $w=\sin(z_1)\sin(z_2)$ using EF based ANN and CNN

| Learning Rate was 0.1. Target Error was 0.0001 | | |
|--|--|--|
| Error Function | Real EF, Epochs (architecture: 4-5-2) | Complex EF Epochs (architecture: 2-5-1) |
| Absolute Error | 250 | 150 |
| Andrew Error | 1500 | 2000 |
| Cauchy Error | 250 | 700 |
| Error Fourth Order | 3000 | 2800 |
| Fair Error | 300 | 800 |
| Geman-McClure Error | 500 | 500 |
| Huber Error | 250 | 150 |
| Hyperbolic Squared | 3000 | 2500 |
| Bipolar Hyperbolic | 3000 | 2700 |
| LogCosh Error | 2000 | 1000 |
| Logarithmic Error | 3000 | 4000 |
| Mean Median Error | 400 | 200 |
| Minkowski Error | 3000 | 3000 |
| Quadratic Error | 250 | 150 |
| Sinh Error | 250 | 400 |
| Tukey Error | 2000 | 1500 |
| Welsh Error | 1900 | 2400 |

The complex mapping problem considered here is depicted in Fig. 3.4. The first complex number z_1 and the second z_2 are respectively the circles constructed by assigning

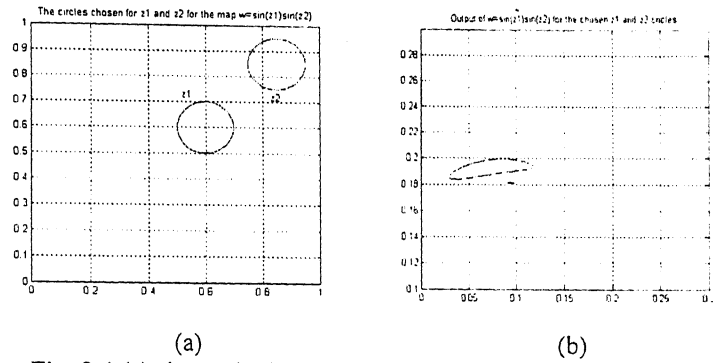


Fig. 3.4 (a) shows the input circles and (b) shows the image of the $w=\sin(z_1)\sin(z_2)$ map. All are normalized to within the unit square. This is the data set for $\sin(z_1)\sin(z_2)$ map.

complex constants to the equation

$$f = c_1 \exp(ik\theta) + c_2 \quad (3.47)$$

Specifically, the constants (c_1, c_2, k) for the two input circles (Fig. 3.4(a)) were: $(0.1, 0.6(1+i), 1)$ and $(0.1, 0.85(1+i), -1)$. EF based ANN and CNN have been used to solve the mapping problem. The real networks and the complex networks have been set to a learning rate of 0.1 with five neurons chosen in the hidden layer. The target error was set to 0.0001 and the average epochs to convergence are displayed in Table 3.14. The simulation of the network with points other than the ones chosen for training are displayed in Table 3.15.

Table 3.15 Mapping $w = \sin(z_1)\sin(z_2)$ using EF based CNNs. The Learn rate was 0.1, architecture was 1-5-1. Fig. 3.4(b) is mapped by the trained networks in this table. Dots are the network outputs

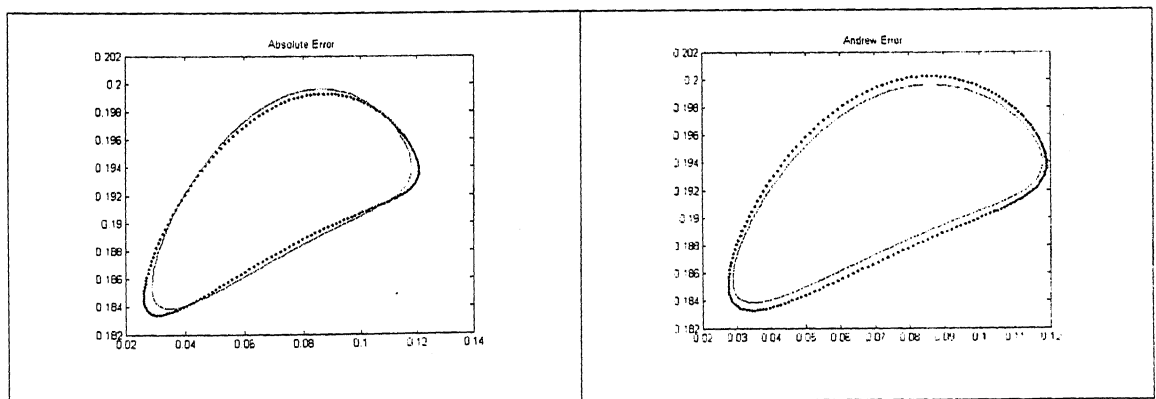


Table 3.15 Continued

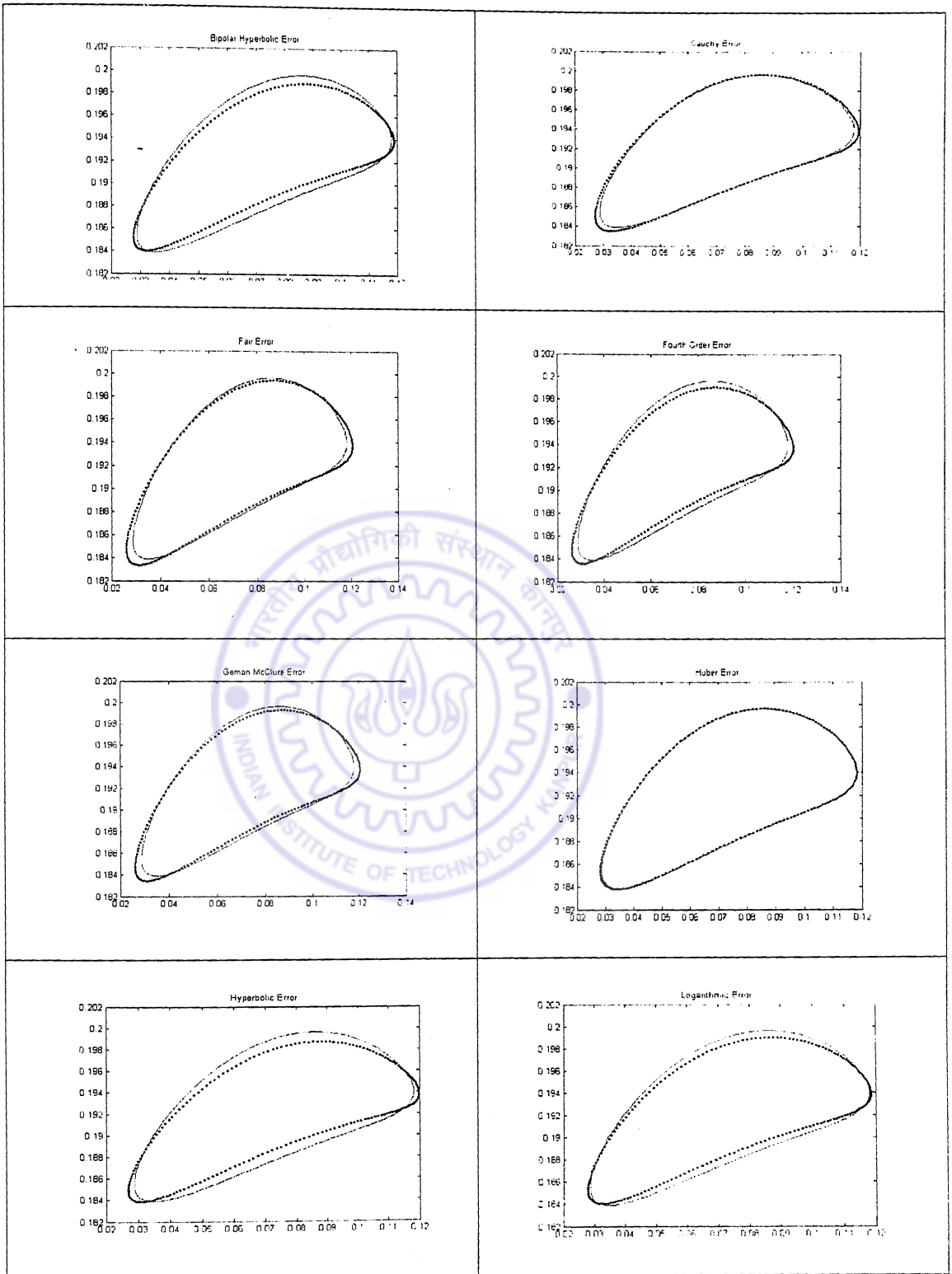
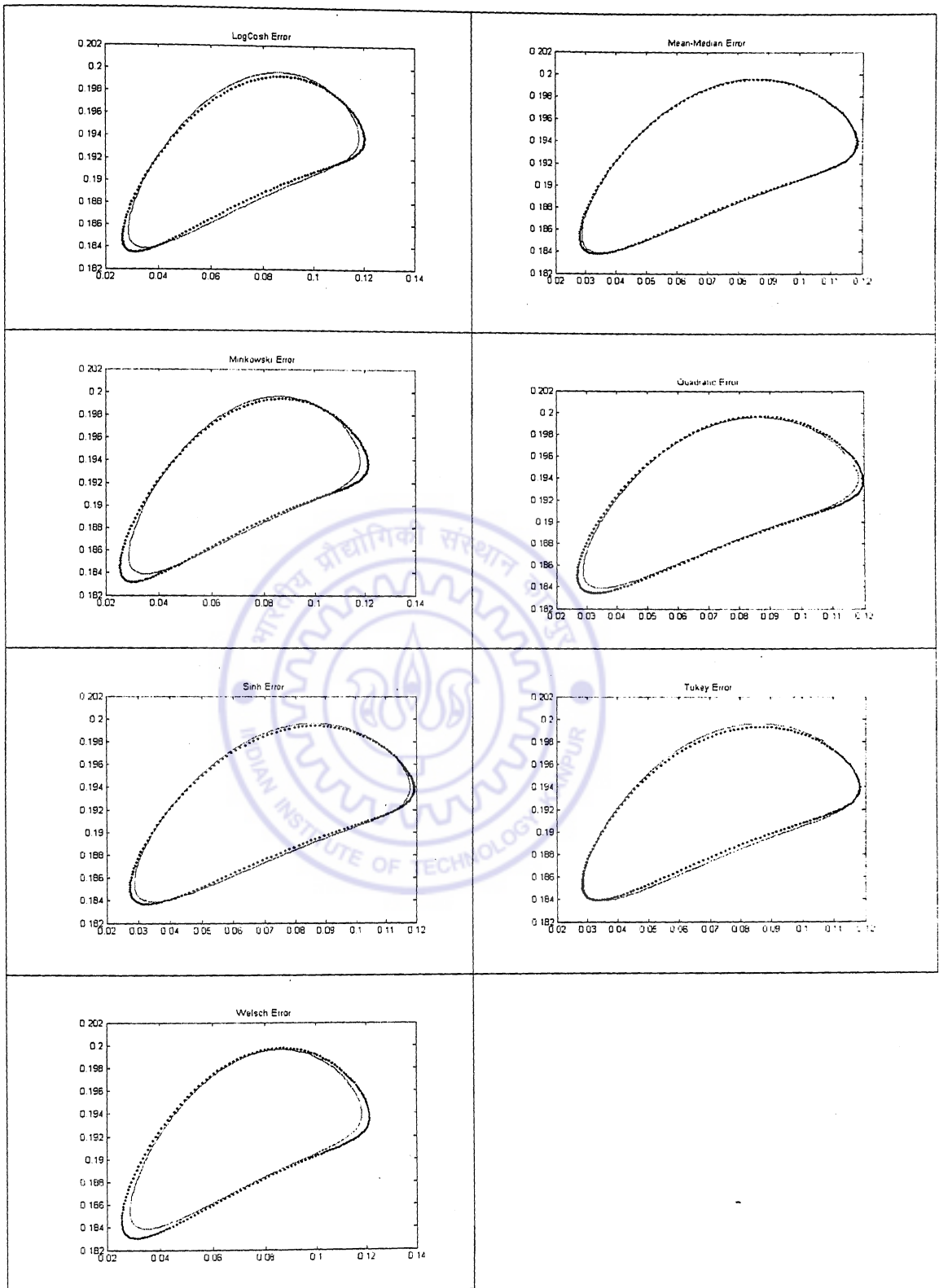


Table 3.15 Continued



It can be seen from Table 3.10 that the complex valued networks perform better than the real valued networks with the surface map, $z = \sin(x)\sin(y)$ when the argument of the function is in the range $[0, \pi/2]$. The real valued nets perform better when the argument

of the function is in the range $[0, 2\pi]$. In the extension to the complex domain of the same problem, where the arguments were chosen complex and the function extended to the complex domain as in eqn. (3.42), both real and complex networks captured the mapping but the differential rates at which each EF based network learned to map is summarized in Table 3.14 where the average epochs over three iterations are presented. As can be observed the EF based CNNs performed better than the EF based ANNs.

3.5.5 The Two-Spirals Problem

The Two-Spiral problem was posed by Wieland of the MITRE Corporation (Lang and Witbrock, 1988). This problem demands the design of a Neural Network that performs classification in a highly non-linear region. Two interlocking spirals each spanning 6π radians were sampled at 194 points. The task for the Neural Network is to map points on the different spirals to two different outputs and points in the vicinity of the spirals (and belonging to the interlocked space) to a correct class as belonging to the spiral in question. This was pronounced Benchmark because it was found to be an extremely hard problem for the algorithms of the BPA family to solve (Dagli, 1994). The following description gives a reason why the two intertwined spirals are a difficult mapping problem. The Archimedean Spirals are given by the polar equation

$$r = \theta \tag{3.38}$$

where r is the radius vector and θ is the polar angle. On re-writing in the Cartesian coordinates and adding the parameter to specify the initial angle of rise ϕ of the spiral, the equation takes the form

$$\begin{aligned} x &= \theta \cos(\theta + \phi) \\ y &= \theta \sin(\theta + \phi) \end{aligned} \tag{3.39}$$

Depending on the value assigned to the parameter ϕ , the spiral takes off from the origin making an initial angle of ϕ with the initial line on the polar plane. In the actual statement of the problem, the two spirals were chosen to be π radians apart. As the Neural Networks perform continuous mapping, the points in the vicinity of the spirals should also get correctly identified as ones lying close to the spiral in question. As a consequence, the region that the Neural Network should correctly identify is highly non-

linear as it closely follows the spirals. Fig. 3.5 shows a plot of the two interconnected linear spirals portraying the non-linear nature of the problem. A radial line drawn pierces the spirals alternately and hence for a fixed angle of the line in question, there are alternating patches of regions that belong to the spirals requiring the network to map them into different classes. As the angle of the radial line changes, the regions shift (according to the geometry of the spirals) along the line again requiring the map to be performed in an alternating fashion as belonging to the first or second spirals. This shifting-alternating nature of the problem requiring the map to be preserved makes it non-linear. The seventeen real EF based real valued networks solved the problem with both architectures at varying rates as shown in the table. The epochs were averaged out for three runs of the algorithm maintaining the initial weights identical in each run with the different EF based BPA. The target error was set to 0.0001. The standard BPA was used with the problem.

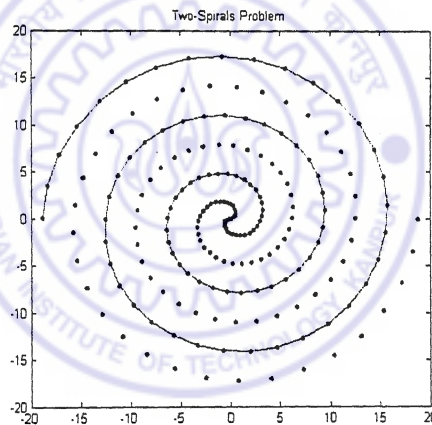


Fig. 3.5 The Problems of Two-Spirals requires telling apart the dotted spiral with the one shown in dots and dashes using a Back-Propagation trained Neural Networks

The runs were made with normalized data by dividing the inputs with the norm of the farthest point on the spirals and later shifting the spirals onto the unit square in the first quadrant of the xy - plane. The activation functions employed were bipolar sigmoid. However, with architectures 1-5-1 and 1-10-1, the CVBP produced saturation at 0.2 with Quadratic Error Function. The activation employed here was the Nitta function (eqn. (4.14)). With other EF like Cauchy, Mean-Median, Absolute, Fourth Power and Huber, the CVBP based training showed saturation for large values of epochs while with the rest of the EF's, the training was erratic. A typical convergence pattern with the real BPA is

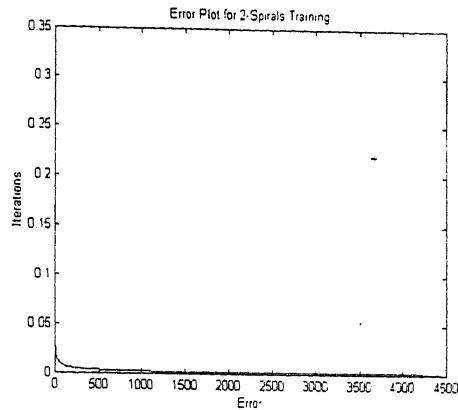


Fig. 3.6 Error convergence characteristic with the two spirals problem with real EF based BPA

displayed in Fig. 3.5. The test data consisted of points randomly picked from the segments of the spirals between the training points and also points in the vicinity of the spirals in the interlocked space between the spirals. A hundred test points were chosen and simulated with the networks designed. The table shows a perfect classification result with the real valued ANN's. The outputs set were $(-0.5, 0)$, $(0.5, 0)$ and accordingly the network's output maps to points in a close vicinity of these points. A second architecture (1-10-1) was also used with the problem to verify how the performance varied if the same were changed, as the CNN produced saturation with all the trials.

Table 3.16 The simulation result with real EF based BPA addressing the Two-Spirals problem is shown in the table. The architecture was 1-5-1. Learning rate was 0.1. The two targets are $(-0.5, 0)$ and $(0.5, 0)$. The point cluster around these points show that each EF based network correctly classified the test-data points constructed from intermediate points on spirals (bypassing the training points) and also points in a close vicinity of the spirals belonging in the inter-locking space.

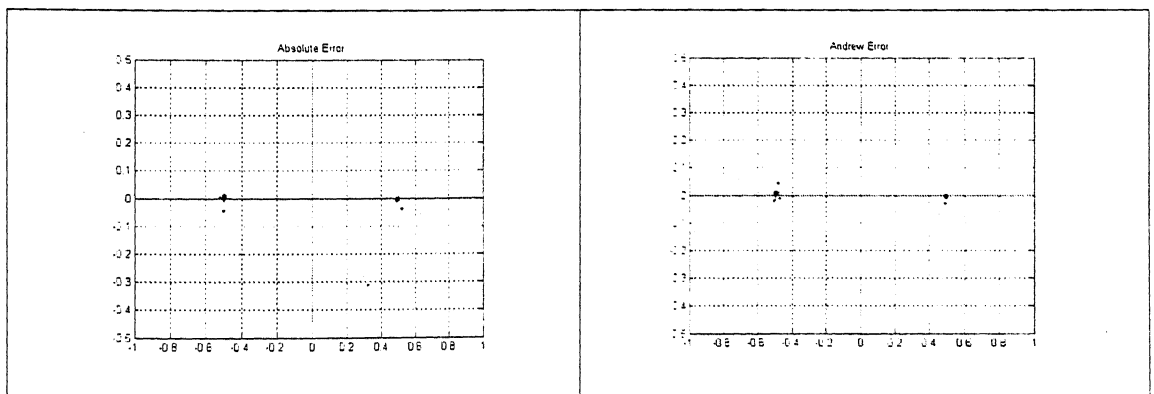


Table 3.16 Continued

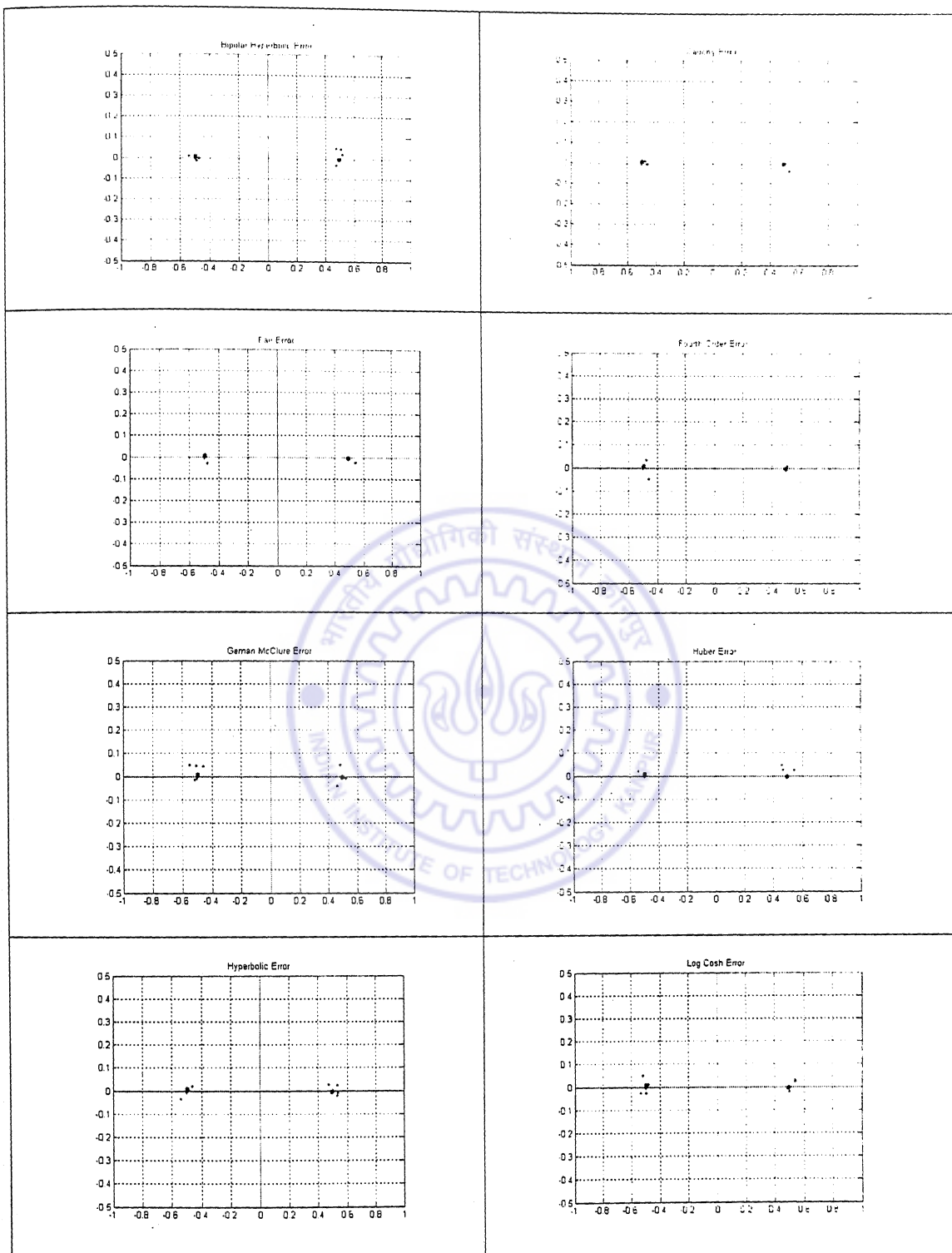
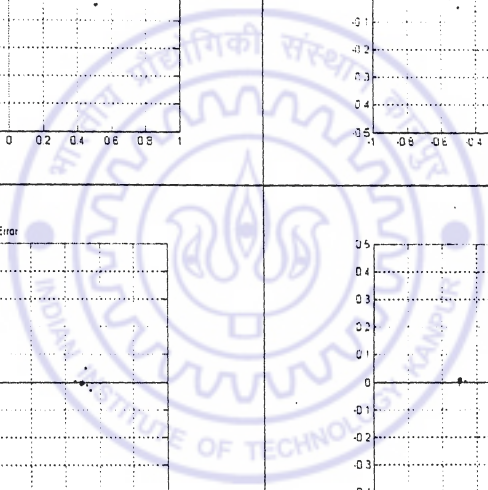
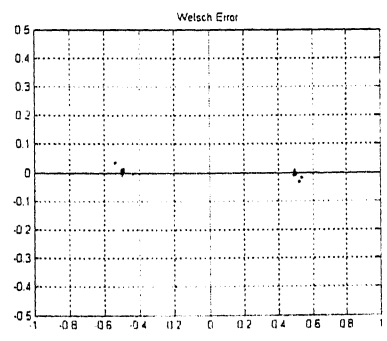
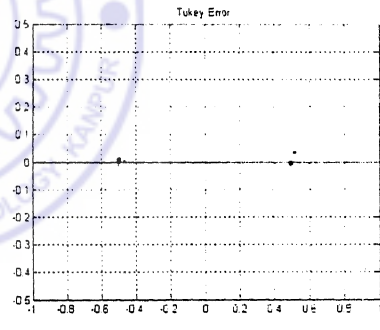
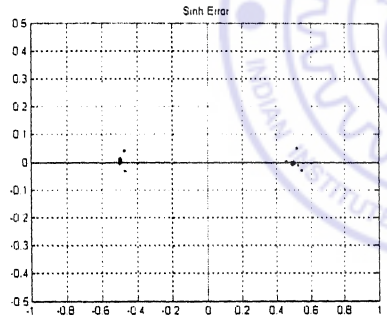
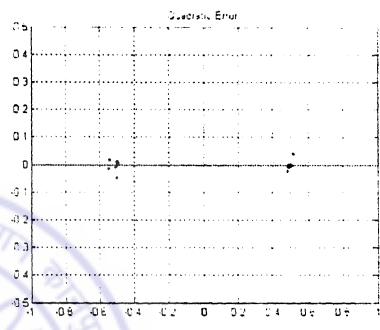
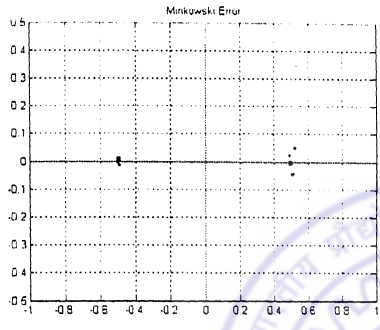
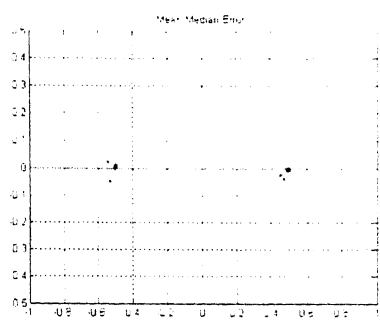
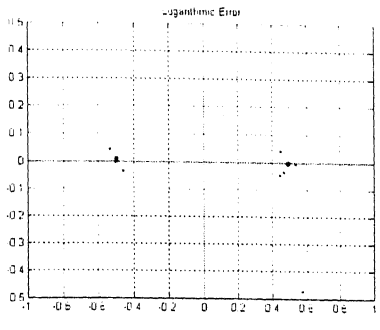


Table 3.16 Continued



Comparing the performances of BPA and CVBP for Two-Spirals Problem. The average epochs for convergence are shown for the BPA while the average saturation value and the average epochs for saturation are tabulated for the CVBP. Learning rate was 0.1.

| Average Result with Real and Complex Error Functions (1-5-1 Architecture) | | |
|--|---|-------------------------------|
| Error Function | Real EF, Epochs Target error: 0.0001 | Complex EF Saturation, Epochs |
| Absolute Error | 1500 | 0.3356, 10500 |
| Andrew Error | 8700 | 2.5467, 25000 |
| Cauchy Error | 1500 | 0.3231, 7500 |
| Error Fourth Order | 1700 | 0.5489, 8500 |
| Fair Error | 1000 | 1.3425, 11000 |
| Geman-McClure Error | 6600 | 4.5149, 15000 |
| Huber Error | 900 | 0.2956, 7500 |
| Hyperbolic Squared | 4200 | 1.3217, 14500 |
| Bipolar Hyperbolic | 7000 | 1.6650, 16000 |
| LogCosh Error | 3500 | 0.4328, 10000 |
| Logarithmic Error | 8700 | 2.3415, 15500 |
| Mean Median Error | 1500 | 0.2112, 7000 |
| Minkowski Error | 1900 | 0.5982, 8000 |
| Quadratic Error | 1000 | 0.2183, 8500 |
| Sinh Error | 3300 | 0.4358, 8000 |
| Tukey Error | 4200 | 1.3571, 10000 |
| Welsch Error | 5200 | 2.3141, 12000 |
| Average Result with Real and Complex Error Functions (1-10-1 Architecture) | | |
| Error Function | Real EF, Epochs | Complex EF, Epochs |
| Absolute Error | 1900 | 0.5356, 12900 |
| Andrew Error | 8700 | 3.5467, 21000 |
| Cauchy Error | 1400 | 0.3712, 7000 |
| Error Fourth Order | 2200 | 0.6181, 8700 |
| Fair Error | 1400 | 2.1563, 9500 |
| Geman-McClure Error | 7000 | 3.1154, 13500 |
| Huber Error | 1400 | 0.1587, 8000 |
| Hyperbolic Squared | 6300 | 2.0121, 11000 |
| Bipolar Hyperbolic | 7000 | 2.1112, 16500 |
| LogCosh Error | 2600 | 0.5227, 9500 |
| Logarithmic Error | 7700 | 2.3415, 15500 |
| Mean Median Error | 1925 | 0.2480, 6500 |
| Minkowski Error | 2200 | 0.6222, 7500 |
| Quadratic Error | 1000 | 0.2326, 8000 |
| Sinh Error | 3500 | 1.0332, 7000 |
| Tukey Error | 3100 | 1.9833, 9000 |
| Welsch Error | 6300 | 3.3849, 10000 |

The table shows the performance of 1-5-1 and 1-10-1 architectures. The real EF based networks captured the Two-Spirals mapping. The performance of the CVBP is tabulated for two architectures. As can be observed, the size of the architecture did not matter as far as the performance of the network was concerned. The EF based CNN's produced a saturation while ANN's learned to capture the mapping.

3.7 Conclusion

- The study reveals that the EF can indeed be treated as a parameter while employing ANN or CNN for training data. The various Benchmarks addressed in the chapter show that the Huber Error Function consistently worked well with most problems with both ANN and CNN.
- The other EF's that performed well and can replace the Quadratic Function depending on the application or requirement are: Absolute Error, Cauchy, Fair, Mean-Medain Error, Fair Error, Fourth Order and Log-Cosh EFs.
- The Sine Error Function is characterized by many maxima and minima and hence shall be useful only when the data set at hand is normalized to a region very close to one of the minima points of the sine curve. It can be seen that this EF based network needed more training runs as the algorithm requires the first few epochs to season the update process and reach to a stage that would ensure a decrease in the error on further training. This is equivalent to stating that the weights reached a point that would localize the training scheme on one of the troughs of the sinusoidal curve, which on further training results in a decrease of the sinusoidal error.
- The Geman-McClure Error, on the other hand, flattens out for larger values of the argument resulting in lower slope and lower updates and increased epochs for the complete training. The number of epochs to training depends basically on the effective slope zone of the Geman-McClure EF, which clearly would render a non-trivial update but once the argument doesn't fall in the flattened part of the EF, the values by which the weights update would be low. One can conclude from these analyses that the Andrew EF, Geman-McClure EF, Tukey and Welsch EFs shall need a prior analysis to fix up the initial weights so that the training could be effectively made.

With an arbitrary initial condition, the EFs consume some initial epochs of training to season the network down to a training course after which the training is more efficient. Some initial epochs are spent to steady the network so that the weights are brought to a point from where the training becomes efficient. This argument holds for both real and

complex valued EF's. Minkowski Error performs like the Fourth Power Error when the order is set to four. EFs like Logarithmic, Bipolar Hyperbolic and Hyperbolic Functions need normalization as their domains are defined only on a subset of the Real line. Hence in conclusion, Huber Function is recommended for completely replacing the Quadratic EF. In fact with a proper choice of the parameter c the function can outperform the QEF as it has the nice features of both QEF and the Absolute EF.



Chapter 4

Investigation of Activation Functions

The present chapter surveys the CAF proposed in literature. EF based CNNs are applied to the Benchmark Problems stated in Chapter 3. Based on the survey of the CAFs, a new CAF is proposed.

4.1 The Complex Activation Function

The function of activation in the neuron of the CNN is a complex valued function unlike the ANN where the functions were real valued. The sigmoid function is among the most frequently employed real valued activation functions. It was possible in the real variable case to tailor activation functions by choosing pieces of smaller curves and joining them at the end points (and establishing the continuity at these points) to obtain curves resembling the sigmoid that performed satisfactorily with training schemes. For instance, among the tailored activation functions is the construct obtained by the join of two straight lines with a stretched and shrunk sine curve. The curve constructed this way resembles the sigmoid; this was found to perform satisfactorily in many practical applications. It can be easily observed that this function satisfies the general set of conditions enumerated (Hassoun, 1995) for testing if a function could be used as function of activation.

It must be noted that such a construction is not possible in the complex valued setting. Equivalently complex activation functions can not be tailored as easily as the real activation functions could be. A first reason for the impediment is the complex activation involves a study of three-dimensional surfaces (as will be elaborated upon), the analyticity of which plays a vital role of course. The presence of singularities is a second important factor. These amounts to saying that the pieces of surfaces that are sewn must be made sure to have their individual parts aligned so as to clear analyticity condition at

each point along the seam so that the slopes match accordingly. As an example, to substantiate the paramount importance singularities can assume, the Haykin activation function given by

$$\frac{1}{(1 + \exp(-z))} \quad (4.1)$$

never converged in any of the experiments conducted by the author (incidentally, the same observation was reported by Nitta (1997)). A study into the behavior revealed that the presence of singularities greatly influenced the convergence. This function of activation has countably infinite singularities on the axis of ordinates (will be elaborated in a later section of this chapter). On the other hand, the complex plane imposes its own constraint in the form of Liouville Theorem which states that if a complex valued function is both analytic and bounded, it must be a constant function. As a ramification of the theorem, the constraints emerge.

To substantiate, on opening up a Neural Network shown with two neurons in the input layer, three in the hidden layer and one in the output layer (referred to as the 2-3-1 architecture), we get the following expression (assuming the output to be y , and activation functions in the three hidden neurons to be f_1, f_2, f_3 and the activation in the output neuron to be g)

$$y = g\left(\sum_{i=1}^3 v_{li} f_i\left(\sum_{j=1}^2 w_{ij} x_j + b_j\right)\right) \quad (4.2)$$

For the particular case of real sigmoid activation function in the hidden layer we get

$$y = g\left(\sum_{i=1}^3 v_{li} \frac{1}{1 + \exp\left(-\sum_{j=1}^2 w_{ij} x_j + b_j\right)}\right) \quad (4.3)$$

where w_{ij} and v_{ij} are respectively the input to hidden and hidden to output layer weights and b 's are biases.

If the networks were assumed complex, the Nitta activation function can be placed inside the second layer of the network defined in eqn. (4.2) above

$$f(z) = \frac{1}{1+e^{-x}} + i \frac{1}{1+e^{-y}}$$

The CVBP put forth by Leung and Haykin (1991) uses an activation function of the type

$$\frac{1}{(1 + \exp(-z))}$$

which is a straight forward extension of the sigmoidal function to the complex variable setting with steepness factor taken unity.

The weight update rule for the complex-variable based Neural Networks is

$$w_{n+1} = w_n + \Delta w_n \quad (4.4)$$

where each weight is complex valued, which can be written as

$$\begin{aligned} \text{Re}[w_{n+1}] &= \text{Re}[w_n] + \text{Re}[\Delta w_n] \\ \text{Im}[w_{n+1}] &= \text{Im}[w_n] + \text{Im}[\Delta w_n] \end{aligned} \quad (4.5)$$

The Learning Convergence Theorem for complex valued Neural Networks (Nitta, 1997) gives a way of updating the weights using the formulae (4.7)

$$\begin{aligned} \text{Re}[\Delta w_n] &= -\varepsilon A \nabla^{\text{Re}} r(z(w_n, x_n), y_n) \\ \text{Im}[\Delta w_n] &= -\varepsilon A \nabla^{\text{Im}} r(z(w_n, x_n), y_n) \end{aligned} \quad (4.6)$$

where A is any positive definite matrix (that is all the eigenvalues of A are positive real numbers, Hoffman and Kunze (1971)) and ε is a small positive constant called learning

constant, ∇^{Re} is the gradient with respect to the real part of w (the weight) and ∇^{Im} is the derivative with respect to the imaginary part of w . Let $P(x,y)$ be the unknown joint probability of two complex valued patterns x and y occurring from two different sources. The set $\{(x,y)\}$ corresponds to the learning pattern in Neural Networks. The purpose of learning is to estimate a complex valued pattern y that occurs from the second information source given that a complex valued pattern x occurred from the first information source. If $z(w,x)$ is a complex function giving an estimate of y where w is a parameter that corresponds to weights and thresholds in a neural network, $z(w,x)$ would then correspond to actual output pattern of the neural network (Nitta, 1997). Let r be the error function which represents an error that occurs when we give an estimate for the true complex-valued pattern y (r is a non-negative real function and not a complex function). The average error $R(w)$ defined by

$$R(w) = \sum_x \sum_y r(z(w,x), y) P(x, y) \quad (4.7)$$

is what is minimized in order to train the CNN. The Learning Convergence Theorem for the complex variables is an extension of the result of Amari (1967) where the result for the update rule in the real domain was reported.

The choice of activation function is extremely crucial for a complex variable based back-propagation algorithm for two reasons. First, the activation function in question is a two variable function (function of x and y), hence a surface in the three-dimensional space and therefore, the convergence study unlike the real variable based networks is more involved. Secondly the properties of the complex plane are different from those of the real line with additional constraints imposed by the properties of the complex plane. As the performance given by a Neural Network depends to a great extent on its activation function (as the approximation offered by a ANN or CNN is in terms of the corresponding activation function), a study of complex functions and their derivatives (because the weight update rule in eqns. (7), (8) involve computation of the derivatives) is important for its proper choice.

4.2 Study of Complex Activation Functions

The Complex Activation Functions with the complex weights as coefficients do the approximation of the data at hand. The Liouville Theorem constraint the complex plane imposes is studied here and the CAFs already proposed are surveyed.

4.2.1 Liouville Theorem

The complex plane unlike the real line is a two dimensional space. The second dimension adds flexibility and at the same time restricts the choice of activation functions for Neural Network applications by imposing certain constraints. More precisely, the important constraint imposed by the complex plane is epitomized in the Liouville Theorem which states that 'If a complex valued function is both analytic and bounded through out the complex plane, then it must be a constant function.' Hence, for a non-trivial complex-valued function, analyticity and boundedness cannot hold together. The contra-positive of the theorem puts it in the most usable form as it lists conditions that serve as search tools when one embarks on a search for complex activation functions. It states that a non-constant complex valued function is either non-analytic and bounded or is analytic and unbounded or non-analytic and unbounded. The three possibilities must be verified, as the new complex activation must clear this constraint. It hence follows that at least one of the above three conditions must be satisfied for otherwise the activation would turn out trivial (the constant complex function). Equivalently, if a non-trivial complex valued function is analytic it must go unbounded at at least one point on the complex plane and if the function is bounded it must be non-analytic in some region for it to qualify as activation function. Hence a search for activation function should make sure these conditions are satisfied. The second dimension of the complex plane necessitates a study of three-dimensional surfaces, as the real and imaginary parts of the complex activation functions are both functions of x and y .

4.2.2 Complex Activation Functions and their Derivatives

The CAFs already used in literature are listed here and their derivatives studied.

4.2.2.1 Nitta Activation Function and its Derivatives

The Nitta Activation is defined by the following equation

$$f(z) = \frac{1}{(1 + \exp(-x))} + i \frac{1}{(1 + \exp(-y))} \quad (4.8)$$

As the real part of the Nitta activation is independent of y and the imaginary part is independent of x , the derivative of the Nitta Activation with respect to the real part of the argument, x is

$$\frac{\exp(-x)}{(1 + \exp(-x))^2} \quad (4.9)$$

in which the imaginary part doesn't appear, consequently the imaginary part of the derivative is zero. The derivative of the Nitta function with respect to the imaginary part, y is

$$i \frac{\exp(-y)}{(1 + \exp(-y))^2} \quad (4.10)$$

in which the real part doesn't appear.

It can be easily seen that the function doesn't have any singular points, nor do the derivatives possess any (as exponential function is always positive). The characteristic of the function is the imaginary and real parts of the weights are separate and appear in the real and imaginary parts of the function separately. The surface plots of the real and imaginary parts of the function and its derivatives are shown.

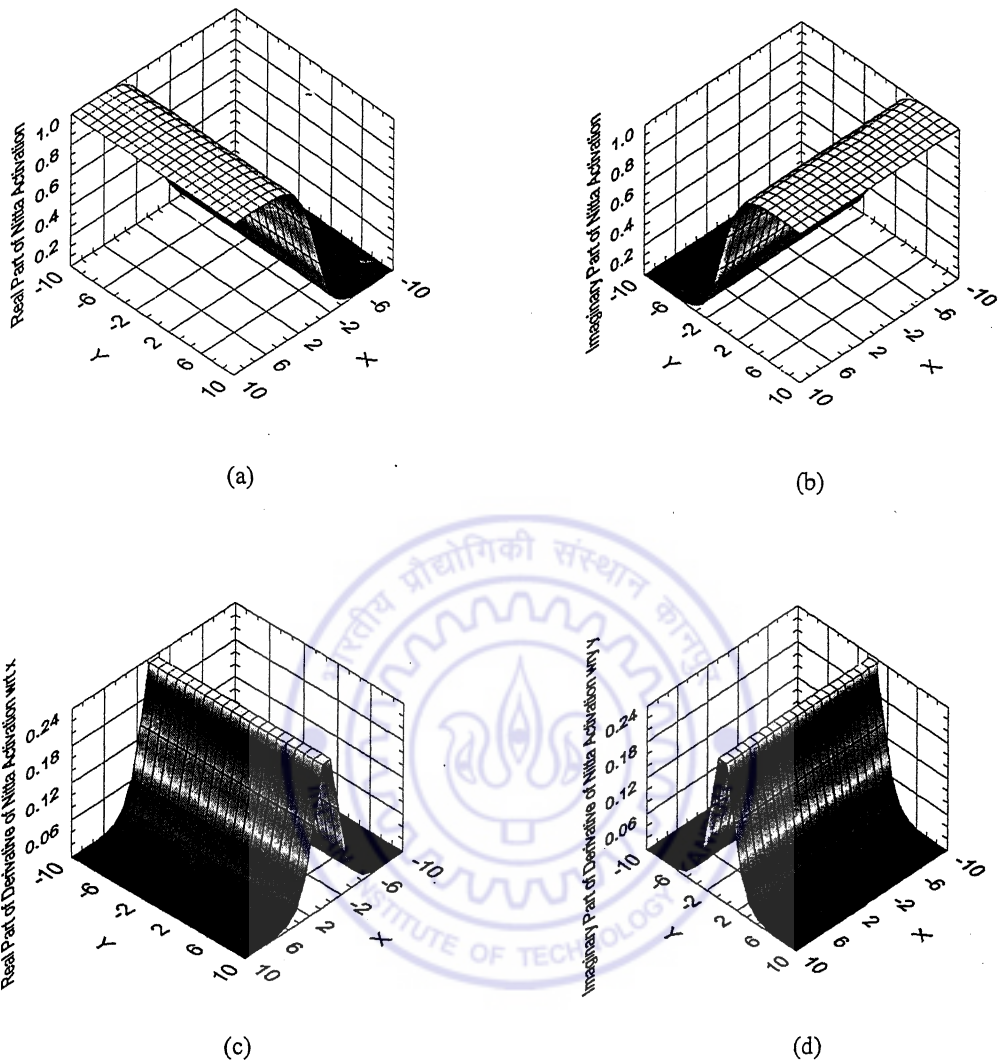


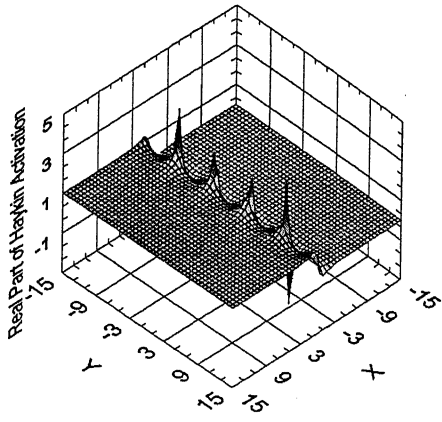
Fig. 4.1 (a) Real Part of Nitta Activation, (b) Imaginary part of Nitta Activation (c) Real part of the derivative with respect to x (d) Imaginary part of the derivative with respect to y .

4.2.2.2 Haykin Activation Function

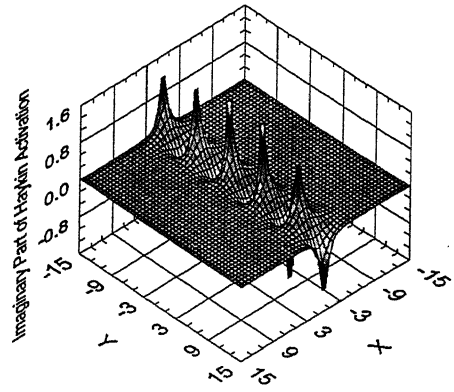
The Haykin Activation function is defined by the following equation

$$f(z) = \frac{1}{(1 + \exp(-z))} \quad (4.11)$$

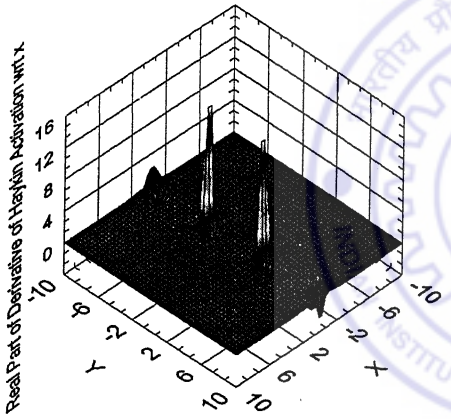
with $z = x + iy$. On breaking the function up into its real and imaginary parts, the RHS takes the following form



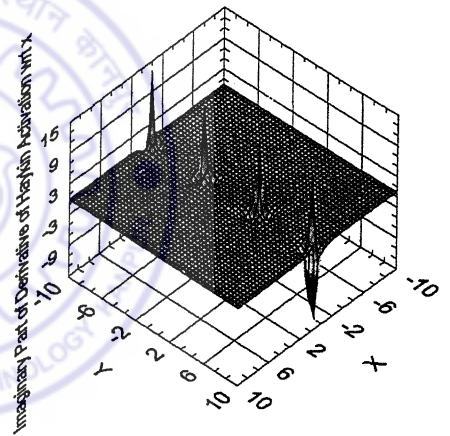
(a)



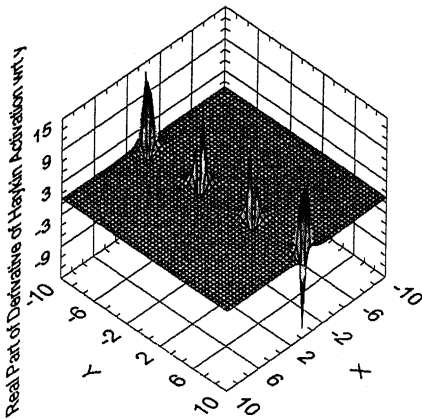
(b)



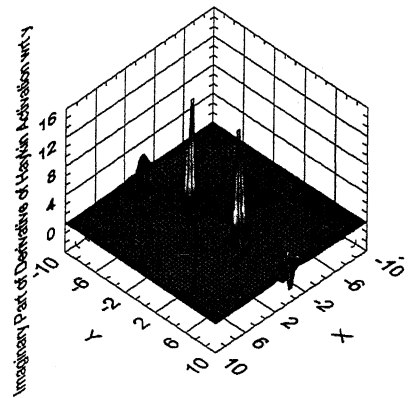
(c)



(d)



(e)



(f)

Fig. 4.2 (a) Real part of Haykin activation (b) Imaginary part of Haykin activation (c) Real part of the derivative with respect to x (d) Imaginary part of the derivative with respect to x (e) Real part of the derivative with respect to y (f) Imaginary part of the derivative with respect to y

$$\frac{(1 + \exp(-x)\cos(y)) + i(\exp(-x)\sin(y))}{(1 + \exp(-2x) + 2\exp(-x)\cos(y))} \quad (4.12)$$

On differentiating the function with respect to the real and imaginary parts separately we get respectively the following expressions

$$-f(z)(1-f(z)) \quad \text{and} \quad (4.13a)$$

$$-if(z)(1-f(z)) \quad (4.13b)$$

On substituting the definition of the activation function and simplifying the expressions above, we find the derivative of the activation with respect to the real part is

$$\frac{-(\exp(-x)\cos(y) + 2\exp(-2x) + \exp(-3x)\cos(y)) + i(\exp(-x) - \exp(-3x))\sin(y)}{(1 + \exp(-4x) + 4\exp(-2x) + 2\exp(-2x)\cos(2y) + 4\exp(-3x)\cos(y) + 4\exp(-x)\cos(y))} \quad (4.14)$$

and the derivative with respect to the imaginary part is

$$\frac{-(\exp(-x) - \exp(-3x))\sin(y) - i(\exp(-x)\cos(y) + 2\exp(-2x) + \exp(-3x)\cos(y)) +}{(1 + \exp(-4x) + 4\exp(-2x) + 2\exp(-2x)\cos(2y) + 4\exp(-3x)\cos(y) + 4\exp(-x)\cos(y))} \quad (4.15)$$

4.2.2.3 Haykin Activation and Singularities

The position of singularities disturb the training scheme as whenever some intermediate weights fall in the vicinity of the singular points, it was observed that the whole training process down the line receives a jolt. This is revealed by the error plot of the function is characterized by peaks. The typical point scatter shown in the figure is a distribution of the hidden layer weights as the training process is on. As can be observed, they cluster around some singular points which eventually results in the peak type error-epochs characteristic. The typical EF graph with Haykin activation is shown in figure.

