A Sparse Neural Network Approach to Model Reference Adaptive Control with Hypersonic Flight Applications

Scott A. Nivison*
University of Florida, Gainesville, FL, 32611, USA
Pramod P. Khargonekar†
University of California, Irvine, CA, 92697, USA

Neural network-based model reference adaptive control (MRAC) is an effective architecture used in the flight control community to combat significant uncertainties where the structure of the uncertainty is unknown. In our previous work, a novel adaptive control architecture called sparse neural network (SNN) was developed in order to improve long-term learning and transient performance of flight vehicles with persistent uncertainties which lie in various regions throughout the operating regime. The SNN is designed to operate with small learning rates in order to avoid high-frequency oscillations and utilizes only a small number of active neurons in the adaptive controller in order to reduce the computational burden on the processor. In this paper, we enhance the SNN architecture by developing an innovative adaptive control term which is used to mitigate a control effectiveness matrix. Furthermore, we design a robust control term and a strict dwell time condition in order to ensure stability while switching between segments. We demonstrate the effectiveness of the SNN approach by controlling a sophisticated hypersonic vehicle with flexible body effects.

I. Introduction

THE control of a hypersonic flight vehicle (HSV) is an especially challenging task due to the extreme changes in aircraft dynamics during flight and the vastness of the encountered flight envelope [1]. Furthermore, these vehicles operate in environments with strong structural, propulsion, thermal, and control system interactions [2]. Fortunately, over the last couple decades, the model fidelity of HSVs have increased due to improved modeling capabilities. For example, models have emerged that account for the fluid-structure interaction of HSVs. These interactions are known to cause unsteady aerodynamic forces and moments that result in vibration of the flight vehicle. This phenomenon violates the standard quasi-steady airflow assumption typically used to model aircraft [3]. Another major modification to the typical flight vehicle model has been the development of a structural dynamic model with flexible body effects from methods such as the assumed modes method. This method uses the volatile mass of the vehicle along with a unsteady heat transfer model to determine frequencies and the mode shapes for the HSV [2, 4]. It is worth noting that flexible body dynamic modeling is also a significant area of interest for extremely large flexible subsonic flight vehicles such as X-HALE (high altitude long endurance). These vehicles possess low-frequency structural vibration modes which can cause large nonlinear body deformations [5]. In addition, endurance flight vehicles operate at high angles of attack where nonlinear effects become prominent uncertainties in the system dynamics.

The goal of this research is to further develop a recently conceptualized novel adaptive control architecture called sparse neural network (SNN) adaptive control (see Ref. [6]) and demonstrate the capabilities of the SNN by controlling a HSV with flexible body effects. The use of an adaptive controller is necessary because of the highly nonlinear and time-varying nature of the flight vehicle dynamics as well as the existence of significant unmodeled dynamics. The adaptive controller aims to reduce or cancel these uncertainties while achieving ideal tracking performance of the HSV. Often the case with flight control, we assume the existence of a nominal controller which is typically designed using linear quadratic (LQ) optimal control methods around various flight conditions (e.g. dynamic pressure, altitude) with regard to transient (e.g. rise-time, settling time) and frequency metrics (e.g. sensitivity functions, gain and phase

*Doctoral Student, Department of Electrical and Computer Engineering, Student Member AIAA; allme@ufl.edu.
†Vice Chancellor for Research and Professor of Electrical Engineering and Computer Science; pramod.khargonekar@uci.edu.

margins) [7]. One adaptive architecture that seamlessly allows the nominal controller to be augmented with an adaptive controller is model reference adaptive control (MRAC). Within the MRAC framework, we employ the use of neural networks (NN) and their function approximation capabilities to compensate for the significant dynamic uncertainties of the system [7, 8].

Typical neural network (NN) approaches for flight control utilize a Gaussian radial basis function (RBF) NN where RBFs are distributed to fill the operating space with fixed centers and widths [9]. The output of the RBFs determines the inner layer weights of the NN while the outer layer weights are determined by the adaptive law. Unfortunately, the accuracy of fixed RBF based approach often varies based on the number of RBFs and the locations of the RBF centers [10]. However, numerous authors have explored dynamically growing and shrinking the number of RBFs as well as changing the locations of the RBF centers during run-time to improve transient performance [10, 11]. In contrast to the RBF approach, the single hidden layer (SHL) neural network-based adaptive controller operates by updating both the inner layer weights and outer layer weights concurrently. Perhaps due to the local support and more desirable learning framework, RBF NNs have remained the more popular choice [7]. However, empirical evidence suggests that SHL networks achieve slightly better tracking performance on control tasks when using moderately high learning rates [7, 12, 13].

Recently, there has been increased research interest in the area of switched nonlinear systems, fuzzy logic, intelligent control, and neural networks due to numerous breakthroughs in Lyapunov-based stability methods for switched and hybrid systems [14, 15]. For instance, neural network-based fuzzy logic techniques have been developed for MRAC systems which aim to modify the reference model online in order to improve transient performance through the use of a supervisory loop [16]. Additionally, adaptive neural networks have recently been used in both SHL and RBF networks in order to augment switched robust baseline controllers for robot manipulator and unmanned aerial vehicles [17, 18]. Moreover, control methodologies have been developed for hypersonic and highly nonlinear flight vehicles using fuzzy logic and switched tracking controllers [17, 19]. Furthermore, the stability of adaptive RBF networks with dwell time conditions that dynamically add and remove nodes were investigated for systems that include switched dynamics [11, 20].

With regards to more conventional adaptive systems, we noted that they typically require high learning rates with a large number of neurons in the controller in order to achieve fast adaptation and swiftly reduce the system tracking errors. Also, the number of neurons is often less than ideal and restricted based on computational constraints [21]. This can result in actuator overuse, increased controller effort, and high-frequency oscillations due to exciting unmodeled dynamics in the control bandwidth [22]. By reducing the size of the learning rates, the oscillations will often subside, but the tracking performance becomes sluggish. Using high learning rates for flight vehicles such as HSVs where there exists a strong coupling between the rigid body dynamics and the structural modes of the vehicle could result in instability [22, 23].

Inspired by the success of distributed sparse neural networks in the machine learning community, the SNN adaptive control architecture was originally conceived to improve controller memory and tracking performance of flight vehicles with practical processor limitations that encounter persistent significant uncertainties [6]. The SNN succeeds by segmenting the flight envelope, increasing the overall number of neurons available to the adaptive system, and only selecting a small percentage of those neurons to be active in the adaptive controller based on location within the flight envelope. The additional neural network memory allows the SNN controller to store and recall neural network weights and uncertainty estimates from each segmented region in the flight envelope. Hence, the beneficial structure allows additional performance benefits over a traditional single hidden layer (SHL) adaptive controller [6].

In this paper, we provide two key developments that lead to significantly improved tracking performance based on simulation results and safe switching. Firstly, motivated by an adaptive term introduced in [7], we develop adaptive control terms which mitigate the effect of an unknown control effectiveness matrix on the baseline, adaptive, and robust controllers. The second development was inspired by previous work in dynamic RBF adaptive control research (see Ref. [20]), we derive a robust control term which is used to calculate a dwell time condition for the switched system. The inclusion of the robust control term along with a newly derived dwell time condition is used to ensure safe switching between segments. In our work, the robust control term is only activated when the norm of the tracking error exceeds preset bounds. While inside the error bounds, we disable the robust controller in order to allow the SNN to maximize learning and control the vehicle more precisely.

There are many differences that provide a clear distinction between the SNN adaptive controller and other switched adaptive control research efforts. First, the SNN architecture uses a segmented flight envelope approach to select a small number of active nodes based on operating conditions which encourages local learning and reduces the computational burden on the processor. Unlike typical RBF based switching schemes, nodes are not activated according

to a performance metric or (RBF) neuron output [10, 11]. Instead, neurons are assigned to specific segments within the flight space and certain positions within that segment. Additionally, the SNN allows segments to share nodes with one another in order to smooth transition between segments and improve transient performance. Hence, certain nodes and associated weights can remain active while operating in multiple segments. In addition, the concept of the Voronoi diagram is used to quickly and accurately partition the flight envelope. During run-time, we use information generated by a nearest neighbor graph to efficiently calculate the current operating segment which is used for control. As mentioned in the previous paragraph, newly developed improvements to the SNN also provide distinction. These include additional adaptive control terms (control effectiveness) and a robust control term (dwell time condition).

In addition to the previously mentioned developments, we assess the performance of the SNN adaptive controller using a HSV model with flexible body effects. For comparison, we also include analysis results for the more conventional SHL approach.

II. Augmented Model Reference Adaptive Control Formulation

We start by considering a class of n dimensional multiple input multiple output (MIMO) system dynamics with m inputs in the form:

$$\begin{aligned} \dot{x} &= Ax + B\Lambda\left(u + f(x)\right) + B_{ref}y_{cmd} \\ y &= Cx \end{aligned} \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $B_{ref} \in \mathbb{R}^{n\times m}$, and $C \in \mathbb{R}^{p\times n}$ are known matrices. We assume an external bounded time-varying command as $y_{cmd} \in \mathbb{R}^m$. The control effectiveness matrix $\Lambda \in \mathbb{R}^{m\times m}$ is an unknown constant diagonal matrix with uncertain diagonal elements, denoted Λ_i, which are assumed to be strictly positive. We assume the control effectiveness matrix (Λ) can be upper bounded in by a constant term which we denote $\bar{\Lambda}$. The matched uncertainty in the system is represented by the continuously differentiable function $f(x) \in \mathbb{R}^m$. We define the state tracking error as $e = x - x_{ref}$ and the output tracking error as $e_y = y - y_{cmd}$. The system state vector, x, which appears in Eq. (1), contains the traditional plant state vector, x_p, along with the integral tracking error as a system state, i.e. $x = (e_I\ x_p) \in \mathbb{R}^n$. The inclusion of this augmented state vector creates the $B_{ref}y_{cmd}$ term in the extended open-loop dynamics, Eq. (1). See [7] for more details and derivations of this model.

We now discuss the reference model for the MRAC architecture which defines the ideal closed-loop behavior of the system under nominal conditions. To begin design, we remove the uncertainties from the system dynamics stated in Eq. (1). This results in the following ideal extended open-loop dynamics

$$\begin{aligned} \dot{x} &= Ax + Bu + B_{ref}y_{cmd} \\ y &= Cx. \end{aligned} \tag{2}$$

Next, we assume the baseline controller takes the following form:

$$u_{BL} = -K_{LQR}x \tag{3}$$

$$= -e_I K_I - x_p K_P \tag{4}$$

where K_{LQR} are fixed baseline controller gains typically designed using systematic optimal linear quadratic (LQ) control methods [24]. The systematic approach to design LQ gains allow the control designer to take into consideration robust (e.g. margins, singular values, and loop shaping) and performance metrics (e.g. rise-time, overshoot, and undershoot) while utilizing traditional analysis tools (e.g. root locus, bode plots). For this paper, we assume the baseline controller is in proportional-integral (PI) form with gains (K_{LQR}) consisting of integral (K_I) and proportional gains (K_P).

By substituting the form of the baseline controller defined in Eq. (3) into the ideal extended open-loop dynamics shown in Eq. (2), we derive the form of the closed-loop reference model dynamics as

$$\begin{aligned} \dot{x}_{ref} &= A_{ref}x_{ref} + B_{ref}y_{cmd} \\ y_{ref} &= C_{ref}x_{ref} \end{aligned} \tag{5}$$

where we assume the pair (A, B) is controllable, $C_{ref} = C \in \mathbb{R}^{p\times n}$ is known, and $A_{ref} = A - BK_{LQR}^T \in \mathbb{R}^{n\times n}$ is designed to be Hurwitz, see Eq. (5). Note that the proper design of the baseline controller gains in Eq. (3) results in a robust linear controller with ideal transient characteristics.

The model matching conditions for the previously defined MRAC system can be described as follows. Given a constant unknown positive definite matrix Λ and a Hurwitz matrix A_{ref}, there exists a constant unknown gain matrix $K_{\Lambda_{BL}}$ such that

$$A_{ref} = A - B\Lambda K_{\Lambda_{BL}}^T. \tag{6}$$

In order to satisfy both the model matching conditions and the closed-loop reference model dynamics in Eq. (5), we define the ideal gain matrix, $K_{\Lambda_{BL}}$, as

$$K_{\Lambda_{BL}} = K_{LQR}\Lambda^{-1} \tag{7}$$

where $K_{\Lambda_{BL}}$ has been proven to exist for any nonsingular matrix Λ and controllable pair (A, B) [7].

