

Parameter Estimation for a Hybrid Adaptive Flight Controller

Stefan F. Campbell*
SGT Inc., Moffett Field, CA, 94035

Nhan T. Nguyen†, John Kaneshige‡
NASA Ames Research Center, Moffett Field, CA, 94035

and

Kalmanje Krishnakumar§
NASA Ames Research Center, Moffett Field, CA, 94035

This paper expands on the hybrid control architecture developed at the NASA Ames Research Center by addressing issues related to indirect adaptation using the recursive least squares (RLS) algorithm. Specifically, the hybrid control architecture is an adaptive flight controller that features both direct and indirect adaptation techniques. This paper will focus almost exclusively on the modifications necessary to achieve quality indirect adaptive control. Additionally this paper will present results that, using a full non-linear aircraft model, demonstrate the effectiveness of the hybrid control architecture given drastic changes in an aircraft's dynamics. Throughout the development of this topic, a thorough discussion of the RLS algorithm as a system identification technique will be provided along with results from seven well-known modifications to the popular RLS algorithm.

I. Introduction

THIS paper was written as an extension of the work done in References 1 and 2 for the Integrated Resilient Aircraft Control (IRAC) project at the NASA Ames Research Center. This project is focused upon developing flight control systems that maintain aircraft flight qualities even under off-nominal conditions. Aircraft control systems are typically designed using a rigid gain-scheduled approach, which allows for precision design of the flight handling-qualities throughout the entire flight envelope. However, in the case of an aircraft experiencing either a damage or a failure condition, this rigid structure breaks down as the flight dynamics of the aircraft are no longer guaranteed to fit within the gain-scheduled design regime. It is this very issue that IRAC aims to resolve using adaptive control techniques. Indeed, much of the research under the IRAC project has focused upon the use of neural networks in a direct adaptive control framework, building particularly on the architecture proposed in Reference 3. Recently, work has extended into a hybrid approach in which direct adaptation is augmented by indirect adaptation techniques, particularly the Recursive Least Squares (RLS) algorithm. The focus of this paper is to present the practical issues associated with using the RLS algorithm to perform real-time system identification in the context of intelligent aircraft control.

The first section of this paper will present the *direct adaptive flight control* (DAFC) architecture and the *hybrid adaptive flight control* (HAFC) architecture for resilient aircraft control (both architectures are named purely for convenience in this paper). For completeness, brief proofs of the stability of these architectures are outlined. Section II then presents a thorough discussion of some general considerations related to doing real-time system identification. Here the goal is to not belabor theory or proofs, but present implementation details necessary for

* Controls Engineer, Intelligent Systems Division, Mail Stop 269-1.

† Computer Scientist, Intelligent Systems Division, Mail Stop 269-1.

‡ Computer Engineer, Intelligent Systems Division, Mail Stop 269-1.

§ Computer Scientist, Intelligent Systems Division, Mail Stop 269-1.

effective use of the RLS algorithm as well as point towards future research pursuits. Though the information presented in this section synthesizes many well-known sources, it remains essential to the current topic in that it fully explains how results were obtained. Section III then presents 7 well-known modifications to the RLS algorithm, briefly discusses their motivation, and then presents simulation results in the context of resilient aircraft control; results will be compared with the baseline direct adaptive flight control system. Concluding remarks will then provide a discussion of practical issues associated with real-time system identification.

II. DAFC and HAFC

The direct adaptive flight control system is presented below in Figure 1. In general, the pilot's rudder, lateral stick, and longitudinal stick inputs are passed to a reference model (filter), which provides both an angular acceleration and an angular rate command to the system. The rate command is used with system feedback to generate a rate command error, which is passed to a PI controller. The output of the PI controller is then augmented by both the direct adaptive neural network command and the feedforward acceleration command (as outputted from the reference model); this results in a *desired* acceleration command or what is often termed a pseudo control command. This desired command is then passed to a dynamic inversion model to calculate the necessary actuator deflections. The direct adaptive neural network is, in this architecture, trained using the rate command errors. The dynamic inversion model is also updated with known changes to the aerodynamic coefficients as dictated by current flight conditions (angle of attack, mach number, sideslip, etc.). A much fuller discussion of this basic architecture may be found in References 4 and 5. Additionally, this architecture is based on the work of Reference 3. It is important to note here that this adaptive approach is intended to provide consistent handling qualities without reliance on extensive gain scheduling.

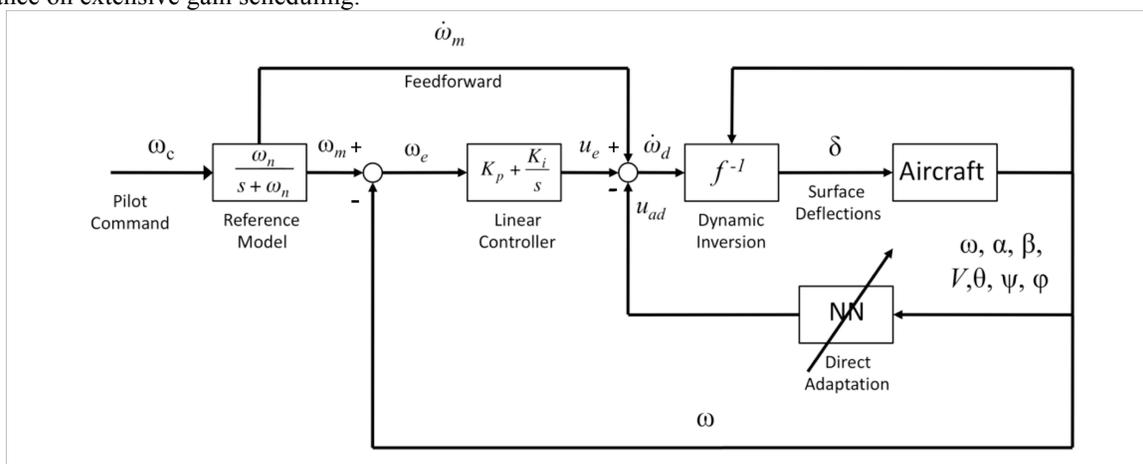


Figure 1. Direct Adaptive Flight Control Architecture.

In the interest of completeness, we provide a brief stability proof for this architecture. Specifically, we model the aircraft as the following:

$$\dot{\omega} = A\omega + Bu + f \quad (1)$$

Here, the matrices A and B are taken as known with f representing an unknown portion of the dynamic. Additionally, u represents the scaled lateral stick, longitudinal stick, and rudder pedal commands. In Figure 1, the surface deflection commands δ are obtained by passing u through control allocation tables, thus B combines the traditional control matrix and the aircraft allocation tables together in this model.

The reference model is then given in the following form:

$$\dot{\omega}_m = A_m\omega_m + B_m\omega_c \quad (2)$$

The pseudo control command is then formulated as the following:

$$\dot{\omega}_d = K_p(\omega_m - \omega) + K_i \int_0^t (\omega_m - \omega) d\tau + \dot{\omega}_m - u_{ad} \quad (3)$$

The gain K_p is a 3x3 matrix defined as $diag(k_{p1}, k_{p2}, k_{p3})$; similarly, the gain K_i is a 3x3 matrix defined as $diag(k_{p1}, k_{p2}, k_{p3})$. The adaptive control command u_{ad} is taken as a linear combination of a series of basis functions $\beta(\omega, u, z)$ (here z represents additional parameters such as angle of attack and sideslip) and some unknown weights W . This is represented algebraically as the following:

$$u_{ad} = \hat{W}^T \beta(\omega, u, z) \quad (4)$$

The dynamic inversion control law is then given as the following:

$$u = B^{-1}(\dot{\omega}_d - Ax) \quad (5)$$

We now define the tracking error as the following:

$$e = \begin{bmatrix} \int_0^t (\omega_m - \omega) d\tau \\ \omega_m - \omega \end{bmatrix} \quad (6)$$

Combining the above equations, we can obtain the ensuing error dynamics.

$$\dot{e} = A_e e + b(u_{ad} - f) \quad (7)$$

Equation 9 below presents the Lyapunov based adaptive control law used to estimate the unknown weights.

$$A_e = \begin{bmatrix} 0 & 1 \\ -K_i & -K_p \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ I \end{bmatrix} \quad (8)$$

The adaptive control law chosen to estimate the unknown weights is the following:

$$\dot{\hat{W}} = -\Gamma \beta(\omega, u, z) e^T L b \quad (9)$$

Here Γ is a diagonal gain matrix and the matrix L is the symmetric, positive definite solution to the standard *Algebraic Lyapunov Equation*:

$$A_e^T L + L A_e = -Q \quad (10)$$

The term Q in Equation 10 is a designer selected, symmetric, positive definite matrix, often chosen as a diagonal matrix. The model uncertainty f is assumed to contain some non-parametric, estimation error (ϵ), as shown below.

$$f = W^T \beta(\omega, u, z) + \epsilon \quad (11)$$

To show stability, we select the following Lyapunov candidate function:

$$V = e^T L e + trace(\tilde{W} \Gamma^{-1} \tilde{W}) \quad (12)$$

Here \tilde{W} represents the difference between the true weights and the predicted weights. The derivative of the Lyapunov function is then obtained as the following:

$$\dot{V} = -e^T Q e - 2e^T P b \varepsilon \quad (13)$$

From this result, we can conclude that:

$$\left\{ \dot{V} < 0 \quad \forall \quad \|e\| : \|e\| > \frac{2\|Pb\|\varepsilon}{\lambda_{\min}(Q)} \right\} \quad (14)$$

The direct adaptive controller is thus *uniformly ultimately bounded* (UUB). Were this analysis to neglect non-parametric uncertainty, Barbalat's lemma could be employed to show that the tracking error will converge to zero.

In order to gain further insight into this controller, we can generalize the control law in Equation 5. By combining Equations 3 and 5 and introducing a set of generalized gains (K_e , K_m , K_c , and K_{ei}) one obtains the result in Equation 15.

$$u = K_e(\omega_m - \omega) + K_m \omega_m + K_c \omega_c + K_{ei} \int_0^t (\omega_m - \omega) d\tau - u_{ad} \quad (15)$$

If we combine this general form of the control law with the system dynamics in Equation 1 and the reference model in Equation 2, we obtain a generalized form of the error dynamics:

$$\dot{\omega} - \omega_m = (A - BK_e)(\omega_m - \omega) + (A_m - A - BK_m)\omega_m + (B_m - BK_c)\omega_c - K_{ei} \int_0^t (\omega_m - \omega) d\tau + u_{ad} - f \quad (16)$$

In order for the real system to track the reference model, the term $(A - BK_e)$ is chosen Hurwitz and the following model matching conditions are to be met:

$$\begin{aligned} A + BK_m &= A_m \\ BK_c &= B_m \end{aligned} \quad (17)$$

For the dynamic inversion control law that has been chosen, the model matching conditions are satisfied by the following relationships.

$$\begin{aligned} K_e &= B^{-1}(K_p + A) \\ K_m &= B^{-1}(A_m - A) \\ K_c &= B^{-1}B_m \\ K_{ei} &= B^{-1}K_i \end{aligned} \quad (18)$$

Though not made explicit in this presentation, the uncertain term f can only be canceled by the adaptive term through the true system's B matrix. If this matrix is not of full rank, then a portion of the uncertainty will persist because the system is not fully controllable.

Thus far we have recapitulated the theory behind the direct adaptive flight control architecture. The hybrid adaptive flight control architecture (the focus of this paper), however, is presented in Figure 2. This architecture utilizes the framework presented in Figure 1 but augments the dynamic inversion model with an indirect adaptive technique to update the model parameters in real-time, thereby introducing a system identification problem. The intention here is to add increased system performance without relying on the use of a high-gain, direct-adaptive control law. Though high-gain adaptive control can aggressively reduce tracking errors, it comes at the cost of high-frequency command signals, which may saturate control actuators or excite higher-order dynamics (such as aeroelastic oscillation modes), and marginal stability (stability margins are effectively lost with high-gain adaptive control).^{1,2}

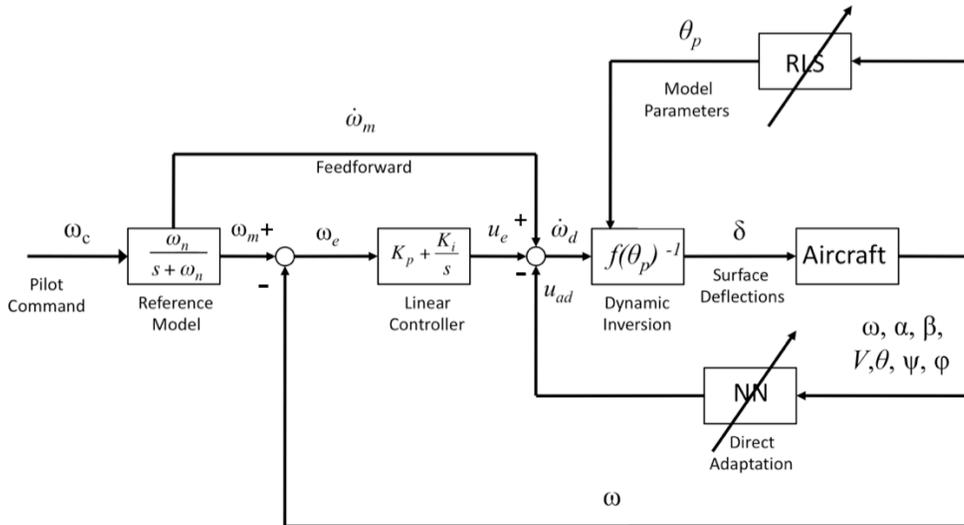


Figure 2. Hybrid architecture.

