

GRADIENT-ENHANCED NEURAL NETWORK RESPONSE SURFACE APPROXIMATIONS

Weiyu Liu*

S.M. Batill†

Department of Aerospace and Mechanical Engineering
University of Notre Dame
Notre Dame, Indiana

ABSTRACT

An approach to develop response surface approximations based upon artificial neural networks trained using both state and sensitivity information is described in this paper. Compared to previous approaches, this approach does not require weighting the residuals of the targets and gradients and is able to approximate gradient-consistent response surfaces with a relatively compact network architecture. Numerical simulation on selected problems is used to evaluate the approach that is implemented with the efficient Levenberg-Marquardt neural network training algorithm. The results show that this gradient-enhanced neural network training approach possesses the capability to develop improved response surface approximations compared to the non-gradient neural network training approach.

NOMENCLATURE

b	biases of neural networks (NNs)
$b1$	biases at the hidden layer
$b2$	biases at the output layer
e	errors
\hat{e}_{max}	global maximum error
\hat{e}_{rms}	global root-mean-square error
\hat{e}_{avg}	global average error
f	merit function
f_a	actual merit function
f_n	NN simulation of merit function
\mathbf{g}	gradient vector
\mathbf{H}	Hessian matrix
I	number of NN input nodes
J	number of NN hidden nodes
\mathbf{J}	Jacobian matrix
K	number of output nodes

M	number of training data points
MSE	mean sum-squared error
N	number of NN weights
N_h	number of NN hidden nodes
SSE	sum-squared error
t	target value
w	neural network weights
\mathbf{w}	neural network weight vector
$w1_{ji}$	weights at the hidden layer
$w2_{kj}$	weights at the output layer
x	design variable, and neural network input
y	state variable, or NN hidden layer output
z	neural network output

I. INTRODUCTION

Artificial neural networks (ANNs) have been used to develop parametric response surface (RS) approximations in a number of Multidisciplinary Design Optimization (MDO) procedures^{1,2,3,4}. Improving the accuracy of these approximations while reducing the amount of design space information required to develop the ANNs are areas of current interest. By incorporating gradient information into the neural network approximations the number of times that the entire system must be analyzed in order to represent these trends can be reduced. Moreover, the benefit of incorporating gradient information also includes improved accuracy of response surface approximations. The gradient information can often be obtained through techniques such as automatic differentiation⁵ or the Global Sensitivity Equations (GSEs)⁶ such that only one complete system analysis is required to yield the gradient information at a single design point.

An approach to develop response surface approximations based upon artificial neural networks trained using both state and sensitivity information is described in this paper. Compared to previous approaches⁷, this approach does not require weighting the residuals of the targets and gradients and is able to ap-

*Graduate Research Assistant

†Associate Dean and Professor, Associate Fellow AIAA

¹Copyright ©2000 by Stephen M. Batill. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

proximate gradient-consistent response surfaces with a relatively compact network architecture. Numerical simulation on selected problems is used to evaluate this approach that is implemented with the efficient Levenberg-Marquardt neural network training algorithm.

II. OVERVIEW OF PREVIOUS WORK

The neural networks considered in this study are based upon the two-layer sigmoid activation feedforward model shown in Figure 1. Using this same class of ANNs, Sellar⁷ presented three approaches to include gradient information in neural network training for response surface approximations.

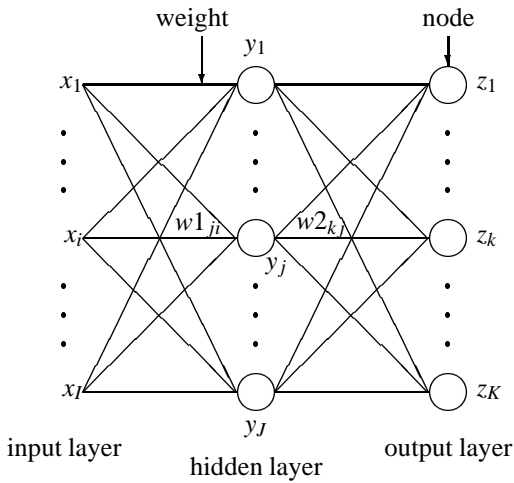


Figure 1: A Feedforward Neural Network Model

The first approach, Multi-point Database Augmentation, assumed a linear relationship in the neighborhood of the original training data points and added approximate state information to the training database. These additional training data were computed from a first order Taylor series approximation using available local sensitivity information around the original training data. An obvious difficulty with this approach is the rapid growth of the training data set and corresponding increase in network training time as the dimension of the design space increases.

The second approach, treated each gradient as an additional, unique output of the neural network. Although the size of the training database remains fixed, this approach can lead to inconsistency between target outputs and gradient outputs that are related to each other through the actual function. This can occur since the weights connecting the hidden nodes to each output of a neural network are unique and the outputs have no specific relationship to each other.

The third approach, Nonlinear Residual-Minimization Gradient Approximation, reduced simultaneously the residuals of the targets and gradients in the least-squares sense. In order to maintain the consistency between target outputs and gradient outputs, the gradients of the network outputs with respect to the network inputs were explicitly expressed using the activation functions of the network, which for the two-layer sigmoid activation feedforward neural network are expressed as⁷

$$\frac{\partial z_k}{\partial x_i} = z_k(1 - z_k) \sum_{q=1}^J w_{1qi} w_{2kq} y_q(1 - y_q), \quad (1)$$

where w_1 and w_2 are the network's weights which have to be adjusted to map the target and gradient information in neural network training. The error function selected for minimization during the algorithm was the vector of residuals of the targets and gradients. This also required that the residuals be weighted such that the product of the targets and corresponding factors, and of the gradients and associated weighting factors, are of the same order. Unfortunately the determination of the appropriate weighting scheme for the residuals was actually a trial-and-error process. Since it is impractical to use a trial-and-error process for response surface approximations in actual engineering applications, alternative approaches need to be considered.

This paper presents a method for Gradient-Enhanced Neural Network Training (GeNNT) that is an extension of the third approach, but can avoid the disadvantages of residual weighting. This approach introduces a new error function, which does not have to weight the residual vector but still can effectively express the combination of difference of targets and gradients between the actual function and the neural network approximation.

