INCREMENTAL AND STABLE TRAINING ALGORITHM FOR WIND TURBINE NEURAL MODELING

S. Abid[1*] – M. Chtourou[1] – M. Djemel[1]

[1]Control & Energy Management Lab (CEM LAB) National School of Engineering of Sfax, University of Sfax, B.P. 1173, 3038 Sfax, Tunisia.

ARTICLE INFO

***Article history*:**
Received: 26.11.2012.
Received in revised form: 25.02.2013.
Accepted: 25.02.2013.

Keywords:
Wind turbine
Neural models
Incremental algorithm
Adaptive learning rate

Abstract:

Training and topology design of artificial neural networks are important issues with large application. This paper deals with an improved algorithm for feed forward neural networks (FNN)$_s$ training. The association of an incremental approach and the Lyapunov stability theory accomplishes both good generalization and stable training process. The algorithm is tested on wind turbine modeling. Compared to the incremental approach and to the Lyapunov stability based method, the association of both strategies gives interesting results.

1 Introduction

Wind energy is currently experiencing an unrecorded growth as the cost price of this energy form has become competitive and considerable technological progress has been achieved in the field of wind turbine. More intelligence is being introduced in modeling and control of these systems [1], [2], [3] and [4]. The objective is to optimize the power efficiency of these systems and to improve quality during operating conditions.

Variable-speed wind turbines exhibit a number of significant advantages with respect to fixed-speed turbines. Fixed-speed operation means that the maximum performance coefficient has been reached only for a specific wind speed, while the performance has been significantly degraded for all other wind speed regimes. Variable-speed wind turbines have the ability to adapt operation conditions to different wind speed regimes, thus improving overall performance. This provides higher energy yields with fewer grid connection power peaks. Nonetheless, these benefits generally come at the cost of more sophisticated control systems and power electronics on the generator part.

Modeling and the simulation of wind turbines aim at analyzing and optimizing the power extraction rate [5], [8] and [9]. These tasks are complex since they include descriptions of aerodynamic interaction, elastic mechanical coupling, electrical and pith actuator subsystems.

The aerodynamic forces acting on wind turbines are turbulent in nature. Moreover, wind speed is known to vary stochastically [7]. As a result, it is impossible to predict the captured aerodynamic torque from single point wind speed measurements. Therefore, one is lead to elaborate more adequate control procedures such as neural networks controllers for wind turbines which enable to deal with the presence of uncertainties [6].This step can be accomplished as well as the model describes the system dynamics correctly, which shows the importance of the choice of modeling strategy.

In this work, modeling of the turbine is provided via a neural model whose architecture is selected based on an approach arising from the association of two strategies for synthesis of neural networks. This approach leads to a new incremental learning algorithm based on Lyapunov stability theory.

The rest of this paper is organized as follows. In section 2 the modeling of an aerodynamic action on wind turbines is described. In section 3 the neural method for the speed modeling of wind turbines is represented. Section 4 illustrates the obtained

* Corresponding author. Tel.: +216 21 472 391
E-mail address: abid_slim_enis@yahoo.fr

simulation results. Finally, the conclusion is presented in section 5.

2 Modeling aerodynamic action on wind turbines

The rigid model with only one degree of freedom [26-28] is described by the following equation:

$$J_t\dot{\omega}_a = T_a - T_g - k_t\dot{\omega}_a. \quad (1)$$

For this model $\omega_g=n_g\omega_a$ is satisfied, where: $\omega_a = \dot{\theta}_a$ is the rotor rotational speed and $\omega_g = \dot{\theta}_g$ is the rotational speed of the high speed shaft, while n_g designates the gear ratio between the primary shaft and the secondary shaft, θ_a and θ_g are the azimuthally rotor position and the azimuthally position of the high speed shaft.

The captured aerodynamic torque T_a is given [10] in terms of the power coefficient $C_p(\lambda,\beta)$ as:

$$T_a(k) = \frac{1}{2}\phi\pi R^2 \frac{v^3}{\omega_a(k)} C_p(\lambda,\beta), \quad (2)$$

where λ is the specific speed defined as:

$$\lambda = \frac{R\omega_a}{v}, \quad (3)$$

v is the effective wind speed, ϕ is the air density, and R designates the blades rotor radius.

The power coefficient $C_p(\lambda,\beta)$ is estimated using aerodynamic data obtained from wind tunnel measurements. It is generally represented under the form of an analytical formula which gives $C_p(\lambda)$ for various values of the pitch angle β.

