Anais do XX Congresso Brasileiro de Automática
Belo Horizonte, MG, 20 a 24 de Setembro de 2014

MODELING PHOTOVOLTAIC ARRAYS AND COMPARING ANN TOPOLOGIES FOR MAXIMUM POWER POINT TRACKING

Chrystian Lenon Remes*, Sérgio Vidal Garcia Olveira*, Yales Rômulo De Novaes*

* *Universidade do Estado de Santa Catarina - UDESC*
R. Paulo Malschitzy S/N, Campus Universitário
Joinville, Santa Catarina, Brazil

Emails: chrystian@windowslive.com, svgo@ieee.org, yales.novaes@udesc.br

Abstract— For Photovoltaic Arrays (PV), the Maximum Power Point Tracking (MPPT) deserves a special attention, ensuring that the maximum power can be extracted for any giving environmental condition. In this context, Artificial Neural Networks (ANN) are a good solution to perform the MPPT. In this paper, a comparative between different ANN topologies can be seen, where the number of epochs for convergence, time for algorithm execution and mean square error are analyzed through different numbers of neurons and training samples. Beyond the ANN simulations, the parameters for the PV used were obtained using Genetic Algorithms (GA).

Keywords— Photovoltaic Arrays, Artificial Neural Networks, Genetic Algorithms, Maximum Power Point Tracking

Resumo— Em Painéis Fotovoltaicos (PV) o Rastreamento do Máximo Ponto de Potência (MPPT) merece uma atenção especial, garantindo que a máxima potência possa ser extraída para uma dada condição do ambiente. Neste sentido, as Redes Neurais Artificiais (ANN) são uma boa opção para a realização do MPPT. Neste trabalho, um comparativo entre diferentes topologias de ANN pode ser visto, onde o número de épocas para convergência, tempo de execução do algoritmo e o erro médio quadrático são analisados através da utilização de diferentes números de neurônios e de amostras de treinamento. Além das simulações da ANN, os parâmetros do PV utilizados foram obtidos através de Algoritmos Genéticos (GA).

Palavras-chave— Painéis Fotovoltaicos, Redes Neurais Artificiais, Algoritmos Genéticos, Rastreamento do Máximo Ponto de Potência

1 Introduction

One of the important highlights on photovoltaic arrays (PV) is their dependence with climatic variables like solar radiation and temperature (Carrijo et al., 2010), making it highly dynamic, especially for their dependence with solar radiation, that may change in a few seconds by a cloud shading, for example. These variables have a large impact on the output power provided by the photovoltaic array. Once the power on photovoltaic arrays is dependent on these climatic variables, the controller needs to be fast enough to find the best operating point for each instantaneous climatic condition, and then achieve the desired power set point. The techniques used to find the maximum operating point are defined as Maximum Power Point Tracking (MPPT). The Artificial Neural Networks (ANN) appears as a MPPT method, since they can learn the behavior of power on Maximum Power Point (MPP), through training, and extrapolate its behavior for any giving climatic input with good accuracy. Also, other variables, climatic or not, may be added as inputs for ANN, and the ANN can then achieves a model that also relates these other variables to the output power, beyond solar radiation and temperature (Esram e Chapman, 2007).

A known problem of using ANN in general is how to determine the best architecture and topology for it, and what is the best number of samples to be used on its training. So, this paper proposes showing the differences between some buildings of ANN used on photovoltaic arrays, and then, provide a way to choose a better approach for the best ANN configuration to be used on MPPT. A better ANN configuration brings as benefits the reduced computational efforts during the training phase, keeping output errors small. Also, while defining some of the PV parameters for this paper, a numerical method was implemented through Genetic Algorithms (GA). The parameters found were used for all ANN simulations.

In Section 2, a brief discussion about the PV modeling and the PV parameters will be shown. Also, a quick description of the ANN will be done. In section 3, the ANN results will be discussed for different ANN configurations, along with the input-output data classification for ANN training and its validation. Finally, Section 4 has the conclusions about the obtained results.

