A COMPARISON OF PSO AND BACKPROPAGATION FOR TRAINING RBF NEURAL NETWORKS FOR IDENTIFICATION OF A POWER SYSTEM WITH STATCOM

Salman Mohaghegi, Yamille del Valle, Ganesh K. Venayagamoorthy and Ronald G. Harley

ABSTRACT

Backpropagation algorithm is the most commonly used algorithm for training artificial neural networks. While being a straightforward procedure, it suffers from extensive computations, relatively slow convergence speed and possible divergence for certain conditions. The efficiency of this method as the training algorithm of a *Radial Basis Function Neural Network* (RBFN) is compared with that of Particle Swarm Optimization, for neural network based identification of a small power system with a Static Compensator. The comparison of the two methods is based on the convergence speed and robustness of each method.

1. INTRODUCTION

Artificial neural networks based closed loop controllers have been proposed in order to improve the dynamic and transient performance of nonlinear systems [1]. These are techniques which can provide robust and adaptive control for a highly nonlinear and non-stationary system in the presence of noise and uncertainties. A neural network based identifier (neuroidentifier) which serves as the model of the plant (process) to be controlled, is required in the majority of these designs in order to ensure an adaptive control performance. Training such a neuroidentifier is an important aspect of its design.

The backpropagation (BP) algorithm is the most commonly used technique for training neural networks. It is an approximation of the *Least Mean Square* (LMS) algorithm, which is based on the steepest descent method. While the BP technique follows a straightforward and well-established algorithm, there are some disadvantages associated with it. The method of backwards calculating weights does not seem to be biologically plausible. Neurons do not seem to work backward to adjust their synaptic weights [2]. Furthermore, it suffers from extensive calculations and therefore in most of the cases has a slow convergence speed [3].

Particle Swarm Optimization (PSO) can be a solution to this problem. It is a population based stochastic optimization technique developed by J. Kennedy and R. Eberhart in 1995. It models the cognitive as well as the social behavior of a flock of birds (solutions) which are flying over an area (solution space) in search of food (optimal solution) [4].

PSO has been applied to improve neural networks in various aspects, such as network connection weights, network architecture and learning algorithms. In recent years, there have been several papers reporting on the replacement of the backpropagation algorithm by PSO for some neural network structures [5]-[7]. This paper investigates the efficiency of PSO and BP in terms of convergence speed and the robustness for training a *Radial Basis Function Neural Network* (RBFN) on a power system identification problem.

2. NEUROIDENTIFIER STRUCTURE

Figure 1 shows the schematic diagram of the power system (plant) to be controlled. It is a single machine infinite bus (SMIB) with a *Static Compensator* (STATCOM) connected in the middle of the transmission line. The generator is modeled together with its automatic voltage regulator (AVR), exciter, governor and turbine dynamics all taken into account. The generator is a 37.5 MVA, 11.85 kV (line voltage) machine. Parameters used for the entire system can be found in [8]. The system is simulated in the PSCAD/EMTDC environment.

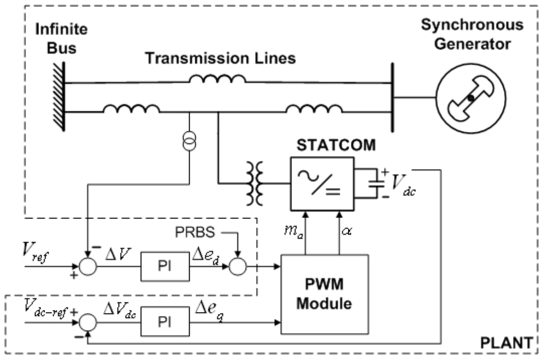

Figure 1. Block diagram of the plant and the controller.

The STATCOM is a power electronics based inverter which is connected to the network in parallel and can control the voltage at the point of connection [9]. Two conventional PI controllers control the line voltage at the point of connection to the network as well as the STATCOM DC link voltage (Fig. 1).

