A robust recurrent simultaneous perturbation stochastic approximation training algorithm for recurrent neural networks

Song, Qing; Xu, Zhao; Wang, Danwei

2014

Xu, Z., Song, Q., & Wang, D. (2014). A robust recurrent simultaneous perturbation stochastic approximation training algorithm for recurrent neural networks. *Neural Computing and Applications*, 24(7-8). 1851-1866.

<https://hdl.handle.net/10356/105067>

<https://doi.org/10.1007/s00521-013-1436-5>

© 2014 Springer-Verlag London Limited. This is the author created version of a work that has been peer reviewed and accepted for publication by *Neural Computing and Applications*, Springer-Verlag London Limited. It incorporates referee's comments but changes resulting from the publishing process, such as copyediting, structural formatting, may not be reflected in this document. The published version is available at: [<http://dx.doi.org/10.1007/s00521-013-1436-5>].

Downloaded on 26 Apr 2023 23:18:04 SGT

Neural Computing and Applications

A Robust Recurrent Simultaneous Perturbation Stochastic Approximation Training Algorithm for Recurrent Neural Networks

--Manuscript Draft--

Manuscript Number:	NCA-1890R2
Full Title:	A Robust Recurrent Simultaneous Perturbation Stochastic Approximation Training Algorithm for Recurrent Neural Networks
Article Type:	Original Article
Keywords:	Recurrent neural networks; stability and convergence analysis; spsa
Corresponding Author:	Zhao Xu, Ph.D. Singapore, SINGAPORE
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	
Corresponding Author's Secondary Institution:	
First Author:	Zhao Xu, Ph.D.
First Author Secondary Information:	
Order of Authors:	Zhao Xu, Ph.D. qing song Danwei Wang
Order of Authors Secondary Information:	
Response to Reviewers:	<p>Replies to reviewer 3:</p> <p>Comments: For simplicity of notations, the time subscripts of several parameters are ignored. However, the time subscripts should be kept in some definitions such as (34), (37) to differentiate variables and constants. Replies: the time subscripts have been added in the equations, please see Page 12 and Page 18.</p> <p>Comments: Spacing is irregular between paragraphs and eqs, e.g. Page 23. Replies: Page 23 has been revised.</p> <p>Comments: In the simulation, the author simply shows the results without detailed explanation the reason why RRSPSA gets better performance compared with other algorithms. Replies: Simulation results show that the BP training algorithm performs worst as compared to other methods. The reason is that the second partial derivatives are not considered in the BP training. For RTRL, the slower convergence compared with RRSPSA is mainly caused by the small fixed learning rate and absent of the recurrent hybrid adaptive parameter for guaranteed weight convergence in the presence of noise. RRSPSA utilizes the specifically designed three adaptive parameters to maximize training speed for recurrent training signal while exhibiting certain weight convergence properties with only two objective function measurements. Combination of the three adaptive parameters makes RRSPSA converge fast with the maximum effective learning rate and guaranteed weight convergence. Please see Page 20.</p> <p>Comments: The paper seems only considers the synaptic dynamic, does the author considers the neuron dynamics? Replies: Neural engineers have usually studied only the synaptic dynamic system with weight updating or the neural dynamic system with neuron state changing independently, which are equally important. However, since the celebrated</p>

backpropagation training algorithm has been successfully applied to multilayered RNNs like the Jordan and Elman networks, taxonomy boundaries between studies of the two dynamic systems become somehow fuzzy. It should be pointed that our study does not deal with the neural dynamic system because the speed of weight convergence is usually much slower than that of neurons.

Comments: For simplicity of notations, the time subscripts of several parameters are ignored. However, the time subscripts should be kept in some definitions such as (34), (37) to differentiate variables and constants.

Replies: the time subscripts have been added in the equations, please see Page 12 and Page 18.

Comments: Spacing is irregular between paragraphs and eqs, e.g. Page 23.

Replies: Page 23 has been revised.

Comments: In the simulation, the author simply shows the results without detailed explanation the reason why RRSPSA gets better performance compared with other algorithms.

Replies: Simulation results show that the BP training algorithm performs worst as compared to other methods. The reason is that the second partial derivatives are not considered in the BP training. For RTRL, the slower convergence compared with RRSPSA is mainly caused by the small fixed learning rate and absent of the recurrent hybrid adaptive parameter for guaranteed weight convergence in the presence of noise. RRSPSA utilizes the specifically designed three adaptive parameters to maximize training speed for recurrent training signal while exhibiting certain weight convergence properties with only two objective function measurements. Combination of the three adaptive parameters makes RRSPSA converge fast with the maximum effective learning rate and guaranteed weight convergence. Please see Page 20.

Comments: The paper seems only considers the synaptic dynamic, does the author considers the neuron dynamics?

Replies: Neural engineers have usually studied only the synaptic dynamic system with weight updating or the neural dynamic system with neuron state changing independently, which are equally important. However, since the celebrated backpropagation training algorithm has been successfully applied to multilayered RNNs like the Jordan and Elman networks, taxonomy boundaries between studies of the two dynamic systems become somehow fuzzy. It should be pointed that our study does not deal with the neural dynamic system because the speed of weight convergence is usually much slower than that of neurons.

Noname manuscript No.
(will be inserted by the editor)

A Robust Recurrent Simultaneous Perturbation Stochastic Approximation Training Algorithm for Recurrent Neural Networks

Zhao Xu · Qing Song · Danwei wang

Received: date / Accepted: date

Abstract Training of recurrent neural networks (RNNs) introduces considerable computational complexities due to the need for gradient evaluations. How to get fast convergence speed and low computational complexity remains a challenging and open topic. Besides, the transient response of learning process of RNNs is a critical issue, especially for on-line applications. Conventional RNNs training algorithms such as the backpropagation through time (BPTT) and real-time recurrent learning (RTRL) have not adequately satisfied these requirements because they often suffer from slow convergence speed. If a large learning rate is chosen to improve performance, the training process may become unstable in terms of weight divergence. In this paper, a novel training algorithm of RNN, named robust recurrent simultaneous perturbation stochastic approximation (RRSPSA), is developed with a specially designed recurrent hybrid adaptive parameter and adaptive learning rates. RRSPSA is a powerful novel twin-engine simultaneous perturbation stochastic approximation (SPSA) type of RNN training algorithm. It utilizes three specially designed adaptive parameters to maximize training speed for a recurrent training signal while exhibiting certain weight convergence properties with only two objective function measurements as the original SPSA algorithm. The RRSPSA is proved with guaranteed weight convergence and system stability in the sense of Lyapunov function. Computer simulations were carried out to demonstrate applicability of the theoretical results.

Zhao Xu
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
Tel.: +65-83590114
E-mail: xuzh0007@e.ntu.edu.sg

Qing Song
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
Tel.: +65-67904530
E-mail: eqsong@e.ntu.edu.sg

Danwei Wang
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
Tel.: +65-67905376
E-mail: EDWWANG@e.ntu.edu.sg

1 Main Notation List

k	Discrete time step.
m	Output dimension.
n_I	Input dimension.
n_h	Hidden neuron number.
p^v	$= m \times n_h$. Dimension of output layer weight vector.
p^w	$= n_i \times n_h$. Dimension of hidden layer weight vector.
l	Number of time-delayed outputs.
$u_i(k)$	$i = 1, \dots, m$. External input at time step k .
$x(k)$	$\in R^{n_I}$. Neural network input vector.
$\Delta x(k)$	$\in R^{n_I}$. Perturbation vectors on input $x(k-1)$.
$y(k)$	$\in R^m$. Desired output vector.
$\hat{y}(k)$	$\in R^m$. Estimated output vector.
$\varepsilon(k)$	$\in R^m$. Total disturbance vector.
$e(k)$	$\in R^m$. Output estimation error vector.
$\bar{V}(k)$	$\in R^{p^v}$. Estimated weight vector of output layer.
$\bar{V}^*(k)$	$\in R^{p^v}$. Ideal weight vector of output layer.
$\tilde{V}(k)$	$\in R^{p^v}$. Estimated error vector of output layer.
$\bar{W}(k)$	$\in R^{p^w}$. Estimated weight vector of hidden layer.
$\bar{W}^*(k)$	$\in R^{p^w}$. Ideal weight vector of hidden layer.
$\tilde{W}(k)$	$\in R^{p^w}$. Estimated weight vector of hidden layer.
$\hat{W}_{j,:}(k)$	$\in R^{n_i}$. The j th row vector of hidden layer weight matrix $\hat{W}(k)$.
$\delta^v(k)$	Equivalent approximation errors of the loss function of output layer.
$\delta^w(k)$	Equivalent approximation errors of the loss function of hidden layer.
$c(k)$	Perturbation gain parameter of the SPSA.
$\Delta^v(k)$	$\in R^{p^v}$. Perturbation vectors of output layer.
$r^v(k)$	$\in R^{p^v}$. Perturbation vectors of output layer.
$\Delta^w(k)$	$\in R^{p^w}$. Perturbation vectors of hidden layer.
$r^w(k)$	$\in R^{p^w}$. Perturbation vectors of hidden layer.
$H(\bar{W}(k), x(k))$	$= H(k) \in R^{m \times p^w}$. Nonlinear activation function matrix.
$h_j(k)$	$= h(\hat{W}_{j,:}(k)x(k))$. The nonlinear activation function and the scalar element of $H(\bar{V}(k), x(k))$.
$\alpha^v(k)$	Adaptive learning rate of output layer.
$\alpha^w(k)$	Adaptive learning rate of hidden layer.
α^v, α^w	Positive scalars.
$\rho^v(k)$	Normalization factor of output layer.
$\rho^w(k)$	Normalization factor of hidden layer.
$\beta^v(k)$	Recurrent hybrid adaptive parameter of output layer.

$\beta^w(k)$	Recurrent hybrid adaptive parameter of hidden layer.
$\mu_j(k)$	$\in \mathcal{R}; 1 \leq j \leq n_h$. Mean value of the input vectors of the j th hidden layer neuron.
$\tilde{\mu}_j(k)$	$\in \mathcal{R}; 1 \leq j \leq n_h$. Mean value of the hidden layer weight vector of the j th hidden layer neuron.
τ	Positive scalar.
λ	Positive gain parameter of the threshold function.
η	A small perturbation parameter.

2 Introduction

Recurrent neural networks (RNNs) are inherently dynamic nonlinear systems which have signal feedback connections and are able to process temporal sequences. RNNs with free synaptic weights are more useful for complex dynamical systems such as nonlinear dynamic system identification, time-series prediction, and adaptive control as compared to Hopfield neural networks (HNNs), in which the weights are fixed and symmetric [1–8].