In the literature [11] one finds the following approximation:

$$C_p(\lambda,\beta) = A\exp[B], \quad (4)$$

with:

$$A = c_1G + c_4B + c_5,\ B = c_6G \text{ and } G = \frac{1}{\lambda + c_2\beta} + \frac{c_3}{\beta^3 + 2}$$

coefficients c_i, i=1... 6 are identified from real C_p curves.

The point (.) designates the first order time derivative, T_g is the generator torque and J_r, J_g, k_r, k_g are the moment of inertia of rotor side masses, the moment of inertia of generator side masses, the mechanical damping in the rotor side and the mechanical damping in the generator side respectively, where:

$J_t = J_r + n_g^2 J_g$ is the total inertia of generator side masses,

$k_t = k_r + n_g^2 k_g$ is the equivalent mechanical damping.

Using the Euler approximation, the discrete model describing the wind turbine can be expressed as follows:

$$\omega_a(k+1) = \omega_a(k) + \Delta t\left[\frac{T_a(k)}{J_t} - \frac{T_g(k)}{J_t} - \frac{k_t}{J_t}\omega_a(k)\right] \quad (5)$$

Δt denotes the sampling period.

3 Neural modeling and synthesis

3.1 Introduction

Neural networks are extraordinary computing and information processing methods can be used to handle the complicated tasks such as pattern recognition, function approximation, time series forecasting and identification of complex systems [12] and [13].

There are various types and architectures of neural networks depending fundamentally on the way they learn. In this work, the multi-layer perceptron approach is used.

Many researchers have studied the problem of learning neural networks and several algorithms have been developed. Faster convergence and function approximation accuracy are two key issues in selecting a training algorithm.

The popular method for training multilayered (FNNs) is the back propagation (BP) algorithm [14] and [15]. The use of this algorithm is not always successful due to its sensitivity to learning parameters, initial state and perturbation [16]. There has been much work on the convergence of (BP) algorithm by using the gradient method [17] and [18]. Also, different versions of (BP) learning algorithms have been proposed, such as on-line algorithm for dealing with time varying inputs [19] and the Levenberg-Marquardt-algorithm [20].

Modeling using neural networks requires the phase of model selection, which is a crucial stage in the design of a neural network. This phase must lead to choose a model that is complex enough to be adjusted with the data but not too excessive.

In this paper, we will develop an improved constructive training algorithm for feedforward neural network using Lyapunov stability theory. It employs an incremental training procedure where training patterns are learned one by one. The

Lyapunov stability theory has been introduced to adjust the learning rate, assuring the stability of training process.

3.2 Training algorithm based on Lyapunov stability theory

Learning based on (BP) algorithm can lead to unsatisfactory results. In addition, this algorithm has unavoidable disadvantages such as its slow convergence and its inability to establish a global convergence. To overcome this problem, the Lyapunov stability theory has been used to provide an adaptive learning rate for improving the convergence speed.

A simple $(FNN)_s$ with a single output is represented in Fig. 1.

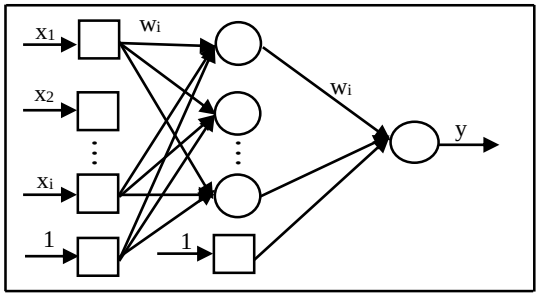

Figure 1. Feedforward Neural Network.

This neural network is parameterized in terms of its weights, where:

$$w = \left[w_1, w_2, ..., w_m\right]^T \in \Re^m \tag{6}$$

The training data consists of N patterns $\{x^i, y^i\}$, $i=1,2,...,N$.

In order to derive a weights update law, a Lyapunov function candidate has been defined as:

$$V = \frac{1}{2}\left(r^T r\right), \tag{7}$$

where r denotes the difference between the real output and the desired output, as:

$$r = \left[y_d^1 - y^1, ..., y_d^i - y^i, ..., y_d^N - y^N\right]^T \tag{8}$$

The stability conditions $\left(\dot{V} \le 0\right)$ give the weights update law with an adaptive learning rate which can be expressed as:

$$w(k+1) = w(k) + \zeta \frac{r_i^2}{\beta + \left\| J_i^T r_i \right\|^2} J_i^T r_i \,, \tag{9}$$

where:

- r_i designates the error signal for sample i, as :

$$r_i = \left(y_d^i - y^i\right) \in \Re \tag{10}$$

- J_i is the instantaneous value of the Jacobian, as:

$$J_i = \frac{\partial y^i}{\partial w} \in \Re^{1 \times m} \tag{11}$$

- ζ, β are a constant and a very small constant to avoid numerical instability when error signal goes to zero respectively, which are selected heuristically.