The overall MRAC design goal is to achieve reasonable bounded tracking of the external time-varying command (y_{cmd}) and reference states (x_{ref}) in the presence of the nonlinear matched uncertainty $(f(x))$ and the control effectiveness term Λ. However, since the baseline controller is a fixed gain controller, unexpected changes in flight dynamics and unmodeled effects create uncertainties which degrades the performance of the controller. The role of the adaptive controller in the MRAC architecture is to reduce or ideally cancel the effect that the uncertainties have on the overall dynamics of the system and restore baseline tracking performance. That is, for any bounded time-varying command (y_{cmd}) the adaptive controller is designed to drive the selected states (x) to track the reference model states (x_{ref}) within bounds while keeping the remaining signals bounded. This goal is achieved through an incremental adaptive control architecture in the following form:

$$u = u_{BL} + u_{AD} + u_{RB} \tag{8}$$

where u_{AD} represents the adaptive control signal, u_{BL} is the optimally designed proportional-integral (PI) baseline controller, and u_{RB} is the robust term used to ensure quick convergence to a specified error region for safe switching. The subsequent sections will provide architectural details of the adaptive controller along with Lyapunov-based stability analysis results.

III. Sparse Neural Network Architecture

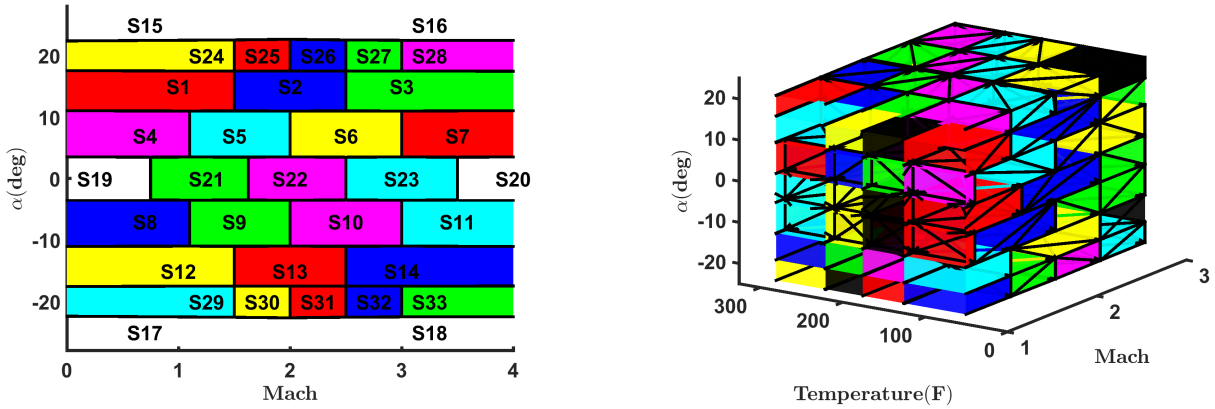
The sparse neural network (SNN) architecture is created by segmenting the flight envelope into regions where each region is assigned a certain number of neurons with associated neural network (NN) weights based on user selections. During operation within that region, the adaptive controller only utilizes and updates weights belonging to neurons that are active in that region while the remaining regions and neuron weights are frozen. The following sections aim to expound on the SNN architecture developed in [6] by providing detailed analysis and simulation results.

Notice that the traditional single hidden layer (SHL) neural network approach can be viewed as a special case of the more extensive SNN concept where the entire operating domain is considered one (passive) segment and the adaptive update laws are established using first-order Taylor series expansion of the matched uncertainty.

A. Sparse Neural Network Control Concept

In order to introduce the SNN concept, we create a pre-defined N_D-dimensional grid that spans the flight envelope. The user selects the dimensions of the grid based on known operating conditions (e.g. velocity, Mach, and altitude). Next, we define a set of segments $S = \{s_1, ..., s_T\}$ with center points $P = \{p_1, ..., p_T\}$ where $T \in \mathbb{N}$ denotes the total number of segments. We let $I = \{1, ..., T\}$ be the index set of the sets S and P. Similar to the grid, the center points are provided by the user and do not require any spacing requirements. For best results, the number of center points should be chosen to be dense in regions with significantly varying dynamics or regions which contain significant or unknown uncertainties (e.g. high angle of attack). We define the total number of nodes in the neural network as $N \in \mathbb{N}$ where every node is assigned to a specific segment. We define the number of nodes per segment as $Q = \frac{N}{T} \in \mathbb{N}$ and refer to $E_{i\in I}$ as the set of nodes allocated to segment s_i. Within each segment, the user determines the spacing of the nodes (e.g. uniform distributed) where every node is allocated to a specific position within its assigned segment. Finally, we define the set of active nodes for each segment s_i as E_{A_i}.

We use a Voronoi diagram to create the T segments using convex polygons where we denote the index of segment s_i as $i \in I$ [25]. In a Voronoi diagram, each segment is defined by a region surrounding a center point which encloses all points in the N_D space that are closer to that center point than any other center point. The Voronoi diagram also allows for an efficient and simple way to partition the flight envelope into segments while still allowing flexibility in selecting the locations of the center points. In addition, by using the nearest neighbor graph generated from Delaunay



(a) 2-D Voronoi Diagram **(b) 3-D Voronoi Diagram**

Fig. 1. Sparse neural network segmented flight envelope for hypersonic control.

triangulation, we can efficiently locate the region for which the current operating point lies within the flight space during flight time.

Consider the examples of static segmented flight envelopes in the $N_D = 2$ and $N_D = 3$ dimensional cases. Voronoi and Delaunay diagrams for a 2-D SNN flight controller utilizing angle of attack (α) and altitude as operating conditions are shown in Fig. 1a and Fig. 2a. For the 3-D case, the geometric diagrams are shown in Fig. 1b and Fig. 2b. The chosen center points for each segment are labeled for the 2-D case. For each case, the neural network will determine the set of active nodes (E_{A_i}) used in the adaptive controller based on the index (i) of the current segment (s_i) of operation. Each enclosed colored region of the Voronoi diagrams in Fig. 1 represents the domain of a single segment (s_i).

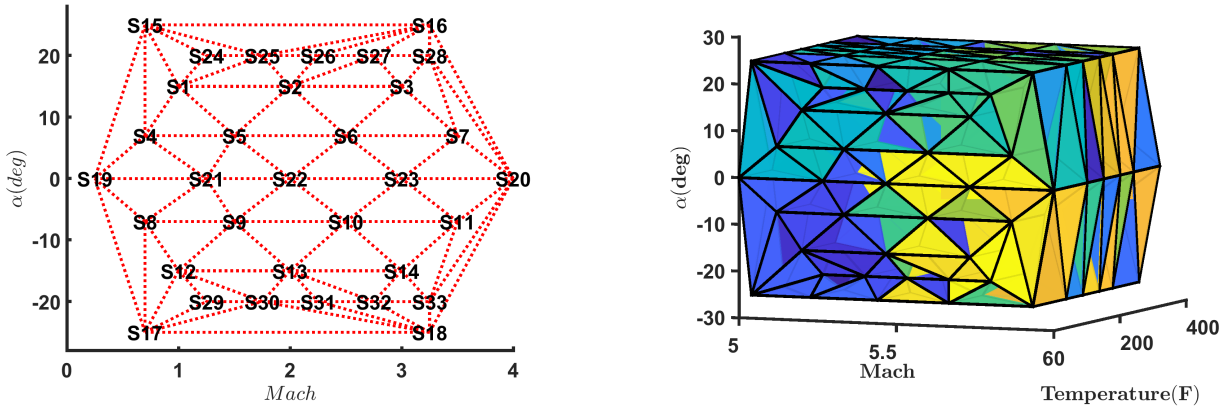
In terms of implementation, consider the feed-forward neural network diagrams in Fig. 3. Each diagram contains an arbitrary number (e.g. $N = 10$) of nodes where the nodes are denoted by $N0 - N9$. Each color indicates a different segment number. All nodes inside a colored segment belong to that segment. For instance in Fig. 3b, nodes N2 and N3 are allocated to the segment represented by the color yellow. An example of the traditional fully connected neural network architecture used for SHL and RBF adaptive control schemes is also shown in Fig. 3a where all nodes belong to the same segment. An example of the SNN architecture is shown in Fig. 3b where there are $N = 10$ nodes with $T = 5$ segments and $Q = 2$ nodes per segment.

For the sake of brevity, the reader is referred to [6] for more intimate details on the sparse neural network (SNN) adaptive control concept with mathematical notation and original implementation results. The next subsection describes the procedure for which the SNN adaptive controller executes during flight.

B. Sparse Neural Network Algorithm

At each point in time, t, the SNN determines the set of active nodes (E_{A_i}) with associated weights, ($\hat{W}_i$, $\hat{V}_i$) to be used in the adaptive controller and adaptive update laws based on the segment number, i, that the flight vehicle is currently operating. Rather than finding the index of the current segment number by brute force, we instead use the previous operating segment index and the nearest neighbor graph, generated from the Delaunay diagram, to determine the current segment number. That is, we use the nearest neighbor graph to generate a table which stores a list of neighbors for each segment. At each time the controller is called, that list is used to calculate the closest center point. We have found that this approach significantly reduces the burden on the processor.

We now define the number of active nodes, denoted $N_{act} \in \mathbb{N}$. Active nodes define the exact number of neurons that are being used for control at all times. The number of active nodes is a parameter can be varied by the user based on processing constraints. We refer to the case where N_{act} is selected to be equal to Q as the pure sparse approach, where only the set of nodes allocated to the segment (E_i) currently in operation are used in the adaptive controller. We now consider the case where the number of active nodes is $N_{act} > Q$. That is, the adaptive controller operating in segment s_i must utilize its nodes along with nodes from nearby segments for control. For this blended approach, the active node list for each segment, $E_{A_{i\in I}}$, is determined by selecting the closest N_{act} nodes to each segment's center point, p_i.



(a) 2-D Delaunay Diagram **(b) 3-D Delaunay Diagram**

Fig. 2. Delaunay diagrams for sparse neural network hypersonic control.

Notice if $N_{act} = Q = N$ then we are resorting to the traditional SHL approach. For the SNN, all the parameters are pre-defined based on user selections and the list of active nodes is created before run-time.

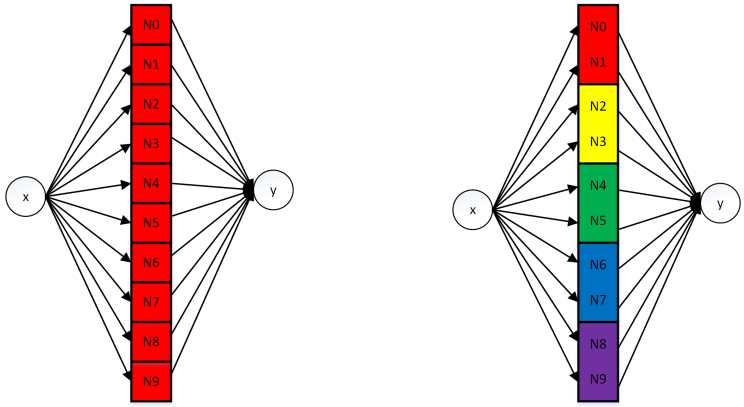
Algorithm 1 shows the step-by-step SNN approach for each time the controller is called.

IV. Adaptive Control Formulation

In order to develop a neural network approximation of the matched uncertainty, $f(x)$, our stability analysis for the sparse neural network (SNN) adaptive controller leverages the universal neural network approximation theorem [26, 27]. We consider a single hidden layer (SHL) feed-forward neural network which takes the form:

$$NN(x) = W^T \sigma(V^T x + b_V) + b_W \tag{9}$$

where $x \in \mathbb{R}^{n\times 1}$ is the input vector and $W \in \mathbb{R}^{N\times m}$, $V \in \mathbb{R}^{n\times N}$, $b_V \in \mathbb{R}^N$, and $b_W \in \mathbb{R}^m$ represent the ideal weights (W, V) and biases (b_V, b_W) of the neural network. We also define $\hat{W}$, $\hat{V}$, $\hat{b}_V$, and $\hat{b}_W$ as the estimates of the ideal weights and biases with error terms defined as $\tilde{W} = \hat{W} - W$, $\tilde{V} = \hat{V} - V$. Below is the neural network approximation theorem that will be utilized in subsequent sections.