2 Photovoltaic Array Parameters And Artificial Neural Network For MPPT

For ANN training, a MATLAB® based model was used for simulating the equivalent circuit approximated by a single-diode. This is a model that can simulate well a given PV with relative simplicity, as described in (Villalva et al., 2009). The catalog parameters provided by the PV manufacturer are shown in Table 1, where V_{mpp}, I_{mpp} and P_{mpp} are the voltage, current and power at the MPP, respectively, K_{Ti} is a constant that relates

the current supplied by PV and its temperature, N_s is the number of connected cells in series, V_{oc} is the open circuit voltage at the STC (Standard Tests Conditions), I_{sc} is the short circuit current at the STC and E_g is the bandgap energy from polycrystalline silicon cells. The STC establishes the reference for temperature at 25 °C and for solar radiation of $1000W/m^2$, with an air mass of 1.5.

Table 1: Catalog PV Parameters.

Parameter	Value	Unit
V_{oc}	21.9	V
I_{sc}	7.65	A
V_{mpp}	17.7	V
I_{mpp}	7.38	A
P_{mpp}	130.63	W
K_{Ti}	2.601×10^{-3}	A/K
N_s	36	–

2.1 Equivalent PV Circuit

The photovoltaic arrays can be approximated by an electrical circuit. The most common electrical model consists of a current source, a diode, a series resistance and a shunt resistance (Mahdi et al., 2010), as shown in Figure 1. The current source represents the energy provided by the solar radiation, and it is represented in Equation 1:

$$I_{ph} = (I_{sc} + K_{Ti}(T - T_r))\frac{S}{S_r} \tag{1}$$

Where:
I_{ph}: Total current provided by solar radiation
I_{sc}: Short circuit current provided in STC
K_{Ti}: Constant - Relates the PV current and temperature
T: Instantaneous PV temperature
T_r: STC temperature
S: Instantaneous solar radiation
S_r: STC solar radiation

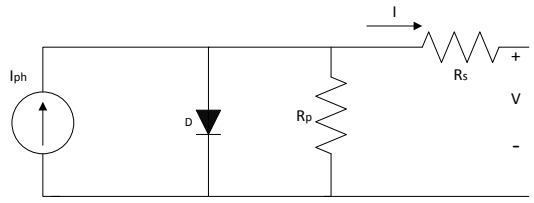

Figure 1: Photovoltaic array equivalent circuit.

This circuit approximation results in Equation 2, which represents the output current of the PV:

$$I = I_{ph} - I_s(e^{(\frac{q}{Ak}\frac{1}{T}(V+I.R_s))} - 1) - \frac{V - I.R_s}{R_p} \tag{2}$$

Where:
I_s: Diode saturation current
q: Elementary charge
A: Ideality diode's constant
k: Boltzmann's constant
R_s: Series resistance
R_p: Shunt resistance

In Equation 2, it is clear that the output current I depends on its own value, and also on the output voltage value. So, the only manner to obtain the PV parameters is through numerical methods (Villalva et al., 2009).

The diode saturation current I_s is a factor dependent on the junction temperature and of a I_{so} constant, which is the diode saturation current at the reference temperature, on STC. I_{so} can be calculated if the PV is on STC conditions and in the Open Circuit operation, which makes the output current null. Under these conditions, the current provided by the photovoltaic effect is the Short Circuit Current, and Equation 2 can be manipulated resulting in Equation 3:

$$I_{so} = \frac{I_{sc} - \frac{V_{oc}}{R_p}}{e^{\left(\frac{q}{Ak}\frac{1}{T_r}V_{oc}\right)} - 1} \tag{3}$$

Once I_{so} is in hands, and now imposing the MPP at the STC conditions, Equation 2 can be manipulated once more, now isolating the ideality diode constant A. The ideality diode constant is given by Equation 4:

$$A = \frac{V_{mpp} - V_{oc}}{\frac{kT_r}{q}.\ln\left(I_{sc} - I_{mpp} - \frac{V_{mpp}}{R_p}\right)} \tag{4}$$

It's important to note that theses constants are depending on the R_p value, and that it will be considered as an infinite value. The section 2.2 shows how R_s and R_p values were calculated.

2.2 Genetic Algorithms for PV Parameters Obtainment

Genetic Algorithms (GA) are a powerful tool for solving optimization problems, where function minimization is included, as seen in (Zagrouba et al., 2010).

