JIMMA UNIVERSITY

JIMMA INSTITUTE OF TECHNOLOGY

Faculty of Electrical and Computer Engineering

Master Program on Control and Instrumentation Engineering

F-16/MATV; Optimal Flight Controller Design Using Robust and Discrete Linear Quadrature Gaussian [RLQG] Controller

A Final Year Thesis Submitted to School of Graduate Studies of Jimma Institute of Technology in Partial Fulfillment for the Requirement for Masters of Science in Control and Instrumentation Engineering.

KASSAHUN BERISHA RM 0132/08

February 28, 2019

Jimma, Ethiopia

F-16/MATV; Optimal Flight Controller Design Using Robust and Discrete Linear Quadrature Gaussian [RLQG] Controller

BY

G. KASSAHUN BERISHA

RM 0132/08

A Graduate Thesis Submitted to School of Graduate Studies of

Jimma Institute of Technology

Jimma University;

In Partial Fulfillment for the Requirement for Masters of Science

In Control and Instrumentation Engineering

F-16/MATV; Optimal Flight Controller Design Using Robust and Discrete Linear Quadrature Gaussian [RLQG] Controller

BY

G. Kassahun Berisha RM 0132/08

ADVISOR: DR. A PRASHANTH

COADVISOR: DR. T. AMRUTH

February 28, 2019

DECLARATION

This is to certify that this thesis entitled **F-16/MATV; Optimal Flight Controller Design Using Robust and Discrete Linear Quadrature Gaussian [LQG] Controller,** submitted in partial fulfillment of the requirements for the award of the degree of M.Sc. in Control and Instrumentation Engineering to the School of Graduate Studies, Jimma Institute of Technology (JiT), through the faculty of Electrical and Computer Engineering under supervision of Industrial Control chair, done by **Mr. G. Kassahun Berisha** is an authentic work carried out by him. The matter embodied in this project work has not been submitted earlier for award of any degree or diploma to the best of my knowledge and belief.

Name of the student: G. Kassahun Berisha

Date._____ Signature _____

APPROVAL OF THE FIRST ADVISOR

Name of the first advisor: Dr. A. Prashanth

Date._____ Signature _____

APPROVAL OF THE SECOND ADVISOR

Name of the co-advisor: Dr. RT. Amruth

Date._____ Signature _____

APPROVAL OF EXAMINERS

	Name	**Signature**	**Data**	
1.	_____	_____	_____	(External)
2.	_____	_____	_____	(Internal)
3.	_____	_____	_____	(C. Chair)
4.	_____	_____	_____	
5.	_____	_____	_____	

TABLE OF CONTENTS

ABSTRACT

This thesis presents the robust controller design for nonlinear F-16/MATV (Multi – Axis – Thrust - Vectoring) flight control system. The linearization of nonlinear flight system is guaranteed by feedback linearization using Tayler series expansion and the optimal LQG controllers are designed to achieve the steady state flight condition. The comparison of the using continuous and discrete LQG is fully discussed for flight control system. First the continuous optimal LQG controller is designed for continuous time state space represented flight dynamic system based on separation principle. But the designed optimal LQG controller requires a very large controller gain to achieve the design objective, which lacks high sensitivity and leads to high cost system. This leads to design a controller using discrete LQG controller for discrete time state space represented flight control system. After implementing this controller into the system in MATLAB Simulink environment, using singular value decomposition technique, it is founded that the performance and robustness of the design objective is satisfied.

Keywords: Robust Controller, Optimal LQG, Discrete LQG, Robustness, **S**ensitivity, Performance, Linearization

ACKNOWLEDGEMENT

Above all I would like to recognize God's gregarious support in getting all things done, the One who created everything, the One who gives me Wisdom and Strength, the One who picks me up when I feel so broken, to my Almighty LORD JESUS.

Second I am grateful to Jimma Institute of Technology (JiT) and faculty of Electrical and Computer Engineering that intended to offer MSc Program in Control and Instrumentation Engineering. The financial sponsorship support by Wolaita Sodo University is highly appreciated.

I would like to express my deepest gratitude to my advisor, **Dr. A. Prashanth**, and co-advisor **Dr. RT. Amruth**, who taught me principles that helped me with doing this thesis and very valuable comments and advice on this thesis. Above all for their continuous follow up of my work up to completion.

I would also like to acknowledge **Mr. Abu Feyo**, Faculty of Electrical and Computer Engineering program coordinator of Jimma Institute of Technology; **Mr. Amanuel Tesfaye,** Assistant Lecture and MSc student at Jimma institute of technology, and **Mr. Ollata Kallano,** Assistant Lecture at Dilla University and MSc student at Adama Science and Technology, as the second reader of this thesis. Really I am very thankful for their important comments.

To my ever-loving parents who are always there to give me courage to pursue my goals and provide financial and emotional support.

To my classmates and **Mr. Tesfabrihan Shoga,** Industrial Control and Instrumentation engineering chair of Jimma Institute of Technology, who are always capable of giving me enough faith in doing this thesis works at times of failures and for all the discussions and material support.

Thank You

G. Kassahun Berisha

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

x_E	=	North Position
y_E	=	East Position
z_E	=	Altitude
q	=	Pitch Angle
r	=	Yaw Angle
p	=	Roll Angle
V_T	=	Total Velocity
α	=	Angle of Attack
β	=	Angle of Sideslip
ṗ	=	Roll Rate
q̇	=	Pitch Rate
ṙ	=	Yaw Rate
a_{nx}	=	Normal Acceleration in X - Axis
a_{ny}	=	Normal Acceleration in Y – Axis
a_{nz}	=	Normal Acceleration in Z – Axis
M	=	Match Number
$\bar{q}$	=	Free Stream Dynamic Pressure
p_s	=	Static Pressure
DAC	=	Direct Adaptive Control
MATV	=	Multi-Axis Thrust Vectoring
RFCS	=	Reconfigurable Flight Control System
LQG	=	Linear Quadrature Gaussian
LQR	=	Linear Quadrature Regulator
IAC	=	Indirect Adaptive Control
FDI	=	Fault Detection and Isolation
SVD	=	Singularity Value Decomposition
	=	Aircraft mass, kg
VISTA	=	Variable Stability In-flight Simulator Test Aircraft
I_X	=	roll moment of inertia, $kg.m^2$
I_y	=	pitch moment of inertia, $kg.m^2$
I_z	=	yaw moment of inertia, $kg.m^2$
H_{eng}	=	Engine angular moment
F_T	=	Total thrust force , N
δ_a	=	Aileron deflection, rad
δ_e	=	Elevator deflection, rad
δ_r	=	Rudder deflection, rad
δ_{lef}	=	Leading edge flap deflection, rad
$\bar{L}$	=	Rolling moment $N.m$
$\bar{M}$	=	Pitching moment $N.m$
$\bar{N}$	=	Yawing moment $N.m$
$\bar{X}$	=	Axial force component, N
$\bar{Y}$	=	Later force component, N
$\bar{Z}$	=	Normal force component, N
C_*	=	Aerodynamic coefficient of, $*$
g	=	Gravity constant m/s^2
δ_{t_y}	=	Horizontal thrust nozzle deflection, rad
δ_{t_z}	=	Vertical thrust nozzle deflection, rad

Superscripts

ref	=	Reference
^	=	Estimate
~	=	Error

CHAPTER ONE

1. Introduction

Guidance and control law design for aerospace engineering and strategic applications has become a celebrated research topic in past few decades with its wide applicability both in industry and academia [1]. In this thesis a new control strategy is designed and discovered for a F-16/MATV. The Multi-Axis Thrust Vectoring (MATV) program has been a joint effort by Lockheed Fort Worth Company (LFWC), Wright Laboratory (WL), General Electric (GE), the Air Force Flight Test Center (AFFTC), and the 422nd Test and Evaluation Squadron (TES). The program consisted of integrating a multi-axis thrust vectoring nozzle system with the Variable Stability In-flight Simulator Test Aircraft (VISTA)/F-16 aircraft. The integrated system was used in flight test to demonstrate flight envelope expansion above the normal F-16 angle-of-attack (AOA) limits and to evaluate potential tactical benefits gained by utilizing thrust vectoring in air-to-air combat [2].

The objective of the MATV program was to utilize multi-axis thrust vectoring to expand the F-16 AOA limits into the post-stall regime and to evaluate the tactical benefits gained by utilizing expanded AOA in air-to-air combat. For the MATV program, a new control law is designed using Linear Quadrature Gaussian technique (LQG).

In control theory, the Linear Quadratic Gaussian (LQG) control problem is one of the most fundamental optimal control problems. It concerns uncertain linear systems disturbed by additive white Gaussian noise, having incomplete state information (i.e. not all the state variables are measured and available for feedback) and undergoing control subject to quadratic costs. Moreover, the solution is unique and constitutes a linear dynamic feedback control law that is easily computed and implemented. Thus, the LQG controller is also fundamental to the optimal control of perturbed non-linear systems [3].

The LQG controller is simply the combination of a Kalman filter, i.e. a Linear Quadratic Estimator (LQE), with a Linear Quadratic Regulator (LQR). The separation principle guarantees that these can be designed and computed independently. LQG control applies to both linear time-

invariant systems as well as linear time-varying systems. The application to linear time-invariant systems is well known. The application to linear time-varying systems enables the design of linear feedback controllers for non-linear uncertain systems [4].

The LQG controller itself is a dynamic system like the system it controls. Both systems have the same state dimension. Therefore, implementing the LQG controller may be problematic if the dimension of the system state is large. The reduced-order LQG problem (fixed-order LQG problem) overcomes this by fixing a priori the number of states of the LQG controller. This problem is more difficult to solve because it is no longer separable. In addition, the solution is no longer unique. Despite these facts, numerical algorithms are available to solve the associated optimal projection equations, which constitute necessary and sufficient conditions for a locally optimal reduced-order LQG controller. Figure 1 shows the general overview of Linear Quadrature Gaussian controller for flight control system.

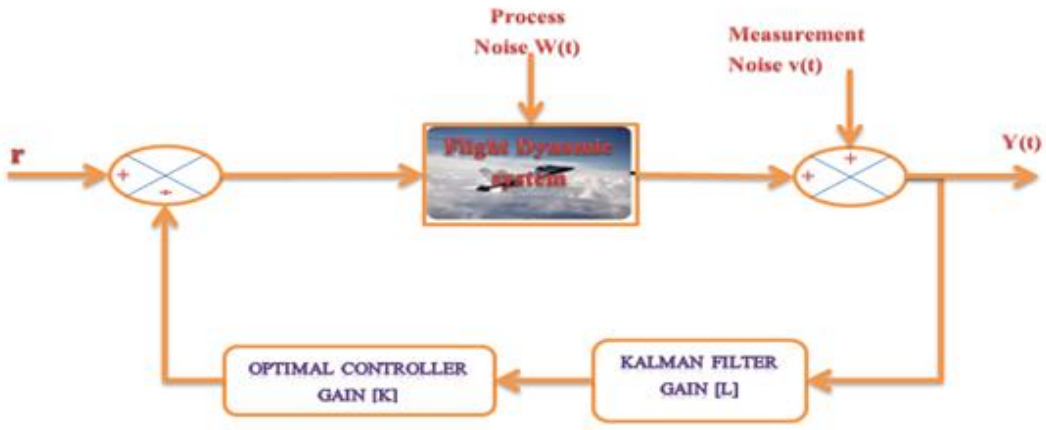

Figure 1: The LQG flight Control model system

LQG optimality does not automatically ensure good robustness properties. The robust stability of the closed loop system must be checked separately after the LQG controller has been designed. To promote robustness some of the system parameters may be assumed stochastic instead of deterministic. The associated more difficult control problem leads to a similar optimal controller of which only the controller parameters are different. Because it is intended that the transfer function of the closed loop system should be close to that of the model.

Thus, a new controller for a Non-Linear F-16/MATV fighter aircraft was designed to enhance the problem of classical control method. The controller employs the Linear Quadratic Gaussian [LQG] with linear quadratic estimator (LQE) and Optimal Regulator. It was applied to minimize the mean-squared estimation error and for achievement of better performance. The good robustness characteristic of Linear Quadrature Gaussian [LQG] controller was checked.

1.1. Problem Statement

Any traditional Flight Control System will work reasonably well, but it may possibly become unstable, poor performance and less robustness. Since the permissible value of adaptive gain depends upon the amplitude and type of the forcing signal, the stability analysis of any model reference adaptive system is invariably difficult.

1.2. Objectives

1.2.1. General Objective

The general objective achieved under this final year thesis is the Optimal robust and discrete Linear Quadratic Gaussian (LQG) controller for the Nonlinear F-16/MATV Flight control system is discovered and designedm.

1.2.2. Specific Objectives

The specific objectives achieved so that the general objectives will be guaranteed are the following.

- ✓ The Flight System is mathematically modeled.
- ✓ A linearization technique was applied to linearize the nonlinear dynamic flight system.
- ✓ An optimal continuous LQG controller for the Flight control system is designed.
- ✓ An optimal discrete LQG controllers for the Flight control system is designed.
- ✓ Comparative analyisis of Performance specificastions.
- ✓ Comparison of the robustness of LQG controller with that of traditional flight control system.

1.3. Significance of the Study

This thesis plays a great role in Aeronautics, Space and Defense industries to overcome a problem of traditional flight control systems approach that is good performance and robustness properties cannot be guaranteed for a highly nonlinear flight dynamics.

1.4. Scope of the Research

The design of optimal robust and discrete Linear Quadratic Gaussian Controller with Feedback linearization was presented, but the control law design was based on the nonlinear model of an F-16 aircraft with MATV (Multi-Axes Thrust Vectoring) model. As a mathematical modeling of the fight dynamics out of the scope of Control and Instrumentation engineering, the nonlinear F-16/MATV model is as derived in the journal of guidance, control, and dynamics [5]. The project is primarily concerns to build the control system for flight control for F-16/MATV model. The scope of the project is:

1. Reviewing of nonlinear F-16/MATV Flight dynamics and implementation on MATLAB
2. Linearization of nonlinear F-16/MATV flight dynamics and implantation on MATLAB
3. Design and implantation of the continuous time LQG Controller on MATLAB
4. Design and Simulate of the Discrete LQG
5. Checking the Robustness of LQG controller
6. Compare the Stability analysis of the two Controllers

1.5. Methodology

The methodology to be followed to meet the specified objectives will be

- ✓ Literature Review about the flight control system
- ✓ Take some assumptions to linearize the model (angle and position to be zero).
- ✓ Designing Filter and LQG controllers.
- ✓ Implementation of controller designed in MATLAB.
- ✓ Discussing on the results obtained.
- ✓ Drawing Conclusion about the overall thesis

- ✓ Writing the Recommendation

The following figure shows the general methodology used in final year thesis for achievement of the set objectives.

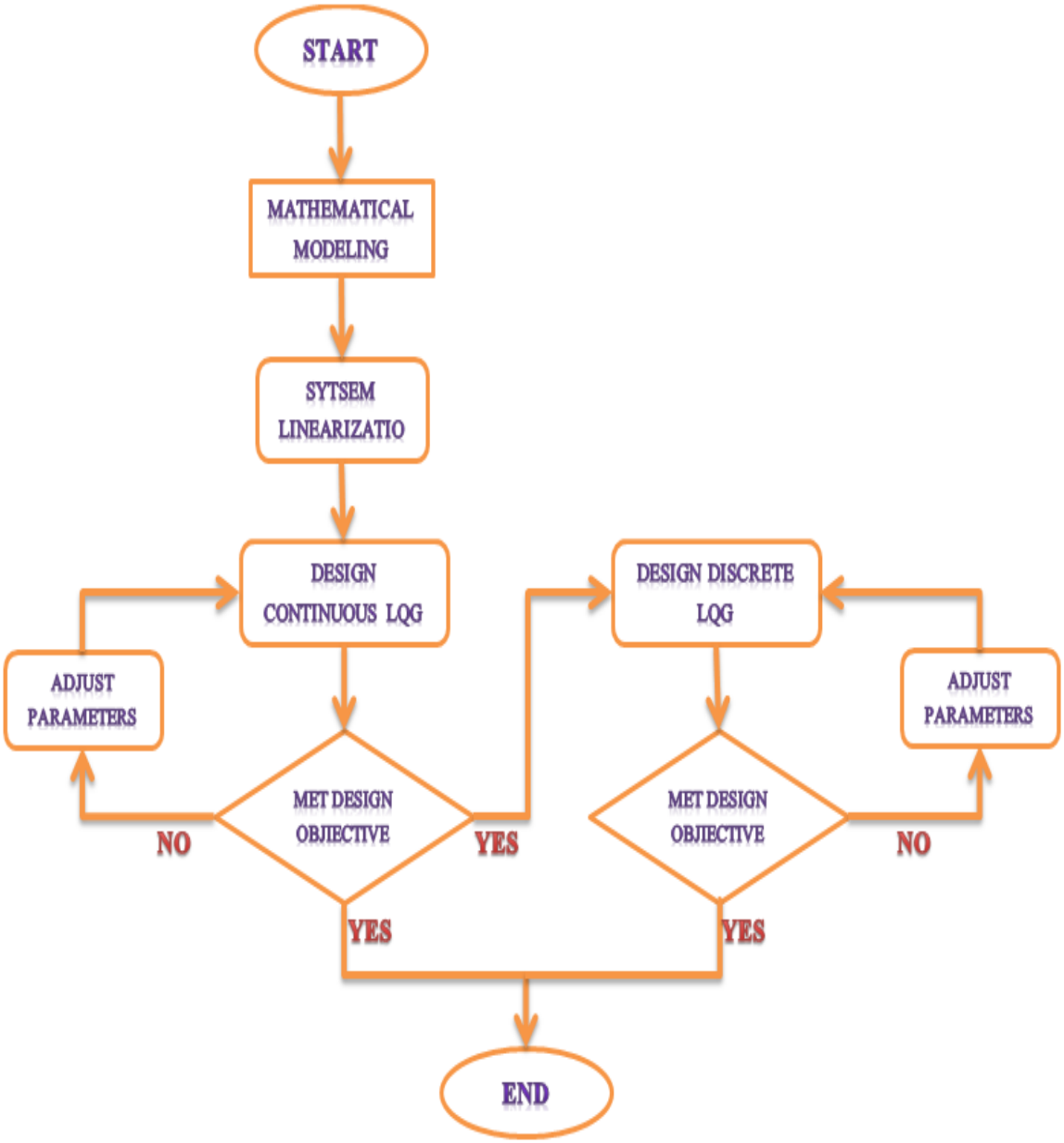

Figure 2 The general flow chart of design methodology

1.6. Thesis Outline

The thesis consists of five chapters that will explain more details about this project. The second chapter focused on the literature reviewed and fully outlines the previous work of scholars under different flight controller consideration. This chapter also highlights the merits and demerits of the work of those previous works.

The third chapter will focus on the mathematical modeling of the F-16/MATV flight dynamic equations used for control system design. These derived dynamic equations are classified as the force, moment, kinematic and navigation equations. Besides this, the aerodynamic coefficient equations are also addressed for full flight controller design. The implementation of nonlinear F-16/MATV flight dynamics in MATLAB environment is also covered under this chapter. It continues with the linearization of nonlinear F-16/MATV dynamics obtained from previous sub – section and it is all about the various LQG controller design and implementation of the controllers in MATLAB Simulink environment. The LQG controllers are designed for both continuous time and discrete time state space represented flight dynamic system.

The fourth chapter will discuss about the MATLAB Simulink results and discussion on the obtained results for flight controller system. This results are in sort of nonlinear response, linearized response, continuous time LQG controller response and discrete time LQG controller responses for steady Wing level flight with consideration of both high and low fidelity flight condition of F-16/MATV flight system.

The fifth chapter will discuss about the conclusion and recommendation of the project. The recommendations suggested are proposed with the intention of improving the project in future.

CHAPTER TWO

2. Literature Review

Control system performance models have long been popular in the field of automatic control. The idea of using a feedback element to ensure that the loop gain was always sufficiently high to guarantee rapid, though stable, dynamic response, and to provide insensitivity to both external disturbances and internal parameter variations, has always been dominant [6].

Considering the feedback linearization approach, one needs to satisfy certain conditions and often they are difficult to achieve in real life control problems. Feedback linearization [7] and [8] is a nonlinear design method that can explicitly handle systems with known nonlinearities. Nonlinear dynamic inversion (NDI) is a special form of FBL especially suited for flight control applications; see [[9], and [10]]. The main idea behind NDI is to use an accurate model of the system to cancel all system nonlinearities in such a way that a single linear system valid over the entire flight envelope remains

By using nonlinear feedback and exact state transformations rather than linear approximations, the nonlinear system is transformed into a constant linear system. Just a single linear controller can in principle control this linear system. This is generally not the case with modern fighter aircraft, because it is very difficult to precisely know and model their complex nonlinear aerodynamic characteristics.

Presented herein are some results in the area of time-domain design of model-reference adaptive control systems using LYAPUNOV's direct method [11]. The time-varying model-reference adaptive control problem is formulated precisely and the adaptation technique is derived analytically.

The use of a LYAPUNOV function in the design of a model reference adaptive control system by Parks has leads to a proliferation of this thesis [12]. Winsor and Roy designed a controller in which the adaptive parameter rates were proportional to the product of the error and plant states [13].

On other hand, adaptive control design approaches have been researched for the Flight Control System. Some of them are based on the Direct Adaptive approach (DAC) and others are based on adaptive control approaches [14], [15], and [16]. However, a problem of this approach is that good performance and robustness properties cannot be guaranteed for a highly nonlinear fighter aircraft.

Linear Quadratic Gaussian (LQG) control is a modern state space technique for designing optimal dynamic regulators. It enables you to trade off regulation performance and control effort, and to take into account process and measurement noise. Like pole placement, LQG design requires a state-space model of the plant. There are several attempts by contemporary researchers to design efficient control scheme for flight control system [4], [17], and [18] using LQG controller.

Some limitations of the research must be acknowledged:

- One drawback of feedback linearization control is that not all system states can be transformed into a linearized form.
- In adaptive control approaches the permissible value of adaptive gain dependence on reference signal type and amplitude.
- Flight Control system using predominantly direct adaptive control theory often difficult in analysis of stability or robustness and performance.

CHAPTER THREE

3. System Dynamics and Controller design

3.1. F-16/ MATV Dynamics

One of the most recent developments in aerospace technology, the performance requirements of modern fighter aircraft became more and more challenging throughout an ever-increasing flight envelope. Extreme maneuverability is achieved by designing the aircraft with thrust-vectored control.

The flight dynamics is based on the non-linear F-16 model with MATV Multi – Axis – Thrust – Vectoring which was created by using aircraft specific data, such as the aircraft geometry and the aerodynamic model obtained from [19]. The F-16/MATV is a single-seat, supersonic, multi-role tactical aircraft with a blended wing fuselage that has been in production since 1976. Over 4,400 have been produced for 24 countries, making it the most common fighter type in the world.

The equation of motion for a general rigid body aircraft, the control variables and the engine model of the F-16/MATV, the geometry and the aerodynamic data are shown below.

3.1.1. Equation of Motion

Before derivation of equation of motion the following assumptions and some frames of reference are needed to be described. The assumptions made are:

- The aircraft is a rigid-body, which means that any two points on or within the airframe remain fixed with respect to each other.
- The earth is flat, non-rotating, and regarded as an inertial reference. This assumption is valid when dealing with controller design of aircraft, but not when analyzing inertial guidance systems.
- Wind gust effects are not taken into account, hence the undisturbed air is assumed to be at rest with respect to the surface of the earth. In other words, the kinematic velocity is equal to the aerodynamic velocity of the aircraft.

- The mass is constant during the time interval over which the motion is considered, the fuel consumption is neglected during this time-interval. This assumption is necessary to apply Newton's motion laws.
- The mass distribution of the aircraft is symmetric relative to the X_BOZ_B -plane, this implies that the products of inertia I_{yz} and I_{xy} are equal to zero. This assumption is valid for most aircraft.

The reference frames used in this final year thesis are:

- The earth-fixed reference frame F_E, used as the inertial frame and the vehicle carried local earth reference frame F_O with its origin fixed in the center of gravity of the aircraft which is assumed to have the same orientation as FE;
- The wind-axes reference frame FW , obtained from FO by three successive rotations of flight path heading angle χ, flight path climb angle γ and aerodynamic bank angle μ;
- The stability-axes reference frame FS , obtained from FW by a rotation of minus sideslip angle β; and
- Finally the body-fixed reference frame FB, obtained from FS by a rotation of angle of attack α.

Under the above assumptions and reference the motion of the aircraft has six degrees of freedom (rotation and translation in three dimensions). The aircraft dynamics can be described by its position, orientation, velocity and angular velocity over time. $p_E = (x_E, y_E, z_E)^T$ is the position vector expressed in an earth-fixed coordinate system. V is the velocity vector given by $V = (u, v, w)^T$, where u is the longitudinal velocity, v the lateral velocity and w the normal velocity. The orientation vector is given by $\Phi = (\varphi, \theta, \psi)^T$, where φ is the roll angle, θ the pitch angle and ψ the yaw angle, and the angular velocity vector is given by $\omega = (p, q, r)^T$, where p, q and r are the roll, pitch and yaw angular velocities, respectively. Various components of the aircraft motions are illustrated in Figure 3.

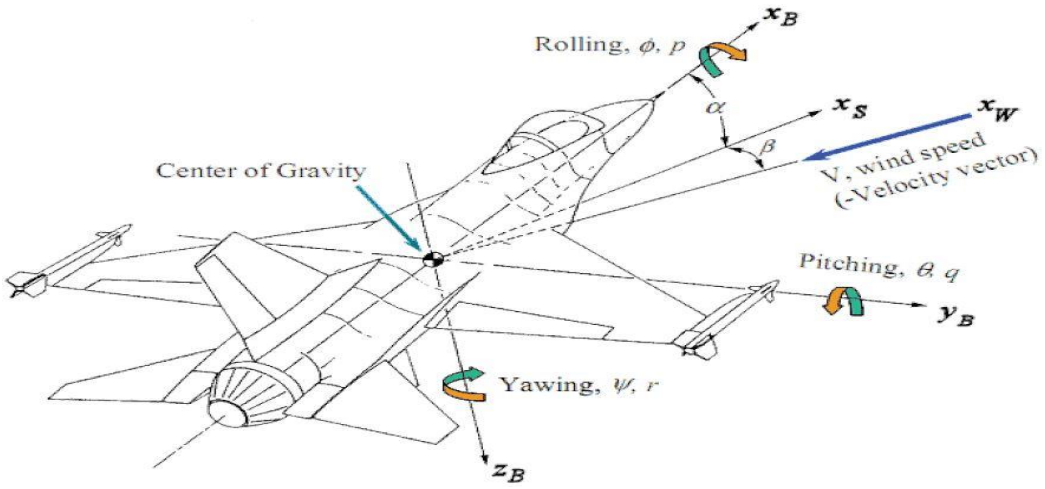

Figure 3: Aircraft orientation angles φ, θ and ψ, aerodynamic angles α and β, and the angular rates p, q and r

The relation between the attitude vector Φ and the angular velocity vector ω is given as

$$\dot{\Phi} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \omega \qquad (1.)$$

Defining V_T as the total velocity and using Figure 3, the following relations can be derived:

$$V_T = \sqrt{u^2 + v^2 + w^2}$$

$$\alpha = arctan\frac{w}{u} \qquad (2.)$$

$$\beta = arcsin\frac{v}{V_T}$$

Furthermore, when $\beta = \varphi = 0$, the flight path angle γ can be defined as

$$\gamma = \theta - \alpha \tag{3.}$$

The equations of motion for the aircraft can be derived from Newton's Second Law of motion, which states that the summation of all external forces acting on a body must be equal to the time rate of change of its momentum, and the summation of the external moments acting on a body must be equal to the time rate of change of its angular momentum. In the inertial, earth-fixed reference frame F_E , Newton's Second Law can be expressed by two vector equations [20]

$$\boldsymbol{F} = \left.\frac{d\left(m\boldsymbol{V}\right)}{dt}\right|_B \tag{4.}$$

$$\boldsymbol{M} = \left.\frac{d\boldsymbol{H}}{dt}\right|_B \tag{5.}$$

Where F represents the sum of all externally applied forces, m is the mass of the aircraft, V is the velocity vector, M represents the sum of all applied torques and H is the angular momentum.

To obtain the expression for the time rate of change of the velocity with respect to earth, the force equation can be further written as

$$\boldsymbol{F} = \left.\frac{d}{dt}\left(m\boldsymbol{V}\right)\right|_B + \omega \times m\boldsymbol{V} \tag{6.}$$

Where ω is the total angular velocity of the aircraft with respect to the earth (inertial reference frame). The velocity components with respect to body – fixed reference frame $\mathbf{F_B}$ is

$$\boldsymbol{V} = \boldsymbol{i}u + \boldsymbol{j}v + \boldsymbol{k}\omega \tag{7.}$$

$$\boldsymbol{\omega} = \boldsymbol{i}p + \boldsymbol{j}q + \boldsymbol{k}r \tag{8.}$$

Where **i**, **j** and **k** are unit vectors along the aircraft's $\mathbf{X_B}$, $\mathbf{Y_B}$ and $\mathbf{Z_B}$ axes, respectively. Expanding (6.) using (7.) *and* (8.) resulting in

$$\boldsymbol{F_x} = m\left(\dot{u} + q\omega - rv\right)$$

$$\boldsymbol{F_y} = m\left(\dot{v} + ru - p\omega\right) \tag{9.}$$

$$\boldsymbol{F_z} = m\left(\dot{\omega} + pv - qu\right)$$

Where the external forces F_x, F_y and F_z depends on the weight vector W, the aerodynamic force vector R and the thrust vector E. assuming the thrust produced by the engine, FT acts in parallel to the aircraft's XB – axis, then the thrust vector **E** becomes

$$\boldsymbol{E_x} = F_T = T\cos\delta_{ty}\cos\delta_{tz}$$

$$\boldsymbol{E_y} = -T\cos\delta_{ty}\cos\delta_{tz} \tag{10.}$$

$$\boldsymbol{E_z} = -T\sin\delta_{ty}$$

The components of W and R along the body – axes are

$$\boldsymbol{W_x} = -mg\sin\theta$$

$$\boldsymbol{W_y} = mg\sin\varnothing\cos\theta \tag{11.}$$

$$\boldsymbol{W_z} = mg\cos\varnothing\cos\theta$$

And

$$\boldsymbol{R_x} = \bar{X}$$

$$\boldsymbol{R_y} = \bar{Y} \tag{12.}$$

$$\boldsymbol{R_z} = \bar{Z}$$

Where, g is the gravitational constant. The size of aerodynamic forces $\bar{X}$, $\bar{Y}$ and $\bar{Z}$ is determined by the amount of air diverted by the aircraft in different directions. In the standard way the aerodynamic force can be modeled as

$$\bar{X} = \bar{q}SC_{X_T}\left(\alpha,\beta,p,q,r,\delta,\cdots\right)$$

$$\bar{Y} = \bar{q}SC_{Y_T}\left(\alpha,\beta,p,q,r,\delta,\cdots\right) \qquad (13.)$$

$$\bar{Z} = \bar{q}SC_{Z_T}\left(\alpha,\beta,p,q,r,\delta,\cdots\right)$$

Where $\bar{q} = \frac{1}{2}\rho V_T^2$ is the aerodynamic pressure, The coefficients C_{X_T}, C_{Y_T} and C_{Z_T} are usually obtain from wind tunnel data and flight tests.

