ISSN 1512-1232

Development of Multilayer Perceptron Models for Predicting Average Void Fraction and PDF of void Fraction in a Vertical 90^o Bend for Air–Silicone Oil Flow

[1] Mukhtar Abdulkadir, [2] Paul Onubi Ayegba

[1]Mukhtar Abdulkadir, Manufacturing Engineering Research Division, Faculty of Engineering, University of Nottingham University Park, Nottingham, NG7 2RD, United Kingdom. Chemical Engineering Department, Federal University of Technology, Minna, Nigeria. Petroleum Engineering Department, African University of Science and Technology (AUST), Abuja, Nigeria, City
[*2]National Environmental Standards and Regulations Enforcement Agency, North-west zonal Headquarters, Kano, Nigeria. Chemical Engineering Department, Ahmadu Bello University, Zaria, Nigeria
Email: paulsnow4christ@gmail.com

Abstract

Multilayer Perceptron (MLP) models have been developed to predict two-phase average void fraction and probability density function (PDF) of void fraction in 90^o bends. The Artificial Neural Network (ANN) methodology was reported using MLP trained with 2 algorithms. Logarithmic sigmoid transfer function was used in a single hidden layer for both algorithms (Gradient descent (GDMV) and Levenberg-Marquardt (LM) algorithms). Both MLP models were optimised by varying the number of neurons in the hidden layer while monitoring the Mean Square Error (MSE). The performance of the models was evaluated using the Average Absolute Relative Error (AARE) and Cross Correlation Coefficient (R). Both MLP models developed for the prediction of average void faction before the bend performed excellently well. However, the MLP model trained with LM algorithm having 3 neurons in the hidden layer gave better performance. Similarly, the MLP model trained with LM algorithm, having 11 neurons in the hidden layer for the prediction of PDF of void fraction before the bend gave excellent prediction. Model performance for the MLP models after the bend gave poor generalisation property. However, the MLP model based on GDMV algorithm gave better prediction for predicting average void fraction and PDF of void fraction after the bend. It was concluded that MLP models may with some confidence be used to predict the average void fraction and the PDFs of void fraction observed before a vertical 90^o bend.

Keywords: *90^o bend, air–silicone oil; void fraction; MLP; ANN; LM algorithm; GDMV algorithm; modelling.*

1. Introduction

Two phase gas-liquid flows are both common and important in oil and gas production and flow lines of different configurations [2]. Two-phase flows are often conveyed through pipes of various configurations as well as pipe fittings such as bends [27]. According to [3] the presence of bends introduces secondary flows, redistribution of multiphase fractions, flow excursions, deviation in flow paths etc. The orientation of the connecting pipe make a difference in the flow patterns for reasons which include sharp deviation in flow path around bends and also because of the role played by gravity and density difference between the two fluids. Two important characteristics of two-phase flows around bends are highly fluctuating voids immediately after bends and changes in flow pattern. It is therefore imperative to develop a means for predicting flow characteristics. A method that has been successfully used for determining flow pattern is the probability density function (PDF) of void fraction. A number of modelling techniques have been employed in the past for void fraction and flow pattern prediction. The models are either empirical, mechanistic of

ISSN 1512-1232

numerical. However, the application of Multilayer Perceptron Modelling (MLP), for the prediction of flow characteristics around bends has received little attention in the peer review literature. Most of the investigations have been restricted to experimental investigation: [2], [11] - [14], [17], and [25] address the issue of gas–liquid systems but most of the reported experiments are not extended to the application of MLP to predict such flows in bends.

MLP modelling is a type of numerical modelling. Numerical modelling technique is a useful tool in visualising the dynamic behaviour of real systems. It has a unique advantage in that there is no limit to the complexity of problem it can solve [26]. MLP models are similar to ANN and are based on the concept of interconnected systems of simple-processing units [10].

1.1. The Back-propagation Algorithm

In back-propagation algorithms the processing elements are arranged in layers, signals are sent forward while errors are propagated backwards. The layers consist of input, hidden and outputs layers. The number of processing elements in the input layer is defined by the number of independent input variables, while that of the hidden layer is determined by method of optimisation. There may be one or more intermediate hidden layers as illustrated in Figure 1.

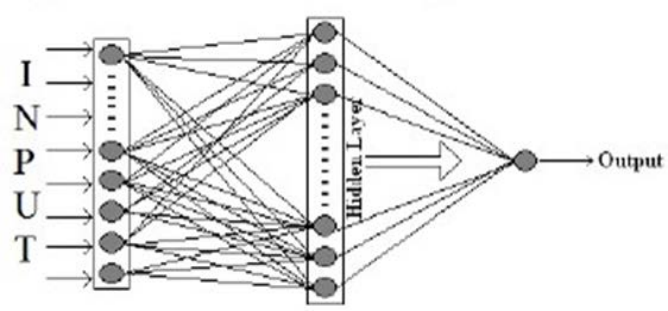


Figure 1: Schematic diagram of the ANN [24].

Generally MLP modelling similar to ANN use either supervised or unsupervised learning during the process of model training. The method used in back-propagation algorithm is supervised learning, which means that the user provides the algorithm with examples of the inputs and outputs for the network to compute, and then the error (difference between actual and expected results) is calculated. The concept of the back-propagation algorithm is to reduce this error, until the ANN learns the training data. The training usually begins with random weights (or user specified weights), and the goal is to adjust them so that the error will be minimal [10].

1.2. Two-phase flow in 90^o bends

A number of experimental and modelling studies have been carried out in the past for two phase gas-liquid flows in bends. Majority of these works were for air-water systems and very few used advanced instrumentation for their investigation. A Visual and experimental study for two-phase flow through a transparent pipe 76-mm diameter was carried out by [14]. Their flow facility consisted of a vertical 90° bends of 305 and 610 mm radii of curvature. The investigation for limited to bubble and slug flow regimes for air-water flow. The goal was to interrogate the competitive effects of centrifugal and gravity forces on the ensuing flow distribution in bends. A 2D modelling study was carried out by [11] for flow around a vertical 90° bends. Model predictions from their work were compared results of experimental investigation of [14]. The results showed poor correlation between model predictions and experimental measurements. An extension of the 2D model to a 3D model was carried out by [12] who obtained better model predictions for experimental measurements. [13] developed an improved 3-D numerical simulation, in which a substantially different solution algorithm, with an additional momentum term, was adopted. They applied the models to both air–water and gas–oil mixtures. The predicted simulation flow data gave better agreement with the experimental data compared to those obtained by the 2-D model in [11]. There was, however, a shortage of experimental data at the time for validation of their gas-oil

ISSN 1512-1232

simulation. An experimental and numerical modelling study was carried out by [17] for air-water flow in a bend of 630 mm radius of curvature. The flow configuration was horizontal to vertical though the bend and the pipe were made of transparent acrylic material of internal diameter 100-mm. Visual observations of the flow regime was done using a 200 Hz digital camera and an auto-regressive modelling method. They observed two flow regimes (Slug and churn flow) in the riser and three flow regimes (stratified, slug and enhanced slug) in the horizontal pipe section. The time dependent behaviour of two-phase flow was modelled by an in-house code named Solution Package for Hyperbolic Functions (SOPHY-2). In most of the tested data an excellent agreement was found between the experimental and modelling results. [2] used electrical capacitance tomography (ECT), high speed video and wire mesh sensor (WMS) to interrogate the effects of 90^o bends on two-phase air–silicone oil flows. The downstream pipe was kept horizontally while the upstream pipe was vertical. The characteristic PDF of void fraction for various flow rates were used to identify the flow regime upstream and downstream of the bend. They observed bubble, stratified, slug and semi-annular flows downstream of the vertical 90^o bend whiles the flow patterns exhibited upstream of the bend was the same as for the horizontal 90^o bend. [25] conducted an experimental study of the behaviour of two-phase air–water flow from the vertical to the horizontal through a 90^o bend using conductance probes technique. Void fraction measurement was done using the conductance probes (3 upstream and 6 downstream of the bend). The characteristic PDF of void fraction, power spectral density (PSD) of the time series of cross sectional average void fraction and visual observations were used to characterize the flow behaviour. For the horizontal pipe, plug, slug and stratified wavy flow patterns were seen while slug and churn flows in the vertical pipe.

1.3. Application of Artificial Neural Network (ANN) methodology to gas–liquid flows

[15] used ANN to process signals measured using a conductivity probe with the aim of classifying them into various flow patterns. They used visual map identification to validate the results of ANN model predictions. The ANN predictions were in good agreement with results of visual identification. A comparative model study was carried out by [21] for the prediction of pressure loss in five venture scrubbers. They compared the results of ANN prediction with a number of other models and concluded that predictions of the ANN models were better within the range of data tested. [30] used adopted inputs, all representing the characteristics of PSD, in training a three-layered feed forward ANN for the prediction of flow pattern. The model predictions gave good agreement to the observed flow patterns. A comparative study of CFD and ANN was carried out by [4]. They used both CFD and ANN for the prediction of pressure drop in two phase gas-liquid flows in 2-cm diameter, 6-m long pipe. The inputs to their ANN model were gas velocity, liquid velocity and tube inclination. Predictions from both models indicated that the CFD model performed better. [19] used ANN technique in predicting the pressure drop observed in three phase flow conditions. The resilient back-propagation algorithm was used to formulate the ANN model. This method gave correlation coefficients up to 98.82 %. [9] employed ANNs in characterizing observed flow pattern. The method was based upon an analysis of the PSD determined from the differential pressures measured within a horizontal pipe conveying. Over the range of data tested, model predictions were in good agreement with experimental data. Three different ANN algorithms were used by [24] to predict the observed void fractions and pressure drops for a two-phase flow in horizontal pipe. The model based on Levenberg Marquardt ANN algorithm with five processing elements gave the best prediction of void fraction and that based on back-propagation ANN algorithm with fifteen processing elements performed best for the prediction of pressure drop. Later, [22] applied a similar methodology for the prediction of flow regime of air–water flow in circular micro channels with reasonable success. [23] further applied ANNs in predicting frictional pressure drop in U-bends. Model predictions gave fairly accurate results of the frictional pressure drop across U-bends. Probabilistic Neural Network (PNN) was used by [16] for flow pattern identification. The range of gas and liquid superficial velocity tested were 0.0026-6.05 m/s and 0.03-2.5 m/s respectively. Within this range they observed five different flow patterns. The accuracy of the PNN when it was used to predict flow patterns was as high as 95.6%. An attempt to predict flow patterns for two phase flow was done by [20]. They used three pairs of inputs (gas

ISSN 1512-1232

velocity and liquid velocity, total pressure and liquid hold-up, Reynolds number for liquid and gas). The study was conducted in a horizontal pipe for four different flow regimes (Annular, Dispersed bubble, intermittent and stratified). Their model predicted over 80% of the flow patterns correctly and performed particularly well within the dispersed bubble region for all input pairs. In general the input pair of the natural logarithm of gas and liquid velocity gave better prediction.

The above literature survey show that limited work has been done in the use of MLP, ANN methodology for the prediction of average void fraction and PDF of void fraction in 90^o bends for two-phase flows through large diameter pipes. Most of the research has been centred on empirical and visualization techniques. Most of the works that use ANN methodology are those for pressure drop and flow pattern prediction.

2. Experimental Methodology

A summary of the experimental facility is provided in [1] and [2]. The experimental test section of the facility is made of a 67-mm internal diameter a transparent acrylic pipe of 6 m length.

A vertical 90^o bend with a radius of curvature 154 mm was maid of the straight pipe as shown in Figure 2. Downstream of the bend there is another straight pipe from which the air – silicone oil mixture enters a flexible pipe that takes it to the phase separator. The behaviour of the air – silicone oil mixture was examined using WMS. This technology, described by [28], can image the dielectric components in the pipe flow phases by measuring rapidly and continually the capacitances of the passing flow across several crossing points in the mesh.

The WMS was first placed at about 4.92 m (73 diameters) away from the mixing section. WMS was afterwards moved to a distance of about 0.21 m (3 diameters) after the bend. The experiments were all performed at an ambient laboratory temperature of 20±0.5 and a pressure of 1 bar.

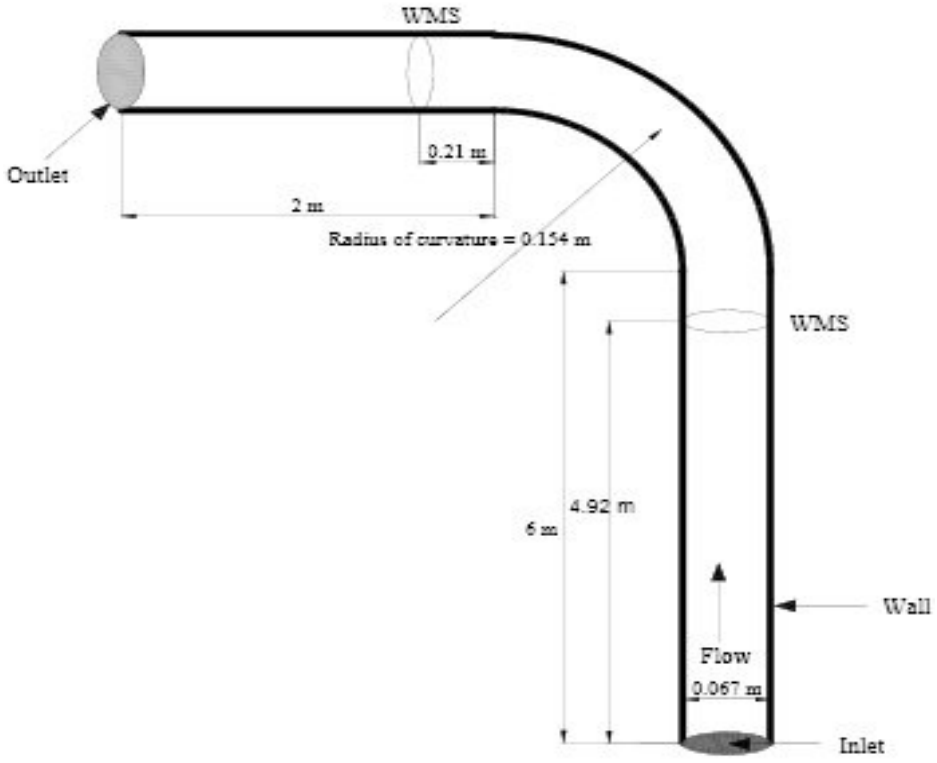


Figure 2: 3-D geometry of the computational domain showing the measurement locations and instrumentation

Table 1: Properties of the fluid at 1 bar and at the operating temperature of $20 \pm 0.5\,^{o}C$

Fluid	Density (kgm^{-3})	Viscosity ($kgm^{-1}s^{-1}$)	Surface tension (Nm^{-1})	Thermal conductivity (Wm^{-}

ISSN 1512-1232

				$^1K^{-1}$)
Air	1.18	0.000018	0.02	0.1
Silicone oil	900	0.00525		

The experimental data collected using WMS was time series void fractions for 38 experimental data sets before the bend and 33 experimental data sets for after the bend scenario.

The input parameters to the ANNs are the gas and liquid superficial velocities. Parameters like liquid and gas densities, liquid and gas viscosities, surface tension and thermal conductivity of liquid were not used in the formulation of the ANN model formulation as they were assumed to remain constant. The input parameters are summarised in Table 2. The output parameters are time averaged cross sectional area void fraction and PDF of void fraction.

Table 2. Inlet flow rates (model inputs)

	Before Bend			After Bend		
U_{SL}	0.05	0.14	0.38	0.05	0.14	0.38
U_{SG}	0.047	0.047	0.047	0.047	0.047	0.061
	0.061	0.061	0.061	0.061	0.061	0.288
	0.288	0.288	0.288	0.288	0.288	0.344
	0.344	0.344	0.344	0.344	0.344	0.404
	0.404	0.404	0.404	0.404	0.404	0.544
	0.544	0.544	0.544	0.544	0.544	0.709
	0.709	0.709	0.709	0.709	0.709	1.891
	0.945	0.945	0.945	0.945	1.891	2.363
	1.418	1.418	1.418	1.418	2.363	2.836
	1.891	1.891	1.891	1.891	2.836	4.73
	2.363	2.363	2.363	2.363	4.73	
	2.836	2.836	2.836	2.836		

3. Multilayer Perceptron Modelling (MLP)

Two MLP codes that mimic Artificial Neural Networks (ANN) models were developed in this paper for the prediction of average void fraction and PDF of void fraction of gas-liquid flow in 90^o bends. The MLP codes were developed in MATLAB and are based on the back-propagation algorithms (Gradient Descent (DC) and Levenberg-Marquardt (LM)). The model topography is such that each model consists of three layers (input, hidden and output layers). The transfer function used in the hidden layer is the *logsigmoid (logsig)* function given by Equation (1).

$$logsig(wx) = \frac{1}{(1+\exp(w*x))} \quad (1)$$

Where w is the weight function and x represents the vector input (gas and liquid superficial velocities in this case).

3.1. Gradient descent algorithm (GDMV)

This is regarded as the simplest of all the back-propagation algorithms. It is a first order algorithm and employs the generalised delta rule and weights are updated using the formula;

$$\Delta w_{ij}(t) = -\eta . \frac{\partial \varepsilon}{\partial w_{ij}} \quad (2)$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \quad (3)$$

ISSN 1512-1232

where η is the step size.

The gradient descent algorithm has a tendency to be slow in terms of convergence [29]. This slowness is often addressed by using a variable step size and a 'momentum term'. When the momentum term is added to Equation (3) Equation (4) is obtained;

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) + \alpha \Delta w_{ij}(t-1) \quad (4)$$

E is the error function, w_{ij} is the weight matrix and Δw_{ij} is the weight update for each iteration. The error function E, used for this work is Sum of Squares of error. Error here refers to the difference between the model output and experimental output.

