ONLINE AERODYNAMIC PARAMETER ESTIMATION FOR A FAULT TOLERANT FLIGHT CONTROL SYSTEM

A Thesis by

Balbahadur Singh

B.E., Nagpur University, 2002

Submitted to the College of Engineering
and the faculty of the Graduate School of
Wichita State University
in partial fulfillment of
the requirements for the degree of
Masters of Science

December 2005

ONLINE AERODYNAMIC PARAMETER ESTIMATION FOR A FAULT TOLERANT FLIGHT CONTROL SYSTEM

I have examined the final copy of this Thesis for form and content and recommend that it be accepted in partial fulfillment of the requirement for the degree of Masters of Science with a major in Aerospace Engineering.

_____
Dr. James E. Steck, Committee Chair

We have read this Thesis
and recommend its acceptance:

_____
Dr. Kamran Rokhsaz, Committee Member

_____
Dr. Kameshwara Rao Namuduri, Committee Member

ACKNOWLEDGMENTS

I would like to give special thanks to my advisor Dr. James E. Steck for his years of thoughtful, patient guidance and support. I also thank my parents and friends for their continuous encouragement.

ABSTRACT

Wichita State University (WSU) and Raytheon Aircraft Company are working toward the development of a flight control system to reduce the workload for a pilot under normal as well as deteriorated flight conditions. An ‘easy fly system’ for a Bonanza Raytheon NASA test-bed has been used by WSU to develop a neural network-based adaptive flight control system. In this thesis an online technique for aerodynamic parameter estimation is presented, which is developed to improve the adaptation. The neural-based adaptive flight controller uses an artificial neural network for immediate adaptation in dynamic inverse control to compensate for modeling error or control failure. Long-term adaptation to modeling error requires a permanent correction of the aerodynamic parameters used in the inverse controller. This method is designed to update parameters inside the controller and to provide slow and long-term adaptation to compliment the existing immediate adaptation provided by neural networks. The method employs gradient descent optimization, guided by the modeling error for updating each parameter. It also uses the linearized equations of motion where the aerodynamic forces are represented by their coefficients and derivatives. Some convergence enhancement techniques are also used to reduce the time required for parameter identification. The aircraft is flown for several specific maneuver patterns to decouple the effect of each parameter on the modeling error. This helps the algorithm identify the correct parameter changes and eliminate modeling error for all flight conditions. In-flight adjustment of parameters also replaces the time required for off-line data analysis used by other techniques. Other online techniques like the discrete time Fourier transform, a frequency domain-based technique, and locally weighted regression, a time domain based method use a computationally complex solver with matrix inversions and multiplications that require more processing power and time. The method described in this report

uses computationally simple and direct equations to update the parameters at each time step, which then become part of the controller. One of the challenges arises from the dependency of the magnitude of the modeling error on each parameter. This creates a limitation on the number of parameters that can be estimated at the same time and also increases the number of time steps for convergence. Introducing some level of prior knowledge into the system can solve the problem. These techniques were developed and tested in a Matlab/Simulink environment. The results demonstrate the ability of the techniques to find a good estimate of the parameters in a reasonable amount of time. Some limitations apply while using each technique, which open doors for future work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AIAA	American Institute of Aeronautics and Astronautics
DOF	Degrees of Freedom
E-Z	Easy
HARV	High Alpha Research Vehicle
NASA	National Aeronautical and Space Administration
PD	Proportional and Derivative Control
PID	Proportional Integral and Derivative Control
WSU	Wichita State University

LIST OF SYMBOLS

$\bar{c}$	Mean aerodynamic chord
C_{D_0}	Drag coefficient of airplane for zero angle of attack
C_{D_k}	Coefficient of induced drag
$(C_D)_{\delta_e=0}$	Drag coefficient of aircraft at zero elevator deflection
C_{L_0}	Lift coefficient of airplane for zero angle of attack
$C_{L\alpha}$	Variation of aircraft lift coefficient with angle of attack
$C_{L_{\delta_e}}$	Variation of airplane lift coefficient with elevator deflection angle
$C_L\big\vert_{\delta_e=0}$	Lift coefficient of aircraft at zero elevator deflection
C_{m0}	Pitching moment of the aircraft at zero angle of attack
$C_{m\alpha}$	Variation of aircraft pitching moment coefficient with angle of attack
C_{mq}	Variation of aircraft pitching moment coefficient with pitch rate
$C_{m\delta e}$	Variation of aircraft pitching moment coefficient with elevator deflection angle
d_T	Distance of a thrust line projection onto the XZ-plane to the center of gravity
$e(t-1)$	Artificial neural network error correction at previous time step
F_{A_x}	Aerodynamic force component along X-direction
$F_{X_{\delta_e}}$	Forces due to elevator deflection in X-direction
$F_{Z_{\delta_e}}$	Forces due to elevator deflection in Z-direction
g	Acceleration due to gravity

I_{yy}	Airplane mass moment of inertia about Y-axis
K	Learning Rate
m	Mass of the aircraft
M_A	Aerodynamic pitching moment
M_{δ_e}	Pitching moment per unit elevator angle
$\overline{q}$	Dynamic pressure
$\hat{q}$	Non-dimensional angular velocity about Y-axis
Q	Airplane angular velocity about Y-direction
S	Area of the wing
T	Thrust
U	Velocity in X-direction in body coordinates
$\dot{U}$	Acceleration in X-direction in body coordinates
V	Component of airplane velocity in Y-direction
V_c	Commanded airspeed
$\dot{V}_c$	Commanded acceleration
$\dot{V}_{error}$	Modeling error in forward acceleration
$\dot{V}(t-1)$	Actual aircraft acceleration at previous time step
W	Velocity in Z-direction in body coordinates
$\dot{W}$	Acceleration in Z-direction in body coordinates
α	Angle of attack
$\dot{\alpha}$	Rate of change of angle of attack
γ_c	Commanded flight path angle

$\ddot{\gamma}_c$	Commanded flight path angle acceleration
$\ddot{\gamma}_{error}$	Modeling error in pitch acceleration
θ	Airplane pitch attitude angle
$\dot{\theta}$	Airplane pitch rate
$\ddot{\theta}$	Airplane pitch acceleration
φ_T	Thrust line inclination angle w.r.t. Y-X-plane

CHAPTER ONE

1. INTRODUCTION

1.1 Neural-Based Adaptive Flight Control System

An automatic flight control system is one of the most important components of a long-range aircraft. It assists the pilot in flying the aircraft at a specified flight condition without direct human assistance, reducing pilot workload significantly. It also helps to improve the flying qualities by adding auxiliary control effort to the pilot command. Generally termed as an autopilot, it requires an electronic or computer-aided fly-by-wire implementation.

Most conventional flight control systems that are designed using state feedback along with classical control theory often need some kind of human assistance and require a retuning at different areas in the flight envelope. These controllers are now being replaced by modern control design techniques that require less human assistance and can be used over a large area in the flight envelope of an aircraft. The modern flight control system eliminates the direct control of the pilots over the control surfaces and allows them to command the flight path angle, airspeed, heading, and altitude from the instrument panel. Usually the pilot switches on these controllers after stabilizing the aircraft to a particular flight condition. They are designed to switch off automatically under failure conditions and allow the pilot to resume control. Modern flight control systems often require an accurate aircraft model. During operation the characteristics of the aircraft change many times due to system degradation, wear and tear, or unanticipated failures. In such conditions, the model used in the flight controller no longer represents the actual aircraft, and this modeling error can cause the controller to malfunction.

To overcome the problems faced by conventional and modern flight control systems, adaptive flight controls systems have been developed, which are capable of adapting to the

changed system characteristics and control the aircraft in degraded conditions. Working toward a fault-tolerant and easy to fly system, Wichita State University (WSU) has developed a neural-based adaptive flight control system that uses an artificial neural network to provide compensation for error due to non-linearities and partial control failure. For a commanded flight condition, the control system uses an inverse controller to calculate the required control inputs for the aircraft. The aircraft response is then compared with the commanded signal and an artificial neural network is trained online to correct the commanded signal and obtain the required response.

The pilot commands the flight path angle and bank angle from the joystick. Airspeed is commanded using an additional speed lever. These commands are passed through a feedback linear controller and converted into accelerations before feeding them to the inverse controller. The inverse controller cancels the dynamics of the aircraft, and the linear controller allows the pilot to achieve desired flying qualities by adding the desired dynamics. The controller also decouples the control inputs and allows the pilot to command one flight variable without affecting the others. This makes the aircraft easy for a non-pilot to fly, and is called an easy fly (E-Z fly) system.

To make the controller fault-tolerant and adapt to the modeling errors, the neural network provides an error-correction signal, which is added to the input commands of the inverse controller. It does not provide permanent adaptation of the parameters inside the inverse controller, which are actually responsible for the modeling error. Permanent adaptation to the error introduced in the command tracking due to a degraded flight condition, requires a permanent change in the aerodynamic parameters in the inverse controller. This requires online parameter estimation and adaptation.

1.2 Online Parameter Identification for Adaptive Flight Control

Development of most of the adaptive control techniques and their implementation in automatic flight controls requires a model of the aircraft that is a close match of the actual aircraft. While developing a new aircraft, the aerodynamic parameters are estimated accurately using techniques like wind tunnel testing and computational fluid dynamics. Aerodynamic parameters are non-dimensional quantities that describe the aerodynamic and stability characteristics of the aircraft. These quantities, along with aircraft equations of motion, are used to model a real aircraft. This model is used for flight simulation, performance analysis, and control system development for an aircraft. However, as the aircraft begins operation these parameters often vary. The reason for change in these parameters may be uneven loading of the aircraft, wear and tear over the surface, damage or deterioration of any control surface, battle damage, etc. These new changed parameters can be extracted from flight data, by plotting the aerodynamic coefficients with respect to angle of attack, and then finding the equations of the curves. This process is tedious and cannot be done online (in-flight). But for the implementation of a fully automated adaptive control, online parameter adaptation that can be used by the model of the aircraft inside the control system is required.

A few more online parameter estimation techniques [2-23] that use time domain or frequency domain methods to analyze flight data online and estimate the parameters have also been proposed. The purpose of this research is not to compare the earlier work done in this field but to develop a parameter estimation technique that matches the structure of the Neural-Based Adaptive Flight Control system developed at WSU and can be used as an integral part of the controller. The method developed is computationally simple and needs less memory.

1.3 Online Parameter Estimation Using Model Reference

The technique described in this report uses a reference model to create a measure of the amount of error in the aerodynamic parameters. This reference model was generated using the linearized equations of motion of the aircraft, which uses previously known parameters to model the aerodynamic characteristics of the aircraft. Then the response of the aircraft and the estimated response of the model were compared. The difference between the two responses is directly proportional to the difference between the actual and known parameters. An error was calculated from the difference in response, which was then used with gradient descent to change each parameter to its optimal value. The method is simple and requires less computational power. Instead of a long flight data recording, it uses data points at the current time step to update parameters at the next time step. This technique will be implemented as an integral part of the Neural-Based Adaptive Flight Controller.

CHAPTER TWO

2. LITERATURE SURVEY

Many techniques exist for estimation of aerodynamic parameters of an aircraft, but online implementation limits the field to a very few.

Since the development of adaptive control algorithms, the need for a good model of the aircraft has stimulated research in the field of parameter estimation. To generate a good aircraft model, it is important to have a good estimate of the aerodynamic parameters that closely represent the characteristics of the actual aircraft. The conventional techniques of parameter estimation are robust and exhaustive but it may not be possible to implement them in flight due to limited processor power and memory on a flight computer.

Conventional methods of aerodynamic parameter estimation are mainly off-line methods and involve post-flight analysis of flight data. Several statistical methods have been used for this purpose. The most common methods use modified forms of regression techniques to generalize the response of the aircraft and extract aerodynamic and stability derivatives. The Maximum-Likelihood method is one of the most widely used approaches. Iliff [1,2] described the basic concepts and the practical aspects of this method. The Maximum-Likelihood method minimizes a cost function indicating the difference between measured and estimated responses.

Ishimoto [3] proposed a new algorithm for maximum-likelihood parameter estimation problems, including process and measurement noise. In the new formulation of equations, equality constraints were added to the minimization, which eliminated convergence problems in previous formulations. Also a sequential quadratic programming method with a Gauss-Newton approximation was used to solve the constrained minimization problem efficiently. This method

demonstrated improvement in convergence and provided reasonable estimates for the parameters.

Napolitano and his team [4] investigated the estimation of lateral directional aerodynamic parameters of the NASA F/A-18 HARV from flight data. They used the maximum-likelihood method along with a Newton-Raphson minimization technique to estimate lateral aerodynamic and control derivatives from the flight data. The mathematical model of the aircraft is based in classical static and dynamic derivative buildup. Flight data was collected over a range of angles of attack from a variety of maneuvers. The parameter estimation code 'pEst', developed at NASA Dryden, was used in this work. Except for a few control derivatives, the technique produced good results for most of the rolling and yawing moment coefficients by showing consistency with well-defined trends and limited scatter at high angles of attack.

M. R. Ananthasayanam, A. K. Sarkar and S. Vathsal [5] used the Broyden-Fletcher-Goldfarb-Shanno optimization algorithm along with the maximum-likelihood method to get better convergence than the Newton-Raphson minimization technique. This study shows that the use of the Broyden-Fletcher-Goldfarb-Shanno optimization algorithm can minimize the cost, even when the number of measurements is less and initial input values to the optimizer are far from the true values.

Kalman Filters have also been used to estimate aerodynamic parameters and have some advantages over the more traditional least-square method. The Kalman filter is an efficient recursive filter, which estimates the state of a dynamic system from a series of incomplete and noisy measurements. It exploits the dynamics of the target, which governs its time evolution, to remove the effects of the noise and get a good estimate of the location of the target at the present time (filtering), at a future time (prediction), or at a time in the past (interpolation or smoothing).

Velo and Walker, from the University of Cincinnati [6], worked on the development of the extended Kalman filter technique to estimate aerodynamic parameters. This approach, compared to other techniques, places no linearity restrictions on the states and parameters in the dynamic equations describing the system. It can also handle time-variant systems and does not require the system's stability. The extended Kalman filter produces estimates of the parameters that minimize the mean square error in the parameter estimates themselves, whereas most other maximum-likelihood and least-square techniques are designed to minimize a cost function that is based on matching the output variable behavior. However the Kalman filter approach requires the designer to select the statistical properties of the process, measurement noise, and model for the parameter dynamics. Some of this information is unknown to the user, so the design becomes somewhat challenging.

Neural networks have also been used for system identification. Recent work in the field of parameter estimation includes a few techniques using computational neural networks.

Online implementation of parameter estimation has become possible with the increase in onboard processing power and memory. Online techniques are mainly divided into time domain and frequency domain methods.

2.1 Time Domain Methods

Time domain techniques that are implemented online prefer least-square algorithms over methods based on the use of gradient and the Hessian, due to their low computational requirement and better convergence characteristics. Thus most of the online time domain parameter estimation methods use some form of modifications in the least-square regression method, such as recursive least square [7, 8], recursive least square with a forgetting factor [9], a

modified sequential least square [10], a real-time batch least square [11, 12], and extended Kalman filtering [13].

Yongkyu Song and his group [14] described an online time domain-based parameter estimation technique while comparing it with online frequency domain methods. They used a locally weighted regression algorithm for parameter identification. The details of this method are originally described in references [15-17]. The method is essentially a least-square method with a local weighting and retention scheme of the most valuable data to the current flight point. The technique uses the retention and deletion scheme along with the weighting mechanism, which retains and places more weight on the more useful data and deletes the less informative data. The technique exhibits desirable accuracy of the estimates, small convergence time, and robustness to noise.

2.2 Frequency Domain Method

Eugene A. Morelli, Research Engineer at the NASA Langley Research Center [18], developed a real-time parameter estimation technique in the frequency domain originally proposed by Klein [19]. This method uses aircraft equation error in the frequency domain with a recursive Fourier transform for the real-time data analysis. The technique analyzes data in the frequency domain with a recursive Fourier transform using fixed discrete values within the frequency range for the dynamic motion of interest. The technique was demonstrated for accurate model parameter estimates with appropriate error bounds using simulation examples and flight test data form F-18 High Alpha Research Vehicle. The technique demonstrated the ability of automatic noise filtration and robustness to high noise levels. It has a low computational requirement as compared to time domain methods, and can be implemented aboard an aircraft in real time.

2.3 Neural Network Methods

Linse and Stengel [20] developed a method that uses neural network models with an estimation-before-modeling paradigm for online training information to generate aerodynamic coefficients for the complete flight envelope. A feed-forward network is used for each parameter. The chosen network contains nine inputs (Mach number, angle of attack, pitch rate, density, angle of attack rate, altitude, throttle position, elevator deflection, and flap position), a single hidden layer with 20 logistic sigmoid nodes, and a single linear node in the output layer. To estimate the weights, a method of incorporating first-partial-derivative information is used. The results show that accurate aerodynamic coefficient models are derived from simulated flight test data using a system identification model composed of an extended Kalman-Bucy filter for state and force estimation and a neural network for aerodynamic modeling.

This system seems well designed to estimate aerodynamic parameters from flight data, or even online, but the system was not designed or tested for in-flight or sudden failure conditions. The neural networks require extensive and well-behaved training data to estimate the parameters and, therefore, may not be capable of estimating a sudden change of parameters.

Another method described by Raisinghani and Ghosh [21] uses a neural model capable of predicting generalized force and moment derivatives (i.e., $C_{L\alpha}$, C_{Lq}, $C_{L\delta}$, $C_{m\alpha}$, C_{mq}, $C_{m\delta}$) using measured motion and control variables. Motion and control variables such as angle of attack α, pitch rate q, and control surface deflection δ were used as network inputs and lift or moment coefficient (C_L or C_m) was used as output. It is also shown that such a neural model can be used to extract equivalent stability and control derivatives of an aircraft. The major advantages of using this method are that it eliminates the need for postulating and solving coupled equations of

motion of an aircraft, it does not require any initial guess for the parameter values, and the on-board trained neural model has the potential for obtaining online parameter estimates.

CHAPTER THREE

3. NEURAL-BASED ADAPTIVE FLIGHT CONTROLLER

An advanced flight control system was developed for a general aviation aircraft to have the capability of providing compensation for unanticipated failures and making an aircraft easy to fly for a non-pilot by decoupling the control inputs. This method was developed by Pesonen, Steck, and Rokhsaz from Wichita State University, and Sam Bruner and Noel Duerksen from Raytheon Aircraft Company [22].

This method uses dynamic inverse control to decouple the flight controls and modify the handling qualities of the aircraft [23]. Artificial neural networks are used to provide compensation for modeling errors in the inverse controller and also to provide adaptation to any kind of unanticipated failures during flight. The controller is demonstrated for longitudinal control of an aircraft in this thesis but can also be extended to lateral directional control.

This chapter explains the development of the advanced flight controller, which includes the development of the inverse control, beginning with basic aircraft equations and development of the artificial neural network that is responsible for the adaptive nature of the controller.

Figure 3.1 shows the control system setup along with a Matlab/Simulink model of the aircraft. Beginning at the left side of the figure, the desired airspeed (V_c) and flight path angle (γ_c) was commanded, which was then passed through an output feedback linear control. At this point the airspeed error was converted into the forward acceleration, and the flight path angle error was converted to pitch acceleration, which were then passed on to the inverse controller. The inverse controller uses an inverse form of the linearized aircraft equations, to calculate the required thrust and elevator deflection for given forward and pitch accelerations.

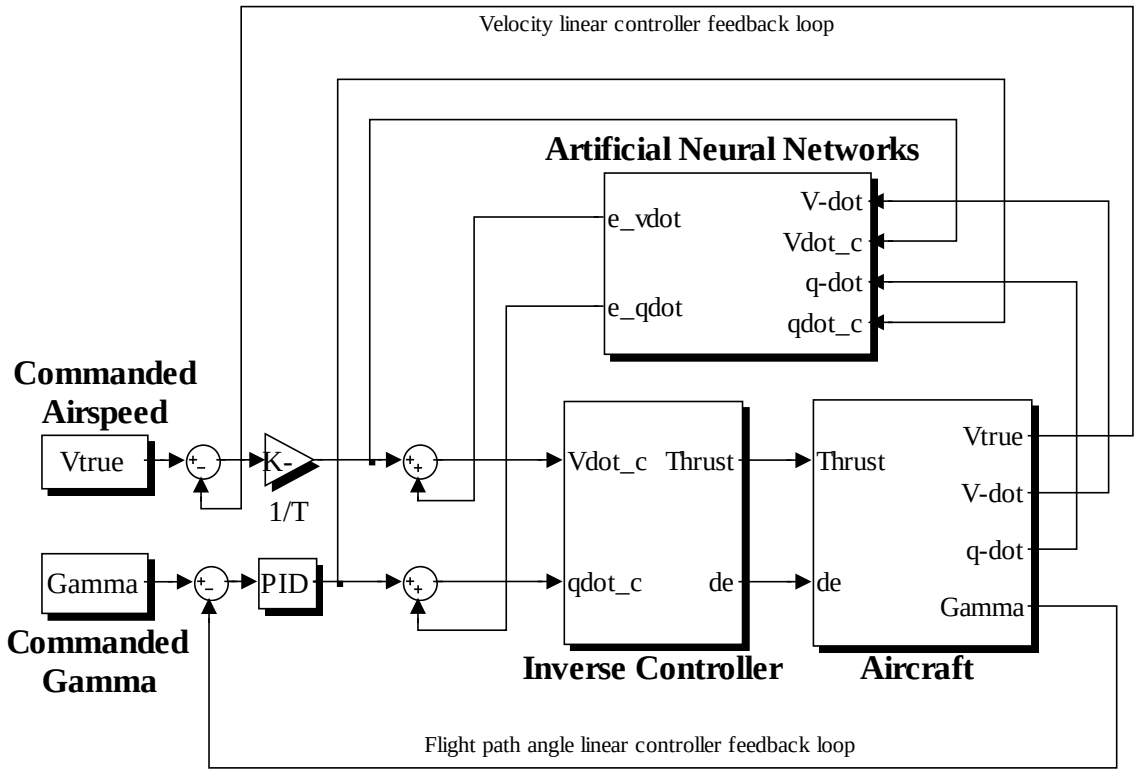


Figure 3.1. Control system setup plus Matlab/Simulink aircraft model.

Outputs of the inverse controller, thrust, and elevator are then fed to the aircraft. The artificial neural network block takes the commanded and actual accelerations as inputs and provides error compensation signals as outputs, which are added to the acceleration inputs to the inverse controller. If the inverse controller is an exact inverse of the aircraft and there is no modeling error, the outputs of the artificial neural network block will be zero. But in the presence of the modeling error, artificial neural networks are trained to output error compensation signals that adjust the input of the inverse controller such that the actual airspeed and flight path angle match the commanded airspeed and flight path angle. The major reason for the modeling error can be the differences between the values of the parameters used in the inverse controller and those of the actual aircraft.

3.1 Aircraft Model used for Simulations

A mathematical model of Beechcraft Bonanza F33A developed in Simulink was used for the purpose of simulation. Bonanza is a four to six seat high performance light aircraft. Few of its features include low straight wing, single continental piston engine with two- or three-blade propeller, swept vertical tail with small dorsal fin and low horizontal tail, and retractable tricycle landing gear. It has a maximum ramp weight of 3412 lb. With a range of 838 nautical miles it can reach a maximum cruise speed of 195 knots and stalls at about 55 knots with flaps down. With a wingspan of 33 feet and 6 inches, it is 26 feet 7 inches in length and 8 feet 3 inches in height.

Figure 3.2. Aircraft used to produce model the simulation

Aerodynamic model of the aircraft for longitudinal control includes lift, drag and pitching moment equations.

Lift equation is given by $$C_L = C_{L_0} + C_{L\alpha}\alpha + C_{Lq}\hat{q} + C_{L\dot{\alpha}}\hat{\dot{\alpha}} + C_{L_{\delta_e}}\delta e \ . \quad (3.1)$$

Drag equation used in the model is $$C_D = C_{D_0} + C_{D_k} . (C_{L_0} + C_{L\alpha}\alpha)^2 + C_{D_{\delta e}}\delta e . \quad (3.2)$$

And pitching moment equation is $$C_m = C_{m0} + C_{m\alpha}\alpha + C_{mq}\hat{q} + C_{m\dot{\alpha}}\hat{\dot{\alpha}} + C_{m\delta_e}\delta e \ . \quad (3.3)$$

The numerical values of the coefficients presented in this thesis have been removed in order to protect the proprietary nature of the actual aerodynamic parameters of the Bonanza aircraft.

