Table of Contents

1.Introduction

Aircrafts are one of the best way to travel long distance in 21st century. It's been over a century from 1903 when wright brothers tested their first flight to the year 2017 where the trials and experiments have been performed to create unmanned perfect autopilot systems. From the air travel, aerial firefighting to the aerial warfare the need of the better aircrafts is increasing at an emerging rate. The better design is a failure without a better controller for the stability of an Aircraft. A good controller must sense and control the parameters as per the requirement and to control anything, we need its Mechanical Model and some sort Intelligence to act on it. So, there are always 2 steps involved in any Aircraft i.e. mechanics of the Aircraft and application of control techniques to it. In this project, a controller is designed to control the Speed, Elevation, Rolling Angle and the Yaw movement of the Aircraft. Various types of controllers (linear, non-linear, adaptive, and fuzzy) have been used by the people over years to control it in a better way. A controller is proposed to create an autopilot for the F-16 model such that it follows the required flight requirements under certain flight conditions. For the mathematical modeling, the standard aircraft parameters are taken, and the dynamics of the system has been found via using various dynamics equations which allow us to present the model in the State Space Form. A PID controller is chosen to control the system state model.

2. F-16 Model Specifications

Over the years with various design enhancements a lot of F-16 models are suggested. For this project "The Lockheed Martin F-16V" configuration is considered which is further used in the project.

The Model specification are as follows:

Parameter	Symbol	Value
aircraft mass (kg)	m	9295.44
reference wing span (m)	b	9.144
reference wing area (m^2)	S	27.87
mean aerodynamic chord (m)	$\bar{c}$	3.45
roll moment of inertia ($kg.m^2$)	I_x	12874.8
pitch moment of inertia ($kg.m^2$)	I_y	75673.6
yaw moment of inertia ($kg.m^2$)	I_z	85552.1
product moment of inertia ($kg.m^2$)	I_{xz}	1331.4
product moment of inertia ($kg.m^2$)	I_{xy}	0.0
product moment of inertia ($kg.m^2$)	I_{yz}	0.0
c.g. location (m)	x_{cg}	$0.3\bar{c}$
reference c.g. location (m)	x_{cg_r}	$0.35\bar{c}$
engine angular momentum ($kg.m^2/s$)	h_E	216.9

Length	493 ft / 15027 m
Height	167 ft / 5090 m
Speed	1,500 mph (Mach 2+)
Wingspan	310 ft / 9449 m
Empty weight	20,300 lb / 9,207 kg
Engine thrust class	29,000 lb / 13,000
Design load factor	9 g

3. Mathematical Modeling and System Identification

To meet the object of controlling the Aircraft parameters for the stable flight there is a need of Mathematical Model which can be deduced by identifying the system under various conditions and assumptions.

3.1 Assumptions

There are always few assumptions to be made to simplify the model for better understanding. In this project, the model of F-16 is considered and few assumptions are made to make the calculations and the study easier.

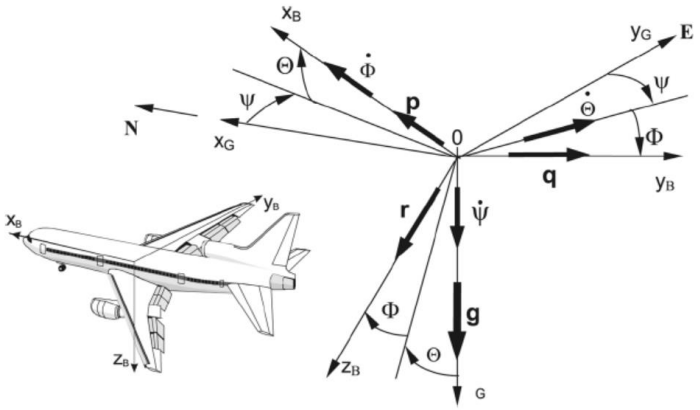
The assumptions are as follows:

- The Earth is considered flat i.e. the earth curvature is taken as infinity.
- The rotating effect of the earth is neglected therefore neglecting the Coriolis acceleration and the Centripetal force.
- The aircraft under consideration is assumed to be rigid body i.e. the aircraft assumed under quasi-static loading all the time, so all the inertial forces can be ignored.
- The Flight is assumed to be in the constant wind i.e. without any shear force which also means no turbulence.
- The Aircraft with exotic configuration is taken i.e. it must have a symmetry.
- As we know the rotating masses effect adversely on the stability of system. Here, we considered our model without any rotating masses i.e. propellers or turbines masses are not considered as rotating masses.
- There is always shift of mass in aircraft like fuel consumption but for this project, no shifting of mass is assumed.

3.2 Reference Frames

To describe any system there is a need of Reference Frame. All the modeling, assumptions, design are done keeping reference frame under consideration. All the sign conventions are defined according to the reference frame selected.

In this project to describe the Aircraft static and dynamic parameters there is a need of reference frame. For the easiness under assumptions the Earth-Fixed frame is taken as “Fe” and the Body-Fixed Frame is taken as “Fb”



3.2.1 Static Transformation

There is a need of method/procedure to define the relative orientation between two frames. In this project the “Euler Angles” have been used to find the homogenous transformation between two frames. The homogenous transformation in Frame 2 w.r.t Frame 1 is given by following, where the rotating angles are θ_x, θ_y, θ_z in x, y, z direction respectively.

$$T_{2,1} = \begin{bmatrix} (\cos\theta_y \cos\theta_z) & (\cos\theta_y \sin\theta_z) & (-\sin\theta_y) \\ \begin{pmatrix}\sin\theta_x \sin\theta_y \cos\theta_z \\ -\cos\theta_x \sin\theta_z\end{pmatrix} & \begin{pmatrix}\sin\theta_x \sin\theta_y \sin\theta_z \\ +\cos\theta_x \cos\theta_z\end{pmatrix} & (\sin\theta_x \cos\theta_y) \\ \begin{pmatrix}\cos\theta_x \sin\theta_y \cos\theta_z \\ +\sin\theta_x \sin\theta_z\end{pmatrix} & \begin{pmatrix}\cos\theta_x \sin\theta_y \sin\theta_z \\ -\sin\theta_x \cos\theta_z\end{pmatrix} & (\cos\theta_x \cos\theta_y) \end{bmatrix}$$

3.2.2 Dynamic Transformation

The aircraft is considered in the flight so the there is a need of method/procedure to find the time derivative between two frames. The acceleration of the center of gravity of an aircraft w.r.t. Earth's Frame is given as:

$$\begin{aligned}\{\mathbf{a}_C\}_E = \{\mathbf{a}_E\}_E &+ \{\mathbf{a}_C^E\}_E \\ &+ \{\dot{\Omega}_E\}_E \{\mathbf{r}_C^E\}_E \\ &+ 2\{\Omega_E\}_E \{\mathbf{v}_C^E\}_E \\ &+ \{\Omega_E\}_E^2 \{\mathbf{r}_C^E\}_E\end{aligned}$$

3.3 Equations of Motion

To get the dynamic and static relation between the input and control variables, a series of relations are required between various parameters. As per the assumption of the Aircraft symmetry, to derive various static and dynamic relations the Motion of the Aircraft is studied in two uncoupled modes as:

- Longitudinal Motion
- Lateral Motion

3.3.1 Force Equations

The External forces that effects the acceleration of an Aircraft is the summation weight, thrust of propulsion system and the aerodynamic forces. To obtain the equations the newton's 2nd law of motion is applied in the "Earth's" frame of reference and then the equation is transformed into the "Body's" frame of reference. So, the force on the body is given as:

$$\{\mathbf{F}\}_B = m\{\dot{\mathbf{v}}_B\}_B + m\{\Omega_B\}_B\{\mathbf{v}\}_B$$

This implies the rate of change of velocity in inertial frame as follows:

$$\{\dot{\mathbf{v}}_B\}_B = \frac{1}{m}\{\mathbf{F}\}_B - \{\Omega_B\}_B\{\mathbf{v}\}_B$$

Where,

$$\{\mathbf{v}\}_B = \begin{Bmatrix} u \\ v \\ w \end{Bmatrix}, \quad \{\dot{\mathbf{v}}_B\}_B = \begin{Bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{Bmatrix}$$

$$\{\Omega_B\}_B = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}$$

The total force on the body is given by:

$$\begin{aligned}\{\mathbf{F}\}_B &= \{\mathbf{F}_A\}_B + \{\mathbf{W}\}_B + \{\mathbf{T}\}_B \\ &= \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} + \begin{Bmatrix} -mg\sin\theta \\ mg\sin\phi\cos\theta \\ mg\cos\phi\cos\theta \end{Bmatrix} + \begin{Bmatrix} T\cos\epsilon \\ 0 \\ T\sin\epsilon \end{Bmatrix}\end{aligned}$$

As a result, the body axis-force equations are given by:

$$\begin{aligned}\dot{u} &= \frac{1}{m}(X + T\cos\epsilon_T) - g\sin\theta + rv - qw \\ \dot{v} &= \frac{1}{m}(Y) + g\sin\phi\cos\theta + pw - ru \\ \dot{w} &= \frac{1}{m}(Z + T\sin\epsilon_T) + g\cos\phi\cos\theta + qu - pv\end{aligned}$$

3.3.2 Moment Equations

Similarly, the moment equations are given by:

$$\{\mathbf{M}\}_B = I_B\{\dot{\omega}_B\}_B + \{\Omega_B\}_B I_B\{\omega_B\}_B$$

This implies the rate of change of angular rotation in inertial frame as follows:

$$\{\dot{\omega}_B\}_B = I_B^{-1}[\{\mathbf{M}\}_B - \{\Omega_B\}_B I_B\{\omega_B\}_B]$$

Where,

$$\{\mathbf{M}\}_B = \begin{Bmatrix} L \\ M + M_T \\ N \end{Bmatrix}$$

$$I_B = \begin{bmatrix} I_{xx} & 0 & -I_{xz} \\ 0 & I_{yy} & 0 \\ -I_{xz} & 0 & I_{zz} \end{bmatrix}$$

$$I_B^{-1} = \frac{1}{I_D}\begin{bmatrix} I_{zz} & 0 & I_{xz} \\ 0 & I_D/I_{yy} & 0 \\ I_{xz} & 0 & I_{xx} \end{bmatrix} \qquad I_D = I_{xx}I_{zz} - I_{xz}^2.$$

As a result, the body axis-moment equations are given by:

$$\dot{p} = \frac{I_{zz}}{I_D}[L + I_{xz}pq - (I_{zz} - I_{yy})qr] + \frac{I_{xz}}{I_D}[N - I_{xz}qr - (I_{yy} - I_{xx})pq]$$
$$\dot{q} = \frac{1}{I_{yy}}[M + M_T - (I_{xx} - I_{zz})pr - I_{xz}(p^2 - r^2)]$$
$$\dot{r} = \frac{I_{xz}}{I_D}[L + I_{xz}pq - (I_{zz} - I_{yy})qr] + \frac{I_{xx}}{I_D}[N - I_{xz}qr - (I_{yy} - I_{xx})pq]$$

3.3.3 Kinematic Relations

To find the Kinematic relations the Euler's Transformation is used. Which implies transformation from the local horizontal to the body axis is given by:

$$\begin{Bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{Bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{Bmatrix} p \\ q \\ r \end{Bmatrix}$$

So, the Roll Rate, Pitch Rate and Yaw Rate is given by:

$$\dot{\phi} = p + (q\sin\phi + r\cos\phi)\tan\theta$$
$$\dot{\theta} = q\cos\phi - r\sin\phi$$
$$\dot{\psi} = (q\sin\phi + r\cos\phi)\sec\theta$$

3.3.4 Body-Axis Navigation Relations

The position of the Aircraft relative to the Earth is found by integrating the aircraft velocity along its path and is given as:

$$\begin{Bmatrix} \dot{x}_E \\ \dot{y}_E \\ \dot{z}_E \end{Bmatrix} = T_{H,B} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix}$$

So, the body-axis navigation equations are given as:

$$\begin{aligned}
\dot{x}_E &= u(\cos\theta\cos\psi) + v(\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi) \\
&\quad + w(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) \\
\dot{y}_E &= u(\cos\theta\sin\psi) + v(\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi) \\
&\quad + w(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi) \\
\dot{h} &= -\dot{z}_E = u\sin\theta - v\sin\phi\cos\theta - w\cos\phi\cos\theta
\end{aligned}$$

3.4 States and Input Control Variables

The Aircraft can be stabled by controlling the four major input parameters within these limits.