To begin, the RLS algorithm is usually applied to a model of the general form:

$$y = \varphi^T \theta_o \quad (19)$$

Here φ is the vector of inputs (termed the regressor vector) and θ is the vector of model parameters that are to be identified (a subscript “o” indicates the ideal parameters), and y is the vector of measured signals (the aircraft roll, pitch, and yaw rate derivatives in the current application). For this work, we intend to use the RLS algorithm to approximate changes to the A and B matrices for use in the dynamic inversion (in proceeding sections a non-linear dynamic inversion will be used, but for simplicity we proceed using this linear model). We can rewrite the system dynamics of Equation 1 as the following:

$$\dot{\omega} = \hat{A}\omega + \hat{B}u - (\hat{A} - A)\omega - (\hat{B} - B)u + \gamma \quad (20)$$

The over hats (^) are here used to indicate estimated parameter values. We have also replaced the uncertainty f with the model estimation error γ , which is defined below.

$$\gamma = \dot{\omega} - \varphi^T \theta \quad (21)$$

Now, the control law is more accurately expressed in the subsequent form:

$$u = \hat{B}^{-1}(\dot{\omega}_d - \hat{A}\omega) \quad (22)$$

By combining Equations 3, 20 and 22, we can now reformulate the error dynamics:

$$\dot{\omega}_m - \dot{\omega} = -K_P(\omega_m - \omega) - K_I \int_0^t (\omega_m - \omega) d\tau - \gamma + \Delta A \omega + \Delta B u. \quad (23)$$

Here the notation has been simplified by representing the error between the true parameter values and their estimates with the Greek Δ . For the hybrid adaptive control architecture, the RLS algorithm is used to estimate the parameters A and B. Defining $\tilde{\theta} = \theta - \theta_o$ we can simplify the error dynamics to the following:

$$\dot{e} = A_e e + b(\hat{W}^T \beta(\omega, u, z) + \varphi^T \tilde{\theta} - \gamma) \quad (24)$$

Without derivation, we present the continuous form of the RLS algorithm below (though we will be more concerned with the discrete form in the remainder of this paper). The newly introduced variable P represents the information matrix of the RLS algorithm.

$$\begin{aligned} \dot{\theta} &= P \varphi (\dot{\omega} - \varphi^T \theta) \\ \dot{P} &= -P \varphi \varphi^T P \end{aligned} \quad (25)$$

To show stability, we add an additional term to the Lyapunov function of Equation 12.

$$V = e^T L e + \text{trace}(\tilde{W} \Gamma^{-1} \tilde{W} + \tilde{\theta}^T P^{-1} \tilde{\theta}) \quad (26)$$

Taking the derivative of this Lyapunov function, one obtains

$$\dot{V} = -e^T Q e + 2e^T L b (\varphi^T \tilde{\theta} + \hat{W}^T \beta(\omega, u, z) - \gamma) + \text{trace}[-2\tilde{W}^T \beta(\omega, u, z) - 2\tilde{\theta}^T \varphi (\varphi^T \tilde{\theta} - \gamma) + \tilde{\theta} \varphi \varphi^T \tilde{\theta}] \quad (27)$$

If the approximation error is assumed bounded, a thorough analysis of Equation 27 reveals that all variables (e , $\tilde{\theta}$, ω , and u) are also bounded. For brevity, further details of this proof have been omitted and left to Reference 6.

III. General Implementation Considerations

The following section outlines and provides discussion for many of the well-known implementation details of the recursive least squares algorithm.

A. The Fundamental RLS Algorithm

The fundamental RLS algorithm is derived through the minimization of the cost function J presented in equation 28. In section IV, a discussion of some potential modifications to this algorithm will be presented. Here, however, we provide a brief summary of some well-known properties of this algorithm.

$$J = \sum_{i=1}^n (y(i) - \varphi(i)^T \theta(i))^2 \quad (28)$$

$$\theta(t) = \theta(t-1) + \frac{P_{t-1} \varphi(t)}{(1 + \varphi(t)^T P_{t-1} \varphi(t))} (y(t) - \varphi(t)^T \theta(t-1)) \quad (29)$$

$$P_t = P_{t-1} - \frac{P_{t-1} \varphi(t) \varphi(t)^T P_{t-1}}{(1 + \varphi(t)^T P_{t-1} \varphi(t))}$$

Of particular relevance to resilient aircraft control is the information matrix (sometimes termed the covariance matrix or gain matrix) P , which is used as a gain to determine the update to the parameter estimates θ . This matrix effectively controls the learning of the algorithm. Specifically, the eigenvalues of this positive definite matrix measure the amount of prior information associated with the respective eigenvectors of P , i.e. if the regressor vector is aligned with an eigenvector having a small eigenvalue, then significant learning has already been achieved in the direction of that input, and vice versa. The RLS algorithm thus weights inputs exploring new directions favorably and reduces the weight associated with well-explored directions. Provided with sufficient excitation over time, the

information matrix will become small and the learning will effectively stop. In the context of IRAC, this presents a challenge as damage is experienced as a step change in the aircraft's dynamics, which may occur at any point in a flight; an algorithm that remains active is essential to tracking these step changes. As a result, it is necessary to explore the RLS modifications designed to improve the tracking of time-varying and/or step-changes in the system parameters. This is a motivating factor for this work. For additional properties of this algorithm, the interested reader is referred to Reference 7.

B. Update Equations for HAFC

The model of Equation 1 is a continuous-time model. To implement the RLS algorithm to update the A and B matrices, one would have to approximate the state derivatives. In the aircraft application, these are the derivatives of the roll, pitch, and yaw rates, or more precisely, the angular accelerations. To avoid approximating these derivatives numerically we manipulate Equation 1 to the following:

$$\omega_i - \omega_{i-2} = 2A\omega_{i-1}\Delta t + 2Bu_{i-1}\Delta t \quad (30)$$

Here the derivatives have been approximated by a finite center difference. The problem can now be cast in the form of Equation 19 with the following equivalencies:

$$\begin{aligned} y &= [\omega_i - \omega_{i-2}]^T \\ \varphi^T &= [2\omega_i\Delta t \quad 2u_i\Delta t] \\ \theta &= \begin{bmatrix} A^T \\ B^T \end{bmatrix} \end{aligned} \quad (31)$$

There are several advantages of casting the problem in the form of Equation 31. First, the multiplication of the input vector by the Δt time step will have a similar effect to the normalization technique that is recommended in Reference 8; this will be discussed further in the next section. Moreover, there is no division by a small Δt , this should naturally improve the numerical quality of the learning as compared to approximating the aircraft angular accelerations explicitly. We should note here that Equation 31 is a multi-output representation, in practice we will work with the individual angular accelerations such that θ is a column vector. Finally, Equation 31 necessitates that the algorithm have an initial delay of 3 time steps to ensure that an adequate discrete-time history (ω_1 , ω_2 , and ω_3) is available.

An alternative approach to this is discussed in both References 9 and 10. In this approach, the structure of Figure 3 is introduced and filters are placed on the input and output of the system. These filters effectively reduce derivatives and eliminate the need to numerically differentiate or estimate signals that cannot be directly measured. In the context of the current problem, $H(s)$ could as simple as an integrator ($H(s) = 1/s$). The resulting system is then presented in Equation 32.

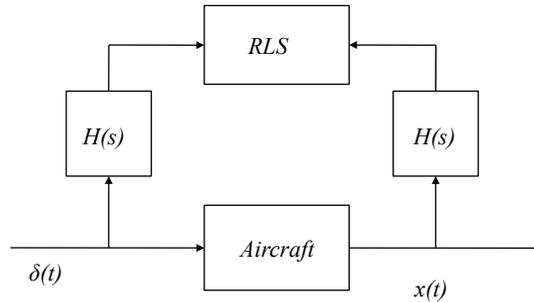


Figure 3. Approach to continuous estimation.

$$\begin{aligned}
y &= [\omega(t) - A \int_0^t x(t)dt - B \int_0^t \delta(t)dt] \\
\varphi^T &= [\int_0^t x(t)dt \quad \int_0^t \delta(t)dt] \\
\theta &= \begin{bmatrix} A^T \\ B^T \end{bmatrix}
\end{aligned} \tag{32}$$

In practice, we found that the former approach provided slightly better results than incorporating the filter $H(s)$. This is in part due to the presence of a constant bias term in the regressor vector, which can grow quite large as time evolves (provided an RLS resetting approach is not taken).

C. Numerical Considerations

The finite precision of computer systems introduces round off errors and, as a result, the numerical quality of the RLS algorithm must be fully considered. Of particular concern is the conditioning of the information matrix P . When the information matrix is poorly conditioned, small variations in the RLS training error result in large variations in the parameter updates. The result is that the algorithm is particularly sensitive to round off error and noise⁸. Additionally, the information matrix can lose positive definiteness as a result of these numerical errors (P should remain > 0 for all time).

To combat this, algorithms have been developed that use a decomposition of the information matrix. Specifically the Potter's Square Root Algorithm and the Bierman' U-D Algorithm are some well-known examples.¹¹ For the current application, the simpler Potter's Square Root Algorithm was selected. This algorithm is based on the Cholesky decomposition of the information matrix and is presented below for convenience. Here Q is the decomposition matrix such that $P = QQ^T$.

$$\begin{aligned}
1. \quad & f(t) = Q(t)^T \varphi(t) \\
2. \quad & \xi(t) = \lambda(t) + f(t)^T f(t) \\
3. \quad & \zeta(t) = 1 / [\xi(t) + \sqrt{\xi(t)\lambda(t)}] \\
4. \quad & K(t) = Q(t-1)f(t) \\
5. \quad & Q(t) = [Q(t-1) - \zeta(t)K(t)f(t)^T] / \sqrt{\lambda(t)}
\end{aligned} \tag{33}$$

From the above, the basic RLS algorithm parameter update is modified as the following:

$$\theta(t) = \theta(t-1) + K(t) / \xi(t) (y - \varphi^T \theta(t-1)) \tag{34}$$

The Cholesky decomposition of a matrix maintains the matrix Q as lower or upper triangular. However, the Potter's Square Root Algorithm will preserve the relationship $P = QQ^T$, but will not preserve the triangular nature of the matrix Q . For potentially improved numerical quality, the algorithm presented in Reference 12 can be used to preserve the triangular nature of the matrix Q . The value of this algorithm may be seen in the light of the constant trace modification to the RLS algorithm that will be discussed in section IV.C. With a triangular Q , the trace of the information matrix can be easily calculated as the following:

$$tr(P) = \sum_{i=1}^m \sum_{j=i+1}^m Q(i,j)^2 \tag{35}$$

This obviates the need to explicitly calculate the matrix P when using the Potter's Square Root algorithm. Additionally, a constant trace can be enforced on the matrix P (where k_0 is the desired trace) by the following:

$$\bar{Q}(t) = \sqrt{\frac{k_0}{\text{tr}(P(t))}} Q(t) \quad (36)$$

In Reference 8 it is also recommended that the regressor vector and the measured signal y be normalized with respect to the magnitude of φ . Thus, the RLS algorithm is trained using the following vectors:

$$y_n = \frac{y}{\max(1, \|\varphi\|)}$$

$$\varphi_n = \frac{\varphi}{\max(1, \|\varphi\|)} \quad (37)$$

This effectively ensures that all of the components of the regressor vector are less than 1 and improves the numerical quality of the algorithm.