3.2 Development of Linear Inverse Control Setup

The longitudinal aircraft model (aircraft block in Figure 3.1) [24] was based on the nonlinear equations in the body fixed *x* and *z*-axis directions.

In body coordinates, the equations of motion for longitudinal maneuvers with the wings level [25] are

$$\dot{U} + g\sin(\theta) = -WQ + \frac{1}{m}F_{A_x} + \frac{1}{m}T\cos(\varphi_T) + \frac{1}{m}F_{X_{\delta_e}}\delta_e \tag{3.4}$$

$$\dot{W} - g\cos(\theta) = +UQ + \frac{1}{m}F_{A_z} - \frac{1}{m}T\sin(\varphi_T) + \frac{1}{m}F_{Z_{\delta_e}}\delta_e \tag{3.5}$$

The pitching moment equation is given as

$$\ddot{\theta} = \frac{1}{I_{yy}}M_A - \frac{1}{I_{yy}}Td_T + \frac{1}{I_{yy}}M_{\delta_e}\delta_e \tag{3.6}$$

where d_T is the thrust moment arm, and φ_T is the thrust angle relative to the fuselage axis.

The aircraft block uses the Simulink Aerospace 6 D.O.F. block with a nonlinear drag polar and linear lift and pitching moment models. The lateral forces and moments were set to zero. Since the aircraft model is longitudinal, thrust, and elevator deflection were the inputs, and the longitudinal state variables (pitch angle θ, forward velocity U, and vertical velocity W) were

outputs. The derived variables (angle of attack α and flight path angle γ) were also calculated outputs.

The true airspeed V and flight path angle γ are given as

$$V = \sqrt{U^2 + W^2} \tag{3.7}$$

$$\theta = \gamma + tan^{-1}\left(\frac{W}{U}\right) \tag{3.8}$$

Pilot-commanded velocity V_c and flight path angle γ_c tracking errors were calculated using linear output feedback loops from the aircraft outputs (true velocity and actual flight path angle). Tracking error was fed to a proportional controller for a velocity feedback loop (see Figure 3.1) and to a proportional derivative controller for flight path angle feedback loop (see Figure 3.1). These controllers output commanded linear acceleration $\dot{V}_c$ and commanded angular acceleration $\ddot{\gamma}_c$, which are inputs to a linear dynamic inverse controller. This linear inverse controller was used to calculate the required inputs to the aircraft model, i.e., thrust and elevator deflection, needed to produce commanded accelerations.

The equations used in the linear inverse controller to solve for the required thrust and elevator deflection from commanded velocity and angular accelerations are discussed here. The commanded accelerations are as follows:

Inputs: $\dot{V}_c$ and $\ddot{\gamma}_c$ (forward and pitch acceleration)

Equations (3.3) to (3.8) show that with the accelerations specified, there are five equations and five unknowns:

Unknowns: $\theta, T, \delta_e, U, W$

The three auxiliary equations, which go along with the equations above:

$$Q = \dot{\theta} \tag{3.9}$$

$$\dot{\theta} = \dot{\gamma} + \dot{\alpha} = \dot{\gamma} + \frac{\dot{W}U - \dot{U}W}{V_c^2} \qquad \text{using equation (3.8)} \tag{3.10}$$

$$\dot{V}_c V_c = \dot{U}U + \dot{W}W \qquad \text{using equation (3.7)} \tag{3.11}$$

Equations (3.3) through (3.8) must be solved simultaneously for the given time histories of V and γ.

Since the equations of motion (3.3), (3.5), and (3.6) are in the body fixed axis system (X,Z), they must be converted to stability coordinates, since commanded forward acceleration $\dot{V}_c$ and commanded flight path angular acceleration $\ddot{\gamma}_c$ are in stability coordinates (i.e., axis system parallel and normal to the velocity vector).

By multiplying equation (3.4) by $\left(\frac{W}{V_c^2}\right)$ and equation (3.5) by $\left(\frac{U}{V_c^2}\right)$, and subtracting equation (3.4) from (3.5) then

$$\frac{U\dot{W} - W\dot{U}}{V_c^2} - \frac{g}{V_c}\cos(\theta - \alpha) = Q + \frac{1}{mV_c}\left[F_{A_z}\cos(\alpha) - F_{A_x}\sin(\alpha)\right]$$

$$- \frac{T}{mV_c} sin(\alpha + \varphi_T) + \frac{1}{mV_c}\left[F_{Z_{\delta_e}} cos(\alpha) - F_{X_{\delta_e}} sin(\alpha) \right]\delta_e$$

Where $cos(\theta - \alpha) = cos(\gamma)$

$$Q = \dot{\theta} = \dot{\gamma} + \dot{\alpha}$$

$$F_{A_z} cos(\alpha) - F_{A_x} sin(\alpha) = - \bar{q} S C_L\big|_{\delta_e = 0}$$

$$F_{Z_{\delta_e}} cos(\alpha) - F_{X_{\delta_e}} sin(\alpha) = - \bar{q} S C_{L_{\delta_e}}$$

Therefore, rearranging and simplifying results in

$$\dot{\gamma} + \frac{g}{V_c} cos(\gamma) - \frac{\bar{q}S}{mV_c} C_L\big|_{\delta_e = 0} - \frac{T}{mV_c} sin(\alpha + \varphi_T) - \frac{\bar{q}S}{mV_c} C_{L_{\delta_e}} \delta_e = 0. \qquad (3.12)$$

In the inverse problem, this equation has three unknowns: $\alpha, T,$ and δ_e. Two other equations with the same unknowns are needed.

Now, multiplying equation (3.4) by $\left(\frac{U}{V_c^2}\right)$ and equation (3.5) by $\left(\frac{W}{V_c^2}\right)$ and adding

$$\frac{\dot{V}_c}{V_c} - \frac{g}{V_c} sin(\theta - \alpha) = \frac{1}{mV_c}\left[F_{A_z} sin(\alpha) + F_{A_x} cos(\alpha) \right]$$

$$+ \frac{T}{mV_c} cos\left(\alpha + \varphi_T\right) + \frac{1}{mV_c}\left[F_{Z_{\delta_e}} sin\left(\alpha\right) + F_{X_{\delta_e}} cos\left(\alpha\right) \right]\delta_e .$$

Where $F_{A_z} sin(\alpha) + F_{A_x} cos(\alpha) = - \bar{q} S C_D\big|_{\delta_e = 0}$

$$F_{Z_{\delta_e}} sin(\alpha) + F_{X_{\delta_e}} cos(\alpha) = - \bar{q} S C_{D_{\delta_e}}$$

Then, again rearranging and simplifying results in

$$- \dot{V}_c - g\sin(\gamma) - \frac{\bar{q}S}{m} C_D\big|_{\delta_e=0} + \frac{T}{m}\cos(\alpha+\varphi_T) - \frac{\bar{q}S}{m} C_{D_{\delta_e}} \delta_e = 0 \tag{3.13}$$

From equation (3.6), since $\ddot{\theta} = \ddot{\gamma} + \ddot{\alpha}$

Then

$$\ddot{\alpha} + \ddot{\gamma} = \ddot{\theta} = \frac{\bar{q}S\bar{c}}{I_{yy}}\left[C_m - C_T \frac{d_T}{\bar{c}} + C_{m_{\delta_e}} \delta_e \right] \tag{3.14}$$

Now equations (3.12) through (3.14) can be solved simultaneously for α, T, and δ_e. Other variables can be determined from the auxiliary equations. Equations (3.12) and (3.13) are algebraic expressions, while equation (3.14) is a differential equation. Assuming $C_{D_{\delta_e}} \approx 0$, an approximate solution can be obtained. Then equation (3.13) can be solved for *T*, from which equation (3.12) can be solved for δ_e. Substituting these in equation (3.14) results in a single differential equation for α.

A speed control lever (replacing the cockpit throttle) is set to a commanded airspeed V_c. This is compared to the feedback of the actual airspeed. A linear proportional controller that outputs a commanded acceleration processes the resulting error $\dot{V}_c$ (see Figure 3.1). Similarly, a longitudinal stick can be set up to command the flight path angle γ_c. The tracking error of this variable is fed to a PD controller that outputs a commanded angular acceleration $\ddot{\gamma}_c$ (see Figure 3.1). The angle of attack (α) is taken from the aircraft and used in equations (3.13) and (3.14) to solve for the control variables T and δ_e. As long as the control saturation does not occur, these equations are linear in terms of the controls.

Equations (3.15) and (3.16) used in the inverse controller are

$$T_c = \frac{1}{\cos(\alpha + \varphi_T)}\left(m\dot{V}_c + (mg)\sin(\gamma) + \bar{q}S(C_D)_{\delta_e=0}\right) \tag{3.15}$$

$$\begin{aligned}\delta_{e_c} = \frac{C_m(\delta_e)}{C_{m_{\delta_e}}} = \frac{I_{yy}}{\bar{q}S\bar{c}C_{m_{\delta_e}}}(\ddot{\gamma}_c \text{ or } \ddot{\theta}_c) - \frac{(C_m)_{\delta_e=0}}{C_{m_{\delta_e}}} \\ + \frac{d_T}{\bar{q}S\bar{c}\cos(\alpha + \varphi_T)C_{m_{\delta_e}}}\left(m\dot{V}_c + (mg)\sin(\gamma) + \bar{q}S(C_D)_{\delta_e=0}\right)\end{aligned} \tag{3.16}$$

where

$$(C_D)_{\delta_e=0} = C_{D_0} + C_{D_k}.\,(C_{L_0} + C_{L_\alpha}\alpha)^2$$

$$(C_m)_{\delta_e=0} = C_{m0} + C_{m\alpha}\alpha + C_{mq}\hat{q}$$

It should be noted that in equation (3.16), $\ddot{\alpha}$ is assumed to be negligible.

For no modeling error in gamma tracking, the combination of inverse controller and aircraft acts like a double integrator block that takes a commanded angular acceleration and outputs aircraft angular displacement. Also for no modeling error in velocity tracking, the inverse controller and aircraft combination acts like a single integrator block that takes a commanded acceleration and outputs aircraft velocity.

The gamma response PID controller parameters were chosen to produce a second order response with damping ratio of 1.5 and a natural frequency of 0.8 resulting in a rise time of 8 sec. Recall these are for γ and not for θ response. Since the elevator has deflection limits and actuator dynamics, there is a limit on the speed of the elevator response and how fast the short period of the aircraft can be damped. The velocity feedback loop has a first order response, so a proportional controller with time constant as 15.0 seconds is chosen to generate the desired aircraft response.

3.3 Artificial Neural Networks for Controller Error Compensation

The linear inverse controller is only able to fly the aircraft with no modeling error. To compensate for the modeling errors in the inverse controller, artificial neural networks are required. Since longitudinal control only is used, two artificial neural networks are necessary, one each for velocity and flight path angle control.

3.3.1 Fundamental Artificial Neural Network

Artificial neural networks are mapping devices that have the capability to map any complex or unknown function, provided they have enough information based on observation [26, 27, 28].

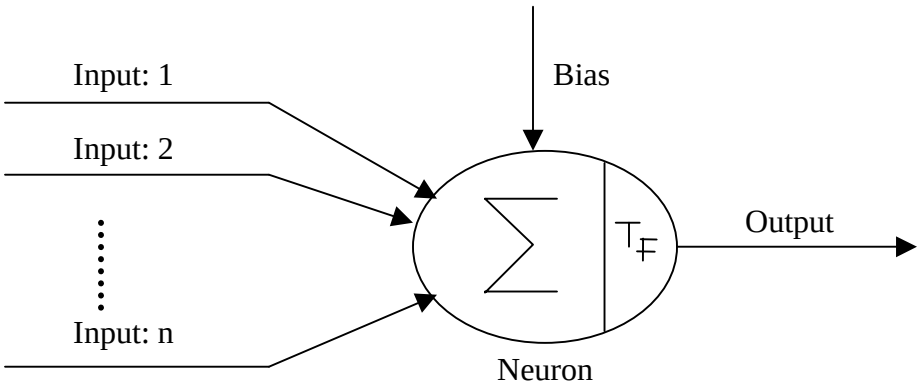


Figure 3.3. Processing element of artificial neural network.

Neural networks are inspired by the biological nervous system. Their structure is composed of artificial neurons (Figure 3.2) connected to each other by weight values. An artificial neuron, also called processing element, is made up of primarily four parts: combining function, transfer function, element output, and weights. The combination function collects the inputs and adds them, which is then passed through the transfer function and then sent to the

output link. The output of one or more neuron is fed as input to another neuron and so on, thus forming a cluster of many neurons connected to each other by output to input links (Figure 3.3).

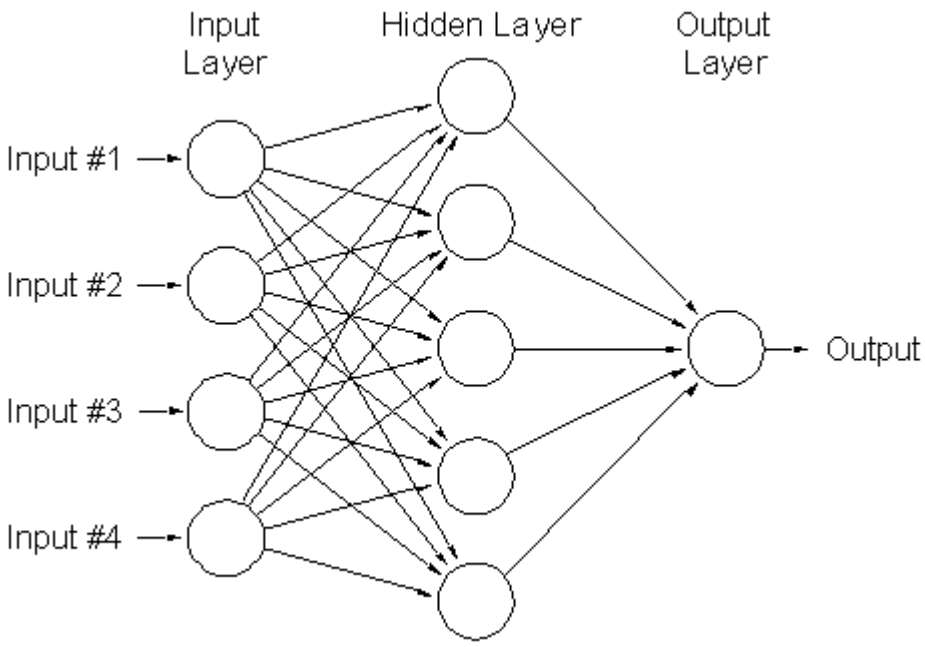


Figure 3.4. Artificial neural network, processing elements, and connections.

These links are associated with gains, which serve as the system's memory or weights. The entire cluster along with weights is called a neural network. It processes an input signal from input neurons to output neurons through this connectivity. Neural networks learn input-output relationships from example data. While training, weights over the connections are updated using different training rules to produce an approximation to the input-output relation.

Artificial neural networks are used to compensate for modeling errors caused by, partial actuator failure, weight change, center of gravity change, and other degraded flight conditions that are not modeled by the inverse controller. With longitudinal flight control, two networks are used: one for flight velocity control and another for flight path angle control.

3.3.2 Artificial Neural Network Setup

The artificial neural network block (Figure 3.1) contains velocity and pitch networks, along with appropriate inputs to each and required outputs. The block performs two tasks: (1) training the neural network, and (2) computing the required output signal from the networks to compensate for the error.

For the neural network to compensate for error, it must output a signal (error correction, $e(t-1)$) that can be added to the commanded signal to obtain an aircraft response that matches the commanded signal.

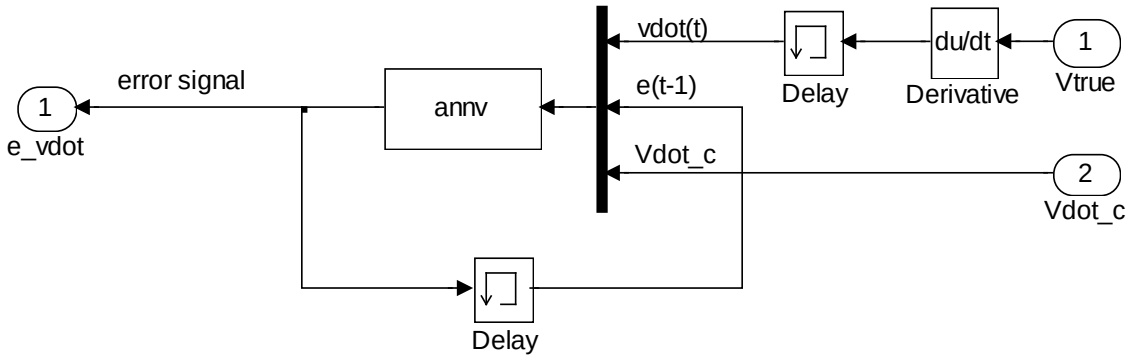


Figure 3.5. Neural network connections for velocity control.

The velocity network as shown in Figure 3.4 is trained at each time step with input as actual acceleration at the previous time step $\dot{V}(t-1)$ and target output as the network output (error correlation signal) that produced that acceleration at the previous time step $e_V(t-1)$. Then at the current time step, the neural network is tested with current commanded acceleration $\dot{V}_c(t)$ as input, and new error correlation signal is generated $e_V(t)$. This new error correlation signal is

added to the commanded signal that goes into the inverse controller. It also will be used in the following time step as the target output to train the network. To elaborate, the output signal (error correlation) from the neural network is added to the commanded acceleration $\dot{V}_c(t)$ to make the actual acceleration $\dot{V}(t)$ match the commanded $\dot{V}_c(t)$. Input used is $\dot{V}(t-1)$ and $e_V(t-1)$ is used as target output, because the aircraft velocity response $\dot{V}(t-1)$ is a result of using that error compensation signal $e_V(t-1)$ and they are correlated. The neural network is trained using this known relation to predict the error compensation signal for some future time step. Now if the inverse controller is an exact inverse of the aircraft, the neural network will learn to output an error correlation signal of zero. If any modeling error exists, the neural network will output the error correlation signal to compensate for that error.

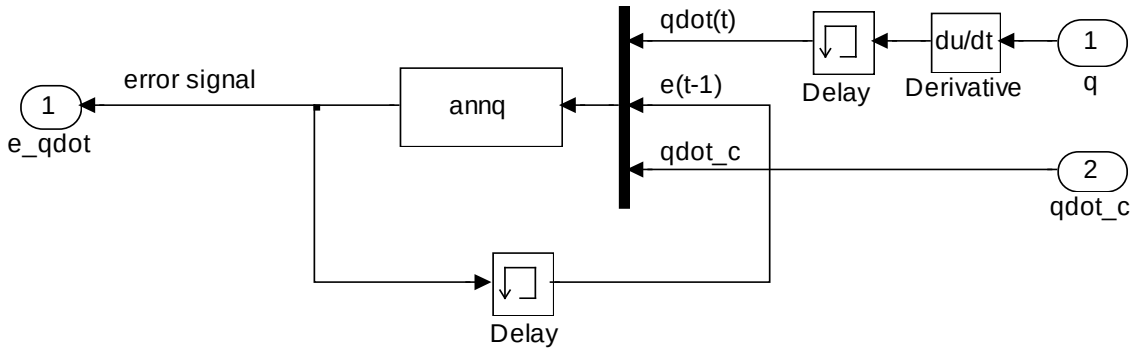


Figure 3.6. Neural network connections for pitch control

Similarly, the pitch network as shown in Figure 3.5, is trained at each time step with input as the pitch acceleration at the pervious time step $\dot{q}(t-1)$ and target output as the error correlation at the previous time step $e_q(t-1)$. This is then tested for the current pitch acceleration $\dot{q}(t)$ as input, and error correlation $e_q(t)$ is generated as output. The error correlation is added to the commanded pitch acceleration to the inverse controller.

Both networks have a similar structure with two hidden layers, both with five processing elements each. The first layer uses a tan-sigmoid activation function, whereas the second and output layers use linear activation functions. It was observed that the structure was ideal for the pitch network, but for the velocity network, some variations in the structure made no significant effect on the results.

The selection of the training algorithm used in the networks was done on the basis of the number of training cycles and how well the network responded to any change in flight condition. A training algorithm called Levenberg-Marquardt [27] was found most suitable for the pitch network, since it was able to train the network in about one or two cycles. For the velocity network gradient descent backpropagation [27] was found to be ideal. Artificial neural networks were developed initially in Neuralworks Professional II [28] and implemented using the Matlab Artificial Neural Networks toolbox.

CHAPTER FOUR

4. ONLINE PARAMETER ESTIMATION

The main purpose of the parameter estimation technique explained in this chapter was to tune parameters inside the inverse controller block and provide permanent adaptation in the controller. This replaced the immediate compensation provided by the artificial neural network and, hence, allowed the network to compensate for a future failure.

The system was designed so that if any parameter changes in the actual aircraft, the artificial neural network will provide immediate compensation for the modeling error caused by the changed parameter in the aircraft. As the online parameter estimation algorithm detects the modeling error, it starts to work towards identifying the changed parameter and its new value. Then these estimated parameters are slowly updated into the inverse controller. The algorithm remains active and keeps updating the parameter inside the inverse controller until the modeling error reaches an acceptable small value. When the changed parameter is updated in the inverse controller, the compensation provided by the neural networks starts decreasing, and gradually approaches zero as the changed parameter approaches its new value in the inverse controller.

4.1. Online Parameter Estimation System Setup

Figure 4.1 shows a conventional setup for the Neural Based Adaptive Flight Controller including two extra blocks, the Model Aircraft block and the Parameter Estimation block. These two blocks form an Online Parameter Estimation System.

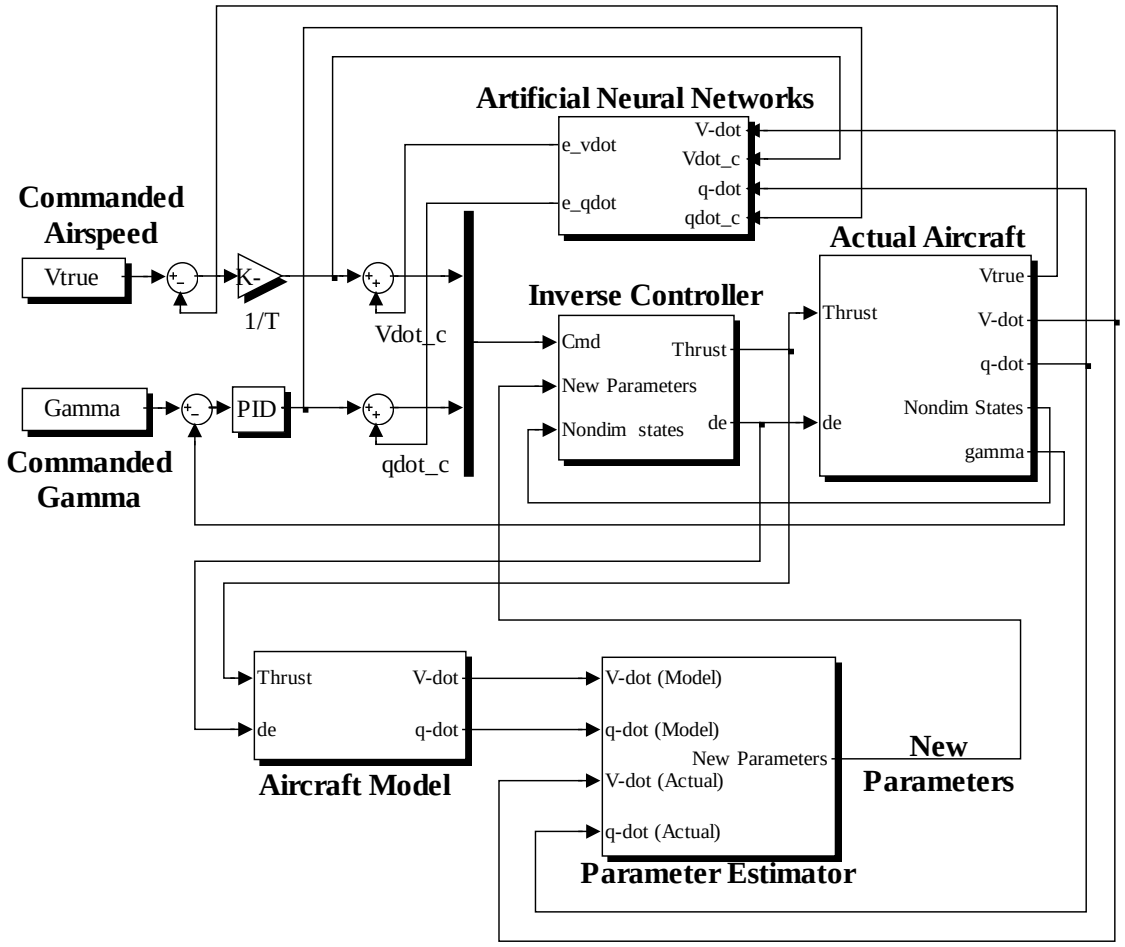


Figure 4.1. Neural-based adaptive flight controller with online parameter estimation system

4.1.1. Model Aircraft Block

The aircraft model block (Figure 4.1) contains a linear aircraft model. A model was generated using the linear aircraft equations, which were generated during the development of an inverse controller as explained in Section 3.1.

