DESIGN OF ARTIFICIAL NEURAL NETWORK FOR SOLVING INVERSE PROBLEMS

L.N.M .Tawfiq
College of Education Ibn Al-Haitham, Baghdad University

Abstract

This paper proposes neural network based forward models in iterative inversion algorithms for solving inverse problems .Iterative algorithms are commonly used to solve inverse problem.

Typical iterative inversion approaches use a numerical forward model to predict the signal for a given input data. The desired output can then be found by iteratively minimizing energy function. The use of numerical models is computationally expensive, and therefore, alternative forward models need to be explored.

This study proposes two different neural network based iterative inverse problem solutions. In addition, specialized neural networks forward models are proposed and used in place of numerical forward models. The first approach uses basis function networks (radial basis function (RBFNN)) to approximate the mapping from the input space to the output space. The back propagation training algorithm are then used to estimate the network parameter. The second approach proposes the use of two networks in a feedback configuration.

Introduction

Three classes of problems may be define: :

(i) Given input x(k) and system P(w), determine the output y(k)

(ii) Given input x(k)and output y(k), determine P(w) .

(iii) Given system P(w)and the output y(k), determine x(k) .

The first case presents the forward problem while the second and third cases are related to inverse problems. An inverse problem is said to be well-posed if the solution satisfies three properties:

Existence, Uniqueness and Continuity: the solution depends continuously on the input .

The forward problem in general is well-posed and is solved analytically or by means of numerical modeling. In contrast, inverse problems in general are ill-posed, lacking both uniqueness and continuous. The algorithm starts with an initial estimate of the parameters and solves the corresponding forward problem to determine the signal (output). The error between the measured and predicted signals is minimized iteratively by updating the parameters. When the error is below a pre-set threshold, the parameters represent the desired solution.

Iterative techniques for inverse problems have been developed using numerical models [1] [2] based on integral and differential formulations [3] [4] to represent the forward process. However, all of these methods have certain drawbacks. Iterative methods using three-dimensional numerical models are, in general, computationally intensive, and therefore have limited practical application. In addition, updating the parameters is also difficult, since gradient-based approaches cannot be easily applied to solutions of numerical models such as finite element models (FEM)[5].

The major objective of this study is the development of solutions to inverse problems. The proposed solutions are described below with a brief discussion of their advantages and disadvantages. In addition, we also compare the proposed solution methods to existing algorithms presented in the literature, and present the differences between the existing and proposed algorithms.

Neural Network Based Iterative Inversion

A single neural network is used instead of a numerical model as the forward model in the inversion approach. The advantages of this approach include its speed and simplicity as the forward model inherits many of the advantages of neural networks.

Feedback Neural Networks

This approach is a modification of section2, and uses two neural networks in feedback configuration, with one neural network modeling the forward problem while the other models the inverse problem.

The major drawback of section 2, 3 is that the performance of the neural networks depends on the data used in training and testing. Mathematically, each of the neural networks approximates the function mapping the input to the output, and as long the test data is similar to the training data, the network can interpolate between the training data points to obtain a reasonable prediction. However, when the test signal is no longer similar to the training data, the network is forced to extrapolate and the performance is seen to degrade. Alternatively, we have to consider the design of neural networks that are capable of extrapolation. But the design of an extrapolation neural network involves several issues. For instance, there are no methods for ensuring that the error in the network prediction is within reasonable bounds during the extrapolation procedure.

Model based methods for extrapolation use numerical models in an iterative approach. These models are capable of correctly predicting the signal, since they solve the underlying governing equations. However, numerical models are computationally intensive, which limits their application. An ideal solution therefore would be to combine the power of numerical models with the computational speed of neural networks, i.e., to create neural networks that are capable of solving the underlying partial differential equations. The use of Hopfield networks makes it difficult to solve the inverse problem, especially if derivatives need to be computed in the inversion procedure. Such networks are therefore not considered in this study.

Our proposed approach is different in that we derive the neural network from the point of view of the inverse problem. The neural network architecture that is eventually developed also makes it easy to solve the forward problem. The structure of the neural network is also simpler than those reported in the literature, making it easier to implement in parallel in both hardware and software. Furthermore, the neural network is not limited to a specific type of domain, and does not require any training. In fact, the FFNN weights are determined solely by the differential equation and associated boundary conditions – an advantage in solving inverse problems.

Two neural networks are used in a feedback configuration. The forward network predicts the signal corresponding to input of inverse network while the inverse (characterization) network predicts the output signal. The forward network replaces the FEM .The overall approach to solving the inverse problem is as follows. The signal from a defect with unknown shape is input into the forward network to obtain the corresponding prediction of the signal. On the other hand, if the error exceeds a threshold, the training mode is invoked and the networks are retrained with the back propagation algorithm.