Combining equation $(12.)$ and $(13.)$ and the thrust component $(10.)$ and $(11.)$, results in the complete body – axes force equation:

$$\bar{X} + T\cos\delta_{ty}\cos\delta_{tz} - mg\sin\theta = \bar{q}SC_{X_T}\left(\alpha,\beta,p,q,r,\delta,\cdots\right)$$

$$\bar{Y} - T\cos\delta_{ty}\cos\delta_{tz} + mg\sin\varnothing\cos\theta = \bar{q}SC_{Y_T}\left(\alpha,\beta,p,q,r,\delta,\cdots\right) \qquad (14.)$$

$$\bar{Z} - T\sin\delta_{ty} + mg\cos\varnothing\cos\theta = \bar{q}SC_{Z_T}\left(\alpha,\beta,p,q,r,\delta,\cdots\right)$$

To obtain the equation of angular motion, the equation of applied torque $(5.)$ can be written as

$$\boldsymbol{M} = \left.\frac{d\boldsymbol{H}}{dt}\right|_B + \omega\times\boldsymbol{H} \qquad (15.)$$

In the body – fixed reference frame, under assumptions 1 and 3, the angular momentum which can change in magnitude and direction, can be written as

$$\boldsymbol{H} = \boldsymbol{I\omega} \qquad (16.)$$

Where under assumption 4, the inertia matrix is defined as

$$\boldsymbol{I} = \begin{bmatrix} I_x & 0 & -I_{xz} \\ 0 & I_y & 0 \\ -I_{xz} & 0 & I_z \end{bmatrix} \tag{17.}$$

Expansion of equations (16.) and (17.) results in

$$\begin{aligned} \boldsymbol{M_x} &= \dot{p}I_x - \dot{r}I_{xz} + qr\left(I_z - I_y\right) - pI_{xz} \\ \boldsymbol{M_y} &= \dot{q}I_y + pq\left(I_x - I_z\right) - \left(p^2 - r^2\right)I_{xz} \\ \boldsymbol{M_z} &= \dot{r}I_z - \dot{p}I_{xz} + pq\left(I_y - I_x\right) - prI_{xz} \end{aligned} \tag{18.}$$

The external Moments $\boldsymbol{M_x}$, $\boldsymbol{M_y}$ and $\boldsymbol{M_z}$ are those due to aerodynamics and thrust. As a result, the aerodynamic moments became

$$\begin{aligned} \boldsymbol{M_x} &= \bar{L} + L_T \\ \boldsymbol{M_y} &= \bar{M} + M_T \\ \boldsymbol{M_z} &= \bar{N} + N_T \end{aligned} \tag{19.}$$

Where $\bar{L}$, $\bar{M}$ and $\bar{L}$ are the aerodynamic moments and L_T, $M_T$and $N_T$are the moments due to thrust. The aerodynamic moments can also be expressed as

$$\begin{aligned} \bar{L} &= \bar{q}SbC_{l_T}\left(\alpha, \beta, p, q, r, \delta, \cdots\right) \\ \bar{M} &= \bar{q}S\bar{c}C_{m_T}\left(\alpha, \beta, p, q, r, \delta, \cdots\right) \end{aligned} \tag{20.}$$

$$\bar{N} = \bar{q}SbC_{n_T}\left(\alpha, \beta, p, q, r, \delta, \cdots\right)$$

Assuming the case if the engine is mounted so that the thrust points lies in the xz-plane of the body – frame reference frame with an offset from the center of gravity by tz along the z – axis, moment due to thrust are defined as

$$L_T = 0$$

$$M_T = F_{T^2T} \tag{21.}$$

$$N_T = 0$$

Combining equation (19.) and (20.), the complete body – axis moment becomes

$$\boldsymbol{M_x} = \bar{L} + L_T = \dot{p}I_x - \dot{r}I_{xz} + qr\left(I_z - I_y\right) - pI_{xz}$$

$$\boldsymbol{M_y} = \bar{M} + M_T = \dot{q}I_y + pq\left(I_x - I_z\right) - \left(p^2 - r^2\right)I_{xz} \tag{22.}$$

$$\boldsymbol{M_z} = \bar{N} + N_T = \dot{r}I_z - \dot{p}I_{xz} + pq\left(I_y - I_x\right) - prI_{xz}$$

The following systems of twelve scalar first order differential equation of motion are used in this project.

$$\dot{u} = rv - qw - g\sin\theta + \frac{1}{m}\left(\bar{X} + T\cos\delta_{ty}\cos\delta_{tz}\right) \tag{23.}$$

$$\dot{v} = pw - ru - g\sin\varnothing\cos\theta + \frac{1}{m}\left(\bar{Y} - T\cos\delta_{ty}\cos\delta_{tz}\right) \tag{24.}$$

$$\dot{\omega} = qu - pv - g\cos\varnothing\cos\theta + \frac{1}{m}\left(\bar{Z} - T\sin\delta_{tz}\right) \tag{25.}$$

$$\dot{p} = (c_1 r + c_2 p) q + c_3 (\bar{L} + T\left(Z_T \cos\delta_{ty} \sin\delta_{tz}\right) + c_4 \left(\bar{N} - T\left(Z_T \cos\delta_{ty} \sin\delta_{tz}\right) + qH_{eng}\right) \quad (26.)$$

$$\dot{q} = c_5 pr - c_6 \left(p^2 - r^2\right) + c_7 \left(\bar{M} + T\left(-X_T \sin\delta_{ty} + Z_T \cos\delta_{ty} \cos\delta_{tz}\right) - rH_{eng}\right) \quad (27.)$$

$$\dot{r} = (c_8 p - c_2 r) q - c_4 \left(\bar{L} + TZ_T \cos\delta_{ty} \sin\delta_{tz}\right) + c_9 \left(\bar{N} - TX_T \cos\delta_{ty} \sin\delta_{tz} - qH_{eng}\right) \quad (28.)$$

$$\dot{\varnothing} = p + \tan\theta\left(q \sin\phi + r \cos\phi\right) \quad (29.)$$

$$\dot{\theta} = q \cos\phi - r \sin\phi \quad (30.)$$

$$\dot{\psi} = \frac{q \sin\phi + r \cos\phi}{\cos\theta} \quad (31.)$$

$$\dot{x}_E = u\cos\psi \sin\theta + v\left(\cos\psi \sin\theta \sin\phi - \sin\psi \cos\phi\right) + \omega\left(\cos\psi \sin\theta \cos\phi + \sin\psi \sin\phi\right) \quad (32.)$$

$$\dot{y}_E = u\sin\psi \cos\theta + v\left(\sin\psi \sin\theta \sin\phi - \cos\psi \cos\phi\right) + \omega\left(\sin\psi \sin\theta \cos\phi + \cos\psi \sin\phi\right) \quad (33.)$$

$$\dot{z}_E = -u \sin\theta + v \cos\theta \sin\phi + \omega \cos\theta \cos\phi \quad (34.)$$

Where, the Moment of Inertias can be

$$\Gamma_{c_1} = \left(I_y - I_z\right) I_z - I_{xz}^2 \qquad \Gamma_{c_2} = (I_x - I_y + I_z) I_x z - I_x z^2$$

$$\Gamma_{c_3} = I_z \qquad \Gamma_{c_4} = I_{xz} \qquad c_5 = \frac{I_z - I_x}{I_y} \quad (35.)$$

$$c_6 = \frac{I_{xz}}{I_y} \qquad c_7 = \frac{1}{I_y} \qquad \Gamma_{c_8} = I_x \left(I_x - I_y\right) + I_{xz}^2 \qquad \Gamma_{c_9} = I_x$$

With $\Gamma = I_x I_z + I_{xz}^2$

3.1.2. Quaternions

When the pitch angle θ, pass through$\pm \pi/2$, a singularity will occur at $\dot{p}$ and $\dot{r}$ since it uses the Euler's angle method to solve this a quaternions was applied for aircrafts orientation presentation.

With the applied quaternions, the equation of motion in body - fixed reference frame can be written as:

$$\dot{u} = rv - qw - g\sin\theta + \frac{1}{m}\left(\bar{X} + T\cos\delta_{ty}\cos\delta_{tz}\right) + 2\left(q_1 q_3 - q_0 q_2\right)g$$

$$\dot{v} = pw - ru - g\sin\varnothing\cos\theta + \frac{1}{m}\left(\bar{Y} - T\cos\delta_{ty}\cos\delta_{tz}\right) + 2\left(q_2 q_3 - q_0 q_1\right)g \qquad (36.)$$

$$\dot{\omega} = qu - pv - g\cos\varnothing\cos\theta + \frac{1}{m}\left(\bar{Z} - T\sin\delta_{tz}\right) + \left({q_0}^2 + {q_1}^2 - {q_2}^2 + {q_3}^2\right)g$$

Where, m is Airplane mass, q, p, $and\ r$ are the angular rates in body axes, g represents the gravity, and q_i are quaternion components. The aerodynamic forces in body reference frame are represented by $\bar{X}, \bar{Y}, and\ \bar{Z}$, T is the engine thrust ,and $\delta_{tx}\ and\ \delta_{tz}$ represents the thrust vectoring deflections in $X_B OZ_B$ and $X_B OY_B$ – plane respectively.

The total airspeed V_T, angle of attack α and angle of sideslip β are redefined through trigonometric relation. Which are obtained from the above ordinary differential equation, when incorporation of aerodynamic force and moments, it is convenient that the state variables derivative of $(\dot{u})$, $(\dot{v})$, $(\dot{w})$, $(\dot{p})$, $(\dot{q})$, and $(\dot{r})$ are replaced with $(\dot{V_T}), (\dot{\beta})$, and $(\dot{\alpha})$ their respective equations can be:

$$V_T = \sqrt{u^2 + v^2 + \omega^2} \Rightarrow \dot{V_T} = \frac{u\dot{u} + v\dot{v} + \omega\dot{\omega}}{V_T}$$

$$\beta = arcsin\frac{v}{V_T} \Rightarrow \dot{\beta} = \frac{\dot{v}V_T - v\dot{V_T}}{{V_T}^2 \cos\beta} \tag{37.}$$

$$\alpha = arctan\frac{\omega}{u} \Rightarrow \dot{\alpha} = \frac{u\dot{\omega} - \omega\dot{u}}{u^2 + \omega^2}$$

The time derivative of angular rates, Roll rate ṗ, Pitch rate q̇ and Yaw rate ṙ in bo body - fixed reference frame can be written as:

$$\dot{p} = \left(c_1 r + c_2 p\right)q + c_3(\bar{L} + T\left(Z_T \cos\delta_{ty} \sin\delta_{tz}\right) + c_4\left(\bar{N} - T\left(Z_T \cos\delta_{ty} \sin\delta_{tz}\right) + qH_{eng}\right)$$

$$\dot{q} = c_5 pr - c_6\left(p^2 - r^2\right) + c_7\left(\bar{M} + T\left(-X_T \sin\delta_{ty} + Z_T \cos\delta_{ty} \cos\delta_{tz}\right) - rH_{eng}\right) \tag{38.}$$

$$\dot{r} = \left(c_8 p - c_2 r\right)q - c_4\left(\bar{L} + TZ_T \cos\delta_{ty} \sin\delta_{tz}\right) + c_9\left(\bar{N} - TX_T \cos\delta_{ty} \sin\delta_{tz} - qH_{eng}\right)$$

Where, $\bar{L}, \bar{M}, and\ \bar{N}$ are the aerodynamic moments, $X_T\ and\ Z_T$ define the distance of engine nozzle with respect to center of gravity, and c_i are coefficients of inertia as defined above in equation (35.).

The relation between Quaternions component and the Euler angles was defined as:

$$\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos\phi/2\cos\theta/2\cos\psi/2 + \sin\phi/2\sin\theta/2\sin\psi/2 \\ \sin\phi/2\cos\theta/2\cos\psi/2 - \cos\phi/2\sin\theta/2\sin\psi/2 \\ \cos\phi/2\sin\theta/2\cos\psi/2 + \sin\phi/2\cos\theta/2\sin\psi/2 \\ \cos\phi/2\cos\theta/2\sin\psi/2 - \sin\phi/2\sin\theta/2\cos\psi/2 \end{bmatrix} \tag{39.}$$

Where, ϕ is Roll Angle, θ is Pitch Angle, and ψ is Yaw angle of Airplane. The time rate of change in Quaternion is given by:

$$\dot{q} = \begin{bmatrix} \dot{q_0} \\ \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{40.}$$

The time rate of change of position of Airplane in the Earths Fixed reference frame was obtained by transforming the Airplane's velocity in body axes to velocities in Earth reference frame:

$$\begin{bmatrix} \dot{x}_E \\ \dot{y}_E \\ \dot{z}_E \end{bmatrix} = \begin{bmatrix} \left(q_0{}^2 + q_1{}^2 - q_2{}^2 - q_3{}^2\right) & 2\left(q_1 q_2 - q_0 q_3\right) & 2\left(q_1 q_3 + q_0 q_2\right) \\ 2\left(q_1 q_2 + q_0 q_3\right) & \left(q_0{}^2 - q_1{}^2 + q_2{}^2 - q_3{}^2\right) & 2\left(q_2 q_3 - q_0 q_1\right) \\ 2\left(q_1 q_3 - q_0 q_2\right) & 2\left(q_2 q_3 + q_0 q_1\right) & \left(q_0{}^2 - q_1{}^2 - q_2{}^2 + q_3{}^2\right) \end{bmatrix} \begin{bmatrix} u \\ v \\ \omega \end{bmatrix} \tag{41.}$$

3.1.3. Control Variables

The F-16/MATV model allows six control variables: the engine thrust T, aileron deflection $\boldsymbol{\delta_a}$, elevator deflection $\boldsymbol{\delta_e}$, rudder deflection $\boldsymbol{\delta_r}$, and the thrust vector deflections in horizontal and vertical directions $\boldsymbol{\delta_{ty}}$ and $\boldsymbol{\delta_{tz}}$. All deflections are defined positive in the conventional manner, i.e. positive thrust results in an acceleration increase along the XB-axis, a positive elevator deflection results in a decrease of pitch rate. The engine nozzle deflections are defined positive downwards and to the left.

The F-16/MATV has two other control surfaces, which will not be used for the controller synthesis: the leading edge flap and the speed brake. The leading edge flap is assumed to deflect according to the transfer function defined in reference [21].

$$\boldsymbol{\delta_{lef}} = 1.38\frac{2\boldsymbol{s} + 7.25}{\boldsymbol{s} + 7.25}\boldsymbol{\alpha} - 9.05\frac{\bar{\boldsymbol{q}}}{\boldsymbol{p_s}} + 1.45 \tag{42.}$$

Where, $\boldsymbol{\delta_{lef}}$ is the leading edge flap deflection, $\boldsymbol{\bar{q}}$ is the dynamic pressure, and $\boldsymbol{p_s}$ the static pressure. The control actuators, i.e. the aircraft servos, are assumed to be ideal. The bounds on the control variables are defined in **Table 1**.

Table 1 The control input units and maximum values

Control	Units	Minimum	Maximum	Rate Limit
Engine Thrust T	N	100	100000	$\pm 40000\ N/Sec$
Elevator $\boldsymbol{\delta_e}$	Deg	-25	25	$\pm 60\ Deg/Sec$
Ailerons $\boldsymbol{\delta_a}$	Deg	-21.5	21.5	$\pm 80\ Deg/Sec$
Rudder $\boldsymbol{\delta_r}$	Deg	-30	30	$\pm 120\ Deg/Sec$
Thrust Vector xz – frame $\boldsymbol{\delta_{ty}}$	Deg	-15	15	$\pm 50\ Deg/Sec$
Thrust Vector xy – frame $\boldsymbol{\delta_{tz}}$	Deg	-15	15	$\pm 50\ Deg/Sec$

3.1.4. Aerodynamic data

The aerodynamic data used in the F-16/MATV model has been derived from low-speed static and dynamic wind tunnel tests conducted with sub-scale models of the F-16 in wind-tunnel facilities at the NASA Ames and Langley Research Centers [22]. The aerodynamic data in reference [21] is given in tabular form and is valid for the following flight envelope:

- $-20 \leq \alpha \leq 90$ degrees;
- $-30 \leq \beta \leq 30$ degrees;
- $0.1 \leq$ Mach ≤ 0.6.

For the X – axis Force Coefficient:

$$C_{X,t} = C_X(\alpha,\beta,\delta_h) + \Delta C_{X,lef}\left(1-\frac{\delta_{lef}}{25}\right) + \Delta C_{X,sb}(\alpha)\left(\frac{\delta_{sb}}{60}\right) + \frac{q\bar{c}}{2V}\left[C_{X_q}(\alpha) + \Delta C_{X_{q,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right] \quad (43.)$$

Where, $\Delta C_{X,lef} = C_{X,lef}(\alpha,\beta) - C_X(\alpha,\beta,\delta_h = 0)$

For the Z – axis Force Coefficient:

$$C_{Z,t} = C_Z(\alpha,\beta,\delta_h) + \Delta C_{Z,lef}\left(1-\frac{\delta_{lef}}{25}\right) + \Delta C_{Z,sb}(\alpha)\left(\frac{\delta_{sb}}{60}\right) + \frac{q\bar{c}}{2V}\left[C_{Z_q}(\alpha) + \Delta C_{Z_{q,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right] \quad (44.)$$

Where, $\Delta C_{Z,lef} = C_{Z,lef}(\alpha,\beta) - C_Z(\alpha,\beta,\delta_h = 0)$

For the Pitching - Moment Coefficient:

$$C_{M,t} = C_M(\alpha,\beta,\delta_h)\eta_{\delta_h}(\delta_h) + C_{Z,t}\left(X_{cg,ref} - X_{cg}\right) + \Delta C_{M,lef}\left(1-\frac{\delta_{lef}}{25}\right) + \Delta C_{M,sb}(\alpha)\left(\frac{\delta_{sb}}{60}\right) + \frac{q\bar{c}}{2V}\left[C_{M_q}(\alpha) + \Delta C_{M_{q,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right] + \Delta C_M(\alpha) + \Delta C_{M,ds}(\alpha,\delta_h) \quad (45.)$$

Where, $\Delta C_{M,lef} = C_{M,lef}(\alpha,\beta) - C_M(\alpha,\beta,\delta_h = 0)$

For the Y – axis Force Coefficient:

$$C_{Y,t} = C_Y(\alpha,\beta) + \Delta C_{Y,lef}\left(1-\frac{\delta_{lef}}{25}\right) + \left[\Delta C_{Y,\delta_{a=20^0}} + \Delta C_{Y,\delta_{a=20^0,lef}}\left(1-\frac{\delta_{lef}}{25}\right)\right]\frac{\delta_a}{20} + \Delta C_{Y,\delta_{r=30^0}}\left(\frac{\delta_r}{30}\right) + \frac{b}{2V}\left\{\left[C_{Y_r}(\alpha) + \Delta C_{Y_{r,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right]r + \left[C_{Y_q}(\alpha) + \Delta C_{Y_{p,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right]p\right\} \quad (46.)$$

Where, $\Delta C_{Y,lef} = C_{Y,lef}(\alpha,\beta) - C_Y(\alpha,\beta)$

$\Delta C_{Y,\delta_{a=20^0}} = C_{Y,\delta_{a=20^0}}(\alpha,\beta) - C_Y(\alpha,\beta)$

$$\Delta C_{Y,\delta_{a=20^0,lef}} = C_{Y,\delta_{a=20^0,lef}}(\alpha,\beta) - C_{Y,lef}(\alpha,\beta) - \left[C_{Y,\delta_{a=20^0,lef}}(\alpha,\beta) - C_Y(\alpha,\beta) \right]$$

$$\Delta C_{Y,\delta_{r=30^0}} = C_{Y,\delta_{r=30^0}}(\alpha,\beta) - C_Y(\alpha,\beta)$$

For the Yawing – Moment Coefficient:

$$\begin{aligned} C_{n,t} &= C_n(\alpha,\beta,\delta_h) + \Delta C_{n,lef}\left(1-\frac{\delta_{lef}}{25}\right) - C_{Y,t}\left(X_{cg,ref} - X_{cg}\right)\frac{\bar{c}}{b} \\ &+ \left[\Delta C_{n,\delta_{a=20^0}} + \Delta C_{n,\delta_{a=20^0,lef}}\left(1-\frac{\delta_{lef}}{25}\right)\right]\frac{\delta_a}{20} + \Delta C_{n,\delta_{r=30^0}}\left(\frac{\delta_r}{30}\right) \\ &+ \frac{b}{2V}\left\{\left[C_{n_r}(\alpha) + \Delta C_{n_{r,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right]r + \left[C_{n_p}(\alpha) + \Delta C_{n_{p,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right]p\right\} \\ &+ \Delta C_{n_\beta}(\alpha)\beta \end{aligned} \qquad (47.)$$

Where, $\Delta C_{n,lef} = C_{n,lef}(\alpha,\beta) - C_n(\alpha,\beta,\delta_h = 0)$

$$\Delta C_{n,\delta_{a=20^0}} = C_{n,\delta_{a=20^0}}(\alpha,\beta) - C_n(\alpha,\beta,\delta_h = 0)$$

$$\Delta C_{n,\delta_{a=20^0,lef}} = C_{n,\delta_{a=20^0,lef}}(\alpha,\beta) - C_{n,lef}(\alpha,\beta) - \left[C_{n,\delta_{a=20^0,lef}}(\alpha,\beta) - C_n(\alpha,\beta,\delta_h = 0) \right]$$

$$\Delta C_{n,\delta_{r=30^0}} = C_{n,\delta_{r=30^0}}(\alpha,\beta) - C_n(\alpha,\beta,\delta_h = 0)$$

For the Rolling – Moment Coefficient:

$$\begin{aligned} C_{l,t} &= C_l(\alpha,\beta,\delta_h) + \Delta C_{l,lef}\left(1-\frac{\delta_{lef}}{25}\right) + \left[\Delta C_{l,\delta_{a=20^0}} + \Delta C_{l,\delta_{a=20^0,lef}}\left(1-\frac{\delta_{lef}}{25}\right)\right]\frac{\delta_a}{20} + \Delta C_{l,\delta_{r=30^0}}\left(\frac{\delta_r}{30}\right) \\ &+ \frac{b}{2V}\left\{\left[C_{l_r}(\alpha) + \Delta C_{l_{r,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right]r + \left[C_{l_p}(\alpha) + \Delta C_{l_{p,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right]p\right\} \\ &+ \Delta C_{l_\beta}(\alpha)\beta \end{aligned} \qquad (48.)$$

Where, $\Delta C_{l,lef} = C_{l,lef}(\alpha,\beta) - C_l(\alpha,\beta,\delta_h = 0)$

$$\Delta C_{l,\delta_{a=20^0}} = C_{l,\delta_{a=20^0}}(\alpha,\beta) - C_l(\alpha,\beta,\delta_h = 0)$$

$$\Delta C_{l,\delta_{a=20^0,lef}} = C_{l,\delta_{a=20^0,lef}}(\alpha,\beta) - C_{l,lef}(\alpha,\beta) - \left[C_{l,\delta_{a=20^0,lef}}(\alpha,\beta) - C_l(\alpha,\beta,\delta_h = 0) \right]$$

$$\Delta C_{l,\delta_{r=30^0}} = C_{l,\delta_{r=30^0}}(\alpha,\beta) - C_l(\alpha,\beta,\delta_h = 0)$$

3.1.5. State Variables

The non – linear F-16/MATV plant state variables considered are: Total Velocity(V_T) angle of attack(α), sideslip angle(β), roll angle(∅), pitch angle(θ), yaw angle(ψ), angular velocity vector components roll rate(p), pitch rate(q) and yaw rate(r), Navigation vectors North Position x_E, East position y_E, and Altitude z_E.

$$x^T = \begin{bmatrix} V_T & \alpha & \beta & \phi & \theta & \psi & p & q & r & x_E & y_E & z_E & a_{nx} & a_{ny} & a_{nz} & M & \bar{q} & Ps \end{bmatrix} \quad (49.)$$

In this flight controller design, final year thesis, the control variables is rewritten as:

$$u^T = \begin{bmatrix} T & \delta_a & \delta_e & \delta_r & \delta_{ty} & \delta_{tz} \end{bmatrix} \quad (50.)$$

Where Engine thrust(T), aileron deflection(δ_a), elevator deflection(δ_e), rudder deflection(δ_r), horizontal thrust vectoring deflection(δ_{ty}), and vertical thrust vectoring deflection(δ_{tz}).

The plant outputs included are the state derivatives and normalized accelerations in the x, y, and z – direction($a_{nx,}\ a_{ny,}\ and\ a_{nz,}\ respectively$), Mach number($M$), free – stream dynamic pressure($\bar{q}$), and static pressure(p_s).

$$y^T = \begin{bmatrix} \dot{x}^T \end{bmatrix} \quad (51.)$$

The following table depicts the state variables used with their unit measurement used for F-16/MATV model in this project.

Table 2 The units of state variables used for F-16/MATV Model

State Symbol	State Name	Unit of State
x_E	North Position	ft
y_E	East Position	ft
z_E	Altitude	ft
q	Pitch Angle	rad
r	Yaw Angle	rad
p	Roll Angle	rad
V_T	Total Velocity	ft/sec
α	Angle of Attack	rad
β	Angle of Sideslip	rad
$\dot{p}$	Roll Rate	rad/sec
$\dot{q}$	Pitch Rate	rad/sec
$\dot{r}$	Yaw Rate	rad/sec
a_{nx}	Normal Acceleration in X - Axis	N/A
a_{ny}	Normal Acceleration in Y – Axis	N/A
a_{nz}	Normal Acceleration in Z – Axis	N/A
M	Match Number	N/A
$\bar{q}$	Free Stream Dynamic Pressure	N/A
p_s	Static Pressure	N/A

In addition, some assumptions have been made for the F-16/MATV model:

I. The elevators only have symmetric deflection. Although differential elevator deflection is possible for the F-16 (and indeed desirable at flight conditions with high dynamic pressure), no data was available for its implementation.

II. Speed brakes are not used.

III. The engine does not cause any gyroscopic effects (no contribution of angular momentum to the aircraft's motion).

The nonlinear F-16/MATV flight dynamics is implemented in MATLAB. The following figure shows the overall structures of nonlinear F-16/MATV flight dynamics.

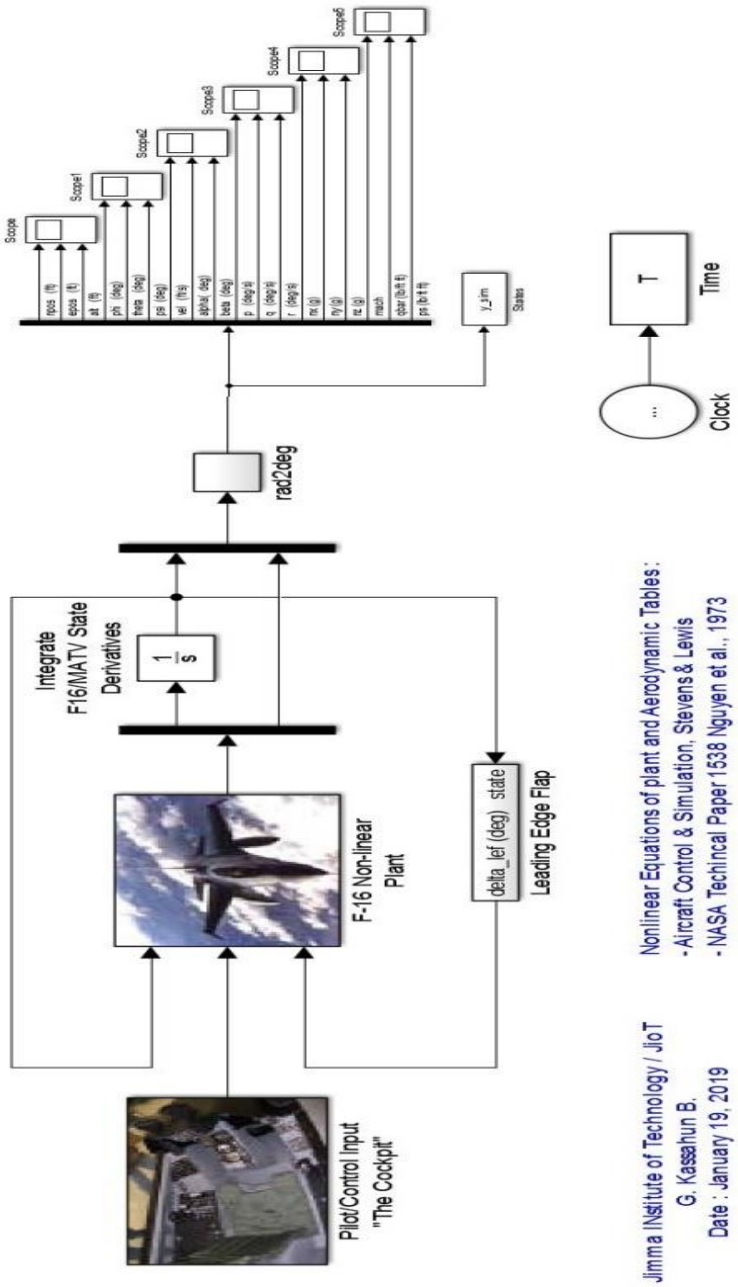

Figure 4 General Structure of Nonlinear F-16/MATV dynamics

3.2. Feedback Linearization

Feedback linearization is a technique to algebraically transform a nonlinear set of equations in to a fully or partially linearized set. The main advantage of this technique is that linear controller synthesis methods can be used. The disadvantages on the other hand are that it depends on the system's point of the exact cancellation of non-linearity, and its sensitivity to modeling errors and the possibility of generating extremely large control inputs.

For introducing of feedback linearization, the above implicit non – linear ordinary differential equations from equation (23) to equation (34) can be modified as:

$$\begin{aligned} \dot{X} &= f_1(X \quad U) = 0 \\ \dot{X} &= f_2(X \quad U) = 0 \\ &\vdots \\ \dot{X} &= f_{12}(X \quad U) = 0 \end{aligned} \tag{52.}$$

By considering the perturbation from the steady state condition (x_e, u_e) and a set of linear constant-coefficient state equations are derived. If the above nonlinear state equations are expanded with a Taylor series about the steady state condition (x_e, u_e) and keep only the first order terms, the perturbations in the state, state derivative, and control vectors is satisfied.