GA are inspired in biological behavior, such as ANN, where a computational population is randomly started and imposed to a condition defined by the function. While the population evolves, the

individual genes approaches the best solution for the given function (Hadji et al., 2011).

For GA used in this paper, the selection of individuals for crossover is done by a stochastic uniform approach. In this selection method, each individual has a probability to be chosen based on its value from the fitness function. So, the algorithm sweeps in equal steps all the range determined by the entire population and an individual is selected during each step. In this method, an individual can be selected more than once. Two individuals are keep automatically for the next generation by elitism. The crossover method creates new individuals from a weighted average of the parents. Mutation occurs in some random individuals and points to a random direction.

To achieve the R_p and R_s values, a function to be minimized was defined using two values of resistance as variables. Having the parameters given in the PV catalog, which are also shown on Table 1, V_{mpp}, I_{mpp}, P_{mpp} and K_{Ti}, Equation 2 can be applied to the MPP point, resulting in Equation 5, and multiplying it for the V_{mpp}, this results in the power provided by the PV on the MPP.

Using the P_{mpp} provided by the catalog along with V_{mpp} and I_{mpp}, Equation 6 emerges naturally as a cost function to be minimized having the resistance values as variables.

$$I_{mpp} = I_{sc} - I_{so}(e^{(\frac{q}{Ak}\frac{1}{T_r}(V_{mpp}+I_{mpp}.R_s))} - 1) - \frac{V_{mpp} - I_{mpp}.R_s}{R_p} \quad (5)$$

$$J(R_s, R_p) = |P_{mpp} - V_{mpp}.I_{mpp}(R_s, R_p)| \quad (6)$$

Where:
J: Cost function to be minimized

Before starting the iterations for minimizing the cost function $J(R_s, R_p)$, the I_{so} and A constants must be defined. As they are values dependent of R_p, an approximation must be done, considering the shunt resistance as infinite. This eliminate the R_p dependence of I_{so} and A, and gives values closer enough to be used on PV model. So, I_{so} and A are calculated and shown in Table 2. Using these values, the R_s and R_p are finally found as the solution for the cost function thorugh GA. As the solution changes every running, because the algorithm starts with a random population, the values presented in Table 2 were fixed for the following steps.

The values given on Table 2 show that I_{so} and A approximates very well to the real values of the PV, once I_{so} has a very small value and A is between 1 and 2, since those are the desired condition for both cases. For the resistances R_s and R_p, the values obtained through GA also reflect a good PV representation, where R_s is very close to zero and R_p can be considered big enough to be approximated as infinite.

Table 2: Catalog PV Parameters.

Parameter	Value	Unit
R_s	4.5132×10^{-5}	Ω
R_p	9.3399×10^{7}	Ω
I_{so}	2.0463×10^{-7}	A
A	1.3579	–

Now, this paper will use these parameters from Table 2 on the next sections to represent the PV for MPPT analysis.

2.3 Artificial Neural Networks

The ANN is a grouping of artificial neurons, inspired in the biological neuron behavior, which is the simplest structure of an ANN. The artificial neuron is composed by a synaptic weights vector, an adder, a bias and the activation function. Its representation is seen in Figure 2. The mathematical model for the neuron is shown in Equation 7.

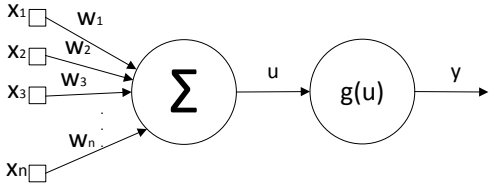

Figure 2: Artificial neuron and its mathematical model.

$$\begin{cases} u = \mathbf{X} \cdot \mathbf{W} - b \\ y = g(u) \end{cases} \quad (7)$$

Where:
X: Neuron Input Vector
W: Neuron Weights Vector
b: Bias
u: Net Input
g: Activation Function
y: Neuron Output

For the supervised training, a pair input-output is given, where for one input sample, its desired output y_d is known. Having them, the training consists to begin with an initial value of **W** for the input **X**, which results in an output y. Comparing y with the desired output y_d, the output error is known, and the process will iterate trying to minimize this error.

