CHAPTER-5

5.0 NEURAL NETWORKS

Neural networks or artificial neural networks finds its applications in many disciplines. Neural networks are used in fields such as modeling time series analysis and control. The main advantage of artificial neural networks is their ability to learn from input data with or without a teacher. Neural networks basically represent biological nervous system and therefore have drawn motivation from the kind of computing performed by a brain. ANN is a network of complex interconnections consisting of a large number of processing elements called neurons. In this chapter models developed by ANN for steady state model of fuel cell and transient model of fuel cell.

5.1 NEURAL NETWORKS BACKGROUND

In 1943, Mc Cullach and Pitts developed artificial neural networks (ANNs). This was done in order understand human brain and it’s working. In the last decade more sophisticated algorithms were developed and using efficient computation tools. In the year 1989, Wassernabm has done extensive research to find out the potential of artificial neural networks as computational tools which acquire, represent and compute the mapping from one multivariate input space to another. ANNs can identify the relationships in given patterns and makes it possible to solve large-scale complex problems such as nonlinear modeling, pattern recognition, classification, association and control. In spite
of the work proposed by McCulloch and Pitts during 1943, the area of ANNs experienced a great change in last decade i.e., in 1982 due to Hopfield’s research in iterative auto association neural networks. Later Rumelthart et al (1986) engineered a mathematically rigorous theoretical framework for neural networks called back propagation algorithm. Since then ANNs have found a wide arena of applications in areas such as physics, neurophysiology, electrical engineering, biomedical engineering, acoustics, computer science, cybernetics, image processing, robotics, financing etc.,

5.2 INTRODUCTION TO ARTIFICIAL NEURAL NETWORK

ANN is basically an information processing system which has a parallel distributed construction. In the year 1994, Haykin proposed that ANN has certain performance characteristics resembling biological neural network of human brain. ANNs were developed based on the mathematical models of human cognition (or) neural biology.

Their development is based on the rules that:

1). Nodes also referred as units, cells or neurons are information processing elements

2). Connection links passes signals between nodes.

3). The weight of a connection link represents its connection strength.
4). Activation function is a nonlinear transformation applied to each node applied to its net input to determine the output signal.

A neural network is characterized based on its architecture which represents the pattern of connection between nodes, its method of determining the connection weights and the activation function. A typical ANN consists of a number of nodes that are organized according to a particular arrangement shown in figure 5.1. Neural networks can be classified depending on the number of layers: Single, bi-layer and multi-layer (most back propagation networks). ANNs can also be classified based on the direction of information flow and processing. If the nodes are arranged in layers, starting from a first input layer and ending at the final output layer it is called as feed forward network. There can also be multiple hidden layers with each layer having one

Figure 5.1: Three layer feed forward Network
(or) more nodes. Information passes from the input to the output side. The nodes in one layer are interconnected to those in the next but not to those in the same layer. Thus, the output of a node in a layer depends only on the inputs it receives from previous layers and the corresponding weights. ANNs in which information flows through the nodes in both directions from the input to the output side and vice – versa are called as recurrent ANNs. This can be generally obtained by feedback i.e., recycling previous network outputs as current inputs. If nodes within a layer are also connected then it is called lateral connections. Many a times, input variables of the input layer comes from the problem at hand. This consists of all quantities which can influence the output. Thus the input layers are transparent and hence provides a means of passing on information to the network. The output layer comprises of values predicated by the network and thus represents model output. Trial and error procedure is used to determine the number of hidden layers and the numbers of nodes in each hidden layer. Links connect the nodes within neighboring layers of the network. Figure 5.1 shows the schematic of a feed forward three layer ANN. These kinds of ANNs can be used for different types of problems, such as classifying patterns, storing and recalling data, grouping similar patterns or finding solutions to constrained optimization problems, performing general mapping from input pattern to output pattern. As shown in the figure, the system input vector comprising of a number of causal variables which influence system behavior is given by ‘X’, system output vector
consisting of a number of resulting variables which indicates the system behavior is given by \( Y \).