Then, at the steady state condition, a new state is defined with perturbation, which transforms the above nonlinear differential equation in to the required form.

$$x = (x_e + \Delta x\ , u_e + \Delta u) \tag{53.}$$

A Tayler series expansion was introduced at the steady state point

$$\dot{x} = f(x \quad u) = 0$$

$$f(x_e + \Delta x\ , u_e + \Delta u) = f(x_e\ , u_e) + \left.\frac{\delta f(x)}{\delta x}\right|_{(x_e,\, u_e)} \Delta x + \left.\frac{\delta f(x)}{\delta u}\right|_{(x_e,\, u_e)} \Delta u + HOT$$

$$f\left(x_e\ ,u_e\right)+f\left(\Delta x\ ,\Delta u\right)=f\left(x_e\ ,u_e\right)+\left.\frac{\delta f\left(x\right)}{\delta x}\right|_{\left(\mathrm{x}_e,\ \mathrm{u}_e\right)}\Delta x+\left.\frac{\delta f\left(x\right)}{\delta u}\right|_{\left(\mathrm{x}_e,\ \mathrm{u}_e\right)}\Delta u+HOT \quad (54.)$$

Eliminating the common term on both sides and higher order terms (HOT) the Tayler series can be modified to

$$f\left(\Delta x\ ,\Delta u\right)=\left.\frac{\delta f\left(x\right)}{\delta x}\right|_{\left(\mathrm{x}_e,\ \mathrm{u}_e\right)}\Delta x+\left.\frac{\delta f\left(x\right)}{\delta u}\right|_{\left(\mathrm{x}_e,\ \mathrm{u}_e\right)}\Delta u \quad (55.)$$

Now let redefine the state variable and control variable as $\dot{x}=f(\Delta x\ ,\Delta u)$, $\Delta x=x$ and $\Delta u=u$, then the Linearized equation can written as

$$\dot{x}=Ax+Bu$$

Where A and B are Jacobean matrixes

$$A=\left.\frac{\delta f\left(x\right)}{\delta x}\right|_{\left(\mathrm{x}_e,\ \mathrm{u}_e\right)}=\left[\begin{matrix}\frac{\delta f_1\left(x\right)}{\delta x_1} & \cdots & \frac{\delta f_1\left(x\right)}{\delta x_{12}}\\ \vdots & \ddots & \vdots\\ \frac{\delta f_{12}\left(x\right)}{\delta x_1} & \cdots & \frac{\delta f_{12}\left(x\right)}{\delta x_{12}}\end{matrix}\right]_{\left(\mathrm{x}_e,\ \mathrm{u}_e\right)}$$

$$B=\left.\frac{\delta f\left(x\right)}{\delta u}\right|_{\left(\mathrm{x}_e,\ \mathrm{u}_e\right)}=\left[\begin{matrix}\frac{\delta f_1\left(x\right)}{\delta u_1} & \cdots & \frac{\delta f_1\left(x\right)}{\delta u_6}\\ \vdots & \ddots & \vdots\\ \frac{\delta f_{12}\left(x\right)}{\delta u_1} & \cdots & \frac{\delta f_{12}\left(x\right)}{\delta u_6}\end{matrix}\right]_{\left(\mathrm{x}_e,\ \mathrm{u}_e\right)} \quad (56.)$$

The Jacobian matrices A and B will be evaluated three rows at a time, that is evaluated respectively at the body-axes Force equations (f_1 to f_3), Moment equations (f_4 to f_6), Kinematic equations (f_7 to f_9) and Navigation equations (f_{10} to f_{12}). The evaluation will be for the steady

state flight condition, with the additional constraint of no sideslip($\beta = 0$). The latter condition greatly simplifies the algebra involved in the linearization and leads to "lateral-longitudinal" decoupling.

This linearization follows for the output equation that, from introducing of feedback linearization, the above implicit non – linear ordinary differential equations from equation (23) to equation (34) the output vector can be modified as:

$$\begin{aligned} \dot{X} &= g_1(X \quad U) = 0 \\ \dot{X} &= g_2(X \quad U) = 0 \\ &\vdots \\ \dot{X} &= g_{12}(X \quad U) = 0 \end{aligned} \tag{57.}$$

By considering the perturbation from the steady state condition (x_e, u_e) and a set of linear constant-coefficient output equations are derived. If the above nonlinear output equations are expanded with a Taylor series about the steady state condition (x_e, u_e) and keep only the first order terms, the perturbations in the state, state derivative, and control vectors is satisfied.

Then, at the steady state condition a new state is defined with perturbation, which transforms the above nonlinear differential equation in to the required form.

$$x = (x_e + \Delta x, u_e + \Delta u) \tag{58.}$$

A Tayler series expansion was introduced at the steady state point

$$\dot{x} = g(x \quad u) = 0$$

$$g(x_e + \Delta x, u_e + \Delta u) = g(x_e, u_e) + \left.\frac{\delta g(x)}{\delta x}\right|_{(x_e,\, u_e)} \Delta x + \left.\frac{\delta g(x)}{\delta u}\right|_{(x_e,\, u_e)} \Delta u + HOT$$

$$g(x_e, u_e) + g(\Delta x, \Delta u) = g(x_e, u_e) + \left.\frac{\delta g(x)}{\delta x}\right|_{(x_e,\, u_e)} \Delta x + \left.\frac{\delta g(x)}{\delta u}\right|_{(x_e,\, u_e)} \Delta u + HOT \qquad (59.)$$

Eliminating the common term on both sides and higher order terms HOT the Tayler series can be modified to

$$g(\Delta x, \Delta u) = \left.\frac{\delta g(x)}{\delta x}\right|_{(x_e,\, u_e)} \Delta x + \left.\frac{\delta g(x)}{\delta u}\right|_{(x_e,\, u_e)} \Delta u \qquad (60.)$$

Now let redefine the state variable and control variable as $\dot{x} = g(\Delta x, \Delta u)$, $\Delta x = x$ and $\Delta u = u$, then the Linearized output equation can written as

$$y = Cx + Du$$

Where C and D are Jacobean matrixes

$$C = \left.\frac{\delta g(x)}{\delta x}\right|_{(x_e,\, u_e)} = \begin{bmatrix} \frac{\delta g_1(x)}{\delta x_1} & \cdots & \frac{\delta g_1(x)}{\delta x_{12}} \\ \vdots & \ddots & \vdots \\ \frac{\delta g_{12}(x)}{\delta x_1} & \cdots & \frac{\delta g_{12}(x)}{\delta x_{12}} \end{bmatrix}_{(x_e,\, u_e)}$$

$$D = \left.\frac{\delta g(x)}{\delta u}\right|_{(x_e,\, u_e)} = \begin{bmatrix} \frac{\delta g_1(x)}{\delta u_1} & \cdots & \frac{\delta g_1(x)}{\delta u_6} \\ \vdots & \ddots & \vdots \\ \frac{\delta g_{12}(x)}{\delta u_1} & \cdots & \frac{\delta g_{12}(x)}{\delta u_6} \end{bmatrix}_{(x_e,\, u_e)} \qquad (61.)$$

In the same manner, the Jacobian matrices C and D will be evaluated three rows at a time, that is evaluated respectively at the body-axes Force equations $(g_1 \; to \; g_3)$, Moment equations $(g_4 \; to \; g_6)$, Kinematic equations $(g_7 \; to \; g_9)$ and Navigation equations $(g_{10} \; to g_{12})$.. The

evaluation will be for the steady state flight condition, with the additional constraint of no sideslip($\beta = 0$). The latter condition greatly simplifies the algebra involved in the linearization and leads to "lateral-longitudinal" decoupling.

The steady state conditions for linearization are:

$$\beta \quad \varnothing \quad p \quad q \quad r \equiv 0$$

$$all\ Derivatives \equiv 0$$

$$V_T = V_{Te},\ \ \alpha = \alpha_e,\ \ \theta = \theta_e,\ \ \psi \equiv 0,\ \ \gamma_e \equiv 0$$

The algebra can be further reduced by taking advantage of some features of the steady states, that when differentiating products containing $\cos\beta$ and $\cos\emptyset$ all of the resulting $\sin\beta$ and $\sin\emptyset$ will be disappeared when steady state conditions $\beta = 0$ and $\alpha = 0$ is applied. From this it is possible to replace every $\cos\beta$ and $\cos\emptyset$ term in ordinary differential equation by a unity before applying any differentiation.

At the steady state condition with trimming total velocity of 500 ft/sec and at altitude 15000 fts and with surface disturbance of elevator disturbance deflection of $5^{0,}$ the general state space representation obtained from MATLAB by using $ss()$ function is as follows.

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y &= Cx(t) + Du(t) \end{aligned} \qquad (62.)$$

Where, A is state matrix, B is input matrix C is output matrix and D direct transmission matrix the following state space matrices are obtained when the flight dynamics is trimmed at steady – Wing – level flight condition with Low fidelity trimming condition.

$$A = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -39 & 0 & 500 & 0 & 0 & 500 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 500 & 0 & 0 & 500 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -32 & 0 & 0 & -7 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -23 & -2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -20 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -20 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -20 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1163 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -7 & 10 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -415 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -7
\end{bmatrix}$$

$$B = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}^T$$

$$C = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 57 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 57 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 57 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 57 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 57 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 57 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 57 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 57 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$D = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}^T$$

3.3. Optimal LQG Controller Design for Continuous Time System

In this section, the optimal controller based on linear Quadrature Gaussian (LQG) used in the simulation was designed. The controller design objective is based on transient response of controller that for a continuous LQG design the selected design

- The Rise time should be less than or equals 0.5 Sec
- The Settling time should be less than or equals 2 Sec
- The Peek Value should be less than or equals 0.5 and
- The steady state value should be less than or equals 0.1

The design of a linear quadratic Gaussian (LQG) controller for the Flight Control System mostly focuses on three parts based on the basic separation principle of the LQG controller mentioned above:

- Design of an optimal linear quadratic Gaussian regular (LQR)
- Design of an optimal linear quadratic estimator (LQE) and
- System integration

Together with the linear quadratic estimator the linear quadratic Gaussian regular (LQR) solves this linear-quadratic-Gaussian control problem. The following figure shows the integration of the LQG system.

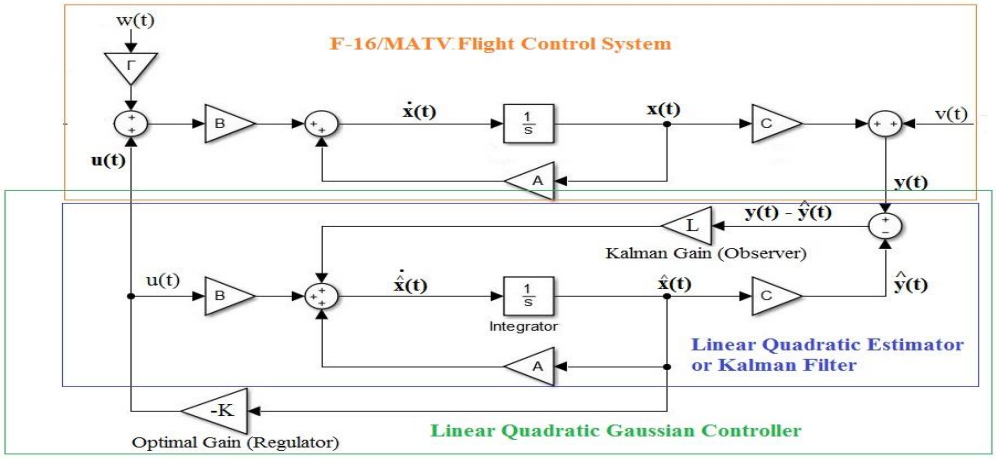

Figure 5 Integrated System of continuous LQG Optimal Controller

The linearized flight dynamic equations with a process noise w(t) and measurement error $v(t)$ was considered, and can be written as:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + \Gamma w(t) \\ y &= Cx(t) + Du(t) + v(t) \end{aligned} \tag{63.}$$

Where, $w(t)$ is stochastic input (Input noise) and measurement noise $v(t)$, which are zero-mean Gaussian stochastic processes $E\{w(t)\} = E\{v(t)\} = 0$; with their covariance matrices given by:

$$\begin{aligned} E\left\{w(t)w^T(\tau)\right\} &= W\delta(t-\tau) \geq 0, \ \textit{i.e it is positive semi-definate} \\ E\left\{v(t)v^T(\tau)\right\} &= V\delta(t-\tau) > 0, \qquad \textit{it is posive definate} \end{aligned} \tag{64.}$$

It was also assumed that $w(t), and\ v(t)$ are uncorrelated with each other and system initial values that follows:

$$E\left\{x(0)\right\} = 0; \quad E\left\{x(0)x^T(0)\right\} = P(0) \geq 0 \ \textit{and symmetric}$$

$$\begin{aligned} E\left\{x(0)v^T(\tau)\right\} &= 0 \\ E\left\{x(0)w^T(\tau)\right\} &= 0 \\ E\left\{w(t)v^T(\tau)\right\} &= 0 \end{aligned} \tag{65.}$$

Where, the mathematical expression $E\{\varepsilon\}$, is the expected value of ε, and $w^T(t),\ and\ v^T(t)$ are the transpose of random noise matrices $w(t)\ and\ v(t)$, respectively and (A, B) and (A, C) are controllable and observable matrix pairs.

For control law design it is not recommended to rely only on a full state feedback, since it is not possible to predict all values of state vector $x(t)$. Therefore an observer is required for the state

vector, based upon a measurement of the output, $y(t)$ and a known input $u(t)$, such observer are said to be a Kalman Filter (Linear Quadratic Estimator).

The general optimal problem of LQG is to find the optimal u(t), that minimizes the average cost function also called Quadratic objective function given by:

$$J = \lim_{T\to\infty} \frac{1}{2T} E\left\{\int_{-T}^{T}\left[X(t)^T QX(t) + U(t)^T RU(t)\right]dt\right\} \tag{66.}$$

For solving the LQG problem via Kalman Filter (Linear quadratic Estimator (LQE)), The estimated state $\hat{x}(t)$ was obtained from Kalman fliter, which minimizes the performance Index given by a trace of covariance of Estimation error matrices as shown below:

$$\begin{aligned} J_e &= tr\left[E\left\{\left(x(t)-\hat{x}(t)\right)(x(t)-\hat{x}(t)^T\right)\right\}\right] \\ &= \sum_{i=1}^{n} E\left\{x_i(t)-x_i(t)\right\}^2 \end{aligned} \tag{67.}$$

3.3.1. Design of Optimal Linear quadratic Regulator (LQR)

Recalling the continuous time state space equation $(63.)$ redefining control variable as

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + \Gamma w(t) \\ u(t) &= -Kx(t) \end{aligned} \tag{68.}$$

Where A is the system matrix, B is the control matrix, and C is the output matrix. LQR theory determines the optimal gain matrix K such that the state-feedback law u = −Kx subject to minimize the quadratic cost function

$$J = \int_0^{\infty}\left(x^T(t)Q(t)x(t) + u^T(t)R(t)u(t)\right)dt \tag{69.}$$

Where, Q and R are the weighting matrices. The corresponding optimal control is given by

$$u = -Kx = R^{-1}B^T Px$$

$$K = -R^{-1}B^T P \tag{70.}$$

Where, K is optimal feedback gain matrix also called controller gain, which can be obtained from the Riccati matrix P. The Riccati matrix is determined by the solution of the following steady state Riccati equation.

$$AP + A^T P - PBR^{-1}B^T P + C^T QC = 0 \tag{71.}$$

For this final year thesis, the MATLAB Function $lqg()$ is used to obtain the following Optimanl regulator gain matrix:

$$K = \begin{bmatrix} 738 & 0 & 160 & 0 & 89976 & 0 & 1501 & -91760 & 0 & 0 & 2014 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 160 & 0 & -738 & 0 & -22733 & 0 & 128 & 188017 & 0 & 0 & -10390 & 0 & 0 & 99 & 0 & 0 & 0 & 0 \\ 0 & -669 & 0 & 10755 & 0 & -202242 & 0 & 0 & -190281 & -987 & 0 & -3425 & 0 & 0 & 99 & 0 & 0 & 0 \\ 0 & -350 & 0 & 10942 & 0 & -20 & 0 & 0 & -190395 & -470 & 0 & -4175 & 0 & 0 & 0 & 2 & 0 & 0 \end{bmatrix} \tag{72.}$$

3.3.2. Design of Optimal Linear quadratic estimator (LQE) or Kalman Filter

The new estimated state equation can be defined as:

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + L\left(y(t) - C\hat{x}(t)\right)$$

$$\dot{\hat{x}}(t) = (A - LC)\hat{x}(t) + Bu(t) + Ly(t) \tag{73.}$$

From the optimal regulator the control problem is now defined as

$$u(t) = -K\hat{x}(t) \tag{74.}$$

Where, L is called a Kalman gain (also called optimal observer gain matrix) of the associated Kalman filter represented in above equation which for each time t, generates an estimate $\hat{x}(t)$ of

the state $x(t)$ using the past measurement and inputs. Can be defined from continuous filter algebraic Riccati equation as:

$$L = K^T = P_e C V^{-1} \tag{75.}$$

Where, P_e is the covariance of estimation error $\varepsilon_0(t)$ defined by:

$$\begin{aligned} \varepsilon_0(t) &= x(t) - \hat{x}(t) \\ P_e &= E\left\{ \varepsilon_0(t)\, \varepsilon_0(t)^T \right\} \end{aligned} \tag{76.}$$

And can be computed by solving a Continues Filter Algebraic Riccati Equation (CFARE) obtained from Algebraic Riccati Equation when replacing $A\ by\ A^T$, $B\ by\ C^T$, $Q\ by\ W$, $and\ R\ by\ V$ given by:

$$AP_e + P_e A^T - P_e C^T V^{-1} C P_e + \Gamma W \Gamma^T = 0 \tag{77.}$$

Using the MATLAB function $kalman()$ the following Kalman fliter gain matix is obtained when the F-16/MATV dynamics is trimmed at the steady Wing flight level with Low fidelity flight condition:

$$L = \begin{bmatrix}
10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 16 & 0 & -1 & 0 & 8 & 0 & 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 16 & 0 & 8 & 0 & 0 & -8 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 \\
0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -6 & 0 & 0 & 0 & 0 & 19 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\
0 & 0 & 3 & 0 & 0 & 0 & 0 & -7 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0
\end{bmatrix}$$

3.3.3. System Integration of LQG Controller

The last step in design of Linear Quadratic Gaussian controller is system integration as shown in Figure 6. Since the optimal regulator and Kalman filter are designed separately (separation principle), they can be selected to have desirable properties that are independent of one another. The closed loop eigenvalues consist of the regulator eigenvalues and the Kalman filter eigenvalues. The closed-loop system's performance can be obtained as desired by suitably selection of the optimal regulator's weighting matrices, Q and R, and the Kalman filter's spectral noise densities, W, V, and G. Hence, the matrices Q, R, W, V, and G are the design parameters for the closed-loop system with an optimal Compensator. The following shows the integrated system of LQG controller.

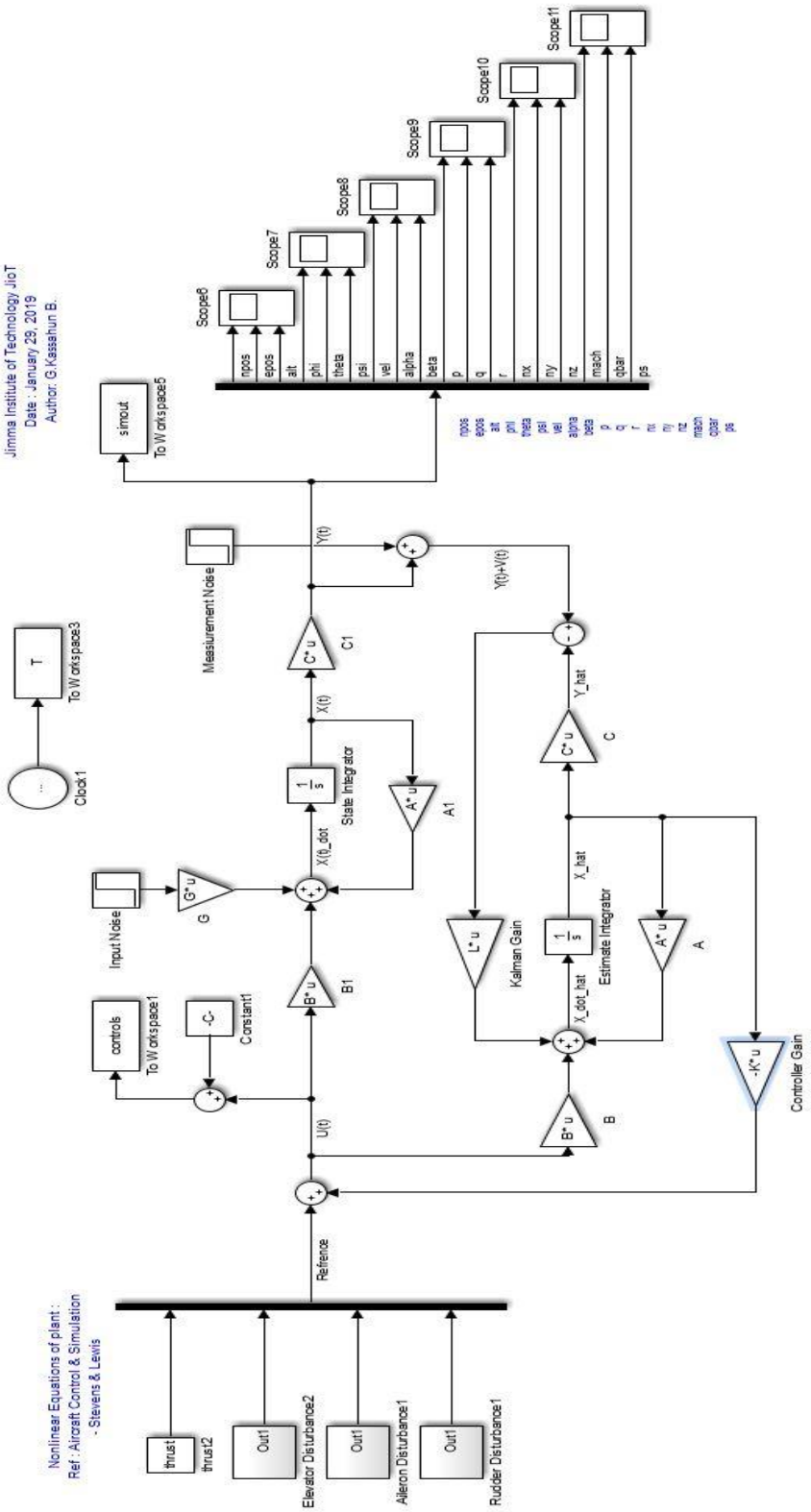

Figure 6 The general optimal continuous LQG Controller configuration structure

3.4. Discrete LQG Controller Design for F-16/MATV Flight System

The design of discrete LQG controller follows the same procedure as the continuous LQG and only differs from continuous time LQG in that the Plant considered should be in written in discrete time state space representation.

The MATLAB Simulink is a powerful tool to convert the above obtained continuous time state space flight dynamics to directly converted to discrete time by using $c2d()$ function with sampling time $ts = 0.4\ sec$ the obtained discrete time will have the form

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k) \qquad (78.)$$

Where the obtained discrete state space matrices are as follows:

$A =$

1.00000	0.00000	0.00001	0.00000	-2.56854	0.00000	0.39896	-0.46474	0.00000	0.00000	-0.45719	0.00000	0.00011	0.00318	0.00000	0.00000	0.00000	0.00000
0.00000	1.00000	0.00000	-13.07233	0.00000	200.00000	0.00000	0.00000	192.38114	0.21403	0.00000	1.22514	0.00000	0.00000	-0.00319	0.00262	0.00000	0.00000
0.00000	0.00000	0.99992	0.00000	199.95367	0.00000	0.00940	-177.32529	0.00000	0.00000	4.89882	0.00000	0.00001	-0.01016	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.98880	0.00000	0.00000	0.00000	0.00000	-1.14812	0.25071	0.00000	0.22735	0.00000	0.00000	-0.00551	0.00023	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.99999	0.00000	0.00000	-0.10081	0.00000	0.00000	0.32501	0.00000	0.00000	-0.00164	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00373	0.00000	1.00000	0.00000	0.00000	0.41349	0.00184	0.00000	0.32115	0.00000	0.00000	-0.00038	-0.00067	0.00000	0.00000
0.00000	0.00000	0.00004	0.00000	-12.83020	0.00000	0.99482	-1.91280	0.00000	0.00000	-3.07983	0.00000	0.00051	0.01777	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00067	0.00000	-0.00009	0.68772	0.00000	0.00000	0.26634	0.00000	0.00000	-0.00142	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.02007	0.00000	0.00000	0.00000	0.00000	0.42422	0.02059	0.00000	-0.28648	0.00000	0.00000	-0.00012	0.00068	0.00000	0.00000
0.00000	0.00000	0.00000	-0.07571	0.00000	0.00000	0.00000	0.00000	-4.28977	0.32311	0.00000	1.24175	0.00000	0.00000	-0.01016	-0.00143	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	-0.00013	0.00000	0.00003	-0.44512	0.00000	0.00000	0.61973	0.00000	0.00000	-0.00373	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.02644	0.00000	0.00000	0.00000	0.00000	1.79730	0.01892	0.00000	0.49163	0.00000	0.00000	-0.00101	-0.00127	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.67032	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00031	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00031	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00031	0.00000	0.00000
0.00000	0.00000	0.00004	0.00000	0.09179	0.00000	-0.01204	64.35809	0.00000	0.00000	21.08627	0.00000	-0.00001	-0.10986	0.00000	0.00000	0.05280	0.21879
0.00000	0.00000	-0.00003	0.00000	-0.02085	0.00000	0.00348	-42.39563	0.00000	0.00000	-11.12099	0.00000	0.00000	0.05662	0.00000	0.00000	0.00000	0.05502

$$B = \begin{bmatrix} 0.00002 & 0.00847 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & -0.00386 & 0.01227 \\ 0.00000 & -0.00592 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & -0.02254 & 0.00171 \\ 0.00000 & -0.00631 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & -0.00140 & -0.00264 \\ 0.00011 & 0.06434 & 0.00000 & 0.00000 \\ 0.00000 & -0.00590 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & -0.00049 & 0.00287 \\ 0.00000 & 0.00000 & -0.11081 & 0.00560 \\ 0.00000 & -0.03308 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & -0.00757 & -0.01343 \\ 0.32968 & 0.00000 & 0.00000 & 0.00000 \\ 0.00000 & 0.99969 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & 0.99969 & 0.00000 \\ 0.00000 & 0.00000 & 0.00000 & 0.99969 \\ 0.00000 & -0.38285 & 0.00000 & 0.00000 \\ 0.00000 & 0.18033 & 0.00000 & 0.00000 \end{bmatrix}$$

$C =$

1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	57.29578	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	57.29578	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	57.29578	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	57.29578	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	57.29578	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	57.29578	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	57.29578	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	57.29578	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00012	0.00000	-0.00010	0.54708	0.00000	0.00000	0.03817	0.00000	0.00005	0.00393	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	-0.00012	0.00000	0.00000	0.00000	0.00000	-3.14234	0.00643	0.00000	0.07809	0.00000	0.00000	0.00268	0.00786	0.00000	0.00000
0.00000	0.00000	-0.00003	0.00000	-0.00001	0.00000	0.00399	9.92978	0.00000	0.00000	0.96642	0.00000	0.00000	0.02084	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00095	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	-0.00609	0.00000	0.00000	0.00000	0.74928	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	-0.04820	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

For design of discrete LQG controller, the discrete time state space representation of flight dynamic equations with a discrete process noise $w(k)$ and measurement error $v(k)$ was considered, and the discrete system can be written as:

$$x(k+1) = Ax(k) + Bu(k) + \Gamma w(k)$$
$$y(k) = Cx(k) + Du(k) + v(k) \tag{79.}$$

Where, $w(k)$ is discrete stochastic process and measurement noise $v(k)$, which are zero-mean Gaussian stochastic process noise $E\{w(k)\} = E\{v(k)\} = 0$; with their covariance matrices given by:

$$E\left\{w(k)w^T(i)\right\} = W\delta(k-i) \geq 0, \text{ i.e it is positive semi} - \text{definate}$$
$$E\left\{v(k)v^T(i)\right\} = V\delta(k-i) > 0, \text{ it is posive definate} \tag{80.}$$

The discrete Gaussian noises are also assumed that $w(k), and\, v(k)$ are uncorrelated with each other and system initial values that follows:

$$E\{x(0)\} = 0; \quad E\left\{x(0)x^T(0)\right\} = s \geq 0 \text{ and symmetric}$$
$$E\left\{x(0)v^T(k)\right\} = 0$$
$$E\left\{x(0)w^T(k)\right\} = 0 \tag{81.}$$
$$E\left\{w(k)v^T(i)\right\} = 0$$

Where, the mathematical expression $E\{\varepsilon\}$, is the expected value of ε, and $w^T(k),\ and\ v^T(k)$ are the transpose of random noise matrices $w(k)\ and\ v(k)$, respectively.

For discretized system it is obvious that (A, B) and (A, C) as controllable and observable matrix pairs assumption is also valid.

For control law design it is not recommended to rely only on a full state feedback, since it is not possible to predict all values of state vector $x(k)$. Therefore an observer is required for the state vector, based upon a measurement of the output, $y(k)$ and a known input $u(k)$, such observer are said to be a discrete Kalman Filter (Discrete Linear Quadratic Estimator).

The general optimal problem of discrete LQG is to find the optimal u(k), that minimizes the average cost function also called Quadratic objective function given by:

$$J = \sum_{k=0}^{\infty} \left[x(k)^T \, q_c(k) x(k) + u(k)^T \, r_c(k) u(k) \right] \tag{82.}$$

Where $q_c(k)\ and\ r_c(k)$ are weighting matrices, i.e design parameter to chosen to meet the desired closed loop performance [23].