Many of the modifications to the RLS algorithm, however, adopt the general form for updating the information matrix as the following:

$$P_t = P_{t-1} - \frac{P_{t-1}\varphi(t)\varphi(t)^T P_{t-1}}{(1 + \varphi(t)^T P_{t-1}\varphi(t))} + R(t) \quad (38)$$

Here $R(t)$ is used to represent any number of different modifications terms, which may or may not be a function of time t . This additional term may be reconciled with the Potter's Square root algorithm by decomposing the matrix $R(t)$ as $V(t)V(t)^T$ and incorporating the Gram-Schmidt process to perform a QR decomposition.¹¹ More specifically, Equation 38 is written as the following:

$$P_t = Q(t)Q(t)^T = \bar{Q}(t)\bar{Q}(t)^T + V(t)V(t)^T \quad (39)$$

Where $\bar{Q}(t)$ is found via the Potters Square root algorithm using $\bar{Q}(t-1)$ as in Equations 33. Equation 39 may now be decomposed as the following:

$$Q(t)Q(t)^T = [\bar{Q}(t) \ V(t)]MM^T[\bar{Q}(t) \ V(t)]^T \quad (40)$$

Here M is an orthogonal matrix found from the following QR decomposition:

$$[\bar{Q}(t) \ V(t)] = M^T Q(t) \quad (41)$$

As a result of the Gram-Schmidt process for the QR decomposition, the matrix $Q(t)$ will remain triangular. This additional discussion is provided to highlight the additional complexity in adding an $R(t)$ modification. Primarily, the $R(t)$ matrix must be properly decomposed and then the Gram-Schmidt procedure must be applied to find $Q(t)$. This will prove relevant when specific modification types are considered.

D. Constrained Parameter Estimation

In order to perform system identification, a model structure $M(\theta)$ with unknown parameters is selected to represent the dynamical system under consideration (in this case an aircraft). The unknown parameters $\theta \in R^n$ are the unknown coefficients of the chosen model. A model set M is then defined as the set of all possible models given a model structure $M(\theta)$ and a domain D_M such that $\theta \in D_M \subset R^n$.^{11,13}

The system identification problem is then to identify the correct model from the model set given correlated input and output data. In the context of the current problem, we wish to do this in a recursive fashion. However, the input data may not be sufficiently rich to identify the true model parameters. In such a case, any two competing models in the model set M may be indistinguishable with respect to the input data. Conversely, if the input data is able to distinguish between any two competing models (from the model set M) then the data is said to be *informative* with respect to M . This condition is guaranteed if the input data is *persistently exciting*.¹³

In more general terms, if the input data is not sufficiently exciting (or rich), the RLS algorithm may move the parameter estimates towards a physically incorrect set of parameters because that set of “wrong” parameters does indeed minimize the cost function J from Equation 28 (as may other parameter combinations in the model set M). As such, it is desirable to select D_M to at least constrain the parameters to a physically realizable set.

To perform this constraining of parameters, the following approach is suggested in Reference 13:

$$\theta(t) = \begin{cases} \theta(t-1) + K(t)(y(t) - \varphi^T \theta(t-1)) & \text{if } \theta(t) \in D_M \\ \theta(t-1) & \text{if } \theta(t) \notin D_M \end{cases} \quad (42)$$

Here the RLS algorithm has been written using the general gain $K(t)$ in order to simplify the presentation. In terms of the recursive least squares algorithm, however, this approach violates the Lyapunov proofs for parameter convergence. More precisely, the Lyapunov function used in Reference 7 to prove parameter convergence (under the persistent excitation assumption) is no longer guaranteed to be a non-increasing function. The Lyapunov function used is the following:

$$V_L(t) = (\theta(t) - \theta_o)^T P(t)^{-1} (\theta(t) - \theta_o) \quad (43)$$

We now use the well-known matrix inversion lemma, given in Equation 44, to illustrate that $V_L(t)$ is an increasing function.

$$P(t)^{-1} = P(t-1)^{-1} + \varphi(t)\varphi^T \quad (44)$$

If the algorithm shown in Equation 42 is enforced at time step t such that the parameter estimates are not changed, the following results (using the algorithm of Equation 29):

$$V_L(t) - V_L(t-1) = (\theta(t-1) - \theta_o)^T P(t)^{-1} (\theta(t-1) - \theta_o) - (\theta(t-1) - \theta_o)^T P(t-1)^{-1} (\theta(t-1) - \theta_o)^T \quad (45)$$

This can be reduced to the following using the matrix inversion lemma:

$$V_L(t) - V_L(t-1) = (\theta(t-1) - \theta_o)^T \varphi(t)\varphi^T (\theta(t-1) - \theta_o). \quad (46)$$

The matrix $\varphi(t)\varphi^T$ is symmetric positive definite. This immediately shows that the Lyapunov function V_L used in Reference 7 may now be increasing. This is an intuitive result in that if the cost function J of Equation 28 is decreasing, the algorithm may push along that trajectory until it encounters the parameter boundary, at which point the Lyapunov function V_L may increase.

As an alternative, Reference 7 suggests a transformation that preserves the nonincreasing nature of the Lyapunov function V_L . This transformation is presented below:

$$\begin{aligned} & \text{if } \theta(t) \notin D_M \\ & \text{Define } \bar{D}_M = P(t)^{-1/2} D_M \\ & 1. \ \rho = P(t)^{-1/2} \theta(t) \\ & 2. \ \text{project } \rho \text{ onto } \bar{D}_M \text{ to obtain } \bar{\rho} \\ & 3. \ \theta(t) = P(t)^{1/2} \bar{\rho} \end{aligned} \quad (47)$$

The complexity of this approach depends entirely on the complexity of the constraint region; for simple upper and lower boundaries on the parameters θ , the complexity is relatively marginal.

A third alternative is discussed in Reference 11. This alternative is given as the following:

$$\begin{aligned}
& \text{if } \theta(t) \notin D_M \\
& 1. \text{ Set } \bar{K}(t) = \lambda K(t) \text{ where } 0 < \lambda < 1 \\
& 2. \theta(t) = \theta(t-1) + \bar{K}(t)(y(t) - \varphi(t)^T \theta(t)) \\
& 3. \text{ Test } \theta(t) \in D_M. \text{ If no } K(t) = \bar{K}(t) \text{ and go to 1, else stop.}
\end{aligned} \tag{48}$$

The above algorithm must be limited to a finite number of iterations. If the algorithm fails to return success, then another action must be taken. A simple solution is to allow this algorithm to degrade to one of the aforementioned approaches.

A final possible approach is discussed in Reference 8. Here the parameters are projected into the parameter domain using the following equation:

$$\theta_p(t) = \theta_c + \min\left(1, \frac{R}{\|\theta(t) - \theta_c\|}\right)(\theta(t) - \theta_c) \tag{49}$$

Here θ_p represents the projected parameters, θ_c is the parameter center (the center of a hypersphere), and R is the allowable L_2 norm deviation from the parameter center.

As a final consideration for parameter constraints, it is notable that in our application the domain of the true parameters may not be exactly specifiable. Different damages and/or failures will produce different model parameters. Selecting a parameter domain D_M that contains the true parameters for all conceivable damaged or failed aircraft is a non-trivial task. As such, the results presented here do not enforce a parameter domain D_M . An ad hoc procedure allowing parameters to only vary by some designer specified multiplicative factor was considered (i.e. each parameter could only increase by a magnitude of 10), but ultimately not employed at this stage. Future work, however, should aim to develop a reasonable parameter domain by thoroughly investigating the dynamics associated with an off-nominal aircraft in which the aircraft remains controllable. The potential importance of these constraints should be viewed in light of the discussion provided in section III.E. Here the motivation for presenting this material was to illustrate how parameter constraints may be enforced, why they were ultimately not enforced in this work, as well as to identify a significant area for future research.

E. Persistency of Excitation

The persistency of excitation condition refers to the richness of the input to the RLS algorithm. If the system is not being sufficiently excited, then it is impossible to distinguish models of a certain model set, as discussed in section III.D. However, this belies the significance of the problem. Poor excitation ultimately leads to a phenomenon known as “bursting” in which the model output suddenly diverges momentarily before returning. This phenomenon is well discussed in References 14 and 15. The core problem is that the excitation signal is not sufficiently rich to distinguish between models of a given model set, thus allowing parameter estimates to drift. Because of the limited excitation, however, this parameter drift does not manifest itself in an error between the measured signal and the model output. The parameters then continue to drift until they suddenly (and somewhat catastrophically) impact the error between the measured signal and the model output. In the context of resilient aircraft control, this presents an unacceptable phenomenon that could fully destabilize the aircraft. Additionally, if the indirect adaptation is to be run nominally throughout the flight of an aircraft in which long periods of little excitation are common, this problem becomes far more relevant. Now, in the current context, it is clear that constraining the parameters to a physically based known region of existence can then serve to prevent excessive parameter drift. Techniques for achieving this were discussed in the previous section. It should here be noted that the bursting phenomenon is somewhat self-correcting as the huge deviation in the model prediction often results in a period of suddenly rich excitation.

For the results presented in this work, however, a two-fold approach was adopted from Reference 8 to handle the bursting phenomenon. First, a simple dead-zone was implemented that turns off learning when the error between the measured signal and the model output falls below a specified threshold. This effectively prevents continued parameter drift by turning off the RLS algorithm. Secondly, a measure of the system excitation was taken at each step to test whether learning was appropriate at the current step. This criteria is given as the following:

$$\text{if } \|P_{t-1}\varphi(t)\| > C \text{ Run RLS, else Stop.} \tag{50}$$

Here it should be understood that the information matrix P serves as a measure of the directions (relative to the vector form of the regressor vector) in which information has been previously collected. Inputs aligned with the eigenvectors of P associated with large eigenvalues are weighted more significantly as this represents a direction in which significant prior information has not been collected (as discussed in section III.A). As a result, Equation 50 measures the relative helpfulness of the current set of inputs (a similar approach has been recommended using the condition number for the matrix P_{t-1}). To simplify selecting the constant C , we use a normalized form of the above condition, as in Equation 51. Here, the two norm would be ideal, but it was desirable not to calculate the eigenvalues of the information matrix.

$$\frac{\|P_{t-1}\varphi(t)\|_1}{\|P_{t-1}\|_1\|\varphi(t)\|_1} > C \quad (51)$$

In practice, implementing both of these criteria may be somewhat redundant. However, a rigorous criteria for selecting either threshold is not entirely obvious. As a result, it seems entirely possible that a regressor vector could produce a small metric for Equation 50 under conditions in which the error between the measured signal and the model prediction is still above the preset value. We thus chose to implement both criteria with reasonable values to indicate when RLS should be active and dormant.

F. Model Parameterization

The model structure chosen can significantly impact the system performance, as is discussed in Reference 16 and 17. Adding additional terms increases the model order (n) and places greater demands on the required level of excitation to uniquely identify the model (the signal must be persistently exciting of order n). However, if the model is not of sufficient order to capture the system dynamics, the unmodeled dynamics can be destabilizing.¹⁶ Our design goal is to then select a set of parameters such that all estimated quantities are slowly varying and all system dynamics may be adequately captured.

Given the above, we now present the parameterization chosen for this work. This primarily entails identifying the appropriate regressor vector. In Reference 2, analysis was performed on the change in the equations of motion for a damaged aircraft where the aircraft C.G. has shifted due to some loss to the structure, as illustrated in Figure 4. Through determination of the angular momentum for the system shown, we find that Equation 52 specifies the aerodynamic moments for the damaged aircraft.

