# PREDICTION OF INDUCED STRESSES WITHIN SOIL MASS USING ARTIFICIAL NEURAL NETWORK

**Zainal, Abdul Kareem Esmat**


## ABSTRACT

In this paper, an Artificial Neural Network (ANN) is applied to predict the soil stress within a soil mass for a variety of depths and displacements under applied loading. The neural network model is created and applied for two cases, point load, and Uniform rectangular load.

Theses two cases were selected among many other cases of loading as a representation of the capabilities of ANN in finding proper solutions. The first case needs one input to get one output, where in the second case we need two inputs, to get one output only.

Results revealed that function approximation using neural network can be applied easily and can give accurate results by choosing the appropriate learning algorithm, number of layers, and number of neurons to solve the problem. ANN model can provide reasonable accuracy for civil engineering problems, and a more effective tool for engineering applications.

الخلاصة

في هذا البحث، تم تطبيق استخدام الشبكات العصبية في محاولة لتخمين قيمة الاجهادات في التربة لأعماق وإزاحات مختلفة تحت موقع تسليط الحمل على سطح التربة. تم إيجاد موديل باستخدام الشبكات العصبية وتطبيقه على حالتين من حالات تسليط الإجهادات وهما الحمل النقطي، والإجهاد المنتظم على شكل مستطيل.

تم اختيار هاتين الحالتين من بين عدة حالات تحميل للتربة كنموذج لبيان إمكانية الشبكات العصبية في إيجاد حلول مناسبة. الحالة الأولى تحتاج إلى مُدخل واحد للحصول على نتيجة واحدة، والحالة الثانية تحتاج إلى مُدخلين اثنين للحصول على نتيجة واحدة فقط.

أسفرت النتائج عن إمكانية استخدام الشبكات العصبية كأداة للحصول على قيم تقريبية للدوال بشكل دقيق مع سهولة الاستخدام وذلك باختيار خوارزمية التعليم المناسبة، عدد الطبقات المناسب، وعدد الخلايا العصبية المناسب لمعالجة المشكلة. إن استخدام الشبكات العصبية في أعمال الهندسة المدنية بشكل عام يعتبر أداة فعالة ومؤثرة لمعالجة بعض المشاكل والتطبيقات التي لا يمكن حلها بالطرق التقليدية إلا بصعوبة.



**Keywords**

Artificial neural network, artificial intelligence, civil engineering, stresses within soil mass, function approximation.


## INTRODUCTION

The origins of artificial neural networks (ANN) are in the field biology. The biological brain consists of billions of highly interconnected neurons forming a neural network. Human information processing depends on this connectionist system of nervous cells. Based on this advantage of information processing, neural networks can easily exploit the massively parallel local processing and distributed storage properties in the brain (Jeng, et al., 2003).

A classical comparison of information processing by a human and a computer is focused on the ability of pattern recognition and learning. The computer can calculate large numbers at high speeds

but it cannot recognize something such as a classification problem, written text, data compression and a learning algorithm. On the contrary, humans easily recognizes and deals with the challenges mentioned above by processing information with highly distributed transformations through thousands of interconnected neurons in the brain.

Generally speaking, an ANN is an informational system simulating the ability of a biological neural network by interconnecting many simple neurons (Fig. 1). The neuron accepts inputs from a single or multiple sources and produces outputs by simple calculations, processing with a predetermined non-linear function. Therefore, the primary characteristics of an ANN can be presented as following: (1) the ability of learning; (2) distributed memory; (3) fault tolerance and (4) operating in parallel.

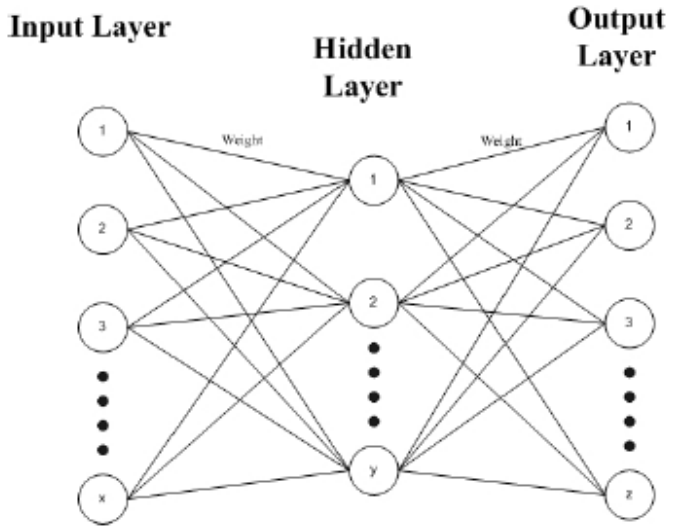



Fig. 1. Structure of Artificial Neural Network model.

Recently, artificial neural network (ANN) models have been widely applied to various relevant civil engineering areas such as geotechnical engineering, water resources and coastal engineering.

## CASES STUDIED

Two cases of loading are studied here to determine the stress within soil mass, the cases are:

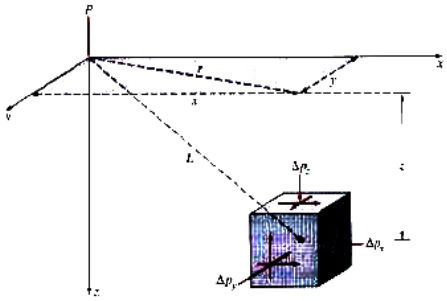
1. Point load (one input, one output),
2. Uniform Rectangular load (two inputs, one output).

To determine the increase of stresses within a soil mass for a specific depth and displacement that results from applying a certain loading condition, we usually use the classical graphs, complex equations, or approximate methods.

Graphical approach gives approximate results, and usually has an error tolerance according to the human judgment of determining the required value, while using the complex equations which are already built on several assumptions may lead to unnecessary excessive effort of calculations.

Using ANN is just like using a memory. Artificial Networks can remember the values that we teach them to memorize (using the appropriate learning algorithm, number of layers, and number of neurons), then we can recall these values back when needed, this process needs to be done only once. With some extensive learning and suitable algorithms, this network can predict values that never been taught before, these predicted values usually have some error tolerance that we try to minimize.

**Case 1 Stresses Inside a Soil Mass Due to Surface Point Load P.**
**Theoretical approach**



$$\Delta p_x = \frac{P}{2\pi}\left\{\frac{3x^2 z}{L^5} - (1-2\mu)\left[\frac{x^2-y^2}{Lr^2(L+z)} + \frac{y^2 z}{L^3 r^2}\right]\right\}$$

$$\Delta p_y = \frac{P}{2\pi}\left\{\frac{3y^2 z}{L^5} - (1-2\mu)\left[\frac{y^2-x^2}{Lr^2(L+z)} + \frac{x^2 z}{L^3 r^2}\right]\right\}$$

$$\Delta p_z = \frac{3P}{2\pi}\frac{z^3}{L^5} = \frac{3P}{2\pi}\frac{z^3}{(r^2+z^2)^{\frac{5}{2}}} = \frac{P}{z^2}\left\{\frac{3}{2\pi}\frac{1}{\left[\left(\frac{r}{z}\right)^2+1\right]^{\frac{5}{2}}}\right\} = \frac{P}{z^2}I_1$$

**Fig 2 Mathematical solution for Point Load**

Boussinesq published in 1883 a mathematical solution to the problem of finding the stress at any point in a homogeneous, elastic and isotropic medium due to a vertical point load P applied upon the surface of a semi-infinitely large space, as shown below (Das, 1998).
For a different ratio of r/z (as defined in eq.1), the value of $I_1$ can be calculated and obtained as shown in the second field of table (1).
The last equation in fig. 2 which is:

$$\Delta P_z = \frac{P}{z^2}\left\{\frac{3}{2\pi}\frac{1}{\left[\left(\frac{r}{z}\right)^2+1\right]^{\frac{5}{2}}}\right\} = \frac{P}{z^2}I_1 \qquad (1)$$

where
P = Point load
z = depth
r = distance from the load
Is used in this case, r/z (which is the only variable here) will represent the input value and $I_1$ will be obtained as the output value. Obtaining $I_1$ makes it possible to calculate $\Delta P_z$ as shown in eq. (1).

**Implementation of ANN**

MatLab version R2008a was used in designing and implementation of the ANN. To find the most appropriate design and learning algorithm, the method of trial and error was used by choosing different learning algorithms, layers, and neurons, as follows:

1. Ten different learning algorithms were used as shown in table A–1.
2. Three different numbers of layers were used, 1 layer, 2 layers, and 3 layers.
3. Four different numbers of neurons were used, 10, 20, 30, and 40 neurons in each layer.

First, the data was calculated using eq. (1) as shown in table (1), the first column (r/z) was used as the input data and the second column ($I_1$) was used as the target data which represents the values that we aim to reach after teaching the neural network. The ratio (r/z) was taken from 0.0 to 4.0 where the calculated values of $I_1$ beyond this range of r/z are very small and can be considered negligible in geotechnical problems.
Verifying data were also prepared, which are random data taken for the ratio r/z and the corresponding values of $I_1$ are calculated as shown in table (2), 75% of these data were for

interpolation where the values of (r/z) are between 0.0 and 4.0, the other 25% are used for extrapolation to examine the reliability of the solution for the values of (r/z) outside the given range. Different number of neurons, networks, and training algorithms (tables A–1, A–2) were tried to get the best solution, which is considered to have:

1. maximum correlation ratio between the target data and the output data obtained,
2. maximum correlation ratio between verifying data and the output obtained, and
3. minimum time to reach solution.

Logsig (log–sigmoid) transfer function was used fig. (3) (Demuth, et al., 2008, Zurada, 1992) for the hidden layers and the output layer. This function gives an output value of $0 \leq a \leq 1$ (eq. 2), which is exactly what we need to obtain the value of the output $I_1$.

