Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 197690, 9 pages
http://dx.doi.org/10.1155/2013/197690

Research Article

Neural Model with Particle Swarm Optimization Kalman Learning for Forecasting in Smart Grids

Alma Y. Alanis,[1] Luis J. Ricalde,[2] Chiara Simetti,[3] and Francesca Odone[3]

[1] *CUCEI, Universidad de Guadalajara, Apartado Postal 51-71, Colonia Las Aguilas, 45080 Zapopan, JAL, Mexico*
[2] *UADY, Faculty of Engineering, Avenida Industrias no Contaminantes por Periferico Norte, Apartado Postal 115 Cordemex, Merida, Yuc, Mexico*
[3] *DISI, Università degli Studi di Genova, Via Dodecaneso 35, 16146 Genova, Italy*

Correspondence should be addressed to Alma Y. Alanis; almayalanis@gmail.com

Received 29 March 2013; Accepted 27 May 2013

Academic Editor: Yudong Zhang

This paper discusses a novel training algorithm for a neural network architecture applied to time series prediction with smart grids applications. The proposed training algorithm is based on an extended Kalman filter (EKF) improved using particle swarm optimization (PSO) to compute the design parameters. The EKF-PSO-based algorithm is employed to update the synaptic weights of the neural network. The size of the regression vector is determined by means of the Cao methodology. The proposed structure captures more efficiently the complex nature of the wind speed, energy generation, and electrical load demand time series that are constantly monitorated in a smart grid benchmark. The proposed model is trained and tested using real data values in order to show the applicability of the proposed scheme.

1. Introduction

The limited existing reserves of fossil fuels and the harmful emissions associated with them have led to an increased focus on renewable energy applications in recent years. The first steps on integrating renewable energy sources began with hybrid wind and solar systems as complementing sources and as a solution for rural applications and weak grid interconnections. Further research has implemented hybrid systems including several small scale renewable energy sources as solar thermal, biomass, fuel cells, and tidal power. Since the production costs for photovoltaic and wind turbine applications have considerably reduced, they have become the primary choice for hybrid energy generation systems. The future of energy production is headed towards this scheme of integration of renewable energy sources with existing conventional generation systems with a high degree of measurement, communications, and control. This integration is defined as a smart grid. This new scheme increases the power quality since the production becomes decentralized and is the main reason for which institutions have increased the research on this concept [1]. Microgrids integrate small scale energy generation systems mainly from renewable energy and implement complex control technologies to improve the flexibility and reliability of the power system. The design of these systems integrates a distributed power generation system and a management unit composed of a communication network which monitor and controls the interconnection between energy sources, storage devices, and electrical loads.

Among renewable energy sources, wind energy is the one with the lowest cost of electricity production [2]. However, in practice the integration of wind energy into the existing electricity supply system is a real challenge because its availability mainly depends on meteorological conditions, particularly on the magnitude of the wind speed, which cannot directly be changed by human intervention. For this reason, it is important to have a reliable estimation of wind velocity and direction which directly affects the energy generation. Integration of the forecast of wind speed and output power is a good way to improve the performance in scheduling for smart grids [3]. Wind prediction is not an easy task; the wind has a stochastic nature with high rate of change. Wind speed time series present highly nonlinear behavior with no typical patterns and a weak seasonal character [4].

Several methods have been proposed to accomplish wind characteristics forecasting like numerical weather prediction systems, statistical approaches, and artificial neural networks using feedforward or recurrent structures [2, 5–9]. In [6], a linear-time-series-based model relating the predicted interval to its corresponding one and data covering a temporal span of two years is developed. The statistical approaches have the advantage of low cost since they only require historical data; on the other hand, the accuracy of the prediction drops for long time horizons. Artificial intelligence methods are more suitable for short-term predictions; these methods are based on time series historical data in order to build a mathematical model which approximates the input-output relationship. Time-series-based models include the autoregressive (AR) and the autoregressive moving average (ARMA) models which predict the average wind speed for one step ahead [9].

For wind generation systems, artificial neural networks (ANN) have been considered as a convenient analysis tool for wind energy systems forecasting and control applications due to the simplicity of the model and the accuracy of the results for nonlinear and stochastic models and have been implemented in several practical applications [8]. In [10], a merged neuro-fuzzy system is developed as a universal approximator in order to estimate the state of charge in a battery bank. For wind generation systems, ANN have been considered as a convenient analysis tool for wind forecasting due to the simplicity of the model and the accuracy of the results.

In [8], a recurrent neural network is applied to forecast the output power of wind generators based on wind speed prediction using one year of historical data achieving from hour-ahead to day-ahead predictions with errors ranging from 5% for one hour horizon to 20% for one day ahead forecasting. In [5], local recurrent neural networks are implemented to forecast wind speed and electrical power in a wind park with a seventy-two hour ahead forecast and obtaining a better performance in comparison with static network approaches.

