JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 30, No. 3, May–June 2007

Adaptive Nonlinear Neural Controller for Aircraft Under Actuator Failures

A. A. Pashilkar*
National Aerospace Laboratories, Bangalore 560 017, India
and
N. Sundararajan† and P. Saratchandran‡
Nanyang Technological University, Singapore 639798, Republic of Singapore

DOI: 10.2514/1.26540

This paper presents an adaptive backstepping neural controller design for aircraft under control surface failures. The control scheme uses radial basis function neural networks in an adaptive backstepping architecture with a full state measurement for trajectory following. The requirement for stability is separated from the network-learning part. This allows us to use any function approximation scheme (including neural networks) for learning. For the radial basis function neural networks, a learning scheme in which the network starts with no neurons and adds new neurons based on the trajectory error is developed. Stable tuning rules are derived for the update of the centers, widths, and weights of the radial basis function neural networks. Using Lyapunov's theory, a proof of stability in the ultimate bounded sense is presented for the resulting controller. The fault-tolerant controller design is illustrated for an unstable high performance aircraft in the terminal-landing phase subjected to multiple control surface failures of hard over type and severe winds. The design uses the full 6-degree-of-freedom nonlinear aircraft model, and the simulation studies show that the above controller is able to successfully stabilize and land the aircraft within tight touchdown dispersions.

I. Introduction

THE flight control system (FCS) in modern aircraft should be designed for adequate performance and stability under both normal and failure conditions (such as control surface failures). Control reconfiguration in the face of such failures can be classified into two broad categories: 1) reconfiguration based on fault detection isolation and accommodation [1], and 2) reconfiguration based on online learning of the system dynamics and using it in an adaptive controller. The main disadvantage of the first approach is that it can handle only anticipated abnormalities/failures. The second approach has the advantage that it has the potential to handle unanticipated abnormalities/failures as well. Neural network-based controllers [2–8] are particularly suited to the second approach because of their ability to learn online. They have thus become very popular. Before describing the proposed method, a brief review of neural network-based controllers for aircraft is given.

Pesonen et al. [2] present an adaptive neural network controller for longitudinal dynamics of a general aviation aircraft. The baseline controller is an inversion-based design, which achieves control decoupling between velocity and pitch attitude. The inversion controller incorporates a priori knowledge of the aircraft dynamics. Adaptive neural networks correct modeling errors in this controller by modifying the outer loop proportional integral derivative commands. A main limitation of this approach is that it does not present any architecture for trajectory following as only the pitch rate and velocity loops are designed (i.e., innermost loops). Ferrari and Stengel [3] proposed a nonlinear control system comprising a network of networks which was taught by a two-phase learning procedure realized through a novel training technique and an adaptive critic design. This technique requires a significant design effort for linear controller design as well as training the networks. Calise et al. [4] developed a direct adaptive tracking control scheme for reconfigurable flight control for tailless aircraft using neural networks and applied it to design the CAS (control augmentation system) of an X-36 aircraft under control surface actuator failures. Adaptive backstepping (Krstic et al. [5]) is a well-known design technique which can be applied to control nonlinear uncertain systems. More recently, Hovakimyan et al. [6] consider adaptive output feedback control of uncertain nonlinear systems, in which both the dynamics and the dimension of the regulated system may be unknown. However, it is assumed that the relative degree of the regulated output is known. They show that it is sufficient to build an observer for the output tracking error in case the states are not available for feedback. The error signals are ultimately bounded as shown through Lyapunov's direct method. It is assumed that for the outputs considered, the zero dynamics of the system are stable. The theoretical results are illustrated in the design of a controller for a high-bandwidth attitude command system for an unmanned R-50 helicopter. The neural network representing the approximate inverse dynamics must be trained offline.

In the recent work of Shin and Kim [7], an adaptive controller based on neural networks, which compensates for the effects of the aerodynamic modeling error, is proposed and subsequently applied to a full dynamic model of an F-16. The major component of the control signal is generated by an inversion design, which is made robust by a neural network, which corrects for the modeling error. The neural networks require gradients of the preceding network outputs with respect to some of the state variables increasing complexity of the controller. A different approach is taken to the problem of control design for nonaffine plants in Boskovic et al. [8]. They differentiate the system equations and show that it is possible to synthesize a controller for the resulting higher order system, which is affine in terms of the derivative of the control signal. However, a drawback of this approach is that the class of nonaffine models considered contains unknown parameters that appear linearly as products with the nonlinear terms. Further, it is assumed that the nonlinearities themselves are known perfectly and introduced in the update rule for the uncertain parameters.

Received 14 July 2006; revision received 4 December 2006; accepted for publication 11 December 2006.

*Scientist, Flight Mechanics and Control Division; apash@css.nal.res.in.

†Professor, School of Electrical and Electronic Engineering; ensundara@ntu.edu.sg.

‡Associate Professor, School of Electrical and Electronic Engineering; epsarat@ntu.edu.sg.

Recently Ge and Zhang [9] developed an adaptive controller for single input/single output (SISO) nonaffine nonlinear systems using multilayer neural networks. When the system states are not available, they show that it is possible to design a high gain observer to obtain them. Their controller with suitable update rules results in all signals being ultimately bounded. The system is assumed to have a strong relative degree less than its order and its zero dynamics are stable. The key assumption used by them is that the plant input gain does not change sign and is bounded. The assumption that zero dynamics are stable may not be generally true for nonlinear systems and limits the application of this approach. In this paper we show how by using some of the same assumptions it is possible to use adaptive backstepping and overcome the limitation in [9].

Bugajski and Enns [10] have used the backstepping approach for nonlinear dynamic inversion control. A feature of this approach is to do away with the demanded derivative of the state at each step of the dynamic inversion process and replace it with a signal proportional to the error in the state. The nonlinear dynamic inversion is made robust by these proportional feedback gains. This removes the problem of high frequency signals arising out of numerical differentiation propagating down the neural network controller and causing undesirable actuator saturation. The approach requires that the control designer separate the plant dynamics into "fast" and "slow" time scales. The stability of the system depends on achieving adequate separation in the time scales. However, under large changes in the aircraft dynamics, the feedback gains of the system may not be sufficient to stabilize the system because the inverse dynamic model of the aircraft is not adaptive. Higher gains could be used to offset this, but may excite high frequency unmodeled dynamics in the plant.

It is to be noted that in all of the above control methods some a priori knowledge of the plant is needed. This is introduced in the form of an approximate inverse in the case of Calise et al. [4], Hovakimyan et al. [6], and Pesonen et al. [2]. Ferrari and Stengel [3] directly absorb the linear controller into the neural network during the offline training. In the case of Shin and Kim [7] plant knowledge is introduced as an affine approximation used to compute the pseudocontrol signal in the backstepping process. In Boskovic et al. [8] this takes the form of perfect knowledge of the nonlinearities appearing in linear combination with the unknown parameters. Furthermore, in all these approaches the underlying neural network structure is fixed a priori and does not vary during the adaptive control phase. This may tend to make the size of the network large. One approach to tackle this issue is the variable structure neural networks of Liu et al. [11], where the number of basis functions can be either increased or decreased with time. The algorithm makes a compromise between orthogonality and smoothness when adding or removing a neuron. A different approach is taken in Lu et al. [12] where a neuron is added based on the novelty of the incoming data in the input space.

Recently, we presented preliminary results for developing an online backstepping neural controller for handling actuator failures during the landing maneuver under severe winds [13]. In this, we had considered only the aircraft longitudinal dynamics and the failure was in the elevator (both left and right). Stability results were presented for this controller. In this paper, we extend the above results to the full 6 degrees of freedom nonlinear dynamics (including both longitudinal and lateral dynamics) and show that an online backstepping neural controller can be designed with stable adaptive tuning rules for trajectory tracking control of aircraft executing a landing maneuver under control surface actuator failures (elevator and ailerons). Both single and double actuator stuck-up failures have been considered.

The architecture of the proposed backstepping neural controller is shown in Fig. 1 and is called adaptive backstepping neural Controller (ABNC). It can be seen from Fig. 1 that the scheme has four neural networks in a cascade. It is noted that Johnson and Kannan [14] proposed a cascade structure for adaptive control of helicopters. The number of neural networks required depends on the relative degree of the output (i.e., the number of times the output is differentiated to make the control input appear in the equation). All the networks receive the aircraft state vector as input apart from their individual inputs. Network NN1 is innermost and produces the elevator, aileron, and rudder control signals. The cascade terminates in NN4 whose input is the desired aircraft output we wish to control during the landing phase (here, altitude and cross track deviation). This is the integrator backstepping process and is described in detail in Sec. II. The individual network learns the inverse nonlinear dynamics of each stage. The parameters of each of the neural networks are adapted by using suitable error signals derived from Lyapunov's theory. The stability proof presented here guarantees the ultimate boundedness of all signals in the closed loop system.

This paper is organized as follows. Section II presents the adaptive backstepping neural controller design. The implementation and update rules for the parameters of the cascaded neural networks are presented. The assumptions used to derive the results are discussed and a theorem on the stability of all the signals in the ABNC is stated along with a discussion on the significance of the design gains. The proof of the stability theorem using Lyapunov arguments is given in the Appendix. A description of the aircraft model and the landing task along with the requirements on touchdown dispersions (called "pill box") is given in Sec. III. Performance evaluation of the ABNC for the landing task with and without failures is presented in Sec. IV. The conclusions are summarized in Sec. V.

II. Adaptive Backstepping Neural Controller Design

In this section, we present the design methodology for the adaptive neural network aircraft controller using the backstepping approach. The integrator backstepping design technique was proposed by Kanellakopoulos et al. [15]. Before discussing the controller design, a brief description of the aircraft model is given.

A. Aircraft Equations of Motion

In this paper, we shall follow the formulation of the equations of motion as given in Bugajski and Enns [10]. They are

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \qquad Y = \boldsymbol{l}(\boldsymbol{x}) \tag{1}$$

where the state vector is

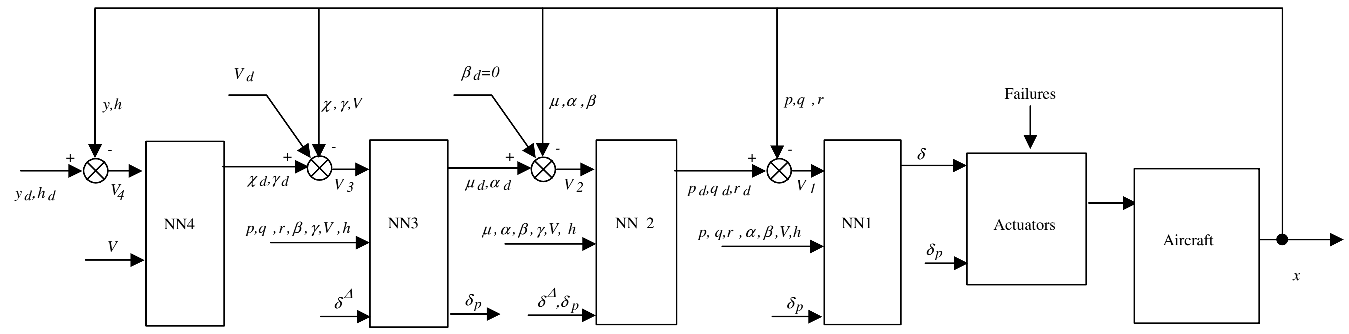


Fig. 1 Adaptive backstepping neural controller architecture.

$$\boldsymbol{x} = \begin{bmatrix} x_1 & \cdots & x_{11} \end{bmatrix}^T = \begin{bmatrix} (p,q,r)^T \\ (\mu,\alpha,\beta)^T \\ (\chi,\gamma,V)^T \\ (y,h)^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}_1 \\ \boldsymbol{X}_2 \\ \boldsymbol{X}_3 \\ \boldsymbol{X}_4 \end{bmatrix} \in R^{11}$$

the control vector is

$$\boldsymbol{u} = \begin{bmatrix} (\delta_a,\delta_e,\delta_r)^T \\ \delta_p \end{bmatrix} = \begin{bmatrix} \delta \\ \delta_p \end{bmatrix} \in R^4$$

the output is $Y = \begin{bmatrix} \beta & y & h & V \end{bmatrix}^4 \in R^4$ and $\boldsymbol{l} \in R^4$, $\boldsymbol{f} \in R^{11}$. The components of the state vector retain their usual meaning [10]. The control vector $\boldsymbol{u}$ consists of the elevator, the aileron, the rudder, and the throttle in that order. The objective is to design a controller to track a given smooth reference trajectory Y_d with bounded errors using state feedback. In shorthand vector notation Eq. (1) can be rewritten as

$$\begin{gathered} \dot{\boldsymbol{X}}_1 = \boldsymbol{F}_1(p,q,r,\alpha,\beta,V,h,\delta,\delta_p) \\ \dot{\boldsymbol{X}}_2 = \boldsymbol{F}_2(p,q,r,\mu,\alpha,\beta,\gamma,V,h,\boldsymbol{u}) \\ \dot{\boldsymbol{X}}_3 = \boldsymbol{F}_3(p,q,r,\mu,\alpha,\beta,\gamma,V,h,\delta,\delta_p) \qquad \dot{\boldsymbol{X}}_4 = \boldsymbol{F}_4(\chi,\gamma,V) \end{gathered} \tag{2}$$

It should be noted that the uncertain terms in Eq. (2) are due in the aerodynamic force and moment coefficients. Further, the throttle δ_p to thrust T mapping appearing in these equations is also assumed to be uncertain. The dependence of the aerodynamic and thrust force and moment coefficients with respect to the elements of the control vector is assumed to be nonaffine.