Equations (3.13) and (3.14) were rearranged to obtain the aircraft model with outputs as accelerations from input thrust and elevator.

$$\dot{V}_{model} = \left(T_c cos(\alpha + \varphi_T) - \bar{q}S(C_D)_{\delta_e=0} - (mg)sin(\gamma)\right)\Big/m \tag{4.1.3}$$

$$\ddot{\gamma}_{model} = \left(\delta_{e_c} C_{m_{\delta e}} + (C_m)_{\delta_e=0} - \frac{T_c d_T}{\bar{q}S\bar{c}}\right).\frac{\bar{q}S\bar{c}}{I_{yy}} - \ddot{\alpha} \tag{4.1.4}$$

where

$$(C_D)_{\delta_e=0} = C_{D_0} + C_{D_K} \cdot (C_{L_0} + C_{L_\alpha}\alpha)^2$$

$$(C_m)_{\delta_e=0} = C_{m0} + C_{m\alpha}\alpha + C_{mq}\hat{q}$$

It should be noted that d_T is considered 0 in this case.

The parameters used for the linear aircraft block are the same as in the linear inverse controller block. This provides a tool to compare the response of the current aircraft model with the actual aircraft. “Current aircraft model” means the currently used aerodynamic and stability parameters.

By comparing the response of the linear aircraft model and the actual aircraft, the modeling error present in the system can be found. Modeling error is caused mainly by parameters that have different values in actual aircraft and linear aircraft block. By investigating the quantity and time history of the modeling error, parameters can be updated such that the modeling error at all times and at all flight conditions goes to zero. If the modeling error is zero at all flight conditions, then the values of the actual parameters are known since the actual aircraft can be modeled perfectly over a reasonable range of flight conditions.

4.1.2. Parameter Estimator Block

The parameter estimator block (Figure 4.1) performs the estimation of the parameters and updates their new values in the inverse controller.

The parameter estimation technique is a simple application of a gradient descent optimization, whereby a single update equation is derived for each parameter and used in

different combinations with other update equations depending upon the type of modeling error observed in the system. The technique is very simple and straightforward as compared to other techniques present in the field, as mentioned in chapter 2, which use different methods like recursive Fourier transform, maximum likelihood estimation, and computational neural networks. The technique developed here requires much less computational power as compared to other techniques and is developed for online implementation. It produces satisfactory results with a limited number of parameters.

4.1.2.1. Development of the Parameter Estimation Equations

The general gradient descent update equation is give by

$$P(t) = P(t-1) - K*\left(\frac{\partial(CF)}{\partial(P)}\right) \tag{4.1.5}$$

where, P stands for parameter value, K for learning rate and CF for cost function.

The cost function is defined as

$$CF = \frac{1}{2}\left(Acceleration\ (Actual) - Acceleration\ (Model)\right)^2 \tag{4.1.6}$$

The modeling error (ME) is defined as

$$ME = \left(Acceleration(Actual) - Acceleration(Model)\right) \tag{4.1.7}$$

Differentiating the cost function with respect to the parameter gives

$$\left(\frac{\partial(CF)}{\partial(P)}\right) = (ME)*\left(\frac{-\partial\left(Acceleration(Model)\right)}{\partial(P)}\right) \tag{4.1.8}$$

Now by substituting equation (4.1.8) in equation (4.1.5), the parameter update equation becomes

$$P(t) = P(t-1) + K*\left(\frac{\partial\left(Acceleration(Model)\right)}{\partial(P)}\right)*ME \tag{4.1.9}$$

where K = Learning rate (constant)

$\left(\frac{\partial\left(Acceleration\left(Model\right)\right)}{\partial\left(P\right)}\right)$ = Partial Derivative of the Acceleration equation (4.1.3) or (4.1.4) with respect to the parameter which is updated.

Modeling Error (*ME*) = $\dot{V}_{error}$ or $\ddot{\gamma}_{error}$

The tasks of parameter estimation block can be divided into three steps:

1) Modeling error calculation.

2) Parameter estimation using estimation equations.

3) Update of parameters in inverse controller.

4.1.2.2. Modeling Error Calculation

First the outputs of the actual aircraft $\dot{V}_{actual}$ and $\ddot{\gamma}_{actual}$ (i.e., linear and pitch acceleration) are compared with the outputs of the model aircraft $\dot{V}_{model}$ and $\ddot{\gamma}_{model}$.

The individual modeling errors are calculated as

$$\dot{V}_{error} = \left(\dot{V}_{actual} - \dot{V}_{model}\right) \tag{4.1.10}$$

$$\ddot{\gamma}_{error} = \left(\ddot{\gamma}_{actual} - \ddot{\gamma}_{model}\right) \tag{4.1.11}$$

At a given flight condition, if the parameters used in inverse controller and aircraft model are the same as that of the actual aircraft, the values of $\dot{V}_{error}$ and $\ddot{\gamma}_{error}$ will be zero. If one or both of the errors are nonzero then one or more parameters have changed during the flight, and must be estimated and updated.

4.1.2.3. Parameter Update Equations

The equations use a simple gradient-descend method to update the parameters at each time step. The general form of the update equation is given in equation (4.1.9).

This thesis examines the estimation of six major aerodynamic and stability parameters, since incorrect values of these parameters can affect the performance of the controller significantly. These parameters are $C_{m0}, C_{m\alpha}, C_{m\delta e}, C_{mq}, C_{L\alpha}$, and mass. Each of these parameters has one update equation, which is derived and written in the form described in equation (4.1.9). Using the generic update equation (4.1.9), the two main steps are as follows:

(a) Derive an expression for the partial derivative of the acceleration equations derived with respect to each parameter. This will be different for each parameter. The expression provides the direction and magnitude of how much each parameter contributes to the accelerations.

(b) Select a good value of learning rate *K* for each parameter update, which is generally chosen by experience and is usually different for each parameter. This value also provides the ability to choose the rate at which one wants to converge to a new estimate.

4.1.2.3.1. Update Equation for C_{m0} :

C_{m0} is the pitching moment of the aircraft at zero angle of attack.

(a) Examining the acceleration equations (4.1.3) and (4.1.4), C_{m0} appears only in the $\ddot{\gamma}_{model}$ equation (4.1.4). Therefore,

$$\left(\frac{\partial \ddot{\gamma}}{\partial C_{m0}}\right) = \left(\bar{q}S\bar{c} \Big/ I_{yy}\right) \tag{4.1.12}$$

(b) The value of *K* (learning rate) was selected by trial and error. The selection criteria for the value of *K* were the minimization of convergence time and stability of the algorithm. Increasing the value of *K* reduces the convergence time. But if the value of *K* is too large, it can drive the modeling error to the opposite sign and to a larger magnitude in a single time step, which will cause the process to diverge and become unstable.

K value selected for Routine 1,2 and 3 (Section 4.2.1, 4.2.2 and 4.2.2) was 0.001.

The final C_{m0} update equation is written as

$$C_{m0}(t) = C_{m0}(t\text{-}1) + \left[K * \left(\bar{q}S\bar{c} \Big/ I_{yy} \right) * \ddot{y}_{error} \right] \tag{4.1.13}$$

4.1.2.3.2. Update Equation for $C_{m\alpha}$:

$C_{m\alpha} = \partial C_m / \partial \alpha$, is the variation of aircraft pitching moment coefficient with angle of attack (equation (4.1.4)).

(a) Examining the acceleration equations (4.1.3) and (4.1.4), $C_{m\alpha}$ appears only in the $\ddot{y}_{model}$ equation (4.1.4). Therefore,

$$\left(\frac{\partial \ddot{y}}{\partial C_{m\alpha}} \right) = \alpha . \left(\bar{q}S\bar{c} \Big/ I_{yy} \right) \tag{4.1.14}$$

(b) The value of *K* (learning rate) was selected by trial and error.

K value selected for Routine 1 and 3 (Section 4.2.1 and 4.2.3) was 0.05.

K value selected for Routine 2 (Section 4.2.2) was 0.01.

The final $C_{m\alpha}$ update equation is written as

$$C_{m\alpha}(t) = C_{m\alpha}(t-1) + \left[K^{*}\alpha . \left(\bar{q}S\bar{c} \Big/ I_{yy} \right) * \ddot{\gamma}_{error} \right] \tag{4.1.15}$$

4.1.2.3.3. Update Equation for $C_{m\delta e}$:

$C_{m\delta e} = \partial C_m / \partial \delta e$ is the variation of aircraft pitching moment coefficient with elevator deflection angle (equation (4.1.4)).

(a) Examining the acceleration equations (4.1.3) and (4.1.4), $C_{m\delta e}$ appears only in the $\ddot{\gamma}_{model}$ equation (4.1.4). Therefore,

$$\left(\frac{\partial \ddot{\gamma}}{\partial C_{m\delta e}} \right) = \delta e . \left(\bar{q}S\bar{c} \Big/ I_{yy} \right) \tag{4.1.16}$$

(b) The value of *K* (learning rate) was selected by trial and error.

K value selected for Routine 1 (Section 4.2.1) was 0.1 and for Routine 3 (Section 4.2.3) was 0.2.

The final $C_{m\delta e}$ update equation is written as

$$C_{m\delta e}(t) = C_{m\delta e}(t-1) + \left[K^{*}\delta e . \left(\bar{q}S\bar{c} \Big/ I_{yy} \right) * \ddot{\gamma}_{error} \right] \tag{4.1.17}$$

4.1.2.3.4. Update Equation for C_{mq} :

$C_{mq} = \partial C_m / \partial \hat{q}$ is the variation of aircraft pitching moment coefficient with pitch rate (equation (4.1.4)).

(a) Examining the acceleration equations (4.1.3) and (4.1.4), C_{mq} appears only in the $\ddot{\gamma}_{model}$ equation (4.1.4). Therefore,

$$\left(\frac{\partial \ddot{y}}{\partial C_{mq}}\right) = \hat{q}.\left(\bar{q}S\bar{c} \Big/ I_{yy}\right) \tag{4.1.18}$$

(b) The value of *K* (learning rate) was selected by trial and error.

K value selected for Routine 1 (Section 4.2.1) was 100000. The modeling error $\ddot{y}_{error}$ due to change in C_{mq} was vary small, so the learning rate was chosen as a large number to reduce the convergence time.

The final C_{mq} update equation is written as

$$C_{mq}(t) = C_{mq}(t-1) + \left[K * \hat{q}.\left(\bar{q}S\bar{c} \Big/ I_{yy}\right) * \ddot{y}_{error}\right] \tag{4.1.19}$$

4.1.2.3.5. Update Equation for $C_{L\alpha}$:

$C_{L\alpha} = \partial C_L / \partial \alpha$ is the variation of aircraft lift coefficient with angle of attack (equation (4.1.3)).

(a) Examining the acceleration equations (4.1.3) and (4.1.4), $C_{L\alpha}$ appears only in $\dot{V}_{model}$ equation (4.1.3). Therefore,

$$\left(\frac{\partial \dot{V}}{\partial C_{L\alpha}}\right) = \left[\left(\bar{q}S \Big/ m\right).\left\{2.C_{DK}.\left(C_{L0}.\alpha + C_{L\alpha}.\alpha^2\right)\right\}\right] \tag{4.1.20}$$

(b) The value of *K* (learning rate) was selected based on trial and error method.

K value selected for Routine 1 and 2 (Section 4.2.1 and 4.2.2) was 0.5.

The final $C_{L\alpha}$ update equation is written as

$$C_{L\alpha}(t) = C_{L\alpha}(t-1) + \left[K^* \left\{ \left(\bar{q}S / m \right) . \left\{ 2.C_{DK} . \left(C_{L0} . \alpha + C_{L\alpha} . \alpha^2 \right) \right\} \right\} * \dot{V}_{error} \right] \qquad (4.1.21)$$

4.1.2.3.6. Update Equation for Mass, *m*:

Mass is a critical quantity to be estimated for the controller to maintain proper velocity control (equation (4.1.3)).

(a) Examining the acceleration equations (4.1.3) and (4.1.4), Mass (*m*) appears only in the $\dot{V}_{model}$ equation (4.1.3). Therefore,

$$\left(\frac{\partial \dot{V}}{\partial m} \right) = \left[\left(-T_c cos(\alpha + \varphi_T) + \bar{q}S(C_D)_{\delta_e = 0} \right) / m^2 \right] \qquad (4.1.22)$$

Where: $(C_D)_{\delta_e = 0} = C_{D_0} + C_{D_K} . \left(C_{L_0} + C_{L_\alpha} \alpha \right)^2$

(b) The value of *K* (learning rate) was selected by trial and error.

K value selected for Routine 1 and 2 (Section 4.2.1 and 4.2.2) was 50.

The final Mass update equation is written as

$$m(t) = m(t\text{-}1) + \left[K^* \left\{ \left(-T_c cos(\alpha + \varphi_T) + \bar{q}S(C_D)_{\delta_e = 0} \right) / m^2 \right\} * \dot{V}_{error} \right] \qquad (4.1.23)$$

All the above equations update the parameters each time step by using the most recent values of the variables.

4.1.2.4. Update of Parameters in Inverse Controller

The new estimated values of the parameters were then updated slowly in the inverse controller. The rates at which the parameters are estimated were determined by the learning rate associated with each update equation. But these estimated parameters were updated in the inverse controller at a different rate, which is much lower than the learning rate. This eliminated

the interference between the adaptation provided by the parameter estimation technique and adaptation of the neural networks. Slow update rate of the newly estimated parameters into the inverse controller makes the parameter estimation technique a slow adaptation scheme and allows the neural network to provide fast and immediate adaptation to the failure condition.

The update rates of estimated parameters in the inverse controller were chosen on the basis of the trial and error method, and are specified in Table 4.1.

Table 4.1. Parameter Update Rate in Inverse Controller

Parameter	**Rate of Update in Inverse Controller**
C_{m0}	0.00002
$C_{m\alpha}$	0.001
$C_{m\delta e}$	0.001
C_{mq}	0.01
$C_{L\alpha}$	0.004
m	0.3

4.2. Development of Parameter Estimation Routines

The basic parameter update equations are described in Section 4.1. Section 4.2 describes the development of several different combinations of parameters updated at the same time, and the assumptions made in the different routines.

As shown in the Section 4.1, two acceleration equations (i.e. $\dot{V}_{model}$ and $\ddot{\gamma}_{model}$) can be used in the entire process of updating parameters, but six parameters are estimated at the same time. This leads us to the problem of more unknowns than equations.

Examining the update equations in Section 4.1 it can be seen that parameters C_{m0}, $C_{m\alpha}$, $C_{m\delta e}$, and C_{mq} depend primarily on the pitch acceleration equation ($\ddot{\gamma}_{model}$). In other words, four variables must be estimated from one equation. Also, parameters $C_{L\alpha}$ and *m* depend primarily on forward acceleration ($\dot{V}_{model}$). Here two variables must be estimated from one equation.

Development of the different approaches was investigated as follows:

(a) Assume that the parameter causing the modeling error is already identified and is updated, thereby keeping all other parameters constant. Therefore, update only one parameter at a time.

(b) To increase the convergence speed and number of parameters to be estimated at the same time, update one parameter, keep the others constant for a short period and then update another keeping the remaining constant. Repeat this process at short intervals until all parameters converge to constant values.

(c) Assume that only one of the parameters can change at any one time during the flight, run a test to find which one has changed (this test is described in detail in Section 4.2.3), and then update that particular parameter using the update equation for that parameter.

4.2.1. Estimate One Parameter Assuming Others Correct

The routine is simple and straightforward, since only one parameter is estimated and the others are assumed to be correct and unchanged during the flight. This routine requires human assistance. Flight condition or degradation condition must be examined to determine and guess which parameter has changed and needs to be estimated. The other parameters are assumed to be

correct and kept constant. During the parameter estimation process, the flight condition was changed continuously in a sinusoidal pattern. The flight maneuver used was a sinusoidal variation of flight speed with time with amplitude of 10 knots and a time period of 125.6 seconds. This helped to estimate parameters like C_{mq}, $C_{L\alpha}$ and *m*, which contribute to the modeling error mainly during acceleration and deceleration.

In practice, it is often difficult to know which parameter has changed, but some of the degraded flight conditions and assumptions made are described here:

(a) Different distributions of load (cargo or passengers) changes the aircraft's center of gravity. The stability parameter that changes due to the movement of the center of gravity is $C_{m\alpha}$, whereas other parameters such as C_{m0} and $C_{m\delta e}$ are usually not affected much. Also, the percentage of error due to a change in C_{mq} is much less than the error due to $C_{m\alpha}$. Therefore, only one parameter, $C_{m\alpha}$, needs to be estimated and only one equation for pitch acceleration ($\ddot{y}_{model}$) needs to be solved. This can be done easily using the update equation for $C_{m\alpha}$.

(b) If there is elevator damage, such as elevator actuator degradation, then the stability parameter, $C_{m\delta e}$, will primarily be affected but not other parameters such as C_{m0}, $C_{m\alpha}$, and C_{mq}. These parameters are assumed to be correct, and are thus kept constant. Now only $C_{m\delta e}$ need to be estimated and only one equation needs to be solved, the pitch acceleration equation ($\ddot{y}_{model}$). The update equation for $C_{m\delta e}$ is used to estimate $C_{m\delta e}$.

(c) If the weight of the aircraft is different from that used in the inverse controller, it may cause degradation in velocity control, resulting in a modeling error in the velocity network ($\dot{V}_{error}$). Since the error is mainly due to a change in weight of the aircraft, the

stability parameter $C_{L\alpha}$ will not change with the change in weight but is kept constant. Then only one equation, the forward acceleration equation $\dot{V}_{model}$ needs to be solved, and there is only one unknown, *m*. By using the update equation, *m* can be estimated easily.

In the above-mentioned and other similar cases where it is decided which parameter could change and that particular parameter is estimated, the performance of the parameter estimation technique depends on the individual performance of the update equations for each parameter. The technique produced excellent results for the separate estimation of all six parameters investigated in this work: C_{m0}, $C_{m\alpha}$, C_{mq}, $C_{m\delta e}$, $C_{L\alpha}$, and *m*.

4.2.2. Estimate More Than One Parameter at a Time

This routine can estimate more than one parameter at a time. As discussed previously, there are two equations to solve ($\dot{V}_{model}$ and $\ddot{\gamma}_{model}$) and six unknown parameters. To estimate more than two parameters at the same time, extra equations must be generated. This can be done by changing the flight condition of the aircraft. Changing the commanded velocity will change the dynamic pressure and ultimately the trim values of elevator deflection, thrust, and angle of attack for that particular flight condition. Using the new values of elevator deflection, thrust, and angle of attack, more data points can be generated for the forward and pitch acceleration (for the longitudinal mode). It was decided to change the flight condition dynamically using the specific flight maneuver described in section 4.2.1, and simultaneously update the four parameters mainly C_{m0}, $C_{m\alpha}$, $C_{L\alpha}$, and *m*.

To increase the convergence speed of the process, one parameter was updated for a short time period and then another parameter, was updated for the same period of time, and so on. Recall, the flight maneuver used was a sinusoidal variation flight speed with time with amplitude

of 10 knots and a time period of 125.6 seconds. C_{m0} and m were updated when the velocity increased, and $C_{m\alpha}$ and $C_{L\alpha}$ were updated when the velocity decreased. This process was continued in a sequential way until all the parameters reached a constant value. In an ideal situation, if only one of the four parameters (C_{m0}, $C_{m\alpha}$, $C_{L\alpha}$ and m) changed then a modeling error would occur in either velocity or pitch control. Looking at equations (4.1.1) and (4.1.2), C_{m0} and $C_{m\alpha}$ appear in the pitch acceleration equation, whereas $C_{L\alpha}$ and m appear in the forward acceleration equation. The routine gets activated and starts estimating C_{m0} and $C_{m\alpha}$ if the error in pitch acceleration goes above a threshold value. And if the error in forward acceleration goes above a threshold value, the routine starts estimating $C_{L\alpha}$ and m. But if the errors in both pitch and forward acceleration rise above their threshold values, the routine starts estimating all four parameters by their corresponding update equations. After going through a few periods of velocity variation, the algorithm should identify the changed parameter and settle into its new value. The number of periods of velocity variations ranging from 4 to 8 periods depends upon the magnitude of change in each parameter and the number of parameters estimated at the same time. Also, the other three parameters should return to their old values, which were changed initially due to the presence of the modeling error. This technique generated good results, fast convergence, and increased automation.

This technique was found to be limited to the four parameters (C_{m0}, $C_{m\alpha}$, $C_{L\alpha}$ and m) at a time and did not work well if the number of parameters was increased to five or six. This may be due to the increased complexity of the solution. Also, the technique did not work well with other combinations of parameters.

4.2.3. Identify the Changed Parameter and Then Update

This technique performs a test to identify the changed parameter and then updates that particular parameter. A major assumption in using the technique is that only one parameter could change at any given time during a flight condition. This assumption may be invalid in some cases of severe system degradation, but is valid for most of the other cases.

Over a nominal range of the flight envelope (e.g., cruise, take-off, landing), the aerodynamic and stability parameters are constant. In other words, these parameters can be used to model the response of the actual aircraft.

This implies that if a single parameter changes at a particular flight condition, and then that particular parameter is estimated by examining the modeling error, the updated value should be good for other flight conditions in that same range of the flight envelope. But if some irrelevant parameter (irrelevant to the cause of modeling error, or the one which did not change in the aircraft) is updated to drive the modeling error to zero at that particular flight condition, as the flight condition is changed the modeling error will increase significantly, resulting in a nonzero modeling error after changing the flight condition.

Consider the aircraft flying in trim. The following two equations must hold simultaneously:

$$C_{Ltrim} = \frac{W}{\bar{q}S} = C_{L_0} + C_{L_\alpha}\alpha + C_{L_{\delta e}}\delta_e \tag{4.2.1}$$

$$C_m = 0 = C_{m0} + C_{m\alpha}\alpha + C_{m\delta e}\delta_e \tag{4.2.2}$$

Writing these equations in Matrix form,

$$\begin{pmatrix} C_{L_\alpha} & C_{L_{\delta e}} \\ C_{m\alpha} & C_{m\delta e} \end{pmatrix} \begin{pmatrix} \alpha \\ \delta e \end{pmatrix} = \begin{pmatrix} \frac{W}{\bar{q}S} - C_{L_0} \\ -C_{m0} \end{pmatrix} \tag{4.2.3}$$

Or $$A.x = b$$

Where $\begin{pmatrix} C_{L_\alpha} & C_{L_{\delta e}} \\ C_{m\alpha} & C_{m\delta e} \end{pmatrix} = A$, $\begin{pmatrix} \alpha \\ \delta e \end{pmatrix} = x$, and $\begin{pmatrix} \frac{W}{\bar{q}S} - C_{L_0} \\ - C_{m0} \end{pmatrix} = b$

Now at flight condition 1:

$$A.x_1 = b_1 \tag{4.2.4}$$

If one of the parameters in matrix A changes in the actual aircraft, the neural network will provide compensation to the modeling error, which will cause the vector x_1 change to x_1', a new trim solution. If the changed parameter is identified and updated in the matrix A to obtain the trim solution, then equation (4.2.4) can be written as

$$A'.x_1' = b_1 \tag{4.2.5}$$

Here A' is the new matrix with the updated parameter.

After going to flight condition 2, the aircraft will still be trimmed since the matrix A' is a perfect solution and can trim the aircraft at all flight conditions, therefore equation (4.2.5) becomes,

$$A'.x_2 = b_2 \tag{4.2.6}$$

But, if the aircraft is trimmed at flight condition 1 by updating some "other" parameter, which actually did not change during flight, it will result in another matrix $\hat{A}$ that is different from A'. Then at flight condition 1, equation (4.2.5) becomes:

$$\hat{A}.x_1' = b_1 \tag{4.2.7}$$

It should be noted that the respective values of x_1', x_2, b_1 and b_2 will not change by updating the "other" parameter in the aircraft model, since these values come from the actual aircraft where the modeling error is compensated using neural networks.