ANN have been previously implemented for wind power short-term predictions, outperforming other classical methods due to the fast learning algorithm which enables on-line implementations and the versatility to vary the prediction horizon [7]. Due to their nonlinear modeling characteristics, neural networks have been successfully applied in control systems, pattern classification, pattern recognition, and time series forecasting problems. There are several previous works that use artificial neural networks to predict wind time series [2, 4, 11]. The best well-known training approach for recurrent neural networks (RNN) is the back propagation through time [12]. However, it is a first-order gradient descent method, and hence its learning speed could be very slow [13]. Another well-known training algorithm is the Levenberg-Marquardt one [14]; its principal disadvantage is that it is not guaranteed that it will find the global minimum and its learning speed could be slow too, and this depends on the initialization. In past years, extended-kalman-filter- (EKF-) based algorithms have been introduced to train neural networks [15, 16]. With the EKF-based algorithm, the learning convergence is improved [13]. The EKF training of neural networks, both feedforward and recurrent ones, has proven to be reliable for many applications over the past ten years [16]. However, EKF training requires the heuristic selection of some design parameters which is not always an easy task [11, 15].

During the past decade, the use of evolutionary computation in engineering applications has increased. Evolutionary algorithms apply adaptation and stochastically in optimization problems in schemes such as evolutionary programming, genetic algorithms, and evolution strategies [17]. Particle swarm optimization (PSO) technique, which is based on the behavior of a flock of birds or school of fish, is a type of evolutionary computing technique [18]. The PSO algorithm uses a population of search points that evolve in a search space using a communication method to transfer the acquired experience from best solutions. This algorithm has several advantages like the simplicity of the updating law, faster convergence time, and less complexity on the reorganization of the population. The PSO methods also have emerged as an excellent tool to improve the performance of neural network learning process [19]. In [20], a PSO learning rule for a neural network is implemented using FPGA for dynamic system identification. In [21], the PSO algorithm is extended to multiple swarms in a neuro-fuzzy network with good results in forecasting applications. It has been shown that the PSO training algorithm takes fewer computations and is faster than the BP algorithm for neural networks to achieve the same performance [18].

In this paper, we propose the use of PSO for tuning the parameters of EKF training algorithm. This scheme is a new one and is suitable for data modeling in smart grids since the forecasting horizon satisfies the requirements for several applications in the grid operation. The length of the regression vector is determined using the Cao methodology which is an improvement to the false neighbors approach [22]. The applicability of this architecture is illustrated via simulation using real data values for electric load demand (ELD), wind speed (WS), and wind energy generation (WEG) in order to show the potential applications in forecasting for energy generation in smart grid schemes.

The remainder of the paper is organized as follows. Section 2 is devoted to describing the neural model, based on the recurrent multilayer perceptron (RMLP), where the training phase relies on an extended Kalman filter which is able to deal with the nonlinearity of the model, and the initialization of the system is based on a particle swarm optimization strategy. Section 3 reports the experimental analysis of the proposed method applied to the problem of predicting variables in smart grids. Finally Section 4 includes the conclusions and future work.

2. Neural Identification

In this paper for the neural model identification, the recurrent multilayer perceptron is chosen, and then the neural model structure problem reduces to dealing with the following issues: (1) selecting the inputs to the network and (2) selecting the internal architecture of the network.

The structure selected in this paper is NNARX [14] (acronym for neural network autoregressive external input);

the output vector for the artificial neural network is defined as the regression vector of an autorregressive external input linear model structure (ARX) [15].

It is common to consider a general nonlinear system; however, for many control applications it is preferred to express the model in an affine form, which can be represented by the following equations:

$$y(k+1) = f\left(y(k), y(k-1), \ldots, y\left(k-q+1\right)\right), \quad (1)$$

where q is the dimension of the regression vector. In other words, a nonlinear mapping f exists, for which the present value of the output $y(k+1)$ is uniquely defined in terms of its past values $y(k), \ldots, y(k-q+1)$ and the present values of the input $u(k)$.

Considering that it is possible to define

$$\phi(k) = \left[y(k), \ldots, y\left(k-q+1\right)\right]^T \quad (2)$$

which is similar to the regression vector of a ARX linear model structure [14], then the nonlinear mapping f can be approximated by a neural network defined as

$$y(k+1) = \varphi\left(\phi(k), w^*\right) + \varepsilon, \quad (3)$$

where w^* is an ideal weight vector, and ε is the modeling error; such neural network can be implemented on predictor form as

$$\hat{y}(k+1) = \varphi\left(\phi(k), w\right), \quad (4)$$

where w is the vector containing the adjustable parameters in the neural network.

The neural network structure used in this work is depicted in Figure 1, which contains sigmoid units only in the hidden layer; the output layer is a linear one. The used sigmoid function $S(\cdot)$ is defined as a logistic one as follows:

$$S(\varsigma) = \frac{1}{1+\exp\left(-\beta\varsigma\right)}, \quad \beta > 0, \quad (5)$$

where ς is any real value variable.

2.1. EKF Training Algorithm. Kalman filter (KF) estimates the state of a linear system with additive state and output white noise. Kalman filter algorithm is developed for a linear, discrete-time dynamical system. For KF-based neural network training, the network weights become the states to be estimated. Due to the fact that the neural network mapping is nonlinear, an EKF type is required [15].

Consider a nonlinear dynamic system described by the next model in state space

$$\begin{aligned} w(k+1) &= f(k, w(k)) + v_1(k), \\ y(k) &= h(k, w(k)) + v_2(k), \end{aligned} \quad (6)$$

where $v_1(k)$ and $v_2(k)$ are zero-mean, white noises with covariance matrices given by $Q(k)$ and $R(k)$, respectively. $f(k, w(k))$ denotes the nonlinear transition matrix function.

