CI&CEQ

Available on line at
Association of the Chemical Engineers AChE
www.ache.org.rs/CICEQ

Chemical Industry & Chemical Engineering Quarterly 15 (2) 103–117 (2009) CI&CEQ

S. K. LAHIRI
K. C. GHANTA

Department of Chemical Engineering, NIT, Durgapur, West Bengal, India

SCIENTIFIC PAPER

UDC 621.643.2:517.9:621.6.07

ARTIFICIAL NEURAL NETWORK MODEL WITH THE PARAMETER TUNING ASSISTED BY A DIFFERENTIAL EVOLUTION TECHNIQUE: THE STUDY OF THE HOLD UP OF THE SLURRY FLOW IN A PIPELINE

This paper describes a robust hybrid artificial neural network (ANN) methodology which can offer a superior performance for the important process engineering problems. The method incorporates a hybrid artificial neural network and differential evolution technique (ANN-DE) for the efficient tuning of ANN meta parameters. The algorithm has been applied for the prediction of the hold up of the solid liquid slurry flow. A comparison with selected correlations in the literature showed that the developed ANN correlation noticeably improved the prediction of hold up over a wide range of operating conditions, physical properties, and pipe diameters.

Key words: artificial neural network; differential evolution; slurry hold up, slurry flow.

In the last decade, artificial neural networks (ANNs) have emerged as attractive tools for nonlinear process modeling especially in situations where the development of phenomenological or conventional regression models becomes impractical or cumbersome. ANN is a computer modeling approach that learns from examples through iterations without requiring a prior knowledge of the relationships of process parameters and is, consequently, capable of adapting to a changing environment. It is also capable of dealing with uncertainties, noisy data, and non-linear relationships [1].

ANN modeling has been known as “effortless computation” and readily used extensively due to their model-free approximation capabilities of complex decision-making processes. The advantages of an ANN-based model are: (*i*) it can be constructed solely from the historic process input-output data (example set), (*ii*) detailed knowledge of the process phenomenology is unnecessary for the model development, (*iii*) a properly trained model possesses an excellent generalization ability owing to which it can accurately predict outputs for a new input data set and (*iv*) even multiple input-multiple output (MIMO) nonlinear relationships can be approximated simultaneously and easily [1]. Owing to their several attractive characteristics, ANNs have been widely used in chemical engineering applications such as steady state and dynamic process modeling, process identification, yield maximization, nonlinear control, and fault detection and diagnosis (see, *e.g.* [2-5]).

The most widely utilized ANN paradigm is the multi-layered perceptron (MLP) that approximates nonlinear relationships existing between an input set of data (causal process variables) and the corresponding output (dependent variables) data set. A three-layer MLP with a single intermediate (hidden) layer housing a sufficiently large number of nodes (also termed neurons or processing elements) can approximate (map) any nonlinear computable function to an arbitrary degree of accuracy. It learns the approximation through a numerical procedure called “network training” wherein network parameters (weights) are adjusted iteratively so that the network, in response to the input patterns in an example set, accurately produces the corresponding outputs. There are a number of algorithms [1] – each possessing certain positive characteristics – to train an MLP network, *e.g.* the most popular error-back-propagation (EBP), Quickprop and Resilient Back-propagation (RPROP) [6]. The training of an ANN involves minimizing a nonlinear error function (*e.g.*, root-mean-squared-error, *RMSE*) that may possess several local minima. Thus, it becomes necessary to employ a heuristic procedure involving mul-

Corresponding author: S. K. Lahiri, Department of Chemical Engineering, NIT, Durgapur, West Bengal, India.
E-mail: sk_lahiri@hotmail.com
Paper received: 11.11.2008
Paper revised: 5.03.2009
Paper accepted: 19.03.2009

tiple training runs to obtain an optimal ANN model parameters (weights) of which correspond to the global or the deepest local minimum of the error function. The building of a back-propagation network involved the specification of the number of hidden layers and the number of neurons in each hidden layer. In addition, several parameters including the learning rule, the transfer function, the learning coefficient ratio, the random number seed, the error minimization algorithm, and the number of learning cycles had to be specified [2].

The existing software implementations of ANN regression usually treat these ANN meta-parameters as user-defined inputs. For a non-expert user it is a very difficult task to choose these parameters as he has no prior knowledge of these parameters for his data. In such a situation, users normally rely on a trial-and-error method. Such an approach, apart from consuming the enormous time, may not really obtain the best possible performance.

This study is motivated by a growing popularity of ANN for the process modeling and regression problems. However, many ANN regression application studies are performed by "expert" users having good understanding of ANN methodology. Since the quality of ANN models depends on a proper setting of ANN architecture and ANN meta-parameters, the main issue for practitioners trying to apply ANN regression is how to set these parameter values (to ensure good generalization performance) for a given data set. Non--expert users face difficulty to find the optimum ANN architecture and are confused about how to choose the ANN meta parameters. Whereas the existing sources [1,7-9] on ANN regression give some recommendations on the appropriate setting of ANN parameters, there is clearly no consensus and plenty of contradictory opinions. Also, there are a lot of ANN algorithms available in literature [1] with their respective advantages. Some algorithms require less computation time and computer storage requirement; some others have a more accurate prediction capability. It is difficult for a non-expert user to choose the best (robust, accurate and fast) algorithm for his case. The present paper addresses this issue and develops a new hybrid procedure to find the optimum ANN architecture and tune the ANN parameters and thus relieve the "non-expert" users.

Basically, the setting of the optimum ANN architecture and tuning of ANN meta parameters can be viewed mathematically as an optimization problem where test set errors (generalization error) have to be minimized. In the recent years, Differential Evolution (DEs) that are the members of the stochastic optimization formalisms have been used with a great success in solving problems involving very large search spaces. The DEs were originally developed as the genetic engineering models mimicking the population evolution in natural systems. Specifically, DE like a genetic algorithm (GA) enforces the "survival-of-the-fittest" and "genetic propagation of characteristics" principles of the biological evolution for searching the solution space of an optimization problem. DE has been used to design several complex digital filters [10] and to design fuzzy logic controllers [11]. DE can also be used for parameter estimations, *e.g.* Babu and Sastry [12] used DE for the estimation of the effective heat transfer parameters in trickle-bed reactors using radial temperature profile measurements. They concluded that DE takes less computational time to converge compared to the existing techniques without compromising with the accuracy of the parameter estimates.

In this paper, we present a hybrid ANN-DE approach, which not only relieves the user from choosing these meta-parameters but also finds out the optimum values of these parameters to minimize the generalization error. The strategy (henceforth referred to as "ANN-DE") uses an ANN as the nonlinear process modeling paradigm, and the DE for optimizing the meta-parameters of the ANN model so that an improved prediction performance is realized. In the present work, we illustrate the ANN-DE approach by applying it for predicting the hold up of the solid-liquid flow.

ARTIFICIAL NEURAL NETWORK (ANN) MODELING

Neural networks are computer algorithms inspired by the way the information is processed in the nervous system. An ANN is a massively parallel-distributed processor that has a natural propensity for storing experimental knowledge and making it available. An important difference between neural networks and standard Information Technology (IT) solutions is their ability to learn (Baughman and Liu, 1995). This learning property has yielded a new generation of algorithms. An ANN paradigm is composed of a large number of highly interconnected processing elements, analogous to biological neurons that are tied together with weighted connections that are analogous to synapses. Learning in biological systems involves adjustments to the synaptic connections between the neurons. This is true for ANNs as well. Learning typically occurs through training or exposure to a true set of input/output data where the training algorithm iteratively adjusts the connection

weights. These connection weights represent the knowledge necessary to solve a specific problem.

