Winter January 29, 2010

Modeling of Commercial Ethylene Oxide Reactor: A Hybrid Approach by Artificial Neural Network & Differential Evolution

Nadeem Muhammed Khalfe

Available at: http://works.bepress.com/nadeem_khalfe/17/
Modeling of Commercial Ethylene Oxide Reactor: A Hybrid Approach by Artificial Neural Network & Differential Evolution

Sandip K. Lahiri∗ Nadeem Khalfe†
Modeling of Commercial Ethylene Oxide Reactor: A Hybrid Approach by Artificial Neural Network & Differential Evolution

Sandip K. Lahiri and Nadeem Khalfe

Abstract

This study is motivated by a growing popularity of artificial neural network for process modeling and regression problems. However, many ANN regression application studies are performed by ‘expert’ users who have a 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 a difficulty in finding an optimum ANN architecture and are often confused about how to choose the ANN meta parameters. The present paper addresses this issue and develops a new hybrid procedure to find the optimum ANN architecture and tunes the ANN parameters, thus relieving the ‘non expert’ users. This method incorporates hybrid artificial neural network and differential evolution technique (ANN-DE) for efficient tuning of ANN meta parameters. The algorithm has been applied for modeling of a commercial ethylene oxide (EO) reactor. The model developed through this hybrid algorithm can predict the EO reactor output with only 0.4% error.

KEYWORDS: artificial neural network, ANN, differential evolution, ethylene oxide reactor
1. Introduction

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.

ANN modeling have 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 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. 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., Lahiri, S.K. and Ghanta K.C, 2008, Lahiri, S.K. and Khalfe N, 2009, Tambe et. al. 1996, Bulsari 1994, Huang 2003 and Stephanopoulos and Han 1996).

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-layered 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 such that the network, in response to the input patterns in an example set, accurately produces the corresponding outputs. There exists a number of algorithms—each possessing certain positive characteristics—to train an MLP network, for example, the most popular error-back-propagation (EBP), Quickprop and Resilient Bck-propagation (RPROP) (Reidmiller, 1993) 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 multiple training runs to obtain an optimal ANN model whose parameters (weights) 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.

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

This study is motivated by a growing popularity of ANN for 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 a optimum ANN architecture and confused about how to choose the ANN meta parameters. Whereas existing sources on ANN regression give some recommendations on appropriate setting of ANN parameters, there is clearly no consensus and plenty of contradictory opinions. Also there is lot of ANN algorithms available in literatures with their respective advantages. Some algorithms requires less computation time and computer storage requirement, some others have more accurate prediction capability. It is difficult for a non-expert users to choose the best (robust, accurate and fast) algorithm for his case. Present paper address this issue and develop 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 optimum ANN architecture and tuning of ANN meta parameters can be viewed mathematically as a optimization problem where test set errors (generalization error) has to be minimized. In the recent years, Differential Evolution (DEs) 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 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. DE has been used to design several complex digital filters (Price and Storn, 1997) and to design fuzzy logic controllers (Sastry et al., 1998). DE can also be used for parameter estimations e.g. Babu and Sastry, 1999 used DE for the estimation of 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 relieve the user from choosing these meta parameters but also find out the optimum values of these parameters to minimize the generalization error. In the present work, we illustrate ANN-DE approach by applying it for modelling of commercial ethylene oxide reactor.

The strategy (henceforth referred to as “ANN-DE”) use an ANN as the nonlinear process modeling paradigm, and the DE for optimizing the meta – parameters of the ANN model such that an improved prediction performance is realized. The paper is organized as follows: section 2 describes process modeling using ANN methods; the brief introduction of DE is given in section 3. The optimization of the ANN model using DEs is explained in section 4. Usage of ANN-DE strategies for case studies of commercial ethylene oxide reactor modelling along with the results described in section 5 and 6 respectively. Finally, section 7 gives a summary of the study.

2. Artificial neural network (ANN) modeling

Neural networks are computer algorithms inspired by the way information is processed in the nervous system.

2.1 Network Architecture

The MLP network used in the model development is depicted in Figure 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’s input layer is equal to the number of inputs in the process whereas the number of output nodes \( (K) \) equals the number of process outputs.
However, the number of hidden nodes ($L$) is an adjustable parameter whose magnitude is determined by issues, such as the desired approximation and generalization capabilities of the network model.

2.2 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.

2.3 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 input as well as desired outputs are known. Once trained, the network weights are frozen and can be used to compute output values for new input samples.

