DISSERTATION

SOLVING THE INVERSE PROBLEM OF CONTAMINANT TRANSPORT EQUATION USING A NEURAL NETWORK

Submitted by

Mohammad Ali Al-Murad

Department of Earth Resources

In partial fulfillment of the requirements

for the degree of Doctor of Philosophy

Colorado State University

Fort Collins Colorado

Fall 2002

COLORADO STATE UNIVERSITY

September 10, 2002

WE HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER OUR SUPERVISION BY MOHAMMAD ALI AL-MURAD ENTITLED SOLVING THE INVERSE PROBLEM OF CONTAMINANT TRANSPORT EQUATION USING A NEURAL NETWORK BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate Work

Adviser

Department Head/Director

ABSTRACT OF DISSERTATION

SOLVING THE INVERSE PROBLEM OF CONTAMINANT TRANSPORT EQUATION USING A NEURAL NETWORK

The artificial neural network is considered to be a universal function approximater. As such, the artificial neural network can be used to solve the inverse contaminant transport equation for parameter estimation. In this study the plausibility of the artificial neural network in solving an inverse problem related to values of longitudinal dispersivities using concentration values derived from a forward model at a fixed time is assessed.

A set of published data corresponding to the Macrodispersivity Experiment (MADE) study at the Columbus Air Force Base (CAFB) in northeastern Mississippi was used as the base to build a two-dimensional, unconfined, homogenous, synthetic case to investigate the applicability of this method under a variety of scenarios.

In this approach, the dispersivity values were estimated by training a feedforward backpropagation network of 20 neurons in one hidden layer, by using concentration fields generated by the MT3D model as an inputs and the dispersivity values were the outputs. The network was able to estimate the dispersivity values with high accuracy using 1600 concentration measurements for each dispersivity value. The total number of dispersivity values was 210, which was divided into four subsets for the propose of training and estimation.

The network was also investigated with different grid size of the domain, different number of training patterns, and different number of concentration measurements. The

method demonstrated a good accuracy in estimating the dispersivity values to a certain limit of grid size. Also it showed that the parameter can be estimated with a good accuracy using at least 147 concentration fields for different grid size and varying concentration measurements. The network demonstrated some limitation in estimating the dispersivity values with low dispersivity values at some grid sizes. One of the limitation of artificial neural networks is that they cannot reveal any physics and explain in a meaningful way the basic process by which they arrive at a decision. However, the artificial neural networks can be a good tool in evaluating the products of different forward models.

The network was able to generalize its behavior in estimating the parameter beyond the training range and with a different discretization level of the domain. The inverse solution was stable with different percentages of random errors added to the concentration fields needed to estimate their dispersivity values. However, the inverse solution was ill-posed due to non-uniqueness.

Mohammad Ali Al-Murad
Dept. of Earth Resources
Colorado State University
Fort Collins, CO, 80523
Fall, 2002

ACKNOWLEDGEMENTS

First of all, the author would like to thank God for everything. The author gratefully acknowledges Dr. William Sanford, Department of Earth Resources, Colorado State University, for serving as his major advisor, and providing careful guidance and support throughout his study. Thanks are also extended to committee members Dr. John Stednik, Department of Earth Resources, Dr. Freeman Smith, Department of Earth Resources, and Dr. James Warner, Department of Civil Engineering for their consultation and guidance.

The author extends his thanks to Kuwait Institute for Scientific Research officials for providing this scholarship.

And last but far not least the author is especially appreciative to his parents Mr. Ali Al-Murad, and Mrs. Fatima Al-Kandari, for their unfailing love and prayer to complete this degree program. Additionally, the author would like to extend his great thanks to his wife Jehad Al-Kandari for her support and understanding during his Ph.D. study. Also thanks are extended to his sons Ahmad, Bader, and Yousef, and to his daughters Maryam, Manar, and Mai.

DEDICATION

To

my parents Mr. Ali Ahmad Al-Murad, and Mrs. Fatima Ahmad Al-Kandari

my wife Jehad Hasan Al-Kandari

my sons Ahmad, Bader, and Yousef

my daughters Maryam, Manar, and Mai.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Introduction

In the past few decades, computational models of groundwater and solute transport have developed into powerful tools for studying groundwater as a resource and the effects of human activities on this resource. In groundwater transport, the models are mainly based on partial differential equations, which can be solved analytically, semi-analytically or numerically (Javandel et al., 1984).

Numerical models require a number of parameters, such as hydraulic conductivity and dispersivity characterizing the physical system. It is well known that the reliability of a model as a prediction tool depends on the accuracy of the parameters. Usually that accuracy is confounded by many factors, such as the difficulty of measuring the parameters in the field in terms of time and cost, error associated with the measurements (Yeh, 1986), spatial variability of the hydraulic parameters or heterogeneity of the domain, scale of the measurements (Barth et. al., 2001), and the parameter structure (Sun and Yeh, 1985; Hyun and Lee, 1998). Due to such confounding factors, the determination of the parameters is one of the difficulties associated with using these models (Wagner, 1992).

An alternative of the direct models is to solve an appropriate inverse problem to indirectly determine the needed parameters (Cannon et al., 1993). Considerable attention has been directed toward solving inverse problems in groundwater flow and more recently, in groundwater solute transport. Generic inverse models identify unknown parameters, which have physical significance but are difficult to measure, by using observations obtained by different techniques. Difficulties are usually encountered in solving inverse problems because they are classified mathematically as ill-posed problems due to their non-existence, non-uniqueness, and instability (Sun, 1994). However, new developments in applied mathematics and other fields have helped in providing hydrologists with many methods to overcome these difficulties. Artificial neural networks are relatively new computing devices that are finding applications in almost all branches of science and engineering (Stauffer, 2001). This study investigates the use of neural networks as an inversion method for groundwater transport parameters.

1.2 Defining the Inverse Problem

In general, inverse problems are contrary to forward or direct problems. The forward problem is defined as "the process of predicting the results of measurements (data) on the basis of some general principle or model and a set of specific conditions relevant to the problem in hand" (Menke, 1989). On the other hand, the inverse problem is to obtain or infer the parameter(s) by a model, from measurements of related dependent variables; in other words, it is a process of "parameter(s) estimation" or "parameter(s) identification" (Neuman and Yakowitz, 1979; Hyun and Lee, 1998).

Solving an inverse problem means making an inference about the physical system from observations based on the model (Scales and Tenorio, 2001). In groundwater

transport inverse problems, the parameters are chemical rate coefficients, such as the dispersivity coefficient, and the observations are solute concentrations. In order to solve such problems both the forward and inverse functions need to be identified. Suppose F is a forward function, which can be expressed as

$$O = F[P] + e \tag{1.1}$$

Where:

F is a forward function that maps "P" from the parameter space to "O" in the observation space (Figure 1.1).

P is a vector of all parameters and lies in the parameter space ***P***.

O is a vector of all discrete observations in the observation space ***O***.

e is a vector of measurement errors and lies in space ***O***.

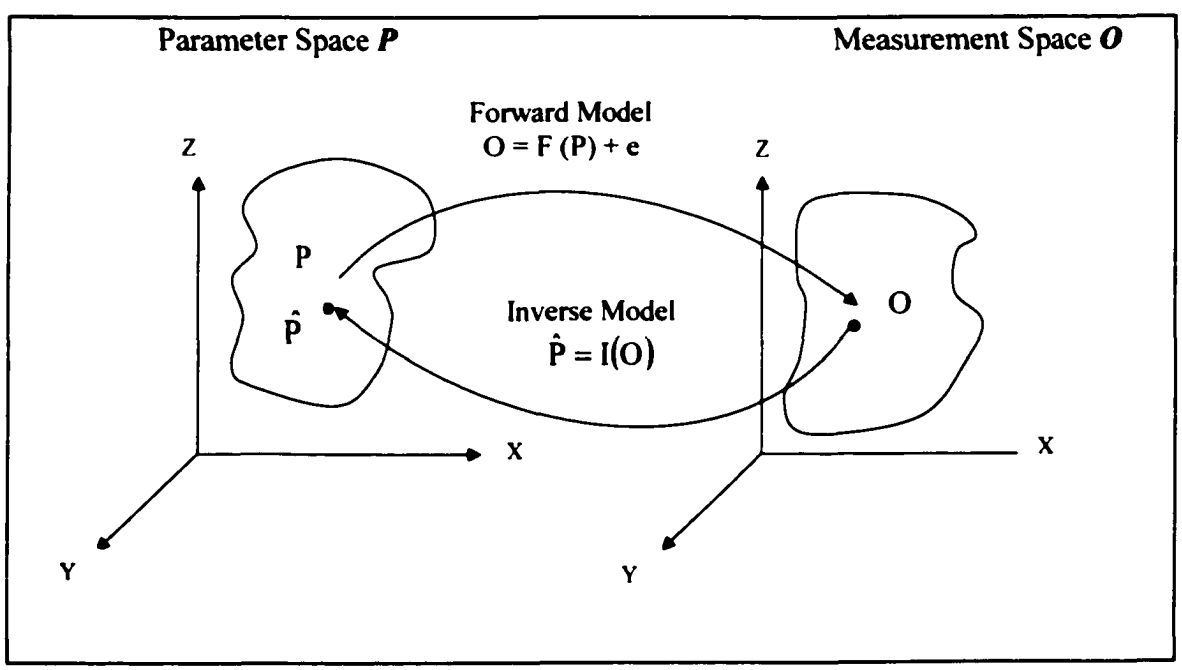


Figure 1.1 Graphical representation of the inverse problem (after McLaughlin and Townley, 1996).

The inverse problem can be expressed as:

$$\hat{P} = I(O) = I[F(P) + e] \tag{1.2}$$

Where:

I is a generalized inverse operator that maps the observation vector O from the observation space O to an estimate of P in the parameter space P (Figure 1.1).

$\hat{P}$ is an estimate of P and lies in the parameter space P.

1.3 Research Objective

The objective of this research is to use a methodology based on artificial neural networks to solve the advection-dispersion inverse problem. The proposed methodology aims to investigate the plausibility of successfully training an artificial neural network to obtain an inverse model for a hypothetical homogeneous, two-dimensional aquifer system. The research aims to examine the capability of the inversion method to accurately estimate values of longitudinal dispersivities using concentration values derived from a forward model at a fixed time. The existence, uniqueness, and stability of the solutions, and their accuracy in obtaining accurate estimations of the parameter, will be examined. The research will also investigate the minimum number of hidden layer(s), neurons, and training patterns that will result in the best solution of the inverse problem, as well as the ability of the neural network to obtain a solution for the advection-dispersion equation with different spatial discretizations of the domain under study.

Chapter 2

LITERATURE REVIEW

2.1 The Inverse Problem in Groundwater Transport

2.1.1 Introduction

Early in the twentieth century Hadamard (1932) introduced the concept of well-posed (correct) problems. He stated that in order for a mathematical model of a physical problem or a boundary value problem for a partial differential equation to be well posed it should satisfy the following conditions:

1. A solution for the problem exists (existence).
2. The solution is unique (uniqueness).
3. The solution is stable (stability).

If the solution lacks any of these conditions the problem is ill posed, as is often the case in inverse problems (Yeh, 1986), especially when solved by traditional methods (Carrera and Neuman, 1986a). In discussing ill-posed problems, mathematicians have repeatedly asserted that consideration of such problems is meaningless (Lavrent'ev et al., 1986). The same opinion has also been expressed by groundwater modelers (Carrera and Neuman, 1986a). As a result, interest in solving inverse problems waned in the 1980s, and the difficulty of overcoming ill-posedness limited the use of the inverse problem as a standard tool in quantitative groundwater modeling. However, interest in solving inverse problems revived with the introduction of methods for solving ill-posed problems,

whether linear or nonlinear. In addition, with the rapid development of high-speed computers, groundwater hydrologists succeeded in developing many methods for solving inverse problems (Colton et al., 2000).

2.1.2 Classification of the Inverse Model

The past three decades saw many advances in solving the inverse problem, and various techniques were developed to solve the inverse problem of groundwater. Neuman (1973) classified the techniques for solving inverse problems into direct and indirect methods. In the direct method, the model parameters (e.g., dispersivity) are treated as unknown or dependent variables and the observations (e.g., concentrations) as known parameters in the governing equation. For example, assume that the model and measurement errors are negligible, and the hydraulic head variations and derivatives over the domain are known. The governing equation in this case becomes a linear first-order partial differential equation in terms of the unknown aquifer parameters, such as hydraulic head or dispersivity, which can be solved directly to obtain the parameters (Yeh , 1986).

In practice the observations are sparse; therefore, an interpolation method is usually used to obtain the missing observations. The interpolated observations contain error due to the interpolation process (as do the real data due to measurement), so when these observations are used in the governing equations, an error term is introduced. Such an error is called the equation error. The goal of the inverse solution is to minimize the equation error; therefore, this method is also called equation error criterion (Yeh, 1986).

The indirect method takes a different approach to solving inverse problems. This method assumes that the observations are the dependent variables, and searches for the

optimal parameters to minimize some criterion function of the residuals between calculated and observed values of the dependent variables (Peck et al., 1988).

The traditional indirect method for solving the groundwater flow and contaminant transport inverse problem is the trial-and-error calibration process, in which the parameters are manually adjusted until there is a reasonable match between simulated and observed site-specific data, such as hydraulic heads and contaminant concentrations. This method has many limitations, such as subjectivity, since it depends on the ability of the modeler to evaluate the complex relationship between model parameters and simulated observations. It is also a time-consuming, tedious procedure since it is not automated (Carrera and Neuman, 1986a), and does not give a statistical framework for parameter estimate uncertainty (Wagner, 1992). Calibration statements could be described statistically, by some standard such as error of residual mean, or residual standard deviation (Walton, 1992). However, because of its limitations, the method cannot guarantee a stable solution and an optimal parameter estimate (unique solution).

Estimation of hydraulic conductivity has long been the primary goal of groundwater inverse research (McLaughlin and Townley, 1996; Mayer and Huang, 1998), resulting in a considerable body of literature devoted to the inverse problem in groundwater flow. However, research directed at solving the inverse problem of solute transport is limited. This scarcity could be due to the complexity of solute transport problems, which is due to numerical difficulties associated with solving the advection-dispersion equation. The in-situ techniques for obtaining the transport parameters, such as field tracer tests, are limited by time and resource constraints and conceptual limitations of the advection-dispersion equation (Peck et al., 1988). Relevant work in the

area of the transport inverse problem has been published only in recent years, probably as a result of the increased attention to water quality and related problems due to human impact on natural systems (Medina and Carrera, 1996).

In the following section reviews of works on the inverse problems of solute transport are presented. In this review these works are classified according to the purpose of the inversion problems; this approach differs from Neuman's, (1973) classification according to the method of solving the inverse problem from a mathematical and statistical viewpoint. Because of the complexity of transport phenomena many of the works include identification of dispersion coefficients, flow parameters, source of pollution, time of release, and other parameters related to solute transport.

2.1.3 Uncoupled Flow and Transport Parameter Estimation

Early inverse problems of solute transport were based on the assumption that the flow field is well known. Murty and Scott (1977) introduced an inverse approach to estimate the dispersivities from observed solute concentration values in a hypothetical two-dimensional aquifer. The solution was formulated as concentration polynomials by using double interpolation, through the extension of Newton's method to find the longitudinal dispersivity and the ratio of longitudinal to transverse dispersivities from a set of selected values. If the estimation was based on just two observed values the solution is unique. For more than two polynomials, the estimation was achieved iteratively by nonlinear least square regression in order to minimize the difference between the calculated and observed concentration values. The accuracy of the

parameters determined by this method depends on the accuracy of the measurements of the concentration values used (Keidser and Rosbjerg, 1991).

Umari et al. (1979) solved the inverse problem by general nonlinear programming using the concept of quasi-linearization to estimate the dispersivity by minimizing the discrepancy between observed and calculated concentration values in a hypothetical two-dimensional aquifer. The method was fast and stable, and the estimation was accurate when the noise introduced into the concentration values was small. However, the method was not tested using errors of the magnitudes typically encountered in the field. Neither Umari et al. (1979) or Murty and Scott (1977) provided means for quantifying the reliability of the estimated parameters.

2.1.4 Coupled Flow and Transport Parameter Estimation

Little work has been done on simultaneous estimation of flow and solute transport parameters (Medina and Carrera, 1996). Strecker and Chu (1986) were the first to study the coupled inverse problem. They used a constrained least squared minimization method formulated as a quadratic programming technique coupled with the method of characteristics (MOC) to estimate transmissivities from piezometric heads, and dispersivity coefficients from concentration observations for a hypothetical aquifer. Two steps achieved the solution. First, the estimation of transmissivity was obtained from head observations, and then the dispersivity coefficient was estimated. The main assumption made was that the difference between the solutions and observations of head and concentrations constitute two error functions. These error functions were due only to the incorrect parameter values used in the MOC model. However, in the indirect method

the total error consists of observation error, model error, and parameterization error (Sun, 1994). The method was able to accurately estimate the unknown parameters under ideal situations, but without any evaluation of the effects of measurement error on the parameter estimates, or statistical assessment of parameter estimate reliability.

Wagner and Gorelick (1986) used weighted nonlinear least squares multiple-regression together with finite difference contaminant transport simulation to estimate parameters of flow and solute transport, such as the dispersion coefficient and first order decay rate. The study was conducted on a one-dimensional hypothetical homogenous soil column and non-equilibrium transport in a stream system in northern California. The researchers attempted to overcome the limitations of previous work by statistically assessing the reliability of parameter estimation, using the means and standard deviations of parameter estimate, which were calculated using the Monte Carlo method. They assumed that the retardation factor and boundary conditions were known, that errors were random variables with zero mean, that unknown errors associated with the i and j observations of concentration were uncorrelated, and that the errors were independent and normally distributed. These assumptions transferred the least squares estimate to a maximum likelihood estimate (Draper and Smith, 1981).

The proposed method accurately estimated the nonlinear parameters even when large random errors were introduced to the observations. The use of spatial data gave estimates that were two to three times more reliable than those based on temporal data for all parameters except velocity. The proposed method was limited to homogenous systems, and the statistical quantification of parameter reliability estimations was limited to linear regression models.

It is worth noting that with the Monte Carlo method the number of realizations needed to achieve convergence for the statistics of concern becomes crucial as medium heterogeneity increases. Studies have shown that to stabilize second-order moments in relatively mild heterogeneity in two-dimension domains, over 1000 Monte Carlo iterations are required, and in three-dimension heterogeneous domains this number of iterations is insufficient (Hassan and Hamed, 2001). Also, Monte Carlo is computationally intensive, and lacks well-established convergence criteria (Guadagnini and Neuman, 2001).

Kauffmann et al. (1990) applied a finite element model coupled with a least squares method based on the Marquardt method to estimate eight parameters of flow and transport, (e.g., longitudinal and transverse dispersivity) in an alluvial sand-gravel aquifer of the Upper Rhine Valley. A finite element scheme was used to obtain the forward transient flow and solute data and parameters. The optimization method was applied as a first step to flow and transport parameters separately, and as a second step to the flow and transport parameters simultaneously.

The results of the first step showed that the inverse solution was ill posed due to non-uniqueness of both flow and transport parameters. In the case of simultaneous optimization of flow and transport, the results were physically acceptable, but the optimized solution was still ill posed due to non-uniqueness and instability.

Medina and Carrera (1996) used maximum likelihood based on the Marquadt method to estimate aquifer parameters such as dispersivity. Prior estimates of the parameters of interest, as well as head and concentration measurements, were used as basic information. A Gaussian distribution of the errors was assumed, which transfers

the likelihood method to a least squares method. Two real examples were used: a laboratory tracer test and a field case of groundwater pollution in Spain. The model for the latter case displayed some limitations, such as lack of data concerning strength of the source and inconsistent sampling. The method showed good capability of successfully estimating the values of the parameters and to provide information about their certainty. The researchers did not study the stability of their solution.

El-Serafy et al. (1996) assumed that the advection-dispersion equation could be interpreted as a Fokker-Planck equation. Therefore, it was possible to derive an Ito stochastic differential equation as a random walk model to represent the movement of a single particle that is consistent with the advection dispersion equation. By applying the partial differential equation to many particles, the distribution of the plume was simulated. The inverse problem was formulated as a Lagrangian multiplier method to minimize an objective function defined according to a quadratic criterion. This method was applied to a two-dimensional homogeneous hypothetical case of radial flow, for an instantaneous injection, a continuous point source, and a point source/sink at a well. The methodology accurately estimated the dispersivity values in the three cases.

Mayer and Huang (1998) presented an inverse problem using maximum likelihood estimation based on a genetic algorithm. The spatial and temporal distributions of head and concentration from a three-dimensional field tracer test were used to estimate the flow and transport parameters, such as zoned hydraulic conductivity, geostatistical hydraulic conductivity, and dispersivity. This method showed that as hydraulic conductivity increases the objective function residual decreases. The estimated parameters where within the range of the field data. Although this method is robust, it

was less efficient computationally than a quasi-Newton algorithm. It was suggested that a gradient-based method might improve the inversion method more accurately than the genetic algorithm.

Valstar (2001) solved the inverse problem using a Bayesian approach on a synthetic heterogeneous aquifer to estimate the hydraulic conductivity values and sorption coefficients on a scale smaller than the grid size. The real data (heads, mobile, and immobile concentrations) and a prior estimate of the state variables were obtained from the forward model. The inverse solution was good in estimating the head and concentration distribution, and the solution was stable as 10 percent Gaussian error was added to the real data of heads and concentrations, but the method failed to obtain detailed distribution of the hydraulic conductivity.

Sun et al. (2001) used weighted least squares based on the Gauss-Newton-Levenberg-Marquardt method to identify hydraulic conductivity, dispersion coefficient and four colloid parameters (geochemical heterogeneity, favorable (fast) colloid deposition rate coefficient, unfavorable (slow) colloid deposition rate coefficient, and colloid release rate coefficient) in a two-dimensional heterogeneous hypothetical model. The inverse solution was performed in two stages. The first stage solved for the hydraulic conductivity and dispersion coefficient; in the second stage the colloid parameters were estimated. This solution was based on the assumption that the colloid dispersion coefficient is similar to the solute dispersion coefficient in porous media where the size of exclusion does not effect the colloid advection velocity and dispersion coefficient. This assumption is valid for sandy aquifers where the solid matrix is several orders of magnitude larger than the colloid size.

2.1.5 Parameter Estimation and Source Identification

One of the early works in source identification is that by Gorelick et al. (1983). These researchers used least squares regression and linear programming combined with a groundwater transport model to identify the location, magnitude, and disposal period of a contaminant source in a hypothetical aquifer system. The linear programming minimized the absolute error while the least squares minimized the sum of the squares. In this method the hydraulic parameters were assumed to be known and the location and concentrations of the source were the unknown model parameters. The minimizing methods obtained accurate estimates when the data were noise free in the case of steady state simulation. In the transient case both methods successfully located the source of pollution and its magnitude because of the abundance of the solute concentration data.

Wagner (1992) presented a method combining groundwater flow and transport simulation with non-linear maximum likelihood estimation to determine optimal estimation of the parameters and source characteristics based on hydraulic head and concentration values. In this study, the problem of treating the hydrologic and source parameters separately was overcome. However, the maximum likelihood estimation, like the least squares estimation, assumes that the model parameters are unknown deterministic parameters, and the randomness is in the concentration measurement errors. In practice, errors in the measurement process are not large enough to explain observed deviation between measured and simulated values. Therefore, it is useful to include the model error in the solution, which was simply assumed by this researcher.

Sidauruk et al. (1998) solved the inverse problem based on correlation coefficient optimization. The method used an analytical solution of contaminant transport and a set of concentrations to obtain an estimate of mass transport parameters, the dispersion coefficient, and flow velocity, as well as source characteristics such as the amount of pollutant, its initial location, and time of origin. The use of the analytical solution can give a linear relationship between the logarithm of concentrations and certain combinations of parameters. By minimizing the correlation coefficient of the linear regression, the parameters needed are determined by explicit formulas similar to linear regression equations. Because this method is based on analytical solutions, which assume simplified conditions, it is restricted to simple geometry and flow conditions in homogenous aquifers.

Sciortino et al. (2000) solved an inverse problem to identify the location of a DNAPL pool in the bottom of a three-dimensional porous medium. The solution was based on the least square minimization problem using the Levenberg-Marquardt method as a search procedure. An analytical model was used to generate the data to be tested by the proposed solution. The approach was first tested by adding normally distributed noise to the generated data, and the model's ability to predict the location and the size of the pool was tested by using data from a bench-scale DNAPL pool dissolution experiment. The goal of the analyses was to test the influence of observation measurement errors and model parameter uncertainty on the capability of the proposed model to predict the location and the dimension of the DNAPL pool.

To identify aquifer parameters and a pollution source Mahar and Datta (2001) used a nonlinear programming model solved by using the projected augmented

Lagrangian algorithm for two-dimensional hypothetical homogeneous groundwater systems. The projected augmented Lagrangian algorithm was used in the computer code Modular In Core Nonlinear Optimization System (MINOS) (Murtagh and Saunders, 1993) to solve a nonlinear programming problem. The assumption was that the flow field and transport parameters were unknown, as opposed to linear programming where the flow parameters are assumed known. This method embeds both the flow and transport equations as constraints in the optimizing solution, and overcomes the drawback of the response matrix proposed by Gorelick and Remson (1982).

Mahar and Datta (2001) solved two steady state problems. In the first the heads were unknown, while the flow and transport parameters (for example the dispersivity coefficient) were known. In the second model all were unknown. For validation purposes, the results of the optimization method were compared with results from the solute transport model USGS-Method of Characteristic (MOC) developed by Konikow and Bredehoeft (1978). The method was shown to be applicable for parameter and source identification, determining the magnitude and specifying duration of unknown groundwater pollution sources. The method was ill posed in terms of source identification due to the non-uniqueness. However, the non-uniqueness may be removed by restricting the source locations and the range of parameter values.

