Proceedings of Science and Mathematics
Faculty of Science, Universiti Teknologi Malaysia
Vol. 4, 2021, page 43-57

Graph Neural Network: A Mathematical Overview And Its Application For Simulation Of Dynamic Viscosity Of Nanofluids

[1]Mohamad Redzuan Firdaus Fozi [2]Tahir Ahmad

[1,2]Department of Mathematical Sciences
Faculty of Science, Universiti Teknologi Malaysia,
81310 Johor Bahru, Johor, Malaysia.

e-mail: [1]redzuan1998@graduate.utm.my [2]tahir@utm.my

Abstract Graph Neural Network (GNN) is a type of Artificial Neural Network that is used to predict the performance of a system. Two types of GNN are used in this research, namely, GraphSAGE, and Artificial Neural Network-Multilayer Perceptron (ANN-MLP). The ANN-MLP with Levenberg-Marquardt training algorithm is used to calculate the flow of nanofluids in porous media. The data used to train the model are obtained from literature. The hidden neurons are chosen for the neural network are carefully selected by analyzing the value of R^2. The GNN can be used to simulate the dynamic viscosity of nanofluid. The simulation offers valuable of information on characteristics of the nanofluid.

Keywords Neural Network; Graph Neural Network; Artificial Neural Network; nanofluids; dynamic viscosity; Levernberg-Marquardt

1 Introduction

Zhou et. al [1] states that a graph is a kind of data structure that can model a set of data into vertices(nodes), and their relationships as edges. The graph has played a huge role in machine learning, and artificial intelligence due to the rise of Industrial Revolution 4 (IR4.0). Researchers who incorporate graph with machine learning receive many attentions as the graph has shown great potential, as it can be used in dealing with a large amount of data or systems such as data related to social sciences (social networks), and natural sciences (physics, chemistry, and biology). On the other hand, Neural Network is used mainly for forecasting. It mimics neurons in the human brain [6] whereby, input and output data are correlated for forecasting purposes.Graph Neural Network(GNN) is a combination of graph and neural network. Graph transforms datasets into nodes (vertices) and their relationships as edges [1]. Concerning simulation for dynamic viscosity of nanofluids, there have many types of neural networks that have been used such as multilayer-perceptron. However, only a few researchers have employed GNN.

Graph Neural Network is one of the computing techniques used in many areas. Irfan et. al [2] states that the dynamic viscosity of nanofluids is important in oil and gases particularly in oil recovery. Ahmadi et. al [3] reported that one of the techniques used in oil recovery is artificial intelligence, and it has been claimed to be effective in modeling the dynamic viscosity of

nanofluids. Ramezanizadeh et. al [4] and Ahmadi et.al [5] claimed that it can be integrated with multilayer-perceptron, regression forest, and others.

The use of multilayer-perceptron, random forest have been reported in the literature. However, almost no literature available on the application of GNN on the dynamic viscosity of nanofluids. This research will investigate the possibility of using GNN for nanofluid-related problems.

This research involves the study of mathematical structure, namely, graph neural network. This includes coding the graph neural network model to simulate the dynamic viscosity of nanofluids. Published data will be used to verify the model.

Mathematical research in the area of artificial intelligence and machine learning has generated many discoveries for future use in many fields such as in oil and gas-related industries. In this research, Graph Neural Network (GNN) is developed for such purpose. The new model promises great potential and its possibility is endless.

2 Literature Review

2.1 Some Concepts of Graph Neural Network

Graph models a set of objects as nodes and their relationship as edges [1]. The history of graph theory begins with the published paper by Leonhard Euler on the famous Seven Bridges of Konigsberg in 1736. There are two types of graphs; directed and undirected graphs. The former contains symmetrically edges between two vertices [7], and the latter is symmetrically [8]. A graph G is a pair (N, E) where N is a set of nodes and E is a set of edges. Every node has at least an edge [9].

Neural Network is a computing technique that was inspired by neuron structure and function in the brain [6] whereby the nodes are the dendrites and the output is the axon terminal such that its signal is transferred by myelinated axons [6]. Zell [10] states, in artificial neural networks, the inputs or signals are real numbers and the connections are transfer functions. Between two nodes, there exists a *weight* that adjusts accordingly when needed during the training process. The weight is essential in the artificial neural network, as it will determine the most likely value or weight during simulation. There are many methods to determine the weight, namely, evaluating mean squared error, evaluating correlation coefficients, and others [4].

Graph Neural Network (GNN) is a combination of graph and neural network. GNN has been used in many areas such as social, and natural sciences [1]. It is mainly used for forecasting. Its main features are nodes as the inputs and outputs, and the relationship between the nodes is linked together with their weight and transfer functions.

2.2 Model of GNN: GraphSAGE

For the simulation process, one must know the frameworks of the neural network. Hamilton et. al [12] introduced GraphSAGE's framework. It is very useful for a graph that has many attributes that represent nodes to generate a low-dimensional vector of them. GraphSAGE generates features from the node's local neighborhood [1]. This is very useful for unseen nodes since they can be represented by their local neighborhoods.

2.3 Model of GNN: Multi-Layer Perceptron

Another type of GNN framework is called multi-layer perceptron. For this research, multilayer perceptron is chosen as the desired framework as it is easy to use with limited resources. As compared to GraphSAGE a high-end processor was needed to run the program. Where MLP can be run using a simple MATLAB program and low-end hardware.

