

# Stable Neuro-Flight-Controller Using Fully Tuned Radial Basis Function Neural Network

Li Yan, Narasimhan Sundararajan<sup>†</sup>, Paramasivan Saratchandran

School of Electrical & Electronic Engineering

Nanyang Technological University, Singapore

## Abstract

In this paper, a flight control scheme in which a Radial Basis Function Network (RBFN) aids a conventional controller has been developed. The RBFN controller, consisting of variable Gaussian functions, uses only on-line learning to represent the local inverse dynamics of the aircraft system. Using Lyapunov synthesis approach, a tuning rule for updating all the parameters of the RBFN (including centers, widths as well as the weights of the output layer) is derived, which extends Kawato's strategy in which only the weights were adaptable. The proposed tuning rule guarantees the stability of the overall system with improved tracking performance. Simulation studies using a F8 aircraft longitudinal model illustrate the superior performance of the proposed scheme over the earlier scheme of Sadhukan and Feteih. The simulation studies further indicate that the results can be extended to a dynamic RBFN in which the hidden neurons can be added/pruned, thus producing a more compact network structure.

**Keywords:** Aircraft, Flight control, Neuro-flight-controller, Radial basis function network (RBFN), Feedback-error-learning, Lyapunov stability theory, Growing and pruning (GAP)

---

<sup>†</sup> is the corresponding author. Mailing address: Block S2, School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore 639798. Email: ENSUNDARA@ntu.edu.sg

# 1 Introduction

Due to their powerful ability of approximating nonlinear functions, control laws and design methods incorporating artificial neural networks have been extensively studied. In the area of flight control, Calise *et al.* has summarized some current research efforts of using Neural Networks (NN) for flight control system design [1]. It has been shown that neural networks with on-line learning can adapt to aircraft dynamics which is poorly known or rapidly changing. NN with on-line learning capability for flight control applications was reported in [2], where Sadhukan and Feteih presented an exact inverse neuro controller with full state feedback for a F8 aircraft. The objective of the controller in an autopilot mode was to closely track the pilot's throttle and pitch rate commands. Napolitano *et al.* [3] studied the performance of various NN based designs of autopilot systems for implementing a variety of flight control functions. In most of these applications, feedforward network with Back Propagation (BP) learning algorithm or its extensions has been the main paradigm. It is known that using feedforward network and the BP algorithm, problems arise with local minima and saddle points, and the algorithm itself has a very slow convergence rate. Recently, results on robust nonlinear adaptive flight controllers have been developed by Calise and his coworkers in [4][5], where stability proofs for the proposed neural (MLP neural networks) flight controllers have been presented. Further in [6], limitations of using neural networks in real world adaptive control applications are analyzed and further needed research work have been pointed out.

Since the early nineties, RBFNs with Gaussian function have been widely used as the basic structure of neural networks in nonlinear control [7][8], due to its good global generalization ability and simple network structure that avoids unnecessary and lengthy calculations [9]. For on-line implementation, Sanner and Slotine [10] developed a direct adaptive tracking control

architecture using Gaussian RBFN to adaptively compensate for plant nonlinearities. Gomi and Kawato [7] proposed a “feedback-error-learning” control strategy, where a Gaussian RBFN is used for on-line learning the inverse dynamics of the system. Recently, Polycarpou *et.al.* has also presented a control design approach for a class of nonlinear plants, where an adaptive bounding technique is employed to handle the unknown network reconstruction error [11][12]. In these schemes, the adaptive tuning rules are derived based on Lyapunov synthesis approach and guarantee the closed-loop stability. However, in aircraft flight control, there are only limited papers which explore the application of RBFN. In [13], Singh *et al.* used a RBFN to suppress wing rock for a slender delta wing configuration. Calise presented the use of RBFN to capture variations in Mach number in [14], as these variations are difficult to be represented by polynomial functions in the transonic region. In all the above mentioned applications, when implementing the network structure, the number of hidden units, centers and widths are always fixed, and then the weights of the network are estimated. In practice, it is difficult to choose the centers and widths appropriately, especially for on-line implementation when the dynamics may be changing. The use of inaccurate centers usually results in the deterioration of the performance. In order to overcome this problem, a large number of hidden neurons need to be selected and this in turn results in slow on-line learning [15][16].

In contrast to the conventional approaches, recently, fully tuned RBFNs have begun to exhibit their potential for accurate approximation. In a fully tuned RBFN, not only the weights of the output layer are updated, but also the other parameters of the network (like the centers and widths) so that the nonlinearities of the dynamic system can be captured as quickly as possible [15][17]. It is in this context that this paper attempts to explore the use of the fully tuned RBFN for aircraft control applications. A simple control architecture originating from Kawato’s feedback-error-learning scheme [7], incorporating a fully tuned RBFN controller is

presented. Using the Lyapunov synthesis approach, a stable tuning rule for updating all the parameters of the RBFN are derived and this rule guarantees the local stability of the overall system along the desired trajectory. In addition, the robustness of the proposed tuning law in the presence of the neural network approximation error as well as the system model error is also analyzed.

In order to evaluate the proposed scheme with an already existing neuro control scheme, the linearized F8 longitudinal aircraft model of [2] is utilized for simulation studies and the results demonstrate the better performance of the proposed method. Also, a dynamic RBFN with Growing And Pruning (GAP) strategy is investigated and the results indicate that with the GAP strategy, the network structure is more compact.

The paper is organized as follows: Section 2 gives a brief description of the problem and section 3 derives the stable adaptive tuning rule for a fully tuned RBFN, where the stability of the overall system is guaranteed. A robustness analysis of the proposed scheme to the approximation error and model error is also presented. Section 4 presents a comparison study of using a fully tuned RBFN and the RBFN with only tuning of the weights based on a F8 longitudinal aircraft model. Section 5 discusses the dynamic structured RBFN and presents simulation results with a comparison to earlier obtained performance. Section 6 summarizes the conclusions from this study.

## 2 Problem Formulation

The aircraft dynamics is represented by a bounded input, bounded output (BIBO) continuous system,

$$\Sigma : \quad \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \quad (1)$$

In Eq.1, smoothness of  $\mathbf{f}()$  is assumed.  $\mathbf{x}$  is a  $n \times 1$  state vector,  $\mathbf{u}$  is  $p \times 1$  control vector. It is assumed that  $\mathbf{f}(0, 0) = 0$ , so that the origin is an equilibrium state. In general, the number of the control inputs is less than that of the states ( $p < n$ ). It is known that under this condition only the  $p$  states can be tracked perfectly. Partitioning  $\mathbf{x}$  into  $\mathbf{x}_t$  and  $\mathbf{x}_r$ , the aircraft dynamics can be written as,

$$\Sigma : \quad \begin{pmatrix} \dot{\mathbf{x}}_t \\ \dot{\mathbf{x}}_r \end{pmatrix} = \begin{pmatrix} \mathbf{f}_t(\mathbf{x}, \mathbf{u}) \\ \mathbf{f}_r(\mathbf{x}, \mathbf{u}) \end{pmatrix} \quad (2)$$

The problem studied in this paper is to design a neuro controller such that the given  $p$  states  $\mathbf{x}_t \in \mathbf{x}$  of the aircraft can track the desired commands  $\mathbf{x}_{dt}$  accurately, while the other states  $\mathbf{x}_r \in \mathbf{x}$  asymptotically approach the equilibrium point  $\mathbf{x}_{dr}$ . Determination of the conditions under which a solution  $\mathbf{u}_d(t)$  exists forms part of the theoretical control problem. Based on the earlier studies in [18], to accomplish the above control target, the system should satisfy Assumption 1.

**Assumption 1:**  $\mathbf{f}_t(\mathbf{x}, \mathbf{u})$  has continuous bounded partial derivative in a certain neighborhood of all the points along the desired trajectory and the matrix  $\frac{\partial \mathbf{f}_t(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}^T}$  is nonsingular.

It follows from the Implicit Function Theorem the desired  $\mathbf{u}_d(t)$  can be expressed as,

$$\mathbf{u}_d(t) = \bar{\mathbf{f}}_t(\mathbf{x}_d, \mathbf{x}_{dt}) \quad (3)$$

where  $\bar{\mathbf{f}}_t$ , a  $p \times 1$  smooth function, is the inverse function of  $\mathbf{f}_t$ , and  $\mathbf{x}_d = [\mathbf{x}_{dt}^T, \mathbf{x}_{dr}^T]^T$ .

In the next section a RBFN is chosen to approximate Eq.3, and the main contribution of this paper is using Lyapunov stability theory to derive the tuning rule for updating all the parameters of the RBFN.