(a) Typical Connectivity **(b) Sparse Connectivity**

Fig. 3. Neural network controller for hypersonic control with varying connectivity.

Algorithm 1 Sparse Neural Network Execution

receive $x(t)$ and the corresponding location in the operating envelope
use previous segment index (j) and Delaunay diagram to determine the index, i, of the closest operating segment s_i
if dwell time condition is met OR inside the error bounds **then**
    retrieve the set of nodes, E_{A_i}, corresponding to the index i
    form a SHL NN using the weights ($\hat{W}_i$, $\hat{V}_i$) associated with the segment i
else
    store current operating segment index (i)
    use previous segment index (j) to form a SHL NN using the weights ($\hat{W}_j$, $\hat{V}_j$) associated with the segment j
end if
use the neural network to form an adaptive control signal following Eq. (13)
create an overall control signal following Eq. (8)
apply the overall control signal to the system dynamics Eq. (1)
update the adaptive weights ($\hat{W}_i, \hat{V}_i, \hat{K}_\Lambda$) according to the adaptive update laws stated in Eqs. (19, 20, 21)

Theorem 1. *Any smooth function, $f(x)$, can be approximated over a compact domain, $x \in X$, by a single hidden layer neural network with a bounded monotonically increasing continuous activation function, σ. That is, for any $\varepsilon^* > 0$, there exist neural network weights W, V, b_V, and b_W with N neurons such that*

$$f(x) = W^T \sigma(V^T x + b_V) + b_W + \varepsilon, \quad ||\varepsilon|| < \varepsilon^\star \tag{10}$$

where ε is called the reconstruction error.

Proof. See [7] or [8].

The objective of the adaptive neural network controller is to adjust the neural network parameters (i.e. $\hat{W}$, $\hat{V}$, $\hat{b}_V$, and $\hat{b}_W$) in order to approximate a smooth nonlinear function within specified thresholds. By redefining the neural network weight matrices ($W = [W^T \; b_W]^T \in \mathbb{R}^{(N+1)\times m}$ and $V = [V^T \; b_V]^T \in \mathbb{R}^{(n+1)\times N}$) and the input vector ($\mu \in \mathbb{R}^{(n+1)\times 1}$), we can simplify notation and be more computationally efficient during run-time [6, 7]. For the stability analysis, we will assume that an ideal neural network approximation exists within a known constant tolerance while operating within a compact set, X, with known bounds. That is, we define the compact set

$$X = \{x \in \mathbb{R}^n : \; ||x|| \leq R\} \tag{11}$$

and approximation bound as

$$||\varepsilon|| \leq \bar{\varepsilon}, \quad \forall \varepsilon \in X. \tag{12}$$

For our stability analysis, the SNN controller requires a discrete update for the selection of the nodes used for control which causes switching in the closed-loop system. That is, for every point in time, t, the SNN controller is operating within a single segment, s_i, using a pre-defined set of active nodes, $E_{A_{i\in I}}$, for control. Using compact neural network notation, the form of the adaptive controller while operating in the i^{th} segment can be stated as

$$u_{NN} = -\hat{W}_i^T \sigma(\hat{V}_i^T \mu) \tag{13}$$

where $\hat{W}_i \in \mathbb{R}^{(N_{act}+1)\times m}$ and $\hat{V}_i \in \mathbb{R}^{(n+1)\times N_{act}}$ denote the estimates of the outer and inner layer weights of the ideal neural network. The total adaptive control signal is given by

$$\begin{aligned} u_{AD} &= u_{NN} + u_{K_\Lambda} && (14)\\ &= -\hat{W}_i^T \sigma(\hat{V}_i^T \mu) - \hat{K}_\Lambda(u_{BL} + u_{NN} + u_{RB}) && (15) \end{aligned}$$

where u_{K_Λ} is designed to negate the control degradation term, Λ. We assume the input vector, μ, is uniformly bounded and stated by

$$||\mu|| \leq \bar{\mu}, \qquad \bar{\mu} > 0$$

where $\bar{\mu}$ is an upper bound.

The open-loop dynamics can be derived by using the form of the baseline controller in Eq. (3) along with the chosen reference model in Eq. (5) and can be stated as

$$\dot{x} = A_{ref}x + B(u_{AD} + u_{RB}) + B\Lambda\left((I - \Lambda^{-1})u + f(x)\right) + B_{ref}y_{cmd} \tag{16}$$

where $I \in \mathbb{R}^{m\times m}$ is an identity matrix.

Recall that the definition of the state tracking error as

$$e = x - x_{ref}, \tag{17}$$

then the state tracking error dynamics can be stated as

$$\dot{e} = A_{ref}e + B(u_{AD} + f_{\Lambda}(x)) + Bu_{RB} + B\Lambda K_{\Lambda}u. \tag{18}$$

This is obtained by utilizing the form of the open-loop dynamics in Eq. (16) and the state tracking error in Eq. (17) where we define $K_{\Lambda} = (I - \Lambda^{-1})$ and $f_{\Lambda}(x) = \Lambda f(x)$.

The neural network weights are updated according to the following adaptive update laws used in the i^{th} segment:

$$\dot{\hat{W}}_i = Proj(2\Gamma_W(\sigma(\hat{V}_i^T\mu) - \dot{\sigma}(\hat{V}_i^T\mu)\hat{V}_i^T\mu)e^TPB) \tag{19}$$

$$\dot{\hat{V}}_i = Proj(2\Gamma_V\mu e^TPB\hat{W}_i^T\dot{\sigma}(\hat{V}_i^T\mu)) \tag{20}$$

where the positive definite matrix $P = P^T > 0$ satisfies the algebraic Lyapunov equation

$$PA_{ref} + A_{ref}^TP = -Q$$

for any positive definite $Q = Q^T > 0$. The estimated neural network weights ($\hat{V}_i$, $\hat{W}_i$) are determined based on the flight vehicle's location in the flight envelope [7]. The adaptive law used for canceling the effect of Λ takes the following form:

$$\dot{\hat{K}}_{\Lambda} = Proj(2\Gamma_K(u_{BL} + u_{NN} + u_{RB})e^TPB). \tag{21}$$

Note that $\Gamma_{W,}$ Γ_V, and Γ_K are diagonal matrices of positive constant learning rates.

A. Neural Network Adaptive Control Law

In this subsection, we formulate the adaptive update laws stated in Eqs. (19-20) through the use of Taylor Series expansion and determine an upper bound on the adaptive error.

Recall from Section III that T refers to the total number of segments in the sparse neural network architecture and N_{act} denotes the number of active nodes per segment which is set based on processing constraints. During operation in segment s_i, we form a SHL neural network with the following notation. Let $\hat{V}_i = [\hat{V}_i(1)\ \cdots\ \hat{V}_i(N_{act}+1)]^T$ and $\hat{V}_i(a) = [\hat{V}_i(a,1)\ \cdots\ \hat{V}_i(a,m)]^T$ where $\hat{V}_i(a,b)$ is the inner layer weight from the a^{th} hidden node to the m^{th} output for the i^{th} segment. Similarly, let $\hat{W}_i = [\hat{W}_i(1)\ \cdots\ \hat{W}_i(n+1)]^T$ and $\hat{W}_i(c) = [\hat{W}_i(c,1)\ \cdots\ \hat{W}_i(c,N_{act}+1)]^T$ where $\hat{W}_i(c,d)$ is the outer layer neural network weight from the c^{th} input node to the d^{th} hidden layer node for the i^{th} segment.

Now, consider the tracking error dynamics given by Eq. (17) and the update laws in Eq. (19) and Eq. (20). Recall, that the neural network portion of the adaptive controller is given by

$$u_{NN} = -\hat{W}_i^T\sigma(\hat{V}_i^T\mu) \tag{22}$$

while the neural network approximation error for the i^{th} segment takes the form:

$$f_{\Lambda}(x) = W_i^T\sigma(V_i^T\mu) + \varepsilon_i. \tag{23}$$

We define upper bounds $\bar{W}_i$ and $\bar{V}_i$ on the ideal neural network weights for the i^{th} interval as

$$||W_i|| < \bar{W}_i,\ ||V_i|| < \bar{V}_i \tag{24}$$

where they satisfy the following inequalities [12]:

$$||\hat{W}_i|| < ||\tilde{W}_i|| + \bar{W}_i,\ ||\hat{V}_i|| < ||\tilde{V}_i|| + \bar{V}_i. \tag{25}$$

We now compute the first order Taylor series expansion of $\sigma(V_i^T\mu)$ around $\hat{V}_i^T\mu$ which yields [8, 26]

$$\sigma(V_i^T\mu) = \sigma(\hat{V}_i^T\mu) + \dot{\sigma}(\hat{V}_i^T\mu)(V_i^T\mu - \hat{V}_i^T\mu) + O(\tilde{V}_i^T\mu)^2 \tag{26}$$

where $O(\tilde{V}_i^T x)^2$ represents the sum of the higher order terms greater than one [28]. The activation function is denoted by σ and is stored as a diagonal matrix with Jacobian $\dot{\sigma}$. For this paper, we use the sigmoid activation function and its derivatives which can be stated as

$$\sigma = \frac{1}{(1+e^{-x})} \tag{27}$$

$$\dot{\sigma} = \frac{e^x}{(e^x+1)^2} = \sigma(1-\sigma) \tag{28}$$

with bounds

$$0 \leq \sigma \leq 1 \tag{29}$$

$$0 \leq \dot{\sigma} \leq \frac{1}{4}. \tag{30}$$

Rearranging terms from Eq. (26) yields

$$O(\tilde{V}_i^T\mu)^2 = \dot{\sigma}(\hat{V}_i^T\mu)\tilde{V}_i^T\mu - (\sigma(\hat{V}_i^T\mu) - \sigma(V_i^T\mu)). \tag{31}$$

Similar to the derivation in [8, 29], this directly leads into the bounds on the adaptive error given by

$$\begin{aligned} -(u_{NN} + f_\Lambda(x)) &= \hat{W}_i^T\sigma(\hat{V}_i^T\mu) - W_i^T\sigma(V_i^T\mu) - \varepsilon_i \\ &= \tilde{W}_i^T(\sigma(\hat{V}_i^T\mu) - \dot{\sigma}(\hat{V}_i^T\mu)(\hat{V}_i^T\mu)) \\ &\quad + \hat{W}_i^T\dot{\sigma}(\hat{V}_i^T\mu)\tilde{V}_i^T\mu + h_i - \varepsilon_i. \end{aligned} \tag{32}$$

where we define h_i as

$$h_i = \tilde{W}_i^T\dot{\sigma}(\hat{V}_i^T\mu)V_i^T\mu - W_i^T O(\tilde{V}_i^T\mu)^2 \tag{33}$$

which was chosen to include terms containing unknown coefficients (e.g. W_i, V_i) and higher order terms. The terms that are linear in $\tilde{W}_i$, $\tilde{V}_i$ with known coefficients can be adapted for [26]. An upper bound can be established using the definition of h_i from Eq. (33) and the higher order terms definition in Eq. (31) [30, 29]:

$$||h_i - \varepsilon_i|| = ||\hat{W}_i^T\dot{\sigma}(\hat{V}_i^T\mu)V_i^T\mu - W_i^T\dot{\sigma}(\hat{V}_i^T\mu)\hat{V}_i^T\mu + W_i^T\sigma(\tilde{V}_i^T\mu) - \varepsilon_i|| \tag{34}$$

$$\leq \zeta_i\psi_i. \tag{35}$$

We define the Frobenius norm of a matrix X as $||X||_F = \sqrt{trace(X^TX)}$ and use the relation $||XY||_F \leq ||X||_F\,||Y||_F$ to form the following [12]:

$$\zeta_i(\bar{V}_i, \bar{W}_i) = max\{\bar{V}_i, \bar{W}_i, \bar{\varepsilon}\} \tag{36}$$

$$\psi_i(\hat{V}_i, \hat{W}_i, \mu) = ||\hat{W}_i^T\dot{\sigma}(\hat{V}_i^T\mu)||_F\,||\mu||_F + ||\dot{\sigma}(\hat{V}_i^T\mu)\hat{V}_i^T\mu||_F + 2. \tag{37}$$

Notice that $\zeta_i(\bar{V}_i, \bar{W}_i)$ is a matrix of norms for unknown coefficients and $\psi_i(\hat{V}_i, \hat{W}_i, \mu)$ is matrix of known terms. We denote an upper bound on $||h_i - \varepsilon_i||$ as $U_i \in \mathbb{R}^{m\times 1}$. The reader is referred to [29] and [7] for detailed derivations of upper bounds for neural network based adaptive controllers.

