WestVirginiaUniversity
THE RESEARCH REPOSITORY @ WVU

Graduate Theses, Dissertations, and Problem Reports

2011

Parameter Estimation Analysis for Hybrid Adaptive Fault Tolerant Control

Peter B. Eshak
West Virginia University

Follow this and additional works at: https://researchrepository.wvu.edu/etd

Recommended Citation

Eshak, Peter B., "Parameter Estimation Analysis for Hybrid Adaptive Fault Tolerant Control" (2011). *Graduate Theses, Dissertations, and Problem Reports*. 2204.
https://researchrepository.wvu.edu/etd/2204

This Thesis is protected by copyright and/or related rights. It has been brought to you by the The Research Repository @ WVU with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you must obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/ or on the work itself. This Thesis has been accepted for inclusion in WVU Graduate Theses, Dissertations, and Problem Reports collection by an authorized administrator of The Research Repository @ WVU. For more information, please contact researchrepository@mail.wvu.edu.

Parameter Estimation Analysis for Hybrid Adaptive Fault Tolerant Control

Peter B. Eshak

Thesis submitted to the

College of Engineering and Mineral Resources

at West Virginia University

in partial fulfillment of the requirements

for the degree of

Master of Science
in
Mechanical Engineering

Approved by

Mario Perhinschi, Ph.D., Chair

John Loth, Ph.D.

Kenneth Means, Ph.D.

Department of Mechanical and Aerospace Engineering

Morgantown, West Virginia

2011

Keywords: Fault Tolerance, Hybrid Adaptive Control,

Neural Networks, Parameter Estimation

Abstract

Parameter Estimation Analysis for Hybrid Adaptive Fault Tolerant Control

Peter B. Eshak

Research efforts have increased in recent years toward the development of intelligent fault tolerant control laws, which are capable of helping the pilot to safely maintain aircraft control at post failure conditions. Researchers at West Virginia University (WVU) have been actively involved in the development of fault tolerant adaptive control laws in all three major categories: direct, indirect, and hybrid. The first implemented design to provide adaptation was a direct adaptive controller, which used artificial neural networks to generate augmentation commands in order to reduce the modeling error. Indirect adaptive laws were implemented in another controller, which utilized online PID to estimate and update the controller parameter. Finally, a new controller design was introduced, which integrated both direct and indirect control laws. This controller is known as hybrid adaptive controller.

This last control design outperformed the two earlier designs in terms of less NNs effort and better tracking quality. The performance of online PID has an important role in the quality of the hybrid controller; therefore, the quality of the estimation will be of a great importance. Unfortunately, PID is not perfect and the online estimation process has some inherited issues; the online PID estimates are primarily affected by delays and biases. In order to ensure updating reliable estimates to the controller, the estimator consumes some time to converge. Moreover, the estimator will often converge to a biased value. This thesis conducts a sensitivity analysis for the estimation issues, delay and bias, and their effect on the tracking quality. In addition, the performance of the hybrid controller as compared to direct adaptive controller is explored.

In order to serve this purpose, a simulation environment in MATLAB/SIMULINK has been created. The simulation environment is customized to provide the user with the flexibility to add different combinations of biases and delays to the explored derivatives. Biases were considered in the range -500% to 500% and delays in the range 0.5 to 40 seconds. The stability and control derivatives considered in this research effort are a combination of decoupled derivatives in the three channels, longitudinal, lateral, and directional. Numerous simulation scenarios and flight conditions are considered to provide more credibility to the obtained results. In addition, a statistical analysis has been conducted to assess the results. The performance of the control laws has been evaluated in terms of the integral of the error in tracking the three desired angular rates, pitch, roll, and yaw. In addition, the effort of the neural networks exerted to compensate for tracking errors is considered in the analysis as well.

The results show that in order to obtain reliable estimates for the investigated derivatives, the estimator needs to generate values with less than five seconds delay. In addition, derivatives estimates are within 50% or -15% off the exact values. Moreover, the importance of updating derivatives depends on the maneuver scenario and the flight condition. The estimation process at quasi-steady state conditions provides reliable estimates as opposed to estimation during fast dynamic changes; also, the estimation process has better performance at large rate of change of derivatives values.

Acknowledgement

Thank you God for your help throughout all times, you are the source of all wonderful things I have in my life. I would like first to thank my parents for their love, encouragement, and support throughout my life. I would like to take this opportunity to thank Dr. Mario Perhinschi for providing much guidance as my advisor; I would not have been able to finish without your help. You are not only the best advisor and teacher any student could ask for, but you are also a wonderful role model.

I would also like to acknowledge and thank Dr. John Loth and Dr. Kenneth Means for being in my defense committee and taking time from your busy schedules to review and contribute your thoughts to this research effort.

I would like to thank the West Virginia University Freshman Program and Dr. Robin Hensel in particular for giving me the opportunity to work there; you helped me to finish my degree. I also want to thank Dr. Samir Shoukry and Dr. Gergis William; you are both the reason I am here now. Thank you Dr. Nithi Sivaneri for your support as my academic advisor; you helped me to finish my degree in a suitable period.

Finally, I would like to thank all my friends for always providing me with the love and support needed to successfully finish my study.

Table of Contents

List of Figures

List of Tables

List of Symbols

Acronyms

AOA	Angle of Attack
EMRAN	Extended Minimal Resource Allocating neural network
FD	Frequency Domain
FDC	Flight Dynamic and Control Toolbox
FTR	Fourier Transform Regression
GUI	Graphical User Interface
IFCS	Intelligent Flight Control System
MRAC	Model Reference Adaptive Control
NASA	National Aeronautics and Space Administration
NLDI	Non-linear Dynamic Inversion
NN	Neural Network
PID	Parameter Identification
PTNN	Pre-Trained Neural Network
SOFFT	stochastic optimal feed-forward and feedback technique
SHL	Single Hidden Layer neural network
TD	Time Domain
WVU	West Virginia University

English Symbols

A	System stability matrix
B	System control matrix
$[F_x]_B$	Total external force in ‘x’ direction with respect to point B
$[F_y]_B$	Total external force in ‘y’ direction with respect to point B
$[F_z]_B$	Total external force in ‘z’ direction with respect to point B
$\mathcal{F}$	Fourier Transformation operator
H	Altitude
q	Body axis - roll rate
L, M, N	Roll, pitch, and yaw moments about the x-y-z axes, respectively

$[M_x]_B$	Total external moment in 'x' direction with respect to point B
$[M_y]_B$	Total external moment in 'y' direction with respect to point B
$[M_z]_B$	Total external moment in 'z' direction with respect to point B
M	Mach number
p	Body axis - pitch rate
r	Body axis - yaw rate
$\tilde{s}$	The frequency domain of parameter "s"
$\hat{s}$	The estimation of parameter "s"
$\dot{s}$	Rate of parameter "s"
t	Time
u	Flight control surface deflection(s), deg
V	Velocity
X	Aircraft state(s)
$\dot{x}$	Rate of change of the state *x*
y	Aircraft output(s)

Greek Symbols

α	Angle of attack
$\dot{\alpha}$	Rate of angle of attack
β	Sideslip angle
$\dot{\beta}$	Rate of sideslip angle
δ_i	Control surface (i) deflection
δ_a	Aileron deflection
$\delta_{e/s}$	Elevator/Stabilator deflection
δ_i	Rudder deflection
θ	Pitch angle
ϕ	Bank angle
ψ	Direction angle
ω	The angular frequency

Derivatives

$C_{l\beta}$	Coefficient of rolling moment due to sideslip angle
C_{lp}	Coefficient of rolling moment due to roll rate
$C_{l\delta a}$	Coefficient of rolling moment due to aileron deflection
$C_{m\alpha}$	Coefficient of pitching moment due to angle of attack
C_{mq}	Coefficient of pitching moment due to pitch rate
$C_{m\delta s}$	Coefficient of pitching moment due to stabilator deflection
$C_{n\beta}$	Coefficient of yawing moment due to sideslip angle
C_{nr}	Coefficient of yawing moment due to yaw rate
$C_{n\delta r}$	Coefficient of yawing moment due to rudder deflection

Chapter 1. Introduction

1. Introduction

Although air travel represents the safest means of transportation with the least fatal accidents, aircrafts safety is the main concern while designing their control systems. System stability and good commands tracking were successfully accomplished by conventional design approaches such as gain scheduling; however, at a moment of hazardous flight conditions, such as failure, two factors decide the outcome of the situation. The first one is the capability of the pilot to become aware of the situation; the second is the extent and severity of the failure, which might overwhelm the pilot. Both mentioned factors encouraged research efforts to develop intelligent flight control systems (IFCS) more powerful in off nominal flight conditions [1].

State-of-the-art intelligent control approaches are required to discover failure and inform the pilot about the situation in a short time; as well as being capable of adapting the aircraft control system online. Conventional gain scheduling failed to satisfy these two crucial requirements for safety after failure. The second requirement in particular was the main objective of the innovative adaptive control approaches.

Researchers introduced different designs and structures for adaptive control laws [2]; however, the ability to adapt the control system online remains the common duty of all of them. Direct adaptive control laws in this study are a good example that exploits the functionality of neural networks in approximating nonlinear function to be used in modeling system uncertainties. Indirect adaptive laws are the other part in this study, which incorporates parameter identification techniques (PID) to update the controller parameters online.

This research effort analyzes the performance of one of the adaptive controllers known hereafter as hybrid adaptive controller. This controller exhibited promising results in enhancing the tracking performance of the F-15 IFCS, developed in NASA, especially after failure occurrence. The hybrid adaptive controller - as shown in Figure 1-1 - combines direct adaptive laws in conjunction with indirect adaptive laws.

The tracking performance of the hybrid controller depends on the quality of its two main parts, direct laws (online neural networks) and indirect laws (online PID). There are different online PID techniques but none of them is perfect in estimation and their estimates might have delays, biases, or both as compared to the actual value. Real parameter identification processes take time before convergence to usable values; furthermore, the convergence does not have to yield to perfect values. This research effort performs a sensitivity analysis for the potential issues in online estimation in order to evaluate the effect of those problems on the performance of the hybrid controller.

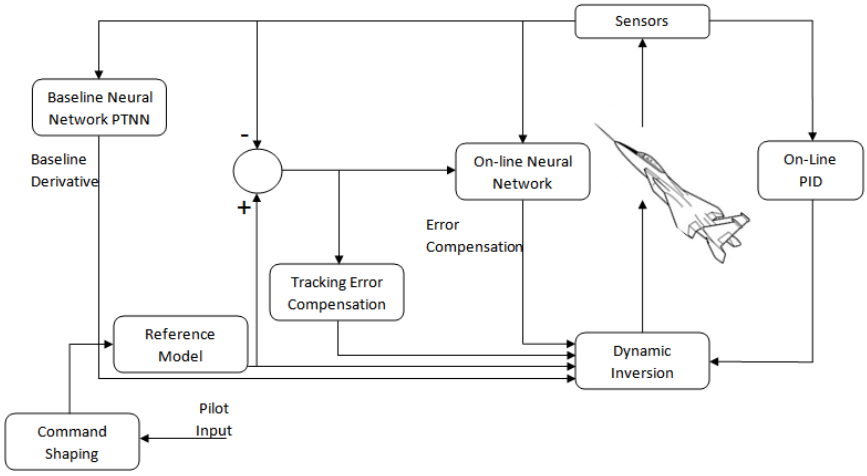

Figure 1-1: Architecture of the IFCS F-15 hybrid adaptive controller

In order to create a simulation environment suitable for this analysis, the implemented hybrid controller in this research replaced real online PID with a direct parameter computation. This allows simulating a perfect PID estimation as well as different levels of PID performance.

The range of delays and biases, which will be added to the perfect estimates for the purpose of this sensitivity analysis, were determined based on literature review and conducted simulations that compared between one of the best PID algorithms [3], Fourier Transform Regression (FTR), and the direct computation process.

A previous research showed that FTR estimators generate estimates with accuracy around 30%, which will degrade further after failure [4]. [3] Moreover, The comparison between different parameter estimation algorithms shows that FTR algorithm is one of the best on-line applications; therefore, the considered bias range for the analysis extended to more than 30%.

In addition, a SIMULINK simulation, as shown in Figure 1-2, provides both computed and FTR estimates to picture the range of potential biases and delays. The simulation showed that the bias range for different derivatives was between 10% and 50%. For instance, the estimation of $C_{m\alpha}$ incurred an error of 44% [Figure 1-3].

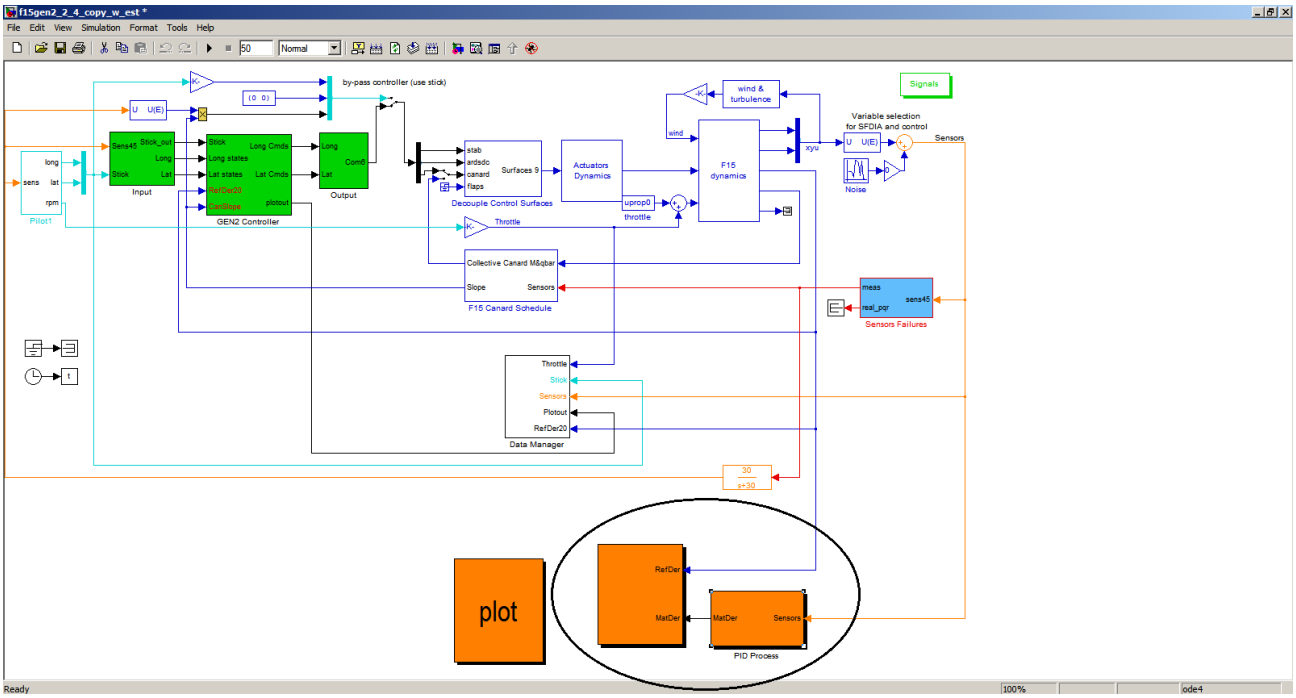

Figure 1-2: FTR derivatives vs. reference derivatives – SIMULINK block

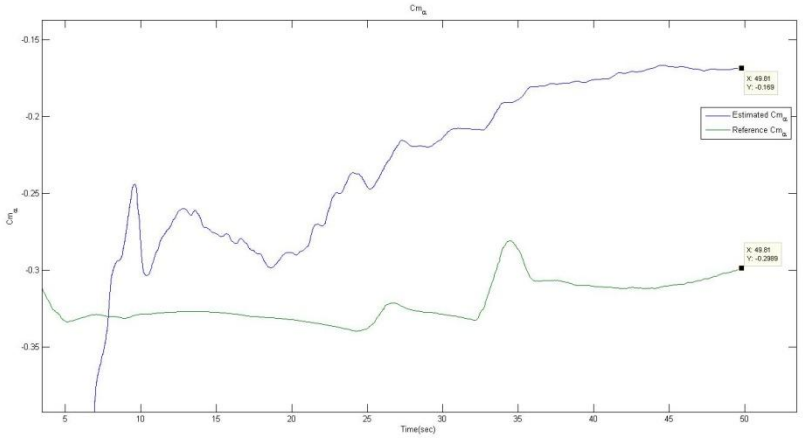

Figure 1-3: Computed vs. FTR estimated $C_{m\alpha}$

Since the study requires exploring all the potential biases, the range between -100% and 100% was sufficient. However, in order to explore the robustness limits of the derivatives, the 100% range was extended to 500%. With respect to determining a delay range, it was harder because neither the literature review nor the simulation succeeded to provide a good pattern. Therefore, this study includes delays in the range from a half second up to an extreme value of forty seconds.

The explored parameters selected for this study are a combination of nine stability and control derivatives over the three flight directions, Longitudinal, Lateral, and Directional. The stability and control derivatives are non-dimensional coefficients, where a stability derivative measures how much

change occurs in a force or moment acting on the vehicle when there is a small change in a flight state such as angle of attack. While a control derivative measures how much change occurs in a force or moment acting on the vehicle when there is a small change in the deflection of a control surface such as the ailerons, elevator, and rudder.

The derivatives are expressed as: $C_{FM_x} = \frac{\delta FM}{\delta x}$

Where FM represents the forces and moments acting on the aircraft: [(forces along X-axis) F_x, (forces along Y-axis) F_y, (forces along Z-axis) F_z, (rolling moment) L, (pitching moment) M, (yawning moment) N].

The aircraft states and controls are denoted by x: [angle of attack (α), pitch rate (q), sideslip angle (β), roll rate (p), yaw rate (r), stabilator deflection (δ_s), aileron deflection (δ_a), rudder deflection (δ_r)]

Aerodynamic derivatives are much more than nine; however, only nine were considered because of their significant effect on the flight dynamics. The nine derivatives are $C_{m\alpha}, C_{mq}, C_{m\delta s}, C_{l\beta}, C_{lp}, C_{l\delta a}, C_{n\beta}, C_{nr}$, and $C_{n\delta r}$. The first three derivatives are longitudinal-wise, the second three derivatives are lateral-wise, and the last three are directional-wise. Moreover, all the selected parameters intentionally excluded the coupling derivatives for their secondary importance. Third, sixth, and ninth derivatives are control derivatives, while others are stability derivatives.

2. Objective

The objective of this research is to analyze the effect of the parameter estimation quality on the performance of the hybrid controller. The real estimator takes time until the estimated parameters converge and the values at which they converge may be inaccurate. Therefore, studying the effect of bias and delay in estimated parameters is very important. In order to analyze the effect of PID issues on the tracking quality of the controller, different delay values and biases were added to the perfect estimates obtained from the direct computation. The errors in tracking the three angular rates and the exerted neural networks effort at each case were used for evaluation.

Additionally, this study compares between the performance of the adaptive controller when the derivatives are updated and the performance when the derivatives are fixed.

3. Thesis Outline

This thesis is organized as follows: Chapter 2 presents the related work that has been developed by other researchers. Chapter 3 is dedicated to explain the technical concept of adaptive control laws and

parameter estimation techniques. The simulation environment and the experimental setup are described in Chapter 4. Chapter 5 presents the final testing results for the sensitivity analysis of online parameter estimation. A conclusion of the analysis and recommendations for future works are provided in Chapter 6.

Chapter 2. Literature Review

This section provides a basic literature review covering the development efforts, problems, and currently undergoing research in the area of fault-tolerant control, which is a cutting-edge topic for the control of aerospace systems. The primary focus will be on adaptive flight control system, which is a class of the fault tolerant control. Adaptive control has three main categories indirect, direct, and hybrid adaptive techniques. [5, 6, 7, 8, 9] Direct adaptive approaches utilize different techniques, such as neural networks, to directly update the controller parameters without an explicit knowledge of the plant parameters. [3, 9, 10, 11] Indirect adaptive approaches, on the other hand, reconfigure the controller structure by using real-time PID techniques to explicitly estimate the plant parameters. [12, 13, 14] Recently, a hybrid technique was introduced, which combines direct with indirect adaptation seeking for better tracking performance.

The aircraft control systems are much more complicated than most other control systems primarily due to the wide range of operational conditions and the complexity of the plant with highly nonlinear characteristics [15, 16]. Therefore, traditional linear feedback control designs may not be sufficient to obtain a satisfactory performance or even maintain stability throughout the flight envelope, especially in the presence of subsystem failures. In order to overcome the limitations of classic control methodologies, new techniques are being developed, which are capable of tolerating failures while still maintaining desirable and robust performance and stability properties. Aforementioned facts encouraged research in fault-tolerant control techniques. Several factors determine the restrictions when designing a fault tolerant control system such as cost, robust stability, the system behavior uncertainty, and the computational limitation.

Flight control design was traditionally based on a gain scheduling approach by dividing flight space into linear-subspaces [17, 18, 19]. Conceptually, gain scheduling is simple and has been proven successful; however, it does not guarantee stability and fulfillment of desired handling qualities after failure. Therefore, several approaches have been investigated in the attempt to enhance tracking quality including adaptive control, such as neural network adaptive controllers [20], nonlinear control such as feedback linearization [21], optimal control [22], and robust control [23]. [24] The robustness of different adaptive control laws was investigated, and the results showed that the baseline dynamic inversion controller was reasonably robust. In addition, the approaches that integrate explicit parameter identification perform the best.

Indirect Adaptive Control System (PID)

During the development process of on-board computational power, [25] Iliff, K.W studied different offline PID techniques, the modified Newton-Raphson technique of minimization, simplified-equations, analog matching, and regression methods.

The Maximum Likelihood method was another popular technique among other various statistical techniques that were used to estimate the stability and control derivatives from flight data that has been extensively conducted as a post flight analysis for several years [26].

Recently great advance in the on-board computational power has allowed the flight control community to consider the application of on-line parameter estimation techniques [3]. The on-line PID attracted attention as a part of fault tolerant control laws for time varying aircraft systems, especially an aircraft subject to drastic changes in aerodynamic characteristics due to potential failures. Two on-line PID techniques, in time domain and in frequency domain, were implemented within the IFCS F-15 program. The performance of the two techniques was investigated. The two methods exhibited similar performance in terms of accuracy estimation, the time needed for convergence, and robustness to noise. However, the frequency domain-based method outperforms the time domain-based method in terms of computational requirements for on-line real time applications.

Fault tolerant techniques typically require PID technology for estimating the values of stability and control derivatives to reconfigure the controller parameters accordingly at different operational conditions. Fourier Transform Regression (FTR) is one of the PID algorithms [4, 27]. The FTR, which is based on the frequency domain, is better than other PID techniques for on-line application. One of the features of the FTR is the on-line calculation of the standard deviations of the estimation error for aerodynamic parameters. Standard deviation can then be used to evaluate the reliability of the estimates prior to feeding these values to the control laws. FTR estimates are not exact; the error is around 30%, which will degrade further after failure.

The F-15 intelligent flight control system (IFCS) research team at NASA has used in early studies a neural network to estimate stability and control derivatives for the control laws [28]. The real-time gains of the control laws were computed by solving the Riccati equation. The research progress on this controller went through three phases, First Phase: developing a pre-trained neural network to store and recall the wind-tunnel-based stability and control derivatives of the vehicle. Second Phase: developing a neural network that can learn how to adjust the stability and control derivatives to account for failures or modeling incompleteness. Third Phase: developing a flight control system that employs the neural network outputs in controlling the aircraft. The team later developed a controller, which relies on the "Stochastic Feedforward and Feedback Technique (SOFFT controller)". This controller, known as "Generation 1", integrates the neural network from first phase to the third phase.

PID was integrated with the SOFFT controller "Gen1", [Figure 2-1], to replace the pre-trained neural network in the previous design [27]. PID scheme's purpose is to estimate and update state-space system matrices of aircraft to the SOFFT controller. two PID approaches were compared; First approach, directly identify the state-space system matrices (matrix approach). Second approach, evaluate the dimensionless stability and control derivatives (derivatives approach), then use those derivatives to build the system matrices. The PID estimates and the standard deviation of the estimation error must first converge within a user-preselected range before they are fed to the controller.

The previous controller showed good results at nominal conditions [10, 29]; however, the controller was not very successful after failure due to PID quality degradation. Therefore, further development was considered by implementing same PID scheme along with an on-line learning neural network for updating aerodynamic coefficients at post failure conditions. [29] The performance of two different neural networks, known as EMRAN (this is the NN used for this research effort) and DCS, was compared. The nonlinear inversion model adaptive controller outperforms the SOFFT controller.

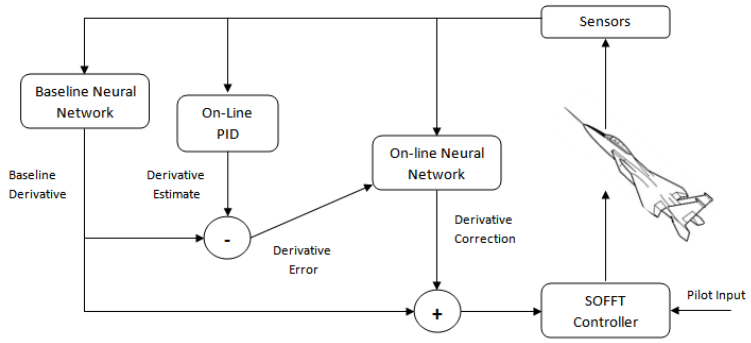

Figure 2-1: Architecture of the IFCS F-15 SOFFT "Generation 1" controller

This research team implemented on-line PID in a fighter jet at subsonic flight conditions to estimate the mathematical model of an aircraft after damage in a primary control surface [30]. The PID then feeds the mathematical model at post-failure conditions to a failure accommodation scheme to compute a compensating control signal to adjust the remaining intact control surfaces for a safe continuation or termination of the flight. The results showed the importance of conducting an 'ad-hoc' small amplitude and short-duration PID maneuver immediately following a positive failure detection to enhance the reliability of the on-line estimated parameters used in the accommodation scheme.

Direct Adaptive Control System

Patricia Melin, discussed using an adaptive controller blending neural networks [31], fuzzy logic and fractal theory. She introduced the use of this controller with nonlinear aircraft dynamic systems. The simulation results showed the efficiency of this scheme in adjusting instability behavior in aircraft systems [Figure 2-2].

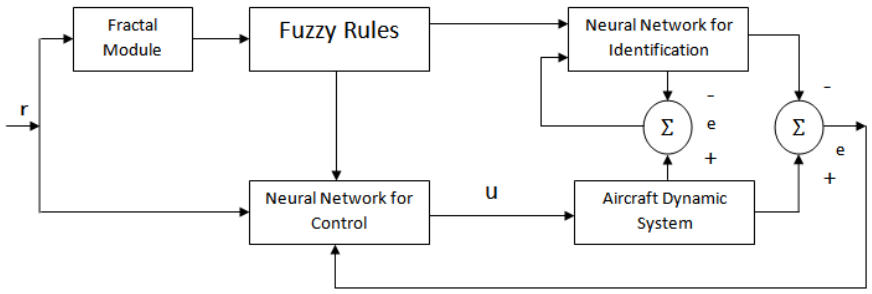

Figure 2-2: Architecture of adaptive neuro-fuzzy-fractal control

A fault-tolerant controller that features a failure detection [32], identification, and reconfiguration (FDIR) is developed. The adaptable controller maintains the desired closed-loop performance; moreover, robustness of the system was validated under a large number of different failure cases. [3] After failure, fault tolerant flight control is required to perform failure detection, identification, and then adaptation.

Similar adaptive control laws were integrated in the flight control system of F/A-18C [16, 33, 34]. This adaptive technique achieves reconfiguration by augmenting the pilot commands rather than the actuator commands. The flight tests show that this technique is effective in restoring acceptable flying qualities following a change in the aircraft dynamics. [15, 30] Implemented failure accommodation strategies using a variety of control surfaces (speed brakes, wing flaps, differential dihedral canards, spoilers, etc.) and thrust mechanisms (differential thrust, thrust vectoring).

The NASA F-15 IFCS team has next developed a direct adaptive neural network-based controller [7]. The objective was to optimize the performance of the aircraft under nominal conditions and to recover the aircraft control after failure occurrence. Failure conditions include actuator failure, sensor failure, structure damage, and adverse flight conditions. This NASA controller is called "Generation 2", as shown in Figure 2-3. Burken presented some changes in the structure of the basic direct adaptive controller "Gen 2" in order to improve the performance under unstable failure flight conditions. The results showed the advantage of the enhanced controller, called "Gen2a", in reducing the occurrence of pilot-induced oscillations and increase system robustness after failure. [6] Using a reference model in the controller "Gen2" to generate command inputs helped to achieve the desired handling qualities.

Testing the dynamic inversion controller with locked stabilator showed significant lateral acceleration and angle of sideslip excursions resulting from lateral stick inputs [35]. The neural network was not able to adjust this behavior. In order to solve this problem, the research controller was modified by combining dynamic inversion in the longitudinal and lateral axes with a classical controller in the directional axis. This modification succeeded in obtaining reasonable handling qualities in the presence of the simulated failure. The results during a stabilator failure demonstrate how the neural network was

able to assist the controller to return the vehicle to nominal flight with lower normal acceleration transients and in a short time.

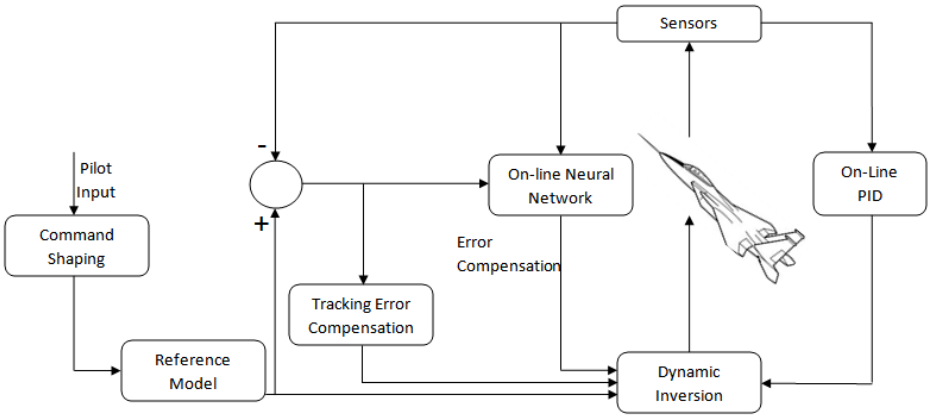

Figure 2-3: Architecture of the IFCS F-15 direct adaptive "Generation 2" controller

The developed direct adaptive controller was based on a non-linear dynamic inversion (NLDI) scheme augmented with a neural network (NN) to compensate inversion errors and changes in aircraft dynamics at post failure occurrence [36]. The performance of three different neural networks (the Extended Minimal Resource Allocating neural network (EMRAN), the Single Hidden Layer neural network (SHL), and the Sigma-Pi neural network was compared in augmenting NLDI scheme inside the direct adaptive controller. The comparison was in terms of achieved handling quality at different failure cases. The simulation results exhibited promising performance for all three neural networks; however, EMRAN (the NN selected for this research) slightly outperformed the other algorithms in terms of the angular rates tracking errors, while requiring a lower computational effort.

Direct adaptive controller “Gen2” exhibited better performance as compared to the SOFFT controller “Gen1” in terms of trajectory tracking as well as pilot workload [27]. Moreover, this study presented the effect of integrating a pre-trained neural network in “Gen2” to provide update values of the stability and control derivatives required by the dynamic inversion.

Before integrating neural network algorithms as an essential part of the adaptive controller, various studies were conducted to show their capabilities, drawbacks, operation conditions, and application [37]. Neural network needs to work in real-time as a part of the flight control system and it has to fit in the flying computer system of the airplane. [38] Neural Networks were used to approximate main relationships of the aircraft dynamic in order to provide virtual measurements capable of replacing real sensors in case of a sensor failure.

Flight control related applications have increasingly integrated Neural Networks [39]. Such as, overcoming issues associated with gain-scheduled control systems [40], the approximation of inverse

dynamics for feedback linearization control of fighter aircraft [41], Unmanned Aerial Vehicle [17], the adaptive control of air-to-air missiles [42], and Neural Adaptive Control for Bank-to-Turn Missiles to adaptively compensate for roll-induced cross coupling [39]. The ability of neural networks in modeling nonlinear systems [39], has been exploited in direct adaptive of a nonlinear controller, as in inverse model control [43].

Hybrid Adaptive Control

Hybrid adaptive controller was introduced to improve the tracking performance of the direct adaptive controller [13]. The investigated hybrid controller combined the indirect adaptive technique from "Gen1" with direct adaptive algorithm from "Gen2". The hybrid controller exploits both PID and NN in conjunction with the NLDI model. PID estimates stability and control derivatives to update B inverse matrix inside the NLDI to reduce the tracking error, and the neural network directly compensates for linearization modeling errors of the NLDI. Two parameter estimation approaches are discussed: a Lyapunov-based indirect adaptive law that uses the tracking error, and a recursive least-squares indirect adaptive law that uses modeling error for adaptation. The adaptive gains are found to be dependent on the learning rate. High-gain learning is analyzed using the root locus analysis of the closed-loop poles of the adaptive control. High-gain learning results in high-frequency oscillations in Lyapunov-based adaptive signals. The hybrid recursive least-squares adaptive law managed to reduce the adverse effect of high-gain learning.

Joshi, discusses another hybrid controller structure utilizing the application of parameter estimation [44]. Tracking problems of direct model reference adaptive control (MRAC) increased during abnormal changes in the plant structure. Hence, direct adaptation of state-feedback gains for state tracking alone was not enough and needed to be combined with estimation of the plant-reference model mismatch. Due to the mismatch, the plant can no longer track the state of the original reference model, but may be able to track a new reference model that still provides satisfactory performance. The reference model is replaced if the estimated plant-model mismatch exceeds a threshold that was determined via robust stability and performance criteria. The resulting controller is a type of hybrid direct-indirect adaptive controllers, which showed good performance in state tracking during the existence of plant-model mismatch as well as parameter deviations.

Although the hybrid controller showed a superior performance, the controller stability is challenging to assess [12, 13]. Traditional phase and gain margins are not applicable for analyzing stability margins of nonlinear adaptive controllers; nevertheless, stability margins are necessary to determine robustness of control laws in the presence of system uncertainties. Therefore, authorizing adaptive control for flight systems is challenging due to the lack of stability standards for adaptive control. Understanding stability issues of adaptive controllers is crucial for further progress on adaptive control technologies. Bounded linear stability and convergence of nonlinear hybrid adaptive control are analyzed using an approximate linear equivalent system. Stability of adaptive flight control was assessed by extending the robust control concept of phase and gain margins to adaptive control laws.

Stability margin analysis shows that large adaptive gains reduce the phase margin [12, 13]. This method can enable metrics-driven adaptive control, where the adaptive gain is adjusted to meet stability margin requirements. The simulation shows that the metrics-driven hybrid adaptive controller has better tracking performance than the basic adaptive controller.

In conclusion, the above represented literature review shows evidently that hybrid adaptive control laws exhibit superior performance as compared to other techniques such as gain scheduling, LQR optimal controller, and direct adaptive controllers. However, the required integrated PID schemes have some issues regarding the convergence delay and the convergence to inaccurate values. Based on this conclusion, an investigation of these two parameter estimation issues has been performed within this research effort. Their effects on the hybrid controller performance have been analyzed and evaluated through a sensitivity study. The outcomes of the study are expected to contribute to a better understanding of the operation of PID schemes within hybrid adaptive control laws and provide tools for design, testing, and evaluation of fault tolerant control laws.

Chapter 3. Technical Background

The following technical discussion has been built based on references [14, 27, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, and 57]

1. Technical Background for Adaptive Control

The word Adaptive Control has started being used as early as 1950. According to Merriam-Webster dictionary, Adapt means to make fit often by modification according to changing circumstances. One of the earliest applications that stimulated the interest in adaptive control was the design of autopilots for high-performance aircrafts.

An adaptive control system is a controller with changeable parameters to adapt accordingly to changes in the controlled system parameters and/or general operational conditions. Although gain scheduling was considered a kind of adaptive control in some books, that cannot be very accurate because an adaptive system has to utilize artificial intelligence techniques such as neural networks. Designing a controller that is capable of maintaining desirable behavior of an aircraft regardless of condition changes is a very challenging mission.

Many flight adaptive control techniques have been investigated over a long time starting from the very primitive adaptive control known as gain scheduling. This basic technique considered designing a controller that functions only around a reference condition. Furthermore, this controller's parameters are updated according to lookup tables that contain information of the entire flight envelop.

Aircraft operates over a wide range of altitudes and speeds; Moreover, aircraft system is highly nonlinear and time varying. However, a linearized aircraft model at different operation points can be used in order to simplify the design process of a suitable controller.

For example, for an operating point (i), the linear aircraft model has the following form:

$$\dot{x} = A_i x + B_i u;\ x(0) = x_0 \qquad (3.1)$$
$$y = C_i x + D_i u \qquad (3.2)$$

Model I

Where A_i, B_i, C_i, and D_i are functions of the operating point (i). As the aircraft flies through different flight conditions, the system matrices will change. Here comes the advantage of adaptation when the controller gains can be adjusted to match the new operation condition and be capable of providing consistent handling qualities.

The controller structure consists of a feedback loop and a controller with adjustable parameters, as shown in Figure 3-1. The way of changing the controller parameters in response to changes in the plant and disturbance dynamics distinguishes one adaptive scheme from another.

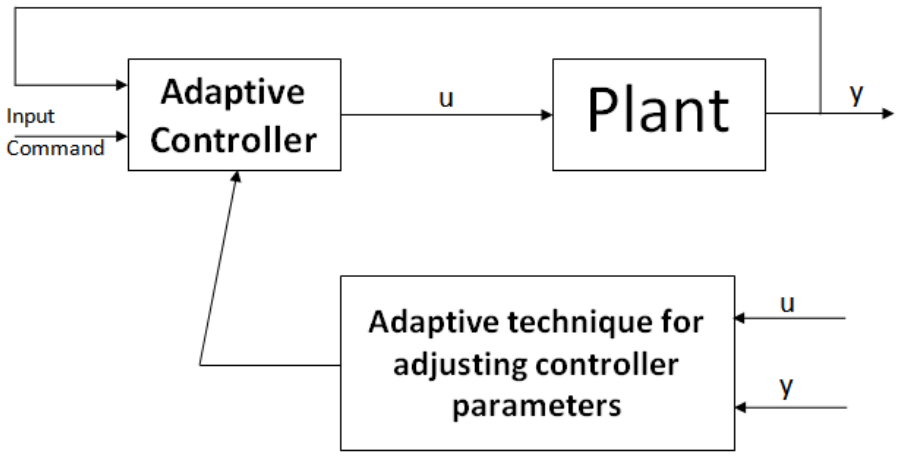

Figure 3-1: Controller structure with adjustable controller parameters

Gain Scheduling

For the afore-presented aircraft model (I), which operates over different operating points i, where $i = 1, 2, \ldots i \ldots, N$. The aircraft parameters change is represented as $Ai, Bi, Ci,$ and Di. Due to the adaptation concept behind gain-scheduling design, this section extends more on this primitive adaptive technique.

The aircraft system is originally a non-linear system, but it can be linearized around each of the operating point. This is achieved by designing multiple controllers that can meet the performance requirements at each of the potential operating points.

The mentioned controller is a traditional feedback controller; the only difference in this case is constant multiple gains to be designed at each operating point. This leads to a controller C (θ) with a set of gains $\{\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_N\}$ covering N operating points [Figure 3-2].

On board, once an operating point is detected, the controller gain changes to the corresponding gain θ, which is obtained from the precopmuted set of gains. Significant parameters changes between designed operating points are handled by interpolation; however, designing more operating points is favorable to obtain better handling qualities. This design technique relies on having a look-up table that stores all the gain values θ_i, which will be called when corresponding operating point is detected. Furthermore, the system needs auxiliary measurements that can be used to detect change in operating points. For aircraft, the auxiliary measurements are Mach number and the dynamic pressure (or Altitude).

This approach adapts the controller for the aircraft parameter variations by changing the controller gains as functions of the auxiliary measurements. The advantage of gain scheduling is that the controller gains can be adapted as quickly as the auxiliary measurements respond to parameter changes.

