

جامعة فيلادلفيا  
نموذج تفويض

أنا "محمد خالد" يوسف محمد شمبور، أفوض جامعة فيلادلفيا بتزويد نسخ من رسالتي للمكتبات أو المؤسسات أو الهيآت أو الأشخاص عند طلبها.

التوقيع:

التاريخ:

Philadelphia University  
Authorization Form

I, Moh'd-Khaled Yousef Mohammed Shambour, authorize Philadelphia University to supply copies of my thesis to libraries or establishment or individuals upon request.

Signature:

Date:



**Prediction of  
the Dead Sea Water Level Using  
Neural Networks**

**"Mohammed Khaled" Yousef Shambour**

200520658

Supervised by:

Dr. Rashid Al-Zubaidy

This Thesis was Submitted in Partial Fulfillment of the Requirements for the  
Master's Degree In Computer Science

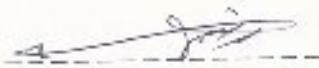



Deanship of Academic Research and Graduate Studies  
Philadelphia University

April 2008

## Philadelphia University

April, 2008

Successfully defended and approved on 17-04-2008

Examination Committee	Signature
Dr. <u>Rashid Al-Zubaidy</u> , Chairman. Academic Rank: <u>Associate Prof</u>	
Dr. <u>Mourad Masouche</u> , Member. Academic Rank: <u>Associate Professor</u>	
Dr. <u>Moayad A. Kadhit</u> , Member. Academic Rank: <u>Associated Professor</u>	
Dr. <u>Khalil M. El Hinali</u> Dr. <u>K. Hindi</u> , External Member. Academic Rank: <u>Associa<sup>n</sup> Prof.</u>	

## **Acknowledgment**

I would like to thank my supervisor Dr. Rashid Al-Zubaidy for his insight suggestions in both the content and presentation of this thesis.

I'm also very grateful and thankful to my brother Qusai who took effort in reading my thesis chapter by chapter and provided me with many valuable comments.

I wish to express my love and gratitude to my parents for there unconditional love, encouragement, support and patience throughout my studies.

I would like to thank Philadelphia University administration for their support and help in the past two years. Specially, President of Philadelphia University, Dean of Scientific Research, Dean of IT Faculty, and Chief of Computer Science Department.

I would like to thank every professor who taught me in the past two years. I would like to thank Dr.Said Alghoul and Dr.Vinous Samawi for their help.

## Table of Contents

Acknowledgements .....	IV
Table of Contents .....	V
List of Figures .....	VII
List of Tables.....	X
List of Abbreviations .....	XI
Abstract .....	XII

### CHAPTER 1: INTRODUCTION

1.1 Overview .....	1
1.2 Study Objectives .....	3
1.3 Research Approach and Scope of Work .....	3
1.4 Significance and Contribution.....	4
1.5 Thesis Outline .....	5

### CHAPTER 2: LITERATURE REVIEW

2.1 General .....	7
2.2 Related Work.....	7
2.2.1 Dead Sea Literature .....	7
2.2.2 Neural Network Literature .....	9

### CHAPTER 3: ARTIFICIAL NEURAL NETWORKS

3.1 Introduction.....	12
3.2. Basic Structure .....	13
3.3 Training. ....	15
3.4 Architecture of Neural networks.....	15
3.4.1 Feed-Forward Networks .....	15
3.4.2 Feedback Network.....	16
3.5 Research Methodology .....	16
3.5.1 Multilayer Perceptron Neural Networks (MLP-NN).....	17
3.5.1.1 Architecture of MLP.....	19

3.5.1.2 Training NN using BackPropagation Algorithms.....	20
3.5.2 Radial Basis Function (RBF) Model .....	25
3.5.2.1 Architecture of RBF .....	25
3.5.2.2 Training RBF Networks .....	26
3.5.3 Selection of Networks Structures.....	29
 <b>CHAPTER 4: DESIGN AND IMPLEMENTATION</b>	
4.1 The Study process.....	30
4.2 Data Set.....	31
4.2.1 Data Pre-Processing .....	31
4.2.2 Data Post-Processing.....	31
4.3 Dead Sea Variables selection.....	31
4.4 NN Parameters .....	36
4.4.1 Learning threshold .....	36
4.4.2 Learning rate .....	37
4.4.3 Momentum .....	37
4.4.4 Number of hidden nodes .....	37
4.5 Performance measurement.....	37
4.5.1 Mean Squared Error (MSE) .....	38
4.6 Design the Neural Networks .....	38
4.7 Tools and Platforms.....	39
 <b>CHAPTER 5: EXPERIMENTAL RESULTS</b>	
5.1 Results .....	40
 <b>CHAPTER 6: CONCLUSIONS AND FUTURE WORK</b>	
6.1 Conclusions .....	57
6.2 Recommendations for Future work .....	58
 <b>REFERENCES</b> .....	 59

## List of Figures

<b>FIGURE No.</b>	<b>FIGURE TITLE</b>	<b>PAGE No.</b>
Figure 1.1	Dead Sea Basin	6
Figure 3.1	Neuron Unit	13
Figure 3.2	A three-layer Neural Networks	14
Figure 3.3	An example of a simple feed forward Networks	16
Figure 3.4	Recurrent Neural Networks	16
Figure 3.5	Structure of a DS surface water level model with one hidden layer	18
Figure 3.6	A three-layer Neural Networks	19
Figure 3.7	Sigmoid function	20
Figure 4.1	Study process	30
Figure 4.1a	Average Salinity and Temperature of DS under 100 meter below sea level	33
Figure 4.1b	Surface level of DS (1992-2006)	33
Figure 4.2	Inflow of water	35
Figure 4.3	GRABIT program	35

## VIII

<b>FIGURE No.</b>	<b>FIGURE TITLE</b>	<b>PAGE No.</b>
Figure 5.1	The performance of GRNN when spread value equal 0.1 for set 4	41
Figure 5.2	The performance of GRNN when spread value equal 0.1 for set 1	42
Figure 5.3	The performance of L-M for set 4	43
Figure 5.4	The performance of BP for set 4	44
Figure 5.5	The performance of BP for set 6	45
Figure 5.6	GRNN training with spread equal to 0.2 for set 1	47
Figure 5.7	GRNN training with spread equal to 0.2 for set 4	47
Figure 5.8	GRNN training with spread equal to 1 for set 1	48
Figure 5.9	GRNN training with spread equal to 1 for set 4	48
Figure 5.10	GRNN training with spread equal to 100 for set 1	49
Figure 5.11	GRNN training with spread equal to 100 for set 4	49
Figure 5.12	GRNN test with spread equal to 0.2 for set 1	50
Figure 5.13	GRNN test with spread equal to 0.2 for set 4	50
Figure 5.14	GRNN test with spread equal to 100 for set 1	51
Figure 5.15	BP test set for the set number 1	53



<b>FIGURE No.</b>	<b>FIGURE TITLE</b>	<b>PAGE No.</b>
Figure 5.16	BP test set for the set number 4	54
Figure 5.17	BP test set for the set number 6	54
Figure 5.18	LM test set for the set number 1	55
Figure 5.19	LM test set for the set number 4	55
Figure 5.20	LM test set for the set number 6	56

## List of Tables

<b>TABLE No.</b>	<b>TABLE TITLE</b>	<b>PAGE No.</b>
Table 4.1	Humidity values (1988-2005)	34
Table 4.2	Factors where effected the DS water level	36
Table 5.1	GRNN Training & Testing values for Spread equal 0.1	41
Table 5.2	L-M Training & Testing values for number of neurons is11	42
Table 5.3	BP Training & Testing values for number of neurons is 5	44
Table 5.4	GRNN Training & Testing values for Spread equal to 0.2	45
Table 5.5	GRNN Training & Testing values for Spread equal to 0.5	46
Table 5.6	GRNN Training & Testing values for Spread equal to 1	46
Table 5.7	GRNN Training & Testing values for Spread equal to 100	46
Table 5.8	L-M Training & Testing values with different number of neurons and 0.0005 goal	52
Table 5.9	Training & Testing values with 6 neurons and 0.0005 goal	52
Table 5.10	BP Training & Testing values with 5 neurons and 0.0005 goal	52
Table 5.11	BP Training & Testing values with 4 neurons and 0.0005 goal.	52

## List of Abbreviations

No.	ABBREVIATION	STANDS FOR
1	ANN	Artificial Neural Network
2	BP	BackPropagation
3	DS	Dead Sea
4	FFNN	Feed Forward Neural Network
5	FL	Fuzzy logic
6	GA	Genetic Algorithm
7	GRNN	General Regression Neural Network
8	LM	Levenberg- Marquardt
9	MLP	Multilayer Perceptron
10	RBF	Radial Basis Function
11	RS	Read Sea
12	SNN	Simulated Neural Networks
13	PDPs	Parallel Distributed Processing System

## ABSTRACT

The Dead Sea (DS) basin plays a major role for regional economic development (industry, tourism and agriculture) in Jordan. Different studies stated that the water level of the DS is dropping an average of 3 feet per year. Accordingly there is a need to provide accurate and reliable estimates for the water level to help the researchers and geologists of the DS to make different kind of studies giving results, so they can understand the state of the DS and its behavior and stop the dropping of the DS water level.

Neural Networks (NN) are computational models with the capacity to learn, to generalize, or to organize data based on parallel processing. Among all kinds of networks, the most widely used are BackPropagation (BP), Levenberg-Marquardt (L-M), and Generalized Regression Neural Networks (GRNN) that are capable of representing non-linear functional mappings between inputs and outputs.

Different NN based DS water level prediction models are built and compared to determine the most effective neural networks work in prediction. It is known that DS water level depends on many factors such as Air temperature, Salinity, Humidity and other environmental information. Our NN models capture different subsets of those effects, reflect them within our models to identify the most effective set, which has significant impact on the water level of DS.

Finally, we can say that the proposed GRNN model provides best significant performance results comparing with other NN models using Mean Square Error (MSE).

# CHAPTER 1

## INTRODUCTION

### 1.1 Overview

The Dead Sea (DS) basin plays a major role for regional economic development (industry, tourism and agriculture). In the last 20 years, the Dead Sea has receded more than 60 feet on its western side, and in the last 40 years, it is believed that the water level has dropped more than 262 feet. By the way, the water level of the DS is dropping an average of 3 feet per year (Abu-Jaber 2004, Salameh and El-Naser 1999, Gertman and Hecht 2002, Asmar and Ergenzinger 1999, Oroud 1994, Hassan and Klein 2002, Yechieli et al, 1998 ).

Water from the natural inflows (the Jordan River) has been blocked and diverted for urban and agricultural uses inside and outside the watershed. In addition, much water is pumped from the DS into evaporation ponds, which alone constitutes about 25 % of the present total evaporation rates, which means the amount of water allowed to flow southward into DS is dwindling constantly (Abu-Jaber 2004, Salameh and El-Naser 1999, Orthofer and Lipchin 2004, Gertman and Hecht 2002, Lipchin 2006).

The DS is dying, that is the conclusion many environmentalists are reaching as water level continues to fall precipitously and at appalling rates. It is really a catastrophe unfolding before our eyes, and there seems to be little that can be done to repair the situation.

Today, the shoreline on the western side is more than a kilometer. And a view from the western slopes overlooking the DS reveals parallel geological strips suggesting that once upon a time, water covered the flat terrain. The receding water is also leaving behind hundreds of mysterious and dangerous pits and holes, some filled with water, others not (Oroud 1995).

However, environmentalists warn that pumping far less salty water into the DS could release toxic gases and wipe out local plant and wildlife. The damage to wildlife would specifically impact the millions of migratory birds for which the DS is a seasonal resting place. Also, some experts have come with recommendations to stem the tide of the DS shrinkage, including allowing a greater amount of water from the north to flow into the Sea, and other recommends to build a tunnel between Read Sea (RS) and DS (Mohsen 2007, Asmar and Ergenzinger 2003).

Applications of neural networks are applied to many different fields such as science, engineering, automotive, banking, medical, business, transportation, defense, industrial, telecommunications, insurance, and economic. In the last few years, the subject of NN or neural computing has generated a lot of interest and receives a lot of coverage in articles and magazine.

NN methods are still gaining popularity, as is evidenced by the increasing number of papers on this topic appearing in engineering and hydrology journals, conferences, seminars, and so on. This modeling tool is still in its growing stage in terms of hydrologic applications (ASCE 2000). Recently there are increasing number of works attempt to apply the NN method for solving various problems in different branches of science and engineering. This highly interconnected multiprocessor architecture in NN is described as parallel distributed processing and has solved many difficult computer science problems, and ability to handle a complex systems that may be poorly defined or understood using mathematical equations.

The innovation of the NN technique has added a new dimension to model such systems and has been applied in recent years, as a successful tool to solve various problems concerned with hydrology and water resources engineering (ASCE 2000, Keskin and Terzi 2006).

The main function of all artificial neural network paradigms is to map a set of inputs to a set of output. However, there are a wide variety of NN algorithms. An attractive feature of NN is their ability to extract the relation between the inputs and outputs of a process, without the physics being explicitly provided to them. They are able to provide a mapping from one multivariate space to another, given a set of data representing that

mapping, even if the data is noisy and contaminated with errors. Therefore, the natural behavior of geological system processes is appropriate for the application of NN methods.

The networks were trained and tested using data that represent different characteristics of the DS area. In this thesis, BackPropagation (BP) model where built in Java and implemented in Weka data mining software (Weka 2007), Levenberg-Marquardt (L-M), and General Regression Neural Network (GRNN) were built within Matlab environment are applied in this research.

It is known that DS water level depends on many factors such as salinity, air speed, humidity and other environment information. Our NN models will firstly capture subset of those effects, reflect them within the system, and provide which is the subset from a given set have strongly relation with water level of DS. Secondly, we will identify the most accurate and reliable NN model among other used models for the future prediction of the DS water level.

## **1.2 Study Objectives**

The overall objective of the present study is to determine the effective factors related to water level of DS by developing a model that are able to provide accurate and reliable estimates from the historical data, The modeling techniques used in this study are based on Neural Networks (NN) algorithms that will model relationships between input and output. These models are developed to provide and to identify the best subset of parameters that have the most impact on the DS water level, also a comparison will be performed between different NN models to select the most accurate and reliable model for the future prediction of the DS water level. on the other hand helps the researchers and geologists of the DS to make different kind of studies giving results which are help to understand the state of the DS and its behavior since there are different studies of RS-DS tunnel project are discussed currently.

## **1.3 Research Approach and Scope of Work**

The present study was undertaken to develop weekly sea water level model using the NN method that can possibly be used to provide reliable and accurate estimates based

on the inputs variables (main factors). It is believed that the NN is able to overcome the non-linear relationship between the main factors that the DS affected by and the water level of sea.

The modeling work was carried out using eight years period of weekly data (Jun-1992 To Jun-2000) this period taken according to data available, cause there is not sufficient continues data available. the main factors records selected from different sites and scientific journals inside and outside of Jordan. Those are the (metrological department of Hashemite Kingdom of Jordan), (The ISRAMAR organization for Oceanographic and limnological Research) and from research papers for (Abu-Jaber 2004, Salameh and El-Naser1999, Gertman and Hecht, 2002 ) and others.

#### **1.4 Significance and Contribution**

There are many different studies concerning DS environment, such as, water level, physical feature, water balance, energy balance, water management, climate, chemical and hydrological evolution, evaporation rate, and so on. Each study take its meteorological variables according to philosophy of researcher, each one have its own view for his research and there is no full matching between them about which meteorological parameters must be used as effective variables for studying the DS water level. So this research will take all available variables and extract subset of variables that are most related and effected to the DS water level.

This is achieved by deploying the NN which has strong generalization ability. This means that once NN has been properly trained, they are able to provide accurate results even for cases they have never seen before.

In this work, we clearly defined the importance of the relationship between input-output variables, as will as identify a predicting model that can be used to provide reliable and accurate estimation.

Comparison between BP, L-M, and GRNN was made giving the best NN algorithms that can be effectively used for future prediction of the water level in the DS.



## **1.5 Thesis Outline**

This thesis consists of five chapters. Chapter 2 provides a literature review for the features and state of DS and review of using various neural network algorithms. Chapter 3 provides a description of NN models. Design and Implementation is also discussed in Chapter 4. Chapter 5 concludes the results, and gives recommendations for future work.

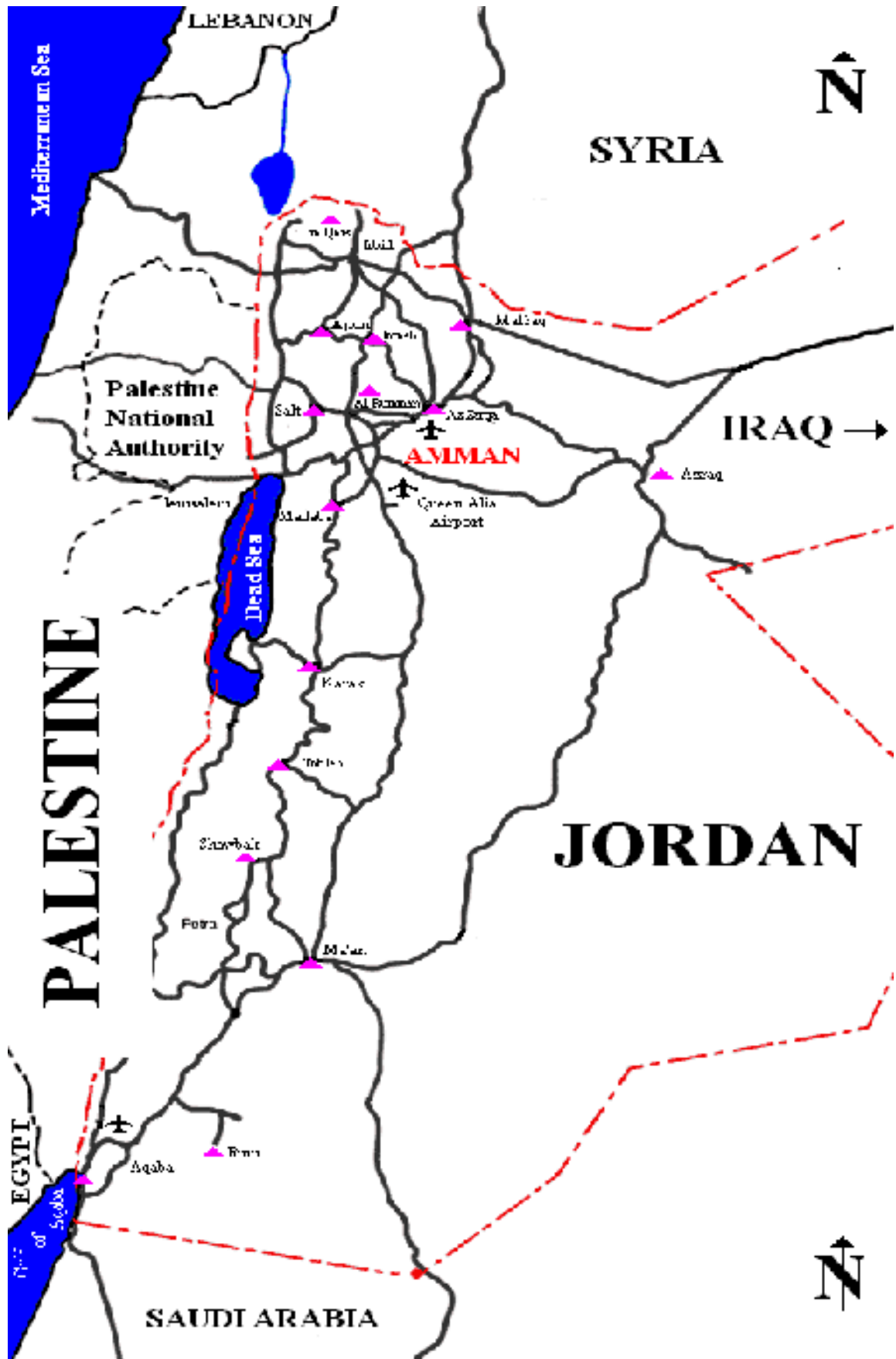


Figure (1.1): Dead Sea Basin

# CHAPTER 2

## LITERATURE REVIEW

### 2.1 General

The development of the computer and neural network techniques provides hydrologists and researchers with enhanced computational power to solve complicated problems. This power increased the possibilities of applying search algorithms, and the ability to simulate models of cognitive processes. This developments stimulated neural networks research. In particular, several models based on time series analysis methods or based on regression techniques have been commonly used to describe the random behaviour of various physical phenomena in hydrology. With the advent of computational and simulation capability using computers, new technique such as neural networks have been developed to represent more accurately complex non-linear stochastic processes Tokar and Markus (1999) and Engrg (2000).