At flight condition 2, equation (4.2.6) becomes,

$$\hat{A}.x_2 = b_2 \tag{4.2.8}$$

Examining equations (4.2.6) and (4.2.8), both cannot be true at the same time. Matrix A should have a unique solution, if A' trims the aircraft at all flight conditions, $\hat{A}$ cannot trim the aircraft at all flight conditions, and its known that $\hat{A} \neq A'$.

Matrix A' is a perfect solution for equation (4.2.3) since it uses updated value of the parameter that actually changed in the aircraft, so it can trim the aircraft at all flight conditions. But matrix $\hat{A}$ uses updated value of a parameter that did not change in the aircraft and it is tuned just to trim the aircraft at flight condition 1. So, matrix $\hat{A}$ is not a perfect solution of equation (4.2.3), and it cannot trim the aircraft at flight condition 2.

Finally it can be concluded that changing any other parameter, which did not change in actual aircraft, cannot trim the aircraft at all flight conditions or will result in non-zero modeling error at other flight conditions.

This concept can be used to perform a test to identify the changed parameter. When a nonzero modeling error is observed, one parameter is changed to drive the modeling error to zero at the same flight condition. Then the flight condition is changed and the modeling error is examined. If the modeling error is still near to zero and does not increase with the change in flight condition, then the changed parameter has been identified and is updated in the inverse controller. But if the modeling error increases significantly at the new flight condition, then this implies that the updated parameter did not change in the aircraft and is not related to the modeling error. Then the original value of that parameter is re-instituted and the same test is performed on another parameter. New parameters are shifted and tested in the same manner until one is found that drives the modeling error to zero at both flight conditions. The number of

forward velocities needed to identify the changed parameter depends upon the number of parameters considered at the same time. Using a greater number of velocities will result in more time required for the estimation process. To overcome this problem the process of identification of all the parameters is done at the same time by generating multiple copies of the estimation algorithm.

This test not only identifies the changed parameter but also updates it in the mean time. In other words, by the time the changed parameter is identified, it already has its new value. Therefore, it does not require extra time to estimate the parameters after identification. Once the changed parameter and its new value are identified, the newly estimated parameter is updated into the inverse controller with a predefined rate.

This technique was used to estimate C_{m0}, $C_{m\alpha}$, and $C_{m\delta e}$ at the same time. It produced good results and did not require any human assistance.

CHAPTER FIVE

5. RESULTS

Matlab/Simulink was used to simulate the controller, aircraft and parameter estimation system. Satisfactory results were obtained and are presented in three Sections 5.1, 5.2, and 5.3 sequentially according to the different approaches described in Section 4.2. In Sections 5.1 and 5.2 results are presented for first two approaches and use similar flight maneuver, whereas in Section 5.3 the results are given for the approach described in Section 4.2.3 and uses a different maneuver. These maneuvers are shown at the beginning of Sections 5.1 and 5.2.

5.1. Results for Estimation of Single Parameter at the Same Time

This section presents the results for the parameter estimation process described in Section 4.2.1. The flight maneuver selected for the estimation process commands a continuously changing velocity in a sinusoidal pattern with amplitude of 10 knots and a time period of 125.6 seconds, as shown in Figure 5.1. After trying different magnitudes and frequencies this maneuver was chosen to provide fast convergence with less control effort.

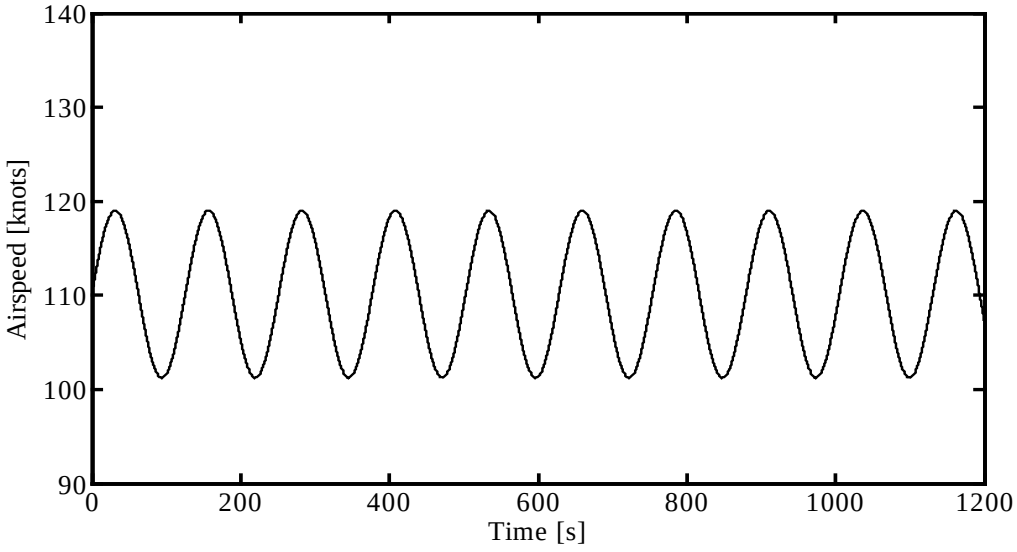


Figure 5.1. Flight maneuver velocity time history.

It should be noted that one can legitimately question the practicality of the chosen maneuver due to large and persistent variation in velocity. Also the technique was not tested for a slow and gradual change in the parameters.

To simulate a failure condition that could cause a change in a particular parameter, the original value of the parameter was deliberately changed in the Simulink model of the actual aircraft after 100 seconds. The neural network provided an immediate compensation to the modeling error introduced due to the change in one of the parameters in the actual aircraft.

The parameter estimation technique looks at the acceleration errors and gets activated when one of the errors rise above a particular threshold value. The threshold value of error in pitch acceleration was 10^{-8} rad/sec^2 and that for forward acceleration was 20^{-8} ft/sec^2. The technique estimates the parameter at its own speed, but the estimated parameter is updated in the inverse controller at a certain predefined rate, as described in Section 4.1.2.3. The update rates for different parameters are presented in Table 4.1. In this section the parameter that can change is assumed to be known and the routine is made to estimate that particular parameter.

As the routine estimates the changed parameter and updates it in the inverse controller, the modeling error gets eliminated and the neural network error compensation goes to zero. The ability of this technique to estimate the changed value of the parameter depends mainly on the rate and magnitude of the change of its original value in the actual aircraft. It also depends upon ability of the neural network to provide initial adaptation for the modeling error caused by the change in the actual parameter value. The flight maneuver may affect the maximum range of the change in parameters that can be estimated. The detailed analysis to test the ability of the technique in different flight maneuvers was not performed. Here the ability of the parameter estimation technique is rated by the maximum value of error that can be compensated before the

controller fails. The parameters were changed inside the actual aircraft and estimated using the estimation technique. The maximum range was set as the values of flight path angle or velocity beyond which they diverged indicating the failure of the controller. Maximum possible range of error in parameters that can be estimated is listed in Table 5.1.

Table 5.1. Possible Range Of Error In Parameters That Can Be Estimated.

Parameter	**Maximum range of error that can be estimated**
C_{m0}	- 400 % to +300 %
$C_{m\alpha}$	- 60 % to +200 %
$C_{m\delta e}$	- 70 % to +70 %
C_{mq}	- 100 % to +1000 %
$C_{L\alpha}$	- 60 % to +1000 %
m	- 70 % to +50 %

Though the parameter estimation technique was able to estimate the changed value of each parameter and the controller was able to control the aircraft for a much higher range of values, but the results presented below are for a nominal error of –50 % change in the original value of each parameter.

5.1.1. Estimation of a Change in C_{m0}

Figure 5.1.1 shows the time history of the C_{m0} values, and Figure 5.1.2 shows the time history of the pitch acceleration and the neural network error correction. Figure 5.1.1 shows that the value of C_{m0} in the actual aircraft was changed deliberately at a time step of 100 seconds. The routine became activated at the same time and started estimating the new value of the parameter. The estimated value followed the actual value very closely, but the updated value in

the inverse controller followed slowly with a limited rate of 0.00002 units per time step. It should be noted that the learning rate used for the estimation was 0.001. Examining the corresponding pitch acceleration plot (Figure 5.1.2), as the value of C_{m0} changed in the actual aircraft the neural network correction in pitch acceleration rose from zero to a value of nearly 0.2 rad/sec^2. This shows that the neural network provided immediate compensation to the modeling error caused due to the change of C_{m0} in the actual aircraft. Then, as the estimated value of C_{m0} was updated in the inverse controller, the neural network correction decreased to a very small value (see Figure 5.1.2(a)). This shows the elimination of the modeling error, as the value of C_{m0} was updated in the inverse controller.

5.1.2. Estimation of a Change in $C_{m\alpha}$

Figures 5.1.3 and 5.1.4 show the time history of the $C_{m\alpha}$ values and time history of the pitch acceleration and the neural network error correction.

Here the value of $C_{m\alpha}$ was changed in actual aircraft at a time step of 100 seconds, which can be seen in Figure 5.1.3. This resulted in a modeling error in pitch acceleration, which was higher than the threshold value of 10^{-8} rad/sec^2. The routine got activated and the new value of the parameter was estimated. The learning rate of 0.01 was used for the estimation of $C_{m\alpha}$ but the estimated value was updated slowly in the inverse controller with a limited rate of 0.001 units per time step. The pitch acceleration plot (Figure 5.1.4) shows the immediate compensation provided by the neural networks as the value of $C_{m\alpha}$ changed in the actual aircraft. As the value of $C_{m\alpha}$ became updated in the inverse controller, the neural network correction decreased to a very small value indicating the elimination of modeling error.

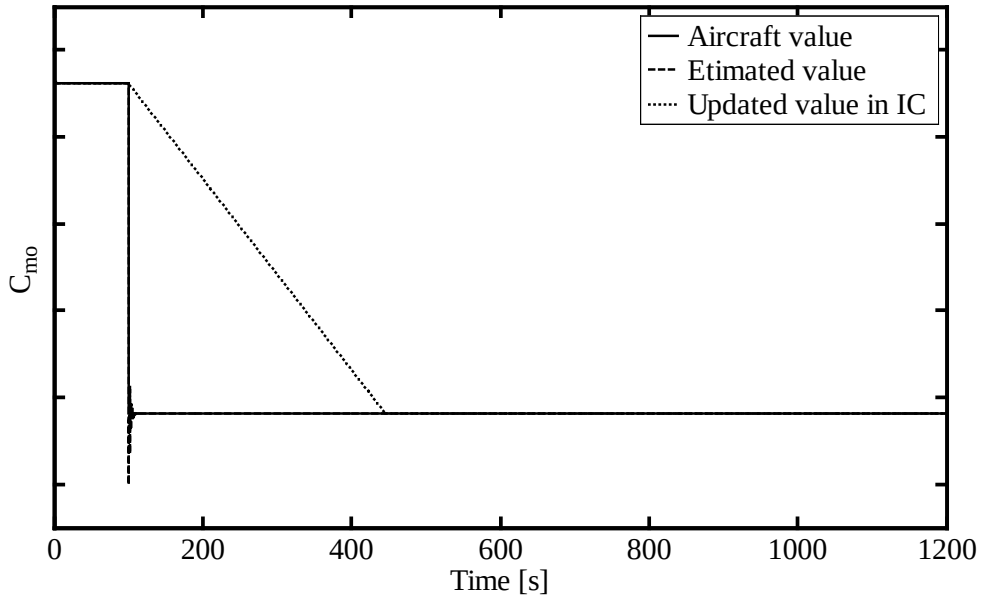


Figure 5.1.1(a). Time history of change in C_{m0} values.

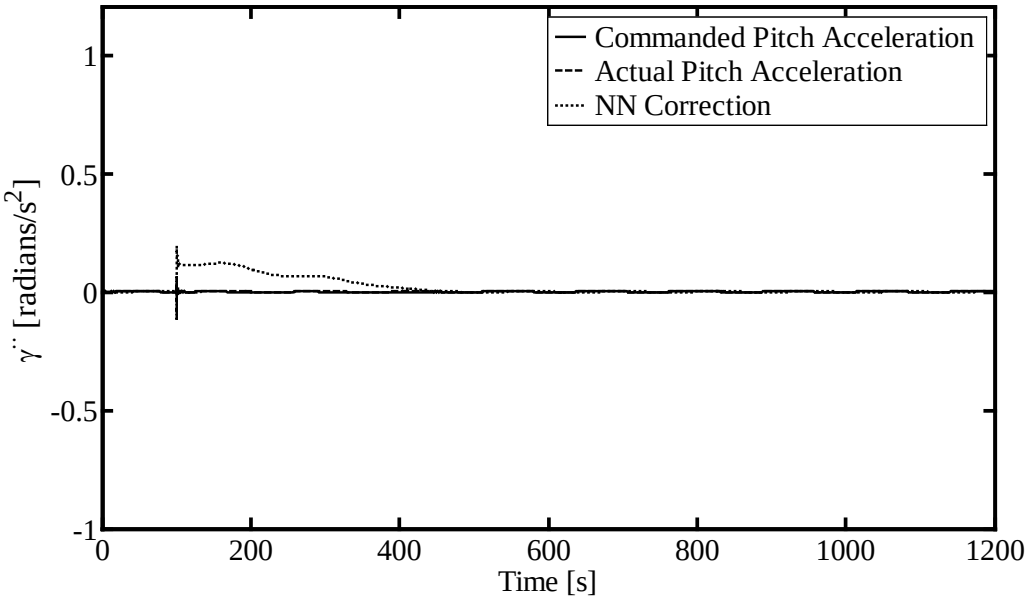


Figure 5.1.2(a). Time history of pitch acceleration and NN correction.

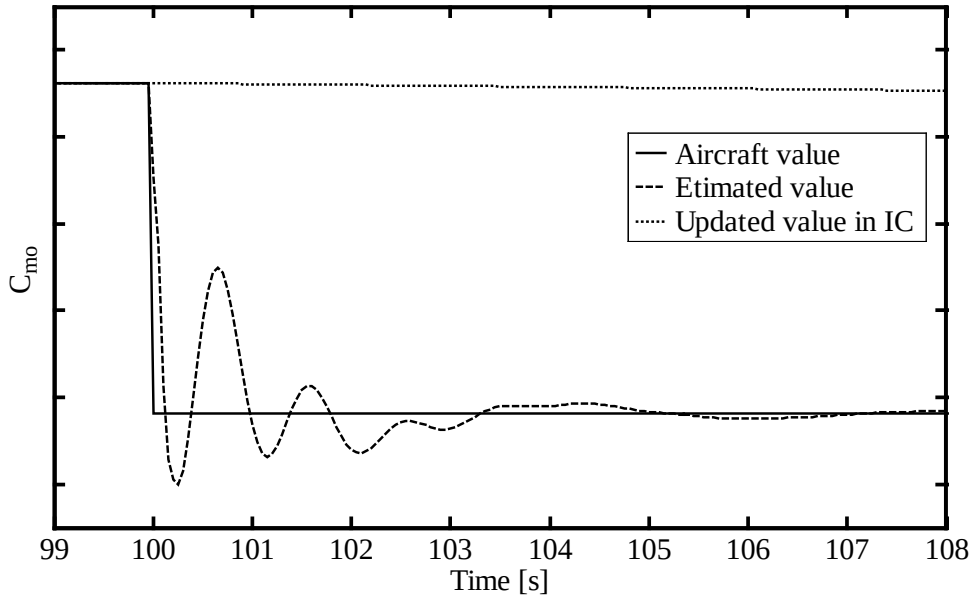


Figure 5.1.1(b). Zoomed in time history of change in C_{m0} values.

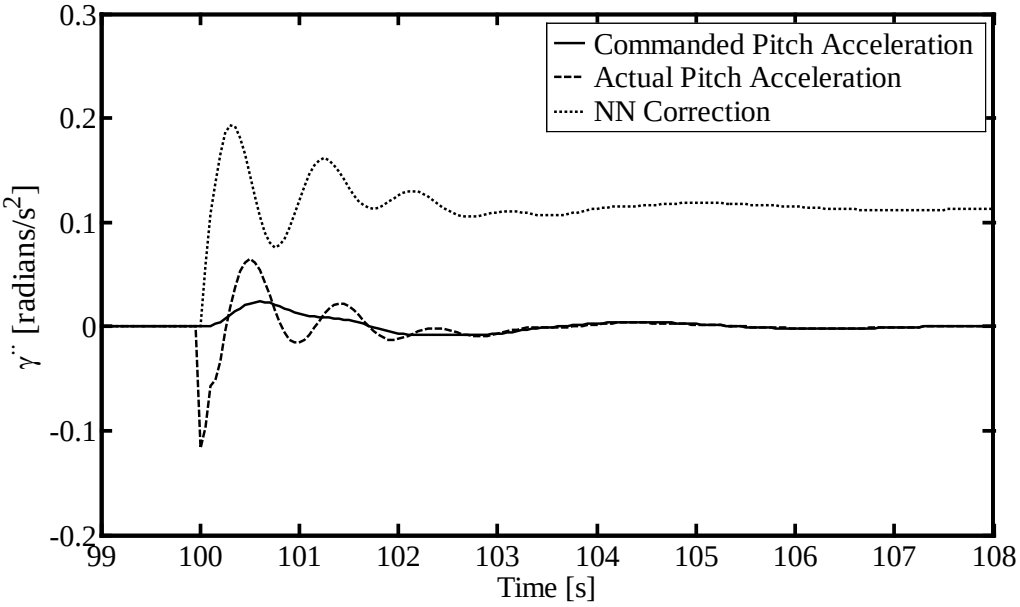


Figure 5.1.2(b). Zoomed in time history of pitch acceleration and NN correction.

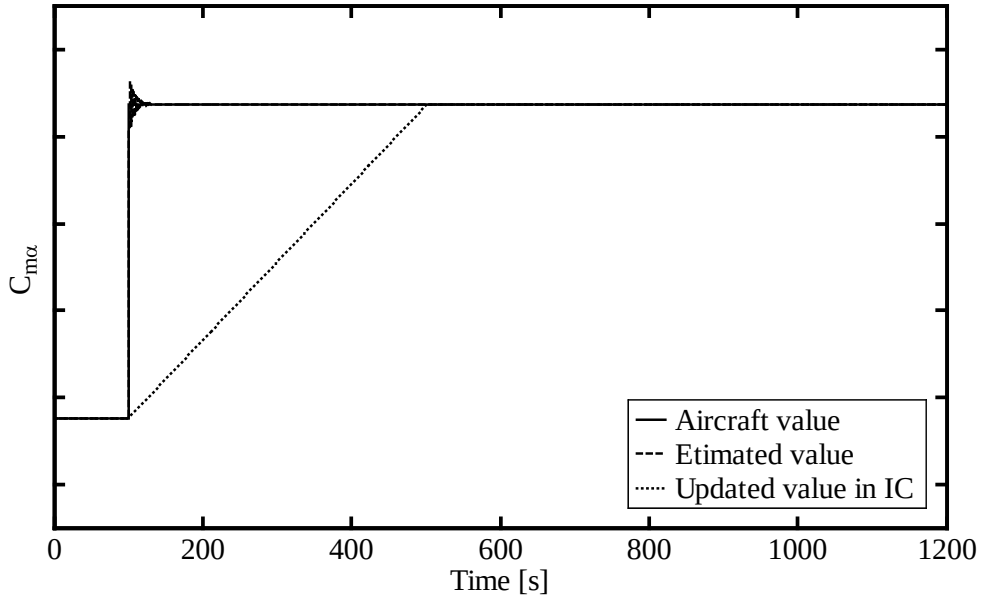


Figure 5.1.3(a). Time history of change in $C_{m\alpha}$ values.

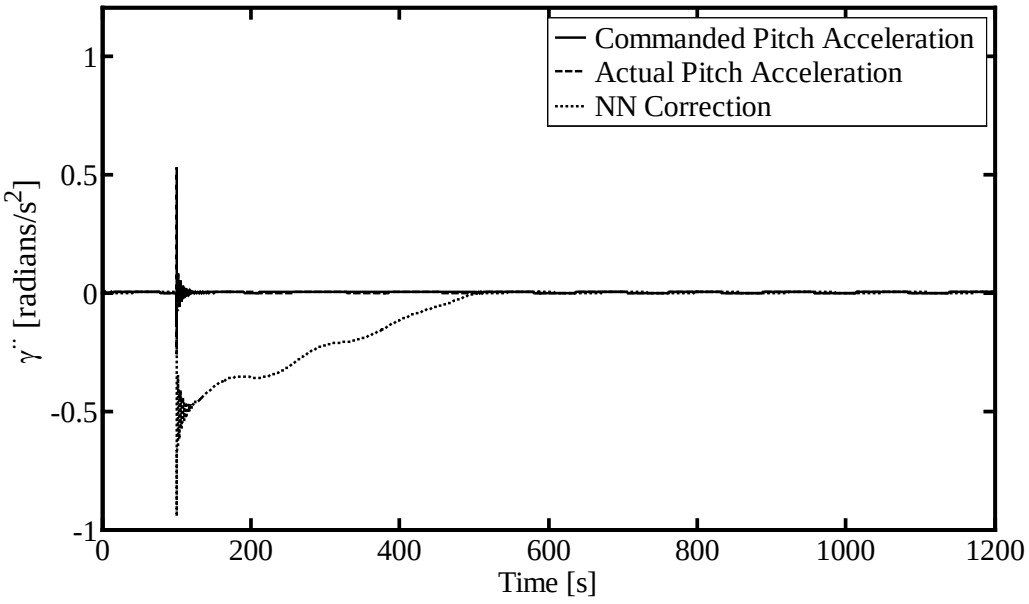


Figure 5.1.4(a). Time history of pitch acceleration and NN correction.

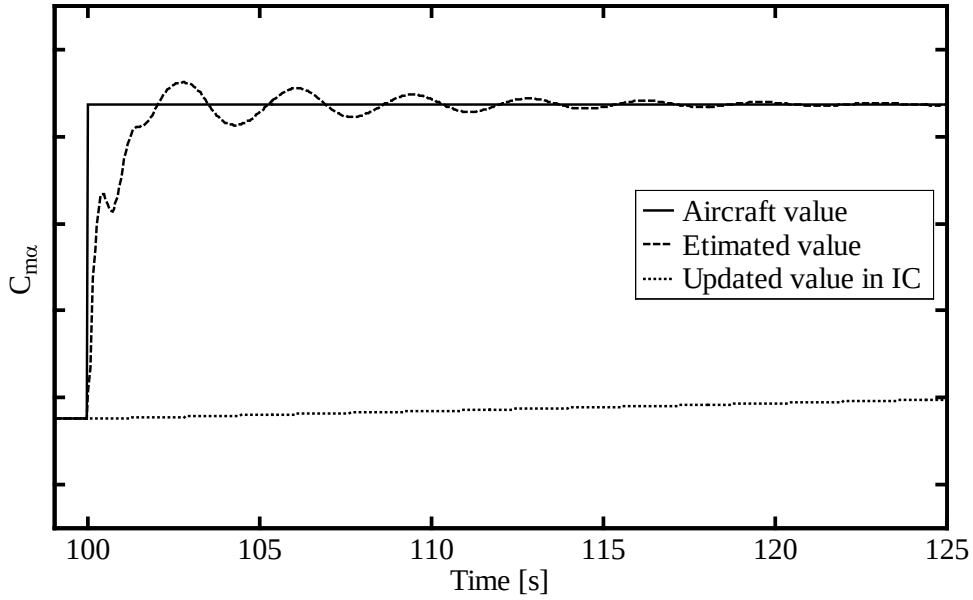


Figure 5.1.3(b). Zoomed in time history of change in $C_{m\alpha}$ values.

