

WEB BASED IONOSPHERIC FORECASTING USING NEURAL NETWORK  
AND NEUROFUZZY MODELS

A THESIS SUBMITTED TO  
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES  
OF  
MIDDLE EAST TECHNICAL UNIVERSITY

BY

YUSUF İBRAHİM ÖZKÖK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR  
THE DEGREE OF MASTER OF SCIENCE  
IN  
ELECTRICAL AND ELECTRONIC ENGINEERING

APRIL 2005

Approval of the Graduate School of Natural and Applied Sciences

---

Prof Dr. Canan ÖZGEN  
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

---

Prof. Dr. İsmet ERKMEN  
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

---

Prof Dr. Ersin TULUNAY  
Supervisor

Examining Committee Members

|                            |           |       |
|----------------------------|-----------|-------|
| Prof. Dr. Önder YÜKSEL     | (METU-EE) | _____ |
| Prof. Dr. Ersin TULUNAY    | (METU-EE) | _____ |
| Prof. Dr. Yurdanur TULUNAY | (METU-AE) | _____ |
| Prof. Dr. Aydan ERKMEN     | (METU-EE) | _____ |
| Prof. Dr. Tayfun AKIN      | (METU-EE) | _____ |

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Last name: Yusuf İbrahim ÖZKÖK

Signature :

## **ABSTRACT**

### **WEB BASED IONOSPHERIC FORECASTING USING NEURAL NETWORK AND NEUROFUZZY MODELS**

ÖZKÖK, Yusuf İbrahim

MSc. Department of Electrical and Electronics Engineering

Supervisor: Prof. Dr. Ersin Tulunay

April 2005, 145 pages

This study presents the implementation of Middle East Technical University Neural Network (METU-NN) models for the ionospheric forecasting together with worldwide usage capability of the Internet. Furthermore, an attempt is made to include expert information in the Neural Network (NN) model in the form of neurofuzzy network (NFN). Middle East Technical University Neurofuzzy Network (METU-NFN) modeling approach is developed which is the first attempt of using a neurofuzzy model in the ionospheric forecasting studies. The Web based applications developed in this study have the ability to be customized such that other NN and NFN models including METU-NFN can also be adapted.

The NFN models developed in this study are compared with the previously developed and matured METU-NN models. At this very early stage of employing neurofuzzy models in this field, ambitious objectives are not aimed. Applicability

of the neurofuzzy systems on the ionospheric forecasting studies is only demonstrated. Training and operating METU-NN and METU-NFN models under equal conditions and with the same data sets, the cross correlation of obtained and measured values are 0.9870 and 0.9086 and the root mean square error (RMSE) values of 1.7425 TECU and 4.7987 TECU are found by operating METU-NN and METU-NFN models respectively. The results obtained by METU-NFN model is close to those found by METU-NN model. These results are reasonable enough to encourage further studies on neurofuzzy models to benefit from expert information.

Availability of these models which already attracted intense international attention will greatly help the related scientific circles to use the models. The models can be architecturally constructed, trained and operated on-line. To the best of our knowledge this is the first application that gives the ability of on-line model usage with these features.

Applicability of NFN models to the ionospheric forecasting is demonstrated. Having ample flexibility the constructed model enables further developments and improvements. Other neurofuzzy systems in the literature might also lead to better achievements.

Keywords: Ionospheric forecasting, METU-NN Model, METU-NFN Model, METU-IFS, neural network models, neurofuzzy models.

## ÖZ

### SİNİRSEL AĞ VE SİNİRSEL BULANIK BENZEKLER KULLANILARAK WEB TABANLI İYONKÜRESEL ÖNGÖRÜ

ÖZKÖK, Yusuf İbrahim

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Ersin Tulunay

Nisan 2005, 145 pages

Bu çalışma iyonküresel öngörü amaçlı geliştirilen Orta Doğu Teknik Üniversitesi Sinirsel Ağ (METU-NN) benzeklerinin iyonküresel öngörüler için olan bir uygulamasını internetin dünya çapında kullanım olanağıyla birlikte sunar. Bundan başka, Sinirsel Ağ (NN) benzeklere uzman bilgisini Sinirsel Bulanık Ağ (NFN) yapısında dahil etmek için bir girişimde bulunulmuştur. İyonküresel öngörü çalışmalarında bir sinirsel bulanık benzek kullanılarak yapılan ilk girişim olan Orta Doğu Teknik Üniversitesi Sinirsel Bulanık Ağ (METU-NFN) benzeleme yöntemi geliştirilmiştir. Bu tezde geliştirilen ağ tabanlı uygulamalar, daha sonra METU-NFN de içinde olmak üzere geliştirilebilecek yeni NN ve NFN benzeklerinin eklenebilmesine de olanak tanımaktadır.

Bu çalışmada geliştirilen NFN benzekleri daha önce geliştirilmiş ve olgunluğa erişmiş METU-NN benzekleri ile karşılaştırıldı. Sinirsel bulanık ağların bu alanda kullanılması çok erken evresinde olduğu için büyük hedefler amaçlanmamıştır. Sinirsel bulanık dizgelerin iyonküresel öngörü çalışmalarına uygulanabilirliği gösterilmiştir. METU-NN ve METU-NFN benzekleri eşit koşullar altında ve aynı verilerle eğitilip işletilerek, METU-NN ve METU-NFN benzekleri için sırasıyla olmak üzere, elde edilen ve ölçülen değerler arasındaki çapraz ilişki, 0.9870 ve 0.9086 ve kök ortalama kare yanılığı (RMSE) değerleri 1.7425 TECU ve 4.7987 TECU bulunmuştur. METU-NFN benzeği ile elde edilen sonuçlar METU-NN benzeği ile bulunanlara yakındır. Bu sonuçlar sinirsel bulanık benzekler üzerinde uzman bilgiden yararlanmak için yapılacak ilerki çalışmaları özendirmeye yetecek derecede onaylanabilir çıkmıştır.

Şimdiden uluslararası çevrelerin yoğun ilgisini çeken bu benzeklere kolayca erişilebilmesi, ilgili bilimsel çevrelerin benzekleri kullanabilmesine yardımcı olacaktır. Bu benzekler uzaktan erişim ile döngü içi yapılandırılabilir, eğitilebilir ve işletilebilir. Bildiğimiz kadarıyla bu, uzaktan erişimle döngü içi benzek kullanımı yeteneğini sağlayan ilk uygulamadır.

BSA benzeklerinin iyonküresel öngörüler için uygulanabilirliği gösterilmiştir. Geniş esnekliğe sahip olması sayesinde kurulan benzek daha da geliştirilebilir. Yazımdaki öteki sinirsel bulanık dizgelerin kullanılması daha başarılı sonuçlara götürebilir.

Anahtar Kelimeler: İyonküresel öngörü, METU-NN Benzeği, METU-NFN Benzeği, METU-IFS, sinirsel ağ benzekleri, sinirsel bulanık benzekler.

To My Parents



## **ACKNOWLEDGEMENTS**

I would like to express my gratitude to my supervisor Prof. Dr. Ersin Tulunay for his suggestions, support and guidance throughout this thesis. I also would like to thank my committee members Prof. Dr. Önder Yüksel , Prof. Dr. Yurdanur Tulunay, Prof. Dr. Aydan Erkmen, and Prof. Dr. Tayfun Akin for their times and valuable comments on my thesis.

I'm also grateful to ASELSAN Inc. for the resources let me use. I wish to thank to my chief engineer Reyhan Ergün for her tolerance during the study.

Thanks to my friends for their encouragement, support, patience and understanding throughout this study.

Last, but not least my sincere thanks go to my parents, Ekrem and Jale Özkök bringing me today and for giving me the strength and courage to finish this study. If I have a writing skill I owe it to my dad.

## TABLE OF CONTENTS

|  |      |
|--|------|
| ABSTRACT .....   | iv   |
| ÖZ.....  | vi   |
| ACKNOWLEDGEMENTS .....   | ix   |
| TABLE OF CONTENTS .....  | x    |
| LIST OF TABLES .....   | xii  |
| LIST OF FIGURES .....  | xiii |
| LIST OF ABBREVIATIONS AND TERMS .....  | xv   |
| Chapter 1 INTRODUCTION .....   | 1    |
| 1.1 Review of Previous Work .....  | 2    |
| 1.1.1 HF Propagation.....  | 3    |
| 1.1.2 foF2 Prediction, Forecasting and Nowcasting.....                         | 5    |
| 1.2 Models.....  | 8    |
| 1.2.1 Mathematical Models.....   | 8    |
| 1.2.2 Data Driven Models.....  | 9    |
| 1.2.3 Expert Aided and Data Driven Models (Hybrid Models).....                 | 11   |
| 1.3 METU Neural Network Ionospheric Forecasting Software.....                  | 12   |
| Chapter 2 NEURAL NETWORKS AND Training ALGORITHMS.....                         | 14   |
| 2.1 A Brief Description of Neural Networks .....                               | 14   |
| 2.1.1 Multilayer Feedforward Neural Network .....                              | 15   |
| 2.1.2 Basic Properties of Neural Networks .....                                | 17   |
| 2.1.3 METU-IFS Implementation of Neural Networks.....                          | 19   |
| 2.1.3.1 Backpropagation Algorithm.....   | 19   |
| 2.1.3.2 Levenberg Marquardt Modification.....                                  | 22   |
| 2.1.4 Tests and Verification of METU-IFS Implementation of METUNN Models ..... | 26   |
| Chapter 3 NEUROFUZZY METHODS.....  | 35   |
| 3.1 General Description of Neurofuzzy Networks .....                           | 35   |
| 3.2 Fuzzy Inference Systems (FIS).....   | 36   |
| 3.3 Combining Advantages of Both Fuzzy Logic and Neural Networks Methods ..... | 39   |
| 3.4 Neurofuzzy Systems in Literature .....                                     | 40   |
| 3.5 ANFIS .....  | 40   |
| 3.6 ASMOD .....  | 43   |
| 3.7 Curse of Dimensionality .....  | 47   |
| 3.7.1 Overcoming Curse of Dimensionality: ANFIS .....                          | 47   |
| 3.7.2 Overcoming Curse of Dimensionality: ASMOD .....                          | 47   |
| 3.8 METU Neurofuzzy Network (METU-NFN) Design .....                            | 50   |
| 3.8.1 Reanalysis of the problem to determine the fuzzy inputs .....            | 51   |
| 3.8.2 METU-NFN Main Structure .....  | 52   |
| 3.8.3 Design of fuzzy components .....   | 54   |

|  |     |
|--|-----|
| 3.8.3.1 Defuzzification Method .....   | 62  |
| 3.9 Comparison of Various NN and NFN models Applied to Ionospheric Forecasting Problem ..... | 64  |
| Chapter 4 METU IONOSPHERIC FORECASTING SOFTWARE DESIGN .....                                 | 70  |
| 4.1 Introduction to METU Ionospheric Forecasting Software (METU-IFS).....                    | 70  |
| 4.2 Web Based Client-Server Applications.....  | 71  |
| 4.2.1 Installation and Maintenance .....   | 71  |
| 4.2.2 Security .....   | 72  |
| 4.2.3 Cost effective development.....  | 72  |
| 4.2.4 Accessibility.....   | 72  |
| 4.3 Software Design .....  | 73  |
| 4.3.1 Three-Tier Architecture .....  | 73  |
| 4.3.2 Client Side.....   | 74  |
| 4.3.2.1 Main Properties .....  | 74  |
| 4.3.2.2 Authorization Module .....   | 76  |
| 4.3.2.3 Neural Network Builder and Selector Module .....                                     | 78  |
| 4.3.2.4 Training Parameters Settings Module.....   | 80  |
| 4.3.2.5 Operation Settings Module.....   | 82  |
| 4.3.2.6 Plotter Module.....  | 83  |
| 4.3.3 Server Side .....  | 86  |
| 4.3.3.1 Main Properties .....  | 86  |
| 4.3.3.2 Listener Module .....  | 87  |
| 4.3.3.3 Client Handle Module .....   | 88  |
| 4.3.3.4 Training Abstract Module.....  | 88  |
| 4.3.3.4.1 Weight Randomization .....   | 88  |
| 4.3.3.5 LM Module .....  | 89  |
| 4.3.3.6 GD Module.....   | 90  |
| 4.3.3.7 New User Definition Module.....  | 90  |
| 4.3.4 Database Design.....   | 91  |
| 4.3.4.1 USERINFOTABLE .....  | 91  |
| 4.3.4.2 NNSTRUCTURETABLE .....   | 92  |
| 4.3.4.3 TRAININGPARAMETERSTABLE .....  | 93  |
| 4.3.4.4 TRAININGRESULTSTABLE .....   | 94  |
| 4.3.4.5 WEIGHTSTABLE .....   | 95  |
| 4.3.5 Development Environment.....   | 96  |
| 4.3.5.1 Software Environment.....  | 96  |
| 4.3.5.2 Hardware Environment.....  | 96  |
| Chapter 5 CONCLUSION.....  | 97  |
| REFERENCES.....  | 99  |
| A. METU-IFS SRS DOCUMENT.....  | 104 |
| B. DURATION AND EFFORT ESTIMATION OF METU-IFS.....   | 124 |

## LIST OF TABLES

|   |    |
|---|----|
| Table 2.1-1: Training Results Report – 1.....   | 27 |
| Table 2.1-2: Training Results Report - 2 .....  | 29 |
| Table 2.1-3: Comparison of the training and operation results of MATLAB and METU-IFS..... | 34 |
| Table 3.8-1: Boundaries of the regions of diurnal variations .....                        | 59 |
| Table 3.8-2: Defuzzification methods comparison [58] .....                                | 63 |
| Table 3.9-1: Data sets organization .....   | 65 |
| Table 3.9-2: Summary of Comparison Results .....  | 65 |
| Table 4.3-1: Authorization Module structure.....  | 78 |
| Table 4.3-2: USERINFO database table.....   | 92 |
| Table 4.3-3: NNSTRUCTURETABLE database table .....  | 92 |
| Table 4.3-4: A sample entry for NNSTRUCTURETABLE table.....                               | 93 |
| Table 4.3-5: TRAININGPARAMETERSTABLE database table .....                                 | 93 |
| Table 4.3-6: TRAININGRESULTSTABLE database table.....                                     | 94 |
| Table 4.3-7: A sample entry for NNSTRUCTURETABLE table.....                               | 94 |
| Table 4.3-8: WEIGHTSTABLE database table .....  | 95 |
| Table 4.3-9: WEIGHTSTABLE database table .....  | 95 |

## LIST OF FIGURES

|   |    |
|---|----|
| Figure 1.2-1: Architecture of METUNN models [26].   | 10 |
| Figure 2.1-1: Decision regions; a single perceptron divides $O_1$ and $O_2$ planes into two regions [26]. | 17 |
| Figure 2.1-2: Variations graph for test – 1.  | 28 |
| Figure 2.1-3: Scatter diagram for test – 1.   | 28 |
| Figure 2.1-4: Variations graph for METU-IFS – 2.  | 30 |
| Figure 2.1-5: Scatter diagram – 2.  | 30 |
| Figure 2.1-6: Scatter diagram; observed values are obtained by MATLAB NN toolbox.                         | 31 |
| Figure 2.1-7: Scatter diagram; observed values are obtained by METU-IFS.                                  | 32 |
| Figure 2.1-8: Variations graph obtained by MATLAB NN toolbox.   | 33 |
| Figure 2.1-9: Variations graph obtained by METU-IFS.  | 33 |
| Figure 3.2-1: Fuzzy inference system.   | 37 |
| Figure 3.2-2: Commonly used fuzzy if-then rules and fuzzy reasoning mechanisms.                           | 39 |
| Figure 3.5-1: (a) Takagi-Sugeno fuzzy reasoning; (b) ANFIS structure.                                     | 41 |
| Figure 3.6-1: Typical B-spline associative memory network [55].   | 44 |
| Figure 3.6-2: B-spline fuzzy membership functions.  | 45 |
| Figure 3.7-1: Illustration of additive ANOVA decomposition of AMNs [56].                                  | 48 |
| Figure 3.8-1: Characteristic diurnal variation of TEC.  | 52 |
| Figure 3.8-2: METU-NFN main structure.  | 53 |
| Figure 3.8-3: Structure of the NN block with fuzzy inputs.  | 53 |
| Figure 3.8-4: Fuzzy inference system for magnetic disturbance; 1 input, 1 output, 2 rules.                | 55 |
| Figure 3.8-5: Input membership functions for $K_p$ .  | 56 |
| Figure 3.8-6: Output membership functions for $K_p$ .   | 57 |
| Figure 3.8-7: Fuzzy inference system for diurnal variation; 1 input, 1 output, 5 rules.                   | 58 |
| Figure 3.8-8: Regions of diurnal variations.  | 59 |
| Figure 3.8-9: Input membership functions for fuzzification of the diurnal variations.                     | 60 |
| Figure 3.8-10: Output membership functions for fuzzification of diurnal variations.                       | 61 |
| Figure 3.8-11: Defuzzification with CoM.  | 63 |
| Figure 3.9-1: Scatter diagram obtained by operating NFN 2NN.  | 66 |
| Figure 3.9-2: Scatter diagram obtained by operating NFN 1NN.  | 67 |
| Figure 3.9-3: Scatter diagram obtained by operating NN + $K_p$ .  | 68 |
| Figure 3.9-4: Scatter diagram obtained by operating NN + $K_{pf}$ .                                       | 69 |
| Figure 3.9-5: Scatter diagram obtained by operating METU-NN.  | 69 |

|  |    |
|--|----|
| Figure 4.3-1: METU-IFS main window.....  | 75 |
| Figure 4.3-2: METU-IFS "About" window.....   | 76 |
| Figure 4.3-3: Connection error window.....   | 76 |
| Figure 4.3-4: Login window. ....   | 77 |
| Figure 4.3-5: NN structure selection window. ....  | 79 |
| Figure 4.3-6: NN structure building window. ....   | 80 |
| Figure 4.3-7: Training parameters selection window when LM is selected as an<br>algorithm..... | 81 |
| Figure 4.3-8: Training parameters selection window when GD is selected as an<br>algorithm..... | 82 |
| Figure 4.3-9: Operation settings window.....   | 83 |
| Figure 4.3-10: Browser window.....   | 83 |
| Figure 4.3-11: A training error graph scaled to 0.2. ....                                      | 84 |
| Figure 4.3-12: A training error graph scaled to 0.02.....                                      | 85 |
| Figure 4.3-13: A scatter diagram window of METU-IFS. ....                                      | 86 |
| Figure 4.3-14: Main window of server.....  | 87 |
| Figure 4.3-15: The window to define a new user. ....   | 90 |

## LIST OF ABBREVIATIONS AND TERMS

|                 |  |
|-----------------|--|
| <b>ANFIS</b>    | : Adaptive Network based Fuzzy Inference System  |
| <b>ASMOD</b>    | : Adaptive Spline Modeling of Observation Data   |
| <b>Client</b>   | : A computer system or process that requests a service of another computer system or process (a "server") using some kind of protocol and accepts the server's responses. A client is part of a client-server software architecture. |
| <b>COST</b>     | : Cooperation in Scientific and Technical Research   |
| <b>FIS</b>      | : Fuzzy Inference System   |
| <b>HF</b>       | : High Frequency   |
| <b>MSE</b>      | : Mean Square Error  |
| <b>METU-IFS</b> | : Middle East Technical University Ionospheric Forecasting Software  |
| <b>METU-NFN</b> | : Middle East Technical University Neurofuzzy Network  |
| <b>METU-NN</b>  | : Middle East Technical University Neural Network  |
| <b>NF</b>       | : Neurofuzzy   |
| <b>NFN</b>      | : Neurofuzzy Network   |
| <b>NN</b>       | : Neural Network   |
| <b>RMSE</b>     | : Root Mean Square Error   |
| <b>Server</b>   | : A program which provides some service to other (client) programs.  |
| <b>SRS</b>      | : Software Requirements Specifications   |
| <b>TEC</b>      | : Total Electron Content   |
| <b>TECU</b>     | : Total Electron Content Unit (el/sqm *10 <sup>16</sup> )  |

# **CHAPTER 1**

## **INTRODUCTION**

The researches have already so improved that the ionospheric forecasting results can effectively be used in the real world problems today. There seems to be, however, no saturation point yet, there is still room for improvement. All over the world some research groups have developed some web pages to publish ionospheric forecasting results which are obtained offline using their own data in their own models. Therefore, such forecasts or predictions are reliable and dependable as far as user's and publisher's data match with each other, seriously limiting the benefits the interested parties wish to get in most cases. In this study an application has been developed to promote widespread use of the models that are known to have made the best ionospheric forecasts. The fact that it is possible to make our models serve the scientific circles in the entire world through the Internet is the primary benefit and advantage this application provides. A mass amount of accumulated data to be collected through such widespread worldwide participation of interested parties is expected to contribute significantly to more advanced models to be developed by METU-NN and METU-NFN. Moreover, the ability to integrate with itself the models to be developed in future is it's another noteworthy feature. To be able to develop this application previous work on the subject is studied in detail. This chapter gives a summary of the previous work of Turkish research group which was done in the scope of European Union Cooperation in Scientific and Technical Research (COST) 251: The Improved Quality of Service in Ionospheric Telecommunication Systems Planning and Operation (IITS) and COST 271 actions. Furthermore, this chapter also discusses some models developed for the ionospheric forecasting, including METU-NN models since they are used in METU-IFS. Chapter 2 discusses in detail



the neural networks and the training algorithms which are implemented in METU-IFS, and Chapter 4 the design details of METU-IFS.

In addition to this application, neurofuzzy systems have also been reviewed hoping to go beyond the point attained by neural network based ionospheric forecasting models and develop neurofuzzy models with all their possible advantages [1].

A review of the literature indicates that neurofuzzy systems are still a long way from reaching maturity, and new developments and improvements on them are following one another day by day. The most common widespread feature of the neurofuzzy applications is that they all have fuzzy but well covered information. So, it indicates that fuzzy but well covered information is amongst the requirements for us to develop a neurofuzzy model. Moreover, applicability of the neurofuzzy techniques seems to be decreased due to the fact that a large number of inputs makes the problem more difficult to be defined and reduces the control over the expert knowledge. Amongst the many neurofuzzy systems in literature research groups only focus on two systems that are known as ANFIS and ASMOD. The primary distinctive feature which is distinguishing them from the others is their being relatively more well defined. The problems arisen from a large number of inputs are also dealt with and discussed in these systems [2]. Reviewing these neurofuzzy systems, METU-NFN system is developed in this study. The main objective of this work is to prove the applicability of the neurofuzzy systems to the ionospheric forecasting studies. Chapter 3 presents a review of the neurofuzzy systems and discusses METU-NFN model.

## **1.1 Review of Previous Work**

This section presents a review of the previous work on the ionospheric forecasting studies. The studies on HF propagation related to the ionospheric forecasting is first reviewed and then the terms "prediction", "forecasting", and "nowcasting" are discussed.

### **1.1.1 HF Propagation**

The research and studies on the influence of the near-Earth space on telecommunication do not go back a long way. The scientific and technical research activities have been going on since early 1990s [3][4][5][6]. For the common advantage of different research groups in solving the problems, an initiative known as Cooperation in Scientific and Technical Research (COST) was formed in the European Union, opening possibilities of cooperation to the researchers from different member states. COST Action 238, 'Prediction and Retrospective Ionospheric Modeling over Europe' (PRIME) [7] was officially signed in March 1991 as a four-year project, aiming to develop improved models of the European Ionosphere for telecommunication applications. This project, PRIME, had been implemented in 1995 with the participation of thirty-one organizations from seventeen countries. Following COST 238 another action, COST 251, 'The Improved Quality of Service in Ionospheric Telecommunication Systems Planning and Operation' (IITS) was given a start in 1995. And the most recent action, COST 271 has the objectives to stimulate international co-operation in predicting and forecasting the ionosphere and plasmasphere; to develop and implement new communication services; to minimize the effects ionospheric perturbations have on communications systems; and to collect new data for now-casting and forecasting.

The Turkish contributions to these three actions, COST 238, COST 251, and COST271 are also important and noteworthy, which are on the topics of 'Total Electron Content (TEC) Variation during the eclipse of the sun on 29<sup>th</sup> of April, 1976' and 'The HF Link Experiment during the eclipse of the sun on 11<sup>th</sup> of August, 1999' [8], and reporting the ionospheric TEC over Ankara on 29<sup>th</sup> of April, 1976, and the effects of the ionosphere on HF radio wave during its propagation over 700 km between Ankara and Elazig on 11<sup>th</sup> of August, 1999. The first research, the ionospheric TEC measurement in Ankara was done by Ertac and Tulunay, Y, in 1979 [9]. The effect of increasing magnetic activity is proven to be produced as a result of the diurnal variation in the ionospheric TEC prior to the solar eclipse. As the daily sum of the three-hour planetary magnetic activity

index,  $K_p$ , increases, pronounced rises occurred in the ionospheric TEC, reaching peak values at noon.

Over some 700 km distance between Ankara ( $40^\circ$  N,  $33^\circ$  E) and Elazig ( $38^\circ$  N,  $39^\circ$  E), an HF radio wave was transmitted during the period of the total eclipse of the sun on 11<sup>th</sup> of August, 1999. Making good use of that very opportunity of August 11, the effect of the eclipse on the ionosphere, and thus, the changes in signal strength that the disturbed ionosphere causes in HF radio waves propagating during the time period of the eclipse were observed and analyzed through this experiment of HF Signal Strength Measurement. The results proved a clear decrease of the strength during the eclipse, reaching a minimum as getting closer to total eclipse over Elazig. After the total eclipse the strength began to increase to a value observed at the beginning of the eclipse.

Thus, it was proven that the signal strength dropped due to the effects on the ionosphere caused by the solar eclipse of August 11. So an inference or conclusion can be drawn from this result; an opposite effect, a rise in the signal strength is expected, because the rate of absorption in the ionosphere falls due to the decrease of the photoelectrons in the shadow of the moon. A portion of the HF waves is believed to be transmitted into space rather than being reflected back to the Earth due to the thinning of the ionosphere, since the HF radio wave frequency of 18.111 MHz is close to the natural ionospheric critical frequency.

One of the two methods [10] to model HF propagation is a neural network model as trained with data obtained through the results of a linear model simulating the behavior of the ionosphere for a particular choice of path. The other one is a neural network model as trained with data obtained through actual communication link of signal strengths. The models are trained with  $R_{12}$  which is a temporal data, the twelve month mean of the international monthly mean relative sunspot number, and also with  $f_oF2$  data. The signal to noise strength ratio, SNR, is predicted by the neural networks [11] for which the data sets of testing and training are introduced to train the neural networks and the predictions are given in [10]. The physical phenomena affecting HF propagation

is modeled in this study, and an emphasis is placed on the importance of  $foF2$ , and preliminary studies on the selection of channel depending on time is aimed.

### **1.1.2 foF2 Prediction, Forecasting and Nowcasting**

In order to forecast ionospheric parameters a variety of methods, some linear, some non-linear, have been developed within the European Union COST 251 Action framework. It seems that, applying the non-linear methods and especially neural network techniques, the efforts to make improvements in ionospheric forecasting capacity are very promising [12].

Before going further, it should be noted here that the terms 'prediction', 'forecasting' and 'nowcasting' are usually confused in literature. So a distinction is needed to be made between them to avoid confusion. The term 'prediction' refers to the estimation of the  $foF2$  values through the median figures of the data available whereas the term 'forecasting' through the actual hourly figures. The term 'nowcasting' is nothing but the estimation of the  $foF2$  values for data gaps, i.e. a kind of short-term forecasting.

Neural networks had already been employed to study various aspects of geophysics such as the applications in forecasting of the magnetic storms [13]. However, for the purpose of one hour - advance forecasting of  $foF2$ , neural network was first used by Altinay, *et al.* [14][15], proving the advantages. Ionospheric prediction and short term forecasting [16], modeling of noon-day variations of  $foF2$  [17], preliminary studies for the prediction of monthly median  $foF2$  values [18], prediction of noon value of  $foF2$  [19] are also noteworthy studies associated with the methods based on the neural network, which only consider the temporal variation of the  $foF2$  values.

Making use of the root mean square error between the measured and predicted values of  $foF2$  as a criterion in determination of the optimum indices of solar and magnetic activity, Willisroft was the first to apply neural networks for the prediction of  $foF2$  [19]. It is known that the ionospheric electron density is a function of latitude, longitude, season, local time, solar and magnetic activities.

Selecting the noon value of  $f_oF2$  for Grahamstown as the target value, latitude, longitude and local time variables are eliminated. The remaining three variables, namely, season, solar activity and magnetic activity, serve as input or in other words, data for training of the neural network, which once trained can predict the daily noon value of  $f_oF2$  in Grahamstown with an absolute error of 0.95 MHz.

In another study on  $f_oF2$  forecasting based on neural network done by Tulunay, Y., Tulunay, E., and Senalp, E., the influence of the electron density trough is further analyzed and data generated by using statistical relationships are used to train the neural network and the trained network is used to forecast one-hour-advance  $f_oF2$  values under the conditions where the influence of the trough is expected to be high.

In this study it is emphasized that more advanced and novel neural network based models for more reliable forecasting of  $f_oF2$  values is a matter of concern for scientists and system operators. Because a highly non-linear problem, both in space and time, and an extensively disturbed process pose a real challenge. To represent finer variations of  $f_oF2$ , which are usually, lost when taking averages, better understanding of the process rather than modeling is believed to promote a more promising and challenging approach.

The influence of the electron density trough at which the altitudes at the  $f_oF2$  is measured cannot be ignored in forecasting  $f_oF2$  values. Therefore, the models proposed to forecast  $f_oF2$  must include the trough effect. In this study, a neural-network- based model has been constructed attempting to include some characterizing behavior of the trough. The statistical behavior of the trough was determined by considering the temporal and magnetic conditions so that there is some probability that trough can be observed in the ionosphere above the Slough and Uppsala. For both conditions, the daytime disturbed and nighttime – quiet, the neural network model did successfully forecast the  $f_oF2$  values one-hour in advance for Slough and Uppsala.

The electron density trough exhibits abrupt gradients of electron densities within relatively short horizontal distances and in time, in particular, over the midlatitude ionospheric regions in both hemispheres. Since the  $f_oF2$  values are directly reflecting the variations of ambient electron densities, in the HF communication process, the behavior and influence of the trough have to be modeled. But the trough is such a complex nonlinear reality that it is almost impossible to model its behavior and influence with analytical methods. It is, however, demonstrated here that a data-driven model, such as the neural-network-based approach, proves to be successful.

The neural network model was employed with  $f_oF2$  values of Slough and Uppsala for the trough case and for the general case in which  $f_oF2$  values are clearly free from the influence of the trough and for randomly chosen  $f_oF2$  values of similar size in order to compare these three groups in terms of the errors. It was successfully concluded and demonstrated that the smallest error was obtained by operating the neural network that was trained by using the restricted data set, which reflects the characteristics of the trough. The other two groups had larger errors than the errors obtained for the trough case.

Would the neural network systems have the ability to learn the shape of any inherent nonlinear variations if they can be properly constructed and trained? This work, by successfully employing this ability to meet such an intellectual challenge of a highly nonlinear and extensively disturbed processes, has proven that there is such an ability in neural network systems.

The neural network system obtained high correlation coefficients between the observed and forecasted values of  $f_oF2$ , and approached the desired operating point, giving rise to some small errors.

The neural network model proposed in this work is also very useful in filling the data gaps. During abrupt density gradients or during severe magnetic disturbances data are missing most of the time. Since such cases often occur in

nature, filling the missing data gaps is a primary feature to be expected from a reliable model.

## **1.2 Models**

This section discusses some models developed to employ on the ionospheric forecasting studies. These models are mainly divided into three groups which are namely mathematical models, data driven models and hybrid models.

### **1.2.1 Mathematical Models**

For a single station prediction and one-hour advance forecasting of the ionospheric critical frequency  $f_oF2$ , a study was done by Bilge and Tulunay, Y. using a novel on-line mathematical method that is based on applying feedback on predicted monthly median values of  $f_oF2$  for each hour [20]. A parabolic dependency on  $R_{12}$  superimposed by a trigonometric expansion in terms of the harmonics of annual variation and linearly modulated by  $R_{12}$  is all what makes the basic model for the prediction of monthly medians. Applying the basic model over a sliding data window, hourly monthly medians can be predicted. For prediction, 'sliding data windows' and for forecasting, 'feedback' are used as the main tools of the method.

Making use of the past data and some predictable parameters, one-hour advance forecasting of  $f_oF2$  can be formulated by estimating a non-stationary time series with deterministic slow variations together with some irregular fast variations. Amongst the variations influencing  $f_oF2$  the slowest one, with a period of 11 years, is the variation which is due to solar activity. All these effects are taken into account in the model by a parabolic dependency in  $R_{12}$ , although a linear fit also works for one-hour advance forecasting purposes. Even though their amplitudes depend on the level of solar activity, the medium range variations of the order of months are the periodicities corresponding to the harmonics of the annual variation. Depending on solar activity a trigonometric expansion with linear coefficients is used to model these variations [21]. Applying this basic

model to a sliding data window using immediate past information for prediction is the novelty of the approach [20].

An estimate for  $foF2$  for each hour of the day of the forthcoming month is obtained by applying the prediction model to the monthly medians of  $foF2$  for each hour. The predicted hourly median values, thus obtained, are arranged in a time series denoted by  $foF2_p$ . The deviations from the predicted values are forwarded to the feedback; the difference between the actual and the predicted values of  $foF2$  is computed at each hour and an appropriate fraction of this error is subtracted from the predicted value of  $foF2$  for the next hour. The resulting time series denoted by  $foF2$  gives the one-hour advance forecast of  $foF2$  as

$$f^*(R_{12}, m) = (a_1 R_{12} + a_0) + \sum_{i=1}^6 \left( (b_i + c_i R_{12}) \sin \left( \frac{2\pi i}{12} m + \frac{\pi}{2} \right) \right) + \sum_{i=1}^5 \left( (d_i + e_i R_{12}) \cos \left( \frac{2\pi i}{12} m + \frac{\pi}{2} \right) \right) \quad (1.2.1)$$

The method of sliding window and the feedback technique were proven to be comparable with the forecasting methods based on neural network and superior to autocovariance prediction [20][22][23]. The models with more sophisticated functional dependencies provide only a little advantage of a minor increase in the overall performance, while the hysteresis effects are eliminated and prediction errors reduced considerably by the restriction of simpler models to shorter periods of time [24]. Although such models are useful in short term forecasting or nowcasting, they can hardly be used for long term predictions.

