Adaptive Backstepping Design for a Longitudinal UAV Model Utilizing a Fully Tuned Growing Radial Basis Function Network

Yi Luo
*Carnegie Mellon University, Department of Mechanical Engineering, Pittsburgh, PA.

Yoo Hsiu Yeh
*Carnegie Mellon University, NASA Ames Research Center, Moffett Field, CA.

Abraham K. Ishihara
*Carnegie Mellon University, NASA Ames Research Center, Moffett Field, CA

Classical Radial Basis Function (RBF) neural network controller designs typically fix the number of basis functions and tune only the weights. In this paper we present a backstepping neural network controller algorithm in which all RBF parameters, including centers, variances and weight matrices are tuned online. By using a Lyapunov approach, tuning rules for updating the RBF parameters are derived and a stability and robustness analysis is presented. Additionally, we incorporate the ability to append RBF neurons such that both tracking performance and computational cost can be optimized. The condition for adding a neuron is based on a sliding RMS error window. In addition to the theoretical results, we present the controller implemented in several simulations of an auto landing sequence for a unmanned aerial vehicle (UAV) model.

Key words: Fully tuned Growing Radial Basis Function, Neural Network Control, Backstepping

I. Introduction

In recent years, neural networks have been investigated for control of nonlinear systems. The primary advantage of using neural networks lies in the universal approximation property, in which sufficiently smooth nonlinear functions can be approximated to arbitrary accuracy. Among various neural networks activation functions, Radial Basis Functions (RBF) have significant advantage because of the simple structure and approximation property holding over a known compact set. In this paper, we use fully tuned RBF neural networks to approximate uncertainties in nonlinear adaptive backstepping controller design. We then apply the backstepping neural network controller to a nonlinear longitudinal model of a UAV.

Neural network controllers have been widely used in control of robot manipulators and other dynamic system which contain highly coupled and nonlinear terms. In the 1990s, Lewis and his colleagues published a series of paper together with a book in which they presented novel nonlinear controllers for robot arm manipulators using neural networks. In, they proposed a nonlinear neural network controller using basis functions with guaranteed tracking performance by appropriate adjustment of the feedback gain. In, they proposed a new nonlinear neural network controller using multi-layer neural networks with the sigmoid functions. Compared with basis function type neural network controllers, the multi-layer sigmoid type can achieve better performance, however, suffer from a higher computational cost. In both papers only the weight matrices were tuned online. Other parameters were not optimized.

In Growing Radial Basis Function Neural Networks (GRBFNN), the number of nodes is optimized as well. Mario and his colleagues published a paper that compared and summarized different growing radial basis function (GRBF) algorithms for control system applications. In the textbook, they present a systematic approach for GRBF neural networks control of nonlinear systems utilizing an extended Kalman Filter.

Neural networks have also been implemented in conjunction with adaptive control designs. In particular, this paper will focus on using neural networks to simplify a backstepping controller design. Chiman proposed a robust backstepping control method for nonlinear systems using neural networks and validated it on several nonlinear systems.
Yahui and his colleagues also proposed a robust and adaptive backstepping controller for nonlinear systems that incorporated a robustification term.

Flight path tracking control for unmanned aerial vehicles (UAVs) poses several difficulties: (1) the flight dynamic system is a highly coupled nonlinear system with 6 degree of freedom (DOF), and (2) the system parameters related to many aerodynamic coefficients cannot be analytically solved. Instead, these parameters are obtained from experimental system identification which inevitably leads to modeling errors. This limits traditional linear control techniques (PID, LQR, etc.) especially during an abrupt failure.

Various types of nonlinear adaptive control techniques have been developed for flight path tracking control. Calise proposed a nonlinear adaptive controller for flight control using neural networks. Johnson proposed a neural network adaptive controller with application to helicopter tracking control. Yi et al., presented a fully tuned growing RBF neural network controller for a highly unstable fighter aircraft. In Yi’s publications, they proposed a fully tuned GRBF controller both in continuous and discrete time in which the centers and standard variations of the RBF was tuned automatically. Lee and Sharma presented neural-adaptive backstepping controllers with application to fixed wing aircraft. More recently, Ju proposed a longitudinal auto-landing adaptive backstepping controller. Instead of using neural networks, he presented an adaptive tuning algorithm to estimate the model uncertainty. Furthermore, he proposed an auto-landing trajectory planning method and demonstrated tracking performance in the presence of wind disturbances.

In this paper we present a fully tuned GRBF neural network control scheme in which all RBF parameters, including centers, variances and weights matrices are tuned online. By using a Lyapunov approach, tuning rules for updating the RBF parameters are derived and stability analysis is presented. The discrete time equations are derived via the gradient descent method. In addition to the fully tuned RBF neural network controller, we incorporate the ability to grow RBF neurons such that both tracking performance and computational cost can be optimized. The condition for adding one neuron is based on a sliding RMS error window. We show the performance of the proposed scheme by a simulating a backstepping-controlled auto-landing sequence of a UAV. The remainder of this paper is organized as follows: Section II discusses the relevant aircraft dynamics. The detailed problem formulation is also presented. Section III describes the damaged UAV model as additive parameter uncertainties. The non-adaptive control scheme and stability proof are provided. Section IV covers the derivation of fully tuned GRBF neural network with proof of stability. Section V presents several simulations of proposed controller performing flight path angle tracking control. We also present the auto-landing flight path command generation which was taken from Ju’s paper. We conclude in Section VI.

## II. Longitudinal Aircraft Dynamics

### II.A. General Aircraft Dynamic

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>θ</td>
<td>Pitch angle</td>
<td>C_D</td>
<td>Drag coefficient</td>
</tr>
<tr>
<td>φ</td>
<td>Roll(bank) angle</td>
<td>C_Y</td>
<td>Side force coefficient</td>
</tr>
<tr>
<td>ψ</td>
<td>Yaw(heading) angle</td>
<td>C_n</td>
<td>Roll moment coefficient</td>
</tr>
<tr>
<td>α</td>
<td>Angle of Attack</td>
<td>C_l</td>
<td>Pitch moment coefficient</td>
</tr>
<tr>
<td>β</td>
<td>Side slip</td>
<td>V</td>
<td>Aircraft velocity</td>
</tr>
<tr>
<td>p</td>
<td>Roll rate</td>
<td>S</td>
<td>Wing area</td>
</tr>
<tr>
<td>q</td>
<td>Pitch rate</td>
<td>T</td>
<td>Thrust force</td>
</tr>
<tr>
<td>r</td>
<td>Yaw rate</td>
<td>m</td>
<td>Mass of aircraft</td>
</tr>
<tr>
<td>C_L</td>
<td>Lift coefficient</td>
<td>g</td>
<td>Gravitational constant</td>
</tr>
<tr>
<td>x_E, y_E, h</td>
<td>Earth coordination values</td>
<td>I_x, I_y, I_z, I_xz</td>
<td>Moment of inertia w.r.t each axis</td>
</tr>
</tbody>
</table>

