Artificial Intelligent Control of a Solar Tracking System

E.T. El Shenawy, M. Kamal and M.A. Mohamad

Solar Energy Dept., National Research Centre, El Behooth st., Dokki, Giza, Egypt

ABSTRACT

Solar tracking is essential for many solar energy based power systems, concentrators or flat-plate, to improve the overall system performance. Process identification, a key element in process control, is the experimental approach to the modeling of the process to be controlled or the plant of unknown parameters. In the present work, artificial neural network has been implemented to identify and to model a two axis solar tracking system. Also an intelligent proportional integral and a derivative type fuzzy logic controller with and without self tuning scaling factors were studied and have been applied to control the solar tracker. A comparison among the logic controllers and a conventional proportional integral and a derivative controller performance has been investigated. The environment has been developed over MATLAB / Simulink software and a real time workshop tools.

Key words: Solar tracker, Artificial intelligent control, neural network identification, fuzzy logic control.

Introduction

At present, solar energy systems are used in many fields of life, especially for solar electric generation. In all solar applications, the output from the solar system depends on the amount of solar radiation received by the system. The solar radiation received by the solar system varies with the angle of the sun's rays made with the plan of the system. The solar tracker is a system that follows the sun and keeps the sun's rays almost normal to the plane of the solar system all time. The solar tracker may be a single or a dual axis tracker. Single axis trackers have one degree of freedom that acts as an axis of rotation. The axis of rotation of single axis trackers is typically aligned along a true North meridian. It is possible to align it in any cardinal direction with advanced tracking algorithms. There are several common implementations of single axis trackers. These include horizontal single axis trackers, vertical single axis trackers, tilted single axis trackers and polar aligned single axis trackers. Dual axis trackers have two degrees of freedom that act as axes of rotation. These axes are typically normal to one another. The axis that is fixed with respect to the ground can be considered a primary axis. The axis that is referenced to the primary axis can be considered a secondary axis. There are several common implementations of dual axis trackers. They are classified by the orientation of their primary axes with respect to the ground. Two common implementations are tip-tilt dual axis trackers and azimuth-altitude dual axis trackers (Saxena et al., 1990) and (Jun and Gi, 2000).

(Saxena et al., 1990) designed their solar tracker by calculating the solar position as a function of time, and the solar collector is oriented at the calculated position in the sky. A highly accurate angle measuring device, such as a digital shaft encoder, must be installed on the rotating axis in order to position the collector to the calculated angle. The solar collectors that fixed upon solar trackers may be point, line focus or flat plate collectors.

The neural network (NN) can be defined as a simply class of mathematical algorithms, since a network can be regarded essentially as a graphic notation for a large scale of algorithms. Such algorithms produce solutions to a number of specific problems. On the other hand, the neural networks can be considered as black boxes which contains processing elements called neurons (nodes), that accept inputs and produce outputs. Neural network must be taught, or trained. They learn new associations, new patterns, and new functional dependence. Neural networks differ from each other in their learning modes. There are a variety of learning rules that establish when and how the connecting weights change. As a result, they also differ in their ability to accurately respond to the information presented at the input (Beale and Jackson, 1990). Neural networks have been used in diverse applications in control, robotics, pattern recognition, forecasting, medicine, power systems, manufacturing, optimization, signal processing and social/psychological sciences. They are particularly useful in system modeling such as in implementing complex mappings and system identification (Malik, 2002) and (Soteris, 2009). The mapping properties of artificial neural networks have been analyzed by many Researchers. (Kumpati and Parthasarathy, 1990) and (Qiao and Mizumoto, 1996) demonstrated that artificial neural networks
could be used successfully for the identification and control of non-linear dynamic systems. (Hornik, 1989) has shown that as long as the hidden layer comprises a sufficient number of nonlinear neurons, a function can be realized with a desired degree of accuracy.

The solar plants have highly nonlinear behavior and are subjected to perturbations (variation in the solar intensity, wind speed, humidity, flow of air). Since they characterized by long lifetime systems, then it is expected that its parameters will be changed with that long time. For such systems it is recommended to apply artificial intelligence control techniques to control them (Chekired, 2011). In recent years, fuzzy logic has become an important approach in designing nonlinear controllers because of its simplicity, ease of design and ease of implementation. The control of knowledge based system using linguistics variables that do not have precise values is of concern, and this allows the use of traditional human experience in designing the system (Cavallo et al., 1996).