2.1.6 Parameter Estimation and Front Location Identification

Anderman et al. (1996) proposed a three-stage iterative method to solve the inverse problem based on an analytical solution and the modified Gauss-Newton non-linear regression to estimate dispersivity and its effect in locating the plume front in a one-dimensional, homogenous synthetic case. The three stages were feed-backward

processes. In the first stage the hydraulic properties were calculated; in the second stage the produced flow field was used as a base for the inversion process to estimate the dispersivity; and in the third stage the estimated dispersivity was used to correct the location of the plume. These three stages were repeated until the correct location of the plume was obtained. The results indicated that this method could not converge to the minimum sum of squared residuals. Furthermore, it is not clear how to implement the third stage in a more complex numerical method.

2.1.7 Limitations of the Direct and Indirect Methods

In practice, observation wells are limited in number and scattered in an arbitrary way in the flow region. As a result, the missing observations needed to solve the direct inverse problem must be interpolated. The interpolated observations will contain errors due to the nature of the interpolation procedure. If field observations, which contain error due to measurements, and interpolated observations are used in the governing equation, an error term will result. Therefore, if the statistical properties of the state variable (field and interpolated observations) were not considered, error terms would be found in the results of parameter identification (Yeh, 1986).

The solution is generally unstable in the presence of noise. The solution is also highly dependent on the level of discretization used in the numerical solution of the governing equation; generally, the stability increases as the number of parameters decreases. If the Euclidean norm is used, the equation error criterion of the least squares will be transferred to a quadratic function. Both least squares and quadratic functions are

stable with noise free data (Sun, 1994). Usually the estimated parameter is an average value of the parameter over the region of interest (Xiang and Elsworth, 1992).

The indirect method suffers from the limitation that there is usually more than one best parameter that minimizes the objective function (minimizer) in the parameter space, and since the method is automated, the computer may select a local minimum other than the global one. Therefore, the solution may be non-unique (Sun, 1994).

2.2 Artificial Neural Networks

2.2.1 Introduction

In the last decade, artificial neural networks (ANNs) have been widely used in various branches of research fields, one of which is hydrology. Due to this increased usage, ANNs are considered as one of the most important current trends in numerical techniques for groundwater models (Stauffer, 2001). The structure of ANNs is derived from real biological nervous systems that perform complex imitation tasks from large amount of information in noisy, fuzzy environments.

The strength of ANNs comes from their inherent characteristic. ANNs are data dependent; they do not impose a functional relationship between the dependent and independent variables. Instead, the functional relationship is determined by the data during the training process (Cybenko, 1989; Hornik et al., 1989). Their ability to handle the implicit relationship between input and output and their non-linearity properties is very important, particularly when the problem of interest is inherently non-linear (Haykin 1999). They are also characterized by their ability to generalize, which makes them a good tool for solving the inverse problem. These characteristics have encouraged many

researchers to investigate and apply ANNs as a modeling and inversion tool in hydrology and other areas, such as geophysics and satellite data analysis.

In general, the process of modeling by ANN consists of two steps. The first step is to train or teach the ANN by supplying some data (input and output) as an example of the environment that the ANN will be asked to simulate. The second step is to present to the ANN new data (input only) that it did not see during the training, and ask the ANN to simulate these data and give the output. The degree of success of the ANN in learning the given data and in producing the desired output can by tested by any statistical test.

ANN can be defined as "a structure (network) composed of a number of interconnected units (artificial neurons). Each unit has an input/output (I/O) characteristic and implements a local computation or function. The output of any unit is determined by its (I/O) characteristic and its interconnection to other units. The network usually develops an overall functionality through one or more forms of training" (Schalkoff, 1997).

A simplified diagram of a feedforward network is shown in Figure 2.1. This network consists of one input layer, one hidden layer and one output layer. The hidden layer consists of computation nodes, which called hidden neurons. The number of input, hidden, and output layers may be one or more depending on the problem in hand and the type of algorithm used in the network.

The connections between the layers are called links or synapses. In each layer there are many neurons (nodes, units, and cells), and the number of neurons in each layer may differ from that of the other layers.

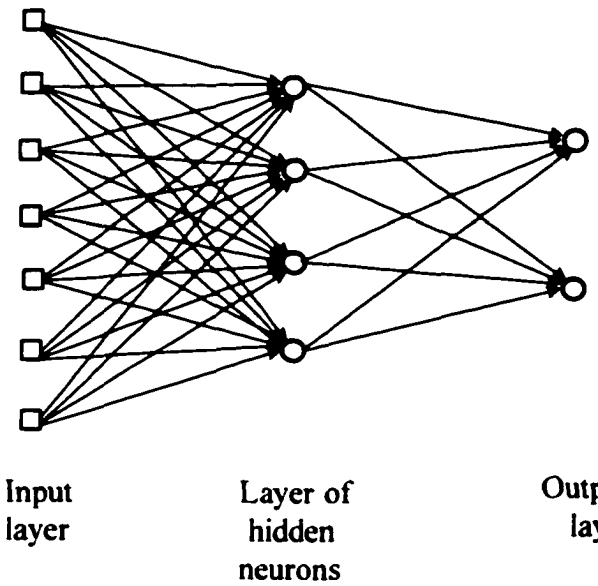


Figure 2.1 A feedforward network with one hidden layer.

The basic processing elements of neural artificial networks are the neurons. The artificial neuron is an information-processing unit which is fundamental to the operation of the neural network (Haykin, 1999). Figure 2.2 shows a model of a neuron, in which $x_1, x_2, \ldots, x_m$ represent *m* scalar inputs to the neurons, which may come from the physical system variables or outputs of other nodes. These *m* inputs form an input vector $X = (x_1, x_2, \ldots, x_m)$. The three basic elements of a neuron are links (synapses), adder, and activation function. Links connect each unit in one layer to those in the next higher layer.

A scalar weight, or strength, characterizes the links; for example, an input signal x_j at the input of link j connected to neuron k is multiplied by the link weight w_{kj}. The subscript "k" refers to the neuron in question, and the "j" subscript refers to the end of the

link to which the weight refers. The weights may lie in a range that includes positive and negative values, and form a vector $W_{m.} = (w_{k1}, w_{k2}, \ldots, w_{km})$.

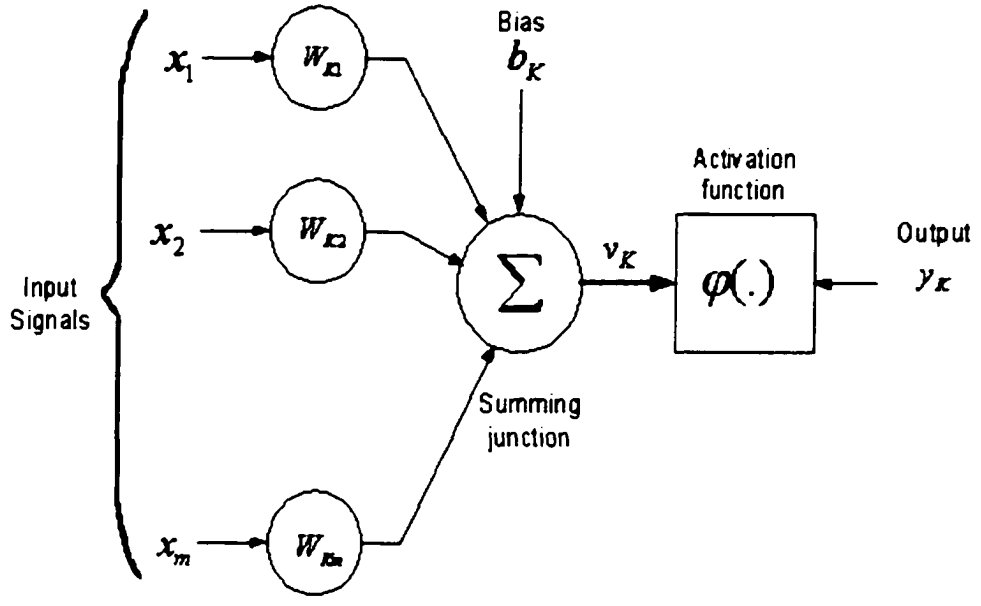


Figure 2.2 Typical neuron model (after Hykin, 1999).

The function of the adder is to sum weighted input signals to the neuron. The product of the summation will be a scalar; let this be denoted $u1$. An external scalar bias denoted b_k or, in this example, b_1 is connected to the adder; the bias can be viewed as being added to $u1$, which will give the scalar v_k or here v_1. The value v_1 is the argument of the activation function also called transfer or squashing function that acts as a limiting function for the amplitude of the output signals of the neuron y_1 to some finite values. Typically the normalized range of the output of a neuron can be between 0.0-1.0 or –1.0-1.0 according to the type of activation function used in the network.

It should be noted that the computation takes place only in the neurons of the hidden layers, while the input layer(s) serves as a holding site for the inputs applied to the neural network, and the output layer(s) acts as the point at which the output of the neural network is available.

The term "feedforward" is a description of the structure of the network. It indicates that the output of each neuron, scaled by the value of the connecting weight, is fed forward for neurons in the next higher layer. Figure 2.1 illustrates a typical feedforward network. The term "backpropagation" expresses how an error at a higher or outer layer can be propagated backwards to nodes at lower or inner layers. The gradient of the backpropagating error measure will be used to determine the needed weight modifications for links between the hidden neurons.

The neural network architecture is the structure of the algorithms used to design the neural network. One of three fundamental classes of neural network architectures is the multilayer feedforward network (Haykin, 1999). The architecture includes the number of neurons in the input and output layers, number of hidden layers, number of neurons in each hidden layer, type of transfer function, and error function used to monitor the network's performance.

2.2.2 Neural Network Modeling and Inverse Problems

ANNs have already proved very successful in modeling various problems in hydrogeology. The following characteristics of ANNs have made them particularly attractive and promising for successful applications in modeling:

1. Artificial neural networks can map problems with a high level of error data (Rumelhart and McClelland, 1989), and they can provide stable solutions without severe loss of accuracy (Govindaraju, 2000 a);
2. ANNs can reach a conclusion from incomplete data (Spichak and Popva, 2000). Transport problems require detailed and extensive field data (Gutjahr, 1992); artificial neural networks could be a good tool for parameter estimation, since the ability to observe the detailed hydraulic behavior of the system is limited.
3. ANNs are data dependent; they do not impose functional relationships between the observations and parameters. Instead, the functional relationship is determined by the data during the training process (Cybenko, 1989; Hornik et al., 1989). Therefore, unlike regression-based techniques, ANNs do not impose the need to make assumptions about the mathematical form of the relation between input and output (Govindaraju, 2000 b). Also, ANNs can approximate any continuous function to any degree of accuracy.
4. ANNs are capable of handling the implicit relationship between input and output without an explicit physical design.
5. ANNs can have a linear or non-linear activation function, and can solve both types of problems (Haykin, 1999). This property has motivated many researchers to use ANNs, particularly when the problem of interest is inherently non-linear, as in hydrological problems (Imrie et al., 2000).
6. ANNs are described as global optimization methods (Peter and Kalogerakis, 2001), and a local minimum is rarely a practical problem (Rumelhart and McClelland, 1986). This characteristic makes ANNs a good tool for solving the inverse problem.

7. ANNs have the ability to adapt to a large number of parameters, and the time necessary for ANNs recognition depends on the number of unknown parameters used in the modeling process rather than the physical dimension of the domain (Spichak and Popva, 2000).

The above-mentioned characteristics have encouraged many researchers to investigate and apply ANNs as a modeling and inversion tool in hydrology. For example, Govindaraju (2000 a, b) gives an introduction to ANNs for hydrologists, a summary of the commonly used algorithms, and landmarks for applying ANNs to hydrologic problems. The researcher describes the similarities and differences between neural networks and other modeling tools, and discusses the ANN's strengths and limitations. The author provides many examples of ANN applications in hydrology, including rainfall-runoff, stream flow, water quality, and groundwater modeling. Dawson and Wilby (2001) surveyed more than fifty articles that applied various types of ANNs to rainfall-runoff modeling, and they concluded that clear guidance in applying ANNs in modeling and a comparison of ANNs and conventional statistical methods are needed.

Morshed and Kaluarachchi (1998) used ANN alone and coupled with a genetic algorithm to evaluate the capability of the two methods to solve an inverse problem related to free-product contaminant in both homogeneous and heterogeneous unconfined aquifers. The ARMOS computer code was applied as a forward model to produce the observed data. The input parameters for the ANN were the grain size index α, of van Genuchten (1980), and the saturated hydraulic conductivity. The outputs were free-product thickness and free-product recovery volume. The input was divided into training and test subsets, and 5% noise was added to the test subset. The hyperbolic tangent

function (tanh) was applied, the input normalized between (-0.1, + 0.1), and the output between (-0.8, + 0.8). The correlation coefficient R was used to assess the performance of the ANN. The ANN was able to solve the inverse problem and produce acceptable values of the parameters in the homogenous case, while in the heterogeneous case the results were not encouraging.

To solve the inverse problem of transmissivity, Torfs and Bier (2000) used a feedforward backpropagation ANN based on the Levenberg-Marquardt algorithm as a local technique and pure random choice as a global one. The researchers used the Carrera and Neuman (1986c) study as an example of applying the neural network as an inversion method. The finite element groundwater flow model MICROFEM was used to generate the forward data. The inverse problem used piezometric head observations to estimate the transmissivity values, and was formulated to adjust transmissivity values so that the difference between the observed and modeled head measurements was minimal.

The results showed that the ANN was capable of estimating the transmissivity values as proposed by the zonation method used by Carrera and Neuman (1986c). Although the solution was not unique, the method overcame the subjectivity limitation of the zonation technique. Instead of creating a bandwidth variation including different zones chosen by the modeller, the ANN randomly chose the starting weights and generated different starting fields of the transmissivity.

Shigidi (2000) estimated the transmissivity values in a two-dimensional hypothetical model using an iterative neural network. About two thousand transmissivity fields were generated by a geostatistical method. These fields were the input for the feedforward backpropagation ANN, and the forward-simulated heads given by the

MODFLOW model were the output of the ANN. During the geostatistical generation, some values of transmissivity and heads in specified locations in the domain were chosen to be known and fixed in order to examine the reliability of the ANN in honoring the known data. The data were divided into four subsets: training, testing, monitoring, and new data. The sigmoid logsig function was used and the input data were normalized between (-0.8, +0.8). The trained networks were inverted using an iterative procedure. The method produced hydraulic head fields that matched to a good extent the forward hydraulic head fields, and each map was a unique solution by itself.

Rogers and Dawla (1994) used feedforward backpropagation artificial networks to optimize aquifer remediation through the search of optimal well locations. The flow and transport parameters were assumed known in a hypothetical heterogeneous domain. The input consisted of fifty pumping realizations under some constraints for twenty wells, and the output represented whether or not the tested realizations were meeting the optimization objectives and constraints. A hybrid finite difference and finite element code was used to generate the forward pumping realizations.

A comparison of the results obtained to those of two conventional optimization techniques, namely nonlinear optimization model NPSOL (Gill et al., 1986) and SUTRA (Voss, 1984), indicated that the results were consistent. The authors noted some advantages of the artificial neural networks over the conventional non-linear optimization management modeling techniques. Usually the flow and transport codes are linked to the optimization techniques; the latter calls the flow and transport codes sequentially several times to reach an optimal solution for the remediation problem. The ANN method gave seven different solutions for the remediation problem with less parallel calling of the flow

and transport codes. The ANN was a lighter computational burden and resulted in significant real time savings.

Gautam et al. (2000) used a black box model based on a feedforward backpropagation three-layered ANN using field measurements of soil moisture to estimate stream runoff. The method was used to verify run-off generation processes in the catchment. The outputs were normalized to (-0.9, +0.1), and the solving function was the sigmoid logsig. The results showed a good consistency with the measurements of stream discharge. The model could be a useful tool for catchment monitoring where other models are time consuming and difficult to calibrate.

Rizzo and Dougherty (1994) presented the concept of neural kriging for characterization of aquifer properties. A feed-forward counterpropagation and an ordinary kriging approach were used to develop the new neural kriging method. A hypothetical two-dimensional domain was discretized into 20x20 cells. The domain was divided into 5 zones of three different classes of log conductivity. From these 5 zones twenty observations were randomly drawn. The coordinates of the cells corresponding to the twenty log conductivity observations were the input, and the outputs were the log conductivity values. The proposed method gave unbiased error and good estimations of the log conductivity. The method showed no need to normalize the input, and is a useful tool in geohydrology when applied to specific well-defined problems.

Clair and Ehrman (1998) used a three-layered backpropagation neural network to study the effect of seasonal climate changes in the Atlantic Provices of Canada on the hydrology and geochemical cycling of 14 river basins influenced by wetlands. The inputs for the neural network were basin slope, basin area, mean minimum and maximum

temperatures, total rain, total snow deposition, and total precipitation, of monthly measurements for ten years. The outputs were discharge, dissolved organic carbon, and dissolved organic nitrogen. The results of the neural network model were compared to measured data. The comparison showed good correlation between the measured data and the predicted data from the neural network model. However, despite the applicability of neural networks on large scales, they cannot be used to explain basin processes.

Yang et al. (1997) applied a three-layered backpropagation ANN with a hyperbolic tangent transfer function to simulate fluctuations in midspan water table depth and drain outflows as influenced by daily rainfall and potential evapotranspiration rates in Ottawa, Canada. Because the measurements of the water table and drain outflow were not taken in the same day, two different ANNs were used. The inputs were daily rainfall, potential evapotranspiration, and either previous water-table depth or previous drain outflow. The outputs were water table depth or drain outflows. A comparison between the measured and predicted data showed that the neural network worked well, except that the ANN performed poorly in simulating water table depth in the deep part of the soil profile (depth>1500mm).

Mukhopadhyay (1999) investigated the utility of an artificial neural network for the spatial interpolation of transmissivity data for a confined aquifer. More than one model was developed, with different inputs in each model. The inputs used for the neural network training were location coordinates, altitude of the top surface of the aquifer, and a logarithm of the total dissolved solids in the groundwater of the aquifer. The output of the network was the logarithm of the transmissivity values of the aquifer under study. The results of the different models were compared with kriged transmissivity values. The

neural network performed better than the kriging technique in estimating transmissivity values at well points, and can be a useful alternative method of parameter estimation.

Koekkoek and Booltink (1999) used three series of neural networks to evaluate pedotransfer functions in order to predict soil water retention characteristics, as well as to compare the performance of the neural network and the regression based pedotransfer functions. The inputs for the first network were topsoil bulk density and organic matter, clay, silt, and sand content. The second network included matric potential, and the third included soil structural data expressed as the upper and lower boundary of the ped-size class. The outputs of the three ANNs were volumetric water content at different matric potentials. The performance of the ANN was tested by using the root of the mean squared error difference between the measured and the predicted water content, and the multiple coefficient of determination (R^2). In this method the neural networks performed better than regression-type pedotransfer functions, but the differences were not significant.

Schaap et al. (1998) calibrated backpropagation neural network models for prediction of van Genuchten (1980) water retention parameters and saturated hydraulic conductivity from basic soil properties, and to determine which basic soil properties are the most relevant for estimation of the hydraulic properties. In order to ensure broader applicability of a hierarchical structure, they used a data set that covered most of the soil textural triangular. The input data were extracted from 4515 laboratory samples obtained from about 30 sources in the USA.

The input for two series of 12 water retention and 19 hydraulic conductivity network models were developed in a hierarchical approach. The inputs were: textural

class, gravel, sand, silt, clay contents and bulk density, porosity, gravel content, soil horizon, water content at different pressures, effective porosity of the different horizons, saturated and residual water contents, and α and n curve shape parameters of the van Genuchten equation. Four additional models were developed to estimate saturated hydraulic conductivity using sand, silt, clay contents and bulk density in one model, and the previous properties and effective porosity in three different pressure heads in the other three models. They also compared the results of the ANN with published pedotransfer functions of Mishra and Parker (1989, 1992) by predicting the saturated hydraulic conductivity using retention parameters (i.e., saturated water contents, residual water contents, α and n curve shape parameters).

The results of the models were compared to fitted or measured hydraulic parameters using the coefficient of determination (R^2), and the root mean residuals were used to compare both the values estimated by the neural networks and the published pedotransfer functions with measured data. In order to evaluate the uncertainty in the predicted hydraulic properties, the authors combined the neural network approach with the bootstrap method.

The researchers found that the neural network was more efficient in the prediction of the outputs than the existing published pedotransfer functions. However, the ANN was unable to predict the saturated water content even with the most relevant inputs, and the performance of the ANN was unacceptable when using only information about soil textural class. It appeared that the water retention curves contain information that cannot be predicted from macroscopic variables such as sand, silt, and clay contents, and bulk density. The results from combining the hierarchical approach with the bootstrap method

showed that there could be considerable uncertainty in predicting the hydraulic properties, which can be used in conjunction with the Richards equation to generate uncertainty estimates in simulated unsaturated flow processes.

Krom and Rosbjerg (2000) investigated the use of a fully connected error backpropagation feedforward ANN to solve general contaminant transport problems in a three-dimensional heterogeneous domain, and to show the impact of problem generality on the size of the data, net architecture, and optimization approach. The well coordinates and the pumping rate were mapped as input to estimate mass recovery, contaminant loss, and failure of the remediation design by using the hyperbolic tangent function (tansig function). Both the input and output were normalized by multiplying and adding to them the experimental standard deviation and experimental mean of the raw data. The ANN performed at accuracy levels comparable to the numerical models used to develop the optimization data set for two different problems, and could estimate mass recovery using unconstrained well and moderately constrained pumping rate.

Chapter 3

METHODLOGY

The methodology used in this research consisted of two tasks. The first task was to generate the forward solutions of the transport model with various dispersivity values. The second task was to train the artificial neural network with the forward data and then use the trained neural network to predict the dispersivity values from new forward solutions (concentration values) of known dispersivity values.

In this chapter the methodology introduced to solve the inverse problem is outlined, the computational tools used to implement the solution are explained, and the neural network is described.

3.1 Methodology

In general, the methodology for this research consisted of training an artificial neural network to solve the inverse problem. The concentration fields generated by forward flow and solute transport models were used as input to train the network to estimate values of the longitudinal dispersivity values which the network had not encountered before. This study was conducted on a hypothetical two-dimensional unconfined aquifer under steady state flow conditions, based on published data from the Macrodispersion Experiment (MADE) at the Columbus Air Force Base site, northeastern Mississippi (Adams and Gelhar, 1992, Boggs et al., 1992). The flow chart shown in

Figure 3.1 presents the steps implemented to meet the objective of this research. These steps constitute the general methodology used and can be summarized as follows:

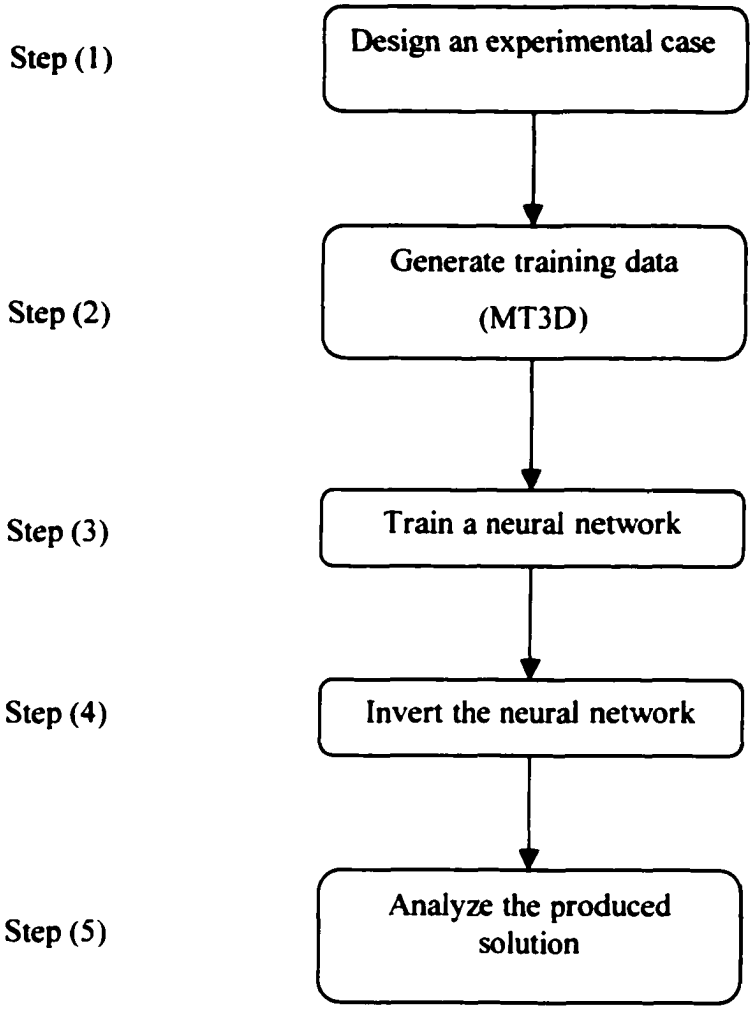


Figure 3.1 Flow chart of the methodology used in this research.

1. Design the hypothetical case study on which the methodology was carried out. The parameters needed to model will be defined from the available published data; for example, (for the flow model) the hydraulic conductivity, storage coefficients, type and location of boundary conditions, aquifer type and thickness, and stresses. For the transport model, it is also necessary to define the boundary conditions, initial aquifer concentration, concentration of injected solute and the duration of the injection process, and the method of solving the forward transport model.
2. Generate the dispersivity values on a random basis using the commercial MATLAB package (From The Math Works, Inc.). These values were used in the forward transport model to obtain the different forward solutions, and also constituted the output for the neural network.
3. A groundwater flow and contaminant transport model was used to generate the concentration values for each simulated longitudinal and transversal value. For this purpose the MODFLOW (McDonald and Harbaugh, 1988) and MT3D (Zheng, 1990) software were used.
4. The MT3D model was modified in order to produce the needed number of concentration fields of the generated dispersivity values.
5. Once the training sets were generated, a neural network was identified and trained to map the relation between the longitudinal dispersivity values and their corresponding concentration fields. The neural network tool in the commercial package MATLAB was used to carry out the training process.

6. After the neural network was trained, new concentration values of known dispersivity values that the neural network had not seen during the training were used to invert the neural network and estimate the longitudinal values.
7. The inverse solution was investigated with dispersivity values within the range and beyond the range of dispersivity values used in training the network.
8. The proposed methodology was applied on different discretization sets of the domain used in this study.
9. The proposed inverse methodology was then examined to assess its existence, uniqueness, and stability, and the accuracy of the results produced by the methodology.
10. During the course of the study the minimum number of hidden neurons in the hidden layer was investigated.
11. The minimum number of training patterns needed by the network to solve the inverse problem was investigated.