The generator, STATCOM, their internal controllers and the transmission lines are considered as the *plant* to be controlled, with the control signal Δe_d as the input and the line

S. Mohagheghi and Y. del Valle are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332-0250 USA (e-mail: salman@ece.gatech.edu).

G.K. Venayagamoorthy is with the Real-Time Power and Intelligent Systems Laboratory, Department of Electrical and Computer Engineering, University of Missouri-Rolla, MO 65409-0249 USA (email: gkumar@ieee.org).

R.G. Harley is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332-0250 USA (e-mail: rharley@ece.gatech.edu), and a Professor Emeritus at the University of KwaZulu-Natal, South Africa .

voltage error ΔV as the output. A neuroidentifier is trained in order to learn the dynamics of the plant and track its output during normal operating conditions as well as during dynamic and transient disturbances. The training procedure of the neuroidentifier is explained in details in the authors' previous work [10].

A RBFN with Gaussian activation function and a single hidden layer with 25 neurons is used for the structure of the neuroidentifier. The neural network receives the plant input and output at time steps $(t-1)$, $(t-2)$ and $(t-3)$, and generates an estimate of the plant output at time *t*. (Fig. 2)

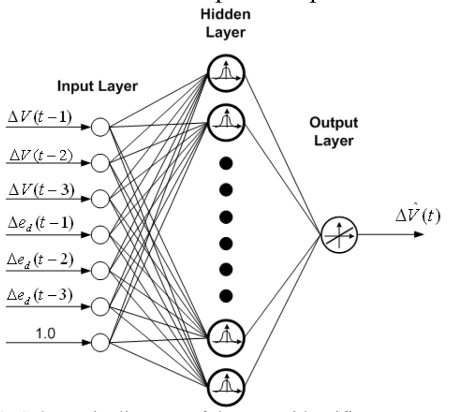

Figure 2. Schematic diagram of the neuroidentifier.

Training samples are collected by applying a *Pseudo-Random Binary Signal* (PRBS) disturbance (Fig. 3) to the plant input and storing the corresponding plant output (forced training). Batch mode training is then carried out in order to update the neural network weights.

The centers of the neural network are determined offline using a batch mode clustering method, in order to avoid the computational complexity of the online recursive clustering method [3]. The training algorithms explained in Section 3 are therefore only applied and compared for updating the output layer weight matrix (Fig. 2).

3. TRAINING ALGORITHMS

Two different methods are compared for updating the output weight matrices of the neuroidentifier: backpropagation and PSO. Both methods are applied offline and are compared in terms of the *Mean Square Error* (MSE):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}|e_i|^2 \tag{1}$$

where *N* is the number of data samples in each epoch and *e* is the instantaneous error between the actual and estimated values of the output.

A. Backpropagation Algorithm

The backpropagation (BP) algorithm was developed by Paul Werbos in 1974. Based on LMS algorithm, BP applies a weight correction to the neural network connection weights which is proportional to the partial derivative of the error function [3]. This adjustment to the weights is in the negative direction of the gradient of the error (steepest descent).

The error function is defined as:

$$E(t) = \frac{1}{2}.|e(t)|^2 \tag{2}$$

where $e(t)$ is the error value, i.e. the difference between the actual output of the plant and the estimated neuroidentifier output. The neuroidentifier weights are adjusted according to:

$$W(t+1) = W(t) - \eta.\frac{\partial E(t)}{\partial W(t)} \tag{3}$$

where η is the learning rate parameter. A large learning rate might lead to oscillations in the convergence trajectory, while a small learning rate provides a smooth trajectory at the cost of slow convergence speed.

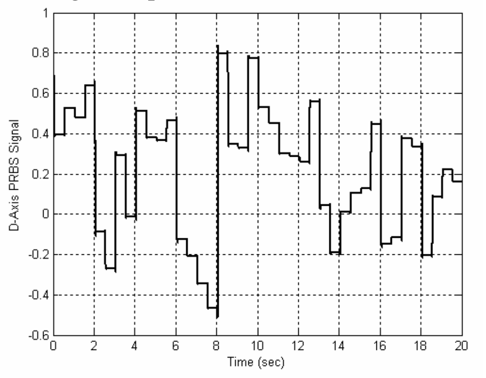

Figure 3. PRBS signal used for forced training.