### 2.2 Related Work

Determining the relationship between DS water level and effective factors for DS area is one of the most important problems faced by geologists and scientists. The historical data are little and discrete for may factors.

This study based on two major fields, the first one depends on the geological state and the main factors of Dead Sea, and the second one based on the methodology of Neural Network.

#### 2.2.1 Dead Sea Literature

There are different kinds of studies concerns DS environment, each study has its own characteristics of selection the main factors the DS affected by.

- Lensky et al, (2005) used dataset consists of meteorological and hydrographical measurements, which includes air temperature, relative humidity, incoming solar radiation, atmospheric pressure, surface water temperature and temperature profile down to 40 meter below the sea surface level, they used these variables as a basic parameters to estimate the water balance of the DS.

- Gertman et al, (2006) applied a standard set of meteorological parameters (wind speed and direction, relative humidity, air temperature, insolation and air pressure), as well as water temperature on 12 levels at upper 40 meters in monitoring the Dead Sea water level.
- Rudolf Orthofer et al, (2004) discussed the physical features that related to DS, which are salinity, rainfall, surface and underground water resources, evaporation and temperature.
- Al-Weshah (2000) determine in his research the water balance of the DS, and describe the main components related to the water balance analysis are: surface inflows from the Jordan river system, surface inflows from eastern wadis, surface inflows from western wadis, rainfall on the DS itself, evaporation from the DS surface itself, abstractions from potash mining and works on the east and west sides of the DS and estimated groundwater inflows.
- Salameh and El-Naser (1999) their research is considered to be one of the most effective studies of the water balance of the DS, it summarizes the water amounts which used to flow into the DS prior to the water resources development in its catchments area, which are surface water (lake Tiberius, east side wadis, west side wadis, DS eastern catchments, DS western catchments, wadi Araba basin eastern side and western side), groundwater (eastern side, western side, northern and southern basins) and precipitation.
- Calder and Neal (1984) developed a method to estimate evaporation rate from the DS based on a Penman formula. They derived at an equation that expresses evaporation as a function of net radiation, air temperature, humidity and wind speed, in addition to salinity, whose effect was included through changing the water activity coefficient. But their model overestimates net radiation, and ignores the dependence of some of the variables on temperature.
- Oroud (1995) used Penman formula approach to evaluate the evaporation from shallow pans near the southern tip of the DS.
- Mero and Simon (1985), in their model to simulate the DS, used a semi-empirical approach, and proposed an equation which depends on surface water density and temperature, in addition to salinity. Their approach accounted for radiation and cloud effects but completely ignored the effect of wind speed.

- Asmar and Ergenzinger (1999) modified a method derived from Penman model that estimates the evaporation as a function of four essential variables which are salinity, humidity, air temperature and wind speed.

### **2.2.2 Neural Network Literature**

Many articles report that neural networks method can produce good models of the problem that accurately represent nonlinearities in the data (Govindaraju 2000). Therefore, the natural behaviour of hydrological processes is appropriate for the application of neural networks method. The characteristics of non-linearity and existence of noise component in hydrological processes demand a solution that can promise a reliable result.

- Wanakule and Aly (2005) they apply ANNs as alternative models that are capable of providing accurate water level forecasts and used it for managing regional well fields in the north-central Tampa Bay area.
- Hsu et al, (1995) used a three layer feedforward ANN to model daily rainfall-runoff relationship. They concluded that the feedforward ANN needed a trial and error procedure to find the appropriate number of time delayed input variables to the model.
- De Vos and Rientjes (2005) used multi-layer feedforward ANNs for rainfallrunoff modeling of the Geer catchment (Belgium) using both daily and hourly data. The daily forecast results indicate that ANNs can be considered good alternatives for traditional rainfall-runoff modeling approaches.
- Gibbs et al, (2006) applied the two different ANN techniques (MLP and GRNN) for predicting chlorine concentration at two key location in the Hope Valley distribution system in South Australia. The inputs to the ANN model consist of the significant parameters produced the most accurate result.
- O. Makarynsky et al, (2004) used NNs to predict hourly sea level variations for 24 hours, for half-daily, daily, 5-daily and 10-daily mean sea levels at Hillarys Boat Harbour, Western Australia, for the period (December 1991–December 2002). Three-layer feed-forward networks were employed in their study, with a non-linear differentiable log-sigmoid transfer function in the hidden layer and linear transfer function in the output layer. The nets were trained with the resilient backpropagation algorithm in 200 training epochs. The results obtained that the feasibility of the neural

network in terms of the correlation coefficient is (0.7–0.9), root mean square error (about 10%).

- Raghuwanshi et al, (2006) developed an ANN models to predict both runoff and sediment yield on a daily and weekly basis, for a small agricultural watershed. The models was train using monsoon season (June to October) data of five years (1991–1995) for different sizes of architecture, and then tested with respective rainfall and temperature data of monsoon season (June to October) of two years (1996–1997). Training was conducted using the Levenberg–Marquardt where the input and output were presented to the neural network as a series of learning sets. They conclude in all cases, the ANN models performed better than the linear regression based models.
- Keskin and Terzi (2006) proposed an alternative approach of evaporation estimation for Lake Egirdir by developed ANN models to estimate daily pan evaporation from measured meteorological data. The measured meteorological variables include daily observations of air and water temperature, sunshine hours, solar radiation, air pressure, relative humidity, and wind speed. The results of the Penman method and ANN models are compared to pan evaporation values. Their study shows that there is better agreement between the ANN estimations and measurements of daily pan evaporation than for other model.
- Engrg (2003) used radial basis function network for forecasting the reference evapotranspiration. The weather parameter data used are (air temperature, relative humidity, wind speed, and sunshine) from January 1977 to December 1996. The results show that ANNs can be used for forecasting reference evapotranspiration with high reliability compared with traditional model.
- Tokar and Markus (2000) compared ANN models with traditional conceptual models in predicting watershed runoff as a function of rainfall, snow water equivalent, and temperature. The results indicated that ANNs can be powerful tools in modeling the precipitation-runoff process for various time scales, topography, and climate patterns.
- Tokar and Johnson (1999) employed ANN to forecast daily runoff as a function of daily precipitation, temperature and snowmelt for the Little Patuxent River watershed in Maryland. They used a three layer feedforward ANN and apply trial and error procedure to find the appropriate number of input nodes to the model. They reported that ANN provides reasonable good solutions for circumstances where there are complex systems that may be poorly defined or understood using mathematical

equations. ANN models provided higher training and testing accuracy when compared with regression and simple conceptual models.

- Finally, there are different studies which employ NNs to many applications in our life and others which are concerning on the comparing different kinds of NN algorithms like studies done by (Kisi et al, 2007) and (Asefa et al, 2007) and (Ardiclioglu et al, 2007).

# CHAPTER 3

## ARTIFICIAL NEURAL NETWORKS

### 3.1 Introduction

The development of ANN began approximately 75 years ago by McCulloch and Pitts in 1943, inspired by a desire to understand the human brain and emulate its functioning. Although the idea of ANN was proposed by McCulloch and Pitts over sixty five years ago, the development of ANN technique has experienced a renaissance only in the last decade, because the current algorithms overcome the limitations of early Networks. A tremendous growth in the interest of this computational mechanism has occurred by rediscovered a mathematically rigorous theoretical framework for Neural Networks, (ASCE, 2000).

Artificial Intelligence (AI) can be broadly defined as computer processes that attempt to emulate the human thought processes that are associated with activities that require the use of intelligence.

The term AI in its broadest sense, encompasses a number of technologies that includes, but is not limited to, expert systems, Neural Networks, genetic algorithms, fuzzy logic systems. Interestingly of artificial intelligence is that, the use of computers to model the behavior aspects of human reasoning and learning. In problem solving, one must proceed from a beginning (the initial state) to the end (the goal state) via a limited number of steps.

ANN, also called a simulated Neural Networks (SNN) or called parallel distributed processing system (PDPs) and connectionist systems, is an interconnected group of artificial neurons that uses a mathematical or computational model for information processing based on a connectionist approach to computation. There is no universally accepted definition of an NN. But perhaps most people in the field would agree that an NN is a network of many simple processors ("units"), each possibly having a small amount of local memory. The units are connected by communication channels ("connections") which usually carry numeric (as opposed to symbolic) data, encoded by any of various means. The units operate only on their local data and on the inputs they



receive via the connections. The restriction to local operations is often relaxed during training. ([ftp://ftp.sas.com/pub/neural/FAQ.html#A\\_cando](ftp://ftp.sas.com/pub/neural/FAQ.html#A_cando), 2008).

Most NNs have some sort of "training" rule whereby the weights of connections are adjusted on the basis of data. In other words, NNs "learn" from examples, as children learn to distinguish dogs from cats based on examples of dogs and cats. If trained carefully, NNs may exhibit some capability for generalization beyond the training data, that is, to produce approximately correct results for new cases that were not used for training.

Some NNs are models of biological neural networks and some are not, but historically, much of the inspiration for the field of NNs came from the desire to produce artificial systems capable of sophisticated, perhaps "intelligent", computations similar to those that the human brain routinely performs, and thereby possibly to enhance our understanding of the human brain. (<ftp://ftp.sas.com/>, 2008).

### 3. 2 Basic Structure

Figure(3.1) presents the ANN as a massively parallel collection of small and simple processing units where the interconnections form a large part of the Networks's intelligence. ANN, however, are quite different from the brain in terms of structure.

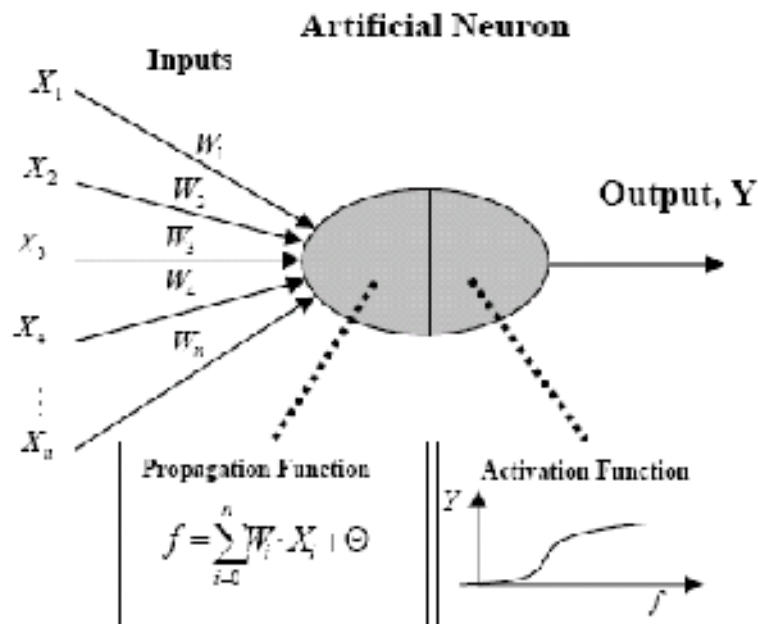


Figure (3.1): Neuron Unit

Structure of an ANN can be classified into 3 groups as per the by arrangement of neurons and the connection patterns of the layers: feed forward (error back propagation Networks), feedback (recurrent Neural Networks and adaptive resonance memories), and self-organizing (Kohonen Networks) (<ftp://ftp.sas.com>, 2007).

Also Neural Networks can be roughly categorized into two types in terms of their learning features: supervised learning algorithms, where Networks learn to fit known inputs to known outputs, and unsupervised learning algorithms, where no desired output to a set of input is defined. The classification is not unique and different research groups make different classifications.

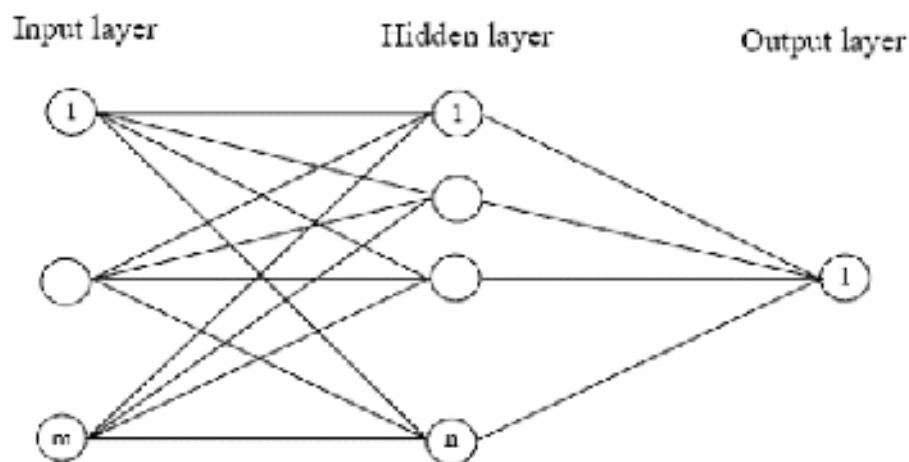


Figure (3.2): A three-layer Neural Networks

The feed forward Neural Networks (FFNNs) consist of three or more layers of nodes: one input layer, one output layer and one or more hidden layers as shown in figure(3.2). The input vector passed to the Networks is directly passed to the node activation output of input layer without any computation. One or more hidden layers of nodes between input and output layer provide additional computations. Then the output layer generates the mapping output vector. Each of the hidden and output layers has a set of connections, with a corresponding strength-weight, between itself and each node of preceding layer. (<http://www.usingneuralnetworks.com> 2007) and (<ftp://ftp.sas.com>, 2007).

### 3.3 Training

Training is the actual process of adjusting weight factors based on trial-and-error. The objective is to minimize the error between the target and actual output and to find weights. To train NN, the weight factors were adjusted until the calculated output pattern based on the given input matches the desired output. These weights are modified until the difference between the Networks output and the actual outputs are close to targets. This training procedure involves the iterative adjustment and optimization of connection weights and threshold values for each node in the Networks. Tsoukalas and Uhrig (1997) reported that, today it is estimated that 80% of all applications utilize this BackPropagation algorithm in one form or another. It is a gradient (derivative) technique that are simple to compute locally, and it performs stochastic gradient descent in weight space (for pattern by pattern updating of synaptic weights). (<http://www.usingneuralnetworks.com>2007), (<http://www.neuralnetwork.com> 2007) and (<ftp://ftp.sas.com>, 2007).

### 3.4 Architecture of Neural Networks

The architecture of a Networks is defined by the number of layers, the number of units per layer, and the interconnection patterns between layers.

#### 3.4.1 Feed-forward Networks

FFNNs, allow signals to travel one way only; from input to output. There is no feedback (loops). The output of any layer does not affect that same layer. FFNNs tend to be straight forward Networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down, as shown in Figure(3.3). (<http://www.usingneuralnetworks.Com> 2007), (<http://www.neuralnetwork.com> 2007) and (<ftp://ftp.sas.com>, 2007).

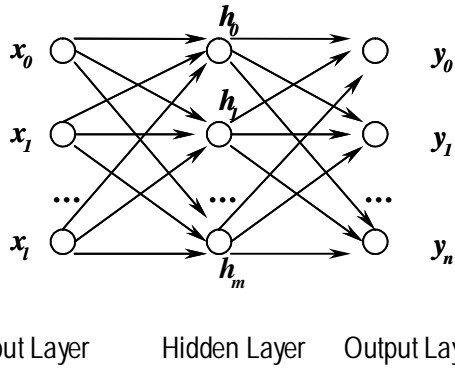


Figure (3.3) :An example of a simple feed forward Networks

### 3. 4.2 Feedback Networks

Feedback Neural Networks (FBNNs), Figure(3.4), have signals traveling in both directions by introducing loops in the Networks. FBNNs are very powerful and can get extremely complicated. FBNNs are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. (<http://www.usingneuralnetworks.com> 2007), (<http://www.neuralnetwork.com> 2007) and ( <ftp://ftp.sas.com>, 2007).

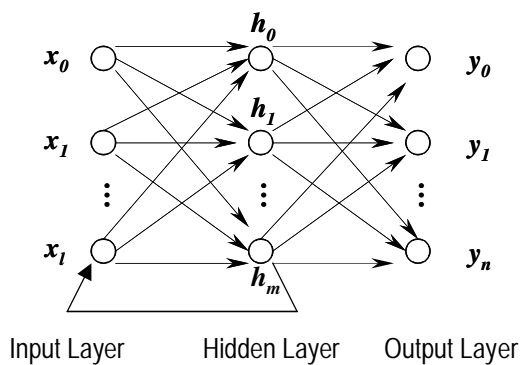


Figure (3.4): Recurrent Neural Networks

### 3.5 Research Methodology

As mentioned before, the DS model process is very complex, highly non-linear, and time varying. Hence, the application of NN methods can be able to describe accurately this process. In the DS modeling, the input nodes constitute the series of observation

(weekly observation from Jun-1992 To Jun-2000) and the output node consists of the DS surface water level data. The data is divided into two subsets. First is training data sets. Training data sets used to train the model, and also to validation data (test set). Validation process is carried out to monitor NN model performance during training. Meanwhile, the test data sets used to measure the model performance. The training data set is the data which the NN uses to learn the solution to the problem. The validation process is used to establish when to stop the algorithm that is to choose the best solution.

After every training iteration the validation data set is passed through the Networks, and the error over the data set is calculated. The best set of weights is defined as those that produce the lowest error over the validation data set. Meanwhile the test data set is the stage of using model for performance test.

The present study employs the supervised training NN models. Once the NN models have been created, their suitability for the application needs to be investigated. This task involves training the models and then testing the performance of the Neural system.

The training or learning phase is critical to the success of the NN. In this study, Multilayer Perceptron FFNN trained by BP, L-M and GRNN algorithms are applied to correct errors.

### **3.5.1 Multilayer Perceptron Neural Network (MLP-NN)**

MLP is a supervised and also FFNN with one or more layers of nodes between input and output nodes. It is a most commonly used Neural computing technique. Each node is the basic element of a Neural Networks called neuron.

The network consists of an input layer linked to the input DS variables, a hidden layer, and an output layer that connects to the output variables, Figure (3.5) illustrates the architecture of the proposed DS model. The decisions that affect the performance of the network models during training include the number of input nodes, the number of hidden nodes, learning rate, momentum constant and the transfer function. Input layer constitutes the input nodes or neurons. The number of input nodes in the input layer should be selected carefully in order to construct a good NN model. The accuracy of

model depends on the selection of input nodes derived from the characteristics of data series.

BackPropagation is numerically intensive technique, and there are many different ways to perform BP to teach the Neural nets how to respond. Any BP Networks is based on a supervised learning technique that compares the actual output from output units to the target or specified output and then readjusts the weights backward in the Networks (Tingsanchali, 2000), (<http://www.usingneuralnetworks.com> 2007), (<http://www.Neuralnetwork.com> 2007) and (<ftp://ftp.sas.com>, 2007).

In order to improve the usefulness of the steepest descent method, two parameters can be altered, the momentum coefficient and the learning coefficient. Learning rate is used to control the amount of weights adjustment at each step of training. The smaller the learning rate parameter  $\alpha$ , the smaller will the changes to the synaptic weight in the Networks from one iteration to the next. However, the smaller learning rate will take too long to reach the minimum.

