Stable Trajectory Generator—Echo State Network Trained by Particle Swarm Optimization

Qingsong Song, and Zuren Feng, *Member, IEEE*

***Abstract*— Recurrent neural networks (RNNs) have good modeling capability for nonlinear dynamic systems, but due to the difficulties for training this superiority is discounted. Echo state network (ESN) is a new paradigm for using RNNs with a simpler training method, where an RNN is generated randomly and only a readout is trained. ESN method has quickly become popular in robotics, such as for motor control, for navigation. However, the classical training method for ESNs can not ensure the dynamics asymptotic stability if the trained ESNs run in a closed-loop self-generative mode. The reason is analyzed at first. We then consider the ESN training problem as an optimization problem with a nonlinear constraint, and take a particle swarm optimization (PSO) algorithm solve it. In our simulation experiments, the ESNs are trained as "figure-eight" trajectory generators. The results show that the proposed PSO-based training method can effectively ensure the dynamics asymptotic stability as well as the precision of generating trajectories of the trained ESNs.**

***Key words*— Echo state networks, particle swarm optimization, recurrent neural networks, asymptotic stability.**

I. Introduction

FOR autonomous mobile robots working in unstructured environments, there are many complex subtasks such as perception, localization, path planning, navigation, motion control. These tasks are usually highly nonlinear, noise interference and time-varying in nature [1]. It has been experimentally shown that such ticklish matter can be satisfactorily dealt with in part by biologically-inspired computational intelligence techniques (artificial neural networks, evolutionary computation, swarm intelligence, fuzzy systems, etc.), which provide a rich ground for achieving learning capabilities as well as robustness to noise [2-7].

Manuscript received July 29, 2009. This work was supported in part by the National Natural Science Foundation of China under Grant No. 60875043 and the State Key Development Program for Basic Research of China under Grant No. 2007CB311006.

Qingsong Song is with the System Engineering Institute, Xi'an Jiaotong University, and with the State Key Laboratory of Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an 710049, CHINA (corresponding author, phone: +86-29-82667771; fax: +86-29-82665487; e-mail: qssong@sei.xjtu.edu.cn).

Zuren Feng is with the System Engineering Institute, Xi'an Jiaotong University, and with the State Key Laboratory of Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an 710049, CHINA (e-mail: fzr9910@xjtu.edu.cn).

As another novel artificial recurrent neural network (RNN) modeling method for dynamical systems, echo state network (ESN) has quickly become popular since being proposed [8]. ESN provides rich dynamics along with a fast and efficient learning algorithm. And besides, ESN overcomes the shortcomings of previous RNN learning algorithms (backpropagation through time, real time recurrent learning, etc.), such as slow or non- convergence, local minimum, and time-consuming [9]. Though ESN is initially proposed for time series predictions and indeed performs remarkably well on this kind of data, it has been generalized to many other application domains, of which one is in robotics [10,11,12].

ESN combining with a Kalman filter is used to design a real-time visual tracking controller [13]. In [14-16] the road sign problem and the navigation problem are considered as a kind of temporal tasks [17], and then are solved by ESNs. In [18], an adaptive ESN-based neurocontrol system is developed for trajectory tracking of a non-holonomic mobile robot. And in [19,20], ESNs are applied to the classical motor speed control problem for a RoboCup robot; the experiment results show that the ESN motor-controllers outperform the original PID based controller. In the literature, the inherent superiorities of ESNs are demonstrated, such as good generalization capability, robustness against noise, and adaptation ability for changing situations.

However, in the literature the asymptotic stability of ESNs running with output feedbacks is not given sufficient attention. The asymptotic stability can not be ensured by the spectral radius of the internal connection weight matrixes any more if the trained ESNs work with output feedbacks. There are few contributions studying this problem. The trick—stabilization through wobbling—is the most common method used for keeping up the stability [21,22]; but, the related noise amount is hard to set correctly for specific tasks. In this paper, we propose another training method based on particle swarm optimization (PSO) algorithm for the ESNs running in closed-loop, which considers requirements of the prediction output accuracy as well as the asymptotic stability. Simulation experiment results show that the proposed PSO-based training method is indeed effective.

