

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900

Open access books available

186,000

International authors and editors

200M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Using MATLAB to Develop Artificial Neural Network Models for Predicting Global Solar Radiation in Al Ain City – UAE

Maitha H. Al Shamisi, Ali H. Assi and Hassan A. N. Hejase
United Arab Emirates University
United Arab Emirates

1. Introduction

Information about the availability of solar radiation on horizontal surface is essential for the optimum design and study of solar energy systems. For a country like UAE, the efficient application of solar energy seems inevitable because of abundant sunshine available throughout the year. The traditional way of knowing the amount of global solar radiation (GSR) in a particular region is to install pyranometers at as many locations as possible in this region thus requiring daily maintenance and data recording, and consequently increasing cost of GSR data collection. Therefore, it is rather more economical to develop methods to estimate the GSR using climatological parameters (Akhlaque et al., 2009; Kassem et al., 2009; Falayi et al., 2008; El-Sebaï & Trabea, 2005).

Many researchers estimated global solar radiation by using artificial neural networks. (Mohandes et al., 1998) applied ANN techniques to predict GSR using weather data from 41 stations in Saudi Arabia. Data from 31 stations was used in training the NN and the remaining data was used for testing. Input variables to the NN included 4 parameters: latitude, longitude, altitude and sunshine duration. Their sample data was not large enough to allow a credible comparison between the ANN models used and empirical regression models. (Lam et al., 2008) have used Artificial Neural networks ANNs to develop predication models for daily global solar radiation using measured sunshine duration for 40 cities covering 9 major thermal climatic zones and sub-zones in China. (Alam et al., 2009) used ANNs to estimate monthly mean hourly and daily diffuse solar radiation based on weather data from 10 Indian stations which have different climatic conditions. (Alawi & Hinai, 1998) applied ANNs to predict solar radiation in areas not covered by direct measurement instrumentation. The input data that was used for building the network were the location, month, mean pressure, mean temperature, mean vapor pressure, mean relative humidity, mean wind speed and mean duration of sunshine. (Tasadduq et al., 2002) have used neural networks for the prediction of hourly mean values of ambient temperature 24 hours in advance. Full year hourly values of ambient temperature are used to train a neural network model for a coastal location – Jeddah, Saudi Arabia. (Elminir et al., 2005) applied ANN modeling techniques to predict solar radiation data in different spectrum bands from data of meteorology for Helwan (Egypt) meteorology monitoring station. (Rehman & Mohandes, 2008) developed ANN-based estimation GSR for Abha city in Saudi Arabia; by

using different combination of day of year, time day of year, air temperature and relative humidity. (Krishnaiah et al., 2007) considered the artificial neural network (ANN) approach for estimating hourly global solar radiation (HGSR) in India. The ANN models are presented and implemented on real meteorological data. The solar radiation data from 7 stations are used for training the ANN and data from two stations are used for testing the predicted values. (Mubiru, 2008) predicted a monthly average daily total solar irradiation on a horizontal surface for locations in Uganda by using artificial neural network technique, where both geographical and meteorological data are used to develop model.

For countries like UAE, where solar radiation has significant strength, the average annual solar hours is 3568 h (i.e. 9.7 h/day), which corresponds to an average annual solar radiation of approximately 2285 kWh/m² (i.e. 6.3 kWh/m² per day) (Assi & Jama, 2010).

This book chapter will show the potential of MATLAB tools in writing scripts that help in developing Artificial Neural Network (ANN) models for the prediction of global solar radiation in Al Ain city, UAE. The developed scripts use built-in commands and functions for customizing data processing, network architecture, training algorithms and testing performance of the ANN models.

2. Background

2.1 Neural network

A neural network is a massively parallel distributed processor made up of simple processing units that have a natural tendency for storing experiential knowledge and making it available for us. Artificial neural network (ANN) is a type of Artificial Intelligence technique that mimics the behavior of the human brain (Haykin, 2009).

ANNs have the ability to model linear and non-linear systems without the need to make assumptions implicitly as in most traditional statistical approaches. They have been applied in various aspects of science and engineering (Rivard & Zmeureanu, 2005; Chantasut et al., 2005).

ANNs can be grouped into two major categories: feed-forward and feedback (recurrent) networks. In the former network, no loops are formed by the network connections, while one or more loops may exist in the latter. The most commonly used family of feed-forward networks is a layered network in which neurons are organized into layers with connections strictly in one direction from one layer to another (Jain et al., 1996).

2.2 Multilayer preceptor (MLP)

MLPs are the most common type of feed-forward networks. Fig. 1 shows an MLP which has three types of layers: an input layer, an output layer and a hidden layer.

Neurons in input layer only act as buffers for distributing the input signals x_i ($i=1, 2 \dots n$) to neurons in the hidden layer. Each neuron j (Fig. 2) in the hidden layer sums up its input signals x_i after weighting them with the strengths of the respective connections w_{ji} from the input layer and computes its output y_j as a function f of the sum.

$$y_j = f\left(\sum_{i=1}^n w_{ji} x_i\right) \quad (1)$$

f can be a simple threshold function or a sigmoidal, hyperbolic tangent or radial basis function.

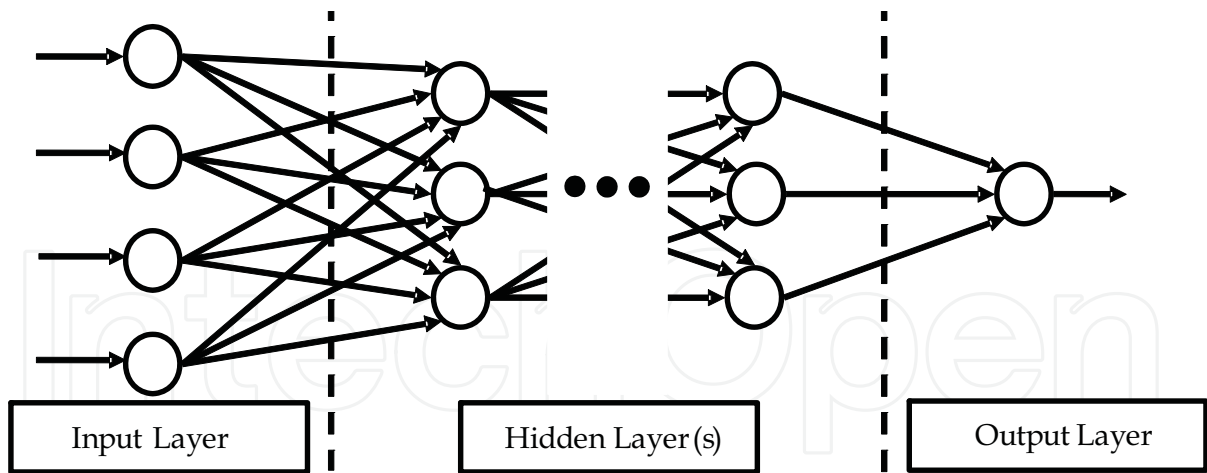


Fig. 1. A Multi-layered perceptron (MLP) network

The output of neurons in the output layer is computed similarly. The backpropagation algorithm, a gradient descent algorithm, is the most commonly adopted MLP training algorithm. It gives the change Δw_{ji} the weight of a connection between neurons i and j as follows:

$$\Delta w_{ij} = \eta \delta_j x_i \tag{2}$$

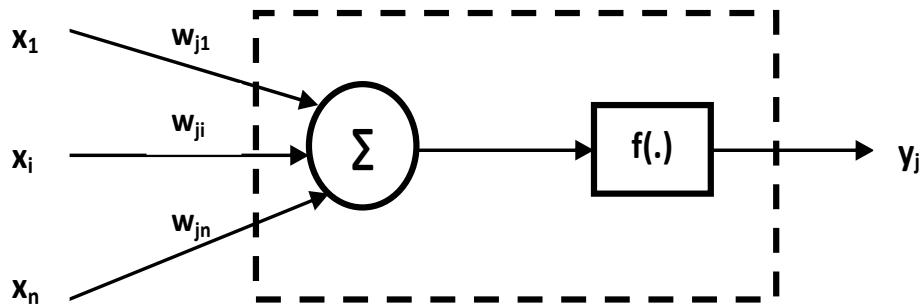


Fig. 2. Detail of the perceptron process

where η is a parameter called the learning rate and δ_j is a factor depending on whether neuron j is an input neuron or a hidden neuron. For output neurons,

$$\delta_j = (\partial f / \partial \text{net}_j)(y_j^{(t)} - y_j) \tag{3}$$

and for hidden neurons

$$\delta_j = (\partial f / \partial \text{net}_j)(\sum_q w_{jq} \delta_q) \tag{4}$$

In Eq. (3), net_j is the total weighted sum of input signals to neurons j and $y_j^{(t)}$ is the target output for neuron j .