A radial basis function neural network (RBFNN) is used as an inverse network for characterizing the desired output. The forward and inverse networks are first trained using the conjugate gradient (CG) method [6][7]. The forward network is tested to ensure that it is capable of accurately modeling the forward process for the training database, and if necessary, the network parameters are optimized. In addition, the inverse network is also trained and its parameters optimized. This process is referred to as the training mode. The goal of the optimization step is to minimize the error due to the inverse RBFNN. Let f be the error between the actual signal (actual output) and the prediction of the forward network in the feedback configuration. In order for f to be zero, the characterization network must be an exact inverse of the forward network. While the functional form of the forward network can be derived easily, obtaining its inverse analytically is difficult. This is due to the fact that the output of the forward network is a function of the number and location of the respective basis function centers in each network. The inverse is, therefore, estimated numerically. Now let

E = the error at the output of the inverse network .

w_{kj} = interconnection weight from node j in the hidden layer to node k in the output layer

c_j = center of the basis function (at node j in the hidden layer)

σ_j = spread of the basis function

y = the measured signal

$x = (x_1, x_2, \ldots, x_k, \ldots, x_n)$ be the desired output of the RBFNN.

$\hat{x} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k, \ldots, \hat{x}_n)$ be the actual output of the RBFNN.

Then, the error *E* can be defined as:

$x = (x_1, x_2, \ldots, x_k, \ldots, x_n)$ be the desired output of the RBFNN.

$\hat{x} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k, \ldots, \hat{x}n)$ be the actual output of the RBFNN.

Then, the error *E* can be defined as :

$$E = \tfrac{1}{2} \| x - \hat{x} \|^2 = \tfrac{1}{2} \sum_{k=1}^{N} (x_k - \hat{x}_k)^2 \quad \ldots\ldots\ldots(1)$$

Where $\hat{x}_k$ is given by :

$$\hat{x}_k = \sum_{j=1}^{L} w_{kj}\, \phi(\| y - c_j \| / 2\sigma_j^2 \quad \ldots\ldots\ldots(2)$$

and the basis function is chosen to be Gaussian function:

$$\phi(\| y - c_j \| / 2\sigma_j^2) = \exp (- \| y - c_j \|^2 / 2\sigma_j^2) \ldots\ldots(3)$$

Substituting(2)and(3)into(1) and taking the derivative with respect to the weights w_{kj}, we have:

$$\partial E / \partial w_{kj} = -(x_k - \hat{x}_k)\, \phi(\| - c_j \| / 2\sigma_j^2) \ldots\ldots\ldots(4)$$

Similarly, the derivative of the error with respect to the other two parameters (c_j and σ_j) can be computed as follows:

$$\partial E / \partial c_{ji} = \sum_{k=1}^{n} -(x_k - \hat{x}_k)[\, w_{kj}\, \phi'(\| y - c_j \| / 2\sigma_j^2)\, ((y_i - c_{ji})/\sigma_j^2)] \ldots\ldots\ldots(5)$$

$$\partial E / \partial \sigma_j = \sum_{k=1}^{n} -(x_k - \hat{x}_k)[w_{kj}\, \phi'(\| y - c_j \| / 2\sigma_j^2)(\| y - c_j \|^2 / \sigma_j^3)] \ldots\ldots\ldots(6)$$

The derivatives are then substituted into the CG equation to derive the update equations for the three parameters. The CG equation is given by : $d^{new} = d^{old} + \eta \rho_k$, $\rho_0 = -g_0$ and $\rho_k = -g_k + \beta_k \rho_{k-1}$, $g_k = \partial E / \partial d_k$,

$$\beta_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}$$, where *d* is the parameter of interest w_{kj} , c_{ji} or σ_j .

Sensitivity Analysis Of The Inverse Problem

Intuitively, an error in the parameter w will result in an error (which can be large for ill-posed problems) in the estimate of the $\hat{x}$. In order to quantify the error in the parameters w_{kj}, c_j and σ_j , we assume that $\hat{w}_{kj} = w_{kj} + \Delta w_{kj}$ is the measured value of the potentials where w_{kj} is the true value of the potential and Δw_{kj} is the error in the measurement at node k and j. Let $\hat{c}_j(n)$ and $\tilde{\sigma}_j(n)$ be the corresponding estimated values of the parameters in the node j at iteration *n*. We assume that $\hat{c}_j(n) = c_j(n) + \delta_j(n)$ where $c_j(n)$is the corresponding estimate of c_j when the measurement error in w_{kj} is zero. *S*imilarly, let $\tilde{\sigma}_j(n) = \sigma_j(n) + \varepsilon_j(n)$. The corresponding sum-squared error at the output of the FBNN is:

$$\bar{E}(n) = \tfrac{1}{2} \sum_{i=1}^{N} \bar{E}_i^2(n) \quad (7)$$

Then, the CG update equations for $\hat{c}$ and $\tilde{\sigma}$ are given by :

$\hat{C}_j(n) = \hat{c}_j(n-1) + \eta \rho_1(n-1)$ where $\rho_1(0) = -g_1(0)$, $\rho_1(n-1) = -g_1(n-1) + \beta_{k1}(n-1)\, \rho_1(n-2)$ .....(8a)

$\tilde{\sigma}_j(n) = \tilde{\sigma}_j(n-1) + \eta \rho_2(n-1)$ where $\rho_2(0) = -g_2(0)$, $\rho_2(n-1) = -g_2(n-1) + \beta_{k2}(n-1)\, \rho_2(n-2)$ .....(8b)

Then, the error in the estimates $\hat{c}$ and $\tilde{\sigma}$ can be computed according to Theorem I below.