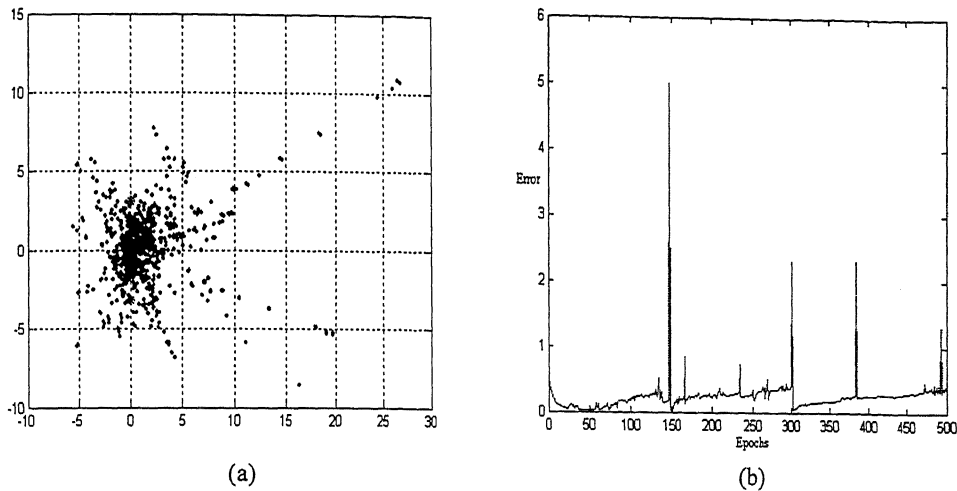


Fig. 4.3 (a) Four singular points of the Haykin Activation Function (explicitly, $(0, \pi)$, $(0, -\pi)$, $(0, 3\pi)$, $(0, -3\pi)$) have been fully engulfed by the point scatter. The points $(0, 3\pi)$ and $(0, -3\pi)$ are shown encircled while the other two points are completely inside the cloud of points. (b) shows the peaks formed during a run of training.

The figure shows four singular points of the Haykin activation that are completely engulfed by the cloud of points. The training process produced many peaks as a result of the singular activation configurations encountered because of the activation function's singular points. The Haykin function has been employed with all the problems addressed in the chapter but didn't produce a satisfactory result owing to the reason just mentioned. On equating the denominator of eqn (4.12) to zero, one observes that the function goes unbounded at points of the type

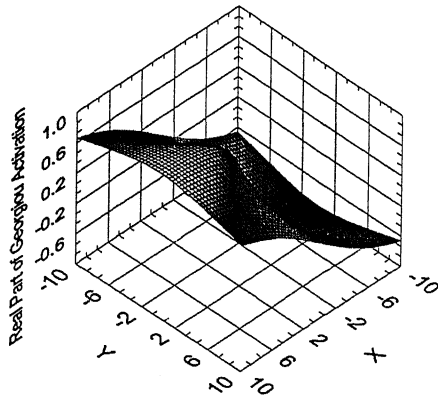
$$(0, (2n+1)\pi) \quad (4.16)$$

for n a Natural number.

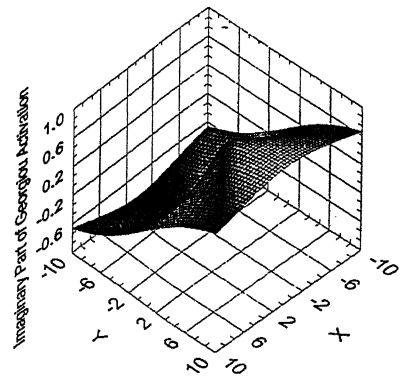
4.2.2.4 Georgiou Activation

The Georgiou Activation is defined by the equation

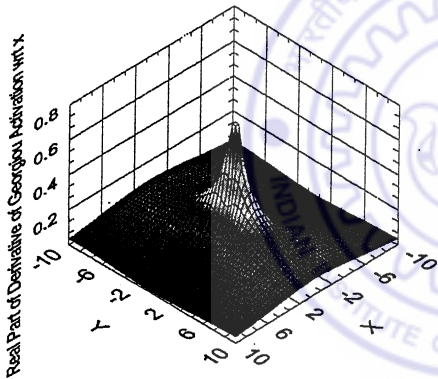
$$f(z) = \frac{cz}{\left(1 + \frac{1}{r}|z|\right)} \quad (4.17)$$



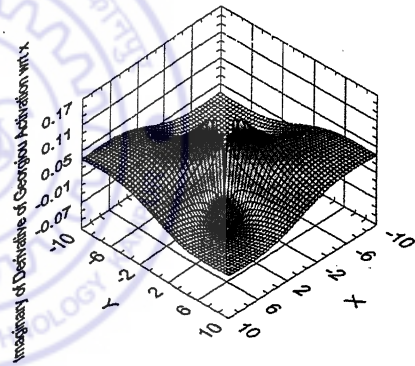
(a)



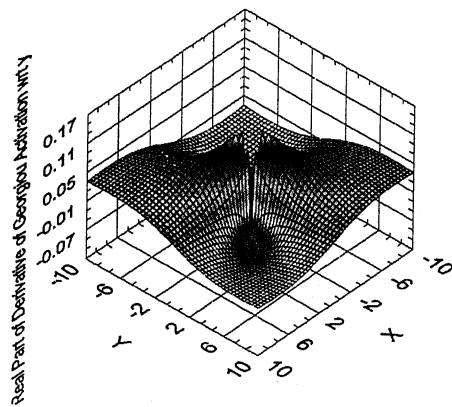
(b)



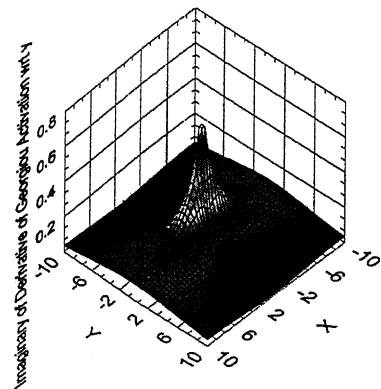
(c)



(d)



(e)



(f)

Fig. 4.4 (a) Real part of Georgiou activation (b) Imaginary part of Georgiou activation (c) Real part of the derivative with respect to x (d) Imaginary part of the derivative with respect to x (e) Real part of the derivative with respect to y (f) Imaginary part of the derivative with respect to y

The c and r parameters have been set equal to unity in the study conducted in this chapter. The derivatives of this function with respect to the real and imaginary parts of z respectively are

$$\frac{(|z| + y^2) - ixy}{|z|(1 + |z|)^2} \quad (4.18)$$

$$\frac{-xy + i(|z| + y^2)}{|z|(1 + |z|)^2} \quad (4.19)$$

Unlike the Haykin activation, the Georgiou activation doesn't go unbounded in its domain of definition (the complex plane).

4.2.2.5 The New Activation Function

The new activation function is defined by the equation

$$f(z) = \frac{x}{(1 + |x|)} + i \frac{y}{(1 + |y|)} \quad (4.20)$$

The function is free from singular points as there exist no points where the function can go unbounded. Moreover, the real and imaginary parts are functions of a single variable. The derivatives of this activation with respect to the real and imaginary parts respectively are

$$\frac{1}{(1 + |x|)^2} \quad \text{and} \quad (4.21)$$

$$i \frac{1}{(1 + |y|)^2} \quad (4.22)$$

The Activation functions surveyed have been coupled with each of the complex EFs analyzed in the preceding chapter. The problems addressed were the ones enumerated

there, the theme of the work however was comparing the performance of Activation functions. Table 4.1 displays the result where the number of epochs to convergence are shown with respect to each problem, by setting the other parameters identical. The new complex

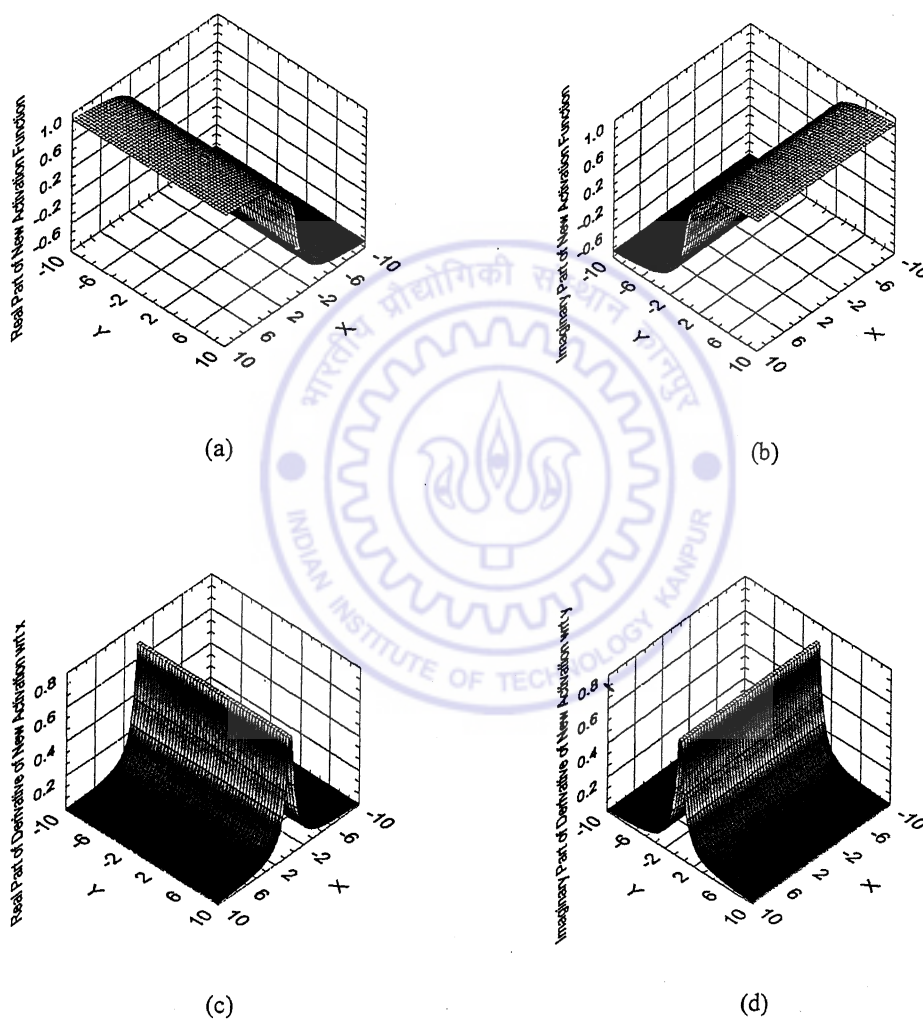


Fig. 4.5 (a) Real part of the New Activation Function (b) Imaginary part of New Activation Function (c) Real part of the derivative with respect to x (d) Imaginary part of the derivative with respect to y .

Activation function proposed here can be seen to have performed better in some cases as compared with the well-known and accepted activation functions of the CNN theory. The

proposed function is simpler than the Nitta function that by far produced the best result among the complex activation functions.

4.3 Conclusion

The various EF based CNN's have been applied to the problems addressed in Chapter 3. The results of the runs have been displayed in Table 4.1. Comparison has been drawn between various complex valued activation functions by subjecting the algorithm based on each of them to a training process. The various EF have been used in the process. The problems addressed are Complex-XOR, Complex CODEC, 3-Parity, $w=\sin(x)\sin(y)$. The learning rate was kept fixed at 0.1, each experiment was carried out thrice and the average epochs to convergence have been tabulated by rounding them to the nearest perfect figures. It can be observed from the table that the Huber, Cauchy, Mean-Median functions have performed on par with the Quadratic Function. The other conclusions that follow from the tabulated results are, the EF is indeed a factor that determines the number of epochs to convergence and a combination EF and activation function can optimize the training scheme substantially.

It can be said that the EF based training scheme with a proper choice of activation function can substantially decrease the epochs for the problem in question. The importance of designing a learning scheme to optimize a training process for the problem at hand immediately comes up in the light of the EF and activation functions study the present chapter reports. To optimize a learning scheme, sticking to one EF and a single activation may not be the best; instead choosing a set of EF and activation function combinations to better the scheme. For instance, if the choice of Absolute Error function with Georgiou activation is heading for a saturation, switch to a Huber function with Nitta activation from the point forward or any other suitable combination to bypass the same. Table 4.1 can be taken as a basis for making these choices and designing a learning scheme. This approach will clearly offer more tooling while designing Neural Networks as firstly, it departs from the present day technique of using the Quadratic Error Function and an activation and invoking the procedure of weight update employing a gradient descent, which in practice doesn't manage to go below a certain value of the error

(although theory assures that an NN exists for an arbitrary error that the investigator may desire). Secondly, the choice of EF and activation function chosen in the training sequence circumvents some of the existing lacunae such as error getting stuck and not progressing below a certain value.



Table 4.1 The average epochs for convergence with various activation functions based CNN, across Error Functions. Architecture was 1-5-1, Learning Rate was 0.1, Target Error was 0.0001.

| EF | CXOR | | | CODEC (0.0001) | | | Complex 3-PARITY | | | $w=\sin(x)\sin(y)$ | | |
|---------|-------|-------|-------|----------------|-------|-------|------------------|-------|-------|--------------------|-------|-------|
| | NTA | GOR | NEW | NTA | GOR | NEW | NTA | GOR | NEW | NTA | GOR | NEW |
| ABS | 12100 | 12000 | 10000 | 1600 | 3500 | 3000 | 4500 | 4000 | 3000 | 6200 | 10000 | 8000 |
| ANDREW | 20300 | 15000 | 15000 | 22000 | 10000 | 8000 | 10000 | 11000 | 13000 | 11000 | 13000 | 19000 |
| BIPHYP | 12900 | 15000 | 15000 | 19000 | 9000 | 8000 | 9000 | 10000 | 10000 | 4000 | 6000 | 4000 |
| CAUCHY | 11000 | 14000 | 15000 | 2000 | 2500 | 2500 | 3000 | 3000 | 3500 | 5500 | 6000 | 5000 |
| FAIR | 15100 | 17000 | 19200 | 10000 | 3000 | 3500 | 4000 | 6000 | 6500 | 5000 | 9000 | 5000 |
| FOURTH | 15000 | 18000 | 17000 | 8000 | 4000 | 3500 | 3500 | 4000 | 3500 | 3500 | 5000 | 4000 |
| GEMAN | 10000 | 17000 | 20000 | 4000 | 7000 | 4000 | 6000 | 8000 | 9000 | 3000 | 10000 | 8500 |
| HUBER | 10000 | 17000 | 14000 | 1700 | 4000 | 5000 | 2500 | 2000 | 2000 | 3000 | 4000 | 5000 |
| HYPERB | 20200 | 19000 | 21000 | 17000 | 6000 | 7000 | 7000 | 7500 | 6000 | 15000 | 15000 | 10000 |
| LOGTHM | 12000 | 23000 | 21000 | 19000 | 11000 | 12000 | 8000 | 10000 | 7000 | 3000 | 10000 | 7000 |
| LOGCOSH | 20000 | 17000 | 14000 | 7700 | 2500 | 2500 | 4000 | 4000 | 5000 | 11000 | 6000 | 5000 |
| MNMEM | 12000 | 21000 | 18000 | 7000 | 4500 | 3000 | 5000 | 4000 | 4000 | 4500 | 6000 | 5000 |
| MINK | 12100 | 19000 | 19000 | 5000 | 5000 | 6000 | 5000 | 5500 | 4000 | 6000 | 5500 | 5000 |
| QUADR | 10000 | 17000 | 15000 | 3500 | 2500 | 2000 | 3000 | 2000 | 2500 | 3500 | 4000 | 3000 |
| SINH | 14200 | 19000 | 20000 | 2000 | 5000 | 5000 | 6000 | 6500 | 6000 | 5000 | 5000 | 4000 |
| TUKEY | 18100 | 20000 | 18000 | 2500 | 6000 | 5000 | 6000 | 8000 | 9000 | 7000 | 9000 | 5000 |
| WELSCH | 17000 | 16000 | 14000 | 13000 | 6000 | 4000 | 7000 | 7000 | 6000 | 6000 | 8500 | 7000 |

Chapter 5

Application to Mapping Problems

The mapping properties of EF based CVBP are studied in the present chapter. Some well-known maps that appear frequently in application have been considered here. The learning constant was kept fixed so also the initial weights and the number of epochs so that the EF's contribution to the CNN's performance can be compared.

5.1 Mapping Properties of Neural Networks

A function is an association that maps points in a domain to points in the range. Many properties are associated with functional maps – continuity, differentiability and compactness among many others. In most problems of practical interest, the actual function that governs the input-output behavior is unknown as it becomes increasingly difficult to treat the differential equations that govern the system as the number of variables increase or the equation becomes highly non-linear. However, Neural Networks can be taught to perform the mapping by treating the dynamical system as a black-box and collecting the input and the corresponding output values and subjecting the network to training based on these data points. It was shown recently (Amir *et. al.*, 1997) that initial condition is the most important factor that affects the generalization performance of the Neural Network design, the other parameters being complexity of the network, small and large momentum rates in that order. In practical applications hence, it is the mapping properties of the Neural Network that are put to use. In this thesis, the validation of the CNN against the benchmarks was shown earlier while the ability of these networks to map the dynamics of the problems at hand is the step of practical importance.

The mapping properties of the CNN are the subject of the present chapter. In actual application, the form of the mapping is unknown but BPA based Neural Network captures the mapping (subject to Kolmogorov conditions (Kolmogorov, 1957)). The Learning Convergence Theorem for complex variables (Nitta, 1997) is the assurance one

needs to establish complex weights exist that solve the mapping problem at hand. To reach to the point in the weights space, the process of training based on gradient descent is employed. The method involves computing the complex gradient and updating the weights based on the slope as obtained from the gradient formula.

The BPA based Neural Networks have been used extensively to tackle the problems posed by the industry. On the other hand, the CNN's mapping properties *per se* haven't been studied in literature as yet. Nitta (1997) reported some problems of mapping to bring forth the differences between CNN and ANN where the stress was on problems that CNN solves and ANN doesn't. The present chapter investigates and explores the mapping properties of the CNN. The Error Function based Networks were arranged in a performance echelon for each problem studied. Some problems of interest that were pointed out in literature (Nitta, 1997) were extended to include the maps of a more general nature. In the present chapter, the number of epochs has been kept fixed for most of the maps considered.

5.2 CNN Applied to Mapping Problems

The mapping properties of CNN are explored in the present section. Some well-known and frequently applied transformations are captured using the CNNs.

5.2.1 The Bilinear Transformation

The transformation affected by the formula

$$w = \frac{az + b}{cz + d} \quad (5.1)$$

is the Bilinear Transformation that maps circles on the z -plane to circles on the w -plane. The map can be thought of as a composition of several elementary maps as the following step shows (Churchill and Brown, 1993),

$$\frac{(az + b)}{(cz + d)} = \frac{a}{c} + \frac{(b/a - ad/c^2)}{(z + d/c)} \quad (5.2)$$

from which it is clear that for z restricted to lie on a circle, the rational function that defines the bilinear transformation maps its image onto a circle. As the break up shown above is a sequence of elementary maps (shifting by a constant and reciprocal maps), the circle retains its form through the transformation as they make up the transformation shift the circle, shrink or expand it, perform a reciprocal map and offset the resultant. These transformations preserve the circle. It should however be noted that the determinant

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} \quad (5.3)$$

should not vanish for otherwise the map collapses to one that restricts the image to lie on a straight line passing through the origin. It can be clearly seen that the Bilinear Transformation is a generalization of the Similarity Transformation obtained by assigning null values to the parameters b and c (eqn. 5.1).

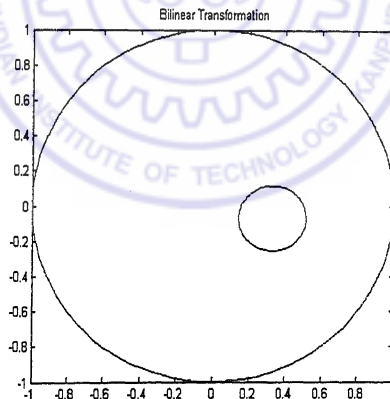


Fig. 5.1 Bilinear Transformation maps circles to circles.

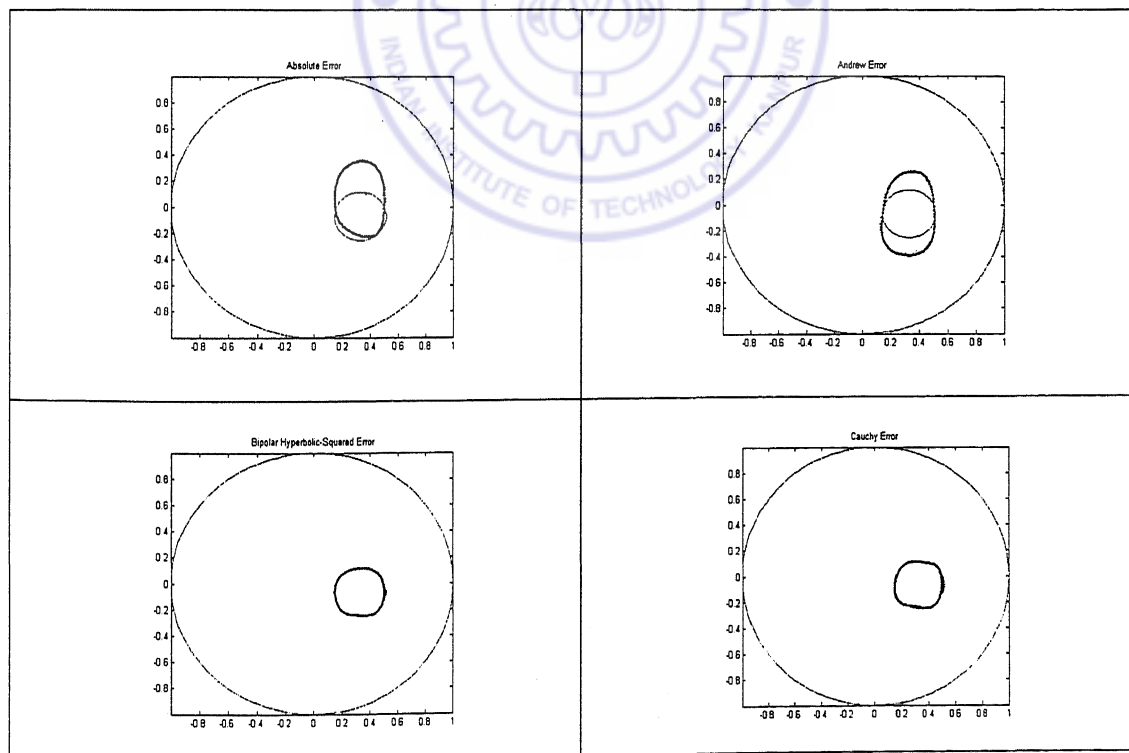
For the present experiment, the Bilinear Transformation considered was

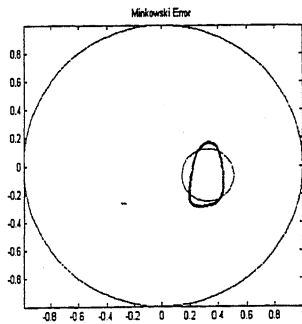
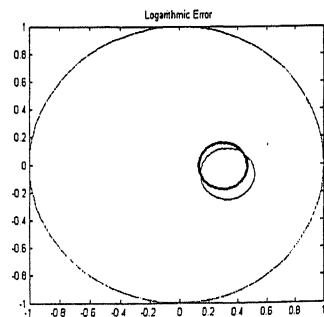
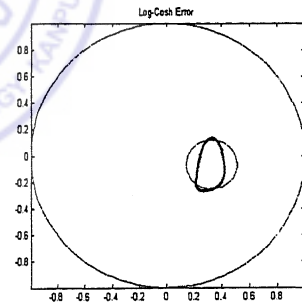
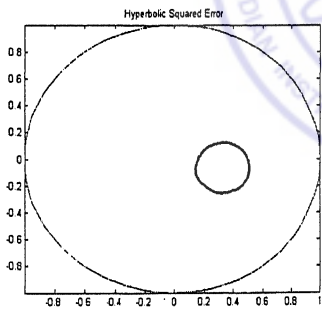
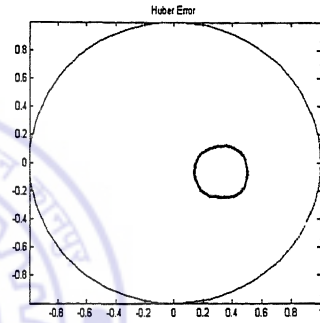
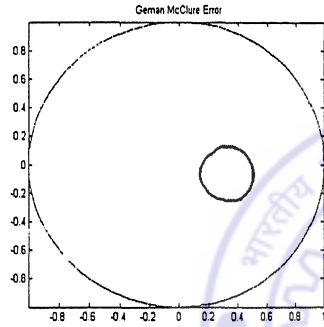
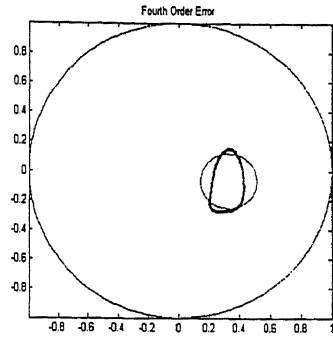
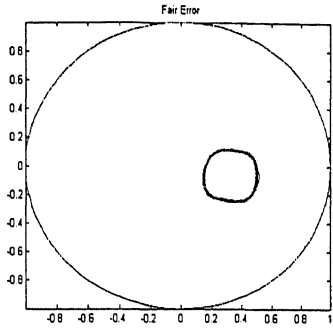
$$w = \frac{(0.2z + 0.2 + 0.3i)}{(i + 0.4)}$$

In real world, this mapping is used to study viscous flow across bodies with elliptic cross-section. As Error Function based CNN learns to capture the transformation, these networks can be employed to perform the map the vector fields in the flow past a cylinder.

The complex mapping was found to be sensitive to the normalization of the input and output patterns. The CAF demand the data to be mapped be restricted to a range for otherwise, the functions' effective contribution to the weights diminishes for larger values of the argument (as the functions become more flat for large arguments and hence less slope). This restriction for implementing the CNN algorithms with the Bilinear Transformation manifest in the form of a constrained range on the parameters a , b , c and d . For circles of a large radius and lying beyond the unit circle, a normalization factor must be introduced to restrict the whole output to within the unit circle before using the CNN with these activation functions. Architecture chosen was 1-5-1, learning rate was 0.1, epochs was 1500. The Nitta and New activation based algorithms were used to solve the problem. The target error was set to 0.000001.

Table 5.1 Bilinear Transformation using Nitta Activation Function. 1-5-1 architecture, 1500 epochs, Learning rate = 0.1. Epochs criterion was chosen while training.





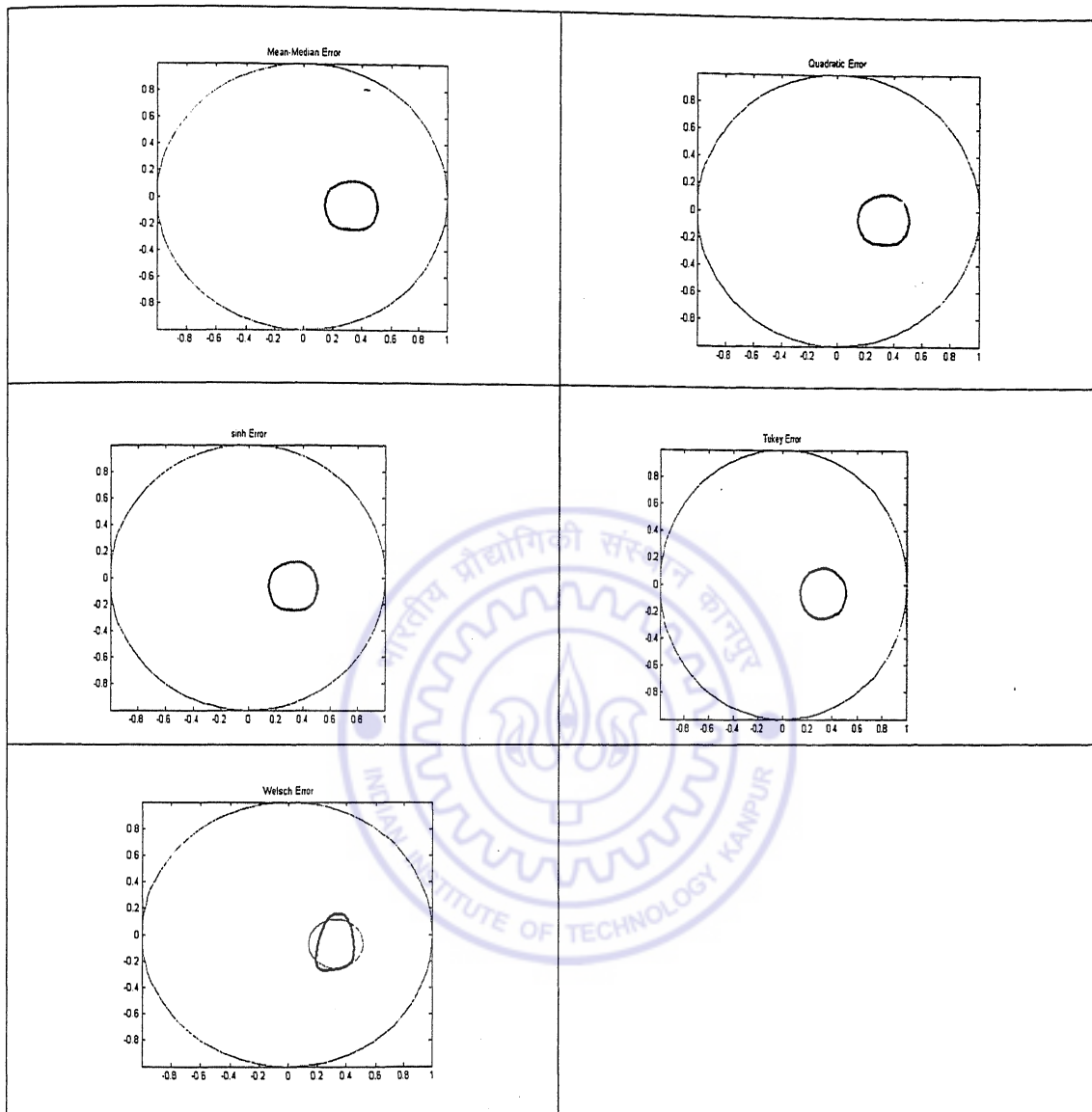
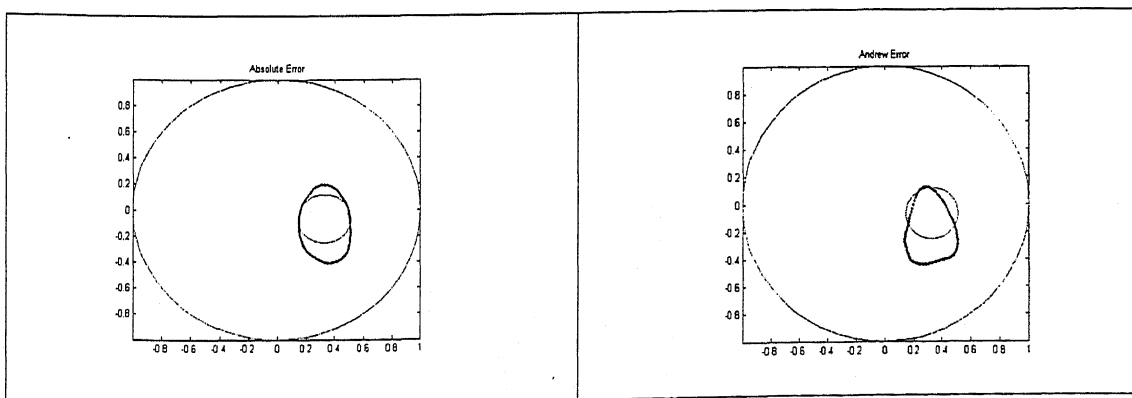
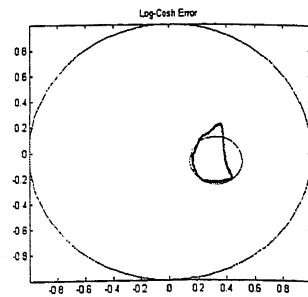
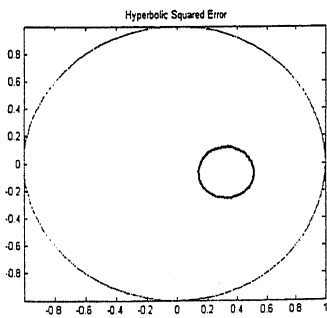
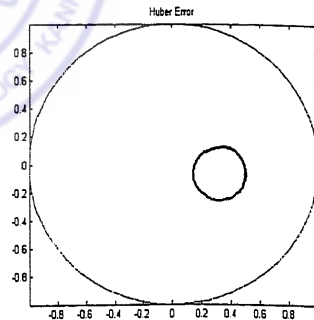
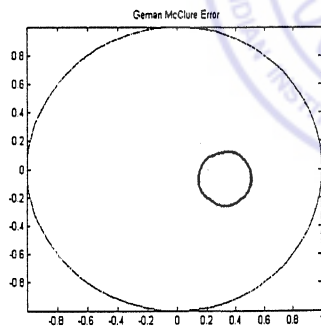
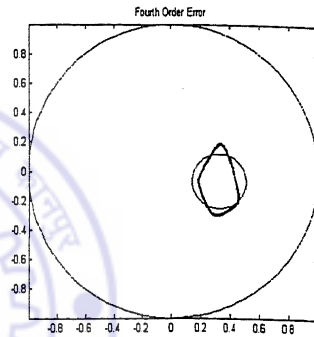
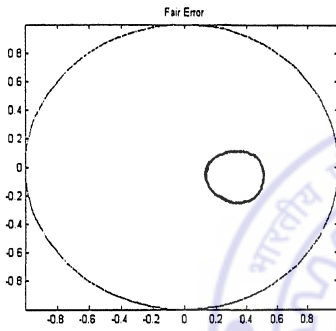
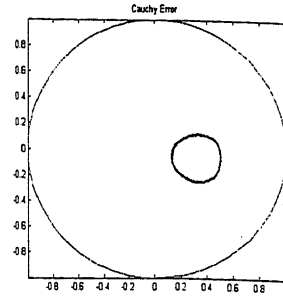
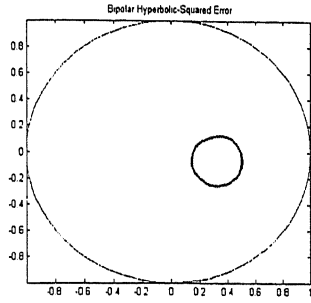
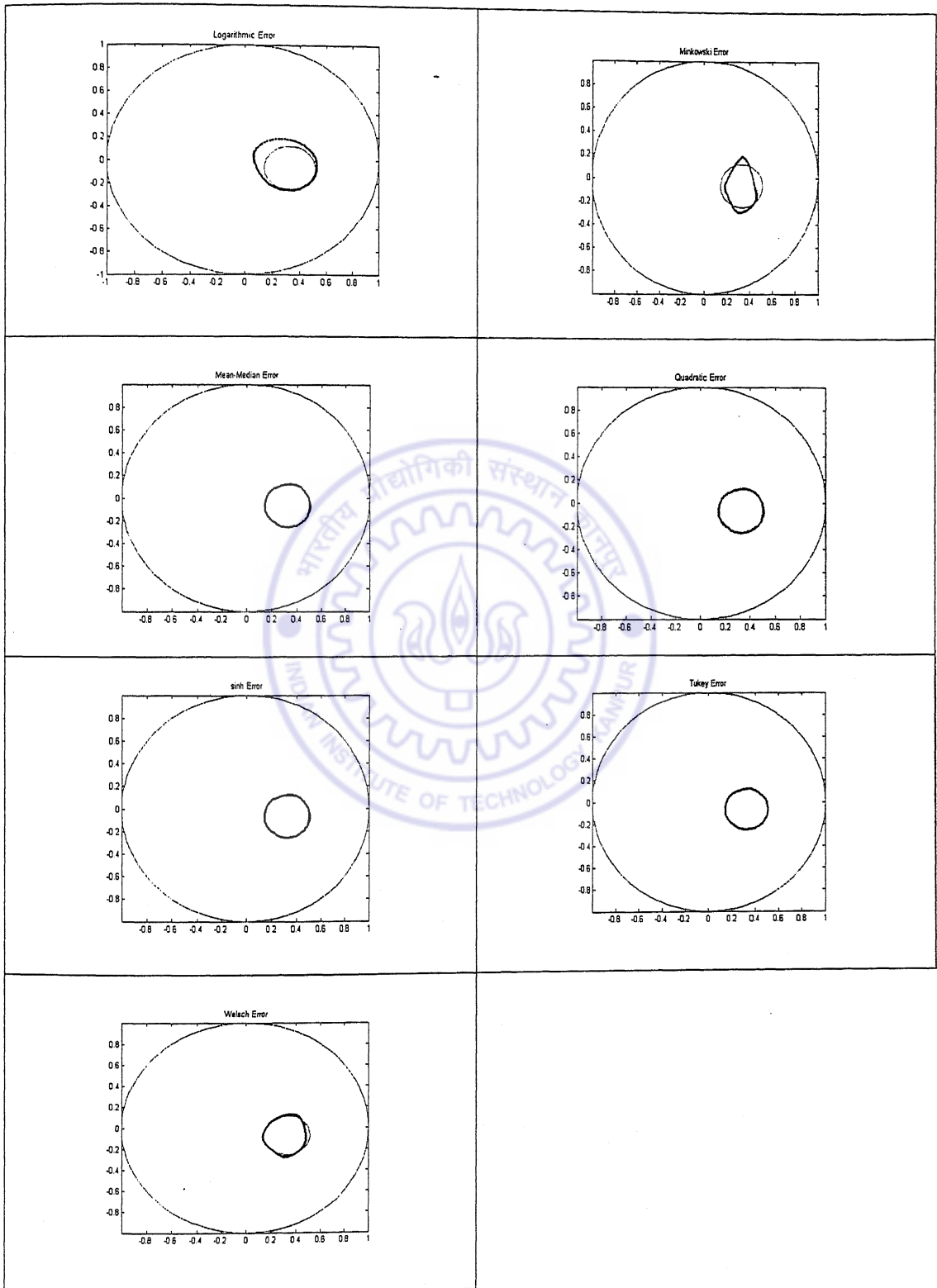


Table 5.2 1-5-1 architecture, New Activation Function. Learning rate 0.1, 1500 epochs. Epochs criterion was selected while training.







5.2.2 The Polynomial Map

The solution to the linear constant coefficient differential equations are obtained as a characteristic-value problem of the characteristic equation obtained by posing the

problem as one of matrix differential operators. Polynomials are frequently employed and the example mentioned here is but a small application of the innumerable existent. As these functions are most frequently applied, the CNN was used to address them to study the convergence. The quadratic and bi-quadratic maps were considered for the experiment. The architectures used were 1-5-1 and 1-10-1, learning rate was 0.1, and Nitta and New activation functions were used for solving the problem. The target error was set to 0.000001. The number of epochs was 2500.

$$y = x^2 \quad (5.4)$$

$$y = x^4 \quad (5.5)$$

were considered for the present study. The quadratic function is less steep than the fourth power polynomial and both are even functions.

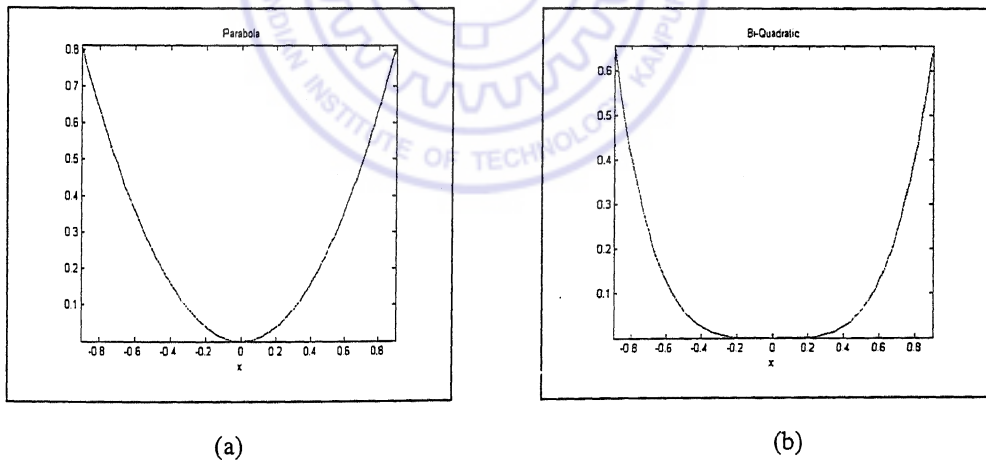
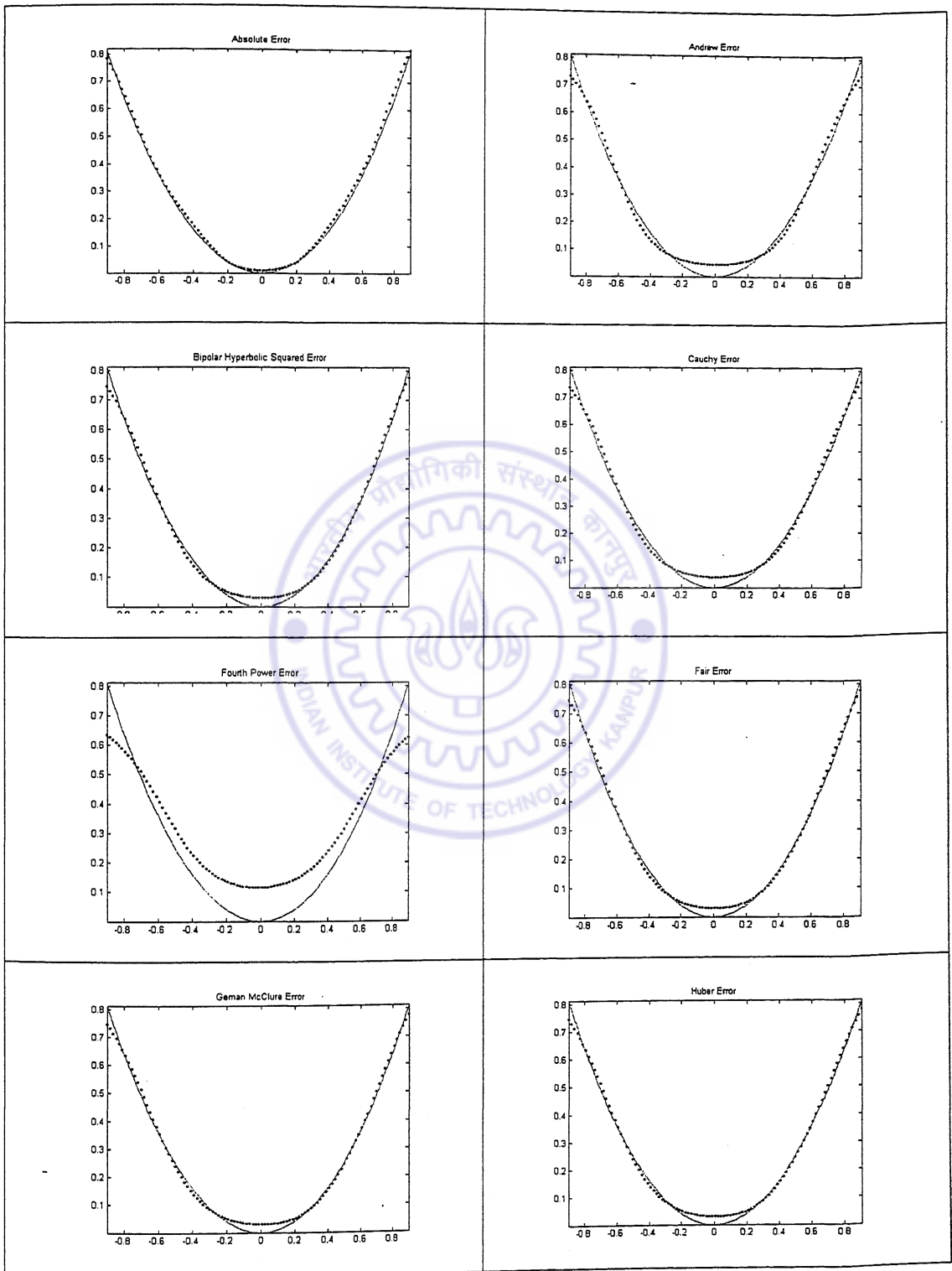
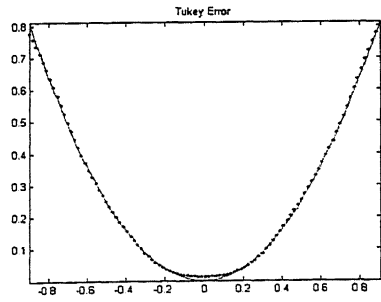
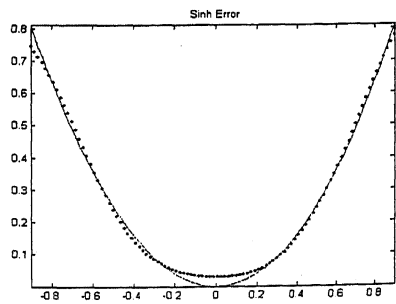
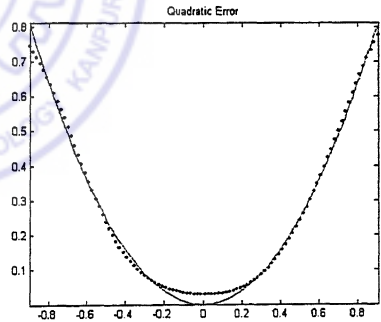
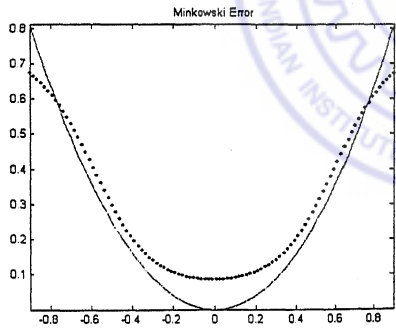
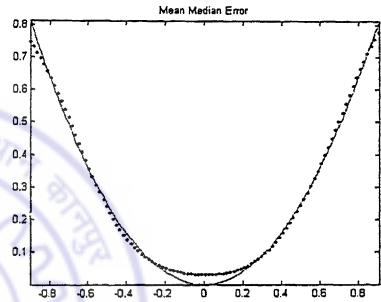
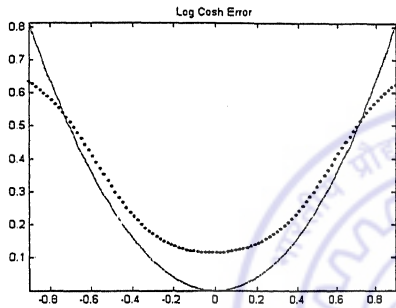
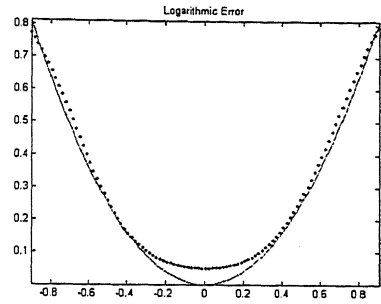
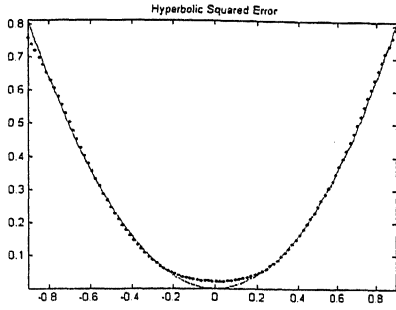
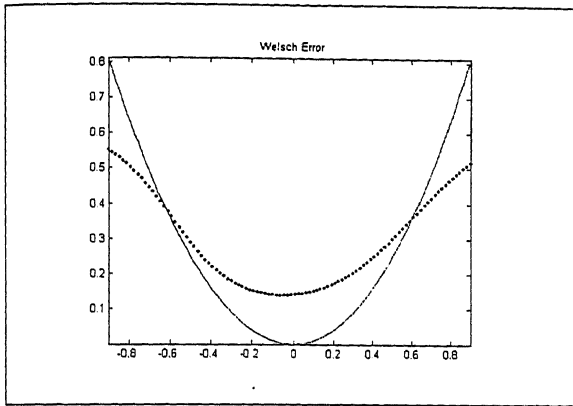


Fig. 5.2 The polynomials of quadratic and Fourth Power are shown in (a) and (b) respectively. A comparison clearly reveals that the curves are a way different with Fourth Power having a slower rise than the Quadratic. Both curves were chosen to

Table 5.3 Capturing Parabola using Nitta Activation, with a 1-5-1 architecture, learn rate of 0.1. Epochs were set to 2500. Epochs criterion was set for the training process.