As there are no target outputs for hidden neurons, in Eq. (4), the difference between the target and actual output of a hidden neurons j is replaced by the weighted sum of the δ_q terms already obtained for neurons q connected to the output of j .

The process begins with the output layer, the δ term is computed for neurons in all layers and weight updates determined for all connections, iteratively. The weight updating process can happen after the presentation of each training pattern (pattern-based training) or after

the presentation of the whole set of training patterns (batch training). Training epoch is completed when all training patterns have been presented once to the MLP. A commonly adopted method to speed up the training is to add a “momentum” term to Eq. (5) which effectively lets the previous weight change influence the new weight change:

$$\Delta w_{ij}(I+1) = \eta \delta_j x_i + \mu \Delta w_{ij}(I) \tag{5}$$

where $\Delta w_{ij}(I+1)$ and $\Delta w_{ij}(I)$ are weight changes in epochs $(I+1)$ and (I) , respectively, and μ is “momentum” coefficient (Jayawardena & Fernando, 1998).

2.3 Radial Basis Function (RBF)

Radial basis function network consists of three layers Fig 3. The input layer has neurons with a linear function that simply feed the input signals to the hidden layer. Moreover, the connections between the input and hidden layer are not weighted. The hidden neurons are processing units that perform the radial basis function. Each unit is mathematically defined as

$$\varphi_j(x) = \varphi(\|x - x_j\|), j = 1, 2, \dots, n \tag{6}$$

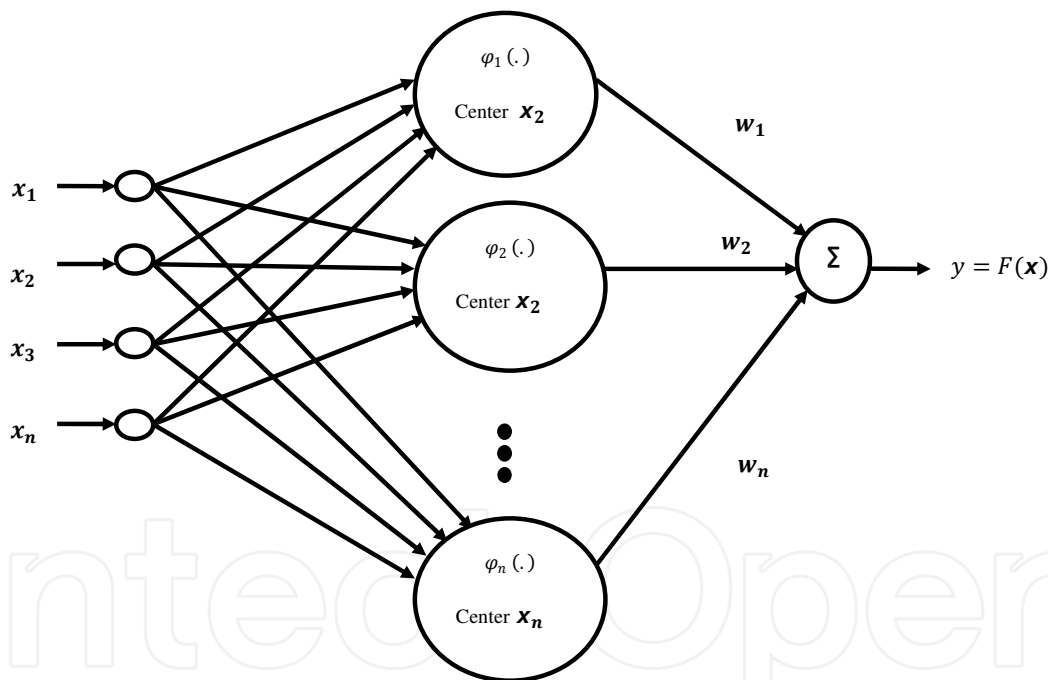


Fig. 3. Radial basis function network

The j^{th} input data point x_j denotes the center of the radial basis function, and the vector x is the pattern applied to input layer. Selecting the basis function is not crucial to the performance of the network the most common being the Gaussian basis function which is used in this study. It is defined as

$$\varphi_j(x) = \exp(-1/2\sigma_j^2(\|x - x_j\|^2)), j = 1, 2, \dots, n \tag{7}$$

The output neuron is a summing unit to produce the output as a weighted sum of the hidden layer outputs as shown by

$$F(x) = \sum_{j=1}^n w_j \phi_j(x) \tag{8}$$

(Haykin, 2009; Jayawardena & Fernando, 1998)

3. Problem definition

Building reliable solar energy systems regardless whether the system is a photovoltaic or thermal solar energy system requires information on the GSR in the region where the system is to be built. Many countries developed models for the GSR to predict the solar radiation and help in building solar energy systems. In UAE we still lack such models for GSR, and this work is the first attempt to generate a weather model for Al Ain city (see Fig.4 for its geographical location in the UAE) and which will later be extended to other UAE cities.

In this work, the maximum temperature (°C), mean wind speed(knot), sunshine(hours), mean relative humidity(%) and solar radiation (kWh/m²) for Al Ain city are provided by Abu Dhabi’s National Center of Meteorology and Seismology (NCMS), for the period between 1995 and 2007. The data between 1995 and 2004 is used for training the MLP and RBF- based ANN techniques while the data between 2005 and 2007 is used for testing the models.



Fig. 4. Geographical location of Al Ain city in the UAE (southwest of UAE at Latitude: 24° 16’ N and Longitude: 55° 36’ E)

Fig. 5 shows the time-series plot of the four measured weather predictors namely, maximum temperature (T), mean wind speed (W), sunshine (SH), relative humidity (RH), as well as the mean daily global solar radiation (GSR, dependent model variable) for Al-Ain city between 1995 and 2007. All the plots show a clear seasonal component of period equals to

365 days. These plots can help examine the correlation between the output variable (GSR) and the four weather predictors.