### 3 Fully Tuned RBFN Controller

The tuning law for adapting the weights of the RBFN has been derived in earlier papers [7][10]. In this paper, the updating rule for a fully tuned RBFN controller is derived based on the feedback-error-learning strategy. The robustness of the proposed algorithm is also analyzed.

#### 3.1 Control Strategy

A simple control strategy, which is similar to the well-known “feedback-error-learning” scheme, is proposed in Fig.1. This scheme is adopted in this study because it has the advantage of generating the desired input signal without requiring that the network be trained initially off-line. Moreover, the outputs of the neuro controller  $\mathbf{u}_{nn}$ , which has fault tolerant ability through on-line learning, only depend on the desired input trajectory profile.

Assuming all the states of the system are accessible, the tracking error  $\mathbf{e}$  is defined as  $\mathbf{e} = \mathbf{x} - \mathbf{x}_d$ . The error dynamics for the overall system is,

$$\dot{\mathbf{e}} = \dot{\mathbf{x}} - \dot{\mathbf{x}}_d = \mathbf{f}(\mathbf{x}, \mathbf{u}) - \mathbf{f}(\mathbf{x}_d, \mathbf{u}_d) \quad (4)$$

Since only the local stability of the system is considered in this paper, Eq. (4) can be expanded

along the desired trajectory using Taylor series. Neglecting all the higher order terms,

$$\dot{\mathbf{e}} = \left. \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}^T} \right|_{\mathbf{x}=\mathbf{x}_d, \mathbf{u}=\mathbf{u}_d} (\mathbf{x} - \mathbf{x}_d) + O(\mathbf{x} - \mathbf{x}_d) + \left. \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}^T} \right|_{\mathbf{x}=\mathbf{x}_d, \mathbf{u}=\mathbf{u}_d} (\mathbf{u} - \mathbf{u}_d) + O(\mathbf{u} - \mathbf{u}_d) \quad (5)$$

where  $O()$  represents higher order terms. Substituting  $\left. \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}^T} \right|_{\mathbf{x}=\mathbf{x}_d, \mathbf{u}=\mathbf{u}_d}$  by  $A(t)$ , and  $\left. \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}^T} \right|_{\mathbf{x}=\mathbf{x}_d, \mathbf{u}=\mathbf{u}_d}$  by  $B(t)$ , neglecting all the higher order terms,

$$\dot{\mathbf{e}} \simeq A(t)(\mathbf{x} - \mathbf{x}_d) + B(t)(\mathbf{u} - \mathbf{u}_d) \quad (6)$$

With the widely used technology of the Control Configured Vehicle (CCV), the basic airframe may be designed to have a low or even negative static stability in certain flight regimes, the augmented stability is then implemented by considering the controller's dynamics. In this study, a conventional controller is used in the strategy to improve the stability and response of the closed-loop system. This conventional controller can be realized by any approach like pole-placement, Linear Quadratic Regulator (LQR), or  $H_\infty$  controller, etc. To illustrate the concept, a proportional controller is designed satisfying Assumption 2.

**Assumption 2:** The closed loop dynamic system, whose feedback controller is designed based on the nominal aircraft model, is stable along the desired flight trajectory.

With only the linear proportional controller  $\mathbf{u} = K_p \mathbf{e}$ , the error dynamics is,

$$\dot{\mathbf{e}} = (A(t) + B(t)K_p)\mathbf{e} - B(t)\mathbf{u}_d \quad (7)$$

Although the system is stable based on Assumption 2, the tracking performance deteriorates greatly because of the existence of  $B(t)\mathbf{u}_d$ . Therefore, a RBFN controller is used for on-line learning of  $\mathbf{u}_d$ . Thus the total control signal to the aircraft is the sum of the proportional controller and the RBFN controller signals,

$$\mathbf{u} = K_p \times \mathbf{e} + \mathbf{u}_{nn} \quad (8)$$

### 3.2 RBFN Parameterization

Setting the RBFN's inputs as  $\zeta$  where  $\zeta = [\mathbf{x}_d^T, \dot{\mathbf{x}}_d^T]^T$ ,  $\mathbf{u}_d$  can be approximated by a RBFN through on-line learning,

$$\begin{aligned} \mathbf{u}_d &= \mathbf{u}_{nn}^* + \epsilon_h \\ &= \sum_{k=1}^h \mathbf{w}_k^* \exp\left(-\frac{1}{\sigma_k^{*2}} \|\zeta - \mu_k^*\|^2\right) + \epsilon_h = W^{*T} \phi(\mu^*, \sigma^*, \zeta) + \epsilon_h \end{aligned} \quad (9)$$

where  $W^*$  is  $h \times p$  optimal weight matrix ( $h$  indicates the number of hidden neurons) and  $\phi$  is  $h \times 1$  Gaussian function, which is determined by the optimal centers  $\mu^*$  and widths  $\sigma^*$ , for simplicity  $\phi^*$  is used instead of  $\phi(\mu^*, \sigma^*, \zeta)$  hereafter. Using approximation theory, the inherent approximation error  $\epsilon_h$  can be reduced arbitrarily by increasing the number of  $h$  [19], then it is reasonable to assume  $\epsilon_h$  is bounded by a constant  $\epsilon_H$ , and

$$\epsilon_H = \sup_{t \in R^+} |\epsilon_h(t)| \quad (10)$$

If the variables ( $\mathbf{x}_d$  and  $\dot{\mathbf{x}}_d$ ) are in the compact sets,  $\epsilon_H$  is finite. If they are not in compact sets, a scaling mechanism can be utilized to convert them into a compact set [20].

With the RBFN controller, using Eq.8, the control input vector  $\mathbf{u}$  is,

$$\mathbf{u} = \sum_{k=1}^h \hat{\mathbf{w}}_k \exp\left(-\frac{1}{\hat{\sigma}_k^2} \|\zeta - \hat{\mu}_k\|^2\right) + K_p \mathbf{e} = \hat{W}^T \hat{\phi} + K_p \mathbf{e} \quad (11)$$

where  $\hat{\mathbf{w}}_k$  is the estimated weights,  $\hat{\sigma}_k$  and  $\hat{\mu}_k$  are the estimated center and width for the  $k$ th hidden neuron.  $\hat{W}^T$  and  $\hat{\phi}$  are the corresponding matrix expressions. Substituting Eq.9, Eq.11 into Eq.6, the error dynamics becomes,

$$\dot{\mathbf{e}} = (A(t) + B(t)K_p)\mathbf{e} + B(t)(\hat{W}^T \hat{\phi} - W^{*T} \phi^* - \epsilon_h) \quad (12)$$

Using  $J(t) = A(t) + B(t)K_p$ , defining  $\tilde{W}$  the difference between  $W^*$  and  $\hat{W}$ ,  $\tilde{\phi}$  the difference

between  $\phi^*$  and  $\hat{\phi}$ , neglecting higher order item  $\tilde{W}\tilde{\phi}$ , the error dynamics may be written as,

$$\dot{\mathbf{e}} \doteq J\mathbf{e} - B(t)(\hat{W}^T\tilde{\phi} + \tilde{W}^T\hat{\phi}) - B(t)\epsilon_h \quad (13)$$

where  $B(t)(\hat{W}^T\tilde{\phi} + \tilde{W}^T\hat{\phi})$  represents the learning error  $E_l$ .