Given its simplicity, the performance of the method can be regarded satisfactory, but not in the presence of strong irregular variations like solar storms. Further improvements can be made introducing more sophisticated signal processing techniques as well as other physical parameters into the model [20].

### 1.2.2 Data Driven Models

Data driven models that are implemented in METU-IFS application are given in this section. Results of Neural Networks applications on highly nonlinear and



complex real world processes are very promising. Previous studies on Ionospheric forecasting also demonstrated the applicability of neural network models on this area with very successful results [25][26][27]. Basic structure and properties of neural networks are briefly explained in second chapter quoted from a well structured paper on neural networks [26].

In this study previously developed Neural Network models named as METU-NN, Middle East Technical University Neural Network, models are used. As a reason METU-NN models will briefly be described here.

The METU-NN models have got one input layer, one hidden layer and one output layer. The basic architecture of the models is demonstrated in Figure 1.2-1 [26][27].

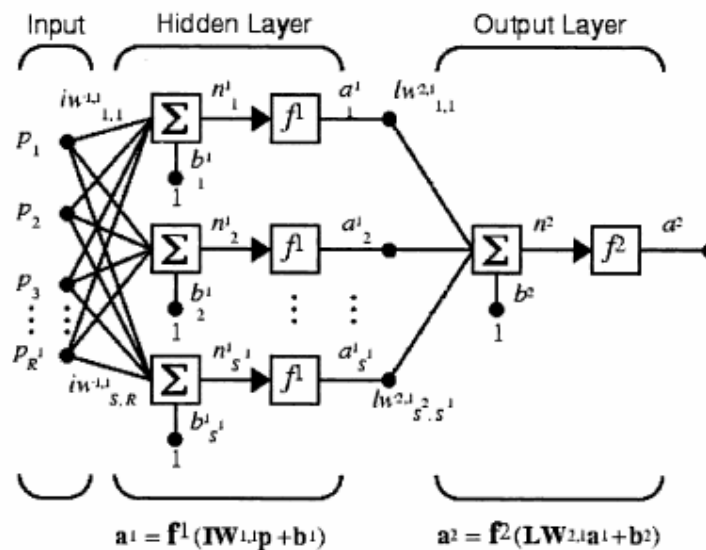


Figure 1.2-1: Architecture of METUNN models [26].

Hyperbolic tangent sigmoid functions are used as the activation functions, in the hidden layer and pure linear transfer function is used as the activation function, in the output layer.

$$f^1(n) = TANSIG(n) = \frac{2}{1 + \exp(-2.n)} - 1 \quad (1.2.2)$$

$$f^2(n) = PURELIN(n) = a.n \quad (1.2.3)$$

One of the models implemented is the one employed to forecast *foF2*. The basic inputs to the model are the temporal inputs, spatial inputs, and inputs related to the history of *foF2*.

The basic inputs related to the history of the *foF2* are:  $f(h)$ : the value of *foF2* at time instant  $h$ , present value;  $\Delta_1(h) = f(h) - f(h-1)$ : First difference;  $\Delta_2(h) = \Delta_1(h) - \Delta_1(h-1)$ : Second difference;  $R \Delta(h) = \Delta_1(h) / f(h)$ : Relative difference. And the output  $f(h+1)$  is the forecast value of *foF2*. These definitions are for the model which can forecast *foF2* one hour in advance. The definitions can easily be converted for 24 hour in advance forecasting by simply changing ones to twenty fours [25].

For temporal inputs, basically The Universal Time (UT) data and time, coded hour, coded day or hour and day trigonometric components to take the adjacency of 24 and 1 for hour 30 and 1 for day into account are used.

Descriptions of training algorithms used to train this model with the training and operation results obtained by using METU-IFS are given in Chapter 2.

### **1.2.3 Expert Aided and Data Driven Models (Hybrid Models)**

The main advantages of using NNs are their flexibility and ability to model nonlinear relationships. In contrast to knowledge-based techniques, no explicit knowledge is needed for the application of neural nets. And the results of Ionospheric forecasting studies based on neural networks are promising. However, despite these advantages, NNs have often been criticized for acting as “black boxes”. The knowledge contained in an NN model is kept in the form of a weight matrix that is hard to interpret and can be misleading at times.

One way to overcome many of these shortcomings is to use neurofuzzy models which are known as expert aided systems. Neurofuzzy systems combine the semantic transparency of rule-based fuzzy systems with the learning capability of neural networks. So the models developed based on neurofuzzy systems are sometimes named as Hybrid Models. They can be trained to perform an input/output mapping, just as with an NN, but with the additional benefit of being able to provide the set of rules on which the model is based. This gives further insight into the process being modeled [28].

Given the facts, neurofuzzy systems are worth to be put on trial on the Ionospheric Forecasting studies. As an early step in this work neurofuzzy systems are reviewed. Amongst the many neurofuzzy systems, ASMOD and ANFIS systems are examined in detail. A hybrid system is then developed mixing the advantages of these systems with our intellectual aggregation on the subject of ionospheric forecasting. Neurofuzzy systems reviewed and our neurofuzzy network system design is discussed in detail in Chapter 3.

### **1.3 METU Neural Network Ionospheric Forecasting Software**

Middle East Technical University Ionospheric Forecasting Software (METU-IFS) has been developed to make it possible that various neural network and neurofuzzy models can be used over the Internet with a user friendly interface. The Neural Network models introduced on the previous sections have been implemented and the design studies of this software initiated with the preparation of the SRS (Software Requirements Specifications) document and progressed to the function point analysis. Chapter 4 gives the details of the software design.

With this software, ionospheric forecasting models are made to be available for the service of the interested scientific circles. Designing new neural network or neurofuzzy architectures, training previously designed ones and operating previously trained ones are the main features of METU-IFS. Remote access to these features is one of the primary objectives of this software. It also possesses

a database to store a vast variety of data to be collected while the software is serving. The data to be collected from so many different sources all over the world while the models are being trained or operated are expected to be useful to improve the models for future researches.

## CHAPTER 2

### NEURAL NETWORKS AND TRAINING ALGORITHMS

Neural networks have been employed in the ionospheric forecasting models developed in the earlier works as detailed in the preceding chapter. In this chapter neural networks and training algorithms will be overviewed first, and then we will discuss how we have implemented in this study the Backpropagation and Levenberg Marquardt algorithms. The implementation is verified by comparing METU-IFS results with those obtained from MATLAB code, as it is shown in the last part of this chapter.

#### 2.1 A Brief Description of Neural Networks

A neuron is an information-processing unit consisting of connecting links, adder and activation function. The adder is for summing bias and the input signals weighted in the neuron's connecting links. It follows an activation function for limiting the amplitude of the neuron's output [29]. An artificial neural network is a system of inter-connected computational elements, the neurons, operating in parallel, arranged in patterns similar to biological neural nets and modeled after the human brain [30]. Individual neurons are characterized by

$$O_i = f_i(x_i) \quad (2.1.1)$$

and

$$x_i = net_i = \sum_{j=1}^{N_i} w_{ij} O_j + \theta_i \quad (2.1.2)$$

where  $O_i$ , is the output of neuron  $i$ ,  $x_i$  is the sum of the weighted inputs or net output of neuron  $i$ ,  $f_i(.)$  is the activation function of neuron  $i$ ,  $w_{ij}$  is the weight of the arc from neuron  $j$  to neuron  $i$ ,  $O_j$  is the output from the  $j$ th neuron of the previous layer. It is also the input to neuron  $i$  from neuron  $j$ .  $\theta_i$  is the internal threshold, bias or offset for node  $i$ , and  $N_i$  is the number of inputs to neuron  $i$ . A neuron thus forms a weighted sum of  $N_i$  inputs and passes the result through a linear or nonlinear activation function. A node with a hard limiting activation function is sometimes called as thresholding unit. Sigmoid units are more complicated but more powerful than hard limiting units because, the sigmoid is an increasing, continuous function. It has non-zero derivatives, which makes it useful in gradient descent learning methods [30]. The linear model has limitations and is not useful in hidden layers, because a linear multi-layer neural network can always be represented as an equivalent linear single layer network [30].

The architecture of a neural network is formed by determination of the neuron structures and their connections. In a layered neural network the neurons are organized in the form of layers. In multi-layer feed forward neural networks the input layer of source nodes projects onto a hidden layer consisting of hidden neurons. If there are more than one hidden layers in the architecture, then the hidden layer projects onto another hidden layer consisting of hidden neurons. The last hidden layer in the architecture projects onto an output layer of nodes. Those networks are strictly feedforward or acyclic type. Also they are fully connected in the sense that every neuron in each layer of the network is connected to every other neuron in the adjacent forward layer [29].

### **2.1.1 Multilayer Feedforward Neural Network**

The capabilities of multi-layer neural networks can be understood as a result of a theorem based on the 13<sup>th</sup> problem of Hilbert and proved by Kolmogorov

[31][32]. This theorem states that any continuous function of N variables  $f(x_1, \dots, x_N)$  can be written using only linear summations and nonlinear but continuously increasing functions  $\Phi_{pq}(x_p)$  in only one variable

$$f(x_1, \dots, x_N) = \sum_{q=1}^{2N+1} g_q \left( \sum_{p=1}^N \phi_{pq}(x_p) \right) \quad (2.1.3)$$

Thus it is seen that, any continuous function of N variables can be computed by using a three-layer network with  $N(2N+1)$  nodes having continuously increasing nonlinearities. However, the theorem does not show how weights and nonlinearities must be selected or how sensitive the output function is to variations in the weights and internal functions [31].

Virtually all multi-layer applications have used two hidden layers or less [33]. As the number of layers increases, data storage efficiency, ability of the network to generalize, and the fail-safe nature of the network increases. A more layered and more highly connected network can generally store the same data in fewer neurons.

The capabilities of perceptrons can be seen by investigating the decision regions they can form in the space defined by the inputs. As a simple example, consider the single unit  $i$  defined by Eq. (2.1.2) and hard limiting activation function for the case of two inputs  $O_1$  and  $O_2$ . It is seen that this unit divides the  $O_1, O_2$  plane into two decision regions separated by the line

$$net_i = \sum_{j=1}^{N_i} w_{ij} O_j + \theta_i = w_{1i} O_1 + w_{2i} O_2 + \theta_i = 0 \quad (2.1.4)$$

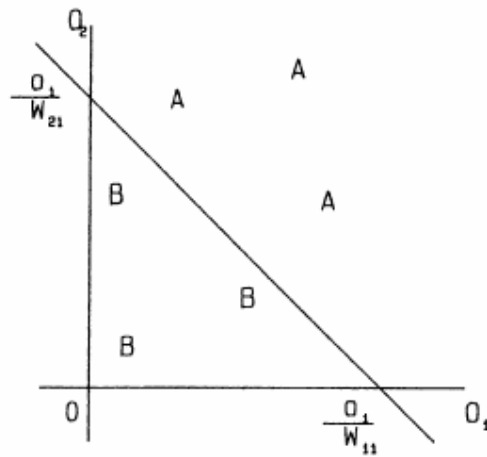


Figure 2.1-1: Decision regions; a single perceptron divides  $O_1$  and  $O_2$  planes into two regions [26].

as shown in Figure 2.1-1. Suppose input pattern A causes the right hand side of Eq. (2.1.2) to be positive. Then the output of the unit is 1. If input pattern B causes the right hand side to be negative then the unit will output 0. Thus the unit decides whether the input pattern belongs to class A or class B as shown in Figure 2.1-1. In general  $net_i$  in Eq. (2.1.4) is the equation of a  $N_i$ , dimensional hyperplane. This hyperplane partitions the space defined by the inputs to that node into two parts. Any point on one part of this hyperplane corresponds to the input class that causes output 1 and any point on the other part corresponds to the input class that causes output 0. The effect of the weights  $W_{ij}$  is to stretch or compact the axes of the hyperspace. The offset or threshold  $\theta_i$  corresponds to the minimum distance the partitioning hyperplane passes from the origin. By using nonlinear but continuous sigmoid units as activation functions various useful curves can be created for partitioning the hyperspace [33]. Further useful variations in partitions can be obtained by using multiple output nodes. In multi-layer feed-forward neural networks there are connections between the elements of successive layers. There are no interlayer connections.

### 2.1.2 Basic Properties of Neural Networks

Conventional signal processing techniques are algorithmic in nature. However, neural networks can perform non-algorithmic signal processing. Neural nets



provide high computation rates because, of their massively parallel nature. Provided that suitable hardware is available, neural networks can be implemented. Thus on-line operations may be possible. Neural nets provide a greater degree of robustness, fault tolerance or fail safety compared to classical sequential computing systems, because, the information is contained in the weighted wiring diagram in the distributed form with nodes having primarily local connections. Presentation of information often contains redundancies. Thus any damage in a few nodes or links or any invalid information contained in these does not impair overall performance significantly. Associative storage and retrieval of knowledge is possible in neural networks. Unlike other classical large scale dynamic systems, the uniform rate of convergence toward a steady state of neural networks is essentially independent of the number of neurons in the network [33].

At present there are also some problems associated with using or designing neural networks:

1. There is no general way of deciding about the network topology to perform a certain task.
2. The convergence of most of the important learning algorithms used in neural networks is not guaranteed.

It is also important to see that the necessary hardware is not presently ready to be used for taking full benefit of the high speed of neural computation.

The problem areas for which neural networks may provide some advantages can be pinpointed by considering the advantages and disadvantages of neural networks. The problems involving complexity, redundancy and speed can be solved more satisfactorily by using neural networks. So far results involving neural networks have been reported in the areas of learning, associative memory, decisions in optimization, adaptive pattern recognition, fuzzy sets, expert systems, adaptive filtering, numeric to symbolic conversion and control. It has already been demonstrated that the neural network based modeling for

forecasting the ionospheric critical frequency,  $foF2$  is a promising approach [34][26][35][36][37].

### 2.1.3 METU-IFS Implementation of Neural Networks

In METU-IFS application a multi layer feedforward type of neural network structure was implemented. Since Java programming language was used, an object oriented design came out naturally. All the work related to the neural network structure was collected in a single module to enable the maintenance to be done further ahead. By this module any type of feed forward neural network structure can be built from scratch to be trained with a selected training algorithm. Further details related to the design are given in Chapter 4.

Levenberg-Marquardt Algorithm is capable of dealing with huge size of input data. For networks having up to a few hundreds of weights it's a very efficient algorithm [38]. The same algorithm is also used to train METU-NN. Given all these considerations, Levenberg Marquardt algorithm was preferred for METU-IFS.

Since Levenberg-Marquardt algorithm [38] is a version of Backpropagation algorithm, it can be considered that Backpropagation is still employed in METU-IFS, although the implementation of Levenberg-Marquardt algorithm is the actual requirement. In the sub sections Backpropagation algorithm is first introduced, and then Levenberg-Marquardt algorithm described based on Backpropagation algorithm. The details of the implementation is not included here, the related theory, however, is discussed. For the implementation and design details see Chapter 4.

#### 2.1.3.1 Backpropagation Algorithm

Consider a multilayer feedforward network with  $p(i)$  as inputs,  $t(i)$  as target outputs,  $w^k(i, j)$  as weights, and  $b^k(i)$  as bias for unit  $i$  in layer  $k$ ,  $n^{k+1}(i)$  as net input to unit  $i$  in layer  $(k+1)$ , and  $a^{k+1}(i)$  as output of unit  $i$  as follows:

$$n^{k+1}(i) = \sum_{j=1}^{S_k} w^{k+1}(i, j).a^k(j) + b^{k+1}(i), \quad (2.1.5)$$

$$a^{k+1}(i) = f^{k+1}(n^{k+1}(i)) \quad (2.1.6)$$

The system equations in matrix form for an M layer network:

$$\underline{a}^0 = \underline{p}, \quad (2.1.7)$$

$$\underline{a}^{k+1} = \underline{f}^{k+1}(W^{k+1} \cdot \underline{a}^k + \underline{b}^{k+1}) \quad (2.1.8)$$

$$k=0, 1 \dots M-1$$

The performance index for the network is

$$V = \frac{1}{2} \sum_{q=1}^Q (t_q - a_q^M)^T (t_q - a_q^M) = \frac{1}{2} \sum_{q=1}^Q (\underline{e}_q^T \cdot \underline{e}_q) \quad (2.1.9)$$

where  $\underline{e}_q^M$  is the output of the network when the  $q^{\text{th}}$  input,  $\underline{p}_q$ , is applied, and  $\underline{e}_q$  is the error. An approximate steepest descent rule is employed for basic Backpropagation algorithm. To approximate the performance index ( $\bar{V}$ ), the total sum of squares is replaced by squared errors for a single input/output pair as

$$\bar{V} = \frac{1}{2} \underline{e}_q^T \underline{e}_q \quad (2.1.10)$$

The approximate steepest (gradient) descent algorithm then:

$$\Delta w^k(i, j) = -\alpha \frac{\partial \bar{V}}{\partial w^k(i, j)}, \quad (2.1.11)$$

$$\Delta b^k(i) = -\alpha \frac{\partial \bar{V}}{\partial b^k(i)} \quad (2.1.12)$$

where  $\alpha$  is the learning rate. The sensitivity of the performance index to changes in the net input of unit  $i$  in layer  $k$  is:

$$\delta^k \equiv \frac{\partial \bar{V}}{\partial n^k(i)}, \quad (2.1.13)$$

$$\frac{\partial \bar{V}}{\partial w^k(i, j)} = \frac{\partial \bar{V}}{\partial n^k(i)} \frac{\partial n^k(i)}{\partial w^k(i, j)} = \delta^k(i) \cdot a^{k-1}(j), \quad (2.1.14)$$

$$\frac{\partial \bar{V}}{\partial b^k(i)} = \frac{\partial \bar{V}}{\partial n^k(i)} \frac{\partial n^k(i)}{\partial b^k(i)} = \delta^k(i). \quad (2.1.15)$$

Sensitivities satisfy the following recurrence relation:

$$\underline{\delta}^k = \dot{F}^k(\underline{n}^k) \mathcal{W}^{k+1T} \underline{\delta}^{k+1}, \quad (2.1.16)$$

$$\dot{F}^k(\underline{n}^k) = \begin{bmatrix} \dot{f}^k(n(1)) & 0 & \dots & 0 \\ 0 & \dot{f}^k(n(2)) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \dot{f}^k(n^k(S_k)) \end{bmatrix}, \quad (2.1.17)$$

$$\dot{f}^k(n) = \frac{df^k(n)}{dn}. \quad (2.1.18)$$

The recurrence relation is initialized at the final layer as

$$\underline{\delta}^M = -\dot{F}^M(\underline{n}^M) (\underline{t}_q - \underline{a}_q). \quad (2.1.19)$$

The overall Backpropagation algorithm proceeds as follows; first propagate the input forward using (2.1.7), (2.1.8); next, propagate the sensitivities back using (2.1.19) and (2.1.16); and finally, update the weights and offsets using (2.1.11), (2.1.12), (2.1.14), and (2.1.15).

### 2.1.3.2 Levenberg Marquardt Modification

While Backpropagation is a steepest descent algorithm, the Levenberg-Marquardt algorithm is an approximation to Newton's method [38]. Newton's method to minimize a function  $V(\underline{x})$  with respect to the parameter vector  $\underline{x}$  is:

$$\Delta \underline{x} = -[\nabla^2 V(\underline{x})]^{-1} \nabla V(\underline{x}) \quad (2.1.20)$$

where  $\nabla^2 V(\underline{x})$  is the Hessian matrix and  $\nabla V(\underline{x})$  is the gradient. If we assume  $V(\underline{x})$  as a sum of squares function as

$$V(\underline{x}) = \sum_{i=1}^N e_i^2(\underline{x}), \quad (2.1.21)$$

$$\nabla V(\underline{x}) = J^T(\underline{x}) \underline{e}(\underline{x}), \quad (2.1.22)$$

$$\nabla^2 V(\underline{x}) = J^T(\underline{x}) J(\underline{x}) + S(\underline{x}), \quad (2.1.23)$$

where  $J(\underline{x})$  is the Jacobian matrix:

$$J(\underline{x}) = \begin{bmatrix} \frac{\partial e_1(\underline{x})}{\partial x_1} & \frac{\partial e_1(\underline{x})}{\partial x_2} & \dots & \frac{\partial e_1(\underline{x})}{\partial x_n} \\ \frac{\partial e_2(\underline{x})}{\partial x_1} & \frac{\partial e_2(\underline{x})}{\partial x_2} & \dots & \frac{\partial e_2(\underline{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_N(\underline{x})}{\partial x_1} & \frac{\partial e_N(\underline{x})}{\partial x_2} & \dots & \frac{\partial e_N(\underline{x})}{\partial x_n} \end{bmatrix}, \quad (2.1.24)$$

$$S(\underline{x}) = \sum_{i=1}^N e_i(\underline{x}) \nabla^2 e_i(\underline{x}). \quad (2.1.25)$$

For the Gauss-Newton method it is assumed that  $S(\underline{x}) \approx 0$  and the update (2.1.20) is:

$$\Delta \underline{x} = [J^T(\underline{x})J(\underline{x})]^{-1} J^T(\underline{x}) \underline{e}(\underline{x}) . \quad (2.1.26)$$

The Levenberg-Marquardt modification to the Gauss-Newton method is:

$$\Delta \underline{x} = [J^T(\underline{x})J(\underline{x}) + \mu I]^{-1} J^T(\underline{x}) \underline{e}(\underline{x}) . \quad (2.1.27)$$

The parameter  $\mu$  is multiplied by some factor ( $\beta$ ) whenever a step would result in an increased  $V(\underline{x})$ . When a step reduces  $V(\underline{x})$ ,  $\mu$  is divided by  $\beta$ . When  $\mu$  is large the algorithm becomes steepest descent with step  $1/\mu$ , while for small  $\mu$  the algorithm becomes Gauss-Newton. For the neural network mapping problem the terms in the Jacobian matrix can be computed by a simple modification to the Backpropagation algorithm. The performance index for the mapping is given by (2.1.9). This is equivalent in form to (2.1.21). For the elements in the Jacobian matrix that are needed to calculate terms like:

$$\frac{\partial e_q(m)}{\partial w^k(i, j)} , \quad (2.1.28)$$

These terms can be calculated using the standard Backpropagation algorithm with one modification in the final layer as

$$\Delta^M = -\dot{F}^M(\underline{n}^M) . \quad (2.1.29)$$

Each column of the matrix in (2.1.29) is a sensitivity vector that must be backpropagated through the network to produce one row of the Jacobian.

The Levenberg-Marquardt modification to the Backpropagation algorithm thus proceeds as [38]:

- 1) Present all inputs to the network and compute the corresponding network outputs (using (2.1.7) and (2.1.8), and errors  $(e_q = t_q - a_q^M)$ . Compute the sum of squares of errors over all inputs  $(V(\underline{x}))$ .
- 2) Compute the Jacobian matrix (using (2.1.29), (2.1.16), (2.1.14), (2.1.15), and (2.1.24)).
- 3) Solve (2.1.27) to obtain  $\Delta \underline{x}$ .
- 4) Compute the sum of squares of errors again using  $\underline{x} + \Delta \underline{x}$ . If this new sum of squares is smaller than that computed in step 1, then reduce  $\mu$  by  $\beta$ , let  $\underline{x} = \underline{x} + \Delta \underline{x}$  and go back to step 1. If the sum of the squares is not reduced, then increase  $\mu$  by  $\beta$  and go back to step 3.
- 5) The algorithm is assumed to have converged when the norm of the gradient (2.1.22) is less than some predetermined value, or when the sum of squares has been reduced to some error goal.

Newton's method is faster and more accurate near an error minimum, so the aim is to shift towards Newton's method as quickly as possible. Thus,  $\mu$  is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function will always be reduced at each iteration of the algorithm.

The construction work of the neural network based model was carried out in two modes, namely the "development mode" and "the operation mode". The development mode is composed of "training or learning phase" and "test phase". Not to hurt generalization independent data sets are used for the phases of training, test and validation.

As the training advances, the training error starts to decrease and eventually, goes to zero that corresponds to the memorization which means the loss of generalization capability of the neural network. To prevent memorization the training should be interrupted when the training error falls while the validation

error tends to rise. In METU-IFS application, a safety lag of response is employed. METU-IFS starts to count number of epochs when this condition first occurs and interrupts the training after five successive repeats.

The following error cross correlation coefficient definitions, are used in the operation mode to measure the performance of the models:

i. Root Mean Square Error:

$$RMS\_Error = \sqrt{\frac{\sum_{i=1}^N [f_i - o_i]^2}{N}} , \quad (2.1.30)$$

ii. Normalized Error:

$$N\_Error = \frac{\sum_{i=1}^N (f_i - o_i)}{N} , \quad (2.1.31)$$

iii. Absolute Error:

$$Abs\_Error = \frac{\sum_{i=1}^N (f_i - o_i)}{N} , \quad (2.1.32)$$

where  $i$ : Forecast time order,  $f_i$ : Forecast foF2 value at time  $i$ ,  $o_i$ : Observed foF2 value at time  $i$ , and  $N$ : Total number of forecast or observed foF2 instants,

iv. Cross Correlation Coefficient of forecast and observed foF2 values ( $r_{fo}$ ):

$$r_{fo} = \frac{C(f,o)}{\sqrt{C(f,f).C(o,o)}} , \quad (2.1.33)$$



where  $C$ :The covariance,  $f$ :Forecast  $foF2$  values, and  $o$ :Observed  $foF2$  values.

#### **2.1.4 Tests and Verification of METU-IFS Implementation of METUNN Models**

Implementation of METU-NN model is tested and verified by comparing it with the results obtained from MATLAB code. The same neural network architecture with 8 inputs, 1 hidden layer including 4 neurons, is built in both METU-IFS side and MATLAB side. As it is determined in METU-NN model, the activation function of hidden layer and the output layer is selected as "Tansig" and "Linear" respectively. Both applications are driven with normalized TEC data sets.

Firstly, the results of the two different trainings and the two different operations, which are all performed by METU-IFS, are demonstrated. Both training processes are repeatedly driven with the same TEC data, but one of them with a validation stop, the other without.

The report taken from METU-IFS after training is shown in Table 2.1-1. Training conditions and results are listed in this report.

Table 2.1-1: Training Results Report – 1.

|                                 |              |
|---------------------------------|--------------|
| Trained with ValSet_1           |              |
| Operated with TrSet_1           |              |
| *** Training Results ***        |              |
| Sun Jul 25 17:48:29 EEST 2004   |              |
|                                 |              |
| Training Algorithm :            | LMO          |
| Number Of Inputs :              | 8            |
| Number Of Neurons in HLayer1 :  | 4            |
| Number Of Outputs :             | 1            |
| Act Func For Hidden Layers :    | Tansig       |
| Act Func For Output Layer :     | Linear       |
| Bias Ex(0)/Inc(1) :             | 0            |
| Initial Momentum :              | 0.01         |
| Scale Factor (Beta) :           | 10           |
| Target Error :                  | 1.00E-05     |
| Final MSE :                     | 6.49E-04     |
| Final Epoch No :                | 75           |
| Average Training Duration :     | 4706ms/epoch |
| <b>Performance</b>              |              |
| <b>Max Error=</b>               | 0.162666     |
| <b>Min Error=</b>               | -0.16129     |
| <b>Max SE=</b>                  | 0.02646      |
| <b>Min SE=</b>                  | 0            |
| <b>SSE=</b>                     | 4.669275     |
| <b>MSE=</b>                     | 0.000543     |
| <b>RMSE=</b>                    | 0.023296     |
| *** End Of Training Results *** |              |

After a successful training, the network is operated with a different set of TEC values. A variation graph, in which the target and the obtained values are plotted on the same diagram, is in Figure 2.1-2. A scatter diagram is plotted in Figure 2.1-3.

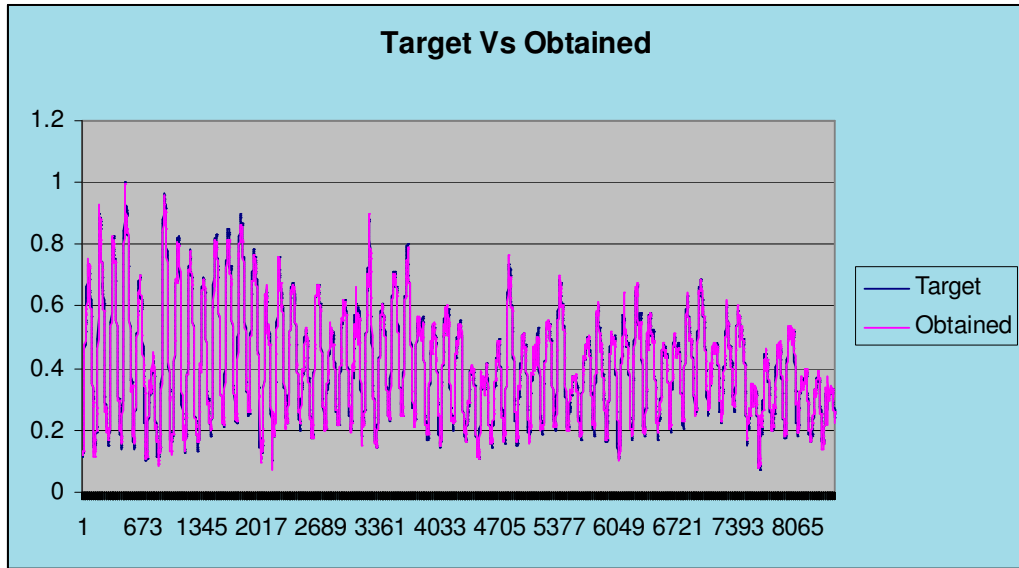


Figure 2.1-2: Variations graph for test – 1.

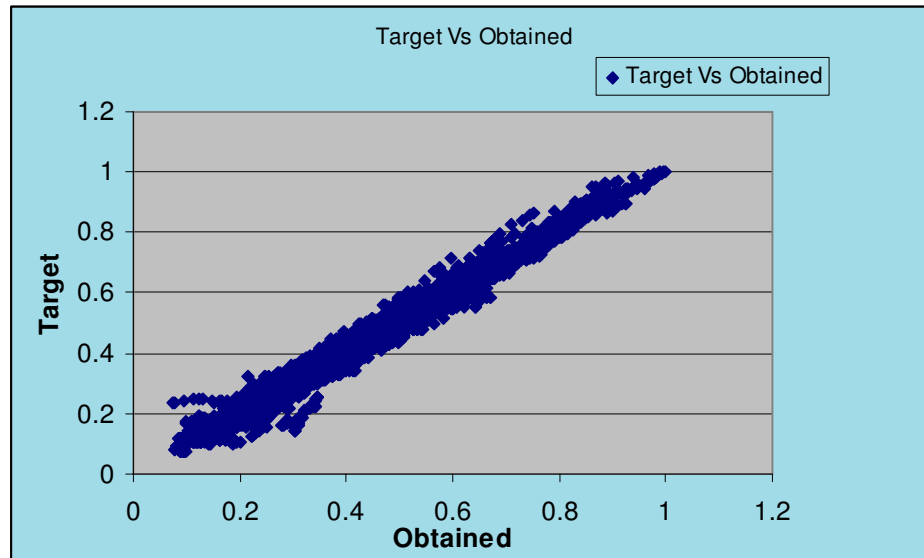


Figure 2.1-3: Scatter diagram for test – 1.

The report of the training results for the second test is shown in Table 2.1-2. In this test, the operation set is also used as the validation set. The variations graph and the scatter diagram for this test are plotted respectively in Figure 2.1-4 and in Figure 2.1-5.

Table 2.1-2: Training Results Report - 2

|                                 |              |
|---------------------------------|--------------|
| Trained with TrSet_1            |              |
| Validated with Val_set_1        |              |
| Operated with Val_set_1         |              |
|                                 |              |
| *** Training Results ***        |              |
| Mon Jul 26 17:26:22 EEST 2004   |              |
|                                 |              |
| Training Algorithm :            | LMO          |
| Number Of Inputs :              | 8            |
| Number Of Neurons in HLayer1 :  | 4            |
| Number Of Outputs :             | 1            |
| Act Func For Hidden Layers :    | Tansig       |
| Act Func For Output Layer :     | Linear       |
| Bias Ex (0)/Inc(1) :            | 0            |
| Initial Momentum :              | 0.01         |
| Scale Factor (Beta) :           | 10           |
| Target Error :                  | 1.00E-04     |
| Final MSE :                     | 5.61E-04     |
| Final Epoch No :                | 10           |
| Average Training Duration :     | 5573ms/epoch |
| <b>Performance</b>              |              |
| <b>Max Error=</b>               | 0.172181704  |
| <b>Min Error=</b>               | -0.353880184 |
| <b>Max SE=</b>                  | 0.125231185  |
| <b>Min SE=</b>                  | 8.61902E-13  |
| <b>SSE=</b>                     | 6.398002964  |
| <b>MSE=</b>                     | 0.000743608  |
| <b>RMSE=</b>                    | 0.027269176  |
| *** End Of Training Results *** |              |

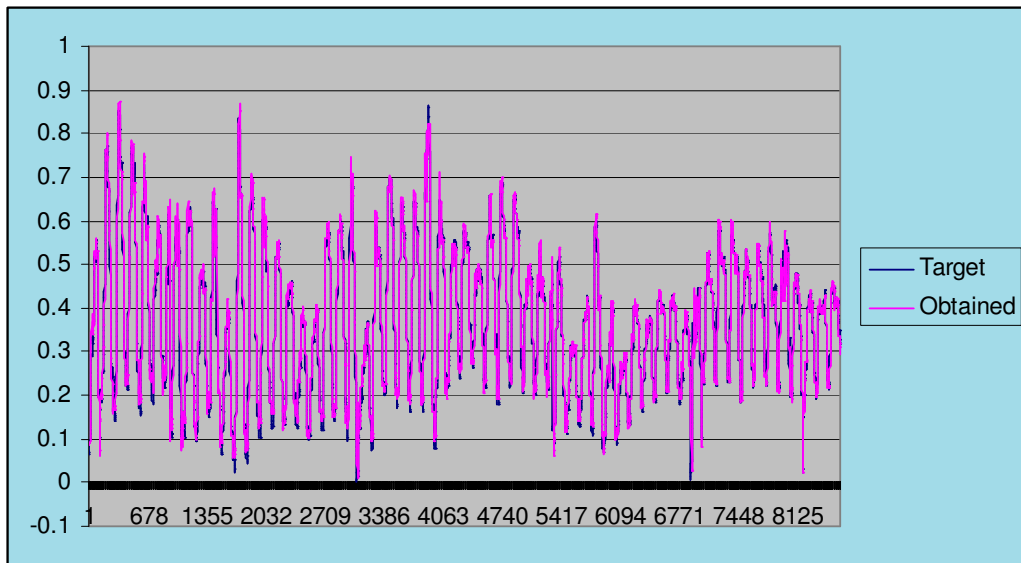


Figure 2.1-4: Variations graph for METU-IFS – 2.