In the present study, the artificial intelligence is used to identify and control the proposed two axis solar tracking system. An artificial neural network has been designed, optimized and implemented to identify a nonlinear model of the solar tracker to help us to determine the initial parameter of the proposed controller. In addition, a proportional integral derivative fuzzy logic controller technique (PIDFL) with and without self tuning capability have been selected to control the solar tracker. For comparison with the fuzzy logic controllers, the normal proportional integral derivative controller is studied.

2. Solar Tracker:

The solar tracker used in the present study is a two-axis azimuth-elevation solar tracking system. Fig. 1 and Fig. 2 show the proposed two-axis solar tracker, while table 1 defines the tracker parameters. The system senses the tracker position via the optical encoders that built within the system and applies inputs to the motors through the I/O devices (Keithley Metabyte DAS-1600 series I/O boards). The noise identification signal was coded on the computer and communicated with the azimuth-elevation tracker via I/O device and motor drives (RS 2000R). This communication has been verified using Matlab / Simulink software and DOS Real-Time application of the Real-Time Workshop (Cavallo et al., 1996) and (Valera et al., 2001). A series-parallel neural network type identifier was applied to improve the stability and the convergence properties of the identification process, while the fuzzy logic controller was used to control the tracker system.

![Schematic diagram of the two-axis tracker.](image-url)
Fig. 2: Two-axis solar tracker.

Table 1: Parameters of the solar tracker.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracker type</td>
<td>Azimuth – elevation two axis tracker.</td>
</tr>
<tr>
<td>Electric motor</td>
<td>Two separate dc motors for azimuth and elevation movements.</td>
</tr>
<tr>
<td>Motor voltage</td>
<td>24 V.</td>
</tr>
<tr>
<td>Power consumption</td>
<td>20 W, max.</td>
</tr>
<tr>
<td>Tracking resolution</td>
<td>&gt; 0.2°</td>
</tr>
<tr>
<td>Speed</td>
<td>Azimuth approx. 4°/sec.</td>
</tr>
<tr>
<td></td>
<td>Elevation approx. 7°/sec.</td>
</tr>
<tr>
<td>Driving range</td>
<td>Azimuth 180°.</td>
</tr>
<tr>
<td></td>
<td>Elevation 65°.</td>
</tr>
<tr>
<td>Wind speed</td>
<td>62 Km/h in operation.</td>
</tr>
<tr>
<td>Reflector size</td>
<td>4m².</td>
</tr>
<tr>
<td>Dimension</td>
<td>Diameter approx. 318 mm, Height approx. 625 mm.</td>
</tr>
</tbody>
</table>

3. Neural network identifier:

The solar tracker operation depends mainly on the climatic conditions (solar radiation and wind speed) and the type of the solar system installed with the tracker (flat plate, line focus or point focus collectors). Since the solar radiation and the wind speed have non-linear characteristics, then the solar tracker system can be considered as a non linear system. Since the neural network matches well with the non linear systems, it can be accurately model the solar tracker (Kumpati and Parthasarathy, 1990). A series parallel feed-forward neural network was used to construct a suitable identification model to the two-axis solar tracker. Fig. 3, shows the basic identification scheme using the artificial neural network.

Fig. 3: Neural network identification scheme.
As shown in Fig. 3, the neural network identifier is placed in parallel with plant (solar tracker). Both the neural network and the plant were subjected to the same input. The error between the plant output and the output from the neural network is used as the training signal to learn the neural network. The proposed neural network is a multilayer feed-forward network, which uses back-propagation training algorithm, that can be considered as a supervised learning procedure (Peter, 1999).

3.1. Training of the neural network:

The training process is the process of determining the connection weights of the neural network. These weights include both the hidden layers and the output layer weights. The neural network training is performed off-line utilizing a previously generated training data set, which consists of the input and output pairs. As shown in Fig. 3, the neural network receive input from each pair of training patterns (input and output) and produce the corresponding output which compared with output of the training patterns. The error between the output from the neural network and the output of the training pair is used to learn the network.