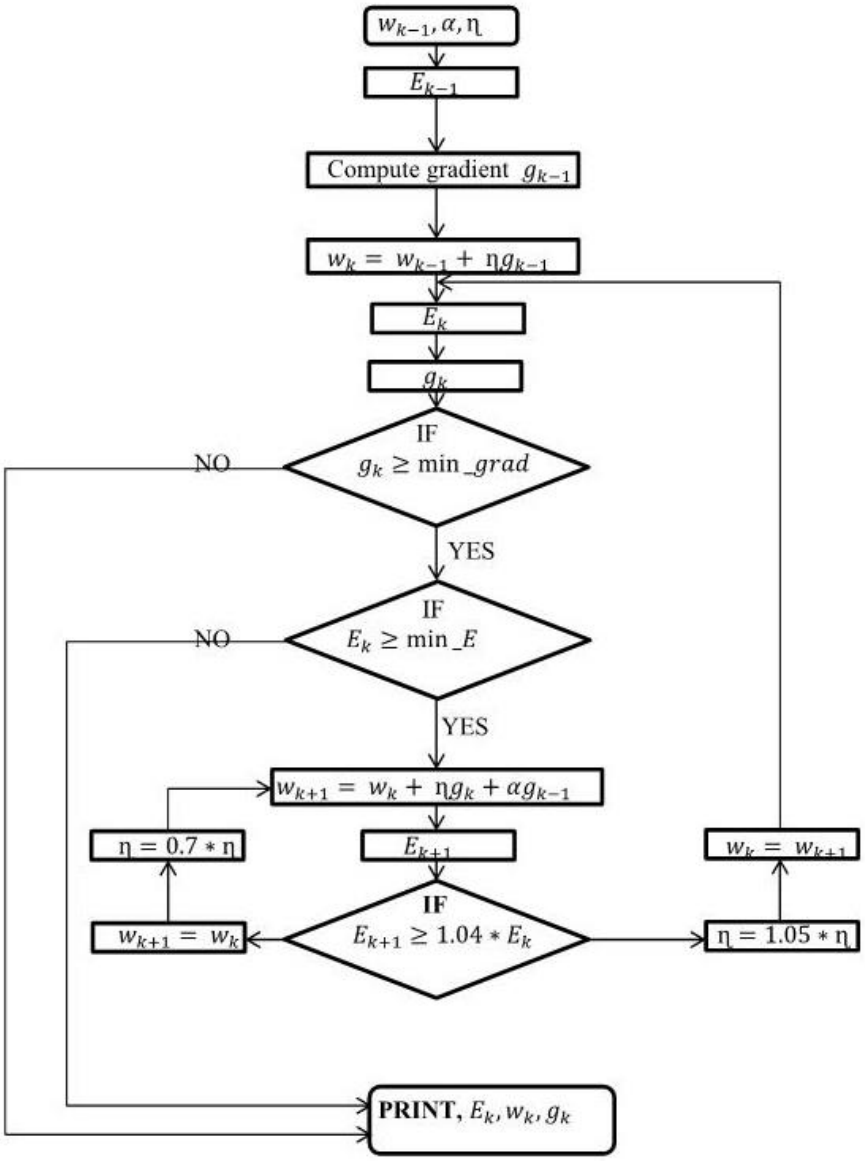


Figure 3: Gradient descent algorithm (with variable step size and momentum term) flowchart

The target is to minimise the partial derivative in Equation (2) (i.e. the gradient of the error function) to zero. First order algorithms has the inherent disadvantage of being slow, but also has a unique advantage, in that the gradient term can be minimised to zero with sufficient amount of iterations.

Figure 3 and 4 depict flow charts of the gradient descent and Levenberg-Marquardt algorithms respectively that were developed here.

ISSN 1512-1232

3.2. Levenberg-Marquardt algorithm (LM)

The Levenberg-Marquardt algorithm combines of Gauss-Newton algorithm (a second order algorithm) and gradient descent algorithm (a first order learning algorithm). Generally, second order algorithm converges much faster than the first order algorithms.

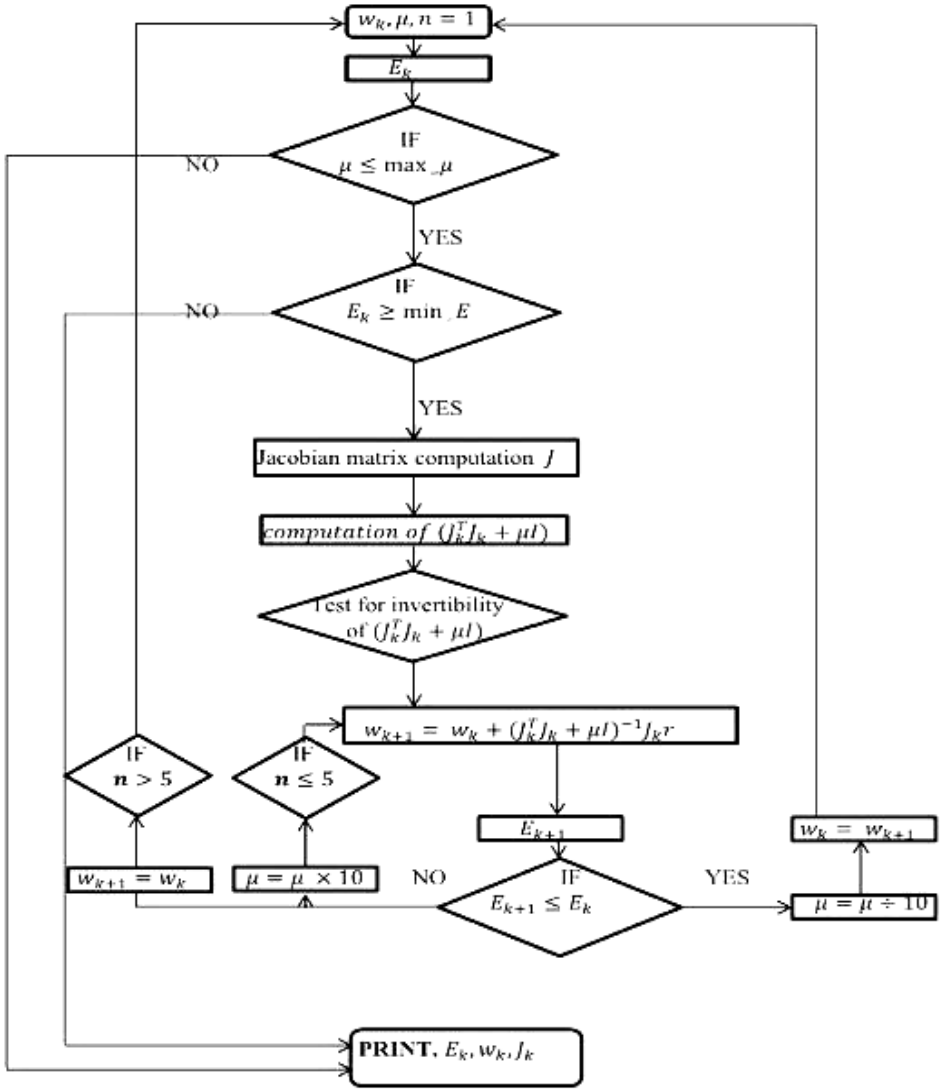


Figure 4: Levenberg-Marquardt algorithm flowchart

The update rule of Levenberg–Marquardt algorithm can be presented as

$$w_{k+1} = w_k - (J_k^T J_k + \mu I)^{-1} J_k e \tag{5}$$

J_k is the Jacobian matrix given by;

ISSN 1512-1232

$$J_k = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w_1} & \frac{\partial e_{1,1}}{\partial w_2} & \cdots & \frac{\partial e_{1,1}}{\partial w_N} \\ \frac{\partial e_{1,2}}{\partial w_1} & \frac{\partial e_{1,2}}{\partial w_2} & & \frac{\partial e_{1,2}}{\partial w_N} \\ \cdots & \cdots & & \cdots \\ \frac{\partial e_{1,M}}{\partial w_1} & \frac{\partial e_{1,M}}{\partial w_2} & \cdots & \frac{\partial e_{1,M}}{\partial w_N} \\ \frac{\partial e_{P,1}}{\partial w_1} & \frac{\partial e_{P,1}}{\partial w_2} & \cdots & \frac{\partial e_{1,1}}{\partial w_N} \\ \frac{\partial e_{P,2}}{\partial w_1} & \frac{\partial e_{P,2}}{\partial w_2} & \cdots & \frac{\partial e_{P,2}}{\partial w_N} \\ \cdots & \cdots & & \cdots \\ \frac{\partial e_{P,M}}{\partial w_1} & \frac{\partial e_{P,M}}{\partial w_2} & & \frac{\partial e_{P,M}}{\partial w_N} \end{bmatrix}$$ Where e is given by $$e = \begin{bmatrix} e_{1,1} \\ e_{1,2} \\ \cdots \\ e_{1,M} \\ \cdots \\ e_{P,1} \\ e_{P,2} \\ \cdots \\ e_{P,M} \end{bmatrix}$$

where, μ is always positive, called combination coefficient and **I** is the identity matrix.

As the combination of the steepest descent algorithm and the Gauss–Newton algorithm, the Levenberg–Marquardt algorithm switches between the two algorithms during the training process.

3.3. MLP Model Optimization

The optimisation process involves determining the network topology and weights that gives best model performance when tested using certain performance equations. It is important to determine the optimum number of neurons in the hidden layer. This optimum number depends on the number of input and output units, the number of training cases, the amount of noise in the targets, the complexity of the error function, the network architecture, and the training algorithm. In most cases according to [19], there is no straight forward method of determining the optimal number of hidden units without training using different numbers of hidden units and estimating the generalization error of each.

In the present work, a single hidden layer with multiple neurons units is used as previous study by [24] showed that a single hidden layer with sufficient number of neurons performed satisfactorily well. The details of the concept of hidden layer can be found elsewhere, example in [18]. In the hidden layer the numbers of processing elements are optimized by varying the number from 1 to 25.The basic approach used in constructing the successful model was trial and error. The generalization error (cross validation error) of each inspected network design was visualized and monitored carefully through plotting the governing statistical parameters such as correlation coefficient, root mean squared errors, and average absolute relative error of each inspected topology.

It is worthy of note that various network topology could be employed for MLP model development. However, the back-propagation network with feed-forward algorithm was chosen here as this has performed satisfactorily well in previous works such as [7], [8], [19], [24] and [30] among others.

3.4. Objective Function and Performance of ANN

The objective function provides the basis for performance evaluation and network algorithm selection. A suitable objective function which provides a good basis for numerical computation and attainment of set goals must be chosen. In this work, sum of squares of error is used as the objective function and it is given by equation (6).

$$E = \frac{1}{2}\sum_{i=1}^{N}(O_i - t_i)^2 \quad (6)$$

The performance of the network is checked using the following parameters: Mean Squared Error (MSE),

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \quad (7)$$

Average Absolute Relative Error (AARE)

$$AARE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{(y_i - x_i)}{x_i}\right| \quad (8)$$

ISSN 1512-1232

The AARE eliminates the possibility of obtaining negative values of relative error. The aim is to obtain values of AARE close to zero.

Cross-correlation coefficient (*R*)

$$R = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2 \sum_{i=1}^{N}(y_i - \bar{y})^2}} \quad (9)$$

A value of R unity (1) means there is a perfect correlation between experimental and predicted results, while a value of R close to zero would indicate that there is little or no correlation between experimental and predicted data [24]. Thus models capable of predicting experimental data such that the value of R is close to one would be considered as good models.

3.5. Developed MLP Algorithms Codes (program development)

Data collection and partitioning, filtering and screening procedure, data randomization, pre-processing, and post-processing were done before running the models. A total number of 38 data sets before the bend and 33 data sets after he bend were utilized for the purpose of this study for modelling ends. Input data are.

The partitioning ratio of 4:1:1 is known to yield better training and testing results (but depends on the number of data set used for training). Other common partitioning ratios are 2:1:1 and 3:1:1 [24]. For this work, a partitioning ratio of 3:1:1 was adopted. The training set is used to develop and adjust the weights in a network. The validation set is used to ensure the generalization of the developed model during the training phase, and the test set is used to examine the final performance of the model. The primary concerns are to ensure two things: (a) the training set contains enough data, and suitable data distributed evenly to cover the entire range of data, and (b) there is no unnecessary similarity between the data in different sets. The number of iterations for each model was initially put at 100,000 iterations but if performance does not improve much after 10,000 iterations then training is stopped.

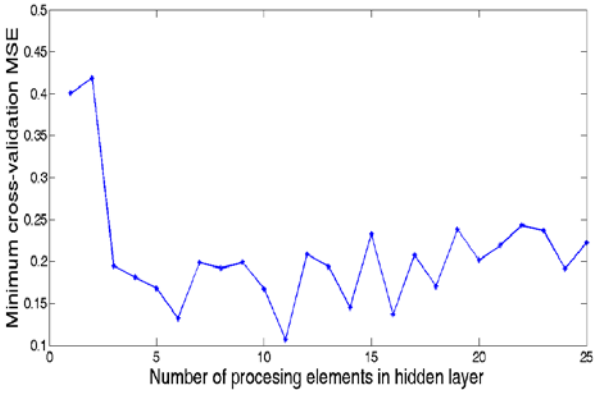
The MLP models developed for the prediction of average void fraction before the bend based on gradient descent algorithm and PDF of void fraction after the bend based on Levenberg-Marquardt algorithm are given in the appendix.

4. RESULTS

In this section results model optimization and performance of the developed MLP models are presented.

4.1 Result of MLP model optimization

Figure 4 is a plot of minimum cross-validation MSE against number of processing elements in the hidden layer of a selected MLP. The optimum network configuration is chosen as that which gives the least value of cross-validation MSE, which is the network configuration with the best generalisation capability.



ISSN 1512-1232

Figure 5: Minimum cross-validation MSE versus number of processing elements in hidden layer for PDF of void fraction using Levenberg-Marquardt algorithm (before bend).

Table 3 gives the optimum number of neurons in the hidden layer for each MLP model developed. All further analysis will be based on these optimal configurations.

Table 3: Optimum number of neurons

Algorithm	Optimum number of processing elements	
	Average void fraction	PDF of void fraction
GDMV (before bend)	2	4
LM (before bend)	3	11
GDMV (after bend)	16	4
LM (after bend)	2	4

4.1. Performance of the Optimum Network Configurations for Various MLP Codes

Table 4 gives performance of the optimum network configurations for various MLP codes for training, validation and testing.

Table 4: Performance of the optimum network configurations for various MLP codes developed

Algorithm	Performance	Training		Validation		Testing	
		Average void fraction	PDF of void fraction	Average void fraction	PDF of void fraction	Average void fraction	PDF of void fraction
GDMV (before bend)	MSE	9.2514×10^{-4}	0.0284	7.4305×10^{-4}	0.0957	7.8331×10^{-4}	0.0270
	AARE	0.0265	0.7428	0.0939	3529.5	0.0388	8.7071
	R	0.9980	0.9390	0.9991	0.5317	0.9988	0.9255
LM (before bend)	MSE	4.1445×10^{-4}	0.0089	0.0024	0.1065	0.0026	0.0279
	AARE	0.0157	0.6537	0.1590	1.3139	0.0274	0.7866
	R	0.9991	0.9743	0.9977	0.5486	0.9958	0.9342
GDMV (after bend)	MSE	0.0074	0.0686	0.1276	0.2646	0.0267	0.1801
	AARE	0.0567	0.9702	0.3362	32.9285	0.1961	1.5740
	R	0.9855	0.7532	0.1219	0.0808	0.9156	0.4507
LM (after bend)	MSE	0.1709	0.1605	0.1101	0.2731	0.1820	0.3227
	AARE	0.3062	1.0291	0.2622	1.0789	0.7013	2.7625
	R	0.5893	0.5841	0.7863	0.3482	0.9330	0.3458

The performance results for the test set is of greater significance here as it gives an indication of the model's ability to predict the parameters of interest, which is the network's generalisation property. The Average Absolute Relative Error is used in selecting the best model, among the ones developed. Small values of AARE and values of correlation coefficient (R) close to unity (1) indicate good model performance [24]. The model chosen, from the models tested, for the prediction of average void fraction or PDF of void fraction before or after the bend is the one with the least value of AARE.

4.2. Performance Analysis before Vertical 90^o Bend

Results of model performance before the bend are presented in this section.

4.2.1 Performance of MLP models for the prediction of average void fraction before bend

Result of Table 4 shows that both MLP algorithms developed for the prediction of average void fraction before the vertical 90^o bend were good as seen from the small values of AARE and the

ISSN 1512-1232

closeness of R to unity in both cases. However, the model based on the Levenberg-Marquardt algorithm is preferred since it has the smallest value of AARE. The Results for training and validation also showed that both MLP models performed satisfactorily, but only result of test data was used as bases for assessing model performance. Figures 6a and 6b show the variation of MSE with number of iterations for the two MLP models (Gradient descent and Levenberg-Marquardt algorithm) developed for the prediction of average void fraction before the bend. The plots show a decrease in the MSE with number of iteration, which is an indication of error convergence. Iteration is stopped after 10000 and 100 iterations respectively as there is no significant improvement after these points.

It is important to note that besides closeness of model results to experimental data, the MSE curve for both validation and testing must follow similar pattern for the models to be reliable over the entire range of data set [24]. This is to eliminate the problem associated with poor data distribution and/or partitioning for training, validation and testing. It is also important to note that validation and testing data do not participate in the weight update process, though validation data is used to stop training when necessary to avoid over-fitting. The performance plots of Figures 6a and 6b show similar trends for validation and testing, thus the models show good generalisation property and can be relied upon over the entire range of data set.