### 5.3 MATHEMATICAL ASPECTS

Figure 5.2 shows a schematic diagram of a typical \( j^{th} \) node. The inputs to such a node may generally obtained from system causal variables (or) outputs of other nodes depending on the layer that the node is located in these inputs form an input vector \( X=(x_1,...,x_i,...,x_n) \). The sequence of weights leading to the node forms a weight vector \( W_j = (w_{ij}...w_{ij}...w_{nj}) \). Where, \( w_{ij} \) gives the connection weight from the \( i^{th} \) node in the preceding layer to \( j^{th} \) node. The output of node \( j \), \( y_j \) can be obtained by calculating the value of function with respect to the inner product of vector \( X \) and \( w_1 \) minus \( b_j \). Where, \( b \) represents the threshold value, also called the bias, associated with the corresponding node. In ANN, if the bias \( b_j \) of the node is exceeded it gets activated. The following equation explains the operation.

\[
y_j = f(X \cdot W_j - b_j) \quad \text{------------- (5.1)}
\]
Activation function is given by the function $f$. Its functional form is determined by the response of a node to the total input signal it receives. Sigmoid function is the most commonly used form of $f(.)$ in eqn (2.1) given as.

$$f(t) = \frac{1}{1 + e^{-t}}$$  \hspace{1cm} (5.2)

The sigmoid function is a monotonic, non decreasing, bounded, which provides a graded, nonlinear response. Any non linear process mapping can be enabled by a network with this function. The simplicity of sigmoid functions derivative that will be used during training process adds to the popularity of the sigmoid function. Bipolar sigmoid and hyperbolic tangent as activation functions-both of which are transformed from the sigmoid function and used by some researchers. A number of such nodes are arranged to form a artificial neural network.
5.4 NETWORK TRAINING

In order for an ANN to develop an output vector $Y = (y_1, y_2, \ldots, y_p)$ that is as close as possible to the target vector $T = (t_1, t_2, \ldots, t_p)$, a training process, also called learning, is used to find optimal weight matrices $W$ and bias vectors $V$, that minimize a predetermined error function that usually has the form.

$$E = \sum \sum (Y_i - t_i)$$

Here $t_i$ indicates the component of required output $T$, $Y_i$ is the respective ANN output, $p$ represents the number of the output nodes and $P$ represents the number of training patterns. The process in which the connection weights of an ANN are adapted through a continuous process of stimulation by the environment in which the network is included is called as training. Training or learning can be classified as supervised and unsupervised. If the training algorithm requires an external teacher to guide the training process, then it is called Supervised training. This indicates that a large number of examples (or) patterns of inputs and outputs are needed for training. The outputs are the effect variables and inputs are cause variables of a system. This training procedure includes the iterative adjustment and optimization of connection weights and threshold values for each of nodes. Minimizing the error function by searching for a set of connection strengths and threshold values, which causes the ANN to produce outputs, which are equal (or) close to targets is the primary goal of training. Once training has been finished, it is understood that
the ANN has the ability to produce reasonable results in new inputs are given. On the other hand, an unsupervised training algorithm does not need a teacher. During training, only an input data set is provided to the ANN which automatically adapts its connection weights to cluster those input patterns into classes with similar properties.

The algorithm for training the neural network is as follows:

1) Apply the input vector to the input units.
2) Compute the net input values of the hidden layer units
   \[ \text{Net}_{jh} = \sum W_{ji} x_i + \theta_{jh} \quad \text{for } I = 1,2,\ldots,n. \]
   Where \( W_{ji} \) represents the connection weight and \( \theta_{jh} \) represents the bias value.
3) Compute the outputs from the hidden layers:
   \[ I_j = f_j^h(\text{net}_{jh}) \]
4) Proceed towards the output layer. Compute the net input values to each unit.
   \[ \text{Net}_{k} = \sum W_{kj} I_i + \theta_{k} \]
5) Compute the outputs:
   \[ O_k = f_k^o(\text{net}_k) \]
6) Compute the error terms for the output units:
   \[ \delta_k^o = (y_k - o_k)f_k^o(\text{net}_k) \]
7) Compute the error terms for the hidden terms:
\[ \delta^h_{j} = f'(h_j) \sum \delta^o_{k} w^o_{kj} \]

It can be observed that the error terms on the hidden units are computed before the connection weights to input layer units have been updated.