A three-dimensional model is used to simulate the flow and solute transport in homogenous media, and will be presented prior to the neural network method. The model consists of two coupled equations, a flow equation and a solute transport equation.

3.2 Groundwater Flow Model

To simulate solute transport in an aquifer the description of the groundwater flow model is an essential matter. The governing equation that describes the transient flow can be described in Cartesian tensor notations:

$$\frac{\partial}{\partial x_i}\left(K_{ij}\frac{\partial h}{\partial x_i}\right) = S_s\frac{\partial h}{\partial t} + W \qquad i,j = 1,2,3 \qquad (3.1)$$

Where:

K_{ij} is the hydraulic conductivity of the porous media (a second order tensor) [L/T];

h is the hydraulic (or potentiometric) head [L] ;

S_s is the specific storage [1/L];

t is time [T];

W is the volumetric flux per unit volume (positive for inflow and negative for outflow) [1/T];

x_i are the Cartesian coordinates [L].

For the steady state flow condition the right-hand side of the equation will be zero, since there is no change of water volume in storage with time. If the porous medium is homogenous (i.e. the component of hydraulic conductivity is independent of position for a particular direction), equation 3.1 may be simplified to:

$$K_x \frac{\partial^2 h}{\partial x^2} + K_y \frac{\partial^2 h}{\partial y^2} + K_z \frac{\partial^2 h}{\partial z^2} = 0 \qquad (3.2)$$

The assumption for this equation is that Darcy's Law applies, the fluid properties (density and viscosity) are homogenous and constant, the porous medium deforms only vertically, and the volume of individual grains remains constant during deformation.

3.3 Groundwater Transport Model

The mass balance equation governing the transport flow through a three-dimensional saturated medium is usually expressed as follows (Javandel et. al., 1984):

$$\frac{\partial C}{\partial t} = \frac{\partial}{\partial x_i}\left(D_{ij} \frac{\partial C}{\partial x_j} \right) - \frac{\partial C}{\partial x_i}(vC) + \frac{q_s}{\theta} C_s + \sum_{K=1}^{N} R_k \qquad i, j = 1, 2, 3 \qquad (3.3)$$

Where:

C is the solute concentration [M/L^3];

t is time [T];

x_i are the Cartesian coordinates [L].

D_{ij} are the components of the dispersion coefficient tensor [L^2/T].

v_i is the seepage velocity component in the x_i direction [L/T]

q_s is the volumetric flux of water per unit volume of aquifer representing sources (positive for inflow and negative for outflow) [1/T];

C_s is the concentration of the sources or sinks [M/L^3];

θ is the effective porosity of the porous media [L^0];

R_k is the rate of solution production in reaction k of N different reactions [M/L^3T].

The velocity of the groundwater controls the movement and mixing of chemicals dissolved in groundwater. It was shown by Lohman (1972) that the seepage velocity of groundwater can be derived from Darcy's Law, and can be expressed in Cartesian tensor notation as:

$$V_i = \frac{K_{ij}}{\theta}\frac{\partial h}{\partial x_i} \tag{3.4}$$

The first term in equation (3.3) represents the change in concentration due to hydrodynamic dispersion, which is assumed to be Fickian; therefore, the driving force is the concentration gradient. Dispersion in porous media refers to spreading of contaminants over an ever-growing region that would be predicted from only the groundwater velocity vector (Bear and Verruijt, 1987). As described by Anderson (1984), hydrodynamic dispersion is a result of a mechanical dispersion due to local

variation of the actual velocity at the microscale from some mean velocity of flow, and molecular diffusion, a result of concentration variations.

The hydrodynamic dispersion coefficient may be related to the seepage velocity of the groundwater and the porous media properties using Scheidegger's (1961) equation:

$$D_{ij} = \alpha_{ijmn} \frac{V_m V_n}{|V|} \qquad i, j, m, n, = 1, 2, 3 \qquad (3.5)$$

Where:

α_{ijmn} is a component of the dispersivity tensor, a fourth rank tenser [L];

V_m is a component of a velocity vector in the m direction;

V_n is a component of a velocity vector in the n direction;

$|V|$ is the magnitude of velocity; and

$|V| = \sqrt{V^2_x + V^2_y + V^2_z}$

For an isotropic porous media, Scheidegger showed that the dispersivity tenser could be defined in terms of the longitudinal and transverse dispersivities, α_L and α_T. This defines two dispersion coefficients oriented with the direction of flow: the longitudinal dispersion coefficient D_L and the transverse dispersion coefficient D_T, where:

$D_L = \alpha_L|V|$; and

$D_T = \alpha_T|V|$.

To account for the molecular diffusion, an isotropic diffusion coefficient, D_m, is added to the effect of the mechanical dispersion. The diffusion coefficient is defined as:

$D_m = \omega D_d$

Where:

ω is a coefficient related to tortuosity; and

D_d is a diffusion coefficient in aqueous phase [L^2/T]

The second term describes the effect of advective transport, or transport of miscible solute at the same velocity of the groundwater. To measure the degree of advection domination, the Peclet number is usually used. The Peclet number is a dimensionless number which measures the effectiveness of advection transport compared to the effectiveness of dispersion transport, and is measured as follows:

$$Pe = \frac{\nu L}{D}$$

Where:

ν is the seepage velocity;

L is the characteristic distance; and

D is the hydrodynamic dispersion coefficient.

According to Bear (1972), from the analysis of one-dimensional flow field experiments presented by Pfannkuch (1963) and Saffman (1960), a Peclet number less than 0.4 represents molecular diffusion transport, a Peclet number between 0.4-5 (approximately) represents a zone of equal effect of molecular diffusion and mechanical dispersion, and a Peclet number larger than 5 represents transport controlled by advection dispersion type.

The third term in the governing equation is the sink/source term. This term may be treated as a point source/sink type, which includes wells, drains, and rivers, or as an areally distributed type, which includes recharge and evapotranspiration. In the case of using a point source, it is necessary to specify the concentration of the source. The last

term represents the chemical reactions of the solute within the aquifer; in the case of a conservative solute, this term will vanish.

3.4 The Artificial Neural Network

3.4.1 Neural Network Architecture

Users of ANNs have the options of designing an application-dependent network architecture to be used in some desired computation, selecting a common preexisting architecture for which training algorithms are available (for example a feedforward network) or adapting a preexisting architecture to suit a specific problem (Schalkoff, 1997). The multilayer feedforward neural network is the best known, and arguably most important, architecture and can be used for inverse problems (Suykens et al., 1996, Schalkoff.1997; Mehrotra et al., 1997). It consists of a number of layers: an input layer, an output layer, and one or more hidden layers connected in a feedforward manner.

Links, or weights, connect each neuron in one layer only to those in the next higher layer. Thus the output of a neuron, scaled by the weight of the link, is fed forward to provide a portion of the activation for the neurons in the next higher layer. These scaled outputs of the neurons in the feedforward direction are referred to as function signals or input signals.

In the input layer the neurons are used only to hold the input values and to distribute these values to the neurons in the next layer. Accordingly the input layer neurons do not implement any calculation, and weights do not exist.

The output layer is a computational layer; it holds the computed outputs of the network-hidden layers. At the neurons of the output layer, an error signal originates and propagates backward layer by layer through the network.

The remaining neurons in the hidden layers between the input and output layers are computational neurons, which are not part of the input or output layers of the network. Each neuron of the hidden layers has an activation function, which may be linear or non-linear.

In this arrangement of the network, each neuron of the hidden and output layers is designed to do two computations. The computation of the functional signals at the output neurons proceeds layer by layer in a forward direction. The second computation proceeds in the same fashion but in the backward direction of the signal error, which is a computation of an estimate of the gradient vector (i.e. the gradient of the error surface with respect to the weights connected to the inputs of a neuron).

3.4.2 The Backpropagation Algorithm

The backpropagation algorithm can be presented after Haykin (1999) as follows:

The error signal at the output of neuron j at iteration n can be defined as:

$$e_j(n) = d_j(n) - y_j(n) \tag{3.1}$$

Where:

$d_j(n)$ is the desired response for neuron j ;

$y_j(n)$ is the functional signal appearing at the output of neuron j at iteration n.

Let δ be the instantaneous sum of error squares for neuron j

$$\delta = \frac{1}{2} e_j^2(n) \tag{3.2}$$

Thus the instantaneous sum of error squares over all neurons in the output layer at iteration n is:

$$\delta(\mathrm{n}) = \frac{1}{2}\sum_{j\in C} e_j^2(n) \tag{3.3}$$

Where:

C includes all neurons in the output layer of the network.

Let N denote the total number of patterns in the training set. The average sum of error squares can be calculated by summing $\delta(\mathrm{n})$ over all n and then normalizing with respect to the size N as follows:

$$\delta_{\mathrm{av}} = \frac{1}{N}\sum_{n=1}^{N}\delta(n) \tag{3.4}$$

The δ_{av} is a function of all free parameters of the network. For a given training set δ_{av} represent the objective function. The goal of the training process is to minimize this function, which can be viewed as a measure of the training performance. In the training process the free parameters are adjusted to minimize the objective function, and the adjustments are made in accordance with the respective errors computed for each pattern presented to the network.

In Figure 3.2 the neuron j is fed by a set of function signals from the layer in the lower level to the left. The induced local field $v_j(\mathrm{n})$ is defined as the weighted sum of all links inputs and bias. The induced local field is produced at the input of the activation function $\varphi_j(.)$ associated with neuron j, is therefore calculated as:

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n)y_i(n) \tag{3.5}$$

Where:

m is the total number of inputs (excluding the bias) fed to neuron *j*; and

w is the link weigh.

The link weight w_{j0} represents the weight of the fixed input y_0 equal to +1.0, which is the bias applied to neuron *j*. Hence the functional signal $y_j(n)$ appearing at the output of neuron j at iteration n is

$$y_j(n) = \varphi_j(v_j(n)) \tag{3.6}$$

The backpropagation algorithm applies a correction Δw_{ji} (n) to the link weight w_{ji} (n), which is proportional to the derivative of $\partial\delta(n)/\partial w_{ji}(n)$. The latter is the error gradient, which may be expressed by using the chain rule as:

$$\frac{\partial\delta(n)}{\partial w_{ji}(n)} = \frac{\partial\delta(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)}\frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{3.7}$$

Differentiating both sides of equation (3.3) with respect to e_j (n) we get:

$$\frac{\partial\delta(n)}{\partial e_j(n)} = e_j(n) \tag{3.8}$$

Differentiating both sides of equation (3.1) with respect to y_j (n) we get:

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{3.9}$$

Also differentiating equation (3.6) with respect to v_j (n) we get:

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi'_j(v_j(n)) \tag{3.10}$$

Next, differentiating equation (3.5) with respect to w_{ji} (n) we get:

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{3.11}$$

Substituting equations (3.8)-(3.11) in equation (3.7) yields:

$$\frac{\partial \delta(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi'_j(v_i(n))y_i(n) \tag{3.12}$$

The correction of the Δw_{ji} applied to w_{ji} is defined by the delta rule:

$$\Delta w_{ji} = -\eta \frac{\partial \delta(n)}{\partial w_{ji}(n)} \tag{3.13}$$

Where:

η is the learning-rate parameter.

The minus sign accounts for gradient descent in weight space (i.e., seeking a direction for weight change that reduces the value of δ(n)). Substituting equation (3.12) in equation (3.13) yields

$$\Delta w_{ji} = \eta e_j(n)\varphi'_j(v_i(n))y_i(n) \tag{3.14}$$

Let the local gradient of neuron j which points to required changes in the link weights can be defined as:

$$\xi_j(n) = -\frac{\partial \delta(n)}{\partial v_j(n)} \tag{3.15}$$

$$= -\frac{\partial \delta(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \tag{3.16}$$

$$= e_j(n)\varphi'_j(v_j(n)) \tag{3.17}$$

Then:

$$\Delta w_{ji} = \eta \xi_j(n) y_i(n) \tag{3.18}$$

Equation (3.18) states that the weight correction is equal to the learning rate parameter multiplied by the local gradient multiplied by the input signals of neuron j.

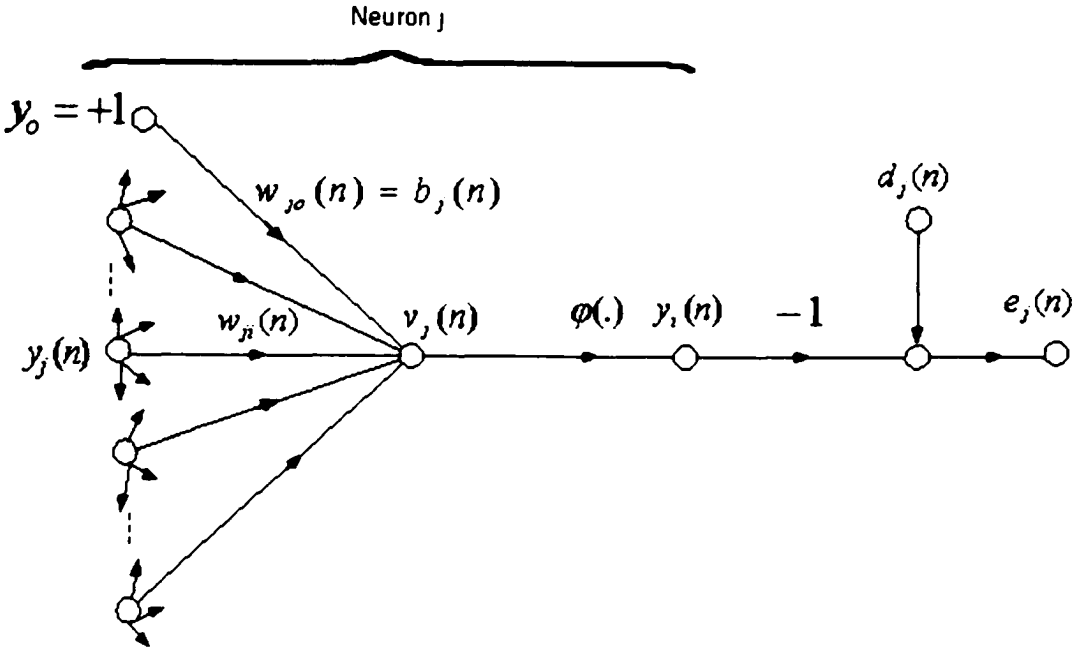


Figure 3.2 Signal-flow graph representing the details of output neuron *j*

(after Haykin, 1999)

3.4.3 Activation Functions

In most cases of network modeling a non-linear activation function is used. The hyperbolic tangent function (tansig) is one form of the sigmoidal non-linear activation functions. This is a continuously differentiable non-linear function. Mathematically, tansig functions can be represented as (after Mehrotra et al ., 1997):

$$\varphi_j(v_i(n)) = a \tanh(bv_i(n)) \tag{3.19}$$

Where:

tanh is the hyperbolic-tangent function; and

a, b are constants >0

The derivative of the hyperbolic function with respect to $v_j(n)$ yields:

$$\varphi'_j(v_j(n)) = ab\sec h^2(bv_j(n)) \tag{3.20}$$

$$= ab(1 - \tanh^2(bvj(n))) \tag{3.21}$$

$$= \frac{b}{a}[a - y_j(n)][a + y_j(n)] \tag{3.22}$$

Chapter 4

Solving the Inverse Problem

4.1 Designing the Hypothetical Problem

4.1.1 Physical Domain

The synthetic case used in this study was based on data from the Macrodispersivity Experiment (MADE) study at the Columbus Air Force Base (CAFB) in northeastern Mississippi (Adams and Gelhar, 1992; Boggs et al., 1992). Part of the MADE study data was used in the present study to build the hypothetical case, and was not used to model the data to obtain the dispersivity values of the site. The MADE experiment was a large-scale, natural gradient, field tracer study conducted in a heterogeneous, alluvial aquifer.

A subset of the MADE physical domain was used in the present model. The physical dimensions of the aquifer subset were 80m wide by 120m long by 13m deep. Figure 4.1 shows the head distribution in the subset of the domain used in this study.

The subset of the unconfined aquifer was discretized into a finite difference grid. The grid contains 40-columns (i) and 40-rows (j). The modeled domain was bounded by constant head boundaries on two sides and no-flow boundary on the other two sides and on the bottom. The hydraulic gradient was 0.00325, the hydraulic conductivity was 17

m/day, the porosity was 0.35 and a specific yield was 0.13. One injection well was used with an injection rate of 2.0 m^3/day.

The synthetic tracer test case consisted of a pulse injection of a conservative tracer solution into the saturated zone of the unconfined aquifer. $10m^3$ of tracer were injected though the injection wells over a period of 48 hours. The tracer was injected at a depth of 5.1 through a screened interval of 0.6 m. The injected tracer concentration was 2500 mg/L, and the aquifer initial concentration of the tracer was 0.05 mg/L.

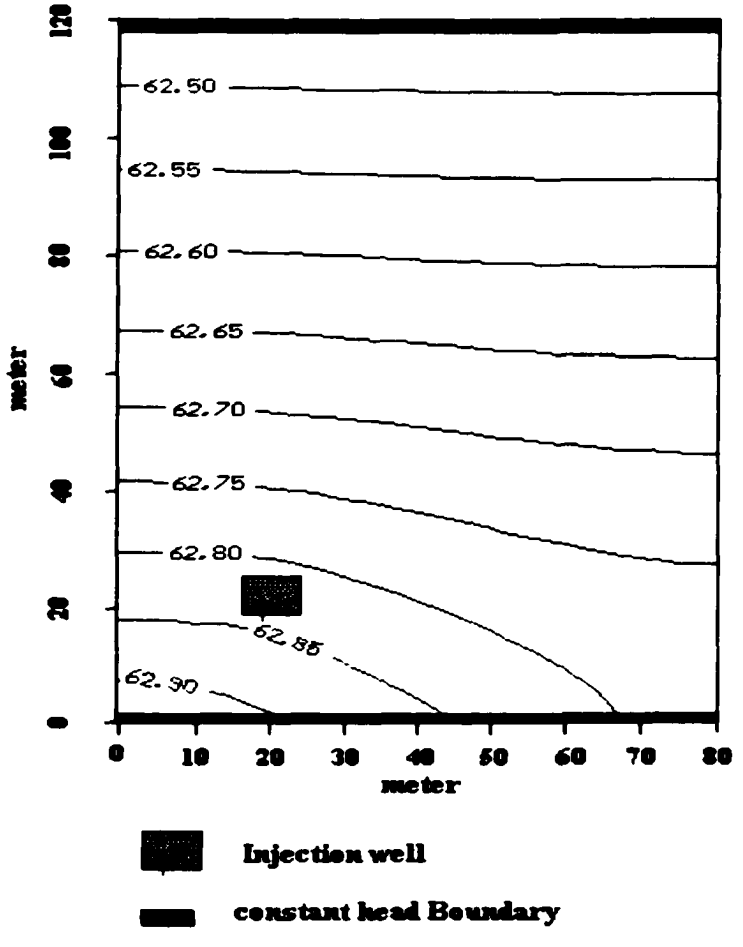


Figure 4.1 Hydraulic Head (in meters) distribution and boundary conditions in the study area.

4.2 Generation of the Training

To apply the neural network method, it is necessary to generate fairly representative examples of the problem to be modeled. For the forward model, the flow problem was solved by using the block-centered finite difference scheme, and the method of characteristic technique was used to solve the advection-dispersion equation. A FORTRAN file was written to generate the needed number of concentration fields. The FORTRAN file called the MT3D.exe file to solve the transport equation for each value of longitudinal and transverse dispersivity from a file containing all the dispersivity values. The calculated concentration fields for all the dispersivity values were saved in a single file. This file and the file containing all the dispersivity values were used as the input and output files respectively, for the neural network. The modified solute transport MT3D simulator was used to generate 210 concentration fields, each field corresponding to a dispersivity value, where the dispersivity values were randomly generated to range between 0.5-10.95 m at an increment of 0.05 m using the commercial MATLAB package. The simulated concentration values were those at day 126 after the injection of the tracer.

4.3 Data Preparation

Artificial neural networks are parallel computing systems, and the feedforward type is analogous to biological neurons, which are arranged in layers. Therefore, the input, output, and weights have to be prepared for the neural network as layers. Data preparation involves a number of processes such as data reshaping, standardizing, and dividing the data into calibration and test sets.

The forward concentration values were prepared to fit in a single matrix of dimensions of 1600 x 210 elements. The 1600 elements are the values of one concentration field (one run of the MT3D) for a single value of dispersivity; each matrix (40x40 cell) of concentrations from the MT3D was reshaped to fit in a column vector of 1600 elements. The dimension 210 is the number of total runs or the total number of dispersivity values. The dispersivity values were also arranged as a column vector of 1x210 dimension. The concentration values were the input for the artificial neural network, and the dispersivity values were the output. In this arrangement each vector of 1600 elements in the input was mapped with one value of the output vector.

For a rigorous analysis by an artificial neural network, three data sets should be used: a training set, a test set and a validation set (Dawson and Wilby, 2001). The training set in a supervised learning network is a pair of input and its corresponding output, which are representative examples of the physical system's behavior. These pairs are used to adjust the weights of the network in the hope that when new untrained (input only) data are presented, the network can give the desired results. This new untrained data constitute the test set. The third set is the validation set, which contains different data from that of the training and testing data sets, and is used to estimate network performance and decide when to stop the training.

The goal of training a network is to be able to generalize its behavior to new data situations. The term "generalization" means that a network was trained correctly (or nearly so), so that if a new data or test set (input only) not used in the training is presented to the network, it will estimate a correct output. The assumption made here is that the new set was drawn from the same population as the training set. The training

process may be viewed as a curve-fitting problem, and may suffer from overfitting or underfitting problems, as with other flexible nonlinear estimation methods such as kernel regression and smoothing splines (Coulibaly et al., 2000). The overfitting problem can be viewed as overtraining or memorization of the training set. If the training data set is large and the training is allowed to continue until the error is minimum, the network may fall into the problem of memorizing the training data by finding a feature in the data, for example noise, which is not related to the function being modeled. Therefore, if new data is presented to the network, the error may be large, or the network may lose its ability to generalize between similar input-output patterns.

A solution to the problem of overfitting is to stop the training before it is completed, at a point when further reduction in the error is possible. This method of stopping the training is referred to as the early stopping method. In a backpropagation algorithm the network learns in stages, proceeding from fairly simple to complex mapping functions as the training progresses. Usually the quality of the training is measured by the mean-square error between the inputs and outputs. As training proceeds the mean-square error decreases as the number of epochs (i.e. one complete presentation of the entire training set during the learning process) increases. It starts off at a large value, decreases rapidly, and then continues to decrease slowly as the network makes its way to a local minimum on.

Given that the training process is a curve fitting, it may be difficult to determine when it is best to stop the training before the network over fits the training data. To overcome this problem cross-validation is used, in which the training set is divided into two subsets: the estimation subset and the validation subset. The estimation subset is

used to train the network; as the training proceeds, the training process is stopped periodically, and the network is tested on the validation subset after each period of training. Because of this periodic stopping the method is called the early stopping method of training.

In the validation subset the error is continuously monitored, and the training process continues as long as the error in the validation subset continues to decrease even if the training process is improving (Mehortra et al., 1997). The monitoring of the error in the validation subset does not mean that the set was used for training, because the training or adjusting of weights only takes place in the estimation subset, as suggested by Hetch-Nielsen (1990). Increase of error in the validation subset can be explained as an indication that further training will likely result in network overfitting.

Figure 4.2 shows the forms of modified curves of estimation, and validation processes. The estimation curve decreases monotonically as the number of epochs increases; while the validation curve decreases monotonically to a minimum, then starts to increase as the training proceeds. At the point where the validation curve starts to increase, the training process is stopped, because beyond this point the network starts to learn the noise in the training data, and the overfitting problem will occur. The process of training is stopped at this point, and the current set of weights is assumed to reflect optimal values.

The generalization of the training process depends on the following factors: the size of the training set and how representative it is to the system under modeling, the architecture of the neural network, and the physical complexity of the system (Haykin,

1999). Clearly there is no control over the complexity of the system, while the other two factors could be under control.

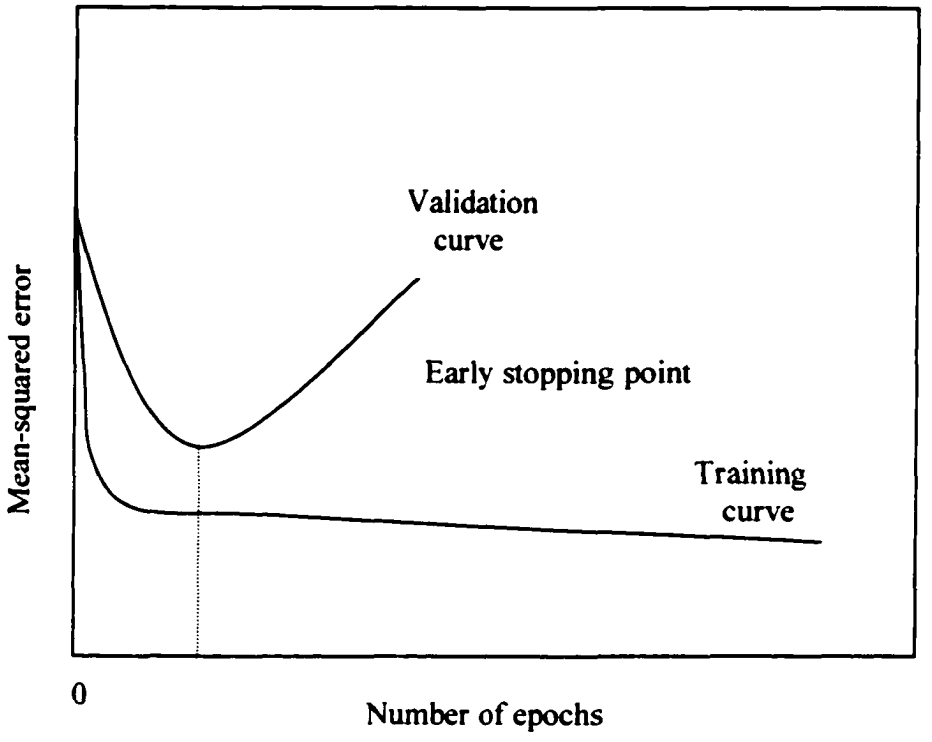


Figure 4.2 Illustration of the early-stopping rule based on cross-validation.