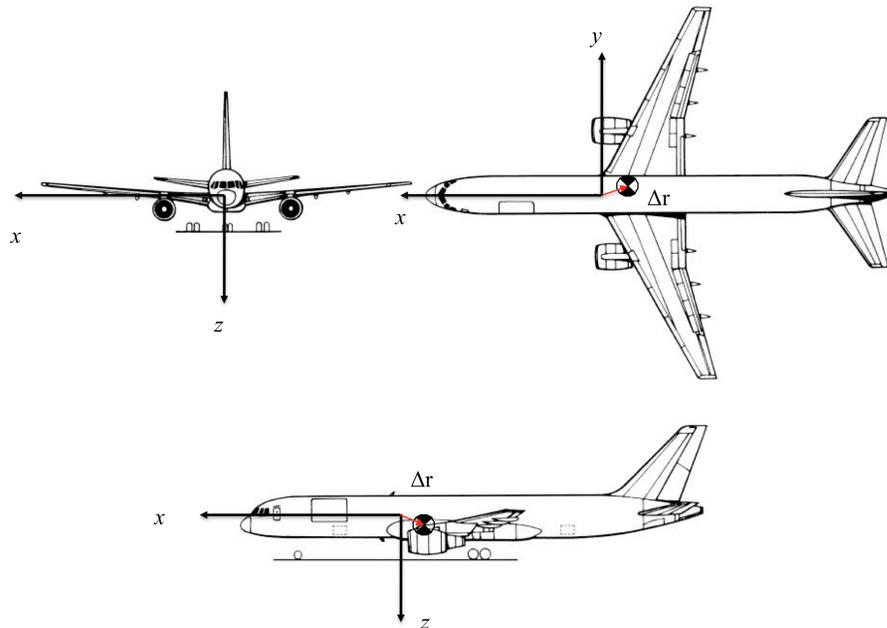


Figure 4. Aircraft coordinate system and potential C.G. shift.

$$M_{aero} = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = I \frac{d\omega}{dt} + \omega \times I\omega + \Delta r \times F_{aero} \quad (52)$$

Here, Δr is the shift in the C.G. location, ω is the angular velocity of the body-fixed frame (a vector $[p \ q \ r]$), I is the inertia tensor (taken about the new C.G. location), and F_{aero} represents the aerodynamic forces. The aerodynamic forces on the system are then represented as the following:

$$F_{aero} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = m\dot{V} + m \frac{d\omega}{dt} \times \Delta r + m\omega \times (\omega \times \Delta r) + m\omega \times V - G \quad (53)$$

Here V is the aircraft velocity as represented in the body-fixed frame and the over dots denote the relative linear accelerations. The vector G is the gravity vector, as represented in the body-fixed frame. Expanding terms and solving for the angular accelerations then yields the following:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = I^{-1} (M - \omega \times I\omega - \Delta r \times F_{aero}) \quad (54)$$

The moments and forces on the aircraft are typically modeled as the following^{8,20,24}:

$$\begin{aligned} X &= \bar{q} S C_x \\ Y &= \bar{q} S C_y \\ Z &= \bar{q} S C_z \\ L &= \bar{q} S b C_l \\ M &= \bar{q} S \bar{c} C_m \\ N &= \bar{q} S b C_n \end{aligned} \quad (55)$$

where,

$$\begin{aligned} C_x &= C_{x0} + C_{x\alpha}\alpha + C_{x\beta}\beta + C_{xh}h + C_{xV}|V| + C_{xp}p + C_{xq}q + C_{xr}r + C_{x\dot{\alpha}}\dot{\alpha} + C_{x\dot{\beta}}\dot{\beta} + \sum_{i=1}^n C_{x\delta_i}\delta_i \\ C_y &= C_{y0} + C_{y\alpha}\alpha + C_{y\beta}\beta + C_{yh}h + C_{yV}|V| + C_{yp}p + C_{yq}q + C_{yr}r + C_{y\dot{\alpha}}\dot{\alpha} + C_{y\dot{\beta}}\dot{\beta} + \sum_{i=1}^n C_{y\delta_i}\delta_i \\ C_z &= C_{z0} + C_{z\alpha}\alpha + C_{z\beta}\beta + C_{zh}h + C_{zV}|V| + C_{zp}p + C_{zq}q + C_{zr}r + C_{z\dot{\alpha}}\dot{\alpha} + C_{z\dot{\beta}}\dot{\beta} + \sum_{i=1}^n C_{z\delta_i}\delta_i \\ C_l &= C_{l0} + C_{l\alpha}\alpha + C_{l\beta}\beta + C_{lh}h + C_{lV}|V| + C_{lp}p + C_{lq}q + C_{lr}r + C_{l\dot{\alpha}}\dot{\alpha} + C_{l\dot{\beta}}\dot{\beta} + \sum_{i=1}^n C_{l\delta_i}\delta_i \\ C_m &= C_{m0} + C_{m\alpha}\alpha + C_{m\beta}\beta + C_{mh}h + C_{mV}|V| + C_{mp}p + C_{mq}q + C_{mr}r + C_{m\dot{\alpha}}\dot{\alpha} + C_{m\dot{\beta}}\dot{\beta} + \sum_{i=1}^n C_{m\delta_i}\delta_i \\ C_n &= C_{n0} + C_{n\alpha}\alpha + C_{n\beta}\beta + C_{nh}h + C_{nV}|V| + C_{np}p + C_{nq}q + C_{nr}r + C_{n\dot{\alpha}}\dot{\alpha} + C_{n\dot{\beta}}\dot{\beta} + \sum_{i=1}^n C_{n\delta_i}\delta_i \end{aligned} \quad (56)$$

We now assume that the velocity magnitude and the altitude are slowly varying. Combining Equations 54, 55, and 56 then suggests that the damaged aircraft may be modeled as the following:

$$\dot{\omega} = [1 \ p \ q \ r \ p^2 \ q^2 \ r^2 \ pq \ pr \ qr \ \alpha \ \beta \ \dot{\alpha} \ \dot{\beta} \ \delta_{lat} \ \delta_{lon} \ \delta_{ped}] \theta_o \quad (57)$$

Here the angular accelerations are treated as a linear combination of 17 parameters. Additionally, the control surfaces have been reduced to the non-dimensional δ_{lat} , δ_{lon} , and δ_{ped} stick and pedal deflections. The terms representing the shift in the C.G. location are now subsumed in the estimate of the unknown parameters as this shift, once it occurs, is constant. Moreover, because the parameters associated with each of the three angular accelerations are disjoint, the system identification problem is broken up into three separate implementations of the identification algorithm (as of yet not entirely specified). Incorporating an appropriate discretization of the derivatives as in Equation 31, we arrive at the following regressor vector (the reader should here refer to Equation 19 for reference):

$$\begin{aligned} \varphi(t)^T = & [1 \ p(t) \ q(t) \ r(t) \ p(t)^2 \ q(t)^2 \ r(t)^2 \dots \\ & p(t)q(t) \ p(t)r(t) \ q(t)r(t) \ \alpha(t) \ \beta(t) \dots \\ & (\alpha(t+1) - \alpha(t-1))/(2\Delta t) \ (\beta(t+1) - \beta(t-1))/(2\Delta t) \dots \\ & \delta_{lat}(t) \ \delta_{lon}(t) \ \delta_{ped}(t)] \Delta t \end{aligned} \quad (58)$$

Issues associated with persistence of excitation have already been addressed in the context of criteria used to define when learning is active and inactive. It is convenient to note here that several methods have been proposed for handling systems that are overparameterized with limited excitation. In References 16 and 17, this issue was addressed by designing an artificial noise signal to be added to the regressor vector. The approach in Reference 17 is particularly noteworthy as it proffers a method for identifying when the system is indeed over-parameterized and thus when artificial noise should be injected into the regressor vector to prevent parameter drift. This approach is not adopted for this work as the more traditional on-off learning approach is favored here. Additionally, the results for all algorithms presented here appeared invariant to the incorporation of the derivatives of sideslip and angle of attack. Thus, these states were removed from the regressor vector. Moreover, a simplified model approach was not adopted because the exact nature of the dynamic coupling cannot be predicted prior to a damage event.

It is also important to note here that our system parameters are assumed, even under nominal operating conditions, to be slowly time-varying with the aircraft's operating conditions. As suggested in Ref. 28, the time-varying nature of the plant parameters can induce bursting under poor excitation, even in the presence of a dead-zone. This provides further justification for the additional excitation check discussed in section III.E. In practice, however, it is anticipated that the damaged aircraft will be trimmed by the pilot and will remain in a neighborhood about a trim state for long periods of time, only deviating significantly during short transitions from one trim state to another. The most critical period of learning will then be the time instant in which the damage or failure actually occurs, which should contain substantial excitation.

G. Dynamic Inversion

The dynamic inversion that is actually implemented in the system is a nonlinear dynamic inversion; the nonlinear model parameters are grouped together and the control matrix is separated from these terms. As long as the control matrix is invertible, a control solution can be found. To guard against the control matrix becoming singular, the determinant is examined at each time step. If the determinant is below a threshold, the newly estimated dynamic inversion parameters are not allowed to propagate to the control system. However, the RLS algorithm is continuously run in order to allow the algorithm to push the control matrix through the zero point. In practice, this is a difficult condition to examine as it is necessary to guard against the control estimate passing through zero, but the true control matrix could indeed be non-invertible (the damage could result in an uncontrollable aircraft). An alternative approach is to consider augmenting the control matrix such that the absolute value of the determinant remains above a specified value.

H. Actuators and Smoothing

For the RLS algorithm, it is necessary to provide actuator information. In practice, however, actuator position information is not available. As such, we adopt an approach where we approximate the actuator position using the

control signal of the previous time step. In a more advanced approach, a notion of the actuators rate limits would be enforced. Early studies prior to the incorporation of a dead zone and/or an excitation check found that poor actuator information could prove destabilizing. However, this destabilization would lead to a period of significant excitation and would eventually self-correct. Unfortunately this is not acceptable in a flight control application. Work on accurately estimating actuator positions is left until more advanced actuator models are available for our simulation. More precisely, the simulation technology used for this study relies upon simple actuator models that do not have higher-order dynamics; as a result, using first order models to approximate actuator positions will invariably prove successful. In practice, the RLS algorithm is turned off during actuator saturation (as is the standard direct adaptation algorithm).

As an additional consideration, experimentation was done to smooth the parameter estimates by passing them through a low-pass filter. This filtering, however, does not adjust the internal states of the RLS algorithm. Thus, the filter shown in Figure 5 does not violate any of the properties of the RLS algorithm. In practice we observed that the filter's cutoff frequency does indeed significantly impact performance; it should be set to allow the RLS algorithm to adapt quickly while rejecting sudden, erroneous parameter jumps. This is an admittedly ad hoc modification. The intention is, however, that if bursting in the parameters does occur, it may provide temporary corrective excitement to the estimation algorithm, but, because of the filtering, the burst parameters will have a dramatically reduced impact on the dynamic inversion model. This technique was not employed in any of the subsequent results that were obtained. Interestingly, this issue will be addressed more explicitly by one of the RLS modifications discussed in section IV.G.

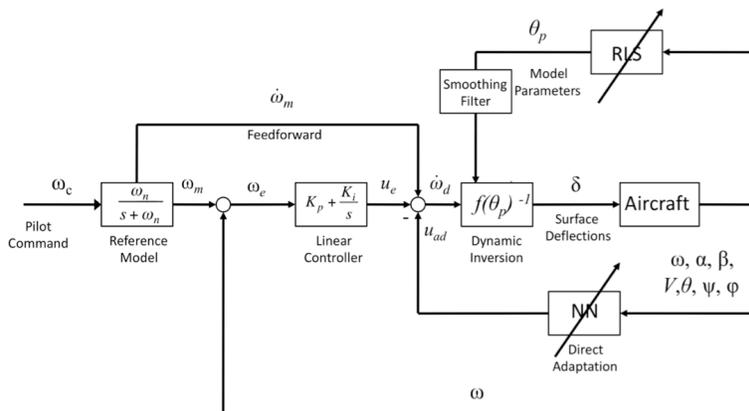


Figure 5. Hybrid architecture with filtering

I. Stochastic Considerations

Much of the literature on system identification contains information on stochastic approaches to the system identification problem. These techniques include the extended least squares algorithm, pseudo-linear regressions, and the Recursive Maximum Likelihood algorithm, to name a few. These approaches all entail adjoining the deterministic model with additional parameters for estimating the noise contribution. Preliminary experimentation with these methods found that they produced little to no appreciable performance improvement in the current context. This result is not surprising since, as pointed out in Reference 11, the RLS algorithm can be given a Kalman Filter interpretation; as such, the algorithm is optimal for Gaussian disturbances. Moreover, the motivation for stochastic techniques is, in larger part, an attempt to correct for parameter biasing that may occur as a result of the stochastic nature of the true system, i.e. when the disturbances are no longer Gaussian. These approaches thus provide for more accurate parameter estimation. For our application, however, no physical meaning is being derived from the estimated parameters, i.e. moments of inertia and moment derivatives are not being interpreted from the estimates. Thus, parameter bias is only a concern in as far as it impacts the accuracy of the dynamic inversion. However, the use of stochastic techniques is an ongoing research pursuit.

IV. RLS Modifications and Results

This section will present the seven RLS modifications schemes that were considered for this work. Here the modifications are presented briefly with minimal justification for their design; this is left to the original authors of these approaches. Results in section V will demonstrate the relative performance of each approach.