MLP is a class of feedforward neural networks, where it is composed of several layers of input, hidden layer, and output. The size of the hidden layer is usually determined by trial and error. It works similar to GraphSAGE where a feature is determined by its previous feature as a result of aggregation.

2.4 Previous Research on Nanofluids Dynamic Viscosity Prediction

In oil recovery, there are three major processes; 1) primary oil recovery, 2) secondary oil recovery, and 3) enhanced oil recovery (EOR) [2]. Many pieces of research have indicated that nanofluids or nanoparticles could greatly help in EOR [2]. Nanofluids offer several physicochemical properties such as thermal conductivity, dynamic viscosity, and others [13]. However, this research investigates the dynamic viscosity of the nanofluids since it influences the flow of the nanofluids as it moves through porous media [13].

Many variables affect the dynamic viscosity, namely the temperature of the nanofluids, the size of the nanoparticles, and the volume fraction of the nanofluids. Many types of research have dealt with these three variables [3, 4, 5]. The respective researchers mainly used multilayer-perceptron as their desired Neural Network. Ahmadi et. al [3, 4] and Ramezanizadeh et. al [5] claimed that this Neural Network can simulate the dynamic viscosity with fairly good results.

3 Frameworks of Neural Network Model

3.1 Frameworks of GraphSAGE

GraphSAGE is used to present every node based on its neighboring nodes that are parametrized by h.

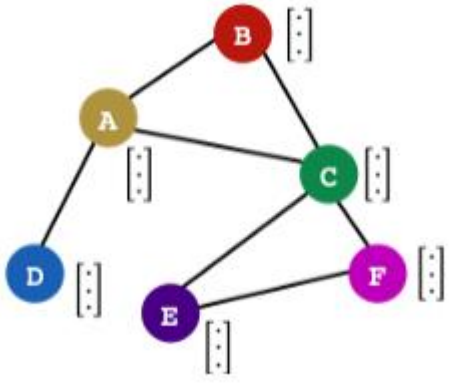

Figure 1: Graph representation

Recall, every node has its feature vector parametrized by X. Every node is assumed to have the same size. GraphSAGE runs with k iterations, which means that one iteration is for one node. Every node is represented by h for k iteration.

Notice the following notation:

$X_v = Node\ features\ for\ a\ random\ node\ v$

$h_v^0 = Initial\ node\ embedding\ representation\ for\ a\ random\ node\ v$

$h_v^k = Node\ embedding\ representation\ for\ a\ random\ node\ v\ at\ the\ k-th\ iteration$

$z_v = Final\ node\ embedding\ representation\ for\ a\ random\ node\ v$

Every node has its neighbors, and these neighbors define the targeted node. By the combination of its neighboring nodes embedding vector, then node A is defined. A similar process is executed for other nodes in the graph.

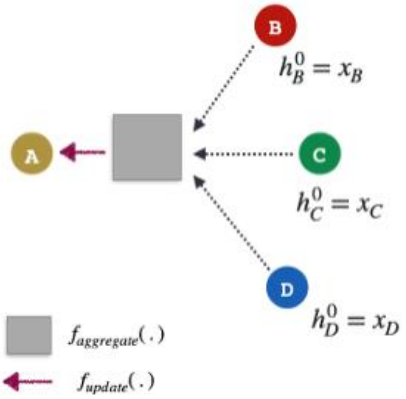

Figure 2:Process to find target node, A

GraphSAGE consists of sampling and aggregation with a two-step process. It is an iterative procedure with initialization steps that set the initials' node embedding vectors to their feature vectors. (i.e. k start from 1...K) such that,

$$h_v^{k-1} = h_v^0 = x_v$$

Figure 2 illustrates the two steps in the algorithm. Based on the figure, the steps are aggregated and updated.

Aggregate means that the algorithm aggregates neighboring nodes for the targeted node. The aggregate function denotes as $f_{aggregate}$. This function comes in many forms such as mean aggregator, and pool aggregator.

$$a_v = f_{aggregate}(\{h_u | \ u \in N(v)\})$$

The targeted node, v depends on the aggregation of all nodes u, and the resultant is denoted as a_v. The current node v is then be updated using a combination of both aggregated node and its previous representations. It will be updated using the aggregated node plus the previous representation of the current node.

$$h_v^k = f_{update}(a_v, h_v^{k-1})$$

The above equation updates node v based on its neighborhood aggregated representations and node v's previous representation.

Figure 3: Target node is node A

Figure 3 shows, node A is selected as the targeted node. The following Figure 4 illustrates how node A (target node) is constructed.