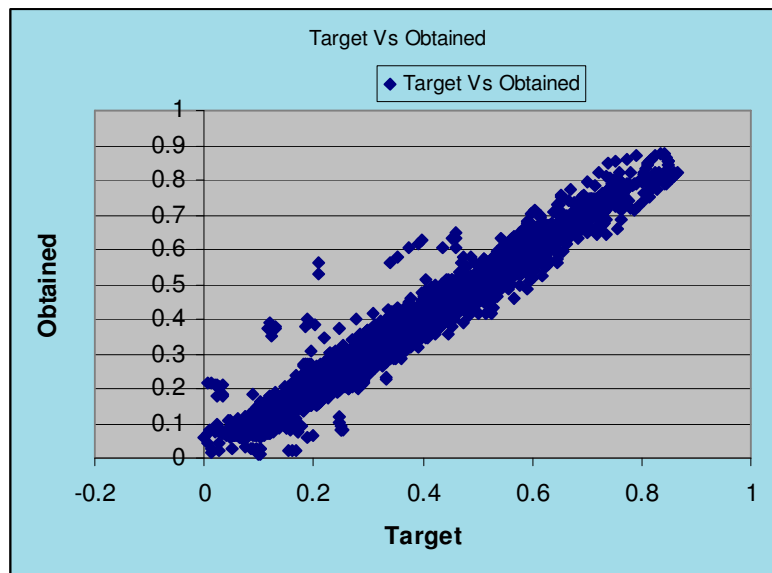


Figure 2.1-5: Scatter diagram – 2.

The same two sets of TEC data, one set for training and the other set for operation are used in both METU-IFS and MATLAB to compare the results obtained by METU-IFS with those obtained by MATLAB. All the other conditions, related to the training parameters and the neural network structure, are also

kept constant. Since the same model and same training algorithm is used, it is expected to obtain the same results with a little difference which may be caused by implementing different programs.

After the training processes, MATLAB has reached a RMSE value of 0.02917 while METU-IFS a value of 0.02727. These values are close to each other as expected. However, it should also be noted that METU-IFS gives a better result.

Below in Figure 2.1-6 and Figure 2.1-7, for MATLAB and METU-IFS respectively, plotted are the scatter diagrams obtained by operating both programs with the same data set. Again the diagrams look similar as expected.

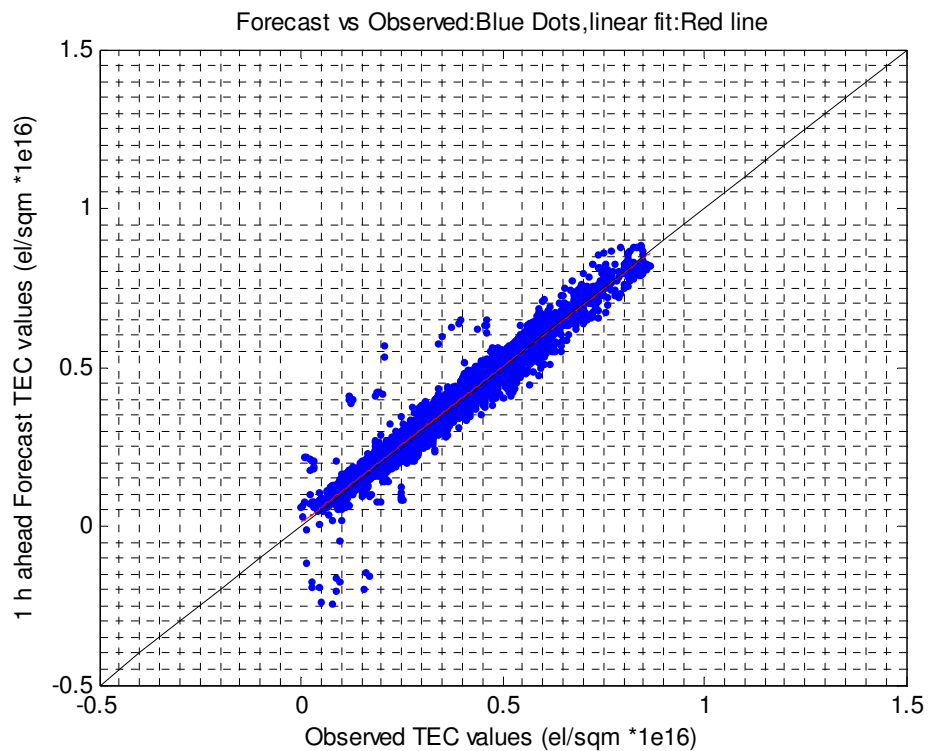


Figure 2.1-6: Scatter diagram; observed values are obtained by MATLAB NN toolbox.

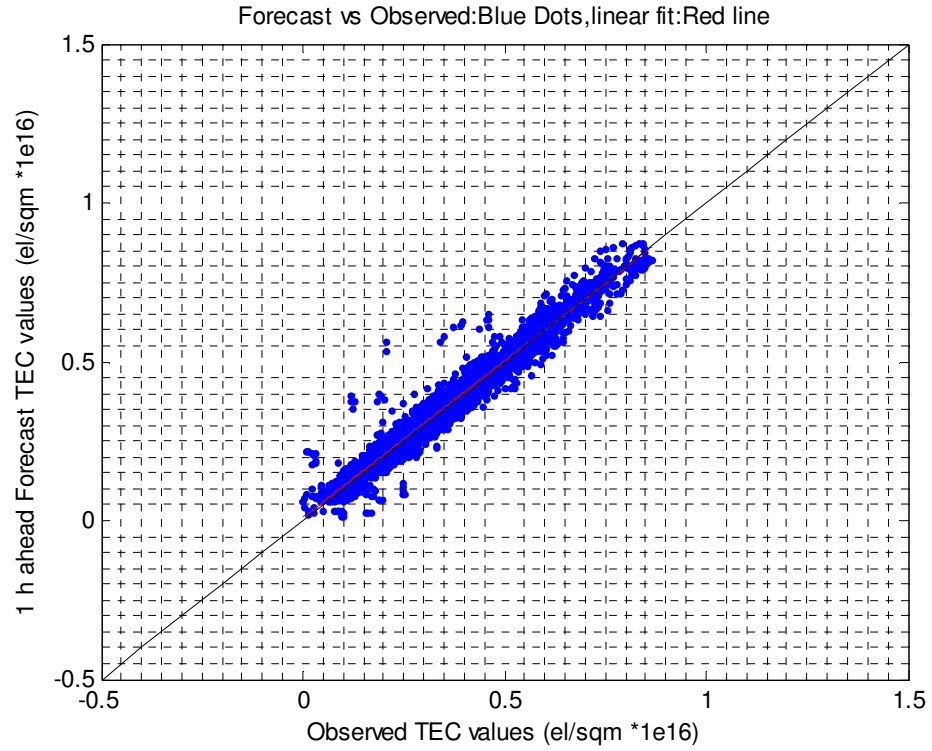


Figure 2.1-7: Scatter diagram; observed values are obtained by METU-IFS.

The similar results appear on the variations graphs. In Figure 2.1-8 and Figure 2.1-9 plotted are the variation graphs for MATLAB and METU-IFS respectively.

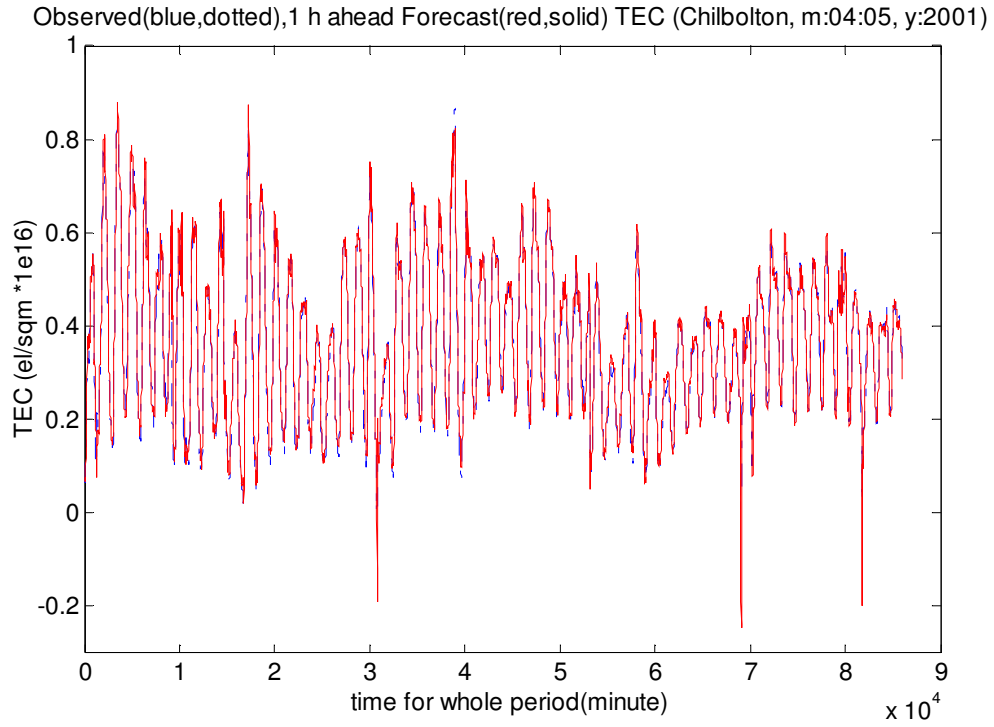


Figure 2.1-8: Variations graph obtained by MATLAB NN toolbox.

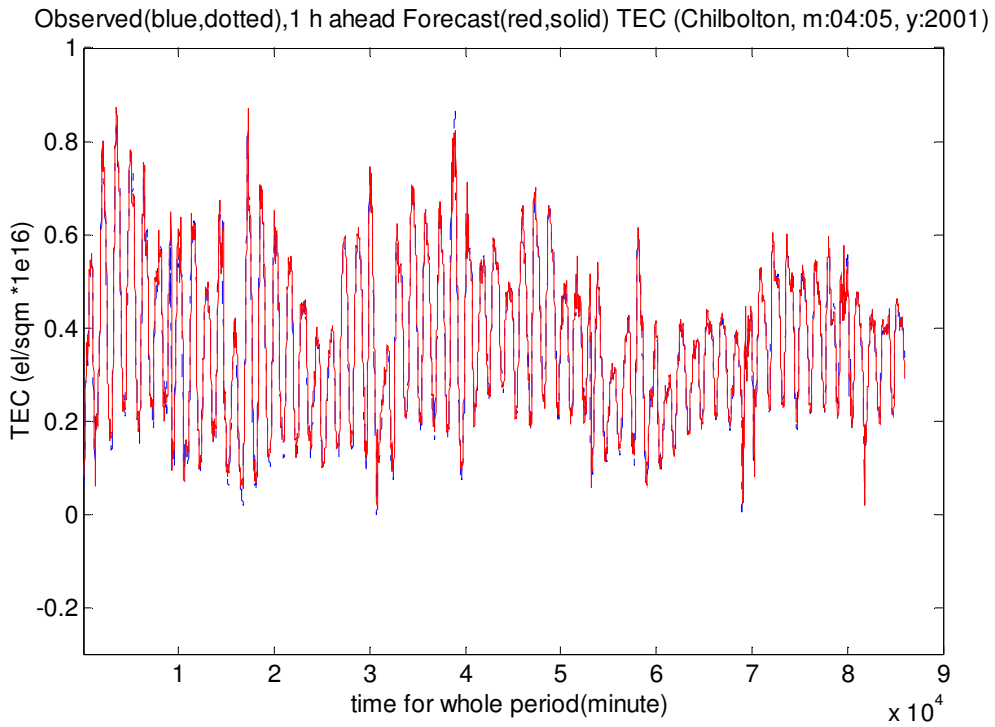


Figure 2.1-9: Variations graph obtained by METU-IFS.



Table 2.1-3 tabulates a summary of this comparison. This table gives the RMSEs, obtained after the training, as well as the cross correlation coefficients of forecasted and observed TEC values, calculated after the operation. This test shows that the results obtained by the implementation of training algorithms and METU-NN model is not worse than those by the implementation of MATLAB, on the contrary, they look even better.

Table 2.1-3: Comparison of the training and operation results of MATLAB and METU-IFS.

|   | <b>MATLAB</b> | <b>METU-IFS</b> |
|---|---------------|-----------------|
| The root mean square error for 1h ahead forecasting by the net for whole time period: (eI/sqm *10 <sup>16</sup> ) | 0.02917       | 0.02727         |
| Absolute error: (eI/sqm *10 <sup>16</sup> )   | 0.01837       | 0.01838         |
| Cross correlation coefficient of Forecast and Observed TEC values:  | 0.98399       | 0.98613         |

## **CHAPTER 3**

### **NEUROFUZZY METHODS**

This chapter discusses the steps in developing a neurofuzzy model. It gives the general description of neurofuzzy concept as an introduction and then points out the advantages and disadvantages of neurofuzzy methods over neural network methods. In the literature there are many neurofuzzy systems developed to model a range of problems in a variety of fields, some of which are reviewed in this chapter. Finally METU-NFN model is compared with METU-NN models developed before.

#### **3.1 General Description of Neurofuzzy Networks**

Both neural networks and fuzzy systems are motivated by imitating human reasoning processes. Neural Networks have been used extensively in modeling real problems with nonlinear characteristics. In most of these applications, feed-forward networks that are trained with the back-propagation algorithm have been used. The main advantages of using NNs are their flexibility and ability to model nonlinear relationships. In contrast to knowledge-based techniques, no explicit knowledge is needed for the application of NNs. However, despite these advantages, NNs have often been criticized for acting as “black boxes”. The knowledge contained in an NN model is kept in the form of a weight matrix that is hard to interpret and can be misleading at times. In NNs, the relations are not explicitly given, but are ‘coded’ in the network and its parameters whereas in fuzzy systems, relationships are represented explicitly in the form of if-then rules. The efficiency of NN models is also another point of concern. Since it is not always possible to determine the significance of the input variables in advance,

any potential candidate may be included in the model. This is particularly important in cases where a large number of potential input variables exist but only a subset of them would actually affect the output. It is therefore important to identify and exclude those input variables that do not have a significant contribution. This would lead to a more efficient model. It is also beneficial to have the ability to insert any available knowledge or expertise into the model if necessary.

One way to overcome many of these shortcomings is to use neurofuzzy models. Neurofuzzy systems combine the semantic transparency of rule-based fuzzy systems with the learning capability of neural networks. In other words, these models combine the transparent, linguistic representation of a fuzzy system with the learning ability of NNs. Therefore, they can be trained to perform an input/output mapping, just as with an NN, but with the additional benefit of being able to provide the set of rules on which the model is based. This gives further insight into the process being modeled [28]. Several merger types of NNs and fuzzy systems have been reported in the literature. They include various representations and architectures and therefore are suitable for different applications (Nauck *et al.* 1997). The most widely used neurofuzzy models are adaptive network based fuzzy inference systems (ANFIS) and B-spline associative memory networks (AMNs) [39].

### **3.2 Fuzzy Inference Systems (FIS)**

For a better understanding of neurofuzzy systems we need to comprehend FIS. Because FIS is an important element in a neurofuzzy system, which is to be used to improve ionospheric forecasting studies. So, at least a brief discussion on FIS seems to be a necessary first step before going into details of outlining a neurofuzzy model.

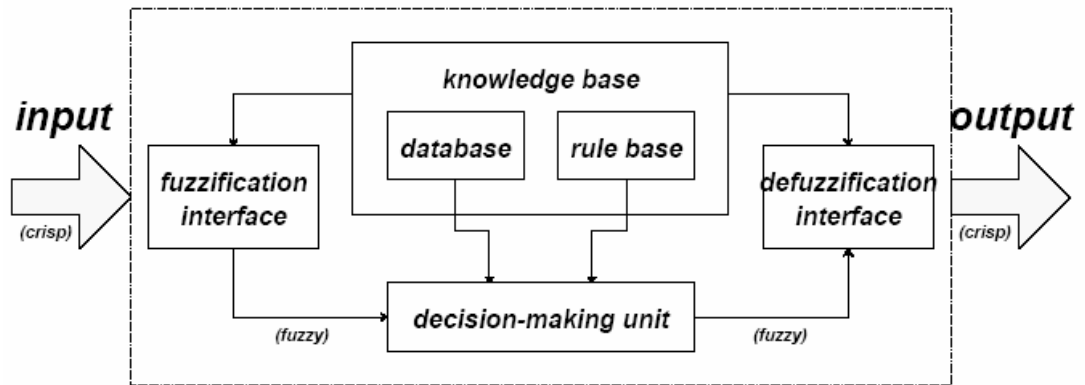


Figure 3.2-1: Fuzzy inference system.

Fuzzy inference systems are also known as fuzzy-rule-based systems, fuzzy models, fuzzy associative memories (FAM), or fuzzy controllers when used as controllers. Basically a fuzzy inference system is composed of five functional blocks (Figure 3.2-1) [40]:

- a rule base containing a number of fuzzy if-then rules;
- a database which defines the membership functions of the fuzzy sets used in the fuzzy rules;
- a decision-making unit which performs the inference operations on the rules;
- a fuzzification interface which transforms the crisp inputs into degrees of match with linguistic values;
- a defuzzification interface which transform the fuzzy results of the inference into a crisp output.

Usually, the rule base and the database are jointly referred to as the knowledge base.

Several types of fuzzy reasoning [41][42] have been proposed in the literature. Depending on the types of fuzzy reasoning and fuzzy if-then rules employed, most fuzzy inference systems can be classified into three types (Figure 3.2-2):

**Type 1:** The overall output is the weighted average of each rule's crisp output induced by the rule's firing strength (the product or minimum of the degrees of match with the premise part) and output membership functions. The output membership functions used in this scheme must be monotonically non-decreasing [43].

**Type 2:** The overall fuzzy output is derived by applying "max" operation to the qualified fuzzy outputs (each of which is equal to the minimum of firing strength and the output membership function of each rule). Various schemes have been proposed to choose the final crisp output based on the overall fuzzy output; some of them are center of area, bisector of area, mean of maxima, maximum criterion, etc [41][42].

**Type 3:** Takagi and Sugeno's fuzzy if-then rules are used [44]. The output of each rule is a linear combination of input variables plus a constant term, and the final output is the weighted average of each rule's output.

Figure 3.2-1 utilizes a two-rule two-input fuzzy inference system to show different types of fuzzy rules and fuzzy reasoning mentioned above. Be aware that most of the differences lie in the specification of the consequent part (monotonically non-decreasing or bell-shaped membership functions, or crisp function) and thus the defuzzification schemes (weighted average, centroid of area, etc) are also different.

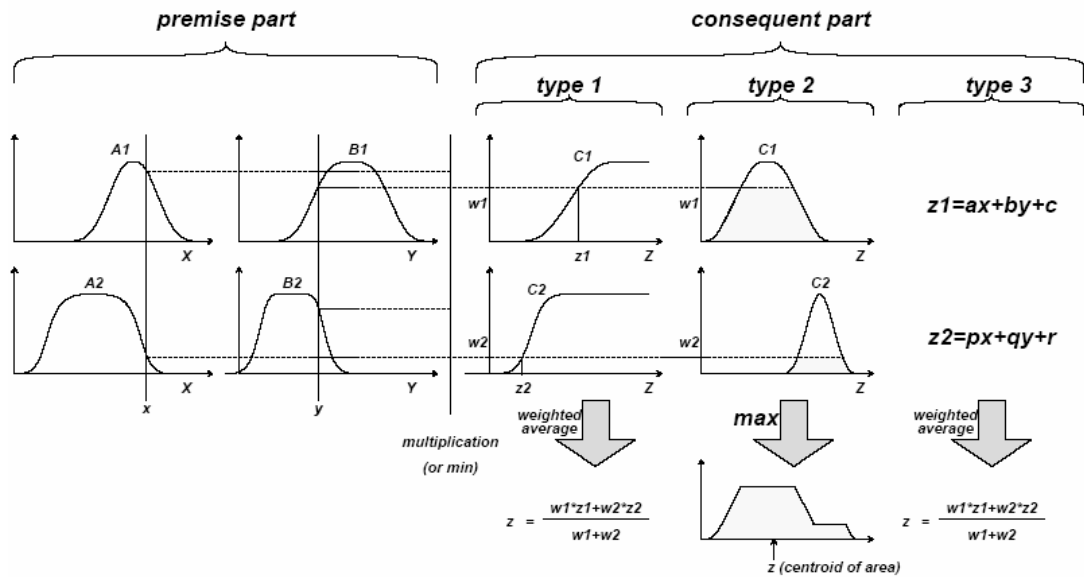


Figure 3.2-2: Commonly used fuzzy if-then rules and fuzzy reasoning mechanisms.

### 3.3 Combining Advantages of Both Fuzzy Logic and Neural Networks Methods

Multilayer feedforward neural networks with an arbitrary large number of units in the hidden layer can approximate any real continuous function [45]. This good training ability is always considered for modeling nonlinear functions. The back propagation (BP) algorithm allows multilayer feedforward neural networks to learn input-output pairs from training samples. However, it is possessed of the disadvantage of slow convergence and the disadvantage of being a complete black box model. Many researches combine fuzzy set theory with neural network configurations for improving the disadvantages.

Neurofuzzy (NF) computing is a popular framework for solving complex problems. If we have knowledge expressed in linguistic rules, we can build a FIS, and if we have data, or can learn from a simulation (training) then we can use NNs. For building a FIS, we have to specify the fuzzy sets, fuzzy operators and the knowledge base. Similarly for constructing a NN for an application the user needs to specify the architecture and learning algorithm. An analysis reveals that

the drawbacks pertaining to these approaches seem complementary and therefore it is natural to consider building an integrated system combining the concepts. While the learning capability is an advantage from the viewpoint of FIS, the formation of linguistic rule base will be advantage from the viewpoint of NNs [46].

### **3.4 Neurofuzzy Systems in Literature**

Neurofuzzy systems are simply developed by the way of applying a learning algorithm to a fuzzy system. A common way to apply a learning algorithm to a fuzzy system is to represent it in a special NN like architecture. However the conventional NN learning algorithms (gradient descent) cannot be applied directly to such a system as the functions used in the inference process are usually non differentiable. This problem can be overcome by using differentiable functions in the inference system. FIS has potential to easily apply a learning rule. As a consequence of attraction of researchers many models developed. Some of the major works in this area are GARIC [47], FALCON [48], ANFIS [49], NEFCON [48], FUN [33], SONFIN [50], FINEST [51], EFuNN [52], dmEFuNN [52], evolutionary design of neurofuzzy systems [53], and many others [46].

And another attention attractive type of neurofuzzy system in literature is ASMOD (Adaptive Spline Modeling of Observation Data) which is structurally different from the ones listed above [54].

### **3.5 ANFIS**

Adaptive Network based Fuzzy Inference System (ANFIS) is a FIS based neurofuzzy system. Other systems similar to ANFIS are listed in the previous section. Among them ANFIS is selected for applying to ionospheric forecasting problems and is one of the candidates for METU NF model.

ANFIS was first proposed by Jang (1993) [40]. A basic ANFIS is shown in Figure 3.5-1. Functionally, there are almost no constraints on the node functions of ANFIS except piecewise differentiability. Structurally, the only limitation of

network configuration is that it should be of feedforward type. These are the general restrictions, coming from Backpropagation learning rule.

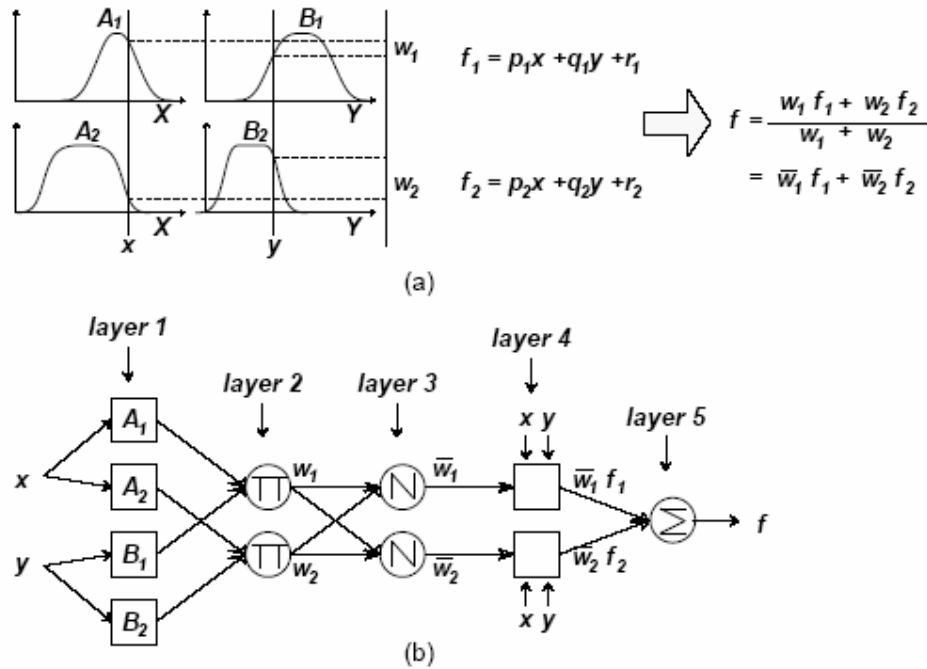


Figure 3.5-1: (a) Takagi-Sugeno fuzzy reasoning; (b) ANFIS structure.

For the sake of simplicity, it's assumed that the fuzzy inference system under consideration has two inputs  $x$  and  $y$  and one output  $z$ . And it's supposed that the rule base contains two fuzzy if then rules of Takagi-Sugeno type [44].

$$\text{Rule 1: If } x \text{ is } A_1 \text{ and } y \text{ is } B_1 \text{ then } f_1 = p_1x + q_1y + r_1 \quad (3.5.1)$$

$$\text{Rule 2: If } x \text{ is } A_2 \text{ and } y \text{ is } B_2 \text{ then } f_2 = p_2x + q_2y + r_2 \quad (3.5.2)$$

The fuzzy reasoning and the corresponding equivalent ANFIS architecture are shown respectively in Figure 3.5-1 (a) and Figure 3.5-1 (b).

### Layer 1

Every node  $i$  in this layer is an adaptive node, representing membership functions described by generalized bell functions, e.g.



$$O_{1,i} = \mu_1(x) = \frac{1}{1 + |(x - c_1)/a_1|^{2b_1}}, \quad (3.5.3)$$

where  $X$  is input to the node and  $a_1$ ,  $b_1$ , and  $c_1$  are adaptable variables known as premise parameters. The outputs of this layer are the membership values of the premise part.

### Layer 2

Every node in this layer is a fixed node with the task of multiplying incoming signals and sending the product out. This product represents the firing strength of a rule. For example, in Figure 3.5-1 (b)

$$O_{2,1} = w_1 = \mu_1(x)\mu_4(y), \quad (3.5.4)$$

### Layer 3

Every node in this layer is a fixed node which calculates the ratio of the  $i$ -th rules firing strength to the sum of all rules' firing strengths

$$O_{3,1} = \bar{w}_1 = \frac{w_1}{w_1 + w_2 + w_3 + w_4}, \quad (3.5.5)$$

### Layer 4

Nodes of this layer are all adaptive with node functions

$$O_{4,1} = \bar{w}_1 f_1 = \bar{w}_1 (p_1 x + q_1 y + r_1), \quad (3.5.6)$$

where  $w_1$  = output of Layer 3 and  $\{p_i, q_i, r_i\}$  parameter set. Parameters of this layer are referred to as consequent parameters.

### Layer 5

The single fixed node of this layer computes the final output as the summation of all incoming signals.

A two-step process may be used for the learning or adjustment of the network parameters. This process is known as hybrid learning [40]. In the first step, the premise parameters are kept fixed and the information is propagated forward in the network to Layer 4, where the consequent parameters are identified by a least-squares estimator. In the second step, the backward pass, the consequent parameters are held fixed while the error is propagated, and the premise parameters are modified using gradient descent. The only user-specified information is the number of membership functions for each input and, the input-output training information.

### **3.6 ASMOD**

ASMOD is the other type of neurofuzzy system in the literature. It was first proposed by Kavli on June 1993 [54]. ASMOD is an acronym for Adaptive Spline Modeling of Observation Data. ASMOD is based on Associative Memory Networks (AMNs). AMNs have the ability to approximate any continuous function, given sufficient degrees of freedom [55]. A set of multidimensional overlapping basis functions covers the input space of AMNs (Figure 3.6-1). The size, shape, and overlap of the basis functions determine the model structure and the complexity. The basis functions can take a number of forms, including B-spline and Gaussian functions. It has been shown that B-spline AMNs and certain types of fuzzy models are learning equivalent [55]. Consequently, B-spline AMNs are a particular type of neurofuzzy model. This model is also valuable to be applied to ionospheric forecasting problems.

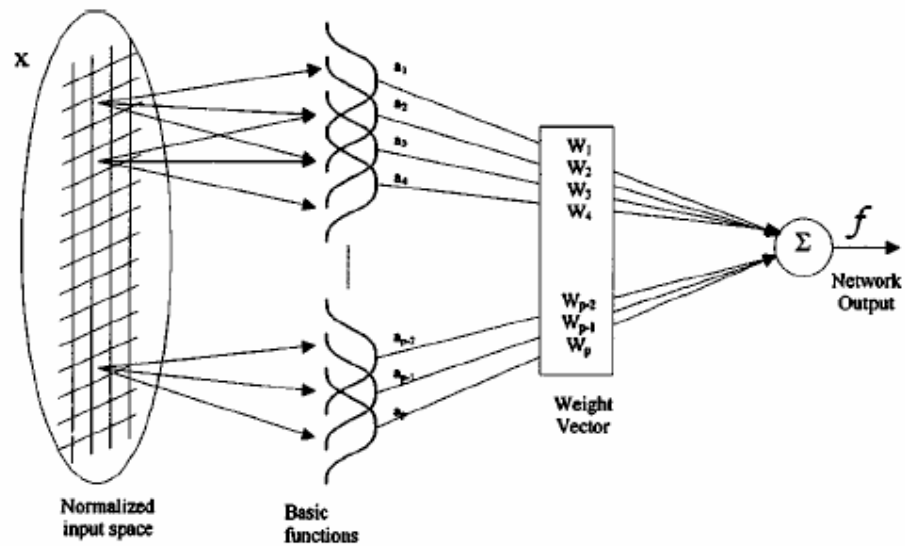


Figure 3.6-1: Typical B-spline associative memory network [55].

B-spline functions (Figure 3.6-2) are piecewise polynomials of order  $k$  which have been widely used in surface fitting applications. They have also been used to represent fuzzy membership functions as they have several desirable properties [55].

- The degree of membership can be evaluated using simple, fast, and stable recurrence relationship.
- The output of the function is nonzero in only a small part of the input space which means that knowledge is stored locally across only a small number of basis functions.
- The basis functions form a partition of unity as:  $\sum_{i=1}^p u_{A_i}(x) \equiv 1$  thus producing accurate smooth approximation.

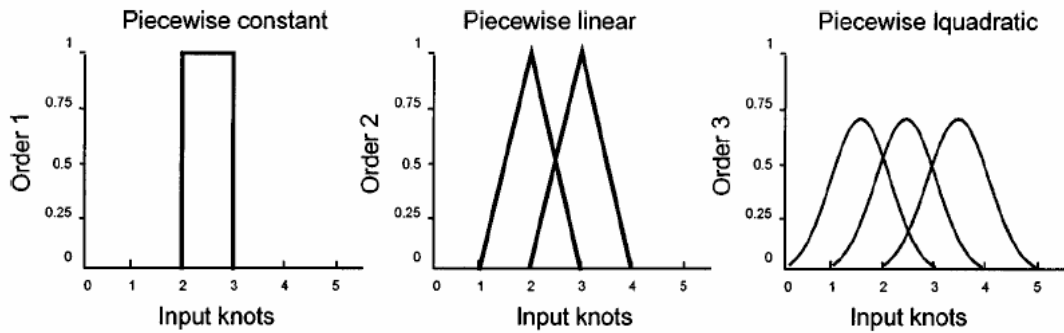


Figure 3.6-2: B-spline fuzzy membership functions.

The order of the B-spline functions determines the smoothness. They can be used to implement crisp fuzzy sets ( $k=1$ ) or the standard triangular fuzzy membership functions ( $k=2$ ) or other smoother representations. A univariate B-spline function of order  $k$  is nonzero only over  $k$  intervals which are generated by a  $(k+1)$  knots. A multivariate B-spline function can be formed by taking the tensor product of  $n$  univariate functions [55].