To maximize potentials and capabilities of the RNNs, one of the preliminary tasks is to design an efficient training algorithm with fast weight convergence speed and low computational complexity. Gradient descent optimization techniques are often used for neural networks which consist of adjustable weights along the negative direction of error gradient. However, due to the dynamic structure of RNNs, the obtained error gradient tends to be very small in a classical BP training algorithm with the fixed learning step, because RNN weight depends on not only the current output but also output of past steps. It is widely acknowledged that recurrent derivative of the multilayered RNNs, i.e. the recurrent training signal (see Fig. 1), which contains information of dynamic time dependence in training, is necessary to obtain good time response of RNNs [9, 10]. From computational efficiency and weight convergence points of view, Spall proposed a simultaneous perturbation stochastic approximation (SPSA) method to find roots and minima of neural network functions which are contaminated with noise in [11]. The name of “simultaneous perturbation” arises from the fact that all elements of unknown weight parameters are being varied simultaneously. SPSA has the potential to be significantly more efficient than the usual p -dimensional algorithms (of Kiefer-Wolfowitz/Blum type) that are based on standard finite-difference gradient approximations and it is shown in [11] that approximately the same level of estimation accuracy can typically be achieved with only $1/p$ th the amount of data needed in the standard approach. The main advantage of SPSA is its simplicity and excellent weight convergence property, in which only two objective function measurements are used. Thus, SPSA is more economical in terms of loss function evaluation which is usually the most expensive part of an optimization problem. SPSA has attracted great attention in application areas such as neural networks training, adaptive control and model parameter estimation [7, 12–16].

Conventional training methods of RNNs are usually classified as backpropagation through time (BPTT) [17, 18] and real time recurrent learning (RTRL) [19, 20] algorithms which often suffer from drawbacks of slow convergence speed and heavy computational load. In fact, BPTT cannot be used for online application due to computational problems. For a standard RTRL training, recursive weight updating requires a small learning step to obtain weight convergence and system stability of RNNs. If a large learning rate is selected to speed up weight updating, the training process of RTRL may lead to a large steady-state error or even weight divergence [6, 9].

In the previous work, we have developed an adaptive SPSA based training scheme for FNNs and demonstrated that the adaptive SPSA algorithm can be used to derive a fast training algorithm and guaranteed weight convergence [7]. In this paper, we will extend our SPSA research to the RNNs with free weight updating and propose a robust recurrent simultaneous perturbation stochastic approximation (RRSPSA) algorithm under the framework of deterministic system with guaranteed weight convergence. Compared with FNNs, RNNs are dynamic systems and contain recurrent connections in their structure. Considering the time dependence of the signals of RNNs, not only the weights at the current time steps are to be perturbed, but also those at the previous time steps, which makes the learning easy to be diverge and the convergence analysis complicated to obtain. The key characteristic of RRSPSA training algorithm is to use a recurrent hybrid adaptive parameter together with adaptive learning rate and dead zone concept. The recurrent hybrid adaptive parameter can automatically switch off recurrent training signal whenever the weight convergence and stability conditions are violated (see Fig. 1 and explanation in Section III and IV). There are several advantages to train RNNs based on RRSPSA algorithm. First, RRSPSA is relatively easy to implement as compared to other RNN training methods such as BPTT due to its RTRL type of training and simplicity. Second, RRSPSA provides excellent convergence property based on simultaneous perturbation of weight, which is similar to the standard SPSA. Finally, RRSPSA has capability to improve RNN training speed with guaranteed weight convergence over the standard RTRL algorithm by using adaptive learning method. RRSPSA is a powerful twin-engine SPSA type of RNN training algorithm. It utilizes the specifically designed three adaptive parameters to maximize training speed for recurrent training signal while exhibiting certain weight convergence properties with only two objective function measurements as a standard SPSA algorithm. Combination of the three adaptive parameters makes RRSPSA converge fast with the maximum effective learning rate and guaranteed weight convergence. Robust stability and weight convergence proofs of RRSPSA algorithm are provided based on Lyapunov function.

The rest of this paper is organized as follows. In Sections II, we briefly introduce the structure of RNN and the conventional SPSA algorithm. In Sections III and IV, the RRSPSA algorithm and robustness analysis are derived for the output and hidden layers, respectively. Computer simulation results are presented in Section V to show the efficiency of our proposed RRSPSA. Section VI draws the final conclusions.

3 Basics of the RNN and SPSA

To simplify notation of this paper, we consider a three-layer m -external input and m -output RNN with the output feedback structure as shown in Fig. 1, which can be easily extended into a RNN with arbitrary numbers of inputs and outputs. Output vector of the RNN is $\hat{y}(k)$ which can be presented as

$$\hat{y}(k) = H(\bar{W}(k), x(k)) \bar{V}(k) = [\hat{y}_1(k), \dots, \hat{y}_m(k)]^T \in R^m \quad (1)$$

where $\bar{V}(k) \in R^{p^v}$ ($p^v = m \times n_h$) is the long vector of the output layer weight matrix $\hat{V}(k) \in R^{n_h \times m}$ defined by

$$\bar{V}(k) = [\hat{V}_{1,1}(k), \dots, \hat{V}_{n_h,1}(k), \dots, \hat{V}_{1,m}(k), \dots, \hat{V}_{n_h,m}(k)]^T \quad (2)$$

and $x(k) \in R^{n_l}$ ($n_l = (l+1) \times m$) is the input vector of the RNN defined by

$$x(k) = [x_1(k), x_2(k), \dots, x_{n_l}(k)]^T = [\hat{y}^T(k-1), \dots, \hat{y}^T(k-l), u_1(k), \dots, u_m(k)]^T \quad (3)$$

and $\hat{W}(k) \in R^{p^w}$ ($p^w = n_l \times n_h$) is the long vector of the hidden layer weight matrix $\hat{W}(k) \in R^{n_h \times n_l}$ defined by

$$\bar{W}(k) = [\hat{W}_{1,:}(k), \hat{W}_{2,:}(k), \dots, \hat{W}_{n_h,:}(k)]. \quad (4)$$

$H(\bar{W}(k), x(k)) \in R^{m \times p^v}$ is the nonlinear activation block-diagonal matrix given by

$$H(k) = H(\bar{W}(k), x(k)) = \begin{bmatrix} h_1(k), \dots, h_{n_h}(k), 0, \dots & 0, \dots \\ 0, \dots & \ddots & \vdots \\ \vdots & & \\ 0, \dots & 0, \dots & h_1(k), \dots, h_{n_h}(k) \end{bmatrix} \quad (5)$$

where $h_j(k)$ with $1 \leq j \leq n_h$ is one of the most popular nonlinear activation functions,

$$h_j(k) = h(\hat{W}_{j,:}(k)x(k)) = \frac{1}{1 + \exp(-4\lambda \hat{W}_{j,:}(k)x(k))} \quad (6)$$

with $\hat{W}_{j,:}(k) \in R^{n_l}$ being the j th row vector of $\hat{W}(k)$ and $\lambda > 0$ is the gain parameter of the threshold function that is defined specifically for ease of use later when deriving the sector condition of the hidden layer.

The output vector $\hat{y}(k)$ is an approximation of the ideal nonlinear function $y(k) = H(\bar{W}^*, x(k)) \bar{V}^*$ with $\bar{V}^*$ and $\bar{W}^*$ being the ideal output layer and hidden layer weight respectively.

SPSA algorithm is to solve the problem of finding an optimal weight θ^* of the gradient equation of neural networks

$$\hat{g}(\hat{\theta}(k)) = \frac{\partial L(\hat{\theta}(k))}{\partial \hat{\theta}(k)} = 0$$

for some differentiable loss function $L: R^p \rightarrow R^1$, where $\hat{\theta}(k)$ is the estimated weight vector of neural networks. The basic idea is to vary all elements of the vector $\hat{\theta}(k)$

Fig. 1 Block diagram of RRSPSA training for the output feedback based RNN.

simultaneously and approximate the gradient function using only two measurements of the loss function (7) and (8) without requiring exact derivatives or a large number of function evaluation which provides significant improvement in efficiency over the standard gradient decent algorithms. Define

$$\hat{g}^+(\hat{\theta}(k)) = L(\hat{\theta}(k) + c(k)\Delta(k)) + \varepsilon^+(k) \quad (7)$$

$$\hat{g}^-(\hat{\theta}(k)) = L(\hat{\theta}(k) - c(k)\Delta(k)) + \varepsilon^-(k) \quad (8)$$

where $\varepsilon^+(k)$ and $\varepsilon^-(k)$ represent measurement noise terms, $c(k)$ is a scalar and $\Delta(k) = [\Delta_1(k), \dots, \Delta_p(k)]$ is the perturbation vector. Then, the form of the estimation of $g(\hat{\theta})$ at the k th iteration is

$$\hat{g}(\hat{\theta}(k)) = \left[\frac{\hat{g}^+(\hat{\theta}(k)) - \hat{g}^-(\hat{\theta}(k))}{2c(k)\Delta_1(k)} \quad \dots \quad \frac{\hat{g}^+(\hat{\theta}(k)) - \hat{g}^-(\hat{\theta}(k))}{2c(k)\Delta_p(k)} \right]^T \quad (9)$$

Theory and numerical experience in [11] indicate that SPSSA can be significantly more efficient than the standard finite difference-based algorithms in large-dimensional problems.

Inspired by the excellent economical and convergence properties of SPSSA, we propose the RRSPSA algorithm under the framework of deterministic system based on the idea of simultaneous perturbation with guaranteed weight convergence. The loss function for RRSPSA is defined as

$$L(\hat{\theta}(k)) = \frac{1}{2} \|e(k)\|^2 \quad (10)$$

$$e(k) = y(k) - \hat{y}(k) + \varepsilon(k) \quad (11)$$

where $y(k)$ is the ideal output and a function of θ^* , $\hat{y}(k)$ is a function of $\hat{\theta}(k)$, and $\varepsilon(k)$ is a disturbance term. Similar to SPSSA, the RRSPSA algorithm seeks to find an optimal weight vector θ^* of gradient equation, i.e., the weight vector that minimized

differentiable $L(\hat{\theta}(k))$. That is, the proposed RRSPSA algorithm for updating $\hat{\theta}(k) \in R^{(p^w+p^v)}$ as an estimation of the ideal weight vector θ^* is of the form

$$\hat{\theta}(k+1) = \hat{\theta}(k) - \alpha(k+1)\hat{g}(\hat{\theta}(k)) \quad (12)$$

where $\alpha(k+1)$ is an adaptive learning rate and $\hat{g}(\hat{\theta}(k))$ is an approximation of the gradient of the loss function. This approximated gradient function (normalized) is of the SPSA form

$$\hat{g}(\hat{\theta}(k)) = \frac{L(\hat{\theta}(k) + c(k+1)\Delta(k+1)) - L(\hat{\theta}(k) - c(k+1)\Delta(k+1)) + \delta(k)}{2c(k+1)\rho(k+1)}r(k+1) \quad (13)$$

where $\delta(k)$ is an equivalent approximation error of the loss function, which is also called the measurement noise, and $\rho(k+1)$ is the normalization factor. $\Delta(k+1)$, $r(k+1)$, and $c(k+1) > 0$ are the controlling parameters of the algorithm defined as

1. The perturbation vector

$$\Delta(k+1) = [\Delta_1(k+1), \dots, \Delta_{p^w+p^v}(k+1)]^T \in R^{(p^w+p^v)} \quad (14)$$

is a bounded random directional vector that is used to perturb the weight vector simultaneously and can be generated randomly.