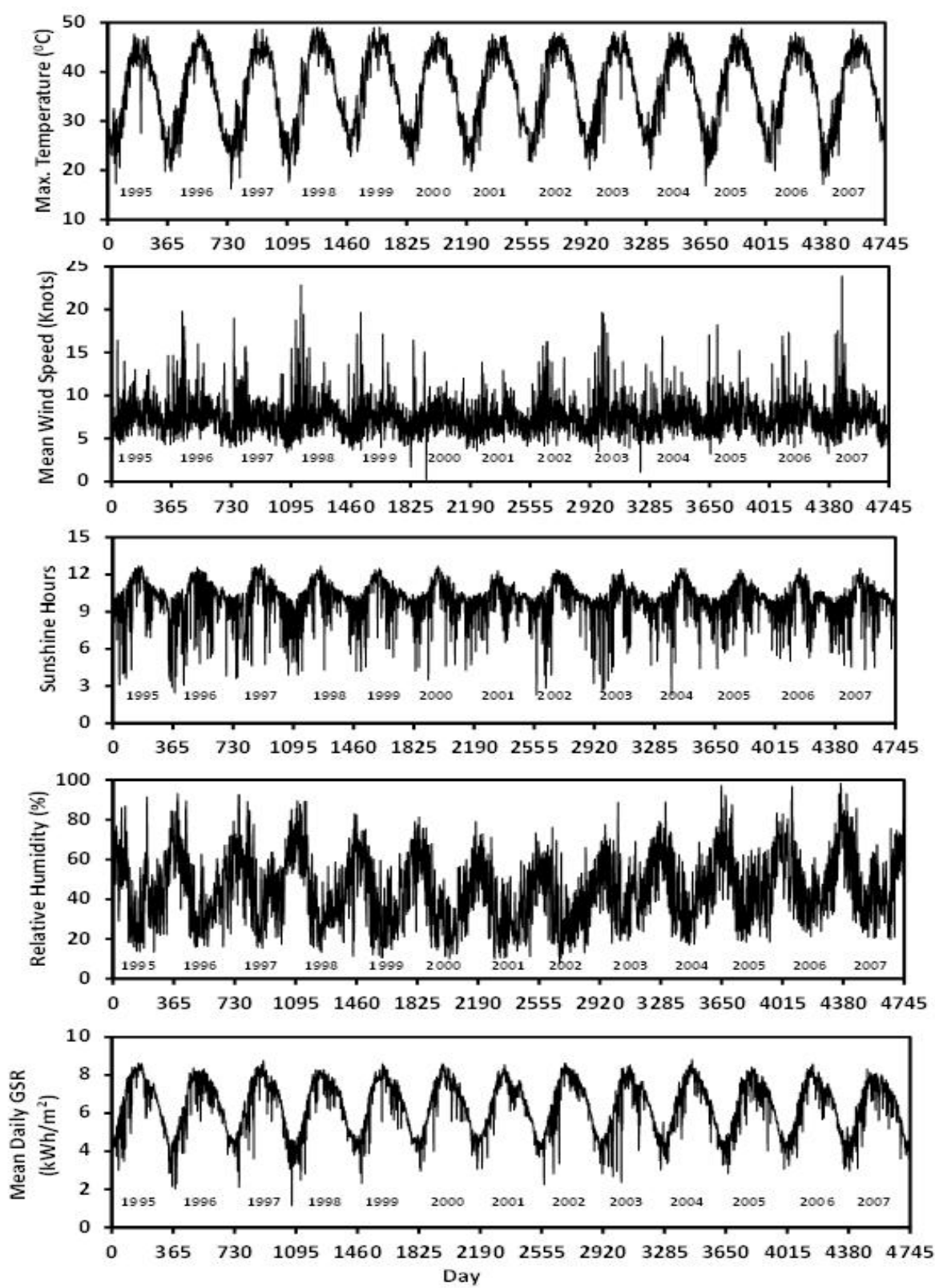


Fig. 5. Measured weather data for Al Ain City, UAE between 1995 and 2007

Eleven combinations of weather predictor variables were considered, as shown in Table 1, in order to investigate their effect on GSR. Both the MLP and RBF neural network methods are applied for predicting the GSR in Al-Ain city based on the combinations shown in Table 1.

Model No.	Input parameters	Model No.	Input parameters
1	T, W, SH & RH	7	T& SH
2	T, W & SH	8	T& RH
3	T, W & RH	9	W & SH
4	T, SH & RH	10	W& RH
5	W, SH & RH	11	SH & RH
6	T & W		

Table 1. Models based on different combinations of input parameters

Various network architectures were investigated in order to determine the optimal MLP architecture (i.e. the highest coefficient of determination, the lowest root mean square error and the lowest mean bias error) for each combination of input variables. Different training algorithms were used with changes in the number of neurons and hidden layers. In addition, different transfer functions including the tangent sigmoid, log sigmoid and linear functions in the hidden layer were also investigated.

Fourteen back-propagation training algorithms were tested in order to obtain the most appropriate algorithm for the training process. The algorithms include: Levenberg-Marquardt, Bayesian regularization, BFGS quasi-Newton, Powell -Beale conjugate gradient, Gradient descent, Gradient descent with momentum, Gradient descent with adaptive learning rate, Gradient descent with momentum & adaptive learning rate , One step secant, Fletcher-Powell conjugate gradient, Random order incremental training with learning, Resilient, Polak-Ribiere conjugate gradient, and Batch training with weight and bias learning rules.

The radius value (known as spread) of the function and the number of neurons are varied for best performance of the RBF network. Basically, the larger the value of spread, the smoother the function approximation will be. However, if the spread value is too large, many neurons may be required to fit the rapidly-changing function. On the other hand, very small spread values will require many neurons in order to fit the smooth function and networks cannot be generalized well (Beale et al, 2010). In this paper, the spread will be varied between 1 and 60 whereas the number of neurons is changed between 5 and 600 in order to come up with the most suitable ANN prediction models.

4. Designing and programming ANN models

4.1 Designing ANN models

Designing ANN models follows a number of systemic procedures. In general, there are five basics steps: (1) collecting data, (2) preprocessing data, (3) building the network, (4) train, and (5) test performance of model as shown in Fig 6.

4.1.1 Data collection

Collecting and preparing sample data is the first step in designing ANN models. As it is outlined in section 3, measurement data of maximum temperature (°C), mean wind speed(knot), sunshine (hours), mean relative humidity(%) and solar radiation (kWh/m²) for Al Ain city for 13-year period from 1995 to 2007 was collected through the NCMS.

4.1.2 Data pre-processing

After data collection, three data preprocessing procedures are conducted to train the ANNs more efficiently. These procedures are: (1) solve the problem of missing data, (2) normalize data and (3) randomize data. The missing data are replaced by the average of neighboring values during the same week. Normalization procedure before presenting the input data to the network is generally a good practice, since mixing variables with large magnitudes and small magnitudes will confuse the learning algorithm on the importance of each variable and may force it to finally reject the variable with the smaller magnitude (Tymvios et al., 2008).

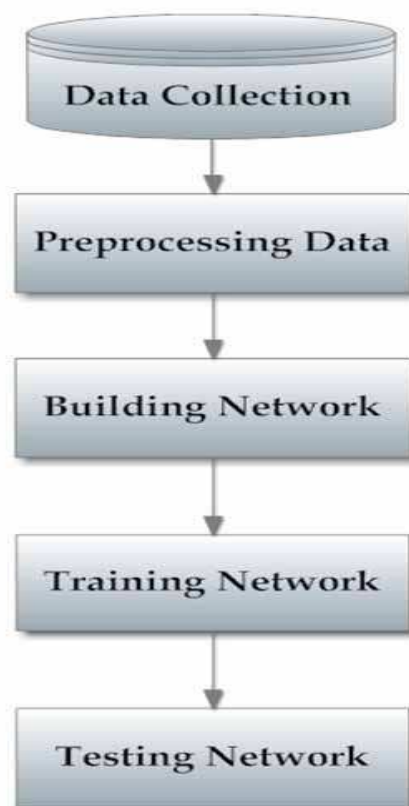


Fig. 6. Basic flow for designing artificial neural network model

4.1.3 Building the network

At this stage, the designer specifies the number of hidden layers, neurons in each layer, transfer function in each layer, training function, weight/bias learning function, and performance function. In this work, multilayer perceptron (MLP) and radial basis function (RBF) networks are used.

4.1.4 Training the network

During the training process, the weights are adjusted in order to make the actual outputs (predicated) close to the target (measured) outputs of the network. In this study, 10-year data period from 1995 to 2004 are used for training. As it is outlined in section 3, fourteen different types of training algorithms are investigated for developing the MLP network. MATLAB provides built-in transfer functions which are used in this study; linear ([purelin](#)),

Hyperbolic Tangent Sigmoid ([logsig](#)) and Logistic Sigmoid ([tansig](#)). The graphical illustration and mathematical form of such functions are shown in Table 2.