The value of ($a$) can be described as:

$$a = f(n) = \frac{1}{1+e^{-\beta n}} \qquad (2)$$

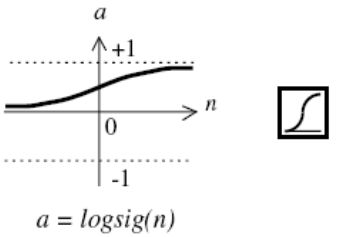



**Fig. 3 Log–Sigmoid Transfer Function**

**Table 1 value of r/z vs. $I_1$ as calculated by eq. (1)**

| r/z | Calculated $I_1$ | r/z | Calculated $I_1$ |
|---|---|---|---|
| 0 | 0.47746 | 2.1 | 0.00701 |
| 0.1 | 0.46573 | 2.2 | 0.00579 |
| 0.2 | 0.43287 | 2.3 | 0.00481 |
| 0.3 | 0.38492 | 2.4 | 0.00402 |
| 0.4 | 0.32946 | 2.5 | 0.00337 |
| 0.5 | 0.27332 | 2.6 | 0.00285 |
| 0.6 | 0.22136 | 2.7 | 0.00241 |
| 0.7 | 0.17619 | 2.8 | 0.00205 |
| 0.8 | 0.13862 | 2.9 | 0.00176 |
| 0.9 | 0.10833 | 3.0 | 0.00151 |
| 1.0 | 0.08440 | 3.1 | 0.00130 |
| 1.1 | 0.06576 | 3.2 | 0.00113 |
| 1.2 | 0.05134 | 3.3 | 0.00098 |
| 1.3 | 0.04023 | 3.4 | 0.00085 |
| 1.4 | 0.03167 | 3.5 | 0.00075 |
| 1.5 | 0.02507 | 3.6 | 0.00066 |
| 1.6 | 0.01997 | 3.7 | 0.00058 |
| 1.7 | 0.01600 | 3.8 | 0.00051 |
| 1.8 | 0.01290 | 3.9 | 0.00045 |
| 1.9 | 0.01046 | 4.0 | 0.00040 |
| 2.0 | 0.00854 | | |

**Table 2 Verifying Data for Case 1**

| r/z | Calculated $I_1$ |
|---|---|
| 0.05 | 0.47449 |
| 0.25 | 0.41032 |
| 0.65 | 0.19559 |
| 1.03 | 0.07831 |
| 1.56 | 0.02186 |
| 1.85 | 0.01161 |
| 2.27 | 0.00508 |
| 2.78 | 0.00212 |
| 3.75 | 0.00054 |
| 4.20 | 0.00032 |
| 4.70 | 0.00019 |
| 5.00 | 0.00014 |

The values of (r/z) were gathered in one vector (the input vector) and the values of $I_1$ are gathered in another vector (the target vector) then the learning process was applied to the network until one of these criteria is met:

- *mse* (Mean Squared Error) of $1 \times 10^{-10}$ is reached, or
- Maximum number of iterations (*Epochs* as called in MATLAB) of 10000 is reached, or
- Minimum gradient of $1 \times 10^{-6}$ is reached, or
- Minimum step size of $1 \times 10^{-6}$ is reached.

Each training algorithm depends on one or more of these criteria to reach the end of the training process and gives the solution, the solution is a trained network that gives the expected output to a specific input value (this process is called *simulation*) .

The results of these tries are shown in table (A–2). Figure (4) summarizes the Maximum correlation ratios and the minimum time to obtain solution in seconds. It is not necessary that the values for the correlation ratio and the minimum time are for the same learning algorithm, but it gives an idea of the optimum performance of each try.

These tries revealed that the most suitable solution for this case is by using (Levenberg-Marquardt with 2 layers 40, 40 neurons) as it gives the most accurate results with minimum time.

The Artificial Network model used was as follows:

1. 1 node as an input layer,
2. 40 nodes as hidden layer 1,
3. 40 nodes as hidden layer 2, and
4. 1 node as an output layer.

The output is shown in table (3) for the simulated data and in table (4) for the verifying data. Time to reach solution was 6 seconds with 9 iterations.

Correlation ratio was calculated to observe the fitness of the simulated data to the calculated data; also the correlation ratio was calculated for the verifying data and the predicted output data. A correlation ratio of 1 and 0.999987812 was obtained for both outputs respectively.

Figure (5) show the data regression between input and target data, figure (6) shows the performance where the value of *mse* was reached in 9 Epochs (iterations), where figure (7) shows the output data, and the verifying data and how they fit to the target data.

**Table 3 r/z, $I_1$ Calculated Values, and $I_1$ Simulated Values for case 1**

| r/z | Calculated $I_1$ | Output Value $I_1$ | r/z | Calculated $I_1$ | Output Value $I_1$ |
|---|---|---|---|---|---|
| 0 | 0.47746 | 0.477463584 | 2.1 | 0.00701 | 0.007013647 |
| 0.1 | 0.46573 | 0.465731470 | 2.2 | 0.00579 | 0.005794614 |
| 0.2 | 0.43287 | 0.432860895 | 2.3 | 0.00481 | 0.004813168 |
| 0.3 | 0.38492 | 0.384911411 | 2.4 | 0.00402 | 0.004020859 |
| 0.4 | 0.32946 | 0.329444563 | 2.5 | 0.00337 | 0.003375093 |
| 0.5 | 0.27332 | 0.273308334 | 2.6 | 0.00285 | 0.002846994 |
| 0.6 | 0.22136 | 0.221351186 | 2.7 | 0.00241 | 0.002413953 |
| 0.7 | 0.17619 | 0.176182178 | 2.8 | 0.00205 | 0.002056527 |
| 0.8 | 0.13862 | 0.138617682 | 2.9 | 0.00176 | 0.001758853 |
| 0.9 | 0.10833 | 0.108323893 | 3.0 | 0.00151 | 0.001510348 |
| 1.0 | 0.08440 | 0.084403204 | 3.1 | 0.00130 | 0.001302479 |
| 1.1 | 0.06576 | 0.065759708 | 3.2 | 0.00113 | 0.001128002 |
| 1.2 | 0.05134 | 0.051341152 | 3.3 | 0.00098 | 0.000980044 |
| 1.3 | 0.04023 | 0.040236740 | 3.4 | 0.00085 | 0.000854248 |
| 1.4 | 0.03167 | 0.031677664 | 3.5 | 0.00075 | 0.000747481 |
| 1.5 | 0.02507 | 0.025078250 | 3.6 | 0.00066 | 0.000656436 |
| 1.6 | 0.01997 | 0.019972652 | 3.7 | 0.00058 | 0.000578020 |
| 1.7 | 0.01600 | 0.016001569 | 3.8 | 0.00051 | 0.000510310 |
| 1.8 | 0.01290 | 0.012901423 | 3.9 | 0.00045 | 0.000452432 |
| 1.9 | 0.01046 | 0.010465445 | 4.0 | 0.00040 | 0.000401220 |
| 2.0 | 0.00854 | 0.008541327 | | | |

**Table 4 Simulated Values for Verifying Data**

| r/z | Calculated $I_1$ | Output Value $I_1$ |
|---|---|---|
| 0.05 | 0.47449 | 0.467757196 |
| 0.25 | 0.41032 | 0.406451657 |
| 0.65 | 0.19559 | 0.192011980 |
| 1.03 | 0.07831 | 0.075212898 |
| 1.56 | 0.02186 | 0.021337445 |
| 1.85 | 0.01161 | 0.011383963 |
| 2.27 | 0.00508 | 0.005019298 |
| 2.78 | 0.00212 | 0.002098753 |
| 3.75 | 0.00054 | 0.000532549 |
| 4.20 | 0.00032 | 0.000366514 |
| 4.70 | 0.00019 | 0.000366513 |
| 5.00 | 0.00014 | 0.000366513 |