The output of a B-spline AMN can be represented by the following equation:

$$f = \sum_{i=1}^p a_i w_i = a(x)^T w \quad (3.6.1)$$

where  $f$ =output;  $a(x)=[a_1(x), \dots, a_p(x)]$ =vector of basis function outputs when presented by input  $x=(x_1, \dots, x_n)$ ; and  $w=(w_1, \dots, w_p)$ =vector of network weights. The power behind this type of network comes from the direct equivalence between the unions of a set of fuzzy rules in the following form:

$$\text{IF } (x \text{ is } A_i) \text{ THEN } (y \text{ is } B_j) c_{ij} \quad (3.6.2)$$

and the weighted sum of the multidimensional fuzzy input membership functions given in Eq.(3.6.1). Where  $(x \text{ is } A_i)$  and  $(y \text{ is } B_j)$ =linguistic expressions for the input and output respectively and  $c_{ij}$  =rule confidence which relates the  $i$ -th fuzzy input set to the  $j$ -th fuzzy output set. In other words, rule confidence indicates the degree to which the above rule has contributed to the output. This means

that a weight can be fuzzified to produce a rule confidence vector  $c_i$  which can then be defuzzified to produce the original weight. This relationship allows the network to be trained in a weight space leading to considerable reduction in computational cost (the network's output is linearly dependent on the weight set), while explaining the output with linguistic rules and the associated rule confidences.

Various training algorithms such as the Backpropagation can be applied to ASMOD model. The output might be compared with the actual measured output to obtain a correction error. Using this error and implementing a learning rule the neurofuzzy network adjusts its weights and determines its fuzzy parameters (i.e. fuzzy sets and fuzzy rules). In literature The Least Mean Squared (LMS) and the Normalized Least Mean Squared (NLMS) learning algorithms are generally used to update the weights [55]. As part of these algorithms, Eq. 3.6.3, and Eq. 3.6.4, respectively, can be used to adjust the weights for the LMS and NLMS algorithms [56]:

$$w_i(t) = w_i(t-1) + \eta(\hat{y}(t) - y(t))a_i(t) \quad (3.6.3)$$

$$w_i(t) = w_i(t-1) + \eta\left(\frac{\hat{y}(t) - y(t)}{\|a_i(t)\|_2^2}\right)a_i(t) \quad (3.6.4)$$

where  $\eta$  is learning rate; and  $\hat{y}$  is desired output.

As mentioned on the previous sections the major feature of a neurofuzzy network is that expert knowledge can be incorporated into the model. In the case of ASMOD, existing engineering knowledge can be incorporated into the trained network to optimize model performance and to enhance the interpretation of a constructed model by optimizing the membership functions. Optimization of membership functions can also be done at the beginning of the training to help training process but this may not help to decrease final error.

### **3.7 Curse of Dimensionality**

One major disadvantage of neurofuzzy networks is that the number of potential fuzzy rules is exponentially dependent on the dimension of the input space. This is often referred to as the "curse of dimensionality" [55]. This exponential growth of fuzzy rules with a number of inputs makes it impractical to use most existing neurofuzzy architectures for problems of high dimensionality. To illustrate this "curse," one may consider a fuzzy system with  $N$  input variables each of which having  $M$  membership functions. In such a system, as many as,  $M^N$  combinations (potential fuzzy rules) would exist. To overcome the curse of dimensionality neurofuzzy models usually employ some sort of a dimension reduction technique.

#### **3.7.1 Overcoming Curse of Dimensionality: ANFIS**

Jang (1996) proposed a quick and straightforward way to do input selection for ANFIS modeling. According to this method, the designer should establish different ANFIS models using a pool of candidate inputs and select the best one (the one with the smallest root mean squared error). Jang (1996) argues that since the least-squares method is the main drive behind the training and gradient descent contributes to the tuning of membership functions, therefore ANFIS can usually generate good results after one epoch of training. For example, using this approach, if we have a modeling problem with ten candidate inputs and we need to find the most influential three inputs to ANFIS, we can construct  $C_{3}^{10} = 5120$  ANFIS models and train them with a single pass of the least-squares method. The ANFIS model with the smallest training error is then chosen for further training using the hybrid learning rule to tune the membership functions as well [57][39]. However, it is obvious that this approach can be exhaustive for high-dimensionality problems and it is not guaranteed to produce optimal results.

#### **3.7.2 Overcoming Curse of Dimensionality: ASMOD**

As described earlier, the number of potential fuzzy rules in a neurofuzzy system is an exponential function of the dimension of the input space. In reality, many of these rules would be redundant for modeling purposes, and therefore a suitable technique should start from a simple architecture and build on it as necessary.

One such approach to reduce the dimensionality of B-spline AMNs is the analysis of variance [56] for decomposition of the output function of dimension  $n$  as

$$f(x) = f_0 + \sum_{i=1}^n f_i(x_i) + \sum_{i=1}^{n-1} \sum_{j=i+1}^n f_{i,j}(x_i, x_j) + \dots + f_{1,2,\dots,n}(x) \quad (3.7.2.1)$$

in which  $f_0$  is constant (the function bias) and other terms are univariate, bivariate, and other sub functions. In many instances, the majority of the higher-order terms are negligible and the input/output mapping can be approximated using a limited number of subnetworks of reduced dimensions. An example of such an additive decomposition is shown in Figure 3.7-1, where a five dimensional function is decomposed into a one-dimensional and two two-dimensional subnetworks. It should be noted that each of these subnetworks represents a separate AMN, the output of which are summed to produce the overall model output.

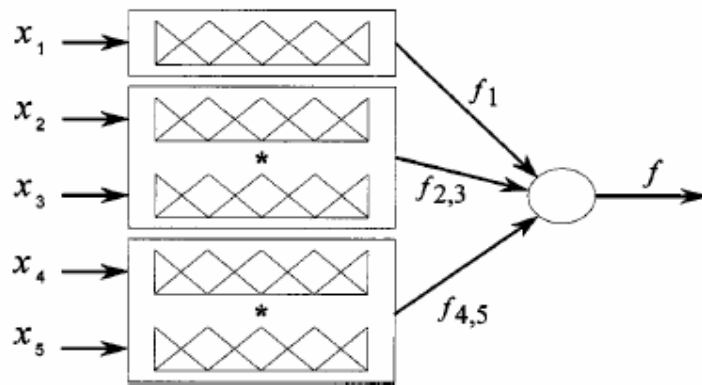


Figure 3.7-1: Illustration of additive ANOVA decomposition of AMNs [56].

A fuzzy rule within each subnetwork may have the following form:

*IF (x2 is large AND x3 is small) THEN (y is small) with confidence c*

where  $c$ =rule confidence. A rule confidence of zero indicates that the rule is not contributing to the output while a rule confidence of one indicates 100%

contribution. Values between zero and one allow the rules to partially fire. The number of fuzzy rules used in each subnetwork depends on the number of membership functions that are used to fuzzify the inputs of that subnetwork. In the above example, assuming that five membership functions are used for each variable, the first, second, and third subnetwork consist of 5, 25, and 25 fuzzy rules, respectively. The consequent part of all the rules will be ORed (i.e., summed in this case) together.

The optimum structure of a B-spline AMN is achieved through the selection of the smallest number of model inputs and the smallest number of basis functions for these inputs. ASMOD (Kavli 1993) is an algorithm that uses the above decomposition to arrive at this optimal structure. Among the other alternative approaches for automatically determining the optimum structure of B-Spline AMNs, the ASMOD algorithm is more commonly used.

In the ASMOD algorithm, for any model structure (i.e., specific combination of subnetworks, the number and location of Splines), one can use the training data to calculate the mean square error (MSE) of the output. The algorithm starts from the simplest structure (e.g., only the first variable in one subnetwork with two triangular splines) and iteratively refines its structure until some stop criteria is satisfied. In each step among a number of potential (single) changes to the structure, the one with the best performance is selected and the process continues. The addition of a new input, combining an existing input to a subnetwork, splitting a subnetwork, and deleting an input are all possible changes to the structure.

The ASMOD algorithm uses the training data to automatically determine the model inputs and the number of basis functions. However, the order of basis functions has to be determined a priori. Higher-order functions result in smoother model outputs, but increase computational cost and can lead to over fitting of the data [55].



### **3.8 METU Neurofuzzy Network (METU-NFN) Design**

Ionospheric Forecasting problem has been studied in a highly detailed manner for many years. In previous works, NNs are employed for the ionospheric forecasting, and quite promising results have been obtained. By these works, data being used for the ionospheric forecasting are studied in detail. In many years a huge amount of well organized and directly usable data for an NN system has been collected by the ionosonda stations all over the world. Normally having a huge amount of data is a reason for employing an NN model instead of neurofuzzy systems. In this work, the benefits of having such a large amount of data is tried to be kept while expert knowledge is being integrated into the model. A new neurofuzzy network designed for the ionospheric forecasting will be introduced here. To the best of our knowledge this will be the first trial application of the NFN systems on the ionospheric forecasting studies.

In this trial we will fuzzify two of the parameters which are "Kp" and "Hour of day". Kp is a parameter which defines the level of disturbance in the ionosphere. Solar disturbance may change the characteristics of the Near Earth Space both in quantity and quality. To quantify the magnetic disturbance or quietness we have chosen the 3h-planetary magnetic index Kp. For the sake of simplicity we identified two different levels of magnetic activity which determines the state of the period to be as "quiet" and "disturbed". Therefore, it will be sufficient to train only two NNs of which the contribution ratio will be determined by fuzzified Kp. In other words, the same architecture will be trained by two different target sets of data representing the "quiet and "disturbed" conditions.

For this exercise, referring to the inherent diurnal variation in TEC and  $f_oF2$ , we have chosen the "hour of day" as the second parameter which is to be employed to enrich the NN by introducing additional expert information in addition to 3h-planetary indices of Kp.

Those magnetic and diurnal variations have well defined characteristic, therefore, such information when employed as expert knowledge, notable results may come out.

### **3.8.1 Reanalysis of the problem to determine the fuzzy inputs**

As a first step of developing a fuzzy system, fuzzy parameters have to be identified. Therefore, in order to identify the fuzzy parameters, the physics of the ionosphere and affecting environmental variables are studied. Since it is a different discipline of science, knowledge of the ionosphere experts is needed. So, in order to find out some fuzzy variables influencing TEC and *foF2* values I have consulted and studied with Prof. Dr. Yurdanur Tulunay from METU Department of Aerospace Engineering, the renowned scientist on the physics of ionosphere, based on such expertise of her over twenty years of highly qualified experience on the ionospheric researches, three different parameters are determined. These are "magnetic disturbance", "diurnal variations", and "seasonal variations". Since a little more detailed study is needed to identify the exact characteristics of "seasonal variation" compared to other two parameters, we decided to exclude it in this first attempt of NFN application on the subject of this study, and include only "magnetic disturbance" and "diurnal variations" as fuzzy parameters.

Magnetic disturbance is caused by the solar system, mainly by sun storms. Magnetic disturbance may change the characteristics of the Near Earth Space, both in quantity and quality. One of the measurement units of Magnetic disturbance is 3h-planetary magnetic index which is abbreviated as Kp. Kp is scaled between 0 and 9. The Kp scale is divided into 27 level. These levels are named as follows; 0, 1-, 1, 1+, 2-, 2, 2+,..... 8+, 9-, 9.

The linguistic equivalent of Kp is quietness or disturbance. Linguistically an expert can say that; "Magnetic condition of the ionosphere is quiet.", "Magnetic condition of the ionosphere is disturbed.", "Magnetic condition of the ionosphere is almost quiet." or "Magnetic condition of the ionosphere is almost disturbed." Therefore this parameter can be fuzzified based on these linguistic terms.

Sunlight has a significant effect on the ionosphere. This effect is best observed on Diurnal variations of TEC or  $f_oF2$ . A Characteristic diurnal variation of TEC can be seen in Figure 3.8-1. At dawn and sunrise TEC and  $f_oF2$  values begin to increase sharply, and at sunset and dusk, decrease steeply. During the night time TEC and  $f_oF2$  values continue to decrease on a smaller slope. At midday they stay stable for a short period of time.

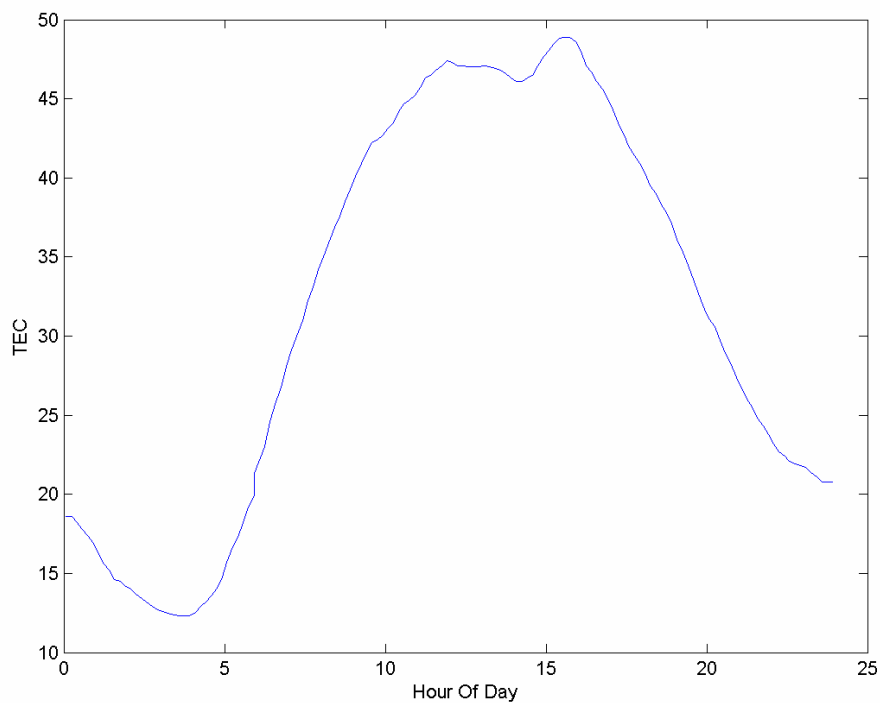


Figure 3.8-1: Characteristic diurnal variation of TEC.

The fuzzification and defuzzification processes of magnetic disturbance and diurnal variations are described in the following sections.

### 3.8.2 METU-NFN Main Structure

The whole NFN system is composed of two NNs of which the contribution to the output is controlled by fuzzified  $K_p$ , four raw inputs and two fuzzy inputs are added to these NNs as inputs. The fuzzy inputs are first processed by the fuzzy logic controller units then entered to NNs.

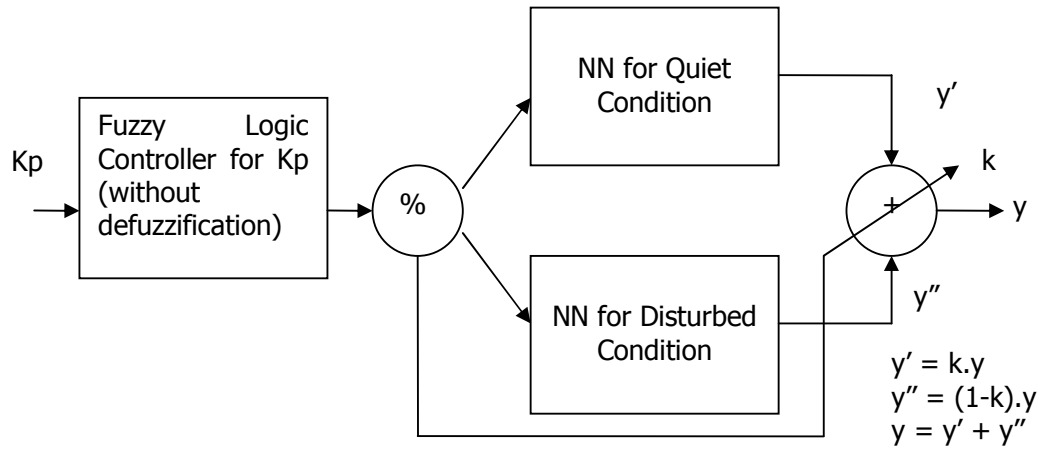


Figure 3.8-2: METU-NFN main structure.

The structure of each NN component in Figure 3.8-2 is depicted in Figure 3.8-3. As seen in the figure the "Hour of day" and the "Kp" are fuzzy inputs. Other four raw inputs which are present values, the first difference, the second difference and the relative difference, are inherited from the METU-NN model.

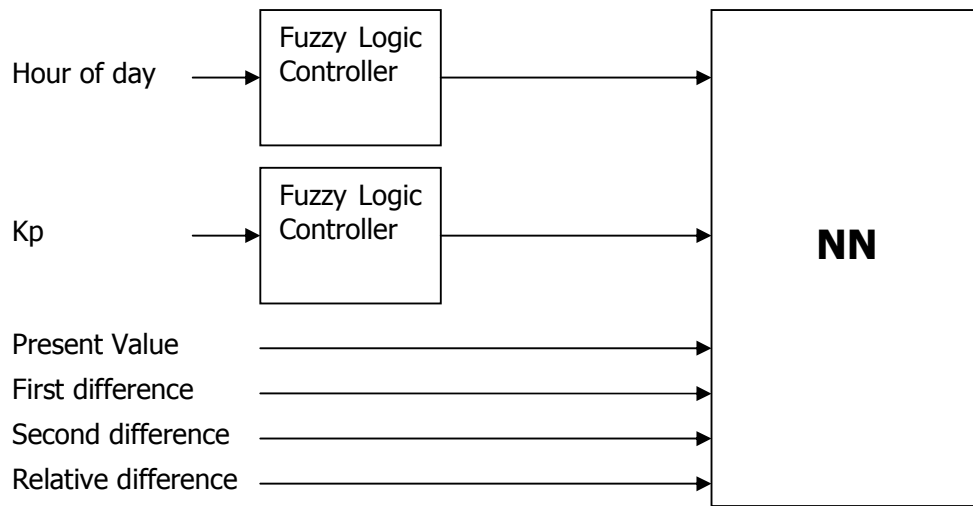


Figure 3.8-3: Structure of the NN block with fuzzy inputs.

The network will be trained with a mixed data set including both quiet and disturbed conditions. This data set should also include Kp values for each entry. This set will be applied to both NNs. The contribution ratios of each NN will be determined by the fuzzified Kp. The target values for each entry in the training set will also be scaled by Kp. Each NN will be trained with this scaled target value. The actual output of the system will be calculated by simply summing each NNs' outputs. The training algorithm is the inherited Levenberg Marquardt algorithm.

Fuzzy logic controller components are designed by employing the expert knowledge for the fuzzy inputs. The design of fuzzy components is described in section 3.8.3.

### **3.8.3 Design of fuzzy components**

Magnetic disturbance of the ionosphere and diurnal variations of TEC or *foF2* are potential fuzzy variables. For each of these two parameters a fuzzy component is employed. Block diagrams of these fuzzy components are shown in Figure 3.8-4 and Figure 3.8-7. These fuzzy components consisted of a fuzzification layer, a fuzzy inference system and a defuzzification layer. A crisp input value entered to the fuzzifier layer. In the fuzzifier, the crisp value is converted to the fuzzy value to enter into the fuzzy inference system. The fuzzy inference system is composed of the fuzzy rules. These rules connect the fuzzy inputs to the fuzzy outputs. The fuzzy output is the input of the defuzzifier. In the defuzzifier fuzzy inputs are converted to crisp values by using a defuzzification method. Having passed through the fuzzy inference system, the obtained crisp values are entered to the NNs in the whole system, which is described in the previous section and depicted in Figure 3.8-2 and Figure 3.8-3.

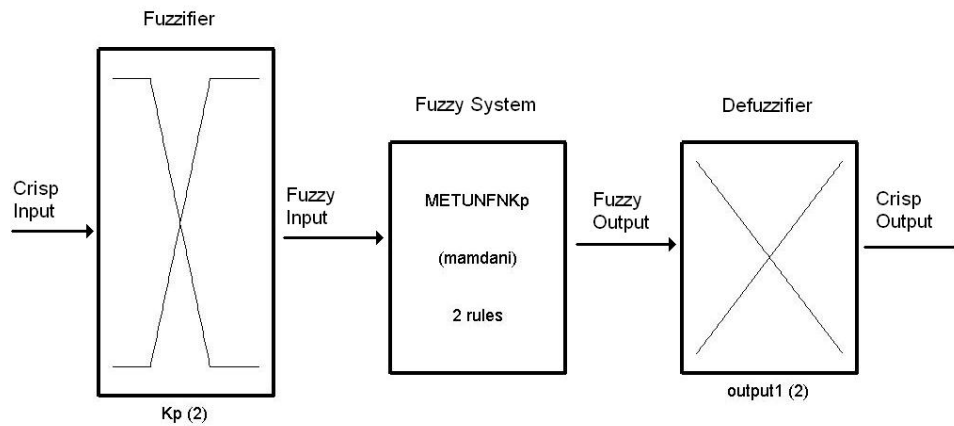


Figure 3.8-4: Fuzzy inference system for magnetic disturbance; 1 input, 1 output, 2 rules.

The main structure of the fuzzy component for the Magnetic disturbance is depicted in Figure 3.8-4. The level of magnetic disturbance is quantified by  $K_p$  as described previously on section 3.8.1.  $K_p$  index is normally quantified into 27 levels. Having consulted with the experts I quantified  $K_p$  with two fuzzy terms. These terms are “Quiet” and “Disturbed”. If  $K_p$  is equal or lower than 2.3, it is absolutely quiet. If  $K_p$  is equal or greater than 6, it’s absolutely disturbed. Levels between 2.3 and 6 can linguistically be called as “almost quiet” or “almost disturbed”. These linguistic terms also determine the input membership functions. Since we have two linguistic terms, one for each, there are two input membership functions. The graphical representation of input membership functions for  $K_p$  can be seen in Figure 3.8-5. For the sake of simplicity linear trapezoidal membership functions have been preferred.

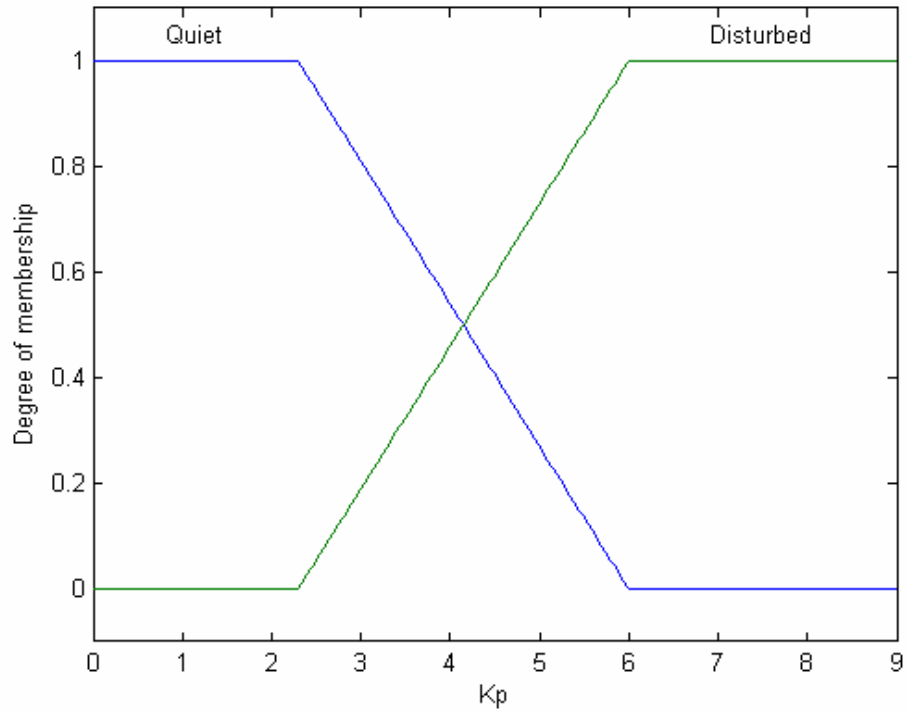


Figure 3.8-5: Input membership functions for  $K_p$ .

The output has also two membership functions. The fuzzified output of  $K_p$  is used to select the contribution ratio of the NNs to the output of the system as depicted in Figure 3.8-2. Input and output layers are connected through a fuzzy inference system which is composed of fuzzy rules. Output membership functions are also linear functions and linguistically define the contribution ratio of the NNs.

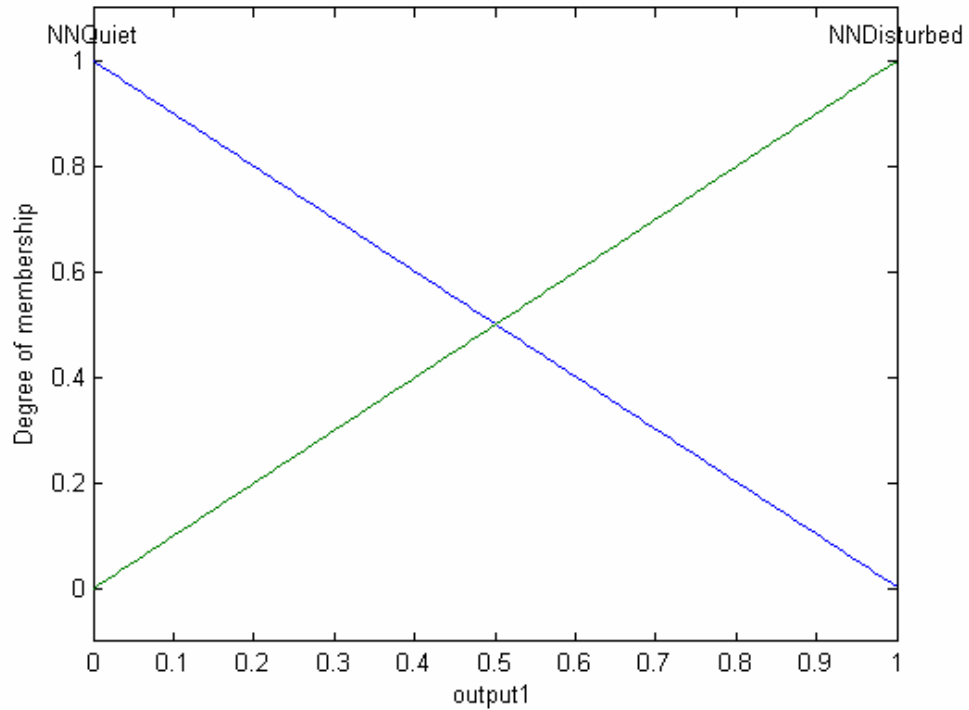


Figure 3.8-6: Output membership functions for Kp.

There are two fuzzy rules for Kp, which are:

1. *If Kp is Quiet then Output is NNQuiet.*
2. *If Kp is Disturbed then Output is NNDisturbed.*

The main structure of the fuzzy component for the diurnal variations of TEC or  $foF2$  is depicted in Figure 3.8-7. The hour of day determines the level of diurnal variations. Therefore, the hour of day is the crisp input to the fuzzy component and is fuzzified in the fuzzification layer.



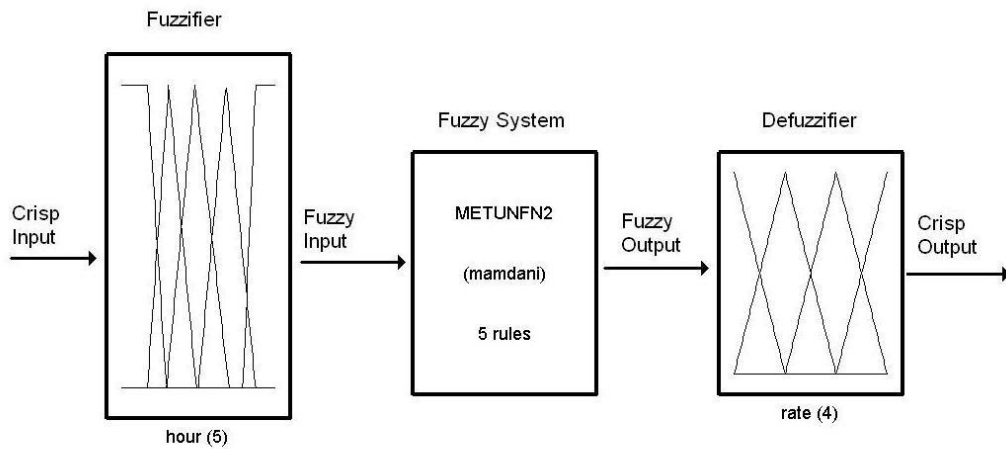


Figure 3.8-7: Fuzzy inference system for diurnal variation; 1 input, 1 output, 5 rules.

For the fuzzification and the subsequent defuzzification processes, input and output membership functions should be determined. Input and output membership functions are produced from the characteristic graph of the diurnal variations (Figure 3.8-1). This graph is divided into four regions, determined by the solar time of the day. This graph is shown in Figure 3.8-8. As it is shown on the graph, the hour of day is divided into four regions: late-night, morning, midday, and afternoon. These regions also represent the fuzzy terms of the input variable. Corresponding diurnal variation regions are negative-low, positive-high, zero, and negative-high.

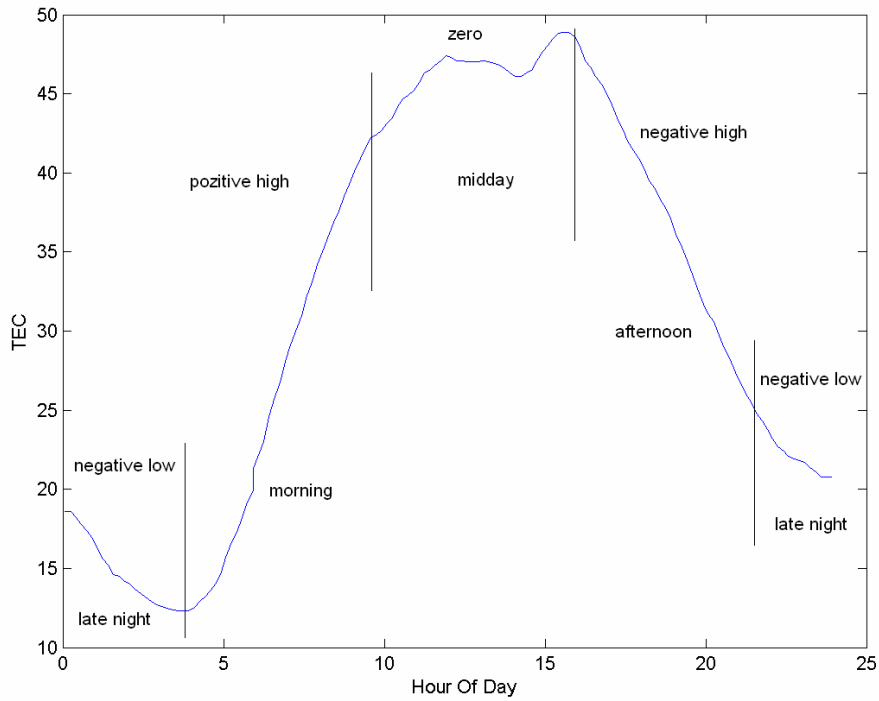


Figure 3.8-8: Regions of diurnal variations.

Hour of day is the crisp input to the fuzzifier layer. In the fuzzifier layer hours of a day are represented by membership functions. These membership functions are depicted in Figure 3.8-9 and the boundaries are given in Table 3.8-1.

Table 3.8-1: Boundaries of the regions of diurnal variations

| Region     | Absolute Time | Quite Time               |
|------------|---------------|--------------------------|
| Late night | 21:00 – 04:00 | 04:00 – 07:00            |
| Morning    | 07:00         | 04:00–07:00, 07:00–12:00 |
| Midday     | 12:00         | 07:00–12:00, 12:00–16:00 |
| Afternoon  | 16:00         | 12:00–16:00, 16:00–21:00 |

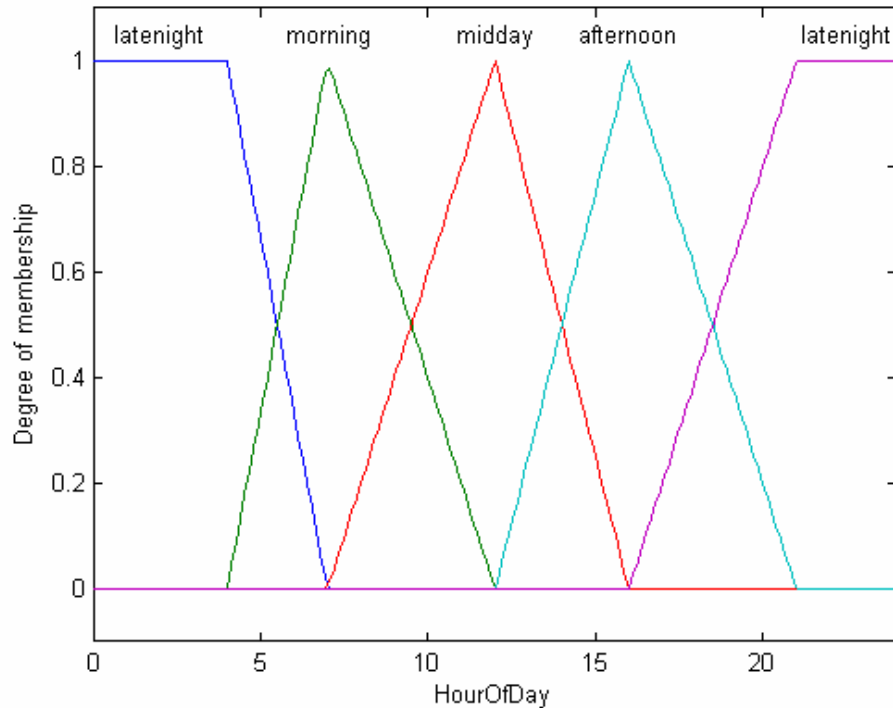


Figure 3.8-9: Input membership functions for fuzzification of the diurnal variations.