In this section, twelve highly coupled nonlinear ordinary differential equations which describe flight dynamics are presented, with notations explanation in Table 1. They are:
(1) Force Equations (Wind-axes):

\[
\begin{align*}
\mathbf{V} &= -\frac{\partial \mathbf{S}}{\partial m} C_{D_\alpha} + \frac{T}{m} \cos \alpha \cos \beta + \frac{q}{m} \rho \omega \cos \theta \sin \alpha \cos \beta + \frac{q}{m} \sin \phi \cos \theta \sin \beta - \frac{p}{m} \sin \theta \cos \alpha \cos \beta \\
\alpha &= -\frac{\partial \mathbf{S}}{\partial m} \mathbf{C}_\alpha - p + \tan \beta \rho \cos \alpha + r \rho \sin \alpha - \frac{\rho \omega \sin \alpha}{m} \\
\dot{\beta} &= \frac{\partial \mathbf{S}}{\partial m} \mathbf{C}_\beta + \rho \sin \alpha - \rho \cos \alpha + \frac{\rho \omega \sin \alpha}{m} - \frac{\rho \omega \cos \alpha \sin \theta}{m} \\
\end{align*}
\]

where

\[
\begin{align*}
C_{D_\alpha} &= C_D \cos \beta - C_Y \sin \beta \\
C_{Y_\alpha} &= C_Y \cos \beta + C_D \sin \beta
\end{align*}
\]

and \( \bar{q} = \frac{1}{2} \rho \omega V^2 \) is dynamical pressure, \( \rho \) is the density of air.

(2) Moment Equations (body-axes):

\[
\begin{align*}
\dot{p} &= (c_1 r + c_2 p - c_4 l_p \Omega_p) q + \bar{q} \bar{S} (c_3 C_l + c_4 C_n) \\
\dot{q} &= (c_5 p + c_7 l_p \Omega_p) r - (c_6 p^2 - r^2) + c_7 \bar{q} \bar{S} \Omega_m \\
\dot{r} &= (c_8 p - c_2 r - c_6 l_p \Omega_p) q + \bar{q} \bar{S} (c_9 C_n + c_4 C_l)
\end{align*}
\]

where

\[
\begin{align*}
c_1 &= \frac{1}{I_x} (I_x - I_y I_z) = \frac{I_x I_z - I_x^2}{I_x} \\
c_2 &= \frac{1}{I_y} (I_y - I_x I_z) = \frac{I_y I_z - I_y^2}{I_y} \\
c_3 &= \frac{1}{I_z} (I_z - I_x I_y) = \frac{I_z I_y - I_z^2}{I_z} \\
c_4 &= \frac{1}{I_x} \Gamma_1 = I_x / I_x \\
c_5 &= \frac{1}{I_y} \Gamma_1 = I_y / I_y \\
c_6 &= \frac{1}{I_z} \Gamma_1 = I_z / I_z \\
c_8 &= \frac{1}{I_x} \Gamma_1 = I_x / I_x \\
\end{align*}
\]

(3) Kinematic Equations:

\[
\begin{align*}
\phi &= p + \tan \theta (q \sin \phi + r \cos \phi) \\
\theta &= q \cos \phi - r \sin \phi \\
\psi &= \frac{q \sin \phi + r \cos \phi}{\cos \theta}
\end{align*}
\]

(4) Navigation Equations:

\[
\begin{align*}
\dot{x}_E &= u \cos \psi \cos \theta + v (\cos \psi \sin \theta \sin \phi - \sin \psi \cos \phi) + w (\cos \psi \sin \theta \cos \phi + \sin \psi \sin \phi) \\
\dot{y}_E &= u \sin \psi \cos \theta + v (\sin \psi \sin \theta \sin \phi + \cos \psi \cos \phi) + w (\sin \psi \sin \theta \cos \phi - \cos \psi \sin \phi) \\
\dot{h} &= u \sin \theta - v \cos \theta \sin \phi - w \cos \theta \cos \phi
\end{align*}
\]

where \( u, v, \) and \( w \) are the components of the velocity vector in the body-axes coordinates. They are related to the wind-axes variables by

\[
\begin{align*}
u &= V \cos \alpha \cos \beta \\
\dot{v} &= V \sin \beta \\
\dot{w} &= V \sin \alpha \cos \beta
\end{align*}
\]

II.B. Special Case: Longitudinal Dynamics

For pure longitudinal aircraft motion, we can assume that \( \phi = \psi = \beta = p = r = 0 \). That is, there is no rolling, yawing and side slipping motion. Then equation (1) can be reduced as

\[
\frac{\partial \mathbf{S}}{\partial m} \mathbf{C}_\alpha = \frac{\bar{q} S}{m V_0} C_L + q - \frac{T_0 \alpha}{m V_0} + \frac{q}{V_0} \cos \gamma
\]
where $T_0$ and $V_0$ are trimmed values of $T$ and $V$, respectively. Here we have assumption that when the unmanned flight is landing autonomously, the thrust $T$ and speed $V$ are set to be constants as $T_0$ and $V_0$, respectively. Also, for small $\alpha$, the approximation $\sin \alpha \approx \alpha$ is used to simplify the equation.

Next, we may model the aerodynamic lift coefficient $C_L$ as

$$C_L = C_{L_0} + C_{L\alpha} \alpha$$

where $C_{L_0}$, $C_{L\alpha}$, and other higher order dependencies are neglected. This kind of assumption and approximation is commonly used in most of fixed-wing aircraft modeling and analysis. Moreover, the equation (5) can be simplified as

$$\dot{\theta} = q$$

with assumptions we made before. Define flight path angle $\gamma$ as

$$\gamma = \theta - \alpha$$

It follows that $\dot{\gamma} = q - \dot{\alpha}$. Substitute equation (9) to equation (8), we have

$$\dot{\gamma} = \frac{\bar{q} S}{m V_0} (C_{L_0} + C_{L\alpha} \alpha) + \frac{T_0 \alpha}{m V_0} - \frac{g}{V_0} \cos \gamma$$

The $\dot{q}$ equation given in (3) is obtained using a linear model for the moment coefficient, $C_m$ given by:

$$C_m = C_{m_0} + C_{m\alpha} \alpha + C_{m_q} q + C_{m_{\delta\epsilon}} \delta_{\epsilon}$$

where $\delta_{\epsilon}$ is the elevator deflection. Hence,

$$\dot{q} = c_7 \bar{q} S \bar{c} \left( C_{m_0} + C_{m\alpha} \alpha + C_{m_q} q + C_{m_{\delta\epsilon}} \delta_{\epsilon} \right)$$

It follows that our dynamic system can be expressed as:

$$\begin{array}{l}
\dot{\gamma} = -d_6 - d_5 \gamma - d_4 \cos \gamma + d_3 \dot{\theta} \\
\dot{\theta} = q \\
\dot{q} = d_0 + d_1 q + d_2 \dot{\theta} - d_2 \gamma + d_3 \dot{\delta} \\
\end{array}$$

where

$$\begin{align*}
  d_0 &= c_7 \bar{q} S \bar{c} C_{m_0} \\
  d_1 &= c_7 \bar{q} S \bar{c} C_{m_q} \\
  d_2 &= c_7 \bar{q} S \bar{c} C_{m_{\delta\epsilon}} \\
  d_3 &= c_7 \bar{q} S \bar{c} C_{m_{\delta}} \\
  d_4 &= \frac{g}{V_0} \\
  d_5 &= \frac{\bar{q} S}{m V_0} C_{L_0} + \frac{T_0}{m V_0} \\
  d_6 &= -\frac{\bar{q} S}{m V_0} C_{L\alpha} \\
\end{align*}$$

In the above, for notational simplicity, we have replaced $\delta_{\epsilon}$ by $\delta$ – the control input to the system of interest given in (11). In summary, equations (11) are simplified pure longitudinal dynamic equation, with state variables $\gamma$, $\theta$, $q$ and input term $\delta$, elevator deflection angle.