III. THE GENNT ALGORITHM

This section briefly presents the derivation of the GeNNT approach that is implemented with the Levenberg-Marquardt algorithm (GeNNT-LM). The complete derivation procedure is detailed in Liu⁸. In order to describe this approach, assume that the actual function f_a , and the approximate function f_n from the neural network are both functions of a single independent variable x shown in Figure 2. The purpose of GeNNT is to make these two response surfaces as close as possible, *i.e.*, the differences between these two response surfaces not only at the design points which are indicated as "X" but also in the neighborhood of the design points are hoped to be reduced. This can be

achieved by minimizing the mean squared error MSE. At any design point, MSE is expressed as

$$MSE = \frac{1}{3}(e_l^2 + e_c^2 + e_r^2), \quad (2)$$

where the left error e_l , the central error e_c and the right error e_r are respectively

$$e_l = t_l - z_l, \quad e_c = t_c - z_c, \quad e_r = t_r - z_r.$$

Here t_c and z_c are target and network output respectively at the design point, z_l and z_r , t_l and t_r , are respectively the approximate values of targets and network outputs near the design point that are computed by a first order Taylor Series approximation.

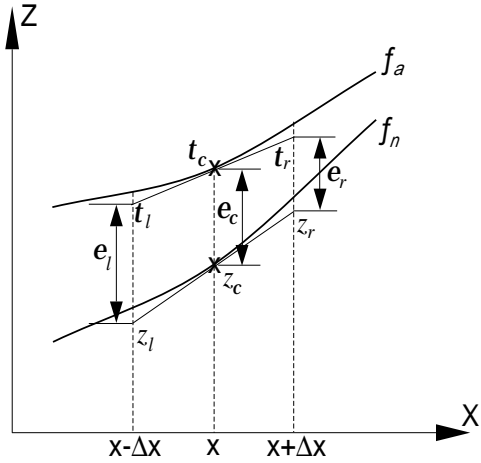


Figure 2: Actual Function f_a and Approximate Function f_n

Formulation of the Error Function

In a general situation with K functions in an I -dimensional design space

$$t_1 = f_{a1}(x_1, x_2, \dots, x_I), \quad t_2 = f_{a2}(x_1, x_2, \dots, x_I), \\ \dots, \quad t_K = f_{aK}(x_1, x_2, \dots, x_I),$$

one could develop a two-layer network with I inputs, J hidden neurons, and K output neurons that is referred to as an I - J - K network, to approximate these K actual functions based on M input and output pairs (IOPs). Of course, one has to assume that the local sensitivities at these design points are available. In the following development, M will not appear because one can set $M = 1$ for a more concise presentation of the approach. The general case with $M > 1$ is similar and will be discussed later. Thus, one can write an expression for the

MSE,

$$MSE = \frac{1}{3K} \sum_{k=1}^K (e_{kl}^2 + e_{kc}^2 + e_{kr}^2) \\ \equiv \frac{1}{3}(MSE_l + MSE_c + MSE_r), \quad (3)$$

where

$$MSE_l = \frac{1}{K} \sum_{k=1}^K e_{kl}^2, \quad MSE_c = \frac{1}{K} \sum_{k=1}^K e_{kc}^2,$$

$$MSE_r = \frac{1}{K} \sum_{k=1}^K e_{kr}^2,$$

$$e_{kl} = t_{kl} - z_{kl}, \quad e_{kc} = t_{kc} - z_{kc}, \quad e_{kr} = t_{kr} - z_{kr}.$$

Using Taylor Series expansion, one can express the functional values z_{kl} at the points in the neighborhood of the design points as,

$$z_{kl} = z_{kc} - \sum_{i=1}^I \frac{\partial z_k}{\partial x_i} \Delta x_i. \quad (4)$$

Similarly, one can obtain the expressions for z_{kr} , t_{kl} and t_{kr} respectively. In above equations, t_{kc} and z_{kc} are the values of the k th target and network output respectively at the design points.

Formulation of the GeNNT-LM Algorithm

Two network training algorithms were considered in this study. Results indicated that the Levenberg-Marquardt algorithm provides a faster convergence compared to the standard backpropagation algorithm. The original Levenberg-Marquardt algorithm⁹ is presented as

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - (\mathbf{H}^{(k)} + \mu^{(k)}\mathbf{I})^{-1} \mathbf{g}^{(k)}, \quad (5)$$

where \mathbf{w} is the N -dimensional vector of weights and biases in the neural network, \mathbf{H} is the Hessian matrix (second derivatives) of the error function, \mathbf{g} is the gradient of the error function, μ is a scalar. Because the error function MSE in this paper has the form of a sum of squares, a Levenberg-Marquardt modification algorithm¹⁰ which was designed to approach second-order training speed without having to compute the Hessian matrix \mathbf{H} was used.

Considering the term MSE_l in Equation 3, one can have

$$\mathbf{g}_l = \frac{\partial(MSE_l)}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial(MSE_l)}{\partial w_1} \\ \frac{\partial(MSE_l)}{\partial w_2} \\ \vdots \\ \frac{\partial(MSE_l)}{\partial w_N} \end{bmatrix} = \frac{2}{K} \mathbf{J}_l^T \mathbf{e}_l, \quad (6)$$

where

$$\mathbf{J}_l \equiv \begin{bmatrix} \frac{\partial e_{1l}}{\partial w_1} & \frac{\partial e_{1l}}{\partial w_2} & \dots & \frac{\partial e_{1l}}{\partial w_N} \\ \frac{\partial e_{2l}}{\partial w_1} & \frac{\partial e_{2l}}{\partial w_2} & \dots & \frac{\partial e_{2l}}{\partial w_N} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial e_{kl}}{\partial w_1} & \frac{\partial e_{kl}}{\partial w_2} & \dots & \frac{\partial e_{kl}}{\partial w_N} \end{bmatrix}_{K \times N},$$

$$\mathbf{e}_l \equiv \begin{bmatrix} e_{1l} \\ e_{2l} \\ \vdots \\ e_{kl} \end{bmatrix}.$$

And one can approximate the $N \times N$ Hessian matrix as¹⁰

$$\mathbf{H}_l \approx \frac{2}{K} \mathbf{J}_l^T \mathbf{J}_l, \quad (7)$$

and similarly,

$$\mathbf{g}_c = \frac{2}{K} \mathbf{J}_c^T \mathbf{e}_c, \quad \mathbf{H}_c \approx \frac{2}{K} \mathbf{J}_c^T \mathbf{J}_c. \quad (8)$$

$$\mathbf{g}_r = \frac{2}{K} \mathbf{J}_r^T \mathbf{e}_r, \quad \mathbf{H}_r \approx \frac{2}{K} \mathbf{J}_r^T \mathbf{J}_r. \quad (9)$$