Network architecture

The MLP network used in the model development is depicted in Fig. 1. As shown, the network usually consists of three layers of nodes. The layers described as input, hidden and output layers, comprise N, L and K number of processing nodes, respectively. Each node in the input (hidden) layer is linked to all the nodes in the hidden (output) layer using weighted connections. In addition to the N and L number of input and hidden nodes, the MLP architecture also houses a bias node (with fixed output of +1) in its input and hidden layers; the bias nodes are also connected to all the nodes in the subsequent layer and they provide additional adjustable parameters (weights) for the model fitting. The number of nodes (N) in the MLP network input layer is equal to the number of inputs in the process whereas the number of output nodes (K) equals the number of the process outputs. However, the number of hidden nodes (L) is an adjustable parameter magnitude of which is determined by issues, such as the desired approximation and generalization capabilities of the network model.

Training

Training a network consists of an iterative process in which the network is given the desired inputs along with the correct outputs for those inputs. It then seeks to alter its weights to try and produce the correct output (within a reasonable error margin). If it succeeds, it has learned the training set and is ready to perform upon previously unseen data. If it fails to produce the correct output it re-reads the input and again tries to produce the correct output. The weights are slightly adjusted during each iteration through the training set (known as a training cycle) until the appropriate weights have been established. Depending upon the complexity of the task to be learned, many thousands of training cycles may be needed for the network to correctly identify the training set. Once the output is correct the weights can be used with the same network on unseen data to examine how well it performs.

Back propagation algorithm (BPA)

The back propagation algorithm modifies network weights to minimize the mean squared error between the desired and the actual outputs of the network. Back propagation uses supervised learning in which the network is trained using data for which the input, as well as desired outputs, is known (Baughman and Liu, 1995). Once trained, the network weights are frozen and can be used to compute the output values for the new input samples.

The feed forward process involves presenting the input data to input layer neurons that pass the input values onto the first hidden layer. Each of the hidden layer nodes computes a weighted sum of its input and passes the sum through its activation function and presents the result to the output layer. The

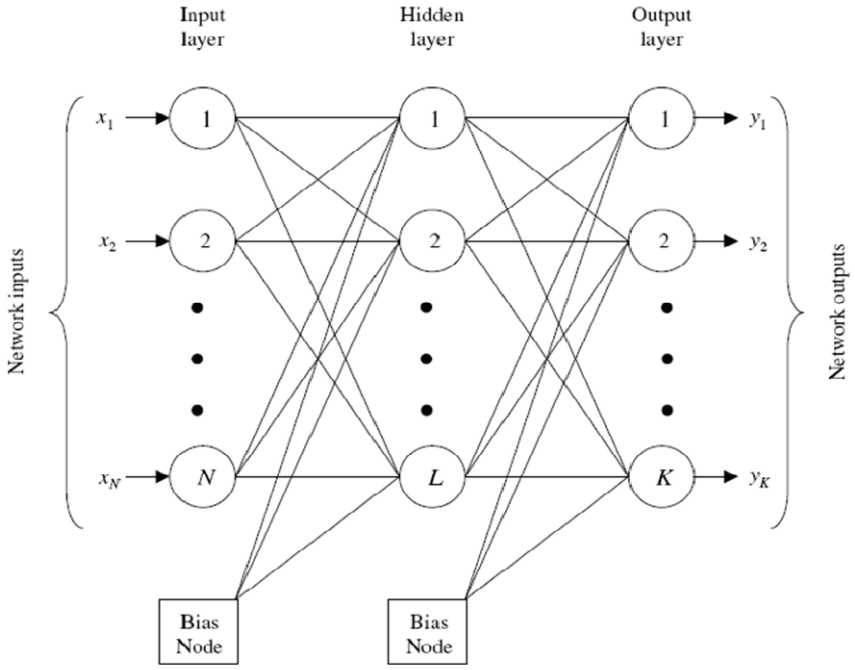

Fig 1. A schematic diagram of artificial neural network and architecture of the feed forward network with one hidden layer.

goal is to find a set of weights that minimize the mean squared error. A typical back propagation algorithm can be given as follows:

The MLP network is a nonlinear function-mapping device that determines the K dimensional nonlinear function vector **f**, where **f**: $X \rightarrow Y$. Here, X is a set of N-dimensional input vectors ($X = \{x_p\}$; p = 1,2, …,P and $x = [x_1,x_2,...,x_n,...,x_N]^T$), and Y is the set of corresponding K-dimensional output vectors ($Y = \{y_p\}$; p = 1,2,…,P where $y = [y_1,y_2,...,y_k,...,y_K]^T$). The precise form of **f** is determined by: (*i*) network topology, (*ii*) choice of the activation function used for computing outputs of the hidden and output nodes and (*iii*) network weight matrices $\mathbf{W}^H$ and $\mathbf{W}^O$ (they refer to the weights between input and hidden nodes, and hidden and output nodes, respectively). Thus, the nonlinear mapping can be expressed as:

$$y = y(x,\mathbf{W}) \tag{1}$$

where, $\mathbf{W} = \{\mathbf{W}^H,\mathbf{W}^O\}$. This equation suggests that y is a function of x, which is parametrized by **W**. It is now possible to write the closed-form expression of the input-output relationship approximated by the three-layered MLP as:

$$y_k = \tilde{f}_2\left[\sum_{l=0}^{L} w_{lk}^{o}\,\tilde{f}_1\left[\sum_{n=0}^{N} \mathbf{W}_{nl}^{H} x_n\right]\right]; \quad k = 1,2,...,K \tag{2}$$

where y_k refers to the kth network output; $\tilde{f}_1$ and $\tilde{f}_2$ denote the nonlinear activation functions; w_{lk}^0 refers to the weight between lth hidden node and kth output node; w_{nl}^0 is the weight between nth input and lth hidden node, and x_n represents the nth network input.

Note that in Eq. (2) the bias node is indexed as the zeroth node in the respective layer. In order that an MLP network approximates the nonlinear relationship existing between the process inputs and outputs, it needs to be trained in a manner such that a prespecified error function is minimized. In essence, the MLP-training procedure aims at obtaining an optimal set (**W**) of the network weight matrices $\mathbf{W}^H$ and $\mathbf{W}^O$, which minimize an error function. The commonly employed error function is the average absolute relative error *(AARE)* defined as:

$$AARE = \frac{1}{N}\sum_{1}^{N}\left|\left(\frac{y_{\text{predicted}} - y_{\text{experimental}}}{y_{\text{experimental}}}\right)\right| \tag{3}$$

The most widely used formalism for the *AARE* minimization is the error-back-propagation (EBP) algorithm utilizing a gradient-descent technique known as the generalized delta rule (GDR) (Baughman and Liu, 1995). In the EBP methodology, the weight matrix set, **W**, is initially randomized. Thereafter, an input vector from the training set is applied to the network's input nodes and the outputs of the hidden and output nodes are computed. The outputs are computed as follows. First the weighted-sum of all the node-specific inputs is evaluated, which is then transformed using a nonlinear activation function, such as the logistic sigmoid. The outputs from the output nodes are then compared with their target values and the difference is used to compute the *AARE* defined in Eq. (3). Upon *AARE*-computation, the weight matrices $\mathbf{W}^H$ and $\mathbf{W}^O$ are updated using the GDR framework. This procedure, when repeated with the remaining input patterns in the training set, completes one network training iteration. For the *AARE* minimization, several training iterations are usually necessary.