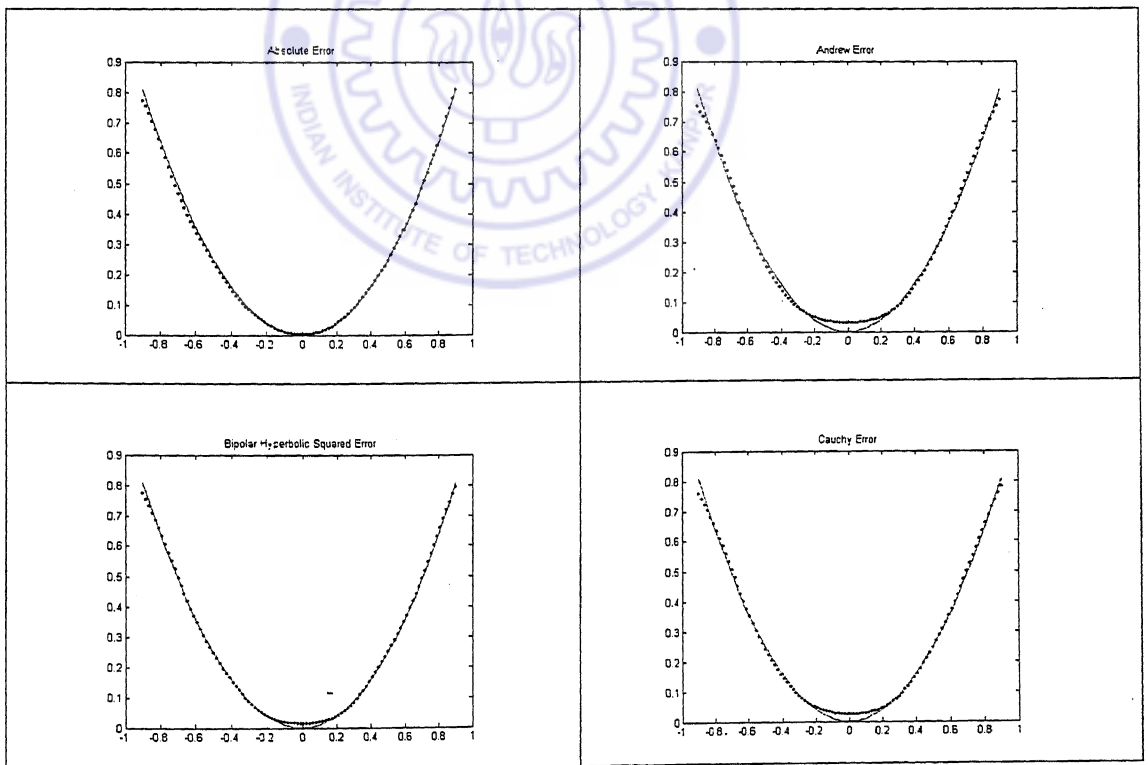


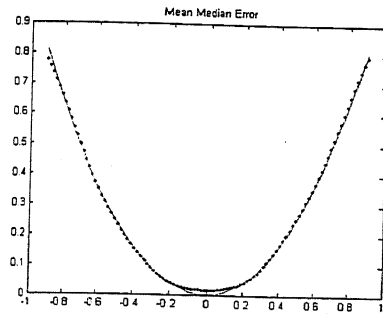
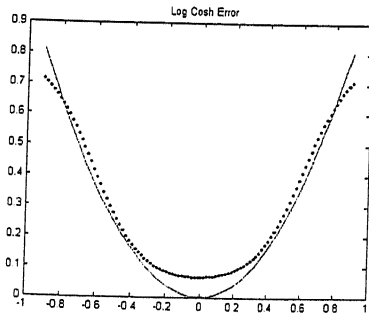
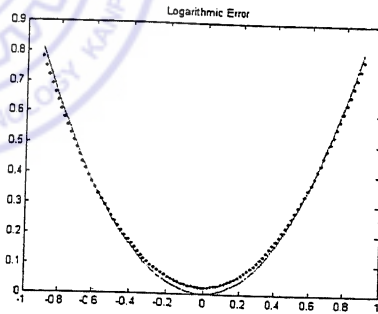
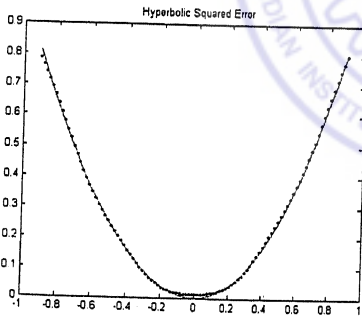
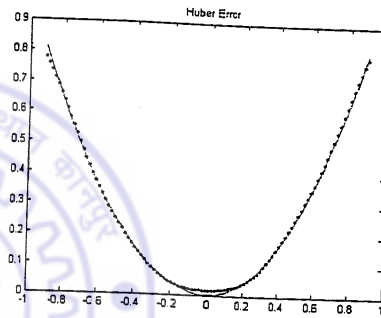
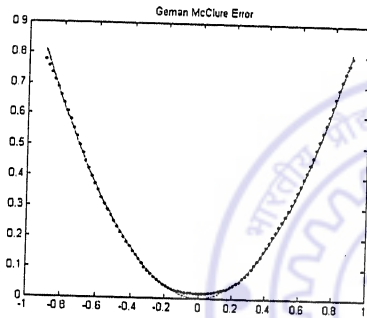
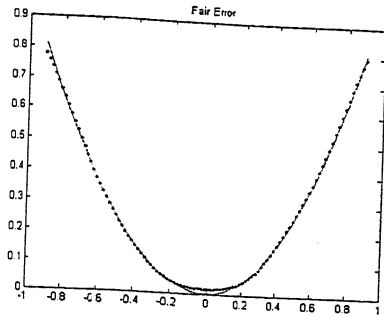
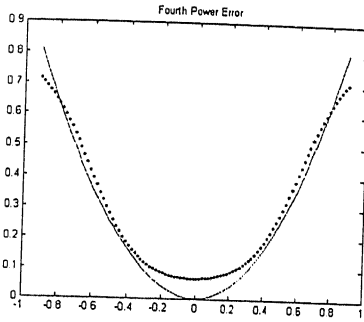




As earlier pointed out, the data sets were chosen normalized for otherwise the activation functions will not be effective.

Table 5.4 Capturing Parabola using New Activation, with a 1-5-1 architecture, learn rate of 0.1, 2500 epochs. Epochs criterion was set for the training process.





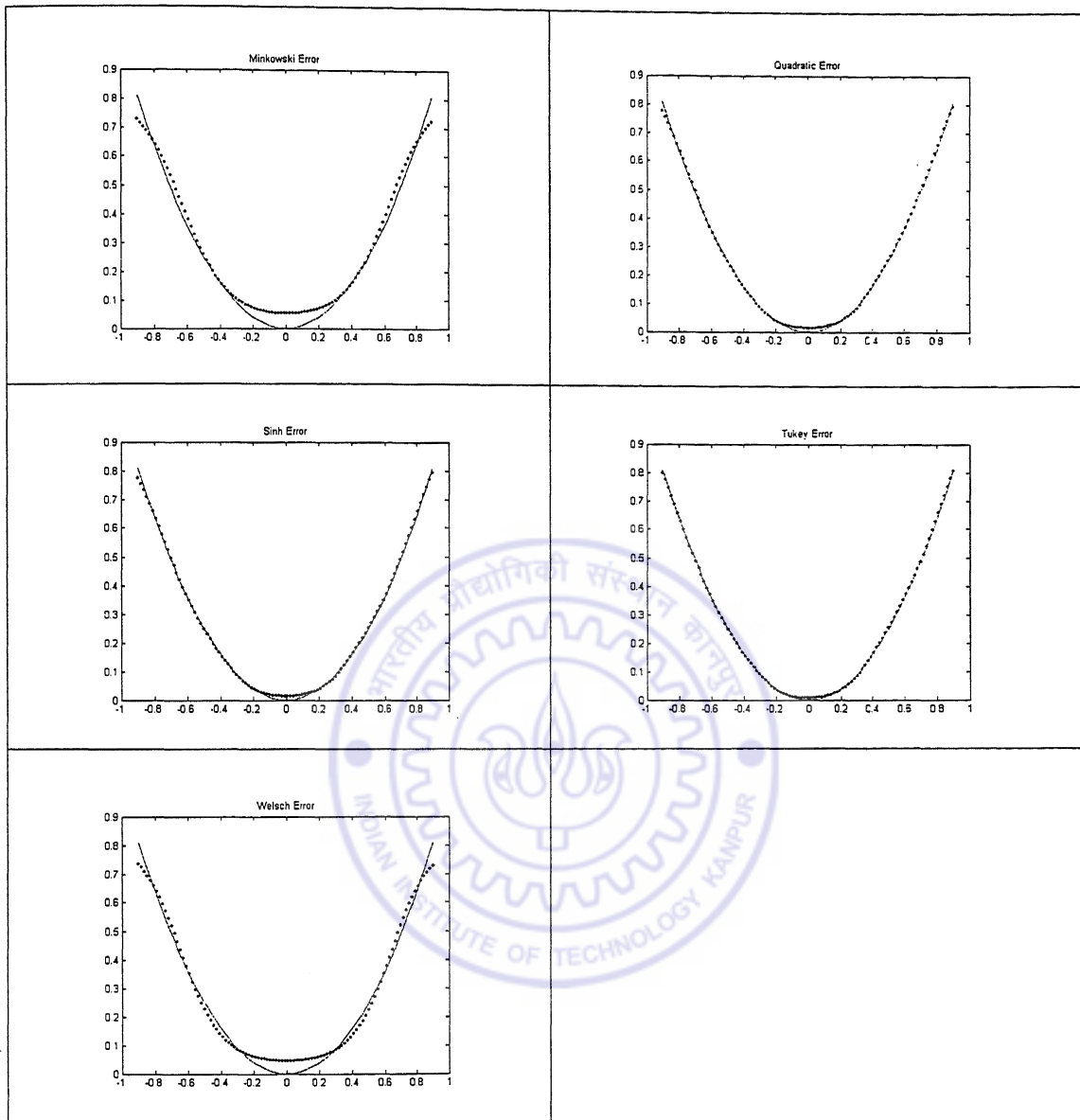
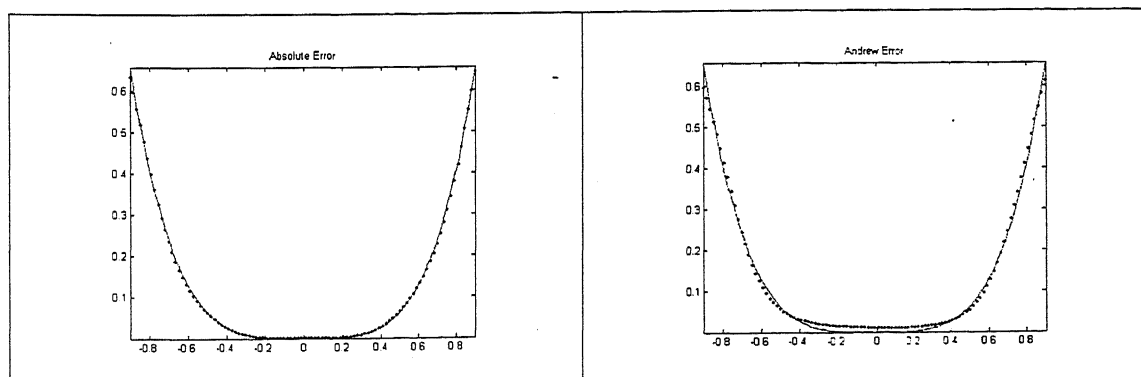
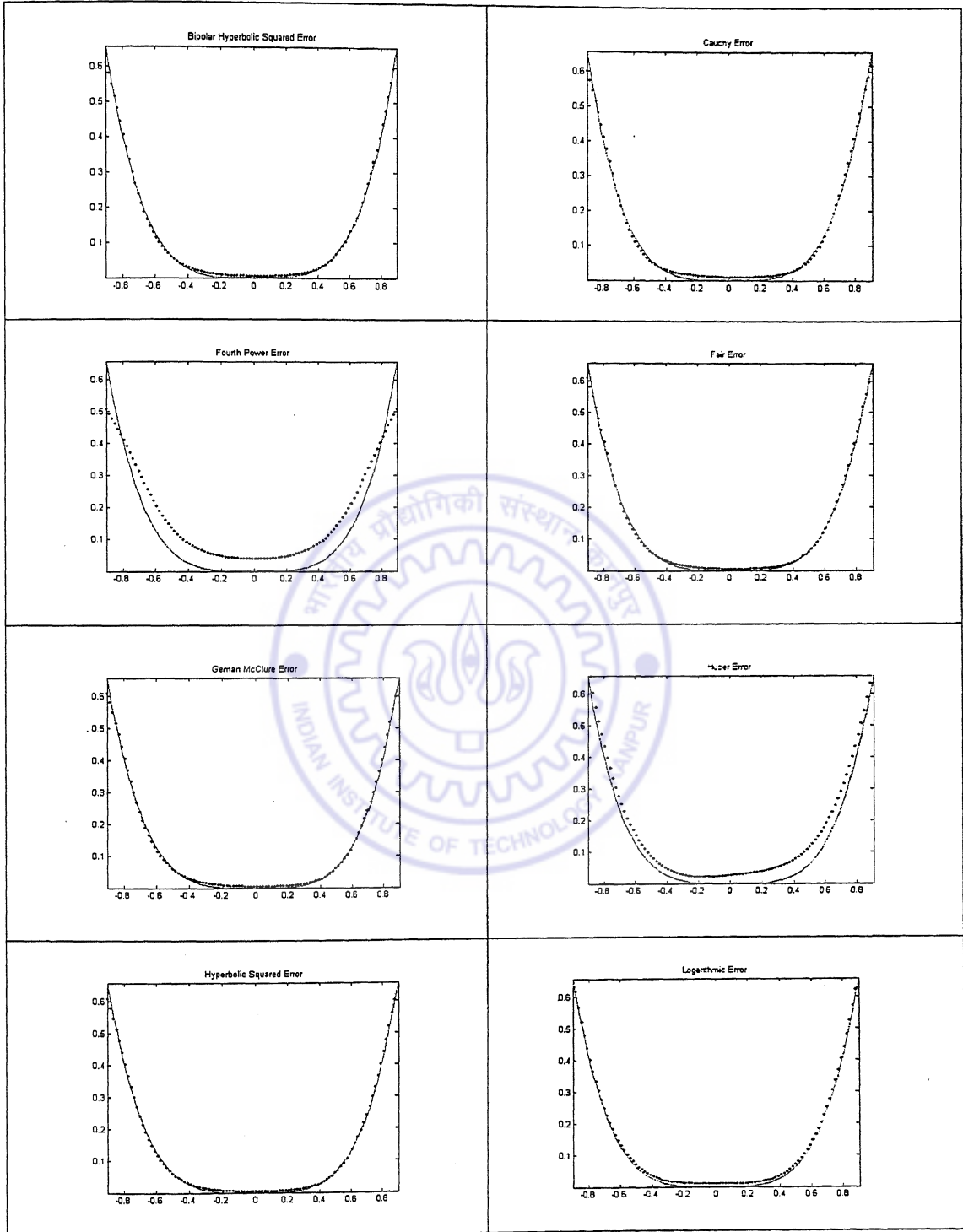


Table 5.5 Capturing Fourth Power using Nitta Activation, with a 1-5-1 architecture, learn rate of 0.1, 2500 epochs. Epochs criterion was set for the training process.





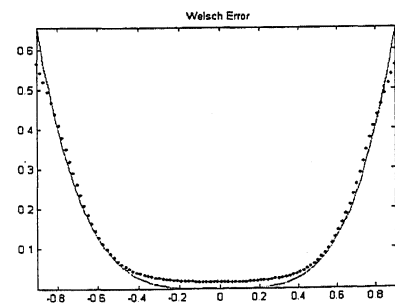
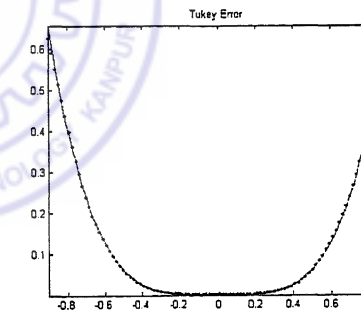
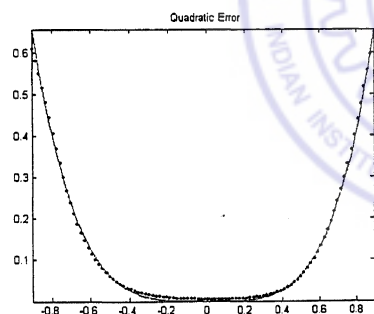
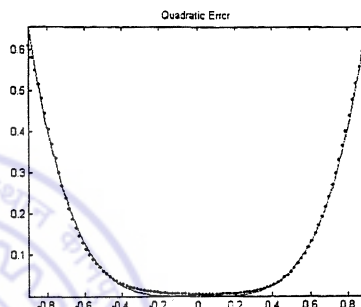
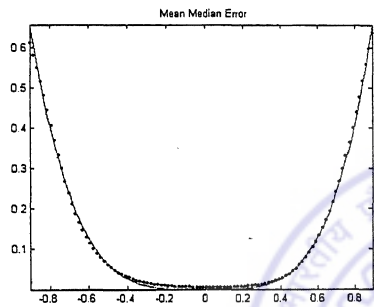
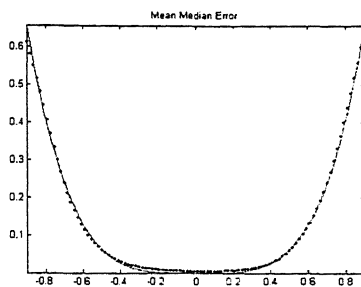
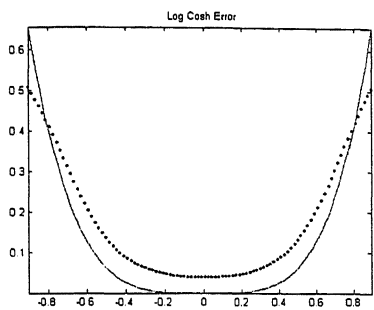
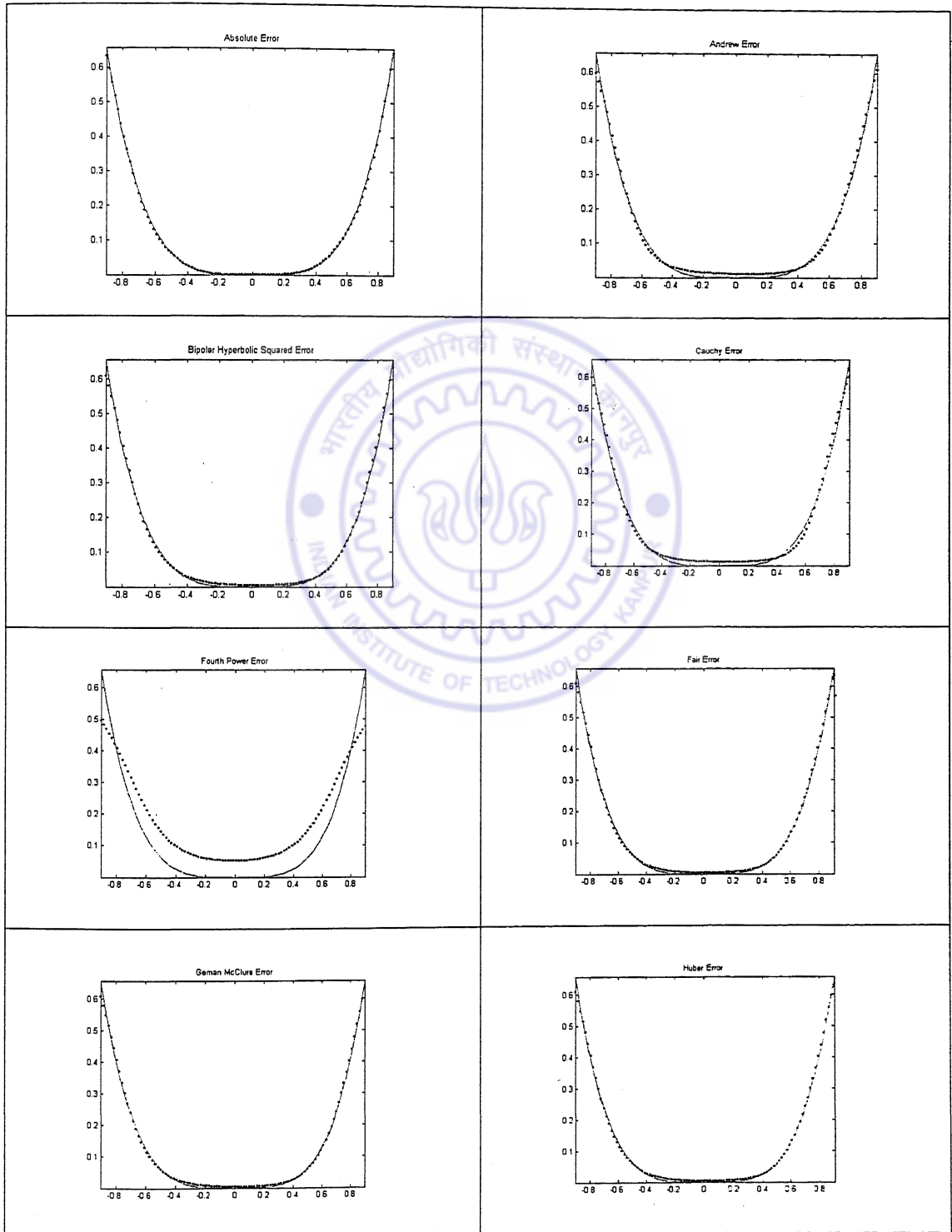
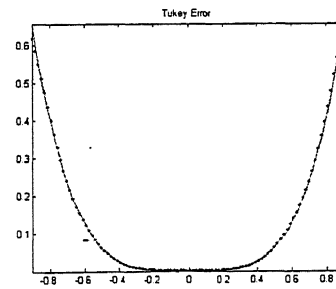
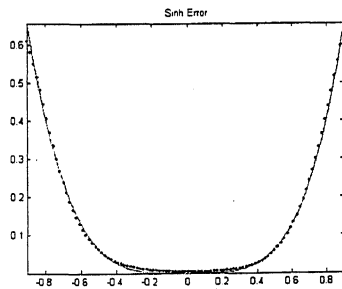
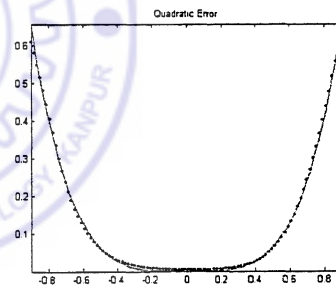
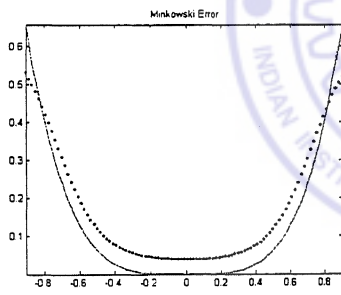
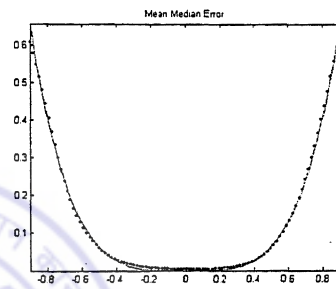
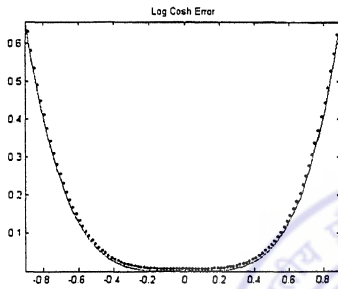
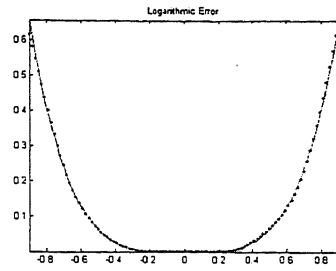
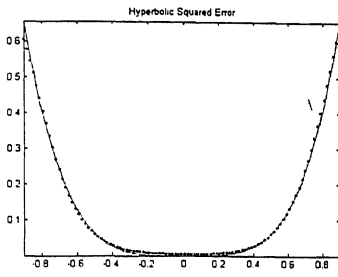


Table 5.6 Capturing Fourth Power using Nitta Activation, with a 1-10-1 architecture, learn rate of 0.1, 2500 epochs. Epochs criterion was set for the training process.





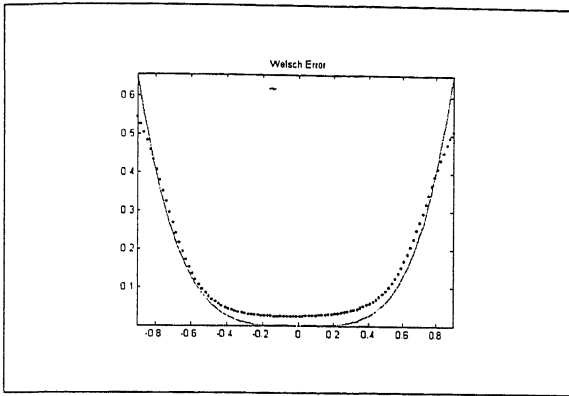
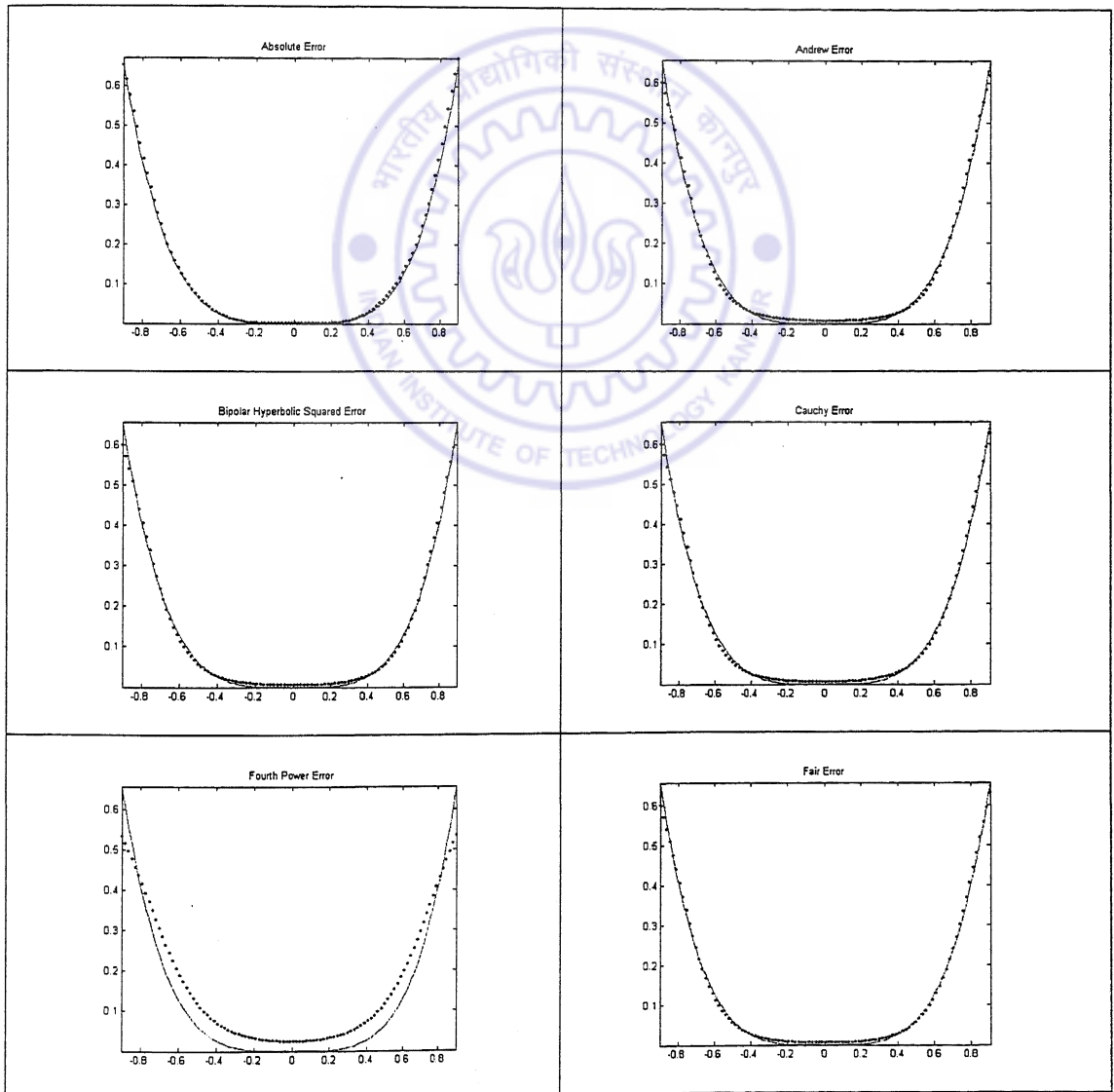
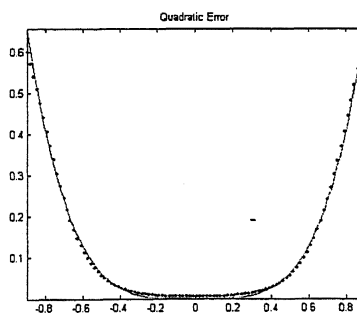
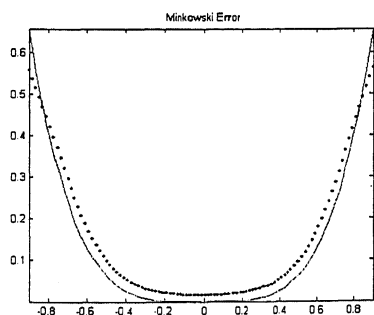
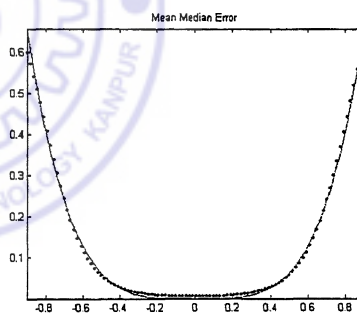
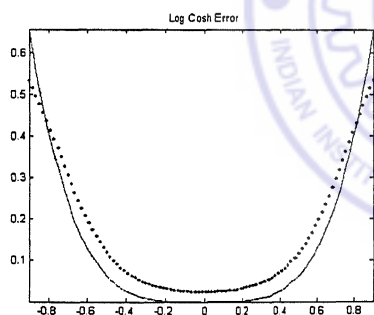
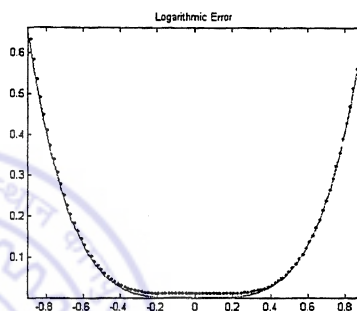
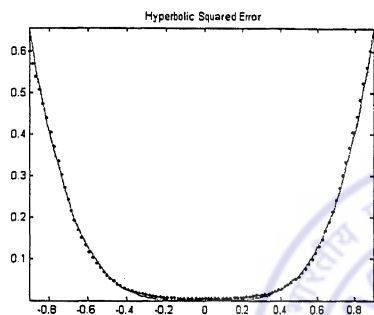
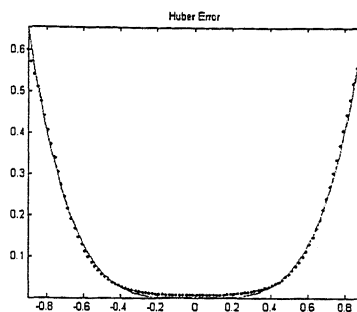
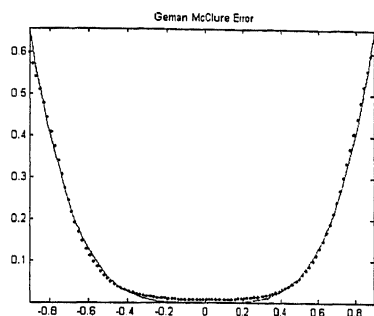


Table 5.7 Capturing Fourth Power using New Activation, with a 1-5-1 architecture, learn rate of 0.1, 2500 epochs. Epochs criterion was set for the training process.





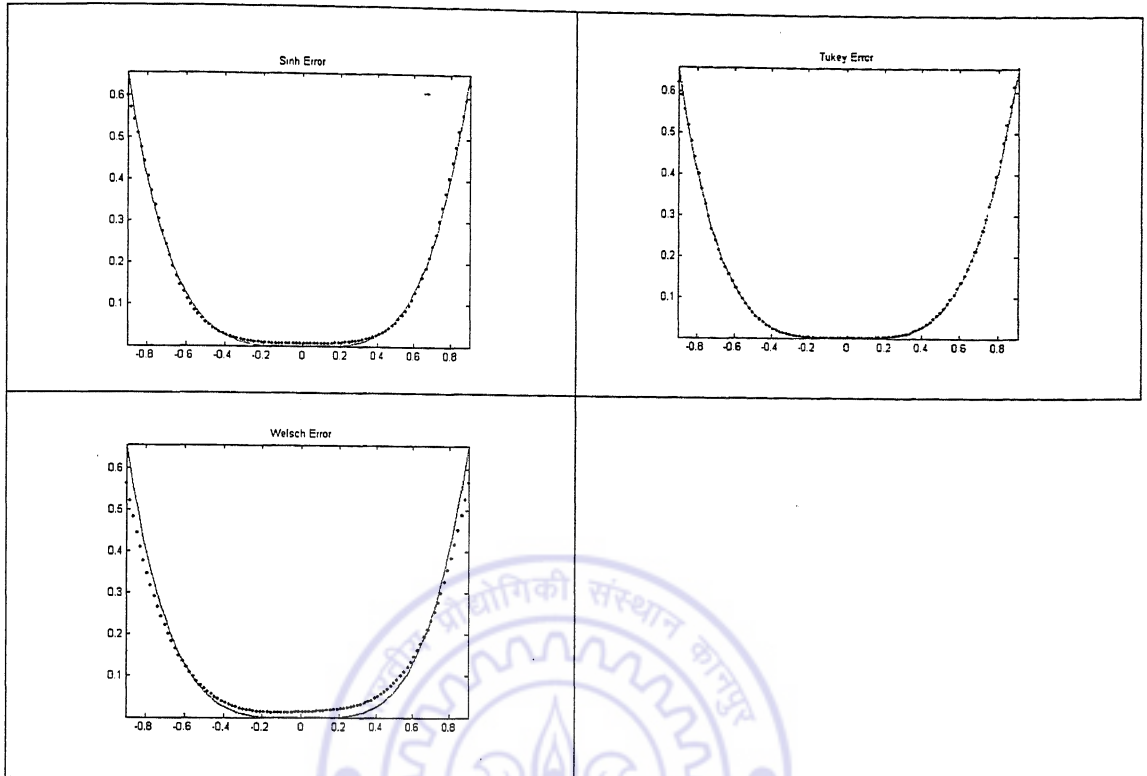
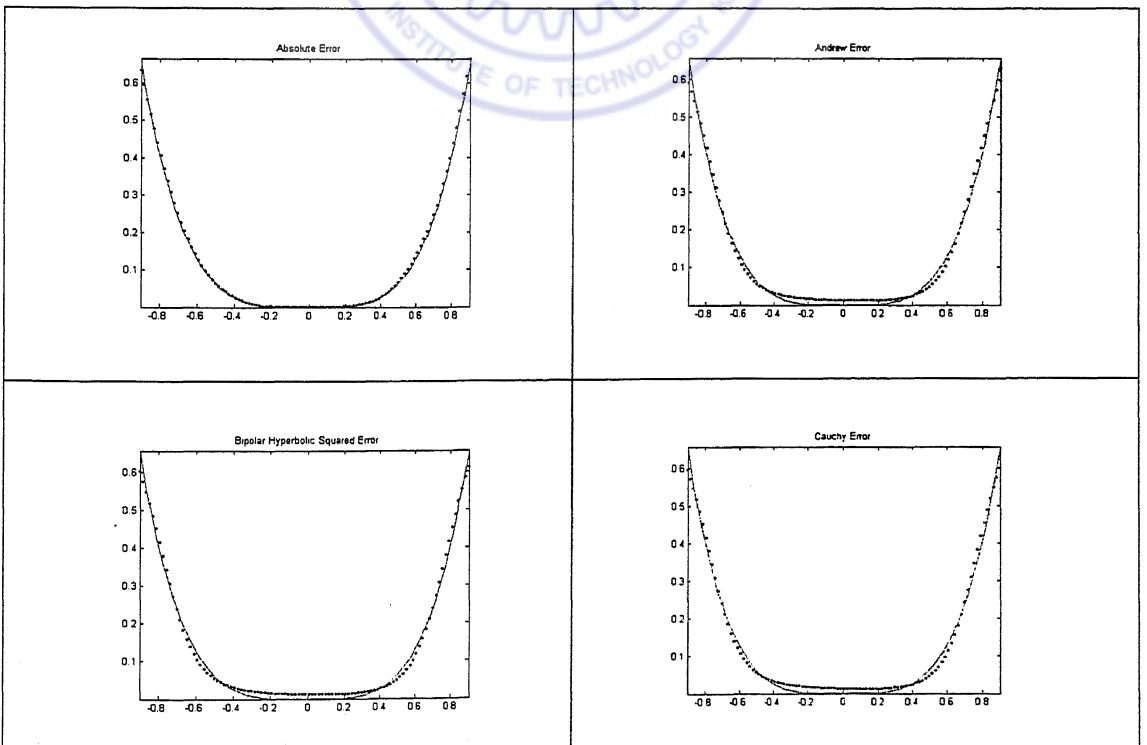
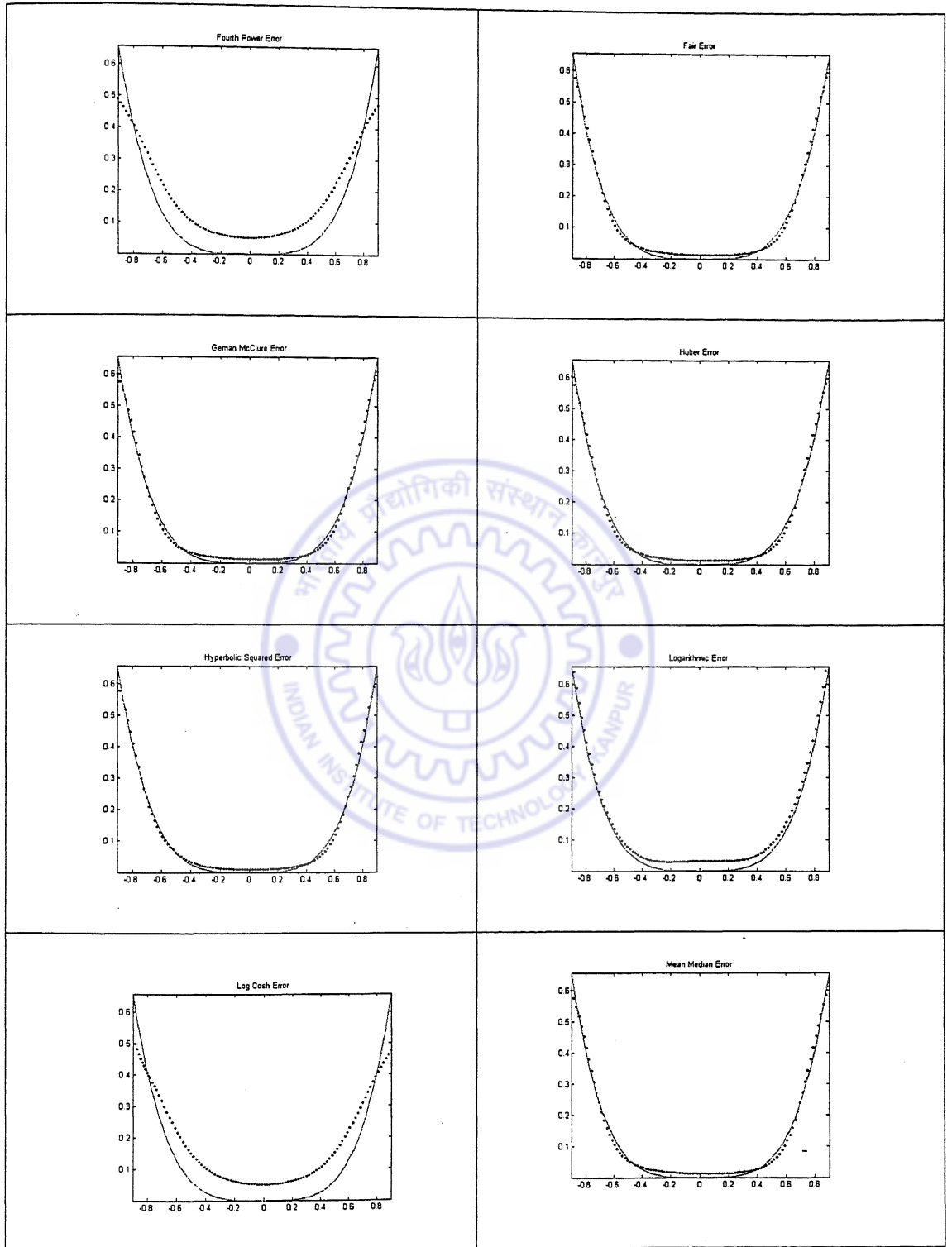
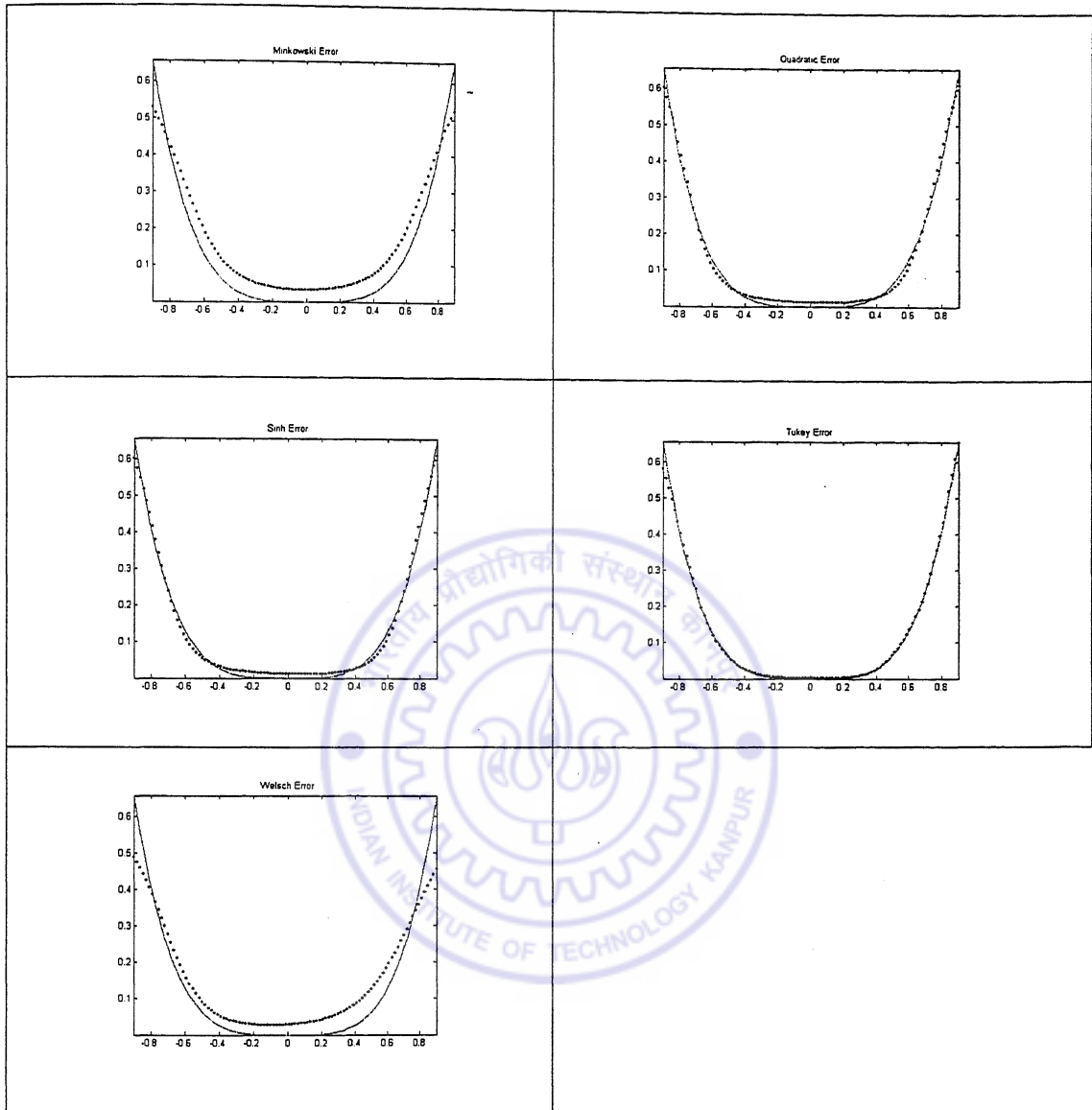


Table 5.8 Capturing Fourth Power using New Activation, with a 1-10-1 architecture, learn rate of 0.1, 2500 epochs. Epochs criterion was set for the training process.







5.2.3 The Similarity Transformation

The Similarity Transformation was studied in Nitta (1997). In the present context, the mapping is studied as a particular case of the Bilinear Transformation. Fig 5.1 describes the problem. The outer circle is of unit radius while the inner one is half the radius of the outer circle. The problem requires a Neural Network to map the points on the outer circle to the ones on the inner one radially. It can be seen that the transformation is a particular case of the Bilinear Transformation as earlier pointed out. The parameters for this experiment were: architecture was 1-5-1, learning rate was 0.1, epochs were set to 1500. Nitta and New activation functions have been used with the problem. The target error was set to 0.000001.

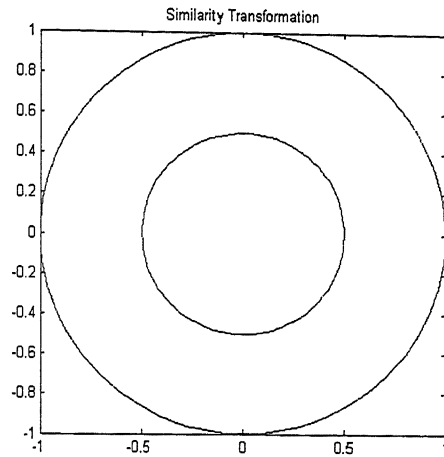
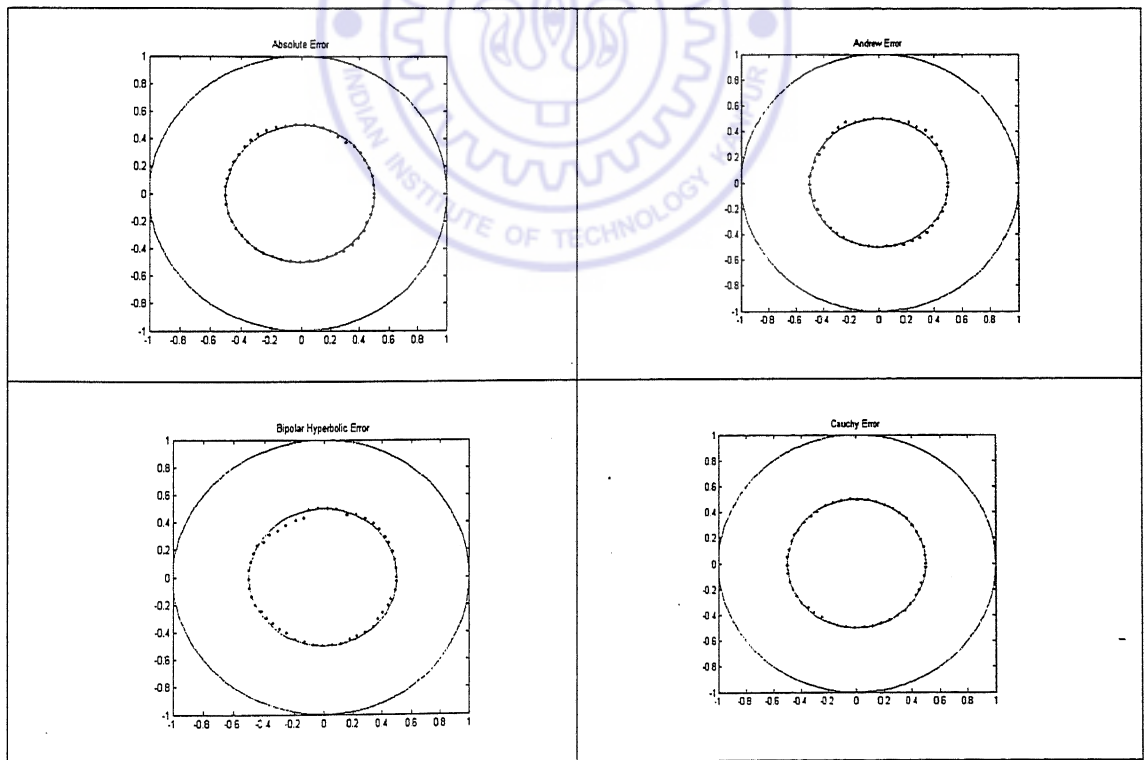
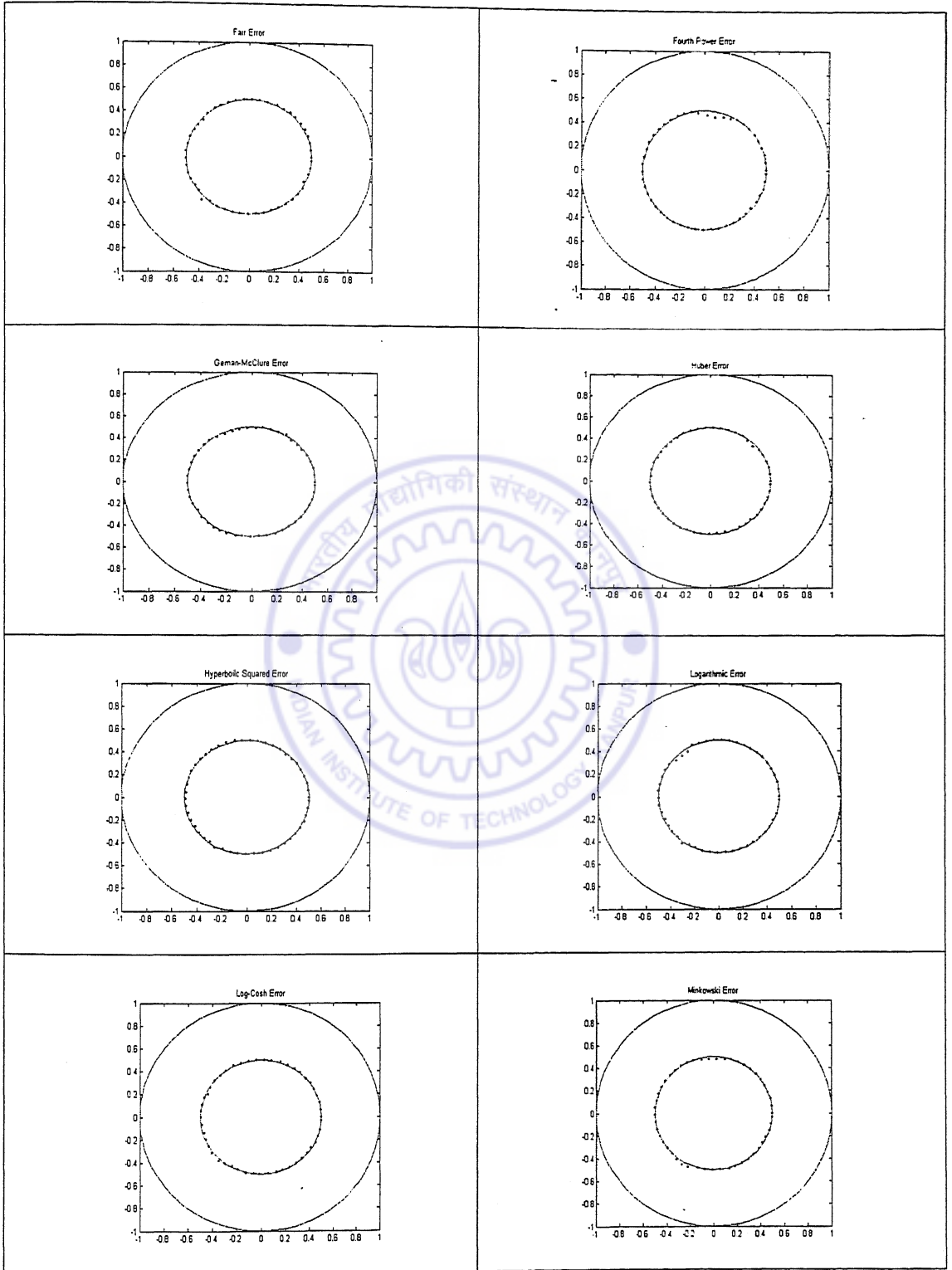


Fig. 5.3 The Similarity Transformation. The outer circle is mapped radially onto the inner one which is concentric with the outer and half its radius

Table 5.9 Capturing Similarity Transformation using NIIta Activation, with a 1-5-1 architecture, learn rate of 0.1, 1500 epochs. Epochs criterion was set for the training process.