Then, the Levenberg-Marquardt modification algorithm for GeNNT with the error function in Equation 3 can be updated as

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - (\mathbf{H}^{(k)} + \mu^{(k)} \mathbf{I})^{-1} \mathbf{g}^{(k)}, \quad (10)$$

where

$$\begin{aligned} \mathbf{g} &= \frac{1}{3} (\mathbf{g}_l + \mathbf{g}_c + \mathbf{g}_r) \\ &= \frac{2}{3K} (\mathbf{J}_l^T \mathbf{e}_l + \mathbf{J}_c^T \mathbf{e}_c + \mathbf{J}_r^T \mathbf{e}_r), \end{aligned} \quad (11)$$

$$\begin{aligned} \mathbf{H} &= \frac{1}{3} (\mathbf{H}_l + \mathbf{H}_c + \mathbf{H}_r) \\ &\approx \frac{2}{3K} (\mathbf{J}_l^T \mathbf{J}_l + \mathbf{J}_c^T \mathbf{J}_c + \mathbf{J}_r^T \mathbf{J}_r). \end{aligned} \quad (12)$$

The algorithm would be complete if one can obtain the three Jacobian matrices \mathbf{J}_c , \mathbf{J}_l and \mathbf{J}_r . \mathbf{J}_c can be calculated using the chain rule similar to that applied in the standard backpropagation algorithm^{10,11}, and the following shows how to derive \mathbf{J}_l and \mathbf{J}_r from \mathbf{J}_c . Because

$$e_{kl} = t_{kl} - z_{kl} = e_{kc} - \sum_{i=1}^I \left(\frac{\partial t_k}{\partial x_i} - \frac{\partial z_k}{\partial x_i} \right) \Delta x_i, \quad (13)$$

the element of \mathbf{J}_l ($K \times N$ matrix) is written as

$$(J_l)_{kn} = \frac{\partial e_{kl}}{\partial w_n} = \frac{\partial e_{kc}}{\partial w_n} + \frac{\partial}{\partial w_n} \left(\sum_{i=1}^I \frac{\partial z_k}{\partial x_i} \Delta x_i \right). \quad (14)$$

So, from the definition of Jacobian matrix \mathbf{J}_c , one can have

$$\mathbf{J}_l = \mathbf{J}_c + \Delta \mathbf{J}, \quad (15)$$

where the element of the $\Delta \mathbf{J}$ matrix is defined as

$$\begin{aligned} (\Delta J)_{kn} &= \frac{\partial}{\partial w_n} \left(\sum_{i=1}^I \frac{\partial z_k}{\partial x_i} \Delta x_i \right) \\ &= \sum_{i=1}^I \frac{\partial}{\partial w_n} \left(\frac{\partial z_k}{\partial x_i} \right) \Delta x_i. \end{aligned} \quad (16)$$

Similarly, one can have

$$\mathbf{J}_r = \mathbf{J}_c - \Delta \mathbf{J}. \quad (17)$$

Then, the final step is developing an expression for the $\Delta \mathbf{J}$ matrix.

Computing $\Delta \mathbf{J}$ Matrix

The weight space \mathbf{w} of an I - J - K network in Figure 1 is composed of four different parts, which are respectively the weights for the hidden layer $w1_{ji}$, the weights for the output layer $w2_{kj}$, the biases for the hidden nodes $b1_j$, and the biases for the output nodes $b2_k$.

In this section, the term $\frac{\partial}{\partial w_n} \left(\frac{\partial z_k}{\partial x_i} \right)$ in the elements of the $\Delta \mathbf{J}$ matrix is derived for each element in the different parts of the weight space \mathbf{w} .

From Equation 1, the gradient output of a two-layer feedforward network with the unipolar sigmoid activation function can be presented as

$$\frac{\partial z_k}{\partial x_i} \equiv Q_1 Q_2, \quad (18)$$

where

$$\begin{aligned} Q_1 &\equiv z_k (1 - z_k), \\ Q_2 &\equiv \sum_{q=1}^J w1_{qi} w2_{kq} y_q (1 - y_q). \end{aligned} \quad (19)$$

Using Equation 19, one can write,

$$\frac{\partial}{\partial w_{2sj}} \left(\frac{\partial z_k}{\partial x_i} \right) = \frac{\partial Q_1}{\partial w_{2sj}} Q_2 + \frac{\partial Q_2}{\partial w_{2sj}} Q_1, \quad (20)$$

where s is used as the subscript of $w2$ in order to be distinguished from k in z_k . Then using the chain rule and the functional relationship in a two-layer feedforward sigmoid network, one can write,

$$\begin{aligned} \frac{\partial Q_1}{\partial w_{2sj}} &= \frac{\partial (z_k (1 - z_k))}{\partial w_{2sj}} \\ &= (1 - 2z_k) \cdot \frac{\partial z_k}{\partial w_{2sj}} \end{aligned}$$

$$\begin{aligned}
&= (1 - 2z_k) \cdot \frac{dz_k}{du_k} \frac{\partial u_k}{\partial w_{2sj}} \\
&= \begin{cases} (1 - 2z_k) \cdot Q_1 y_j, & \text{as } s = k \\ 0, & \text{as } s \neq k \end{cases} \quad (21)
\end{aligned}$$

where u_{2k} is the input to the activation function on the output nodes, and

$$\begin{aligned}
\frac{\partial Q_2}{\partial w_{2sj}} &= \sum_{q=1}^J \left(w_{1qi} y_q (1 - y_q) \frac{\partial w_{2kq}}{\partial w_{2sj}} \right) \\
&= w_{1ji} y_j (1 - y_j) \frac{\partial w_{2kj}}{\partial w_{2sj}} \\
&= \begin{cases} w_{1ji} y_j (1 - y_j), & \text{as } s = k \\ 0, & \text{as } s \neq k \end{cases} \quad (22)
\end{aligned}$$