The remainder of this paper is organized as follows. Section Ⅱ reviews the classical ESN generating and training method and clarifies why the asymptotic stability loses for the ESNs running in closed-loop. The proposed PSO-based training method is described in section Ⅲ. In section Ⅳ, simulation experiments are made to test effectiveness of the proposed method. Finally, conclusions are drawn out in section Ⅴ.

II. ECHO STATE NETWORKS AND ITS ASYMPTOTIC STABILITY

A. Echo State Network

In general, the classical ESN consists of three parts as illustrated with Fig. 1: the front is input, the mid is a large RNN (it is called dynamical reservoir (DR) [8]), and the end is a readout. It is assumed that the ESN has L input neurons, N reservoir neurons and M readout neurons. At time step k, the activations of input neurons are $\mathbf{u}(k)=(u_1(k),u_2(k),...,u_L(k))$, of reservoir neurons are $\mathbf{x}(k)=(x_1(k),x_2(k),...,x_N(k))$, and of readout neurons are $\mathbf{y}(k)=(y_1(k),y_2(k),...,y_M(k))$. Weights for the input connection are in matrix $\mathbf{W}^{in}$, for the reservoir connection in $\mathbf{W}$, for the output connection in $\mathbf{W}^{out}$ and for the connection from the readout output to the reservoir neurons in $\mathbf{W}^{back}$. The corresponding sizes are $N\times L$, $N\times N$, $M\times N$ and $N\times M$. Here assuming there are no direct connections from input neuron to readout neuron and no connections between readout neurons.

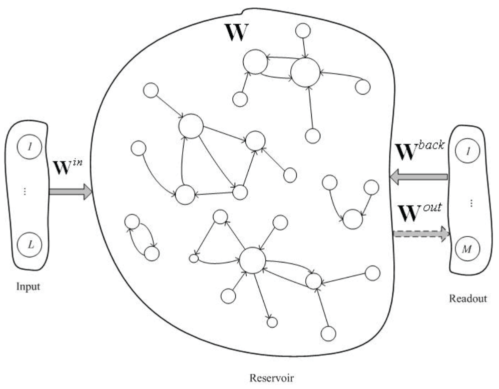

Fig.1 System architecture of ESN.

Three phases are included in an implementation process, i.e., initialization, training and exploitation [23]. We assume in this paper that ESNs are trained as some trajectory-generators.

1) Initialization

Generate randomly the matrixes ($\mathbf{W}^{in}$, $\mathbf{W}$, $\mathbf{W}^{back}$) according to Equation 1~3, where the constants a, b, c and d are the scales and shifts for $\mathbf{W}^{in}$ and $\mathbf{W}^{back}$; rand and sprand are Matlab functions, the former generates uniformly distributed random numbers, the latter generates a sparse uniformly distributed random matrix; $|\lambda_{\max}|$ is the spectral radius of $\mathbf{W}_0$; D and $\rho_{\mathbf{W}}$ respectively stand for the connectivity density and the spectral radius of $\mathbf{W}$. After having been generated in such manner, these three matrixes are fixed all along in the implementation process.

$$\mathbf{W}^{in}=a(\mathrm{rand}(N,L)-b) \tag{1}$$

$$\mathbf{W}^{back}=c(\mathrm{rand}(N,M)-d) \tag{2}$$

$$\begin{aligned}&\mathbf{W}_0=\mathrm{sprand}(N,N,D)\\&\mathbf{W}_0=\mathbf{W}_0/|\lambda_{\max}|\\&\mathbf{W}=\rho_{\mathbf{W}}\mathbf{W}_0\end{aligned} \tag{3}$$

2) Training

From the desired trajectory we get a training sequence $\mathbf{y}_d(k)$, $k=1,2,...,T_t$. The reservoir state is initialized with $\mathbf{x}(0)=\mathbf{0}$. The generating ESN ($\mathbf{W}^{in}$, $\mathbf{W}$, $\mathbf{W}^{back}$) is driven by teacher-forcing the training sequence as Equation 4, where f is the reservoir neurons' activation function (hyperbolic tangent function in this paper); $\mathbf{u}(k)$ is often a constant bias input; $\mathbf{v}(k)$ is a noise term sampled from a uniform distribution over $[-e, e]$.

$$\mathbf{x}(k+1)=f(\mathbf{W}^{in}\mathbf{u}(k+1)+\mathbf{W}\mathbf{x}(k)+\mathbf{W}^{back}\mathbf{y}_d(k)+\mathbf{v}(k+1)) \tag{4}$$

From time step T_0+1 to T_t, $T_0<T_t$, the reservoir states $\mathbf{x}(k)$ s are collected row-wise in a state collecting matrix $\mathbf{M}$. At the same time, the sigmoid-inverted training data $(f^o)^{-1}(\mathbf{y}_d(k))$ s are collected row-wise in another matrix $\mathbf{T}$, here f^o is the readout neurons' activation function.