For solving the discrete LQG problem via discrete Kalman Filter (Linear quadratic Estimator (LQE)), The discrete estimated state $\hat{x}(k)$ was obtained from discrete Kalman filter, which minimizes the performance Index given by a trace of covariance of discrete Estimation error matrices as shown below:

$$\begin{aligned} J_e &= tr\left[E\left\{ \left(x(k) - \hat{x}(k) \right)\left(x(k) - \hat{x}(k)^T \right) \right\} \right] \\ &= \sum_{i=1}^{n} E\left\{ x_i(k) - x_i(k) \right\}^2 \end{aligned} \tag{83.}$$

3.4.1. Design of Discrete Optimal Linear quadratic Regulator (LQR)

The estimated discrete state equation can be defined as

$$\begin{aligned} \hat{x}(k+1 \mid k) &= (A - LC)\hat{x}(k \mid k-1) + Bu(k \mid k-1) + Ly(k \mid k-1) \\ u(k) &= -K_d \hat{x}(k \mid k-1) \end{aligned} \tag{84.}$$

Where, the notation $\hat{x}(k+1|k)$ denotes that the estimate (or prediction) of $\hat{x}(k+1)$ was made using a measurement available at time k, and A is the system matrix, B is the control matrix, and

C is the output matrix. LQR theory determines the optimal gain matrix K such that the state-feedback law u(k) $= -K_d\hat{x}(k)$ subject to minimize the quadratic cost function

$$J = \sum_{k=0}^{\infty}\left(x^T(k)q_c(k)x(k)+u^T(k)r_c(k)u(k)\right) \tag{85.}$$

Where, Q and R are the weighting matrices. The corresponding discrete optimal control is given by

$$u = -K_d x(k\,|\,k-1)$$

$$K_d = R^{-1}B^T P_d \tag{86.}$$

Where, K_d is optimal feedback gain matrix also called controller gain, which can be obtained from the Riccati matrix P_d. The Riccati matrix is determined by the solution of the following steady state Discrete Algebraic Riccati Equation (D.A.R.E.).

$$AP_d + A^T P_d - P_d BR^{-1}B^T P_d + C^T QC = 0 \tag{87.}$$

For this final year thesis, the MATLAB Function $dlqg()$ is used to obtain the following optimal regulator gain matrix:

$K_d =$

4.1076	0.0000	5.1674	0.0000	4734.9396	0.0000	75.1390	-4854.1808	0.0000	0.0000	106.0041	0.0000	0.1727	0.0034	0.0000	0.0000	0.0000	0.0000
0.9305	0.0000	-0.7141	0.0000	-205.2861	0.0000	5.9252	66.7428	0.0000	0.0000	-40.5016	0.0000	0.0074	0.2162	0.0000	0.0000	0.0000	0.0000
0.0000	-2.2589	0.0000	53.5695	0.0000	-1214.1507	0.0000	0.0000	-1115.4102	-7.0504	0.0000	-43.5504	0.0000	0.0000	0.1823	0.0461	0.0000	0.0000
0.0000	1.6789	0.0000	-29.4954	0.0000	426.9117	0.0000	0.0000	402.1446	-0.2041	0.0000	-4.3381	0.0000	0.0000	0.0168	0.0210	0.0000	0.0000

3.4.2. Design of Discrete Optimal Linear quadratic Estimator (LQE)

The new estimated discrete state equation can be defined as:

$$x(k+1) = A\hat{x}(k) + Bu(k) + L\left(y(k) - C\hat{x}(k)\right)$$

$$x(k+1) = \left(A - LC - BK\right)\hat{x}(k) + Ly(k) \tag{88.}$$

$$u(k) = -K_d\hat{x}(k)$$

Where, L is called a discrete Kalman gain (also called discrete optimal observer gain matrix) of the associated discrete Kalman filter represented in above equation which for each time k, generates an estimate $\hat{x}(k)$ of the state $x(k)$ using the past measurement and inputs. Can be defined from discrete filter algebraic Riccati equation as:

$$L = K^T = S_e C V^{-1} \tag{89.}$$

Where, S_e is the discrete covariance of estimation error $\tilde{x}(k)$ defined by:

$$\tilde{x}(k) = x(k) - \hat{x}(k)$$

$$S_e = E\left\{\tilde{x}(k)\tilde{x}(k)^T\right\} \tag{90.}$$

And can be computed by solving a Discrete Filter Algebraic Riccati Equation (DFARE) obtained from Algebraic Riccati Equation when replacing A *by* A^T, B *by* C^T, Q *by* W *, and* R *by* V given by:

$$AS_e + S_eA^T - S_eC^TV^{-1}CS_e + \Gamma W\Gamma^T = 0 \tag{91.}$$

Using the MATLAB function $kalman()$ the following Kalman filter gain matrix is obtained:

$$L = \begin{bmatrix}
0.9999 & 0.0000 & 0.0012 & 0.0000 & -0.0448 & 0.0000 & 0.2552 & -0.0291 & 0.0000 & 0.0000 & -0.0100 & 0.0000 & 0.0232 & 0.0000 & 0.1199 & 0.0002 & 0.1912 & -0.0001 \\
0.0000 & 0.9999 & 0.0000 & -0.2282 & 0.0000 & 3.4907 & 0.0000 & 0.0000 & 3.3556 & 0.0037 & 0.0000 & 0.0214 & 0.0000 & -0.0378 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.9975 & 0.0000 & 3.4898 & 0.0000 & 0.0102 & -3.0098 & 0.0000 & 0.0000 & 0.0938 & 0.0000 & -0.0230 & 0.0000 & -0.4899 & 0.0000 & 0.0016 & -0.0481 \\
0.0000 & 0.0000 & 0.0000 & 0.0173 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & -0.0242 & 0.0044 & 0.0000 & 0.0041 & 0.0000 & -0.0762 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0175 & 0.0000 & 0.0002 & 0.0091 & 0.0000 & 0.0000 & 0.0067 & 0.0000 & -0.0119 & 0.0000 & -0.0617 & 0.0000 & 0.0001 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0001 & 0.0000 & 0.0175 & 0.0000 & 0.0000 & 0.0052 & 0.0000 & 0.0000 & 0.0057 & 0.0000 & -0.0371 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0030 & 0.0000 & -0.2239 & 0.0000 & 0.6354 & -0.1505 & 0.0000 & 0.0000 & -0.0651 & 0.0000 & 0.1297 & 0.0000 & 0.6685 & 0.0006 & 0.4761 & -0.0001 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0001 & 0.0213 & 0.0000 & 0.0000 & 0.0055 & 0.0000 & -0.0105 & 0.0000 & -0.0529 & 0.0000 & 0.0001 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0004 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0090 & 0.0004 & 0.0000 & -0.0050 & 0.0000 & 0.0296 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & -0.0013 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & -0.0872 & 0.0057 & 0.0000 & 0.0220 & 0.0000 & -0.2248 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0004 & 0.0169 & 0.0000 & 0.0000 & 0.0132 & 0.0000 & -0.0270 & 0.0000 & -0.1407 & 0.0000 & 0.0003 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0005 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0272 & 0.0003 & 0.0000 & 0.0087 & 0.0000 & -0.0759 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0001 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0005 & 0.0098 & 0.0000 & 0.0000 & 0.0011 & 0.0000 & 0.5771 & 0.0000 & -0.0882 & 0.0000 & 0.0004 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & -0.0020 & 0.0000 & 0.0000 & -0.0002 & 0.0000 & 0.0022 & 0.0000 & 0.0117 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0003 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0049 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0008 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0144 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0016 & 0.0000 & 0.0027 & 1.8414 & 0.0000 & 0.0000 & 0.4377 & 0.0000 & -0.8173 & 0.0000 & -4.0985 & 0.0000 & 0.0020 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & -0.0004 & 0.0000 & -0.0031 & -1.1091 & 0.0000 & 0.0000 & -0.2299 & 0.0000 & 0.4239 & 0.0000 & 2.1068 & 0.0000 & -0.0023 & 0.0000
\end{bmatrix}$$

3.4.3. System Integration of Discrete LQG Controller

At the last a discrete Linear Quadratic Gaussian controller is integrated as shown in Figure 7. Since the discrete optimal regulator and discrete Kalman filter are designed separately (separation principle), they can be selected to have desirable properties that are independent of one another. The closed loop eigenvalues consist of the regulator eigenvalues and the Kalman filter eigenvalues. The closed-loop system's performance can be obtained as desired by suitably selection of the optimal regulator's weighting matrices, Q and R, and the Kalman filter's spectral noise densities, W, V, and G. Hence, the matrices Q, R, W, V, and G are the design parameters for the closed-loop system with an optimal Compensator. The following figure shows the system integration of discrete time LQG controller.

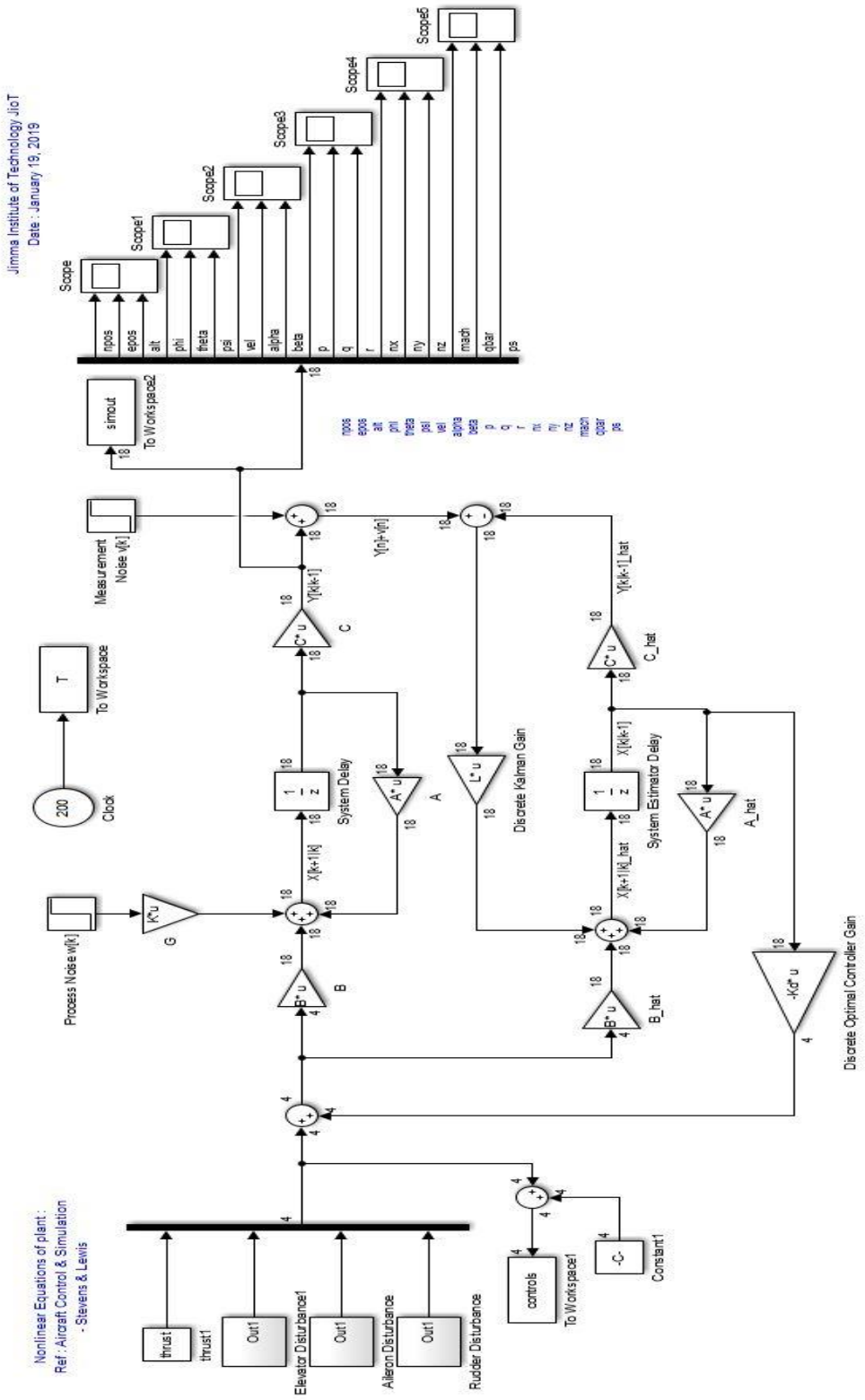

Figure 7 The system configuration of Discrete LQG controller

3.5. Robustness Checking for LQG Controller

The robustness of LQG controller is related to the controllers' stability performance in sense of controller gain margin and phase margins.

Recalling the LQG optimal problem by considering the state estimate dynamic equation, one can had

$$\dot{\hat{x}}(t) = (A - BK - LC)\hat{x}(t) + Ly(t) \tag{92.}$$

Where $K \; and \; L$ are LQG optimal controller Gain and Kalman fliter gain respectively with

$$u(t) = -K\hat{x}(t)$$

Taking the Laplace transform of the above equation

$$\begin{gathered} L_T\{u(t)\} = L_T\{-K\hat{x}(t)\} \\ u[s] = -K\hat{x}[s] \end{gathered} \tag{93.}$$

But from equation (92.) taking the Laplace transform to both sides and rearranging assuming $\hat{x}[0] = 0$,

$$\hat{x}[s] = (sI - (A - BK - LC))^{-1} Ly[s] \tag{94.}$$

Substituting for $\hat{x}[s]$ into equation (93.)

$$u[s] = -K(sI - (A - BK - LC))^{-1} Ly[s] \tag{95.}$$

Letting

$$u[s] = -K[s]y[s]$$

Where $K[s] = K(sI - (A - BK - LC))^{-1}L$ then introducing the state resolvent matrixes as

$$Q_r[s] = (sI - (A - BK - LC))^{-1} \tag{96.}$$

$$K[s] = KQ_r[s]L$$

Consider the general closed loop feedback system of the flight control system as

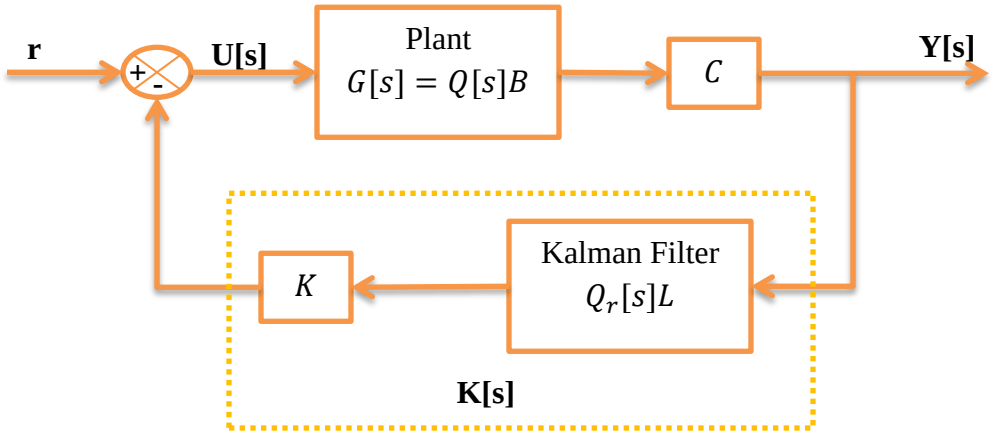

Figure 8 A closed loop control system of LQG for the flight control system

The loop gain of the closed loop system with LQG controller, can be written as

$$L_r[s] = KQ_r[s]LCQ[s]B \tag{97.}$$

Now considering the closed loop system for LQR in sense of separation principle

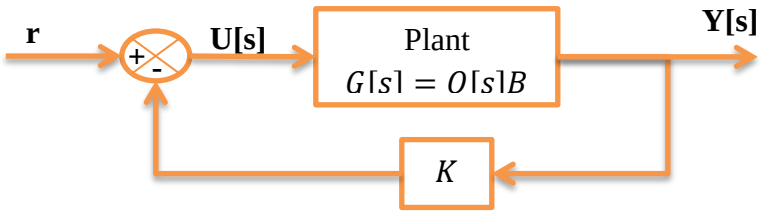

Figure 9 A closed loop control system of LQR for the flight control system

The loop gain of LQR control system will be

$$L = KQ[s]B$$

In general LQG controller does not have the same properties as that of LQR method due to the introduction of Kalman Filter. Therefore, it had lower stability margins than the sensitivity

property is not good as that of LQR method. Thus the robust property of LQG controller is that there is a possibility to recover some of the desirable properties like gain and phase margins for LQG by selecting the appropriate process noise covariance matrix.

Defining the following properties of state transition matrices

$$Q[s] = (sI - A)^{-1}$$
$$Q_c[s] = (sI - A - BK)^{-1}$$

From equation (97.) the loop gain can be rewritten as

$$L_r[s] = K(sI - (A - BK - LC))^{-1} LCQ[s]B$$

$$= K((sI - (A - BK)) - LC))^{-1} LCQ[s]B \quad (98.)$$

$$= K(Q_c[s]^{-1} - LC))^{-1} LCQ[s]B$$

Applying the matrix inversion lemma

$$(\bar{A} - \bar{B}\bar{C}\bar{D})^{-1} = \bar{A}^{-1} - \bar{A}^{-1}\bar{B}(\bar{D}\bar{A}^{-1}\bar{B} + \bar{C}^{-1})\bar{D}\bar{A}^{-1}$$

Setting $\bar{A} = Q_c[s]^{-1}, \bar{B} = L, \bar{C} = I \text{ and } \bar{D} = C$

Substituting into equation (98.)

$$L_r[s] = K\{Q_c[s] - Q_c[s]L(CQ_c[s]L + I)^{-1} CQ_c[s]\}LCQ[s]B$$

$$L_r[s] = KQ_c[s]L\{I - (I + CQ_c[s]L)^{-1} CQ_c[s]L)CQ[s]B$$

$$L_r[s] = KQ_c[s]L\{(I + CQ_c[s]L)^{-1}(I + CQ_c[s]L) - CQ_c[s]L)CQ[s]B$$

$$L_r[s] = KQ_c[s]L\left(I + CQ_c[s]L\right)^{-1} CQ[s]B \tag{99.}$$

From the basic flight system, the continuous time dynamic equation (63.)

$$\dot{x}(t) = Ax(t) + Bu(t) + \Gamma w(t)$$

To recover the robustness properties associated with full-state feedback, the Kalman filter must be designed such that the sensitivity of the plant's input to process and measurement noise is minimized. To do so its covenant that the process noise covariance can be redefined as

$$\Gamma \mathrm{w} \Gamma^T = \Gamma \mathrm{w}_0 \Gamma^T + q^2 BB^T \tag{100.}$$

Where $q > 0$, is any positive scalar quantity and w_0 is initial guess of process noise covariance matrix. Using this equation in the algebraic Reccati of kalman filter equation (77.)

$$AP_e + P_e A^T - P_e C^T V^{-1} CP_e + \Gamma \mathrm{W} \Gamma^T = 0$$

$$AP_e + P_e A^T - P_e C^T V^{-1} CP_e + \Gamma \mathrm{w}_0 \Gamma^T + q^2 BB^T = 0 \tag{101.}$$

$$A\frac{P_e}{q^2} + \frac{P_e}{q^2}A^T - \frac{P_e}{q^2}C^T V^{-1} C\frac{P_e}{q^2}q^2 + \frac{\Gamma \mathrm{w}_0 \Gamma^T}{q^2} + BB^T = 0$$

Assuming if the plant is minimum phase system and B and C are full rank with $m = p$

$$\lim_{q \to \infty} \frac{P_e}{q^2} = 0$$

Equation (101.) becomes

$$q^2 \frac{P_e}{q^2} C^T V^{-1} C \frac{P_e}{q^2} - BB^T = 0$$

$$q^2 \frac{P_e}{q^2} C^T V^{-1} C \frac{P_e}{q^2} \to BB^T$$

$$\left(P_e C^T V^{-1}\right) V (V^{-1} C P_e) \frac{1}{q^2} \to BB^T$$

But $P_e C^T V^{(-1)} = L$ is a Kalman gain then

$$LVL^T \frac{1}{q^2} \to BB^T$$

Since V is the measurement noise matrix which is symmetric matrix

$$\frac{\left(LV^{1/2}\right)}{q} \frac{\left(LV^{1/2}\right)^T}{q} \to BB^T \qquad (102.)$$

The general solution of equation $(102.)$ is

$$L \to qBV^{-1/2} \qquad as \qquad V \to \infty \qquad (103.)$$

Using equation (103.) in equation (99.)

$$L_r[s] = KQ_c[s] qBV^{-1/2} \left(I + CQ_c[s] qBV^{-1/2}\right)^{-1} CQ[s] B \qquad as \;\; V \to \infty$$

$$L_r[s] = KQ_c[s] qBV^{-1/2} \left(CQ_c[s] qBV^{-1/2}\right)^{-1} CQ[s] B \qquad as \;\; V \to \infty$$

$$L_r[s] = KQ_c[s] B\left(CQ_c[s] B\right)^{-1} CQ[s] B \qquad as \;\; V \to \infty \qquad (104.)$$

Using the Identity

$$Q_c[s] = (sI - A - BK)^{-1}$$

$$Q_c[s] = ((sI - A) - BK)^{-1}$$

$$Q_c[s] = \left(Q[s]^{-1} - BK\right)^{-1} \tag{105.}$$

$$Q_c[s] = \left\{\left(I - BKQ[s]\right)Q[s]^{-1}\right\}^{-1}$$

$$Q_c[s] = Q[s]\left(I - BKQ[s]\right)^{-1}$$

Substituting for $Q_c[s]$ from equation (105.) into equation (104.)

$$L_r[s] = K\left(Q[s]\left(I - BKQ[s]\right)^{-1}\right)B\left(C\left(Q[s]\left(I - BKQ[s]\right)^{-1}\right)B\right)^{-1}CQ[s]B \qquad as \;\; V \to \infty$$

Using the Identity

$$\left(I - BKQ[s]\right)^{-1}B = B\left(I - BKQ[s]B\right)^{-1}$$

$$L_r[s] = K(Q[s]\left\{B\left(I - BKQ[s]B\right)^{-1}\right\}\left(C(Q[s]\left\{B\left(I - BKQ[s]B\right)^{-1}\right\}\right)^{-1}CQ[s]B \qquad as \;\; V \to \infty$$

$$L_r[s] = K(Q[s]B\left(I - BKQ[s]B\right)^{-1}\left(I - BKQ[s]B\right)\left(C(Q[s]B\right)^{-1}CQ[s]B \qquad as \;\; V \to \infty$$

$$L_r[s] = K(Q[s]B = L \tag{106.}$$

This is the loop gain of LQR. Thus the robustness of LQG controller is recovered that the Gain margin and Phase margin of LQR is attained.

CHAPTER FOUR

4. Simulation Results and Discussion

The simulation results shown here are the results obtained from MATLAB/Simulink environment when the F16/MATV is trimmed at Steady state conditions with total velocity of 500 ft/sec and altitude of 15000 fts.

4.1. Non – Linear F16/MATV Flight Response

The following nonlinear response are obtained from MATLAB Simulation environment when the F-16/MATV flight dynamics are trimmed at a steady Wings – level flight state condition with inclusion of surface disturbance of elevator disturbance deflection of 5^0 (degrees). From this one can deduce that the obtained results of nonlinear flight responses are unstable and the system responses at the steady state do not attain the steady state condition.

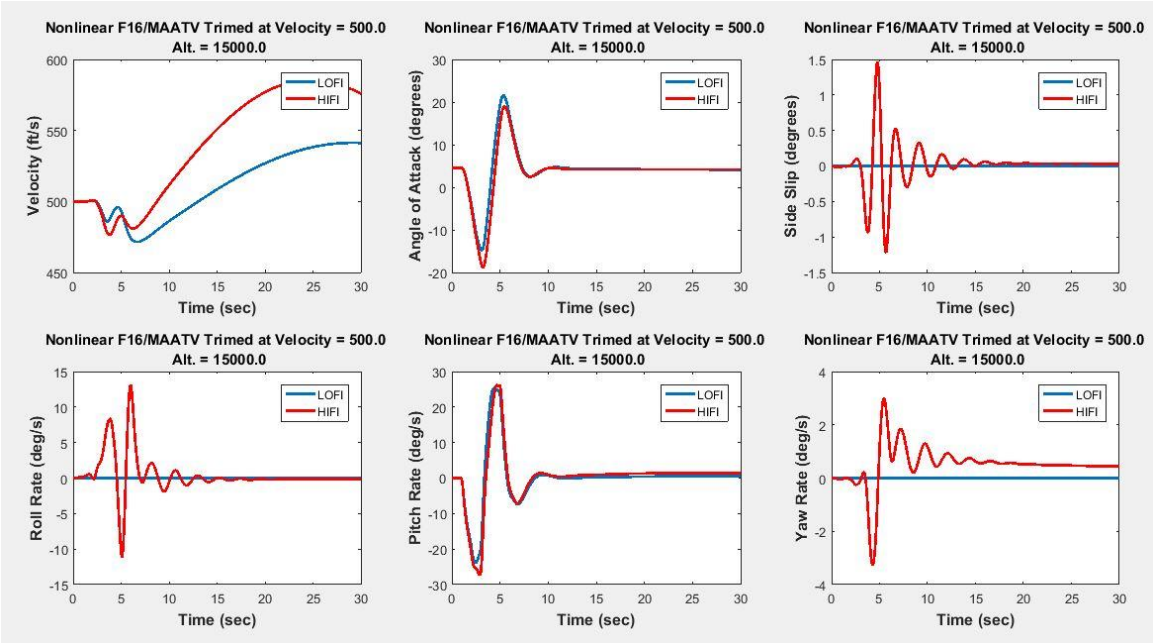

Figure 10 Nonlinear Response of F-16/MATV system of Force dynamic condition

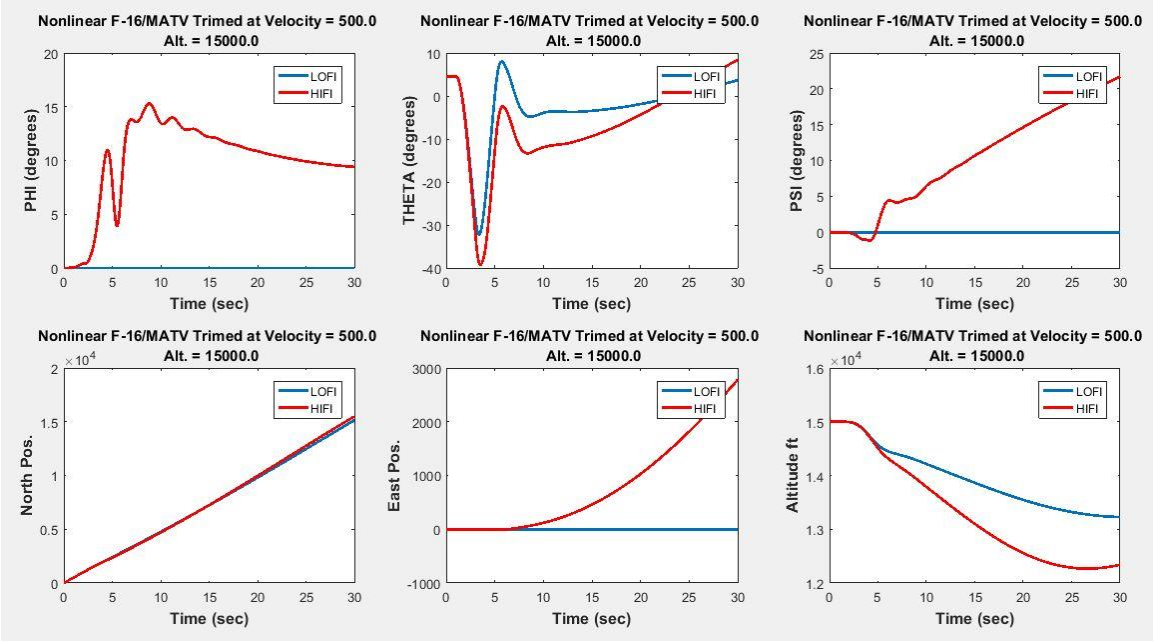

Figure 11 Nonlinear Response of F-16/MATV flight to the Kinematic dynamic condition

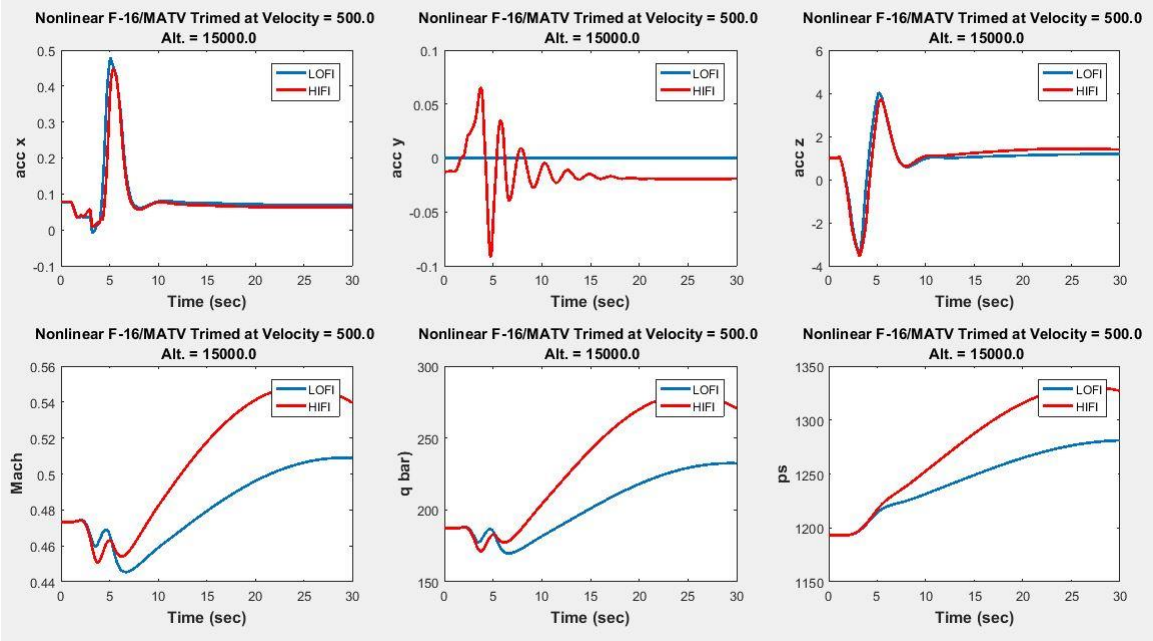

Figure 12 Nonlinear F-16/MATV flight system response of navigation flight condition

4.2. Linearized F16/MATV Flight Response

For the linearized F-16/MATV system responses shows that the steady state condition is attained and since system is not applied to any controller so that the steady state time is so larger and the system response looks like unstable system.

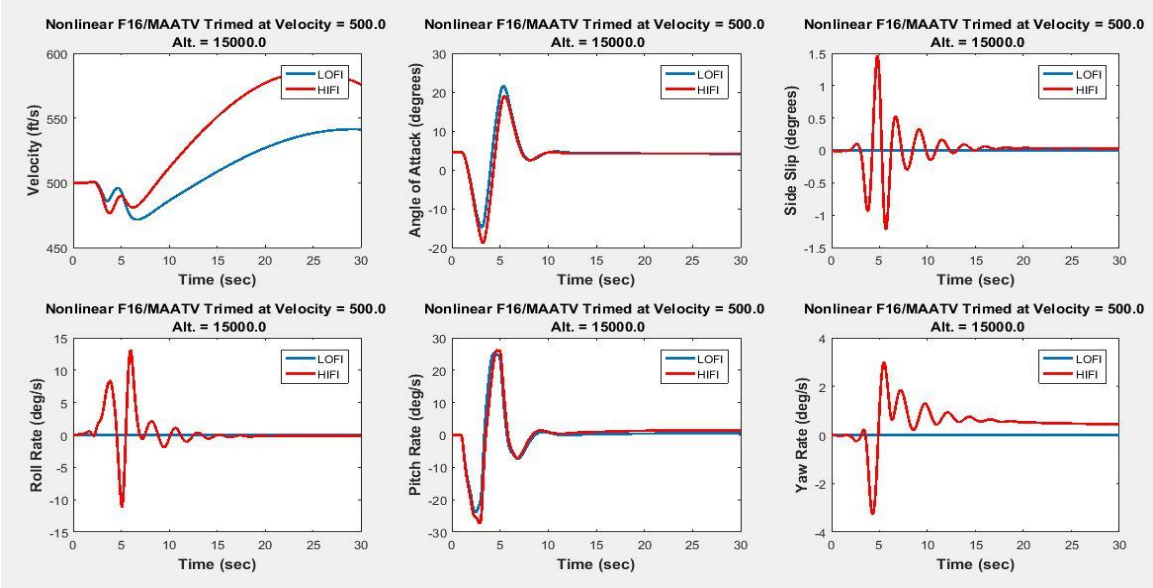

Figure 13 The Linearized F-16/MATV response of Force dynamics

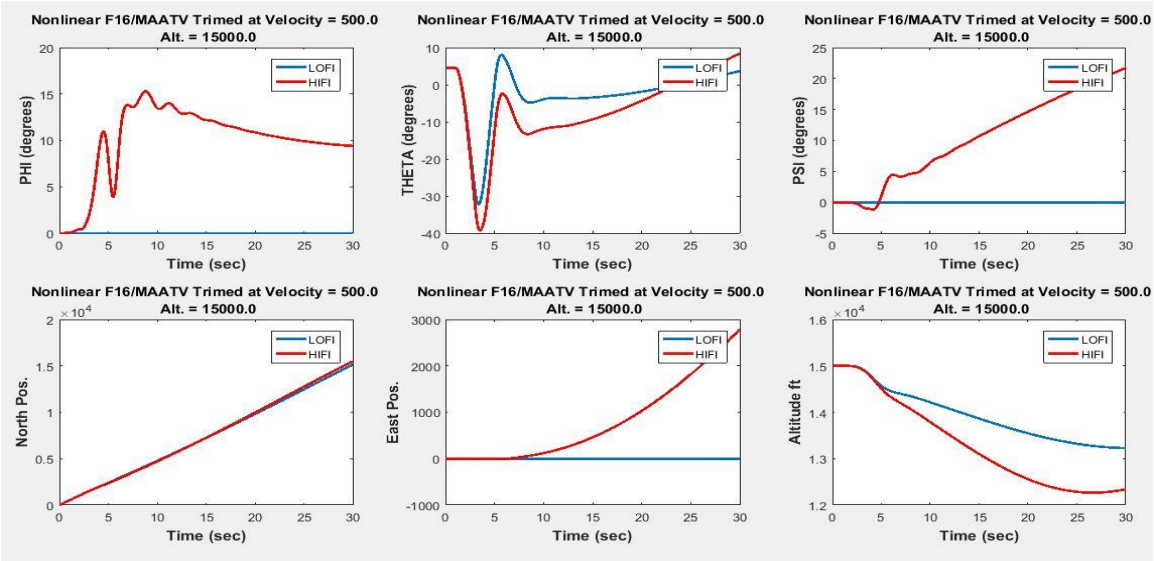

Figure 14 The Linearized F-16/MATV response of Kinematic dynamics

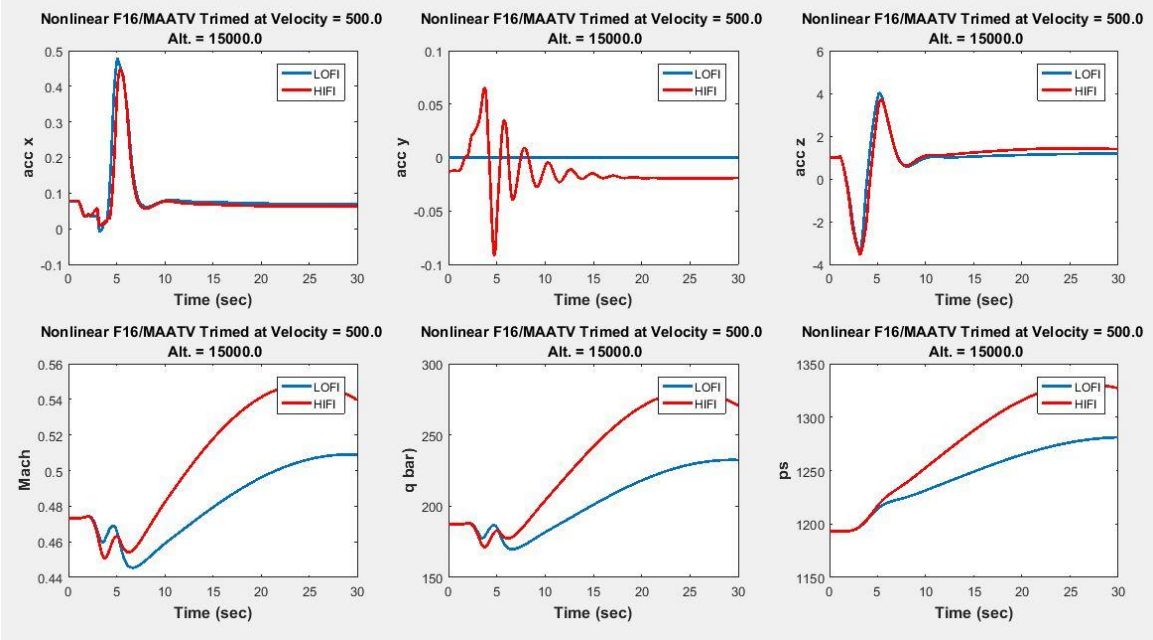

Figure 15 The Linearized F-16/MATV response of Navigation dynamics

4.3. The Continuous time LQG Controller Response

At the same trimming steady state condition when the continuous Linear Quadrature Gaussian Controller is applied to control the linearized F-16/MATV flight system the design objectives are attained on time and the system is more stable. The following simulation results are obtained from MATLAB for the same flight condition.