### 3.3 Stable Adaptive Law for a Fully Tuned RBFN

To derive the stable tuning law, the following Lyapunov function is chosen,

$$V = \frac{1}{2}\mathbf{e}^T P\mathbf{e} + \frac{1}{2}tr(\tilde{W}^T\Theta\tilde{W}) + \frac{1}{2}\tilde{\phi}^T\Lambda\tilde{\phi} \quad (14)$$

where  $P$  is an  $n \times n$  symmetric positive definite matrix, and  $\Theta, \Lambda$  are  $h \times h$  non-negative definite matrices. The derivative of the Lyapunov function is given by,

$$\dot{V} = \frac{1}{2}[\dot{\mathbf{e}}^T P\mathbf{e} + \mathbf{e}^T P\dot{\mathbf{e}}] + tr(\tilde{W}^T\Theta\dot{\tilde{W}}) + \tilde{\phi}^T\Lambda\dot{\tilde{\phi}} \quad (15)$$

Substituting Eq.13 into Eq.15,

$$\dot{V} = -\mathbf{e}^T Q\mathbf{e} - \epsilon_h^T B(t)^T P\mathbf{e} - \tilde{\phi}^T \hat{W} B(t)^T P\mathbf{e} - \hat{\phi}^T \tilde{W} B(t)^T P\mathbf{e} + tr(\tilde{W}^T\Theta\dot{\tilde{W}}) + \tilde{\phi}^T\Lambda\dot{\tilde{\phi}} \quad (16)$$

where  $Q = -\frac{1}{2}(J^T P + PJ)$ . Since  $J$  is Hurwitz, the Lyapunov function can always be found and has a unique solution. Noting in Eq.16,

$$tr(\tilde{W}^T\Theta\dot{\tilde{W}}) = \sum_{i=1}^p \tilde{\mathbf{w}}_i^T \Theta \dot{\tilde{\mathbf{w}}}_i \quad (17)$$

$$\hat{\phi}^T \tilde{W} B(t)^T P\mathbf{e} = \sum_{i=1}^p \hat{\phi}^T \tilde{\mathbf{w}}_i \mathbf{B}(t)_i^T P\mathbf{e} \quad (18)$$

Eq.16 becomes,

$$\begin{aligned} \dot{V} = & -\mathbf{e}^T Q\mathbf{e} - \epsilon_h^T B(t)^T P\mathbf{e} + \tilde{\phi}^T (-\hat{W} B(t)^T P\mathbf{e} + \Lambda\dot{\tilde{\phi}}) + \sum_{i=1}^p (-\tilde{\mathbf{w}}_i^T \hat{\phi} B(t)_i^T P\mathbf{e} \\ & + \tilde{\mathbf{w}}_i^T \Theta \dot{\tilde{\mathbf{w}}}_i) \end{aligned} \quad (19)$$

where  $\dot{\tilde{\mathbf{w}}}_i$  is the  $i$ th column of matrix  $\dot{\tilde{W}}$ ,  $B(t)_i$  is the  $i$ th column of matrix  $B(t)$ .

If  $\dot{\tilde{\mathbf{w}}}_i$  and  $\dot{\tilde{\phi}}$  are selected as,

$$\dot{\tilde{\mathbf{w}}}_i = \Theta^{-1} \hat{\phi} B(t)_i^T P \mathbf{e}, \quad i = 1, \dots, p \quad (20)$$

$$\dot{\tilde{\phi}} = \Lambda^{-1} \hat{W} B(t)^T P \mathbf{e} \quad (21)$$

Then Eq.19 becomes,

$$\dot{V} = -\mathbf{e}^T Q \mathbf{e} - \epsilon_h^T B(t)^T P \mathbf{e} \quad (22)$$

$\dot{V}$  can be demonstrated negative according to Eq.10,

$$\dot{V} \leq -\|\mathbf{e}\| \lambda_{\min}(Q) \|\mathbf{e}\| + \|\mathbf{e}\| \|P\| \|B(t)\| \epsilon_H \quad (23)$$

Let  $\|B(t)\| \epsilon_H = \delta_{\epsilon_h}$ , it can be shown directly that  $\dot{V}$  is negative when

$$\|\mathbf{e}\| \geq \frac{\|P\|}{\lambda_{\min}(Q)} \delta_{\epsilon_h} = E_a \quad (24)$$

Using the universal approximation property [19], by increasing the number  $h$ ,  $\epsilon_H$  can be reduced arbitrarily small, which means that  $E_a$  tends to zero when  $h \rightarrow \infty$  and the negativeness of the Lyapunov function can be guaranteed, resulting in the overall system to be stable.

Since  $\dot{\tilde{\mathbf{w}}}_i = \dot{\mathbf{w}}_i^* - \dot{\tilde{\mathbf{w}}}_i$ ,  $\dot{\tilde{\phi}} = \dot{\phi}^* - \dot{\hat{\phi}}$  and  $\dot{\mathbf{w}}_i^* = \mathbf{0}$ ,  $\dot{\phi}^* = \mathbf{0}$ , the tuning rules are,

$$\dot{\tilde{\mathbf{w}}}_i = -\Theta^{-1} \hat{\phi} B(t)_i^T P \mathbf{e}, \quad i = 1, \dots, p \quad (25)$$

$$\dot{\tilde{\phi}} = -\Lambda^{-1} \hat{W} B(t)^T P \mathbf{e} \quad (26)$$

### 3.4 Robustness Analysis

From the above derivation it can be seen that using the proposed method, provided Assumption 2 stands, the matrices  $A(t)$  and  $B(t)$  need not be known exactly. However, since the tuning

rules derived involve the prior knowledge of matrix  $B(t)$ , the robustness of the tuning laws is analyzed in case  $B(t)$  is partially known or unknown (actuator failure or model error).

Case 1: Matrix  $B(t)$  is varying, but satisfies  $B(t) = k(t)B_0$ , where  $B_0$  is a known nominal value,  $k(t)$  is a positive scalar varying with time.

Provided the weight tuning rule and the tuning rule of  $\phi$  are given separately as,

$$\dot{\hat{\mathbf{w}}}_i = -k\Theta^{-1}\hat{\phi}B_{0i}^T P\mathbf{e}, \quad i = 1, \dots, p \quad (27)$$

$$\dot{\hat{\phi}} = -k\Lambda^{-1}\hat{W}B_0^T P\mathbf{e} \quad (28)$$

$\dot{V}$  can be demonstrated negative using the same condition as in Eq.24. In implementation,  $k$  can be replaced by a positive scalar  $\eta$ , because  $\eta$  and  $k$  has the same sign, the system's convergence characteristic will not change.

Case 2:  $B(t)$  is unknown and is represented by a nominal value plus its variation,  $B(t) = B_0(I + \Delta B(t))$ .

Assume  $\|\Delta B(t)\| \leq M$ ,  $M$  is the upper bound for the uncertainty. The derivative of Lyapunov function of Eq.15 is re-analyzed and is given by,

$$\begin{aligned} \dot{V} = & -\mathbf{e}^T Q\mathbf{e} - \mathbf{e}^T P(B_0 + B_0\Delta B(t))\epsilon - (\tilde{\phi}^T \hat{\mathbf{w}}B_0\Delta B(t)P\mathbf{e} + \hat{\phi}^T \tilde{W}B_0\Delta B(t)P\mathbf{e}) + \tilde{\phi}^T (-\hat{W} \\ & (B_0 + B_0\Delta B(t))^T P\mathbf{e} + \Lambda\dot{\tilde{\phi}}) + \sum_{i=1}^p (-\tilde{\mathbf{w}}_i^T \hat{\phi}(B_{0i} + B_{0i}\Delta B_i)^T P\mathbf{e} + \tilde{\mathbf{w}}_i^T \Theta\dot{\tilde{\mathbf{w}}}_i) \end{aligned} \quad (29)$$

Rewriting the weight tuning rule and the tuning rule of  $\phi$  using  $B_0$ ,

$$\dot{\hat{\mathbf{w}}}_i = -\Theta^{-1}\hat{\phi}B_{0i}^T P\mathbf{e}, \quad i = 1, \dots, p \quad (30)$$

$$\dot{\hat{\phi}} = -\Lambda^{-1}\hat{W}B_0^T P\mathbf{e} \quad (31)$$

then,

$$\dot{V} = -\mathbf{e}^T Q\mathbf{e} - \mathbf{e}^T P(B_0 + B_0\Delta B(t))\epsilon_h - (\tilde{\phi}^T \hat{\mathbf{w}} + \hat{\phi}^T \tilde{W})B_0\Delta B(t)P\mathbf{e} \quad (32)$$

Re-analyzing  $\dot{V}$ , we have,

$$\dot{V} \leq -\|\mathbf{e}\| \lambda_{\min}(Q) \|\mathbf{e}\| + \|\mathbf{e}\| \|P\| \|B_0\| \|I + \Delta B(t)\| \epsilon_H + \|\tilde{\Phi}^T \hat{W} + \hat{\Phi}^T \tilde{W}\| \|B_0\| \|\Delta B(t)\| \|P\| \|\mathbf{e}\| \quad (33)$$

Since  $\|I + \Delta B(t)\| \epsilon_H \leq \delta'_{\epsilon_h}$  and defining  $\|(\tilde{\Phi}^T \hat{W} + \hat{\Phi}^T \tilde{W})\| \|P\| = \zeta(\tilde{\phi}, \tilde{W})$  to be brief, it can be derived directly that  $\dot{V}$  is negative when

$$\|\mathbf{e}\| \geq \frac{\|P\| \|B_0\|}{\lambda_{\min}(Q)} \delta'_{\epsilon_h} + \frac{\|P\| \|B_0\|}{\lambda_{\min}(Q)} \zeta(\tilde{\phi}, \tilde{W}) \|\Delta B(t)\| = E_a + E_{lm} \quad (34)$$

In the above equation,  $E_a$  is caused by the approximation inaccuracy  $\epsilon_h$ ,  $E_{lm}$  is caused by the product of the learning error  $E_l(\|(\tilde{\Phi}^T \hat{W} + \hat{\Phi}^T \tilde{W})\|)$  and modeling error  $E_m(\|\Delta B(t)\|)$ .

