CHAPTER 4

AN OPTIMAL APPROACH IN MAXIMUM POWER POINT TRACKING USING NEURAL NETWORK

4.1 INTRODUCTION

The non-linear variation of voltage and the current with solar radiation levels, aging and load current are the prime reasons for the low electrical efficiency of photovoltaic systems. Tsai-Fu Wu et al (1999) proposed an optimum point of operation, of PV modules, with the initial reference values fixed arbitrarily and without constraints. This reduced the performance of the optimal operation point of PV systems. In order to improve this performance, a MPPT method based on a nonlinear approach is proposed for estimation of the initial value. For prediction of maximum power generation, a three layer artificial neural network was proposed by Ires Iskender et al (2003). The input signals are voltage, irradiation and current. The output signal is the predicted maximum power from the PV module. For training, data sets of the proposed neural network are selected from each season namely summer, winter, spring and autumn. The neural network has been trained by using data set corresponding to each season.

Zhang et al (2000 and 2002) presented a novel genetic algorithm trained radial basis function neural network-based model to carry out the MPPT for grid-connected PV power generation control systems. The hidden layer of the neural network was self-organised. The system was trained to predict the maximum power point of a PV array using measured
environmental data. The neural network was used to identify the optimal operating voltage of the PV system. Takashi Hiyama et al (1995) proposed a system in which the controller generated a control signal in real time and the control signal was fed back to the voltage control loop of the inverter to shift the terminal voltage of the PV system to the specified optimal level which yielded the maximum power generation. The continuous measurement was required for the open circuit voltage. The PC based controller gave results with time delay. To overcome this time delay, sigmoidal activation function is used in this thesis to track the maximum power point voltage.

4.2 ARTIFICIAL NEURAL NETWORK - AN OVERVIEW

Many alternative ways of integrating neural network have been proposed in the scientific literature. The first artificial neural network (ANN) implementation dates over 50 years back. Most research dealt with learning techniques and algorithms. One major milestone in the development of neural network technology was the invention of the so-called error back propagation algorithm about a decade ago.

In PV system power generation, the tracking problem arises due to noise and others resulting from non-idealities such as measurement bias. Al-Atrash and Rustom (2005) discussed the different noise fighting techniques, in order to predict their validity and optimum utilization of power from the solar arrays which considerably reduces the size and weight of the system.

The ANN is one of the supervised learning networks discussed by Laurene Fausett (2001), which consist the following distinct features:
1. Artificial neural network operates like a black box model, requiring no detailed information about the system.

2. They learn the relationship between input and output variables by studying the previously recorded data.

3. Ability to handle large and complex systems with many interrelated parameters.

4. These trained ANN is used to approximate an arbitrary input-output mapping of the system.

5. The numbers of nodes in each layer varies and are user dependent.

6. Among the available training algorithms, the back-propagation algorithm is most widely used, as it is stable, robust and easy to implement.

4.2.1 Back-Propagation Neural Network

This research is implemented with Back-propagation (BP) model. It is most popular in the supervised learning architecture because of the weight error correct rules. It is considered a generalization of the delta rule for nonlinear activation functions and multilayer networks.

In a back-propagation neural network, the learning algorithm has two phases. First, a training input pattern is presented to the network input layer. The network propagates the input pattern from layer to layer until the output pattern is generated by the output layer. If this pattern is different from the desired output, an error is calculated and then propagated backward through the network from the output layer to the input layer. The weights are
modified as the error is propagated. Insung Jung et al (2007), represented a step of the back-propagation training algorithm with an iterative gradient designed to minimize the mean square error between the actual output of multi-layer feed forward perceptron and the desired output.

4.2.2 Learning by Error Back Propagation

The error back propagation algorithm soon became the standard for most neural network implementation due to its high performance in accordance with Freeman and Skapura (2000). First, it selected one of the examples of the training data set. Second, it computed the neural net output values for the current training example inputs. Next it compared these output values with the desired output value of the training example. The difference, called error, determined which neuron in the network is modified. The mathematical mapping of the error back into the neurons of the network is called error back propagation.