2. The sequence of $r(k+1)$ is defined as

$$r(k+1) = \left[\frac{1}{\Delta_1(k+1)}, \dots, \frac{1}{\Delta_{p^w+p^v}(k+1)} \right]^T \in R^{(p^w+p^v)}. \quad (15)$$

3. $c(k+1) > 0$ is a sequence of positive numbers [11, 14, 15]. $c(k+1)$ can be chosen as a small constant number for a slow time-varying system as pointed in [15] and it controls the size of perturbations.

Note that for simplicity of notations, we ignore the time subscripts of several parameters where no confusion is caused such as $\Delta(k+1)$, $r(k+1)$, $c(k+1)$, $\delta(k)$ and $\alpha(k+1)$.

4 Output Layer Training and Stability Analysis

In a multilayered neural network, the output layer weight vector $\bar{V}(k)$ and hidden layer weight vector $\bar{W}(k)$ are normally updated separately using different learning rates as shown in Fig. 1 as in the standard BP training algorithm [9]. During the updating of the output layer weights, the hidden layer weights are fixed. We are now ready to propose the RRSPSA algorithm to update the estimated weight vector $\bar{V}(k)$ of output layer. That is,

$$\bar{V}(k+1) = \bar{V}(k) - \alpha^v(k+1)\hat{g}^v(\bar{V}(k)) \quad (16)$$

where $\hat{g}^v(\bar{V}(k)) \in R^{p^v}$ is the normalized gradient approximation that uses simultaneous perturbation vectors $\Delta^v(k+1) \in R^{p^v}$ and $r^v(k+1) \in R^{p^v}$ to stimulate weight.

$\Delta^v(k+1)$ and $r^v(k+1)$ are, respectively, the first p^v components of perturbation vectors defined in (14) and (15).

$$\hat{g}^v(\bar{V}(k)) = -e^T(k)H(k)\Delta^v r^v(\rho^v)^{-1} - \beta^v(k+1)e^T(k)A(k)r^v(2c\rho^v(k+1))^{-1} \quad (17)$$

where $A(k) \in R^m$ is the extended recurrent gradient defined as

$$A(k) = H\left(\bar{W}(k), \hat{D}^v(k)(\bar{V}(k) + c\Delta^v)\right)\bar{V}(k) - H\left(\bar{W}(k), \hat{D}^v(k)(\bar{V}(k) - c\Delta^v)\right)\bar{V}(k) \quad (18)$$

$$\hat{D}^v(k) = \left[H^T(k-1), \dots, H^T(k-l), 0, \dots, 0 \right]^T. \quad (19)$$

For simplicity of notations, we ignore the time subscripts of $\Delta^v(k+1)$, $r^v(k+1)$, $\beta^v(k+1)$, $\rho^v(k+1)$ and $\alpha^v(k+1)$ in the following part of the paper.

Remark 1 Our purpose is to find the desired output layer weight vector $\bar{V}^*$ of the gradient equation

$$\hat{g}(\bar{V}(k)) = \frac{\partial L(\bar{V}(k))}{\partial \bar{V}(k)} = 0.$$

Inspired by the excellent properties of SPSA, we obtain $\hat{g}(\bar{V}(k))$ by varying all elements of unknown weight parameters simultaneously as shown in the proof of Lemma 1 under the framework of deterministic system. We will show that RRSPSA has the same form as SPSA, and only two objective function measurements are used at each iteration, which maintains the efficiency of SPSA. However, the weight convergence proof and stability analysis are quite different from those in [11] under the framework of deterministic system as shown in Theorem 1 and 2. For RNNs, the outputs depend on not only the weights at current time steps, but also those at the previous time steps since the current input vector contains the delayed outputs. The first term on the right side of equation in (17) is the result of perturbation of the feedforward signals in RNNs and $A(k)$ in (18) is the result of perturbation of the recurrent signals which makes the training complicated and the system easy to be unstable. In addition to the simplicity and efficiency of RRSPSA, one powerful aspect of the developed RRSPSA algorithm is that it is able to use effectively large adaptive learning rate α^v to accelerate weight convergence speed with guaranteed system stability. To avoid weight divergence at each time step, an adaptive learning rate β^v is used in (17) to control the recurrent signals and switch off the recurrent connections whenever the convergence condition is violated as shown in (38). The normalization factor ρ^v is also used to bound the signals in learning algorithms as traditionally did in adaptive control system [5, 7]. α^v is to guarantee the weight convergence during training, which means the weights are only updated when the convergence and stability requirements are met and it does not need to be small as in RTRL training algorithm. The sufficient conditions for the three key parameters are shown in (34), (37), and (38). $\hat{D}^v(k)$ is obtained during the mathematical deduction in the proof of Lemma 1.

Lemma 1 The normalized gradient approximation in (17) is an equivalent presentation of the standard SPSA algorithm in equation (13).

Proof We will prove that (17) is obtained by perturbing all the weights simultaneously as SPSA did. Rewrite the loss function defined in (10) as

$$L(\hat{\theta}(k)) = L(\bar{V}(k), \bar{W}(k)) = \frac{1}{2} \|y(k) - \hat{y}(k) + \varepsilon(k)\|^2. \quad (20)$$

Further, we use $\hat{y}(k) = H(k)\bar{V}(k)$ in (1) to define

$$\begin{aligned} \hat{y}^{v+}(\bar{V}(k)) &= H(\bar{V}(k) + c\Delta^v) + H(\bar{W}, (x(k) + \Delta x(k+1))) \bar{V}(k) \\ &= H(\bar{V}(k) + c\Delta^v) + H\left(\bar{W}, \left([\hat{y}^T(k-1), \dots, \hat{y}^T(k-l), 0, \dots, 0]^T + \Delta x(k+1)\right)\right) \bar{V}(k) \\ &= H(\bar{V}(k) + c\Delta^v) + H\left(\bar{W}, \left([\bar{V}^T(k-1)H^T(k-1), \dots, \right. \right. \\ &\quad \left. \left. \bar{V}^T(k-l)H^T(k-l), 0, \dots, 0]^T + \Delta x(k+1)\right)\right) \bar{V}(k) \\ &= H(\bar{V}(k) + c\Delta^v) + H\left(\bar{W}, \left[(\bar{V}(k-1) + c\Delta^v)^T H^T(k-1), \right. \right. \\ &\quad \left. \left. \dots, (\bar{V}(k-l) + c\Delta^v)^T H^T(k-l), 0, \dots, 0]^T\right)\right) \bar{V}(k). \end{aligned} \quad (21)$$

$\hat{y}^{v+}(\bar{V}(k))$ has the same motivation as $g^+(\hat{\theta}(k))$ of SPSA shown as in (7) which is to perturb all the weights simultaneously. Different from FNN training which only perturb the current weight, the input vector must be perturbed as well for RNN training because of the time dependence in RNNs. The disturbance $\Delta x(k+1)$ is to perturb the weights at previous time steps which are contained in $x(k)$ as seen in (3). However, the last equation of (21) is difficult to implement. Here we assume the model parameters do not change apparently between each iteration, i.e. $\bar{V}(k) \approx \bar{V}(k-1) \dots \approx \bar{V}(k-l)$. We can derive a similar approach as RTRL introduced by Williams and Zipser [19, 20]. Moreover, such approximation makes the proposed algorithm real time and adaptive in a recursive fashion. Under the approximation,

$$\begin{aligned} \hat{y}^{v+}(\bar{V}(k)) &\approx H(k)(\bar{V}(k) + c\Delta^v) + H\left(\bar{W}(k), \left[(\bar{V}(k) + c\Delta^v)^T H^T(k-1), \right. \right. \\ &\quad \left. \left. \dots, (\bar{V}(k) + c\Delta^v)^T H^T(k-l), 0, \dots, 0]^T\right)\right) \bar{V}(k) \\ &= H(k)(\bar{V}(k) + c\Delta^v) + H\left(\bar{W}(k), \left[H^T(k-1), \dots, H^T(k-l), 0, \dots, 0\right]^T\right) \\ &\quad \times (\bar{V}(k) + c\Delta^v) \bar{V}(k) \\ &= H(k)(\bar{V}(k) + c\Delta^v) + H\left(\bar{W}(k), \hat{D}^v(k)(\bar{V}(k) + c\Delta^v)\right) \bar{V}(k) \end{aligned} \quad (22)$$

where $\hat{D}^v(k)$ is defined in (19). By the approximation $\bar{V}(k) \approx \bar{V}(k-1) \cdots \approx \bar{V}(k-l)$, the recurrent term $H^T(k-1), \dots, H^T(k-l)$ of (19) can be iteratively obtained from the previous steps to speed up the calculation of RRSPSA (normally, the input $u(k)$ is independent from weight so that the last entries of $\hat{D}^v(k)$ are zero vectors). Weight convergence and system stability can be guaranteed by using the three adaptive learning rates which will be explained detailly later.

Similar to (21), we can get

$$\begin{aligned} \hat{y}^{v-}(\bar{V}(k)) &= H(k)(\bar{V}(k) - c\Delta^v) + H(\bar{W}(k), (x(k) - \Delta x(k+1)))\bar{V}(k) \\ &\approx H(k)(\bar{V}(k) - c\Delta^v) + H\left(\bar{W}(k), \hat{D}^v(k)(\bar{V}(k) - c\Delta^v)\right)\bar{V}(k) \end{aligned} \quad (23)$$

so that

$$\hat{y}^{v+}(\bar{V}(k)) - \hat{y}^{v-}(\bar{V}(k)) \approx (\bar{V}(k))2H(k)c\Delta^v + A(k) \quad (24)$$

where the extended recurrent gradient $A(k)$ is defined as (18). $A(k)$ in (24) is the result of recurrent connections. To guarantee weight convergence and system stability during training, we insert an adaptive recurrent hybrid parameter $\beta^v(k+1)$ (see Fig. 1) to control the recurrent training signal. As shown in (38), the convergence condition will be examined at each step and the recurrent contribution will be switched off whenever the convergence condition is violated. Thus, (24) can be rewritten as follows

$$\hat{y}^{v+}(\bar{V}(k)) - \hat{y}^{v-}(\bar{V}(k)) \approx 2H(k)c\Delta^v + \beta^v A(k). \quad (25)$$

According to (13), we can get

$$\begin{aligned} &\hat{g}^v(\bar{V}(k)) \\ &= \frac{L(\bar{W}(k), \bar{V}(k) + c\Delta^v) - L(\bar{W}(k), \bar{V}(k) - c\Delta^v) + \delta^v(k)}{2c\rho^v} r^v \\ &= \frac{\|y(k) - \hat{y}^{v+}(\bar{V}(k))\|^2 - \|y(k) - \hat{y}^{v-}(\bar{V}(k))\|^2 + 2\delta^v(k)}{4c\rho^v} r^v \\ &= \left\{ (y(k) - \hat{y}^{v+}(\bar{V}(k)) + y(k) - \hat{y}^{v-}(\bar{V}(k)))^T \right. \\ &\quad \times (\hat{y}^{v-}(\bar{V}(k)) - \hat{y}^{v+}(\bar{V}(k))) + 2\delta^v(k) \Big\} r^v (4c\rho^v)^{-1} \\ &= (y(k) - \hat{y}(k) + \varepsilon(k))^T (\hat{y}^{v-}(\bar{V}(k)) - \hat{y}^{v+}(\bar{V}(k))) r^v (2c\rho^v)^{-1} \\ &= e^T(k) (\hat{y}^{v-}(\bar{V}(k)) - \hat{y}^{v+}(\bar{V}(k))) r^v (2c\rho^v)^{-1} \\ &\approx -e^T(k) H(k) \Delta^v r^v (\rho^v)^{-1} - \beta^v e^T(k) A(k) r^v (2c\rho^v)^{-1}. \end{aligned} \quad (26)$$

