Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 69 (2006) 2097–2111
NEUROCOMPUTING
www.elsevier.com/locate/neucom

# Dynamic pruning algorithm for multilayer perceptron based neural control systems

Jie Ni, Qing Song*

*School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798, Singapore*

Received 13 February 2004; received in revised form 26 May 2005; accepted 19 September 2005
Available online 15 February 2006
Communicated by T. Heskes

**Abstract**

Generalization ability of neural networks is very important and a rule of thumb for good generalization in neural systems is that the smallest system should be used to fit the training data. Unfortunately, it is normally difficult to determine the optimal size of networks, particularly, in the sequential training applications such as online control. In this paper, an online training algorithm with a dynamic pruning procedure is proposed for the online tuning and pruning the neural tracking control system. The conic sector theory is introduced in the design of this robust neural control system, which aims at providing guaranteed boundedness for both the input–output signals and the weights of the neural network. The proposed algorithm is applied to a multilayer perceptron with adjustable weights and a complete convergence proof is provided. The neural control system guarantees the closed-loop stability of the estimation, and in turn, a good tracking performance. The performance improvement of the proposed system over existing systems can be qualified in terms of better generalization ability, preventing weight shifts, fast convergence and robustness against system disturbance.
© 2006 Elsevier B.V. All rights reserved.

*Keywords:* Dynamic pruning; Conic sector; Multilayer perceptron; Neural network; Robust control

## 1. Introduction

Recently, extensive research and significant progress have been made in the area of robust discrete time neural controller designed for nonlinear system [25,14,5,30,7,10,21,2,15,18,19,23,28,31]. However, when a neural system is used to handle unlimited examples, including training date and testing data, an important issue is how well it generalizes to patterns of the testing data, which is known as generalization ability [1,12]. For large discrete time domain sequential signals such as online control applications, it is usually impossible to cover every sample data even with proper training [9,11]. One would like the system to generalize from training samples to the underlying function and give reasonable answers to novel inputs with the existing training data. A rule of thumb for obtaining good generalization is to use the smallest system that fits the data. When a network has too many free parameters (i.e. weights and/or units), it may end up by just memorizing the training patterns. Both theoretical [1] and practical results [6,12] show that networks with minimal free parameters exhibit better generalization performance, which can be illustrated by recalling the analogy between neural network learning and curve fitting. Moreover, knowledge embedded in smaller trained networks is presumably easier to interpret and thus the extraction of simple rules can hopefully be facilitated. Lastly, from an implementation standpoint, small networks only require limited resources in any physical computational environment.

In this paper, a dynamic training and pruning algorithm for a generic neural control systems is proposed in which the impact of pruning is taken as a dynamic term and a general stability proof for the neural control system is derived. The plant under consideration is a nonlinear dynamic system and neural network is applied in the system to estimate the nonlinear function in the closed-loop. Conic sector theory [17,20] is introduced to design the robust control system, which aims to provide guaranteed

*Corresponding author.
*E-mail address:* eqsong@ntu.edu.sg (Q. Song).

boundedness for both the input–output signals and the weights of the neural network. The neural network is trained by this algorithm in the close-loop to provide an improved training and generalization performance over the standard back-propagation algorithm, in terms of guaranteed stability of the weights, which in turn, yields good tracking performance for the dynamical control system.

Since the stability is the primary concern of a closed-loop system, instead of a direct convergence analysis in [29], we take the traditional approach of adaptive control system to provide a robust input–output (I/O) stability design and analysis for the neural control system, which does not require the weights to converge to the ideal values. And different to [16], we treat the effect of pruning as a dynamic term, which will be discussed in the robustness analysis part. The conic sector theory is applied to isolate the learning and pruning part from the rest of the closed-loop system. A special normalized cost function is provided to the algorithm to reject disturbance and solve the so-called vanished cone problem. The improved performance of the proposed algorithm can be described in terms of better generalization ability, preventing weight shifting, fast convergence and robustness against system disturbance.

## 2. NN tracking controller and the dynamic training and pruning algorithm

### 2.1. Design of the controller

A dynamic control system can be presented at an input–output form [25,24] as the following:

$$y_k = f_{k-1} + u_{k-1} + \varepsilon_k, \tag{1}$$

where $f_{k-1} \in R^m$ is a dynamic nonlinear function, $\varepsilon_k \in R^m$ denotes the system noise vector and $u_{k-1} \in R^m$ is the control signal vector. The tracking error of the control system can be defined as

$$s_k = y_k - d_k, \tag{2}$$

where $d_k \in R^m$ is the command signal.

Define the control signal as

$$u_{k-1} = -\hat{f}_{k-1} + d_k + k_v s_{k-1}, \tag{3}$$

where $k_v$ is the gain parameter of the fixed controller and $\hat{f}_{k-1}$ is the estimate of the nonlinear function $f_{k-1}$ by the neural network. Then the error vector can be presented as

$$e_k = f_{k-1} - \hat{f}_{k-1} + \varepsilon_k \tag{4}$$

to train the neural network as shown in Fig. 1 where the $\bar{\delta}_k$ denotes the impact of the pruning over the whole system which will be further investigated in Sections 3 and 4.

Note that the training error $e_k$ may not be directly measurable, so we should use the tracking error to generate it using the closed-loop relationship (1), (3) and (4)

$$e_k = (1 - z^{-1}k_v)s_k. \tag{5}$$

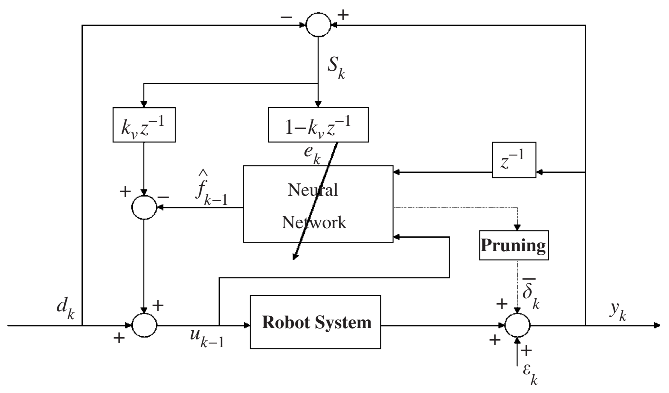

Fig. 1. Structure of the control scheme.

### 2.2. Basic form of adaptive simultaneous perturbation algorithm

The adaptive simultaneous perturbation (ASP) approach [27] is composed of two parallel recursions: one for the weights $W$ and one for the Hessian of the loss function, $L(W)$. The two core recursions are, respectively:

$$\hat{W}_k = \hat{W}_{k-1} - a_k(\overline{\overline{H}})_k^{-1} G_k(\hat{W}_{k-1}), \tag{6}$$

$$\overline{\overline{H}}_k = M_k(\overline{H}_{k-1}), \tag{7}$$

$$\overline{H}_k = \frac{k}{k+1}\overline{H}_{k-1} + \frac{1}{k+1}\hat{H}_k, \tag{8}$$

where $a_k$ is a non-negative scalar gain coefficient, $G_k(\hat{W}_{k-1})$ is the input information related to the gradient or the gradient approximation. $M_k$ is a mapping designed to cope with possible non-positive definiteness of $\overline{H}_k$, and $\hat{H}_k$ is a per-iteration estimation of the Hessian discussed below. The parallel recursions can be implemented once $\hat{H}_k$ is specified and the formula for estimating the Hessian at each iteration can be presented as

$$\hat{H}_k = \frac{1}{2}\left[\frac{\delta G_k^{\mathrm{T}}}{2c_k} r_k + \left(\frac{\delta G_k^{\mathrm{T}}}{2c_k} r_k\right)^{\mathrm{T}}\right], \tag{9}$$

where

$$\delta G_k = G_k^{(1)}(\hat{W}_k + c_k\Delta_k) - G_k^{(1)}(\hat{W}_k - c_k\Delta_k), \tag{10}$$

$$G_k^{(1)}(\hat{W}_k \pm c_k\Delta_k) = \frac{L(\hat{W}_k \pm c_k\Delta_k + \tilde{c}_k\tilde{\Delta}_k) - L(\hat{W}_k \pm c_k\Delta_k - \tilde{c}_k\tilde{\Delta}_k)}{2\tilde{c}_k}\tilde{r}_k \tag{11}$$

with $\Delta_k = (\Delta_{k1}, \Delta_{k1}, \ldots, \Delta_{kp})^{\mathrm{T}}$ is generated via Monte Carlo according to conditions specified in [26,29] and $r_k = (\Delta_{k1}^{-1}, \Delta_{k2}^{-1}, \ldots, \Delta_{kn}^{-1})$. $\tilde{\Delta}_k = (\tilde{\Delta}_{k1}, \tilde{\Delta}_{k2}, \ldots, \tilde{\Delta}_{kp})^{\mathrm{T}}$ is generated in the same statistical manner as $\Delta_k$, but independently of $\Delta_k$, and $\tilde{r}_k = (\tilde{\Delta}_{k1}^{-1}, \tilde{\Delta}_{k2}^{-1}, \ldots, \tilde{\Delta}_{kp}^{-1})^{\mathrm{T}}$. $c_k$ is a positive scalar satisfying certain regularity conditions [26,29] and with $\tilde{c}_k$ satisfying conditions similar to $c_k$.

The output of a three-layer neural network can be presented as

$$\hat{f}_{k-1} = H(\hat{w}_{k-1}, x_{k-1})\hat{v}_{k-1}, \tag{12}$$

where the input vector $x_{k-1} \in R^{n_\mathrm{i}}$ of the neural network is

$$x_{k-1} = [y_{k-1}^\mathrm{T}, y_{k-2}^\mathrm{T}, \ldots, u_{k-2}^\mathrm{T}, u_{k-3}^\mathrm{T} \ldots]^\mathrm{T} \tag{13}$$

$\hat{v}_{k-1} \in R^{p_v}$ is the weight vector of the output layer, and $\hat{w}_{k-1} \in R^{p_w}$ is the weight vector of the hidden layer of the neural network with $p_v = m \times n_\mathrm{h}$ and $p_w = n_\mathrm{h} \times n_\mathrm{i}$, where $n_\mathrm{i}$ and $n_\mathrm{h}$ are the number of the neurons in the input and hidden layers of the network, respectively. $H(\hat{w}_{k-1}, x_{k-1}) \in R^{m \times P_v}$ is the nonlinear activation function matrix:

$$H(\hat{w}_{k-1}, x_{k-1}) = \begin{bmatrix} h_{k-1,1}, h_{k-1,2}, \ldots, h_{k-1,n_\mathrm{h}} & 0\ldots & \ldots 0 \\ 0\ldots & h_{k-1,1}, h_{k-1,2}, \ldots, h_{k-1,n_\mathrm{h}} & \ldots 0 \\ \ldots & \ldots & \ldots \end{bmatrix}$$

with $h_{k-1,i}$ is the nonlinear activation function

$$h_{k-1,i} = h(x_{k-1}^\mathrm{T}\hat{w}_{k-1,i}) = \frac{1}{1 + \mathrm{e}^{-4\lambda x_{k-1}^\mathrm{T}\hat{w}_{k-1,i}}} \tag{14}$$

with $\hat{w}_{k-1,i} \in R^{n_\mathrm{i}}$, $\hat{w}_{k-1} = [\hat{w}_{k-1,1}^\mathrm{T}, \ldots, \hat{w}_{k-1,n_\mathrm{h}}^\mathrm{T}]^\mathrm{T}$ and $4\lambda > 0$, which is the gain parameter of the threshold function.

### 2.3. *Hessian based pruning*

The basic idea of this approach is to use the information on the second-order derivatives of the error surface in order to make a trade-off between network complexity and training error minimization. A similar idea was originated from optimal brain damage (OBD) procedure [13] or optimal brain surgeon (OBS) procedure [8]. The starting point in the construction of such a model is the approximation of the cost function $\xi_\mathrm{av}$ using a *Taylor series* about the operating point, described as follows:

$$\begin{aligned}\xi_\mathrm{av}(\hat{W}_k + \Delta\hat{W}_k) = \xi_\mathrm{av}(\hat{W}_k) &+ G_k^\mathrm{T}(\hat{W}_k)\Delta\hat{W}_k \\ &+ \tfrac{1}{2}\Delta\hat{W}_k^\mathrm{T}H_k(\hat{W}_k)\Delta\hat{W}_k \\ &+ O(\|\Delta\hat{W}_k\|^3),\end{aligned} \tag{15}$$

where $\Delta\hat{W}_k$ is a perturbation applied to the operating point $\hat{W}_k$, with the $G_k(\hat{W}_k)$ is the gradient vector and $H_k(\hat{W}_k)$ is the per-iteration estimated Hessian matrix. The requirement is to identify a set of parameters whose deletion from multilayer perceptron that cause the minimal increase in the value of the cost function $\xi_\mathrm{av}$. However, for our dynamic pruning algorithm, the criteria for pruning should be based on a consecutive $L$-step information (the estimation error of all these $L$ steps are smaller than the criteria, so we regard it reaches a local minimum), where $L$ is a finite positive integer. So

$$\begin{aligned}&\xi_\mathrm{av}(\hat{W}_k + \Delta\hat{W}_k) \\ &\quad = \xi_\mathrm{av}(\hat{W}_k) + \bar{G}_L^\mathrm{T}(\hat{W}_k)\Delta\hat{W}_k \\ &\qquad + \tfrac{1}{2}\Delta\hat{W}_k^\mathrm{T}\bar{H}_L(\hat{W}_k)\Delta\hat{W}_k + O(\|\Delta\hat{W}_k\|^3),\end{aligned}$$

$$\bar{G}_L^\mathrm{T}(\hat{W}_k) = \sum_{k-1+L}^{k} (G_k^\mathrm{T}(\hat{W}_k))/L,$$

$$\bar{H}_L(\hat{W}_k) = \sum_{k-1+L}^{k} (H_k(\hat{W}_k))/L. \tag{16}$$

To solve this problem in practical terms, the following approximations are made:

(1) *Quadratic approximation*: The error surface around a local minimum or global minimum is nearly "quadratic". Then the higher-order terms in Eq. (16) may be neglected.
(2) *Extremal approximation*: The parameters have a set of values corresponding to a local minimum or global minimum of the error surface. In such a case, the gradient vector $\bar{G}_k(\hat{W}_k)$ may be set equal to zero and the term $\bar{G}_k^\mathrm{T}(\hat{W}_k)\Delta\hat{W}_k$ on the right-hand of Eq. (16) may therefore be ignored.

