http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 11 (7): 775-781, 2012
ISSN 1812-5638 / DOI: 10.3923/itj.2012.775.781

Effect of Leaky Rate on the Stability of Autonomously Echo State Network

[1,2]Qingsong Song
[1]Shaanxi Automobile Group Co., Ltd., Xi'an 710200, China
[2]School of Information Engineering, Chang'an University, Xi'an 710064, China

Abstract: Echo State Neural Network (ESN) has becoming much attractive in Artificial Neural Network (ANN) community since it can be easily constructed and adapted. Its superiority over traditional ANN has been demonstrated in many applications. The Wiener-Hopf solution is usually exploited to account for the adaptations. However, the solution can hardly ensure the Lyapunov stability of the trained Leaky-integrator ESNs (LiESNs), when they run in a closed-loop autonomously generative mode. LiESN is another type of ESN, which consists of leaky-integrator neurons. In this study, a sufficient condition of the Lyapunov stability for the autonomously running LiESNs is proposed and proved at first. And then, the output connection weight learning problem is translated into an optimization problem with a nonlinear restriction. Particle swarm optimization algorithm is explored to solve the optimization problem. The simulation experiment results show that the output weight adaptation algorithm, we proposed (we call it PSOESN) can effectively ensure the output precision as well as the Lyapunov stability of the trained LiESNs. It is concluded that the PSOESN is a more effective solution to the output connection weight adaptation problem of such autonomously running ESNs.

Key words: Leaky-integrator echo state network, Lyapunov stability, particle swarm optimization

INTRODUCTION

Artificial Neural Networks (ANNs) are excellent biologically inspired computational devices for a range of real world problems. It has been applied to dynamical system identification related problems, such as temperature prediction (Sharma and Agarwal, 2012), glutathione fermentation process modeling (Zuo-Ping *et al*., 2008), active noise control (Manikandan and Madheswaran, 2007). It is also exploited for intelligent control (Xinhan *et al*., 2002), rule extraction (Yedjour *et al*., 2011), decision-supporting systems (Fidele *et al*., 2009), besides being demonstrated in GPS satellite selection (Mosavi, 2011), seam pucker evaluation in apparels (Hati and Das, 2011), face recognition and object tracking (Wakaf and Saii, 2009; Bouzenada *et al*., 2007) etc. There are limitations for such ANNs computational paradigms, however. For example, error back-propagation based training methods (BP for feed-forward neural networks, Back-propagation through time and real-time recurrent learning for recurrent neural networks) are usually time-consuming, or lose in local minimums. Such limitations impede ANNs to extend themselves for more comprehensive applications.

Recently Echo State Network (ESN) is proposed as a novel paradigm for using recurrent neural networks with a simpler training method. Since being proposed in (Jaeger and Haas, 2004), ESNs have been successfully applied in many cases, such as in nonlinear time series prediction (Song and Feng, 2010), in speech recognition (Verstraeten *et al*., 2005), in mobile robot modeling and control (Salmen and Ploger, 2005). Leaky-integrator ESN (LiESN) is one of the typical ESN representatives which consists of leaky integrator neurons (Jaeger *et al*., 2007). Usually, an LiESN consists of two main component parts: a large-scale recurrent neural network (RNN) (it is called "reservoir" (Jaeger and Haas, 2004) and a linear readout. The most distinctive characteristic of the LiESN is that, only the weights for the connections from the reservoir neurons to the readout neurons need to be adapted for certain task. In addition, because the reservoir has rich enough dynamics, the adaptation can be treated as a simple linear regression problem. The troubling issues on RNN weight adaptation are almost entirely evaded, such as local minima, slow convergence and even non-convergence (Jang *et al*., 1997).

However, the LiESNs may lose their Lyapunov stability as they run in a closed-loop generative mode. In such mode, the outputs are fed back into the reservoir, a closed-loop forms (Fig. 2). It is known that the stability is prerequisite to successful neural network implementations. There are lots of literature concerning of the stability of Hopfield networks, cellular neural networks and bi-directional associative memory networks, etc. Constructing some Lyapunov functional is one of the most common method (Lou and Cui, 2006; Xu *et al*., 2007).