The fuzzifier layer produces a fuzzy output corresponding to the crisp input. For example if the hour is 03:00 o'clock, then the fuzzy output will simply be 100% latenight. In other words 3 o'clock is linguistically expressed as "late night". For example if the hour is 11 o'clock, then the fuzzy output will be 80% midday and 20% morning. Similarly it is linguistically expressed as "just midday" or "late morning".

Fuzzy outputs are processed in the fuzzy inference system which is composed of fuzzy rules. In this fuzzy inference system four fuzzy rules are employed to convert fuzzy input to fuzzy output. These rules are listed below.

Fuzzy rules for fuzzy inference system of diurnal variations:

1. *If hour of day is "latenight" then rate is "neglow"*
2. *If hour of day is "morning" then rate is "pozhigh"*

3. *If hour of day is "midday" then rate is "zero"*
4. *If hour of day is "afternoon" then rate is "neghigh"*

By using these rules output membership functions are selected in the defuzzifier layer with a ratio determined in the fuzzifier layer. Defuzzifier layer consists of output membership functions and a defuzzification engine. The output membership functions are determined examining the diurnal variations graph in Figure 3.8-8.

Similar to input membership functions, the regions on the graph represent the fuzzy terms of output of the defuzzifier layer. These terms are also used as output membership functions, which are organized by defining the boundary regions as depicted in Figure 3.8-10.

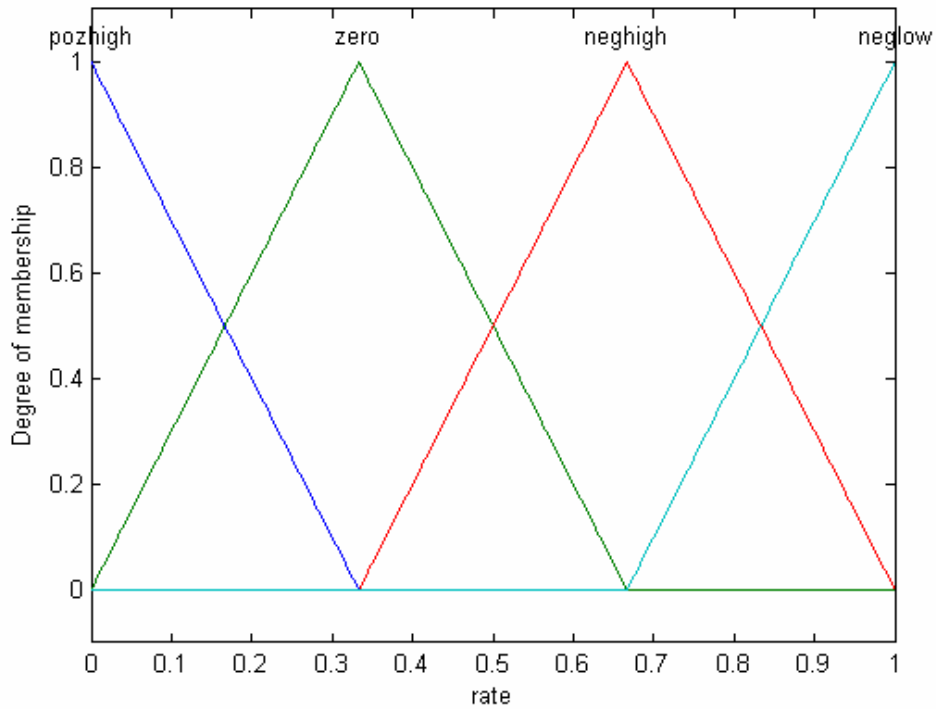


Figure 3.8-10: Output membership functions for fuzzification of diurnal variations.

In Figure 3.8-10, x axis represents the crisp value of the rate of the diurnal variations. A defuzzification method has to be used to find out this crisp value. Fuzzy output of the fuzzy inference system is something like that; 80% midday, 20% morning. This fuzzy information is the input of defuzzification layer. In the defuzzification layer by using output membership functions in Figure 3.8-10 and the defuzzification method as described on the section 3.8.3.1 a crisp value is obtained. The rate is chosen between 0 and 1 because of the other inputs to the NNs are normalized between 0 and 1. Therefore there is no need to normalize these crisp values.

That is the final step of fuzzy component used in the model. This crisp value is utilized in NNs as similar to the other inputs.

### **3.8.3.1 Defuzzification Method**

The conversion of a linguistic (fuzzy) result to a real (crisp) value is called as defuzzification. Fuzzy logic mimics the human decision and evaluation process. Therefore a well established defuzzification method should approximate this approach. Defuzzification is usually a two step process. In the first step a typical value is computed for each term in the linguistic variable. In the second step, the "best compromise" is determined by balancing out the results. For the sake of computational efficiency "Center of Maximum" (CoM) defuzzification method is used in this study. A comparison chart of defuzzification methods is given in Table 3.8-2 [58].

Table 3.8-2: Defuzzification methods comparison [58]

|                           | <b>Center of Area (CoA, CoG)</b>                       | <b>Center of Maximum (CoM)</b>           | <b>Mean Of Maximum (MoM)</b>          |
|---------------------------|--|--|---------------------------------------|
| Linguistic Characteristic | "Best Compromise"                                      | "Best Compromise"                        | "Most plausible solution"             |
| Fit with Intuition        | Implausible various MBF shapes and strong overlap MBFs | Good                                     | Good                                  |
| Continuity                | YES  | YES                                      | NO                                    |
| Computational Efficiency  | Very low   | High                                     | Very High                             |
| Applications              | Control, decision support, data analysis               | Control, decision support, data analysis | Pattern recognition, decision support |

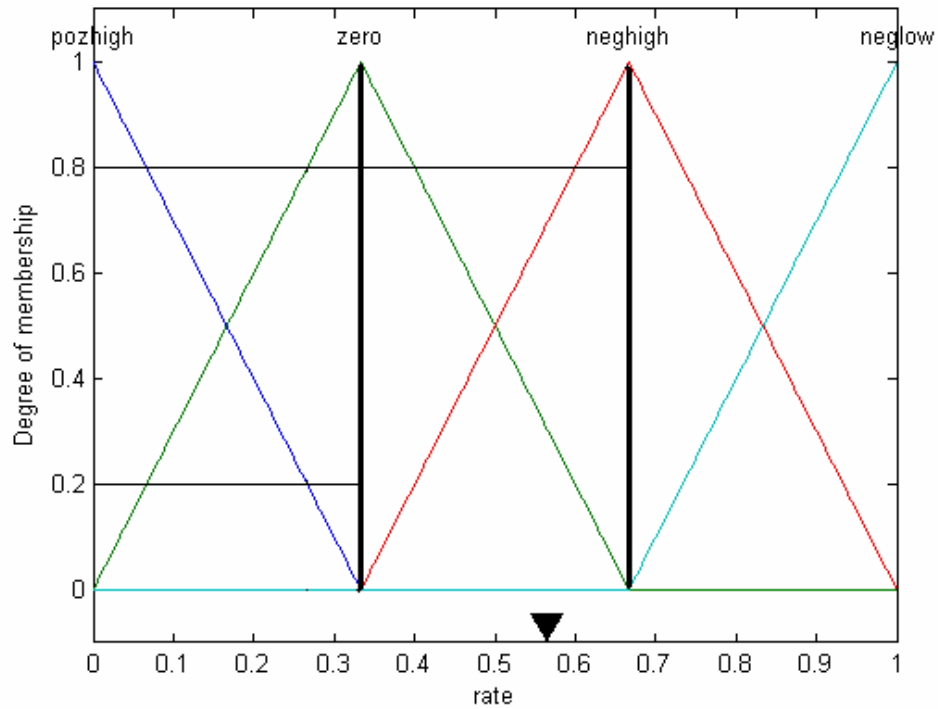


Figure 3.8-11: Defuzzification with CoM.

An example of CoM defuzzification method is depicted in Figure 3.8-11. As an example lets say that the fuzzy output of our fuzzy inference system be 0.8

neghigh and 0.2 zero. The maximum value of zero is 0.32 and neghigh is 0.68. Then the crisp value by using CoM will come out  $0.68 \times 0.8 + 0.32 \times 0.2 = 0.608$ .

### **3.9 Comparison of Various NN and NFN models Applied to Ionospheric Forecasting Problem**

In this section, NFN models developed in this study will be compared to NN models. NN systems are far more mature compared to neurofuzzy systems. And the application of NN systems to the problem of ionospheric forecasting, as a result the METU-NN model produced very successful results. In this thesis an attempt is made to develop an NFN model for ionospheric forecasting for the first time. The aim is to make use of the raw data and expert knowledge both, in the same system. In the literature it's usually assumed that having plentiful usable data to train an NN system is the reason to use an NN system instead of a neurofuzzy system [40][54][47][59]. It's expected that the results of NFN model developed during this study may not reach the performance of NN models. But if reasonable results may be obtained it will show that NFN models can be applicable and worth to study on. To obtain better results the physical domain of the problem has to be examined in more detail. All the expert knowledge should be collected and organized to develop much more appropriate NFN models. For this study we only used readily available expert knowledge on Magnetic disturbance and diurnal variations of TEC and  $f_oF2$ , to develop an NFN system to just see what kind of result would be obtained in a short time.

Five different architectures are prepared to compare. These architectures are as follows

1. METU-NN : The model described in section 1.2.2 and depicted in Figure 1.2-1
2. METU-NN with additional Kp column : Kp is added as an extra input.
3. METU-NN with additional Kpf column : Fuzzified Kp is added as an extra input.
4. METU-NFN 6 Inputs double NN : The NFN model described in the section 3.8.2

5. METU-NFN 6 Inputs single NN : Single NN is used instead of double NN.

For the comparisons all other variables except the model should be equal. Therefore in each model NNs' have the same properties. That is each NN has 4 hidden neurons with Tansig activation functions and one output neuron with linear activation function. Levenberg Marquardt training algorithm is used as a training algorithm. Three different sets of data are prepared to use in training, validation, and operation phases of each model. These are the TEC data sets which are obtained under the conditions given in [60]. Organization of data sets is given in Table 3.9-1. All the data sets are GPS TEC data having 10 minute intervals.

Table 3.9-1: Data sets organization

|                       | <b>Month</b>        | <b>Year</b> | <b>Station</b>                   |
|-----------------------|---------------------|-------------|----------------------------------|
| <b>Training Set</b>   | 1 April – 31<br>May | 2000        | Chilbolton (51.8° N; 1.26°<br>W) |
| <b>Validation Set</b> | 1 April – 31<br>May | 2001        | Chilbolton (51.8° N; 1.26°<br>W) |
| <b>Operation Set</b>  | 1 April – 31<br>May | 2002        | Hailsham (50.9° N; 0.3° E)       |

The Kp data is retrieved from the ftp site of The National Geophysical Data Center, USA.

([ftp://ftp.ngdc.noaa.gov/STP/GEOMAGNETIC\\_DATA/INDICES/KP\\_AP/](ftp://ftp.ngdc.noaa.gov/STP/GEOMAGNETIC_DATA/INDICES/KP_AP/)). The data gaps in the Kp data are filled with linear interpolation. Under these conditions the summary of the results are tabulated in Table 3.9-2.

Table 3.9-2: Summary of Comparison Results

|                                      | <b>NN + Kp</b> | <b>NN + Kpf</b> | <b>NFN 2NN</b> | <b>NFN 1NN</b> | <b>METU-NN</b> |
|--------------------------------------|----------------|-----------------|----------------|----------------|----------------|
| <b>Cross correlation</b>             | 0.986790       | 0.982862        | 0.908643       | 0.983900       | 0.987037       |
| <b>MSE</b>                           | 3.730893       | 4.022666        | 23.028308      | 3.770725       | 3.036461       |
| <b>RMSE</b>                          | 1.931552       | 2.005659        | 4.798782       | 1.941836       | 1.742545       |
| <b>Average Absolute Error (TECU)</b> | 1.348231       | 1.387980        | 3.630039       | 1.319473       | 1.162553       |
| <b>Average Epoch Duration (ms)</b>   | 3999           | 4094            | 3537           | 1717           | 3233           |



Interpreting the results, the best performance is obtained by METU-NN model with the RMSE of 1.7425. "NN + Kp" and NFN with single NN models produced similar results with RMSE of 1.9315 and 1.9418 respectively. Adding a fuzzified Kp input to the METU-NN model slightly decreased the performance compared to the model with a raw Kp input added to METU-NN model. The NFN model described in 3.8.2 produced the worst performance with the RMSE of 4.7987. A RMSE value below 2 is accepted for a usable system. However, both the cross correlation value of 0.90 and the RMSE value of 4.7 are very reasonable results and even can be considered promising for the first attempt of a new model. It should be noted that, the worse results are expected at the beginning. These results show us that, it's worth to study physical details of the ionosphere to obtain more comprehensive expert knowledge to develop more advanced neurofuzzy models.

After training phases all the models are operated with the operation data set. For each type of model scatter diagrams are plotted with METU-NFN Software and the obtained scatter diagrams are printed below.

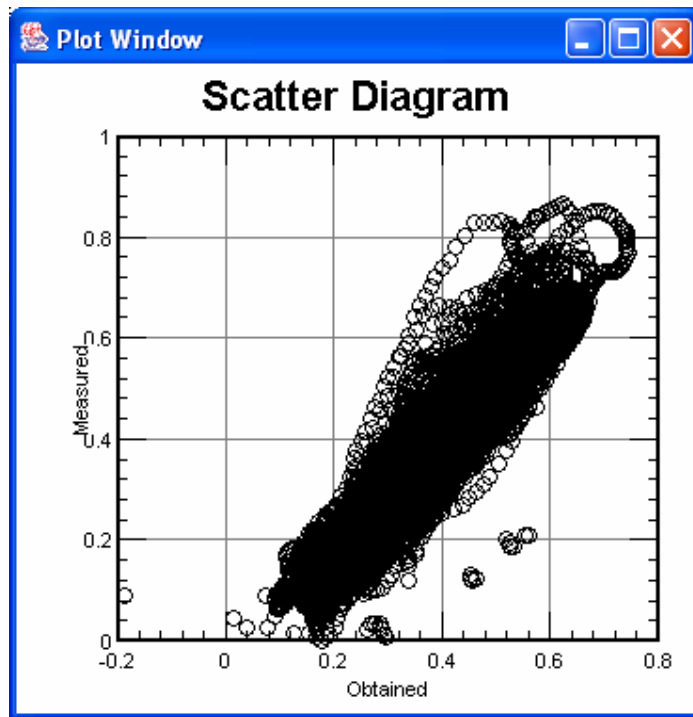


Figure 3.9-1: Scatter diagram obtained by operating NFN 2NN.

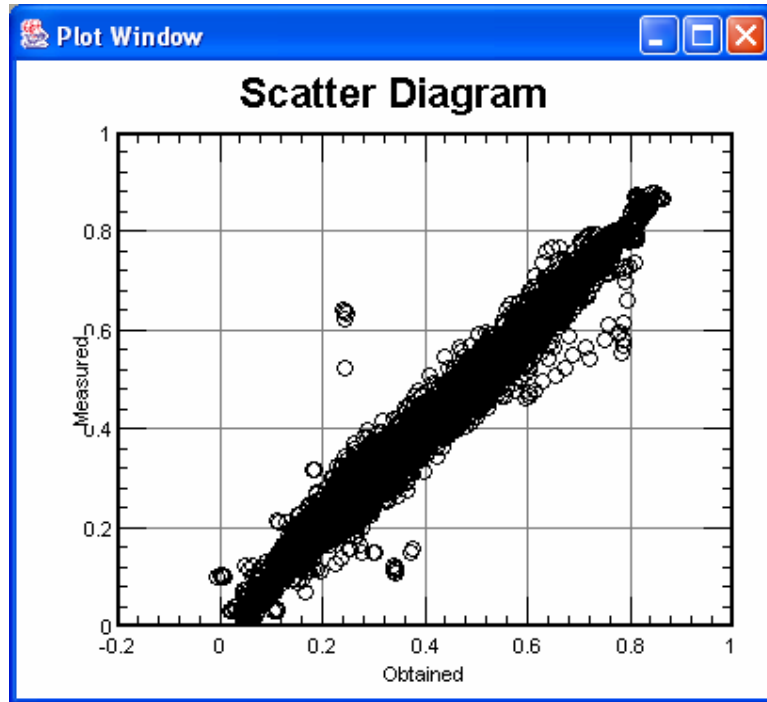


Figure 3.9-2: Scatter diagram obtained by operating NFN 1NN.

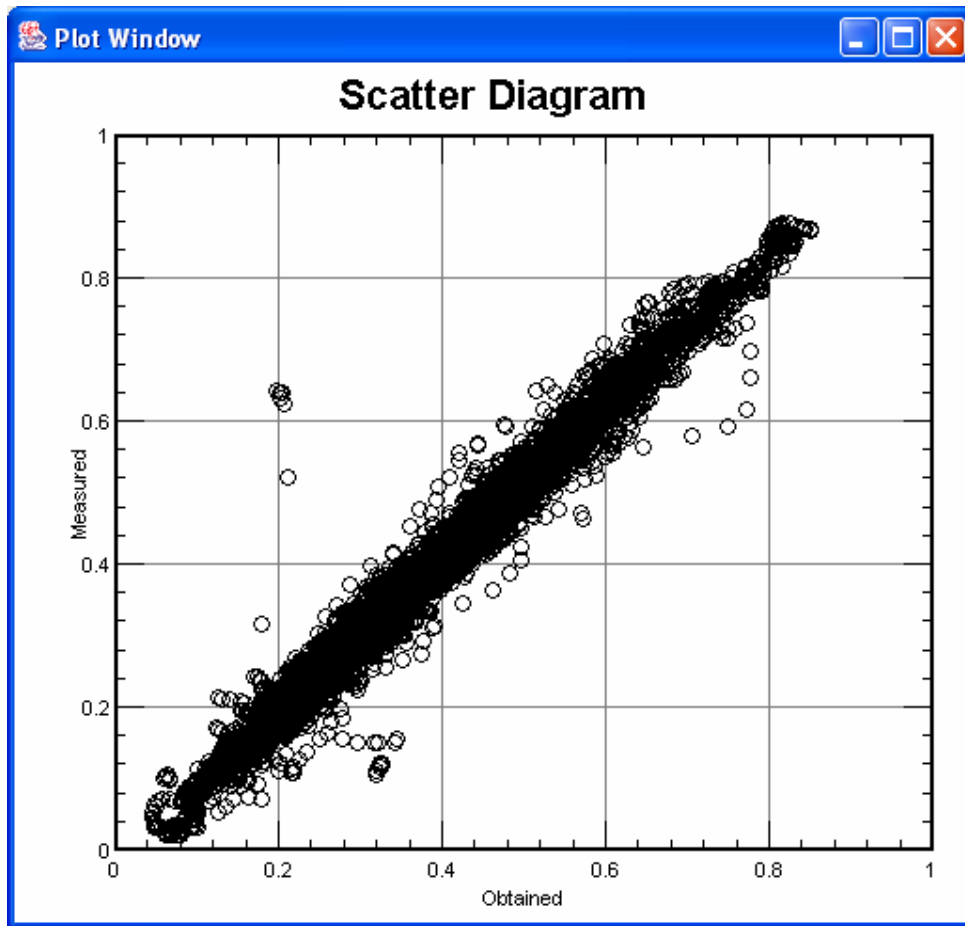


Figure 3.9-3: Scatter diagram obtained by operating NN + Kp.

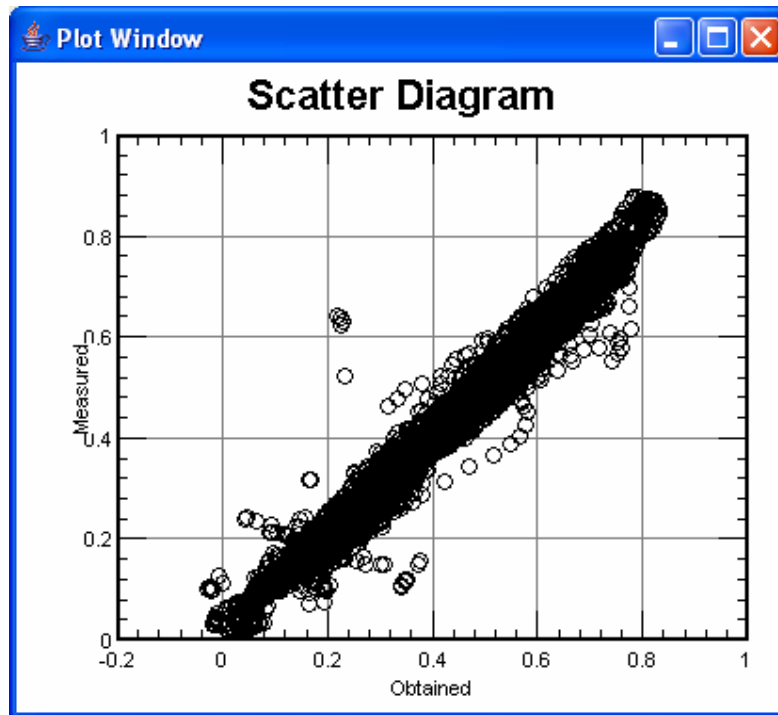


Figure 3.9-4: Scatter diagram obtained by operating NN + Kpf.

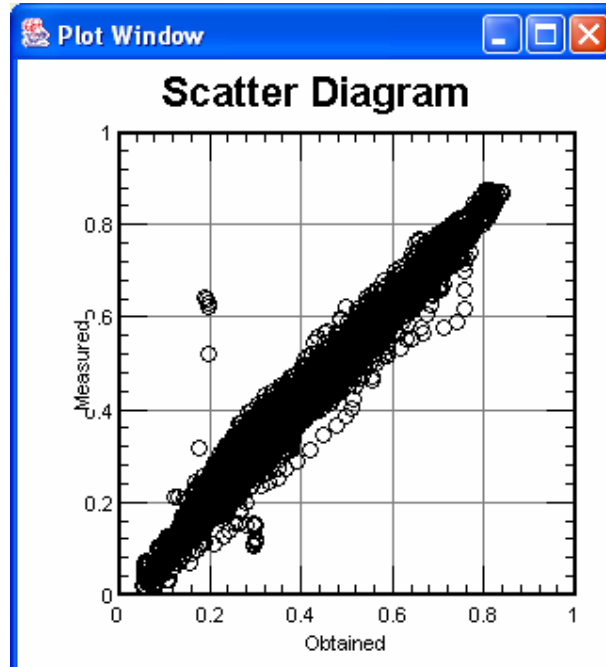


Figure 3.9-5: Scatter diagram obtained by operating METU-NN.

## **CHAPTER 4**

### **METU IONOSPHERIC FORECASTING SOFTWARE DESIGN**

The knowledge gained through many years of theoretical studies on the ionospheric forecasting finally produced a mature NN model, which is made usable over the Web by developing METU-IFS for the first time in this study with such remarkable features. This chapter discusses the outstanding properties of this software and the design details.

#### **4.1 Introduction to METU Ionospheric Forecasting Software (METU-IFS)**

Middle East Technical University Ionospheric Forecasting Software (METU-IFS) has been developed to make usable various neural network and neurofuzzy models over the Internet with a user friendly interface. Availability of these models will greatly help the related scientific circles. Designing new neural network or neurofuzzy architectures, training previously designed ones and operating previously trained ones are the main features of METU-IFS. Access to these features is protected with a three level of authorization. According to this authorization mechanism, users are divided into three groups which are namely "Power User", "Researcher" and "Ordinary User". All the data collected during the usage of METU-IFS is saved into a database. This data will be received from a wide variety of the users all over the world while the models presented with this software are trained or operated. The data may prove useful for improving the models to be developed in future researches.

In developing METU-IFS, Java is chosen as a software language, due to platform independence and powerful object oriented structure of this language. METU-IFS is designed as a client-server application based on “three-tier” architecture. Details of design are given in the following sections.

## **4.2 Web Based Client-Server Applications**

METU-IFS is a kind of web based client-server application. Main terminologies to be seen in the following the sections and the technologies applied in the design of this software are explained here. Besides that, some advantages of the client-server applications are also discussed.

In software engineering, a web application is the one delivered to users from a web server over the World Wide Web. Users can have access to the applications from any computer connected to the Internet via a secure, password-protected login page. Though other varieties are possible, a web application is commonly structured as a three-tiered application. In its most common form, a web browser or a client software is the first tier, an engine created using some dynamic web content technology (e.g., CGI, PHP, Java servlets or Active Server Pages) is the middle tier which is generally called as server, and a database is the third tier. The client sends requests to the middle tier, which services them by processing request data, making queries and updates against the database, in some cases generating a user interface and responding to the client with end data [61].

Web-based applications are becoming so popular in our daily life that not a single day we pass without having any connection to them. These applications range from simple to more sophisticated ones, where millions of dollars in revenue are made. Below several pros are listed, which make them so popular.

### **4.2.1 Installation and Maintenance**

Installation and maintenance are not complicated. All the intelligence is collected in a server side application. Once a new version or upgrade is installed on the

server, all the users can have access to it straight away. There is no need to upgrade each client PC with all the related migration problems that it might bring forth. And as the upgrades on the server are only performed by an experienced professional, the results are more predictable and reliable.

Increasing processor capacity also becomes a far simpler operation. If an application requires more power to perform tasks, only the server hardware needs to be upgraded.

#### **4.2.2 Security**

The server application which includes all the intelligence will be deployed on a dedicated secure server, which is monitored and maintained by experienced server administrators. This is far more effective than monitoring hundreds or even thousands of client computers, as is the case with new standalone desktop applications. And besides that, clients are authorized with a password controlled login mechanism. Thus, the clients may be distributed into several user groups to restrict accesses to the server side application.

#### **4.2.3 Cost effective development**

With web-based applications, users have access to the system via a uniform environment, the web browser, where there's no need to test the application on all possible operating system versions and configurations. This also makes troubleshooting much easier.

Using internet technologies based on industry-wide standards, it is possible to achieve a far greater level of interoperability between applications than with isolated desktop systems. This means that it is possible to rapidly integrate them within existing infrastructures and platforms.

#### **4.2.4 Accessibility**

Unlike traditional applications, web systems are accessible anytime, anywhere, via a PC with an Internet connection.

### **4.3 Software Design**

Design of METU-IFS is started with production of a Software Requirements Specifications (SRS) document in which all the requirements are listed. This SRS document is given in appendix-A. Following the SRS document, a function point analysis (or cost estimation analysis) based on COCOMO model is done at the very beginning of the thesis. The planned software was a very light version of the software we have actually realized. The analysis has shown that the estimated software size would be about 6000 LOC (Lines Of Code). The man-hour required to complete the project is estimated to be 14.7 person-month. Duration required to complete the project is estimated to be 7 months which means that, with an average of 2 full time men effort the project duration is 7 months. This function point analysis is given in appendix-B. The realizations are as follows; size of the realized software is about 10000 LOC, duration is about 12 months with one part time staff.

Generally the development of web-based applications can at least be divided in two parts such as a content presentation part which is known as client side, and behind the scene, a data processing part which is known as the server side. This approach is known as Three-Tier architecture. METU Ionospheric Forecasting Software (METU-IFS) is also designed based on Three-Tier architecture.

#### **4.3.1 Three-Tier Architecture**

The three-tier model is a software architecture and a software design pattern. The three tier architecture is used when an effectively distributed client/server design is needed, which provides increased performance, flexibility, maintainability, reusability, and scalability while hiding the complexity of distributed processing from the user. These characteristics make three layer architectures a popular choice for Internet applications and net-centric information systems.

In computing, three-tier is a client-server architecture in which the user interface, functional process logic ("business rules") and data storage and data access are



developed and maintained as independent modules, most often on separate platforms. The term "three-tier" or "three-layer", as well as the concept of multi-tier architectures, seems to have originated from Rational Software or Microsoft.

Apart from the usual advantages of modular software with well defined interfaces, the three-tier architecture is intended to allow any of the three tiers to be upgraded or replaced independently as requirements and/or technology change. For example, an upgrade of desktop operating system from Microsoft Windows to UNIX would only affect the user interface code.

### **4.3.2 Client Side**

This section discusses the properties of the client side and gives the design details.

#### **4.3.2.1 Main Properties**

The client side of METU-IFS is designed actually as a dummy user interface. No intelligent processing is done on the client side. That is, algorithms or models, which are intellectual assets, are not implemented on the client software. In the software world it is a common problem that executable software files may be reverse engineered to obtain the source code. And it is relatively easier if a programming language such as Java is used where the executables run on a virtual machine. This virtual machine in Java is called Java Runtime Environment (JRE). By this design our intellectual assets are protected against such attacks.

Client application is mainly designed as graphical user interface (GUI) of the whole portion of METU-IFS. The User interface is built on runtime by selecting one of the three predefined graphical user interfaces (GUI). Users are collected into three groups which are named as "Power User", "Researcher" and "Ordinary User". Since each user has different access rights the GUI is also differs by user type. GUI design related to each user type will be described later.

Client application mainly runs around a main thread and a connection thread. While all the GUI related work is handled with main thread, all the

communication related work is handled with connection thread. The main advantage of this design is that, user interface is not blocked in case of a probable communication delay or even communication problems. The main screen of METU-IFS can be seen in Figure 4.3-1. And the "About" window is in Figure 4.3-2.

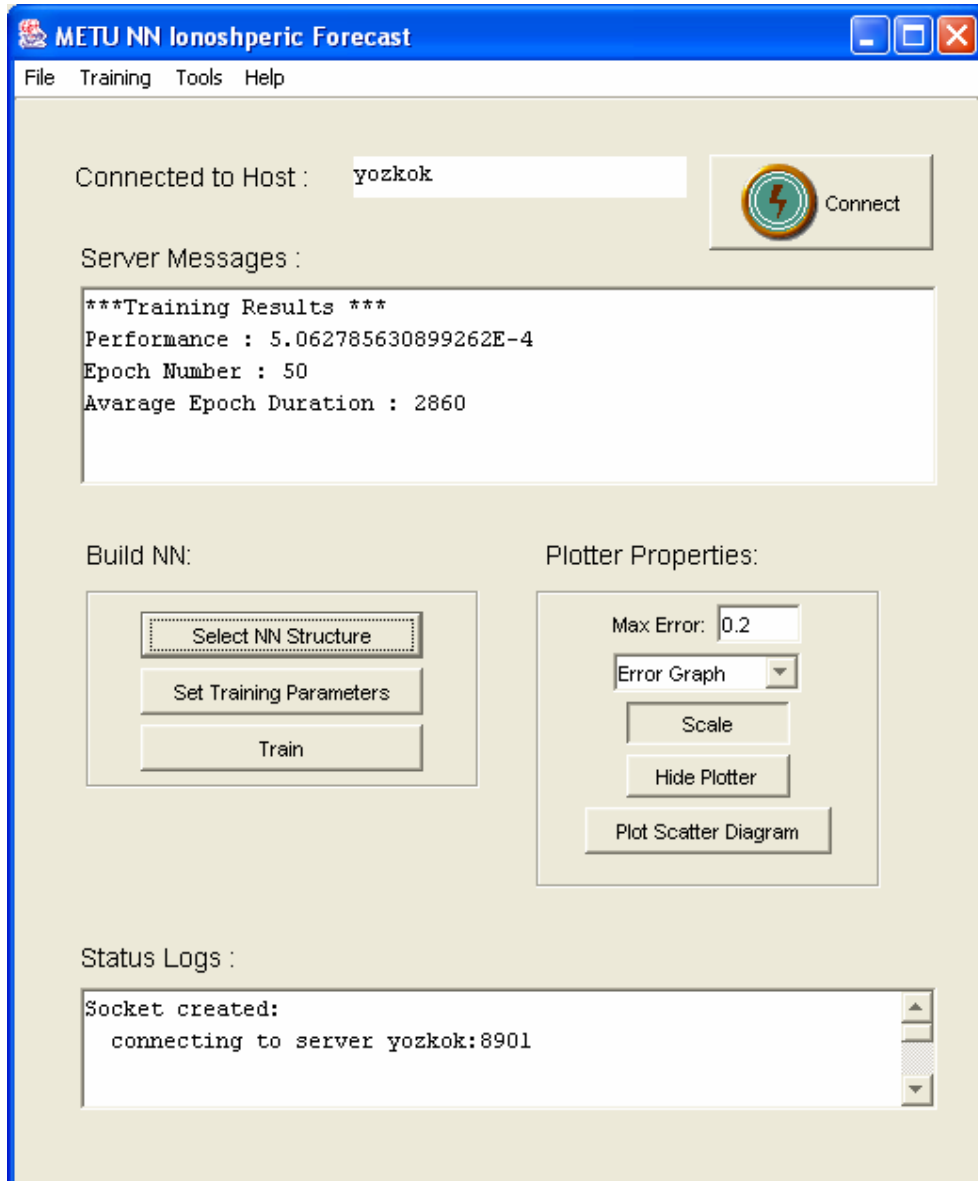


Figure 4.3-1: METU-IFS main window.

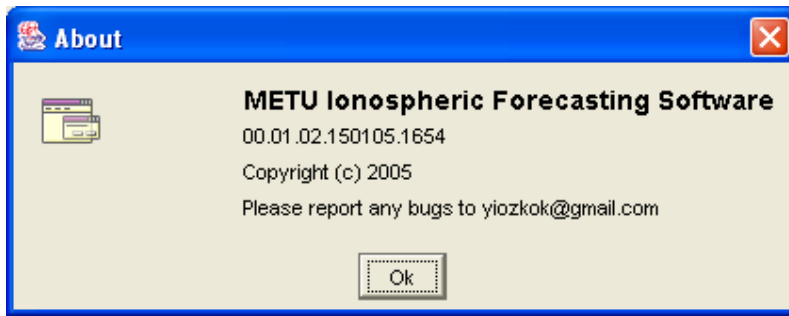


Figure 4.3-2: METU-IFS “About” window.