Under these two assumptions, Eq. (16) can be presented approximately as

$$\begin{aligned}\Delta\xi_\mathrm{av} &= \xi_\mathrm{av}(\hat{W}_k + \Delta\hat{W}_k) - \xi_\mathrm{av}(\hat{W}_k) \\ &\simeq \tfrac{1}{2}\Delta\hat{W}_k^\mathrm{T}\bar{H}_L(\hat{W}_k)\Delta\hat{W}_k.\end{aligned} \tag{17}$$

The goal of OBS is to set one of the synaptic weights to zero to minimize the incremental increase in $\xi_\mathrm{av}$ given in Eq. (17). Let $\hat{W}_{ki}$ denote this particular synaptic weight. The elimination of this weight is equivalent to the condition

$$I_i^\mathrm{T}\Delta\hat{W}_k + \hat{W}_{ki} = 0, \tag{18}$$

where $I_i$ is the *unit vector* whose elements are all zero, except for the $i$th element, which is equal to unity. And the goal is to minimize the quadratic term $\frac{1}{2}\Delta\hat{W}_k^\mathrm{T}\bar{H}_L(\hat{W}_k)\Delta\hat{W}_k$ with respect to the perturbation $\Delta\hat{W}_k$, subject to the constraint that $I_i^\mathrm{T}\Delta\hat{W}_k + \hat{W}_{ki}$ is zero, and then minimize the result with respect to the index $i$.

To solve this constraint optimization problem, the following penalty function is used:

$$S = \tfrac{1}{2}\Delta\hat{W}_k^\mathrm{T}\bar{H}_L(\hat{W}_k)\Delta\hat{W}_k - \lambda(I_i^\mathrm{T}\Delta\hat{W}_k + \hat{W}_{ki}), \tag{19}$$

where $\lambda$ is the Lagrange multiplier. Apply the derivative of the risk function $S$ with respect to $\Delta\hat{W}_k$, use the constraint of Eq. (18), and matrix inversion, the optimum solution of the weight vector $\hat{W}_k$ is

$$\Delta\hat{W}_k = -\frac{\hat{W}_{ki}}{[H_{i,i}^{-1}]}H^{-1}I_i \tag{20}$$

and the corresponding optimum value of the risk function $S$ for element $W_i$ is

$$S_i = \frac{(\hat{W}_{ki})^2}{2[H^{-1}_{i,i}]}, \tag{21}$$

where $H^{-1}$ is the inverse of the Hessian matrix $\bar{H}_L$, and $[H^{-1}_{i,i}]$ is the $ii$th element of the inverse matrix. The value $S_i$ optimized with respect to $\Delta\hat{W}_k$, subject to the constraint that the $i$th synaptic weights $\hat{W}_{ki}$ be eliminated, is called the *saliency* of $\hat{W}_{ki}$.

In this method, the weight corresponding to the smallest saliency is the one selected for deletion. Moreover, the corresponding optimal changes in the remainder of the weights are given in Eq. (20).

And till now, the idea of this novel dynamic training and pruning (DTP) algorithm is going to be established. The perturbation is applied to update the weights of both layers by using ASP algorithm and some useful information is extracted in this process to do the pruning when some criteria are satisfied. The detailed steps for implementing this novel algorithm are illustrated next.

**Remark 1.** To simplify the notation, we could define the overall estimate parameter vector $\hat{W}_k = [\hat{v}_k^{\mathrm{T}} \ \hat{w}_k^{\mathrm{T}}] \in R^p$ with $p = p_v + p_w$ in the DTP algorithm. However, for a multi-layered neural network, it may not be possible to update all the estimated parameters with a single gradient approximation function to meet the stability requirement. Therefore, it is better that the estimated parameter vectors $\hat{v}_k$ and $\hat{w}_k$ are updated separately in the DTP algorithm using different gradient approximation functions as in the standard BP training algorithm. This point will be explored further in the robustness analysis in the next section.

## 2.4. The DTP algorithm

Summary of the DTP algorithm for neural controller

*Step* 1.
Initializing: Form the new input vector $x_{k-1}$ of the neural network defined in Eq. (13).

*Step* 2.
Calculating the output $\hat{f}_{k-1}$ of the neural network: Use the input state $x_{k-1}$ and the existing or initial weights of the network in the first iteration.

*Step* 3.
Calculating the control input $u_{k-1}$ by using
$u_{k-1} = -\hat{f}_{k-1} + d_k + k_v s_{k-1}$.

*Step* 4.
Evaluating the estimation error $e_k$ by feeding the tracking error signal $s_k$ into a fixed filter.

*Step* 5.
Evaluate the squared error of $L$ consecutive training samples: If all of them are less than the criteria for pruning $\xi$ (where we assume it reaches a local minimum), goto step 6; else, goto step 7.

*Step* 6.
Do the pruning by choosing the weight corresponding to the minimum saliency, then goto step 7.

*Step* 7.
Updating the weights for the output layer
$\hat{\theta}^v_k = \hat{\theta}^v_{k-1} - \frac{a^v_k (M^v_k)^{-1} e^{p\mathrm{T}}_k H(\hat{\theta}^w_{k-1}, x_{k-1}) 2\Delta^v_k}{\rho^v_k} r^v_k$ (see Eq. (35) in Section 3)
and hidden layer
$\hat{\theta}_k = \hat{\theta}_{k-1} - \frac{a_k (M^w_k)^{-1} e^{p\mathrm{T}}_k (\hat{f}^{w-}_{k-1} - \hat{f}^{w+}_{k-1})}{c^w_k \rho^w_k} r^w_k$ (see Eq. (68) in Section 4),
respectively, and the role of $\rho^v_k$ and $\rho^w_k$ is also illustrated later.

*Step*8.
Go back to step 2 to continue the iteration.

After summarizing the DTP algorithm, it is necessary to investigate the robustness for this algorithm.

## 3. Conic sector condition for the robustness analysis of the learning law in the output layer

The general idea is that we do not deal with the convergence property of the parameter estimate of the dynamic training and pruning algorithm directly, which is well established under certain conditions. Rather we shall prove the tracking error and the parameter estimation error of the output layer of the neural network, which are derived after pruning, are bounded using the conic sector theory under some mild assumptions. Furthermore, the boundedness condition for the parameter estimation error of the hidden layer is also derived in the next section. The purpose of the training algorithm is to make the estimate parameter vector $\hat{W}_k$ approximate the optimal one, and in turn, to produce an optimal tracking error for the control system. To do this, one important condition is that the time-vary equation should be bounded as required for adaptive control systems. To guarantee the boundedness condition, the robust neural controller uses a normalized DTP algorithm. In this paper, the robustness of the system is analyzed using the conic sector theory. A two-stage normalized training strategy is proposed for the DTP algorithm with guaranteed I/O stability using conic sector condition, which also provides guidelines for the selection of the DTP learning parameters and normalization to obtain an improved performance.

As for this training and pruning algorithm, it is more convenient to analyze the conic conditions for the training and pruning processes separately.

## 3.1. Conic sector condition for the training process in the output layer

We use $\|.\|$ to denote the Frobenius norm of a matrix and Euclidean norm of a vector in this paper [4]. The stability of the input–output neural control system can be analyzed by the conic sector theory. The main concern is with the discrete time tracking error signal $s_k$, which is an infinite sequence of real vectors. Consider, the extended space $L_{2e}$, in which the variable truncations lie in $L_2$ with

$$\|s\|_{2,t} = \left\{ \sum_{k=1}^{t} s_k^{\mathrm{T}} s_k \right\}^{1/2} < \infty \tag{22}$$

$\forall t \in Z_+$ (the set of positive integer). The following theory is a necessary extension to the conic sector stability of Safanov [20] for discrete time control systems.

**Theorem 1.** *Consider the following error feedback system*:

$$s_k = e_k^* - P_k,$$
$$\Phi_k = H_1 s_k,$$
$$P_k = H_2 \Phi_k$$

*with operators $H_1, H_2 : L_{2e} \to L_{2e}$ and discrete time signal $s_k, P_k, \Phi_k \in L_{2e}$ and $e_k^* \in L_2$. If*

(a) $H_1 : s_k \to \Phi_k$ *satisfies*

$$\sum_{k=1}^{N} [s_k^{\mathrm{T}} \Phi_k + \sigma s_k^{\mathrm{T}} s_k / 2] > -\gamma,$$

(b) $H_2 : \Phi_k \to P_k$ *satisfies*

$$\sum_{k=1}^{N} [\sigma P_k^{\mathrm{T}} P_k / 2 - P_k^{\mathrm{T}} \Phi_k] \leqslant -\eta \|(P_k, \Phi_k)\|_N^2$$

*for some $\sigma, \eta, \gamma > 0$, then the above feedback system is stable with $s_k, \Phi_k \in L_2$.*

**Proof.** See Corollary 2.1 [3]. □

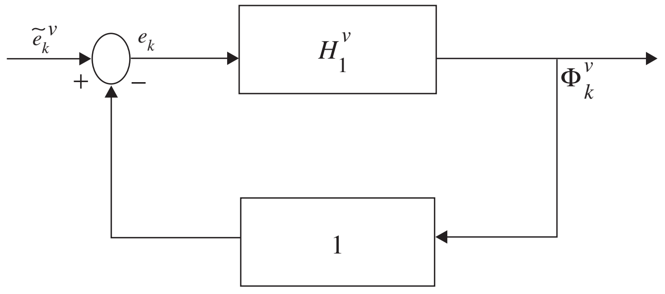

Fig. 2. The equivalent error feedback systems using the conic sector conditions for the estimate error $e_k^{\mathrm{T}}$ and the output $\Phi_k^v = -H(\hat{w}_{k-1}, x_{k-1})\tilde{v}_k$, where $H_2 = 1$.

Note that operator $H_1$ represents the training algorithm here, the input error signal is the tracking error $s_k$ defined in Eq. (2) and the output is $\Phi_k$, which will be defined later and is related to the weight error vectors, and in turn, the estimation parameter error vector $e_k$ and tracking error $s_k$ through Eq. (5). $H_2$ usually represents the mismatched linear model uncertainty in a typical adaptive linear control system and will be defined later in this section.

The first step to use the conic sector stability Theorem 1 is to restructure the control system into an equivalent error feedback system as shown in Fig. 2. Then the parameter estimation error vector should be derived and referred to the output signal $\Phi_k$. For this purpose, define the desired output of the neural network as the plant nonlinear function in Eq. (1)

$$f_{k-1} = H(w_{k-1}^*, x_{k-1}) v^*, \tag{23}$$

where $v^* \in R^{p_v}$ is the ideal weight vector in the output layer of the neural network, $w^* \in R^{p_w}$ is the desired weight vector of the hidden layer. Therefore, the parameter estimate error vectors can be defined as $\tilde{v}_k = v^* - \hat{v}_k \in R^{p_v}$ and $\tilde{w}_k = w^* - \hat{w}_k \in R^{p_w}$ for the output and hidden layers, respectively.

**Assumptions.** (a) The sum of the system disturbance $\varepsilon_k$ is L2 norm bounded.

(b) The ideal weight vector $v^*$ and $w^*$ are L2 norm bounded.

Now we are ready to establish the relationship between the tracking error signal $s_k$ and the parameter estimate vectors of the neural network, which is referred to the operator $H_1$ in Theorem 1 (i.e. the DTP algorithm). According to Eq. (4) the error signals can be extended as

$$\begin{aligned}
e_k &= f_{k-1} - \hat{f}_{k-1} + \varepsilon_k \\
&= H_2(f_{k-1} - \hat{f}_{k-1} + \varepsilon_k) \\
&= H_2(H(w^*, x_{k-1})v^* - H(\hat{w}_{k-1}, x_{k-1})\hat{v}_{k-1} + \varepsilon_k) \\
&= H_2(H(w^*, x_{k-1})v^* - H(\hat{w}_{k-1}, x_{k-1})v^* \\
&\quad + H(\hat{w}_{k-1}, x_{k-1})v^* - H(\hat{w}_{k-1}, x_{k-1})\hat{v}_{k-1} + \varepsilon_k) \\
&= H_2(H(\hat{w}_{k-1}, x_{k-1})\tilde{v}_{k-1} + \tilde{H}(\hat{w}_{k-1}, x_{k-1})v^* + \varepsilon_k) \\
&= H_2 H(\hat{w}_{k-1}, x_{k-1})\tilde{v}_{k-1} + \tilde{e}_k^v \\
&= -H_2 \Phi_k^v + \tilde{e}_k^v,
\end{aligned} \tag{24}$$

where the operator $H_2 = 1$ and

$$\Phi_k^v = -H(\hat{w}_{k-1}, x_{k-1})\tilde{v}_{k-1}, \tag{25}$$

$$\tilde{e}_k^v = H_2(\tilde{H}(\hat{w}_{k-1}, x_{k-1})v^* + \varepsilon_k), \tag{26}$$

$$\tilde{H}(\hat{w}_{k-1}, x_{k-1})v^* = (H(w^*, x_{k-1}) - H(\hat{w}_{k-1}, x_{k-1}))v^*, \tag{27}$$

$$\hat{f}_{k-1}^{v+} = H(\hat{w}_{k-1}, x_{k-1})(\hat{v}_{k-1} + c_k \Delta_k^v), \tag{28}$$

$$\hat{f}_{k-1}^{v-} = H(\hat{w}_{k-1}, x_{k-1})(\hat{v}_{k-1} - c_k \Delta_k^v) \tag{29}$$

with $\Delta_k^v$ and $r_k^v \in R^{p_v}$, which can be viewed as the first $p_v$ components of $\Delta_k$ and $r_k$ defined before.