Remarks:

- (1) If there is neither approximation error (i.e.  $\epsilon_h = 0$ ) nor model error (i.e.  $\Delta B(t) = 0$ ), from Eq.33,  $\dot{V}$  is negative semidefinite and hence the stability of the overall scheme is guaranteed.
- (2) The approximation error  $E_a$  is related to the number of hidden neurons used in the RBFN. Given enough hidden units, this number converges to a very small number.
- (3) The second error item  $E_{lm}$  is a combination of the learning error and modeling error. The product of  $\zeta$  and  $\Delta B(t)$  indicates that even under large model error  $\Delta B(t)$ , if the RBFN learns the desired value  $W^*$  and  $\phi^*$  correctly,  $E_{lm}$  is still very small.
- (4) Matrix  $B(t)$  in fact plays a very important role in the strategy. It is better to use some strategies to identify this matrix or incorporate a robust control strategy to obtain good performance.

### 3.5 Implementation of the Tuning Rule

In Eq.26, the Gaussian function  $\hat{\phi}$  is embedded with the centers' locations  $\hat{\mu}$  and widths  $\hat{\sigma}$ , i.e.  $\hat{\phi} = \phi(\hat{\mu}, \hat{\sigma}, \zeta)$ . Combining all the adaptable parameters into a big vector,  $\chi =$

$[\hat{\mathbf{w}}_1^T, \hat{\mu}_1^T, \hat{\sigma}_1, \dots, \hat{\mathbf{w}}_h^T, \hat{\mu}_h^T, \hat{\sigma}_h]^T$ , a simple updating rule for  $\chi$  can be derived. Because the real implementation is carried out in a discrete-time framework, the updating laws is also derived under discrete form.

First, Eq.25 can be converted into,

$$\begin{aligned} \dot{\hat{\mathbf{w}}}_i &= -\Theta^{-1} \hat{\phi} B_i^T P \mathbf{e}, \quad i = 1, \dots, p \\ \Rightarrow \dot{\hat{W}}^T &= -B(t)^T P \mathbf{e} \hat{\phi}^T \Rightarrow \dot{\hat{\mathbf{w}}}_j = -B(t)^T P \mathbf{e} \hat{\phi}_j, \quad j = 1, \dots, h \quad \Theta = I \end{aligned} \quad (35)$$

Where  $\hat{\mathbf{w}}_j$ , a column vector, is the  $j$ th row of matrix  $\hat{W}$ . Define  $\hat{g} = \hat{W}^T \hat{\phi}$  the output of the RBFN,  $\hat{\phi}$  is the derivative of  $\hat{\mathbf{g}}(\cdot)$  to the weights  $\hat{\mathbf{w}}$ , this equation can be converted into a discrete form,

$$\hat{\mathbf{w}}_i(n+1) = \hat{\mathbf{w}}_i(n) - \eta_1 \frac{\partial \hat{\mathbf{g}}}{\partial \hat{\mathbf{w}}_i^T} B(t)^T P \mathbf{e}(n) \quad i = 1, \dots, h \quad (36)$$

Next, from Eq.26,

$$\Delta \hat{\phi}(n) = \hat{\phi}(n+1) - \hat{\phi}(n) = -\eta_2 \Lambda^{-1} \hat{W} B(t)^T P \mathbf{e}(n) \quad (37)$$

As mentioned above, Gaussian function  $\hat{\phi} = \phi(\hat{\mu}, \hat{\theta}, \zeta)$  and the RBFN's output  $\hat{\mathbf{g}} = \hat{W}^T \hat{\phi}$ , therefore the updating laws for centers and widths of the basis functions become,

$$\begin{aligned} \hat{\mu}_i(n+1) &= \hat{\mu}_i(n) + \eta_3 \frac{\partial \hat{\phi}(n)}{\partial \hat{\mu}_i^T} \Delta \hat{\phi}(n) = \hat{\mu}_i(n) - \eta_2 \eta_3 \frac{\partial \hat{\phi}(n)}{\partial \hat{\mu}_i^T} \hat{W} B(t)^T P \mathbf{e}(n) \\ &= \hat{\mu}_i(n) - \eta_2 \eta_3 \frac{\partial \hat{\mathbf{g}}}{\partial \hat{\mu}_i^T} B(t)^T P \mathbf{e}(n) \quad i = 1, \dots, h \end{aligned} \quad (38)$$

$$\begin{aligned} \hat{\sigma}_i(n+1) &= \hat{\sigma}_i(n) + \eta_4 \frac{\partial \hat{\phi}(n)}{\partial \hat{\sigma}_i^T} \Delta \hat{\phi}(n) = \hat{\sigma}_i(n) - \eta_2 \eta_4 \frac{\partial \hat{\phi}(n)}{\partial \hat{\sigma}_i^T} \hat{W} B(t)^T P \mathbf{e}(n) \\ &= \hat{\sigma}_i(n) - \eta_2 \eta_4 \frac{\partial \hat{\mathbf{g}}}{\partial \hat{\sigma}_i^T} B(t)^T P \mathbf{e}(n) \quad i = 1, \dots, h \end{aligned} \quad (39)$$

In the above equations,  $\eta_1, \eta_2, \eta_3$  and  $\eta_4$  are positive scalars to be selected properly. Integrating Eq.36 , Eq.38 and Eq.39 by using the vector  $\chi$ , the updating rule is,

$$\chi(n+1) = \chi(n) - \eta \alpha(n) B(t)^T P \mathbf{e}(n) \quad (40)$$

Where  $\eta$  is learning rate.  $\alpha(n) = \nabla_{\chi} \hat{\mathbf{g}}(\zeta_n)$  is the gradient of the function  $\hat{\mathbf{g}}()$  with respect to the parameter vector  $\chi$  evaluated at  $\chi(n-1)$ , and

$$\alpha(n) = \left[ \hat{\phi}_1, \hat{\phi}_1 \frac{2\hat{\mathbf{w}}_1}{\hat{\sigma}_1^2} (\zeta_n - \hat{\mu}_1)^T, \hat{\phi}_1 \frac{2\hat{\mathbf{w}}_1}{\hat{\sigma}_1^3} \|\zeta_n - \hat{\mu}_1\|^2, \dots, \right. \\ \left. \hat{\phi}_h, \hat{\phi}_h \frac{2\hat{\mathbf{w}}_h}{\hat{\sigma}_h^2} (\zeta_n - \hat{\mu}_h)^T, \hat{\phi}_h \frac{2\hat{\mathbf{w}}_h}{\hat{\sigma}_h^3} \|\zeta_n - \hat{\mu}_h\|^2 \right]^T \quad (41)$$

Since the number of hidden neurons of the RBFN is fixed in this scheme, the approximation error  $E_a$  is quite large at the initial learning period. If  $\|\mathbf{e}\| < E_a$ , then it is possible that  $\dot{V} > 0$ , which implies that the parameters of the network may drift to infinity. A common way to avoid this problem and ensure the convergence of approximation error is to incorporate a dead zone in the tuning rules[7]. The tuning rule is given now as,

$$\chi(n+1) = \begin{cases} \chi(n) - \eta \alpha(n) \mathbf{e}(n) & \text{if } \|\mathbf{e}\| \geq e_0 \text{ and } \|\mathbf{w}\| \leq h^{\frac{1}{2}} M \\ 0 & \text{otherwise} \end{cases} \quad (42)$$

In Eq.42,  $M$  is a positive scalar and  $h^{\frac{1}{2}} M$  is an upper bound on  $\|\mathbf{W}\|$  (Euclidean norm of weight vector),  $e_0$  is selected as the required accuracy on the state error  $\mathbf{e}$ . According to Eq.24, the size of the dead zone should be set to  $E_a$ , so that if the error  $\|\mathbf{e}\| < E_a$ , the tuning is stopped and the parameters will not drift away. Unfortunately, since  $\epsilon_H$  is time-varying and unknown, the exact value for  $E_a$  can not be estimated, therefore  $e_0$  is used in the place of  $E_a(t)$ . If  $e_0 > E_a$ , then  $\dot{V}$  is always negative. If  $e_0 < E_a$ , when the error  $\|\mathbf{e}\|$  converges to  $e_0 < \|\mathbf{e}\| < E_a$ ,  $\dot{V}$  may be positive and in this case the upper bound  $h^{\frac{1}{2}} M$  is used to prevent the parameter from drifting away.