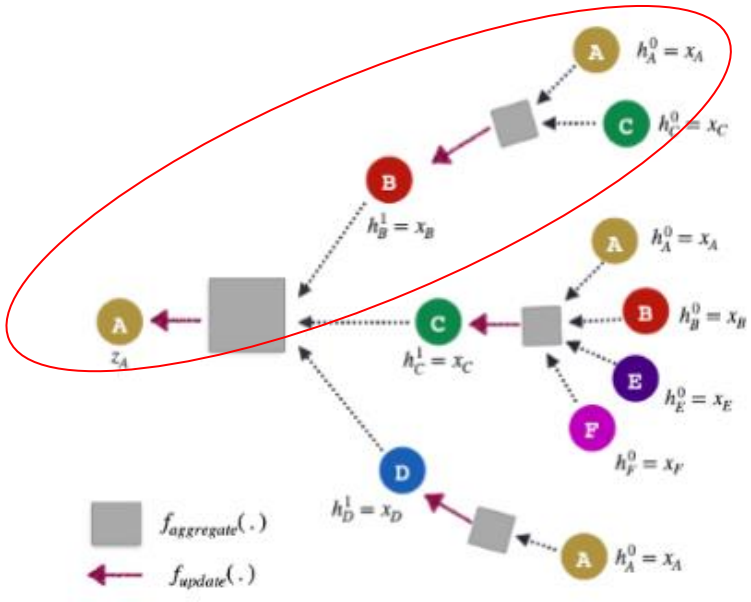

Figure 4:Process in obtaining targeted node

The process starts from the outer to the inner layers. In Figure 4 Node B is the neighbor of node A (target node). The algorithm starts by aggregating from nodes B, C, and D. However, node B, has its neighbors which are nodes A and C. Therefore, the algorithm aggregates these nodes first. This aggregation process is determined by the number of parameters K. For example, K=2 implies the aggregation starts from node A neighbor and the neighbor of its neighbor. (i.e. A has neighbor B, and B has neighbor C and A).

3.1 Formulation of ANN-MLP

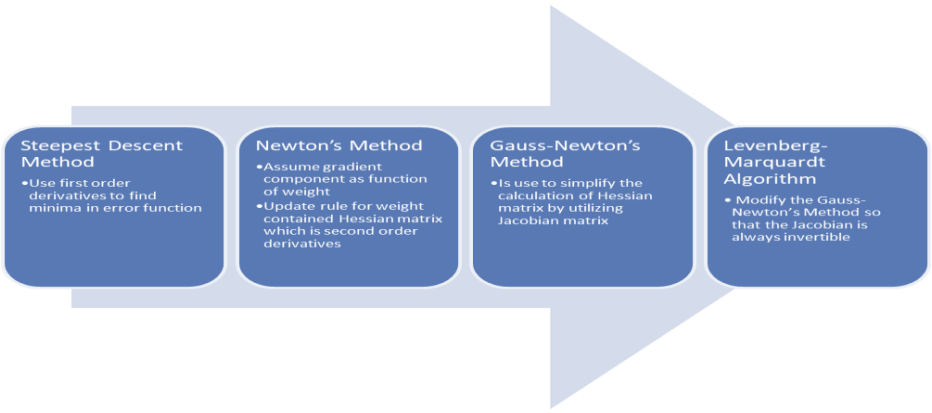

Figure 5: Steps of derivation for the Levenberg-Marquardt algorithm

The Levenberg-Marquardt algorithm composes of four major parts (1) the steepest descent method, (2) Newton's method, (3) Gauss-Newton's method, and (4) Levenberg-Marquardt algorithm. All these parts are interconnected with each other.

Sum square error (SSE) is used to evaluate error during the training process. It is calculated as follows.

$$E(x,w) = \frac{1}{2}\sum_1^p \sum_1^m e_{p,m}^2 \qquad (1)$$

where

x is the input vector,

w is weight vector,

$e_{p,m}$ is the training error at output m and is defined as

$$e_{p,m} = d_{p,m} - o_{p,m} \qquad (2)$$

such that

d is the desired output vector

o is the actual output vector

3.1.1 Steepest Descent Method

The steepest descent algorithm uses the first-order derivation. To find minima in error space, the derivation on E with respect to w is determined. It is denoted as g.

$$g = \frac{\delta E(x,w)}{\delta w} = \left[\frac{\delta E}{\delta w_1} \quad \frac{\delta E}{\delta w_2} \quad \frac{\delta E}{\delta w_3} \ldots \frac{\delta E}{\delta W_N}\right] \qquad (3)$$

The weight, w is defined as

$$w_{k+1} = w_k - \alpha g_k \qquad (4)$$

where

α is a learning constant (step size)

3.1.2 Newton's Method

Newton's method assumes that gradient components, g, as a function of weight and each of them is a linearly independent

$$\begin{cases} g_1 = F_1(w_1, w_2, w_3, \dots, w_N \\ g_2 = F_2(w_1, w_2, w_3, \dots, w_N \\ \vdots \\ g_N = F_N(w_1, w_2, w_3, \dots, w_N \end{cases} \quad (5)$$

and by employing the Taylor's series

$$\begin{cases} g_1 = g_{1,0} + \frac{\delta g_1}{\delta w_1}\Delta w_1 + \frac{\delta g_1}{\delta w_2}\Delta w_2 + \cdots + \frac{\delta g_1}{\delta w_N}\Delta w_N \\ g_2 = g_{2,0} + \frac{\delta g_2}{\delta w_1}\Delta w_1 + \frac{\delta g_2}{\delta w_2}\Delta w_2 + \cdots + \frac{\delta g_2}{\delta w_N}\Delta w_N \\ \vdots \\ g_N = g_{N,0} + \frac{\delta g_N}{\delta w_1}\Delta w_1 + \frac{\delta g_N}{\delta w_2}\Delta w_2 + \cdots + \frac{\delta g_N}{\delta w_N}\Delta w_N \end{cases} \quad (6)$$

Hence,

$$\frac{\delta g_i}{\delta w_j} = \frac{\delta(\frac{\delta E}{\delta w_i})}{\delta w_j} = \frac{\delta^2 E}{\delta w_i \delta w_j} \quad (7)$$