The output weight matrix $\mathbf{W}^{out}$ is computed simply with Equation 5, where $[\cdot]^{-1}$ is the inverse matrix of $[\cdot]$, $[\cdot]'$ is the transpose matrix of $[\cdot]$.

$$\mathbf{W}^{out}=(\mathbf{M}^{-1}\mathbf{T})' \tag{5}$$

Now, we obtain the trained ESN ($\mathbf{W}^{in}$, $\mathbf{W}$, $\mathbf{W}^{out}$, $\mathbf{W}^{back}$).

3) Exploitation

The trained ESN is driven by an exploitation sequence $\mathbf{y}(k)$, $k=1,2,...,T_e$. From $k=1$ to $k=T_0$, the reservoir state is updated with Equation 4. In this period, the outputs are no significant and discarded. However, from T_0+1 to T_e, the reservoir state $\mathbf{x}(k)$ and prediction output $\hat{\mathbf{y}}(k)$ are updated as follows.

$$\mathbf{x}(k+1)=f(\mathbf{W}^{in}\mathbf{u}(k+1)+\mathbf{W}\mathbf{x}(k)+\mathbf{W}^{back}\hat{\mathbf{y}}(k)) \tag{6}$$

$$\hat{\mathbf{y}}(k+1)=f^o(\mathbf{W}^{out}\mathbf{x}(k+1)) \tag{7}$$

B. Problem on Asymptotic Stability

Echo state property is expected to prevent ESNs from having a chaotic, unbounded behavior [24]. In the initialization phase, in order to ensure echo state property the spectral radius $\rho_{\mathbf{W}}$ is assigned to be smaller than unit. It is correct for the ESNs running in opened-loop; however, for the ESNs running in closed-loop, it is wrong.

Assuming f^o is identity function and $\mathbf{u}(k)\equiv\mathbf{0}$. By substituting $\mathbf{W}^{out}\mathbf{x}(k)$ for $\hat{\mathbf{y}}(k)$ in Equation 6, we obtain Equation 8. It can be see that during the exploitation, the trained ESN is an autonomous difference dynamical system with an initial value $\mathbf{x}(T_0+1)$.

$$\mathbf{x}(k+1)=f((\mathbf{W}+\mathbf{W}^{back}\mathbf{W}^{out})\mathbf{x}(k)) \tag{8}$$

$$\tilde{\mathbf{W}}=(\mathbf{W}+\mathbf{W}^{back}\mathbf{W}^{out}) \tag{9}$$

Hyperbolic tangent function satisfies the Lipschitz condition [25], so the asymptotic stability of the trained ESN is only determined by the property of the matrix $\tilde{\mathbf{W}}$

(Equation 9). According to the idea of the proving method for the proposition 3 in [22], it can be proved that it is the spectral radius of $\tilde{\mathbf{W}}$ (we call it effective spectral radius (ESR), and note it as $\rho(\tilde{\mathbf{W}})$) rather than $\mathbf{W}$ which determines the trained ESN's stability.

III. Training Method Based on PSO Algorithm

Based on Equation 9, it can be said that $\mathbf{W}^{out}$ not only determines the prediction accuracy, but also has effect on the asymptotic stability of the trained ESNs. The classical training method as Equation 5 takes the prediction accuracy into account, neglects the requirement for the asymptotic stability, however. We consider it as an important factor which induces the unstable results in [21]. Here by considering both the precision and the stability in a comprehensive way, we translate the training problem into a continuous optimization problem with a nonlinear constraint, which is illustrated with Equation 10.

$$\begin{aligned} \min \quad & \sum_{k=T_0+1}^{T_t} (\mathbf{y}_d(k) - \mathbf{W}^{out}\mathbf{x}(k))^2 / (T_t - T_0) \\ s.t. \quad & \rho(\widetilde{\mathbf{W}}) < 1 \end{aligned} \tag{10}$$

We find that gradient-based deterministic methods can hardly solve Equation 10 due to the highly nonlinear constraint. PSO is a stochastic optimization algorithm modeled on swarm intelligence [26]. It has been demonstrated that in many research and application areas, PSO can find a better solution in a search space in a faster, cheaper way compared with other methods [27,28,29]. We choose PSO algorithm to solve Equation 10.