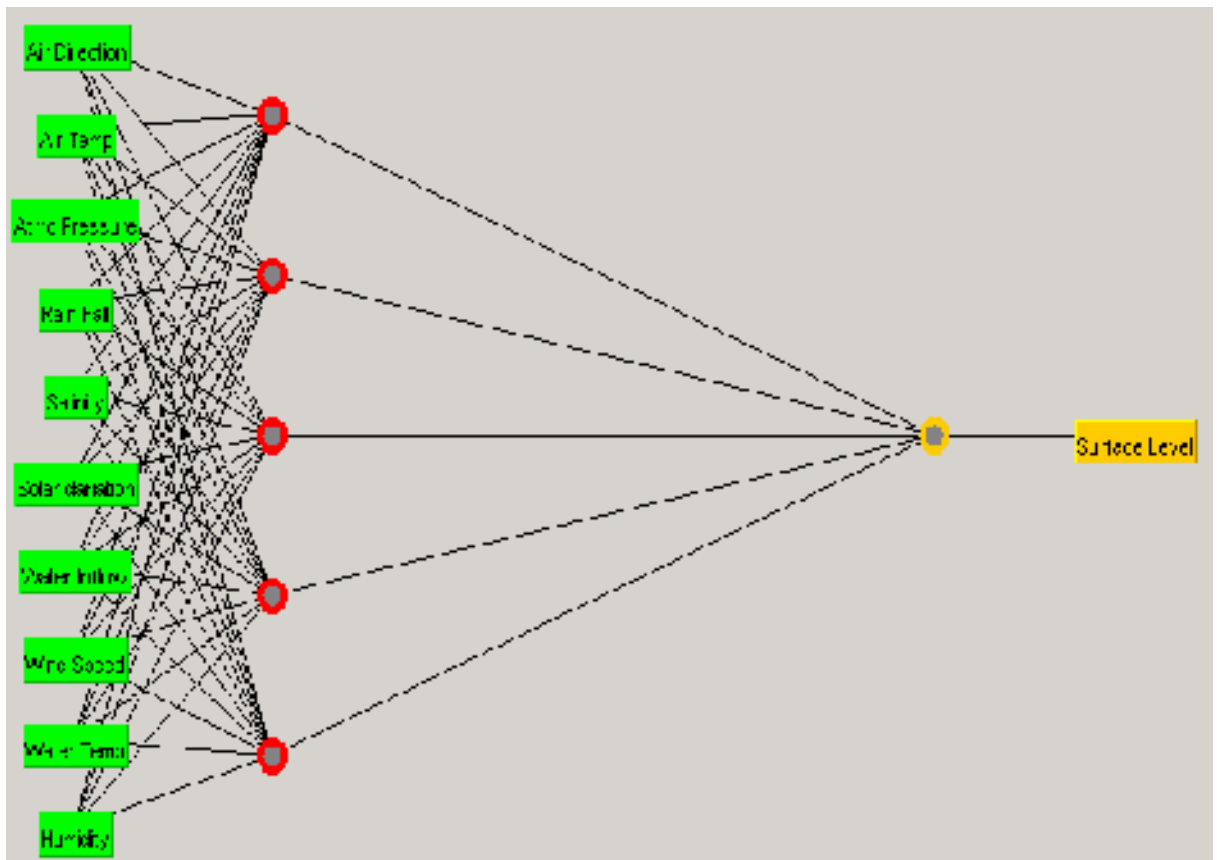


Figure (3.5): Structure of a DS surface water level model with one hidden layer

In the standard BP, the learning rate may be constant. There is no hard and fast rule about what value the learning coefficient should have (Fausett, 1994). A simple method of increasing the rate of learning is to include the momentum term,  $\mu$ . The momentum term is usually a positive number. The incorporation of momentum in the BP algorithm represents a minor modification to the weight update. The momentum term have the benefit of preventing the learning rate process from terminating in a shallow local minimum on the error surface. ( <ftp://ftp.sas.com>, 2007).

### 3.5.1.1 Architecture of MLP

Basically, NN model consists of an input layer linked to the input, a hidden layer, and an output layer that connects to the output. Input layer is the DS variables data and the output layer is surface water level data. The information of the data in the input layer transfer to the next consecutive layers in the system of FFNN. The activation function will process the signal send by input data that passes from each node. Associated with each incoming input signal is a weight.

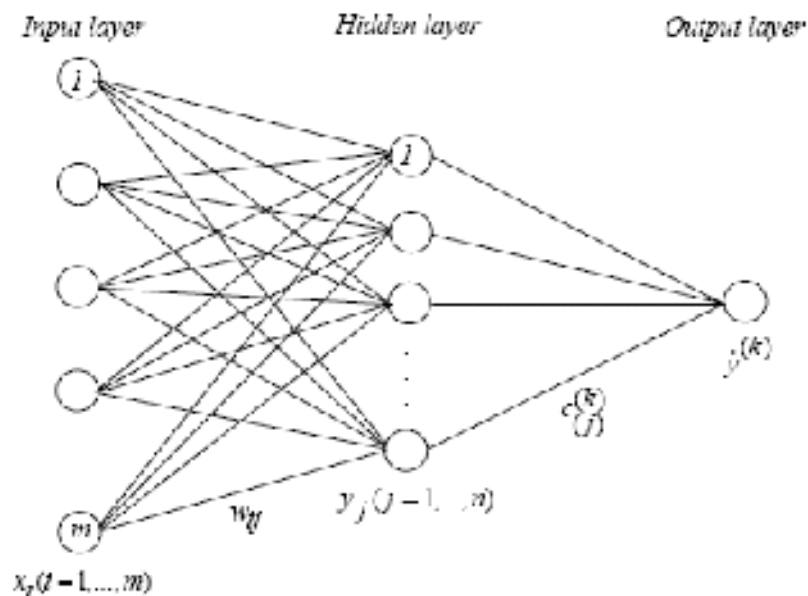


Figure (3.6): A three-layer Neural Networks

As shown in figure(3.6), Each input node unit ( $i = 1, \dots, m$ ) in input layer broadcasts the input signal to the hidden layer. Each hidden node ( $j = 1, \dots, n$ ) sums its weighted input signals, applies its activation function to compute its output signal, and sends this

signal to all units in the hidden layer, which  $W_{ij}$  is the weight between input layer and hidden layer,  $w_{0j}$  is the weight for the bias; and  $x_i$  is the input DS variables signal. In this work, as shown in Figure (3.7) the activation function applied here is sigmoid function which is the most activation function used, (ASCE 2000).

$$g(z) = \frac{1}{1 + e^{-z}} \quad 3.1$$

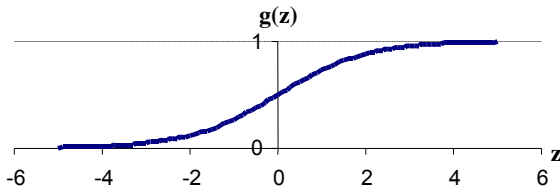


Figure (3.7): Sigmoid function

The sigmoid activation function will process the signal that passes from each node:

$$g(z) = \frac{1}{1 + e^{-y_k}} \quad 3.2$$

Then from second layer the signal is transmitted to third layer. The output unit ( $k=1$ ) sums its weighted input signals,

$$x_{in_k} = c_0^{(k)} + \sum_{j=1}^n z_j c_j^{(k)} \quad 3.3$$

and applies its activation function to compute its output signal,

$$y_o = f(x_{in_k}) \quad 3.4$$

where  $c_j^{(k)}$  is the weight between second layer and third layer; and  $c_0^{(k)}$  is the weight for the bias. The output node receives a target pattern corresponding to the input training pattern, computes its error information term,

$$\delta_k = (t_k - y_o) f'(x_{in_k}) \quad 3.5$$

calculates its weight correction term (used to update  $c_j^{(k)}$  later), and

$$\Delta c_j^{(k)} = \alpha \delta_k y_j \quad 3.6$$

calculates its bias correction term (used to update  $c_0^{(k)}$  later),

$$\Delta c_0^{(k)} = \alpha \delta_k \quad 3.7$$



where  $\alpha$  is learning rate;  $t_k$  is the target NN output;  $y_o$  is the NN output as net inflow variable. The error information is transfer from the output layer back to early layers. This is known as the BP of the output error to the input nodes to correct the weights. This method uses partial derivatives of error with respect to weights to update the weights of the connections. Each hidden unit ( $j=1, \dots, m$ ) sums its delta inputs (from units in the layer above),

$$\delta_{in j} = \sum_{k=1}^p \delta_k c_j^{(k)} \quad 3.8$$

calculates its weight correction term (used to update  $w_{ij}$  later),

$$\Delta w_{ij} = \alpha \delta_j x_i \quad 3.9$$

and calculates its bias correction term (used to update  $w_{0j}$  later),

$$\Delta w_{0j} = \alpha \delta_j \quad 3.10$$

The output unit ( $k=1$ ) updates its bias and weight s ( $j=0, \dots, n$ )

$$c_j^k (new) = c_j^k (old) + \Delta c_j^k \quad 3.11$$

and each hidden unit ( $j=1, \dots, n$ ) updates its bias and weights ( $i=0, \dots, m$ )

$$w_{ij} (new) = w_{ij} (old) + \Delta w_{ij} \quad 3.12$$

The weight update formulas for BP with momentum are,

$$\Delta c_j^{(k)} (t+1) = a \delta_k y_j + \mu \Delta c_j^{(k)} (t) \quad 3.13$$

and

$$\Delta w_{ij} (t+1) = a \delta_j y_i + \mu \Delta w_{ij} (t) \quad 3.14$$

The process is terminated when this difference is achieved a specified value. The training phase needs to produce a NN that is both stable and convergence, to produce accurate input-output relations. After this, the Networks can be tested using data have not been assigned during learning (ASCE 2000).

In general, the processes or procedures of BP algorithm can be summarized as follows:

- (1) Obtain a set of training patterns
- (2) Setup NN model (number of input neurons, hidden neurons, and output neurons)

- (3) Set a model parameters (learning rate  $\alpha$ , momentum rate  $\mu$ , etc)
- (4) Initialize all connection, weights, and biases, to random values  $w_{ij}$
- (5) Set minimum error,  $E_{\min}$
- (6) Start training by applying input and desired outputs and propagate through the layers then calculate total error.
- (7) Backpropagate error through output and hidden layer and adapt weights.
- (8) Backpropagate error through hidden and input layer and adapt weights.
- (9) Check if error  $< E_{\min}$ . If not repeat steps 6-9. If yes stop training.

### 3.5.1.2 Training NN using BackPropagation Algorithms

Two types of training algorithms approaches used to train NN, namely Gradient Decent and L-M algorithms.

Standard BackPropagation is a gradient descent algorithm, in which the Networks weights are moved along the negative of the gradient of the performance function.

This algorithm depends on minimizing an error function based on adjusting the weights of the NN, The error can computed as following,

$$E(W) = \frac{1}{2} \sum_{m=1}^M \sum_{i=1}^m (T_{im} - O_{im})^2 \quad 3.15$$

where M indexes the input vector, i is the iteration for network, m is number of neuron, and  $T_{im}$  and  $O_{im}$  are, the target and actual networks output for the ith output unit on the Mth pattern respectively, where

$$O_{im} = g\left(\sum_{k=1}^n w_{ik} e_{km}\right) \quad 3.16$$

$w_{ik}$  and  $e_{km}$  are the weight and input for the network respectively. The algorithm moves weight in direction opposite to gradient of error,

$$\Delta W_{ik}^M = -\alpha \frac{\partial E}{\partial W} \quad 3.17$$

$$= \alpha (T_{im} - O_{im}) * e_{km} \quad 3.18$$

where  $\alpha$  is the learning rate.

$$w_{k+1} = w_k + \Delta w \quad 3.19$$

On the other hand, the L-M is considered to be the most efficient algorithm for training median sized artificial neural networks. Like Quasi-Newton methods, the L-M algorithm was designed to approach second order training speed with out having to compute Hessian matrix (Burneyet al, 2005). When the performance function has the form of a sum of squares that is

$$F(\underline{w}) = \frac{1}{2} e^T e = \frac{1}{2} e^T(w) e(w) = \frac{1}{2} \sum_i^k \sum_j^p (O_{ij} - t_{ij})^2 \quad 3.20$$

where  $\underline{W} = [w_1, w_1, w_1, \dots, w_N]^t$  consists of all weights of the network, the function of sum of squared errors is defined as

$$F(w) = \frac{1}{2} e^t e \quad 3.21$$

Newton's method for minimizing objective function is generated using well known recurrence formula.

$$w_{i+1} = w_i - H^{-1} \nabla F(w) \quad 3.22$$

when  $F(w) = \frac{1}{2} e^T e$  and  $\nabla F(w)$  is the gradient of F(W) then the Hessian matrix

can be approximated as,

$$H = J^T J \quad 3.23$$

and the gradient can be computed as

$$g = J^T \underline{e} \quad 3.24$$

where the Jacobian matrix J contains the first derivatives of the network errors with respect to weights and biases, and e is a vector of network errors. The Gauss-Newton update formula can be

$$w_{i+1} = w_i - (J_i^T J_i)^{-1} J_i^T \delta_i \quad 3.25$$

$$J = \begin{bmatrix} \frac{\partial e_{11}}{\partial w_1} & \frac{\partial e_{11}}{\partial w_2} & \dots & \frac{\partial e_{11}}{\partial w_N} \\ \frac{\partial e_{21}}{\partial w_1} & \frac{\partial e_{21}}{\partial w_2} & \dots & \frac{\partial e_{21}}{\partial w_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{k1}}{\partial w_1} & \frac{\partial e_{k1}}{\partial w_2} & \dots & \frac{\partial e_{k1}}{\partial w_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{1p}}{\partial w_1} & \frac{\partial e_{1p}}{\partial w_2} & \dots & \frac{\partial e_{1p}}{\partial w_N} \\ \frac{\partial e_{2p}}{\partial w_1} & \frac{\partial e_{2p}}{\partial w_2} & \dots & \frac{\partial e_{2p}}{\partial w_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{kp}}{\partial w_1} & \frac{\partial e_{kp}}{\partial w_2} & \dots & \frac{\partial e_{kp}}{\partial w_N} \end{bmatrix} \quad 3.26$$

Where  $(J^T J)$  is positive definite, but if it is not, then, we make some perturbation into it that will control the probability of being non positive definite.

Thus,

$$H \approx J^T J + \mu I \quad 3.27$$

$$w_{i+1} = w_i - (J_i^T J_i + \lambda I)^{-1} J_i^T \delta_i \quad 3.28$$

Where in neural computing context the quantity  $\lambda$  is called the learning parameter, it ensures that  $J^T J$  is positive definite. It is always possible to choose  $\lambda$  sufficiently large enough to ensure a descent step. The learning parameter is decreased as the iterative process approaches to a minimum.

Thus, in Levenberg-Marquardt optimization, inversion of square matrix  $J^T J + \lambda I$  is involved which requires large memory space to store the Jacobian matrix and the approximated Hessian matrix along with inversion of approximate H matrix of order  $N \times N$  in each iteration, thus for large networks, Levenberg-Marquardt algorithm is not suitable (Burneyet al, 2005).

### 3.5.2 Radial Basis Function (RBF) Model

The Radial Basis Function (RBF) neural network was proposed by Broomhead and Lowe. This neural network is very different from neural networks with sigmoidal activation functions in that it utilizes basis functions in the hidden layer that are locally responsive to input stimulus. These hidden nodes are usually implemented using a Gaussian kernel function (Heimes and Heuveln, 1998).

#### 3.5.2.1 Architecture of RBF

The RBF training procedure requires that a clustering algorithm determine the location (center) of each Gaussian kernel and the width of each kernel. Once the hidden layer nodes are selected the output layer weights can be determined analytically using a Least Mean Squares (LMS) method which determines the optimum output layer weights based on the training data and the selected hidden node parameters.

The hidden layer of the RBF compares a new input vector ( $x$ ) with a number of stored pattern vectors ( $c$ ) based on an arbitrary distance function  $\|x-c\|$ . (Heimes and Heuveln, 1998).

Usually the distance function is the Euclidean norm with an additional normalization constant in each dimension:

$$\|x - c\| = \sqrt{\sum_{k=1}^n \left(\frac{x_k - c_k}{\sigma_k}\right)^2} \quad 3.29$$

where  $x$  and  $c$  are both vectors of length  $n$  and  $\sigma_k$  is the normalization constant that controls the width of the basis function. A clustering algorithm is normally used to define the number, location, and widths of the centers of the basis functions. It is obvious that the more basis functions that are included in the network the better its accuracy. However, the number of basis functions normally needs to be limited for practical reasons.

This distance measure is then used to evaluate the basis functions:  $\Phi\|x-c\|$ . (Heimes and Heuveln, 1998).

If one uses a Gaussian basis function the hidden layer outputs are evaluated as follows

$$\phi (\|x -c \|) = e^{-\|x-c \|^2} \quad 3.30$$

Given a network with p basis functions, a predicted scalar output  $y'_j$  is computed using the following equation:

$$y'_j = \sum_{i=1}^p \lambda_{ij} \phi (\|x -c_i \|) \quad 3.31$$

Since the output layer equation is a pure linear transformation, the coefficients  $\lambda_{ij}$  are analytically determined using the LMS algorithm. Consider matrix A to be the q-by-p matrix, where:

$$A_{li} = \phi (\|x_l -c_i \|) \quad l = 1, 2, \dots, q, \quad i = 1, 2, \dots, p \quad 3.32$$

In equation (3.32), each  $x_l$  is one of q training vectors for which the actual output is known to be an m element vector  $y_l$ . Using A, we can express the computation of the RBF output layer as a matrix multiplication:

$$y' = A \lambda \quad 3.33$$

where  $\lambda$  is a p-by-m matrix of weights, A is a q-by-p matrix of hidden layer outputs for all training patterns, and  $y'$  is a q-by-m matrix of network outputs. Since ideally the prediction equals the desired output:

$$y' = y \quad 3.34$$

we can get an optimum prediction based on the training data by setting the matrix of weights ( $\lambda$ ) according to:

$$\lambda = A^* y \quad 3.35$$

In Equation (3.35),  $A^*$  is either the inverse if  $p = q$ , or the pseudo inverse if  $p < q$ , (Heimes and Heuveln, 1998).

### 3.5.2.2 Training RBF Networks

Adapting the values of the weights and centers of Networks by presenting the input and output data is known as learning or training. Training is the actual process of adjusting

weight factors based on trial-and-error. A supervised training requires target patterns or signals to guide the training process. The objective is to minimize the error between the target and actual output and to find weights. To train RBF Networks, the weight factors were adjusted until the calculated output pattern based on the given input matches the desired output

There are several types of learning algorithms can be used in RBF Networks such as Orthogonal Least Squares (OLS), GRNN, K-means Clustering, and Probability Density Function (PDF).

The GRNN algorithm were used is a kind of Radial Basis Neural Network (RBNN) that is often used for function approximation. The GRNN was introduced as a memory based NN that would store all the independent and dependent training data available for a particular mapping (Heimes and Heuveln, 1998).

GRNN can be designed very quickly, fast learning, and effectively uses historical data to estimates values for continuous dependent variables. The learning process is equivalent to finding a surface in a multidimensional space that provides a best fit to the training data, with the criterion for the 'best fit' being measured in some statistical sense. Using the features of learning and training processes which it learned from past experience, or generalization of previous examples, RBF is capable of performing a basis for system modeling and forecasting (Heimes and Heuveln, 1998).

The GRNN predicts the value of one or more dependent variables, given the value of one or more independent variables. According to Heimes and Heuveln (1998) and Specht (1991), the GRNN thus takes as an input vector  $x$  of length  $n$  and generates an output vector (or scalar) of length  $y$  of length  $m$ , where  $y$  is the prediction of the  $y$ . The GRNN does this by comparing a new input pattern  $x$  with a set of  $p$  stored patterns  $x^i$  (pattern nodes) for which the actual output  $y_i$  is known. The predicted output  $y$  is the weighted average of all these associated stored output  $y_{ij}$ . Equation 3.36 expresses how each predicted output component  $y'_j$  is a function of the corresponding output components  $y_j$  associated with each stored pattern  $x^i$ . The weight  $W(x, x^i)$  reflects the contribution of each known output  $y_i$  to the predicted output. It is a measure of the

similarly of each pattern node with the input pattern, (Heimes and Heuveln, 1998).

$$y_j = \frac{N_j}{D} = \frac{\sum_{i=1}^p y_{ij} w(x, x^i)}{\sum_{i=1}^p w(x, x^i)} \quad j = 1, 2, \dots, m \quad 3.36$$

It is clear from equation 3.36 that the predicted output magnitude will always lie between the minimum and maximum magnitude of the desired output,  $y_{ij}$  associated with the stored patterns (since  $0 \leq w \leq 1$ ). In the GRNN algorithm, the output weights are set to the desired outputs. The GRNN is best seen as an interpolator, which interpolates between the desired outputs of pattern layer nodes that are located near the input vector (or scalar) in the input space.