Substituting the above two results into Equation 20 one can write

$$\frac{\partial}{\partial w_{2sj}} \left(\frac{\partial z_k}{\partial x_i} \right) = \begin{cases} [(1 - 2z_k) Q_2 + w_{1ji} (1 - y_j)] y_j Q_1, & \text{as } s = k \\ 0, & \text{as } s \neq k \end{cases} \quad (23)$$

Similarly as w_{2kj} , one can derive

$$\frac{\partial}{\partial b_{2s}} \left(\frac{\partial z_k}{\partial x_i} \right), \frac{\partial}{\partial w_{1jp}} \left(\frac{\partial z_k}{\partial x_i} \right) \text{ and } \frac{\partial}{\partial b_{1j}} \left(\frac{\partial z_k}{\partial x_i} \right),$$

where p is used as the subscript of w_1 in order to be distinguished from i in x_i . In all above equations,

$$i, p = 1, \dots, I, \quad j = 1, \dots, J, \quad \text{and } k, s = 1, \dots, K.$$

The GENNT-LM Algorithm with $M > 1$

In this section the general case of the GENNT-LM algorithm with M IOPs and the formulation of the Jacobian matrix are overviewed. Similarly as in Equation 3, the error function with M IOPs can be presented as

$$MSE = \frac{1}{M} \sum_{m=1}^M MSE_m$$

where

$$MSE_m = \frac{1}{3} (MSE_{ml} + MSE_{mc} + MSE_{mr}),$$

and

$$MSE_{ml} = \frac{1}{K} \sum_{k=1}^K e_{mkl}^2, \quad MSE_{mc} = \frac{1}{K} \sum_{k=1}^K e_{mkc}^2,$$

$$MSE_{mr} = \frac{1}{K} \sum_{k=1}^K e_{mkr}^2,$$

where e_{mkl} , e_{mkc} and e_{mkr} are respectively the left error, the central error and the right error of the k th state at

the m th design point. Similarly as the procedure in the case with $M = 1$, one can get

$$\mathbf{g} = \frac{2}{3MK} (\mathbf{J}_l^T \mathbf{e}_l + \mathbf{J}_c^T \mathbf{e}_c + \mathbf{J}_r^T \mathbf{e}_r). \quad (24)$$

$$\mathbf{H} = \frac{2}{3MK} (\mathbf{J}_l^T \mathbf{J}_l + \mathbf{J}_c^T \mathbf{J}_c + \mathbf{J}_r^T \mathbf{J}_r). \quad (25)$$

\mathbf{J}_c and $\Delta \mathbf{J}$ are denoted as

$$\mathbf{J}_c \equiv \begin{bmatrix} \mathbf{J}_{1c} \\ \mathbf{J}_{2c} \\ \vdots \\ \mathbf{J}_{mc} \\ \vdots \\ \mathbf{J}_{Mc} \end{bmatrix}, \quad \Delta \mathbf{J} \equiv \begin{bmatrix} \Delta \mathbf{J}_1 \\ \Delta \mathbf{J}_2 \\ \vdots \\ \Delta \mathbf{J}_m \\ \vdots \\ \Delta \mathbf{J}_M \end{bmatrix},$$

where the elements of \mathbf{J}_{mc} and $\Delta \mathbf{J}_m$ are expressed respectively as

$$(J_{mc})_{kn} = \frac{\partial e_{mkc}}{\partial w_n}, \quad (26)$$

$$(\Delta \mathbf{J}_m)_{kn} = \sum_{i=1}^I \frac{\partial}{\partial w_n} \left(\frac{\partial z_{mk}}{\partial x_i} \right) \Delta x_i. \quad (27)$$

There is also

$$\mathbf{J}_l = \mathbf{J}_c + \Delta \mathbf{J}, \quad \mathbf{J}_r = \mathbf{J}_c - \Delta \mathbf{J}.$$

With Equation 24 and 25, the Levenberg-Marquardt modification algorithm for GeNNT can still be presented in the same form as Equation 5. The key step in the above algorithm is the computation of the $\Delta \mathbf{J}$ matrix and the Jacobian matrix \mathbf{J}_c . The $\Delta \mathbf{J}$ matrix can be obtained using Equation 27 in the similar way in the previous section. To compute the Jacobian matrix \mathbf{J}_c efficiently, the weight vector \mathbf{w} is reconstructed into a matrix. The N -dimensional weight space \mathbf{w} is composed of four parts, $[\mathbf{w}_1 \ \mathbf{b}_1 \ \mathbf{w}_2 \ \mathbf{b}_2]$, where \mathbf{w}_1 and \mathbf{w}_2 were reshaped from the weight matrix w_{1ji} and w_{2kj} respectively. For example,

$$\mathbf{w}_1 = [w_{111}, w_{112}, \dots, w_{11I}, \quad w_{121}, w_{122}, \dots, w_{12I}, \\
\quad \dots, \quad w_{1J1}, w_{1J2}, \dots, w_{1JI}]^T.$$

So, the element block \mathbf{J}_{mc} in the Jacobian matrix \mathbf{J}_c can be presented as

$$\mathbf{J}_m^T = \begin{bmatrix} \frac{\partial e_{m1}}{\partial \mathbf{w}_1} & \frac{\partial e_{m2}}{\partial \mathbf{w}_1} & \dots & \frac{\partial e_{mK}}{\partial \mathbf{w}_1} \\ \frac{\partial e_{m1}}{\partial \mathbf{b}_1} & \frac{\partial e_{m2}}{\partial \mathbf{b}_1} & \dots & \frac{\partial e_{mK}}{\partial \mathbf{b}_1} \\ \frac{\partial e_{m1}}{\partial \mathbf{w}_2} & \frac{\partial e_{m2}}{\partial \mathbf{w}_2} & \dots & \frac{\partial e_{mK}}{\partial \mathbf{w}_2} \\ \frac{\partial e_{m1}}{\partial \mathbf{b}_2} & \frac{\partial e_{m2}}{\partial \mathbf{b}_2} & \dots & \frac{\partial e_{mK}}{\partial \mathbf{b}_2} \end{bmatrix}_{N \times K}, \quad (28)$$

where the subscript c is dropped from \mathbf{J}_{mc} and e_{mkc} for the sake of conciseness.

IV. APPROACH EVALUATION

In this section the proposed approach is applied to two test problems and compared to a non-gradient response surface approximation also based on the neural network training algorithm. Comparisons were made with the quality of response surfaces approximated and reliability of the algorithms.

