Flight Test Validation of a Neural Network based Long Term Learning Adaptive Flight Controller

Girish Chowdhary,[*] *and Eric N. Johnson*[†]

Department of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332,

The purpose of this paper is to present and analyze flight test results of a Long Term Learning Adaptive Flight Controller implemented on a rotorcraft and a fixed wing Unmanned Aerial Vehicle. The adaptive control architecture used is based on a proven Model Reference Adaptive Control (MRAC) architecture employing a Neural Network as the adaptive element. The method employed for training the Neural Network for these flight tests is unique since it uses current (online) as well as stored (background) information *concurrently* for adaptation. This ability allows the adaptive element to simulate long term memory by retaining specifically stored input output data pairs and using them for concurrent adaptation. Furthermore, the structure of the adaptive law ensures that concurrent training on past data does not affect the responsiveness of the adaptive element to current data. The results show that the concurrent use of current and background data does not affect the practical stability properties of the MRAC control architecture. The results also confirm expected improvements in performance.

Nomenclature

Frequently used symbols:		**subscripts and acronyms:**	
e	model tracking error vector	ad	Adaptive element pseudo control output
δ	actuator input	cmd	Command
Δ	model error	crm	Reference model pseudo control output
Γ_V, Γ_W	NN learning rates	pd	Linear part of the pseudo control output
ν	pseudo control		
P	Solution to the Lyapunov equation	LIP	Linear In Parameters
A	State matrix in the canonical linear system	NN	Neural Network
B	Input matrix in the canonical linear system	PCH	Pseudo Control Hedging
r	Error vector	SHL	Single Hidden Layer
σ	sigmoidal activation function	UAS	Unmanned Aerial System
V,W	NN weights	UAV	Unmanned Aerial Vehicle
x	state vector	**Learning Paradigms**	
$\bar{x}$	neural network input vector	Online	Only current data is used for learning.
		Concurrent	Both stored (background) and current data are used ***concurrently*** for learning.

I. Introduction

Adaptive control has been extensively studied for Aerospace applications. Many active research directions exist, for example Calise[6], Johnson[7,8,5,13], Kannan[22] and others have developed Model Reference Adaptive control methodology for control of Unmanned Aerial Vehicles. Lewis[11,10] et al and Patiño[29] et al have developed methods in adaptive control for the control of robotic arms. Cao, Yang, Hovaykiman, and others have developed the paradigm

[*] Graduate Research Assistant, Girish.Chowdhary@gatech.edu
[†] Associate Professor, Eric.Johnson@gatech.edu

of L1 adaptive control[33,34]. Lavertsky[30] et al have extended direct adaptive control methods to fault tolerant control. The increasing interest in adaptive control stems from its ability to handle changes in system dynamics, provable robustness to uncertainty, and the relatively direct extension to fault tolerant control.

Central to all these approaches, is an adaptive element (e.g. Neural Network) which recursively trains based on some form of input output data pairs presented in a sequential manner. The purpose of training is to develop a parameterized map that maps the given input data to the output data. Ideally, we would like this parametric map to map the complete input space to the complete output space. Where the complete input space should be considered as the feasible regions of the state space that the plant can traverse, and the complete output space should be considered as the expected range of the modeling error. This parametric map is then used in the control architecture in order to improve the performance by handling the unknown model error. Radial Basis Function (RBF) Neural Networks (NN), or Single Hidden Layer (SHL) feedforward NN are often employed as the adaptive element. The attractive Universal Approximation properties that these NN posses are the leading reason behind their choice as the adaptive element. Particularly, given sufficient number of Neurons, the SHL NN is able to approximate any piece wise continuous function (in the sense of a norm) over a compact domain. Hence, it can be expected that the NN used in the adaptive control approaches mentioned previously should be able to successfully parameterize the complete input output space. However, in practice only a local parameterization is ever achieved.

If a global parameterization of the error were indeed to be achieved, then an improvement in the performance of the adaptive control algorithm would be seen. For example, if a global parameterization of the error were to be achieved, the flight controller would have improved performance when the aircraft performs a maneuver that has been previously performed. This improvement in performance would be a result of retention of adaptation from the previous maneuver. In biological analogy, this behavior indicates long term learning in the NN. Whereas, if the adaptation were not retained, then the no improvement in performance over repeated maneuvers would be seen as the adaptive element relearns the underlying model error function every time the maneuver is performed. In biological analogy, this behavior indicates *short term learning* in the NN. Current adaptive training algorithms are only able to retain a local parameterization of the error, that is, no improvement in performance is seen when a maneuver is performed repetitively. We now discuss the reasons for this suboptimal behavior.

Consider the commonly used approach of Model Reference Adaptive Control (MRAC) with a SHL or an RBF NN used as an adaptive element. This approach has been extensively employed in the work of Lewis[11,10], Calise[6], Johson[7,8], Kannan[22], Kim[19], Hovaykiman[33], Lavertsky[30], and others for flight control purposes. In the most common form of this approach, the tracking error equations can be effectively reduced to the following form:

$$\begin{aligned}&\dot{e} = Ae + B\left[\nu_{ad}(x,\dot{x},\delta) - \Delta(x,\dot{x},\delta)\right]\\ &\textit{where},\\ &\Delta(x,\dot{x},\delta) = f(x,\dot{x},\delta) - \hat{f}(x,\dot{x},\delta).\end{aligned}$$

Where $\Delta(.)$ is the model error and between f *and* $\hat{f}$ (real and the approximate model) that needs to be cancelled by the NN output ν_{ad}. The Neural Network is often trained using a linear function of the tracking error ($r = e^T PB$) as output data and the system state vector as input data. This approach has proven to be fairly successful in designing controllers that guarantee bounded tracking error. However, since the NN output needs to cancel the model error, classical NN training techniques[26,20,21] require the output data to be the difference between the NN estimate of the model error and the actual model error ($r_c = \nu_{ad} - \Delta$). The main reason this input is not used for training NN online is because the model error Δ is unknown.

This can be contrasted with the relatively accurate parameterization over all presented data set achieved when the Neural Network is trained offline using standard methods such as batch processed error backpropagation, and the Levenberg Marquardt algorithm. The reason for this improved performance is two folds. Firstly, in most offline training approaches, the model error ($r_c = \nu_{ad} - \Delta$) is directly used for training instead of the tracking error. This allows the NN to directly adapt to the model error. Secondly, offline approaches allow flexibility in the way the input-output data pairs can be presented and weighs updated. This flexibility manifests itself in preconditioning of the training signals which may consists of smoothing, normalizing, scaling, batch processing etc.

Based on these observations, the reasons for achieving only a local parameterization when using instantaneous (sequential) data for recursively training NN in adaptive control architectures can be summarized as follows:

- **Lack of direct training information**: Direct training information ($r_c = \nu_{ad} - \Delta$) is not presented to the NN, instead a linear function of the tracking error is presented as training input ($r = e^T PB$). This information is not directly presented because the accurate model of the system is not normally available, hence it may not be possible to accurately calculate Δ online. Conceptually, NN trained in this way could be thought to provide advanced integrator control like action which cancels only the local steady state tracking error.
- **Use of only instantaneous data**: The sequential method of training is susceptible to local adaptation because weight updates occur only based on the instantaneous data.
- **The rank-1 condition**[1,2,12,7]: The rank of the NN weight dynamic is always at most one when only current data is used for NN training. This is especially true for most modern NN update laws that are based on backpropagation, which essentially approximates the underlying instantaneous error space as a gradient hyperplane. Conceptually, the rank one condition indicates that the NN law is constrained to searching NN weights only along one direction in the underlying vector space at that instant. The resulting adaptation has a non trivial null space to which important information may have been lost.

Few methods have been proposed in the past that address the issue of local adaptation. The most commonly used approach is the use of a *momentum term*[26,20,21] in the backpropagation based adaptive law. The momentum term scales the most recent weight update in the direction of the last weight update. This speeds up the convergence of the weight update when it is in the vicinity of local minima, and slows the divergence. This modification is heuristic[26,21,21], and results only in a modest improvement. Furthermore, it does not address the issue of susceptibility to local training due to high reliance on instantaneous data. Another common approach is the use of a forgetting factor[31] which can be tuned to indicate how fast the relevance of the past data needs to be decreased. This approach suffers from the drawback that a small value of the forgetting factor indicates high reliance on only the recent data, which leads to local parameterization. While a small value indicates higher reliance on the past data, which leads to sluggish adaptation performance. The sluggishness is a result of slow response to instantaneous data. Patiño[29] et al suggested the use of a bank of NN trained around different operating conditions which are used as a basis for the space of all operating conditions. The required model error is then calculated by using a linear combination of the outputs of these different NN. In order to overcome the shortcomings of online training algorithms, Patiño et al also suggested that the bank of NN be adapted off-line using recorded data. The reliance on off-line training makes this approach undesirable for adaptive flight applications. Volyanskii[35] et al have proposed a method that augments the training of the NN with an integral of the tracking error within a finite time window of the current data. They name this method as Q-modification. Q-modification allows for faster convergence of NN weights by taking the integral of the tracking error term into account, however, this integral is evaluated over a finite time interval, hence accounting only for tracking error within a relatively small time window of the instantaneous data. Furthermore, Q-modification does not allow the direct use of stored input – output data pairs in the adaptation. Hence, to the best of the author's knowledge, no current method exists that can incorporate long term learning by using arbitrary stored data along with sequential instantaneous data in order to account for the local learning phenomena exhibited by current adaptive control algorithms.

The goal of our work can now be stated:

This work is aimed towards developing a robust, long term learning, adaptive flight control architecture that guarantees uniform ultimate boundedness of all system signals.

The long term learning would be characterized by improved performance when the aircraft performs a maneuver that it has performed before. In order to achieve this goal we have developed a unique method for online training of Neural Networks that exhibits the properties of semi-global adaptation and long term learning. We term this method as Concurrent Learning. Concurrent Learning incorporates long term learning in the learning law by simulating memory. The memory of the learning law consists of carefully selected and stored input output data pairs that can be further processed in the *background* and can be used for NN training. Our method ensures that the adaptation based on stored data in the memory does not sacrifice the instantaneous adaptability of the NN by exploiting the separability of the underlying subspaces. In reference 1 we showed that uniform ultimate boundedness of all system signals can be guaranteed when this approach is used in the framework of MRAC with a SHL NN as the adaptive element. Simulation results and preliminary flight test results affirming the practical stability and indicating improved performance were also presented[1,2].

The purpose of this paper is to present further flight test results where the Concurrent learning adaptive law is used for flight control. To that effect, we present several flight test results for a rotary wing UAV. We begin by presenting a brief overview of Approximate Model Inversion NN based Adaptive Control in Section II. In section III we discuss the structure of the Concurrent learning law and provide notes on its implementation. Section IV, V, and VI are dedicated to the discussion of results which outline the benefits of using the combined online and Concurrent learning approach.

II. Neural Network Based Adaptive Control

To facilitate further discussion, a brief overview of a baseline approximate dynamic inversion based adaptive control system is given here. The reader is referred to [ref: 5-11] for further details.

A. Approximate Model Inversion based Adaptive Control

Consider a system of the form:

1
$$\ddot{x} = f(x, \dot{x}, \delta) \tag{1}$$

Where $x, \dot{x}, \delta \in \Re^n$. We introduce a pseudo control input ν which represents a desired $\ddot{x}$ and is expected to be approximately achieved by the actuating signal δ, in the following manner:

2
$$\ddot{x} = \nu \tag{2}$$

Where,

3
$$\nu = f(x, \dot{x}, \delta) \tag{3}$$

In a model inversion scheme the actual control input δ is found by inverting Eq.(3). However since the function $f(x, \dot{x}, \delta)$ is usually not exactly known or hard to invert, an approximation is introduced as:

4
$$\nu = \hat{f}(x, \dot{x}, \delta). \tag{4}$$

Based on the approximation above the actuator command is determined by an approximate dynamic inversion of the form

5
$$\delta_{cmd} = \hat{f}^{-1}(x, \dot{x}, \nu). \tag{5}$$

This results in a modeling error in the system dynamics,

6
$$\ddot{x} = \nu + \Delta(x, \dot{x}, \delta) \tag{6}$$

Where,

7
$$\Delta(x, \dot{x}, \delta) = f(x, \dot{x}, \delta) - \hat{f}(x, \dot{x}, \delta) \tag{7}$$

The approximation, $\hat{f}$ is chosen such that an inverse with respect to δ exists. Figure 1 depicts a more specific form of an approximate dynamic inversion-based Neural Network adaptive controller including actuator and PCH compensation.