A standard way to define the similarity function,  $W$  is to base it on a distance function,  $D(x_1, x_2)$ , that gives a measure of the distance or dissimilarity between two patterns  $x_1$  and  $x_2$ . The desired property of the weight function  $w(x, x^i)$  is that its magnitude for a stored pattern  $x^i$  be inversely proportional to its distance from the input pattern  $x$  (if the distance is zero the weight is a maximum of unity). The standard distance and weight functions are given by the following equations, respectively:

$$w(x, x^i) = e^{-D(x, x^i)} \quad 3.37$$

$$D(x_1, x_2) = \sum_{k=1}^n \left[ \frac{x_{1k} - x_{2k}}{\sigma_k} \right]^2 \quad 3.38$$

In equation 3.38, each input variable has its own sigma value,  $\sigma_k$ , where  $\sigma_k$  is the normalization constant that controls the width of the basis function.

The procedures of GRNN algorithm can be summarized as follows:

- (1) Input unit stores an input vector  $x$ .
- (2) The pattern units which computes the distances  $D(x_1, x_2)$  between the incoming patterns  $x$  and stored patterns  $x^i$ . The pattern nodes output the quantities  $W(x, x^i)$ .
- (3) The summation units computes  $N_j$ , the sums of the products of  $W(x, x^i)$  and the associated known output component  $y_i$ . This unit also has a node to compute  $D$ , the sum of all  $W(x, x^i)$ .



- (4) Finally, the output unit divides  $N_j$  by  $D$  to produce the estimated output  $y'_j$  component that is a localized average of the stored output patterns.

### **3.5.3 Selection of Networks Structures**

Networks defined by various combinations of DS variables sets at present and previous time periods were trained and tested using different NN configurations. For all different combinations of the input variables, Networks were trained using a one hidden layer. In this particular study, the structure of NN model is designed based on trial and error procedure to find the appropriate number of neuron and input variables to the model.

# CHAPTER 4

## DESIGN AND IMPLEMENTATION

### 4.1 The Study process

The method that was conducted in this work can be explained into steps as follows:

- Collect the data .
- Find the most significant data set.
- Generate different subsets of data using InfoGainAttributeEval algorithm (feature selection).
- Select the Data Mining Method (we choose NN).
- Built different NN models.
- Compare different NN models based on the most significant data set.
- Analyzing the result.
- Determine the most accurate and reliable model that can be used for future prediction.

Study processing starts from the data collection and analysis, followed by pre-processing and then feeds to the neural network. Finally, post-processing is needed to transform the outputs of the network to the required outputs. Figure (4.1) shows the process of this study.

This chapter discusses some of the most important considerations involved in processing data for neural networks.

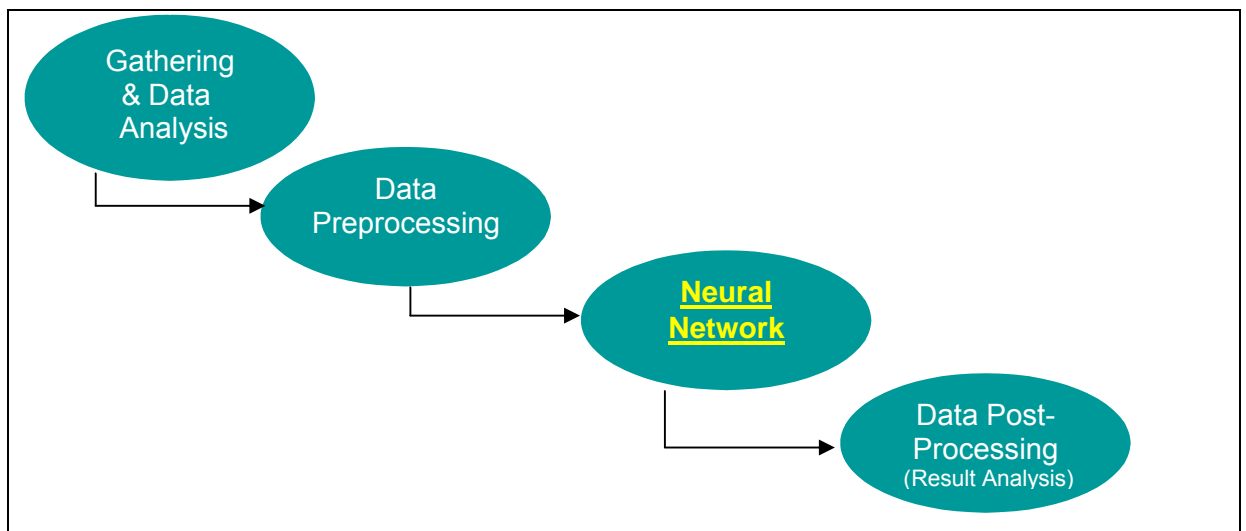


Figure 4.1 :study process

## **4.2 Data Set**

One of the most important components in the success any research is the data. The quality, availability, reliability, repeatability, and relevance of the data used to develop and run the system is critical to it's success. Even a primitive model can perform well if the input data has been processed in such a way that it clearly reveals the important information.

### **4.2.1 Data Pre-Processing**

Theoretically, a neural network could be used to map the raw input data directly to required output data. But sometimes it is critical to apply pre-processing to the input data before they are fed to a network. There are many techniques and considerations relevant to data pre-processing. Pre-processing can vary from simple filtering (as in time-series data), to complex processes for extracting features from image data. Since the choice of pre-processing algorithms depends on the application and the nature of the data, the range of possibilities is vast.

### **4.2.2 Data Post-Processing**

Post-processing covers any process that is applied to the output of the network. As with pre-processing, it is entirely dependent on the application, or it may using the output of a network as one input to a rule-based processor. Sometimes it is just the reverse process of data pre-processing.

## **4.3 Dead Sea Variables selection**

Variable selection improves prediction by searching for the subset of features, which can best predict the future target. The variables that will be taken into account are:

1. AD: Air Direction
2. At: Air Temperature
3. Pr: Atmosphere Pressure
4. Ra: Rain fall
5. Sa: Salinity

6. Ws: Wind Speed
7. Wt: Water Temp
8. H: Humidity
9. So: Solar radiation
10. In: Water Inflow
11. Sl: Surface Level of the DS

From the data set which consists of 384 vectors, we have chosen 300 as a training set, and 84 records as testing set. Also, we have generated different 6 sets of data with different combination of variables from both training and testing sets using Ranker Selection Attribute algorithm provided by Weka Program. (for identifying the most significant set of variables). We applied these datasets to evaluate their performance and select the best set among the other sets.

The variable chosen comes from Jordan weather station, Isramer and from annual reports and journals. These organizations were chosen based on several criteria. First and foremost, the quality and quantity of data for each one was individually screened. Some of them had numerous days of missing data in a year that make the annual maximum values questionable. The amount of data available for each organization varies between 6 years to up of 100 years according to some of organization. So in this study we take (Jan 1992 – Jan 2000) period as a raw data in this research cause in this period the most significant variables are available and there isn't missing data.

In this study, the mathematical programming based on neural network methods was applied to model the DS water level relationship. A total of more than three resources have been defined where its located in Jordan, Palestine and from the World Wide Web (WWW) as a scientific journals or annual reports.

The source data which are the basic of this work, gathered from:

- The Hashemite kingdom of Jordan ( Meteorological department )  
[ Mean relative humidity, direction & speed of wind and rainfall ]
- The ISRAMAR National repository and dissemination facility for oceanographic data and data products.  
[air temp , water temperature below 100 m, solar radiation, atmosphere pressure, and surface level ]

- Other variables like surface & ground water extracted from electronic scientific papers such as Salamah and Elnasser authors in ( Does the Actual Drop in Dead Sea Level Reflect the Development of Water Sources Within its Drainage Basin?)

The data were selected from these resources dose not have uniform scale, Figure(4.1a) &(4.1b) represent some of data selected from Isramar Organization, all data forms from this organization are graphical form.

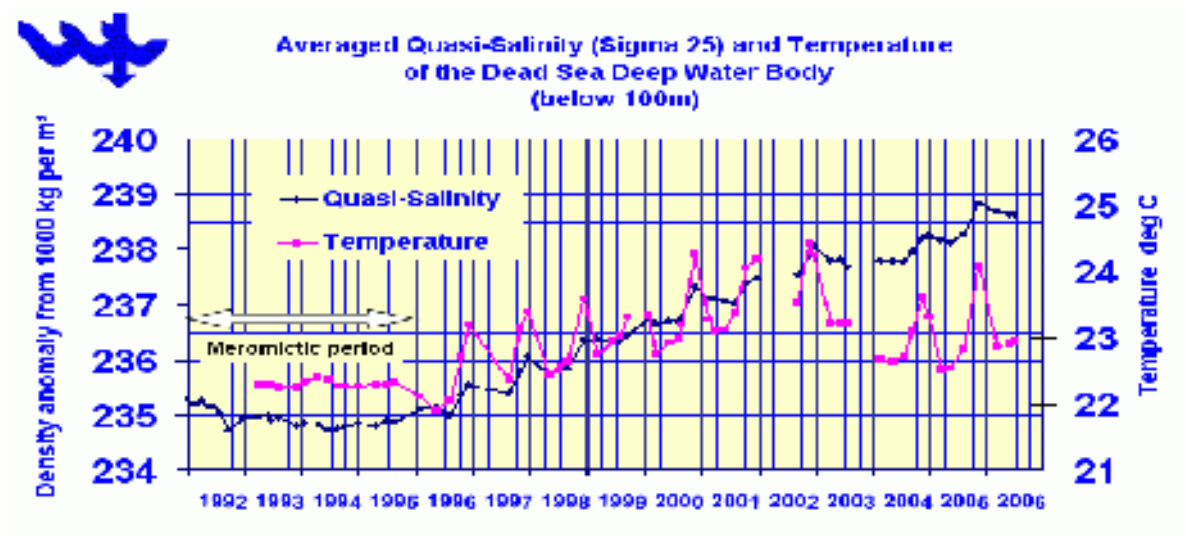


Figure (4.1a): Average Salinity and Temperature of DS under 100 meter below sea level

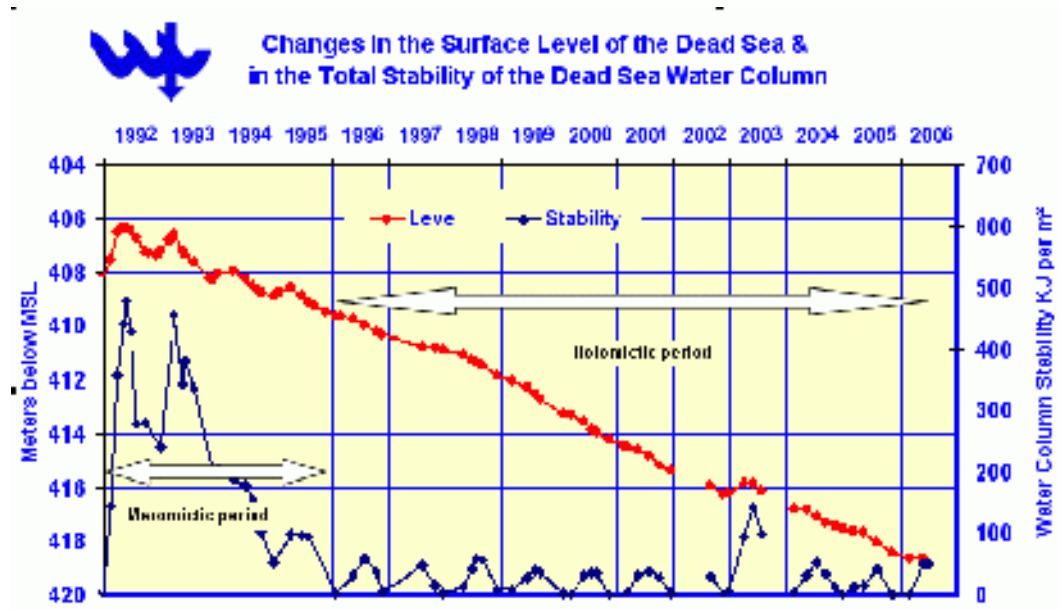


Figure (4.1b): Surface level of DS (1992-2006)

On the other hand the data were obtained from Meteorological department state in Amman-Jordan Table (4.1) represent some of the data. Other data collected from the net and from Journals concerning in DS area, Fig (4.2).

The continuous data where gathered is between Jan-1992 till Jan-2000, from those eight year we extracted one observation per week, four for one month and as a result we obtained 384 observation for eight year. The data must be manipulated (as we discussed earlier) by different tools to be have same scale (numerical scale) and it was prepared as input to our NN models, we used a GRABIT program written by Jiro Doke since 2003 using Matlab programming language to convert graphical data to numerical data, Figure (4.3).

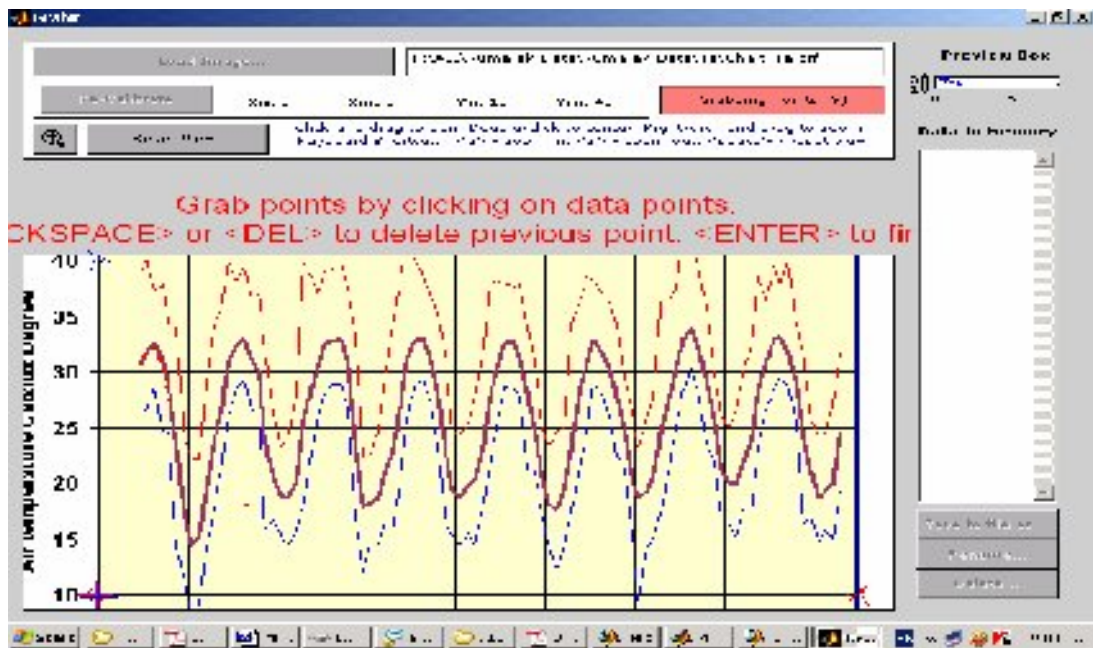
Table (4.1): humidity values (1988-2005)

<b>Dec</b>	<b>Nov</b>	<b>Oct</b>	<b>Sep</b>	<b>Aug</b>	<b>Jul</b>	<b>Jun</b>	<b>May</b>	<b>April</b>	<b>March</b>	<b>Feb</b>	<b>Jan</b>	
56	45	53	46	46	43	42	39	46	62	64	68	<b>1988</b>
62	53	51	49	53	49	47	41	37	56	45	60	<b>1989</b>
47	43	47	54	52	50	48	49	52	55	63	62	<b>1990</b>
58	49	45	51	53	49	44	43	46	57	54	57	<b>1991</b>
52	33	34	38	40	40	39	40	48	51	63	56	<b>1992</b>
57	47	42	48	49	48	46	50	45	55	71	53	<b>1993</b>
58	63	47	50	57	52	51	48	42	65	62	63	<b>1994</b>
57	47	61	62	52	48	43	39	48	58	66	60	<b>1995</b>
50	35	44	49	50	48	55	53	54	68	63	65	<b>1996</b>
77	64	44	43	43	41	49	42	48	61	50	50	<b>1997</b>
53.1	66.2	60.1	63.5	66.5	61	63	58.4	59.5	66.6	70.1	76.5	<b>1998</b>
52.3	36.8	44.9	44.1	44.6	42.7	55	55.6	59.8	59.4	63.4	59.9	<b>1999</b>
74	44	59	65	64	58	57	55	58	63	63	68	<b>2000</b>
73.4	56.9	59.6	63.9	63.1	60.5	55	52	55.9	55.1	73.2	71.3	<b>2001</b>
80	52.7	66.9	56.1	58.6	61.8	62.4	65.9		57.3	54.8	69.1	<b>2002</b>
70.5	49.7	65.8	49.3	34	34.5	64.2	53	69.8	81.4	83.9	75.6	<b>2003</b>
67.7	59.2	49.4	52.2	60.9	70.9	62.5	68.5	68.4	73.9	84.1	78.9	<b>2004</b>
59.3	52.5	48.3	50.9	53.6	50.8	50.4	48.6	46.7	59.1	68.4	63.8	<b>2005</b>

The average outflow from Lake Tiberias	40 Mio m <sup>3</sup> /a
The average surface runoff of the western JR catchment	10 Mio m <sup>3</sup> /a
The average surface runoff of the Yarmouk	50 Mio m <sup>3</sup> /a
Eastern side of the Jordan River	42 Mio m <sup>3</sup> /a
Eastern side of the DS (surface)	147 Mio m <sup>3</sup> /a
Eastern side of the DS (ground)	90 Mio m <sup>3</sup> /a
Western side of the DS (surface)	13 Mio m <sup>3</sup> /a
Western side of the DS (ground)	50 Mio m <sup>3</sup> /a
Western wadi Araba	10 Mio m <sup>3</sup> /a
Eastern wadi Araba	10 Mio m <sup>3</sup> /a
Irrigation return flows + subsurface flows and saltwater diversions into the JR	85 Mio m <sup>3</sup> /a
Precipitation over the DS (750 km <sup>2</sup> )	70 Mio m <sup>3</sup> /a
<b>Total Inflows</b>	<b>617 Mio m<sup>3</sup>/a</b>

Figure (4.2): Inflow of water

GRABIT starts a Graphical User Interface (GUI) program for extracting data from an image file. It is capable of reading in BMP, JPG, TIF, GIF, and PNG files. Multiple data sets can be extracted from a single image file, and the data is saved as an n-by-2 matrix variable in the workspace. And it can also be renamed and saved as a MAT file (Matrix file generated by MATLAB).



Figure(4.3):GRABIT program

The other data from Jordan meteorological department manipulated using Excel Microsoft Office to become same format as other observed data, as a result all variable

have the same scale and ready to be entered as a vector to the NN, Table (4.2) show all factors of the DS after manipulated.

Table (4.2): factors where effected the DS water level

<b>Ad</b>	<b>At</b>	<b>Pr</b>	<b>Ra</b>	<b>Sa</b>	<b>So</b>	<b>In</b>	<b>As</b>	<b>Wt</b>	<b>H</b>	<b>SL</b>
312	30.8	1053.4	0	235.08	7.7627	12.5	4.16	22.16	40	406.61
310	31.05	1053.2	0	235.06	7.6029	12.5	4.22	22.164	44	406.67
315	31.3	1053.1	0	235.04	7.3498	12.5	4.21	22.18	38	406.71
311	31.5	1053.1	0	235.01	7.2033	12.5	4.16	22.18	38	406.79
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
338	33.11	1050.8	0	236.7	7.46	12.5	3.67	23.532	59	414.03

In this work, the input vector of the neural network for both MLP and RBNN represents a set of variables (five variables for each dataset) generated by selection algorithm provided by Weka data mining software.

We take 6 datasets (the most sets related to surface level) for training and testing (300 records for training and 84 records for testing). Then we applied these datasets to NN algorithms.

## 4.4 NN Parameters

### 4.4.1 Learning threshold

The error range of training set that can be accepted in stopping training can be identified by learning threshold. It depends upon how complicated the network is to stop the training, the learning threshold where used equal to 0.0005.



#### **4.4.2 Learning rate ( $\alpha$ )**

The rate at which NNs learn depends upon several controllable factors. In selecting the approach there are many trade-offs to consider. Obviously, a slower rate means a lot more time is spent in accomplishing the off-line learning to produce an adequately trained system. With the faster learning rates, however, the network may not be able to make the fine discriminations possible with a system that learns more slowly. The learning rate used equal to 0.3.

#### **4.4.3 Momentum ( $\mu$ )**

Momentum factor describes the proportion of the weight change that is added to each subsequent weight change. Low momentum causes weight oscillation and instability, preventing the network from learning. High momentum cripples network adaptability. Momentum is limited to vary from 0 to 1, we set this value to 0.2 in our study.