**Fig. 4 Max. Correlation Ratios for Case 1(see Table A–2)**

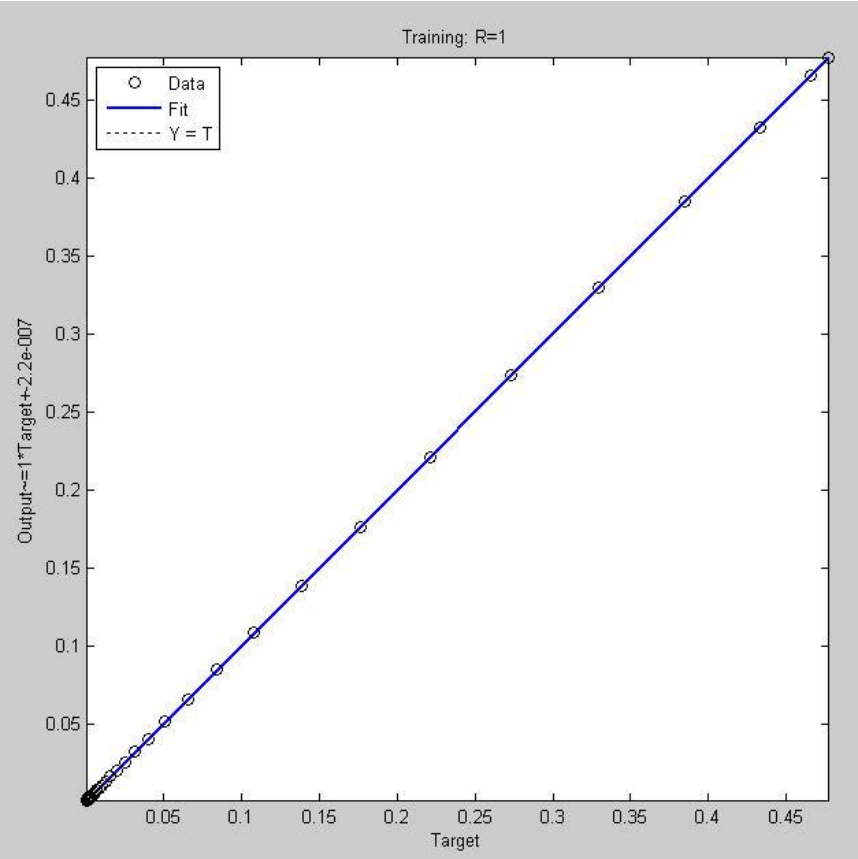



**Fig. 5 Regression for ANN Simulated Data**

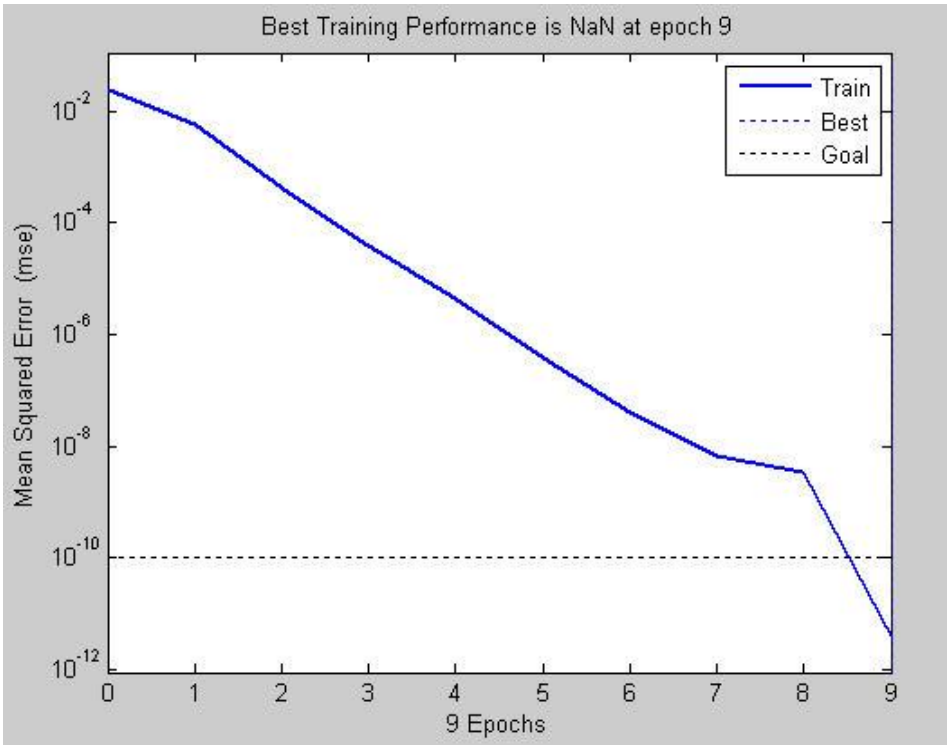



**Fig. 6 Performance for Simulated Data with LM Algorithm**

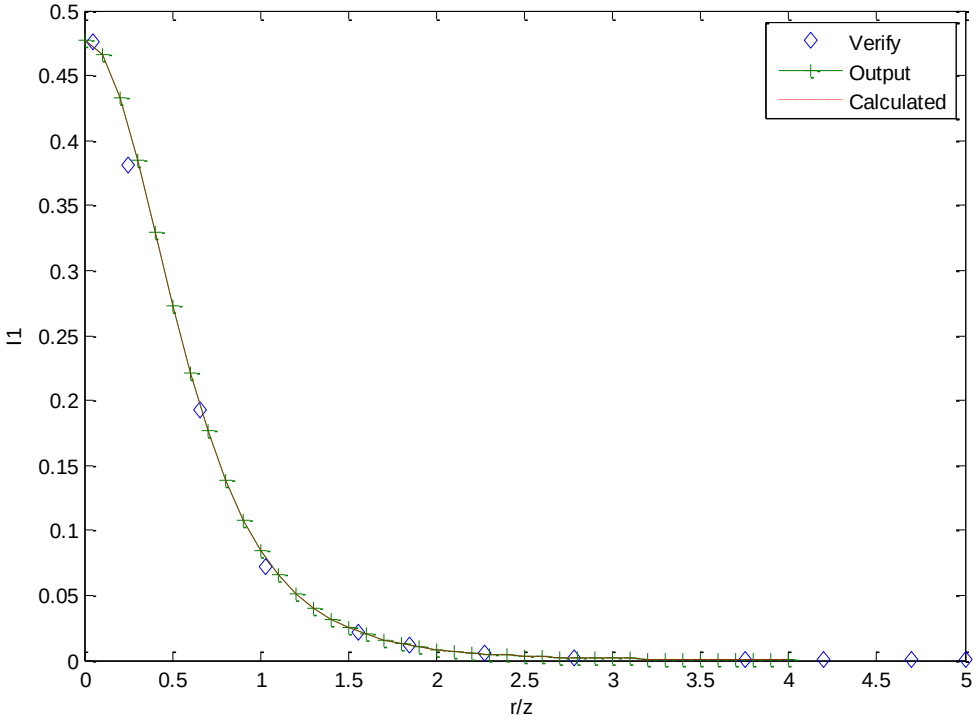



**Fig. 7 (r/z) vs. $I_1$ for Output, Calculated, and Verify Data**

## Case 2 Uniform Rectangular Load
## Theoretical Approach

The increase in the vertical stress at point A is due to the contribution of each *dq* of area (*dxdy*), added up over the length **L** and the width **B**.
Therefore, *dq* = q *dx dy* and the increase in vertical stress is, (Das, 1998)

$$dp = \frac{3q\,dx\,dy\,z^3}{2\pi(x^2+y^2+z^2)^{5/2}}$$

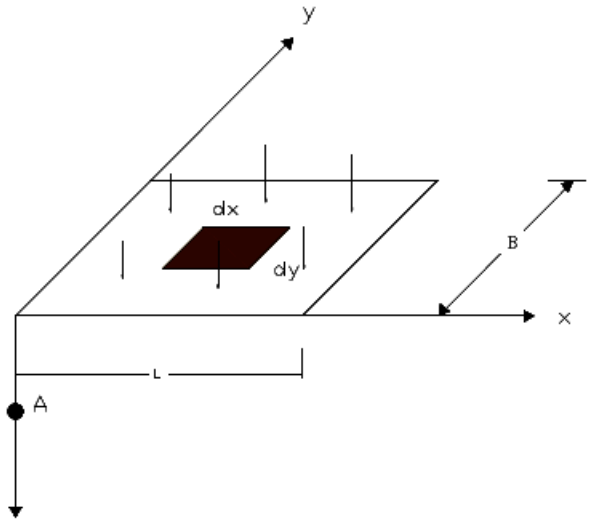



**Fig. 8 Mathematical Solution for Uniform Rectangular Load**

$$\Delta p = \int dp = \int_{y=0}^{B} \int_{x=0}^{L} \frac{3q\, dx\, dy\, z^3}{2\pi (x^2 + y^2 + z^2)^{5/2}} = qI_3$$

*where,*

$$I_3 = \frac{1}{4\pi}\left[\frac{2mn\sqrt{m^2+n^2+1}}{m^2+n^2+m^2n^2+1}\left(\frac{m^2+n^2+2}{m^2+n^2+1}\right) + \tan^{-1}\left(\frac{2mn\sqrt{m^2+n^2+1}}{m^2+n^2-m^2n^2+1}\right)\right] \quad (3)$$

$$m = \frac{B}{z} \quad and \quad n = \frac{L}{z}$$

The increase of the vertical stress $\Delta p_z$ at point A due to the load *dq* can be determined by integration that gives $\Delta p_z = qI_3$, where $I_3$ is defined by eq. (3) for different values of *m* and *n*.
Therefore, finding the value of $I_3$ (through this complex equation (3)) is the key for finding $\Delta p_z$ that represents the increase in the vertical stress under the corner of a rectangular loaded area.

## Implementation of ANN

MatLab version R2008a was also used in implementing the ANN for the uniformly loaded rectangular area. The input data used are shown in table (5). Values of *m* and *n* were transformed into 2 × 400 input matrix holding different combinations of *m* and *n* in each column, and 1 × 400 target vector holding the value of $I_3$ that correspond to the values of *m* and *n* (in this case we need 2 inputs, *m* and *n* (as they are the two variables here), and to get 1 output which is $I_3$). The values of *m* and *n* were taken from 0.0 to 6.0 as shown in the table (5). Verifying data were also prepared as random combinations of *m* and *n* with the corresponding values of $I_3$ as shown in table (7), 75% of these data were taken for interpolation where the values of *m* and *n* are between 0.0 and 6.0, the other 25% are used for extrapolation to examine the reliability of the solution for the values of *m* and *n* outside the given range.
Different number of neurons, networks, and training algorithms (table A–1) were tried to get the best solution, which is also considered to have:

1. maximum correlation ratio between the target data and the output data obtained,
2. maximum correlation ratio between verifying data and the output obtained, and
3. minimum time to reach solution.

Logsig (log–sigmoid) transfer function was also used here as the transfer function for the same reason. The results of these tries are shown in table A–3 and summarized in fig. (9) where a close examination of the values obtained could help in choosing a suitable learning algorithm, number of layers, and number of neurons.
The most suitable algorithm was found to be (**CGB** Conjugate Gradient with Powell/Beale Restarts, with 2 layers, each layer having 10 neurons) as it gave high correlation ratio for both, simulated data and verifying data with little execution time as shown in table (A–3). So, the CGB learning algorithm is considered to be the most promising model with:

1. 2 node as an input layer,
2. 10 nodes as hidden layer 1,
3. 10 nodes as hidden layer 2, and
4. 1 node as an output layer.