Test Problem 1

In this problem the system analysis representing the source for the design information are explicit, analytic expressions in the design variables and non-local states. This problem operates on three design variables $\{\vec{x}\}$ and two states $\{\vec{y}\}$ as given in Equation 29. The size of this problem was chosen to allow for graphical presentation of selected results.

$$\begin{aligned} \text{State 1 :} \\ y_1 &= x_1^2 + x_2 + x_3 - 0.2y_2 \\ \text{State 2 :} \\ y_2 &= \sqrt{y_1} + x_1 + x_3 \end{aligned} \quad (29)$$

The system analysis, an iterative process, for this fully-coupled, nonlinear problem was performed to obtain the non-parametric response surfaces y_1 and y_2 . The state and gradient values at 36 design points, *i.e.*, all combinations of

$$\begin{aligned} x_1 &= (-10, -3, 3, 10) \\ x_2 &= (0, 5, 10) \\ x_3 &= (0, 5, 10), \end{aligned}$$

were used as the training data to approximate the response surfaces in the design space below.

$$\begin{aligned} -10.0 &\leq x_1 \leq 10.0 \\ 0.0 &\leq x_2 \leq 10.0 \\ 0.0 &\leq x_3 \leq 10.0 \end{aligned}$$

In all figures hereafter, the state values are linearly scaled and mapped into $[0.1, 0.9]$.

Results

Because most conclusions drawn from the artificial neural networks based response surface (RS) approximation of y_2 are the same as that of y_1 , only the approximation of y_1 is shown and discussed here. A $3-N_{hidden}-1$ network was developed to approximate the response surface of y_1 , where the number of hidden nodes N_{hidden} is case dependent. The only training criterion is $MSE = 0.0025$, or $\bar{e}_{rms} = \sqrt{MSE} = 0.05$. Results show that the training criterion was reached for all

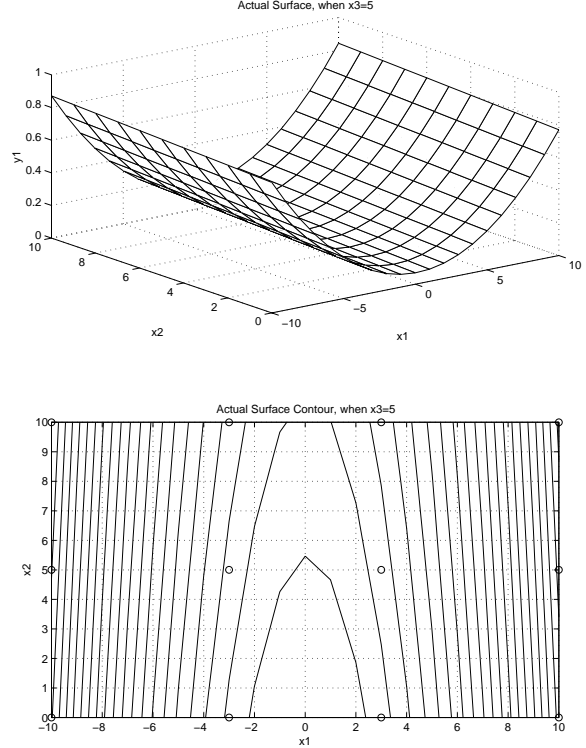


Figure 3: Nonparametric Response Surface and Contour of y_1 at $x_3 = 5$

cases with $N_{hidden} = 6$. Because the response surfaces of y_1 in the design space of (x_1, x_2) for different values of x_3 have similar characteristics, the discussion will focus on the RS at $x_3 = 5$. The actual nonparametric response surface and the approximation by the GeNNT approach are shown in Figure 3 and 4, respectively.

The GeNNT approach was compared with the non-gradient neural network training (NNT) approach. Six trials were performed to obtain the approximate RS of y_1 for each of five cases shown in Table 1, which are distinguished from each other by the termination criterion MSE (or \bar{e}_{rms}), and the training approach. Comparisons are based upon a number of “global errors”: the maximum error \hat{e}_{max} , the root-mean-square error \hat{e}_{rms} and the average error \hat{e}_{avg} . These global errors were calculated from target and NN output values at all combinations of integer values of (x_1, x_2, x_3) , *i.e.* $21 \times 11 \times 11 = 2541$ data points, across the whole design space (x_1, x_2, x_3) .

The average values of different global errors for all trials of each group are shown in Table 1 and Figure 5, and the detailed data of \hat{e}_{rms} , \hat{e}_{avg} and \hat{e}_{max} for each trial are shown in Figure 6, 7 and 8 respectively. The cross-section plots of the nonparametric and approximate response surface at $x_3 = 5$ and $x_2 = 0, 5$ and 10

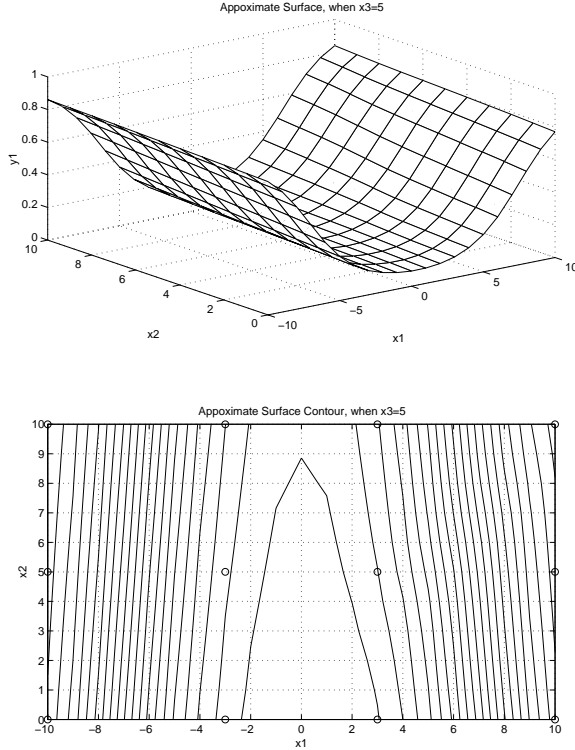


Figure 4: Approximate Response Surface and Contour of y_1 at $x_3 = 5$ using GeNNT

respectively, of one typical trial in Group A2 and B2 are shown in Figure 9 and 10. Because the difference between Group A1 and Group B1 is similar to A2 vs B2, and there is no Group A3 corresponding to B3, the cross-section plots of Group A1, B1 and B3 are omitted here.