Corresponding Author: Qingsong Song, School of Information Engineering, Chang'an University, Xi'an 710064, China
Tel: +86-29-82334952 Fax: +86-29-62630027

There are few literature concerning the stability of the LiESNs, however. Two representative methods are the ridge regression and the noise immunization (Jaeger *et al*., 2007; Wyffels *et al*., 2008). But, these two methods have obviously insufficiencies: it is not an easy thing to correctly set the regularization factor for the ridge regression or the noise amount for the noise immunization for specific tasks.

In order to ensure the Lyapunov stability of the LiESNs running in a closed-loop generative mode, we proposes another output weight adaptation algorithm, which bases on Particle Swarm Optimization (PSO) algorithm (Clerc, 1999; Clerc and Kennedy, 2002; Eberhart and Shi, 2000). We call it PSOESN. PSOESN considers the output accuracy and the stability restraint during the adaptation. The simulation experiment results show that our algorithm can not only result in high-precision prediction outputs but also ensure the Lyapunov stability of the LiESNs.

LYAPUNOV STABILITY OF THE AUTONOMOUS LiESN

Figure 1 plots the system architecture of the LiESN. The LiESN consists of three parts: the front-part is input, the middle reservoir and the back-end readout. And the LiESN is assumed having L input neurons, M output neurons and N reservoir neurons. Real-valued connection weights are collected in matrix W^{in} for the input weights, in matrix W for the reservoir connections, in matrix W^{out} for the output weight matrix and in matrix W^{back} for the output feedback matrix.

The task is to train a LiESN as a trajectory-generator or a pattern-generator, since the dynamics of the leaky integrator neuron can be slowed down or sped up (Jaeger *et al*., 2007). The training algorithm which is usually used in the literature consists of four Steps (Jaeger, 2002).

In the first step, an untrained LiESN (w^{in}, w, w^{back}) is constructed which is consistent with the assignment to the parameters (the leaking rate (α) and the gain (γ) of the leaky integrator neurons, the spectral radius ($\rho(w)$) and the connectivity density (D) of the reservoir and the L, M, N).

In the second step, the dynamics of the untrained LiESN is sampled. Give a training sequence $y_d(k)$, k = 1, ...,T_t. The reservoir state is updated according to Eq. 1:

$$X(k+1) = (1-a\gamma)X(k) + \gamma f^{PE}(W^{in}u(k) + WX(k) + W^{back}y_d(k) + v(k)) \quad (1)$$

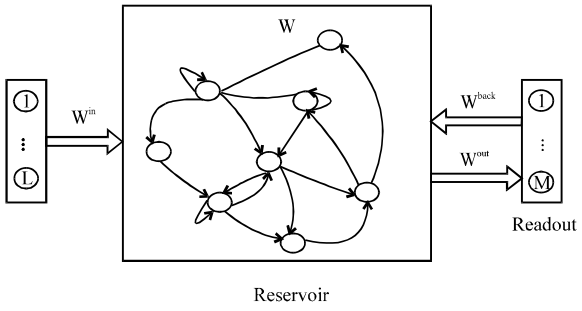

Fig. 1: System architecture of the LiESN

where, f^{PE} is a hyperbolic tangent function applied component-wise; X(k), u(k) and v(k) are the reservoir state, constant input bias and small uniform noise at time instant k, respectively; α and γ are two parameters of the leaky-integrator neurons, α is called the leaky rate, $\alpha>0$, $\gamma>0$, $\alpha\gamma\leq 1_{\Delta}$ (Jaeger *et al*., 2007). X(1) = 0. From the time instant T_0+1 to the time instant T_t, the reservoir state X(k) and training data point $y_d(k)$ are respectively collected into matrix M and matrix T per time instant. In order to eliminate the effect the initial reservoir state results in, the dynamics is not sampled before time instant T_0.

In the third, W^{out} is computed. As a linear readout neuron is assumed, the training problem is equivalent to a linear regression problem. The Wiener-Hopf solution is one of the most common methods to solve the regression problem:

$$y(k+1) = W^{out}X(k+1) \quad (2)$$

$$W^{out} = ((M'M)^{-1}(M'T))' \quad (3)$$

where, the superscript ' means the transpose of a matrix; the superscript $^{-1}$ means the inverse of a matrix.