The feed forward process involves presenting an 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 goal is to find a set of weights that minimize 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 \( f : X \rightarrow \mathbb{Y} \). 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 \( W^H \) and \( W^O \) (they refer to the weights between input and hidden nodes, and hidden and output nodes, respectively). Thus, the nonlinear mapping can be expressed as

\[
y = y(x; W) = y(x; W)
\]

where, \( W = \{W^H, W^O\} \). This equation suggests that \( y \) is a function of \( x \), which is parameterized by \( W \) (Bishop21). It is now possible to write the closed-form expression of the input-output relationship approximated by the three-layered MLP as:

\[
y_k = \tilde{f}_2 \left[ \sum_{l=0}^{L} W_{lk}^O \tilde{f}_1 \left[ \sum_{n=0}^{N} W_{nl}^H x_n \right] \right]; k = 1,2,\ldots,K
\]

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 the outputs, it needs to be trained in a manner such that a pre specified error function is minimized. In essence, the MLP-training procedure aims at obtaining an optimal set \( W \) of the network weight matrices \( W^H \) and \( W^O \), which minimize an error function. The commonly employed error function is the average absolute relative error (AARE) defined as:

\[
AARE = \frac{1}{N} \sum_{1}^{N} \left| \frac{y_{\text{predicted}} - y_{\text{experimental}}}{y_{\text{experimental}}} \right|
\]
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). 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 $W^H$ and $W^O$ 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.

### 2.4 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 successively improve performance of the network on training data e.g. by “memorizing” training samples, but the resulting network may perform poorly on 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 (1990) 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.

### 2.5 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 hidden layer normally calculated by 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 input layer: Each hidden node and output node applies the activation function to its net input. Five types of activation function reported in literature and used in this work are shown in table 1 and figure 2. There is no consensus in literature which type of activation function is to be used and it depends on 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 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 large. This causes a problem when using steepest descent to train a multilayer network with sigmoid functions, since the gradient can have a very small magnitude; and therefore, cause small changes in the weights and biases, even though the weights and biases are far from their optimal values.

<table>
<thead>
<tr>
<th>Case</th>
<th>Name of activation function</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>log sigmoid function (logsig)</td>
<td>$y_i = \frac{1}{(1 + \exp(-net_i))}$</td>
</tr>
<tr>
<td>2</td>
<td>tan hyperbolic function (tansig)</td>
<td>$y_i = \tanh(net_i)$</td>
</tr>
<tr>
<td>3</td>
<td>linear function (purelin)</td>
<td>$y_i = (net_i)$</td>
</tr>
<tr>
<td>4</td>
<td>radial basis function (radbas)</td>
<td>$y_i = \exp(-net_i^2)$</td>
</tr>
<tr>
<td>5</td>
<td>triangular basis function (tribas)</td>
<td>$y_i = \begin{cases} 1 - \text{abs}(net_i) &amp; \text{if } -1 \leq (net_i) \leq 1 \ 0 &amp; \text{otherwise} \end{cases}$</td>
</tr>
</tbody>
</table>

Where $y_i$ is the output from node $i$, $net_i$ is the input to the node $i=\Sigma WiX_i$
3) The activation function of output layer: same remarks of 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 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 execution time and computer storage requirements. There are several different back propagation training algorithms published in literatures. Figure 3 shows some of those algorithms used in the present study. They have a variety of
different computation and storage requirements, and no one algorithm is best suited to all locations. The basic differences between these algorithms are how they handle the weight up-gradation in equation (1) and (2) to reduce error and how they modify learning rate ($\eta$) to reduce convergence time.

Most of the available ANN software’s requires the above four parameters as user inputs or calculated the above parameters on trial and error basis. This needs long execution time to explore all the combinations of above parameters to really find the best possible solutions. In the present paper we use differential evolution technique to find out the optimum solution.

3. 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 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) 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 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 the successful application of Differential Evolution 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 third randomly selected vector. The result is called the *noisy random vector*. Subsequently, 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 in appendix 1.
4. DE-based Optimization of ANN Models

There are different measures by which ANN performance is assessed, 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 training data but the ANN performance is estimated on test data.