B. Robust Adaptive Control

Next, we discuss the adaptive laws stated in Eqs. (19-20) which are needed in the stability analysis. Each adaptive law includes the projection operator (see [7] and [31]) as the chosen robust adaptive control technique which forces the weight estimates to remain within a known convex bounded set (bounded). It is worth noting that we chose not to use the $e-modification$ or $\sigma-modification$ operators in our adaptive laws due to their adverse effects on adaptation

when operating with large tracking errors [7]. Details regarding the projection operator, including definitions used in this paper, is provided in Appendix A.

The projection operator ensures that a system starting with any initial weights ($\hat{\Theta}(t_0)$) within the set Ω_0 will evolve with weights ($\hat{\Theta}(t)$) within the set Ω_1 for all $t \geq t_0$. In our neural network adaptive control set-up, we can use the properties of the projection operator to define upper-bounds for the weight matrices ($\hat{W}_i$ and $\hat{V}_i$). That is, we assume that the initial conditions of $\hat{W}_i$ and $\hat{V}_i$ lie in the compact sets Ω_{W_0} and Ω_{V_0} which are defined in the same manner as Ω_0 in Eq. (113). Then, we can define the maximum values for the norm of the weight matrices $\hat{W}_i$ and $\hat{V}_i$ as [29]

$$\bar{\hat{W}}_i = max_{\hat{W}_i \in \Omega_{W_1}} ||\hat{W}_i(t)|| \qquad \forall t \geq t_0 \tag{38}$$

$$\bar{\hat{V}}_i = max_{\hat{V}_i \in \Omega_{V_1}} ||\hat{V}_i(t)||. \tag{39}$$

Similarly, the bound of the adaptive control effectiveness term can be stated as

$$\bar{\hat{K}}_\Lambda = max_{\hat{K}_\Lambda \in \Omega_{K_{\Lambda_1}}} ||\hat{K}_\Lambda(t)|| \qquad \forall t \geq t_0. \tag{40}$$

V. Robust Control with Safe Switching Analysis

Using the multiple Lyapunov function approach [15], consider the following Lyapunov candidate for each segment, s_i, as

$$\begin{aligned} V &= e^T Pe + \frac{1}{2} trace(\tilde{V}_i^T \Gamma_V^{-1} \tilde{V}_i) + \frac{1}{2} trace(\tilde{W}_i^T \Gamma_W^{-1} \tilde{W}_i) \\ &\quad + \frac{1}{2} trace((\tilde{K}_\Lambda \Lambda^{\frac{1}{2}})^T \Gamma_K^{-1} (\tilde{K}_\Lambda \Lambda^{\frac{1}{2}})) \end{aligned} \tag{41}$$

where time differentiating along the trajectories of Eq. (18) results in

$$\begin{aligned} \dot{V} &= (\dot{e}^T Pe + e^T P\dot{e}) + trace(\tilde{V}_i^T \Gamma_V^{-1} \dot{\tilde{V}}_i) + trace(\tilde{W}_i^T \Gamma_V^{-1} \dot{\tilde{W}}_i) + trace(\tilde{K}_\Lambda^T \Gamma_K^{-1} \dot{\tilde{K}}_\Lambda \Lambda) \\ &= -e^T Q_{ref} e + 2e^T PB(u_{AD} + f_\Lambda(x)) + 2e^T PB\Lambda(K_\Lambda(u_{AD} + u_{BL}) + u_{RB}) \\ &\quad + trace(\tilde{K}_\Lambda^T \Gamma_K^{-1} \dot{\tilde{K}}_\Lambda \Lambda) + trace(\tilde{V}_i^T \Gamma_V^{-1} \dot{\tilde{V}}_i) + trace(\tilde{W}_i^T \Gamma_W^{-1} \dot{\tilde{W}}_i). \end{aligned} \tag{42}$$

By substituting the previous result into Eq. (32), we have

$$\begin{aligned} \dot{V} &= -e^T Q_{ref} e + trace(\tilde{V}_i^T \Gamma_V^{-1} \dot{\tilde{V}}_i) + trace(\tilde{W}_i^T \Gamma_W^{-1} \dot{\tilde{W}}_i) + trace(\tilde{K}_\Lambda^T \Gamma_K^{-1} \dot{\tilde{K}}_\Lambda \Lambda) \\ &\quad -2e^T PB(\tilde{W}_i^T(\sigma(\hat{V}_i^T \mu) - \dot{\sigma}(\hat{V}_i^T \mu)(\hat{V}_i^T \mu)) + \hat{W}_i^T \dot{\sigma}(\hat{V}_i^T \mu)\tilde{V}_i^T \mu + h_i - \varepsilon_i) \\ &\quad -2e^T PB\Lambda(\tilde{K}_\Lambda(u_{BL} + u_{NN} + u_{RB})) + 2e^T PBu_{RB}. \end{aligned} \tag{43}$$

By using the form of the adaptive update laws stated in Eqs. (19-21), the projection operator property stated in Eq. (118), the vector property $diag(a)b = a \odot b$, and the trace property $trace(a+b) = trace(a) + trace(b)$, we can establish a simplified upper bound of Eq. (43) which is given by

$$\dot{V} \leq -e^T Q_{ref} e - 2e^T PB(h_i - \varepsilon_i) + 2e^T PBu_{RB}. \tag{44}$$

Note that the derivations and details are provided in Appendix B.

Since we know that for all $e \in \mathbb{R}^n$,

$$\lambda_{min}(Q_{ref}) ||e||^2 \leq e^T Q_{ref} e \leq \lambda_{max}(Q_{ref}) ||e||^2 \tag{45}$$

then we can rewrite Eq. (44) using the upper bounds stated in Eq. (35) as

$$\dot{V} \leq -\lambda_{min}(Q_{ref})||e||^2 + 2||e||\lambda_{max}(P)||B||(U_i) + 2e^T PBu_{RB}. \tag{46}$$

For our problem, switching in the system dynamics is introduced through the use of the sparse neural network adaptive controller. By using a robust control term and a multiple Lyapunov function approach with strict dwell

time condition, we will ensure that switching between different segments in the adaptive controller does not result in instability. Consider the open-loop dynamics stated in Eq. (1) in the form of a family of dynamic systems [14]:

$$\dot{x} = f_i(x, u) \tag{47}$$

where $i \in I$ is the segment number and f_i is locally Lipschitz.

Now consider the switched system generated from Eq. (47) [32]:

$$\dot{x} = f_S(x, u) \tag{48}$$

where a switching signal $S : \mathbb{R}^+ \to I$ specifies the index of the active segment at time t. For the remainder of this work, we will assume that the switching signal, S, is piecewise constant and right continuous.

Prevalent in switched system and hybrid literature, the average dwell time condition of a switching signal, S, can be stated as [14]

$$N_{S(T,t)} \leq N_o + \frac{T-t}{\tau_\alpha} \tag{49}$$

where the switching signal, S, has an average dwell time of τ_α if there exist two numbers $N_o \in \mathbb{N}$ and $\tau_\alpha \in \mathbb{R}^+$ that satisfy the average dwell time condition stated in Eq. (49) where $N_{S(T,t)}$ denotes the number of switches on the time interval from $[t, T)$.

For the sparse neural network controller, we are interested in calculating a strict dwell time condition for the controller to follow. The dwell time condition holds if there exists a dwell time, $T_{dwell} \in \mathbb{R}^+$, such that

$$t_{S+1} - t_S \geq T_{dwell}, \qquad \forall s \in S \tag{50}$$

where t_S denotes the time of the S^{th} switch with dwelling interval $[t_S, t_{S+1})$. It can easily be shown that the strict dwell time condition is a special case of the average dwell time condition where $N_o = 1$ [14, 15]. In the case of the average dwell time condition, some switching intervals can be less than the specified average dwell time, τ_α.

In general, the use of the robust control can lead to high gain control and limit the learning performance of adaptive controllers. However, robust control terms can be used effectively in order to ensure safe switching between intervals. This is accomplished by enabling a robust control term when the system error becomes larger than a predetermined threshold. While the error remains larger than the threshold, the robust control term remains active, and the sparse neural network is required to satisfy a dwelling time, T_{dwell}, requirement before switching to the next segment. This set-up ensures convergence to the predetermined error bound where the robust control term is deactivated and the controller performance is then determined based on the sparse adaptive neural network and the baseline controller.

Suppose u_{RB} takes the form:

$$u_{RB} = -(1 - f_{RB}) k_{RB} sgn(e^T PB) \tag{51}$$

where $k_{RB} > 0$ is selected to be the robust control gain and f_{RB} is used to fade out the effect of this control term. We define the fade out function, f_{RB}, by

$$f_{RB} = \begin{cases} 0 & ||e|| \geq r_{0_E} + \Delta_{RB} \\ 1 & ||e|| \leq r_{0_E} \end{cases} \tag{52}$$

where r_{0_E} is a design parameter used to define error bounds for the active region of the robust control term and Δ_{RB} and is selected to be the length of the fade out region.

Consider when $||e|| \geq r_{0_E} + \Delta_{RB}$ and the robust control term is active. By substituting Eq. (51) into Eq. (44), we derive the following inequality:

$$\dot{V} \leq -e^T Q_{ref} e - 2e^T PB((h_i - \varepsilon_i) + k_{RB} sgn(e^T PB)). \tag{53}$$

Using the equation [7]

$$e^T PB u_{RB} = -k_{RB} \sum_{j=1}^{m} |e^T PB|_j \tag{54}$$

and Eq. (53), where $|\cdot|$ denotes absolute value, results in the following

$$\dot{V} \leq -e^T Q_{ref} e + 2 \sum_{j=1}^{m} |e^T PB|_j (||h_i - \varepsilon_i|| - k_{RB}). \tag{55}$$

If we select the robust controller gain, k_{RB}, to satisfy the following inequality:

$$k_{RB} \geq U_i \tag{56}$$

then Eq. (55) becomes

$$\dot{V} \leq -e^T Q_{ref} e. \tag{57}$$

Consider the sphere set, $S_{r_0} \subset X$:

$$S_{r_0} = \left\{ \{e, \tilde{V}_i, \tilde{W}_i, \tilde{K}_\Lambda\} \, : \, ||e|| \leq r_{0_E} + \Delta_{RB} \right\} \tag{58}$$

where we define the radius associated with the largest Lyapunov function value inside S_{r_0} as

$$r_0 = \left(r_{0_E} + \Delta_{RB}, \bar{\tilde{V}}_i, \bar{\tilde{W}}_i, \bar{\tilde{K}}_\Lambda \right)^T \tag{59}$$

which will be referenced in the later sections.

Without loss of generality, we define the upper bounds for the adaptive error terms in the Lyapunov candidate in Eq. (41) for the i^{th} segment to be:

$$\bar{k}_{\tilde{V}_i} = max_{\hat{V}_i \in \Omega_{V_1}} \frac{1}{2} trace(\tilde{V}_i^T \Gamma_V^{-1} \tilde{V}_i) \tag{60}$$

$$\bar{k}_{\tilde{W}_i} = max_{\hat{W}_i \in \Omega_{W_1}} \frac{1}{2} trace(\tilde{W}_i^T \Gamma_W^{-1} \tilde{W}_i) \tag{61}$$

$$\bar{k}_{\tilde{K}_\Lambda} = max_{K_\Lambda \in \Omega_{K_{\Lambda_1}}} \frac{1}{2} trace((\tilde{K}_\Lambda \Lambda^{\frac{1}{2}})^T \Gamma_K^{-1} (\tilde{K}_\Lambda \Lambda^{\frac{1}{2}})) \tag{62}$$

where we denote the upper bounds of the adaptive errors as $\bar{\tilde{V}}_i$, $\bar{\tilde{W}}_i$, and $\bar{\tilde{K}}_\Lambda$.