$e(k)$ in the fifth equality of (26) comes from (11). The fourth equality of (26) reveals the relationship between gradient approximation error $\delta^v(k)$ and system disturbance

$\varepsilon(k)$, i.e.,

$$\begin{aligned}\delta^v(k) &= (-2\hat{y}(k) + \hat{y}^{v+}(\bar{V}(k)) + \hat{y}^{v-}(\bar{V}(k)) + \varepsilon(k))^T \\ &\quad \times (\hat{y}^{v-}(\bar{V}(k)) - \hat{y}^{v+}(\bar{V}(k))) \\ &= \varepsilon^T(k) (\hat{y}^{v-}(\bar{V}(k)) - \hat{y}^{v+}(\bar{V}(k))).\end{aligned}\quad (27)$$

In the fifth equality of (26), the gradient approximation presentation appears exactly the same as the original one in [11] (see [11, eq. (2.2)]). Only two objective function measurements are used at each step which maintains the efficiency of SPSA.

The training error $e(k)$ comes from the weight estimation error and the disturbance. For stability analysis, $e(k)$ will be linked to the weight estimating error of the output layer and the disturbance. Define the weight estimating errors of the output layer and hidden layer as

$$\tilde{V}(k) = \bar{V}^* - \bar{V}(k) \quad (28)$$

$$\tilde{W}(k) = \bar{W}^* - \bar{W}(k) \quad (29)$$

where $\bar{W}^*$ and $\bar{V}^*$ are the ideal (optimal) weight vectors of $\bar{W}(k)$ and $\bar{V}(k)$. According to (11), the training error can be rewritten as

$$\begin{aligned}e(k) &= y(k) - \hat{y}(k) + \varepsilon(k) \\ &= H(\bar{W}^*, x(k)) \bar{V}^* - H(k) \bar{V}(k) + \varepsilon(k) \\ &= H(\bar{W}^*, x(k)) \bar{V}^* - H(k) \bar{V}^* + H(k) \bar{V}^* - H(k) \bar{V}(k) + \varepsilon(k).\end{aligned}\quad (30)$$

Because the term $H(\bar{W}^*, x(k)) \bar{V}^* - H(k) \bar{V}^*$ is temporarily bounded during output layer training (hidden layer weights are fixed), we define

$$\begin{aligned}\tilde{e}^v(k) &= H(\bar{W}^*, x(k)) \bar{V}^* - H(k) \bar{V}^* + \varepsilon(k) \\ &= \tilde{H}(\bar{W}(k), \bar{W}^*, x(k)) \bar{V}^* + \varepsilon(k)\end{aligned}\quad (31)$$

with $\tilde{H}(\bar{W}(k), \bar{W}^*, x(k)) = H(\bar{W}^*, x(k)) - H(k)$. The training error caused by the weight estimation error of the output layer is defined as

$$\phi^v(k) = -H(k) \tilde{V}(k). \quad (32)$$

Then, (30) can be rewritten as

$$e(k) = -\phi^v(k) + \tilde{e}^v(k). \quad (33)$$

Hence, the training error is directly linked to the weight estimating error of the output layer contained in $\phi^v(k)$ and the disturbance $\tilde{e}^v(k)$. This is critical for the convergence proof.

Theorem 1 *If the output layer of the RNN is trained by RRSPSA algorithm (16), $\bar{V}(k)$ is guaranteed to be convergent in the sense of Lyapunov function $\left\| \tilde{V}(k+1) \right\|^2 - \left\| \tilde{V}(k) \right\|^2 \leq 0$, $\forall k$ with $\tilde{V}(k) = \bar{V}^* - \bar{V}(k)$ and the training error will be L_2 -stable in the sense of $\tilde{e}^v(k) \in L_2$ and $\phi^v(k) \in L_2$.*

Proof See Section 7.

The conditions for the three adaptive learning rates in (16) and (17) are as follows. The detailed procedure to obtain these conditions is shown in the proof of Theorem 1.

1. $\rho^v(k+1)$ is the normalization factor defined by

$$\rho^v(k+1) = \tau \rho^v(k) + \max \left\{ \frac{\alpha^v Z(k)}{\rho^v + 0.5 \beta^v Y(k)} + 1, \bar{\rho} \right\} \quad (34)$$

where $0 < \tau < 1$, $0 < \alpha^v \leq 1$, and $\bar{\rho}$ are positive constants and

$$Y(k) = \frac{(r^v)^T (\eta I + H^T(k)H(k))^{-1} H^T(k)A(k)}{c} \quad (35)$$

where η is a small perturbation parameter.

$$Z(k) = \|(2H(k)c\Delta^v + \beta^v A(k))\|^2 \|r^v\|^2 (4(c)^2)^{-1}. \quad (36)$$

2. $\alpha^v(k+1)$ is an adaptive learning rate similar to dead zone approach in the classical neural network and adaptive control

$$\begin{cases} \alpha^v(k+1) = \alpha^v, & \text{if } \|e(k)\| \geq \Lambda(k) \\ \alpha^v(k+1) = 0, & \text{if } \|e(k)\| < \Lambda(k) \end{cases} \quad (37)$$

with $\Lambda(k) = \tilde{e}_m^v(k) / \sqrt{1 - \frac{\alpha^v(\rho^v)^{-1} Z(k)}{\rho^v + 0.5 \beta^v Y(k)}}$ and $\tilde{e}_m^v(k)$ being the maximum value of $\tilde{e}^v(k)$ defined in (31).

3. $\beta^v(k+1)$ is the recurrent hybrid adaptive parameter determined by

$$\begin{cases} \beta^v(k+1) = 1, & \text{if } Y(k) \geq 0 \\ \beta^v(k+1) = 0, & \text{if } Y(k) < 0. \end{cases} \quad (38)$$

Remark 2 According to the theoretical analysis, three parameters $\beta^v(k+1)$, $\alpha^v(k+1)$ and $\rho^v(k+1)$ play a key role in the proof of weight convergence and stability analysis. They are designed to guarantee the negativeness of Lyapunov function as in (72). The function of $\beta^v(k+1)$ is to automatically switch off the important recurrent training signal in the lower left corner of Fig.1 when the convergence and stability requirements are violated resulted from recurrent training signals. The normalization factor $\rho^v(k+1)$ is also used to bound the signals in learning algorithms as traditionally did in adaptive control system [5, 7]. $\alpha^v(k+1)$ is to guarantee weight convergence during training, which means that the weights are only updated when convergence and stability requirements are met. Besides, $\alpha^v(k+1)$ does not need to be small as in RTRL training algorithm which will make the training by RRSPSA faster than that by RTRL. From the definition of $\alpha^v(k+1)$ in (37), it can be seen that the weights will be updated ($\alpha^v(k+1) = \alpha^v$) only if the training error is bigger than the dead zone value which means the learning will be stopped if the training error is lower than the dead zone level. This is to provide good generalization other than to learn the system noise. In this paper, we will choose a suitable value for $\tilde{e}_m^v(k)$ according to the noise level.

Remark 3 Perturbation size $c(k+1)$ is a critical variable as discussed in the original SPSSA papers [7, 11, 14]. A natural question is whether $c(k+1)$ will affect the learning procedure in the recurrent learning algorithm. The answer is yes. It is not obvious to see from the updating formula of $\hat{g}^v(\bar{V}(k))$ in (17). However, this can be easily seen by expanding the second term $A(k)$ in (17) to be explicit in $c(k)$.

According to the mean value theory and that activation function in (6) is a non-decreasing function, it can be readily shown that there exist unique positive mean values $\mu_j(k)$ such that

$$h(\hat{W}_{j,:}(k)x_1(k)) - h(\hat{W}_{j,:}(k)x_2(k)) = \mu_j(k)\hat{W}_{j,:}(k)(x_1(k) - x_2(k)) \quad (39)$$

where $\hat{W}_{j,:}(k)$ is the estimated weight vector linked to the j th hidden layer neurons. From (39) and the matrix in (5), $A(k)$ in (18) can be further extended as

$$A(k) = 2c\Omega(k) \quad (40)$$

where $\Omega(k) \in R^m$ is defined as

$$\Omega(k) = \begin{bmatrix} \Omega_1(k), \dots, \Omega_{n_h}(k), 0, \dots & 0, \dots \\ 0, \dots & \ddots & \vdots \\ \vdots & & \\ 0, \dots & 0, \dots & \Omega_1(k), \dots, \Omega_{n_h}(k) \end{bmatrix} \bar{V}(k) \quad (41)$$

and $\Omega_j(k) = \mu_j(k)\hat{W}_{j,:}(k)\hat{D}^v(k)\Delta^v$, $j = 1, \dots, n_h$. By such expanding of $A(k)$, $c(k)$ is explicit as shown in (40). To reveal the effect of $c(k)$, we further expand (25) as

$$\hat{y}^{v+}(\bar{V}(k)) - \hat{y}^{v-}(\bar{V}(k)) \approx 2H(k)c\Delta^v + 2\beta^v(k+1)c\Omega(k). \quad (42)$$

Using (25) and (27), we have the presentation of the system disturbance in terms of the measurement noise $\delta^v(k)$ of the RRSPSA algorithm as

$$\begin{aligned} \varepsilon(k) = & - \left\{ (H(k)\Delta^v + \beta^v\Omega(k))((H(k)\Delta^v + \beta^v\Omega(k)))^T \right\}^{-1} \\ & \times (H(k)\Delta^v + \beta^v\Omega(k))\delta^v(k)(2c)^{-1}. \end{aligned} \quad (43)$$

Furthermore, according to the equivalent disturbance (31), we have

$$\begin{aligned} \tilde{e}^v(k) = & \tilde{H}(\bar{W}(k), \bar{W}^*, x(k))\bar{V}^* + \varepsilon(k) \\ = & \tilde{H}(\bar{W}(k), \bar{W}^*, x(k))\bar{V}^* - \left\{ (H(k)\Delta^v + \beta^v\Omega(k)) \right. \\ & \left. \times ((H(k)\Delta^v + \beta^v\Omega(k)))^T \right\}^{-1} \frac{(H(k)\Delta^v + \beta^v\Omega(k))\delta^v(k)}{2c}. \end{aligned} \quad (44)$$

The equivalent disturbance $\tilde{e}^v(k)$ is important since its maximum value $\tilde{e}_m^v(k)$ decides the range of dead zone for α^v in (37). Note also that the measurement noise $\delta^v(k)$ is defined the same way as in [11], which measures difference between true gradient and gradient approximation as derived in (26). Because the system disturbance $\varepsilon(k)$ is

decided only by external conditions, a relatively smaller gain parameter c will imply a smaller measurement noise $\delta^v(k)$ according to (43). This allows us to choose the dead zone range mainly according to the external system disturbance $\varepsilon(k)$. Furthermore, as shown in (37), α^v will be nonzero whenever the norm of $e(k)$ is bigger than a dead zone value which is related to $\tilde{e}_m^v(k)$. If the external disturbance is also small, we can choose a small dead zone range, which ensures that the adaptive learning rate α^v has the nonzero value for a prolong period of training time.