With correlation ratio of 0.999928637 for simulated data and 0.999904893 for verifying data, with 22 seconds needed to complete the solution.
This compromization of choosing the most suitable solution is absolutely a human judgment and may differ according to the accuracy needed and the type of problem solved.

- **RESULTS AND DISCUSSION**

Results are shown in tables (3, 4, 6, and 7) and figures (5, 6, 7, and 10). In general, the correlation ratio for all tries show a great relationship between the results obtained and the target values (the calculated values). A little decrease in correlation ratio was observed for the verifying data results.

Verifying data was further investigated, and a few samples were taken randomly for analysis for both cases where the 75% interpolation data were separated from the 25% extrapolation data and the correlation ratio was examined for each case separately.

For case 1, it was observed that the interpolation results agree very much with the calculated data (higher correlation ratio) where the extrapolation data have relatively lower correlation ratio, which indicates that the decrease in total correlation ratio is mainly due to the extrapolation data results.

For case 2, after examining the interpolation data separately from the extrapolation data, there was no indication that the decrease in total correlation ratio is due to extrapolation only, but it is the result of the contribution of both interpolation and extrapolation data.

Generally, the results reveal that the use of ANN gives very good agreement between calculated data and simulated data which makes the ANN a very good tool for function approximation method to obtain results, it is also very good for obtaining interpolation results, but less dependant on obtaining extrapolation results on some situations.

**Table 5 Input Data for case 2**

Table with the variation of $I_3$ with $m$ and $n$ for a rectangular loaded area.

| | | | | | | | | | *m* | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *n* | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 | 1.2 | 1.4 | 1.6 | 1.8 | 2.0 | 2.5 | 3.0 | 4.0 | 5.0 | 6.0 |
| 0.1 | 0.0047 | 0.0092 | 0.0132 | 0.0168 | 0.0198 | 0.0222 | 0.0242 | 0.0258 | 0.0270 | 0.0279 | 0.0293 | 0.0301 | 0.0306 | 0.0309 | 0.0311 | 0.0314 | 0.0315 | 0.0316 | 0.0316 | 0.0316 |
| 0.2 | 0.0092 | 0.0179 | 0.0259 | 0.0387 | 0.0387 | 0.0435 | 0.0474 | 0.0504 | 0.0528 | 0.0547 | 0.0573 | 0.0589 | 0.0599 | 0.0606 | 0.0610 | 0.0616 | 0.0618 | 0.0619 | 0.0620 | 0.0620 |
| 0.3 | 0.0132 | 0.0259 | 0.0374 | 0.0474 | 0.0559 | 0.0629 | 0.0686 | 0.0731 | 0.0766 | 0.0794 | 0.0832 | 0.0856 | 0.0871 | 0.0880 | 0.8870 | 0.0895 | 0.0898 | 0.0901 | 0.0901 | 0.0902 |
| 0.4 | 0.0168 | 0.0328 | 0.0474 | 0.0602 | 0.0711 | 0.0801 | 0.0873 | 0.0931 | 0.0977 | 0.1013 | 0.1063 | 0.1094 | 0.1140 | 0.1126 | 0.1134 | 0.1145 | 0.1150 | 0.1153 | 0.1154 | 0.1154 |
| 0.5 | 0.0198 | 0.0387 | 0.0559 | 0.0711 | 0.0840 | 0.0947 | 0.1034 | 0.1104 | 0.1158 | 0.1202 | 0.1263 | 0.1300 | 0.1324 | 0.1340 | 0.1350 | 0.1363 | 0.1368 | 0.1372 | 0.1374 | 0.1374 |
| 0.6 | 0.0222 | 0.0435 | 0.0629 | 0.0801 | 0.0947 | 0.1069 | 0.1168 | 0.1247 | 0.1311 | 0.1361 | 0.1431 | 0.1475 | 0.1503 | 0.1521 | 0.1533 | 0.1548 | 0.1555 | 0.1560 | 0.1561 | 0.1562 |
| 0.7 | 0.0242 | 0.0474 | 0.0686 | 0.0873 | 0.1034 | 0.1169 | 0.1277 | 0.1365 | 0.1436 | 0.1491 | 0.1570 | 0.1620 | 0.1652 | 0.1672 | 0.1686 | 0.1704 | 0.1711 | 0.1717 | 0.1719 | 0.1719 |
| 0.8 | 0.0258 | 0.0504 | 0.0731 | 0.0931 | 0.1104 | 0.1247 | 0.1365 | 0.1461 | 0.1537 | 0.1598 | 0.1684 | 0.1739 | 0.1774 | 0.1797 | 0.1812 | 0.1832 | 0.1841 | 0.1847 | 0.1849 | 0.1850 |
| 0.9 | 0.0270 | 0.0528 | 0.0766 | 0.0977 | 0.1158 | 0.1311 | 0.1436 | 0.1537 | 0.1619 | 0.1684 | 0.1777 | 0.1836 | 0.1874 | 0.1899 | 0.1915 | 0.1938 | 0.1947 | 0.1954 | 0.2956 | 0.1957 |
| 1.0 | 0.0279 | 0.0547 | 0.0794 | 0.1013 | 0.1202 | 0.1361 | 0.1491 | 0.1598 | 0.1684 | 0.1752 | 0.1851 | 0.1914 | 0.1955 | 0.1981 | 0.1999 | 0.2024 | 0.2034 | 0.2042 | 0.2044 | 0.2045 |
| 1.2 | 0.0293 | 0.0573 | 0.0832 | 0.1063 | 0.1263 | 0.1431 | 0.1570 | 0.1684 | 0.1777 | 0.1851 | 0.1958 | 0.2028 | 0.2073 | 0.2103 | 0.2124 | 0.2151 | 0.2163 | 0.2172 | 0.2175 | 0.2176 |
| 1.4 | 0.0301 | 0.0589 | 0.0856 | 0.1094 | 0.1300 | 0.1475 | 0.1620 | 0.1739 | 0.1836 | 0.1914 | 0.2028 | 0.2102 | 0.2151 | 0.2184 | 0.2206 | 0.2236 | 0.2250 | 0.2260 | 0.2263 | 0.2264 |
| 1.6 | 0.0306 | 0.0599 | 0.0871 | 0.1114 | 0.1324 | 0.1503 | 0.1652 | 0.1774 | 0.1874 | 0.1955 | 0.2073 | 0.2151 | 0.2203 | 0.2237 | 0.2261 | 0.2294 | 0.2309 | 0.2320 | 0.2323 | 0.2325 |
| 1.8 | 0.0309 | 0.0606 | 0.0880 | 0.1126 | 0.1340 | 0.1521 | 0.1672 | 0.1797 | 0.1899 | 0.1981 | 0.2103 | 0.2183 | 0.2237 | 0.2274 | 0.2299 | 0.2333 | 0.2350 | 0.2362 | 0.2366 | 0.2367 |
| 2.0 | 0.0311 | 0.0610 | 0.0887 | 0.1134 | 0.1350 | 0.1533 | 0.1686 | 0.1812 | 0.1915 | 0.1999 | 0.2124 | 0.2206 | 0.2261 | 0.2299 | 0.2325 | 0.2361 | 0.2378 | 0.2391 | 0.2395 | 0.2397 |
| 2.5 | 0.0314 | 0.0616 | 0.0895 | 0.1145 | 0.1363 | 0.1548 | 0.1704 | 0.1832 | 0.1938 | 0.2024 | 0.2151 | 0.2236 | 0.2294 | 0.2333 | 0.2361 | 0.2401 | 0.2420 | 0.2434 | 0.2439 | 0.2441 |
| 3.0 | 0.0315 | 0.0618 | 0.0898 | 0.1150 | 0.1368 | 0.1555 | 0.1711 | 0.1841 | 0.1947 | 0.2034 | 0.2163 | 0.2250 | 0.2309 | 0.2350 | 0.2378 | 0.2420 | 0.2439 | 0.2455 | 0.2461 | 0.2463 |
| 4.0 | 0.0316 | 0.0619 | 0.0901 | 0.1153 | 0.1372 | 0.1560 | 0.1717 | 0.1847 | 0.1954 | 0.2042 | 0.2172 | 0.2260 | 0.2320 | 0.2362 | 0.2391 | 0.2434 | 0.2455 | 0.2472 | 0.2479 | 0.2481 |
| 5.0 | 0.0316 | 0.0620 | 0.0901 | 0.1154 | 0.1374 | 0.1561 | 0.1719 | 0.1849 | 0.1956 | 0.2044 | 0.2175 | 0.2263 | 0.2324 | 0.2366 | 0.2395 | 0.2439 | 0.2460 | 0.2479 | 0.2486 | 0.2489 |
| 6.0 | 0.0316 | 0.0620 | 0.0902 | 0.1154 | 0.1374 | 0.1562 | 0.1719 | 0.1850 | 0.1957 | 0.2045 | 0.2176 | 0.2264 | 0.2325 | 0.2367 | 0.2397 | 0.2441 | 0.2463 | 0.2482 | 0.2489 | 0.2492 |