Discussion of Results

- 1) From the global errors \hat{e}_{rms} , \hat{e}_{avg} and \hat{e}_{max} in Table 1 and Figure 5, the quality of GeNNT group A1 and A2 is better than the corresponding NNT group B1 and B2. This conclusion also can be drawn from comparison between the cross-section response surface of Group A2 and B2 as shown in Figure 9 and Figure 10 respectively. Even the RS of Group A1 with the training criterion $MSE = 0.01$ is better than that of Group B3 with $MSE = 0.0001$. The cross-section plots of the approximation using the GeNNT approach in Figure 9 show that the approximate RS in the neighborhood of the training data is very close to the actual RS, and this results from the influence of the gradient information constraints on the ANN training.
- 2) The detailed error data of each trial for all groups

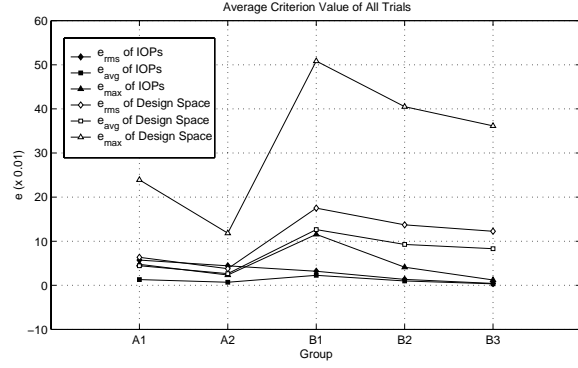


Figure 5: Average Criterion Value of All Trials

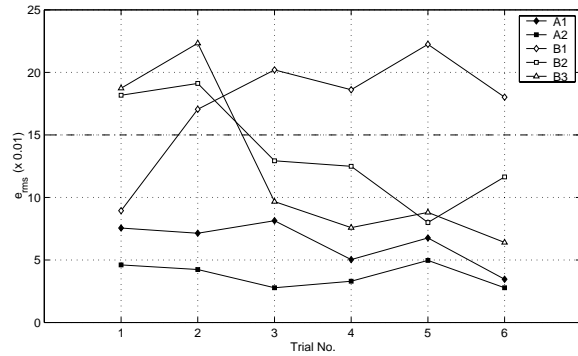


Figure 6: RMS Errors of Each Trial

are shown in Figure 6, 7 and 8. These data indicate that the results from the GeNNT approach are much more consistent than that from NNT. When taking a look at the approximate response surfaces of Group A2 and B2 shown in Figure 9 and 10, one observes that Group A2 is a better approximation across the range of the variable x_1 . Actually, these response surfaces are Trial 2 in Group A2 and Trial 3 in Group B2 respectively. One can also use the mathematical criterion to arrive at the same conclusion. Consider the error \hat{e}_{rms} for example. If the dashed line $\hat{e}_{rms} = 0.15$ in Figure 6 is set as the criterion, and it is assumed that the RS whose \hat{e}_{rms} value is above the line is inadequate, then there are 2 of 6 trials in Group B2 that are inadequate. The results of other groups are listed in the last column in Table 1, which also shows that the results of the GeNNT approach are very consistent. One can also draw the same conclusion if one uses $\hat{e}_{avg} = 0.10$ or $\hat{e}_{max} = 0.40$ as the criterion.

- 3) Even though the NNT approach is not as effective as the GeNNT approach in most of the above areas, it requires much less computing time which

Table 1: Comparison between GeNNT and NNT

Group	\bar{e}_{rms} (10^{-2})	Approach	Result Errors(10^{-2})			Adequate RS
			\hat{e}_{rms}	\hat{e}_{avg}	\hat{e}_{max}	
A1	10.0	GeNNT	6.35	4.46	23.93	6/6
A2	5.0	GeNNT	3.78	2.66	11.83	6/6
B1	10.0	NNT	17.51	12.64	50.84	1/6
B2	5.0	NNT	13.73	9.28	40.51	4/6
B3	1.0	NNT	12.26	8.32	36.14	4/6

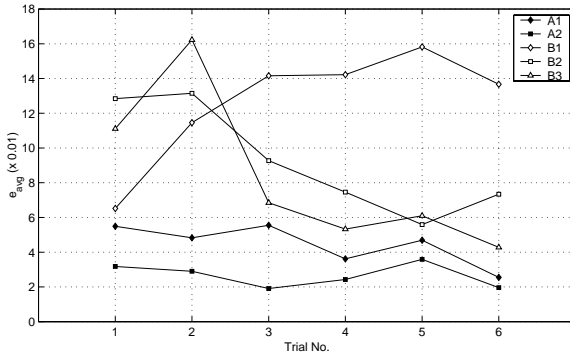


Figure 7: Average Errors of Each Trial

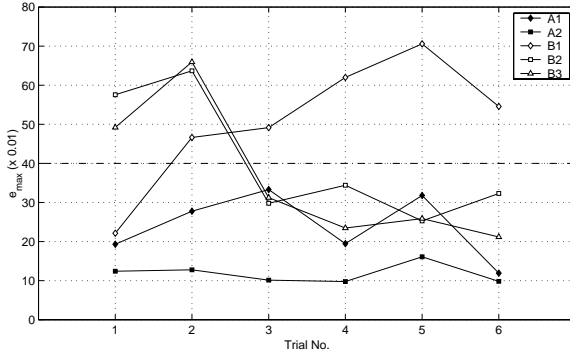


Figure 8: Maximum Errors of Each Trial

is an advantage. So, if one constrains the approximate RS with a more rigorous termination criterion (e.g. the combination of e_{rms} and e_{avg}), one could get improved training results. But it is still difficult for the NNT approach to outperform the GeNNT approach, because the local errors at the training points that are used in the training process have limited influence on the global errors for the NNT approach.