Generalizability

Neural learning is considered successful only if the system can perform well on test data on which the system has not been trained. This capability of a network is called generalizability. Given a large network, it is possible that repeated training iterations successsively improve the performance of the network on the training data, *e.g.* by "memorizing" training samples, but the resulting network may perform poorly on the test data (unseen data). This phenomenon is called "overtraining". The proposed solution is to constantly monitor the performance of the network on the test data. Hecht-Nielsen [13] proposes that the weight should be adjusted only on the basis of the training set, but the error should be monitored on the test set. Here we apply the same strategy: training continues as long as the error on the test set continues to decrease and is terminated if the error on the test set increases. Training may thus be halted even if the network performance on the training set continues to improve.

Tuning parameters of ANN

It is well known that ANN generalization performance (estimation accuracy) depends on a good setting of meta-parameters listed below.

1. No of nodes in hidden layer: the number of nodes in hidden layer has a profound effect on ANN performance. Too few nodes could not learn the relationship in data properly and too large number of nodes increases the network complexity and execution time. From literature, it is found that the optimal number of nodes in a hidden layer is normally calculated by the trial and error method. Such an approach, apart from consuming enormous time, may not really obtain the best possible performance.

2. The activation functions in the input layer: each hidden node and output node applies the acti-

vation function to its net input. Five types of the activation function, reported in literature and used in this work, are shown in Table 1 and Fig. 2.

Table 1. Different activation function

Case	Name of activation function	Equation[a]
1	Log sigmoid function (*logsig*)	$Yi = \frac{1}{(1+\exp(-neti))}$
2	Tan hyperbolic function (*tansig*)	$Yi = \tanh(neti)$
3	Linear function (*purelin*)	$Yi = (neti)$
4	Radial basis function (*radbas*)	$Yi = \exp[-(neti)^2]$
5	Triangular basis function (*tribas*)	$Yi = 1 - \text{abs}(neti)$ if $-1 \le (neti) \le 1$; otherwise, $Yi = 0$

[a] *Yi* is the output from node *I*, *neti* is the input to the node $i = \Sigma WiXi$

There is no consensus in literature on which type of the activation function is to be used and it depends on the type of input training data and the case under investigation. For a new user it is difficult to choose the activation function for his data as he has no guidelines on what to choose. Multilayer networks typically use sigmoid transfer functions in the hidden layers. Sigmoid functions are characterized by the fact that their slope must approach zero, as the input gets larger. This causes a problem when using the steepest descent to train a multilayer network with sigmoid functions, since the gradient can have a very small magnitude; and therefore, it causes small changes in the weights and biases, even though the weights and biases are far from their optimal values.

3. The activation function of the output layer: the same remarks of the input activation are applicable for it.

4. The learning rate: the performance of the back propagation algorithm can be improved if we estimate the optimal learning rate. For a new user choosing the optimal learning rate is very difficult. The learning rate is multiplied with the negative of the gradient to determine the changes to the weights and biases. The larger the learning rate, the bigger the step. If the learning rate is made too large, the algorithm becomes unstable. If the learning rate is set too small, the algorithm takes a long time to converge.

Apart from the above 4 parameters, the ANN performance also depends upon the training algorithm used for back propagation. Over the years, different researchers have developed many ANN training algorithms to reduce the execution time and computer storage requirements. There are several different back propagation training algorithms published in literature. Fig. 3 shows some of those algorithms used in the present study. They have a variety of different computation and storage requirements, and no algorithm is best suited to all locations. The basic differences between these algorithms are how they handle the weight up-gradation in Eqs. (1) and (2) to reduce error and how they modify the learning rate (η) to reduce the convergence time.

Most of the available ANN software applications require the above four parameters from user. Otherwise, in absence of user input, the software automatically calculated the above parameters on trial and error basis. This needs a long execution time to explore all the combinations of the above parameters to really

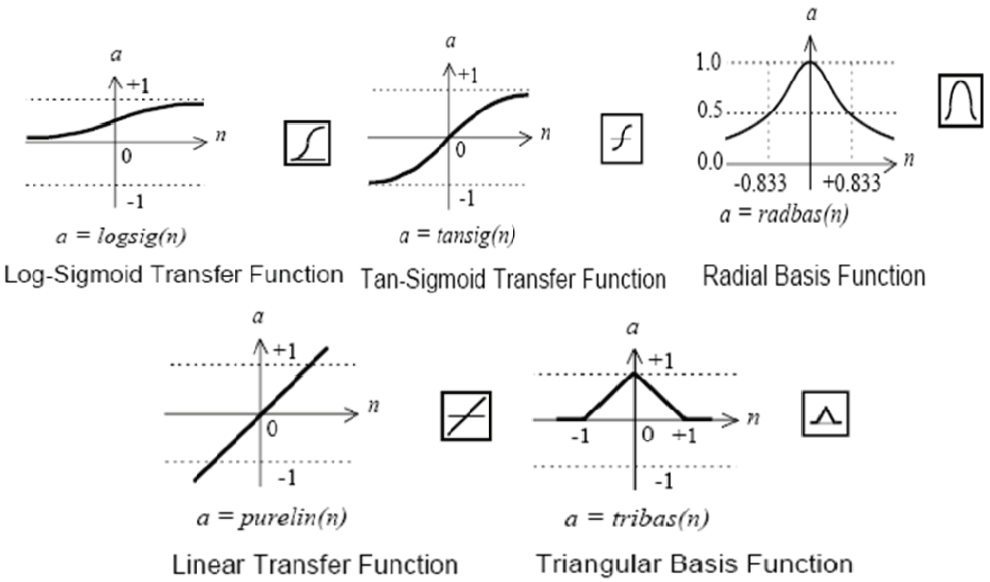

Fig. 2. Structure of different activation function.

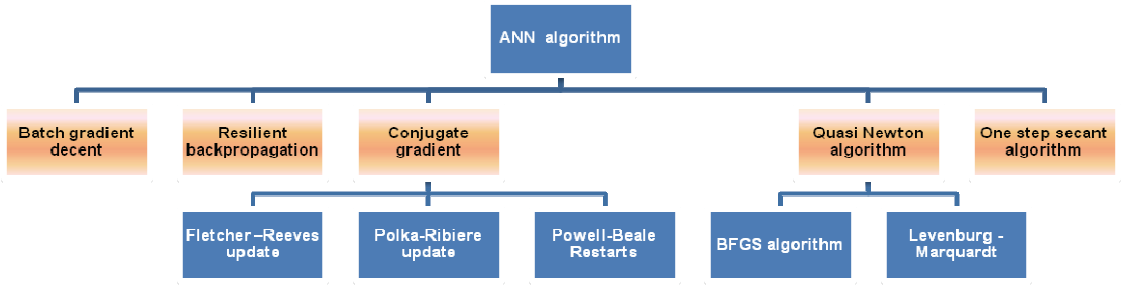

Fig. 3. Different ANN algorithms published in various literatures.

find the best possible solutions. In the present paper we use a differential evolution technique to find out the optimum solution.

DIFFERENTIAL EVOLUTION (DE): AT A GLANCE

Having developed an ANN-based process model, a DE algorithm is used to optimize the *N*- dimensional input space of the ANN model. Conventionally, various deterministic gradient-based methods are used for performing optimization of the phenomenological models. Most of these methods require that the objective function should simultaneously satisfy the smoothness, continuity, and differentiability criteria. Although the nonlinear relationships approximated by an ANN model can be expressed in the form of generic closed-form expressions, the objective function(s) derived thereby cannot be guaranteed to satisfy the smoothness criteria. Thus, the gradient-based methods cannot be efficiently used for optimizing the input space of an ANN model and, therefore, it becomes necessary to explore alternative optimization formalisms, which are lenient towards the form of the objective function.