**Remark 2.** There is an important implication in Eq. (44). The tracking error signal $s_k$ is directly linked to the output signal $\Phi_k^v$, and in turn, the parameter estimation error vector $\tilde{v}_k$ of the output layer of the network, which implies that the training procedure of the output layer of the neural network should be treated separately from the hidden layer of the network to obtain a bounded disturbance term $\tilde{e}_k^v$ as defined in Eq. (26), i.e. $\tilde{e}_k^v \in L_2$ as required by Theorem 1. Therefore, using Eq. (44), we are able to form an equivalent error feedback system Fig. 2 as the one in Theorem 1. Note that $H_2$ usually represents the mismatched linear model uncertainty in a typical adaptive linear control system. Since the neural network has powerful approximation ability to match the nonlinear function without the need to worry about the linear model mismatch.

We define the operator $H_1^v$, which represents the training process of the output layer, and the loss function as

$$L(\hat{v}_{k-1}, \hat{w}_{k-1}) = \|f_k - \hat{f}_k\|^2 \tag{30}$$

and with definitions in equation, we have a gradient approximation using the simultaneous perturbation vector $\Delta_k^v \in R^{p_v}$ to stimulate the weights of the output layer:

$$\begin{aligned}
&\hat{g}(\hat{v}_{k-1}, \hat{w}_{k-1}, \Delta_k^v)\\
&\quad= \frac{L(\hat{v}_{k-1} + c_k^v\Delta_k^v, \hat{w}_{k-1}) - L(\hat{v}_{k-1} - c_k^v\Delta_k^v, \hat{w}_{k-1}) + \varepsilon_k^v}{2c_k^v} r_k^v\\
&\quad= \frac{(f_{k-1} - \hat{f}_{k-1}^{v+})^2 - (f_{k-1} - \hat{f}_{k-1}^{v-})^2 + \varepsilon_k^v}{2c_k^v} r_k^v\\
&\quad= \frac{(f_{k-1} - \hat{f}_{k-1}^{v+} + f_{k-1} - \hat{f}_{k-1}^{v-})(\hat{f}_{k-1}^{v-} - \hat{f}_{k-1}^{v+}) + \varepsilon_k^v}{2c_k^v} r_k^v\\
&\quad= \frac{e_k^{\mathrm{T}}(\hat{f}_{k-1}^{v-} - \hat{f}_{k-1}^{v+})}{c_k^v} r_k^v\\
&\quad= e_k^{\mathrm{T}} H(\hat{w}_{k-1}, x_{k-1}) 2\Delta_k^v r_k^v.
\end{aligned} \tag{31}$$

Since in the basic form of ASP, $\overline{H}_k$ is actually the sample mean of $\hat{H}_k$ during the period, which is

$$\overline{H}_k = \frac{1}{k+1}\sum_{k=0}^{k} \hat{H}_k \tag{32}$$

and according to the definition of $\hat{H}_k$, we can get

$$\hat{H}_k{}^v = \frac{1}{2}\left[\frac{\delta G_k^{v\mathrm{T}}}{2c_k^v} r_k^v + \left(\frac{\delta G_k^{v\mathrm{T}}}{2c_k^v} r_k^v\right)^{\mathrm{T}}\right]$$

$$\begin{aligned}
&\delta G_{k-1}^v\\
&\quad= \frac{L(\hat{w}_{k-1}, \hat{v}_{k-1} + c_k^v\Delta_k^v + \tilde{c}_k^v\tilde{\Delta}_k^v) - L(\hat{w}_{k-1}, \hat{v}_{k-1} + c_k^v\Delta_k^v - \tilde{c}_k^v\tilde{\Delta}_k^v) + \tilde{\varepsilon}_k^v}{2\tilde{c}_k^v} \tilde{r}_k^v\\
&\qquad+ \frac{L(\hat{w}_{k-1}, \hat{v}_{k-1} - c_k^v\Delta_k^v + \tilde{c}_k^v\tilde{\Delta}_k^v) - L(\hat{w}_{k-1}, \hat{v}_{k-1} - c_k^v\Delta_k^v - \tilde{c}_k^v\tilde{\Delta}_k^v) + \tilde{\varepsilon}_k^v}{2\tilde{c}_k^v} \tilde{r}_k^v\\
&\quad= g(\hat{w}_{k-1}, \hat{v}_{k-1} + c_k^v\Delta_k^v) - g(\hat{w}_{k-1}, \hat{v}_{k-1} - c_k^v\Delta_k^v)\\
&\quad= e(\hat{v}_{k-1} + c_k^v\Delta_k^v) H(\hat{w}_{k-1}, x_{k-1}) 2\tilde{\Delta}_k^v\tilde{r}_k^v\\
&\qquad- e(\hat{v}_{k-1} - c_k^v\Delta_k^v) H(\hat{w}_{k-1}, x_{k-1}) 2\tilde{\Delta}_k^v\tilde{r}_k^v\\
&\quad= (f_{k-1} - \hat{f}_{k-1}^{v+} + \varepsilon_k) H(\hat{w}_{k-1}, x_{k-1}) 2\tilde{\Delta}_k^v\tilde{r}_k^v\\
&\qquad- (f_{k-1} - \hat{f}_{k-1}^{v-} + \varepsilon_k) H(\hat{w}_{k-1}, x_{k-1}) 2\tilde{\Delta}_k^v\tilde{r}_k^v\\
&\quad= (\hat{f}_{k-1}^{v-} - \hat{f}_{k-1}^{v+}) H(\hat{w}_{k-1}, x_{k-1}) 2\tilde{\Delta}_k^v\tilde{r}_k^v\\
&\quad= 4H(\hat{w}_{k-1}, x_{k-1}) H^{\mathrm{T}}(\hat{w}_{k-1}, x_{k-1}) c_k^v\Delta_k^v
\end{aligned} \tag{33}$$

with $\Delta_k^v, \tilde{\Delta}_k^v, r_k^v, \tilde{r}_k^v \in R^{p_v}$, which can be viewed as the first $p_v$ components of $\Delta_k, \tilde{\Delta}_k, r_k, \tilde{r}_k$, respectively.

And then we can get the $\overline{H}_k^v$

$$\overline{H}_k^v = \frac{1}{k+1}\sum_{k=0}^{k} \hat{H}_k^v = \frac{1}{k+1}\sum_{k=0}^{k} (2H(\hat{w}_k, x_k) H^{\mathrm{T}}(\hat{w}_k, x_k)). \tag{34}$$

After we define the $M_k^v$, which is after the mapping of $\overline{H}_k^v$ and consider the pruning, we have a normalized learning law for the weight of the output layer:

$$\hat{v}_k = \hat{v}_{k-1} - \frac{a_k^v (M_k^v)^{-1} e_k^{\mathrm{T}} H(\hat{w}_{k-1}, x_{k-1}) 2\Delta_k^v}{\rho_k^v} r_k^v, \tag{35}$$

where $\rho$ is the bounded normalization factor, which is traditionally used in adaptive control system to bound the signals in learning algorithm [3], is defined as

$$\rho_k^v = \mu\rho_{k-1}^v + \max\{[(M_k^v)^{-1} H(\hat{w}_{k-1}, x_{k-1})]^2, \bar{\rho}\} \tag{36}$$

with $\bar{\rho} > 0, u \in (0, 1)$.

Then the stability analysis of the robust neural controller can be justified by the conic sector condition, which requires the feedback system in Fig. 2 to meet certain dissipative condition as in Theorem 1 and can be justified as in the following theorem.

**Theorem 2.** *The operator $H_1^v$: $e_k \to \Phi_k^v$, which represents the training process of the output layer* (*see Fig.* 2), *satisfies the condition* (a) *and* (b) *of Theorem* 1, *i.e.* $e_k, \Phi_k^v \in L_2$.

**Proof.** As an easy starting point, we establish the conic sector condition between the estimate error $e_k$ and the output $\Phi_k^v$ first and then extend to the tracking error $s_k$ later. Using the learning law, we have

$$\begin{aligned}
&\|\tilde{v}_k\|^2 - \|\tilde{v}_{k-1}\|^2\\
&\quad= -2\{a_k^v (M_k^v)^{-1} e_k^{\mathrm{T}} H(\hat{w}_{k-1}, x_{k-1}) \Delta_k^v r_k^{v\mathrm{T}} \tilde{v}_{k-1}\} (\rho_k^v)^{-1}\\
&\qquad+ (a_k^v (M_k^v)^{-1} e_k^{\mathrm{T}} H(\hat{w}_{k-1}, x_{k-1}) \Delta_k^v r_k^{v\mathrm{T}} (\rho_k^v)^{-1})^2\\
&\quad= 2\{a_k^v (M_k^v)^{-1}\} | - e_k^{\mathrm{T}} H(\hat{w}_{k-1}, x_{k-1}) \tilde{v}_{k-1}|\\
&\qquad\times |\tilde{v}_{k-1}^{\mathrm{T}} (\tilde{v}_{k-1}\tilde{v}_{k-1}^{\mathrm{T}})^{-1} \Delta_k^v r_k^{v\mathrm{T}} \tilde{v}_{k-1}| (\rho_k^v)^{-1}\\
&\qquad+ (a_k^v (M_k^v)^{-1} e_k^{\mathrm{T}} H(\hat{w}_{k-1}, x_{k-1}) \Delta_k^v r_k^{v\mathrm{T}} (\rho_k^v)^{-1})^2.
\end{aligned} \tag{37}$$

Consider the property of the gradient of the square error signal around attractor basin [25]

$$-\frac{\partial(e_k^{\mathrm{T}} e_k)}{\partial(\hat{v}_{k-1})^{\mathrm{T}}} \tilde{v}_{k-1} = -e_k^{\mathrm{T}} H(\hat{w}_{k-1}, x_{k-1}) \tilde{v}_{k-1} = e_k^{\mathrm{T}} \Phi_k^{\mathrm{T}} \geqslant 0 \tag{38}$$

and the trace property

$$\begin{aligned}
&|\tilde{v}_{k-1}^{\mathrm{T}} (\tilde{v}_{k-1}\tilde{v}_{k-1}^{\mathrm{T}})^{-1} \Delta_k^v r_k^{v\mathrm{T}} \tilde{v}_{k-1}|\\
&\quad= |\mathrm{tr}\{\tilde{v}_{k-1}^{\mathrm{T}} (\tilde{v}_{k-1}\tilde{v}_{k-1}^{\mathrm{T}})^{-1} \Delta_k^v r_k^{v\mathrm{T}} \tilde{v}_{k-1}\}|,
\end{aligned}$$

$$|\mathrm{tr}\{\tilde{v}_{k-1}\tilde{v}_{k-1}^{\mathrm{T}} (\tilde{v}_{k-1}\tilde{v}_{k-1}^{\mathrm{T}})^{-1} \Delta_k^v r_k^{v\mathrm{T}}\}| = |\mathrm{tr}\{\Delta_k^v r_k^{v\mathrm{T}}\}| \Delta q p_v. \tag{39}$$

Eq. (48) can be rewritten as

$$\|\tilde{v}_k\|^2 - \|\tilde{v}_{k-1}\|^2 \leqslant 2a_k(M_k^v)^{-1}p_v\{e_k^{\mathrm{T}}\Phi_k^v\}(\rho_k^v)^{-1} + \|(a_k^v(M_k^v)^{-1}e_k^{\mathrm{T}}H(\hat{w}_{k-1},x_{k-1})\Delta_k^v r_k^{v\mathrm{T}}(\rho_k^v)^{-1})\|^2.$$

Summing the above equation upon to $N$ steps, we are able to establish the conic condition with a constant $\breve{\sigma}^v$ for the estimate error $e_k$ as the input in Theorem 1 and the output $\Phi_k^v$ similar to Ref. [22]:

$$\sum_{k=1}^{N}\left\{e_k^{\mathrm{T}}\Phi_k^v + e_k^{\mathrm{T}}e_k\frac{\breve{\sigma}^v}{2}\right\} \geqslant -(\tilde{v}_0)^2\left\{\frac{M_k^v\,\breve{\rho}_k^v}{2a_kp_v}\right\} \tag{40}$$

by selecting a suitable normalized factor $\breve{\rho}_k^v$ to obtain the constant number $\breve{\sigma}^v$ such that

$$1 > \breve{\sigma}^v \geqslant \frac{a_k^v(M_k^v)^{-1}}{p_v}\|\Delta_k^v\|^2\|r_k^v\|^2(\breve{\rho}_k^v)^{-1}. \tag{41}$$

Note that the conic sector condition guarantees that the estimate error $e_k$ and the output $\Phi_k^v = -H(\hat{w}_{k-1},x_{k-1})\tilde{v}_k$ are bounded according to Theorem 1, and in turn, the parameter estimate error $\tilde{v}_k$.