On the other hand, the controller gains are precomputed off-line; therefore, operation outside the designed flight envelop and/or unpredictable changes in the plant dynamics will lead to deterioration in the performance. Another possible drawback of gain scheduling is the high design and implementation costs that increase with the number of operating points.

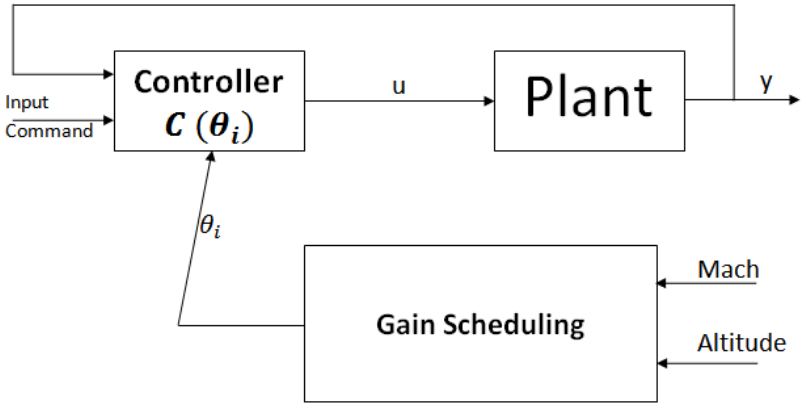

Figure 3-2: Gain scheduling structure

Gain scheduling, despite its limitations, is still a popular adaptation technique for handling parameter variations in flight control.

Direct and Indirect adaptive Control

The adaptive controller design, theoretically, contains an on-line parameter identification scheme to estimate the unknown plant parameters, then update the estimated parameters to control laws. Moreover, adaptive techniques have two categories based on the method in which the parameter estimator is connected to the control laws. The practical adaptive laws implemented in this document were inspired from the two theoretical approaches after introducing some modification, as presented at the end of this section.

1- Indirect adaptive control

The plant parameters are explicitly estimated on-line and used to calculate the controller parameters. The scheme outline is [Figure 3-3, [49]]:

a. The plant model $P(\theta^*)$ is defined in terms of some unknown parameter vector θ^*.
b. An on-line parameter estimator generates an estimate $\theta(t)$ of θ^* at each time t by processing the plant input u and output y.
c. The parameter estimate $\theta(t)$ is used to form an estimated plant model $\hat{P}(\theta(t))$.
d. This estimated model is then used in the design process of the controller parameters or gain vector $\theta_c(t)$, which is found by solving a certain algebraic equation: $\theta_c(t) = F(\theta(t))$ at each time t.

e. $C\ (\theta_c(t))$ is designed at each time t to satisfy the performance requirements for the estimated plant model $\hat{P}(\theta(t))$, which is not necessarily identical to the unknown plant model $P(\theta^*)$.

In short, the main goal when designing an indirect adaptive controller is to select the control laws $C\ (\theta_c)$ and the appropriate parameter estimators that generate $\theta\ (t)$, as well as the algebraic equation $\theta_c(t) = F\ (\theta(t))$ so that $C\ (\theta_c(t))$ meets the performance requirements for the plant model $P\ (\theta^*)$ with unknown θ^*.

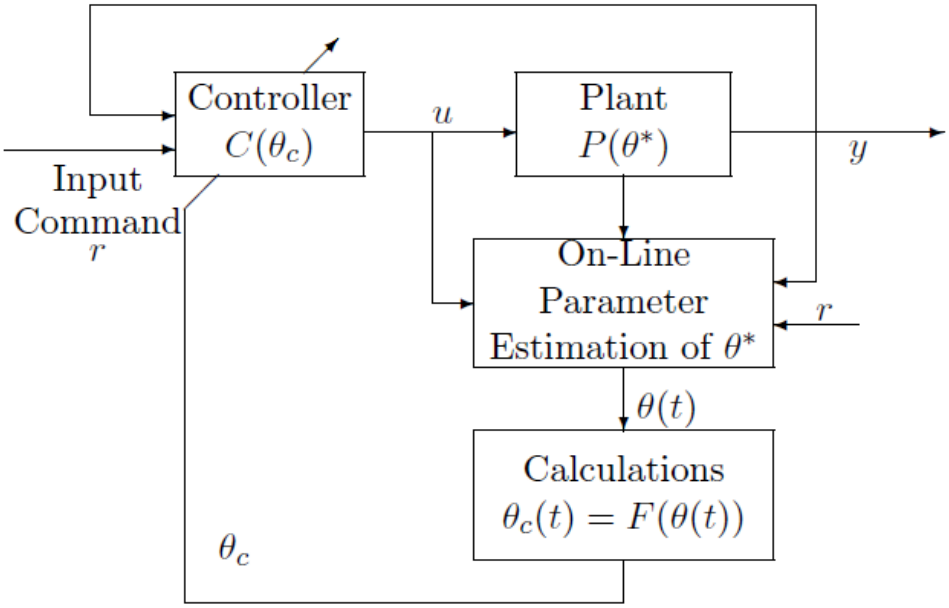

Figure 3-3: Indirect adaptive controller structure

2- direct adaptive control

The controller gains, which are expressed in terms of the plant model parameters, are estimated directly without explicit knowledge of the plant parameters. Therefore, this approach does not involve intermediate calculations for the plant parameter estimates.
The scheme outline is [Figure 3-4, [49]]:

a. The plant model $P\ (\theta^*)$ is expressed in terms of the unknown controller gains vector $\theta_c{}^*$, for which $C\ (\theta_c{}^*)$ meets the performance requirements for the plant model $P_c\ (\theta_c{}^*)$.
b. The on-line parameter estimator is designed based on $P_c\ (\theta_c{}^*)$ instead of $P\ (\theta^*)$ to provide direct estimates $\theta_c(t)$ of $\theta_c{}^*$ at each time t by processing the plant input u and output y.
c. The estimate $\theta_c(t)$ is then used to update the controller gains vector θ_c without intermediate calculations.

In short, the main problem in direct adaptive control is to select the control laws $C(\theta_c)$ as well as the suitable parameter estimator that generates $\theta_c(t)$. $C(\theta_c)$ must meet the performance requirements for the plant model $P_c(\theta_c{}^*)$.

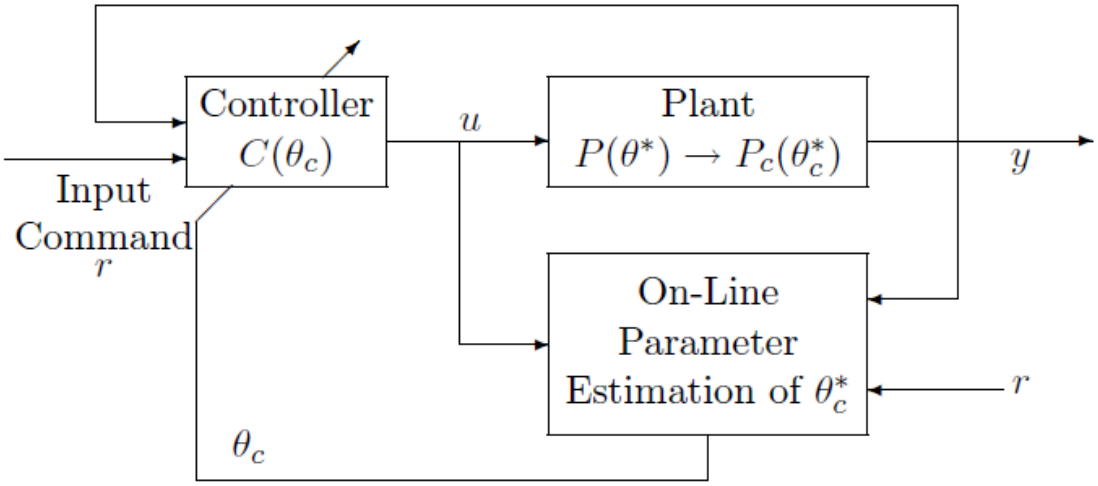

Figure 3-4: Direct adaptive controller structure

As mentioned earlier, both represented techniques are the cornerstone of the discussed hybrid adaptive controller in this research. The first adaptive algorithm discussed above, indirect adaptive, is the concept behind integrating PID schemes to indirectly estimate the system parameters and update the controller correspondingly, such as SOFFT controller (Gen1). The second algorithm, direct adaptive, is the concept behind directly reconfigure the controller structure to adapt for system changes (Gen2). Both algorithms can be combined to build the hybrid adaptive controller, which is the particular controller used for this study.

2. Technical Background for Parameter Estimation

Parameter estimation can be utilized either online, when the system is in operation, or offline, post-operation. From the viewpoint of this study, online estimation is implemented as a part of the hybrid adaptive controller to indirectly compensate for the aircraft parameter change by updating the controller parameters. On the other hand, offline estimation process is popularly used for modeling systems especially when the nature of the physical system is hard to predict. Moreover, this technique is used to analyze the performance of a system using measured data history.

I. Offline Parameter Estimation

A. Aircraft Parameter Modeling

Mathematical models of physical systems are necessary to predict future response; or in other words, predict the output of a system for its corresponding input. In most cases, experience and knowledge are not enough when modeling a new complex plant. Modeling a plant without having sufficient insight of its physical structure depends on two factors approximations and axioms. First, approximations are made to simplify the complex physical plant. Those approximations will only be acceptable within certain regions and for specific applications. Axioms are logical facts that have not been mathematically proved, but they either make sense or have not been contradicted by former experiments. Offline parameter estimation is utilized for modeling those types of new systems to help deciding the suitable approximations and axioms depending on the data history obtained from experiments (further discussion on this matter in section "B. Aircraft Parameter Identification").

Modeling a flight system is a great example that displays utilizing both axioms and assumptions to create a mathematical model. There are some basic axioms used when building aircraft models, such as isotropic characteristics, i.e. properties of materials are not dependent on direction). With respect to assumptions, for instance, the aircraft vehicle can be safely assumed a rigid body with constant mass and inertia.

Equation of Motions: The resulting mathematical model consists of 12 equations, six dynamic and six kinematic equations:

1- Linear momentum or force equations:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}_B \begin{bmatrix} u \\ v \\ w \end{bmatrix}_B = \frac{1}{m_c} \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix}_B \qquad (3.3)$$

2- Angular momentum or moment equations:

$$\begin{bmatrix}\dot{P}\\ \dot{q}\\ \dot{r}\end{bmatrix}_B + \begin{bmatrix}I_{xx} & 0 & I_{xz}\\ 0 & I_{yy} & 0\\ I_{zx} & 0 & I_{zz}\end{bmatrix}^{-1}_B \begin{bmatrix}0 & -r & q\\ r & 0 & -p\\ -q & p & 0\end{bmatrix}_B \begin{bmatrix}I_{xx} & 0 & I_{xz}\\ 0 & I_{yy} & 0\\ I_{zx} & 0 & I_{zz}\end{bmatrix}_B \begin{bmatrix}p\\ q\\ r\end{bmatrix}_B = \begin{bmatrix}I_{xx} & 0 & I_{xz}\\ 0 & I_{yy} & 0\\ I_{zx} & 0 & I_{zz}\end{bmatrix}^{-1}_B \begin{bmatrix}M_x\\ M_y\\ M_z\end{bmatrix}_B \tag{3.4}$$

3- Kinematic equations of translation:

$$\begin{bmatrix}\dot{x}_E\\ \dot{y}_E\\ \dot{z}_E\end{bmatrix}_E = \begin{bmatrix}\cos(\psi)\cos(\theta) & \cos(\psi)\sin(\theta)\sin(\phi)-\sin(\psi)\cos(\phi) & \cos(\psi)\sin(\theta)\cos(\phi)+\sin(\psi)\sin(\phi)\\ \sin(\psi)\cos(\theta) & \sin(\psi)\sin(\theta)\sin(\phi)+\cos(\psi)\cos(\phi) & \sin(\psi)\sin(\theta)\cos(\phi)-\cos(\psi)\sin(\phi)\\ -\sin(\theta) & \cos(\theta)\sin(\phi) & \cos(\theta)\cos(\phi)\end{bmatrix}\begin{bmatrix}u\\ v\\ w\end{bmatrix}_B \tag{3.5}$$

4- Kinematic equations of rotation:

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta)\\ 0 & \cos(\phi) & -\sin(\phi)\\ 0 & \sin(\phi)/\cos(\theta) & \cos(\phi)/\cos(\theta)\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{3.6}$$

B. Aircraft Parameter Identification

System identification and simulation process are two sides of the same coin. Simulation utilizes the mathematical model to predict an output for every input, while system identification uses inputs and outputs data history to estimate the model. If the model structure is fixed and only model parameters are determined, the aircraft system identification becomes a parameter estimation process.

Aircraft system identification includes the following phases:

1- Model postulation: the model general framework is defined based on prior experience with the system.
2- Experiment design: determine suitable instrumentation to record the data history.
3- Data compatibility analysis: data measurement is subject to error, and this error should be analyzed.
4- Model structure determination: The exact model structure is set based on prior knowledge.
5- Parameter and state estimation: Two classes of methods are currently used for aircraft parameter estimation:
   I. Equation-error method: estimate unknown aerodynamic parameters (stability and control derivatives) by minimizing the sum of squared differences between measured and modeled aerodynamic forces and moments.

II. Output-error methods: parameters estimation is based on minimizing the sum of weighted square differences between the measured and modeled aircraft system outputs.

6- Collinearity diagnostics: no correlation between data used for estimation; otherwise, the results are unreliable.

7- Model validation: new data must be used for this last step.

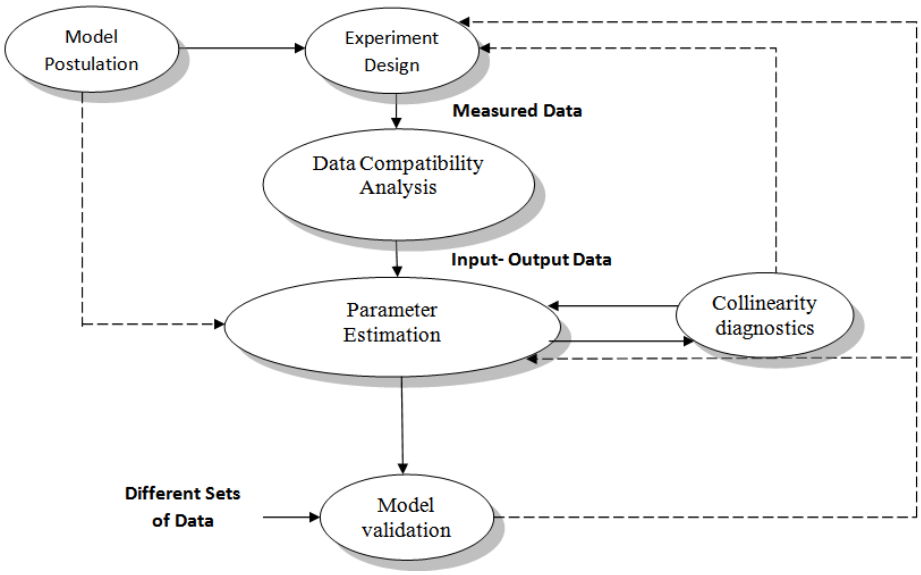

Figure 3-5: Block diagram of aircraft system identification

II. Online Parameter Estimation

A. Hybrid Adaptive

Within this research effort, estimating parameters throughout the operation period is an important part of the implemented hybrid adaptive controller in order to reduce the error generating from leaving the reference condition. The estimation scheme is used to create the inverse B matrix at different operation points, which exists in the nonlinear dynamic inversion mode.

$$u = B^{-1}(\dot{x} - Ax) \qquad (3.7)$$

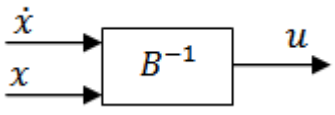

Figure 3-6: LDI block

B. Fault Detection

Fault detection techniques are another application of on-line PID. Each aircraft sub-system failure has unique dynamic fingerprint. Online PID is utilized to estimate particular flight data that will then be compared to the actual aircraft sensor measurements. Depending on the nature and significance

of the difference between the two data sets, a failure may be detected and identified. Two techniques of estimation are used for detecting failure in real-time:

a. States estimation-based or filter-based detection method:
   The states are estimated using filters or observers, such as Kalman filter. Subsequently, the estimates are compared against sensor measurements and the differences determine the detection outcome.
b. Parameter estimation- based detection method:
   The estimated parameters are directly compared against predefined thresholds for different failure types. Similarly, the online PID can be used to create an estimated model, whose output is compared against sensors measurements.

A. Time-Domain Regression:

Linear Regression Least Squares Method:

Least squares method is intended for finding the unknown set of parameters θ. The model can be nonlinear in the parameters $y = \theta(x)$ In this section a linear parameter-wise model is presented with noisy measurements y:

$$y = \theta\, x + \mu \qquad (3.8)$$

y: *is the noisy measured model output*

x: *is the measured model input*

θ: *are the unknown parameters which need to be estimated*

μ: *is the measurement error, which are assumed to be zero − mean and uncorrelated.*

The measured y is a column vector of size N measurement samples. The purpose of using linear regression method is to determine θ; so that, the sum of squared differences between the measurement and the model is minimized:

$$J(\theta) = \frac{1}{2}\,(y - x\,\theta)^T (y - x\,\theta) \qquad (3.9)$$

$\hat{\theta}$ parameters are the necessary estimates to minimize the cost function (J) which can be found through:

$$\frac{\partial J}{\partial \theta}\Big|_{\theta=\hat{\theta}} = \frac{\partial}{\partial \theta}\left[\frac{1}{2}\,(y - x\,\theta)^T (y - x\,\theta)\right]\Big|_{\theta=\hat{\theta}} = 0 \qquad (3.10)$$

By taking the derivative of equation(3.10)

$$-X^T y + X^T X\hat{\theta} = 0 \qquad (3.11)$$

The least-square estimator results are given:

$\hat{\theta} = (X^T X)^{-1} X^T y$ (3.12)

The matrix $X^T X$ is always square and symmetric. If the columns of X are linearly independent then $X^T X$ is invertible. Finally, the covariance matrix of the parameter estimates is found to assess the reliability of the estimation process.

B. Frequency-Domain Regression

Similarly, an estimation algorithm can be implemented in frequency domain instead of time domain. However, Frequency-Domain (FD) parameter identification algorithms slightly outperform Timed-Domain (TD) algorithms, considering that FD needs less data points for parameter estimation and direct relevance to frequency control design techniques.

I. Fourier Transform

The FD algorithm first uses "Fourier Transform" to transform time-domain data into frequency-domain data:

$$\mathcal{F}[x(t)] = \tilde{x}(\omega) = \int_0^T x(t) e^{-j\omega t}\, dt \qquad (3.13)$$

$\mathcal{F}$: $Fourier\ operator$

ω: $The\ angular\ frequency\ rad/s$

The finite interval [0, T] must be wide enough such that includes all the dynamics of the physical system. Aircraft dynamic system as an example, designing the finite interval must count for short period as well as roll mode as they are the fastest dynamic modes in the system.

II. Linear Regression Method

The linear regression algorithm in frequency-domain is very similar to the counterpart algorithm in time-domain. Fourier transforms of all variables are found to obtain a corresponding equation in frequency domain:

$\tilde{y} = \theta\, \tilde{x} + \tilde{\mu}$ (3.14)

$\tilde{y} = \mathcal{F}[y(t)]$: N x 1 vector of Fourier transform of the noisy sensors measurement.

$\tilde{x} = \mathcal{F}[x(t)]$: *N x P* matrix of the Fourier transform of the states measurements.

θ: P *x 1* vector of unknown parameters.

$\tilde{\mu} = \mathcal{F}[\mu(t)]$: N x 1 vector of measurement noise,

Following the same time-domain approach, the least-square estimator in frequency domain becomes:

$$\hat{\theta} = \left(\tilde{X}^T \tilde{X} \right)^{-1} \tilde{X}^T \tilde{y} \quad (3.15)$$

$\tilde{X}^T$ *is the transpose conjugate matrix of* $\tilde{X}$

For practical aircraft system identification, the vector of unknown parameters $\hat{\theta}$ is real; therefore, the estimator equation can be simplified to:

$$\hat{\theta} = \left(RE(\tilde{X}^T \tilde{X}) \right)^{-1} RE(\tilde{X}^T \tilde{y}) \quad (3.16)$$

Finally, the covariance matrix is used to find the variance of the estimation error, which provides an on-line assessment of the accuracy of the estimation to determine the convergence criteria of the parameter estimation process.

3. Implemented Control laws

A) Indirect adaptive control laws:

This part computes stability and control derivatives while the aircraft is moving throughout the flight envelop. Linear equations are used to compute the ideal values of these derivatives. Then those estimates update the NLDI model to reduce the modeling error and subsequently the neural network effort. Computation scheme was used as an ideal replacement of the online PID, frequency domain algorithm-FTR. More details on the parameter estimation background were presented earlier in this section.

B) Direct adaptive control laws:

The control laws follow the model trajectory using a non-linear dynamic inversion model (NLDI) augmented with artificial neural networks (NNs). As part of the NLDI control laws, pilot inputs are first converted into angular rate commands. Then those angular commands are used to produce the desired angular rates and their derivatives through first and second order reference models, such that desired handling qualities are achieved. The tracking error between the model references output and sensors measurements are adjusted using proportional, integral, and derivative tracking controller. Online learning NNs generate augmentation commands to compensate for inversion modeling errors. The reference models output (desired angular rates and their derivatives) along with tracking error compensation and NNs output are exposed to the dynamic inversion model, which will generate the plant inputs. Those generated commands, in the ideal case, will produce the desired angular values as obtained from the reference model, but three types of error will deform the output; the errors are tracking error, modeling error, and aerodynamic changes.

$$\begin{bmatrix} \delta_{a_{com}} \\ \delta_{e_{com}} \\ \delta_{r_{com}} \end{bmatrix} = B^{-1} \begin{bmatrix} \dot{p}_c - L_1 \\ \dot{q}_c - M_1 \\ \dot{r}_c - N_1 \end{bmatrix} \qquad (3.17)$$

$\dot{p}_c, \dot{q}_c, and\ \dot{r}_c$ are desired angular accelerations generated by the reference model. L_1, M_1, and N_1 are the non-linear terms of the moment equations. B is the state-space control matrix computed at departure-reference flight condition. $\delta a_{com}, \delta e_{com}, and\ \delta r_{com}$ are the commands that will be used to compute the actual control surface deflections. The B matrix might also be updated according to aerodynamic parameters change and in this case the control laws upgrade to hybrid adaptive laws. Finally, the type of the neural networks integrated in this controller is EMRAN.

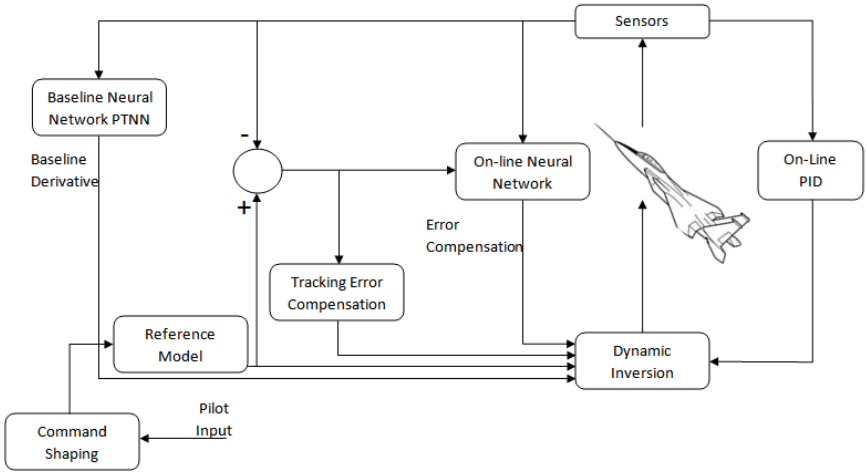

Figure 3-7: Architecture of the IFCS F-15 hybrid adaptive controller

Chapter 4. Overview of Simulation and Experiments

1. Simulation Environment

This section presents the different components of MATLAB/SIMULINK- based environment developed at WVU and then updated as a part of this research effort. This section is organized as follows: the first subsection provides a brief clarification for the parts existed prior to this study, [Figure 4-1]. Second subsection shows the blocks and parts updated for the purpose of this analysis.

A. SIMULINK - Previous Work

The Flight dynamic and Control toolbox (FDC) was used to provide the general framework for solving the equations of motion integrated within the aircraft model [51, 58, 59]. The aircraft model implemented in this simulation is an approximate nonlinear model of F-15 aircraft with canards. Look-up tables are integrated to provide the aerodynamic and thrust characteristics. The simulation environment gives the user three choices in flying the aircraft, joystick, a set of pre-recorded maneuvers history, or a combination of both. To provide graphical environment for the simulation and pilot interaction, the model was interfaced with Aviator Visual Design Simulator (AVDS) [4Figure 4-2].

The previous controller integrated in conjunction with the aircraft model was "Gen2" [51, 58, 59]. The control laws are based on NLDI augmented with NNs to compensate for the inversion error. The NLDI parameters were updated by directly computing the parameters throughout the simulation, [Figure 4-3].

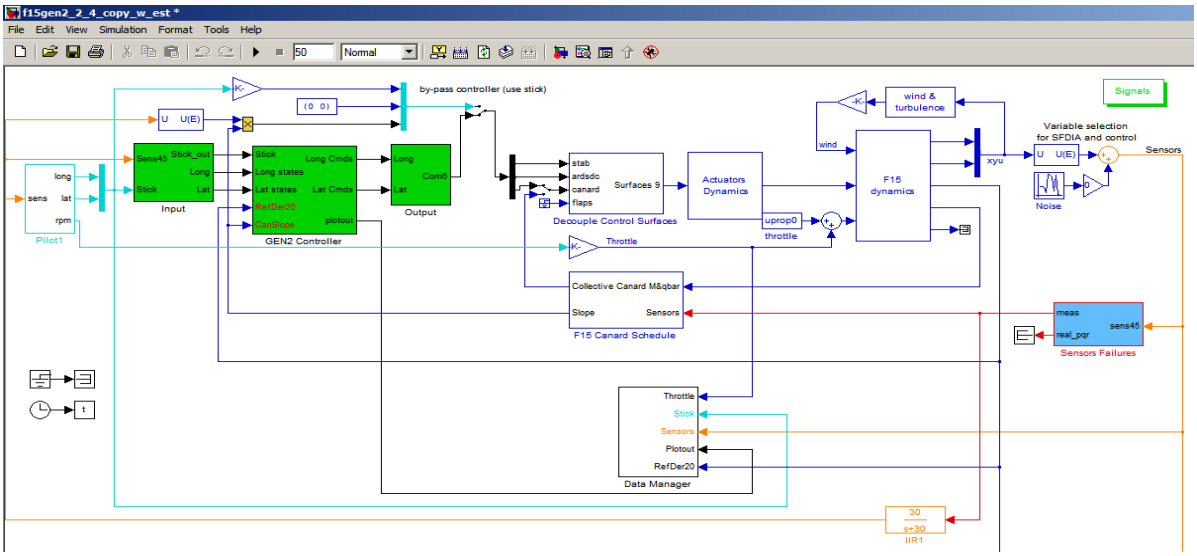

Figure 4-1: Overview of the previous SIMULINK-based simulation environment

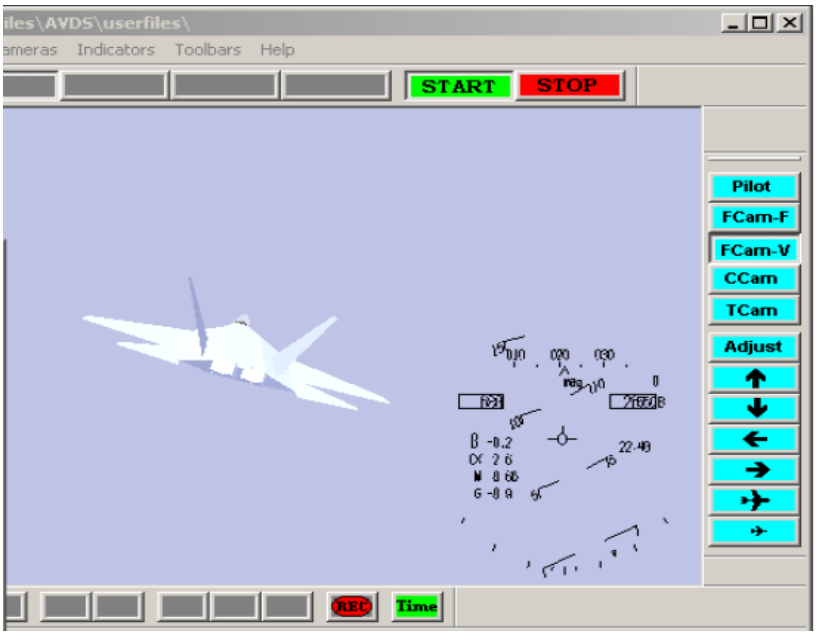

Figure 4-2: Aviator Visual Design Simulator (AVDS)

4

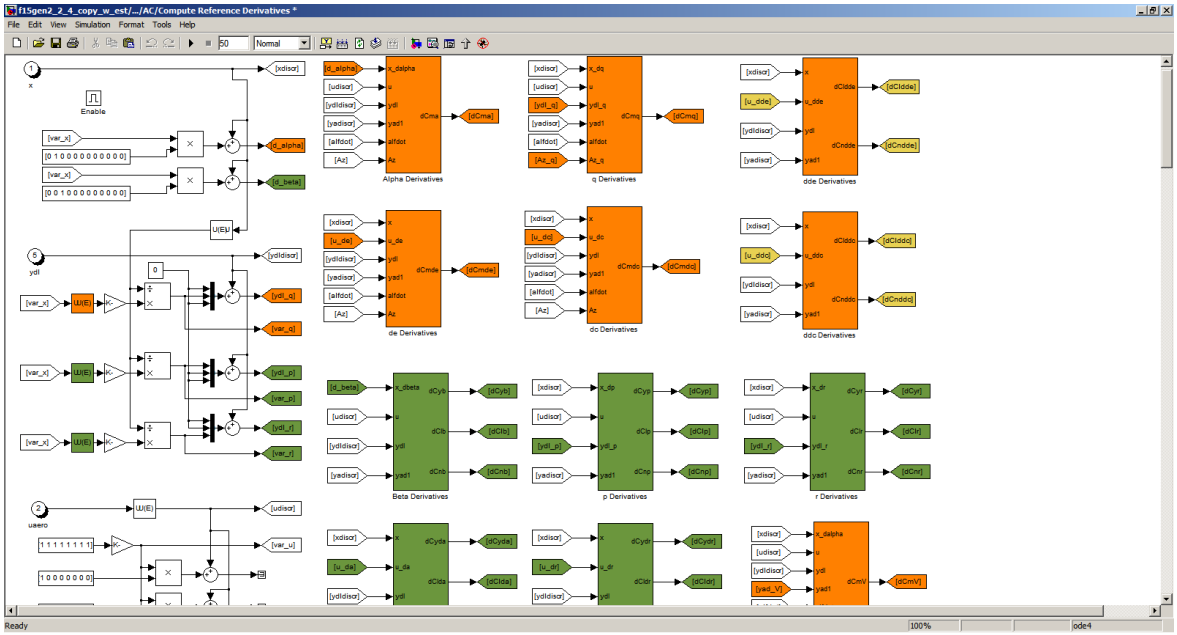

Figure 4-3: Direct computation process of the inversion model parameters

B. SIMULINK - Current Work

This section presents all the new parts added to the previous program as part of this research effort. The new simulation environment includes a real FTR parameter identification scheme as well as the old direct computation model, as shown in Figure 4-6. These two blocks were implemented side to side in the new program in order to provide the user with the flexibility to choose between updating the NLDI model with perfect estimates or real estimates. The implemented PID, as shown in Figure 4-4, uses the sensor measurements to estimate the required parameters.

For the purpose of the sensitivity analysis, the old direct computation scheme was modified to provide the flexibility of adding different levels of delays and biases to a single derivative, to a group of the derivatives, or to all of them before being updated to the controller. In addition, the flexibility to either update or fix a single derivative, a group of the derivatives, or all of them was considered. The modified derivatives include the stability derivatives inside the controller and the control derivatives inside the B-inverse matrix, [Figure 4-8].

Another block named "scopes1" was created for two purposes. First, provide comparing plots between the computed derivatives and the estimated derivatives. Second, sort the estimated derivatives to match the order of the computed derivatives to be updated later to the correct location inside the controller, [Figure 4-7].

In addition to this SIMULINK program, many MATLAB codes were written with different objectives to support the analysis and simulation. A block named "plot", as shown in Figure 4-9, was created to connect between the simulation process and the written codes. This block was programmed to run automatically after the simulation ends, and call the specific MATLAB code according to the explored derivative, [Figure 4-10].

A block named "Data Plots" was created to provide plots of the simulation states, as well as saving them to the work space to be used later by the MATLAB codes. Different MATLAB codes were written for this study, sample of the codes are presented in Appendix-A.

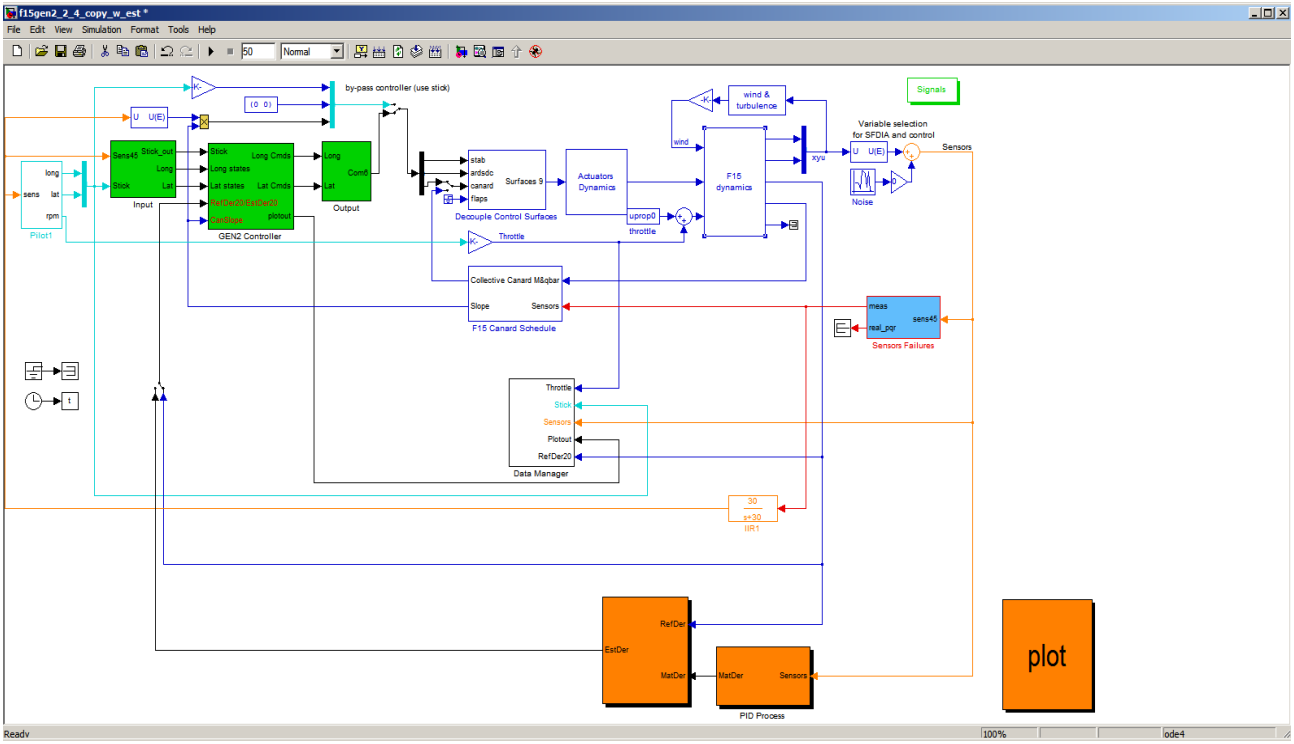

Figure 4-4: Overview of the new SIMULINK-based simulation environment

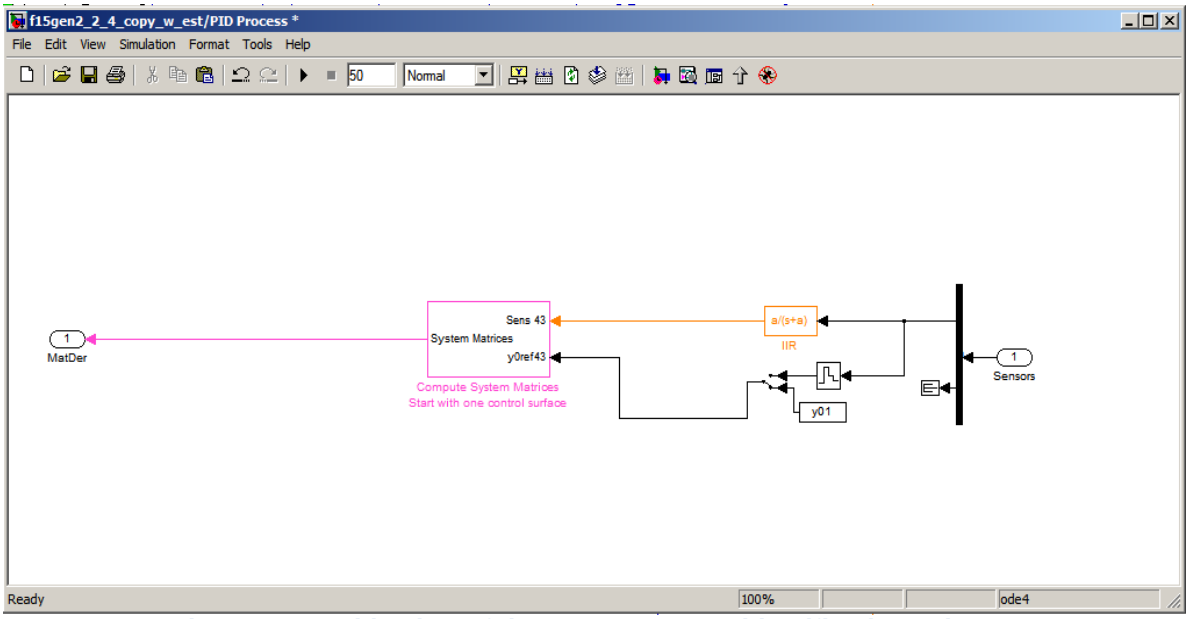

Figure 4-5: Inside-view of the FTR parameter identification scheme

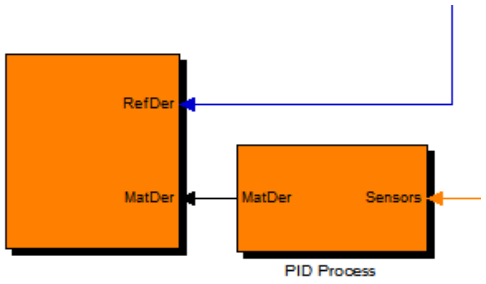

Figure 4-6: FTR estimation vs. computation

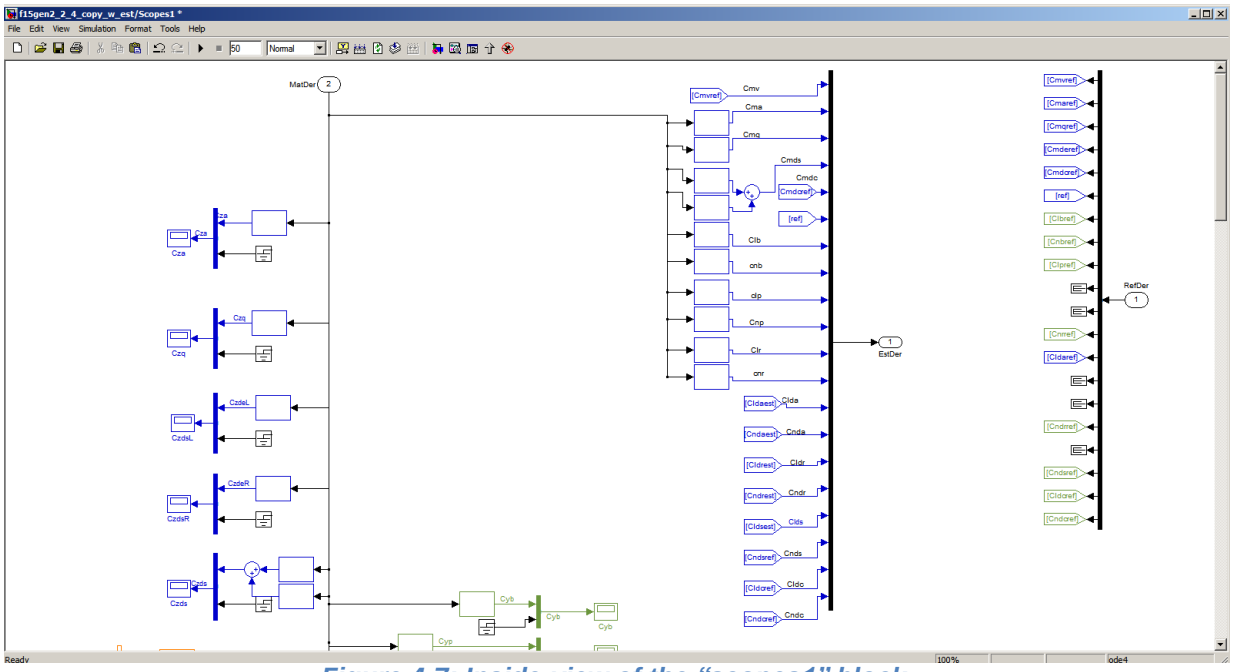

Figure 4-7: Inside-view of the "scopes1" block

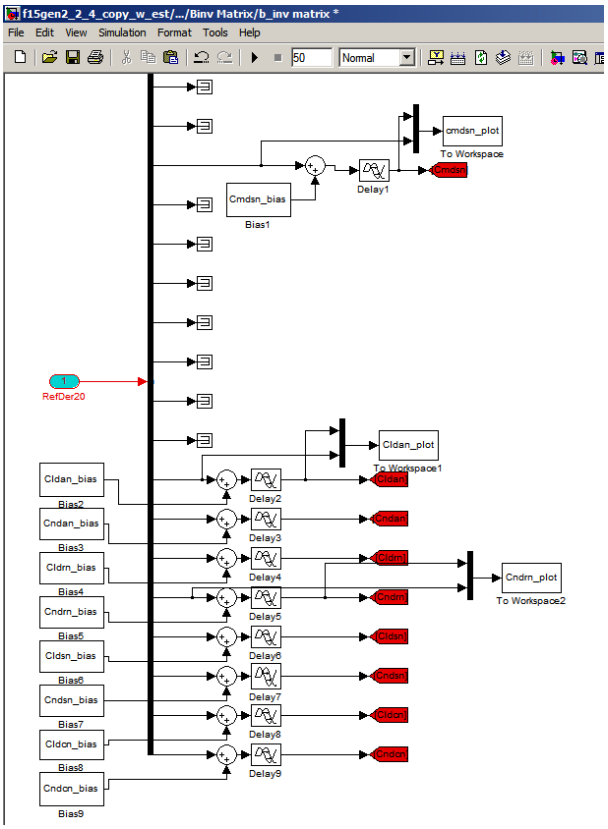

Figure 4-8: Biases and delays Introduced to the derivatives inside the B-inverse matrix