B. Controller Design Methodology

Equation (2) can be written implicitly in terms of $\boldsymbol{x}$ and $\boldsymbol{u}$ as

$$\begin{aligned} \boldsymbol{h}_1(\boldsymbol{x},\boldsymbol{u},\dot{\boldsymbol{X}}_1) &= \dot{\boldsymbol{X}}_1 - \boldsymbol{F}_1(\boldsymbol{x},\boldsymbol{u}) = 0 \\ \boldsymbol{h}_2(\boldsymbol{x},\boldsymbol{u},\dot{\boldsymbol{X}}_2) &= \dot{\boldsymbol{X}}_2 - \boldsymbol{F}_2(\boldsymbol{x},\boldsymbol{u}) = 0 \\ \boldsymbol{h}_3(\boldsymbol{x},\boldsymbol{u},\dot{\boldsymbol{X}}_3) &= \dot{\boldsymbol{X}}_3 - \boldsymbol{F}_3(\boldsymbol{x},\boldsymbol{u}) = 0 \\ \boldsymbol{h}_4(\boldsymbol{x},\dot{\boldsymbol{X}}_4) &= \dot{\boldsymbol{X}}_4 - \boldsymbol{F}_4(\boldsymbol{x}) = 0 \end{aligned} \tag{3}$$

Assumption 1: The Jacobian matrices

$$\frac{\partial \boldsymbol{F}_1(\boldsymbol{x},u)}{\partial(\delta_a,\delta_e,\delta_r)}$$

$$\frac{\partial \boldsymbol{F}_2(\boldsymbol{x},\dot{x}_3)}{\partial(p,q,r)}$$

$$\frac{\partial \boldsymbol{F}_3(\boldsymbol{x},u)}{\partial(\mu,\alpha,\delta_p)}$$

$$\frac{\partial \boldsymbol{F}_4(\boldsymbol{x},u)}{\partial(\chi,\gamma)}$$

have nonzero determinant for all $(\boldsymbol{x},\boldsymbol{u}) \in U$, where $U \in R^{15}$ is a compact set.

Assumption 2: The desired trajectory and the corresponding states have continuous first derivatives Y_d, $x_i^d \in C^1$ and these derivatives are bounded [3,4,6,7]. Also, the desired trajectory itself is bounded.

1. Nonlinear Dynamic Inversion (NDI)

The functions $\boldsymbol{h}_1,\ldots,\boldsymbol{h}_4$ in Eq. (3) are the implicit form of the equations of motion. Using the implicit function theorem on these equations, there exist functions $\boldsymbol{K}_i$, $i = 1,\ldots,4$ such that

$$\begin{aligned} [\chi,\gamma]^T &= \boldsymbol{K}_4(\dot{\boldsymbol{X}}_4, V) \\ [\mu,\alpha,\delta_p]^T &= \boldsymbol{K}_3(p,q,r,\beta,\dot{\boldsymbol{X}}_3,\gamma,V,h,\delta) \\ [p,q,r]^T &= \boldsymbol{K}_2(\dot{\boldsymbol{X}}_2,\mu,\alpha,\beta,\gamma,V,h,\boldsymbol{u}) \\ [\delta_a,\delta_e,\delta_r]^T &= \boldsymbol{K}_1(p,q,r,\alpha,\beta,V,h,\dot{\boldsymbol{X}}_1,\delta_p) \end{aligned} \tag{4}$$

The above equations represent the exact inverse functions of Eq. (2). The controller consists of evaluating the above multivariable vector functions starting from $\boldsymbol{K}_4$ through to $\boldsymbol{K}_1$ in that order to obtain the various control signals ($\boldsymbol{u}$). The functions have to be evaluated with the desired state derivatives $\dot{\boldsymbol{X}}_i$, $i = 1,\ldots,4$. The desired state vector is partly obtained from the output vector Y and from Eq. (4).

As discussed in [10,16], the NDI controller consists of evaluating the inverse functions K_i by setting the desired state derivatives equal to signals proportional to the state trajectory error. Toward this end, we define the pseudocontrol variables v_i, $i = 1,\ldots,4$.

$$\boldsymbol{V}_1 = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} -\Gamma_1 e_1 \\ -\Gamma_2 e_2 \\ -\Gamma_3 e_3 \end{bmatrix}, \qquad \boldsymbol{V}_2 = \begin{bmatrix} v_4 \\ v_5 \\ v_6 \end{bmatrix} = \begin{bmatrix} -\Gamma_4 e_4 \\ -\Gamma_5 e_5 \\ -\Gamma_6 e_6 \end{bmatrix}$$
$$\boldsymbol{V}_3 = \begin{bmatrix} v_7 \\ v_8 \\ v_9 \end{bmatrix} = \begin{bmatrix} -\Gamma_7 e_7 \\ -\Gamma_8 e_8 \\ -\Gamma_9 e_9 \end{bmatrix}, \qquad \boldsymbol{V}_4 = \begin{bmatrix} v_{10} \\ v_{11} \end{bmatrix} = \begin{bmatrix} -\Gamma_{10} e_{10} \\ -\Gamma_{11} e_{11} \end{bmatrix} \tag{5}$$

where Γ_i are the gains to be designed and e_i are the components of the error vector

$$e_i = x_i - x_i^d, \qquad i = 1,\ldots,11 \tag{6}$$

This ensures that the states converge to the desired values and the controller is robust to external disturbances. We denote this controller by

$$\begin{aligned} [\chi^*,\gamma^*]^T &= \boldsymbol{K}_4(\boldsymbol{V}_4, V) \\ [\mu^*,\alpha^*,\delta_p^*]^T &= \boldsymbol{K}_3(p,q,r,\beta,\boldsymbol{V}_3,\gamma,V,h,\delta) \\ [p^*,q^*,r^*]^T &= \boldsymbol{K}_2(\boldsymbol{V}_2,\mu,\alpha,\beta,\gamma,V,h,\boldsymbol{u}) \\ [\delta_a^*,\delta_e^*,\delta_r^*]^T &= \boldsymbol{K}_1(p,q,r,\alpha,\beta,V,h,\boldsymbol{V}_1,\delta_p) \end{aligned} \tag{7}$$

The asterisks on the left-hand sides are used to indicate that these controls are obtained by evaluating the exact dynamic inverse functions.

The robustness of the NDI controller described above depends on the gains Γ_i and an exact knowledge of the system inverse dynamics in Eq. (4). In problems where the plant dynamics is uncertain, large gains may be required to maintain stability of the NDI controller under rapid changes in the plant (e.g., partial loss of control effectiveness). The robustness of the controller can be enhanced if an online learning process is introduced. In this paper we show that neural networks can be used with suitable tuning rules to obtain a controller robust to large changes in the plant. It is also shown that a neural network-based fault-tolerance controller can be designed without the need for a fault detection and identification (FDI) scheme. The stability of the neural network controller is demonstrated using Lyapunov's theory. The first step consists of replacing functions in Eq. (7) with neural network function approximators.

2. RBF Approximation for the Inverse Functions

In this section, the procedure for approximating the desired inverse functions $\boldsymbol{K}_i$ using radial basis neural networks (RBFN) is described. If the RBFN input vector is $\boldsymbol{x}$, then its output y is

$$y = W_0 + \sum_{i=1}^{m} W_i \cdot \phi_i(\boldsymbol{x}) = W_0 + W^T \cdot \Phi(\boldsymbol{x}) \tag{8}$$

In the above equations m is the number of neurons in the hidden layer, W_0 is the network bias, and W_i are the neuron weights (W is the vector of weights and Φ the vector consisting of basis functions). The radial basis neural network has the Gaussian basis functions $\phi_i = \exp(-\|\boldsymbol{x} - m_i\|^2/\sigma_i^2)$. The quantities μ_i and σ_i represent the ith neuron center vector and its width, respectively.

Thus, the inverse functions in Eq. (7) are replaced by neural network stages to arrive at adaptive backstepping neural controller architecture for uncertain systems (Fig. 1). The stages are implemented as

$$\begin{aligned}
&[\chi^d, \gamma^d]^T = \boldsymbol{NN}_4(\boldsymbol{V}_4, V) = \hat{\boldsymbol{W}}_{04} + \hat{\boldsymbol{W}}_4^T \hat{\Phi}_4(\boldsymbol{V}_4, V)\\
&[\mu^d, \alpha^d, \delta_p^d]^T = \boldsymbol{NN}_3(p, q, r, \beta, \boldsymbol{V}_3, \gamma, V, h, \delta) = \hat{\boldsymbol{W}}_{03}\\
&\quad + \hat{\boldsymbol{W}}_3^T \hat{\Phi}_3(p, q, r, \beta, \boldsymbol{V}_3, \gamma, V, h, \delta^\Delta)\\
&[p^d, q^d, r^d]^T = \boldsymbol{NN}_2(\boldsymbol{V}_2, \mu, \alpha, \beta, \gamma, V, h, u) = \hat{\boldsymbol{W}}_{02}\\
&\quad + \hat{\boldsymbol{W}}_2^T \hat{\Phi}_2(\boldsymbol{V}_2, \mu, \alpha, \beta, \gamma, V, h, \delta^\Delta, \delta_p)\\
&[\delta_a^d, \delta_e^d, \delta_r^d]^T = \boldsymbol{NN}_1(p, q, r, \alpha, \beta, V, h, \boldsymbol{V}_1, \delta_p) = \hat{\boldsymbol{W}}_{01}\\
&\quad + \hat{\boldsymbol{W}}_1^T \hat{\Phi}_1(p, q, r, \alpha, \beta, V, h, \boldsymbol{V}_1, \delta_p)
\end{aligned} \tag{9}$$

where $\hat{\boldsymbol{W}}_{0i}$ are estimates of the network bias terms, $\hat{\boldsymbol{W}}_i$ are the estimates of optimal RBF weight vectors, and $\hat{\Phi}_i$ are estimates of the optimal radial basis function vectors. The quantities χ^d, γ^d, μ^d, α^d, δ_p^d, p^d, q^d, r^d, δ_a^d, δ_e^d, and δ_r^d are time varying neural network outputs representing the desired intermediate control signals in the approximating neural network cascade shown in Fig. 1. It is noted that the neural network biases and weights in Eq. (9) represent multivariable vector functions. Thus, each bias and weight in Eq. (9) is a vector and a matrix, respectively. The update rules for these networks must be designed in such a way that the system is driven to follow the desired trajectory. The functions $\boldsymbol{NN}_i$ represent the neural networks arranged in a cascade. The sequence of evaluation starts on the left-hand side from $\boldsymbol{NN}_4$ and ends in $\boldsymbol{NN}_1$. The latter produces the aerodynamic control vector δ that drives the aircraft actuators. It is seen that the networks $\boldsymbol{NN}_3$ and $\boldsymbol{NN}_2$ require the aerodynamic control deflections as input. This gives rise to an algebraic loop as these controls are computed downstream in network $\boldsymbol{NN}_1$. Aerodynamic control inputs are delayed by one sample and used in the neural networks (denoted by δ^Δ) for solving this problem. It is assumed that if the sampling rate is sufficiently high, the delay will not affect the network approximation process.

In this paper the radial basis function neural networks (RBFNN) are used for approximation. Any other neural network can also be used as the universal approximator. This will become apparent from the theorem presented in the next section.