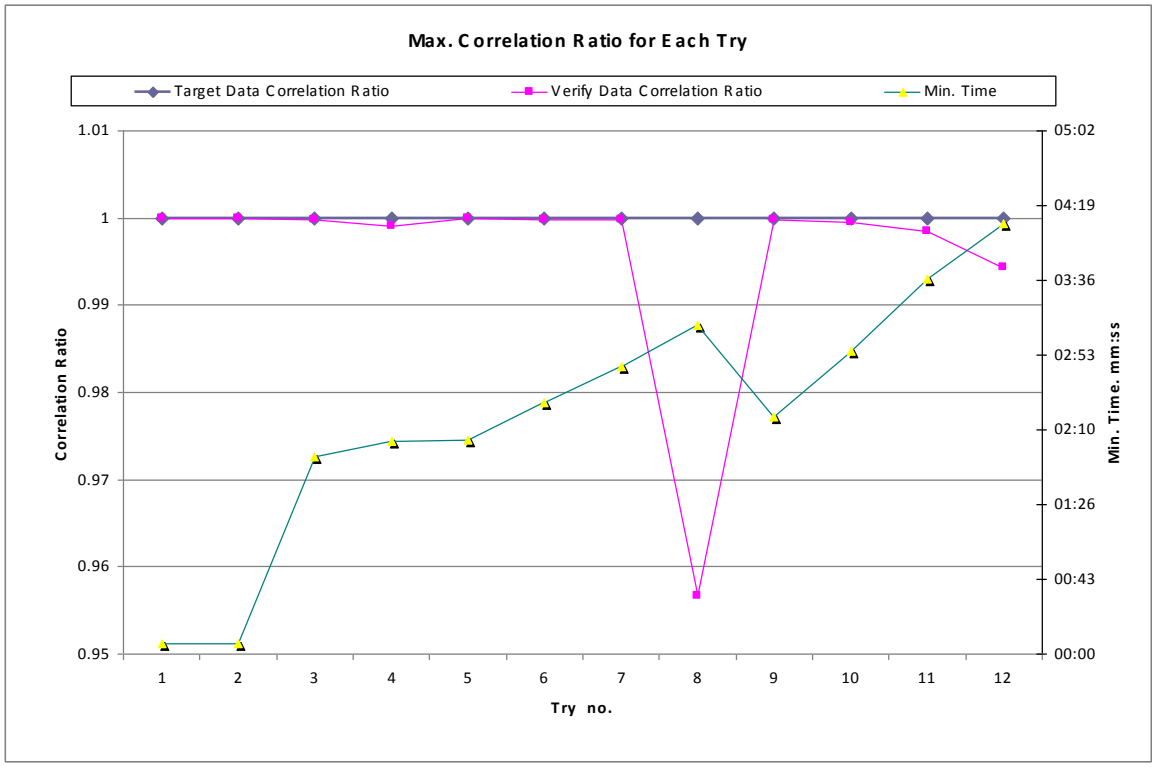



**Fig. 9 Correlation Ratios for Case 2 (Table A–3)**

## Table 6 Output Data (Simulated Data) for case 2

| *n* \ *m* | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 | 1.2 | 1.4 | 1.6 | 1.8 | 2.0 | 2.5 | 3.0 | 4.0 | 5.0 | 6.0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.1 | 0.0061 | 0.0102 | 0.0142 | 0.0177 | 0.0205 | 0.0230 | 0.0250 | 0.0266 | 0.0279 | 0.0288 | 0.0302 | 0.0310 | 0.0316 | 0.0321 | 0.0325 | 0.0332 | 0.0334 | 0.0315 | 0.0330 | 0.0328 |
| 0.2 | 0.0105 | 0.0184 | 0.0261 | 0.0326 | 0.0380 | 0.0426 | 0.0463 | 0.0493 | 0.0516 | 0.0534 | 0.0558 | 0.0573 | 0.0582 | 0.0589 | 0.0595 | 0.0606 | 0.0607 | 0.0590 | 0.0610 | 0.0605 |
| 0.3 | 0.0148 | 0.0265 | 0.0381 | 0.0479 | 0.0561 | 0.0630 | 0.0686 | 0.0732 | 0.0767 | 0.0795 | 0.0831 | 0.0853 | 0.0867 | 0.0877 | 0.0885 | 0.0897 | 0.0899 | 0.0891 | 0.0906 | 0.0898 |
| 0.4 | 0.0183 | 0.0333 | 0.0482 | 0.0610 | 0.0716 | 0.0806 | 0.0879 | 0.0938 | 0.0985 | 0.1022 | 0.1071 | 0.1100 | 0.1119 | 0.1131 | 0.1140 | 0.1154 | 0.1159 | 0.1160 | 0.1167 | 0.1157 |
| 0.5 | 0.0210 | 0.0387 | 0.0563 | 0.0715 | 0.0843 | 0.0949 | 0.1036 | 0.1107 | 0.1164 | 0.1208 | 0.1268 | 0.1305 | 0.1327 | 0.1342 | 0.1352 | 0.1368 | 0.1376 | 0.1384 | 0.1383 | 0.1372 |
| 0.6 | 0.0232 | 0.0430 | 0.0630 | 0.0802 | 0.0946 | 0.1066 | 0.1165 | 0.1246 | 0.1310 | 0.1361 | 0.1431 | 0.1474 | 0.1500 | 0.1517 | 0.1528 | 0.1545 | 0.1559 | 0.1567 | 0.1563 | 0.1552 |
| 0.7 | 0.0249 | 0.0466 | 0.0684 | 0.0873 | 0.1031 | 0.1163 | 0.1272 | 0.1361 | 0.1432 | 0.1488 | 0.1567 | 0.1615 | 0.1645 | 0.1664 | 0.1677 | 0.1695 | 0.1713 | 0.1719 | 0.1714 | 0.1705 |
| 0.8 | 0.0263 | 0.0495 | 0.0730 | 0.0933 | 0.1102 | 0.1243 | 0.1360 | 0.1456 | 0.1533 | 0.1595 | 0.1682 | 0.1735 | 0.1768 | 0.1789 | 0.1802 | 0.1822 | 0.1845 | 0.1846 | 0.1842 | 0.1834 |
| 0.9 | 0.0274 | 0.0518 | 0.0767 | 0.0982 | 0.1161 | 0.1310 | 0.1433 | 0.1535 | 0.1617 | 0.1683 | 0.1777 | 0.1835 | 0.1871 | 0.1893 | 0.1908 | 0.1929 | 0.1955 | 0.1951 | 0.1950 | 0.1944 |
| 1.0 | 0.0283 | 0.0537 | 0.0796 | 0.1021 | 0.1208 | 0.1363 | 0.1492 | 0.1598 | 0.1685 | 0.1755 | 0.1855 | 0.1917 | 0.1956 | 0.1980 | 0.1995 | 0.2017 | 0.2046 | 0.2038 | 0.2041 | 0.2038 |
| 1.2 | 0.0294 | 0.0561 | 0.0836 | 0.1074 | 0.1272 | 0.1436 | 0.1573 | 0.1686 | 0.1780 | 0.1856 | 0.1966 | 0.2036 | 0.2080 | 0.2107 | 0.2124 | 0.2147 | 0.2177 | 0.2170 | 0.2180 | 0.2180 |
| 1.4 | 0.0300 | 0.0574 | 0.0857 | 0.1103 | 0.1308 | 0.1478 | 0.1620 | 0.1739 | 0.1837 | 0.1917 | 0.2036 | 0.2112 | 0.2159 | 0.2188 | 0.2206 | 0.2230 | 0.2258 | 0.2260 | 0.2275 | 0.2276 |
| 1.6 | 0.0304 | 0.0581 | 0.0868 | 0.1120 | 0.1329 | 0.1503 | 0.1649 | 0.1770 | 0.1872 | 0.1955 | 0.2080 | 0.2160 | 0.2211 | 0.2241 | 0.2260 | 0.2284 | 0.2308 | 0.2322 | 0.2338 | 0.2339 |
| 1.8 | 0.0308 | 0.0587 | 0.0876 | 0.1130 | 0.1342 | 0.1519 | 0.1667 | 0.1791 | 0.1894 | 0.1980 | 0.2108 | 0.2192 | 0.2244 | 0.2277 | 0.2296 | 0.2321 | 0.2343 | 0.2364 | 0.2378 | 0.2380 |
| 2.0 | 0.0313 | 0.0593 | 0.0883 | 0.1138 | 0.1352 | 0.1530 | 0.1680 | 0.1805 | 0.1910 | 0.1997 | 0.2127 | 0.2213 | 0.2267 | 0.2300 | 0.2321 | 0.2346 | 0.2371 | 0.2393 | 0.2405 | 0.2407 |
| 2.5 | 0.0325 | 0.0608 | 0.0900 | 0.1156 | 0.1370 | 0.1549 | 0.1699 | 0.1825 | 0.1930 | 0.2017 | 0.2150 | 0.2238 | 0.2295 | 0.2332 | 0.2357 | 0.2391 | 0.2414 | 0.2430 | 0.2437 | 0.2439 |
| 3.0 | 0.0336 | 0.0620 | 0.0910 | 0.1165 | 0.1379 | 0.1557 | 0.1706 | 0.1830 | 0.1935 | 0.2022 | 0.2156 | 0.2248 | 0.2311 | 0.2353 | 0.2381 | 0.2422 | 0.2429 | 0.2444 | 0.2449 | 0.2453 |
| 4.0 | 0.0329 | 0.0605 | 0.0894 | 0.1154 | 0.1376 | 0.1562 | 0.1716 | 0.1845 | 0.1952 | 0.2042 | 0.2177 | 0.2269 | 0.2333 | 0.2376 | 0.2401 | 0.2423 | 0.2441 | 0.2460 | 0.2474 | 0.2496 |
| 5.0 | 0.0330 | 0.0601 | 0.0890 | 0.1156 | 0.1383 | 0.1570 | 0.1722 | 0.1844 | 0.1945 | 0.2031 | 0.2166 | 0.2261 | 0.2326 | 0.2371 | 0.2403 | 0.2441 | 0.2448 | 0.2469 | 0.2493 | 0.2490 |
| 6.0 | 0.0330 | 0.0612 | 0.0900 | 0.1156 | 0.1375 | 0.1561 | 0.1719 | 0.1851 | 0.1960 | 0.2048 | 0.2178 | 0.2265 | 0.2325 | 0.2366 | 0.2394 | 0.2436 | 0.2463 | 0.2489 | 0.2486 | 0.2491 |