Combining Eq. 7 with Eq. 6, it

$$\begin{cases} g_1 = g_{1,0} + \frac{\delta^2 E}{\delta w_1^2}\Delta w_1 + \frac{\delta^2 E}{\delta w_1 \delta w_2}\Delta w_2 + \cdots + \frac{\delta^2 E}{\delta w_1 \delta w_N}\Delta w_N \\ g_2 = g_{2,0} + \frac{\delta^2 E}{\delta w_2 \delta w_1}\Delta w_1 + \frac{\delta^2 E}{\delta w_2^2}\Delta w_2 + \cdots + \frac{\delta^2 E}{\delta w_2 \delta w_N}\Delta w_N \\ \vdots \\ g_N = g_{N,0} + \frac{\delta^2 E}{\delta w_n \delta w_1}\Delta w_1 + \frac{\delta^2 E}{\delta w_N \delta w_2}\Delta w_2 + \cdots + \frac{\delta^2 E}{\delta w_N^2}\Delta w_N \end{cases} \quad (8)$$

To find minima of the total error function, E, each element of gradient vector, g, must be zero. Hence, the left-hand sides of Eq. 8 are zeroes.

$$\begin{cases} 0 = g_{1,0} + \frac{\delta^2 E}{\delta w_1^2}\Delta w_1 + \frac{\delta^2 E}{\delta w_1 \delta w_2}\Delta w_2 + \cdots + \frac{\delta^2 E}{\delta w_1 \delta w_N}\Delta w_N \\ 0 = g_{2,0} + \frac{\delta^2 E}{\delta w_2 \delta w_1}\Delta w_1 + \frac{\delta^2 E}{\delta w_2^2}\Delta w_2 + \cdots + \frac{\delta^2 E}{\delta w_2 \delta w_N}\Delta w_N \\ \vdots \\ 0 = g_{N,0} + \frac{\delta^2 E}{\delta w_n \delta w_1}\Delta w_1 + \frac{\delta^2 E}{\delta w_N \delta w_2}\Delta w_2 + \cdots + \frac{\delta^2 E}{\delta w_N^2}\Delta w_N \end{cases} \quad (9)$$

Combining Eq. 3, with Eq. 9, the equation become

$$\begin{cases} -\frac{\delta E}{\delta w_1} = -g_1 = g_{1,0} + \frac{\delta^2 E}{\delta w_1^2}\Delta w_1 + \frac{\delta^2 E}{\delta w_1 \delta w_2}\Delta w_2 + \cdots + \frac{\delta^2 E}{\delta w_1 \delta w_N}\Delta w_N \\ -\frac{\delta E}{\delta w_2} = -g_2 = g_{2,0} + \frac{\delta^2 E}{\delta w_2 \delta w_1}\Delta w_1 + \frac{\delta^2 E}{\delta w_2^2}\Delta w_2 + \cdots + \frac{\delta^2 E}{\delta w_2 \delta w_N}\Delta w_N \\ \vdots \\ -\frac{\delta E}{\delta w_N} = -g_N = g_{N,0} + \frac{\delta^2 E}{\delta w_n \delta w_1}\Delta w_1 + \frac{\delta^2 E}{\delta w_N \delta w_2}\Delta w_2 + \cdots + \frac{\delta^2 E}{\delta w_N^2}\Delta w_N \end{cases} \quad (10)$$

Eq. 10, could be written as matrix form

$$\begin{bmatrix} -g_1 \\ -g_2 \\ \vdots \\ -g_N \end{bmatrix} = \begin{bmatrix} -\dfrac{\delta E}{\delta w_1} \\ -\dfrac{\delta E}{\delta w_2} \\ \vdots \\ -\dfrac{\delta E}{\delta w_N} \end{bmatrix} = \begin{bmatrix} \dfrac{\delta^2 E}{\delta w_1^2} & \dfrac{\delta^2 E}{\delta w_1 \delta w_2} & \cdots & \dfrac{\delta^2 E}{\delta w_1 \delta w_N} \\ \dfrac{\delta^2 E}{\delta w_2 \delta w_1} & \dfrac{\delta^2 E}{\delta w_2^2} & \cdots & \dfrac{\delta^2 E}{\delta w_2 \delta w_N} \\ \cdots & \cdots & \ddots & \cdots \\ \dfrac{\delta^2 E}{\delta w_N \delta w_1} & \dfrac{\delta^2 E}{\delta w_N \delta w_2} & \cdots & \dfrac{\delta^2 E}{\delta w_N^2} \end{bmatrix} \times \begin{bmatrix} \Delta w_1 \\ \Delta w_2 \\ \vdots \\ \Delta w_N \end{bmatrix} \quad (11)$$

where the square matrix is called Hessian matrix, H

$$H = \begin{bmatrix} \dfrac{\delta^2 E}{\delta w_1^2} & \dfrac{\delta^2 E}{\delta w_1 \delta w_2} & \cdots & \dfrac{\delta^2 E}{\delta w_1 \delta w_N} \\ \dfrac{\delta^2 E}{\delta w_2 \delta w_1} & \dfrac{\delta^2 E}{\delta w_2^2} & \cdots & \dfrac{\delta^2 E}{\delta w_2 \delta w_N} \\ \cdots & \cdots & \ddots & \cdots \\ \dfrac{\delta^2 E}{\delta w_N \delta w_1} & \dfrac{\delta^2 E}{\delta w_N \delta w_2} & \cdots & \dfrac{\delta^2 E}{\delta w_N^2} \end{bmatrix} \quad (12)$$