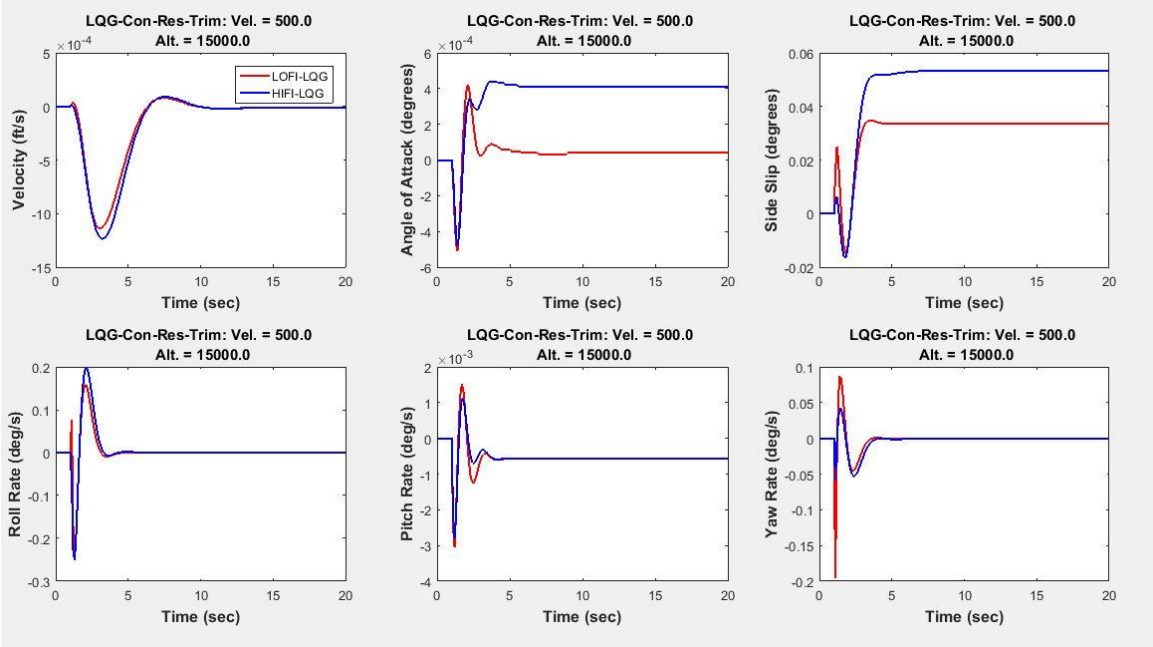

Figure 16 LQG Controller Response for F-16/MATV Force and Moment Dynamics

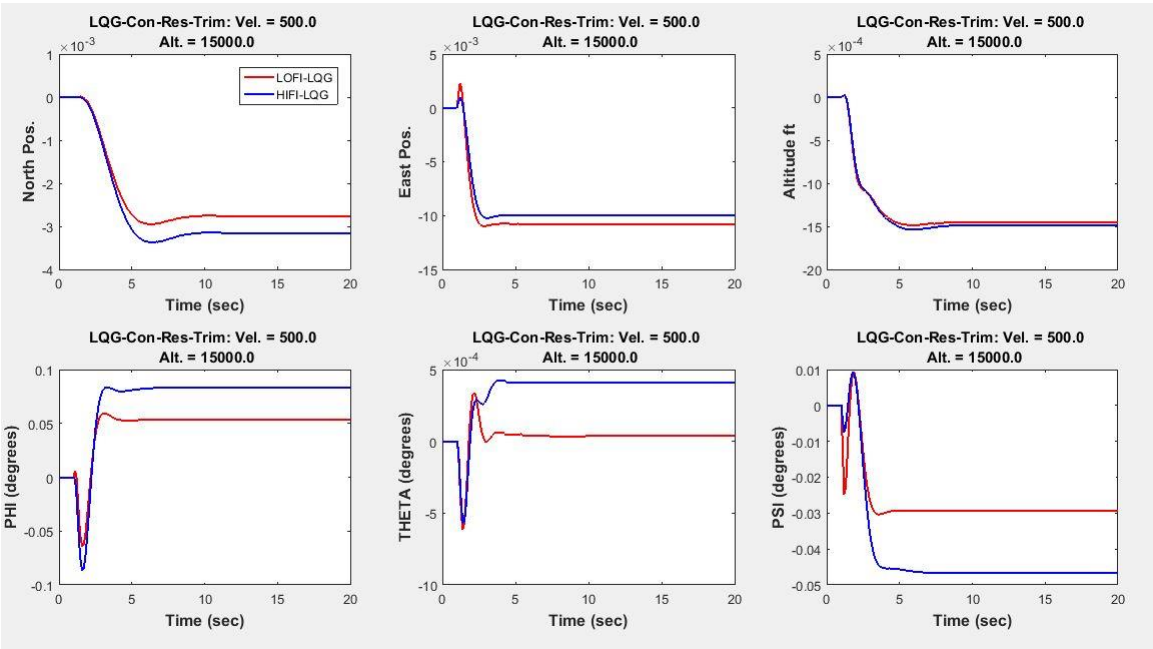

Figure 17 The LQG Controller Response for F-16/MATV flight Kinematic dynamics

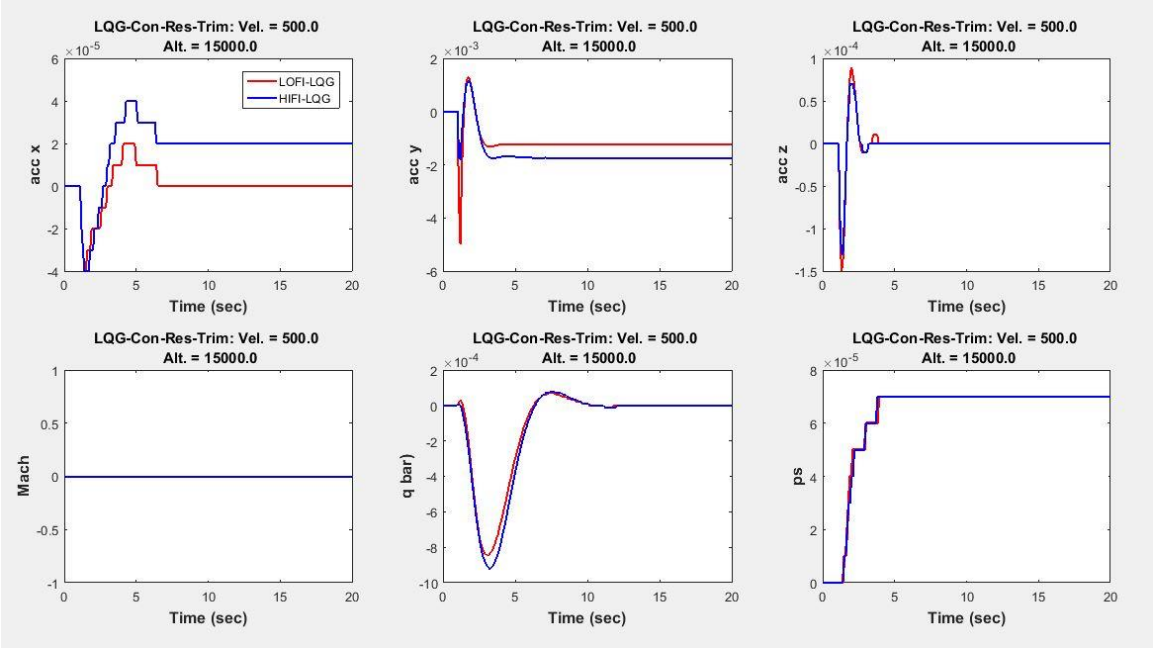

Figure 18 The LQG Controller Response for F-16/MATV Navigation dynamics

The above obtained results requires a controller with larger controller gain which is very expensive relatively.

4.4. The Discrete Linearized F-16/MATV Flight System Response

In the case of discrete system when compared to that of linearized continues time system for the system to attain its steady state condition it requires less time and the following results are obtained for the same flight condition.

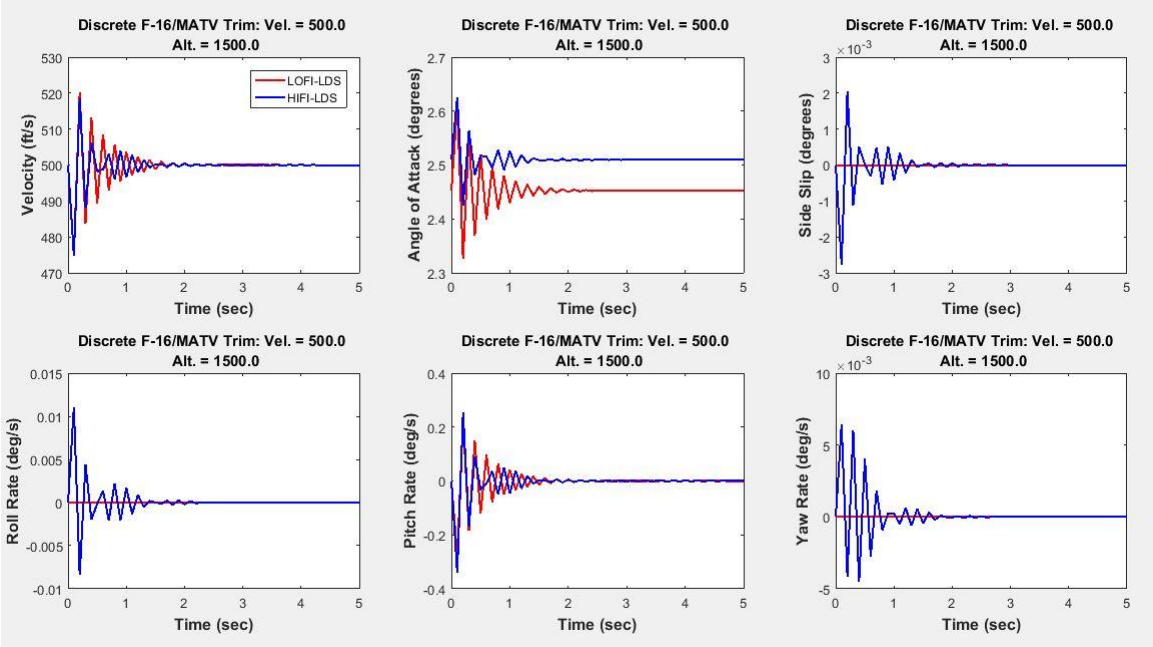

Figure 19 The Linearized Discrete time F-16/MATV Flight System Response to Force and Moment dynamics

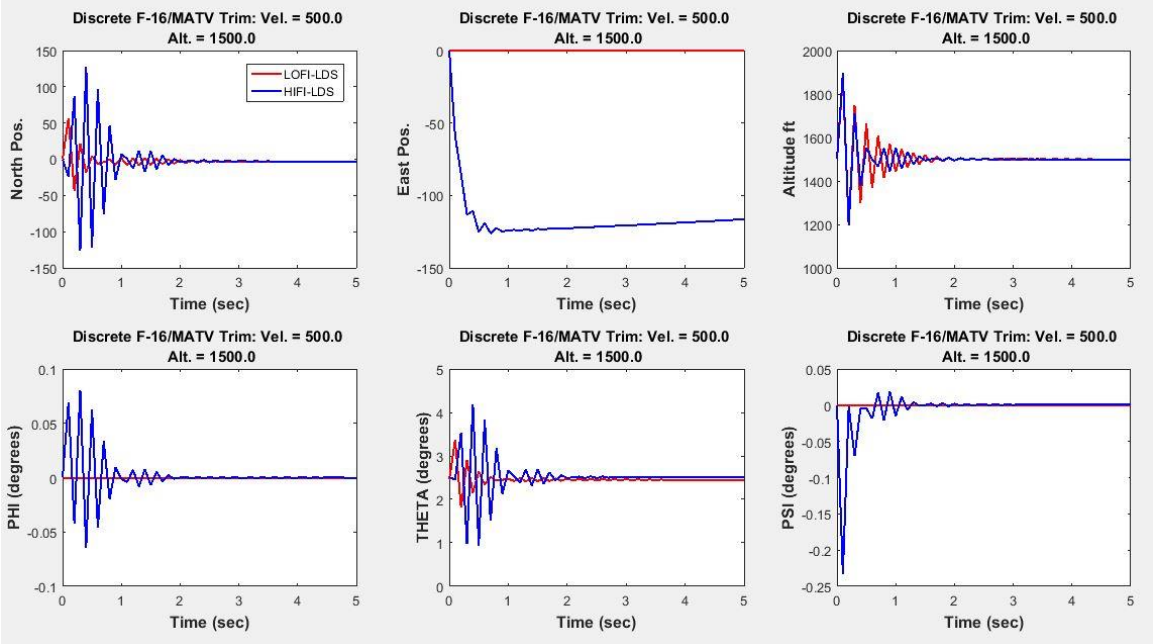

Figure 20 The Linearized Discrete time F-16/MATV Flight System Response to Kinematic dynamics

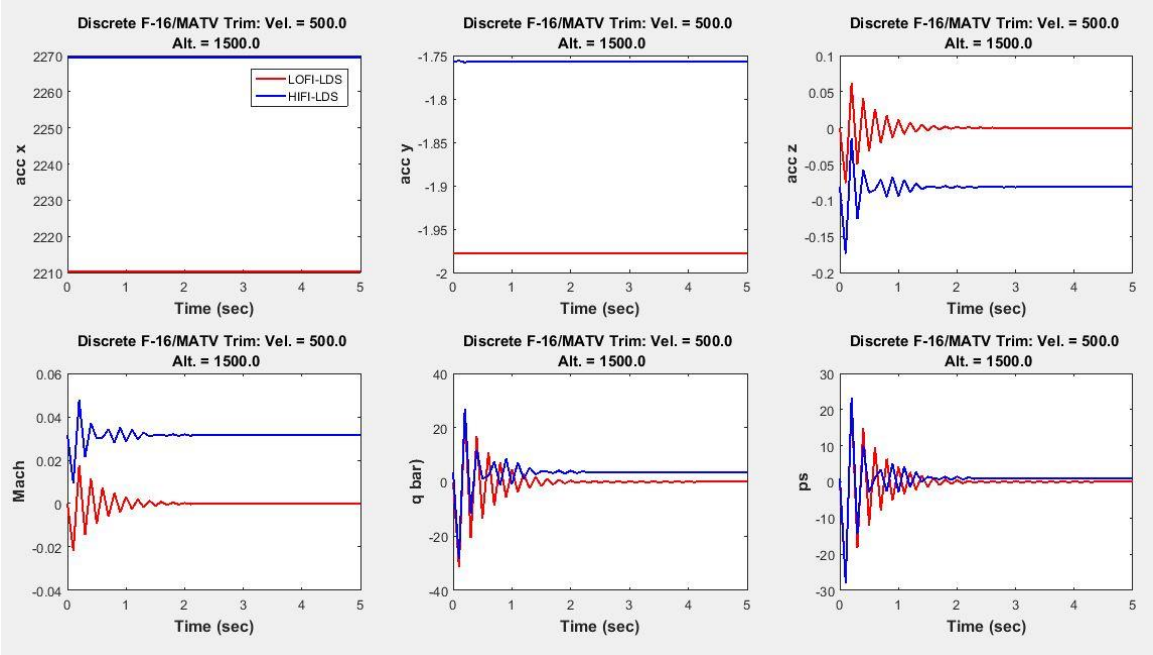

Figure 21 The Linearized Discrete time F-16/MATV Flight System Response to Navigation dynamics

4.5. The Discrete LQG Controller response to F-16/MATV Flight System dynamics

The discrete time Linear Quadratic Gaussian controller is the one with high efficiency and stability, sensitivity and robustness property as the following simulation results obtained from the MATLAB Simulink shows for the same flight condition.

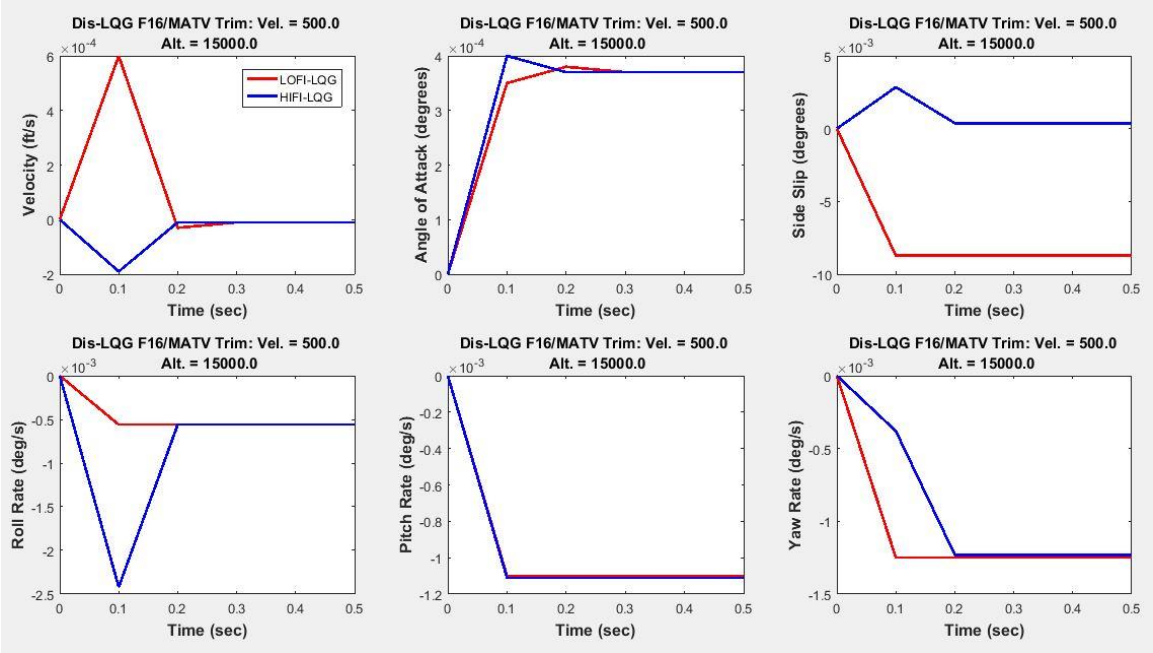

Figure 22 The Discrete LQG Controller response to F-16/MATV Flight System Response of Force and Moment dynamics

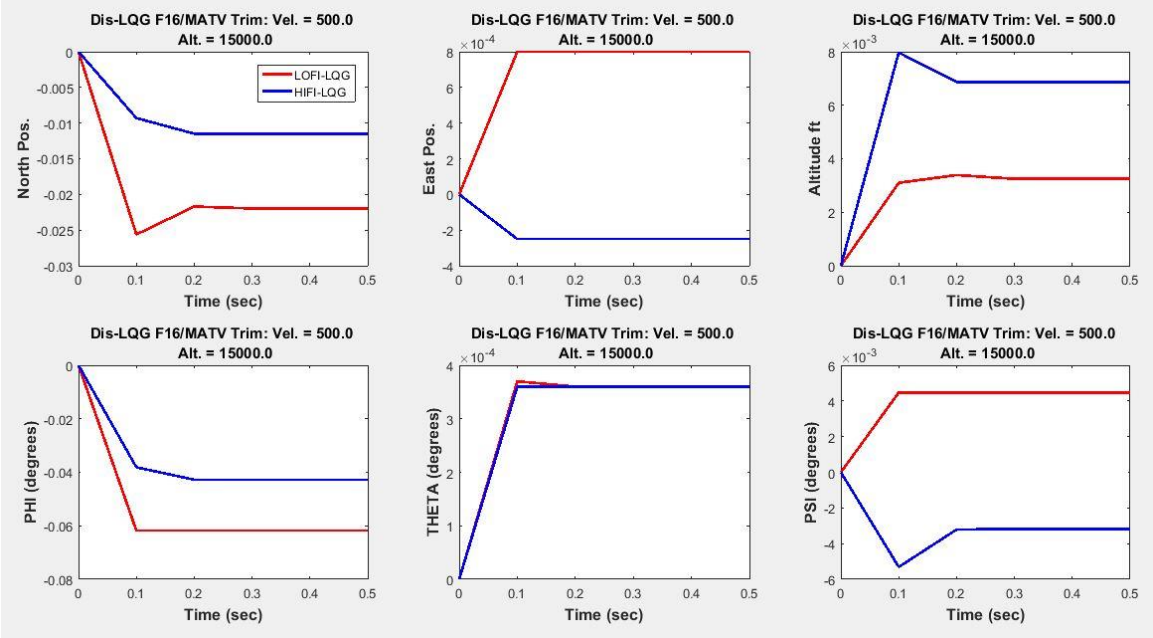

Figure 23 The Discrete LQG Controller response to F-16/MATV Flight System Response of Kinematic dynamics

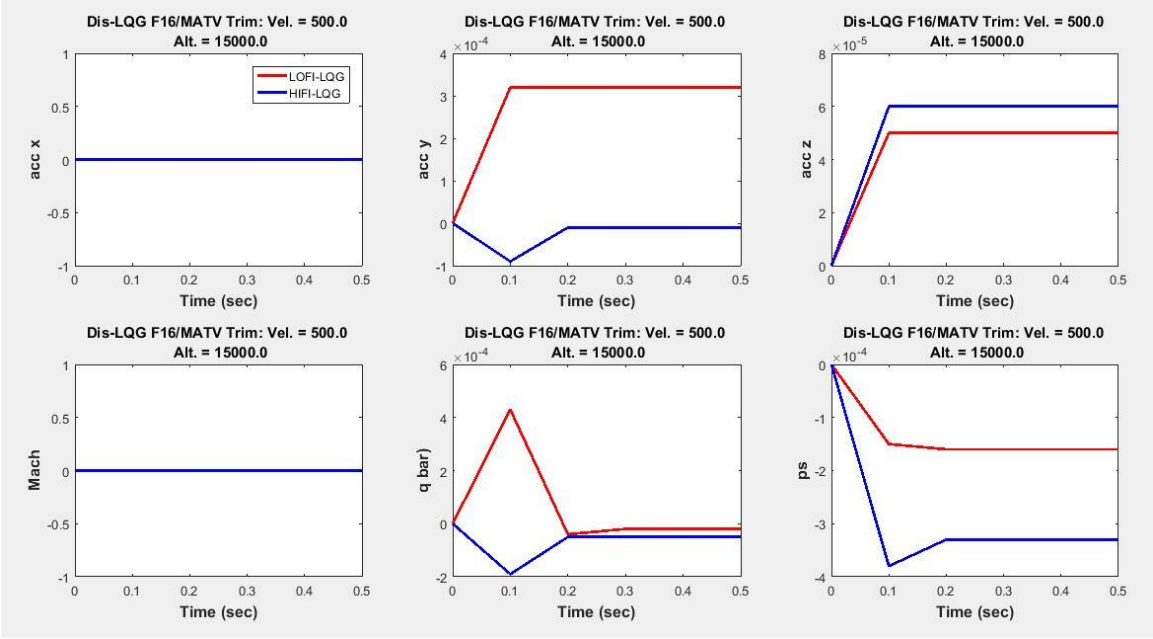

Figure 24 The Discrete LQG Controller response to F-16/MATV Flight System Response of Navigation dynamics

4.6. Comparison of Continuous LQG and Discrete LQG Controller

Transient and steady state value comparison of continuous LQG and discrete LQG is shown below are data taken for low fidelity flight condition at steady wings – level condition. This clearly shows that the discrete LQG controller is having a high performance and stability characteristics compared with the continuous LQG controller. Since the design objectives set under this thesis (i.e settling time, Rise time, peak amplitude and steady state value) are fully achieved with high robustness property by discrete LQG controller. The LQG controller almost gives the same performance as that of when all states are available for measurement and the model is perfectly known. Therefore, the designed controller is acceptable

Table 3 Comparison of Continuous LQG and Discrete LQG Controller

No	State variables	Controllers							
		Continuous LQG Controller				Discrete LQG Controller			
		Settling Time (Sec)	Rise Time (Sec)	Peek Amp	SS Value	Settling Time (Sec)	Rise Time (Sec)	Peek Amp	SS Value
1.	Total Velocity (ft/Sec)	9.9	1.6	0.00095	0	0.3	0.1	0.0006	-0.00001
2.	Angle of Attack(Degree)	9	1.3	-0.00447	0.00041	0.3	0.1	0.00035	0.00037
3.	Sideslip angel(Degree)	6.2	1.2	0.025	0.03353	0.1	0.1	-0.00873	-0.00873
4.	Roll Rate(Degree/Sec)	5.8	1.1	0.07629	-0.0005	0.1	0.1	-0.00056	-0.00056
5.	Pitch Rate(Degree/Sec)	8.4	1.1	-0.02139	-0.00057	0.1	0.1	-0.0011	-0.0011
6.	Yaw Rate(Degree/Sec)	4.7	1.1	-0.196	-0.00057	0.1	0.1	-0.00125	-0.00125
7.	North Position(ft)	8.7	2.1	0.00078	-0.00277	0.3	0.1	-0.02561	-0.02197
8.	East Position(ft)	5.3	1.2	0.00228	-0.01078	0.1	0.1	0.0008	0.0008
9.	Altitude(ft)	7.4	1.2	0.00015	-0.00149	0.4	0.1	0.0031	0.00326
10.	Roll Angle(Degree)	7	1.5	-0.05831	0.05305	0.1	0.1	-0.06179	-0.06179
11.	Pitch Angle(Degree)	8.4	1.4	-0.00538	0.00004	0.2	0.1	0.00037	0.00036
12.	Yaw Angle(Degree)	4.6	1.2	-0.02478	-0.02939	0.1	0.1	0.00447	0.00447
13.	Normal Acceleration in X-axis(N/A)	7.2	1	0.00005	0	-	-	-	0
14.	Normal Acceleration in Y-axis(N/A)	4.3	1.2	-0.00499	-0.00122	0.1	0.1	0.00032	0.00032
15.	Normal Acceleration in Z-axis(N/A)	8.7	1.3	-0.00131	0	0.1	0.1	0.0005	0.00005
16.	Match Number(N/A)	-	-	-	0	-	-	-	0
17.	Free stream Dynamic Pressure(N/A)	9.9	1.6	0.00081	0	0.3	0.1	0.00043	-0.00002
18.	Static Pressure(N/A)	6.8	2.4	0.00039	0.00007	0.2	0.1	-0.00015	-0.00016

4.7. SVD Robustness Checking Result

The result obtained when checking robustness using singular value decomposition for LQG controller shows that for the frequency lower than that 100 Hz the singularity value of LQR response and LQG Controller response are almost the same. Which shows the LQG controller is almost robust to external disturbance and after this frequency there is big gap in between them, but roll off (which is approximately 50dB/decade) is very high. Therefore, at high frequency the amplitude of unwanted signals is highly degraded. This makes the designed controller is robust to un-modeled dynamics.