Amari et al. (1996) suggested an equation on the basis of batch learning with a single hidden layer for determining the size of estimation and validation subsets, which may give the best generalization by using the early stopping method. For a training set of size N and number of weights and biases (free parameters) W, if $N<W$, the early stopping method does improve the generalization process over exhaustive training (i.e., when the

complete set of examples is used for training and the training process is not stopped). This equation suggested that overfitting may take place if $N<30W$, and the cross-validation may be needed to stop the training (Haykin, 1999). Amari et al. (1996) suggested the following equation to determine the size of the validation set:

$$r_{opt} = 1 - \frac{\sqrt{2W-1}-1}{\sqrt{(2W-1)}} \tag{4.1}$$

Where:

r_{opt} is the parameter that determines the split of the training data between estimation and validation. For example, if W is 21, the r_{opt} is 0.156, which means that about 84 percent of the training data were assigned to the estimation subset, and the remaining 16 percent were assigned to the validation subset

For large values of W, the suggested equation may be approximated to

$$r_{opt} = 1 - \frac{1}{\sqrt{2W}} \tag{4.2}$$

If the training set is larger than $30W$ ($N>30W$) exhaustive training is satisfactory to generalize the training process. This could be explained by the fact that the improvement in the generalization behavior of the network achieved by using the early stopping method is small. In practice the only known set size is the training set; the number of weights is not known, and is usually determined by trial and error. Therefore, this equation can be applied to calculate the percentage of the validation subset from the training set only after determining the number of weights.

To determine how many samples are needed for a good generalization, the rule of thumb is to have at least five to ten times as many training samples as the number of

weights in the network. This rule of thumb is a guide to determine in a rough way the number of weights that may be sufficient for good architecture.

The forward data, 210 input-output examples generated by MT3D, were subdivided into four subsets of data, after analyzing the effect of the training pattern on the ability of the network to generalize its behavior. The first subset is the training data. This subset comprised 70 percent of the total input and output data, or 147 values of dispersivity and their corresponding concentration fields. This percentage value of the training set was about 7 times greater than the number of weights used in this study. The second and third subsets were the validation subset and test subset, which each comprised 10 percent (21 values) of the outputs and their corresponding concentration fields. The fourth subset comprised 10 percent of the total data (21) values. These were data that the network had not seen during training, testing, and validation. The new data consisted of concentration fields (inputs) presented to the network after training for simulation by the network to calculate their dispersivity values. The dispersivity values of these 21 concentration fields were used for measuring the performance of the network by different statistical means.

The performance of the network was measured using the mean squared error as regularization performance function. This is a typical performance function used in training a feedforward neural network and can be calculated as:

$$mse = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{1}^{N}(t_i - a_i)^2 \qquad (4.3)$$

Where:

e is the error between estimated and known parameters

t is the N parameters estimated by the network

a is the N known parameters

The error criterion used in this study was a modified performance function of the mean squared error constructed by adding a term that consisted of the mean of the sum of squares of the network weights and biases. It can be measured as:

$$meserg = \gamma mse + (1-\gamma)msw \tag{4.4}$$

Where:

γ is the performance ratio, and

$$msw = \frac{1}{n}\sum_{j=1}^{n} w_j^2 \tag{4.5}$$

Using this performance function caused the network to have smaller weights and biases, and this forced the network response to be smoother and less likely to overfit.

The purpose of the training process is to adjust weights through the network to reach a good solution to the problem. The adjusted weights are usually chosen randomly, and typically the chosen weights are small between –1.0 and 1.0 or –0.5 and +0.5. In feedforward networks the most common functions used are squash functions, the logistic function, and the hyperbolic tangent function. The input squash function can have any values between plus and minus infinity, and the outputs of these functions are squashed between 0.0 and +1.0, or –1.0 and +1.0. The input to the network should be standardized so that its mean value, averaged over the entire set, is close to zero, or is small compared to its standard deviation (LeCun, 1993). The standardization ensures that each input receives equal attention during the training process (Maier and Dandy, 2000). If the magnitude of some of the inputs is much larger than others, the random choice of the weights may be biased by giving much greater importance to inputs of high magnitudes.

In this study the activation function used is the hyperbolic tangent function, and the input values were standardized to fall between −1.0 and +1.0.

4.4 Neural Network Architecture

The way in which the neurons of a neural network are connected is intimately linked with the learning algorithm used in training the network (Haykin, 1999). In the process of designing a feedforward backpropagation network the main task is to search for the optimal architecture that yields the best performance in terms of error minimization, while retaining a simple and compact structure (Govindaraju, 2000 a).

If we considered the factors that control the generalization then we may need either to determine the best architecture while fixing the training size or to determine the best size of training set and fixing the architecture that achieves the goal of neural network modeling (Hush and Horne, 1993). Determining the training size or sample size is an open research area because there is a huge numerical gap between the proposed formulas and the practical results (Haykin, 1999). In this study the training size was fixed and the architecture was sought for the best generalization of the data. The smallest training set that may give a good generalization is also investigated in this research.

Because the size of the input-output layer was fixed, the process of determining the best architecture involved determining the numbers of hidden layers and neurons in each hidden layer that achieve the generalization goal. In practice the trail and error approach is used to achieve the determination. There are two approaches to determine the number of hidden layers and neurons, either to start with a small architecture and introduce more neurons until performance is satisfactory, or start with large architecture

and successively remove neurons until the performance degrades to unsatisfactory level. The first approach was used to determine the best architecture. For the problem in hand, the number of input layers is one with 1600 neurons representing a concentration field, and the output layer is one layer with one neuron representing the dispersivity value corresponding to the concentration field. The total number of patterns is 210 for both concentration and dispersivity fields.

In designing the network architecture there are other factors that need to be selected, such as activation function and error criteria. The choice of activation function depends on the application area or the problem in hand. For application in hydrology where most of the problems are non-linear, a nonlinear function is usually used. The logistic sigmoid and hyperbolic tangent functions (tanh) are normally used (Dawson and Wilby, 2001; Govindaraju, 2000 a). In this research the hyperbolic tangent function tanh $(x) = (1-\exp(-2x))/(1+\exp(-2x))$ was used, where x is the weighted sum of inputs to the neuron.

4.5 Neural Network Training

4.5.1 Effect of Hidden Neurons

According to Kolmogorov (1957) and Sprecher (1965), a multilayer network with one hidden layer can learn any continuous mapping to an arbitrary degree of accuracy. More than one hidden layer may be beneficial for some applications and can make training easier in some situations (Fausett, 1994). One hidden layer was used in training the current network, and the number of hidden neurons was investigated in order to

determine the minimum number of neurons that was satisfactory to achieve the generalization.

As mentioned earlier, the approach of increasing the number of neurons was selected to investigate the best architecture. The network was trained with 5, 10, 15, 20, 25, 30, 40, 50, 60, 80, 100, 150, 200, and 220 neurons. To evaluate the performance, the results of 15 runs of the network with the proposed number of neurons were subjected to regression analysis of the difference between the results of the network and the known results (dispersivity). The same results were also analyzed by surface error percentage, which is the difference between the estimated and known values after sorting the values, calculated according to the following equation:

$$\text{Surface Error Percentage of NN} = (\Sigma(\Sigma(\text{abs}(NT-ST)))/\Sigma(\Sigma(\text{abs}(NT)))*100)\ \% \qquad (4.6)$$

Where:

NT is known dispersivity values;

ST is simulated dispersivity values by the network; and

abs is the absolute value.

Table 4.1 represents the average and variance values of both the surface error percentage and the correlation coefficient for the 15 runs of the proposed number of neurons. The lowest average value of the surface error percentage was 17.16 with 20 neurons, and the highest average correlation coefficient was 0.879 with 5 neurons. The lowest variances of the surface error percentage and correlation coefficient were 377.28 and .035 with 20 and 5 neurons respectively.

Figure 4.3 (a, b) represents the variation of the average of the 15 correlation coefficient (R) and surface error percentage. Figure 4.3 (a) represents average of

correlation coefficient values and shows that the network had the highest correlation coefficient at 5, 20, and 80 neurons, and the lowest value at 40 neurons. Figure 4.3 (b) shows the average of surface error percentage. The highest surface error was obtained with 200 and 40 neurons, and the lowest value with 20 neurons. The figure showed no trend in the surface error percentage, but gives an indication that the best training could be with 20 neurons. In addition, from the correlation coefficient analysis, it may be concluded that the performance started to degrade with the use of more than 5 neurons. It started to improve with more than 15 neurons and achieved optimum performance with 20 neurons; then it started to oscillate.

Table 4.1 Average and variance values of surface error percentage and correlation coefficient.

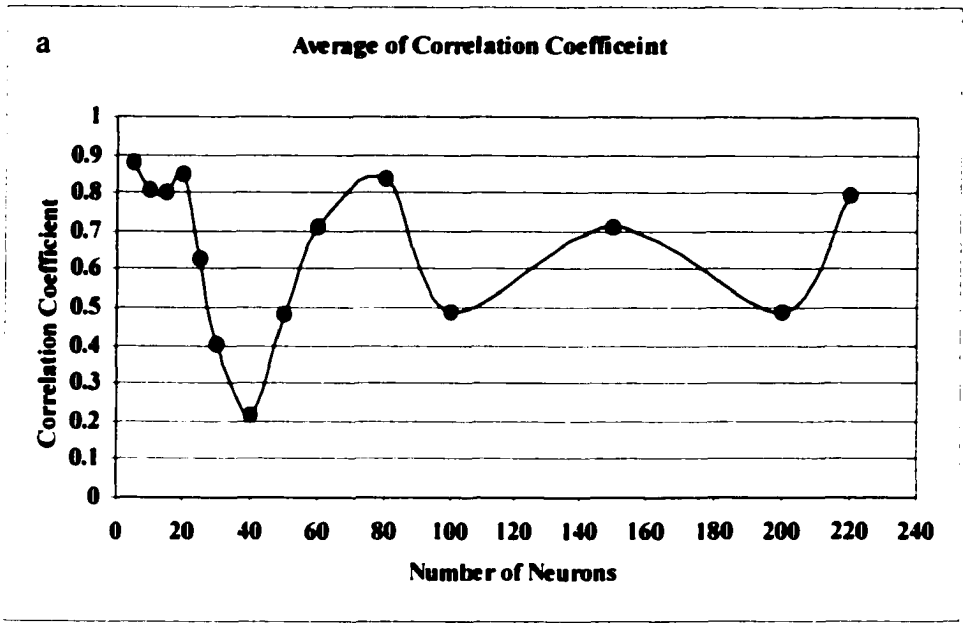
Number of neurons	Surface Error Percentage		Correlation Coefficient	
	Average	Variance	Average	Variance
5	21.20	458.07	0.879	0.035
10	22.73	648.22	0.808	0.118
15	27.45	741.22	0.798	0.109
20	17.16	377.28	0.848	0.152
25	38.17	906.66	0.628	0.229
30	37.06	929.50	0.403	0.605
40	54.51	928.88	0.216	0.640
50	45.51	883.79	0.484	0.391
60	33.86	709.38	0.713	0.259
80	31.74	836.55	0.838	0.041
100	43.16	1187.91	0.488	0.473
150	44.16	402.56	0.713	0.233
200	54.95	847.81	0.486	0.244
220	41.25	970.33	0.793	0.157

Another analysis of the data by calculating the variance of the values of the same type of performance measurement shows the same results. Figure 4.4 (a) shows the

variance of the correlation coefficient values. The figure shows no trend in the data. The lowest variation was achieved with 5 and 80 neurons, and the highest with 40 neurons. For surface error data, the variation in figure 4.4 (b) shows that the lowest variance is with 20 neurons and the highest is with 100 neurons.

The curves have no distinctive trend, and are oscillating; they cannot give any indication of an increasing or decreasing trend in the performance of the network. This type of behavior makes the decision of selecting a specific number of hidden neurons more difficult. The best number of hidden neurons to be selected for this network seems to be 20, because this number of neurons produced the lowest average and variance of the surface error percentage, and the highest average of the correlation coefficient.

The training with 20 neurons looks like an outlier data point in the two types of analyses. This is not true, because in the network the main calculations are in the neurons and its result is the sum of the number of random weights adjusted to learn the main features of the examples. To show that the network has the same behavior with 20 neurons, another 15 runs of the network with 20 neurons were conducted. Figure 4.5 represents variation of the variance of surface error percentage of the second training process of 15 runs of the network. The variance increased to the value of 455.5 m^2 in the second 15 runs from value of 377.28 m^2 in the first 15 runs. The figure shows the same trend for network behavior in this second 15 runs of training process, which indicates that the network produced the same behavior with 20 neurons in the hidden layer.



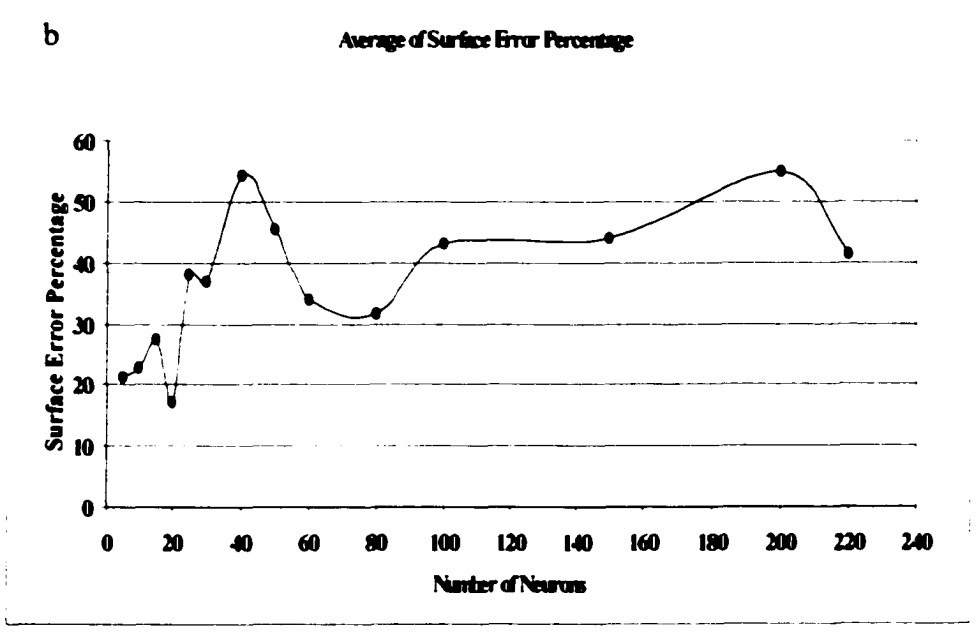
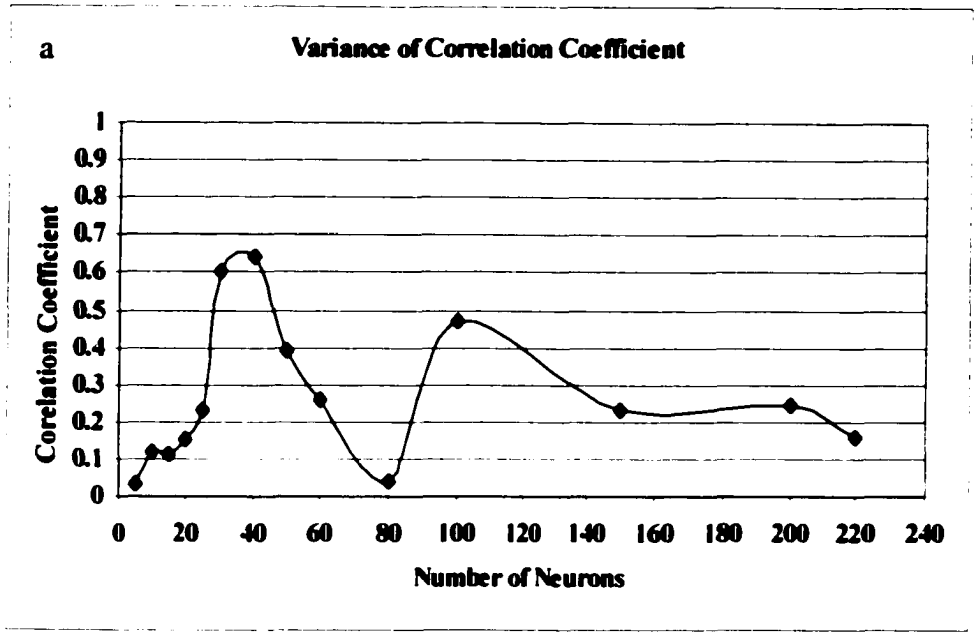


Figure 4.3 (a, b) Variation of average correlation coefficient and surface error percentage as a function of number of neurons.



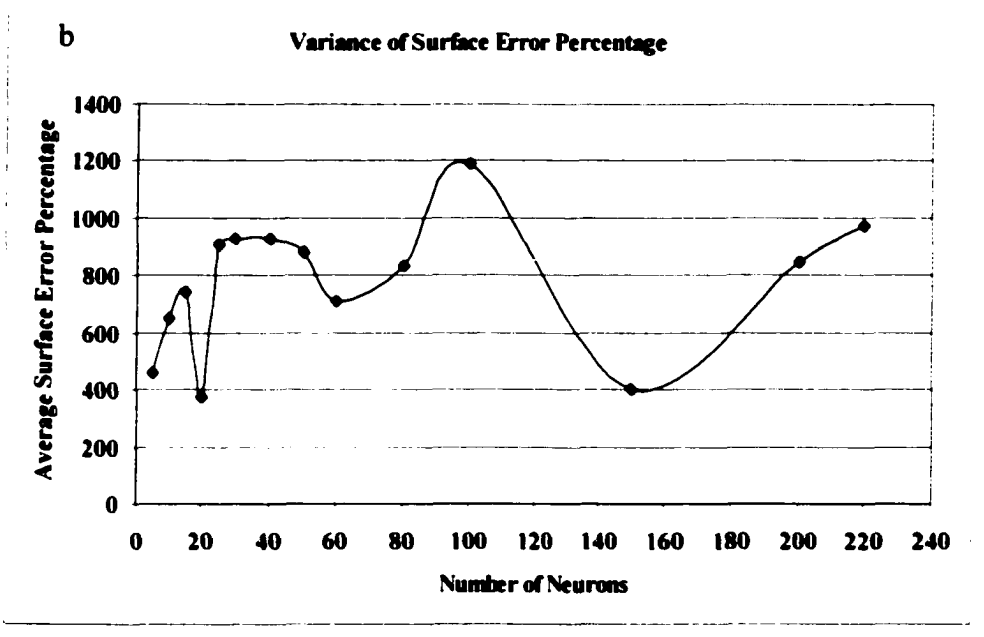


Figure 4.4 (a, b) Variation of variance of correlation coefficient and variance of surface error percentage as a function of number of neurons.

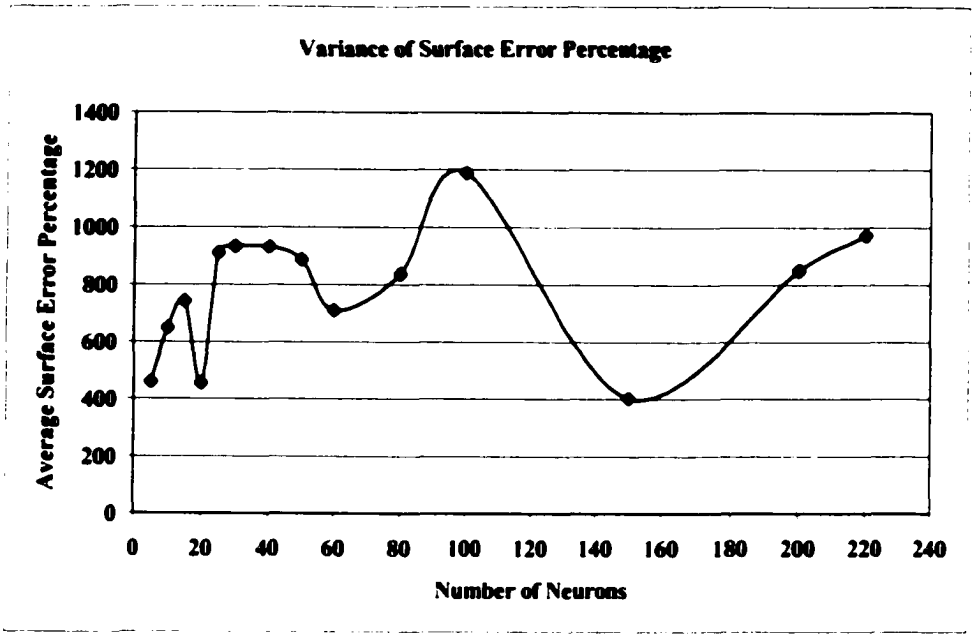


Figure 4.5 Variation of the variance of the surface error percentage in the second training run.

4.6 Number of Training Patterns

The dispersivity values generated randomly fell between 0.5 and 10.95 meters, with an increment of 0.05 meters. Therefore, the total number of generated dispersivity values was 210. To investigate the effect of the number of samples or patterns which give an excellent generalization, the network was trained with the following number of training patterns: 60%, 61%, 62%, 63%, 65%, 67%, 69%, and 70% of the total number of patterns. Table 4.2 shows the lowest value of surface error percentage, the highest correlation coefficient of the result of training the network, average surface error percentage, and average correlation coefficient of 15 runs of the ANN.

The results show that the network was able to generalize its behavior to all of the proposed number of training patterns, but the best results were obtained with using 70% of the data for training, and 10% for the validation and testing sets. Using 70% of the total pattern, the average surface error percentage for 15 runs was 17.17, and the average correlation coefficient was 0.848; these were the lowest and highest values, respectively. These values indicated that 70% (147 patterns) was the optimum number of patterns to be select to train the network.

Table 4.2 Effect of training patterns on generalization behavior.

Number of patterns as percent	Surface Error Percentage	R	Average Surface Error Percentage	Average R	%of validation
60	9.42	0.985	44.93	0.422	20
61	6.74	0.993	39.59	0.555	19
62	4.06	0.997	25.87	0.823	18
63	2.01	0.999	26.36	0.839	17
65	1.96	0.999	23.81	0.796	15
67	2.64	0.998	32.50	0.658	13
69	3.67	0.997	36.30	0.662	11
70	1.27	1.0	17.17	.848	10

4.7 The Inverse Solution

4.7.1 The Inverse Results

The goal of this study was to evaluate the neural network as a possible tool for estimating dispersivity values of an aquifer, given the concentration fields. A MATLAB script was written to perform the needed tasks of training the network and estimating the parameter under study (Appendix A). After training, the network was ready to simulate

the new input or concentration fields and estimate the dispersivity values. As in other modeling approaches in hydrology, the generalization capability of the trained network can be evaluated by presenting to it some data not used in the training and validation and not previously encountered. A set of concentration fields was presented to the trained network to perform the simulation and estimate the dispersivity. The new data consisted of 21 concentration fields corresponding to 21 known dispersivity values. The values of the known dispersivities were selected from the range of the trained dispersivity values. Two more sets of concentration fields corresponding to another two sets selected from the range of dispersivity values used in the training were tested with the trained network. The trained network simulated the 21 concentration fields and estimated their dispersivity values.

In order to evaluate the accuracy of the simulation a regression analysis and surface error percentage analysis were performed between the known dispersivity values and the simulated values given by the network. The correlation coefficient, the slope, and the y-intercept were obtained by regression analysis. Figure 4.6 illustrates a regression analysis between the estimated dispersivity values and the known values. The X-axis represents the 21 known dispersivity values, and the Y-axis represents the response of the network or the estimated values. The figure showed a correlation coefficient equal to 1.0, with a slope of 1.0, and a y-intercept of 0.0095. These results show an excellent fit between the known dispersivity values and those estimated by the network.

Figure 4.7 shows the surface error analysis between the estimated dispersivity values and the known values. The X-axis represents the total number of simulated

dispersivity values given by the network, and the Y-axis represents surface error percentage. The absolute difference between the simulated and the known values is 1.85 percent. The highest error was 0.321 percent of dispersivity value 0.5 m (Table 4.3). Some dispersivity values were estimated without any discrepancy, such as 1.1, and 5.1 m. The results show that the ANN is capable of estimating the dispersivity values with high accuracy.

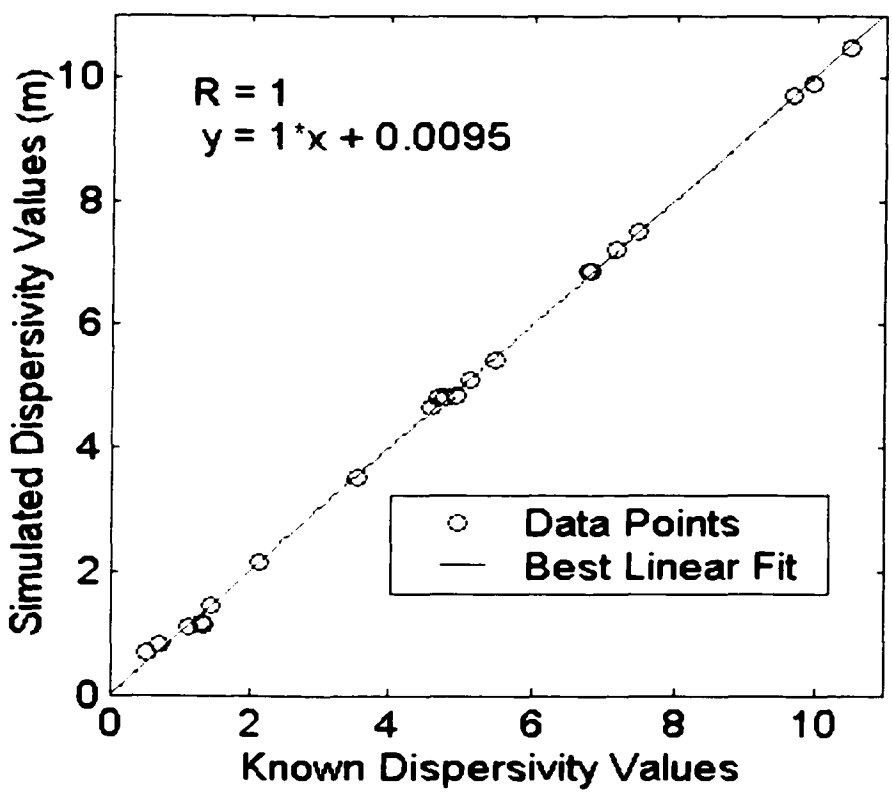


Figure 4.6 Regression analysis between the known dispersivities and the estimated values.