In the last, the resulting LiESN (W^{in}, W, W^{back}, W^{out}) is ready for exploitation. Given an exploitation sequence y(k), k = 1,...,T_e, $T_e>T_0$. The exploitation sequence and the training sequence originate from the same one system but they may have different initial values ($y(1)\neq y_d(1)$). In order to excite the reservoir, while $k\leq T_0$, y(k) is fed into the reservoir via the feedback connections, the reservoir state is updated according to Eq. 1. While $k>T_0$, however, the resulting LiESN is updated as follows:

$$X(k+1) = (1-a\gamma)X(k) + \gamma f^{PE}(W^{in}u(k) + WX(k) + W^{back}\hat{y}(k)) \quad (4)$$

$$\hat{y}(k+1) = W^{out}X(k+1) \quad (5)$$

where, $\hat{y}(k)$ is the predicting output at time instant k.

We can see from Eq. 4 and 5 that, the predicting output $\hat{y}(k)$ is fed back into the reservoir; a closed-loop emerges. Assuming $u(k) = \Delta$ holds. We can say the LiESN acts as an autonomous dynamical system and carries out a one-step predicting task. Its system architecture is shown in Fig. 2.

By comparing Eq. 1 and 4, it can be said that the sufficient condition ($\rho((1-\alpha\gamma)I+\gamma W)<1$) (Jaeger *et al*., 2007) for the echo state property for the untrained LiESN can not ensure the Lyapunov stability of the resulting LiESN. The stability of the later is mainly determined with another matrix, we note it as $\widetilde{W}$ and its spectral radius as $\rho(\widetilde{W})$, here we call it Effective Spectral Radius (ESR):

$$\widetilde{W} = W + W^{back}W^{out} \tag{6}$$

Theorem 1: For the autonomous LiESN difference dynamical system described by Eq. 4, a sufficient condition for its Lyapunov stability is ESR$\leq\alpha$.

Proof: According to Eq. 4, we have:

$$\|X(k+1)\| = \|(1-a\gamma)X(k) + \gamma f^{PE}(\widetilde{W}X(k))\| \tag{7a}$$

And due to f^{PE} is hyperbolic tangent function, there is:

$$\|f^{PE}(X(k))\| \leq \|X(k)\| \tag{7b}$$

So:

$$\begin{aligned}\|X(k+1)\| &= \|(1-a\gamma)X(k) + \gamma f^{PE}(\widetilde{W}X(k))\| \\ &\leq \|(1-a\gamma)X(k)\| + \gamma\left\|f^{PE}(\widetilde{W}X(k))\right\| \\ &\leq \|(1-a\gamma)X(k)\| + \gamma\left\|\widetilde{W}X(k)\right\| \\ &\leq \left(|1-a\gamma| + \gamma\left\|\widetilde{W}\right\|\right)\|X(k)\|\end{aligned} \tag{7c}$$

Remember that for leaky-integrator neurons, $\alpha\gamma\leq 1$ holds.

So, the Lyapunov stability can be fulfilled only with $\|\widetilde{W}\|\leq\alpha$, i.e. in order to hold $|1-\alpha\gamma|+\gamma\|\widetilde{W}\|\leq 1$, we just only need:

$$\rho(\widetilde{W}) \leq a \tag{7d}$$

Proof ends: In other word, if the loop gain of the autonomous system illustrated with Fig. 2, is smaller than unit, its Lyapunov stability can be ensured.

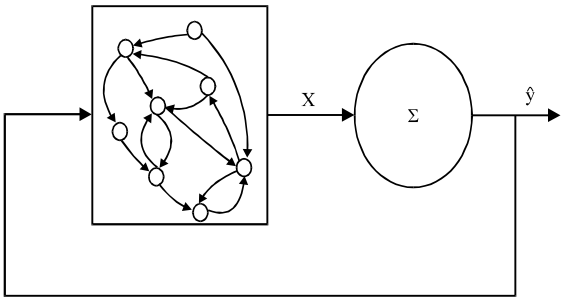

Fig. 2: System architecture of the autonomously running LiESN

PROBLEM-SOLVING BY PSOESN

It can be seen from Theorem 1 that matrix W^{out} has important effect not only on the predicting output precision but also on the Lyapunov stability of the resulting LiESN. Equation 3 only takes into account the predicting output precision, neglects the stability restriction, however, we consider both of them. The output connection weight adaptation problem can be translated into a constrained optimization problem as follows:

$$\begin{aligned}\min_{W^{out}} \quad & \sum_{k=T_0+1}^{T_t}(y_d(k) - W^{out}X(k))^2 \\ \text{s.t.} \quad & \rho(\widetilde{W}) < 1\end{aligned} \tag{8}$$