The variation of the input data amplitude covers the full variations of the system input. The training of network was run for 1054 learning epochs. The root-mean-square error was selected as a performance quantifying for the back-propagation training of the neural network model (Safak and Turkay, 2000).

3.2. Training data:

A random generated signal was generated by a function which is built in the Simulink and was applied as input signal to solar tracking system and then both the input and the output response of the system were collected and stored as training data for the neural network. The input training data is a driving voltage (that ranges from $-12$ to $+12$, V). The time step was chosen as 50 ms.

The collected experimental data set has been divided into training and validation subsets (70% of the data for the training set). The collected data set was formed of 10,000 patterns.

- Training set: A set of input output examples used for learning, to adjust the different weights of the neural network.
- Validation set: A set of examples used to choose the size of hidden units in a neural network and to improve the performance of a fully-specified model.

3.3. Training algorithm:

The complete training procedures that depend on the input and output data pairs, see Fig. 3, are described as follows (Zurada, 1992):

Step 1:

Determine the structure of the network and the size of hidden layer.

Step 2:

Weights of hidden layer ($W_{ji}$) and weights of output layer ($W_{oj}$) are initialized at small random values.

Step 3:

Using training pattern pairs, compute the hidden layer’s output,

$$ netY_j = \sum_{i=1}^{4} Z_i \cdot W_{ji} \quad \text{for } j = 1 \text{ to } nh, \quad i = 1 \text{ to } 4 $$

(1)

$$ Y_j = \frac{1}{1 + e^{-netY_j}} \quad \text{for } j = 1 \text{ to } nh $$

(2)

Step 4:

The output from the neural network can be calculated,
\[ neto = \sum_{j=1}^{nh} Y_j \cdot W_{oj} \quad \text{for } j = 1 \text{ to } nh \]  
\[ o = \frac{1}{1 + e^{-neto}} \]  

**Step 5:**

The error between the calculated output from the neural network and the desired output from the training data set can be calculated,

\[ E = \frac{1}{2} (d - o)^2 + E \]  

**Step 6:**

Compute the output layer error signal term,

\[ \delta_o = (d - o)(1 - o) \cdot o \]  

**Step 7:**

Adjust the output layer weights, (Howard, 2001),

\[ W_{oj} = W_{oj} + \beta_j \delta_o Y_j \quad \text{for } j = 1 \text{ to } nh \]  

**Step 8:**

Compute the hidden layer error signal term,

\[ \delta_j = Y_j (1 - Y_j) \cdot \delta_o W_{oj} \quad \text{for } j = 1 \text{ to } nh \]  

**Step 9:**

Adjust the hidden layer weights as following,

\[ W_{ji} = W_{ji} + \beta_j \delta_j Z_i \quad \text{for } j = 1 \text{ to } nh, \ i = 1 \text{ to } 4 \]  

**Step 10:**

Repeat the above steps starting from step 3 for the next training pattern until all patterns are finished. Validate the trained model using the test subset.

**Step 11:**

Compute root-mean-square error,

\[ E_{rms} = \frac{1}{2} \left[ \sum_p (d_p - o_p)^2 \right]^{1/2} \]  

**Step 12:**

Check the validation error if it is starts to increase or not for few epochs. If it starts to increase, then store the final weight values for hidden (Wji) and output (Woj) layers for minimum validation error epoch. Check \( E_{rms} \), if it isn’t with the permissible value, go to step 1 to increment hidden node to construct another neural network. Else end the validation set.

**Step 13:**

Test the epoch of different learned networks models. If the difference between errors for different number of hidden nodes is within a tolerance level, the neural network has a smaller number of hidden nodes is selected. Then the selected neural network model is ready to model the solar tracker.
3.4. Selection of neural network:

A Matlab program has been written and Simulink software and Real-Time Workshop tools have been used to achieve the presented training procedures. A popular and very powerful method for improving the network generalization is early stopping method that can improve the network generalization through modifying the performance function (Howard, 2001). The training of this method don’t proceeds until a minimum of the error on the training set is reached, but only until a minimum of the error on the validation set is reached during training. The validation error will normally decrease during the initial phase of training, as does the training set error. However, when the network begins to overfit the data, the error on the validation set will typically begin to rise. When the validation error increases for a specified number of iterations, the training is stopped and the weights and biases at the minimum of the validation error are returned. Fig. 4 shows the steps of selecting the neural network identifier. It starts with the collection of data, selection of a proposed neural network structure (structure and size of hidden layers), train, validate and select another structure of the network, and finally implement the optimized one for solar tracker model.