#### 4.3.2.2 Authorization Module

Authorization is the first step of all the rest. When the application is started this module first reads the server address and port number from a configuration file and tries to connect to the server. If the connection is successful, a “Login Window” (Figure 4.3-4) pops up, otherwise a warning message (Figure 4.3-3) is displayed that says to the user to check the internet connection.



Figure 4.3-3: Connection error window.

After a connection is established with the server the user enters its user name and the password into the “Login Window” to authorize. This user name and password will be e-mailed to the user for a first time usage. The first time the user is connected to the server, a new password is sent to the server which is generated by using “Ethernet address” (that is also known as Mac address) of the user’s PC and the given password. For the later connections this password is saved in the database of METU-IFS. With this authentication mechanism METU-

IFS Client application will be PC locked and would not be moved to any other PC without the administrators' knowledge and permission.

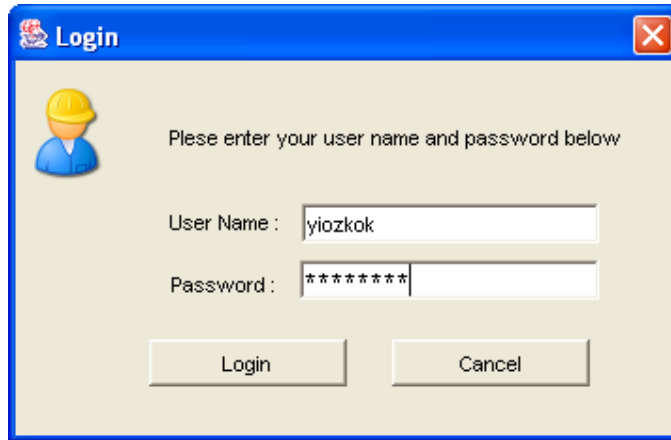


Figure 4.3-4: Login window.

Server sends the user type to the client application, if the user name and password is verified. As soon as the client receives user type, it builds up the GUI according to user type. So clients can access the services served by the server as long as their rights allow. Authorization module with its main methods is summarized in Table 4.3-1.

Table 4.3-1: Authorization Module structure

| <b>Authorization Module</b> |                      |                       |
|-----------------------------|----------------------|-----------------------|
| <b>ConnectServer Method</b> |                      |                       |
| Call by                     | Main frame           |                       |
| Input Name                  | Input Type           | Input Source          |
| Server Address              | string               | Configuration<br>File |
| Port Number                 | integer              |                       |
| Output Name                 | Output Type          |                       |
| Connection Status           | as Boolean           |                       |
| <b>Login Method</b>         |                      |                       |
| Call by                     | Authorization Module |                       |
| Input Name                  | Input Type           | Input Source          |
| User Name                   | String               | User Input            |
| Password                    | String               |                       |
| Output Name                 | Output Type          |                       |
| Authorization Status        | as Boolean           |                       |
| User Type                   | String               |                       |

#### 4.3.2.3 Neural Network Builder and Selector Module

This module is used to build or select a neural network architecture. Each of the three user types has different accesses to this module that is reflected to the GUI also. A "Power User" user has all the rights of building a new architecture and selecting a previously built or trained architecture. The GUI for a "Power User" to select an architecture and to build a new architecture is respectively shown in Figure 4.3-5 and Figure 4.3-6. "Power User" user can also save its newly built NN structure.

"Researcher" user has only rights to select a previously built architecture. It can select a NN architecture whether it is trained or not.

An "Ordinary" user can only select previously trained architectures in order to operate them.

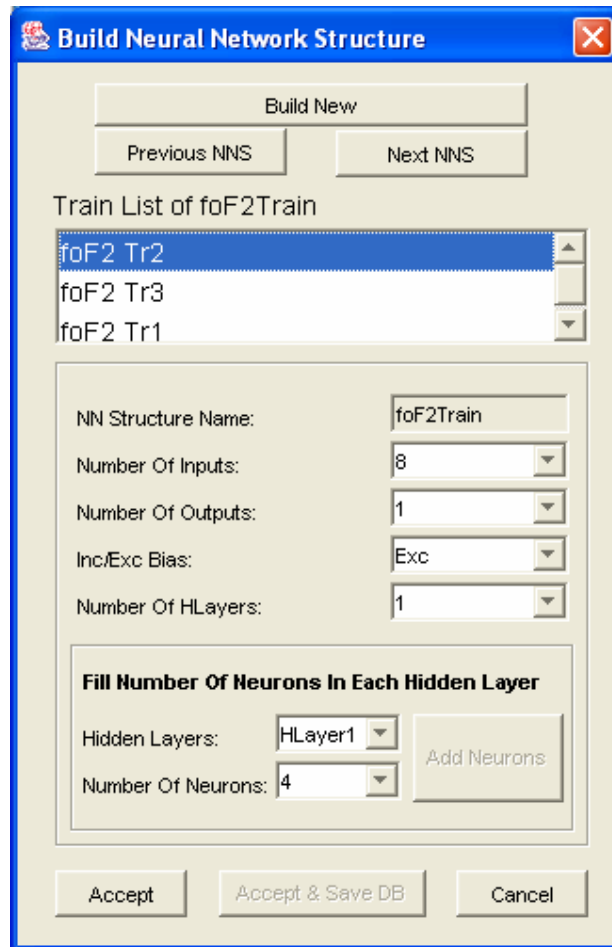


Figure 4.3-5: NN structure selection window.

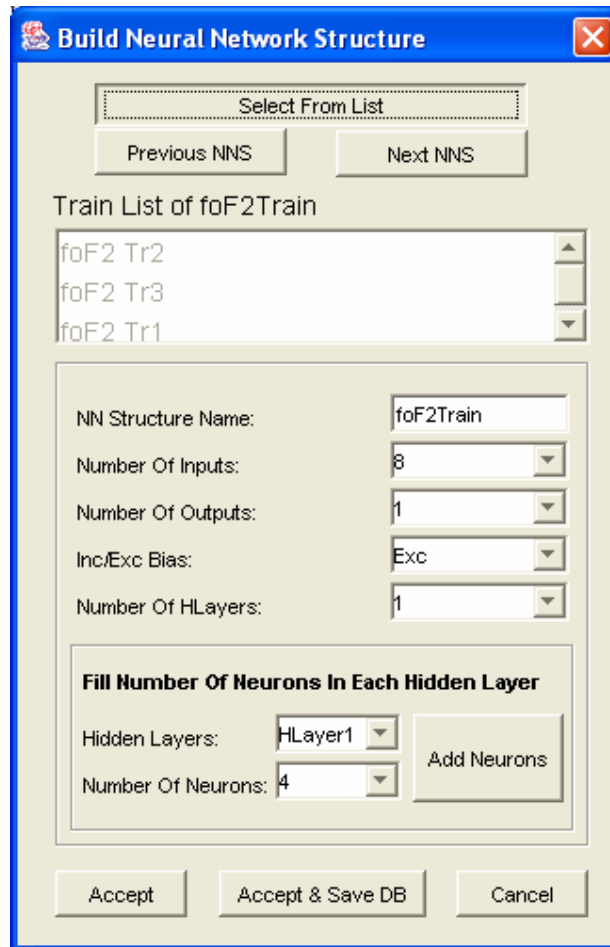


Figure 4.3-6: NN structure building window.

The neural network structures list and each structures training results list are retrieved from database of METU-IFS through the server. Just before this module is loaded the server is asked to send these lists. Client application receives these lists and then just lists them to the user. If the user leaves this module by accepting a NN structure, according to the users' modifications or selections this NN structure is saved into a local structure.

#### 4.3.2.4 Training Parameters Settings Module

This module is designed to select training parameters. This module is forbidden for "Ordinary" type of users. The GUI design for the module is shown in Figure 4.3-7. In this module all the parameters related to the training algorithm are defined. Training and validation data sets are also entered within this module.

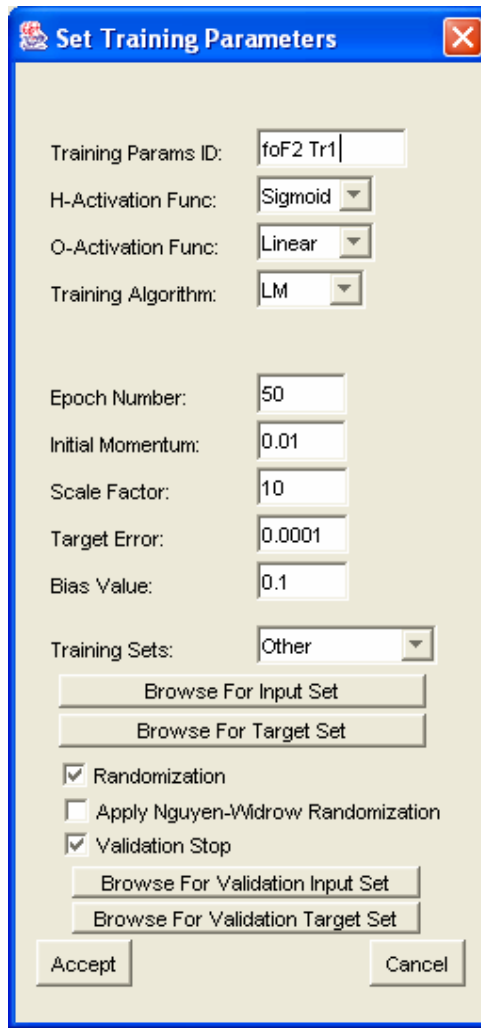


Figure 4.3-7: Training parameters selection window when LM is selected as an algorithm.



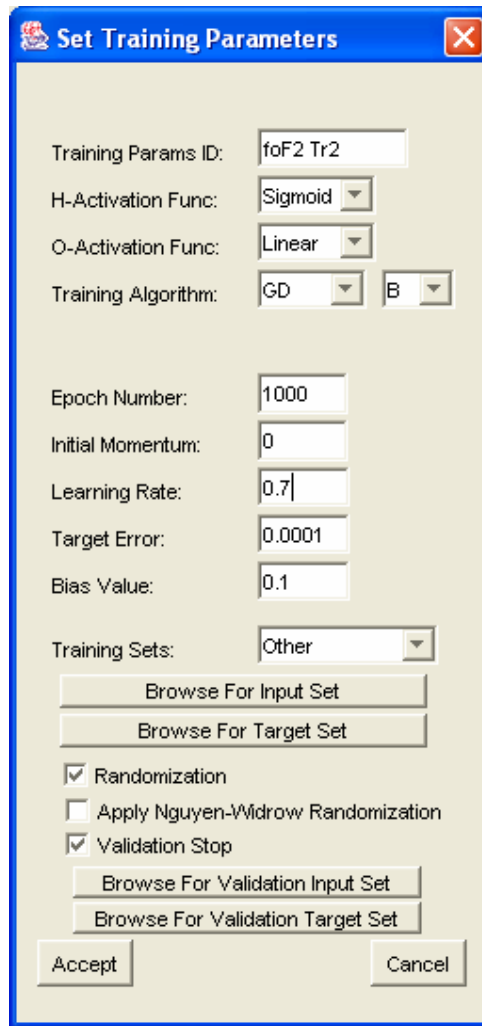


Figure 4.3-8: Training parameters selection window when GD is selected as an algorithm.

#### 4.3.2.5 Operation Settings Module

In this module operation settings are retrieved from user. Operation set files are selected by the user. And the name of the file to save operation results is entered by the user.

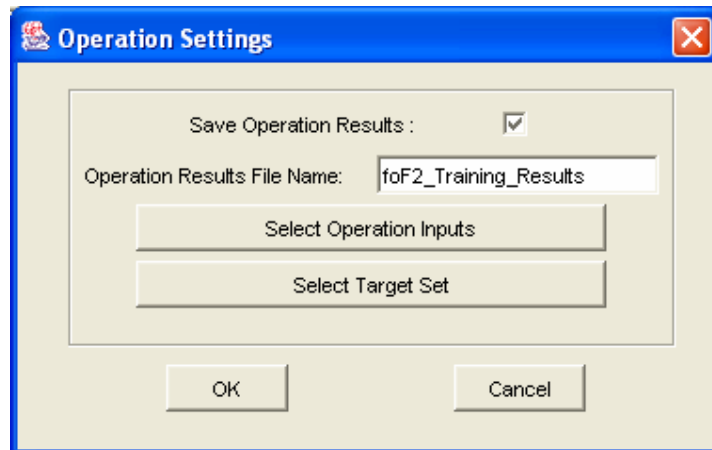


Figure 4.3-9: Operation settings window.

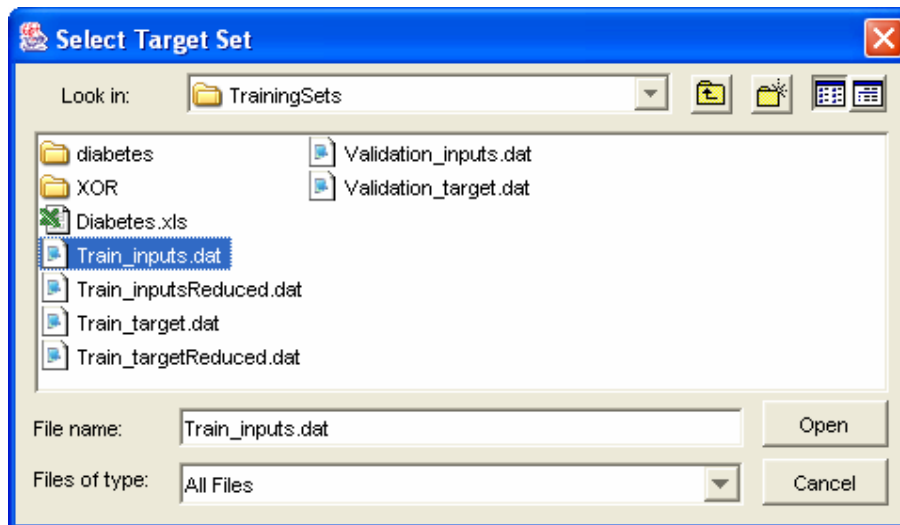


Figure 4.3-10: Browser window.

#### 4.3.2.6 Plotter Module

This module is employed to display training results and operation results with graphs. The training error graph and the scatter diagram can be plotted. Some sample plots are shown below. A training error graph scaled 0.2 is shown in Figure 4.3-11, and a scatter diagram is in Figure 4.3-13.

This plotter module is mainly developed by Joseph A. Huwaldt, and distributed as a free software library. In METU-IFS this library is used by writing some glue

code. This library was not supporting scatter diagrams. So some modifications have been made for scatter diagram plots.

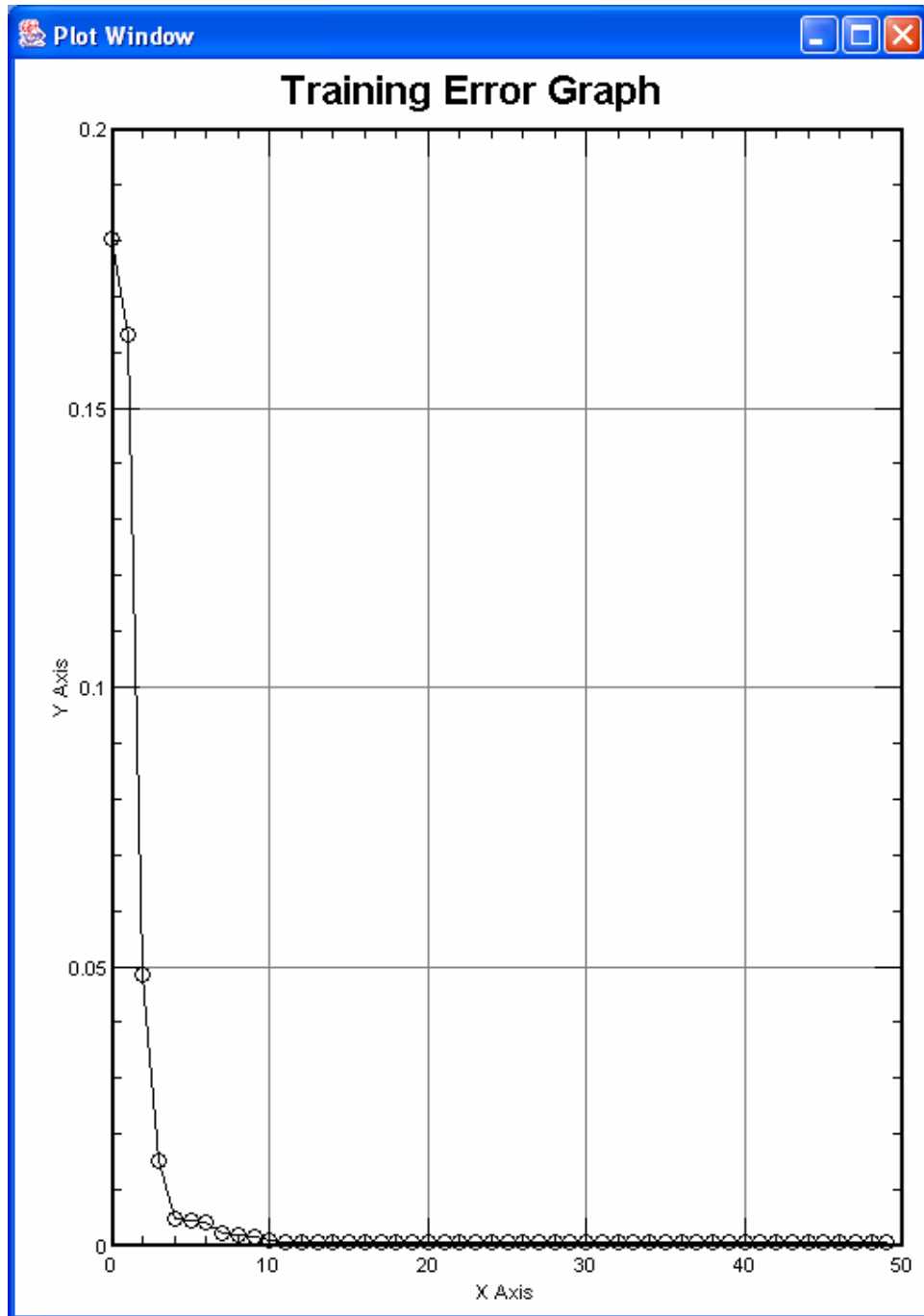


Figure 4.3-11: A training error graph scaled to 0.2.

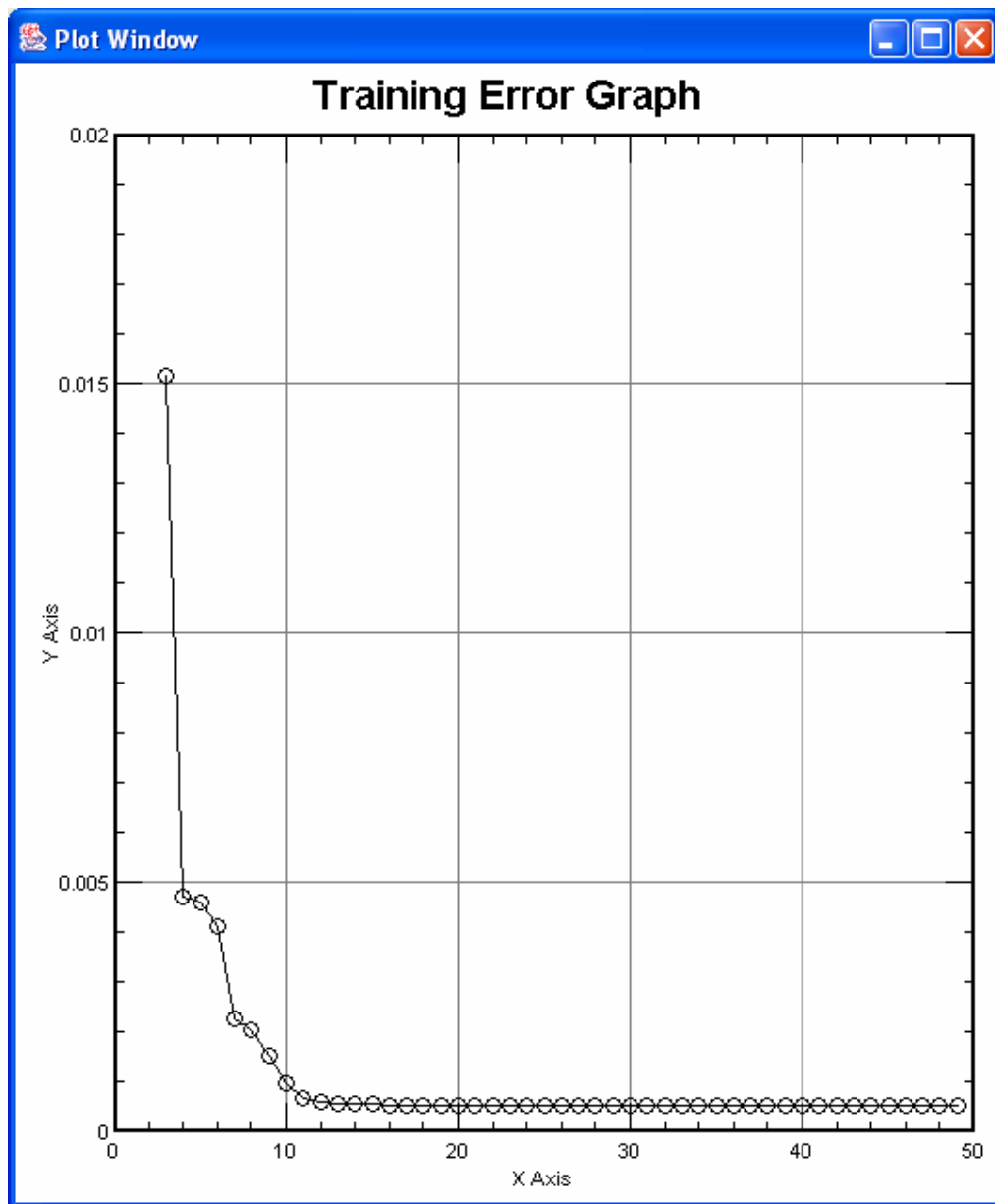


Figure 4.3-12: A training error graph scaled to 0.02.

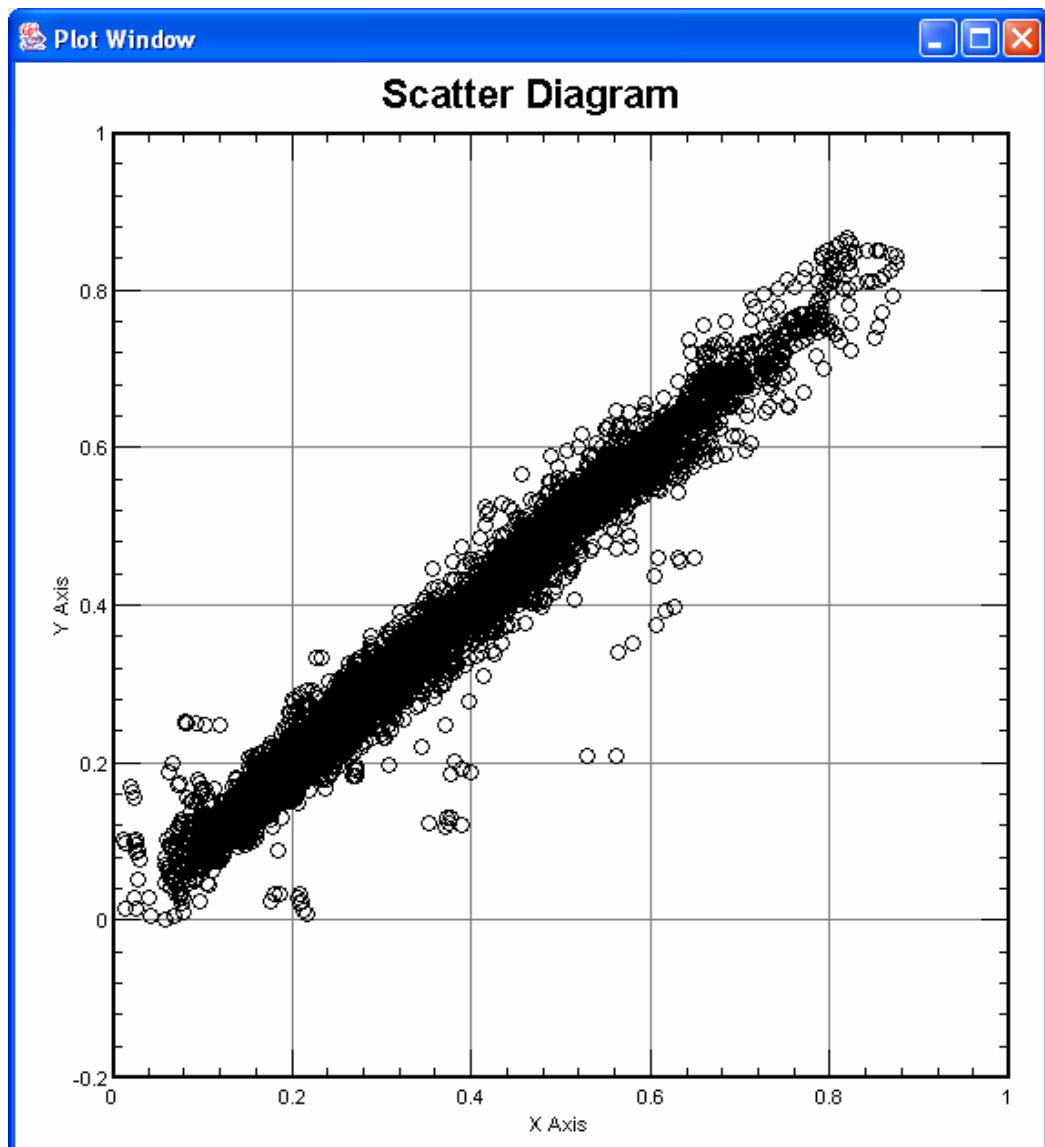


Figure 4.3-13: A scatter diagram window of METU-IFS.

### 4.3.3 Server Side

This section describes the properties of the server side.

#### 4.3.3.1 Main Properties

The server side of METU-IFS is actually the core of whole application. All training algorithms and models are implemented in the server application. When the server is started running, it listens to the predefined port of the server machine.

As soon as a client requests a connection, server creates a thread dedicated to the client, than continues to listen to the port for the other clients. All of the server services are shared by the threads dedicated to the clients. That means, when a service is busy for another client, the client should wait it to be completed.

Although it is not needed, a simple GUI is designed for the server (Figure 4.3-14). The server administrators can use this GUI to monitor, what is going on the server. Besides, new users can be created by using this GUI.

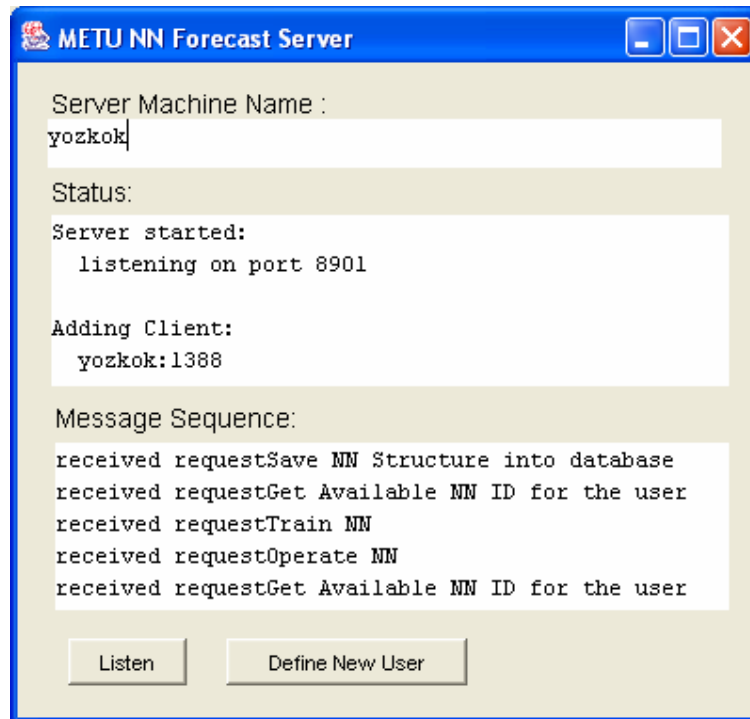


Figure 4.3-14: Main window of server.

#### 4.3.3.2 Listener Module

This module runs in an infinite loop and listens to the socket for if any client is requesting a connection. When a client requests a connection this module catches it and creates a new thread dedicated to this client. The listener module

continues to listen to other clients and a dedicated thread is created to take care of the client.

#### **4.3.3.3 Client Handle Module**

This module is created by Listener module as soon as a client requests a connection. The Client Handler Module runs in an infinite loop as a separated thread until the connection is closed by the client. This module listens to the client's requests and performs requested tasks such as "Authorize", "Save NN", "Train NN", "Get NN List", "Operate NN", "Save Training Results" etc.

#### **4.3.3.4 Training Abstract Module**

The common methods for "TrainLM" and "TrainGD" are implemented in this module while the other algorithm specific methods are stored as abstract methods. The common methods are "BuildNetwork", "UpdateWeights", "RandomizeWeights", "CalculateActivation", "CalculateDerivativeOfActivation", "FindNumOfWeights", "SetWeights", and "OperateWholeSet".

Among these methods, "BuildNetwork" is the most important one. It takes an "NNStructure" type parameter which defines the network structure, and builds the network according to this structure. Since the network information is independent of training algorithm it is hold in this module. Most of the other methods implemented in this module are also network structure related methods. All other training algorithm related methods are implemented in the related training algorithm module.

##### **4.3.3.4.1 Weight Randomization**

This method is called before the training starts. NN weights are randomized in this module. Two kinds of randomization algorithm are implemented. One of them is just randomizes the weights uniformly between -1 and 1. The other randomization algorithm is called as Nguyen-Widrow Randomization algorithm. Nguyen and Widrow developed a randomization formula by analyzing two-layer neural networks to improve the learning speed [62]. This randomization algorithm randomizes the weights depending on the number of neurons in the hidden layer and the number of inputs. Nguyen-Widrow randomization algorithm

is used if the related check box in the "Set Training Parameters" window (Figure 4.3-7) is checked.

#### **4.3.3.5 LM Module**

In this module Levenberg Marquardt training algorithm is implemented as described in the famous paper of Hagan (1994) [38]. In training the first thing to be done is to calculate the Jacobian Matrix which is given in Eq. (2.1.24). The calculation is done in "calculateJacobianMatrix" method. In this method "ForwardPass" and "BackwardPass" methods are called in a loop. For each training set, a forward pass is performed to calculate the error vector, and then a backward pass operation is performed. In "ForwardPass" each neuron's net inputs and outputs are calculated as in Eq.(2.1.5) and Eq.(2.1.6). In "BackwardPass" sensitivity factor (Eq.(2.1.16)) of each neuron is calculated by calling "CalculateSensitivityFactorsLM" method. Then a row of Jacobian matrix is built in "BackwardPass". So for each training set, a row of Jacobian matrix is built.

After the completion of calculating Jacobian matrix, the code enters into a loop. In this loop, performance of the network will be calculated. The loop starts by calculating the initial performance of the network and updating the weights temporarily. If the sum of squares of errors, is not decreased in a turn, then momentum is decreased by beta. For each successful turn (i.e. sum of squares of errors is decreased) "DW" matrix is updated that is weights are updated as Eq. (2.1.27). If it's not a successful turn, momentum is decreased by beta and "try\_number" is decremented. If the "try\_number" exceeds the MAXTRYNUMBER, then regardless the turn is successful or not, weights are updated. That is the tricky point which is not described in the algorithm [38]. If the algorithm is implemented as it is described, then the program may be blocked sometimes. Finally, completion of the loop with a successful turn ends the epoch.

In this module there are much more sub methods which are not mentioned here, but the idea is as described.



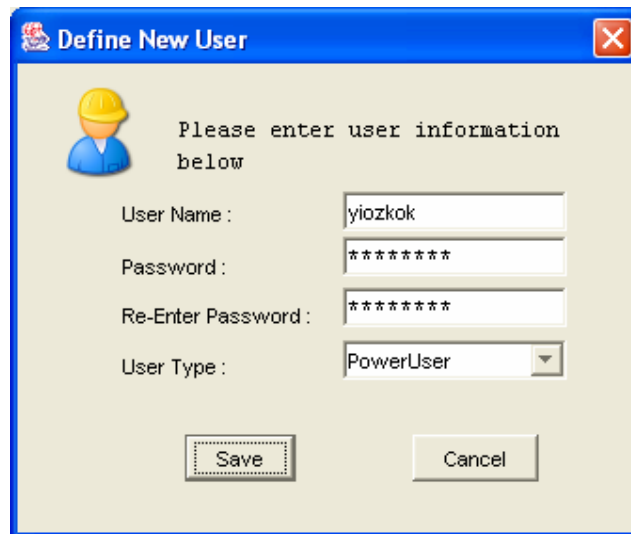
#### 4.3.3.6 GD Module

In this module Backpropagation training algorithm is implemented as described in the famous paper of Hagan (1994) [38]. The implementation of Backpropagation is straight forward. Batch and online training issue is also handled in this module. If "online" training is selected, then after each set operated, the weights are updated. If "online" training is not selected, the errors are added incrementally until all sets are operated, and then the weights are updated as in Eq.(2.1.11).

The main methods of this module are "Train", "ForwardPass", "CalculateSensitivityFactorsGD", and "CalculateDeltaWeights".

#### 4.3.3.7 New User Definition Module

A server administrator can define new users by using this module. This module has a GUI (Figure 4.3-15) to easily define new users. A user is defined by its "user name", "user type" and with a temporary password.



The image shows a Windows-style dialog box titled "Define New User". It contains a yellow hard hat icon and the text "Please enter user information below". There are four input fields: "User Name" with the value "yiozkok", "Password" with "\*\*\*\*\*", "Re-Enter Password" with "\*\*\*\*\*", and "User Type" with a dropdown menu showing "PowerUser". At the bottom are "Save" and "Cancel" buttons.