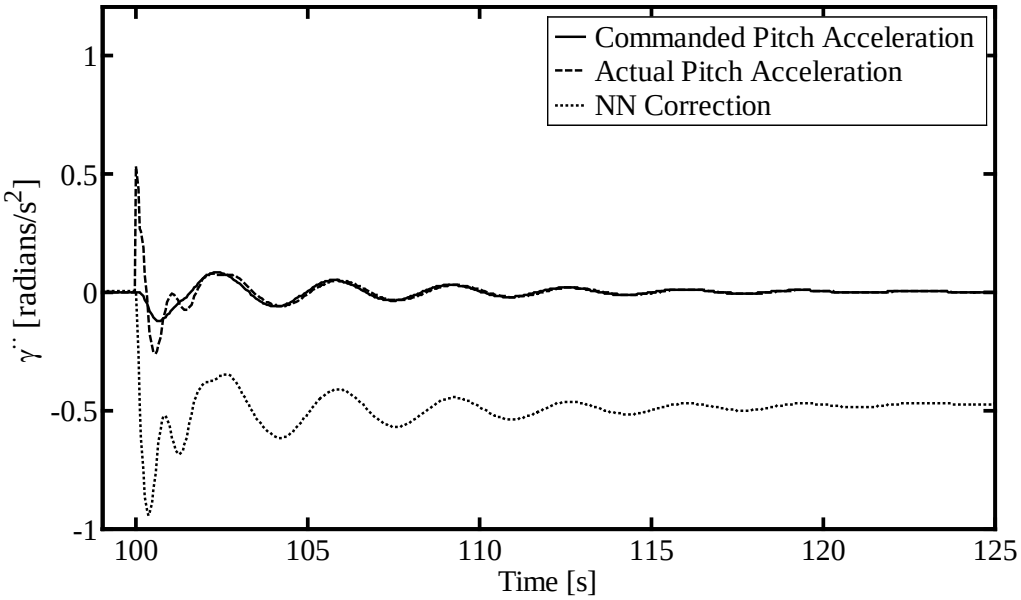


Figure 5.1.4(b). Zoomed in time history of pitch acceleration and NN correction.

5.1.3. Estimation of a Change in C_{mq}

Time history of the C_{mq} values is shown in Figure 5.1.5, and time history of the pitch acceleration and the neural network error correction is shown in Figure 5.1.6. At a time step of 100 seconds the value of C_{mq} was changed in the actual aircraft. The routine became activated and started estimating new value of C_{mq} since the modeling error in pitch acceleration exceeded the threshold value of 10^{-8} rad/sec^2 . The learning rate used for the estimation was 100000. Such high rate was established by examining the slow response of the estimation process. The estimated values were updated slowly into the inverse controller with a limited rate of 0.01 units per time step.

Examining the zoomed in time history for pitch acceleration (Figure 5.1.6(b)), it can be seen that as neural network adapted immediately to the modeling error, and the correction decreased to small value as the new value of C_{mq} was updated in the inverse controller.

5.1.4. Estimation of a Change in $C_{m\delta e}$

Here Figure 5.1.7 shows the time history of the $C_{m\delta e}$ values, and Figure 5.1.8 shows the time history of the pitch acceleration and the neural network error correction.

As the value of $C_{m\delta e}$ was changed deliberately in the actual aircraft at a time step of 100 seconds, the routine started estimating its new value. Here the learning rate used was 0.1 and the estimated value was updated slowly in the inverse controller with a rate of 0.001 units per time step.

Figure 5.1.8 shows the adaptation provided by the neural network as the value of $C_{m\delta e}$ changed in the actual aircraft. From the same figure it can also be seen that as the new value of

$C_{m\delta e}$ was updated inside the inverse controller the neural network correction diminished to a very small value.

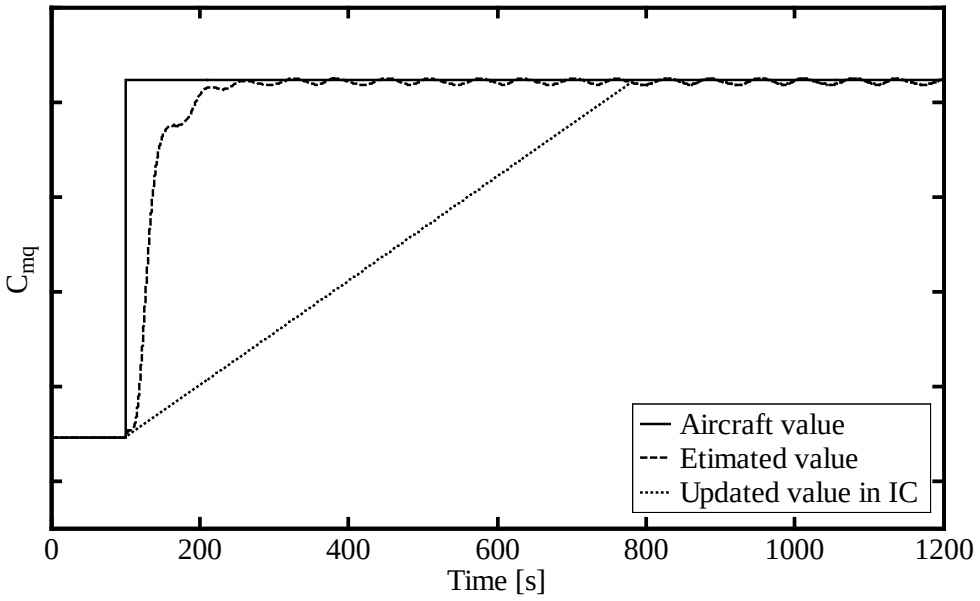


Figure 5.1.5. Time history of change in C_{mq} values.

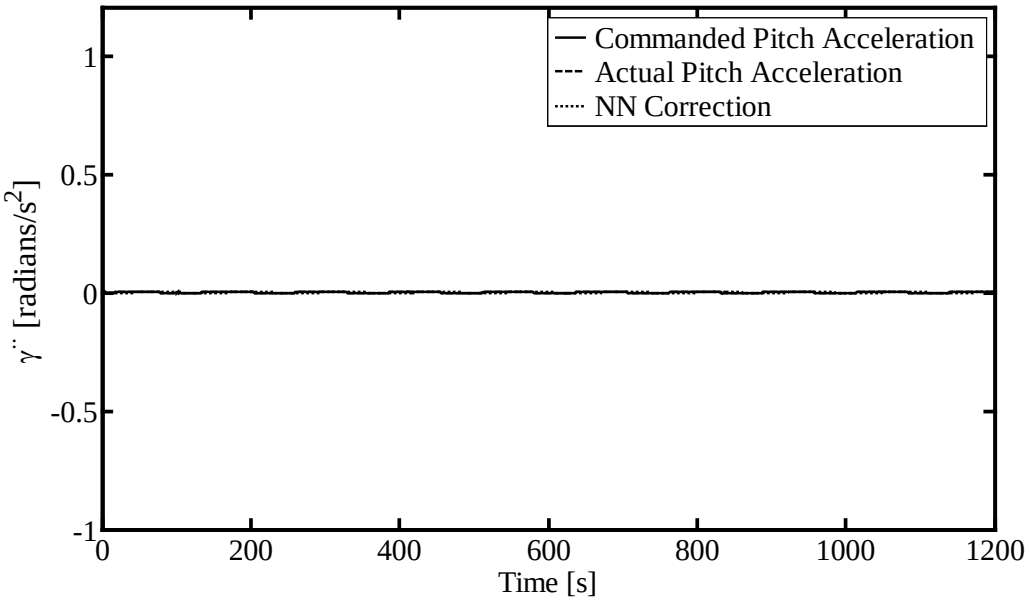


Figure 5.1.6(a). Time history of pitch acceleration and NN correction.

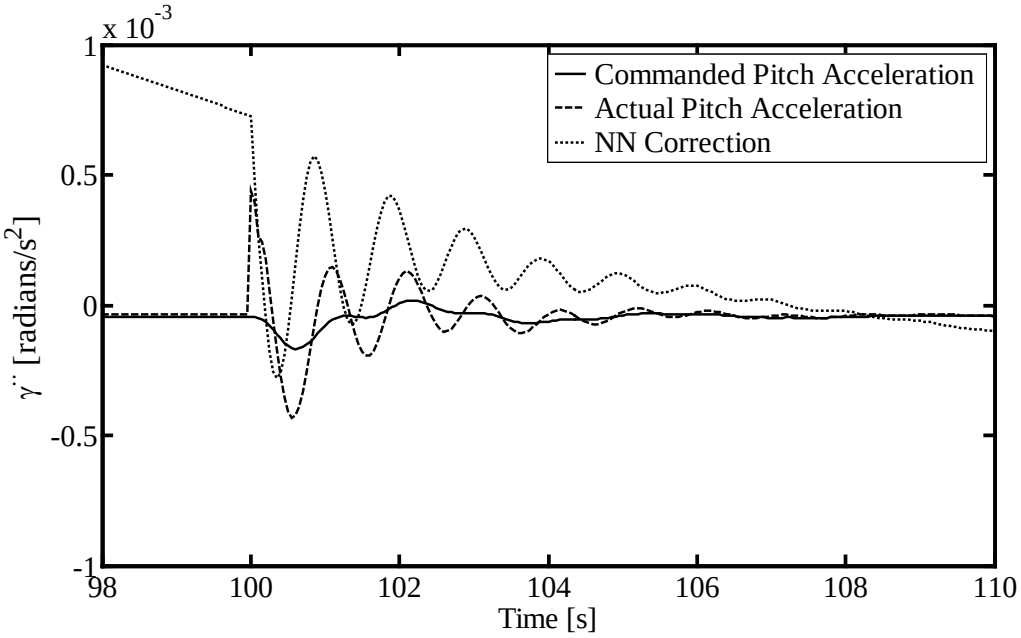


Figure 5.1.6(b). Zoomed in time history of pitch acceleration and NN correction.

5.1.5. Estimation of a Change in $C_{L\alpha}$

Figure 5.1.9 shows that the value of $C_{L\alpha}$ in actual aircraft was changed deliberately at time step of 100 seconds, which resulted in a modeling error of about –4 ft/s^2. As a reminder the threshold value used for detecting modeling error in forward acceleration was kept as 20^{-8} ft/sec^2. The routine got activated at the same time and started estimating the new value of $C_{L\alpha}$. The estimation was done with a learning rate of 0.5, and the new value was updated in the inverse controller with a limited rate of 0.004 units per time step.

Forward acceleration plot (Figure 5.1.10), shows the adaptation provided by neural network as the value of $C_{L\alpha}$ changed in the actual aircraft. It also shows that as the new value of $C_{L\alpha}$ was updated inside the inverse controller the neural network correction approached zero or a very small value.

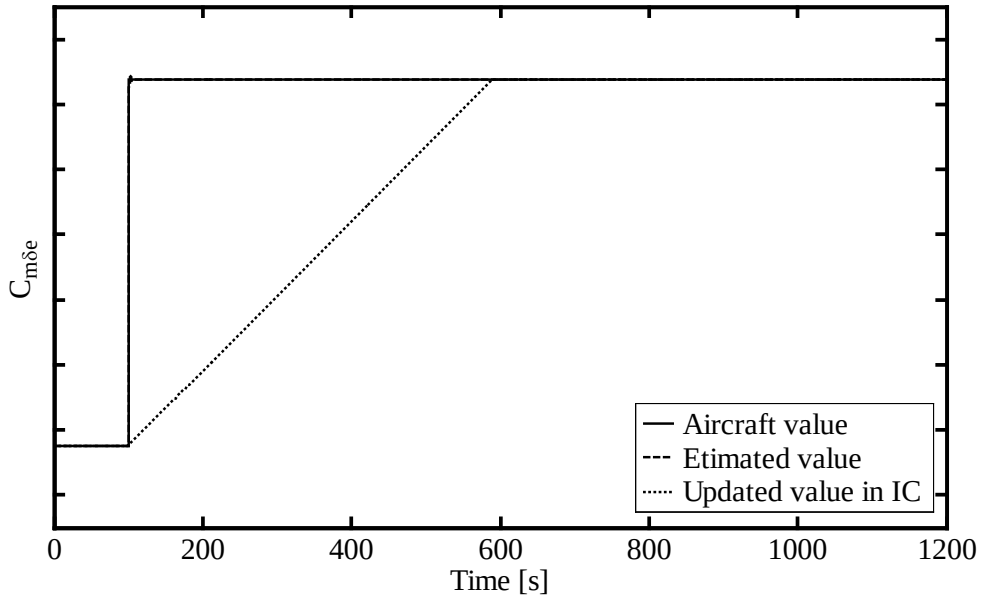


Figure 5.1.7(a). Time history of change in $C_{m\delta e}$ values.

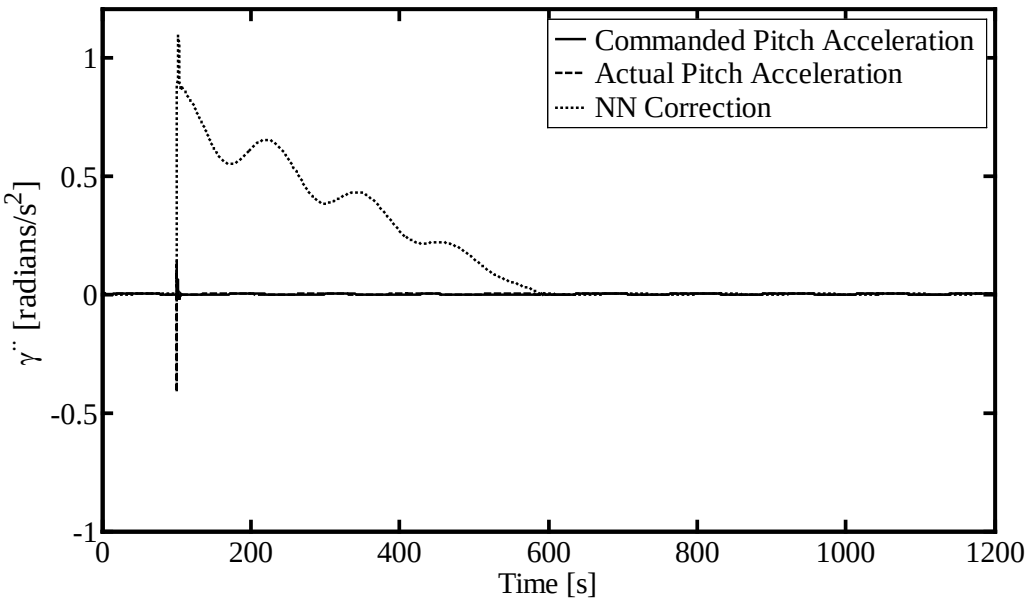


Figure 5.1.8(a). Time history of pitch acceleration and NN correction.

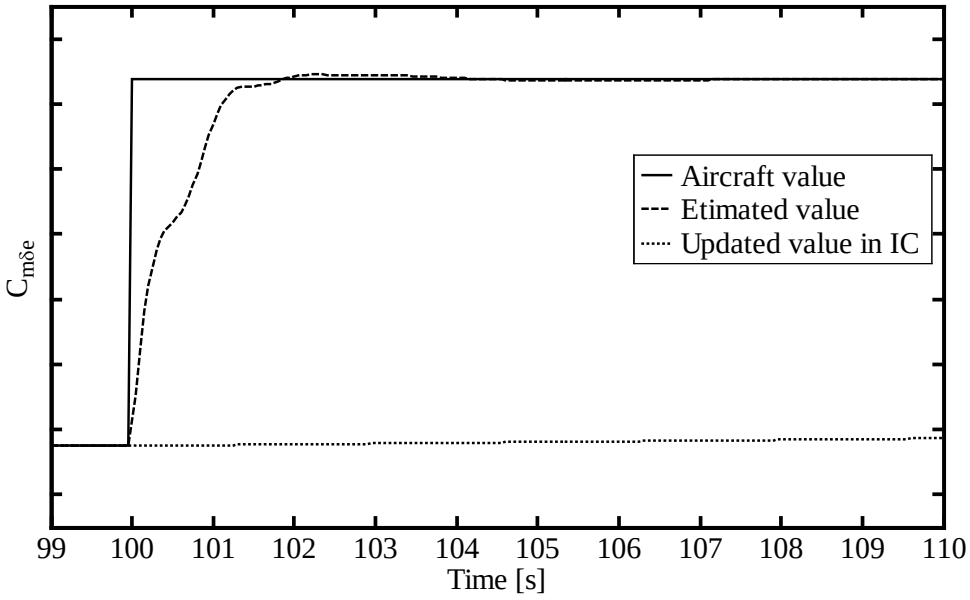


Figure 5.1.7(b). Zoomed time history of change in $C_{m\delta e}$ values.

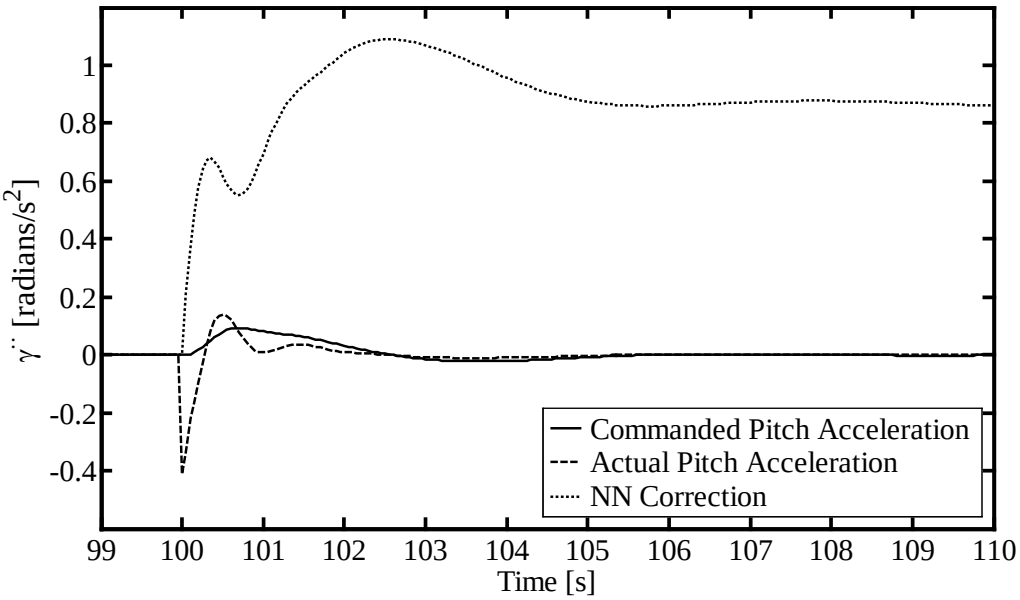


Figure 5.1.8(b). Zoomed time history of pitch acceleration and NN correction.

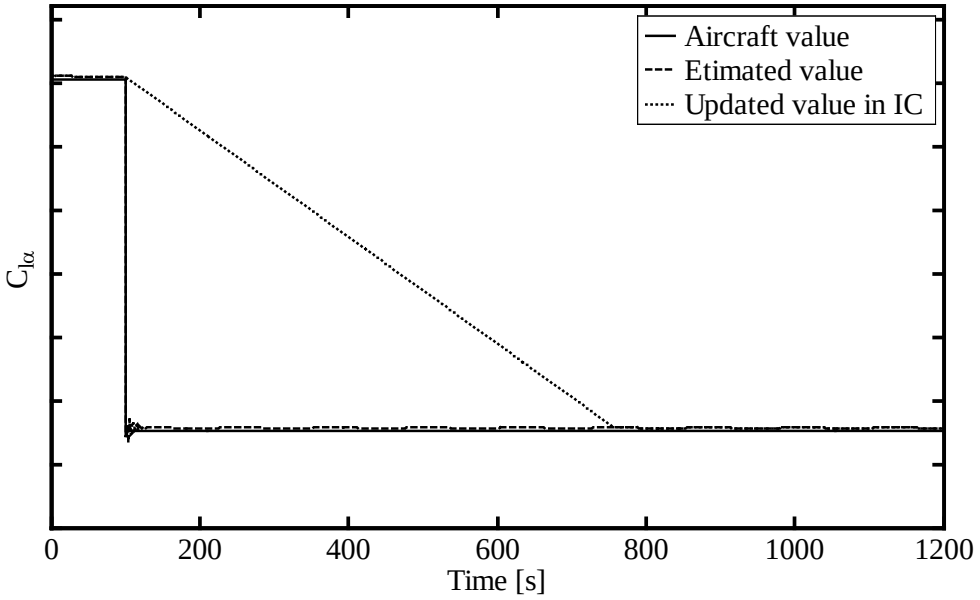


Figure 5.1.9(a). Time history of change in $C_{L\alpha}$ values.

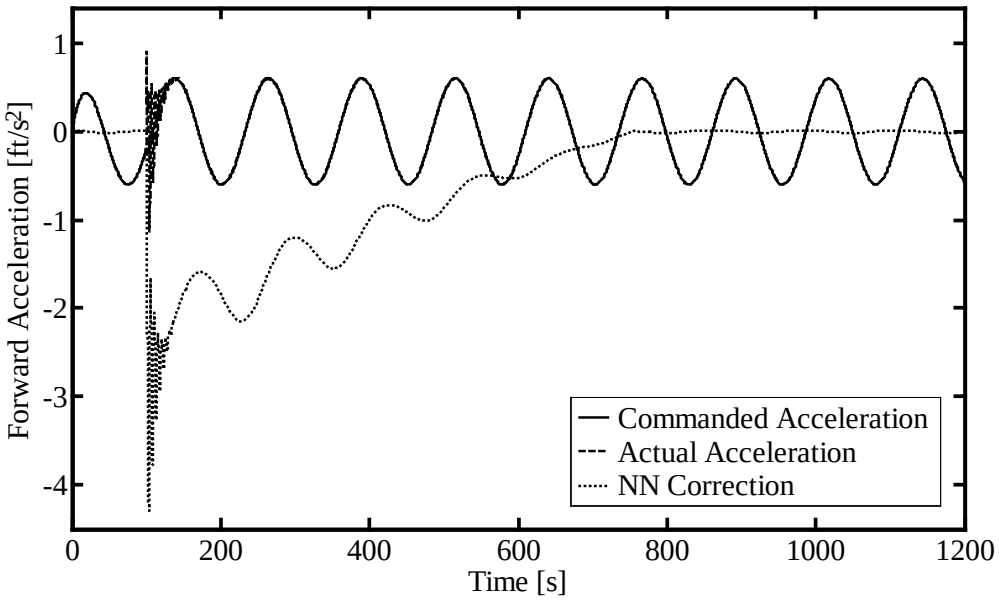


Figure 5.1.10(a). Time history of forward acceleration and NN correction.

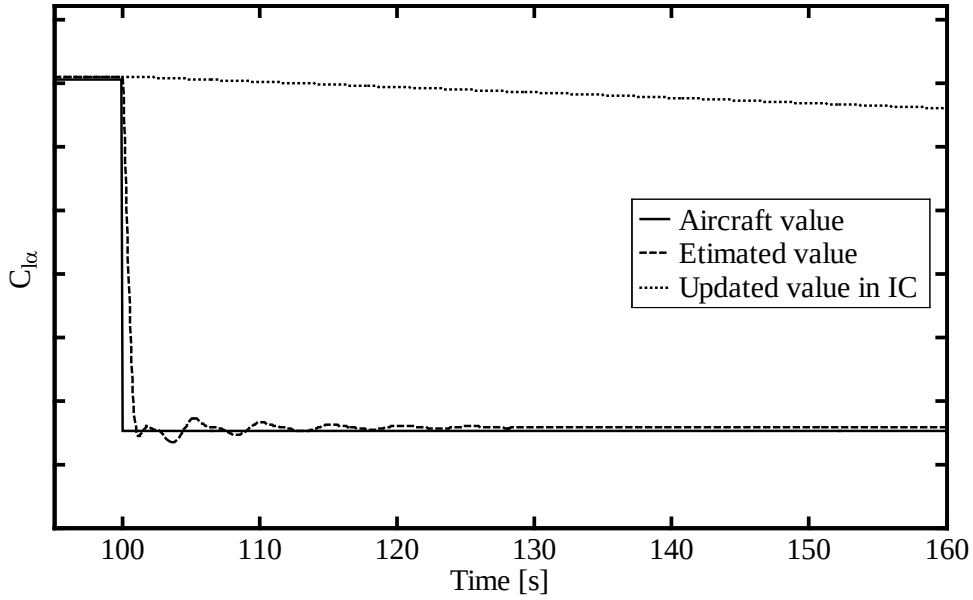


Figure 5.1.9(b). Zoomed time history of change in $C_{L\alpha}$ values.