**Table 7 Verifying Data (Calculated and Simulated)**

| *m* | *n* | Calculated Data | Simulated Data |
|---|---|---|---|
| 0.05 | 0.05 | 0.001188709 | 0.003061156 |
| 1.10 | 2.20 | 0.208105692 | 0.208031476 |
| 2.40 | 0.75 | 0.176836343 | 0.175824553 |
| 1.03 | 0.85 | 0.165927416 | 0.165769742 |
| 1.56 | 0.30 | 0.086847463 | 0.086495926 |
| 1.90 | 4.30 | 0.237935443 | 0.238111898 |
| 2.25 | 4.70 | 0.242047436 | 0.241577197 |
| 3.78 | 2.50 | 0.243242079 | 0.242648952 |
| 4.45 | 3.10 | 0.246134235 | 0.244890620 |
| 6.40 | 2.50 | 0.244163366 | 0.244000887 |
| 7.30 | 3.60 | 0.247790018 | 0.250191178 |
| 8.00 | 0.11 | 0.034731558 | 0.032693384 |

- **CONCLUSIONS AND RECOMMENDATIONS**

ANN implementation in geotechnical problems as a function approximation for predicting stresses in soil mass is found to be very good. The results obtained for the two cases of loading gave a correlation ratio of not less than 0.95 for the simulated data in the worst case.

As mentioned earlier, ANN is just like memory that can be taught some information to memorize then we can recall these information back. Therefore, with the appropriate learning rules and a suitable number of neurons and layers, a very good solution can be obtained in solving geotechnical problems, as one good trained network can remember a lot of information and can be used for prediction in a way that is similar to the human brain.

This method can be combined with more complex artificial intelligence and expert systems to solve more complex problems in geotechnical engineering, and generally civil engineering problems.

The following recommendations could be of great benefits for future work:

1- Finding a solution for more complex functions solving geotechnical problems.
2- Combining more cases of stress loading with one network where this network can give a solution for more than one case of loading.
3- Using more fields of Artificial Intelligence and expert systems to find solutions for geotechnical problems.

**REFERENCES**


- Das, B.M., 1998, "Principles of Geotechnical Engineering", PWS Publishing Company, Boston.
- Demuth, H., Mark B., and Martin H., 2008, " Neural Network Toolbox™ 6 User's Guide", The MathWorks, Inc.
- Jeng, D.S., Cha D. H. and Blumenstein M., 2003, "Application of Neural Network in Civil Engineering Problems", Proceedings of the International Conference on Advances in the Internet, Processing, Systems and Interdisciplinary Research (IPSI-2003).
- Zurada, J. M., 1992, "Introduction to Artificial Neural Systems", PWS Publishing Company.

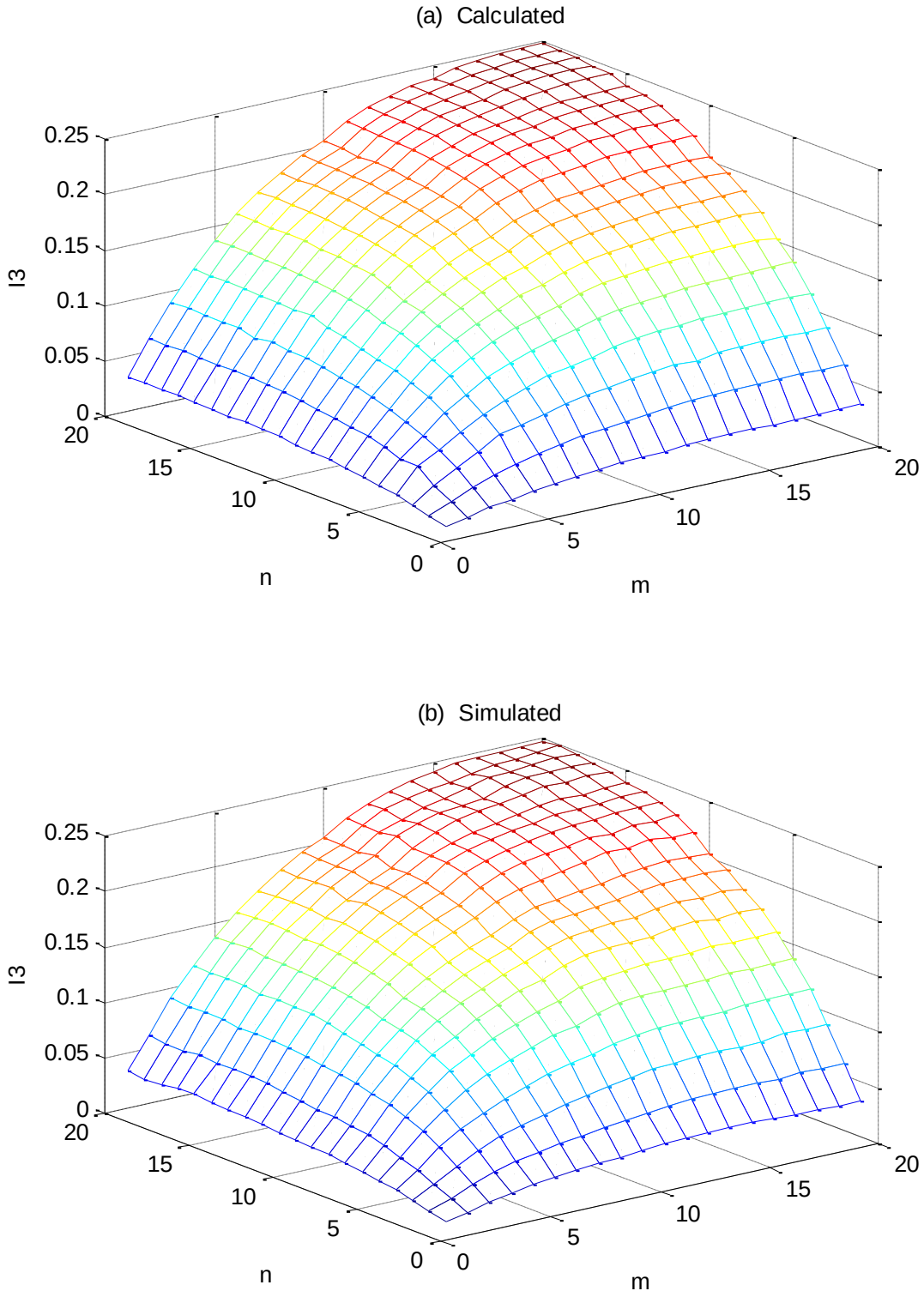



**Fig. 10 a) Calculated Data, b) Simulated Data Using CGB 10,10 Neurons**

Appendix A

This appendix contains three tables, table A–1 is a list of ten learning algorithms and their abbreviations as used by MATLAB (Demuth, et. al., 2008). These abbreviations were used in table A–2 and A–3 to show the performance of each learning algorithm when tested for a specified number of layers and neurons.

**Table A–1 Training Algorithms Names and Symbols**

| Symbol | Algorithm Name |
|---|---|
| GDA | Backpropagation training with an adaptive learning rate |
| GDX | adaptive learning rate with momentum training |
| RP | Resilient Backpropagation |
| CGF | Fletcher-Powell Conjugate Gradient |
| CGP | Polak-Ribiére Conjugate Gradient |
| LM | Levenberg-Marquardt |
| BFG | BFGS Quasi-Newton |
| SCG | Scaled Conjugate Gradient |
| CGB | Conjugate Gradient with Powell/Beale Restarts |
| OSS | One Step Secant |