Figure 4-9: A block to call the MATLAB codes

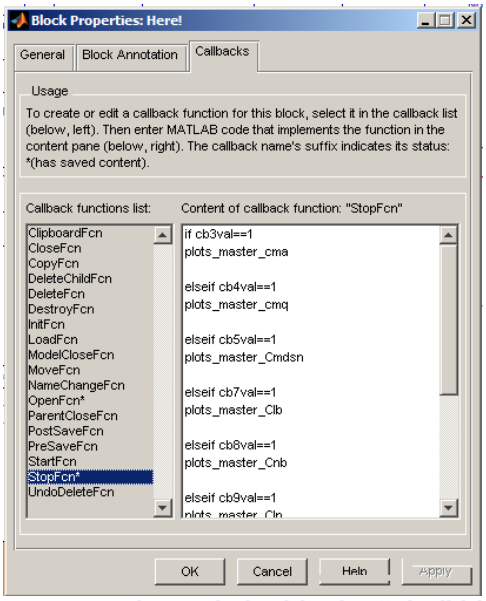

Figure 4-10: The code inside the "plot" block

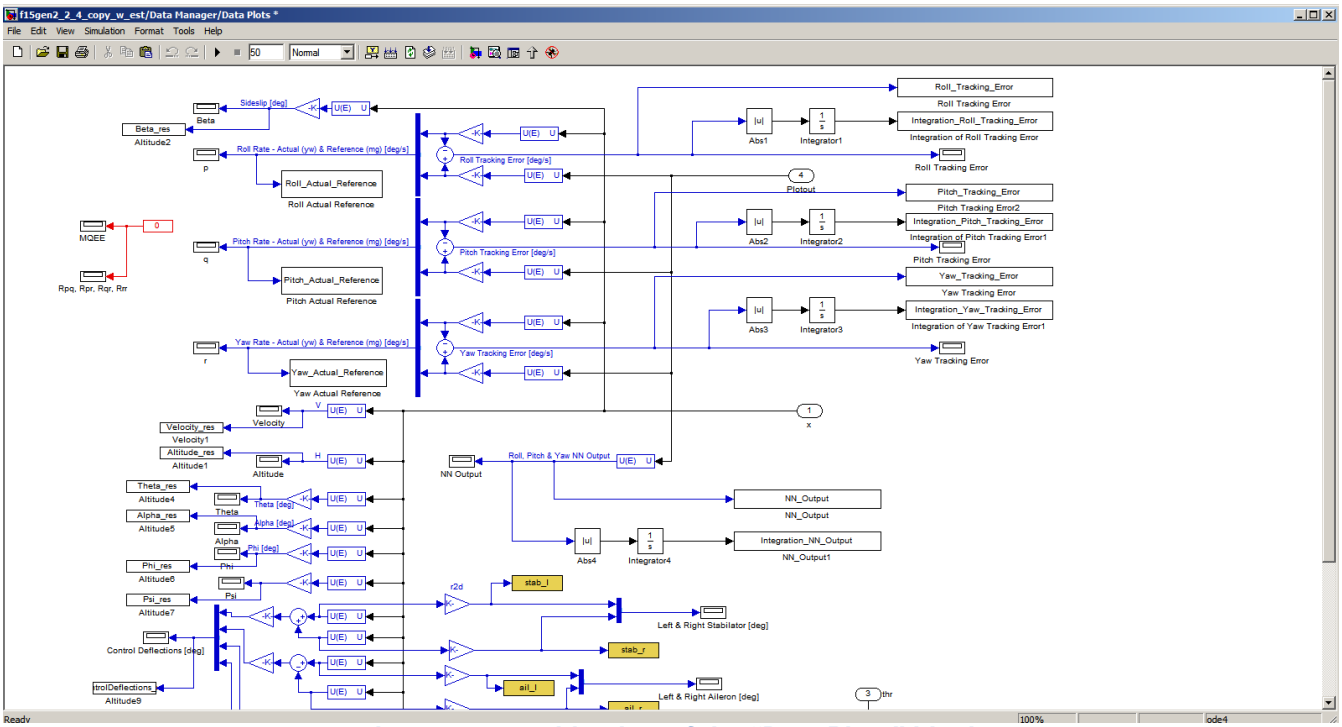

Figure 4-11: Inside-view of the "Data Plots" block

C. Graphical User Interface - Previous Work

This part presents the previous Graphical User Interface (GUI) menus designed prior to this work. These GUIs were built to provide the user with the flexibility to select the desired flight simulation scenario. The first menu, as shown in Figure 4-12, allows the user to select the desired flight and simulation scenarios. Then, the second menu, as shown in Figure 4-13, includes three options for flying the aircraft. The third menu, as shown in Figure 4-14, includes choices between different NNs. Finally in the last menu, the user can select which parameters to be displayed during the simulation, as shown in Figure 4-15.

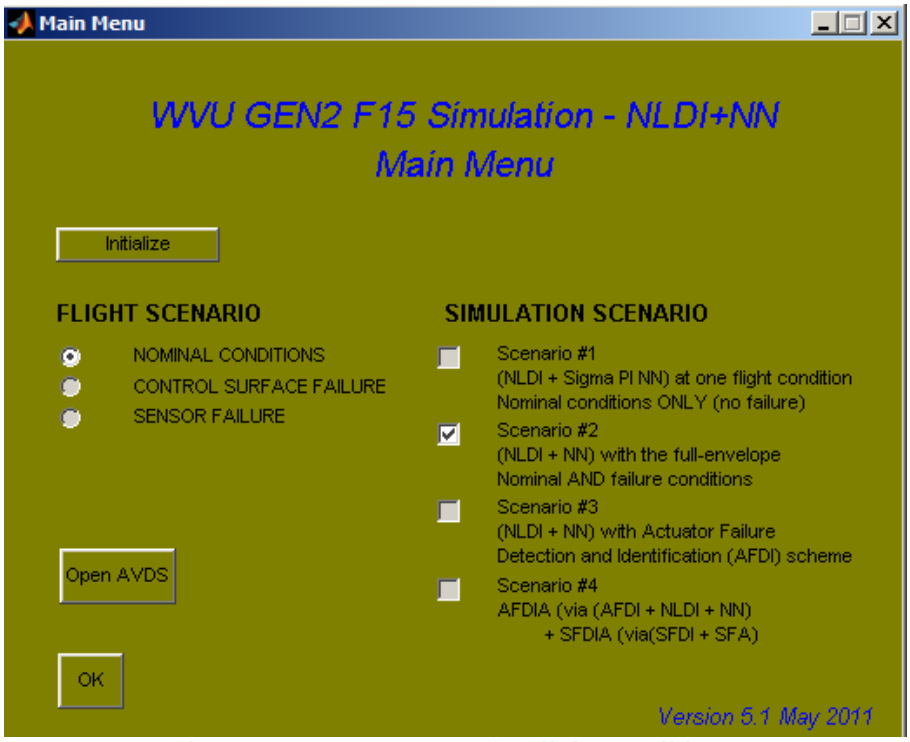

Figure 4-12: Setup menu for flight conditions

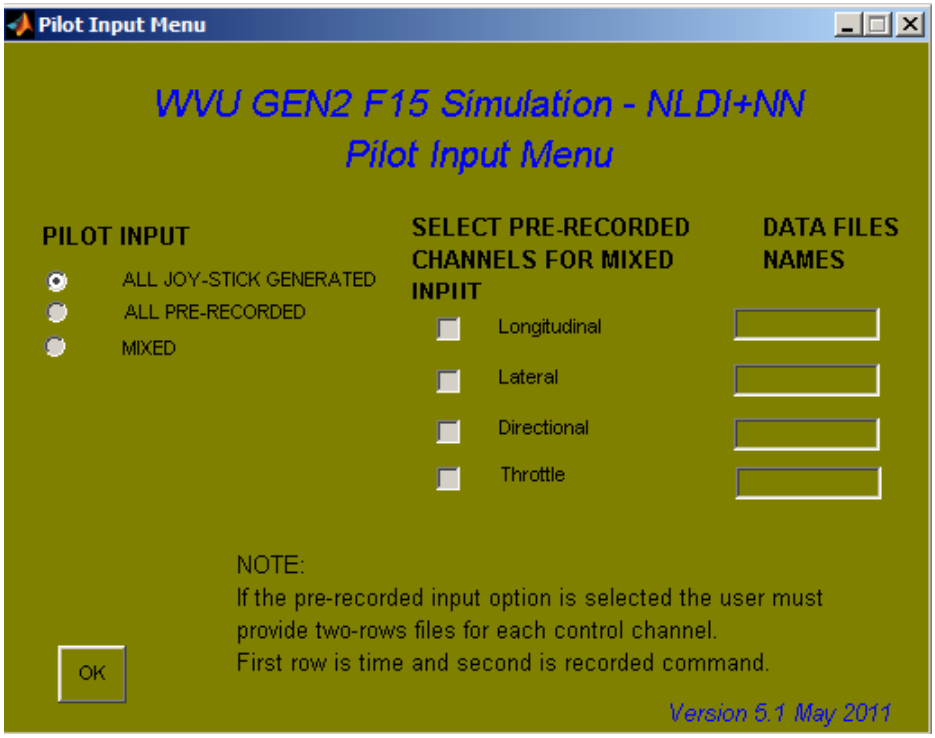

Figure 4-13: Setup menu for pilot input

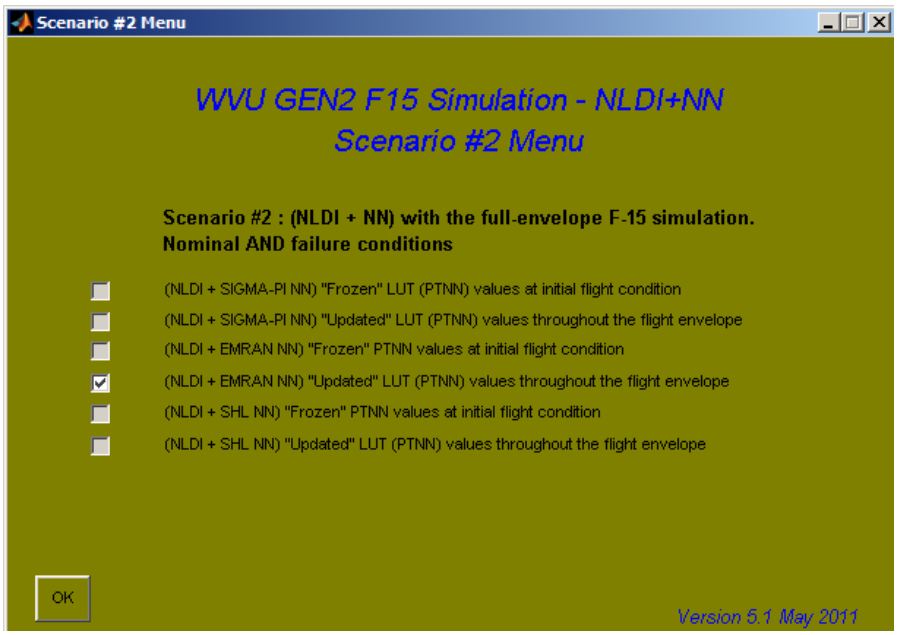

Figure 4-14: Setup menu for adaptation structure

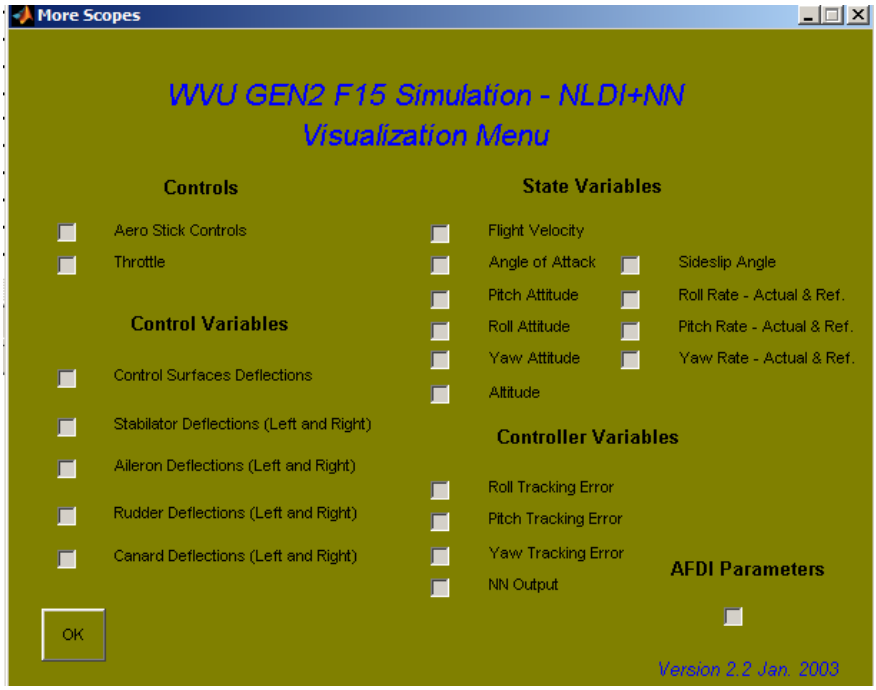

Figure 4-15:Setup menu for flight scopes

D. Graphical User Interface - Current Work

Two new GUIs were designed to support this research. The first GUI menu provides the user with the flexibility to select any combination of delays and biases to be added to the derivatives inside the SIMULINK environment, [Figure 4-16]. The second menu allows the user to display comparing plots between the computed derivatives and their estimated values, as well as the standard deviations of their estimates, [Figure 4-17].

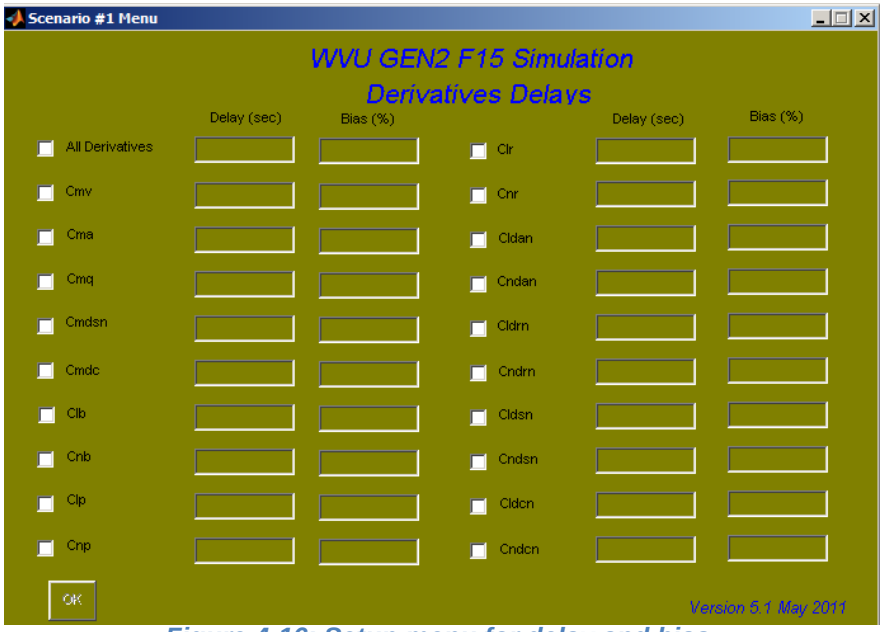

Figure 4-16: Setup menu for delay and bias

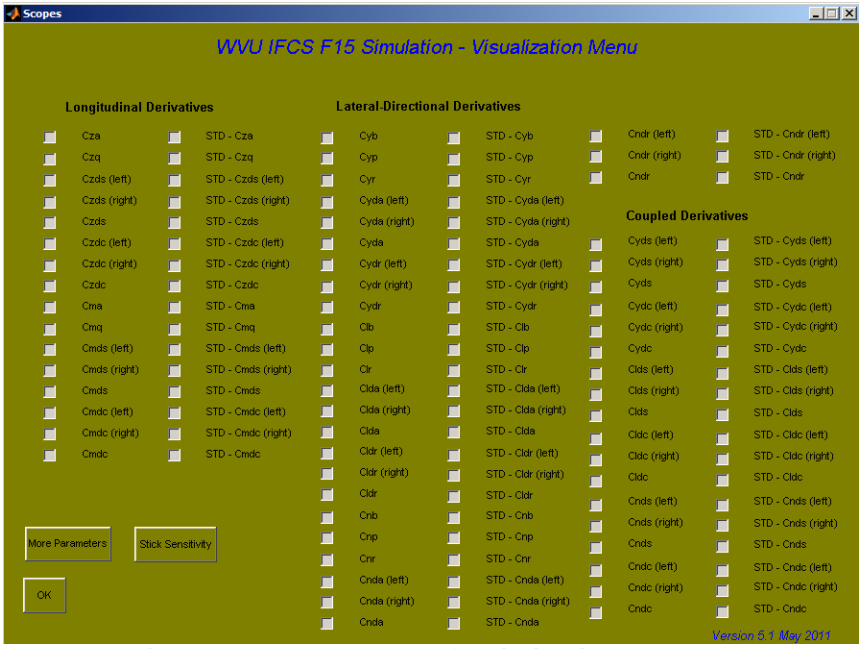

Figure 4-17: Setup menu for derivatives scopes

2. Experimental Setup

Numerous flight simulation scenarios were conducted to investigate the inherited imperfection in the derivatives estimation process and its subsequent effect on the hybrid adaptive controller tracking quality. The controller tracking results are compared with the reference ideal situation when all the derivatives are perfectly computed with no delay or error. The flight tests were conducted at different flight conditions and maneuvers, under both nominal and actuator failure conditions.

The results and discussion are based on twelve flight scenarios. The first six scenarios are at nominal flight conditions, while the last six scenarios are replica of the first six but at failure conditions. A stabilator failure was considered as an example with a complete control loss of the right surface at 10° deflection occurring after five seconds from the start of the simulation.

The selected flight tests in this chapter were considered such that a significant change in the values of the investigated derivatives over the flight period is present. In addition, the scenarios included some decoupled maneuvers to study the individual effects in each channel, as well as, some coupled maneuvers to analyze the interactions between channels. The final decision is considered based on both the average of the obtained data from all the scenarios and separate observations from each scenario.

Some general conditions apply to all flight scenarios; such as, all the tests last for 50 seconds. In addition, no flight turbulence is considered in either of the tests. All the flight tests start at altitude 6100 m and speed 240 m/s.

1- First Flight Test Scenario - Medium Coordinated turn:

(Medium turns are characterized by a bank angle between 20 and 45 degrees [60]).

a. The simulation started at altitude 6100 m with speed 240 m/s, as shown in Figure 4-18 and Figure 4-19 respectively.
b. The turn started with a combination of aileron and rudder inputs applied together, as shown in Figure 4-20, in the same direction to cancel sideslip adverse effect. The entire turn maneuver was performed at constant altitude.
c. Once the desired bank is attained, the aileron input was relaxed to stop the roll; however, a small amount of residual aileron pressure was required in the direction of the turn to keep the bank constant.
d. The rudder was re-adjusted as well to maintain coordinated flight (ideally β = zero, yet in this scenario the sideslip angle was kept under 1 degree, as shown in Figure 4-23).
e. Moreover, the throttle power was increased to keep the flight speed from decreasing during the turn, as shown in Figure 4-21.
f. After the turn maneuver is completed, an opposite aileron pressure was applied to negate the bank angle and return to a level flight.
g. Then an elevator input was applied to climb and end the flight scenario at a new altitude of 8000 m, as shown in Figure 4-18.

h. The angle of attack throughout the simulation is presented in Figure 4-22.

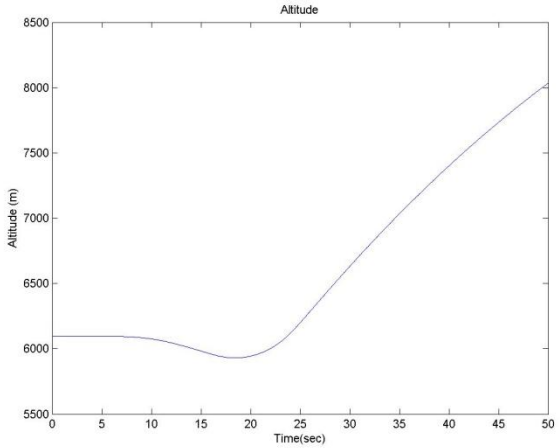

Figure 4-18: First flight test scenario - Altitude

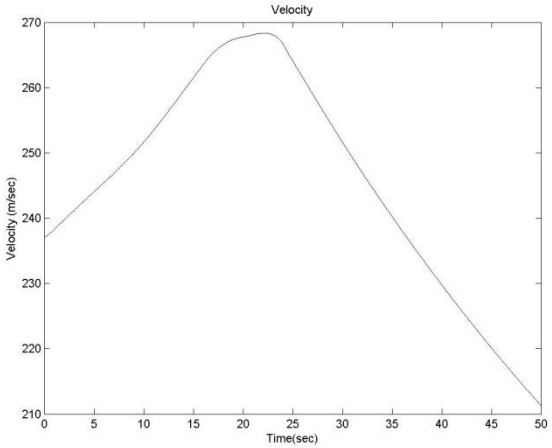

Figure 4-19: First flight test scenario - Velocity

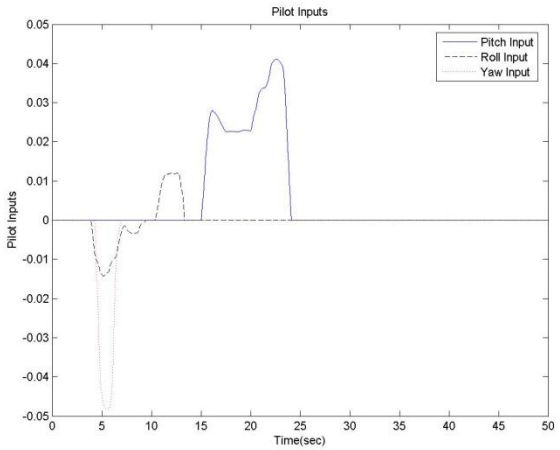

Figure 4-20: First flight test scenario – Pilot inputs

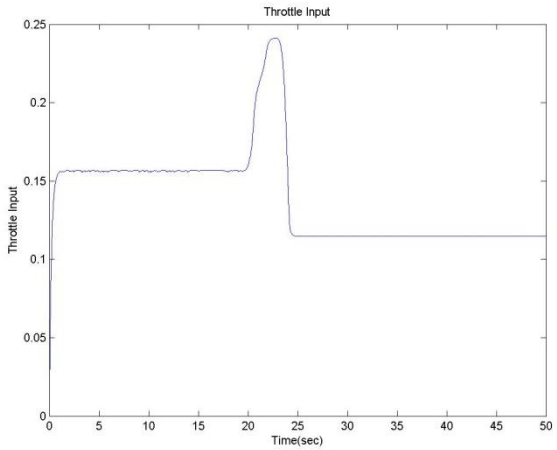

Figure 4-21: First flight test scenario – Throttle inputs

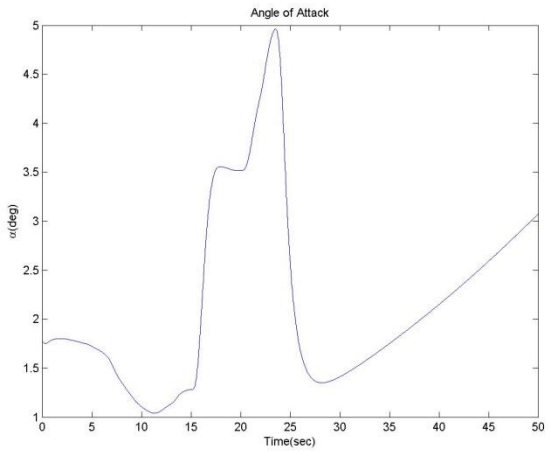

Figure 4-22: : First flight test scenario – AOA

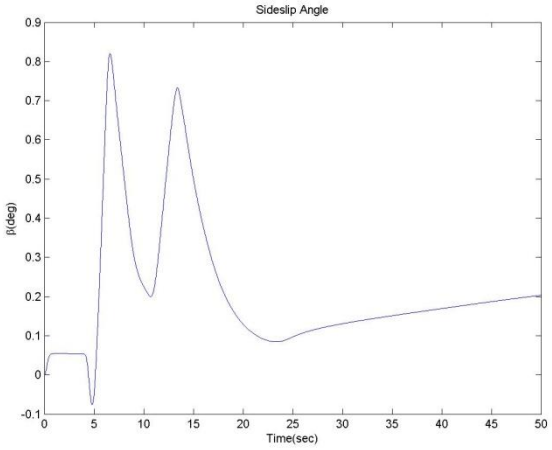

Figure 4-23: : First flight test scenario – Sideslip Angle

2- Second Flight Test Scenario - Pitch Doublet [61]:

a. Start the test at altitude 6100 m and speed 240 m/s, as shown in Figure 4-24 and Figure 4-25.
b. Pull back the elevator column sharply and hold input for 3 seconds, as shown in Figure 4-26.
c. Push forward on elevator column sharply and hold input for 3 seconds.
d. Release the control column to neutral position for 4 seconds.
e. Then repeat steps 2, 3, and 4 to produce a similar pitch doublet yet stronger.
f. Aileron and rudder inputs were kept to zero during this scenario.
g. This simulation considers no throttle position changes, as shown in Figure 4-27.
h. Angle of attack and sideslip angle are presented in Figure 4-28 and Figure 4-29 respectively.

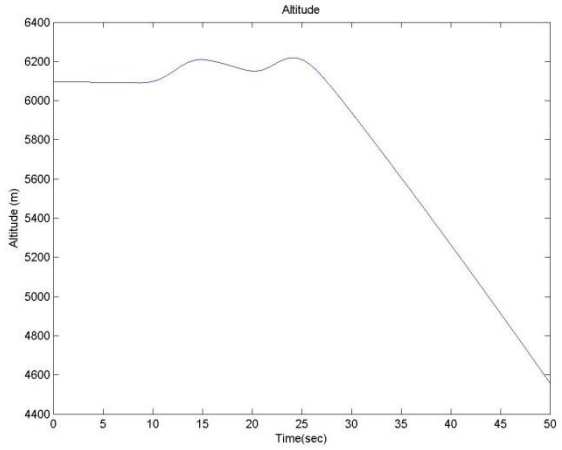

Figure 4-24: Second flight test scenario - Altitude

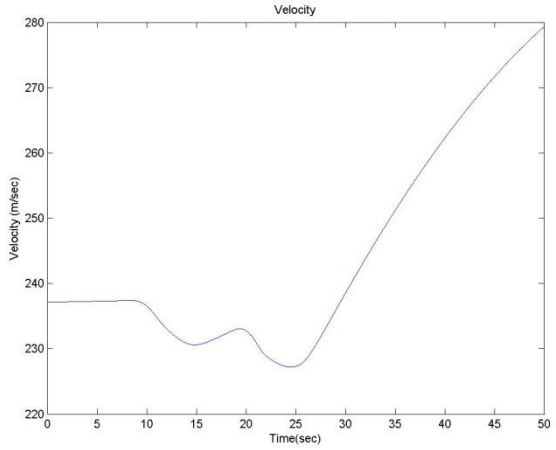

Figure 4-25: Second flight test scenario - Velocity

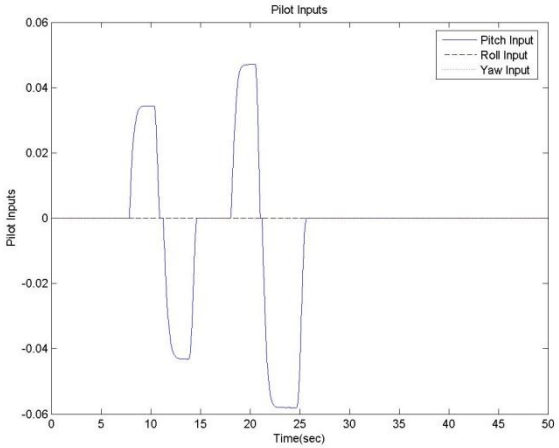

Figure 4-26: Second flight test scenario – Pilot inputs

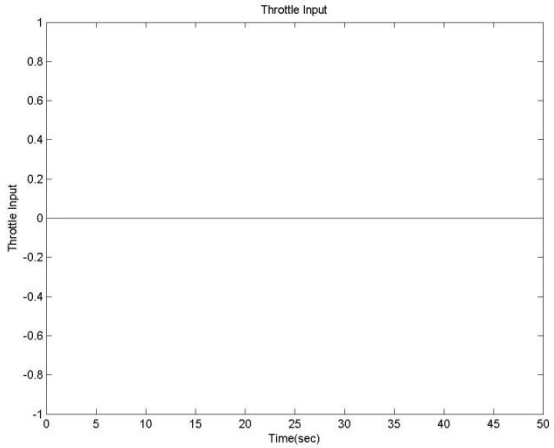

Figure 4-27: Second flight test scenario – Throttle input

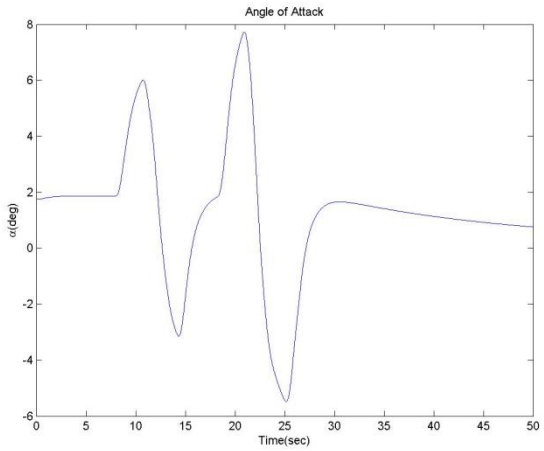

Figure 4-28: Second flight test scenario - AOA

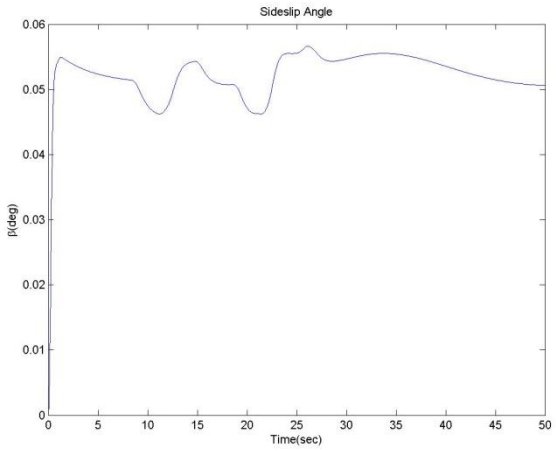

Figure 4-29: Second flight test scenario – Sideslip angle

3- Third Flight Test Scenario - Roll Doublet [61]:

a. Start the test at altitude 6100 m and speed 240 m/s, as shown in Figure 4-30 and Figure 4-31.
b. Positive sharp aileron input was produced and held for 3 seconds, as shown in Figure 4-32.
c. Negative sharp aileron input was then produced for 3 seconds.
d. Release the input to neutral position for 4 seconds.
e. Then repeat steps 2, 3, and 4 to produce a similar roll doublet yet stronger.
f. Elevator and rudder inputs were kept to zero during this scenario.
g. This simulation considers no throttle position changes, as shown in Figure 4-33.
h. Angle of attack and sideslip angle are shown in Figure 4-34 and Figure 4-35 respectively.

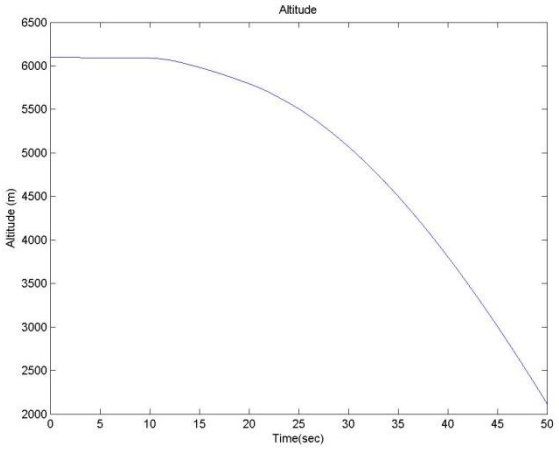

Figure 4-30: Third flight test scenario - Altitude

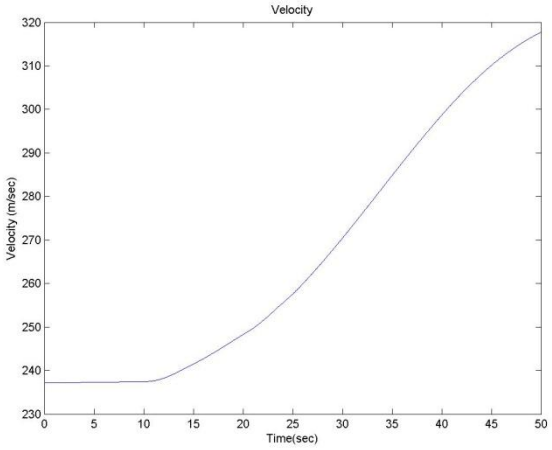

Figure 4-31: Third Flight test scenario - Velocity

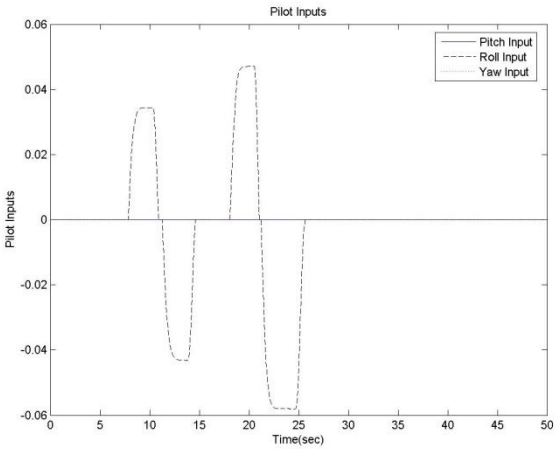

Figure 4-32: Third Flight test scenario – Pilot inputs

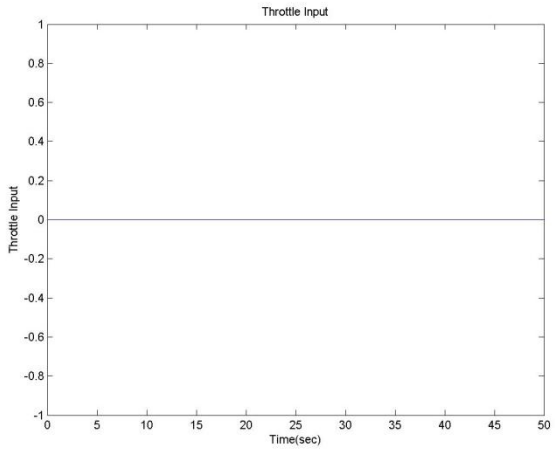

Figure 4-33: Third Flight test scenario – Throttle input

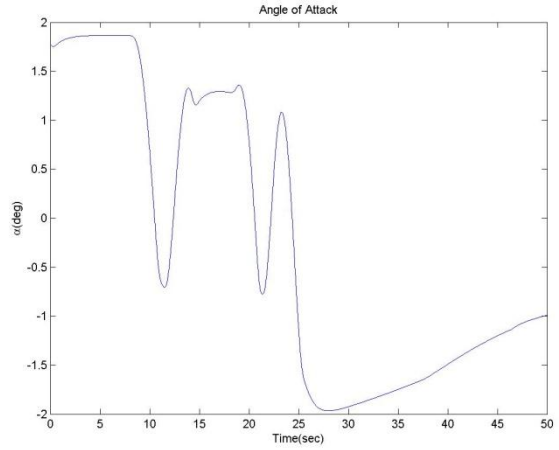

Figure 4-34: Third Flight test scenario - AOA

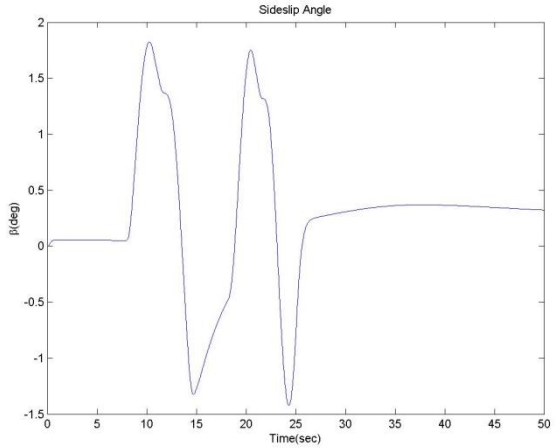

Figure 4-35: Third Flight test scenario – Sideslip angle

4- Fourth Flight Test Scenario - Yaw Doublet [61]:

a. Start the test at altitude 6100 m and speed 240 m/s, as shown in Figure 4-36 and Figure 4-37.
b. Positive sharp rudder input was produced and held for 3 seconds, as shown in Figure 4-38.
c. Negative sharp rudder input was then produced for 3 seconds.
d. Release the input to neutral position for 4 seconds.
e. Then repeat steps 2, 3, and 4 to produce a similar yaw doublet yet stronger.
f. Elevator and aileron inputs were kept to zero during this scenario.
g. This simulation considers no throttle position changes, as shown in Figure 4-39.
h. Angle of attack and sideslip angle are presented in Figure 4-40 and Figure 4-41 respectively.