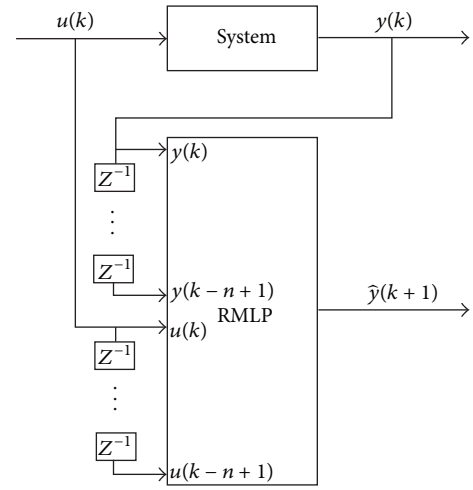

Figure 1: Neural network structure.

The basic idea of the extended Kalman filter is to linearize the state space model 4 at each time instant around the most recent state estimate, which is taken to be $\hat{w}(k)$. The training goal is to find the optimal weight values which minimize the prediction error. The modified extended kalman filter (EKF) algorithm is defined by

$$\begin{aligned} \hat{w}(k+1) &= \hat{w}(k) + K(k)\left[y(k) - \hat{y}(k)\right], \\ K(k) &= P(k) H^T(k) M(k), \\ P(k+1) &= P(k) - K(k) H(k) P(k) + Q(k) \end{aligned} \quad (7)$$

with

$$\begin{aligned} M(k) &= \left[R(k) + H(k) P(k) H^T(k)\right]^{-1}, \\ e(k) &= y(k) - \hat{y}(k), \end{aligned} \quad (8)$$

where $e(k) \in \mathfrak{R}$ is the respective approximation error, $P \in \mathfrak{R}^{L\times L}$ is the prediction error associated covariance matrix at step k, $w \in \mathfrak{R}^L$ is the weight (state) vector, L is the respective number of neural network weights, y is the system output, $\hat{y}$ is the neural network output, $K \in \mathfrak{R}^L$ is the Kalman gain vector, $Q \in \mathfrak{R}^{L\times L}$ is the state noise associated covariance matrix, $R \in \mathfrak{R}$ is the measurement noise associated covariance, $H \in \mathfrak{R}$ is a vector, in which each entry (H_{ij}) is the derivative of one of the neural network outputs, $(\hat{y})$, with respect to one neural network weight, and (w_j) is defined as follows:

$$H_{ij}(k) = \left[\frac{\partial \hat{y}(k)}{\partial w_j(k)}\right]^T_{w_i(k)=\hat{w}_i(k)}, \quad (9)$$

where $i = 1, \ldots, m$; $j = 1, \ldots, L$. Usually P, Q, and R are initialized as diagonal matrices, with entries $P(0)$, $Q(0)$, and $R(0)$, respectively. It is important to note that for the EKF training algorithm $P(0)$, $Q(0)$, and $R(0)$ are considered as design parameters that are typically heuristically determined; however, in this paper we propose the use of particle swarm optimization for determining such parameters [15].

2.2. PSO Improvement for EKF Training Algorithm. Particle swarm optimization (PSO) is a swarm intelligence technique developed by Kennedy and Eberhart in 1995 [23]. In fact, natural flocking and swarm social behavior of birds and insects inspired the PSO. This technique has been used in several optimization and engineering problems [2, 18, 24]. In the basic PSO technique proposed by Kennedy and Eberhart [23], a great number of particles move around in a multidimensional space and each particle memorizes its position vector and velocity vector as well as the time at which the particle has acquired the best fitness. Furthermore, related particles can share data at the best-fitness time. The velocity of each particle is updated with the best positions acquired for all particles over iterations and the best positions are acquired by the related particles over generations.

To improve the performance of the basic PSO algorithm, some new versions of it have been proposed. At first, the concept of an inertia weight is developed to better control exploration and exploitation in [18, 25]. Then, the research done by Clerc [26] indicated that using a constriction factor may be necessary to insure convergence of the particle swarm algorithm. After these two important modifications in the basic PSO, the multiphase particle swarm optimization (MPSO), the particle swarm optimization with Gaussian mutation, the quantum particle swarm optimization, a modified PSO with increasing inertia weight schedule, the Gaussian particle swarm optimization (GPSO), and the guaranteed convergence PSO (GCPSO) were introduced in [27], respectively.

In this paper, the algorithm proposed in [18] is used in order to determine the design parameters for the EFK-learning algorithm. Initially, a set of random solutions or a set of particles are considered. A random velocity is given to each particle and they are flown through the problem space. Each particle has a memory which is used to keep track of the previous best position and corresponding fitness. The best value of the position of each individual is stored as p_{id}. In other words, p_{id} is the best position acquired by an individual particle during the course of its movement within the swarm. It has another value called the p_{gd}, which is the best value of all the particles p_{id} in the swarm. The basic concept of the PSO technique lies in accelerating each particle towards its p_{id} and p_{gd} locations at each time step. The PSO algorithm used in this paper is defined as follows [18].

(1) Initialize a population of particles with random positions and velocities in the problem space.

(2) For each particle, evaluate the desired optimization fitness function.

(3) Compare the particles fitness evaluation with the particles p_{id} and if the current value is better than the p_{id}, then set p_{id} value equal to the current location.

(4) Compare the best fitness evaluation with the population's overall previous best. If the current value is better than the p_{gd}, then set p_{gd} to the particle's array and index value.

(5) Update the particle's velocity and position as follows.