It's important to know that the variables chosen for ANN inputs can influence drastically on ANN's internal structure. Therefore, adding or

removing input variables will result in a totally different structure of ANN. However, even if two ANNs have different internal structure, not necessarily they will have a different input-output behavior. So, a good practice is to evaluate the ANN behavior using different sets of inputs vector, and then analyze which of them brings a better representation of the real solution.

If compared with other MPPT techniques, ANN can take advantage on reducing the output oscillation (Bastos et al., 2012), having fast convergence for MPP (after training phase) and achieving a real MPP, besides of quickly adjusting the MPP even under abrupt changes on climatic conditions. On the other hand, ANN has a high computational cost during the training phase, which makes its implementation difficult, beyond of being dependent of the PV used (Esram e Chapman, 2007).

2.4 Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP) is an ANN architecture that uses multiple neuron layers between the input and the output. An example of MLP is the one adopted in this paper, represented in Figure 3, where there are one input layer, one hidden layer with n neurons, and the output layer with just one neuron. These combination of multiple neurons in multiple layers results in a larger flexibility on dealing with complex and nonlinear problems (Mashaly et al., 1994).

For the MLP training, the back-propagation algorithm is a common used method. It consists in two steps, where the first calculates the ANN output and the second will adjust the synaptic weights for all ANN's neurons. In the first step, the ANN calculates an output using the existing weight values without changing them. This step is called forward, and is necessary just to produce an output to be compared with the desired output, thus generating the error. In the second step, called backward, the ANN will propagate the error through the neurons in the opposite way, starting from the output layer to the input layer. Then, the weights are adjusted achieving the error minimization.

As this method is based on the minimization of the error function through its derivatives, the activation function of the neurons must be continuous and derivable. For this reason, activation functions like the step or signal functions are not applicable in this architecture.

3 ANN MPPT Results

Only the MLP architecture was considered while obtaining the results. The reason for this is that the MLP in general has a good performance. Also, a good advantage on using ANN for

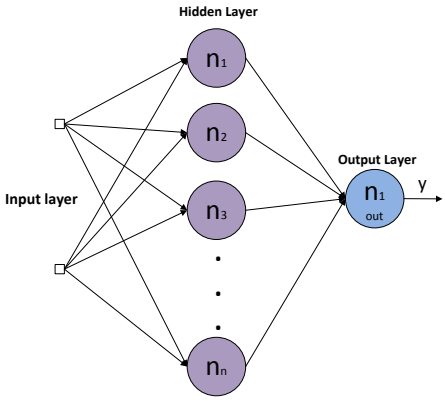

Figure 3: Generic representation of the ANN used in simulations.

MPPT algorithms is the reduction of the oscillation for the output signal, as described in (Bastos et al., 2012). Variations for the ANN were imposed on its topology: two activation functions were used, and the number of hidden layer neurons and the quantity of samples were evaluated to analyze the ANN performance for training. An important observation is about how climatic samples were obtained and analyzed.

The climatic database was provided by INPE (*Instituto Nacional de Pesquisas Espaciais*), with a project known as SONDA. All data were taken in Joinville, located in Brazil. Another important observation is that all the simulations executed have the same initial weights vector. Having the equivalent model, the PV can be simulated for any climatic condition, and its curves of current and power versus voltage are built. Through the built curves the values of Maximum Power Point can be extracted, along with its voltage at this point (V_{mpp}). Now, the database is complete, being composed by the climatic variables and its MPP, desired as output.

Once all the climatic samples and their respective MPP values are available, the next step is how to classify which input samples could be used on ANN training phase and its validation. Since the climatic variables changes through the year, a good approach is to use variables taken along the four seasons of weather.

That being said, the first classification was done choosing samples from the first day of one month and each season. The months chosen were January, April, July and October, having then the input-output pair for summer, autumn, winter and spring seasons for southern hemisphere, respectively. Another classification for ANN samples is separating them by two days per season, one entire month of samples per season, one entire year (2009) and two entire years (2009 and 2011). This data classification is not the best method to select data for ANN training, where probably a better statistical analysis could be used in order

to achieve a better classification of the samples. However, the intent of this work is to show that the ANN is able to be trained even with a non-optimized amount of input-output data, since this data has variability enough to represent the behavior of the system.