Rewriting Eq. (55) in terms of the Lyapunov candidate in Eq. (41), we obtain:

$$\dot{V} \leq -c_V V + c_V \bar{k}_T \tag{63}$$

where $\bar{k}_T = \bar{k}_{\tilde{V}_i} + \bar{k}_{\tilde{W}_i} + \bar{k}_{\tilde{K}_\Lambda}$ and $c_V = \frac{\lambda_{min}(Q_{ref})}{\lambda_{max}(P)}$. Let t_0 and t_F be the initial time and final time while operating in a single segment, i, then Eq. (63) implies that

$$V(t) \leq \bar{k}_T + (V(t_0) - \bar{k}_T) e^{-c_V t} \tag{64}$$

for $\forall t \in [t_0, t_F]$ [33].

We now calculate an upper bound for the dwell time of our adaptive system, T_{dwell}, based on the previous equations. Recall that we use dwell time to define the minimum time for the system to wait before switching to a different segment which ensures safe switching between segments.

Theorem 2. *Suppose that there exists continuously differentiable positive definite Lyapunov candidate functions $V_{i \in I}$. Let $t = [t_0, t_F]$ represent the complete time segment for which the robust control term is active, i.e. $||e|| > r_{0_E} + \Delta_{RB}$, where t_F is the final time and t_0 is the starting time. Suppose that the complete time segment (t) can be broken into a finite number of time segments (N_S) denoted Δt_S where the subscript S denotes the time segment number of the time segment defined by $\Delta t_S = [t_{S-1}, t_S)$. Consider a switching sequence $\lambda = \{(\Delta t_1), (\Delta t_2), ...(\Delta t_{N_S})\}$ where exactly one set of nodes (i) with corresponding neural network weights ($\hat{V}_i, \hat{W}_i$) is active for each time segment in λ. If we assume the dwelling time, T_{dwell}, of the SNN adaptive controller is chosen to satisfy*

$$T_{dwell} \geq \frac{1}{c_V} ln(2) \tag{65}$$

then the system is guaranteed to enter the sphere set S_{r_0} in finite time with all closed-loop signals bounded.

Proof. Suppose the system is transitioning from segment Δt_S into segment Δt_{S+1} at time t_S. We imagine the set of active nodes for segment Δt_S is p while the set of active nodes for the segment Δt_{S+1} is c. Also let $V_p(t_S)$ and $V_c(t_S)$ denote the Lyapunov candidate values of the segments p and c, respectively, at the time instant t_S. Our goal is to show that by choosing a dwell time, T_{dwell}, that satisfies Eq. (65) then the Lyapunov candidate values will satisfy

$V_p(t_{S-1}) > V_c(t_S)$ and $V_p(t_S) > V_c(t_{S+1})$. Hence, this guarantees that the system will enter the sphere set S_{r_0} in finite time.

Consider the time instant, t_S, where the switch occurs, the tracking error $e_p(t_S)$ will initially be equivalent to $e_c(t_S)$ but the new set of (bounded) neural network weights $(\tilde{V}_c, \tilde{W}_c)$ in the adaptive controller will cause a different Lyapunov result. That is, we define the contribution of the neural network weight errors to the Lyapunov candidates for segments c and p in the form stated in Eq. (41) as

$$k_c = \frac{1}{2} trace(\tilde{V}_c^T \Gamma_V^{-1} \tilde{V}_c) + \frac{1}{2} trace(\tilde{W}_c{}^T \Gamma_V^{-1} \tilde{W}_c) \tag{66}$$

$$k_p = \frac{1}{2} trace(\tilde{V}_p^T \Gamma_V^{-1} \tilde{V}_p) + \frac{1}{2} trace(\tilde{W}_p{}^T \Gamma_V^{-1} \tilde{W}_p) \tag{67}$$

which implies the instantaneous change in the Lyapunov value is upper bounded by

$$\begin{aligned} \varepsilon_V &= V_c(t_S) - V_p(t_S) \\ &= k_c - k_p \\ &\leq \bar{k}_{NN} \leq \bar{k}_T \end{aligned} \tag{68}$$

where $\bar{k}_{NN} = \bar{k}_{\tilde{V}_i} + \bar{k}_{\tilde{W}_i}$ is defined in Eq. (63).

If we assume $||e|| > r_{0_E} + \Delta_{RB}$ while operating in the c^{th} segment during the time interval Δt_{S+1}, then Eq. (64) becomes

$$V_c(t_{S+1}) \leq \bar{k}_T + (V_c(t_S) - \bar{k}_T) e^{-c_V \Delta t_{S+1}} \tag{69}$$

By forcing the system to abide by the dwell time requirement, i.e. $\forall \Delta t_S \geq T_{dwell}$, then Eq. (69) becomes

$$V_c(t_{S+1}) \leq \bar{k}_T + (V_c(t_S) - \bar{k}_T) e^{-c_V T_{dwell}} \tag{70}$$

Using the inequality in Eq. (68), we obtain

$$V_c(t_{S+1}) \leq \bar{k}_T + (V_p(t_S) + \bar{k}_{NN} - \bar{k}_T) e^{-c_V T_{dwell}} \tag{71}$$

Since our goal is to find a T_{dwell} such that $V_c(t_{S+1}) < V_p(t_S)$, then it is sufficient to prove that

$$\bar{k}_T + (V_p(t_S) + \bar{k}_{NN} - \bar{k}_T) e^{-c_V T_{dwell}} < V_p(t_S) \tag{72}$$

which implies that

$$T_{dwell} > \frac{1}{c_V} ln \left(\frac{V_p(t_S) + \bar{k}_{NN} - \bar{k}_T}{V_p(t_S) - \bar{k}_T} \right). \tag{73}$$

By assuming that $V_p(t_S) > \bar{k}_T + \bar{k}_{NN} + \bar{k}_{\tilde{K}_\Lambda} = 2\bar{k}_T$ where $\bar{k}_T \geq \bar{k}_{NN}$, then it follows that

$$\frac{1}{c_V} ln(2) \geq \frac{1}{c_V} ln \left(\frac{\bar{k}_{NN} + \bar{k}_T}{\bar{k}_T} \right) > \frac{1}{c_V} ln \left(\frac{V_p(t_S) + \bar{k}_{NN} - \bar{k}_T}{V_p(t_S) - \bar{k}_T} \right). \tag{74}$$

Hence, if T_{dwell} is selected to satisfy Eq. (65) then Eq. (73) will also be satisfied. It is worth noting that a larger lower bound assumption on $V_p(t_S)$ would result in a smaller dwell time requirement.

Next, let us assume that $||e|| > r_{0_E} + \Delta_{RB}$ while operating in the segment p during the time interval Δt_S, then Eq. (64) becomes

$$V_p(t_S) \leq \bar{k}_T + (V_p(t_{S-1}) - \bar{k}_T) e^{-c_V \Delta t_S} \leq \bar{k}_T + (V_p(t_{S-1}) - \bar{k}_T) e^{-c_V T_{dwell}} \tag{75}$$

and after rearranging terms results in

$$V_p(t_{S-1}) \geq \bar{k}_T + (V_p(t_S) - \bar{k}_T) e^{c_V T_{dwell}}. \tag{76}$$

By plugging the lower bound derived in Eq. (72) into Eq. (76) results in the following inequality:

$$V_p(t_{S-1}) > V_p(t_S) + \bar{k}_{NN} \tag{77}$$

which implies that $V_c(t_S) < V_p(t_{S-1})$. The previous inequality also suggests that the previous assumption used to derive a dwell time requirement, i.e. $V_p(t_S) > 2\bar{k}_T$, implies $V_p(t_{S-1}) > 2\bar{k}_T + \bar{k}_{NN}$.

Notice that we are interested in finding an error bound, $\bar{e}_R$, such that the tracking error is guaranteed to enter the sphere set, S_{r_0}, in finite time, t_F, where

$$||e(t)|| = ||x(t) - x_{ref}(t)|| \leq \bar{e}_R. \tag{78}$$

By using Eq. (41), Eq. (64), and the upper bound

$$V(t) \geq e^T P e \geq \lambda_{min}(P)||e||^2 \tag{79}$$

we find the following relationship

$$||e(t)||^2 \leq \left(\frac{V(t_0) - \bar{k}_T}{\lambda_{min}(P)}\right) e^{-c_V t} + \frac{\bar{k}_T}{\lambda_{min}(P)} \tag{80}$$

for $\forall t$. By using the assumption $V(t_0) > 2\bar{k}_T + \bar{k}_{NN}$ and the dwell time requirement, T_{dwell}, from Eq. (74), results in

$$||e(t)|| \leq \sqrt{\frac{2\bar{k}_T}{\lambda_{min}(P)}} = \bar{e}_R \tag{81}$$

where $r_{0_E} + \Delta_{RB}$ must be selected to satisfy $r_{0_E} + \Delta_{RB} > \bar{e}_R$.

By the design of the SNN discussed in Section IV, it is clear that only one set of nodes is active for each time segment for which the controller is active. In practice, time segments are not designed to be equivalent in length due to the possible variations in segment sizes. Also, the varying processing speed of the on-board processors along with fast switching could result in instability. However, by using the robust control term with error threshold and enforcing the dwelling time condition of Eq. (65), then $V_c(t_{S+1}) < V_p(t_S)$ holds for $\forall S$ for which $||e|| > r_{0_E} + \Delta_{RB}$. Notice that if $||e|| \leq r_{0_E} + \Delta_{RB}$ while operating in time segment Δt_S, then the system already belongs to the sphere set S_{r_0} and the robust control term is not active. This process will continue until the end of flight time. It can be shown that all signals in the closed loop system remain uniformly bounded based on Eq. (3), Eq. (14), Eq. (41), and Eq. (63).

VI. Hypersonic Flight Vehicle Dynamics with Flexible Body Effects

In typical subsonic flight control systems the stiffness of the aircraft body, the benign operating conditions, and the natural frequency separation between the rigid body modes and the flexible body modes allow flexible body effects to be ignored in modeling due to their negligible effect during flight. Since hypersonic flight vehicles operate at extreme temperatures and high speeds, the vehicle's flexibility and state coupling can create dramatic changes in the overall flow field which causes variations in the pressure distribution on the flight vehicle [2, 23]. In addition, the flexing of the fuselage can cause unexpected control moment effects from the control surfaces. For these reasons, we investigate the performance of the SNN controller versus the traditional SHL approach on a hypersonic vehicle with flexible body effects.

We consider a highly nonlinear hypersonic flight vehicle model with four independent servo-controlled fins (δ_1, δ_2, δ_3, δ_4) oriented in an X-configuration [24]. For convenience, we created virtual fins (aileron δ_a, elevator δ_e, and rudder δ_r) used for control design in the conventional autopilot reference frame. We created a static mapping from the virtual fins to actual fin displacement in order to calculate forces and moments of the flight vehicle (see Ref. [24]). In this research we consider only longitudinal dynamics of the flight vehicle, which are assumed to be entirely decoupled from the lateral dynamics. We can write the longitudinal 3-DoF equations of motion for a hypersonic flight vehicle in

following form (see [34] and [2]):

$$\begin{aligned}
\dot{V}_T &= \frac{1}{m}(T cos(\alpha) - D) - g sin(\theta - \alpha) \\
\dot{\alpha} &= \frac{1}{mV_T}(-T sin(\alpha) - L) + q + \frac{g}{V_T} cos(\theta - \alpha) \\
\dot{\Theta} &= q \\
\dot{q} &= \frac{M}{I_{YY}} \\
\dot{h} &= V_T sin(\theta - \alpha) \\
\ddot{\eta}_i &= -2\zeta_i \omega_i \dot{\eta}_i - \omega_i^2 \eta_i + N_i, \qquad i = 1, 2, .., n_\eta
\end{aligned} \tag{82}$$

where m is the mass of the vehicle, Θ is the pitch angle, q is pitch rate, I_{YY} is the moment of inertia, and g is gravity. The equations for the ith structural mode of the flight vehicle are defined by the natural frequency (w_i), the damping ratio (ζ_i), and the generalized force (N_i). The natural frequencies of the hypersonic vehicle's body modes vary significantly based on temperature changes experiences throughout flight [4]. Hence, we will consider w_i a function of temperature, T. The forces and moments acting on the flight vehicle consist of thrust (T), drag (D), lift (L), and pitch moment (M). If we assume three elastic modes of the flight vehicle are active, the state vector, $x \in \mathbb{R}^{11}$, is given by

$$x = [V_T, \alpha, \Theta, q, h, \eta_1, \dot{\eta}_1, \eta_2, \dot{\eta}_2, \eta_3, \dot{\eta}_3] \tag{83}$$

where V_T is the true airspeed, α is angle of attack, and h is the altitude of the flight vehicle.