Backpropagation can be applied in two modes: sequential and batch mode. The former is the online mode of training a neural network, where weight updating is performed after the presentation of each training sample, while in the latter the weight matrices are updated after the presentation of all the training samples that constitute an epoch [3].

B. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population (swarm) based optimization tool. Every single solution (called a particle) "flies" over the solution space in search for the optimal solution. The particles are evaluated using a fitness function to see how close they are to the optimal solution [4]. Particles have a tendency to duplicate their individual past behavior that has been successful (cognition) as well as to follow the successes of the other particles (socialization).

The neural network weight matrix is rewritten as an array to form a particle. Particles are then initialized randomly and updated afterwards according to (4):

$$\begin{aligned} &W(t+1) = W(t) + \Delta W(t+1) \\ &\Delta W(t+1) = w.\Delta W(t) + c_1.rand().[pBest(t) - W(t)] \\ &\qquad + c_2.Rand().[gBest(t) - W(t)] \end{aligned} \tag{4}$$

where w, c_1, c_2 are inertia, cognitive and social acceleration constants respectively.

For every specific particle, $pBest$ is the best solution that the particle has achieved so far and indicates the tendency of the individual particles to replicate their corresponding past behaviors that have been successful. $gBest$ is the global best solution so far, i.e. the best solution that any particle (in the whole population) has achieved so far. This quantity indicates the tendency of the particles to follow the success of others. [11], [12]

Another important parameter associated with PSO, is the maximum velocity $V_{\max}$, which determines the resolution or fineness with which the search space is searched. A large value might cause the particles to fly past good solutions, while a small number can cause the particles to get trapped in the local optima [11].

Selection of the constant parameters, the population size, neighborhood size and suchlike are problem dependent and for this specific problem will be explained in the next section.

4. SIMULATION RESULTS

The PRBS disturbance applied to the plant has a combination of three different frequencies *0.5*, *1* and *2* Hz, and its magnitude is selected in a way that it causes $\pm 5\%$ deviations in the plant output. Forced training is applied to the system for 100 sec. The PSCAD time step is chosen to be *100* µs, while the neural network sampling time is *5* ms.

Batch mode training is first applied for training the neuroidentifier using the backpropagation algorithm. Various learning gains are examined in the range of [0.001,0.5], and the best results are achieved with the learning gain of *0.211* (MSE is *0.0084* after *20* epochs). Figure 5 shows the BP results for the best case.

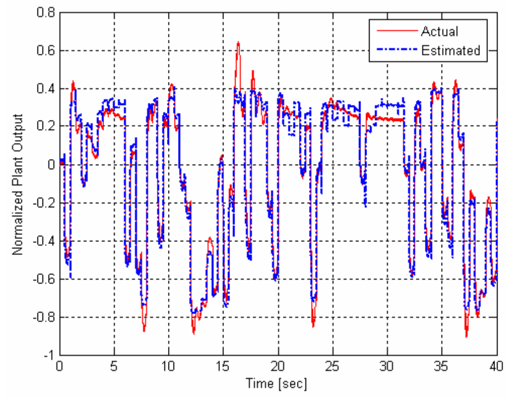

Figure 5. Actual and estimated values of the plant output during forced training using BP (best case).

Parameters of PSO are also fine tuned in order to get the best results. Table I shows the range in which each parameter is searched, as well as the optimum values for each parameter. The optimum values for the PSO are derived after an extensive search over the ranges defined in Table I.

Table I. Parameters for PSO.

Parameter	**Range**	**Optimum Value**
Number of Particles	[15,30]	30
Inertia Constant (Base Value)	[0.5,0.9]	0.5
Cognitive Acceleration Constant	[1,3]	3
Social Acceleration Constant	[3,1]	1
Maximum Velocity	[1,20]	16
Maximum Search Space Range	[1,40]	21

It should be noted that the inertia constant that is applied in this study is as follows:

$$w = w_{base} + \frac{2 \times rand() - 1}{2} \qquad (5)$$

where w_{base} is the value mentioned in Table I.