Based on approximation in Eq.(4), the actuator command is determined by an approximate dynamic inversion of the form

8
$$\delta_{cmd} = \hat{f}^{-1}(x, \dot{x}, \delta) \tag{8}$$

Where ν is termed the 'pseudo-control', and represents a desired $\ddot{x}$ that is expected to be approximately achieved by δ_{cmd}. This dynamic inversion assumes perfect actuator dynamics and hence does not take into account effects such as actuator saturation, or rate limitation. As a result the actual command may not equal the achieved command due to the characteristics of the actuator (which may further vary with time). Incorporating the actuator dynamics in the actual nonlinear inversion presents other difficulties arising due to various discontinuous actuator characteristics, such as actuator saturation, discrete (quantized) control, rate limitation, time delays, and unmodelled dynamics. The Neural Network element will attempt to adapt to these characteristics even when it might not be desirable to do so. Pseudo Control Hedging (PCH)[7,8] is one method that can handle this problem. This method prevents the adaptive elements of the adaptive control system from trying to adapt to a class of unwanted plant input characteristics.

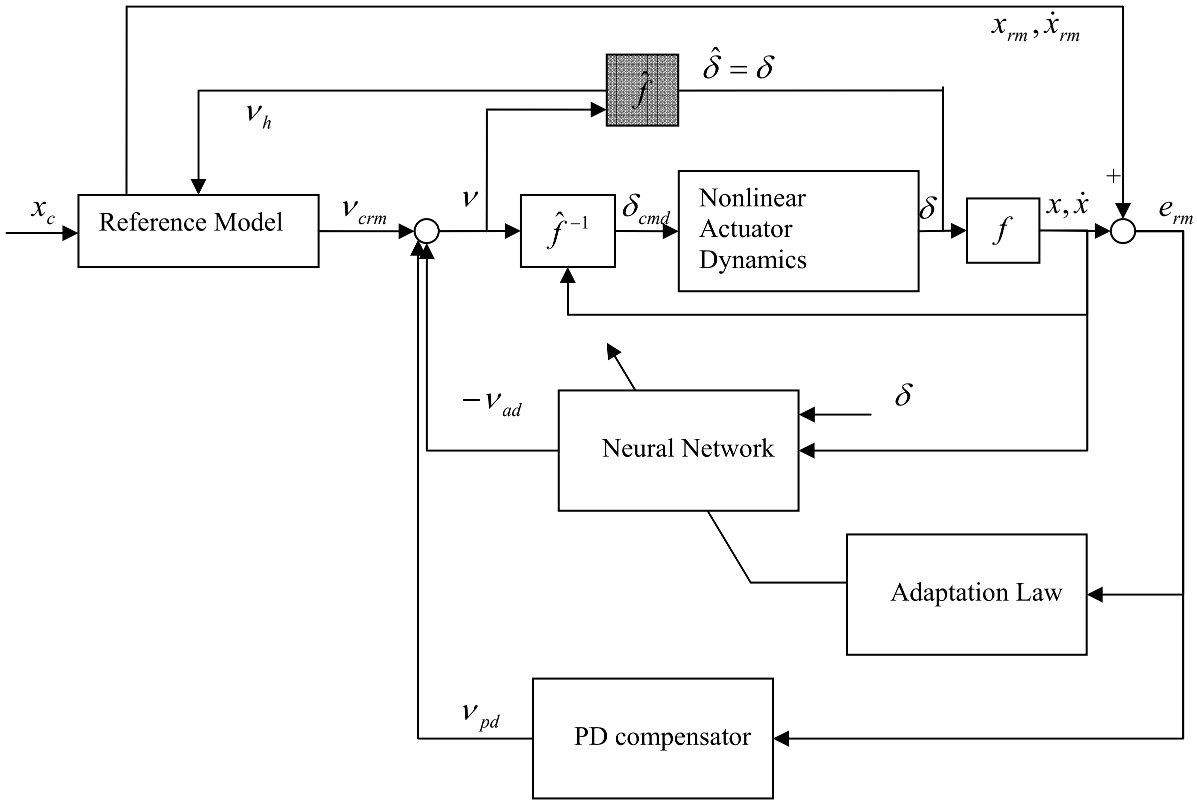

Figure 1 Neural Network Adaptive Control using Approximate Model Inversion and PCH compensation

The pseudo-control hedge signal (ν_h) is defined as the difference between the commanded pseudo-control input and the actually achieved pseudo-control input. This difference is computed by using an estimated actuator position based on a model or measurement. This estimate is then used to get the pseudo-control hedge as the difference between commanded pseudo-control and the estimated actual pseudo-control.

9
$$\nu_h = \hat{f}(x, \dot{x}, \delta_{cmd}) - \hat{f}(x, \dot{x}, \hat{\delta}) = \nu - \hat{\nu} \tag{9}$$

Figure 1 illustrates the manner in which pseudo control hedging can be achieved for a position and rate limited actuator. The PCH signal is introduced as an addition input into the reference model, forcing it to 'move back'. Hence the reference model dynamics with PCH become:

10
$$\ddot{x}_{rm} = \nu_{crm}(x_{rm}, \dot{x}_{rm}, x_c, \dot{x}_c) - \nu_h \tag{10}$$

Where $x_c, \dot{x}_c$ represent external commands. The instantaneous pseudo-control output of the reference model in the feed-forward path is not changed by the use of PCH and is ν_{crm}.

11
$$\nu_{crm} = f_{rm}(x_{rm}, \dot{x}_{rm}, x_c, \dot{x}_c) \tag{11}$$

In the PCH framework, the reference model can be thought of as a command filter. Specifically, the reference model ensures that all acceleration commands are sufficiently smooth so that the NN does not adapt to undesirable dynamics.

B. Model Tracking Error Dynamics

The total pseudo-control signal for the system is now constructed by the three components:

12
$$\nu = \nu_{crm} + \nu_{pd} - \nu_{ad} \tag{12}$$

Where ν_{crm} is the pseudo-control signal generated by the reference model in Eq. (11), ν_{pd} is the output of a linear compensator, and ν_{ad} is the Neural Network adaptation signal. The linear compensator (ν_{pd}) can be designed using standard linear control design techniques which render the closed loop system stable, these include P-D(Proportional-Derivative) compensation or LQR (Linear Quadratic Regulator) compensation. For the second order system PD compensation is expressed by

13
$$\nu_{pd} = [K_p \quad K_d]e \tag{13}$$

Where the reference model tracking error is defined as:

14
$$e = \begin{bmatrix} x_{rm} - x \\ \dot{x}_{rm} - \dot{x} \end{bmatrix} \tag{14}$$

The model tracking error dynamics are found by differentiating e:

15
$$\dot{e} = Ae + B\left[\nu_{ad}(x,\dot{x},\delta) - f(x,\dot{x},\delta) + \hat{f}(x,\dot{x},\delta)\right] \tag{15}$$

Where,

16
$$A = \begin{bmatrix} 0 & I \\ -K_p & -K_d \end{bmatrix}, B = \begin{bmatrix} 0 \\ I \end{bmatrix} \tag{16}$$

Where both K_p and K_d are real positive matrices. With the above form, A is Hurwitz. When one assumes that the plant inputs δ are exactly known then the error dynamics can be represented as:

17
$$\begin{gathered} \dot{e} = Ae + B[\nu_{ad}(x,\dot{x},\delta) - \Delta(x,\dot{x},\delta)] \\ where, \\ \Delta(x,\dot{x},\delta) = f(x,\dot{x},\delta) - \hat{f}(x,\dot{x},\delta) \end{gathered} \tag{17}$$

Is regarded as the model error to be approximated and cancelled by ν_{ad}, the output of the Neural Network. We define the signal r as:

18
$$r = e^T PB \qquad \in \Re^{n_3 \times 1} \tag{18}$$

Where $P \in \Re^{2n\times 2n}$ is the positive definite solution to the Lyapunov equation:

19
$$A^T P + PA + Q = 0 \tag{19}$$

C. Neural Network Based Adaptation

Single Hidden Layer (SHL) Perceptron NNs are universal approximators[20]. They can approximate any smooth nonlinear function to within arbitrary accuracy given sufficient number of hidden layer neurons and input information (Universal Approximation Theorem)[4,10]. The input output map of the SHL NN can be expressed in compact matrix form as:

20
$$\nu_{ad}(W,V,\bar{x}) = W^T\sigma(V^T\bar{x}) \qquad \in \Re^{n_3\times 1} \tag{20}$$

Where the following definitions are used:

21
$$\bar{x} = \begin{bmatrix} b_v \\ x_{in} \end{bmatrix} = \begin{bmatrix} b_v \\ x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_n \end{bmatrix} \qquad \in \Re^{(n_1+1)\times 1} \tag{21}$$

22
$$\sigma(z) = \begin{bmatrix} b_w \\ \sigma_1(z_1) \\ \sigma_2(z_2) \\ \cdot \\ \cdot \\ \sigma_n(z_n) \end{bmatrix} \qquad \in \Re^{(n_2+1)\times 1} \tag{22}$$

23
$$V = \begin{bmatrix} \theta_{v,1} & \cdots & \theta_{v,n2} \\ v_{1,1} & \cdots & v_{1,n2} \\ \vdots & \ddots & \vdots \\ v_{n1,1} & \cdots & v_{n1,n2} \end{bmatrix} \in \Re^{(n_1+1)\times n_2} \tag{23}$$

24
$$W = \begin{bmatrix} \theta_{w,1} & \cdots & \theta_{w,n_3} \\ w_{1,1} & \cdots & w_{1,n_3} \\ \vdots & \ddots & \vdots \\ w_{n_{2+1},1} & \cdots & w_{n_{2+1},n_3} \end{bmatrix} \in \Re^{(n_2+1)\times n_3} \tag{24}$$

Where, $\bar{x}$ is the input vector, σ is the Sigmoidal activation function vector, V is an input layer to hidden layer weight matrix, W is a hidden layer to output layer weight matrix, and ν_{ad} is the NN output. $b_v \geq 0$ and $b_w \geq 0$ are input biases that allow the thresholds θ_V and θ_W to be included in the weight matrix V and W. n_1, n_2, and n_3 represent the number of input, hidden, and output layer nodes respectively.

Input to hidden layer neuron is:

25
$$z = V^T \bar{x} = \begin{bmatrix} z_1 \\ \vdots \\ \vdots \\ z_{n2} \end{bmatrix} \in \Re^{n_2 \times 1} \tag{25}$$

The sigmoidal activation function used is:

26
$$\sigma_j(z_j) = \frac{1}{1 + e^{-a_j z_j}} \tag{26}$$

Details on Neural Network theory can be found in reference [2,10,11,20,26].

III. Description of the Concurrent Learning Training Law

SHL Neural Networks are considered to be excellent function approximators[20,21], that is, they can approximate any smooth nonlinear function within a compact set to arbitrary accuracy given enough number of hidden layer neurons and proper inputs. In the presented control setting a recursive adaptive law based NN function as the adaptive element. That is, the model uncertainties are parameterized using the NN and an adaptive law is designed which uses the available information in order to adapt to the unknown model dynamics. This allows the controller to compensate for the model uncertainties resulting in better performance. Various authors, including Johnson[7], Kannan[22], Kim[23] have analyzed the stability properties of using SHL NN for model error parameterization in approximate model inversion based adaptive flight control scheme as presented in section II. Notably, it has been shown that by choosing the appropriate values for the NN training rate parameters, the model tracking error and the NN weights (W,V) are uniformly ultimately bounded[1,5,6,7,9,22,23].