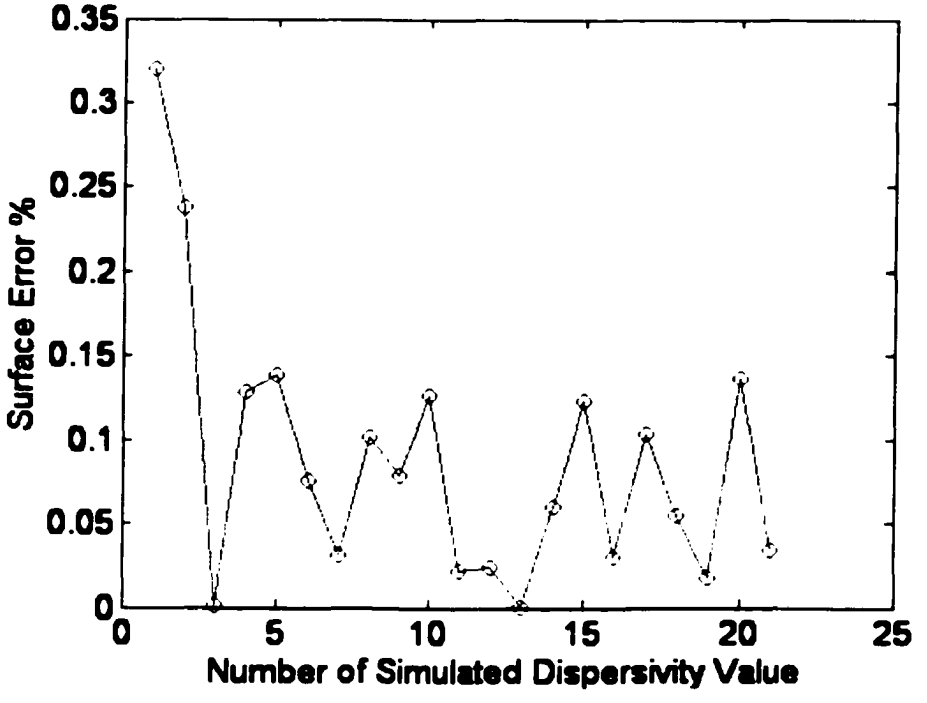


Figure 4.7 Surface error analyses between simulated and known dispersivity values.

Table 4.3 known and estimated dispersivity values (m).

No. of dispersivity value	Known	Estimated	Discrepancy
1	0.5	0.82	0.32
2	0.7	0.94	0.24
3	1.1	1.10	0.00
4	1.3	1.17	0.13
5	1.35	1.21	0.14
6	1.45	1.37	0.08
7	2.15	2.18	0.03
8	3.55	3.65	0.10
9	4.55	4.47	0.08
10	4.65	4.52	0.13
11	4.75	4.73	0.02
12	4.9	4.92	0.02
13	5.1	5.10	0.00
14	5.45	5.51	0.06
15	6.75	6.87	0.12
16	6.8	6.77	0.03
17	7.15	7.05	0.10
18	7.45	7.51	0.06
19	9.65	9.63	0.02
20	9.95	9.81	0.14
21	10.5	10.48	0.02

Chapter 5

Discussion

It has been shown that the neural network can be successfully trained to estimate the dispersivity values with high accuracy from examples of the plume distribution of different dispersivity values of the aquifer. For any inverse technique the following questions arise: does the solution exist, is it unique, and is it stable? In other words, is the solution well posed? Due to the nature of the neural network architecture other questions arise, such as whether the network can estimate values beyond the values used in the training set, and whether the network can accurately estimate parameter values with a smaller number of measurement. These questions will be addressed in this chapter.

5.1 Effect of the Grid Size

The goal in this section is to investigate what size of discretization of the physical domain the neural network can estimates dispersivity values for. This question stems from the fact that it is very costly to have a large number of measurement points in the field, as for example in the trained 40x40 network. In order to study the effect of problem complexity on the behavior of the network in estimating dispersivity values, several discretizations of the physical domain were used. The following discretizations were used in this study: 20x20, 15x35, 15x32, 15x17, 10x32, 10x22, 10x21, 10x20, 10x19, and 10x18 cells. In all of these new models the dimensions of the head boundaries were maintained the same in the Y direction. The forward model MT3D was

run to generate the same number of outputs (210) as for the 40x40 discretization. The 10 networks were trained using the same parameters and percentages of the training, validation, testing, and new known data as for the 40x40 network. The networks is number differed in the number of cells or inputs in the input layer, and their weights adjusted by the network. The output layer of each network had the same number of outputs, which was one.

Table 5.1 represents average surface error percentage values obtained from training the networks of 10x21, 10x20, 10x19, and 10x18 cells (input from hereafter), with varying numbers of neurons in the hidden layer. The table also presents the best value of surface error percentage for networks of 20x20, 15x35, 15x32, 15x17, 10x32, and 10x22 input in the input layer and only 20 neurons in the hidden layer. The average surface error percentages for networks of 10x21, 10x20, 10x19, and 10x18 input are around 26.52 percent, while for networks having 10x22 input and greater the surface error percentage is around 1.49 percent.

Table 5.1 Average surface error percentage for the different discretizations

Grid Size	Number of Neurons				
	20	50	100	150	200
20x20	1.28				
15x35	1.56				
15x32	1.49				
15x17	1.77				
10x32	1.46				
10x22	1.38				
10x21	26.78	30.44	29.18	34.39	44.37
10x20	27.46	33.35	47.09	68.47	57.21
10x19	26.36	25.56	55.81	43.30	62.64
10x18	25.51	45.74	25.17	35.08	38.22

Figure 5.1, 5.2, and 5.3 present scatter diagrams of the known and estimated dispersivity values, and surface error percentages for networks of 10x22, 15x17, and 20x20 input respectively. The figures show a good fit between the estimated and the known data, and a small surface error (Table 5.1). In general, however, networks with less than 10x22 input failed to estimate the dispersivity values, even after increasing the number of neurons in the hidden layer.

The results shown in Figures 5.1, 5.2, and 5.3 cannot explain why networks with 10x21 input and less failed in estimating the dispersivity values.

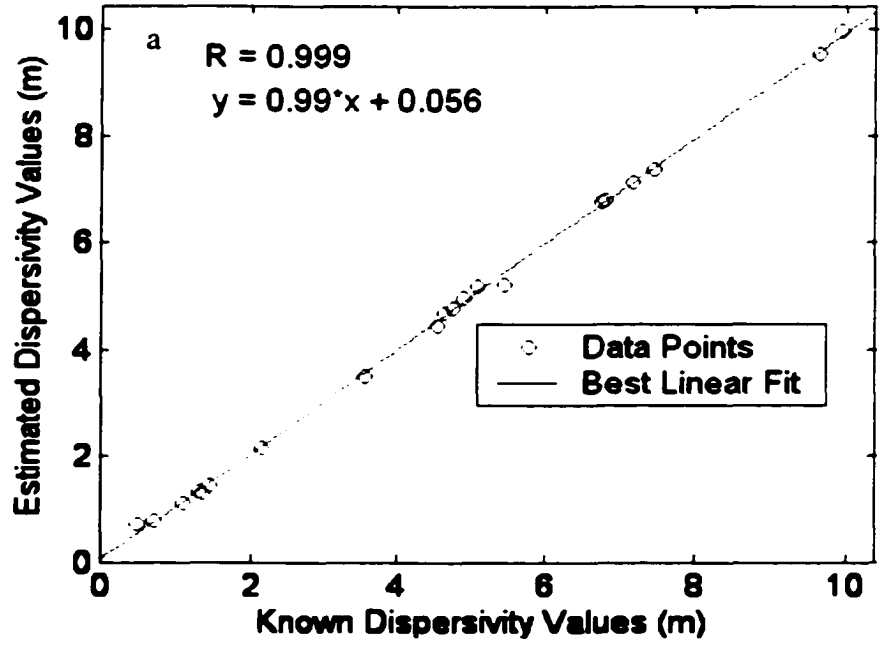


Figure 5.1 (a) NN simulation versus known dispersivity values of network of 10x22 input.

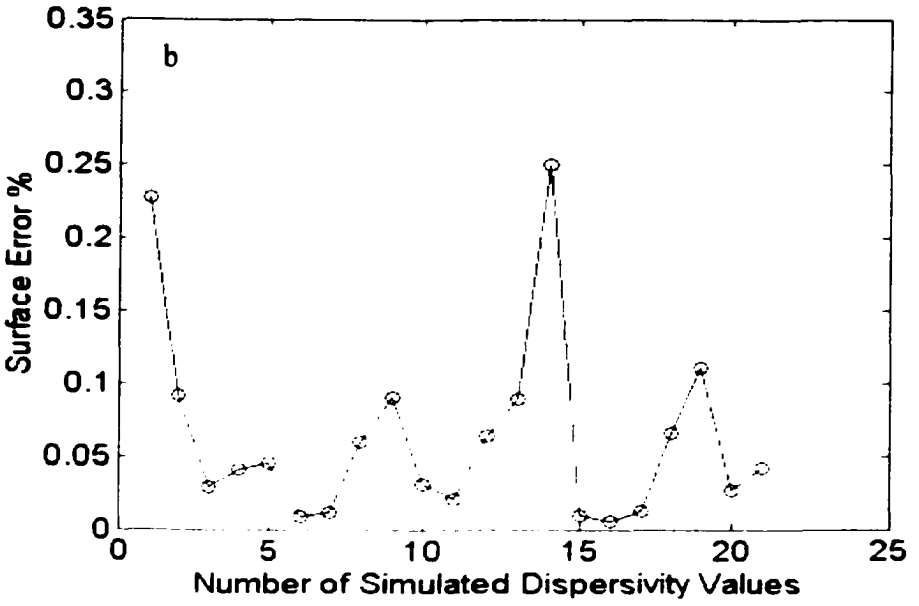


Figure 5.1 (b) Surface error percentage of network of 10x22 input.

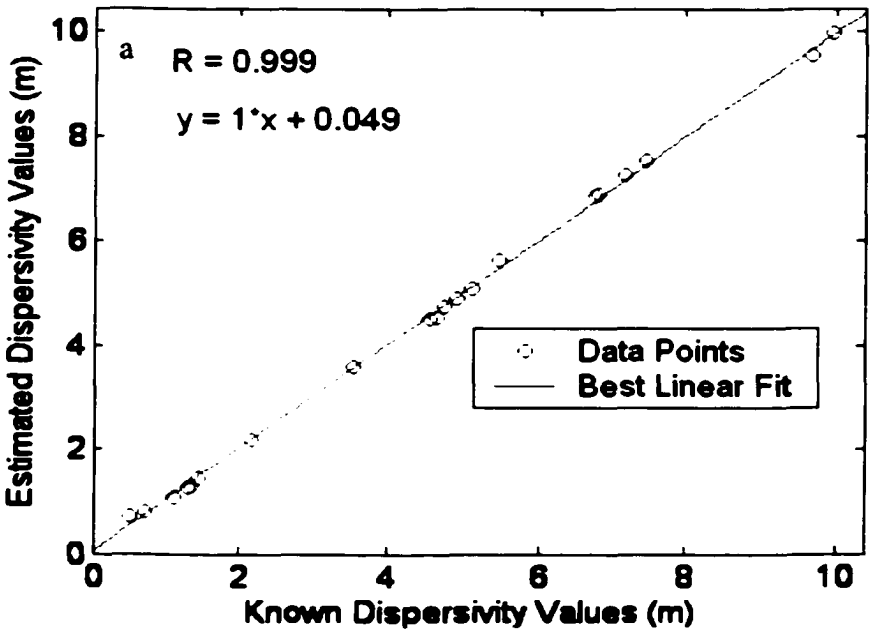


Figure 5.2 (a) NN simulation versus known dispersivity values of network of 15x17 input.

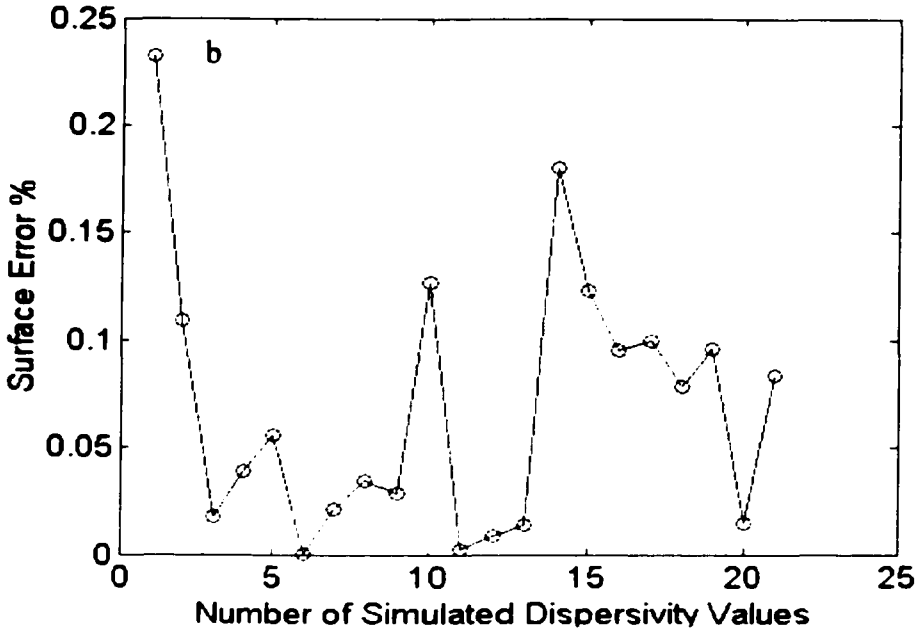


Figure 5.2 (b) Surface error percentage of network of 15x17 input.

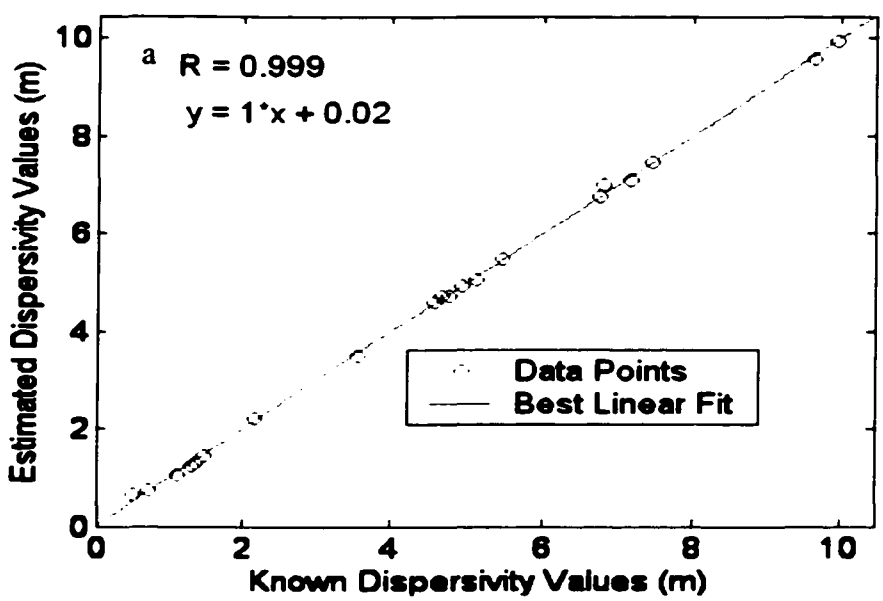


Figure 5.3 (a) NN simulation versus known dispersivity values of network of 20x20 input.

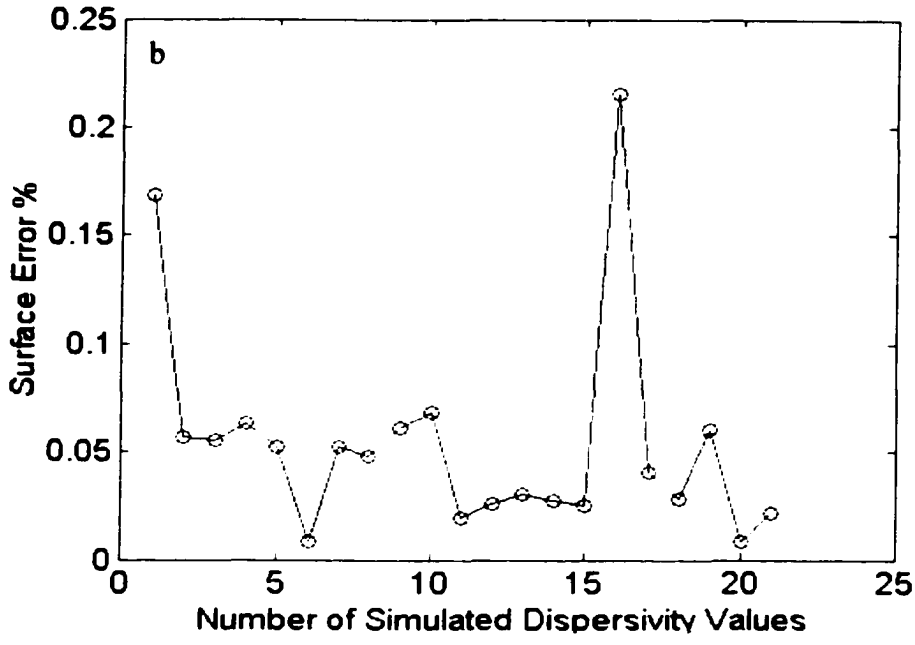


Figure 5.3 (b) Surface error percentage of network of 20x20 input.

During the training process of those networks which failed to generalize their behavior, the same shape of surface error percentage persisted with each run of training and simulation. Figure 5.4 represents the surface error percentage of a network of 10x21 input. The figure shows a high surface error between dispersivity number 3 and 5 (values less than 1.3 m; see Table 4.3). Table 5.1 shows that the average surface error for this network was about 26.78. This shape indicates that the network and other networks of low numbers of inputs have difficulty with dispersivity values between number 3 and 5.

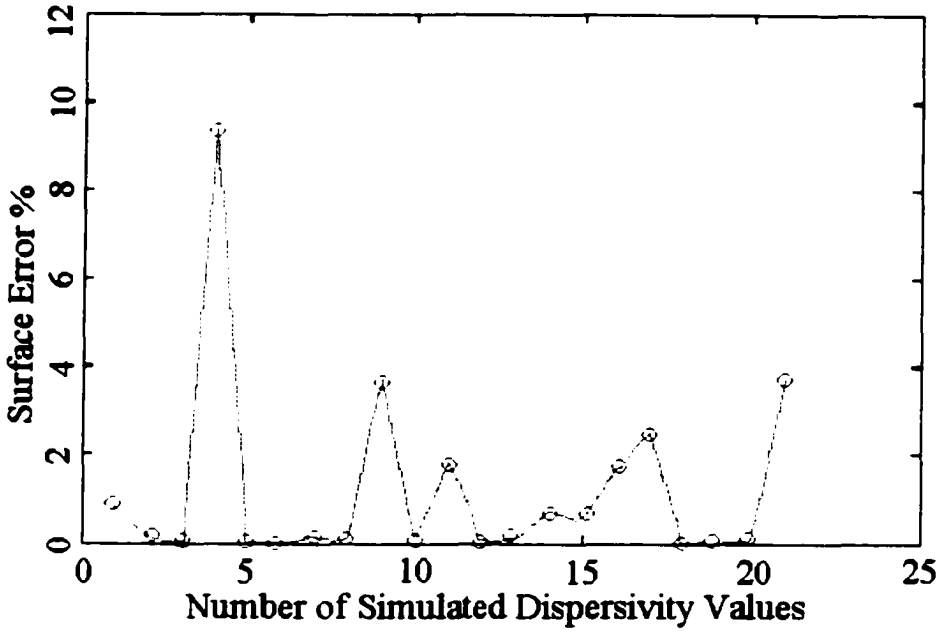


Figure 5.4 Surface error percentages of network of 10x21 input.

To test the effect of low dispersivity values on the generalizing behavior of this network of 10x21 input, two sets of concentration fields were generated. The first set excluded dispersivity values between 0.5 m and 1.25 m, and the second set excluded dispersivity values between 0.5 m and 1.0 m. The network was trained and simulated to predict the known values. Figure 5.5 is a scatter diagram of the results of this data after removing dispersivity values less than 1.3m. The figure shows a good fit between the estimated and known values where the correlation coefficient is 0.999 and the slope is 1.0. Figure 5.6 shows a surface error percentage of 2.07 percent, which is higher than that for the network of 10x22 input (Table 5.1), but in general the network changed its generalization behavior from 26.78 percent surface error to this small value. With the second set, the network again achieved a good estimation of the values. Figure 5.7 is a

scatter diagram of the results with the second set, which have a correlation coefficient of 0.999 and a slope of 1.0. Figure 5.8 shows a surface error percentage of 1.87 for the same data.

The effect of removing small values was also investigated with a network of 10x18 input, where dispersivity values less than 1.0 m were removed from the training and known values set. Figure 5.9 is a scatter diagram for the results with this network, which show a correlation coefficient of 0.998 and a slope of 0.98. All networks that failed to estimate the dispersivity values changed their behavior when small values were removed from the training and known values set.

For good training, the samples should represent the problem; otherwise, the network will not be able to learn the general trend in the problem in hand, and will not be able to arrive at a decision. It is clear that concentration fields of small values are not representative examples for this problem; therefore, the network failed to learn from them. It is worth noting that the mixed Eulerian-Lagrangian method implemented in the MT3D transport model is free of numerical dispersion and artificial oscillation, and can handle the entire range of Peclet numbers. Also, method of characteristic is free of numerical dispersion, and the finite difference scheme used in MODFLOW is unconditionally stable (Zheng, 1990).

One of the main limitations of neural networks is that they cannot reveal any physics. In other words, they are unable to explain in a meaningful way the basic process by which they arrive at a decision.

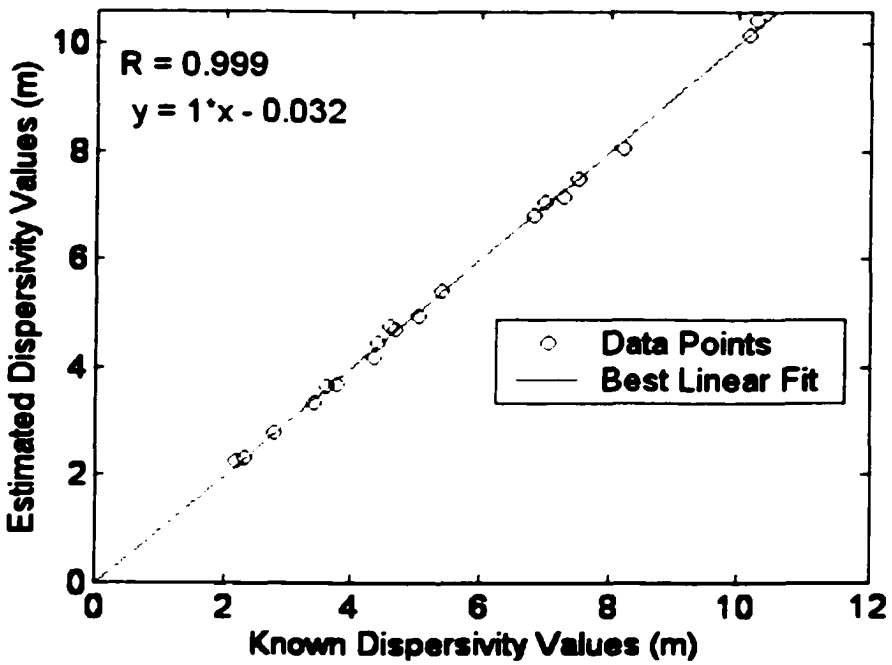


Figure 5.5 NN simulation versus known dispersivity values without dispersivity values 0.5-1.25 m.

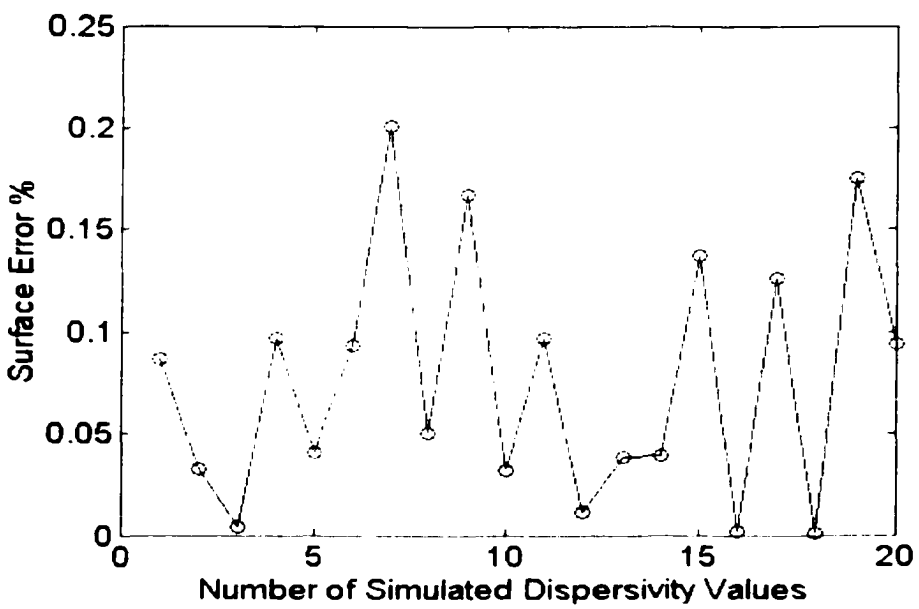


Figure 5.6 NN simulation versus known dispersivity values without dispersivity values 0.5-1.25 m.