5 Hidden Layer Training and Stability Analysis

The algorithm for updating estimated weight vector $\bar{W}(k)$ of the hidden layer is

$$\bar{W}(k+1) = \bar{W}(k) - \alpha^w(k+1) \hat{g}^w(\bar{W}(k)) \quad (45)$$

where $\hat{g}^w(\bar{W}(k)) \in R^{p^w}$ is the normalized gradient approximation that uses simultaneous perturbation vectors $\Delta^w(k+1) \in R^{p^w}$ and $r^w(k+1) \in R^{p^w}$ to stimulate weight of the hidden layer. $\Delta^w(k+1)$ and $r^w(k+1)$ are, respectively, the last p^w components of perturbation vectors defined in (14) and (15).

$$\hat{g}^w(\bar{W}(k)) = - \frac{e^T(k) (B^+(k) - B^-(k) + \beta^w(k+1) \hat{D}^w(k))}{2c\rho^w(k+1)} r^w(k+1) \quad (46)$$

where

$$B^+(k) = H(\bar{W}(k) + c\Delta^w(k+1), x(k)) \bar{V}(k) \in R^m \quad (47)$$

$$B^-(k) = H(\bar{W}(k) - c\Delta^w(k+1), x(k)) \bar{V}(k) \in R^m \quad (48)$$

$$\begin{aligned} \hat{D}^w(k) = & \left(H \left(\bar{W}(k), \left[(B^+(k-1))^T, \dots, (B^+(k-l))^T, 0, \dots, 0 \right]^T \right) \right. \\ & \left. - H \left(\bar{W}(k), \left[(B^-(k-1))^T, \dots, (B^-(k-l))^T, 0, \dots, 0 \right]^T \right) \right) \bar{V}(k) \in R^m. \end{aligned} \quad (49)$$

For simplicity of notations, we ignore the time subscripts of $\Delta^w(k+1)$, $r^w(k+1)$, $\beta^w(k+1)$, $\rho^w(k+1)$ and $\alpha^w(k+1)$ in the following part of the paper.

Remark 4 The same as the output layer, our purpose is to find the desired hidden layer weight vector $\bar{W}^*$ of the gradient equation

$$\hat{g}(\bar{W}(k)) = \frac{\partial L(\bar{W}(k))}{\partial \bar{W}(k)} = 0.$$

The detailed mathematical deduction is given in the proof of Lemma 2. $B^+(k) - B^-(k)$ is the result of the perturbations of the feedforward signals and $\hat{D}^w(k)$ is the result of the perturbations of the recurrent signals in RNNs. The function of β^w is to automatically switch off recurrent contribution when the convergence and stability requirements are violated. The conditions for the key parameters appeared in (45) and (46) are shown in (64), (67), and (68).

Lemma 2 *The normalized gradient approximation in (46) is an equivalent presentation of the standard SPSA algorithm in equation (13).*

Proof Similar to the definitions of $\hat{g}^{v+}(\bar{V}(k))$ and $\hat{g}^{v-}(\bar{V}(k))$ in (21) and (23), we define two perturbation functions $\hat{g}^{w+}(\bar{V}(k))$ and $\hat{g}^{w-}(\bar{V}(k))$, i.e.,

$$\hat{g}^{w+}(\bar{W}(k)) = H(\bar{W}(k) + c\Delta^w, x(k))\bar{V}(k) + H(\bar{W}(k), (x(k) + \Delta x(k+1)))\bar{V}(k). \quad (50)$$

From the definition of $B^+(k)$ in (47),

$$\begin{aligned} & \hat{g}^{w+}(\bar{W}(k)) \\ &= H(\bar{W}(k) + c\Delta^w, x(k))\bar{V}(k) + H(\bar{W}(k), (x(k) + \Delta x(k+1)))\bar{V}(k) \\ &= B^+(k) + H\left(\bar{W}(k), \left([\hat{y}^T(k-1), \dots, \hat{y}^T(k-l), 0, \dots, 0]^T + \Delta x(k+1)\right)\right)\bar{V}(k) \\ &= B^+(k) + H\left(\bar{W}(k), \left[\bar{V}^T(k-1)H^T(\bar{W}(k-1) + c\Delta^w, x(k-1)), \dots \right.\right. \\ & \quad \left.\left.\bar{V}^T(k-l)H^T(\bar{W}(k-l) + c\Delta^w, x(k-l)), 0, \dots, 0\right]^T\right)\bar{V}(k) \\ &\approx B^+(k) + H\left(\bar{W}(k), \left[(B^+(k-1))^T, \dots, (B^+(k-l))^T, 0, \dots, 0\right]^T\right)\bar{V}(k). \end{aligned} \quad (51)$$

From the definition of $B^-(k)$ in (48), and similar to (51), we get

$$\hat{g}^{w-}(\bar{W}(k)) \approx B^-(k) + H\left(\bar{W}(k), \left[(B^-(k-1))^T, \dots, (B^-(k-l))^T, 0, \dots, 0\right]^T\right)\bar{V}(k). \quad (52)$$

Then, we have

$$\hat{g}^{w+}(\bar{W}(k)) - \hat{g}^{w-}(\bar{W}(k)) \approx B^+(k) - B^-(k) + \hat{D}^w(k). \quad (53)$$

$\hat{D}^w(k)$ in (53) is the result of recurrent training signal as shown in Fig. 1. To guarantee weight convergence and system stability during training, we add an adaptive recurrent hybrid parameter $\beta^w(k+1)$ to control the recurrent learning signal path. That is,

$$\hat{g}^{w+}(\bar{W}(k)) - \hat{g}^{w-}(\bar{W}(k)) \approx B^+(k) - B^-(k) + \beta^w \hat{D}^w(k). \quad (54)$$

According to (13), we can get

$$\begin{aligned}
 \hat{g}^w(\bar{W}(k)) &= \frac{L(\bar{W}(k) + c\Delta^w, \bar{V}(k)) - L(\bar{W}(k) - c\Delta^w, \bar{V}(k)) + \delta^w(k)}{2c\rho^w} r^w \\
 &= \frac{\|y(k) - \hat{y}^{w+}(k)\|^2 - \|y(k) - \hat{y}^{w-}(k)\|^2 + 2\delta^w(k)}{4c\rho^w} r^w \\
 &= \frac{(y(k) - \hat{y}^{w+}(k) + y(k) - \hat{y}^{w-}(k))^T (\hat{y}^{w-}(k) - \hat{y}^{w+}(k)) + 2\delta^w(k)}{4c\rho^w} r^w \\
 &= \left(2y(k) - \hat{y}^{w+}(k) - \hat{y}^{w-}(k) + \hat{y}^{w+}(k) + \hat{y}^{w-}(k) - 2\hat{y}(k) + 2\varepsilon(k) \right)^T \\
 &\quad \times \left((\hat{y}^{w-}(k) - \hat{y}^{w+}(k)) + 2\delta^w(k) \right) r^w (4c\rho^w)^{-1} \\
 &= (y(k) - \hat{y}(k) + \varepsilon(k))^T (\hat{y}^{w-}(k) - \hat{y}^{w+}(k)) r^w (2c\rho^w)^{-1} \\
 &= e^T(k) (\hat{y}^{w-}(k) - \hat{y}^{w+}(k)) r^w (2c\rho^w)^{-1} \\
 &\approx -e^T(k) (B^+(k) - B^-(k) + \beta^w \hat{D}^w(k)) r^w (2c\rho^w)^{-1}
 \end{aligned} \tag{55}$$

where the equivalent disturbance $\delta^w(k)$ is defined as

$$\begin{aligned}
 \delta^w(k) &= (\hat{y}^{w-}(\bar{W}(k)) - \hat{y}^{w+}(\bar{W}(k)))^T \varepsilon(k) + \frac{1}{2} (\hat{y}^{w-}(\bar{W}(k)) - \hat{y}^{w+}(\bar{W}(k)))^T \\
 &\quad \times (\hat{y}^{w+}(\bar{W}(k)) + \hat{y}^{w-}(\bar{W}(k)) - 2\hat{y}(k)).
 \end{aligned} \tag{56}$$

The fifth equality in (55) reveals the relationship between $\delta^w(k)$ and $\varepsilon(k)$ as in (56).

For stability analysis, the training error will be linked to the parameter estimating error of the hidden layer and the disturbance. According to mean value theorem and that activation function is a nondecreasing function, it can be readily shown that there exist unique positive mean values $\tilde{\mu}_j(k)$ such that

$$h(W_{j,:}^* x(k)) - h(\hat{W}_{j,:}(k) x(k)) = \tilde{\mu}_j(k) (W_{j,:}^* - \hat{W}_{j,:}(k)) x(k) = \tilde{\mu}_j(k) \tilde{\tilde{W}}_{j,:}(k) x(k) \tag{57}$$

where $\hat{W}_{j,:}(k)$, $W_{j,:}^*$, $\tilde{\tilde{W}}_{j,:}(k)$ are the estimated weight, ideal weight, and weight estimate error vectors linked to the j th hidden layer neurons, respectively. From (57) and the matrix in (5), it can be shown that

$$\tilde{H}(\bar{W}(k), \bar{W}^*, x(k)) \bar{V}(k) = (H(\bar{W}^*, x(k)) - H(k)) \bar{V}(k) = \tilde{\Omega}(k) \tilde{\tilde{W}}(k) \tag{58}$$

where $\tilde{\Omega}(k) \in R^{m \times p^w}$ is defined as

$$\begin{aligned}
 \tilde{\Omega}(k) &= \begin{bmatrix} \tilde{\mu}_1(k) \bar{V}_1(k) x^T(k) & \cdots & \tilde{\mu}_{n_h}(k) \bar{V}_{n_h}(k) x^T(k) \\ \tilde{\mu}_1(k) \bar{V}_{n_h+1}(k) x^T(k) & \cdots & \tilde{\mu}_{n_h}(k) \bar{V}_{2n_h}(k) x^T(k) \\ \vdots & & \vdots \\ \tilde{\mu}_1(k) \bar{V}_{(m-1)n_h+1}(k) x^T(k) & \cdots & \tilde{\mu}_{n_h}(k) \bar{V}_{p^w}(k) x^T(k) \end{bmatrix} \\
 &= [\tilde{\mu}_1(k) \hat{V}_{1,:}^T(k) x^T(k) \cdots \tilde{\mu}_{n_h}(k) \hat{V}_{n_h,:}^T(k) x^T(k)]
 \end{aligned} \tag{59}$$

and $\hat{V}_{j,:}^T(k)$ is inverse of j th row of the estimated weight matrix $\hat{V}(k)$ of output layer.