The optimal RBFN represents the choice of network parameters, which gives the lowest upper bound of the approximation error over a suitable compact set. The optimal RBFN is related to the ideal implicit inverse functions defined in Eq. (7):

$$\begin{aligned}
&[\chi^*, \gamma^*]^T = \boldsymbol{K}_4(\boldsymbol{V}_4, V) = \boldsymbol{W}_{04} + \boldsymbol{W}_4^T \Phi_4(\boldsymbol{V}_4, V) + \boldsymbol{N}_{4,nn}\\
&\quad = [\chi_{nn}, \gamma_{nn}]^T + \boldsymbol{N}_{4,nn}\\
&[\mu^*, \alpha^*, \delta_p^*]^T = \boldsymbol{K}_3(p, q, r, \beta, \boldsymbol{V}_3, \gamma, V, h, \delta)\\
&\quad = \boldsymbol{W}_{03} + \boldsymbol{W}_3^T \Phi_3(p, q, r, \beta, \boldsymbol{V}_3, \gamma, V, h, \delta) + \boldsymbol{N}_{3,nn}\\
&\quad = [\mu_{nn}, \alpha_{nn}, \delta_{pnn}]^T + \boldsymbol{N}_{3,nn}\\
&[p^*, q^*, r^*]^T = \boldsymbol{K}_2(\boldsymbol{V}_2, \mu, \alpha, \beta, \gamma, V, h, u)\\
&\quad = \boldsymbol{W}_{02} + \boldsymbol{W}_2^T \Phi_2(\boldsymbol{V}_2, \mu, \alpha, \beta, \gamma, V, h, u) + \boldsymbol{N}_{2,nn}\\
&\quad = [p_{nn}, q_{nn}, r_{nn}]^T + \boldsymbol{N}_{2,nn}
\end{aligned} \tag{10}$$

$$\begin{aligned}
&[\delta_a^*, \delta_e^*, \delta_r^*]^T = \boldsymbol{K}_1(p, q, r, \alpha, \beta, V, h, \boldsymbol{V}_1, \delta_p)\\
&\quad = \boldsymbol{W}_{01} + \boldsymbol{W}_1^T \Phi_1(p, q, r, \alpha, \beta, V, h, \boldsymbol{V}_1, \delta_p) + \boldsymbol{N}_{1,nn}\\
&\quad = [\delta_{ann}, \delta_{enn}, \delta_{rnn}]^T + \boldsymbol{N}_{1,nn}
\end{aligned}$$

where $\boldsymbol{N}_{i,nn}$, $i = \{1, 2, 3, 4\}$ are bounded functions, $\boldsymbol{W}_i$ are the optimal RBF weight vectors, $\boldsymbol{W}_{0i}$ are the bias terms, and Φ_i the radial basis function vectors. In Eq. (10) the quantities χ_{nn}, γ_{nn}, μ_{nn}, α_{nn}, δ_{pnn}, p_{nn}, q_{nn}, r_{nn}, δ_{ann}, δ_{enn}, and δ_{rnn} represent the desired intermediate control signals, which will result when the approximating neural networks in the cascade shown in Fig. 1 are replaced by the corresponding optimal RBFN networks. Functions $\boldsymbol{N}_{i,nn}$, $i = \{1, 2, 3, 4\}$ are defined as

$$\begin{aligned}
\begin{bmatrix} \eta_{1,nn} & \cdots & \eta_{11,nn} \end{bmatrix}^T &= \begin{bmatrix} (\delta_a^* - \delta_{ann}, \delta_e^* - \delta_{enn}, \delta_r^* - \delta_{rnn})^T \\ (p^* - p_{nn}, q^* - q_{nn}, r^* - r_{nn})^T \\ (\mu^* - \mu_{nn}, \alpha^* - \alpha_{nn}, \delta_p^* - \delta_{pnn}^*)^T \\ (\chi^* - \chi_{nn}, \gamma^* - \gamma_{nn})^T \end{bmatrix}\\
&= \begin{bmatrix} \boldsymbol{N}_{1,nn} \\ \boldsymbol{N}_{2,nn} \\ \boldsymbol{N}_{3,nn} \\ \boldsymbol{N}_{4,nn} \end{bmatrix}
\end{aligned} \tag{11}$$

From the universal approximation property of the neural networks [16], one can conclude that the functions $\boldsymbol{N}_{i,nn}$ can be made arbitrarily small by increasing the maximum number of neurons ($N_{\max}$) for each neural network $\boldsymbol{NN}_1, \ldots, \boldsymbol{NN}_4$ and proper selection of the corresponding network parameters. Because of the finite approximation error introduced by the RBFN, the system states will only approach the desired states in an ultimate bounded sense. The following shorthand notation is introduced:

$$\begin{aligned}
\begin{bmatrix} \sigma_1 & \cdots & \sigma_{11} \end{bmatrix}^T &= \begin{bmatrix} \left(\delta_a^d, \delta_e^d, \delta_r^d\right)^T \\ (p^d, q^d, r^d)^T \\ (\mu^d, \alpha^d, \delta_p)^T \\ (\chi^d, \gamma^d)^T \end{bmatrix}\\
\begin{bmatrix} \sigma_{1,nn} & \cdots & \sigma_{11,nn} \end{bmatrix}^T &= \begin{bmatrix} (\delta_{ann}, \delta_{enn}, \delta_{rnn})^T \\ (p_{nn}, q_{nn}, r_{nn})^T \\ (\mu_{nn}, \alpha_{nn}, \delta_{pnn})^T \\ (\chi_{nn}, \gamma_{nn})^T \end{bmatrix}\\
\Delta\sigma_i = \sigma_i - \sigma_{i,nn}, &\qquad i = 1, \ldots, 11
\end{aligned} \tag{12}$$

Assumption 3: The desired trajectories can be achieved using a control deflection $\boldsymbol{u}$ and states $\boldsymbol{x}$ within the set $(\boldsymbol{x}, \boldsymbol{u}) \in U$.

The above assumption implies that the ideal inverse function outputs χ^*, γ^*, μ^*, α^*, δ_p^*, p^*, q^*, r^*, δ_a^*, δ_e^*, and δ_r^* are bounded and because the functions $\boldsymbol{N}_{i,nn}$ are also bounded (from the universal approximation property) for a suitable neural network size ($N_{\max}$), this implies that the optimal RBFN outputs are bounded. From Assumption 2, it follows that the time derivatives of the ideal implicit inverse functions χ^*, γ^*, μ^*, α^*, δ_p^*, p^*, q^*, r^*, δ_a^*, δ_e^*, and δ_r^* are bounded. Because the approximating optimal RBFN network is also a smooth function of its arguments, the time derivatives of the optimal RBFN outputs ($\dot{\chi}_{nn}, \dot{\gamma}_{nn}, \dot{\mu}_{nn}, \dot{\alpha}_{nn}, \dot{\delta}_{pnn}, \dot{p}_{nn}, \dot{q}_{nn}, \dot{r}_{nn}, \dot{\delta}_{ann}, \dot{\delta}_{enn}, \dot{\delta}_{rnn}$) are bounded. Thus, using the shorthand notation in Eq. (12)

$$\begin{gathered}
|\dot{x}_i^d| \le x_{i,md}, \qquad |\sigma_{i,nn}| \le \sigma_{i,m}, \qquad |\dot{\sigma}_{i,nn}| \le \sigma_{i,md}\\
i = \{1, \ldots, 11\}
\end{gathered} \tag{13}$$

For a given $N_{\max}$, tuning rules for the individual RBFN in the ABNC scheme described below will ensure that the approximate network parameters approach their optimal values. Using the shorthand notation in Eq. (12), the network update rules are given in terms of the rate of change of the network output:

$$\begin{bmatrix}\dot{\sigma}_1\\ \dot{\sigma}_2\\ \dot{\sigma}_3\end{bmatrix} = -\begin{bmatrix}\Lambda_{11} & \Lambda_{12} & \Lambda_{13}\\ \Lambda_{21} & \Lambda_{22} & \Lambda_{23}\\ \Lambda_{31} & \Lambda_{32} & \Lambda_{33}\end{bmatrix}\cdot\begin{bmatrix}e_1\\ e_2\\ e_3\end{bmatrix} - \begin{bmatrix}\Omega_1\cdot\sigma_1\\ \Omega_2\cdot\sigma_2\\ \Omega_3\cdot\sigma_3\end{bmatrix}$$
$$\begin{bmatrix}\dot{\sigma}_4\\ \dot{\sigma}_5\\ \dot{\sigma}_6\end{bmatrix} = -\begin{bmatrix}\Lambda_{44} & \Lambda_{45} & \Lambda_{46}\\ \Lambda_{54} & \Lambda_{55} & \Lambda_{56}\\ \Lambda_{64} & \Lambda_{65} & \Lambda_{66}\end{bmatrix}\cdot\begin{bmatrix}e_4\\ e_5\\ e_6\end{bmatrix} - \begin{bmatrix}\Omega_4\cdot\sigma_4\\ \Omega_5\cdot\sigma_5\\ \Omega_6\cdot\sigma_6\end{bmatrix}$$
$$\begin{bmatrix}\dot{\sigma}_7\\ \dot{\sigma}_8\\ \dot{\sigma}_9\end{bmatrix} = -\begin{bmatrix}\Lambda_{77} & \Lambda_{78} & \Lambda_{79}\\ \Lambda_{87} & \Lambda_{88} & \Lambda_{89}\\ \Lambda_{97} & \Lambda_{98} & \Lambda_{99}\end{bmatrix}\cdot\begin{bmatrix}e_7\\ e_8\\ e_9\end{bmatrix} - \begin{bmatrix}\Omega_7\cdot\sigma_7\\ \Omega_8\cdot\sigma_8\\ \Omega_9\cdot\sigma_9\end{bmatrix}$$
$$\begin{bmatrix}\dot{\sigma}_{10}\\ \dot{\sigma}_{11}\end{bmatrix} = -\begin{bmatrix}\Lambda_{10,10} & \Lambda_{10,11}\\ \Lambda_{11,10} & \Lambda_{11,11}\end{bmatrix}\cdot\begin{bmatrix}e_{10}\\ e_{11}\end{bmatrix} - \begin{bmatrix}\Omega_{10}\cdot\sigma_{10}\\ \Omega_{11}\cdot\sigma_{11}\end{bmatrix} \tag{14}$$

Note that the network update rule is not specific to the radial basis function type of network. We shall discuss the actual implementation of these rules in terms of the network parameters (biases, weights, centers, and widths) in the next section. The ABNC scheme has been shown schematically in Fig. 1.

Assuming that all the states are available for feedback, the main result of this paper can now be stated as follows:

Theorem 1: For the system (1) with Assumptions 1–3 satisfied, let the control scheme be implemented as Fig. 1, network update rule given in Eq. (14), and there exist gains Γ_i, Λ_{ij}, and Ω_i all positive satisfying

$$B_i > 1, \qquad F_i > \frac{D_i^2}{4}, \qquad i = \{1,\ldots,11\} \tag{15}$$

then the trajectory $(\boldsymbol{x},\boldsymbol{u})$ of the system remains in the compact set U and the tracking error converges to a neighborhood of the origin which depends on $\sigma_{i,md}$, $\sigma_{i,m}$, η_i, and $i = \{1,\ldots,11\}$. The quantities B_i, F_i, and D_i depend only on the gains and certain constants (constructed in the proof).

Proof: See the Appendix.

C. Tuning Rules for RBFN

In the previous section, ABNC architecture was developed with stable tuning rules as in Eq. (14) in terms of the estimated outputs σ_i, $i = \{1,\ldots,11\}$. In this section, the implementation of these tuning rules for a RBFN approximating the inverse functions are presented. In the RBFN, the centers, width, and the weights are adjusted adaptively. Further, the number of neurons is allowed to grow with a cap on the maximum number and no pruning of neurons is done. Equation (9) differentiated with respect to time gives

$$\dot{\sigma}_i = \nabla\hat{g}_i\cdot\dot{w}_i, \qquad i = \{1,\ldots,11\} \tag{16}$$

where

$$\nabla\hat{g}_i = [I, \hat{\phi}_{i1}\cdot I, 2\hat{\phi}_{i1}/\hat{\sigma}_{i1}^2\cdot\hat{W}_{i1}\cdot(x-\hat{\mu}_{i1})^T, 2\hat{\phi}_{i1}/\hat{\sigma}_{i1}^3\cdot\hat{W}_{i1}\cdot\|x - \hat{\mu}_{i1}\|^2$$
$$\ldots$$
$$\hat{\phi}_{ik}\cdot I, 2\hat{\phi}_{ik}/\hat{\sigma}_{ik}^2\cdot\hat{W}_{ik}\cdot(x-\hat{\mu}_{ik})^T, 2\hat{\phi}_{ik}/\hat{\sigma}_{ik}^3\cdot\hat{W}_{ik}\cdot\|x-\hat{\mu}_{ik}\|^2]^T \tag{17}$$

is the gradient vector of the ith network with the second subscript running over the number of neurons $(1,\ldots,m)$. The parameter vector $w_i = [W_{0i1}, \hat{W}_{i1}, \hat{\mu}_{i1}, \hat{\sigma}_{i1}, \ldots, \hat{W}_{ik}, \hat{\mu}_{ik}, \hat{\sigma}_{ik}]^T$ consists of the RBFN biases, weights, and center widths. Using the matrix pseudoinverse of the gradient vector, one may obtain the rate of change of the RBFN parameters from the rate of change of the adaptive controls σ_i, $i = \{1,\ldots,11\}$ as

$$\dot{w}_i = [\nabla\hat{g}_i]^+\cdot\dot{\sigma}_i, \qquad i = \{1,\ldots,11\} \tag{18}$$

where $[\,]^+$ denotes the pseudoinverse.

Generally, when using a RBFN for control only the weights are updated adaptively and the number of neurons, the centers, and the widths are fixed “a priori.” Recently, Lu et al. [12] developed a RBF called MRAN in which the neurons are allowed to adaptively grow as well as to be pruned based on certain criteria. In this paper, pruning of neurons is not used but growing is used with an upper limit. When the number of neurons reaches this limit, no more neurons are added but the updates of the parameters are carried out.