8) The input layer weights are updated:
\[ W^o_{kj}(t+1) = W^o_{kj}(t) + \eta \delta^o_{kj} \]

9) The hidden layer weights are updated:
\[ W^h_{ji} = W^h_{ji}(t) + \eta \delta^h_j x_i \]

5.5 BACK – PROPAGATION ALGORITHM

The most popular algorithm for training ANNs is back – propagation algorithm. It is basically a gradient descent method which minimizes the network error function- equ (5.3). Each input pattern of the training data set is passed through the network from the input layer to the output layer. The network output is compared with the desired target output, and the value of error is calculated based on equ (5.3). This error is propagated backward through the network to each node and correspondingly the connection weights are updated based on the equation:

\[ \Delta W_{ij}(n) = -\epsilon \frac{\partial E}{\partial W_{ij}} + \alpha \Delta W_{ij}(n-1) \]  

\[ \text{---------------------(5.4)} \]
For connection of bias values a similar equation can be written. In eqn (5.4), and are called learning rate and momentum, respectively. Oscillations in the weights can be prevented by the momentum factor which can speed up training in very flat regions of the error surface. A learning rate helps in increasing the chance of avoiding the training processes being trapped in local minima rather than global minima. The back propagation algorithm involves two-steps. The effect of the input is passed forward through the network to reach the output layer in the first step i.e., forward pass. A second step starts backward through the network after computation of the error. The errors at the output layer are propagated back towards the input layer with the weights being modified according to equ (5.4). Back-Propagation is a first order method based on steepest gradient descent and the direction vector is set equal to the negative of the gradient vector. As a result, the solution mostly follows a zigzag path while trying to reach a minimum error position, which may slow down the training process. Sometimes, it is also possible for the training process to be trapped in the local minimum even though learning rate is used. To design and control a fuel cell system (FCS) for the maximum power performance, a designer must have sufficient knowledge related to the internal structure of the process, physical process and as well as the dominant input/output variables of the system. Accordingly, an accurate mathematical representation of the system may be developed which is sufficient to accurately reflect the behavior of the physical process. Several one-dimensional (1D) and multi-dimensional (MD) models have been developed to explain the
electrochemical and thermodynamic phenomena inside the fuel cells during the last decade. However, some of these models may need specific knowledge of parameters like resistance and membrane thickness which can be either unknown or known to the manufacturers. Therefore, to achieve FCS optimum performance the availability of the electrochemical equations or models may not be sufficient for accurate design. In addition, for large-scale FCSs these models as described above are commonly very complicated. On the other hand, in most of control applications, the designer may be interested in the internal structure of the system as well as relationship between inputs and outputs. Such knowledge will enable the designers with the sufficient tool to control the inputs in order to reach the desired outputs were stack voltage and stack current for appropriate application. Artificial neural networks can perform such a prediction. Investigate the reliability of the BP networks for the output prediction of an 18W FCS. The 18W fuel cell BP networks are explained and the reliability of the constructed neural networks to predict the performance of the FCS is studied. In addition, a comparison of the BP networks in term of network architectures, error goal and training algorithms have been investigated. Finally, the results and conclusions are discussed.
5.6 ARTIFICIAL NEURAL NETWORKS 18W 4CELL FUEL CELL

5.6.1. FCS

The 18W FCS comprises of a fuel cell stack as well as all the auxiliary equipment such as temperature monitor, current monitor, voltage monitor, hydrogen leakage detector, hydrogen delivery and oxygen delivery necessary for fuel cell operation. The stack voltage and current ranges from 4.5V/1A at idle to 2.5V/8A at full load. Regulator valve regulates the hydrogen pressure.

5.6.2. LOAD BANK

The load bank consisting of 3.8VDC light 9 bulbs system each 2w is developed. The load bank can increase the stack current up to 7A. This load is inexpensive, simple and more reliable.

5.6.3. DATA COLLECTION AND ANALYSIS

5.6.3.1 DATA COLLECTION

The FCS was operated with the load bank up to hydrogen pressure up to 1 bar and maximum current of 7A. Neural network algorithm is trained by taking 60 data points. The data sets such as H₂ pressure (bar), stack voltage (V), stack current (A) and all variables are constant.

5.6.3.2 SELECTION OF PROCESS VARIABLES

It is necessary to select the appropriate variables like network inputs and outputs in order to train the network successfully and efficiently. For
selecting the reliable variables, one need to understand the process, how each variable affects the system performance and then identify which variables are dominant and which variables can be discarded. Network will become unnecessarily complicated and the training may be difficult or take too long time to succeed if there are too many variables as network inputs. In contrast, the network will be small and can provide the fastest training and instant recall if the fewest dominant variables are selected. In the present system, there are several variables to be selected as inputs/outputs for the proposed neural networks. Current of the stack is varied by using load bank and Hydrogen pressure is varied by using a regulated valve. Discard the other input variables which have been forced to be constant by the system that are mass air flow and stack temperature (°C) as the dominant input variables. Stack voltage (\( V \)) is selected as the output variable because of its obvious relation to the fuel cell power generation.