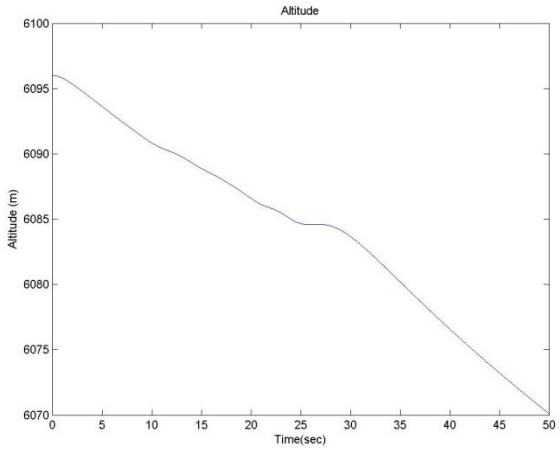

Figure 4-36: Fourth flight test scenario - Altitude

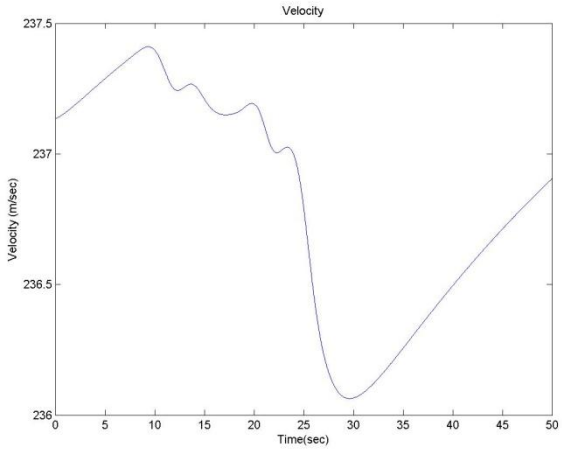

Figure 4-37: Fourth Flight test scenario – Velocity

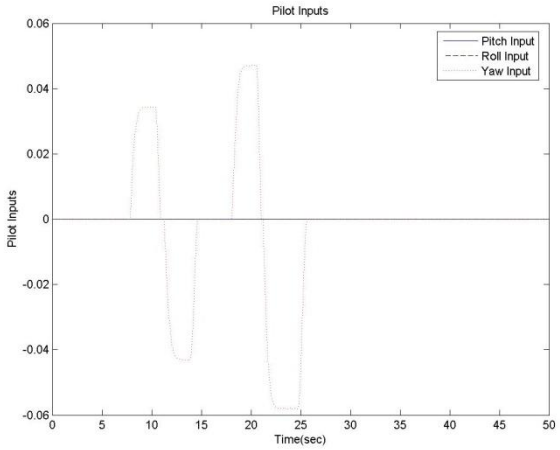

Figure 4-38: Fourth Flight test scenario – Pilot inputs

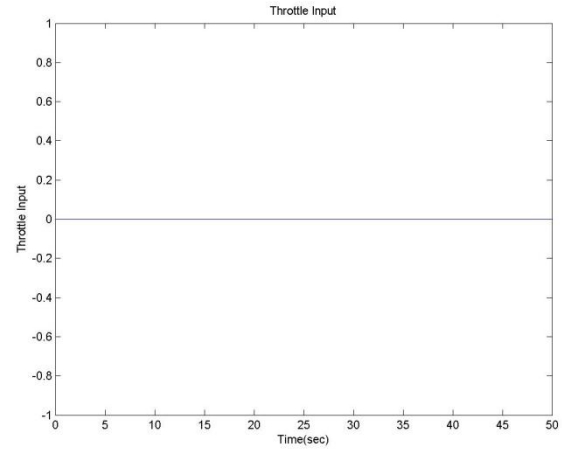

Figure 4-39: Fourth Flight test scenario – Throttle input

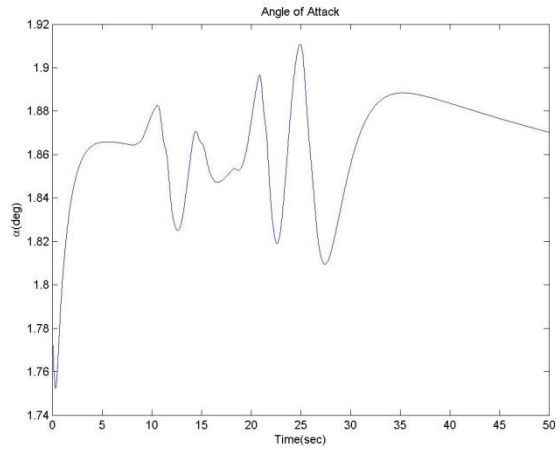

Figure 4-40: Fourth Flight test scenario - AOA

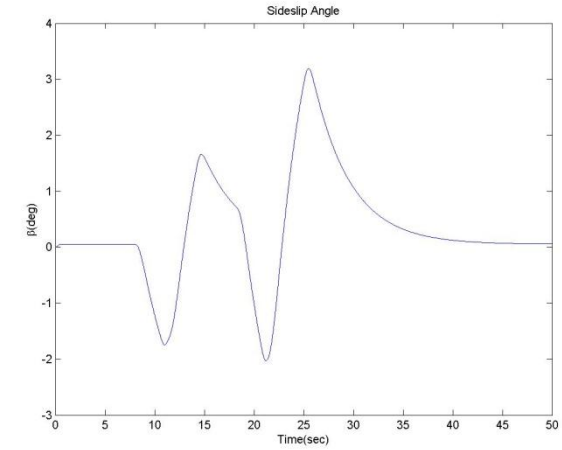

Figure 4-41: Fourth Flight test scenario - Sideslip angle

5- Fifth Flight Test Scenario - High AOA Climbing [61]:

a. Start the simulation with a steady level flight at altitude 6100 m and speed 240 m/s for 5 seconds, as shown in Figure 4-42 and Figure 4-43.
b. Positive elevator input is exerted to start the climbing operation with a high AOA, as shown in Figure 4-44.
c. The throttle power was increased gradually into full power, as shown in Figure 4-45, in order to support the high AOA ($\alpha = 10°, slope = 0.5$) flight,[Figure 4-46], without causing a stall flight.
d. The flight altitude reached 12000 m and speed 120 m/s.
e. Sideslip angle is shown in Figure 4-47.

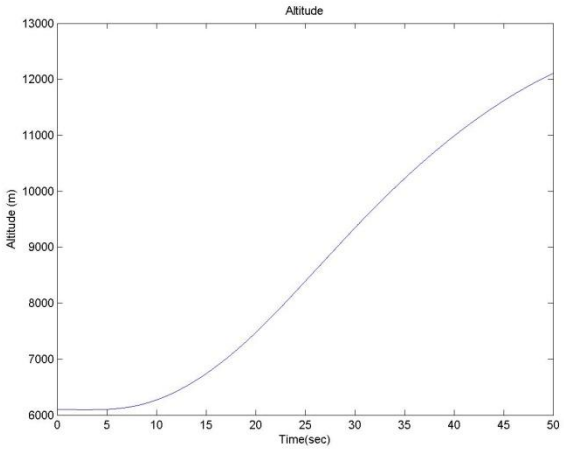

Figure 4-42: Fifth flight test scenario - Altitude

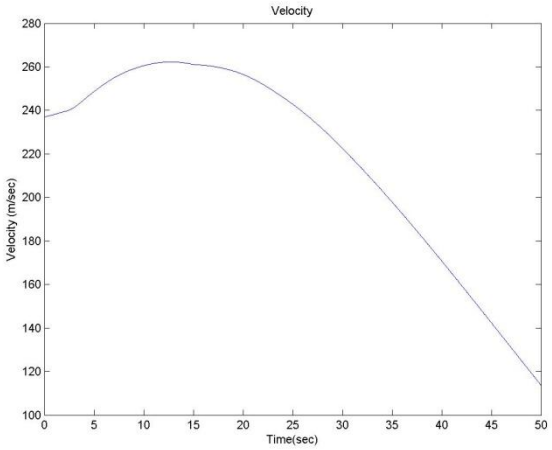

Figure 4-43: Fifth Flight test scenario – Velocity

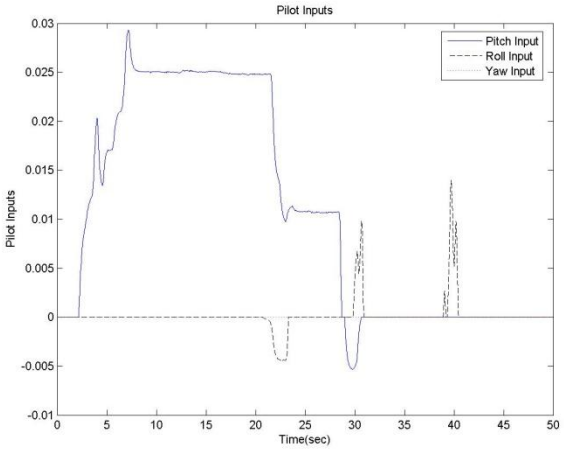

Figure 4-44: Fifth Flight test scenario – Pilot inputs

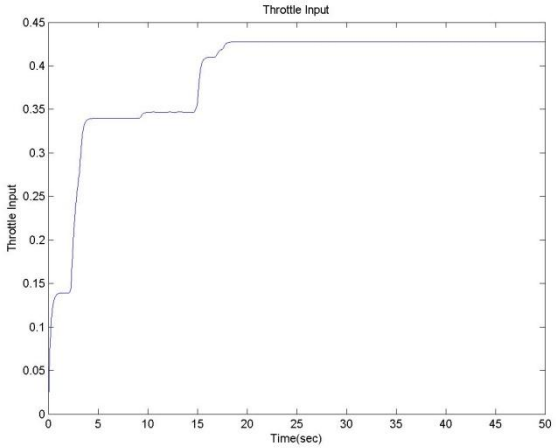

Figure 4-45: Fifth Flight test scenario - Throttle inputs

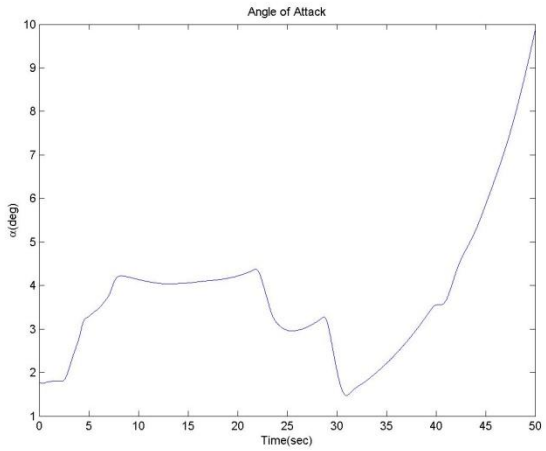

Figure 4-46: Fifth Flight test scenario - AOA

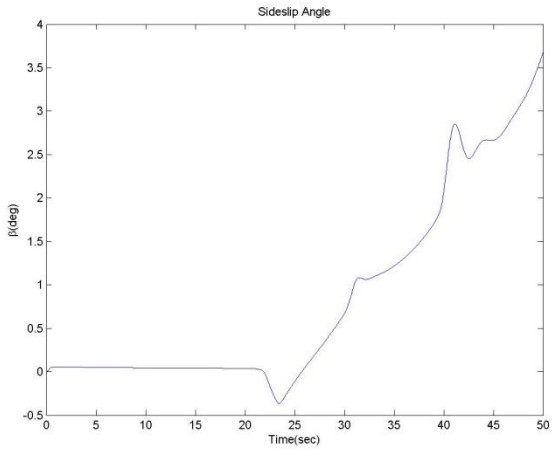

Figure 4-47: Fifth Flight test scenario – Sideslip angle

f. Sixth Flight Test Scenario - All Doublets [61]:

a. Start the flight scenario at altitude 6100 m and speed 240 m/s, as shown in Figure 4-48 and Figure 4-49.
b. This flight test consists of a series of doublets on the three channels alternately, as shown in Figure 4-50.
c. Pull back the elevator column sharply and hold input for 3 seconds.
d. Push forward on elevator column sharply and hold input for 3 seconds.
e. Release the control column to neutral position for 4 seconds.
f. Positive sharp aileron input was produced and held for 3 seconds.
g. Negative sharp aileron input was then produced for 3 seconds.
h. Release the input to neutral position for 4 seconds.
i. Positive sharp rudder input was produced and held for 3 seconds.
j. Negative sharp rudder input was then produced for 3 seconds.
k. Release the input to neutral position for the rest of the flight scenario.
l. During each channel input, the two other channels were kept to zero.
m. This simulation considers and no throttle position changes, as shown in Figure 4-51.
n. The flight ends at altitude 4100 m and speed 290 m/s.
o. Angle of attack and sideslip angle are presented in Figure 4-52 and Figure 4-53 respectively.

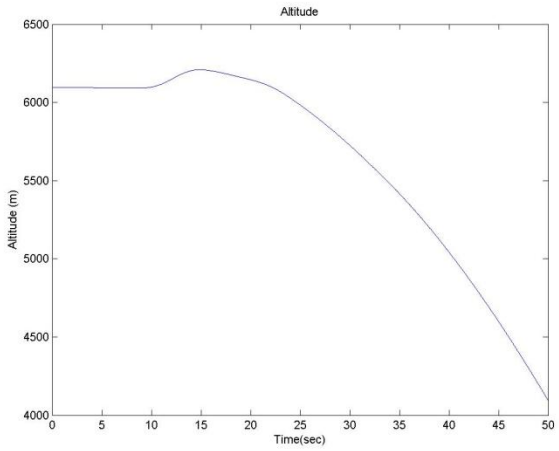

Figure 4-48: Sixth flight test scenario - Altitude

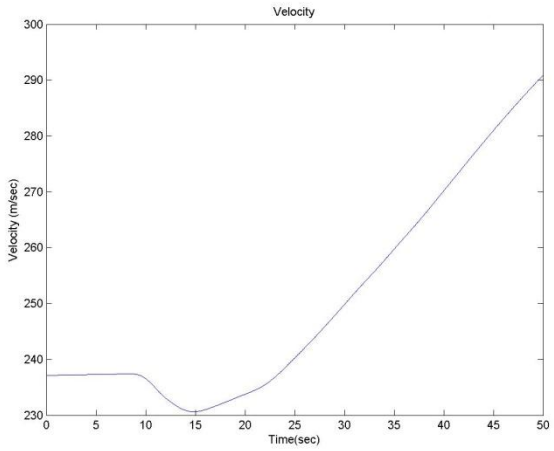

Figure 4-49: Sixth flight test scenario - Velocity

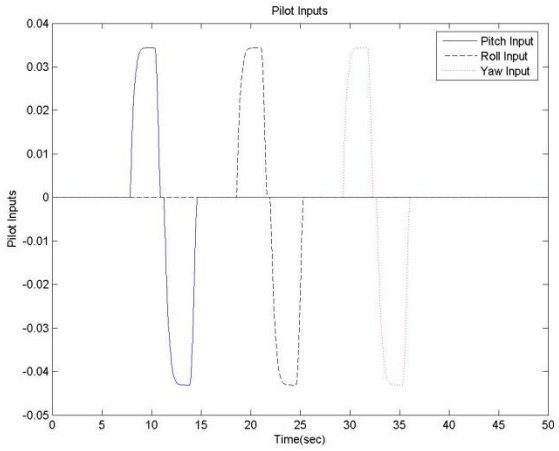

Figure 4-50: Sixth flight test scenario – Pilot inputs

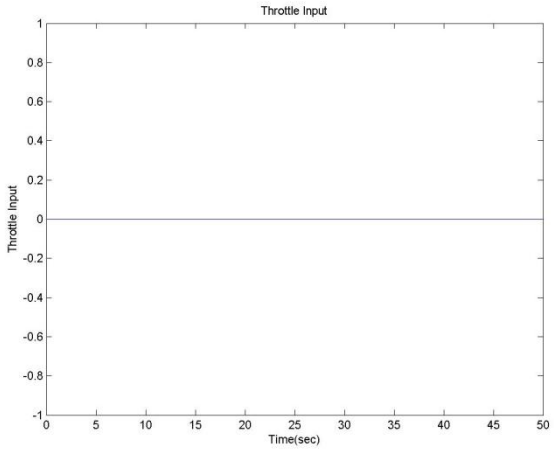

Figure 4-51: Sixth flight test scenario – Throttle input

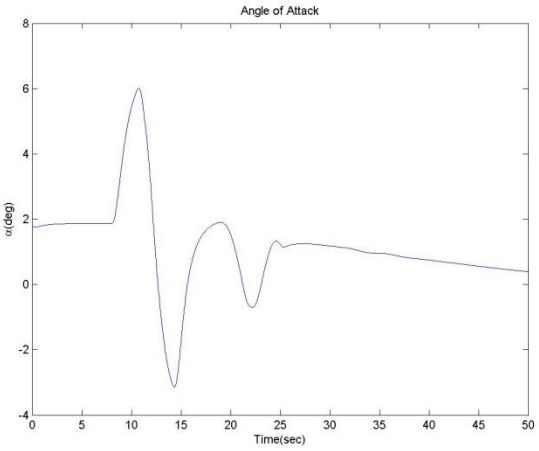

Figure 4-52: Sixth flight test scenario - AOA

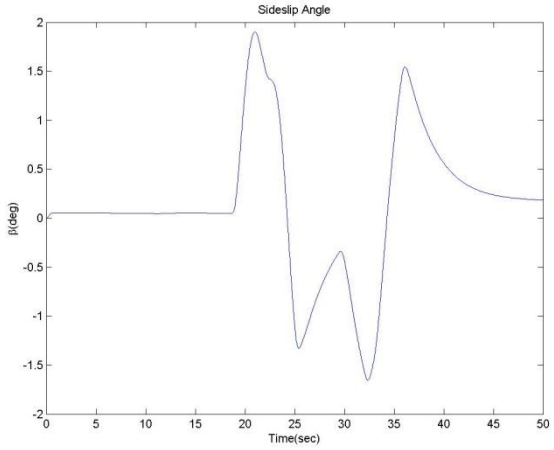

Figure 4-53: Sixth flight test scenario – Sideslip angle

Chapter 5. Results and Discussion

The results and discussion in this section are based on the twelve flight scenarios presented in the last chapter. The discussion is divided for each derivative of the nine derivatives, which were introduced in the first chapter. The sensitivity analysis is based on the following aspects:

1- Ideal Estimation: The flight tests are conducted first with all the derivatives perfectly computed and used to update the controller on time and with no bias.
2- No Estimation: The experiment is conducted again with one of the derivatives is fixed, while others are perfectly computed and fed to the controller to study the effect of fixing this specific derivative.
3- Delayed Estimation: The tests are conducted with all the derivatives updated perfectly except one derivative, which is updated with a delay to study the effect of any potential delay in the estimation process on tracking the desired flight states.
4- Biased Estimation: Conduct the same flight tests with all derivatives updated perfectly except one derivative, which is updated with different biases to study the effect of convergence to a different value during the estimation process.
5- Delayed and Biased Estimation: Flight scenarios that combine delay and bias in the estimation of individual derivatives were conducted to explore the extreme effect of combining both types of errors.

The investigated delay values extend from 0.5 to 40 seconds. The bias values range from 5% to 500% and -5% to -500%.

The errors between the reference (desired) and actual roll, pitch, and yaw rates (deg/sec) were used as performance metrics and basis of comparison. Additionally, the neural network output was recorded to assess the NN effort in compensating the error in each case. The integral of the three errors and the integral of the NNs' effort exerted to compensate for those errors over the test period are also presented to support the comparison.

Finally, the comparison will be divided into three main sections: updated vs. fixed, updated vs. updated with a delay, and updated vs updated with a bias. As mentioned above, there are twelve flight tests that consist of six different fligh manuevers, which were performed twice at both nominal and stabilator failure conditions. A limited number of tests with both estimation delay and bias were also performed, analyzed, and discussed.

1. Nominal Condition

$1 - Cm_{\alpha}$

I) Updated Derivative Against Fixed Derivative

[Figure 5-1] shows both updated and fixed $C_{m\alpha}$.

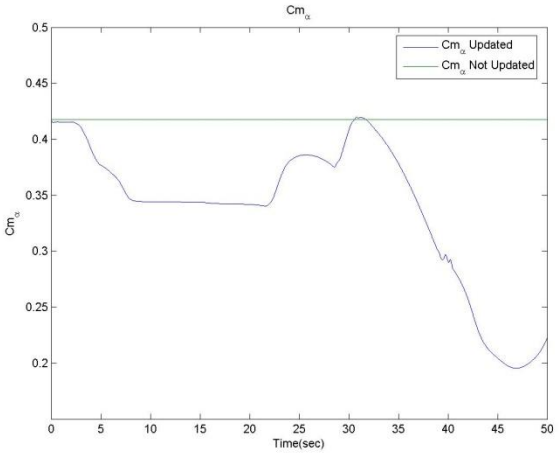

Figure 5-1: Updated vs. fixed Cm_{α}- Fifth scenario

The hybrid controller succeeded in tracking the desired pitch, roll, and yaw rates in both cases; updated and fixed $C_{m\alpha}$. A clear comparison between the effect of updating and fixing $C_{m\alpha}$ on the three rates tracking errors is presented through calculating the sum of the tracking errors on each channel as well as the total neural network effort exerted to compensate for the corresponding errors in both cases.

The average results of all six scenarios, [Table 5-1], show that the hybrid controller with fixed $C_{m\alpha}$ had a better performance as compared to the ideal hybrid controller with updated $C_{m\alpha}$; this unexpected behavior will be explained further at the end of this chapter. The pitch neural network effort decreased by updating $C_{m\alpha}$.

The analysis of individual scenarios shows that the importance of updating $C_{m\alpha}$ depends greatly on the variation range of $C_{m\alpha}$ as well as the nature of the maneuvers and inputs. Fifth scenario, [Table 5-2], which was characterized by 55% variation in $C_{m\alpha}$ along with a large pitch input to climb with high AOA, shows that updating the derivative will enhance the quality of the controller in tracking the pitch rate by 36% as compared to using a fixed derivative, [Figure 5-2]. However, that scenario did not exhibit any change in the neural network effort to compensate for this huge error degradation.

In the second scenario, [Table 5-3] - two pitch doublets, the performance of the controller unexpectedly degraded by 30% after updating the $C_{m\alpha}$. On the other hand, the neural network effort decreased by 21% when updating the derivative, as shown in Figure 5-3. As expected, due to

decoupling, updating this derivative did not have any effect on tracking either roll or yaw rates. Moreover, the neural network effor of both channels did not change when fixing $C_{m\alpha}$.

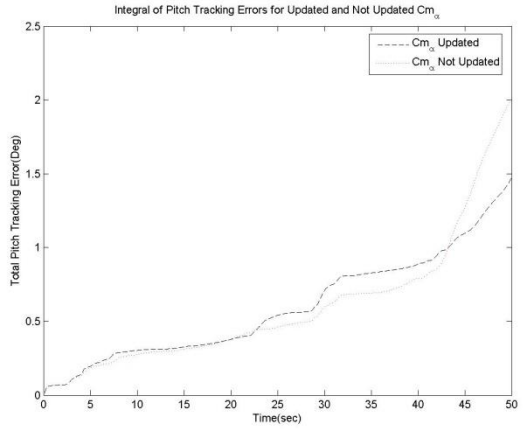

Figure 5-2: Integral of pitch tracking error for updated vs. fixed Cm_{α}- Fifth scenario

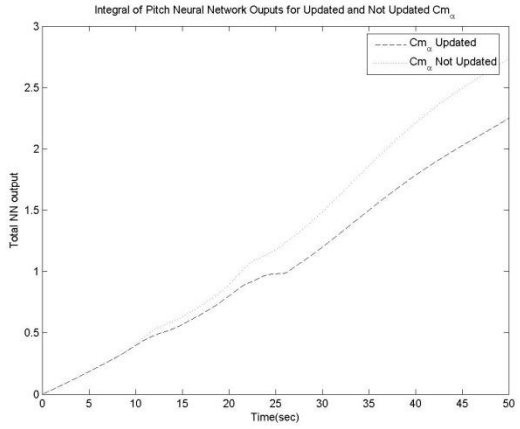

Figure 5-3: Integral of pitch neural network output for updated vs. fixed Cm_{α}- Second scenario

Table 5-1: Updated vs. fixed Cm_{α}- Average results

	Updated $C_{m\alpha}$	Fixed $C_{m\alpha}$
Integral of Pitch Tracking Error (%)	100.00	92.26
Integral of Roll Tracking Error (%)	100.00	99.78
Integral of Yaw Tracking Error ()	100.00	99.53
Integral of Pitch Neural Network (%)	100.00	109.50
Integral of Roll Neural Network (%)	100.00	9.99
Integral of Yaw Neural Network (%)	100.00	99.78

Table 5-2: Updated vs. fixed Cm_α– Fifth scenario

	Updated $C_{m\alpha}$	Fixed $C_{m\alpha}$
Integral of Pitch Tracking Error (%)	100	136.5
Integral of Roll Tracking Error (%)	100	99.382
Integral of Yaw Tracking Error (%)	100	97.8
Integral of Pitch Neural Network (%)	100	100.45
Integral of Roll Neural Network (%	100	99.714
Integral of Yaw Neural Network (%)	100	98.53

Table 5-3: Updated vs. fixed Cm_α– Second scenario

	Updated $C_{m\alpha}$	Fixed $C_{m\alpha}$
Integral of Pitch Tracking Error (%)	10	69.24
Integral of Roll Tracking Error (%)	100	99.408
Integral of Yaw Tracking Error (%)	100	99.631
Integral of Pitch Neural Network (%)	100	121.54
Integral of Roll Neural Network (%)	100	100.06
Integral of Yaw Neural Network (%)	100	100.17

II) Updated Derivative Against Delayed Derivative

The effect of potential delays during the estimation process of $C_{m\alpha}$ is discussed in this section. $C_{m\alpha}$ is updated with different delays from 0.5 to 40 seconds and the tracking errors in every case are compared with the ideally updated derivative.

Updating the derivative with a delay did not have any effect on Roll or Yaw rates in any of the flight tests, as expected. On the other hand, the delay had a significant effect on the quality of tracking pitch rate in some scenarios specially the ones that included considerable longitudinal input. For instance, the second scenario with a pitch doublet experiences 27% more error in tracking pitch rate with only 2 seconds delay in estimating $C_{m\alpha}$. Fifth scenario experiences 43% more error in tracking pitch rate

when there is 17 seconds delay. However, the average of all the scenarios does not exhibit considerable or consistent effect for the delay on pitch tracking error, as listed in Table 5-4. Finally, no more effort was needed from the neural network on roll or yaw channel, yet the pitch neural network exerted more effort.

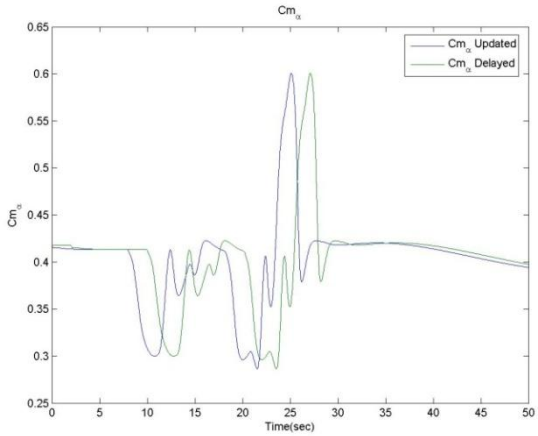

Figure 5-4 Updated vs. delayed (2-seconds) Cm_α- Second scenario

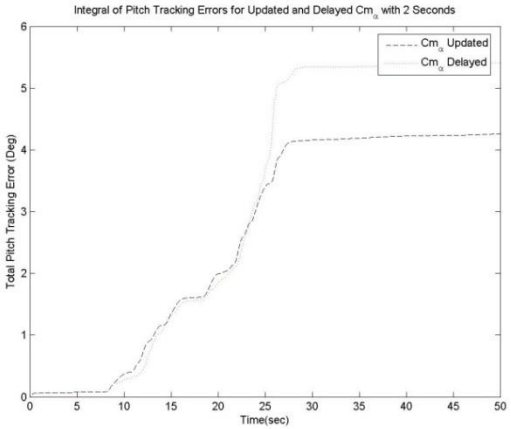

Figure 5-5: Integral of pitch tracking error for updated vs. delayed (2-seconds) Cm_α- Second scenario

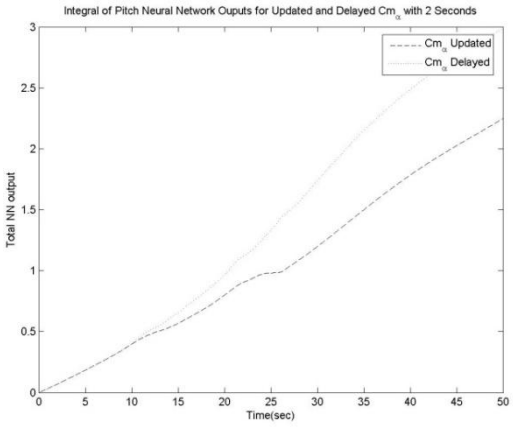

Figure 5-6: Integral of pitch neural network output for updated vs. delayed (2-seconds) Cm_α- Second scenario

Table 5-4: Updated vs. delayed Cm_α– Average results

Delay (sec)	0	0.5	1	2	3	4	5	7	9	10	12	15	17	20	25	30	35	40
Integral of Pitch Tracking Error(deg)	100	99.65	98.7	98.26	101.1	101.6	101.7	91.57	93.41	95.16	99.57	99.99	98.62	98.76	97.25	91.56	91.17	90.84
Integral of Roll Tracking Error(deg)	100	100.1	100.4	99.98	100.2	100.1	100.3	100	99.92	100.2	99.92	100.1	99.88	99.83	99.79	99.79	99.78	99.78
Integral of Yaw Tracking Error(deg)	100	100.1	100.1	99.74	99.91	100.1	99.88	99.65	99.77	99.87	99.8	99.66	99.57	99.65	99.59	99.58	99.55	99.57
Integral of Pitch Neural Network	100	100.7	102.9	108.3	113.6	113.5	111.9	108.8	108.7	108.7	111.9	112.1	111	112.1	110.7	112.9	110.8	111
Integral of Roll Neural Network	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Integral of Yaw Neural Network	100	99.98	99.96	99.95	99.95	99.92	99.88	99.83	99.77	99.75	99.78	99.78	99.84	99.88	99.86	99.86	99.83	99.79

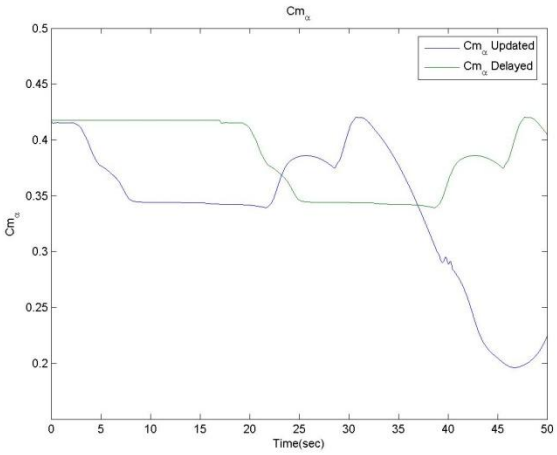

Figure 5-7: Updated vs. delayed (17-seconds) Cm_α- Second scenario

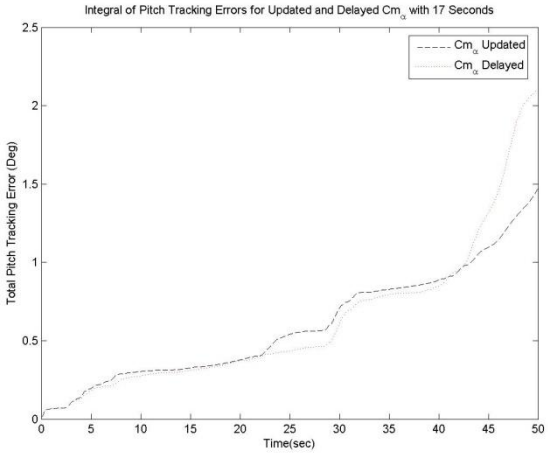

Figure 5-8: Integral of pitch tracking error for updated vs. delayed (17-seconds) Cm_α- Second scenario

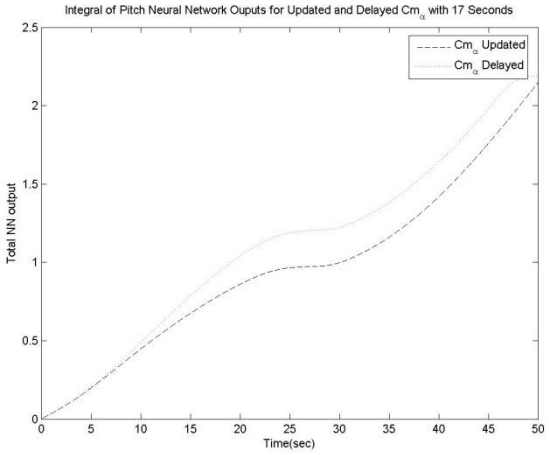

Figure 5-9: Integral of pitch neural network output updated vs. delayed (17-seconds) Cm_α- Second scenario

III) Updated Derivative Against Biased Derivative

$C_{m\alpha}$ estimate might practically converge to a wrong value; therefore, the effects of potential bias in the estimation are investigated in this section. $C_{m\alpha}$ is updated with different biases from -500% up to 500% of the initial value of the derivative. The tracking errors in every case are compared with the ideally updated derivative. When the bias is less than 50% the neural network was able to maintain the pitch tracking error close to the original error produced whilst using the updated derivative. Starting from 50% bias, the error will become greater as compared to the fixed derivative. That makes sense because 50% bias results in as much difference as between the fixed value and the actual value of the derivative, as shown in Figure 5-1 and Figure 5-10.

Increasing the bias more will significantly increase the pitch tracking error. Moreover, starting from this bias level, the neural network effort will significantly increase in vain. Consequently, when the bias is more than 50%, not updating the derivative will be better. For negative bias, -15% will have as

much error as %50 bias. Positive bias does not drive the system into instability but negative bias produced an unstable system after -400%. Fifth scenario then second scenario showed the steepest error increase with increasing the bias value. At low biases, there is no effect on the roll or yaw tracking errors neither on their respective neural networks. However, at high negative biases, the tracking quality of yaw and roll angular rates has degraded along with an increase in their corresponding NNs. High Positive bias did not have any effect on the quality of roll and yaw channels or their neural networks' effort.

Table 5-5: Updated vs. biased C_{m_α} - Average results

Bias (%)	Integral of Pitch Tracking Error(deg)	Integral of Roll Tracking Error(deg)	Integral of Yaw Tracking Error(deg)	Integral of Pitch Neural Network	Integral of Roll Neural Network	Integral of Yaw Neural Network
0	100.00	100.00	100.00	100.00	100.00	100.00
5	93.43	99.84	99.75	98.95	100.00	99.96
10	88.72	99.65	99.49	97.69	100.00	99.91
15	86.17	99.47	99.24	96.71	100.00	99.87
20	85.71	99.30	98.99	95.98	100.01	99.83
25	87.28	99.12	98.75	95.11	100.01	99.79
30	90.68	98.95	98.51	94.52	100.01	99.75
50	116.87	98.29	97.62	93.93	100.02	99.60
80	164.72	97.39	96.39	102.90	100.03	99.39
100	197.98	96.80	95.67	113.12	100.04	99.26
120	227.63	96.24	95.13	131.48	100.01	99.13
150	277.58	95.48	94.26	156.18	100.03	98.98
200	358.25	94.23	92.79	193.91	100.03	98.77
250	428.47	92.96	91.40	223.92	100.00	98.47
300	494.82	91.86	90.29	256.62	100.00	98.25
350	556.45	90.86	89.30	288.19	99.99	98.04
400	613.75	89.95	88.42	318.59	99.98	97.86
450	667.08	89.11	87.65	347.68	99.97	97.68
500	716.74	88.34	87.00	375.83	99.95	97.51
-5	106.91	100.19	100.27	101.38	100.00	100.05
-10	113.16	100.37	100.53	103.49	100.00	100.09
-15	120.10	100.55	100.79	106.04	99.99	100.13
-20	128.37	100.74	101.03	107.51	99.99	100.17
-25	139.92	100.94	101.27	106.25	99.99	100.22
-30	150.02	101.13	101.55	107.67	99.99	100.27
-50	171.89	109.70	95.96	124.76	104.35	99.39
-80	236.12	111.11	98.14	142.36	104.36	99.82
-100	290.59	112.18	99.91	149.74	104.36	100.08
-120	347.36	113.35	101.77	159.20	104.35	100.39
-150	446.81	115.48	105.10	171.13	104.33	100.88
-200	663.65	120.42	112.04	193.96	104.25	102.01
-250	1073.06	128.57	125.78	229.44	103.99	104.23

-300	2608.23	166.23	254.64	245.65	103.97	110.60
-350	3244.10	162.07	218.58	256.97	106.84	108.90
-400	unstable	unstable	unstable	unstable	unstable	unstable
-450	unstable	unstable	unstable	unstable	unstable	unstable
-500	unstable	unstable	unstable	unstable	unstable	unstable

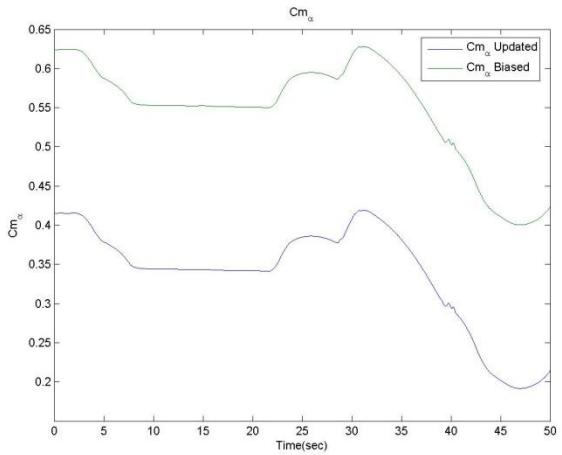

Figure 5-10: Updated vs. biased (50%) Cm_α- Fifth scenario

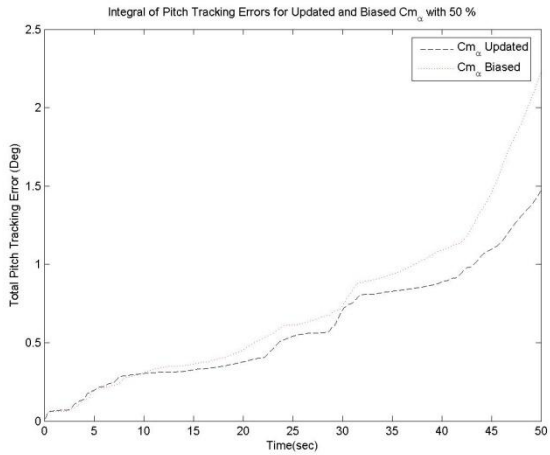

Figure 5-11: Integral of pitch tracking error for updated vs. biased (50%) Cm_α- Fifth scenario

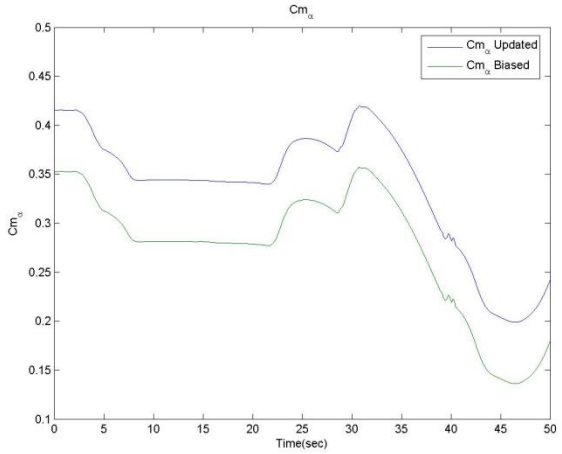

Figure 5-12: Updated vs. biased (-15%) Cm_α- Fifth scenario

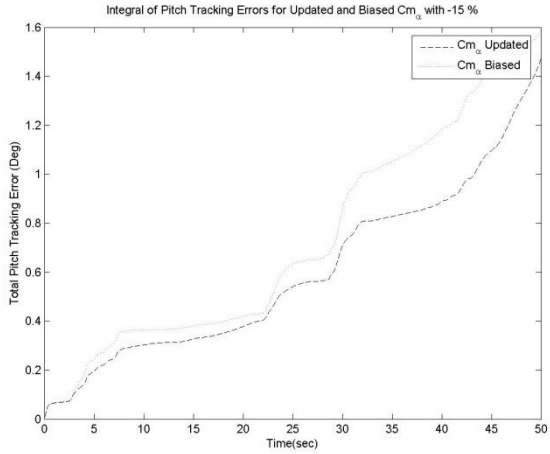

Figure 5-13: Integral of pitch tracking error for updated vs. biased (-15%) Cm_α- Fifth scenario

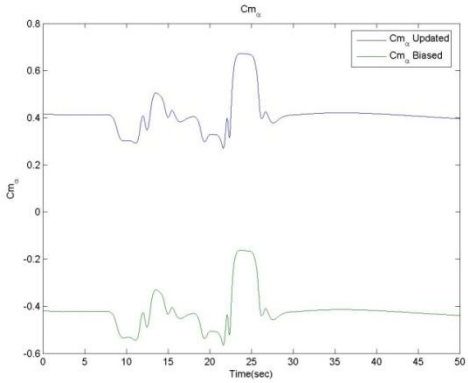

Figure 5-14: Updated vs. biased (-200%) Cm_α- Second scenario

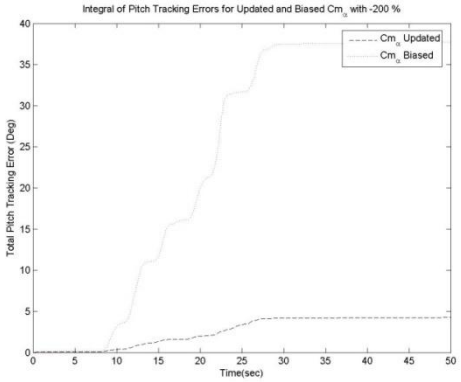

Figure 5-15: Integral of pitch tracking error for updated vs. biased (-200%) Cm_α- Second scenario

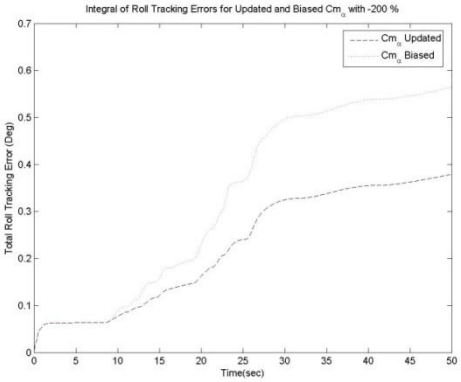

Figure 5-16: Integral of roll tracking error for updated vs. biased (-200%) Cm_α- Second scenario

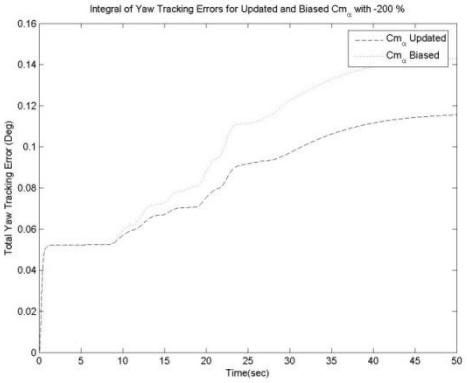

Figure 5-17: Integral of yaw tracking error for updated vs. biased (-200%) Cm_α- Second scenario

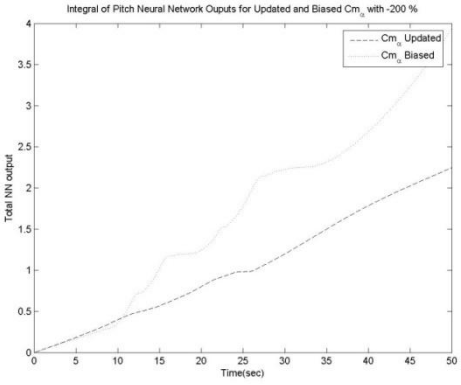

Figure 5-18: Integral of pitch neural network output for updated vs. biased (-200%) Cm_α- Second scenario

IV) Bias and Delay

Tests were conducted using the fifth scenario to show the effect of combining both delay and bias. The results showed that the system became unstable at the bias value (-400%), and there were some inconsistently insignificant increase in the error as compared to the case of pure bias. In short, adding delay to bias did not show worse effect than the single bias.