## 4 Simulation Results

In [2], Sadhukhan and Fetieh presented an exact inverse neuro-controller for the linearized longitudinal dynamics of an F8 aircraft model  $\dot{x} = Ax + Bu$ . The linearized longitudinal model of F8 aircraft was obtained under the following trim condition:  $v(0) = 650 ft/s$  (Mach number=0.6),  $\alpha(0) = 0.078 rad$ , and altitude  $20,000 ft$ . The state vector  $x$  comprises of the velocity ( $v$ ), angle of attack ( $\alpha$ ), pitch rate ( $q$ ) and pitch angle ( $\theta$ ). The control inputs  $u$  include the elevator's output  $\delta_{ea}$  and the throttle's output  $\delta_{ta}$ . Physical constraints on the elevator are defined as  $-26.5^\circ \leq \delta_{ea} \leq 6.75^\circ$ . Simulation studies showed that the controller provided stable and near desired response in the presence of modeling uncertainty up to 30% (all the elements of matrix A and B are changed by 30%). However, under 40% model error, the velocity oscillates in the first few seconds though eventually settles down.

In this study, to assess the performance of the neuro-flight-controller, the model errors are simulated by changing all the elements of the state matrix  $A$  and control matrix  $B$  by 70%. Although simulation studies showed that the neuro controller can also be applied to control the system even when model error is greater than 70%, the results indicate that elevator actuator saturation will be the limiting factor resulting in deterioration of the tracking performance. The proportional controller gain was selected as  $K_p = \begin{bmatrix} -0.01 & 0.002 & 2.92 & 0.02 \\ -2.02 & 0 & -0.12 & -0.30 \end{bmatrix}$  to satisfy the Assumption 2, and good performance is observed in nominal condition from Fig.2. However, from Fig.2, it can be seen that by using the traditional controller alone, for the case of the 70% model error, the tracking results deteriorate greatly. Hence the RBFN controller is added as in Fig. 1 to improve the tracking performance by learning the inverse dynamics of the system. The aim of the study is to compare the tracking accuracy of velocity and pitch rate using a fully

tuned RBFN and a traditional RBFN with fixed centers and widths.

At the start of the on-line learning, based on the *a priori* knowledge, the number of hidden units were chosen as 6 and 15, the centers of the hidden units were fixed with a grid method described in [20] and their widths were set to 0.5. Fig.3 shows the tracking performance under nominal case, while Fig.4 shows the performance with 70% model error. Fig.3 indicates that under the nominal condition, all the strategies can achieve good performance because of the existence of the pre-designed feedback controller. However, when there is 70% model error, better tracking results are observed using fully tuned RBFN. From Fig.4 it can be easily seen that using a fully tuned RBFN, the tracking performance is quite good with only 6 hidden units, while if only the weights of the RBFN are updated, it is obvious that the controller can not track the desired commands correctly. Although increasing the number of hidden units to 15 improves the performance, it is still not as good as the RBFN with tuning all the parameters. Fig.5 depicts the evolution of tracking errors of both velocity and pitch rate. A detailed quantitative comparison in terms of the average  $E(ave)$  and peak tracking errors  $E(max)$  for different size of the RBFNs are given in Table.1 (All the results are compared under 70% model error). This phenomena, in fact indicate that in using a traditional RBFN, *a priori* knowledge should be embedded into the centers and widths of the hidden units accurately. On the other hand, with a fully tuned RBFN, since its centers, widths can be modified during the on-line learning, this requirement is not so stringent.

Fig.6 shows the control signals and indicates how the learning is carried out. The continuous line in the figure represents the outputs of the RBFN controller, while the dotted line represents the conventional controller's outputs. Using a RBFN controller with only tuning the weights, the outputs of the two controllers under nominal condition and model error are given in (Fig.6(a)) and

(Fig.6(b)) respectively. It can be seen from the figures that in the initial period, the conventional controller contributes a lot to maintain the performance, while the RBFN's outputs are varying, which indicates a process of learning. In nominal condition, after 10 seconds, the steady outputs are all due to the RBFN controller, and the outputs of the feedback controller shrinks to zero, which means RBFN has learned the system inverse. However, under model error, the traditional RBFN fails to catch up with the inverse dynamics of the system. This is actually caused by the incorrect centers and widths used. Compared to Fig.6(b), (Fig.6(c)) shows that with a fully tuned RBFN, the changes in the model dynamics can be grasped quickly. As a result, the control inputs generated are mainly formed by the RBFN controller. It may be noted that they are well within the saturation limits.

## 5 Dynamic Structured RBFN and Comparison Results

To implement a fully tuned RBFN as discussed in the previous sections, the number of the hidden neurons should be selected *a priori* by a *ad hoc* or through a trial-and-error procedure. However, in case of approximating the inverse dynamics of a very complicated system, this procedure will waste a lot of time, and superfluous hidden neurons will be added to the network. To avoid this problem, the performance of the dynamic structured RBFNs are investigated next. The RBFN with only a growing strategy is called "GRBFN" and a RBFN incorporating both Growing And Pruning strategies is referred to as a "GAP RBFN" in this paper.

For sequential learning of RBF networks, Platt [21] had proposed a Resource Allocating Network (RAN), where hidden neurons were added based on the novelty of the new data. Lu Yingwei *et al.* [22] extended Platt's algorithm by incorporating a pruning strategy so that those neurons

that consistently made little contribution to the network output were removed. The resulting network, known as Minimal RAN (MRAN), was shown to be even more parsimonious for several applications in the areas of function approximation and nonlinear system identification than RAN with better accuracy. The same growing only and growing and pruning strategies as described in [21][22] respectively are utilized in this section to design the GRBFN controller and GAP RBFN controller. The simulations compare the tracking accuracy as well as the resulting network complexity using fixed size RBFN and the dynamically structured RBFN (including GRBFN and GAP RBFN).

Fig.7(a) shows the tracking result under the 70% model error case using GRBFN and GAP RBFN separately, while Fig.7(b) describes the evolution history of the hidden neurons. From the figures it can be seen the network gradually learns the system inverse by recruiting new hidden neurons according to the novelty of the input data. The results also illustrates two facts, namely that using a fully tuned strategy, the output error is much smaller than the RBFN with only tuning the weights, and the second being that the network is more compact (less hidden neurons). The other phenomenon that can be observed in the figures is that with a GAP RBFN, a more compact network structure can be implemented even compared to the fully tuned GRBFN and tracking error using the two dynamic structured RBFNs is very close.

A quantitative comparison between the above two approaches is given in Table 2. It can be seen from the table that using GRBFN, with only tuning the weights, 24 hidden neurons are used, and the average tracking error for velocity and pitch rate are 0.79,0.21 respectively, while with a fully tuned GRBFN, these numbers decrease to 0.06 and 0.074, and the number of hidden units used is only 10. Compared with a fully tuned GRBFN, the GAP RBFN with a pruning strategy can even achieve better performance; i.e, less hidden neurons (only 4 units are needed)

with a smaller tracking error ( $E_v(ave) = 0.026, E_q(ave) = 0.037$ ).

## 6 Conclusions

In this paper, a stable on-line learning control strategy for a fully tuned RBFN is presented based on the feedback-error-learning scheme, with the aim of improving the tracking accuracy of the control system. The Lyapunov stability theory is utilized to derive a stable tuning rule for updating all the parameters of the RBFN, guaranteeing the local stability of the overall system. Simulation studies based on a F8 aircraft system demonstrate the benefits of using the “fully tuned” strategy and also confirms that the proposed scheme can tolerate more errors ( 70 %) compared to the earlier methods. Further studies on the network implementation reveal that dynamic structured RBFN can generate a more compact network structure with smaller tracking error, which is especially useful in a practical realization.

## References

- [1] A.J. Calise and R.T. Rysdyk, “Nonlinear adaptive flight control using neural networks”, IEEE Control Systems Magazine, Vol.18, No.6, Dec.1998, pp. 14-25.
- [2] D. Sadhukhan and S. Feteih, “F8 neuro controller based on dynamic inversion”, Journal of Guidance, Control and Dynamics, Vol.19, No.1, 1996, pp. 150-156.
- [3] M.R. Napolitano, S. Naylor, C. Neppach and V. Casdorff, “On-Line learning non-linear direct neuro-controllers for restructurable control systems”, AIAA Journal of Guidance, Control and Dynamics, Vol.18, No.1, 1995, pp. 170-176.