The specified normalized factor $\rho_k^v$ plays two important roles. Firstly, it guarantees $\sigma^v < 1$ to avoid the so-called vanished cone problem [3]; secondly, it guarantees the sector conditions of Theorem 1 to be simultaneously satisfied by both the original feedback system and the normalized equivalent feedback system. Therefore, both conditions (a) and (b) of Theorem 1 for $e_k$ are fulfilled.

According to Theorem 1, $\Phi_k^v = -H(\hat{w}_{k-1},x_{k-1})\tilde{v}_{k-1} \in L_2$ leads to the bounded parameter estimation error $\tilde{v}_{k-1}$ for the training process of the output layer. □

## 3.2. *Conic sector condition for the pruning process in the output layer*

However, from the criterion of pruning, we know that there is pruning once per step at most. So here a new term $e_k^p$ is defined to denote the estimation error instead.

$$e_k^p = H_2(f_{k-1} - \hat{f}_{k-1}^p + \varepsilon_k),$$

where $\hat{f}_{k-1}^p$ is the network output with pruning. And from the definition of this perturbation based pruning, it is obvious that

$$\hat{f}_{k-1}^p = \hat{f}_{k-1}(\hat{v}_{k-1} + \Delta v_{k-1}), \tag{42}$$

where the $\Delta v_{k-1}$ is the added perturbation for pruning. According to the pruning criteria and Eq. (20), we know

$$\Delta v_{k-1} = -\frac{v_{k-1,i}}{[\bar{H}_{L_{k-1,i,i}}^v]^{-1}}\bar{H}_{L_{k-1}}^v I_{k-1,i},$$

$$\bar{H}_{L_{k-1}}^v = \sum_{m=k-L}^{k-1}(\hat{H}_m^v)/L$$

with $i$ denoting the specific weight to be pruned, $I_{k-1,i}$ is a unit vector whose elements are all zero, except for the $i$th element, which is equal to unity. $\hat{H}_{k-1}^v$ is the per-iteration estimation Hessian matrix for the output layer. From the mean value theory, we have

$$\hat{f}_{k-1}(\hat{v}_{k-1} + \Delta v_{k-1}) - \hat{f}_{k-1}(\hat{v}_{k-1}) = \Delta v_{k-1}\cdot\hat{f}'_{k-1,v}, \tag{43}$$

where $\hat{f}'_{k-1,v}$ is the first derivative of the network output with respect to the weights of the output layer. It is easy to see that $\|\hat{f}'_{k-1,v}\| < 1$.

From above analysis, $e_k^p$ can be rewritten as

$$\begin{aligned} e_k^p &= H_2(f_{k-1} - \hat{f}_{k-1}^p + \varepsilon_k) \\ &= H_2(f_{k-1} - (\hat{f}_{k-1} + \Delta v_{k-1}\hat{f}'_{k-1,v}) + \varepsilon_k) \\ &= -H_2(\Phi_k^v - \Delta v_{k-1}\hat{f}'_{k-1,v}) + \tilde{e}_k^v. \end{aligned} \tag{44}$$

From the basic concept of pruning, we know that some weight is deleted during the pruning. For example, let $\hat{v}_{k-1}^p \in R^n$ denote the weight vector after pruning of $\hat{v}_{k-1}$, and we assume that the $n$th weight $\hat{v}_{k,n}$ is pruned, so we have

$$\hat{v}_{k-1}^p = \Delta v_{k-1} + \hat{v}_{k-1}$$

$$\begin{aligned} \Delta v_{k-1} &= \hat{v}_{k-1}^p - \hat{v}_{k-1} \\ &= [\hat{v}_{k-1,1},\ldots,0] - [\hat{v}_{k-1,1},\ldots,\hat{v}_{k-1,n}] \\ &= [0,\ldots,-\hat{v}_{k-1,n}] \end{aligned} \tag{45}$$

which shows that the $\Delta v_{k-1}$ is related to the current weight value, so from Theorem 2, $\|\Delta v_{k-1}\|$ is a bounded term.

For the convenience of the analysis later, we can rewrite the $e_k^p$ as a term so that it can describe the estimation error with pruning

$$e_k^p = -H_2\Phi_k^{vp} + \tilde{e}_k^v,$$

where

$$\Phi_k^{vp} = \Phi_k^v - \tau_k^v, \tag{46}$$

where $\tau_k^v = \Delta v_{k-1}\hat{f}'_{k-1,v}$ and $\|\tau_k^v\| \leqslant \tau_{\max}^v$ with $\tau_{\max}^v$ is a positive constant.

And similarly, we have the learning law for the output layer after pruning:

$$\hat{v}_k = \hat{v}_{k-1} - \frac{a_k^v(M_k^v)^{-1}e_k^{p\mathrm{T}}H(\hat{w}_{k-1},x_{k-1})2\Delta_k^v}{\rho_k^v}r_k^v. \tag{47}$$

After reconstruct a new feedback system by using the new input $e_k^{p\mathrm{T}}$ and output term $\Phi_k^{vp} = \Phi_k^v - \tau_k^v$, we can justify the conic sector condition for the learning law with pruning for the output layer as in Fig. 3

$$\begin{aligned} &\|\tilde{v}_k\|^2 - \|\tilde{v}_{k-1}\|^2 \\ &= -2\{a_k^v(M_k^v)^{-1}e_k^{p\mathrm{T}}H(\hat{w}_{k-1},x_{k-1})\Delta_k^{pv}r_k^{v\mathrm{T}}\tilde{v}_{k-1}\}(\rho_k^v)^{-1} \\ &\quad + (a_k^v(M_k^v)^{-1}e_k^{p\mathrm{T}}H(\hat{w}_{k-1},x_{k-1})\Delta_k^v r_k^{v\mathrm{T}}(\rho_k^v)^{-1})^2 \\ &= 2\{a_k^v(M_k^v)^{-1}\}|-e_k^{p\mathrm{T}}H(\hat{w}_{k-1},x_{k-1})\tilde{v}_{k-1}| \\ &\quad \times|\tilde{v}_{k-1}^{\mathrm{T}}(\tilde{v}_{k-1}\tilde{v}_{k-1}^{\mathrm{T}})^{-1}\Delta_k^v r_k^{v\mathrm{T}}\tilde{v}_{k-1}|(\rho_k^v)^{-1} \\ &\quad + (a_k^v(M_k^v)^{-1}e_k^{p\mathrm{T}}H(\hat{w}_{k-1},x_{k-1})\Delta_k^v r_k^{v\mathrm{T}}(\rho_k^v)^{-1})^2 \\ &= 2\{a_k^v(M_k^v)^{-1}\}|e_k^{p\mathrm{T}}(\Phi_k^{vp} + \tau_k^v)| \\ &\quad \times|\tilde{v}_{k-1}^{\mathrm{T}}(\tilde{v}_{k-1}\tilde{v}_{k-1}^{\mathrm{T}})^{-1}\Delta_k^v r_k^{v\mathrm{T}}\tilde{v}_{k-1}|(\rho_k^v)^{-1} \\ &\quad + (a_k^v(M_k^v)^{-1}e_k^{p\mathrm{T}}H(\hat{w}_{k-1},x_{k-1})\Delta_k^v r_k^{v\mathrm{T}}(\rho_k^v)^{-1})^2. \end{aligned} \tag{48}$$

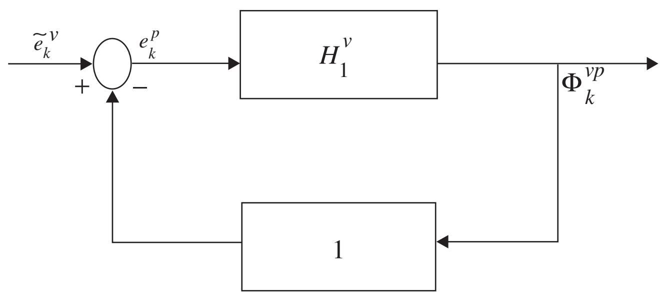

Fig. 3. The equivalent error feedback systems using the conic sector conditions for the estimate error $e_k^{p\mathrm{T}}$ and the output $\Phi_k^{vp} = -H(\hat{w}_{k-1}, x_{k-1})\tilde{v}_k - \tau_k^v$, where $H_2 = 1$.

Consider the trace property

$$\begin{aligned}&|\tilde{v}_{k-1}^{\mathrm{T}}(\tilde{v}_{k-1}\tilde{v}_{k-1}^{\mathrm{T}})^{-1}\Delta_k^v r_k^{v\mathrm{T}}\tilde{v}_{k-1}|\\&\quad = |\mathrm{tr}\{\tilde{v}_{k-1}^{\mathrm{T}}(\tilde{v}_{k-1}\tilde{v}_{k-1}^{\mathrm{T}})^{-1}\Delta_k^v r_k^{v\mathrm{T}}\tilde{v}_{k-1}\}|,\end{aligned}$$

$$|\mathrm{tr}\{\tilde{v}_{k-1}\tilde{v}_{k-1}^{\mathrm{T}}(\tilde{v}_{k-1}\tilde{v}_{k-1}^{\mathrm{T}})^{-1}\Delta_k^v r_k^{v\mathrm{T}}\}| = |\mathrm{tr}\{\Delta_k^v r_k^{v\mathrm{T}}\}|\Delta q p_v. \tag{49}$$

Eq. (48) can be rewritten as

$$\begin{aligned}&\|\tilde{v}_k\|^2 - \|\tilde{v}_{k-1}\|^2\\&\quad = 2a_k(M_k^v)^{-1}p_v\{e_k^{p\mathrm{T}}\Phi_k^{vp}\}(\rho_k^v)^{-1} + 2a_k(M_k^v)^{-1}p_v e_k^{p\mathrm{T}}\tau_k^v(\rho_k^v)^{-1}\\&\qquad + \|(a_k^v(M_k^v)^{-1}e_k^{\mathrm{T}}H(\hat{w}_{k-1}, x_{k-1})\Delta_k^v r_k^{v\mathrm{T}}(\rho_k^v)^{-1})\|^2.\end{aligned}$$

And from $\frac{1}{2}a^2 + 2b^2 \geqslant 2ab$, we have

$$\begin{aligned}&\|\tilde{v}_k\|^2 - \|\tilde{v}_{k-1}\|^2\\&\quad \leqslant 2a_k(M_k^v)^{-1}p_v\{e_k^{p\mathrm{T}}\Phi_k^{vp}\}(\rho_k^v)^{-1}\\&\qquad + a_k(M_k^v)^{-1}p_v(\tfrac{1}{2}\|e_k^{p\mathrm{T}}\|^2 + 2\|\tau_k^v\|^2)(\rho_k^v)^{-1}\\&\qquad + \|(a_k^v(M_k^v)^{-1}e_k^{p\mathrm{T}}H(\hat{w}_{k-1}, x_{k-1})\Delta_k^v r_k^{v\mathrm{T}}(\rho_k^v)^{-1})\|^2\\&\quad \leqslant 2a_k(M_k^v)^{-1}p_v\{e_k^{p\mathrm{T}}\Phi_k^{vp}\}(\rho_k^v)^{-1}\\&\qquad + a_k(M_k^v)^{-1}p_v(\tfrac{1}{2}\|e_k^{p\mathrm{T}}\|^2 + 2\|\tau_{\max}^v\|^2)(\rho_k^v)^{-1}\\&\qquad + \|(a_k^v(M_k^v)^{-1}e_k^{p\mathrm{T}}H(\hat{w}_{k-1}, x_{k-1})\Delta_k^v r_k^{v\mathrm{T}}(\rho_k^v)^{-1})\|^2.\end{aligned} \tag{50}$$

Summing the above equation upon to $N$ steps, we are able to establish the conic condition with a constant $\breve{\sigma}^v$ for the estimate error $e_k^p$ as the input in Fig. 3 and the output $\Phi_k^{vp}$:

$$\sum_{k=1}^{N}\left\{e_k^{p\mathrm{T}}\Phi_k^{vp} + e_k^{p\mathrm{T}}e_k^p\frac{\breve{\sigma}^v}{2}\right\} \geqslant -\left[(\tilde{v}_0)^2\left\{\frac{M_k^v\,\rho_k^v}{2a_k p_v}\right\} + N\|\tau_{\max}^v\|^2\right] \tag{51}$$

by selecting a suitable normalized factor $\rho_k^v$ to obtain the constant number $\breve{\sigma}^v$ such that

$$1 > \breve{\sigma}^v > \frac{1}{4} + \frac{a_k^v(M_k^v)^{-1}}{2p_v}\|\Delta_k^v\|^2\|r_k^v\|^2(\rho_k^v)^{-1}. \tag{52}$$

Note that the conic sector condition guarantees that the estimate error $e_k^p$ and the output $\Phi_k^{pv} = -H(\hat{w}_{k-1}, x_{k-1})\tilde{v}_k - \eta_k^v\tau_k^v$ are bounded according to Theorem 1.

The specified normalized factor $\rho_k^v$ plays two important roles. Firstly, it guarantees $\sigma^v < 1$ to avoid the so-called vanished cone problem [3]; secondly, it guarantees the sector conditions of Theorem 1 to be simultaneously satisfied by both the original feedback system and the normalized equivalent feedback system. Therefore, both conditions (a) and (b) of Theorem 1 for $e_k^p$ are fulfilled.

This completes the proof. □

## 4. Conic sector condition for the robustness analysis of the learning law in the hidden layer

As justified in Remark 1, the hidden layer parameter of the network should be estimated separately. Therefore, a conic sector condition will be established for the hidden layer training in this section.