4.3 NEURAL NETWORK ARCHITECTURE

The architecture of a multilayer neural network with one layer of hidden units (Z units) proposed by Zurada (1997) is shown in Figure 4.1. The output units (the Y units) and the hidden units have biases. The bias on a typical output unit $Y_k$ is denoted by $w_{ok}$; the bias on a typical hidden unit $Z_j$ is denoted $v_{oj}$. These bias terms acts like weights on connections from units whose output is always 1. Only the direction of information flow for the feed forward phase of operation is shown. During the back propagation phase of learning, signals are sent in the reverse direction.
4.3.1 Training Algorithm

As mentioned by Jang et al (1997), training a network by back propagation involves three stages: the feed forward of the input training pattern, the back propagation of the associated error, and the adjustment of the weights.

During feed forward, each input unit (Xi), receives an input signal and broad casts this signal to the each of the hidden units Z1,……, Zp. Each hidden unit then computes its activation and sends (zj) to each output unit. Each output unit (Yk) computes its activation (yk) to form the response of the net for the given input pattern.
During training, each output unit compares its computed activation \( y_k \) with its target value \( t_k \) to determine the associated error for that pattern with that unit. Based on this error, the factor \( \delta_k \) (\( k=1,\ldots, m \)) is computed. \( \delta_k \) is used to distribute the error at output unit \( Y_k \) back to all units in the previous layer (the hidden units that are connected to \( Y_k \)). It is also used (later) to update the weights between the output and the hidden layer. In the similar manner, the factor \( \delta_j \) (\( j=1,\ldots, p \)) is computed for each hidden unit \( Z_j \). It is not necessary to propagate the error back to the input layer, but \( \delta_j \) is used to update the weights between the hidden layer and the input layer.

After all of the \( \delta \) factors have been determined, the weights for all layers are adjusted simultaneously. The adjustment to the weights \( w_{jk} \) (from hidden unit \( Z_j \) to output unit \( Y_k \)) is based on the factor \( \delta_k \) and the activation \( z_j \) of the hidden unit \( Z_j \). The adjustment to the weights \( v_{ij} \) (from input unit \( X_i \) to hidden unit \( Z_j \)) is based on the factor \( \delta_j \) and the activation \( x_i \) of the input unit.

### 4.4 NETWORK OF THE SOLAR PANEL

Bekker (2004) proposed the photovoltaic panel power-voltage curve scanned periodically, and the maximum power point tracker was controlled by current. Hussein et al (2002) proposed an adaptive neural network controller (ANNC) used to track the maximum power point (MPP) of the PV generator by controlling the converter duty ratio. Problems with this implementation are due to sudden variation in solar radiation. In this proposed thesis, the maximum power point voltage and current is obtained by training Artificial Neural Network. Figure 4.2 shows the block diagram of neural network in which the current and voltage are taken as input parameters from the PV source at given insolation. The basic structure of a three layer feed forward neural network is shown in Figure 4.3. This neural network has input layer, hidden layer and output layer respectively.
Figure 4.2 Neural Network Block Diagram

Figure 4.3 Basic Structure of Feed Forward Neural Network

Back propagation training algorithm is most commonly used in feed forward ANN. When a set of input values are presented to the ANN, step by step calculations are made in the forward direction to drive the output pattern. The relationships between the maximum power point and circuit variables such as open circuit voltage and short circuit current were examined under various conditions appearing in the PV power generation system. Mutoh et al. (2002) discussed the relationship between the maximum power and the current that the output power was almost a linear function in the actual solar arrays, regardless of the weather conditions. Squared difference between the actual output and the desired output for the set of input patterns
was generated and this was minimized by gradient descent method altering
the weights one at a time starting from the output layer.

Table 4.1 represents the number of neurons or units in each layer of
the network which is represented in Figure 4.3.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Number of neurons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layer</td>
<td>2 + bias</td>
</tr>
<tr>
<td>Hidden layer</td>
<td>25 + bias</td>
</tr>
<tr>
<td>Output layer</td>
<td>2</td>
</tr>
</tbody>
</table>

4.5 TRAINING ALGORITHM

Figure 4.4 shows the flow chart for neural network training
algorithm.

i) Initialize the network synaptic weights with small random
values.

ii) Apply an input vector (Solar insolation) to the network and
calculate the corresponding output values.

iii) Compare the actual outputs with the desired outputs
(reference voltage) and determine the measure of error
(evaluating function).

iv) Determine the amount by which each weight is to be
changed and update all the connection weights.
v) Repeat the steps 2 to 4 with all training vectors until the error for the training vectors in the training set is reduced to an acceptable value.