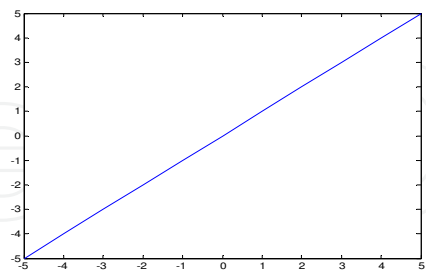
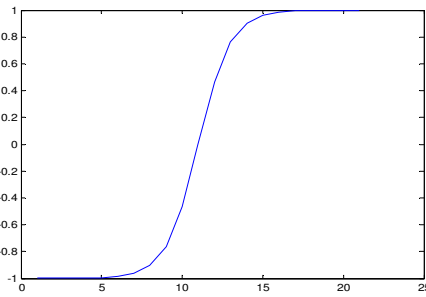
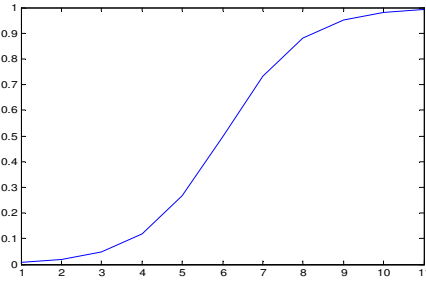
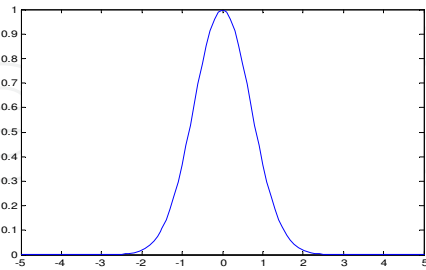
Function Name	Graphical Illustration	Mathematical form
Linear		$f(x) = x$
Hyperbolic Tangent Sigmoid		$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Logistic Sigmoid		$f(x) = \frac{1}{1 + e^{-x}}$
Gaussian RBF		$\varphi_j(x) = \exp\left(-\frac{1}{2\sigma_j^2}\ x - x_j\ ^2\right)$

Table 2. MATLAB built-in transfer functions

4.1.5 Testing the network

The next step is to test the performance of the developed model. At this stage unseen data are exposed to the model. For the case study of Al-Ain city, weather data between 2005 and 2007 have been used for testing the ANN models.

In order to evaluate the performance of the developed ANN models quantitatively and verify whether there is any underlying trend in performance of ANN models, statistical analysis involving the coefficient of determination (R^2), the root mean square error (RMSE), and the mean bias error (MBE) were conducted. RMSE provides information on the short term performance which is a measure of the variation of predicated values around the measured data. The lower the RMSE, the more accurate is the estimation. MBE is an indication of the average deviation of the predicted values from the corresponding measured data and can provide information on long term performance of the models; the lower MBE the better is the long term model prediction. A positive MBE value indicates the amount of overestimation in the predicated GSR and vice versa. The expressions for the aforementioned statistical parameters are:

$$MBE = \frac{1}{n} \sum_{i=1}^n (I_{p,i} - I_i) \quad (9)$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (I_{p,i} - I_i)^2} \quad (10)$$

where $I_{p,i}$ denotes the predicted GSR on horizontal surface in kWh/m², I_i denotes the measured GSR on horizontal surface in kWh/m², and n denotes the number of observations.

4.2 Programming the neural network model

MATLAB is a numerical computing environment and also a programming language. It allows easy matrix manipulation, plotting of functions and data, implementation of algorithms, creating user interfaces and interfacing with programs in other languages. The Neural Network Toolbox contains the MATLAB tools for designing, implementing, visualizing and simulating neural networks. It also provides comprehensive support for many proven network paradigms, as well as graphical user interfaces (GUIs) that enable the user to design and manage neural networks in a very simple way (<http://www.mathworks.com/products/neuralnet>).

In this paper MATLAB (R2010a) is used to write script files for developing MLP and RBF ANN models and performance functions for calculating the model performance error statistics such as R^2 , RMSE and MBE. Fig. 7 shows the procedural steps to develop the ANN models. The MLP program starts by reading data from an Excel file ([Training.xlsx](#) and [Testing.xlsx](#)). “xlsread” function is used to read the data specified in the Excel file.

```
Data_Inputs = xlsread('AlAin_TrainingSet_1995-2004_Model_1.xlsx');
```

```
Testing_Data = xlsread('AlAin_TestingSet_2005-2007_Model1.xlsx');
```

Training data samples are randomized by using the function “randperm”. This function returns a random permutation of the 3650 integers (training samples) while the order of columns is kept unchanged (T, W, SH, RH & GSR).

```
Shuffling_Inputs = Data_Inputs(randperm(3650),1:5);
```

Next, the input variables (maximum temperature, mean wind speed, sunshine and mean relative humidity) and output variable (global solar radiation) are specified for training and testing. The first four columns present the inputs while the last column denotes the output data (target). The total number of training samples is 3650 (10 years from 1995 to 2004 excluding leap days) while the testing samples are 1095 (3 years from 2005 to 2007).

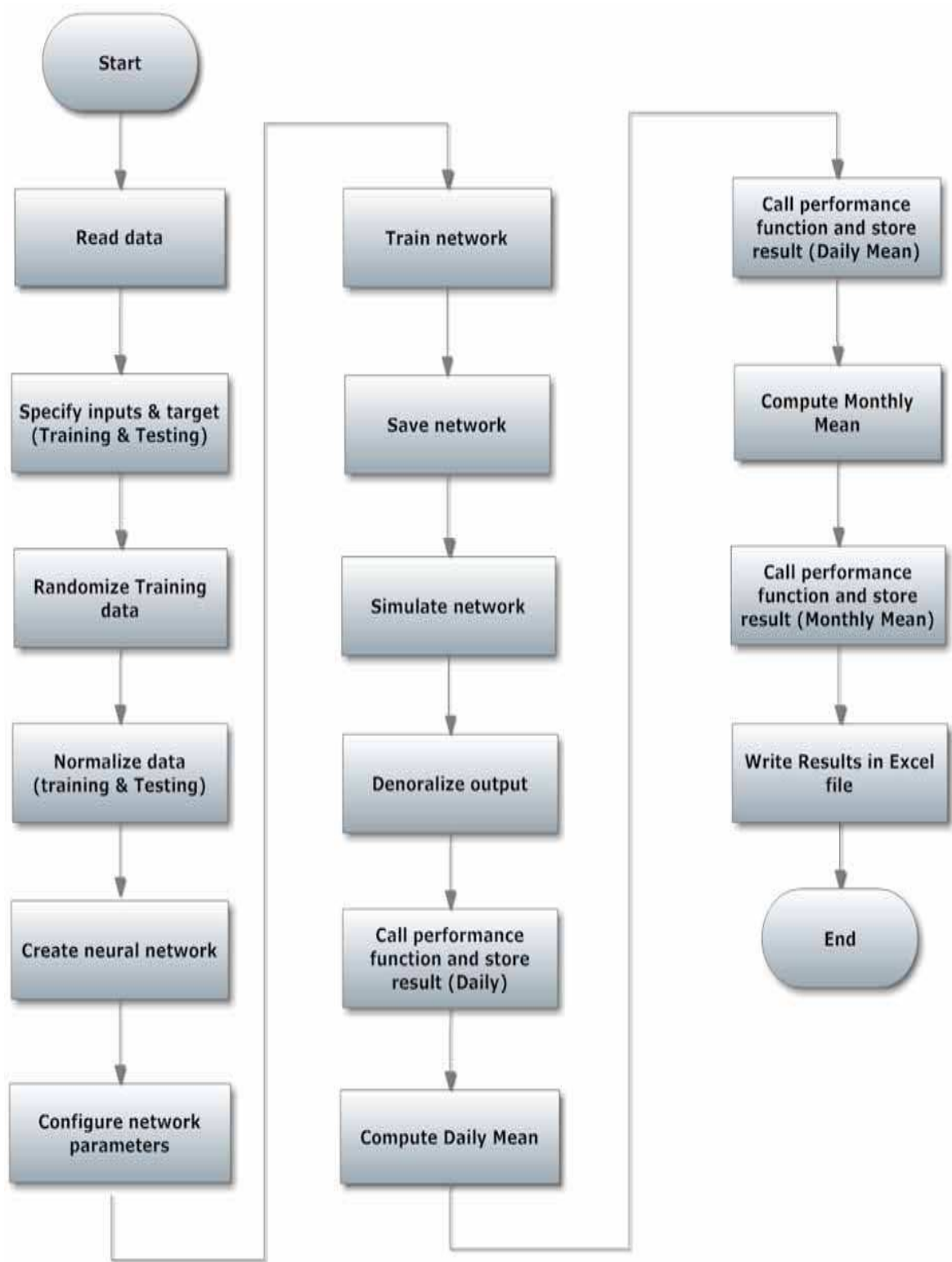


Fig. 7. Flow Chart for developing MLP and RBF networks using MATLAB

```
Training_Set = Shuffling_Inputs(1:3650,1:4)% specify training set [1995 - 2004]
Target_Set = Shuffling_Inputs(1:3650,5) % specify target set [1995 - 2004]
Testing_Set = Testing_Data(1:1095,1:4) % specify Testing set [2005- 2007]
Testing_Target_Set = Testing_Data(1:1095, 5) % specify Testing set, Target [2005- 2007]
```

A normalization process is applied for training and testing the data. Function “**mapstd**” is used to normalize the inputs and target in order to yield zero mean and unity standard deviation. The output is converted back into the same unit that is used for the original target. Training and testing data are converted to rows (MATLAB requires that all data must be presented as row vectors).

```
[pn, ps] = mapstd(Training_Set');
```

```
[tn, ts] = mapstd(Target_Set');
```

The settings structures **pn** and **tn** contain normalized values of inputs and output, respectively, while **ps** and **ts** contain the means and standard deviations of the original inputs and targets.