### 4.1. Conic sector condition for the training process in the hidden layer

Similar to the error equation Eq. (24), one can rewrite it as

$$\begin{aligned}e_k &= H_2(f_{k-1} - \hat{f}_{k-1} + \varepsilon_k)\\&= H_2(H(w^*, x_{k-1})v^* - H(\hat{w}_{k-1}, x_{k-1})\hat{v}_{k-1} + \varepsilon_k)\\&= H_2(H(w^*, x_{k-1})v^* - H(w^*, x_{k-1})\hat{v}_{k-1} + H(\hat{w}^*, x_{k-1})\hat{v}_{k-1}\\&\qquad - H(\hat{w}_{k-1}, x_{k-1})\hat{v}_{k-1} + \varepsilon_k)\\&= H_2(\tilde{H}(\hat{w}_{k-1}, x_{k-1})\hat{v}_{k-1} + H(w^*, x_{k-1})\tilde{v}_{k-1} + \varepsilon_k)\\&= H_2[\tilde{H}(\hat{w}_{k-1}, x_{k-1})\hat{v}_{k-1} + H(w^*, x_{k-1})\tilde{v}_{k-1} + \varepsilon_k]\\&= -H_2\Phi_k^w + \tilde{e}_k^w\end{aligned} \tag{53}$$

with $\tilde{e}_k^w = H_2(H(w^*, x_{k-1})\tilde{v}_{k-1} + \varepsilon_k)$ and $\Phi_k^w = -\tilde{\Omega}_{k-1}\tilde{w}_{k-1}$ where $\tilde{H}(\hat{w}_{k-1}) = H(w^*, x_{k-1}) - H(\hat{w}_{k-1}, x_{k-1}), \tilde{v}_{k-1,i} = v_i^* - \hat{v}_{k-1,i}, \tilde{w}_{k-1,i} = w_i^* - \hat{w}_{k-1,i}$, and $\tilde{\Omega}_{k-1} \in R^{m\times p_w}$ is the following matrix:

$$\tilde{\Omega}_{k-1} = \begin{bmatrix}\tilde{\mu}_{k-1,1}x_{k-1,1}^{\mathrm{T}}\hat{v}_{k-1,(1,1)}, \ldots, \tilde{\mu}_{k-1,1}x_{k-1,n_{\mathrm{i}}}^{\mathrm{T}}\hat{v}_{k-1,(1,1)}, \ldots, \tilde{\mu}_{k-1,n_{\mathrm{h}}}x_{k-1,n_{\mathrm{i}}}^{\mathrm{T}}\hat{v}_{k-1,(1,n_{\mathrm{h}})}\\ \vdots \\ \tilde{\mu}_{k-1,1}x_{k-1,1}^{\mathrm{T}}\hat{v}_{k-1,(m,1)}, \ldots \tilde{\mu}_{k-1,1}x_{k-1,n_{\mathrm{i}}}^{\mathrm{T}}\hat{v}_{k-1,(m,1)}, \ldots \tilde{\mu}_{k-1,n_{\mathrm{h}}}x_{k-1,n_{\mathrm{i}}}^{\mathrm{T}}\hat{v}_{k-1,(m,n_{\mathrm{h}})}\end{bmatrix}$$

$\in R^{m\times p_w}$ with $\hat{v}_{k-1,i} \in R^{n_{\mathrm{h}}}$, $\hat{v}_{k-1} = [\hat{v}_{k-1,1}^{\mathrm{T}}, \ldots, \hat{v}_{k-1,m}^{\mathrm{T}}]^{\mathrm{T}}$ where $n_{\mathrm{h}}$ is the number of neurons in the hidden layer of the network.

Note the above equation is derived from the mean value theorem and the activation function is a non-decreasing function, so there exist unique positive numbers $\tilde{\mu}_{k-1,i}$

$$\begin{aligned}&h_{k-1,i}(w_i^*, x_{k-1}) - h_{k-1,i}(\hat{w}_{k-1,i}, x_{k-1})\\&\quad = \tilde{\mu}_{k-1,i}x_{k-1}^{\mathrm{T}}(w_i^* - \hat{w}_{k-1,i}),\end{aligned} \tag{54}$$

where $\hat{w}_{k-1,i}, w_i^* \in R^{n_{\mathrm{i}}}$ are the estimation and ideal weight vectors linked to the $i$th hidden layer neuron, respectively. The maximum value of the derivative $h'_{k-1,i}$ is $\lambda$, therefore

$$\lambda \geqslant \tilde{\mu}_{k-1,i} \geqslant 0 \quad (1 \leqslant i \leqslant n_{\mathrm{h}}). \tag{55}$$

After we define

$$\hat{f}_{k-1}^{w+} = H(\hat{w}_{k-1} + c_k^w\Delta_k^w, x_{k-1})\hat{v}_{k-1}, \tag{56}$$

$$\hat{f}_{k-1}^{w-} = H(\hat{w}_{k-1} - c_k^w\Delta_k^w, x_{k-1})\hat{v}_{k-1}. \tag{57}$$

The gradient approximation of the hidden layer can be written as

$$\begin{aligned}
&\hat{g}(\hat{v}_{k-1}, \hat{w}_{k-1}, \Delta^w_{k-1})\\
&= \frac{L(\hat{v}_{k-1}, \hat{w}_{k-1} + c^w_k \Delta^w_k) - L(\hat{v}_{k-1}, \hat{w}_{k-1} - c^w_k \Delta^w_k) + \varepsilon^w_k}{2c^w_k} r^w_k\\
&= \frac{(f_{k-1} - \hat{f}^{w+}_{k-1})^2 - (f_{k-1} - \hat{f}^{w-}_{k-1})^2 + \varepsilon^w_k}{2c^w_k} r^w_k\\
&= \frac{(f_{k-1} - \hat{f}^{w+}_{k-1} + f_{k-1} - \hat{f}^{w-}_{k-1})(\hat{f}^{w-}_{k-1} - \hat{f}^{w+}_{k-1}) + \varepsilon^w_k}{2c^w_k} r^w_k\\
&= \frac{[2(f_{k-1} - \hat{f}_{k-1}) + (\hat{f}_{k-1} - \hat{f}^{w+}_{k-1}) + (\hat{f}_{k-1} - \hat{f}^{w-}_{k-1})]^{\mathrm{T}}(\hat{f}^{w-}_{k-1} - \hat{f}^{w+}_{k-1}) + \varepsilon^w_k}{2c^w_k} r^w_k\\
&= \frac{e^{\mathrm{T}}_k(\hat{f}^{w-}_{k-1} - \hat{f}^{w+}_{k-1})}{c^w_k} r^w_k \qquad (58)\\
&= \frac{e^{\mathrm{T}}_k c^w_k \Delta^w_k \hat{\Omega}_{k-1}}{c^w_k} r^w_k, \qquad (59)
\end{aligned}$$

where $\hat{\Omega}_{k-1}$ is defined similarly to $\tilde{\Omega}_{k-1}$. And similarly, after define

$$\hat{f}^{w++}_{k-1} = H(\hat{w}_{k-1} + c^w_k \Delta^w_k + \tilde{c}_k \tilde{\Delta}_k, x_{k-1})\hat{v}_{k-1}, \qquad (60)$$

$$\hat{f}^{w+-}_{k-1} = H(\hat{w}_{k-1} + c^w_k \Delta^w_k - \tilde{c}_k \tilde{\Delta}_k, x_{k-1})\hat{v}_{k-1}, \qquad (61)$$

$$\hat{f}^{w-+}_{k-1} = H(\hat{w}_{k-1} - c^w_k \Delta^w_k + \tilde{c}_k \tilde{\Delta}_k, x_{k-1})\hat{v}_{k-1}, \qquad (62)$$

$$\hat{f}^{w--}_{k-1} = H(\hat{w}_{k-1} - c^w_k \Delta^w_k - \tilde{c}_k \tilde{\Delta}_k, x_{k-1})\hat{v}_{k-1} \qquad (63)$$

we can get

$$\begin{aligned}
&G^{(1)}_k(\hat{w}_k + c_k \Delta_k)\\
&= \frac{L(\hat{w}_k + c_k\Delta_k + \tilde{c}_k\tilde{\Delta}_k) - L(\hat{w}_k + c_k\Delta_k - \tilde{c}_k\tilde{\Delta}_k)}{2\tilde{c}_k} \tilde{r}_k\\
&= \frac{L(\hat{w}^{w+}_k + \tilde{c}_k\tilde{\Delta}_k) - L(\hat{w}^{w+}_k - \tilde{c}_k\tilde{\Delta}_k)}{2\tilde{c}_k} \tilde{r}_k\\
&= g(\hat{w}^{w+}_k), \qquad (64)
\end{aligned}$$

where $\hat{w}^{w+}_k = \hat{w}_k + c^w_k \Delta^w_k$. Then we can obtain $\delta G^w(k)$ as follows:

$$\delta G^w(k) = \hat{g}(\hat{w}^{w+}_k) - \hat{g}(\hat{w}^{w-}_k). \qquad (65)$$

So we can get

$$\begin{aligned}
\delta G^w(k) &= e^{\mathrm{T}}_k(w_k + c^w_k\Delta^w_k)\hat{\Omega}^+_{k-1} - e^{\mathrm{T}}_k(w_k - c^w_k\Delta^w_k)\hat{\Omega}^-_{k-1}\\
&= (f_{k-1} - \hat{f}^{w+}_{k-1} + \varepsilon_k)\hat{\Omega}^+_{k-1} - (f_{k-1} - \hat{f}^{w-}_{k-1} + \varepsilon_k)\hat{\Omega}^-_{k-1},
\end{aligned}$$

where $\hat{\Omega}^+_{k-1}$ and $\hat{\Omega}^-_{k-1}$ are also similar to $\tilde{\Omega}_{k-1}$.

And we can take it as a new nonlinear function like $R(\hat{w}_k) = (f_{k-1} - \hat{f}_{k-1} + \varepsilon_k)\hat{\Omega}_{k-1}$, so we can take the equation above as $R(\hat{w}_k + c^w_k\Delta^w_k) - R(\hat{w}_k - c^w_k\Delta^w_k)$. Using the mean value theorem again, we can get

$$R(\hat{w}_{k-1} + c^w_k\Delta^w_k) - R(\hat{w}_{k-1} - c^w_k\Delta^w_k) = 2c^w_k\Delta^w_k\bar{\Omega}_{k-1} \qquad (66)$$

with $\bar{\Omega}_{k-1}$ defined similarly to $\tilde{\Omega}_{k-1}$.

Thus the per-iteration estimation of Hessian is calculated as

$$\overline{H}^w_k = \frac{1}{k+1}\sum_{k=0}^{k}(\bar{\Omega}_k). \qquad (67)$$

Let $M^w_k$ denote $\overline{\overline{H}}{}^w_k$ which is after the mapping of $\overline{H}^w_k$, so we can rewrite the learning algorithm after considering pruning for hidden layer as

$$\hat{w}_k = \hat{w}_{k-1} - \frac{a_k(M^w_k)^{-1}e^{\mathrm{T}}_k(\hat{f}^{w-}_{k-1} - \hat{f}^{w+}_{k-1})}{c^w_k\rho^w_k} r^w_k. \qquad (68)$$

**Theorem 3.** *The operator* $H^w_1 : e_k \to \Phi^w_k$, *which represents the training algorithm of the hidden layer, and* $H_2$ *satisfy the condition* (a) *and* (b) *of Theorem* 1.

**Proof.** Consider that the DTP is an approximation algorithm, and using the property of local minimum points of the gradient $-(\partial/\partial(w_{k-1,i})^{\mathrm{T}})\,e^{\mathrm{T}}_k e_k \tilde{w}_{k-1,i} \geqslant 0$. We have

$$\begin{aligned}
0 &\leqslant -\frac{\partial}{\partial(\hat{w}_{k-1})^{\mathrm{T}}} e^{\mathrm{T}}_k e_k \tilde{w}_{k-1} = \sum_{i=1}^{n_{\mathrm{h}}}\left\{\frac{\partial}{\partial(\hat{w}_{k-1,i})^{\mathrm{T}}} e^{\mathrm{T}}_k e_k \tilde{w}_{k-1,i}\right\}\\
&= \sum_{i=1}^{n_{\mathrm{h}}}\{h'_{k-1,i}e^{\mathrm{T}}_k\hat{v}_{k-1,i}x^{\mathrm{T}}_{k-1}\tilde{w}_{k-1,i}\}\\
&= \sum_{i=1}^{n_{\mathrm{h}}}\left\{\frac{h'_{k-1,i}}{\tilde{\mu}_{k-1,i}}\tilde{\mu}_{k-1,i}e^{\mathrm{T}}_k\hat{v}_{k-1}x^{\mathrm{T}}_{k-1}\tilde{w}_{k-1,i}\right\}\\
&\leqslant \frac{\lambda}{\lambda_{\min}}\sum_{i=1}^{n_{\mathrm{h}}}\{\tilde{\mu}_{k-1,i}e^{\mathrm{T}}_k\hat{v}_{k-1}x^{\mathrm{T}}_{k-1}\tilde{w}_{k-1,i}\}\\
&= -\frac{\lambda}{\lambda_{\min}}e^{\mathrm{T}}_k\Phi^w_k,
\end{aligned}$$

where $\hat{v}_{k-1,i} \in R^m$, $\hat{v}_{k-1} = [\hat{v}^{\mathrm{T}}_{k-1,1}, \ldots, \hat{v}^{\mathrm{T}}_{k-1,n_{\mathrm{h}}}]^{\mathrm{T}}$ and $\tilde{w}_{k-1,i} = w^*_{k-1,i} - \hat{w}_{k-1,i} \in R^{n_{\mathrm{i}}}$ are weight vector components of the hidden and output layers linked to the $i$th hidden layer neuron, $\lambda \geqslant h'_{k-1,i}$ is the maximum value of the derivative $h'_{k-1,i}$ of the activation function in Eq. (14), and $\lambda_{\min} \neq 0$ is the minimum non-zero value of the parameter $\tilde{\mu}_{k-1,i}$ defined in $\tilde{\Omega}_{k-1}$.