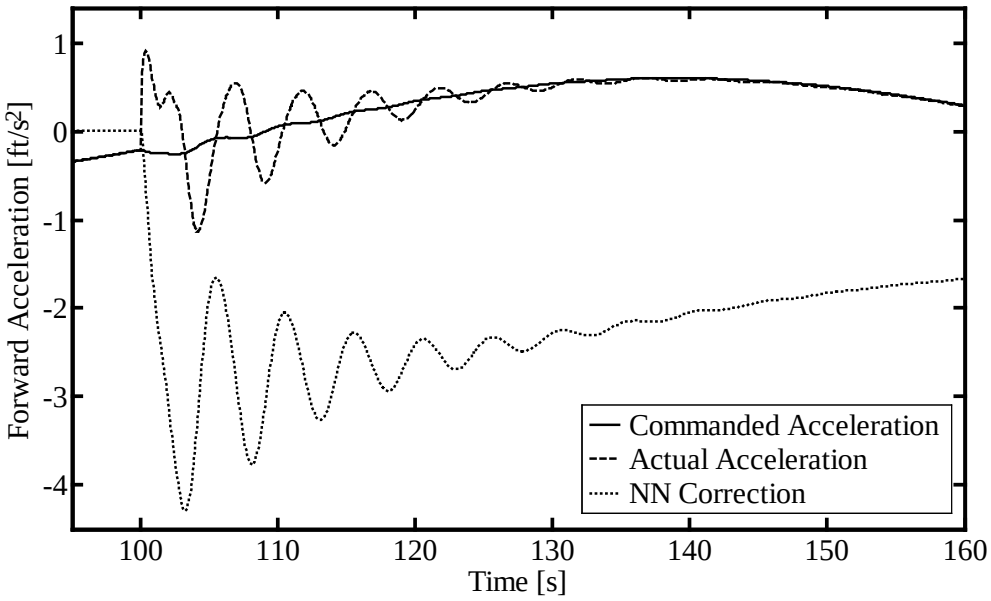


Figure 5.1.10(b). Zoomed time history of pitch acceleration and NN correction.

5.1.6. Estimation of a Change in Mass

The value of mass was reduced to half of its value in the actual aircraft at a time step of 100 seconds. Figure 5.1.11 shows the time history of mass values. It caused a modeling error that activated the routine and the new value of mass was estimated at a learning rate of 50. Then the new value of mass was updated in the inverse controller at a limited rate of 0.3 units per time step.

The neural network correction plot (Figure 5.1.12) shows the correction provided by the neural network as the mass of actual aircraft decreased at a time step of 100 seconds. The Figure 5.1.12 also shows that as the new value of mass was updated inside the inverse controller the correction provided by neural network went down to zero.

5.2. Simultaneous Estimation of Multiple Parameters

In this routine the system estimates four parameters, C_{m0}, $C_{m\alpha}$, $C_{L\alpha}$, and mass at the same time. For the purpose of increasing the number of equations and decoupling the effect of each parameter on the corresponding acceleration error, a flight maneuver with constantly varying velocity was used. Also, to increase the convergence rate of parameter estimation, each parameter was updated at a specific part of the maneuver. C_{m0} and mass were estimated when the velocity increased, and $C_{m\alpha}$ and $C_{L\alpha}$ were updated when velocity decreased.

The flight maneuver selected for the estimation process is similar to that used in previous routine, which used a continuously changing velocity in a sinusoidal pattern with amplitude of 10 knots and a frequency of 0.05 radians/sec. Figure 5.2 is a repeated Figure 5.1 showing the velocity time history for the flight maneuver common for the estimation of all four parameters.

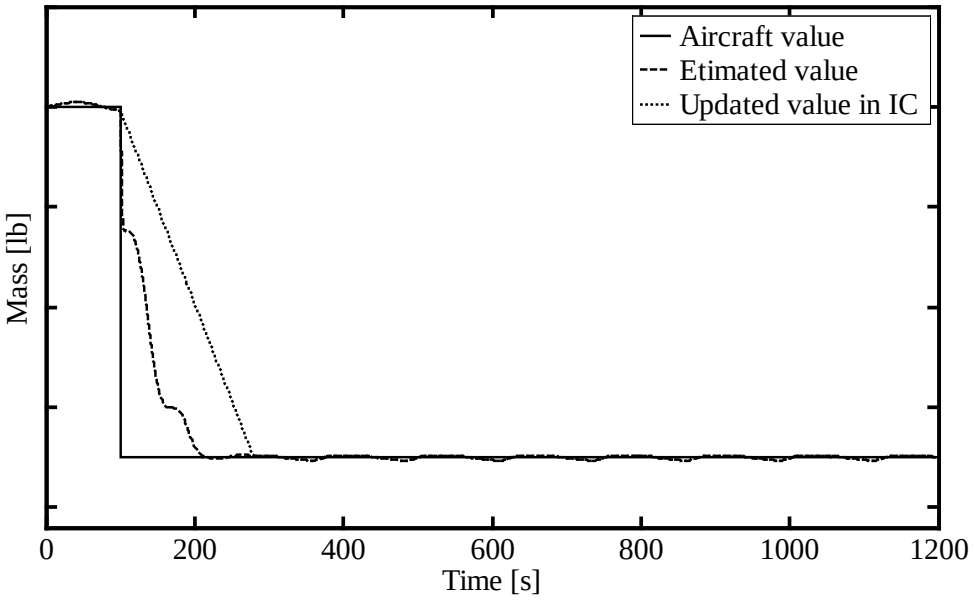


Figure 5.1.11. Time history of change in Mass values.

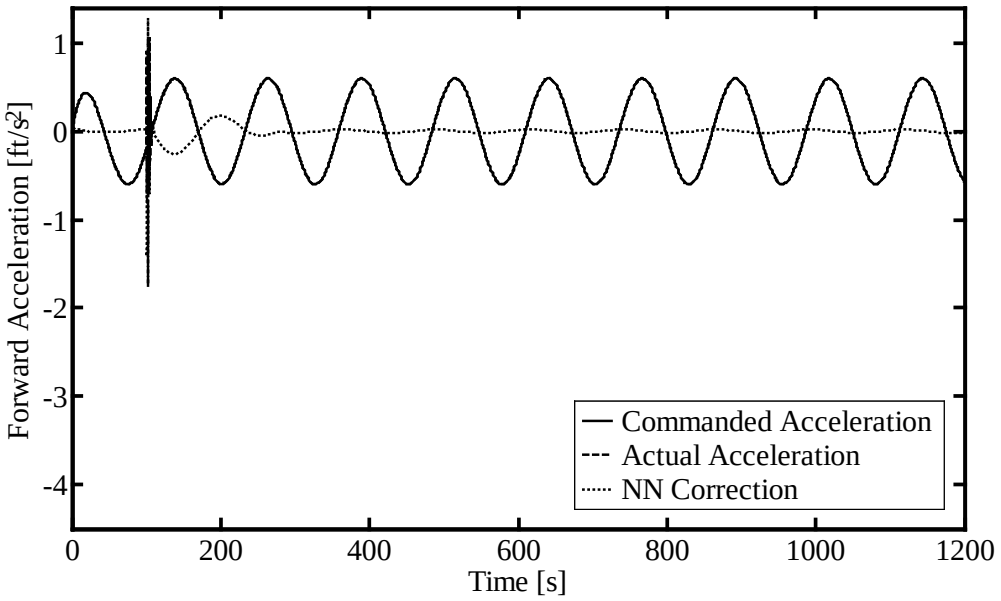


Figure 5.1.12(a). Time history of pitch acceleration and NN correction.

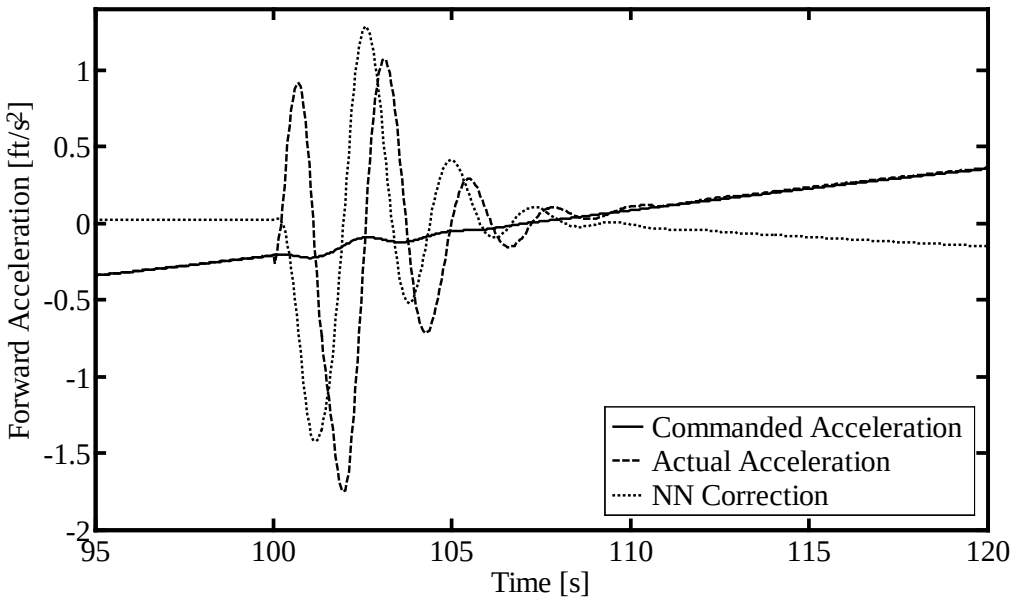


Figure 5.1.12(b). Zoomed time history of pitch acceleration and NN correction.

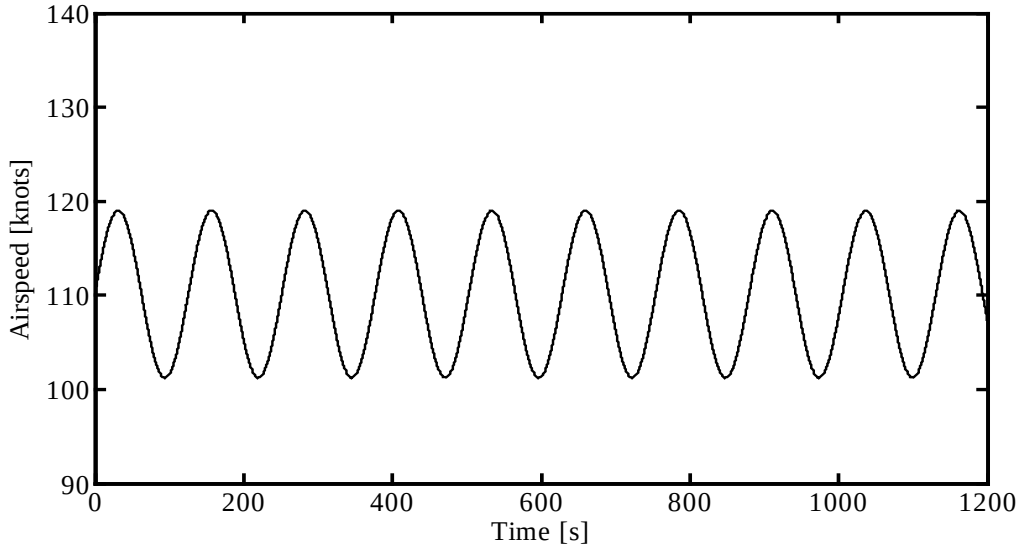


Figure 5.2. Velocity time history.

5.2.1. Estimation of a Change in C_{m0}

Figures 5.2.1 and 5.2.2 show the time history of C_{m0} and $C_{m\alpha}$ values, whereas Figure 5.2.3 shows the time history of the pitch acceleration and the neural network error correction.

Figure 5.2.1 shows that the value of C_{m0} in the actual aircraft was decreased deliberately to about 50% from its original value at time step of 100 seconds. This gave rise to a modeling error in pitch acceleration. As the modeling error went above a threshold value of 10^{-8} rad/sec^2 the routine got activated and started estimating the new values of C_{m0}, $C_{m\alpha}$, $C_{L\alpha}$, and mass. The modeling error in the pitch acceleration could mainly be caused due to change in C_{m0} or $C_{m\alpha}$. From the plots for C_{m0} and $C_{m\alpha}$ (Figure 5.2.1 and 5.2.2) it can be seen that the estimated value of C_{m0} settled to its newly changed value, whereas the estimated value of $C_{m\alpha}$ settled to its original value after a time of approximately 1000 to 1200 seconds, which may be reasonable since the technique is expected to provide slow adaptation. The plots for $C_{L\alpha}$ and Mass are not shown, as the variations in their values were negligible. The corresponding plot of pitch acceleration (Figure 5.2.3) shows the compensation provided by the neural network starting from a time step of 100 seconds. The compensation provided by the neural networks goes to zero (at about 200 seconds) much before the new value of C_{m0} is estimated. The adaptation provided by the changing value of $C_{m\alpha}$ eliminated the modeling error much before the parameters settled to their correct values. But as the flight condition changed with time (Figure 5.2), the estimated values of the parameters settled to their correct values. The dotted lines in the Figures 5.2.1 and 5.2.2 show the updated values of the parameters in the inverse controller. The update rates were mentioned in Table 4.1.

5.2.2. Estimation of a Change in $C_{m\alpha}$

Time history of C_{m0} and $C_{m\alpha}$ values are shown in Figures 5.2.4 and 5.2.5, and time history of pitch acceleration and the neural network correction is shown in Figure 5.2.6. The value of $C_{m\alpha}$ was decreased to about 50% of its original value at a time step on 100 seconds.

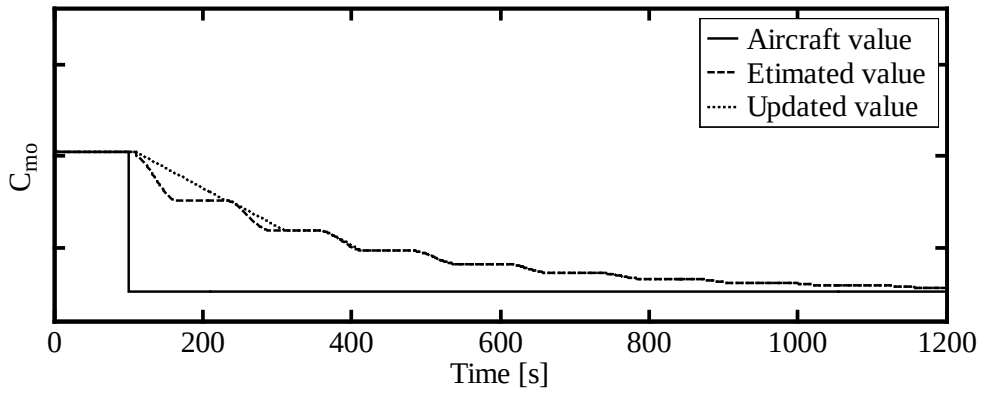


Figure 5.2.1. C_{m0} update time history.

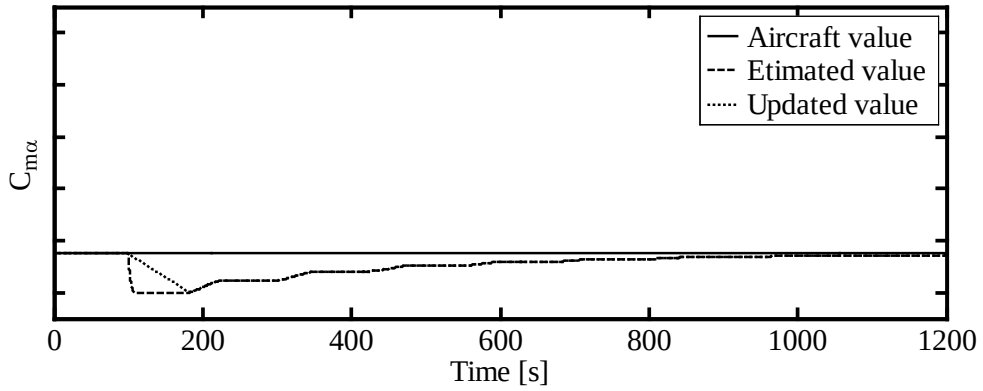


Figure 5.2.2. $C_{m\alpha}$ update time history.

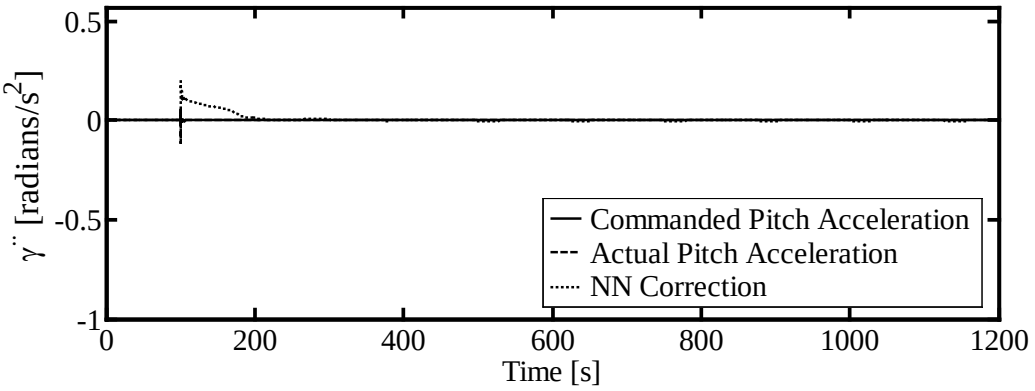


Figure 5.2.3. NN correction time history.

The routine started estimating the new values of C_{m0}, $C_{m\alpha}$, $C_{L\alpha}$, and mass, as the modeling error in pitch acceleration went above the value of 10^{-8} rad/sec^2. The modeling error in the pitch acceleration can mainly be caused due to change in C_{m0} or $C_{m\alpha}$. The plots for C_{m0} and $C_{m\alpha}$ (Figures 5.2.4 and 5.2.5) shows that the estimated value of $C_{m\alpha}$ settled to its newly changed value, whereas the estimated value of C_{m0} settled to its original value after a time of approximately 1200 seconds. Since the variations in the values of $C_{L\alpha}$ and Mass are negligible, their plots are shown. The C_{m0} estimation equation uses a learning rate of 0.001 and $C_{m\alpha}$ estimation equation uses a learning rate of 0.01. The pitch acceleration plot (Figure 5.2.6) shows the compensation provided by the neural network starting from time step of 100 seconds, it also shows that the compensation went down to zero (at about 500 seconds) much before the new value of $C_{m\alpha}$ was estimated. The reason for early elimination of the modeling error is the adaptation provided by the changing value of C_{m0} which eliminated the modeling error much before the parameters settled to their correct values. As the flight condition changed the parameters settled to there correct values. The dotted lines in the Figures 5.2.4 and 5.2.5 show the updated values of the parameters in the inverse controller.

5.2.3. Estimation of a Change in $C_{L\alpha}$

The value of $C_{L\alpha}$ is changed in the actual aircraft to half of its value at about 100 seconds, it can be seen in Figures 5.2.7. Figure 5.2.8 show the time history of Mass values, and Figure 5.2.9 shows the time history of the forward acceleration and the neural network error correction. The changed value of $C_{L\alpha}$ gave rise to a modeling error in forward acceleration. As the modeling error in the forward acceleration can mainly be caused due to change in $C_{L\alpha}$ and m.

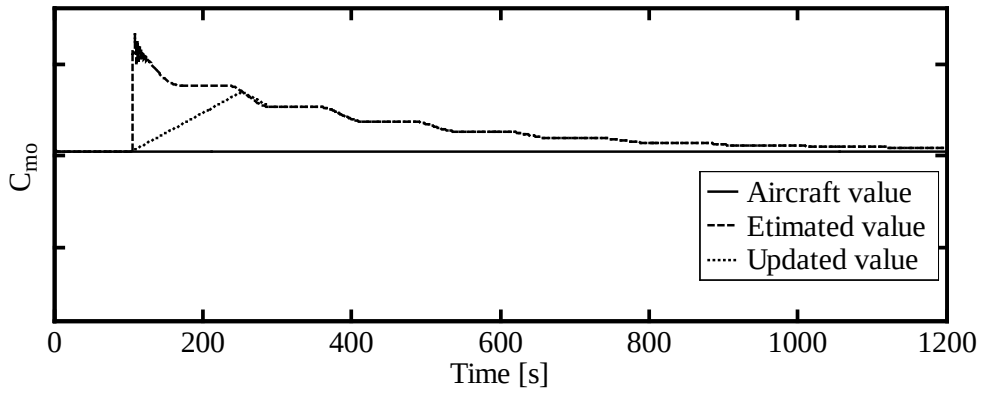


Figure 5.2.4. C_{m0} update time history

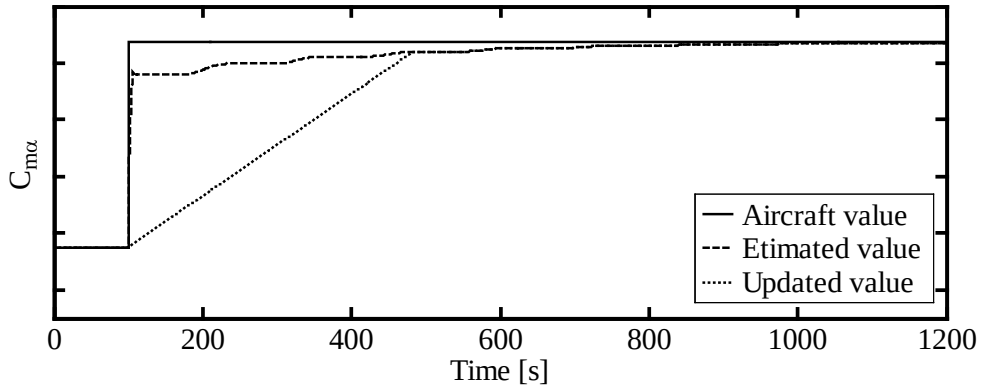


Figure 5.2.5. $C_{m\alpha}$ update time history

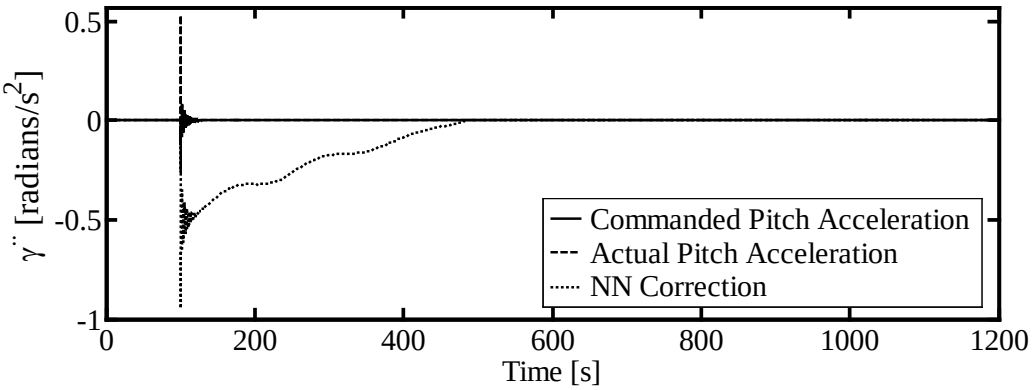


Figure 5.2.6. NN correction time history.

The Figures 5.2.7 and 5.2.8 show that $C_{L\alpha}$ settled to its new value and mass settled to its original value at about 780 seconds. The estimation equations used a learning rate of 50 for mass

and 0.5 for $C_{L\alpha}$. The neural network provided immediate compensation starting from time step of 100 seconds, which went down to zero at about 750 seconds as the new value of $C_{L\alpha}$ was estimated. It can also be seen that the adaptation provided by the changing value of mass eliminated the modeling error before the parameters settled to their correct values. The estimated value of mass during the estimation process may not be small but that did not affect the controller, since its updated value in the inverse controller was very small. As the flight condition changed with time (Figure 5.2), the estimated values of the parameters settled to their correct values. The oscillatory response of the commanded and actual accelerations in the neural network error correction plot was due to the change in commanded airspeed.

5.2.4. Estimation of a Change in Mass

As the value of mass in actual aircraft was decreased to about 50% of its original value (Figure 5.2.10) at time step of 100 seconds the modeling error in forward acceleration went above the threshold value of 20^{-8} ft/sec^2. Though the drop in mass of 50% is not practical, it is simulated to demonstrate the ability of the parameter estimation technique. The routine got activated and started estimating the new value of C_{m0}, $C_{m\alpha}$, $C_{L\alpha}$, and mass. The change in the values of $C_{L\alpha}$ and mass can cause a modeling error in the forward acceleration. The learning rate used for the estimation of mass was 50 and that for the estimation of $C_{L\alpha}$ was 0.5. Figures 5.2.10 and 5.2.11 show the estimated value of mass that settled to its newly changed value, and the estimated value of $C_{L\alpha}$ settled to its original value after a time of approximately 600 seconds. Compensation provided by the neural network can be seen from Figure 5.2.12, which rises at time step of 100 seconds and then goes down to zero at about 350 seconds as the new

value of mass gets updated inside the inverse controller. The doted lines in Figures 5.2.10 and 5.2.11 show parameters updated inside the inverse controller.