We can hardly find a gradient-descent based non-stochastic algorithm to solve Eq. 8. PSO is a population-based stochastic algorithm, inspired by social behavior of bird flocking. PSO algorithm can successfully optimize a wide range of continuous functions in a faster, cheaper way compared with other methods (Clerc, 1999; Marinakis *et al*., 2009). The detailed descriptions of PSO algorithm are given in (Clerc, 1999; Clerc and Kennedy, 2002; Eberhart and Shi, 2000). We use it to solve Eq. 8.

The objective function and the stability restriction in Eq. 8 are synthesized into a single evaluation function $\Phi(W^{out})$:

$$\Phi(W^{out}) = f(W^{out}) + r_1 \cdot \theta \cdot h(W^{out})^{\kappa} \tag{9}$$

$$f(W^{out}) = \frac{1}{T_0 - T_t}\sum_{k=T_0+1}^{T_t}(y_d(k) - W^{out}X(k))^2 \tag{10}$$

$$h(W^{out}) = \max[0, (\rho(\widetilde{W}) - 1)] \tag{11}$$

Table 1: Pseudocode for the PSOESN algorithm

```
Initialization
Specify the particle population size (N_p), the maximal number of the
generations (step_m) and the parameters of the penalty function (r_1, r_2, θ)
Generate the initial population of the particles
Evaluate each particle using Eq. 9
Save the optimum solution of each particle
Save the optimum particle of the whole swarm
Main Phase
DO until the maximal number of the generations has been reached
    Calculate the velocity and new position of each particle according to
    Eq. 12
    Calculate the new evaluation of each particle using Eq. 9
    Update the optimum solution of each particle and the optimum particle
    of the whole swarm
End DO
Return the best particle
```

where, $f(W^{out})$ scores the training error, i.e., Mean Square Error (MSE); $h(W^{out})$ scores the restriction violation amount; the parameters r_1, r_2 and θ determine the penalty function.

In order to insure the convergence of PSO algorithm, we use the PSO algorithm with inertial factor (Eberhart and Shi, 2000). Each particle has two parameters: its position (p) and its velocity (v). The evaluation function $\Phi(W^{out})$ is computed using each particle's positional coordinates as input values. Each position and each velocity are adjusted according to Eq. 12 and the evaluation function is computed with the new coordinates at each time step:

$$V(step+1) = \omega_{pso} V(step) + c_1 \cdot rand_1 \cdot (p_{best}(step) - p(step)) + c_2 \cdot rand_2 \cdot (g_{best}(step) - p(step)) \quad (12a)$$

$$p(step+1) = p(step) + V(step+1) \quad (12b)$$

$$\omega_{pso} = \omega_{pso,max} - \frac{\omega_{pso,max} - \omega_{pso,min}}{step_m} \times step \quad (12c)$$

where, ω_{pso} is the inertial factor, $\omega_{pso,\,max}$ and $\omega_{pso,\,min}$ are its maximum and the minimum; $step_m$ is the maximum iteration steps; step is iteration counter; c_1 and c_2 are acceleration coefficients, $c_1 = c_2 = 2$; $rand_1$ and $rand_2$ are two uniform random numbers in [0, 1]; p_{best} is the best position encountered by the particle so-far, g_{best} represents the best position found by any member in the whole swarm population.

Once the matrixes M and T have been collected, Our PSO-based output connection weight adaptation algorithm (PSOESN) is called to seek for the best values of W^{out}, of which the pseudocode is presented in Table 1.