**Table A–2 Correlation Ratio and Time to Obtain Solution for Each Learning Algorithm, Case 1**

| Size | | GDX | LM | BFG | RP | CGP | CGF |
|---|---|---|---|---|---|---|---|
| 1<br>1 Layer<br>10 Neurons | simulated data | 0.999654268 | 0.999999998 | 0.999974471 | 0.999994856 | 0.999689497 | 0.998935533 |
| | verify data | 0.999564824 | 0.999998664 | 0.999985423 | 0.999992449 | 0.999697215 | 0.99748201 |
| | Time to reach Solution mm:ss | 01:13 | 00:05 | 00:02 | 01:30 | 00:02 | 00:01 |
| 2<br>1 Layer<br>20 Neurons | simulated data | 0.999887993 | 0.999999998 | 0.999969126 | 0.999998361 | 0.994836809 | 0.999975446 |
| | verify data | 0.99991002 | 0.999994758 | 0.999883621 | 0.999565628 | 0.998366133 | 0.999942052 |
| | Time to reach Solution mm:ss | 01:13 | 00:05 | 00:02 | 01:30 | 00:01 | 00:04 |
| 3<br>1 Layer<br>30 Neurons | simulated data | 0.999903148 | 0.999999998 | 0.999982061 | 0.999998876 | 0.999985799 | 0.999979182 |
| | verify data | 0.999657258 | 0.999725836 | 0.999817677 | 0.999124495 | 0.999853799 | 0.999849619 |
| | Time to reach Solution mm:ss | 01:13 | 00:01 | 00:02 | 01:31 | 03:15 | 00:05 |
| 4<br>1 Layer<br>40 Neurons | simulated data | 0.999925388 | 0.999999998 | 0.999982977 | 0.999999976 | 0.999977561 | 0.999827721 |
| | verify data | 0.999868643 | 0.992037779 | 0.999846374 | 0.998608843 | 0.997504938 | 0.999472775 |
| | Time to reach Solution mm:ss | 01:15 | 00:01 | 00:03 | 00:17 | 01:30 | 00:03 |
| 5<br>2 Layers<br>10 x 10<br>Neurons | simulated data | 0.999886681 | 0.999999997 | 0.999977361 | 0.999999359 | 0.970815555 | 0.999993597 |
| | verify data | 0.999936907 | 0.999998713 | 0.999991644 | 0.999996637 | 0.979495316 | 0.999994631 |
| | Time to reach Solution mm:ss | 01:19 | 00:07 | 00:07 | 01:37 | 00:01 | 00:08 |
| 6<br>2 Layers<br>20 x 20<br>Neurons | simulated data | 0.999952208 | 1 | 0.999982813 | 0.999999997 | 0.999988324 | 0.999996521 |
| | verify data | 0.999880492 | 0.999951412 | 0.999955936 | 0.999903009 | 0.999950285 | 0.999961913 |
| | Time to reach Solution mm:ss | 01:22 | 00:01 | 00:57 | 00:23 | 00:08 | 00:17 |
| 7<br>2 Layers<br>30 x 30<br>Neurons | simulated data | 0.999978976 | 1 | 0.999997074 | 0.999999998 | 0.999995185 | 0.996383211 |
| | verify data | 0.999430234 | 0.999661826 | 0.999859665 | 0.99790594 | 0.999716004 | 0.996033786 |
| | Time to reach Solution mm:ss | 01:34 | 00:03 | 07:17 | 00:01 | 00:06 | 00:01 |
| 8<br>2 Layers<br>40 x 40<br>Neurons | simulated data | 0.999985139 | 1 | 0.506677097 | 0.999999997 | 0.999987578 | 0.999998062 |
| | verify data | 0.999780676 | 0.999987812 | 0.697834879 | 0.996756997 | 0.99990708 | 0.999912867 |
| | Time to reach Solution mm:ss | 01:26 | 00:06 | 00:28 | 00:14 | 00:05 | 00:26 |
| 9<br>3 Layers<br>10 x 10 x 10<br>Neurons | simulated data | 0.99998686 | 0.999999997 | 0.999988897 | 0.999999993 | 0.999981582 | 0.999997282 |
| | verify data | 0.999988507 | 0.99999969 | 0.999310864 | 0.999995224 | 0.999965553 | 0.999997825 |
| | Time to reach Solution mm:ss | 01:31 | 00:08 | 00:33 | 01:45 | 00:09 | 00:19 |
| 10<br>3 Layers<br>20 x 20 x 20<br>Neurons | simulated data | 0.999967055 | 0.999999998 | 0.99998969 | 0.999999997 | 0.999996346 | 0.999997943 |
| | verify data | 0.999896482 | 0.999939485 | 0.999980226 | 0.999875488 | 0.999955602 | 0.999966711 |
| | Time to reach Solution mm:ss | 01:30 | 00:16 | 07:04 | 00:03 | 00:11 | 00:19 |
| 11<br>3 Layers<br>30 x 30 x 30<br>Neurons | simulated data | 0.999993711 | 1 | 0.999998249 | 0.999999998 | 0.999997149 | 0.999999645 |
| | verify data | 0.999506264 | 0.999275545 | 0.999604536 | 0.999122072 | 0.999470174 | 0.999626002 |
| | Time to reach Solution mm:ss | 02:09 | 00:24 | 36:29 | 00:01 | 00:09 | 00:33 |
| 12<br>3 Layers<br>40 x 40 x 40<br>Neurons | simulated data | 0.999971275 | 1 | 0.999264321 | 0.999999998 | 0.99999236 | 0.999999724 |
| | verify data | 0.999928194 | 0.999944212 | 0.998731643 | 0.999069412 | 0.999913901 | 0.999712451 |
| | Time to reach Solution mm:ss | 01:45 | 00:38 | 39:27 | 00:02 | 00:06 | 00:26 |

## Table A–2 Continued

| Size | | GDA | CGB | SCG | OSS | Max. Value Min Time |
|---|---|---|---|---|---|---|
| 1<br>1 Layer<br>10 Neurons | simulated data | 0.999666514 | 0.998761755 | 0.999925306 | 0.999935355 | 0.999999998 |
| | verify data | 0.999955245 | 0.998627971 | 0.999924218 | 0.999978577 | 0.999998664 |
| | Time to reach Solution mm:ss | 01:15 | 00:01 | 00:04 | 03:12 | 00:01 |
| 2<br>1 Layer<br>20 Neurons | simulated data | 0.999591775 | 0.999847813 | 0.999979631 | 0.999997225 | 0.999999998 |
| | verify data | 0.999663038 | 0.999891158 | 0.999962418 | 0.999926147 | 0.999994758 |
| | Time to reach Solution mm:ss | 01:15 | 00:01 | 00:06 | 02:53 | 00:01 |
| 3<br>1 Layer<br>30 Neurons | simulated data | 0.999936424 | 0.99997307 | 0.999995115 | 0.999998675 | 0.999999998 |
| | verify data | 0.999852982 | 0.999782103 | 0.999847051 | 0.999821996 | 0.999853799 |
| | Time to reach Solution mm:ss | 01:20 | 00:03 | 00:04 | 02:55 | 00:03 |
| 4<br>1 Layer<br>40 Neurons | simulated data | 0.99969054 | 0.999992138 | 0.999994962 | 0.999998074 | 0.999999998 |
| | verify data | 0.999814941 | 0.999138924 | 0.997849328 | 0.999845586 | 0.999868643 |
| | Time to reach Solution mm:ss | 01:02 | 00:11 | 00:09 | 02:41 | 00:01 |
| 5<br>2 Layers<br>10 x 10<br>Neurons | simulated data | 0.999734388 | 0.999866841 | 0.999989427 | 0.999996428 | 0.999999997 |
| | verify data | 0.992158466 | 0.999841064 | 0.999975637 | 0.999994309 | 0.999998713 |
| | Time to reach Solution mm:ss | 01:06 | 00:02 | 00:07 | 03:00 | 00:01 |
| 6<br>2 Layers<br>20 x 20<br>Neurons | simulated data | 0.999902432 | 0.999984293 | 0.999996627 | 0.999996659 | 1 |
| | verify data | 0.999902629 | 0.99994639 | 0.999906829 | 0.999981756 | 0.999981756 |
| | Time to reach Solution mm:ss | 01:09 | 00:05 | 00:12 | 03:00 | 00:01 |
| 7<br>2 Layers<br>30 x 30<br>Neurons | simulated data | 0.999940514 | 0.999997308 | 0.999999045 | 0.99999981 | 1 |
| | verify data | 0.999676993 | 0.999799644 | 0.999798234 | 0.999974154 | 0.999974154 |
| | Time to reach Solution mm:ss | 01:12 | 00:04 | 00:11 | 03:06 | 00:01 |
| 8<br>2 Layers<br>40 x 40<br>Neurons | simulated data | 0.99992327 | 0.999992737 | 0.999999439 | 0.999999841 | 1 |
| | verify data | 0.999288164 | 0.999707704 | 0.999961055 | 0.999956722 | 0.999987812 |
| | Time to reach Solution mm:ss | 01:14 | 00:04 | 00:13 | 03:12 | 00:04 |
| 9<br>3 Layers<br>10 x 10 x 10<br>Neurons | simulated data | 0.999882979 | 0.999966795 | 0.999992395 | 0.999997898 | 0.999999997 |
| | verify data | 0.999910773 | 0.999975854 | 0.99998874 | 0.999954502 | 0.99999969 |
| | Time to reach Solution mm:ss | 01:13 | 00:03 | 00:18 | 03:11 | 00:03 |
| 10<br>3 Layers<br>20 x 20 x 20<br>Neurons | simulated data | 0.999871778 | 0.999965098 | 0.999997622 | 0.999997538 | 0.999999998 |
| | verify data | 0.999919554 | 0.999861734 | 0.999978875 | 0.999933998 | 0.999980226 |
| | Time to reach Solution mm:ss | 01:17 | 00:03 | 00:13 | 03:19 | 00:03 |
| 11<br>3 Layers<br>30 x 30 x 30<br>Neurons | simulated data | 0.999937628 | 0.999996323 | 0.999999768 | 0.999999808 | 1 |
| | verify data | 0.999245187 | 0.999541516 | 0.999665832 | 0.999474113 | 0.999665832 |
| | Time to reach Solution mm:ss | 01:35 | 00:04 | 00:21 | 03:50 | 00:01 |
| 12<br>3 Layers<br>40 x 40 x 40<br>Neurons | simulated data | 0.999973088 | 0.999996136 | 0.999999415 | 0.999999903 | 1 |
| | verify data | 0.99982772 | 0.999764675 | 0.999670247 | 0.999983231 | 0.999983231 |
| | Time to reach Solution mm:ss | 01:38 | 00:04 | 00:14 | 03:57 | 00:02 |