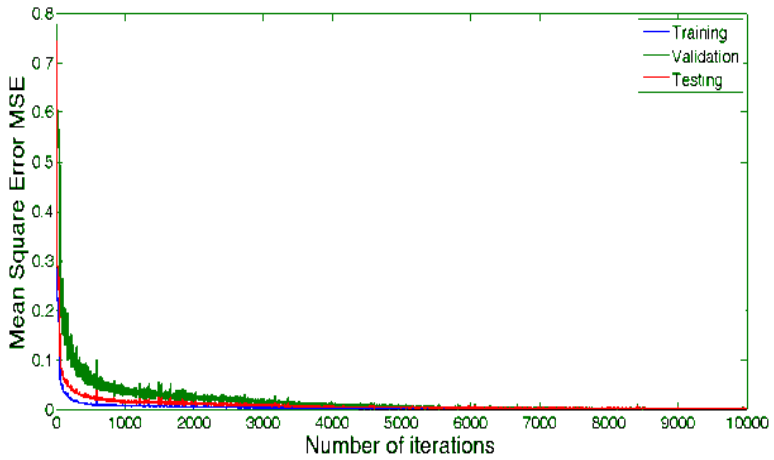


Figure 6a (GDMV model): Variation of MSE with number of iterations for prediction of average void fraction before bend

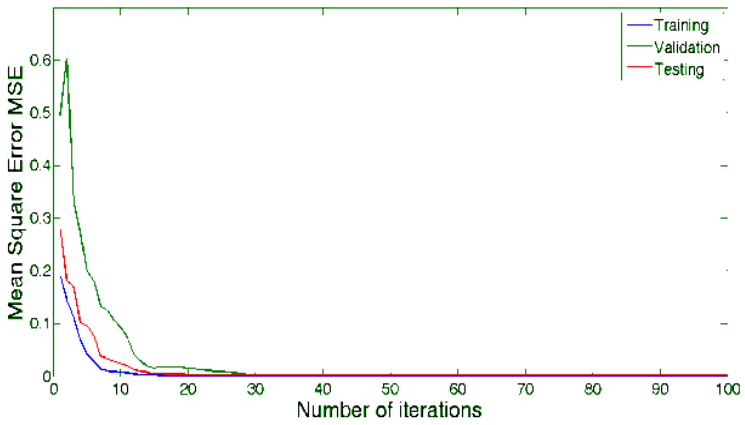


Figure 6b (LM model): Variation of MSE with number of iterations for prediction of average void fraction before bend

Figures 7a and 7b are regression plots of test data for the two MLP models developed for the

ISSN 1512-1232

prediction of average void fraction before the vertical 90^o bend. The plots are linear with most of the points close to or directly on the line. Also the intercept of the line is close to zero and slope close to one, thus given an almost perfect correlation between experimental and predicted values of average void fraction before bend. These plots further justify the reliability of the proposed models.

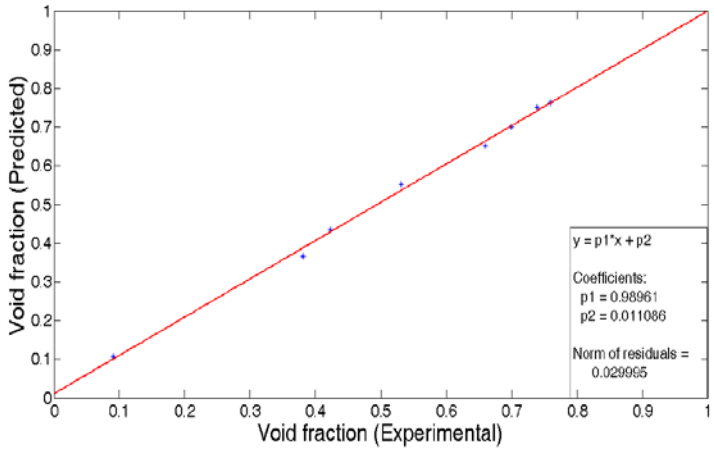


Figure 7a (GDMV model): Regression plot of test data before 90^o bend for prediction of average void fraction.

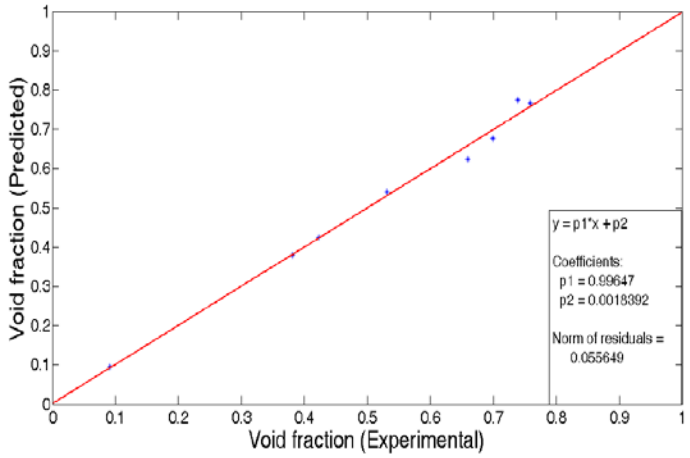


Figure 7b (LM model): Regression plot of test data before 90^o bend for prediction of average void fraction.

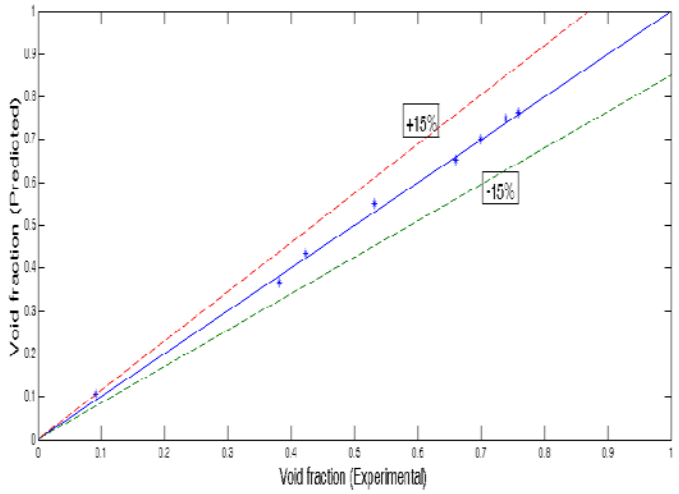


Figure 7c (GDMV model): Comparison between predicted and experimental void fraction for test

ISSN 1512-1232

data before 90°.

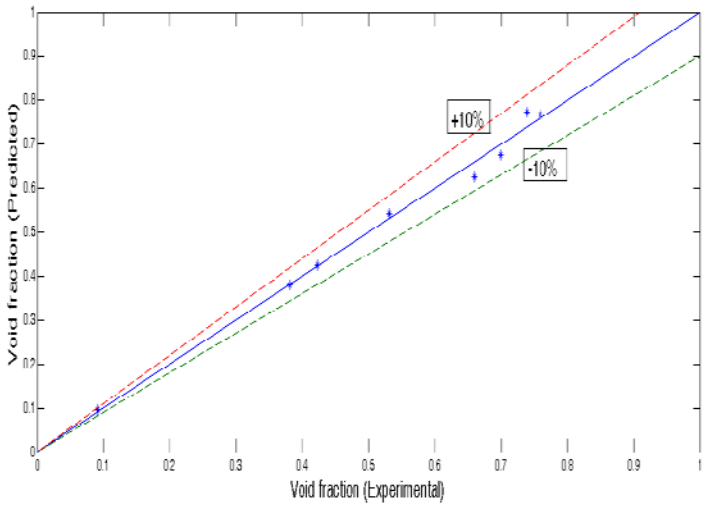


Figure 7d (LM model): Comparison between predicted and experimental void fraction for test data before 90°.

Results of Figure 7 c and & d show that all the test data predicted by the GDMV model and LM model were with ± 15% and ± 10% of the experimental data respectively. This further confirms that the LM model performed better for the prediction of average void fraction before the bend.

Table 5 gives a comparison between experimental and predicted average void fraction before the bend. The results showed that predictions for both models were close to experimental results especially at low liquid superficial velocities.

Table 5: Comparison between experimental and predicted average void fraction of air/silicone oil two-phase flow before vertical 90° bend

U_{SL}	U_{SG}	Experimental average void fraction	GDMV model predicted average void fraction	LM model predicted average void fraction
0.050	0.344	0.3806	0.3668	0.3789
0.050	1.418	0.6592	0.6522	0.6249
0.050	1.891	0.6990	0.7008	0.6759
0.050	2.836	0.7591	0.7628	0.7665
0.140	0.047	0.0910	0.1067	0.0954
0.140	0.544	0.4225	0.4352	0.4250
0.140	0.945	0.5315	0.5513	0.5409
0.140	2.836	0.7389	0.7503	0.7739

4.2.2. Performance of MLP models for the prediction of PDF of void fraction before bend

Both MLP models developed for the prediction of PDF of void fraction after the bend also performed excellently. However, the model based on Levenberg-Marquardt algorithm is preferred since it has a smaller value of AARE. MSE plot (Figures 8a and 8b) also show a consistent pattern

ISSN 1512-1232

for training, validation and testing.

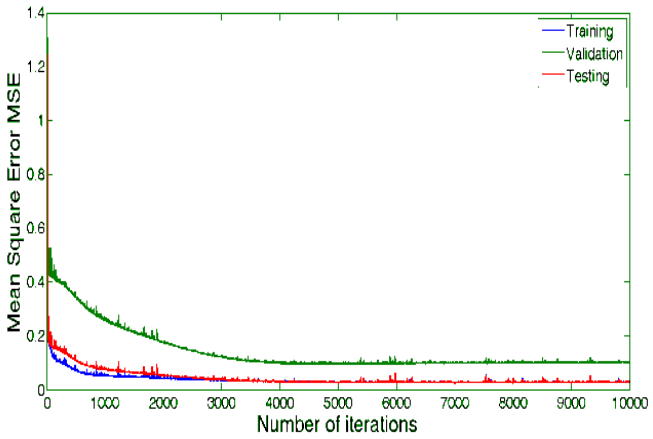


Figure 8a (GDMV model): Variation of MSE with number of iterations for prediction of PDF of void fraction before the bend

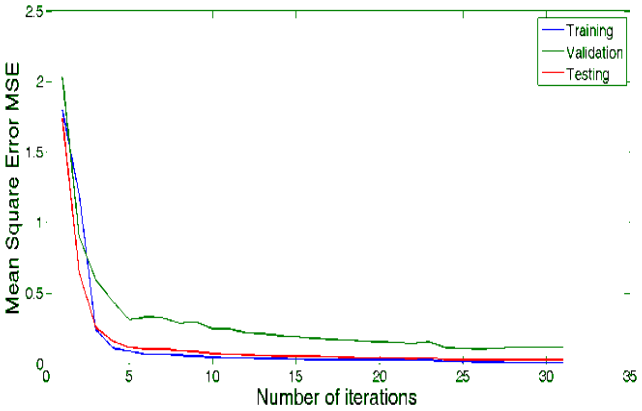
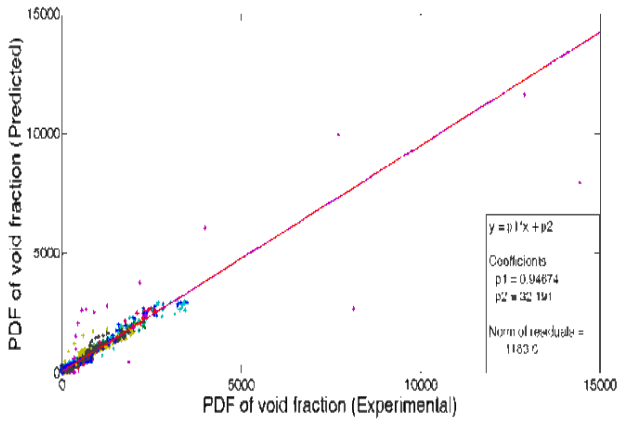


Figure 8b (LM model): Variation of MSE with number of iterations for prediction of PDF of void fraction before the bend

Regression plot for test data is shown in Figures 9a and 9b. These plots further validate the MLP models.



ISSN 1512-1232

Figure 9a (GDMV model): Regression plot of test data before 90^o bend for prediction of PDF of void fraction

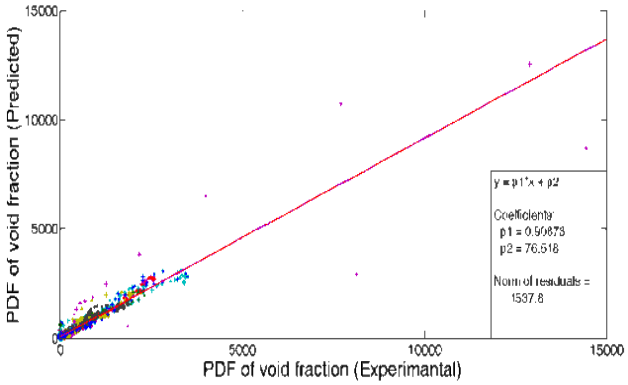


Figure 9b (LM model): Regression plot of test data before 90^o bend for prediction of PDF of void fraction

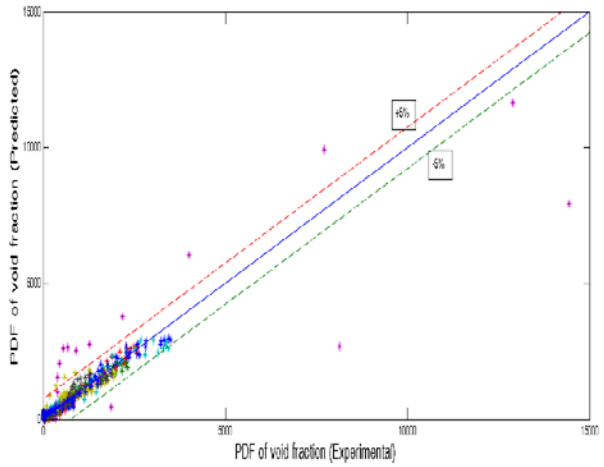


Figure 9c (GDMV model): Comparison between Predicted and Experimental PDF of void fraction before bend

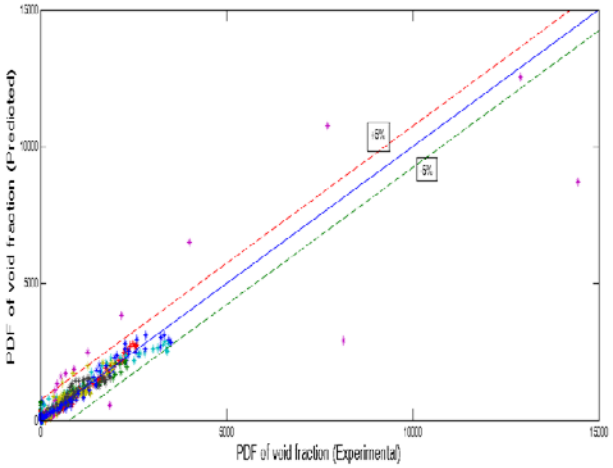


Figure 9d (LM model): Comparison between Predicted and Experimental PDF of void fraction

ISSN 1512-1232

before bend

Figures 9c and 9d show that both MLP models developed for the prediction of PDF of void fraction before the bend predicted over 90% of the data with ±5% of the experimental values with the model based on Levenberg-Marquardt algorithm again giving slightly better predictions.

Figures 10a to 10h show comparisons (at various values of U_{SG} and U_{SL}) of the predicted PDF of void fraction before the bend with those from experimental data for MLP model based on the Levenberg-Marquardt algorithm. The results depicted show that the MLP model predicts the PDF of void fraction before the vertical 90^o bend to a reasonable level of accuracy over a wide range of data set.

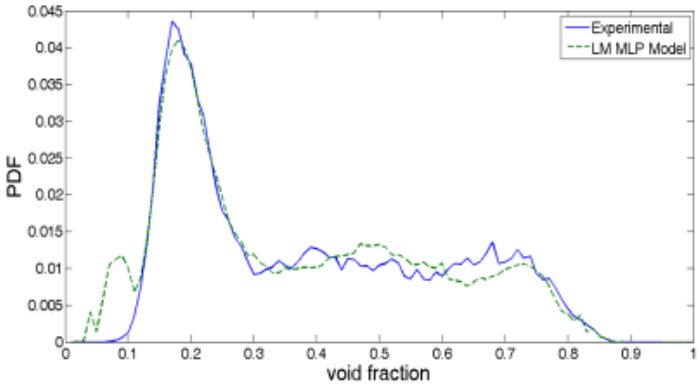


Fig 10a (LM model): Comparison between Predicted and Experimental PDF of void fraction before bend for U_{SL}=0.05 m/s U_{SG}=0.344 m/s

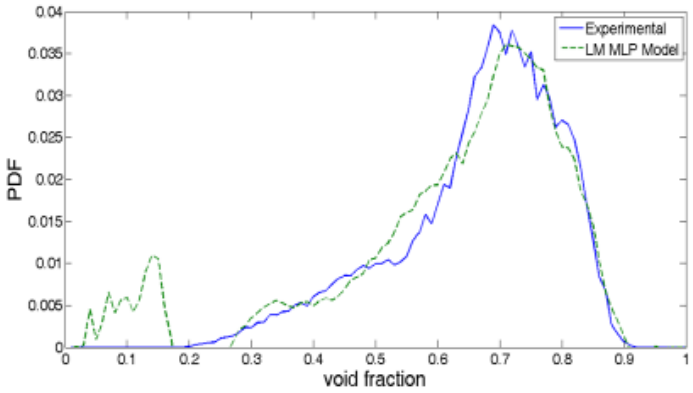
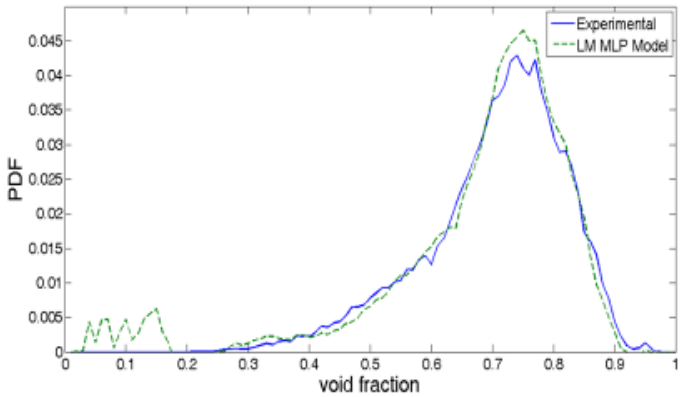


Fig 10b (LM model): Comparison between Predicted and Experimental PDF of void fraction before bend for U_{SL}=0.05 m/s U_{SG}=1.418 m/s



ISSN 1512-1232

Fig 10c (LM model): Comparison between Predicted and Experimental PDF of void fraction before bend for U_{SL}=0.05 m/s U_{SG}=1.891 m/s

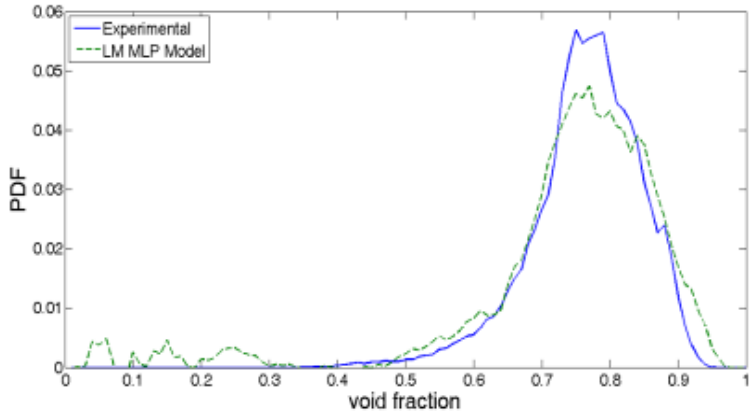


Fig 10d (LM model): Comparison between Predicted and Experimental PDF of void fraction before bend for U_{SL}=0.05 m/s U_{SG}=2.836 m/s