More details about this algorithm noted LF1 can be found in Laxmidhar et al. [25].

In the following section, we present some improvements on the above algorithm to deal with an incremental structure of the $(FNN)_s$.

3.3 Improved incremental algorithm based on Lyapunov stability theory

Liu et al. [21] elaborated a constructive training algorithm for determining the network size. In his approach, the training begins with a single training pattern and a single hidden layer neuron. The aim is to get such a neural network topology that the overall error of training is less than a specified error tolerance.

Although the constructive learning strategy can lead to a neural network with minimal structure, the neural model is risking being over trained. To solve this problem, a modified version of this algorithm that helps in avoiding poor generalization performances based on regularization technique (early-stopping) has been proposed in Abid et al. [22]. Early-stopping consists on stopping the training when a moderate value of training error is reached. Indeed, in the first step, learning and generalization criteria begins to decrease. In a following step, the learning criterions continue to decrease nevertheless the generalization one starts to increase. In this moment, the training should be stopped [23] and [24].

It is to be noted that we are interested in a multi input-single output (MISO) model and the weights update is based on the equation (9).

The proposed incremental training algorithm can be described as follows:

Step 1: choose one pattern from the training base (*L=1*). Train the neural network with one hidden node using the chosen pattern and calculate the *EQMA(1)*, where: *EQMA* represents the average quadratic error of training, which is defined as:

$$EQMA = \sqrt{\frac{1}{N_A}\sum_{i=1}^{N_A} r_i^2}\,. \tag{12}$$

Here: N_A and r_i indicates the number of samples in the training set and the difference between the real output of sample *i* and output estimated by the neural model, respectively.

Step 2: if ($L<N_A$), choose the next pattern ($L=L+1$) and go to *step 3* for training; else ($L= N_A$), end of the algorithm.

Step 3: train the neural network with N_c hidden nodes using *L* patterns from the training set and calculate the values of *EQMA(L)* and *EQMV(N_c)*, where *EQMV* designates the average quadratic error of validation,

$$EQMV = \sqrt{\frac{1}{N_v}\sum_{i=1}^{N_v} r_i^2}. \quad (13)$$

Here: N_v indicates the number of samples in the validation set.
If *(EQMA(L)<$EQMA_{tol}$)*, go back to *step 2*; otherwise, go to *step 4* for growing where $EQMA_{tol}$ represents a tolerated value of the average quadratic error of training.

Step 4: if ($N_c=1$), then, ($N_c= N_c+1$) and go back to *step 3*; else ($N_c>1$), two tests should be done to decide about the evolution of the network structure.
In the case of the growth of the generalization criterion (*EQMV*) with a value greater than a tolerated threshold ($EQMV_{tol}$), the algorithm should go to *step 5*. The same step will be executed when the generalization criterion decreases. These cases are summarized as follows:

$$\text{if} \left\{ \begin{array}{l} \left(\begin{array}{l} (EQMV(N_c) > EQMV(N_c - 1)) \\ and \quad (EQMV(N_c) > EQMV_{tol}) \end{array} \right) \\ or\,(EQMV(N_c) < EQMV(N_c - 1)) \end{array} \right\}, \text{ then go to } step5.$$

These tests are particularly satisfied in the beginning of the learning step when the generalization criterion can have an oscillatory behaviour.
In the third case and when the generalization criterion grows with a value lower than $EQMV_{tol}$, then increase slightly the $EQMA_{tol}$ and re-execute the *step 3*. This case is summarized by:

$$\text{if} \left\{ \begin{array}{l} (EQMV(N_c) > EQMV(N_c - 1)) \\ and \\ (EQMV(N_c) < EQMV_{tol}) \end{array} \right\}, \quad \text{then}$$

$(EQMA_{tol} = \alpha\, EQMA_{tol})$ where α is a constant slightly higher than 1, and go back to *step 3*.
In this case, the network structure has a sufficient hidden nodes and neural network accomplishes good learning performance with generalization error tending to increase. In this case, the ($EQMA_{tol}$) is increased so as to slow down the recruitment of hidden nodes.