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

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

\[ AARE = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_{\text{predicted, i}} - y_{\text{experimental, i}}}{y_{\text{experimental, i}}} \right| \]  \hspace{1cm} (4)

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

\[ \sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} \left[ \frac{y_{\text{pred, i}} - y_{\text{exp, i}}}{y_{\text{exp, i}}} \right]^2} \]  \hspace{1cm} (5)

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

\[ R = \frac{\sum_{i=1}^{N} (y_{\text{exp, i}} - \bar{y}_{\text{exp}})(y_{\text{pred, i}} - \bar{y}_{\text{pred}})}{\sqrt{\sum_{i=1}^{N} (y_{\text{exp, i}} - \bar{y}_{\text{exp}})^2} \sqrt{\sum_{i=1}^{N} (y_{\text{pred, i}} - \bar{y}_{\text{pred}})^2}} \]  \hspace{1cm} (6)

ANN learning is considered successful only if the system can perform well on 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 $\text{AARE}(\mathbf{X})$ on test set

$\mathbf{X} \in \{x_1, x_2, x_3, x_4, x_5\}$

where

$x_1 =$ Number of nodes in hidden layer $\{1, 2, \ldots, 100\}$

$x_2 =$ Input layer activation function $\{1, 2, 3, 4 \text{ and } 5 \text{ corresponds to five activation function in table 1}\}$

$x_3 =$ Output layer activation function $\{1, 2, 3, 4 \text{ and } 5 \text{ corresponds to five activation function in table 1}\}$

$x_4 =$ Learning rate $\{0 \text{ to } 5\}$

$x_5 =$ Training algorithm $\{1, 2, \ldots, 8 \text{ corresponds to eight training algorithm as per figure 3}\}$

The objective function is minimization of average absolute relative error (AARE) on test set and $\mathbf{X}$ is a solution string representing a design configuration of ANN architecture. The design variable $x_1$ takes any integer values for number of nodes in the range of 1 to 100, $x_2$ represents the Input layer activation function taking any values in the range of 1 to 5 corresponds to five activation function in table 1. $x_3$ represents the output layer activation function taking any values in the range of 1 to 5 corresponds to five activation function in table 1. $x_4$ represents learning rates and can take any value between 0 to 5. The variable $x_5$ takes eight values of the training algorithm which corresponds to eight ANN training algorithm in figure 3.

The total number of design combinations with these variables is $100 \times 5 \times 5 \times 5 \times 8 = 100,000$. This means that if an exhaustive search is to be performed it will take at the maximum 1000,000 function evaluations before arriving at the global minimum AARE for the test set (assuming 5 trials for to arrive optimum learning rate). So the strategy which takes few function evaluations is the best one. Considering minimization of AARE as the objective function, differential evolution technique is applied to find the optimum design configuration of ANN model. The methodology adopted is shown in figure 4.

5. Case study: Study of industrial ethylene oxide reactor

5.1. Background of EO reactor

The objective of the present case study is to model the industrial ethylene oxide reactor. The main purpose of modeling of EO reactor is to develop a mathematical equation which can predict the reactor outlet EO production, CO2 production and
catalyst selectivity if reactor inlet feed compositions and process parameters (temperature and pressure) is given. Unlike academic studies, this project is based on actual operating plant. That’s why the considerations and real issues to develop EO reactor models are quite complex and the constraints are complex to actual plant operations.

Oxidation of ethylene to produce ethylene oxide (EO) is an important reaction in the petrochemical industry for synthesis of glycol. Commercially EO is produced in shell and tube type EO reactor by reacting oxygen and ethylene at high temperature and pressure in presence of silver based catalyst. The oxidation of ethylene involves a main reaction producing EO and an undesirable side reaction producing carbon dioxide (CO2).

**Desired Main Reaction**

\[ \text{Ethylene} + \text{Oxygen} \rightarrow \text{Ethylene oxide} \quad \Delta H_r @ 25 \ C = -25,550 \text{ kcal/kg mol C}_2\text{H}_4 \]

**Undesired Side Reaction**

\[ \text{Ethylene} + \text{Oxygen} \rightarrow \text{carbon di oxide} + \text{water} \quad \Delta H_r @ 25 \ C = -316,200 \text{ kcal/kg mol C}_2\text{H}_4 \]