A SHL NN has the simple form given by equation 20. The reason for the choice of SHL NN follows from the desirable Universal Approximation Property of these NN. The Universal Approximation Property[10,20,21] of SHL NN ensures that given an $\bar{\varepsilon} > 0$, then for all $\bar{x} \in D$, where D is a compact set, there exist an $\bar{n}_2$ and an ideal set of weights (W^*,V^*) that brings the output of the NN to within an ε neighborhood of the function approximation error. With the largest such ε given by,

27
$$\bar{\varepsilon} = \sup_{\bar{x} \in D} \left\| W^{*^T} \sigma(V^{*^T} \bar{x}) - \Delta(\bar{x}) \right\| \tag{27}$$

The weights (W^*,V^*) may be viewed as the ideal or optimal values for (W,V), in the sense that they minimize $\bar{\varepsilon}$ on D. The universal approximation property does not guarantee the uniqueness of these values, nor does it indicate a method by which these values could be attained, it merely states that if the NN input $\bar{x}$ is chosen to represent the functional dependencies of the model error $\Delta(.)$, then by choosing n_2, $\bar{\varepsilon}$ may be made arbitrarily small.

Hence, the NN adaptive element should be able to form an arbitrarily accurate map relating the input space to the model error of equation 7, leading asymptotically reducing bounds on the error dynamics of equation 17. Furthermore, the Universal Approximation Property indicates that a single SHL NN should be sufficient to form an arbitrarily accurate parameterization of the complete input space to the output space given sufficient number of neurons. Where the complete input space should be considered as the feasible regions of the state space that the system may traverse, and the complete output space should be considered as the expected domain of the modeling error. In a practical sense, the Universal Approximation Property should thus lead to a global parameterization of the model error limited only by the input space that has been used for the training purpose. However, for reasons mentioned in the introduction, current NN training laws only achieve a local parameterization of the model error.

In reference 1 we proposed a novel NN training law which allows long term learning and semi-global adaptation in the NN. We term Neural Networks equipped with this law as *Concurrent learning NN*. The unique contribution of the Concurrent learning law is its ability to use stored data and instantaneous data concurrently for NN adaptation. In this way, the Concurrent learning law overcomes the limitations of other NN training laws that are designed for improving NN model error parameterization. Particularly, the Concurrent learning law is not susceptible to sluggish adaptation performance because of its reliance on past data as NN training laws equipped with a forgetting factor[31] are. This property of the Concurrent learning law is enforced by exploiting the separable nature of the underlying subspaces using orthogonal projection operators. Furthermore, as in reference 29, the Concurrent learning law does not require a pre-adapted bank of NN for improving global model error parameterization properties. Instead, it seeks to exploit the Universal Approximation Property by searching the parameter space for the ideal NN weights W^*, V^*. Finally, the Concurrent learning law overcomes the rank-one limitation [1,2] which allows the learning law to search for the ideal NN weights in the complete parameter space.

The underlying concept in the Concurrent learning law is to train the NN using stored data and current data *concurrently* by exploiting the seperablity[24] of the underlying subspace in order to improve global learning behavior of the NN and guarantee long term adaptation. In the architecture implemented in this paper, this is achieved by using current data as well as a stored (background) 'history stack'[12] which simulates long term memory. Both data are used concurrently in the adaptation process. Suppose P data points are stored in the history stack, then it is proposed that the total Concurrent learning be found by simply summing the individual contributions of the stored (background) data point adaptation and then by projecting the total contribution into the nullspace of the current learning. Since the learning on stored (background) data takes place in the nullspace of the learning based on instantaneous data, it does not affect the weight update based on the instantaneous data. That is, the weight update based on background data is restricted to the subspace orthogonal to the linear combinations of the weight update based on instantaneous data (i.e. the nullspace). Furthermore, since the weight update based on the instantaneous data is always at most rank-1[1,2,7,12], Concurrent Learning can occur in an n-1 dimensional subspace, where n is the dimension of the appropriate weight matrix. In this way, the separability of the underlying subspace can be exploited. Particularly, stored data can be used for concurrent adaptation without sacrificing performance of the adaptive law to adapt to new data points. The complete Concurrent learning law is simply a linear combination of the background learning law and the current learning law.

A. Choice of the Concurrent learning law

In reference 1 we proved using Lyapunov Stability Analysis that Concurrent learning NN when used in a MRAC framework guarantee the uniform ultimate boundedenss of all system signals. The training law that was derived is given as follows:

28
$$\dot{W} = -(\sigma - \sigma' V^T \bar{x}) r^T \Gamma_w - W_c \sum_{i=1}^{p} (\sigma_i - \sigma_i' V^T \bar{x}_i) r_{c_i}^T \Gamma_w \tag{28}$$

29
$$\dot{V} = -\Gamma_V \bar{x} r^T W^T \sigma'(V^T \bar{x}) - V_c \sum_{i=1}^{p} \Gamma_V \bar{x}_i r_{c_i}^T W^T \sigma'(V^T \bar{x}_i) \tag{29}$$

Where the background learning NN training signal is given by $r_{c_i} = \nu_{ad_i} - \Delta(\bar{x}_i, \delta_i)$ for every stored data point i, $\sigma, \bar{x}, r^T, \Gamma_W, \Gamma_V$ are as defined in section II. The orthogonal projection matrices $W_c, and\ V_c$ are dependent on the form of the NN training law. For the training law given in equation 28 and 29, the following projection matrices[1] can be used:

30
$$W_c = \left(I - \frac{\sigma \sigma^T}{\sigma^T \sigma} \right), \tag{30}$$

31
$$V_c = \left(I - \frac{\Gamma_V \bar{x} \bar{x}^T \Gamma_V}{\bar{x}^T \Gamma_V \Gamma_V \bar{x}} \right). \tag{31}$$

B. General form of the Concurrent learning law

The Concurrent learning law can also be expressed in a more general form as follows:

$$\dot{W} = \dot{W}_t\big(\theta, \sigma(\bar{x}), \acute{\sigma(\bar{x})}\ , r,\ \bar{x}\big) + W_c \textstyle\sum_{i=1}^{P} \dot{W}_t\big(\theta, \sigma(\bar{x}_i), \acute{\sigma(\bar{x}_\imath)}, r_{c_i},\ \bar{x}_\imath\big)\ , \tag{32}$$

$$\dot{V} = \dot{V}_t\big(\theta, \sigma(\bar{x}), \acute{\sigma(\bar{x})}, r,\ \bar{x}\big) + V_c \textstyle\sum_{i=1}^{P} \dot{V}_t\big(\theta, \sigma(\bar{x}_\imath), \acute{\sigma(\bar{x}_\imath)}, r_{c_i},\ \bar{x}_\imath\big)\ . \tag{33}$$

Where θ represents the learning parameters of the NN such as $\Gamma_V,\ and, \Gamma_W$, while $W_c, and\ V_c$ are orthogonal projection operators[24] defined as follows:

$$W_c = I - \dot{W}_t \left(\dot{W}_t{}^T \dot{W}_t\right)^+ \dot{W}_t{}^T\ , \tag{34}$$

$$V_c = I - \dot{V}_t\big(\dot{V}_t^T \dot{V}_t\big)^+ \dot{V}_t{}^T . \tag{35}$$

Where the + sign indicates the Moore Penrose pseudo inverse and I denotes an identity matrix of the appropriate dimension. In its more general form of equation 32, and equation 33, the Concurrent learning methodology allows any form of NN training laws to be used concurrently for adaptation. The orthogonality of the subspaces spanned by the range of the current and the background learning law is guaranteed[24,32] by the projection operators $W_c, and\ V_c$ of equation 34, and 35. This allows flexibility in choosing NN training algorithms that will be used to train on the background data. Since the choice of the projection matrices ensure that background learning does not affect the current learning, it is possible to use any proven NN training methods for training in the background. The augmentation of current adaptive law with proven offline NN training methods is currently an active area in our research.

C. Selection of data points for Concurrent learning

Selection of NN inputs for Concurrent learning is not a trivial problem, since these inputs impact the global learning properties of the concurrent learning approach. Detailed discussion on some methods of selecting data points can be found in [12], we suffice here by mentioning that we select data points that satisfy the following criterion:

$$\frac{\left(\bar{x} - \bar{x}_p\right)^T \left(\bar{x} - \bar{x}_p\right)}{\bar{x}^T \bar{x}} > \varepsilon_{\bar{x}} \tag{36}$$

Here the subscript p denotes the index of the last data point stored. The above method ascertains that only those data points are selected that are sufficiently different from the last data point stored. Once the data points $\bar{x}_i$ are selected, the model error (Δ) of equation 7 needs to be estimated. The benefit of using Concurrent learning becomes clear in this context. Since Concurrent learning does not affect the performance of the primary learning law, it is possible to process the data point $x_i$further to extract information about the model error dynamics. We achieve this by using an online implementation of optimal fixed point smoothing[16]. In the given framework of adaptive control the model error Δ_i for the ith data point is

$$\Delta_i(x_i, \dot{x}_i, \delta_i) = f_i(x_i, \dot{x}_i, \delta_i) - \hat{f}(x_i, \dot{x}_i, \delta_i) \tag{37}$$

Using equation 5, the above can be expressed as:

$$\Delta_i(x_i, \dot{x}_i, \delta_i) = \ddot{x}_i - \nu_i\ . \tag{38}$$

Once a point is selected for storing, the fixed point smoothing algorithm is initiated until a sufficiently accurate estimate of $\ddot{x}_i$ is obtained. Using this estimate and stored values of ν_i an estimate of the model error for the ith data point is obtained. This estimate of the model error allows us to directly use the difference between the NN estimate of the model error for the ith data point and the actual estimate of the model error for background training. In this way, the limitation on Lack of Direct Training Information mentioned in the introduction can be alleviated for Concurrent training. The residual signal that is used in the Concurrent learning adaptation is:

39
$$r_{c_i} = W^T \sigma(V^T \bar{x}_i) - \Delta_i . \tag{39}$$

The residual signal in this form for the Concurrent learning NN attempts to reduce is the difference between the current estimate of the model error and a stored estimate of the model error. Hence, by using this residual signal in a stable adaptation law, the Concurrent learning NN is made to adapt the W and V matrices of the NN in such a way that the difference between the NN adaptation and the model error for multiple data points is simultaneously reduced. It is important to note here that fixed point smoothing[16] uses a forward and a backward Kalman filter for improving the estimate of $\ddot{x}$. Hence, using fixed point smoothing for estimating the model error will entail a finite time delay before the data point can be inducted into the training law of equation 32, and 33. However, since the Concurrent learning law does not affect the behavior of the primary learning law, this delay is of no consequence.

Alternatively, the residual signal r can be formed by simulating the error dynamics[12] by integrating equation 17 for the ith data point and then using equation 18.

D. Notes on implementation

Figure 2 details the schematic of the Concurrent learning architecture. It is seen that data points are processed in a sequential manner as they are presented to the NN. Data points that are found to be of particular interest are then stored in a history stack that simulates long term memory. A fixed point iterative smoother is then used on the stored data point to estimate the model error information. The NN then trains concurrently on the recent and stored data.