2- Cm_q

I) Updated Derivative Against Fixed Derivative

The average results all the six scenarios, when C_{mq} is updated and fixed, shows that updating this derivative does not exhibit significant change in tracking performance or neural networks effort. Updating the derivative improved pitch tracking quality by only 1.5% as compared to no update. Conversely, the pitch neural network effort increased by about 1.5% after updating the derivative, [Table 5-6].

The first flight scenario, coordinated turn, had the largest effect of updating C_{mq} on pitch tracking error among all the scenarios whereas the pitch tracking improved by 10%, [Figure 5-19 and Figure 5-20]. The neural network effort, however, increased by 5% as compared to fixing the derivative, [Table 5-7].

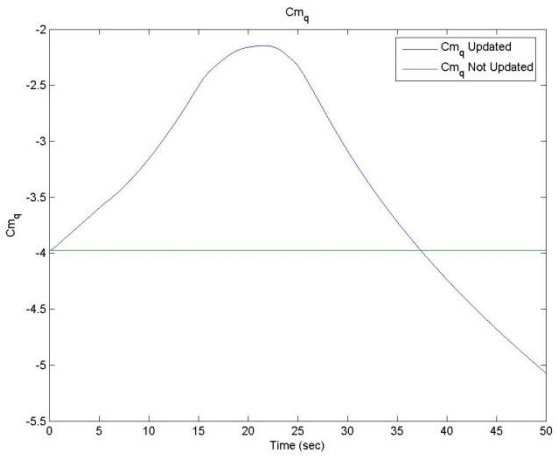

Figure 5-19: Updated vs. fixed Cm_q- First scenario

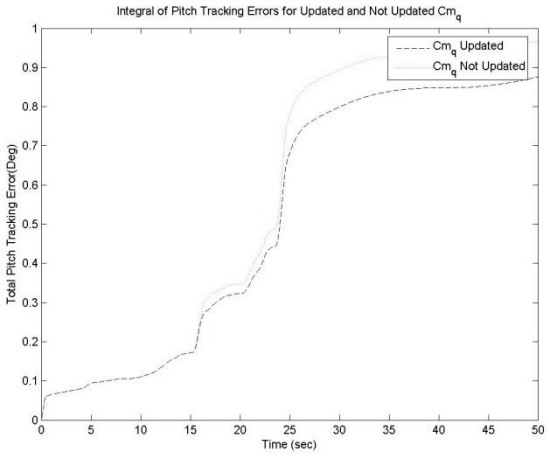

Figure 5-20: Integral of pitch tracking errors for updated vs. fixed Cm_q- First scenario

Table 5-6: Updated vs. fixed Cm_q- Average results

	Updated C_{mq}	Fixed C_{mq}
Integral of Pitch Tracking Error (%)	100.00	101.45
Integral of Roll Tracking Error (%)	100.00	100.03
Integral of Yaw Tracking Error (%)	100.00	100.05
Integral of Pitch Neural Network (%)	100.00	98.54
Integral of Roll Neural Network (%)	100.00	100.00
Integral of Yaw Neural Network (%)	100.00	100.02

Table 5-7: Updated vs. fixed Cm_q- First scenario

	Updated C_{mq}	Fixed C_{mq}
Integral of Pitch Tracking Error (%)	100	110.51
Integral of Roll Tracking Error (%)	100	100.01
Integral of Yaw Tracking Error (%)	100	100.01
Integral of Pitch Neural Network (%)	100	94.893
Integral of Roll Neural Network (%)	100	99.972
Integral of Yaw Neural Network (%)	100	99.982

II) Updated Derivative Against Delayed Derivative

The delay does not have any effect on roll or yaw tracking along with their neural networks. The average of all the scenarios does not show a significant effect in either pitch tracking error or its NN. The first scenario, [Table 5-8], was the only scenario to show an effect to delay. When the delay becomes 7 seconds, pitch-tracking error starts to increase from the ideal case. Moreover, 20 seconds delay in updating C_{mq} generates as much error as not updating this derivative, [Figure 5-21, Figure 5-22, and Table 5-8]. The pitch neural network during this scenario does not show any success in reducing the error.

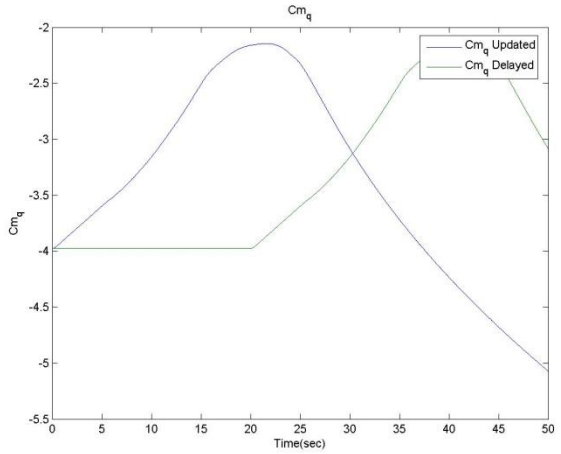

Figure 5-21: Updated vs. delayed (20-seconds) Cm_q- First scenario

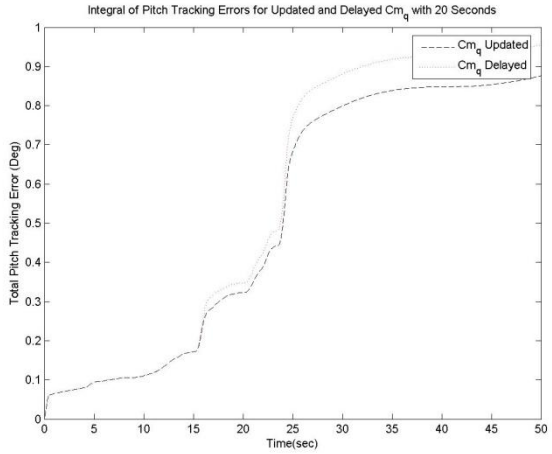

Figure 5-22: Integral of pitch tracking error for updated vs. delayed (20-seconds) Cm_q- First scenario

Table 5-8: Updated vs. delayed Cm_α– First scenario

Delay (sec)	0	0.5	1	2	3	4	5	7	9	10	12	15	17	20	25	30	35	40
Integral of Pitch Tracking Error(deg)	100	99.88	99.83	99.83	99.87	100	100.4	101.1	102.7	103.5	105	106.9	107.8	109.1	110.5	110.5	110.5	110.5
Integral of Roll Tracking Error(deg)	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Integral of Yaw Tracking Error(deg)	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Integral of Pitch Neural Network	100	100	99.89	99.66	99.51	99.42	99.36	99.2	99.2	99.12	98.87	98.71	98.64	98.57	98.54	98.54	98.54	98.54
Integral of Roll Neural Network	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Integral of Yaw Neural Network	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100

III) Updated Derivative Against Biased Derivative

The biased estimates of C_{mq} affect the quality of tracking pitch rate and the pitch neural network. The biased derivative does not present any effect on tracking either roll or yaw rates, or their respective neural networks outputs. Increasing the negative bias slowly increases the pitch neural network effort, contrary to increasing the positive bias, which slightly decreases the pitch neural network effort. In average, when C_{mq} was updated with a bias, even with a very small value such as 10%, the pitch tracking error increased. When the bias reaches 80%, the pitch tracking error increases by 11%, which is worse than not updating the derivative. That means the estimation process becomes useless if it yields a convergence error greater than 80%. On the contrary, this derivative was not as sensitive to negative biases such that -500% bias only showed 14% more error as compared to the perfect derivative.

The third scenario, two roll doublets, was the least sensitive to biased C_{mq}, followed by the fourth scenario; two yaw doublets. On the other hand, the second scenario, [Figure 5-23 and Figure 5-24] – two pitch doublets, was the most sensitive to bias, both positive and negative, followed by the fifth scenario; climbing with high AOA.

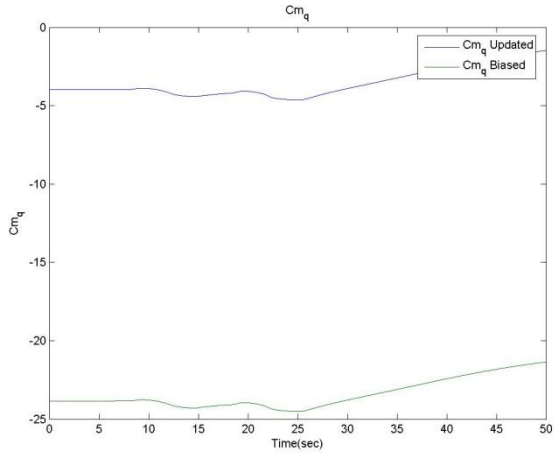

Figure 5-23: Updated vs. biased (500%) Cm_q- Second scenario

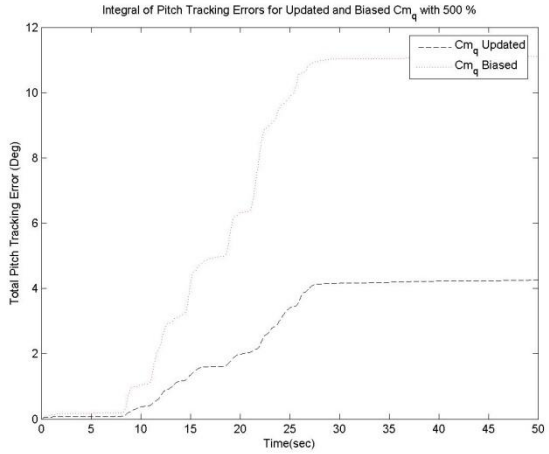

Figure 5-24: Integral of pitch tracking errors for Updated vs. biased (500%) Cm_q- Second scenario

Table 5-9: Updated vs. biased C_{mq} - Average results

Bias (%)	Integral of Pitch Tracking Error(deg)	Integral of Roll Tracking Error(deg)	Integral of Yaw Tracking Error(deg)	Integral of Pitch Neural Network	Integral of Roll Neural Network	Integral of Yaw Neural Network
0	100.00	100.00	100.00	100.00	100.00	100.00
5	100.75	100.01	100.01	99.89	100.00	100.00
10	101.50	100.02	100.02	99.76	100.00	100.00
15	102.25	100.03	100.03	99.64	100.00	100.01
20	103.00	100.03	100.04	99.55	100.00	100.01
25	103.76	100.04	100.06	99.42	100.00	100.01
30	104.52	100.05	100.06	99.32	100.00	100.01
50	107.59	100.08	100.11	98.88	100.00	100.03
80	111.49	100.13	100.17	99.08	100.01	100.04
100	114.52	100.16	100.21	98.72	100.01	100.05
120	117.76	100.19	100.25	98.36	100.01	100.06
150	122.96	100.24	100.33	98.69	100.01	100.07
200	131.48	100.29	100.42	95.89	100.01	100.11
250	140.47	100.38	100.53	97.00	100.01	100.14
300	150.16	100.46	100.67	95.31	100.01	100.22
350	159.60	100.55	100.77	93.58	100.04	100.27
400	172.39	100.63	100.88	94.24	100.05	100.31
450	188.03	100.71	100.99	95.60	100.05	100.36
500	207.94	100.78	101.05	97.88	100.05	100.38
-5	99.05	100.00	100.00	100.21	100.00	100.00
-10	98.31	99.99	99.98	100.32	100.00	99.99
-15	97.55	99.99	99.97	100.45	100.00	99.99
-20	96.62	99.99	99.97	100.80	100.00	99.99
-25	95.85	99.98	99.96	101.02	100.00	99.99
-30	95.05	99.98	99.95	101.14	100.00	99.98
-50	92.51	99.94	99.90	101.38	100.00	99.97
-80	88.46	99.88	99.83	102.25	100.00	99.96
-100	86.04	99.84	99.79	102.91	99.99	99.95
-120	83.82	99.80	99.75	103.37	99.99	99.94
-150	81.61	99.73	99.67	103.80	99.99	99.92
-200	80.99	99.64	99.56	105.00	99.99	99.90
-250	84.05	99.55	99.45	106.37	99.99	99.87
-300	89.53	99.46	99.34	107.80	99.98	99.85
-350	94.12	99.39	99.23	109.65	99.98	99.83
-400	100.06	99.31	99.11	111.90	99.98	99.80
-450	107.40	99.20	99.00	112.71	99.98	99.78
-500	113.95	99.10	98.89	114.41	99.98	99.76

3- $Cm_{\delta s}$

I) Updated Derivative Against Fixed Derivative

Generally, tracking each of the three reference rates does not improve by updating $C_{m_{\delta s}}$ as compared to fixing $C_{m_{\delta s}}$. First, [Table 5-10], and fifth scenarios were the only exception to that, where the pitch tracking improved by about 17.5%, [Figure 5-26]. Second, [Table 5-12], and sixth scenarios, on the other hand, had better performance when the derivative is fixed. Finally, the neural networks in the three channel were unaffected by this derivative in all the scenarios. [Figure 5-25] shows both updated and fixed $C_{m_{\delta s}}$.

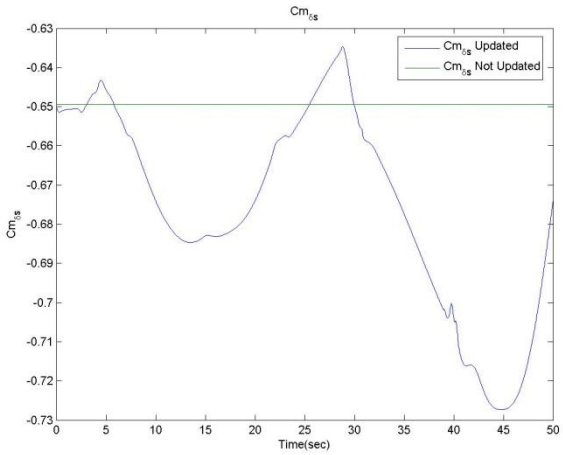

Figure 5-25: Updated vs. fixed $Cm_{\delta s}$- Fifth scenario

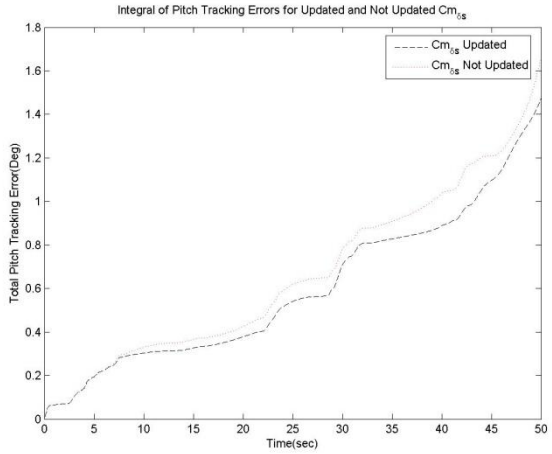

Figure 5-26: Integral of pitch tracking errors for updated vs. fixed $Cm_{\delta s}$- Fifth scenario

Table 5-10: Updated vs. fixed $Cm_{\delta s}$ - First scenario

	Updated $C_{m_{\delta s}}$	Fixed $C_{m_{\delta s}}$
Integral of Pitch Tracking Error (%)	100	117.57
Integral of Roll Tracking Error (%)	100	100.01
Integral of Yaw Tracking Error (%)	100	100.02
Integral of Pitch Neural Network (%)	100	91.047
Integral of Roll Neural Network (%)	100	99.946
Integral of Yaw Neural Network (%)	100	99.965

Table 5-11: Updated vs. fixed $Cm_{\delta s}$- Second scenario

	Updated $C_{m_{\delta s}}$	Fixed $C_{m_{\delta s}}$
Integral of Pitch Tracking Error (%)	100.00	82.75
Integral of Roll Tracking Error (%)	100.00	99.02
Integral of Yaw Tracking Error (%)	100.00	99.57
Integral of Pitch Neural Network (%)	100.00	101.04
Integral of Roll Neural Network (%)	100.00	100.02
Integral of Yaw Neural Network (%)	100.00	100.02

II) Updated Derivative Against Delayed Derivative

The average results of the six scenarios show slight difference between the ideal update and delayed update. The pitch tracking error will start increasing when the delay is 5 seconds, but the increase is very little. Moreover, roll and yaw tracking errors as well as neural networks effort in the three channels do not change.

At first scenario, starting from 4 seconds delay, the pitch tracking error starts to increase consistently. Therefore, the estimation process for this derivative is inefficient if it produces estimates with 4 seconds delay. The error reaches its maximum value of 21% more than the ideal case when the delay is 20 seconds.

Fifth scenario, [Table 5-12], was similar to the first one, yet more sensitive; this scenario was the most sensitive scenario with respect to this derivative estimation delay. Half a second delay was enough to start producing more error than the ideal case, and a second delay results in 5% more error. The maximum increase in error, of about 31%, occurs for a 10 seconds delay, as shown in Figure 5-28.

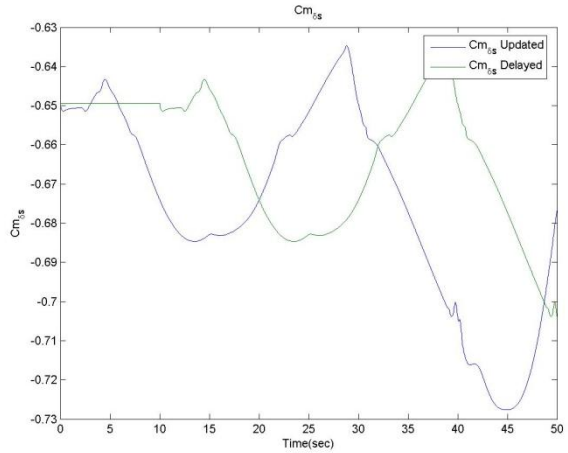

Figure 5-27: Updated vs. delayed (10-seconds) $Cm_{\delta s}$- Fifth scenario

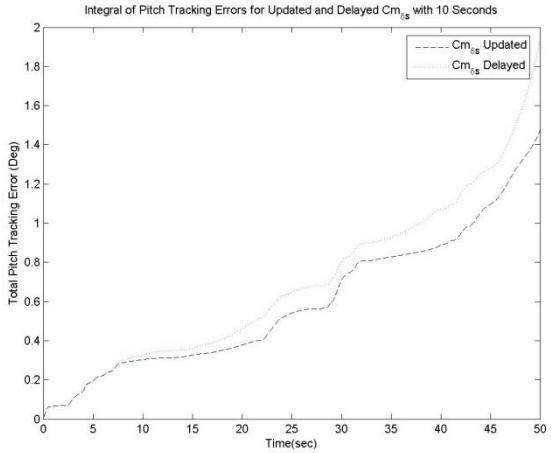

Figure 5-28: Integral of pitch tracking errors for updated vs. delayed (10-seconds) $Cm_{\delta s}$- Fifth scenario

Table 5-12: Updated vs. delayed $Cm_{\delta s}$– Fifth scenario

Delay (sec)	0	0.5	1	2	3	4	5	7	9	10	12	15	17	20	25	30	35	40
Integral of Pitch Tracking Error(deg)	100	101.7	105.5	116	120.3	123.6	126.8	130.3	129.3	131.4	127.5	126.7	121	108.6	104.9	112.2	122.9	121.9
Integral of Roll Tracking Error(deg)	100	99.98	99.96	99.98	100	99.95	100	99.99	100	100.1	100	99.97	100.1	100.1	100.1	100.2	100.1	100.1
Integral of Yaw Tracking Error(deg)	100	100	100.1	100.3	100.1	100.3	100.5	100.5	100.6	100.6	100.8	100.9	100.7	100.8	100.8	100.7	100.8	101
Integral of Pitch Neural Network	100	99.84	99.67	99.36	99.07	98.81	98.53	98.19	97.89	97.8	97.68	97.62	97.62	97.58	97.6	97.65	97.68	97.65
Integral of Roll Neural Network	100	100	99.99	99.99	99.98	99.98	99.99	99.99	100	100	100	100	100.1	100.1	100.1	100.1	100	100.1
Integral of Yaw Neural Network	100	100	100.1	100.2	100.2	100.3	100.4	100.5	100.5	100.6	100.6	100.5	100.5	100.4	100.4	100.4	100.7	100.7

III) Updated Derivative Against Biased Derivative

In average, increasing the bias value affects both the quality of tracking pitch rate and the pitch neural network. In contrast, this derivative has negligible effect on roll and yaw rates and their neural networks. The controller is more sensitive to negative biases than to positive values. In average, the system becomes unstable when the bias value is -100% and in some scenarios, instability occurs from -80% bias. This derivative is better being fixed than using a biased derivative with more than 50% or -15% bias.

The pitch neural network increases proportionally with positive bias growth; nevertheless, the negative bias did not have as much effect on the pitch NN. This derivative does not have any effect on roll and yaw NNs. On the other hand and to a smaller scale, the quality of tracking roll and yaw rates experienced small degradation with negative bias growth, but this effect was not present with positive bias increase.

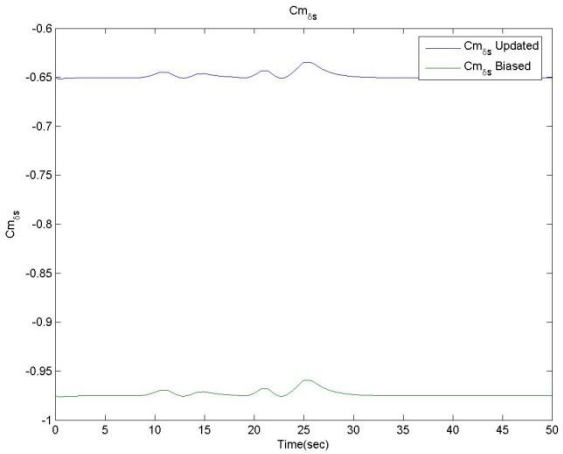

Figure 5-29: Updated vs. biased (50%) $Cm_{\delta s}$- Fourth scenario

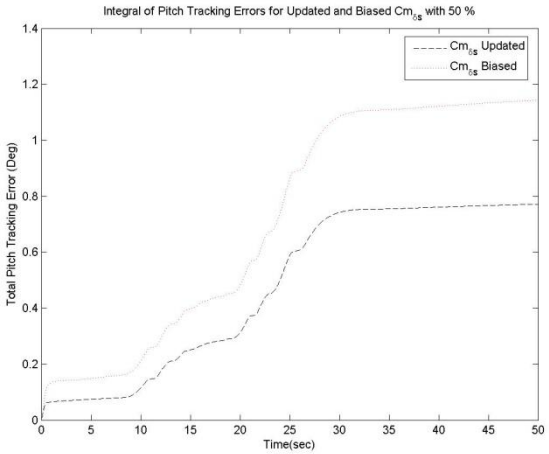

Figure 5-30: Integral of pitch tracking error for updated vs. biased (50%) $Cm_{\delta s}$- Fourth scenario

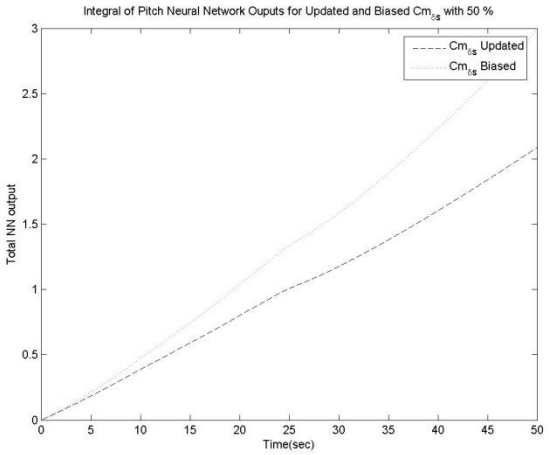

Figure 5-31: Integral of pitch neural network output for updated vs. biased (50%) $Cm_{\delta s}$- Fourth scenario

Table 5-13: Updated vs. biased $Cm_{\delta s}$– Average results

Bias (%)	Integral of Pitch Tracking Error(deg)	Integral of Roll Tracking Error(deg)	Integral of Yaw Tracking Error(deg)	Integral of Pitch Neural Network	Integral of Roll Neural Network	Integral of Yaw Neural Network
0	100.00	100.00	100.00	100.00	100.00	100.00
5	95.15	99.82	99.75	102.60	100.00	99.92
10	92.05	99.62	99.50	104.82	100.00	99.85
15	90.89	99.43	99.25	107.24	100.00	99.78
20	91.61	99.24	99.01	109.66	100.00	99.70
25	94.20	99.05	98.76	111.45	100.00	99.64
30	99.17	98.87	98.53	113.26	100.00	99.57
50	134.68	98.15	97.62	124.61	100.00	99.31
80	198.64	97.12	96.39	146.91	100.00	98.98
100	241.84	96.43	95.63	161.65	100.01	98.78
120	279.20	95.81	94.89	191.52	99.96	98.57
150	344.01	94.92	93.84	218.80	99.96	98.35
200	434.32	93.43	92.12	260.01	99.95	97.98
250	516.61	92.10	90.63	303.55	99.96	97.69
300	592.24	90.87	89.37	343.31	99.97	97.45
350	662.12	89.75	88.32	380.92	99.98	97.25
400	726.45	88.73	87.41	418.27	99.98	97.08
450	785.09	87.81	86.60	457.08	99.99	96.91
500	840.07	86.95	85.90	492.67	100.00	96.77
-5	105.32	100.20	100.26	97.86	100.00	100.08
-10	109.38	100.38	100.50	97.18	100.00	100.15
-15	115.82	100.59	100.77	95.34	100.00	100.24
-20	124.60	100.80	101.06	92.73	100.01	100.33
-25	133.44	101.01	101.34	91.13	100.01	100.42
-30	142.93	101.18	101.61	88.57	100.02	100.51
-50	187.13	102.13	102.80	94.62	100.03	100.89
-80	3906.04	280.62	477.04	118.10	91.65	102.89
-100	unstable	unstable	unstable	unstable	unstable	unstable
-120	unstable	unstable	unstable	unstable	unstable	unstable
-150	unstable	unstable	unstable	unstable	unstable	unstable
-200	unstable	unstable	unstable	unstable	unstable	unstable
-250	unstable	unstable	unstable	unstable	unstable	unstable
-300	unstable	unstable	unstable	unstable	unstable	unstable
-350	unstable	unstable	unstable	unstable	unstable	unstable
-400	unstable	unstable	unstable	unstable	unstable	unstable
-450	unstable	unstable	unstable	unstable	unstable	unstable
-500	unstable	unstable	unstable	unstable	unstable	unstable

4- C_{l_β}

I) Updated Derivative Against Fixed Derivative

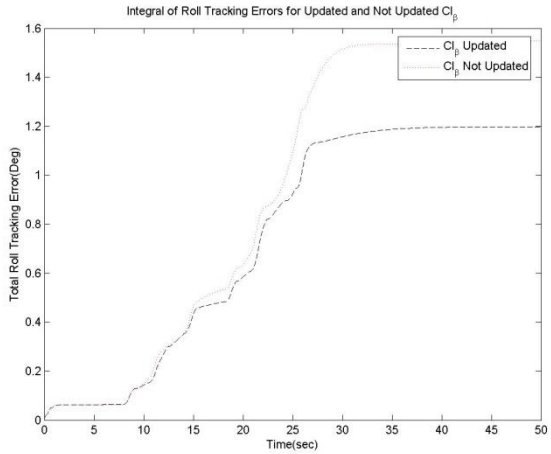

Figure 5-32: Integral of roll tracking error for updated vs. fixed C_{l_β}- Fourth scenario

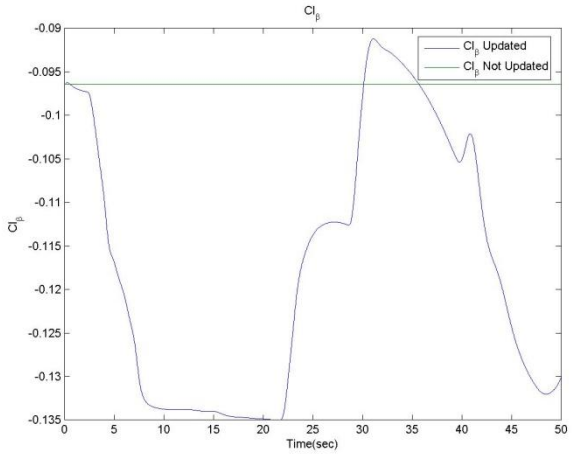

Figure 5-33: Updated vs. fixed C_{l_β}- Fourth scenario

[Figure 5-33], shows both updated and fixed C_{l_β}. This is a lateral derivative; therefore, updating C_{l_β} does not improve the quality of pitch rate tracking, or reduce the pitch NN effort. As expected, because C_{l_β} directly connects the side slip angle to the roll moment, updating this derivative enhances the tracking of roll angular rate by about 12% in average, as listed in Figure 5-14.

The average neural network effort from all the scenarios does not increase when the derivative was fixed; however, the following detailed analysis of each scenario will show that the increase occurs in some of the scenarios. The roll-NN, in third scenario (two roll doublets) and fourth scenario (two yaw doublets), had to increase the effort when the derivative is fixed. Moreover, the fourth scenario experienced the most significant performance degradation on roll channel when the derivative is fixed by 30% more error, [Table 5-15 and Figure 5-32]. The yaw tracking quality was affected in some of the scenarios, but the error scale was very small; around 3%. This derivative did not have any effect on the yaw-NN.

5-14: Updated vs. fixed C_{l_β}– Average results

	Updated C_{l_β}	Fixed C_{l_β}
Integral of Pitch Tracking Error (%)	100.00	100.01
Integral of Roll Tracking Error (%)	100.00	112.25
Integral of Yaw Tracking Error (%)	100.00	101.83
Integral of Pitch Neural Network (%)	100.00	100.02
Integral of Roll Neural Network (%)	100.00	99.60
Integral of Yaw Neural Network (%)	100.00	100.11

Table 5-15: Updated vs. fixed C_{l_β} - Fourth scenario

	Updated C_{l_β}	Fixed C_{l_β}
Integral of Pitch Tracking Error (%)	100	100.33
Integral of Roll Tracking Error (%)	100	129.32
Integral of Yaw Tracking Error (%)	100	101.15
Integral of Pitch Neural Network (%)	100	99.966
Integral of Roll Neural Network (%)	100	104.52
Integral of Yaw Neural Network (%)	100	100.03

II) **Updated Derivative Against Delayed Derivative**

[Table 5-16], shows that updating Cl_β with a delay increases the error in tracking the desired roll rate. A second delay causes an 8% increase in the roll tracking error, which is less than the resulting error from completely fixing the derivative. Any delay greater than a second will be worse than not updating Cl_β. At 25 seconds delay, the error hits its maximum of 26% more than the ideal case.

Detailed analysis of the scenarios shows that the maximum error in tracking the roll rate occurred in the fourth scenario by a 53% more error at 3 seconds delay, [Figure 5-34, and Figure 5-35].

The average results do not present a significant degradation on tracking yaw rate; however, detailed analysis shows that the fifth scenario had a very sensitive yaw rate tracking quality towards the update timing of Cl_β such that a second delay was enough to cause an 11% error, [Figure 5-36, and Figure 5-37].

The delay does not cause any change in the tracking quality of pitch rate. Finally, the general effect of delayed Cl_β on the neural networks varied from an increase in the roll-NN effort to no effect on the yaw or pitch NNs.

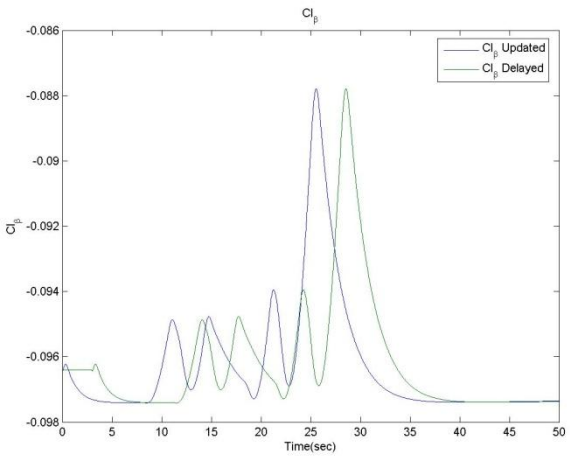

Figure 5-34: Updated vs. delayed (3-seconds) C_{l_β}- Fourth scenario

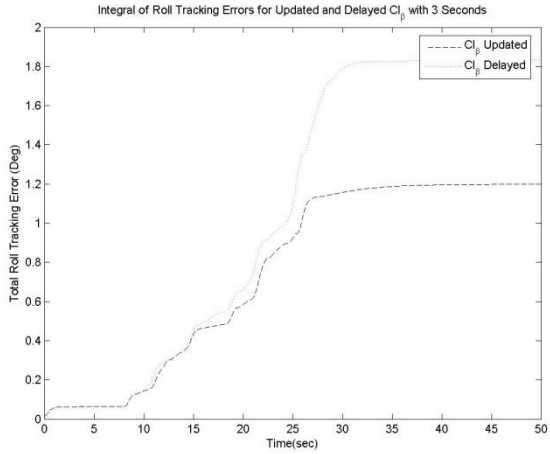

Figure 5-35: Integral of roll tracking error for updated vs. delayed (3-seconds) C_{l_β}- Fourth scenario

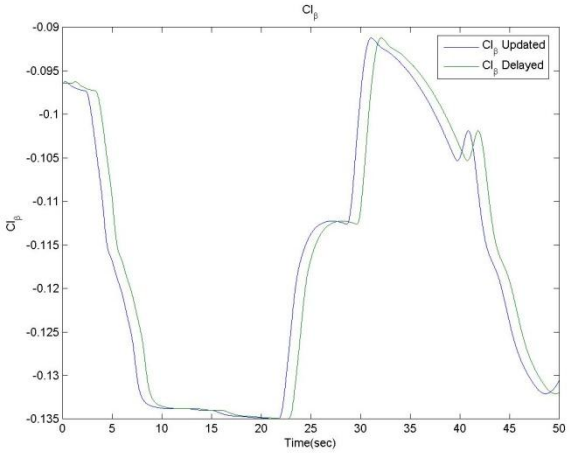

Figure 5-36: Updated vs. delayed (1-second) C_{l_β}- Fifth scenario

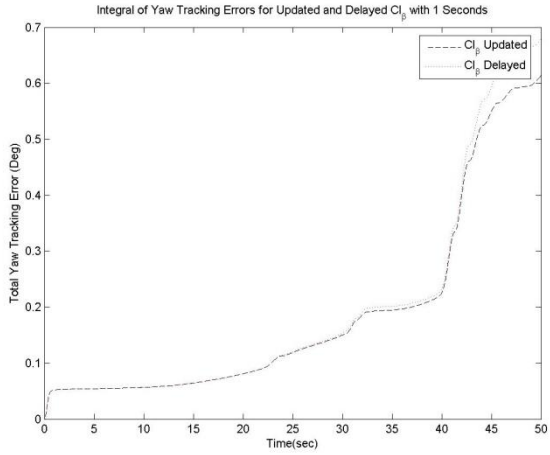

Figure 5-37: Integral of yaw tracking error for updated vs. delayed (1-second) C_{l_β}- Fifth scenario

Table 5-16: Updated vs. delayed C_{l_β}– Average results

Delay (sec)	0	0.5	1	2	3	4	5	7	9	10	12	15	17	20	25	30	35	40
Integral of Pitch Tracking Error(deg)	100.00	100.03	100.02	100.04	100.04	99.97	99.98	99.91	99.87	99.83	99.88	99.85	99.85	99.86	99.96	100.09	100.18	100.08
Integral of Roll Tracking Error(deg)	100.00	101.20	107.69	116.91	122.83	121.44	119.63	118.41	119.14	120.02	122.73	122.50	123.31	123.82	125.72	118.49	115.10	112.41
Integral of Yaw Tracking Error(deg)	100.00	101.89	102.80	102.00	102.66	103.12	102.59	102.31	101.79	101.83	103.04	103.41	102.89	101.58	103.18	102.19	102.52	102.20
Integral of Pitch Neural Network	100.00	100.01	100.00	99.99	100.01	100.00	100.00	99.99	100.00	100.02	100.00	100.00	99.99	99.99	100.00	100.01	100.02	100.03
Integral of Roll Neural Network	100.00	100.06	100.18	100.54	100.85	100.86	100.63	99.98	101.24	102.37	102.97	100.21	100.84	101.07	102.88	102.71	101.92	100.84
Integral of Yaw Neural Network	100.00	100.06	100.11	100.13	100.09	100.02	99.95	99.72	99.48	99.34	99.01	98.64	98.39	98.23	98.38	99.03	99.71	100.31

III) Updated Derivative Against Biased Derivative

[Table 5-17] shows that if Cl_β estimate has a 10% bias, the roll tracking error is 21% more, which is worse than not updating the derivative at all. Therefore, the estimation process becomes ineffective if it yields a bias more than 10%. Same rule applies to negative biases, where a -10% bias will result in 18% more error, [Figure 5-38, and Figure 5-39].