5.6.3.3 RANGE OF DATA

In general, neural network performance is better in interpolation rather than extrapolation; therefore, the recall data should be in the range of training data, in order to obtain a better prediction. We randomly select the collected data to cover the range from minimum to maximum values as training data and the remaining data as recall data. Hence this approach makes sure that the recall data will always lie in the range of training data. The ranges of inputs/outputs data sets are as follows.
5.6.3.4. RANGES OF INPUTS

The range inputs selected based on the importance of the output result. Here selected two input ranges one is hydrogen pressure is in the range of 0.2 to 1 bar and second one is stack current (A) is in the range of 1 to 8A.

5.6.3.5. RANGES OF OUTPUTS

The range of outputs depends on application accuracy. So according to accuracy range will be selected. In this Stack voltage is in the range of 2.0 to 4.5V.

5.6.3.6. SIZE OF TRAINING AND RECALL DATA

When training data set is presented to the network, the weights and biases are updated on a pattern-by-pattern basis till the entire training data set is finished which is known as one epoch. This training phase is repeated till the network performs well to meet the error goal as needed by the designer. Accordingly, the recall data is presented to make sure that the network has learned the general patterns, instead of simply memorizing the data set. Training is said to be completed if the network still performs well. Training data set must be fairly large and must include a variety of data in order to contain all the needed information. Therefore, in our investigation, from 60 data points collected, 20 data points is for recall phase and 40 data points is for training phase.
5.6.3.7. PREPROCESS DATA

Preprocess data needs two additional data sets; normalized data. In the subsequent sections, we will find out which data set provides a faster and better prediction. By using base value selection the raw data will be normalized to have a range of [0, 1].

5.7. MODELING

Inspired by the biological neural networks, an ANN is a massively parallel distributed processor consisting of simple processing units, known as neurons. Since ANNs has the ability to learn from input data with or without a teacher, modeling and training will train the network. In this section, investigation is done on how well the BP networks can predict the performance of the 18W FCS.

5.7.1. BP NETWORKS ARCHITECTURE

A BP network consisting of one hidden layers is constructed. The number of neurons in hidden layer is varied to investigate the network prediction performance. The input layer has two input variables which are stack current and hydrogen pressure. The output layer has only one neuron for stack voltage. In this work considered two layer feed forward network, Two inputs, one hidden layer and one output, 13 hidden nodes, Sigmoid function, input matrix size(2,60), output matrix size(1,60).
5.7.2. IMPLEMENTATION

The following three criteria’s are selected to investigate the prediction performances of the BP networks for 18W FCS:

(a) Number of epochs(10000): they indicates the training speed.

(b) Root mean square error ($E_2$): this indicates average error of the prediction.

(c) Learning Rate(0.1): this indicates the speed.

5.8. TEST CASES AND SIMULATION RESULTS

Test cases results by varying the load current and hydrogen pressure of the stack are shown in table 5.1 for both the practical and ANN model. Fig. 5.1 shows the graphs of voltage (V) versus the current density (A/sq. cm) for various hydrogen pressures. Using weight matrix of $w_1(13,2)$ and $w_2(1,13)$, 20 testing patterns are tested.

5.8.1. DATA SET SELECTION

The time expenditure and accuracy for the FCS are shown in table 5.1. These data sets are given to the BP network whose bias values and weights are updated using the BP algorithm with the error goal of 0.001. The fastest in training phase. The average error ($E_2$) and maximum error ($E_\infty$) using these three data sets are not so different. The predictions of stack voltage for the FCS are shown. The results show the satisfactory predictions for the entire operating range except at the initial phase. It is observed that, at the starting
phase of the FCS, the inputs are almost constant even though the outputs are changing due to transient response of the feedback system at the initial phase. Therefore, the NN prediction does not perform well at the initial phase.

5.8.2 ERROR GOALS

In the training phase, the network will adjust its biases and weights till the output error reaches the designated error goal. If the selected error goal is relatively too large, then the training will be completed in a shorter time. However, a large error is achieved in the recall phase. On the other hand, if the error goal is selected to be too small, it will take a very long time. The speed and accuracy of error goals 0.1 and 0.01 are too large while the error of 0.0001 is too small. BP network is used to predict the performance of the 18w FCS. Stack voltage predictions are done by using load currents and hydrogen pressure.