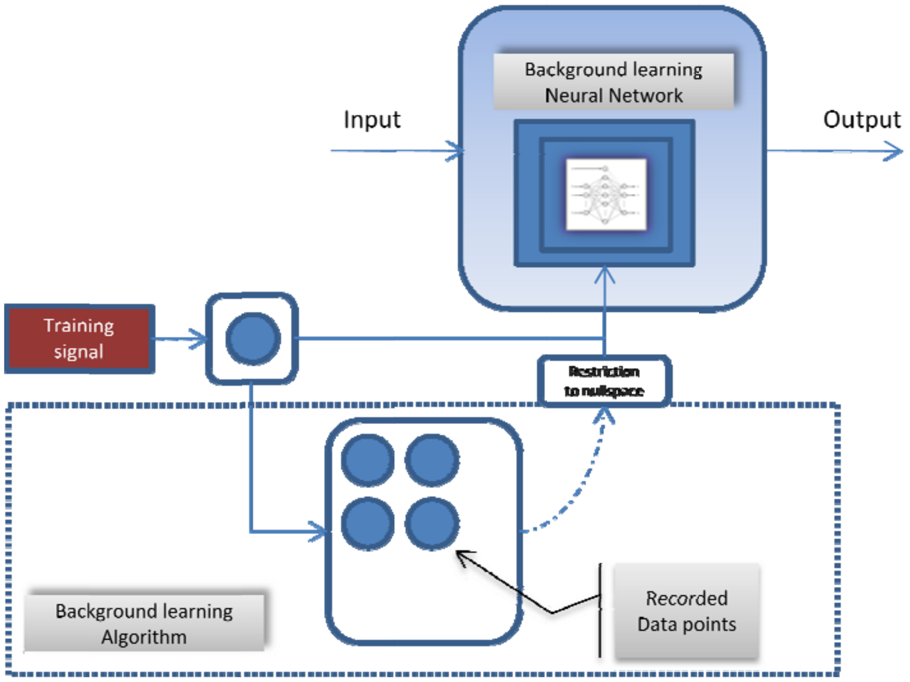

Figure 2 Schematic of the Concurrent learning NN architecture

IV. Implementation on a high fidelity flight simulator

The Georgia Tech UAV lab maintains a high fidelity Software In the Loop (SITL) flight simulator, complete with sensor emulation, detailed actuator models, disturbance simulation, and a high fidelity dynamical model. Our target platform for the methods proposed in this paper is the Georgia Tech GTMax Unmanned Aerial System (UAS). The GTMax is based on the versatile YAMAHA RMAX helicopter (Figure 3). The following results have been simulated on the GTMax SITL simulation. Since this is a higher dimensional problem, theory indicates that the impact of Concurrent learning should be more significant.

Figure 3 The Georgia Tech GTMax, in landing auto approach

The GTMax uses an approximate model inversion adaptive controller characterized equivalently to the description in section II, a detailed description can be found in reference 5 and reference 22.

We command four successive forward step inputs with arbitrary delay between any two successive steps. This type of input is used to mimic control tasks which involve commands that are repeated after an arbitrary time interval. Through these maneuvers, the UAS is expected to transition through forward flight and hover domain repeatedly. The performance of the inner loop controller is characterized by the errors in the three body angular rates (namely roll rate *p*, pitch rate *q* and yaw rate *r*). As the rotorcraft accelerates and decelerates in forward step inputs the body roll rate *q* dominates. Figure 4 shows the performance of the inner loop controller with only instantaneous adaptation in the NN. It is clearly seen that there is no considerable improvement in the roll rate error as the controller follows successive step inputs.

The forgetting nature of the controller is further characterized by the evolution of NN weights in the W and V matrices of equation 20. Figure 5 and Figure 6 clearly show that the NN weights do not converge to a constant value, in fact as the rotorcraft performs the successive step maneuvers the NN weights oscillate accordingly, clearly characterizing the instantaneous (forgetting) nature of the adaptation.

On the other hand, when concurrent learning NN learning law of equations 28 and 29 is used a clear improvement in performance is seen characterized by the reduction in pitch rate error after the first two step inputs. Figure 7 shows the performance of the Concurrent learning augmented controller. The long term adaptation nature of the Concurrent learning augmented adaptive controller is further characterized by the convergence of NN weights in the W and V matrices of equation 20. Figure 8 and Figure 9 show that when Concurrent learning is used along with instantaneous learning the NN weights do not exhibit periodic behavior and tend to converge to constant values. This indicates that the NN learns faster and retains the learning even when there is a lack of persistent excitation. This indicates that the concurrent learning controller will be able to perform better when performing a maneuver that it has previously performed, a clear indication of long term memory and semi-global learning.

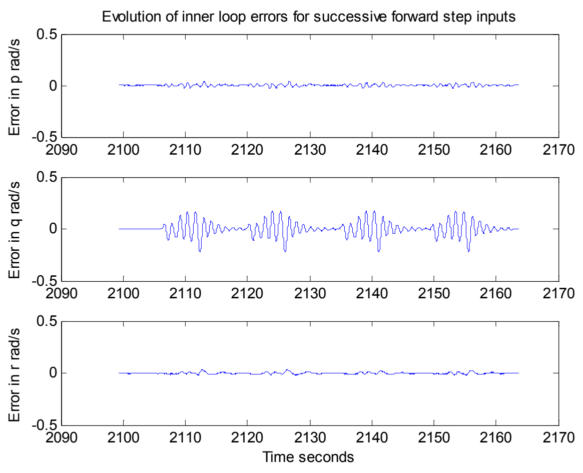

Figure 4 Evolution of inner loop errors for successive forward step inputs without Concurrent adaptation

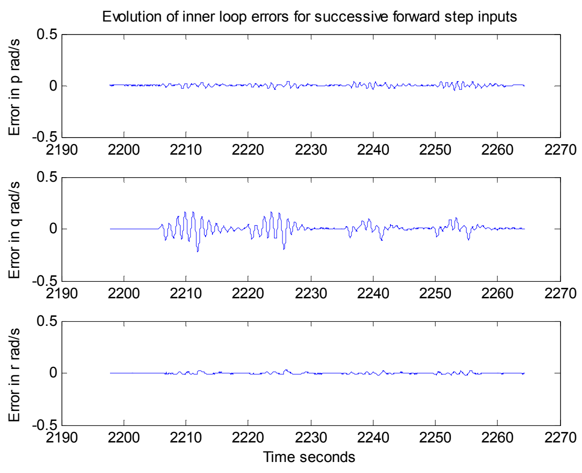

Figure 7 evolution of inner loop error with Concurrent adaptation

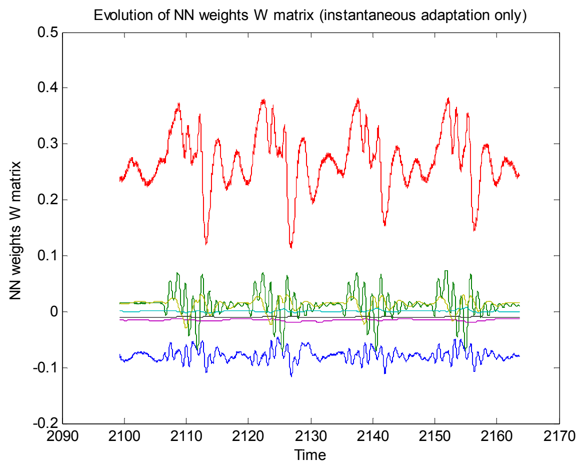

Figure 5 Evolution of NN weights, V matrix, without Concurrent adaptation

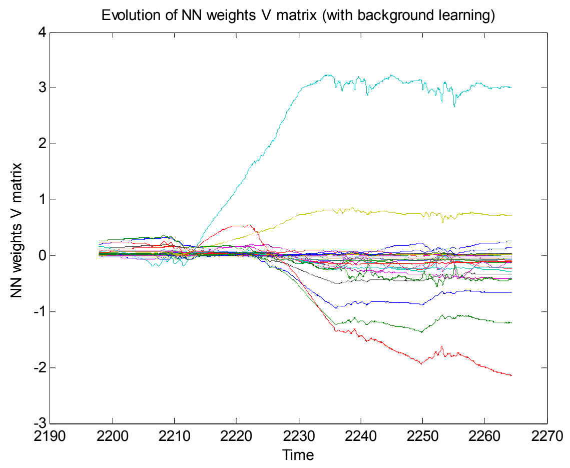

Figure 8 Evolution of NN weights, V matrix, with Concurrent adaptation

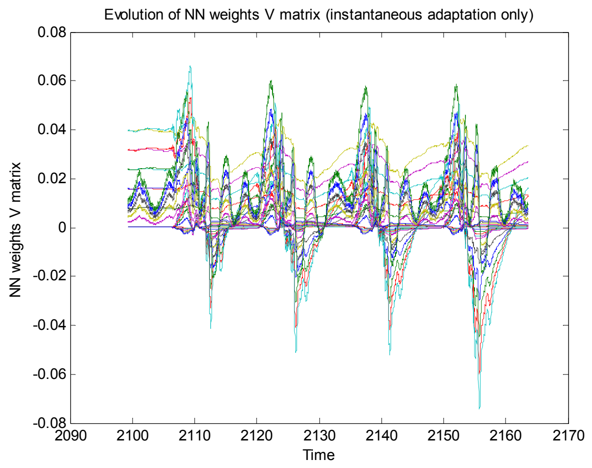

Figure 6 Evolution of NN weights, W matrix, without Concurrent adaptation

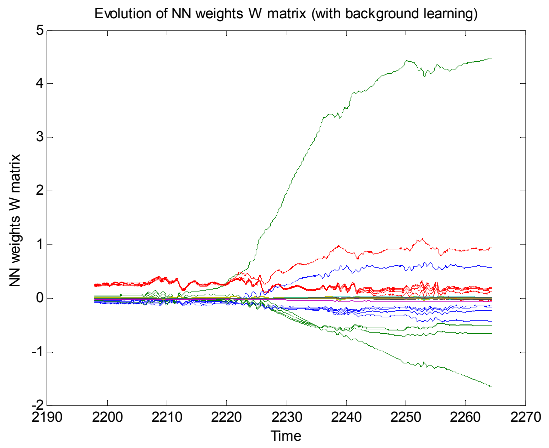

Figure 9 Evolution of NN weights, W matrix, with Concurrent adaptation

V. Implementation of Concurrent learning in flight for a Rotary Wing UAV

In this section we present some flight test results that characterize the benefits of using concurrent learning adaptive control. The flight tests presented here were executed on the Georgia Tech GTMAX rotorcraft UAV (Figure 3). We begin by presenting flight test results for a series of forward steps. This series of maneuvers serves to demonstrate explicitly the effect of Concurrent learning by showing improved weight convergence and reduction in the tracking error. We then present results from more complicated and aggressive maneuvers where it is highly desirable to have long term learning in order to improve performance. For this purpose we choose an aggressive trajectory tracking maneuver, in which the rotorcraft UAV tracks an elliptical trajectory with aggressive velocity and acceleration profile. The final maneuver chosen is an aggressive reversal of direction maneuver which first exchanges the kinetic energy of the rotorcraft for potential energy by climbing up. From the apex of its trajectory the rotorcraft falls back and reverses its direction of flight by continually aligning the heading with the local velocity vector.

A. Repeated Forward Step Maneuvers

The repeated forward step maneuvers are chosen in order to create a relatively simple situation in which the controller performs a repeated task. By using concurrent learning NN we expect to see improved performance through repeated maneuvers and a faster convergence of weights. Figure 10 shows the body frame states from recorded flight data for a chain of forward step inputs. Figure 11 shows the evolution of inner and outer loop errors. These results assert the stability (in the ultimate boundedness sense) of the concurrent learning approach.