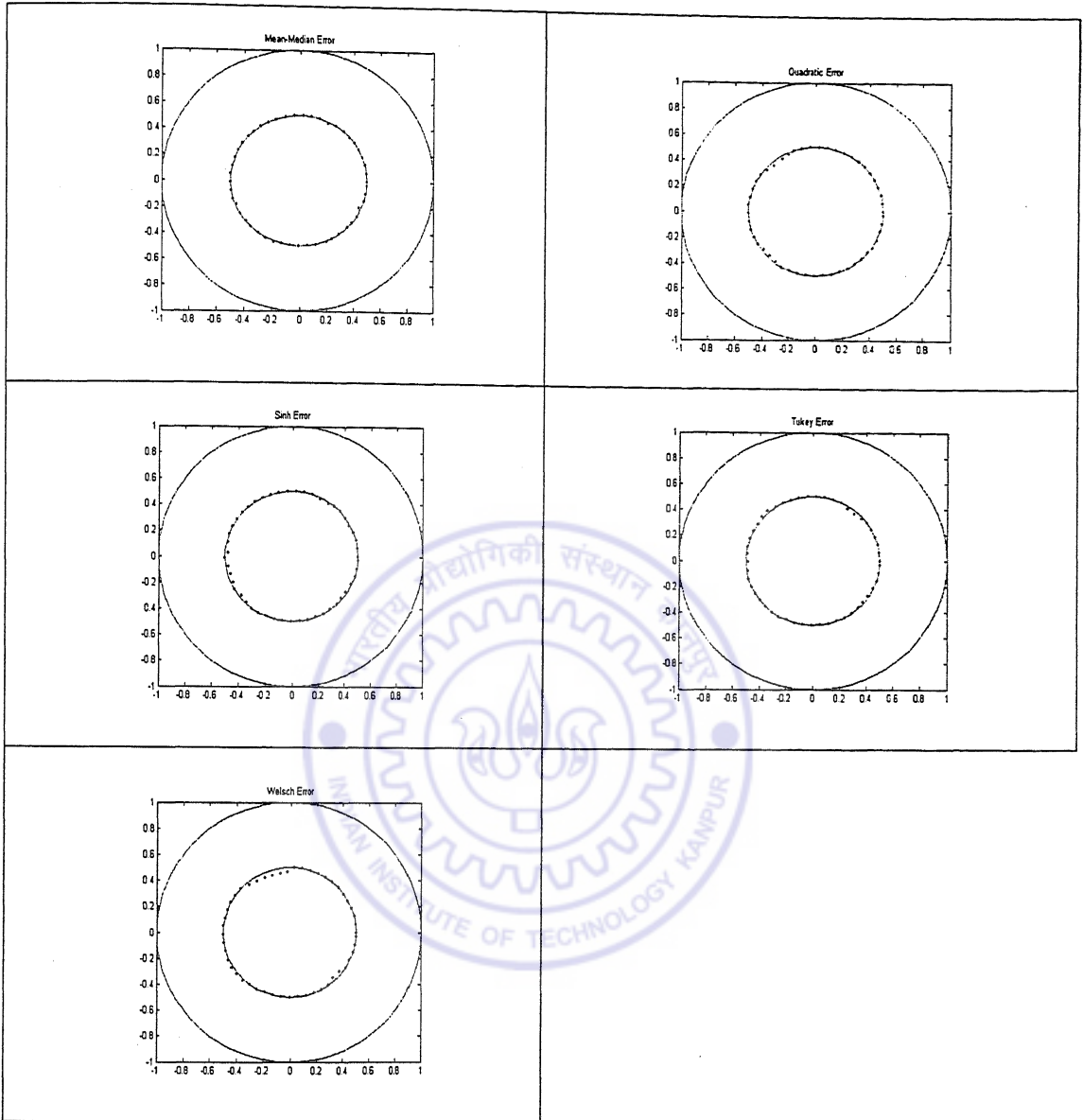
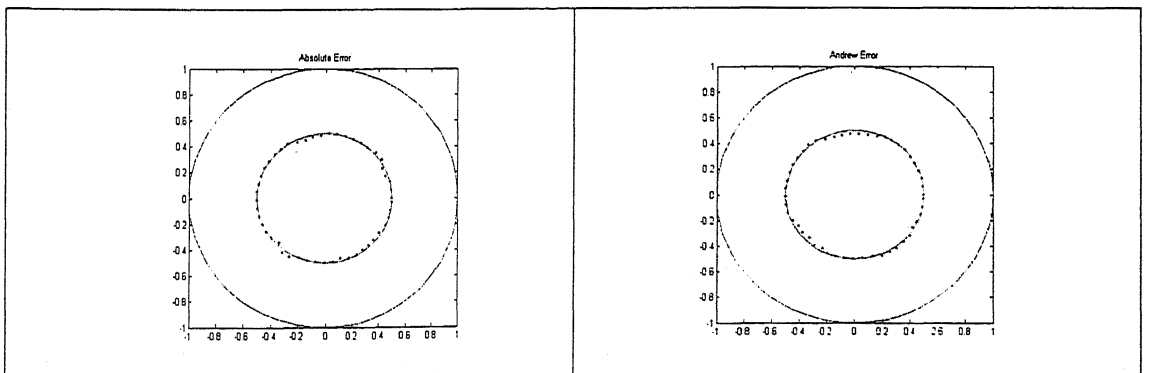
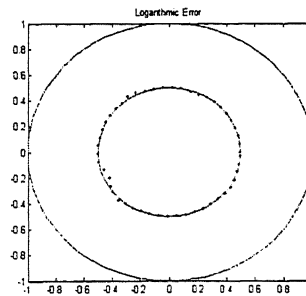
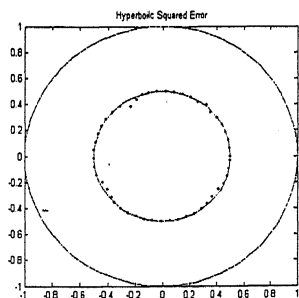
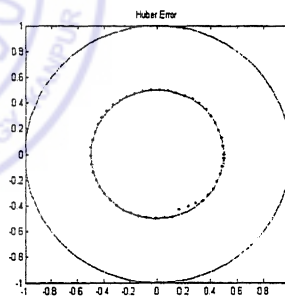
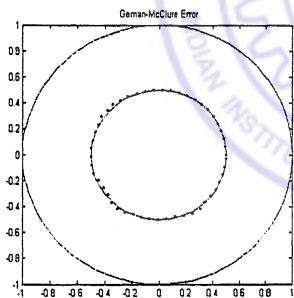
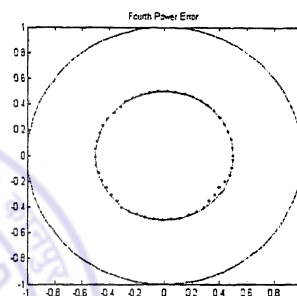
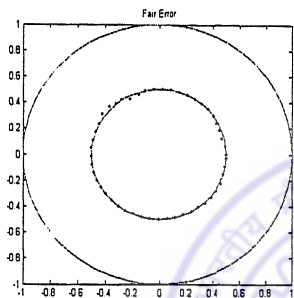
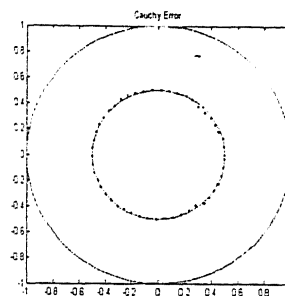
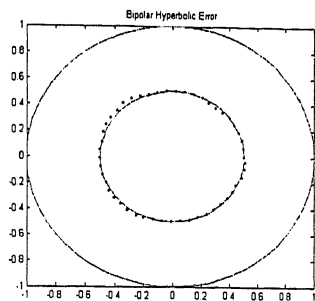
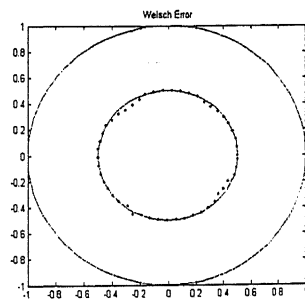
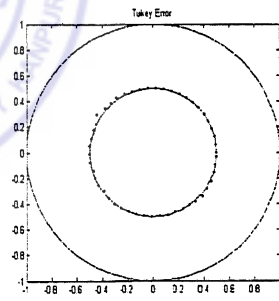
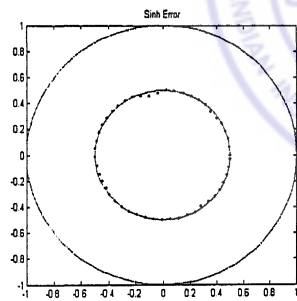
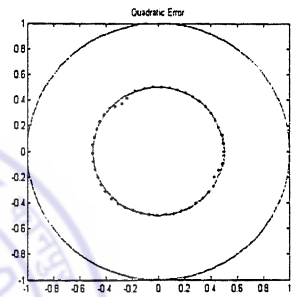
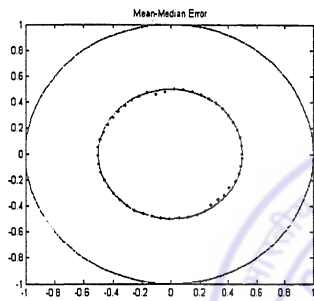
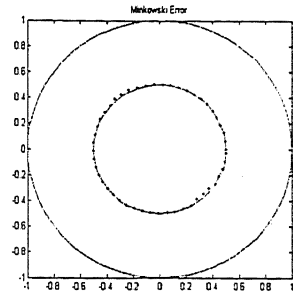
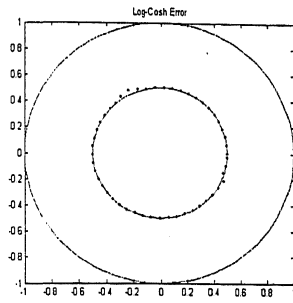


Table 5.10 Capturing Similarity Transformation using New Activation, with a 1-5-1 architecture, learn rate of 0.1, 1500 epochs. Epochs criterion was set for the training process.







5.2.4 Complex Exponential Map

The mapping defined by the equation

$$w = e^z \quad (5.6)$$

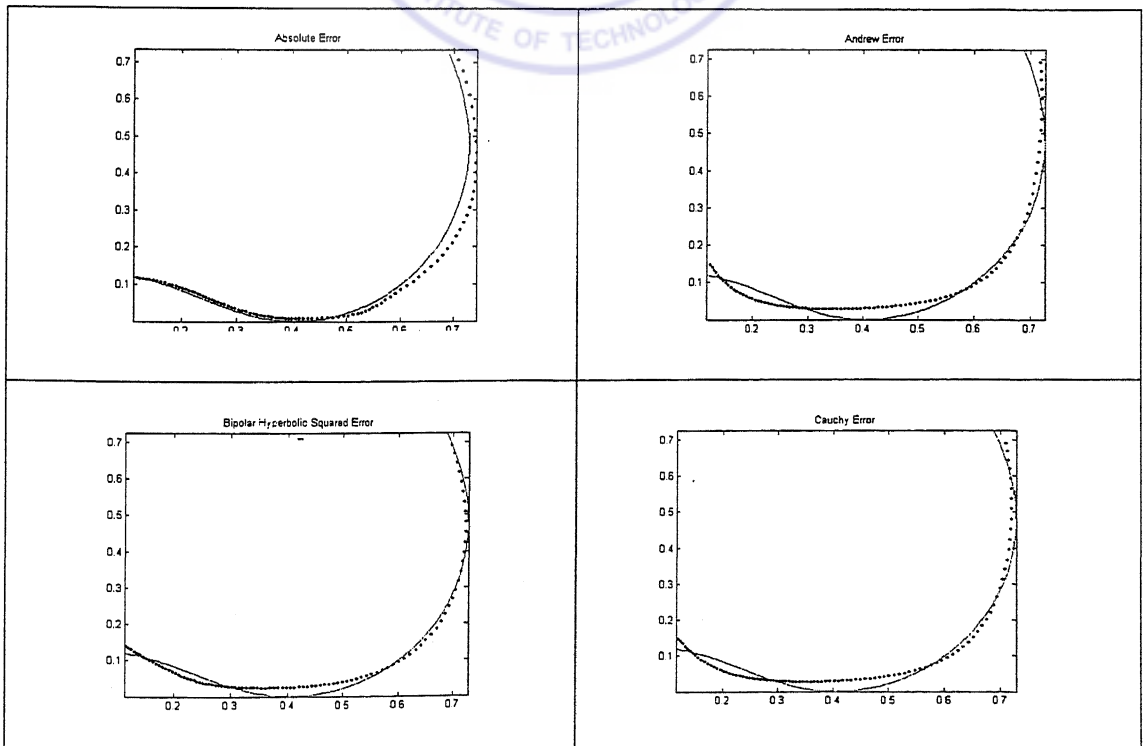
maps points on the z -plane to points on the w -plane. As the map takes points on one plane to points on the other, the best way to study the transformation is by considering a curve on the z -plane and study the image of the map on the w -plane. The complex variable z was chosen to lie on a unit circle given by the formula $e^{i\theta}$ the transformation affected by the formula is given by the following equations

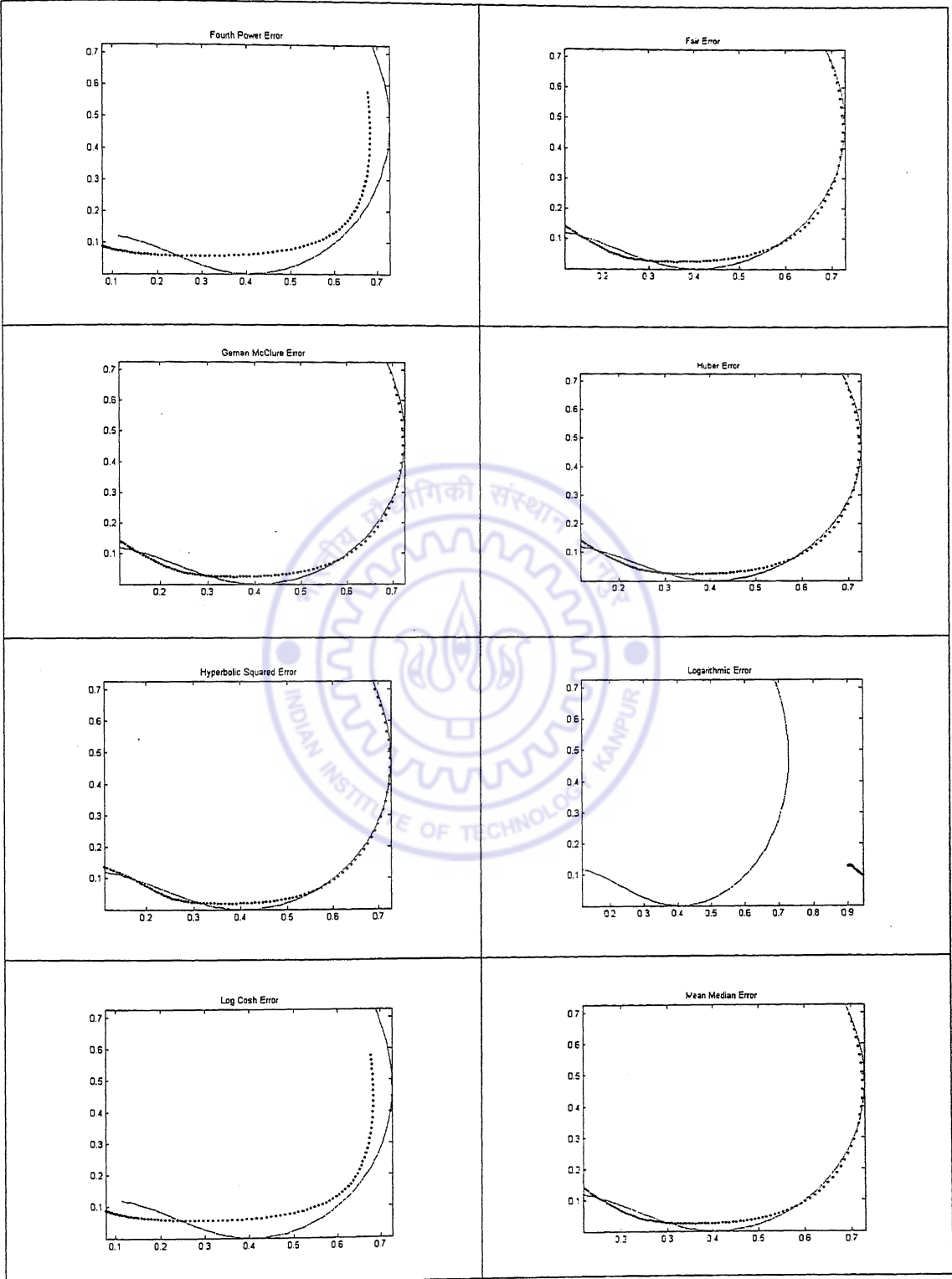
$$\operatorname{Re}(w) = e^{\cos(\theta)} \cos(\sin(\theta)) \quad (5.7a)$$

$$\operatorname{Im}(w) = e^{\cos(\theta)} \sin(\sin(\theta)) \quad (5.7b)$$

where θ is the amplitude of the complex number z . The solid line in the plots shown in Table 5.11 is the image of the transformation.

Table 5.11 $w=\exp(z)$ with 1-5-1 architecture, Nitta Activation, 4000 epochs, Learning rate = 0.1. Epochs criterion was set for the training process. Epochs criterion was set for the training process.





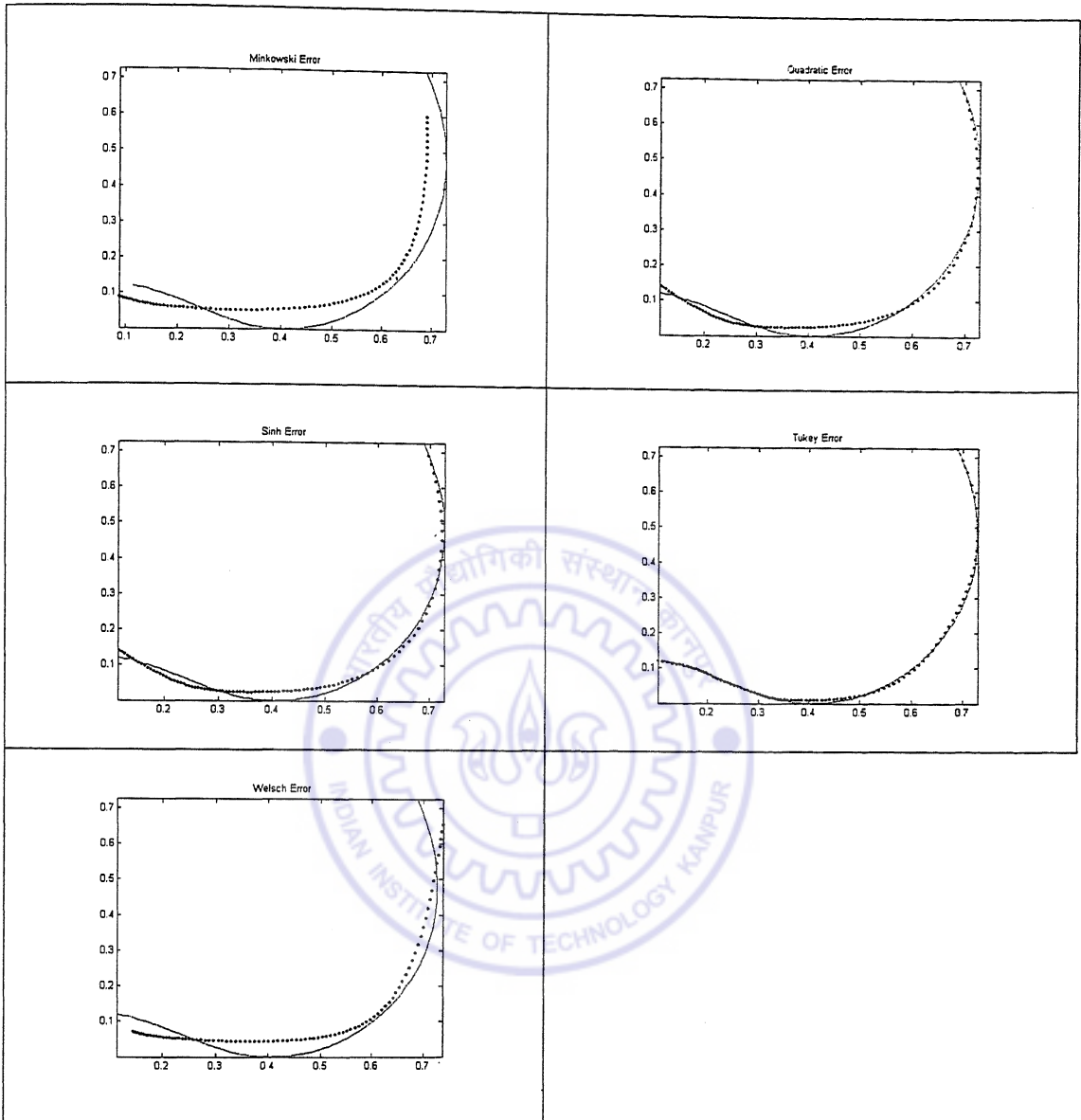
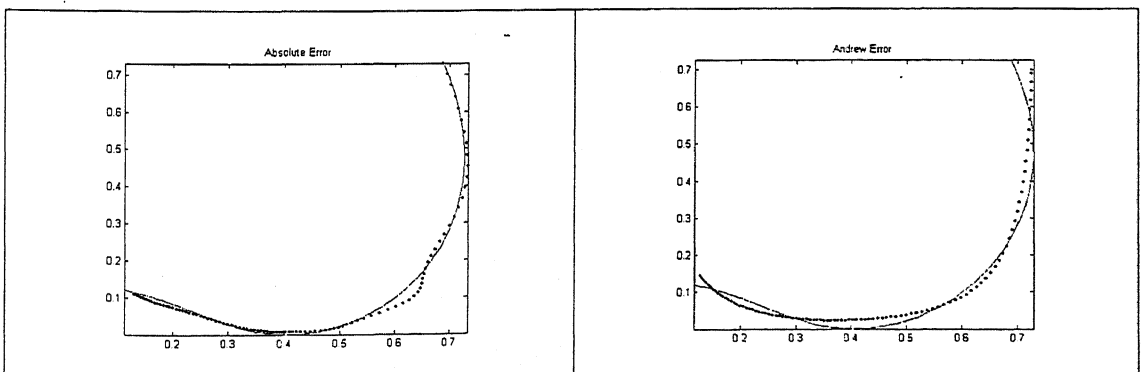
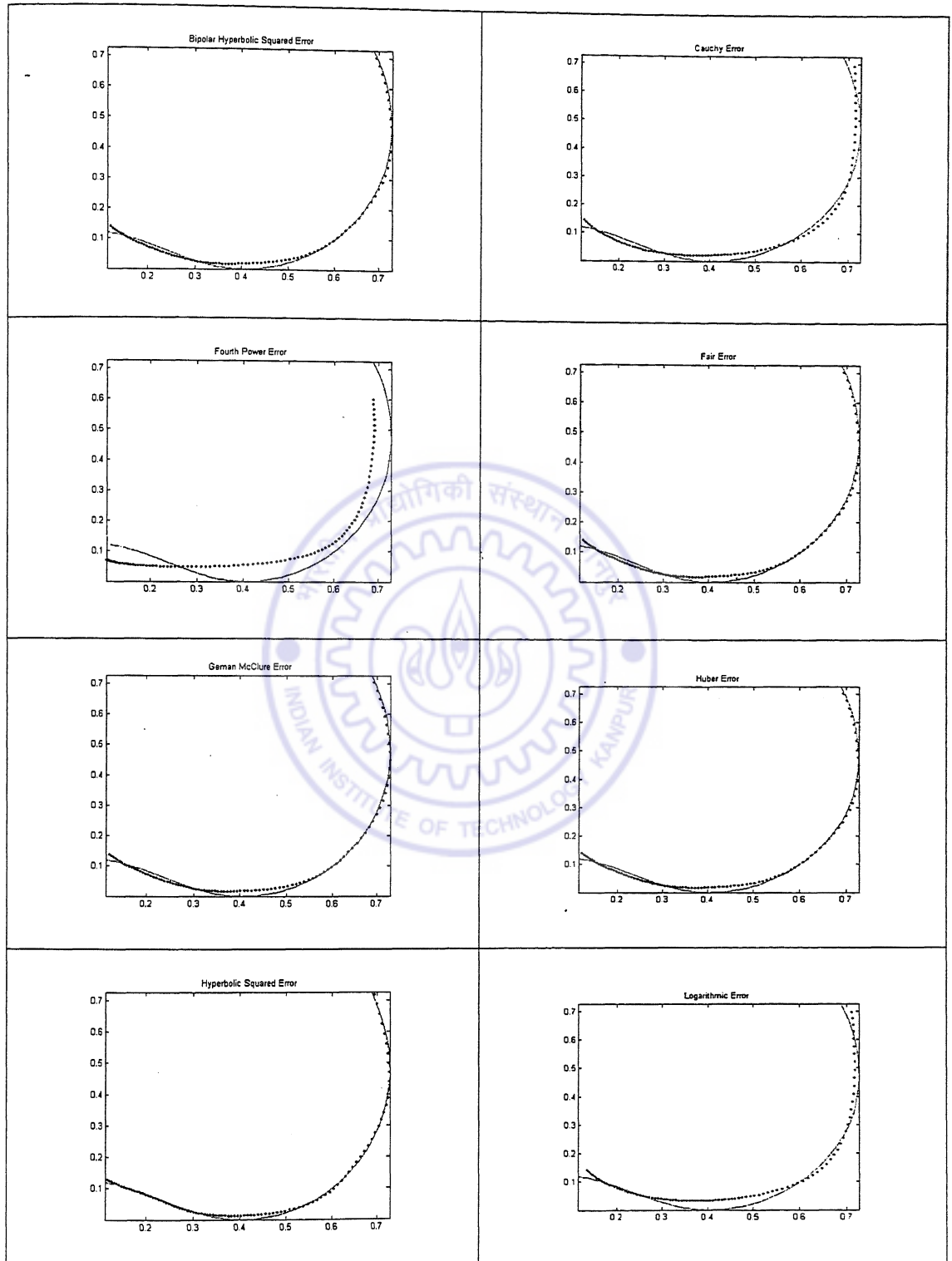


Table 5.12 $w=\exp(z)$ with 1-10-1 architecture, Nitta Activation, 4000 epochs, Learning rate = 0.1, Epochs criterion was set for the training process.





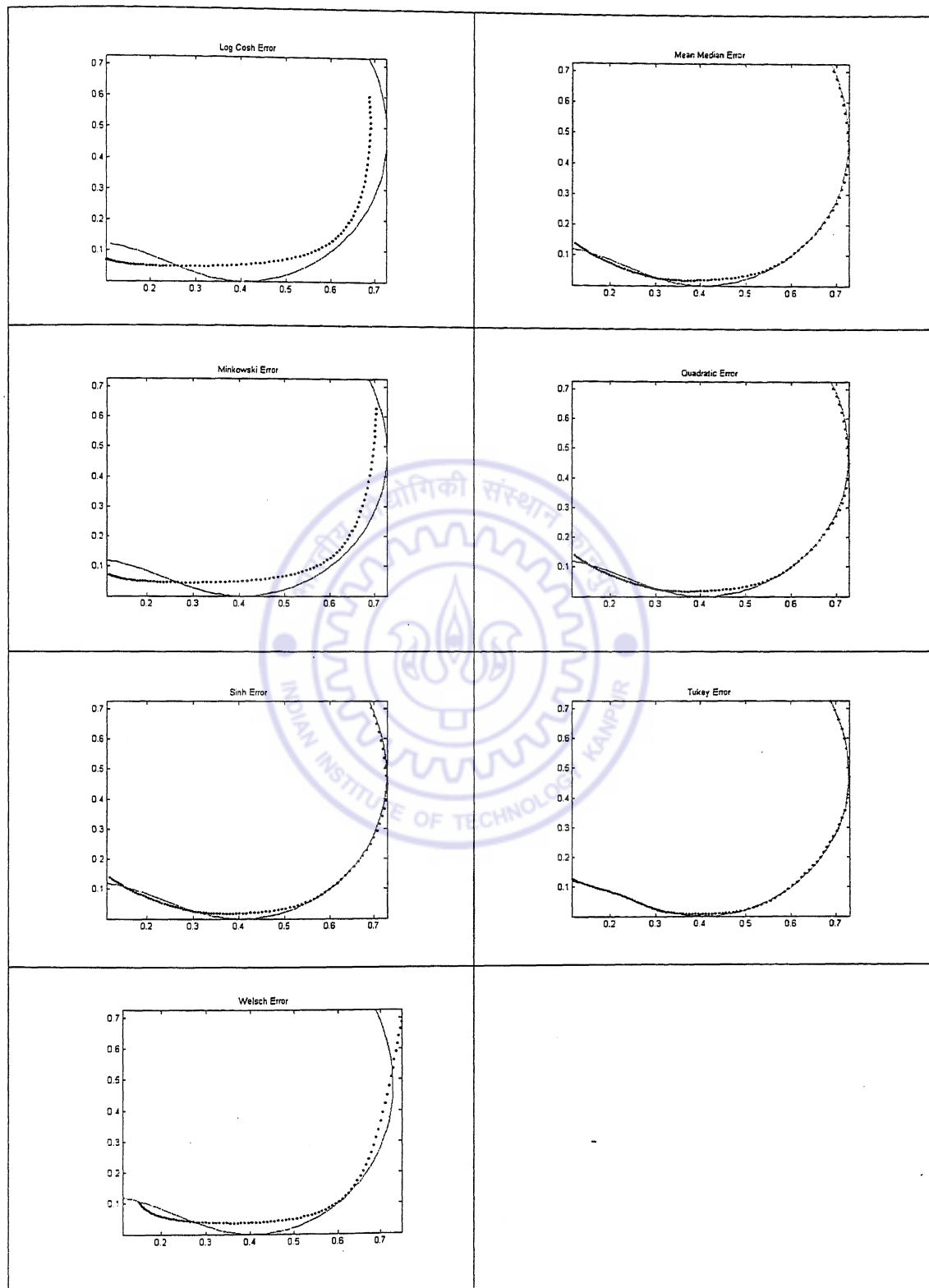
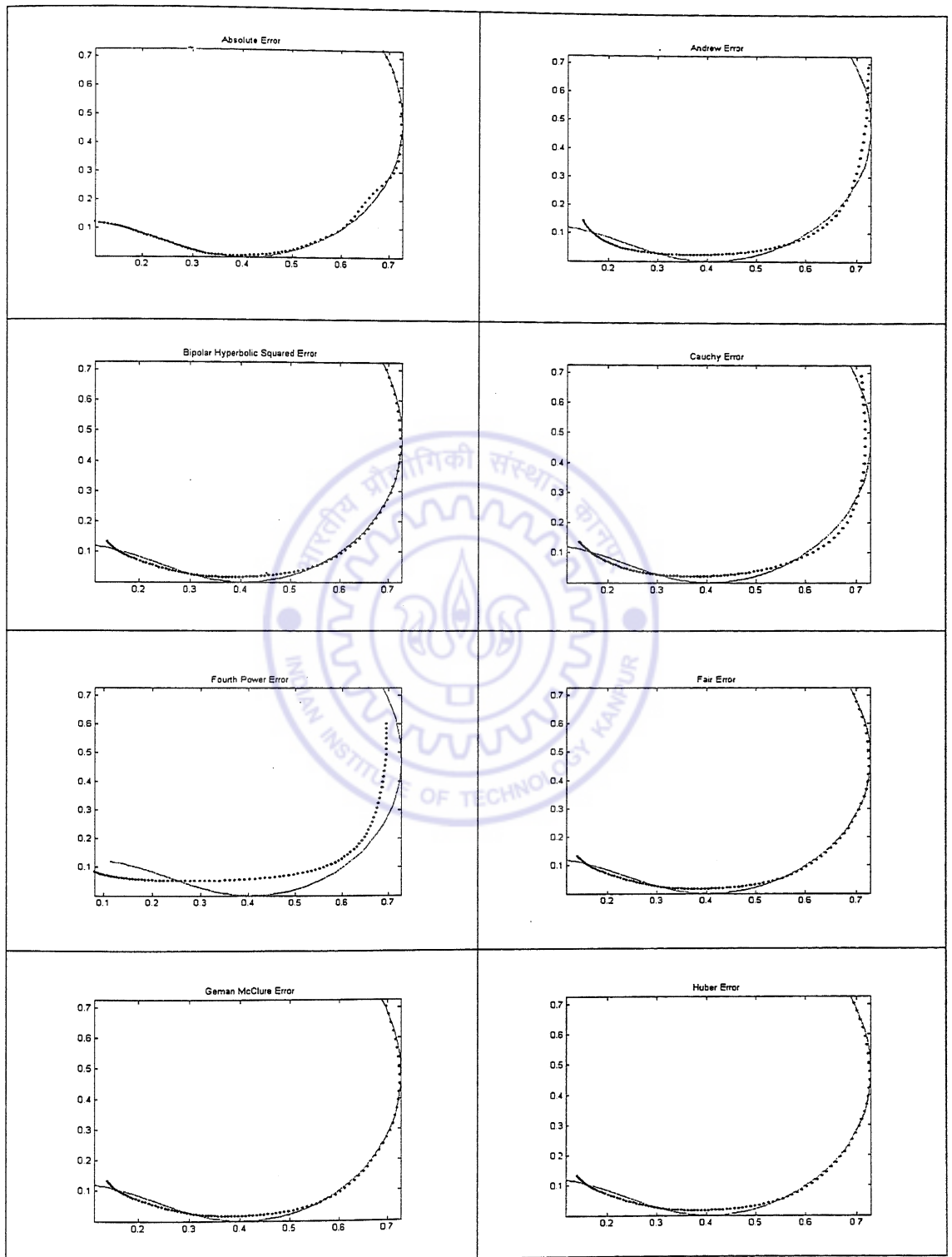
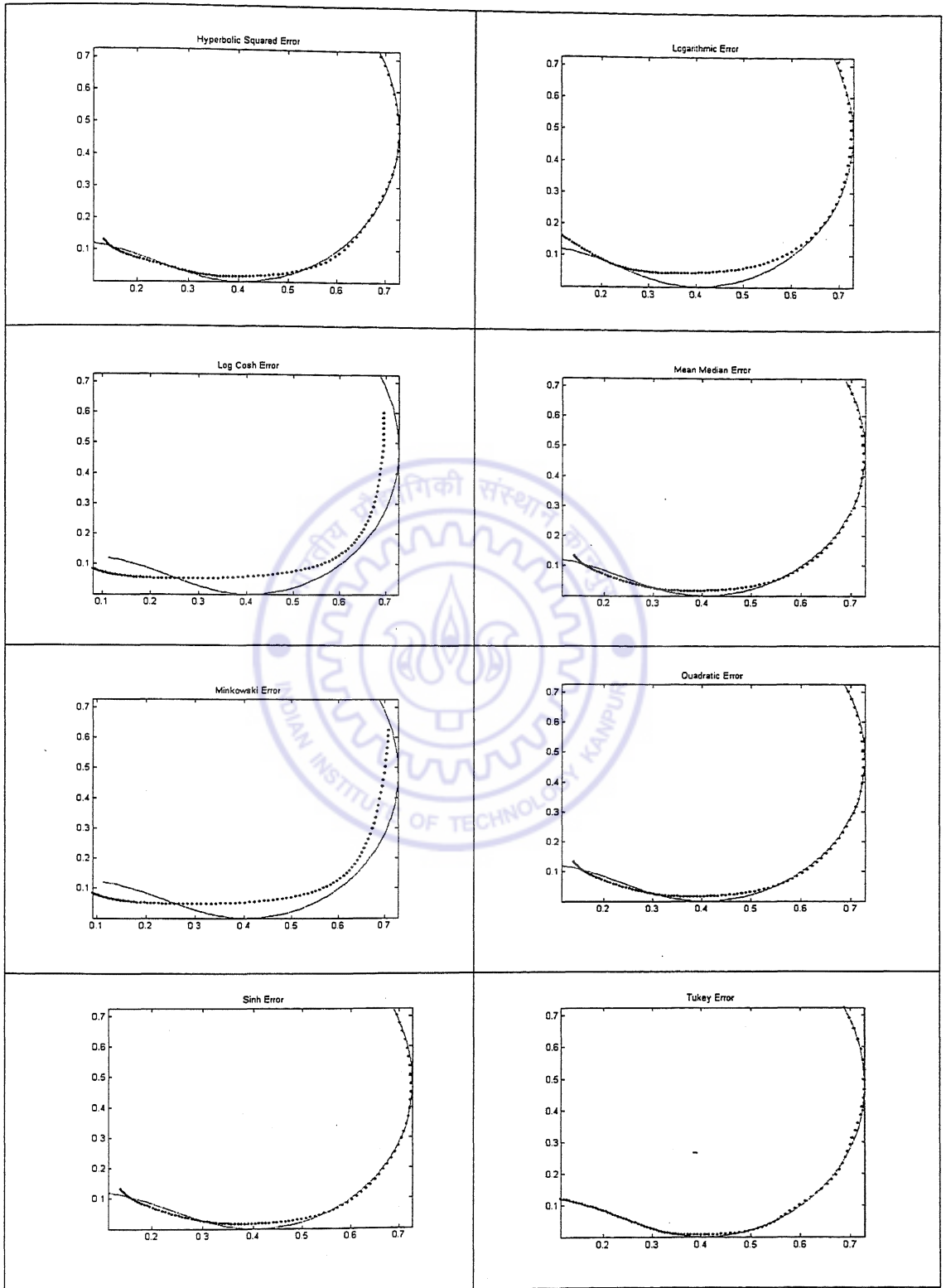


Table 5.13 $w=\exp(z)$ with 1-10-1 architecture, New Activation, 4000 epochs, Learning rate = 0.1. Epochs criterion was set for the training process.





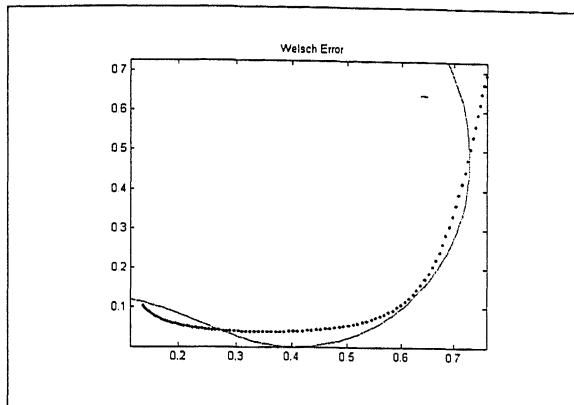
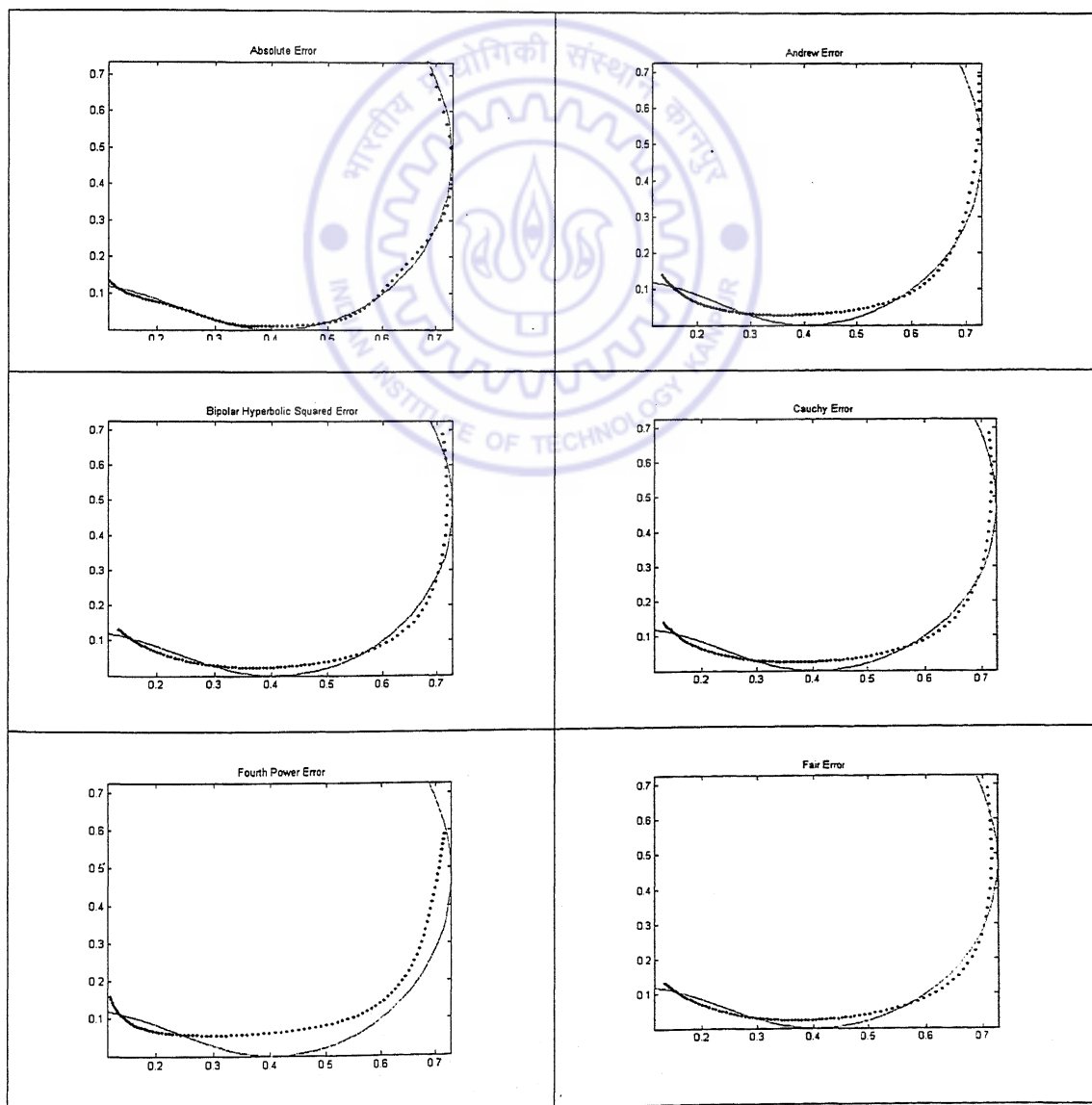
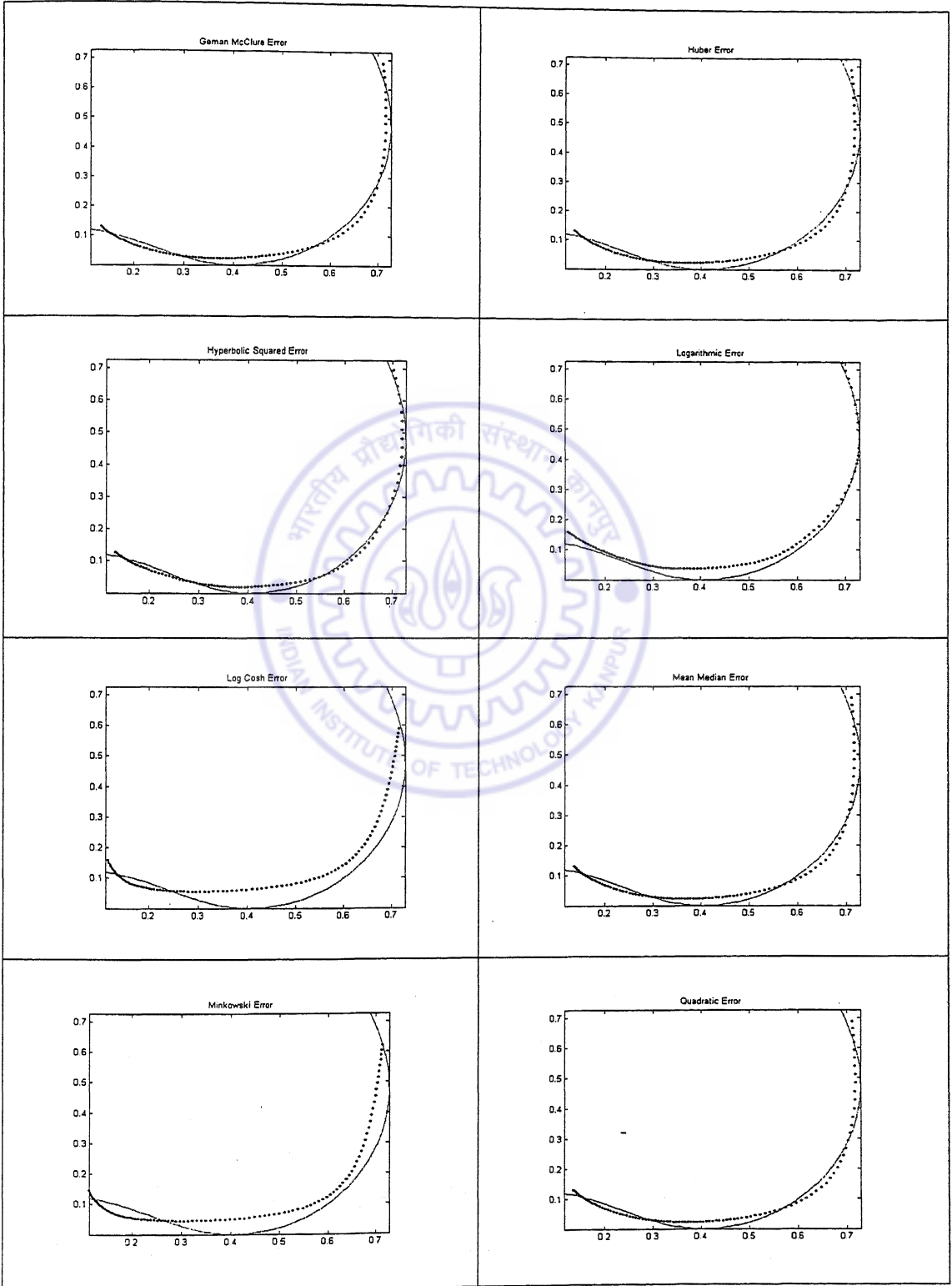
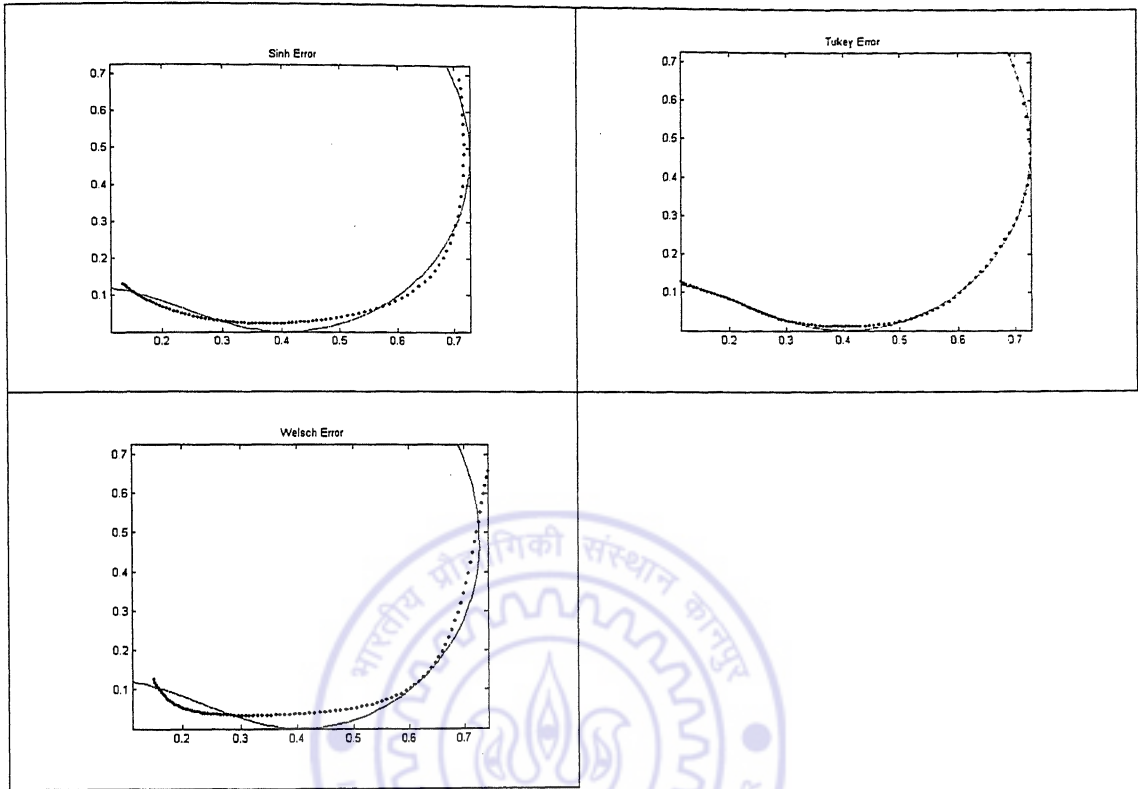


Table 5.14 $w=\exp(z)$ with 1-5-1 architecture, New Activation, 4000 epochs, Learning rate = 0.1. Epochs criterion was set for the training process.