Since the reservoir of ESN does not have to be trained for a particular task, it supports parallel computing: multiple readout neurons can be independently trained to perform different tasks in parallel [30]. If M >1, the readout neurons are trained one by one by the PSO-based method in this paper.

Firstly, the objective function and the constraint in Equation 10 are integrated into a single evaluation function $\Phi(\mathbf{W}^{out})$.

$$\Phi(\mathbf{W}^{out}) = g(\mathbf{W}^{out}) + r \cdot \theta \cdot h(\mathbf{W}^{out}) \tag{11}$$

$$g(\mathbf{W}^{out}) = \sum_{k=T_0+1}^{T_t} (\mathbf{y}_d(k) - \mathbf{W}^{out}\mathbf{x}(k))^2 / (T_t - T_0) \tag{12}$$

$$h(\mathbf{W}^{out}) = \max[0, (\rho(\widetilde{\mathbf{W}}) - 1)] \tag{13}$$

where $g(\mathbf{W}^{out})$ measures the mean squared training error (MSE); $h(\mathbf{W}^{out})$ characters violation amount of the restraint; r and θ are dynamic parameters of the evaluation function.

And subsequently, PSO algorithm is called to solve Equation 11. The swarm has N_s particles; each particle has a position $\mathbf{p}$ and a velocity $\mathbf{V}$ in the N dimensional space. Here, $\mathbf{p}$ corresponds to $\mathbf{W}^{out}$, i.e., each value of $\mathbf{p}$ denotes a possible solution of Equation 10. At each evolution step ($step$), $\mathbf{p}$ and $\mathbf{V}$ are updated by Equation 14 and 15.

$$\begin{aligned} \mathbf{V}(step+1) = \chi(\mathbf{V}(step) & \\ + c_1 \cdot \text{rand} \cdot (\mathbf{p}_{best}(step) - \mathbf{p}(step)) & \\ + c_2 \cdot \text{rand} \cdot (\mathbf{g}_{best}(step) - \mathbf{p}(step))) & \end{aligned} \tag{14}$$

$$\mathbf{p}(step+1) = \mathbf{p}(step) + \mathbf{V}(step+1) \tag{15}$$

where χ is constriction factor, c_1 and c_2 are acceleration coefficients; $\mathbf{p}_{best}$ and $\mathbf{g}_{best}$ are the particle's and the whole swarm's best position at current evolution step, respectively. Here, to ensure the convergence, we use the PSO algorithm with a constriction factor, χ = 0.729, c_1 = c_2 = 1.496 [31,32].

The proposed PSO-based training method is implemented as follows.

s1. Initialization. Once $\mathbf{M}$ and $\mathbf{T}$ have been obtained, set N_s , r , θ and $step_m$ (the maximum of evolution step). Randomly place all particles in the N dimensional space, and initialize their velocities. Each particle's current position is recorded as its best position $\mathbf{p}_{best}(1)$. Among all N_s $\mathbf{p}_{best}(1)$ s, the one having the minimum $\Phi(\mathbf{W}^{out})$ is recorded as $\mathbf{g}_{best}(1)$.

s2. Update $\mathbf{p}$ and $\mathbf{V}$. Update each particle according with Equation 14 and 15, obtain a new swarm.

s3. Evaluation. Substitute new $\mathbf{p}$ for $\mathbf{W}^{out}$ in Equation 11; get new evaluation for each particle.

s4. Update $\mathbf{p}_{best}$ and $\mathbf{g}_{best}$. For each particle, compare its $\Phi(\mathbf{p}(step))$ with $\Phi(\mathbf{p}_{best}(step-1))$. If $\Phi(\mathbf{p}(step)) < \Phi(\mathbf{p}_{best}(step-1))$, then $\mathbf{p}(step) \to \mathbf{p}_{best}$. Note $\Phi_{\min}$ as the minimum of all $\Phi(\mathbf{p}(step))$ s. Compare $\Phi_{\min}$ and $\Phi(\mathbf{g}_{best})$. If $\Phi_{\min} < \Phi(\mathbf{g}_{best})$, then $\mathbf{p}(step)|_{\Phi(\mathbf{p}(step))=\Phi_{\min}} \to \mathbf{g}_{best}$.

s5. Repeat s2~s4 until $step = step_m$. Now, the resulting $\mathbf{g}_{best}$ is just the trained result of $\mathbf{W}^{out}$.