MATLAB helps devise the MLP model by using the built-in function “**newff**” which creates a feed-forward back-propagation network. The designer can specify the number of hidden layers, the neurons in each layer, the transfer function in each layer, the training function, the weight/bias learning function, and the performance function. Moreover, this command will automatically initialize the weights and biases. The function is called as follows:

```
MyNetwork = newff(pn,tn, [i] , {tf});
```

where the arguments **pn** and **tn** are the normalized input and output, respectively. **[i]** implies that the network structure consists of one hidden layer and “i” neurons. For “j” hidden layers we use [i, j]. The argument **{tf}** denotes the transfer function of the ith layer.

The network is next configured as follows

```
MyNetwork.trainFcn = LM;
```

```
MyNetwork.trainparam.min_grad = 0.00000001;
```

```
MyNetwork.trainParam.epochs = 1000;
```

```
MyNetwork.trainParam.lr = 0.4;
```

```
MyNetwork.trainParam.max_fail = 20;
```

where

“**trainFcn**”: defines the function used to train the network. It can be set to the name of any training function (LM = 'trainlm'; %Levenberg-Marquardt back-propagation)

“**trainparam.min_grad**”: denotes the minimum performance gradient

“**trainParam.epochs**”: denotes the maximum number of epochs to train

“**trainParam.lr**”: denotes the learning rate

“**trainParam.max_fail**”: denotes the maximum validation failures

The function “**newrb**” is used for iteratively creating an RBF network by including one neuron at a time. Neurons are added to the network until the sum squared error is found to be very small or the maximum numbers of neurons are reached. The call for this function is:

```
MyNetwork= newrb(pn,tn,goal,spread, mn, df);
```

where **pn** and **tn** are the input and target, respectively. The argument “**goal**” denotes the mean squared error goal, set here to be 0.01, and “**spread**” represents the spread of radial basis functions, changed between 1 and 60 and denoted here as “i”. The argument “**mn**” is the maximum number of neurons, changed between 5 and 600 and denoted here “j”, while “**df**” represents the number of neurons to add between displays and is set to 50.

The MLP network is trained using the “**trainFcn**” and “**trainParam**” train functions. The trained network is then saved in MATLAB by calling the functions

```
MyNetwork = train(MyNetwork,pn,tn);
```

```
save(NetworkName,'MyNetwork');
```

“**NetworkName**” is a variable name which can be changed according to the network train function and configuration in order to yield a meaningful name.

When the training is complete, the network performance should be checked. Therefore, unseen data (testing) will be exposed to the network. The testing simulation process is called with the statement:

```
y = sim(MyNetwork, testn); % simulate network
```

Fig. 8 and Fig. 9 show, respectively, screen captions of the MLP and RBF ANN training windows obtained using the “`nntraintool`” GUI toolbox in MATLAB.

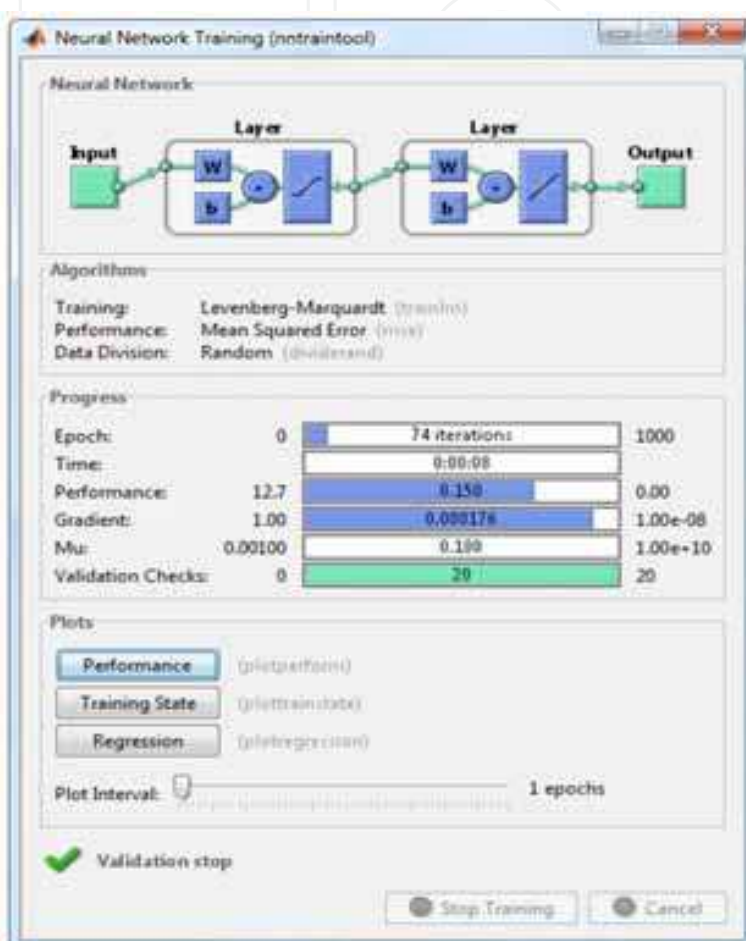


Fig. 8. MLP network Training Window

The output from the network (daily GSR) is denormalized in order to compare it with the measured data. After the network has been trained, one should use these settings to transform any future inputs that are applied to the network.

```
y_again = mapstd('reverse',y, targets) % denormalized
```

The performance function is then called to calculate and store the performance error statistics as follows

```
Daily_Result = performance(y,t,'all', 'v') % calculate statistics
```

Daily measured and predicted data are processed to compute and store the daily mean values along with corresponding error statistics. Afterwards, the daily mean of measured and predicted data are processed to produce monthly mean of measured as well as predicted data. This process is followed by using a MATLAB script to calculate the statistical results. The last step concludes with writing the mean daily and monthly test results along with corresponding statistical data to an Excel sheet.