In the recent years, Differential Evolution (DE) that are members of the stochastic optimization formalisms have been used with a great success in solving problems involving very large search spaces. The DEs were originally developed as the genetic engineering models mimicking the population evolution in natural systems. Specifically, DE like genetic algorithm (GA) enforce the "survival-of-the-fittest" and "genetic propagation of characteristics" principles of biological evolution for searching the solution space of an optimization problem. The principal features possessed by the DEs are: (*i*) they require only scalar values and not the second- and/or first-order derivatives of the objective function, (*ii*) the capability to handle nonlinear and noisy objective functions, (*iii*) they perform global search and thus are more likely to arrive at or near the global optimum and (*iv*) DEs do not impose pre-conditions, such as smoothness, differentiability and continuity, on the form of the objective function.

Differential Evolution (DE), an improved version of GA, is an exceptionally simple evolution strategy that is significantly faster and robust at numerical optimization and is more likely to find a function's true global optimum. Unlike simple GA that uses a binary coding for representing problem parameters, DE uses real coding of floating point numbers. The mutation operator here is the addition instead of bit-wise flipping used in GA. And DE uses non-uniform crossover and tournament selection operators to create new solution strings. Among the DEs advantages are its simple structure, ease of use, speed and robustness. It can be used for optimizing functions with real variables and many local optima.

This paper demonstrates a successful application of DE to the practical optimization problem. As already stated, DE in principle is similar to GA. So, as in GA, we use a population of points in our search for the optimum. The population size is denoted by NP. The dimension of each vector is denoted by *D.* The main operation is the *NP* number of competitions that are to be carried out to decide the next generation.

To start with, we have a population of *NP* vectors within the range of the objective function. We select one of these *NP* vectors as our target vector. We then randomly select two vectors from the population and find the difference between them (vector subtraction). This difference is multiplied by a factor *F* (specified at the start) and added to the third randomly selected vector. The result is called the noisy random vector. Subsequently, the crossover is performed between the target vector and noisy random vector to produce the trial vector. Then, a competition between the trial vector and target vector is performed and the winner is replaced into the population. The same procedure is carried out *NP* times to decide the next generation of vectors. This sequence is continued till some convergence criterion is met. This summarizes the basic procedure carried out in differential

evolution. The details of this procedure are described below.

Steps performed in DE

Assume that the objective function is of *D* dimensions and that it has to be optimized. The weighting constants *F* and the crossover constant *CR* are specified.

Step 1. Generate *NP* random vectors as the initial population: generate (NP×*D*) random numbers and linearize the range between 0 and 1 to cover the entire range of the function. From these (NP×*D*) numbers, generate NP random vectors, each of dimension *D,* by mapping the random numbers over the range of the function.

Step 2. Choose a target vector from the population of size *NP*: first generate a random number between 0 and 1. From the value of the random number decide which population member is to be selected as the target vector (*Xi*) (a linear mapping rule can be used).

Step 3. Choose two vectors from the population at random and find the weighted difference: Generate two random numbers. Decide which two population members are to be selected (X*a,Xb).* Find the vector difference between the two vectors (X*a - Xb).* Multiply this difference by *F* to obtain the weighted difference.

Weighted difference = *F* (*Xa - Xb)*

Step 4. Find the noisy random vector: generate a random number. Choose the third random vector from the population (*Xc*). Add this vector to the weighted difference to obtain the noisy random vector (*X'c*).

Step 5. Perform the crossover between *Xi* and *X'c* to find *Xt,* the trial vector: generate *D* random numbers. For each of the *D* dimensions, if the random number is greater than *CR*, copy from *Xi* into the trial vector; if the random number is less than *CR*, copy the value from *X'c* into the trial vector.

Step 6. Calculate the cost of the trial vector and the target vector: for a minimization problem, calculate the function value directly and this is the cost. For a maximization problem, transform the objective function *f(x)* using the rule *F(x)* = 1/[1 + *f(x)*] and calculate the value of the cost. Alternatively, directly calculate the value of *f(x)* and this yields the profit. In case the cost is calculated, the vector that yields the lower cost replaces the population member in the initial population. In case the profit is calculated, the vector with the greater profit replaces the population member in the initial population.

Steps 1-6 are continued till some stopping criterion is met. This may be of two kinds. One may be some convergence criterion that states that the error in the minimum or maximum between two previous generations should be less than some specified value. The other may be an upper bound on the number of generations. The stopping criterion may be a combination of the two. Either way, once the stopping criterion is met, the computations are terminated.

Choosing DE key parameters *NP*, *F*, and *CR* is seldom difficult and some general guidelines are available. Normally, *NP* ought to be about 5 to 10 times the number of parameters in a vector. As for *F*, it lies in the range 0.4 to 1.0. Initially, *F* = 0.5 can be tried and then *F* and/or *NP* is increased if the population converges prematurely. A good first choice for *CR* is 0.1, but in general *CR* should be as large as possible (Price and Storn, 1997).

DE has already been successfully applied for solving several complex problems and is now being identified as a potential source for the accurate and faster optimization.

DE-BASED OPTIMIZATION OF ANN MODELS

There are different measures by which ANN performance is assessed, the validation and leave-one--out error estimates being the most commonly used one. Here, we divide the total available data as training data (75% of data) and test data (25% data chosen randomly). While ANN algorithm was trained on the training data but the ANN performance is estimated on the test data.

The statistical analysis of ANN prediction is based on the following performance criteria:

1. The average absolute relative error (*AARE*) on the test data should be minimum:

$$AARE = \frac{1}{N}\sum_{1}^{N}\left|\left(\frac{y_{\text{predicted}} - y_{\text{experimental}}}{y_{\text{experimenta/}}}\right)\right|$$

2. The standard deviation of error (σ) on the test data should be minimum:

$$\sigma = \sqrt{\sum_{1}^{N}\frac{1}{N-1}[\left|\left(y_{\text{predicted}(i)} - y_{\text{experimental}(i)}\right)/y_{\text{experimental}(i)}\right| - AARE]^2}$$

3. The cross-correlation co-efficient (*R*) between the input and output should be around unity:

$$R = \frac{\sum_{i=1}^{N}(y_{\text{experimental}(i)} - y_{\text{experimental(mean)}})(y_{\text{predicted}(i)} - y_{\text{predicted(mean)}})}{\sqrt{\sum_{i=1}^{N}(y_{\text{experimental}(i)} - y_{\text{experimental(mean)}})^2}\sqrt{\sum_{i=1}^{N}(y_{\text{predicted}(i)} - y_{\text{predicted(mean)}})^2}}$$

ANN learning is considered successful only if the system can perform well on the test data on which the system has not been trained. The above five parameters of ANN are optimized by DE algorithm stated below.

The objective function and the optimal problem of ANN model of the present study are represented as:

Minimize $AARE(X)$ on test set:

$X \in \{x1, x2, x3, x4, x5\}$

where $x1$ = number of nodes in hidden layer {1,2, …,100}, $x2$ = input layer activation function {1,2,3,4,5} (correspond to five activation function in Table 1), $x3$ = output layer activation function {1,2,3,4,5} (correspond to five activation function in Table 1), $x4$ = = learning rate {0 to 5}, $x5$ = training algorithm {1,2,…,8} (correspond to eight training algorithm as per Fig. 3).