The biased value of this derivative affects also the yaw tracking quality, where 50% bias causes a 10% more error. Yaw-tracking quality is less affected by negative biases, where -500% bias will only result in 10% error. On the contrary, the biased derivative does not change the quality of tracking pitch rate.

With respect to the neural networks, the roll-NN effort increased proportionally with bias growth in both directions: positive and negative. Furthermore, the negative bias growth was of less effect. Additionally, the bias had small proportional effect on the yaw-NN effort; also, the negative bias increase produces more increase in the effort than positive increase. Finally, this derivative does not affect the pitch-NN.

Detailed analysis shows that the fourth scenario was the most sensitive to biased estimation in terms of roll tracking quality, where a 5% bias was enough to produce a 35% more error. Likewise, a a -10% bias produces a 105% more error. Moreover, the fifth scenario was the only scenario to show degradation in tracking pitch rate quality with bias increase.

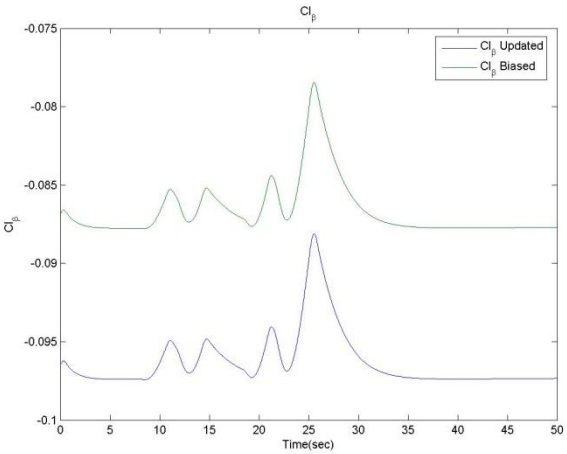

Figure 5-38: Updated vs. biased (-10%) C_{l_β}- Fourth scenario

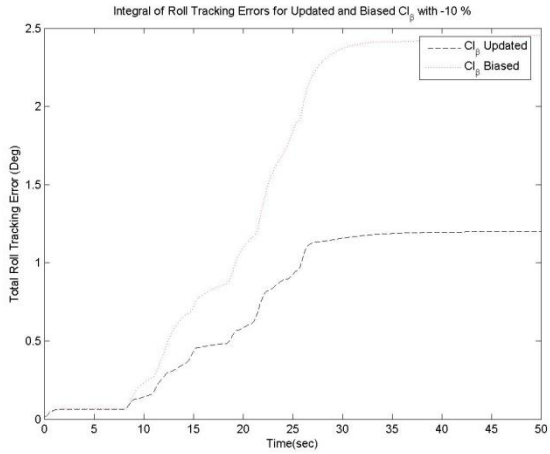

Figure 5-39: Integral of roll tracking error for updated vs. biased (-10%) C_{l_β}- Fourth scenario

IV) Bias and delay

The fourth scenario was used to test the effect of combining delay and bias. The results did not exhibit a considerable change in the errors resulted because of bias only.

Table 5-17: Updated vs. biased C_{l_β} – Average results

Bias (%)	Integral of Pitch Tracking Error(deg)	Integral of Roll Tracking Error(deg)	Integral of Yaw Tracking Error(deg)	Integral of Pitch Neural Network	Integral of Roll Neural Network	Integral of Yaw Neural Network
0	100.00	100.00	100.00	100.00	100.00	100.00
5	100.11	108.25	100.88	100.00	102.83	99.80
10	100.21	121.22	101.77	100.00	105.65	99.60
15	100.32	136.57	102.69	100.00	108.49	99.39
20	100.42	152.95	103.63	99.99	111.41	99.19
25	100.51	170.05	104.60	100.00	114.37	98.99
30	100.61	187.69	105.59	99.99	117.34	98.79
50	100.98	261.21	109.96	99.99	129.81	97.99
80	101.48	378.84	117.62	99.97	149.88	96.83
100	101.80	462.57	123.45	99.96	163.28	96.41
120	102.12	550.00	129.57	99.95	176.59	96.56
150	102.65	687.32	139.27	99.94	196.32	97.12
200	103.65	929.60	156.80	99.92	228.88	98.48
250	104.84	1182.22	175.61	99.93	262.10	99.89
300	106.26	1439.63	192.74	99.94	297.18	100.86
350	107.82	1725.26	228.05	99.98	336.03	101.14
400	109.50	2035.52	282.60	100.09	377.09	102.70
450	119.27	2366.33	347.00	100.34	419.05	106.84
500	131.36	3011.53	410.13	100.05	468.84	110.13
-5	99.88	103.64	99.15	100.01	97.16	100.20
-10	99.77	117.16	98.32	100.01	94.33	100.40
-15	99.65	131.79	97.51	100.01	91.49	100.60
-20	99.53	147.07	96.72	100.01	88.90	100.80
-25	99.41	162.68	95.94	100.01	86.92	101.00
-30	99.29	178.59	95.18	100.01	85.33	101.19
-50	99.05	243.56	92.21	100.02	81.08	101.96
-80	99.26	341.99	88.11	100.03	79.27	103.01
-100	99.57	407.52	86.13	100.04	80.55	103.64
-120	99.88	472.49	84.93	100.05	85.09	104.20
-150	100.22	568.44	84.22	100.07	95.86	104.92
-200	100.32	722.80	83.87	100.10	116.63	105.80
-250	99.91	869.35	84.09	100.15	139.39	106.36
-300	99.16	1008.26	85.44	100.21	163.64	106.67
-350	98.25	1138.70	90.25	100.28	188.71	106.78
-400	97.28	1261.14	96.41	100.37	214.08	106.74
-450	96.31	1376.00	102.92	100.47	239.99	106.60
-500	95.30	1483.43	109.88	100.61	266.95	106.39

5- C_{l_p}

I) Updated Derivative Against Fixed Derivative

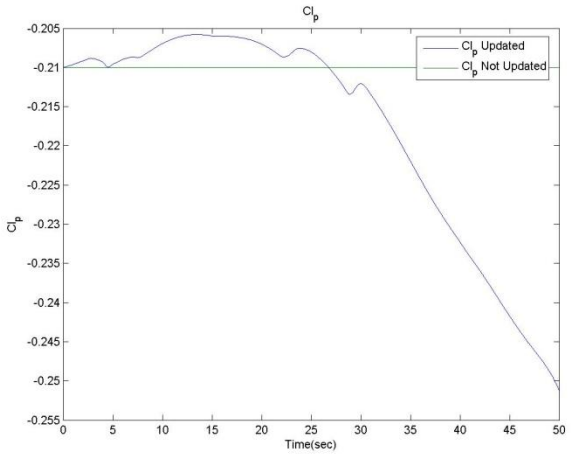

Figure 5-40: Updated vs. fixed C_{l_p}- Fifth scenario

[Figure 5-40], shows both updated and fixed C_{l_p}; this derivative is a lateral derivative. The average results show an unexpected slight improvement, by about 2%, in the tracking of roll and yaw rates when the derivative is fixed as compared to updating the derivative. The detailed analysis of every single scenario, except the first scenario, supported the aforementioned conclusion.Third, fifth, and sixth scenarios experienced greater roll and yaw tracking quality when C_{l_p} is fixed. On the other hand, the first scenario was the only scenario to show a very small improvement in the roll-tracking when updating the derivative; 3% error reduction, [Table 5-18].

Table 5-18: Updated vs. fixed C_{l_p} - First scenario

	Updated C_{l_p}	Fixed C_{l_p}
Integral of Pitch Tracking Error (%)	100	100
Integral of Roll Tracking Error (%)	100	102.9
Integral of Yaw Tracking Error (%)	100	100.11
Integral of Pitch Neural Network (%)	100	99.998
Integral of Roll Neural Network (%)	100	100.15
Integral of Yaw Neural Network (%)	100	100

Table 5-19: Updated vs. fixed C_{l_p}- Fifth scenario

	Updated C_{l_p}	Fixed C_{l_p}
Integral of Pitch Tracking Error (%)	100	99.674
Integral of Roll Tracking Error (%)	100	94.177
Integral of Yaw Tracking Error (%)	100	94.907
Integral of Pitch Neural Network (%)	100	100
Integral of Roll Neural Network (%)	100	99.914
Integral of Yaw Neural Network (%)	100	100.12

II) Updated Derivative Against Delayed Derivative

The average results of all scenarios show that the hybrid controller is insensitive to the delay in updating C_{l_p}, [Table 5-20]. In fact, the average results show no consistent difference between the ideal case and any of the delay cases. Except for the sixth scenario, which is more sensitive to the delay in this derivative in terms of roll-tracking quality. The roll tracking error was the greatest at 10 seconds delay by a 19% more error, [Figure 5-41 and Figure 5-42]. 3 seconds delay in updating the derivative during this scenario will result in 3% error. Therefore, it is recommended that if the estimation process for this scenario has a delay more than 3 seconds, the fixed derivative becomes more efficient.

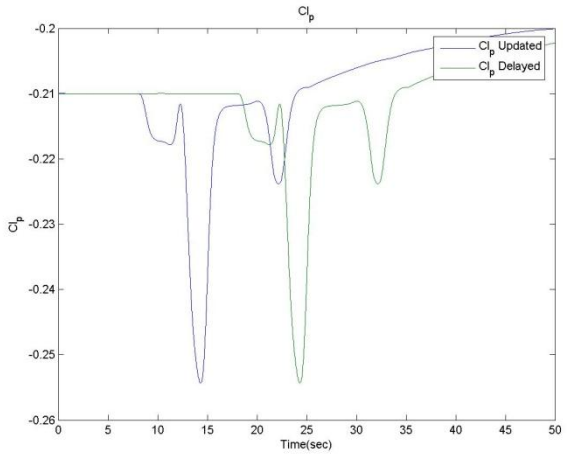

Figure 5-41: Updated vs. delayed (10-seconds) C_{l_p}- sixth scenario

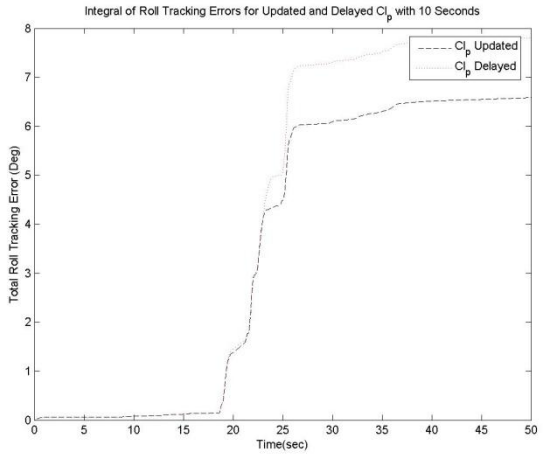

Figure 5-42: Integral of roll tracking error for updated vs. delayed (10-seconds) C_{l_p}- Sixth scenario

Table 5-20: Updated vs. delayed C_{l_p} – Average results

Delay (sec)	0	0.5	1	2	3	4	5	7	9	10	12	15	17	20	25	30	35	40
Integral of Pitch Tracking Error(deg)	100.00	99.98	99.97	99.96	100.00	99.98	99.96	99.96	99.93	99.95	99.95	99.93	99.92	99.92	99.91	99.92	99.92	99.92
Integral of Roll Tracking Error(deg)	100.00	100.08	100.31	100.05	100.55	99.10	99.91	101.12	102.15	102.47	100.35	98.63	98.17	98.16	98.14	98.16	98.25	98.27
Integral of Yaw Tracking Error(deg)	100.00	99.73	99.33	98.54	99.25	99.14	98.78	99.17	99.04	98.87	98.54	98.37	98.30	98.29	98.25	98.27	98.35	98.37
Integral of Pitch Neural Network	100.00	100.01	100.01	100.03	100.02	100.01	100.01	100.01	100.02	100.02	100.03	100.01	100.02	100.02	100.02	100.02	100.02	100.02
Integral of Roll Neural Network	100.00	99.87	99.80	99.76	99.84	100.11	100.58	100.62	99.98	99.85	99.78	99.96	100.01	100.01	100.01	100.01	100.01	100.01
Integral of Yaw Neural Network	100.00	100.00	99.99	100.00	100.02	100.01	100.02	100.02	100.01	100.02	100.03	100.03	100.03	100.03	100.03	100.03	100.03	100.02

III) Updated Derivative Against Biased Derivative

The average results show that updating the derivative with only 5% bias results in a 4% more error in tracking roll rate, [Figure 5-43 and Figure 5-44]. Even though the error is small, but because the fixed derivative produces no error, any bias greater than 5% will yield lower tracking efficiency.

Pitch rate tracking quality is decreasing proportionally with positive bias growth. On the contrary, the negative bias increase does not have the same effect on pitch tracking. Yaw tracking quality is affected also by bias increase in both directions; however, it is more sensitive to positive bias increase than negative bias.

Roll-NN effort increases proportionally with the bias increase in both signs. Similarly, pitch and yaw NNs effort increases with positive bias growth; nonetheless, the negative bias growth does not affect these two neural networks.

In general, the hybrid controller seems to be more sensitive to positive bias increase than to negative bias in terms of stability and error development.

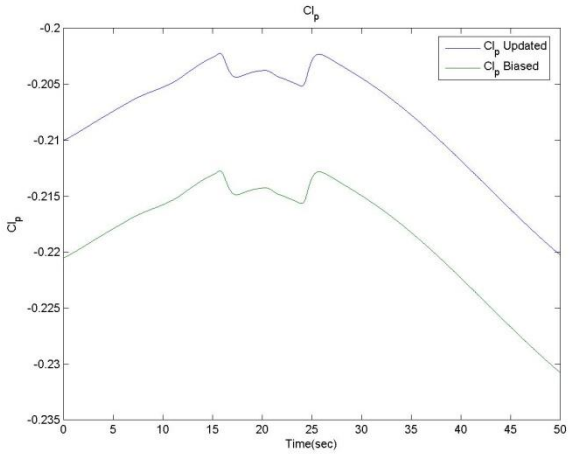

Figure 5-43: Updated vs. biased (5%) C_{l_p}- First scenario

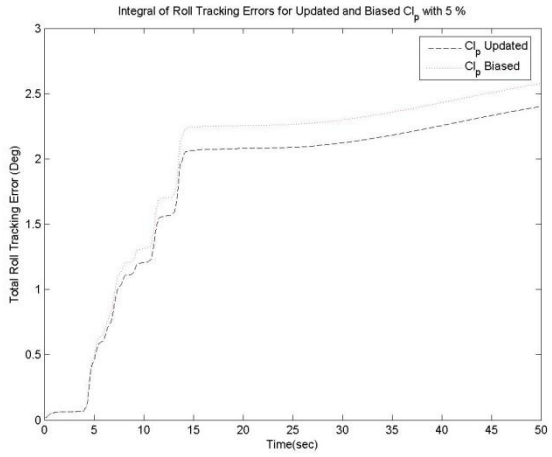

Figure 5-44: Integral of roll tracking error for updated vs. biased (5%) C_{l_p}- First scenario

Table 5-21: Updated vs. biased C_{l_p} – Average results

Bias (%)	Integral of Pitch Tracking Error(deg)	Integral of Roll Tracking Error(deg)	Integral of Yaw Tracking Error(deg)	Integral of Pitch Neural Network	Integral of Roll Neural Network	Integral of Yaw Neural Network
0	100.00	100.00	100.00	100.00	100.00	100.00
5	100.06	104.83	101.25	100.02	100.15	100.00
10	100.12	109.83	102.59	100.04	100.31	100.01
15	100.18	114.98	104.00	100.05	100.49	100.01
20	100.24	120.26	105.48	100.07	100.67	100.02
25	100.31	125.68	107.04	100.08	100.88	100.02
30	100.37	131.24	108.68	100.10	101.12	100.03
50	100.65	154.97	115.99	100.16	102.49	100.05
80	101.16	196.41	129.81	100.25	105.61	100.06
100	101.68	229.67	141.82	100.30	107.97	100.07
120	102.41	269.86	157.27	100.35	110.45	100.07
150	104.00	350.98	190.39	100.43	114.56	100.04
200	110.32	583.77	293.70	100.52	123.38	100.23
250	113.95	1587.12	365.93	100.63	135.35	100.72
300	7971.51	58088.38	12288.45	210.07	1899.46	120.23
350	30718.33	101759.00	33569.82	1741.16	8871.68	123.85
400	24048.87	116656.67	27483.12	467.04	10497.15	155.46
450	47143.20	177616.80	40633.00	2683.16	11045.70	139.65
500	unstable	unstable	unstable	unstable	unstable	unstable
-5	99.94	95.34	98.84	99.99	99.86	100.00
-10	99.88	90.88	97.80	99.97	99.80	99.99
-15	99.83	86.66	96.86	99.95	99.80	99.99
-20	99.77	82.71	96.05	99.94	99.85	99.98
-25	99.71	79.18	95.50	99.92	99.96	99.98
-30	99.66	76.17	95.09	99.90	100.13	99.97
-50	99.44	71.49	94.17	99.84	101.12	99.95
-80	99.13	88.11	94.96	99.74	103.09	99.92
-100	98.93	105.18	97.32	99.68	105.01	99.90
-120	98.73	122.87	100.72	99.61	107.37	99.88
-150	98.45	149.58	106.23	99.52	111.48	99.85
-200	98.00	193.94	116.32	99.38	118.95	99.80
-250	97.59	237.65	127.25	99.25	126.90	99.75
-300	97.21	280.63	138.61	99.13	135.23	99.71
-350	96.87	322.64	150.15	99.03	143.84	99.67
-400	96.57	363.74	161.75	98.94	152.70	99.63
-450	96.29	403.97	173.34	98.87	161.76	99.60
-500	95.97	443.17	184.86	98.81	171.01	99.55

6- $C_{\ell_{\delta a}}$

I) Updated Derivative Against Fixed Derivative

[Figure 5-45] shows both updated and fixed $C_{\ell_{\delta a}}$.

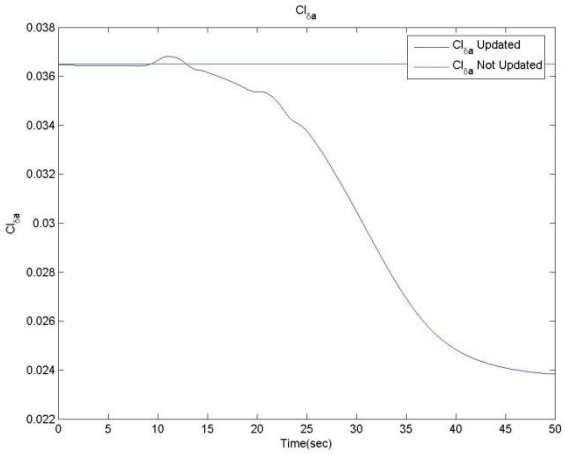

Figure 5-45: Updated vs. fixed $C_{l_{\delta a}}$- Third scenario

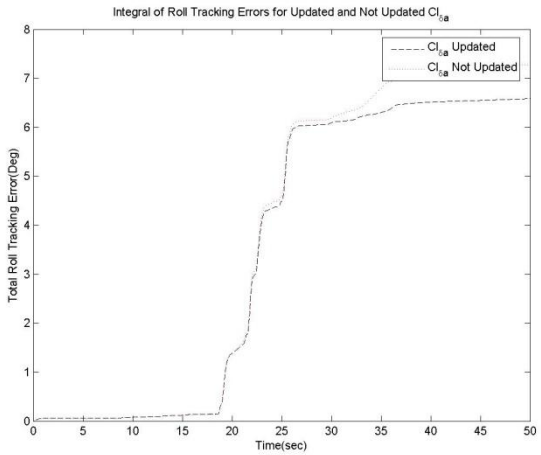

Figure 5-46: Integral of roll tracking error for updated vs. fixed $C_{l_{\delta a}}$- Sixth scenario

The average results show that the roll tracking quality improved by about 2% when updating the derivative. Moreover, detailed analysis shows that the sixth scenario, [Table 5-22], had the greatest improvement of 11%, [Figure 5-46], followed by the fifth scenario by a 7% improvement in a roll tracking quality.

Table 5-22: Updated vs. fixed $C_{\ell_{\delta a}}$ – Sixth scenario

	Updated $C_{\ell_{\delta a}}$	Fixed $C_{\ell_{\delta a}}$
Integral of Pitch Tracking Error (%)	100	99.964
Integral of Roll Tracking Error (%)	100	110.54
Integral of Yaw Tracking Error (%)	100	97.443
Integral of Pitch Neural Network (%)	100	100.08
Integral of Roll Neural Network (%)	100	95.641
Integral of Yaw Neural Network (%)	100	99.916

II) Updated Derivative Against Delayed Derivative

The average results show that the delay does not have a dangerous effect on roll tracking quality since updating the derivative with 7 seconds delay will increase the error in tracking roll rate by 2%, which is worse than not updating the derivative, [Table 5-23]. That means if the estimation process takes longer than 5 seconds to converge, the whole estimation process is not likely to improve the performance of the controller. Delay does not show any effect on any of the other five parameters.

The detailed analysis shows that the fifth and sixth scenarios were the two most scenarios affected by the delay in terms of roll tracking quality. The fifth scenario develops 11% error in tracking roll rate at only 2 seconds delay. Likewise, the sixth scenario obtains the greatest error of 15% when there is a 17 seconds delay, [Figure 5-47 and Figure 5-48].

The only scenario that shows significant increase in the roll-NN effort, in order to compensate for the error, was the fifth scenario. The maximum effort increase is 16% and occurs at 30 seconds delay. The effect of the delay in updating this derivative on pitch and yaw tracking and their neural networks effort was negligible.

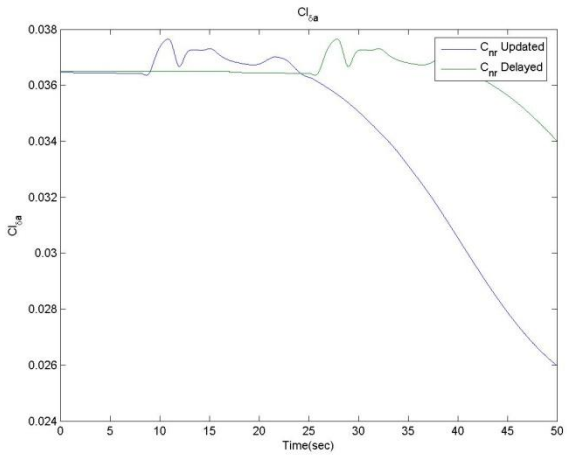

Figure 5-47: Updated vs. delayed (17-seconds) $C_{l_{\delta a}}$- Sixth scenario

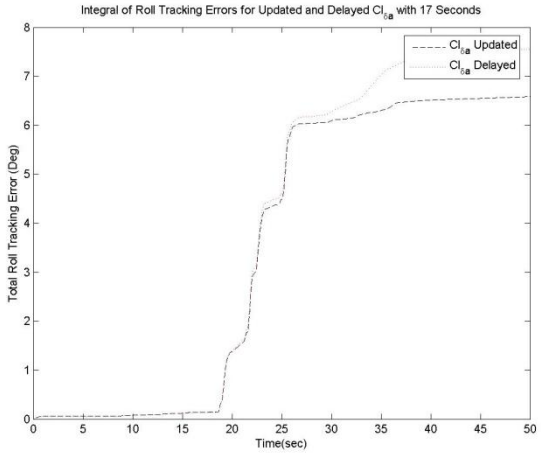

Figure 5-48: Integral of roll tracking error for updated vs. delayed (17-seconds) $C_{l_{\delta a}}$- Sixth scenario

Table 5-23: Updated vs. delayed $C_{\ell_{\delta a}}$ – Average results

Delay (sec)	0	0.5	1	2	3	4	5	7	9	10	12	15	17	20	25	30	35	40
Integral of Pitch Tracking Error(deg)	100.00	100.01	100.03	100.05	100.05	100.07	100.07	100.06	100.05	100.03	100.02	100.03	100.06	100.09	100.16	100.23	100.21	100.13
Integral of Roll Tracking Error(deg)	100.00	100.27	100.60	100.88	101.18	101.25	101.47	101.33	102.01	101.68	101.94	102.06	102.14	102.07	101.95	101.99	102.22	102.18
Integral of Yaw Tracking Error(deg)	100.00	99.60	99.57	100.01	99.87	99.68	99.66	99.27	98.99	98.76	98.71	98.49	98.61	98.60	98.33	98.56	98.59	98.61
Integral of Pitch Neural Network	100.00	100.00	99.99	99.98	99.98	99.98	99.97	99.96	99.95	99.95	99.94	99.93	99.92	99.91	99.91	99.92	99.93	99.93
Integral of Roll Neural Network	100.00	99.94	99.91	99.87	99.85	99.75	99.61	99.35	99.11	99.06	99.09	98.82	98.80	98.69	98.47	98.58	98.32	97.86
Integral of Yaw Neural Network	100.00	99.93	99.88	99.81	99.77	99.75	99.72	99.64	99.52	99.44	99.26	98.94	98.76	98.59	98.68	99.06	99.47	99.67

III) Updated Derivative Against Biased Derivative

The roll tracking quality decreases proportionally with bias increase. A 5% bias is enough to increase the error by 13%; likewise, a -5% bias produces a 20% more error, [Table 5-24]. This fact indicates that if the bias cannot be limited within 5%, the derivative with biased update will be worse than not updating the derivative at all. Fourth scenario experiences the strongest roll tracking degradation with increasing the bias, [Figure 5-49, and Figure 5-50].

With respect to roll-NN effort, at less than 80% bias, the effort does not increasing. At higher biases, nevertheless, the roll-NN effort increases proportionally with the bias increase. On the other hand, the roll-NN effort started increasing proportionally with the negative bias growth starting from -5% until the system became unstable at -100% bias.

At low bias, the bias increase does not have an effect on either pitch or yaw tracking qualities; nonetheless, when the bias value grows higher, both tracking qualities start to degrade proportionally until the system becomes unstable. In all the cases, the bias does not present considerable effect on pitch and yaw neural networks.

Generally, the hybrid controller was more sensitive to negative bias growth than to positive bias; the system becomes unstable when the bias reaches -100%, besides the error growth slope is steeper with negative bias increase.

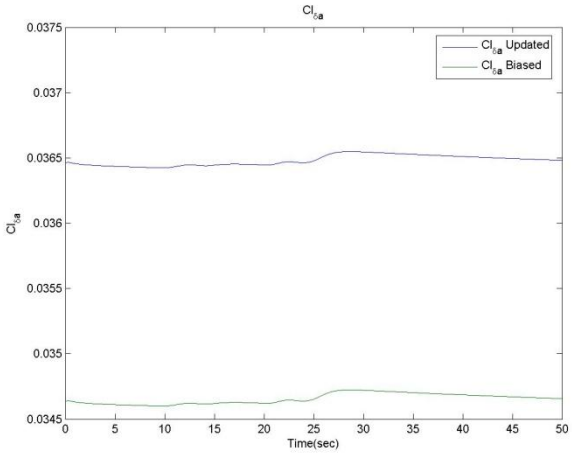

Figure 5-49: Updated vs. biased (-5%) $C_{l_{\delta a}}$- Fourth scenario

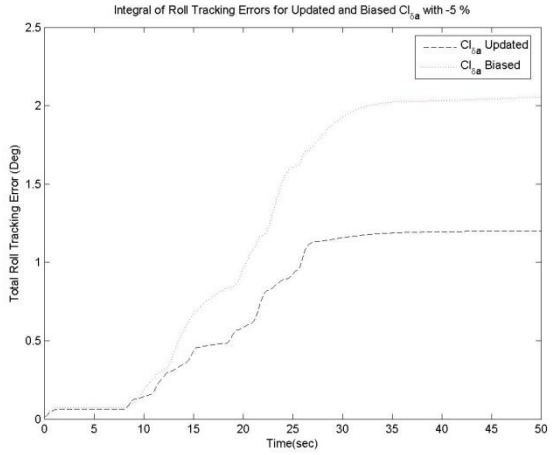

Figure 5-50: Integral of roll tracking error for updated vs. biased (-5%) $C_{l_{\delta a}}$- Fourth scenario

Table 5-24: Updated vs. biased $C_{\ell_{\delta a}}$ – Average results

Bias (%)	Integral of Pitch Tracking Error(deg)	Integral of Roll Tracking Error(deg)	Integral of Yaw Tracking Error(deg)	Integral of Pitch Neural Network	Integral of Roll Neural Network	Integral of Yaw Neural Network
0	100.00	100.00	100.00	100.00	100.00	100.00
5	99.74	113.51	98.45	99.99	95.97	100.28
10	99.48	138.69	97.05	99.97	92.05	100.56
15	99.22	165.53	95.79	99.96	89.55	100.84
20	98.96	193.34	94.70	99.95	88.17	101.11
25	98.75	222.11	93.71	99.94	87.37	101.38
30	98.60	251.81	92.83	99.92	86.93	101.64
50	98.48	375.84	90.57	99.89	89.81	102.59
80	98.86	569.20	91.62	99.86	109.06	103.73
100	99.03	695.75	94.27	99.86	127.21	104.27
120	99.04	817.24	97.63	99.89	146.18	104.66
150	98.71	989.61	105.16	99.97	175.22	105.00
200	97.22	1250.02	126.01	100.20	226.18	105.04
250	95.45	1479.77	147.94	100.42	280.29	104.64
300	93.64	1683.15	168.50	100.64	335.56	103.95
350	91.96	1863.51	187.82	100.86	391.06	103.10
400	90.55	2024.45	206.16	101.08	446.37	102.13
450	89.52	2191.44	225.23	101.29	503.35	101.21
500	88.79	2336.94	243.40	101.50	559.04	100.26
-5	100.25	119.50	101.72	100.01	104.04	99.71
-10	100.49	149.71	103.62	100.04	108.00	99.43
-15	100.74	182.85	105.67	100.05	111.94	99.14
-20	100.97	217.64	107.88	100.07	116.01	98.85
-25	101.20	253.87	110.26	100.08	120.41	98.56
-30	101.42	290.88	112.75	100.10	124.97	98.28
-50	102.27	444.64	123.89	100.16	143.28	97.16
-80	140.78	849.88	202.47	98.93	169.82	95.49
-100	unstable	unstable	unstable	unstable	unstable	unstable
-120	unstable	unstable	unstable	unstable	unstable	unstable
-150	unstable	unstable	unstable	unstable	unstable	unstable
-200	unstable	unstable	unstable	unstable	unstable	unstable
-250	unstable	unstable	unstable	unstable	unstable	unstable
-300	unstable	unstable	unstable	unstable	unstable	unstable
-350	unstable	unstable	unstable	unstable	unstable	unstable
-400	unstable	unstable	unstable	unstable	unstable	unstable
-450	unstable	unstable	unstable	unstable	unstable	unstable
-500	unstable	unstable	unstable	unstable	unstable	unstable

7- C_{n_β}

I) Updated Derivative Against Fixed Derivative

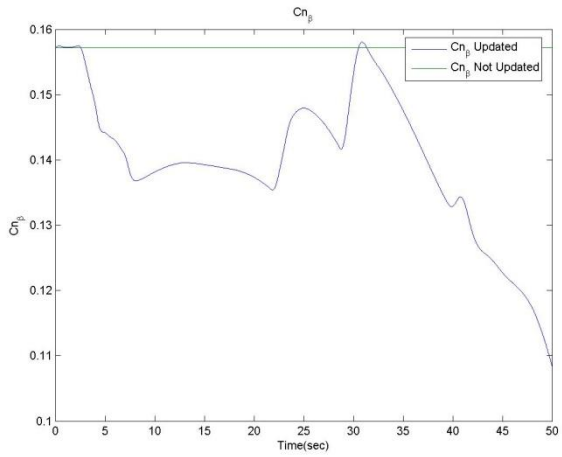

Figure 5-51: Updated vs. fixed C_{n_β}- Fifth scenario

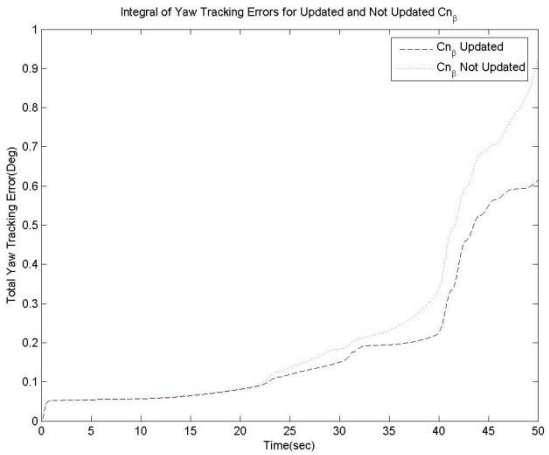

Figure 5-52: Integral of yaw tracking error for updated vs. fixed C_{n_β}- Fifth scenario

[Figure 5-51] shows both updated and fixed C_{n_β}. The average results show that updating this derivative only enhances the quality of tracking yaw rate, which improved by 3%. Detailed analysis shows that the fifth scenario utilizes updating the derivative to improve the yaw tracking quality by a 51% and to reduce the yaw-NN effort by a 7%, [Table 5-26]. Similarly, the fourth scenario shows an improvement in the yaw tracking quality by a 9% after updating the derivative, but the yaw neural network effort remained unchanged. On the contrary, the yaw tracking quality in the sixth, and third scenarios degraded after updating C_{n_β} by a 15% and a 6% respectively. Detailed analysis also supports the conclusion from the average results that all the other parameters are not significantly affected by updating this derivative.

Table 5-25: Updated vs. fixed C_{n_β}- Fifth scenario

	Updated C_{n_β}	Fixed C_{n_β}
Integral of Pitch Tracking Error (%)	100	100.09
Integral of Roll Tracking Error (%)	100	100.35
Integral of Yaw Tracking Error (%)	100	151.02
Integral of Pitch Neural Network (%)	100	100.09
Integral of Roll Neural Network (%)	100	100.11
Integral of Yaw Neural Network (%)	100	107.18

Table 5-26: Updated vs. fixed C_{n_β}- Sixth scenario

	Updated C_{n_β}	Fixed C_{n_β}
Integral of Pitch Tracking Error (%)	100	99.526
Integral of Roll Tracking Error (%)	100	99.741
Integral of Yaw Tracking Error (%)	100	84.918
Integral of Pitch Neural Network (%)	100	100.22
Integral of Roll Neural Network (%)	100	99.981
Integral of Yaw Neural Network (%)	100	99.422

II) Updated Derivative Against Delayed Derivative

The average results show that yaw rate tracking quality is the only affected parameter by delayed update of C_{n_β}. A 3 seconds delay results in a 4% more error, which is worse than not updating the derivative at all. Therefore, if the estimation process needs more than 3 seconds to converge, the fixed derivative will be more efficient. The worst average yaw tracking error of 8% occurs when the delay is 10 seconds, [Table 5-27].

The fifth scenario experiences the greatest yaw tracking quality degradation because of delay. A 2 seconds delay causes 8% error and the maximum error of a 48% occurs at a 17 seconds delay, [Figure 5-53, and Figure 5-54]. This was the only scenario that has a proportional, small increase in yaw-NN effort with delay increase as an attempt to decrease the tracking error. Followed by the sixth scenario, three seconds delay results in an error of 7%. The worst yaw tracking error of 26% occurs when the delay is 12 seconds.

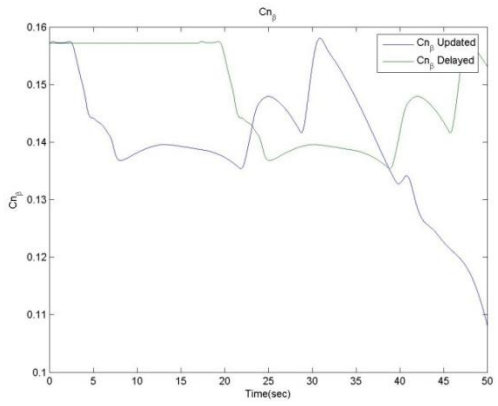

Figure 5-53: Updated vs. delayed (17-seconds) C_{n_β}- Fifth scenario

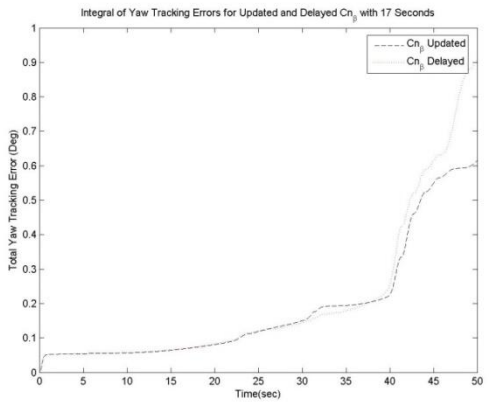

Figure 5-54: Integral of yaw tracking error for updated vs. delayed (17-seconds) C_{n_β}- Fifth scenario

Table 5-27: Updated vs. delayed C_{n_β}- Average results

Delay (sec)	0	0.5	1	2	3	4	5	7	9	10	12	15	17	20	25	30	35	40
Integral of Pitch Tracking Error(deg)	100.00	99.96	100.01	100.09	100.03	100.04	100.03	100.16	99.98	99.97	100.09	99.87	99.97	99.86	99.87	99.89	99.88	99.96
Integral of Roll Tracking Error(deg)	100.00	100.05	100.07	100.14	100.10	100.00	100.06	100.15	100.05	100.08	100.18	100.00	100.13	100.18	100.05	100.10	100.14	100.17
Integral of Yaw Tracking Error(deg)	100.00	96.62	94.66	101.58	104.31	102.34	105.80	102.26	106.65	108.44	103.94	107.54	109.67	106.40	108.45	102.18	101.04	102.12
Integral of Pitch Neural Network	100.00	99.99	100.01	100.03	100.05	100.06	100.07	100.06	100.05	100.05	100.10	100.09	100.07	100.09	100.09	100.08	100.08	100.09
Integral of Roll Neural Network	100.00	100.00	99.99	99.99	99.99	99.99	99.99	100.00	99.98	99.96	99.97	100.00	99.99	100.00	99.98	99.98	99.99	99.99
Integral of Yaw Neural Network	100.00	100.02	100.05	100.11	100.17	100.21	100.26	100.43	100.54	100.56	100.54	100.55	100.58	100.35	100.20	100.25	100.39	100.62

III) Updated Derivative Against Biased Derivative

The Yaw tracking quality inversely decreases with the bias increase. A 5% bias produces a 14% average error, [Figure 5-55, and Figure 5-56]; nevertheless, the controller is not as sensitive to the negative bias to the extent that a -30% bias had no significant average error. However, greater negative bias results in a high yaw tracking error, such that a -50% bias produces a 49% error. Therefore, if the estimation process converges to a biased value more than 5% or -30%, the entire derivative updating scheme is inefficient.