- [4] M.B. McFarland, R.T. Rysdyk and A.J. Calise, "Robust adaptive control using single-hidden-layer feedforward neural networks", Proceedings of the American Control Conference, pp. 4178-4182, June.1999, California, U.S.A.
- [5] R. Rysdyk, F. Nardi and A.J. Calise, "Robust adaptive nonlinear flight control applications using neural networks", Proceedings of the American Control Conference, pp. 2595-2599, June.1999, California, U.S.A.
- [6] T.H. Kerr, "Critique of some neural network architectures and claims for control and estimation", IEEE Trans. on Aerospace and Electronic Systems, Vol.34, No.2, 1998, pp. 406-419.
- [7] H. Gomi and M. Kawato, "Neural network control for a closed-loop system using feedback-error-learning", Neural Networks, Vol.6, No.7, 1993, pp. 933-946.
- [8] D. Gorinevsky, A. Kapitanovsky and A. Goldenberg, "Radial basis function network architecture for nonholonomic motion planning and control of free-flying manipulators", IEEE Tran. on Robotics and Automation, Vol.12, No.3, June 1996, pp. 491-496.
- [9] V. Kadiramanathan and M. Niranjan, "A function estimation approach to sequential learning with neural networks", Neural Computation, Vol.5, 1993, pp. 954-975.
- [10] R.M. Sanner and J.J. Slotine, "Gaussian networks for direct adaptive control", IEEE Tran. on Neural Networks, Vol.3, No.6, 1992, pp. 837-863.
- [11] M.M. Polycarpou, "Stable adaptive neural control scheme for nonlinear systems", IEEE Trans. on Automatic Control, Vol.41, No.3, 1996, pp. 447-451.
- [12] M.M. Polycarpou and M.J. Mears, "Stable adaptive tracking of uncertain systems using nonlinearly parameterized on-line approximators, I.J.C, Vol.70, NO.3, 1998, pp. 363-384.
- [13] S.N. Singh and W.R. Wells, "Direct Adaptive and Neural Control of Wing-Rock Motion of Slender Delta Wings", Journal of Guidance, Control, and Dynamics, Vol.18, No.1, 1995, pp. 25-30.

- [14] B.S. Kim and A.J. Calise, "Nonlinear flight control using neural networks", *Journal of Guidance, Control, and Dynamics*, Vol.20, No.1, 1997, pp. 26-33.
- [15] Birgmeier, M., "A fully Kalman-trained radial basis function network for nonlinear speech modeling", *Proceedings of IEEE International Conference on Neural Networks*, Vol.1, 1995, pp.259-264.
- [16] Kevin Warwick, "The control of dynamical systems by neural networks", *IEEE/IAS International Conference on Industrial Automation and Control*, 1995, pp.341-346.
- [17] Panchapakesan, C., Ralph, D. and Palaniswami, M., "Effects of moving the centers in an RBF network", *Proceedings of the 1998 IEEE International Joint Conference on Neural Networks*, Vol.2, 1998, pp.1256-1260.
- [18] K.S.Narendra, Snehasis Mukhopadhyay, "Adaptive control using neural networks and approximate models", *IEEE Tran. on Neural Networks*, Vol.8, No.3, 1997, pp.475-485.
- [19] J. Park and I.W. Sandberg, "Universal approximation using radial basis function networks", *Neural Computation*, Vol.3, 1991, pp. 246-257.
- [20] Simon Fabri, Visakan Kadiramanathan, "Dynamic structure neural networks for stable adaptive control of nonlinear systems", *IEEE Tran. on Neural Networks*, Vol.7, No.5, Sep,1996, pp.1151-1167.
- [21] J.C. Platt, "A resource allocating network for function interpolation", *Neural Computation*, Vol.3, 1991, pp. 213-225.
- [22] Y.W. Lu, N. Sundararajan and P. Saratchandran, "Performance evolution of a sequential minimal RBF neural network learning algorithm", *IEEE Trans. on Neural Networks*, Vol.9, No.2, 1998, pp. 308-318.

Table 1: Tracking Results Comparison Using RBFN

Network	Adapted	Hidden Units	Tracking Error			
Structure	Parameters		$E_v(max)$	$E_v(ave)$	$E_q(max)$	$E_q(ave)$
RBFN	$\mathbf{w}_i$	6	6.78	3.37	2.79	0.68
RBFN	$\mathbf{w}_i, \mu_i, \theta_i$	6	1.02	0.11	2.78	0.09
RBFN	$\mathbf{w}_i$	15	4.96	0.83	2.67	0.22
RBFN	$\mathbf{w}_i, \mu_i, \theta_i$	15	0.92	0.042	2.63	0.062

Table 2: Tracking Results Comparison Using Different RBFN Structures

Network Structure	Initiali- zation	Adapted Parameters	Hidden Units	Tracking Error			
				$E_v(max)$	$E_v(ave)$	$E_q(max)$	$E_q(ave)$
RBFN	Grid	$\mathbf{w}_i$	15	4.96	0.83	2.67	0.22
RBFN	Grid	$\mathbf{w}_i, \mu_i, \theta_i$	15	0.92	0.042	2.63	0.062
GRBFN	no need	$\mathbf{w}_i$	24	3.67	0.79	2.75	0.21
GRBFN	no need	$\mathbf{w}_i, \mu_i, \theta_i$	10	1.69	0.06	2.55	0.074
GAP RBFN	no need	$\mathbf{w}_i, \mu_i, \theta_i$	4	0.69	0.026	1.75	0.037