Figure 4.3-15: The window to define a new user.

#### **4.3.4 Database Design**

METU-IFS has a database on the server PC to store various data. METU-IFS database is developed on MS Access Environment. It is assumed that Server application will run on a MS Windows operating system. METU-IFS Java program is then connected to this database over JDBC-ODBC Bridge.

In this database different kinds of data are stored in 5 different tables. These tables are named as follows:

|                         |  |
|-------------------------|--|
| NNSTRUCTURETABLE        | : Stores neural network structure related data |
| TRAININGPARAMETERSTABLE | : Stores training parameters                   |
| TRAININGRESULTSTABLE    | : Stores training results                      |
| WEIGHTSTABLE            | : Stores weights                               |
| USERSINFO               | : Stores information of users                  |

New entries for NNSTRUCTURETABLE, TRAININGPARAMETERSTABLE and USERINFO can be stored into the database independently. But TRAININGRESULTS and WEIGHTSTABLE entries are dependent on other three tables. Basic design decisions for each table are given in the subsections.

##### **4.3.4.1 USERINFOTABLE**

In this table user information is stored. When a new user is defined by administrator, a new entry is created in this table. To create a new user UserName, TempPassword and UserType should be entered by administrator. "isActivated" field is set to "false" at the beginning. As soon as the client is connected to the server a password sent by client application is set into Password field besides setting "isActivated" field to "false". Data types and field details are shown below in Table 4.3-2.

Table 4.3-2: USERINFO database table

| Field Name   | Data Type | Description  |
|--------------|-----------|--|
| UserName     | Text      | A unique User Name.                                  |
| TempPassword | Text      | Temporary password. 64bit hashed.                    |
| Password     | Text      | Password. 64 bit hashed.                             |
| isActivated  | Boolean   | Status flag to hold if user activated or not.        |
| UserType     | Text      | User Type; "PowerUser", "OrdinaryUser", "Researcher" |

#### 4.3.4.2 NNSTRUCTURETABLE

In this table neural network structures are stored. Each NN table is labeled with an ID which is generated on runtime by METU-IFS and a name which is given by user. Each field of this table, data types and field details are shown below in Table 4.3-3 and a sample entry is given for this table below in Table 4.3-4.

Table 4.3-3: NNSTRUCTURETABLE database table

| Field Name        | Data Type    | Description   |
|-------------------|--------------|---|
| ID                | Long Integer | ID number of the entry starting from 1.                   |
| NNStName          | Text         | NN Structure Name that is optionally defined by user      |
| TrainingStatus    | Boolean      | A status flag, holds if the nn structure is trained once. |
| InputNumber       | Integer      |   |
| OutputNumber      | Integer      |   |
| IncludeBias       | Boolean      |   |
| HiddenLayerNumber | Integer      |   |
| NeuronNumberInHL1 | Integer      |   |
| NeuronNumberInHL2 | Integer      |   |
| UserType          | Text         |   |

Table 4.3-4: A sample entry for NNSTRUCTURETABLE table

| NNSTRUCTURETABLE |           |             |             |              |             |                   |                   |                   |
|------------------|-----------|-------------|-------------|--------------|-------------|-------------------|-------------------|-------------------|
| ID               | NNStName  | TrainStatus | InputNumber | OutputNumber | IncludeBias | HiddenLayerNumber | NeuronNumberInHL1 | NeuronNumberInHL2 |
| 1                | foF2Train | No          | 8           | 1            | No          | 1                 | 4                 | 0                 |
| 2                | XOR       | Yes         | 2           | 1            | No          | 1                 | 3                 | 0                 |

#### 4.3.4.3 TRAININGPARAMETERSTABLE

In this table all training parameters are stored for each training. Each field of this table is described by data types and field names, as shown below in Table 4.3-5.

Table 4.3-5: TRAININGPARAMETERSTABLE database table

| Field Name        | Data Type    | Description  |
|-------------------|--------------|--|
| ID                | Long Integer | ID number of the entry starting from 1.  |
| Name              | Text         | Entry Name that is optionally defined by user  |
| Algorithm         | Text         | "LM" Stands for "Levenberg Marquardt", "GDB" Stands for "Gradient Descent" with batch, "GDO" for "Gradient Descent" with online. |
| OutputLayerAF     | Text         | "LINEAR", "SIGMOID" or "TANSIG"  |
| HiddenLayerAF     | Text         | "LINEAR", "SIGMOID" or "TANSIG"  |
| BiasValue         | Double       |  |
| EpochNumber       | Integer      |  |
| Momentum          | Double       |  |
| LearningRate      | Double       | Available only when GD* selected.  |
| ScaleFactor       | Double       | Available only when LM* selected.  |
| TargetError       | Double       |  |
| UseRandAlgorithm  | Boolean      | If "yes" Nguyen-Widrow Randomization algorithm is used.  |
| UseValidationStop | Boolean      |  |

#### 4.3.4.4 TRAININGRESULTSTABLE

In this table training results are stored for each training. Each field of this table is described by data types and field names, as shown below in Table 4.3-6 and a sample entry is given for this table below in Table 4.3-7.

Table 4.3-6: TRAININGRESULTSTABLE database table

| Field Name           | Data Type    | Description   |
|----------------------|--------------|---|
| ID                   | Long Integer | ID number of the entry starting from 1.                   |
| Name                 | Text         | Entry Name that is optionally defined by user             |
| NNStructureID        | Long Integer | Trained NNStructure ID                                    |
| TrainingParametersID | Long Integer | ID of TrainingParameters used for training                |
| WeightsID            | Long Integer | Weights ID related with the training results              |
| Performance          | Double       | Final performance value                                   |
| EpochNo              | Integer      | Last epoch number before training ended.                  |
| StopType             | Integer      | 2 - Succeeded, 3 - Validation Stop, 1 - Epoch No Exceeded |
| AvarageEpochDuration | Integer      | integer value in milliseconds                             |

Table 4.3-7: A sample entry for NNSTRUCTURETABLE table

| TRAININGRESULTSTABLE |          |                |                      |           |               |         |          |                      |
|----------------------|----------|----------------|----------------------|-----------|---------------|---------|----------|----------------------|
| ID                   | Name     | NNStructure ID | TrainingParametersID | WeightsID | Performance   | EpochNo | StopType | AvarageEpochDuration |
| 1                    | foF2_Tr1 | 1              | 1                    | 1         | 5.6127531E-04 | 10      | 3        | 5573                 |

#### 4.3.4.5 WEIGHTSTABLE

In this table, weights of each trained neural network are stored. Each field of this table is described by data types and field names, as shown below in Table 4.3-8.

Table 4.3-8: WEIGHTSTABLE database table

| Field Name  | Data Type    | Description                             |
|-------------|--------------|---|
| ID          | Long Integer | ID number of the entry starting from 1. |
| WeightValue | Double       |   |

WEIGHTSTABLE design is a bit tricky. It is described below with an example. Assume that we have two training results for two different neural networks; one of them has 6 weights and other has 4 weights. Then we have to have two different weight sets. These weight sets are stored in WEIGHTSTABLE as seen in Table 4.3-9.

Table 4.3-9: WEIGHTSTABLE database table

| WEIGHTSTABLE |                    |
|--------------|--------------------|
| ID           | WeightValue        |
| 1            | -0.802363337698952 |
| 1            | -0.397536062176062 |
| 1            | 0.118202085389262  |
| 1            | -0.164606526196443 |
| 1            | 0.536329574689754  |
| 1            | 0.234299343344433  |
| 2            | 0.039930293493029  |
| 2            | 0.299384872456362  |
| 2            | -0.112343876483726 |
| 2            | 0.548938294493828  |

#### **4.3.5 Development Environment**

This section describes the development environment used while developing METU-IFS.

##### **4.3.5.1 Software Environment**

This application is developed on Borland Java Builder 9.0 Integrated Development Environment (IDE). The database is developed on MS Office Access. MS Windows XP is used as an operating system.

##### **4.3.5.2 Hardware Environment**

This application is developed on an x386 architecture with an AMD2500 CPU and 512 MBs of Ram.

## **CHAPTER 5**

### **CONCLUSION**

In this thesis, the previous studies on Ionospheric Forecasting are reviewed, contributing greatly to our knowledge we need to develop a successful application of our own. The realization of putting the models into service over the Internet is demonstrated with this application, the models of which we have developed successfully and those developed previously. Some security concerns are also taken into account during the development. That is, when the models are in service over the Internet, it is ensured that the intellectual assets are safely kept in a secure environment. A manageable database is developed besides the application. All the data are stored in that database, enabling us to further analyze them on various environments such as MS Excel or MATLAB. Design considerations for both the database and the application are given in Chapter 4. The application is realized in full conformity with all the requirements stated in the SRS document (Appendix-A).

The models and neural network algorithms are implemented efficiently. METU-IFS is verified by comparing it with MATLAB, obtaining even slightly better results, and thus verifying the correctness of the implementation of the algorithms as well. This comparison is shown in Chapter 2.

An attempt is made to develop new Ionospheric Forecasting models based on neurofuzzy systems, and their applicability on ionospheric forecasting is discussed in Chapter 3. After a review of two neurofuzzy system ASMOD and ANFIS, METU-NFN system is introduced. METU-NFN is developed as the first attempt of a neurofuzzy system that has ever been applied to ionospheric



studies. Operating the developed neurofuzzy systems, reasonable results are obtained. Some variations of NN and NFN models are compared. The results of neurofuzzy models developed during this study seem to be less successful and remain behind the results obtained by NN models at the first glance. However, considering that the success of a neurofuzzy system is directly dependent on the interpretation of the physical infrastructure of the problem and considering that the application of neurofuzzy models to the ionospheric studies is quite young yet, the results should be accepted as quite successful. The comparison results showed that neurofuzzy systems are applicable to the ionospheric studies. A little more detailed study on the physics of the ionosphere to develop a neurofuzzy system may result in much better performance values.

For further studies, METU-IFS application can be improved by adding new models in future studies. Moreover, the feedback from users can also contribute to make further improvement on the user interface. It is also possible to build online learning capabilities into the models.

Eliminating ineffective inputs while imposing more specified expert knowledge, METU-NFN models can be improved further. For a very early work seasonal variations can be studied to add new fuzzy inputs to the METU-NFN model. Then some other indexes or parameters having an influence on TEC or  $f_oF2$  should be studied. Moreover, completely different neurofuzzy models may be developed to obtain better results. That is the fact that the model developed in this thesis is a preliminary work.

Although the neural network methods are widely used and produced promising results, and neurofuzzy methods tend to be developed as a second alternative with some apparent advantages, the horizons can further be broadened by trying some other newly developing methods like genetic algorithms.

## REFERENCES

1. O. Huwendiek, W. Brockmann, "On the applicability of the NetFAN-approach to function approximation Fuzzy Systems," Proceedings of the Sixth IEEE International Conference on Fuzzy Systems, volume 1, 1-5 July 1997, pages 477 – 482, 1997.
2. M. Brown, K. M. Bossley, D. J. Mills, and C. J. Harris, "High dimensional neurofuzzy systems: overcoming the curse of dimensionality," Proceedings of IEEE International Conference on Fuzzy Systems, volume 4, pages 2139 – 2146, Yokohama, Japan, 1995.
3. E. Tulunay, "Non-Linear Systems and with Reference to Some Geophysical Processes," EGS XXIV General Assembly, The Hague, ND, 19-23 April 1999.
4. A. J. Gibson, *et al.*, "Characteristics of Fading of HF Signal and Noise Intensities on Three Paths Between the UK and Turkey," Radio Science, volume 30, no. 3, pages 649-658, May-June 1995.
5. L. J. Lanzerotti, D. J. Thomson, C. G. MacLennan, "Engineering Issues in Space Weather," Modern Radio Science, Oxford University Press, 1999.
6. G. Siscione, "The Space Weather Enterprise: Past, Present, and Future," Atmospheric and Solar-Terrestrial Phys., volume 62, pages 1223-1232, 2000.
7. O. Altinay, "Nonlinear Blackbox Modeling of Ionospheric Critical Frequency Process Using Neural Networks," MSc. Thesis, Middle East Technical University, Department of Electrical and Electronics Engineering, Ankara, Turkey, June 1996.
8. E. Tulunay, Y. Tulunay, C. Ozkaptan, E. T. Senalp, M. Aydogdu, O. Ozcan, E. Guzel, Y. Aydogdu, A. Yesil, I. Unal, M. Canyilmaz, E. Ipekcioglu, "Two Solar Eclipses Observations in Turkey," IL NUOVO CIMENTO, Nuovo Cimento Della Societa Italiana Di Fisica C-Geophysics and Space Physics, 25 C, N.2, pages 251-258, Publisher: Editrice Compositori Bologna, Siena, March – April 2002.
9. E. Ertac, Y. Tulunay, "Ionospheric Total Electron Content Measurements from Turkey During the Solar Eclipse of 29 April 1976," Advanced Study Institute on Dynamical, Neutral and Ionized Atmospheres, Nord Torpa, Norway, 1979.
10. C. Ozkaptan, "Modeling of Ionospheric Propagation in the HF Band Using Neural Networks: HF Propagation Modeling," MSc. Thesis, Middle East

Technical University, Department of Electrical and Electronics Engineering, Ankara, Turkey, December 1999.

- 11.** E. T. Senalp, E. Tulunay, Y. Tulunay, C. Ozkaptan, "Neural Network Based Signal to Noise Ratio Prediction of a HF Radio Link Using Temporal and Geophysical Parameters," Nordic HF 01, Faro, Sweden, 2001.
- 12.** E. Tulunay, "Non-Linear Systems and with Reference to Some Geophysical Processes," EGS XXIV General Assembly, The Hague, ND, 19-23 April 1999.
- 13.** H. Lundstedt, "AI Techniques in Geomagnetism Storm Forecasting," Magnetic Storms, AGU Geophysical Monograph 97, Washington, USA, 1997.
- 14.** O. Altinay, E. Tulunay, Y. Tulunay, "Prediction of Ionospheric Critical Frequency Using Neural Networks," COST 251 (Improved Quality of Ionospheric Telecommunication Systems Planning and Operation), 3rd Management Committee Meeting, COST 251 TD (96)016, Erice, Italy, 5-9 March 1996.
- 15.** Y. Tulunay, "Neural Network Based foF2 Process Modeling," Report submitted to WG3 Development of Models, WP 3.4.4 Forecast Models, 3rd Management Committee Meeting, COST 251 DOC 3052, Erice, Italy, 5-9 March 1996.
- 16.** L. R. Cander, X. Lamming, "Neural Networks in Ionospheric Prediction and Short Term Forecasting," 10<sup>th</sup> International Conference on Antennas and Propagation, Edinburgh, 14-17 April 1997, IEE Conference Publications, 436, 2.27,2.30, 1997.
- 17.** N. M. Francis, A. G. Brown, A. Akram, P. S. Cannon, D. S. Broomhead, "Non-Linear Prediction of the Ionospheric Parameter foF2," AI Applications in Solar-Terrestrial Physics, Lund, Sweden, 29-31 July 1997, ESA WPP-148, pages 219-223, 1998.
- 18.** X. Lamming, L. R. Cander, "Appropriateness of the Neural Network Approach for Monthly Median Ionospheric Prediction," COST 251 4<sup>th</sup> Management Committee Meeting, COST 251 DOC 402, September 1996.
- 19.** L. A. Willisroft, A. W. V. Poole, "Neural Networks, foF2, Sunspot Number and Magnetic Activity," Geophysical Researches Letters 23, pages 3659-3662, 1996.
- 20.** A. H. Bilge, Y. Tulunay, "A Novel On-Line Method For Single Station Prediction and Forecasting of the Ionospheric Critical Frequency foF2 One Hour Ahead," Geophysical Research Letters, volume 27, no. 9, pages 1383-1386, May 1, 2000.
- 21.** L. F. Alberca, G. Juchnikowski, S. S. Kouris, A. V. Mikhailov, V. V. Mikhailov, G. Miro, B. Morena, P. Muhtarov, D. Pancheva, G. J. Sole, I. Stanilawska, T. Xenos, "Comparison of Various foF2 Single Station Models for European Area," Acta Geophys. Pol., volume 47, pages 42-56, 1999.
- 22.** L. R. Cander, "Artificial Neural Network Applications In Ionospheric Studies," Annali di Geofisica, volume 41, pages 757-766, 1998.

23. I. Stanislawska, "A Single Station Prediction Models as a Contribution to Instantaneous Mapping," *Annali di Geofisica*, volume 37, pages 153-157, 1994.
24. A. H. Bilge, Y. Tulunay, F. Ozdemir, "Semi Empirical Single Station Modeling of foF2 Variations: Spectral Analysis," Poster presented at the 23rd General Assembly of European Geophysical Society, Nice, France, 1998.
25. E. T. Senalp, "Neural Network Based Forecasting For Telecommunications Via Ionosphere," MSc. Thesis, Middle East Technical University, Department of Electrical and Electronics Engineering, Ankara, Turkey, August 2001.
26. Y. Tulunay, E. Tulunay, E. T. Senalp, "The Neural Network Technique-1: A General Exposition, *Advances in Space Research*," Publisher: Elsevier, volume 33/6 pages 983-987, 2004.
27. Y. Tulunay, E. Tulunay, E. T. Senalp, "The Neural Network Technique-2: An Ionospheric Example Illustrating Its Application," *Advances in Space Research*, Publisher: Elsevier, volume 33/6 pages 988-992, 2004.
28. T. Sayed, and A. Razavi, "Comparison of neural and conventional approaches to mode choice analysis," *J. Comput. Civ. Eng.*, volume 14(1), pages 23-30, 2000.
29. S. Haykin, "Neural Networks: A Comprehensive Foundation," Second Edition, Prentice-Hall Inc., pages 2, 83-84, 169, 215, 1999.
30. E. Tulunay, "Introduction to Neural Networks and Their Applications to Process Control," one chapter in the book, *Neural Networks Advances and Applications*, pp.241-273, Ed. E. Gelenbe, Elsevier Science Publishers B. V., North Holland, 1991.
31. R. P. Lippmann, "An Introduction to Computing with Neural Networks," *IEEE ASSP Magazine*, pages. 4-22, 1987.
32. G. G. Lorentz, "The 13th problem of Hilbert," *Proceedings of Symposia in Pure Mathematics*, volume 28, pages 419-430, 1976.
33. S. M. Sulzberger, N. N. Tschicholg-Gurman, S. J. Vestli, "FUN: Optimization of Fuzzy Rule Based Systems Using Neural Networks," In *Proceedings of IEEE Conference on Neural Networks*, San Francisco, pages 312-316, March 1993.
34. O. Altinay, E. Tulunay, Y. Tulunay, "Forecasting of Ionospheric Critical Frequency Using Neural Networks," *Geophysical Research Letter*, pages 1467-1470, June 1997.
35. Y. Tulunay, E. Tulunay, E. T. Senalp, "An Attempt to Model the Influence of the Trough on HF Communication by Using Neural Network," *Radio Science*, volume 36, no. 5, pages 1027-1041, Publisher: American Geophysical Union, Washington, September – October 2001.
36. A. Kumluca, E Tulunay, I. Topalli, Y. Tulunay, "Temporal and spatial forecasting of ionospheric critical frequency using neural networks," *Radio Science* volume 34 (6), pages 1497-1506, 1999.

37. E. Tulunay, C. Ozkaptan, Y. Tulunay, "Temporal and spatial forecasting of the foF2 values up to twenty-four hours in advance," *Phys. Chem. Earth (C)* volume 25 (4), pages 281–285, 2000.
38. M. T. Hagan, M. B. Menhaj, "Training Feed Forward Networks with the Marquardt Algorithm," *IEEE transactions on Neural Networks*, volume 5(6), 989-993, November 1993.
39. T. Sayed, A. Tavakoli, A. Razavi, "Comparison of Adaptive Network Based Fuzzy Inference Systems and B-spline Neuro-Fuzzy Mode Choice Models," *Journal of Computing in Civil Engineering*, April 2003.
40. J. -S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference systems," *IEEE Trans. Systems, Man and Cybernetics*. volume 23 (03) pages 665–685, 1993.
41. C. -C. Lee, "Fuzzy logic in control systems: fuzzy logic controller-part 1," *IEEE Trans. on Systems, Man, and Cybernetics*, volume 20(2), pages 404–418, 1990.
42. C. -C. Lee, "Fuzzy logic in control systems: fuzzy logic controller-part 2," *IEEE Trans. on Systems, Man, and Cybernetics*, volume 20(2) pages 419–435, 1990.
43. Y. Tsukamoto, "An approach to fuzzy reasoning method," In Madan M. Gupta, Rammohan K. Ragade, and Ronald R. Yager, editors, *Advances in Fuzzy Set Theory and Applications*, pages 137–149, North-Holland, Amsterdam, 1979.
44. T. Takagi and M. Sugeno, "Derivation of fuzzy control rules from human operator's control actions," *Proc. Of the IFAC Symp. on Fuzzy Information, Knowledge Representation and Decision Analysis*, pages 55–60, July 1983.
45. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, volume 2, pages 183–192, 1989.
46. A. Abraham, "Neuro Fuzzy Systems: State-of-the-art Modeling Techniques," In *Proceedings of the Sixth International Work Conference on Artificial and Natural Neural Networks, IWANN 2001*, Granada, Springer Verlag Germany, June 2001.
47. H. R. Bherenji, and P. Khedkar, "Learning and Tuning Fuzzy Logic Controllers through Reinforcements," *IEEE Transactions on Neural Networks*, volume 3, pages 724-740, 1992.
48. C. T. Lin, C. S. G. Lee, "Neural Network based Fuzzy Logic Control and Decision System," *IEEE Transactions on Comput.*, volume 40 (12) pages 1320-1336, 1991.
49. R. Jang, "Neuro-Fuzzy Modeling: Architectures, Analyses and Applications," PhD Thesis, University of California, Berkeley, July 1992.
50. C. F. Juang, C. T. Lin, "An Online Self Constructing Neural Fuzzy Inference Network and its Applications," *IEEE Transactions on Fuzzy Systems*, volume 6, no. 1, pages 12-32, 1998.

51. S. Tano, T. Oyama, T. Arnould, "Deep combination of Fuzzy Inference and Neural Network in Fuzzy Inference," *Fuzzy Sets and Systems*, volume 82 (2) pages 151-160, 1996.
52. N. Kasabov, and Q. Song, "Dynamic Evolving Fuzzy Neural Networks with 'm-out-of-n' Activation Nodes for On-line Adaptive Systems," Technical Report TR99/04, Department of information science, University of Otago, 1999.
53. A. Abraham, B. Nath, "Evolutionary Design of Neuro-Fuzzy Systems – A Generic Framework," In Proceedings of the 4th Japan – Australia joint Workshop on Intelligent and Evolutionary Systems, Japan, November 2000.
54. T. Kavli, "ASMOD, an algorithm for Adaptive Spline Modelling of Observation Data," *International Journal of Control*, volume 58, pages 947-967, 1993.
55. M. Brown, and C. Harris, "Neurofuzzy Adaptive Modelling and Control," Prentice Hall, New York, 1994.
56. M. Brown, and C. Harris, "A perspective and critique of adaptive neurofuzzy systems used for modeling and control applications," *Int J Neural Systems*, volume 6(2) pages 197–220, 1995.
57. J. Jang, and C. Sun, "Neuro-fuzzy modeling and control," *Proc. IEEE*, volume 83(3), pages 378–406, 1995.
58. C. Von Altrock, "Fuzzy Logic and Neurofuzzy Applications Explained," Prentice Hall, 1995.
59. L. Jouffe, "Fuzzy Inference System Learning by Reinforcement Methods," *IEEE Transactions on Systems, Man, And Cybernetics – Part C: Applications And Reviews*, volume 28, no. 3, pages 338-354, August 1998.
60. COST271 WG 4 STM, 2002. Effects of the Upper Atmosphere on Terrestrial and Earth-Space Communications, Short term scientific mission (STM) work on the TEC data available at RAL to organize the input data for the METU NN model by Mr. E.T. Senalp under the supervision of Dr. Lj. Cander who provided the GPS data during the STM as a joint action between UK and Turkey, Terms of Reference, COST 271 Action, WG 4, 30 June 2002 – 7 July 2002, RAL, Chilton, Didcot, U.K.
61. Webster online dictionary, <http://www.webster-dictionary.org>, last accessed on 18/01/2005.
62. D. Nguyen, B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," *Proc. of the Int. Joint Conference on Neural Networks*, volume 3, pages 21–26, 1990.

## **APPENDIX A**

# **METU IONOSPHERIC FORECASTING SOFTWARE SOFTWARE REQUIREMENTS SPECIFICATIONS DOCUMENT**

**Prepared By**

Yusuf İbrahim Özkök

**Reviewed By**

Prof Dr. Ersin Tulunay

Erdem Türker Şenalp



### Revision History

\*A - Added C - Changed D - Deleted

| No | Revision Date | Location | *   | Comments              | Rev |
|----|---------------|----------|-----|-----------------------|-----|
| 1  | 01/12/03      |          |     | First Release         | AA  |
| 2  | 25/06/04      | All      | ACD | Revised after reviews | AB  |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |
|    |               |          |     |                       |     |

## CONTENTS

|  |     |
|--|-----|
| 1. SCOPE.....                                      | 108 |
| 2. REFERENCE DOCUMENTS .....                       | 108 |
| 3. REQUIREMENTS.....                               | 108 |
| 3.1. Modes .....                                   | 108 |
| 3.2. Unit Requirements.....                        | 109 |
| 3.2.1. Training Mode Requirements.....             | 109 |
| 3.2.2. Validation Requirements.....                | 110 |
| 3.2.3. Operation Mode Requirements.....            | 110 |
| 3.2.4. User Interface Requirements .....           | 111 |
| 3.2.5. Database Requirements.....                  | 111 |
| 3.3. EXTERNAL INTERFACE REQUIREMENTS.....          | 111 |
| 3.4. INTERNAL INTERFACE REQUIREMENTS.....          | 111 |
| 3.5. INTERNAL DATA REQUIREMENTS .....              | 112 |
| 3.6. EXTERNAL DATA REQUIREMENTS .....              | 113 |
| 3.7. SECURITY REQUIREMENTS .....                   | 113 |
| 3.8. COMPUTER RESOURCES REQUIREMENTS .....         | 114 |
| 3.8.1. Computer Resources Usage Requirements ..... | 114 |
| 3.8.2. Communication Requirements .....            | 114 |
| 3.9. LOGISTIC REQUIREMENTS.....                    | 114 |
| 3.10. OTHER REQUIREMENTS .....                     | 114 |
| 3.11. PRIORITIES OF REQUIREMENTS.....              | 114 |
| 4. OBSERVABILITY AND EVALUATION TECHNIQUES.....    | 114 |
| 5. ABBREVIATIONS .....                             | 114 |
| 6. ATTACHMENTS .....                               | 115 |
| 6.1. Database Structure.....                       | 115 |
| 6.2. User Interface .....                          | 116 |
| 6.3. Use Cases Message Sequence Charts.....        | 118 |

## 1. SCOPE

This document defines the requirements of “Metu NeuroFuzzy Ionospheric Forecasting System Software”

Project No :

Project Name : Metu NeuroFuzzy Ionospheric Forecasting System Software Project

Software Name: Metu NeuroFuzzy Ionospheric Forecasting System Software

This software will perform training-validation and operation processes of a fuzzy neural network (FNN) system for the sake of forecasting most popular parameters of ionosphere which are TEC and foF2 values.

At the rest of this document the name of this software will be called as “MNFIFS”

## 2. REFERENCE DOCUMENTS

| No | Reference Name | Document No | Rev | Date | Location |
|----|----------------|-------------|-----|------|----------|
| 1  |                |             |     |      |          |

## 3. REQUIREMENTS

### 3.1. Modes

This Software will have three major operation modes. These modes are;

- User Login Mode
- Training and Validation Mode
- Operation Mode

Users will login to the system by using “User Login Mode” Each user will have unique ID. There will be three types of users. These types are; Admin User, Super user, Normal User. An admin user can use all properties of the software. Super users can define new neural network structures and can train and validate them. They also can use the Operation Mode. Normal users can only use the Operation Mode.

In the training mode user will build a neural network architecture by using selectable architectural elements. After building the network, user will give the data sets for training and validation. Then software will perform the training process. By the completion of the training phase the values of the weights will be finalized and saved in the name of architecture.

In the operation mode user will select one of the previously trained and validated neural network structures. Then user will give an input data set which is arranged for the selected network. The network will calculate an output, by using this data set.

The software has a secure and a non secure part. The non secure part is the user interface part. The secure part is the part which contains all neurofuzzy algorithms and works to generate outputs.

### 3.2. Unit Requirements

The software is divided into logical units. Requirements for the software are grouped in these units.

#### 3.2.1. Training Mode Requirements

1. MNFIFS software will enable user to build a fuzzy neural network structure.
2. MNFIFS software will restrict the user to build only FeedForward Networks.
3. MNFIFS software will enable the user to select number of inputs of the neural network.
4. MNFIFS software will restrict the number of inputs with 14 inputs at most.
5. MNFIFS software will enable the user to select the number of layers.
6. MNFIFS software will enable to design a neural network with at most 3 hidden layers and at most 7 neurons in each layer.
7. MNFIFS software will enable user to select the type of output. These outputs would represent TEC or foF2.
8. MNFIFS software will enable to design an architecture with at most one output at the same time.
9. MNFIFS software will list all possibly selectable activation functions to the user.
10. MNFIFS software will list the following activation functions to the user;
  - Sigmoidal activation :  $g(a) = [1 + \exp(-a)]^{-1}$
  - Tansig activation :  $TANSIG(n) = [2 / (1 + \exp(-2n))] - 1$
  - Pure Linear activation :  $PURELIN(n) = a.n$

11. MNFIFS software will enable the user to select one of the listed activation functions.
12. MNFIFS software will enable the user to select different activation functions for each layer.
13. MNFIFS software will use supervised learning algorithm for training.
14. MNFIFS software will need the data for training.
15. MNFIFS software will enable the user to enter the training data in the text format.
16. MNFIFS software will accept only predefined formats, which is defined in part “3.5. External Data Requirements”, for data entry.
17. MNFIFS software will accept only correctly arranged data sets for the constructed neural network.
18. MNFIFS software will adjust the weights between the neurons by using Backpropagation rule with Levenberg Marquardt Learning algorithm.
19. MNFIFS software will use the epoch number which is entered by user in an edit box.
20. MNFIFS software will use the initial learning rate which is entered by user in an edit box.
21. MNFIFS software will enable the user to initialize the weights.

### **3.2.2. Validation Requirements**

22. MNFIFS software will enable the user to enter a validation data set.
23. MNFIFS software will automatically stop training when the validation error begins rising while training error is decreasing.

### **3.2.3. Operation Mode Requirements**

24. MNFIFS software will enable the user to select one of the previously trained and validated Fuzzy Neural Networks.
25. MNFIFS software will list available inputs for the selected architecture.
26. MNFIFS software will operate with the inputs which will be entered by the user.
27. MNFIFS software will calculate the output by using Backpropagation Algorithm.

28. MNFIFS software will calculate a single output for each neural network architecture.

#### **3.2.4. User Interface Requirements**

29. MNFIFS software will have a user friendly GUI based on Java Swing class.
30. MNFIFS software GUI drafts are explained in the attachments of this document (6.2).

#### **3.2.5. Database Requirements**

31. MNFIFS software will access to a database which is defined in the attachments of this document in “Database Structure”.
32. MNFIFS software will save user information in a database.
33. MNFIFS software will recall user information from the database.
34. MNFIFS software will save FNN architectures in a database with the architect ID.
35. MNFIFS software will save completed and uncompleted architectures in separate tables.
36. MNFIFS software will save results of operations related with the used architecture and with inputs and operator ID, in a table.

### **3.3. EXTERNAL INTERFACE REQUIREMENTS**

37. MNFIFS software user interface side will get training and validation data files.

### **3.4. INTERNAL INTERFACE REQUIREMENTS**

38. MNFIFS software will have a communication interface between user interface side and server side.
39. MNFIFS software user interface side and server side will communicate by using TCP/IP protocol.
40. MNFIFS software user interface side will send user login information to server side.
41. MNFIFS software user interface side will send the built FNN architecture data to the server side.

- 42. MNFIFS software user interface side will send a data file for training and validation purposes to the server side.
- 43. MNFIFS software user interface side will send inputs to for the selected FNN architecture for operation to the server side.
- 44. MNFIFS software server side will send user information to the user interface side.
- 45. MNFIFS software server side will send the IDs of the trained and non trained FNN architectures.
- 46. MNFIFS software server side will send the obtained results after a forecasting operation.

**3.5. INTERNAL DATA REQUIREMENTS**

- 47. MNFIFS software will define a FNN by the following data structure

| Architecture of a FNN                       |                   |                                   |
|---|-------------------|-----------------------------------|
| Item Name                                   | Data Type         | Comment                           |
| Name of The FNN                             | String            |                                   |
| Number Of Inputs                            | Java Integer      | Defined as “n”                    |
| Number Of Hidden Layers                     | Java Integer      |                                   |
| Number Of Hidden Layers                     | Java Integer      |                                   |
| Number Of neurons in Hidden Layer-1         | Java Integer      |                                   |
| ...   | Java Integer      |                                   |
| Number Of neurons in Hidden Layer “n”       | Java Integer      |                                   |
| Activation Function Code For Input Layer    | Enumerated Number | 0:Sigmoid<br>1:Tansig<br>2:Linear |
| Activation Function Code For Output Layer   |                   |                                   |
| Activation Function Code For Hidden Layer 1 |                   |                                   |
| ..  |                   |                                   |
| Activation Function Code For                |                   |                                   |

|                  |  |  |
|------------------|--|--|
| Hidden Layer “n” |  |  |
| End Of File      |  |  |

**3.6. EXTERNAL DATA REQUIREMENTS**

- 48. MNFIFS software will accept training and validation data set in the following format.

| Data Set File Header    |                       |                   |
|-------------------------|-----------------------|-------------------|
| <i>Name of the Item</i> | <i>Data Type</i>      | <i>Comment</i>    |
| Number of inputs        | Java Integer          |                   |
| Input 1 Type            | Enumerated number     | 0:character       |
| ...                     |                       | 1:decimal number  |
| Input n Type            |                       | 2:floating number |
| Output Type             |                       |                   |
| End of Header           |                       |                   |
| <i>Name of the Item</i> | <i>Data Type</i>      | <i>Comment</i>    |
| Input 1 Value           | As defined in header. |                   |
| Input 2 Value           |                       |                   |
| .                       |                       |                   |
| Input n Value           |                       |                   |
| Output Value            |                       |                   |
|                         |                       |                   |

**3.7. SECURITY REQUIREMENTS**

- 49. MNFIFS software will have two separately executable file, one of them will be responsible for user interface and the other will be responsible for algorithms.
- 50. MNFIFS software algorithm part will be secure in the manner of source code accessibility.