#### **4.4.4 Number of hidden neurons**

Number of hidden nodes is the number which identifies how complex the network is. Hidden nodes are the additional modification layer to change the output function from a linear function to a nonlinear function. Generally, connections are allowed from the input layer to the first (and possibly only) hidden layer; from the first hidden layer to the second and from the last hidden layer to the output layer. The hidden layer learns to recode the inputs. More than one hidden layer can be used. There is no set formula for determining the number of hidden nodes to use. in our study we set the number of neurons to 5, 11 for BP and L-M respectively.

### **4.5 Performance measurement**

After selecting the most related datasets to DS surface level, each dataset applied to each network of three learning algorithms which are BackPropagation and Levenberg-Marquardt, and the General Regression Neural Network. The performance of these models have to be checked to know the accuracy. There are several measurement

methods to test the performance of model. The most accurate result compared to others will be implemented, which makes the performance checking of the model an important step. The accuracy measurement (both training and testing) used in this study is the Mean Square Error (MSE).

#### 4.5.1 Mean Squared Error (MSE)

The mean squared error is defined as the average of the squared forecast error.

$$MSE = \frac{\sum (A_t - T_t)^2}{n}$$

Where,

- $A_t$  : Actual value at time t
- $T_t$  : Target value at time t
- n : Number of time period n

As we mentioned before, we have 6 datasets for training and testing (300 records for training and 84 records for testing). We applied these datasets to evaluate their performance.

Each network tested after the training stage by the same learning data set, then subtracted the target value from the predicted value giving the error of the network, then we compared each network model by comparing the MSE for each one of the sets.

#### 4.6 Design the Neural Networks

From the data set which consists of 384 vectors, we have chosen 300 for training set, and the rest for testing, we applied these data to evaluate their performance, For BP, the network consisted of 5 neurons in hidden layer, 11 neuron for L-M, a spread of 1 was a default for GRNN, and after training, the Best MSE given by GRNN of spread 0.1.

- The DS parameters divided to six sets, each one have five parameter, these parameter was determined by Ranker Selection Attribute provided by Weka Program.
- The architecture of BP and L-M are (5-5-1) and (5-11-1) respectively.

- Spread was 0.1, 0.2, 0.5, 1, and 100 for GRNN.
- Mean squared error (MSE) was chosen as error performance algorithm. Goal error of 0.0005 was chosen.

#### **4.7 Tools and Platforms**

In this study, the computer programming used in this study is MATLAB copyrighted by MathWorks Inc.(2007) and Weka Program using Java as tool to develop model structures for determine the relationship between factors of DS. The modeling technique approach used in the present study is based on artificial neural network method in modeling input-output relationship.

MATLAB can be incorporated effectively to enhance understanding and enabling the researcher actively to put theory into practice. This software is known are friendly user and flexible with high capability for analysis and design the geology processes.

WEKA stands for Waikato Environment for Knowledge Analysis and is a very popular open-source Data Mining tool written in Java. It is developed at the University of Waikato in New Zealand and can be downloaded for free from the web (<http://www.cs.waikato.ac.nz/~ml/weka/>). It's applicable with Windows, Linux and basically every other operating system with a Java Virtual Machine of version 1.4 or higher.

# CHAPTER 5

## EXPERIMENTAL RESULTS

### 5.1 Results

The training and simulation were released using Weka Program and MATLAB 7 Neural Network. Three different neural network structures, which are BackPropagation, Levenberg Marquardt and General Regression Neural Network. Six Different datasets of DS variables were applied.

The variable used in datasets,

- Set number one = {Air Direction, Atmosphere Pressure, Water Inflow, Wind Speed, Rain fall }.
- Set number two = {Humidity, Wind Speed, Water Temp, Rain fall, Atmosphere Pressure }.
- Set number three = {Atmosphere Pressure, Solar radiation, Air Direction, Air Temperature, Rain fall}.
- Set number four = {Salinity, Air Direction, Water Inflow, Humidity, Atmosphere Pressure}.
- Set number five = {Air Temperature, Atmosphere Pressure, Water Inflow, Water Temp, Rain fall}.
- Set number six = { Air Temperature, Water Temp, Solar radiation, Humidity, Salinity }

Each set of these sets has one observation per week during eight years ago, total observations for each set is 384 observations.

In this study, all neural network models were applied to these sets by different parameters, and data sets were divided into training set and test sets.

The number of neurons was different from neural network to another, as discussed in the previous chapter. The error goal for MLP training function networks was 0.0005. For GRNN, the optimum spread values founded by trial-and-error and used for both the training and test data. The spread values were used are 0.1, 0.2, 0.5, 1 and 100 respectively.

The results of this study are summarized as follows:

- The performance of GRNN was better than other at all sets, and the performance of the BP & ML approximately closed to other.

GRNN performance for all sets was better than the other networks and as shown in table(5.2) the error for all sets nearly closed to zero and the best set with minimum error achieved by set number 4, figure(5.1) show the set 4 error performance.

Table (5.1): GRNN Training & Testing values for Spread equal 0.1

GRNN	Spread Value	Training Error	Testing Error
Set 1	0.1	0.0044	1.11 e-07
Set 2	0.1	0.0025	5.4 e-005
Set 3	0.1	0.004	0.006
Set 4	<b>0.1</b>	<b>0</b>	<b>1.01e-025</b>
Set 5	0.1	0.126	3.4 e-005
Set 6	0.1	1.10 e-004	2.07 e- 006

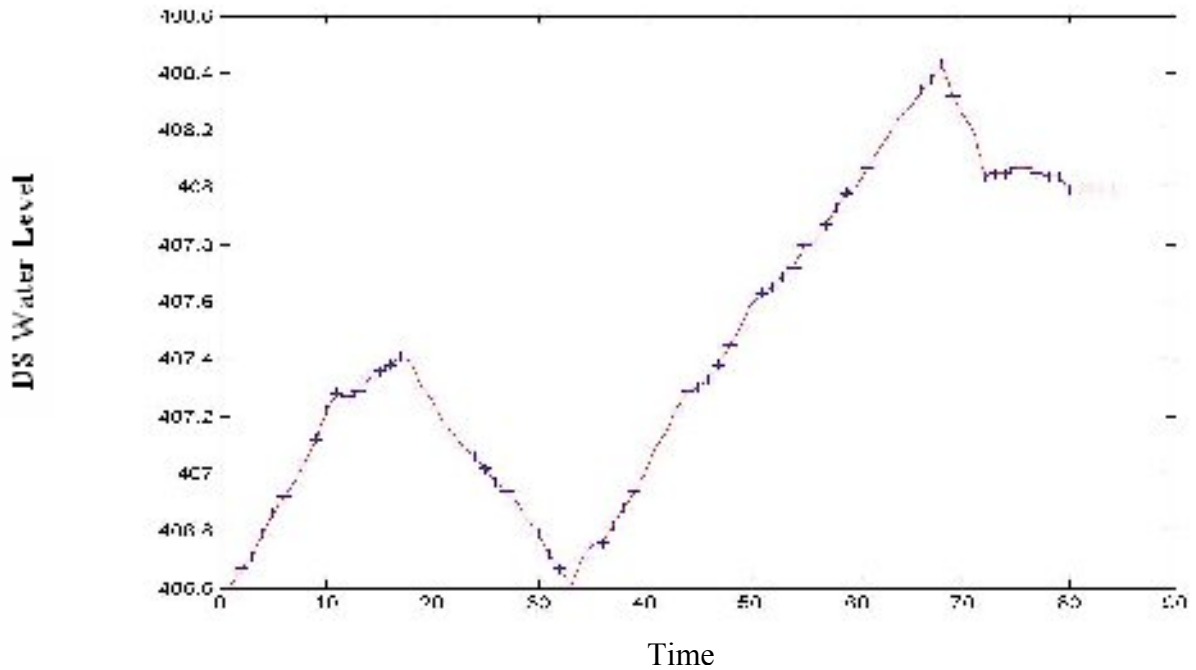


Figure (5.1): the performance of GRNN when spread value equal 0.1 for set 4

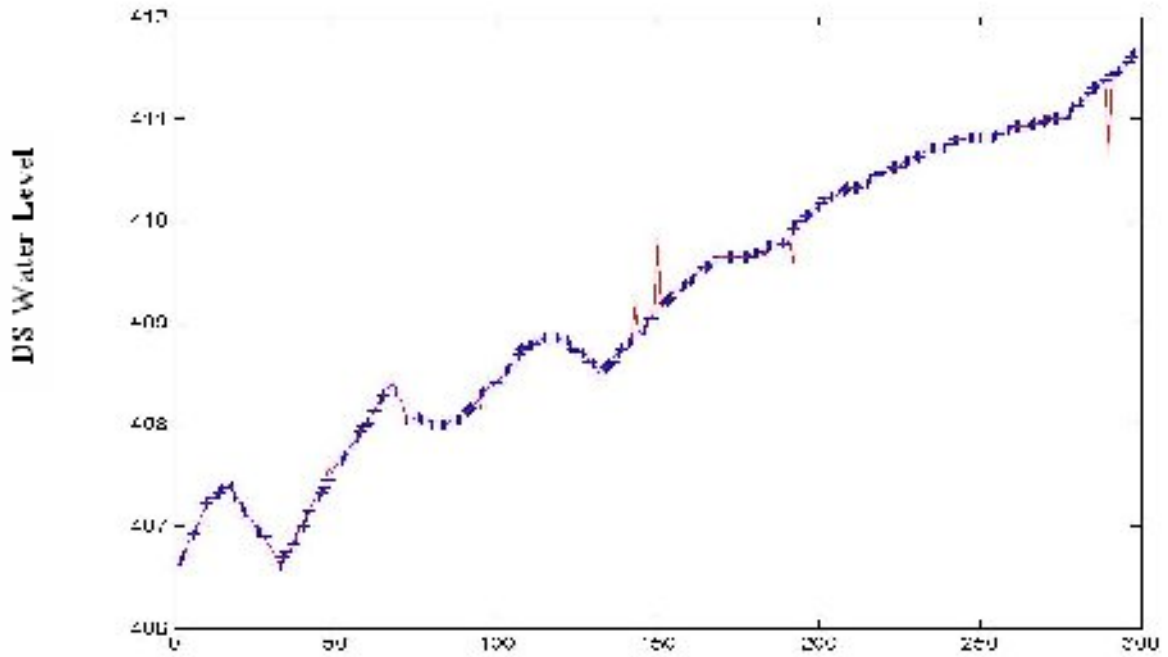


Figure (5.2): the performance of GRNN when spread value equal 0.1 for set 1

Table (5.2): L-M Training & Testing values for number of neurons is 11

L-M	Number of neuron	epochs	Training Error	Testing Error
Set 1	11	40000	1.908	2.9954
Set 2	11	40000	0.9768	1.1292
Set 3	11	40000	1.0053	2.7677
Set 4	11	40000	1.6420	1.0606
Set 5	11	40000	1.7023	1.1039
Set 6	11	40000	1.8888	0.9227

As show in table(5.2), the best performance value for L-M network (5-11-1) founded in set number 6 having the minimum error comparing with other sets equal to 0.9227, figures (5.3) and (5.4) shows the performance of L-M for set 4 and set 6.

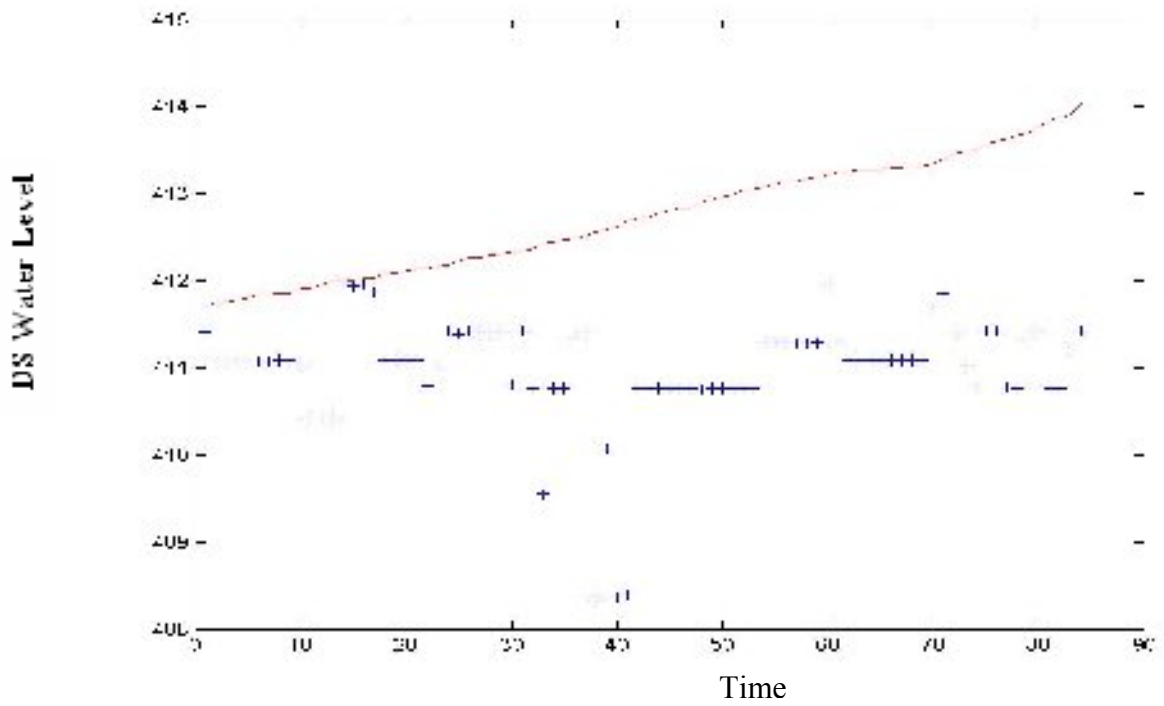


Figure (5.3): the performance of L-M for set 4

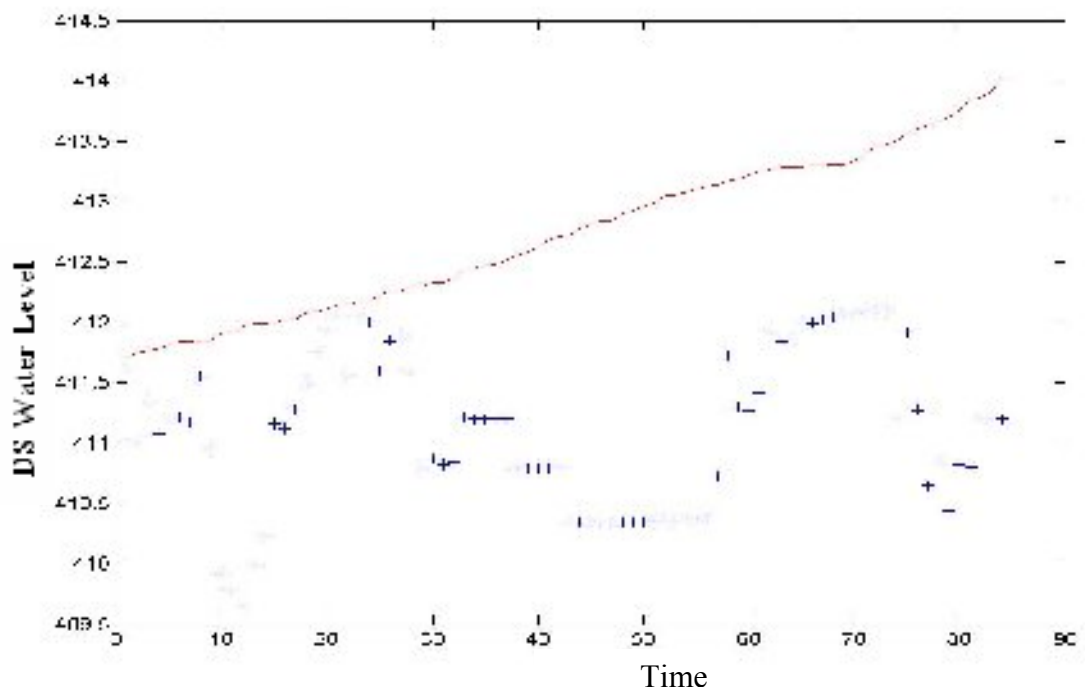


Figure (5.4): the performance of L-M for set 6

Table (5.3): MLP Training &amp; Testing values for number of neurons is 5

<b>BP</b>	<b>Number of neuron</b>	<b>epochs</b>	<b>Training Error</b>	<b>Testing Error</b>
<b>Set 1</b>	5	40000	0.78	0.8536
<b>Set 2</b>	5	40000	0.51	0.5908
<b>Set 3</b>	5	40000	0.32	0.631
<b>Set 4</b>	5	40000	0.364	0.502
<b>Set 5</b>	5	40000	0.181	0.2979
<b>Set 6</b>	5	40000	0.073	0.1213

As show in table(5.3), the best performance value for BP network (5-5-1) founded in set number 6 having the minimum error comparing with other sets equal to 0.1213, figures (5.4) and (5.5) shows the performance of BP for set 4 and set 6 respectively.

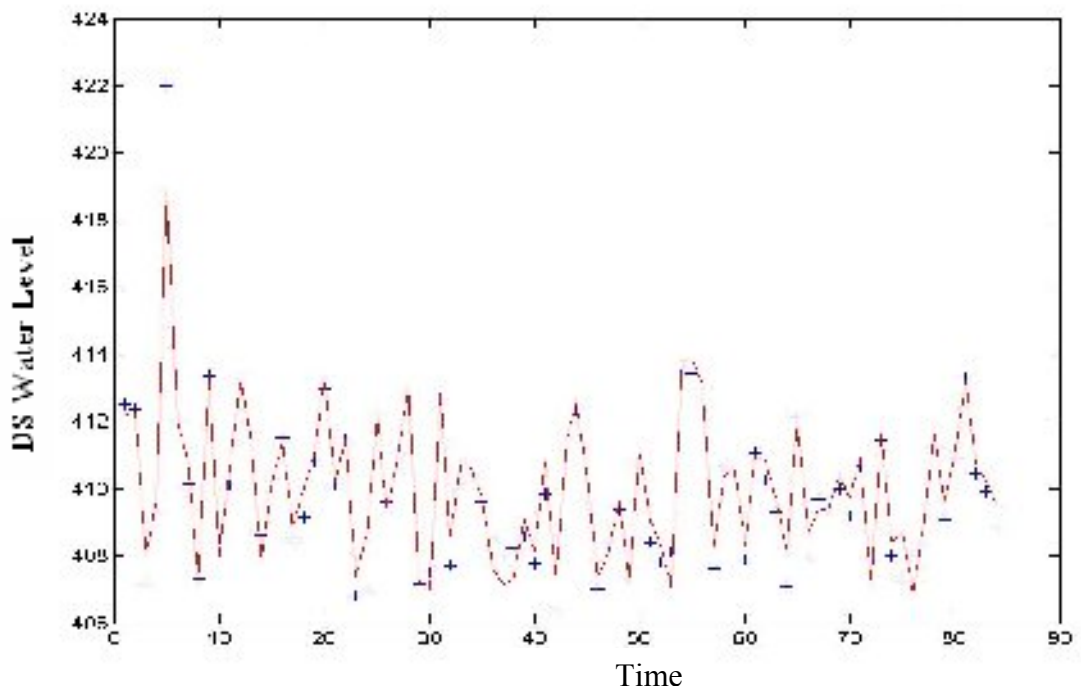


Figure (5.4): the performance of BP for set 4



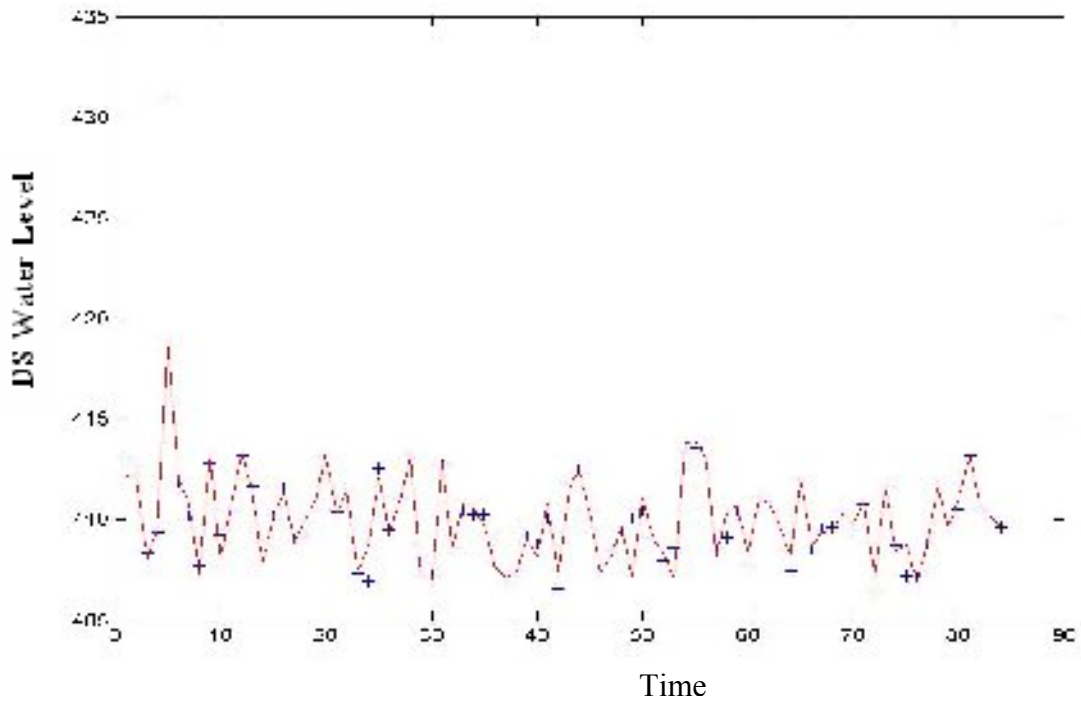


Figure (5.5): the performance of BP for set 6

We justify that GRNN performed so well for these reasons:

1. Choosing an optimal spread of 0.1 made the performance optimal (See table (5.2) ).
2. GRNN is performing better than the other algorithms rather than there is not many training set available.
3. There was a little redundancy vector in training data.