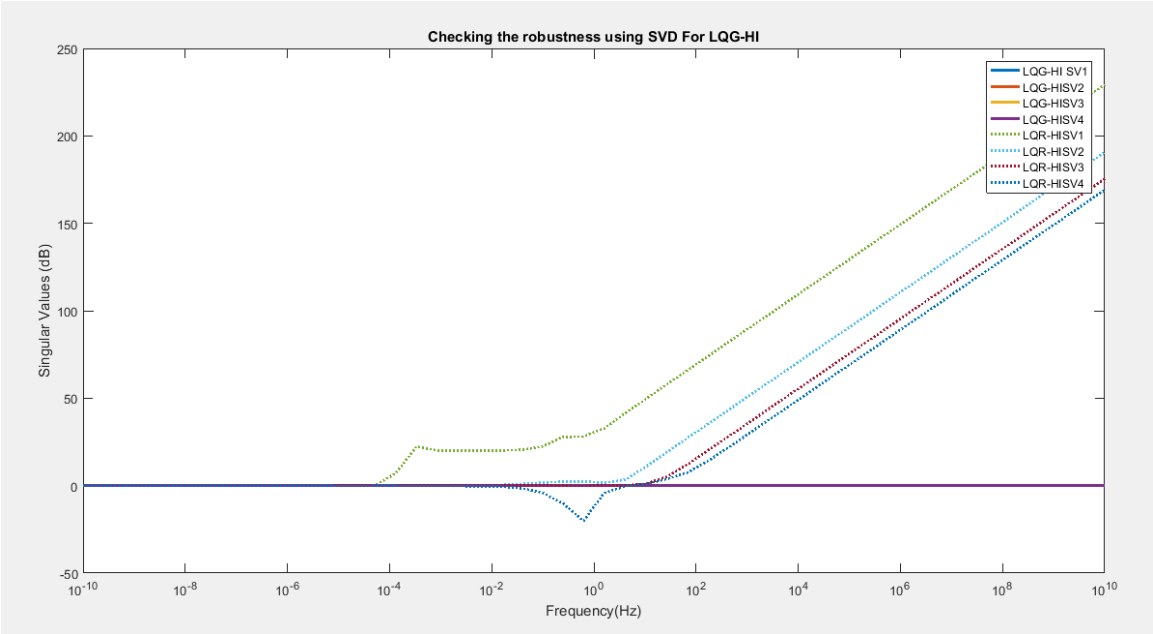

Figure 25 Robustness Checking using SVD for LQG

CHAPTER FIVE

5. Conclusions and Recommendations

5.1. Conclusions

This final year thesis presents optimal control approach, Linear Quadratic Gaussian for flight control system of F-16 fighter jet with MATV (Multi – Axis – Thrust – Vectoring) by applying both continuous time and discrete time control system.

All the required design objectives (time and frequency specifications) are met by applying both continuous and discrete time LQG controllers. Thus, LQG controller, which is based on the separation principle, is designed to stabilize the flight condition of F-16/MATV in steady state condition. Firstly, a LQR controller is designed based on the assumptions that all the states are available for measurement and the model is perfectly known. However, some states are not available for measurement and in addition to this, the model is not perfect. So these unmeasured states are estimated optimally in the presence of process and measurement noises, which come from linearization and inaccuracy of sensors. This observer is designed using Kalman filter.

The Continuous LQG Controller response predicted that even if the controller is having a good performance, stability and robustness property, it takes a large settling time with some neglected harmonic response (i.e the peek amplitude is very small for all response) which is a good characteristics of LQG controller.

The Discrete LQG Controller as compared with the continuous LQG is showed excellent property of robustness, stability and performance this makes the controller more selectable for the flight control system. For existence of large noise in the system the controller response is very quickly (i.e the settling time is very small) stabilized, thus the discrete LQG is highly recommended for flight control system.

The controllers considered satisfied the design objectives (settling time, rise time, peek amplitude, and steady state error) including robustness. Robustness of LQG is checked and

found to be close to that of LQR, which is an ideal robust optimal regulator, until 100 Hz. Beyond this frequency there is big gap in between them. Nevertheless, roll off (which is approximately 0dB/decade) is very high which degrades the amplitude of high frequency components. Below 100 Hz the roll off is greater than 0dB/decade. Therefore, the larger the bandwidth (the range of frequency for which singular values of the LQG is close to that of LQR), the smaller would be the range of frequencies over which high noise attenuation is provided by the compensator since the roll off is below 0db/decade in this range. Therefore, the selected controller with the given bandwidth is acceptable.

5.2. Recommendations

This controller is not only applicable to this system but also another systems that have nonlinear nature, have no perfect model, and for systems that affected by unmeasured states.

Even though the LQG controller in this is recommended to control flight system in steady state condition, there are controllers that are more robust than the proposed controller, since robustness is something comparative. The robustness and performance of LQG controller was done by adjusting the noise covariance matrix and state-weighting matrix by using trial and error, which is something tedious. So it is recommended to design a controller that directly addresses the problem of robustness and performance despite the presence of noise in the system. These controllers are H-infinity optimal controller, nonlinear controller or input shaping which are beyond the scope of study.

REFERENCES

[1] S. Yang, "Analysis of optimal midcourse guidance law," *IEE Transactions on Aerospace and Electronic Systems,* vol. 32, no. 1, pp. 419-425, Jan. 1996.

[2] W. DoZWERNEMAN and B. G. ELLER, "VISTA/F-16 Multi Axis Thrust Vectoring (MATV) Control law Design and Evaluation," Lockheed Fort Worth Company, 13 July, 1994.

[3] Wikipeda, "Linear-quadratic-gaussian-control," [Online]. Available: http://en.wikipedia.org/wiki/Linearquadratic-Gaussian_control. [Accessed 3 1 2018].

[4] C. Labane and M. K. Zemalache, "Aircraft Control System Using LQG and LQR Controller with Optimal Estimation-Kalman Filter Design," *3rd International Symposium on Aircraft Airworthiness, ISAA 2013,* no. 80, p. 245 – 257, 2014.

[5] L. Sonneveldt, Q. P. Chu and J. A. Mulder, "Nonlinear Flight Control Design Using Constrained Adaptive Backstepping," *JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS*, pp. 324 - 325, 2007.

[6] D. McLEAN, Automatic Fight Control System, Southampton, UK: Prentice Hall International, 1990.

[7] J. Slotine and W. Li, Applied Nonlinear Control, Upper Saddle River, NJ: Prentice–Hall, 1991.

[8] A. Isidori, Nonlinear Control Systems, New York: Springer, 1995.

[9] D. M. Littleboy and P. R. Smith, "Using Bifurcation Methods to Aid Nonlinear Dynamic Inversion Control Law Design," *Journal of Guidance, Control, and Dynamics,* vol. 21, p. 632–638, 1998.

[10] E. Lavretsky and N. Hovakimyan, "Positive μ-modification for Stable Adaptation in

Dynamic Inversion Based Adaptive Control with Input Saturation," in *In Proc. Of the American Control Conference*, 2005.

[11] R. Kalman and E. Bertram, "Control System Analysis and Design Via the 'Second Method' of Lyapunov,Part I : Continuous Time Systems," *Journal of Basic Engineering,TRANS. ASME, Series D,,* vol. 82, pp. 371-393, June I960.

[12] P. C Parks, "Liapunov Redesign of Model Reference Adaptive Control Systems," *IEEE Transactionson Automatic Control,* Vols. AC-11, pp. 362-367, July 1966.

[13] C. A. Winsqr and R. J. Roy, "Design of Model - Reference Adaptive Control Systems by Liapunov's Second Method," *IEEE, Transactions on Automatic Control,* Vols. AC-13, Apr. 1968.

[14] K. Ki-Seok, L. Keum-Jin and K. Youdan, "Reconfigurable Flight Control System Design Using Direct Adaptive Method," *JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS,* vol. 46, July– August 2003.

[15] K. Ki-Seok, L. Keum-Jin and K. Youdan, "MODEL FOLLOWING RECONFIGURABLE FLIGHT CONTROL SYSTEM DESIGN USING DIRECT ADAPTIVE SCHEME," *AIAA Guidance, Navigation, and Control Conference and Exhibit,* pp. 151-742, 5-8 August 2002.

[16] K. S. Hyung and Youdan, "Reconfigurable Flight Control System Design Using Discrete Model Reference Adaptive Control," *AIAA Guidance, Navigation, and Control Conference and Exhibit,* 15 - 18 August 2005.

[17] J. Shaji and B. Aswin R, "Pitch Control of Aircraft Using LQR & LQG Control," *International Journal of Advanced Research in Electrical,Electronics and Instrumentation Engineering,* vol. 4, no. 8, p. 2320 – 3765, August 2015.

[18] S. Das and K. Halder, "Missile Attitude Control via a Hybrid LQG-LTR-LQI Control Scheme with Optimum Weight Selection," *journal of Guidance and control,* 2016.

[19] L. Sonneveldt, Adaptive Backstepping Flight Control for Modern Fighter Aircraft, Zutphen, Netherlands: W¨ohrmann Print Service, 2010.

[20] J. A. Mulder, W. H. J. J. v. Staveren, J. C. v. d. Vaart and E. d. Weerdt, "Flight Dynamics, Lecture Notes AE3-302 Technical report," Delft University of Technology, 2006.

[21] L. Nguyen, M. Ogburn, W. Gilbert, K. Kibler, P. Brown and P. Deal, "Technical Paper 1538: Simulator study of stall/post-stall characteristics of a fighter airplane with relaxed longitudinal static stability," NASA, 1979.

[22] E. v. Oort, Q. Chu, J. Mulder and T. v. d. Boom, "Robust Model Predictive Control of a Feedback Linearized Nonlinear F-16/MATV Aircraft Model," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Keystone, Colorado, 21 - 24 August 2006.

[23] A. Ferreira, J. Barreiros, W. B. Jr and J. R. Brito-de-Souza, "A robust adaptive LQG/LTR TCSC controller applied to damp power system oscillations," *Electric Power Systems Research*, vol. 77, p. 956–964, 2007.

APPENDICES

Appendix A

```
#include <math.h>
#include "mex.h"
/*  Merging the nlplant.c (lofi) and nlplant_hifi.c to use
    same equations of motion, navigation equations and use
    own look-up tables decided by a flag.              */
void nlplant(double*,double*);
void atmos(double,double,double*);          /* Used by both */
void accels(double*,double*,double*);       /* Used by both */
#include "lofi_F16_AeroData.c"              /* LOFI Look-up header file*/
#include "hifi_F16_AeroData.c"              /* HIFI Look-up header file*/
/*### Added for mex function in MATLAB ###*/
int fix(double);
int sign(double);
void mexFunction(int nlhs, mxArray *plhs[],
                 int nrhs, const mxArray *prhs[])
{
#define XU prhs[0]
#define XDOTY plhs[0]
int i;
double *xup, *xdotp;
if (mxGetM(XU)==18 && mxGetN(XU)==1){
      /* Calling Program */
      xup = mxGetPr(XU);
      XDOTY = mxCreateDoubleMatrix(18, 1, mxREAL);
      xdotp = mxGetPr(XDOTY);
      nlplant(xup,xdotp);
} /* End if */
else{
      mexErrMsgTxt("Input and/or output is wrong size.");
} /* End else */

} /* end mexFunction */
```

```
/*########################################*/
void nlplant(double *xu, double *xdot){
  int fi_flag;
  /* #include f16_constants */
  double g    = 32.17;          /* gravity, ft/s^2 */
  double m    = 636.94;         /* mass, slugs */
  double B    = 30.0;             /* span, ft */
  double S    = 300.0;            /* planform area, ft^2 */
  double cbar = 11.32;          /* mean aero chord, ft */
  double xcgr = 0.35;      /* reference center of gravity as a fraction of
cbar */
  double xcg  = 0.30;      /* center of gravity as a fraction of cbar. */
  double Heng = 0.0;              /* turbine momentum along roll axis. */
  double pi   = acos(-1);
  double r2d;                   /* radians to degrees */
/*NasaData        %translated via eq. 2.4-6 on pg 80 of Stevens and Lewis*/
  double Jy  = 55814.0;           /* slug-ft^2 */
  double Jxz = 982.0;             /* slug-ft^2 */
  double Jz  = 63100.0;           /* slug-ft^2 */
  double Jx  = 9496.0;            /* slug-ft^2 */
  double *temp;
  double npos, epos, alt, phi, theta, psi, vt, alpha, beta, P, Q, R;
  double sa, ca, sb, cb, tb, st, ct, tt, sphi, cphi, spsi, cpsi;
  double T, el, ail, rud, dail, drud, lef, dlef;
  double qbar, mach, ps;
  double U, V, W, Udot,Vdot,Wdot;
  double L_tot, M_tot, N_tot, denom;
  double Cx_tot, Cx, delta_Cx_lef, dXdQ, Cxq, delta_Cxq_lef;
  double Cz_tot, Cz, delta_Cz_lef, dZdQ, Czq, delta_Czq_lef;
  double Cm_tot, Cm, eta_el, delta_Cm_lef, dMdQ, Cmq, delta_Cmq_lef,
delta_Cm, delta_Cm_ds;
  double Cy_tot, Cy, delta_Cy_lef, dYdail, delta_Cy_r30, dYdR, dYdP;
  double delta_Cy_a20, delta_Cy_a20_lef, Cyr, delta_Cyr_lef, Cyp,
delta_Cyp_lef;
  double Cn_tot, Cn, delta_Cn_lef, dNdail, delta_Cn_r30, dNdR, dNdP,
delta_Cnbeta;
```

```
  double delta_Cn_a20, delta_Cn_a20_lef, Cnr, delta_Cnr_lef, Cnp,
delta_Cnp_lef;
  double Cl_tot, Cl, delta_Cl_lef, dLdail, delta_Cl_r30, dLdR, dLdP,
delta_Clbeta;
  double delta_Cl_a20, delta_Cl_a20_lef, Clr, delta_Clr_lef, Clp,
delta_Clp_lef;
  temp = (double *)malloc(9*sizeof(double));  /*size of 9.1 array*/
  r2d  = 180.0/pi;     /* radians to degrees */
  /*   States */
  npos  = xu[0];   /* north position */
  epos  = xu[1];   /* east position */
  alt   = xu[2];   /* altitude */
  phi   = xu[3];   /* orientation angles in rad. */
  theta = xu[4];
  psi   = xu[5];
  vt    = xu[6];     /* total velocity */
  alpha = xu[7]*r2d; /* angle of attack in degrees */
  beta  = xu[8]*r2d; /* sideslip angle in degrees */
  P     = xu[9];     /* Roll Rate --- rolling  moment is Lbar */
  Q     = xu[10];    /* Pitch Rate--- pitching moment is M */
  R     = xu[11];    /* Yaw Rate  --- yawing   moment is N */
  sa    = sin(xu[7]); /* sin(alpha) */
  ca    = cos(xu[7]); /* cos(alpha) */
  sb    = sin(xu[8]); /* sin(beta)  */
  cb    = cos(xu[8]); /* cos(beta)  */
  tb    = tan(xu[8]); /* tan(beta)  */
  st    = sin(theta);
  ct    = cos(theta);
  tt    = tan(theta);
  sphi  = sin(phi);
  cphi  = cos(phi);
  spsi  = sin(psi);
  cpsi  = cos(psi);

  if (vt <= 0.01) {vt = 0.01;}

  /* %%%%%%%%%%%%%%%%%%  Control inputs  %%%%%%%%%%%%%%%%%% */
```

```
  T     = xu[12];   /* thrust */
  el    = xu[13];   /* Elevator setting in degrees. */
  ail   = xu[14];   /* Ailerons mex setting in degrees. */
  rud   = xu[15];   /* Rudder setting in degrees. */
  lef   = xu[16];   /* Leading edge flap setting in degrees */
  fi_flag = xu[17]/1;       /* fi_flag */
  /* dail  = ail/20.0;   aileron normalized against max angle */
  /* The aileron was normalized using 20.0 but the NASA report and
     S&L both have 21.5 deg. as maximum deflection. */
  /* As a result... */
  dail  = ail/21.5;
  drud  = rud/30.0;  /* rudder normalized against max angle */
  dlef  = (1 - lef/25.0);  /* leading edge flap normalized against max angle
*/
  /* %%%%%%%%%%%%%%%%%%   Atmospheric effects  sets dynamic pressure and mach
number*/
atmos(alt,vt,temp);
   mach = temp[0];
   qbar = temp[1];
   ps   = temp[2];
  /* %%%%%%%%%%%%%%%%%%
     Navigation Equations
     %%%%%%%%%%%%%%%%%% */
   U = vt*ca*cb;  /* directional velocities. */
   V = vt*sb;
   W = vt*sa*cb;
/* nposdot */
xdot[0] = U*(ct*cpsi) +
            V*(sphi*cpsi*st - cphi*spsi) +
            W*(cphi*st*cpsi + sphi*spsi);
/* eposdot */
xdot[1] = U*(ct*spsi) +
            V*(sphi*spsi*st + cphi*cpsi) +
            W*(cphi*st*spsi - sphi*cpsi);
/* altdot */
xdot[2] = U*st - V*(sphi*ct) - W*(cphi*ct);
```

```
  /* %%%%%%%%%%%%%%%%%%%%  Kinematic equations   %%%%%%%%%%%%%%%%%%%% */
/* phidot */
xdot[3] = P + tt*(Q*sphi + R*cphi);

/* theta dot */
xdot[4] = Q*cphi - R*sphi;

/* psidot */
xdot[5] = (Q*sphi + R*cphi)/ct;

/* %%%%%%%%%%%%%%%%%%      Table lookup    %%%%%%%%%%%%%%%%%% */

if (fi_flag == 1)          /* HIFI Table */
{
    hifi_C(alpha,beta,el,temp);
        Cx = temp[0];
        Cz = temp[1];
        Cm = temp[2];
        Cy = temp[3];
        Cn = temp[4];
        Cl = temp[5];

    hifi_damping(alpha,temp);
        Cxq = temp[0];
        Cyr = temp[1];
        Cyp = temp[2];
        Czq = temp[3];
        Clr = temp[4];
        Clp = temp[5];
        Cmq = temp[6];
        Cnr = temp[7];
        Cnp = temp[8];

    hifi_C_lef(alpha,beta,temp);
        delta_Cx_lef = temp[0];
```

```
    delta_Cz_lef = temp[1];
    delta_Cm_lef = temp[2];
    delta_Cy_lef = temp[3];
    delta_Cn_lef = temp[4];
    delta_Cl_lef = temp[5];

hifi_damping_lef(alpha,temp);
    delta_Cxq_lef = temp[0];
    delta_Cyr_lef = temp[1];
    delta_Cyp_lef = temp[2];
    delta_Czq_lef = temp[3];
    delta_Clr_lef = temp[4];
    delta_Clp_lef = temp[5];
    delta_Cmq_lef = temp[6];
    delta_Cnr_lef = temp[7];
    delta_Cnp_lef = temp[8];

hifi_rudder(alpha,beta,temp);
    delta_Cy_r30 = temp[0];
    delta_Cn_r30 = temp[1];
    delta_Cl_r30 = temp[2];

hifi_ailerons(alpha,beta,temp);
    delta_Cy_a20     = temp[0];
    delta_Cy_a20_lef = temp[1];
    delta_Cn_a20     = temp[2];
    delta_Cn_a20_lef = temp[3];
    delta_Cl_a20     = temp[4];
    delta_Cl_a20_lef = temp[5];

hifi_other_coeffs(alpha,el,temp);
    delta_Cnbeta = temp[0];
    delta_Clbeta = temp[1];
    delta_Cm     = temp[2];
    eta_el       = temp[3];
    delta_Cm_ds  = 0;        /* ignore deep-stall effect */
```

```
}

else if (fi_flag == 0)
{
/* ##############################################
   ##########LOFI Table Look-up #################
   ##############################################*/

/* The lofi model does not include the
   leading edge flap.  All terms multiplied
   dlef have been set to zero but just to
   be sure we will set it to zero. */

    dlef = 0.0;

    damping(alpha,temp);
        Cxq = temp[0];
        Cyr = temp[1];
        Cyp = temp[2];
        Czq = temp[3];
        Clr = temp[4];
        Clp = temp[5];
        Cmq = temp[6];
        Cnr = temp[7];
        Cnp = temp[8];

    dmomdcon(alpha,beta, temp);
        delta_Cl_a20 = temp[0];     /* Formerly dLda in nlplant.c */
        delta_Cl_r30 = temp[1];     /* Formerly dLdr in nlplant.c */
        delta_Cn_a20 = temp[2];     /* Formerly dNda in nlplant.c */
        delta_Cn_r30 = temp[3];     /* Formerly dNdr in nlplant.c */

    clcn(alpha,beta,temp);
        Cl = temp[0];
        Cn = temp[1];
```

```
    cxcm(alpha,el,temp);
        Cx = temp[0];
        Cm = temp[1];

    Cy = -.02*beta + .021*dail + .086*drud;

    cz(alpha,beta,el,temp);
        Cz = temp[0];
/*################################################
  ##  Set all higher order terms of hifi that are ##
  ##  not applicable to lofi equal to zero. ########
  ################################################*/

        delta_Cx_lef    = 0.0;
        delta_Cz_lef    = 0.0;
        delta_Cm_lef    = 0.0;
        delta_Cy_lef    = 0.0;
        delta_Cn_lef    = 0.0;
        delta_Cl_lef    = 0.0;
        delta_Cxq_lef   = 0.0;
        delta_Cyr_lef   = 0.0;
        delta_Cyp_lef   = 0.0;
        delta_Czq_lef   = 0.0;
        delta_Clr_lef   = 0.0;
        delta_Clp_lef   = 0.0;
        delta_Cmq_lef   = 0.0;
        delta_Cnr_lef   = 0.0;
        delta_Cnp_lef   = 0.0;
        delta_Cy_r30    = 0.0;
        delta_Cy_a20    = 0.0;
        delta_Cy_a20_lef= 0.0;
        delta_Cn_a20_lef= 0.0;
        delta_Cl_a20_lef= 0.0;
        delta_Cnbeta    = 0.0;
        delta_Clbeta    = 0.0;
```

```
        delta_Cm        = 0.0;
        eta_el          = 1.0;     /* Needs to be one. See equation for
Cm_tot*/
        delta_Cm_ds     = 0.0;

/*################################################
  ################################################*/
}
/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
compute Cx_tot, Cz_tot, Cm_tot, Cy_tot, Cn_tot, and Cl_tot
(as on NASA report p37-40)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */

dXdQ = (cbar/(2*vt))*(Cxq + delta_Cxq_lef*dlef);

Cx_tot = Cx + delta_Cx_lef*dlef + dXdQ*Q;

   /* ZZZZZZZZ Cz_tot ZZZZZZZZ */

dZdQ = (cbar/(2*vt))*(Czq + delta_Cz_lef*dlef);

Cz_tot = Cz + delta_Cz_lef*dlef + dZdQ*Q;

   /* MMMMMMMM Cm_tot MMMMMMMM */

dMdQ = (cbar/(2*vt))*(Cmq + delta_Cmq_lef*dlef);

Cm_tot = Cm*eta_el + Cz_tot*(xcgr-xcg) + delta_Cm_lef*dlef + dMdQ*Q +
delta_Cm + delta_Cm_ds;

   /* YYYYYYYY Cy_tot YYYYYYYY */

dYdail = delta_Cy_a20 + delta_Cy_a20_lef*dlef;

dYdR = (B/(2*vt))*(Cyr + delta_Cyr_lef*dlef);
```

```
dYdP = (B/(2*vt))*(Cyp + delta_Cyp_lef*dlef);

Cy_tot = Cy + delta_Cy_lef*dlef + dYdail*dail + delta_Cy_r30*drud + dYdR*R +
dYdP*P;

   /* NNNNNNNN Cn_tot NNNNNNNN */

dNdail = delta_Cn_a20 + delta_Cn_a20_lef*dlef;

dNdR = (B/(2*vt))*(Cnr + delta_Cnr_lef*dlef);

dNdP = (B/(2*vt))*(Cnp + delta_Cnp_lef*dlef);

Cn_tot = Cn + delta_Cn_lef*dlef - Cy_tot*(xcgr-xcg)*(cbar/B) + dNdail*dail +
delta_Cn_r30*drud + dNdR*R + dNdP*P + delta_Cnbeta*beta;

   /* LLLLLLLL Cl_tot LLLLLLLL */

dLdail = delta_Cl_a20 + delta_Cl_a20_lef*dlef;

dLdR = (B/(2*vt))*(Clr + delta_Clr_lef*dlef);

dLdP = (B/(2*vt))*(Clp + delta_Clp_lef*dlef);

Cl_tot = Cl + delta_Cl_lef*dlef + dLdail*dail + delta_Cl_r30*drud + dLdR*R +
dLdP*P + delta_Clbeta*beta;

   /* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      compute Udot,Vdot, Wdot,(as on NASA report p36)
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */

Udot = R*V - Q*W - g*st + qbar*S*Cx_tot/m + T/m;
```

```
Vdot = P*W - R*U + g*ct*sphi + qbar*S*Cy_tot/m;

Wdot = Q*U - P*V + g*ct*cphi + qbar*S*Cz_tot/m;

   /* %%%%%%%%%%     vt_dot equation (from S&L, p82)      %%%%%%%%%%%%% */

xdot[6] = (U*Udot + V*Vdot + W*Wdot)/vt;

   /* %%%%%%%%%%      alpha_dot equation      %%%%%%%%%%%%%%%%%% */

xdot[7] = (U*Wdot - W*Udot)/(U*U + W*W);

  /* %%%%%%%%%%%%%%%     beta_dot equation     %%%%%%%%%%%%%%%%% */

xdot[8] = (Vdot*vt - V*xdot[6])/(vt*vt*cb);

  /* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     compute Pdot, Qdot, and Rdot (as in Stevens and Lewis p32)
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */

L_tot = Cl_tot*qbar*S*B;       /* get moments from coefficients */
M_tot = Cm_tot*qbar*S*cbar;
N_tot = Cn_tot*qbar*S*B;

denom = Jx*Jz - Jxz*Jxz;

  /* %%%%%%%%%%%%%%%%%%%%%%     Pdot     %%%%%%%%%%%%%%%%%%%%%%% */

xdot[9] =  (Jz*L_tot + Jxz*N_tot - (Jz*(Jz-Jy)+Jxz*Jxz)*Q*R + Jxz*(Jx-
Jy+Jz)*P*Q + Jxz*Q*Heng)/denom;

  /* %%%%%%%%%%%%%%%%%%%%%%     Qdot     %%%%%%%%%%%%%%%%%%%%%%% */

xdot[10] = (M_tot + (Jz-Jx)*P*R - Jxz*(P*P-R*R) - R*Heng)/Jy;
```

```
  /* %%%%%%%%%%%%%%%%%%%%%%%%      Rdot      %%%%%%%%%%%%%%%%%%%%%%%% */

xdot[11] = (Jx*N_tot + Jxz*L_tot + (Jx*(Jx-Jy)+Jxz*Jxz)*P*Q - Jxz*(Jx-
Jy+Jz)*Q*R +  Jx*Q*Heng)/denom;

/*### Create accelerations anx_cg, any_cg */
/*### ans anz_cg as outputs ###############*/
/*########################################*/

accels(xu,xdot,temp);

xdot[12]  = temp[0];    /* anx_cg */
xdot[13]  = temp[1];    /* any_cg */
xdot[14]  = temp[2];    /* anz_cg */
xdot[15]  = mach;
xdot[16]  = qbar;
xdot[17]  = ps;
/*########################################*/
free(temp);

}; /*##### END of nlplant() ####*/
 /*### Called Sub-Functions  ###############*/
/* Function for mach and qbar */
/*########################################*/
void atmos(double alt, double vt, double *coeff ){
    double rho0 = 2.377e-3;
    double tfac, temp, rho, mach, qbar, ps;
    tfac =1 - .703e-5*(alt);
    temp = 519.0*tfac;
    if (alt >= 35000.0) {
       temp=390;
    }
    rho=rho0*pow(tfac,4.14);
    mach = (vt)/sqrt(1.4*1716.3*temp);
    qbar = .5*rho*pow(vt,2);
```

```
    ps   = 1715.0*rho*temp;
    if (ps == 0){
      ps = 1715;
      }
    coeff[0] = mach;
    coeff[1] = qbar;
    coeff[2] = ps;
}

/*### Port from MATLAB fix() function ####*/

int fix(double in){
    int out;

    if (in >= 0.0){
       out = (int)floor(in);
    }
    else if (in < 0.0){
       out = (int)ceil(in);
    }

    return out;
}

/* port from MATLAB sign() function */
int sign(double in){
    int out;

    if (in > 0.0){
       out = 1;
    }
    else if (in < 0.0){
       out = -1;
    }
    else if (in == 0.0){
       out = 0;
```

```
    }
    return out;
}

/*### Calculate accelerations from states */
/*### and state derivatives. ############ */
/*########################################*/

void accels(double *state,
            double *xdot,
            double *y)
{

#define grav 32.174

double sina, cosa, sinb, cosb ;
double vel_u, vel_v, vel_w ;
double u_dot, v_dot, w_dot ;
double nx_cg, ny_cg, nz_cg ;

sina = sin(state[7]) ;
cosa = cos(state[7]) ;
sinb = sin(state[8]) ;
cosb = cos(state[8]) ;
vel_u = state[6]*cosb*cosa ;
vel_v = state[6]*sinb ;
vel_w = state[6]*cosb*sina ;
u_dot =          cosb*cosa*xdot[6]
      - state[6]*sinb*cosa*xdot[8]
      - state[6]*cosb*sina*xdot[7] ;
v_dot =          sinb*xdot[6]
      + state[6]*cosb*xdot[8] ;
w_dot =          cosb*sina*xdot[6]
      - state[6]*sinb*sina*xdot[8]
      + state[6]*cosb*cosa*xdot[7] ;
nx_cg = 1.0/grav*(u_dot + state[10]*vel_w - state[11]*vel_v)
```

```
      + sin(state[4]) ;
ny_cg = 1.0/grav*(v_dot + state[11]*vel_u - state[9]*vel_w)
      - cos(state[4])*sin(state[3]) ;
nz_cg = -1.0/grav*(w_dot + state[9]*vel_v - state[10]*vel_u)
      + cos(state[4])*cos(state[3]) ;

y[0] = nx_cg ;
y[1] = ny_cg ;
y[2] = nz_cg ;

}
```