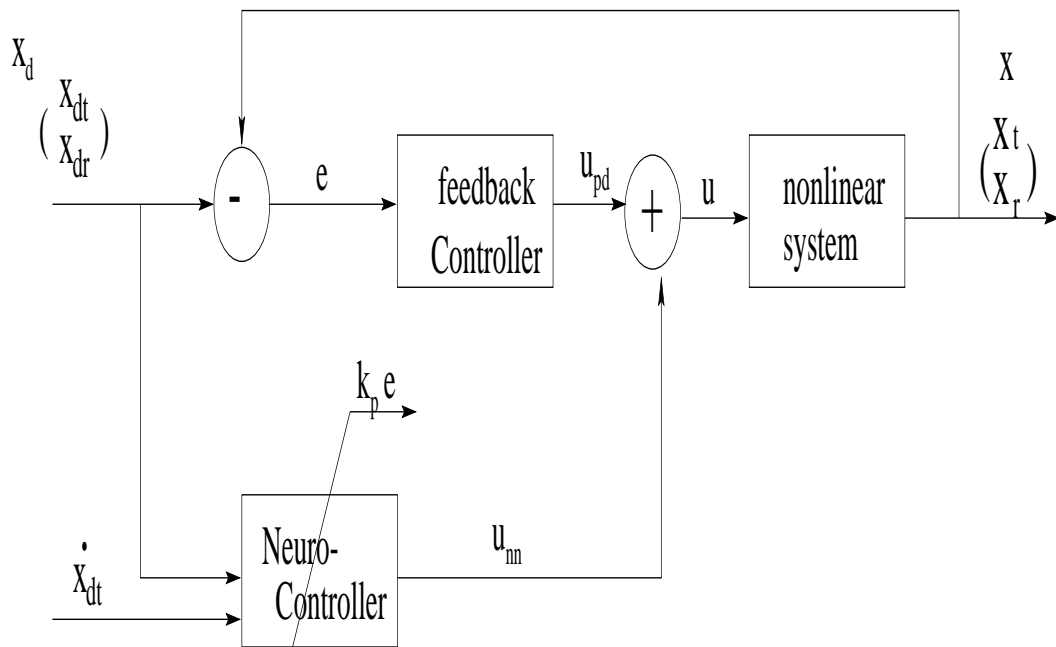


Figure 1: Control Structure of NN-PD Controller

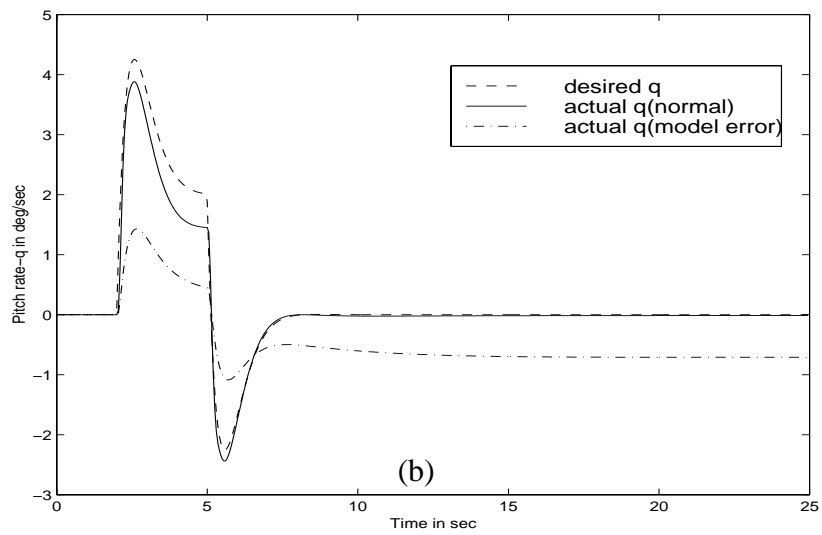
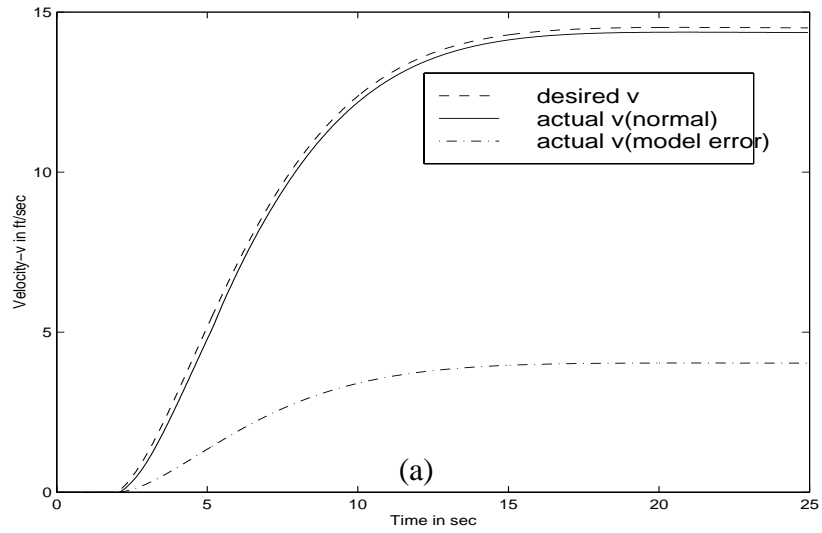


Figure 2: Desired trajectory and traditional controller response.

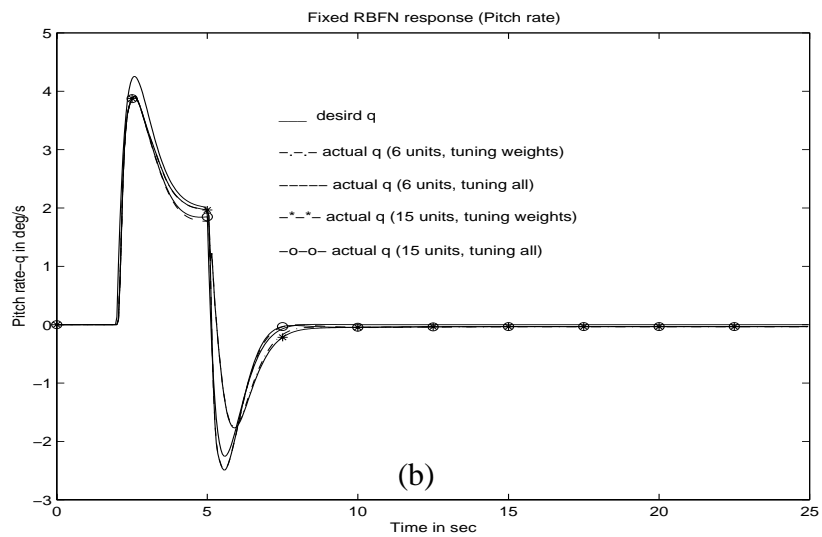
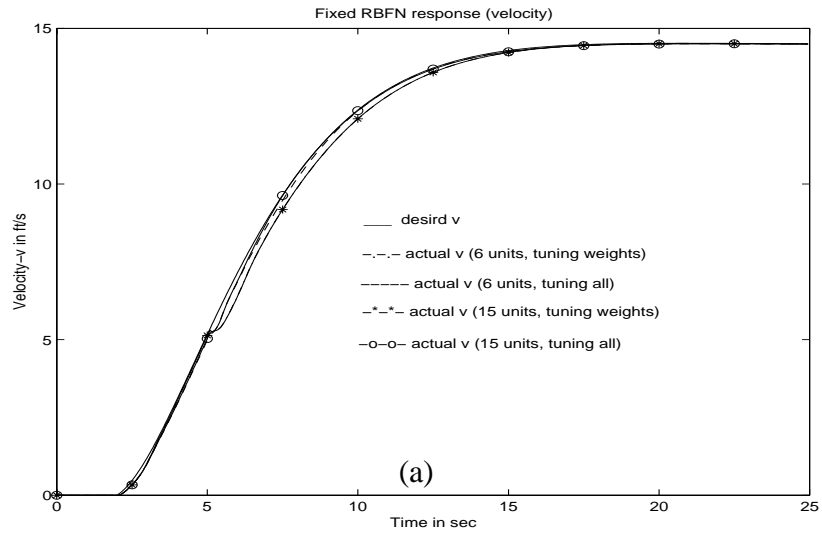


Figure 3: RBFN response under normal condition.

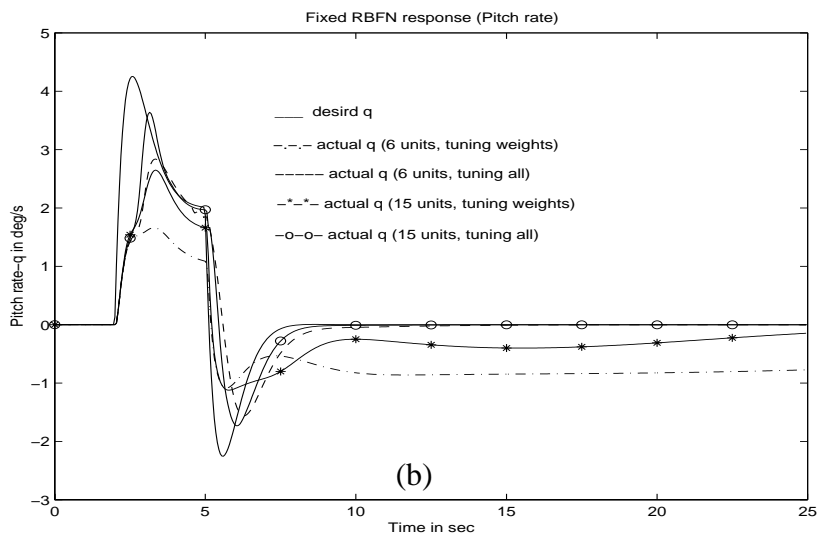
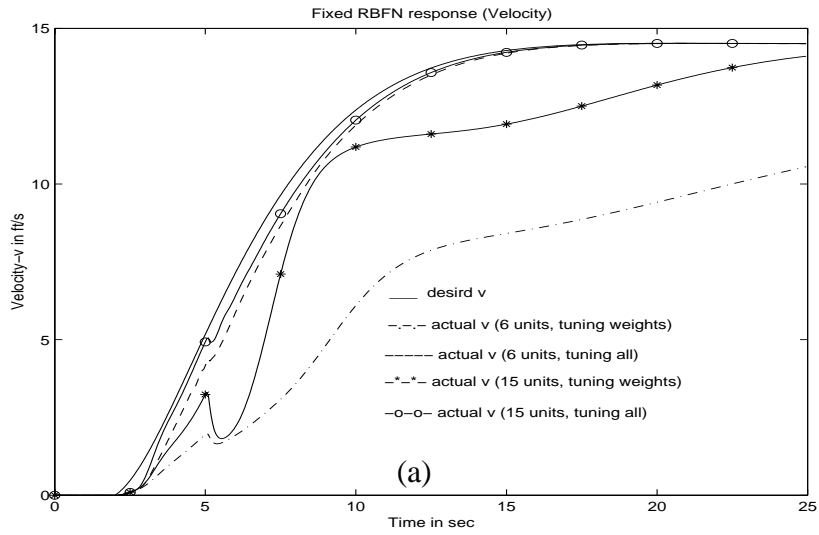


Figure 4: RBFN response under 70% model error.