4. Fuzzy logic controller for the solar tracker:

A proportional derivative integral self tuning fuzzy logic controller (PIDSTFL) is used to control the two-axis tracker system. The proportional integral derivative (PIDFL) is simply connects the proportional derivative (PD) and proportional integral (PI) type fuzzy controllers together in parallel. The inputs to controller are the error and rate change of error. The detailed block diagram of PIDSTFL controller is shown in Fig. 5 (Ketata et al., 1995) and (Victor and Dourado, 1997).

The output of the PIDFL controller is: (Zhi-Wei et al., 2000) and (Qiao and Mizumoto, 1996)

\[ u_c = \alpha u + \beta \int u dt \]

\[ = \alpha (A + PK_e + DK_d e^*) + \beta \int (A + PK_e + DK_d e^*) dt \]

\[ = \alpha A + \beta A t + (\alpha K_p + \beta K_d) e + \beta K_p \int e dt + \alpha K_d D e^* \]

Where \( K_e, K_d, \alpha \) and \( \beta \) are PIDFL controller parameters, \((\alpha K_p + \beta K_d)\) is the proportional component, \((\beta K_p)\) is the integral component and \((\alpha K_d)\) is the derivative component of the logic controller.

---

**Fig. 4:** Flow chart of the selection of the neural network.
To improve both the transient and the steady state performance, the self-tuning scaling factors changing with time are described as follows (Qiao and Mizumoto, 1996):

$$\beta(e(t)) = \beta_s \times f(e(t))$$  \hspace{1cm} (12)  

$$K_{de}(e(t)) = K_{des} \times g(e(t))$$  \hspace{1cm} (13)  

$$f(e(t)) = a_1 \times (\text{abs}(e(t)) + a_2)$$  \hspace{1cm} (14)  

$$g(e(t)) = b_1 \times (1 - \text{abs}(e(t))) + b_2$$  \hspace{1cm} (15)  

To resist the oscillations in the case of slight oscillation response, the following mechanism is applied,

$$K_{de} = \frac{K_{des}}{\sigma_k}, \quad \beta = \beta_s \sigma_k$$  \hspace{1cm} (16)  

Where a1, a2, b1 and b2 are positive constants, K_{des} and \beta_s are the initial values of K_{de} and \beta respectively. \sigma_k is the absolute peak value at the peak time t_k (k=0,1,2,3…). The objective of f(e(t)) is to decrease \beta(e(t)) with the change of error as mentioned before. The function g(e(t)) is the inverse objective (Qiao and Mizumoto, 1996).

### 4.1. Controller technique:

In the PIDSTFL control technique, the program will access the position signal and then calculate the error and the derivative of the error to tune the controller parameters and improves the transient and steady state performance. Fig. 6, shows the membership functions of error, change of error and output of PIDSTFL, while the simplified simulink model of the fuzzy logic control of solar tracker is shown in Fig. 7. Table 2 shows its rule base. The implementation of self tuning fuzzy logic controller can be summarized in the following steps:

1. Measure the load angular position \( \theta(t) \).
2. Calculate both e and e’.
3. Normalize e and e’ and fuzzify the inputs using the rule base table (membership functions) with IF-THEN operation.
4. Transform the fuzzified inputs into fuzzy inference using the minimum-maximum operation. The minimum implication method (min) has been implemented to each rule and the maximum aggregation method (max) has been implemented to the consequents of the rules (10).
5. Defuzzify the information using centre of the gravity method to convert to fuzzy control output u(t) and denormalize this fuzzy signal to produce the real life control action (Cavallo et al., 1996).