From (6), the maximum value of the derivative of $h_j(k)$, $\dot{h}_j(k) = (\partial h(\hat{W}_{j,:}(k)x(k)))/(\partial(\hat{W}_{j,:}(k)x(k)))$ is λ , and hence $0 \leq \tilde{\mu}_j(k) \leq \lambda$ ($1 \leq i \leq n_h$).

We can rewrite the estimation error as

$$\begin{aligned} e(k) &= y(k) - \hat{y}(k) + \varepsilon(k) \\ &= H(\bar{W}^*, x(k))\bar{V}^* - H(\bar{W}^*, x(k))\bar{V}(k) + H(\bar{W}^*, x(k))\bar{V} - H(k)\bar{V}(k) + \varepsilon(k) \\ &= \tilde{H}(\bar{W}(k), \bar{W}^*, x(k))\bar{V}(k) + H(\bar{W}^*, x(k))\tilde{\bar{V}}(k) + \varepsilon(k) \\ &= -\phi^w(k) + \tilde{e}^w(k) \end{aligned} \quad (60)$$

where

$$\phi^w(k) = -\tilde{H}(\bar{W}(k), \bar{W}^*, x(k))\bar{V}(k) = -\tilde{\Omega}(k)\tilde{\bar{W}}(k) \quad (61)$$

and the bounded equivalent disturbance $\tilde{e}^w(k)$ of the hidden layer is defined as

$$\tilde{e}^w(k) = H(\bar{W}^*, x(k))\tilde{\bar{V}}(k) + \varepsilon(k). \quad (62)$$

Note that $\tilde{e}^w(k)$ is a bounded signal because $\tilde{\bar{V}}(k)$ has been proven to be bounded in Section III.

According to the definition of $B^+(k)$ and $B^-(k)$ defined in (47) and (48), we can get

$$(B^+(k) - B^-(k))(2c)^{-1} = \hat{\Omega}(k)\Delta^w \quad (63)$$

where $\hat{\Omega}(k) \in R^{m \times p^w}$ is similar to $\tilde{\Omega}(k)$ in (59) except with different mean values.

Theorem 2 *If the hidden layer of the RNN is trained by the adaptive RRSPSA algorithm (45), weight $\bar{W}(k)$ is guaranteed to be convergent in the sense of Lyapunov function*

$$\left\| \tilde{\bar{W}}(k+1) \right\|^2 - \left\| \tilde{\bar{W}}(k) \right\|^2 \leq 0, \forall k$$

with $\tilde{\bar{W}}(k) = \bar{W}^* - \bar{W}(k)$ and the training error will be L_2 -stable in the sense of $\tilde{e}^w(k) \in L_2$ and $\phi^w(k) \in L_2$.

Proof See Section 8.

The three adaptive learning rates appeared in (45) and (46) are as follows.

1. The normalization factor is defined as

$$\rho^w(k+1) = \tau \rho^w(k) + \max \left\{ 1 + \alpha^w M(k) \|r^w\|^2 \Upsilon^{-1}(k), \bar{\rho} \right\} \quad (64)$$

where

$$\begin{aligned} M(k) &= \left\| \frac{B^+(k) - B^-(k) + \beta^w \hat{D}^w(k)}{2c} \right\|^2 \\ \Upsilon(k) &= \frac{\lambda_{\min}}{\lambda} p^w + \frac{1}{2\lambda_c} \beta^w (r^w)^T (\eta I + \bar{\Omega}^T(k) \bar{\Omega}(k))^{-1} \bar{\Omega}^T(k) \hat{D}^w(k) \end{aligned}$$

and $0 < \alpha^w \leq 1$ being a positive constant, λ being the maximum value of the derivative $\dot{h}_j(k)$ of the activation function in (6), and $\lambda_{\min} \neq 0$ being the minimum nonzero value of the mean value $\bar{\mu}_j(k)$ defined as

$$0 \leq \bar{\mu}_j(k) \leq \lambda \quad (1 \leq i \leq n_h) \quad (65)$$

and

$$\begin{aligned} \bar{\Omega}(k) &= \begin{bmatrix} \bar{V}_1(k)x^T(k) & \cdots & \bar{V}_{n_h}(k)x^T(k) \\ \bar{V}_{n_h+1}(k)x^T(k) & \cdots & \bar{V}_{2n_h}(k)x^T(k) \\ \vdots & & \vdots \\ \bar{V}_{(m-1)n_h+1}(k)x^T(k) & \cdots & \bar{V}_{p^v}(k)x^T(k) \end{bmatrix} \\ &= [\hat{V}_{1,:}^T(k)x^T(k) \cdots \hat{V}_{n_h,:}^T(k)x^T(k)] \in \mathbb{R}^{m \times p^w} \end{aligned} \quad (66)$$

where $\bar{V}_j(k)$ is the j th element of $\bar{V}(k)$, $1 \leq j \leq p^v$.

2. The adaptive dead zone learning rate of hidden layer is defined as

$$\begin{cases} \alpha^w(k+1) = \alpha^w, & \text{if } \|e(k)\| \geq \Gamma(k) \\ \alpha^w(k+1) = 0, & \text{if } \|e(k)\| < \Gamma(k) \end{cases} \quad (67)$$

with $\Gamma = \bar{e}_m^w(k) / \sqrt{\left(1 - \frac{\alpha^w \|r^w\|^2}{\rho^w T(k)} M(k)\right)}$ and $\bar{e}_m^w(k)$ being the maximum value of $\bar{e}^w(k)$ defined in (62).

3. The recurrent hybrid adaptive parameter of hidden layer β^w is defined as

$$\begin{cases} \beta^w(k+1) = 1, & \text{if } (r^w)^T \chi(k) \geq 0 \\ \beta^w(k+1) = 0, & \text{if } (r^w)^T \chi(k) < 0 \end{cases} \quad (68)$$

with $\chi(k) = (\eta I + \bar{\Omega}^T(k)\bar{\Omega}(k))^{-1} \bar{\Omega}^T(k)\hat{D}^w(k)$. $\beta^w(k+1)$ can automatically cut off the recurrent training signal as in the lower right corner of Fig.1 whenever weight convergence and stability conditions of the RNN are violated as shown in Theorem 2 of section IV.

Remark 5 Summary of the RRSPSA algorithm for RNNs.

1. Set a maximum training step as the stop criterion;
2. Initialize the weights for the RNN;
3. Form new input $x(k)$ as defined in equation (3);
4. Perform forward computation and calculate the output $\hat{y}(k)$ and error $e(k)$ with the input and current estimated weights;
5. Calculate $\rho^v(k+1)$, $\alpha^v(k+1)$, and $\beta^v(k+1)$ using (34), (37), and (38), respectively;
6. Update the output layer weights $\bar{V}(k+1)$ via the learning law (16) for the next iteration;
7. Calculate $\rho^w(k+1)$, $\alpha^w(k+1)$, and $\beta^w(k+1)$ using (64), (67), and (68), respectively;
8. Update the hidden layer weights $\bar{W}(k+1)$ via the learning law (45) for the next iteration;
9. Go back to step 3 to continue the iteration until the stop criterion is reached.

6 Examples

6.1 Time series prediction

Sunspots are dark blotches on the sun which are caused by magnetic storms on the surface of the sun. The underlying mechanism for sunspot appearances is not exactly known. The sunspots data is a classical example of a combination of periodic and chaotic phenomena which has served as a benchmark in the statistics literature of time series and much work has been done in trying to analyze the sunspot data using neural networks [21, 22]. One set of the high quality America Sunspot Numbers, monthly sunspot numbers from December 1949 to October 2009 in the database, is used as an example to minimize unnecessary human measurement errors. The total America sunspot number in the monthly database is 778 points. The output sequence is scaled by dividing by 100, to get into a range suitable for the network. The hidden neuron number of the RNN is ten. The selected learning rates for BP and RTRL are 0.09 and 0.01, respectively. Perturbation step size for RRSPSA is $c = 0.01$. Fig. 2 shows outputs of the plant $y^*(k)$ and estimated outputs $y(k)$ of the RNN trained by the three different methods. Table 1 gives comparison results of the three methods in terms of MSE and STD.

Table 1 Comparison of Mean Squared Error and Standard Deviation of the Training Error

Squared Error	RRSPSA	RTRL	BP
MSE	$6.01e^{-4}$	$1.30e^{-3}$	$4.30e^{-3}$
STD	$2.45e^{-2}$	$3.61e^{-2}$	$6.57e^{-2}$

6.2 Michaelis-Menten equation

Consider a continue time Michaelis-Menten equation [23]

$$\dot{y}(t) = 0.45 \frac{y(t)}{2.27 + y(t)} - y(t) + u(t) \quad (69)$$

Given an unit impulse as input $u(t)$, a fourth-order Runge-Kutta algorithm is used to simulate this model with integral step size $\Delta t = 0.01$, and 1000 equi-spaced samples are obtained from input and output with a sampling interval of $T = 0.02$ time units. The measured noise is white Gaussian noise with mean 0 and variance 0.05. An RNN is utilized to emulate the dynamics of the given system and approximate the system output in (69). Gradient function of this model has many local minima and the global minimum has a very narrow domain of attraction [23]. We compare performance of the RRSPSA to that of both BP and RTRL training algorithms. The hidden neuron number of RNNs in this example is selected as ten. The fixed learning rates for BP and RTRL are 0.03 and 0.01, respectively. The perturbation size $c = 0.01$ is selected for RRSPSA. Fig. 3 shows outputs of the plant $y^*(k)$ and estimated outputs $y(k)$ of

Table 2 Comparison of the Mean Squared Error and Standard Deviation of the Error

	RRSPSA	RTRL	BP
MSE	$4.2e^{-3}$	$1.64e^{-2}$	$2.98e^{-2}$
STD	$6.46e^{-2}$	$1.28e^{-1}$	$1.72e^{-1}$

the RNN trained by the three different methods. Table 2 shows the comparison results of the three methods in terms of MSE and STD.

Simulation results show that the BP training algorithm performs worst as compared to other methods. The reason is that the second partial derivatives similar to that of as in (17) are not considered in the BP training. For RTRL, the slower convergence compared with RRSPSA is mainly caused by the small fixed learning rate and absent of the recurrent hybrid adaptive parameter for guaranteed weight convergence in the presence of noise. RRSPSA utilizes the specifically designed three adaptive parameters to maximize training speed for recurrent training signal while exhibiting certain weight convergence properties with only two objective function measurements. Combination of the three adaptive parameters makes RRSPSA converge fast with the maximum effective learning rate and guaranteed weight convergence.