5.2.5 Exponential Map

The exponential map envisaged in this section is a particular case of the function studied in the article 5.2.4; at the same time the map is significant because the imaginary part is zero. The complex variable based exponential function involves capturing the real and imaginary parts of the image of the map separately. The present map is purely real in nature, that is essentially the imaginary part of the function is zero. The function considered for mapping is given by the equation

$$y = 0.3253e^{-x} \quad (5.8)$$

which is the exponential plane curve. The input x was chosen to be a hundred points in the interval $[-0.9, 0.9]$. The testing was done on a hundred points chosen in between bypassing the points chosen for training. A plot of this map is shown in Fig. 5.4. The architecture was 1-5-1, learning rate was 0.1 and epochs were set to 2500. The solid line is the function while dotted lines are the simulated values of the network.

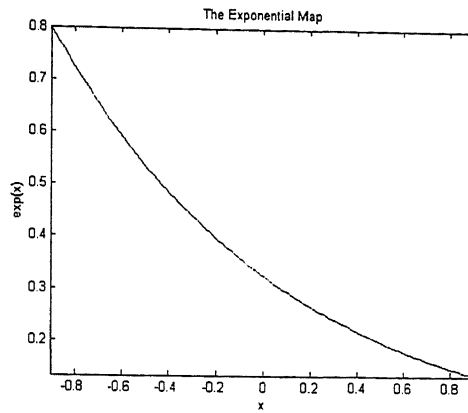
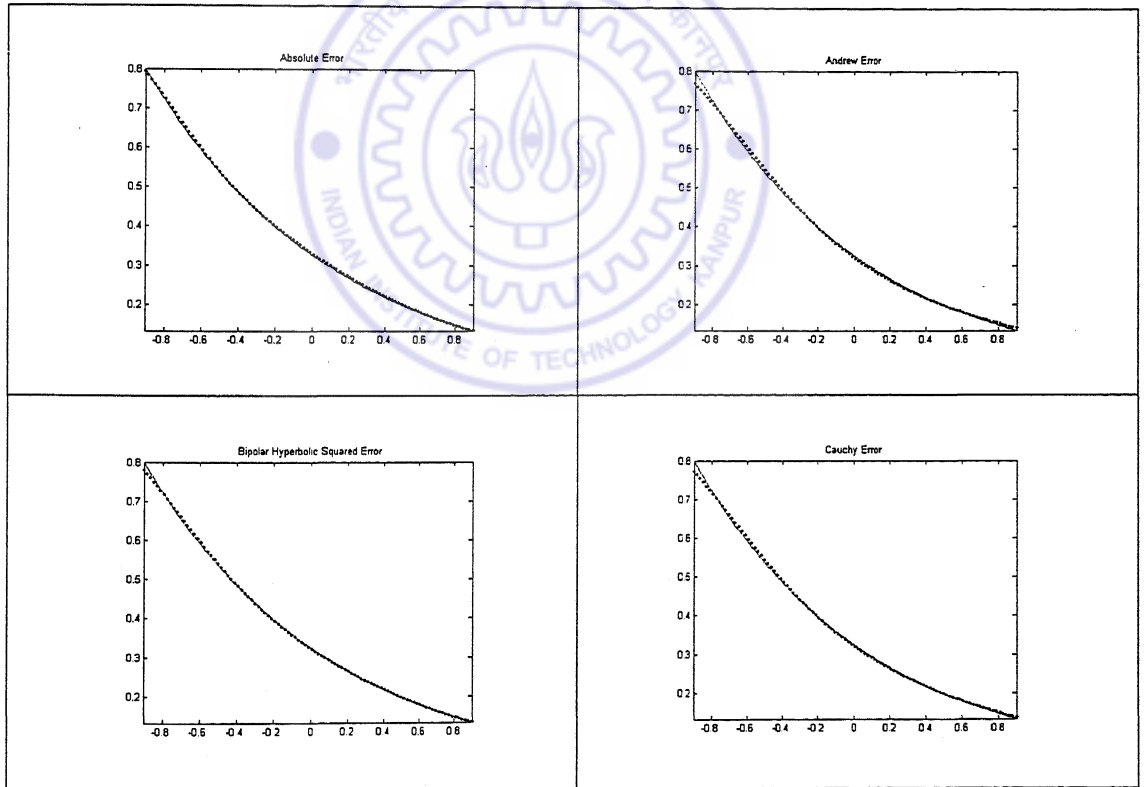
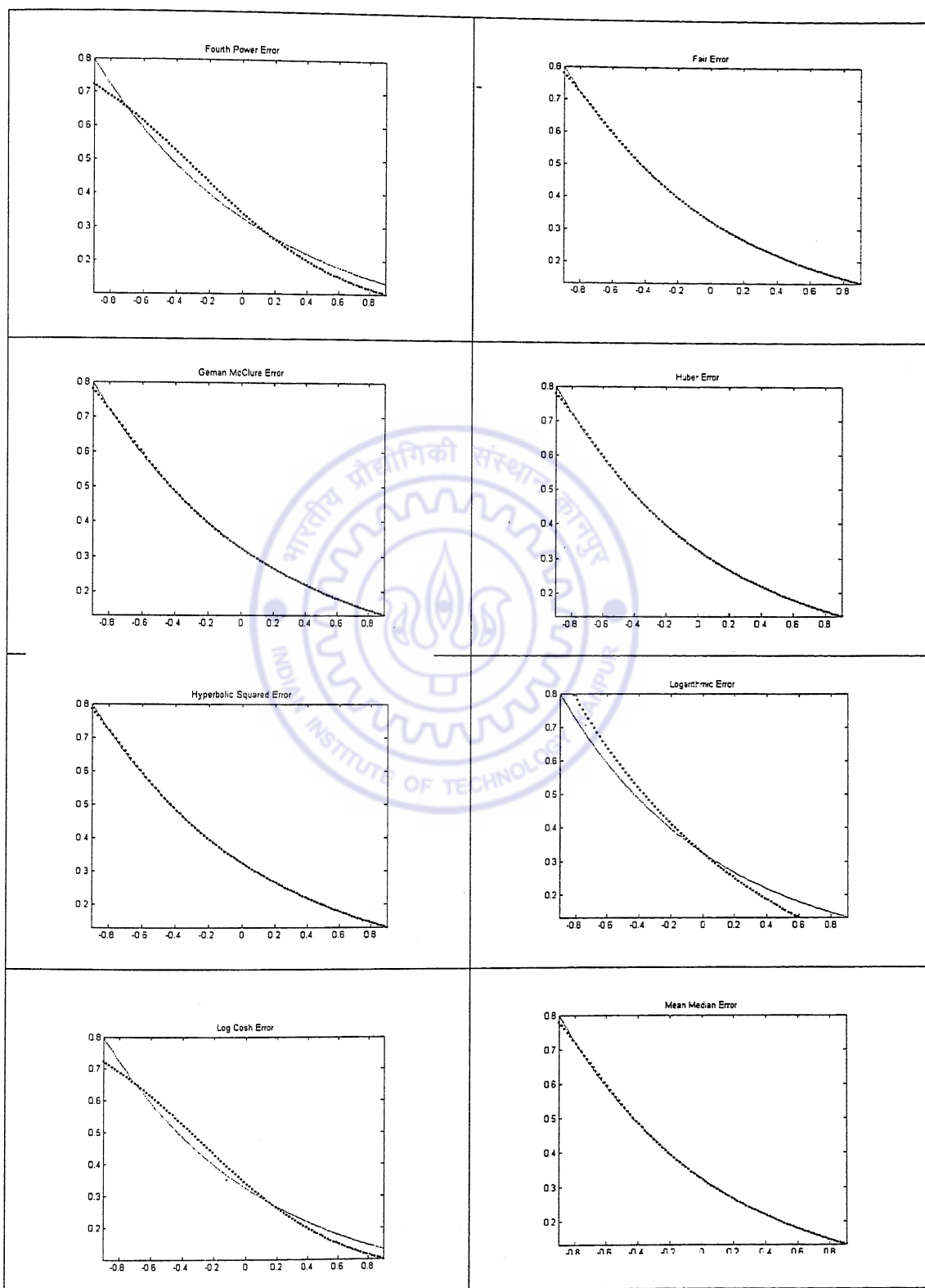


Fig. 5.4 The Exponential Map trained with CNN based on Nitta and New Activation Functions

Table 5.15 $y=\exp(x)$ captured by 1-5-1 Nitta activation based CNNs. Learning rate was 0.1, 2500 epochs. Epochs criterion was set for the training process.





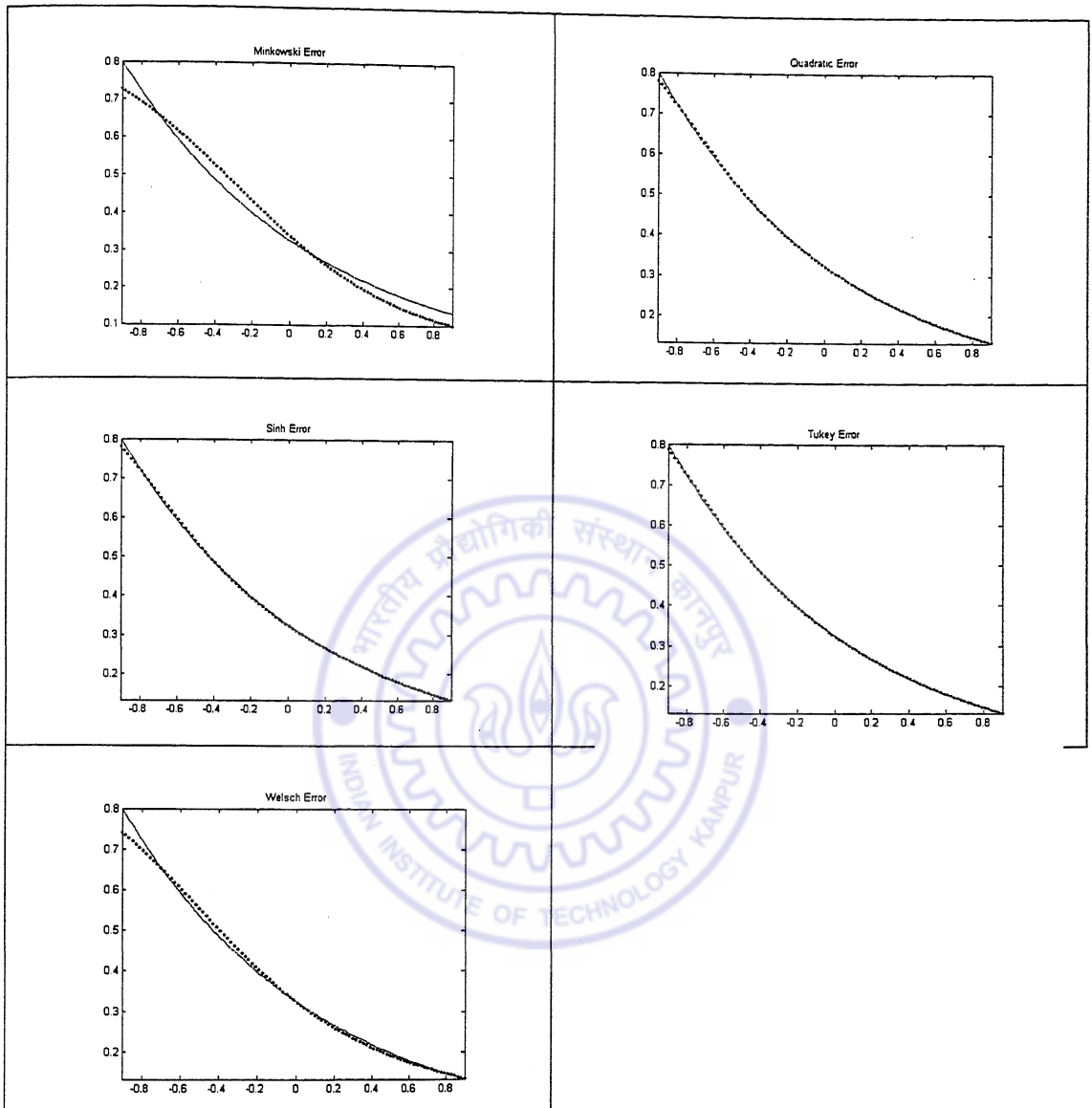
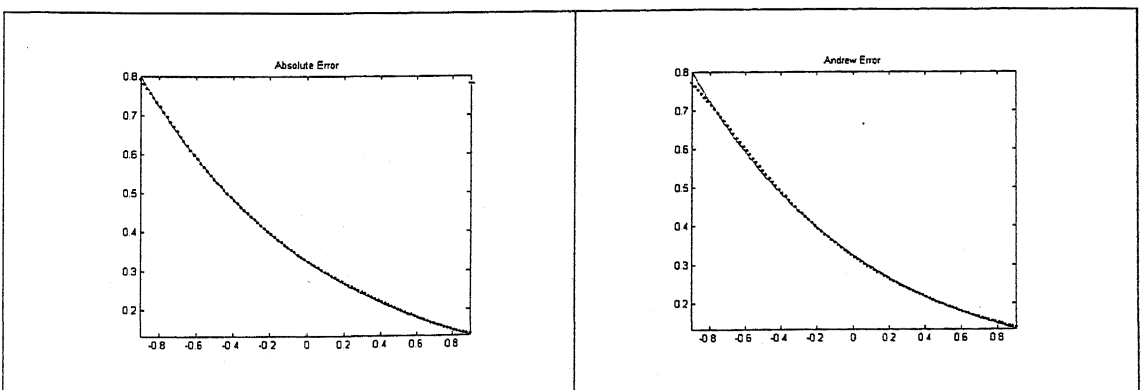
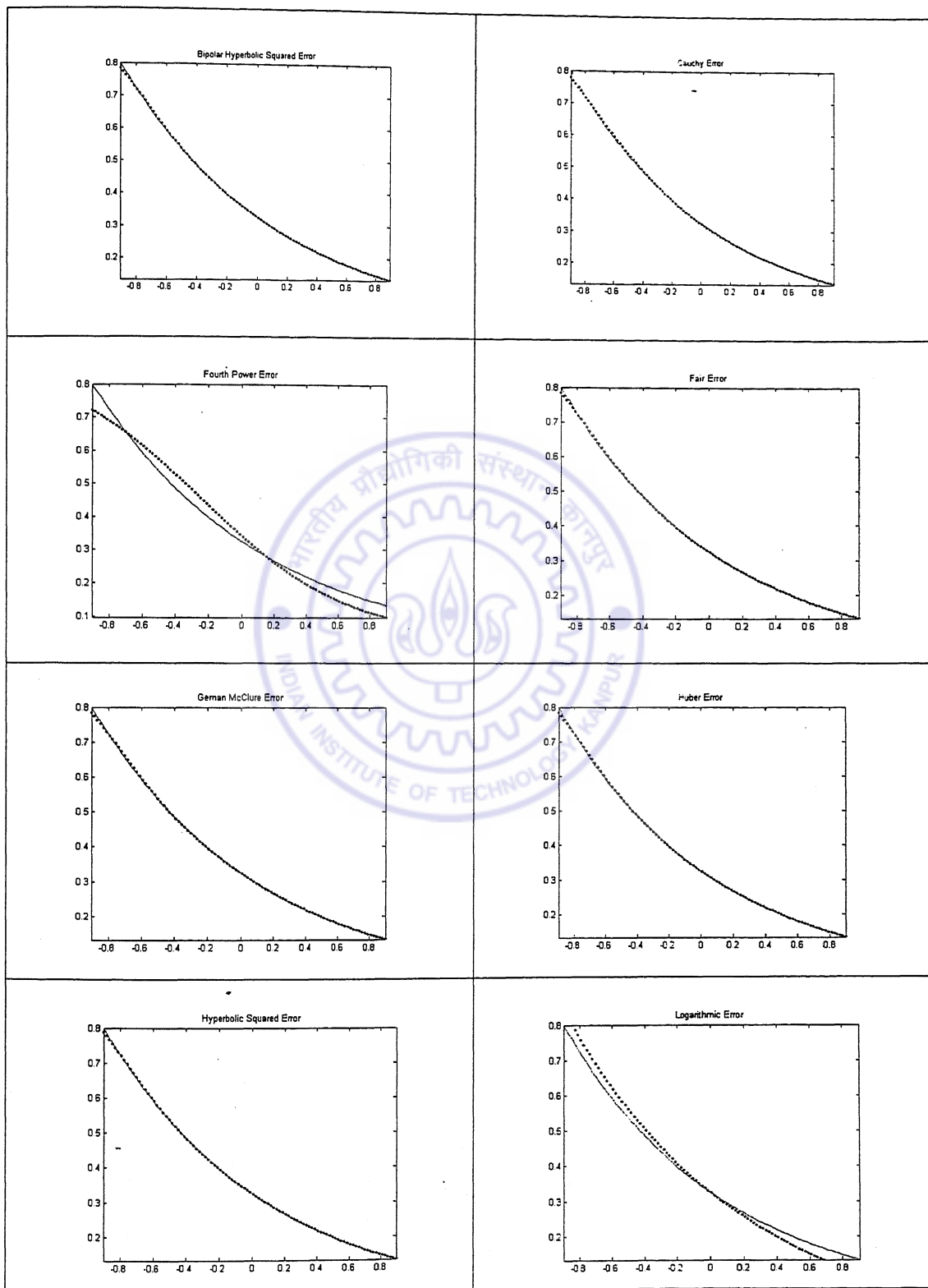
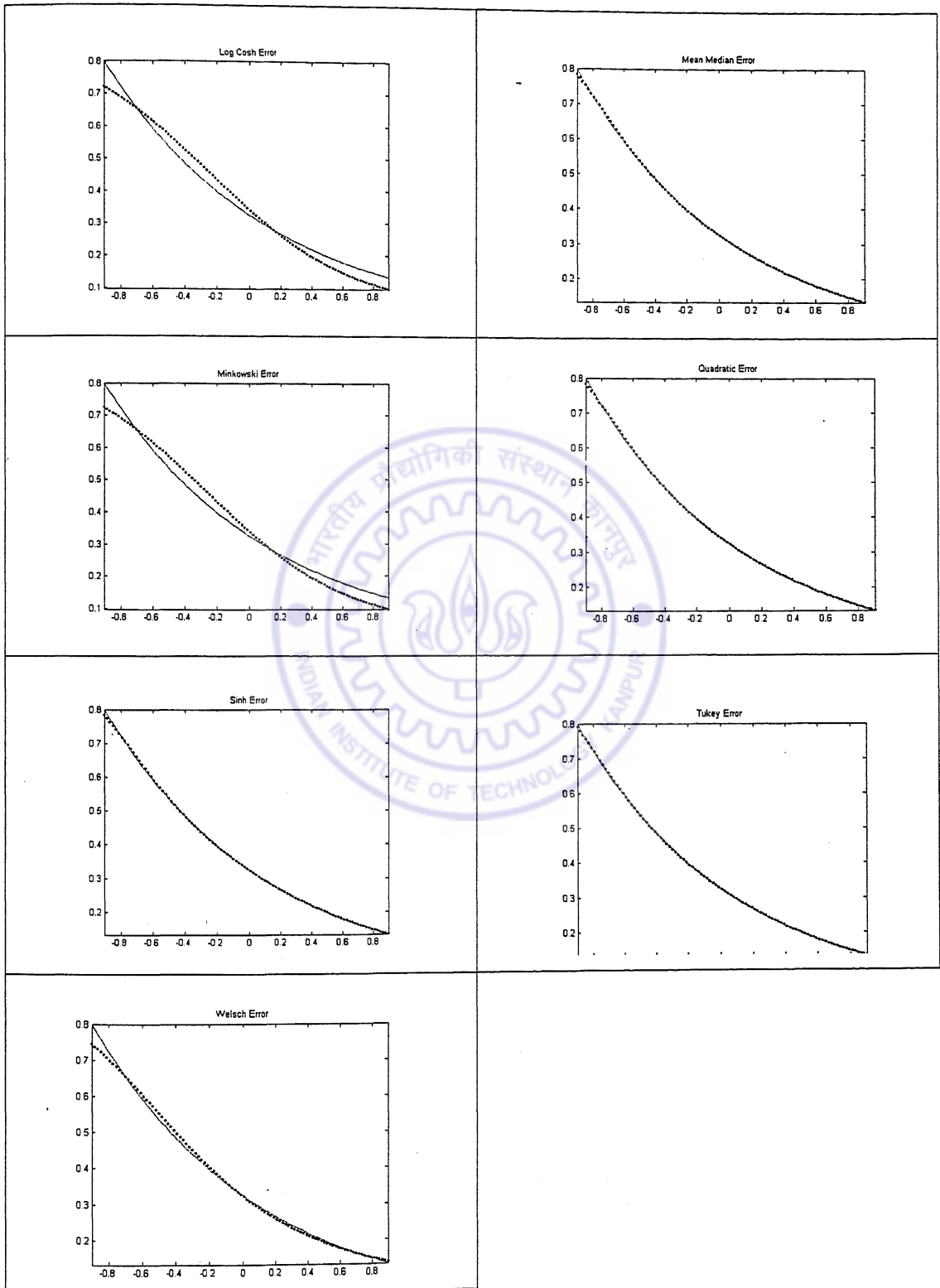


Table 5.16 $y = \exp(x)$ captured by 1-5-1, New activation based CNNs. Epochs criterion was set for the training process.





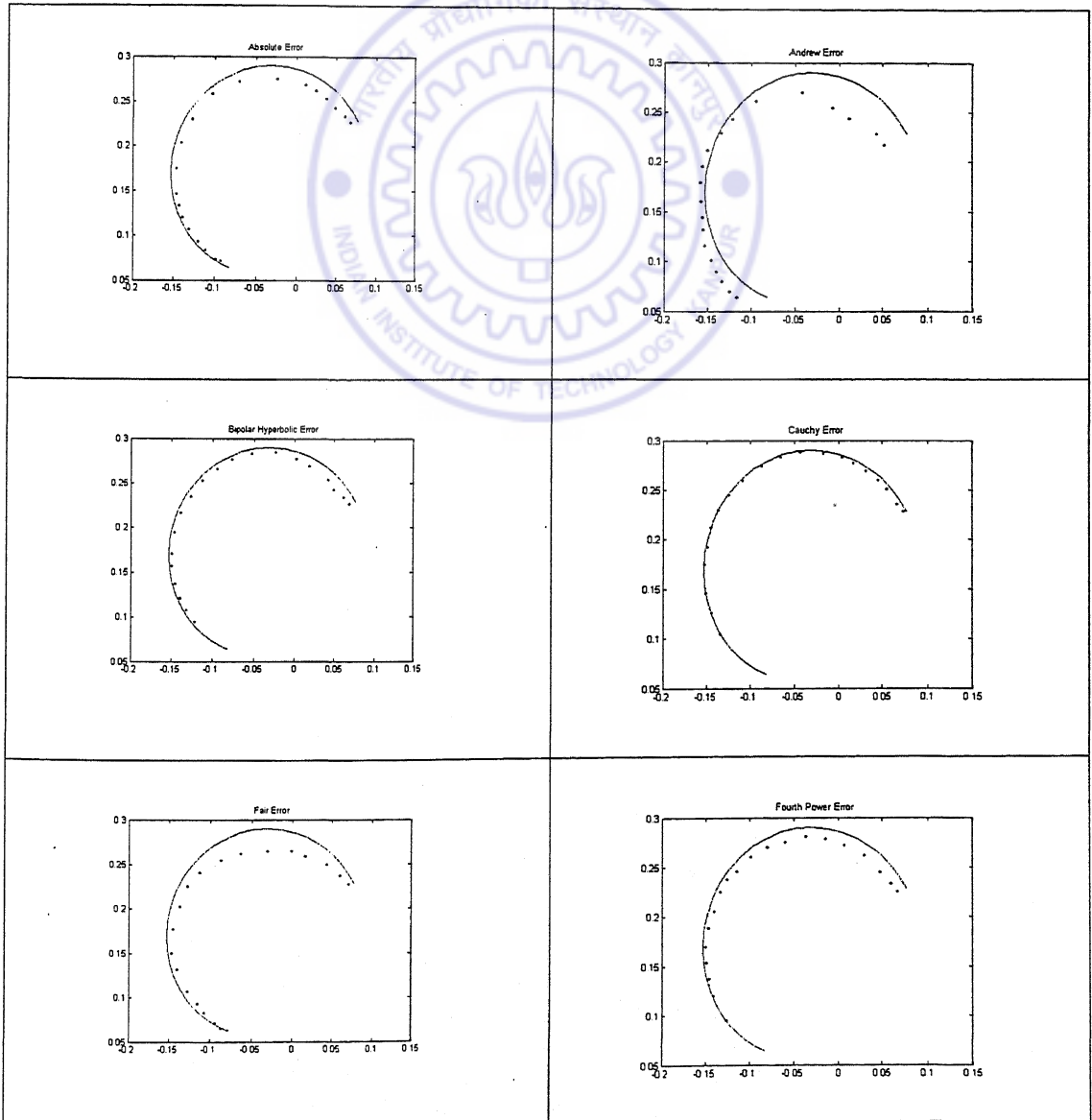


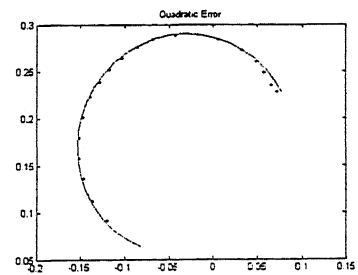
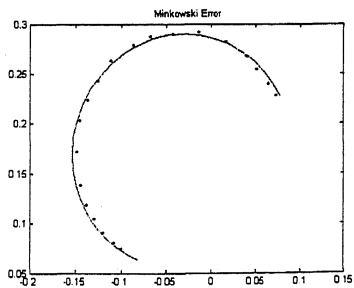
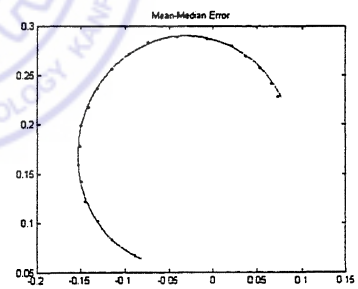
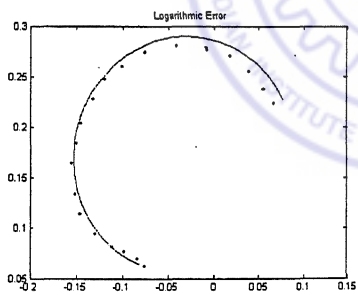
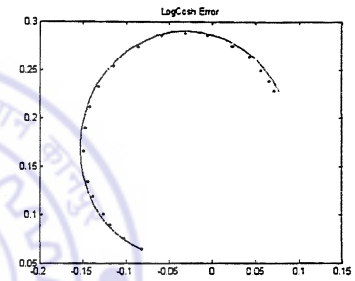
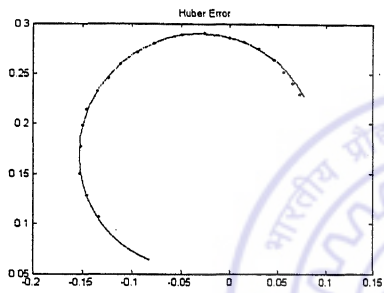
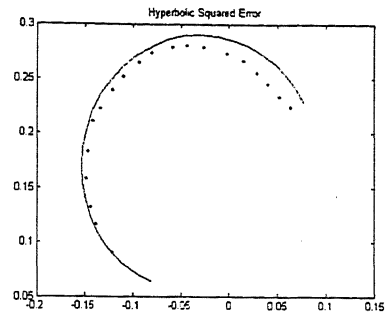
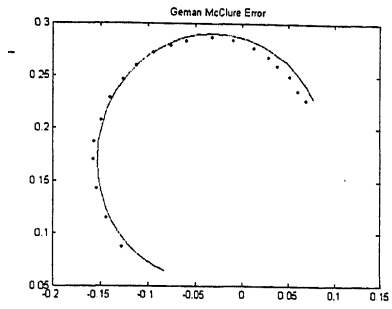
5.2.6 Mapping $w = \sin(z_1)\sin(z_2)$ using a CNN

The Benchmark problem of mapping the surface $w = \sin(z_1)\sin(z_2)$, actually posed for the real variable based BPA was extended to the complex domain and posed as a

Benchmark for the CNN. The actual Benchmark maps a region on the xy – plane to a real number shown plotted on the z – axis, which makes the surface plot. The map was by definition from points on the z_1 plane and the z_2 plane with image on a third plane. Curves were chosen on the first and second planes, the image of the map is plotted on the w -Plane. To train the CNNs, the learning parameter was set at 0.1, the standard parameters were set for each of the Error Functions, the architecture chosen was 1-5-1. The networks were trained for 1500 epochs. The solid line is the actual curve and the dots are the output of the network at twenty intermediate points.

Table 5.17 $w=\sin(z_1)\sin(z_2)$ captured by 1-5-1, Nitta activation based CNNs. Epochs criterion was set for the training process.





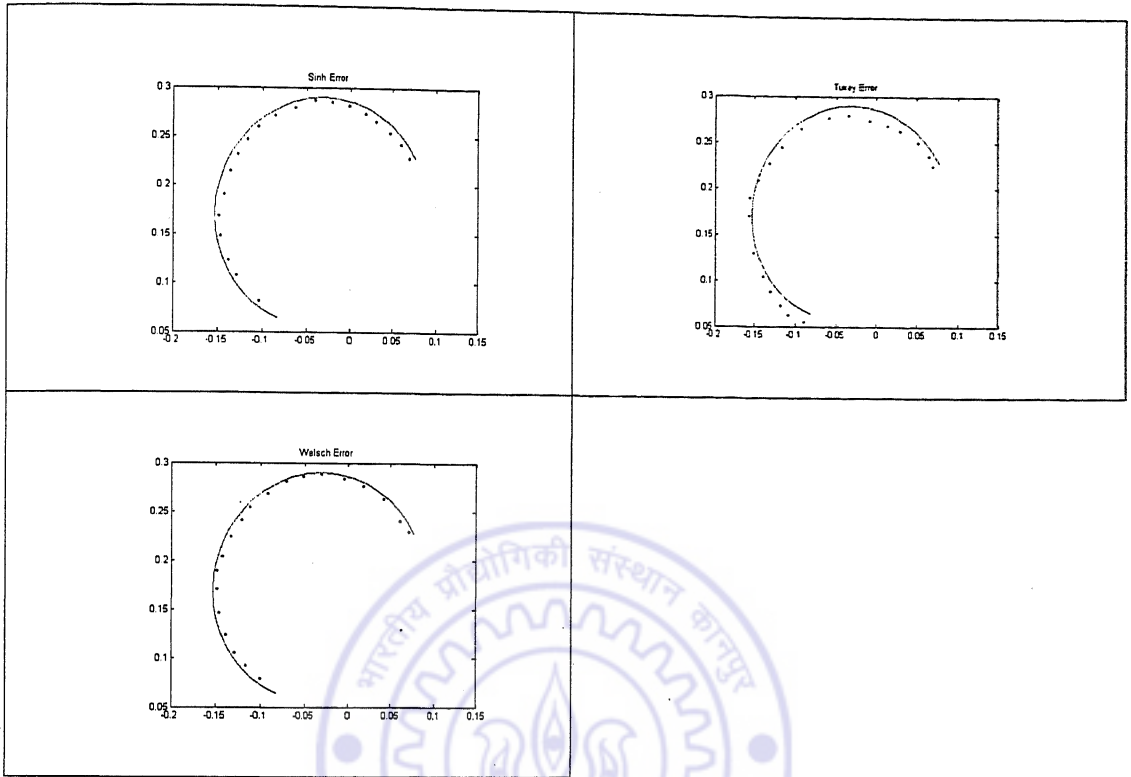
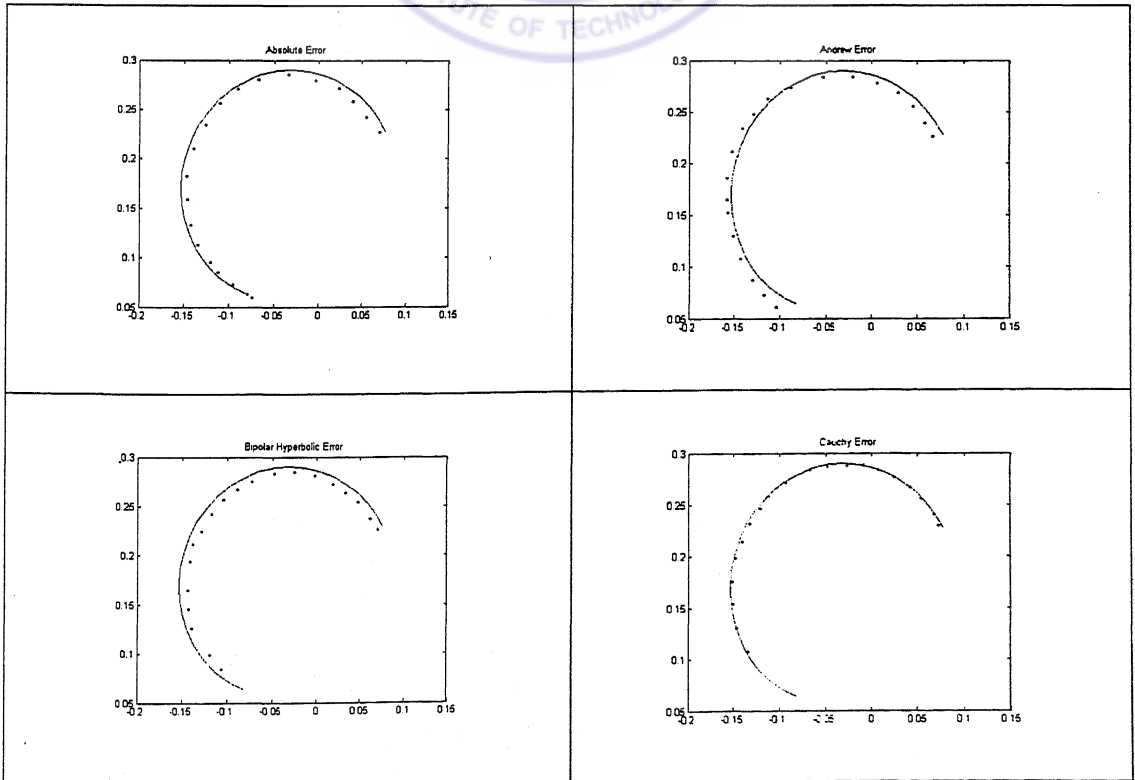
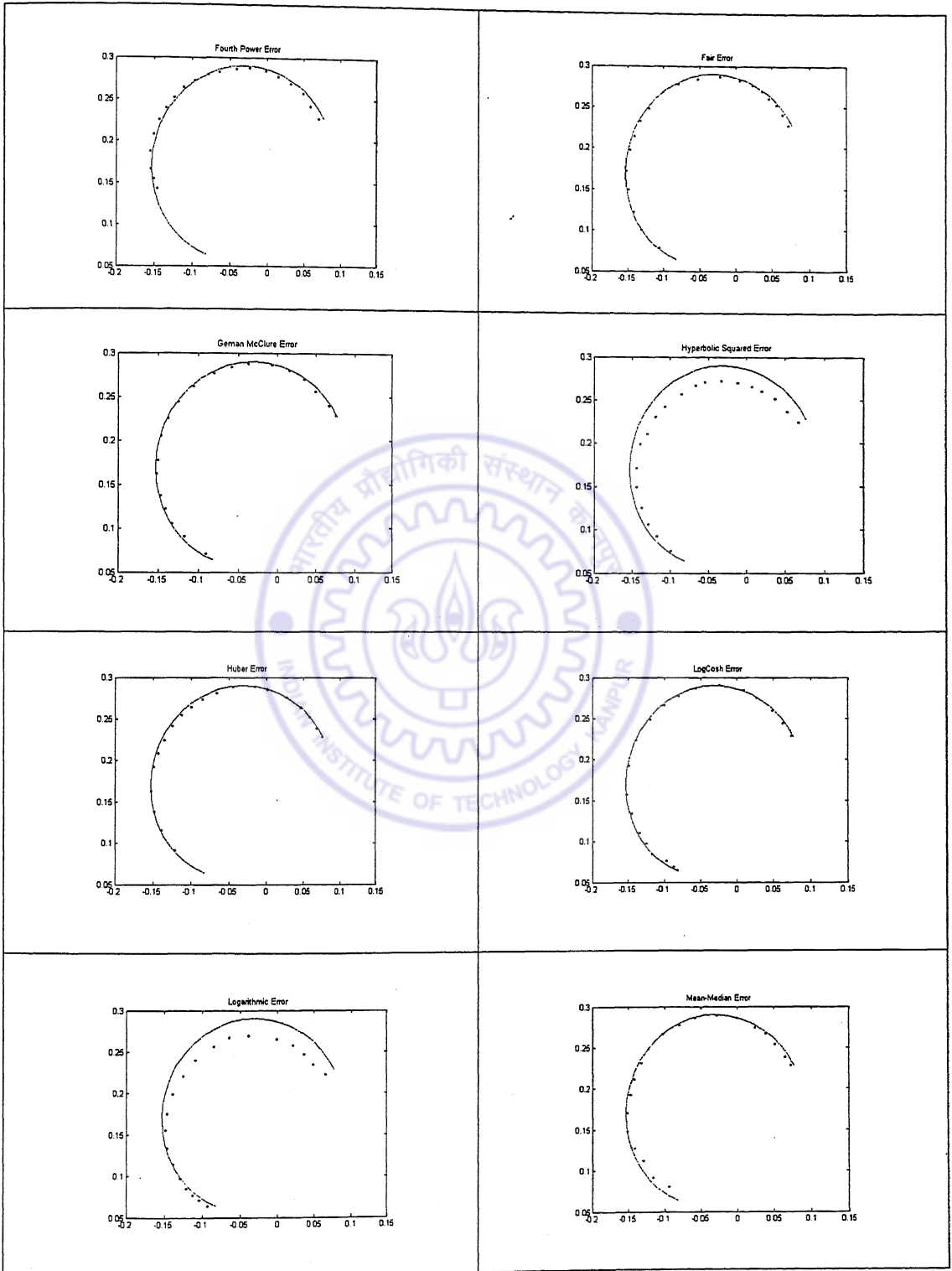
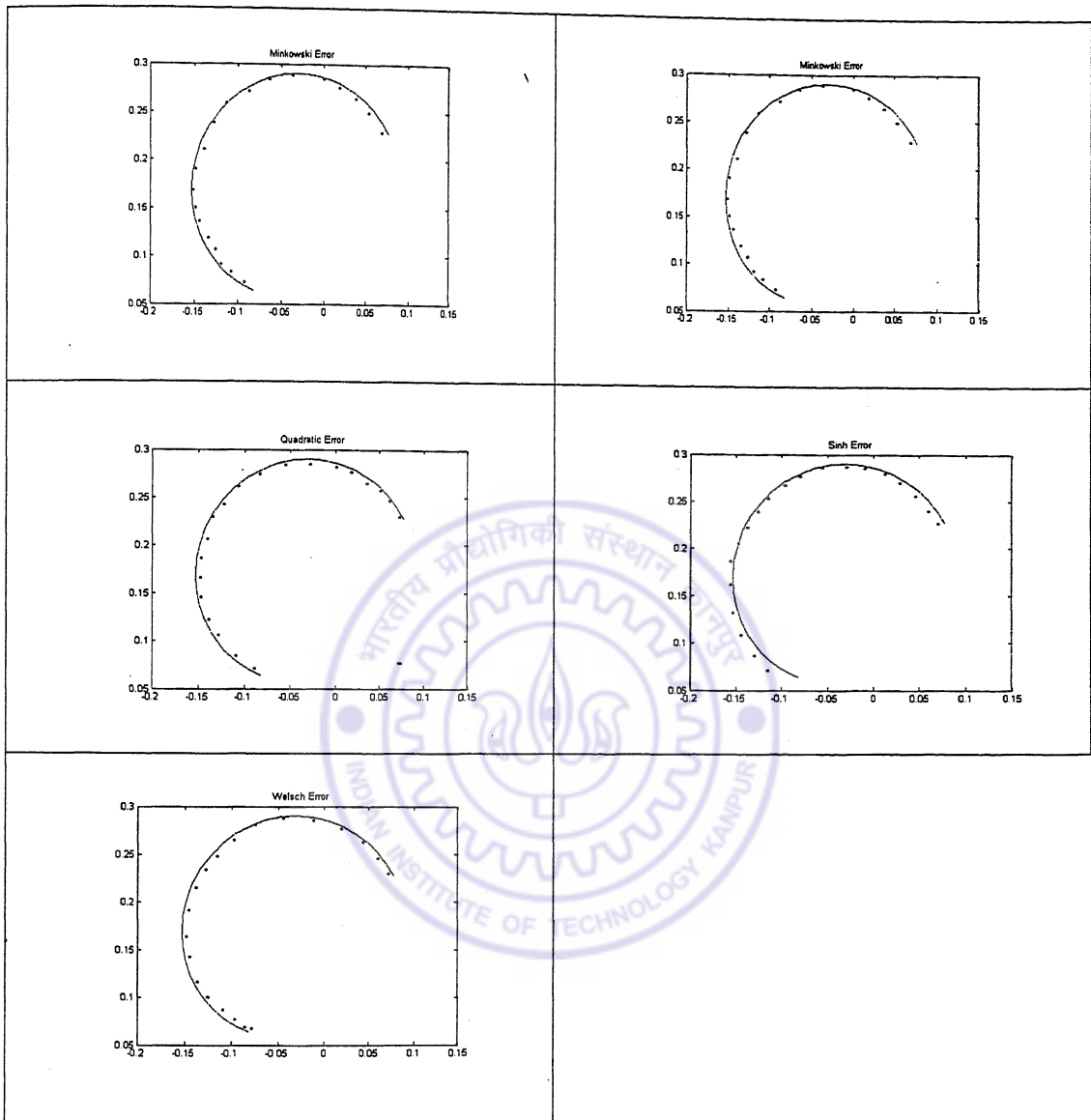


Table 5.18 $w = \sin(z_1)\sin(z_2)$ captured by 1-5-1, New activation based CNNs. Epochs criterion was set for the training process.







The errors produced by each of the CNN based networks for the mapping problems of the present chapter are shown tabulated in Table 5.19. The errors were averaged over three runs of the algorithms.

5.3 Conclusion

The mapping problems demonstrated in this chapter represent typical functions encountered in practice. The results tabulated show that the CNNs capture all the complex maps. The New Activation Function based CNNs also learned all the maps and performed on par with the Nitta activation based CNNs. In particular, the Mean-Median Error, Cauchy Error, Fair Error, Geman-McClure Error, Huber Error and Welsch Error

based CNNs with the New Activation Function performed on par with the Quadratic Error based CNN or better. Hence in mapping applications, the standard CNN can be replaced with the one developed over any of the other EFs listed here.

Table 5.19 Errors with test data averaged across three runs. CExp is Complex Exponential Map, Exp is the Exponential Map, BTrans is Bilinear Transformation, Polynomial and $\sin(z_1)\sin(z_2)$ maps in that order. Epochs have been kept fixed during training (shown bracketed in each column.)

| Average Simulation Error for various Maps | | | | | | | | | | |
|---|---------------------------|---------------|------------------|----------------|------------------|-------------------------|---------------|------------------|----------------|------------------|
| Error Function | Nitta Activation Function | | | | | New Activation Function | | | | |
| | CExp (4000) | Exp (2500) | BTrans (1500) | Poly (1500) | sz1sz2 (1500) | CExp (4000) | Exp (2500) | BTrans (1500) | Poly (1500) | sz1sz2 (1500) |
| Absolute Error | 0.0247 | 0.0013 | 0.0003 | 0.0005 | 0.0012 | 0.0151 | 0.0001 | 0.0011 | 0.0011 | 0.0018 |
| Andrew Error | 0.0435 | 0.0568 | 0.0212 | 0.0035 | 0.1514 | 0.0313 | 0.1167 | 0.1211 | 0.0121 | 0.2223 |
| Cauchy Error | 0.0047 | 0.0014 | 0.0001 | 0.0008 | 0.0001 | 0.0031 | 0.0010 | 0.0005 | 0.0003 | 0.0001 |
| Error Fourth Order | 0.0391 | 0.0307 | 0.0167 | 0.0017 | 0.0040 | 0.0288 | 0.0088 | 0.0089 | 0.0026 | 0.0021 |
| Fair Error | 0.0048 | 0.0012 | 0.0001 | 0.0002 | 0.0001 | 0.0059 | 0.0042 | 0.0003 | 0.0011 | 0.0001 |
| Geman-McClure Error | 0.0026 | 0.0191 | 0.0002 | 0.0014 | 0.0005 | 0.0111 | 0.0087 | 0.0001 | 0.0005 | 0.0007 |
| Huber Error | 0.0029 | 0.0017 | 0.0003 | 0.0006 | 0.0002 | 0.0019 | 0.0006 | 0.0002 | 0.0006 | 0.0002 |
| Hyperbolic Squared | 0.0055 | 0.0377 | 0.0100 | 0.0489 | 0.0015 | 0.0231 | 0.0305 | 0.0561 | 0.0551 | 0.0034 |
| Bipolar Hyperbolic | 0.0026 | 0.0419 | 0.0781 | 0.0056 | 0.0088 | 0.0142 | 0.1044 | 0.0587 | 0.0149 | 0.0112 |
| LogCosh Error | 0.0088 | 0.0005 | 0.0211 | 0.0003 | 0.0622 | 0.0012 | 0.0008 | 0.0101 | 0.0002 | 0.0066 |
| Logarithmic Error | 2.9730 | 0.0253 | 0.5869 | 0.2828 | 0.0466 | 0.8816 | 0.0812 | 0.8979 | 0.2116 | 0.0867 |
| Mean Median Error | 0.0026 | 0.0011 | 0.0002 | 0.0021 | 0.0259 | 0.0011 | 0.0031 | 0.0011 | 0.0002 | 0.0033 |
| Minkowski Error | 0.0308 | 0.0051 | 0.0126 | 0.0023 | 0.0026 | 0.0404 | 0.0017 | 0.0091 | 0.0022 | 0.0033 |
| Quadratic Error | 0.0026 | 0.0003 | 0.0003 | 0.0004 | 0.0006 | 0.0008 | 0.0003 | 0.0001 | 0.0003 | 0.0003 |
| Sinh Error | 0.0031 | 0.0012 | 0.0003 | 0.0004 | 0.0004 | 0.0022 | 0.0004 | 0.0003 | 0.0004 | 0.0004 |
| Tukey Error | 0.0019 | 0.0006 | 0.0002 | 0.0018 | 0.0004 | 0.0082 | 0.0011 | 0.0001 | 0.0025 | 0.0011 |
| Welsch Error | 0.0352 | 0.0013 | 0.0100 | 0.0102 | 0.0042 | 0.0081 | 0.0029 | 0.0094 | 0.0088 | 0.0031 |

The table of errors reveals the different rates at which the problem solution was possible as the EFs varied. This establishes a sequence of preference with the choice of EF so that an optimal selection of EF could be made to suit the application at hand. Huber Function, Geman McClure Function, Cauchy, Tukey, Quadratic, Log Cosh and Welsch have performed consistently well across the varied mapping problems. And hence, these functions are recommended for other applications to replace the Quadratic Function. The other EF based CNNs were slow on the convergence compared with the ones mentioned. Unlike the Surface Classification Problem (Chapter 6) where some functions were not recommended owing to their poor performance, all the EF based CNNs can be applied but the different rates of convergence should of course be considered while making a judicious choice.