All signals in the controller are ultimately bounded and approach a compact set which depends on the network approximation error. To prevent loss of stability when the trajectory enters the compact set a standard solution is to apply a dead zone [17]. As the network learns, the approximation error reduces. Hence, the dead zone is larger initially ($\varepsilon_{\max}$) and subsequently reduced to a lower value ($\varepsilon_{\min}$). The instantaneous value of the dead zone is given by $\varepsilon = \max(\varepsilon_{\max}\cdot r^n, \varepsilon_{\min})$, where n is the iteration count and the rate of change $r \in (0,1)$. The control law is implemented in discrete form. The computations to be performed by the controller are given below:

```
**Begin**
  Compute the RBF network output
  Compute the dead zone threshold $\varepsilon$
  *if* $\|e\| > \varepsilon$ *then*
    Update the network parameters using Eqs. (14) and (18).
    *if* $m < N_{\max}$ *then*
      Add a neuron at the current location with fixed overlap ($\kappa_p$) to
    the nearest neuron in input space (see [12] for details).
    *End*
  *End*
**End**
```

It is important to choose the inputs to each of the cascaded neural networks carefully. Thus, for example, for the $\hat{g}_1$ network instead of using $\{v_1, \alpha, q\}$ as inputs, we give

$$\left\{\frac{I_y}{\bar{q}S\bar{c}C_{m\delta e}}\cdot v_1, -\frac{C_{m\alpha}}{C_{m\delta e}}\cdot\alpha, -\frac{C_{mq}\bar{c}}{C_{m\delta e}2V}\cdot q\right\}$$

where $C_{m\delta e}$, $C_{m\alpha}$, and C_{mq} are the usual aerodynamic derivatives. This amounts to using the terms of the linear inversion controller as inputs to the neural network. These derivatives are to be evaluated at a nominal starting flight condition and remain fixed during simulation. This is a natural scaling rule for the aircraft controller.

D. Choice of Design Gains

In this section, the significance of the two-component update rule derived for RBFNs is highlighted. For the first neural network, the gains to be tuned are Γ_i, Ω_i, Λ_{ij}, $i = 1, 2, 3$, $j = 1, 2, 3$. The control loop between networks NN1–NN2 is opened and three step inputs separately, namely, (p^d, q^d, r^d) are injected. The gains are designed based on the time response to these inputs. Diagonal entries of the gain matrix multiplying the trajectory error (Λ_{ii}) primarily affect the speed of response, while the gain multiplying the output term (Ω_i, $i = 1, 2, 3$) affects the stability of the closed loop. The gain Γ_i, $i = 1, 2, 3$ is the backstepping correction as discussed in Sec. II.B.

To analyze the combined effect of NN1 and NN2, the control loop is broken in Fig. 1 between the networks NN2, NN3 and a step demand in angle of attack is injected to the network NN2. If the outer loop speed of response and overall stability needs to be improved, the inner loop must be of sufficiently high bandwidth. Tuning of the other loops can proceed on similar lines. A detailed procedure to select the gains based on the time response is given in [18] and the final gain values are given in Tables 2 and 3.

Remarks: The ABNC design proposed in this paper delinks the requirement of stability from the type of neural network. The update rule is specified in terms of the output. This allows for the use of any function approximation scheme in place of the neural network, provided such a scheme has the universal approximation property. Another advantage of the separation of stability from the function approximation is that the neural network growing and pruning can be optimized without bringing in any requirements from the stability point of view. This allows us to build up a neural network by online learning starting from zero neurons. We have used the matrix pseudoinverse for obtaining the network parameter rate in Eq. (18). It is well known that the matrix pseudoinverse generates the minimum 2-norm solution to the overdetermined system of Eq. (16). This ensures that the smallest control deflections consistent with the desired state derivatives will be generated by each of the neural networks NN1–NN4. If a particular backstep does not involve any uncertainty, the exact analytical inverse may be used [e.g., $\gamma_d = \sin^{-1}(\dot{h}_d/V)c$]. The design method applies to systems that may have unstable zero dynamics or may be a nonminimum phase. If the network adaptation rate is sufficiently fast, the neural network can learn to provide control to stabilize an unstable aircraft and also tolerate large changes in dynamics like hardover single elevator failure. The upper limit to the adaptation rate is determined by the actuator bandwidth and the sampling rate of the digital implementation. The lower limit can be chosen for a desired closed loop performance.

E. Application of ABNC Controller to a Linear System

The properties of the ABNC controller are illustrated by applying it to linear short-period dynamics of the aircraft used in this paper. The aircraft dynamics for short period is of the order of 2. The linear model can be written as

$$\begin{bmatrix}\dot{\alpha}\\ \dot{q}\end{bmatrix} = \begin{bmatrix}-0.6485 & 0.8883\\ 0.8122 & -0.6491\end{bmatrix}\begin{bmatrix}\alpha\\ q\end{bmatrix} + \begin{bmatrix}-0.0470 & -0.0470\\ -1.7134 & -1.7134\end{bmatrix}\begin{bmatrix}\delta_{\mathrm{el}}\\ \delta_{\mathrm{er}}\end{bmatrix}$$

For this simplified example, there are two states (angle of attack and pitch rate in units of radians and rad/s, respectively), two control inputs (left and right elevator in radians), and the desired output is the angle of attack. This aircraft model is described in detail in the next section. It is unstable in pitch. The neural network controller is shown in Fig. 2. The controller tracks the reference angle of attack. It is noted that the first network NN1 in this figure represents the inverse dynamic model of the pitch rate equation, while the second network NN2 corresponds to the inversion of the angle of attack equation.

A simulation of the controller is performed with a series of periodic bidirectional angle of attack command pulses (α_d) as input for a total time period of 500 s. The pulses are passed through a first order filter with cutoff frequency 2.5 to generate a smooth command signal. At 300 s, the left elevator control surface actuator is failed and the control surface is held at a hardover position of –10 deg. The actual and desired angle of attack signals are shown in Fig. 3a. The left elevator failure introduces a bias in the aircraft signal after 300 s. This is because the ABNC by itself does not provide for an integral effect to ensure steady-state tracking. Examination of the update rule shows that it involves only a signal proportional to the trajectory error (i.e., proportional feedback). It is possible to improve the response by introducing an integral error term. This is accomplished in the full ABNC controller used for the trajectory following described in the next section. The RBFNN neural network update rule is enforced at each instant of time. This rule ensures stability even when there are no neurons because the bias of the neural network is free to vary. This is confirmed by an examination of the neural network parameters for NN1 in Fig. 3b. It is seen that initially the bias of the network changes rapidly, while the other parameters show much less variation. Near 150 s, the repeated periodic input reinforces the learning and the neural network parameters appear to be converging. Simultaneously, the bias parameter variation reduces and it appears to converge to a new value. At 300 s when the left elevator fails, there is a sharp change in some of the parameters.

At 50 Hz, the neural network parameters of both NN1 and NN2 are updated. In Fig. 3c, we find the plot of the multi-input/single-output neural network function NN2 corresponding to neural network parameters at the time instants of 0, 150, 250, and 500 s. This function has v_2, α, and δ_e^{Δ} as input arguments and produces q_c as the output. In the figure, we show the dependence against v_2, thereby collapsing the dependence with respect to the other two parameters on the same plot. It is seen that the primary dependence is on v_2 because the scatter in the plot with respect to the other parameters is not significant. At 0 s the NN2 is essentially zero. At 150 s the network bias has changed from zero as required by the update rule (15). Some scatter is seen, representing the variation with respect to the input arguments α, δ_e^{Δ} around zero value of v_2. As discussed before, the network parameters at 250 s appear to converge to values corresponding to the no elevator failure condition. A plot of the NN2 function at this time shows that it passes close to the origin. Near the origin, it appears to approximate a linear function. The 500 s NN2 parameter values correspond to the left elevator failed case. The approximation to a linear function is more prominent in the plot of the NN2 function for 500 s. Assumption 1 requires that the derivative of q_c with respect to v_2 does not change sign. In this aircraft short-period example, NN2 appears to approximate this slope.

Therefore, one may conclude that the neural network learning takes place over a larger interval of time and is reinforced by repeated application of the similar control inputs. The stability of the ABNC controller is guaranteed by the update rule and thus does not depend on the network-learning rate. A proper choice of network growth criteria will result in optimal network learning.

F. Extension to Systems with Multiple Redundant Controls

Theorem 1 has been proved for a system with essentially three aerodynamic controls (elevator, aileron, and rudder) and one throttle control. Henceforth, we refer to this as the similar surface redundant control (SSRC) implementation. The aircraft model used has redundancies in the form of left elevator–right elevator and left aileron–right aileron combinations. If the redundancy is limited to the use of the same surface to overcome failure, for example, left elevator to be used for right elevator failure and vice versa, then the results of Theorem 1 are applied as follows. First the gains are designed for the left elevator and right aileron control surfaces. The left elevator control signal computed by the ABNC is given identically to the right elevator actuator because the elevators have identical effect on the flight dynamics. In the case of the single aileron control signal computed by the ABNC, it is given directly to the right aileron actuator and with a sign inversion to the left aileron actuator. This is because left and right ailerons have opposite effects on the lateral-directional dynamics. This strategy assures us that the ABNC will achieve fault tolerance irrespective of whether there is a single surface failure (left or right elevator or left or right aileron) or multiple dissimilar control surface failure (left elevator and left aileron or left elevator and right aileron or right elevator and left aileron or right elevator and right aileron). This method will not work for the case of two similar control surface failures (left and right ailerons).

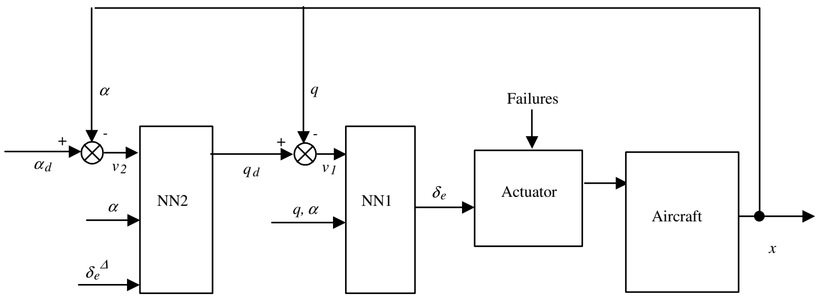
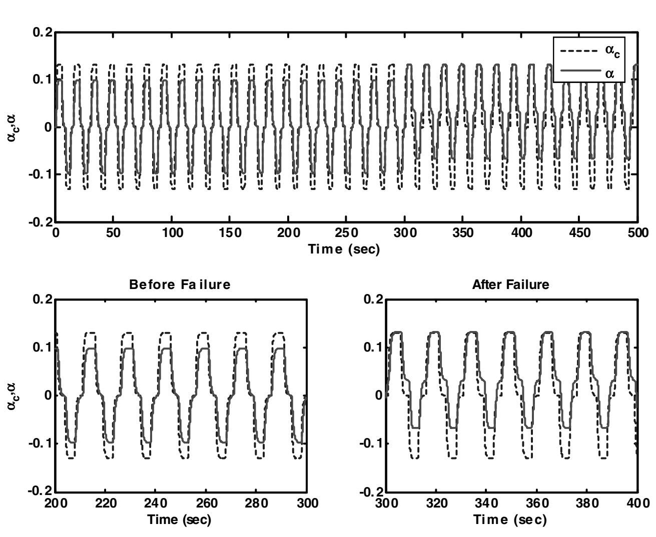
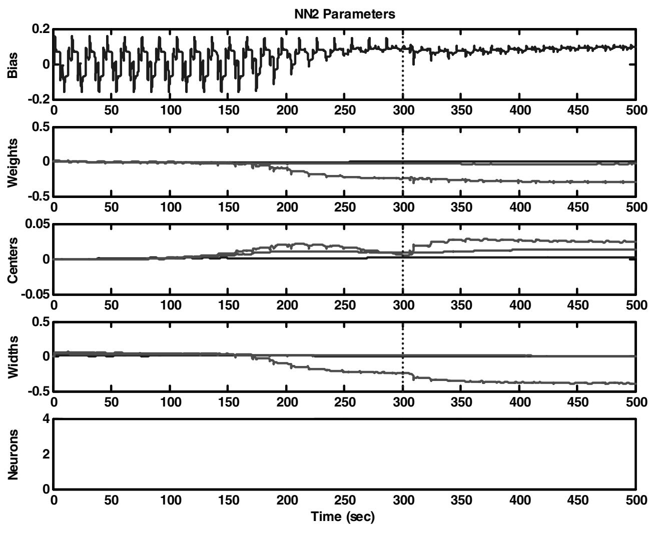


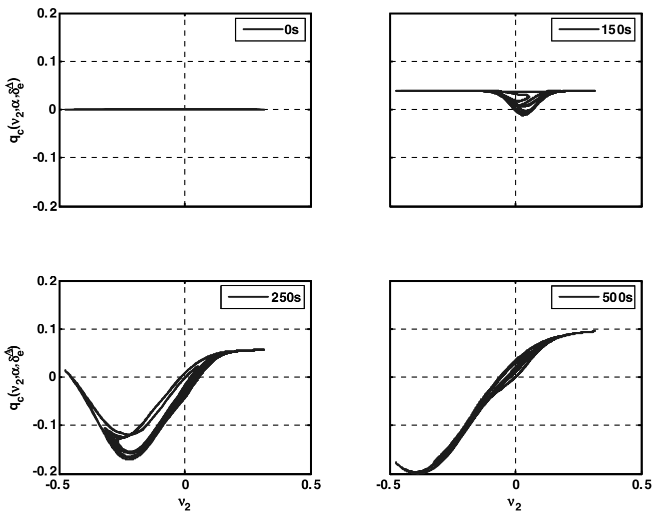
Fig. 2 ABNC scheme for control of linear short-period dynamics.



a)



b)



c)

Fig. 3 a) Command and aircraft response of ABNC (linear short-period dynamics). Left elevator hardover failure to −10 deg at 300 s. b) NN2 network parameters of ABNC (linear short-period dynamics). Left elevator hardover failure to −10 deg at 300 s. c) NN2 network function (linear short-period dynamics). Left elevator hardover failure to −10 deg at 300 s.