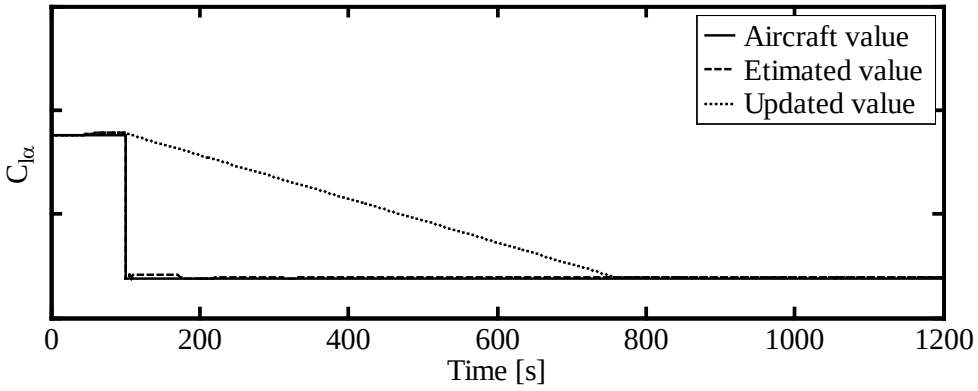


Figure 5.2.7. $C_{L\alpha}$ update time history

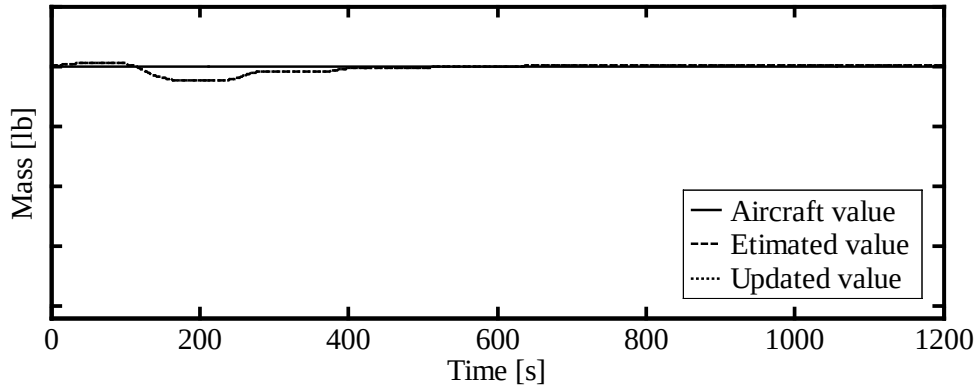


Figure 5.2.8. Mass update time history

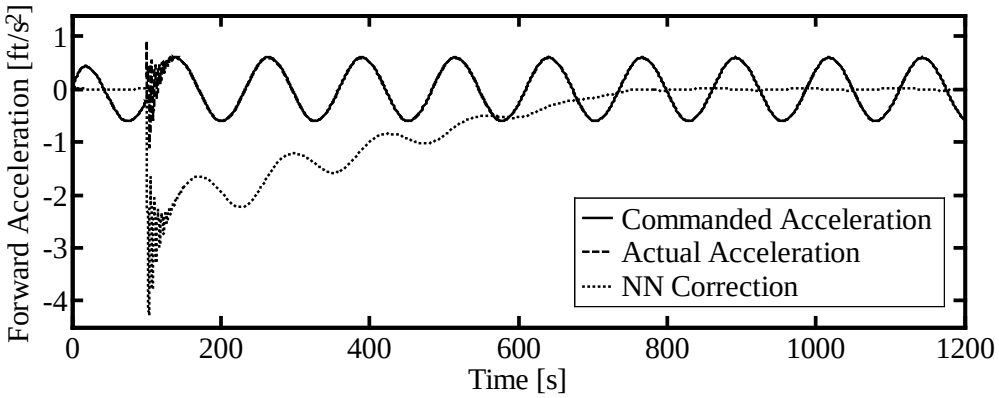


Figure 5.2.9. NN correction time history.

Again due to the continuous change in the commanded airspeed an oscillatory response of the commanded and actual accelerations is observed in the neural network error correction plot.

5.2.5. Estimation of a Change in C_{m0}, $C_{m\alpha}$, $C_{L\alpha}$, and mass

Figures 5.2.13, 5.2.14, 5.2.16 and 5.2.17 show the time history of C_{m0}, $C_{m\alpha}$, $C_{L\alpha}$, and mass values, whereas Figures 5.2.15 and 5.2.18 show the time history of the pitch acceleration and forward acceleration along with their corresponding neural network error corrections. The values of C_{m0}, $C_{m\alpha}$, $C_{L\alpha}$, and mass were decreased deliberately to about 50% from their original values at time step of 100 seconds. This gave rise to a modeling error in both pitch and forward acceleration. As the modeling error in pitch acceleration exceeded a threshold value of 10^{-8} rad/sec^2 or modeling error in forward acceleration above 20^{-8} ft/sec^2 the routine got activated and started estimating the new value of C_{m0}, $C_{m\alpha}$, $C_{L\alpha}$, and mass. Figures 5.2.13, 5.2.14, 5.2.16 and 5.2.17 show that approximately between 400 and 1000 seconds the estimated values of C_{m0}, $C_{m\alpha}$, $C_{L\alpha}$, and mass settled to their newly changed values. The learning rates used for the estimation of all four parameters are specified in section 4.1.2.3. Figures 5.2.15 and 5.2.18 show that the compensation provided by the neural network starts from a time step of 100 seconds and goes to zero (at about 480 seconds for pitch acceleration and about 600 seconds for forward acceleration) before the new value of C_{m0}, $C_{m\alpha}$, $C_{L\alpha}$, and mass were estimated. The adaptation provided by the changing values of C_{m0}, $C_{m\alpha}$, $C_{L\alpha}$, and mass eliminated the modeling error before the parameters settled to their correct values. But as the flight condition changed with time (Figure 5.2), the estimated values of the parameters settled to their correct values.

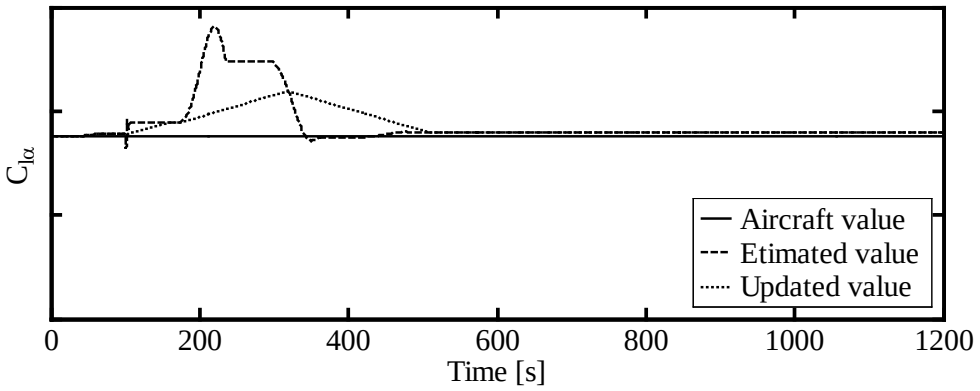


Figure 5.2.10. $C_{L\alpha}$ update time history

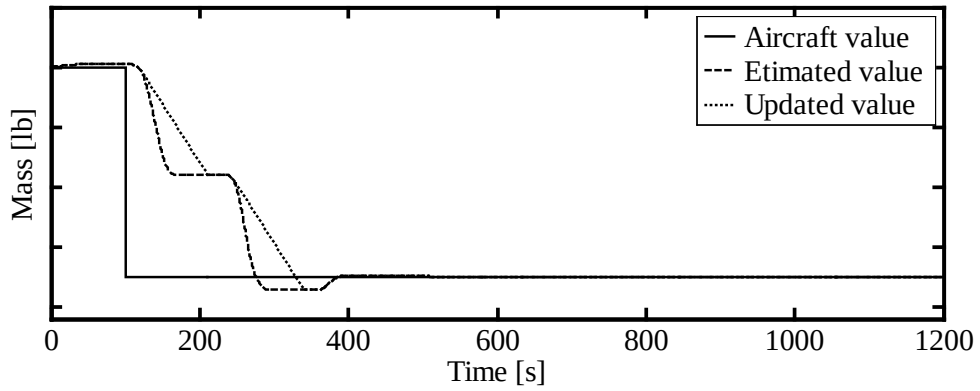


Figure 5.2.11. Mass update time history

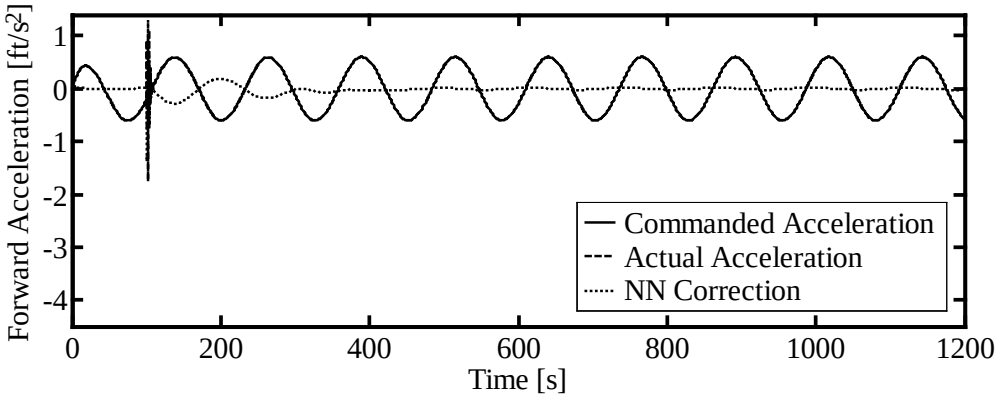


Figure 5.2.12. NN correction time history.

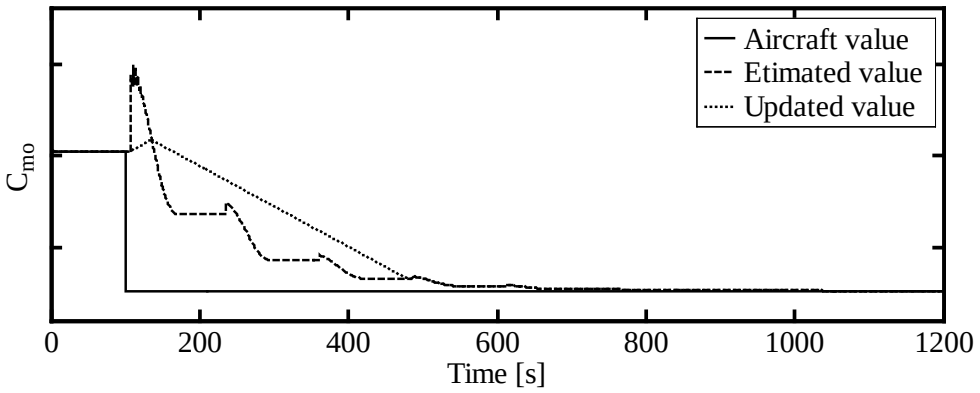


Figure 5.2.13. C_{m0} update time history.

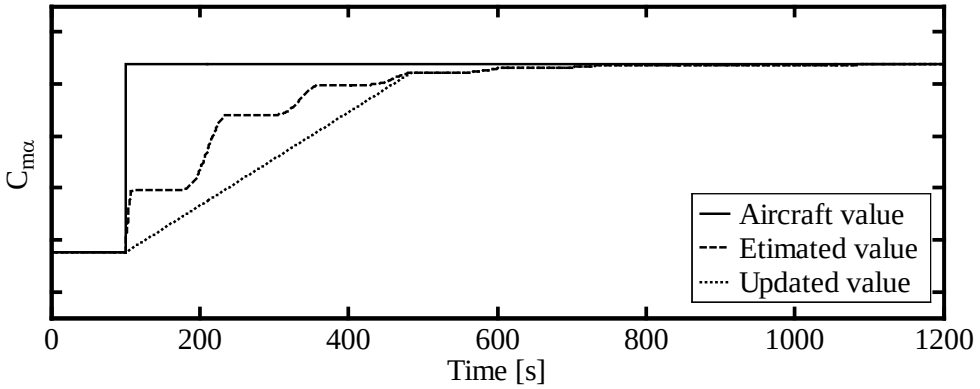


Figure 5.2.14. $C_{m\alpha}$ update time history.

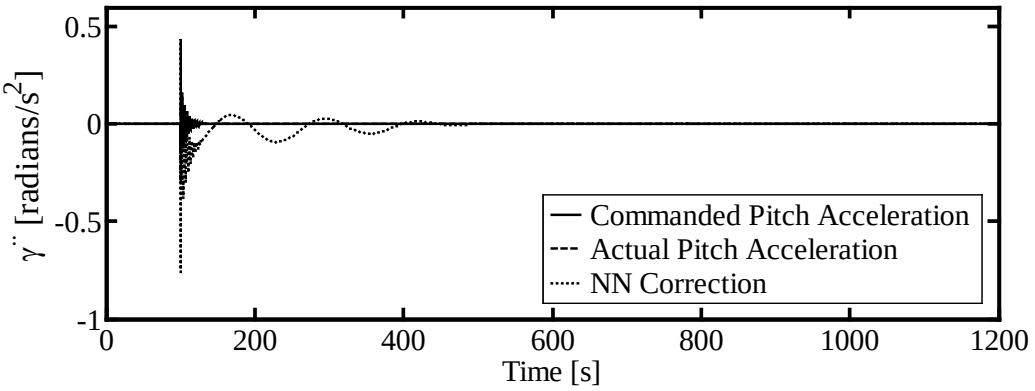


Figure 5.2.15. NN correction in pitch acceleration time history.

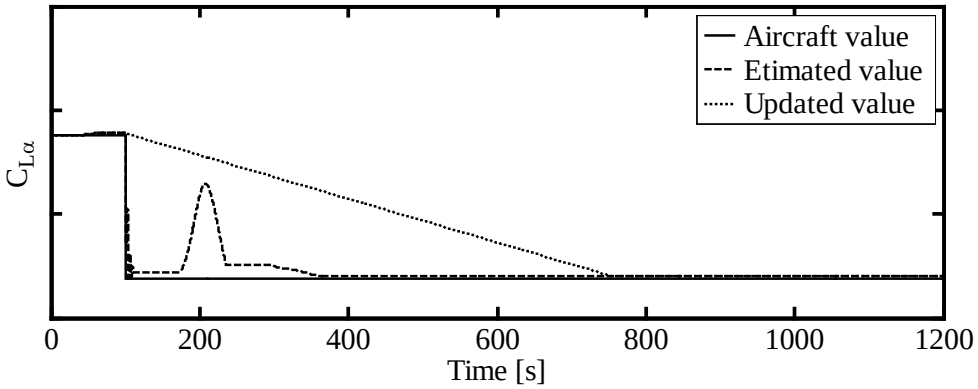


Figure 5.2.16. $C_{L\alpha}$ update time history

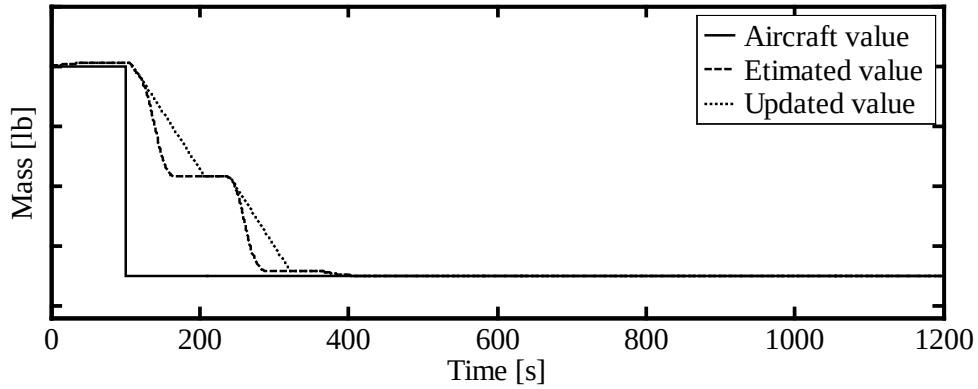


Figure 5.2.17. Mass update time history

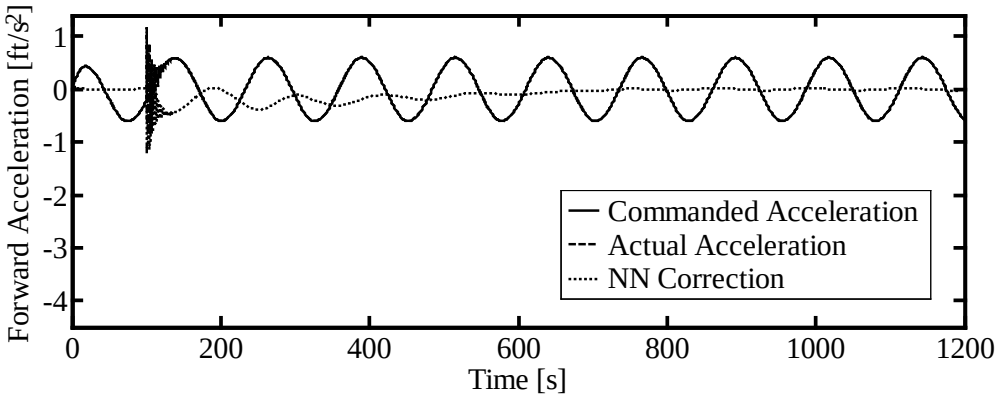


Figure 5.2.18. NN correction in forward acceleration time history.

5.3. Estimation of Parameter After Identifying the Changed Parameter

This routine allowed for the estimation of three variables (C_{m0}, $C_{m\alpha}$ and $C_{m\delta e}$) at the same time. It was not possible to estimate these parameters simultaneously using the technique mentioned in Section 4.2.2, as they are highly coupled to single pitch acceleration error ($\ddot{\gamma}_{error}$). It was not possible to update all three parameters at the same time and drive the modeling error to zero for all flight conditions, since there were three unknowns (i.e. C_{m0}, $C_{m\alpha}$ and $C_{m\delta e}$) and one equation (i.e. $\ddot{\gamma}_{model}$) to solve. In this routine a test was performed to identify the changed parameter and then estimate that particular parameter instead of estimating all possible parameters at the same time. An important assumption made in the development of the technique was that only one parameter could change at a time during any flight condition. This technique was described in Section 4.2.3.

The flight maneuver used to estimate the parameters in this routine was a steady and leveled flight condition at airspeed of 110 knots. During the test for identification of changed parameter, the airspeed was changed in a step command from 110 knots to 130 knots.

It is important to note that all the parameters were tested at the same time to identify the changed parameter during the flight. A new linear aircraft model was generated for each parameter tested. Remember a linear aircraft model was used to calculate modeling error, by comparing it with the actual aircraft. So a separate modeling error was calculated for each parameter and used in its parameter estimation equation. This allowed the estimation and testing of all the possible parameters at the same time without affecting each other. This process took only one change in flight condition to perform the identification test for all three parameters and hence reduced the process time significantly. The word “local modeling error” is used to

represent the modeling error that was calculated by comparing the response of actual aircraft and the aircraft model dedicated for that particular parameter estimation.

5.3.1. Estimation of a Change in C_{m0}

In Figure 5.3.1 plots 1, 3, and 5 show the update of the three parameters C_{m0}, $C_{m\alpha}$, and $C_{m\delta e}$ independent of each other, whereas plots 2, 4, and 6 show the corresponding response of the $\ddot{\gamma}_{error}$ with the update of each parameter. Finally plots 7 and 8 show the airspeed and neural network correction with time. All the plots are presented on the same page to make it convenient for the reader to examine them on same time scale.

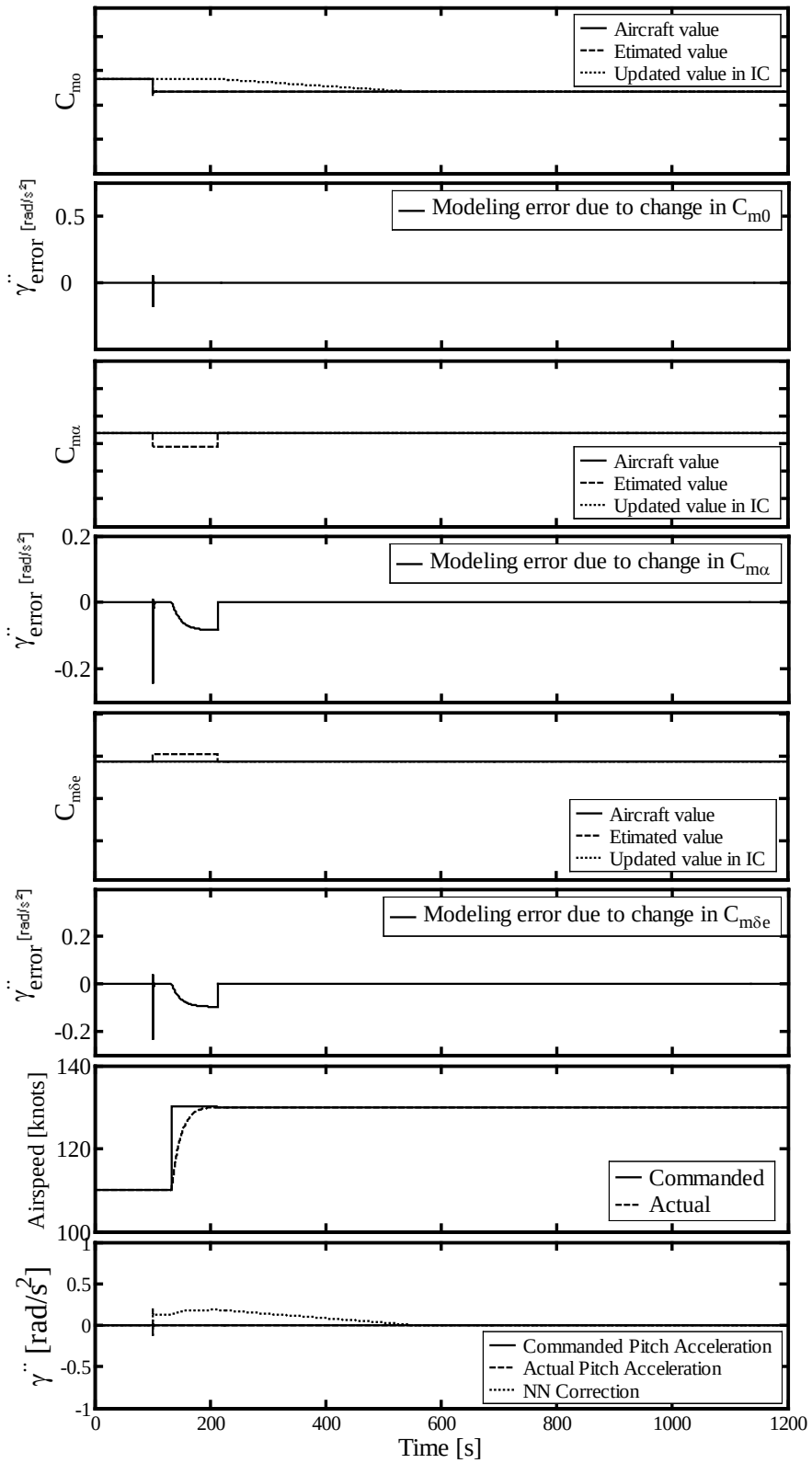
The aircraft was flown in a level flight condition at airspeed of 110 knots. To simulate the modeling error, C_{m0} was reduced by 50% of its original value in the actual aircraft at a time step of 100 seconds. This caused a modeling error in pitch acceleration exceed the threshold of 10^{-8} rad/sec^2, which activated the parameter estimation technique.

To perform the test for identification of the changed parameter, each parameter was estimated separately, which drove the local modeling error to zero (using update equations described in Section 4.1.2) at the present flight condition. Then by keeping the new estimated value of that particular parameter constant, the airspeed was changed to a new value and the magnitude of the local modeling error was observed. If the test was performed with the parameter that had actually changed (i.e. C_{m0} in this case) and its changed value was estimated, then the local modeling error at the new flight condition would be near zero. But if the test was performed using the parameter that did not change in the actual aircraft the value of local modeling error would be higher than that of other two parameters after being tested similarly.