Control	units	MIN.	MAX.	rate limit
Elevator	deg	-25	25	± 60 deg/s
Ailerons	deg	-21.5	21.5	± 80 deg/s
Rudder	deg	-30	30	± 120 deg/s
Leading edge flap	deg	0	25	± 25 deg/s

Also, the input variables have direct impact on the states of the system which must be controlled to attain the Aircraft stability.

3.5 Engine and Atmospheric Pressure Model

In this project the F-16 model that is taken under consideration is powered by turbofan jet engine which is controlled by the position of the throttle. The relation between the power delivered to the throttle position is given as:

$$\begin{aligned}
64.94\delta_{th} \quad & if \quad \delta_{th} \leq 0.77 \\
217.38\delta_{th} - 117.38 \quad & if \quad \delta_{th} > 0.77
\end{aligned}$$

When it comes to the Atmospheric Pressure model, for the flight at different heights and weather conditions the various parameters like Mach number, Air density changes which are used to find the thrust required to keep the plane at stable conditions.

The relation between those parameters are calculated as:

$$\begin{aligned} T &= T_0 - 0.0065h \\ \rho &= \rho_0 e^{-\frac{g}{287.05T}h} \\ a &= \sqrt{1.4 \times 287.05T} \end{aligned}$$

Where,

$T_0 = 288.15$

$p_{0=}1.225$

3.6 Trimming

when the aircraft is in steady state. Depending upon steady state various parameters are will be either zero or constant. Thus system (aircraft) can be said to be stable around that point (called trimming) . Thus system can be linearized around trimming point .

The system of equations in this case is :

x_dot = f(x, u)

y = c(x, u)

At trim point we have:

0 = f(xe, ue)

using Taylor series expansion and the perturbed variables which are ,

δx = x − xe

δy = y − ye

δu = u – ue

we have developed matlab code for finding trimming point. Even in steady state we have mane floating parameters , which give infinity solution for same flight condition . Thus this gives us freedom to choose various possible values. Thus cost minimizing function is used to find the minimum possible set of solution for trimming point. The trim routine will trim to the desired altitude and velocity. All other parameters will be varied until level flight is achieved. Trim tries to find an equilibrium point by minimizing the cost function which is as follows:

$$cost = 5\dot{h}^2 + W_\phi\dot{\phi}^2 + W_\theta\dot{\theta}^2 + W_\psi\dot{\psi}^2 + 2\dot{v_{tot}}^2 + 10\dot{\alpha}^2 + 10\dot{\beta}^2 + 10\dot{p}^2 + 10\dot{q}^2 + 10\dot{r}^2$$

where:

W_ϕ = weight of ϕ in cost function.
W_θ = weight of θ in cost function.
W_ψ = weight of ψ in cost function.

F-16 nonlinear model trimming routine for longitudinal motion, steady level flight. The code is also able to trim at three additional flight conditions.

1. Steady Turning Flight given turn rate.
2. Steady Pull-up flight
3. Steady Roll

The cost function final value can be different for different types of flights as follows:

W_ϕ, W_θ, W_ψ will have values of 10 for steady level flight. But,
$W_\phi = 0$, when trimming for steady roll flight.
$W_\theta = 0$, when trimming steady for pull-up flight.
$W_\psi = 0$, when trim-min for steady turning flight.

The idea is to minimize the cost function. The low fidelity model show values of cost of 10e-10 for it to be considered trimmed. It is noticed that after running a few iterations of the trim function the values for the

aileron and rudder continue to decrease. This is because this model lacks coupling between the longitudinal and lateral modes. However, we reasonably stop the iterations if the cost function results have reached a desired value.

3.7 Flight Conditions and Limits

The desired condition parameters for trimming the low fidelity model are:

	Units	Min	Max
Altitude	ft	5000	40000
AOA	deg	-10	45
Thrust	lbs	1000	19000
Elevator	deg	-25.0	25.0
Aileron	deg	-21.5	21.5
Rudder	deg	-30	30
Velocity	ft/s	300	900

The program allows to trim the F16 at which ever condition which choose during this step before linearization and control.

3.8 Linearization

To design the controller for the longitudinal and lateral modes, a linearized model is needed which can be obtained by using the simple linearized techniques. The techniques can be analytical or numerical in nature. Usually for the system of equations Jacobian is used which makes it much easier. The general expression for the linearization is as:

$$y = f(x_0) + fx_1(X_0)\Delta x_1 + fx_2(X_0)\Delta x_2 + ... + fx_n(X_0)\Delta x_n$$

With the initial conditions we will have equations of motions in more simpler form which will make the controlling much easier.

Now, From the linearized model we can extract the longitudinal and lateral directional modes. With these results we can plot the pole-zero mapping. We can also obtain the Bode plots for each of the control surfaces.

The states for the longitudinal mode are: altitude, pitch angle, magnitude of the total velocity, angle of attack and pitch rate. Furthermore, the inputs for this model are thrust and elevator deflection.

The states for the lateral direction are: roll angle, yaw angle, magnitude of the total velocity, side-slip angle, roll rate and yaw rate. Furthermore, the inputs for this this state space representation are thrust, aileron deflection and rudder deflection.

3.9 State Space Representation

To study the system there is need of standard representation which can make the controlling easier. Once the flight conditions are chosen, the state space model is found using the Trimmed Conditions with the Linearized Equations of Motion. The State Space Matrices can be varied which depends on the type of flight conditions we chose as follows:

The steady-state flight conditions and definitions are:

Steady Wings-Level Flight.	$\phi,\dot{\phi},\dot{\theta},\dot{\psi} \equiv 0$	
Steady Turning Flight	$\dot{\phi},\dot{\theta} \equiv 0$	$\dot{\psi} =$ turn rate
Steady Pull-Up	$\phi,\dot{\phi},\dot{\psi} \equiv 0$	$\dot{\theta} =$ pull-up rate
Steady Roll	$\dot{\theta},\dot{\psi} \equiv 0$	$\dot{\phi} =$ roll rate

For all flight conditions: $\dot{P},\dot{Q},\dot{R},\dot{U},\dot{V},\dot{W}$ (or $\dot{V}_T,\dot{\alpha},\dot{\beta} \equiv 0$)

From the state space we can find the resulting eigenvalues and save them for future calculations.

Let consider our flight under following conditions for stability analysis:

Height = 32000ft Velocity= 400ft/s

Center Of Gravity= .35ft Number of Iterations used for trimming = 5000

The above user-input gave us the following State Space using MATLAB:

```
A =

   -0.0232   -0.0000  -14.6815  -31.6936   -2.1247    0.1085   -0.0003    0.0002    0.0000    0.0000
         0         0 -394.0269  394.0269         0         0         0   -0.0024         0         0
   -0.0004    0.0000   -0.2807    0.0000    0.9635   -0.0001    0.0000         0   -0.0000   -0.0000
         0         0         0         0    1.0000         0         0         0         0         0
   -0.0000    0.0000    0.2317         0   -0.3955         0         0         0         0   -0.0029
         0         0         0         0         0   -1.0000         0         0         0         0
   -0.0000    0.0000    0.0000    0.0000    0.0000   -0.0000   -0.0935    0.0772    0.2271   -0.9710
         0         0         0         0         0         0         0         0    1.0000    0.2324
   -0.0000    0.0000   -0.0001         0    0.0003         0  -11.3585         0   -0.9612    0.5462
   -0.0000    0.0000   -0.0000         0    0.0025         0    1.8075         0   -0.0225   -0.1522

>> B

B =

         0   -0.1388    0.0000    0.0000
         0         0         0         0
         0   -0.0006         0         0
         0         0         0         0
         0   -0.0420         0         0
   37.6920         0         0         0
         0    0.0000    0.0001    0.0002
         0         0         0         0
         0         0   -0.1629    0.0300
         0         0   -0.0060   -0.0147

>> C

C =

    0.0049   -0.0000    3.6695         0    0.4565         0   -0.0000         0         0         0
         0         0         0         0    1.0000         0         0         0         0         0
         0         0   57.2958         0         0         0         0         0         0         0

>> D

D =

         0    0.0079         0         0
         0         0         0         0
         0         0         0         0
```

4. Control Strategies

4.1 Controllability and Observability

Controllability Check:

To check the controllability of the system, controllability matrix ‘Co’ is produced and is checked for the rank. If this matrix is full rank, then the system is said to be controllable.

$$C_o = |B \quad AB \quad \cdots \quad A^{n-1}B|$$

No. of Uncontrollable states = Length(A)- Rank(Co)= 10-10 = 0

Hence, the system is Controllable.

Observability Check:

To check the observability of the system, observability matrix 'Ob' is produced and is checked for the rank. If this matrix is full rank, then the system is observable

$$O_b = \begin{vmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{vmatrix}$$

No. of Unobservable states = Length(A)- Rank(Ob)= 10-10 = 0

Hence, the system is Observable.

4.2 Control by PID

In this method of controlling a separate PID controlling parameters are found for each state/output to be controlled. The advantage of this method is the ability to use the three control terms of proportional, integral and derivative influence on the controller output to apply accurate and optimal control.

4.3 Control by LQR

The linear quadratic regulator (LQR) is a well-known design technique that provides practical feedback gains. However, difficulty in finding the right weighting factors limits the application of the LQR based controller synthesis.