Fig. 8, shows the flowchart of the software program that controls the system operation. Once the program starts, all variables such as input/output data are initialized, then the next step is to run the Simulink model. Using the Simulink model and the Real-Time Workshop toolbox that are tools of Matlab software to create an
executable codes for DOS and then download the executable codes into the target hardware. The operator should determine the type of the fixed solar collector on the tracker stand, and the collector acceptance angle in the case of the concentrating solar collectors (such as line or point focus collector). Afterwards the program calculates the corresponding tracking resolution. Then, the program will drive the next surface azimuth and tilt angels.

![Fig. 6: Memberships of error, change of error and controller output.](image)

**Fig. 6:** Memberships of error, change of error and controller output.

![Fig. 7: The simplified simulink model of the fuzzy logic control of solar tracker.](image)

**Fig. 7:** The simplified simulink model of the fuzzy logic control of solar tracker.

<table>
<thead>
<tr>
<th>Table 2: Rule base of the self tuning fuzzy logic controller.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e )</td>
</tr>
<tr>
<td>NB</td>
</tr>
<tr>
<td>NM</td>
</tr>
<tr>
<td>NS</td>
</tr>
<tr>
<td>ZO</td>
</tr>
<tr>
<td>PS</td>
</tr>
<tr>
<td>PM</td>
</tr>
<tr>
<td>PB</td>
</tr>
</tbody>
</table>

**Results And Discussion**

5.1. Solar tracker identification:

According to the training and validation technique shown in Fig. 4, many neural networks have been designed for the proposed model of the solar tracking system. Fig. 9 shows the optimized neural network model for identification of the solar tracker. The neural networks consists of three layers; input, hidden and output layers. The input layer has four inputs; \( u(k) \), the network output at samples \( k \) and \( (k-1) \), and a bias signal. The hidden layer has 12 neurons with one bias signal. The hidden layer has 12 neurons with one bias signal. The output layer has one output defining the tracker output model.

Fig. 10 and Fig. 11 show the training and validation of the neural network. From the figures, it is clear that the neural network solar tracker model is quit accurately estimates of solar tracker performance with an acceptable error of 0.2 degree. The root mean square error for both the training and validation of the neural network is shown in Fig. 12. From the figure, the selected network resulted in a root-mean-square error, \( E_{rms} \), of \( 0.72*10^{-8} \), which is quit accurately estimates of solar tracker model. In Fig. 12, within epochs 1- 600 the \( E_{rms} \) over the validation set decreased as the neural network began to generalize to a better degree. The increased generalization capability of the neural network during training epochs 600 - 750 is obvious as the \( E_{rms} \) over the training data was very close to that of the \( E_{rms} \) over the validation data set. However during epoch 800 – 1054...
the network goes to overfit the data, the $E_{\text{rms}}$ over the validation set began to increase. The best training epoch is number 800, then the weights of the network at this epoch has been returned. The performance results here is reasonable, as shown in Fig. 12, since the training and validation set errors have similar characteristics.

**Fig. 8:** Flowchart of the model and control of the dual-axis sun tracker.

5.2. **PIDSTFL controller:**

The conventional PID, PIDFL, and PIDSTFL control techniques are implemented to the sun tracking system for comparison. First, the controller parameters were selected off-line with the neural network tracker model and then tuned on-line with the real plant. Fig. 13 shows the comparison between PIDFL and PIDSTFL controllers. The figure illustrates the efficiency of the PIDSTFL controller over the normal PIDFL one, and the response of the PIDSTFL controller is relatively more stable than PIDFL controller.

The effect of the PIDSTFL controller parameter, $\beta$, can be seen from Fig. 14, which shows the effect of varying the parameter on the overall performance of the controller at $K_p=1$, $K_{de}=0.25$, $\alpha=0.05$, $\beta=0.1$, 1.6 and 3.5. As shown in the figure, decreasing the value of the controller parameter $\beta$ gradually, decreases the integral components, that increases the damping of the system which leads to increasing the system stability.
Adding the parameters $a_1$, $a_2$, $b_1$, $b_2$ in equations 14 and 15 results in expanding the region of tuning parameters and then improve the performance of PIDFL. Fig. 15 shows the solar tracker response in case of PIDFL and PIDSTFL fuzzy logic controllers at $K_p=1$, $K_{de}=0.25$, $\beta=0.1$, $\alpha=1.5$, $a_1=1.5$, $a_2=2$, $b_1=5$ and $b_2=0.8$. The figure shows that the two mechanisms are came to action and results in improving the transient performance, resisting the overshoot and improving the steady state performance. It also reflects the efficiency of the self tuned fuzzy logic controller over the normal PIDFL one.