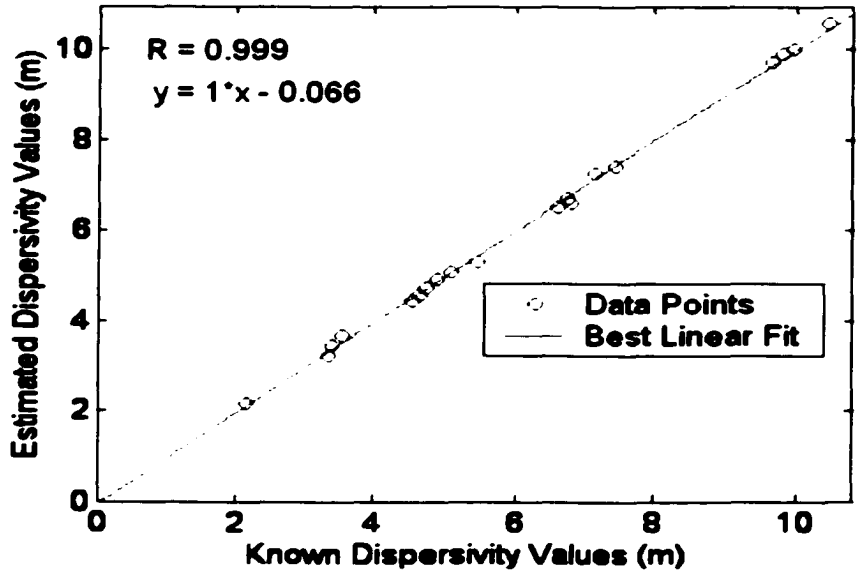


Figure 5.7 NN simulation versus known dispersivity values of network of 10x21 input without dispersivity values 0.5- 0.95 m.

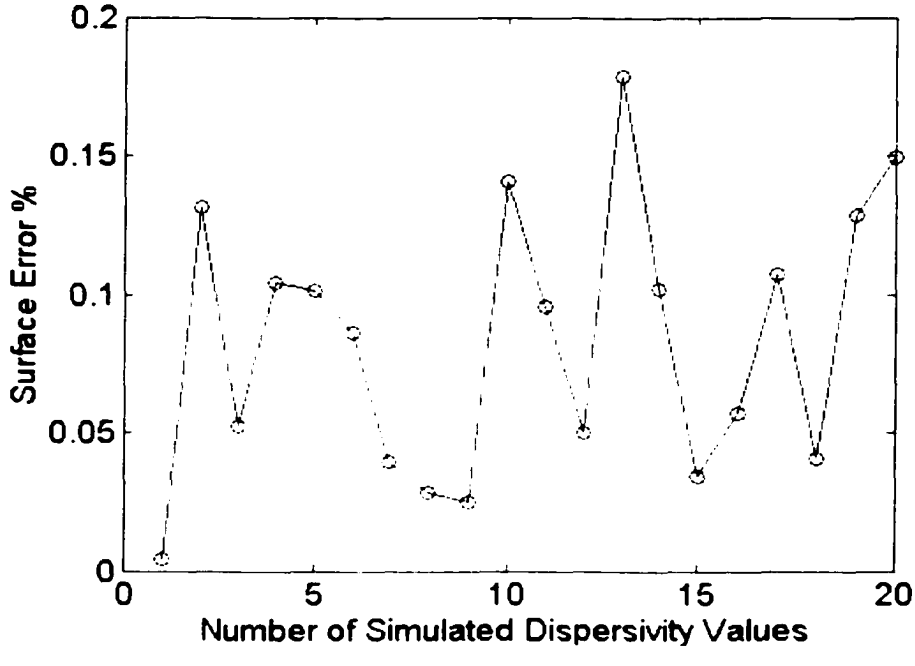


Figure 5.8 Surface error percentage of network of 10x21 input without dispersivity values 0.5-0.95 m.

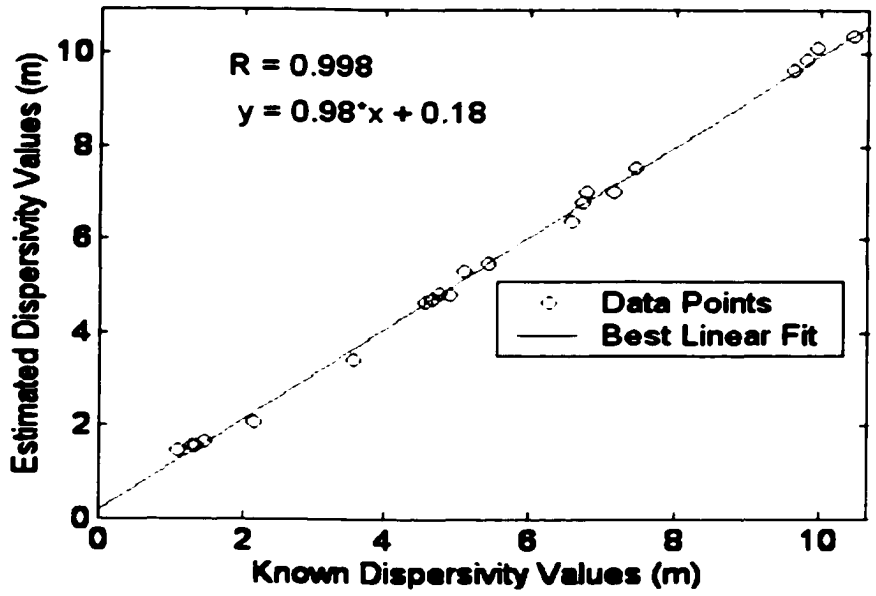


Figure 5.9 NN simulation versus known dispersivity values of network of 10x18 input without dispersivity values 0.5-0.95 m.

5.2 Training with Selected Cells

To evaluate groundwater pollution, characterize and determine solute transport parameters, and conduct field tracer tests, there is a need to monitor the solute transport in the subsurface environment with a well-designed sampling network. In some cases this condition or objective cannot be achieved.

One feature of artificial neural networks that they can reach conclusion from incomplete data (Spichak and Popva, 2000). The MATLAB script used in this study (see Appendix A) offers help in selecting any number of cells from the domain to be used in training and estimating. Figure 5.10 is a scatter diagram of a network of 10x22 neurons using cells numbers 11-150 for training and estimating. The figure shows that with this number of cells the network was able to estimate the dispersivity values with good

accuracy. The correlation coefficient was 0.964 with a slope of 0.82. Figure 5.11 represents the surface error percentage, which was 1.87. This method may be practically helpful in cases where the plume cannot be sampled completely, or the sampling network is not encompassing the whole dimension of the plume.

It is worth noting that the network of 10x21 input had 210 input, which was more than the number of input (140 input) selected from the network of 10x22. The 10x21 network neurons failed to estimate the dispersivity values despite its larger number of input. It is not the number of neurons in the network that forces the network to achieve at least acceptable estimation of the parameter. It seems that the quality of the data presented to the network is the main factor in forcing the network to reach the desired estimation. This could be an indication of the quality of the numerical solution used to generate the forward data in the 10x21 network.

Most studies of the solute transport inverse problem have chosen the normal distribution of the concentration residuals. The log normal distribution is usually discarded because it places too emphasis on low concentration data, besides its of inapplicability to zero measurement (Medina and Carrera, 1996). The MATLAB script used here to train the network and estimate the dispersivity value (Appendix A) can help in discarding some of the small values and selecting different sizes from the entire input and output data. In this way, the network can estimate the dispersivity values without the need to transform the data to log normal or fill measurement gaps with zero values and interpolate the rest of the non-sampled locations.

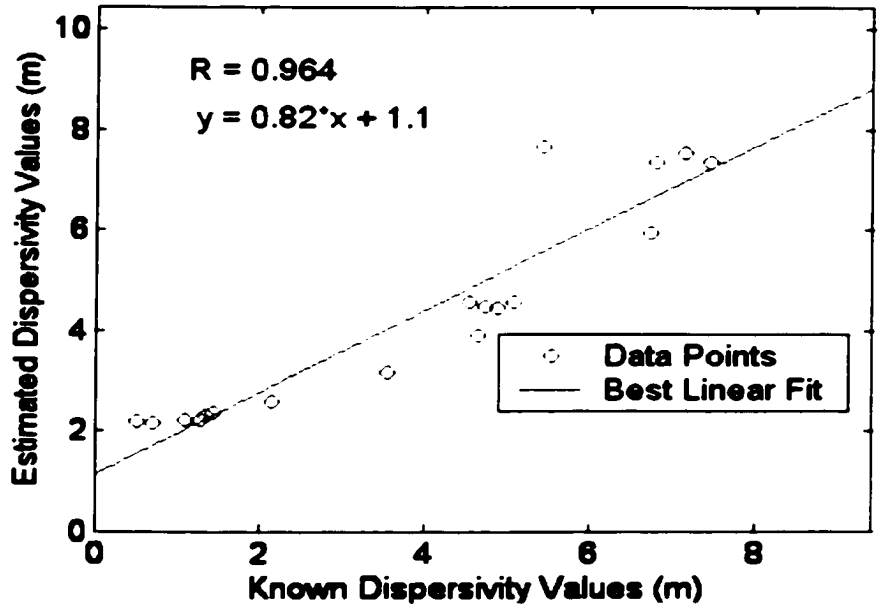


Figure 5.10 NN simulation versus known dispersivity values using cells 11-150.

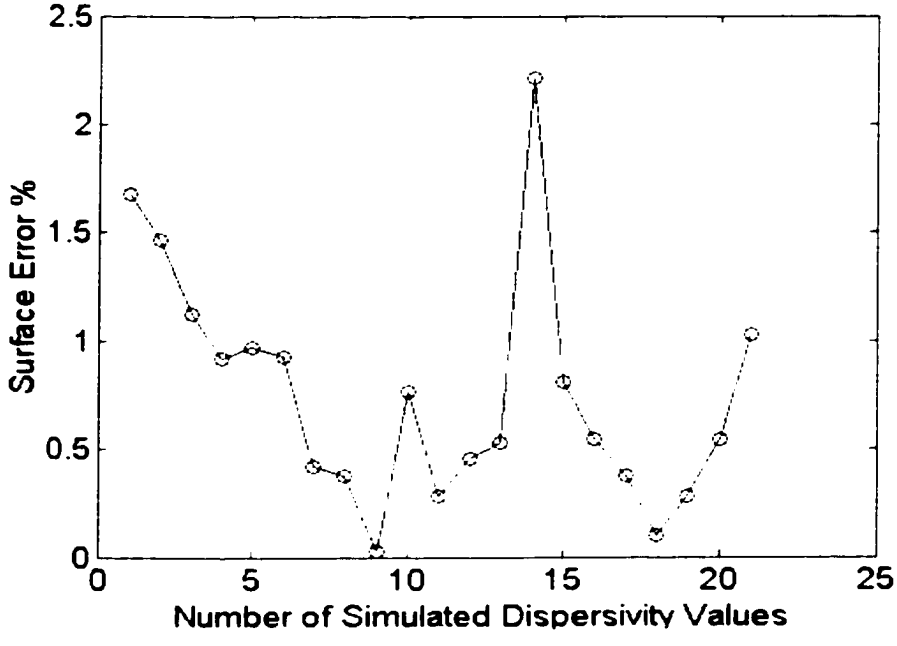


Figure 5.11 Surface error percentage of network of 10x22 input using cells 11-150.

5.3 Estimation of Dispersivity Values with Different Sampling Networks

The real case monitoring of solute transport problems will not use as many sampling points as the 40x40 network (1600 sampling points). In order to investigate the ability of the ANN to estimate the dispersivity values with fewer measurements, similar to real field monitoring, several new networks were used. The inputs for the different networks consisted of the following numbers of measurement points: 199, 146, 98, and 73. These numbers of measurements correspond to different designs of cell selection. The first number (199) refers to a design in which the fifth and subsequent concentration values were selected, starting from cell number 50 at the domain, in order to avoid selecting the concentration values in the constant head boundaries. The other numbers of measurements were selected in the same manner, except that the selected cells were the 10^{th}, 15^{th}, and 20^{th}. Figure 5.12 represents the selected concentration cells with the different designs.

A second design for the ANN was tested, in which the 11^{th} cell and subsequent cells were selected, starting from cell number 50 and cell number 561 at the domain. The numbers of concentration measurements were 133 and 87, respectively.

Figure 5.13 is a scatter diagram showing the results obtained by selecting the 5^{th} and subsequent concentration values. The network was able to estimate the dispersivity values with 199 measurements. The correlation coefficient was 0.998. With further reduction in the number of concentration values presented, the network was still able to estimate the dispersivity values. Figures 5.14, 5.15, and 5.16 are scatter diagrams showing the results obtained with 144, 98, and 73 measurements, respectively. The correlation coefficients were 0.999, 0.997, and 0.985, respectively.

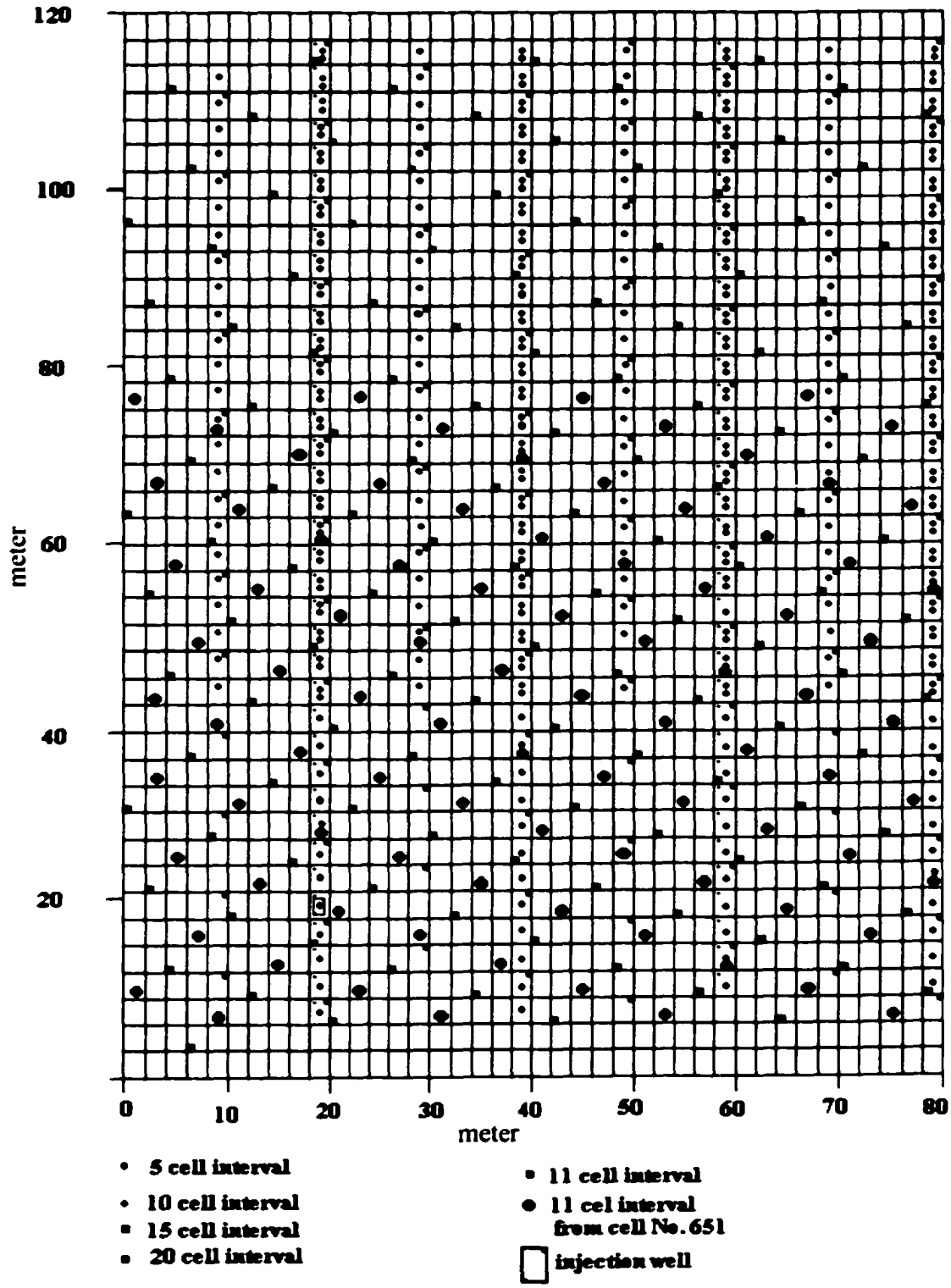


Figure 5.12 Selected concentration cells with different sampling networks.

Using the second realistic design, the network was able to estimate the dispersivity values using 133 measurements as in the case of selecting the concentration value at cell 11 and subsequent cells. Figure 5.17 shows a correlation coefficient 0.999, which indicates an excellent estimation of the dispersivity values. Also, with further reduction in the number of measurements from 133 to 87, the network was able to estimate the dispersivity values with a correlation coefficient 0.998 (Figure 5.18).

The results indicate that the neural network was able to estimate the dispersivity values by using all the concentration values obtained from the MT3D model, or by selecting some portion of the domain, or even by selecting some concentration measurements from the domain. The latter conclusion may be of great practical significance, since it is very costly to have 1600 measuring points in the field.

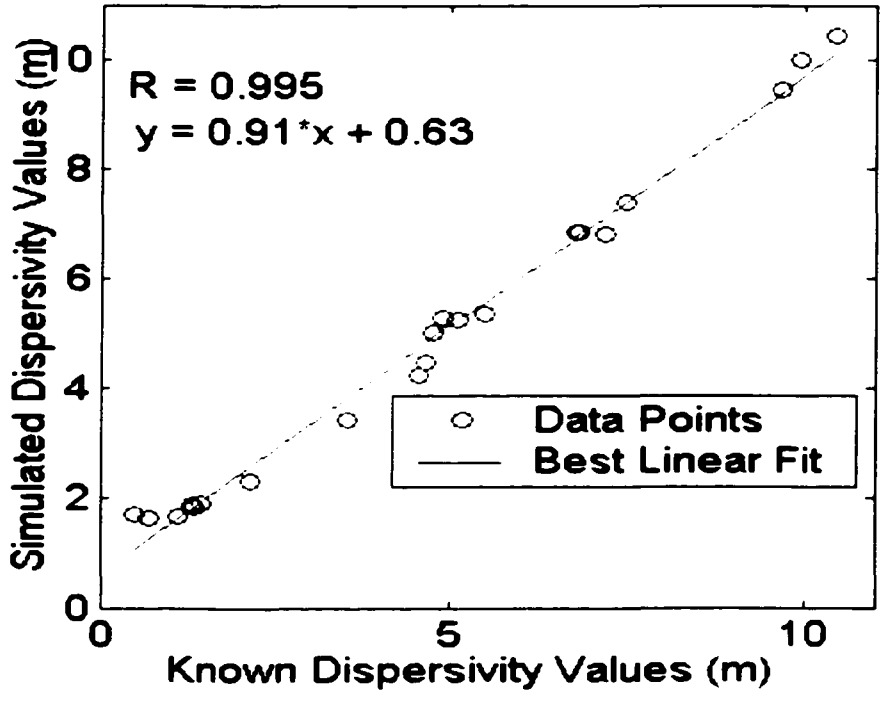


Figure 5.13 Simulated versus known dispersivity values with 199 measurements (taken from cell 50 at 5-cell intervals)

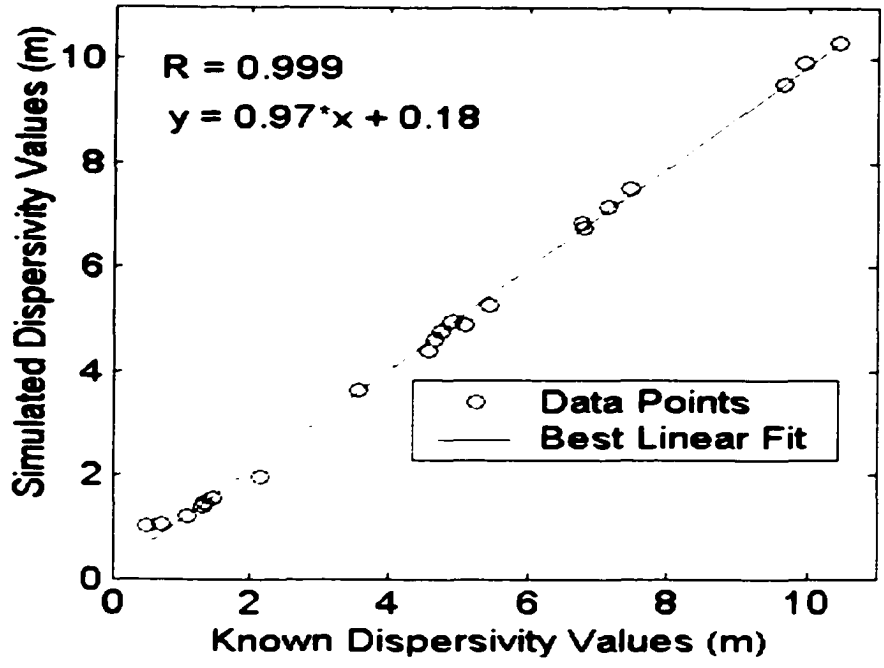


Figure 5.14 Simulated versus known dispersivity values with 146 measurements (taken from cell 50 at 10-cell intervals).

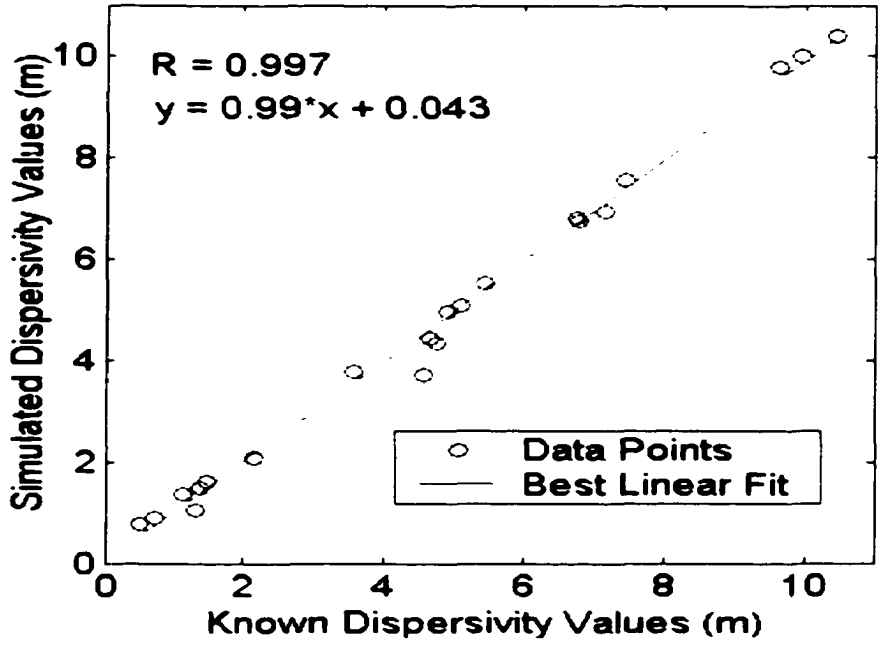


Figure 5.15 Simulated versus known dispersivity values with 98 measurements (taken at cell 50 at 15-cell intervals).

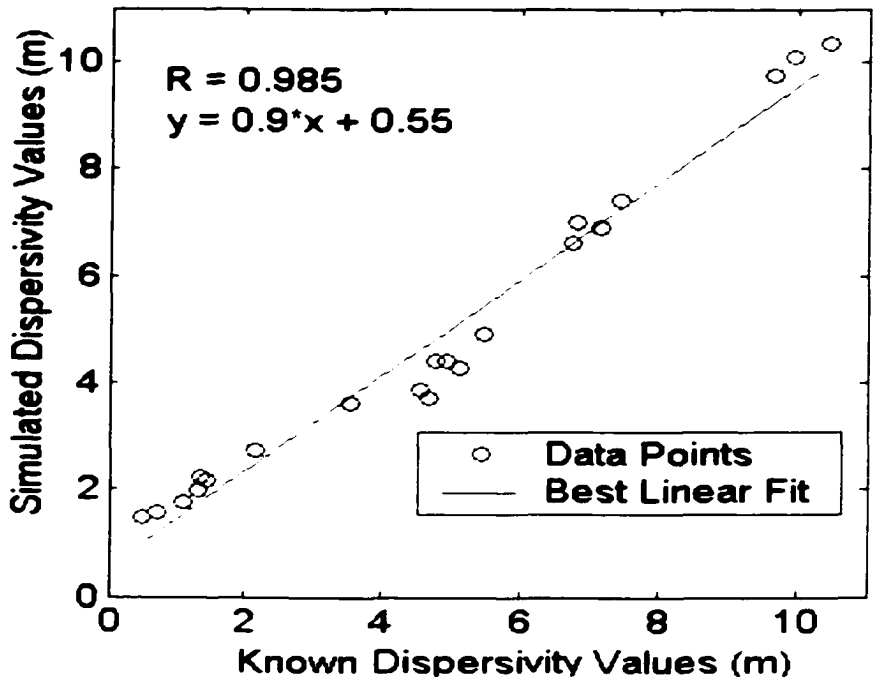


Figure 5.16 Simulated versus known dispersivity values with 73 measurements (taken at cell 50 and at 20-cell intervals).