A. Exponential Forgetting

The most well known approach to tracking time-varying parameters is the exponential forgetting algorithm. In this approach, old data is exponentially weighted less than new data. The resulting algorithm is the following:

$$\begin{aligned}\theta(t) &= \theta(t-1) + \frac{\alpha P_{t-1} \varphi(t)}{(\alpha + \varphi(t)^T P_{t-1} \varphi(t))} (y(t) - \varphi(t)^T \theta(t-1)) \\ P_t &= \frac{P_{t-1}}{\alpha} - \frac{P_{t-1} \varphi(t) \varphi(t)^T P_{t-1}}{\alpha(\alpha + \varphi(t)^T P_{t-1} \varphi(t))}\end{aligned}\quad (59)$$

Here the parameter α is termed the forget factor and usually assumes a value of .90 to .99. A significant challenge associated with this algorithm may occur when the system being identified reaches a steady-state (i.e. the regression vector becomes a constant).⁹ In such a case, the information matrix P_t will begin to grow rapidly. The result is increased sensitivity to noise and round off error (the matrix P_t will become quite ill-conditioned). The information matrix can, of course, be periodically reset in order to reduce this problem (i.e. when the trace of the matrix exceeds a set value it can be reset to its initialization value). Turning off the algorithm in periods of low excitation and when only small modeling errors are present further mitigates this phenomenon. Under the presence of persistent excitation, however, the information matrix is uniformly bounded and the algorithm is exponentially convergent.¹⁹

B. Exponential Forgetting and Resetting

The exponential forgetting and resetting algorithm is presented in Equations 60.²⁰ Here it is noted that this update law is of the form previously presented in Equation 38. We note that the parameters α , β , δ , and λ are all tunable constants. Additionally, it should be observed that the addition of the constant β is identical to a term that would appear in a Kalman filter interpretation of the RLS algorithm; this term alone is an oft-recommended modification.⁷ The additional δ ensures that P_t is upper bounded. This incorporation of a term to ensure an upper bound for P_t will be revisited again in the Levenber-Marquadt approach. Finally, it should be noted that the term α is not the forgetting factor of the previous section, but is instead a gain adjustment parameter that effectively reduces the parameter update of the traditional RLS algorithm.

$$\begin{aligned}\theta(t) &= \theta(t-1) + \frac{\alpha P_{t-1} \varphi(t)}{(1 + \varphi(t)^T P_{t-1} \varphi(t))} (y(t) - \varphi(t)^T \theta(t-1)) \\ P_t &= \frac{P_{t-1}}{\lambda} - \frac{\alpha P_{t-1} \varphi(t) \varphi(t)^T P_{t-1}}{(1 + \varphi(t)^T P_{t-1} \varphi(t))} + \beta I - \delta P_{t-1}^2\end{aligned}\quad (60)$$

C. Constant Trace

The constant trace algorithm is presented below in Equation 61.⁹ Here the trace of the information matrix is taken as a measure of the amount of information contained in P_t . The algorithm thus enforces a constant on this metric.

$$\begin{aligned}\theta(t) &= \theta(t-1) + \frac{P_{t-1} \varphi(t)}{(1 + \varphi(t)^T P_{t-1} \varphi(t))} (y(t) - \varphi(t)^T \theta(t-1)) \\ \tilde{P}_t &= P_{t-1} - \frac{P_{t-1} \varphi(t) \varphi(t)^T P_{t-1}}{(1 + \varphi(t)^T P_{t-1} \varphi(t))} \\ P_t &= \frac{k_o}{tr(\tilde{P}_t)} \tilde{P}_t\end{aligned}\quad (61)$$

D. Self-Tuning Variable Forgetting Factor

A self-tuning approach to selecting a forgetting factor was proposed in Reference 21. Here the errors associated with the model (Equation 19) estimate are used to derive an information measure for the algorithm. Qualitatively, when the model and the measured signal differ by only a small amount, the forgetting factor is close to unity. When

the errors are large, the forgetting factor is automatically reduced to improve the systems sensitivity by forgetting old information.

$$\begin{aligned}
\theta(t) &= \theta(t-1) + \frac{P_{t-1}\varphi(t)}{(1 + \varphi(t)^T P_{t-1}\varphi(t))} (y(t) - \varphi(t)^T \theta(t-1)) \\
\alpha &= 1 - \left(1 - \frac{\varphi(t)^T P_{t-1}\varphi(t)}{1 + \varphi(t)^T P_{t-1}\varphi(t)} \right) \frac{(y(t) - \varphi(t)^T \theta(t-1))^2}{\Sigma_o} \\
P_t &= \frac{P_{t-1}}{\alpha} - \frac{P_{t-1}\varphi(t)\varphi(t)^T P_{t-1}}{\alpha(1 + \varphi(t)^T P_{t-1}\varphi(t))}
\end{aligned} \tag{62}$$

It should be noted that lower bounding the forget factor at a value of .6 is recommended by the original author of this algorithm (this lower bound is enforced in the following simulation results).

E. Directional Forgetting and Adaptive Directional Forgetting

The directional forgetting algorithm was developed on the idea of adjusting the forgetting in the direction of the incoming regressor vector. The algorithm is given below.

$$\begin{aligned}
\theta(t) &= \theta(t-1) + \frac{P_{t-1}\varphi(t)}{(1 + \varphi(t)^T P_{t-1}\varphi(t))} (y(t) - \varphi(t)^T \theta(t-1)) \\
P_t &= P_{t-1} - \frac{P_{t-1}\varphi(t)\varphi(t)^T P_{t-1}}{\beta^{-1} + \varphi(t)^T P_{t-1}\varphi(t)} \\
\beta(t) &= \begin{cases} \alpha - \frac{1 - \alpha}{\varphi(t)^T P_{t-1}\varphi(t)} & \text{if } \varphi(t)^T P_{t-1}\varphi(t) > 0 \\ 1 & \text{else} \end{cases}
\end{aligned} \tag{63}$$

The information matrix in this algorithm, with persistent excitation, is bounded from above.²² However, the positive definiteness of the information matrix cannot, in general, be guaranteed. As a result, the algorithm may lose its ability to track jumps changes in model parameters. However, Reference 22 offers a simple solution to this with the incorporation of an additional δI term to the information matrix update equation (as in Equation 60). This ensures exponential convergence of the directional forgetting algorithm under persistent excitation.

A more complicated version of the directional forgetting algorithm, referred to here as the *adaptive directional forgetting algorithm* was proposed in Reference 23 and is given below. The justification for this approach is left to the original authors.

$$\begin{aligned}
\xi(t) &= \varphi(t)^T P_{t-1} \varphi(t) \\
\theta(t) &= \theta(t-1) + \frac{P_{t-1} \varphi(t)}{(1 + \xi(t))} (y(t) - \varphi(t)^T \theta(t-1)) \\
P_t &= P_{t-1} - \frac{P_{t-1} \varphi(t) \varphi(t)^T P_{t-1}}{\beta(t)^{-1} + \xi(t)} \\
\beta(t) &= \vartheta(t) - \frac{1 - \vartheta(t)}{\xi(t-1)} \\
\vartheta(t)^{-1} &= 1 + (1 + \rho)(\ln(1 + \xi(t))) + \left[\frac{(v(t) + 1)\eta(t)}{1 + \xi(t) + \eta(t)} - 1 \right] \frac{\xi(t)}{1 + \xi(t)} \\
\eta(t) &= \frac{(y(t) - \varphi(t)^T \theta(t-1))^2}{\lambda(t)} \\
v &= \vartheta(t)(v(t-1) + 1) \\
\lambda(t) &= \vartheta(t) \left(\lambda(t-1) + \frac{(y(t) - \varphi(t)^T \theta(t-1))^2}{1 + \xi(t)} \right)
\end{aligned} \tag{64}$$

In implementation, if $\xi = 0$, then the above update to the information matrix is ignored (i.e. $P_t = P_{t-1}$). The addition of numerous additional parameters is clear from Equation 64 and is a known limitation of this approach, i.e. the previously discussed numerical techniques for improving the learning quality are not obviously applicable.

F. Levenberg-Marquardt

The Levenberg-Marquardt Method upper bounds the information matrix by adding an additional term to the calculation of the information matrix's inverse. This results in the following algorithm (the reader should refer to the matrix inverse lemma of Equation 44):

$$\begin{aligned}
\theta(t) &= \theta(t-1) + P_t \varphi(t) (y(t) - \varphi(t)^T \theta(t-1)) \\
P_t^{-1} &= \alpha P_{t-1}^{-1} + (1 - \alpha) (\varphi(t) \varphi(t)^T + \beta I)
\end{aligned} \tag{65}$$

Here, the obvious challenge is that a matrix inversion calculation must now be performed. Presented in Reference 24 is a technique for obviating this problem. This approach is, however, not clearly amenable to the decomposition techniques designed to improve numerical quality (section III.C).

G. Information-Dependent Data Forgetting

Proposed in Reference 25 is an approach that incorporates a penalty for large changes in the parameter estimates. The minimization criteria now becomes:

$$J = \sum_{i=1}^n (y(i) - \varphi(i)^T \theta(i))^2 \alpha^{n-i} + \lambda |\theta(n) - \theta(n-1)|^2 \tag{66}$$

Here α is the standard exponential forgetting factor and λ is the penalty for parameter changes. The resulting recursive update laws are then derived as the following:

$$\begin{aligned}
\theta(t) &= \theta(t-1) + P_t \varphi(t) (y(t) - \varphi(t)^T \theta(t-1)) + \alpha \lambda (\theta(t-1) - \theta(t-2)) \\
P_t^{-1} &= \alpha P_{t-1}^{-1} + \varphi(t) \varphi(t)^T + \lambda (1 - \alpha) I
\end{aligned} \tag{67}$$

As with the Levenberg-Marquardt technique we observe that the update law for the information matrix requires a matrix inversion. Techniques for reducing this inversion to the inversion of a 2X2 matrix are discussed in Reference 25 and are identical to the approaches used for the Levenberg-Marquardt approach. However, as with the Levenberg-

Marquardt algorithm, this structure does not appear amenable to the Potter's Square Root algorithm or the Bierman U-D factorization. More specifically, the techniques of Reference 25 render the numerator of the standard information matrix update equation (Equation 29) a matrix quantity, as opposed to a scalar value. This fact invalidates the derivations of the standard Potter's Square Root algorithm and the Bierman U-D factorization

H. General Considerations

We have provided a brief survey of some well-known RLS modifications that will be considered for this work. In doing so, the challenge of using these algorithms should be highlighted: there is no rigorous tool for making design decisions for the tunable parameters. Rules-of-thumb have been developed for some of these approaches, but the design approach is lacking in sophistication. Additionally, there are a myriad of additional algorithms that have not been discussed as well as various combinations of techniques that are not explored here. Ultimately, the primary design drivers are the algorithms effectiveness, simplicity of implementation, ease of variable selection, and general numerical quality. These issues will be revisited at the conclusion of the subsequent section.

V. Results

Brief results are first presented here for the fundamental RLS algorithm and the seven aforementioned, well-known modifications. These results are presented for the general transport model (GTM) operating with a 20% loss to the left wing and a corresponding loss of the left aileron. Ultimately, from these results we conclude that the RLS modifications of section IV can all be tuned to produce very similar results.

A. Simulation Specifications

The results presented here are obtained using the FLTz (pronounced *Flight Z*) desktop flight simulator used at the NASA Ames Research Center. This flight simulator is a *GNU C* based software package that features a full non-linear, rigid-body aircraft model. This simulator is limited by the accuracy of the aerodynamic model used to define the aircraft, as well as an inability to model aeroservoelasticity effects and some auxiliary systems (such as the fuel system). For control studies on the GTM, however, the FLTz simulation environment is the most accurate tool currently available.

To simulate the aircraft, aerodynamic data for the undamaged aircraft is obtained from wind tunnel tests on a scaled model. For the damaged aircraft, Vortex Lattice code is used to estimate the aerodynamic derivatives for coefficient buildup. A damage event is then simulated as a step change in the aerodynamic model of the aircraft – i.e. the dynamics of the actual damage occurrence are effectively ignored. To evaluate the hybrid adaptive flight control architecture, we allow the controller to run for a period of steady-state flight, a failure is then injected, followed by a series of doublet maneuvers to help evaluate the systems ability to adapt to the new aircraft dynamics.