To validate ANN results, the values for desired output were also built by simulations in MATLAB® obtained with climatic values not used previously, with samples from 2012 year. Two activation functions were used: the sigmoid function and the hyperbolic tangent function. The number of neurons was fixed in 4, 5, 8 and 9 neurons, where this values were determined after some initial simulations, varying the number of neurons in the hidden layer in a range of 2 to 10. The best four results were taken to compose the final results. To better understand how the data results were obtained, a sampling tree with collected data is shown in Figure 4.

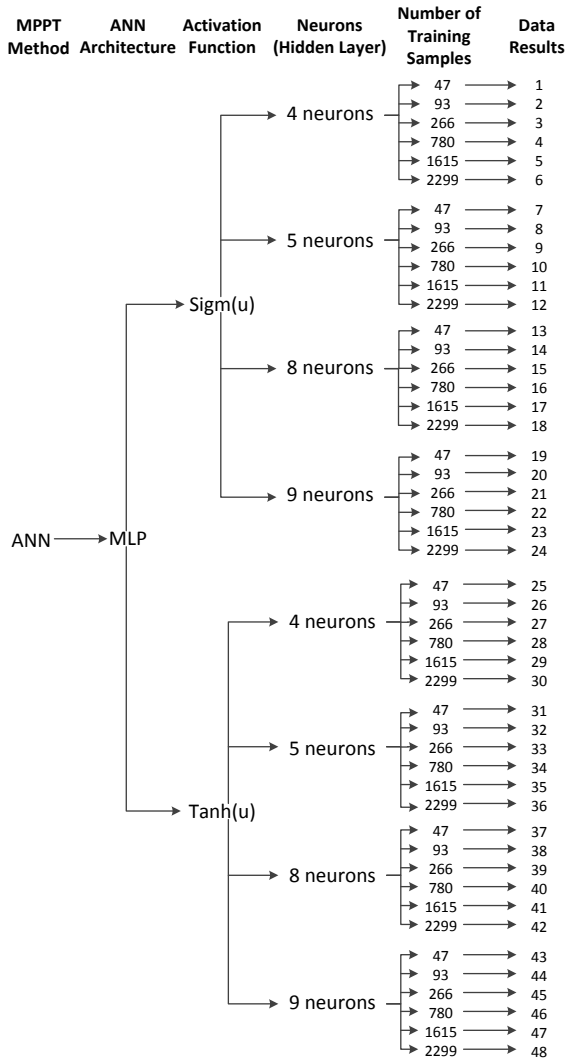

Figure 4: Sampling tree for building ANN results.

To analyze the ANN methods, the data results were composed by three factors: the final *mse* (mean square error) obtained on validation phase, the number of epochs "k" taken by the ANN to converge into a solution (the tolerance was fixed in 10^{-3} for the *mse* on the training phase) and the time for algorithm convergence. The results for both functions varying the number of neurons and the number of training samples can be observed in Figures 5, 6, 7 and 8. Figures 5 and 6 reflect the behavior for the number of epochs, while Figures 7 and 8 show the results for the time for executing the algorithm.

The first observation is that the hyperbolic tangent presents better values than the sigmoid functions, in terms of number of epochs for convergence and time for algorithm execution, independent of the number of neurons or the number of training samples. Another important tendency is that in general the time for algorithm execution and the number of epochs for convergence decreases as the number of training samples increases. The behavior for the neurons quantity also can be evaluated. As the number of neurons increases, initially there is a reduction on time for algorithm execution. But after, a small increment is observed.

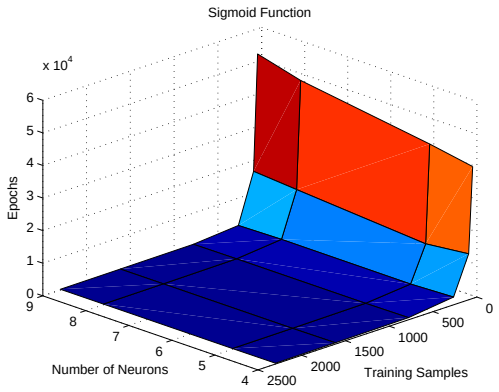

Figure 5: Number of training epochs for sigmoid function.