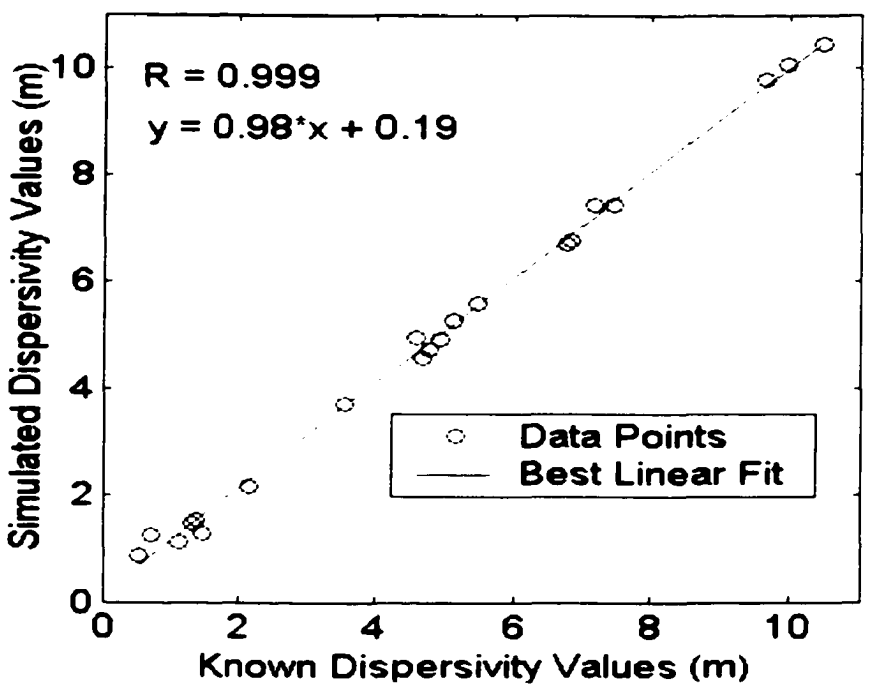


Figure 5.17 Simulated versus known dispersivity values with 133 measurements (taken at cell 50 and at 11-cell intervals).

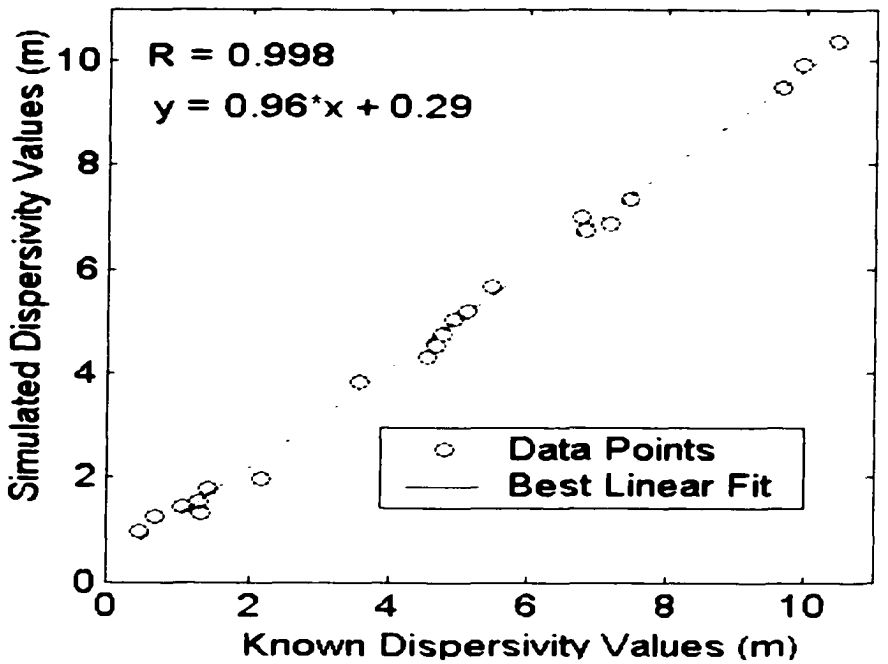


Figure 5.18 Simulated versus known dispersivity values with 87 measurements (taken at cell 651 and at 11-cell intervals).

5.3 Estimation of Dispersivity Values Beyond the Range Used in Training

The previous results have shown that the neural network can be used to estimate dispersivity values from the concentration fields. The inverse problem was used to estimate the known data, which were prepared to be within the range of the data used for training the network of 40x40 neurons (Table 4.3). The question that arises is: can the network generalize well in new data beyond the range of values included in the calibration range? To answer this question, a new set of 21 dispersivity values in the range between 11 m and 20 m with an increment of 0.45 m was generated using the

MATLAB package. The concentration fields of the dispersivity were generated by the modified MT3D model, and were used as new inputs for the network to estimate their dispersivity values.

These values of dispersivities were between about 0.12 and 0.22 of the scale of the experiment, where the scale in this experiment can be defined as the distance traveled from the source. Lallemand-Barres and Peaudecerf (1978) found that longitudinal dispersivity could be 0.1 of the scale, and this has been adopted as a general rule of thumb. Gelhar et al. (1992) state that at a given scale the longitudinal dispersivity ranges over 2-3 orders of magnitude, and there is no clear pattern regarding scale dependency of dispersivity.

The proposed dispersivity values (11-20 m) will be larger than the 0.1 rule of thumb, varying by a small fraction to double that value; according to Gelhar et al. (1992), the values will be acceptable.

Figure 5.19 is a scatter diagram showing the estimated and the known dispersivity values for dispersivity values beyond the range used in training the 40x40 network. The figure shows a correlation coefficient of 0.993 and a slope of 1.0. These results indicate a good estimation of the dispersivity values beyond the range used in training the network.

Figure 5.20 represents surface error percentages for the dispersivity values in range 11 m to 20 m. The figure shows no trend for the error between the estimated and the known values. The error is oscillating without a clear trend.

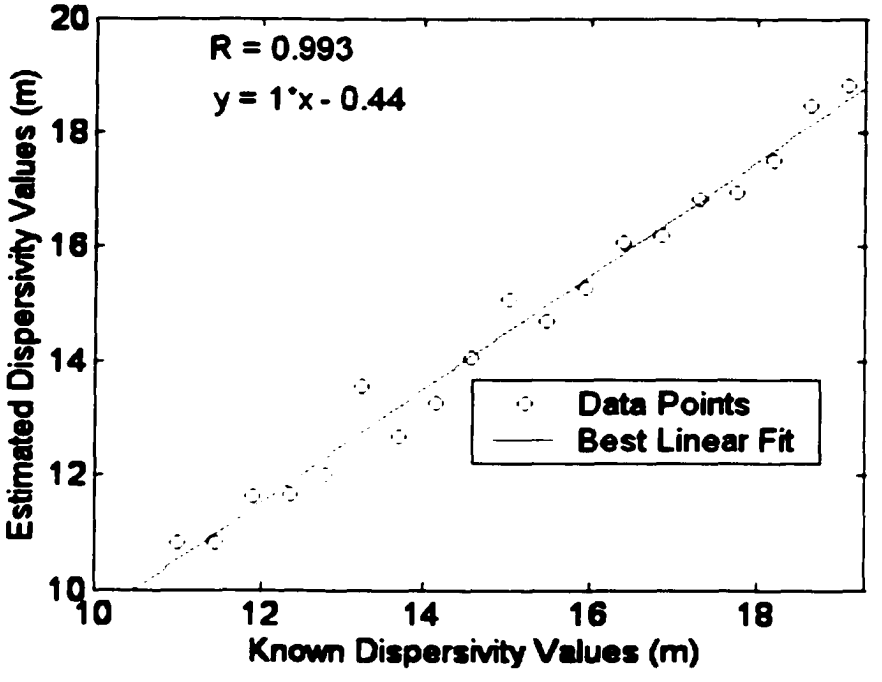


Figure 5.19 NN simulation versus known dispersivity values for the range 11-20 m.

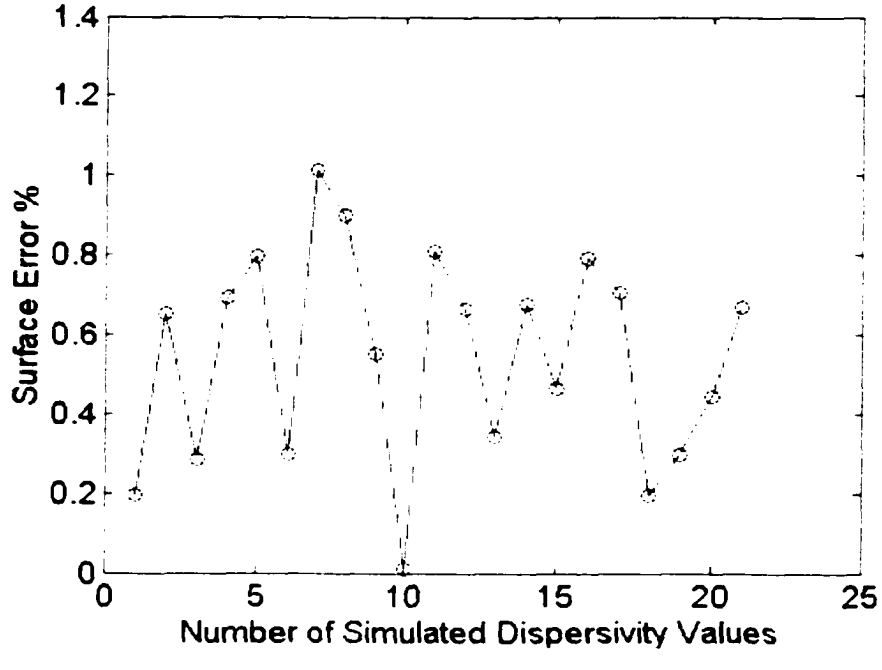


Figure 5.20 Surface error percentage for the range 11-20 m dispersivity values.

The capability of the network to generalize its behavior with values of dispersivity beyond the range used in training was also investigated with different sizes of discretization of the domain. The network was tested to estimate the dispersivity values with 10x22 grids. In this network the size of the constant head grid was maintained constant in the Y direction, as in the previous discretization. Figures 5.21 is a scatter diagram showing the estimated and the known dispersivity values. The figure shows a correlation coefficient of 0.99 and a slope of 0.91. The result indicated that the ANN was able to generalize its behavior beyond the training range with a different discretization level of the domain.

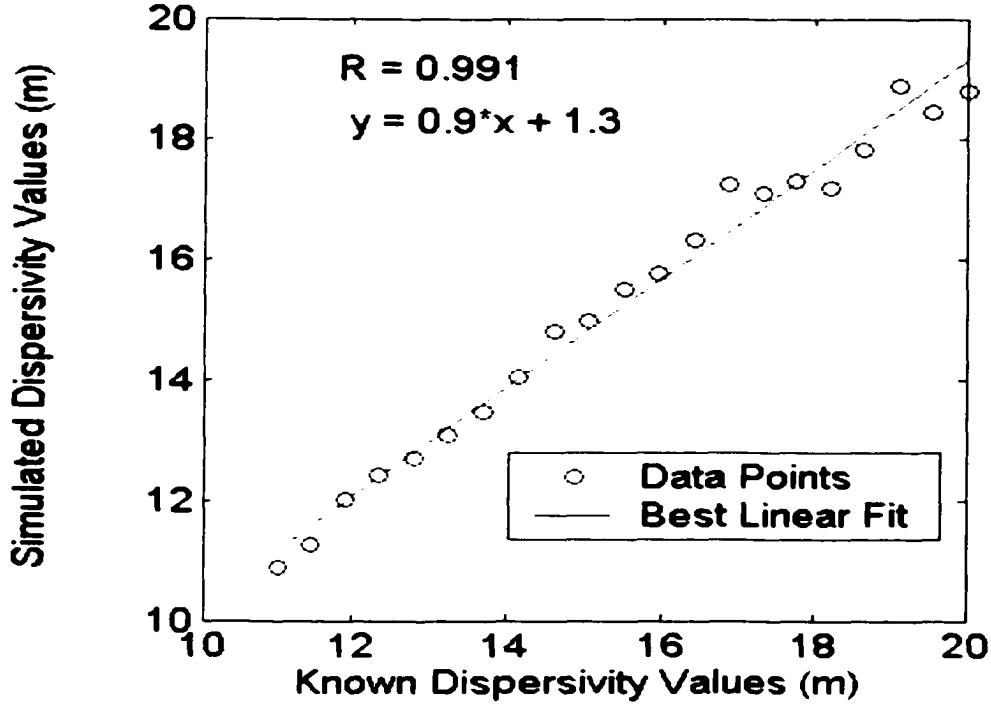


Figure 5.21 NN simulation versus known dispersivity values of the range 11-20m for network 10x22 inputs.

5.4 Uniqueness

The solution is defined as non-unique if more than one solution is found. It is well known that the inverse solution in groundwater modeling is often non-unique. However, the non-uniqueness problem may be removed in some cases by additional information or appropriate limitation of unknown parameters. The non-uniqueness of the inverse solution strongly displays itself in the indirect method through the existence of many local minimizers (Sun, 1994).

The neural network technique is an indirect inverse method. The question of uniqueness could be investigated by increasing the number of measurements, and by changing the number of patterns presented to the network. Table 5.1 shows that as the number of measurements increased in the domain from a network of 10x21 inputs to a network of 10x22 inputs, the neural network was able to obtain accurate solution, however, the solution was different for each network (different surface error percentage). It was demonstrated that removing concentration fields related to small dispersivity values enabled the networks that had previously failed in generalizing their estimates to obtain a solution. This may be understood as the removal of very noisy data or unrepresentative examples of the problem, which prevent an accurate solution existing. However, existence cannot be considered a major difficulty of inverse problems, because the physical reality may be considered at least as an approximate solution (Sun, 1994).

For the second case, changing the number of patterns presented to the network, a review of table 4.2 it shows that for each number of patterns the network produced a different solution, which again indicates a non-unique solution. Changing the number of patterns effects the generalization behavior of the network, and in some problems a small

number of training sets will not guarantee a solution (Mehrotra et al., 1997). In other words, testing for the minimum number of training example is the same as testing for the existence of a solution. Table 4.2 shows that a solution can exist with at least 126 patterns. The network cannot guarantee a unique solution with each run, with different discretizations of the domain, and with different patterns, but the solution of each run, discretization or pattern can be considered a unique solution in itself.

5.5 Stability

Stability is a crucial condition for well-posed problems, and a solution cannot be accepted if it is unstable, even if it exists and is unique. The stability condition ensures that the inverse solution is physically meaningful. It is well known that forward solutions in groundwater modeling are always stable, and inverse solutions are often unstable. The instability problem of the inverse solution may be due to two causes: the way the problem is posed, and inherent capability of the inverse technique (Neuman, 1973).

To investigate the instability of the inverse solution due to the inherent nature of the technique, 10, 20, 30, and 40 percent random noise with mean of 1.0 and zero variance was added to the concentration fields of the known dispersivities for the network of 40x40 inputs. In neural networks noise can be added to the components of the training or testing sets. Adding noise to the training set leads to improvement in the generalization behavior of the network, especially when the training set is small. Adding noise to the testing set has a very different effect. It presents to the network new data with different degrees of error, and the network seeks the correct estimation of the corresponding dispersivity values related to these noisy concentration fields.

The goal of the investigation was to discover whether the error in the concentration values would lead to large fluctuations in the solution of the network. Figures 5.22 through 5.25 represent a first order polynomial fit between the noisy and noise-free data. The polynomial fit is calculated as a least squares fit. The Figures show an excellent fit between the noise-free and noisy data, with a correlation coefficient of 0.999 for the 10 and 20 percent noise in the data. The correlation coefficient of 30 and 40 percent noise was 0.996, which was less than that for the previous percentages of error, but still indicates a good estimation of the dispersivity values even with higher error in the measurements. These results indicated that the solution of the neural network is stable with as high as 40 percent error in the new measurements presented to the network to estimate the dispersivity values.

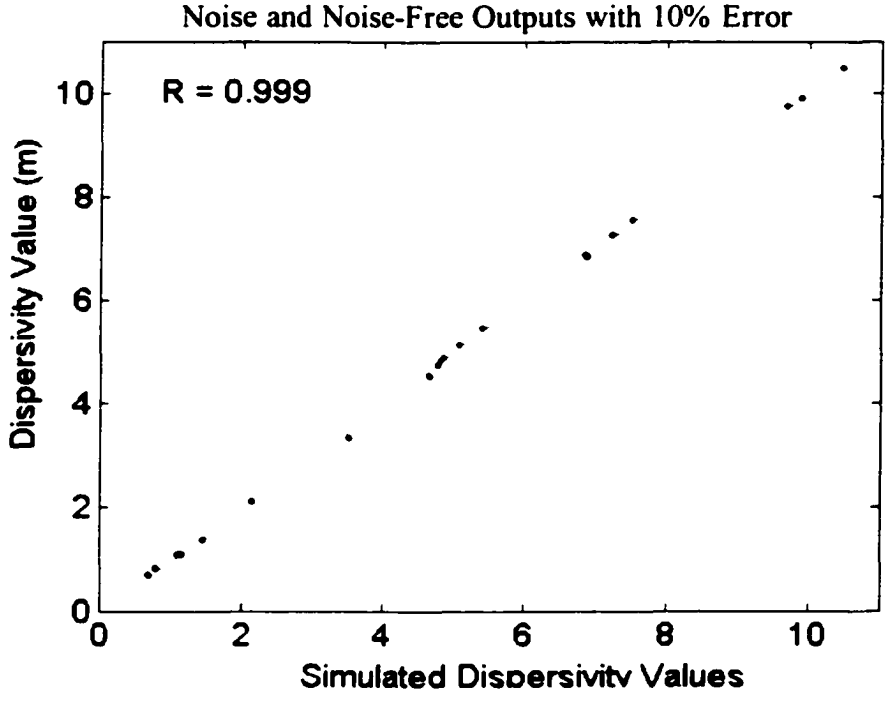


Figure 5.22 NN simulation versus 10 percent noisy known dispersivity values.

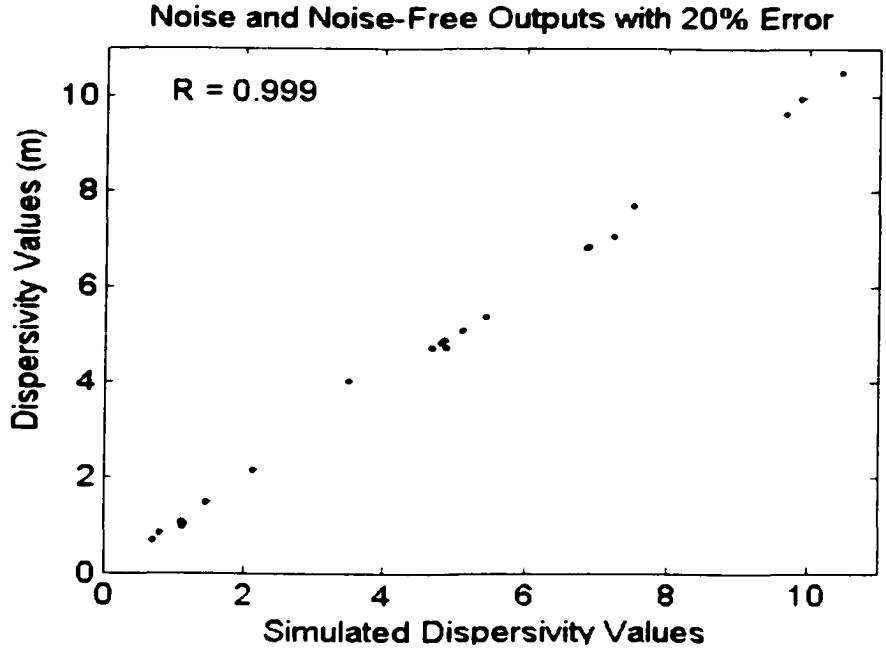


Figure 5.23 NN simulation versus 20 percent noisy known dispersivity values.

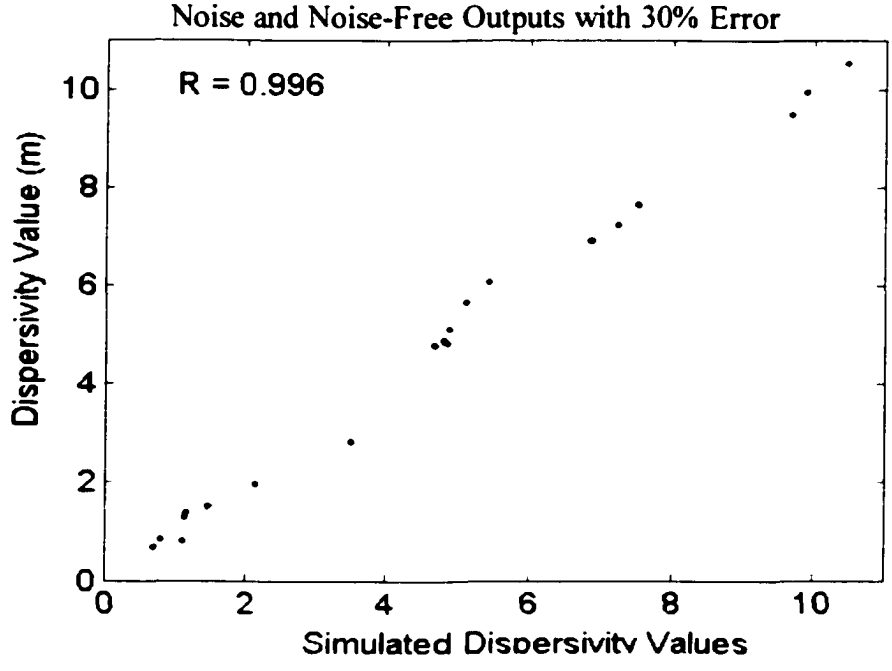


Figure 5.24 NN simulation versus 30 percent noisy known dispersivity values.

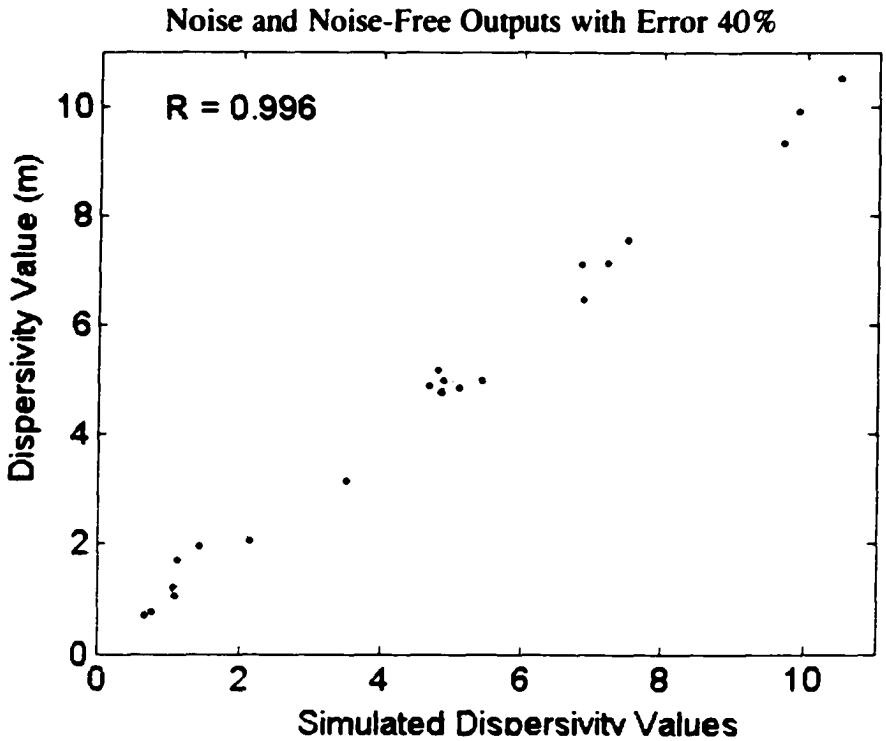


Figure 5.25 NN simulation versus 40 percent noisy known dispersivity values.

Chapter 6

CONCLUSION AND RECOMMENDATION

6.1 Conclusion

This study has demonstrated that a feed-forward backpropagation artificial neural network of one layer with 20 neurons generalizes well on the inverse problem of solute transport. The neural network was tested using synthetic data created by forward modeling for a two-dimensional solute transport model of a steady state flow in an unconfined aquifer. The method presented in this research is simple to apply and its data requirements are limited to good forward measurement of concentrations.

The performance of a network of 10x18 inputs was excellent with transport controlled by dispersion type of network. However, the method demonstrated some limitations in estimating parameter values in network of 10x21 input and less. It was shown that the method could estimate the parameter values with high accuracy by selecting various numbers of concentration values from various locations of the domain.

In this study there was no clear evidence of the dependency of the artificial neural network size on the problem to be solved. It was found that twenty neurons were enough to estimate the parameter, and the performance of the network was not stable with an increasing number of neurons. The neural network is able to extrapolate well beyond the range of dispersivity values included in the calibration range. The extrapolation of the parameter values was good, even if the parameter values are two times beyond its scale

limits. Increasing the size of the training set by including examples beyond the scale limits of the problem will enhance the generalization properties of artificial networks, and will lead to accurate estimation of the parameter.

The solution was stable with up to 40 percent error in the concentration measurements. This feature of neural networks makes them a very powerful modeling method and inversion technique. Despite that fact that the solution existed with at least 126 examples of the plume distribution in the domain, and was very stable, the inverse problem was ill-posed due to non-uniqueness. The non-uniqueness was observed with various patterns, and with various numbers of concentration fields. However, ill-posedness is not a major problem in this method, because it can be overcome by considering the best estimation achieved by the network a unique solution for that run alone.

6.2 Recommendation

In the light of the results obtained during the course of this research, there are a number of aspects of artificial neural network inverse methodology which warrant further research:

1. Since a strength of neural networks is in approximating functions of several variables, it might be possible to optimize other parameters simultaneously, such as hydraulic conductivity, dispersivity, and sorption coefficients, and to estimate parameters of different media.

2. Applicability of the methodology needs to be tested in real cases, with careful assessment of neural network solutions versus other estimation approaches.
3. There is a need to assess backpropagation network solutions versus these of other networks, such as radial-basis function networks, since they use the same equation for the fundamental solution of transport as an activation function.
4. It would be useful to estimate the dispersivity values as a function of time, either by stochastic modeling or deterministic modeling, for example as a Bayesian analysis.
5. The quality of neural networks depends entirely on the experience of the user, for example in choosing the error criteria, the number of hidden layers and neurons, and the activation functions. Future research should be guided towards an automotive approach to determining optimum artificial neural network architecture.

REFRENCES

Adams, E. E., and L. W. Gelhar, Field study of dispersion in a heterogeneous aquifer 2 spatial moments analysis, Water Resources Research, Vol. 28, No. 12, pp. 3293-3307, 1992.

Amari, S., N. Murata, K.-R. Muller, M. Finke, and H. Yang, Statistical theory of overfitting-Is cross-validation asymptotically effective?, Advances in Neural Information Processing Systems, Vol. 8, pp. 176-182, 1996.