### III. Damage Modeling and Stability of Non-adaptive Case

#### III.A. Damage Modeling of Aircraft Longitudinal Dynamics

In general, the aerodynamic coefficients $d_0$ through $d_6$ in previous section are determined experimentally. Therefore, those parameters are not the exact number but the approximation with certain range of uncertainty. In order to model parameter uncertainty among $d_0$ to $d_6$, a Damage Model is introduced.

**Damage Modeling:** We model a damage event by additive perturbations to each coefficient given by $\Delta d_i$. That is,

$$\dot{\gamma} = -(d_6 + \Delta d_6) - (d_5 + \Delta d_5) \gamma - (d_4 + \Delta d_4) \cos \gamma + (d_3 + \Delta d_3) \dot{\theta}$$

In general, the aerodynamic coefficients $d_0$ through $d_6$ in previous section are determined experimentally. Therefore, those parameters are not the exact number but the approximation with certain range of uncertainty. In order to model parameter uncertainty among $d_0$ to $d_6$, a Damage Model is introduced.

**Damage Modeling:** We model a damage event by additive perturbations to each coefficient given by $\Delta d_i$. That is,

$$\dot{\gamma} = -(d_6 + \Delta d_6) - (d_5 + \Delta d_5) \gamma - (d_4 + \Delta d_4) \cos \gamma + (d_3 + \Delta d_3) \dot{\theta}$$

In the above, for notational simplicity, we have replaced $\delta_{\epsilon}$ by $\delta$ – the control input to the system of interest given in (11). In summary, equations (11) are simplified pure longitudinal dynamic equation, with state variables $\gamma$, $\theta$, $q$ and input term $\delta$, elevator deflection angle.
\[ \dot{\theta} = q \]  
\[ \dot{q} = (d_0 + \Delta d_0) + (d_1 + \Delta d_1)q + (d_2 + \Delta d_2)\alpha + (d_3 + \Delta d_3)\delta \]  
\[ \text{where } \alpha = \theta - \gamma. \]

Defining the vectors:

\[
\begin{bmatrix}
\dot{W}_\gamma \\
\dot{W}_q
\end{bmatrix} = \begin{bmatrix}
\begin{bmatrix}
\Delta d_6 \\
\Delta d_5 \\
\Delta d_4
\end{bmatrix} \\
\begin{bmatrix}
\Delta d_3 \\
\Delta d_2 \\
\Delta d_1
\end{bmatrix}
\end{bmatrix}, \quad \Phi_\gamma = \begin{bmatrix} 1 \\ \gamma \\ \cos \gamma \\
\theta \end{bmatrix}, \quad \Phi_q = \begin{bmatrix} 1 \\ q \\ \alpha \\ \delta \end{bmatrix}
\]

Equations (12)-(14) can be expressed more succinctly as:

\[
\begin{align*}
\dot{\gamma} &= -\dot{W}_\gamma^T \Phi_\gamma - W_\gamma^{*T} \Phi_\gamma \\
\dot{\theta} &= q \\
\dot{q} &= -\dot{W}_q^T \Phi_q - W_q^{*T} \Phi_q
\end{align*}
\]

**Bounds on Damage Representation:** We have represented the damage model using the additive perturbations, \( \Delta d_i \), as shown in (12) to (14). In most adaptive control schemes, bounds on these terms are required. Let us define the family of perturbations \( D \subset \mathbb{R} \). We assume that we have the following information regarding this family of perturbations \( \mathcal{D} \):

**Assumption 1 (Damage Modeling):** Given the family of perturbations, \( \mathcal{D} \), there exist known positive constants, \( b_{d_i} \), such that for all \( \Delta d_i \in \mathcal{D} \), we have \( \| \Delta d_i \| \leq b_{d_i} < \infty \).

Before proceeding, we define the following bracket notation that will be used repeatedly throughout the development. Let \( x \) and \( y \) be \( m \) dimensional vectors. Then, we define

\[
[x^Ty]_j = \sum_{i \neq j}^n x_i y_i = x^T y - x_j y_j
\]

Hence, using the above notation, we can further express (17)-(19) as:

\[
\begin{align*}
\dot{\gamma} &= -[\dot{W}_\gamma^T \Phi_\gamma]_\theta + d_5 \theta_d - d_5 \dot{\theta} - W_\gamma^{*T} \Phi_\gamma \\
\dot{\theta} &= q_d - q \\
\dot{q} &= -[\dot{W}_q^T \Phi_q]_\delta + d_3 \delta - W_q^{*T} \Phi_q
\end{align*}
\]

where \( \overline{\theta} = \theta_d - \theta, \overline{q} = q_d - q \) and \( \overline{\gamma} = \gamma_d - \gamma \). Here \( \gamma_d \) is desired flight path angle trajectory which will be given from trajectory generator, \( \theta_d \) and \( q_d \) are desired pitch angle and pitch rate which are assigned to be virtual control input. The purpose of the above expression is to highlight (1) the nominal feedforward cancelation terms, \(-[\dot{W}_\gamma^T \Phi_\gamma]_\theta \) and \(-[\dot{W}_q^T \Phi_q]_\delta \) (2) the virtual and actual control components. In this case, \( \theta_d \) and \( q_d \), are viewed as virtual controls of (20) and (21), respectively, and \( \delta \) is viewed as the actual control input.