Appendix B

```
/*##########################################*/
%%================================================
%                   Jimma University
%
%            Jimma Institute of Technology
%       Control and Instrumentation Engineering
%   Project on Flight Control system Design Using LQG
%%%
%     MATLAB Script File used to run the
%     non-linear F-16/MATV Simulation.  The results
%     will also be saved to a file and plotted.
%
%       Author: Kassahun Berisha {Msc}
%               Year: 2019
%================================================
%%
clear;
clc;

global altitude fi_type velocity fi_flag_Simulink FC_flag;
global surface1 surface2 surface3;
```

```
global ElevatorDis AileronDis RudderDis;


surface1 = 'ele_';
surface2 = 'ail_';
surface3 = 'rud_';
newline = sprintf('\n');
disp('    This is Nonlinear F-16/MATV Simulation.');
disp('The simulation will begin by asking you for the flight ');
disp('conditions for which the simulation will be performed.');
disp(newline);
disp('Accpetable values for flight condition parameters are:');
disp(newline);
disp('                                  Model');
disp('  Variable                LOFI            HIFI');
disp('               Units   Min     Max     Min     Max');
disp('  Altitude:    ft      5000    40000   5000    40000');
disp('  AOA          deg    -25      45     -10      90');
disp('  Thrust       lbs     100    100000   100    100000');
disp('  Elevator     deg    -25.0    25.0   -25.0    25.0');
disp('  Aileron      deg    -21.5    21.5   -21.5    21.5');
disp('  Rudder       deg    -30      30     -30      30');
disp('  Velocity     ft/s    300     900     300     900');
disp(newline);
disp('The flight condition you choose will be used to trim the F-16/MATV.');
disp('Note:  The trim routine will trim to the desired');
disp('altitude and velocity.  All other parameters');
disp('will be varied until level flight is achieved.  ');
disp(newline);
%% Ask user which simulation to run.
disp('Which model would you like to use to trim the aircraft:')
disp('  1. Low Fidelity F-16/MATV Trim')
disp('  2. High Fidelity F-16/MATV Trim')
fi_flag = input('Your Selection:  ');
disp(newline);
%% Determine from flag the correct simulation.
if fi_flag == 1;
   fi_type = 'lofi';
```

```
   fi_flag_Simulink = 0;
 elseif fi_flag == 2;
   fi_type = 'hifi';
   fi_flag_Simulink = 1;
 else
   disp('Invalid selection');

 end
 %% Trim aircraft to desired altitude and velocity
%%
altitude = input('Enter the altitude for the simulation (ft)   :  ');
velocity = input('Enter the velocity for the simulation (ft/s):  ');
disp(newline);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initialize some varibles used to create disturbances. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 DisEle_1 = 0;    DisEle_2 = 0;    DisEle_3 = 0;
DisAil_1 = 0;    DisAil_2 = 0;    DisAil_3 = 0;
DisRud_1 = 0;    DisRud_2 = 0;    DisRud_3 = 0;
ElevatorDis = 0; AileronDis = 0;  RudderDis = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Find out which surface to create a disturbance on.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
dis_flag = input('Would you like to create a disturbance on a surface (y/n):
', 's');

if dis_flag == 'y'
  ElevatorDis = input('Enter the elevator distrubance deflection
(deg) :  ');
  DisEle_1 = ElevatorDis;    DisEle_2 = -2*ElevatorDis;    DisEle_3 =
ElevatorDis; surfacedef = 'elevator';

  AileronDis = input('Enter the aileron distrubance deflection
(deg) :  ');
```

```
  DisAil_1 = AileronDis;    DisAil_2 = -2*AileronDis;    DisAil_3 =
AileronDis; surfacedef = 'aileron';

  RudderDis = input('Enter the rudder distrubance deflection
(deg) :  ');
  DisRud_1 = RudderDis;    DisRud_2 = -2*RudderDis;    DisRud_3 = RudderDis;
surfacedef = 'rudder';

elseif dis_flag == 'n'
  surfacedef = 'none';  %do nothing
else
  disp('Invalid Selection');

end
disp(newline);

%%
delta_T = 0.001;

TStart = 0; TFinal = 30;

%% flight condition.  If the F16/MATV does not trim
%% Change these values.
thrust = 5000;          % thrust, lbs
elevator = -0.09;       % elevator, degrees
alpha = 8.49;           % AOA, degrees
rudder = -0.01;         % rudder angle, degrees
aileron = 0.01;         % aileron, degrees
%%
%
disp('At what flight condition would you like to trim the F-16?');
disp('1.  Steady Wings-Level Flight.');
disp('2.  Steady Turning Flight.');
disp('3.  Steady Pull-Up Flight.');
disp('4.  Steady Roll Flight.');
FC_flag = input('Your Selection:  ');
```

```
disp(newline);
%%
[trim_state, trim_thrust, trim_control, dLEF, UX] = trim_F16(thrust,
elevator, alpha, aileron, rudder, velocity, altitude);

sim( 'F16Block' ,[TStart TFinal]);

trim_file = sprintf('%s%.3f%s%.3f%s%.3f_%smodel_alt%0.f_vel%.0f.txt',
surface1, ElevatorDis, surface2, AileronDis, surface3, RudderDis, fi_type,
altitude, velocity);

fid_trim = fopen(trim_file, 'w');

heading1 = sprintf('%% \n\t\t  %s DATA Trim-Doublet on %s: Alt %.0f, Alpha
%.0f\n\n', fi_type, surfacedef, altitude, alpha);
heading2 =
sprintf('\ntime,npos,epos,alt,phi,theta,psi,vel,alpha,beta,p,q,r,nx,ny,nz,mac
h,qbar,ps,\n\n');

fprintf(fid_trim,heading1);
fprintf(fid_trim,heading2);

fid_trim = fopen(trim_file, 'a');

for row = 1 : 1 : length(y_sim(:,1))
  fprintf(fid_trim,'%8.5f,',T(row,:));
  for column = 1 : 1 : length(y_sim(1,:))
    fprintf(fid_trim,'%8.5f,',y_sim(row,column));
  end
  for column = 1:1:length(surfaces(1,:))
    fprintf(fid_trim,'%8.5f,',surfaces(row,column));
  end
  fprintf(fid_trim,'\n');
end
```

```
fclose(fid_trim);

plot_flag = input('Plot results (y/n):  ', 's');

if plot_flag == 'y'
  graphF16;
else

end
```

Appendix C

```
%================================================
%     MATLAB Script File used to Design A LQG Controller
%     for non-linear F-16/MATV model. The program will
%     also run the linearized F-16/MATV Simulation,
%     save the LTI state-space matrices to a file,
%     save and plot the simulation results.
%
% Author: Kassahun Berisha Gilede
% Jimma Institute of Technology
% Control and Instrumentation Engineering (2010)
%================================================

clear;
close all;
clc;

global fi_flag_Simulink FC_flag
global ElevatorDis AileronDis RudderDis;

newline = sprintf('\n');

disp('This is a LQG Controller Design for F-16/MATV Simulation.');
disp('The simulation will begin by asking you for the flight ');
```

```
disp('conditions for which the simulation will be performed.');
disp(newline);
%%
disp('Accpetable values for flight condition parameters are:');
disp(newline);
disp('                                  Model');
disp('  Variable                LOFI            HIFI');
disp('              Units   Min     Max     Min     Max');
disp('  Altitude:   ft       5000    40000   5000    40000');
disp('  AOA         deg     -10      45     -10      90');
disp('  Thrust      lbs      100    100000   100    100000');
disp('  Elevator    deg     -25.0    25.0   -25.0    25.0');
disp('  Aileron     deg     -21.5    21.5   -21.5    21.5');
disp('  Rudder      deg     -30      30     -30      30');
disp('  Velocity    ft/s     300     900     300     900');
disp(newline);
%%
disp('The flight condition you choose will be used to trim the F16/MATV');
disp('Note:  The trim routine will trim to the desired');
disp('altitude and velocity.  All other parameters');
disp('will be varied until level flight is achieved.  ');
disp('You may need to view the results of the simulation');
disp(' and retrim accordingly.');
disp(newline);

%% Determine from flag the correct simulation.



%% Determine from flag the correct simulation.

%% Trim aircraft to desired altitude and velocity
%%
altitude = input('Enter the altitude for the simulation (ft)   :  ');
velocity = input('Enter the velocity for the simulation (ft/s):  ');
disp(newline);
%% Initial guess for trim
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Find out which surface to create a disturbance on.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%
thrust = 5000;          % thrust, lbs
elevator = -0.09;       % elevator, degrees
alpha = 8.49;              % AOA, degrees
rudder = -0.01;             % rudder angle, degrees
aileron = 0.01;            % aileron, degrees

%% Find trim for Hifi model at desired altitude and velocity
%%
%
disp('At what flight condition would you like to trim the F-16?');
disp('1.  Steady Wings-Level Flight.');
disp('2.  Steady Turning Flight.');
disp('3.  Steady Pull-Up Flight.');
disp('4.  Steady Roll Flight.');
FC_flag = input('Your Selection:  ');
disp(newline);
%%
disp('Trimming High Fidelity Model:');
fi_flag_Simulink = 1;
[trim_state_hi, trim_thrust_hi, trim_control_hi, dLEF, xu_hi] =
trim_F16(thrust, elevator, alpha, aileron, rudder, velocity, altitude);
trim_state_lin = trim_state_hi; trim_thrust_lin = trim_thrust_hi;
trim_control_lin = trim_control_hi;

%% Find the state space model for the hifi model at the desired alt and vel.
%%
[A_hi,B_hi,C_hi,D_hi] = linmod('LIN_F16Block', [trim_state_lin;
trim_thrust_lin; trim_control_lin(1); trim_control_lin(2);
trim_control_lin(3); dLEF; -trim_state_lin(8)*180/pi], [trim_thrust_lin;
trim_control_lin(1); trim_control_lin(2); trim_control_lin(3)]);
```

```
disp(newline);
%% Find trim for lofi model at desired altitude and velocity
%%
disp('Trimming Low Fidelity Model:');
fi_flag_Simulink = 0;
[trim_state_lo, trim_thrust_lo, trim_control_lo, dLEF, xu_lo] =
trim_F16(thrust, elevator, alpha, aileron, rudder, velocity, altitude);
trim_state_lin = trim_state_lo; trim_thrust_lin = trim_thrust_lo;
trim_control_lin = trim_control_lo;

%% Find the state space model for the hifi model at the desired alt and vel.
%%
[A_lo,B_lo,C_lo,D_lo] = linmod('LIN_F16Block', [trim_state_lin;
trim_thrust_lin; trim_control_lin(1); trim_control_lin(2);
trim_control_lin(3); dLEF; -trim_state_lin(8)*180/pi], [trim_thrust_lin;
trim_control_lin(1); trim_control_lin(2); trim_control_lin(3)]);

disp(newline);
%% Save State Space and eigenvalues to file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
trim_file = sprintf('StateSpace_alt%.0f_vel%.0f.txt', altitude, velocity);
fid_trim = fopen(trim_file, 'w');
%% For Hifi
%% Print A
%%
fprintf(fid_trim,'A_hi = \n');
for i = 1:1:length(A_hi(:,1))
    for j = 1:1:length(A_hi(1,:))
        fprintf(fid_trim, '%8.5f,', A_hi(i,j));
    end
    fprintf(fid_trim, '\n');
end
fprintf(fid_trim, '\n\n');

%% Print B
%%
```

```
fprintf(fid_trim,'B_hi = \n');
for i = 1:1:length(B_hi(:,1))
    for j = 1:1:length(B_hi(1,:))
        fprintf(fid_trim, '%8.5f,', B_hi(i,j));
    end
    fprintf(fid_trim, '\n');
end
fprintf(fid_trim, '\n\n');

%% Print C
%%
fprintf(fid_trim,'C_hi = \n');
for i = 1:1:length(C_hi(:,1))
    for j = 1:1:length(C_hi(1,:))
        fprintf(fid_trim, '%8.5f,', C_hi(i,j));
    end
    fprintf(fid_trim, '\n');
end
fprintf(fid_trim, '\n\n');



%% Print D
%%
fprintf(fid_trim,'D_hi = \n');
for i = 1:1:length(D_hi(:,1))
    for j = 1:1:length(D_hi(1,:))
        fprintf(fid_trim, '%8.5f,', D_hi(i,j));
    end
    fprintf(fid_trim, '\n');
end
fprintf(fid_trim, '\n\n');

%% For Lofi
%% Print A
%%
fprintf(fid_trim,'A_lo = \n');
```

```
for i = 1:1:length(A_lo(:,1))
    for j = 1:1:length(A_lo(1,:))
        fprintf(fid_trim, '%8.5f,', A_lo(i,j));
    end
    fprintf(fid_trim, '\n');
end
fprintf(fid_trim, '\n\n');

%% Print B
%%
fprintf(fid_trim,'B_lo = \n');
for i = 1:1:length(B_lo(:,1))
    for j = 1:1:length(B_lo(1,:))
        fprintf(fid_trim, '%8.5f,', B_lo(i,j));
    end
    fprintf(fid_trim, '\n');
end
fprintf(fid_trim, '\n\n');

%% Print C
%%
fprintf(fid_trim,'C_lo = \n');
for i = 1:1:length(C_lo(:,1))
    for j = 1:1:length(C_lo(1,:))
        fprintf(fid_trim, '%8.5f,', C_lo(i,j));
    end
    fprintf(fid_trim, '\n');
end
fprintf(fid_trim, '\n\n');

%% Print D
%%
fprintf(fid_trim,'D_lo = \n');
for i = 1:1:length(D_lo(:,1))
    for j = 1:1:length(D_lo(1,:))
        fprintf(fid_trim, '%8.5f,', D_lo(i,j));
```

```
    end
    fprintf(fid_trim, '\n');
end
fprintf(fid_trim, '\n\n');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initialize some varibles used to create disturbances. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

DisEle_1 = 0;    DisEle_2 = 0;    DisEle_3 = 0;
DisAil_1 = 0;    DisAil_2 = 0;    DisAil_3 = 0;
DisRud_1 = 0;    DisRud_2 = 0;    DisRud_3 = 0;
ElevatorDis = 0; AileronDis = 0;  RudderDis = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Find out which surface to creat a disturbance.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dis_flag = input('Would you like to create a disturbance on a surface (y/n):
', 's');

if dis_flag == 'y'
    ElevatorDis = input('Enter the elevator distrubance deflection
(deg) :  ');
    DisEle_1 = ElevatorDis;    DisEle_2 = -2*ElevatorDis;    DisEle_3 =
ElevatorDis;

    AileronDis = input('Enter the aileron distrubance deflection
(deg) :  ');
    DisAil_1 = AileronDis;    DisAil_2 = -2*AileronDis;    DisAil_3 =
AileronDis;

    RudderDis = input('Enter the rudder distrubance deflection
(deg) :  ');
    DisRud_1 = RudderDis;    DisRud_2 = -2*RudderDis;    DisRud_3 =
RudderDis;
elseif dis_flag == 'n'
```

```
    %do nothing
else
    disp('Invalid Selection');
%     break;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Conditions for model
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
thrust = 0;  % Since this a linear model
deltaT = 0.001;
TStart = 0;
TFinal = 20;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%       LQG Design For High Fidelity Flight Condition
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx= 18; nu = 4; ny = 18; %State Vector , Input and Output Vector Sizes

plant_hi = ss(A_hi,B_hi,C_hi,D_hi); %Continuous time State Space
Representation of Plant
G_hi = tf(plant_hi); %Transfer Function of the F-16/MATV Dynamics

%% Optimal LQR Design
%%
P_hi = size(C_hi,1);
[n_hi,M_hi] = size(B_hi);
Qi = [57 57 57 57 1 57 1 0 0 57 0 57 0 1  1 0 0 0]; %state weights
Q_hi = diag(Qi);
R_hi = 0.01^2.*eye(M_hi);
Kr_hi = lqr(A_hi,B_hi,Q_hi,R_hi); %Linear Quadratic Regulator(LQR) or Optimal
%%Gain
 %% Design Of Linear Quadratic Estimator or Kalman Filter Gain
%%
B_hi_noise = eye(n_hi);
W_hi = eye(n_hi);
V_hi = 0.1^2.*eye(n_hi);
plantKest_hi = ss(A_hi,[B_hi, B_hi_noise],C_hi,zeros([18,22]));
```

```
[Kest_hi,Ke_hi] = kalman(plantKest_hi,W_hi,V_hi); %Linear Quadratic
Estimator(LQE) or Kalman Gain
%   Design of LQG Controller For Low Fidelity Flight Condition
%%
plant_lo = ss(A_lo,B_lo,C_lo,D_lo);

G_lo = tf(plant_lo); %Transfer Function of the F-16/MATV Dynamics

%% Optimal LQR Design
%%
P_lo = size(C_lo,1);
[n_lo,M_lo] = size(B_lo);
Q_lo = diag(Qi);
R_lo = 0.01^2.*eye(M_lo);
Kr_lo = lqr(A_lo,B_lo,Q_lo,R_lo); %Linear Quadratic Regulator(LQR) or Optimal
Gain
%% Design Of Linear Quadratic Estimator or Kalman Filter Gain
%%
B_lo_noise = eye(n_lo);
W_lo = eye(n_lo);
V_lo = 0.1^2*eye(n_lo);
Kestplant_lo = ss(A_lo,[B_lo, B_lo_noise],C_lo,zeros([18,22]));
[Kest_lo,Ke_lo] = kalman(Kestplant_lo,W_lo,V_lo); %Linear Quadratic
Estimator(LQE) or Kalman Gain
%% Run and save hifi then lofi linearized simulation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for fi_flag_Simulink = 0:1:1
    if fi_flag_Simulink == 0
        fi_model = 'hifi';
        A = A_hi; B = B_hi; C = C_hi; D = D_hi;
        Ke = Ke_hi;
        Kr = Kr_hi;
        trim_state = xu_hi;
        trim_thrust = trim_thrust_hi;
        trim_control = trim_control_hi;
    else
        fi_model = 'lofi';
```

```
        A = A_lo; B = B_lo; C = C_lo; D = D_lo;
        Ke = Ke_lo;
        Kr = Kr_lo;
        trim_state = xu_lo;
        trim_thrust = trim_thrust_lo;
        trim_control = trim_control_lo;
    end



    sim( 'LQG_F16_Block' ,[TStart TFinal]);

    trim_file = sprintf('%s%.3f%s%.3f%s%.3f_%smodel_alt%0.f_vel%.0f_LQG.txt',
'ele_', ElevatorDis, 'ail_', AileronDis, 'rud_', RudderDis, fi_model,
altitude, velocity);
    fid_trim = fopen(trim_file, 'w');

    heading =
sprintf('\ntime,npos,epos,alt,phi,theta,psi,vel,alpha,beta,p,q,r,nx,ny,nz,mac
h,qbar,ps,thrust,ele,ail,rud\n\n');

    fprintf(fid_trim,heading);

    fid_trim = fopen(trim_file, 'a');

    for row = 1:1:length(simout(:,1))
        fprintf(fid_trim,'%8.5f,',T(row,:));
        for column = 1:1:length(simout(1,:))
            fprintf(fid_trim,'%8.5f,',simout(row,column));
        end
        for column = 1:1:length(controls(1,:))
            fprintf(fid_trim,'%8.5f,',controls(row,column));
        end
        fprintf(fid_trim,'\n');
    end
```

```
    fclose(fid_trim);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plot_flag = input('Plot results (y/n):  ', 's');

if plot_flag == 'y'
   graphLQGF16_all;
else
    %break;
end
```

Appendix D

```
%================================================
%     MATLAB Script File used to Design a Discrete LQG Controller
%     for non-linear F-16/MATV model. The program will
%     also run a dsicrete linearized F-16 Simulation,
%     save the discrete state-space matrices to a file,
%     save and plot the simulation results.
%
% Author: Kassahun Berisha Gilede
% Jimma Institute of Technology
% Control and Instrumentation Engineering (2019)
%================================================
clear;
close all;
clc;

global fi_flag_Simulink FC_flag
global ElevatorDis AileronDis RudderDis;

newline = sprintf('\n');

disp('This is a Discrete LQG Controller Design for F-16/MATV Simulation.');
disp('The simulation will begin by asking you for the flight ');
disp('conditions for which the simulation will be performed.');
```

```
disp(newline);
%%
disp('Accpetable values for flight condition parameters are:');
disp(newline);
disp('                                  Model');
disp('  Variable                LOFI            HIFI');
disp('              Units   Min     Max     Min     Max');
disp('  Altitude:   ft      5000    40000   5000    40000');
disp('  AOA         deg    -10      45     -10      90');
disp('  Thrust      lbs     100    100000   100    100000');
disp('  Elevator    deg    -25.0    25.0   -25.0    25.0');
disp('  Aileron     deg    -21.5    21.5   -21.5    21.5');
disp('  Rudder      deg    -30      30     -30      30');
disp('  Velocity    ft/s    300     900     300     900');
disp(newline);
%%
%
disp('The flight condition you choose will be used to trim the F16/MATV');
disp('for both Low Fidelity F-16/MATV Trim and  High Fidelity F-16/MATV
Trim');
disp('Note:  The trim routine will trim to the desired');
disp('altitude and velocity.  All other parameters');
disp('will be varied until level flight is achieved.  ');
disp(newline);


%%
% % Trim aircraft to desired altitude and velocity
%%
altitude = input('Enter the altitude for the simulation (ft)  :  ');
velocity = input('Enter the velocity for the simulation (ft/s):  ');
disp(newline);
%% Initial guess for trim
%% Find out which surface to create a disturbance on.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%
thrust = 5000;          % thrust, lbs
```

```
elevator = -0.09;       % elevator, degrees
alpha = 8.49;              % AOA, degrees
rudder = -0.01;             % rudder angle, degrees
aileron = 0.01;            % aileron, degrees

%% Find trim for Hifi model at desired altitude and velocity
%%
%
disp('At what flight condition would you like to trim the F-16/MATV?');
disp('1.  Steady Wings-Level Flight.');
disp('2.  Steady Turning Flight.');
disp('3.  Steady Pull-Up Flight.');
disp('4.  Steady Roll Flight.');
FC_flag = input('Your Selection:  ');
disp(newline);
%%
disp('Trimming High Fidelity Discrete Model:');
fi_flag_Simulink = 1;
[trim_state_DLQGhi, trim_thrust_DLQGhi, trim_control_DLQGhi, dLEF, xu_DLQGhi]
= trim_F16(thrust, elevator, alpha, aileron, rudder, velocity, altitude);
trim_state_lin = trim_state_DLQGhi; trim_thrust_lin = trim_thrust_DLQGhi;
trim_control_lin = trim_control_DLQGhi;

%% Find the state space model for the hifi model at the desired alt and vel.
%%
[A_hi,B_hi,C_hi,D_hi] = linmod('LIN_F16Block', [trim_state_lin;
trim_thrust_lin; trim_control_lin(1); trim_control_lin(2);
trim_control_lin(3); dLEF; -trim_state_lin(8)*180/pi], [trim_thrust_lin;
trim_control_lin(1); trim_control_lin(2); trim_control_lin(3)]);
plant_DLQGhi = ss(A_hi,B_hi,C_hi,D_hi);

%%
%  The Linear High Fidelity F_16/MATV System is converted here to the
coresponding
%  Discrete System
%%
```

```
Ts = 0.4;   %%Sampling Time

DLQGplant_hi = c2d(plant_DLQGhi, Ts,'zoh');
% Retrieve the matrices
[A_DLQGhi,B_DLQGhi,C_DLQGhi,D_DLQGhi] = ssdata(DLQGplant_hi);
%-------------------------------------------------------------

%% Find trim for lofi model at desired altitude and velocity
%%
disp('Trimming Low Fidelity Discrete Model:');
fi_flag_Simulink = 0;
[trim_state_DLQGlo, trim_thrust_DLQGlo, trim_control_DLQGlo, dLEF, xu_DLQGlo]
= trim_F16(thrust, elevator, alpha, aileron, rudder, velocity, altitude);
trim_state_lin = trim_state_DLQGlo; trim_thrust_lin = trim_thrust_DLQGlo;
trim_control_lin = trim_control_DLQGlo;

%% Find the state space model for the hifi model at the desired alt and vel.
%%
[A_lo,B_lo,C_lo,D_lo] = linmod('LIN_F16Block', [trim_state_lin;
trim_thrust_lin; trim_control_lin(1); trim_control_lin(2);
trim_control_lin(3); dLEF; -trim_state_lin(8)*180/pi], [trim_thrust_lin;
trim_control_lin(1); trim_control_lin(2); trim_control_lin(3)]);
plant_DLQGlo = ss(A_lo,B_lo,C_lo,D_lo);

%  Discrete System
%%
% Ts = 0.4;   %%Sampling Time

DLQGplant_lo = c2d(plant_DLQGlo, Ts,'zoh');
% Retrieve the matrices
[A_DLQGlo,B_DLQGlo,C_DLQGlo,D_DLQGlo] = ssdata(DLQGplant_lo);
%% Save State Space and eigenvalues to file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
trim_file = sprintf('Discrete_State_Space_alt%.0f_vel%.0f_DLQG.txt',
altitude, velocity);
fid_trim = fopen(trim_file, 'w');
```

```
%% For Hifi
%% Print A
%%
fprintf(fid_trim,'A_DLQGhi = \n');
for i = 1:1:length(A_DLQGhi(:,1))
    for j = 1:1:length(A_DLQGhi(1,:))
        fprintf(fid_trim, '%8.5f,', A_DLQGhi(i,j));
    end
    fprintf(fid_trim, '\n');
end
fprintf(fid_trim, '\n\n');

%% Print B
%%
fprintf(fid_trim,'B_DLQGhi = \n');
for i = 1:1:length(B_DLQGhi(:,1))
    for j = 1:1:length(B_DLQGhi(1,:))
        fprintf(fid_trim, '%8.5f,', B_DLQGhi(i,j));
    end
    fprintf(fid_trim, '\n');
end
fprintf(fid_trim, '\n\n');

%% Print C
%%
fprintf(fid_trim,'C_DLQGhi = \n');
for i = 1:1:length(C_DLQGhi(:,1))
    for j = 1:1:length(C_DLQGhi(1,:))
        fprintf(fid_trim, '%8.5f,', C_DLQGhi(i,j));
    end
    fprintf(fid_trim, '\n');
end
fprintf(fid_trim, '\n\n');

%% Print D
%%
```

```
fprintf(fid_trim,'D_DLQGhi = \n');
for i = 1:1:length(D_DLQGhi(:,1))
    for j = 1:1:length(D_DLQGhi(1,:))
        fprintf(fid_trim, '%8.5f,', D_DLQGhi(i,j));
    end
    fprintf(fid_trim, '\n');
end
fprintf(fid_trim, '\n\n');

%% For Lofi
%% Print A
%%
fprintf(fid_trim,'A_DLQGlo = \n');
for i = 1:1:length(A_DLQGlo(:,1))
    for j = 1:1:length(A_DLQGlo(1,:))
        fprintf(fid_trim, '%8.5f,', A_DLQGlo(i,j));
    end
    fprintf(fid_trim, '\n');
end
fprintf(fid_trim, '\n\n');
%% Print B
%%
fprintf(fid_trim,'B_DLQGlo = \n');
for i = 1:1:length(B_DLQGlo(:,1))
    for j = 1:1:length(B_DLQGlo(1,:))
        fprintf(fid_trim, '%8.5f,', B_DLQGlo(i,j));
    end
    fprintf(fid_trim, '\n');
end
fprintf(fid_trim, '\n\n');

%% Print C
%%
fprintf(fid_trim,'C_DLQGlo = \n');
for i = 1:1:length(C_DLQGlo(:,1))
    for j = 1:1:length(C_DLQGlo(1,:))
        fprintf(fid_trim, '%8.5f,', C_DLQGlo(i,j));
```

```
    end
    fprintf(fid_trim, '\n');
end
fprintf(fid_trim, '\n\n');

%% Print D
%%
fprintf(fid_trim,'D_DLQGlo = \n');
for i = 1:1:length(D_DLQGlo(:,1))
    for j = 1:1:length(D_DLQGlo(1,:))
        fprintf(fid_trim, '%8.5f,', D_DLQGlo(i,j));
    end
    fprintf(fid_trim, '\n');
end
fprintf(fid_trim, '\n\n');
%% Initialize some varibles used to create disturbances. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

DisEle_1 = 0;    DisEle_2 = 0;    DisEle_3 = 0;
DisAil_1 = 0;    DisAil_2 = 0;    DisAil_3 = 0;
DisRud_1 = 0;    DisRud_2 = 0;    DisRud_3 = 0;
ElevatorDis = 0; AileronDis = 0;  RudderDis = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Find out which surface to creat a disturbance.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dis_flag = input('Would you like to create a disturbance on a surface (y/n):
', 's');

if dis_flag == 'y'
    ElevatorDis = input('Enter the elevator distrubance deflection
(deg) :  ');
    DisEle_1 = ElevatorDis;    DisEle_2 = -2*ElevatorDis;    DisEle_3 =
ElevatorDis;
```

```
    AileronDis = input('Enter the aileron distrubance deflection
(deg) :  ');
    DisAil_1 = AileronDis;    DisAil_2 = -2*AileronDis;    DisAil_3 =
AileronDis;

    RudderDis = input('Enter the rudder distrubance deflection
(deg) :  ');
    DisRud_1 = RudderDis;    DisRud_2 = -2*RudderDis;    DisRud_3 =
RudderDis;
elseif dis_flag == 'n'
    %do nothing
else
    disp('Invalid Selection');
%     break;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Conditions for model
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
thrust = 0;  % Since this a linear model

deltaT = 0.001;
TStart = 0;
TFinal = 200;
%%       LQG Design For High Fidelity Flight Condition
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx= 18; nu = 4; ny = 18; %State Vector , Input and Output Vector Sizes

%DLQGplant_hi = ss(A_hi,B_hi,C_hi,D_hi); %Continuous time State Space
Representation of Plant
DLQGTR_hi = tf(DLQGplant_hi); %Transfer Function of the F-16/MATV Dynamics

%%
% Discrete Optimal LQR Controller Design
 [n_DLQGhi,M_DLQGhi] = size(B_DLQGhi);
DLQGQi = [49.2869 49.2869 46.1114 46.1114 25.1462 6.3934 15.8585 15.8585
4.4937 1.4638 1.2255 0.1980 0.1829 0.0003 0.0004 0 0 0]; %state weights
```

```
Q_DLQGhi = diag(DLQGQi);
R_DLQGhi = eye(M_DLQGhi);
Kr_DLQGhi = dlqr(A_DLQGhi,B_DLQGhi,Q_DLQGhi,R_DLQGhi); %Linear Quadratic
Regulator(LQR) or Optimal Gain
 %%
%% Design Of Linear Quadratic Estimator or Kalman Filter Gain
%%
B_DLQGhi_noise = eye(n_DLQGhi);
W_DLQGhi = eye(n_DLQGhi);
V_DLQGhi = 0.01^2*eye(n_DLQGhi);
% D_hi = B_hi.*0;
Plant_DLQGhi = ss(A_DLQGhi,[B_DLQGhi,
B_DLQGhi_noise],C_DLQGhi,zeros([18,22]),Ts);
[KEST_DLQGhi,L_DLQGhi,P_DLQGhi] = kalman(Plant_DLQGhi,W_DLQGhi,V_DLQGhi);
%Linear Quadratic Estimator(LQE) or Kalman Gain
%   Design of Discrete LQG Controller For Low Fidelity Flight Condition
%%

DLQGTf_lo = tf(DLQGplant_lo); %Transfer Function of the F-16/MATV Dynamics

%% Discrete Optimal LQR Controller Design
%%
% P_DLQGlo = size(C_DLQGlo,1);
[n_DLQGlo,M_DLQGlo] = size(B_DLQGlo);
%state weights
Q_DLQGlo = diag(DLQGQi);
R_DLQGlo = eye(M_DLQGlo);
Kr_DLQGlo = dlqr(A_DLQGlo,B_DLQGlo,Q_DLQGlo,R_DLQGlo); %Linear Quadratic
Regulator(LQR) or Optimal Gain
%%
%% Design Of Linear Quadratic Estimator or Kalman Filter Gain
%%
B_DLQGlo_noise = eye(n_DLQGlo);
W_DLQGlo = eye(n_DLQGlo);
V_DLQGlo = 0.01^2*eye(n_DLQGlo);
plant_DLQGlo = ss(A_DLQGlo,[B_DLQGlo,
B_DLQGlo_noise],C_lo,zeros([18,22]),Ts);
```

```
[KEST_DLQGlo,L_DLQGlo] = kalman(plant_DLQGlo,W_DLQGlo,V_DLQGlo); %Linear
Quadratic Estimator(LQE) or Kalman Gain
%% Run and save hifi then lofi linearized simulation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for fi_flag_Simulink = 0:1:1
    if fi_flag_Simulink == 0
        fi_model = 'hifi';
        A = A_DLQGhi; B = B_DLQGhi; C = C_DLQGhi; D = D_DLQGhi;
        L = L_DLQGhi; Kd = Kr_DLQGhi;
        trim_state = xu_DLQGhi;
        trim_thrust = trim_thrust_DLQGhi;
        trim_control = trim_control_DLQGhi;
    else
        fi_model = 'lofi';
        A = A_DLQGlo; B = B_DLQGlo; C = C_DLQGlo; D = D_DLQGlo;
        L = L_DLQGlo; Kd = Kr_DLQGlo;
        trim_state = xu_DLQGlo;
        trim_thrust = trim_thrust_DLQGlo;
        trim_control = trim_control_DLQGlo;
    end



    sim( 'DLQGSS_F16_Block' ,[TStart TFinal]);


    trim_file =
sprintf('%s%.3f%s%.3f%s%.3f_%smodel_alt%0.f_vel%.0f_DLQG.txt', 'ele_',
ElevatorDis, 'ail_', AileronDis, 'rud_', RudderDis, fi_model, altitude,
velocity);
    fid_trim = fopen(trim_file, 'w');


    heading =
sprintf('\ntime,npos,epos,alt,phi,theta,psi,vel,alpha,beta,p,q,r,nx,ny,nz,mac
h,qbar,ps,thrust,ele,ail,rud\n\n');


    fprintf(fid_trim,heading);
```

```
    fid_trim = fopen(trim_file, 'a');

    for row = 1:1:length(simout(:,1))
        fprintf(fid_trim,'%8.5f,',T(row,:));
        for column = 1:1:length(simout(1,:))
            fprintf(fid_trim,'%8.5f,',simout(row,column));
        end
        for column = 1:1:length(controls(1,:))
            fprintf(fid_trim,'%8.5f,',controls(row,column));
        end
        fprintf(fid_trim,'\n');
    end

    fclose(fid_trim);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plot_flag = input('Plot results (y/n):  ', 's');

if plot_flag == 'y'
   graphDLQGF16_all;
else
    %break;
end
```