EXPERIMENTS AND RESULTS

The PSOESN is tested on the Multiple Superimposed Oscillators (MSO) problem (Wierstra *et al*., 2005; Xue *et al*., 2007). The trajectories to be learned by LiESN are of the function sin(0.2n)+sin(0.311n), n = 1, 2,...,. The related parameters are listed in Table 2. The non-zero entries of W and W^{back} are all uniformly distributed random variables within the range [-0.5, 0.5]. The parameters of the penalty function (r_1, r_2, θ) are specified as Eq. 13. As the evolvement goes on, the infeasible solutions are punished more and more severely. And if the restriction violation amounts are more, the punishment for the solutions will be severer also:

Table 2: Parameter values used in the simulation experiments

L	N	M	ρ(W)	D	α	T_0	T_t	T_e	N_p	$step_m$
0	10	1	0.02	0.8	1	100	400	1000	30	500

$$r_1 = (step)^{3/2} \quad (13a)$$

$$r_2 = \begin{cases} 3 & h(w^{out}) \geq 2 \\ 2 & 1 \leq h(w^{out}) < 2 \\ 1 & h(w^{out}) < 1 \end{cases} \quad (13b)$$

$$\theta = \begin{cases} 20, & h(W^{out}) \geq 2; \\ 5, & 1 \leq h(W^{out}) < 2; \\ 1, & h(W^{out}) < 1; \end{cases} \quad (13c)$$

We implement the simulation experiments by using MATLAB R14 on PC (Pentium 4, CPU 2.4 GHz, DDRII 1.0 GB). The four-step training algorithm given in Section 2 runs independently 20 times; during each run the third step of the four-step training algorithm is replaced with the PSOESN. Each run consumes about 4 min. The results of all the runs are averaged to evaluate the PSOESN.

Figure 3 and 4 give the evolution curves of $f(W^{out})$ and $\rho(\widetilde{W})$, respectively. It can be seen that the PSOESN has good robustness to different initial values of the matrix W^{out} and both the values of $f(W^{out})$ and the values of $\rho(\widetilde{W})$ have converged on the 200th step. The values of $\rho(\widetilde{W})$ converge at 0.99998. Finally; it satisfies Theorem 1, so the Lyapunov stability of the resulting LiESN can be insured.

During exploitation, the generalization MSE (E_{mse}) and the successful design ratio (S_{ratio}) (Xu *et al*., 2007) are computed:

$$E^i_{mse} = \left(\sum_{k=101}^{1100} (\hat{y}(k) - y(k))^2\right) \Big/ 1000 \quad (14a)$$

$$E_{mse} = \left(\sum_{i=1}^{20} E^i_{mse}\right) \Big/ 20 \quad (14b)$$

$$S_{ratio} = \left(\sum_{i=1}^{20} \delta(E^i_{mse} - \theta)\right) \Big/ 20 \quad (14c)$$

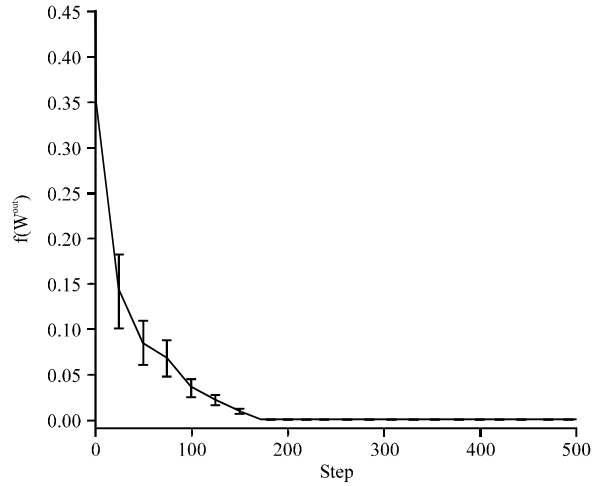

Fig. 3: Evolution of the values of the training error

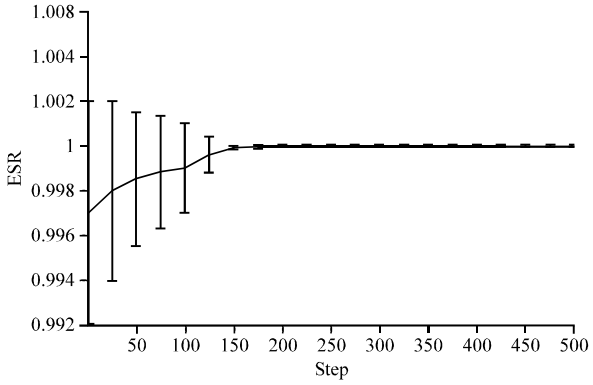

Fig. 4: Evolution of the values of ESR

where, $\hat{y}(k)$ and $y(k)$ are the predicting output and desired output at time instant k; E^i_{mse} is the MSE of the ith run; if E^i_{mse}, then $\delta' = 1$ and else if $E^i_{mse} \geq \theta$, $\delta' = 0$; θ is a threshold, for convenience of comparison, let θ equal 0.0034 (Xu *et al*., 2007).