The performance of the reaction is measured by selectivity which is calculated by the percentage of ethylene used to produce EO as compared to total ethylene used to produce EO and CO2. Indirectly, selectivity measures the extent of the first reaction as compared to second reaction and it has a profound effect of overall economics of the glycol plant. The main objective of EO reactor modeling is to increase the selectivity by changing the process parameters. EO reactor is just like a shell & tube heat exchanger where silver catalyst was put as fixed bed in tube side. Water is circulated through the shell side to remove the heat of reaction as both the reactions are exothermic. The conversion of ethylene to EO is very low, that’s why ethylene and oxygen are recycled back as per figure 5. In industrial set up, ethylene and oxygen is mixed with the cycle gas (a mixture of gas comprising ethylene, oxygen, methane, carbon di oxide, nitrogen, argon, ethylene oxide etc) and heated before feeding to the EO reactor. In EO reactor, some of the ethylene and oxygen reacted over catalyst surface and produce EO and undesirable carbon di oxide and water. The reactor outlet gas is fed to EO absorber to practically absorb all the EO from cycle gas by a liquid absorbent. The EO free cycle gas then fed to CO2 absorber to absorb some of the carbon di oxide from cycle gas before recycle it back to the reactor by cycle gas compressor.
Read actual plant input and output data as $X_i$ and $Y_i$ respectively.

Segregate randomly the 80% of the records as training set and rest 20% as test set.

Develop ANN model and calculate weights that give lowest $AAE$ in test set.

Initialize generation index $N_{gen} = 0$.

Generate $NP$ random vectors as the initial population within user-defined bounds.

Apply these random vectors to ANN model to compute the output for each of the population. Find out the vector with the highest profit, i.e., the best vector so far.

Choose a target vector from the population of size $NP$.

Choose two vectors at random from the population and find the weighted difference. From that find the noisy random vector.

Perform crossover between target vector and random vector to find the trial vector.

Calculate the trial and the target vector profits by ANN model.

The vector with the greater profit replaces the population member in the initial population.

Update generation index $N_{gen} = N_{gen} + 1$.

If $N_{gen} < N_{gen_{max}}$ then go to step 3, else stop.

Top-ranked vectors constitute the optimal solution vector, $X^*_f$. Apply these optimum solution vectors to ANN model to find out the optimal output corresponding to maximum profit.
Economics of the process depends on the extent of first reaction as compared to second. The main objective of reactor optimization is to suppress the second reaction and promote first reaction to facilitate valuable ethylene to convert EO rather than wasting to CO2 and Water.

Inhibitors are added to control the reaction rate and improve the selectivity of catalyst. The industrial gas phase inhibitor is usually 1, 2 Dichloro ethane and its concentration usually 4-6 ppm. Dichloroethane inhibits the combustion reaction (second reaction) to a greater extent than the epoxydation reaction (first reaction). In this way it promotes the selectivity for EO. Excess quantity of this inhibitor though increase selectivity but reduce the production rate by occupying the active catalyst sites. Less than optimum quantity of inhibitor reduces selectivity and produces more carbon di oxide. Thus the optimum value of inhibitor concentration at reactor inlet is absolutely necessary for maximizing EO production. It is very difficult to adjust the inhibitor flow in the reactor without
knowing its concentration and panel operators in commercial plant usually adjust the inhibitor flow heuristically based on experience and watching other parameters such as steam flow, reactor outlet oxygen concentration etc. due to absence of reactor model. Online chloride analyzer is not available in the plant where this research work was carried out and this is the case in most of the commercial ethylene oxide plant around world. Offline samples are drawn once in a week or so and in the event of a process malfunction or operating under a suboptimal condition, the plant will continue to produce in non optimum fashion until lab results become available and corrective action taken.

In industrial EO reactor the catalyst selectivity is affected by many things such as concentration of catalyst poisons (such as acetylene) in cycle gas, catalyst age and its sintering effect, back history of catalyst usage, back history of water or other impurities carryover in reactor (it can damage some of the catalyst active site permanently) due to maloperation etc. That’s why it is very difficult for any phenomenological model to succeed in industrial scenario.

The increase of selectivity in operating plant is considered as a very difficult task as the quantitative effect of different process parameters on the selectivity is not known. The selectivity also drops with catalyst age due to sintering effect and permanent deactivation of active sites of catalyst.

A rigorous mathematical model for EO reactor is still not available in literature which can quantify all the process parameters effect on selectivity and can be used confidently in industry to optimize the EO reactor. The comprehensive reactor model is expected to take into account the various subjects, such as chemistry, chemical reaction and kinetics, catalysis and physics which consequently become very complex. Industry needs this mathematical model to optimize inhibitor flow and other process parameters so that selectivity is maximized. To develop such model from basic principles of chemical engineering is very difficult due to reaction kinetics and catalyst mechanism is unknown. Owing to the complex nature of industrial EO reaction processes, the underlying physico-chemical phenomenon is seldom fully understood. Difficulties associated with the construction and solution of phenomenological models necessitates exploration of alternative modeling formalisms.