Figure 6, which shows the best case using optimum parameters for PSO, indicates a MSE equal to *0.0027* after *20* epochs, (three times less than backpropagation algorithm). A further comparison of the best cases of training the neuroidentifier with BP and PSO appears in Figure 7.

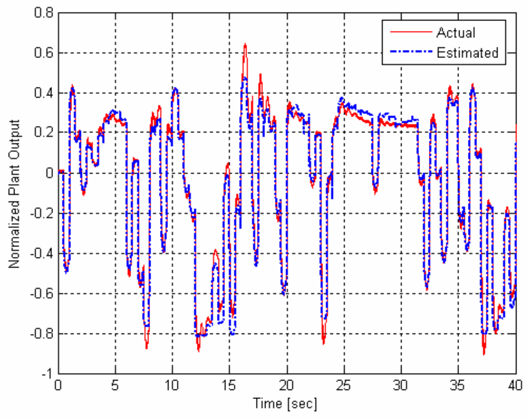

Figure 6. Actual and estimated values of the plant output during forced training using PSO (best case).

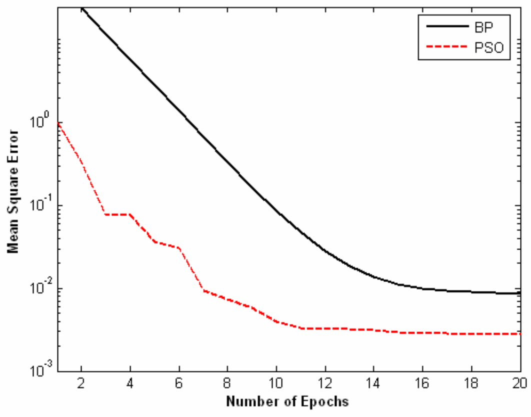

Figure 7. Mean square error with BP and PSO training algorithms.

Since PSO is a stochastic optimization method, several sets of simulations are carried out (50 trials) with the same set of optimum parameters in order to provide statistical results. The same number of trials considering the optimum learning gain was performed in the case of backpropagation algorithm. Table II shows the comparison between the two methods.

It is also observed during the trials that if the number of the particles used in PSO is reduced, for the sake of improving the computational effort, the performance is slightly degraded as shown in Table II.

Table II. Statistical results.

Statistics	**PSO**		**BP**
Number of Particles	30	20	----
Minimum MSE	0.0027	0.0036	0.0084
Maximum MSE	0.2266	0.1203	0.0566
Mean MSE	0.0222	0.0238	0.0225

A comparison between the two training algorithms indicates that even though the minimum MSE in PSO is considerably smaller (about three times less) than the minimum MSE in BP, the performance of PSO, on average, is only slightly better. The authors believe this is due to the fact that for a RBFN, the centers of the hidden layer are determined using a clustering method, irrespective of the training method chosen for updating the output synaptic weight matrix. Therefore there are less degrees of freedom for the training methods. A comparison of the PSO and BP for a neuroidentifier using *Multilayer Perceptron Neural Networks* (MLPN) for the same power system supports this assertion (see Appndix).

Extensive simulations showed that PSO algorithm is especially robust with respect to wide-range variations on its parameters. For this specific application, no convergence problems are observed in any of the 2,500 combinations generated for the different values of inertia constant, cognitive acceleration, social acceleration and maximum velocity. This shows a major advantage over BP algorithm in which the learning gain has to be accurately determined.

5. CONCLUDING REMARKS

Two different learning algorithms were applied in this paper for training a radial basis function network based neuroidentifier: Backpropagation (BP) and Particle Swarm Optimization techniques (PSO). The training algorithms have been applied only for updating the output synaptic weight matrix, and in both cases the centers and widths of the neurons in the hidden layer are derived using an offline clustering method.