Using the similar way to the proof of Theorem 2, we have

$$\begin{aligned}
&\|\tilde{w}_k\|^2 - \|\tilde{w}_{k-1}\|^2\\
&= -\left\{2a^w_k(M^w_k)^{-1}\frac{e^{\mathrm{T}}_k(\hat{f}^{w-}_{k-1} - \hat{f}^{w+}_{k-1})}{\rho^w_k c^w_k}(r^w_k)^{\mathrm{T}}\tilde{w}_{k-1}\right\}\\
&\quad + \left\|a^w_k(M^w_k)^{-1}\frac{e^{\mathrm{T}}_k(\hat{f}^{w-}_{k-1} - \hat{f}^{w+}_{k-1})}{\rho^w_k c^w_k}(r^w_k)^{\mathrm{T}}\right\|^2\\
&= \sum_{i=1}^{n_{\mathrm{h}}}\{-\mu_{k-1,i}e^{\mathrm{T}}_k\hat{v}_{k-1}x^{\mathrm{T}}_{k-1}\tilde{w}_{k-1,i}\}\\
&\quad \times(\tilde{w}^{\mathrm{T}}_{k-1}(\tilde{w}_{k-1}\tilde{w}^{\mathrm{T}}_{k-1})^{-1}\Delta^w_k r^{w\mathrm{T}}_k\tilde{w}_{k-1})\frac{2a_k(M^w_k)^{-1}}{\rho^w_k}\\
&\quad + \left\|a^w_k(M^w_k)^{-1}\frac{e^{\mathrm{T}}_k(\hat{f}^{w-}_{k-1} - \hat{f}^{w+}_{k-1})}{\rho^w_k c^w_k}(r^w_k)^{\mathrm{T}}\right\|^2
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{n_h}\left\{\frac{\mu_{k-1,i}}{h'_{k-1,i}}(-h'_{k-1,i}e_k^{\mathrm T}\hat{v}_{k-1}x_{k-1}^{\mathrm T}\tilde{w}_{k-1,i})\right\}\\
&\quad\times(\tilde{w}_{k-1}^{\mathrm T}(\tilde{w}_{k-1}\tilde{w}_{k-1}^{\mathrm T})^{-1}\Delta_k^w r_k^{w\mathrm T}\tilde{w}_{k-1})\,\frac{2a_k(M_k^w)^{-1}}{\rho_k^w}\\
&\quad+\left\|a_k^w(M_k^w)^{-1}\,\frac{e_k^{\mathrm T}(\hat{f}_{k-1}^{w-}-\hat{f}_{k-1}^{w+})}{\rho_k^w c_k^w}\,(r_k^w)^{\mathrm T}\right\|^2\\
&\leqslant \frac{\lambda}{\lambda_{\min}}\{|-h'_{k-1,i}e_k^{\mathrm T}\hat{v}_{k-1}x_{k-1}^{\mathrm T}\tilde{w}_{k-1,i}|\}\\
&\quad\times\left|(\tilde{w}_{k-1}^{\mathrm T}(\tilde{w}_{k-1}\tilde{w}_{k-1}^{\mathrm T})^{-1}\Delta_k^w r_k^{w\mathrm T}\tilde{w}_{k-1})\,\frac{2a_k(M_k^w)^{-1}}{\rho_k^w}\right|\\
&\quad+\left\|a_k^w(M_k^w)^{-1}\,\frac{e_k^{\mathrm T}(\hat{f}_{k-1}^{w-}-\hat{f}_{k-1}^{w+})}{\rho_k^w c_k^w}\,(r_k^w)^{\mathrm T}\right\|^2\\
&\leqslant 2a_k^w(M_k^w)^{-1}\left(\frac{\lambda}{\lambda_{\min}}\right)^2\frac{p_w}{\rho_k^w}\sum_{i=1}^{n_h}\{-\mu_{k-1,i}e_k^{\mathrm T}\hat{v}_{k-1}x_{k-1}^{\mathrm T}\tilde{w}_{k-1,i}\}\\
&\quad+\left\|a_k^w(M_k^w)^{-1}\,\frac{e_k^{\mathrm T}(\hat{f}_{k-1}^{w-}-\hat{f}_{k-1}^{w+})}{\rho_k^w c_k^w}(r_k^w)^{\mathrm T}\right\|^2\\
&\leqslant 2a_k^w(M_k^w)^{-1}\left(\frac{\lambda}{\lambda_{\min}}\right)^2\frac{p_w}{\rho_k^w}\,e_k^{\mathrm T}\Phi_k^w\\
&\quad+\|e_k\|^2\|\hat{f}_{k-1}^{w-}-\hat{f}_{k-1}^{w+}\|^2\|r_k^w\|^2\left(\frac{a_k^w(M_k^w)^{-1}}{\rho_k^w c_k^w}\right)^2\\
&= 2a_k^w(M_k^w)^{-1}\left(\frac{\lambda}{\lambda_{\min}}\right)^2\frac{p_w}{\rho_k^w}\,e_k^{\mathrm T}\Phi_k^w\\
&\quad+\|e_k\|^2\|\Delta_k^w\|^2\|r_k^w\|^2\|\hat{\Omega}_{k-1}\|^2\left(\frac{a_k^w(M_k^w)^{-1}}{\rho_k^w}\right)^2.
\end{aligned}
$$

Summing the above equation upon to $N$ steps, we are able to establish the conic condition with a constant $\breve{\sigma}^w$ for the estimate error $e_k$ as the input in Theorem 1 and the output $\Phi_k^w$ similar to last part:

$$
\sum_{k=1}^{N}\left\{e_k\Phi_k^w+\frac{\breve{\sigma}^w}{2}e_k^{\mathrm T}e_k\right\}\geqslant-(\tilde{w}_0)^2\left(\frac{\lambda_{\min}}{\lambda}\right)^2\frac{M_k^w\rho_k^w}{2a_k^w p_w} \tag{69}
$$

by selecting a suitable normalized factor $\rho_k^w$ to obtain the constant number $\breve{\sigma}^w$ such that

$$
1\geqslant\breve{\sigma}^w\geqslant\frac{a_k^w\|\Delta_k^w\|^2\|r_k^w\|^2\|\hat{\Omega}_{k-1}\|^2}{p_w\rho_k^w M_k^w}\left(\frac{\lambda_{\min}}{\lambda}\right)^2. \qquad \square
$$

## 4.2. *Conic section condition for the pruning process in the hidden layer*

Similar to the analysis of the output layer, we can rewrite the estimation error after pruning as

$$
e_k^p=-H_2(\Phi_k^w+\Delta w_{k-1}\hat{f}'_{k-1,w})+\tilde{e}_k^w \tag{70}
$$

with $\Delta w_{k-1}=-((w_{k-1,i})/[\sum_{m=k-L}^{k-1}(\bar{\Omega}_{m,i,i})]^{-1})[\sum_{m=k-L}^{k-1}(\bar{\Omega}_m)]^{-1}$ $I_{k-1,i}$ where $I$ is defined similar to the one in last section and $\bar{\Omega}_{k-1}$ is the per-iteration estimation of the Hessian matrix of the hidden layer, details of which is discussed next.

Note $\hat{f}'_{k-1,w}$ is the first derivative of the network output with respect to the weight value of the hidden layer and we have proven that the weight of the output layer is a bounded term, so $\|\hat{f}'_{k-1,w}\|\leqslant\lambda\cdot v_{\max}$, where $\|\hat{v}_k\|\leqslant v_{\max}$. And taken the similar analysis of the pruning for output layer, we know that $\Delta w_{k-1}$ is also a bounded term since it is related to the current weight value of the hidden layer. Thus we can rewrite the $e_k^p$ like

$$
\begin{aligned}
e_k^p&=-H_2(\Phi_k^w-\Delta w_{k-1}\hat{f}'_{k-1,w})+\tilde{e}_k^w\\
&=-H_2(\Phi_k^w-\tau_k^w)+\tilde{e}_k^w\\
&=-H_2\Phi_k^{pw}+\tilde{e}_k^w,
\end{aligned} \tag{71}
$$

where $\tau_k^w=\Delta w_{k-1}\cdot\hat{f}'_{k-1,w}$, and $\|\tau_k^w\|\leqslant\tau_{\max}^w$.

So we have the learning law after pruning as

$$
\hat{w}_k=\hat{w}_{k-1}-\frac{a_k(M_k^w)^{-1}e_k^{p\mathrm T}(\hat{f}_{k-1}^{w-}-\hat{f}_{k-1}^{w+})}{c_k^w\rho_k^w}\,r_k^w. \tag{72}
$$

After reconstructing a new feedback system by using the input $e_k^p$ and output $\Phi_k^{pw}$, we can justify the conic sector condition as follows:

$$
\begin{aligned}
&\|\tilde{w}_k\|^2-\|\tilde{w}_{k-1}\|^2\\
&\quad=-\left\{2a_k^w(M_k^w)^{-1}\,\frac{e_k^{p\mathrm T}(\hat{f}_{k-1}^{w-}-\hat{f}_{k-1}^{w+})}{\rho_k^w c_k^w}\,(r_k^w)^{\mathrm T}\tilde{w}_{k-1}\right\}\\
&\qquad+\left\|a_k^w(M_k^w)^{-1}\,\frac{e_k^{p\mathrm T}(\hat{f}_{k-1}^{w-}-\hat{f}_{k-1}^{w+})}{\rho_k^w c_k^w}\,(r_k^w)^{\mathrm T}\right\|^2\\
&\quad=\sum_{i=1}^{n_h}\{-\mu_{k-1,i}e_k^{p\mathrm T}\hat{v}_{k-1}x_{k-1}^{\mathrm T}\tilde{w}_{k-1,i}\}\\
&\qquad\times(\tilde{w}_{k-1}^{\mathrm T}(\tilde{w}_{k-1}\tilde{w}_{k-1}^{\mathrm T})^{-1}\Delta_k^w r_k^{w\mathrm T}\tilde{w}_{k-1})\,\frac{2a_k(M_k^w)^{-1}}{\rho_k^w}\\
&\qquad+\left\|a_k^w(M_k^w)^{-1}\frac{e_k^{p\mathrm T}(\hat{f}_{k-1}^{w-}-\hat{f}_{k-1}^{w+})}{\rho_k^w c_k^w}(r_k^w)^{\mathrm T}\right\|^2\\
&\quad=\sum_{i=1}^{n_h}\left\{\frac{\mu_{k-1,i}}{h'_{k-1,i}}(-h'_{k-1,i}e_k^{p\mathrm T}\hat{v}_{k-1}x_{k-1}^{\mathrm T}\tilde{w}_{k-1,i})\right\}\\
&\qquad\times(\tilde{w}_{k-1}^{\mathrm T}(\tilde{w}_{k-1}\tilde{w}_{k-1}^{\mathrm T})^{-1}\Delta_k^w r_k^{w\mathrm T}\tilde{w}_{k-1})\,\frac{2a_k(M_k^w)^{-1}}{\rho_k^w}\\
&\qquad+\left\|a_k^w(M_k^w)^{-1}\frac{e_k^{p\mathrm T}(\hat{f}_{k-1}^{w-}-\hat{f}_{k-1}^{w+})}{\rho_k^w c_k^w}\,(r_k^w)^{\mathrm T}\right\|^2\\
&\quad\leqslant\frac{\lambda}{\lambda_{\min}}\{|-h'_{k-1,i}e_k^{p\mathrm T}\hat{v}_{k-1}x_{k-1}^{\mathrm T}\tilde{w}_{k-1,i}|\}\\
&\qquad\times\left|(\tilde{w}_{k-1}^{\mathrm T}(\tilde{w}_{k-1}\tilde{w}_{k-1}^{\mathrm T})^{-1}\Delta_k^w r_k^{w\mathrm T}\tilde{w}_{k-1})\,\frac{2a_k(M_k^w)^{-1}}{\rho_k^w}\right|\\
&\qquad+\left\|a_k^w(M_k^w)^{-1}\frac{e_k^{p\mathrm T}(\hat{f}_{k-1}^{w-}-\hat{f}_{k-1}^{w+})}{\rho_k^w c_k^w}\,(r_k^w)^{\mathrm T}\right\|^2\\
&\quad\leqslant 2a_k^w(M_k^w)^{-1}\left(\frac{\lambda}{\lambda_{\min}}\right)^2\frac{p_w}{\rho_k^w}\sum_{i=1}^{n_h}\{-\mu_{k-1,i}e_k^{p\mathrm T}\hat{v}_{k-1}x_{k-1}^{\mathrm T}\tilde{w}_{k-1,i}\}
\end{aligned}
$$