7 Conclusion

We proposed a robust recurrent simultaneous perturbation stochastic approximation (RRSPSA) algorithm under the framework of deterministic system with guaranteed weight convergence. Compared with FNNs, RNNs are dynamic systems and contain recurrent connections in their structure. Considering the time dependence of the signals of RNNs, not only the weights at the current time steps are to be perturbed, but also those at the previous time steps, which makes the learning easy to be diverge and the convergence analysis complicated to obtain. The key characteristic of RRSPSA training algorithm is to use a recurrent hybrid adaptive parameter together with adaptive learning rate and dead zone concept. The recurrent hybrid adaptive parameter can automatically switch off recurrent training signal whenever the weight convergence and stability conditions are violated (see Fig. 1 and explanation in Section III and IV). There are several advantages to train RNNs based on RRSPSA algorithm. First, RRSPSA is relatively easy to implement as compared to other RNN training methods such as BPTT due to its RTRL type of training and simplicity. Second, RRSPSA provides excellent convergence property based on simultaneous perturbation of weight, which is similar to the standard SPSA. Finally, RRSPSA has capability to improve RNN training speed with guaranteed weight convergence over the standard RTRL algorithm by using adaptive learning method. Robust stability and weight convergence proofs of RRSPSA algorithm are provided based on Lyapunov function.

8 Proof of Theorem 1

Proof According to the training algorithm in (16), we can get

$$\tilde{V}(k+1) = \tilde{V}(k) + \alpha^v \hat{g}^v(\tilde{V}(k)) \quad (70)$$

where $\hat{g}^v(\bar{V}(k))$ is defined in (26), then we have

$$\begin{aligned}
& \left\| \tilde{V}(k+1) \right\|^2 - \left\| \tilde{V}(k) \right\|^2 \\
&= 2\alpha^v \hat{g}^v(\bar{V}(k))^T \tilde{V}(k) + \|\alpha^v \hat{g}^v(\bar{V}(k))\|^2 \\
&\approx -\alpha^v \frac{e^T(k) (2H(k) c\Delta^v + \beta^v A(k))}{c\rho^v} (r^v)^T \tilde{V}(k) + \|\alpha^v \hat{g}^v(\bar{V}(k))\|^2 \\
&= -\alpha^v \frac{2c \left\{ e^T(k) H(k) \Delta^v (r^v)^T \tilde{V}(k) \right\} + \left\{ \beta^v e^T(k) A(k) (r^v)^T \tilde{V}(k) \right\}}{c\rho^v} + \|\alpha^v \hat{g}^v(\bar{V}(k))\|^2 \\
&= -2\alpha^v \left\{ e^T(k) H(k) \tilde{V}(k) \right\} \left\{ \tilde{V}^T(k) \left(\tilde{V}(k) \tilde{V}^T(k) \right)^{-1} \Delta^v (r^v)^T \tilde{V}(k) \right\} (\rho^v)^{-1} \\
&\quad - \alpha^v \left\{ \beta^v e^T(k) H(k) \tilde{V}(k) \left(H(k) \tilde{V}(k) \right)^T \left(H(k) \tilde{V}(k) \left(H(k) \tilde{V}(k) \right)^T \right)^{-1} \right. \\
&\quad \times A(k) (r^v)^T \left(H^T(k) H(k) \right)^{-1} H^T(k) H(k) \tilde{V}(k) \left. \right\} (c\rho^v)^{-1} + \|\alpha^v \hat{g}^v(\bar{V}(k))\|^2 \\
&= 2\alpha^v p^v \left\{ e^T(k) \phi^v(k) \right\} (\rho^v)^{-1} \\
&\quad + \alpha^v \beta^v \left\{ e^T(k) \phi^v(k) \right\} \text{trace} \left\{ \left(H(k) \tilde{V}(k) \right)^T \left(H(k) \tilde{V}(k) \left(H(k) \tilde{V}(k) \right)^T \right)^{-1} A(k) (r^v)^T \right. \\
&\quad \times \left. \left(H^T(k) H(k) \right)^{-1} H^T(k) H(k) \tilde{V}(k) \right\} (c\rho^v)^{-1} + \|\alpha^v \hat{g}^v(\bar{V}(k))\|^2 \\
&\approx 2\alpha^v p^v \left\{ e^T(k) \phi^v(k) \right\} (\rho^v)^{-1} + \alpha^v \beta^v \left\{ e^T(k) \phi^v(k) \right\} \\
&\quad \times \text{trace} \left\{ A(k) (r^v)^T \left(\eta I + H^T(k) H(k) \right)^{-1} H^T(k) H(k) \tilde{V}(k) \left(H(k) \tilde{V}(k) \right)^T \right. \\
&\quad \times \left. \left(H(k) \tilde{V}(k) \left(H(k) \tilde{V}(k) \right)^T \right)^{-1} \right\} (c\rho^v)^{-1} + \|\alpha^v \hat{g}^v(\bar{V}(k))\|^2 \\
&= 2\alpha^v p^v \left\{ e^T(k) \phi^v(k) \right\} (\rho^v)^{-1} + \alpha^v \left\{ e^T(k) \phi^v(k) \right\} \beta^v (r^v)^T \\
&\quad \times \left(\eta I + H^T(k) H(k) \right)^{-1} H^T(k) A(k) (c\rho^v)^{-1} + \|\alpha^v \hat{g}^v(\bar{V}(k))\|^2
\end{aligned} \tag{71}$$

Note that $H(k) \tilde{V}(k)$ in the fourth equality is replaced with $\phi^v(k)$ as defined in (32). A small perturbation parameter is added to approximate $[H^T(k) H(k)]^{-1}$ and guarantee a full rank of the inverse matrix of $[\eta I + H^T(k) H(k)]^{-1}$, which is similar to a general

nonlinear iteration algorithm [9].

$$\begin{aligned}
& \left\| \tilde{V}(k+1) \right\|^2 - \left\| \tilde{V}(k) \right\|^2 \\
& \leq 2\alpha^v p^v \left\{ e^T(k) (\tilde{e}^v(k) - e(k)) \right\} (\rho^v)^{-1} + \alpha^v \left\{ e^T(k) (\tilde{e}^v(k) - e(k)) \right\} \beta^v (r^v)^T \\
& \quad \times (\eta I + H^T(k)H(k))^{-1} H^T(k)A(k)(c\rho^v)^{-1} \\
& \quad + \|e(k)\|^2 \left\| \alpha^v (2H(k)c\Delta^v + \beta^v A(k)) \right\|^2 \|r^v\|^2 (2c\rho^v)^2 \\
& \leq \alpha^v p^v (\rho^v)^{-1} \left\{ \|\tilde{e}^v(k)\|^2 - \|e(k)\|^2 \right\} + \alpha^v \left\{ \|\tilde{e}^v(k)\|^2 - \|e(k)\|^2 \right\} \beta^v (r^v)^T \\
& \quad \times (\eta I + H^T(k)H(k))^{-1} H^T(k)A(k)(2c\rho^v)^{-1} \\
& \quad + \|e(k)\|^2 \left\| \alpha^v (2H(k)c\Delta^v + \beta^v A(k)) \right\|^2 \frac{\|r^v\|^2}{(2c\rho^v)^2} \\
& = \alpha^v (\rho^v)^{-1} \{p^v + 0.5\beta^v Y(k)\} \|\tilde{e}^v(k)\|^2 \\
& \quad - \alpha^v (\rho^v)^{-1} \left\{ p^v + 0.5\beta^v Y(k) - \alpha^v (\rho^v)^{-1} Z(k) \right\} \|e(k)\|^2 \\
& \leq \alpha^v (\rho^v)^{-1} \{p^v + 0.5\beta^v Y(k)\} \left\{ \|\tilde{e}^v(k)\|^2 - \left(1 - \frac{\alpha^v \rho(k+1)^{-1} Z(k)}{p^v + 0.5\beta^v Y(k)} \right) \|e(k)\|^2 \right\} \\
& \tag{72}
\end{aligned}$$

where $Y(k)$ and $Z(k)$ are defined in (35) and (36), respectively. Note also that the second inequality in (72) is from the fact that $2e^T(k)\tilde{e}^v(k) \leq \|\tilde{e}^v(k)\|^2 + \|e(k)\|^2$. p^v is the dimension of the output layer weight vector which is positive. α^v is a fixed positive scalar. $\beta^v Y(k) \geq 0$ due to the definition of recurrent adaptive learning rate $\beta^v(k+1)$ in (38) and $\rho^v > \frac{\alpha^v Z(k)}{p^v + 0.5\beta^v Y(k)}$ due to definition of the normalization factor in (34). The adaptive learning rate $\alpha^v(k+1)$ in (37) implies that recurrent training of RRSPSA will be α^v only if the output error of RNN is bigger than a predefined value otherwise it will be zero, which guarantees $\left\| \tilde{V}(k+1) \right\|^2 - \left\| \tilde{V}(k) \right\|^2$ is bounded below zero and is not increasing. The limit of $\left\| \tilde{V}(k+1) \right\|^2 - \left\| \tilde{V}(k) \right\|^2$ therefore exists and $\left\{ \|\tilde{e}^v(k)\|^2 - \left(1 - \frac{\alpha^v \rho(k+1)^{-1} Z(k)}{p^v + 0.5\beta^v Y(k)} \right) \|e(k)\|^2 \right\}$ will go to zero. Then, we finally have

$$\left\| \tilde{V}(k+1) \right\|^2 - \left\| \tilde{V}(k) \right\|^2 \leq 0.$$

9 Proof of Theorem 2

Proof Consider that SPSA is an approximation of the gradient algorithm [11], and using the property of local minimum points of the gradient

$$-(\partial(e^T(k)e(k))) / (\partial(\bar{W}(k))^T) \tilde{W}(k) \geq 0$$

[5], we have

$$\begin{aligned}
0 &\leq -\frac{\partial (e^T(k)e(k))}{\partial (\bar{W}(k))^T} \tilde{W}(k) \\
&= \sum_{i=1}^{n_h} \left\{ \dot{h}_i(k) e^T(k) \hat{V}_{i,:}^T(k) x^T(k) \tilde{W}_{i,:}(k) \right\} \\
&= \sum_{i=1}^{n_h} \left\{ \frac{\dot{h}_i(k)}{\tilde{\mu}_i(k)} \left(\tilde{\mu}_i(k) e^T(k) \hat{V}_{i,:}^T(k) x^T(k) \tilde{W}_{i,:}(k) \right) \right\} \\
&\leq \frac{\lambda}{\lambda_{\min}} \sum_{i=1}^{n_h} \left\{ \tilde{\mu}_i(k) e^T(k) \hat{V}_{i,:}^T(k) x^T(k) \tilde{W}_{i,:}(k) \right\} \\
&= \frac{\lambda}{\lambda_{\min}} e^T(k) \left(\tilde{\Omega}(k) \tilde{W}(k) \right) \\
&= -\frac{\lambda}{\lambda_{\min}} e^T(k) \phi^w(k)
\end{aligned} \tag{73}$$

where λ is the maximum value of derivative $\dot{h}_j(k)$ of activation function in (6), and $\lambda_{\min} \neq 0$ is the minimum nonzero value of mean value $\tilde{\mu}_j(k)$ defined in (65). (Note that the inequality is always true if $\lambda_{\min} = 0$, which implies $\dot{h}_j(k) = \tilde{\mu}_j(k) = 0$.) According to the training algorithm for hidden layer in (45), we get