Table (5.4): GRNN Training & Testing values for Spread equal to 0.2

GRNN	Spread Value	Training Error	Testing Error
Set 1	0.2	0.015	6.1 e-005
Set 2	0.2	0.0132	0.011
Set 3	0.2	0.0065	5.61 e-23
Set 4	0.2	2.572 e-011	6.17 e-020
Set 5	0.2	0.648	0.0011
Set 6	0.2	0.0068	2.53 e-005

Table (5.5): GRNN Training &amp; Testing values for Spread equal to 0.5

<b>GRNN</b>	<b>Spread Value</b>	<b>Training Error</b>	<b>Testing Error</b>
<b>Set 1</b>	0.5	0.379	3.5 e-04
<b>Set 2</b>	0.5	0.0137	0.055
<b>Set 3</b>	0.5	0.0102	1.13 e-007
<b>Set 4</b>	0.5	5.2 e-004	8.34 e-007
<b>Set 5</b>	0.5	0.4067	0.105
<b>Set 6</b>	0.5	0.08	0.0034

Table (5.6): GRNN Training &amp; Testing values for Spread equal to 1

<b>GRNN</b>	<b>Spread Value</b>	<b>Training Error</b>	<b>Testing Error</b>
<b>Set 1</b>	1	0.2086	0.0014
<b>Set 2</b>	1	0.1258	0.13
<b>Set 3</b>	1	0.477	3.7 e-005
<b>Set 4</b>	1	0.126	4.7 e-005
<b>Set 5</b>	1	0.84	0.022
<b>Set 6</b>	1	0.42	0.018

Table (5.7): GRNN Training &amp; Testing values for Spread equal to 100

<b>GRNN</b>	<b>Spread Value</b>	<b>Training Error</b>	<b>Testing Error</b>
<b>Set 1</b>	100	1.93	0.066
<b>Set 2</b>	100	1.99	0.79
<b>Set 3</b>	100	1.939	0.066
<b>Set 4</b>	100	1.933	0.067
<b>Set 5</b>	100	2.003	0.07
<b>Set 6</b>	100	1.99	0.0796

As shown in table (5.4) through table (5.7) above, we notice that:

- when chose a spread of 0.1, it gave us performance closed to 100%, and every time the spread gets larger, the performance gets down.
- the performance of the GRNN was better than other NN algorithms in both training and testing stage.

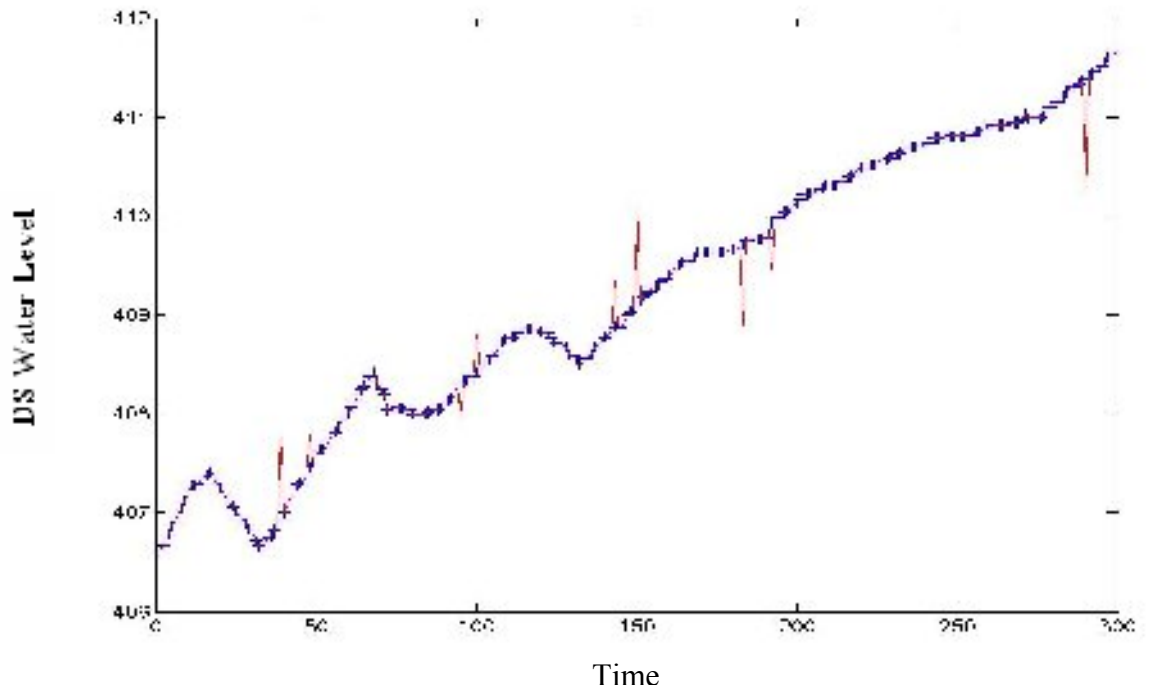


Figure (5.6): GRNN training with spread equal to 0.2 for set 1

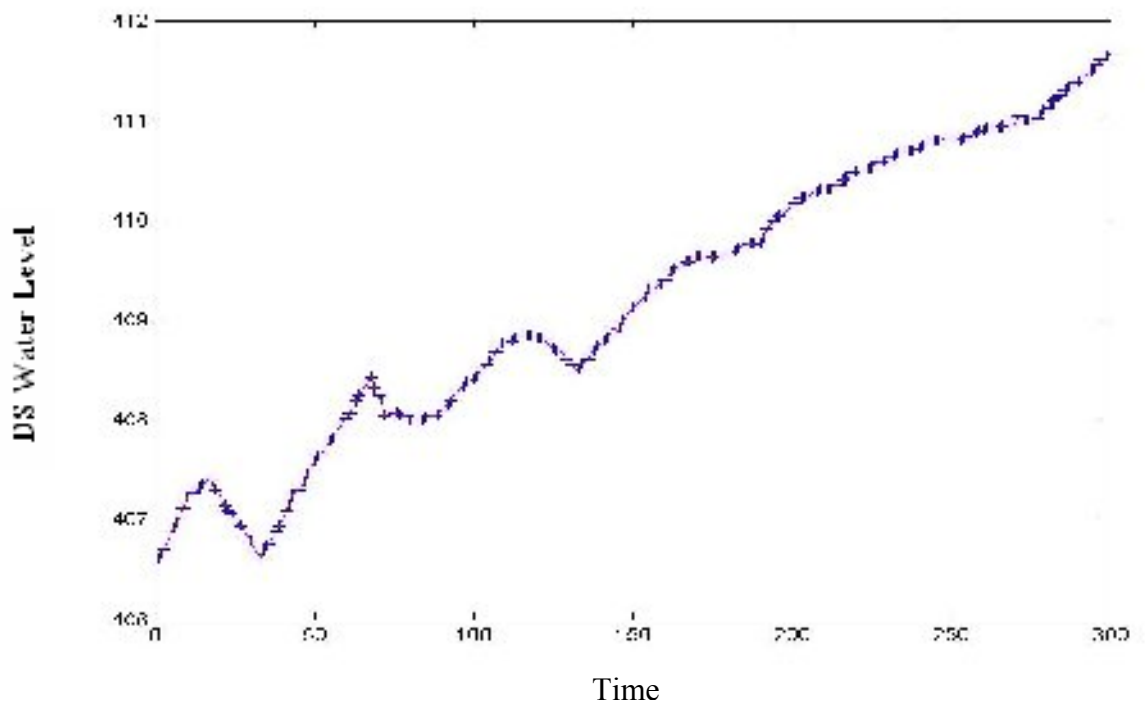


Figure (5.7): GRNN training with spread equal to 0.2 for set 4

From figures (5.6), ( 5.7 ) we can say that the performance of these network is better than the error performance of other network as we will show later in figures (5.8), (5.9), (5.10) and (5.11) when a spread values becomes rise.