The velocity of the ith particle of d dimension is given by

$$v_{id}(k+1) = c_0 v_{id}(k) + c_1 \text{rand}_1\left(p_{id}(k) - x_{id}(k)\right) + c_2 \text{rand}_2\left(p_{gd}(k) - x_{id}(k)\right). \tag{10}$$

The position vector of the ith particle of d dimension is updated as follows:

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k), \tag{11}$$

where c_0 is the inertia weight, c_1 is the cognition acceleration constant, and c_2 is the social acceleration constant.

(6) Repeat step (2) until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations or epochs.

In case the velocity of the particle exceeds $V_{\max}$ (the maximum velocity for the particles), then it is reduced to $V_{\max}$. Thus, the resolution and fitness of search depends on the $V_{\max}$. If $V_{\max}$ is too high, then particles will move in larger steps and so the solution reached may not be as good as expected. If $V_{\max}$ is too low, then particles will take a long time to reach the desired solution [18]. The above explained PSO are very suitable models of noisy problems, as the one we are considering.

2.3. Regressor Structure. We now discuss the choice of an appropriate number of delayed signals to be used in the training phase; a wrong number of delayed signals, used as regressors, could have a substantially negative impact on the training process, while a too small number imply that essential dynamics will not be modeled. Additionally, a large number of regression terms increase the required computation time. Also, if too many delayed signals are included in the regression vector, it will contain redundant information. For a good behavior of the model structure, it is necessary to have both a sufficiently large lag space and an adequate number of hidden units. If the lag space is properly determined, the model structure selection problem is substantially reduced. There have been many discussions of how to determine the optimal embedding dimension from a scalar time series based on Takens' theorem [22]. The basic methods, which are usually used to choose the minimum embedding dimension, are (1) computing some invariants on the attractor, (2) singular value decomposition, and (3) the method of false neighbors. However, a practical method to select the lag space is the one proposed by Cao [22] to determine the minimum embedding dimension; it overcomes most of the shortcomings of the above mentioned methodologies, like high dependence from design parameters and high computational cost, among others [22]. Besides, it has several advantages: it does not contain any subjective parameters except for the time-delay embedding; it does not strongly depend on how many data points are available; it can clearly distinguish deterministic signals from stochastic signals; it works well for time series from high-dimensional attractors, and it is computationally

efficient. In this paper, this technique for determination of the optimal regressor structure is used.

Let us consider a time series $x_1, x_2, \ldots, x_n$ and define a set of time-delay vectors as

$$y_i = \left[x_i \;\; x_{i+\tau} \;\; \cdots \;\; x_{i+(d-1)\tau}\right], \quad i = 1, 2, \ldots, N - (d-1)\tau, \tag{12}$$

where d is the embedding dimension. This dimension is determined from the evolution of a function $E(d)$ defined as

$$E(d) = \frac{1}{N - d\tau} \sum_{i=1}^{N-d\tau} \frac{\left\| y_i(d+1) - y_{n(i,d)}(d+1) \right\|}{\left\| y_i(d) - y_{n(i,d)}(d) \right\|}, \quad i = 1, 2, \ldots, N - d\tau, \tag{13}$$

where $n(i,d)$ is an integer such that $y_{n(i,d)}(d)$ is the nearest neighbor of $y_i(d)$ [28]. The minimum embedding dimension $d_0 + 1$ is determined when $E(d)$ stops changing for any d_0.

3. Simulation Results

In this section, two application examples to validate the proposed PSO-EKF learning algorithm are presented. First, the experimental analysis of the proposed method applied to the problem of predicting the wind speed in order to compare the performance with similar approaches [29] is discussed. As a second test for the proposed method, the neural predictor with data obtained for the microgrid in the UADY School of Engineering is implemented in order to evaluate the performance with time series of a different nature but related with the energy production and demand in smart grids.

3.1. Comparison of the PSO Algorithm for Wind Speed Forecasting. In order to evaluate the performance of the PSO algorithm and compare with similar methods, a neural network predictor for wind speed is implemented, on the basis of the proposed training algorithm. The proposed algorithm requires the following methodology.

(1) Define the training set. Training is performed using minute data from the first 3 hours from January 1, 2011 and the testing is performed using the 3 hours subsequent to the data training. Experimental data is taken from [30].

(2) Determine the optimal dimension of the regression vector (1) for dataset of step (1).

(3) Select the neural structure to be used (4).

(4) Train the neural identifier.

(5) Validate the neural identifier using the testing data.

The neural network used is an RMLP trained with an PSO-EKF, whose structure is presented in Figure 1; the hidden layer has 5 units with logistic sigmoid activation functions (5), whose β is fixed in 1 and the output layer is composed of just one neuron, with a linear activation function.

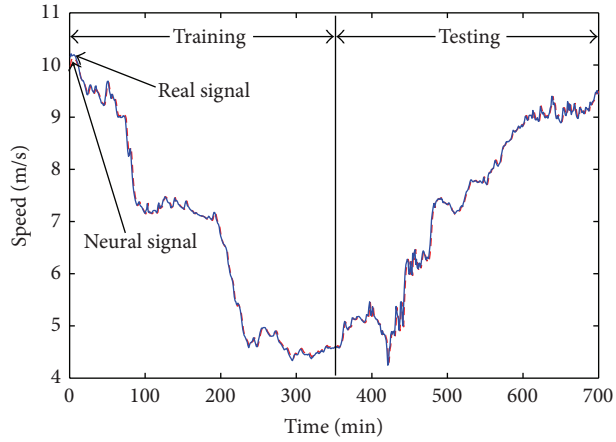

Figure 2: Identification performance for wind speed forecast.