B. Baseline Performance

Presented here are results for the baseline dynamic inversion controller (with and without direct adaptation) performing a repeated pitch doublet maneuver. For these simulations, we initialized a nominal aircraft at 8000ft and an indicated airspeed of 300Kts (approximately .52 mach). A failure is then simulated as an instantaneous change in the aircraft dynamics at a time of 25 seconds. To ensure that the aircraft speed remained relatively constant, the aircraft was operated in the auto-throttle control mode (speed mode). Without the benefit of auto throttles, speed drops and the effectiveness of the remaining aileron is reduced to a point where the aircraft becomes uncontrollable (the remaining aileron saturates and can no longer control the aircraft). As a further note, the control system on the yaw channel is an attitude hold system designed to track commanded sideslip angles; the following results thus show roll, pitch, and sideslip tracking.

In general, we are here using brief studies to provide a baseline for qualitatively comparing the RLS algorithms.

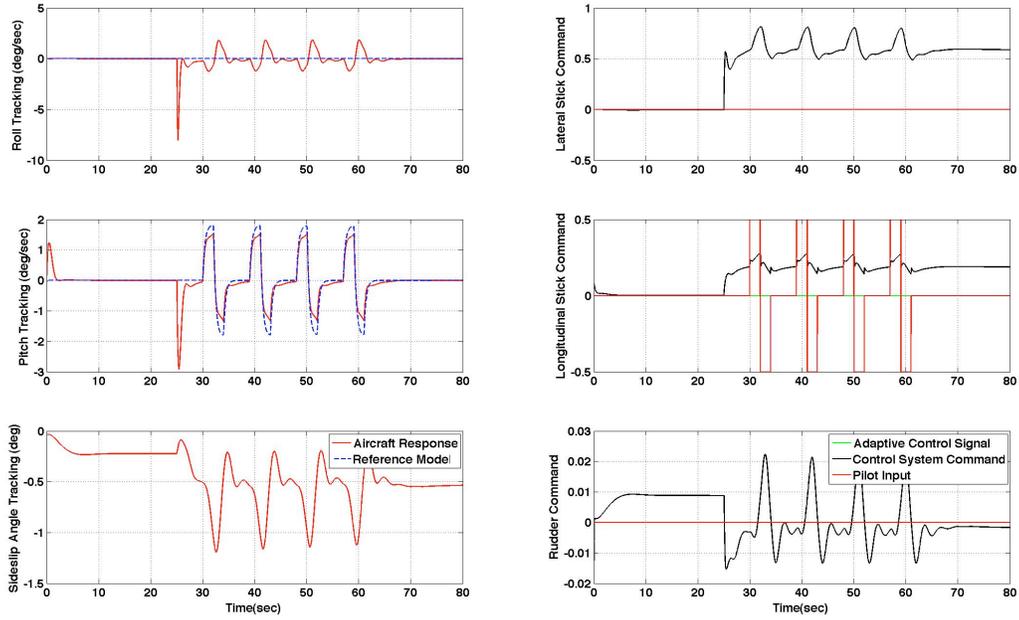


Figure 6. Baseline Controller Tracking without Direct Adaptation

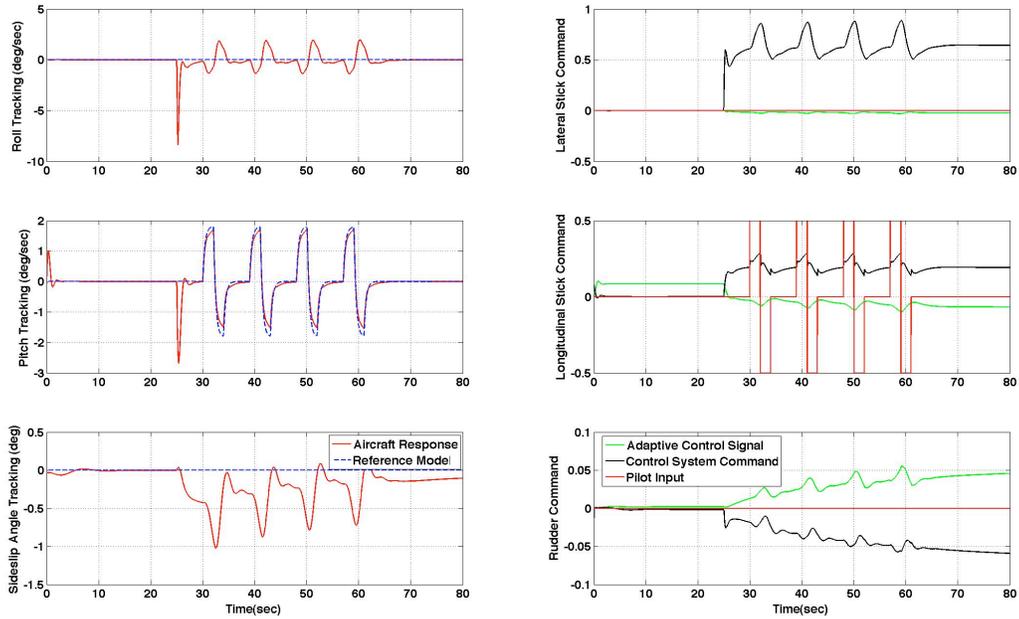


Figure 7. Baseline Controller with Direct Adaptation ($\Gamma = 50.0$)

From Figures 6 and 7 we see that the direct adaptation provides an improvement in the systems tracking quality. These results were obtained using a small gain for the direct adaptive control laws presented in Equation 9 ($\Gamma = 50.0$); higher gains do provide a greater improvement in the tracking quality with a corresponding increase in the adaptive control contribution to the control signal.

C. RLS Algorithm Results

The following figures present results for the hybrid adaptive control architecture using the RLS algorithms of section IV. The scenario considered is identical to that described in the previous section (V.B).