Backstepping, roughly speaking, attempts to determine what the virtual controls should be if in fact they were actual controls. The actual control is then determined as function of the virtual controls. We point out, however, that our approach is slightly different then that in the literature, in that at each level of the procedure we specifically apply an MRAC control law. In order to do this, we start at the inner-most level which is in contrast to the conventional approach. The inner-most level is defined to be the differential equation in which the control input appears. This will be made clear in the following development.
III.B. Stability of Non-adaptive Case

In this section, we present the baseline stability results for non-adaptive case. The results presented will serve as a basis for which we will extend to RBF neural networks controller design. For non-adaptive case, we have \( \Delta d_i = 0, i = 1 \ldots 6 \), or equivalently, \( W_q^* = W_q^* = 0 \). This yields the following non-damaged set of equations:

\[
\begin{align*}
\dot{\gamma} &= -[\bar{W}^T_q \Phi_q]_d + d_s \theta_d - d_s \bar{\theta} \\
\dot{\theta} &= q_d - q \\
\dot{\bar{q}} &= -[\bar{W}^T_q \Phi_q]_\delta + d_s \delta 
\end{align*}
\] (23)

\[
\begin{align*}
\dot{\gamma} &= -[\bar{W}^T_q \Phi_q]_d + d_s \theta_d - d_s \bar{\theta} \\
\dot{\theta} &= q_d - q \\
\dot{\bar{q}} &= -[\bar{W}^T_q \Phi_q]_\delta + d_s \delta 
\end{align*}
\] (25)

**Forward Step I:** We begin with equation (25). We apply the conventional non-adaptive MRAC formulation, also known as dynamic inversion, to obtain a control law for \( \delta \). This is given by

\[
\delta = d_3^{-1} (\delta_{ff} + \delta_{fb} + \delta_p) 
\] (26)

where

\[
\begin{align*}
\delta_{ff} &= \bar{\delta}_{ff} = \dot{\gamma}_d + [\bar{W}^T_q \Phi_q]_\delta \\
\delta_{fb} &= K_p \tilde{\theta}
\end{align*}
\] (27)

and \( \tilde{q} = q_d - q \). The bar notation given in (27) is used to indicate that the feedforward control is a function of nominal or known terms. In the adaptive case, the feedforward term will contain a nominal component and an adaptive component, which will be RBF neural networks compensator. In the case of conventional dynamic inversion, \( q_d \) is typically the output of a model reference system whose input is the pilots command such as a pitch rate. The term \( \delta_p \) is used to cancel cross coupling terms that arise in the Lyapunov analysis due to the backstepping procedure. It can be shown that in the closed-loop system, these cross-coupling cancellation control terms constitute a skew-symmetric matrix times the state vector. It follows that upon differentiation of a quadratic Lyapunov function that these terms are zero due to the skew-symmetric property. For our purposes, we need not explore this issue further since we focus exclusively on this simple three dimensional outer-loop model.

Substitution of the control law given in (26) into (25) yields

\[
\dot{\bar{q}} = -\delta_{fb} - \delta_p 
\] (29)

**Forward Step II:** We next proceed to equation (24). We select the dynamic inversion control law considering \( q_d \) as the virtual control:

\[
q_d = u_{ff} + u_{fb} + u_p 
\] (30)

where

\[
\begin{align*}
u_{ff} &= \dot{\theta}_d \\
u_{fb} &= K_p \tilde{\theta}
\end{align*}
\] (31)

(32)

The control, \( u_p \), is a used to cancel cross coupling terms that arise in the Lyapunov analysis. Application of this virtual control to (24) yields

\[
\dot{\theta} = -u_{fb} - u_p + \tilde{q} 
\] (33)

**Forward Step III:** We next proceed to equation (25). We select

\[
\theta_d = d_3^{-1} (\sigma_{ff} + \sigma_{fb}) 
\] (34)

where

\[
\begin{align*}
\sigma_{ff} &= \bar{\sigma}_{ff} = \gamma_d + [\bar{W}^T_q \Phi_q]_\theta \\
\sigma_{fb} &= K_p \tilde{\theta}
\end{align*}
\] (28)

This yields

\[
\dot{\gamma} = -\sigma_{fb} + d_s \bar{\theta} 
\] (35)

To prove the stability, we select the following Lyapunov function candidate.

\[
V(\bar{q}, \tilde{\theta}, \gamma) = \frac{1}{2} \bar{q}^2 + \frac{1}{2} \tilde{\theta}^2 + \frac{1}{2} \gamma^2
\]

6 of 23

American Institute of Aeronautics and Astronautics
This function is positive definite. Differentiating we get
\[ V = -K_p \dot{q}^2 - K_p \dot{\theta}^2 - K_{pr} \dot{\gamma}^2 - \ddot{q} \delta_p + \ddot{\theta} - u_p \dot{\gamma} + ds \gamma \dot{\gamma} \]  
(36)

Selecting
\[ \delta_p = \dot{\theta} \quad u_p = ds \gamma \]  
(37)

it follows that
\[ \dot{V} = -K_p \dot{q}^2 - K_p \dot{\theta}^2 - K_{pr} \dot{\gamma}^2 < 0 \]

Hence, the closed loop non-damaged system is asymptotically stable. Later, the damaged nonlinear system control strategy will be presented by introducing RBF neural networks to compensate for the unknown system error.

IV. Fully Tuned Growing RBF Neural Network Controller Design

In this section, we assume that the perturbations, \( W_{sT}^\gamma \Phi_\gamma \) and \( W_{sT}^q \Phi_q \) are non-zero and represent possible damage to the aircraft. The system dynamic equations used for the longitudinal modes are (20) to (22) instead of (23) to (25). Compare with those two sets of equations, the unknown but bounded perturbations term \( W_{sT}^\gamma \Phi_\gamma \) and \( W_{sT}^q \Phi_q \) are the only difference and they may disturb the feedback control performance. The RBF neural network controllers for both \( W_{sT}^\gamma \Phi_\gamma \) and \( W_{sT}^q \Phi_q \) are introduced to perform estimation since RBF neural networks has an excellent approximation property to nonlinear function.\(^{15}\) The RBF neural networks controller design will be derived for the continuous case, then discretized for the closed loop system, with the growing neuron mechanism.

IV.A. Fully Tuned RBF Neural Network Backstepping Controller, Continuous Case

Typically, RBF neural networks can be expressed as \( \hat{f} = W^T \phi + \epsilon_N \), where \( W \in \mathbb{R}^{N \times n} \) and \( \phi = [\phi_1(x) \phi_2(x) \ldots \phi_N(x)]^T \). Here \( N \) is the number of neurons, \( n \) is the output dimension and \( \epsilon_N \) is approximation error, depending on the number of neuron \( N \). For each neuron \( \phi_i(x) \), the expression is
\[ \phi_i(x) = \exp(-\frac{\|x - \mu_i\|^2}{\sigma_i^2}) \]
where \( \mu_i \) and \( \sigma_i^2 \) are the center and variance of the RBF neuron, respectively. Thus \( \phi \) is the stack vector of each neuron \( \phi_i \).

For this particular problem, define \( W_{sT}^\gamma \Phi_\gamma \) and \( W_{sT}^q \Phi_q \) are going to be approximated by \( W_{sT}^\gamma \Phi_\gamma \) and \( W_{sT}^q \Phi_q \), respectively, where \( W_{sT}^\gamma \) and \( W_{sT}^q \) are weights and \( \phi_1^\gamma \) and \( \phi_2^\gamma \) are ideal neuron output vectors. Since these two approximated neural network outputs are scalars, the two weights \( W_1^\gamma \) and \( W_2^\gamma \) are vectors. The star notation indicates that \( W_1^\gamma \phi_1^\gamma \) and \( W_2^\gamma \phi_2^\gamma \) are ideal weights and neural network output vectors. We define the vectors \( W_1 \), \( W_2 \), \( \phi_1 \) and \( \phi_2 \) as the neural network approximation of \( W_{sT}^\gamma \Phi_\gamma \) and \( W_{sT}^q \Phi_q \) are approximated by \( W_1^\gamma \phi_1 \) and \( W_2^\gamma \phi_2 \), respectively. Therefore, we have equations:
\[ W_{sT}^\gamma \Phi_\gamma = W_1^\gamma \phi_1^\gamma + \epsilon_{1,N_1} \]
\[ W_{sT}^q \Phi_q = W_2^\gamma \phi_2^\gamma + \epsilon_{2,N_2} \]

where \( \epsilon_{1,N_1} \) and \( \epsilon_{2,N_2} \) are RBF neural networks approximation errors, which depend on number of neuron \( N_1 \) and \( N_2 \) for each case.