Step 5: keep the weights of the last successfully trained neural network, increase the number of hidden neurons by one and assign its initial weights. Go to *step 3*.
These steps can be summarized by the flowchart presented in Fig. 2.

4 Experiments and discussions

In this section, we present the simulation results. The capacities of the proposed algorithm are analyzed. We use this algorithm for the neural identification of a wind turbine.
The goal of our simulation is to determine the adequate structure of the input-output neural model which describes the dynamics of the wind turbine by using the approach presented in section 3.
The wind turbine parameters used in simulations are the following [29]:

$$\left\{ \begin{array}{l} \phi = 1.225 kgm^{-3}, R = 21.38m, n_g = 43.165, \\ J_r = 3.25.10^5 kgm^2, J_g = 34.4 kgm^2, \beta = -1°, \\ k_r = 1.5 Nmrad^{-1}s, k_g = 3.7 Nmrad^{-1}s, \lambda_{opt} = 7.5, \\ c_1 = 1.1023.10^2, c_2 = -0.02, c_3 = 0.003.10^2, \\ c_4 = -0.309082, c_5 = -9.636.10^2 \text{ et } c_6 = -18.4 \end{array} \right.$$

The input-output neural model describing the wind turbine is represented in Fig. 3.
The model input vector is constituted by the actual and previous torque generator ($T_g(k)$ and $T_g(k-1)$), the actual and previous rotor rotational speed ($\omega_a(k)$ and $\omega_a(k-1)$) and the actual value of wind speed $v(k)$. The model output is the future value of the rotor rotational speed $\omega_a(k+1)$.
Based on several simulations, the input vector and the value of sampling period ($\Delta t=0.1s$) are selected to obtain the best performance.
The chosen mean wind speed was set at $v_{moy}=12ms^{-1}$.
Fig. 4 and 5 represent the input signals $T_g(k)$ and $v(k)$ respectively, used in the training and validation phases.
The simulation results describing the performances of algorithms presented in this paper are illustrated in Table 1.

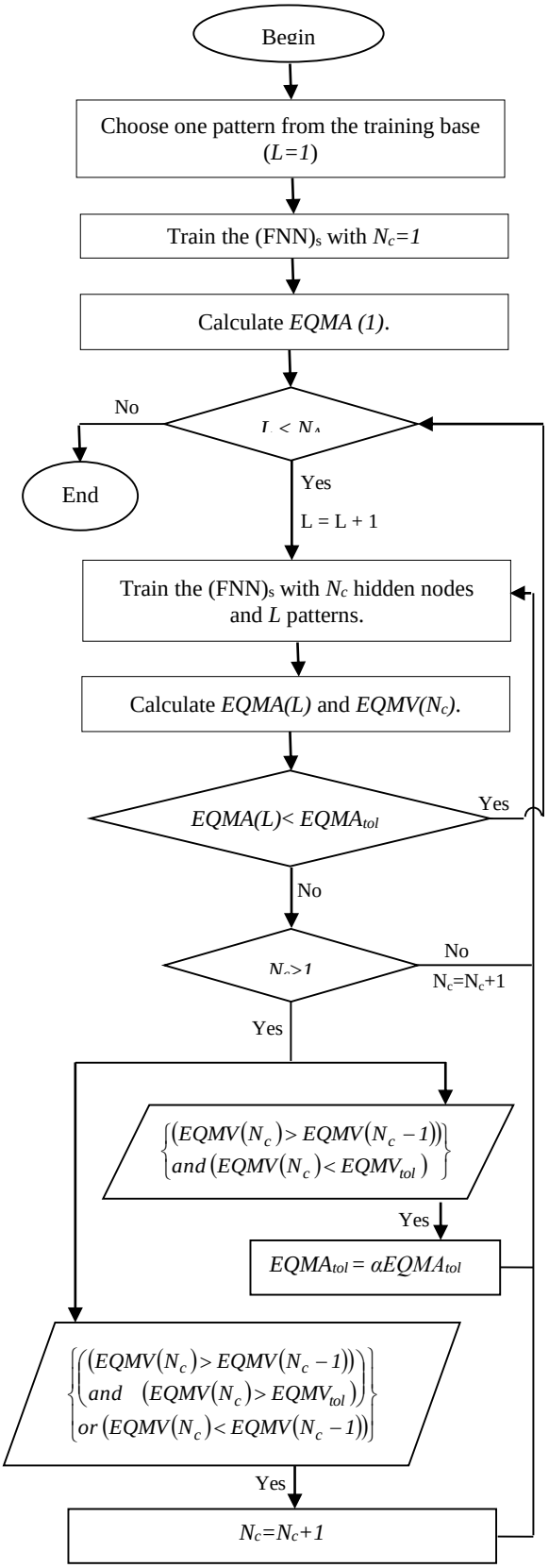

Figure 2. Improved incremental algorithm based on Lyapunov stability theory.