$$\tilde{W}(k+1) = \tilde{W}(k) + \alpha^w \hat{g}^w(\bar{W}(k)) \tag{74}$$

where $\hat{g}^w(\bar{W}(k))$ is defined in (55). Using (54), (74) and (73), by the definition of $\tilde{\Omega}(k)$ in (66), we get

$$\begin{aligned}
&\left\| \tilde{W}(k+1) \right\|^2 - \left\| \tilde{W}(k) \right\|^2 \\
&= 2\alpha^w \hat{g}^w(\bar{W}(k)) \tilde{W}(k) + \left\| \alpha^w \hat{g}^w(\bar{W}(k)) \right\|^2 \\
&\approx -\alpha^w e^T(k) \frac{(B^+(k) - B^-(k)) + \beta^w \hat{D}^w(k)}{c\Delta^w} (r^w)^T \tilde{W}(k) + \left\| \alpha^w \hat{g}^w(\bar{W}(k)) \right\|^2 \\
&= \sum_{i=1}^{n_h} \left\{ -\mu_i(k) e^T(k) \hat{V}_{i,:}^T(k) x^T(k) \tilde{W}_{i,:}(k) \right\} \left\{ \tilde{W}^T(k) \left(\tilde{W}(k) \tilde{W}^T(k) \right)^{-1} \Delta^w (r^w)^T \tilde{W}(k) \right\} \\
&\quad \times \frac{2\alpha^w}{\Delta^w} - \alpha^w e^T(k) \left(\tilde{\Omega}(k) \tilde{W}(k) \right) \left(\tilde{\Omega}(k) \tilde{W}(k) \right)^T \left[\left(\tilde{\Omega}(k) \tilde{W}(k) \right) \left(\tilde{\Omega}(k) \tilde{W}(k) \right)^T \right]^{-1} \\
&\quad \times \hat{D}^w(k) (r^w)^T \tilde{W}(k) \beta^w (c\Delta^w)^{-1} + \left\| \alpha^w \hat{g}^w(\bar{W}(k)) \right\|^2 \\
&= \sum_{i=1}^{n_h} \left\{ \frac{\mu_i(k)}{\tilde{\mu}_i(k)} \left(-\tilde{\mu}_i(k) e^T(k) \hat{V}_{i,:}^T(k) x^T(k) \tilde{W}_{i,:}(k) \right) \right\} 2p^w \alpha^w (\Delta^w)^{-1} \\
&\quad + \sum_{i=1}^{n_h} \left\{ \frac{1}{\tilde{\mu}_i(k)} \left(-\tilde{\mu}_i(k) e^T(k) \hat{V}_{i,:}^T(k) x^T(k) \tilde{W}_{i,:}(k) \right) \right\} \alpha^w (c\Delta^w)^{-1}
\end{aligned}$$

$$\begin{aligned}
& \times \beta^w (r^w)^T \left(\bar{\Omega}^T(k) \bar{\Omega}(k) \right)^{-1} \bar{\Omega}^T(k) \bar{\Omega}(k) \tilde{W}(k) \left(\bar{\Omega}(k) \tilde{W}(k) \right)^T \\
& \times \left[\left(\bar{\Omega}(k) \tilde{W}(k) \right) \left(\bar{\Omega}(k) \tilde{W}(k) \right)^T \right]^{-1} \hat{D}^w(k) + \|\alpha^w \hat{g}^w(\bar{W}(k))\|^2 \\
& \approx \sum_{i=1}^{n_h} \left\{ \frac{\mu_i(k)}{\tilde{\mu}_i(k)} \left(-\tilde{\mu}_i(k) e^T(k) \hat{V}_{i,:}^T(k) x^T(k) \tilde{W}_{i,:}(k) \right) \right\} 2p^w \alpha^w (\Delta^w)^{-1} \\
& + \sum_{i=1}^{n_h} \left\{ \frac{1}{\tilde{\mu}_i(k)} \left(-\tilde{\mu}_i(k) e^T(k) \hat{V}_{i,:}^T(k) x^T(k) \tilde{W}_{i,:}(k) \right) \right\} \alpha^w (c\Delta^w)^{-1} \\
& \times \beta^w (r^w)^T \left(\eta I + \bar{\Omega}^T(k) \bar{\Omega}(k) \right)^{-1} \bar{\Omega}^T(k) \hat{D}^w(k) + \|\alpha^w \hat{g}^w(\bar{W}(k))\|^2 \\
& \leq 2\alpha^w \frac{\lambda_{min}}{\lambda} \frac{p^w}{\Delta^w} e^T(k) \phi^w(k) \\
& + \alpha^w \frac{1}{\lambda} \frac{\beta^w}{c\Delta^w} e^T(k) \phi^w(k) (r^w)^T \left(\eta I + \bar{\Omega}^T(k) \bar{\Omega}(k) \right)^{-1} \bar{\Omega}^T(k) \hat{D}^w(k) \\
& + \|e(k)\|^2 \left\| \frac{(B^+(k) - B^-(k) + \beta^w \hat{D}^w(k))}{2c} \right\|^2 \|r^w\|^2 \left(\frac{\alpha^w}{\Delta^w} \right)^2.
\end{aligned} \tag{75}$$

Similar to the output layer case, $\phi^w(k)$ is replaced with $(\tilde{e}^w(k) - e(k))$, we have the following Lyapunov function for convergence analysis:

$$\begin{aligned}
& \left\| \tilde{W}(k+1) \right\|^2 - \left\| \tilde{W}(k) \right\|^2 \\
& \leq \frac{\alpha^w}{\Delta^w} \left[\frac{\lambda_{min}}{\lambda} p^w + \frac{\beta^w}{2\lambda c} (r^w)^T \left(\eta I + \bar{\Omega}^T(k) \bar{\Omega}(k) \right)^{-1} \bar{\Omega}^T(k) \hat{D}^w(k) \right] \\
& \times \left\{ \left\| \tilde{e}^w(k) \right\|^2 - \left(1 - \frac{\alpha^w}{\Delta^w} \left\| \frac{B^+(k) - B^-(k) + \beta^w \hat{D}^w(k)}{2c} \right\|^2 \right) \|r^w\|^2 \right. \\
& \times \left. \left[\frac{\lambda_{min}}{\lambda} p^w + \frac{\beta^w}{2\lambda c} (r^w)^T \left(\eta I + \bar{\Omega}^T(k) \bar{\Omega}(k) \right)^{-1} \bar{\Omega}^T(k) \hat{D}^w(k) \right] \right\}^{-1} \|e(k)\|^2 \Big\} \\
& \tag{76}
\end{aligned}$$

The three adaptive parameters defined in (64), (67) and (68) guarantee the negativity of Lyapunov function. That is

$$\left\| \tilde{W}(k+1) \right\|^2 - \left\| \tilde{W}(k) \right\|^2 \leq 0, \forall k$$

References

1. Hopfield, J. J., Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the national academy of sciences*, 79, 2554-2558 (1982)
2. Talebi H. A., A recurrent neural-network-based sensor and actuator fault detection and isolation for nonlinear systems with application to the satellite's attitude control subsystem, *IEEE Transactions on Neural Networks*, 20, 45-60 (2009)

3. Hou, Z.G. and Gupta, M.M. and Nikiforuk, P.N. and Tan, M. and Cheng, L., A recurrent neural network for hierarchical control of interconnected dynamic systems, *IEEE Transactions on Neural Networks*, 18, 466-481 (2007)
4. Al Seyab, R. K. and Cao, Y., Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation, *Journal of Process Control*, 18, 568-581 (2008)
5. Song Q., Xiao J., Soh Y. C., Robust backpropagation training algorithm for multilayered neural tracking controller, *IEEE Transactions on Neural Networks*, 10, 1133-1141 (1999)
6. Song Q., Wu Y., Soh Y. C., Robust adaptive gradient-descent training algorithm for recurrent neural networks in discrete time domain, *IEEE Transactions on Neural Networks*, 19, 1841-1853 (2008)
7. Song, Q. and Spall, J.C. and Soh, Y.C. and Ni, J., Robust neural network tracking controller using simultaneous perturbation stochastic approximation, *IEEE Transactions on Neural Networks*, 19, 817-835 (2008)
8. Song, Q., On the weight convergence of Elman networks, *IEEE Transactions on Neural Networks*, 21, 463-480 (2008)
9. Haykin S., *Neural networks: a comprehensive foundation*. Printice Hall, New Jersey (1999)
10. Mandic, D.P. and Chambers, J.A., *Recurrent neural networks for prediction: Learning algorithms, architectures and stability*. Wiley, New York (2001)
11. Spall, J. C., Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, *IEEE Transactions on Automatic Control*, 37, 332-341 (1992)
12. Maeda, Y. and Wakamura, M., Simultaneous perturbation learning rule for recurrent neural networks and its FPGA implementation, *IEEE Transactions on Neural Networks*, 16, 1664-1672 (2005)
13. Trunov, A. B. and Polycarpou, M. M., Automated fault diagnosis in nonlinear multivariable systems using a learning methodology, *IEEE Transactions on Neural Networks*, 11, 91-101 (2000)
14. Spall, J. C. and Cristion, J. A., Model-free control of nonlinear stochastic systems with discrete-time measurements, *IEEE Transactions on Automatic Control*, 43, 1198-1210 (1998)
15. Spall, J. C. and Cristion, J. A., A neural network controller for systems with unmodeled dynamics with applications to wastewater treatment, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 27, 369-375 (1997)
16. Maeda, Y. and De Figueiredo, R. J. P., Learning rules for neuro-controller via simultaneous perturbation, *IEEE Transactions on Neural Networks*, 8, 1119-1130 (1997)
17. Werbos P. J., Generalization of backpropagation with application to a recurrent gas market model *Neural Networks*, 1, 339-356 (1988)
18. Rumelhart D., Hinton, G., Williams R., Learning internal representations by error backpropagation *Parallel distributed processing*, 1, 318-362 (1986)
19. Williams R. J., Zipser D., A learning algorithm for continually running fully recurrent neural networks, *Neural computation*, 1, 270-280 (1989)
20. Williams R. J., Zipser D., Gradient-based learning algorithms for recurrent networks and their computational complexity, *Backpropagation: Theory, architectures, and applications*, 2, 433-501 (1995)
21. Lin T., Giles C., Horne B., Kung S., A delay damage model selection algorithm for narx neural networks, *IEEE Transactions on Signal Processing*, 45, 2719-2730 (1997)
22. Park Y., Murray T., Chen C., Predicting sun spots using a layered perceptron neural network, *IEEE Transactions on Neural Networks*, 7, 501-505 (2002)
23. Ljung L., *Perspectives on system identification*, *Annual Reviews in Control*, (2010)

Fig. 2 Output $y(k)$ and reference signal $y^*(k)$ and error $e(k)$ using BP and RTRL and RRPSA training algorithms for time series prediction.

Fig. 3 Output $y(k)$ and reference signal $y^*(k)$ and error $e(k)$ using BP and RTRL and RRPSA training algorithms for simulation of Michaelis-Menten equation.