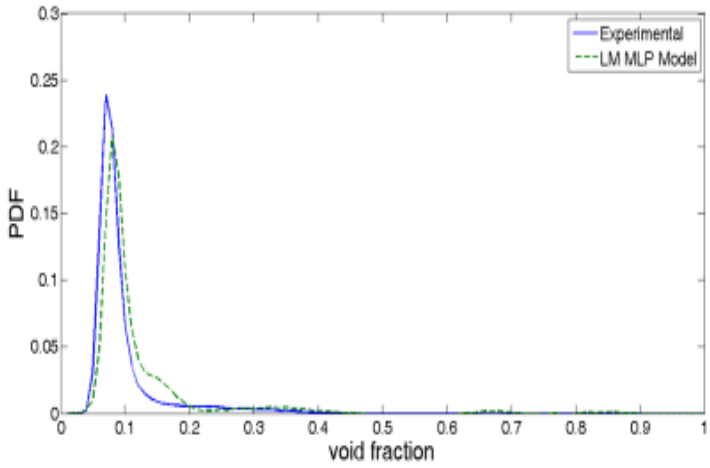
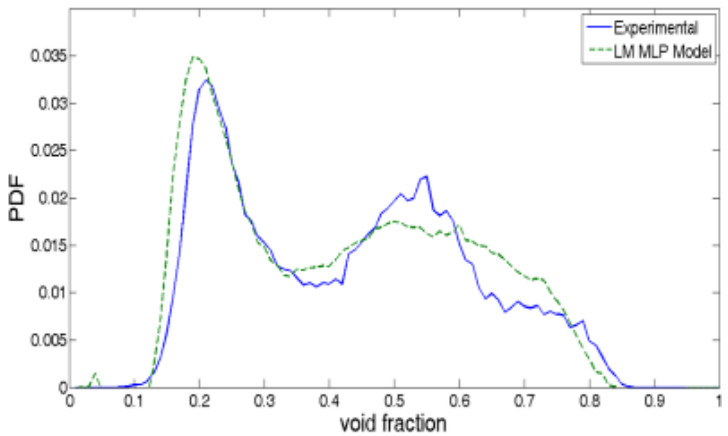


Fig 10e (LM model): Comparison between Predicted and Experimental PDF of void fraction before bend for U_{SL}=1.14 m/s U_{SG}=0.047 m/s



ISSN 1512-1232

Fig 10f (LM model): Comparison between Predicted and Experimental PDF of void fraction before bend for U_{SL}=1.14 m/s U_{SG}=0.544 m/s

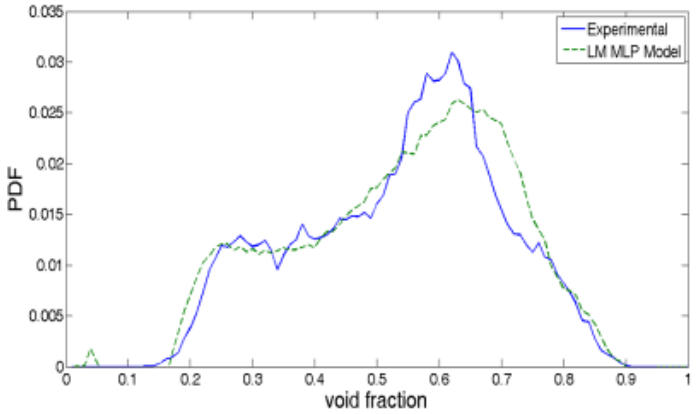


Fig 10g (LM model): Comparison between Predicted and Experimental PDF of void fraction before bend for U_{SL}=1.14 m/s U_{SG}=0.945 m/s

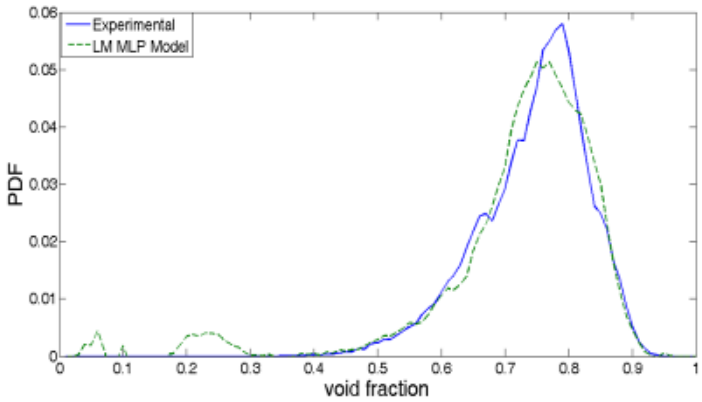


Fig 10h (LM model): Comparison between Predicted and Experimental PDF of void fraction before bend for U_{SL}=1.14 m/s U_{SG}=2.836 m/s

Careful inspections of Figures 10a-h reveal that models predictions were particularly good in Figures 10a, 10, 10c, 10d, 10g and 10h. The results of Figures 10a, 10b, 10c, 10d, 10g and 10h show that the model can be used to classify flow pattern before the bend with great confidence.

4.3. Performance Analysis Immediately after Vertical 90^o Bend

4.3.1 Performance of MLP models for the prediction of average void fraction after bend

A first look at the results of Table 4 for test data set gives an indication of excellent model performance for both MLP algorithms developed for the prediction of average void fraction immediately after the vertical 90^o bend as can be seen from the small values of AARE and the closeness of R to unity in both cases. However results from validation data set show that the values of R for both models were not close to unity. For an MLP model to be valid, both validation and testing performance curve must follow similar pattern. Figures 11a and 11b give the variation of MSE with number of iterations for the two MLP models developed for the prediction of average void fraction immediately after the bend. Figure 11a show a significant divergence in the patterns traced by the validation MSE and the testing MSE. The implication of this would be poor generalisation capability of the model. This significant divergence could be attributed to two things,

ISSN 1512-1232

first; the flow immediately after the bend is not fully developed and second; poor data distribution and/or partitioning among training, validation and testing. Similar conclusion can be drawn from Figure 11b. Even though the divergence between the MSE plots for validation and testing is not well pronounced in this case there is still a significant difference in their patterns.

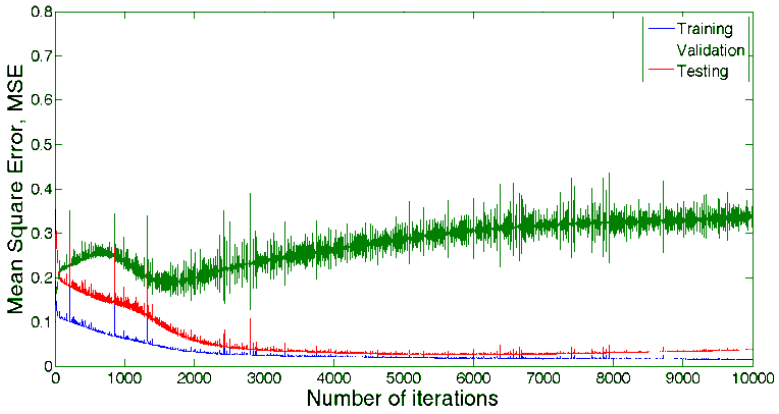


Figure 11a (GDMV model): variation of MSE with number of iterations for prediction of average void fraction after bend.

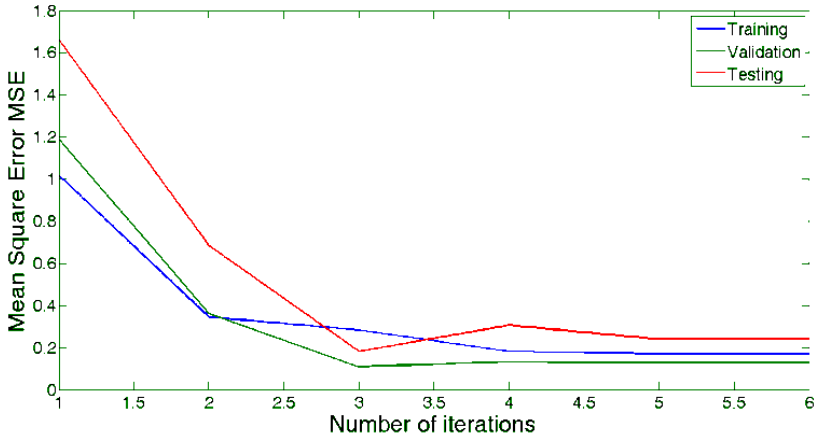


Figure 11b (LM model): variation of MSE with number of iterations for prediction of average void fraction after bend.

Figures 12a and 12b are regression plots of test data for the two MLP models developed for the prediction of average void fraction immediately after the vertical 90^{o} bend. These plots again suggest that the models give a good representation of the data. However Figures 13a and 13b, which represents the regression plots for the validation data sets, indicates that the models can only be useful for a limited range of data set.

ISSN 1512-1232

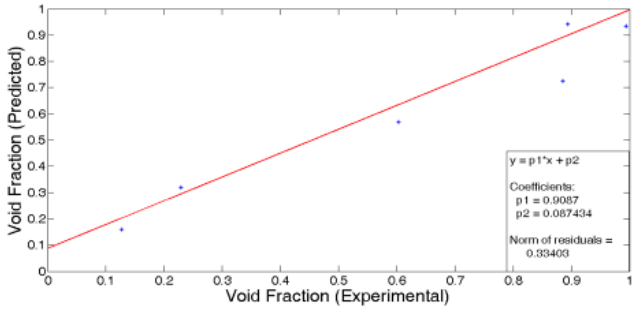


Figure 12a (GDMV model): Regression plot of test data after 90^{o} bend for prediction of average void fraction

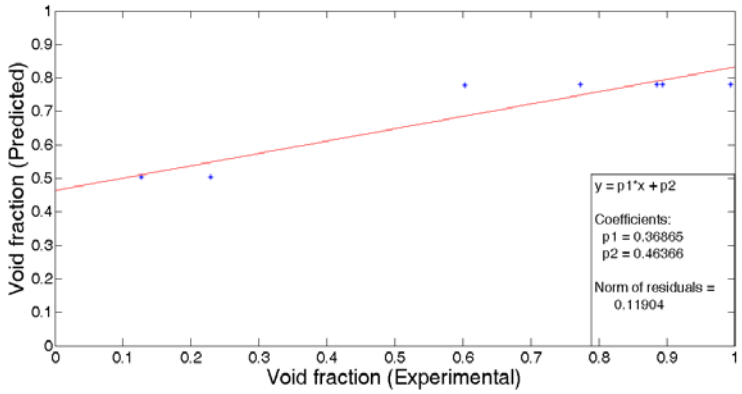


Figure 12b (LM model): Regression plot of test data after 90^{o} bend for prediction of average void fraction

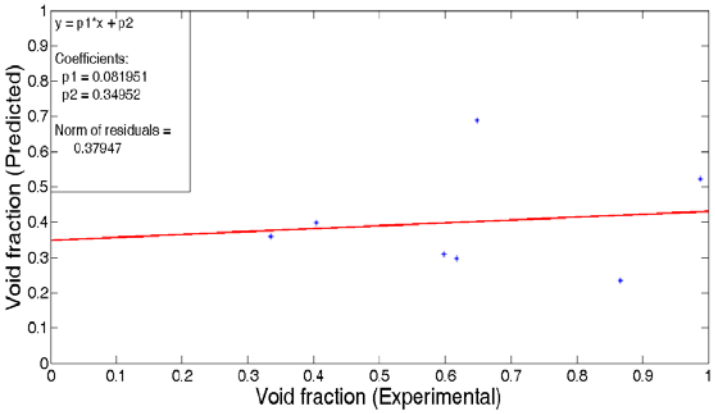


Figure 13a (GDMV model): Regression plot of validation data after 90^{o} bend for prediction of average void fraction

ISSN 1512-1232

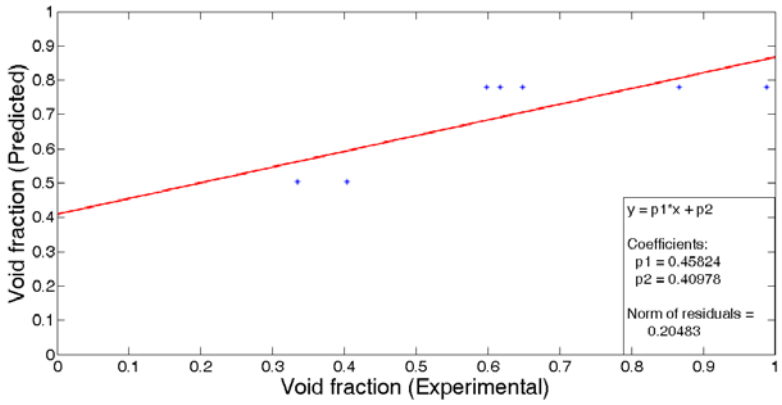


Figure 13a (LM model): Regression plot of validation data after 90^o bend for prediction of average void fraction

The model based on the gradient descent with momentum term and variable steps size gives a better performance when comparing the two MLP models based on their AARE.

Table 6:
Comparison between experimental and predicted average void fraction of air/silicone oil two-phase flow after vertical 90^o bend

U_{SL}	U_{SG}	Experimental average void fraction	GDMV model predicted average void fraction	LM model predicted average void fraction
0.050	1.418	0.9941	0.9344	0.7807
0.050	1.891	0.8941	0.9415	0.7807
0.050	2.836	0.8857	0.7256	0.7807
0.140	0.047	0.1274	0.1599	0.5033
0.140	0.544	0.6032	0.5680	0.7776
0.140	1.891	0.7728	1.0578	0.7807
0.380	0.061	0.2289	0.3196	0.5033

4.3.2. Performance of MLP models for the prediction of PDF of void fraction after bend

Results of Table 4 shows that both MLP algorithms developed for the prediction of PDF of void fraction after the vertical 90^o bend does not give a good representation of the testing and validation data sets. This is seen from fairly large values of AARE and the lack of closeness of R to unity in both cases. However, the model based on gradient descent algorithm is preferred since it

ISSN 1512-1232

has a smaller value of AARE. Figures 14a and 14b give the variation of MSE with number of iterations for the two MLP models developed for the prediction of PDF of void fraction after the bend. The plots show a divergence in the performance function for validation and testing. This further highlights the lack of generality of the models.

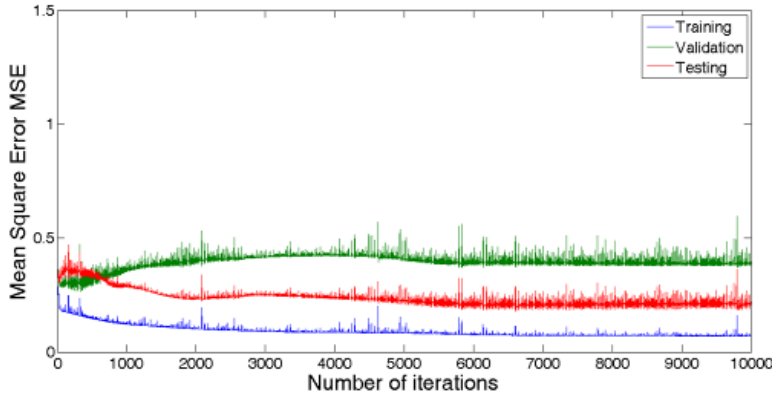


Figure 14a (GDMV model): Variation of MSE with number of iterations for prediction of PDF of void fraction after the bend.

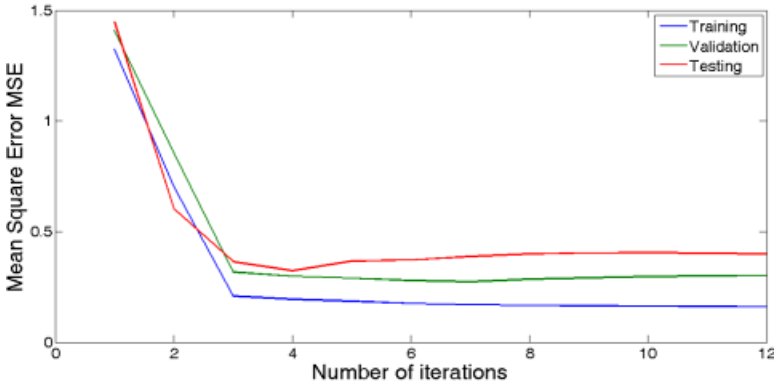
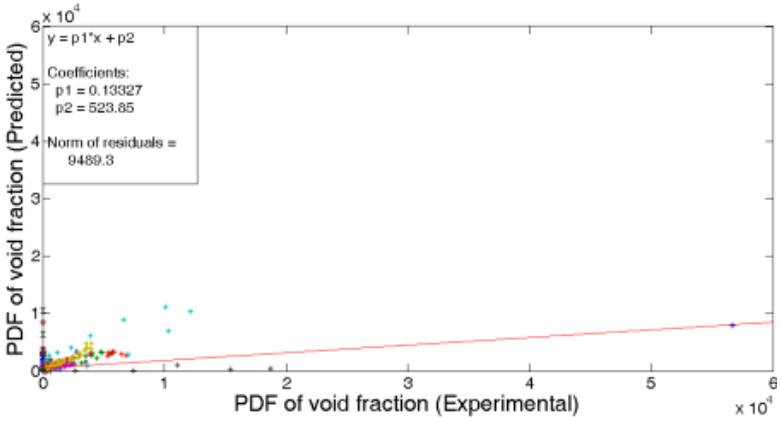


Figure 14b (LM model): Variation of MSE with number of iterations for prediction of PDF of void fraction after the bend.

Figures 15a and 15b are regression plots for test data for the two MLP models developed for the prediction of PDF of void fraction immediately after the vertical 90^o bend. Careful inspection show that the intercept of both curves are close to the origin, however a number of data points do not fall close to the line of best fit. The results further indicate the lack of reliability of the models in predicting the PDF of void fraction immediately after the vertical 90^o bend.