$$
\begin{aligned}
&+\left\| a_k^w (M_k^w)^{-1} \frac{e_k^{p\mathrm{T}}(\hat{f}_{k-1}^{w-} - \hat{f}_{k-1}^{w+})}{\rho_k^w c_k^w} (r_k^w)^{\mathrm{T}} \right\|^2 \\
&\leqslant 2a_k^w (M_k^w)^{-1} \left(\frac{\lambda}{\lambda_{\min}}\right)^2 \frac{p_w}{\rho_k^w} e_k^{p\mathrm{T}} \Phi_k^w \\
&\quad + \|e_k\|^2 \|\hat{f}_{k-1}^{w-} - \hat{f}_{k-1}^{w+}\|^2 \|r_k^w\|^2 \left(\frac{a_k^w (M_k^w)^{-1}}{\rho_k^w c_k^w}\right)^2 \\
&= 2a_k^w (M_k^w)^{-1} \left(\frac{\lambda}{\lambda_{\min}}\right)^2 \frac{p_w}{\rho_k^w} e_k^{p\mathrm{T}} \Phi_k^w \\
&\quad + \|e_k\|^2 \|\Delta_k^w\|^2 \|r_k^w\|^2 \|\hat{\Omega}_{k-1}\|^2 \left(\frac{a_k^w (M_k^w)^{-1}}{\rho_k^w}\right)^2 \\
&= 2a_k^w (M_k^w)^{-1} \left(\frac{\lambda}{\lambda_{\min}}\right)^2 \frac{p_w}{\rho_k^w} e_k^{p\mathrm{T}} (\Phi_k^{pw} + \tau_k^w) \\
&\quad + \|e_k\|^2 \|\Delta_k^w\|^2 \|r_k^w\|^2 \|\hat{\Omega}_{k-1}\|^2 \left(\frac{a_k^w (M_k^w)^{-1}}{\rho_k^w}\right)^2 \\
&\leqslant 2a_k^w (M_k^w)^{-1} \left(\frac{\lambda}{\lambda_{\min}}\right)^2 \frac{p_w}{\rho_k^w} e_k^{p\mathrm{T}} \Phi_k^{pw} \\
&\quad + a_k^w (M_k^w)^{-1} \left(\frac{\lambda}{\lambda_{\min}}\right)^2 \frac{p_w}{\rho_k^w} \left(\frac{1}{2}\|e_k^p\|^2 + 2\|\tau_k^w\|^2\right) \\
&\quad + \|e_k\|^2 \|\Delta_k^w\|^2 \|r_k^w\|^2 \|\hat{\Omega}_{k-1}\|^2 \left(\frac{a_k^w (M_k^w)^{-1}}{\rho_k^w}\right)^2 \\
&\leqslant 2a_k^w (M_k^w)^{-1} \left(\frac{\lambda}{\lambda_{\min}}\right)^2 \frac{p_w}{\rho_k^w} e_k^{p\mathrm{T}} \Phi_k^{pw} \\
&\quad + a_k^w (M_k^w)^{-1} \left(\frac{\lambda}{\lambda_{\min}}\right)^2 \frac{p_w}{\rho_k^w} \left(\frac{1}{2}\|e_k^p\|^2 + 2\|\tau_{\max}^w\|^2\right) \\
&\quad + \|e_k\|^2 \|\Delta_k^w\|^2 \|r_k^w\|^2 \|\hat{\Omega}_{k-1}\|^2 \left(\frac{a_k^w (M_k^w)^{-1}}{\rho_k^w}\right)^2.
\end{aligned}
$$

Summing the above equation upon to $N$ steps, we are able to establish the conic condition with a constant $\breve{\sigma}^w$ for the estimate error $e_k$ as the input in Theorem 1 and the output $\Phi_k^w$ similar to the output layer:

$$
\begin{aligned}
&\sum_{k=1}^{N} \left\{ e_k^p \Phi_k^{pw} + \frac{\breve{\sigma}^w}{2} e_k^{p\mathrm{T}} e_k^p \right\} \\
&\geqslant -\left[ (\tilde{w}_0)^2 \left(\frac{\lambda_{\min}}{\lambda}\right)^2 \frac{M_k^w \rho_k^w}{2a_k^w p_w} + N\|\tau_{\max}^w\|^2 \right]
\end{aligned} \tag{73}
$$

by selecting a suitable normalized factor $\rho_k^w$ to obtain the constant number $\breve{\sigma}^w$ such that

$$
1 \geqslant \breve{\sigma}^w \geqslant \frac{1}{4} + \frac{a_k^w \|\Delta_k^w\|^2 \|r_k^w\|^2 \|\hat{\Omega}_{k-1}\|^2}{p_w \rho_k^w M_k^w} \left(\frac{\lambda_{\min}}{\lambda}\right)^2.
$$

Although the conic conditions for the learning and pruning of both layers are obtained between the estimation error $e_k^p$ and the output, the result can be easily extended to the tracking error $s_k$ as following steps. Taking the output layer as an example,

**Theorem 4.** *If the discrete time signal $e_k$ and $s_k$ in the Fig. 1 satisfy*

(a) $e_k \in L_2$
(b) $e_k = (1 - k_v z^{-1}) s_k$

*with $\|k_v\| \leqslant 1$ then the above feedback system is stable with $s_{ks} \in L_2$.*

**Proof.** See Corollary 1 [25]. □

## 5. Simulation results

Consider a discrete-time single-link manipulator [25]

$$
\begin{aligned}
y(k+2) &= 2(1-T)y(k+1) + (2T-1)y(k) \\
&\quad + 10T^2 \sin y(k) + u(k) + d(k),
\end{aligned} \tag{74}
$$

where $y$ is the tracking position signal, $u$ is the control signal $T = 0.002\,\mathrm{s}$ is the sampling time, and $d$ is the disturbance generated from a normally distributed random number with the bound $\|d(k)\| \leqslant d_m = 0.2$.

And it can be converted to the form as follows:

$$
\begin{aligned}
x_1(k+1) &= x_2(k), \\
x_2(k+1) &= f(x(k)) + u(k) + d(k),
\end{aligned}
$$

where $x(k) = [x_1(k), x_2(k)]^{\mathrm{T}}$

$$
f(x(k)) = (2T-1)x_1(k) + (2-2T)x_2(k) + 10T^2 \sin x_1(k) \tag{75}
$$

and $x_2(k)$ represents the actual trajectory. The tracking error is defined as

$$
r(k) = e_n(k) = x_2(k) - x_{2d}(k), \tag{76}
$$

where $x_{2d}$ is the desired trajectory signal.

The control signal is defined as

$$
u(k) = x_{2d}(k) - \hat{f}(k) + k_v s(k), \tag{77}
$$

where $\hat{f}(k)$ is the output of a three-layered NN, with three input neurons, 100 hidden layer neurons initially and one output neuron, to estimate $f(x(k))$.

First, the feed-forward neural network with 100 hidden neurons is initially used. By the definition of the perturbation, $\Delta_k$ is generated as a vector with 100 components satisfying some regularity conditions (e.g., $\Delta_k$ being a vector of independent Bernoulli $\pm 1$ random variables satisfies these conditions). Once the perturbation is decided, the gradient approximation of the output layer can be obtained by Eq. (31), and in turn, the Hessian matrix for the output layer using Eq. (34). Similarly, Eqs. (59) and (67) can be applied to update the parameters in the hidden layer through Eqs. (35) and (68).

Fig. 4 shows the output of the plant using the standard ASP algorithm without pruning using the command signal $x_{2d} = \sin(\pi/5)kT$ and Fig. 5 shows the result by using DTP algorithm, in which the overfitting has been removed.

And the similar results can be obtained when the command signal is switched to square signal. Fig. 6 shows the system output by using the standard ASP algorithm

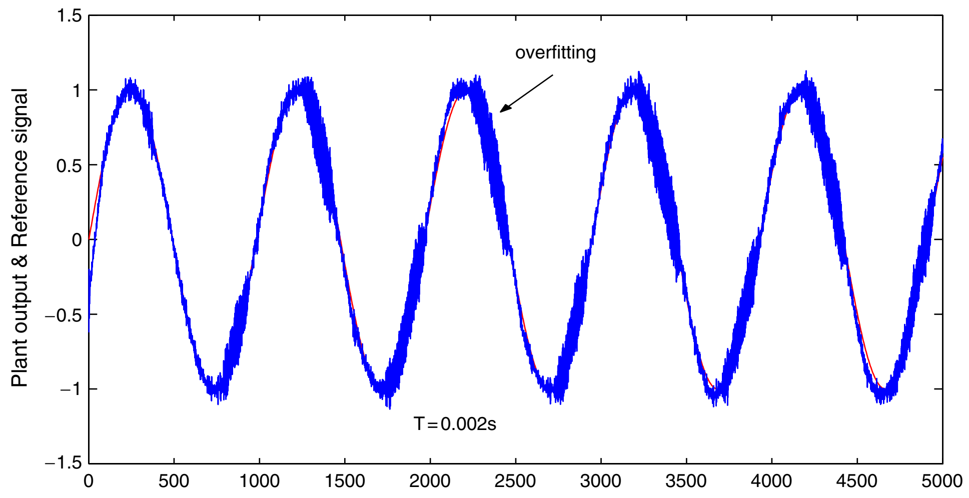

Fig. 4. Output $y_k$ and reference signal using the standard ASP based neural controller with 100 neurons.

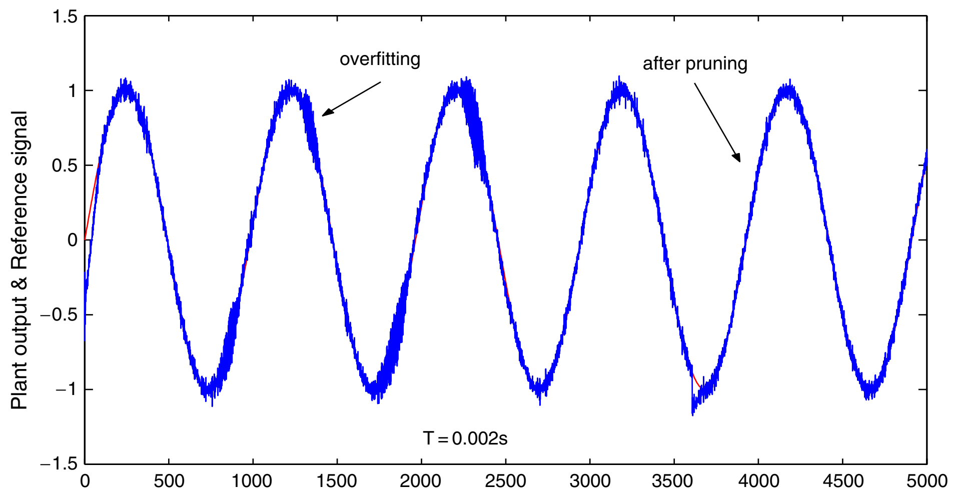

Fig. 5. Output $y_k$ and reference signal using the DTP based neural controller.

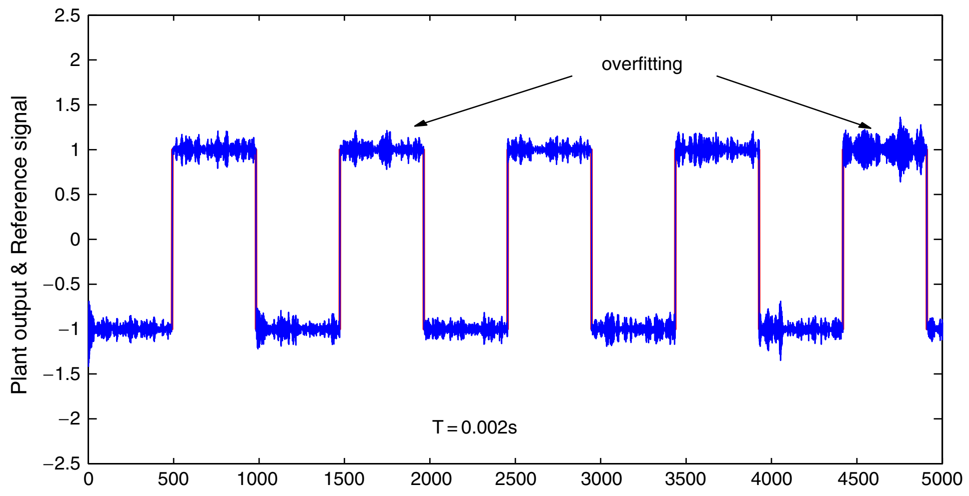

Fig. 6. Output $y_k$ and reference signal using the standard ASP based neural controller with 100 neurons.

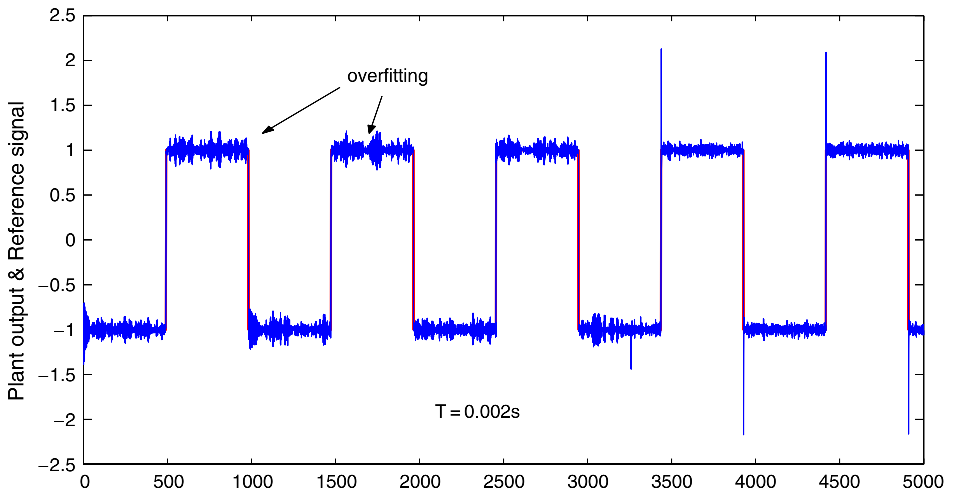

Fig. 7. Output $y_k$ and reference signal using the DTP based neural controller.