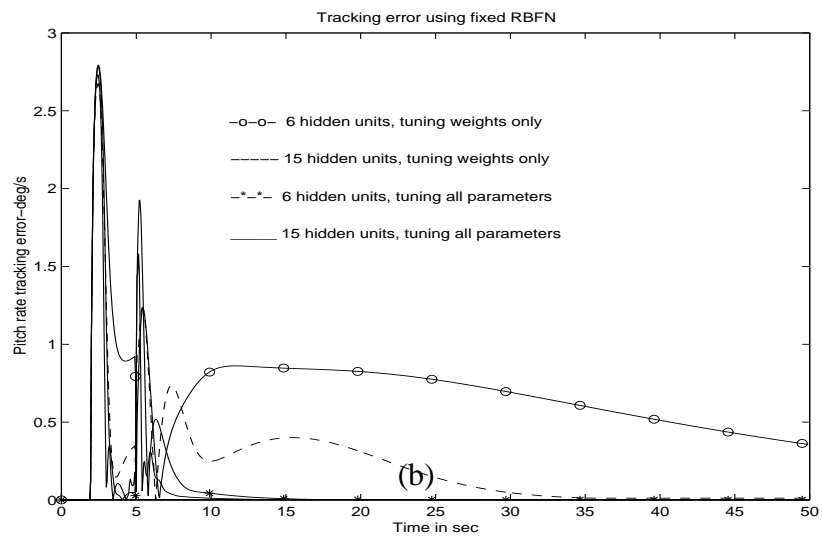
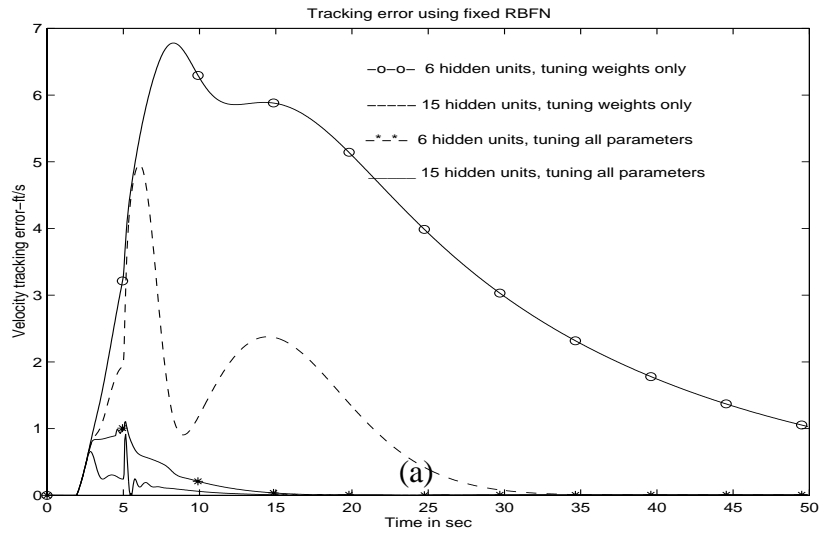


Figure 5: Evolution of tracking errors under 70% model error.

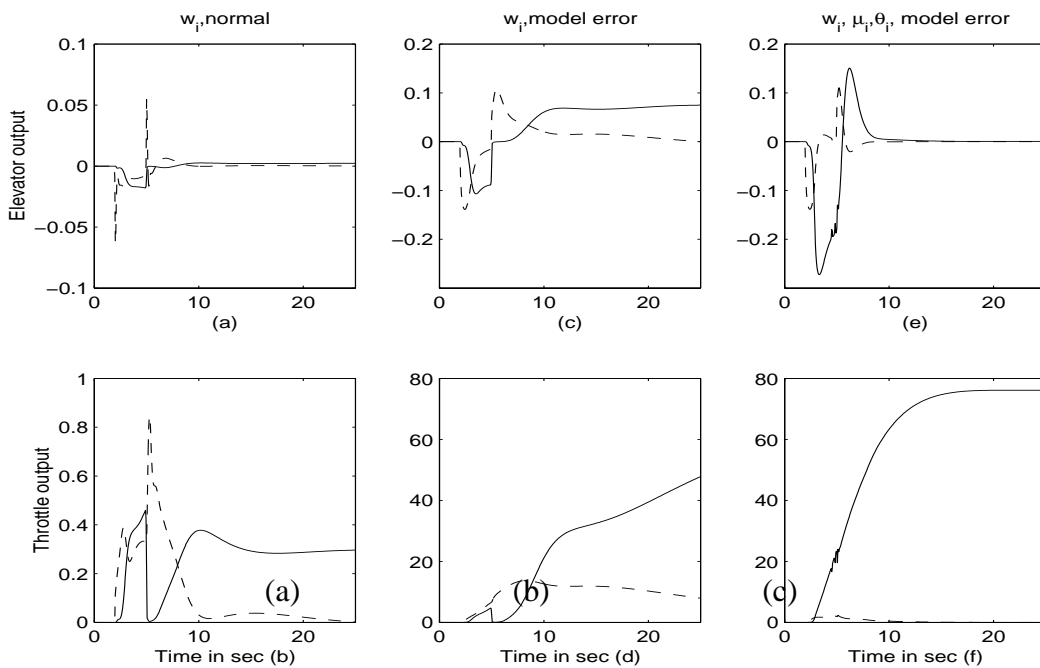


Figure 6: Control inputs (RBFN with 6 hidden units).

(a) Normal; (b) Model error (tuning weights); (c) Model error (fully tuned). The dot line is the output of the conventional controller while line refers to the output of the RBFN controller.

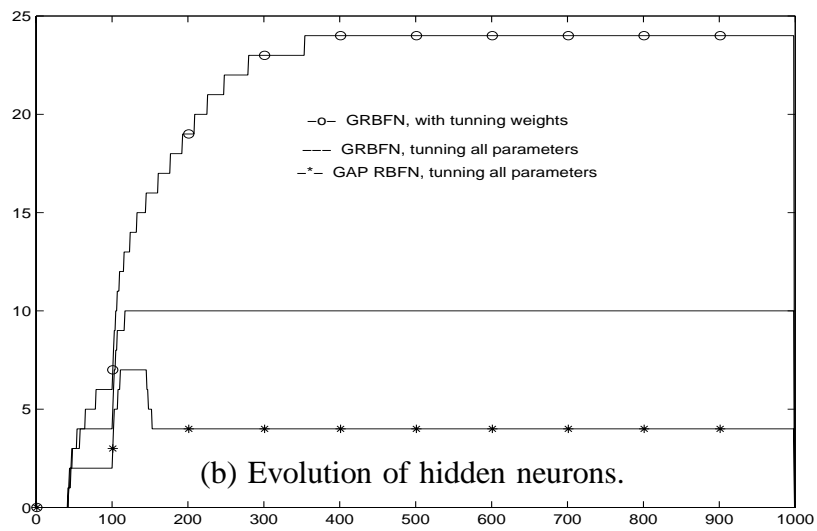
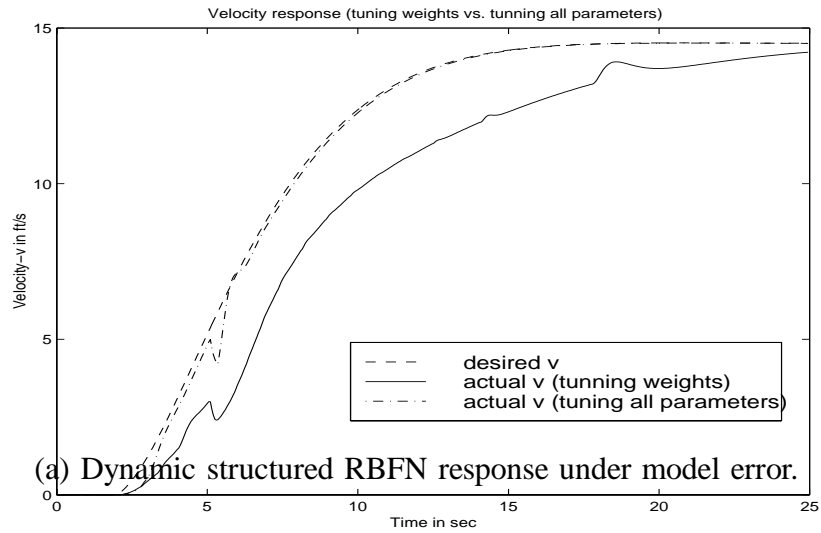


Figure 7: System performance using different network structures.