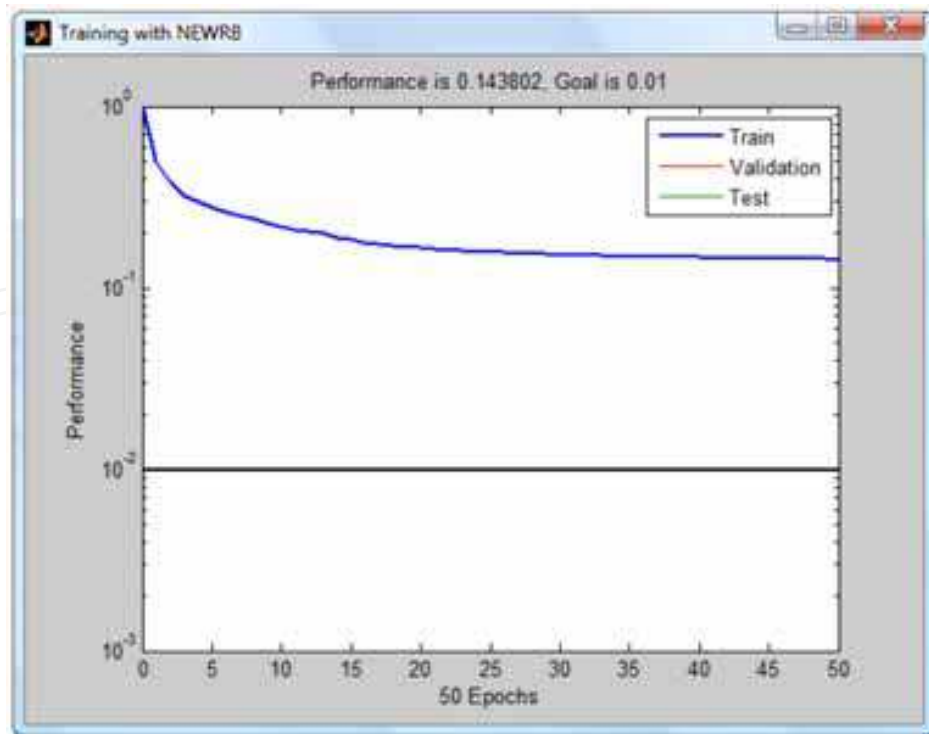


Fig. 9. RBF network Training Window

These steps are performed repeatedly where in MLP the control loop is a number of hidden layers and a number of neurons, while in RBF, the loop is controlled by spread and number of neurons. The stored results are checked and compared for all tested models in order to find the optimal network structure which has highest R^2 and lowest RMSE and MBE.

5. Results and discussion

This section presents the best achieved results for both the MLP and RBF ANN models. Tables 3 and 4 show the computed values of R^2 , RMSE and MBE for the developed ANN models (MLP and RBF) considering different network structures. For the network structure identification used in the second column of Tables 3-4, the first number indicates number of neurons in the input layer, the last number represents neurons in the output layer, and the numbers in between represent neurons in the hidden layers.

From Table 3 we note that model one is the best among all the investigated MLP models for long performance prediction as it yields the lowest values of MBE, RMSE and a 92 % coefficient of determination. The input parameters used in the selected model are the maximum temperature, daily sunshine hours, and the mean relative humidity. Table 4 on the other hand, shows that model four is the best among all the investigated RBF models; the input parameters for this model are again the maximum temperature, sunshine hours, and mean relative humidity.

In the seventh, eighth and eleventh models, the MLP technique has better accuracy for predicting the average monthly GSR compared to RBF. For these models, the computed MBE values of 0.00376, 0.06900 and 0.00193, respectively, are promising and provide good accuracy for long term prediction. On the other hand, some RBF models perform better in terms of low MBE values.

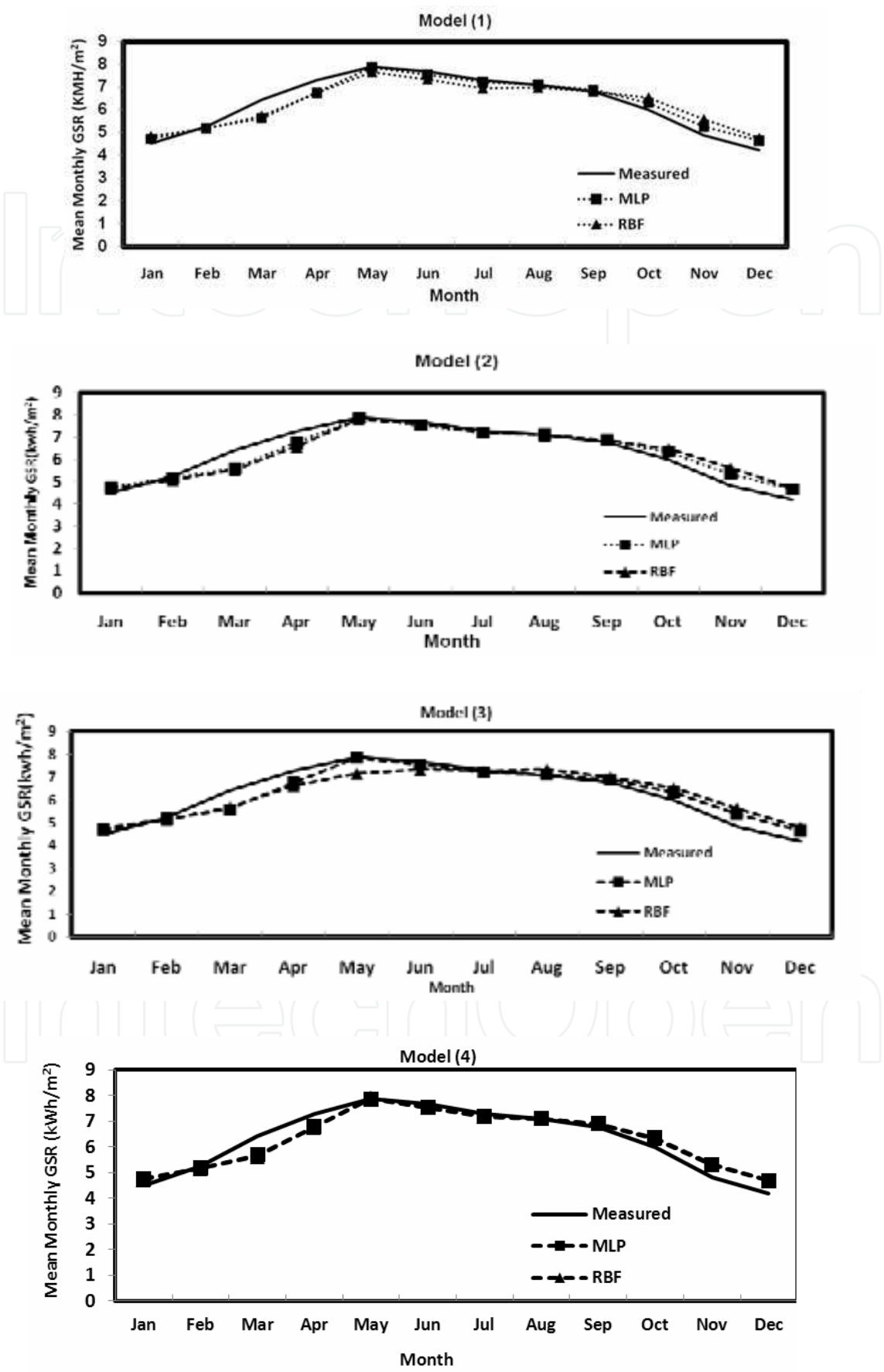
Model	Network Structure	R^2	RMSE	MBE
1	4 -100- 1	0.92	0.349	-0.00571
2	3 - 30 -1	0.91	0.370	-0.00754
3	3 - 35 -1	0.84	0.495	-0.01611
4	3 - 20 - 30 - 25 -1	0.92	0.356	-0.02271
5	3- 125 - 1	0.95	0.351	-0.07429
6	3 - 190 -1	0.83	0.518	-0.05402
7	2 - 80 - 1	0.91	0.384	0.00376
8	2 - 195 - 1	0.89	0.485	0.06900
9	2 - 110 - 1	0.91	0.519	-0.03544
10	2- 85 - 1	0.91	0.576	-0.02722
11	2 - 155 - 1	0.93	0.435	0.00193