Yaw-NN effort is sensitive to positive bias growth; the NN effort increases proportionally, [Figure 5-57].Conversely, the negative bias growth does not increase the yaw-NN effort.

Negative bias growth has minor effect on pitch and roll average tracking errors as well as their neural networks average effort. On the contrary, positive bias growth reduces the quality of tracking pitch and roll rates but still does not affect their neural networks average effort.

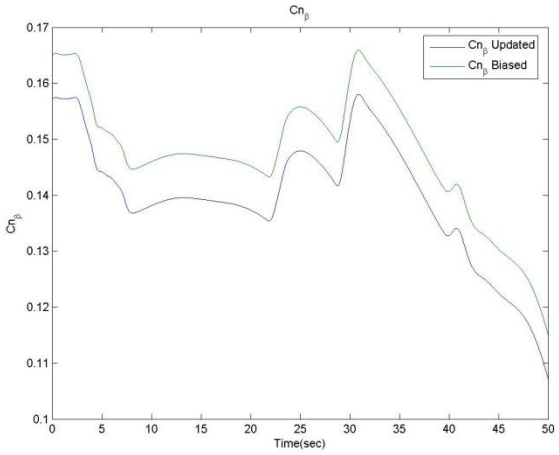

Figure 5-55: Updated vs. biased (5%) C_{n_β}- Fifth scenario

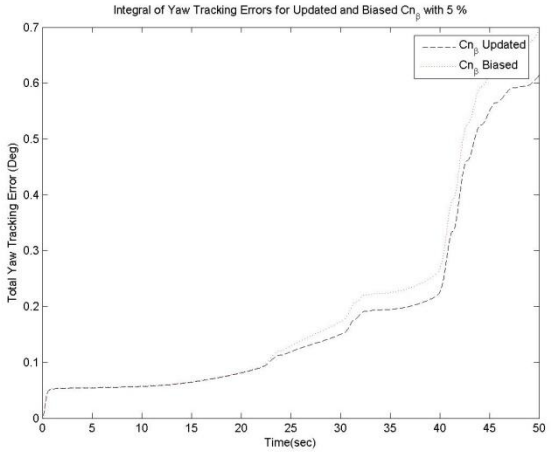

Figure 5-56: Integral of yaw tracking error for updated vs. biased (5%) C_{n_β}- Fifth scenario

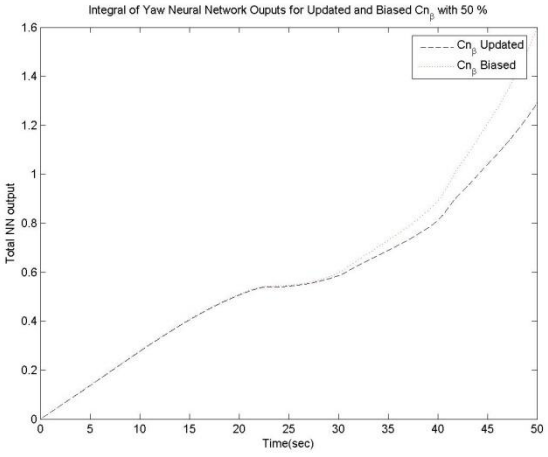

Figure 5-57: Integral of yaw neural network output for updated vs. biased (50%) C_{n_β}- Fifth scenario

Generally, the hybrid controller was more sensitive to positive bias growth than to negative bias; the system becomes unstable when the bias reaches 400%, besides the slope of the error growth is steeper with positive bias increase.

Table 5-28: Updated vs. biased C_{n_β}- Average results

Bias (%)	Integral of Pitch Tracking Error(deg)	Integral of Roll Tracking Error(deg)	Integral of Yaw Tracking Error(deg)	Integral of Pitch Neural Network	Integral of Roll Neural Network	Integral of Yaw Neural Network
0	100.00	100.00	100.00	100.00	100.00	100.00
5	100.55	100.58	114.29	99.91	100.04	100.65
10	101.12	101.19	131.10	99.80	100.09	101.33
15	101.70	101.82	149.57	99.70	100.13	102.02
20	102.31	102.49	169.22	99.59	100.17	102.74
25	102.93	103.18	189.78	99.49	100.21	103.48
30	103.57	103.91	211.21	99.37	100.25	104.23
50	106.31	107.17	306.06	98.94	100.37	108.28
80	111.51	113.49	476.35	98.08	100.50	114.81
100	116.07	118.76	612.65	97.46	100.51	118.54
120	121.51	125.13	770.01	96.80	100.47	123.52
150	133.22	136.83	1052.56	94.75	100.32	131.53
200	158.31	176.41	1713.10	92.73	99.81	147.19
250	193.73	287.35	2806.72	90.57	99.40	165.88
300	289.87	613.75	5942.32	84.41	101.29	205.36
350	1328.03	8854.75	49497.67	76.84	96.65	458.00
400	unstable	unstable	unstable	unstable	unstable	unstable
450	unstable	unstable	unstable	unstable	unstable	unstable
500	unstable	unstable	unstable	unstable	unstable	unstable
-5	99.46	99.44	89.52	100.10	99.96	99.36
-10	98.94	98.91	84.20	100.20	99.91	98.75
-15	98.44	98.40	81.74	100.28	99.86	98.15
-20	97.95	97.91	81.99	100.37	99.81	97.57
-25	97.48	97.44	89.06	100.46	99.76	97.00
-30	97.02	96.99	100.08	100.54	99.72	96.45
-50	95.32	95.38	149.08	100.86	99.51	94.41
-80	93.14	93.54	222.54	101.28	99.20	91.70
-100	91.90	92.73	267.65	101.53	98.98	90.10
-120	90.79	92.57	309.41	101.77	98.76	88.68
-150	89.33	92.84	366.08	102.08	98.43	85.81
-200	87.32	93.43	445.92	102.53	97.89	82.14
-250	85.74	94.02	511.59	102.90	97.34	80.11
-300	84.45	94.57	566.47	103.22	96.81	79.90
-350	83.42	95.08	613.07	103.48	96.29	78.96
-400	82.56	95.54	653.20	103.73	95.79	77.78
-450	81.85	95.96	688.24	103.93	95.31	76.92
-500	81.23	96.37	719.91	104.12	94.84	76.35

8- C_{n_r}

I) Updated Derivative Against Fixed Derivative

[Figure 5-58], shows both updated and fixed C_{n_r}.

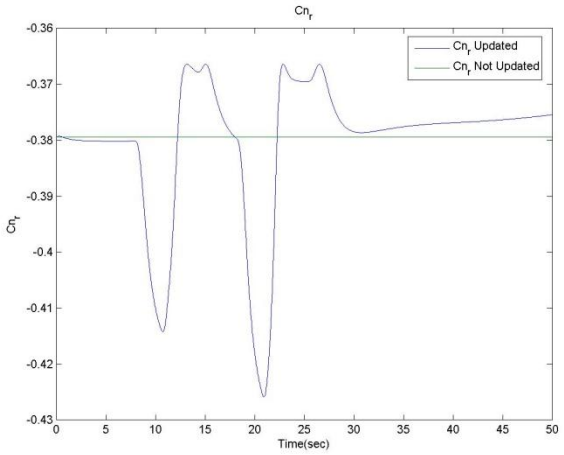

Figure 5-58: Updated vs. fixed Cn_r - Second scenario

The average results show no difference between updating and fixing the derivative. It should be noted that the range of variation of this derivative for the flight scenarios considered is rather limited. The hybrid controller performance will not improve by updating C_{n_r} within the considered range. Furthermore, the detailed analysis of each scenario supports the same conclusion.

Table 5-29: Updated vs. fixed Cn_r – Average results

	Updated C_{n_r}	Fixed C_{n_r}
Integral of Pitch Tracking Error (%)	100.00	100.00
Integral of Roll Tracking Error (%)	100.00	100.00
Integral of Yaw Tracking Error (%)	100.00	99.87
Integral of Pitch Neural Network (%)	100.00	100.00
Integral of Roll Neural Network (%)	100.00	100.00
Integral of Yaw Neural Network (%)	100.00	99.99

II) Updated Derivative Against Delayed Derivative

Since updating the derivative is not necessary, this section is not important but to keep the analysis consistent along the study. Delay in updating this derivative also does not produce any change in the performance of the hybrid controller, [Table 5-30].

Table 5-30: Updated vs. delayed Cn_r – Average results

Delay (sec)	0	0.5	1	2	3	4	5	7	9	10	12	15	17	20	25	30	35	40
Integral of Pitch Tracking Error(deg)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.99	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Integral of Roll Tracking Error(deg)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Integral of Yaw Tracking Error(deg)	100.00	99.96	99.93	99.99	100.10	100.07	99.97	99.91	99.96	99.74	99.98	99.81	99.91	99.99	100.08	99.94	100.01	99.88
Integral of Pitch Neural Network	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Integral of Roll Neural Network	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Integral of Yaw Neural Network	100.00	100.00	100.00	100.00	99.99	99.99	99.99	99.99	99.99	99.99	99.99	99.99	99.99	99.99	100.00	100.00	99.99	99.99

III) Updated Derivative Against Biased Derivative

Average results, [Table 5-31], show that yaw tracking quality is the only affected parameter by any bias in the estimation of this derivative. Yaw tracking quality degrades as the bias increases, yet the tracking quality degradation is small compared to the other derivatives effect; for instance, 500% bias develops 111% average error in tracking yaw rate, [Figure 5-59, and Figure 5-60].

Detailed analysis shows that first and fourth scenarios experience very small increase, about 4%, in the yaw neural network effort at very high biases.

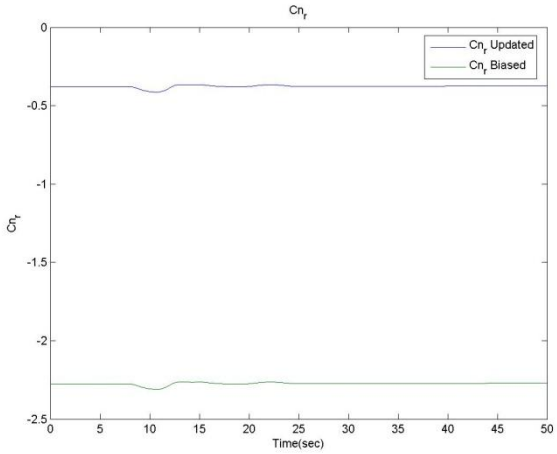

Figure 5-59: Updated vs. biased (500%) Cn_r- Sixth scenario

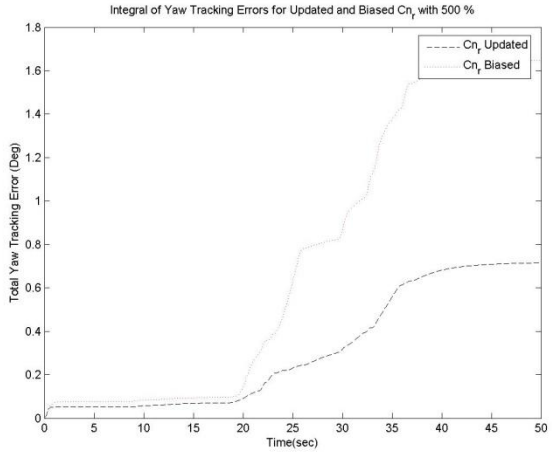

Figure 5-60: Integral of yaw tracking error for updated vs. biased (500%) Cn_r- Sixth scenario

Table 5-31: Updated vs. biased Cn_r – Average results

Bias (%)	Integral of Pitch Tracking Error(deg)	Integral of Roll Tracking Error(deg)	Integral of Yaw Tracking Error(deg)	Integral of Pitch Neural Network	Integral of Roll Neural Network	Integral of Yaw Neural Network
0	100.00	100.00	100.00	100.00	100.00	100.00
5	100.00	100.00	100.21	100.01	100.01	100.02
10	100.00	100.00	100.44	100.01	100.01	100.04
15	100.00	100.00	100.70	100.02	100.02	100.05
20	100.00	100.00	100.97	100.03	100.03	100.07
25	100.00	100.00	101.27	100.03	100.03	100.09
30	99.99	100.00	101.60	100.04	100.04	100.11
50	99.99	100.01	103.22	100.07	100.07	100.18
80	99.99	100.01	106.88	100.10	100.10	100.29
100	99.99	100.01	110.24	100.13	100.13	100.36
120	99.99	100.02	114.29	100.16	100.15	100.43
150	99.99	100.02	120.83	100.19	100.19	100.54
200	100.00	100.03	132.32	100.25	100.26	100.73
250	100.01	100.04	144.20	100.32	100.32	100.92
300	100.02	100.05	156.45	100.38	100.38	101.11
350	100.03	100.07	169.17	100.44	100.44	101.30
400	100.05	100.09	182.44	100.50	100.51	101.49
450	100.07	100.13	196.43	100.56	100.57	101.68
500	100.08	100.18	211.28	100.62	100.63	101.88
-5	100.00	100.00	99.81	100.00	99.99	99.98
-10	100.00	100.00	99.65	99.99	99.99	99.96
-15	100.00	100.00	99.50	99.98	99.98	99.95
-20	100.00	100.00	99.38	99.97	99.97	99.93
-25	100.00	100.00	99.28	99.97	99.97	99.91
-30	100.00	100.00	99.21	99.97	99.96	99.89
-50	100.01	99.99	99.23	99.94	99.94	99.82
-80	100.01	99.99	99.85	99.90	99.90	99.72
-100	100.02	99.99	100.57	99.87	99.87	99.65
-120	100.03	99.99	101.49	99.85	99.84	99.58
-150	100.04	99.98	103.28	99.80	99.81	99.47
-200	100.05	99.98	107.87	99.74	99.74	99.30
-250	100.06	99.97	113.82	99.68	99.67	99.13
-300	100.08	99.97	120.93	99.61	99.61	98.96
-350	100.10	99.96	129.09	99.55	99.54	98.79
-400	100.09	99.96	137.84	99.53	99.48	98.63
-450	100.11	99.96	146.81	99.47	99.41	98.46
-500	100.13	99.96	155.91	99.41	99.34	98.30

9- $C_{n_{\delta r}}$

I) Updated Derivative Against Fixed Derivative

[Figure 5-45] shows both updated and fixed $C_{n_{\delta r}}$.

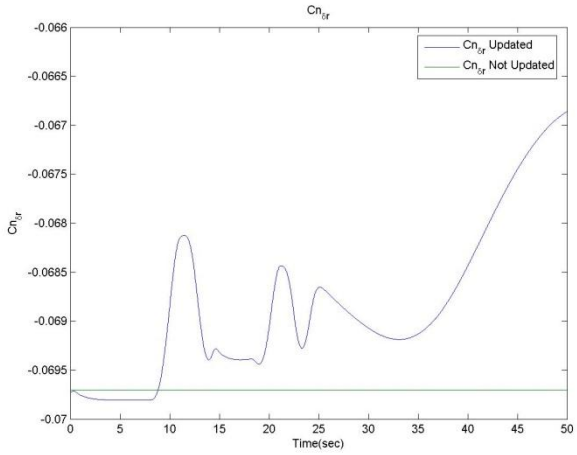

Figure 5-61: Updated vs. fixed $C_{n_{\delta r}}$- Third scenario

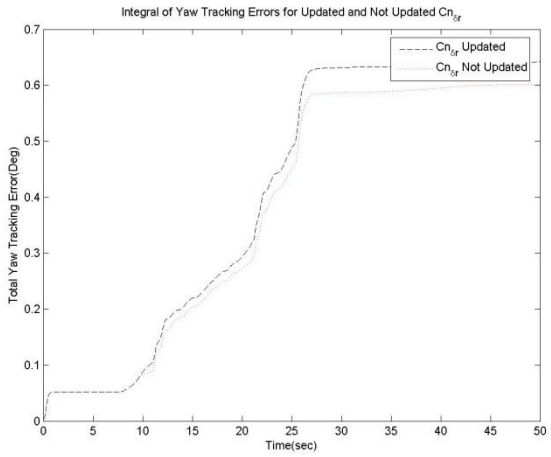

Figure 5-62: Integral of yaw tracking error for updated vs. fixed $C_{n_{\delta r}}$- Third scenario

The average results show no difference between updating and fixing $C_{n_{\delta r}}$. Detailed analysis shows that the quality of the yaw tracking in the fifth scenario improved by about a 10%, and the yaw-NN effort decreased by a 2% after updating this derivative, [Table 5-32]. Contrary to third and sixth scenarios, where the yaw tracking performance degraded after updating the derivative by a 6% and 4.5% respectively, [Table 5-33, Figure 5-61, and Figure 5-62]. All other parameters remained unchanged.

Table 5-32: Updated vs. fixed $C_{n_{\delta r}}$ – Fifth scenario

	Updated $C_{\ell_{\delta a}}$	Fixed $C_{\ell_{\delta a}}$
Integral of Pitch Tracking Error (%)	100	100.04
Integral of Roll Tracking Error (%)	100	99.999
Integral of Yaw Tracking Error (%)	100	110.06
Integral of Pitch Neural Network (%)	100	100.02
Integral of Roll Neural Network (%)	100	100.03
Integral of Yaw Neural Network (%)	100	102.06

Table 5-33: Updated vs. fixed $C_{n_{\delta r}}$ – Third scenario

	Updated $C_{\ell_{\delta a}}$	Fixed $C_{\ell_{\delta a}}$
Integral of Pitch Tracking Error (%)	100	99.795
Integral of Roll Tracking Error (%)	100	99.988
Integral of Yaw Tracking Error (%)	100	93.767
Integral of Pitch Neural Network (%)	100	100.03
Integral of Roll Neural Network (%)	100	99.991
Integral of Yaw Neural Network (%)	100	99.812

II) Updated Derivative Against Delayed Derivative

The average results show that the delay does not have any effect. Detailed analysis shows that the fifth scenario's yaw quality degrades by about a 10% at a 10 seconds delay. For the fifth scenario, therefore, updating this derivative will only be effective if the delay is less than a 10 seconds. Same scenario, the greatest degradation of 12% occurs at 17 seconds delay, [Figure 5-63, Figure 5-64, and Table 5-34].

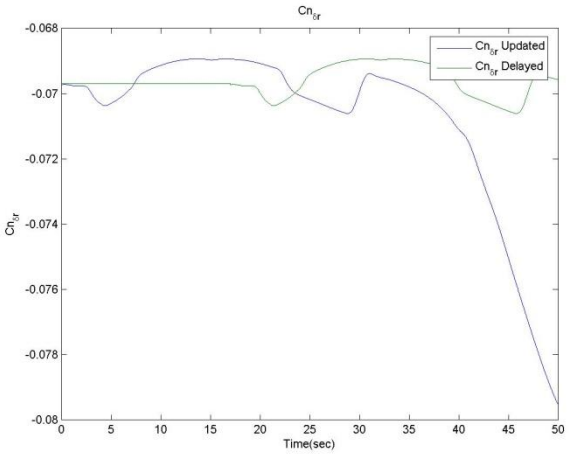

Figure 5-63: Updated vs. delayed (17-seconds) $C_{n_{\delta r}}$- Fifth scenario

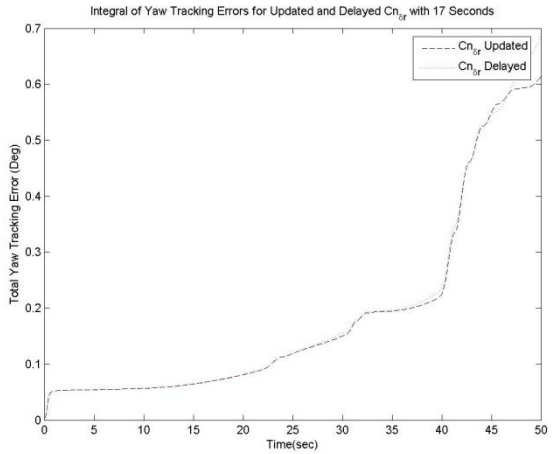

Figure 5-64: Integral of yaw tracking error for updated vs. delayed (17-seconds) $C_{n_{\delta r}}$- Fifth scenario

Table 5-34: Updated vs. delayed $C_{n_{\delta r}}$ – Fifth scenario

Delay (sec)	0	0.5	1	2	3	4	5	7	9	10	12	15	17	20	25	30	35	40
Integral of Pitch Tracking Error(deg)	100	100	100.1	100.2	100.2	100.2	100.2	100.2	100.1	100.1	100.1	99.94	99.97	100.1	100.1	100.1	100	100.1
Integral of Roll Tracking Error(deg)	100	100	99.99	99.97	100	100	99.99	99.99	99.99	100	99.94	99.95	100	100	100	100	99.99	99.97
Integral of Yaw Tracking Error(deg)	100	98.52	96.98	99.23	100.7	100	102.5	105.3	108.6	109.9	107.1	109.4	111.5	108.4	107.7	111.8	110.2	112
Integral of Pitch Neural Network	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Integral of Roll Neural Network	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Integral of Yaw Neural Network	100	100.2	100.3	100.6	100.9	101.1	101.3	101.7	101.9	101.9	102	102	102	102.1	102.3	102.3	102.2	102

III) Updated Derivative Against Biased Derivative

Increasing the bias in either positive or negative direction increases the yaw tracking error. The controller is more sensitive to negative biases such that a -100% bias will destabilize the system. On the contrary, the positive bias increase has an unexpected favorable effect on the pitch and roll tracking performance; nevertheless, the negative bias has a slightly bad effect on them, [Table 5-35].

Table 5-35: Updated vs. biased $C_{n_{\delta r}}$ – Average results

Bias (%)	Integral of Pitch Tracking Error(deg)	Integral of Roll Tracking Error(deg)	Integral of Yaw Tracking Error(deg)	Integral of Pitch Neural Network	Integral of Roll Neural Network	Integral of Yaw Neural Network
0	100.00	100.00	100.00	100.00	100.00	100.00
5	99.34	99.50	88.39	100.08	99.93	99.29
10	98.71	99.03	83.25	100.16	99.86	98.60
15	98.11	98.58	82.53	100.23	99.79	97.93
20	97.51	98.14	87.06	100.31	99.72	97.28
25	96.94	97.73	100.05	100.38	99.65	96.65
30	96.38	97.34	114.64	100.46	99.58	96.04
50	94.34	95.94	176.18	100.73	99.30	93.74
80	91.75	94.45	262.28	101.10	98.87	90.67
100	90.30	93.99	314.07	101.32	98.59	88.98
120	89.02	94.04	361.40	101.53	98.31	86.92
150	87.39	94.46	424.72	101.81	97.89	83.74
200	85.25	95.25	513.63	102.21	97.22	79.69
250	83.64	96.01	586.60	102.54	96.58	77.32
300	82.41	96.66	647.82	102.82	95.95	76.02
350	81.46	97.21	699.88	103.06	95.36	74.65
400	80.69	97.68	745.20	103.27	94.79	74.10
450	80.08	98.11	784.55	103.46	94.25	73.26
500	79.57	98.49	819.08	103.63	93.73	72.36
-5	100.67	100.52	116.31	99.92	100.07	100.73
-10	101.37	101.07	135.63	99.84	100.13	101.48
-15	102.08	101.64	156.84	99.75	100.20	102.25
-20	102.80	102.25	179.35	99.68	100.27	103.04
-25	103.55	102.88	202.86	99.59	100.33	103.86
-30	104.31	103.54	227.41	99.50	100.39	104.70
-50	107.49	106.53	336.49	99.19	100.63	108.83
-80	112.79	119.29	368.07	98.21	101.01	118.07
-100	unstable	unstable	unstable	unstable	unstable	unstable
-120	unstable	unstable	unstable	unstable	unstable	unstable
-150	unstable	unstable	unstable	unstable	unstable	unstable
-200	unstable	unstable	unstable	unstable	unstable	unstable
-250	unstable	unstable	unstable	unstable	unstable	unstable
-300	unstable	unstable	unstable	unstable	unstable	unstable
-350	unstable	unstable	unstable	unstable	unstable	unstable
-400	unstable	unstable	unstable	unstable	unstable	unstable
-450	unstable	unstable	unstable	unstable	unstable	unstable
-500	unstable	unstable	unstable	unstable	unstable	unstable

2. Failure Condition

The last six flight scenarios are repeated considering a stabilator failure in order to study the importance of parameter estimation and the effect of the estimation issues on the performance of the hybrid controller at post failure conditions. Although a thorough systematic analysis has been carried out to include the entire set of nine derivatives, in this section only those particular derivatives will be discussed that showed a different trend at post-failure conditions than at nominal conditions. The derivatives discussed in this section are presented in decreasing order of the impact that the failure had on them.

$1 - Cm_{\delta s}$

I) Updated Derivative Against Fixed Derivative

This derivative was the most affected by the stabilator failure in terms of the controller performance and neural networks effort when updating the derivative as compared to fixing it. As presented previously, at nominal condition, updating this derivative had no favorable impact on the controller; however, at failure condition, the average results show that the pitch tracking improved by 15% after updating the derivative. Detailed analysis shows that every single scenario experiences an enhancement in tracking pitch rate after updating $Cm_{\delta s}$. Not only pitch rate, but also tracking roll and yaw rates improved by the update, which did not happen at nominal condition.

The three neural networks' effort, to different extents, also decreases when the derivative is updated. For instance, third flight scenario shows that pitch-NN, roll-NN, and yaw-NN effort increases when fixing the dertivative by 19% [Figure 5-69], 1.5%, 5% [Figure 5-70] respectively. The same flight scenario shows when the derivative is fixed and regardless of the increase in the neural networks effort, the controller tracking performance worsen; The pitch, roll, and yaw tracking errors increase by 25% [Figure 5-66], 11% [Figure 5-67], and 11% [Figure 5-68] respectively, [Table 5-36].

Table 5-36: Updated vs. fixed $Cm_{\delta s}$ – Third scenario – Stabilator failure

	Updated $Cm_{\delta s}$	Fixed $Cm_{\delta s}$
Integral of Pitch Tracking Error (%)	100.00	124.73
Integral of Roll Tracking Error (%)	100.00	110.92
Integral of Yaw Tracking Error (%)	100.00	110.90
Integral of Pitch Neural Network (%)	100.00	118.87
Integral of Roll Neural Network (%)	100.00	101.49
Integral of Yaw Neural Network (%)	100.00	104.05

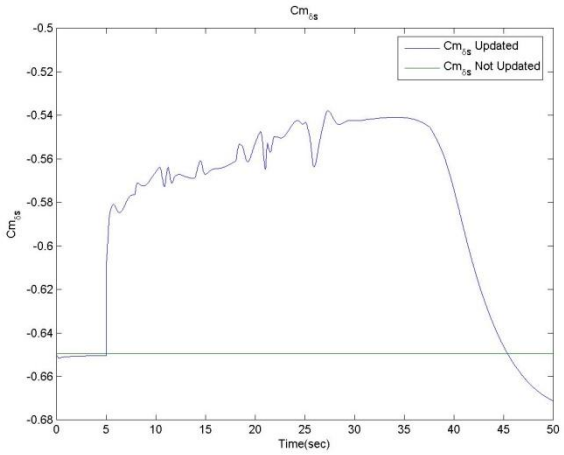

Figure 5-65: Updated vs. fixed $Cm_{\delta s}$- Third scenario – Stabilator failure

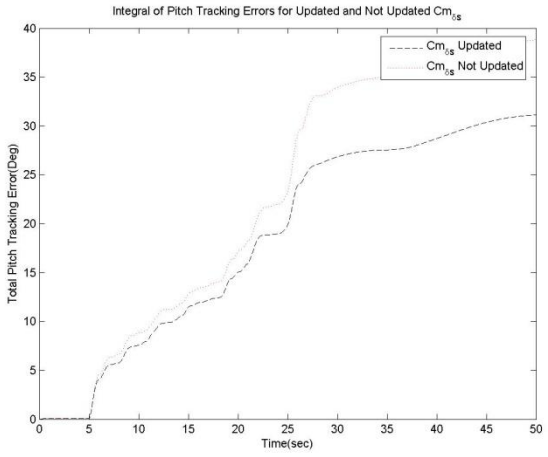

Figure 5-66: Integral of pitch tracking error for updated vs. fixed $Cm_{\delta s}$- Third scenario – Stabilator failure

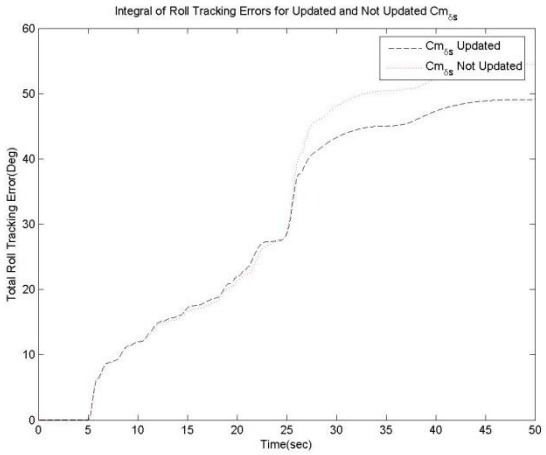

Figure 5-67: Integral of roll tracking error for updated vs. fixed $Cm_{\delta s}$- Third scenario – Stabilator failure

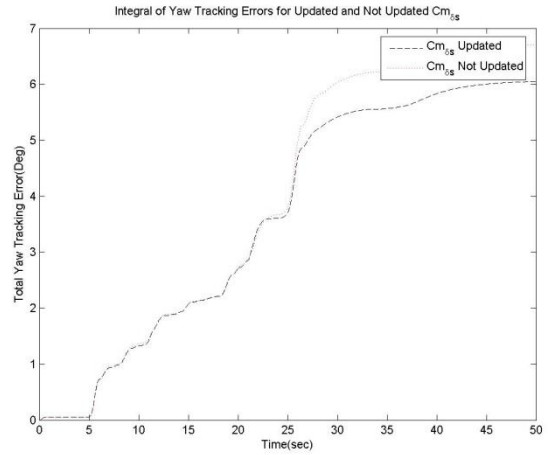

Figure 5-68: Integral of yaw tracking error for updated vs. fixed $Cm_{\delta s}$- Third scenario – Stabilator failure

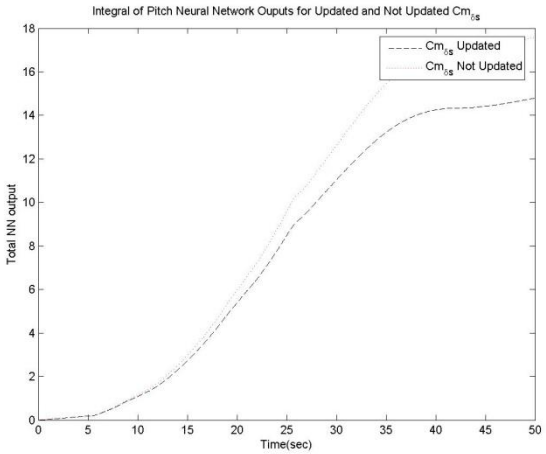

Figure 5-69: Integral of pitch neural network output for updated vs. fixed $Cm_{\delta s}$- Third scenario – Stabilator failure

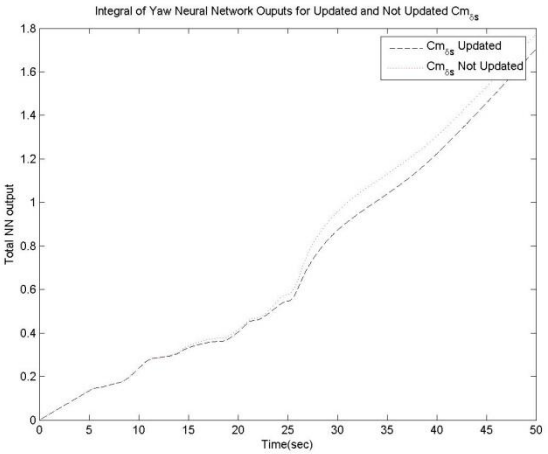

Figure 5-70: Integral of yaw neural network output for updated vs. fixed $Cm_{\delta s}$- Third scenario – Stabilator failure

II) Updated Derivative Against Delayed Derivative

The delay in updating this derivative affects the quality of tracking pitch, roll, and yaw rates. The pitch tracking was the most affected, since the average results show that fives seconds delay increases the error by 12%. The resulted error in tracking roll and yaw rates, however, was smaller; in the range of 5% due to five seconds delay. Detailed analysis shows that the fifth scenario is very sensitive to delay, such that a one second delay results in a 20% increase in the pitch tracking error. With respect to neural networks, the delay in this derivative only increases the pitch-NN effort, but the delay has no impact on the roll or yaw neural networks. The average results show that if the estimation process causes more than five seconds delay, updating $Cm_{\delta s}$ will be inefficient.

III) Updated Derivative Against Biased Derivative

The bias impact on this derivative at failure is similar to nominal condition as the entire controller performance degrades as the bias increases. The controller is more sensitive towards negative more than positive bias so that the system becomes unstable when the bias reaches -100%.

$2 - Cm_{\alpha}$

The average results show that updating this derivative enhances the pitch tracking quality by about 2.5% as compared to not updating it in addition to a yaw-tracking enhancement by 1.5% and no change in the roll tracking quality. Detailed analysis shows that the controller performance remained unchanged after updating Cm_{α} in all the flight scenarios except the second scenario. In the second scenario, the pitch, roll, and yaw tracking quality improved after updating the derivative by 10.5%, 3%, and 7% respectively, [Table 5-37]. With respect to delay and bias, the delay impact was similar to the aforementioned effect at nominal condition but with greater error growth. Additionally, the bias increase results in unstable behavior faster than at nominal flight.

Table 5-37: Updated vs. fixed $C_{m_{\alpha}}$ – Second scenario – Stabilator failure

	Updated $C_{\ell_{\delta a}}$	Fixed $C_{\ell_{\delta a}}$
Integral of Pitch Tracking Error (%)	100	110.45
Integral of Roll Tracking Error (%)	100	103.13
Integral of Yaw Tracking Error (%)	100	107.16
Integral of Pitch Neural Network (%)	100	96.945
Integral of Roll Neural Network (%)	100	99.671
Integral of Yaw Neural Network (%)	100	106.6

$3 - Cn_{\beta}$

The average results show that updating Cn_{β} increases the quality of tracking yaw rate by 7% as compared to not updating this derivative. On the contrary, the average results does not show any change in the other parameters. The detailed analysis shows that the yaw tracking quality improved in all the scenarios to different extents. The fifth scenario presents the maximum improvement in yaw tracking after updating Cn_{β} by 37%, [Table 5-38, Figure 5-71, and Figure 5-72].

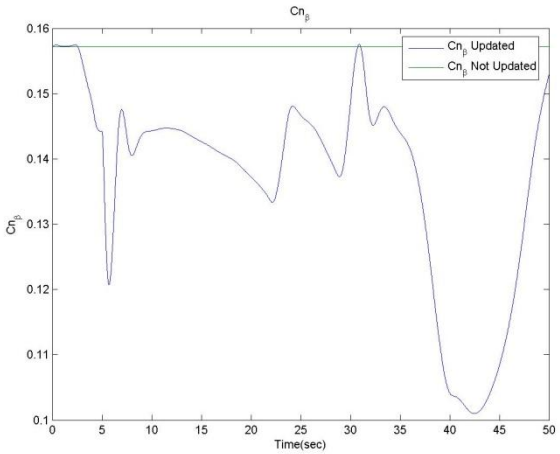

Figure 5-71: Updated vs. fixed $C_{n_{\beta}}$ - Fifth scenario – Stabilator failure

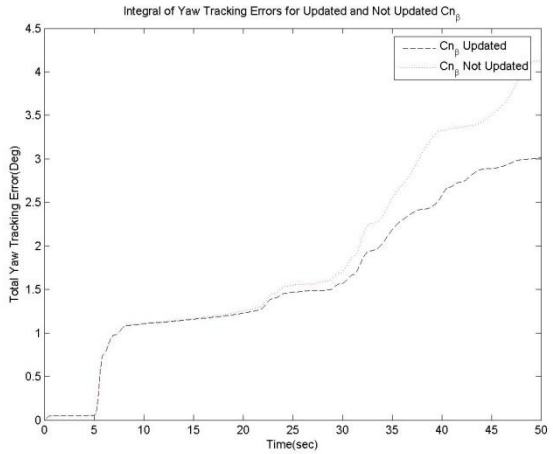

Figure 5-72: Integral of yaw tracking error for updated vs. fixed $C_{n_{\beta}}$ - Fifth scenario – Stabilator failure

Table 5-38: Updated vs. fixed $C_{n_{\beta}}$ – Fifth scenario – Stabilator failure

	Updated $C_{\ell_{\delta a}}$	Fixed $C_{\ell_{\delta a}}$
Integral of Pitch Tracking Error (%)	100	100.11
Integral of Roll Tracking Error (%)	100	96.411
Integral of Yaw Tracking Error (%)	100	136.87
Integral of Pitch Neural Network (%)	100	100.47
Integral of Roll Neural Network (%)	100	100.05
Integral of Yaw Neural Network (%)	100	101.94

3. Unexpected results

Preceding the start of this study, it was expected to observe a consistent improvement in the controller performance by updating the derivatives. However, that was not always true and even in some cases, it was the opposite and the fixed derivative provided a slightly better tracking. Based on the obtained results, this section proposes potential reasons for this unexpected observation. This discussion utilizes four figures included down to support the determined viewpoint. Both [Figure 5-73] and [Figure 5-74] present a steady-state level flight extends from the beginning to 20 seconds later followed by a pitch input at around the 25th second then a steady state climbing till the end of the scenario (hereafter known as Flight-1). [Figure 5-75 and Figure 5-76] show another type of flight scenarios that consists of three doublets in the three channels (hereafter known as Flight-2). This maneuver introduces the controller to a small variation in Cm_{α}, while producing a high disturbance all the maneuver period and keeps the flight away from steady state.

Updating Cm_{α} in Flight-1 improved the performance of the controller; however, updating the same derivative in Flight-2 confused the controller and resulted in a worse performance as compared to no update. The comparison between those two flights seems to suggest that estimating and updating a derivative at unsteady-state flight conditions may generate greater tracking error as compared to not updating the derivative. The reason is that the estimation process is based on linearized equations derived at steady state reference conditions, and those equations are more accurate around reference conditions.

[Figure 5-73 and Figure 5-74], at the time between the 23rd and the 40th second, the aircraft was introduced to the only input over the maneuver time, which results in a disturbed flight condition. Moreover, this was the only period when the controller performance was worse while updating Cm_{α}. Flight-2, along those lines, [Figure 5-75] shows at the 7th second the beginning of the disturbance, which coincides with the start of obtaining a worse performance from updating the derivative.

The second reason can be shown by dividing Flight-1 period into three sections; 1-22 seconds, 22-40 seconds, and 40-50 seconds. The last section shows the part of the flight that included the greatest variation in value of Cm_{α}. This part experiences the best improvement of the controller by updating the derivative. In addition, Flight-2 shows a smaller change in Cm_{α} with no favorable performance for the updated controller as compared to the controller with fixed Cm_{α}. The two cases support the second reason that the range of change in the value of the investigated derivative is very important for the performance of the estimation scheme.