**Figure 4.4 Flow Chart for Neural Network**

The nomenclature used in training the algorithm for back propagation net is as follows.

Input training vector:

\[ X = V = (v_1, \ldots, v_n) \]

\[ Y = I = (i_1, \ldots, i_n) \]

\[ \delta_k \] Portion of error correction weight adjustment for \( w_{jk} \) that is due to an error at output unit \( Y_k \); also, the information about the error at unit \( Y_k \) that is propagated back to the hidden units that feed into unit \( Y_k \).
δ_j Portion of error correction weight adjustment for W_{ij} that is due to the back propagation of error information from the output layer to the hidden unit Z_j.

α Learning rate

X_i Input voltage parameter for unit i.

Y_i Input current parameter for unit i.

θ^o_j Bias on hidden unit j.

W_j Hidden weights for unit j.

The net input to W_j is denoted Net^h_j:

\[ Net^h_j = \sum W_{ji} Y_i + \theta^h_j \] (4.1)

The output signal (activation) of X_j is denoted X_j:

\[ X_j = f(Net^h_j) \] (4.2)

θ^o_k Bias on output unit k;

X_k Output voltage parameter for unit k:

The net input to X_k is denoted Net^o_k

\[ Net^o_k = \sum W_{ki} X_j + \theta^o_k \] (4.3)

The output signal (activation) of X_k is denoted as X_k:

\[ X_k = f(Net^o_k) \] (4.4)

Either of the activation functions defined in the previous section can be used in the standard back propagation algorithm given here. The form of the data (especially the target values) is an important factor in choosing the
appropriate function. The relevant considerations are discussed in further. Other suitable activations functions are considered. Note that because of the simple relationship between the value of the function and its derivative, no additional evaluations of the exponential are required to compute the derivatives during the back propagation phase of the algorithm.

The algorithm is as follows.

Step 0. Initialize weights.
Step 1. While stopping condition is false, do steps 2-9
Step 2. For each training pair, do steps 3-8

4.5.1 Feed Forward

Step 3. Each input unit (X\textsubscript{i}, i=1,\ldots,n) receives input signal \(x_i\) and broadcasts this signal to all units in the next layer (hidden units).

Step 4. Each hidden unit (X\textsubscript{j}, j=1,\ldots,p) sums its weighted input signal.

\[
Net_j^h = \sum W_{ji} Y_i + \theta_j^h
\]  

applies its activation function to compute its output signal.

\[
X_j = f(Net_j^h)
\]

and sends this signal to all units in the layer next.
Step 5. Each output unit \((Y_k, k=1, \ldots , m)\) sums its weighted input signals

\[
Net_k^o = \sum W_{hk} X_j^h + \theta_k^o \tag{4.6}
\]

and applies its activation function to compute its output signal.

\[
X_j = f(Net_k^o) \tag{4.7}
\]

### 4.5.2 Back Propagation of Error

Step 6. Each output reference current unit \((Y_k, k=1, \ldots , m)\) receives a target pattern corresponding to the input training pattern, computes its error information terms,

\[
\delta_k = (t_k - y_k) f'(Net_k^o), \tag{4.8}
\]

calculates its weight correction term (used to update \(w_{jk}\) later),

\[
\Delta w_{jk} = \alpha \delta_k X_j \tag{4.9}
\]

Calculates its bias correction term (used to update \(w^k_0\) later),

\[
\Delta \theta_k^0 = \alpha \delta_k \tag{4.10}
\]

and sends \(\delta_k\) to units in the layer previous.