Test Problem 2

This test problem is a response surface approximation of a polynomial with 20 terms and 15 design vari-

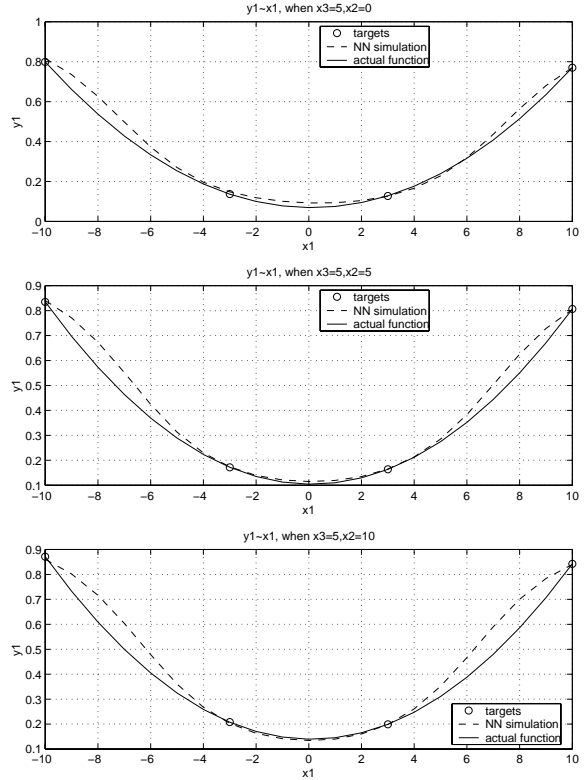


Figure 9: Cross-section Response Surface of y_1 at $x_3 = 5$: A2

ables, a problem size consistent with numerous low-order MDO problems. The analytic expression for the polynomial is,

$$\begin{aligned}
 y = & 5 + 3x_1 + 20x_2 + 6x_3 + x_8 + x_9 + 50x_1^2 \\
 & + 20x_2^2 + 7x_4^2 + 13x_5^2 + x_6^2 + x_7^2 + 28x_{12}^2 \\
 & + x_8^2 + x_1x_2 + x_2x_{10} + 18x_4x_5 + x_9x_{10} \\
 & + 21x_{11}x_{12} + 3x_{13}x_{14} + x_1x_{15}. \quad (30)
 \end{aligned}$$

Figure 11 shows the nonparametric response surface of the polynomial on

$$-10.0 \leq x_1 \leq 10.0, -10.0 \leq x_2 \leq 10.0,$$

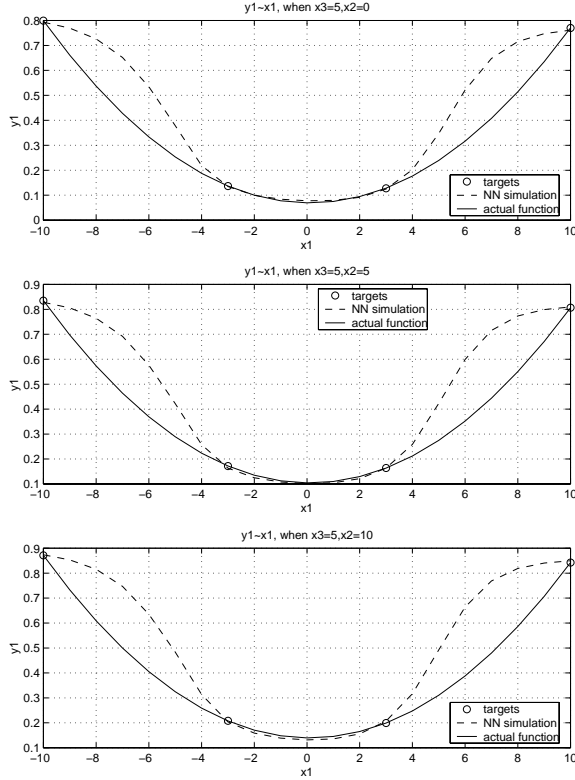


Figure 10: Cross-section Response Surface of y_1 at $x_3 = 5$: B2

where $x_i = 10.0$, $i = 3, \dots, 15$. In all figures hereafter, the functional value of the polynomial in the 15-dimensional design space is linearly scaled and mapped into $[0.1, 0.9]$.

A 15- N_{hidden} -1 network was used to approximate the polynomial, where the number of hidden nodes N_{hidden} is case dependent. The training criterion was $MSE = 0.0025$, or $\bar{e}_{rms} = \sqrt{MSE} = 0.05$. A selected number, 50 and 100 respectively, of training data pairs were randomly chosen from the design space, which was set as $[-10, 10]$ for all x_i , $i = 1, \dots, 15$. This problem was used to explore the effect of selected factors, like the number of the training data pairs, on the performance of the NNT and GeNNT approach. Therefore, four groups, the combinations of different conditions, were developed for comparison. For example, Group ‘‘G-100’’ presents a gradient-enhanced approximation with 100 random training data pairs. In the numerical experiments, five trials were performed for each group, whose results were averaged and compared between the different groups.

The overall characteristics of the response surface in the 15-D design space can not be depicted by a 2-D graphical presentation. The global error values

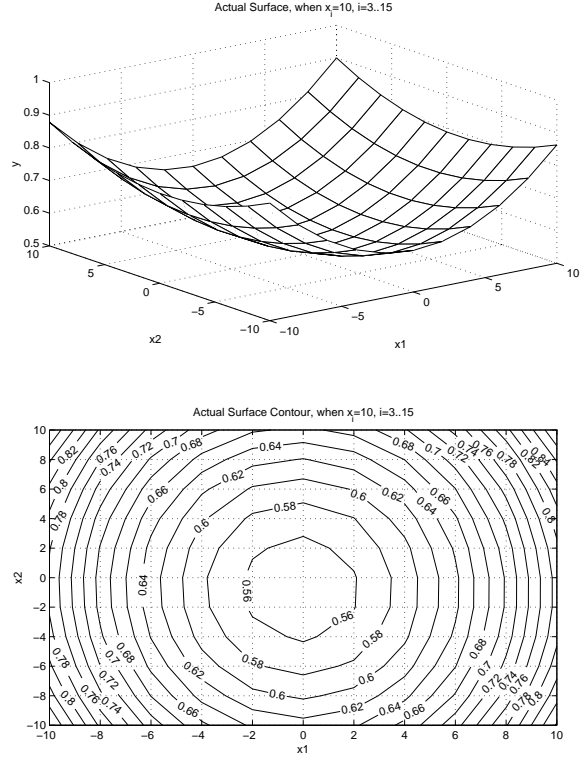


Figure 11: Nonparametric Response Surface

throughout the design space are used for this evaluation. Here, the global error values \hat{e}_{max} , \hat{e}_{rms} and \hat{e}_{avg} , which are defined similarly as in Test Problem 1, were selected as performance criteria. The global errors here were calculated from the errors at 10,000 randomly selected design points in the 15-D design space. To some extent these values are able to show characteristics of the response surface for the whole design space. The average global error values for all trials in each group are presented in Table 2. The global error \hat{e}_{rms} is the most useful in evaluating the overall fitness of approximation to the actual response surface, so discussion will focus on the comparison of \hat{e}_{rms} between the different groups.