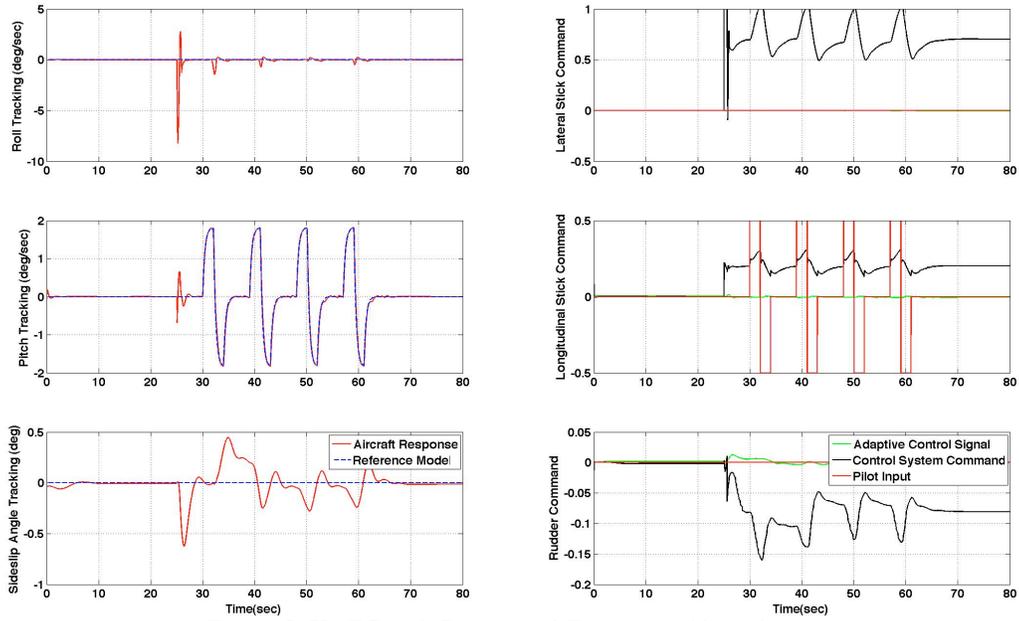


Figure 8. HAFC with Directional Forgetting Algorithm

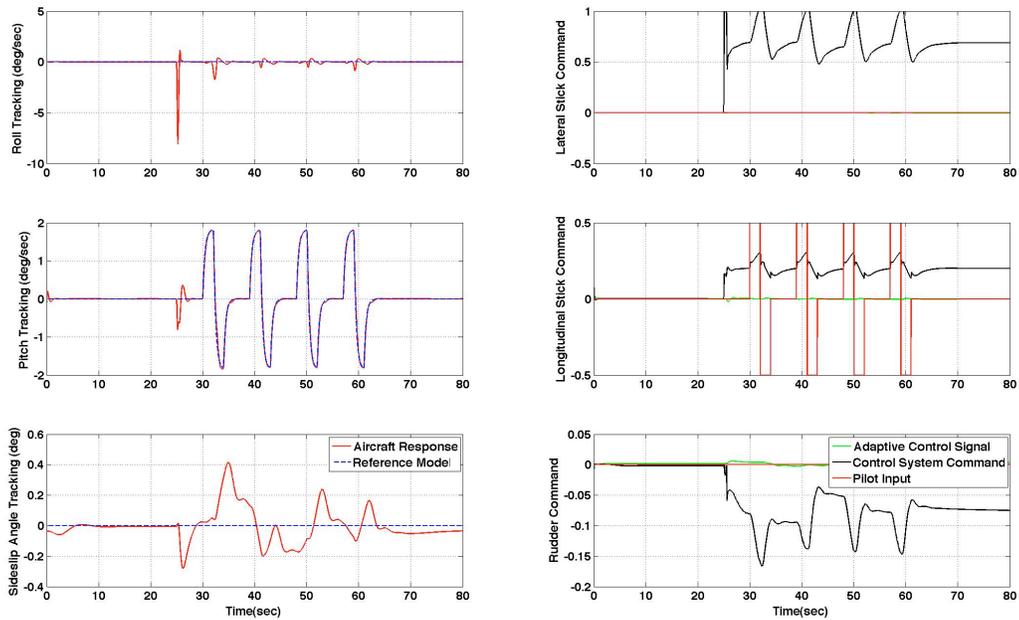


Figure 9. HAFC with Exponential Forgetting and Resetting Algorithm

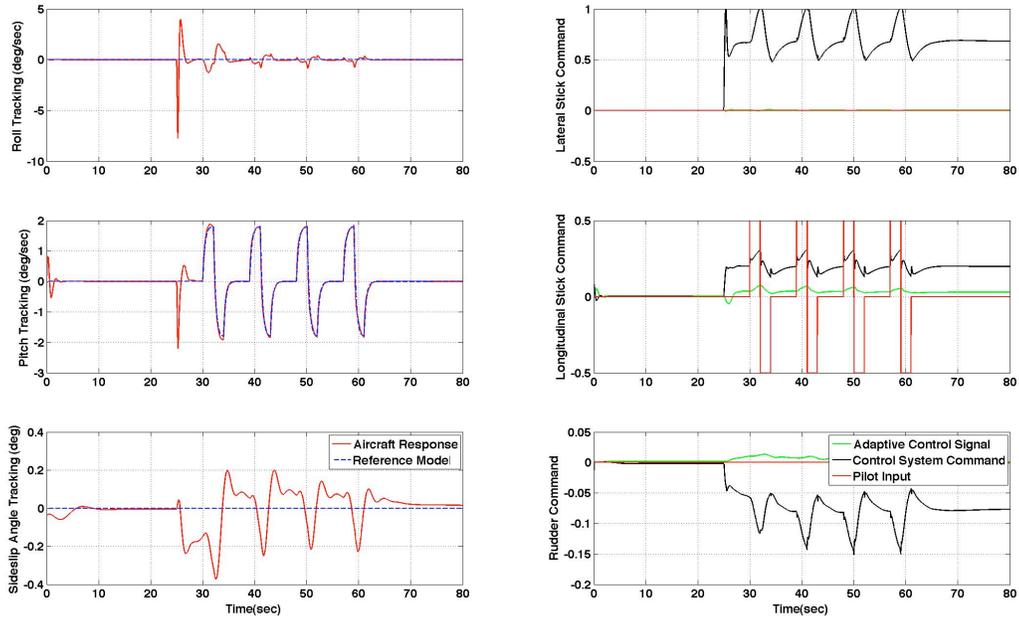


Figure 10. HAFC with Information-Dependent Data Forgetting Algorithm

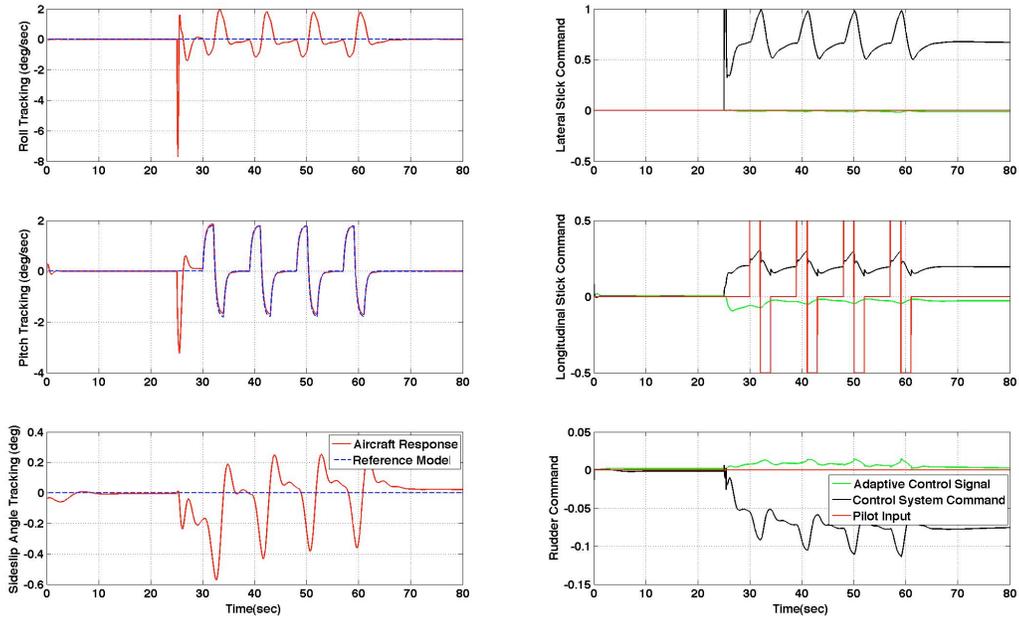


Figure 11. HAFC with the Levenberg-Marquardt Algorithm

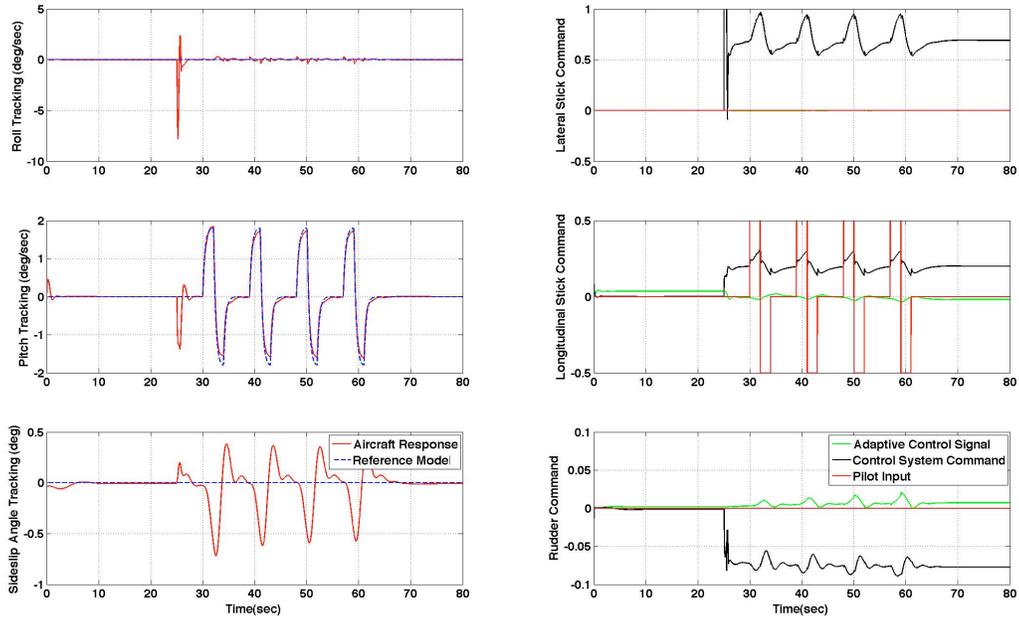


Figure 12. HAFC with Self-Tuning Variable Forgetting Factor Algorithm

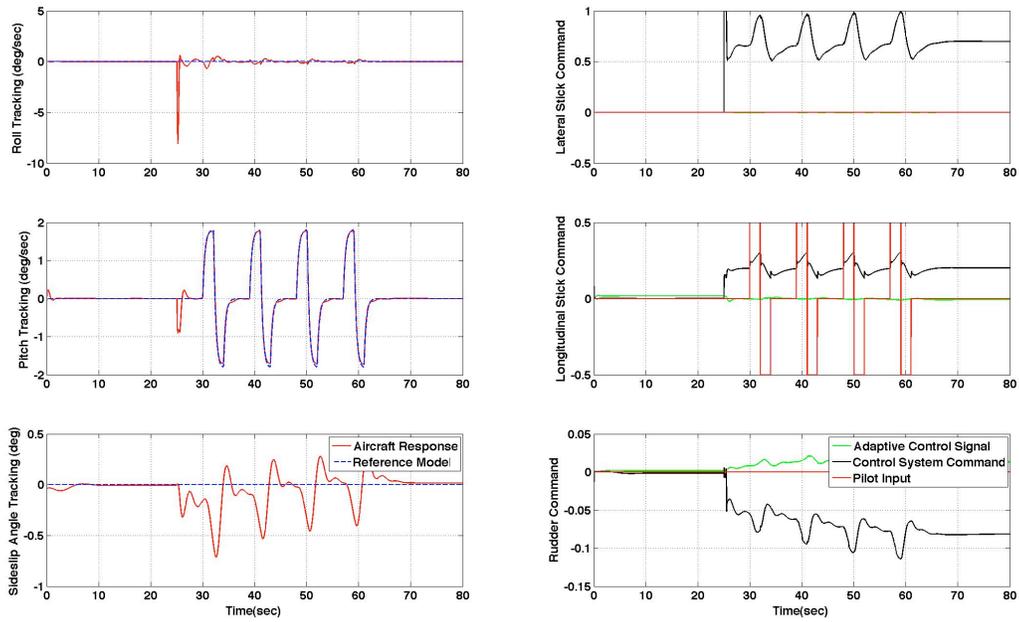


Figure 13. HAFC with Constant Trace Algorithm

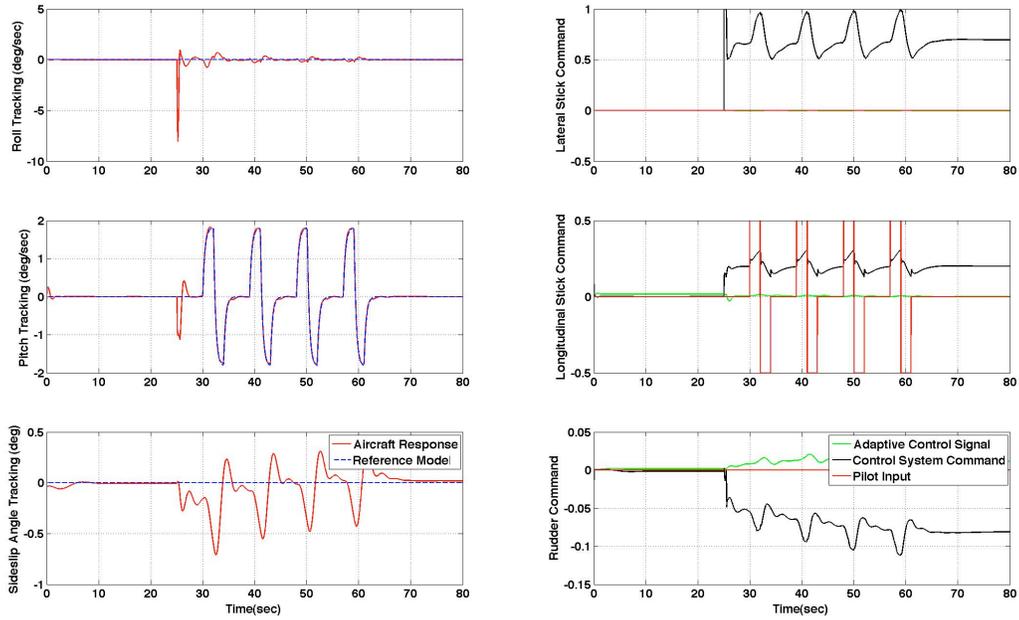


Figure 14. HAFC with Fundamental RLS Algorithm (Equation 29)

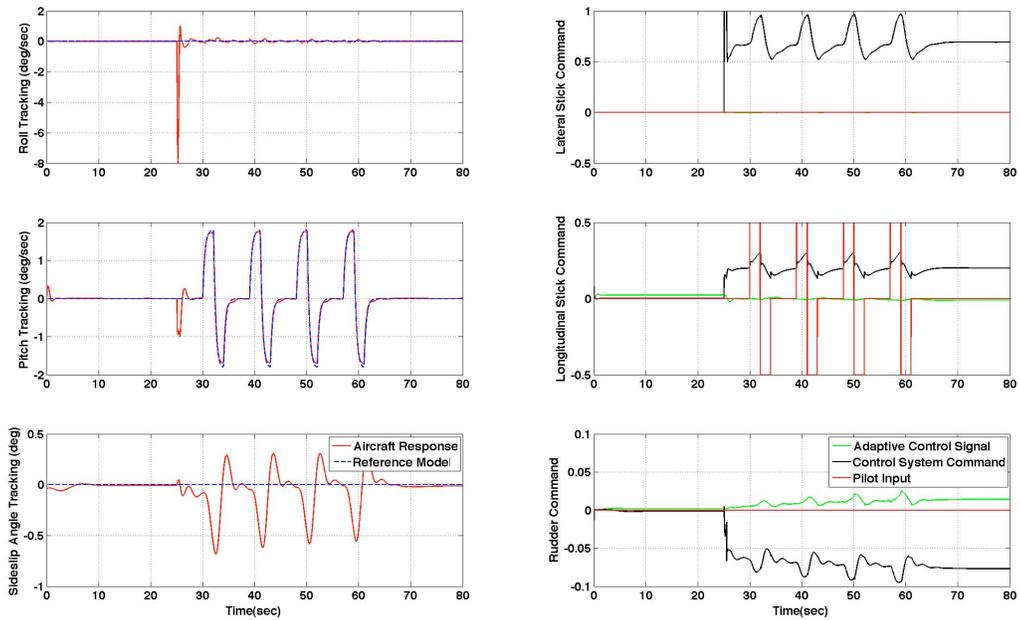


Figure 15. HAFC with RLS Algorithm and a Forget Factor of .985

In analyzing the above results, it is generally observed that the RLS algorithm, irrespective of the algorithm chosen, provided an improvement in the reference model tracking and a reduction in the direct adaptive control signal. Moreover, cross coupling between control axes was reduced. These results are summarized with the calculated RMS values presented below in Table 1.

Controller	RMS Adaptive Control Signal			RMS Tracking Error (deg/sec)		
	Roll	Pitch	Sideslip	Roll	Pitch	Sideslip
Baseline-No Adaptation	0.0000	0.0000	0.0000	0.7328	0.3287	0.5089
Baseline-Direct Adaptation	0.0174	0.0647	0.0286	0.7846	0.2521	0.2847
HAFC: Fundamental RLS	0.0010	0.0106	0.0110	0.4792	0.1079	0.1686
HAFC: RLS Exponential Forgetting	0.0005	0.0142	0.0101	0.4592	0.0982	0.1749
HAFC: Exponential Forgetting and Resetting	0.0005	0.0027	0.0024	0.4924	0.0770	0.1058
HAFC: Constant Trace	0.0008	0.0109	0.0118	0.4763	0.0887	0.1629
HAFC: Self-Tuning Variable Forgetting Factor	0.0007	0.0230	0.0060	0.4579	0.1448	0.1836
HAFC: Adaptive Directional Forgetting	0.0006	0.0042	0.0028	0.5166	0.0616	0.1337
HAFC: Levenberg-Marquardt	0.0088	0.0317	0.0061	0.6956	0.2873	0.1398
HAFC: Information-Dependent Data Forgetting	0.0024	0.0304	0.0048	0.5811	0.1676	0.0973

Table 1. RMS values for Control Performance

To this point, the simulation has been somewhat idealized. The aircraft is simulated at a frequency of 100Hz and the control system is allowed to operate at 100 Hz. Moreover, the sensors provide perfect information. In the following results, we choose to pursue the *exponential forgetting and resetting algorithm* as it produced some of the better results from above. To improve the reality of the simulation, we incorporate random Gaussian noise into the sensor readings, reduce the control frequency to 25Hz, incorporate a time delay of .03 seconds in the control signal, and place a time delay of .02 seconds in the sensor readings. The results of this simulation are presented below.