The axial and normal body forces (A, N) and pitching moment (M) can be approximated by (see Ref. [35, 34]):

$$A \approx \frac{1}{2}\rho V_T^2 S C_A \tag{84}$$

$$N \approx \frac{1}{2}\rho V_T^2 S C_N \tag{85}$$

$$M \approx \frac{1}{2}\rho V_T^2 S c_{ref} C_m \tag{86}$$

$$N_i \approx \frac{1}{2}\rho V_T^2 S C_{N_i} \qquad i = 1, 2, .., n_\eta \tag{87}$$

where ρ denotes air density, S is the reference area, c_{ref} is the mean aerodynamic chord, and we assume zero thrust (i.e. $T = 0$). We assume the following mapping from axial and normal (A, N) forces to lift and drag forces (L, D) used in Eq. (82):

$$L = N cos(\alpha) - A sin(\alpha) \tag{88}$$

$$D = N sin(\alpha) + A cos(\alpha). \tag{89}$$

The axial force coefficient (C_A), normal force coefficient (C_N), pitch moment coefficient (C_m), and the generalized force (N_i) appearing in Eqs. (84-86) take the following form:

$$C_A = C_{A_{ALT}}(h, Mach) + C_{A_{AB}}(\alpha, Mach) + \sum_{j=1}^{4} C_{A_{\delta_j}}(\alpha, Mach, \delta_j) \tag{90}$$

$$C_N = C_{N_0}(\alpha, Mach) + \sum_{j=1}^{4} C_{N_{\delta_j}}(\alpha, Mach, \delta_j) + C_{N_q}(\alpha, Mach, q) \tag{91}$$

$$C_m = C_{m_0}(\alpha, Mach) + \sum_{j=1}^{4} C_{m_{\delta_j}}(\alpha, Mach, \delta_j) + C_{m_q}(\alpha, Mach, q) \tag{92}$$

$$C_{N_i} = N_{i_{\alpha^2}}(\alpha^2) + N_{i_\alpha}(\alpha) + \sum_{j=1}^{4} N_{i_{\delta_j}}(\delta_j) + \sum_{k=1}^{3} N_{i_{\eta_k}}(\eta_k) \qquad i = 1, 2, .., n_\eta \tag{93}$$

where the aerodynamic coefficients can be computed using a look-up table based on the flight condition (α, M, h, q), the control inputs ($\delta_1, \delta_2, \delta_3, \delta_4$), and the flexible body states (η_1, η_2, η_3). The total lift force, drag force, and pitch

moment equations can be stated as

$$L_T = L + L_{flex} \tag{94}$$
$$D_T = D + D_{flex} \tag{95}$$
$$M_T = M + M_{flex} \tag{96}$$

where the contributions due to the flexible modes take the form:

$$L_{flex} = \frac{1}{2}\rho V_T^2 S(c_{L1}\eta_1 + c_{L2}\eta_2 + c_{L3}\eta_3) \tag{97}$$
$$D_{flex} = \frac{1}{2}\rho V_T^2 S(c_{D1}\eta_1 + c_{D2}\eta_2 + c_{D3}\eta_3) \tag{98}$$
$$M_{flex} = \frac{1}{2}\rho V_T^2 S(c_{M1}\eta_1 + c_{M2}\eta_2 + c_{M3}\eta_3) \tag{99}$$

where c_{L1}, c_{L2}, c_{L3}, c_{D1}, c_{D2}, c_{D3}, c_{M1}, c_{M2}, and c_{M3} are constants.

In order to determine control-oriented models suitable for MRAC, we select a dense set of trim points to suitably fill the potential flight envelope. The flight conditions of the trim points are selected to accurately represent the variations of the modeled dynamics of the flight vehicle throughout the flight envelope. For each flight condition for which a trim point is located, we are interested in a determining a longitudinal linear vehicle model which includes the short-period modes and the structural bending modes. In order to accomplish this task, we decouple the short period and phugoid modes of the vehicle by substituting trim values for V_T, h and Θ into Eq. (82).

Next, by representing the dynamical equations in Eq. (82) as a set of ordinary differential equations given by

$$\dot{x} = f(t,x), \quad x(t_0) = x_0, \quad t \geq t_0 \tag{100}$$

allows us to compute the linear short period plant matrices $A_p \in \mathbb{R}^{n_p \times n_p}$, $B_p \in \mathbb{R}^{n_p \times m}$, $C_p \in \mathbb{R}^{p \times n_p}$, and $D_p \in \mathbb{R}^{p \times m}$ [24]. That is, the flight model is numerically linearized with respect to the states and control inputs around each flight condition where the short period matrices take the form:

$$\begin{aligned} A_p(i,j) &= \left.\frac{\partial f(i)}{\partial x(j)}\right|_{\substack{x=x^* \\ u=u^*}}, \quad B_p(i,k) = \left.\frac{\partial f(i)}{\partial u(k)}\right|_{\substack{x=x^* \\ u=u^*}}, \\ C_p(i,j) &= \left.\frac{\partial x(i)}{\partial x(j)}\right|_{\substack{x=x^* \\ u=u^*}}, \quad D_p(i,k) = \left.\frac{\partial x(i)}{\partial u(k)}\right|_{\substack{x=x^* \\ u=u^*}}, \end{aligned} \tag{101}$$

where trim conditions are denoted by asterisks as x^* and u^*, i and j are indices of the state vector, and k is the index of the control input [24]. For control purposes, we create an additional integral error of tracking (e_I) state to include in the linear model. The resulting augmented short period linearized model takes the form:

$$\begin{bmatrix} \dot{e}_I \\ \dot{x}_p \end{bmatrix} = \begin{bmatrix} 0 & C_{reg} \\ 0 & A_p \end{bmatrix} \begin{bmatrix} e_I \\ x_p \end{bmatrix} + \begin{bmatrix} 0 \\ B_p \end{bmatrix} u + \begin{bmatrix} -1 \\ 0 \end{bmatrix} y_{cmd} \tag{102}$$

where $C_{reg} \in \mathbb{R}^{m \times n_p}$ selects the regulated state, the state vector is given by

$$x = [e_I, \alpha, q, \eta_1, \dot{\eta}_1, \eta_2, \dot{\eta}_2, \eta_3, \dot{\eta}_3] \in \mathbb{R}^9 \tag{103}$$

which includes the integral error of tracking, angle of attack, pitch rate, flexible mode positions, and flexible mode rates. We assume the flexible modal coordinates are unmodeled and not measured or used for feedback. The controller output elevator deflection ($u = \delta_e$) is produced an actuator. Notice by introducing the form of the uncertainties in the system (Λ and $f(x)$), Eq. (102) takes the general form of the MRAC problem shown in Eq. (1). For each flight condition, we then determine fixed baseline controller gains using LQ methods as discussed in Section II. For implementation, the complete baseline control signal of the nonlinear flight vehicle is determined by gain-scheduling the linear controller gains (K_I, K_P) by Mach, angle of attack (α), and altitude (h).

VII. Adaptive Control Results

In this section, we reveal the simulation results using various adaptive controllers (SHL and SNN) on the hypersonic vehicle model while tracking a simple sinusoidal command with relatively slow frequency ($f \approx 0.24 Hz$) between -3 and 3 degrees angle of attack (α) for $t_f = 250 sec$. The command requires the vehicle to spend similar amounts of time in each region of the flight envelope while repeating each sinusoidal maneuver approximately six times. We will refer to one complete (period) sinusoidal maneuver as a "pass". This simulation was designed to provide an environment for adaptive control long-term learning and tracking comparison. During testing, we assumed constant flexible mode damping terms where $\zeta_1 = \zeta_2 = \zeta_3 = 0.02$ and learning rate matrices for the adaptive controllers were set to either small (e.g. SHL-S or SNN-S) ($\Gamma_W = 5e^{-5} \times I$ and $\Gamma_V = 0.5e^{-5} \times I$) or moderate (e.g. SHL-M or SNN-M) ($\Gamma_W = 10e^{-5} \times I$ and $\Gamma_V = 1e^{-5} \times I$) where I denotes the identity matrix of appropriate size. In order to provide a proper comparison of the SHL and SNN architectures, the same number of active nodes were used in each test case ($i.e.\ N_{act} = 16$). Also, note that a constant $\Delta t = 0.01 sec$ time step with a second order integration method (AB-2) was used for the simulation.

For the sake of brevity, we selected a region surrounding a specific trim point in the flight envelope to analyze (i.e. $Mach = 6.0$, $\Theta = 0$, and $h = 14km$). In addition to varying angle of attack (α) and pitch rate (q) in the nonlinear model during simulation, we also assume that temperature varies due to angle of attack (α) (assuming constant velocity) where we use the following relationship between temperature and angle of attack (α):

$$Temp = k_{1t}\alpha^2 + k_{2t}\alpha + k_{3t} \tag{104}$$

where k_{1t}, k_{2t}, and k_{3t} are constants. We also assume a similar relationship between temperature and natural frequency (ω_i) of each flexible mode:

$$\omega_1 = k_{1a}Temp^2 + k_{1b}Temp + k_{1c} \tag{105}$$
$$\omega_2 = k_{2a}Temp^2 + k_{2b}Temp + k_{2c} \tag{106}$$
$$\omega_3 = k_{3a}Temp^2 + k_{3b}Temp + k_{3c} \tag{107}$$

where k_{1a}, k_{1b}, k_{1c}, k_{2a}, k_{2b}, k_{2c}, k_{3a}, k_{3b}, and k_{3c} are constants and the range of temperatures and mode frequencies are shown in Table 1.

The uncertainty term, $f(x)$, that was used during the simulation was created using the summation of several radial basis functions (RBFs) centered at various angles of attack (α). The magnitudes for the RBFs were set to bring the baseline controller to the brink of instability when operating without adaptive control. The spacing of the RBFs was chosen to significantly vary the uncertainty based on angle of attack (α). The uncertainty term and baseline tracking performance can be seen in Fig. 4a.

A. Single Hidden Layer (SHL) Neural Network Adaptive Control

We provide the results of the traditional SHL MRAC adaptive controller with adaptive update laws in the following form [26, 36]:

$$\dot{\hat{W}} = Proj(2\Gamma_W((\sigma(\hat{V}^T\mu) - \dot{\sigma}(\hat{V}^T\mu)\hat{V}^T\mu)e^T PB) \tag{108}$$
$$\dot{\hat{V}} = Proj(2\Gamma_V \mu e^T PB\hat{W}^T\dot{\sigma}(\hat{V}^T\mu)) \tag{109}$$

where we will refer to this controller as SHL-TS1. For this approach, the number of active nodes (N_{act}) is equal to the number of total nodes (N). The adaptive laws in the traditional case were derived using a first order Taylor expansion.

	MIN	MAX
$Temp(F)$	-100	2000
$\omega_1(rad/sec)$	15	20
$\omega_2(rad/sec)$	35	45
$\omega_3(rad/sec)$	50	60

Table 1. Range of flexible mode frequencies and temperature of HSV

As we see in the tracking plots of Fig. 5, the traditional SHL adaptive controller achieves similar performance for each sinusoidal maneuver and does not improve with repeated passes due to the lack of learning capability in the traditional SHL adaptive architecture. The results are demonstrated more clearly in the error plots of Fig. 5b, where $||e||$ is defined as the 2-norm where $e = x - x_{ref}$ and $\bar{e}$ is defined by the following equation, $\bar{e} = e^T PB$.

B. Sparse Neural Network (SNN) Adaptive Control

In this section, we show results of the SNN MRAC adaptive controller while utilizing the same values for learning rates and number of active nodes as the previous section. The SNN adaptive control laws used in this section take the form specified in Eqs. (19-20). Since the frequencies of the flexible modes explicitly depend on temperature and the flexible modes impact the moment and force equations, we will utilize a 2-D SNN architecture shown in Fig. 1 where each red rectangle signifies the border of a segment in the flight envelope and each blue X indicates a particular location of a node in the SNN architecture. We chose to divide the flight envelope into $T = 4050$ segments with $Q = 4$ nodes allocated to each segment where each segment operates using $N_{act} = 16$ active nodes. A close-up view of an arbitrary operating segment can be seen in Fig. 6a where the magenta circles denote the active nodes of the current segment.