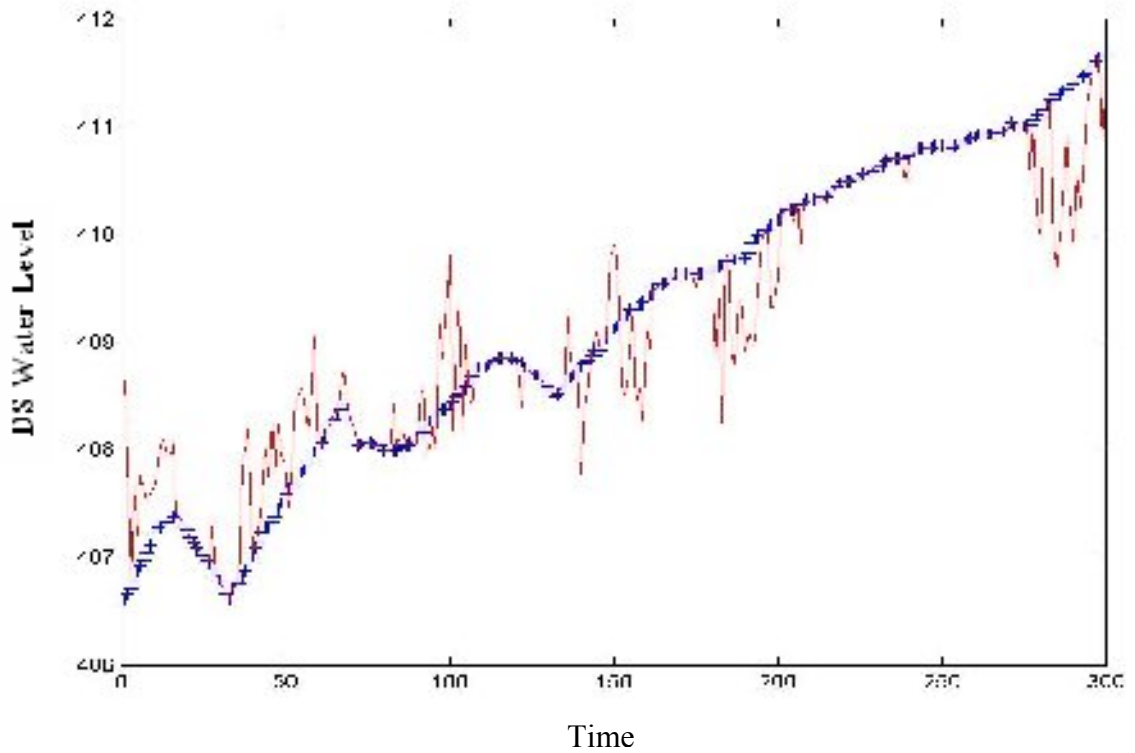


Figure (5.8): GRNN training with spread equal to 1 for set 1

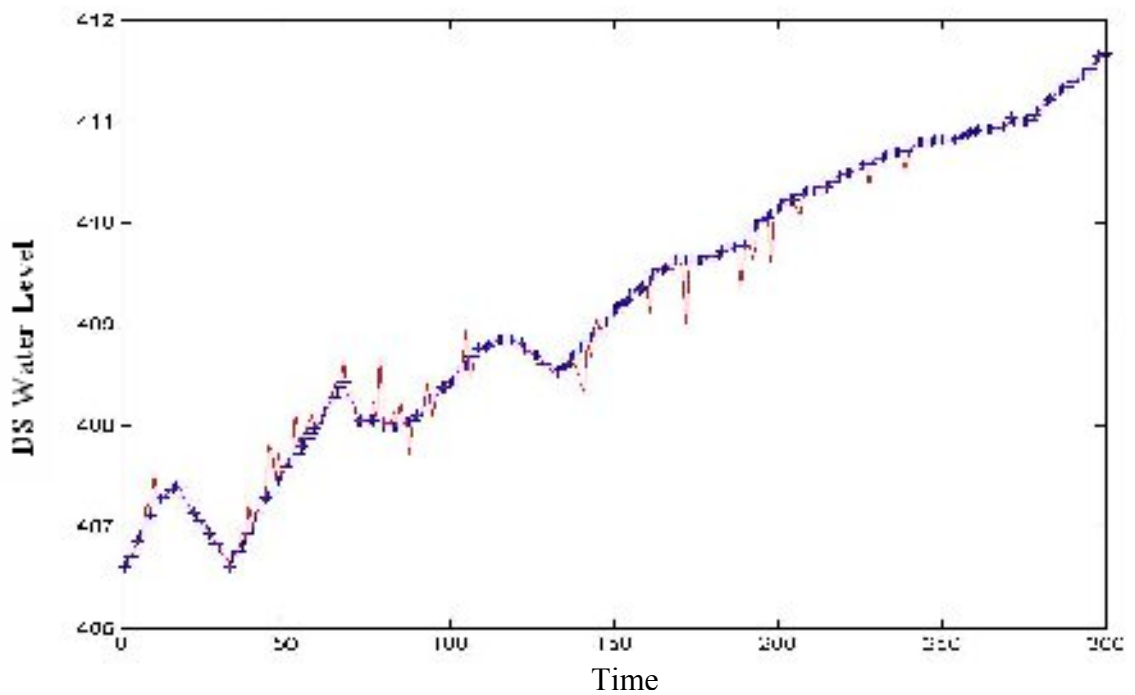


Figure (5.9): GRNN training with spread equal to 1 for set 4

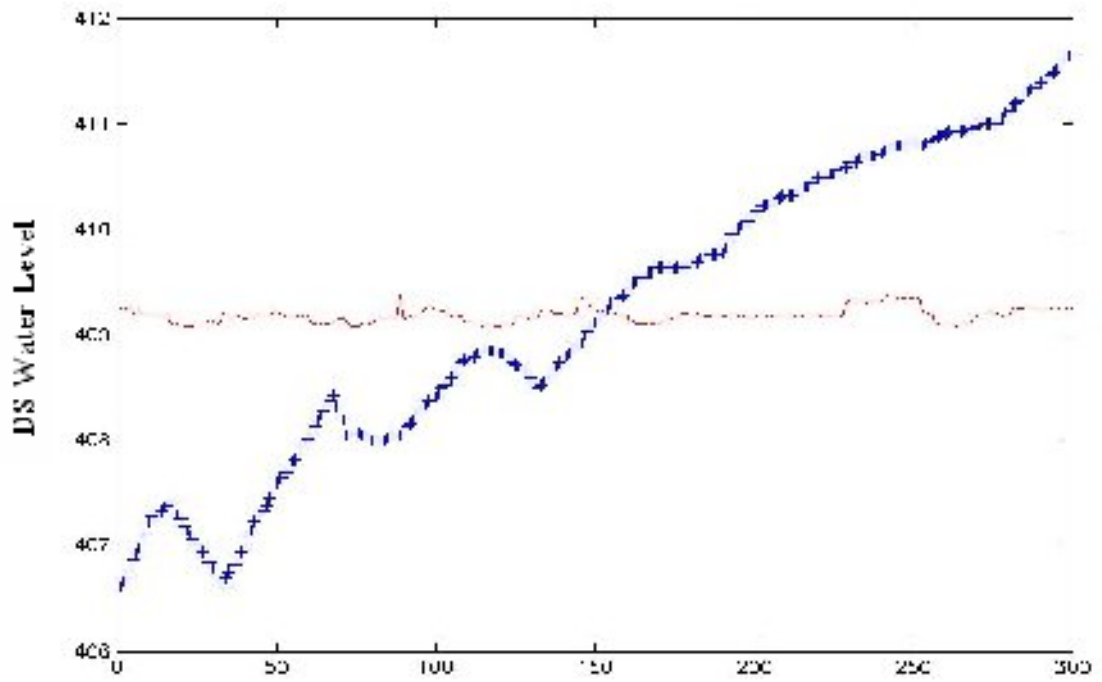


Figure (5.10): GRNN training with spread equal to 100 for set 1

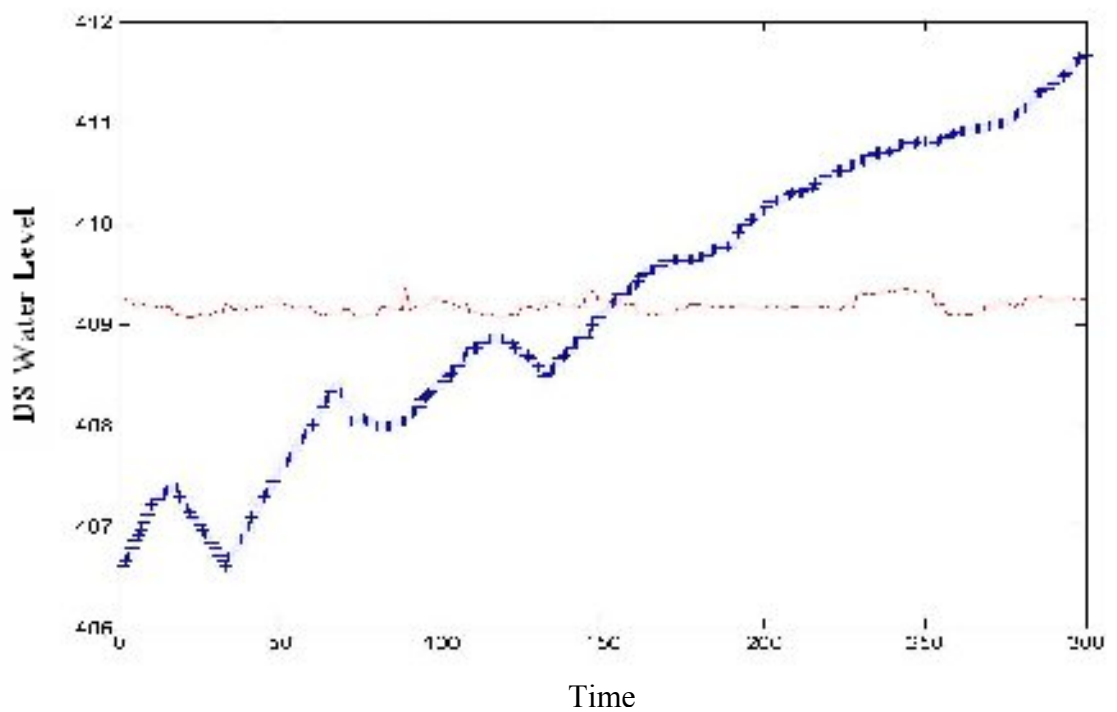


Figure (5.11): GRNN training with spread equal to 100 for set 4

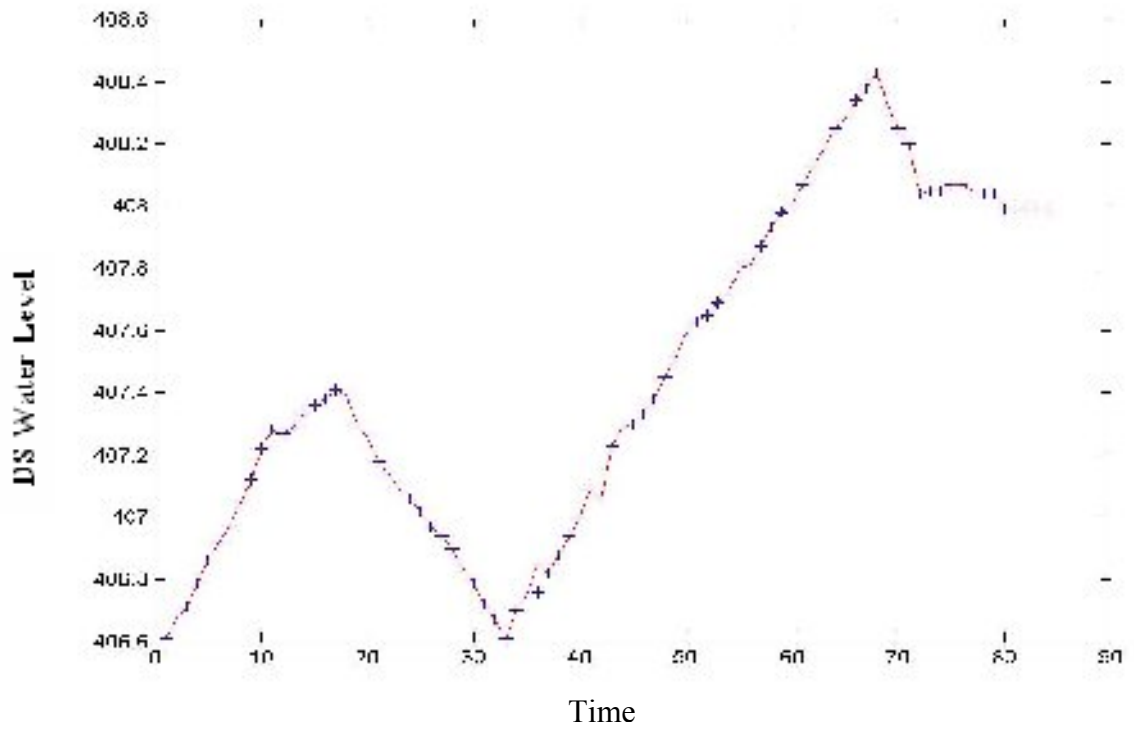


Figure (5.12): GRNN test with spread equal to 0.2 for set 1

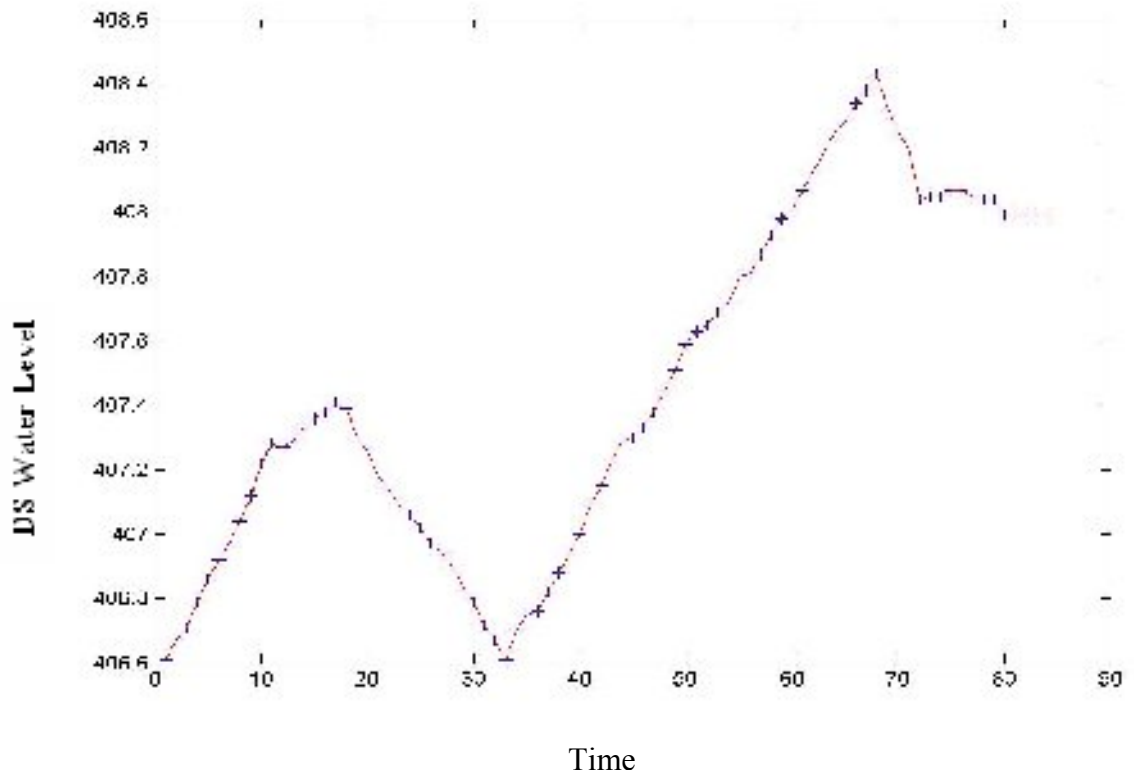


Figure (5.13): GRNN test with spread equal to 0.2 for set 4

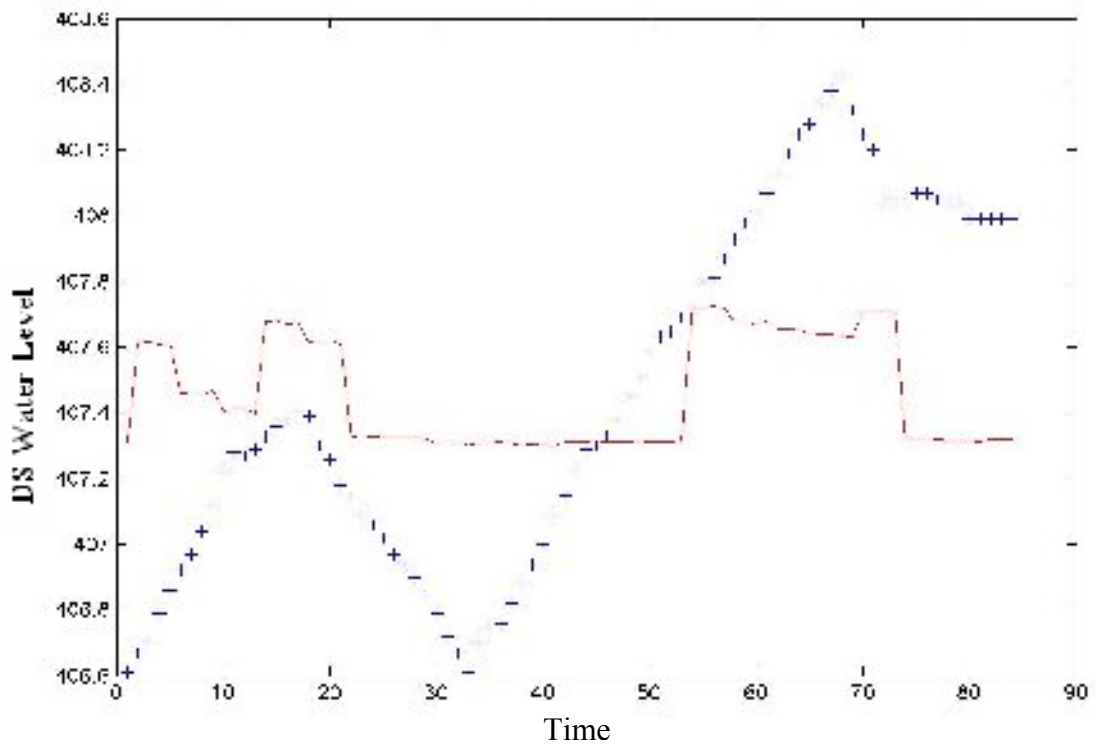


Figure (5.14): GRNN test with spread equal to 100 for set 1

As we discuss before, we conclude that as a spread value become rise, the performance of training and testing become decline, (continuous line mean tested value & the '+' means actual value).

Table(5.8): LM Training & Testing values with different number of neurons and 0.0005 goal

LM	Number of neuron	epochs	Training Error	Testing Error
Set 1	11	20000	1.996	3.09
Set 2	11	20000	1.120	1.39
Set 3	11	20000	1.593	2.965
Set 4	11	20000	1.841	1.268
Set 5	11	20000	1.859	1.659
Set 6	11	20000	1.996	1.062
Set 1	10	20000	2.69	3.97
Set 2	10	20000	1.99	2.46
Set 3	10	20000	2.63	3.16
Set 4	10	20000	2.45	3.87
Set 5	10	20000	3.08	3.91
Set 6	10	20000	4.36	5.026
Set 1	9	20000	3.9	4.28
Set 2	9	20000	2.75	3.36
Set 3	9	20000	4.05	5.91

<b>Set 4</b>	9	20000	5.37	5.94
<b>Set 5</b>	9	20000	6.97	8.598
<b>Set 6</b>	9	20000	4.258	7.147

Table(5.9): BP Training &amp; Testing values with 6 neurons and 0.0005 goal

<b>BP</b>	<b>Number of neuron</b>	<b>epochs</b>	<b>Training Error</b>	<b>Testing Error</b>
<b>Set 1</b>	6	40000	1.908	2.9954
<b>Set 2</b>	6	40000	0.9768	1.1292
<b>Set 3</b>	6	40000	1.0053	2.7677
<b>Set 4</b>	6	40000	1.6420	1.0606
<b>Set 5</b>	6	40000	1.7023	1.1039
<b>Set 6</b>	6	40000	1.8888	0.9227

Table(5.10): BP Training &amp; Testing values with 5 neurons and 0.0005 goal

<b>BP</b>	<b>Number of neuron</b>	<b>epochs</b>	<b>Training Error</b>	<b>Testing Error</b>
<b>Set 1</b>	5	40000	0.78	0.8536
<b>Set 2</b>	5	40000	0.51	0.5908
<b>Set 3</b>	5	40000	0.32	0.631
<b>Set 4</b>	5	40000	0.364	0.502
<b>Set 5</b>	5	40000	0.181	0.2979
<b>Set 6</b>	5	40000	0.073	0.1213

Table(5.11): BP Training &amp; Testing values with 4 neurons and 0.0005 goal

<b>BP</b>	<b>Number of neuron</b>	<b>epochs</b>	<b>Training Error</b>	<b>Testing Error</b>
<b>Set 1</b>	4	10000	0.86	0.931
<b>Set 2</b>	4	10000	0.947	1.0887
<b>Set 3</b>	4	10000	0.742	0.9371
<b>Set 4</b>	4	10000	0.699	0.802
<b>Set 5</b>	4	10000	0.898	0.914
<b>Set 6</b>	4	10000	0.108	0.213

- Tables (5.8), (5.9), (5.10) and (5.11) shows how the BP & L-M Testing values with respect to the numbers of neurons and its goal which is set to 0.0005. The performance of BB become down as number of BP network neurons is more than 5 and less than 4 of goal 0.0005 and variance numbers of epochs. On the



other hand the performance of L-M networks become down for any number of neurons except 11 neurons.

Finally, figures (5.15), (5.16), ..., (5.20) shows the error performance for some test BP and L-M for some datasets, the figures plotted the targets and output vectors for DS water level.