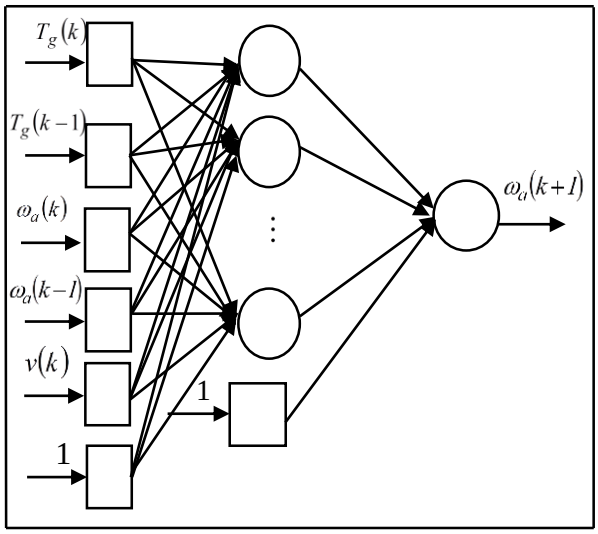

Figure 3. Input-output neural model.

It can be seen that the proposed algorithm provides better convergence properties in training and validation phases when compared to LF1 and incremental algorithms.

Table 1 shows the contribution from the mixture of the constructive approach and the algorithm LF1. In fact we note that the incremental algorithm leads to satisfactory performances but with a slow convergence time. Moreover, the algorithm LF1 presents a minimal convergence time with performances degradation of the obtained model. The proposed algorithm guarantees both fast convergence and better learning and generalization abilities.

The simulation results, which relate to the selection of hidden neurons number using the incremental training algorithm and Lyapunov stability theory, are presented in Fig. 6.

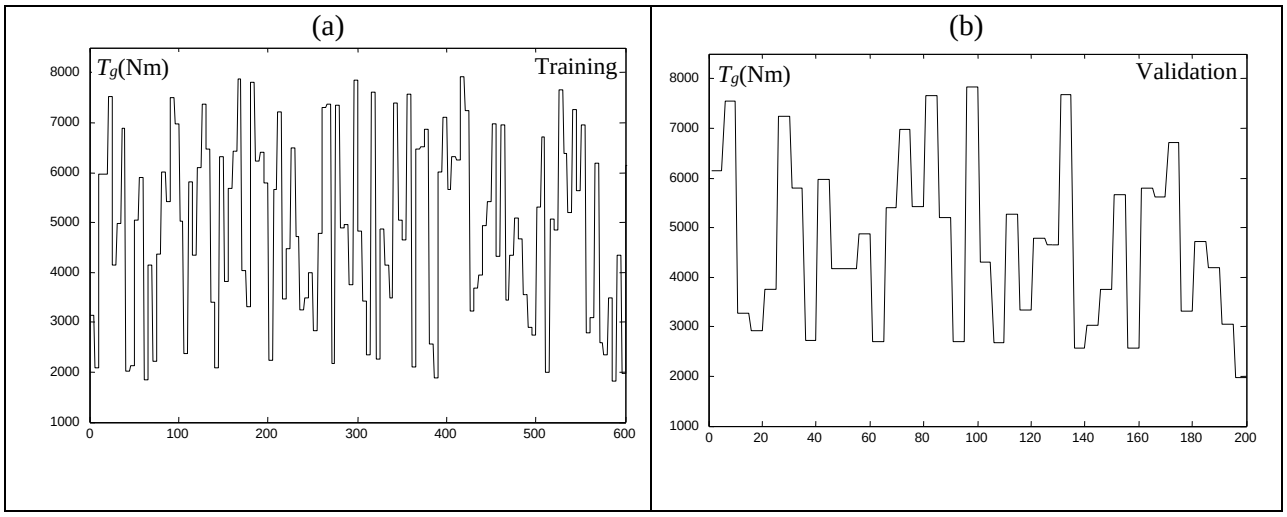

Figure 4.Tg (k) used in the training and validation phases ((a): training, (b): validation).