Then we define the error term \( \hat{W}_i = W_i^s - W_i \), \( \hat{\phi}_i = \phi_i^s - \phi_i \) for \( i = 1,2 \) and propose the following backstepping controller with RBF neural network term:
\[ \theta_d = d_2^{-1} (\Delta_{ff} + \sigma_{fb} + W_2^\gamma \phi_2) \]
\[ q_d = \Delta_{ff} + u_{fb} + \Delta_p \]
\[ \delta = d_2^{-1} (\Delta_{ff} + \Delta_{fb} + \dot{\theta}_p + W_2^\gamma \phi_1) \]

where everything on the right hand side of the equations, except for RBF neural networks term \( W_{sT}^\gamma \Phi_\gamma \) and \( W_{sT}^q \Phi_q \), are defined from equations (26) to (35). In fact, \( W_1^\gamma \phi_1 \) and \( W_2^\gamma \phi_2 \) are the adaptive term added to \( \Delta_{ff} \) and \( \sigma_{ff} \). To find the tuning algorithm for \( W_i \) and \( \phi_i \) (\( i = 1,2 \)), propose a Lyapunov function candidate
\[ V = \frac{1}{2} q^2 + \dot{\theta}^2 + \dot{\gamma}^2 + \phi_1^T G_1^{-1} \phi_1 + \phi_2^T G_2^{-1} \phi_2 + \hat{W}_1^T F_1^{-1} \hat{W}_1 + \hat{W}_2^T F_2^{-1} \hat{W}_2 \]
where $F_1, G_1$ and $F_1, F_2$ are positive definite matrix which can determine the backstepping RBF neural networks control performance. Differentiate function $V$ with respect to time, we have

$$V = \dot{q} + \dot{\theta} + \dot{\gamma} + W_1 F^{-1}_1 \dot{\hat{W}}_1 + W_2 F^{-1}_2 \dot{\hat{W}}_2 + \phi_1^T G^{-1}_1 \dot{\hat{\phi}}_1 + \phi_2^T G^{-1}_2 \dot{\hat{\phi}}_2$$

Substitute (38), (26) to (35) and

$$W_i^T \Phi_i = W_i^T \phi_i + \epsilon_{1,N_i} $$
$$W_i^T \Phi_{i,q} = W_i^T \phi_{i,q} + \epsilon_{2,N_i}$$

to the expression of $V$, we have

$$V = -K_{p_0} \dot{q}^2 - K_{p_0} \dot{\theta}^2 - K_{p_7} \dot{\gamma}^2 + \dot{q} (W_2^T \phi_{i}^* - W_2^T \phi_i + \epsilon_{2,N_i}) + \dot{\theta} (W_1^T \phi_{i}^* - W_1^T \phi_i + \epsilon_{1,N_i}) + W_1 F^{-1}_1 \dot{\hat{W}}_1 + W_2 F^{-1}_2 \dot{\hat{W}}_2 + \phi_1^T G^{-1}_1 \dot{\hat{\phi}}_1 + \phi_2^T G^{-1}_2 \dot{\hat{\phi}}_2$$

For $W_i^T \phi_{i}^* - W_i^T \phi_i, \ i = 1, 2$, we have

$$W_i^T \phi_{i}^* - W_i^T \phi_i = W_i^T \phi_{i}^* - W_i^T \phi_{i} + W_i^T \phi_i - W_i^T \phi_i$$

$$= W_i^T \phi_i + W_i^T \phi_i - W_i^T \phi_i$$

$$= W_i^T \phi_i + W_i^T \phi_i - \omega_i$$

where $\omega_i = W_i^T \phi_i$ are bounded based on the assumption of RBF neural networks.\textsuperscript{10-12} Then substitute this expression to $V$ and rearrange the term, we have

$$V = -K_{p_0} \dot{q}^2 - K_{p_0} \dot{\theta}^2 - K_{p_7} \dot{\gamma}^2 + \dot{q} (W_2^T \phi_{i}^* - W_2^T \phi_i + \epsilon_{2,N_i}) + \dot{\theta} (W_1^T \phi_{i}^* - W_1^T \phi_i + \epsilon_{1,N_i}) + W_1 F^{-1}_1 \dot{\hat{W}}_1 + W_2 F^{-1}_2 \dot{\hat{W}}_2 + \phi_1^T G^{-1}_1 \dot{\hat{\phi}}_1 + \phi_2^T G^{-1}_2 \dot{\hat{\phi}}_2$$

where $\omega_1 = \omega_1 + \epsilon_{1,N_i}$ and $\omega_2 = \omega_2 + \epsilon_{2,N_i}$. Then, assign

$$\dot{\phi}_1 + F^{-1}_1 \dot{\hat{W}}_1 = 0$$
$$\gamma \phi_2 + F^{-1}_2 \dot{\hat{W}}_2 = 0$$
$$q \phi_1 + G^{-1}_1 \dot{\hat{\phi}}_1 = 0$$
$$\gamma W_2 + G^{-1}_2 \dot{\hat{\phi}}_2 = 0$$

which indicate the tuning rules for RBF neural networks controller as

$$W_1 = F_1 \dot{\phi}_1$$
$$W_2 = F_2 \gamma \phi_2$$
$$\phi_1 = G_1 q \phi_1$$
$$\phi_2 = G_2 \gamma W_2$$

Then the time derivative of $V$ is reduced to

$$\dot{V} = -K_{p_0} \dot{q}^2 - K_{p_0} \dot{\theta}^2 - K_{p_7} \dot{\gamma}^2 + \dot{q} \omega_1 + \gamma \omega_2$$

$$\leq -K_{min} ||e||^2 + N ||e||$$

where

$$\Omega = \begin{bmatrix} \omega_1 \\ 0 \\ \omega_2 \end{bmatrix}$$
$$e = \begin{bmatrix} \dot{q} \\ \dot{\theta} \\ \dot{\gamma} \end{bmatrix}$$
$$\begin{bmatrix} K_{p_0} \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \end{bmatrix}$$

$$N \geq \sqrt{\omega_1^2 + \omega_2^2}$$
$$K_{min} = \min\{K_{p_0}, K_{p_7}, K_{p_7}\}$$

8 of 23

American Institute of Aeronautics and Astronautics
and \( \tilde{\omega}_1 \) and \( \tilde{\omega}_2 \) are bounded based on our assumption. Then we can prove that the control strategy (38) with RBF neural networks tuning rule (39) can lead \( \| e \| \) to be UUB. That is, \( \exists B = \frac{N}{K_{\text{min}}} > 0 \) such that when \( \| e \| > B, V \leq -K_{\text{min}} \| e \|^2 + N \| e \| < 0 \).