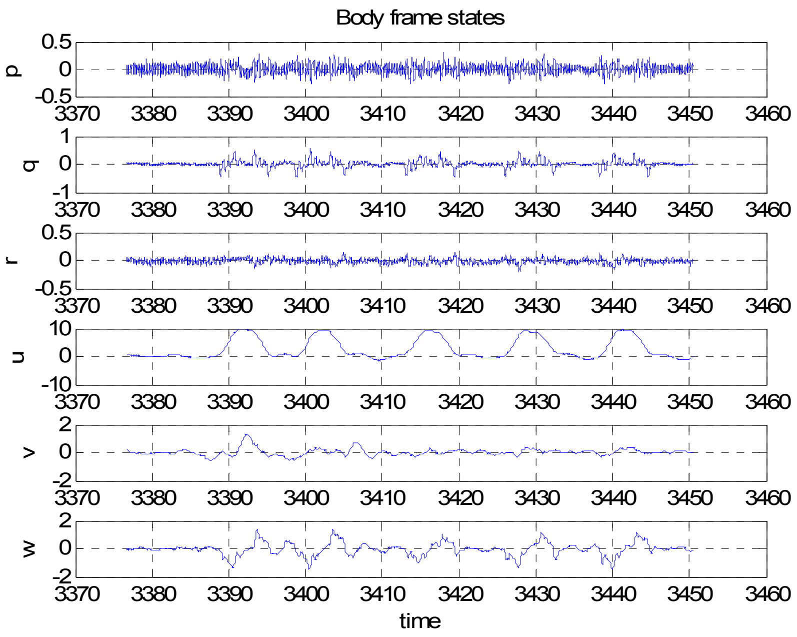

Figure 10 Body frame states

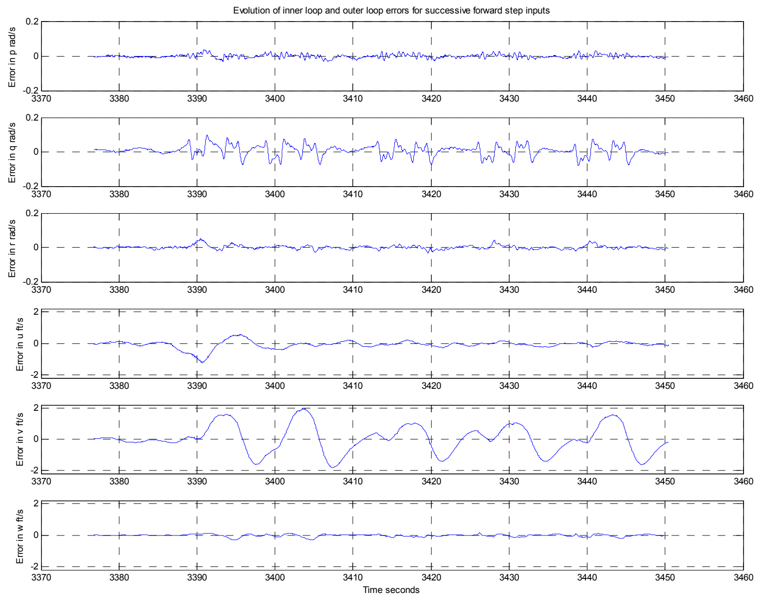

Figure 11 Evolution of inner and outer loop errors

Figure 13 and Figure 15 show the evolution of NN weights as the rotorcraft performs repeated step maneuvers and the NN is trained using concurrent learning method of Theorem 2. The NN V weights (Figure 13) appear to go to constant values when concurrent learning adaptation is used, this can be contrasted with Figure 12 which shows the V weight adaptation for a similar maneuver without Concurrent learning. NN W weights for both cases remain bounded, however it is seen that with Concurrent learning adaptation the NN W weights seem to separate, this indicates alleviation of the rank-1 condition. The flight test results indicate a modest but noticeable improvement in the error profile. In Figure 10 we see that the UAV tends not to have a smaller component of lateral velocity through each successive step. This is also seen in Figure 11 where we note that the error in v (body y axis velocity) reduces through successive steps. These effects in combination indicate that the concurrent learning system is able to improve performance over the baseline controller through repeated maneuvers, indicating long term learning. These results are of particular interest, since the maneuvers performed were conservative, and the baseline adaptive MRAC controller had already been extensively tuned.

B. Aggressive Trajectory Tracking Maneuver

Forward step maneuvers serve as a great test pattern due to their decoupled nature; however in the real world the UAV is expected to perform more complex maneuvers. In order to demonstrate the benefits of using the concurrent learning NN we present flight test results for trajectory tracking maneuver in which the UAV repeatedly tracks an elliptical trajectory with aggressive velocity (50 ft/s) and acceleration (~20 ft/s^2) profile. Since these maneuvers involve state commands in more than one system state it is harder to visually inspect the data and see whether an improvement in performance is seen. In this paper we address this issue by using the Euclidian norm of the error signal at each time step as a rudimentary metric. Further research needs to be undertaken in determining a suitable metric for this task.

Figure 16 shows the recorded inner and outer loop states as the rotorcraft repeatedly tracks an oval trajectory pattern. In this flight, the first two ovals (until t = 5415 s) are tracked with a commanded acceleration of 30ft/sec^2, while the rest of the ovals are tracked at 20ft/sec^2. In the following we treat both these parts of the flight test separately.

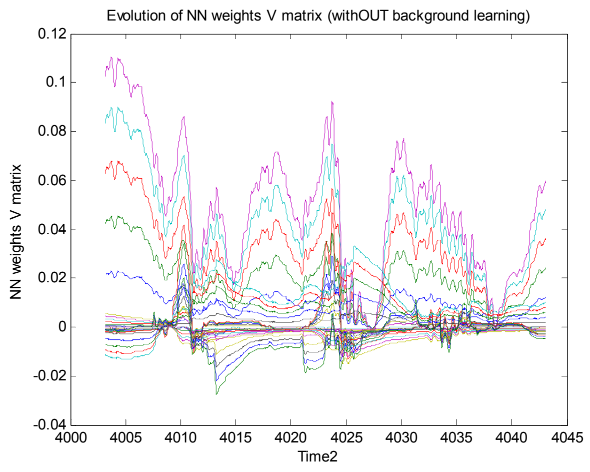

Figure 12 Evolution NN weights V matrix without concurrent learning

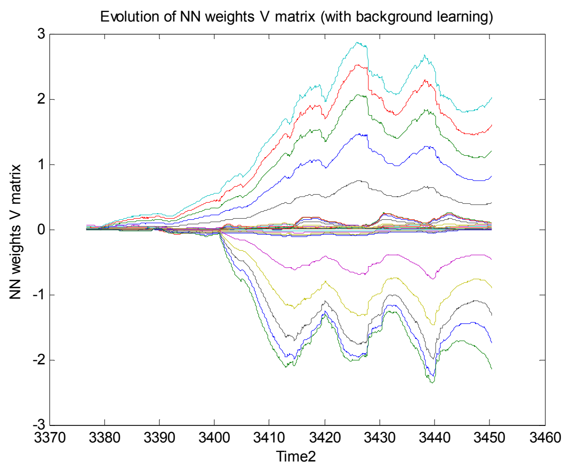

Figure 13 Evolution NN weights V matrix with concurrent learning

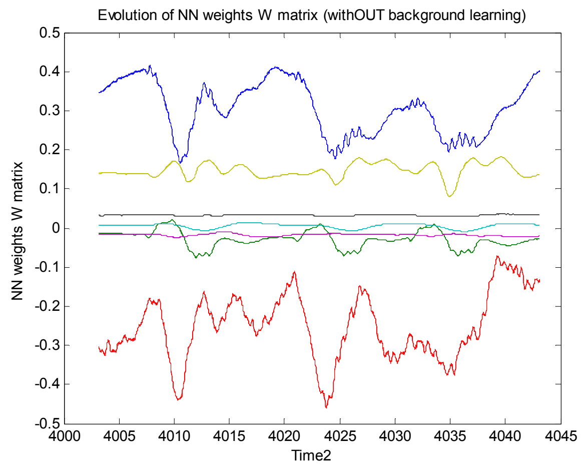

Figure 14 Evolution of NN weights W matrix without concurrent learning

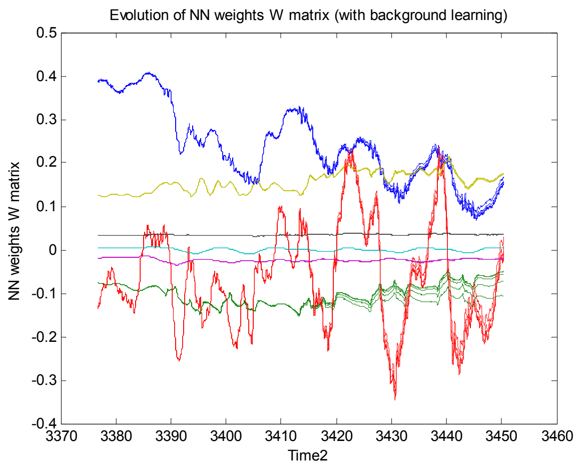

Figure 15 Evolution NN weights W matrix with concurrent learning

1. Part 1: Aggressive trajectory tracking with saturation in the collective channel

Due to the aggressive acceleration profile of 30ft/s^2 the rotorcraft collective channels were observed to saturate while performing high velocity turns. This leads to an interesting challenge for the adaptive controller. Figure 17 shows the evolution of the tracking error. It can be clearly seen that the tracking error in the u channel reduces in the second pass through the ellipse indicating long term learning by the concurrent learning adaptive control system. This result is further characterized by the noticeable reduction in the norm of the tracking error at every time step as shown in Figure 18.

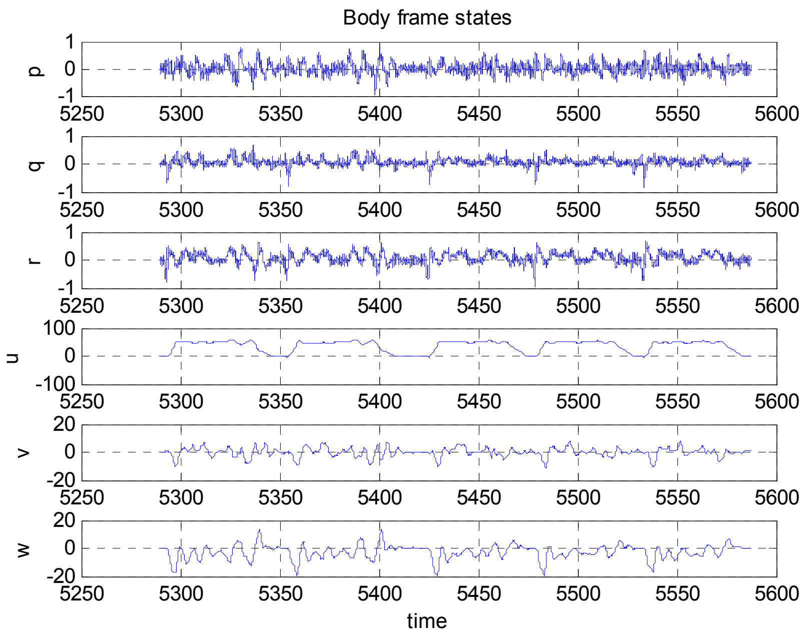

Figure 16 Recorded inner and outer loop states for repeated oval maneuvers

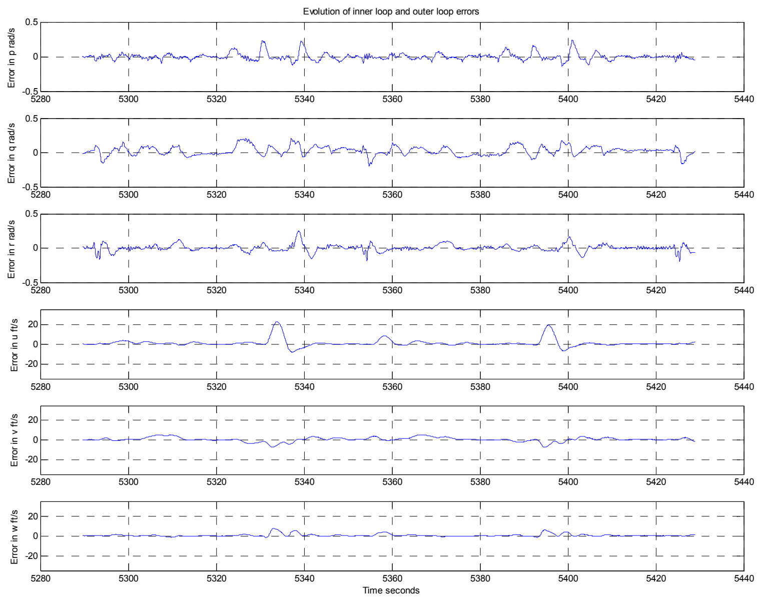

Figure 17 Tracking error profile for aggressive trajectory tracking with saturation in the collective channel