Theorem I: The following results hold for the error in the estimates in$\hat{c}$ and $\tilde{\sigma}$:

1- $\delta_j(n) = \delta_j(n-1) + f\,(E_i(n-1), c_j(n-1), \delta_j(n-1), \varepsilon_j(n-1), \beta_{k1}(n-1)\, \rho_1(n-2))$.

2- $\varepsilon_j(n) = \varepsilon_j(n-1) + h\,(E_i(n-1), c_j(n-1), \delta_j(n-1), \varepsilon_j(n-1), \beta_{k2}(n-1) \rho_2(n-2))$.

Proof: In order to prove Theorem I (1), we start with (8) :

$\hat{c}_j(n) = \hat{c}_j(n-1) + \eta \rho_1(n-1) = \hat{c}_j(n-1) + \eta\{-g_1(n-1) + \beta_{k1}(n-1) \rho_1(n-2)\}$ where $\rho_1(0) = -g_1(0) = -\partial \bar{E}(0) / \partial \hat{c}_j$.

$$\hat{c}_j(n) = \hat{c}_j(n-1) + \eta(-\partial \bar{E}(n-1) / \partial \hat{c}_j + \beta_{k1}(n-1)\, \rho_1(n-2)) \ldots\ldots\ldots(9)$$

substituting (7) in (9) : $\hat{c}_j(n) = \hat{c}_j(n-1) + \eta(\sum_{i=1}^{N} \sum_{j=1}^{N} \bar{E}_i(n-1) \phi_j \hat{w}_{ij} + \beta_{k1}(n-1)\, \rho_1(n-2))$

$\hat{c}_j(n) = \hat{c}_j(n-2) + \eta(\sum_{i=1}^{N}\sum_{j=1}^{N} \bar{E}_i(n-2)\phi\hat{w}_{ij} + \beta_{k1}(n-2)\rho_1(n-3)) + \eta(\sum_{i=1}^{N}\sum_{j=1}^{N} \bar{E}_i(n-1)\ \phi_j\hat{w}_{ij} + \beta_{k1}(n-1)\rho_1(n-2)$) .Continuing with the recursion, we get :

$\hat{c}_j(n) = c_j(0) + \eta(\sum_{i=1}^{N}\sum_{j=1}^{N} \bar{E}_i(0)\phi_j\hat{w}_{ij}) + \eta(\sum_{i=1}^{N}\sum_{j=1}^{N} \bar{E}_i(1)\phi_j\hat{w}_{ij} + \beta_{11}\rho_1(0)) + \eta(\sum_{i=1}^{N}\sum_{j=1}^{N} \bar{E}_i(2)\phi_j\ \hat{w}_{ij} + \beta_{21}\rho_1(1)) +\ldots. + \eta(\sum_{i=1}^{N}\sum_{j=1}^{N} \bar{E}_i(n-1)\phi_j\ \hat{w}_{ij} + \beta_{(n-1)1}\ \rho_1(n-2))$ or

$$\hat{c}_j(n) = c_j(0) + \eta(\sum_{i=1}^{N}\sum_{j=1}^{N} \phi_j\ \hat{w}_{ij} \sum_{t=o}^{n-1} \bar{E}_i(t) + \sum_{t=1}^{n-1} \beta_{t1}\rho_1(t-1)) \quad \ldots\ldots(10)$$

From definition of $\hat{c}$ and (10), we have:

$$\hat{c}_j(n) = c_j(0) + \eta\ [\sum_{i,j=1}^{N} \phi_j\ \hat{w}_{ij} \sum_{t=o}^{n-1} \{\ E_i(t) - \sum_{k=1}^{N} \Delta\phi_k \sum_{m=1}^{M} (\ c_m(t)\ \hat{w}_{ijm} + \beta_m(t)\ \hat{g}_{ijm}\ ) - \sum_{k=1}^{N} \phi_k \sum_{m=1}^{M} (\delta_m(t)\ \hat{w}_{ijm} + \varepsilon_m(t)\hat{g}_{ijm}\ )\} + \sum_{t=1}^{n-1} \beta_{t1}\rho_1(t-1)] \quad \ldots\ldots(11)$$