In this aircraft, the elevators can be used in differential mode (i.e., similar to the ailerons) to achieve control in the lateral axis. This means that it is possible to handle two aileron failures using the healthy elevators. Theorem 1 can be extended to apply to this case. We sketch the modifications in the proof here. We first note that if all the redundant controls are considered separately, the control vector $\boldsymbol{u}$ will expand to contain five aerodynamic controls and one throttle control. The function $\boldsymbol{F}_1$ in Eq. (2) will now be a dependent on these controls. A unique inverse function $\boldsymbol{K}_1$ in Eq. (4) will not exist, as there will be multiple control inputs satisfying the given value of $\dot{\boldsymbol{X}}_1$. However, this situation can be handled by requiring that in addition to achieving the desired value of $\dot{\boldsymbol{X}}_1$, the inverse function $\boldsymbol{K}_1$ must result in a set of control deflections that have minimum 2-norm. As discussed before, the matrix pseudoinverse can be used for this purpose. Our neural network implementation described previously implements this in the update of the network parameters (18). The remaining development of the proof can then be handled as discussed in the Appendix.

The first set of three equations in the update rule (14) will be suitably modified to generate the five derivatives from three errors. Accordingly, gains Λ_{ij}, $i = 1, \ldots, 5, 2, 3$ and Ω_i, $i = 1, \ldots, 5$ need to be designed for network NN1. In the remainder of this paper we shall refer to the implementation with five aerodynamic control surfaces as the multiple surface redundant control (MSRC).

III. Aircraft Landing Task Under Elevator Failures

The aircraft model used in this study is that of a high performance fighter aircraft. Details of the model can be found in [19]. The longitudinal force and pitching moment coefficients for our aircraft model are

$$\begin{aligned}
C_X &= C_X(\alpha,\delta_e) + \Delta C_{X,\mathrm{LEF}}\left(1-\frac{\delta_{\mathrm{LEF}}}{25}\right)\\
&\quad+\frac{q\bar{c}}{2V}\left[C_{Xq}(\alpha)+\Delta C_{Xq,\mathrm{LEF}}(\alpha)\left(1-\frac{\delta_{\mathrm{LEF}}}{25}\right)\right]\\
C_Z &= C_Z(\alpha,\delta_e) + \Delta C_{Z,\mathrm{LEF}}\left(1-\frac{\delta_{\mathrm{LEF}}}{25}\right)\\
&\quad+\frac{q\bar{c}}{2V}\left[C_{Zq}(\alpha)+\Delta C_{Zq,\mathrm{LEF}}(\alpha)\left(1-\frac{\delta_{\mathrm{LEF}}}{25}\right)\right]\\
C_L &= C_X\sin\alpha + C_Z\cos\alpha\\
C_m &= C_m(\alpha,\delta_e) + \Delta C_{m,\mathrm{LEF}}\left(1-\frac{\delta_{\mathrm{LEF}}}{25}\right)\\
&\quad+\frac{q\bar{c}}{2V}\left[C_{mq}(\alpha)+\Delta C_{mq,\mathrm{LEF}}(\alpha)\left(1-\frac{\delta_{\mathrm{LEF}}}{25}\right)\right]
\end{aligned} \tag{19}$$

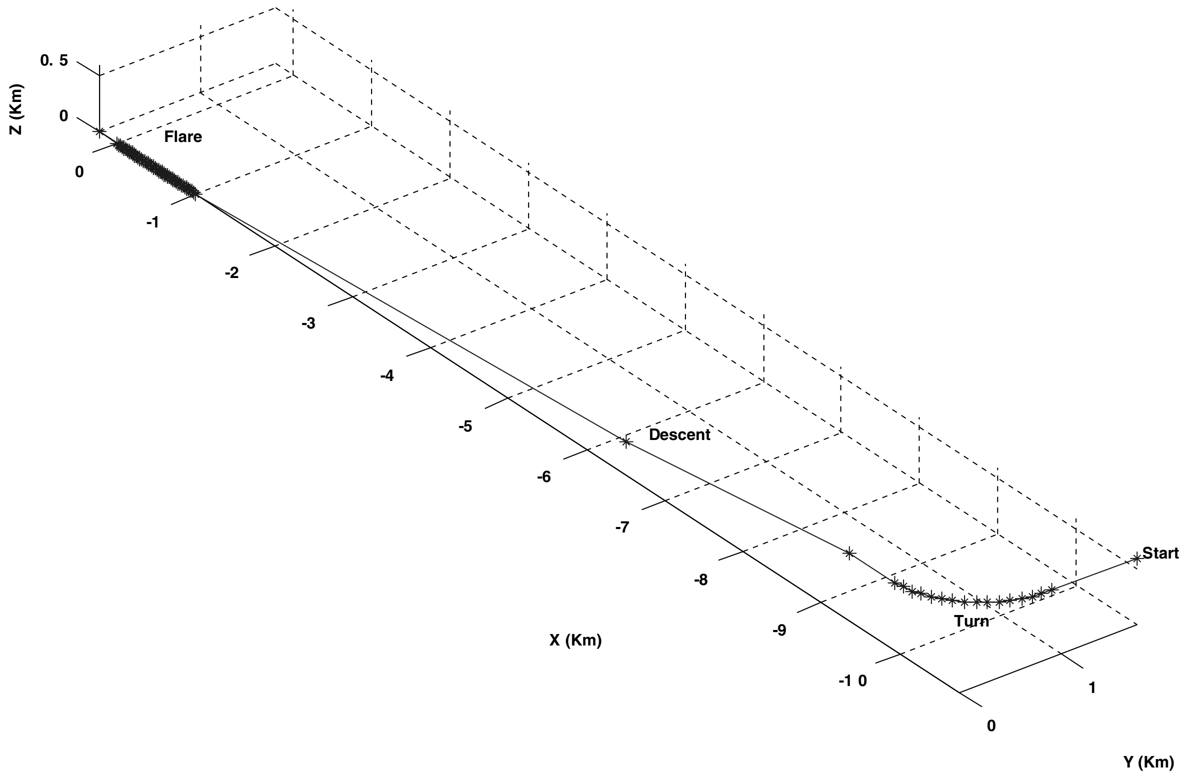


Fig. 4 Landing trajectory.

It is seen that the aerodynamic terms are nonaffine with respect to the elevator control surface deflection. The leading edge flap (LEF) deflection is scheduled with angle of attack, dynamic, and static pressure as $2.4751\alpha - 9.05\bar{q}/P_s + 1.45$. For the purposes of this study, the elevator control surface aerodynamic data have been split into two parts corresponding to left and right surfaces using CFD computations. The aerodynamic model also contains the ground effects. The aircraft has two elevators with −25 to 25 deg deflection range. The engine model (represented by a 5 s lag) completes the 6-degree of freedom simulation. The aircraft has hydraulic actuators, which drive the primary control surfaces that are modeled as first order lags with a time constant of 50 ms. The rate limit for the actuators is set at 60 deg/s. The controller update rate is set to 20 ms (50 Hz).

The landing task to be executed by the aircraft is shown in Fig. 4. It comprises six distinct phases:

1) Segment 1: Level flight at 600 m, heading −90 deg (from East to West). Velocity is maintained at 83 m/s.

2) Segment 2: A coordinated right turn with bank angle 40 deg at 600 m to align with the runway 0 deg (heading North). Velocity is maintained at 83 m/s.

3) Segment 3: A level flight at 600 m heading 0 deg (toward North). Velocity is maintained at 83 m/s.

4) Segment 4: Descent on glide slope of −6 deg to altitude of 300 m

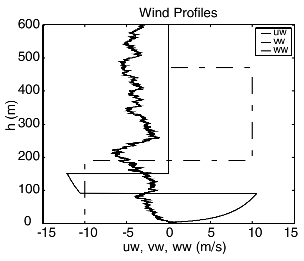


Fig. 5 Wind profiles during landing.

5) Segment 5: Descent at glide slope of −3 deg to 12 m.

Segment 6: Flare and touchdown. Velocity is reduced from 83 m/s to 79 m/s during the flare. The flare segment starts when the altitude descends to 12 m. During this segment the desired altitude follows an exponential model.

Table 1 Touchdown specifications (pill box conditions)

X distance	$100 \le x \le 300$ m
Y distance	$\lvert y\rvert \le 5$ m
Total velocity	$V_T \ge 60$ m/s
Sink rate	$\dot{h} \le 1.0$ m/s
Bank angle	$\lvert\phi\rvert \le 10$ deg

Table 2 Design gains for the SSRC

Network	Λ	Ω	Γ	$\varepsilon_{\min}$	$\varepsilon_{\max}$	r	κ_p
NN1	$\begin{bmatrix}-60 & 0 & 0\\ 0 & -70 & 0\\ 0 & 0 & -5\end{bmatrix}$	$\begin{bmatrix}60\\ 37.8\\ 5\end{bmatrix}$	$\begin{bmatrix}5\\ 5\\ 5\end{bmatrix}$	0.001	0.002	0.877	2.5
NN2	$\begin{bmatrix}18 & 0 & 0\\ 0 & 25 & 0\\ 0 & 0 & -1\end{bmatrix}$	$\begin{bmatrix}13.5\\ 7\\ 1\end{bmatrix}$	$\begin{bmatrix}5\\ 4\\ 5\end{bmatrix}$	0.001	0.002	0.877	2.5
NN3	$\begin{bmatrix}12 & 0 & 0\\ 0 & 15 & 0\\ 0 & 0 & 1\end{bmatrix}$	$\begin{bmatrix}7.5\\ 16\\ 5\end{bmatrix}$	$\begin{bmatrix}5\\ 0.5\\ 5\end{bmatrix}$	0.002	0.003	0.877	2.5

Table 3 Design gains for the MSRC

Network	Λ	Ω	Γ	$\varepsilon_{\min}$	$\varepsilon_{\max}$	r	κ_p
NN1	$\begin{bmatrix} 60 & 0 & 0 \\ -60 & 0 & 0 \\ 7 & -70 & 0 \\ -7 & -70 & 0 \\ 0 & 0 & -5 \end{bmatrix}$	$\begin{bmatrix} 60 \\ 60 \\ 37.5 \\ 37.5 \\ 5 \end{bmatrix}$	$\begin{bmatrix} 5 \\ 5 \\ 5 \end{bmatrix}$	0.001	0.002	0.877	2.5
NN2	$\begin{bmatrix} 18 & 0 & 0 \\ 0 & 25 & 0 \\ 0 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 13.5 \\ 7 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 5 \\ 4 \\ 5 \end{bmatrix}$	0.001	0.002	0.877	2.5
NN3	$\begin{bmatrix} 12 & 0 & 0 \\ 0 & 15 & 0 \\ 0 & 0 & 1.5 \end{bmatrix}$	$\begin{bmatrix} 7.5 \\ 16 \\ 5 \end{bmatrix}$	$\begin{bmatrix} 5 \\ 0.5 \\ 5 \end{bmatrix}$	0.002	0.003	0.877	2.5

The aircraft is subjected to winds while flying this trajectory. The winds are a combination of Dryden turbulence along the x-body axis and deterministic winds in the other two axes (Fig. 5). The sharp step changes in the v_w (at 470 and 190 m) and w_w (150 and 90 m) are particularly noticeable. These profiles represent large horizontal and vertical wind shears, respectively.

The success of the landing mission is evaluated based on a successful touchdown in the landing pill box. The pill box conditions are defined in Table 1.