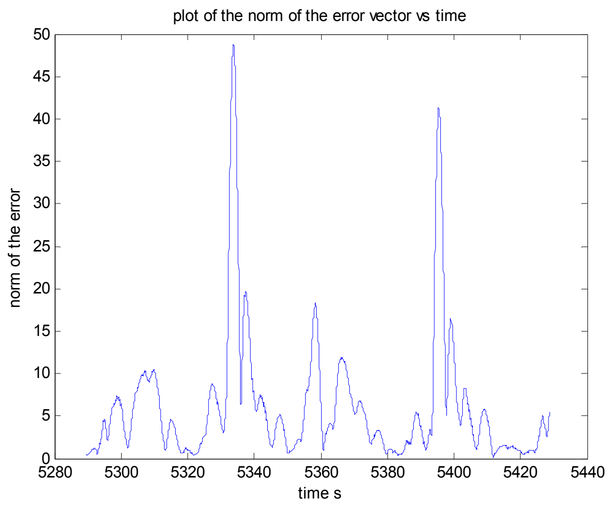

Figure 18 Plot of the norm of the error at each time step for aggressive trajectory tracking with collective saturation

2. *Part 2: Aggressive trajectory tracking maneuver*

In this part of the maneuver the acceleration profile was reduced to 20ft/sec^2. At this acceleration profile, no saturation in the collective input was noted. Figure 19 shows the evolution of tracking error, and Figure 21 shows the plot of the norm of the tracking error at each time step.

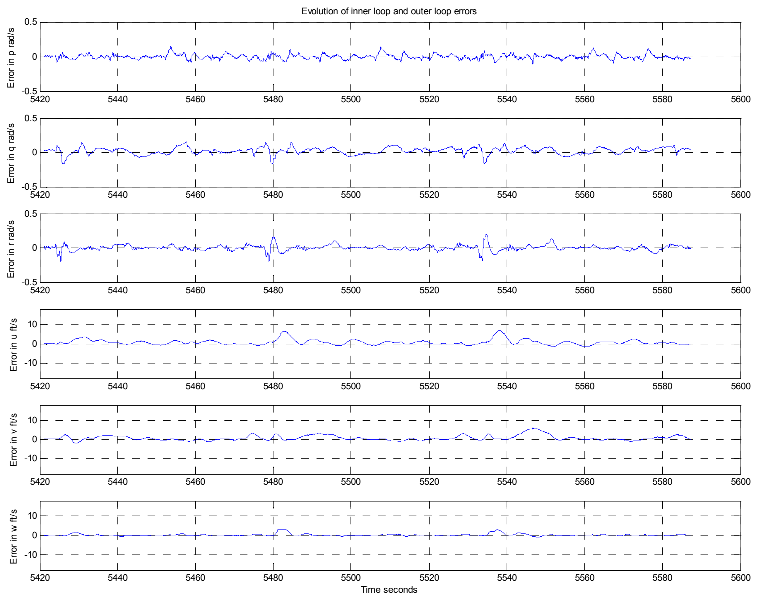

Figure 19 Evolution of tracking error for aggressive trajectory tracking maneuver

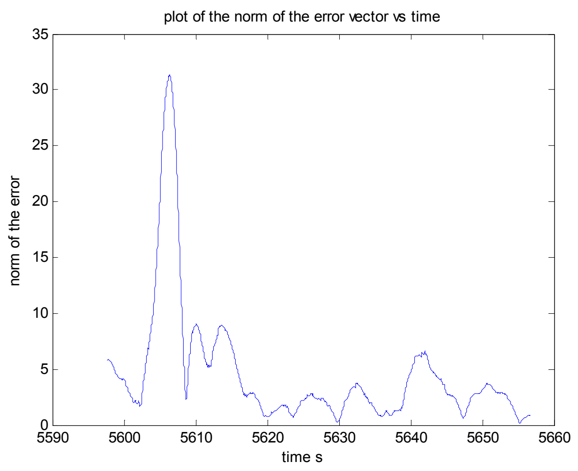

Figure 20 evolution of the norm of the tracking error vector without concurrent learning adaptation

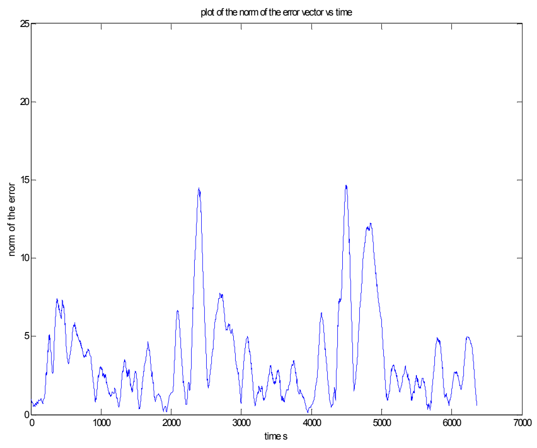

Figure 21 Evolution of the norm of the tracking error with concurrent learning

3. Aggressive trajectory maneuvers with only online learning NN

In order to illustrate the benefit of the concurrent learning adaptive controller we present flight test results as the rotorcraft tracks the same trajectory command as in *Part 2: Aggressive trajectory tracking maneuver*, but with only online learning in the NN.

It is instructive to compare Figure 22, and Figure 24 which show the evolution of the NN weights with only online learning with Figure 23, and Figure 25 which show evolution of the NN weights with concurrent learning. Although absolute convergence of weights is not seen, it is interesting to see that when concurrent learning is used, the weights tend to be less oscillatory than when only online learning is on. Also, with concurrent learning, the weights do not tend to go to zero as the rotorcraft hovers between two successive tracking maneuver. Figure 20 shows the plot of the tracking error norm as a function of time. Comparing this figure with Figure 21 it can be clearly seen that the norm of the error vector is much higher when only online learning is used. This indicates that the concurrent adaptive controller has improved trajectory tracking performance than.

C. Aggressive Reversal of Direction Maneuver

The final maneuver chosen is an aggressive reversal of direction maneuver which initially attempts to exchange kinetic energy of the rotorcraft for potential energy by climbing up. From the apex of its trajectory the rotorcraft falls back and reverses its direction of flight by continually aligning the heading with the local velocity vector. Figure 28 shows the recorded inner and outerloop states as the rotorcraft performs repeated aggressive reversal of direction maneuvers. The aggressiveness of this maneuver is clearly indicated by the large angular rates that the rotorcraft achieves. Figure 29 shows the evolution of the tracking error as the rotorcraft performs the aggressive direction reversal maneuver with concurrent learning adaptive controller. It can be seen that the concurrent control results in the ultimate boundedness of all error signals. Furthermore, when comparing Figure 27 and Figure 26 which show the norm of the error vector as a function of time when the same maneuver is performed (twice in Figure 27), it can be seen that the peaks of the norm of the error when concurrent learning is used are smaller in magnitude than when only online learning is used. Also, in Figure 27 we can see that the profile of the metric marginally improves the second time the UAV performs the maneuver.

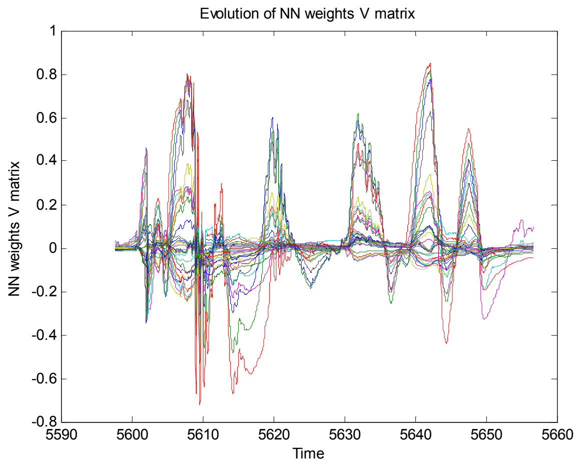

Figure 22 Evolution of NN V matrix weights without concurrent learning adaptation

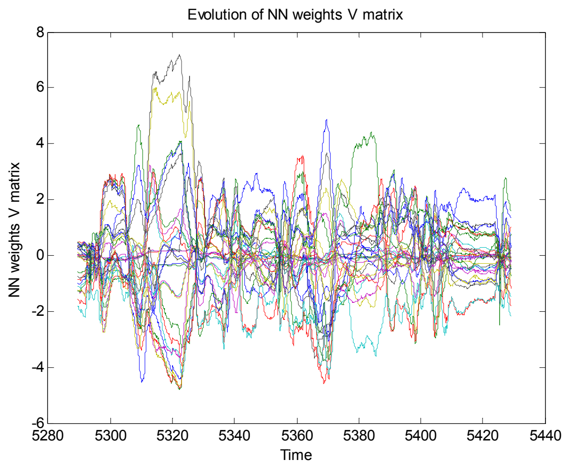

Figure 23 Evolution of the NN V matrix weights as the UAV tracks aggressive trajectory with concurrent learning adaptation

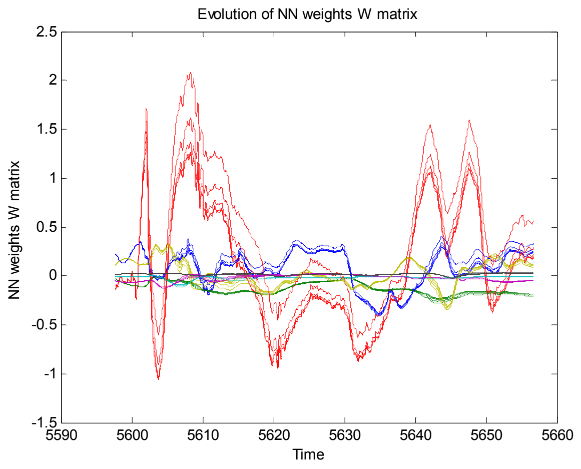

Figure 24 Evolution of NN W matrix weights without concurrent learning adaptation

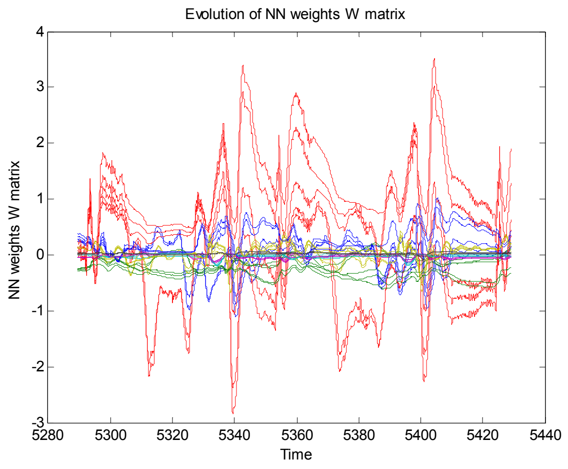

Figure 25 Evolution of the NN W matrix weights as the UAV tracks aggressive trajectory with concurrent learning adaptation.