Fig. 16 shows the effect of the step disturbance on the performance of the conventional PID and PIDFL and PIDSTFL controllers (proportional gain $K_p=0.5$, integral gain $K_i=0.1$ and derivative gain $K_d=0.05$). The responses of the three control systems have a small overshoot due to the disturbance, but PIDSTFL controller resists the disturbance faster than PIDFL one, and it is obvious that the conventional PID controller has a poor performance in the present of a disturbance.
Fig. 11: The validation of the neural network.

Fig. 12: Neural network model training and validation performance.
Fig. 13: Comparison between PID and PIFST fuzzy logic controllers, ($\beta=0.1$).

Fig. 14: Effect of the controller parameter $\beta$ on the performance of the controller, $k_e=5$, $k_{de}=0.25$, $\alpha=0.05$.

Fig. 15: Slar tracker response in case of PIDFL and PIDSTFL fuzzy logic controllers at $K_e=1$, $K_{de}=0.25$, $\beta=0.1$, $\alpha=1.5$, $a_1=1.5$, $a_2=2$, $b_1=5$ and $b_2=0.8$. 
Conclusions:

A two-axis azimuth-elevation solar tracker performance is studied, modeled, and controlled using artificial intelligence via a Matlab/Simulink software and Real-Time Workshop tools. An identifier neural network is designed, trained and validated to model the solar tracker. A proportional integral derivative with and without self-tuning fuzzy logic controller are used to control the tracker system. From the study, it can be concluded that the choice of the neural network via the training and validation process depends on several factors such as the selections of the representative training data and the number of these data. Also choosing the activation functions, initial weights and the size of nodes per hidden layer should be considered. The error evaluation and the training stopping methods are most important factors for improving the network generalization and estimation.

The proportional integral derivative with and without self-tuning fuzzy logic controllers provide a good performance as controllers of the solar tracker system. The fuzzy logic self-tuning one always provides a better performance in comparison with the normal fuzzy logic proportional integral derivative. Both of the fuzzy controllers provide better performance than the conventional proportional integral derivative one. The proportional integral derivative type controller works well when the process under control is in stable conditions, but it doesn’t work well as in the presence of disturbances. The study showed that the neural network can accurately model and validate the solar tracker which can operate accurately under the proposed fuzzy logic controllers controllers.

Nomenclature

\(e\) System error.
\(e^*\) Change of error.
\(E\) Sum of errors until this pattern.
\(E_{rms}\) Root-mean-square error.
\(d\) Desired output corresponding to associated input value.
\(D\) Derivative parameter of the controller.
\(nh\) Number of neurons per hidden layer.
\(o\) Calculated output from the neural network corresponds to the input values.
\(p\) Vectors over all the pattern.
\(P\) Proportional parameter of the controller.
\(u_c\) Controller output.
\(W_{ih}\) Connection weights from the input layer to the hidden layer.
\(W_{ho}\) Connection weights from the hidden layer to the output layer.
\(Y_i\) Output at each node in the hidden layer.
\(Z_i\) Input to the neural network, \(i = 1\) to \(4\). (\(Z_1\) is process input \(u_k\), \(Z_2\) is the delayed output \(y_p(k-1)\), \(Z_3\) is the delayed output \(y_p(k-2)\) and \(Z_4=-1\) is bias signal).
Greek letters:

- $\beta_j$: Learning constant (0.1).
- $\delta_o$: Output layer error signal.
- $\delta_{Yj}$: Hidden layer error signal.

References

Hornik, K., 1989 “Feed-forward Networks are Universal Approximators”, Neural Networks, 2: 359-366.
Peter, V., 1999, “Artificial-Intelligence-Based Electrical Machines and Drives”, Oxford University press, USA.