5.8.3 CODE ALGORITHM FOR TEST CASE 1

The following procedure explains about program code and functions. Here we have three important main MATLAB ANN functions. Apart from training parameter play major roles, approximately 4 training parameters are considered figure 5.3.
Figure 5.3: 18w 4cell stack voltage curves at various pressures and loads

The function initff is used to initialize the inputs and it also selects the activation functions. This function will provide initialized weight matrices and biases matrices. Another function is trainbpx this function trains the pattern by using architecture and it gives generalized biases and generalized weight matrices. By using generalized matrices and testing input simuff function calculates the actual outputs of ANN model. The table 5.1 gives test results of the 18w 4cell fuel cell stack.

Inputs: Pressure variation
load current variation

Outputs: Stack Voltage
Input data range=[0.1 2; 0.5 10];

[w1,b1,w2,b2]=initff (input data range, 13, 'tansig', 1, 'tansig')

validation=1000;

epochs =230000;

learning rate=0.01;

error goal=0.03;

TP=[df me lr eg];

[W1,b1w2,b2, TE]=trainbpx (w1,b1, 'tansig', w2,b2, 'tansig', input data, target data, training parameters)

output=simuff (test data,w1,b1, 'tansig', w2,b2, 'tansig')

Table 5.1 18w 4cell stack ANN model voltage

<table>
<thead>
<tr>
<th>Load (watts)</th>
<th>Current (Amps)</th>
<th>H₂ pressure 0.2 bar (Volts)</th>
<th>H₂ pressure 0.4 bar (Volts)</th>
<th>H₂ pressure 0.5 bar (Volts)</th>
<th>H₂ pressure 0.6 bar (Volts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>2.6000</td>
<td>2.9000</td>
<td>2.8000</td>
<td>2.8900</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2.5700</td>
<td>2.7000</td>
<td>2.6700</td>
<td>2.7700</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>2.5680</td>
<td>2.6000</td>
<td>2.4500</td>
<td>2.5700</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>2.4500</td>
<td>2.5000</td>
<td>2.5100</td>
<td>2.4580</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>2.2900</td>
<td>2.4000</td>
<td>2.3400</td>
<td>2.4400</td>
</tr>
<tr>
<td>12</td>
<td>5.7</td>
<td>2.1900</td>
<td>2.2300</td>
<td>2.2300</td>
<td>2.3000</td>
</tr>
<tr>
<td>14</td>
<td>6.6</td>
<td>2.0000</td>
<td>2.1000</td>
<td>2.1000</td>
<td>2.2000</td>
</tr>
<tr>
<td>16</td>
<td>7.5</td>
<td>2.0000</td>
<td>2.0000</td>
<td>2.0000</td>
<td>2.0000</td>
</tr>
<tr>
<td>18</td>
<td>7.7</td>
<td>2.0000</td>
<td>2.0000</td>
<td>2.0000</td>
<td>2.0000</td>
</tr>
</tbody>
</table>
5.8.4 CODE ALGORITHM FOR TEST CASE 2

In this test case 2, the patterns generated from simulation model. Here after training validated the ANN model with simulation model. In this case study training patterns 500 generated from tables 4.1-4.3 and trained the network. The final test results shown in table 5.2.