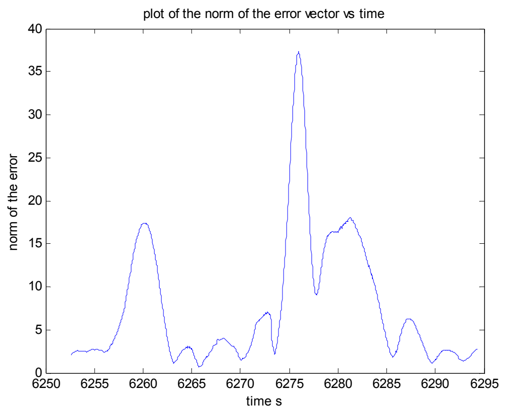

Figure 26 Evolution of the norm of the tracking error vector without concurrent learning

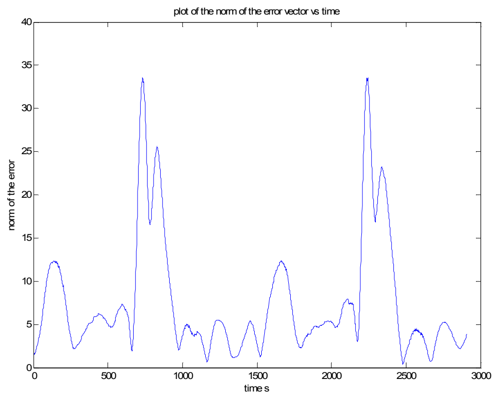

Figure 27 Evolution of the norm of the tracking error vector with concurrent learning

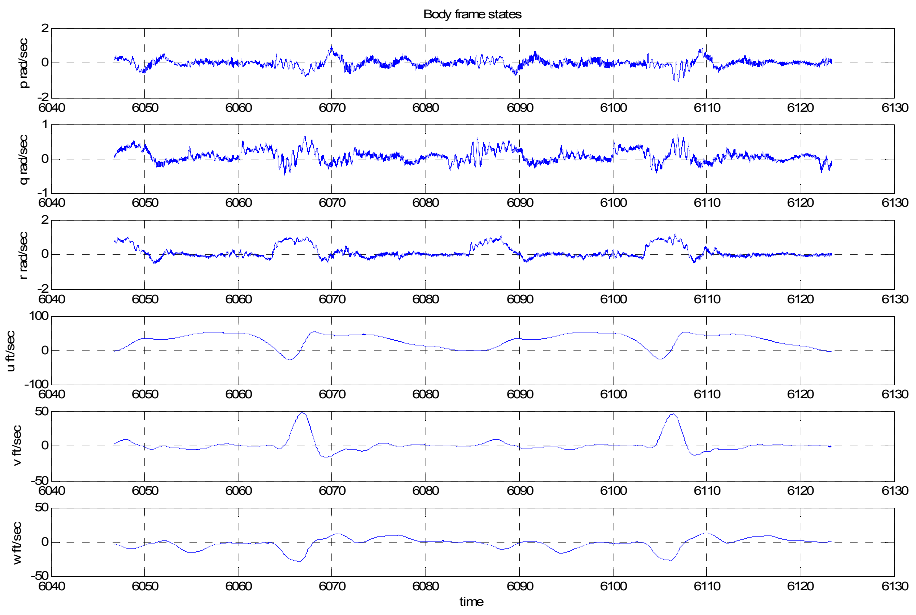

Figure 28 Recorded inner and outer loop states for repeated aggressive reversal of direction maneuvers

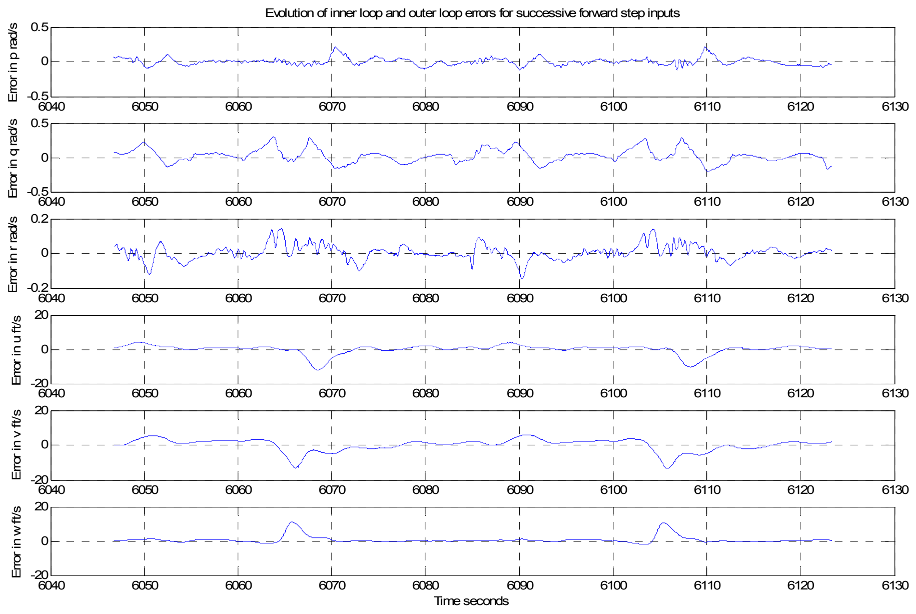

Figure 29 Inner and outerloop tracking error profile

Figure 32 shows the evolution of the NN V matrix weights with concurrent learning adaptation for this maneuver, while Figure 31 shows the evolution of the NN V matrix weights without concurrent learning. It can be seen that with concurrent learning, the weights are less susceptible to evolve in groups. Figure 34 shows the evolution of the NN W matrix weights with concurrent learning, while Figure 33 shows the evolution of the NN matrix weights without concurrent learning. Apart from a slight tendency to not cluster, no visible improvement is seen. This is consistent with the modest improvement seen in the error profile.

A. Concluding remarks on flight test implementation, and directions of future work

The flight test results support the result on uniform ultimate boundedness of all system signals while using concurrent learning adaptation as proved herein. Furthermore, the evolution of NN *W* and *V* matrix weights were observed to have different characteristics when concurrent learning was employed. Particularly, for the simple maneuvers in section A, a tendency for the *V* weights to converge with concurrent learning was seen. This behavior resulted in improved tracking error. For the aggressive maneuvers in the following section, it was seen that the weight matrices behave differently with concurrent learning on, however no clear convergence was seen. As a result, the improvement in performance was marginal. These results indicate as expected that achieving better weight convergence leads to improvement in performance. Due to the fact that other training algorithms are enabled by the approach to theorem 2, and since many other ways for selecting data points are available, we feel this can be achieved even more consistently by improved implementation.

VI. Implementation of Concurrent Learning on a Fixed Wing UAV

The UAV lab at the Daniel Guggenheim school of Aerospace Engineering at the Georgia Institute of Technology maintains a fixed wing UAV based on the commercially available Twinstar RC airplane platform. The GTTwinstar (Figure 30) is made of reinforced Styrofoam which allows for easy reconstruction and light weight. The GTTwinstar is currently being developed as a platform for reliably assessing the robustness and performance of various different adaptive laws using reliable metrics. The concurrent learning NN based MRAC method presented in section III have been implemented on the GT twinstar. Figure 35 shows the performance of the velocity tracking controller in the North East Down (NED) frame of reference. In the flight test from which this data is presented, the inner loop attitude control was augmented with concurrent learning NN based MRAC control. The flight test results support the result on uniform ultimate boundedness of all system signals while using Concurrent learning adaptation as proven in reference 1.

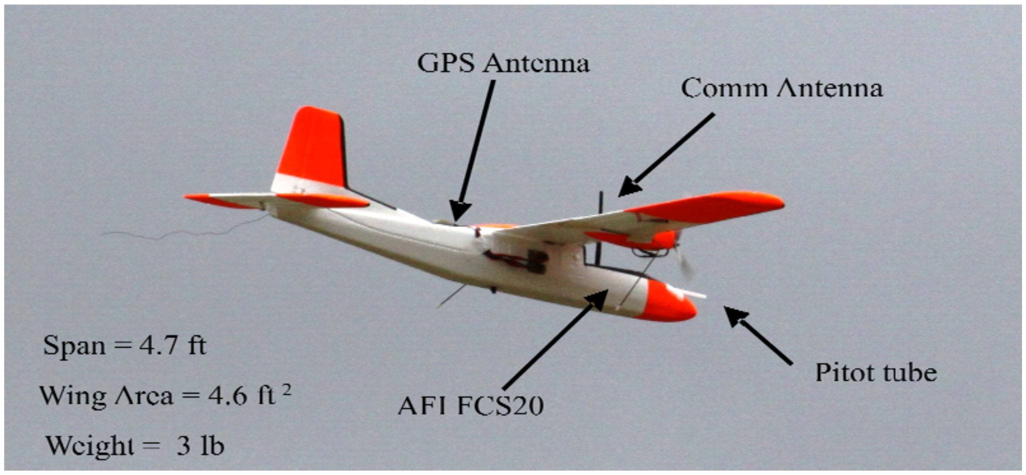

Figure 30 The Georgia Tech GTTwinstar Fixed Wing UAV

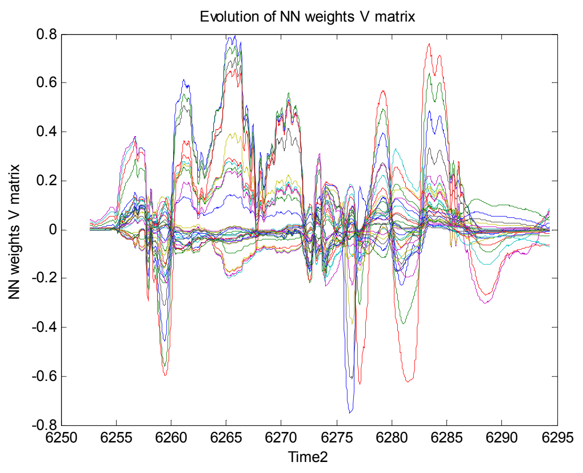

Figure 31 Evolution of NN V matrix weights without Concurrent learning adaptation

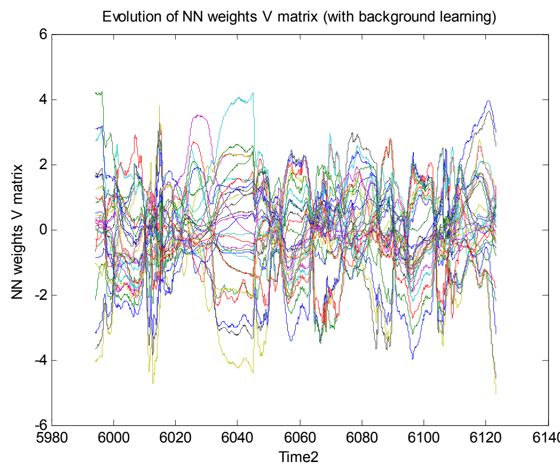

Figure 32 Evolution of NN V matrix weights with Concurrent learning adaptation

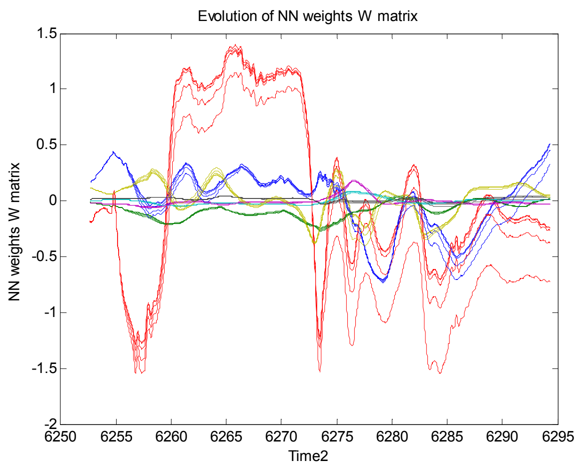

Figure 33 Evolution of NN W matrix weights without Concurrent learning adaptation