Step 7. Each hidden unit \((X_j, j=1, \ldots , p)\) sums its delta inputs (from units in the layer next),

\[
\delta_{inj} = \sum_{k=l}^m \delta_k w_{jk} \tag{4.11}
\]

Multiplies by the derivative of its activation function to calculate its error information term,
\[ \delta_j = \delta_{inj} f'(Net_j^h) \]  \hspace{1cm} (4.12)

calculates its weight correction term used to update \( W_{ij} \).

\[ \Delta W_{ij} = \alpha \delta_j x_i \]  \hspace{1cm} (4.13)

And calculates its bias correction term used to update \( \theta_j^h \).

\[ \Delta \theta_j^h = \alpha \tilde{\delta}_j \]  \hspace{1cm} (4.14)

### 4.5.3 Update Weights and Biases

**Step 8.** Each output unit \((X_k, \ k=1,\ldots,m)\) updates its bias and weights \((j = 0,\ldots,p)\).

\[ w_{jk}(\text{new}) = w_{jk}(\text{old}) + \Delta w_{jk} \]  \hspace{1cm} (4.15)

Each hidden unit \((X_j, \ j =1,\ldots,p)\) updates its bias and weights \((i=0,\ldots,n)\).

\[ W_{ij}(\text{new}) = W_{ij}(\text{old}) + \Delta W_{ij} \]  \hspace{1cm} (4.16)

**Step 9:** Test stopping condition.

An epoch is one cycle through the entire set of training vectors. Typically, many epochs are required for training a back propagation neural net. The foregoing algorithm updates the weights after each training pattern is presented. A common variation is batch updating, in which weight updates are accumulated over an entire epoch before being applied.

### 4.5.4 Random Initialization

The choice of initial weights influence the net reached a global minimum of the error and time taken to converge. The update of the weight between two units depends on both the derivative of the upper unit’s
activation function and the activation of the lower unit. The values for the initial weights are not too large, or the initial input signals to each hidden or output unit is likely to fall in the region where the derivative of the sigmoid function has a very small value. On the other hand, if the initial weights are too small, the net input to a hidden or output unit is close to zero, which also causes extremely slow learning.

A common procedure is to initialize the weights (and biases) to random values between -0.5 to 0.5 (or between -1 and 1 or some other suitable interval). The values are positive or negative because the final weights after training are of either sign.

4.5.5 Training Pattern

The error for the training testing pattern decreases, training continues. When the error begins to increase, the net is starting to memorize the training patterns too specifically. At this point, training is terminated, a relationship among the number of training patterns available $P$, the number of weights to be trained $W$, and the accuracy of classification expected $e$, are given by the equation (4.17).

$$\frac{W}{P} = e$$  \hspace{1cm} (4.17)

4.5.6 Data Representation

The input vectors and the output vectors have components in the same range of values. Due to one factor in the weight correction, expression is the activation of the lower unit. Units whose activations are zero will not
learn. This suggests that learning is improved if the input is represented in bipolar form and bipolar sigmoid is used for the activation function.

In voltage based peak power tracking scheme, the reference voltage to feed forward loop is to be adjusted such that it is equal to the MPP voltage at that solar insolation. Since the solar insolation is varying, the corresponding reference voltage \( V_{\text{ref}} = V_{\text{mpp}} \) for the feed forward loop also changes which is estimated.

MPP voltages are nonlinearly related with solar insolation. The input variables are PV array parameters like open circuit voltage \( V_{\text{oc}} \) and short circuit current \( I_{\text{sc}} \), atmospheric data like irradiance and temperature, or any combination of these. The output of the network is used to vary the duty cycle which is used to drive the power inverter to operate at or close to the MPP.

Figure 4.5 shows the neural network model. The input parameters are obtained from PV array. The input variables are PV array parameters namely open circuit voltage \( V_{\text{oc}} \) and short circuit current \( I_{\text{sc}} \) and the bias is atmospheric data, solar irradiance. The voltage to the feed forward loop is adjusted such that it is equal to the MPP voltage at solar insolation by sigmoidal activation function. The solar insolation varies, the corresponding reference voltage \( V_{\text{ref}} = V_{\text{mpp}} \) for the feed forward loop also changes to obtain the MPPV. The data for the training pattern are presented in the Appendix 1.
4.6 TRAINED RESULTS

Table 4.2 shows the neural network result for 18 patterns. In this the trained output and the expected value of voltage and current are listed. The trained output and the expected output are found to be deviated. Table 4.3 shows the neural network result for 45 patterns. Comparing the above results the trained output and the expected output of voltage are same with slight variation in current.