IV. Simulation Experiments

The "figure eight" generation task has been considered as a benchmark problem for ESNs to test whether the trained ESNs are stable or not. A "figure eight" can be simply interpreted as the superposition of a sine (for the x direction) and a cosine of half the sine's frequency (for the y direction) [21], as shown in Fig. 2.

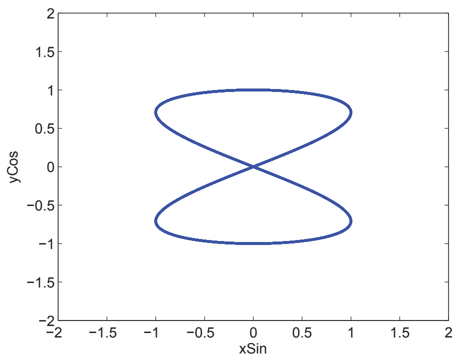

Fig. 2 Schematic diagram of the "figure eight".

Two ESN groups are independently tested on the same "figure eight" task. Each group consists of 20 independently generated ESNs (N_e =20). One group is trained by the classical method described in section Ⅱ, the other is trained by the proposed PSO-based method.

The training sequence and exploitation sequences are processed as the manner being used in [21]. The figure-eight trajectory is discrete and one full ride along the eight consists of 200 sample points.

Table Ⅰ. Parameter assignments in the simulation experiments

a	b	c	d	ρ_W	D	L	N	M
2	0.5	2	0.5	0.8	0.8	0	20	2

(cont.).

e	T_0	T_t	T_e	N_s	$step_m$
1e-6	1,000	3,000	100,000	30	500

Assignments to the parameters involved in the simulation experiments are listed in Table Ⅰ. The dynamic parameters of the evaluation function are set as Equation 16 and 17.

$$r = step^{3/2} \tag{16}$$

$$\theta = \begin{cases} 50, & \text{if } h(\mathbf{W}^{out}) > 1; \\ 20, & \text{if } 0.5 < h(\mathbf{W}^{out}) \le 1; \\ 2, & \text{if } 0.2 < h(\mathbf{W}^{out}) \le 0.5; \\ 1, & \text{if } h(\mathbf{W}^{out}) \le 0.2; \end{cases} \tag{17}$$

The performances are evaluated with two indexes. One is the mean squared error during the exploitation (exploitation MSE), and the other is the successful training ratio, which is defined as Equation 18.

$$s_r = \sum_{i=1}^{N_e} \delta(1-\rho(\widetilde{\mathbf{W}}))/N_e$$
$$\delta(x_v) = \begin{cases} 1 & x_v > 0 \\ 0 & x_v \le 0 \end{cases} \tag{18}$$

where N_e is the number of ESNs which are randomly independently generated.

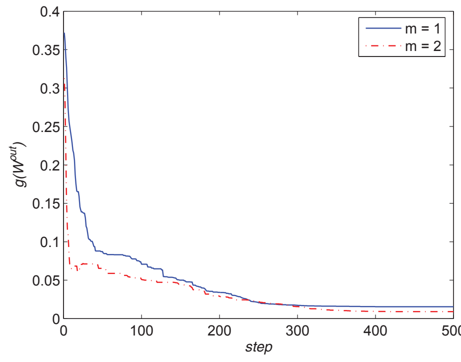

Fig.3 Evolution curves of the training MSE. "m=1" denotes one readout neuron, of which the corresponding training MSE evolution curve is denoted as the blue solid curve. "m=2" denotes another readout neuron, of which the corresponding training MSE evolution curve is denoted as the red dashed curve.