Chapter 6

Application to Surface Classification Problem

6.1 Introduction

The problem of classification in general refers to sorting a set of elements into categories with predefined characteristics, in which the number of classes is usually fixed (this is opposed to clustering problems where the number of clusters can be treated as unknown. Moreover clustering is a constrained extremization problem (Babuska, 1998) while classification is a problem of mapping). A mathematical approach to the same would involve reposing the problem in terms of functions and maps. Classification from a mathematical viewpoint is a map that assigns a fixed number to elements belonging to one set, a second fixed number to elements of the second set and so on. Classification problem as opposed to curve fitting is a many-to-one mapping in which the class representative is the target. The above distinction was brought about to underline the fact that the Neural Network performs a map be it curve-fitting (where the mapping is one-to-one) or classification (in which the map is many-to-one).

The CNN was applied to the various benchmarks and mapping problems in the previous chapters. It should be noted that all maps considered were one-to-one. It is essential to establish the properties of the CNN as a classification tool so that the same can be employed practically and reliably to many-to-one maps. In the present chapter the problem of sorting point clouds in algebraic and transcendental types is considered. The problem is solved with the ANN varying the Error Function. Error Functions based CNN's were later used to address the same problem. The results of these networks were studied from a comparative viewpoint.

6.2 Point Clouds in Practical Application

The problem of Reverse Engineering (Ingle, 1994), deals in constructing surfaces from clouds of points. A typical problem in the area has a data set at hand obtained by running a scanning-equipment across the object of interest. The scan must be performed in an orderly fashion to ensure the data points are well organized (and do not appear at irregular intervals or appear disorderly). After the operation, a point cloud of the object gets generated that is

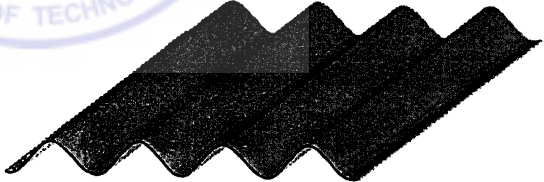
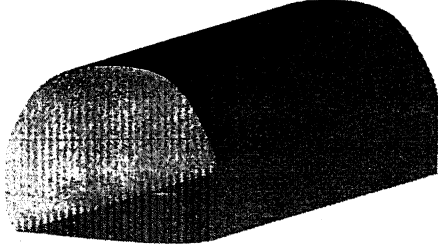
next subject to further analysis. As a second step in the process, patches of surface are fit to small sets of the data points of the cloud and the procedure repeated for sets of points in a close vicinity. The whole point cloud gets an approximation as the surface patches can be thought of as sewn along the boundaries. The usual method used to generate the patches is the one that employs methods of Rational B-Splines (Farin, 1995). Polynomial functions are also used to fit the point clouds with patches.

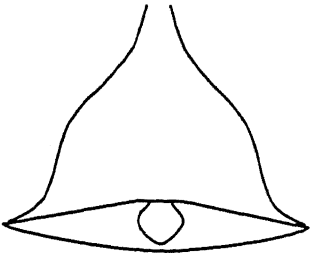
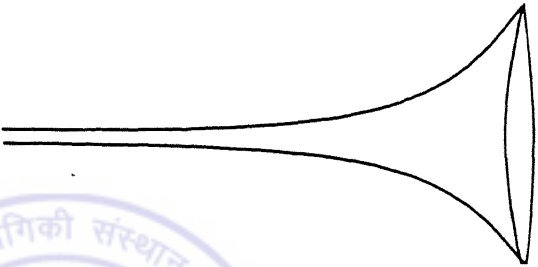
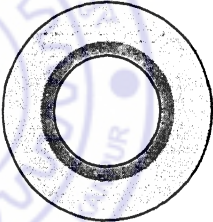
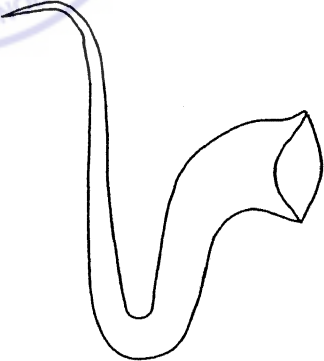
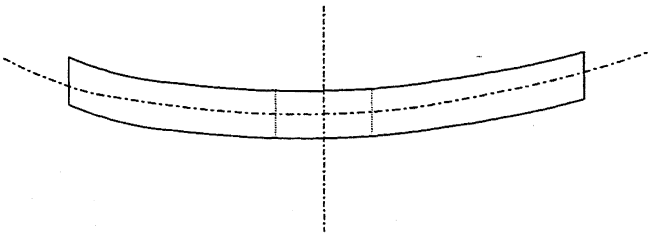
It is evident from the standard procedure employed to approach the problem that the set of points are approximated by Rational B-Splines or Polynomials while the actual function from which the object was designed remains unknown. So essentially only algebraic functions (polynomials) are employed in almost all cases to fit point clouds. It can be seen however that the actual form of the surface if known would give a more accurate picture of the surface and the analysis following the step more correctly placed. The analysis in question could be one of computing stresses if the element were a machine component for example or computing the frequency distribution if it were a vibrating part of a machine or the air-column of a musical instrument. This aspect is usually not addressed because the approximation with algebraic polynomials suffices for the experiment in question. As the function form of the surface would be unknown in most cases, the best one can do is develop steps that can provide information about what kind of function better approximates the surface – algebraic or transcendental. The problem hence can be reposed as a problem of Classification to enable a Neural Network based judgement. If a scheme exists by which the type of function that fits the point cloud could be sensed, the surface approximation procedure can be bettered as an appropriate function type can be chosen for approximating the point cloud. In fact there exist problems that demand the correct form of the surface be known while reverse-engineering. The following example illustrates where a classification of this sort is inevitable. The actual surface of the saxophone (or a trumpet) is the surface of revolution of the exponential curve (Fletcher and Rossing, 1995). The shape is arrived at by modeling the problem and subjecting to variational methods of solution. Now, on approximating the surface of the instrument with a polynomial for vibration analysis of the air-column, the tonal quality and harmonic distribution get dislocated as the exponential function is the optimal surface for the instrument and hence, only a function of the type e^{-kx} (for some k) can be taken as the point of start. This problem has a stringent constraint in terms of tonal quality and harmonics that should be always met if an analysis of the harmonics based on the profile of the instrument were to be carried out. So in general, surfaces that are optimal solutions to variational

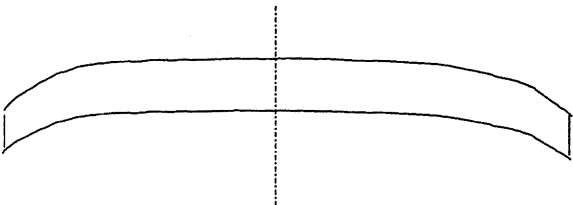
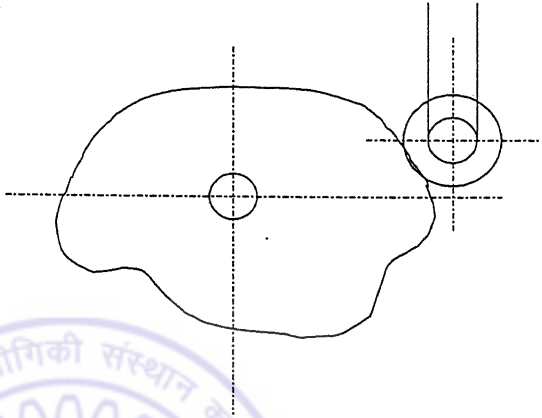

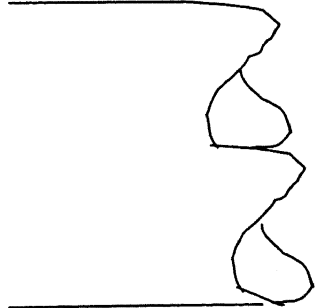
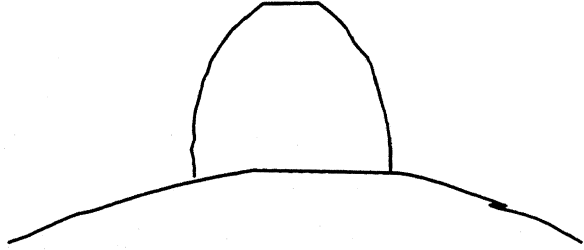
problems demand the constraint of the problem be always met even while choosing an approximation. It is clear from this example that in problems of this sort (in which the constraint is extremely important), it becomes necessary to know the kind of approximating function. It is here that it becomes necessary that we retain the form of the function even while selecting functions for approximating and a polynomial form wouldn't suit the problem if the optimal solution were not it. In light of this fact, to address such applications as the one projected here, a study of algebraic and transcendental based classification of surfaces is essential.

To place the argument on a firm ground the following surfaces that arise practically at various places are listed (from design point of view involving developing surfaces for the first time or analysis viewpoint that typically involves studying the surface of a given object by employing an appropriate method like stress analysis, vibration analysis). Table 6.1 shows that transcendental surfaces occur at many places in practice.

Table 6.1 List of Transcendental Surfaces in Practice

| | |
|--|--|
| <p>The asbestos sheet is a Transcendental Surface</p> |  |
| <p>The profile of the godown is the plane curve, cycloid extended cylindrically.</p> |  |

| | |
|---|--|
| <p>The church bell is the Gaussian Surface obtained by rotating the Gaussian Function about the axis of ordinates</p> |  |
| <p>The Clarinet is the Surface of revolution of the exponential curve.</p> |  |
| <p>The torus is the surface of revolution of one circle about another.</p> |  |
| <p>The Saxophone is the surface of revolution of the plane exponential curve about the axis of ordinates and later bent into the shape shown.</p> |  |
| <p>Circular plate with a circular bore at the center assumes a logarithmic surface when an axial force acts on it</p> |  |

| | |
|---|--|
| <p>Cylindrical bending of a thick plate on an elastic foundation</p> |  |
| <p>Cam with a part cycloid, part cardioid and part sinusoidal profile with a roller follower. The curves that make up this cam are transcendental in nature.</p> |  |
| <p>Simply supported isosceles triangle with simply supported edges. The deflection surface of the buckled plate that satisfies the boundary conditions is shown here. The surface is $z = \sin(x)\sin(y)$.</p> |  |
| <p>The spiral jaw clutch is a good example of a transcendental surface in practical application.</p> |  |
| <p>The involute profile teeth of the spur gear is a transcendental surface</p> |  |

In fact, mathematically it was proved that the set of transcendental functions is more abundant than the set of polynomials (Lang, 1966). The problem here is reposed as a problem of classification by constructing data sets from known algebraic and transcendental functions. The classification stated this way implies telling apart algebraic surfaces from transcendental surfaces. To start with, the BPA and some variants were run to train the Neural Network architecture of appropriate size. To keep the study uniform, a 100-5-1 architecture was chosen. A learning rate of 0.1 was set and the Network was trained for 10000 epochs. Hence an epochs criterion was chosen for training the network. To test the Neural Network three sets of surfaces were constructed. Four different algorithms were used to train the Neural Networks.

6.3 Training Surfaces

Twenty-five surfaces, shown in Table 6.2 (a1)-(y1), were considered for training the first stage of network to identify the surface type. The surface equations are also given. Surfaces have been denoted as TrS1, TrS2,...,TrS25. These surfaces were chosen to include symmetry (rotational, symmetry about an axis, symmetry about sectional planes) and polynomial order up to five. Both bounded and unbounded surfaces were included in the training set (Wexler (1962), Musili (1990), Stein (1987)). These surfaces include among the algebraic surfaces, quadratic, bi-quadratic and quintic equations (Sommerville, 1951) while the transcendental surfaces include functions involving sine and exponential function. Surfaces involving product and sum of algebraic and transcendental functions, e.g. TrS18 and TrS22, respectively, have also been included in the training set. The other properties of the networks were kept identical for the comparison to be facilitated.

A ten-dimensional equi-spaced vector was considered in the interval $[-1,1]$ on the x -axis. A similar vector was considered on the y -axis also. These points hatch out a hundred coordinates (x_i, y_i) on the xy -plane (Fig. 6.1), with i taking values from 1 to 100: this is a 10×10 matrix. The points are numbered column-wise (that is, the coordinate $(-1,1)$ is numbered one. The coordinate $(-1,-1)$ is numbered 10, the coordinate $(-0.7778,1)$ is numbered 11 while $(-0.7778,-1)$ is numbered 20 and so on; hence $(1,-1)$ is the hundredth coordinate in the data vector). At each of these hundred points, z -coordinates were computed for each surface using their governing equations. The point clouds thus generated for the twenty-five surfaces are shown in Table 6.2 (a2)-(y2). The z -coordinates at the 10×10 matrix points were stacked into a single vector of dimension 100×1 . The data vector thus generated is normalized

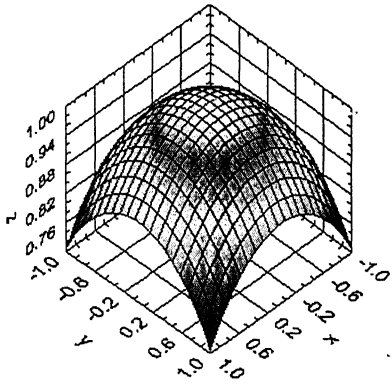
by dividing all entries of the vector by the magnitude of the vector joining the farthest point in the data set. This vector constitutes one column of the 100x25 matrix, which forms the input to the neural network.

Table 6.2 Algebraic and Transcendental Surfaces

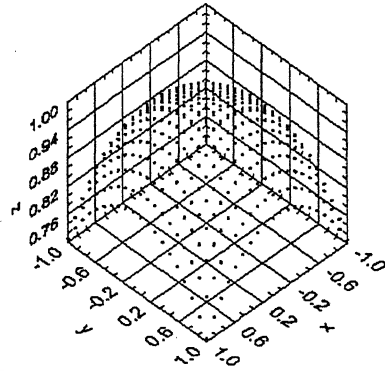
| | |
|---|--------------------------|
| $\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$ | |
| (a1) TrS1 | (a2) Point Cloud of TrS1 |
| | |
| $z = \frac{x^2}{a^2} + \frac{y^2}{b^2}$ | |
| (b1) TrS2 | (b2) Point Cloud of TrS2 |
| | |
| $z^2 = \frac{x^2}{a^2} + \frac{y^2}{b^2}$ | |
| (c1) TrS3 | (c2) Point Cloud of TrS3 |
| | |

$$x^2 + y^2 + z^2 = 1$$

(d1) TrS3

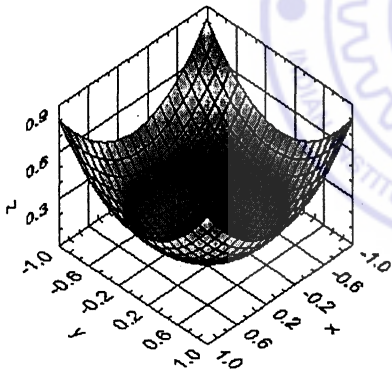


(d2) Point Cloud of TrS4

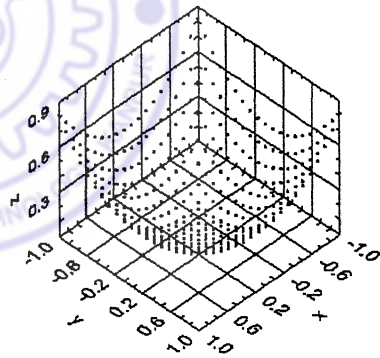


$$z = x^2 + y^2$$

(e1) TrS5

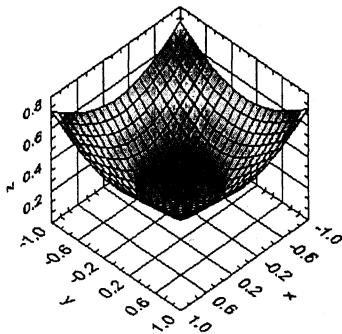


(e2) Point Cloud of TrS5

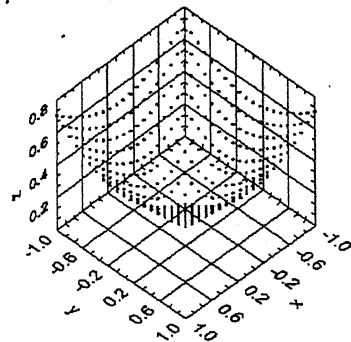


$$z^2 = x^2 + y^2$$

(f1) TrS6

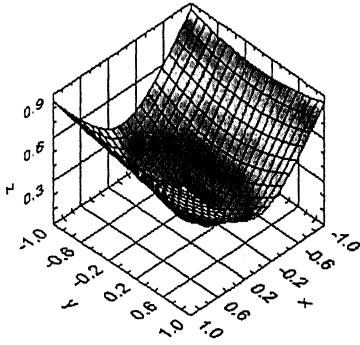


(f2) Point Cloud of TrS6

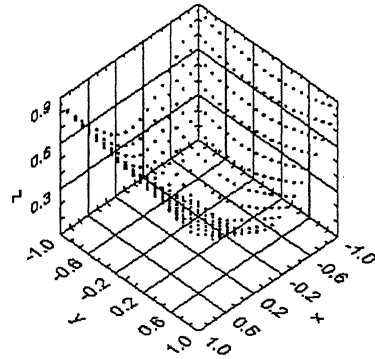


$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1$$

(g1) TrS7

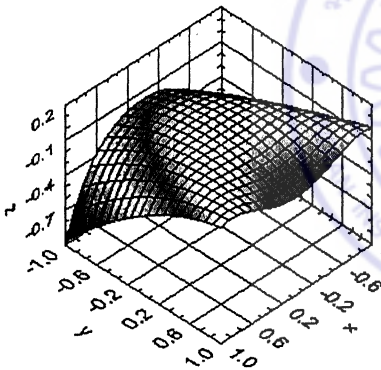


(g2) Point Cloud of TrS7

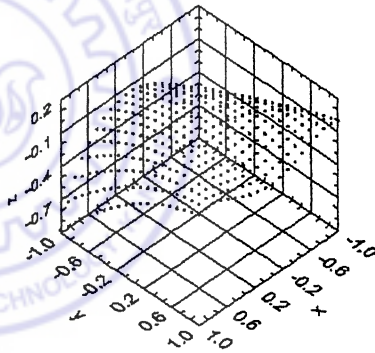


$$z = (2x^2 - y)(y - x^2)$$

(h1) TrS8

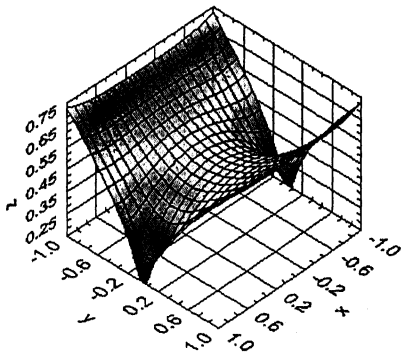


(h1) Point Cloud of TrS8

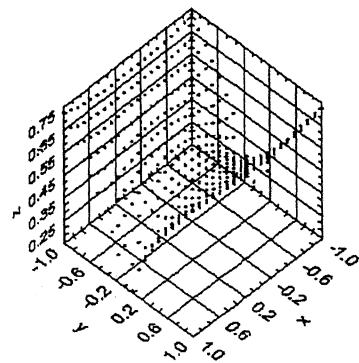


$$z^4 = [(x-a)^2 + y^2][(x+a)^2 + y^2]$$

(i1) TrS9

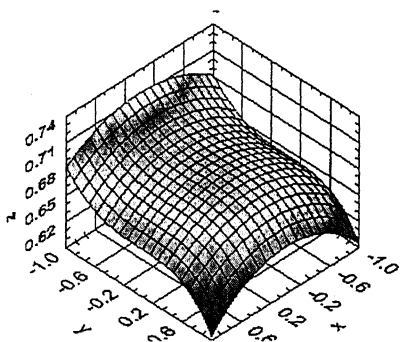


(i2) Point Cloud of TrS9

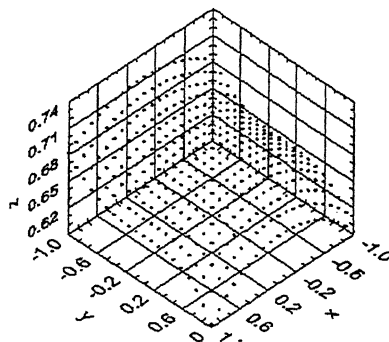


$$x^2 + y^3 + z^5 = 1$$

(j1) TrS10

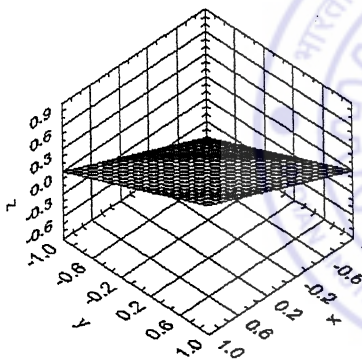


(j2) Point Cloud of TrS10

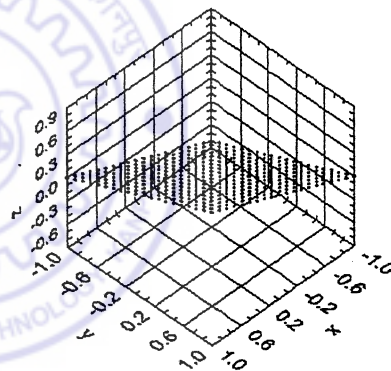


$$z = x + 2y$$

(k1) TrS11

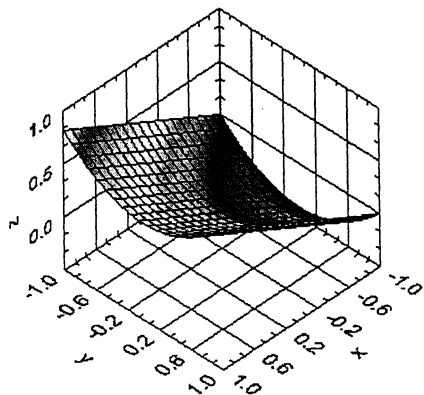


(k2) Point Cloud of TrS11

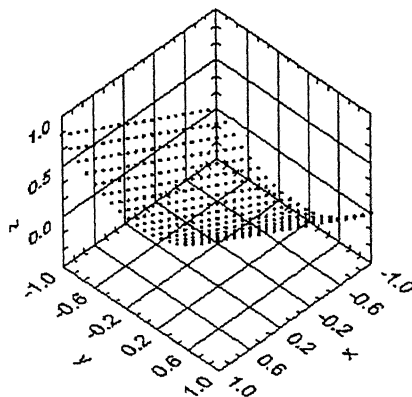


$$z = x + y^2$$

(l1) TrS12

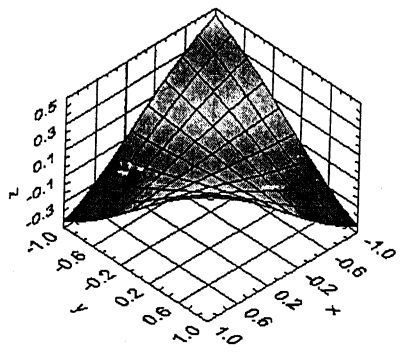


(l2) Point Cloud of TrS12

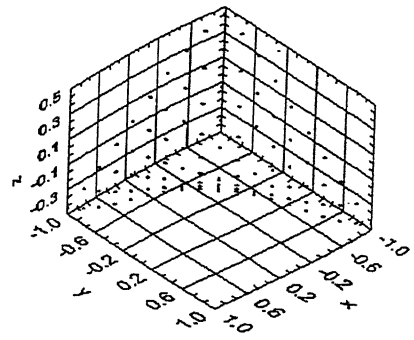


$$z = \sin(x) \sin(y)$$

(m1) TrS13

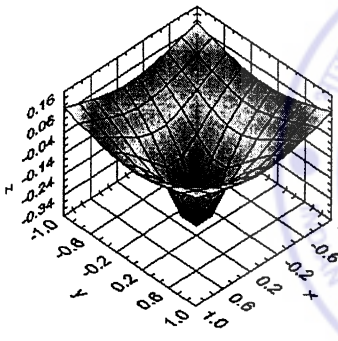


(m2) Point Cloud of TrS13

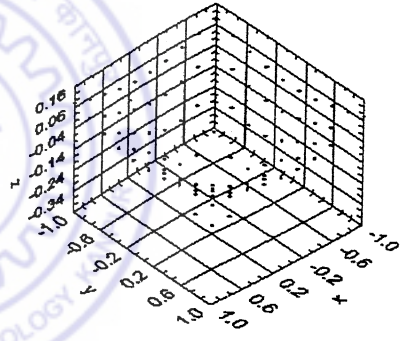


$$z = \ln(x^2 + y^2)$$

(n1) TrS14

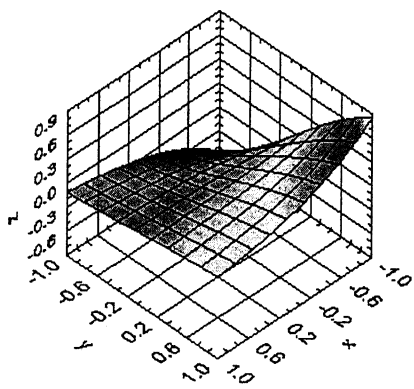


(n2) Point Cloud of TrS14

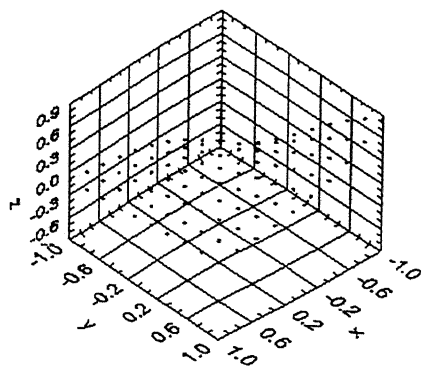


$$z = \exp(-x) \sin(y)$$

(o1) TrS15



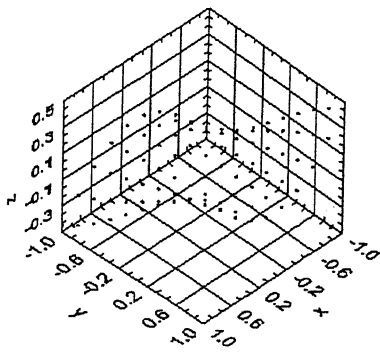
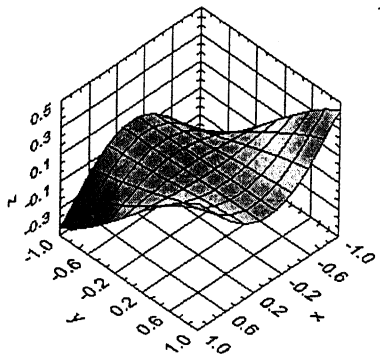
(o2) Point Cloud of TrS15



$$z = \sin(x^2) \sin(y)$$

(p1) TrS16

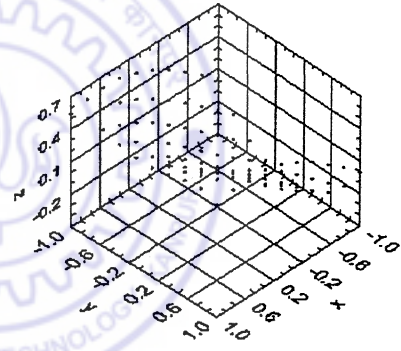
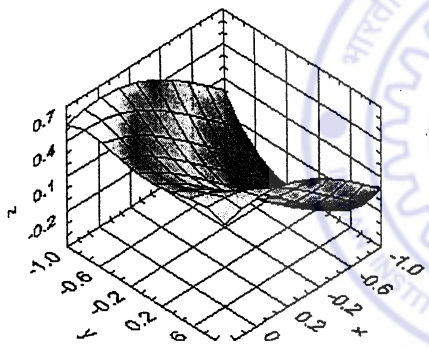
(p2) Point Cloud of TrS16



$$z = \sin(x + y^2)$$

(q1) TrS17

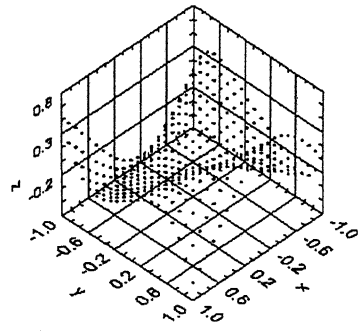
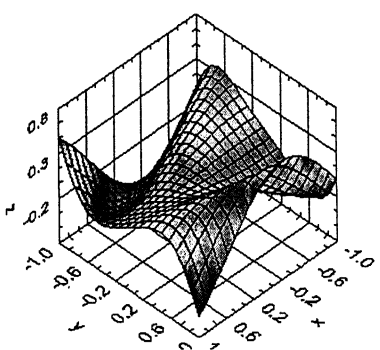
(q2) Point Cloud of TrS17



$$z = y \sin(2(x + y^2))$$

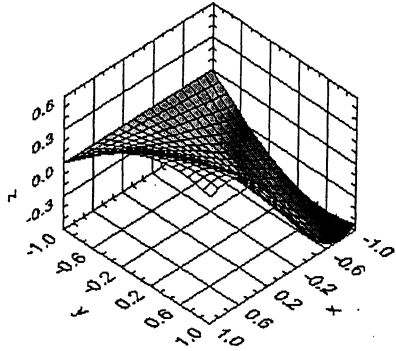
(r1) TrS18

(r2) Point Cloud of TrS18

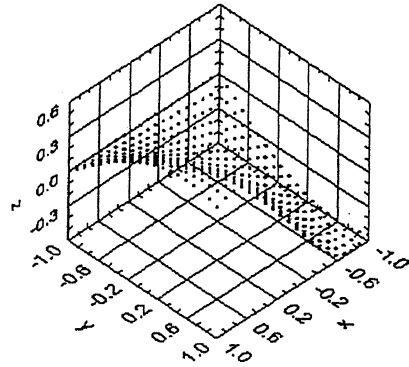


$$z = \sin(xy + x)$$

(s1) TrS19

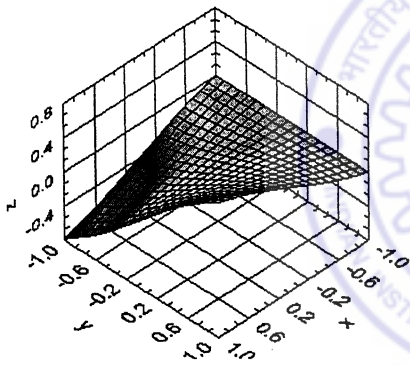


(s2) Point Cloud of TrS19

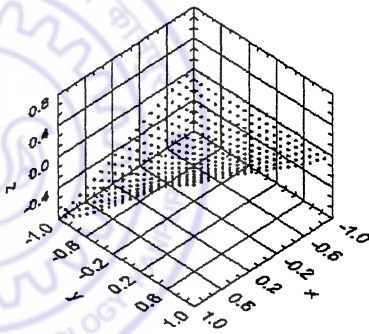


$$z = \sin(xy) + \sin(y)$$

(t1) TrS20

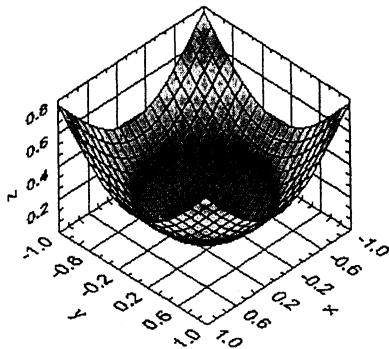


(t2) Point Cloud of TrS20

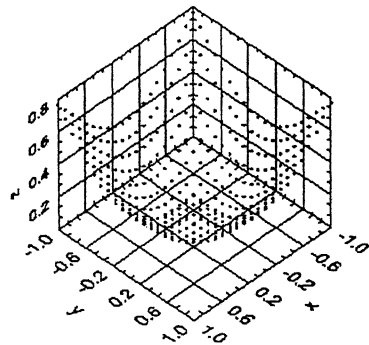


$$z = \sin(x^2) + \sin(y^2)$$

(u1) TrS21

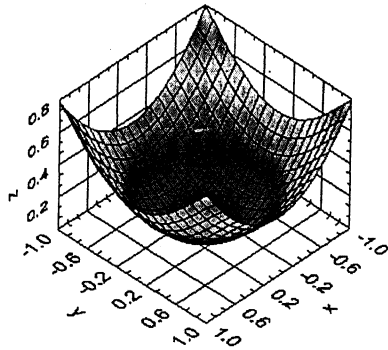


(u2) Point Cloud of TrS21

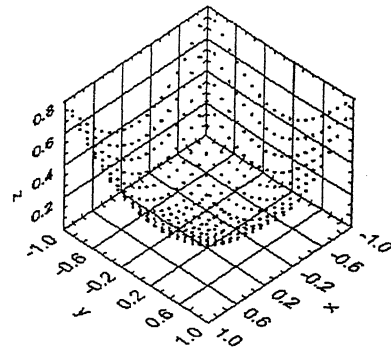


$$z = x^2 + \sin(y^2)$$

(v1) TrS22

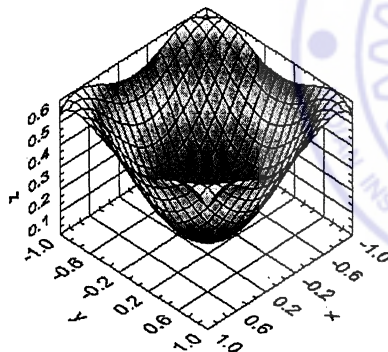


(v2) Point Cloud of TrS22

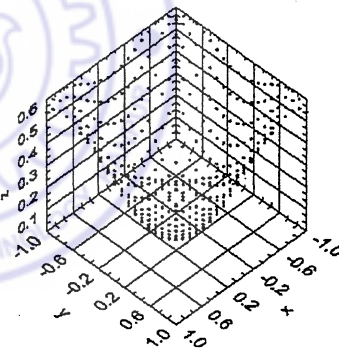


$$z = \sin(x^2 + y^2)$$

(w1) TrS23

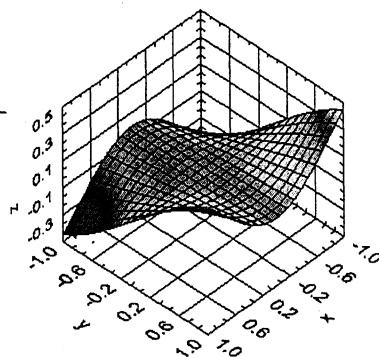


(w2) Point Cloud of TrS23

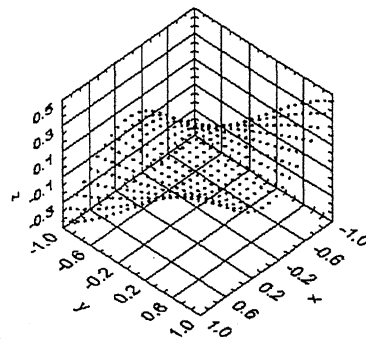


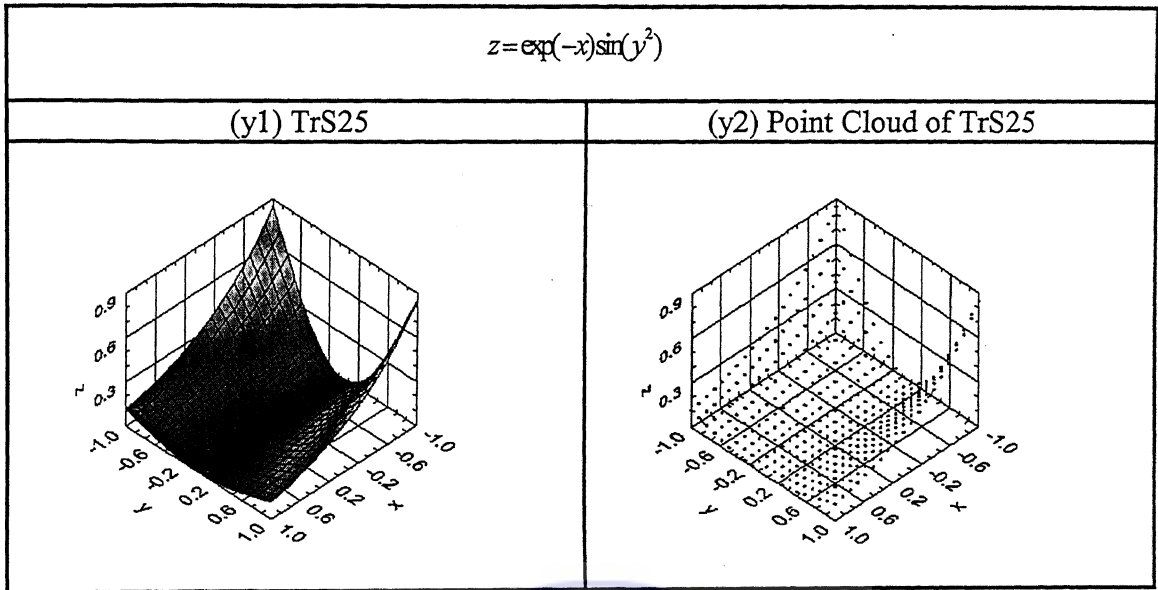
$$z = \sin(x) \sin(xy)$$

(x1) TrS24



(x2) Point Cloud of TrS24





neural network. A hundred points were chosen because a prior analysis showed that for the present classification, at least seventy points were required as otherwise the Neural Networks either required unusually long training runs or required high learning rates. On the other hand, two hundred were found to be on the excess as the weights got saturated for some initializations without epochs progressing much. It was concluded from this study that a minimum of seventy points was essential and a two hundred was on the high that was better avoiding. A break-even was adopted and a hundred points were selected for modeling the problem.

The plots of these input vectors (henceforth referred to as Surface Signatures) are given in Fig. 6.2. Elements '1' and '-1' were chosen as target elements for algebraic and transcendental surfaces respectively.

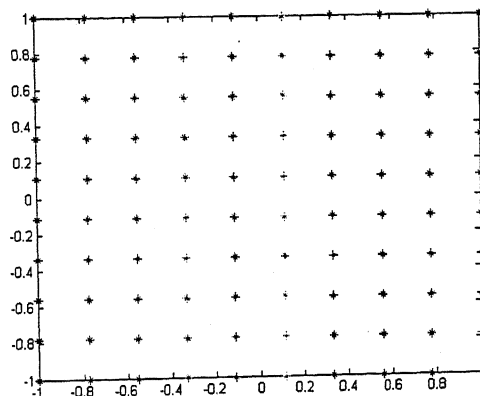


Fig. 6.1 Points on the xy -Plane where all the surfaces were sampled

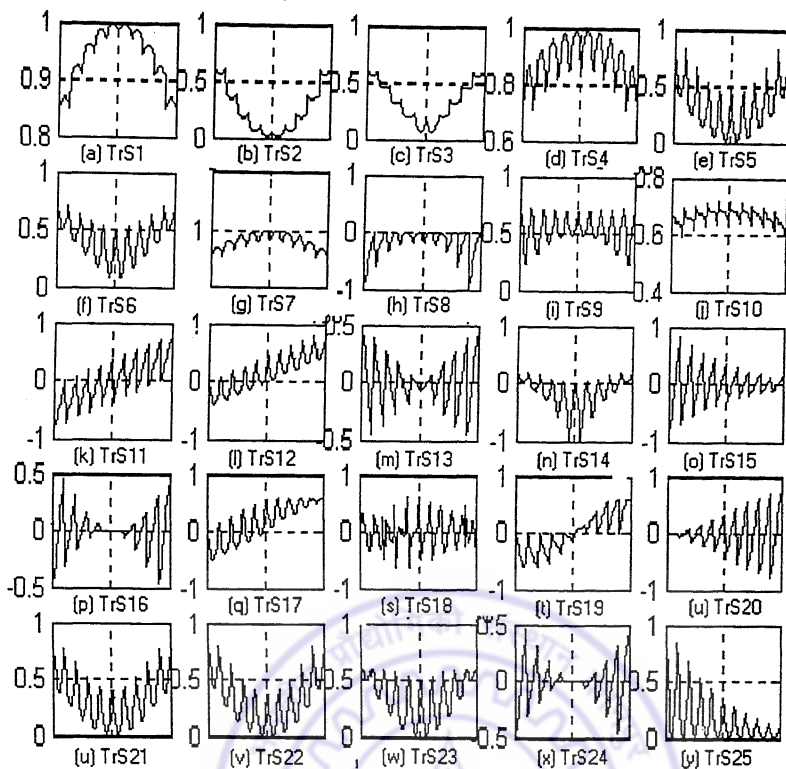


Fig. 6.2 Signatures of Training Surfaces

The BPA was invoked for four architectures (i) 100-5-1 (Sigmoidal Activation Function) (ii) 100-5-1 (Linear Activation Function), (iii) 100-5-5-1 (Sigmoidal Activation Function) (iv) 100-5-5-1 (Linear Activation Function). The architecture 100-5-1 was trained with the sigmoid activation using other BPAs discussed earlier. The initial weights were kept same for all algorithms. As expected, BPA with momentum was found to converge faster in comparison with the standard BPA (Qian, 1999). The additional inertia term arising from the equation of update of weights causes an accelerated convergence. The Resilient BPA is characterized by updates in steps. The error surface has a number of minima and maxima points but the exact number of these is specific to the architecture chosen for the problem. In a dynamic sequence of errors as the epochs progress, there is a greater likelihood, owing to the step update characteristic of the algorithm, of the error at a particular epoch missing the local minimum point and hitting a point in the vicinity of the same. This might result in over training of the network as the number of such hits depends on how the network was initialized and a concomitantly results in a reduction in the test performance (as some weights saturate in the process). The update rules for each of the scaled conjugate gradient (SCG) algorithms is identical. The difference lies in the computation of the Hessian in these update rules. The SCG algorithm requires the computation of a conjugate direction for weight update to avoid local minima for which the Hessian is approximated by an appropriate formula

(Moller, 1993). Lavenberg-Marquardt involves the computation of the Jacobian on which the weight update is based. As the value of the parameter γ changes, Lavenberg-Marquardt scheme keeps moving towards or away from the gradient descent scheme. This results in an accelerated or diminished rate of convergence.

6.4 Network Testing

Three sets of test surfaces were constructed. These sets are listed in Table 6.3. The first two sets were constructed from the surface functions of the training set itself. In Set 1, surfaces are constructed by making minor variations in the coefficients of the training surface equations. In the second set, these variations are relatively large. The surfaces in Set 1 have been denoted as Tst1S1, Tst1S2, etc. while those in Set 2 are denoted as Tst2S1, Tst2S2, etc. Unlike the first and second sets, surfaces in Set 3 are constructed through different functions (this set includes composite functions also). The surfaces in this set are denoted as Tst3S1, Tst3S2 etc.

All the four BPAs listed in the previous section were tested with the four architectures, to identify the first set of test surfaces. As earlier said, initial weights and target error have been kept identical while training networks with different algorithms. The results are given in Table 6.4. The 100-5-1 architecture with Sigmoidal Activation Function, is found to give the best performance. The identification is correct in more than ninety percent cases and Lavenberg-Marquardt scheme is the most accurate. The 100-5-1 architecture with Sigmoidal Activation Function, is used further for identification of Set 2 and Set 3 surfaces. The percentages of surfaces correctly identified through the four algorithms are given in Table 6.x. It can be observed that the identification is fairly good for the first two sets of test surfaces. It should be noted that low percentage of correct identification for Set 3 surfaces is due to the fact that the functions are those for which the networks have not been trained for and that it contains composite functions as well as functions involving product of more than two transcendental terms.