Multiple control surface failures of hard over type are studied in this paper. A method to compute the feasible region within which successful landings can be undertaken with single or multiple control surface failures is described in [20]

IV. Performance Evaluation

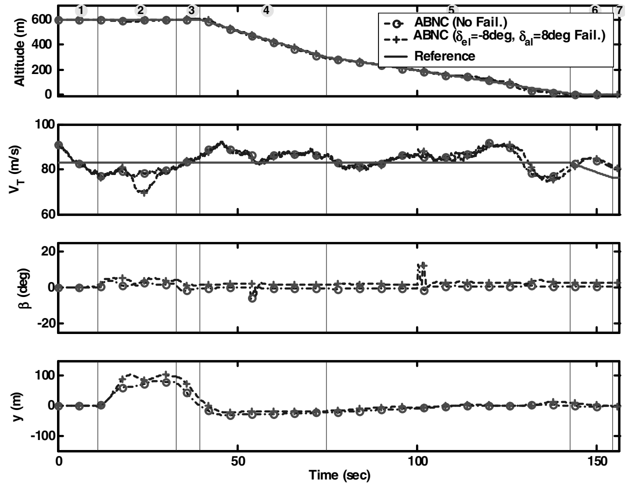
In this section, results are presented which show the ability of ABNC design to meet the touchdown specifications in the presence of failures. In the overall control scheme shown in Fig. 1, the tracking command generator computes the reference command for cross track deviation, altitude, velocity, and sideslip angle (y_d, h_d, V_d, β_d). A detailed description of the navigation algorithm used to compute the reference commands for the autolanding task can be found in [20]. In this study, the controller consists of three neural networks NN1, NN2, and NN3 in cascade with the fourth neural network replaced by an analytic inverse. This is because in the equations motion of the aircraft [10], it can be seen that the equations for $\dot{y}$, $\dot{h}$ are analytic and do not have any uncertainty in them. It is noted that the general proof in Sec. II does not require any of the dynamic equations to possess an analytic inverse. The final design gains are given in Table 2 for the SSRC design. Table 3 gives the gains for the MSRC design.

Simulations were conducted for SSRC and MSRC controllers subjected to left elevator–left aileron and left aileron–right aileron failure scenarios. Figures 6–8 present the results and are analyzed in detail.

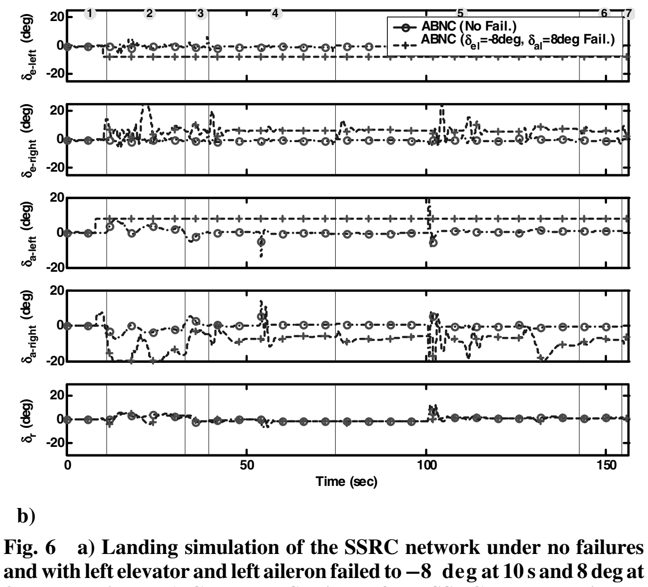
Figure 6a shows the time history of altitude, velocity, sideslip, and cross track deviation for the complete landing trajectory for both normal and failure cases for the SSRC design. The failure case corresponds to the left elevator stuck at −8 deg at 10 s and the left aileron stuck at 8 deg at 8 s. From the altitude and velocity time histories, it can be seen that the performance of SSRC under elevator failure is close to the normal case, except for a residual sideslip in the failure condition. The result is a successful touchdown under failure meeting all the specifications.

Figure 6b shows the time history of the aerodynamic control surfaces as computed by the SSRC. It should also be noted that under the no failure case, the elevator deflections required to achieve the landing are fairly small. In case of the failure of the left elevator and the left aileron to −8 deg and 8 deg, respectively, it is seen that the healthy elevator (right) maintains a deflection of about +6 deg and the healthy aileron maintains a deflection of about −6 deg to offset the pitching and rolling moments of the failed elevator for most of the trajectory. The right aileron saturation seen at 15 s is due to the controller trying to correct the disturbance arising out of the failure while maintaining the turn. The saturation at 100 s in the right elevator and right aileron is due to an encounter with the change in side gust. Subsequent saturations are due to the updraft and downdraft encountered during the severe microburst encounter.

Figure 7a shows the time history for the complete landing trajectory for both normal and failure cases for the MSRC implementation of the ABNC. The failure case is identical to that for the SSRC in Figs. 6a and 6b. It is seen that the overall trajectory in the case of the MSRC is similar to that for the SSRC results except that the dip in velocity at 25 s is more prominent for the former. Figure 7b shows the control surface deflections due to the MSRC. The result is

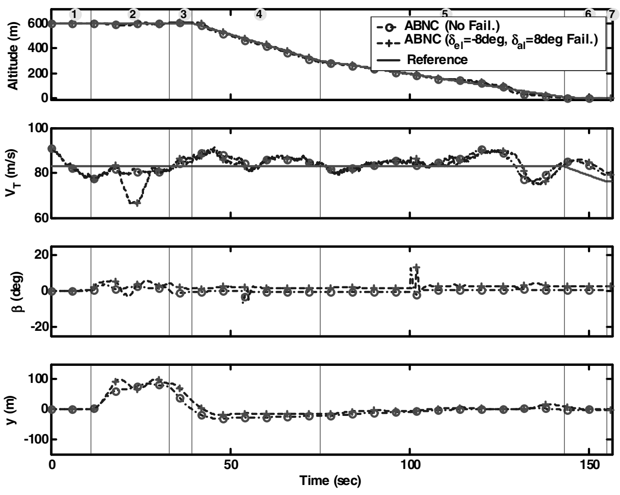


a)

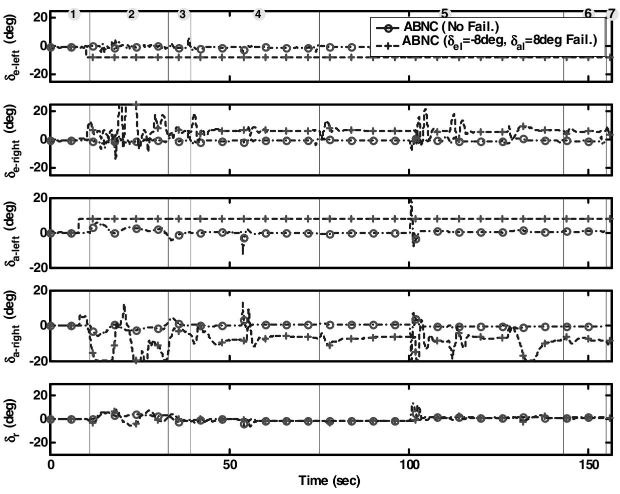


b)

Fig. 6 a) Landing simulation of the SSRC network under no failures and with left elevator and left aileron failed to −8 deg at 10 s and 8 deg at 8 s, respectively. b) Control deflections of the SSRC network with no failures and with left elevator and left aileron failed to −8 deg at 10 s and 8 deg at 8 s, respectively.

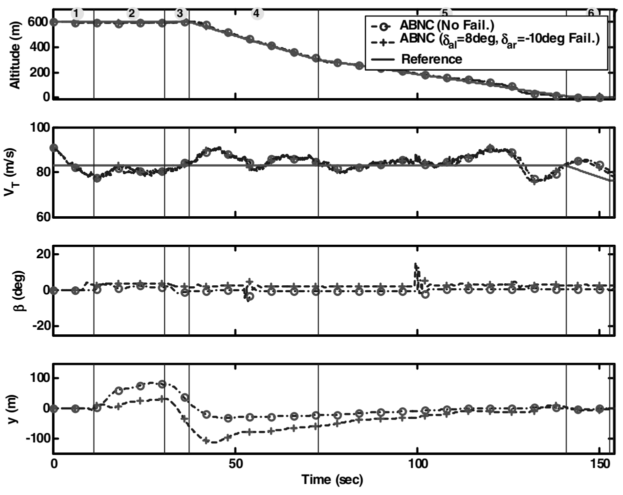


a)

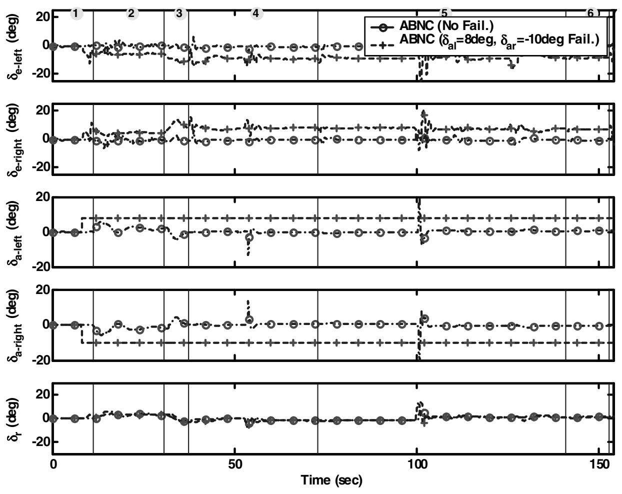


b)

Fig. 7 a) Landing simulation of the MSRC network under no failures and with left elevator and left aileron failed to −8 deg at 10 s and 8 deg at 8 s, respectively. b) Control deflections of the MSRC network with no failures and with left elevator and left aileron failed to −8 deg at 10 s and 8 deg at 8 s, respectively.



a)



b)

Fig. 8 a) Landing simulation of the MSRC network under no failures and with left aileron and right aileron failed to 8 deg at 8 s and −10 deg at 8 s, respectively. b) Control deflections of the MSRC network with no failures and with left aileron and right aileron failed to 8 deg at 8 s and −10 deg at 8 s, respectively.

similar to that for the SSRC, except that the healthy (right) aileron and healthy (right) elevator saturations are of longer duration at 25 s.

The MSRC implementation has the ability to handle both left and right aileron failure. This is not possible with the SSRC. This is demonstrated for the MSRC in the simulation plots shown in Fig. 8a, where the left aileron is failed to 8 deg and simultaneously the right aileron is also failed to −10 deg at 8 s. It is seen that the MSRC is able to handle the two-aileron failure case. The ability of the neural network to apply differential elevator control to achieve fault tolerance in this case arises from the off-diagonal entries in the gain matrix (Table 3).

V. Conclusions

An online learning neural network controller design method called adaptive backstepping neural controller is proposed for reconfigurable flight control systems in the presence of large changes in the aerodynamic characteristics and also control surface failures. The neural controller uses adaptive backstepping and builds up a radial basis neural network starting from zero neurons. Using Lyapnov's theory, tuning rules for all the parameters of the RBFN are derived and a proof of stability in the ultimate bounded sense is given for the ABNC. The stability update rule is not specific to radial basis function networks. It can be used for any other neural network satisfying the universal approximation property. It can also be used with other function representations like polynomials or splines. Application to the trajectory control of an unstable aircraft in landing phase with control surface failures under severe winds is used to demonstrate the ability of the ABNC to stabilize the aircraft and achieve tight touchdown specifications. The above design methodology has been extended to provide a complete fault-tolerant controller design for aircraft with multiple redundant controls.

Appendix

Proof of Theorem 1:

Lyapunov's theory is used to prove the result. The equations for rate of change of error for the system in Eq. (2) are derived first. We illustrate the development for the first and fourth equations in the set of vector differential equations in Eq. (2) and then develop the final result. Differentiating the first and fourth equations in Eq. (6) with respect to time we have

$$\dot{e}_1 = f_1 - \dot{x}_1^d \qquad \dot{e}_4 = f_4 - \dot{x}_4^d \tag{A1}$$

One may expand the first and fourth functions in Eq. (2) as a Taylor series about the applied control and use the mean value theorem to obtain

$$\begin{aligned} f_1 &= f_{1,nn} + b_{11}\cdot\Delta\sigma_1 + b_{12}\cdot\Delta\sigma_2 + b_{13}\cdot\Delta\sigma_3 \\ f_4 &= f_{4,nn} + b_{44}\cdot(\Delta\sigma_4 + e_1) + b_{45}\cdot(\Delta\sigma_5 + e_2) \\ &\quad + b_{46}\cdot(\Delta\sigma_6 + e_3) \end{aligned} \tag{A2}$$

where b_{ij} are the derivatives of the functions f_i with respect to the desired controls σ_j, $j = \{1, \ldots, 11\}$. The shorthand $f_{i,nn}$ represents the functions f_i evaluated at the optimal desired controls $\sigma_{i,nn}$, $i = \{1, \ldots, 11\}$. Substituting Eq. (A2) into Eq. (A1) and using Eq. (5) we obtain

$$\begin{aligned}\dot{e}_1 &= (f_{1,nn} - v_1) + v_1 + b_{11}\cdot\Delta\sigma_1 + b_{12}\cdot\Delta\sigma_2 + b_{13}\cdot\Delta\sigma_{31} \\ &\quad - \dot{x}_1^d = (f_{1,nn} - v_1) - \Gamma_1 e_1 + b_{11}\cdot\Delta\sigma_1 + b_{12}\cdot\Delta\sigma_2 \\ &\quad + b_{13}\cdot\Delta\sigma_{31} - \dot{x}_1^d \\ \dot{e}_4 &= (f_{4,nn} - v_4) + v_4 + b_{44}\cdot(\Delta\sigma_4 + e_1) + b_{45}\cdot(\Delta\sigma_5 + e_2) \\ &\quad + b_{46}\cdot(\Delta\sigma_6 + e_3) - \dot{x}_4^d = (f_{4,nn} - v_4) - \Gamma_4 e_4 + b_{44} \\ &\quad \cdot(\Delta\sigma_4 + e_1) + b_{45}\cdot(\Delta\sigma_5 + e_2) + b_{46}\cdot(\Delta\sigma_6 + e_3) - \dot{x}_4^d\end{aligned} \tag{A3}$$