ISSN 1512-1232

Figure 15a (GDMV model): Regression plot of test data after 90^o bend of gradient descent with momentum term and variable step size model for prediction of PDF of void fraction

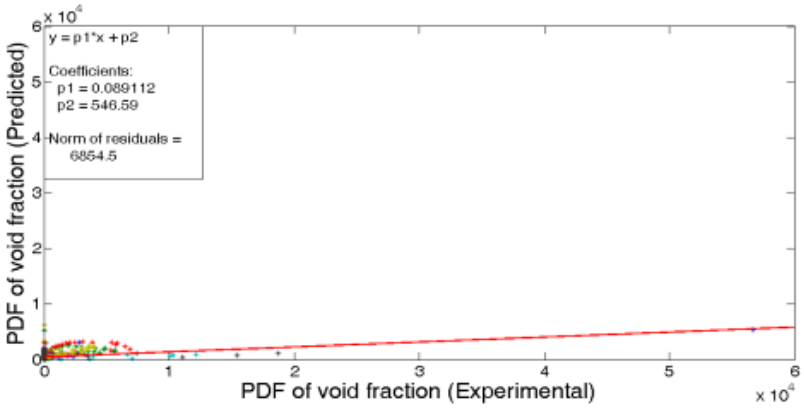


Figure 15b (LM model): Regression plot of test data after 90^o bend of gradient descent with momentum term and variable step size model for prediction of PDF of void fraction

Figures 16a to 16g give comparisons between predicted and experimental PDF of void fraction (at various values of U_{SG} and U_{SL}) for MLP model based on gradient descent algorithm. The results depicted show that the developed model predicts the PDF of void fraction for gas-liquid flow after vertical 90^o bends only within limited range of data sets.

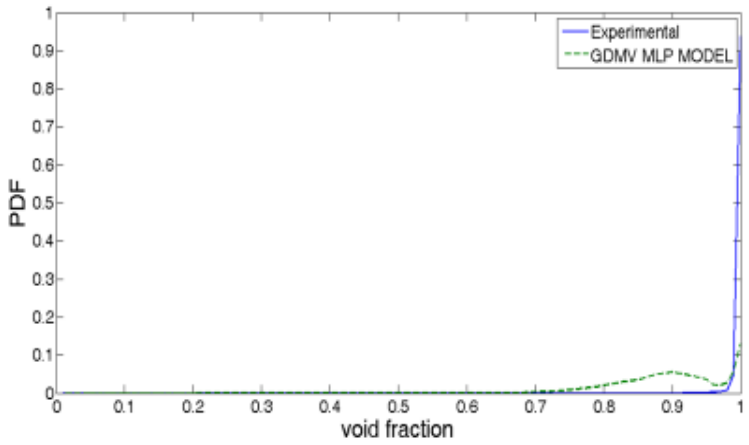
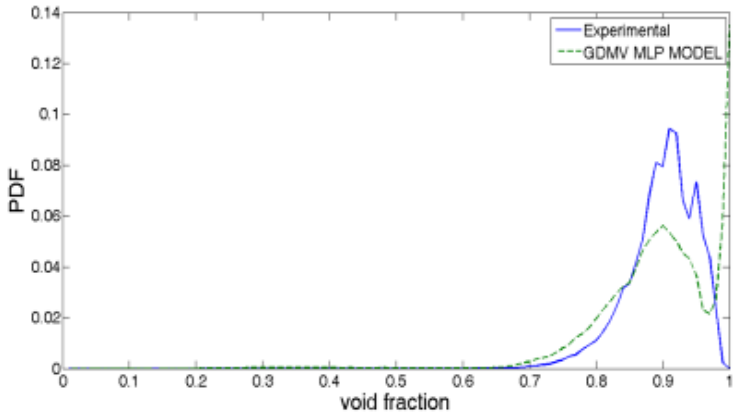


Fig 16a (GDMV model): Comparison between Predicted and Experimental PDF of void fraction after bend for U_{SL}=0.05 m/s U_{SG}=1.418 m/s



ISSN 1512-1232

Fig 16b (GDMV model): Comparison between Predicted and Experimental PDF of void fraction after bend for U_{SL}=0.05 m/s U_{SG}=1.891 m/s

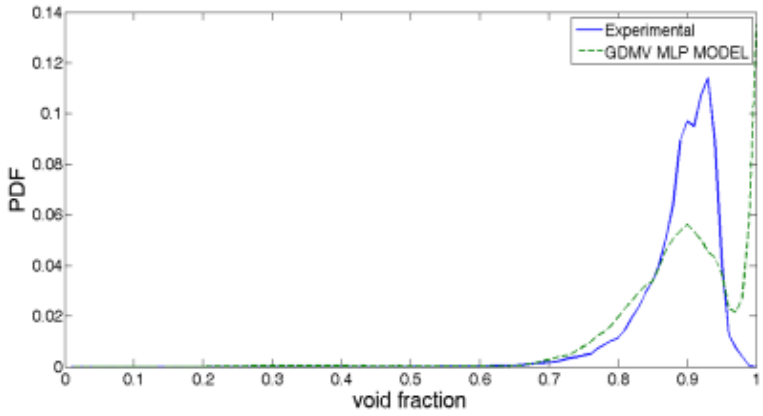


Fig 16c (GDMV model): Comparison between Predicted and Experimental PDF of void fraction after bend for U_{SL}=0.05 m/s U_{SG}=2.863 m/s

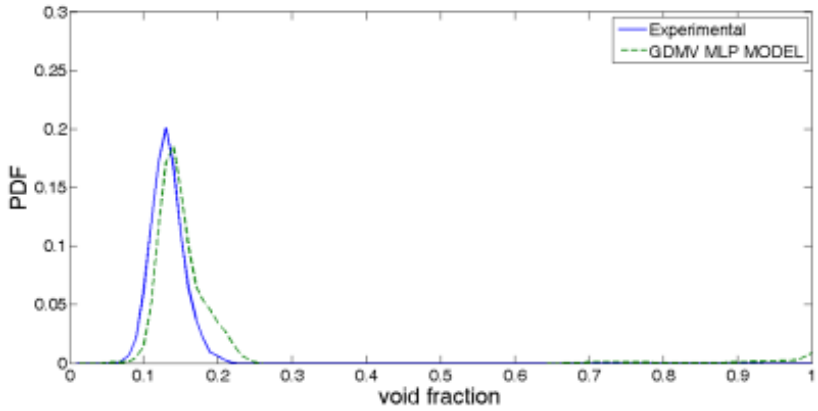


Fig 16d (GDMV model): Comparison between Predicted and Experimental PDF of void fraction after bend for U_{SL}=0.14 m/s U_{SG}=0.047 m/s

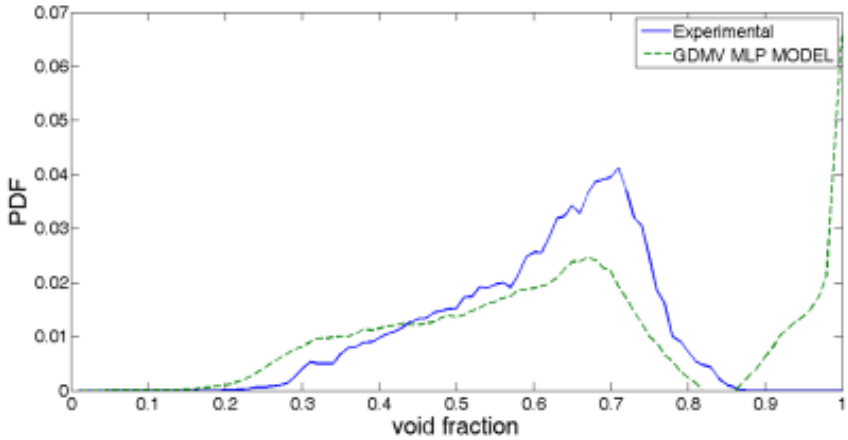


Fig 16e (GDMV model): Comparison between Predicted and Experimental PDF of void fraction after bend for U_{SL}=0.14 m/s U_{SG}=0.544 m/s

ISSN 1512-1232

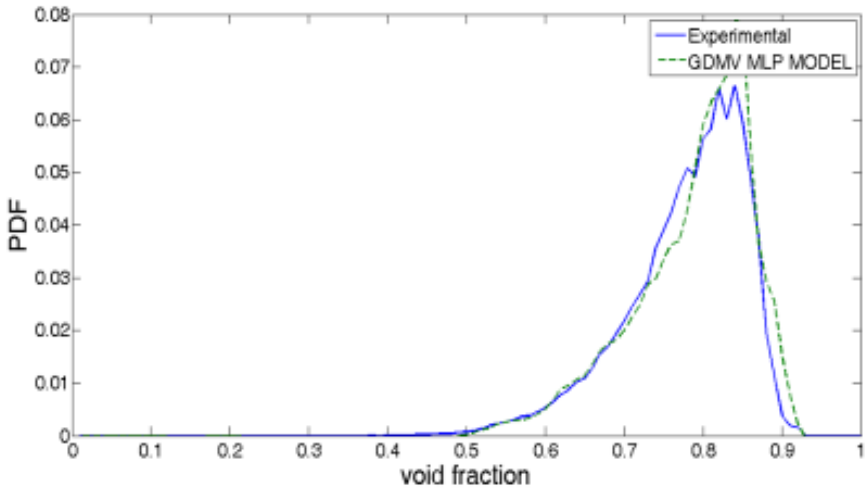


Fig 16f (GDMV model): Comparison between Predicted and Experimental PDF of void fraction after bend for U_{SL}=0.14 m/s U_{SG}=1.891 m/s

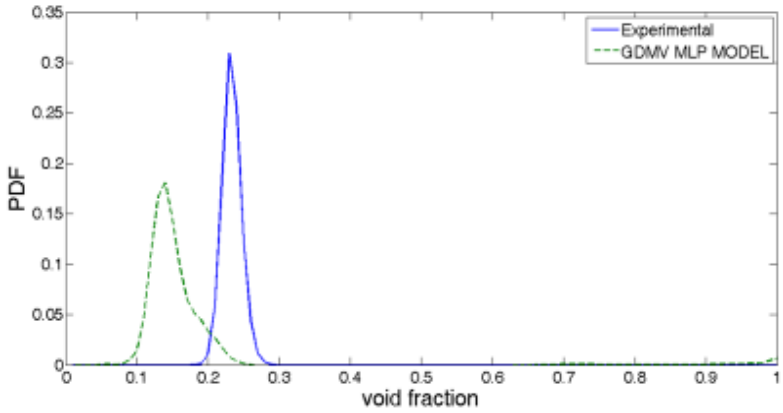


Fig 16g (GDMV model): Comparison between Predicted and Experimental PDF of void fraction after bend for U_{SL}=0.38 m/s U_{SG}=0.061 m/s

Careful inspections of Figures 16a-g show that the model prediction as depicted in Figures 16d, 16e and 16f are good while model predictions as depicted in Figures 16a, 16b, 16c and 16g are not very good. These results indicate that the model can only be used to predict flow behaviour after the bend over a limited range of data.

The lack of generality of the MLP models could be associated to two things; one is the fact that the flow is not fully developed and the other is poor data distribution/partitioning among training, validation and testing. The latter is taken care by data redistribution and the use of different partitioning ratio. The data was redistributed using interleaved indices in as opposed to random indices and the data partitioning ratio changed from 6:2:2 to 4:1:1. Similar results were obtained and the models still showed poor generalisation properties.

4.4 Results of Data Interpolation/Extrapolation before 90^o Bends

The following liquid superficial velocities were used to test the best performing models at various gas superficial velocities.

U_{SL}=0.02ms^{-1}, U_{SL} =0.15 ms^{-1}, U_{SL} =0.30 ms^{-1}, U_{SL} =0.45 ms^{-1}.

Figure 17 gives a plot of predicted average void fraction against gas superficial velocity at liquid superficial velocities not covered by experiment. The plot patterns are in close conformity

ISSN 1512-1232

with expected patterns especially at small and large gas superficial velocities. At intermediate gas superficial velocities the model predictions are not very accurate as can be seen in the crossing of lines.

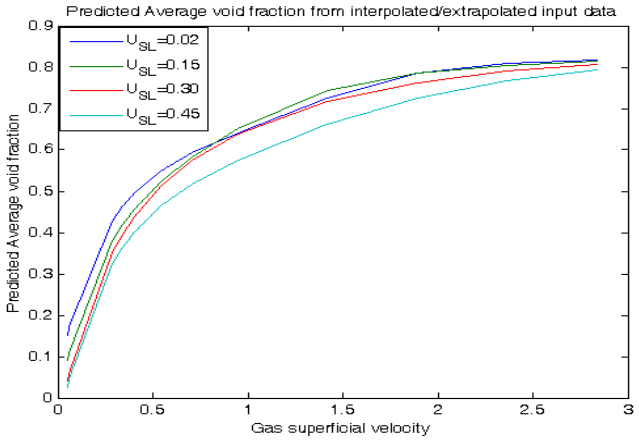


Figure 17: Plot of predicted average void fraction before bend against gas superficial velocities for various liquid superficial velocities different from those covered by experiment.

Figures 18a to 18d are plots of predicted PDF of void fraction before bend against gas superficial velocities for various liquid superficial velocities different from those covered by experiment.

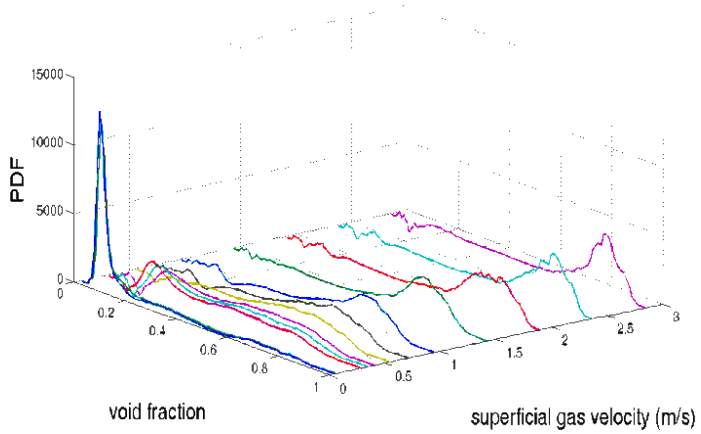
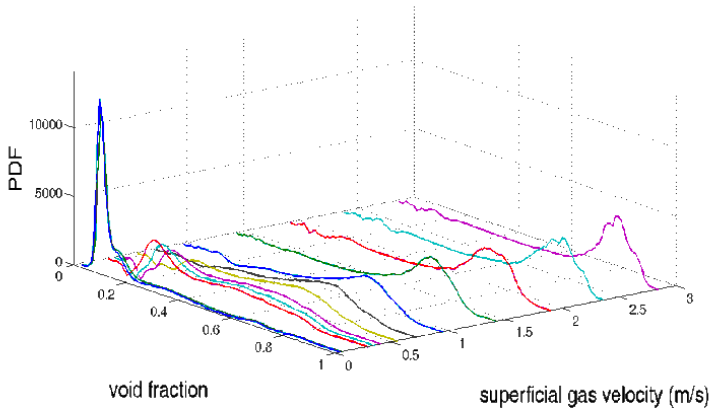


Figure 18a: Plot of predicted PDF of void fraction before bend against gas superficial velocities for various liquid superficial velocities different from those covered by experiment (U_{SL}=0.02 ms^{-1})



ISSN 1512-1232

Figure 18b: Plot of predicted PDF of void fraction before bend against gas superficial velocities for various liquid superficial velocities different from those covered by experiment (U_{SL} =0.15 ms^{-1})

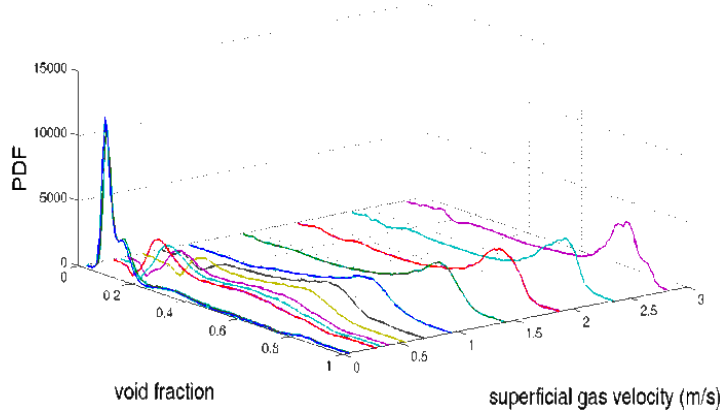


Figure 18c: Plot of predicted PDF of void fraction before bend against gas superficial velocities for various liquid superficial velocities different from those covered by experiment (U_{SL} =0.30 ms^{-1})

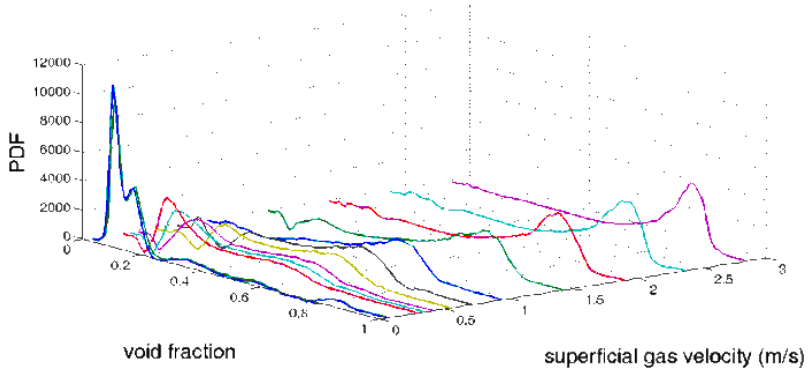


Figure 18d: Plot of predicted PDF of void fraction before bend against gas superficial velocities for various liquid superficial velocities different from those covered by experiment (U_{SL} =0.45 ms^{-1})

Plot patterns of Figures 18a to 18d follow expected patterns of PDF of void fraction as gas and liquid superficial velocities changes. This patterns show the changes in flow characteristics from bubbly flow to churn flow as gas superficial velocities increases.

5. Conclusions

In this work, Multilayer Perceptron Modelling (MLP) models have been developed to predict two-phase average void fraction and PDF of void fraction in 90^o bends. The experiment data were from experiments performed in earlier work in a flow domain of 6 m long vertical pipe with a 67 mm internal diameter connected to a 2 m long flow-line via a vertical 90^o bend (bend radius of 154 mm) at various liquid and gas superficial velocities.