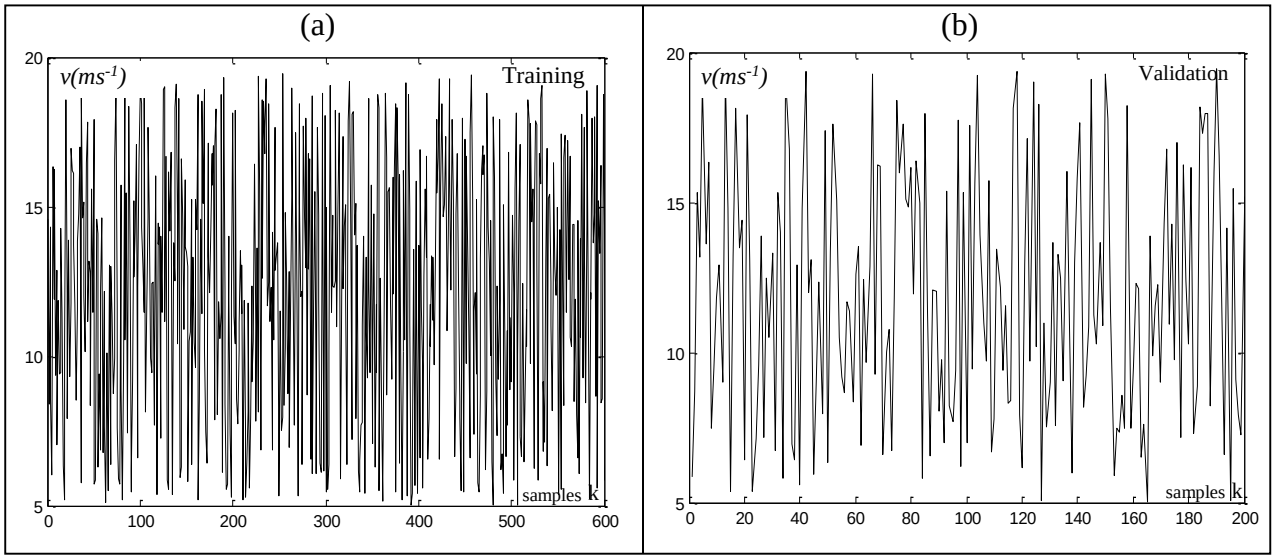

Figure 5. v(k) used in the training and validation phases ((a): training, (b): validation).

Table 1. Performances of algorithms

Algorithm	Numerical simulation parameters	*EQMA*	*EQMV*	N_c	Run time
LF1(fixed structure)	ζ=0.6, β=0.001	0,0047	0,004	5	9'
Incremental algorithm	α=1,01, $EQMA_{tol}$=0,035, $EQMV_{tol}$=0,045	0,0032	0,0027	5	17'43''
Incremental algorithm combined with LF1	ζ=0,6, β=0,001, α=1,01, $EQMA_{tol}$=0,035, $EQMV_{tol}$=0,045	0,0032	0,0023	5	13'