Table 5.2 Case 2 ANN model output results

<table>
<thead>
<tr>
<th>H2com %</th>
<th>O2com %</th>
<th>H2flow l/min</th>
<th>O2flow l/min</th>
<th>Stemp oC</th>
<th>H2pre bar</th>
<th>O2pre bar</th>
<th>Voltage volts</th>
<th>H2uti %</th>
<th>O2uti %</th>
<th>Seff %</th>
<th>Spow watts</th>
</tr>
</thead>
<tbody>
<tr>
<td>97</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>95</td>
<td>10</td>
<td>40</td>
<td>12</td>
</tr>
<tr>
<td>97</td>
<td>21</td>
<td>1</td>
<td>1</td>
<td>31</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>95</td>
<td>12</td>
<td>41</td>
<td>12</td>
</tr>
<tr>
<td>97</td>
<td>21</td>
<td>1</td>
<td>2</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>95</td>
<td>13</td>
<td>42</td>
<td>13</td>
</tr>
<tr>
<td>97</td>
<td>22</td>
<td>1</td>
<td>2</td>
<td>33</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>96</td>
<td>15</td>
<td>42</td>
<td>13</td>
</tr>
<tr>
<td>97</td>
<td>22</td>
<td>1</td>
<td>3</td>
<td>34</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>96</td>
<td>16</td>
<td>43</td>
<td>13</td>
</tr>
<tr>
<td>97</td>
<td>23</td>
<td>2</td>
<td>3</td>
<td>35</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>96</td>
<td>18</td>
<td>44</td>
<td>14</td>
</tr>
<tr>
<td>97</td>
<td>23</td>
<td>2</td>
<td>4</td>
<td>36</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>96</td>
<td>19</td>
<td>45</td>
<td>14</td>
</tr>
<tr>
<td>98</td>
<td>24</td>
<td>2</td>
<td>4</td>
<td>37</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>96</td>
<td>21</td>
<td>45</td>
<td>14</td>
</tr>
<tr>
<td>98</td>
<td>24</td>
<td>2</td>
<td>5</td>
<td>38</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>97</td>
<td>22</td>
<td>46</td>
<td>14</td>
</tr>
<tr>
<td>98</td>
<td>25</td>
<td>2</td>
<td>5</td>
<td>39</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>97</td>
<td>24</td>
<td>47</td>
<td>15</td>
</tr>
<tr>
<td>99</td>
<td>25</td>
<td>2</td>
<td>6</td>
<td>40</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>97</td>
<td>25</td>
<td>48</td>
<td>15</td>
</tr>
<tr>
<td>99</td>
<td>26</td>
<td>2</td>
<td>6</td>
<td>41</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>97</td>
<td>27</td>
<td>48</td>
<td>15</td>
</tr>
<tr>
<td>99</td>
<td>26</td>
<td>2</td>
<td>6</td>
<td>42</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>97</td>
<td>28</td>
<td>49</td>
<td>16</td>
</tr>
<tr>
<td>99</td>
<td>27</td>
<td>2</td>
<td>7</td>
<td>43</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>98</td>
<td>30</td>
<td>50</td>
<td>16</td>
</tr>
<tr>
<td>99</td>
<td>27</td>
<td>2</td>
<td>7</td>
<td>44</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>98</td>
<td>31</td>
<td>51</td>
<td>16</td>
</tr>
<tr>
<td>99</td>
<td>28</td>
<td>3</td>
<td>8</td>
<td>45</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>98</td>
<td>33</td>
<td>51</td>
<td>17</td>
</tr>
<tr>
<td>99</td>
<td>28</td>
<td>3</td>
<td>8</td>
<td>46</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>98</td>
<td>34</td>
<td>52</td>
<td>17</td>
</tr>
<tr>
<td>100</td>
<td>29</td>
<td>3</td>
<td>9</td>
<td>47</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>98</td>
<td>36</td>
<td>53</td>
<td>17</td>
</tr>
<tr>
<td>100</td>
<td>29</td>
<td>3</td>
<td>9</td>
<td>48</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>99</td>
<td>37</td>
<td>54</td>
<td>17</td>
</tr>
<tr>
<td>100</td>
<td>30</td>
<td>3</td>
<td>10</td>
<td>49</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>99</td>
<td>39</td>
<td>54</td>
<td>18</td>
</tr>
<tr>
<td>100</td>
<td>30</td>
<td>3</td>
<td>10</td>
<td>50</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>99</td>
<td>40</td>
<td>55</td>
<td>18</td>
</tr>
</tbody>
</table>
Inputs: Hydrogen composition(97%, 99.99%)

Oxidant composition(20%, 30%)

Hydrogen flow rate(1lpm, 3lpm)

Air flow rate(1lpm, 10lpm)

System temperature(30°C, 50°C)

Hydrogen supply pressure(.1bar, 2bar)

Air supply pressure(.1bar, 4bar)

Outputs: Stack Voltage(.9volts, 5volts)

Utilization of Hydrogen(95%, 99%)

Utilization of Oxygen(10%, 40%)

Stack Efficiency(40%, 55%)

Stack Power(12watts, 18watts)

\[ P = [\text{inputs(minimum range, maximum range)}] ;\]

\[ [w1,b1,w2,b2]=\text{initff (input data range, 13, 'tansig', 1, 'tansig')} \]

validation=1000; epochs = 230000; learning rate=0.01; error goal=0.03;

TP=[df me lr eg]; W1,b1w2,b2, TE=\text{trainbpx (w1,b1, 'tansig', w2,b2, 'tansig', input data, target data, training parameters)}

output=\text{simuff (test data, w1,b1, 'tansig', w2,b2, 'tansig',)}