### **3.8. COMPUTER RESOURCES REQUIREMENTS**

#### **3.8.1. Computer Resources Usage Requirements**

51. MNFIFS software will work on any platform
52. MNFIFS software will require the latest java runtime environment and virtual machine be installed on the target system.

#### **3.8.2. Communication Requirements**

53. MNFIFS software will communicate by using TCP/IP protocol.
54. MNFIFS software will need TCP/IP connection.

### **3.9. LOGISTIC REQUIREMENTS**

55. MNFIFS software will have an installation program for user interface part.
56. MNFIFS software installation program will be downloaded over internet.

### **3.10. OTHER REQUIREMENTS**

#### **3.11. PRIORITIES OF REQUIREMENTS**

Every requirement has the same priority.

## **4. OBSERVABILITY AND EVALUATION TECHNIQUES**

Every unit of the software will be tested by the designer in the manner of unit testing. After integration of each tested unit the whole system will be tested by testers. In this phase a black box testing method will be used. In this black box test every requirement will be observed and tested.

## **5. ABBREVIATIONS**

|        |   |
|--------|---|
| MNFIFS | Metu NeuroFuzzy Ionospheric Forecasting System Software |
| FNN    | Fuzzy Neural Network                                    |
| TEC    | Total Electron Content                                  |
| foF2   | Critical Frequency of F2 Layer                          |

## 6. ATTACHMENTS

### 6.1. Database Structure

**Tables;**

- User List Table: This table will hold all the user with their Access rights

| User Name | Access Right | Password |
|-----------|--------------|----------|
| yozkok    | Admin        | *****    |
| esenalp   | Super User   | *****    |
| userx     | Operator     | *****    |

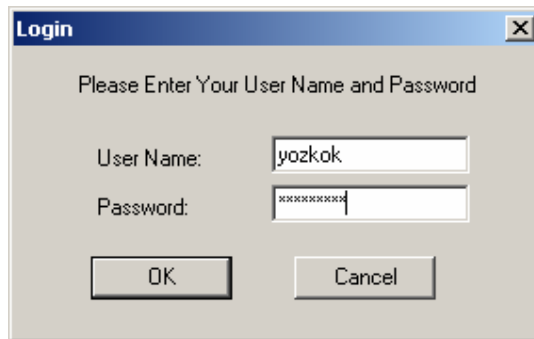
- Cooked Architectures Table: This table will hold NN architectures which have been trained and ready to operate. Each entry in this table will have an architecture name, the user name or ID who build this architecture.
- Active Architectures Table: This table will hold the NN architecture which is already in development phase. The developer name and architecture name will be hold in the table.
- Operation results tables: When architecture is signed as cooked by the developer of the architecture, a results table will be opened for this architecture. In this operation phase the selected architectures results table will be filled with the inputs, ID of the operator, date and time of the operation and the obtained result.

| Operator's user Name | Input 1 | Input 2 | . | . | . | Input n | Date Time | foF2 or TEC |
|----------------------|---------|---------|---|---|---|---------|-----------|-------------|
|                      |         |         |   |   |   |         |           |             |
|                      |         |         |   |   |   |         |           |             |

## 6.2. User Interface

In this part user interface of MNFIFS software will be described. In the descriptions user interface windows will be pictured, but these windows may change later.

**Login Interface:** Every user first of all has to login to the system to use the software. Every user name has to be defined on the database.



**Main Menu:** After entering a valid password user will drop into the main menu. This menu will dynamically be organized related with the user type which has dropped into. For an administrator every item in this window will be accessible. User will be routed into other menus related with the choice done in this menu. There will be four selections in this menu. These are listed as follows;

- Build a new FNN
- Continue with a previous FNN
- Train a FNN
- Forecasting (Operate a FNN)

By the selections in this menu user will be faced with the following menus.

**Build a new FNN :** When the user selects this item in the main menu, he will drop into this menu. This item will be accessible for only super users and administrators. In this menu user can build a new FNN and can save this network into the database with a name. Later these FNNs can be called for training and operation if trained. The user interface properties for this menu are listed below.

- Enter Number of Inputs
- Enter Number of Hidden Layers
- Enter Number of neurons in each Hidden Layer.
- Define Output Type (TEC, foF2)

- Select Activation Function For each layer.
- Save the architecture with a unique name.

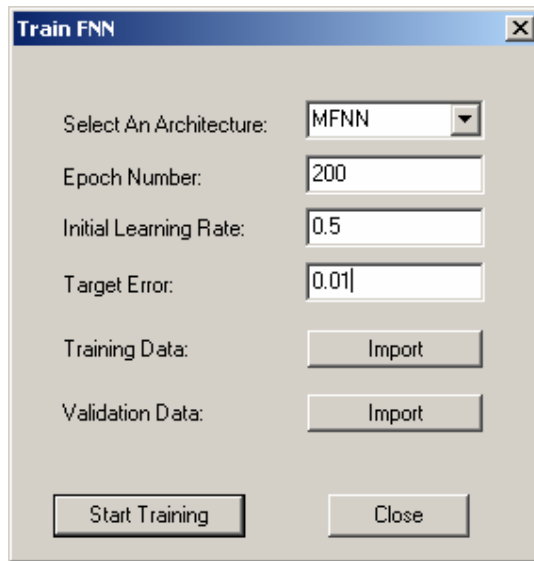
The screenshot shows a dialog box titled "Build New FNN". It contains the following fields and controls:

- Name of the Architecture: MFNN
- Number Of Inputs: 11
- Number Of Hidden Layers: 1
- Select Output Type: foF2
- Hidden Layer Properties:
  - Select Layer: Layer-1
  - Select Activation Function: Bipolar Sigmoid
  - Select Number of Neurons: 7

Buttons: Save (inside Hidden Layer Properties), Save, Cancel.

**Train a FNN:** This item will be accessible for only super users and administrators. In this menu the user will select one of the previously built FNNs from a list. After the FNN selection the user will import an appropriate training and validation data files. Then training parameters such as epoch number, desired error, and initial learning rate will be determined by the user. Basic elements of this menu are listed below;

- Select an Architecture
- Import Training Data File
- Import Validation Data File
- Set Initial Learning rate
- Set Epoch Number
- Set Desired Error



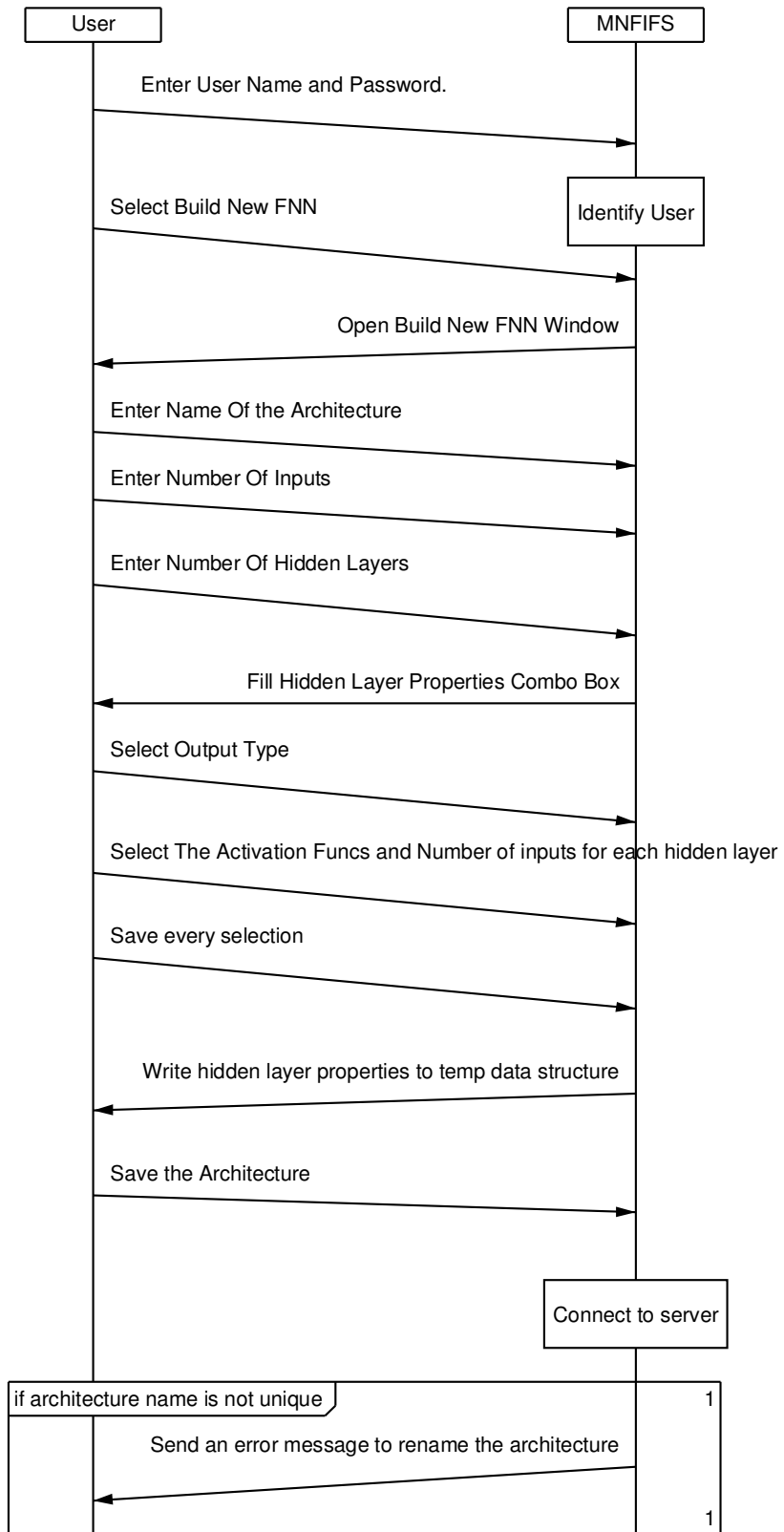
**Forecasting (Operation) Menu:** There will not be any restriction for the accessibility of this menu. Every user may use this property of the software. But there should be at least one trained FNN architecture in the system. In this menu the user first of all will select one of the previously trained FNNs. Then the input types for this architecture will be displayed. Then the user will fill up all these inputs with appropriate values. And finally the operation will be started. After the completion of the operation the obtained results will be displayed and also this result will be stored into the database with the inputs for this result and by relating with the name of the FNN.

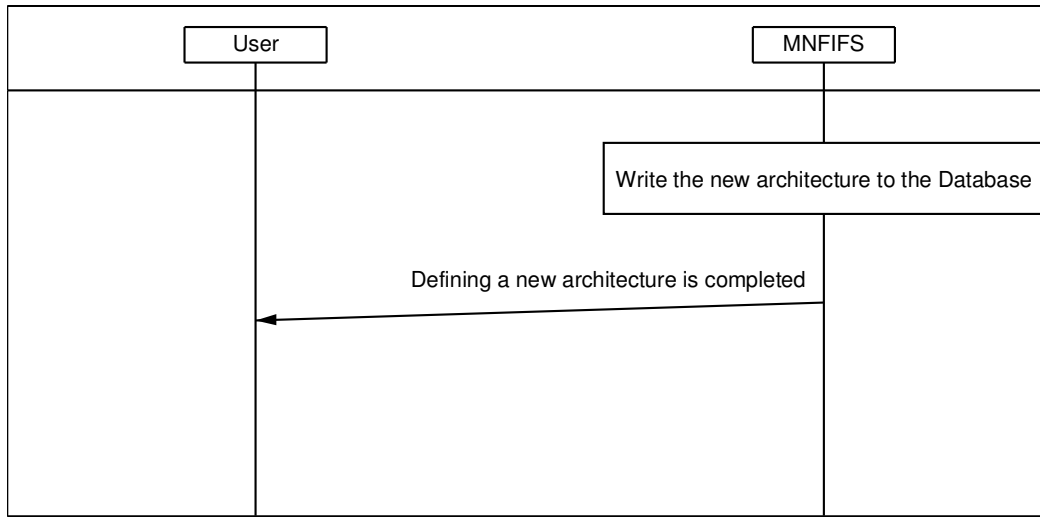
- Select a previously trained FNN
- Enter all input values
- Start Operation

### 6.3. Use Cases Message Sequence Charts

#### Building a new FNN:

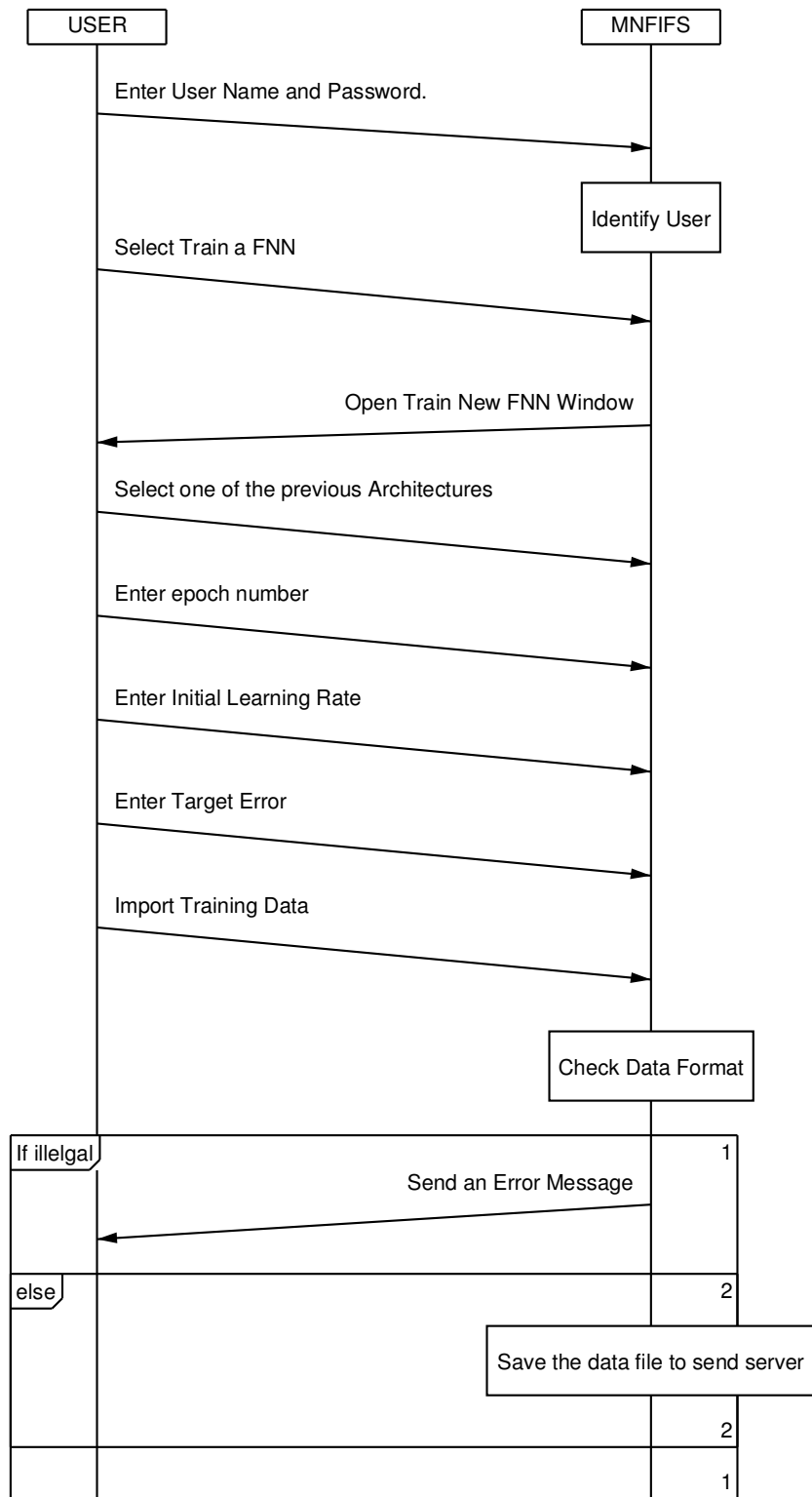
MSC MNFIFSBUILD



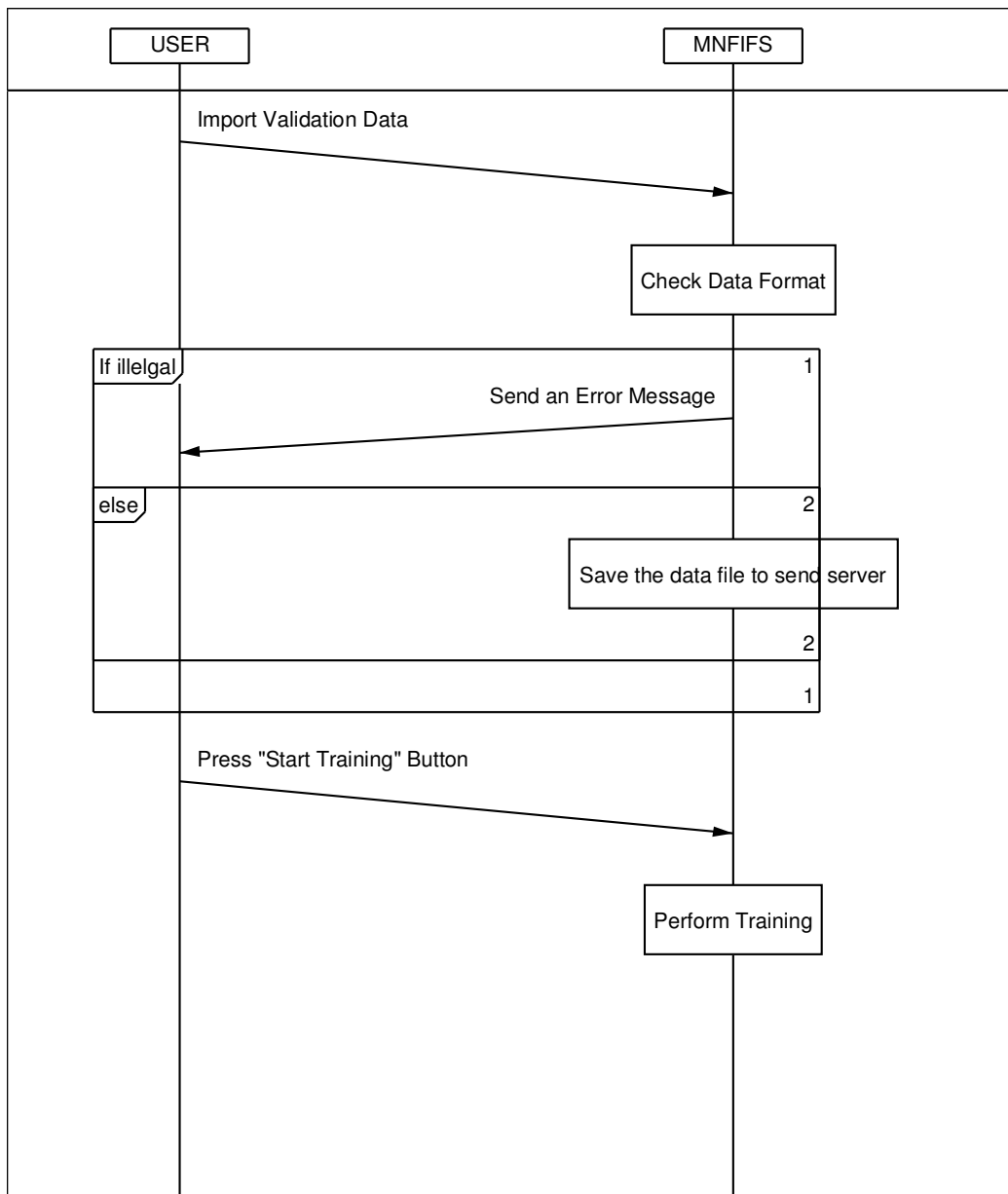


**Train A FNN:**

MSC TrainFNN

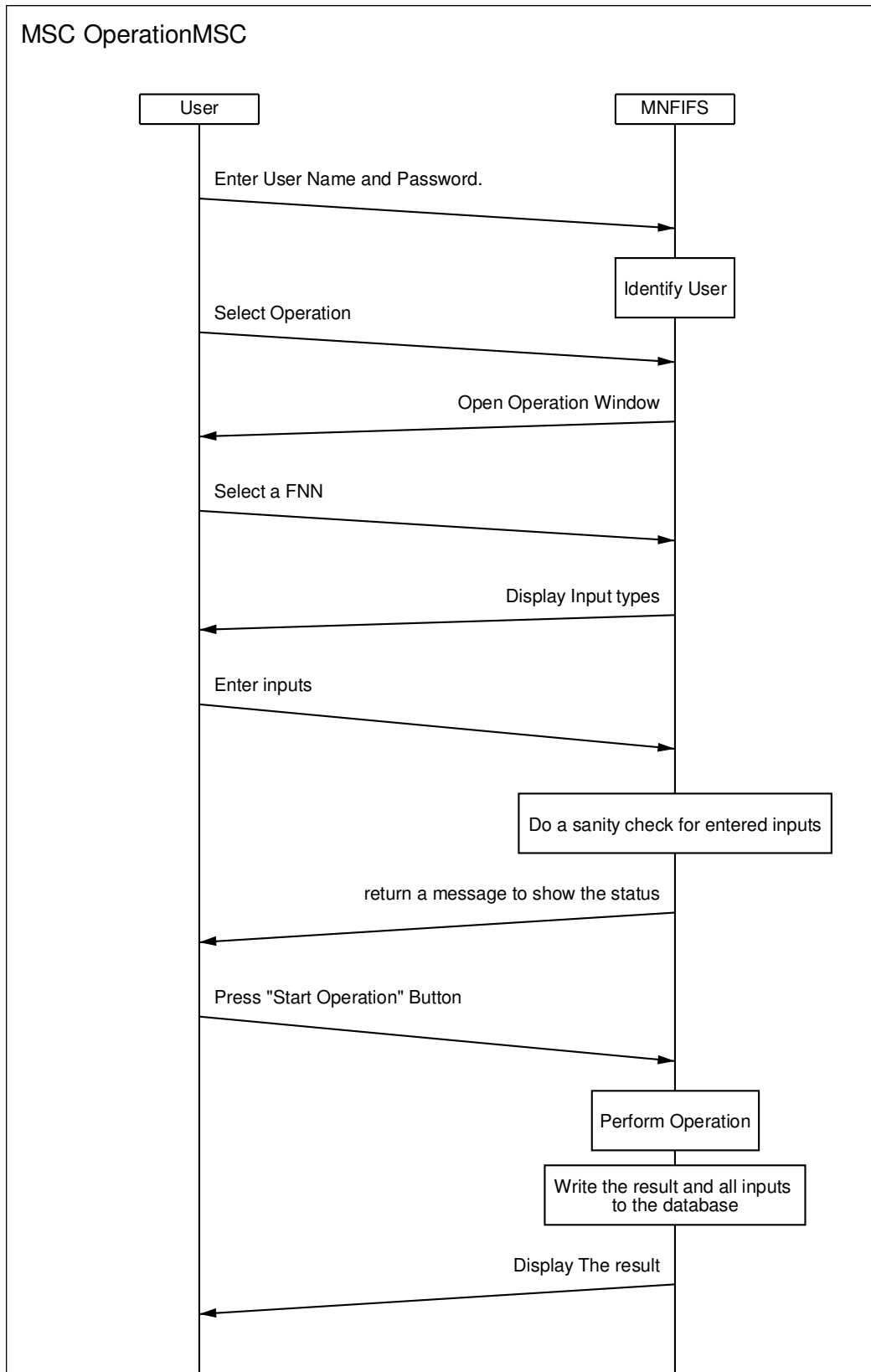






**Operate a FNN:**

### MSC OperationMSC



## APPENDIX B

### DURATION AND EFFORT ESTIMATION OF METU-IFS

This document presents the estimation of duration and effort for the METU-IFS software. As a first step the lines of code should be estimated. For this purpose, all Function Point Analysis (FPA) will be done firstly. Then the Unadjusted Function Point Analysis (UFPA) will be done in the second part. At the third part Lines Of Code (LOC) will be estimated by using the results of the previous analysis. And finally at the fourth part, effort and duration will be calculated according to the basic COCOMO model.

#### 1. Function Point Analysis

##### 1.1. Total Unadjusted Function Point Calculation

##### 1.1.1. Unadjusted Function Point Calculation for Server

###### *1.1.1.1 External Inputs Of Server*

| No | Input Name                   | From   | Weighting |
|----|------------------------------|--------|-----------|
| 1  | Neural Net Structure         | Client | Avarage   |
| 2  | Training and Validation Data | Client | Simple    |
| 3  | NN inputs for forecasting    | Client | Simple    |
| 4  | User Login Inputs            | Client | Simple    |

###### *1.1.1.2 External Outputs of Server*

| No | Output Name            | To     | Weighting |
|----|------------------------|--------|-----------|
| 1  | Previous NN Structures | Client | Avarage   |

|   |                               |        |        |
|---|-------------------------------|--------|--------|
| 2 | Types of Activation functions | Client | Simple |
| 3 | Operation Result              | Client | Simple |
| 4 | Connection Type               | Client | Simple |

#### *1.1.1.3 File Storage*

| No | File Info                          | From   | Weighting |
|----|------------------------------------|--------|-----------|
| 2  | NN Structures                      | Client | Simple    |
| 4  | Training and Validation Data files | Client | Simple    |

#### *1.1.1.4 External SW Interfaces*

| No | Interface Name  | From    | Weighting |
|----|---|---------|-----------|
| 1  | TCP/IP Data transfer interface (Between clients and server) | Clients | Simple    |

#### *1.1.1.5 External Query Types*

| No | Query Type        | Weighting |
|----|-------------------|-----------|
| 1  | User Types        | Simple    |
| 2  | NN Structures     | Simple    |
| 3  | Operation Results | Simple    |

| Measurement Parameter  | Weighting Factor |         |         | Total     | Weighting |    |    |
|------------------------|------------------|---------|---------|-----------|-----------|----|----|
|                        | Simple           | Average | Complex |           | S         | A  | C  |
| External Inputs        | 3                | 1       | 0       | 13        | 3         | 4  | 6  |
| External Outputs       | 3                | 1       | 0       | 17        | 4         | 5  | 7  |
| File Storage           | 2                | 0       | 0       | 14        | 7         | 10 | 15 |
| External SW Interfaces | 1                | 0       | 0       | 5         | 5         | 7  | 10 |
| External Inquiries     | 3                | 0       | 0       | 9         | 3         | 4  | 6  |
| <b>Count Total</b>     |                  |         |         | <b>58</b> |           |    |    |

Table-1 Unadjusted function points for server

Total unadjusted function points for server, came out to be 58.

### 1.1.2. Unadjusted Function Point Calculation for Client

#### 1.1.2.1. External Inputs Of Client

| No | Input Name                    | From | Weighting |
|----|-------------------------------|------|-----------|
| 1  | Login Input                   | User | Simple    |
| 2  | Main Menu                     | User | Simple    |
| 3  | Building a new NN Structure   | User | Simple    |
| 4  | Training inputs               | User | Simple    |
| 5  | Inputs for Forecasting window | User | Simple    |

#### 1.1.2.2. External Outputs

| No | Output Name                    | To   | Weighting |
|----|--------------------------------|------|-----------|
| 2  | Activation functions combo box | User | Simple    |
| 3  | Previous NNs                   | User | Simple    |
| 4  | Operation Results              | User | Simple    |
| 5  | Trained neural nets combo box  | User | Simple    |

#### 1.1.2.3. File Storage

| No | File Info                         | From | Weighting |
|----|-----------------------------------|------|-----------|
| 1  | New NNs structure                 | User | Simple    |
| 2  | NN inputs for operation           | User | Simple    |
| 3  | Training and validation file sets | User | Simple    |

**1.1.2.4. External SW Interfaces**

| No | Interface Name  | With   | Weighting |
|----|---|--------|-----------|
| 1  | TCP/IP Data transfer interface (Between clients and server) | Server | Simple    |
| 2  | User Interface  | User   | Simple    |

**1.1.2.5. External Query Types**

No query

| Measurement Parameter  | Weighting Factor |         |         | Total     | Weighting |    |    |
|------------------------|------------------|---------|---------|-----------|-----------|----|----|
|                        | Simple           | Average | Complex |           | S         | A  | C  |
| External Inputs        | 5                | 0       | 0       | 15        | 3         | 4  | 6  |
| External Outputs       | 4                | 0       | 0       | 16        | 4         | 5  | 7  |
| File Storage           | 3                | 0       | 0       | 21        | 7         | 10 | 15 |
| External SW Interfaces | 2                | 0       | 0       | 10        | 5         | 7  | 10 |
| External Inquiries     | 0                | 0       | 0       | 0         | 3         | 4  | 6  |
| <b>Count Total</b>     |                  |         |         | <b>62</b> |           |    |    |

Table-2 Unadjusted function points for client

Total unadjusted function points for client came out to be 62.

**2. Adjusted Function Point Calculation**

The function point count is adjusted for the complexity of the software by assessing each of the answers to the following questions on a scale of 0 to 5.

|              | <b>F<sub>i</sub> value</b> |
|--------------|----------------------------|
| None         | 0                          |
| Slightly     | 1                          |
| Present      | 2                          |
| Medium level | 3                          |
| Important    | 4                          |
| Vital        | 5                          |

Table-3 Questions and values

|                                   |              |          |
|-----------------------------------|--------------|----------|
| 1. Data Communications            | Present      | <b>2</b> |
| 2. Distributed Processing         | Slightly     | <b>1</b> |
| 3. Performance Requirements       | Present      | <b>2</b> |
| 4. Operational Configuration Load | Slightly     | <b>1</b> |
| 5. Transaction Rate               | None         | <b>0</b> |
| 6. Online Data Input              | Present      | <b>2</b> |
| 7. End User Quality               | Medium level | <b>3</b> |
| 8. Online File Update             | Present      | <b>2</b> |
| 9. Algorithmic Complexity         | Present      | <b>2</b> |
| 10. Reusability                   | Slightly     | <b>1</b> |
| 11. Ease of Installation          | Slightly     | <b>1</b> |
| 12. Operational Ease              | Present      | <b>2</b> |
| 13. Multi-site System             | None         | <b>0</b> |
| 14. Maintainability               | Slightly     | <b>1</b> |

Table 4 Questions and answers for adjusted function point calculation

These 14 complexity adjustment values are summed to give the value of  $\Sigma (F_i)$ .

$$\Sigma (F_i) = 20$$

$$\text{Adjusted Function points} = \text{Count Total} * (0.65 + 0.01 * \Sigma (F_i))$$

$$=(58+62)*0.67 = \mathbf{80,4}$$

### 3. Lines of Code Estimation

The total adjusted function points came out to be **80,4** for the MNFIFS application(including both server and client). For estimating lines of code the Table 5 below can be used.

| Language | LOC/FP (Low & Jeffery 1990) |
|----------|-----------------------------|
| C        | 150                         |
| C++      | 80                          |
| Java     | 70                          |
| C#       | 75                          |

Table-5 Estimated LOC for each FP.

| Language    | LOC/FP (Low & Jeffery 1990) | LOC for WB |
|-------------|-----------------------------|------------|
| C           | 150                         | 12060      |
| C++         | 80                          | 6432       |
| Java        | 70                          | 5628       |
| C#          | 75                          | 6030       |
| Adjusted FP | 80,4                        |            |

Table-6 LOC calculation.

For Java Software language, the results are came up to be **5628 LOC** according to Low & Jeffery 1990.

### 4. Effort and Duration Estimation

In this part effort and duration will be estimated according to Basic COCOMO model by using the lines of code which has been estimated at the previous part.

For the Basic COCOMO model effort and duration are calculated as the following way;

$$E = a_b(KLOC)^b_b \quad E: \text{Effort (person-month)}$$

$$D = c_b(E)^d_b \quad D: \text{Duration (months)}$$

KLOC: estimated thousand lines of codes



The constants are as following;

|               | a <sub>b</sub> | b <sub>b</sub> | c <sub>b</sub> | d <sub>b</sub> |
|---------------|----------------|----------------|----------------|----------------|
| Organic       | 2.4            | 1.05           | 2.5            | 0.38           |
| Semi-detached | 3.0            | 1.12           | 2.5            | 0.35           |
| Embedded      | 3.6            | 1.2            | 2.5            | 0.32           |

For MNFIFS Software effort and duration will be calculated by using the constants for Organic type of software. Because the software will be designed and implemented in an object oriented environment.

So the Effort and Duration comes out as following;

$$E = 2.4 * (5,628)^{1.05} = 14,7 \text{ (person-month)}$$

$$D = 2.5 * (14,7)^{0.38} = 6,94 \text{ months}$$

$$\text{Average staffing} = 14.7(\text{person - months}) / 6.94(\text{months}) = 2.11 \sim 2$$

This means that with 2 full time person each month the duration will be about 7 months.