This can be simplified to give :

$$\hat{c}_j(n) = c_j(0) + \eta[\sum_{i,j=1}^{N} \phi_j\ \hat{w}_{ij} \sum_{t=o}^{n-1} E_i(t) + \sum_{i,j=1}^{N} \beta_{t1}\rho_1(t-1)] - \eta[\sum_{i,j=1}^{N} \phi_j\hat{w}_{ij} \sum_{t=o}^{n-1} \sum_{k=1}^{N} \Delta\phi_k \sum_{m=1}^{M} (c_m(t)\hat{w}_{ijm} + \beta_{m1}(t)\hat{g}_{ijm}) + \sum_{t=1}^{n-1} \beta_{t1}\rho_1(t-1)] - \eta[\sum_{i,j=1}^{N} \phi_j\hat{w}_{ij} \sum_{t=o}^{n-1} \sum_{k=1}^{N} \phi_k \sum_{m=1}^{M} (\delta_m(t)\hat{w}_{ijm} + \varepsilon_m(t)\hat{g}_{ijm}\ ) + \sum_{t=1}^{n-1} \beta_{t1}\rho_1(t-1)] \quad \ldots\ldots(12)$$

From the definition ofϕ_i ,we get :

$\hat{c}(n) = c_j(0 + \eta[\sum_{i,j=1}^{N} \phi_j\hat{w}_{ij} \sum_{t=o}^{n-1} E_i(t) + \sum_{t=1}^{n-1} \beta_{t1}\rho(t-1)] + \eta[\sum_{i,j=1}^{N} \Delta\phi_j\hat{w}_{ij} \sum_{t=o}^{n-1} E_i(t) + \sum_{t=1}^{n-1} \beta_{t1}\rho_1(t-1)] - [\sum_{i,j=1}^{N} \phi_j\hat{w}_{ij} \sum_{t=o}^{n-1} \sum_{k=1}^{N} \Delta\phi_k \sum_{m=1}^{M} (\ c_m(t)\hat{w}_{ijm} + \beta_m(t)\ \hat{g}_{ijm}\ ) + \sum_{t=1}^{n-1} \beta_{t1}\rho_1(t-1)] - \eta\ [\sum_{i,j=1}^{N} \phi_j\hat{w}_{ij} \sum_{t=o}^{n-1} \sum_{k=1}^{N} \phi_k \sum_{m=1}^{M} (\delta_m(t)\hat{w}_{ijm} + \varepsilon_m(t)\hat{g}_{ijm}) + \sum_{t=1}^{n-1} \beta_{t1}\rho_1(t-1)]$

But $c_j(n) = c_j(0) + \eta[\sum_{i,j=1}^{N} \phi_j\ \hat{w}_{ij} \sum_{t=o}^{n-1} E_i(t) + \sum_{t=1}^{n-1} \beta_{t1}\rho(t-1)]$

So, $\hat{c}_j(n) = c_j(n) + \eta\ [\sum_{i,j=1}^{N} \Delta\phi_j\ \hat{w}_{ij} \sum_{t=o}^{n-1} E_i(t) + \sum_{t=1}^{n-1} \beta_{t1}\ \rho_1(t-1)] - \eta[\sum_{i,j=1}^{N} \phi_j\ \hat{w}_{ij} \sum_{t=o}^{n-1} \sum_{k=1}^{N} \Delta\phi_k \sum_{m=1}^{M} (c_m(t)\ \hat{w}_{ijm} + \beta_{m1}(t)\ \hat{g}_{ijm}\ ) + \sum_{t=1}^{n-1} \beta_{t1}\rho_1(t-1)] - \eta[\sum_{i,j=1}^{N} \phi_j\ \hat{w}_{ij} \sum_{t=o}^{n-1} \sum_{k=1}^{N} \phi_k \sum_{m=1}^{M} (\delta_m(t)\ \hat{w}_{ijm} + \varepsilon_m(t)\hat{g}_{ijm}\ ) + \sum_{t=1}^{n-1} \beta_{t1}\rho_1(t-1)]$ Or $\hat{c}_j(n) = c_j(n) + \delta_j(n)$ .....(13) , where

$\delta_j(n) = \eta\ [\sum_{i,j=1}^{N} \Delta\phi_j\ \hat{w}_{ij} \sum_{t=o}^{n-1} E_i(t) + \sum_{t=1}^{n-1} \beta_{t1}\rho_1(t-1)] - \eta\ [\sum_{i,j=1}^{N} \phi_j\ \hat{w}_{ij} \sum_{t=o}^{n-1} \sum_{k=1}^{N} \Delta\phi_k \sum_{m=1}^{M} (c_m(t)\ \hat{w}_{ijm} + \beta_m(t)\ \hat{g}_{ijm}\ ) + \sum_{t=1}^{n-1} \beta_{t1}\rho_1(t-1)] - \eta[\sum_{i,j=1}^{N} \phi_j\hat{w}_{ij} \sum_{t=o}^{n-1} \sum_{k=1}^{N} \phi_k \sum_{m=1}^{M} (\ \delta_m(t)\hat{w}_{ijm} + \varepsilon_m(t)\hat{g}_{ijm}\ ) + \sum_{t=1}^{n-1} \beta_{t1}\rho_1(t-1)]$ .....(14)