By combining Eq.3 and Eq.12 with Eq. 11, then

$$-g = H\Delta w \quad (13)$$

and,

$$\Delta w = -H^{-1} g \quad (14)$$

Therefore, the update rule for weight, w, is

$$w_{k+1} = w_k - H_k^{-1} g_k \quad (15)$$

3.1.3 Gauss-Newton's Method

In Newton's method, one can see that calculation for the Hessian matrix is complicated as it deals with second-order derivatives. Jacobian matrix is introduced to solve the problem.

$$J = \begin{bmatrix} \dfrac{\delta e_{1,1}}{\delta w_1} & \dfrac{\delta e_{1,1}}{\delta w_2} & \cdots & \dfrac{\delta e_{1,1}}{\delta w_N} \\ \dfrac{\delta e_{1,2}}{\delta w_1} & \dfrac{\delta e_{1,2}}{\delta w_2} & \cdots & \dfrac{\delta e_{1,2}}{\delta w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \dfrac{\delta e_{1,M}}{\delta w_1} & \dfrac{\delta e_{1,M}}{\delta w_2} & \cdots & \dfrac{\delta e_{1,M}}{\delta w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \dfrac{\delta e_{p,1}}{\delta w_1} & \dfrac{\delta e_{p,1}}{\delta w_2} & \cdots & \dfrac{\delta e_{p,1}}{\delta w_N} \\ \dfrac{\delta e_{p,2}}{\delta w_1} & \dfrac{\delta e_{p,2}}{\delta w_2} & \cdots & \dfrac{\delta e_{p,2}}{\delta w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \dfrac{\delta e_{p,m}}{\delta w_1} & \dfrac{\delta e_{p,m}}{\delta w_2} & \cdots & \dfrac{\delta e_{p,m}}{\delta w_N} \end{bmatrix} \quad (16)$$

By integrating Eq.1 and Eq.3, the elements of gradient vector, g, become

$$g_i = \frac{\delta E}{\delta w_i} = \frac{\delta\left[\frac{1}{2}\sum_1^p \sum_1^m e_{p,m}^2\right]}{\delta w_i} = \sum_1^p \sum_1^m \left(\frac{\delta e_{p,m}}{\delta w_i} e_{p,m}\right) \quad (17)$$

Combining Eq.16 and Eq.17, the relationship between J and g is

$$g = Je \quad (18)$$

where
e is an error vector in the form of

$$e = \begin{bmatrix} e_{1,1} \\ e_{1,2} \\ \cdots \\ e_{1,M} \\ \cdots \\ e_{p,1} \\ e_{p,2} \\ \cdots \\ e_{p,m} \end{bmatrix} \qquad (19)$$

Integrating Eq.1 and Eq.11, the Hessian matrix is

$$H_{ij} = \frac{\delta^2 E}{\delta w_i \delta w_j} = \frac{\delta^2 \left[\frac{1}{2}\sum_1^p \sum_1^m e_{p,m}^2\right]}{\delta w_i \delta w_j} = \sum_1^p \sum_1^m \left(\frac{\delta e_{p,m}}{\delta w_i}\frac{\delta e_{p,m}}{\delta w_j} + s_{ij}\right) \qquad (20)$$

The basic assumption of Newton's method is S_{ij} is closed to zero, then the relationship between *H* and *J* is written as

$$H = J^T J \qquad (21)$$

By combining Eq. 15, Eq.19, and Eq.21, the update rule for weight for Gauss-Newton's algorithm is,

$$w_{k+1} = w_k - \left(J_k^T J_k\right)^{-1} J_k e_k \qquad (22)$$

3.1.4 Levenberg-Marquardt algorithm

The problem with the Gauss-Newton algorithm is that $J^T J$ may not always be invertible. To overcome this problem another approximation is

$$H = J^T J + \mu I \qquad (23)$$

such that
μ is combination coefficient (positive)
I is an identity matrix
One can notice that the main diagonal is always positive, thus it will always invertible. So, by combining Eq.23 with Eq.22, update rule for weight, w, for Levenberg-Marquardt algorithm is

$$w_{k+1} = w_k - \left(J_k^T J_k + \mu I\right)^{-1} J_k e_k \qquad (24)$$

4 Results and Discussion

4.1 Dynamics viscosity of nanofluids with respect to volume fraction

After the model is trained, it is then used to determine the dynamic viscosity of nanofluids. It is made to justify whether the model is well trained or not. The result can give insight into dynamicity of viscosity of nanofluids with respect to other parameters such as the influences of the temperature of the nanofluids, the volume of the nanofluids, and the size of the nanoparticles. Moreover, the properties of the nanofluids as it flows through porous media are gathered.

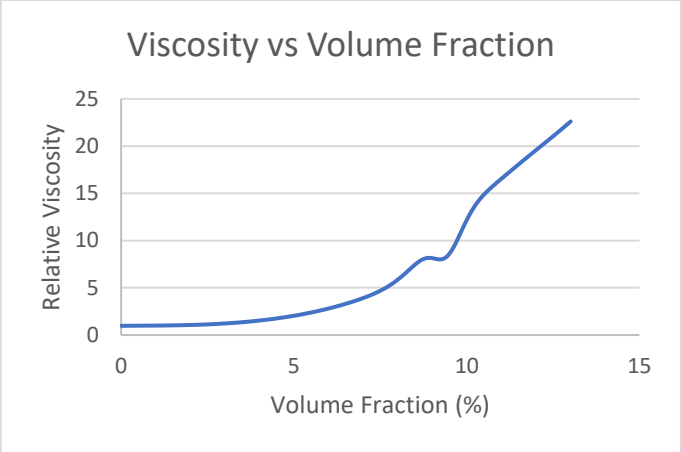

Figure 6: Viscosity vs Volume fraction plots

Figure 14 shows the graph of volume fraction vs viscosity for 0.01%-13%. The curve shows a relatively constant from 0.01% until approximately 5% of volume fraction. This shows that volume fraction in that range has relatively low importance that affects viscosity. The curve steadily increase in viscosity from 5% until 10% volume fraction. This indicates that the volume fraction in that range starts to influence the viscosity of the nanofluids. The curve greatly increase with respect to viscosity from 10% to 13% volume fraction. Thus, the volume fraction in that range affects viscosity. Therefore, the hypothesis made earlier is verified; i.e. increase in volume fraction leads to increasing in viscosity of nanofluid.