This paper presents a systematic approach using robust hybrid ANN-DE techniques to build EO reaction correlation from available plant data. This correlation has been derived from a broad plant historical data bank.

5.2 Development of the ANN and DE based correlation

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

The quality and quantity of data is very crucial in ANN modeling as neural learning is primarily based on these data. Hourly average of actual plant operating data at steady state was collected for approximately one year. Data was checked and cleaned for obvious inaccuracy and retains those data when plant operation was in steady state and smooth. Finally 6273 records are qualified for neural regression. This wide range of database includes plant operation data at various capacities starting from 35% capacity to 110% of design capacity. Also these data captures plant operations at different age of catalyst life.

5.2.2 Identification of input and output parameters

The reactor performance was monitored in terms of three reaction output variables namely EO production (MT/Hr), CO2 production (MT/Hr) and catalyst selectivity (%). Based on the operating experience in glycol plant, all physical parameters that influence EO, carbon-di-oxide production and selectivity are put in a so-called ‘wish-list’.

Out of the number of inputs in ‘wish list’, we used ANN regression to establish the best set of chosen inputs, which describes reaction behavior. The following criteria guide the choice of the set of inputs:

- The inputs should be as few as possible.
- Each input should be highly cross-correlated to the output parameter.
- These inputs should be weakly cross-correlated to each other.
- The selected input set should give the best output prediction, which is checked by using statistical analysis [e.g. average absolute relative error (AARE), standard deviation, cross-correlation coefficient].
- There should be minimum complexity in neural network architecture, i.e. a minimum number of hidden layers.

The above mentioned criteria were then used to identify the most pertinent set of input groups. Based on the above analysis, the ten input variables (in table 2) have been finalized to predict EO production, CO2 production and catalyst selectivity.
5.2.3 Neural regression and DE based tuning

For modeling purposes, the reaction operating conditions data (see table 2) can be viewed as an example input matrix \( X \) of size (6273 X 10), and the corresponding reaction output data as the example output matrix \( Y \) of size (6273 X 3). For ANN training, each row of \( X \) represents a seven-dimensional input vector \( x = [x_1, x_2, ..., x_{10}]^T \), and the corresponding row of matrix \( Y \) denotes the three-dimensional desired (target) output vector \( y = [y_1, y_2, y_3]^T \). As the magnitude of inputs and outputs greatly differ from each other, they are normalized in 0-1 scales. To avoid ‘over training’ phenomena described earlier, 80% of total dataset was chosen randomly for training and rest 20% was selected for validation and testing.

Ten parameters were identified as input (table 2) for ANN and the EO Production, Selectivity and CO2 Production are put as target. These data then exposed to hybrid ANN-DE model described above. An advantage of the ANN-based modeling is that a comprehensive multiinput-multioutput (MIMO) model can be constructed for all the process outputs namely EO production \( y_1 \), catalyst selectivity \( y_2 \) and CO2 production \( y_3 \). While the training set was utilized for the EBP based iterative updation of the network weights, the test set was used for simultaneously monitoring the generalization ability of the MLP model. The MLP architecture comprised ten input \((N = 10)\) and three output \((K = 3)\) nodes.

For developing an optimal MLP model, its structural parameter, namely the number of hidden nodes \((L)\), activation functions in input and output layer, learning rate and ANN algorithm was varied systematically using DE. For choosing an overall optimal network model, the criterion used was least AARE for the test set. DE algorithm is used to find out the ANN structure which corresponds to least AARE.
Once an ANN based process model is developed, it can be used for process optimization to obtain the optimal values of the process input variables that maximize or minimize a specified objective function. Thus, it is possible to obtain the optimal values of process operating variables, which for instance, maximize reactant conversion and selectivity of the desired products, or minimize reactor temperature and the selectivity of undesired by-products.

6. Results and discussion

6.1 Prediction performance of hybrid ANN-DE model

After optimization of ten ANN parameters described above, the model output was summarized in table 3. Out of all the possibilities, Marquard Levenburg algorithm with ten number of nodes in hidden layer and tan hyperbolic and linear function in input and output layer has emerge out as the best solution (with lowest AARE) for the present case. The low AARE for EO production, selectivity and CO2 production are 0.4, 0.05 and 0.48% respectively may be considered as a excellent prediction performance considering the poor understanding of EO reaction phenomena and large databank for training comprising various capacity and different catalyst age. Figure 6 shows the goodness of the ANN-DE model prediction with actual plant data.
6.2 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 try to optimize the different parameters based on exhaustive search. We found that it was not possible to reach the best solutions starting from arbitrary initial conditions. Especially the optimum choice of learning rate is very difficult to arrive 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 exhaustive search along with ANN parameters was
summarized in table 4. From the table it is clear that even after 100,000 runs, the ANN algorithm is unable to locate the global minima and the time of execution is 4 hrs in Pentium 4 processor. On the other hand, the hybrid ANN-DE technique is able to locate the global minima with 2000 runs within 0.5 hr. The prediction accuracy is also much better. Moreover it relieves the non expert users to choose the different parameters and find an optimum ANN meta parameters with a good accuracy.