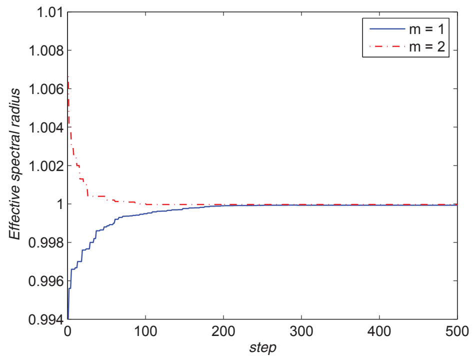

Fig. 4 Evolution curves of the effective spectral radius. "m=1" denotes one readout neuron, of which the corresponding ESR evolution curve is denoted as the blue solid curve, "m=2" denotes another readout neuron, of which the corresponding ESR evolution curve is denoted as the red dashed curve.

Fig. 3 and Fig. 4 respectively give the evolution curves of $g(\mathbf{W}^{out})$ and $\rho(\widetilde{\mathbf{W}})$. It can be seen that both $g(\mathbf{W}^{out})$ and $\rho(\widetilde{\mathbf{W}})$ have already converged about at $step$ =300. For the x direction data, the corresponding ESR converges at 0.99997, as being shown with the red dashed curve (with legend "m = 2"); while for the y direction data, the ESR converges at 0.99994, as being shown with the blue solid curve (with legend "m = 1"). So echo state property is ensured, the trained ESNs have the asymptotic stability.

Fig. 5 demonstrates the generated trajectory from an ESN trained by the classical method (red dashed curve). The blue solid curve denotes a true trajectory of the "figure-eight". It can be seen that the ESN initially generates a crude "figure-eight"; however, the trajectory in the x direction loses its stability quickly: the x-coordinate data is far away from [-1, 1], and the corresponding ESR value is about 11.2 for the output neuron generating the x direction data.

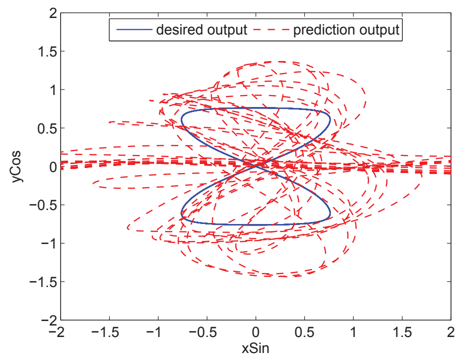

Fig.5 Prediction output curve of an ESN trained by the classical method.

Fig. 6 demonstrates a predicting output trajectory of an ESN trained by the proposed PSO-based method. The output weight values correspond to the position of the best particle at the 100th evolvement step of the PSO algorithm. The trained ESN is stable: the ESR value is about 0.99997 for the output neuron generating the x direction data, about 0.99949 for another output neuron generating the y direction data. But the prediction precisions are insufficient: the exploitation MSE value is about 0.16 for the x direction data, about 0.28 for the y direction data.

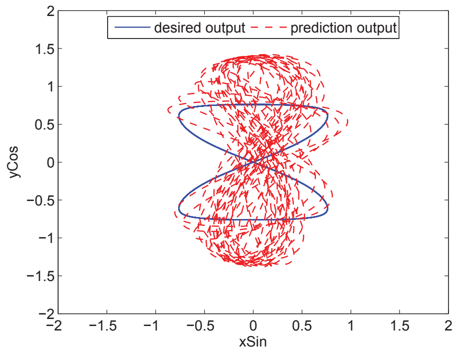

Fig. 6 Prediction output curve of an ESN trained by the proposed PSO-based method. The particle swarm is evolved over 100 steps (*step* =100).

Fig. 7 also demonstrates a predicting output trajectory of another ESN trained by the same PSO-based method; however, the output weight values correspond to the position of the best particle at the 500th evolvement step of the PSO algorithm. The trained ESN is also stable: the corresponding ESR values are about 0.99997 and 0.99994 respectively. But in this case, the prediction precisions are sufficient: the corresponding MSE values are about 0.08 and 0.12 respectively.

The noise-robustness of the ESNs trained by the proposed PSO-based method is also verified. While the ESNs run in self-generative mode, their predicting outputs are perturbed with Gaussian noise (the mean: 0, the variance: 0.05) from time steps 1001~1200, i.e., just one full ride along the figure-eight. After that period, the noise is dispelled. It is demonstrated from the experiment results that about 900 time steps (four and a half full rides along the figure-eight) after having been perturbed, the ESNs have already converged back to the “figure-eight” attractor. The PSO trained ESNs have good noise-robustness.