Following this procedure, one obtains the rate of change of error equations as

$$\begin{aligned}\dot{e}_i &= f_{i,nn} - v_i - \Gamma_i e_i - \dot{x}_i^d \\ &\quad + \begin{cases} \sum_{j=1}^{3} b_{ij}\cdot\Delta\sigma_j, & i = 1,2,3 \\ \sum_{j=4}^{6} b_{ij}\cdot(\Delta\sigma_j + e_{j-3}), & i = 4,5,6 \\ \sum_{j=7}^{9} b_{ij}\cdot(\Delta\sigma_j + e_{j-3}), & i = 7,8,9 \\ \sum_{j=10}^{11} b_{ij}\cdot(\Delta\sigma_j + e_{j-3}), & i = 10,11 \end{cases}\end{aligned} \tag{A4}$$

Expanding Eq. (2) as a Taylor series about the ideal inverse control given by Eq. (7) one obtains

$$\begin{aligned}f_{1,nn} - v_1 &= f_{1,nn} - f_1^* = b_{11}^*\cdot\eta_{1,nn} + b_{12}^*\cdot\eta_{2,nn} + b_{13}^*\cdot\eta_{3,nn} \\ f_{4,nn} - v_4 &= f_{4,nn} - f_4^* = b_{44}^*\cdot\eta_{4,nn} + b_{45}^*\cdot\eta_{5,nn} + b_{46}^*\cdot\eta_{6,nn}\end{aligned} \tag{A5}$$

where b_{ij}^* are the derivatives of the functions f_i with respect to the desired controls σ_j, $j = \{1, \ldots, 11\}$. The shorthand f_i^* represents the functions f_i evaluated at the ideal inverse desired controls $\chi^*, \gamma^*, \mu^*, \alpha^*, \delta_p^*, p^*, q^*, r^*, \delta_a^*, \delta_e^*$, and δ_r^*. As b_{ij}^* are continuous functions, they are Lipschitz on the compact set $\Omega \subset U$. Because the ideal neural network weights and basis functions are bounded, its output is bounded and there exist constants $a_{i,j,k}$, $c_{i,j,k}$ such that

$$\begin{aligned}|f_{1,nn} - v_1| &\le \{a_{1,1,1}|x_1| + a_{1,2,1}|x_2| + a_{1,3,1}|x_3| + a_{1,5,1}|x_5| \\ &\quad + a_{1,6,1}|x_6| + a_{1,9,1}|x_9| + a_{1,11,1}|x_{11}| + c_{1,1,1}|e_1| + c_{1,2,1}|e_2| \\ &\quad + c_{1,3,1}|e_3| + a_{1,0,1}\}\cdot\eta_1 + \{a_{1,1,2}|x_1| + a_{1,2,2}|x_2| + a_{1,3,2}|x_3| \\ &\quad + a_{1,5,2}|x_5| + a_{1,6,2}|x_6| + a_{1,9,2}|x_9| + a_{1,11,2}|x_{11}| + c_{1,1,2}|e_1| \\ &\quad + c_{1,2,2}|e_2| + c_{1,3,2}|e_3| + a_{1,0,2}\}\cdot\eta_2 + \{a_{1,1,3}|x_1| + a_{1,2,3}|x_2| \\ &\quad + a_{1,3,3}|x_3| + a_{1,5,3}|x_5| + a_{1,6,3}|x_6| + +a_{1,9,3}|x_9| \\ &\quad + a_{1,11,3}|x_{11}| + c_{1,1,3}|e_1| + c_{1,2,3}|e_2| + c_{1,3,3}|e_3| + a_{1,0,3}\}\cdot\eta_3 \\ |f_{4,nn} - v_4| &\le \{a_{4,4,4}|x_4| + a_{4,5,4}|x_5| + a_{4,6,4}|x_6| + a_{4,8,4}|x_8| \\ &\quad + a_{4,9,4}|x_9| + a_{4,11,4}|x_{11}| + c_{4,4,4}|e_4| + c_{4,5,4}|e_5| \\ &\quad + c_{4,6,4}|e_6| + a_{4,0,4}\}\cdot\eta_4 + \{a_{4,4,5}|x_4| + a_{4,5,5}|x_5| \\ &\quad + a_{4,6,5}|x_6| + a_{4,8,5}|x_8| + a_{4,9,5}|x_9| + a_{4,11,5}|x_{11}| \\ &\quad + c_{4,4,5}|e_4| + c_{4,5,5}|e_5| + c_{4,6,5}|e_6| + a_{4,0,5}\}\cdot\eta_5 \\ &\quad + \{a_{4,4,6}|x_4| + a_{4,5,6}|x_5| + a_{4,6,6}|x_6| + a_{4,8,6}|x_8| \\ &\quad + a_{4,9,6}|x_9| + a_{4,11,6}|x_{11}| + c_{4,4,6}|e_4| + c_{4,5,6}|e_5| \\ &\quad + c_{4,6,6}|e_6| + a_{4,0,6}\}\cdot\eta_6\end{aligned} \tag{A6}$$

where $\eta_i = \sup_{x\in\Omega}|\eta_{i,nn}|$, $i = \{1, \ldots, 11\}$. Note that the function f_4 also depends on the control vector $\boldsymbol{u}$. From Assumption 3, the control inputs required to complete the desired maneuver are bounded ($\boldsymbol{u} \le \boldsymbol{u}_m$). Therefore, in Eq. (A6), this dependency is absorbed in the constants $a_{4,0,i}$, $i = 4,5,6$ [13]. The terms with $c_{i,j,k}|e_j|$ appearing on the right-hand sides of Eq. (A6) arise due to the dependence of b_{ij}^* on v_i.

As a consequence of Assumptions 2 and 3 it is seen that the reference commands generated by the cascaded neural networks are also bounded provided the weight update rule results in a stable system (to be proved later in this section). This taken with Eq. (6) gives

$$|x_i| \le d_{i,1}\cdot|e_i| + d_{i,0}, \qquad i = \{1, \ldots, 11\} \tag{A7}$$

From Eqs. (A6) and (A7) one can write

$$\begin{aligned}|f_{1,nn} - v_1| &\le \{C_{1,1,1}|e_1| + C_{1,2,1}|e_2| + C_{1,3,1}|e_3| + C_{1,5,1}|e_5| \\ &\quad + C_{1,6,1}|e_6| + C_{1,9,1}|e_9| + C_{1,11,1}|e_{11}| + C_{1,0,1}\}\cdot\eta_1 \\ &\quad + \{C_{1,1,2}|e_1| + C_{1,2,2}|e_2| + C_{1,3,2}|e_3| + C_{1,5,2}|e_5| + C_{1,6,2}|e_6| \\ &\quad + C_{1,9,2}|e_9| + C_{1,11,2}|e_{11}| + C_{1,0,2}\}\cdot\eta_2 + \{C_{1,1,3}|e_1| \\ &\quad + C_{1,2,3}|e_2| + C_{1,3,3}|e_3| + C_{1,5,3}|e_5| + C_{1,6,3}|e_6| + C_{1,9,3}|e_9| \\ &\quad + C_{1,11,3}|e_{11}| + C_{1,0,3}\}\cdot\eta_3 \\ |f_{4,nn} - v_4| &\le \{C_{4,4,4}|e_4| + C_{4,5,4}|e_5| + C_{4,6,4}|e_6| + C_{4,8,4}|e_8| \\ &\quad + C_{4,9,4}|e_9| + C_{4,11,4}|e_{11}| + C_{4,0,4}\}\cdot\eta_4 + \{C_{4,4,5}|e_4| \\ &\quad + C_{4,5,5}|e_5| + C_{4,6,5}|e_6| + C_{4,8,5}|e_8| + C_{4,9,5}|e_9| + C_{4,11,5}|e_{11}| \\ &\quad + C_{4,0,5}\}\cdot\eta_5 + \{C_{4,4,6}|e_4| + C_{4,5,6}|e_5| + C_{4,6,6}|e_6| \\ &\quad + C_{4,8,6}|e_8| + C_{4,9,6}|e_9| + C_{4,11,6}|e_{11}| + C_{4,0,6}\}\cdot\eta_6\end{aligned} \tag{A8}$$

where the $C_{i,j,k}$ are related to the $a_{i,j,k}$, $c_{i,j,k}$, and $d_{i,j}$. Equations for the rate of change of error similar to Eq. (A3) and the inequalities similar to Eq. (A8) are obtained for each of the 11 equations in (2). After this we are ready to consider the stability of the system.

Consider the candidate Lyapunov function

$$V(e_1, \Delta\sigma_1, \ldots, e_{11}, \Delta\sigma_{11}) = \sum_{i=1}^{11}\left(\frac{e_i^2}{2} + \frac{\Delta\sigma_i^2}{2}\right) \tag{A9}$$

This function is radially unbounded. Taking its derivative along the error system for Eq. (2), one obtains

$$\dot{V} = \sum_{i=1}^{11}(e_i\cdot\dot{e}_i + \Delta\sigma_i\cdot\Delta\dot{\sigma}_i) \tag{A10}$$

Apply the update rule (14) to the above equation after noting that $\Delta\dot{\sigma}_i = \dot{\sigma}_i - \dot{\sigma}_{i,nn}$, $i = 1, \ldots, 11$ and use Eq. (A4). This results in

$$\begin{aligned}\dot{V} &= \sum_{i=1}^{3}\left[e_i\cdot\left(f_{i,nn} - v_i - \Gamma_i e_i - \dot{x}_i^d + \sum_{j=1}^{3} b_{ij}\cdot\Delta\sigma_j\right)\right. \\ &\quad \left. - \Delta\sigma_i\cdot\left(\Omega_i\sigma_i + \dot{\sigma}_{i,nn} + \sum_{j=1}^{3}\Lambda_{ij}\cdot e_j\right)\right] \\ &\quad + \sum_{i=4}^{6}\left[e_i\cdot\left(f_{i,nn} - v_i - \Gamma_i e_i - \dot{x}_i^d + \sum_{j=4}^{6} b_{ij}\cdot(\Delta\sigma_j + e_{j-3})\right)\right. \\ &\quad \left. - \Delta\sigma_i\cdot\left(\Omega_i\sigma_i + \dot{\sigma}_{i,nn} + \sum_{j=4}^{6}\Lambda_{ij}\cdot e_j\right)\right] \\ &\quad + \sum_{i=7}^{9}\left[e_i\cdot\left(f_{i,nn} - v_i - \Gamma_i e_i - \dot{x}_i^d + \sum_{j=7}^{9} b_{ij}\cdot(\Delta\sigma_j + e_{j-3})\right)\right. \\ &\quad \left. - \Delta\sigma_i\cdot\left(\Omega_i\sigma_i + \dot{\sigma}_{i,nn} + \sum_{j=7}^{9}\Lambda_{ij}\cdot e_j\right)\right] \\ &\quad + \sum_{i=10}^{11}\left[e_i\cdot\left(f_{i,nn} - v_i - \Gamma_i e_i - \dot{x}_i^d + \sum_{j=10}^{11} b_{ij}\cdot(\Delta\sigma_j + e_{j-3})\right)\right. \\ &\quad \left. - \Delta\sigma_i\cdot\left(\Omega_i\sigma_i + \dot{\sigma}_{i,nn} + \sum_{j=10}^{11}\Lambda_{ij}\cdot e_j\right)\right]\end{aligned} \tag{A11}$$

We take absolute values on the right-hand side of Eq. (A11), substitute equations similar to Eq. (A8) for the entire system, and use

Eq. (12). This gives us terms like $C \cdot e_i^2$, $C \cdot |e_i|$, $-|\dot{\sigma}_{i,nn}||\Delta\sigma_i|$, $-\Omega_i \cdot |\sigma_i||\Delta\sigma_i|$, $C \cdot |e_i||e_j|$, and $C \cdot |e_i||\Delta\sigma_j|$. The last three terms are simplified using the inequalities $C \cdot ab \leq C^2 \cdot a^2/4 + b^2$ and $-C \cdot a(a+b) \leq -C \cdot |a|(|a| - |b|)$, where a, b are real numbers and C is a positive constant.