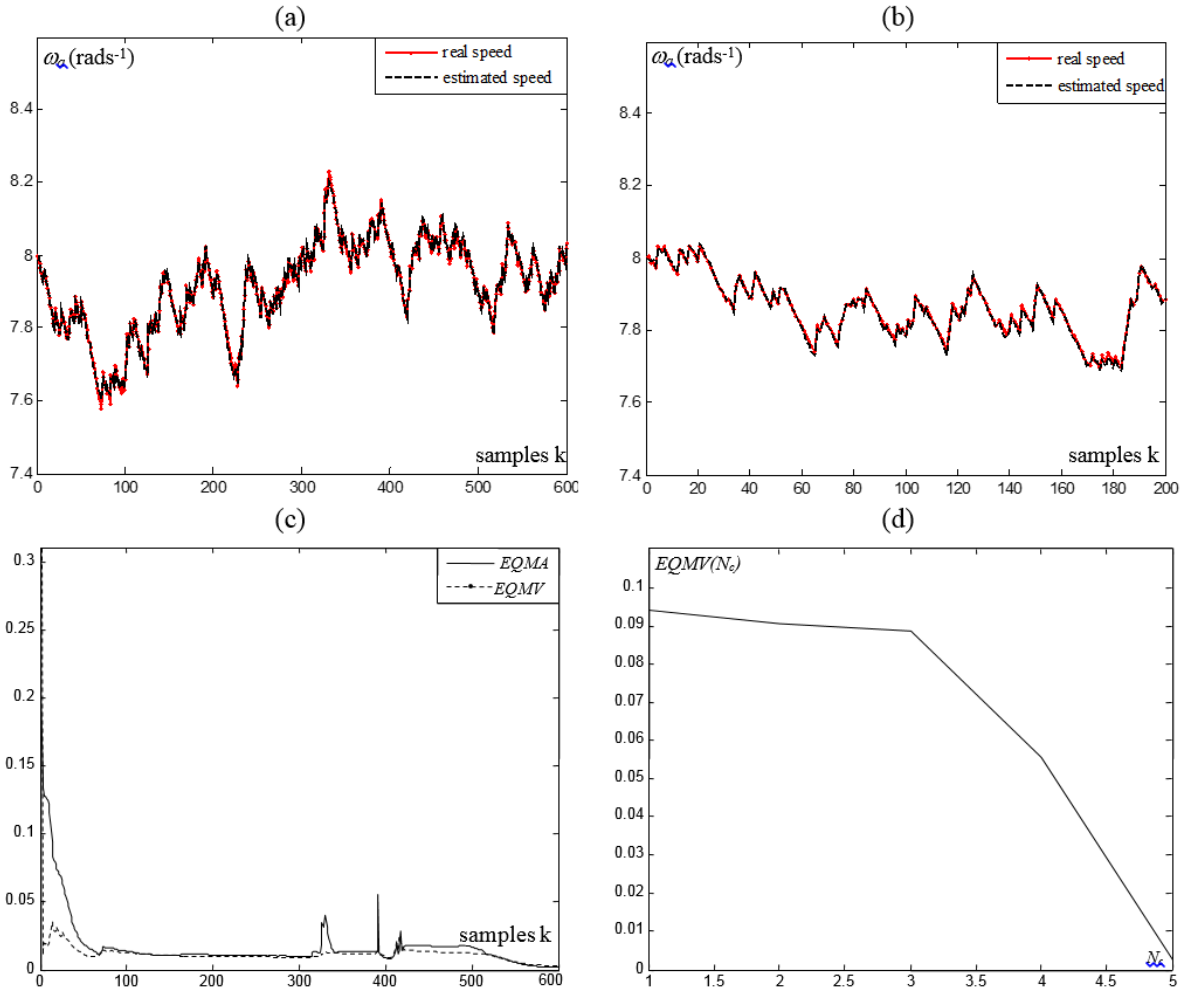

Figure 6. Training and validation performances for the proposed algorithm ((a): training, (b): validation, (c): EQMA and EQMV, (d): EQMV(N_c)).

5 Conclusion

In this paper, an improved approach for neural models selection is proposed. The main contribution of this method is to show the usefulness of the association of the constructive strategy and the Lyapunov stability theory for the synthesis of neural networks. To confirm the effectiveness of the developed algorithm, we have used to the neural modeling of the speed of wind turbine. The simulation results have demonstrated that the proposed algorithm not only enhance the training and generalization abilities but also shortens the runtime remarkably improving the practicability of this algorithm in both theoretical and real problems.

References

[1] Burton T., Sharpe D., Jenkins N. and Bossanyi E.: *Wind Energy Handbook*, Wiley, Chichester, UK, (2001).

[2] D. Bianchi F., De Battista H. and Mantz R.J.: *Wind Turbine Control Systems: Principles, Modeling and Gain Scheduling Design*, Springer-Verlag London Limited, (2007).

[3] Wright A.: *Modern Control Design for Flexible Wind Turbines*, Ph.D. thesis. Boulder, CO: University of Colorado, USA, (2003).

[4] Boukhezzar A.: *Nonlinear control of variable-speed wind turbines for generator torque limiting and power optimization*, Journal Solar Energy Engineering, Trans. ASME, 128 (2006) 4, 516 – 530.

[5] Fingersh L., Hand M. and Laxson A.: *Wind turbine design cost and scaling model*, National Renewable Energy Laboratory, 45 (2006) 5, 35 – 36.

[6] Novak P., Ekelund T., Jovilk I. and Schmidtbauer B.: *Modelling and control of variable speed wind turbine drive systems dynamics*, IEEE Control Systems Magazine, 15 (1995) 4, 28 – 38.

[7] Larsen T.J., Madsen H. and Thomsen K.: *Aero elastic effects of large blade deflections for wind turbines*, Delft University of Technology. The Science of making Torque from Wind, Roskilde, Denmark, (2004), 238 – 246.