with $\delta_j(0) = \varepsilon_j(0) = 0$. Using this fact and (14) we have :

$$\delta_j(1) = \eta \sum_{i,j=1}^{N} \Delta\phi_j \hat{w}_{ij} E_i(0) - \eta \sum_{i,j=1}^{N} \phi_j \hat{w}_{ij} \sum_{k=1}^{N} \Delta\phi_k \sum_{m=1}^{M} (c_m(0) \hat{w}_{ijm} + \beta_{m1}(0) \hat{g}_{ijm}) - \eta \sum_{i,j=1}^{N} \phi_j \hat{w}_{ij} \sum_{k=1}^{N} \phi_k \sum_{m=1}^{M} (\delta_m(0) \hat{w}_{ijm} + \varepsilon_m(0) \hat{g}_{ijm}) \quad (15)$$

$$\delta_j(2) = \eta [\sum_{i,j=1}^{N} \Delta\phi_j \hat{w}_{ij} \sum_{t=o}^{L} E_i(t) + \beta_{11} \rho_1(0)] - \eta [\sum_{i,j=1}^{N} \phi_j \hat{w}_{ij} \sum_{t=o}^{L} \sum_{k=1}^{N} \Delta\phi_k \sum_{m=1}^{M} (c_m(t) \hat{w}_{ijm} + \beta_m(t) \hat{g}_{ij}) + \beta_{11} \rho_1(0)] - \eta [\sum_{i,j=1}^{N} \phi_j \hat{w}_{ij} \sum_{t=o}^{L} \sum_{k=1}^{N} \phi_k \sum_{m=1}^{M} (\delta_m(t) \hat{w}_{ijm} + \varepsilon_m(t) \hat{g}_{ijm}) + \beta_{11} \rho_1(0)]$$ or

$$\delta_j(2) = \delta_j(1) + \eta[\sum_{i,j=1}^{N} \Delta\phi_j \hat{w}_{ij} E_i(1) + \beta_{11} \rho_1(0)] - \eta [\sum_{i,j=1}^{N} \phi_j \hat{w}_{ij} \sum_{k=1}^{N} \Delta\phi_k \sum_{m=1}^{M} (c_m(1) \hat{w}_{ijm} + \beta_{m1}(1) \hat{g}_{ijm}) + \beta_{11} \rho_1(0)] - \eta[\sum_{i,j=1}^{N} \phi_j \hat{w}_{ij} \sum_{k=1}^{N} \phi_k \sum_{m=1}^{M} (\delta_m(1) \hat{w}_{ijm} + \varepsilon_m(1) \hat{g}_{ijm})] \quad (16)$$

This process can be carried out for all *n*, and the recursive relation is given by

$$\delta_j(n) = \delta_j(n-1) + \eta[\sum_{i,j=1}^{N} E_i(n-1) \Delta\phi_j \hat{w}_{ij} + \beta_{(n-1)1} \rho_1(n-2)] - \eta[\sum_{i,j=1}^{N} \phi_j \hat{w}_{ij} \sum_{k=1}^{N} \Delta\phi_k \sum_{m=1}^{M} (c_m(n-1) \hat{w}_{ikm} + \beta_m(n-1) \hat{g}_{ikm}) + \beta_{1(n-1)} \rho_1(n-2)] - \eta[\sum_{i,j=1}^{N} \phi_j \hat{w}_{ij} \sum_{k=1}^{N} \phi_k \sum_{m=1}^{M} (\delta_m(n-1) \hat{w}_{ikm} + \varepsilon_m(n-1) \hat{g}_{ikm}) + \beta_{1(n-1)} \rho_1(n-2)]$$ or $\delta_j(n) = \delta_j(n-1) + f (E_i(n-1) , c_j(n-1), \delta_j(n-1), \varepsilon_j(n-1), \beta_{1k}(n-1) \rho_1(n-2))$

By a similar process, we can prove Theorem I (2).

Theorem I confirms an important and intuitive fact: that the error in the estimates of the material properties at any iteration depends on the error in the material property estimates in all previous iterations. This dependence is shown explicitly in equation (16). This fact is not a drawback of the FENN-based inversion algorithm. Rather, it shows the ill-posed nature of the inverse problem, and illustrates the point that the inversion results are only as good as the measured data.