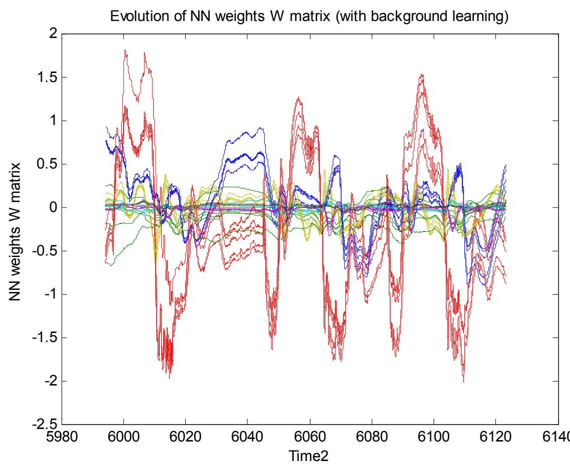

Figure 34 Evolution of NN W matrix weights with Concurrent learning adaptation

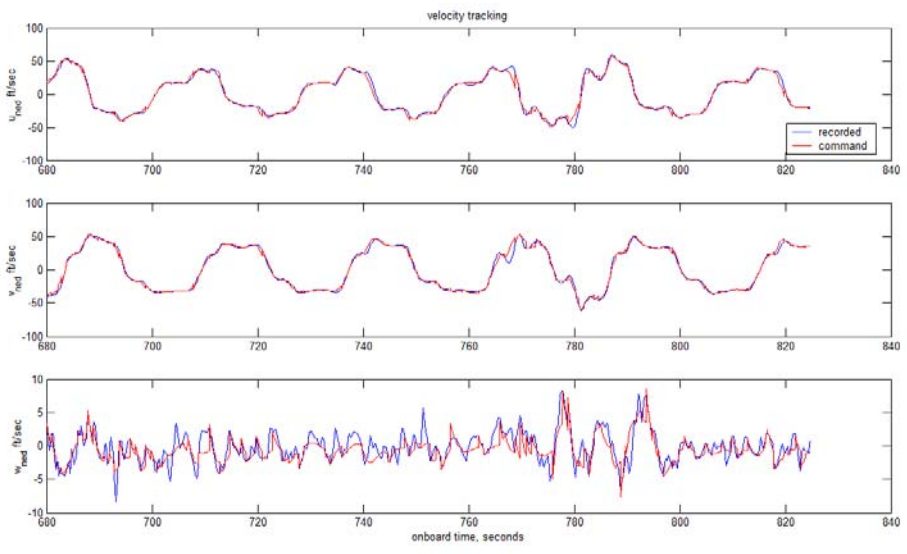

Figure 35 NED velocity tracking error evolution with concurrent learning

VII. Conclusion

In this paper we presented flight test results for a long term learning adaptive control algorithm implemented on a rotarcraft UAV. Flight test results for the implementation of the algorithm on a fixed wing UAV will be presented at the time of the conference. The learning law for the adaptive element (NN) that was used in these flight tests has desirable stability properties[1]. Furthermore, the learning law simulates long term memory by using a history stack of data for concurrent adaptation. The learning law is able to adapt to current as well as past data concurrently, furthermore, the learning law exploits the separability of the underlying subspaces to ensure that the updates based on the stored data do not affect the weight updates based on the current data. In this way, the law is able to incorporate long term learning into the adaptive system without sacrificing the responsiveness of the adaptive controller to instantaneous data.

In the Introduction (section I) we mentioned some of the limitations of the current recursive NN training laws that hinder the adaptive laws from forming a global parameterization of the model error. Here we note how Concurrent Learning alleviates these limitations:

- **Concurrent Learning allows the use of direct training information**: As mentioned in section C, it is possible to present the Concurrent learning NN training law with the direct training information ($r_c = \nu_{ad} - \Delta$). The availability of the direct information can be used to improve the performance of the NN training algorithm.
- **Concurrent Learning allows the concurrent use of both past and recent data:** The use of both past and recent data can be used to simulate long term memory in the NN. Furthermore, the form of Concurrent learning law as presented in equation 32, and 33 ensures that the use of past data does not affect the weight update based on recent data. The framework of Concurrent learning allows the use of past data for adaptation without sacrificing the responsiveness of the adaptation to current data. This ability can be exploited to use proven fixed point iteration schemes for adaptation in the Concurrent.
- **Concurrent Learning alleviates the rank-1 condition[1,2]**: By using orthogonal projection operators as defined in equation 34,35 the Concurrent learning NN increases the rank of the training law. In this way, the training law is able to search in more than one direction in the parameter space for the ideal weights.

A significant contribution of this paper was to present flight test results with Long Term Learning Adaptive Controllers employing Concurrent Learning implemented on a rotorcraft UAV. To the author's knowledge, this is the first time comprehensive flight test results have been presented when the adaptive control uses both the current and the past data *concurrently* for adaptation. The flight test results adhere to the conclusion of theorem 2 of reference 1 that all states remain uniformly ultimately bounded when using Concurrent learning adaptation. Furthermore, expected improvement in tracking performance and weight convergence properties over an approach that only uses current data was noted.

Acknowledgments

This work was supported in part by NSF #ECS-0238993. The authors also thank the members of the GT UAV Research Facility for their support in flight testing activities. Particularly, the authors wish to thank Suresh Kannan, Jeong Hur, Wayne Pickell, Henrik Christophersen, Nimord Rooz, Allen Wu, Claus Christmann, and Scott Kimbrell.

References

[1] Chowdhary G., Johnson E., *Theory and Flight Test Validation of a Long Term Learning Adaptive Flight Controller,* AIAA GNC 2009, Honolulu HI.

[2] Chowdhary G., Johnson E., *Adaptive Neural Network Flight Control using both Current and Recorded Data,* AIAA GNC 2007, Hilton Head Island, SC.

[3] Kutay Ali, Chowdhary G., Calise A., Johnson E., *A Comparision of Two Novel Direct Adaptive Control Methods under Actuator Failure Accomodation*, AIAA GNC 2009, Honolulu, HI.

[4] Spooner J.T., Maggiore, M., Ordonez,R. and Passino, K..M., *Stable Adaptive Control and Estimation for Nonlinear Systems: Neural and Fuzzy Approximation Techniques*, John Wiley and Sons, 2002.

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

[6] Calise A., Hovakimyan N.,Idan M., *Adaptive Output Feedback Control of Nonlinear Systems Using Neural Networks*, Automatica 37(8):1201-1211, August 2001,Note**:** Special Issue on Neural Networks for Feedback Control.

[7] Johnson, E. N. *Limited Authority Adaptive Flight Control*, Ph.D. Thesis, Georgia Institute of Technology, 2000

[8] Johnson E, Calise A., *Limited Authority Adaptive Flight Control for Reusable Launch Vehicles*, 0731-5090 vol.26 no.6 (906-913), Journal of Guidance Control and Dynamics, AIAA 2003.

[9] Kim, Byoung S., Calise, Anthony J., *Nonlinear Flight Control Using Neural Networks*, AIAA-1994-3646, August 1994.

[10] Kim, Y. H., Lewis, F.L., *High Level Feedback Control with Neural Networks, World Scientific Series in Robotics and Intelligent Systems Vol. 21*, World Science Publishing Co. Pte. Ltd. 1998.

[11] Lewis, F. L., *Nonlinear Network Structures for Feedback Control*, Asian Journal of Control, Vol.1, pp. 205-228, December 1999

[12] Johnson, Eric N, Oh, Seung-Min, *Adaptive control Using Combined Online and Background Learning Neural Network*, AIAA CDC 2004, USA.

[13] E. Johnson, M. Turbe and A. Wu, Georgia Institute of Technology, Atlanta, GA; S. Kannan, *Flight Results of Autonomous Fixed-Wing UAV Transitions to and from Stationary Hover,* AIAA-2006-6775, AIAA GNC 2006

[14] Hadad, W., Chellebaonia V., *Nonlinear Dynamical Systems and Control, A Lyapunov Based Approach*, Preprint, 2006, copyrighted by the Authors.

[15] Khalil H., *Nonlinear Systems,* Prentice Hall USA, 3 edition, December 18, 2001

[16] Gelb A., Applied Optimal Estimation, MIT press, 1974.

[17] Narendra K., Annaswamy, A. *A New Adaptive Law for Robust Adaptation without Persistent Excitation*, IEEE Transactions on Automatic control, Vol. 32, No.2, Feb. 1987 pp 134-145.

[18] Ioannous P., J. Sun *Robust Adaptive Control* Prentice Hall USA 1996 (out of print in 2003), Electronic copy at: http://www-rcf.usc.edu/~ioannou/Robust_Adaptive_Control.htm [cited 11 September 2007]

[19] Kim N., *Improved Methods in Neural Network Based Adaptive Output Feedback Control, with Applications to Flight Control,* PhD thesis, Georgia Institute of Technology, 2003

[20] Ponzyak, Alexander S., Sanchez Edgar N., Yu Wen. *Differential Neural Networks for Robust Nonlinear Control, Identification, State Estimation, and Trajectory Tracking,* World Scientific, 2001

[21] Suykens Johan A.K., Vandewalle Joos P.L., De Moor Bart L.R., *Artificial Neural Networks for Modelling and Control of Non-Linear Systems*, Kluwer Academic Publishers, the Netherlands 1996.

[22] Kannan Suresh K., *Adaptive Control of Systems in Cascade with Saturation*, Ph. D. Thesis, Georgia Institute of Technology Atlanta Ga, 2005.

[23] Nakwan Kim, *Improved Methods in Neural Network Based Adaptive Output Feedback Control, with Applications to Flight Control,* Ph.D. Thesis, Georgia institute of Technology, Atlanta, Ga, 2003.

[24] Berberian, Sterling k., *Introduction to Hilbert Spaces*, AMS Chelsea Publishing, 1976.

[25] Akhiezer, N.I., Glazman, I.M., *Theory of Linear Operators in Hilbert Space*, Pitman Publishin Ltd., UK, 1981

[26] Haykin Simon, *Neural Networks a Comprehensive Foundation*, Second Edition, Prentice Hall USA, 1998.

[27] Ochial Keihiro, Toda Naohiro, Usui Shiro, *Kick-Out Learning Algorithm to Reduce the Oscillation of Weights"*, Neural Networks, Vol 7, No. 5, pp 797-807, 1994 Elsevier.

[28] J. A. Jankt, S. M. Scoggins, S. M. Schultz, W. E. Snyder, M. W. White and J. C. Sutton, I11, *Shocking: An Approach to Stabilize Backprop Training with Greedy Adaptive Learning Rates,*0-7803-4859-1/98, IEEE USA 1998.

[29] Patino Daniel H., Carelli Ricardo, Kuchen Benjamin R., *Neural networks for Advanced Control of Robot Manipulators*, IEEE Transactions on Neural Networks, Vol 13, No.2, March 2002.

[30] Lavertsky Eugene, Wise Kevin, *Adaptive Flight Control of Manned/Unmanned Military Aircraft*, Proceedings of the American Control Conference, Portland, June 2005.

[31] Ngia Lester S.H., Sjoeberg Jonas, *Efficient Training of Nerual Nets for Nonlinear Adaptive Filtering Using a Recursive Levenberg-Marquardt Algorithm*, IEEE Transaction on Signal Processing, Vol 48., No. 7, July 2000.

[32] Debnath Lokenath, Mikusinski Piotr, *Introduction to Hilbert Spaces with Applications,* Academic Press, USA, 1999.

[33] N. Hovakimyan, B.-J. Yang, A. Calise, Robust Adaptive Output Feedback Control Methodology for Multivariable Non-Minimum Phase Nonlinear Systems, Automatica, vol. 42, No.4, pp. 513-522, 2006.

[34] Chengyu Cao Hovakimyan, N. , *L1 Adaptive Output Feedback Controller for Systems with Time-varying Unknown Parameters and Bounded Disturbances,* in the proceedings of the 2007 American Control Conference, NY USA.

[35] Volyanskyy, K., Calise, A. J., Yang, B. J., and Lavretsky, E., *An Error Minimization Method in Adaptive Control*, AIAA GNC 2006, Keystone, CO