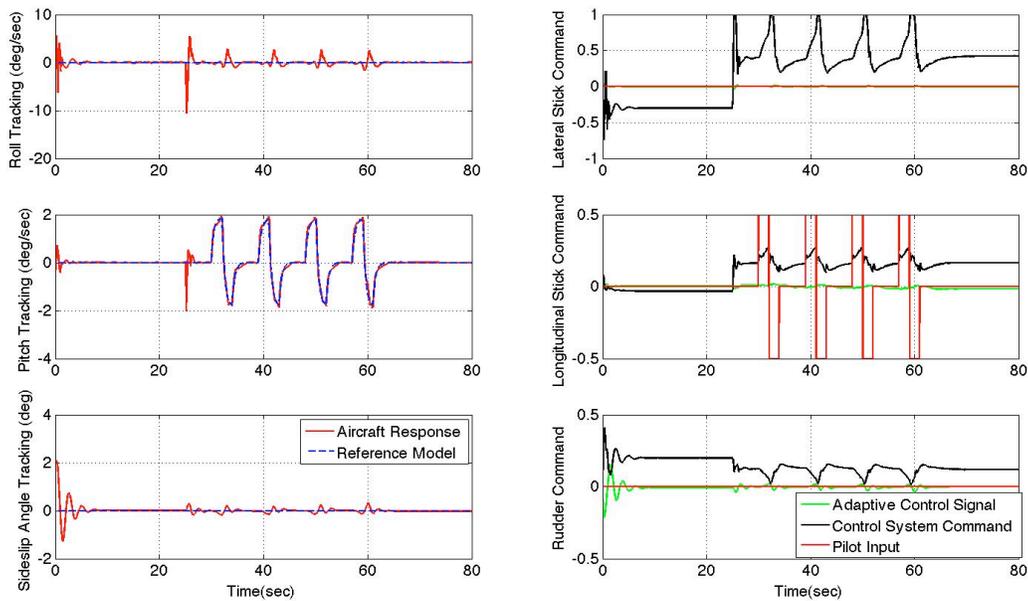


Figure 16. HAFC Exponential Forgetting and Resetting Algorithm under Realistic Operating Conditions

For comparison, we present the baseline direct adaptive controller operating under identical conditions:

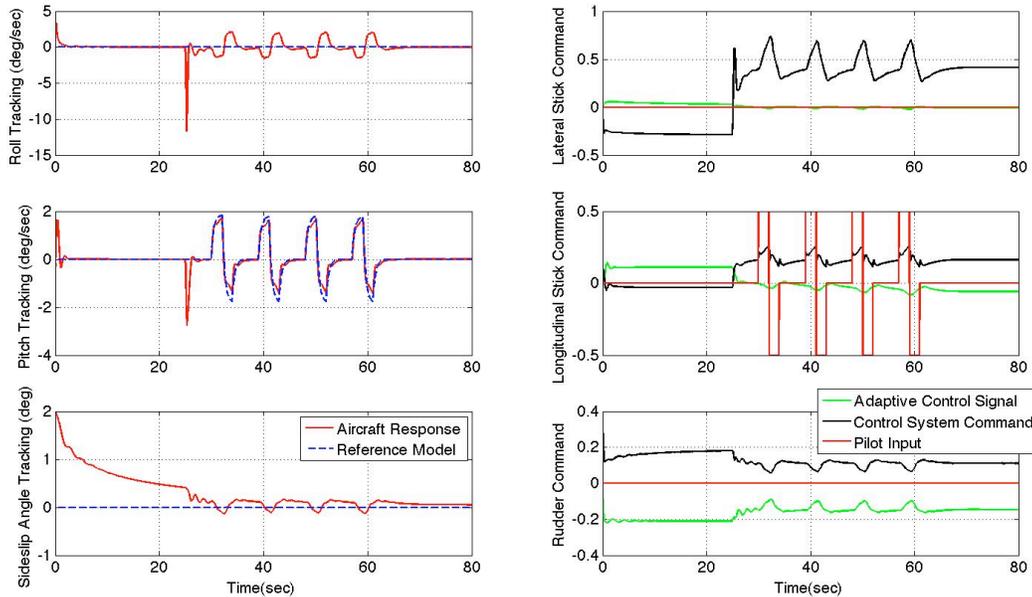


Figure 17. Baseline Direct Adaptive Controller Operating under Realistic Operating Conditions

From Figure 16, we see that the system performance is degraded with respect to the original performance (Figure 9), but that the results are still quite good. As with the previous results, the hybrid architecture improves system tracking and reduces the direct adaptive control signal. An initial high-frequency transient is observed in Figure 16; however, experience has shown that tweaking the on/off thresholds and the initial value of the information matrix can eliminate these transients. Though an in-depth study of robustness to time-delay is beyond the scope of this paper, the results suggest that the measures discussed in this paper produce a control architecture that may not be overly sensitive to such delays.

VI. Conclusions

The results of this paper show that the hybrid adaptive flight control architecture can be very effective. In particular, the hybrid adaptive control architecture exhibits excellent tracking performance and a reduced dependency on the direct adaptive control signal. Moreover, with proper tuning, many of the well-known RLS modification schemes will produce comparable results. The challenge of this tuning process is, however, that there are no apparent rigorous tools for selecting the adjustable parameters or the thresholds used to shut down learning. For instance, selecting dead-zone and excitation thresholds too small or too larger can both be detrimental to performance. In the former, the system is subject to parameter burst, in the latter, the system will not have sufficient time to learn an adequate model of the dynamics. Moreover, many of the RLS algorithms that produce exceptional results introduce many more tunable parameters, this is particularly true for the *adaptive directional forgetting* algorithm and the *exponential forgetting and resetting algorithm*. Simpler algorithms, such as the *constant trace* algorithm, are then much more appealing for their simplicity and their compatibility with existing numerical techniques.

References

- ¹Nguyen, N., Krishnakumar, K., "A Hybrid Intelligent Flight Control with Adaptive Learning Parameter Estimation," *AIAA Infotech@Aerospace 2007 Conference and Exhibit*, Rohnert Park, California, May 2007.
- ²Nguyen, N., Krishnakumar, K., Kaneshige, J., and Nespeca, P., "Dynamics and Adaptive Control for Stability Recovery of Damaged Asymmetric Aircraft," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Keystone, Colorado, August 2006.
- ³Rysdyk, R.T., Calise, A.J., "Fault Tolerant Flight Control Via Adaptive Neural Network Augmentation," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, August 1998.
- ⁴Kaneshige, J., "Integrated Neural Flight and Propulsion Control System," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Montreal, Canada, August 2001.

- ⁵Kaneshige, J., Bull, J., Totah, J.J., "Generic Neural Flight Control and Autopilot System," *AIAA*, August 2000.
- ⁶Nguyen, N., Boskovic, J., "Bounded Linear Stability Margin Analysis of Nonlinear Hybrid Adaptive Control," 2008 American Control Conference, June, 2008.
- ⁷Goodwin, G.C., Sin, K.S., *Adaptive Filtering Prediction and Control*, Prentice-Hall, Englewood Cliffs, New Jersey, 1984.
- ⁸Sripada, N.R., Fisher, D.G., "Improved Least Squares Identification," *Int. J. Control*, Vol. 46, No. 6, 1987, pp. 1889-1913.
- ⁹Astrom, K.J., and Wittenmark, B., *Adaptive Control*, 2nd ed., Addison-Wesley, Reading, MA, 1995.
- ¹⁰Hsia, T.C., *System Identification*, Lexington Books, Lexington, MA, 1977.
- ¹¹Ljung, L., Soderstrom, T., *Theory and Practice of Recursive Identification*, MIT Press, Cambridge, Massachusetts, 1983.
- ¹²Carlson, N.A., "Fast Triangular Formulation of the Square Root Filter," Vol. 11, No. 9, 1973, pp. 1259-1265.
- ¹³Ljung, L., *System Identification: Theory for the User*, Prentice-Hall, Upper Saddle River, New Jersey, 1987.
- ¹⁴Anderson, B.D.O., "Adaptive Systems, Lack of Persistency of Excitation and Bursting Phenomena," *Automatica*, Vol. 21, No. 3, 1985, pp. 247-258.
- ¹⁵Anderson, B.D.O., "Two Decades of Adaptive Control Pitfalls," *International Conference on Control, Automation, Robotics and Vision*, Kunming, China, December 2004.
- ¹⁶Xia, L., and Moore, J.B., "Recursive Identification of Overparameterized Systems," *IEEE Transactions on Automatic Control*, Vol. 34, No. 3, 1989, pp. 327-331.
- ¹⁷Xia, L., Krishnamurthy, V., and Moore, J.B., "Adaptive Estimation in the Presence of Order and Parameter Changes," *International Journal of Adaptive Control and Signal Processing*, Vol. 3, 1989, pp. 283-292.
- ¹⁸Tsakalis, K.S., "Performance Limitations of Adaptive Parameter Estimation and System Identification Algorithms in the Absence of Excitation," *Proceedings of the American Control Conference*, Vol. 2, 1994, pp. 1260-1264.
- ¹⁹Johnstone, R.M., Johnson, R., Bitmead, R.R., and Anderson, B.D.O., "Exponential Convergence of Recursive Least Squares with Exponential Forgetting Factor," *IEEE Conference on Decision and Control*, Vol. 21, 1982, pp. 994-997.
- ²⁰Salgado, M.E., Goodwin, G.C., and Middleton, R.H., "Modified least squares algorithm incorporating exponential resetting and forgetting," *Int. J. Control*, Vol. 47, No. 2, 1988, pp. 477-491.
- ²¹Fortescue, T.R., Kershenbaum, L.S., and Ydstie, B.E., "Implementation of Self-tuning Regulators with Variable Forgetting Factors," *Automatica*, Vol. 17, No. 6, 1981, pp. 831-835.
- ²²Bittanti, S., Bolzern, P., and Campi, M., "Convergence and Exponential Convergence of Identification Algorithms with Directional Forgetting Factor," *Automatica*, Vol. 26, No. 5, 1990, pp. 929-932.
- ²³Kulhavy, R., "Restricted Exponential Forgetting in Real-time Identification," *Automatica*, Vol. 23, No. 5, 1987, pp. 589-600.
- ²⁴Ngia, L.S.H., Sjoberg, J., Viberg, M., "Adaptive Neural Nets Filter Using a Recursive Levenberg-Marquardt Search Direction," *Asilomar Conference on Signals, Systems and Computers*, Vol. 1, 1998, pp.697-701.
- ²⁵Bodson, M., "An Adaptive Algorithm with Information-Dependent Data Forgetting," *Proceedings of the American Control Conference*, Vol. 5, Seattle, 1995, pp. 3485-3489.
- ²⁶Cao, L., and Schwartz, H.M., "A Directional Forgetting Algorithm Based on the Decomposition of the Information Matrix," *Automatic*, Vol. 36, No. 11, 2000, pp. 1725-1731.
- ²⁷Duke, E.L, Antoniewicz, R.F., and Krambeer, K.D., "Derivation and Definition of a Linear Aircraft Model," NASA Reference Publication 1207, August 1988.
- ²⁸Haddadi, A., and Hashtrudi-Zaad, K., "A New Fast Online Identification Method for Linear Time-Varying Systems," *American Control Conference*, Seattle, 2008.
- ²⁹Kreisselmeier, G., "An Approach to Stable Indirect Adaptive Control," *Automatic*, Vol. 21, No. 4, 1985, pp. 425-431.
- ³⁰McRuer, D., Ashekenas, I., and Graham, D., *Aircraft Dynamics and Automatic Control*, Princeton University Press, Princeton, New Jersey, 1973.
- ³¹Roskam, J., *Flight Dynamics of Rigid and Elastic Airplanes*, self-published by Jan Roskam, 1973.
- ³²Vahidi, A., Stefanopoulou, A., and Peng, H., "Recursive Least Squares with Forgetting for Online Estimation of Vehicle Mass and Road Grade: Theory and Experiments," *Vehicle System Dynamics*, Vol. 43, 2005, pp.31-55.
- ³³Wittenmark, B., Astrom, K.J., "Practical Issues in the Implementation of Self-tuning Control," *Automatic*, Vol. 20, No. 5, 1984, pp. 595-605.