Applications

Forward And Inverse Problems In

Two Dimensions :

The general form of Poisson's equation in two dimensions is

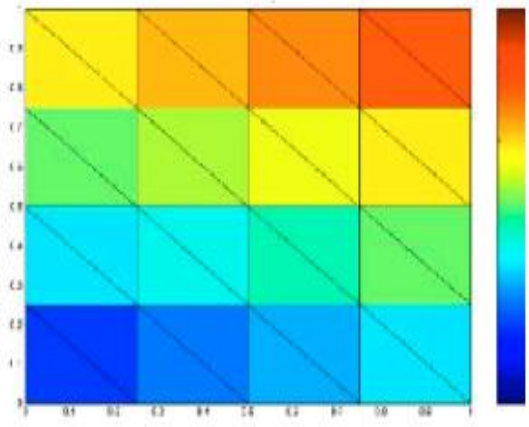
$$-\partial(\alpha_x \partial\phi / \partial x) / \partial x - \partial(\alpha_y \partial\phi / \partial y) / \partial y + \beta\phi = f \quad (17)$$

with boundary conditions: $\phi = p$ on Γ_1 and $(\alpha_x (\partial\phi/\partial x)\hat{x} + \alpha_y (\partial\phi/\partial y)\hat{y})\cdot\tilde{n} + \gamma\phi = q$ on Γ_2. Several forward and inverse problem examples based on (17) were solved using the FFNN algorithm. These are :

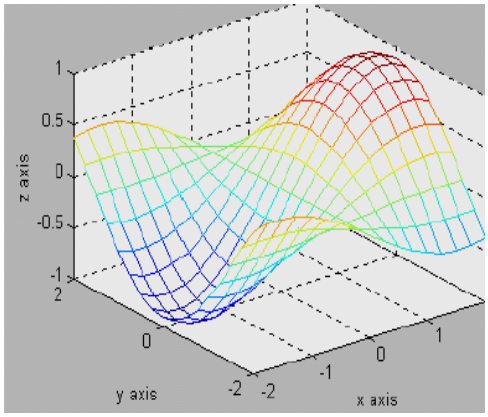
1.Problem I used $\alpha_x = \alpha_y = \alpha = x + y$, $(x,y) \in [0,1]\times[0,1]$, $\beta = \gamma = q = 0$ and $f = -2$ The analytical solution to the forward problem is $\phi = x + y$ when the Dirichlet boundary conditions are: $\phi = y$, $x = 0$, $\phi = 1 + y$, $x = 1$, $\phi = x$, $y = 0$, $\phi = 1 + x$, $y = 1$.

Conversely, the inverse problem in this case is to estimate α in each element ($\beta = 0$) given the potentials $\phi = x + y$ at each of the nodes .

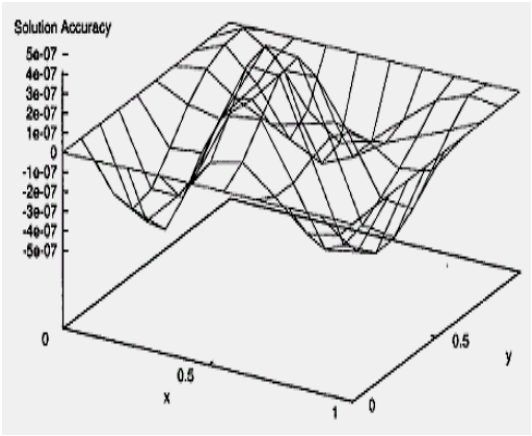
The inverse problem solution is presented in Figure1, with the analytical solution for α ,the FFNN inversion and the error between the analytical and FFNN results in Figure1(a), (b) and (c) respectively. Several different discretizations were tested for solving the inverse problem , and the results presented here were obtained using 5 nodes in each direction. Results obtained from a different discretization (11 nodes in each direction) are presented in Figure2. The discretization was observed to affect the number of iterations needed for convergence, with the smaller mesh requiring a smaller number of iterations.



(a)



(b)

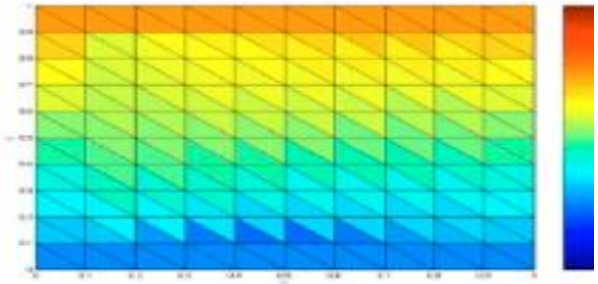


(c)

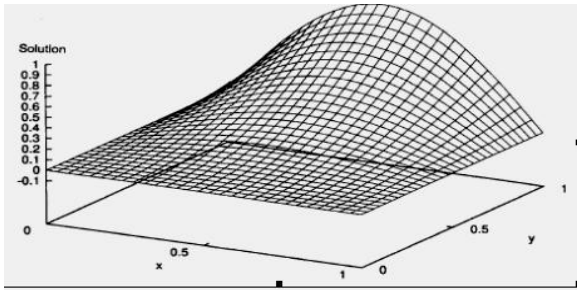
Fig.(1) : Inverse problem solution for Problem I with an 5×5 discretization (a) Analytical solution (b) FFNN inversion (c) Error between (a) and (b).