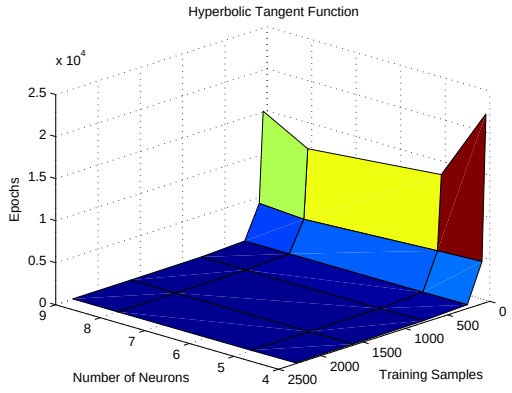

Figure 6: Number of training epochs for hyperbolic tangent function.

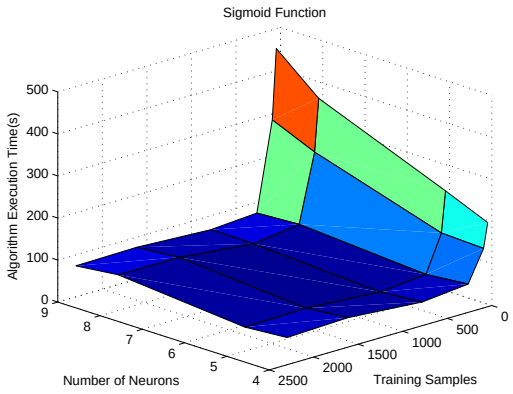

Figure 7: Time to execute the training for the sigmoid function.

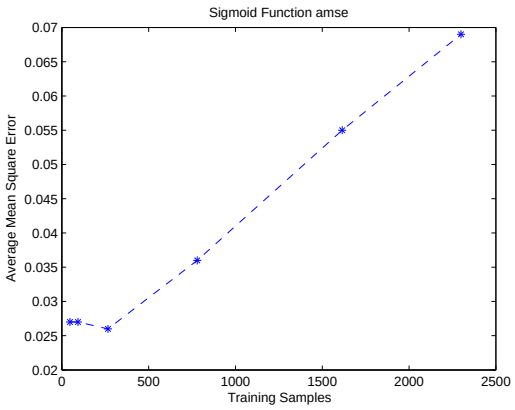

Figure 9: Average Mean Square Error for sigmoid function.

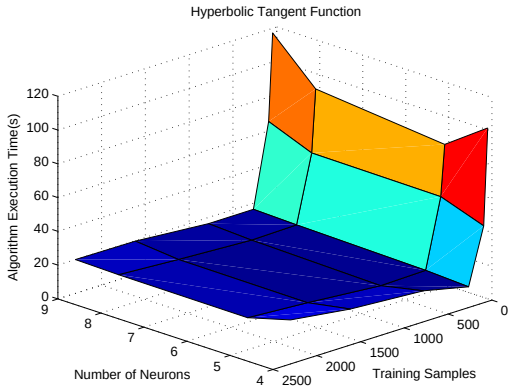

Figure 8: Time to execute the training for the hyperbolic tangent function.

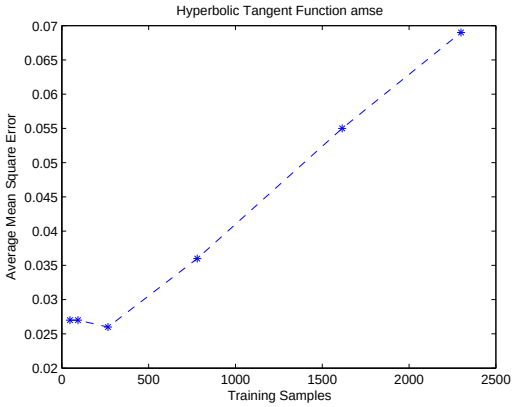

Figure 10: Average mean square error for hyperbolic tangent function.