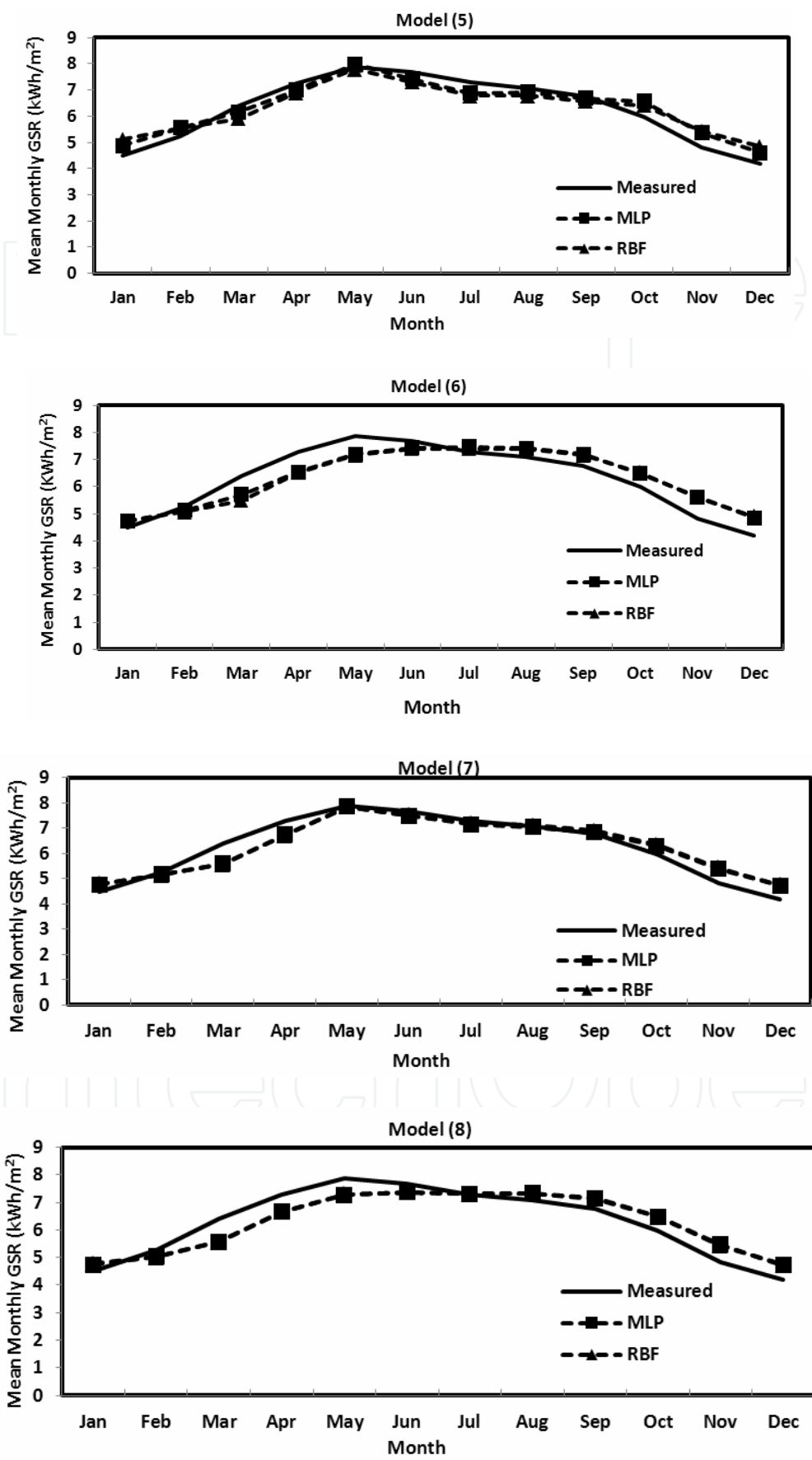
Table 3. Statistical error parameters of developed MLP models for different network structures

Model	Network Structure	R^2	RMSE	MBE
1	4- 50 -1	0.91	0.445	0.00069
2	3- 35 -1	0.86	0.459	-0.00055
3	3-75 -1	0.84	0.503	0.00073
4	3- 140 -1	0.92	0.356	-0.00307
5	3- 50 -1	0.93	0.446	0.01989
6	2- 50 - 1	0.80	0.556	-0.0208
7	2- 55- 1	0.90	0.401	-0.01835
8	2- 65 -1	0.85	0.483	0.48336
9	2- 15 -1	0.91	0.564	-0.03223
10	2 - 65 - 1	0.91	0.616	0.00611
11	2 - 385 - 1	0.92	0.466	-0.02073

Table 4. Statistical error parameters of developed RBF models for different network structures

Fig. 10 shows the comparison between predicted ANN models and the measured GSR data for all the eleven MLP and RBF-based networks models. This validation is done using measured GSR data for years 2005-2007. The predicted ANN models provide a very good prediction of monthly GSR behavior in Al-Ain, city.





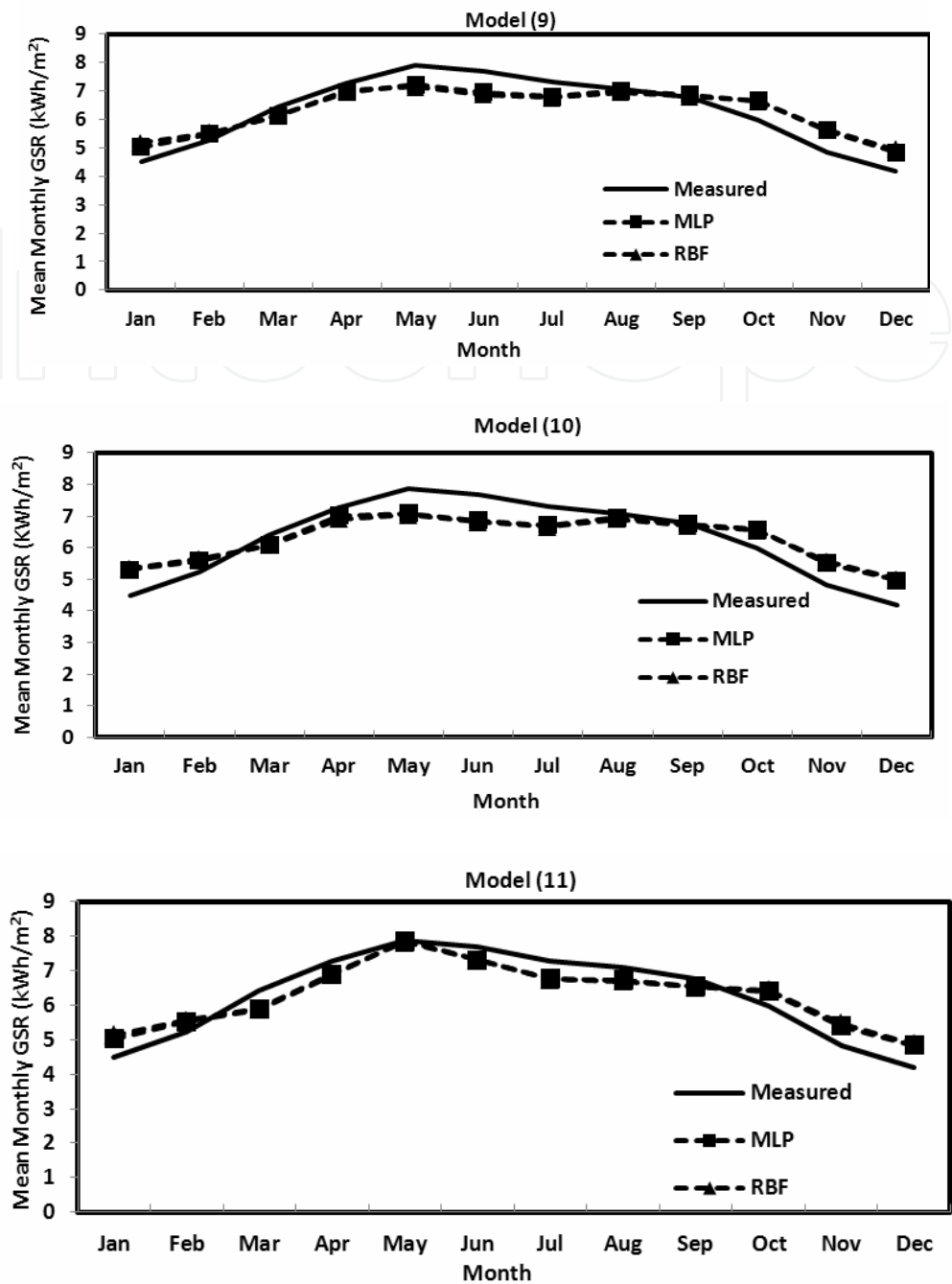


Fig. 10. Comparison between measured data and predicted ANN (MLP and RBF) models (1- 11)

6. Conclusion

In this work, MATLAB tools are used to predict monthly average global solar radiation in Al Ain city - UAE. Weather data between 1995 and 2004 are used for training the neural network, while data between 2005 and 2007 are used for testing. Eleven models with different input combinations are modeled with MLP and RBF ANN techniques. The obtained results confirm the superiority of the RBF technique over the MLP technique in most of the cases, namely, models 2- 6 and 9- 10, with deterministic coefficients near 90 %

and low MBE, MAPE and RMSE values. Moreover, ANN models have in general good performance even if one or more input parameters are unavailable. Currently, our focus is on modeling the GSR for other UAE cities and comparing the optimal ANN models with classical empirical regression and time-series regression models being investigated by this chapter's co-authors.