Backpropagation learning algorithm is the most commonly used technique for updating neural network weight parameters. While being straightforward, it has a slow convergence speed and might at times diverge. It also requires extensive calculations if the size of the network increases and it may be difficult to implement when no gradient information is available for all activation functions.

On the other hand, PSO algorithm has shown to have several advantages, both in terms of robustness and the efficiency in finding the optimal weights for the RBFN neuroidentifier. Furthermore, the algorithm has proven to be efficient even for a reduced number of particles, thus the computational effort is comparable and even less significant than in the case of backpropagation. Statistical results are provided that confirm PSO as a reliable algorithm for training such a neural network.

6. APPENDIX
COMPARISON OF BP AND PSO FOR MLPN

The efficiency of the two training methods is compared for a MLPN based neuroidentifier for the power system shown in Fig. 1. The MLPN neuroidentifier has 16 neurons in the hidden layer. Table III summarizes the results.

Table III. Statistical results for MLPN neuroidentifier.

Statistics	**PSO**	**BP**
Number of Particles	15	----
Minimum MSE	0.0011	0.0095
Maximum MSE	0.0268	0.0791
Mean MSE	0.0059	0.0366

7. REFERENCES

[1] K.S. Narendra and K. Parthasarathy, “Identification and Control of Dynamical Systems Using Neural Networks”, *IEEE Transactions on Neural Networks*, Vol. 1, No. 1, March 1990, pp 4-27.

[2] S.V. Kartalopoulos, *Understanding Neural Networks and Fuzzy Logic*, IEEE Press, 1996, ISBN 0-7803-1128-0.

[3] S.S. Haykin, *Neural Networks- A Comprehensive Foundation*, Prentice-Hall, 2nd Edition, 1998, ISBN 0-1327-3350-1.

[4] X. Hu, Y. Shi and R. Eberhart, “Recent Advances in Particle Swarm”, *Proceedings of the Congress on Evolutionary Computation*, Portland, OR, USA, June 19-23, 2004, Vol. 1, pp 90-97.

[5] F. Van den Bergh and A.P. Engelbrecht, “Cooperative Learning in Neural Networks using Particle Swarm Optimizers”, *South African Computer Journal*, Vol. 26, 2000, pp 84-90.

[6] R.C. Eberhart and Y. Shi, “Evolving Artificial Neural Networks”, *Proceedings of the International Conference on Neural Networks and Brain*, 1998, Beijing, China, pp 5-13.

[7] V.G. Gudise and G.K. Venayagamoorthy, “Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithms for Neural Networks”, *Proceedings of the Swarm Intelligent Symposium*, Indianapolis, IN, USA, April 24-26, 2003, pp 110-117.

[8] S. Mohagheghi et al, “Adaptive Critic Design Based Neurocontroller for a STATCOM Connected to a Power System”, Proceedings of the IEEE Industry Applications Society Annual Conference, Salt Lake City, UT, USA, October 12-16, 2003, pp 749-754.

[9] N.G. Hingorani and L. Gyugyi, *Understanding FACTS, Concepts and Technology of Flexible AC Transmission Systems,* IEEE, New York 1999, ISBN 0-7803-3455-8

[10] S. Mohagheghi et al, “An Adaptive Neural Network Identifier for Effective Control of a Static Compensator Connected to a Power System”, *Proceedings of the IEEE-INNS International Joint Conference on Neural Networks* (IJCNN’03), Portland, OR, USA, July 20-24, 2003, pp 2964-2969.

[11] R.C. Eberhart and Y. Shi, “Particle Swarm Optimization: Developments, Applications and Resources”, Proceedings of the 2001 Congress on Evolutionary Computation, May 27-30, 2001, Vol. 1, pp 81-86.

[12] X. Hu, Y. Shi and R.C. Eberhart, “Recent Advances in Particle Swarm” Proceedings of the 2004 Congress on Evolutionary Computation, June 19-23, 2004, Vol. 1, pp 90-97.