Appendix E

```
function graphLQGF16_all()

%% Function to Graph the results of Contuinuous System
%%

global altitude velocity ;
global ElevatorDis AileronDis RudderDis;
```

```
surface1 = 'ele_';
surface2 = 'ail_';
surface3 = 'rud_';

lofi_data_file = sprintf('%s%.3f%s%.3f%s%.3f_%smodel_alt%0.f_vel%.0f.txt',
surface1, ElevatorDis, surface2, AileronDis, surface3, RudderDis, 'lofi',
altitude, velocity);
hifi_data_file = sprintf('%s%.3f%s%.3f%s%.3f_%smodel_alt%0.f_vel%.0f.txt',
surface1, ElevatorDis, surface2, AileronDis, surface3, RudderDis, 'hifi',
altitude, velocity);
lofi_LQG_file = sprintf('%s%.3f%s%.3f%s%.3f_%smodel_alt%0.f_vel%.0f_LQG.txt',
surface1, ElevatorDis, surface2, AileronDis, surface3, RudderDis, 'lofi',
altitude, velocity);
hifi_LQG_file = sprintf('%s%.3f%s%.3f%s%.3f_%smodel_alt%0.f_vel%.0f_LQG.txt',
surface1, ElevatorDis, surface2, AileronDis, surface3, RudderDis, 'hifi',
altitude, velocity);

lofiID = fopen(lofi_data_file);
hifiID = fopen(hifi_data_file);
lofiLQGID = fopen(lofi_LQG_file);
hifiLQGID = fopen(hifi_LQG_file);

if (lofiID > 0)
    [time_lo, npos_lo, epos_lo, alt_lo, phi_lo, theta_lo, psi_lo, vel_lo,
alpha_lo, sideslip_lo, roll_lo, pitch_lo, yaw_lo, nx_lo, ny_lo, nz_lo,
mach_lo, qbar_lo, ps_lo, thrust_lo, ele_lo, ail_lo, rud_lo] =
textread(lofi_data_file,'%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f
%f %f %f %f %f %f', 'delimiter', ',' , 'headerlines',3);
else
    time_lo = 0; npos_lo = 0; epos_lo = 0; alt_lo = 0; phi_lo= 0; theta_lo=
0; psi_lo= 0; vel_lo= 0; alpha_lo= 0; sideslip_lo= 0; roll_lo= 0; pitch_lo=
0; yaw_lo= 0; nx_lo= 0; ny_lo= 0; nz_lo= 0; mach_lo= 0; qbar_lo= 0; ps_lo= 0;
thrust_lo= 0; ele_lo= 0; ail_lo= 0; rud_lo= 0;
end

if (hifiID > 0)
```

```
    [time_hi, npos_hi, epos_hi, alt_hi, phi_hi, theta_hi, psi_hi, vel_hi,
alpha_hi, sideslip_hi, roll_hi, pitch_hi, yaw_hi, nx_hi, ny_hi, nz_hi,
mach_hi, qbar_hi, ps_hi, thrust_hi, ele_hi, ail_hi, rud_hi] =
textread(hifi_data_file,'%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f
%f %f %f %f %f %f', 'delimiter', ',' , 'headerlines',3);
else
    time_hi = 0; npos_hi = 0; epos_hi = 0; alt_hi = 0; phi_hi= 0; theta_hi=
0; psi_hi= 0; vel_hi= 0; alpha_hi= 0; sideslip_hi= 0; roll_hi= 0; pitch_hi=
0; yaw_hi= 0; nx_hi= 0; ny_hi= 0; nz_hi= 0; mach_hi= 0; qbar_hi= 0; ps_hi= 0;
thrust_hi= 0; ele_hi= 0; ail_hi= 0; rud_hi= 0;
end
%%%

if (lofiLQGID > 0)
    [time_lo_LQG, npos_lo_LQG, epos_lo_LQG, alt_lo_LQG, phi_lo_LQG,
theta_lo_LQG, psi_lo_LQG, vel_lo_LQG, alpha_lo_LQG, sideslip_lo_LQG,
roll_lo_LQG, pitch_lo_LQG, yaw_lo_LQG, nx_lo_LQG, ny_lo_LQG, nz_lo_LQG,
mach_lo_LQG, qbar_lo_LQG, ps_lo_LQG, thrust_lo_LQG, ele_lo_LQG, ail_lo_LQG,
rud_lo_LQG] = textread(lofi_LQG_file,'%f %f %f %f %f %f %f %f %f %f %f %f %f
%f %f %f %f %f %f %f %f %f %f', 'delimiter', ',' , 'headerlines',3);
else
    time_lo_LQG = 0; npos_lo_LQG = 0; epos_lo_LQG = 0; alt_lo_LQG = 0;
phi_lo_LQG= 0; theta_lo_LQG= 0; psi_lo_LQG= 0; vel_lo_LQG= 0; alpha_lo_LQG=
0; sideslip_lo_LQG= 0; roll_lo_LQG= 0; pitch_lo_LQG= 0; yaw_lo_LQG= 0;
nx_lo_LQG= 0; ny_lo_LQG= 0; nz_lo_LQG= 0; mach_lo_LQG= 0; qbar_lo_LQG= 0;
ps_lo_LQG= 0; thrust_lo_LQG= 0; ele_lo_LQG= 0; ail_lo_LQG= 0; rud_lo_LQG= 0;
end

if (hifiLQGID > 0)
    [time_hi_LQG, npos_hi_LQG, epos_hi_LQG, alt_hi_LQG, phi_hi_LQG,
theta_hi_LQG, psi_hi_LQG, vel_hi_LQG, alpha_hi_LQG, sideslip_hi_LQG,
roll_hi_LQG, pitch_hi_LQG, yaw_hi_LQG, nx_hi_LQG, ny_hi_LQG, nz_hi_LQG,
mach_hi_LQG, qbar_hi_LQG, ps_hi_LQG, thrust_hi_LQG, ele_hi_LQG, ail_hi_LQG,
rud_hi_LQG] = textread(hifi_LQG_file,'%f %f %f %f %f %f %f %f %f %f %f %f %f
%f %f %f %f %f %f %f %f %f %f', 'delimiter', ',' , 'headerlines',3);
else
```

```
    time_hi_LQG = 0; npos_hi_LQG = 0; epos_hi_LQG = 0; alt_hi_LQG = 0;
phi_hi_LQG= 0; theta_hi_LQG= 0; psi_hi_LQG= 0; vel_hi_LQG= 0; alpha_hi_LQG=
0; sideslip_hi_LQG= 0; roll_hi_LQG= 0; pitch_hi_LQG= 0; yaw_hi_LQG= 0;
nx_hi_LQG= 0; ny_hi_LQG= 0; nz_hi_LQG= 0; mach_hi_LQG= 0; qbar_hi_LQG= 0;
ps_hi_LQG= 0; thrust_hi_LQG= 0; ele_hi_LQG= 0; ail_hi_LQG= 0; rud_hi_LQG= 0;
end

title_string = sprintf('LQG-Con-Res-Trim: Vel. = %.1f \n Alt. = %.1f',
velocity, altitude);

figure(1);
set(gca,'FontSize',12);
get(gca,'default');
title(title_string);
subplot(231)
plot(time_lo,vel_lo,'-c*','LineWidth',1.5);
hold on;
plot(time_hi, vel_hi, '-g','LineWidth',1.5);
plot(time_lo_LQG,vel_lo_LQG, '-r','LineWidth',1.5); hold on;
plot(time_hi_LQG, vel_hi_LQG,'-b','LineWidth',1.5);
ylabel('Velocity (ft/s)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
legend('LOFI-LQG', 'HIFI-LQG');
title(title_string);

subplot(232)
plot(time_lo, alpha_lo,'-c*','LineWidth',1.5);
hold on;
plot(time_hi, alpha_hi, '-g','LineWidth',1.5);
plot(time_lo_LQG, alpha_lo_LQG, '-r','LineWidth',1.5); hold on;
plot(time_hi_LQG, alpha_hi_LQG,'-b','LineWidth',1.5);
ylabel('Angle of Attack (degrees)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);
```

```
subplot(233)
plot(time_lo , sideslip_lo,'-c*','LineWidth',1.5);
hold on;
plot(time_hi, sideslip_hi, '-g','LineWidth',1.5);
plot(time_lo_LQG , sideslip_lo_LQG, '-r','LineWidth',1.5); hold on;
plot(time_hi_LQG, sideslip_hi_LQG, '-b','LineWidth',1.5);
ylabel('Side Slip (degrees)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(234)
plot(time_lo,roll_lo,'-c*','LineWidth',1.5);
hold on;
plot(time_hi, roll_hi, '-g','LineWidth',1.5);
plot(time_lo_LQG,roll_lo_LQG,   '-r','LineWidth',1.5); hold on;
plot(time_hi_LQG, roll_hi_LQG, '-b','LineWidth',1.5);
ylabel('Roll Rate (deg/s)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(235)
plot(time_lo, pitch_lo, '-c*','LineWidth',1.5);
hold on;
plot(time_hi, pitch_hi, '-g','LineWidth',1.5);
plot(time_lo_LQG, pitch_lo_LQG ,  '-r','LineWidth',1.5); hold on;
plot(time_hi_LQG, pitch_hi_LQG,'-b','LineWidth',1.5);
ylabel('Pitch Rate (deg/s)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(236)
plot(time_lo, yaw_lo,'-c*','LineWidth',1.5);
hold on;
plot(time_hi, yaw_hi, '-g','LineWidth',1.5);
plot(time_lo_LQG, yaw_lo_LQG, '-r','LineWidth',1.5); hold on;
plot(time_hi_LQG, yaw_hi_LQG,'-b','LineWidth',1.5);
```

```
ylabel('Yaw Rate (deg/s)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

%% Figure 2
%%
figure(2);

subplot(231)
plot(time_lo, npos_lo,'-c*','LineWidth',1.5);
hold on;
plot(time_hi, npos_hi, '-g','LineWidth',1.5);
plot(time_lo_LQG, npos_lo_LQG, '-r','LineWidth',1.5); hold on;
plot(time_hi_LQG, npos_hi_LQG, '-b','LineWidth',1.5);
ylabel('North Pos.','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
legend('LOFI-LQG', 'HIFI-LQG');
title(title_string);

subplot(232)
plot(time_lo, epos_lo,'-c*','LineWidth',1.5);
hold on;
plot(time_hi, epos_hi, '-g','LineWidth',1.5);
plot(time_lo_LQG, epos_lo_LQG, '-r','LineWidth',1.5); hold on;
plot(time_hi_LQG, epos_hi_LQG,  '-b','LineWidth',1.5);
ylabel('East Pos.','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(233)
plot(time_lo, alt_lo, '-c*','LineWidth',1.5);
hold on;
plot(time_hi , alt_hi, '-g','LineWidth',1.5);
plot(time_lo_LQG, alt_lo_LQG ,'-r','LineWidth',1.5);
plot(time_hi_LQG , alt_hi_LQG, '-b','LineWidth',1.5) ;
ylabel('Altitude ft','FontWeight','bold','FontSize',12);
```

```
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(234)
plot(time_lo, phi_lo,'-c*','LineWidth',1.5);
hold on;
plot(time_hi, phi_hi, '-g','LineWidth',1.5);
plot(time_lo_LQG, phi_lo_LQG, '-r','LineWidth',1.5); hold on;
plot(time_hi_LQG, phi_hi_LQG, '-b','LineWidth',1.5);
ylabel('PHI (degrees)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(235)
plot(time_lo, theta_lo, '-c*','LineWidth',1.5);
hold on;
plot(time_hi, theta_hi,'-g','LineWidth',1.5);
plot(time_lo_LQG, theta_lo_LQG,'-r','LineWidth',1.5); hold on;
plot(time_hi_LQG, theta_hi_LQG,'-b','LineWidth',1.5);
ylabel('THETA (degrees)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(236)
plot(time_lo ,psi_lo,'-c*','LineWidth',1.5);
hold on;
plot(time_hi, psi_hi, '-g','LineWidth',1.5);
plot(time_lo_LQG ,psi_lo_LQG, '-r','LineWidth',1.5); hold on;
plot(time_hi_LQG, psi_hi_LQG,'-b','LineWidth',1.5);
ylabel('PSI (degrees)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

%% Figure 3
%%
figure(3);
```

```
title(title_string);
subplot(231)
plot(time_lo, nx_lo, '-c*','LineWidth',1.5);
hold on;
plot(time_hi, nx_hi, '-g','LineWidth',1.5);
plot(time_lo_LQG, nx_lo_LQG, '-r','LineWidth',1.5); hold on;
plot(time_hi_LQG, nx_hi_LQG,'-b','LineWidth',1.5);
ylabel('acc x','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
legend('LOFI-LQG', 'HIFI-LQG');
title(title_string);

subplot(232)
plot(time_lo, ny_lo,'-c*','LineWidth',1.5);
hold on;
plot(time_hi, ny_hi, '-g','LineWidth',1.5);
plot(time_lo_LQG, ny_lo_LQG,  '-r','LineWidth',1.5); hold on;
plot(time_hi_LQG, ny_hi_LQG,'-b','LineWidth',1.5);
ylabel('acc y','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(233)
plot(time_lo , nz_lo,'-c*','LineWidth',1.5);
hold on;
plot(time_hi, nz_hi, '-g','LineWidth',1.5);
plot(time_lo_LQG , nz_lo_LQG, '-r','LineWidth',1.5); hold on;
plot(time_hi_LQG, nz_hi_LQG, '-b','LineWidth',1.5);
ylabel('acc z','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(234)
plot(time_lo, mach_lo,'-c*','LineWidth',1.5);
hold on;
plot(time_hi, mach_hi,'-g','LineWidth',1.5);
```

```
plot(time_lo_LQG, mach_lo_LQG, '-r','LineWidth',1.5); hold on;
plot( time_hi_LQG, mach_hi_LQG,'-b','LineWidth',1.5);
ylabel('Mach','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(235)
plot(time_lo, qbar_lo,'-c*','LineWidth',1.5);
hold on;
plot(time_hi, qbar_hi, '-g','LineWidth',1.5);
plot( time_lo_LQG, qbar_lo_LQG,  '-r','LineWidth',1.5); hold on;
plot(time_hi_LQG, qbar_hi_LQG, '-b','LineWidth',1.5);
ylabel('q bar)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(236)
plot(time_lo, ps_lo, '-c*','LineWidth',1.5);
hold on;
plot( time_hi, ps_hi, '-g', 'LineWidth',2);
plot(time_lo_LQG, ps_lo_LQG, '-r', 'LineWidth',2); hold on;
plot(time_hi_LQG, ps_hi_LQG, '-b','LineWidth',1.5);
ylabel('ps','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);
```

Appendix F

```
function graphLDSF16_all()

%% Function to Graph the results of Discrete System%

global altitude velocity ;
global ElevatorDis AileronDis RudderDis;

surface1 = 'ele_';
surface2 = 'ail_';
```

```
surface3 = 'rud_';

lofi_LDS_file = sprintf('%s%.3f%s%.3f%s%.3f_%smodel_alt%0.f_vel%.0f_LDS.txt',
surface1, ElevatorDis, surface2, AileronDis, surface3, RudderDis, 'lofi',
altitude, velocity);
hifi_LDS_file = sprintf('%s%.3f%s%.3f%s%.3f_%smodel_alt%0.f_vel%.0f_LDS.txt',
surface1, ElevatorDis, surface2, AileronDis, surface3, RudderDis, 'hifi',
altitude, velocity);

lofiLDSID = fopen(lofi_LDS_file);
hifiLDSID = fopen(hifi_LDS_file);

if (lofiLDSID > 0)
    [time_lo_LDS, npos_lo_LDS, epos_lo_LDS, alt_lo_LDS, phi_lo_LDS,
theta_lo_LDS, psi_lo_LDS, vel_lo_LDS, alpha_lo_LDS, sideslip_lo_LDS,
roll_lo_LDS, pitch_lo_LDS, yaw_lo_LDS, nx_lo_LDS, ny_lo_LDS, nz_lo_LDS,
mach_lo_LDS, qbar_lo_LDS, ps_lo_LDS, thrust_lo_LDS, ele_lo_LDS, ail_lo_LDS,
rud_lo_LDS] = textread(lofi_LDS_file,'%f %f %f %f %f %f %f %f %f %f %f %f
%f %f %f %f %f %f %f %f %f %f', 'delimiter', ',' , 'headerlines',3);
else
    time_lo_LDS = 0; npos_lo_LDS = 0; epos_lo_LDS = 0; alt_lo_LDS = 0;
phi_lo_LDS= 0; theta_lo_LDS= 0; psi_lo_LDS= 0; vel_lo_LDS= 0; alpha_lo_LDS=
0; sideslip_lo_LDS= 0; roll_lo_LDS= 0; pitch_lo_LDS= 0; yaw_lo_LDS= 0;
nx_lo_LDS= 0; ny_lo_LDS= 0; nz_lo_LDS= 0; mach_lo_LDS= 0; qbar_lo_LDS= 0;
ps_lo_LDS= 0; thrust_lo_LDS= 0; ele_lo_LDS= 0; ail_lo_LDS= 0; rud_lo_LDS= 0;
end

if (hifiLDSID > 0)
    [time_hi_LDS, npos_hi_LDS, epos_hi_LDS, alt_hi_LDS, phi_hi_LDS,
theta_hi_LDS, psi_hi_LDS, vel_hi_LDS, alpha_hi_LDS, sideslip_hi_LDS,
roll_hi_LDS, pitch_hi_LDS, yaw_hi_LDS, nx_hi_LDS, ny_hi_LDS, nz_hi_LDS,
mach_hi_LDS, qbar_hi_LDS, ps_hi_LDS, thrust_hi_LDS, ele_hi_LDS, ail_hi_LDS,
rud_hi_LDS] = textread(hifi_LDS_file,'%f %f %f %f %f %f %f %f %f %f %f %f
%f %f %f %f %f %f %f %f %f %f', 'delimiter', ',' , 'headerlines',3);
else
```

```
    time_hi_LDS = 0; npos_hi_LDS = 0; epos_hi_LDS = 0; alt_hi_LDS = 0;
phi_hi_LDS= 0; theta_hi_LDS= 0; psi_hi_LDS= 0; vel_hi_LDS= 0; alpha_hi_LDS=
0; sideslip_hi_LDS= 0; roll_hi_LDS= 0; pitch_hi_LDS= 0; yaw_hi_LDS= 0;
nx_hi_LDS= 0; ny_hi_LDS= 0; nz_hi_LDS= 0; mach_hi_LDS= 0; qbar_hi_LDS= 0;
ps_hi_LDS= 0; thrust_hi_LDS= 0; ele_hi_LDS= 0; ail_hi_LDS= 0; rud_hi_LDS= 0;
end

title_string = sprintf('Discrete F-16/MATV Trim: Vel. = %.1f \n Alt. = %.1f',
velocity, altitude);

figure(1);
set(gca,'FontWeight','bold','FontSize',12);
get(gca,'default');
title(title_string);
subplot(231)
plot(time_lo_LDS,vel_lo_LDS, '-r','LineWidth',1.5); hold on;
plot(time_hi_LDS, vel_hi_LDS,'-b','LineWidth',1.5);
ylabel('Velocity (ft/s)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
legend('LOFI-LDS', 'HIFI-LDS');
title(title_string);

subplot(232)
plot(time_lo_LDS, alpha_lo_LDS, '-r','LineWidth',1.5); hold on;
plot(time_hi_LDS, alpha_hi_LDS,'-b','LineWidth',1.5);
ylabel('Angle of Attack (degrees)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(233)
plot(time_lo_LDS , sideslip_lo_LDS, '-r','LineWidth',1.5); hold on;
plot(time_hi_LDS, sideslip_hi_LDS, '-b','LineWidth',1.5);
ylabel('Side Slip (degrees)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);
```

```
subplot(234)
plot(time_lo_LDS,roll_lo_LDS,   '-r','LineWidth',1.5); hold on;
plot(time_hi_LDS, roll_hi_LDS, '-b','LineWidth',1.5);
ylabel('Roll Rate (deg/s)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(235)
plot(time_lo_LDS, pitch_lo_LDS ,  '-r','LineWidth',1.5); hold on;
plot(time_hi_LDS, pitch_hi_LDS,'-b','LineWidth',1.5);
ylabel('Pitch Rate (deg/s)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(236)
plot(time_lo_LDS, yaw_lo_LDS, '-r','LineWidth',1.5); hold on;
plot(time_hi_LDS, yaw_hi_LDS,'-b','LineWidth',1.5);
ylabel('Yaw Rate (deg/s)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

%% Figure 2
%%
figure(2);

subplot(231)
plot(time_lo_LDS, npos_lo_LDS, '-r','LineWidth',1.5); hold on;
plot(time_hi_LDS, npos_hi_LDS, '-b','LineWidth',1.5);
ylabel('North Pos.','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
legend('LOFI-LDS', 'HIFI-LDS');
title(title_string);

subplot(232)
plot(time_lo_LDS, epos_lo_LDS, '-r','LineWidth',1.5); hold on;
plot(time_hi_LDS, epos_hi_LDS,  '-b','LineWidth',1.5);
```

```
ylabel('East Pos.','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(233)
plot(time_lo_LDS, alt_lo_LDS ,'-r','LineWidth',1.5); hold on;
plot(time_hi_LDS , alt_hi_LDS, '-b','LineWidth',1.5) ;
ylabel('Altitude ft','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(234)
plot(time_lo_LDS, phi_lo_LDS, '-r','LineWidth',1.5); hold on;
plot(time_hi_LDS, phi_hi_LDS, '-b','LineWidth',1.5);
ylabel('PHI (degrees)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(235)
plot(time_lo_LDS, theta_lo_LDS,'-r','LineWidth',1.5); hold on;
plot(time_hi_LDS, theta_hi_LDS,'-b','LineWidth',1.5);
ylabel('THETA (degrees)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(236)
plot(time_lo_LDS ,psi_lo_LDS, '-r','LineWidth',1.5); hold on;
plot(time_hi_LDS, psi_hi_LDS,'-b','LineWidth',1.5);
ylabel('PSI (degrees)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

%% Figure 3
%%
figure(3);
title(title_string);
```

```
subplot(231)
plot(time_lo_LDS, nx_lo_LDS, '-r','LineWidth',1.5); hold on;
plot(time_hi_LDS, nx_hi_LDS,'-b','LineWidth',1.5);
ylabel('acc x','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
legend('LOFI-LDS', 'HIFI-LDS');
title(title_string);

subplot(232)
plot(time_lo_LDS, ny_lo_LDS,  '-r','LineWidth',1.5); hold on;
plot(time_hi_LDS, ny_hi_LDS,'-b','LineWidth',1.5);
ylabel('acc y','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(233)
plot(time_lo_LDS , nz_lo_LDS, '-r','LineWidth',1.5); hold on;
plot(time_hi_LDS, nz_hi_LDS, '-b','LineWidth',1.5);
ylabel('acc z','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(234)
plot(time_lo_LDS, mach_lo_LDS, '-r','LineWidth',1.5); hold on;
plot( time_hi_LDS, mach_hi_LDS,'-b','LineWidth',1.5);
ylabel('Mach','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);

subplot(235)
plot( time_lo_LDS, qbar_lo_LDS,  '-r','LineWidth',1.5); hold on;
plot(time_hi_LDS, qbar_hi_LDS, '-b','LineWidth',1.5);
ylabel('q bar)','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);
```

```
subplot(236)
plot(time_lo_LDS, ps_lo_LDS, '-r', 'LineWidth',1.5); hold on;
plot(time_hi_LDS, ps_hi_LDS, '-b','LineWidth',1.5);
ylabel('ps','FontWeight','bold','FontSize',12);
xlabel('Time (sec)','FontWeight','bold','FontSize',12);
title(title_string);
```

Appendix G

```
%=====================================================
%      F16/MATV nonlinear model trim cost function
%   for longitudinal motion, steady level flight
% (cost = sum of weighted squared state derivatives)
%
% Author: G. Kassahun B.
% Date:   January 19, 2019
%
%      Added addtional functionality.
%      This trim function can now trim at three
%      additional flight conditions
%         -  Steady Turning Flight given turn rate
%         -  Steady Pull-up flight - given pull-up rate
%         -  Steady Roll - given roll rate
%
%%
%=====================================================
%%*********************************************%%
%   Altered to work as a trimming function       %
%   for the  HIFI F_16/MATV Model                   %
%%*********************************************%%

function [cost, Xdot, xu] = trimfun(UX0)

global phi psi p q r phi_weight theta_weight psi_weight dLEF
global altitude velocity fi_flag_Simulink
```

```
% Implementing limits:
% Thrust limits
if UX0(1) > 100000
    UX0(1) = 100000;
elseif UX0(1) < 100
    UX0(1) = 100;
end;

% elevator limits
if UX0(2) > 25
    UX0(2) = 25;
elseif UX0(2) < -25
    UX0(2) = -25;
end;

% angle of attack limits
if (fi_flag_Simulink == 0)
  if UX0(3) > 45*pi/180
    UX0(3) = 45*pi/180;
  elseif UX0(3) < -10*pi/180
    UX0(3) = -10*pi/180;
  end
elseif (fi_flag_Simulink == 1)
  if UX0(3) > 90*pi/180
    UX0(3) = 90*pi/180;
  elseif UX0(3) < -20*pi/180
    UX0(3) = -20*pi/180;
  end
end

%  Aileron limits
if UX0(4) > 21.5
    UX0(4) = 21.5;
elseif UX0(4) < -21.5
    UX0(4) = -21.5;
end;
```

```
% Rudder limits
if UX0(5) > 30
    UX0(5) = 30;
elseif UX0(5) < -30
    UX0(5) = -30;
end;

if (fi_flag_Simulink == 1)
    % Calculating qbar, ps and steady state leading edge flap deflection:
    % (see pg. 43 NASA report)
    rho0 = 2.377e-3; tfac = 1 - 0.703e-5*altitude;
    temp = 519*tfac; if (altitude >= 35000), temp = 390; end;
    rho = rho0*tfac^4.14;
    qbar = 0.5*rho*velocity^2;
    ps = 1715*rho*temp;

    dLEF = 1.38*UX0(3)*180/pi - 9.05*qbar/ps + 1.45;

elseif (fi_flag_Simulink == 0)
    dLEF = 0.0;
end

% Verify that the calculated leading edge flap
% have not been violated.
if (dLEF > 25)
    dLEF = 25;
elseif (dLEF < 0)
    dLEF = 0;
end;

xu = [  0               ... %npos (ft)
        0               ... %epos (ft)
        altitude        ... %altitude (ft)
        phi*(pi/180)    ... %phi (rad)
        UX0(3)          ... %theta (rad)
```

```
        psi*(pi/180)   ... %psi (rad)
        velocity       ... %velocity (ft/s)
        UX0(3)         ... %alpha (rad)
        0              ... %beta (rad)
        p*(pi/180)     ... %p (rad/s)
        q*(pi/180)     ... %q (rad/s)
        r*(pi/180)     ... %r (rad/s)
        UX0(1)         ... %thrust (lbs)
        UX0(2)         ... %ele (deg)
        UX0(4)         ... %ail (deg)
        UX0(5)         ... %rud (deg)
        dLEF           ... %dLEF (deg)
        fi_flag_Simulink ...% fidelity flag
        ]';
OUT = feval('nlplant',xu);

Xdot = OUT(1:12,1);

% Create weight function
weight = [  0             ...%npos_dot
            0             ...%epos_dot
            5             ...%alt_dot
            phi_weight    ...%phi_dot
            theta_weight  ...%theta_dot
            psi_weight    ...%psi_dot
            2             ...%V_dot
            10            ...%alpha_dpt
            10            ...%beta_dot
            10            ...%P_dot
            10            ...%Q_dot
            10            ...%R_dot
            ];

cost = weight*(Xdot.*Xdot);
end
```

```
function [trim_state, trim_thrust, trim_control, dLEF, xu] = trim_F16(thrust,
elevator, alpha, ail, rud, vel, alt)
%================================================
%            Jimma University;
%      F-16/MATV nonlinear model trimming routine
%  for longitudinal motion, steady level flight
%
% Author: G. Kassahun B.
% Date:   January 19, 2019%
%
%      Added addtional functionality.
%      This trim function can now trim at three
%      additional flight conditions
%         -  Steady Turning Flight given turn rate
%         -  Steady Pull-up flight - given pull-up rate
%         -  Steady Roll - given roll rate
%
%================================================

global altitude velocity fi_flag_Simulink FC_flag
global phi psi p q r phi_weight theta_weight psi_weight

altitude = alt;
velocity = vel;
alpha = alpha*pi/180;  %convert to radians

% OUTPUTS: trimmed values for states and controls
% INPUTS:  guess values for thrust, elevator, alpha  (assuming steady level
flight)
% Initial Guess for free parameters
UX0 = [thrust; elevator; alpha; ail; rud];  % free parameters: two control
values & angle of attack
% Initialize some varibles
%
phi = 0; psi = 0;
```

```
p = 0; q = 0; r = 0;
phi_weight = 10; theta_weight = 10; psi_weight = 10;

switch FC_flag
    case 1
        % do nothing
    case 2
        r = input('Enter the turning rate (deg/s):  ');
        psi_weight = 0;
    case 3
        q = input('Enter the pull-up rate (deg/s):  ');
        theta_weight = 0;
    case 4
        p = input('Enter the Roll rate    (deg/s):  ');
        phi_weight = 0;
    otherwise
        disp('Invalid Selection')
%        break;
end

% Initializing optimization options and running optimization:
OPTIONS = optimset('TolFun',1e-10,'TolX',1e-
10,'MaxFunEvals',5e+04,'MaxIter',1e+04);

iter = 1;
for iter = 1:1:2

    [UX,FVAL,EXITFLAG,OUTPUT] = fminsearch('trimfun',UX0,OPTIONS);

    [cost, Xdot, xu] = trimfun(UX);

    disp(['Iteration ' num2str(iter) ' Trim Values and Cost:']);
    disp(['cost   = ' num2str(cost)])
    disp(['thrust = ' num2str(xu(13)) ' lb'])
    disp(['elev   = ' num2str(xu(14)) ' deg'])
    disp(['ail    = ' num2str(xu(15)) ' deg'])
```

```
    disp(['rud    = ' num2str(xu(16)) ' deg'])
    disp(['alpha  = ' num2str(xu(8)*180/pi) ' deg'])
    disp(['dLEF   = ' num2str(xu(17)) ' deg'])
    disp(['Vel.   = ' num2str(velocity) 'ft/s'])

    UX0 = UX;
end

% For simulink:
trim_state=xu(1:12);
trim_thrust=UX(1);
trim_ele=UX(2);
trim_ail=UX(4);
trim_rud=UX(5);
trim_control=[UX(2);UX(4);UX(5)];
dLEF = xu(17);
end
```