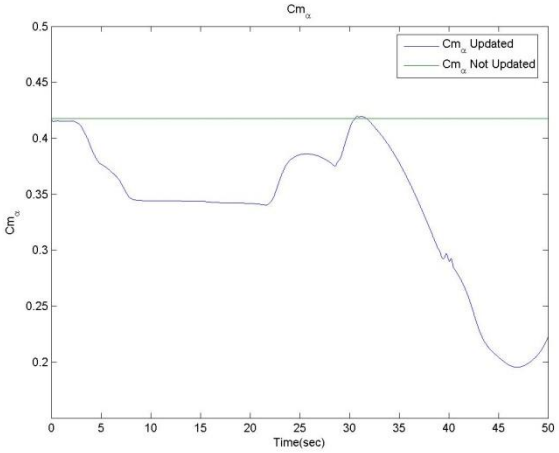

Figure 5-73: Updated vs. fixed Cm_α (Flight-1)

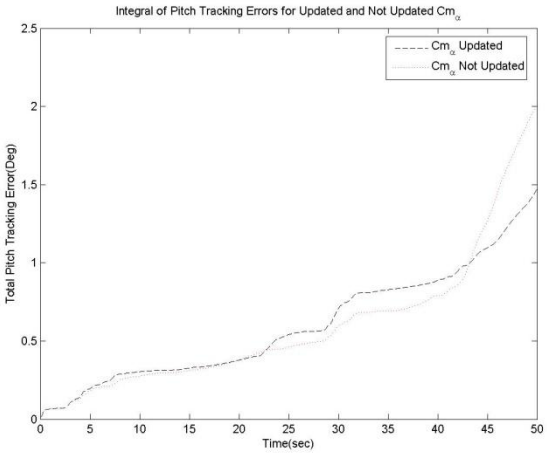

Figure 5-74: Integral of pitch tracking error for updated vs. fixed Cm_α (Flight-1)

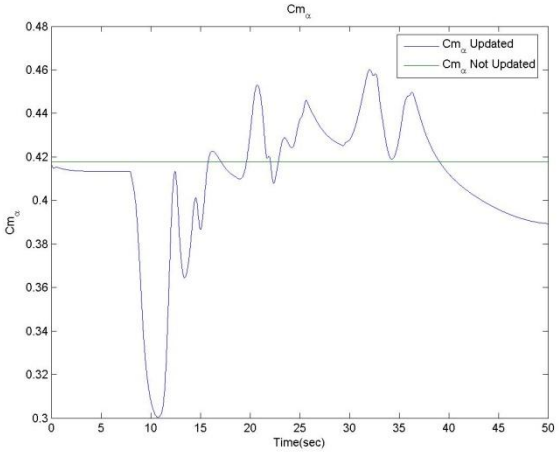

Figure 5-75: Updated vs. fixed Cm_α(Flight-2)

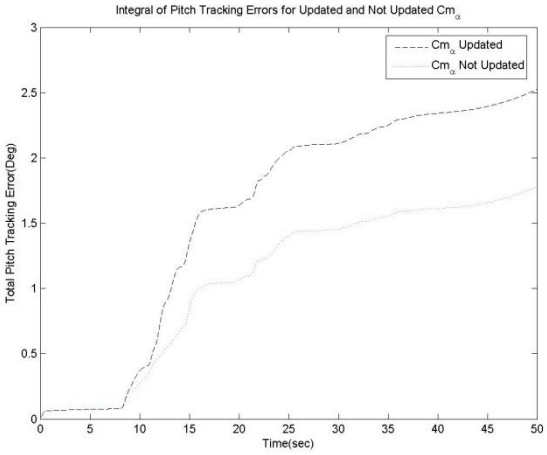

Figure 5-76: Integral of pitch neural network output for updated vs. fixed Cm_α (Flight-2)

Chapter 6. Conclusions and Recommendations

1. Conclusions

Update

The following aspects can summarize the importance of updating derivatives within a fault tolerant control system as revealed through this study:

A. Derivative variation rate: The change in the derivative value must be significant for the estimation/updating process to enhance the controller performance.

B. Steady-State estimation: the estimation/updating process produces best results if it takes place during low acceleration maneuvers. It seems that abrupt changes in the updated parameters may interfere adversely with the NN on-line learning process.

C. Failure conditions: the study of a stabilator failure condition shows that the controller performance improved by about 36% by updating $Cm_{\delta s}$. This improvement was not obtained at nominal conditions when the same derivative was updated. Therefore, updating particular derivatives is crucial at post failure conditions as compared to nominal condition.

The analysis has shown that the benefits of continuously updating the derivatives during the flight are affected by numerous factors such as the nature of the derivative, the derivative rate of change, the type and dynamics of the maneuver, and NN learning. Finally, updating some derivatives is much more important at failure than at nominal flight.

Delay

The analysis included all the potential delays from a half second up to forty seconds. Although inconsistencies have been recorded, most derivatives show worse performance after five seconds delay as compared to not updating the derivative at all. Therefore, the convergence time needs to be less than five seconds in order for the estimation scheme to be more efficient than a controller with fixed derivatives. However, a longer delay does not necessarily induce a dangerous effect on the controller performance, at least for the flight conditions considered.

It was also noted that the reason of this inconsistency is that sometimes the value of the delayed derivative can coincidently fall near the actual value of the derivative. [Figure 6-1 and Figure 6-2] show an example of a situation when the bigger delay is closer to the actual value than the smaller delay.

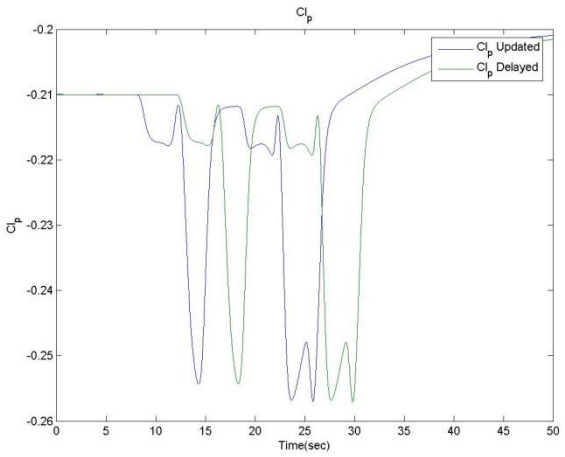

Figure 6-1: Updated vs. Delayed Cl_p (4 seconds)

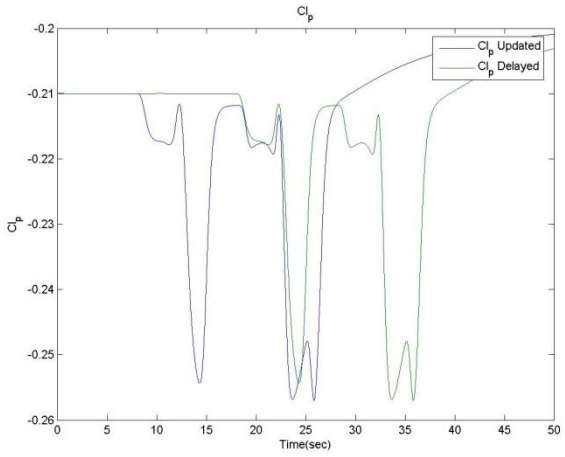

Figure 6-2: Updated vs. Delayed Cl_p (10 seconds)

Bias

The bias, on the other hand, showed a consistent and expected impact. The average results of all the derivatives show that when the bias is less than 30%, the error is acceptable. As long as the bias is less than 50%, the updated control scheme will provide more efficiency than the fixed one. Similarly, in the opposite direction, the bias must be less than -15%. The bias could have a dangerous impact particularly at higher levels; moreover, the controller is more sensitive to negative bias than to positive bias. The FTR scheme provides estimates at accuracy within a 30% from the perfect value of the derivative; hence, the FTR will provide reliable assistance to the controller.

Bias and Delay

The effect of simultaneous delays and biased values showed insignificant difference from the tracking error generated by pure bias. Although the conclusion was obtained from limited number of tests, the tests were conducted on derivatives with primary effect on their channels

2. Recommendations

The primary purpose of this study was to explore the impact of delays and biases in the estimation process on the controller performance. The study showed the positive impact of parameter updating during quasi-steady state conditions and the possibly adverse impact during fast dynamic changes. In addition, the analysis showed that controller performance appears to depend on the rate of change of the derivatives. This conclusion opens two questions for further exploration in future research.

The objective of the first question is to obtain a map for the derivatives change during the flight in response to flight condition changes and different system inputs. In other words, the question is "what is the effect of different flight inputs and flight scenarios on derivatives change rate?" Answering this question will help us know which maneuvers make bigger change in derivatives, know at which maneuvers the controller will be enhanced by updating derivatives, and know at which maneuvers the

update should be avoided. The second question is “when do fast dynamic changes occur during flight?” Answering this question will help us explore the best timing for updating derivatives, in terms of steady and unsteady times, and their effect on the controller performance.

References

1- Nguyen, Nhan T.; and Jacklin, Stephen A.; "Neural Net Adaptive Flight Control Stability, Verification and Validation Challenges, and Future Research", NASA Ames Research Center, Moffett Field, CA 94035, IJCNN Conference 2007

2- Nguyen, N.; and Krishnakumar, K.; "A Hybrid Flight Control with Adaptive Learning Parameter Estimation", AIAA Infotech@Aerospace Conference, AIAA-2007-2841,May 2007

3- Song, Y.; Campa, G.; Napolitano, M.; Seanor, B.; and and Perhinschi, M.; "Comparison of On-Line Parameter Estimation Techniques Within a Fault Tolerant Flight Control System", AIAA Journal of Guidance Control and Dymanics (2001)

4- Perhinschi, M.G.; Lando, M.; Massotti, L.; Campa, G.; Napolitano, M.R.; and Fravolini, M.L.; "On-Line Parameter Estimation Issues for the NASA IFCS F-15 Fault Tolerant Systems", American Control Conference, 2002. Proceedings of the 2002, 191 - 196 vol.1

5- Williams-Hayes, Peggy S.; "Flight Test Implementation of a Second Generation Intelligent Flight Control System", NASA/TM-2005-213669, NASA Dryden Flight Research Center, Edwards, CA

6- Krishnakumar, K.; Limes, G.; Gundy-burlet, K.; and Bryant, D.; "An Adaptive Critic Approach to Reference Model Adaptation", AIAA GN&C Conf. 2003 15 Rysdyk, Rolf

7- Burken, J.; Hanson, C.; Lee, J.; and Kaneshige, J.; "Flight Test Comparison of Different Adaptive Augmentations of Fault Tolerant Control Laws for a Modified F-15 Aircraft", NASA Dryden Flight Research Center, Edwards, CA, AIAA-2009-2056

8- Larson, R.; Burken, J.; and Butler, B.; "Implementation of an Adaptive Controller System from Concept to Flight Test", NASA Dryden Flight Research Center Edwards, California, July 2009, AIAA-2009-2055

9- Ostroff, Aaron I.; "Study of a Simulation Tool To Determine Achievable Control Dynamics and Control Power Requirements With Perfect Tracking", NASA/TM- 1998-208699

10- Hageman, Jacob; Smith, Mark; and Stachowiak, Susan; "Integration of Online Parameter Identification and Neural Network for In-Flight Adaptive Control", AIAA-2003-5700, NASA Dryden Flight Research Center, Edwards, CA

11- Joy Z. Sun and Suresh M. Joshi, "An Indirect Adaptive Control Scheme in the Presence of Actuator and Sensor Failures", AIAA Guidance, Navigation, and Control Conference , AIAA-2009-5740

12- Nguyen, N.T.; and Boskovic, J.D.; "Bounded Linear Stability Margin Analysis of Nonlinear Hybrid Adaptive Control", American Control Conference, 2008, Seattle, WA,

13- Nguyen, N.T.; Bakhtiari-Nejad, M.; and Huang, Y. ;"Hybrid Adaptive Flight Control with Bounded Linear Stability Analysis" the AIAA Guidance, Navigation, and Control Conference and Exhibit, AIAA 2007-6422, Hilton Head, South Carolina

14- Perhinschi M. G.; Napolitano, M.R.; and Campa, G.; "A Simulation Environment for Design and Testing of Aircraft Adaptive Fault-Tolerant Control Systems", Aircraft Engineering and Aerospace Technology, 2008, Vol. 80 Iss: 6, pp.620 - 632

15- Patton, Ron J.; "FAULT-TOLERANT CONTROL SYSTEMS: THE 1997 SITUATION", IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes Month: 8 Year: 1997 Volume: 3 Pages: 1033-1054

16- Monaco, Jeffrey F; Ward, David G; and Bateman, Alec J. D.; "A Retrofit Architecture for Model-Based Adaptive Flight Control", AIAA 1st Intelligent Systems Technical Conference 20 - 22 September 2004, Chicago, Illinois
17- Necsulescu, Dan; Jiang, Yi-Wu; and Kim, Bumsoo; "Neural Network Based Feedback Linearization Control of an Unmanned Aerial Vehicle", International Journal of Automation and Computing 04(1), January 2007, 71-79.
18- Reichert, R.T.; "Dynamic scheduling of modern robust control autopilot designs for missiles", IEEE Control Systems Magazine, October 1992, vol. 12, issue 5
19- Shamma, J.S.; and Athans, M.; "Gain scheduling: potential hazards and possible remedies", Control Systems Magazine, IEEE, Jun 1992, vol. 12, issue 3
20- Li, J. H.; and Lee, Pan M.; "A neural network adaptive controller design for free-pitch-angle diving behavior of an autonomous underwater vehicle", Robotics and Autonomous Systems, vol 52, (2005) 132–147
21- Byoung Soo Kim; Calise, A.J.; and Kam, M.; "Nonlinear Flight Control Using Neural Networks and Feedback Linearization", Aerospace Control Systems, 1993. Proceedings. The First IEEE Regional Conference on 25-27 May 1993, 176-181
22- Campa, G.; Fravolini, M.L.; Napolitano, M.R.; Perhinschi, M.G.; and Battipede, M.; "A Stochastically Optimal Feedforward and Feedback Technique for Flight Control Systems of High Performance Aircrafts", Proceeding of the 2004 American Control Conference Boston, Massachusetts June 30. JUIY 2, 2004
23- Hyde, R.A.; and Glover, K.; "VSTOL aircraft flight control system design using H_∞ controllers and a switching strategy", Decision and Control, 1990, Proceedings of the 29th IEEE Conference on 5-7 Dec 1990, Honolulu, HI , USA, 2975 - 2980 vol.6
24- Steinberg, M. L.; "A Comparison of Intelligent, Adaptive, and Nonlinear Flight Control Laws", NAVAL AIR WARFARE CENTER AIRCRAFT DIV PATUXENT RIVER MD, 04 JUN 1999
25- Iliff, K.W.; and Taylor, L.W.; "Determination of Stability Derivatives From Flight Data Using a Newton-Raphson Minimization Technique", NASA TN D-6579, 1972, Research Engineering, NASA Dryden Flight Research Center
26- Iliff, K.W.; and Maine, R.E.; "Practical Aspects of Using a Maximum Likelihood Estimation Method to Extract Stability and Control Derivatives From Flight Data", NASA TN D-8209, 1976, NASA Dryden Flight Research Center
27- Perhinschi, M. G.; Campa, G.; Napolitano, M. R.; Fravolini, M. L.; Lando, M.; and Massotti, L; "Performance Comparison of Fault Tolerant Control Laws Within the NASA IFCS F-15 WVU Simulator", American Control Conference 2003, June 4-6, 2003 Denver CO, USA
28- Davidson, Ron; Bosworth, John T; Jacobson, Steven R;Thomson, Michael P.; and Jorgensen, Charles C.; "Flight test of an intelligent flight-control system", Oct 2003 issue of NASA Tech Briefs, Dryden Flight Research Center, Edwards, California
29- Perhinschi, M. G.; Campa, G.; Napolitano, M. R.; Lando, M.; Massotti, L.; and Fravolini, M. L.; "A Simulation Tool for On-line Real-Time Parameter Identification", AIAA Guidance Navigation and Control Conference 2002, Aug 8-10, 2002 Monterey, CA, USA
30- Napolitano, M.R.; Naylor, S.; Neppach, C.; and Casdorph, V.; "On-Line Learning Non-Linear Direct Neurocontrollers for Restructurable Control Systems", Journal of Guidance, Control and Dynamics, Vol. 18, No. 1, pp. 170-176, January-February 1995

31- Melin, Patricia; and Castillo, Oscar; "Intelligent control of aircraft dynamic systems with a new hybrid neuro–fuzzy–fractal approach", Department of Computer Science, Tijuana Institute of Technology, CA 91909, USA, Volume 142, Issues 1-4, May 2002, Pages 161-175

32- Boskovic, J.D.; Jackson, J.A.; Mehra, R.K.; and Nguyen, N.T.; "Adaptive Fault Tolerant Control Design for a Model of DC-X Dynamics", American Control Conference, 2008, 11-13 June 2008, 1046 – 1051, Seattle, WA

33- Ward, David G.; and Monaco, Jeffrey F.; "System Identification for Retrofit Reconfigurable Control of an F/A-18 Aircraft", AIAA, Journal of Aircraft, Vol. 42 No. 1 PP. 63-72 Year: 2005

34- Page, Anthony B.; Meloney, E. Dean; and Monaco, Jeffrey F.; "Flight Testing of a Retrofit Reconfigurable Control Law Architecture Using an F/A-18C", AIAA Guidance, Navigation, and Control Conference and Exhibit, 21 - 24 August 2006, Keystone, Colorado

35- "NASA Dryden Flight Research Center Fact Sheets: Intelligent Flight Control System", NASA Dryden Flight Research Center. July 21, 2006, <http://www.nasa.gov/centers/dryden/news/FactSheets/FS-076-DFRC.html>

36- Campa, G.; Fravolini, M. L.; and Napolitano, M. R.; "A Library of Adaptive Neural Networks for Control Purposes", IEEE International Symposium on Computer Aided Control System Design, September 18-20, 2002 Glasgow, Scotland, UK.

37- Campa, G.; Fravolini, M. L.; Napolitano, M. R.; and Seanor, B.; "Neural Networks-Based Sensor Validation for the Flight Control System of a B777 Research Model", American Control Conference 2002, May 8-10, 2002 Anchorage, AK, USA

38- McDowell, D.M.; Irwin, G.W.; Lightbody, G.; and McConnell, G.; "Hybrid Neural Adaptive Control for Bank-to-Turn Missiles", IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, VOL. 5, NO. 3, MAY 1997

39- DiGirolamo, Robert D. ; and Donley, Shawn T.; "Flight control law synthesis using neural network theory", in Proc. AIAA Guidance, Navigation, Contr. Conf., 1992, pp. 385–394

40- Calise, Anthony J; Kim, Byoung S; Kam, Moshe; and Azam, Misbahul; "Neural networks for feedback linearization in aircraft control", in Proc. AIAA Guidance, Navigation and Control Conference, Hilton Head Island, SC, Technical Papers. Pt. 1; 10-12 Aug. 1992. pp. 395-406. 1992

41- McFarland, M. B.; and Calise A. J.; "Neural networks for stable adaptive control of air-to-air missiles", in Proc. AIAA Guidance, Navigation, Contr. Conf., vol. 2, 1995, pp. 1280–1285

42- Hunt, K. J.; Sbarbaro, D.; bikowski, R. ; and Gawthrop, P. J.; "Neural networks for control systems—A survey", Automatica, vol. 28, no. 6, pp. 1083–1112, November 1992.

43- Joshi, Suresh M.; Tao, Gang; and Khong, Thuan; "A Direct Adaptive Control Approach in the Presence of Model Mismatch", AIAA Guidance, Navigation, and Control Conference, Chicago, Illinois, AIAA-2009-5620

44- Perhinschi, M. G.; "Introduction to Artificial Intelligence Techniques", Department of Mechanical and Aerospace Engineering Course Notes, West Virginia University, February 2011

45- Klein, V.; and Morelli, E. A.; "Aircraft System Identification – Theory and Practice", AIAA Education Series, AIAA inc., Reston, Virginia, 2006, ISBN 1-56347-832-3

46- Perhinschi, M. G.; "Aircraft Parameter Identification and Estimation – A Conceptual Overview", Department of Mechanical and Aerospace Engineering West Virginia University Course Notes, October 2010

47- Perhinschi, M. G.; "Development of Schemes for Aircraft Sub-System Failure Detection, Identification, and Evaluation", Department of Mechanical and Aerospace Engineering West Virginia University Course Notes, February 2011
48- Perhinschi, M. G.; "Development of Schemes for Aircraft Sub-System Failure Detection, Identification, and Evaluation, Part B", Department of Mechanical and Aerospace Engineering West Virginia University Course Notes, February 2011

49- Ioannou, Petros A.; and Fidan, Baris; "Adaptive Control Tutorial (Advances in Design and control)", Book, Siam, (Dec 19, 2006), ISBN:0898716152
50- Morelli, E.A.; "Real-Time Parameter Estimation in the Frequency Domain", Proceedings of the 1999 AIAA Atmospheric Flight Mechanics Conference, AIAA paper 99-4043, Portland, OR
51- Perhinschi, M. G.; Napolitano, M. R.; and Campa, G.; "A Simulation Environment For Testing And Research Of Neurally Augmented Fault Tolerant Control Laws Based On Non-Linear Dynamic Inversion", AIAA Modeling and Simulation Technologies Conference 2004, Aug 16-19, 2004 Providence, RI, USA
52- Perhinschi, M. G.; Napolitano, M. R.; Campa, G.; Fravolini, M. L.; Massotti, L.; and Lando, M.; "Augmentation of a Non Linear Dynamic Inversion Scheme within the NASA IFCS F-15 WVU Simulator", American Control Conference 2003, June 4-6, 2003 Denver CO, USA
53- Tischler, Mark B.; and Remple, Robert K.; "Aircraft and rotorcraft system identification, engineering methods with flight test examples", Book, Am. Inst. of Aero. & Astro., AIAA, ISBN 1-56347-837-4
54- Stowell, Rich; "All types of turns", April 2011, <http://flighttraining.aopa.org/students/presolo/skills/turns.html>
55- Curry, Timothy J.; "Estimation of Handling Qualities Parameters of the Tu-144 Supersonic Transport Aircraft From Flight Test Data", April 2011, <http://www.tpub.com/content/nasa2000/NASA-2000-cr210290/NASA-2000-cr2102900060.htm>
56- Yasser, M.; Phuah, J.; Jianming Lu; and Yahagi, T.; "Simple Adaptive Control for SISO Nonlinear System Using Neural Networks for Magnetic Levitation Plant", The 2004 47th Midwest Symposium on Circuits and Systems, 2004 MWSCAS '04, 25-28 July 2004, pp, 113-16 vol.3
57- Perhinschi M. G., "Introduction to Automatic Accommodation of Aircraft Sub-System Abnormal Conditions", West Virginia University Course Notes, November 2010
58- Perhinschi, M. G.; Napolitano, M. R.; Campa, G.; Seanor, B.; Burken J.; and Larson R., "An Adaptive Threshold Approach for the Design of an Actuator Failure Detection and Identification Scheme", IEEE Transactions on Control Systems Technology, Vol.14, No 3, pp 519-525, May 2006
59- Perhinschi, M. G.; and Napolitano, M. R.; "Using Flight Simulation for Flight Dynamics and Control Education Enhancement", AIAA Atmospheric Flight Mechanics Conference and Exhibit 20-23 August 2007, Hilton Head, South California
60- Stowell, Rich; "All types of turns", April 2011, <http://flighttraining.aopa.org/students/presolo/skills/turns.html>

61- Curry, Timothy J.; "Estimation of Handling Qualities Parameters of the Tu-144 Supersonic Transport Aircraft From Flight Test Data", April 2011, <http://www.tpub.com/content/nasa2000/NASA-2000-cr210290/NASA-2000-cr2102900060.htm>

Appendices

A. MATLAB Codes

Various MATLAB codes were written to support the analysis and simulation in this research effort. However, this appendix presents only two MATLAB codes as an example.

1. plots_master_cma.m

This section shows an example of a code out of nine written codes for each derivative. Moreover, the presented code is only a part of a 1060 lines code. Each of the nine codes performs various tasks:

a. Save all the required variables for the analysis of each derivative after the simulation in SIMULINK.
b. Plot the relevant figures to different cases, updated, fixed, delayed, and biased derivatives.
c. Create folders on the hard disk and save all the figures in their corresponding folders.
d. Create and save matrices including all the tracking error calculations and save them in the folder to be used later in Excel for statistical analysis.

The code

```
nlast=length(t);
t_last=t(nlast);

counteri=counteri+1;
% this value is defined to zero in handle_01
% Then get updated every time this file is called

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Assigning variables
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 1) Updated Derivative
if (Cma_delay==0) & (Cma_bias==0)
counteru=1;

% Simulation Attitude
RAR=Roll_Actual_Reference.signals.values;
PAR=Pitch_Actual_Reference.signals.values;
YAR=Yaw_Actual_Reference.signals.values;

% Tracking Errors
RTE=Roll_Tracking_Error.signals.values;
PTE=Pitch_Tracking_Error.signals.values;
YTE=Yaw_Tracking_Error.signals.values;

% Integration of Tracking Errors
RTEI=Integration_Roll_Tracking_Error.signals.values;
PTEI=Integration_Pitch_Tracking_Error.signals.values;
YTEI=Integration_Yaw_Tracking_Error.signals.values;
```

```
% Neural Network Outputs
NN_roll=NN_Output.signals.values(:,1);
NN_pitch=NN_Output.signals.values(:,2);
NN_yaw=NN_Output.signals.values(:,3);

% Integration of Neural Network Outputs
NNI_roll=Integration_NN_Output.signals.values(:,1);
NNI_pitch=Integration_NN_Output.signals.values(:,2);
NNI_yaw=Integration_NN_Output.signals.values(:,3);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

% 4) Biased Derivative
else
counteru=4;

% Simulation Attitude
RAR_Bias=Roll_Actual_Reference.signals.values;
PAR_Bias=Pitch_Actual_Reference.signals.values;
YAR_Bias=Yaw_Actual_Reference.signals.values;

% Tracking Errors
RTE_Bias=Roll_Tracking_Error.signals.values;
PTE_Bias=Pitch_Tracking_Error.signals.values;
YTE_Bias=Yaw_Tracking_Error.signals.values;

% Integration of Tracking Errors
RTEI_Bias=Integration_Roll_Tracking_Error.signals.values;
PTEI_Bias=Integration_Pitch_Tracking_Error.signals.values;
YTEI_Bias=Integration_Yaw_Tracking_Error.signals.values;

% Neural Network Outputs
NN_roll_Bias=NN_Output.signals.values(:,1);
NN_pitch_Bias=NN_Output.signals.values(:,2);
NN_yaw_Bias=NN_Output.signals.values(:,3);

% Integration of Neural Network Outputs
NNI_roll_Bias=Integration_NN_Output.signals.values(:,1);
NNI_pitch_Bias=Integration_NN_Output.signals.values(:,2);
NNI_yaw_Bias=Integration_NN_Output.signals.values(:,3);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

% 5) Creating Folders
tf11 = isdir('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated\');
tf12 = isdir('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Not Updated\');
tf13 = isdir('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\');
```

```
% To create the folder only if it wasn't created before
if tf11==0
mkdir('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated\')
end

if tf12==0
mkdir('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Not Updated\')
end

if tf13==0
mkdir('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\')
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
% Plot the comparisons between Updated and Not Updated
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

%To plot only when I have the two values (Updated and Not Updated)
if counteri==2;

% Roll Tracking Errors Updated vs Not Updated
figure
plot(t,RTE,'--k',t,RTE_U,':r')
legend('Cm_\alpha Updated','Cm_\alpha Not Updated')
title('Roll Tracking Errors for Updated and Not Updated Cm_\alpha')
xlabel('Time(sec)')
ylabel('Roll Tracking Error(Deg/s)')
saveas(gcf,'C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Roll Tracking error cma.jpg')
hgsave('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Roll Tracking error cma')
close

% Integral of Roll Tracking Errors Updated vs Not Updated
figure
plot(t,RTEI,'--k',t,RTEI_U,':r')
legend('Cm_\alpha Updated','Cm_\alpha Not Updated')
title('Integral of Roll Tracking Errors for Updated and Not Updated Cm_\alpha')
xlabel('Time(sec)')
ylabel('Total Roll Tracking Error(Deg)')
saveas(gcf,'C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Integral of Roll Tracking error cma.jpg')
hgsave('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Integral of Roll Tracking error cma')
close

% Pitch Tracking Errors Updated vs Not Updated
figure
plot(t,PTE,'--k',t,PTE_U,':r')
legend('Cm_\alpha Updated','Cm_\alpha Not Updated')
title('Pitch Tracking Errors for Updated and Not Updated Cm_\alpha')
```

```
xlabel('Time(sec)')
ylabel('Pitch Tracking Error(Deg/s)')
saveas(gcf,'C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Pitch Tracking error cma.jpg')
hgsave('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Pitch Tracking error cma')
close

% Integral of Pitch Tracking Errors Updated vs Not Updated
figure
plot(t,PTEI,'--k',t,PTEI_U,':r')
legend('Cm_\alpha Updated','Cm_\alpha Not Updated')
title('Integral of Pitch Tracking Errors for Updated and Not Updated Cm_\alpha')
xlabel('Time(sec)')
ylabel('Total Pitch Tracking Error(Deg)')
saveas(gcf,'C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Integral of Pitch Tracking error cma.jpg')
hgsave('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Integral of Pitch Tracking error cma')
close

% Yaw Tracking Errors Updated vs Not Updated
figure
plot(t,YTE,'--k',t,YTE_U,':r')
legend('Cm_\alpha Updated','Cm_\alpha Not Updated')
title('Yaw Tracking Errors for Updated and Not Updated Cm_\alpha')
xlabel('Time(sec)')
ylabel('Yaw Tracking Error(Deg/s)')
saveas(gcf,'C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Yaw Tracking error cma.jpg')
hgsave('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Yaw Tracking error cma')
close

% Integral of Yaw Tracking Errors Updated vs Not Updated
figure
plot(t,YTEI,'--k',t,YTEI_U,':r')
legend('Cm_\alpha Updated','Cm_\alpha Not Updated')
title('Integral of Yaw Tracking Errors for Updated and Not Updated Cm_\alpha')
xlabel('Time(sec)')
ylabel('Total Yaw Tracking Error(Deg)')
saveas(gcf,'C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Integral of Yaw Tracking error cma.jpg')
hgsave('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Integral of Yaw Tracking error cma')
close

% The difference between tracking errors when the derivative is Updated and
% Not Updated
RTE_error=abs(RTE_U)-abs(RTE);
PTE_error=abs(PTE_U)-abs(PTE);
YTE_error=abs(YTE_U)-abs(YTE);

figure
plot(t,RTE_error,t,PTE_error,'--k',t,YTE_error,':r')
legend('Roll Tracking Error','Pitch Tracking Error','Yaw Tracking Error')
```

```
title('The Difference between the Tracking Errors for Updated and Not Updated
Cm_\alpha')
xlabel('Time(sec)')
ylabel('Degrees')
saveas(gcf,'C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\The Difference between the Tracking Errors for Updated and Not Updated
Cma.jpg')
hgsave('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\The Difference between the Tracking Errors for Updated and Not Updated
Cma')
close

% Roll Neural Network Outputs Updated vs Not Updated
figure
plot(t,NN_roll,'--k',t,NN_roll_U,':r')
legend('Cm_\alpha Updated','Cm_\alpha Not Updated')
title('Roll Neural Network Ouputs for Updated and Not Updated Cm_\alpha')
xlabel('Time(sec)')
ylabel('NN output')
saveas(gcf,'C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Roll Neural Network Ouputs cma.jpg')
hgsave('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Roll Neural Network Ouputs cma')
close

% Integral of Roll Neural Network Outputs Updated vs Not Updated
figure
plot(t,NNI_roll,'--k',t,NNI_roll_U,':r')
legend('Cm_\alpha Updated','Cm_\alpha Not Updated')
title('Integral of Roll Neural Network Ouputs for Updated and Not Updated
Cm_\alpha')
xlabel('Time(sec)')
ylabel('Total NN output')
saveas(gcf,'C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Integral of Roll Neural Network Ouputs cma.jpg')
hgsave('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Integral of Roll Neural Network Ouputs cma')
close

% Pitch Neural Network Outputs Updated vs Not Updated
figure
plot(t,NN_pitch,'--k',t,NN_pitch_U,':r')
legend('Cm_\alpha Updated','Cm_\alpha Not Updated')
title('Pitch Neural Network Ouputs for Updated and Not Updated Cm_\alpha')
xlabel('Time(sec)')
ylabel('NN output')
saveas(gcf,'C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Pitch Neural Network Ouputs cma.jpg')
hgsave('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Pitch Neural Network Ouputs cma')
close

% Integral of Pitch Neural Network Outputs Updated vs Not Updated
figure
plot(t,NNI_pitch,'--k',t,NNI_pitch_U,':r')
legend('Cm_\alpha Updated','Cm_\alpha Not Updated')
```

```
title('Integral of Pitch Neural Network Ouputs for Updated and Not Updated
Cm_\alpha')
xlabel('Time(sec)')
ylabel('Total NN output')
saveas(gcf,'C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Integral of Pitch Neural Network Ouputs cma.jpg')
hgsave('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Integral of Pitch Neural Network Ouputs cma')
close

% Yaw Neural Network Outputs Updated vs Not Updated
figure
plot(t,NN_yaw,'--k',t,NN_yaw_U,':r')
legend('Cm_\alpha Updated','Cm_\alpha Not Updated')
title('Yaw Neural Network Ouputs for Updated and Not Updated Cm_\alpha')
xlabel('Time(sec)')
ylabel('NN output')
saveas(gcf,'C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Yaw Neural Network Ouputs cma.jpg')
hgsave('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Yaw Neural Network Ouputs cma')
close

% Integral of Yaw Neural Network Outputs Updated vs Not Updated
figure
plot(t,NNI_yaw,'--k',t,NNI_yaw_U,':r')
legend('Cm_\alpha Updated','Cm_\alpha Not Updated')
title('Integral of Yaw Neural Network Ouputs for Updated and Not Updated
Cm_\alpha')
xlabel('Time(sec)')
ylabel('Total NN output')
saveas(gcf,'C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Integral of Yaw Neural Network Ouputs cma.jpg')
hgsave('C:\Users\Peter\Desktop\Runs\cma\Updated_Not Updated\Updated vs Not
Updated\Updated vs Not Updated_Integral of Yaw Neural Network Ouputs cma')
close

% Statistics Percentage Matrix for Updated vs Unupdated
%  (unupdated/updated)*100

Upd_vs_Unupd_Percent1(1,2)=(PTEI_U(end)/PTEI(end))*100;
Upd_vs_Unupd_Percent1(2,2)=((RTEI_U(end))/RTEI(end))*100;
Upd_vs_Unupd_Percent1(3,2)=(YTEI_U(end)/YTEI(end))*100;
Upd_vs_Unupd_Percent1(4,2)=(NNI_pitch_U(end)/NNI_pitch(end))*100;
Upd_vs_Unupd_Percent1(5,2)=(NNI_roll_U(end)/NNI_roll(end))*100;
Upd_vs_Unupd_Percent1(6,2)=(NNI_yaw_U(end)/NNI_yaw(end))*100;
Upd_vs_Unupd_Percent1(:,1)=100;

% Statistics Matrix for Updated vs Unupdated
Upd_vs_Unupd(1,1)=PTEI(end);
Upd_vs_Unupd(2,1)=RTEI(end);
Upd_vs_Unupd(3,1)=YTEI(end);
Upd_vs_Unupd(4,1)=NNI_pitch(end);
Upd_vs_Unupd(5,1)=NNI_roll(end);
Upd_vs_Unupd(6,1)=NNI_yaw(end);
Upd_vs_Unupd(1,2)=PTEI_U(end);
```

```
Upd_vs_Unupd(2,2)=RTEI_U(end);
Upd_vs_Unupd(3,2)=YTEI_U(end);
Upd_vs_Unupd(4,2)=NNI_pitch_U(end);
Upd_vs_Unupd(5,2)=NNI_roll_U(end);
Upd_vs_Unupd(6,2)=NNI_yaw_U(end);

% Initiate Statistics Percentage Matrix for Updated vs. Delayed vs. Biased

% Pitch Tracking Error: (delayed/updated)*100
PTE_Percent1(1,1)=100;
% Roll Tracking Error
RTE_Percent1(1,1)=100;
% Yaw Tracking Error
YTE_Percent1(1,1)=100;
% Pitch Neural Network Effort
PNN_Percent1(1,1)=100;
% Roll Neural Network Effort
RNN_Percent1(1,1)=100;
% Yaw Neural Network Effort
YNN_Percent1(1,1)=100;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Delay
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Statistics Percentage Matrix for Updated vs Delayed vs Biased

% Pitch Tracking Error: (delayed/updated)*100
PTE_Percent1(1,end+1)=(PTEI_delay(end)/PTEI(end))*100;
% Roll Tracking Error
RTE_Percent1(1,end+1)=(RTEI_delay(end)/RTEI(end))*100;
% Yaw Tracking Error
YTE_Percent1(1,end+1)=(YTEI_delay(end)/YTEI(end))*100;
% Pitch Neural Network Effort
PNN_Percent1(1,end+1)=(NNI_pitch_delay(end)/NNI_pitch(end))*100;
% Roll Neural Network Effort
RNN_Percent1(1,end+1)=(NNI_roll_delay(end)/NNI_roll(end))*100;
% Yaw Neural Network Effort
YNN_Percent1(1,end+1)=(NNI_yaw_delay(end)/NNI_yaw(end))*100;

end

% Save the statistics Data every run after getting both updated and
% unupdated data
if counteri >= 2
save('C:\Users\Peter\Desktop\Runs\cma\Statistics.mat', 'Upd_vs_Unupd_Percent1',
'Upd_vs_Unupd',
'PTE_Percent1','RTE_Percent1','YTE_Percent1','PNN_Percent1','RNN_Percent1','YNN_Per
cent1')
end
```

2. Call_plots_master.m

The analysis in this research was very complex because it considers various simulations for nine derivatives each at two cases updated and not updated, as well as simulations at 17 delay and 36 bias levels. The MATLAB code presented in this section was written in order to automate the analysis process as much as possible. Since this code is very long, only a short part is presented out of 3160 lines code.

The code perfoms the following tasks:

a. Sequentially change the delay and bias variables corresponding to the explored case and derivative, and update them to the SIMULINK environment for simulation.
b. Run the SIMULINK model for each derivative.

The code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% cma
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%updated
set(cb3_handle,'Value',1);
set(edit3_handle,'String','0'); %Delay
set(edit23_handle,'String','0'); %Bias
del_03; % Imitate pressing 'OK'
sim('f15gen2_2_4_copy_w_est')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%unupdated
set(cb3_handle,'Value',1);
set(edit3_handle,'String','50'); %Delay
set(edit23_handle,'String','0'); %Bias
del_03; % Imitate pressing 'OK'
sim('f15gen2_2_4_copy_w_est')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Delay
set(cb3_handle,'Value',1);
set(edit3_handle,'String','.5'); %Delay
set(edit23_handle,'String','0'); %Bias
del_03; % Imitate pressing 'OK'
sim('f15gen2_2_4_copy_w_est')

set(cb3_handle,'Value',1);
set(edit3_handle,'String','1'); %Delay
set(edit23_handle,'String','0'); %Bias
del_03; % Imitate pressing 'OK'
sim('f15gen2_2_4_copy_w_est')
```

```
set(cb3_handle,'Value',1);
set(edit3_handle,'String','2'); %Delay
set(edit23_handle,'String','0'); %Bias
del_03; % Imitate pressing 'OK'
sim('f15gen2_2_4_copy_w_est')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Bias
set(cb3_handle,'Value',1);
set(edit3_handle,'String','0'); %Delay
set(edit23_handle,'String','5'); %Bias
del_03; % Imitate pressing 'OK'
sim('f15gen2_2_4_copy_w_est')

set(cb3_handle,'Value',1);
set(edit3_handle,'String','0'); %Delay
set(edit23_handle,'String','10'); %Bias
del_03; % Imitate pressing 'OK'
sim('f15gen2_2_4_copy_w_est')

set(cb3_handle,'Value',1);
set(edit3_handle,'String','0'); %Delay
set(edit23_handle,'String','15'); %Bias
del_03; % Imitate pressing 'OK'
sim('f15gen2_2_4_copy_w_est')

% Negative Bias
set(cb3_handle,'Value',1);
set(edit3_handle,'String','0'); %Delay
set(edit23_handle,'String','-5'); %Bias
del_03; % Imitate pressing 'OK'
sim('f15gen2_2_4_copy_w_est')

set(cb3_handle,'Value',1);
set(edit3_handle,'String','0'); %Delay
set(edit23_handle,'String','-10'); %Bias
del_03; % Imitate pressing 'OK'
sim('f15gen2_2_4_copy_w_est')

set(cb3_handle,'Value',1);
set(edit3_handle,'String','0'); %Delay
set(edit23_handle,'String','-15'); %Bias
del_03; % Imitate pressing 'OK'
sim('f15gen2_2_4_copy_w_est')
```

John H. Hagen
Digitally signed by John H. Hagen
DN: cn=John H. Hagen, o=West Virginia University Libraries, ou=Acquisitions Department, email=John.Hagen@mail.wvu.edu, c=US
Date: 2011.05.04 18:00:22 -04'00'