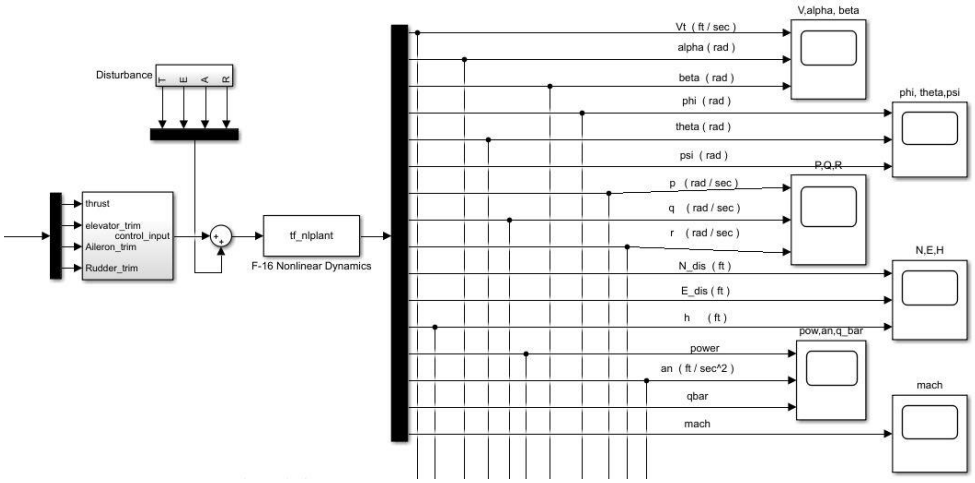
4.4 Control with the Estimator

In the practical situations we don't have enough sensors to have all the states available always. Therefore, the estimator is used to estimate the states the system states for the required stable conditions and the output is calculated using them which is further feedback to the real system.

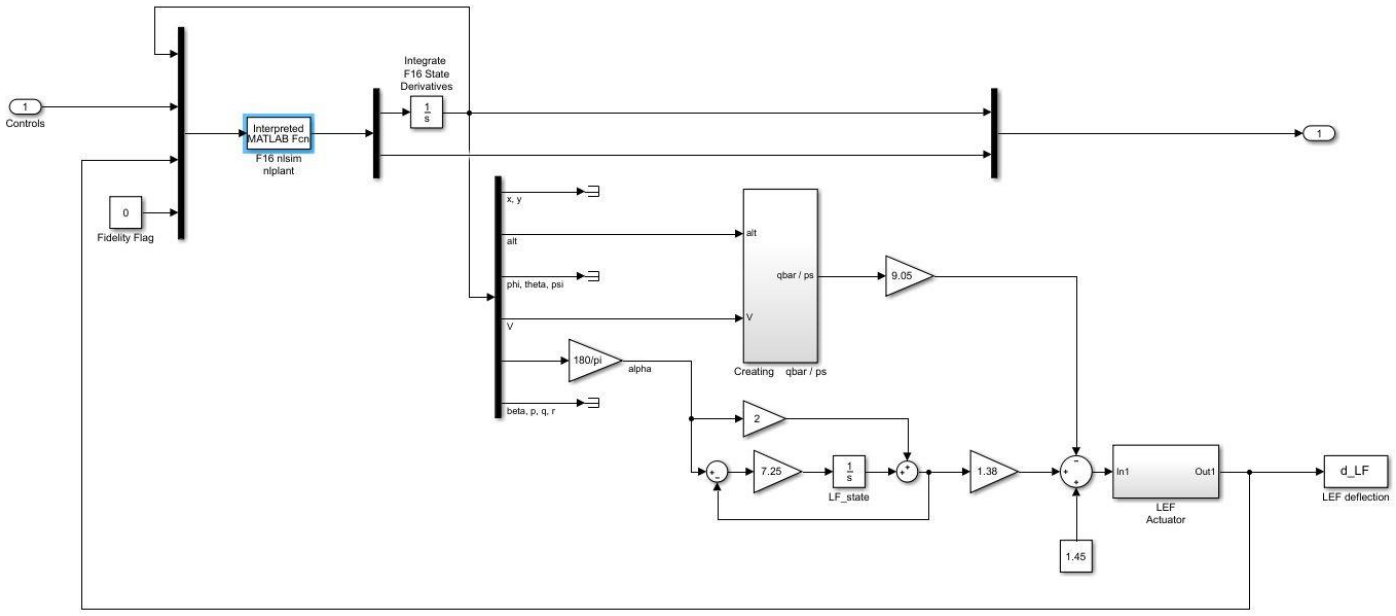
5. MATLAB/SIMULINK Model Representation

Since we are dealing with uncoupled system to study the longitudinal and the lateral motion of an Aircraft. So, the basic PID controller is used to control and stabilize the parameters. A linear and non-linearized model is designed as per the equation for the desired flight conditions.

5.1 Non-Linear Model



Non Linear Model



Non-Linear Simulink Model (tf_nplant)

5.2 Linear Model

For the simulation and controller design, let consider our flight under following conditions for stability analysis:

Height = 32000ft Velocity= 400ft/s

Center Of Gravity= .35ft Number of Iterations used for trimming = 5000

Using the above inputs the state derivatives are trimmed and we found the trim points as:

control_trim =[1.0922;-19.4958;-2.3649;2.7123]
state_trim =1.0e+04 *[0.0400;0.0000;0.0000;0;0.0000;0;0;0;0; 0; 0;4.0000;0.0120]

The above user-input gave us the following State Space using MATLAB via taking JACOBIAN (taking higher order as zero) and arranging them in matrix form :

```
A =

   -0.0232   -0.0000  -14.6815  -31.6936   -2.1247    0.1085   -0.0003    0.0002    0.0000    0.0000
         0         0 -394.0269  394.0269         0         0         0   -0.0024         0         0
   -0.0004    0.0000   -0.2807    0.0000    0.9635   -0.0001    0.0000         0   -0.0000   -0.0000
         0         0         0         0    1.0000         0         0         0         0         0
   -0.0000    0.0000    0.2317         0   -0.3955         0         0         0         0   -0.0029
         0         0         0         0         0   -1.0000         0         0         0         0
   -0.0000    0.0000    0.0000    0.0000    0.0000   -0.0000   -0.0935    0.0772    0.2271   -0.9710
         0         0         0         0         0         0         0         0    1.0000    0.2324
   -0.0000    0.0000   -0.0001         0    0.0003         0  -11.3585         0   -0.9612    0.5462
   -0.0000    0.0000   -0.0000         0    0.0025         0    1.8075         0   -0.0225   -0.1522

>> B

B =

         0   -0.1388    0.0000    0.0000
         0         0         0         0
         0   -0.0006         0         0
         0         0         0         0
         0   -0.0420         0         0
   37.6920         0         0         0
         0    0.0000    0.0001    0.0002
         0         0         0         0
         0         0   -0.1629    0.0300
         0         0   -0.0060   -0.0147

>> C

C =

    0.0049   -0.0000    3.6695         0    0.4565         0   -0.0000         0         0         0
         0         0         0         0    1.0000         0         0         0         0         0
         0         0   57.2958         0         0         0         0         0         0         0

>> D

D =

         0    0.0079         0         0
         0         0         0         0
         0         0         0         0
```

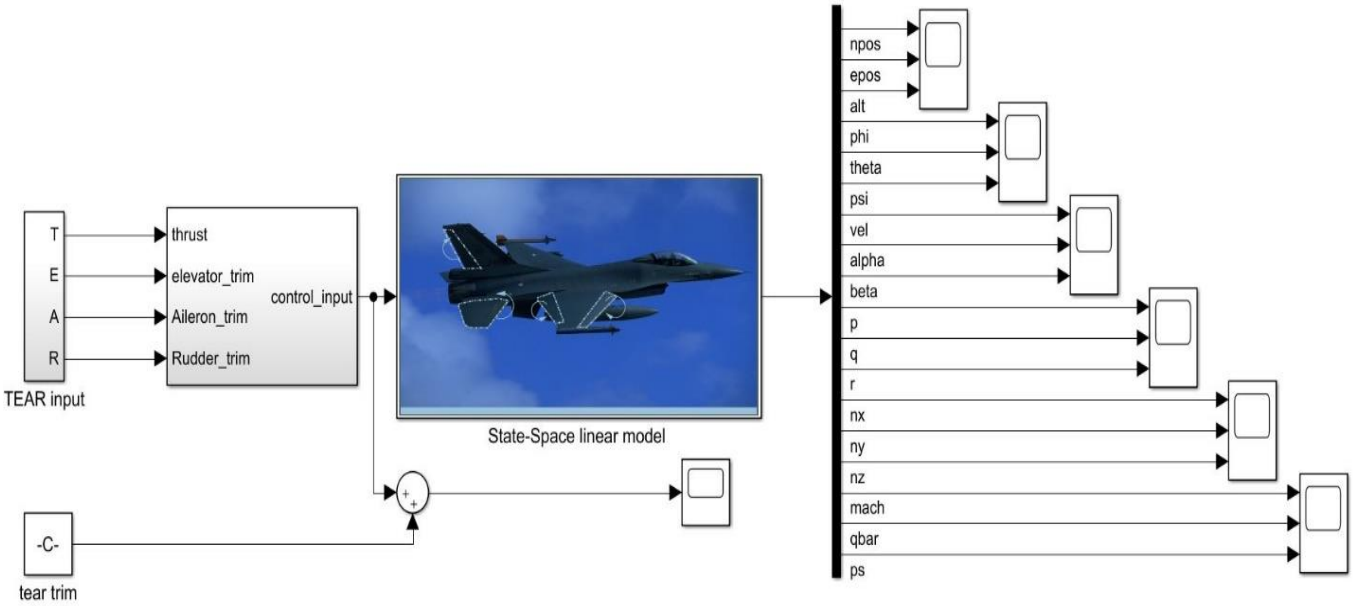
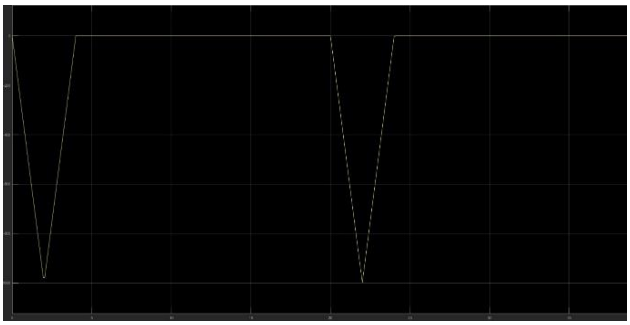


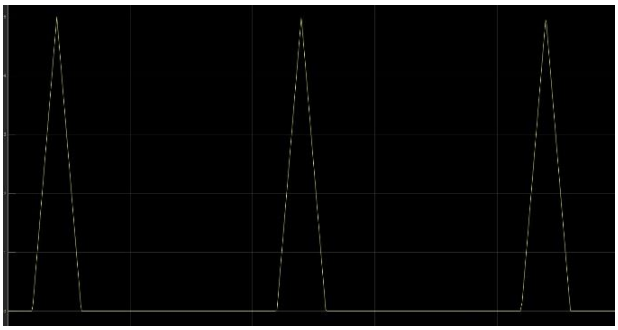
Figure 1: Linear State Space Model (For all States as output)