4.2 Dynamics viscosity of nanofluids with respect to temperature

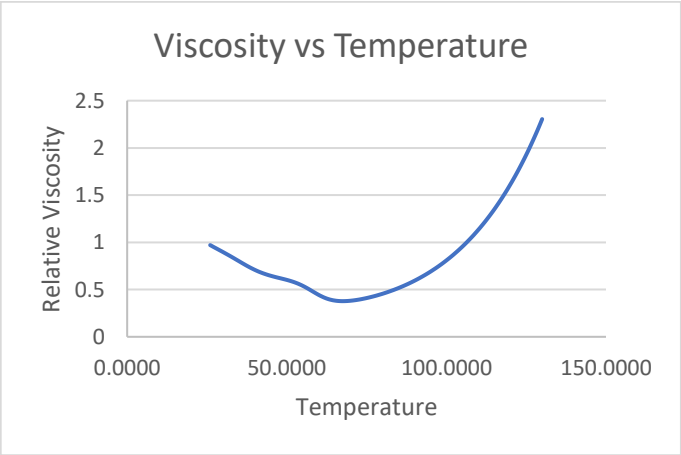

Figure 7: Viscosity vs Temperature plots

Figure 15 shows, the graph of viscosity vs temperature. When the temperature in between 25 to 70 degrees Celsius, the viscosity decreases as the temperature increases and concur to Lee J.H et al.s'[20] claim. When the temperature is between 70 to 150 degrees Celsius, a great increasing in viscosity as the temperature increase is detected. There exists a critical point where the viscosity increases with temperature. This effect is called the hysteresis phenomenon. [23]

4.3 Dynamics viscosity of nanofluids with respect to size of nanoparticles

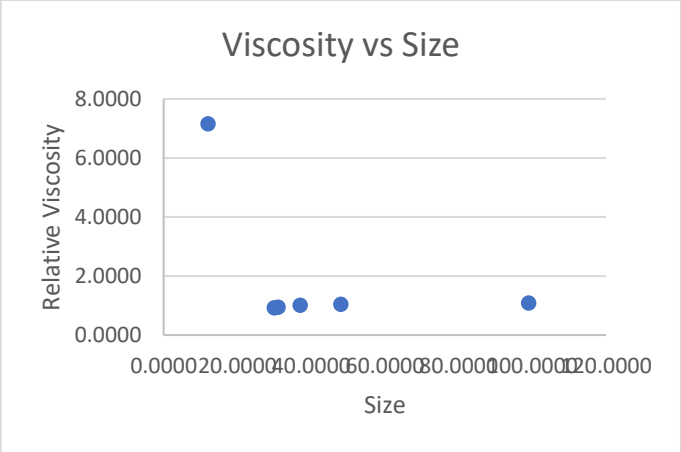

Figure 7: Viscosity vs Size plots

Figure 16 shows the relation between viscosity and size. The size of nanoparticles used are 12,30,31,37,48, and 99nm. Almost no relationship exists between the size of nanoparticles and the viscosity of the nanofluid. This means that size of the particle is almost negligible, as it neither increases nor decreases the viscosity of the nanofluids. These results also confirm the studies made by Pastoriza-Gallego, M. J. et al. [22] who stated that the size of nanoparticle is negligible.

4.2 Comparison between Actual and Predicted Data and their Error Analysis

In this section, a comparison is made between calculated and the actual values. The purpose is to measure the relative error between them. The error analysis is done by finding relative error between the calculated and actual values. The mean relative error is calculated as follows.

$$Relative\ Error\ \% = \frac{\left|data_{predicted} - data_{actual}\right|}{data_{actual}} * 100 \qquad Eq.\ 1$$

$$Mean\ Relative\ Error\ \% = \frac{\sum Relative\ Error\ \%}{Total\ Number\ of\ Data} \qquad Eq.\ 2$$