Table 1: Mean value (μ) and standard deviation (σ) for identification error.

	PSO-EKF	EKF	LM
μ	-5.6693×10^{-6}	-1.2547×10^{-5}	1.8354×10^{-4}
σ	0.0749	0.0807	0.0724

The initial values for the covariance matrices (R, Q, P) are determined using the PSO algorithm, with 200 as the maximum number of iterations, 4 generations, 3 particles, and $c_1 = c_2 = 0.1$. The initial values for neural weights are randomly selected. The length of the regression vector is 5 because that is the order of the system, which is determined using the Cao methodology.

The training is performed offline, using a series-parallel configuration; for this case the delayed output is taken from the wind speed. The mean square error (MSE) reached in training is 1.735×10^{-5} in 200 iterations and the mean absolute relative error reached is 0.6912%. Besides, to measure the performance of the neural network, the absolute relative error (ARE) is calculated from

$$\text{ARE} = \left| \frac{y(n) - \hat{y}(n)}{y(n)} \right|, \tag{14}$$

where $\hat{y}(n)$ is the predicted wind speed time series achieved by the network. Simulation results are presented as follows: Figure 2 displays the wind speed time series neural identification, Figure 3 includes the identification error (8), Figure 4 shows the time evolution of mean square error, and Figure 5 displays the absolute relative error obtained with (14).

Figure 6 depicts the comparison detail of the proposed PSO-EKF training algorithm against the classical EKF one and the well-known Levenberg-Marquardt one.

Table 1 includes a comparison between the proposed PSO-EKF learning algorithm, the EKF one and the well-known Levenberg-Marquardt (LM) one.

Results included in Table 1 show that the proposed methodology leads to an improvement of the results. Therefore for the second example only PSO-EKF results are presented.

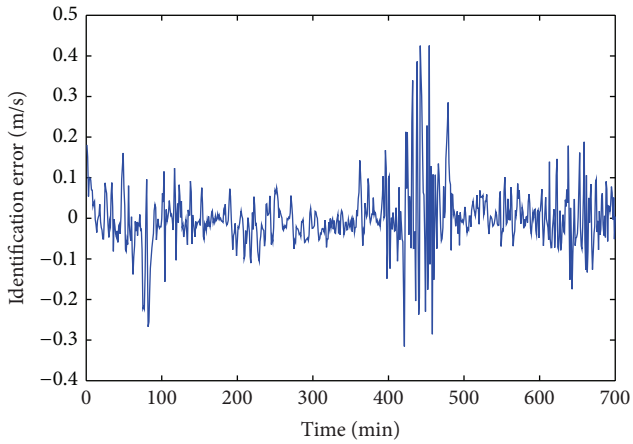

Figure 3: Identification error.

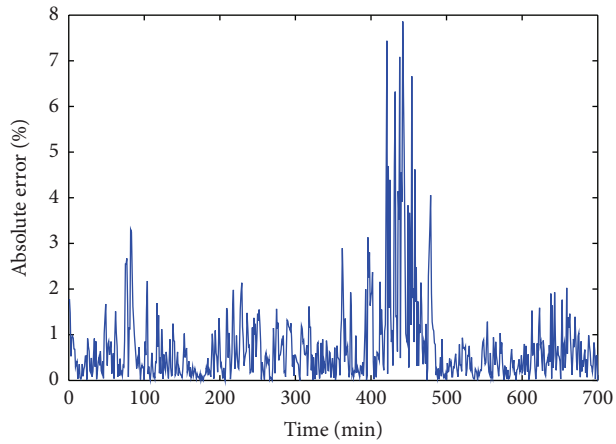

Figure 5: Absolute relative error.

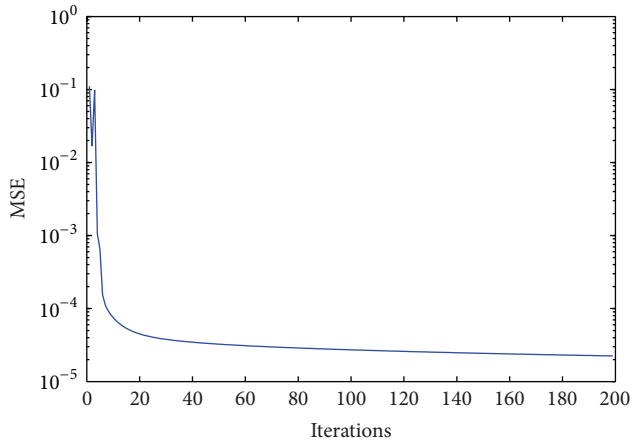

Figure 4: Mean square error.

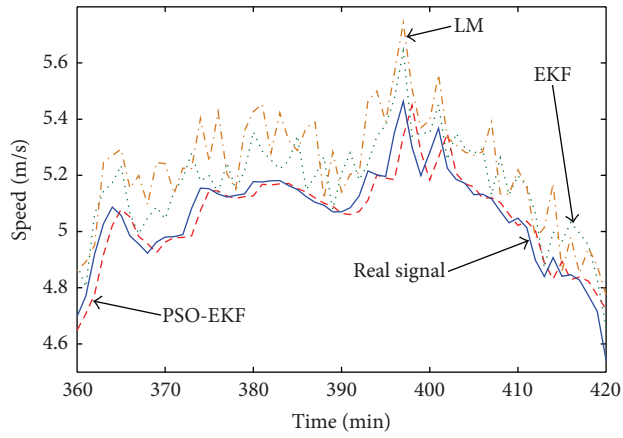

Figure 6: Identification performance comparison for wind speed forecast.