All three parameters were estimated independently from a time step of 100 seconds to

133 seconds (plots 1, 3, and 5 in Figure 5.3.1) in order to reduce the local modeling error at that particular flight condition to zero (plots 2, 4, and 6 in Figure 5.3.1). Then the airspeed was changed from 110 knots to 130 knots (see plot 7), keeping the estimated values of the parameters constant. In Figures 5.3.1 plots 4 and 6 show that as the airspeed reached 130 knots (at about 213 seconds) the local modeling errors due to the updates of $C_{m\alpha}$ and $C_{m\delta e}$ increased, whereas plot 2 shows that the local modeling error due to the update of C_{m0} was comparatively much less and close to zero. Since the local modeling errors of C_{m0} was less than that of the other parameters, it was decided that the value of C_{m0} changed in the actual aircraft. Small or near zero local modeling error in C_{m0} also indicated that updating its value in the inverse controller could reduce the overall modeling error. Then the new value of C_{m0} was updated inside the inverse controller at a rate of 0.00002 units per time step (dotted line in plot 1 of Figure 5.3.1). Also, the old values of the other parameters (i.e. $C_{m\alpha}$ and $C_{m\delta e}$) were retained. Ultimately by the time step of 559 seconds (plot 1), the new value of C_{m0} became fully updated in the inverse controller. Plot 8 shows that the neural network error compensation value, which increased from a time step of 100 seconds, and decreased to zero (at about 559 seconds) as the new value of C_{m0} was updated in the inverse controller.

It should be noted that in plot 1 (Figure 5.3.1), due to fast estimation of C_{m0}, its estimated value (dashed line) coincides with its actual value (solid line). Whereas in plots 3 and 5 (Figure 5.3.1 and 5.3.1) the updated values of $C_{m\alpha}$ and $C_{m\delta e}$ coincide with their actual value (solid line). Also in plot 8 (Figure 5.3.1), the actual pitch acceleration coincides with its commanded value.



Plot 1:
Update of C_{m0}

Plot 2:
Modeling error with C_{m0} update.

Plot 3:
Update of $C_{m\alpha}$

Plot 4:
Modeling error with $C_{m\alpha}$ update.

Plot 5:
Update of $C_{m\delta e}$

Plot 6:
Modeling error with $C_{m\delta e}$ update.

Plot 7:
Airspeed Time History

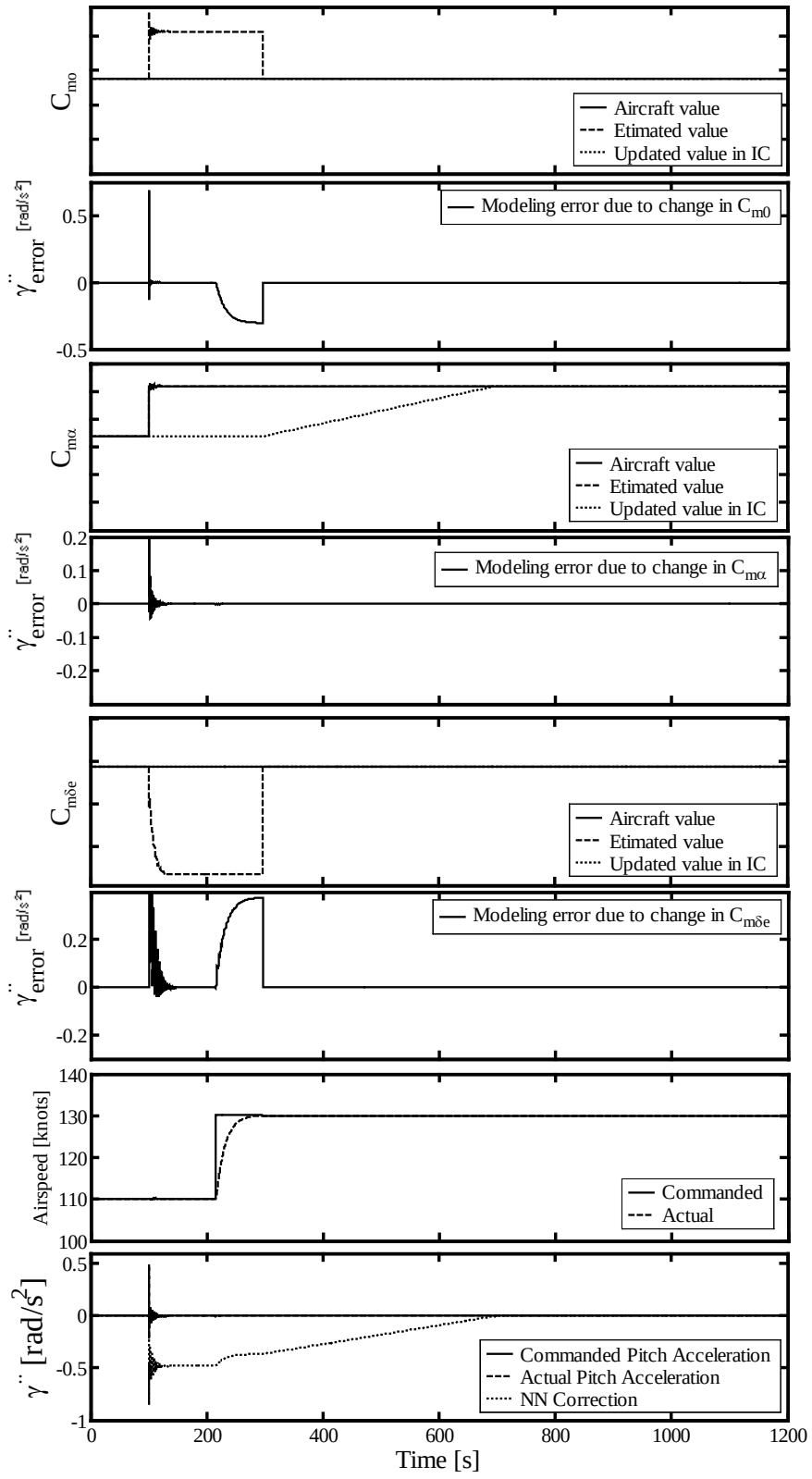
Plot 8:
Pitch Acceleration and NN correction.

Figure 5.3.1. Estimation of a change in C_{m0}.

5.3.2. Estimation of a Change in $C_{m\alpha}$

The aircraft was flown in a level flight condition at airspeed of 110 knots and at a time step of 100 seconds $C_{m\alpha}$ was reduced by 50% of its original value in the actual aircraft. This activated the estimation routine, since the modeling error in pitch acceleration exceeded the threshold value of 10^{-8} rad/sec^2. All three parameters were estimated (plots 1, 3, and 5 in Figure 5.3.2) from 100 to 215 seconds such that there local modeling errors (plots 2, 4, and 6 in Figure 5.3.2) reduce below the threshold value. Then the airspeed was changed to 130 knots and the new local modeling errors of the parameters were compared at about 296 seconds. Since the local modeling error due to $C_{m\alpha}$ was much less than that of others, $C_{m\alpha}$ was identified as the changed parameter in the actual aircraft. Then starting from 296 seconds the new value $C_{m\alpha}$ was slowly updated inside the inverse controller at a rate of 0.001 units per time step. The old values of C_{m0} and $C_{m\delta e}$ were restored at 296 seconds. Ultimately by the time step of 697 seconds (plot 3 in Figure 5.3.2), the new value of $C_{m\alpha}$ was fully updated in the inverse controller. Neural network error compensation value that increased from the time step of 100 seconds went to zero (at about 697 seconds) as the new value of $C_{m\alpha}$ was updated in the inverse controller (plot 8 in Figure 5.3.2).

Again the reader is cautioned that in plots 1 and 5 (Figure 5.3.2), the dotted lines representing the updated values of C_{m0} and $C_{m\delta e}$ coincide with their actual values. In plot 3 the estimated value of $C_{m\alpha}$ coincides with its actual value. Also in plot 8, the actual pitch acceleration (dashed line) hides under the commanded pitch acceleration (solid line).



Plot 1:
Update of C_{m0}

Plot 2:
Modeling error with C_{m0} update.

Plot 3:
Update of $C_{m\alpha}$

Plot 4:
Modeling error with $C_{m\alpha}$ update.

Plot 5:
Update of $C_{m\delta e}$

Plot 6:
Modeling error with $C_{m\delta e}$ update.

Plot 7:
Airspeed Time History

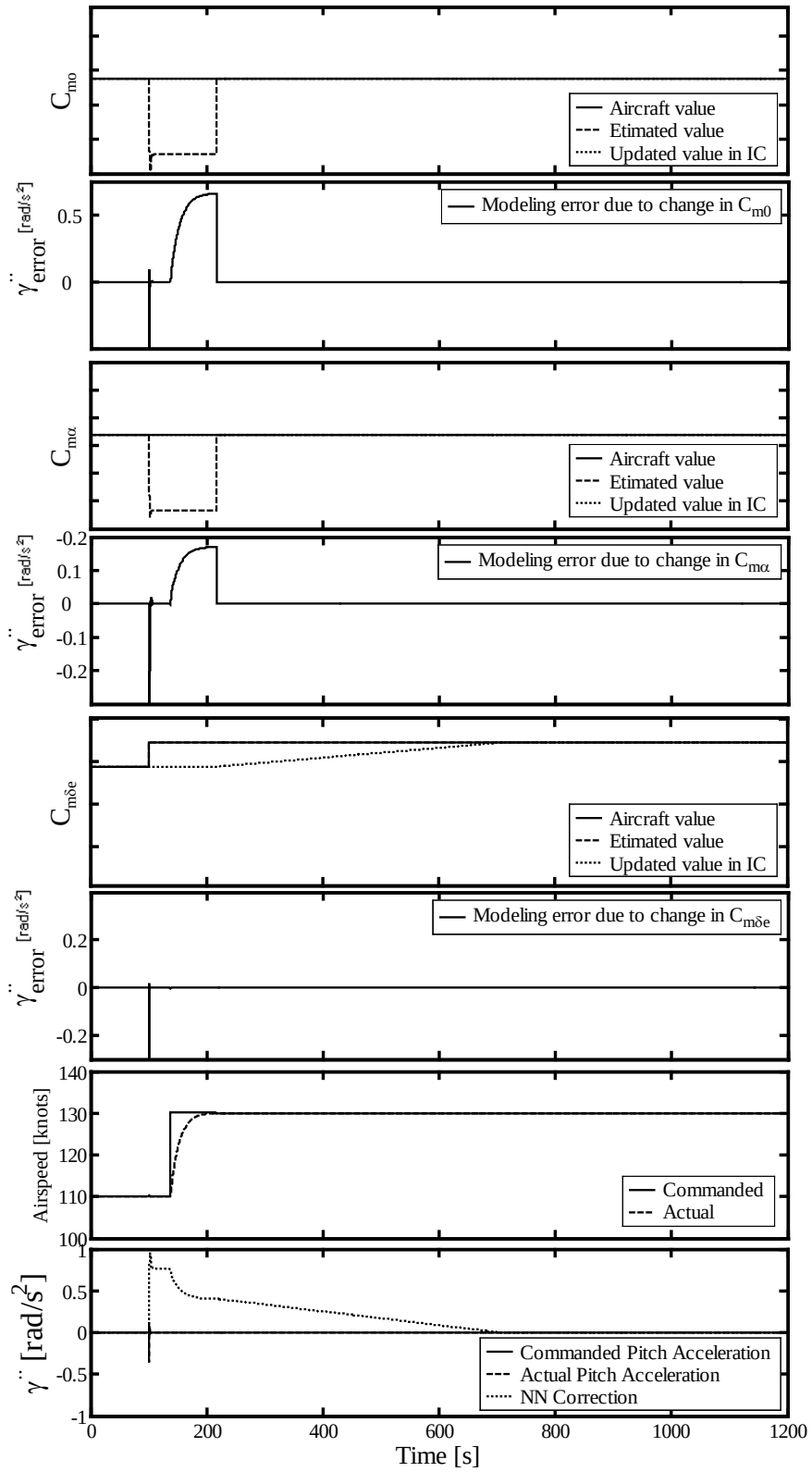
Plot 8:
Pitch Acceleration and NN correction.

Figure 5.3.2. Estimation of a change in $C_{m\alpha}$.

5.3.3. Estimation of a Change in $C_{m\delta e}$.

Figure 5.3.3 shows the parameter identification test and update in the inverse controller when $C_{m\delta e}$ was changed in actual aircraft. At 100 seconds the value of $C_{m\delta e}$ (plot 5 in Figure 5.3.3) was reduced to its half. This caused the local modeling errors (plots 2, 4, and 6 in Figure 5.3.3) to exceed the threshold of 10^{-8} rad/sec^2 at the same time step, activating the routine. All three parameters were estimated (plots 1, 3, and 5 in Figure 5.3.3) and their local modeling errors (plots 2, 4, and 6 in Figure 5.3.3) were reduced below their threshold values. Airspeed was changed to 130 knots at about 136 seconds (see plot 7) after the new values of all three parameters were estimated. At the new airspeed the local modeling errors were compared at about 217 seconds (plots 2, 4, and 6 in Figure 5.3.3). The local modeling error due to $C_{m\delta e}$ was found to be lower than those of other, so it was identified as the changed parameter. The new value of $C_{m\delta e}$ was updated in the inverse controller at a rate of 0.001 units per time step starting from 217 seconds. At the same time step the old values of C_{m0} and $C_{m\alpha}$ were restored. The neural network error compensation (plot 8 in Figure 5.3.3) that increased at 100 seconds decreased to zero as the new value of $C_{m\delta e}$ was updated (plot 5) inside the inverse controller at about 704 seconds.

It should be noted that in plots 1 and 3 in Figure 5.3.3 the updated values of C_{m0} and $C_{m\alpha}$ coincide with their actual values, and in plot 5 the estimated value of $C_{m\delta e}$ coincide with its actual value. Also in plot 8 (Figure 5.3.3), the actual pitch acceleration (dashed line) coincides with its commanded value (solid line).



Plot 1:
Update of C_{m0}

Plot 2:
Modeling error
with C_{m0} update.

Plot 3:
Update of $C_{m\alpha}$

Plot 4:
Modeling error
with $C_{m\alpha}$ update.

Plot 5:
Update of $C_{m\delta e}$

Plot 6:
Modeling error
with $C_{m\delta e}$ update.

Plot 7:
Airspeed
Time History

Plot 8:
Pitch Acceleration
and NN correction.

Figure 5.3.3. Estimation of a change in $C_{m\delta e}$.

CHAPTER SIX

6. DISCUSSION

The results shown in Chapter 5 demonstrate that the parameter estimation technique provides a good estimate of their changed values during flight. Though the number of parameters estimated is limited, the technique requires less computational power as compared to other techniques, and it can be easily implemented online.

The technique presented in this work is specially developed to work with the architecture of the neural-based advanced flight control system developed at Wichita State University, though it can be easily modified to work with other adaptive flight control systems that use online parameter adaptation.

Like most other techniques, the one mentioned in this thesis is a numerical method for extracting parameters from real-time flight data. The computational simplicity of this technique lies in the fact that it solves a single equation for one parameter using flight data at the current time step, and updates the new value of the parameter slowly with time. Most other techniques of parameter estimation solve a set of coupled equations using flight data from a set of previous time steps and arrive at the new value of the parameter. This makes them computationally intensive.

Another problem with conventional techniques is the selection of useful flight data from previous time steps and the elimination of less informative data from a prolonged similar flight condition. The technique presented in this thesis circumvents this problem because the parameters are not updated at a single time step depending upon a set of previous time step flight data, but rather are updated slowly by looking at the modeling error at the current time step. It is not necessary to worry about eliminating the less informative flight data, since it will not saturate

the value of the parameter or make it less responsive to further change but will simply stop the update of the parameter value unless it sees a modeling error from new useful flight data. The magnitude of the parameter update simply depends upon the modeling error due to the current value of that parameter at the current time step.

Certainly the computational simplicity of the technique presented penalizes the convergence speed of the parameter value to the new value, but this is desired for the controller for which it is designed. The controller uses artificial neural networks for immediate and temporary adaptation to the modeling error and maintains control over the aircraft, whereas parameter estimation is used for slow and permanent adaptation of the controller for modeling error caused by system degradation.

Examining the results in Section 5.2, the parameter estimation looks slow, but the controller still maintains control over the aircraft because of the presence of a neural network adaptation. As the new value of the changed parameter is approached, the neural network adaptation approaches zero. This shows that permanent adaptation has been provided to the inverse controller by updating the changed parameter.

Examining the results in Section 5.3 for the update routine presented in Section 4.2.3, the parameter estimation is fast enough but the newly estimated parameter is slowly updated into the inverse controller, since priority is given to the neural networks to provide primary adaptation for the modeling error. The results presented in Section 5.3 are for three parameters (i.e., C_{m0}, $C_{m\alpha}$, and $C_{m\delta e}$) since these parameters are highly coupled to the modeling error in pitch acceleration and were not possible to be estimated using the second method (method described in Section 4.2.3). But this technique can be easily extended to more number of parameters without any

significant effect on convergence time, since it estimates all the parameters at the same time using a single change in airspeed.

The three methods have been described and their corresponding results have been presented. The first method (described in Section 4.2.1) requires human intervention and cannot be implemented effectively since it is often difficult to judge the cause of modeling error during flight. The second method (described in Section 4.2.2) can be implemented since it is fully autonomous, but is limited to the estimation of four parameters at a time. The third method (described in Section 4.2.3) is more effective and preferred over other methods since it is autonomous and can be extended to estimate any number of parameters at a time, provided only one parameter would change at any given time during flight.

CHAPTER SEVEN

7. CONCLUSIONS

The purpose for developing the new parameter estimation technique was to provide permanent adaptation to the modeling errors due to system degradation in the neural-based advanced flight control system developed at Wichita State University. Unlike most other parameter estimation techniques, this method was designed to estimate changes in parameters during the flight, therefore it requires an initial value of the parameters that are used inside the controller. The technique was implemented in a longitudinal controller, so the parameters estimated were C_{m0}, $C_{m\alpha}$, $C_{m\delta e}$, $C_{L\alpha}$, and m. Different routines were presented (Section 4.2) along with the assumptions made in each, and the performance of each routine for different test cases is presented as results to Chapter 5.

The results demonstrate the ability of the parameter estimation technique to provide the required performance and improve the overall adaptation of the control system. The technique is slow compared to most other online parameter estimation techniques but this is desired because the neural network provides a fast adaptation. Certainly the technique is very simple computationally and requires less processing power than other techniques. The number of parameters estimated is limited in one of the routines presented (Section 4.2.2), but another routine that identifies the changed parameter before updating it (Section 4.2.3) can estimate any number of parameters.

The technique presented in Section 4.2.2 can be improved to increase the convergence speed of parameters and increase the number of parameters estimated simultaneously. Also, using some variation in the gradient decent method can increase the performance of the technique. The technique could be modified to work independently of the controller for which it

is designed and can be used along with other adaptive control systems. The technique was not tested for process and measurement noise, which could also provide future work.

LIST OF REFERENCES

LIST OF REFERENCES

[1] 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.

[2] Iliff, K. W., “Aircraft Parameter Estimation,” AIAA-1987-623, Aerospace Sciences Meeting, 25th, Reno, NV, Jan. 12-15, 1987.

[3] Ishimoto, S., “ New Algorithm of Maximum Likelihood Parameter Estimation for Flight Vehicles,” AIAA-97-3784, AIAA Atmospheric Flight Mechanics Conference, New Orleans, LA, Aug. 11-13, 1997

[4] Napolitano, M. R., Paris, A. C., Seanor, B. A., “Estimation of Lateral-Directional Aerodynamic Parameters from Flight Data for the NASA F/A-18 HARV,” AIAA-96-3420, AIAA Atmospheric Flight Mechanics Conference, San Diego, CA, July 29-31, 1996.

[5] Ananthasayanam, M. R., Sarkar, A. K., Vathsal, S., “Parameter Estimation Of A Flight Vehicle Using MMLE/BFGS Estimator Under Limited Measurements”, AIAA-2002-4624, AIAA Atmospheric Flight Mechanics Conference and Exhibit, Monterey, California, Aug. 5-8, 2002.

[6] Velo, J. G., and Bruce, K. W., "Aerodynamic Parameter Estimation for High Performance Aircraft Using Extended Kalman Filtering", *AIAA Journal of Guidance and Control*, vol. 20 (6), pp. 1257-1259.

[7] Mendel, J. M., *Discrete Techniques of Parameter Estimation*, Marcel Dekker, New York, 1973.

[8] Ljung, L., *System Identification: Theory for the User*, Prentice–Hall, Englewood Cliffs, NJ, 1987.

[9] Bodson, M., “An Information-Dependent Data Forgetting Adaptive Algorithm,” *Proceedings of the American Control Conference*, Seattle, WA, June 1995.

[10] Monaco, J., Ward, D., Barron, R., and Bird, R., “Implementation and Flight Test Assessment of an Adaptive, Reconfigurable Flight Control System,” AIAA Paper 97-3738, AIAA Guidance, Navigation, and Control Conference, New Orleans, LA, Aug. 11-13, 1997.

[11] Smith, L., Chandler, P. R., and Patcher, M., “Regularization Techniques for Real-Time Identification of Aircraft Parameters,” AIAA Paper 97-3740, AIAA Guidance, Navigation, and Control Conference, New Orleans, LA, Aug. 11-13, 1997.

[12] Chandler, P., Patcher, M., and Mears, M., "System Identification for Adaptive and Reconfigurable Control," *Journal of Guidance, Control, and Dynamics,* vol. 18(3), 1995, pp. 516–524.

[13] Gelb, A., "*Applied Optimal Estimation*", MIT Press, Cambridge, MA, 1973.

[14] Song, Y., Campa, G., Napolitano, M., Seanor, B., and Perhinschi, M. G., "Online Parameter Estimation Techniques Comparison Within a Fault Tolerant Flight Control System," *Journal Of Guidance, Control, And Dynamics,* vol. 25(3), May–June 2002, pp. 528-537.

[15] Neter, J., Kutner, M. H., Nachtsheim, C. J., and Wasserman, W., *Applied Linear Regression Models*, 3rd ed., Richard D. Irwin, Inc., Burr Ridge, IL, 1996.

[16] Atkeson, C. G., Moore, A.W., and Schaal, S., Locally Weighted Learning, *Artificial Intelligence Review*, vol. 11, Feb. 1997, pp. 11–73.

[17] Atkeson, C. G., Moore, A.W., and Schaal, S., Locally Weighted Learning for Control, *Artificial Intelligence Review*, vol. 11, Feb. 1997, pp.75–113.

[18] Eugene, A. M., "Real Time Parameter Estimation In The Frequency Domain," AIAA-99-4043, AIAA Guidance, Navigation, and Control Conference and Exhibit, Portland, OR, Aug. 9-11, 1999.

[19] Klein, V., "Aircraft parameter estimation in frequency domain," AIAA-1978-1344, Atmospheric Flight Mechanics Conference, Palo Alto, Calif., August 7-9, 1978.

[20] Linse D.J. and Stengel, R. F., "Identification of Aerodynamic Coefficients Using Computational Neural Networks," AIAA 92-0172, Aerospace Sciences Meeting and Exhibit, 30th, Reno, NV, Jan. 6-9, 1992.

[21] Raisinghani, S. C., and Ghosh, A. K., "Parameter Estimation of an Aeroelastic Aircraft using Neural Networks," *Sadhana*, vol. 25(2), April 2000, pp. 181-191.

[22] Pesonen, U. J., Steck, J. E., Rokhsaz, K., Bruner, S., Duerksen, N., "Adaptive Neural Network Inverse Controller for General Aviation Safety," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 27(3), May – June 2004, pp. 434-443.

[23] Rysdyk, R., Nardi, F., and Calise, A., "Robust Adaptive Nonlinear Flight Control Applications Using Neural Networks," *Proceedings of the American Control Conference*, Albuquerque, NM, June 1997.

[24] *Matlab Aerospace Blockset User Manual*, The Mathworks Inc., Natick, Massachusetts, 2002.

[25] Roskam, J., *Airplane Flight Dynamics and Automatic Flight Controls,* DAR Corporation, KS 66049, USA, 2003.

[26] Fausett, L., *Fundamentals of Neural Networks*, Prentice Hall, New York, 1994.

[27] *Matlab Neural Network Toolbox User Manual*, The Mathworks Inc., Natick, Massachusetts, 2002.

[28] *NeuralWorks Professional II User Manual*, NeuralWare Inc., Carnegie, Pennsylvania, 1997.