The objective function is minimization of average absolute relative error (*AARE*) on the test set and X is a solution string representing a design configuration of ANN architecture. The best topology of ANN architecture should have lowest *AARE*, the lowest standard deviation of error and the highest correlation coefficient. The design variable $x1$ takes any integer values for the number of nodes in the range of 1 to 100, $x2$ represents the input layer activation function taking any values in the range of 1 to 5 corresponding to five activation function in Table 1. $X3$ represents the output layer activation function taking any values in the range of 1 to 5 corresponding to five activation function in table 1. $X4$ represents learning rates and can take any value between 0 to 5. The variable $x5$ takes eight values of the training algorithm which corresponds to eight ANN training algorithm in Fig. 3.

The total number of design combinations with these variables is 100×5×5×5×8 = 100000. This means that if an exhaustive search is to be performed it will take at the maximum 1000000 function evaluations before arriving at the global minimum *AARE* for the test set (assuming 5 trials for arriving at the optimum learning rate). So, the strategy which takes few function evaluations is the best one. Considering the minimization of *AARE* as the objective function, a differential evolution technique is applied to find the optimum design configuration of ANN model. The methodology adopted is shown in Fig. 4.

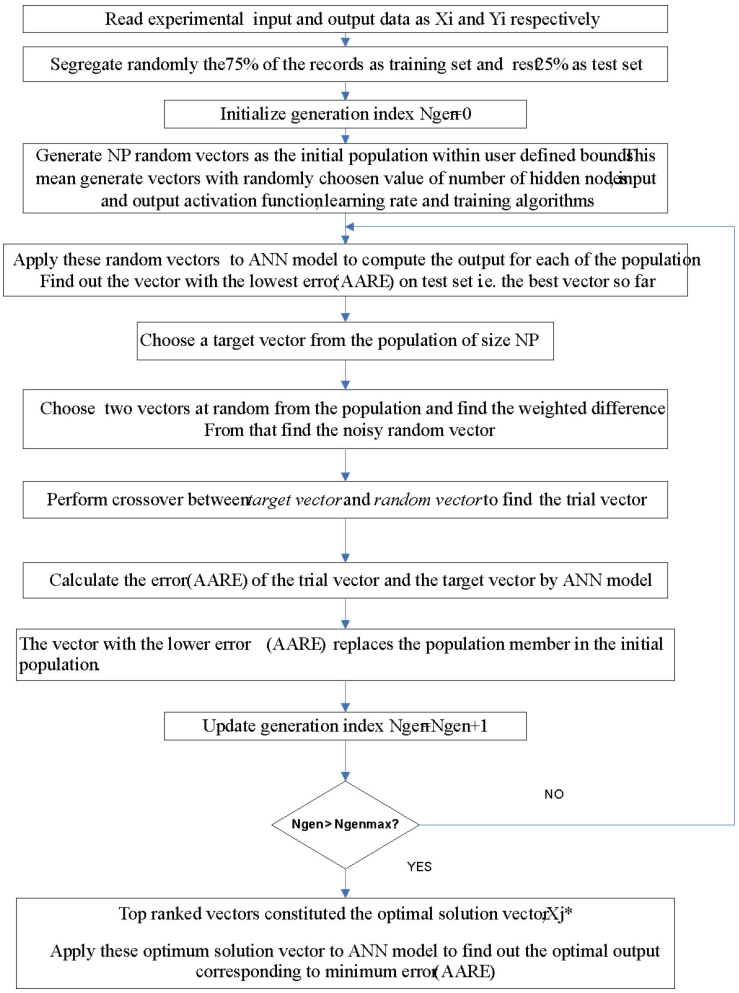

Fig. 4. Schematic for hybrid ANN-DE algorithm implementation.

A matlab program was made by combining the matlab neural network toolbox and a differential evolution algorithm. This matlab program was used throughout the project.

CASE STUDY: PREDICTION OF HOLD UP IN SOLID-LIQUID SLURRY FLOW

The hybrid ANN-DE algorithm has been applied for the prediction of the hold up of the solid-liquid slurry flow. The hold up case study was chosen as a phenomenological model that is not available for it and very few published correlations are available to predict the hold up in slurry pipelines.

Background and importance of hold up

The transportation of slurries through a pipeline is common in solid handling, mineral, hydrocarbon and petrochemical industries as the use of the pipeline ensures a dust free environment, demands substantially less space, makes the full automation possible and requires less operating staff. On the other hand, higher operational pressures and considerable demands are needed for the high quality of the pumping equipment and control system. Due to cut throat global competition, process industries are now forced to reduce the power consumption of the slurry transport to remain competitive in the world market. A slurry pipeline design requires the pressure drop predictiton which in turn depends on the hold up. This case study develops a hybrid ANN-DE technique to minimize the power consumption in the slurry transport in a long pipeline by predicting the hold up accurately.

Hold up

To understand the hold up phenomena it is very important to know the different flow regimes of the slurry flow. There are four main flow regimes in a horizontal pipeline flow (Fig. 5) [14]. These are:

1. flow with a stationary bed;
2. flow with a moving bed and saltation (with or without suspension);
3. heterogeneous mixture with all solids in suspension;
4. Pseudo homogeneous or homogeneous mixtures with all solids in suspension.

The tendency that the solid particles have to settle under the influence of gravity has a significant effect on the behavior of the slurry that is transported in a horizontal pipeline. The settling tendency leads to a significant gradation in the concentration of solids in the slurry. The concentration of solids is higher in the lower sections of the horizontal pipe. The extent of the accumulation of solids in the lower section depends strongly on the velocity of the slurry in the pipeline. The higher the velocity, the higher the turbulence level and the greater the ability of the carrier fluid to keep the particles in suspension. It is the upward motion of eddy currents transverse to the main direction of the slurry flow that is responsible for maintaining the particles in suspension. At very high turbulence levels the suspension is almost homogeneous with very good dispersion of the solids while at low turbulence levels the particles settle towards the floor of the channel and can in fact remain in contact with the flow and are transported as a sliding bed under the influence of the pressure gradient in the fluid. Between these two extremes of behavior, two other more or less clearly defined flow regimes can be identified. When the turbulence level is not high to prevent any deposition of particles on the floor of the channel, the flow regime is described as being a heterogeneous suspension. As the velocity of the slurry is reduced further, a distinct mode of transport known as saltation develops. In the saltation regimes, there is a visible layer of particles on the floor of the channel and these are being continually picked up by turbulent eddies and dropped to the floor again further down the pipeline.

The solids therefore spend some of their time on the floor and the rest in the suspension in the flowing

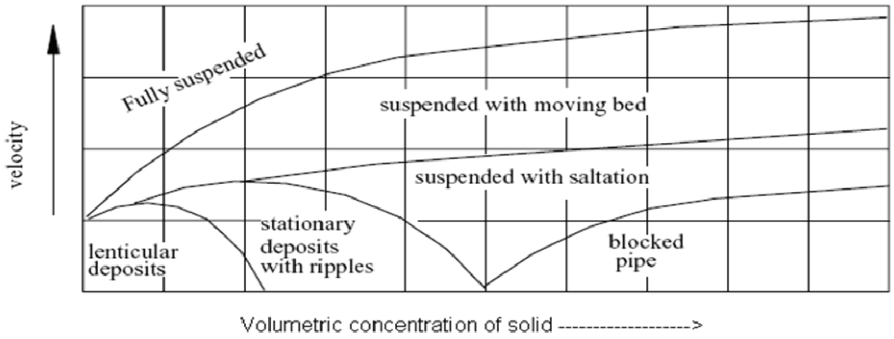

Fig. 5. Schematic of flow regimes in terms of velocity versus volumetric concentration [15].

fluid. Under saltation conditions, the concentration of solids is strongly non-uniform.