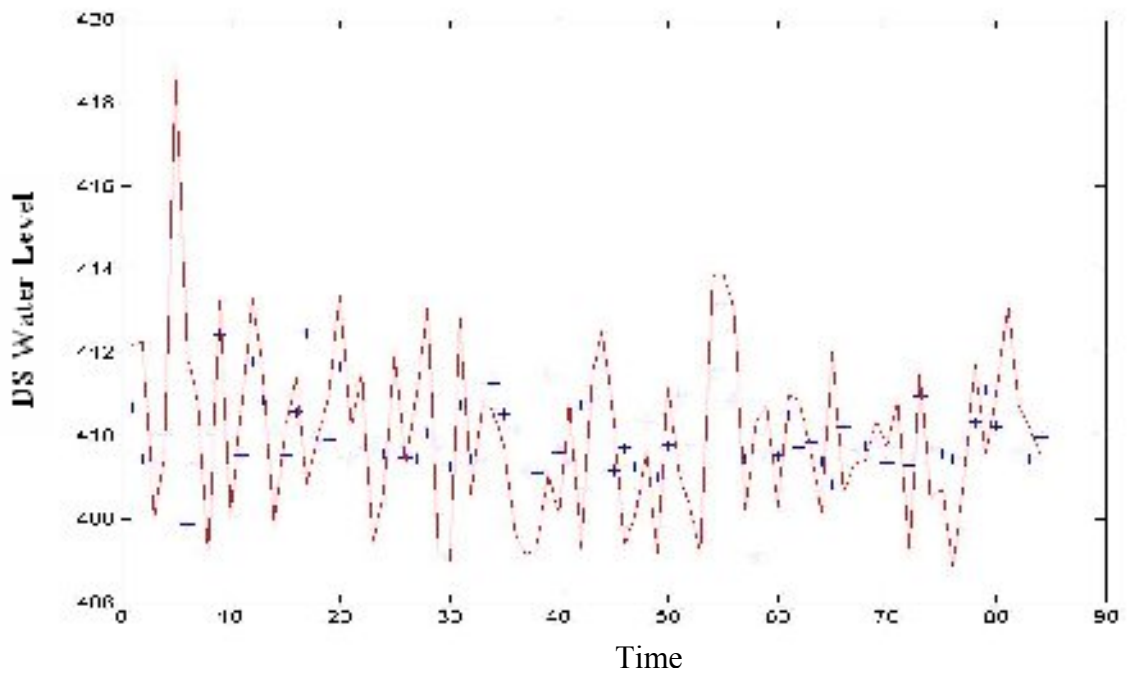


Figure (5.15) BP test set for the set number 1

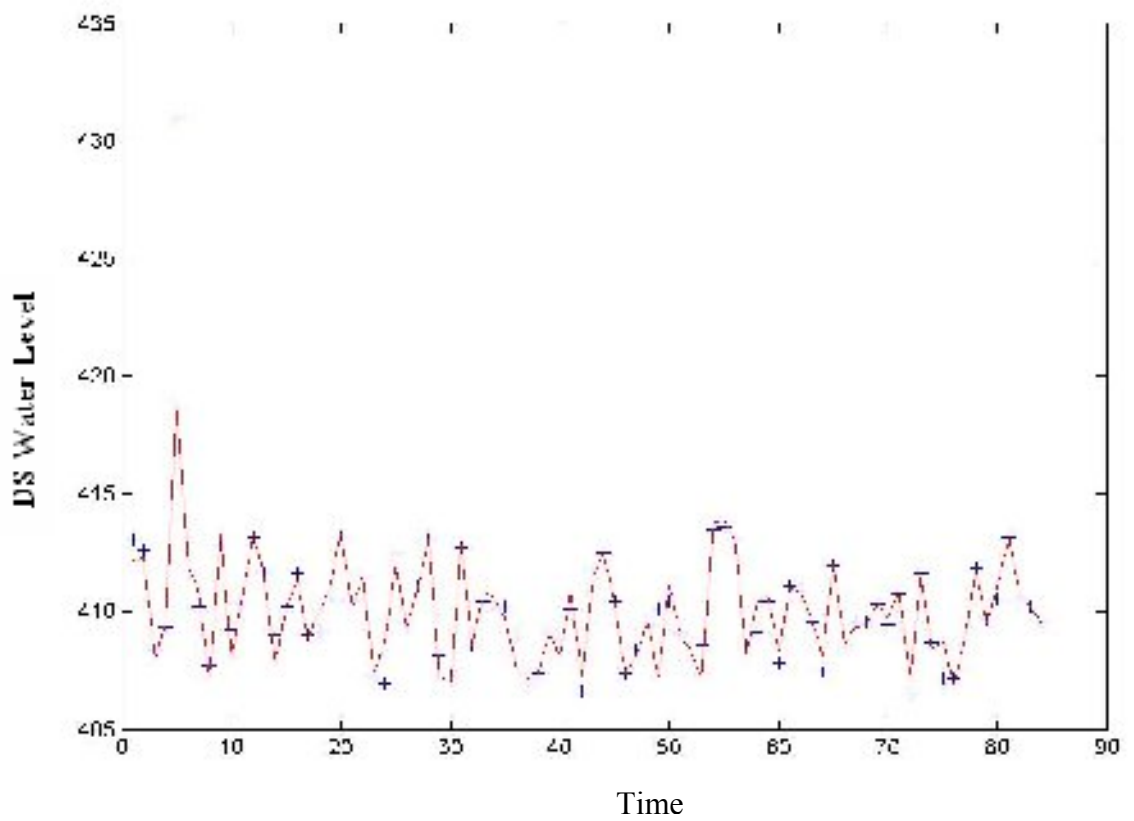


Figure (5.16) BP test set for the set number 4

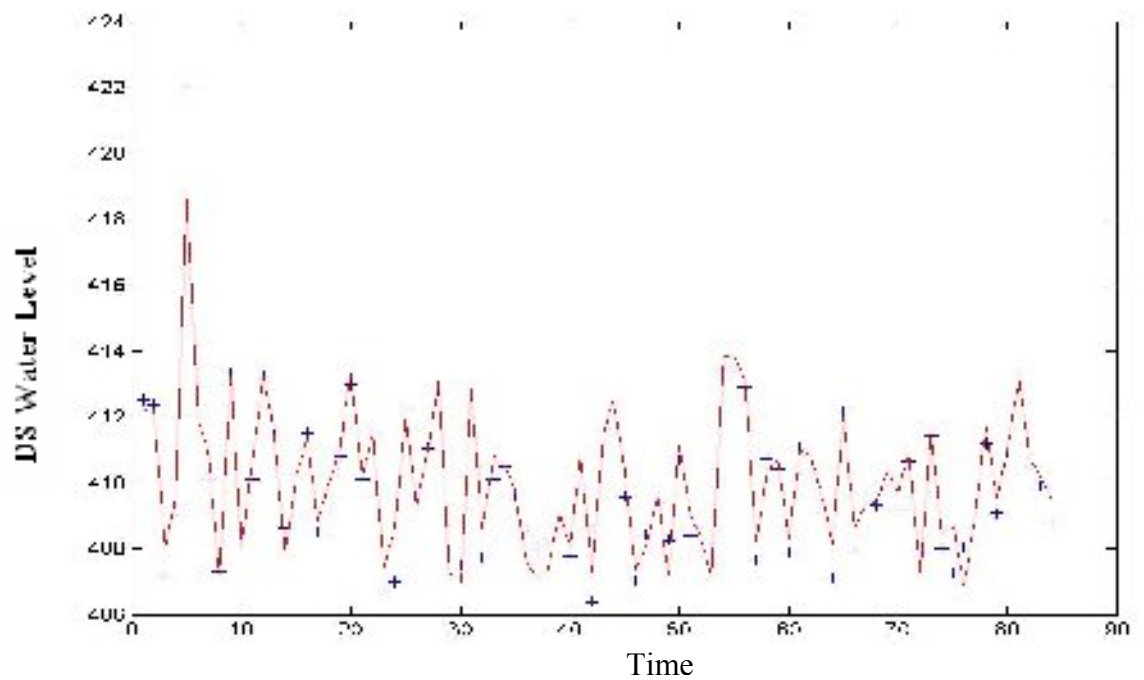


Figure (5.17) BP test set for the set number 6

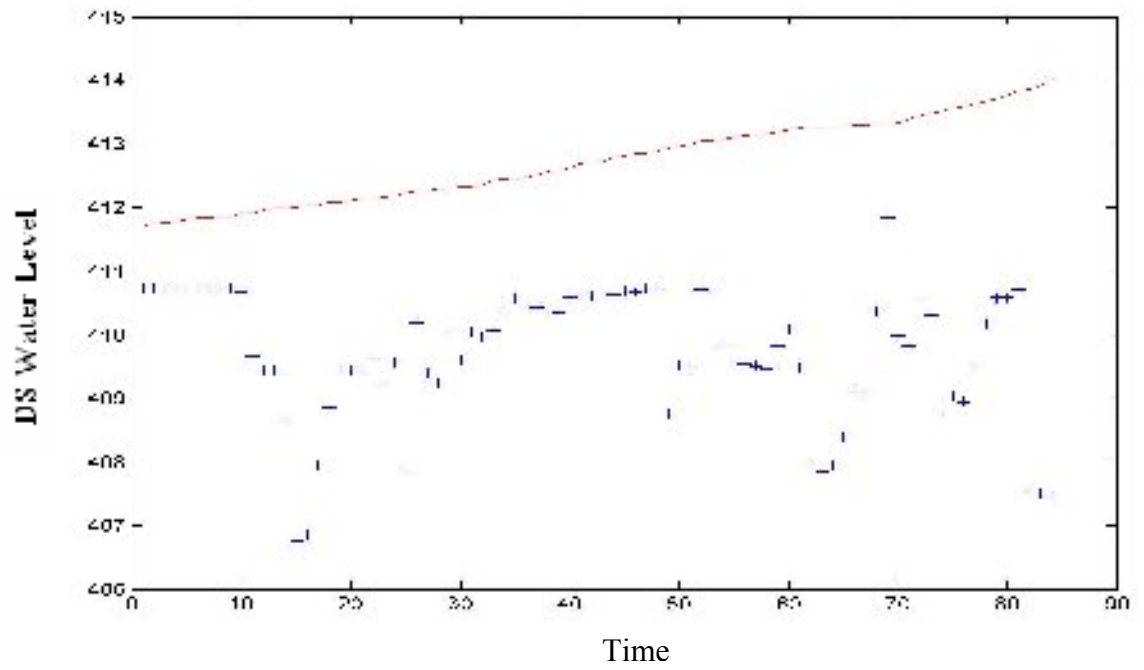


Figure (5.18) LM test set for the set number 1

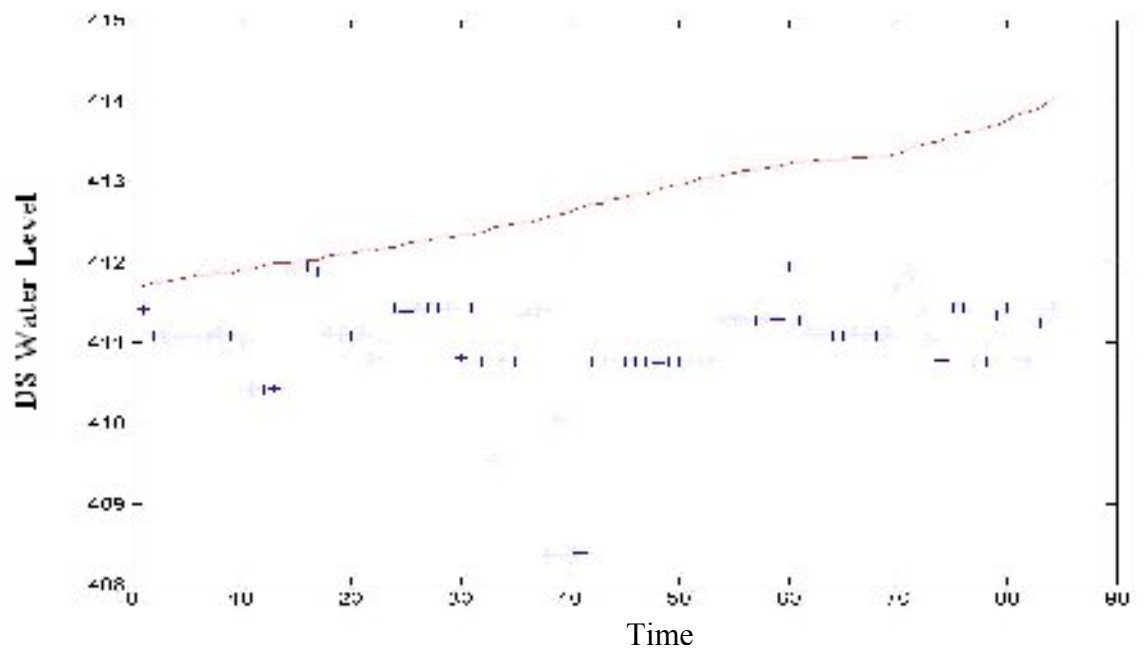


Figure (5.19) LM test set for the set number 4

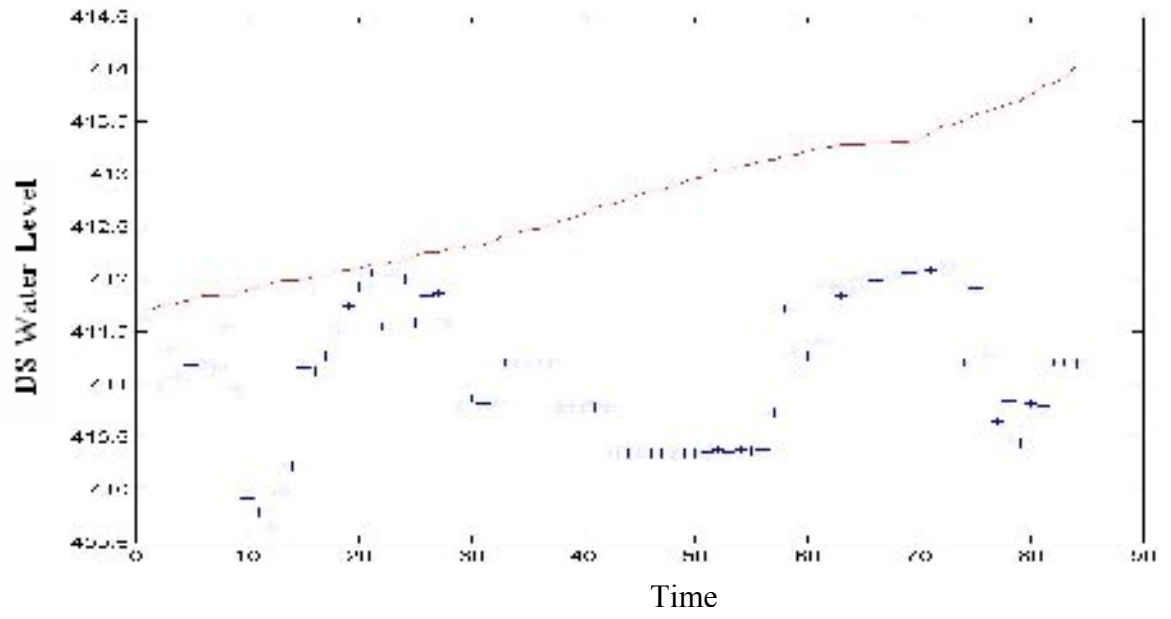


Figure (5.20) LM test set for the set number 6

# CHAPTER 6

## CONCLUSION AND FUTURE WORK

### 6.1 Conclusion

The main objective of this study is to model the Dead Sea geological parameters relationship because of the limitations of studies concerning of the Dead Sea geological nature. In fact, there is no agreement about the main factors affected by the Dead Sea, this study illustrates the way which may help the geologists of the Dead sea to improve their studies and also researchers in the future.

In this thesis we compared three NN algorithms, two of them are the learning algorithms that train the MLP: Back-Propagation (BP) and Levenberg-Marquardt (L-M), the third one is the General Regression Neural Network (GRNN) algorithms. Results have shown that we obtain a lower error for GRNN model than both of BP and L-M models. This result is illustrated by the comparison of performances of the three algorithms in order to identify the best NN model to be applied to the prediction of the DS water level.

These NN structures applied to Dead Sea data for six different sets, each set have different five parameters. When comparing BP, L-M and GRNN models, we find that GRNN is better than BP and L-M in training and also in testing stage.

Recall that the performance of GRNN was better than other for all spread values tried and the performance of BP for training and testing is more accurate than the performance results gained by L-M.

Choosing the number of hidden neurons and spread values in this study was based on trial-and error. GRNN training algorithm gave the best result for the training data, the most important result should be considered with the test data is  $1.01e-025$ .

Unfortunately, we still have some limitations and drawbacks in this study, we determined the optimal parameters (especially, spread and number of neurons) more

or less by trial-and-error. A more sophisticated method to determine the optimal parameter values is desirable and it improves the performance considerably.

We conclude that the result is quite satisfactory and it reflects the nature of given algorithms, BP, L-M and GRNN. Results showed the advantages and disadvantages of these algorithms. Although results would still require improvements by adapting automatic techniques choosing optimal parameters, applying the neural network models to external data set.

## **6.2 Recommendations for Future work**

It has been shown in this study that the NN models are capable to model the complex relationship between Dead Sea and its factors. As part of Artificial Intelligent groups, the Fuzzy Logic (FL) and Genetic Algorithm (GA) models have a good characteristics and capability to model such relationship. FL and GA models can be designed through experience of the experts. Therefore, for further research the study of these algorithms should be carried out. Several studies has found that the combination of two methods are more powerful and effectiveness for many application.

In this study, the training of NN algorithms was accomplished using the same transfer function for all neurons and layers in a network. The use of different transfer functions for each layer will be helpful to obtain better accuracy results.

## REFERENCES

1. A. Sezin Tokar and Momcilo Markus, "Precipitation-Runoff Modeling Using Artificial Neural Networks and Conceptual Models" J. Hydrologic Engrg, Volume 5, Issue 2, pp. 156-161 (April 2000).
2. A. Sezin Tokar and Peggy A. Johnson, "Rainfall-Runoff Modeling Using Artificial Neural Networks", J. Hydrologic Engrg., Volume 4, Issue 3, pp. 232-239 (July 1999)
3. Abdelaziz L. AL-Khlaifat, "Dead Sea Rate of Evaporation", American Journal of Applied Sciences 5 (8): 934-942, 2008
4. ASCE (2000). Artificial Neural Networks in Hydrology.II: Hydrologic Applications, J. Hydrologic Engrg., Volume 5, Issue 2, pp. 124-137 (April 2000) Rao S. Govindaraju, Assoc. Prof., Purdue Univ., School of Civ. Engrg., 1284 Civ. Engrg. Build., West Lafayette, IN 47907-1284.
5. ASCE (2000). Artificial Neural Networks In Hydrology, Part I: Preliminary Concepts. Journal of Hydrology Engineering. 2. 115-123.
6. Asefa T (Asefa, Tirusew), Wanakule N (Wanakule, Nisai), Adams A (Adams, Alison), "Field-Scale Application Of Three Types Of Neural Networks To Predict Ground-Water Levels", Journal Of The American Water Resources Association 43 (5): 1245-1256 Oct 2007.
7. Ardicioglu M (Ardicioglu, Mehmet), Kisi O (Kisi, Ozgur), Haktanir T (Haktanir, Tefaruk). "Suspended Sediment Prediction Using Two Different Feed-Forward Back-Propagation Algorithms", Canadian Journal Of Civil Engineering 34 (1): 120-125 JAN 2007.
8. B. N. Asmar and Peter Ergenzinger, "Estimation of evaporation from the Dead Sea", Hydrological Processes Hydrol. Process. 13, 2743±2750 (1999).
9. B.N.Asmar, Peter erezinger, "Effective Of The Dead Sea-Read Sea Canal Modeling On The Prediction Of The Ds Conditions", Hydro 17,1607-1621, 2003.
10. Calder, I. R. and Neal,C. "Evaporation from Saline Lakes: a Combination Equation Approach", Hydrol. Sci. J., 29, 89±97. 1984.
11. Clive Lipchin, "A Future for the Dead Sea Basin", Social Science Research Network Electronic Paper Collection: Year 2006 Paper 97.
12. Fausett, L. (1994). "Fundamentals of Neural Networks". New Jersey: Prentice Hall, Englewood Cliffs.
13. [ftp://ftp.sas.com/pub/neural/FAQ.html#A\\_cando](ftp://ftp.sas.com/pub/neural/FAQ.html#A_cando), accessed 2008.

14. Heimes, F. and Heuveln, B. V. (1998). "The Normalized Radial Basis Function Neural Network". IEEE Transactions on Neural Networks. Vol. 1. 1609-1614
15. Hsu, K. Gupta, H.V. and Sorooshian, S. (1995). "Artificial Neural Network Modelling of the Rainfall-Runoff Proces"s. Water Resources Research. Vol. 31(10). 2517-2530.
16. <http://www.isramar.ocean.org.il>, accessed 2008.
17. <http://www.cs.waikato.ac.nz/ml/weka>, accessed 2008.
18. <http://www.usingneuralnetworks.com>, accessed 2008.
19. <http://www.neuralnetwork.com>, accessed 2008.
20. Gertman, A. Hecht " The Dead Sea Hydrography from 1992 to 2000", Journal of Marine Systems 35 (2002) 169– 181.
21. I.M.Oroud, "Evaluation Of Saturation Vapor Pressure Over Hypersaline Edge Of The Dead Sea, Jordan", solar energe vol.53, no 6, pp 497-503, 1994.
22. Isaac Gertman, Artur Hecht, Alexey Murashkovsky, Nadav Lensky, "Hydrometeorological Monitoring of the Dead Sea", Geophysical Research, Vol. 8, 07295, 2006.
23. J.Irrig. and Drain. Engrg, "Forecasting of Reference Evapotranspiration by Artificial Neural Networks", Volume 129, Issue 6, pp. 454-457 (November/December 2003).
24. Kisi O (Kisi, Ozgur), Cigizoglu HK (Cigizoglu, H. Kerem), "Comparison Of Different Ann Techniques In River Flow Prediction" , Civil Engineering And Environmental Systems 24 (3): 211-231 2007.
25. M. Abelson, Y. Yechieli, A. Bein, O. Crouvi, V. Shtivelman, G.Baer, "Sinkhole Swarms Along The Dead Sea Coast", Geophysical Research Abstracts, Vol. 9, 05191, 2007.
26. Maier, H.R. and Dandy, G.C. (1996). "The Use of Artificial Neural Networks for the Prediction of Water Quality Parameters". Water Resources Research. Vol. 32(4).
27. Marwan A.Hassann, Michaklein, "Flurial Adjustment Of The Lower Jordan River To Drop In The Dead Sea Level ", Elsevier science 21-33, 2002.
28. Mero, F. and Simon, E. 1985. "A Daily Simulation Model For Evaluation Future Dead Sea Levels", in Scientic Basis for Water Resources Management (Proceedings of the Jerusalem Symposium September 1985), Diskin, M. (Ed.), IAHS Publ. 153, 265±276.



29. M. Erol Keskin and Özlem Terzi , " Artificial Neural Network Models of Daily Pan Evaporation", J. Hydrologic Engrg., Volume 11, Issue 1, pp. 65-70 (January/February 2006).
30. MKousa Mohsen, "Water Strategies And Potential Of Desalination In Jordan", science direct, Desalination 2003 ,pages 27-46(2007).
31. N. G. Lensky, Y. Dvorkin, and V. Lyakhovsky, "Water, Salt, And Energy Balances Of The Dead Sea", Water Resources Research, vol. 41, W12418, doi:10.1029/2005WR004084, 2005.
32. N. S. Raghuwanshi, R. Singh, and L. S. Reddy, "Runoff and Sediment Yield Modeling Using Artificial Neural Networks", Upper Siwane River, India J. Hydrologic Engrg., Volume 11, Issue 1, pp. 71-79 (January/February 2006).
33. N. J. de Vos<sup>1</sup> and T. H. M. Rientjes, "Constraints Of Artificial Neural Networks For Rainfall-Runoff Modelling: Trade-Offs In Hydrological State Representation And Model Evaluation", Hydrology and Earth System Sciences, 9, 111–126, 2005.
34. Nisai Wanakule, M.ASCE and Alaa Aly, "Computing in Civil Engineering 2005 Artificial Neural Networks for Forecasting Groundwater Levels", July 12–15, 2005.
35. Nizar Abu-Jaber, "A New Look At The Chemical And Hydrological; Evolution Of The Dead Sea", Geochimica et Cosmochimica Acta, vol.62, no.9, pp.1471-1479,1998.
36. Nizar Abu-Jaber, "Geochemical Modeling of the Effects of the Proposed Red Sea-Dead Sea Canal", ABHATH AL-YARMOUK: "Basic Sci. & Eng." Vol 13,no.2,pp.269-281, 2004.
37. O. Makarynsky, D. Makarynska, M. Kuhn, W.E. Featherstone, "Predicting Sea Level Variations With Artificial Neural Networks At Hillarys Boat Harbour, Western Australia", [www.elsevier.com/locate/ECSS](http://www.elsevier.com/locate/ECSS) Estuarine, Coastal and Shelf Science 61 (2004) 351–360.
38. Oroud, I. M. "Effect Of Salinity Upon Evaporation From Pans And Shallow Lakes Near The Dead Sea", Theor. Appl. Climatol., 52,231±240. 1995.
39. Radwan A. Al-Weshah , "The Water Balance Of The Dead Sea:An Integrated Approach", Hydrol. Process. 14, 145±154 (2000).
40. Rudolf Orthofer , Clive Lipchin, "The Dead Sea Project: Is a More Sustainable Water Management in the Dead Sea Basin Possible?", MEDAQUA Conference, Amman, 14-15 Jun 2004.
41. Salameh and H. El-Naser, " Does the Actual Drop in Dead Sea Level Reflect the Development of Water Sources Within its Drainage Basin?", WILEY-VCH Verlag GmbH, D-69451 Weinheim, 1999.

42. Specht, D. F. (1991). "A General Regression Neural Network". IEEE Transactions on Neural Networks. Vol. 2. 568-576.
43. Tingsanchali, T. (2000). "Forecasting Model of Chao Phraya River Flood Levels at Bangkok". Thailand: Research Report, Asian Institute of Technology.
44. Tokar, A. S. (1996). "Rainfall-Runoff Modelling in an Uncertain Environment". University of Maryland: Ph.D. Dissertation.
45. Tsoukalas, L. H. and Uhrig, R. E. "Fuzzy and Neural Approaches in Engineering". New York: John Wiley & Sons Inc. (1997).
46. Yoseph Yechieli, Ittai Gavrieli, Brian Berkowitz, Danial Roner, "Will the DS Die", Geology, August 1998, v.26; no 8 , p755-758.

## الخلاصة

إن حوض البحر الميت يلعب دوراً مهماً في التطور الاقتصادي والصناعي والزراعي في الأردن. أبحاث عديدة أشارت إلى أن مستوى المياه في البحر الميت انخفض منذ تم بنسبة 30% لكل سنة. وبناءً على ذلك هناك حاجة ماسة لتزويد الباحثين والجيولوجيين بتقديرات عن مستوى المياه في البحر الميت بطريقة دقيقة من خلال أبحاث تعطي نتائج فعالة لكي يكونوا قادرين على فهم التغيرات التي تطرأ على مستوى المياه في البحر الميت من أجل العمل على إيقاف الانخفاض المستمر لمستوى المياه في البحر الميت.

الشبكات العصبونية (Neural Network) هي عبارة عن نموذج رياضي لديه القدرة على التعلم وتنظيم البيانات بشكل متوازٍ ومن أكثر نماذج الشبكات العصبونية انتشاراً (GRNN, L-M) وBack Propation) لديها القدرة على تمثيل معادلات غير خطية مما يسهل دخولها والمخرجات.

من المعروف أن مستوى سطح البحر يعتمد على مجموعه من العوامل منها حرارة الهواء ودرجة الملوحة ودرجة الرطوبة ودرجة الحرارة من العوامل المؤثرة على البيئته. ودج المتبع في هذه الدراسة يعتمد بأخذ مجموعات مختلفة من هذه العوامل لتحديد أيها الأكثر تأثيراً وفاعلية على تحديد مستوى المياه في البحر الميت ومن ثم استخدامها كل مجموعة بشكل منفصل وتطبيقها على أكثر من نموذج للشبكات العصبونية للتنبؤ بمستوى سطح البحر ولتحديد أي مجموعة لها التأثير الأكبر على مستوى سطح البحر الميت.

أخيراً يمكن القول أن النموذج المقترح (GRNN) يطي أفضل نتائج مقارنة مع النمادج الأخرى من الشبكات العصبونية من خلال استخدام متوسط الخطأ التربيعي (MSE).

التنبؤ بمستوى سطح البحر الميت باستخدام  
الشبكات العصبونية

من قبل

"محمد خالد" يوسف محمد شمبور

باشرف

د. رشيد الزبيدي

قدمت هذه الرسالة استكمالاً لمتطلبات  
الحصول على درجة الماجستير في علم الحاسوب

عمادة البحث العلمي و الدراسات العليا  
جامعة فيلادلفيا

نيسان، ٢٠٠٨