[8] Sargolzaei J., Kianifar A.: *Modeling and simulation of wind turbine Savonius rotors using artificial neural networks for estimation of the power ratio and torque*, Simulation Modelling Practice and Theory, Elsevier, 17 (2009), 1290–1298.

[9] Ekonomou L., Lazarou S., Chatzarakis G.E., Vita V.: *Estimation of wind turbines optimal number and produced power in a wind farm using an artificial neural network model*, Simulation Modelling Practice and Theory, Elsevier, 21 (2012), 21–25.

[10] Camblong H. : *Minimisation de l'impact des perturbations d'origine éolienne dans la génération d'électricité par des aéroturbines á vitesse variable*, Thése de l'Ecole Nationale Supérieure d'Arts et Métiers, Centre de Bordeaux, (2003).

[11] Reif K., Sonnemann F., and Unbehauen R.: *Nonlinear State Observation Using H_∞ Filtering Riccati Design*, IEEE Transactions On Automatic Control, 44 (1999) 1.

[12] Mehra P. and Wah B. W.: *Artificial Neural Networks: Concepts and Theory*, IEEE Comput. Society Press, (1992).

[13] Rivals I., and Personnaz L. : *Réseaux de neurones formels pour la modélisation, la commande et la classification*, Collection sciences et techniques de l'ingénieur dirigée par Suzanne LAVAL, (2003).

[14] Lippmann R. P.: *An introduction to computing with neural networks*, IEEE Acoust. Speech, Signal Process. Mag., 4 (1987) 2, 4-22.

[15] Narendra K. S. and Parthasarathly K.: *Gradient methods for optimisation of dynamical systems containing neural networks*, IEEE Trans. Neural Netw., 2 (1991) 2, 252-262.

[16] Yu X., Efe M. O., and Kaynak O.: *A general backpropagation algorithm for feedforward neural networks learning* , IEEE Trans. Neural Netw., 13 (2002), 251-254.

[17] Gori M., and Maggini M.: *Optimal convergence of on-line backpropagation*, IEEE Trans. Neural Netw., (1994), 251-254.

[18] Finnoff W., *Diffusion approximations for the constant learning rate backpropagation algorithm and resistance to local minima*, Neural Computat., 6 (1994), 285-295.

[19] Zhao Y.: *On-line neural network learning algorithm with exponential convergence rate*, Electrn. Lett., 32 (1996) 15, 1381-1382.

[20] Hagan M. T. and Menhaj M. B.: *Training feedforward neural networks with the Marquardt algorithm* , IEEE Trans. Neural Netw., 5 (1994), 989-993.

[21] Liu D., Chang T. S., and Zhang Y.: *A Constructive Algorithm for Feedforward Neural Networks with Incremental Training*, IEEE Transactions on Circuits and Systems. Fundamental Theory and Applications, 49 (2002) 12, 1876-1879.

[22] Slim A., Mohamed C., Mohamed C. : *Méthodes Statistique et Incrémentale pour la Sélection de Modèles Neuronaux,* 8ème conférence internationale: Sciences et Techniques de l'Automatique, STA'07, ASC, 2007.

[23] Hagiwara K. and Kuno K.: *Regularization learning and early stopping in linear networks*, IEEE, (2000).

[24] Chan Z.S.H., Ngan H.W. and Rad A.B.: *Improving Bayesian Regularization of ANN via Pre-training with Early-stopping*, Neural Processing Letters, 18 (2003), 29-34.

[25] Laxmidhar B., Swagat K., and Awhan P.: *On Adaptive Learning Rate That Guarantees Convergence in Feedforward Networks*, IEEE Transactions On Neural Networks, 17 (2006) 5, 1116-1125.

[26] Christou P.: *Advanced materials for turbine blade manufacture*, Journal of Solar Energy Engineering, (2007)

[27] Fingersh L., Hand M. and Laxson A.: *Wind turbine design cost and scaling model*, National Renewable Energy Laboratory, 45 (2006), 5, 35-36.

[28] Song, Y. D., Dhinakaran B. and Bao X. Y.: *Variable speed control of wind turbines using nonlinear and adaptive algorithms*, Wind Engineering and Industrial Aerodynamics, 85 (2000), 293-308.

[29] Khamlichi A., Ayyat B., Bezzazi M., El Bakkali L., Vivas V.C., Castano C.L.F.: *Modelling and control of flexible wind turbines without wind speed measurements*, Australian Journal of Basic and Applied Sciences, 3 (2009), 3246-3.