The tracking plots of Fig. 7 show the superior transient performance of the SNN over the SHL approach due to the improved learning architecture. The controller clearly improves in performance with each repeated maneuver. In addition to the excellent tracking of the SNN controller, the SNN has the ability to retain reasonable estimates of the matched system uncertainty throughout the flight envelope. See Fig. 8 where the uncertainty estimate for the SNN was obtained after the simulation by sweeping the neural network input (e_I, α, q) across the flight envelope. In this case, we set $e_I = q = 0$ and varied angle of attack (α). It is worth noting that we chose $r_{0_E} = 1$, $\Delta_{RB} = 0.05$, and $U_i = 5$ for parameters of the robust control term. However for this analysis, the robust control term never becomes necessary due to the norm of the tracking error not eclipsing the robust control error threshold.

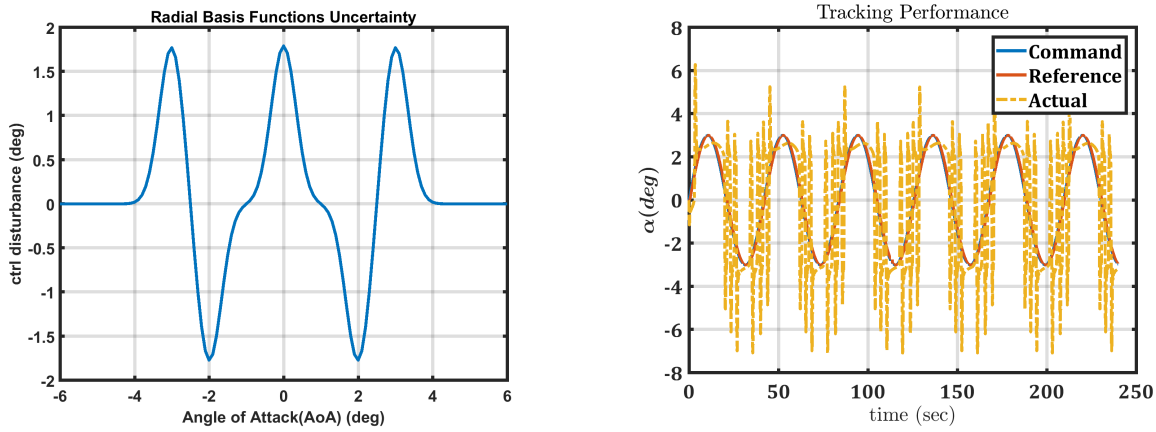
In order to properly compare the performance of the adaptive controllers with various learning rates, denoted SHL-S, SHL-M, SNN-S, and SNN-M, we created a table which quantifies the tracking performance of each controller over each pass. That is, for each complete sinusoidal maneuver (i.e. pass), we calculate the norm of the error for that time interval (TE_p) using the following equation:

$$TE_p = \sum_{t=Ts_p}^{Tf_p} ||e_p(t)||_2 \quad p = 1,..,6 \tag{110}$$

where Ts_p and Tf_p define the bounds of the time interval and e_p denotes the tracking error at time t where $e_p(t) = x(t) - x_{ref}(t)$. The results are shown in Table 2.

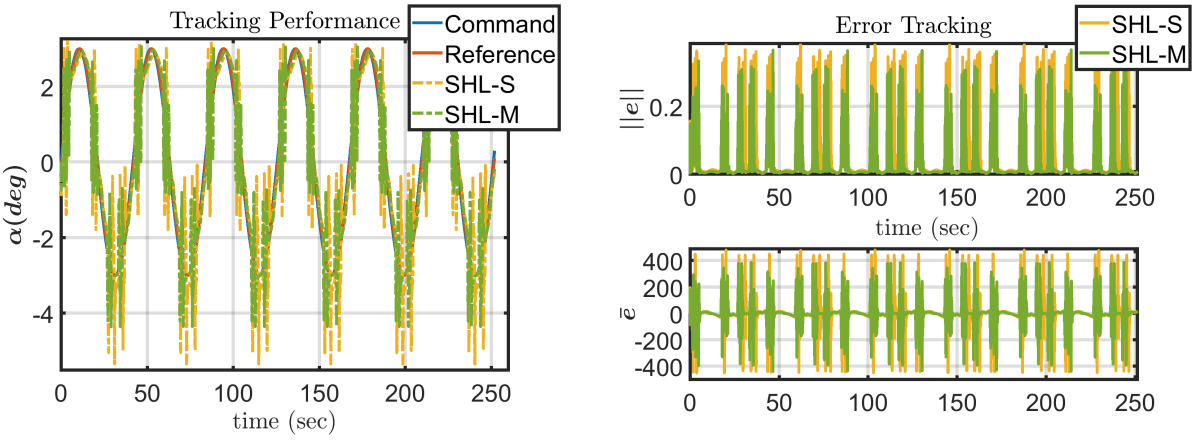
	TE_1	TE_2	TE_3	TE_4	TE_5	TE_6
SHL-S	150.67	145.64	144.94	145.65	144.52	144.43
SHL-M	147.82	141.23	141.87	141.09	141.11	141.13
SNN-S	116.22	90.84	74.27	64.91	56.01	53.35
SNN-M	101.17	68.88	51.52	46.17	43.39	41.77

Table 2. Tracking Performance Comparison



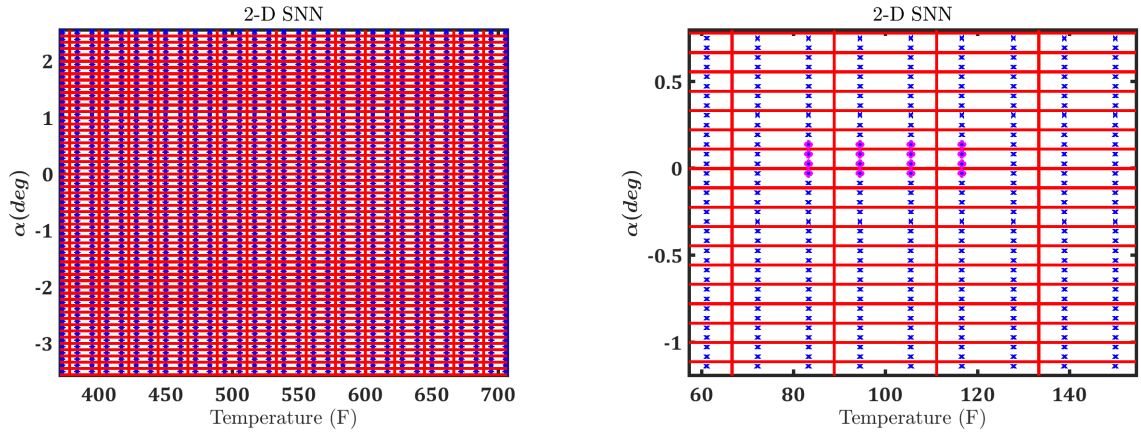
(a) RBF Uncertainty **(b) Baseline Tracking Performance**

Fig. 4. Hypersonic baseline controller under significant RBF based matched uncertainty with resulting tracking performance.



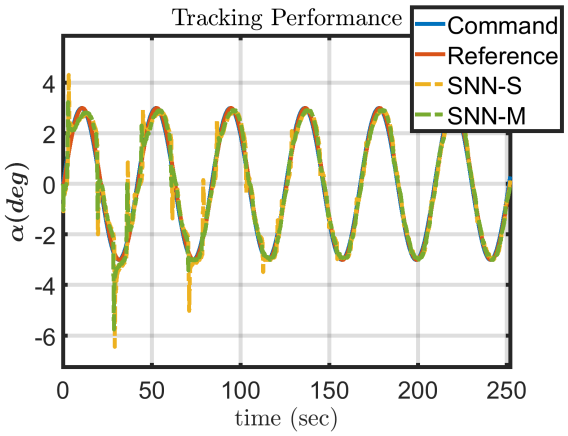
(a) SHL1 Tracking Performance **(b) SHL Error Tracking Performance**

Fig. 5. Single hidden layer (SHL) hypersonic tracking performance and error tracking with learning rate performance comparison.

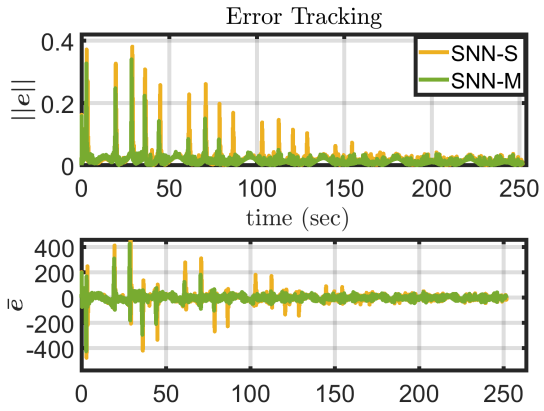


(a) Segment View **(b) Active Node View**

Fig. 6. Hypersonic two dimensional flight envelope partition in near view and bird's eye form.



(a) SNN Tracking Performance



(b) SNN Error Tracking Performance

Fig. 7. Hypersonic SNN transient performance including tracking performance and error tracking with learning rate performance comparison.

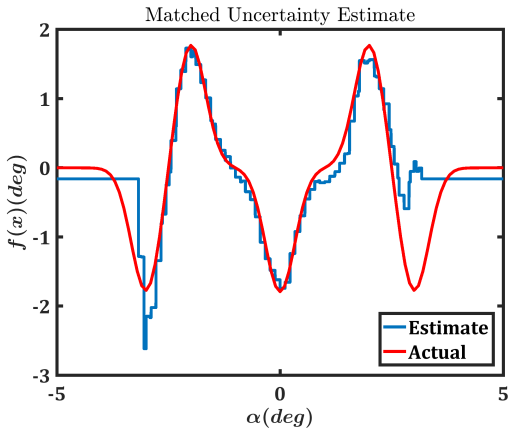


Fig. 8. Hypersonic sparse neural network (SNN) matched uncertainty estimation.

VIII. Conclusion

By using small learning rates and a relatively small number of neurons, we were able to control a sophisticated HSV model with flexible body effects using the SHL and SNN architectures. The SNN architecture provided superior performance in tracking and learning due to its sparse architecture. We derived an innovative adaptive control term which mitigated the effect of the control effectiveness matrix. In addition, we included a robust control term, which is activated outside pre-specified error thresholds, and a strict dwell-time condition to ensure safe switching between segments.

Appendix A: Projection Operator Definitions

In order to introduce the projection operator, we first define a chosen smooth convex function, $f : \mathbb{R}^N \to \mathbb{R}$, which takes the form [7]:

$$f = f(\hat{\Theta}) = \frac{(1+\varepsilon)||\hat{\Theta}||^2 - \bar{\Theta}^2}{\varepsilon\bar{\Theta}^2} \tag{111}$$

where $\varepsilon \in [0,1]$ is often referred to as the projection tolerance and $\hat{\Theta} \in \mathbb{R}^N$ is a vector of adaptive weights with an upper bound denoted by $\bar{\Theta} \in \mathbb{R}$. The projection tolerance, ε, and the upper bound on the adaptive weights, $\bar{\Theta}$, are

predefined parameters set by the user and used by the adaptive law during run-time. We define the gradient of f by

$$\nabla f = \frac{2(1+\varepsilon)}{\varepsilon\bar{\Theta}^2}\hat{\Theta} \tag{112}$$

and two convex sets (Ω_0 and Ω_1):

$$\begin{aligned}
\Omega_0 &= \{f(\hat{\Theta}) \leq 0\} && (113)\\
&= \{\hat{\Theta} \in \mathbb{R}^N : \ ||\hat{\Theta}|| \leq \frac{\bar{\Theta}}{\sqrt{1+\varepsilon}}\} && (114)\\
\Omega_1 &= \{f(\hat{\Theta}) \leq 1\} && (115)\\
&= \{\hat{\Theta} \in \mathbb{R}^N : \ ||\hat{\Theta}|| \leq \bar{\Theta}\}. && (116)
\end{aligned}$$

For use in the adaptive laws, the projection operator operates by the following equation [7]:

$$\begin{aligned}
\dot{\hat{\Theta}} &= Proj(\hat{\Theta}, \Gamma y) && (117)\\
&= \Gamma \begin{cases} y - \frac{\Gamma\nabla f(\nabla f)^T}{(\nabla f)^T\Gamma\nabla f}\Gamma y f & if\ f(\hat{\Theta}) > 0\ and\ y^T\Gamma\nabla f > 0 \\ y & otherwise \end{cases}
\end{aligned}$$

where $y \in \mathbb{R}^N$ is a known piecewise continuous vector. Then, the following useful property can be utilized in Lyapunov-based stability analysis:

$$(\tilde{\Theta})(\Gamma^{-1}Proj(\hat{\Theta}, \Gamma y) - y) \leq 0 \tag{118}$$

where we define the adaptive weight error as $\tilde{\Theta} = \hat{\Theta} - \Theta$.