3.2. Forecasting Results for Smart Grid Variables. Due to random variations in weather conditions, power generation from renewable sources is constantly changing. Combining the forecast of wind speed and output power is a good way to improve the performance in scheduling of wind power [3]. Reliability is one of the most important factors in smart grid operation, so constant monitoring and control is necessary to achieve this goal. An accurate forecast can improve the performance of intelligent controllers and management systems in the grid.

This project is implemented in the Mechatronics Building of the UADY Faculty of Engineering using the data obtained from a 10 kW wind turbine as shown in Figure 7. To characterize the energy consumption from the public grid a year of data has been collected. To characterize the wind and solar potential, irradiance and wind speed data are collected from the meteorological station installed in the FI-UADY. The statistical values obtained from a one-year analysis are applied to train the neural predictor.

Figures 8, 10, and 12 display the computation of the minimum embedding dimension for each one of the analyzed time series. We select 6 regressors to be included in the neural

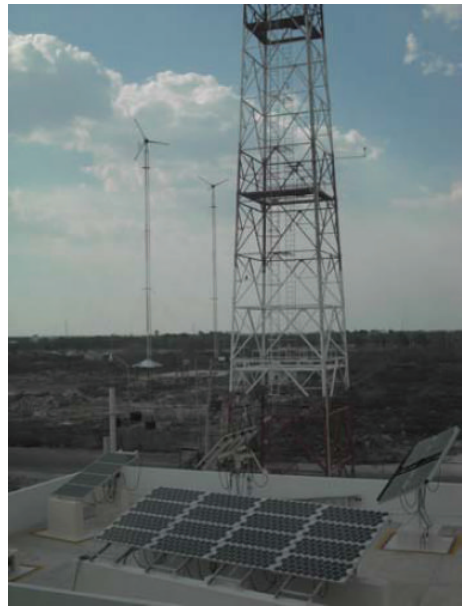

Figure 7: Wind turbines and photovoltaics modules in the FI-UADY.

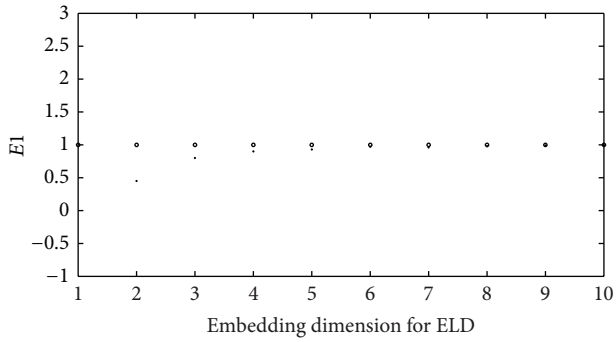

Figure 8: Embedding dimension for the electrical load demand time series.

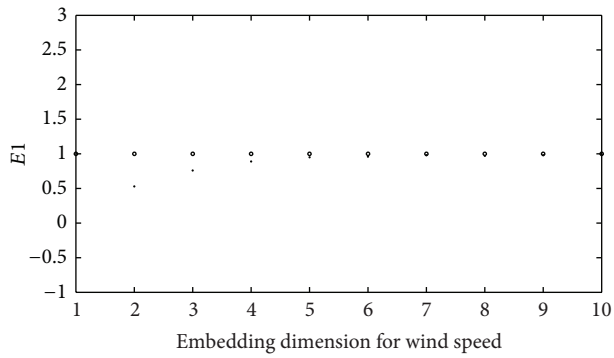

Figure 10: Embedding dimension for wind speed.

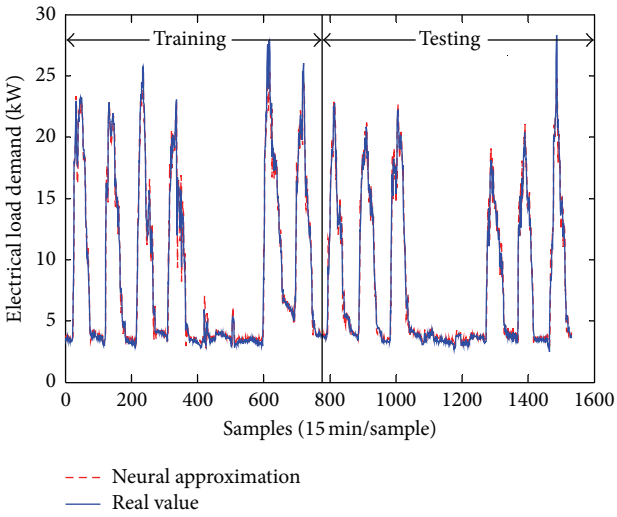

Figure 9: Electric load demand time series forecasting.