For this we have given disturbance as the

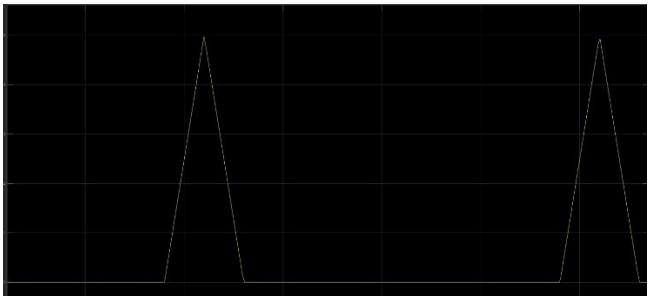
- Aerodynamic Disturbance
- Disturbance in the Elevation
- Disturbance in the Aileron



Aerodynamic Disturbance



Elevation Disturbance

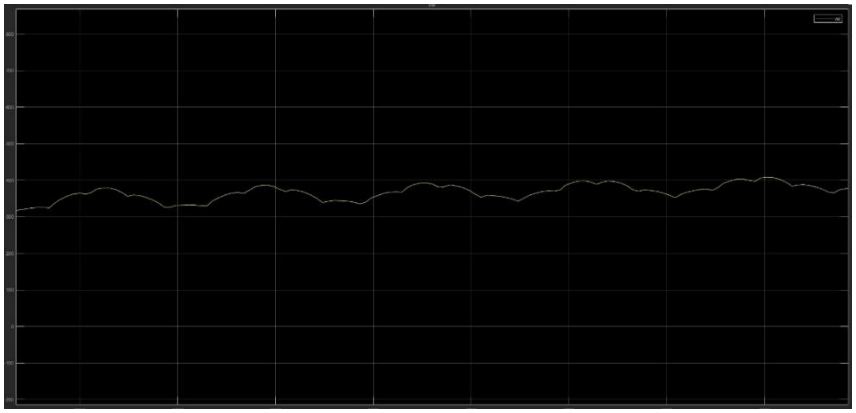


Disturbance in Aileron

On the action of the above disturbances we got the following results for:

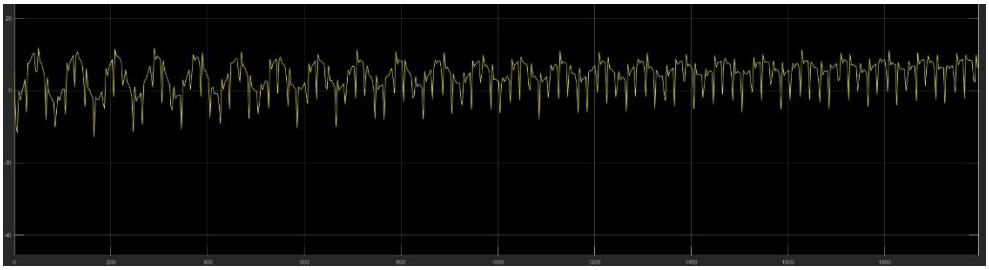
We can see that the Aircraft is stable by nature but due to very small damping we can see high fluctuation in states. So, there is need of inner-loop controller (SAS) to stabilize the states of the system mode.

- Velocity



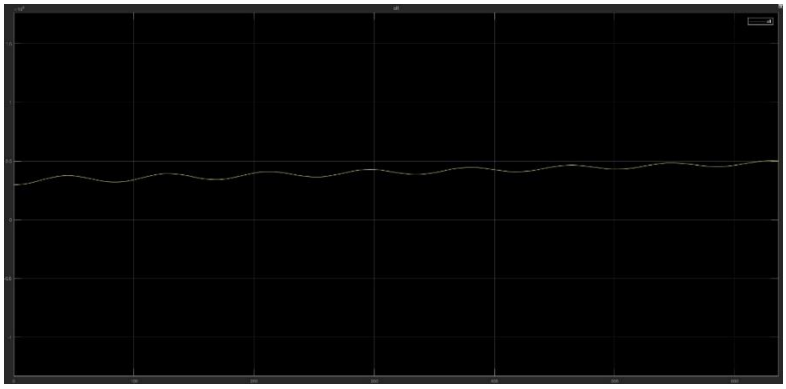
Velocity response with disturbance

- Angle of attack



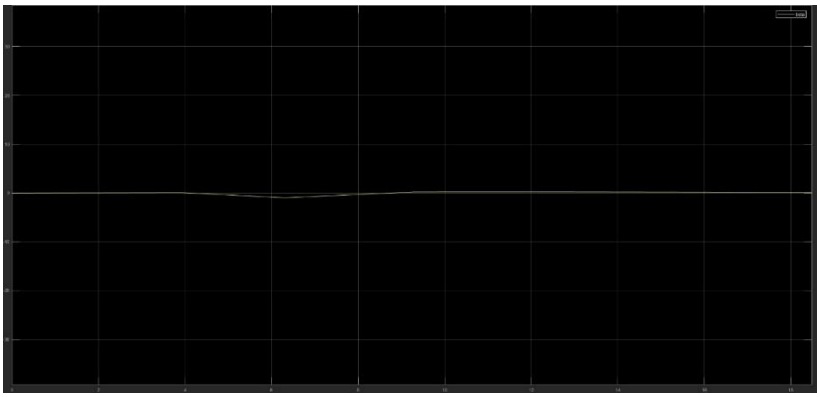
Angle of attack response with disturbance

- Altitude



Altitude response with disturbance

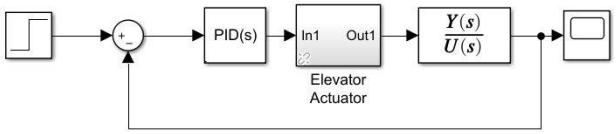
- Sideslip



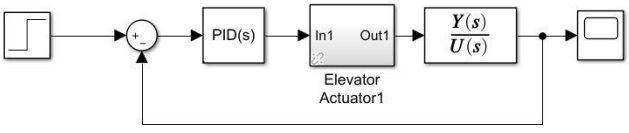
Sideslip Response with disturbance

6. Controller Design

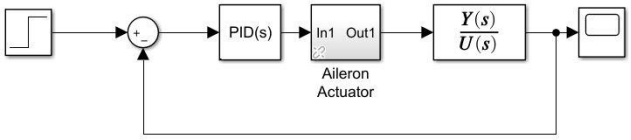
For the system stability a controller has been designed



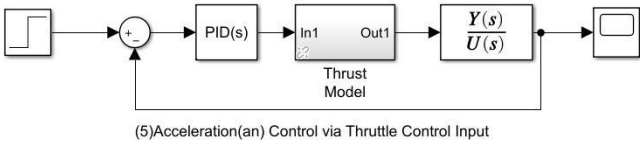
(1)Pitch Rate(q) Control via Elevation Control Input



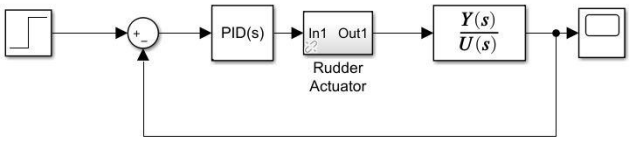
(4)height(h) Control via Elevation Control Input



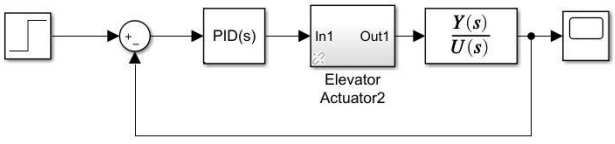
(2)Rol Angle(Phi) Control via Aerlon Control Input



(5)Acceleration(an) Control via Thruttle Control Input



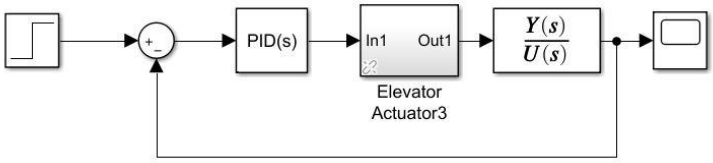
(3)SideSlip(Beta) Control via Rudder Control Input



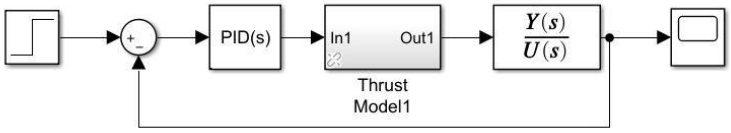
(6)Angle Of Attack(alpha) Control via elevator Control Input

PID Controlling of significant parameters

To Control Longitudinal states, we need to design SAS for (angle of attack Vs Elevator) and (Acceleration Vs Throttle)

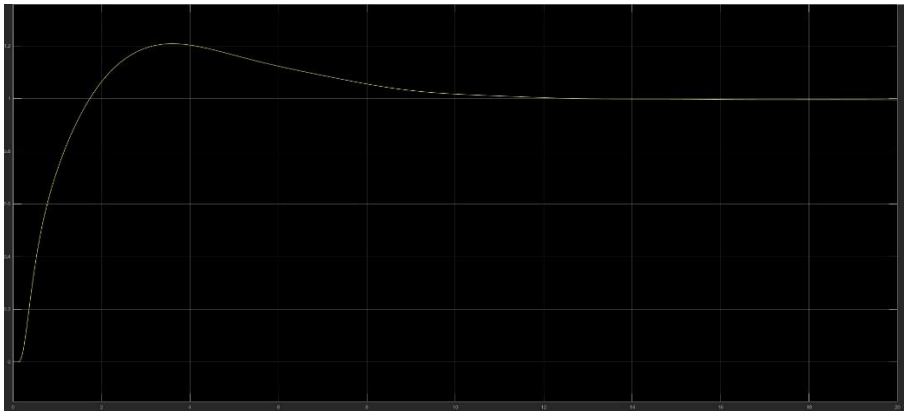


(6)Angle Of Attack(alpha) Control via elevator Control Input



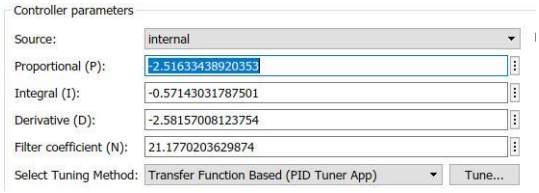
Acceleration(an) Control via Thruttle Control Input

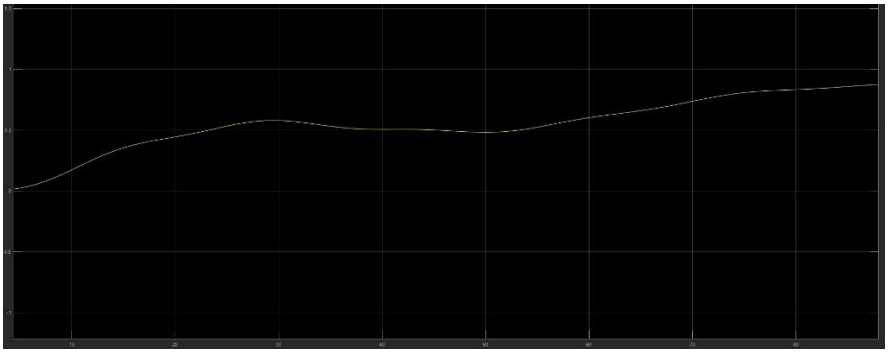
Both the parameters, Angle of attack and the Acceleration are controlled via PID using PID-Tuner in MATLAB. The response of the tuning is found as:



Angle Of Attack Response

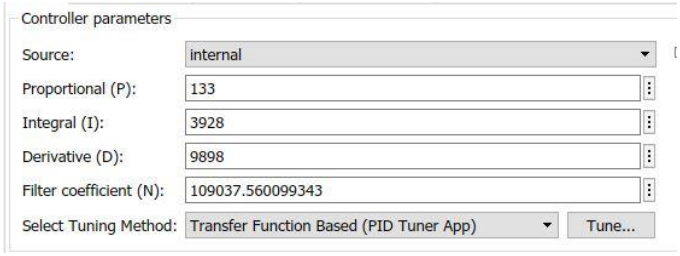
To control the Angle of attack the PID parameters are found as follows:

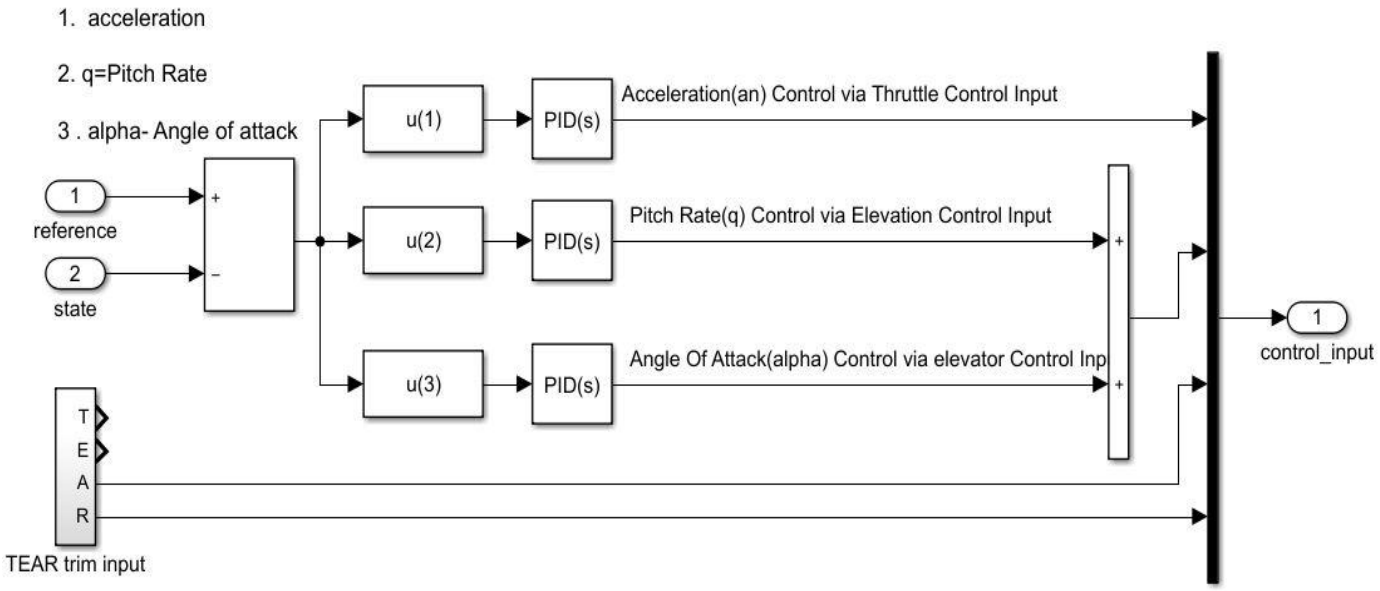




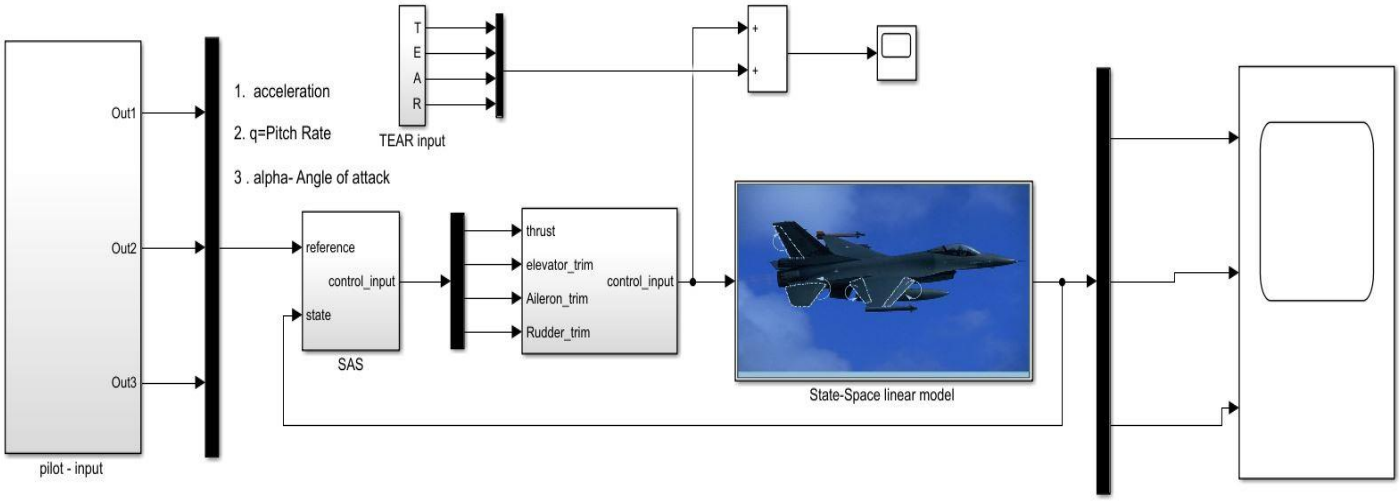
Acceleration Response

To control the Acceleration the PID parameters are found as follows:

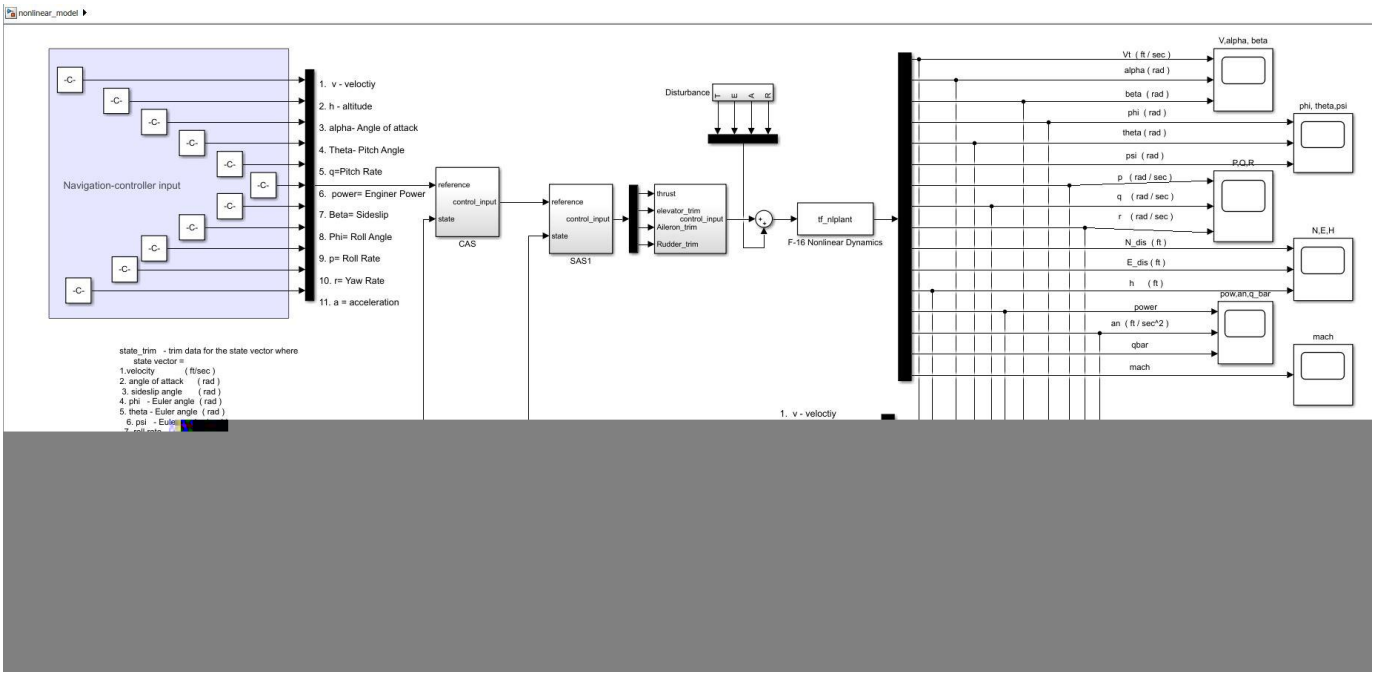




SAS- Inner loop

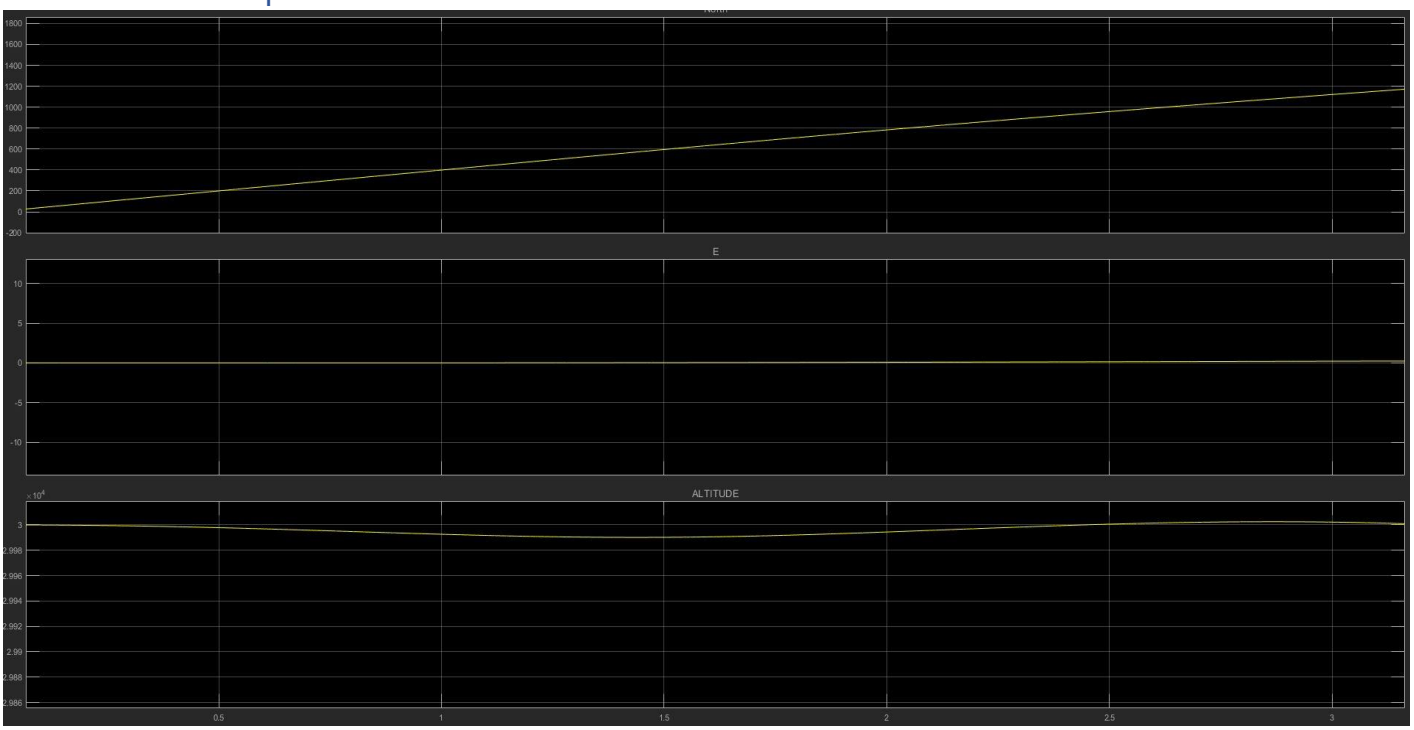


Final Longitudinal Linear Model (Acceleration, Pitch Rate and Angle of attack as output)