Up to this point, the new proposed fully tuned RBF neural networks control law has been derived for continuous case, which is not practical in flight path tracking control. In next subsection, the discretization process will be introduced to the tuning rule (39) such that the tuning rule for center and variation of the RBF neural networks parameters will be expressed explicitly.

**IV.B. Discretization and Growing Neuron Mechanism**

Recall the tuning rule (39), the discretized weight update equation can be expressed as

\[
\begin{align*}
W_{1,n+1} &= W_{1,n} + \eta_1 \phi_{1,n} \tilde{q}_n \\
W_{2,n+1} &= W_{2,n} + \eta_2 \phi_{2,n} \tilde{\gamma}_n
\end{align*}
\]

(40)

where \( \eta_1 \) and \( \eta_2 \) are design parameters, which are usually called learning rate and sub-index \( n \) stands for step index. According to the definition of damage model in equation (15) and (16), the input for the first RBF neural networks \( \phi_1 \) is \( x_1 = [1 \, \gamma \, \cos \theta]^T \), the input for the second RBF neural networks \( \phi_2 \) is \( x_1 = [1 \, q \, \gamma \, \theta]^T \).

Similarly, using equation (39) with the gradient descent method, the discretized center and variation update rules are

\[
\begin{align*}
\mu_{1,n+1}^{k} &= \mu_{1,n}^{k} + \eta_1 \frac{\partial g_1}{\partial \mu_{1,n}^{k}} \tilde{q}_n \\
\mu_{2,n+1}^{k} &= \mu_{2,n}^{k} + \eta_2 \frac{\partial g_2}{\partial \mu_{2,n}^{k}} \tilde{\gamma}_n \\
\sigma_{1,n+1}^{k} &= \sigma_{1,n}^{k} + \eta_1 \frac{\partial g_1}{\partial \sigma_{1,n}^{k}} \tilde{q}_n \\
\sigma_{2,n+1}^{k} &= \sigma_{2,n}^{k} + \eta_2 \frac{\partial g_2}{\partial \sigma_{2,n}^{k}} \tilde{\gamma}_n
\end{align*}
\]

(41)

where \( g_i = \sum_{j=1}^{N_i} w_{i,j} \phi_{i,j}(x_{i,n}) (i = 1, 2) \) and \( \eta_1, \eta_2, \eta_3, 1 \) and \( \eta_3, 2 \) are learning rates. Here, \( w_{i,j} \) is the \( j-th \) element of weight \( W_i \) \( (i = 1, 2) \). \( \phi_{i,j}(x_{i,n}) \) is the \( j-th \) element of RBF neural networks output \( \phi_i \). The sup-index \( k \) in (41) indicates the \( k-th \) neuron, \( N_i \) is the number of neuron. Using the chain rule of derivatives, we have the explicit expression of RBF neural network parameter update rules

\[
\begin{align*}
\mu_{1,n+1}^{k} &= \mu_{1,n}^{k} - \eta_1 w_{1,j}^{T} \frac{2(x_{i,n} - \mu_{1,n}^{k})}{\sigma_{1,n}^{k}} \tilde{q} \\
\mu_{2,n+1}^{k} &= \mu_{2,n}^{k} - \eta_2 w_{2,j}^{T} \frac{2(x_{i,n} - \mu_{2,n}^{k})}{\sigma_{2,n}^{k}} \tilde{\gamma} \\
\sigma_{1,n+1}^{k} &= \sigma_{1,n}^{k} - \eta_1 w_{1,j}^{T} \frac{2\| x_{i,n} - \mu_{1,n}^{k} \|^2}{\sigma_{1,n}^{k}} \tilde{q} \\
\sigma_{2,n+1}^{k} &= \sigma_{2,n}^{k} - \eta_2 w_{2,j}^{T} \frac{2\| x_{i,n} - \mu_{2,n}^{k} \|^2}{\sigma_{2,n}^{k}} \tilde{\gamma}
\end{align*}
\]

(42)

Equation (40) and equation (42) show the tuning rules for a fully tuned RBF neural network controller in the discrete case, which can be applied to simulation and real control engineering applications. Notice that the learning rates \( \eta_{1,i}, \eta_{2,j} \) and \( \eta_{3,i} \) \( (i = 1, 2) \) are qualitatively equivalent to the positive definite matrix \( F_1, F_2, G_1 \) and \( G_2 \) in equation (39). However, in this RBF neural network control law, the number of neurons is fixed during the control training process. Normally, the number of neurons \( N_i \) \( (i = 1, 2) \) has a positive effect on approximation error \( \varepsilon_{N_i} \), meaning that when \( N_i \) increases, \( \varepsilon_{N_i} \) decreases. But in some cases, the controller does not need as many neurons as originally fixed. Therefore, the concept of growing neurons is introduced. In this idea, a small number of neurons are assigned to the controller, then, according to the criteria of growing neurons, the number of neurons is increased automatically to meet the suitable performance.

**Criteria for adding one neuron**: One new neuron will be added into the current network as long as all of these following criteria are satisfied: (The sub-index \( i = 1, 2 \) indicates for different RBF neural networks \( W_i^{T} \phi_i \))

1. **Current estimation error criteria:**

\[
\begin{align*}
\| \tilde{q}_n \| &> E_{1,1} \\
\| \tilde{\gamma}_n \| &> E_{2,1}
\end{align*}
\]
where $E_{1,1}$ and $E_{2,1}$ are pre-defined threshold values.

(2) **Novelty criteria:**

$$\inf \| x_{i,n} - \mu_{i,n} \| > E_{i,2}$$

where $\inf \| x_{i,n} - \mu_{i,n} \|$ is the minimum deviation from the current estimation state to the center of RBF over all centers of RBF neurons. $E_{i,2}$ is pre-defined threshold value. Here, $k = 1, 2, \ldots, N$, where $N$ is the number of neurons.

(3) **Window root mean square (RMS) error criteria:**

$$\frac{1}{N_w} \sum_{j=n-N}^{n} \| e_j \|^2 > E_3$$

where $\| e_j \|$ is the norm of tracking error, $N_w$ is the window size and $E_3$ is the pre-defined threshold value.

Then, after determining to add one neuron, the parameter assignment rules are listed as below, as proposed in.\textsuperscript{3,18}

**Parameter configuration for new neuron:**

1. The new neurons RBF center is $\mu_{i,N_i+1} = x_{i,n}$.
2. The new neurons RBF variance is $\sigma_{i,N_i+1} = \kappa \| x_{i,n} - \mu_{i,N_i+1} \|$.
3. The new column of the weight matrices are $w_{1,N_i+1} = q_n$ and $w_{2,N_i+1} = \theta_n$.