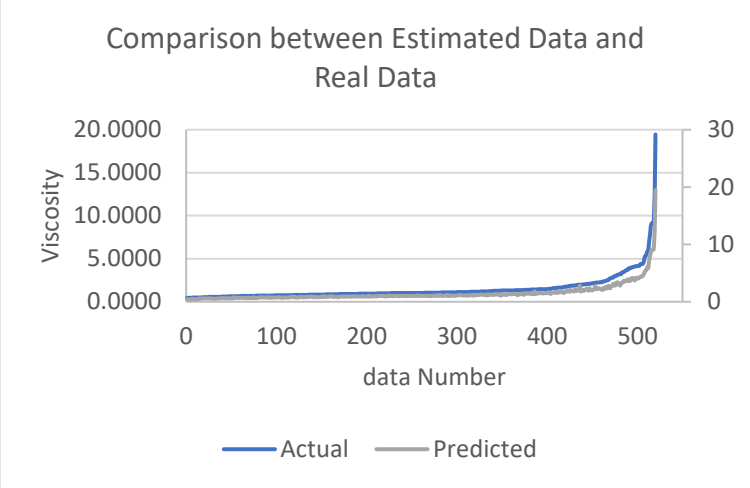

Figure 8: Comparison between calculated data and actual data

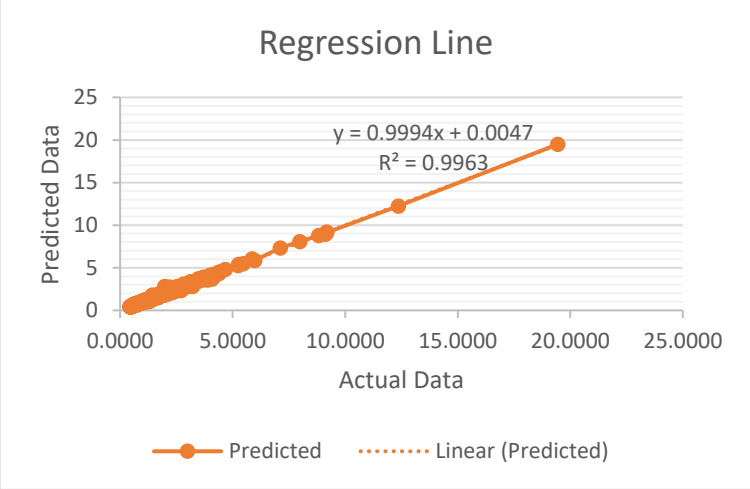

Figure 9: Regression line between actual data and calculated data

Figure 17 shows the comparison between the actual data and calculated data. Both actual data and calculated data lie closely to each other. In other words, the calculated data and the actual data are highly correlated to each other. This means that the model is well trained and can be used to model the dynamic viscosity of nanofluids.

Figure 18 shows the regression line is drawn between the actual and calculated data. The correlation coefficient between the data is 0.9963. The closer the value to 1, the higher is the correlation.

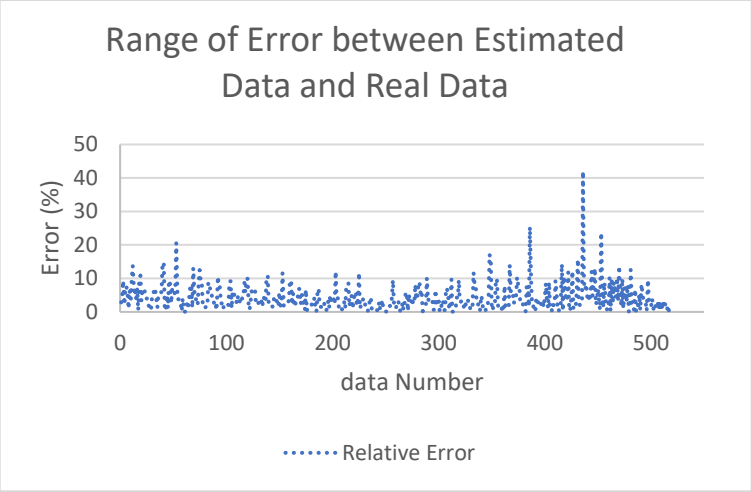

Figure 10: Range of error between actual data and calculated data

Table 1: Mean relative error between the calculated and actual data

	ANN-MLP
Mean Relative Error (%)	4.098634493

Figure 19 is the range of error between actual and calculated data. The range of error is mostly fall under 10% of error while the largest error is approximately 40%. Table 5 shows that the mean relative error for this model is only 4%, which is acceptable as the value of MSE for the training, validation, and testing is 0.0078, 0.1124, and 0.0597 respectively.

4 Conclusion

In this study, ANN-MLP is used as a tool to develop a model to calculate the dynamic viscosity of nanofluids by considering the temperature of nanofluids, volume fraction of nanofluids, and the size of nanofluids as input variables. Several hidden neurons are tested to identify the optimum number of hidden neurons in the network model. Statistical benchmark that was used for determining the hidden neurons is the value of correlation coefficient R^2. With 17 hidden neurons, the highest value of R^2, 0.9983 for training, 0.9847 for validation, and 0.9821 for testing were obtained. These values are all close to 1 as compared to other number of hidden neurons.

Low correlation between data gives the best performance for the neural network. The correlation between data all fall below 0.5, where the lowest correlation between viscosity is size with 0.0570 and the highest correlation between viscosity is volume fraction which is 0.5131. The correlation for temperature with viscosity is -0.2061. Therefore, they are all low correlations that imply good performance for the network.

The size has low importance in determining dynamic viscosity of nanofluid, and volume fraction gives linear relationship with the dynamic viscosity. Similarly for temperature. However, inverse linear relationship is obtained, namely, as temperature increases the dynamic viscosity decrease. The critical temperature where the dynamic viscosity increases as the temperature increase is called the hysteresis phenomenon, that is the temperature is above 70 degrees Celsius. The actual and calculated data are shown to be closely lied. The correlation coefficient between the actual and estimated data is 0.9963. This model is suitable for simulating the dynamic viscosity

of nanofluids. Based on the error analysis, the average relative error is 4.098% hence, the network is well trained and can be used to calculate the viscosity.

In conclusion, the Levenberg-Marquardt training algorithm with 17 hidden neurons is the optimal setup calculating the dynamic viscosity of nanofluids based on 520 sets of data.

5 References

[1] Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., ... & Sun, M. (2018). Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*.