Model optimization was carried out by varying network configuration while monitoring the generalisation error (cross validation error). The developed models were trained using feed-forward approach. A comparison of the results obtained from model predictions and experiments both before and after the vertical 90^o bend has been successfully carried.

The models developed for the prediction of average void fraction and PDF of void fraction before the bend performed very well. However, the AARE confirmed that the best model for prediction of average void fraction before bend is the MLP model trained with Levenberg-Marquardt algorithm in the hidden and output layer with, having three (3) neurons in the hidden

ISSN 1512-1232

layer. The values of AARE and R for this MLP model are 0.0274 and 0.9958 respectively. The AARE also confirmed that the best model for prediction of PDF of void fraction before the bend is the MLP model trained with Levenberg-Marquardt algorithm in the hidden and output layer, having eleven (11) neurons in the hidden layer. The values of AARE and R for this MLP model are 0.7866 and 0.9342 respectively.

The models proposed for the prediction of average void fraction and PDF of void fraction after the bend showed weak generalisation properties. However, the MLP model based on gradient descent algorithm gave better performance for predicting average void fraction after the bend, with AARE and R values of 0.1961 and 0.9156 respectively. Also, the MLP model based on gradient descent algorithm having four (4) neurons in the hidden layer performed best for the prediction of PDF of void fraction after the bend. The values of AARE and R for this MLP model are 1.5740 and 0.4507 respectively.

The results show that MLP models are effective in predicting average void fraction and PDF of void fraction before the bend but are not effective for predicting after the bend scenario because the flow is not fully developed. Similarly the modified empirical model showed good results for prediction before the bend but could not predict average void fraction after the bend.

REFERENCES

[1] M. Abdulkadir, V. Hernandez-Perez, I. S. Lowndes, B. J. Azzopardi, and S. Dzomeku. Experimental study of hydrodynamic behaviour of slug flow in a vertical riser. Chemical Engineering Science, 2013, vol. 106, pp. 60-75.

[2] M. Abdulkadir, D. Zhao, S. Sharaf, L. Abdulkareem, I. S. Lowndes, and B. J. Azzopardi. Interrogating the effect of 90^{o} bends on air–silicone oil flows using advanced instrumentation. Chemical Engineering Science 2011, vol. 66, pp. 2453 - 2467.

[3] M. Abdulkadir. Experimental and computational fluid dynamics (CFD) studies of gas–liquid flow in bends. PhD Thesis, University of Nottingham, 2011.

[4] A. Asghar, R. Masoud, and A. Abdulaziz. CFD and artificial neural network modelling of two-phase flow pressure drop. International Communications in Heat and Mass Transfer, 2009, vol. 36 No. 8, pp. 850-856.

[5] B. J. Azzopardi. Drops in annular two-phase flow. International Journal of Multiphase Flow, 1997, vol. 23, pp. 1-53.

[6] B. J. Azzopardi, L. A. Abdulkareem, S. Sharaf, M. Abdulkadir, V. Hernandez-Perez, and A Ijioma. Using tomography to interrogate gas-liquid flow. In: 28^{th} UIT Heat Transfer Congress, Brescia, Italy, 21 - 23 June 2010.

[7] A. R. Bansal, J. Kanuffman, and R. R. Weitz,. Comparing the modelling performance of regression and neural networks as data quality varies: A Business value approach. Journal of Management Information Systems, 1993, vol. 10, pp. 11 – 32.

[8] E. Bernard and L. Wessels. Extrapolation and interpolation in Neural Network Classifiers. IEEE Control Systems, 1992, vol. 12, pp. 50-53.

[9] Budi, S., Indarto, Deendarlianto and S. W. Thomas. The identification of gas–liquid co-current two phase flow pattern in a horizontal pipe using the power spectral density and the artificial neural network (ANN). Modern Applied Science 2012, vol. 6, pp. 56-67.

[10] G. Carlos. Artificial neural networks for beginners. Http://www.datajobs.com/data-science-repo/Neural-Networks-for-Beginners. 2011, pp. 3-7.

[11] M. B. Carver. Numerical computation of phase separation in two fluid flow, 1984, ASME Paper No. 82-FE-2, Vol. 106/153.

[12] M. B. Carver and M. Salcudean. Three-dimensional numerical modelling of phase distribution of two-fluid flow in elbows and return bends. Numerical Heat Transfer, 1986, vol. 10, pp. 229-251.

[13] I. R. Ellul and R. I. Issa. Prediction of the flow of interspersed gas and liquid phases through pipe bends. Transaction of Institution of Chemical Engineers, 1987, vol. 65, pp. 84-96.

[14] G. C. Gardner and P. H. Neller, P.H. Phase distributions flow of an air-water mixture round bends and past obstructions. Proceedings of Institution of Mechanical Engineers, 1969, vol. 184, pp. 93-101.

ISSN 1512-1232

[15] L. Hernandez, J. E. Julia, S. Chiva, S. Paranjape and M. Ishii. Fast classification of two-phase flow regimes based on conductivity signals and artificial neural networks. Measurement Science and Technology, 2006, vol. 17.
[16] H. Karimi, M. R. Rahimi, J. Lashkarmanesh and S. Azizi. Prediction of flow pattern in vertical gas-liquid two phase flow using pressure fluctuations and Artificial Neural Network. The 8th International Chemical Engineering Congress and Exhibition (IChEC, 2014) Kish, Iran.
[17] H. J. W/ M. Legius, T. J. Narumo and H. E. A. van den Akker. Measurements on wave propagation and bubble and slug velocities in concurrent upward two-phase flow. In: Celata, G.P., and Shah, R.K. (Eds.), Two-Phase Flow Modelling and Experimentation, Rome, 1995, pp. 907-914.
[18] H. B. Mark, T.H. Martin, and B. D. Howard. Neural Network Toolbox (MATLAB User's Guide, R2012b). Release 2012b, pp. 20-30.
[19] A. A. Mohammed and M. D. Birol. Application of Resilient Back-Propagation Neural Networks for generating a universal pressure drop model in pipelines. University of Khartoum Engineering Journal, 2011.
[20] A. Mustafa, E. Moustafa and A. Abdulsalam. Two phase flow regime identification using Artificial Neural Network with non-linear normalization. 2nd International Conference on Fluid Flow, Heat and Mass Transfer, Ottawa, Ontario, Canada, 2015, Paper No. 133.
[21] S. Nasseh, A. Mohebbi, Z. Jeirani and A. Sarrafi. Predicting pressure drop in venturi scrubber with artificial neural networks. Journal of Hazardous Materials, 2007, vol. 143, pp. 144-149.
[22] B. Nirjhar and K. D. Sudip. Prediction of flow regime for air-water flow in circular micro channels using ANN. Journal of Petroleum Science and Engineering, 2013, vol. 10, pp. 813-821.
[23] B. Nirjhar, K. D. Sudip and B. Manindra. Prediction of frictional pressure drop using Artificial Neural Network for air-water flow through U-bends. International Conference on Computational Intelligence: Modelleing techniques and Applications, 2013, vol. 10, pp. 242-252.
[24] B. Nirjhar and K. D. Sudip. Gas-non-Newtonian liquid flow through horizontal pipe–gas holdup and pressure drop prediction using Multilayer Perceptron. American Journal of Fluid Dynamics, 2012, vol. 2,pp. 7-16.
[25] F. Saidj, R. Kibboua, A. Azzi, N. Ababou, B. J. Azzopardi. Experimental investigation of air-water two-phase flow through vertical 90° bend. Experimental Thermal Fluid Science, 2014, vol. 57, pp. 226-234.
[26] K. Soetaert and P. M. J. Herman. A practical guide to ecological modelling, 2009. Sringer ISBN: 978-1-4020-8623-6.
[27] K. B. Tarun and K. D. Sudip. Non-Newtonian pseudo-plastic liquid flow through small diameter piping components. 2007, vol. 55, pp. 156-166
[28] S. Thiele, M. J. Da Silva, U. Hampel, L. Abdulkareem and B. J. Azzopardi. High-resolution oil-gas two-phase flow measurement with a new capacitance wire-mesh tomography. In: 5th International Symposium on Process Tomography, Poland, Zakopane, 25-26 August 2008.
[29] B. M. Wilamowski and H. Yu. Improved computation for Levenberg-Marquardt training. IEEE Transaction on Neural Network, 2010, vol. 21, pp. 930-937.
[30] S. Zhigiang and Z. Hongjian. Neural networks approach for prediction of gas–liquid two-phase flow pattern based on frequency domain analysis of vortex flow meter signals. Measurement Science and Technology, 2008, vol. 19, pp. 1-8.

Number of Figures: 45

Number of Tables: 6

APPENDIX

```
%MODEL THAT USES LEVENBERG MARQUADT ALGORITHM

%This a program written by Paul Onubi Ayegba with the aim of developing a
%numerical model capable of predicting the average void fraction of
silicon
%air misture flowing in a 90 degree bend of 67 mm internal diameter. This

% NOTE THE PROGRAM CAN BE RUN BY SIMPLY CLICKING THE GREEN ARROW ON THE
% TOOL BAR ABOVE, BUT THERE ARE MANY USER VARIABLES E.G. NUMBER OF
```

ISSN 1512-1232

```
% ITERATIONS, ACCEPTABLE ERROR LIMIT ETC WHICH YOU MAY WISH TO ALTER

%NOTE THE PROGRAM MAY RUN FOR AS LONG AS A FEW MINUTES TO AS MUCH AS A
FEW
%HOURS DEPENDING ON THE NUMBER OF ITERATIONS YOU CHOOSE

%intial program seeks to mimic the Artificial Neural Network model which
%employs similar algorithm, that is Levenberg–Marquardt algorithm,
%employing backpropagation technique. The structure of the function
%(sigmoid function/linear function) used is also similar to that used by
%ANN. The model, here, developed gave very good agreement with
experimental
%data, though it required more number of iterations compared to the ANN
model.
%One of the main objectives is to replace this function (sigmoid
function/linear function)
%with some existing mechanistic models (mechanistic models/Linear
%functions) and compare results of models developed.
% p is a matrix of input variable in this case gas and liquid superficial
% velocities. This is a user input and may be changed as the required by
problem.
p=[0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.14 0.14
0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.38 0.38 0.38
0.38 0.38 0.38 0.38 0.38 0.38 0.38 0.38 0.38 0.38;0.047 0.061 0.288 0.344
0.404 0.544 0.709 0.945 1.418 1.891 2.363 2.836 0.047 0.061 0.288 0.344
0.404 0.544 0.709 0.945 1.418 1.891 2.363 2.836 4.73 0.047 0.061 0.288
0.344 0.404 0.544 0.709 0.945 1.418 1.891 2.363 2.836 4.73];
% for optimal behaviour it is necessary to remove constant rows as they
do
% not contribute to model optimization rather they may result in poorly
% scaled matrix whose determinants close to singularity. in this case all
% rows that are constant or rows which have maximum range less or equal
to
% 0.01 is romoved.
[p PS]=removeconstantrows(p,0.01);

%The algorithm can be made more efficient if we perform some
preprocessing
%operations. The operation employed here scale the input and outputs to
%fall within the range [-1 1].
[pn, ps]=mapminmax(p);
x=numel(pn(:,1)); %Number of input parameter(in this case liquid and gas
velocities)
z=numel(pn(1,:)); %Number of input patterns
pn(x+1,:)=ones(1,z);
[pt,pv,pts,trainInd,valInd,testInd] = dividerand(p,0.6,0.2,0.2);
[ptn,pvn,ptsn] = divideind(pn,trainInd,valInd,testInd);
zt=numel(pt(1,:));
ztv=numel(pv(1,:));
zts=numel(pts(1,:));

%OUTPUT
eL_005=[0.127858878 0.142017541 0.357683154 0.380600243 0.407763152
0.471174376 0.500041818 0.568779025 0.659248509 0.698963155 0.728148425
0.759148473];
```

ISSN 1512-1232

```
eL_014=[0.090956304 0.105694861 0.307511372 0.333432331 0.354319881
0.422505167 0.473094667 0.531520114 0.636547444 0.688847949 0.721594917
0.738922184 0.824985470];
eL_038=[0.049916006 0.064464705 0.247366456 0.270821176 0.290103160
0.368996198 0.418661835 0.478407434 0.558563925 0.615894456 0.662988627
0.709819272 0.776042882];
t=[eL_005 eL_014 eL_038];
[t TS]=removeconstantrows(t);

%The algorithm can be made more efficient if we perform some
preprocessing
%operations. The operation employed here scale the input and outputs to
%fall within the range [-1 1].

[tn, ts]=mapminmax(t);
[tt,tv,tts] = divideind(t,trainInd,valInd,testInd);
[ttn,tvn,ttsn] = divideind(tn,trainInd,valInd,testInd);
% s is the number of neuron (simple processing units), This is a user
% variable, the number of neurons can between 1 and infinity until the
best
% performance is obtained. For the purpose of understanding this may be
% taken as that configuration (number of neurons in the hidden layer in
% this case) that gives the smallest value of minimum mean square error
% 'mse'.
s=3;
l=0;
%The program is developed for a two layer network(one hidden layer of
%variable number of neurons and one output layer whose number of neurons
%is determined by the output matrix supplied. Also the neurons in the
%hidden layer is designed to have a 'sigmoid transfer function' that is
% a function of the form 'Y1=(1./(1+exp(sum(-w.*pn)+b1))' where w and b1
are
%weight and bias matrices and pn is the normalized of preprocessed inputs
%and the output layer is a linear function of the form Y2= sum(v*Y1)+b2.
%This configuration has proven very effective for most problems of
function
%fitting. The summation of output from the hidden layer is the input to
the
% output layer Thus the function takes the following basic form for each
%neuron. Note the value of the weights and biases in each neuron are not
%usually the same.
y=numel(t(:,1)); %Number of output parameter (in this case grouped
frequencies void fraction)
yt=numel(tt);
yv=numel(tv);
yts=numel(tts);
bin=0.01:0.01:1;
bin=bin';
syms v w p1 L % helps us in finding the the partial derivative of the
function with respect to the symbolic variables
pr=x;  % pr number of non-constant input variables p
ws=rands(s,pr+1); % Input weight matrix, normalized random values are
used
Ls=1;  % Layer weight matrix, normalized random values are used
vs=rands(y,s+1);  % Layer weight matrix, normalized random values are
used
wsR(:,:,1)=ws;
```

ISSN 1512-1232

```
vsR(:,:,1)=vs;
f=v*(1/(1+exp(-L*(w*p1))));
ftn1=(1./(1+exp(-Ls*(ws*ptn))));
ftn1(s+1,:)=ones(1,zt);
ftn=vs*ftn1;
fvn1=(1./(1+exp(-Ls*(ws*pvn))));
fvn1(s+1,:)=ones(1,ztv);
fvn=vs*fvn1;
ftsn1=(1./(1+exp(-Ls*(ws*ptsn))));
ftsn1(s+1,:)=ones(1,zts);
ftsn=vs*ftsn1;

%fw=diff(f,w);% Derivative of the function with respect to layer bias w
%Matlab generates fw=v*((Ls*p1)./(exp(Ls*(w*p1+b1))*(1/exp(Ls*(w*p1+b1))
+ 1)^2)) this must
%be rearranged for ease of matrix operation

%The Levenberg–Marquardt algorithm will be used in a backpropagation
%fashion in determining the weights and bias and it takes the form
%w(k+1)=w(k)–(JT(k)*J(k)+ mu*I)^-1 *J(k)*e(k)
%The choice of Levenberg–Marquardt algorithm will be given in the
%literature. If mu is small the algorithm becomes Gauss Newton alorithm
and
%if mu is very large the algoritm becomes steepest descent alorithm.
mu=0.001; %initial value of mu.
r=(ttn-ftn); %error residual, difference between model output and
expected output
rv=(tvn-fvn);
rts=(ttsn-ftsn);
d=r.^2;
dv=rv.^2;
dts=rts.^2;
%mean square error is used as the performance function and the target of
%0.0000001 is used, but it can be varied by user.
mse(1)=sum(sum(d),2)/yt;
msev(1)=sum(sum(dv),2)/yv;
msets(1)=sum(sum(dts),2)/yts;

for k=1:20000;  % k is the number of iterations, this can be varied by
user
    if ((mse(k)>=0.0000001) && (mu<=1*10^10));
        %fw=v*((L*p1)./(exp(L*(w*p1)).*(1./exp(L*(w*p1)) + 1)^2));
        aw=(Ls./(exp(Ls*(ws*ptn)).*(1./exp(Ls*(ws*ptn)) + 1).^2));
        m=0;
        for Q=1:x+1;
            for q=1:s;
                m=m+1;
                fwa(q,:,Q)=ptn(Q,:).*aw(q,:);
                fwb=vs(:,q)*fwa(q,:,Q);
                fwb=fwb';
                for n=1:y;
                    fwc(n*zt-zt+1:n*zt,m)=fwb(:,n);
                end
            end
        end
        for n=1:y;
            r1=r';
```

ISSN 1512-1232

```
            R(n*zt-zt+1:n*zt,1)=r1(:,n);
        end
        fw=fwc;
        fv1=1./(1./exp(Ls*ws*ptn) + 1);
        fv1(s+1,:)=ones(1,zt);
        fvb=0*eye(zt*y,y);
        for Q=1:s+1;
             fva=fv1(Q,:)';
             for q=1:y;
             fvb((q-1)*zt+1:q*zt,q,Q)=fva;
             end
         end
         fv=fvb(:,:);
         J=[fw fv];
         JT=J';
         H=JT*J;
         HD=(JT*J+mu*diag(diag(H)));
         T(k)=rcond(HD);
         G(k)=norm(JT);
         if (T(k)>=0.00000000000001) && (G(k)>=1e-10);
                 HinvJT=HD\JT;
                 a=HinvJT*R;
                 n=0;
                 for q=1:x+1
                     n=n+1;
                 ws(:,q)=ws(:,q)+a(n*s-s+1:n*s,1);
                 end
                 for q=1:s+1
                 vs(:,q)=vs(:,q)+a(n*s+(q-1)*y+1:n*s+q*y,1);
                 end
                 ftn1=(1./(1+exp(-Ls*(ws*ptn))));
                 ftn1(s+1,:)=ones(1,zt);
                 ftn=vs*ftn1;
                 r1=(ttn-ftn);
                 d=r1.^2;
                 mse(k+1)=sum(sum(d),2)/yt;
                 if  mse(k+1)>=mse(k);
                     for j=1:5;
                         n=0;
                         for q=1:x+1
                             n=n+1;
                             ws(:,q)=ws(:,q)-a(n*s-s+1:n*s,1);
                         end
                         for q=1:s+1
                             vs(:,q)=vs(:,q)-a(n*s+(q-
1)*y+1:n*s+q*y,1);
                         end
                         mu=mu*10;
                         HD=(JT*J+mu*diag(diag(H)));
                         T(k)=rcond(HD);
                         G(k)=norm(JT);
                         if (T(k)>=0.00000000000001) && (G(k)>=1e-
10);
```