Figure 5 shows the predicting outputs (red dash line) and the desired outputs (blue solid line) of a resulting LiESN during being exploited which is trained by Eq. 3. The LiESN loses its stability (ESR>1). In fact, according to our experiments, stable LiESN can not be obtained by Eq. 3, the corresponding ESR values always exceed unit for all the 20 times experiments we conducted.

Figure 6 shows the predicting outputs of two resulting LiESNs during being exploited which are trained by the PSOESN. The blue dot line corresponds with the predicting output curve of a resulting LiESN, of which the output weight values are the position of the best particle at the 100th step of the PSO algorithm. Whereas the red dash line corresponds with the predicting output curve of another resulting LiESN, of which the output weight

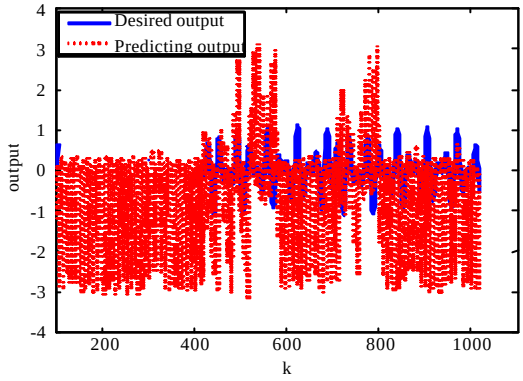

Fig. 5: Outputs of the LiESN trained by Eq. 3

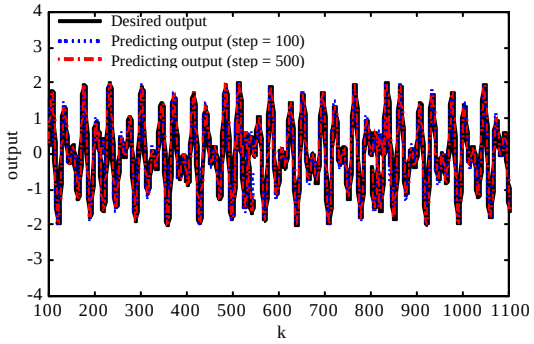

Fig. 6: Outputs of the LiESN trained by the PSOESN

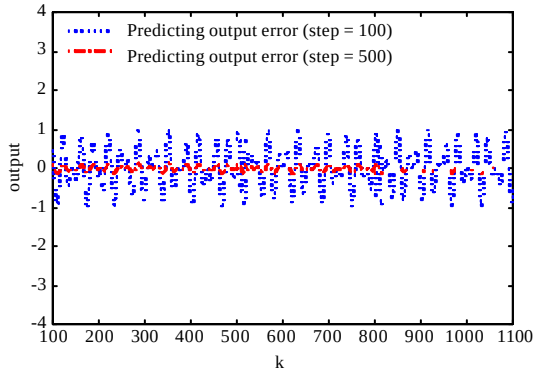

Fig. 7: The corresponding output errors of the LiESN trained by the PSOESN

values are evolved until the 500th step of the PSO algorithm. The black solid line is the desired output curve. The corresponding predicting output errors are plotted in Fig. 7. When the PSO algorithm evolves at the 100th step, a stable LiESN is obtained: the corresponding ESR equals about 0.999. However, the output precision is not enough: the corresponding E_{mse} value is about 0.02. When the PSO

Table 3: Comparison of the results in literature

Method	N	E_{mse}	S_{ratio}
Evolino+LSTM	10	0.0034	/
DESN+RP	400	0.0021	0.228
DESN+MaxInfo	400	0.0003	0.635
Our algorithm	10	0.0005	1