At last, Figure 9 and Figure 10 are representing the behavior for the average mean square error (*amse*). The mean square error is the sum of all square error values, for each output sample, divided by the number of samples, as shown in Equation 8. As the values of *mse* don't vary substantially with the neuron numbers, but only with the training samples, was chosen to take the average of the *mse* for number of neurons for each function.

$$mse = \sum_{s=1}^{n} \frac{(y^{(s)} - y_d^{(s)})^2}{n} \tag{8}$$

This average of the *mse* for number of neurons and each function resulted in the *amse*, a value which represents the *mse* dependent only with the number of samples, and almost constant for any number of neurons. The *amse* clearly increases with the number of samples used for training, for both functions. Another interesting fact is that the total mean of *mse* for each function is bigger for hyperbolic tangent function than the sigmoid function, but nevertheless the difference between the *amse* for the two functions is sufficiently small.

4 Conclusions

In this work, the comparison between different ANN topologies was shown, along the parameters obtainment for the PV. It could be seen that the number of epochs on training phase decreases when increasing the number of training samples. The time for training algorithm is also smaller as the number of samples increases, but after a given value, the cost-benefit for using bigger samples quantity decreases.

Higher number of neurons can also increases the execution time of the algorithm. But using a small number of neurons is not recommended, because the algorithm may not achieves the convergence, or may take a longer time to converge. Values of *mse* increases as the number of training samples increases, and this phenomena is known as overtraining, as described in (Balzani e Reatti, 2005), and observed on the results. The hyperbolic tangent function has a superior computational performance, but it also has a slightly increased error than the sigmoid function. But as seen in results, the smaller computational efforts compensates this very small difference on ANN

mse for PVs.

The techniques shown in this paper could be used to recognize the PV parameters and extract the maximum power for a given PV. It's important to note that, having one system to find the PV parameters and another one to realize the MPPT, since the PV parameters may change with time, this system could adapt over the years, resulting in a good advantage of this model, especially when the system is used in remote places with difficulties of maintenance.

Acknowledgement

Special thanks for INPE (*Instituto Nacional de Pesquisas Espaciais*) for all provided climatic data and for CAPES (*Coordenação de Aperfeiçoamento de Pessoal de Nível Superior*).

References

Balzani, M. e Reatti, A. (2005). *Neural Network Based Model of a PV Array For The Optimum Performance Of PV System*, Research in Microelectronics and Electronics.

Bastos, R. F., Moçambique, N. E. M., Machado, R. Q. e Aguiar, C. R. (2012). *Rede Neural Artificial Aplicada Na Busca Do Ponto De Máxima Potência Em Painéis Fotovoltaicos*, XIX CBA.

Carrijo, D. R., Ferreira, R. S., Guimarães, S. C. e Camacho, J. R. (2010). *Uma Proposta De Técnica De Rastreamento Do Ponto De Máxima Potência De Um Painel Fotovoltaico*, XVIII CBA.

Esram, T. e Chapman, P. L. (2007). *Comparison of Photovoltaic Array Maximum Power Point Tracking Techniques*, IEEE Transactions On Energy Conversion.

Hadji, S., Gaubert, J.-P. e Krim, F. (2011). *Genetic algorithms for maximum power point tracking in photovoltaic systems*, European Conference on Power Electronics and Applications.

Mahdi, A. J., Tang, W. H. e Wu, Q. H. (2010). *Improvement of a MPPT Algorithm for PV Systems and Its Experimental Validation*, International Conference on Renewable Energies and Power Quality.

Mashaly, H. M., Sharaf, A. M., Mansour, M. e El-Sattar, A. A. (1994). *A Photovoltaic Maximum Power Tracking Using Neural Networks*, IEEE CCA.

Villalva, M. G., Gazoli, J. R. e Filho, E. R. (2009). *Comprehensive Approach to Modeling and Simulation of Photovoltaic Arrays*, IEEE Transactions On Power Electronics.

Zagrouba, M., Sellami, A., Bouaïcha, M. e Ksouri, M. (2010). *Identification of PV solar cells and modules parameters using the genetic algorithms: Application to maximum power extraction*, Solar Energy.