ISSN 1512-1232

```
                            HinvJT=HD\JT;
                            a=HinvJT*R;
                            n=0;
                            for q=1:x+1
                                n=n+1;
                                ws(:,q)=ws(:,q)+a(n*s-s+1:n*s,1);
                            end
                            for q=1:s+1
                                vs(:,q)=vs(:,q)+a(n*s+(q-
1)*y+1:n*s+q*y,1);
                            end
                            ftn1=(1./(1+exp(-Ls*(ws*ptn))));
                            ftn1(s+1,:)=ones(1,zt);
                            ftn=vs*ftn1;
                            r=(ttn-ftn);
                            d=r.^2;
                            mse(k+1)=sum(sum(d),2)/yt;
                            if ((j>=5) || (mse(k+1)<mse(k)))
                            break
                            end
                        else
                        break
                        end
                    end
                else
                    mu=mu/10;
                    r=r1;
                end
        else
        break
        end
        wsR(:,:,k+1)=ws;
        vsR(:,:,k+1)=vs;
        %Test
            ftsn1=(1./(1+exp(-Ls*(ws*ptsn))));
            ftsn1(s+1,:)=ones(1,zts);
            ftsn=vs*ftsn1;
            rts=(ttsn-ftsn);
            dts=rts.^2;
            msets(k+1)=sum(sum(dts),2)/yts;

        %Validation
            fvn1=(1./(1+exp(-Ls*(ws*pvn))));
            fvn1(s+1,:)=ones(1,ztv);
            fvn=vs*fvn1;
            rv=(tvn-fvn);
            dv=rv.^2;
            msev(k+1)=sum(sum(dv),2)/yv;
            if msev(k+1)>=msev(k);
                l=l+1;
                if l==5;
                    break
```

ISSN 1512-1232

```
                end
            else
                l=0;
            end
    else
        break
    end

end
if l>=5;
    ws=wsR(:,:,k-4);
    vs=vsR(:,:,k-4);
end
ftn1=(1./(1+exp(-Ls*(ws*ptn))));
ftn1(s+1,:)=ones(1,zt);
ftn=vs*ftn1;
fvn1=(1./(1+exp(-Ls*(ws*pvn))));
fvn1(s+1,:)=ones(1,ztv);
fvn=vs*fvn1;
ftsn1=(1./(1+exp(-Ls*(ws*ptsn))));
ftsn1(s+1,:)=ones(1,zts);
ftsn=vs*ftsn1;
t=removeconstantrows('reverse',t,TS);
tt_exp=removeconstantrows('reverse',tt,TS);
tv_exp=removeconstantrows('reverse',tv,TS);
tts_exp=removeconstantrows('reverse',tts,TS);
t_exp=t;
fft = mapminmax('reverse',ftn,ts);
ffv = mapminmax('reverse',fvn,ts);
ffts = mapminmax('reverse',ftsn,ts);
tt_model=removeconstantrows('reverse',fft,TS);
tv_model=removeconstantrows('reverse',ffv,TS);
tts_model=removeconstantrows('reverse',ffts,TS);
p=removeconstantrows('reverse',p,PS);
pt=removeconstantrows('reverse',pt,PS);
pv=removeconstantrows('reverse',pv,PS);
pts=removeconstantrows('reverse',pts,PS);
mse_minimum_train=min(mse)
mse_minimum_val=min(msev)
mse_minimum_test=min(msets)
number_iteration=k

% PERFORMANCE ANALYSIS
%Cross Correlation coefficient
%R=C(i,j)/(C(i,i)C(j,j))^1/2
%Training
Avt_exp=mean(mean(tt_exp,2));
Avt_model=mean(mean(tt_model,2));
Nmt=(sum(sum((tt_model-Avt_model).*(tt_exp-Avt_exp)),2));
Dnt=((sum(sum((tt_model-Avt_model).^2),2)).*(sum(sum((tt_exp-
Avt_exp).^2),2)));
Rtrain=Nmt/sqrt(Dnt);
```

ISSN 1512-1232

```
%Validation
Avv_exp=mean(mean(tv_exp,2));
Avv_model=mean(mean(tv_model,2));
Nmv=(sum(sum((tv_model-Avv_model).*(tv_exp-Avv_exp)),2));
Dnv=((sum(sum((tv_model-Avv_model).^2),2)).*(sum(sum((tv_exp-
Avv_exp).^2),2)));
Rvalidation=Nmv/sqrt(Dnv);
%Testing
Avts_exp=mean(mean(tts_exp,2));
Avts_model=mean(mean(tts_model,2));
Nmts=(sum(sum((tts_model-Avts_model).*(tts_exp-Avts_exp)),2));
Dnts=((sum(sum((tts_model-Avts_model).^2),2)).*(sum(sum((tts_exp-
Avts_exp).^2),2)));
Rtest=Nmts/sqrt(Dnts);

%Average absolute relative error (AARE)
%AARE=(1/N)*sum(absolute(model output-exp value)/exp value));
AT=abs((tt_model-tt_exp)./tt_exp);
AAREtrain=(1/yt)*sum(sum(AT),2);
AV=abs((tv_model-tv_exp)./tv_exp);
AAREvalidation=(1/yv)*sum(sum(AV),2);
ATS=abs((tts_model-tts_exp)./tts_exp);
AAREtest=(1/yts)*sum(sum(ATS),2);

%PLOTS
plot(tt_exp,tt_model,'*')
xlabel('experimantal average void fraction')
ylabel('MLP model average void fraction')
legend('Training')
title('Gradient Descent with adaptive step and momentum term')
figure
plot(tv_exp,tv_model,'*')
xlabel('experimantal average void fraction')
ylabel('MLP model average void fraction')
legend('Validation')
title('Gradient Descent with adaptive step and momentum term')
figure
plot(tts_exp,tts_model,'*')
xlabel('Void fraction (Experimental)')
ylabel('Void fraction (Predicted)')
hold on
x1=[0,1];
y1=[0,1];
x2=[0.05*1,1];
y2=[0,0.95*1];
x3=[0,1];
y3=[0.05*1,1.05*1];
plot(x1,y1,x2,y2,'--',x3,y3,'--')
figure
u=1:k+1;
plot(u,mse,u,msev,u,msets)
xlabel('Number of iterations')
```

ISSN 1512-1232

```
ylabel('Mean Square Error MSE')
legend('Training','Validation','Testing')
title('Gradient Descent with adaptive step and momentum term
performance plot')
%MODEL THAT USES GRADIENT DESCENT ALGORITHM WITH VARIABLE STEP
%SIZE AND MOMENTUM TERM

%This a program written by the autor with the aim of developing a
%numerical model capable of predicting the PDF of void fraction of
silicon
%air mixture flowing in a 90 degree bend of 67 mm internal
diameter. This

% NOTE THE PROGRAM CAN BE RUN BY SIMPLY CLICKING THE GREEN ARROW
ON THE
% TOOL BAR ABOVE, BUT THERE ARE MANY USER VARIABLES E.G. NUMBER OF
% ITERATIONS, ACCEPTABLE ERROR LIMIT ETC WHICH YOU MAY WISH TO
ALTER

%NOTE THE PROGRAM MAY RUN FOR AS LONG AS A FEW MINUTES TO AS MUCH
AS A FEW
%HOURS DEPENDING ON THE NUMBER OF ITERATIONS AND MODEL TOPOGRAPHY

%intial program seeks to mimic the Artificial Neural Network model
which
%employs similar algorithm, that is gradient descent algorithm
with variable step size and momentum term,
%employing backpropagation technique. The structure of the
function
%(sigmoid function/linear function) used is also similar to that
used by
%ANN.

%INPUTS
% p is a matrix of input variable in this case gas and liquid
superficial
% velocities. This is a user input and may be changed as the
required by problem.
p=[0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.38 0.38
0.38 0.38 0.38 0.38 0.38 0.38 0.38 0.38;0.047 0.061 0.288 0.344
0.404 0.544 0.709 0.945 1.418 1.891 2.363 2.836 0.047 0.061 0.288
0.344 0.404 0.544 0.709 1.891 2.363 2.836 4.73 0.061 0.288 0.344
0.404 0.544 0.709 0.945 1.891 2.836 4.73];
% for optimal behaviour it is necessary to remove constant rows as
they do
% not contribute to model optimization rather they may result in
poorly
% scaled matrix whose determinants close to singularity. in this
case all
% rows that are constant or rows which have maximum range less or
equal to
```

ISSN 1512-1232

```
% 0.01 is romoved.
[p PS]=removeconstantrows(p,0.01);

%The algorithm can be made more efficient if we perform some
preprocessing
%operations. The operation employed here scale the input and
outputs to
%fall within the range [-1 1].
[pn, ps]=mapminmax(p);
x=numel(pn(:,1)); %Number of input parameter(in this case liquid
and gas velocities)
z=numel(pn(1,:)); %Number of input patterns
pn(x+1,:)=ones(1,z);
[pt,pv,pts,trainInd,valInd,testInd] = dividerand(p,6,2,2);
[ptn,pvn,ptsn] = divideind(pn,trainInd,valInd,testInd);
zt=numel(ptn(1,:));
ztv=numel(pvn(1,:));
zts=numel(ptsn(1,:));

%OUTPUTS
f1=[0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   3
15  13  43  150 212 1067    3003    3552    4672    6750    7462
6076    5604    6322    5322    4764    3088    1293    672 277 80
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0]';
f2=[0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   15
334 746 1257    2859    3838    4518    4469    4182    4364
5170    5137    4740    4176    3383    3261    3191    2238
1121    821 353 111 58  60  38  0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0]';
f3=[0   0   0   0   0   0   0   0   0   0   0   0   0   3   20
29  59  168 187 252 390 603 721 779 789 819 936 1119    1107
1029    1186    1309    1285    1176    1277    1364    1469
1490    1521    1493    1460    1461    1440    1547    1507
1370    1393    1299    1367    1254    1205    1236    1181
1185    1251    1241    1322    1371    1345    1195    946 1141
1286    1107    1334    1459    1017    716 763 709 558 144 50  0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0]';
f4=[0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   1   42
102 76  138 256 262 329 437 576 545 957 1150    1398    1670
1957    3668    46876]';
```

ISSN 1512-1232

```
f5=[0   0   0   0   0   0   0   0   0   3   8   2   3   16  24  24
67  99  123 175 184 194 269 347 452 433 521 654 703 614 597 632
620 584 656 673 857 960 995 924 1036    1117    1188    1109
1158    1166    1142    1131    1235    1298    1427    1609
1451    1476    1384    1230    1258    1298    1136    1122
1235    1232    1261    1292    1482    1685    2018    2026
1882    2022    1783    1298    953 869 1084    600 275 59  0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0]';
f6=[0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   8   34  39  64  49  46  48  42  33  48  63  83  102 96  91  78
111 111 102 115 123 139 148 240 255 217 255 226 235 276 306 335
417 497 486 518 549 692 807 861 837 836 780 847 1230    1235
1213    1368    1468    1537    1465    1447    1479    1377
1684    2133    2075    2200    2420    2278    2896    3645
4344    3898    2695    2053    1166    560 177 93  56  3   0   0
0   0   0   0   0   0   0   0   0]';
f7=[0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   3   7
7   6   5   4   3   10  7   8   9   9   14  18  48  54  59  74  59
41  51  98  166 197 194 147 159 201 257 394 471 582 676 569 577
678 728 749 845 982 1143    1360    1415    1513    1520    1630
2103    2442    2564    3021    3774    4375    6013    7180
4548    2257    1655    1811    747 195 8   0   0   0   0   0   0
0   0]';
f8=[0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   1   6   11
4   10  5   14  19  18  15  18  17  20  18  13  24  25  34  54  54
81  168 217 284 252 237 266 334 404 556 663 741 759 748 838 1093
1300    1420    1394    1496    1776    1757    1760    1732
1998    2227    1978    2035    2629    3476    3863    3948
4096    5023    4065    1903    1228    644 376 157 92  46  0   0
0   0   0   0   0   0]';
f9=[0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   9   6   41  74  152 228 460 2812
56658]';
f10=[0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   1   3   4   3   12  17
15  28  49  68  88  106 156 212 289 331 455 564 665 865 1106
1444    1895    2029    2551    3057    4106    4885    4811
5702    5600    3998    3563    4433    3173    2661    1364
131 0]';
f11=[0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   1   0   1   1   2   3   5   1   11  13  15  8   16  22
```

ISSN 1512-1232

```
25  41  51  73  86  118 152 217 297 336 446 523 719 937 1099
1359    1628    1831    2090    2778    3721    4205    5075
5869    6370    6138    5947    4663    1707    879 580 381 0]';
f12=[0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
1   0   0   5   11  3   2   8   6   9   14  21  20  28  34  35  56
84  89  103 118 157 207 226 275 294 423 534 622 693 878 1166
1431    1760    2051    2470    3095    3878    5403    5858
5737    6468    6894    5381    2453    762 451 225 1   0]';

f40=[0  0   0   0   0   0   99  408 1304    3668    7049    10363
12165   10140   6672    3956    2330    1236    550 363 130 7   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0]';
f41=[0  0   0   0   7   51  41  81  286 828 2716    6602    9819
10594   8433    5791    3840    3053    2631    1986    1584
1024    512 315 147 90  9   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0]';
f42=[0  0   0   0   0   0   0   0   1   27  53  117 188 304 371
580 995 1258    1472    1653    1675    1674    1906    1830
1955    2076    2081    1999    1913    1865    1746    1756
1720    1539    1569    1601    1505    1666    1708    1705
1721    1728    1652    1558    1339    1276    1145    1060
904 695 578 565 636 517 466 437 355 365 217 175 165 60  53  46  52
47  43  41  36  0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0]';
f43=[0  0   0   0   0   0   0   0   1   6   20  29  60  73  176
298 491 576 633 703 901 1174    1266    1513    1729    1808
1753    1753    1706    1747    1646    1635    1646    1502
1441    1460    1550    1590    1533    1553    1578    1696
1701    1702    1681    1689    1511    1724    1736    1605
1486    1311    1166    952 963 777 638 419 445 325 275 161 137
171 162 146 98  71  47  23  17  14  41  0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0]';
f44=[0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   2   12  40  51  51  95  148 147 161 179 188 251 307 411
417 611 737 744 810 991 1079    1373    1454    1411    1449
1431    1515    1475    1596    1682    1702    1487    1554
1814    1810    1737    1746    1731    1733    1637    1818
1841    1778    1656    1592    1722    1836    1727    1733
1369    1235    840 731 743 456 482 427 231 160 89  35  46  91  33
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0]';
```

ISSN 1512-1232

```
f45=[0  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   3   6   11  15  38  36  47  60  79  155 243 318 301
303 301 408 479 488 540 544 590 634 657 706 766 802 802 868 889
911 916 1046    1047    1156    1149    1185    1210    1152
1291    1494    1545    1543    1720    1926    1951    2069
1983    2199    2337    2358    2387    2489    2244    1931
1842    1485    1115    967 603 550 439 329 286 261 131 64  38  2
0   0   0   0   0   0   0   0   0   0   0   0   0]';
f46=[0  0   0   0   0   0   0   0   0   0   0   0   0   0   0
2   8   15  19  22  32  36  52  49  59  84  110 124 164 167 172
172 183 178 209 212 229 269 306 327 291 400 384 465 478 579 618
696 661 703 694 733 790 810 912 969 1005    1128    1274    1341
1387    1537    1627    1783    1911    2134    2194    2328
2617    2914    2782    2615    2824    2627    2489    1988
1627    1349    1037    773 509 417 320 239 141 43  52  45  0   0
0   0   0   0   0   0   0   0   0]';
f49=[0  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   2   0   3   6   7   6   11  14  17  23  20  36  44
54  79  121 134 152 186 217 227 279 321 386 474 533 615 653 779
943 1032    1171    1311    1484    1615    1773    2167    2350
2578    2856    3069    2975    3406    3519    3984    3630
4015    3599    2956    2291    1191    686 229 115 96  0   0   0
0   0   0   0   0]';
f50=[0  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   10  13  19  36  57
89  124 195 292 478 690 972 1223    1564    2072    2369    2877
3442    5004    11317   27597]';
f51=[0  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   2   1   0   0   2   0   2   1   2   2   7   9   8   15
18  9   24  34  35  54  44  53  49  94  121 123 150 189 241 266
380 416 589 654 793 898 1004    1153    1408    1579    1877
2056    2323    2616    2960    3199    3502    3661    3791
4241    4272    3908    3698    3267    1990    1257    792 356
237 8   0   0   0   0]';
f52=[0  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   1   0   0   6   5   9   11  17  23  33  47
72  65  91  125 142 201 270 341 456 624 814 1095    1332    1596
1972    2390    2948    3370    4127    4990    5991    6186
6465    5534    4590    2540    1178    459 278 46  0   0   0]';

f67=[0  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   2   57  653 3532    11098   18704   15447   7425    2673
669 165 15  0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

ISSN 1512-1232