The previous section describes how different layers of solids move at different speeds from the bottom, having coarser particles, to the finer particles at the top layer in the horizontal pipe. Hold-ups are due to the velocity slip of layers of particles of larger sizes, particularly in the moving bed flow pattern.

Newitt [15] conducted speed measurements of a slurry mixture in a horizontal pipe. In the case of light plexiglas pipe, zircon or fine sand did not result in local slip; particles and water moved at the same speed. However, for coarse sand and gravel, they observed asymmetric suspension and a sliding bed. They also observed that in the upper layers of the horizontal pipe, the concentrations of larger particles were the same as for finer solids, but were marked by differences in the magnitude of the discharge rate of the lower layers.

Thus, in the solid-liquid multiphase flow, the separate phases move at different average velocities and the in-situ concentrations are not same as the concentrations in which the phases are introduced or removed from the system. The variation of in-situ concentrations from the supply concentrations is referred to as hold up phenomenon. The hold up effect is measured by the hold up ratio, given by the ratio of the average in-situ concentration and mean discharge concentration [16]:

$$\text{Hold up ratio} = \frac{c_{\text{ins}}}{c_{\text{dis}}} = \frac{\int_A c\,\mathrm{d}A}{\int_A \frac{V_{\text{m,ins}}}{V_{\text{m,avg}}} c\,\mathrm{d}A}$$

In the homogeneous suspension flow regime, it is common to assume that hold up is negligible and therefore in-situ and the input concentrations are considered identical. This is not practically correct but is a suitable approximation in many cases. Hold up has been clearly observed, especially in the flow regimes involving bed formation. For coarse particles, the difference in fluid and particle mean velocities are quite appreciable and this difference is referred to as the slip velocity. It is probable that a large contribution to the apparent slip velocity is due to the fluid flowing at high velocity through the portion of low solids concentration in the pipe. Obviously the hold up plays an important role in the failure of many empirical correlations in predicting a head loss in flow regimes involveing bed formation.

To facilitate the reduction of power consumption in the slurry transport, there is a need for a model that can predict the hold up and slurry pressure drop over a wide range of operating conditions, physical properties and particle size distributions. A successful predictive model with the sound understanding of the fundamentals of the particle laden turbulent flow (Fig. 6), including all significant interactions, has not yet been fully developed till today as seen from various literatures [17-20]. Despite poor understanding of the solid liquid slurry flow, different researchers at different point of time try to predict the hold up and came up with few correlations [16,19,20]. Table 7 listed two major correlations available in literature to predict the hold up. To test the suitability of different correlations we have collected 800 experimental hold up data from open literature spanning over the years 1973 to 2002. The majority of data (for earlier works from 1973

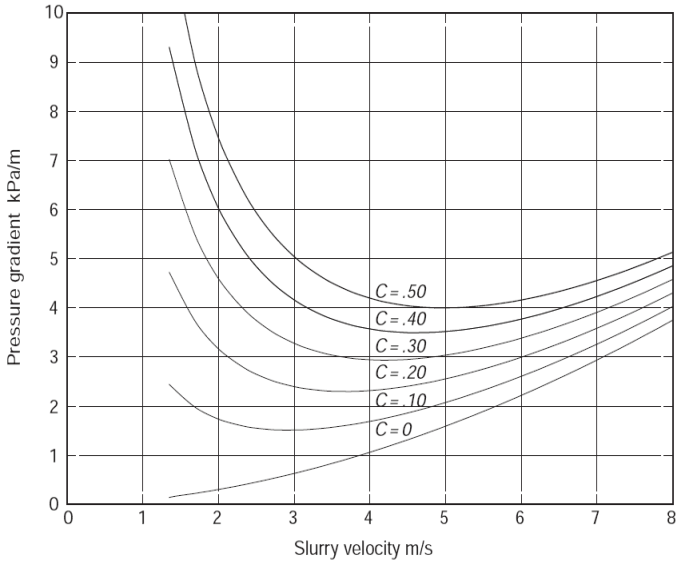

Fig. 6. Plot of transitional mixture velocity with pressure drop.

to 1987 [21-24]) was listed and summarized in PhD thesis of Hsu [16]. Rest of the data was collected from Govier and Aziz [19] and Kaushal and Tomita [20].

For all of the above empirical equations, when exposed to these collected hold up data for different systems, it was found that they predicted the hold up above 25% error on an average (refer Table 7). This may be due to the fact that empirical equations derived for the coal water slurry flow may be not fitted well for the sand water slurry and vice versa. One of the motivations of the present work is to develop a general hold up correlation which will reduce this prediction error.

To facilitate the optimization of power consumption in the slurry transport, the accurate prediction of hold up is very crucial. Once one can accurately predict the hold up, the slurry velocity can be maintained to reduce this hold up to minimize the pressure drop and power consumption. The industry needs quick and easily implement able solutions. The model derived from the first principle is no doubt the best solution. But in the scenario where the basic principles of the hold up model for accounting all the interactions for slurry flow are absent, the numerical model may be promising to give some quick, easy solutions for slurry hold up prediction.

This paper presents a systematic approach using robust hybrid ANN-DE techniques to build a hold up correlation from available experimental data. This correlation has been derived from a broad experimental data bank collected from the open literature (800 measurements covering a wide range of pipe dimensions, operating conditions and physical properties).

Development of the artificial neural network based correlation

The development of the ANN-based correlation was started with the collection of a large databank. The next step was to perform an artificial neural network, and to validate it statistically.

Collection of data

As mentioned earlier, over the years researchers have simply quantified the hold up of the slurry flow in a pipeline. In this work, about 800 experimental points have been collected from 7 sources of open literature spanning the years 1973-2002. The data were screened for incompleteness, redundancies and evident inaccuracies. This wide range of database includes experimental information from different physiccal systems to develop a unified correlation for the hold up. Table 2 indicates the wide range of the collected databank for the hold up.

Table 2. System and parameter studied [16,19,20-24]; slurry system: coal water, copper ore water, sand water, gypsum water, glass propanol and gravel water

Parameter	Value
Pipe diameter, m	0.04–0.495
Particle diameter, μm	38.3–13000
Liquid density, kg/m^3	1000–1250
Solids' density, kg/m^3	1370 – 2844
Liquid viscosity, mPa s	0.12–4
Velocity, m/s	1.05–4.81
Solids concentration (volume fraction)	0.0372–0.333
Max. packing conc. (volume fraction)	0.58–0.8
Pressure drop, Pa/m	99.9–4727.7

Identification of input parameters

After the extensive literature survey, all physical parameters that influence the hold up are put in a so-called "wish-list". Based on different combinations of inputs, a trial and error method was used to finalize the input set which gives a reasonable low prediction error (*AARE*) when exposed to artificial neural network.

Based on the above analysis, the input variables such as pipe diameter, particle diameter, solids concentration, solid and liquid density and viscosity of flowing medium have been finalized to predict the hold up in a slurry pipeline. Table 3 shows some typical data used for artificial neural network.

RESULTS AND DISCUSSION

Prediction performance of hybrid ANN-DE model

As the magnitude of inputs and outputs greatly differ from each other, they are normalized in -1 to +1 scale. 75% of total dataset was chosen randomly for training and the rest 25% was selected for validation and testing.