2.Problem II used $\alpha_x=\alpha_y = \alpha = x + y$, $(x,y) \in [0,1]\times[0,1]$, $\beta= \gamma =q =0$ and $f = -6(x +y)$. The analytical solution to the forward problem is $\phi=x^2+y^2$ when the Dirichlet boundary conditions are $\phi= y^2$, $x = 0$, $\phi =1+ y^2$, $x =1$, $\phi = x^2$, $y = 0$, $\phi =1+ x^2$, $y =1$.

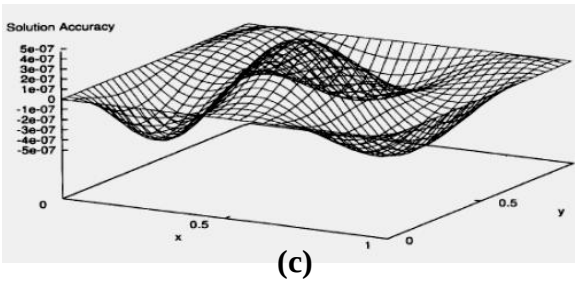
Conversely, the inverse problem in this case is to estimate α in each element given the potentials $\phi = x^2 + y^2$ at each of the nodes. The inverse problem solution is presented in Figure 3, with Figure 3(a), (b) and (c) showing analytical solution for α , the FFNN inversion result and the error in the FFNN inversion respectively. As in Problem I, several discretizations were used for solving the inverse problem, and the results presented in Fig.(3).



(a)



(b)



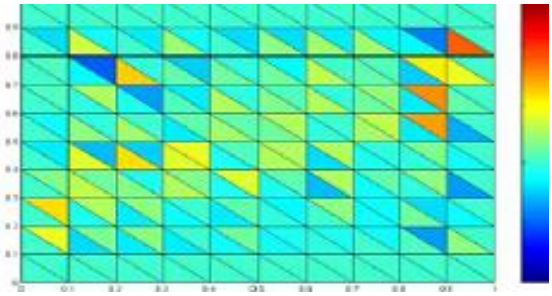
(c)

Fig.(2) : Inversion results for problemI with 11×11 mesh (a) Analytical solution (b) FFNN Inversion (c) Error between (a) and (b).

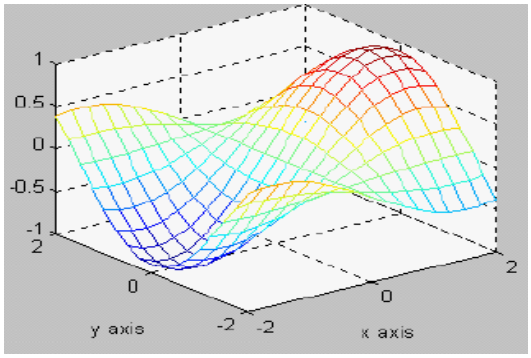
were obtained using 11 nodes in each direction.

3.Problem III used $\alpha_x = y$,$\alpha_y = x$, $(x,y) \in [0,1]\times[0,1]$, $\beta = \gamma = q = 0$and $f = -2(x+y)$.The analytical solution to the forward problem is $\phi = x^2 + y^2$ when the Dirichlet boundary conditions are $\phi = y^2$, $x = 0$, $\phi = 1+y^2$, $x = 1$, $\phi = x^2$, $y = 0$, $\phi = 1 + x^2$, $y = 1$. Conversely, the inverse problem in this case is to estimate α_x and α_y in each element given the potentials $\phi = x^2 + y^2$ at each of the nodes.

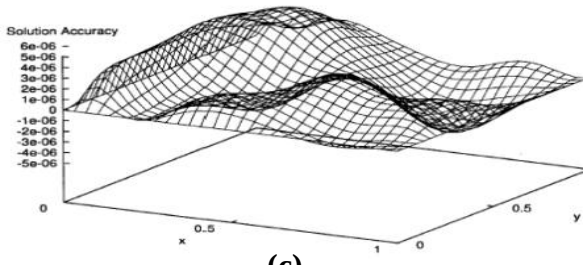
The inverse problem solution is presented in Fig.(4), with Figs. 4(a), (b) and (c) showing the analytical solution , the FFNN inversion and the error in the FFNN inversion respectively with triangular elements.