After collecting like terms and using the bounds given by Eq. (13), Eq. (A11) can be rewritten in shorthand as

$$\dot{V} \leq \sum_{i=1}^{11} -B_i \cdot e_i^2 + D_i \cdot |\Delta\sigma_i| \cdot |e_i| + E_i \cdot |e_i| - |\Delta\sigma_i| \cdot (F_i \cdot |\Delta\sigma_i| - G_i) \tag{A12}$$

where

$$D_i = |b_{ii} - \Lambda_{ii}|, \qquad i = 1, \ldots, 11$$

$$G_i = \Omega_i \cdot \sigma_{i,m} + \sigma_{i,md}, \qquad i = 1, \ldots, 11$$

$$E_i = x_{i,md} + \begin{cases} \sum_{j=1}^{3} C_{i,o,j} \cdot \eta_j \\ \sum_{j=4}^{6} C_{i,o,j} \cdot \eta_j \\ \sum_{j=7}^{9} C_{i,o,j} \cdot \eta_j \\ \sum_{j=10}^{11} C_{i,o,j} \cdot \eta_j \end{cases} \qquad i = 1, \ldots, 11$$

$$F_1 = \Omega_1 - (b_{21} - \Lambda_{12})^2/4 - (b_{31} - \Lambda_{13})^2/4$$

$$F_2 = \Omega_2 - (b_{12} - \Lambda_{21})^2/4 - (b_{32} - \Lambda_{23})^2/4$$

$$F_3 = \Omega_3 - (b_{13} - \Lambda_{31})^2/4 - (b_{23} - \Lambda_{32})^2/4$$

$$F_4 = \Omega_4 - (b_{54} - \Lambda_{45})^2/4 - (b_{64} - \Lambda_{46})^2/4$$

$$F_5 = \Omega_5 - (b_{45} - \Lambda_{54})^2/4 - (b_{65} - \Lambda_{56})^2/4$$

$$F_6 = \Omega_6 - (b_{46} - \Lambda_{64})^2/4 - (b_{56} - \Lambda_{65})^2/4$$

$$F_7 = \Omega_7 - (b_{87} - \Lambda_{78})^2/4 - (b_{97} - \Lambda_{79})^2/4$$

$$F_8 = \Omega_8 - (b_{78} - \Lambda_{87})^2/4 - (b_{98} - \Lambda_{89})^2/4$$

$$F_9 = \Omega_9 - (b_{79} - \Lambda_{97})^2/4 - (b_{89} - \Lambda_{98})^2/4$$

$$F_{10} = \Omega_{10} - (b_{11,10} - \Lambda_{10,11})^2/4$$

$$F_{11} = \Omega_{11} - (b_{10,11} - \Lambda_{11,10})^2/4$$

$$B_1 = \Gamma_1 - \sum_{i=1}^{3} C_{1,1,i} \cdot \eta_i - 9 - \frac{1}{4}\sum_{j=7}^{8}\left(\sum_{i=7}^{9} C_{j,1,i} \cdot \eta_i\right)^2$$

$$B_2 = \Gamma_2 - \sum_{i=1}^{3} C_{2,2,i} \cdot \eta_i - 8 - \frac{1}{4}\left(\sum_{i=1}^{3}(C_{1,2,i} + C_{2,1,i}) \cdot \eta_i\right)^2 - \frac{1}{4}\sum_{j=7}^{8}\left(\sum_{i=7}^{9} C_{j,2,i} \cdot \eta_i\right)^2$$

$$B_3 = \Gamma_3 - \sum_{i=1}^{3} C_{3,3,i} \cdot \eta_i - 7 - \frac{1}{4}\sum_{j=1}^{2}\left(\sum_{i=1}^{3}(C_{j,3,i} + C_{3,j,i}) \cdot \eta_i\right)^2 - \frac{1}{4}\sum_{j=7}^{8}\left(\sum_{i=7}^{9} C_{j,3,i} \cdot \eta_i\right)^2$$

$$B_4 = \Gamma_4 - \sum_{i=4}^{6} C_{4,4,i} \cdot \eta_i - 8 - \frac{1}{4}\sum_{i=4}^{6} b_{4,i}^2$$

$$B_5 = \Gamma_5 - \sum_{i=4}^{6} C_{5,5,i} \cdot \eta_i - 7 - \frac{1}{4}\sum_{j=1}^{3}\left(\sum_{i=1}^{3} C_{j,5,i} \cdot \eta_i + b_{53+j}\right)^2 - \frac{1}{4}\left(\sum_{i=4}^{6}(C_{4,5,i} + C_{5,4,i}) \cdot \eta_i\right)^2$$

$$B_6 = \Gamma_6 - \sum_{i=4}^{6} C_{6,6,i} \cdot \eta_i - 6 - -\frac{1}{4}\sum_{j=1}^{3}\left(\sum_{i=1}^{3} C_{j,6,i} \cdot \eta_i + b_{63+j}\right)^2 - \frac{1}{4}\sum_{j=4}^{5}\left(\sum_{i=4}^{6}(C_{j,6,i} + C_{6,j,i}) \cdot \eta_i\right)^2 - \frac{1}{4}\left(\sum_{i=7}^{9}(C_{7,6,i} + C_{8,6,i} + C_{9,6,i}) \cdot \eta_i\right)^2$$

$$B_7 = \Gamma_7 - \sum_{i=7}^{9} C_{7,7,i} \cdot \eta_i - 10 - \frac{1}{4}\sum_{i=7}^{9} b_{7,i}^2$$

$$B_8 = \Gamma_8 - \sum_{i=7}^{9} C_{8,8,i} \cdot \eta_i - 9 - \frac{1}{4}\sum_{j=4}^{6}\left(\sum_{i=4}^{6} C_{j,8,i} \cdot \eta_i + b_{83+j}\right)^2 - \frac{1}{4}\left(\sum_{i=7}^{9}(C_{7,8,i} + C_{8,7,i}) \cdot \eta_i\right)^2$$

$$B_9 = \Gamma_9 - \sum_{i=7}^{9} C_{9,9,i} \cdot \eta_i - 4 - \frac{1}{4}\sum_{j=4}^{6}\left(\sum_{i=4}^{6} C_{j,9,i} \cdot \eta_i + b_{93+j}\right)^2 - \frac{1}{4}\sum_{j=1}^{3}\left(\sum_{i=1}^{3} C_{j,9,i} \cdot \eta_i + \sum_{i=7}^{9} C_{9,j,i} \cdot \eta_i\right)^2 - \frac{1}{4}\sum_{j=7}^{8}\left(\sum_{i=7}^{9}(C_{j,9,i} + C_{9,j,i}) \cdot \eta_i\right)^2 - \frac{1}{4}\left(\sum_{i=10}^{11} C_{10,9,i} \cdot \eta_i\right)^2$$

$$B_{10} = \Gamma_{10} - \sum_{i=10}^{11} C_{10,10,i} \cdot \eta_i - 3 - \frac{1}{4}\sum_{i=10}^{11} b_{10,i}^2$$

$$B_{11} = \Gamma_{11} - \sum_{i=10}^{11} C_{11,11,i} \cdot \eta_i - 1 - \frac{1}{4}\sum_{j=1}^{3}\left(\sum_{i=1}^{3} C_{j,11,i} \cdot \eta_i\right)^2 - \frac{1}{4}\sum_{j=4}^{6}\left(\sum_{i=4}^{6} C_{j,11,i} \cdot \eta_i\right)^2 - \frac{1}{4}\sum_{j=7}^{8}\left(\sum_{i=7}^{9} C_{j,11,i} \cdot \eta_i + b_{11,3+j}\right)^2 - \frac{1}{4}\left(\sum_{i=7}^{9} C_{9,11,i} \cdot \eta_i + \sum_{i=10}^{11} C_{11,9,i} \cdot \eta_i\right)^2 - \frac{1}{4}\left(\sum_{i=10}^{11} C_{10,11,i} \cdot \eta_i + \sum_{i=10}^{11} C_{11,10,i} \cdot \eta_i\right)^2$$

With a choice of gains Γ_i, Λ_{ij}, and Ω_i such that Eq. (15) holds, we have

$$\dot{V} \leq \sum_{i=1}^{11} \Bigg\{ -\left(|e_i| \cdot \sqrt{B_i - 1} - \frac{E_i/2}{\sqrt{B_i - 1}}\right)^2 - \left(|e_i| - \frac{D_i}{2}|\Delta\sigma_i|\right)^2 + \frac{E_i^2/4}{B_i - 1} - \left(|\Delta\sigma_i| \cdot \sqrt{F_i - D_i^2/4} - \frac{G_i/2}{\sqrt{F_i - D_i^2/4}}\right)^2 + \frac{G_i^2/4}{F_i - D_i^2/4}\Bigg\} \tag{A13}$$

We find that the right-hand side of Eq. (A13) represents a hyper ellipsoid in the space of the error variables $\Delta\sigma_i$, e_i, $i = \{1, \ldots, 11\}$. The condition $\dot{V} \leq 0$ holds outside of this ellipsoid subject to the requirements given by the update rule (14) and Assumptions 1–3. Further, the ellipsoid touches the origin of the error space. Therefore, as discussed in [21], this proves the uniform ultimate boundedness of all signals.

References

[1] Napolitano, M. R., Naylor, S., Neppach, C., and Casdorph, V., "On-Line Learning Nonlinear Direct Neurocontrollers for Restructurable Control Systems," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 1, 1995, pp. 170–176.

[2] Pesonen, U. J., Steck, J. E., Rokhsaz, K., Bruner, H. S., and Duerksen, N., "Adaptive Neural Network Inverse Controller for General Aviation Safety," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 3, 2004, pp. 434–443.

[3] Ferrari, S., and Stengel, R. F., "Online Adaptive Critic Flight Control," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 5, 2004, pp. 777–786.

[4] Calise, A. J., Lee, S., and Sharma, M., "Development of a Reconfigurable Flight Control Law for Tailless Aircraft," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 5, 2001, pp. 896–902.

[5] Krstic, M., Kanellakopoulos, I., and Kokotovic, P. V., *Nonlinear Adaptive Control Design*, Wiley, New York, 1995.

[6] Hovakimyan, N., Nardi, F., Calise A. J., and Kim, N., "Adaptive Output Feedback Control of Uncertain Nonlinear Systems Using Single-Hidden-Layer Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 13, No. 6, 2002, pp. 1420–1431.

[7] Shin, D.-H., and Kim, Y., "Reconfigurable Flight Control System Design Using Adaptive Neural Networks," *IEEE Transactions on Control Systems Technology*, Vol. 12, No. 1, 2004, pp. 87–100.

[8] Boskovic, J. D., Chen, L., and Mehra, R. K., "Adaptive Control Design for Nonaffine Models Arising in Flight Control," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 2, 2004, pp. 209–217.

[9] Ge, S. S., and Zhang, J., "Neural-Network Control of Nonaffine Nonlinear System with Zero Dynamics by State and Output Feedback," *IEEE Transactions on Neural Networks*, Vol. 14, No. 4, 2003, pp. 900–918.

[10] Bugajski, D. J., and Enns, D. F., "Nonlinear Control Law with Application to High Angle-of-Attack Flight," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 3, 1992, pp. 761–767.

[11] Liu, G. P., Kadirkamanathan, V., and Billings, S. A., "Variable Neural Networks for Adaptive Control of Nonlinear Systems," *Transactions on Systems, Man and Cybernetics—Part C: Applications And Reviews*, Vol. 29, No. 1, 1999, pp. 34-43.

[12] Lu, Y., Sundararajan, N., and Saratchandran, P., "A Sequential Learning Scheme for Function Approximation Using Minimal Radial Basis Function Neural Networks," *Neural Computation*, Vol. 9, No. 2, 1997, pp. 461–478.

[13] Pashilkar, A. A., Sundararajan, N., and Saratchandran, P., "Adaptive Back-Stepping Neural Controller for Reconfigurable Flight Control Systems," *IEEE Transactions on Control Systems Technology*, Vol. 14, No. 3, May 2006, pp. 553–561.

[14] Johnson, E. N., and Kannan, S. K., "Adaptive Trajectory Control for Autonomous Helicopters," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 3, May/June 2005, pp. 524–538.

[15] Kanellakopoulos, I., Kokotovic, P. V., and Morse, A. S., "Systematic Design of Adaptive Controllers for Feedback Linearizable Systems," *IEEE Transactions on Automatic Control*, Vol. 36, No. 11, 1991, pp. 1241–1253.

[16] Park, J., and Sandberg, I. W., "Universal Approximation Using Radial Basis Function Networks," *Neural Computation*, Vol. 3, No. 2, 1991, pp. 246–257.

[17] Slotine, J.-J. E., and Li, W., *Applied Nonlinear Control*, Prentice–Hall, Englewood Cliffs, NJ, 1991.

[18] Pashilkar, A. A., Sundararajan, N., and Saratchandran, P., "Adaptive Back-Stepping Neural Controller for Aircraft," School of Electrical and Electronic Engineering, Nanyang Technological University, TR EEE4/003/05, Feb. 2005.

[19] Nguyen, L. T., Ogburn, M. E., Gilbert, W. P., Kibler, K. S., Brown, P. W., and Deal, P. L., "Simulator Study of Stall/Post-Stall Characteristics of a Fighter Airplane with Relaxed Longitudinal Static Stability," NASA Technical Paper 1538, Dec. 1979.

[20] Pashilkar, A. A., Sundararajan, N., and Saratchandran, P., "Neuro Controller for Aircraft Auto Landing Under Severe Winds and Control Surface Failures, AIAA Paper 2005-0914, 2005.

[21] Narendra, K., and Annaswamy, A., *Stable Adaptive Control*, Prentice–Hall, Englewood Cliffs, NJ, 1995.