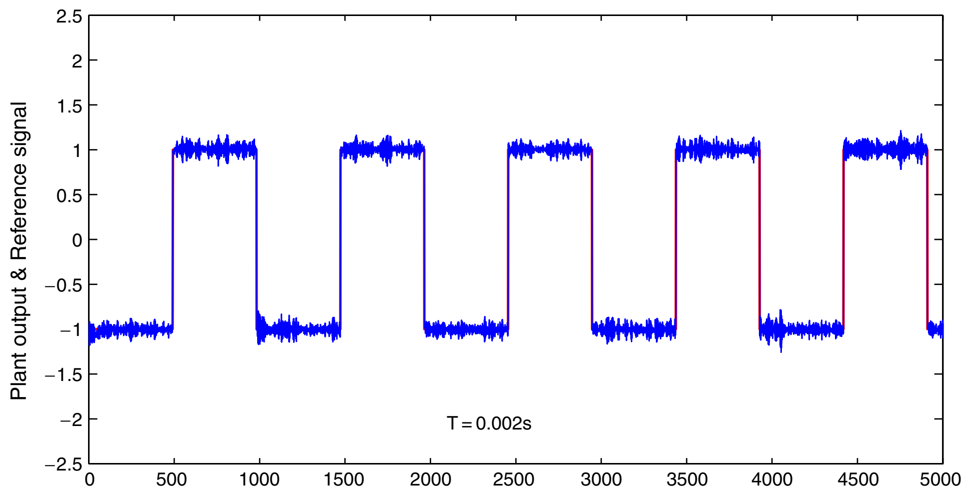

Fig. 8. Output $y_k$ and reference signal using the standard BP based neural controller.

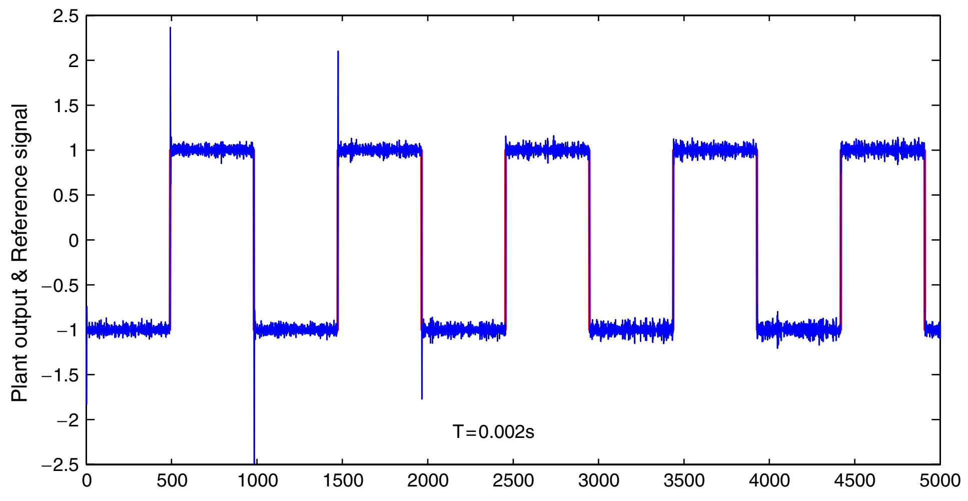

Fig. 9. Output $y_k$ and reference signal using the DTP based neural controller.

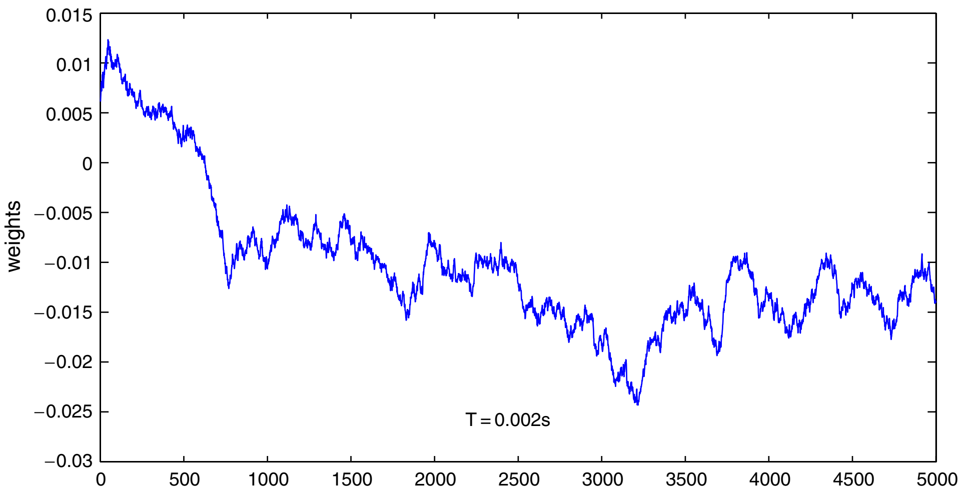

Fig. 10. Estimated parameter $\hat{w}_{k,1}$ of the hidden layer using the DTP based algorithm.

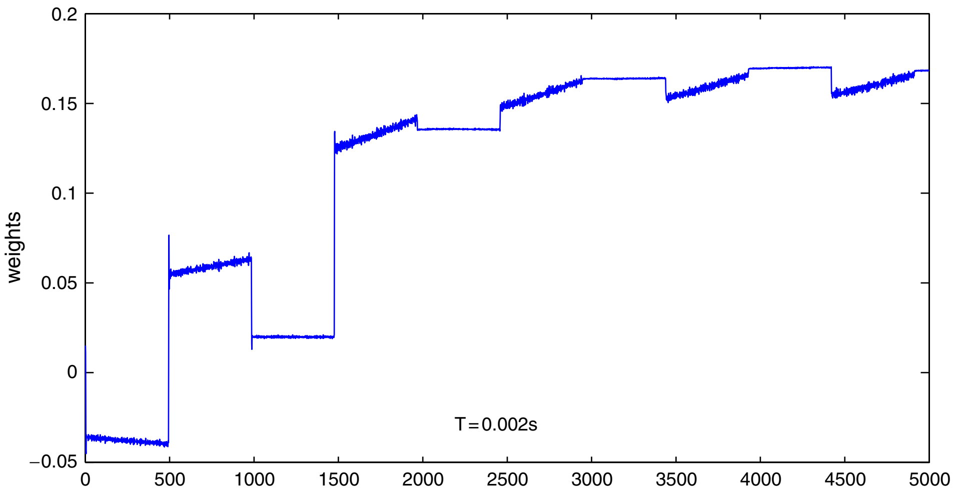

Fig. 11. Estimated parameter $\hat{w}_{k,1}$ of the hidden layer using the standard BP based algorithm.

without pruning, in which the overfitting may not be avoided. And Fig. 7 shows the result by DTP algorithm, where the overfitting has been removed.

When the criterion for pruning is not satisfied, which means pruning is not necessary, the system with DTP algorithm still performs better than the standard BP algorithm. Fig. 9 shows the system output with DTP based controller and Fig. 8 shows the result with standard BP algorithm by using the same number neurons in the hidden layer. And it can be explained by the reason of weight drifting, which can be illustrated in Figs. 11 and 10.

## 6. Conclusion

The DTP based pruning method for neural controller has been developed to obtain the guaranteed stability with improved generalization ability. A complete stability analysis is performed for this closed-loop control system. Simulation results show that the proposed neural controller performs better than a neural controller based on the standard back-propagation algorithm or standard ASP algorithm without pruning in case of overfitting.

## References

[1] E.B. Baum, D. Haussler, What size gives valide generalization, Neural Comput. 1 (1989) 151–160.
[2] J.F.D. Canete, A. Barreiro, A.G. Cerezo, I.G. Moral, An input–output based stabilization criterion for neural-network control of nonlinear systems, IEEE Trans. Neural Networks 12 (2001) 1491–1497.
[3] V.R. Cluett, L. Shah, G. Fisher, Robustness analysis of discrete-time adaptive control systems using input–output stability theory: a tutorial, IEE Proc. Part D 135 (1988) 133–141.
[4] G.H. Colub, C.F.V. Loan, Matrix Computation, North Oxford Academic, Oxford, UK, 1983.

[5] H. Daniel Patino, R. Carelli, B.R. Kuchen, Neural networks for advanced control of robot manipulators, IEEE Trans. Neural Networks 13 (2002) 343–354.
[6] J. Denker, D. Schwartz, B. Wittner, S. Solla, R. Howard, Large automatic learning, rule extraction, and generalization, Complex Syst. 1 (1987) 877–922.
[7] S.S. Ge, C. Wang, Direct adaptive NN control of a class of nonlinear systems, IEEE Trans. Neural Networks 13 (2002) 214–221.
[8] B. Hassibi, D.G. Stork, Second-order derivatives for network pruning: optimal brain surgeon, Advances in Neural Information Processing Systems, 1993, pp. 164–171.
[9] S. Haykin, Neural Network—A Comprehensive Foundation, Prentice-Hall, Englewood Cliffs, NJ, 1999.
[10] S. Jagannathan, Control of a class of nonlinear discrete-time systems using multilayer neural networks, IEEE Trans. Neural Networks 12 (2001) 1113–1120.
[11] G.N. Karystinos, D.A. Pados, On overfitting, generalization, and randomly expanded training sets, IEEE Trans. Neural Networks 11 (2000) 1050–1057.
[12] Y. Le Cun, Generalization and network design strategies, Connectionism in Perspective, 1989, pp. 143–155.
[13] Y. Le Cun, J.S. Denker, S.A. Solla, Optimal brain damage, Advances in Neural Information Processing Systems 2, 1993, pp. 598–605.
[14] M. Lee, H.-S. Choi, A robust neural controller for underwater robot manipulators, IEEE Trans. Neural Networks 11 (2000) 1465–1470.
[15] F.L. Lewis, C.T. Abdallah, D.M. Dawson, Control of Robot Manipulators, MacMillan, New York, 1993.
[16] J. Ni, Q. Song, Adaptive simultaneous perturbation based pruning for a class of nonlinear control systems, Neural Networks, submitted for publication.
[17] R. Ortega, L. Praly, I.D. Laudau, Robustness of discrete-time direct adaptive controllers, IEEE Trans. Automat. Control 30 (1985) 1179–1187.
[18] M.M. Polycarpou, P.A. Ioannou, Learning and convergence analysis of neural-type structured networks, IEEE Trans. Neural Networks 3 (1992) 39–50.
[19] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by backpropagation errors, Nature 323 (1986) 533–536.
[20] M.G. Safanov, Stability and Robustness of Multivariable Feedback Systems, MIT Press, Cambridge, MA, 1980.
[21] Q. Song, W.J. Hu, L. Yin, Y.C. Soh, Robust adaptive dead zone technology for fault-tolerant control of robot manipulators, J. Intell. Robotic Syst. 33 (2002) 113–137.
[22] Q. Song, W.J. Hu, T.N. Zhao, Robust neural controller for variable airflow volume system, IEE Proc. Control Theory Appl. 150 (2003) 112–118.
[23] Q. Song, R. Katebi, M.J. Grimble, Robust multivariable implicit adaptive control, IMA J. Math. Control Inf. 10 (1993) 49–70.
[24] Q. Song, J.C. Spall, Y.C. Soh, Robust neural network tracking controller using simultaneous perturbation stochastic approximation, Proceedings of the IEEE Conference on Decision and Control, 9–12 December 2003, Maui, Hawaii, pp. 6194–6199.
[25] Q. Song, J. Xiao, Y.C. Soh, Robust backpropagation training algorithm for multilayered neural tracking controller, IEEE Trans. Neural Networks 10 (1999) 1133–1141.
[26] J.C. Spall, Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, IEEE Trans. Automat. Control 37 (1992) 332–341.
[27] J.C. Spall, Adaptive stochastic approximation by the simultaneous perturbation method, IEEE Trans. Automat. Control 45 (2000) 1839–1853.
[28] J.C. Spall, J.A. Cristion, A neural network controller for systems with unmodeled dynamics with applications to water plant treatment, IEEE Trans. Syst. Man Cybern. Part B 27 (1997) 369–375.
[29] J.C. Spall, J.A. Cristion, Model free control of nonlinear stochastic systems with discrete-time measurements, IEEE Trans. Automat. Control 43 (1998) 1198–1210.
[30] F. Sun, Z. Sun, P. Woo, Stable neural-network-based adaptive control for sampled-data nonlinear systems, IEEE Trans. Neural Networks 9 (1998) 956–968.
[31] I.J. Wang, E.K.P. Chong, A deterministic analysis of stochastic approximation with randomized directions, IEEE Trans. Automat. Control 43 (1998) 1745–1749.

**Jie Ni** (IEEE Student Member 41604367) received the B.E. from Huazhong University of Science & Technology, Hubei, PR China in 2002. He is pursuing the Ph.D. degree in the school of Electrical & Electronic Engineering, Nanyang Technological University in Singapore under Singapore government scholarship. His research interests include Artificial Intelligent System, Neural Networks, Adaptive Control, Robust Control, Machine Learning, etc.

**Qing Song** (IEEE Member 00116020) received the B.S. and the M.S. degrees from Harbin Shipbuilding Engineering Institute and Dalian Maritime University, China in 1982 and 1986, respectively. He obtained the Ph.D. degree from the Industrial Control Centre at Strathclyde University, UK in 1992. He was an assistant professor at the Dalian Maritime University from 1986 to 1987. He worked as a research engineer at the System and Control Pte Ltd, UK, before joined the School of Electrical and Electronic Engineering, Nanyang Technological University as a lecturer in 1992. He is now an associate professor.

Dr. Qing Song is the first inventor of a neural network related patent and was the recipient of a few of Academic Exchange Fellowships with a number of referred international journal publications. He is also an active industrial consultant. He has been the principal investigator of a few research programs targeted to industry in the area of computational intelligence.