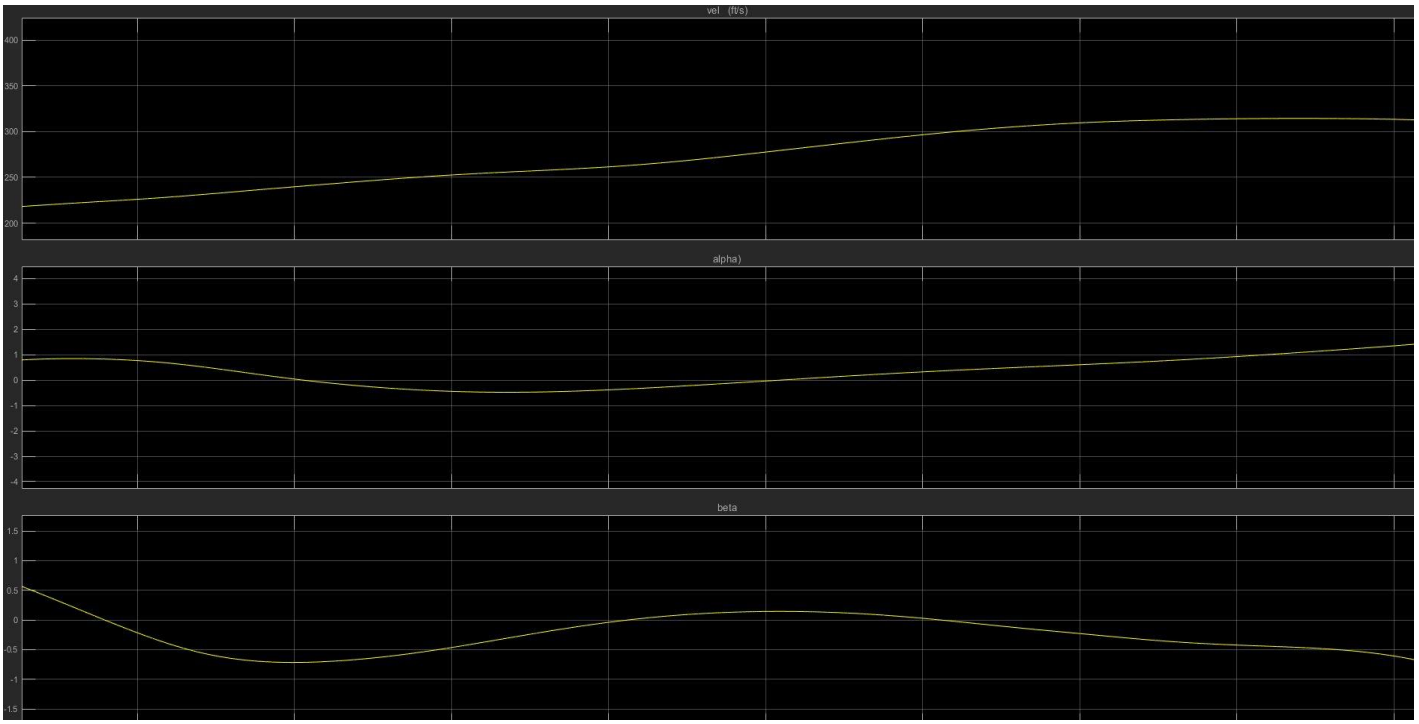


Non-Linear Model with controller

6.1 Non-Linear Response



Non Linear response for North, East and Altitude Position vs time



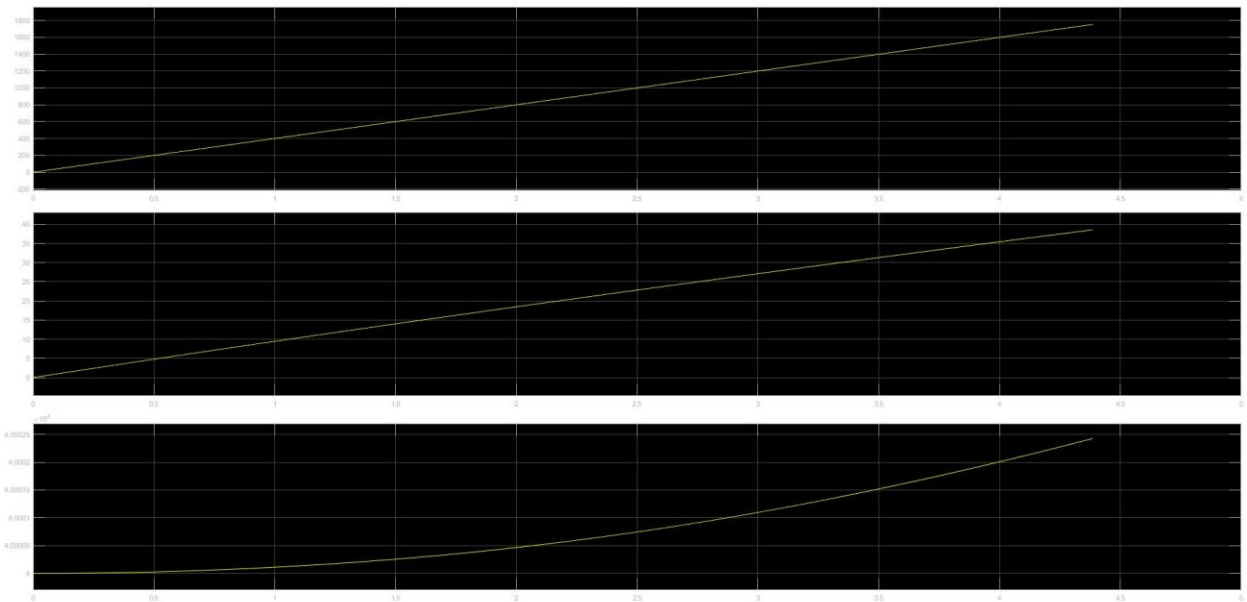
Non Linear response for Velocity, Roll Angle and Sideslip vs time

Trimming parameters for the Steady State Turning

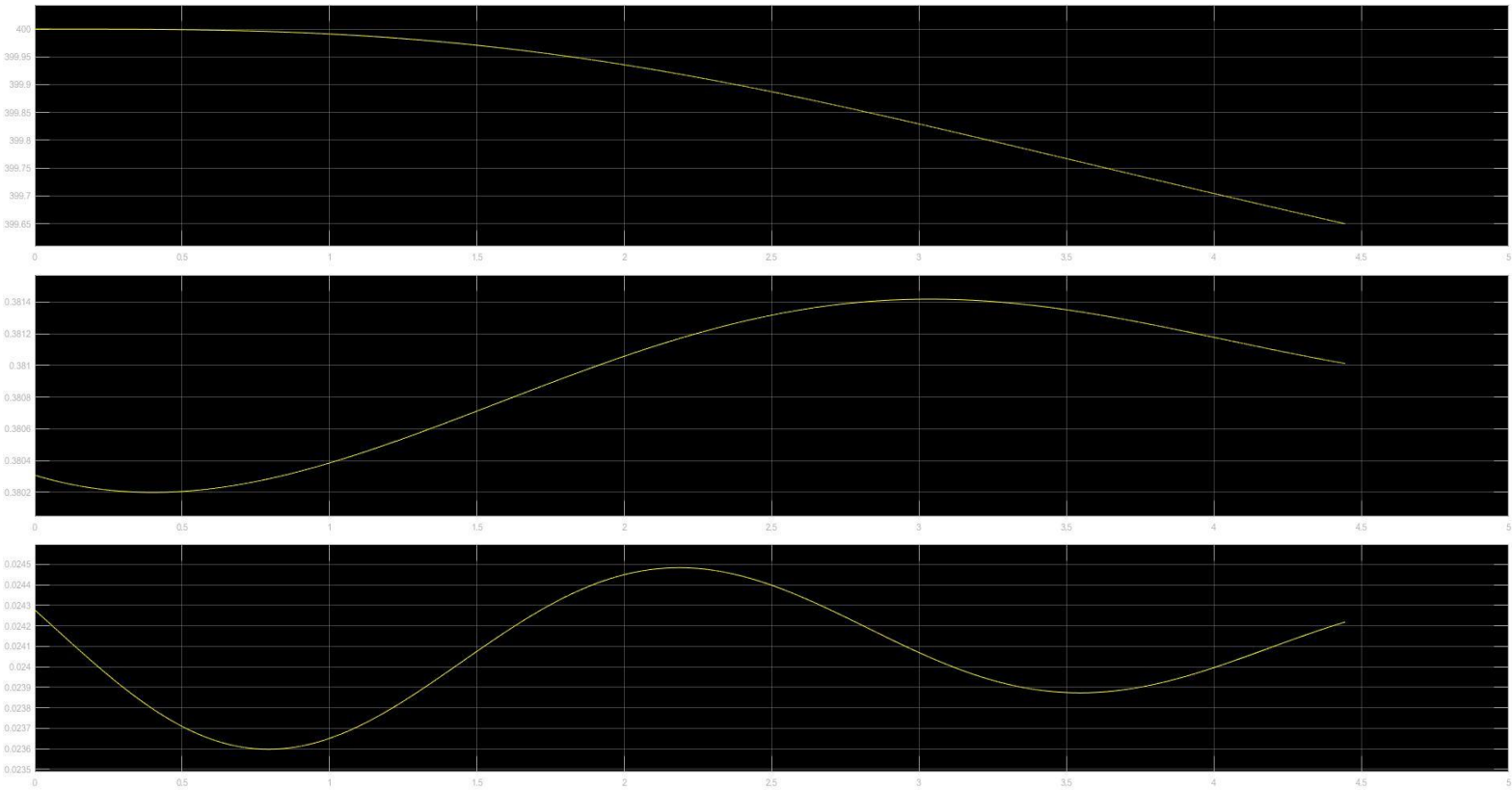
control_trim = [1.0922;-19.4958;-2.3649; 2.7123]

state_trim=1.0e+04*[0.0400;0.0000;0.0000;0;0.0000;0;0;0;0;0;0;4.0000;0.0120]

Which gives the following response:



Non Linear response for North, East and Altitude Position


400
399.95
399.9
399.85
399.8
399.75
399.7
399.65
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

0.3814
0.3812
0.381
0.3808
0.3806
0.3804
0.3802
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

0.0245
0.0244
0.0243
0.0242
0.0241
0.024
0.0239
0.0238
0.0237
0.0236
0.0235
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

7. Conclusion

The controller is good for an aircraft still there are some conditions where the model don't hold good and needs human intervention. The longitudinal SAS have been designed based on the linear model using classical control (PID tuning) techniques. Each channel has to be tuned to achieve a good response. In nonlinear simulation, the controller performance decreased because of dependency of states and inputs on each other. However, the robustness of the controller is expected because of the integral presence in PID, the system will eventually track the reference value.