(a)

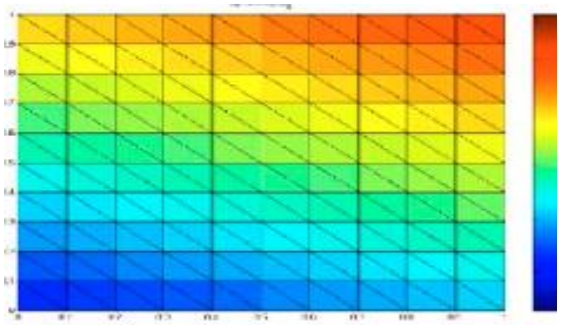


(b)

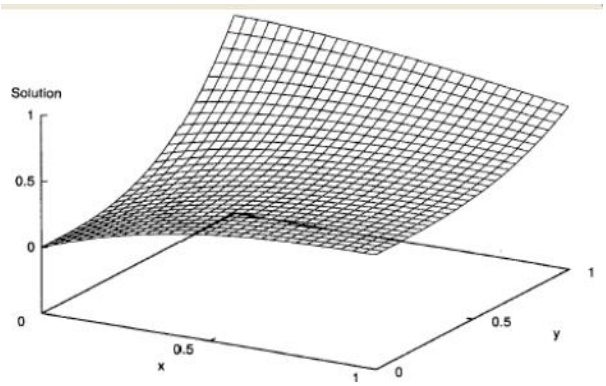


(c)

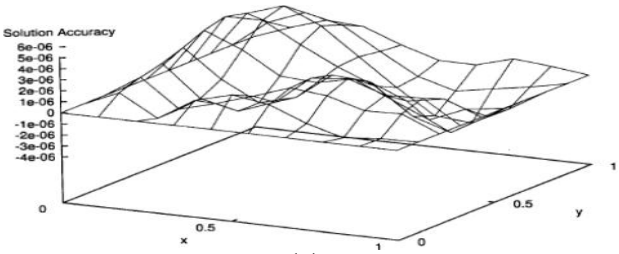
Fig.(3) : Inversion results for Problem II with an 11x11 mesh (a) Analytical solution (b) FFNN inversion (c) Error between (a) and (b) .



(a)



(b)



(c)

Fig. (4) : Inversion results for Problem III, with an 11x11 mesh (a) Analytical solution (b) FFNN Inversion (c) Error between (a) and (b) .

Conclusions

This study proposed the use of neural network based forward models in iterative algorithms for inversion problems. The use of neural network based forward models offers several advantages over numerical models in terms of both implementation of gradient calculations in the updates of the parameters and overall computational cost. Two different types of neural networks–radial basis function neural networks and ridge basis function neural networks–were initially used to represent the forward model. These forward models were used, in a simple iterative scheme, or in combination with an inverse model in feedback configuration, to solve the inverse

الخلاصة

يتضمن البحث اقـــتراح شبكـــة عصبيـــة على أســـاس النمـــوذج التقـــدمي لخوارزميـــة التكرار العكسي . حيث استخدمنا خوارزميـــة التكرار لحل المسائل العكسية بدلا من النماذج العددية و التي تكون حساباتها مكلفـــة .وقـــد اقترحنـــا شبكتـــان عصبيتـــان مختلفتان على أساس التكرار لحل المسائل العكسيـــة إضافـــة الى شبكات عصبيـــة خاصـــة ذات النموذج التقدمي استخدمت بدلا من النماذج العدد يـــة التقدميـــة . النوع الأول يستخدم شبكات دوال الأســـاس (RBFNN) لتقريب الدوال من فضاء المدخلات الي فضاء المخرجات و استخدمنا خوارزميـــة التدريب المرتـــد لحساب معاملات الشبكـــة . النوع الثاني اقترحنا شبكتـــان في تشـــكل التغذيـــة التراجـــعي .

problem. One drawback of these approaches is that the forward models are not accurate when the inputs are not similar to those used in the training database. This paper proposed the design of neural networks that are capable of solving differential equations and hence does not depend on training data. This specialized neural network has a weight structure that allows both the forward and inverse problems to be solved using simple gradient - based algorithms. Results of applying the FFNN in two-dimensional problems were presented and show that the proposed FFNN accurately models the forward problem. Application of this neural network for inverse problem solutions indicates that the solution closely matches the analytical solution.

References

[1] S. A. Terekhoff and N. Nfedorova ″Cascade Neural Networks in Variational thods for Boundary Value Problems ″, Russian Federal Nuclear Center, 2000.

[2] S.Lawrence, D.Back ,A.Tsoi and C.Lee Giles,″ On the Distribution of performance from Multiple Neural Network Trials″, IEEE Transactions on Neural Networks, Vol.8, NO. 6, 2000, P. 1507-1517.

[3] B. Xinzhou and B. Jammes, ″Solving Steady-State Partial Derivative Equation with Neural Network″. Application to Steady-State Heat Transfer Problem-systems(LAAS), 2001.

[4] J. S. Baras and A. Lavigna, ″Convergence of a Neural Network Classifier, Systems Research Center″, University of Maryland, College Park , 2002.

[5] A. R. Mitchell and R. Wait,″ The Finite Element Method in Partial Differential Equations″, 1978.

[6] A. Al-Bayati and N. Al–Assady, ″Conjugate Gradient Methods″, Technical Research Report, NO.1, School of Computer Studies, Leeds University,U.K., 1996.

[7] Q.Wang and T. Aoyama,″A neural network solver for differential equations ″, Gakuen Kibanadai–Nishi, Neuron Computing, Vol. 1, NO. 1, 2001.