Here, $\kappa$ are pre-defined positive values called overlap factor, and $N_i$ is the number of neurons before adding one new neuron.

**V. Auto-landing Simulation**

**V.A. System Identification**

The longitudinal UAV model used in this paper was derived from system identification data of the NASA Ames Exploration Aerial Vehicle (EAV). A brief description of the EAV is shown below in Figure 1.

![NASA Ames Exploration Aerial Vehicle Overview](image)

The dimensional stability and control derivatives were derived using the least squares method. The derivatives for the EAV are given in Table 2. The plots in Figure 2 show the model fit versus the actual flight data. Then, use the equation 43 cited from book,\textsuperscript{16} we can compute the coefficients $d_0$ to $d_6$ appeared in equation (11), where $\bar{q} = \frac{1}{2} \rho_{air} V_0^2$ is the dynamic pressure. Here, the aerodynamical coefficients $C_{D_u}$ and $C_{L_u}$ are approximated to be zero for low speed aircraft.\textsuperscript{16} ($V_0 \ll V_{\text{sound}}$)

$$
\begin{align*}
X_u &= -\bar{q} \delta (C_{D_u} + 2C_{D_0}) \\
X_w &= -\bar{q} \delta (C_{D_u} - C_{D_0}) \\
Z_u &= -\bar{q} \delta (C_{L_u} + 2C_{L_0}) \\
Z_w &= -\bar{q} \delta (C_{L_u} - 2C_{L_0}) \\
M_w &= C_{m_1} \frac{\delta S}{\delta_{\text{flap}}} \\
M_q &= C_{m_2} \frac{\delta S^2}{\delta_{\text{hull}}} \\
M_{\delta e} &= C_{m_3} \frac{\delta S}{\delta_e}
\end{align*}
$$

(43)
$$X_u \quad X_v \quad Z_u \quad Z_w$$

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$-0.0665s^{-1}$</td>
<td>$-0.2893s^{-1}$</td>
<td>$0.0980s^{-1}$</td>
<td>$-1.7836s^{-1}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$M_u$</th>
<th>$M_v$</th>
<th>$M_q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-0.0201m^{-1}s^{-1}$</td>
<td>$0.8228m^{-1}s^{-1}$</td>
<td>$-0.6997s^{-1}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$X_{\delta_e}$</th>
<th>$Z_{\delta_e}$</th>
<th>$M_{\delta_T}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$4.7690m/s^2$</td>
<td>$28.3783m/s^2$</td>
<td>$-0.05453s^{-2}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$X_{\delta_T}$</th>
<th>$Z_{\delta_T}$</th>
<th>$M_{\delta_e}$</th>
<th>$U_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$3.1933m/s^2$</td>
<td>$0.1438m/s^2$</td>
<td>$-14.5874s^{-2}$</td>
<td>$-13.144m/s$</td>
</tr>
</tbody>
</table>

Table 2. EAV Dimensional Stability and Control Derivatives.

Figure 2. EAV Model and Simulation Fit, Longitudinal Doublets.

(a) Elevator Doublet  
(b) Throttle Doublet
V.B. Auto-landing Trajectory Generation

In this section, we present an auto-landing trajectory command for desired flight path angle $\gamma_d$. A typical landing process for aircraft is divided into four states: approaching, flare, touchdown and after-landing roll. Generally speaking, the first two states, approaching and flare, are the most important states during landing process. After touchdown to the runway, the aircraft can be easily decelerated and stopped. Therefore, in this paper, we only concern about the trajectory generation and tracking control for approaching and flare states.

Usually the landing trajectory is planned through the flight path, which is the declination angle about the real flight path. For flight path angle, we have the relationship

$$\frac{dy_d}{dx_d} = \tan \gamma_d$$

where $y_d = y_d(x_d)$ is the landing trajectory in earth coordinate system. As shown in Fig. 3, $(x_1, H_0)$ is the point that aircraft will start landing. $(x_0, h_0)$ is the point that aircraft landing is transited from approaching state to flare state.

![Figure 3. Flight landing trajectory](image)

For the approaching state, the height will decrease with constant speed, constant declination angle, and with the aircraft in a steady state flight condition. Therefore, the flight path is a straight line with path angle $\gamma_0 = \gamma$. In general case, this constant path angle $\gamma_0$ is set to be $-3^\circ$, which can perform a stable and safe landing.

For the flare state, the aircraft will gradually decrease the descent angle $\gamma_d$ from approaching state $\gamma_0$ to prepare touchdown. Usually, the landing trajectory flare state will be modeled as an exponential curve with respect to horizontal position $x_d$ and height $y_d$. This kind of trajectory can give the aircraft a very smooth transition from approaching to touchdown. Followed by schematic diagram shown in Fig. 3, the parametric equations for landing trajectory can be expressed as

$$x_d = x_0 + V_0 t$$

$$y_d = h_0 e^{-\frac{x_d-x_0}{\tau}}$$

where $\tau$ is time constant for exponential flare trajectory. Here, we use the approximation $V_0 \cos \gamma_d \approx V_0$ for small flight path angle $\gamma_d$. Combining these two equations in (44), we have the explicit form of landing trajectory for flare mode:

$$y_d = h_0 e^{-\frac{x_d-x_0}{\tau V_0 \gamma_0}}$$

Since flight path angle $\gamma_d = \frac{dy_d}{dx_d}$, we have the expression for flight path angle $\gamma_d$:

$$\gamma_d = -\frac{h_0}{V_0 \tau} e^{-\frac{x_d-x_0}{\tau V_0 \gamma_0}}$$

Here, we have continuity property for landing trajectory as $\gamma_0 = \gamma |_{x_d=x_0}$. Therefore, we have $h_0 = -V_0 \tau \gamma_0$.

Differentiate the second equation of (44), we have the expression about the sink rate $\dot{y}_d$ as

$$\dot{y}_d = -\frac{h_0}{\tau} e^{-\frac{x_d-x_0}{\tau V_0 \gamma_0}}$$

$$\dot{y}_d = -\frac{h_0}{\tau} e^{-\frac{1}{\tau}}$$
Substitute equation (46) into (47) at touchdown point, we have

\[ \dot{y}_{TD} = V_0 \gamma_T \]

where \( \dot{y}_{TD} \) and \( \gamma_T \) are the sink rate and flight path angle at touchdown point, respectively. Actually this equation can also be derived from velocity decomposition into vertical direction \( \dot{y}_{TD} = V_0 \sin \gamma_T \) and approximated equation:

\[ \sin \gamma_T : \approx \gamma_T \].