Table 4.2 Comparison of Neural Network Results for 18 Patterns

<table>
<thead>
<tr>
<th>No of Epochs</th>
<th>Performance With 18 Patterns (KW/SqM/Day)</th>
<th>Insolation (KW/SqM/Day)</th>
<th>Input Voltage (Volts)</th>
<th>Input Current (Ampere)</th>
<th>Trained Output Voltage (Volts)</th>
<th>Trained Output Current (Ampere)</th>
<th>Expected Voltage (Volts)</th>
<th>Expected Current (Ampere)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100000</td>
<td>0.209168</td>
<td>6</td>
<td>48</td>
<td>5.4369</td>
<td>52.515</td>
<td>4.9904</td>
<td>52.105</td>
<td>5.1000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>48</td>
<td>4.2668</td>
<td>51.812</td>
<td>4.1054</td>
<td>52.142</td>
<td>4.0972</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>48</td>
<td>3.4435</td>
<td>49.94</td>
<td>3.323</td>
<td>50.12</td>
<td>3.2934</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>48</td>
<td>2.6141</td>
<td>49.89</td>
<td>2.3256</td>
<td>50.00</td>
<td>2.4240</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>48</td>
<td>1.7081</td>
<td>48.16</td>
<td>1.6125</td>
<td>48.20</td>
<td>1.6640</td>
</tr>
</tbody>
</table>
Table 4.3 Comparison of Neural Network Results for 45 Patterns

<table>
<thead>
<tr>
<th>No. of EPOCHS</th>
<th>Performance With 45 patterns (KW/SqM/Day)</th>
<th>Insolation</th>
<th>Input</th>
<th>Trained Output</th>
<th>Expected</th>
</tr>
</thead>
<tbody>
<tr>
<td>75000</td>
<td>0.0742512</td>
<td>6</td>
<td>48</td>
<td>Voltage (Volts) 5.4369   Current (Ampere) 52.105</td>
<td>Voltage (Volts) 5.0954   Current (Ampere) 52.105</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>48</td>
<td>Voltage (Volts) 4.2668   Current (Ampere) 52.142</td>
<td>Voltage (Volts) 4.0698   Current (Ampere) 52.142</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>48</td>
<td>Voltage (Volts) 3.4435   Current (Ampere) 50.105</td>
<td>Voltage (Volts) 3.2940   Current (Ampere) 50.120</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>48</td>
<td>Voltage (Volts) 2.6141   Current (Ampere) 50.000</td>
<td>Voltage (Volts) 2.4166   Current (Ampere) 50.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>48</td>
<td>Voltage (Volts) 1.71081  Current (Ampere) 48.200</td>
<td>Voltage (Volts) 1.6636   Current (Ampere) 48.200</td>
</tr>
</tbody>
</table>

4.7 SIMULATION RESULTS

Figure 4.6 shows the simulation model of the system using ANN. The duty cycle varies according to the variation of voltage in the trained network, and the firing angle to the inverter varies accordingly. Most PV arrays have different characteristics, a neural network is specifically trained for the PV array with which it is used. The characteristics of the PV array also changes with time, implying that neural network is periodically trained to guarantee accurate MPPT.

![Figure 4.6 Simulation Model of the PV System with ANN](image)
Figure 4.7 shows the inverter output voltage and current to the corresponding pulse width modulation (PWM) generation. The solar insolation is varying, the corresponding reference voltage \( V_{\text{ref}} = V_{\text{mpp}} \) for the feed forward loop also changes.

![PWM Pulses and Inverter Output](image)

**Figure 4.7 Simulation Results of the PWM Pulses and Inverter Output**

### 4.8 CONCLUSION

The result shows that the performance of the system depends on the number of input patterns and the number of hidden units, which increases the error. Since the MPP voltages are nonlinearly related with solar insolation, it is difficult to find a closed relation between maximum voltage and insolation. The change in the trained output is related in a limited range. Due to this limitation, the above method is not suitable to predict the true peak power tracker. To overcome this, an alternative method is suggested for implementation of MPPT.