Nine parameters were identified as input (Table 3) for ANN and the hold up is put as a target. These data were then exposed to a hybrid ANN-DE model described above. After optimization of five ANN parameters described above, the model output was summarized in Table 4. Out of all the possibilities, Fletcher Reeves update algorithm with six nodes in a hidden layer and a log sigmoidal and linear function in the input and output layer has emerged as the best solution (with the lowest *AARE* and σ along with the highest correlation coefficient, *R*) for the present case. The low *AARE* (2.5%) may be considered as an excellent prediction performance considering the poor understanding of the slurry flow phenomena and a large databank for training comprising various systems. The parity plot of the experimental and predicted hold up is shown in Figure 7 as a goodness of the hybrid ANN-DE performance.

Table 3. Typical input and output data for ANN training

Input parameters									Output
Pipe diameter cm	Particle diameter μm	Liquid density g/cm^3	Solids density g/cm^3	Liquid viscosity cp	Max. packing concentration (volume fraction)	Velocity m/s	Solids concentration (volume fraction)	Pressure drop Pa/m	Hold up ratio
5.26	38.3	1	2.33	1.00	0.69	1.11	0.1070	294.1	1.001
5.26	38.3	1	2.33	1.00	0.69	3.01	0.1070	1651.3	1.000
5.26	38.3	1	2.33	1.00	0.69	4.81	0.1070	3822.9	1.000
5.26	38.3	1	2.33	1.00	0.69	1.33	0.3060	542.9	1.000
5.26	38.3	1	2.33	1.00	0.69	3.12	0.3060	2352.6	1.000
5.26	38.3	1	2.33	1.00	0.69	4.70	0.3060	4727.7	1.000
20.85	190.0	1	1.37	1.14	0.78	2.59	0.3260	266.5	1.002
20.85	190.0	1	1.37	1.14	0.78	2.34	0.3270	226.3	1.003
20.85	190.0	1	1.37	1.14	0.78	2.01	0.3330	177.3	1.007
20.85	190.0	1	1.37	1.14	0.78	1.78	0.3270	147.0	1.030
20.85	190.0	1	1.37	1.14	0.78	1.59	0.3230	123.4	1.048
20.85	190.0	1	1.37	1.14	0.78	1.37	0.3270	99.9	1.064
5.15	165.0	1	2.65	1.00	0.58	1.66	0.0741	666.2	1.133
5.15	165.0	1	2.65	1.00	0.58	3.78	0.0897	2449.2	1.026
5.15	165.0	1	2.65	1.00	0.58	1.66	0.1694	901.3	1.104
5.15	165.0	1	2.65	1.00	0.58	4.17	0.1886	3428.9	1.002
5.15	165.0	1	2.65	1.00	0.58	1.66	0.2669	1136.4	1.049
5.15	165.0	1	2.65	1.00	0.58	4.33	0.2860	4408.1	1.000
26.30	165.0	1	2.65	1.00	0.58	2.90	0.0932	261.6	1.105
26.30	165.0	1	2.65	1.00	0.58	3.50	0.0921	334.1	1.086
26.30	165.0	1	2.65	1.00	0.58	2.90	0.1759	305.7	1.080
26.30	165.0	1	2.65	1.00	0.58	3.50	0.1726	382.1	1.066
26.30	165.0	1	2.65	1.00	0.58	2.90	0.2586	355.6	1.044
26.30	165.0	1	2.65	1.00	0.58	3.50	0.2597	453.6	1.032
26.30	165.0	1	2.65	1.00	0.58	2.90	0.3292	414.4	1.036
26.30	165.0	1	2.65	1.00	0.58	3.50	0.3241	526.1	1.043
49.50	165.0	1	2.65	1.00	0.58	3.16	0.0943	143.0	1.103
49.50	165.0	1	2.65	1.00	0.58	3.76	0.0923	186.1	1.083
49.50	165.0	1	2.65	1.00	0.58	3.07	0.1727	157.7	1.083
49.50	165.0	1	2.65	1.00	0.58	3.76	0.1726	210.6	1.066
49.50	165.0	1	2.65	1.00	0.58	3.16	0.2617	193.0	1.043
49.50	165.0	1	2.65	1.00	0.58	3.76	0.2602	254.7	1.034
15.85	190.0	1	2.65	1.30	0.58	2.50	0.1365	475.2	1.099
15.85	190.0	1	2.65	1.30	0.58	2.50	0.2899	630.9	1.035
15.85	190.0	1	2.65	0.12	0.58	3.00	0.1267	648.9	1.184
15.85	190.0	1	2.65	0.12	0.58	2.90	0.2775	866.7	1.081
5.07	520.0	1	2.65	1.00	0.7	1.90	0.0925	1175.6	1.310
5.07	520.0	1	2.65	1.00	0.7	2.00	0.2057	1763.4	1.202
4.00	580.0	1.25	2.27	4.00	0.65	2.88	0.1610	3926.0	1.056
4.00	580.0	1.25	2.27	4.00	0.65	2.70	0.1412	3580.0	1.133
4.00	580.0	1.25	2.27	4.00	0.65	2.01	0.1020	2217.0	1.177
4.00	580.0	1.25	2.27	4.00	0.65	1.05	0.0612	845.0	1.307
26.30	13000.0	1	2.65	1.00	0.8	3.20	0.0372	842.5	2.420
26.30	13000.0	1	2.65	1.00	0.8	4.00	0.0440	989.5	2.045

The weight, bias and final equations to calculate the hold up for any solid liquid slurry flow in pipelines are summarized in Table 5. After predicting the hold up accurately, the slurry velocity can be maintained to reduce this hold up to minimize the pressure drop and power consumption.

Table 4. Prediction error by hybrid ANN-DE based model

Parameter	Prediction performance by ANN	
	Training	Testing
AARE	0.024	0.025
Sigma	0.028	0.030
R	0.985	0.980
Optimum number of nodes	6	
Input activation function	Log sigmoidal function	
Output activation function	Linear	
Optimum learning rate	0.045	
Best training algorithm	Fletcher-Reeves update	

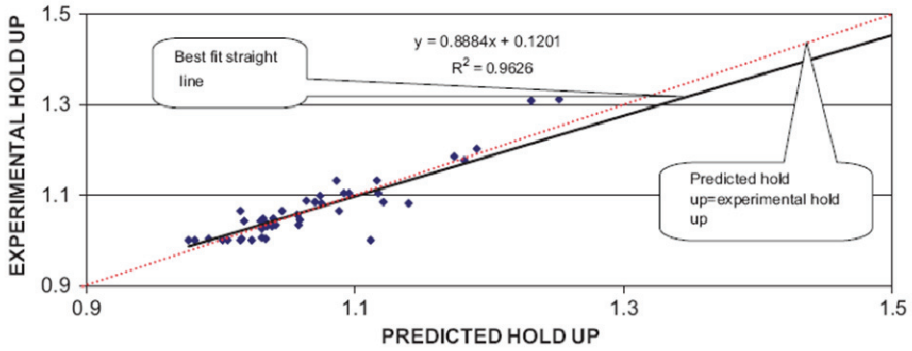

Fig. 7. Experimental vs. predicted hold up for Marquard Levenburg algorithm.