The developed model for the Linear and Non-Linear system with the MATLAB SIMULINK model of the controller has been attached with the report in the CD

8. Future Work

As a future development, a different way to obtain inertial trajectory control is of great interest. This would allow the control of the aircraft to travel through a predetermined path through a certain set of target points in three-dimensional space. There are two main methods that could help in achieving this control. First, it would be to separate the guidance and control laws. This means that the given reference trajectory is converted to a form of velocity and attitude commands for the controller. However, a second method that could be implemented for higher accuracy is to integrate the guidance and control into one system. This could be done by trimming the flight conditions along the reference trajectory. An automatic navigation system can be developed as aircrafts fall under non-holonomic states. A nonholonomic system in physics and mathematics is a system whose state depends on the path taken to achieve it. As although we have full control on system states, but we have kinematic constrains of system. Thus, is will be interesting to develop a navigation control system, One the possible way to do this using **Lyapunov energy** function and then defining **sliding mode control** algorithm.

Appendix

run_me.m to initiate the program

```
disp(' You have chosen  linearze model. ');
disp('  ');
disp(' linearized model : x_dot = A * x + B * u' );
disp('                    y   = C * x + D * u');
disp('  ');
disp('  ------system States vector  --------                ');
disp(' x= [ vt; h; alpha; theta; Q; pow; beta; phi; P: R] ');
disp('     where x(1) -vt    ( ft/sec )    - velocity             ');
disp('           x(2) -h     ( ft )        - altitude             ');
disp('           x(3) -alpha ( rad )       - angle of attack      ');
disp('           x(4) -theta ( rad )       - Euler angle          ');
disp('           x(5) -Q     ( rad/sec )   - pitch rate           ');
disp('           x(6) -pow                 - power                ');
disp('           x(7) -beta  ( rad )       - sideslip angle       ');
disp('           x(8) -phi   ( rad )       - Euler angle          ');
disp('           x(9) -P     ( rad/sec )   - roll rate            ');
disp('           x(10)-R     ( rad/sec )   - yaw rate ];          ');
disp('                                                            ');
disp('  ---- system Control vector --------                  ');
disp('   u = [ thtl ;el  ;ail   ;rdr]                        ');
disp('   where  thtl ( 0 ~ 1.0 )    - throttle setting       ');
disp('          u(1) -el   ( deg )       - elevator deflection    ');
disp('          u(2) -ail  ( deg )       - aileron deflection     ');
disp('          u(3) -rdr  ( deg )       - rudder deflection ];   ');
disp('                                                           ');

disp('  ---- Output  vector( first 4 longitidunal last 6 lateral -------- ');
disp(' y= [ vt; h; alpha; theta; Q; pow; beta; phi; P: R] ');
disp('     where y(1) -vt    ( ft/sec )    - velocity             ');
disp('           y(2) -h     ( ft )        - altitude             ');
disp('           y(3) -alpha ( rad )       - angle of attack      ');
disp('           y(4) -theta ( rad )       - Euler angle          ');
disp('           y(5) -Q     ( rad/sec )   - pitch rate           ');
disp('           y(6) -pow                 - power                ');
disp('           y(7) -beta  ( rad )       - sideslip angle       ');
disp('           y(8) -phi   ( rad )       - Euler angle          ');
disp('           y(9) -P     ( rad/sec )   - roll rate            ');
disp('           y(10)-R     ( rad/sec )   - yaw rate ];          ');

                                  disp('  ');

% disp(' simulation of nonlinear model. ');
disp('  ');
disp(' To simulate the aircraft, please choose the flight conditions to calculate optimized trim conditions');
disp(' The control variables are left as inputs to the block of  nonlinear model');
disp(' Please input the following flight parameters:');
disp('  ');
velocity = input(' true velocity ( 350 ~ 1000 ft/sec ) : ');
disp('  ');
altitude = input(' altitude ( 0 ~ 40000 ft ) : ');
disp('  ');
xcg = input(' center of gravity position ( 0.2 ~ 0.5, reference xcg = 0.35 )  ');
disp('  ');
[ state_trim, control_trim ] = find_trim( velocity, altitude, xcg )              ;
disp('  ');
disp( [ ' trimmed angle of attack    ( rad )     = ', num2str( state_trim(2) ) ] )  ;
disp( [ ' trimmed sideslip angle     ( rad )     = ', num2str( state_trim(3) ) ] )  ;
disp('  ');
disp( [ ' trimmed throttle          ( 0-1 )     = ', num2str( control_trim(1) ) ] );
```

```
        disp( [ ' trimmed elevator          ( deg )    = ', num2str( control_trim(2) ) ] );
        disp( [ ' trimmed aileron           ( deg )    = ', num2str( control_trim(3) ) ] );
        disp( [ ' trimmed rudder            ( deg )    = ', num2str( control_trim(4) ) ] );
        disp('  ');
        disp(' transfer function matrix (TF) gives dynamics of the airplane in the horizontal');
        disp(' TF matrix is [3*4] order ');
        disp(' where output = y = [ an; q; alpha ] and input is T.E.A.R. ');

%         rlocus(TF(1,1)) ;

        disp(' state stability anaysis ');
        disp(' taking states as output , where first 4 elememts are longitudnal and next 6 are lateral parameters, 11th state is accelerastion ');
        %disp(' state stability anaysis calculating states as output new C and D matrix will be  ');
        [ A, B, C, D ] = ssm_f16( velocity, altitude, xcg );
        C_1 =eye(10);
        D_1 =zeros(10,4);
        sys_y = ss(A,B,C,D);
        TF_y = tf(sys_y);
        sys_x = ss(A,B,C_1,D_1);
        TF_x = tf(sys_x);
        [num_y_2,den_y_2] = tfdata(TF_y(2,2)) ; num_y_2=cell2mat(num_y_2);den_y_2=cell2mat(den_y_2);
        [num_x_8,den_x_8] = tfdata(TF_x(8,3)) ;num_x_8=cell2mat(num_x_8);den_x_8=cell2mat(den_x_8);
        [num_x_7,den_x_7] = tfdata(TF_x(7,4)) ;num_x_7=cell2mat(num_x_7);den_x_7=cell2mat(den_x_7);
        [num_x_2,den_x_2] = tfdata(TF_x(2,2)) ;num_x_2=cell2mat(num_x_2);den_x_2=cell2mat(den_x_2);
        [num_y_1,den_y_1] = tfdata(TF_y(1,1)) ;num_y_1=cell2mat(num_y_1);den_y_1=cell2mat(den_y_1);
        [num_y_3,den_y_3] = tfdata(TF_y(3,2)) ;num_y_3=cell2mat(num_y_3);den_y_3=cell2mat(den_y_3);

        disp( [ 'now you can simulate state space model' ] );

        k = input(' enter any key to continue ');
        clear k

        sim( 'simF16_openloop', [ 0, 10 ]);
        % plot the figures
```

Code to find trim point

```
% [ state_trim, control_trim ] = find_trim( velocity, altitude, xcg )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%              find_trim.m                     %%
%%  This is the function to find the trim condition for   %%
%%  straight&level flight at given velocity and altitude. %%
%%  The trim data are derived by optimizing the cost      %%
%%  function cost_energy.m using simplex search.          %%
%% ---- Input Variables --------                          %%
%% velocity (ft/sec)- true velocity                       %%
%% altitude (ft)    - altitude                            %%
%% xcg            - center of gravity position as         %%
%%                  fraction of mean aerodynamic chord    %%
%%                                                        %%
%% ---- Output Variables --------                         %%
%% state_trim   - trim data for the state vector where    %%
%%                                                        %%
%%            state vector =                              %%
%%                                                        %%
%%            [ 1.velocity           ( ft/sec )      %%
%%              2. angle of attack      ( rad )        %%
%%              3. sideslip angle       ( rad )        %%
%%              4. phi   - Euler angle  ( rad )        %%
%%%             5. theta - Euler angle  ( rad )        %%
%%%             6. psi   - Euler angle  ( rad )        %%
%%              7. roll rate           ( rad/sec )     %%
%%              8. pitch rate          ( rad/sec )     %%
%%              9. yaw rate            ( rad/sec )     %%
```

```
%%              10.north displacement   ( ft )          %%
%%              11.east displacement    ( ft )         %%
%%              12.altitude            ( ft )          %%
%%              13.power  ( percent, 0 <= pow <= 100 ) ] %%
%%                                          %%
%% control_trim - trim data for the control vector where  %%
%%                                          %%
%%             control vector =                    %%
%%                                          %%
%%             [ throttle setting ( 0 - 1 )           %%
%%               elevon          ( deg )            %%
%%               aileron         ( deg )            %%
%%               rudder          ( deg ) ];          %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ state_trim, control_trim ] = find_trim( velocity, altitude, xcg )
%---- flight steady_states --------------------------
global climb_angle; % rate-of-climbing steady_state
% climb_angle = input( 'Input the climb angel(deg)' );
climb_angle = 0.0;
global coordi_turn; % coordinate turn steady_state
coordi_turn = 0;
global stab; % stability-axis roll steady_state
stab = 0;
global skid_turn; % skidding turn steady_state
skid_turn = 0;
global rad_gamma; % flight path angle gamma in radian
rad_gamma = 0;
global phi_r; % reference phi
phi_r = 0;
global roll_rate; % reference roll rate
roll_rate = 0;
global pitch_rate; % reference pitch rate
pitch_rate = 0;

% OUTPUTS: trimmed values for states and controls
% INPUTS:  guess values for thrust, elevator, alpha  (assuming steady level flight)
% Initial Guess for free parameters
phi_weight = 10; theta_weight = 10; psi_weight = 10;
phi = 0; psi = 0;
p = 0; q = 0; r = 0;
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:  ');
switch FC_flag
   case 1
      disp('Trimming for Steady Wings-Level Flight.');
      % do nothing
   case 2
      disp('Trimming for Steady Turning Flight.');
      r = input('Enter the turning rate (deg/s):  ');
      psi_weight = 0;
   case 3
      disp('Trimming for Steady Pull-Up Flight.');
      q = input('Enter the pull-up rate (deg/s):  ');
      theta_weight = 0;
   case 4
      disp('Trimming for Steady Roll Flight.');
      p = input('Enter the Roll rate    (deg/s):  ');
      phi_weight = 0;
   otherwise
      disp('Invalid Selection')
%       break;
```

```
end

weight = [ 1    100     100      phi_weight   theta_weight  psi_weight  10    10   10   0   0    0  0 ];
%weight = [ V_dot alpha_dpt  beta_dot  phi_dot      theta_dot    psi_dot     P_dot  Q_dot R_dot P_do Q_dot h  power];
%---- data --------------------------
rtod = 57.29577951; % radian to degree
no_step = 5000; % no. of iteration steps for trimming
disp('  ');
read_no = input('Please input the # of trim iterations ( default = 5000 ):  ');
if read_no == []
   no_step = read_no;
end
disp('  ');
disp('-----------------------------------------------------');
epsilon = -1.0;
%---- initial condition ----
x0 = [ velocity; 0.0; 0.0; 0.0; 0.0; p  ; q  ; r  ; 0.0; 0.0; altitude; 90 ];
   %[  1     ; 2  ; 3  ; 4  ; 5 ;  6  ; 7  ; 8  ; 9  ; 10 ; 11 ; 12     ; 13 ]
 u0 = [  0.73; -1.0; 0.0 ; 0.0];
   % [  T   ;  E  ; A   ; R  ];
%---- define initial steady state for iteration --------------------
s = [ u0(1); u0(2); x0(2); u0(3); u0(4); x0(3) ];
ds = [ 0.2; 1.0; 0.02; 1.0; 1.0; 0.02 ];

%---- simplex(trim) algorithm ------------------
init_cost =  cost_energy( x0, u0, s, xcg,weight);
[ s_trim, f_final ] = ss_trim( s, ds, x0, u0, no_step, epsilon, xcg, weight );

%---- output the trim result ------------
control_trim = u0;
control_trim(1) = s_trim(1);
control_trim(2) = s_trim(2);
control_trim(3) = s_trim(4);
control_trim(4) = s_trim(5);
state = x0;
state(2) = s_trim(3);
state(3) = s_trim(6);
final_cost = cost_energy( state, control_trim, s_trim, xcg,weight );
state(13) = p_thro( control_trim(1) );
state_trim = steady_state( state );
control_trim;
```