Appendix B: Adaptive Law Development

In the following equations, we will show how an upper-bound of zero is developed for a number of terms in Eq. (43) by using the chosen form of the adaptive control laws stated in Eqs. (20-21). We refer to the group of terms that will be upper-bounded as V_{impact}, W_{impact}, and K_{impact}.

For instance, the effect of the adaptive law stated in Eq. (20) on $\dot{V}$ results in:

$$\begin{aligned}
V_{impact} &= -2e^TPB\hat{W}_i^T\dot{\sigma}(\hat{V}_i^T\mu)\tilde{V}_i^T\mu + trace(\tilde{V}_i^T\Gamma_V^{-1}\dot{\hat{V}}_i)\\
&= -2e^TPB\hat{W}_i^T\dot{\sigma}(\hat{V}_i^T\mu)\tilde{V}_i^T\mu + 2trace(\tilde{V}_i^T\Gamma_V^{-1}Proj(\hat{V}_i, \Gamma_V\mu e^TPB\hat{W}_i^T\dot{\sigma}(\hat{V}_i^T\mu)))\\
&= 2trace(\tilde{V}_i^T(\Gamma_V^{-1}Proj(\hat{V}_i, \Gamma_V\mu e^TPB\hat{W}_i^T\dot{\sigma}(\hat{V}_i^T\mu)) - \mu e^TPB\hat{W}_i^T\dot{\sigma}(\hat{V}_i^T\mu)))\\
&\leq 0. && (119)
\end{aligned}$$

Similarly, the impact of $\dot{\hat{W}}_i$ yields:

$$\begin{aligned}
W_{impact} &= -2e^TPB\tilde{W}_i^T(\sigma(\hat{V}_i^T\mu) - \dot{\sigma}(\hat{V}_i^T\mu)(\hat{V}_i^T\mu)) + trace(\tilde{W}_i^T\Gamma_W^{-1}\dot{\hat{W}}_i)\\
&= -2e^TPB(\tilde{W}_i^T(\sigma(\hat{V}_i^T\mu) - \dot{\sigma}(\hat{V}_i^T\mu)(\hat{V}_i^T\mu))\\
&\quad +2trace(\tilde{W}_i^T\Gamma_W^{-1}Proj(\hat{W}_i, \Gamma_W(\sigma(\hat{V}_i^T\mu) - \dot{\sigma}(\hat{V}_i^T\mu)(\hat{V}_i^T\mu))e^TPB))\\
&= 2trace(\tilde{W}_i^T\Gamma_W^{-1}Proj(\hat{W}_i, \Gamma_W(\sigma(\hat{V}_i^T\mu) - \dot{\sigma}(\hat{V}_i^T\mu)(\hat{V}_i^T\mu))e^TPB)\\
&\quad -(\sigma(\hat{V}_i^T\mu) - \dot{\sigma}(\hat{V}_i^T\mu)\hat{V}_i^T\mu)e^TPB) && (120)\\
&\leq 0.
\end{aligned}$$

Finally, $\dot{\hat{K}}_\Lambda$ in Eq. (21) forms the following upper bound:

$$\begin{aligned}
K_{impact} &= 2e^TPB(K_\Lambda - \hat{K}_\Lambda)(u_{BL} + u_{NN} + u_{RB}) + trace(\tilde{K}_\Lambda^T\Gamma_K^{-1}\dot{\hat{K}}_\Lambda\Lambda)\\
&= -2e^TPB\Lambda\tilde{K}_\Lambda(u_{BL} + u_{NN} + u_{RB}) + trace(\tilde{K}_\Lambda^T\Gamma_K^{-1}\dot{\hat{K}}_\Lambda\Lambda)\\
&= 2trace(\tilde{K}_\Lambda^T\Gamma_K^{-1}Proj(\hat{K}_\Lambda, \Gamma_K(u_{BL} + u_{NN} + u_{RB})e^TPB\Lambda)\\
&\quad -(u_{BL} + u_{NN} + u_{RB})e^TPB\Lambda)\\
&\leq 0. && (121)
\end{aligned}$$

Acknowledgments

This research was funded by the Air Force Research Laboratory and the Science, Mathematics and Research for Transformation (SMART) program. S. A. Nivison greatly appreciates the support. Approved for Public Release 96TW-2017-0260.

References

[1] Creagh, M., Kearney, M., and Beasley, P., "Adaptive Control for a Hypersonic Glider using Parameter Feedback from System Identification," *Guidance, Navigation, and Control Conference*, AIAA, 2011, p. 6230.

[2] Bolender, M. A., "An Overview on Dynamics and Controls Modelling of Hypersonic Vehicles," *American Control Conference*, 2009, pp. 2507–2512.

[3] Oppenheimer, M., and Doman, D., "A Hypersonic Vehicle Model Developed with Piston Theory," *Atmospheric Flight Mechanics Conference and Exhibit*, AIAA, 2006, p. 6637.

[4] Williams, T., Bolender, M., Doman, D., and Morataya, O., "An Aerothermal Flexible Mode Analysis of a Hypersonic Vehicle," *Atmospheric Flight Mechanics Conference*, AIAA, 2006, p. 6647.

[5] Cesnik, C., Senatore, P., Su, W., Atkins, E., Shearer, C., and Pitchter, N., "X-HALE: A Very Flexible UAV for Nonlinear Aeroelastic Tests," *AIAA Journal*, 2010, p. 2715.

[6] Nivison, S. A., and Khargonekar, P., "Improving Long-Term Learning of Model Reference Adaptive Controllers for Flight Applications: A Sparse Neural Network Approach," *Guidance, Navigation, and Control Conference*, AIAA, 2017, p. 1249.

[7] Lavretsky, E., and Wise, K., *Robust and Adaptive Control: With Aerospace Applications*, Springer Science & Business Media, 2012.

[8] Shin, Y., "Neural Network based Adaptive Control for Nonlinear Dynamic Regimes," Ph.D. thesis, 2005.

[9] Seshagiri, S., and Khalil, H. K., "Output Feedback Control of Nonlinear Systems using RBF Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 11, No. 1, 2000, pp. 69–79.

[10] Chowdhary, G., Kingravi, H. A., How, J. P., and Vela, P. A., "Bayesian Nonparametric Adaptive Control using Gaussian Processes," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 26, No. 3, 2015, pp. 537–550.

[11] Lian, J., Lee, Y., Sudhoff, S. D., and Zak, S. H., "Variable Structure Neural Network based Direct Adaptive Robust Control of Uncertain Systems," *American Control Conference*, 2008, pp. 3402–3407.

[12] Hovakimyan, N., Nardi, F., Calise, A., and Kim, N., "Adaptive Output Feedback Control of Uncertain Multi-input Multi-output systems using Single Hidden Layer Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 13, No. 6, 2002, pp. 1420–1431.

[13] Anderson, R. T., Chowdhary, G., and Johnson, E. N., "Comparison of RBF and SHL Neural Network based Adaptive Control," *Unmanned Aircraft Systems*, Springer, 2008, pp. 183–199.

[14] Hespanha, J. P., "Uniform Stability of Switched Linear Systems: Extensions of LaSalle's Invariance Principle," *IEEE Transactions on Automatic Control*, Vol. 49, No. 4, 2004, pp. 470–482.

[15] Liberzon, D., *Switching in Systems and Control*, Springer Science & Business Media, 2012.

[16] Kamalasadan, S., "A New Generation of Adaptive Control: An Intelligent Supervisory Loop Approach," Ph.D. thesis, University of Toledo, 2004.

[17] Liu, Z., and Wang, Y., "Fuzzy Adaptive Tracking Control within the Full Envelope for an Unmanned Aerial Vehicle," *Chinese Journal of Aeronautics*, Vol. 27, No. 5, 2014, pp. 1273–1287.

[18] Yu, L., and Fei, S., "Robustly Stable Switching Neural Control of Robotic Manipulators using Average Dwell-time Approach," *Transactions of the Institute of Measurement and Control*, Vol. 36, No. 6, 2014, pp. 789–796.

[19] Huang, Y., Sun, C., Qian, C., and Wang, L., "Non-fragile Switching Tracking Control for a Flexible Air-breathing Hypersonic Vehicle based on Polytopic LPV Model," *Chinese Journal of Aeronautics*, Vol. 26, No. 4, 2013, pp. 948–959.

[20] Lian, J., Hu, J., and Zak, S. H., "Adaptive Robust Control: A Piecewise Lyapunov Function Approach," *American Control Conference*, 2009, pp. 568–573.

[21] Dydek, Z. T., "Adaptive Control of Unmanned Aerial Systems," Ph.D. thesis, Massachusetts Institute of Technology, 2010.

[22] Yucelen, T., and Haddad, W. M., "Low-frequency Learning and Fast Adaptation in Model Reference Adaptive Control," *IEEE Transactions on Automatic Control*, Vol. 58, No. 4, 2013, pp. 1080–1085.

[23] Bolender, M. A., and Doman, D. B., "Nonlinear Longitudinal Dynamical Model of an Air-breathing Hypersonic Vehicle," *Journal of Spacecraft Rockets*, Vol. 44, No. 2, 2007, pp. 374–387.

[24] Dickinson, B., Nivison, S., Hart, A., Hung, C., Bialy, B., and Stockbridge, S., "Robust and Adaptive Control of a Rocket Boosted Missile," *American Control Conference*, 2015, pp. 2520–2532.

[25] Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., and Edwards, D. D., *Artificial Intelligence: A Modern Approach*, Vol. 2, Prentice Hall Upper Saddle River, 2003.

[26] Lewis, F., "Nonlinear Network Structures for Feedback Control," *Asian Journal of Control*, Vol. 1, No. 4, 1999, pp. 205–228.

[27] Lewis, F., and Ge, S. S., "Neural Networks in Feedback Control Systems," *Mechanical Engineers' Handbook: Instrumentation, Systems, Controls, and MEMS*, Vol. 2, 2006, pp. 791–825.

[28] Lewis, F. L., Yeşildirek, A., and Liu, K., "Neural Net Robot Controller: Structure and Stability Proofs," *Journal of Intelligent Robotic Systems*, Vol. 12, No. 3, 1995, pp. 277–299.

[29] Kim, N., "Improved Methods in Neural Network-based Adaptive Output Feedback Control with Applications to Flight Control," Ph.D. thesis, Georgia Institute of Technology, 2003.

[30] Zhang, T., Ge, S. S., and Hang, C. C., "Design and Performance Analysis of a Direct Adaptive Controller for Nonlinear Systems," *Automatica*, Vol. 35, No. 11, 1999, pp. 1809–1817.

[31] Petros, P., Ioannou, A., and Fidan, B., "Adaptive Control Tutorial," *Society for Industrial and Applied Mathematics*, 2006.

[32] Vu, L., Chatterjee, D., and Liberzon, D., "ISS of Switched Systems and Applications to Switching Adaptive Control," *Conference on Decision and Control*, IEEE, 2005, pp. 120–125.

[33] Polycarpou, M. M., and Mears, M. J., "Stable Adaptive Tracking of Uncertain Systems using Nonlinearly Parametrized On-line Approximators," *International Journal of Control*, Vol. 70, No. 3, 1998, pp. 363–384.

[34] Stevens, B. L., and Lewis, F. L., *Aircraft Control and Simulation*, John Wiley & Sons, 2003.

[35] Nivison, S. A., and Khargonekar, P. P., "Development of a Robust Deep Recurrent Neural Network Controller for Flight Applications," *American Control Conference*, IEEE, 2017, pp. 5336–5342.

[36] Hovakimyan, N., and Calise, A. J., "Adaptive Output Feedback Control of Uncertain Multi-input Multi-output systems using Single Hidden Layer Neural Networks," *American Control Conference*, Vol. 2, IEEE, 2002, pp. 1555–1560.