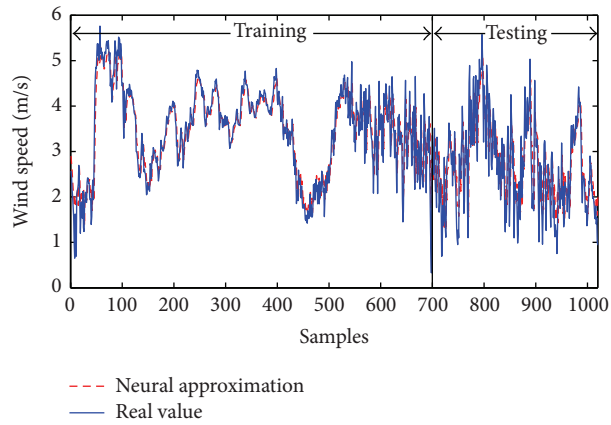

Figure 11: Wind speed time series forecasting.

network input vector for the electrical load demand (ELD), 8 for wind energy generation (WEG), and 7 for the wind speed (WS). To train the HONN for each variable, we kept the following design parameters: 2 external inputs corresponding to hours and days, 25 units in the hidden layer, 1 neuron in the output layer, 300 iterations maximum, initial values for synaptic weights randomly selected in the range, and MSE required to end the training less than 1×10^{-4}. The training is performed offline, using a parallel configuration; for this case the delayed output is taken from the neural network output. The initial values for the covariance matrices are for the electrical load demand (ELD) and for the wind speed (WS). The data for the ELD is collected every 5 minutes and averaged each 15 minutes, in the case of the WS wich is taken every minute and without average; therefore, each variable is plotted as a function of the sample. The data for the wind energy generation (WEG) is collected every 18 minutes.

For the ELD, 765 are used as samples to accomplish the network training. The data size to train the neural network for WS forecasting is 715 samples. In order to verify if the proposed scheme is adequate using less samples, 200 samples are employed for the WEG and good results are obtained using this reduced samples number, as exemplified by the simulation results. The results for the ELD, WS, and WEG time series forecasting are shown. As can be seen from Figures 9, 11, and 13, the forecasting is successfully done with a good prediction horizon.

4. Conclusions

This paper proposes the use of an RMLP trained with a PSO-EKF learning algorithm, to predict minutely wind speed with good results as shown in Table 1. The proposed method has a compact structure but taking into account the dynamic nature of the system where behavior is required to be predicted. The proposed neural identifier proved in our experiments to be a model that captures very well the complexity associated with important variables in smart grids operation. Future work on implementing higher order neural networks aims for the design of optimal operation algorithms for smart grids composed of wind and photovoltaic generation systems interconnected to the utility grid. This management system can use the forecasting data to operate the global system, fulfilling the load demand, minimizing the power supplied by the utility grid, and maximizing the one supplied by renewable sources.

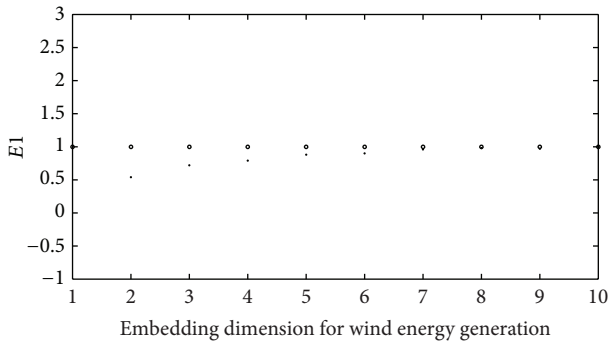

Figure 12: Embedding dimension for wind energy generation.

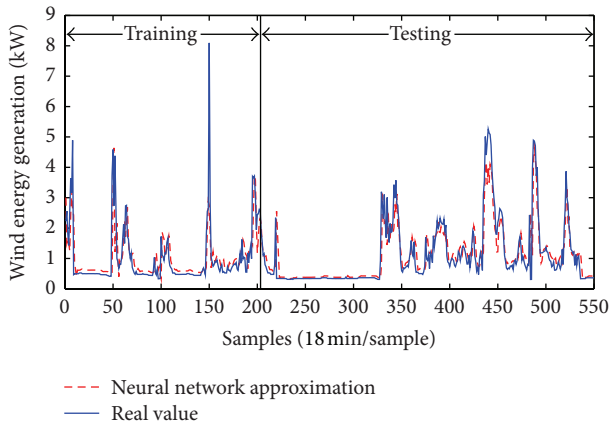

Figure 13: Wind energy generation forecasting.

Acknowledgments

The authors thank the support of CONACYT Mexico, through Project 103191Y and FOMIX 170414. They also thank the very useful comments of the anonymous reviewers, which helped to improve the paper.

References

[1] M. Meiqin, D. Ming, S. Jianhui, L. Chang, S. Min, and Z. Guorong, "Testbed for microgrid with multi-energy Generators," in *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE '08)*, pp. 637–640, May 2008.

[2] R. L. Welch, S. M. Ruffing, and G. K. Venayagamoorthy, "Comparison of feedforward and feedback neural network architectures for short term wind speed prediction," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '09)*, pp. 3335–3340, Atlanta, Ga, USA, June 2009.

[3] J. Wu, S. Chen, J. Zeng, and L. Gao, "Control technologies in distributed generation system based on renewable energy," *Asian Power Electronics Journal*, vol. 3, no. 1, pp. 39–52, 2009.

[4] A. Y. Alanis, L. J. Ricalde, and E. N. Sanchez, "High Order Neural Networks for wind speed time series prediction," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '09)*, pp. 76–80, Atlanta, Ga, USA, June 2009.