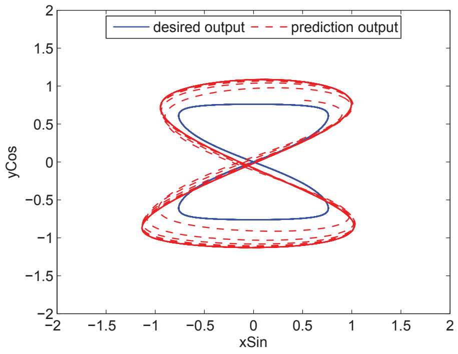

Fig. 7 Prediction output curve of an ESN trained by the proposed PSO-based method. The particle swarm is evolved over 500 steps (*step* =500).

The simulation experiment results are summarized in Table Ⅱ, they are the averages over the results of all 20 randomly independently generated ESNs (N_e = 20). The classical training method described in section Ⅱ can not ensure the ESNs stability: the exploitation MSE values are always far larger than 2; the ESR values are far larger than unit; the obtain s_r equals zero for the x direction data, equals 0.4 for the y direction data. As a comparison, the proposed PSO-based training method can effectively assure the stability of the trained ESNs as well as the precision of the generated “figure-eight” trajectories: the obtained ESR values are always smaller than unit (unexpectedly, the ESRs nearly approach unit); and the s_r values equal 1 as the PSO algorithm has evolved over 500 steps.

Table Ⅱ. Contrast of the results between the classical and the PSO-based training method.

/	MSE exploitation		ESR		s_r	
	m = 1	m = 2	m = 1	m = 2	m = 1	m = 2
The classical method	$\gg 2$	$\gg 2$	6.23186	11.8225	0.4	0
PSO-based method (*step* =100)	0.28	0.16	0.99949	0.99997	1	0.9
PSO-based method (*step* =500)	0.12	0.08	0.99994	0.99997	1	1

V. CONCLUSIONS

We have reviewed the applications of ESNs in robotics, and pointed out that the asymptotic stability of the trained ESNs running in closed-loop can not be ensured any longer by the parameter-setting that the spectral radius of reservoir connection weight matrix is smaller than unit. And then we translate the output weight training problem into a nonlinearly constrained optimization problem, which is solved by the PSO algorithm. The simulation experiment results show that the proposed PSO-based training method can effectively ensure the stability as well as the precision of the generating trajectories of the trained ESNs.

REFERENCES

[1] R. Siegwart, I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. The MIT Press, 2004.
[2] R. D. Beer, R. D. Quinn, H. J. Chiel, R. E. Ritzmann, “Biologically inspired approaches to robotics: what can we learn from insects?” Communications of the ACM, vol. 40, issue 3, pp. 30-38, 1997.
[3] S. Nolfi, D. Floreano, *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines.* Cambridge, MA: MIT Press/Bradford Books, 2000.
[4] R. Pfeifer, M. Lungarella, F. Iida, “Self-Organization, Embodiment, and Biologically Inspired Robotics,” Science, vol. 318, no. 5853, pp. 1088-1093, 2007.
[5] D. A. Pomerleau, *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishers, Norwell, MA, USA, 1993.
[6] Z. Zenn Bien, M. –J. Chung, P.-H. Chang, D.-S. Kwon, et al., "Integration of a Rehabilitation Robotic System (KARES II) with Human-friendly Man-Machine Interaction Units,” Autonomous Robots, vol. 16, no. 2, pp. 165-191, 2004.
[7] J.-S. Kim, J.-B. Kim, K.-J. Song, B. Min, Z. Zenn Bien, "On-line Motion Control of Avatar Using Hand Gesture Recognition," The Institute of Electronics Engineers of Korea, vol. 36-C, no. 6, pp. 52-62, 1999.

[8] H. Jaeger, H. Haass, "Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication," Science, vol. 304, pp. 78-80, 2004.

[9] J. S. R. Jang, C. T. Sun, E. Mizutani, *Neuro-Fuzzy and Soft Computing: a Computational Approach to Learning and Machine Intelligence*. Prentice Hall, 1997.