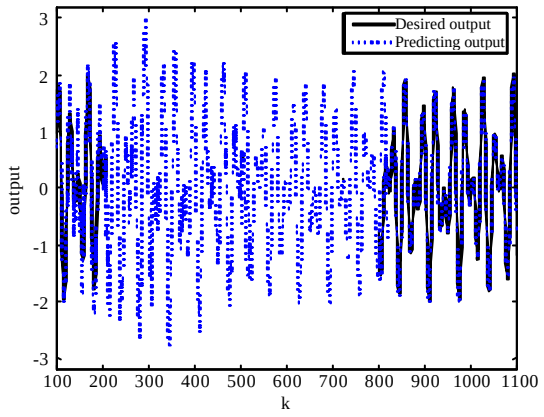

Fig. 8: Outputs of the LiESN perturbed with noise. The LiESN is trained by the PSOESN

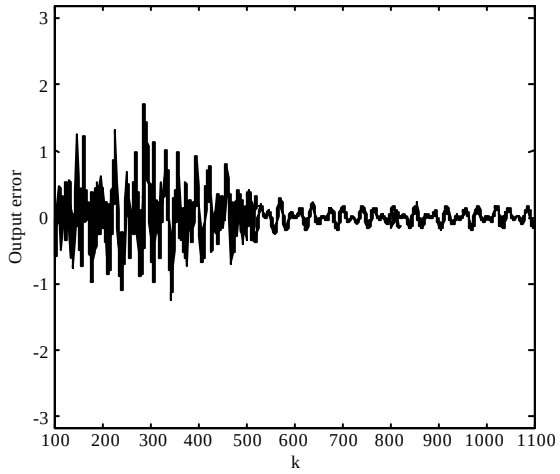

Fig. 9: The corresponding output errors of the LiESN perturbed with noise

algorithm stops its evolvement at the 500th step, the resulting LiESN generates a perfect predicting output curve (E_{mse} = 0.0005, ESR = 0.99998).

The results from literature are collected in Table 3. It can be said that the PSOESN outperforms Evolino+LSTM (Wierstra *et al*., 2005) and DESN+RP (Xu *et al*., 2007) for the same MSO problem. Though the PSOESN is slightly inferior to DESN+MaxInfo (Xu *et al*., 2007) on E_{mse}, it has obvious advantage on N and S_{ratio}.

We verify the noise-robustness of the resulting LiESN trained by the PSOESN. While the LiESN runs in the closed-loop generative mode, its predicting output is perturbed with Gaussian noise (the mean: 0, the variance: 0.01) from time steps 101 - 150. After that time, the noise is dispelled. Figure 8 plots the predicting output curve (the blue dash line) and the desired output curve (the black solid line). Figure 9 plots the corresponding output errors. We can see that about 500 time steps after having been perturbed, the LiESN has already converged back to the MSO attractor.

In fact, we test the effect of different values of α on the resulted LiESNs performances. α is assigned from 0.1 to 2 by step 0.1, i.e., 0.1, 0.2, 0.3, ..., up to 2, 20 totally; another group is from 2.4 to 10 by step 0.4, i.e., 2.4, 2.8, 3.2, ..., up to 10, 20 totally also. However, the experiment results show that the performances (E_{mse} and S_{ratio}) are independent of these α values and all the obtained ESR values are approximated to but always less than theirs corresponding α values. So for conciseness, only the cases with α = 1 are given here.

CONCLUSION

We have analyzed what results in the loss of the Lyapunov stability of the LiESNs running in a closed-loop generative mode. And then, we offer a sufficient condition for its Lyapunov stability. Besides that, we consider the output connection weight adaptation problem as a kind of constrained optimization problem and propose the PSOESN to solve the optimization problem. Our simulation experiment results show that the PSOESN can effectively ensure the output precision as well as the Lyapunov stability. It can be said that the PSOESN is a more effective solution to the output connection weight adaptation problem of such autonomously running LiESNs.

ACKNOWLEDGMENT

The study was supported in part by the Special Fund for Basic Scientific Research of Central colleges, Chang'an University (Grant No. CHD2011JC013).

REFERENCES

Bouzenada, M., M.C. Batouche and Z. Telli, 2007. Neural network for object tracking. Inform. Technol. J., 6: 526-533.

Clerc, M., 1999. The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. Proceedings of the Congress on Evolutionary Computation, (CEC'99), Washington, DC., pp: 1951-1957.

Clerc, M. and J. Kennedy, 2002. The particle swarm-explosion, stability and convergence in a multidimensional complex space. IEEE Trans. Evol. Computat., 6: 58-73.