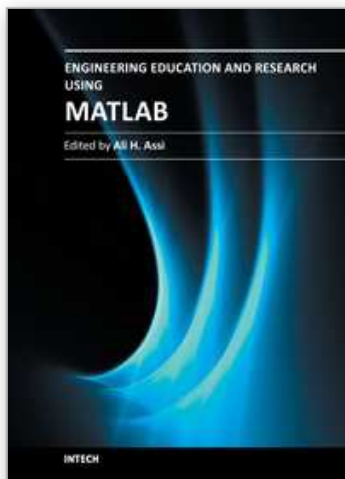
7. Acknowledgment

The authors would like to thank the National Center of Meteorology and Seismology (NCMS), Abu Dhabi for providing the weather data. This work is financially supported by UAE University Research Affairs under the contract #1542-07-01-10.

8. References

- Ahmed, M., Ahmad, F. & Wasim Akhtar, M. (2009). Estimation of global and diffuse solar radiation for Hyderabad, Sindh, Pakistan, *Jurnal of Basic and Applied Sciences*, Vol. 5, No. 2, pp. 73-77.
- Alam, S., Kaushik, S. & Garg, N. (2009). Assessment of Diffuse Solar Energy under General Sky Condition using Artificial Neural Network, *Applied Energy*, Vol. 86, pp. 554-564.
- Al-Alawi, S. & Al-Hinai, A. (1998). An ANN-based Approach for Predicting Global Solar Radiation in Locations with no Measurements, *Renewable Energy*, Vol. 14, No. 1-4, pp. 199-204.
- Assi, A. & Jama, M. (2010). Estimating Global Solar Radiation on Horizontal from Sunshine Hours in Abu Dhabi -UAE, *Advances in Energy Planning, Environmental Education and Renewable Energy Sources, 4th WSEAS international Conference on Renewable Energy Sources*, pp. 101 - 108, ISBN 978-960-474-187-8, Kantaoui, Sousse, Tunisia, May 3-6, 2010.
- Beale, M., Hagan, M. & Demut, H. (2010). Neural Network Toolbox User's Guide, 14.03.2011, Available from http://www.mathworks.com/help/pdf_doc/nnet/nnet.pdf
- Behrang, M., Assareh, E., Ghanbarzadeh, A. & Noghrehabadi, A. (2010). The potential of different artificial neural network (ANN) techniques in daily global solar radiation modeling based on meteorological data, *Solar Energy*, Vol. 84, pp. 1468-1480.
- Chantasut, N., Charoenjit, C. & Tanprasert, C. (2004). Predictive Mining of Rainfall Predictions Using Artificial Neural Networks for Chao Phraya River, *4th International Conference of The Asian Federation of Information Technology in Agriculture and The 2nd World Congress on Computers in Agriculture and Natural Resources*, August 9-12.
- Elminir, H., Areed, F. & Elsayed, T. (2005). Estimation of solar radiation components incident on Helwan site using neural networks, *Solar Energy*, Vol. 79, pp. 270-279.
- El-Sebaai, A & Trabea, A. (2005). Estimation of Global Solar Radiation on Horizontal Surfaces Over Egypt, Egypt. *J Solids*, Vol. 28, No. 1, pp. 163-175.
- Falayi, E., Adepitan, J. & Rabi, A. (2008). Empirical models for the correlation of global solar radiation with meteorological data for Iseyin, Nigeria, *Physical Sciences*, Vol. 3, No. 9, pp. 210-216.
- Haykin, S. (2009). Neural Networks and Learning Machines. 3rd edition, Pearson Education, Inc., New Jersey.

- Jain, A., Mao, J. & Mohiuddin, K. (March 1996). Artificial Neural Networks: A Tutorial, *IEEE Computer*, Vol. 29, No.3, pp.31-44.
- Jayawardena, A., Achela, D. & Fernando, K. (1998). Use of Radial Basis Function Type Artificial Neural Networks for Runoff Simulation, *Computer-Aided Civil and Infrastructure Engineering*, Vol. 13, pp. 91-99.
- Kassem, A., Aboukarima, A. & El Ashmawy, N. (2009). Development of Neural Network Model to Estimate Hourly Total and Diffuse Solar Radiation on Horizontal Surface at Alexandria City (Egypt), *Journal of Applied Sciences Research*, Vol. 5, No. 11, pp. 2006-2015.
- Krishnaiah, T., SrinivasaRao, S., Madhumurthy, K. & Reddy, K. (2007). A Neural Network Approach for Modelling Global Solar Radiation, *Applied Science Research*, Vol. 3, No. 10, pp. 1105-1111.
- Lam, J., Wan, K. & Liu, Y. (2008), Solar Radiation Modeling Using ANNs for Different Climates in China, *Energy Conversion and Management*, Vol. 49, pp. 1080-1090.
- Mohandes, M.; Rehman, S. & Halawani, T. (1998). Estimation of Global Solar Radiation Using Artificial Neural Networks, *Renewable Energy*, Vol. 14, pp. 179-184.
- Mubiru J. (2008), Predicting total solar irradiation values using artificial neural networks, *Renewable Energy*, Vol. 33, No. 10, pp. 2329-2332.
- Rehman, S. & Mohandes, M. (2008). Artificial neural network estimation of global solar radiation using air temperature and relative humidity, *Energy Policy*, (36), pp. 571-576.
- Tasadduq, I., Rehman, S. & K. Bubshait. (2002). Application of neural networks for the prediction of hourly mean surface temperature in Saudi Arabia, *Renewable Energy*, Vol. 25, pp. 545-554.
- The MathWorks (2011). MATLAB. URL: <http://www.mathworks.com/products/neuralnet>
- Tymvios, F., Michaelides, S. & Skouteli, C. (2008). Estimation of Surface solar radiation with Artificial neural networks, In: *Modeling Solar Radiation at the Earth Surface*, Viorel Badescu, pp. (221-256), Springer, ISBN 978-3-540-77454-9, Germany.
- Yang, J., Rivard, H. & Zmeureanu, R. (2005). Building Energy Predication with Adaptive Artificial Neural Networks, *Ninth International IBPSA Conference Montréal*, August 15-18.



Engineering Education and Research Using MATLAB

Edited by Dr. Ali Assi

ISBN 978-953-307-656-0

Hard cover, 480 pages

Publisher InTech

Published online 10, October, 2011

Published in print edition October, 2011

MATLAB is a software package used primarily in the field of engineering for signal processing, numerical data analysis, modeling, programming, simulation, and computer graphic visualization. In the last few years, it has become widely accepted as an efficient tool, and, therefore, its use has significantly increased in scientific communities and academic institutions. This book consists of 20 chapters presenting research works using MATLAB tools. Chapters include techniques for programming and developing Graphical User Interfaces (GUIs), dynamic systems, electric machines, signal and image processing, power electronics, mixed signal circuits, genetic programming, digital watermarking, control systems, time-series regression modeling, and artificial neural networks.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Maitha H. Al Shamisi, Ali H. Assi and Hassan A. N. Hejase (2011). Using MATLAB to Develop Artificial Neural Network Models for Predicting Global Solar Radiation in Al Ain City – UAE, Engineering Education and Research Using MATLAB, Dr. Ali Assi (Ed.), ISBN: 978-953-307-656-0, InTech, Available from: <http://www.intechopen.com/books/engineering-education-and-research-using-matlab/using-matlab-to-develop-artificial-neural-network-models-for-predicting-global-solar-radiation-in-al>

INTECH
open science | open minds

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2011 The Author(s). Licensee IntechOpen. This is an open access article distributed under the terms of the [Creative Commons Attribution 3.0 License](https://creativecommons.org/licenses/by/3.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

IntechOpen

IntechOpen