One issue regarding the distribution of the training data needs to be addressed here. The error values in Table 2, where \hat{e}_{max} and \hat{e}_{rms} reach 80% and 20% respectively, are rather large in general sense. For this specific demonstration problem, however, it does not imply that the NNT or GeNNT approaches lack the ability to achieve the reasonable response surface approximations. The reason for the large error values is the sparsity of training information in specific regions of the design space. Assume generally only three training points are needed for each dimension to obtain an approximation of a quadratic function with reasonable

Table 2: Comparison between GeNNT and NNT ($\bar{e}_{rms} = 5\%$)

Group	Result Errors(%)			N_{hidden}	Time (s)
	\hat{e}_{max}	\hat{e}_{rms}	\hat{e}_{avg}		
G-50 [†]	70.982	19.494	14.782	3 ~ 4	~ 100
N-50	82.400	29.348	21.924	2 ~ 4	~ 1
G-100	64.906	14.956	11.608	7 ~ 10	~ 1000
N-100	72.130	18.800	14.680	5 ~ 6	~ 10

[†] $\bar{e}_{rms} = 10\%$

accuracy, the $3^{15} = 14,348,907$ training points would be required for this 15-dimensional problem. However, only 100 design points, which are a very small subset of the training data required for an accurate approximation, were used in the neural network training.

Some conclusions can be drawn from Table 2. First, for this somewhat complex problem, the advantage of gradient enhanced approximation over non-gradient-enhanced approximation is not as significant as Test Problem 1. From comparisons between the corresponding groups, like Group G-100 and Group N-100, the most important criterion \hat{e}_{rms} of the GeNNT approach is only 20-35% less than that of the NNT approach. However, that is still a significant improvement in overall performance.

Second, comparing groups of 100 data pairs with groups of 50 data pairs shows there is performance improvement due to the increase in training data pairs. For example, there is about 25% improvement of \hat{e}_{rms} from Group G-50 to Group G-100, and 30% from Group N-50 to Group N-100. So, more training data pairs yield a better approximation as one would expect.

Third, it was not possible to achieve the training criterion $\bar{e}_{rms} = 5\%$ for Group G-50, so $\bar{e}_{rms} = 10\%$ was used for this group. Even so, its performance is still better than Group N-50 and has smaller global error values. Moreover, its performance is very close to that of Group N-100, which shows the potential of the GeNNT approach to develop reasonable approximation accuracy with much less training data.

Last, the gradient-enhanced groups considered in this paper have more complicated NN architectures with more hidden nodes, and they require more computing time, which is a result of the computing algorithm presented in the previous sections.

V. CONCLUSIONS

An approach intended to improve the accuracy of the response surface approximations for application in

an MDO environment has been presented. The approach has developed the response surface approximations based upon artificial neural networks trained using both state and sensitivity information. This approach introduces an error function, which can effectively express the combination of difference of targets and gradients between the true function and neural network approximation. Compared to previous approaches, this approach does not require weighting the residuals of the targets and gradients and is able to approximate gradient-consistent response surfaces with a relatively compact network architecture. The implementation of the approach combined with the Levenberg-Marquardt neural network training algorithm for a two-layer feedforward sigmoid network with an I - J - K architecture was described in detail.

Numerical simulation on selected problems was used to evaluate the approach. The approach has been compared to the non-gradient response surface approximation also based on the neural network training algorithm. Comparisons has been made with the quality of response surfaces approximated and reliability of the algorithms. The results showed that the gradient-enhanced neural network training (GeNNT) approach has superior performance over the non-gradient neural network training (NNT) approach.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the contributions from many discussions provided by Mr. Dhanesh Padmanabhan of the University of Notre Dame. This effort was supported in part by the National Science Foundation under grant DMI98-12857.

REFERENCES

1. R. S. Sellar, J. E. Renaud, and S. M. Batill. Optimization of Mixed Discrete/Continuous Design Variable Systems Using Neural Networks. AIAA Paper 94-4348, 5th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and

- Optimization, Panama City, Florida, September 1994.
2. R.S. Sellar, M.A. Stelmack, S.M. Batill, and J.E. Renaud. Response Surface Approximations for Discipline Coordination in Multidisciplinary Design Optimization. 37th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, AIAA Paper 96-2311, Salt Lake City, Utah, April 1996.
 3. M. Stelmack and S.M. Batill. Neural Network Approximations of Mixed Continuous/Discrete Systems in Multidisciplinary Design. AIAA Paper 98-0916, AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 1998.
 4. M. Stelmack, Nakishima N., and S.M. Batill. Genetic Algorithms for Mixed Discrete/Continuous Optimization in Multidisciplinary Design. AIAA Paper 98-2033, 38th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Long Beach, California, April 1998.
 5. J. Barthelemy and L.E. Hall. Automatic Differentiation as a Tool in Engineering Design. 4th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Cleveland, Ohio, September 1992.
 6. J. Sobieszczanski-Sobieski. Sensitivity of Complex, Internally Coupled Systems. 29th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, 1988.
 7. R. S. Sellar, S. M. Batill, and J. E. Renaud. Concurrent Subspace Optimization Using Gradient-Based Neural Network Response Surface Mappings. AIAA Paper 96-4019, 6th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, Washington, September 1996.
 8. Weiyu Liu. Gradient-enhanced Response Surface Approximations Using Neural Networks Training. M.S. Thesis, University of Notre Dame, 2000.
 9. D.W. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11:431–441, 1963.
 10. M.T. Hagan and M.B. Menhaj. Training Feed-forward Networks with the Marquardt Algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993, 1994.
 11. J.M. Zurada. *Introduction to Artificial Neural Systems*. PWS Publishing Company, 1992.