```
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0]';
f68=[0  0   0   0   0   0   0   0   0   0   0   0   2   13  15  37
82  134 208 315 483 707 963 1195    1325    1427    1629    1790
2150    2319    2475    2621    2693    2654    2471    2421
2340    2088    2076    1899    1660    1322    1324    1359
1453    1492    1365    1558    1533    1509    1374    1293
1169    1078    793 559 398 249 172 134 86  21  7   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]';
f69=[0  0   0   0   0   0   0   0   0   0   0   0   0   7   16  26
69  141 252 356 478 634 826 1094    1173    1215    1295    1335
1452    1595    1635    1654    1798    1879    2042    1884
1853    1855    1828    1753    1650    1637    1729    1746
1736    1621    1578    1633    1641    1599    1680    1734
1592    1492    1362    1131    1002    873 836 663 465 326 223
153 60  61  46  26  0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0]';
f70=[0  0   0   0   0   0   0   0   0   0   0   0   1   6   25  37
55  74  136 225 297 480 752 889 1021    1192    1266    1241
1264    1285    1377    1424    1509    1568    1494    1484
1498    1490    1509    1422    1461    1662    1714    1702
1624    1767    1575    1679    1628    1621    1599    1624
1656    1538    1609    1555    1541    1554    1371    1282
1006    701 634 483 339 208 148 65  50  12  10  1   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0]';
f71=[0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   6
8   13  45  92  144 241 343 446 578 633 766 810 871 880 965 917
984 996 907 910 986 1059    1017    926 989 963 967 991 1079
1133    1174    1139    1251    1287    1239    1246    1388
1588    1632    1739    1820    1856    1920    1850    1858
1858    1943    1946    1874    1596    1446    1378    1212
930 571 499 251 146 63  45  15  3   3   5   4   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0]';
f72=[0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   2   9   7   17  45  102 194 259 417 543 670 781 803 871 951
868 819 828 868 821 804 771 828 790 768 826 748 693 746 883 963
1032    955 1087    1084    1080    1237    1216    1304    1325
1518    1409    1484    1546    1640    1801    1887    2090
2155    2169    2373    2122    2090    1842    1470    1276
961 580 435 253 135 65  53  39  30  2   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0]';
f73=[0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   2   5   5   13  20  54  127 168 231 306 410 582 649
699 728 746 690 705 766 697 672 667 672 696 700 676 682 626 699
668 717 672 748 926 941 1039    1128    1144    1311    1397
1459    1520    1622    1726    1994    2104    2114    2281
2227    2265    2362    2196    2085    1931    1715    1425
1461    1197    1050    614 323 216 121 35  7   6   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0]';
```

ISSN 1512-1232

```
f75=[0  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   1   17  53  137
230 304 488 660 930 1279    1665    2010    2508    2793    3100
3369    3786    4327    5319    10986   16478]';
f77=[0  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   1   2   3   0   3   0
0   3   2   1   2   5   2   8   10  6   9   8   18  29  23  25  29
51  63  70  95  110 164 138 207 270 339 393 486 510 720 808 921
1053    1201    1287    1381    1507    1582    1842    1957
2113    2324    2493    2676    2801    2908    3122    3369
3452    3490    3475    3254    2557    2192    1475    930 393 65
7   0   0   0   0   0   0   0]';
f78=[0  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   2
3   2   8   5   5   5   7   11  9   15  27  30  42  55  80  85
136 172 253 286 363 467 610 794 980 1160    1522    1875    2207
2655    2958    3306    3594    3851    4050    4354    4553
4680    4731    4109    3197    1940    868 336 39  1   0   0   0
0]';

% t is a matrix of experimental outputs, in this case the
experimental out
% is average void fraction this is a user input and may be changed
as
% required.
t=[f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11 f12 f40 f41 f42 f43 f44 f45
f46 f49 f50 f51 f52 f67 f68 f69 f70 f71 f72 f73 f75 f77 f78];
[t TS]=removeconstantrows(t);

%The algorithm can be made more efficient if we perform some
preprocessing
%operations. The operation employed here scale the input and
outputs to
%fall within the range [-1 1].

[tn, ts]=mapminmax(t);
[tt,tv,tts] = divideind(t,trainInd,valInd,testInd);
[ttn,tvn,ttsn] = divideind(tn,trainInd,valInd,testInd);
% s is the number of neuron (simple processing units), This is a
user
% variable, the number of neurons can between 1 and infinity until
the best
% performance is obtained. For the purpose of understanding this
may be
% taken as that configuration (number of neurons in the hidden
layer in
% this case) that gives the smallest value of minimum mean square
error
```

ISSN 1512-1232

```
% 'mse'.
s=4;
l=0;
%The program is developed for a two layer network(one hidden layer
of
%variable number of neurons and one output layer whose number of
neurons
%is determined by the output matrix supplied. Also the neurons in
the
%hidden layer is designed to have a 'sigmoid transfer function'
that is
% a function of the form 'Y1=(1./(1+exp(sum(-w.*pn)+b1))' where w
and b1 are
%weight and bias matrices and pn is the normalized of preprocessed
inputs
%and the output layer is a linear function of the form Y2=
sum(v*Y1)+b2.
%This configuration has proven very effective for most problems of
function
%fitting. The summation of output from the hidden layer is the
input to the
% output layer Thus the function takes the following basic form
for each
%neuron. Note the value of the weights and biases in each neuron
are not
%usually the same.
y=numel(t(:,1)); %Number of output parameter (in this case grouped
frequencies void fraction)
yt=numel(tt);
yv=numel(tv);
yts=numel(tts);
syms v w p1 L % helps us in finding the the partial derivative of
the function with respect to the symbolic variables
pr=x;  % pr number of non-constant input variables p
ws=rands(s,pr+1); % Input weight matrix, normalized random values
are used
Ls=1;  % Layer weight matrix, normalized random values are used
vs=rands(y,s+1);  % Layer weight matrix, normalized random values
are used
wsR(:,:,1)=ws;
vsR(:,:,1)=vs;
f=v*(1/(1+exp(-L*(w*p1))));
ftn1=(1./(1+exp(-Ls*(ws*ptn))));
ftn1(s+1,:)=ones(1,zt);
ftn=vs*ftn1;
fvn1=(1./(1+exp(-Ls*(ws*pvn))));
fvn1(s+1,:)=ones(1,ztv);
fvn=vs*fvn1;
ftsn1=(1./(1+exp(-Ls*(ws*ptsn))));
ftsn1(s+1,:)=ones(1,zts);
ftsn=vs*ftsn1;
```

ISSN 1512-1232

```
%The gradient descent algorithm with momentum term and variable
step size
%will be used in a backpropagation fashion in determining the
weights and
%bias and it takes the form w(k+1)=w(k)+Lr*g(k)*e(k)+ mc*w(k-1)

mc=0.9;
Lr=0.01; %initial value of Lr.
r=(ttn-ftn); %error residual, difference between model output and
expected output
rv=(tvn-fvn);
rts=(ttsn-ftsn);
d=r.^2;
dv=rv.^2;
dts=rts.^2;
%mean square error is used as the performance function and the
target of
%0.0000001 is used, but it can be varied by user.
mse(1)=sum(sum(d),2)/yt;
msev(1)=sum(sum(dv),2)/yv;
msets(1)=sum(sum(dts),2)/yts;

for k=1;  % k is the number of iterations, this can be varied by
user
    if ((mse(k)>=0.0000001) && (Lr<=1*10^10));
        %fw=v*((L*p1)./(exp(L*(w*p1)).*(1./exp(L*(w*p1)) + 1)^2));
        aw=(Ls./(exp(Ls*(ws*ptn)).*(1./exp(Ls*(ws*ptn)) + 1).^2));
        vr=vs'*r;
        m=0;
        for Q=1:x+1;
            for q=1:s;
                m=m+1;
                fwa(q,:,Q)=ptn(Q,:).*aw(q,:);
                fwb(q,Q)=fwa(q,:,Q)*vr(q,:)';
            end
        end
        fw=fwb;
        fv=1./(1./exp(Ls*ws*ptn) + 1);
        fv(s+1,:)=ones(1,zt);
        J2=fv;
        JT2=J2';
        G1(k)=norm(fw);
        G2(k)=norm(JT2);
            if (G1(k)>=1e-10) && (G2(k)>=1e-10);
                a1=Lr*fw;
                a2=Lr*r*JT2;
                ws=ws+a1;
                vs=vs+a2;
                ftn1=(1./(1+exp(-Ls*(ws*ptn))));
                ftn1(s+1,:)=ones(1,zt);
                ftn=vs*ftn1;
                r1=(ttn-ftn);
```

ISSN 1512-1232

```
            d=r1.^2;
            mse(k+1)=sum(sum(d),2)/yt;
            if  mse(k+1)>=1.04*mse(k);
                    ws=ws-a1;
                    vs=vs-a2;
                    Lr=0.7*Lr;
                    a1=Lr*fw;
                    a2=Lr*r*JT2;
                    ws=ws+a1;
                    vs=vs+a2;
                    ftn1=(1./(1+exp(-Ls*(ws*ptn))));
                    ftn1(s+1,:)=ones(1,zt);
                    ftn=vs*ftn1;
                    r=(ttn-ftn);
                    d=r.^2;
                    mse(k+1)=sum(sum(d),2)/yt;
            else
                Lr=Lr*1.05;
                r=r1;
            end
        else
            break
        end
    wsR(:,:,k+1)=ws;
    vsR(:,:,k+1)=vs;
    %Test
        ftsn1=(1./(1+exp(-Ls*(ws*ptsn))));
        ftsn1(s+1,:)=ones(1,zts);
        ftsn=vs*ftsn1;
        rts=(ttsn-ftsn);
        dts=rts.^2;
        msets(k+1)=sum(sum(dts),2)/yts;

        %Validation
        fvn1=(1./(1+exp(-Ls*(ws*pvn))));
        fvn1(s+1,:)=ones(1,ztv);
        fvn=vs*fvn1;
        rv=(tvn-fvn);
        dv=rv.^2;
        msev(k+1)=sum(sum(dv),2)/yv;
        if msev(k+1)>=msev(k);
            l=l+1;
            if l==5;
                break
            end
        else
            l=0;
        end
else
    break
end
```

ISSN 1512-1232

```
end

for k=2:100000;  % k is the number of iterations, this can be
varied by user
    if ((mse(k)>=0.0000001) && (Lr<=1*10^10));
        %fw=v*((L*p1)./(exp(L*(w*p1)).*(1./exp(L*(w*p1)) + 1)^2));
        aw=(Ls./(exp(Ls*(ws*ptn)).*(1./exp(Ls*(ws*ptn)) + 1).^2));
        vr=vs'*r;
        m=0;
        for Q=1:x+1;
            for q=1:s;
                m=m+1;
                fwa(q,:,Q)=ptn(Q,:).*aw(q,:);
                fwb(q,Q)=fwa(q,:,Q)*vr(q,:)';
            end
        end
        fw=fwb;
        fv=1./(1./exp(Ls*ws*ptn) + 1);
        fv(s+1,:)=ones(1,zt);
        J2=fv;
        JT2=J2';
        G1(k)=norm(fw);
        G2(k)=norm(JT2);
            if (G1(k)>=1e-10) && (G2(k)>=1e-10);
                a1R=a1;
                a2R=a2;
                a1=Lr*fw;
                a2=Lr*r*JT2;
                ws=ws+a1+mc*a1R;
                vs=vs+a2+mc*a2R;
                ftn1=(1./(1+exp(-Ls*(ws*ptn))));
                ftn1(s+1,:)=ones(1,zt);
                ftn=vs*ftn1;
                r1=(ttn-ftn);
                d=r1.^2;
                mse(k+1)=sum(sum(d),2)/yt;
                if  mse(k+1)>=1.04*mse(k);
                        ws=ws-a1-mc*a1R;
                        vs=vs-a2-mc*a2R;
                        Lr=0.7*Lr;
                        a1=Lr*fw;
                        a2=Lr*r*JT2;
                        ws=ws+a1+mc*a1R;
                        vs=vs+a2+mc*a2R;
                        ftn1=(1./(1+exp(-Ls*(ws*ptn))));
                        ftn1(s+1,:)=ones(1,zt);
                        ftn=vs*ftn1;
                        r=(ttn-ftn);
                        d=r.^2;
                        mse(k+1)=sum(sum(d),2)/yt;
                else
                    Lr=Lr*1.05;
```

ISSN 1512-1232

```
                    r=r1;
                end
            else
                break
            end
        wsR(:,:,k+1)=ws;
        vsR(:,:,k+1)=vs;
        %Test
            ftsn1=(1./(1+exp(-Ls*(ws*ptsn))));
            ftsn1(s+1,:)=ones(1,zts);
            ftsn=vs*ftsn1;
            rts=(ttsn-ftsn);
            dts=rts.^2;
            msets(k+1)=sum(sum(dts),2)/yts;

            %Validation
            fvn1=(1./(1+exp(-Ls*(ws*pvn))));
            fvn1(s+1,:)=ones(1,ztv);
            fvn=vs*fvn1;
            rv=(tvn-fvn);
            dv=rv.^2;
            msev(k+1)=sum(sum(dv),2)/yv;
            if msev(k+1)>=msev(k);
                l=l+1;
                if l==5;
                    break
                end
            else
                l=0;
            end
    else
        break
    end

end
if l>=5;
    ws=wsR(:,:,k-4);
    vs=vsR(:,:,k-4);
end
ftn1=(1./(1+exp(-Ls*(ws*ptn))));
ftn1(s+1,:)=ones(1,zt);
ftn=vs*ftn1;
fvn1=(1./(1+exp(-Ls*(ws*pvn))));
fvn1(s+1,:)=ones(1,ztv);
fvn=vs*fvn1;
ftsn1=(1./(1+exp(-Ls*(ws*ptsn))));
ftsn1(s+1,:)=ones(1,zts);
ftsn=vs*ftsn1;
t=removeconstantrows('reverse',t,TS);
tt_exp=removeconstantrows('reverse',tt,TS);
tv_exp=removeconstantrows('reverse',tv,TS);
tts_exp=removeconstantrows('reverse',tts,TS);
```

ISSN 1512-1232

```
t_exp=t;
fft = mapminmax('reverse',ftn,ts);
ffv = mapminmax('reverse',fvn,ts);
ffts = mapminmax('reverse',ftsn,ts);
tt_model=removeconstantrows('reverse',fft,TS);
tv_model=removeconstantrows('reverse',ffv,TS);
tts_model=removeconstantrows('reverse',ffts,TS);
p=removeconstantrows('reverse',p,PS);
pt=removeconstantrows('reverse',pt,PS);
pv=removeconstantrows('reverse',pv,PS);
pts=removeconstantrows('reverse',pts,PS);
mse_minimum_train=min(mse)
mse_minimum_val=min(msev)
mse_minimum_test=min(msets)
number_iteration=k

% PERFORMANCE ANALYSIS
%Cross Correlation coefficient
%R=C(i,j)/(C(i,i)C(j,j))^1/2
%Training
Avt_exp=mean(mean(tt_exp,2));
Avt_model=mean(mean(tt_model,2));
Nmt=(sum(sum((tt_model-Avt_model).*(tt_exp-Avt_exp)),2));
Dnt=((sum(sum((tt_model-Avt_model).^2),2)).*(sum(sum((tt_exp-
Avt_exp).^2),2)));
Rtrain=Nmt/sqrt(Dnt);
%Validation
Avv_exp=mean(mean(tv_exp,2));
Avv_model=mean(mean(tv_model,2));
Nmv=(sum(sum((tv_model-Avv_model).*(tv_exp-Avv_exp)),2));
Dnv=((sum(sum((tv_model-Avv_model).^2),2)).*(sum(sum((tv_exp-
Avv_exp).^2),2)));
Rvalidation=Nmv/sqrt(Dnv);
%Testing
Avts_exp=mean(mean(tts_exp,2));
Avts_model=mean(mean(tts_model,2));
Nmts=(sum(sum((tts_model-Avts_model).*(tts_exp-Avts_exp)),2));
Dnts=((sum(sum((tts_model-Avts_model).^2),2)).*(sum(sum((tts_exp-
Avts_exp).^2),2)));
Rtest=Nmts/sqrt(Dnts);

%Average absolute relative error (AARE)
%AARE=(1/N)*sum(absolute(model output-exp value)/exp value));
AT=abs((tt-fft)./fft);
AAREtrain=(1/yt)*sum(sum(AT),2);
AV=abs((tv-ffv)./ffv);
AAREvalidation=(1/yv)*sum(sum(AV),2);
ATS=abs((tts-ffts)./ffts);
AAREtest=(1/yts)*sum(sum(ATS),2);

%PLOTS
plot(tt_exp,tt_model,'*')
```

ISSN 1512-1232

```
xlabel('experimantal average void fraction')
ylabel('MLP model average void fraction')
legend('Training')
title('Gradient Descent with adaptive step and momentum term')
figure
plot(tv_exp,tv_model,'*')
xlabel('experimantal average void fraction')
ylabel('MLP model average void fraction')
legend('Validation')
title('Gradient Descent with adaptive step and momentum term')
figure
plot(tts_exp,tts_model,'*')
xlabel('experimantal average void fraction')
ylabel('MLP model average void fraction')
legend('Testing')
title('Gradient Descent with adaptive step and momentum term')
figure
u=1:k+1;
plot(u,mse,u,msev,u,msets)
xlabel('Number of iterations')
ylabel('Mean Square Error MSE')
legend('Training','Validation','Testing')
title('Gradient Descent with adaptive step and momentum term
performance plot')

bin=0.01:0.01:1;
bin=bin';
%Train
%for i=0:18;
%comparetrain(:,2*i+1:2*i+2)=[tt_exp(:,i+1) tt_model(:,i+1)];
%end
%comparetrain;

%PDF=comparetrain/60440;
%for u=0:numel(PDF(1,:))/2-1
 %   figure
%plot(bin',PDF(:,2*u+1),bin',PDF(:,2*u+2),'--')
%xlabel('void fraction')
%ylabel('PDF')
%legend('Experimental','GDMV MLP MODEL')
%title('Probabilty Density Function')
%end

%Validation
%for i=0:6;
%comparevalidation(:,2*i+1:2*i+2)=[tv_exp(:,i+1) tv_model(:,i+1)];
%end
%comparevalidation;

%PDF=comparevalidation/60440;
%for u=0:numel(PDF(1,:))/2-1
 %   figure
```

ISSN 1512-1232

```
%plot(bin',PDF(:,2*u+1),bin',PDF(:,2*u+2),'--')
%xlabel('void fraction')
%ylabel('PDF')
%legend('Experimental','GDMV MLP MODEL')
%title('Probabilty Density Function')
%end

%Test
for i=0:6;
comparetest(:,2*i+1:2*i+2)=[tts_exp(:,i+1) tts_model(:,i+1)];
end
comparetest;

PDF=comparetest/60440;
for u=0:numel(PDF(1,:))/2-1
    figure
plot(bin',PDF(:,2*u+1),bin',PDF(:,2*u+2),'--')
xlabel('void fraction')
ylabel('PDF')
legend('Experimental','GDMV MLP MODEL')
title('Probabilty Density Function')
end
```

Article received: 2016-08-29