Table 3: Prediction error by hybrid ANN-DE based model

<table>
<thead>
<tr>
<th>Performance criteria</th>
<th>EO Production (Training Testing)</th>
<th>Selectivity (Training Testing)</th>
<th>CO2 Production (Training Testing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AARE</td>
<td>0.0040 0.0041</td>
<td>0.0005 0.0006</td>
<td>0.0048 0.0077</td>
</tr>
<tr>
<td>Sigma</td>
<td>0.0057 0.0058</td>
<td>0.0005 0.0006</td>
<td>0.0057 0.0067</td>
</tr>
<tr>
<td>R</td>
<td>0.999 0.998</td>
<td>0.999 0.998</td>
<td>0.999 0.998</td>
</tr>
<tr>
<td>Optimum number of nodes</td>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>Input activation function</td>
<td>tan hyperbolic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output activation function</td>
<td>Linear</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimum learning rate</td>
<td></td>
<td></td>
<td>0.042</td>
</tr>
<tr>
<td>Best training algorithm</td>
<td>Marquard</td>
<td></td>
<td>Levenburg</td>
</tr>
</tbody>
</table>
Table 4: Comparison of performance of ANN-DE hybrid model Vs ANN model

<table>
<thead>
<tr>
<th>Performance criteria</th>
<th>Prediction performance by Hybrid ANN-DE model</th>
<th>Prediction performance by ANN model only</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>testing</td>
<td>testing</td>
</tr>
<tr>
<td>AARE</td>
<td>0.004133</td>
<td>0.00751</td>
</tr>
<tr>
<td>Sigma</td>
<td>0.004367</td>
<td>0.00655</td>
</tr>
<tr>
<td>R</td>
<td>0.998</td>
<td>0.901</td>
</tr>
<tr>
<td>Execution time (hr)</td>
<td>0.5</td>
<td>4</td>
</tr>
</tbody>
</table>

7. 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 commercial EO reactor and a comparison with selected correlations in the literature (Aryana et al., 2009) showed that the developed ANN correlation noticeably improved prediction of EO/CO2 production and selectivity over a wide range of operating conditions. The proposed hybrid technique (ANN-DE) also relieve the non-expert users to choose the meta parameters of ANN algorithm for his case study and find out optimum value of these meta parameters on its own. The results indicate that ANN based technique with the DE based parameters tuning approach described in this work can yield excellent generalization and can be advantageously employed for a large class of regression problems encountered in process engineering.

Notations

\[
\begin{align*}
    y_k & : k\text{th network output} \\
    \tilde{f}_2, \tilde{f}_1 & : \text{nonlinear activation functions} \\
    w^o_{lk} & : \text{weight between } l\text{th hidden node and } k\text{th output node;} \\
    w^H_{nl} & : \text{weight between } n\text{th input and } l\text{th hidden node} \\
    X_n & : n\text{th network input.}
\end{align*}
\]
Appendix -1

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 $(NP \times D)$ random numbers and linearize the range between 0 and 1 to cover the entire range of the function. From these $(NP \times D)$ 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 ($X_i$) (a linear mapping rule can be used).

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

\[
\text{Weighted difference} = F \times (X_a - X_b)
\]

**Step 4** Find the noisy random vector: Generate a random number. Choose a third random vector from the population ($X_c$). Add this vector to the weighted difference to obtain the noisy random vector ($X'_c$).

**Step 5** Perform crossover between $X_i$ and $X'_c$ to find $X_t$, the trial vector: Generate $D$ random numbers. For each of the $D$ dimensions, if the random number is greater than $CR$, copy the from $X_i$ 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 cost is calculated, the vector that yields the lesser cost replaces the population member in the initial
population. In case profit is calculated, the vector with the greater profit replaces the population member in the initial population.

Step 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 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 a possible (Price and Storn, 1997).

References