Anderman, E. R., E. P. Poeter, and M. Hill, Presentation and evaluation of multi-stage parameter estimation method using advective transport observation, IAHS Publication, No. 237, pp.179-188, 1996.

Anderson, M. P., Movement of contaminants in groundwater: groundwater transport-advection and dispersion, In Groundwater Contamination, pp. 37-54, National Academy Press. Washington, DC, 1984.

Barth, G. R., M. C. Hill, T. H. Illangasekare, and R. Harihar, Predictive modeling of flow and transport in a two-dimensional intermediate-scale, heterogeneous porous medium, Water Resources Research, Vol. 37, No. 10, pp. 2503-2512, 2001.

Bear, J., and A. Verruijt, Modeling groundwater flow and pollution, D. Reidel Publishing Company, Dordrecht, Holland, 414 pp., 1987.

Bear, J., Dynamics of fluids in porous media, American Elsevier Publishing Company, Inc., NY, 746 pp., 1972

Boggs, J. M., S. C. Young, L. S. Beard, L.W. Gelhar, K. R. Rehfeldt, and E. E. Adams, Field study of dispersion in a heterogeneous aquifer 1 overview and site description, Water Resources Research, Vol. 28, No. 12, pp. 3281-3291, 1992.

Cannon, J. R., P. DuChaeau, and S. Moaveni, An unknown ingredient inverse problem reduced to a functional differential equation: in W. E. Fitzgibbon, and M. F. Wheeler (ed.), Computational methods in geosciences, Society for Industrial and Applied Mathematics, Philadelphia, PA, 207 pp., 1993.

Carrera, J., and S. P. Neuman, Estimation of aquifer parameters under transient and steady state conditions 1 maximum likelihood method incorporating prior information, Water Resources Research, Vol. 22, No. 2, pp. 199-210, 1986a.

Carrera, J., and S. P. Neuman, Estimation of aquifer parameters under transient and steady state conditions 3 application to synthetic and field data, Water Resources Research, Vol. 22, No. 2, pp. 228-242, 1986c.

Clair, T. A., and J. M. Ehrman, Using neural networks to assess the influence of changing seasonal climates in modifying discharge, dissolved organic carbon, and nitrogen export in eastern Canadian rivers, Water Resources Research, Vol. 34, No.3, pp. 447-455, 1998.

Colton, D., H. W. Engl, A .K. Louis, J. R. MchLaughlin, and W. Rundell, Surveys on solution methods for inverse problems, Springer, Wien, NY, 275 pp., 2000.

Coulibaly,P., F. Anctil, and B. Bobe'e, Daily reservoir inflow forecasting using artificial neural networks with stopped raining approach, Journal of Hydrology, Vol. 230, pp. 244-257, 2000.

Cybenko, G., Approximation by superpositions of a sigmoid function, Mathematics of Control Signals and Systems, Vol. 2, pp. 303-314, 1989.

Dawson, C. W., and R. L. Wilby, Hydrological modelling using artificial neural networks, Progress in Physical Geography, Vol. 25, No. 1, pp. 80-108, 2001.

Draper, N. R., and H. Smith, Applied Regression Analysis, John Wiley, NY, 709 pp. 1981.

El-Serafy, G., A. Heemnk, C. T. Stroet, and F. V. Geer, Parameter identification in particle models for groundwater contamination using the adjoint method, IAHS Publication, No. 237, pp. 231-240, 1996.

Fausett, L., Fundamentals of neural networks architecture algorithms and applications, Prentice Hall, NJ, 461 pp., 1994.

Gautam, M. R., K. Watanabe, and H. Saegusa, Runoff analysis in humid forest catchment with artificial neural network, Journal of Hydrology, Vol. 235, pp. 117-136, 2000.

Gelhar, W. G., C. Welty, and K. R. Rehfeldt, A critical review of data on field-scale dispersion in aquifers, Water Resources Research, Vol. 28, No. 7, pp. 1955-1974, 1992.

Gill, P. D., W. Murray, M. A. Saunders, and M. H. Wright, Users guide for NPSOL: a FORTRAN package for non-linear programming, Technical Report SOL 86-2, Stanford University, Stanford, CA.,1986.

Gorelick, S. M., and I. Remson, Optimal dynamic management of groundwater pollution source, Water Resources Research, Vol. 18, No. 1, pp.71-76, 1982.

Gorelick, S. M., B. Evans, and I. Remson, Identifying source of groundwater pollution an optimization approach, Water Resources Research, Vol. 19, No. 3, pp. 779-790, 1983.

Govindaraju, R. S., Artificial neural networks in hydrology I preliminary concepts, Journal of Hydrologic Engineering, Vol. 5, No. 2, pp. 115-123, 2000 a.

Govindaraju, R.S., Artificial neural networks in hydrology II hydrologic applications, Journal of Hydrologic Engineering, Vol. 5, No. 2, pp. 124-137, 2000 b.

Guadagnini, A., and S. P. Neuman, Recursive conditional moment equations for advective transport in randomly heterogeneous velocity fields, pp. 37-67. In Berkowitz B. (ed.), Dispersion in heterogeneous geological formations, Kluwer Academic Publishers, MA, 263 pp., 2001.

Gutjahr, A. L., Hydrology, pp. 75-97. In Hunter, R. L., and C. J. Mann (ed.), Techniques for determining probabilities of geologic evens and processes, Oxford University Press, NY, 364 pp., 1992.

Hadamard, J., Lectures on Cauch's problem in Linear Partial Differential Equations, Dover Publication, NY, 316 pp., 1932.

Hassan, E. A., and K. H. Hamed, Predicting of plume migration in heterogeneous media using artificial neural networks, Water Resources Research, Vol. 37, No. 3, pp. 605-623, 2001.

Haykin, S., Neural networks a comprehensive foundation, Prentice-Hall, Inc., NJ, 842 pp., 1999.

Hornik, K., M. Stinchcombe, and H. White, Multilayer feedforward networks are universal approximators, Neural Networks, Vol. 2, No. 5, pp. 359-366, 1989.

Hush, D. R., and B. G. Horne, Progress in supervised neural networks: What's new since Lippmann?, IEEE Signal Processing Magazine, Vol. 10, pp. 8-39, 1993.

Hyun, Y., and K.-Kun Lee, Model identification criteria for inverse estimation of hydraulic parameters, Ground Water, Vol. 36, No. 2, pp. 230-239, 1998.

Imrie, C. E., S. Durucan, and A. Korre, River flow prediction using artificial neural networks: generalization beyond the calibration range, Journal of Hydrology, Vol. 233, pp. 138-153, 2000.

Javandel, I., C. Doughty, and C. F. Tasng, Groundwater transport handbook of mathematical models, American Geophysical Union Water Resources Monogram 10, Washington, DC, 228 pp., 1984.

Kauffmann, C., W. Kinzelbach, and J. J. Fried, Simultaneous calibration of flow and transport models and optimization of remediation measures, IAHS Publication, No. 195, pp. 159-170, 1990.

Keidser A., and D. Rosbjerg, A comparison of four inverse approaches to groundwater flow and transport parameter identification, Water Resources Research, Vol. 27, No. 9, pp. 2219-2232, 1991.

Koekkoek, E. J. W., and H. Booltink, Neural network models to predict soil water retention, European Journal of Soil Since, Vol. 50, pp. 489-495, 1999.

Kolmogorov, A. N., On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition, Dokaldy Akademii Nauk USSR, Vol. 144, pp. 679-681, 1957, (American Mathematical Society Translation, Vol. 28, pp.55-59, 1963).

Konikow, L. F., and J. D. Bredehoeft, Computer model of two-dimensional solute transport and dispersion in groundwater, U. S. Geological Survey Techniques of Water Resources Investigation, Book 7, Chapter C, 70 pp., 1978.

Krom, T. D., and D. Rosbjerg, Artificial neural networks development and application in groundwater pollution remediation design, IAHS Publication, No. 265, pp. 34-40, 2000.

Lallemand-Barres, P., and P. Peaudecerf, Recherche des relations entre la valeur de la dispersivite macroscopique d'un milieu aquifers, ses autres caracteristiques et les conditions de mesure, etude bibiographique, Bulletin, Bureau de Recbercbes Goologiques et Minieres, Sec. 3/4:277-87, 1978.

Lavrent'ev, M. M., V. G. Romanov, and S. P. ShishatSkii, Ill-posed problems of mathematical physics and analysis, translated from the Russian by J.R. Schulenberger ; translation edited by Lev J. Leifman, American Mathematical Society, Providence Rhode Island, 290 pp., 1986.

Lohman, S. W., Groundwater hydraulics, U.S. Geological Survey, Professional Paper 708, 70 pp., 1972.

Mahar, P. S., and B. Datta, Optimal identification of ground-water pollution sources and parameter estimation, Journal of Water Resources Planning and Management, Vol. 127, No. 1, pp. 20-29, 2001.

Maier, H. R., and G. C. Dandy, Neural networks for prediction and forecasting of water resources variables: a review of modelling issues and applications, Environmental Modelling and Softwares, Vol. 15, pp. 101-123, 2000.

Mayer, A. S., and C. Huang, Development and application of a coupled-process parameter inversion model based on the maximum likelihood estimation method, Advances in Water Resources, Vol. 22, No. 8, pp. 841-853, 1999.

McDonald, J. M., and A. W. Harbaugh, A modular three-dimensional finite difference flow model, Techniques of Water Resources Investigations of the U.S. Geological Survey, Book 6, 586 pp., 1988

McLaughlin, D., and L. R. Townley, A reassessment of the groundwater inverse problem, Water Resources Research, Vol. 32, No.5, pp.1131-1161, 1996.

Medina, A., and J. Carrera, Coupled estimation of flow and solute transport parameters, Water Resources Research, Vol. 32, No. 10, pp. 3063-3076, 1996.

Mehrotra, K., C. K. Mohan, and S. Ranka, Elements of artificial neural networks, The MIT Press, MA, 344 pp., 1997.

Menke, W., Geophysical data analysis discrete inverse theory, Academic Press Inc., NY, 289 pp., 1989.

Mishra, S., and J. C. Parker, On the relation between saturated conductivity and capillary retention characteristics, Ground Water, Vol. 28, pp. 775-777, 1989.

Mishra, S., and J. C. Parker, Predictive model for unsaturated flow parameters, pp.393-301. In M. Th. van Genuchten et al., (ed.), Indirect methods for estimating hydraulic properties of unsaturated soils, Proc. Int. Worksh., Riverside, CA, 11-13 Oct., 1992, University of California, Riverside.

Morshed, J., and J. J. Kaluarachchi, Parameter estimation using artificial neural network and genetic algorithm for free-product migration and recovery, Water Resources Research, Vol. 34, No.5, pp. 1101-1113, 1998.

Mukhopadhyay, A., Spatial estimation of transmissivity using neural network, Ground Water, Vol. 37, No. 3, pp. 458-464, 1999.

Murtagh, B. A., and M. A. Saunders, Minos 5.1 user's guide technical report SOL 83-20r, Department of Operation Research Stanford University, Stanford, CA, 1993.

Murty, V. V. N., and V. H. Scott, Determination of transport model parameters in groundwater aquifers, Water Resources Research, Vol. 13, No. 6, pp. 941-947, 1977.

Neuman, S. P., and S. Yakowitz, A statistical approach to the inverse problem of the aquifer hydrology 1 theory, Water Resources Research, Vol. 15, No. 4, pp. 845-860, 1979.

Neuman, S., Calibration of distributed parameter groundwater flow models viewed as a multiple-objective decision process under uncertainty, Water Resources Research, Vol. 9, No. 4, pp. 1006-1021, 1973.

Peck, A., S. Gorelick, G. de Marsily, S. Foster, and V. Kovalevsky, Consequences of spatial variability in aquifer properties and data limitations for groundwater modeling practice, IAHS Publication, No.175, 272 pp., 1988.

Perkins, T. K., and O. C. Johnson, A review of diffusion and dispersion in porous media, Society of Petroleum Engineers Journal, Vol. 3, pp.70-94, 1963.

Peter, E., and N. Kalogerakis, Applied parameter estimation for chemical engineering, Marcel Dekker, Inc., 198 pp., NY, 2001.

Rizzo, D. M., and D. E. Dougherty, Characterization of aquifer properties using artificial neural networks: neural kriging, Water Resources Research, Vol. 30, No. 2, pp. 483-497, 1994.

Rogers, L. L., and F. U. Dawla, Optimization of groundwater remediation using artificial neural networks with parallel solute transport modeling, Water Resources Research, Vol. 30 No. 3, pp.457-481, 1994.

Ross, G. J. S., Nonlinear estimation, Springer-Verlag, NY, Inc, 189 pp., 1990.

Rumelhart, D. E, and J. L. McClelland, Parallel distributed processing explorations in the microstructure of cognition, MIT Press, Cambridge, MA, 344 pp., 1989.

Scales, J. A., and L. Tenorio, Prior information and uncertainty in inverse problems, Geophysics, Vol. 66, No, 2, pp. 389-397, 2001.

Schaap, M. G., F. J. Leij, and M. Th. van Genuchten, Neural network analysis for hierarchical prediction of soil hydraulic properties, Soil Science Society of America Journal, Vol. 62, No. 4, pp. 847-855, 1998.

Schalkoff, R. J., Artificial neural networks, McGraw-Hill Companies, Inc., NY, 422 pp., 1997.

Sciortino, A., T. C. Harmon, and W. W-G. Yeh, Inverse modeling for locating dense nonaqueous pools in groundwater under steady flow conditions, Water Resources Research, Vol. 36, No. 7, pp. 1723-1735, 2000.

Shigidi, A. M., Solving the Inverse problem in groundwater flow by iterative inversion of a neural network, Ph.D. dissertation, Colorado State University, 140 pp., 2000.

Sidauruk, P., A. H-D. Cheng, and D. Ouazar, Ground water contaminant source and transport parameter identification by correlation coefficient optimization, Ground Water, Vol. 36, No. 2, pp. 208-214, 1998.

Spichak, V., and I. Popova, Artificial neural network inversion of magnetotelluric data in terms of three-dimensional earth macroparameters, Geophysical Journal International, Vol. 142, pp. 15-26, 2000.

Sprecher, D. A., On the structure of continuous functions of several variables, Transactions of the American Mathematical Society, Vol. 115, pp. 340-355, 1965.

Stauffer, F., Current trends in groundwater hydraulics, in Sato K., and Y. Iwasa (ed.), Groundwater updates, Springer, NY, 491 pp., 2001.

Strecker, E. W., and W. Chu, Parameter identification of a ground-water contaminant transport model, Ground Water, Vol. 24, No. 1, pp. 56-62, 1986.

Sun, N., N-Z. Sun, M. Elimelech, and J. N. Ryan, Sensitivity analysis and parameter identifiably for colloid transport in geochemically heterogeneous porous media, Water Resources Research, Vol. 37, No. 2, pp. 209-222, 2001.

Sun, Ne-Z., and W. W-G Yeh, identification of parameter structure in groundwater inverse problem, Water Resource Research, Vol. 21, No. 6, pp. 869-883, 1985.

Sun, Ne-Z., Inverse problems in groundwater modeling, Kluwer Academic Publishers, Dordrecht, Netherlands, 537 pp., 1994.

Suykens, J. O., J. P. L. Vandewalle, and B. L. R. De Moor, Artificial neural networks for modelling and control of non-linear systems, Kluwer Academic Publication, Dordrecht, Netherlands, pp. 235, 1996.

Torfs, P., and G. Bier, Automatic model calibration using a universal approximator, IAHS Publication, No. 265, pp.236-242, 2000.

Umari, A., R. Willis, and P. L-F Liu, identification of aquifer dispersivities in two-dimensional transient groundwater contaminant transport an optimization approach, Water Resources Research, Vol. 15, No. 4, pp. 815-831, 1979.

Valstar, J., Inverse modeling of groundwater flow and groundwater mass transport, IAHS Publication, No. 269, pp. 239-245, 2001.

Van Genuchten, M Th., A closed-form equation for predicting the hydraulic conductivity of unsaturated soil. Soil Science Society of America Journal, Vol. 44, pp.892-898, 1980.

Voss, C. I., A finite-element simulation model for saturated-unsaturated fluid density dependent groundwater flow with energy transport or chemically reactive single-species solute transport, US Geological Survey, Water Resources Investigation, 84-4369, 409 pp., 1984.

Wagner, B.J., and S. M. Gorelick, A statistical methodology for estimating transport parameters: theory and application to one-dimensional advection-dispersion systems, Water Resources Research, Vol. 22, No. 8, pp.1303-1315, 1986.

Wagner, B.J., Simultaneous parameter estimation and contaminant source characterization for coupled groundwater flow and contaminant transport modelling, Journal of Hydrology, No. 135, pp. 275-303, 1992.

Walton, W. C., Groundwater modeling utilities, Lewis Publishers, Boca Raton, 640 pp., 1992.

Xiang, J., and D. Elsworth, Direct and integration methods of parameter estimation in groundwater transport system, Applied Mathematical Modeling, Vol. 16, pp. 404-413, 1992.

Yang, C.-C., S. O. Prasher, R. Lacroix, S. Sreekanth, N. K. Patni and L. Masse, Artificial neural network model for subsurface-drained farmlands, Journal of Irrigation and Drainage Engineering, Vol. 123, No. 4, pp. 285-292, 1997.

Yeh, W. W-G., Review of parameter identification procedures in groundwater hydrology: the inverse problem, Water Resources Research, Vol. 22, No. 2, pp. 95-108, 1986.

Zheng, C., MT3D: A modular three-dimensional transport model for simulation of advection, dispersion, and chemical reactions of contaminants in groundwater systems, S. S. Papadopulos and Associates, Inc., MD, 1990.

Hetch-Nielson, R., Neurocomputing, Reading, Addison-Wesley, MA, 1990.

LeCun, Y., Efficient learning and second-order methods, A Tutorial at NIPS 93, Denver, 1993.

Pfankuch, H. O., Contribution a l'etude des deplacement de fluids miscible dans unmilieu poreux, Rev. Inst. Fr. Petrol. Vol. 18, No. 2, pp. 215-270. 1963.

Scheidegger, A. E., General theory of dispersion in porous media, Journal of Geophysical Research, Vol. 66, pp. 3273-3278, 1961.

Saffman, P. G., Dispersion due to molecular diffusion and macroscopic mixing in flow through a network of capillaries, Journal of Fluid Mechanics, Vol. 7, No. 2, pp. 194-208, 1960.

Appendix A

MATLAB SCRIPT

```
%This Script is to train the Neural Network and estimate the dispersivity values.
clear all
close all hidden
% Field grid size
R=80;
C=20;
% Load the Dispersivity values generated by the MATLAB
% if ~exist('def')
% def=1;
load filename.dat
  T=filename;
  T=T';
%end;
% Load the concentration matrix generated by the MT3D model.
if ~exist('def')
  def=1;
  load filename.cnc
  P=filename;
%For selection
  c=1;
  NewP=[];
```

```
  for (i=1:1600)
    if (i==1)
      NewP=[NewP;P(1+(i-1)*80:60+(i-1)*80,1:20)];
    else
      NewP=[NewP;P(1+(i-1)*80+1:60+(i-1)*80+1,1:20)];
    end;
  end;
  clear P;
  P=NewP;
end;
  [ro,co]=size(P);
  P=reshape(P',[R*C ro/R]);
end;
% Select only some cells from the concentration map to be trained
% P=P([1:1600],:);
 valid=[];
 for i=1:size(P,1)
  if(min(P(i,:))~=max(P(i,:)))
    valid=[valid i];
 end;
end;
 P=P(valid,:);
% To select some cells at specific interval from the domain
```

```
c=1;
for i=1:size(P,2)
  c=1;
    for j=50:5:size(P,1)
      NewP(c,i)=P(j,i);
      c=c+1;
      if (c==200) break
      end;
    end;
end
% Normalize the input and the output data
[P,minP,maxP]=premnmx(P);
[T,minT,maxT]=premnmx(T);
% Prepare the net for training
% S is the range of the inputs
clear s
s(1:size(P,1),1)=-1;
s(1:size(P,1),2)=1;
% The neural network training commands [type of neural network, number of hidden
layers and neurons, activation functions, and training algorithm]:
net=newff(s,[20 1],{'tansig','tansig'},'trainrp');
net.performFcn = 'msereg';
net.performParam.ratio = .5;
```

```
net.trainParam.show=10;
net.trainParam.mc=.9;
net.trainParam.lr=0.1;
net.trainParam.lr_inc=1.05;
net.trainParam.epochs=800;
net.trainParam.goal=.00001;
% The Training Center
% Choose a percentage for each part of the data
%    pp: percentage of training data
%    pm: percentage of validation data
%    pn: percentage of testing data
pp=70/100;
pv=10/100;
pt=10/100;
% Length of the data
%Q=size(P,2); % change it to Pthe size of the input data
Q=210
% Training data
n=floor(pp*Q);
PP=P(:,1:n);
TP=T(:,1:n);
% Testing data
m=floor(pv*Q);
```

```
V.P=P(:,n+1:n+m);

V.T=T(:,n+1:n+m);

% Validation data testing

k=floor(pt*Q);

S.P=P(:,n+m+1:n+m+k);

S.T=T(:,n+m+1:n+m+k);

% The new known data and the noisy data

sig=10/100;   %percent of error

sig1=20/100;

sig2=30/100;

sig3=40/100;

NP=P(:,n+m+k+1:Q);

NPn=NP.*(1+sig*randn(size(NP)));

NPn1=NP.*(1+sig1*randn(size(NP)));

NPn2=NP.*(1+sig2*randn(size(NP)));

NPn3=NP.*(1+sig3*randn(size(NP)));

NT=T(:,n+m+k+1:Q);

% Start training

[net,tr]=train(net,PP,TP,[],[],V,S);

% Simulation by the trained network to estimate the dispersivity values.

ST=sim(net,NP);

STn=sim(net,NPn);

STn1=sim(net,NPn1);
```

```
STn2=sim(net,NPn2);

STn3=sim(net,NPn3);

ST=postmnmx(ST,minT,maxT);

STn=postmnmx(STn,minT,maxT);

STn1=postmnmx(STn1,minT,maxT);

STn2=postmnmx(STn2,minT,maxT);

STn3=postmnmx(STn3,minT,maxT);

NT=postmnmx(NT,minT,maxT);

NP=postmnmx(NP,minP,maxP);

NPn=postmnmx(NPn,minP,maxP);

NPn1=postmnmx(NPn1,minP,maxP);

NPn2=postmnmx(NPn2,minP,maxP);

NPn3=postmnmx(NPn3,minP,maxP);

% Plots of the outputs.

figure;

 postreg(ST,NT);    % new known data.

% Plot the output as one variable sorted from low to high

% Then plot its corresponding NN output and measure the Surface error percentage

sz=size(NT,1)*size(NT,2);

ZR=reshape(NT,[1 sz]);

ZN=reshape(ST,[1 sz]);

[x,i]=sort(ZR);

plot([1:sz],ZR(i),[1:sz],abs(ZN(i)-ZR(i)),'--')
```

```
%legend('Original Output','Error between NN output');
xlabel('Number of Simulated Dispersivity value');
ylabel('Dispersivity value');
disp(['Surface Error Percentage of NN is = 'num2str(sum(sum(abs(NT-
ST)))/sum(sum(abs(NT)))*100) '%']);
% Noisy outputs
figure;
subplot(2,1,1); plot([1:21],STn,[1:21],ST,'--');
title(['Noise and Nois-Free Outputs with \sigma=' num2str(sig*100) '%']);
xlabel('Number of Simulated Dispersivity value');
ylabel('Dispersivity value');
%legend('With Noise','Noise Free');
c=corrcoef(ST,STn); c=c(1,2);
[x,i]=sort(ST);
subplot(2,1,2); plot(ST,STn,'.',x,polyval(polyfit(x,STn(i),1),x)); %title(['\rho = '
num2str(c)]);
xlabel('Simulated Dispersivity value');
ylabel(' Dispersivity value');
%legend('Noise Free','With Noise');
legend(['R = ' num2str(c)])
figure;
subplot(2,1,1); plot([1:21],STn1,[1:21],ST,'--');
title(['Noise and Noise-Free Outputs with \sigma =' num2str(sig1*100) '%']);
```

```
xlabel('Number of Simulated Dispersivity value');
ylabel('Dispersivity value');
legend('With Noise','Noise Free');
c1=corrcoef(ST,STn1); c1=c1(1,2);
[x,i]=sort(ST);
subplot(2,1,2); plot(ST,STn1,'.',x,polyval(polyfit(x,STn1(i),1),x));% title(['\rho = '
num2str(c)]);
xlabel('Simulated Dispersivity value');
ylabel(' Dispersivity value');
legend('Noise Free','With Noise');
legend(['R = ' num2str(c)])
figure;
subplot(2,1,1); plot([1:21],STn2,[1:21],ST,'--');
title(['Noise and Noise-Free Outputs with \sigma=' num2str(sig2*100) '%']);
xlabel('Number of Simulated Dispersivity value');
ylabel('Dispersivity value');
legend('With Noise','Noise Free');
c2=corrcoef(ST,STn2); c2=c2(1,2);
[x,i]=sort(ST);
subplot(2,1,2); plot(ST,STn2,'.',x,polyval(polyfit(x,STn2(i),1),x)); %title(['\rho = '
num2str(c)]);
xlabel('Simulated Dispersivity value');
ylabel(' Dispersivity value');
```

```
legend('Noise Free','With Noise');
legend(['R = ' num2str(c)])
figure;
subplot(2,1,1); plot([1:21],STn3,[1:21],ST,'--');
title(['Noise and Noise-Free Outputs with \sigma=' num2str(sig3*100) '%']);
xlabel('Number of Simulated Dispersivity value');
ylabel('Dispersivity value');
legend('With Noise','Noise Free');
c3=corrcoef(ST,STn3); c3=c3(1,2);
[x,i]=sort(ST);
subplot(2,1,2); plot(ST,STn3,'.',x,polyval(polyfit(x,STn3(i),1),x)); %title(['\rho = '
num2str(c)]);
xlabel('Simulated Dispersivity value');
ylabel(' Dispersivity value');
legend(['R = ' num2str(c)])%('Noise Free','With Noise');
ST=ST';
save ST ST -ascii;
save NT NT -ascii;
```