[10] K. Ishii, T. van der Zant, V. Becanovic, P. G. Plöger, "Identification of motion with echo state network," in: MTTS/IEEE TECHNO-OCEAN, vol. 3, pp. 1205-1210, 2004.

[11] C. Hartland, N. Bredeche, M. Sebag, "Memory-enhanced Evolutionary Robotics: The Echo State Network Approach," in: Proc. IEEE Congress on Evolutionary Computation, pp.2788-2795, IEEE, 2009.

[12] T. van der Zant, V. Becanovic, K. Ishii, H. U. Kobialka, P. G. Plöger, "Finding good echo state networks to control an underwater robot using evolutionary computations," in: the 5th IFAC/EURON symposium on Intelligent autonomous vehicles, vol. 1, pp.221-226, 2004.

[13] C. Y. Tsai, X. Dutoit, K. T. Song, H. Van Brussel, M. Nuttin, "Visual state estimation using self-tuning Kalman filter and echo state network," in: IEEE International Conference on Robotics and Automation, pp. 917-922, 2008.

[14] E. Antonelo, B. Schrauwen, D. Stroobandt, "Mobile Robot Control in the Road Sign Problem using Reservoir Computing Networks," in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 911-916, 2008.

[15] S. Hellbach, S. Strauss, J. Eggert, E. Körner, H.-M. Gross, "Echo State Networks for Online Prediction of Movement Data - Comparing Investigations," In: Int. Conf. on Artificial Neural Networks (ICANN), Prague, LNCS 5163, pp. 710-719, Springer Verlag, 2008.

[16] C. Hartland, N. Bredeche, "Using echo state networks for robot navigation behavior acquisition," in: Proc. IEEE International Conference on Robotics and Biomimetics, pp. 201-206, IEEE, 2007.

[17] M. Lukoševičius, H. Jaeger, "Reservoir computing approaches to recurrent neural network training," Computer Science Review, doi:10.1016/j.cosrev.2009.03.005, in press.

[18] M. Oubbati, M. Schanz, P. Levi, "Kinematic and dynamic adaptive control of a nonholonomic mobile robot using a RNN," in: Proceedings of 2005 IEEE International Symposium on Computational Intelligence in Robotics and Automation, (CIRA), pp. 27-33, 2005.

[19] P. G. Ploeger, A. Arghir, T. Guenther, R. Hosseiny, "Echo state networks for mobile robot modeling and control," in: Proc. RoboCup, pp. 157-168, 2003.

[20] M. Salmen, P. G. Ploger, "Echo State Networks used for Motor Control," in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 1953-1958, 2005.

[21] H. Jaeger, M. Lukoševičius, D. Popovici, U. Siewert, "Optimization and applications of echo state networks with leaky-integrator neurons," Neural Networks, vol.20, pp. 335-352, 2007.

[22] H. Jaeger, "the echo state approach to analyzing and training recurrent neural networks," GMD Report 148, German National Research Center for Information Technology, 2001.

[23] H. Jaeger, "Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "echo state network" approach," Tech. Rep. No. 159, Bremen: German National Research Center for Information Technology, 2002.

[24] H. Jaeger, "Echo state network," Scholarpedia, vol. 2, pp.2330, 2007. Available: http://www.scholarpedia.org/article/Echo_state_network

[25] J. T. Oden, *Applied Functional Analysis*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1979.

[26] M. Clerc, *Particle Swarm Optimization*. Wiley-ISTE, 2006.

[27] M. Clerc, J. Kennedy, "The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space," IEEE Transactions on Evolutionary Computation, vol.6, pp. 58-73, 2002.

[28] J. Pugh, A. Martinoli, "Multi-robot learning with particle swarm optimization," in: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, pp. 441-448, 2006.

[29] W. Gueaieb, M.S. Miah, "Mobile robot navigation using particle swarm optimization and noisy RFID communication," in: 2008 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, pp. 111-116, 2008.

[30] W. Maass, T. Natschläger, H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, pp. 2531-2560, 2002.

[31] R.C. Eberhart, Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in: Proceedings of the 2000 Congress on Evolutionary Computation, vol. 1, pp. 84-88, 2000.

[32] M. Clerc, "The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization," in: Proc. 1999 ICEC, Washington, DC, pp. 1951-1957, 1999.