Table 5. Set of equations and fitting parameters for neural network correlations (i = 7, j = 8, k = 1)

$W_{i,j}$	1	2	3	4	5	6
1	0.5471	-0.1887	0.0812	-0.1319	-0.0427	0.2064
2	-0.3640	-0.4401	0.1656	0.9680	0.2904	0.1854
3	0.9943	2.3325	-2.6623	-4.2112	-1.6937	0.9411
4	-0.1299	0.5851	0.2368	-0.6478	1.0155	-0.5753
5	-0.7460	-0.7184	-0.5126	0.3785	-0.6674	-0.7037
6	-1.1726	1.2875	-1.7379	1.1721	-1.6191	-0.1233
7	2.7968	-0.3366	-2.5890	2.4545	3.2347	-3.9557
8	1.8683	-0.3992	3.2420	0.1343	2.2240	-0.5547
9	-1.4972	-2.5752	-2.3393	-2.4976	-1.7759	0.3350
10	-3.0206	2.4845	-2.3495	-1.7999	3.1200	4.4585

$W_{j,k}$	1	2	3	4	5	6	7
1	-0.8403	-2.1097	2.8565	0.1720	1.0012	-0.1895	0.9710

$$H_j = \frac{1}{1+\exp(-\sum_{i=1}^{i+1} w_{i,j} U_i)} \rightarrow S_l = \sum_{j=1}^{j+1} w_{j,k} H_j \rightarrow \text{Hold up ratio}$$

Comparison of hybrid ANN-DE model with ANN model

In a separate study, we exposed the same dataset to ANN algorithm only (without the DE algorithm) and tried to optimize different parameters based on the exhaustive search. We found out that it was not possible to reach the best solutions starting from arbitrary initial conditions. The optimum choice of the learning rate is especially very difficult to arrive at after starting with some discrete value. Many times the solutions got stuck up in sub optimal local minima. These experiments justified the use of a hybrid technique for ANN parameter tuning. The best prediction after the exhaustive search along with ANN parameters was summarized in Table 6. From the Table, it is clear that even after 100000 runs, the ANN algorithm

is unable to locate the global minima and the time of execution is 4 h in Pentium 4 processor. On the other hand, the hybrid ANN-DE technique is able to locate the global minima with 2000 runs within 1 h. The prediction accuracy is also much better. Moreover, it relieves the non-expert users to choose the different parameters and find optimum ANN meta parameters with a good accuracy.

Table 6. Comparison of the performance of ANN-DE hybrid model vs. ANN model

Parameter	Prediction performance by hybrid ANN-DE model	Prediction performance by ANN model only
AARE	0.025	0.030
Sigma	0.030	0.035
R	0.980	0.979
Execution time, h	1	4

Comparison with other published correlations

All the 800 experimental data collected from open literature were also exposed to different formulas and correlations for the hold up available in open literature and *AARE* were calculated for each of them (Table 7). From Table 7, it is evident that the prediction error of the hold up has reduced considerably in the present work.

Table 7. Performance of different correlations to predict the hold up

Sl no.	Author	*AARE*, %
1	Gillies [18]	25.46
2	Hsu [16]	22.01
3	Present work	2.5

CONCLUSION

ANN regression methodology with a robust parameter tuning procedure has been described in this work. The method employs a hybrid ANN-DE approach for minimizing the generalization error. Superior prediction performances were obtained for the case study of the hold up and a comparison with selected correlations in the literature showed that the developed ANN correlation noticeably improved the prediction of the hold up over a wide range of operating conditions, physical properties, and pipe diameters. The proposed hybrid technique (ANN-DE) also relieves the non-expert users to choose the meta parameters of ANN algorithm for his case study and finds out the optimum value of these meta parameters on its own. The results indicate that the ANN based technique with the DE based parameter(s) tuning approach described in this work can yield an excellent generalization and can be advantageously employed for a large class of regression problems encountered in process engineering.

Nomenclature

AARE - Average absolute relative error
R - Cross-correlation coefficient
σ - Standard deviation of error
w_{lk}^{O} - Weight between lth hidden node and kth output node
w_{nl}^{H} - Weight between nth input and lth hidden node
X_n - nth network input
y_k - kth network output
$\tilde{f}_1$, $\tilde{f}_2$ - Nonlinear activation functions
c_{ins} - *In situ* solid concentration
c_{dis} - Discharge solid concentration.

REFERENCES

[1] D.R. Baughman, Y.A. Liu, Neural Networks in Bioprocessing and Chemical Engineering, Academic Press, New York, 1995

[2] S. S Tambe, B. D Kulkarni, P. B. Deshpande, Elements of Artificial Neural Networks with selected applications in Chemical Engineering and Chemical and Biological Sciences, Simulations & Advanced Controls, Louisville, KY, 1996

[3] A. B. Bulsari, J. Syst. Eng. **4** (1994) 131-170

[4] K. Huang, X.L. Zhan, F.Q Chen, W. Lü, Chem. Eng. Sci. **58** (2003) 58-63

[5] G. Stephanopoulos, C. Han, Comp. Chem. Eng. **20** (1996) 743-791

[6] M. Riedmiller, H. Braun, Proc. of the IEEE Int. Conf. On Neural Networks, San Fransisco CA, Mar 28-Apr 1 (1993)

[7] J. Bandyopadhyay, S. Annamalai, K.L. Gauri, AIChE J. **42** (1996) 2295-2302

[8] N. Bhatt, T.J. McAvoy, Comp. Chem. Eng. **14** (1990) 573-582

[9] V. Venkatasubramanian, K. Chan, K, AIChE J. **35** (1989) 1993

[10] K. Price, R. Storn, Differential Evolution: Numerical Optimization Made Easy, Dr. Dobb's Journal, April 97, pp. 18-24

[11] K.K. N Sastry, L. Behra, I.J Nagrath, Fundamenta Informaticae, Special Issue on Soft Comput., (1998)

[12] B.V. Babu, K.K.N Sastry, Comp. Chem. Engg. **23** (1999) 327-339

[13] R. Hecht-Nielsen, Proceedings of the International Joint Conference on Neural Networks **1** (1989) 593-611

[14] R.P King, Introduction to practical fluid flow, Butterworth-heinemann, Burlington, MA01803, 2002

[15] D.M. Newitt, J.F Richardson,M. Abbott, R.B. Turtle, Trans. Inst. Chem. Eng., **33** (1955) 93-113

[16] F.L. Hsu, Flow of non-colloidal slurries in pipelines: Experiments and numerical simulations, PhD Thesis, University of Illinois, Chicago, IL, 1987

[17] P. Doron, D. Granica, D. Barnea, Int. J. of Multiphase Flow **13** (1987) 535-547

[18] R.G. Gillies, C.A Shook, Can. J. Chem. Eng. **78** (2000) 709-716

[19] G.W. Govier, K. Aziz, The Flow of Complex Mixtures in Pipes. Krieger, Malabar, FL, 1982

[20] D.R. Kaushal, Y. Tomita, Inter. J. Multiphase Flow **28** (2002) 1697-1717

[21] W. Schriek, L.G. Smith, D.B Hass, W.H.W Husband, report E73-10, Saskatchewan research council, Saskatoon, Canada, August, 1973

[22] B. Scarlett, A. Grimley, Proceedings of the 3rd International Conference on Hydraulic Transport of Solids, BHRA Fluid Engineering, Cranfield UK Paper **3** (1974) pp. 23

[23] M.C. Roco, C.A. Shook, PCH Physicochem. Hydrodyn. J. **2** (1982) 5-7

[24] R. Gillis, W.H.W Husband, M. Small, C.A. Shook, Proc 8th Technical Conference, Slurry Transport Association, (1983) 131-142.