Table 6.3 Test Surfaces for the Surface Classification

| Test Set 1 | | Test Set 2 | | Test Set 3 | |
|------------|---|------------|---|------------|------------------------------------|
| Surface | Surface Equation | Surface | Surface Equation | Surface | Surface Equation |
| Tst1S1 | $x^2/4 + y^2/18 + z^2/64 = 1$ | Tst2S1 | $x^2/64 + y^2/128 + z^2/256 = 1$ | Tst3S1 | $z = x^3 + y^3$ |
| Tst1S2 | $z = 2x^2 + y^2/8$ | Tst2S2 | $z = x^2/4 + y^2/8$ | Tst3S2 | $z = x^2/9 + y^2/16$ |
| Tst1S3 | $2x^2 + y^2/16 - z^2 = 0$ | Tst2S3 | $z^2 = 2x^2 + y^2/25$ | Tst3S3 | $z^2 = x^2/4 + y^2/16$ |
| Tst1S4 | $x^2/2 + y^2/4 + z^2/4 = 1$ | Tst2S4 | $x^2/2 + y^2/64 + z^2/4 = 1$ | Tst3S4 | $z = xy$ |
| Tst1S5 | $z = 0.5(x^2 + y^2)$ | Tst2S5 | $z = x^2 + y^2/18$ | Tst3S5 | $z = x^2y^2$ |
| Tst1S6 | $x^2 + 2y^2 - z^2 = 0$ | Tst2S6 | $z^2 = x^2/9 + 2y^2$ | Tst3S6 | $z^2 = x^2 + y^2/9$ |
| Tst1S7 | $2x^2/9 + y^2/16 - z^2/25 = 1$ | Tst2S7 | $2x^2/9 + y^2/25 - z^2/25 = 1$ | Tst3S7 | $z = xy^2$ |
| Tst1S8 | $z = (2x^2 - y)(y/2 - x^2)$ | Tst2S8 | $z = (2x^2 - y)(4y - x^2)$ | Tst3S8 | $z = xy - x^2$ |
| Tst1S9 | $z^2 = \sqrt{((x-1)^2 + y^2)((x+1)^2 + y^2)}$ | Tst2S9 | $z^2 = \sqrt{((x-1)^2 + y^2/9)((x+1)^2 + y^2/4)}$ | Tst3S9 | $z = xy + x^2y^2$ |
| Tst1S10 | $x^2 + y^3 + z^5 = 5$ | Tst2S10 | $x^2/20 + y^2/135 + z^2/5 = 1$ | Tst3S10 | $z = x^2 + y^3$ |
| Tst1S11 | $z = x + y/2$ | Tst2S11 | $z = 2x + y/2$ | Tst3S11 | $z = 3x + y/2$ |
| Tst1S12 | $z = 2x + y^2$ | Tst2S12 | $z = 2x + y^2/4$ | Tst3S12 | $z = x + y^2 + xy$ |
| Tst1S13 | $z = \sin(x)\sin(2y)$ | Tst2S13 | $z = \sin(3x)\sin(2y)$ | Tst3S13 | $z = \sin(x)\sin(y)\sin(xy)$ |
| Tst1S14 | $z = \ln(x^2 + y^2/2)$ | Tst2S14 | $z = \ln(2x^2 + y^2/2)$ | Tst3S14 | $z = \ln(x^2 + y^2)\sin(x)$ |
| Tst1S15 | $z = e^{-x}\sin(y/2)$ | Tst2S15 | $z = e^{-3x}\sin(y/2)$ | Tst3S15 | $z = e^{-x}\sin(y)\sin(e^y)$ |
| Tst1S16 | $z = \sin(x^2)\sin(y/2)$ | Tst2S16 | $z = \sin(3x^2)\sin(y/2)$ | Tst3S16 | $z = \sin(x^2)\cos(y)$ |
| Tst1S17 | $z = \sin(2x + y^2)$ | Tst2S17 | $z = \sin(2x + y^2/4)$ | Tst3S17 | $z = \sin(x)e^{\sin(y)}$ |
| Tst1S18 | $z = y\sin(2(x/2 + y^2))$ | Tst2S18 | $z = y\sin(x + 6y^2)$ | Tst3S18 | $z = \sin((x/2 + y^2)\sin(x))$ |
| Tst1S19 | $z = \sin(xy + 2x)$ | Tst2S19 | $z = \sin(x^2 + xy/4)$ | Tst3S19 | $z = \sin(xy) + \cos(xy)$ |
| Tst1S20 | $z = \sin(xy) + \sin(2y)$ | Tst2S20 | $z = \sin(xy/4) + \sin(y^2)$ | Tst3S20 | $z = \sin(x)\cos(y) + \sin(y^2)$ |
| Tst1S21 | $z = \sin(x^2) + \sin(y^2/2)$ | Tst2S21 | $z = \sin(2x^2) + \sin(y^2/2)$ | Tst3S21 | $z = \sin(x^2) + \sin(y^2)\sin(y)$ |
| Tst1S22 | $z = x^2 + \sin(2y^2)$ | Tst2S22 | $z = x^2 + \sin(3y^2)$ | Tst3S22 | $z = \sin(x^2)\sin(y^2)$ |
| Tst1S23 | $z = \sin(x^2/2 + y^2)$ | Tst2S23 | $z = \sin(x^2/2 + y^2/3)$ | Tst3S23 | $z = \sin(x^2/2 + y^2)$ |
| Tst1S24 | $z = \sin(x/2)\sin(xy)$ | Tst2S24 | $z = \sin(x/2)\sin(xy/4)$ | Tst3S24 | $z = \sin(x/2)\sin(xy^2)$ |
| Tst1S25 | $z = e^{-x/2}\sin(y^2)$ | Tst2S25 | $z = e^{-x/2}\sin(y^2/4)$ | Tst3S25 | $z = e^{-y/3}\sin(x)$ |

6.5 Classification Errors and Surface Signatures

The input signature, as defined previously, is crucial to the training scheme. The network gets trained to identify surfaces on the basis of these input patterns. The signatures of all the testing surfaces considered for study are plotted in Figs. 6.4 – 6.6. These signatures are dependent on the arrangement of the various entries in the input matrix, *i.e.* a different arrangement would produce a different input signature and consequently trace out a different training course. The signatures

Table 6.4(a) Test performance of the 100-5-1 network on each test set ('1' indicates correct identification, '0' indicates incorrect identification)

| Test Set 1 | | | | | Test Set 2 | | | | | Test Set 3 | | | | |
|------------|-----|-----|-----|----|------------|-----|-----|-----|----|------------|---------|-----|-----|----|
| Surface | GDM | Res | SCG | LM | Surface | GDM | Res | SCG | LM | Surface | GD M | Res | SCG | LM |
| Tst1S1 | 1 | 1 | 1 | 1 | Tst2S1 | 1 | 1 | 1 | 1 | Tst3S1 | 1 | 1 | 1 | 1 |
| Tst1S2 | 1 | 1 | 1 | 1 | Tst2S2 | 0 | 1 | 0 | 1 | Tst3S2 | 0 | 1 | 0 | 1 |
| Tst1S3 | 1 | 1 | 1 | 1 | Tst2S3 | 1 | 1 | 1 | 1 | Tst3S3 | 1 | 1 | 1 | 1 |
| Tst1S4 | 1 | 1 | 1 | 1 | Tst2S4 | 1 | 1 | 1 | 1 | Tst3S4 | 0 | 0 | 0 | 0 |
| Tst1S5 | 0 | 1 | 0 | 1 | Tst2S5 | 1 | 1 | 1 | 1 | Tst3S5 | 0 | 0 | 0 | 0 |
| Tst1S6 | 1 | 1 | 1 | 1 | Tst2S6 | 0 | 1 | 0 | 0 | Tst3S6 | 1 | 1 | 1 | 1 |
| Tst1S7 | 1 | 1 | 1 | 1 | Tst2S7 | 1 | 1 | 1 | 1 | Tst3S7 | 1 | 1 | 1 | 1 |
| Tst1S8 | 1 | 1 | 1 | 1 | Tst2S8 | 1 | 1 | 1 | 1 | Tst3S8 | 0 | 1 | 1 | 1 |
| Tst1S9 | 1 | 1 | 1 | 1 | Tst2S9 | 1 | 1 | 1 | 1 | Tst3S9 | 0 | 0 | 0 | 0 |
| Tst1S10 | 1 | 1 | 1 | 1 | Tst2S10 | 1 | 1 | 1 | 1 | Tst3S10 | 0 | 1 | 1 | 0 |
| Tst1S11 | 1 | 1 | 1 | 1 | Tst2S11 | 1 | 1 | 1 | 1 | Tst3S11 | 1 | 1 | 1 | 1 |
| Tst1S12 | 1 | 1 | 1 | 1 | Tst2S12 | 1 | 1 | 1 | 1 | Tst3S12 | 0 | 0 | 0 | 0 |
| Tst1S13 | 1 | 1 | 1 | 1 | Tst2S13 | 1 | 1 | 1 | 1 | Tst3S13 | 1 | 1 | 1 | 1 |
| Tst1S14 | 1 | 1 | 1 | 1 | Tst2S14 | 1 | 1 | 1 | 1 | Tst3S14 | 0 | 1 | 0 | 0 |
| Tst1S15 | 1 | 1 | 1 | 1 | Tst2S15 | 1 | 1 | 1 | 1 | Tst3S15 | 1 | 1 | 1 | 1 |
| Tst1S16 | 1 | 1 | 1 | 1 | Tst2S16 | 1 | 1 | 1 | 1 | Tst3S16 | 0 | 0 | 0 | 0 |
| Tst1S17 | 1 | 1 | 1 | 1 | Tst2S17 | 1 | 0 | 0 | 0 | Tst3S17 | 1 | 1 | 0 | 1 |
| Tst1S18 | 1 | 1 | 1 | 1 | Tst2S18 | 1 | 1 | 1 | 0 | Tst3S18 | 0 | 0 | 0 | 0 |
| Tst1S19 | 1 | 1 | 1 | 1 | Tst2S19 | 1 | 0 | 0 | 0 | Tst3S19 | 0 | 0 | 0 | 0 |
| Tst1S20 | 1 | 1 | 1 | 1 | Tst2S20 | 1 | 1 | 1 | 1 | Tst3S20 | 1 | 0 | 1 | 1 |
| Tst1S21 | 1 | 0 | 0 | 1 | Tst2S21 | 0 | 1 | 1 | 0 | Tst3S21 | 1 | 0 | 0 | 1 |
| Tst1S22 | 1 | 1 | 1 | 1 | Tst2S22 | 1 | 1 | 1 | 1 | Tst3S22 | 1 | 1 | 1 | 1 |
| Tst1S23 | 1 | 1 | 1 | 0 | Tst2S23 | 1 | 0 | 1 | 0 | Tst3S23 | 1 | 1 | 1 | 1 |
| Tst1S24 | 1 | 1 | 1 | 1 | Tst2S24 | 1 | 1 | 1 | 1 | Tst3S24 | 1 | 1 | 1 | 1 |
| Tst1S25 | 1 | 1 | 1 | 1 | Tst2S25 | 1 | 1 | 1 | 1 | Tst3S25 | 0 | 0 | 0 | 0 |
| Percent | 96 | 96 | 92 | 96 | Percent | 88 | 88 | 84 | 76 | Percent | 52 | 60 | 52 | 60 |

Table 6.4 (b) Performance of various backpropagation algorithms with test surfaces of Set 1

| Percentage of Test Surfaces Correctly Identified by the Network | | | | | |
|---|---------|-----|-------|--------|----|
| Arch | Acti | GDM | Resil | Scaled | LM |
| 100-5-1 | linear | 84 | 80 | 76 | 76 |
| 100-5-1 | sigmoid | 92 | 92 | 92 | 96 |
| 100-5-5-1 | linear | 80 | 84 | 76 | 72 |
| 100-5-5-1 | sigmoid | 88 | 80 | 92 | 84 |

of Figs. 6.2, and 6.5-6.6 were constructed by placing one column below another in the original 10x10 data matrix. The first set test surfaces were constructed by introducing minor variation in the governing equation's parameters and their signatures bear similarity with those of the training surfaces. The overall trend of these signatures remains in close vicinity with those of the training surfaces. The second set test surfaces that were constructed with a greater variation in the parameter values shows a considerable deviation from the training surfaces in signatures. The third testing set has many new algebraic functions (Tst3S1, Tst3S4, Tst3S5, Tst3S7, Tst3S8,

Tst3S9, Tst3S10, Tst3S12). In particular, surface equations Tst3S1, Tst3S10 are cubic functions that were not a part of the training. Among the transcendental functions composite functions (Tst3S15, Tst3S17, Tst3S18) and functions involving product of more than two transcendental functions (Tst3S13, Tst3S15), which were not present for training have been included in Set 3. These new surfaces have completely different signatures.

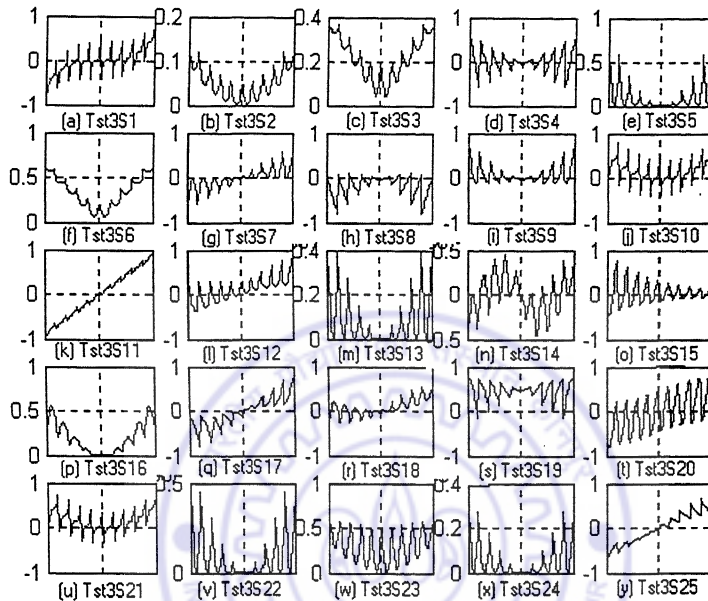


Fig. 6.3 Signatures of surfaces of Test Set 1

It is to be noted that the training set included algebraic surfaces up to the fifth order. Third order algebraic surfaces have deliberately not been included in the training, while surfaces of all other orders have been included (fourth, second and first). This was done in order to test if the network thus trained would be able to generalize and identify a third order surface. Order three algebraic surfaces have been included in Set 3 test functions (Tst3S1, Tst3S7 and Tst3S10). The test performance reveals that a purely third order algebraic surface Tst3S1 and Tst3S7 correctly got identified by all the networks while Tst3S10 was correctly classified by two of the four networks.

The transcendental surfaces in the training set included various forms of sine and exponential functions. Surfaces involving product and sum of algebraic and transcendental functions, e.g. TrS18 and TrS22, respectively, were also employed for training. It can be seen from Table 3 that test surfaces, Tst1S18, Tst1S22, Tst2S18 and Tst2S22 have been correctly identified by all the four algorithms which confirms that the network learnt to sense combination functions.

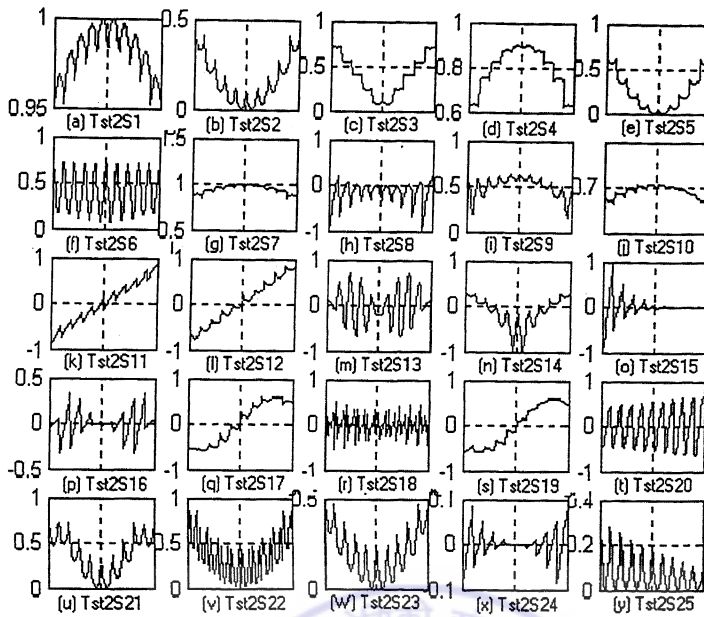


Fig. 6.4 Signatures of surfaces of Test Set 2

The training set signatures of surfaces TrS5, TrS21 and TrS22, exhibit similar patterns. While TrS5 is an algebraic surface (corresponding to a paraboloid of revolution), TrS21 and TrS22 are transcendental surfaces, with TrS22 involving summation of algebraic and transcendental functions. This similarity of signatures results in erroneous identification of test surfaces Tst1S5, Tst1S21, Tst2S21, Tst1S22 and Tst2S22. Errors are found in the identification of test surfaces Tst1S23, Tst2S23 due to similar reasons. The signature in Fig. 6.2(w), for training surface TrS23, is the sine of the paraboloid of revolution, which is similar to that of the paraboloid surface TrS5. The training surface TrS23 was the sine of a regular paraboloid while all three test surfaces (Fig. 6.3(w), Fig. 6.4(w), Fig. 6.5(w)), are sines of an elliptic paraboloid, of which the regular paraboloid is only a particular case, i.e. the test surfaces are more general than the training surface. It is evident that though networks have learnt to generalize to cases for which they have not been trained, caution needs to be exercised in making distinction in cases where training signatures have identical patterns.

Training surfaces TrS13, TrS24 (Figs. 6.2(m), (x)) involve product of two sine functions. It can be seen from Table 6.3, that testing surfaces Tst1S13, Tst2S13, Tst1S24, Tst2S24, which also similarly involve products of two sine functions, get correctly identified. In addition test surface Tst3S13 involving product of three sine functions also gets correctly identified,

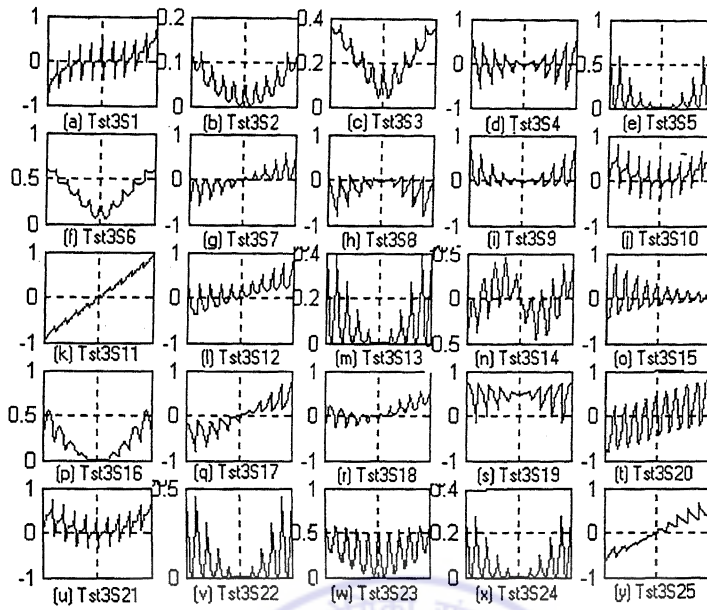


Fig. 6.5 Signatures of surfaces of Test Set 3

despite the network not being specifically trained to recognize such a surface. Here also, the networks have developed generalization capabilities.

Identification of test surfaces with trained neural network for the first set was best with success rate ranging from 92 to 96 percent. This is because the set was constructed with minor variation in the training set parameters. The second set of surfaces had a lesser success rate with correct identification percentage ranging between 76 and 88 percent. The third set with radically new testing functions had a success rate between 52 and 60 percent.

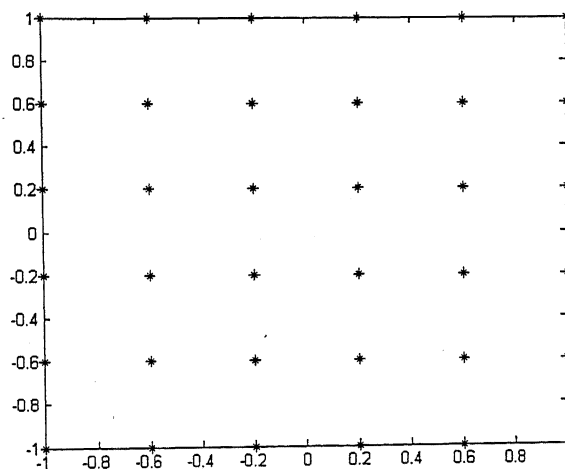


Fig. 6.6 Points for sampling the surfaces for order determination

6.6 Network Robustness

For testing neural network for algebraic-transcendental classification for robustness, random noise was introduced at all the points in the input set of the training surfaces matrix. Table 6.5 shows the comparative performances. The perturbed data set was input to the network after training it with an unperturbed training sata. Noise ranged from one percent to five percent. The robustness is on the decline for random noise in the five percent range while the net remained robust for a one-percent random perturbation. The robustness of the networks trained with resilient propagation, gradient descent with momentum and scaled conjugate gradient algorithm are also shown for the same noise range. Gradient descent with momentum gave the best performance.

6.7 Order Determination of Algebraic Surfaces

For surfaces classified as algebraic, order determination is the next step of the identification process. In the present study, classification up to third order algebraic polynomial has been considered. The most general first, second and third order polynomial surfaces, which can be expressed explicitly as $z = f(x, y)$, are

Table 6.5 Testing the Robustness of the Neural Network

| Gradient Descent with Momentum | | |
|--------------------------------|-----------------------------|----------------------------------|
| Percent Noise | Percent Algebraic Incorrect | Percent Transcendental Incorrect |
| 0.1 | 0 | 0 |
| 1 | 0 | 0 |
| 2 | 5.7 | 2.56 |
| 3 | 13.02 | 6.25 |
| 5 | 22.5 | 20.77 |
| Resilient Propagation | | |
| Percent Noise | Percent Algebraic Incorrect | Percent Transcendental Incorrect |
| 0.1 | 0 | 0 |
| 1 | 3.84 | 3.55 |
| 2 | 17.23 | 14.36 |
| 3 | 30.27 | 25.79 |
| 5 | 29.86 | 35.87 |
| Scaled Conjugate Gradient | | |
| Percent Noise | Percent Algebraic Incorrect | Percent Transcendental Incorrect |
| 0.1 | 0 | 0 |
| 1 | 0 | 0 |
| 2 | 10.42 | 7.05 |
| 3 | 23.33 | 13.85 |
| 5 | 30.83 | 29.23 |

| Lavenberg-Marquardt Algorithm | | |
|-------------------------------|-----------------------------|----------------------------------|
| Percent Noise | Percent Algebraic Incorrect | Percent Transcendental Incorrect |
| 0.1 | 0 | 0 |
| 1 | 0.6 | 0 |
| 2 | 16 | 9.23 |
| 3 | 20.83 | 15.38 |
| 5 | 30.8 | 38.46 |

First order surface : $ax + by + cz + d = 0$

Second order surface : $z = ax^2 + by^2 + cxy + dx + ey + f$

Third order surface : $z = ax^3 + by^3 + cx^2y + dxy^2 + ex^2 + fy^2 + gxy + hx + jy + k$

The most general first order surface is a plane. A quadratic surface with six coefficients is the most general second order explicit surface while a cubic surface with ten coefficients is the most general third order explicit surface.

For order classification, 216 planes, 216 quadratic and 216 cubic surfaces were considered. The equation of the plane can be put in the intercept form

$$x/A + y/B + z/C = 1$$

(with $A = -d/a$; $B = -d/b$; $C = -d/c$ A, B, C are the x, y and z intercepts respectively).

Six equi-spaced points were considered on the x, y and z axes to define 216 planes (6x6x6). Each of the 216 planes was sampled at the thirty- six points on the xy -plane shown in Fig. 6.6. The partial derivative with respect to y is computed from the 6x6 matrix. The second partial derivative with respect to y is also computed using the 5x6 matrix giving a 4x6 matrix. The three matrices are put in a single column vector of size 90x1 (36+30+24). For example, with -0.8311, 0.3237, 0.0330, -0.5747, -0.0300, -0.1774, -0.5824, -0.5097, -0.9208, 0.7707 as coefficients the z - coordinates at each of the thirty-six points shown in Fig. 9, for the cubic formula (16c) can be arranged in a 6x6 matrix as

$$[z] = \begin{bmatrix} 2.4604 & 1.5025 & 1.1977 & 1.2267 & 1.2704 & 1.0098 \\ 2.4836 & 1.5712 & 1.2833 & 1.1657 & 1.0342 & 0.5248 \\ 2.2530 & 1.3704 & 1.0366 & 0.8636 & 0.5611 & -0.1928 \\ 1.8930 & 1.0843 & 0.7079 & 0.4446 & -0.0245 & -1.0188 \\ 1.5278 & 0.7772 & 0.3854 & 0.0332 & -0.5985 & -1.8288 \\ 1.2817 & 0.5935 & 0.1905 & -0.2465 & -1.0365 & -2.4987 \end{bmatrix}$$

The partial derivative with respect to y provides the following slope matrix (Halmos [29], Hoffman [30]) of the order 5×6

$$\partial[z]/\partial y = \begin{pmatrix} 0.0232 & 0.0687 & 0.0406 & -0.0610 & -0.2362 & -0.4850 \\ -0.2306 & -0.1809 & -0.2047 & -0.3021 & -0.4731 & -0.7176 \\ -0.3600 & -0.3061 & -0.3257 & -0.4189 & -0.5857 & -0.8260 \\ -0.3652 & -0.3071 & -0.3225 & -0.4114 & -0.5740 & -0.8101 \\ -0.2461 & -0.1837 & -0.1949 & -0.2737 & -0.4380 & -0.6998 \end{pmatrix}$$

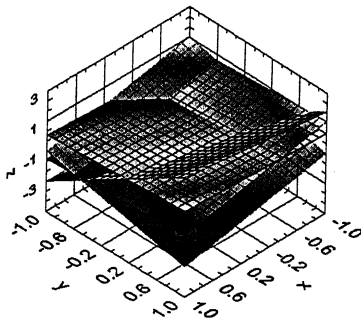
The second derivative matrix turns out to be

$$\partial^2[z]/\partial y^2 = \begin{pmatrix} -0.2538 & -0.2495 & -0.2453 & -0.2411 & -0.2369 & -0.2327 \\ -0.1295 & -0.1253 & -0.1210 & -0.1168 & -0.1126 & -0.1084 \\ -0.0052 & -0.0010 & 0.0033 & 0.0075 & 0.0117 & 0.0159 \\ 0.1191 & 0.1233 & 0.1276 & 0.1318 & 0.1360 & 0.1402 \end{pmatrix}$$

The three matrices can be arranged into a single 90×1 vector by sequentially arranging the columns of $[z]$ one below the other, followed by the columns of $\partial[z]/\partial y$ and $\partial^2[z]/\partial y^2$. All the data sets (for planes, quadratic surfaces and cubic surfaces) were modeled in the form given in the example. The coefficients of the quadratic as well as cubic surfaces were randomly generated in the interval $[-1,1]$. Each of the three kinds of surfaces was sampled at the thirty-six points shown in Fig. 6.6 and the procedure shown in the example above was applied for constructing the data matrices. The 90×1 vectors thus constructed for each of the 216 planes, 216 quadratic surfaces and 216 cubic surfaces were arranged into a 90×648 matrix. Fig. 6.7 shows four each of the planes, quadratic and cubic surfaces that formed a part of the input. Fig. 6.8 shows typical signatures of the input columns corresponding to plane, quadratic and the cubic. The target corresponding to each of the planes was chosen as '-1', each of the quadratic surfaces had zero and each of the cubic surfaces had a '1' for target. The target vector hence, was of size 1×648 . For the data set thus constructed, a neural network of architecture 90-10-1 was trained. The input matrix was normalized to -0.9 and 0.9 . A bipolar sigmoidal activation function was used for all the layers of the architecture. The network thus trained was tested with data sets constructed following the procedure illustrated in the example above for each of the sixty-four planes, quadratic and cubic surfaces.

Among the other approaches to the same problem, the ANN and CNN based classification schemes based of different Error Functions were employed. The important idea is to differentiate and evaluate the performance of the ANN and CNN based networks. The Surface Classification

was also solved by Error Function based Neural Networks. The Error Functions listed in the Chapter 3 earlier and the algorithms developed over them have been used to solve the classification problem.



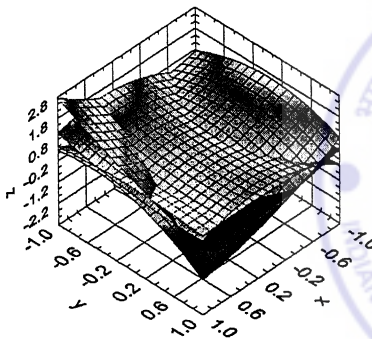
(a) Planes

$$(i) \frac{x}{-1} + \frac{y}{-1} + \frac{z}{-1} = 1$$

$$(ii) \frac{x}{-1} + \frac{y}{-0.6} + \frac{z}{0.2} = 1$$

$$(iii) \frac{x}{1} + \frac{y}{0.2} + \frac{z}{-0.6} = 1$$

$$(iv) \frac{x}{-0.2} + \frac{y}{0.6} + \frac{z}{-0.2} = 1$$



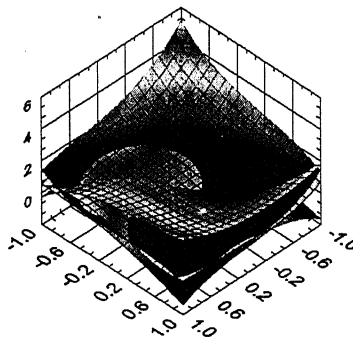
(b) Quadratic Surfaces

$$(i) z = 0.3328x^2 - 0.827y^2 - 0.4889xy - 0.7145x - 0.4454y + 0.5529$$

$$(ii) z = 0.4160x^2 + 0.2058y^2 - 0.9212xy + 0.6914x - 0.5405y - 0.0099$$

$$(iii) z = -0.9399x^2 + 0.2959y^2 - 0.1991xy + 0.535x - 0.2323y + 0.7828$$

$$(iv) z = -0.1844x^2 - 0.2463y^2 - 0.8231xy + 0.9652x - 0.8434y - 0.0762$$



(c) Cubic Surfaces

$$(i) z = -0.737x^3 - 0.858y^3 - 0.0767x^2y - 0.102xy^2 + 0.8445x^2 + 0.3939y^2 + 0.5376xy + 0.4702x - 0.3797y + 0.7226$$

$$(ii) z = -0.0542x^3 + 0.769y^3 + 0.4673x^2y - 0.8268xy^2 + 0.4213x^2 + 0.036y^2 + 0.446xy + 0.7864x - 0.2073y + 0.7407$$

$$(iii) z = -0.6656x^3 + 0.0055y^3 + 0.1609x^2y - 0.1363xy^2 - 0.1654x^2 - 0.3797y^2 + 0.2587xy + 0.0157x - 0.8847y + 0.8850$$

$$(iv) z = 0.5239x^3 + 0.996y^3 - 0.4363x^2y + 0.3752xy^2 - 0.2532x^2 - 0.7474y^2 + 0.1553xy + 0.1743x - 0.8822y + 0.5717$$

Fig. 6.7 Planes, Quadratic and Cubic Surfaces for Order Determination

6.8 BPA based on different Error Functions

The same twenty-five surfaces, shown in Table 6.2 (a1)-(y1), were considered for training the ANN by altering the Error Functions. The aim here is to see how well the ANN with Error Function altered learns to perform the classification and generalize. The next section extends the

domain further by bringing CNN's in, in which once again, the Error Functions have been varied and CVBP's built over them.

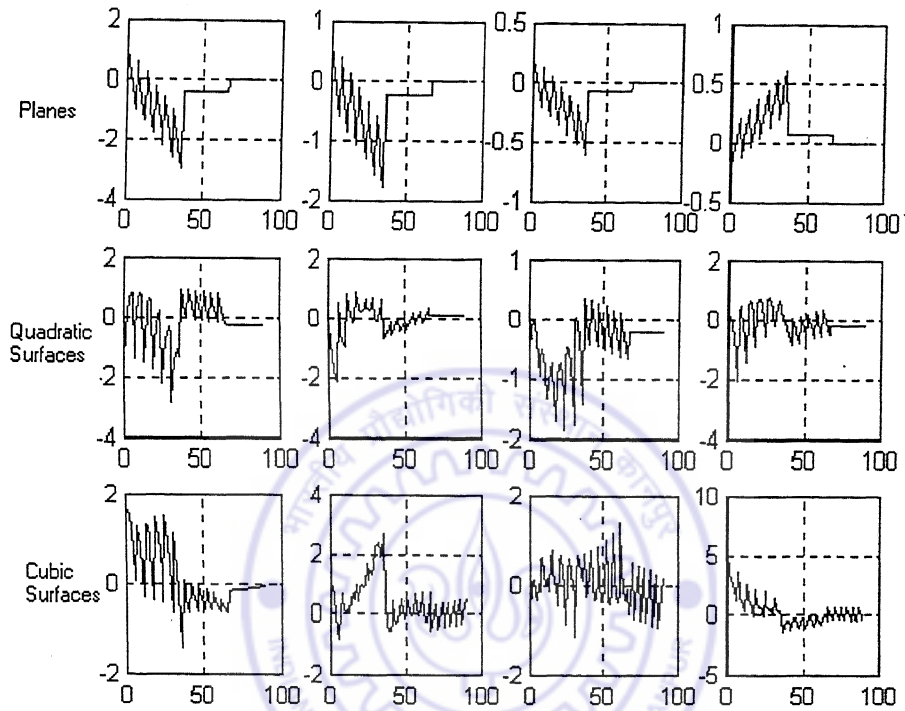


Fig. 6.8 Typical signatures of Planes, Quadratic surfaces and Cubic surfaces for order determination

The data set chosen was the same. The output chosen was '1' for algebraic surfaces and '-1' for transcendental surfaces. Two architectures, 100-5-1 and 100-10-1 were chosen for the experiment. These were selected to maintain the uniformity in the whole experiment as in the earlier section these architectures was chosen.

The following parameters were set for the training and testing procedure. Two different architectures of sizes 100-5-1 and 100-10-1 were chosen. These were selected because the Back-Propagation Algorithm with the standard Quadratic Error Function produced good results with these. Learning rate for updating the weights was set at 0.1, bias was enabled for both hidden and output layers with a constant input of -1 . The initial weight matrices for both architectures were kept fixed for all runs of the Back-Propagation Algorithm with different error functions.

Table 6.6 Classification based on Error Function based BPA. Network was trained for 10000 epochs

| Result with 100-5-1 Architecture | | | | | | |
|-----------------------------------|-----------------------------------|-------|-------|---------------------------------------|-------|-------|
| Error Function | Algebraic (% Correct identified) | | | Transcendental (% Correct identified) | | |
| | Test1 | Test2 | Test3 | Test1 | Test2 | Test3 |
| Absolute Error | 83.34 | 83.34 | 25 | 84.62 | 58.33 | 16.67 |
| Andrew Error | 100 | 100 | 100 | 0 | 0 | 0 |
| Cauchy Error | 83.33 | 83.33 | 41.67 | 92.3 | 58.33 | 58.33 |
| Error Fourth Order | 50 | 5 | 0 | 100 | 100 | 100 |
| Fair Error | 91.67 | 83.33 | 50 | 84.62 | 53.85 | 46.15 |
| Geman-McClure Error | 91.67 | 58.33 | 33.33 | 92.31 | 69.23 | 46.15 |
| Huber Error | 91.67 | 83.33 | 33.33 | 84.62 | 61.54 | 58.33 |
| Hyperbolic Squared | 83.33 | 58.33 | 33.33 | 58.33 | 41.67 | 16.67 |
| Bipolar Hyperbolic | 66.67 | 50 | 41.67 | 92.31 | 58.33 | 16.67 |
| LogCosh Error | 83.33 | 83.33 | 50 | 92.3 | 46.15 | 69.23 |
| Logarithmic Error | 58.33 | 58.33 | 0 | 100 | 100 | 92.3 |
| Mean Median Error | 58.33 | 58.33 | 8.33 | 100 | 84.62 | 84.62 |
| Minkowski Error | 91.67 | 75 | 33.34 | 100 | 76.92 | 69.23 |
| Quadratic Error | 91.67 | 75 | 41.67 | 84.62 | 69.23 | 46.15 |
| Sinh Error | 50 | 50 | 0 | 100 | 92.3 | 92.3 |
| Tukey Error | 66.67 | 75 | 25 | 100 | 84.62 | 69.23 |
| Welsch Error | 83.33 | 75 | 33.34 | 92.31 | 69.23 | 53.85 |
| Result with 100-10-1 Architecture | | | | | | |
| Error Function | Algebraic (% Correct identified) | | | Transcendental (% Correct identified) | | |
| | Test1 | Test2 | Test3 | Test1 | Test2 | Test3 |
| Absolute Error | 83.33 | 75 | 33.34 | 76.92 | 38.46 | 46.15 |
| Andrew Error | 100 | 100 | 100 | 0 | 0 | 0 |
| Cauchy Error | 91.67 | 83.33 | 50 | 92.31 | 53.85 | 53.85 |
| Error Fourth Order | 41.67 | 41.67 | 0 | 100 | 100 | 92.31 |
| Fair Error | 0 | 0 | 0 | 100 | 100 | 100 |
| Geman-McClure Error | 83.33 | 41.67 | 50 | 92.31 | 46.15 | 16.67 |
| Huber Error | 91.67 | 83.33 | 33.33 | 92.31 | 76.92 | 46.15 |
| Hyperbolic Squared | 58.33 | 41.67 | 25 | 84.62 | 53.85 | 46.15 |
| Bipolar Hyperbolic | 66.67 | 58.33 | 16.67 | 76.92 | 38.46 | 46.15 |
| LogCosh Error | 91.67 | 83.33 | 41.67 | 92.31 | 53.85 | 61.54 |
| Logarithmic Error | 58.33 | 58.33 | 8.33 | 76.92 | 84.62 | 84.62 |
| Mean Median Error | 91.67 | 83.33 | 50 | 84.62 | 53.85 | 46.15 |
| Minkowski Error | 83.33 | 83.33 | 41.67 | 84.62 | 38.46 | 61.54 |
| Quadratic Error | 91.67 | 83.33 | 58.33 | 100 | 46.15 | 46.15 |
| Sinh Error | 33.34 | 41.67 | 0 | 100 | 100 | 92.31 |
| Tukey Error | 83.33 | 83.33 | 58.33 | 84.62 | 38.46 | 46.15 |
| Welsch Error | 100 | 100 | 100 | 0 | 0 | 0 |

The target error was set at 0.00001. The maximum epochs were set to 10000. The activation functions for both hidden and output layers were bipolar sigmoid. Caution was observed while implementing Hyperbolic Sigmoid and Bipolar Hyperbolic Sigmoid Errors for which error functions were accordingly set with Hyperbolic Sigmoid having unipolar activation and Bipolar Sigmoid having a bipolar activation function. The training parameters set show that an epochs criterion was preferred by assigning a larger target error than the network could achieve (that was arrived at in a sequence of trial runs before conducting the experiment.)

6.8.1 Classification Result

The performance of BPA across various EF's is displayed in Table 6.6. It is evident from the results that a proper choice of EF can aid the training scheme by speeding the learning process or bypassing a local minima as the case may be. The initial weights were kept identical for both architectures during the whole experiment.

6.9 CVBP based of different Error Functions

The complex Error Function based CVBP is applied to the Surface Classification problem in the present section.

6.9.1 The Complex-Variable Based Algorithm

The architectures 100-5-1, 100-10-1 were trained with the Nitta activation using the CVBPs based on the error functions described. The initial weights were kept same for all the training algorithms, the learning rate set was at 0.1.

Table 6.7 Test performance of complex network on each test set. a1 is 100-5-1
a2 is 100-10-1, a3 is 100-15-1, a4 is 100-20-1.

('1' indicates correct identification. '0' indicates incorrect identification)

| Test Set 1 | | | | | Test Set 2 | | | | | Test Set 3 | | | | |
|------------|----|----|----|----|------------|----|----|----|----|------------|----|----|----|----|
| Surface | a1 | a2 | a3 | a4 | Surface | a1 | a2 | a3 | a4 | Surface | a1 | a2 | a3 | a4 |
| Tst1S1 | 1 | 1 | 1 | 1 | Tst2S1 | 1 | 1 | 1 | 1 | Tst3S1 | 1 | 1 | 1 | 1 |
| Tst1S2 | 1 | 1 | 1 | 1 | Tst2S2 | 0 | 1 | 0 | 1 | Tst3S2 | 0 | 1 | 0 | 1 |
| Tst1S3 | 1 | 1 | 1 | 1 | Tst2S3 | 1 | 1 | 1 | 1 | Tst3S3 | 1 | 1 | 1 | 1 |
| Tst1S4 | 1 | 1 | 1 | 1 | Tst2S4 | 1 | 1 | 1 | 1 | Tst3S4 | 0 | 0 | 0 | 0 |
| Tst1S5 | 0 | 0 | 0 | 0 | Tst2S5 | 1 | 1 | 1 | 1 | Tst3S5 | 0 | 0 | 0 | 1 |
| Tst1S6 | 1 | 1 | 1 | 1 | Tst2S6 | 0 | 0 | 0 | 0 | Tst3S6 | 1 | 1 | 1 | 1 |
| Tst1S7 | 1 | 1 | 1 | 1 | Tst2S7 | 1 | 1 | 1 | 1 | Tst3S7 | 1 | 1 | 1 | 1 |
| Tst1S8 | 0 | 1 | 0 | 1 | Tst2S8 | 0 | 1 | 0 | 1 | Tst3S8 | 1 | 0 | 0 | 0 |
| Tst1S9 | 1 | 1 | 1 | 1 | Tst2S9 | 1 | 1 | 1 | 1 | Tst3S9 | 0 | 0 | 0 | 1 |
| Tst1S10 | 1 | 1 | 1 | 1 | Tst2S10 | 1 | 1 | 1 | 1 | Tst3S10 | 0 | 1 | 0 | 1 |
| Tst1S11 | 1 | 1 | 1 | 1 | Tst2S11 | 1 | 0 | 1 | 1 | Tst3S11 | 1 | 0 | 1 | 0 |
| Tst1S12 | 1 | 0 | 1 | 1 | Tst2S12 | 1 | 0 | 1 | 1 | Tst3S12 | 0 | 0 | 0 | 0 |
| Tst1S13 | 1 | 1 | 1 | 1 | Tst2S13 | 1 | 1 | 1 | 1 | Tst3S13 | 1 | 1 | 0 | 1 |
| Tst1S14 | 1 | 1 | 1 | 1 | Tst2S14 | 1 | 1 | 1 | 1 | Tst3S14 | 0 | 0 | 1 | 0 |
| Tst1S15 | 1 | 1 | 1 | 1 | Tst2S15 | 1 | 1 | 1 | 0 | Tst3S15 | 1 | 1 | 0 | 1 |
| Tst1S16 | 1 | 1 | 1 | 1 | Tst2S16 | 1 | 1 | 1 | 1 | Tst3S16 | 0 | 0 | 1 | 0 |
| Tst1S17 | 1 | 1 | 1 | 0 | Tst2S17 | 0 | 1 | 0 | 0 | Tst3S17 | 0 | 1 | 1 | 0 |
| Tst1S18 | 1 | 1 | 1 | 1 | Tst2S18 | 1 | 0 | 1 | 0 | Tst3S18 | 0 | 0 | 0 | 0 |
| Tst1S19 | 0 | 1 | 1 | 0 | Tst2S19 | 0 | 1 | 0 | 0 | Tst3S19 | 0 | 0 | 0 | 0 |
| Tst1S20 | 1 | 1 | 1 | 1 | Tst2S20 | 1 | 1 | 1 | 1 | Tst3S20 | 1 | 0 | 1 | 1 |
| Tst1S21 | 0 | 1 | 1 | 0 | Tst2S21 | 0 | 0 | 0 | 0 | Tst3S21 | 1 | 0 | 1 | 1 |
| Tst1S22 | 1 | 1 | 1 | 1 | Tst2S22 | 1 | 0 | 1 | 0 | Tst3S22 | 1 | 1 | 1 | 1 |
| Tst1S23 | 1 | 1 | 1 | 1 | Tst2S23 | 1 | 0 | 1 | 1 | Tst3S23 | 1 | 1 | 1 | 1 |
| Tst1S24 | 1 | 1 | 1 | 1 | Tst2S24 | 1 | 0 | 1 | 0 | Tst3S24 | 1 | 1 | 1 | 0 |
| Tst1S25 | 1 | 1 | 1 | 1 | Tst2S25 | 1 | 1 | 1 | 1 | Tst3S25 | 0 | 1 | 0 | 0 |
| Percent | 84 | 92 | 92 | 84 | Percent | 76 | 68 | 76 | 72 | Percent | 52 | 52 | 52 | 56 |

Table 6.8 Classification based on Error Function based CVBP (Nitta activation).
Training was performed for 10000 epochs.

| Result with 100-5-1 Architecture | | | | | | |
|-----------------------------------|-----------------------------------|-------|-------|---------------------------------------|-------|-------|
| | Algebraic (% Correct identified) | | | Transcendental (% Correct identified) | | |
| | Test1 | Test2 | Test3 | Test1 | Test2 | Test3 |
| Absolute Error | 58.33 | 58.34 | 16.67 | 92.31 | 69.23 | 84.62 |
| Andrew Error | 0 | 0 | 0 | 100 | 100 | 100 |
| Cauchy Error | 91.67 | 100 | 41.67 | 100 | 76.92 | 46.15 |
| Error Fourth Order | 100 | 91.67 | 66.67 | 92.31 | 46.15 | 38.46 |
| Fair Error | 91.67 | 91.67 | 41.67 | 100 | 69.23 | 46.15 |
| Geman-McClure Error | 100 | 100 | 100 | 0 | 0 | 0 |
| Huber Error | 91.67 | 83.34 | 41.67 | 100 | 69.23 | 46.15 |
| Hyperbolic Squared | 91.67 | 83.34 | 41.67 | 100 | 69.23 | 53.85 |
| Bipolar Hyperbolic | 91.67 | 91.67 | 41.67 | 100 | 69.23 | 53.85 |
| LogCosh Error | 66.67 | 58.34 | 16.67 | 100 | 92.31 | 84.62 |
| Logarithmic Error | 0 | 0 | 0 | 100 | 100 | 100 |
| Mean Median Error | 91.67 | 91.67 | 41.67 | 100 | 76.92 | 53.85 |
| Minkowski Error | 83.34 | 66.67 | 50 | 92.31 | 53.85 | 23.08 |
| Quadratic Error | 91.67 | 83.34 | 41.67 | 100 | 69.23 | 46.15 |
| Sinh Error | 91.67 | 83.34 | 41.67 | 100 | 69.23 | 46.15 |
| Tukey Error | 83.34 | 58.34 | 33.34 | 92.31 | 69.23 | 61.54 |
| Welsch Error | 58.34 | 58.34 | 25 | 100 | 76.92 | 61.54 |
| Result with 100-10-1 Architecture | | | | | | |
| | Algebraic (% Correct identified) | | | Transcendental (% Correct identified) | | |
| | Test1 | Test2 | Test3 | Test1 | Test2 | Test3 |
| Absolute Error | 50 | 50 | 16.67 | 92.31 | 69.23 | 69.24 |
| Andrew Error | 91.67 | 100 | 100 | 61.54 | 84.62 | 7.69 |
| Cauchy Error | 91.67 | 83.34 | 41.67 | 92.31 | 76.92 | 53.85 |
| Error Fourth Order | 83.34 | 83.34 | 50 | 76.92 | 69.21 | 53.85 |
| Fair Error | 83.34 | 91.67 | 50 | 92.31 | 76.92 | 38.46 |
| Geman-McClure Error | 75 | 58.34 | 33.34 | 84.62 | 76.92 | 53.85 |
| Huber Error | 83.34 | 91.67 | 50 | 92.31 | 76.92 | 38.46 |
| Hyperbolic Squared | 75 | 58.33 | 41.67 | 92.31 | 69.23 | 53.85 |
| Bipolar Hyperbolic | 75 | 58.34 | 33.34 | 100 | 69.21 | 69.23 |
| LogCosh Error | 83.34 | 83.34 | 33.34 | 61.54 | 61.54 | 15.38 |
| Logarithmic Error | 0 | 0 | 0 | 100 | 100 | 100 |
| Mean Median Error | 91.67 | 91.67 | 41.67 | 92.31 | 61.54 | 69.23 |
| Minkowski Error | 66.67 | 66.67 | 25 | 100 | 76.92 | 53.85 |
| Quadratic Error | 66.67 | 66.67 | 25 | 100 | 84.62 | 53.85 |
| Sinh Error | 83.34 | 75 | 41.67 | 92.31 | 69.23 | 53.85 |
| Tukey Error | 83.34 | 75 | 33.34 | 92.31 | 69.23 | 53.85 |
| Welsch Error | 66.67 | 66.67 | 8.34 | 100 | 69.23 | 61.53 |

A large target error was purposely set so that the problem can be modeled in terms of an epochs criterion. All the update rules listed in the Appendix 1 have been used to train the CNN's. The results are displayed in Table 6.8 (Nitta Activation) and Table 6.9 (new Activation). It can be seen that the New Activation based CNN's performed on par with Nitta's or better with the third set of test surfaces.

6.10 Some Remarks

A few observations are in order. First, it must be noted that the architectures of the CNN are also chosen to be 100-5-1 and 100-10-1. In fact the number of variables a complex number can accommodate are double that an ANN can take (as each complex number is dimension two with respect to the set of Real numbers). But as the aim of the present work was to compare the CNN and the ANN maintaining the parameters constant, hence the architectures have been chosen identical. If by some scheme, the size of the CNN were reduced (by coupling some inputs for example) the geometry of the problem gets distorted and the complex input vector will not mean what it actually means in the construction. In fact, if the scheme in question had reduced the architecture of the CNN to a 50-5-1 network, the space-complexity of the CNN would still be higher than the ANN as the number of variables would be 522 in number (weights and biases = $2(250+5+5+1)=522$, assuming the architecture is 50-5-1) compared with 511 ($500+5+5+1=511$) in the case of ANN (with architecture 100-5-1). The input neurons to the CNN in the present problem have purely real inputs and the imaginary parts have no signal in them. This is justified because the CNN can take real inputs as a particular case of complex inputs. The stand taken hence is fully justified. Second, there is no criterion by which one can ascertain if a certain problem gets better solved with the CVBP than with the standard BPA (Nitta, 1991). The fact

Table 6.9 Classification based on Error Function based CVBP (New activation).
Training was performed for 10000 epochs.

| Result with 100-5-1 Architecture | | | | | | |
|-----------------------------------|----------------------------------|-------|-------|---------------------------------------|-------|-------|
| | Algebraic (% Correct identified) | | | Transcendental (% Correct identified) | | |
| | Test1 | Test2 | Test3 | Test1 | Test2 | Test3 |
| Absolute Error | 58.33 | 46.15 | 25 | 84.62 | 61.54 | 69.23 |
| Andrew Error | 25 | 25 | 8.33 | 76.92 | 53.85 | 38.46 |
| Cauchy Error | 91.67 | 91.67 | 50 | 92.31 | 76.92 | 53.85 |
| Error Fourth Order | 91.67 | 75 | 58.33 | 92.31 | 46.15 | 46.15 |
| Fair Error | 83.33 | 58.33 | 50 | 76.92 | 69.23 | 53.85 |
| Geman-McClure Error | 83.33 | 75 | 58.33 | 61.54 | 46.15 | 30.77 |
| Huber Error | 91.67 | 75 | 41.67 | 84.62 | 61.54 | 53.85 |
| Hyperbolic Squared | 75 | 66.67 | 33.34 | 92.31 | 61.54 | 46.15 |
| Bipolar Hyperbolic | 83.33 | 91.67 | 50 | 92.31 | 46.15 | 38.46 |
| LogCosh Error | 66.67 | 66.67 | 25 | 84.62 | 84.62 | 38.46 |
| Logarithmic Error | 25 | 8.33 | 8.33 | 84.62 | 23.08 | 23.08 |
| Mean Median Error | 91.67 | 75 | 33.34 | 83.33 | 84.62 | 46.15 |
| Minkowski Error | 83.34 | 58.33 | 50 | 92.31 | 53.85 | 23.08 |
| Quadratic Error | 91.67 | 83.34 | 41.67 | 100 | 69.23 | 46.15 |
| Sinh Error | 91.67 | 83.34 | 41.67 | 100 | 69.23 | 46.15 |
| Tukey Error | 83.34 | 58.34 | 33.34 | 92.31 | 69.23 | 61.54 |
| Welsch Error | 58.34 | 58.34 | 25 | 100 | 76.92 | 61.54 |
| Result with 100-10-1 Architecture | | | | | | |
| | Algebraic (% Correct identified) | | | Transcendental (% Correct identified) | | |
| | Test1 | Test2 | Test3 | Test1 | Test2 | Test3 |
| Absolute Error | 50 | 50 | 16.67 | 92.31 | 69.23 | 69.24 |
| Andrew Error | 91.67 | 100 | 100 | 61.54 | 84.62 | 7.69 |

| | | | | | | |
|---------------------|-------|-------|-------|-------|-------|-------|
| Cauchy Error | 91.67 | 83.34 | 41.67 | 92.31 | 76.92 | 53.85 |
| Error Fourth Order | 83.34 | 83.34 | 50 | 76.92 | 69.21 | 53.85 |
| Fair Error | 83.34 | 91.67 | 50 | 92.31 | 76.92 | 38.46 |
| Geman-McClure Error | 75 | 58.34 | 33.34 | 84.62 | 76.92 | 53.85 |
| Huber Error | 83.34 | 91.67 | 50 | 92.31 | 76.92 | 38.46 |
| Hyperbolic Squared | 75 | 58.33 | 41.67 | 92.31 | 69.23 | 53.85 |
| Bipolar Hyperbolic | 75 | 58.34 | 33.34 | 100 | 69.21 | 69.23 |
| LogCosh Error | 83.34 | 83.34 | 33.34 | 61.54 | 61.54 | 15.38 |
| Logarithmic Error | 0 | 0 | 0 | 100 | 100 | 100 |
| Mean Median Error | 91.67 | 91.67 | 41.67 | 92.31 | 61.54 | 69.23 |
| Minkowski Error | 66.67 | 66.67 | 25 | 100 | 76.92 | 53.85 |
| Quadratic Error | 66.67 | 66.67 | 25 | 100 | 84.62 | 53.85 |
| Sinh Error | 83.34 | 75 | 41.67 | 92.31 | 69.23 | 53.85 |
| Tukey Error | 83.34 | 75 | 33.34 | 92.31 | 69.23 | 53.85 |
| Welsch Error | 66.67 | 66.67 | 8.34 | 100 | 69.23 | 61.53 |

still stands and the present problem of Surface Classification was chosen as a case to study the classification of surfaces as stated. The results of this do not generalize to all problems of classification. Therefore, there can exist classification problems in which the CVBP outperforms the BPA and there can be cases in which the result would be otherwise.