## Table A–3 Correlation Ratio and time to obtain solution for each learning Algorithm case 2

| Size | | GDX | LM | BFG | RP | CGP | CGF |
|---|---|---|---|---|---|---|---|
| 1<br>1 Layer<br>10<br>Neurons | simulated data | 0.997421716 | 0.999969763 | 0.999014114 | 0.999213905 | 0.99771077 | 0.998553091 |
| | verify data | 0.995884694 | 0.999893625 | 0.999367354 | 0.999172877 | 0.998582663 | 0.998684718 |
| | Time to reach Solution mm:ss | 00:38 | 01:17 | 00:04 | 00:46 | 00:04 | 00:09 |
| 2<br>1 Layer<br>20<br>Neurons | simulated data | 0.995336219 | 0.999982973 | 0.999852751 | 0.999502653 | 0.999848426 | 0.998388063 |
| | verify data | 0.924061144 | 0.999998649 | 0.991147479 | 0.969938723 | 0.999954383 | 0.993569279 |
| | Time to reach Solution mm:ss | 00:43 | 01:33 | 00:25 | 00:51 | 00:50 | 00:07 |
| 3<br>1 Layer<br>30<br>Neurons | simulated data | 0.996806751 | 0.999985597 | 0.999903201 | 0.998806597 | 0.998807389 | 0.999438118 |
| | verify data | 0.992795757 | 0.975812633 | 0.99986166 | 0.964759798 | 0.966495175 | 0.997367799 |
| | Time to reach Solution mm:ss | 00:51 | 01:59 | 00:21 | 00:57 | 00:18 | 00:14 |
| 4<br>1 Layer<br>40<br>Neurons | simulated data | 0.997892925 | 0.9999861 | 0.999905937 | 0.998788153 | 0.99945839 | 0.999224168 |
| | verify data | 0.986097452 | 0.143796811 | 0.867275277 | 0.737279256 | 0.980675097 | 0.998992426 |
| | Time to reach Solution mm:ss | 00:50 | 02:21 | 00:46 | 00:59 | 00:27 | 00:26 |
| 5<br>2 Layers<br>10 x 10<br>Neurons | simulated data | 0.996890395 | 0.999997582 | 0.999852129 | 0.999569505 | 0.9986369 | 0.999854326 |
| | verify data | 0.912433348 | 0.999404815 | 0.512562258 | 0.910469786 | 0.999706186 | 0.99876493 |
| | Time to reach Solution mm:ss | 00:58 | 02:32 | 01:12 | 01:02 | 00:10 | 00:24 |
| 6<br>2 Layers<br>20 x 20<br>Neurons | simulated data | 0.996142221 | 0.99999999 | 0.999884894 | 0.999850083 | 0.99973115 | 0.999603252 |
| | verify data | 0.909444391 | 0.936508981 | 0.994932753 | 0.942481183 | 0.999825778 | 0.946697531 |
| | Time to reach Solution mm:ss | 00:59 | 10:20 | 04:16 | 01:12 | 00:15 | 00:30 |
| 7<br>2 Layers<br>30 x 30<br>Neurons | simulated data | 0.996598329 | 0.99999999 | 0.999942195 | 0.999960039 | 0.999768294 | 0.999905451 |
| | verify data | 0.636405873 | 0.67423607 | 0.973376501 | 0.967667176 | 0.998951902 | 0.999808203 |
| | Time to reach Solution mm:ss | 01:08 | 12:50 | 28:50 | 01:17 | 00:29 | 01:08 |
| 8<br>2 Layers<br>40 x 40<br>Neurons | simulated data | 0.997748529 | 0.99999901 | 0.998822264 | 0.999960711 | 0.999767041 | 0.999879413 |
| | verify data | 0.630516212 | 0.667252486 | 0.851514957 | 0.956629571 | 0.833867299 | 0.822350015 |
| | Time to reach Solution mm:ss | 01:18 | 26:05 | 56:25 | 01:27 | 00:53 | 01:05 |
| 9<br>3 Layers<br>10 x 10 x 10<br>Neurons | simulated data | 0.997350202 | 0.999999773 | 0.999888087 | 0.999929652 | 0.999682006 | 0.999907425 |
| | verify data | 0.992936228 | 0.999353912 | 0.998986038 | 0.989582256 | 0.998938213 | 0.999864811 |
| | Time to reach Solution mm:ss | 00:55 | 04:17 | 01:57 | 01:10 | 00:27 | 00:36 |
| 10<br>3 Layers<br>20 x 20 x 20<br>Neurons | simulated data | 0.996257627 | 0.99999999 | 0.999940914 | 0.999945295 | 0.999762082 | 0.999829242 |
| | verify data | 0.978996204 | 0.685547129 | 0.814073295 | 0.732601706 | 0.862005909 | 0.994203739 |
| | Time to reach Solution mm:ss | 01:29 | 23:27 | 19:18 | 01:23 | 00:26 | 00:56 |
| 11<br>3 Layers<br>30 x 30 x 30<br>Neurons | simulated data | 0.997686728 | 0.99999999 | 0.999585112 | 0.999972564 | 0.999949771 | 0.999943743 |
| | verify data | 0.947105816 | 0.970780874 | 0.977114937 | 0.893065588 | 0.986427798 | 0.998443019 |
| | Time to reach Solution mm:ss | 01:37 | 32:07 | 01:32:04 | 01:53 | 01:07 | 01:20 |
| 12<br>3 Layers<br>40 x 40 x 40<br>Neurons | simulated data | 0.995687483 | 0.999999966 | 0.997994709 | 0.99998398 | 0.999916312 | 0.999956099 |
| | verify data | 0.445215417 | 0.939654085 | 0.984840882 | 0.988761096 | 0.994385995 | 0.881900269 |
| | Time to reach Solution mm:ss | 02:00 | 01:47:39 | 03:55:40 | 01:49 | 00:59 | 02:53 |

**Table A–3 Continued**

| Size | | GDA | CGB | SCG | OSS | Max. Value Min. Time |
|---|---|---|---|---|---|---|
| **1 1 Layer 10 Neurons** | simulated data | 0.993069265 | 0.998929287 | 0.998957846 | 0.998689024 | 0.999969763 |
| | verify data | 0.995344777 | 0.999273071 | 0.994336215 | 0.999396451 | 0.999893625 |
| | Time to reach Solution mm:ss | 00:39 | 00:10 | 00:22 | 01:32 | 00:04 |
| **2 1 Layer 20 Neurons** | simulated data | 0.992847871 | 0.998185191 | 0.999893027 | 0.999363396 | 0.999982973 |
| | verify data | 0.796550721 | 0.998664137 | 0.999726489 | 0.999642399 | 0.999998649 |
| | Time to reach Solution mm:ss | 00:44 | 00:06 | 00:27 | 01:42 | 00:06 |
| **3 1 Layer 30 Neurons** | simulated data | 0.957142332 | 0.999768934 | 0.999880579 | 0.999429077 | 0.999985597 |
| | verify data | 0.617977696 | 0.996636178 | 0.98631191 | 0.99255058 | 0.99986166 |
| | Time to reach Solution mm:ss | 00:51 | 00:29 | 00:36 | 01:54 | 00:14 |
| **4 1 Layer 40 Neurons** | simulated data | 0.975834402 | 0.999865384 | 0.999918645 | 0.99936063 | 0.9999861 |
| | verify data | 0.920799101 | 0.968838149 | 0.993225527 | 0.963971922 | 0.998992426 |
| | Time to reach Solution mm:ss | 00:51 | 00:37 | 00:52 | 02:03 | 00:26 |
| **5 1 Layer 10 x 10 Neurons** | simulated data | 0.968764374 | 0.999928637 | 0.999749657 | 0.999307618 | 0.999997582 |
| | verify data | 0.776141122 | 0.999904893 | 0.99879928 | 0.821808487 | 0.999904893 |
| | Time to reach Solution mm:ss | 00:51 | 00:22 | 00:29 | 02:04 | 00:10 |
| **6 2 Layers 20 x 20 Neurons** | simulated data | 0.992261753 | 0.99980772 | 0.999936415 | 0.999589259 | 0.99999999 |
| | verify data | 0.974546767 | 0.994357189 | 0.993112716 | 0.893312454 | 0.999825778 |
| | Time to reach Solution mm:ss | 01:02 | 00:16 | 00:47 | 02:25 | 00:15 |
| **7 2 Layers 30 x 30 Neurons** | simulated data | 0.992262839 | 0.999921395 | 0.999954975 | 0.999464822 | 0.99999999 |
| | verify data | 0.976899059 | 0.912104144 | 0.953475843 | 0.961125015 | 0.999808203 |
| | Time to reach Solution mm:ss | 01:07 | 00:27 | 00:35 | 02:46 | 00:27 |
| **8 2 Layers 40 x 40 Neurons** | simulated data | 0.986202936 | 0.999896374 | 0.999973346 | 0.999797966 | 0.99999901 |
| | verify data | 0.935001304 | 0.875996341 | 0.756037951 | 0.332080011 | 0.956629571 |
| | Time to reach Solution mm:ss | 01:17 | 00:31 | 01:53 | 03:10 | 00:31 |
| **9 3 Layers 10 x 10 x 10 Neurons** | simulated data | 0.994952231 | 0.997526044 | 0.99994114 | 0.999439923 | 0.999999773 |
| | verify data | 0.986227565 | 0.996553965 | 0.994876201 | 0.998729027 | 0.999864811 |
| | Time to reach Solution mm:ss | 00:58 | 00:03 | 00:28 | 02:17 | 00:03 |
| **10 3 Layers 20 x 20 x 20 Neurons** | simulated data | 0.988980895 | 0.999873191 | 0.999965168 | 0.999747848 | 0.99999999 |
| | verify data | 0.792296992 | 0.999569785 | 0.979908348 | 0.837947791 | 0.999569785 |
| | Time to reach Solution mm:ss | 01:13 | 00:28 | 01:21 | 02:55 | 00:26 |
| **11 3 Layers 30 x 30 x 30 Neurons** | simulated data | 0.992739047 | 0.999899804 | 0.99997531 | 0.999843149 | 0.99999999 |
| | verify data | 0.974915566 | 0.861619169 | 0.833139766 | 0.952912463 | 0.998443019 |
| | Time to reach Solution mm:ss | 01:27 | 00:58 | 01:41 | 03:37 | 00:58 |
| **12 3 Layers 40 x 40 x 40 Neurons** | simulated data | 0.984708694 | 0.99980022 | 0.999977051 | 0.999820369 | 0.999999966 |
| | verify data | 0.880740521 | 0.896155879 | 0.659881142 | 0.908629323 | 0.994385995 |
| | Time to reach Solution mm:ss | 01:40 | 00:33 | 02:07 | 04:09 | 00:33 |