Eberhart, R.C. and Y. Shi, 2000. Comparing inertia weights and constriction factors in particle swarm optimization. Proceedings of the Congress on Evolutionary Computation, July 16-19, 2000, IEEE Service Center, San Diego, California, pp: 84-88.

Fidele, B., J. Cheeneebash, A. Gopaul and S.S.D. Goorah, 2009. Artificial neural network as a clinical decision-supporting tool to predict cardiovascular disease. Trends Applied Sci. Res., 4: 36-46.

Hati, S. and B.R. Das, 2011. Seam Pucker in Apparels: A Critical Review of Evaluation Methods Asian J. Text., 1: 60-73.

Jaeger, H., 2002. Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach (Tech. Rep. No. 159). German National Research Center for Information Technology, Bremen.

Jaeger, H. and H. Haas, 2004. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless telecommunication. Sci., 304: 78-80.

Jaeger, H., M. Lukosevicius, D. Popovici and U. Siewert, 2007. Optimization and applications of echo state networks with leaky-integrator neurons. Neural Networks, 20: 335-352.

Jang, J.S.R., C.T. Sun and E. Mizutani, 1997. Neuro-fuzzy Soft Computing : A Computational Approach to Learning and Machine Intelligence. Prentice Hall, Upper Saddle River, New Jersey.

Lou, X. and B. Cui, 2006. On the global robust asymptotic stability of BAM neural networks with time-varying delays. Neurocomputing, 70: 273-279.

Manikandan, S. and M. Madheswaran, 2007. A new design of active noise feedforward control systems using delta rule algorithm. Inform. Technol. J., 6: 1162-1165.

Marinakis, Y., M. Marinaki, M. Doumpos and C. Zopounidis, 2009. Ant colony and particle swarm optimization for financial classification problems. Expert Syst. Appl., 36: 10604-10611.

Mosavi, M.R., 2011. Applying genetic algorithm to fast and precise selection of GPS satellites. Asian J. Applied Sci., 4: 229-237.

Salmen, M. and P.G. Ploger, 2005. Echo state networks used for motor control. Proceedings of the IEEE International Conference on Robotics and Automation, April 18-22, 2005, Barcelona, Spain, pp: 1953-1958.

Sharma, A. and S. Agarwal, 2012. Temperature prediction using wavelet neural network. Res. J. Inform. Technol., 4: 22-30.

Song, Q. and Z. Feng, 2010. Effects of connectivity structure of complex echo state network on its prediction performance for nonlinear time series. Neurocomput., 73: 2177-2185.

Verstraeten, D., B. Schrauwen, D. Stroobandt and J. Van Campenhout, 2005. Isolated word recognition with the liquid state machine: A case study. Inform. Process. Lett., 95: 521-528.

Wakaf, Z. and M.M. Saii, 2009. Frontal colored face recognition system. Res. J. Inform. Technol., 1: 17-29.

Wierstra, D., F.J. Gomez and J. Schmidhuber, 2005. Modeling systems with internal state using Evolino. Proceedings of the Conference on Genetic and Evolutionary Computation, June 25-29, 2005, Washington, DC., USA., pp: 1795-1802.

Wyffels, F., B. Schrauwen and D. Stroobandt, 2008. Stable output feedback in reservoir computing using ridge regression. Proceedings of the 18th International Conference on Artificial Neural Networks, September 3-6, 2008, Prague, Czech Republic, pp: 808-817.

Xinhan, H., M. Arif and W. Min, 2002. Intelligent control: A review. Inform. Technol. J., 1: 132-135.

Xu, J., D. Pi, Y.Y. Cao and S. Zhong, 2007. On stability of neural networks by a lyapunov functional-based approach. IEEE Trans. Circuits Syst. I: Regul. Pap., 54: 912-924.

Xue, Y., L. Yang and S. Haykin, 2007. Decoupled echo state networks with lateral inhibition. Neural Networks, 20: 365-376.

Yedjour, D., H. Yedjour and A. Benyettou, 2011. Combining quine mc-cluskey and genetic algorithms for extracting rules from trained neural networks. Asian J. Applied Sci., 4: 72-80.

Zuo-Ping, T., W. Shi-Tong and Du Guo-Cheng, 2008. Glutathione fermentation process modeling based on CCTSK fuzzy neural network. Biotechnology, 7: 73-79.