[2] Irfan, S. A., Shafie, A., Yahya, N., & Zainuddin, N. (2019). Mathematical Modeling and Simulation of Nanoparticle-Assisted Enhanced Oil Recovery—A Review. Energies, 12(8), 1575.

[3] Ahmadi, M. H., Mohseni-Gharyehsafa, B., Ghazvini, M., Goodarzi, M., Jilte, R. D., & Kumar, R. (2020). Comparing various machine learning approaches in modeling the dynamic viscosity of CuO/water nanofluid. Journal of Thermal Analysis and Calorimetry, 139(4), 2585-2599.

[4] Ramezanizadeh, M., Ahmadi, M. A., Ahmadi, M. H., & Nazari, M. A. (2019). Rigorous smart model for predicting dynamic viscosity of Al 2 O 3/water nanofluid. Journal of Thermal Analysis and Calorimetry, 137(1), 307-316.

[5] Ahmadi, M. H., Mohseni-Gharyehsafa, B., Farzaneh-Gord, M., Jilte, R. D., Kumar, R., & Chau, K. W. (2019). Applicability of connectionist methods to predict dynamic viscosity of silver/water nanofluid by using ANN-MLP, MARS, and MPR algorithms. Engineering Applications of Computational Fluid Mechanics, 13(1), 220-228.

[6] Chen, Y. Y., Lin, Y. H., Kung, C. C., Chung, M. H., & Yen, I. Hsuan (January 2019)." Design and Implementation of Cloud Analytics-Assisted Smart Power Meters Considering Advanced Artificial Intelligence as Edge Analytics in Demand-Side Management for Smart Homes. *Sensors*, *19*(9), 2047.

[7] Kamada, T., & Kawai, S. (1989). An algorithm for drawing general undirected graphs. *Information processing letters*, *31*(1), 7-15.

[8] Gansner, E. R., Koutsofios, E., North, S. C., & Vo, K. P. (1993). A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, *19*(3), 214-230.

[9] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*(1), 61-80.

[10] Zell, A. (2003). Simulation Neuronaler Netze (Simulation with Neuronal Networks). *Wissenschaftsverlag: Oldenbourg*.

[11] Anand, R. (2020, March 30). An Illustrated Guide to Graph Neural Networks. Retrieved November 23, 2020, from https://medium.com/dair-ai/an-illustrated-guide-to-graph-neural-networks-d5564a551783

[12] Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in neural information processing systems* (pp. 1024-1034).

[13] Goldberg, E., Scheringer, M., Bucheli, T. D., & Hungerbühler, K. (2015). Prediction of nanoparticle transport behavior from physicochemical properties: machine learning provides insights to guide the next generation of transport models. Environmental Science: Nano, 2(4), 352-360.

[14] Savyakhosla (2019, August 26). CNN: Introduction to Pooling Layer. Retrieved January 17, 2021, from https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

[15] Alvim, R. S., Babilonia, O. A., Celaschi, Y. M., & Miranda, C. R. (2017). Nanoscience applied to oil recovery and mitigation: a multiscale computational approach. MRS Advances, 2(9), 477.

[16] Sangani, M. F., Owens, G., Nazari, B., Astaraei, A., Fotovat, A., & Emami, H. (2019). Different modeling approaches for predicting titanium dioxide nanoparticles mobility in intact soil media. Science of The Total Environment, 665, 1168-1181.

[17] Ahmadi, M. H., Sadeghzadeh, M., Maddah, H., Solouk, A., Kumar, R., & Chau, K. W. (2019). Precise smart model for estimating dynamic viscosity of SiO2/ethylene glycol–water nanofluid. Engineering Applications of Computational Fluid Mechanics, 13(1), 1095-1105.

[18] Yu, H., & Wilamowski, B. M. (2011). Levenberg-marquardt training. Industrial electronics handbook, 5(12), 1.

[19] Cook, N. D. (1995). Correlations between input and output units in neural networks. Cognitive Science, 19(4), 563-574.

[20] Lee, J. H., Hwang, K. S., Jang, S. P., Lee, B. H., Kim, J. H., Choi, S. U., & Choi, C. J. (2008). Effective viscosities and thermal conductivities of aqueous nanofluids containing low volume concentrations of Al2O3 nanoparticles. International Journal of Heat and Mass Transfer, 51(11-12), 2651-2656.

[21] Esfe, M. H., Saedodin, S., Wongwises, S., & Toghraie, D. (2015). An experimental study on the effect of diameter on thermal conductivity and dynamic viscosity of Fe/water nanofluids. Journal of Thermal Analysis and Calorimetry, 119(3), 1817-1824.

[22] Pastoriza-Gallego, M. J., Casanova, C., Legido, J. L., & Piñeiro, M. M. (2011). CuO in water nanofluid: influence of particle size and polydispersity on volumetric behavior and viscosity. Fluid phase equilibria, 300(1-2), 188-196.

[23] Nguyen, C. T., Desgranges, F., Roy, G., Galanis, N., Maré, T., Boucher, S., & Mintsa, H. A. (2007). Temperature and particle-size dependent viscosity data for water-based nanofluids–hysteresis phenomenon. International journal of heat and fluid flow, 28(6), 1492-1506.

[24] Heidari, E., Sobati, M. A., & Movahedirad, S. (2016). Accurate prediction of nanofluid viscosity using a multilayer perceptron artificial neural network (MLP-ANN). Chemometrics and intelligent laboratory systems, 155, 73-85.