6.11 Conclusion

The results clearly indicate that altering the Error Function can be significant as some networks were found to perform on par with the standard Quadratic Error Function based networks, while some displayed sensitivity to activation. This clearly indicates that an optimal choice in terms of EF and Activation Function would yield better result with the problem. With practical data prone to errors a proper choice of Error Function and a Neural Network algorithm built over this can lead to a better network design than retaining the standard Quadratic Error Function based ANN designs. The Error Function based CNN have also exhibited a similar performance for the present problem of classification. This establishes that Error Function together with Activation Function is indeed a parameter that can be varied to advantage while practically applying Neural Networks.

Chapter 7

Conclusion and Scope for Future Work

7.1 Summary

The work presented in the thesis was an attempt to explore the CVBP. The stress was laid on its performance when the Error Function is varied, on how it performs as a classification tool and how some complex valued functions can be mapped. The thesis generalizes the existing Benchmarks listed for the ANN to a form that can accommodate the complex error and be operational in the complex domain and yet retain the functions involved in the mapping. These Benchmark problems have been used as platforms of comparison when the CVBP was evaluated against the BPA and some of its variants. Depending on the results, the Efs have been segregated as good ones, mediocre ones and accordingly recommended or not recommended for replacing the standard Quadratic EF. Some intricate points as to how certain of the EFs could be brought into a useful for to suit the application have also been stated.

The thesis emphasizes an Error Function based approach to data analysis. A few attempts have been reported in literature in this direction from a Neural Network viewpoint. In the present study, not only have most Error Functions employed been studied by developing BPA over each of them but also extended to the complex domain to train CNN forming a wider spectrum of Error Function based Neural Networks. The Error Function echelon formed after the runs using the Benchmarks could be used to develop an Error Function based training and simulation sequence for the new problem at hand.

The study revealed that singular points of the CAF is critical and determines whether the choice of architecture and activation can solve a specific problem. The fact was demonstrated for the Haykin Activation function that had countably many singular points

on the axis of ordinates. In a series of experiments with the Haykin Activation experiment concluded that during the run of the algorithm, some complex numbers fell in a close vicinity of the singular points of the activation function resulting in jolts in the convergence pattern. The other CAFs employed in literature were studied from this viewpoint and a new one was put forth (that was independent of singular points) that converged better than the existing one when applied to the Benchmarks and classification problem discussed.

The CNN's performance as a classification tool was studied by applying the Error Function based networks to the problem of sorting three-dimensional point clouds into Algebraic and Transcendental types. The problem is significant to applications where approximating with polynomials is not appropriate (for instance for analyzing the vibrating air-column of trumpet, approximating the instrument's surface with a polynomial offsets the location of nodes and anti-nodes of the air-column resulting in a diminished tonal quality) as the constraint on the problem is of paramount importance. The standard BPA over different Error Functions as well as CVBP over Error Functions were applied to solve the problem. It was observed that for the third set of test surfaces (that had surfaces not considered for training) the CVBP based CNNs performed better leading to the conclusion that the CNN can be useful when applied to new situations.

7.2 Scope for Future Work

In most research conducted on the CVBP, the learning constant employed was real valued. In principle however, a complex learning constant could be employed. As a ramification, the forms of the update rules for the real and imaginary parts of the weights separately would be different from the usual resulting in a new set of update equations in which the real and imaginary of the complex learning rate appear coupled. The following steps indicate how the problem of complex learning rate could be modeled. The update rule however remains intact in this case except of course the learning rate η is complex

$$w_{ij}(n+1) = w_{ij}(n) + \eta \frac{\partial E}{\partial w_{ij}(n)} \quad (7.1)$$

(E is the Error Function employed) which on breaking into real and imaginary parts assumes the following form

$$w_{ijR}(n+1) = w_{ijR}(n) + (\eta_1 \frac{\partial E}{\partial w_{ijR}(n)} - \eta_2 \frac{\partial E}{\partial w_{ijI}(n)}) \quad (7.2a)$$

$$w_{ijI}(n+1) = w_{ijI}(n) + (\eta_1 \frac{\partial E}{\partial w_{ijI}(n)} + \eta_2 \frac{\partial E}{\partial w_{ijR}(n)}) \quad (7.2b)$$

separately for the real and imaginary parts. It must be noted that a theoretical framework for an investigation of the sort just described must be carried out to start with. That is, a Learning Convergence Theorem must be developed for the case of complex learning rate so the CNNs with complex learning parameters could be studied and later applied. As the rules of update are coupled, the increment of the real part is influenced by the partial derivative of the imaginary part and vice-versa. This characteristic is unlike the CNN with real learning rate where no coupling existed in the update rules. Another interesting fact is the imaginary part of the complex learning constant can be used as a control parameter which when improvised suitably can prove useful. A single real learning parameter was sufficient to train the CNN with the CVBP. As the imaginary part of the complex learning rate η_2 is the parameter that couples the update rules, making the parameter very small would get back the real learning constant case while a larger η_2 would update the weights using the complex learning constant. The description justifies the statement earlier made in this paragraph about the controllability property the complex learning rate can induce into the CVBP. These aspects can be researched into from the viewpoint of Error Functions the present thesis reports.

To accelerate the training process using the BPA, a momentum term is added that acts a force term (Qian, 1995) and enhances the speed of learning. No momentum term has been reported to the CVBP till date. It is not clear how a momentum term functions while training CNNs using the CVBP. The update rule takes the following form in the presence of the momentum term

$$\frac{\sum_i e_i^2}{\sum_i (1+e_i^2)} \quad (7.5)$$

which could be another way of implementing the Geman-McClure function. Clearly the ANN and CNN designs based on composing the functions this way would be different. It is not clear at this time if this scheme would be inferior or superior to the one used in the thesis.

Another point that could be a research direction is to estimate how much bias would be required for performing the input-output map in question. It was observed in some simulations that on tripping the bias connection away, the BPA training progressed well and in fact designed a network that gave satisfactory performance, while, on the other hand, tripping all the bias connections does not solve the problem as the BPA loses its faculty to capture the non-linearity of the data. So a break-even must exist that specifies a minimal bias that suffices for the given data. This aspect can also be researched into.

Needless to say that a performance of the CVBP could be studied by conducting all the runs in a batch-mode and pattern mode training schemes separately for all the directions listed above. Pruning algorithms could be developed for CNNs. Recurrent CNN's could be developed and a Kalman-type approach (Haykin, 1994) for CNNs could be worked out. It is evident in fact that all the concepts of the ANN can be re-worked addressing the CNN. The thesis took a computational approach and EFs have been studied and validated.

Bibliography

- Abramowitz, M. and Stegun, I. A. (Eds.), 1972, "Fresnel Integrals," Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, 9th printing. New York: Dover.
- Ahlfors, L.V., 1979, "Complex Analysis: An Introduction to the Theory of Analytic Functions with one Complex Variable," McGraw-Hill, New York.
- Aihara, K., Takabe, T., Toyoda, M., 1990, "Chaotic Neural Networks," Physics Letters A, vol. 144, no. 6/7.
- Amari, S., 1967, "A Theory of Adaptive Pattern Classifiers," IEEE Trans. On Electronic Computers, EC-16(3), pp 299-307.
- Amir, A., Chuanyi Ji, 1997, "How Initial Conditions Affect Generalization Performance in Large Networks," IEEE Transactions on Neural Networks, vol. 8, no. 2, pp. 448-451.
- Andrew, R.W., 1994, "Functional Approximation by Feed Forward Networks: A Least Squares Approach to Generalization," IEEE Transactions on Neural Networks, vol. 5, no. 3.
- Armstrong, M.A., 1983, "Basic Topology," Springer-Verlag, New-York.
- Artin, M., 1991, "Algebra," Englewood-Cliffs, NJ.
- Askill, J., 1979, "Physics of Musical Sounds," D. Van Norstrand Company, NY.
- Babuska, R., 1998, "Fuzzy Modelling for Control," Kluwer Academic Publishers.
- Battiti, R., 1992, "First and Second Order Methods for Learning: Between Steepest Descent and Newton-Raphson Method," Neural Computation, vol. 4, No. 2, pp. 141-166.
- Beale, E. M. L., 1972, "A Derivation of Conjugate Gradients," F. A. Lootsma, ed., Numerical Methods for Non-linear Optimization, London: Academic Press.
- Benvenuto, N and Piazza, F., 1992, "On the Complex Back-Propagation Algorithm," IEEE Trans. on Signal Processing, Vol. 40, No. 4, pp. 967-969, April.
- Berg, R. E., and Stork, D. G., 1982, "The Physics of Sound," Prentice-Hall, Englewood-Cliffs, NJ.
- Bold, B., 1982, "The Problem of Squaring the Circle," New-York, Dover.

- Bose, N.K. and Liang, P., 1996, "Neural Network Fundamentals with Graphs, Algorithms and Applications," McGraw-Hill International Editions.
- Brent, R. P., 1973, "Algorithms for Minimization without Derivatives," Englewood Cliffs, NJ: Prentice-Hall.
- Brown, J.W. and Churchill, R.V., 1996, "Complex Variables and Applications," Sixth Edition, McGraw-Hill Inc.
- Charalambous, C., 1992, "Conjugate Gradient Algorithm for Efficient Training of Neural Networks," IEEE Proceedings, vol. 139, No. 3, pp. 301-310.
- Chen, S., Cowan, C. F. N., and Grant, P. M., 1991, "Orthogonal Least-Squares Learning Algorithm for Radial Basis Function Networks," IEEE Transactions on Neural Networks, vol. 2, no. 2, pp. 302-309.
- Chun-Shin, L., Chen-Kuo Ki, 2000, "A Sum-of Product Neural Network (SOPNN)," Neurocomputing, vol. 30, pp. 273-291.
- Churchill, R. V., Brown, J., 1993, "Complex-Variables and Applications," 6th Edition, McGraw-Hill, NY.
- Clarke, T.L., 1990, "Generalisation of Neural Networks to the Complex Plane," Proc. IJCNN, San Diego, June.
- Clarke, E., Dewhurst, K., 1972, "An Illustrated History of Brain Function," Berkeley, University of California Press.
- Dagli, C. H., 1994, "Artificial Neural Networks for Intelligent Manufacturing," Chapman and Hall, UK.
- Dai, H., MacBeth, C., 1997, "Effects of Learning Parameters on Learning Procedure and Performance of a BPNN," Neural Networks, vol. 10, no. 8, pp. 1505-1521.
- Deville, Y., 1993, "A Neural Network Implementation of Complex Activation Function for Digital VLSI Neural Networks," Microelectronics journal, vol. 24, pp. 259-262.
- Dodge, Y., 1987, "Statistical Data Analysis based on the L_1 Norm and Related Methods," North-Holland, Amsterdam.
- Eargle, J. M., 1990, "Music Sound and Technology," Van Nostrand Reinhold, NY.
- Fahlman, S. E., Lebiere, C., 1990, "The Cascade-Correlation Learning Architecture," School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213.

- Farin, G., 1995, "NURB Curves and Surfaces: From Projective Geometry to Practical Use," K.A. Peters, Wellesley, Massachusetts.
- Fernandez, B., 1991, "Tools for Artificial Neural Networks Learning," Intelligent Engineering Systems through Artificial Neural Networks (eds. C.H.Dagli, S.R.T. Kumara, Y.C.Shin), 69-76, ASME Press, New York.
- Finger, S., 1994, "Origins of Neuroscience," New York, Oxford University Press.
- Finger, S., 2000a, "Minds behind the Brain: A History of the Pioneers and Their Discoveries," New York, Oxford University Press.
- Fischer, A., Manor, A., 1999, "Utilizing Image Processing Techniques for 3D Reconstruction of Laser-Scanned Data," Annals of the CIBR vol 18, pp 323-331.
- Fletcher, N. H., Rossing, T. H., 1995, "The Physics of Musical Instruments," Springer-Verlag, Third Edition.
- Folland, G. B., 1996, "Introduction to Partial Differential Equations," 2nd Edition, Princeton, NJ: Princeton University Press.
- Francois, C.B., Chauvet, G., 1992, "Stable, Oscillatory and Chaotic Regimes in the Dynamics of Small Neural Networks with Delay," Neural Networks, vol. 5, pp. 735-743.
- Gelfand, I. M., Fomin, S. V., 1963, "Calculus of Variations," Prentice-Hall, Englewood Cliffs.
- Georgiou, G. M. and Koutsougeras, C., 1992, "Complex Domain Back-Propagation," IEEE Trans. on Circuits and Systems – II: Analog and Digital Signal Processing, vol. 39, no.5, May.
- Gill, P. E. and Wright, M. H., 1981, "Practical Optimization," Academic Press.
- Gross, C.G., 1998, "Brain, Vision, Memory. Tales in the History of Neuroscience," Cambridge, MIT Press.
- Guarnieri, S., Piazza, F., 1999, "Multilayer Feedforward Networks with Adaptive Spline Activation Function," IEEE Transactions on Neural Networks, vol. 10, no. 3.
- Gueziec, A., 1999, "Locally Toleranced Surface Simplification," IEEE Transactions on Visualization and Computer Graphics, vol. 5, no. 2, April.
- Halmos, P. R., 1974, "Finite Dimensional Vector Spaces," Springer-Verlag.

- Hampel, F.R., Ronchetti, Rousseeuw, J.P, Stahel, W.A., 1986, "Robust Statistics", Wiley Series in Probability and Statistics, John Wiley & Sons, New-York.
- Hagan, M.T., Menhaj, M., 1994, "Training Feed-Forward Networks with the Marquardt Algorithm," IEEE Trans. on Neural Networks, Vol. 5, No. 6, pp 989-993.
- Hartshorne, R., 1997, "Algebraic Geometry," Springer-Verlag, New York.
- Hassoun, M., 1995, "Fundamentals of Artificial Neural Networks," Prentice-Hall of India.
- Haykin, S., 1994, "Neural Networks: A Comprehensive Foundation" Macmillan College Publishing Company, New York.
- Hinton, G. E., McClelland, J. L., Rumelhart, D. E., 1986, "Distributed Representations. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations," MIT Press, Cambridge, Massachusetts.
- Hirose, A., 1992, "Dynamics of Fully Complex-Valued Neural Networks," Electronics Letters, 28, pp. 1492-1494.
- Hoffman, K. and Kunze. R., 1971, "Linear Algebra," Englewood Cliffs, Prentice Hall, NJ.
- Huang, G. B., Babri, H. A., 1998, "Upper Bounds on the Number of Hidden Neurons in Feedforward Networks with Arbitrary Bounded Nonlinear Activation Functions," IEEE Trans. on Neural Networks, vol. 9, no. 1, pp. 224-233.
- Huber, P.J., 1981, "Robust Statistics," NY: Wiley.
- Hui, K.C., and Li Yadong, "A Feature-Based Shape Blending Technique for Industrial Design," Computer Aided Design, vol. 30, No. 10, pp. 823-834.
- Ingle, K.A., 1994, "Reverse Engineering," McGraw-Hill, New York.
- Ishi, S., Fukumizu, K., Watanabe, S., 1996, "A Network of Chaotic Elements for Information Processing," Neural Networks, vol. 9, no. 1, pp. 25-40.
- Jha, K., Gurumoorhty, B., 2000, "Automatic Propagation of Feature Modification Across Domains," Computer-Aided Design, vol. 32, pp 691-706.
- Jochem, T.M., Pomerleau, D.A., Thorpe, C.E., 1995, "Vision-Based Neural Network Road and Intersection Detection Transversal," IEEE Conference on Intelligent Robots and Systems, August 5-9, Pittsburgh, Pennsylvania, USA.
- Jolliffe, I. T., 1986, "Principal Component Analysis," New York, Springer-Verlag.

- Joris, S.M. Vergeest, Sander, S., Hovarth, I., Jos J.O. Jelier, 2000, "Matching 3D Freeform Shapes to scanned Objects," Proc. of DETC'00, ASME 2000 Design Engineering Technical Conferences And Computers and Information in Engineering Conference, Baltimore, Maryland, September 10-13.
- Karnin, U. D., 1990, "A Simple Procedure for Pruning Backpropagation Trained Networks," IEEE Trans. on Neural Networks, vol. 1, no. 2.
- Kazuyuki, S., 1996, "A Tutorial Review on Bioprocess Systems Engineering," Computers chem. Emgmg, vol 20, no. 6/7.
- Kim, M.S. and Guest, C.C., 1990, "Modification of Back-Propagation for Complex-Valued Signal Processing in Frequency Domain," IJCNN Int. Joint Conf. Neural Networks, pp. III - 27 - III - 31, June.
- Kolmogorov, A.N., 1957, "On the Representation of Continuous Functions of Several Variables by Superposition of Continuous Functions of One Variable and Addition," Doklady Akademii Nauk USSR, vol. 114.
- Kreyszig, E., 1998, "Advanced Engineering Mathematics," 8th Edition, John Wiley and Sons, NY.
- Krishna, K. M., Kalra, P. K., 2000, "Perception and Remembrance of the Environment during Real-time Navigation for a Mobile Robot," accepted for publication in Robotics and Autonomous Systems, Elsevier.
- Lang, S., 1966, "Introduction to Transcendental Numbers," Addison-Wesley Publishing Company.
- Lang, K. J., Witbrock, M. J., 1988, "Learning to tell Two Spirals Apart," Proceedings of the Connectionist Models Summer School, edited by D. Touretzky, G. Hinton, and T. Sejnowski, Morgan Kaufmann, San Mateo, CA., pp. 52-59.
- Lee In-Kwon, 1998, "Curve Reconstruction from Unrecognized Points," Computer Aided Geometric Design, vol. 17, 161-177 Elsevier Science Ltd.
- Leung, H. and Haykin. S., 1991, "The Complex Back-Propagation Algorithm," IEEE Trans. On Signal Processing, Vol. 39, No. 9, September.
- LiMin Fu, 1994, "Neural Networks in Computer Intelligence," McGraw-Hill International Editions, New york.

- Lippman, R. P., 1987, "An Introduction to Computing with Nets," IEEE ASSP Magazine.
- Liu Xiaodong, "CFACA: Component Framework for Feature based Design and Process Planning," Computer-Aided Design, vol. 32, pp 397-408.
- Marshall, L.H., Magoun, H.W., 1998, "Discoveries in the Human Brain," Totowa, Humana Press.
- Massey, W. S., 1991, "A Basic Course in Algebraic Topology," Springer-Verlag, NY.
- Matsuoka, K., & Yi, J., 1991, "Backpropagation based on the Logarithmic Error Function and Elimination of Local Minima," Proceedings of the International Joint Conference on Neural Networks, Singapore, 2, pp 1117-1122.
- Minsky, M, Papert, S., 1969, "Perceptrons: An Introduction to Computational Geometry," MIT Press, Cambridge, Massachussetts.
- Mirchandani, G., Cao, W., 1989, "On Hidden Nodes for Neural Networks," IEEE Transactions on Circuits and Systems, vol. 36, no. 5, pp. 661-664.
- McCulloch, W. S., Pitts, W. H., 1943, "A Logical Calculus of Ideas Immanent in Nervous Activity," Bulletin of Mathematical biophysics, vol. 5.
- Moller, M. F., 1993, "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning," Neural Networks, vol. 6, pp. 525-533.
- Musili, M., 1990, "Analytical Solid Geometry," Narosa Publishing House, New Delhi.
- Nitta, T., 1994, "Decision Boundaries of the Complex Valued Neural Networks," INNS World Congress on Neural Networks, San Diego, vol. 4, pp. 727-732.
- Nitta, T., 1995, "A Quarternary Version of the Backpropagation Algorithm," Int. Conference on Neural Networks, Perth, vol. 5, pp. 2753-2756.
- Nitta. T., 1997, "An Extension of the Back-Propagation Algorithm to Complex Numbers," Neural Networks, Vol. 10, No. 8, pp 1391-1415.
- Ooyen, V., Nienhaus, 1992, "Improving the Convergence of Backpropagation Algorithm," Neural Networks, vol. 5, pp. 465-571.
- Phelan, R. M., 1962, "Fundamentals of Mechanical Design," McGraw-Hill Book Company, NY.

- Qian N, 1999, "On the momentum term in Gradient Descent learning schemes," *Neural Networks* 12, pp145-151.
- Rey, W. J. J., 1983, "Introduction to Robust and Quasi-Robust Statistical Methods," Springer-Verlag, Berlin.
- Rose, F.C., Bynum, W.F., 1982, "Historical Aspects of Neurosciences. Aestschrift for Macdonald Critchley," New York, Raven Press.
- Rudin, W., 1976, "Principles of Mathematical Analysis," McGraw-Hill.
- Rudin, W., 1986, "Real and Complex Analysis," McGraw-Hill.
- Salmon, G., Reginald, A.P., 1965, "A Treatise on Analytic Geometry of Three Dimensions," Chelsea Publishing Company, New-York.
- Siegel, C. L., 1965, "Transcendental Numbers," Chelsea, New York.
- Sinha, M., Kalra, P.K., Kumar, K., 2000, "Parameter Estimation using Compensatory Neural Network," *Sadhana*, Vol. 25, 2, pp. 193-203.
- Sinha, M., Kumar, K., Kalra, P.K., 2000a, "Some new Neural Network Architectures with Improved Learning Schemes," *Softcomputing*, Springer-Verlag, 4, pp. 214-223.
- Smith, M. R., Hui, Y., 1997, "A Data Extrapolation Algorithm Using a Complex Domain Neural Network," *IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing*, vol. 22, no. 2.
- Sommerville, D.M.Y., 1951, "Analytical Geometry of Three Dimensions," Cambridge, At the University Press, Cambridge.
- Spotts, M. F., 1985, "Design of Machine Elements," Third Edition, Prentice-Hall India, New Delhi.
- Stein, S., 1987, "Calculus and Analytic Geometry," McGraw-Hill International Editions, Fourth Edition, New York.
- Tim, H.J.J.V.D.H, 1992, "The Scaling Parameter of the Sigmoid Function in Artificial Neural Networks," *Nuclear Technology*, vol. 106.
- Timoshenko, S. P., Young, D. H., 1988, "Engineering Mechanics," 5th Edition, McGraw-Hill International Editions, NY.
- Timoshenko, S., Woinowski, K., 1959, "Theory of Plates and Shells," McGraw-Hill Book Company, NY.

- Uncini, A., Lorenze, V., Campolucci, Piazza, F., 1999, "Complex Valued Neural Networks with Adaptive Spline Activation Function for Digital Radio Links Nonlinear Equalization," *IEEE Transactions on Signal Processing*, vol. 47, no. 2, pp. 505-515.
- Wang, J., 1992, "Recurrent Neural Networks for Solving Systems of Complex-Valued Linear Equations," *Electronics Letters*, vol. 28, no. 18, pp. 1751-1753.
- Weber, D. M., Casasent, D. P., 1998, "The Extended Piecewise Quadratic Neural Network," *Neural Networks*, vol. 11, pp. 837-850.
- Werbos, P.J., 1974, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," PhD Thesis, Harvard University.
- Werbos, P. J. and Titus, J., 1978, "An Empirical Test of New Forecasting Methods derived from a Theory of Intelligence: The Prediction of Conflict in Latin America," *IEEE Transactions on Systems, Man and Cybernetics*, September.
- Wexler, C., 1962, "Analytic Geometry, A Vector Approach," Addison-Wesley Publishing, Reading, Massachusetts.
- Widrow, B., McCool, J. and Ball, M., 1975, "The Complex LMS Algorithm," *Proc. of the IEEE*, April.
- Yan, X., Yamazaki, K., Liu, J., "Recognition of Machining Features and Feature Topologies from NC Programs," *Computer-Aided Design*, vol. 32, pp. 605-616.
- Zurada, J.M., 1997, "Introduction to Artificial Neural Systems," Jaico Publishing House, New Delhi.

Appendix 1

Error Function based Update Rules

Deriving the Back-Propagation rule for Error Function based algorithms requires incorporating the derivative of the form of the Error Function at appropriate places in the actual derivation of the CVBP (Leung and Haykin, 1992). In the following, the update rules are obtained in a general form for M hidden layers for a general activation function f for the Fourth Power Error Function. The update rules for the other Error Function based algorithms are listed at the end of the derivation.

1. Fourth Power Error Function

$$\varepsilon_j = d_j - y_j$$

The Fourth Power Error Function is defined by the equation

$$E(n) = \sum_{j=1}^{N_M} (\varepsilon_j(n) \varepsilon_j^*(n))^2 \quad (A1.1)$$

$$w_{ij}^{(l)}(n) = wr_{ij}^{(l)}(n) + iw_{ij}^{(l)}(n)$$

$$wr_{ij}^{(M-1)}(n+1) = wr_{ij}^{(M-1)}(n) - \mu \frac{\partial E(n)}{\partial wr_{ij}^{(M-1)}} \quad (A1.2a)$$

$$wi_{ij}^{(M-1)}(n+1) = wi_{ij}^{(M-1)}(n) - \mu \frac{\partial E(n)}{\partial wi_{ij}^{(M-1)}} \quad (A1.2b)$$

Combining the above two equations

$$w_{ij}^{(M-1)}(n+1) + iw_{ij}^{(M-1)}(n+1) = wr_{ij}^{(M-1)}(n) + iw_{ij}^{(M-1)}(n) - \mu \left(\frac{\partial E(n)}{\partial wr_{ij}^{(M-1)}} + i \frac{\partial E(n)}{\partial wi_{ij}^{(M-1)}} \right)$$

which is

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) - \mu \left(\frac{\partial E(n)}{\partial w_{ij}^{(M-1)}} + i \frac{\partial E(n)}{\partial w_{ij}^{(M-1)*}} \right)$$

Evaluating the partial derivatives to open up the expression above

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}^{(M-1)}} &= \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial net_i^{(M)}} \frac{\partial net_i^{(M)}}{\partial w_{ij}^{(M-1)}} + \frac{\partial E}{\partial y_i^*} \frac{\partial y_i^*}{\partial net_i^{(M)*}} \frac{\partial net_i^{(M)*}}{\partial w_{ij}^{(M-1)}} \\ &= 2(\varepsilon_i \varepsilon_i^*) \left(\frac{\partial E_Q}{\partial y_i} \frac{\partial y_i}{\partial net_i^{(M)}} \frac{\partial net_i^{(M)}}{\partial w_{ij}^{(M-1)}} + \frac{\partial E_Q}{\partial y_i^*} \frac{\partial y_i^*}{\partial net_i^{(M)*}} \frac{\partial net_i^{(M)*}}{\partial w_{ij}^{(M-1)}} \right) \end{aligned}$$

with

$$E_Q = \sum_i \varepsilon_i \varepsilon_i^* \quad (\text{the Quadratic Error})$$

$$y_i = f(net_i)$$

$$net_i^{(M)} = \sum_{j=1}^{N_{M-1}} w_{ij}^{(M-1)} x_j^{(M-1)} + \theta_i^{(M-1)}$$

$$x_i^* = f(net_i^{*(l+1)})$$

Evaluating the partials to obtain expressions to compute the weight update formula

$$\frac{\partial E_Q}{\partial w_{ij}^{(M-1)}} = -(d_i^* - y_i^*) f'(net_i) x_j^{(M-1)} - (d_i - y_i) f'(net_i^*) x_j^{*(M-1)}$$

$$\frac{\partial E_Q}{\partial w_{ij}^{(M-1)*}} = -i(d_i^* - y_i^*) f'(net_i) x_j^{(M-1)} + i(d_i - y_i) f'(net_i^*) x_j^{*(M-1)}$$

For computing the contribution from the partial derivatives:

$$\frac{\partial E_Q}{\partial w_{ij}^{(M-1)}} + i \frac{\partial E_Q}{\partial w_{ij}^{*(M-1)}} = -2(d_i - y_i) f'(net_i^*) x_j^{*(M-1)}$$

Multiplying both sides of the above equation by $2\varepsilon_i \varepsilon_i^*$, we get

$$2\varepsilon_i \varepsilon_i^* \left(\frac{\partial E_Q}{\partial w_{ij}^{(M-1)}} + i \frac{\partial E_Q}{\partial w_{ij}^{*(M-1)}} \right) = -2(2\varepsilon_i \varepsilon_i^*) (d_i - y_i) f'(net_i^*) x_j^{*(M-1)}$$

which implies

$$\frac{\partial E}{\partial w_{ij}^{(M-1)}} + i \frac{\partial E}{\partial w_{ij}^{*(M-1)}} = -(4\varepsilon_i \varepsilon_i^*) (d_i - y_i) f'(net_i^*) x_j^{*(M-1)}$$

The net update rule hence takes the following form

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + 2\mu \varepsilon_i \varepsilon_i^* (d_i(n) - y_i(n)) f'(net_i^*) x_j^{*(M-1)}(n) \quad (A1.3)$$

It can be seen that the additional factor $(2\varepsilon_i \varepsilon_i^*)$ that appears in the update is due to the form of the Fourth Power Error Function (comparing with the Quadratic Error Function update rule, displayed in the list below).

The update rule for the hidden layer weights

$$\frac{\partial E(n)}{\partial w_{ij}^{(M-2)}} = \frac{\partial E(n)}{\partial x_i^{(M-1)}} \frac{\partial x_i^{(M-1)}}{\partial net_i^{(M-1)}} \frac{\partial net_i^{(M-1)}}{\partial w_{ij}^{(M-1)}} + \frac{\partial E(n)}{\partial x_i^{*(M-1)}} \frac{\partial x_i^{*(M-1)}}{\partial net_i^{*(M-1)}} \frac{\partial net_i^{*(M-1)}}{\partial w_{ij}^{*(M-2)}}$$

Now since the Error Function is Fourth Power

$$\frac{\partial E(n)}{\partial x_i^{(M-1)}} = -\sum_k 2\varepsilon_k \varepsilon_k^* (d_k^* - y_k^*) \frac{\partial y_k}{\partial x_i^{(M-1)}}$$

$$y_k = f(\text{net}_i^{(M)}) = f\left(\sum_{l=1}^{N_{M-1}} w_{kl}^{(M-1)} x_l^{(M-1)} + \theta_k^{(M-1)}\right)$$

$$\frac{\partial E(n)}{\partial x_i^{(M-1)}} = -\sum_k 2\varepsilon_k \varepsilon_k^* (d_k^* - y_k^*) f'(\text{net}_k^{(M)}) w_{ki}^{(M-1)}$$

Similarly,

$$\frac{\partial E(n)}{\partial x_i^{*(M-1)}} = -\sum_k 2\varepsilon_k \varepsilon_k^* (d_k - y_k) f'(\text{net}_k^{*(M)}) w_{ki}^{*(M-1)}$$

$$\begin{aligned} \frac{\partial E(n)}{\partial w_{ij}^{(M-2)}} &= \left[-\sum_k (2\varepsilon_k \varepsilon_k^*) (d_k^* - y_k^*) f'(\text{net}_k^{(M)}) w_{ki}^{(M-1)}\right] f'(\text{net}_i^{(M-1)}) x_j^{(M-2)} + \\ &\quad \left[-\sum_k (2\varepsilon_k \varepsilon_k^*) (d_k - y_k) f'(\text{net}_k^{*(M)}) w_{ki}^{*(M-1)}\right] f'(\text{net}_i^{*(M-1)}) x_j^{*(M-2)} \end{aligned}$$

Using the above two expressions to eliminate their LHS in an above equation

$$\begin{aligned} \frac{\partial E(n)}{\partial w_{ij}^{(M-2)}} &= i \left[-\sum_k (2\varepsilon_k \varepsilon_k^*) (d_k^* - y_k^*) f'(\text{net}_k^{(M)}) w_{ki}^{(M-1)}\right] f'(\text{net}_i^{(M-1)}) x_j^{(M-2)} \\ &\quad - i \left[-\sum_k (2\varepsilon_k \varepsilon_k^*) (d_k - y_k) f'(\text{net}_k^{*(M)}) w_{ki}^{*(M-1)}\right] f'(\text{net}_i^{*(M-1)}) x_j^{*(M-2)} \end{aligned}$$

Combining the above two into a complex form yields the following equation

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \left[\sum_k (2\varepsilon_k \varepsilon_k^*) (d_k - y_k) f'(\text{net}_k^{*(M)}) w_{ki}^{*(M-1)} \right] f'(\text{net}_i^{*(M-1)}) x_j^{*(M-2)} \quad (\text{A1.4})$$

2. Absolute Error Function

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + \mu \left(\frac{1}{2\sqrt{\varepsilon_i \varepsilon_i^*}} \right) (d_i(n) - y_i(n)) f'(\text{net}_i^*) x_j^{*(M-1)}(n) \quad (\text{A1.5})$$

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \sum_k \frac{1}{2\sqrt{\varepsilon_k \varepsilon_k^*}} (d_k - y_k) f'(net_k^{*(M)}) w_{ki}^{*(M-1)}] f'(net_i^{*(M-1)}) x_j^{*(M-2)} \quad (A1.6)$$

3. Andrew Error Function

For the first part of the definition, the update rule is

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + \mu \left(-\frac{1}{2\pi\sqrt{\varepsilon_i \varepsilon_i^*}} \sin(\pi\sqrt{\varepsilon_i \varepsilon_i^*}) \right) (d_i(n) - y_i(n)) f'(net_i^*) x_j^{*(M-1)}(n) \quad (A1.7)$$

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \sum_k \frac{1}{2\pi\sqrt{\varepsilon_k \varepsilon_k^*}} \sin(\pi\sqrt{\varepsilon_k \varepsilon_k^*}) (d_k - y_k) f'(net_k^{*(M)}) w_{ki}^{*(M-1)}] f'(net_i^{*(M-1)}) x_j^{*(M-2)} \quad (A1.8)$$

and for the second part of the error function, the weight update is zero for the function is defined to be constant for the absolute value of the argument greater than c .

4. Bipolar Hyperbolic Squared Error Function

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + \mu \left(\frac{4}{(4 - (\varepsilon_i \varepsilon_i^*)^2)} \right) (d_i(n) - y_i(n)) f'(net_i^*) x_j^{*(M-1)}(n) \quad (A1.9)$$

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \sum_k \frac{4}{(4 - (\varepsilon_k \varepsilon_k^*)^2)} (d_k - y_k) f'(net_k^{*(M)}) w_{ki}^{*(M-1)}] f'(net_i^{*(M-1)}) x_j^{*(M-2)} \quad (A1.10)$$

5. Cauchy Error Function

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + \mu \left(\frac{1}{1 + \frac{\varepsilon_i \varepsilon_i^*}{c^2}} \right) (d_i(n) - y_i(n)) f'(net_i^*) x_j^{*(M-1)}(n) \quad (A1.11)$$

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \sum_k \frac{1}{1 + \frac{\varepsilon_k \varepsilon_k^*}{c^2}} (d_k - y_k) f'(net_k^{*(M)}) w_{ki}^{*(M-1)}] f'(net_i^{*(M-1)}) x_j^{*(M-2)} \quad (A1.12)$$

6. Fair Error Function

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + \mu \left(\frac{c}{2\sqrt{\varepsilon_i \varepsilon_i^*}} \left(1 - \frac{1}{\left(1 + \frac{\sqrt{\varepsilon_i \varepsilon_i^*}}{c} \right)} \right) \right) (d_i(n) - y_i(n)) f'(net_i^*) x_j^{*(M-1)}(n) \quad (A1.13)$$

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \sum_k \frac{c}{2\sqrt{\varepsilon_k \varepsilon_k^*}} \left(1 - \frac{1}{\left(1 + \frac{\sqrt{\varepsilon_k \varepsilon_k^*}}{c} \right)} \right) (d_k - y_k) f'(net_k^{*(M)}) w_{ki}^{*(M-1)} f'(net_i^{*(M-1)}) x_j^{*(M-2)} \quad (A1.14)$$

7. Geman-McClure Error Function

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + \mu \left(\frac{1}{(1 + \varepsilon_i \varepsilon_i^*)^2} \right) (d_i(n) - y_i(n)) f'(net_i^*) x_j^{*(M-1)}(n) \quad (A1.15)$$

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \sum_k \frac{1}{(1 + \varepsilon_k \varepsilon_k^*)^2} (d_k - y_k) f'(net_k^{*(M)}) w_{ki}^{*(M-1)} f'(net_i^{*(M-1)}) x_j^{*(M-2)} \quad (A1.16)$$

8. Huber Error Function

For the first part of the function in which the error is computed by a quadratic formula. the update rules are

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + \mu (d_i(n) - y_i(n)) f'(net_i^*) x_j^{*(M-1)}(n) \quad (A1.17)$$

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \left[\sum_k (d_k - y_k) f'(net_k^{*(M)}) w_{ki}^{*(M-1)} \right] f'(net_i^{*(M-1)}) x_j^{*(M-2)} \quad (A1.18)$$

For the second part of the function where the definition involves an absolute function. the update rules take the following form

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + \mu \left(\frac{c}{2\sqrt{\varepsilon_i \varepsilon_i^*}} \right) (d_i(n) - y_i(n)) f'(net_i^*) x_j^{*(M-1)}(n) \quad (A1.19)$$

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \left[\sum_k \frac{c}{2\sqrt{\varepsilon_k \varepsilon_k^*}} (d_k - y_k) f'(net_k^{*(M)}) w_{ki}^{*(M-1)} \right] f'(net_i^{*(M-1)}) x_j^{*(M-2)} \quad (A1.20)$$

9. Hyperbolic Squared Error Function

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + \mu \left(\frac{2}{((\varepsilon_i \varepsilon_i^*)^2 - 1)} \right) (d_i(n) - y_i(n)) f'(net_i^*) x_j^{*(M-1)}(n) \quad (A1.21)$$

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \left[\sum_k \frac{2}{((\varepsilon_k \varepsilon_k^*)^2 - 1)} (d_k - y_k) f'(net_k^{*(M)}) w_{ki}^{*(M-1)} \right] f'(net_i^{*(M-1)}) x_j^{*(M-2)} \quad (A1.22)$$

10. Log Cosh Error Function

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + \mu \tanh(\varepsilon_i \varepsilon_i^*) (d_i(n) - y_i(n)) f'(net_i^*) x_j^{*(M-1)}(n) \quad (A1.23)$$

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \left[\sum_k \tanh(\varepsilon_k \varepsilon_k^*) (d_k - y_k) f'(net_k^{*(M)}) w_{ki}^{*(M-1)} \right] f'(net_i^{*(M-1)}) x_j^{*(M-2)} \quad (A1.24)$$

11. Logarithmic Error Function

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + \mu (d_i(n) - y_i(n)) f'(net_i^*) x_j^{*(M-1)}(n) \quad (A1.25)$$

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \left[\sum_k (d_k - y_k) f'(net_k^{*(M)}) w_{ki}^{*(M-1)} \right] f'(net_i^{*(M-1)}) x_j^{*(M-2)} \quad (A1.26)$$

12. Mean-Median Error Function

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + \mu \left(\frac{1}{2(1 + \frac{\varepsilon_i \varepsilon_i^*}{2})} \right) (d_i(n) - y_i(n)) f'(net_i^*) x_j^{*(M-1)}(n) \quad (A1.27)$$

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \sum_k \left(\frac{1}{2(1 + \frac{\varepsilon_k \varepsilon_k^*}{2})} \right) (d_k - y_k) f'(net_k^{*(M)}) w_{ki}^{*(M-1)}] f'(net_i^{*(M-1)}) x_j^{*(M-2)} \quad (A1.28)$$

13. Minkowski Error Function

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + \mu \left(\frac{r}{2} (\varepsilon_i \varepsilon_i^*)^{\frac{r-1}{2}} \right) (d_i(n) - y_i(n)) f'(net_i^*) x_j^{*(M-1)}(n) \quad (A1.29)$$

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \sum_k \frac{r}{2} (\varepsilon_k \varepsilon_k^*)^{\frac{r-1}{2}} (d_k - y_k) f'(net_k^{*(M)}) w_{ki}^{*(M-1)}] f'(net_i^{*(M-1)}) x_j^{*(M-2)} \quad (A1.30)$$

14. Quadratic Error Function

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + \mu (d_i(n) - y_i(n)) f'(net_i^*) x_j^{*(M-1)}(n) \quad (A1.31)$$

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \left[\sum_k (d_k - y_k) f'(net_k^{*(M)}) w_{ki}^{*(M-1)} \right] f'(net_i^{*(M-1)}) x_j^{*(M-2)} \quad (A1.32)$$

15. Sinh Error Function

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + \mu \cosh(\varepsilon_i \varepsilon_i^*) (d_i(n) - y_i(n)) f'(net_i^*) x_j^{*(M-1)}(n) \quad (A1.33)$$

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \sum_k \cosh(\varepsilon_k \varepsilon_k^*) (d_k - y_k) f'(net_k^{*(M)}) w_{ki}^{*(M-1)}] f'(net_i^{*(M-1)}) x_j^{*(M-2)} \quad (A1.34)$$

16. Tukey Error Function

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + \mu \left(\frac{3}{c^2} \left(1 - \frac{\varepsilon_i \varepsilon_i^*}{c^2} \right) \right) (d_i(n) - y_i(n)) f'(net_i^*) x_j^{*(M-1)}(n) \quad (A1.35)$$

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \sum_k \frac{3}{c^2} \left(1 - \frac{\varepsilon_k \varepsilon_k^*}{c^2} \right) (d_k - y_k) f'(net_k^{*(M)}) w_{ki}^{*(M-1)}] f'(net_i^{*(M-1)}) x_j^{*(M-2)} \quad (A1.36)$$

For the other half of the Tukey Error Function, the update is simply zero (for the function is constant in this range of the argument).

17. Welsch Error Function

$$w_{ij}^{(M-1)}(n+1) = w_{ij}^{(M-1)}(n) + \mu \left(\frac{1}{2} \exp\left(-\frac{\varepsilon_i \varepsilon_i^*}{c^2}\right) \right) (d_i(n) - y_i(n)) f'(net_i^*) x_i^{*(M-1)}(n) \quad (A1.37)$$

$$w_{ij}^{(M-2)}(n+1) = w_{ij}^{(M-2)}(n) + \mu \left[\sum_k \frac{1}{2} \exp\left(-\frac{\varepsilon_k \varepsilon_k^*}{c^2}\right) (d_k - y_k) f'(net_k^{*(M)}) w_{ki}^{*(M-1)} \right] f'(net_i^{*(M-1)}) x_i^{*(M-2)}(n) \quad (A1.38)$$



Errata

In all pictures frames through out the thesis where Error Function results have been reported, the functions have been arranged alphabetically in the following sequence: Absolute Error, Andrew Error, Bipolar Hyperbolic Error, Cauchy Error, Fair Error, Fourth Power Error, Geman McClure Error, Huber Error, Hyperbolic Error, Logarithmic Error, Log Cosh Error, Mean-Median Error, Minkowski Error, Quadratic Error, Sinh Error, Tukey Error, Welsch Error.

Table 3.4, Complex XOR Problem, Page 44: The x -axis shows real part of the output while the y -axis shows the imaginary part of the output.

Table 3.6, Complex 3-Parity Problem, Page 49: The x -axis shows real part of the Complex 3-Parity map while the y -axis shows the imaginary part of the very map when the composition is according to the order displayed in expression (3.36).

Table 3.7, Complex 3-Parity Problem, Page 51: The x -axis shows real part of the Complex 3-Parity map while the y -axis shows the imaginary part of the very map when the composition is according to the order displayed in expression (3.37).

Table 3.8, Norm of Complex 3-Parity Map, Page 54: The x -axis shows vector index while the y -axis shows the norms computed as explained in the caption to the figure.

Table 3.11, $z=\sin(x)\sin(y)$ Map, Page 58: The abscissae are respectively the x - and y - as they appear in the definition of the surface. The z -axis shows a plot of the surface within the ranges of the argument as indicated ($[0, \pi/2]$).

Table 3.12, $z=\sin(x)\sin(y)$ Map, Page 60: The abscissae are respectively the x - and y - as they appear in the definition of the surface. The z -axis shows a plot of the surface within the ranges of the argument as indicated ($[0, 2\pi]$).

Fig. 3.3, Complex Map $w=\sin(z)$, Page 64: The x -axis and the y -axis respectively are real and imaginary parts of the function $\sin(z)$.

Table 3.13, Approximating $\sin(z)$, Page 65: The x and y axes respectively are the real and imaginary parts successively with increasing number of terms in a Taylor Series expansion of $\sin(z)$.

Fig. 3.4, Page 67: (a) The x and y axes respectively are the real and imaginary parts of the circle function of equation (3.47) while (b) is the image of the same map shown on a different complex plane.

Table 3.15, Mapping $w = \sin(z_1)\sin(z_2)$, Page 67: The pictures here are a magnification of the one in Fig. 3.4(b).

Table 3.16, Classification and Two-Spirals, Page 72: The output for the Real BPA based classification was chosen to be the coordinates $(0.5,0)$, $(-0.5,0)$. The test data points cluster about these target points.

Fig. 4.1, 4.2 Nitta Activation, Page 84: The abscissae and ordinates are the real and imaginary parts and of the derivatives of the Nitta Activation, equation (4.8).

Fig. 4.3(a), Singular points, Page 87: The figure shows the complex plane of z as defined in equation (4.11). The x - and y - axes are the real and imaginary of the z aforementioned.

Fig. 4.4, Georgiou Activation, Page 88: The figure shows the surfaces of the real and imaginary parts of the Georgiou Activation function as defined in equation (4.17). The x - and y - axes are the real and imaginary of the z aforementioned.

Fig. 4.5, New Activation, Page 90: The figure shows the surfaces of the real and imaginary parts of the New Activation function as defined in equation (4.20). The x - and y - axes are the real and imaginary of the z in the definition.

Fig. 5.1, Bilinear Transformation, Page 96: The complex plane shows the real and imaginary parts of the definition in equation (5.1) for specific values of a , b , c and d .

Table 5.1, Bilinear Transformation, Page 97: The frames show real and imaginary parts on the x and y -axes respectively, of the Bilinear Transformation defined in equation (5.1).

Table 5.2, Bilinear Transformation, Page 99: The frames show real and imaginary parts on the x and y -axes respectively, of the Bilinear Transformation defined in equation (5.1).

Fig. 5.2, Polynomial Map, Page 102: The x - and y - are as in the defining equations (5.4) and (5.5).

Table 5.3, Parabola, Page 103: The x - and y - are as in the defining equation (5.4).

Table 5.4, Parabola, Page 105: The x - and y - are as in the defining equation (5.4).

Table 5.5, Fourth Power, Page 107: The x - and y - are as in the defining equation (5.5).

Table 5.6, Fourth Power, Page 110: The x - and y - are as in the defining equation (5.5).

Table 5.7, Fourth Power, Page 112: The x - and y - are as in the defining equation (5.5).

Table 5.8, Fourth Power, Page 112: The x - and y - are as in the defining equation (5.5).

Fig. 5.3, Similarity Transformation, Page 117: A general complex plane shown with two concentric circles

Table 5.9, Similarity Transformation, Page 117: The axes show real and imaginary parts respectively of a Similarity Transformation.

Table 5.10, Similarity Transformation, Page 119: The axes show real and imaginary parts respectively of a Similarity Transformation.

Table 5.11, Exponential Transformation, Page 122: The axes show real and imaginary parts respectively of the Exponential Transformation defined according to equations (5.7a) and (5.7b).

Table 5.12, Exponential Transformation, Page 124: The axes show real and imaginary parts respectively of the Exponential Transformation defined according to equations (5.7a) and (5.7b).

Table 5.13, Exponential Transformation, Page 127: The axes show real and imaginary parts respectively of the Exponential Transformation defined according to equations (5.7a) and (5.7b).

Table 5.14, Exponential Transformation, Page 129: The axes show real and imaginary parts respectively of the Exponential Transformation defined according to equations (5.7a) and (5.7b).

Fig. 5.4, Exponential Function, Page 132: The axes are according to the defining equation (5.8).

Table 5.15, Exponential Function, Page 132: The axes are according to the defining equation (5.8).

Table 5.16, Exponential Function, Page 134: The axes are according to the defining equation (5.8).

Table 5.17, $w = \sin(z_1) \sin(z_2)$, Page 137: The axes are the real and imaginary parts of the function $\sin(z_1) \sin(z_2)$.

Table 5.18, $w = \sin(z_1) \sin(z_2)$, Page 139: The axes are the real and imaginary parts of the function $\sin(z_1) \sin(z_2)$.

Table 6.2, Algebraic and Transcendental Surfaces, Page 149: The x - and y - are according to the defining equations that appear at the top in each frame.

Fig. 6.1, Points on xy -Plane, Page 157: The xy -Plane shown with points marked.

Figs. 6.2-6.5, Signatures, Pages 158, 163, 164, 165: The abscissa is vector index for each small frame in these figures. The ordinate is the value under the function marked as TrS** (training surface) or Tst** (for test surface). The two asterisks represent the serial number of the function in question.

Fig. 6.6, Points Order Determination, Page 165: The xy -Plane shown with points marked.

Fig. 6.7, Planes Quadratic Cubic, Page 169: The x - and y - are according to the defining equations shown alongside in the very figure.

Fig. 6.8, Signatures, Page 170: The abscissa is vector index, ordinate is the value of the corresponding planar, quadratic or cubic surface defined in Fig. 6.7.