In order to perform safe and smooth landing, the sink rate \( \dot{y}_{TD} \) between \(-1 \) ft/s and \(-2 \) ft/s is highly recommended. Then the desired flight path angle at touchdown \( \gamma_T \) can be determined by reformatting the equation (48) as

\[ \gamma_T = \frac{\dot{y}_{TD}}{V_0} \] (48)

In order to make the aircraft landing safe and smooth, not only the sink rate, but also the expected horizontal position of touchdown point \((x_d - x_0)_{exp}\), should also be carefully selected. Then, substituting \((x_d - x_0)_{exp}\) into equation (46), we have

\[ \gamma_d = -\frac{h_0}{V_0} e^{-\frac{(x_d - x_0)_{exp}}{V_0}} \]

Solve for \( \tau \), we have

\[ \tau = -\frac{(x - x_0)_{exp}}{V_0} \left[ \ln \left( \frac{\gamma_T}{\gamma_0} \right) \right]^{-1} \] (49)

In summary, the landing trajectory for flight path angle \( \gamma_d \) is set to be

\[ \gamma_d = \begin{cases} \gamma_0 & \text{when } x_d \leq x_0 \\ \gamma = -\frac{h_0}{V_0} e^{-\frac{x_d - x_0}{V_0}} & \text{when } x_d > x_0 \end{cases} \] (50)

The parameters for the desired trajectory expressed in (50) are summarized in Table 3. The sample flight paths generated from the criteria shown in this section are shown in Fig. 4.

<table>
<thead>
<tr>
<th>Table 3. Parameter setup for landing trajectory generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \gamma_0 )</td>
</tr>
<tr>
<td>( \gamma_T )</td>
</tr>
<tr>
<td>( (x - x_0)_{exp} )</td>
</tr>
<tr>
<td>( V_0 )</td>
</tr>
<tr>
<td>( H_0 )</td>
</tr>
<tr>
<td>( \gamma_T )</td>
</tr>
<tr>
<td>( \tau )</td>
</tr>
<tr>
<td>( h_0 )</td>
</tr>
<tr>
<td>( x_0 )</td>
</tr>
<tr>
<td>( x_1 )</td>
</tr>
</tbody>
</table>

### V.C. Auto-landing Simulation Results

In this subsection, we simulate the fully tuned growing RBF neural network controller for both the non-damaged and damaged longitudinal UAV model. In each simulation case, two flight path tracking models are simulated: auto-landing trajectory and sine shape trajectory.

#### Case I: Non-damage flight path tracking

In the non-damaged flight path tracking problem, the longitudinal UAV model can be simplified as equations (23) to (25), with system parameters identified in subsection V.A. Applying fully tuned growing RBF neural networks controller in (38) with discretized tuning rules (40) and (42) to non-damage UAV model (23) to (25), we can build up a closed-loop system with control input (38) and plant dynamic equations (23), (24) and (25).

The simulation results for non-damaged UAV auto-landing trajectory tracking control are presented in Fig. 5 to Fig. 7, where Fig. 5 shows the tracking results for all state variables \( \gamma, \theta, q \) and control input \( \delta \), Fig. 6 shows the
number of neurons with respect to time and Fig. 7 shows the flight path and flight path angle tracking result with respect to height and horizontal position.

The simulation results for non-damaged UAV sine shape trajectory tracking control are presented in Fig. 8 to Fig. 10, where Fig. 8 shows the tracking results for all state variables $\gamma$, $\theta$, $q$ and control input $\delta$, Fig. 9 shows the number of neurons with respect to time and Fig. 10 shows the flight path and flight path angle tracking result with respect to height and horizontal position.

Case II: Damaged flight path tracking

In the damaged case, the longitudinal UAV model becomes the equations (20) to (22), with system parameters identified in subsection V.A. Applying the fully tuned growing RBF neural networks controller in (38) with discretized tuning rules (40) and (42) to the damaged UAV model (20) to (22), we can build up a closed-loop system with control input (38) and plant dynamic equations (20), (21) and (22). In this damaged model simulation, the system parameters $d_1, d_3$ in (11) are set to be 20% and 2% deviation from nominal values, respectively.

The simulation results for damaged UAV auto-landing trajectory tracking control are presented in Fig. 11 to Fig. 13, where Fig. 11 shows the tracking results for all state variables $\gamma$, $\theta$, $q$ and control input $\delta$, Fig. 12 shows the number of neurons with respect to time and Fig. 13 shows the flight path and flight path angle tracking result with respect to height and horizontal position.

The simulation result for damaged UAV sine shape trajectory tracking control is presented in Fig. 14 to Fig. 16, where Fig. 14 shows the tracking results for all state variables $\gamma$, $\theta$, $q$ and control input $\delta$, Fig. 15 shows the number of neuron with respect to time and Fig. 16 shows the flight path and flight path angle tracking result with respect to height and horizontal position.

Comparing damaged case with non-damaged case, the tracking performances are both very good, with very limited tracking error in both auto-landing and sine shape flight path trajectory. Moreover, in the damaged case, the disturbance from the sudden change of system parameters can be easily rejected by the fully tuned growing RBF neural network controller. Notice that the number of neurons in the damaged case is larger than in the non-damaged case, and the number of neurons in the damaged case will increase once the system parameters undergo sudden change. This phenomena makes sense, in that once the disturbance is generated by sudden change of system parameters, the tracking error grows larger, which activates the growing neuron function.

VI. Conclusion

The fully tuned growing RBF neural network controller is developed in this paper with the application of longitudinal UAV auto-landing and sine shape backstepping flight path tracking control. The fully tuned RBF neural network controller does not require the precise knowledge of system dynamics. the RBF neural network term can compensate
Figure 5. Auto-landing trajectory tracking result for non-damaged UAV model: time history plot
Figure 6. Auto-landing trajectory tracking result for non-damaged UAV model: number of neuron

Figure 7. Auto-landing trajectory tracking result for non-damaged UAV model: flight path tracking plot
Figure 8. Sine shape trajectory tracking result for non-damaged UAV model: time history plot
Figure 9. Sine shape trajectory tracking result for non-damaged UAV model: number of neuron

Figure 10. Sine shape trajectory tracking result for non-damaged UAV model: flight path tracking plot
Figure 11. Auto-landing trajectory tracking result for damaged UAV model: time history plot
Figure 12. Auto-landing trajectory tracking result for damaged UAV model: number of neuron

Figure 13. Auto-landing trajectory tracking result for damaged UAV model: flight path tracking plot
Figure 14. Sine shape trajectory tracking result for damaged UAV model: time history plot
Figure 15. Sine shape trajectory tracking result for damaged UAV model: number of neuron

Figure 16. Sine shape trajectory tracking result for damaged UAV model: flight path tracking plot
for errors in the system parameters. Compared with the conventional RBF controllers, the centers, variations and number of neurons are no longer fixed, instead, these parameters will automatically change with respect to error information. Therefore, the parameter selection can be simplified requiring only the setting of an initial value. To verify the proposed controller for a longitudinal UAV model with damaged system model, two types of flight path (auto-landing trajectory and sine shape trajectory) tracking control simulation were presented for both non-damage and damaged UAV model. The trajectory tracking performances demonstrated the effectiveness of the proposed controller.

References

5. Corey Ippolito, Yoo-Hsiu Yeh, and John Kaneshige. Neural adaptive flight control testing on an unmanned experimental aerial vehicle.
19. Y. Yeh, P. Kumar, C. Ippolito, and A.K. Ishihara. Hardware implementation of cots avionics on unmanned aerial vehicle platforms. AIAA Infotech@Aerospace, April 2010.