[5] T. G. Barbounis, J. B. Theocharis, M. C. Alexiadis, and P. S. Dokopoulos, "Long-term wind speed and power forecasting using local recurrent neural network models," *IEEE Transactions on Energy Conversion*, vol. 21, no. 1, pp. 273–284, 2006.

[6] T. H. M. El-Fouly, E. F. El-Saadany, and M. M. A. Salama, "One day ahead prediction of wind speed and direction," *IEEE Transactions on Energy Conversion*, vol. 23, no. 1, pp. 191–201, 2008.

[7] G. N. Kariniotakis, G. S. Stavrakakis, and E. F. Nogaret, "Wind power forecasting using advanced neural networks models," *IEEE Transactions on Energy Conversion*, vol. 11, no. 4, pp. 762–767, 1996.

[8] T. Senjyu, A. Yona, N. Urasaki, and T. Funabashi, "Application of recurrent neural network to long-term-ahead generating power forecasting for wind power generator," in *Proceedings of the IEEE PES Power Systems Conference and Exposition (PSCE '06)*, pp. 1260–1265, November 2006.

[9] Y.-K. Wu and J.-S. Hong, "A literature review of wind forecasting technology in the world," in *Proceedings of the IEEE Lausanne POWERTECH*, pp. 504–509, July 2007.

[10] I.-H. Li, W.-Y. Wang, S.-F. Su, and Y.-S. Lee, "A merged fuzzy neural network and its applications in battery state-of-charge estimation," *IEEE Transactions on Energy Conversion*, vol. 22, no. 3, pp. 697–708, 2007.

[11] L. J. Ricalde, G. A. Catzin, A. Y. Alanis, and E. N. Sanchez, "Higher order wavelet neural networks with Kalman learning for wind speed forecasting," in *Proceedings of the IEEE Symposium on Computational Intelligence Applications in Smart Grid (CIASG '11)*, Symposium Series on Computational Intelligence, pp. 55–60, Paris, France, April 2011.

[12] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270–280, 1989.

[13] C.-S. Leung and L.-W. Chan, "Dual extended Kalman filtering in recurrent neural networks," *Neural Networks*, vol. 16, no. 2, pp. 223–239, 2003.

[14] M. Norgaard, N. K. Poulsen, and O. Ravn, "Advances in derivative-free state estimation for nonlinear systems," Technical Report IMM-REP-1988-15, Technical University of Denmark, 2000.

[15] A. Y. Alanis, E. N. Sanchez, and A. G. Loukianov, "Discrete-time adaptive backstepping nonlinear control via high-order neural networks," *IEEE Transactions on Neural Networks*, vol. 18, no. 4, pp. 1185–1195, 2007.

[16] L. A. Feldkamp, D. V. Prokhorov, and T. M. Feldkamp, "Simple and conditioned adaptive behavior from Kalman filter trained recurrent networks," *Neural Networks*, vol. 16, no. 5-6, pp. 683–689, 2003.

[17] K. E. Parsopoulos and M. N. Vrahatis, *Particle Swarm Optimization*, IGI Global, 2010.

[18] R. Kiran, S. R. Jetti, and G. K. Venayagamoorthy, "Online training of a generalized neuron with particle swarm optimization," in *Proceedings of the International Joint Conference on Neural Networks 2006 (IJCNN '06)*, pp. 5088–5095, Vancouver, Canada, July 2006.

[19] T. Su, J. Jhang, and C. Hou, "A hybrid artificial neural networks and particle swarm optimization for function approximation," *International Journal of Innovative Computing, Information and Control*, vol. 4, no. 9, pp. 2363–2374, 2008.

[20] C.-J. Lin and H.-M. Tsai, "FPGA implementation of a wavelet neural network with particle swarm optimization learning," *Mathematical and Computer Modelling*, vol. 47, no. 9-10, pp. 982–996, 2008.

[21] C.-J. Lin, C.-H. Chen, and C.-T. Lin, "A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications," *IEEE Transactions on Systems, Man and Cybernetics C*, vol. 39, no. 1, pp. 55–68, 2009.

[22] L. Cao, "Practical method for determining the minimum embedding dimension of a scalar time series," *Physica D*, vol. 110, no. 1-2, pp. 43–50, 1997.

[23] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.

[24] A. Lazinica, *Particle Swarm Optimization*, In-Tech, 2009.

[25] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, May 1998.

[26] M. Clerc, "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation*, pp. 1951–1957, 1999.

[27] B. Al-kazemi and C. K. Mohan, "Multi-phase generalization of the particle swarm optimization algorithm," in *Proceedings of the Congress on Evolutionary Computation*, vol. 1, pp. 489–494, 2002.

[28] M. B. Kennel, R. Brown, and H. D. I. Abarbanel, "Determining embedding dimension for phase-space reconstruction using a geometrical construction," *Physical Review A*, vol. 45, no. 6, pp. 3403–3411, 1992.

[29] A. Y. Alanis, C. Simetti, L. J. Ricalde, and F. Odone, "A wind speed neural model with particle swarm optimization kalman learning," in *Proceedings of the 9th International Symposium on Intelligent Automation and Control*, Puerto Vallarta, Mexico, June 2012.

[30] National Renewable Energy Laboratory's National Wind Technology Center, http://www.nrel.gov/wind/.