Code to find State Space model (linear)

```
%{
   SSM_F16.m
   x_dot = A * x + B * u
     y  = C * x + D * u
   [ A, B, C, D ] = SSM_F16( velocity, altitude, xcg )
   A, B, C, D are state space matrix of linearized model
   for variouus flight conditions
   i.e. 1. traight
        2. level flight at  sepecified velocity
        3. altitude
        4. c.g. position with zero banking angle
also  longitudinal and the lateral states are decoupled.

system Control ( input)
   u=[ thro; el; ail ]
where
   thro ( 0 ~ 1.0 )    throttle setting
   el   ( deg )        elevon deflection
   ail  ( deg )        aileron deflection
```

```
sysetm Stat
  x = [ vt; h; alpha; theta; Q; pow; beta; phi; P;R]
where
   vt    ( ft/sec )    velocity
   h     ( ft )        altitude
   alpha ( rad )       angle of attack
   theta ( rad )       Euler angle
   Q     ( rad/sec )   pitch rate
   pow                 power
   beta  ( rad )       sideslip angle
   phi   ( rad )       Euler angle
   P     ( rad/sec )   roll rate
   R     ( rad/sec )   yaw rate ]

 rdr  ( deg )        rudder deflection ]

system Output
y =[ an   ;q  ;alpha]
where
an    ( ft/sec^2 )  - normal acceleration
 q    ( rad/sec )   - pitch rate
alpha ( rad )       - angle of attack

 Function Inputs
velocity ( ft/sec )- true velocity
altitude ( ft )    - altitude
xcg                - CoG fraction of mean aerodynamic chord
 %}
function [ A, B, C, D ] = ssm_f16( velocity, altitude, xcg, x_control)

%---- trim the flight ----
[ state_trim, control_trim ] = find_trim( velocity, altitude, xcg );
state_dot_trim = zeros( length( state_trim ), 1 );
state_dot_trim(10) = velocity;
disp('  ');
disp( [ ' trimmed angle of attack    ( rad )     = ', num2str( state_trim(2) ) ] );
disp( [ ' trimmed sideslip angle     ( rad )     = ', num2str( state_trim(3) ) ] );
disp('  ');
disp( [ ' trimmed throttle          ( 0-1 )     = ', num2str( control_trim(1) ) ] );
disp( [ ' trimmed elevator           ( deg )     = ', num2str( control_trim(2) ) ] );
disp( [ ' trimmed aileron           ( deg )     = ', num2str( control_trim(3) ) ] );
disp( [ ' trimmed rudder            ( deg )     = ', num2str( control_trim(4) ) ] );
disp('  ');
% **************************state sapce Matrix*****************************
% system state = [ Vt; h; alpha; theta; q; pow; beta; phi; p; r ];
% system control = [ delta_T; delta_E; delta_A; delta_R ];
% system output  = [ an - normal acceleration; q - pitch rate; alpha - angle of attack
%-------------------------------------------------------------
%   The linerization algorithm chooses smaller and      %
%   smaller perturbations in the independent variable   %
%   and compares three successive approximations to     %
%   the particular partial derivative. If they agree    %
%   within a certain tolerance, then the size of the    %
%   perturbation is reduced to determine if an even     %
%   smaller tolerance can be satisfied. The algorithm   %
%   terminates successfully when a tolerance TOLMIN     %
%   is reached.                                         %

x_control  = state_trim  ;
u_control = control_trim ;

%%              1. velocity           ( ft/sec )      %%
%%              2. angle of attack      ( rad )       %%
%%              3. sideslip angle       ( rad )       %%
%%              4. phi   - Euler angle  ( rad )       %%
%%%             5. theta - Euler angle  ( rad )       %%
```

```
%%%              6. psi   - Euler angle  ( rad )         %%
%%               7. roll rate           ( rad/sec )      %%
%%               8. pitch rate          ( rad/sec )      %%
%%               9. yaw rate            ( rad/sec )      %%
%%               10.north displacement   ( ft )          %%
%%               11.east displacement    ( ft )          %%
%%               12.altitude            ( ft )          %%
%%               13.power  ( percent, 0 <= pow <= 100 ) ]

disp('state space matrices');
disp('-------------------------');

disp('system state(x_CONTROL)  = [ Vt; h; alpha; theta; q; pow; beta; phi; p; r ]')
disp('system control(u_control)= [ delta_T; delta_E; delta_A; delta_R ]');
A = ssm_a( state_trim, state_dot_trim, control_trim, xcg );
B = ssm_b( state_trim, state_dot_trim, control_trim, xcg );
C = ssm_c( state_trim, state_dot_trim, control_trim, xcg );
D = ssm_d( state_trim, state_dot_trim, control_trim, xcg );
```

F-16 Dynamics Calculations (non-linear model)

```
% [ x_dot, an, alat, qbar, amach, q, alpha ] = f16_dynam ( time, x, control, xcg )
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                %%
%%%     6-DOF full nonlinear model for F-16 aircraft    %%
%%%     [  rigid-body equations referenced to          %%
%%%        body-fixed axis coordinate system  ]       %%
%%%                                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                        %%
%%% ---- State Variables --------                         %%
%%% x = [1. vt    ( ft/sec )    - velocity                 %%
%%%     2. alpha ( rad )      - angle of attack           %%
%%%     3. beta  ( rad )      - sideslip angle            %%
%%%     4.  phi   ( rad )      - Euler angle               %%
%%%     5. theta ( rad )      - Euler angle               %%
%%%     6. psi   ( rad )      - Euler angle               %%
%%%     7. P     ( rad/sec )   - roll rate                %%
%%%     8. Q     ( rad/sec )   - pitch rate               %%
%%%     9. R     ( rad/sec )   - yaw rate                 %%
%%%     10. integral of north speed    ( ft )  - north displacement      %%
%%%     11. integral of east speed     ( ft )  - east displacement       %%
%%%     12. integral of vertical spped ( ft )  - altitude              %%
%%%     13. pow  ( percent, 0 <= pow <= 100 )   - power ];             %%
%%%                                        %%
%%% ---- control Variables --------                        %%
%%% u = [ thro ( 0 <= thro <= 1.0 ) - throttle              %%
%%%      el   ( deg )           - elevator                 %%
%%%      ail  ( deg )           - aileron                  %%
%%%      rdr  ( deg )           - rudder ];                %%
%%%                                        %%
%%% ---- parameters ---------                              %%
%%% xcg - center of gravity position as                    %%
%%%      fraction of mean aerodynamic chord                %%
%%%                                        %%
%%% ---- output Variables --------                         %%
%%% output = [ 1.(13sub states ) x_dot              - 1st order derivative of state x  %%
%%%          2.            an    ( ft/sec^2 )   - normal acceleration            %%
%%%          3.            alat  ( ft/sec^2 )   - lateral acceleration in y-axis   %%
%%%          4.            qbar  ( psf )       - dynamic pressure               %%
%%%          5.            amach              - mach number                    %%
%%%          6.            q     ( rad/sec )    - pitch rate                   %%
%%%          7.            alpha ( rad )       - angle of attack ];             %%
```

```
%%%                                              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function  [ x_dot, an, alat, qbar, amach, q, alpha ] = f16_dynam ( time, x, control, xcg )

%% ---- constant variable ----------------
s = 300;
b = 30;
cbar = 11.32;
rm = 1.57e-3; % 1 / mass
xcgr = 0.35;
he = 160.0;

%% ---- Inertia constants ----------------
c1 = -0.770;
c2 = 0.02755;
c3 = 1.055e-4;
c4 = 1.642e-6;
c5 = 0.9604;
c6 = 1.759e-2;
c7 = 1.792e-5;
c8 = -0.7336;
c9 = 1.587e-5;

rtod = 180 / pi; %% radians to degrees
g = 32.174;

%% ---- Assign state variables ---------------
vt = x(1);
alpha = x(2) * rtod; %% x(2) in radians, alpha in degrees.
beta = x(3) * rtod;  %% x(3) in radians, beta in degrees.
phi = x(4);
theta = x(5);
psi = x(6);
p = x(7);
q = x(8);
r = x(9);
alt = x(12);
pow = x(13); % power

%% ---- Assign state & control variables ---------------
thro = control(1);
el   = control(2);
ail  = control(3);
rdr  = control(4);

%% ---- Air Data computer and engine model ------------
[tfac, t, rho, amach, qbar, ps ] = atm_std ( vt, alt );
cpow = p_thro (thro);
x_dot (13) = p_dot ( pow, cpow );  %% x_dot(13) = power derivative
t = thrust ( pow, alt, amach );

%% ---- Look-up tables and component biuldup ------------
cxt = c_x ( alpha, el );
cyt = c_y ( beta, ail, rdr );
czt = c_z ( alpha, beta, el );
dail = ail / 20.0;
drdr = rdr / 30.0;
dlda_value = l_a( alpha, beta );
dldr_value = l_r( alpha, beta );
clt = c_l( alpha, beta ) + dlda_value * dail + dldr_value * drdr;
cmt = c_m( alpha, el );
dnda_value = n_a( alpha, beta );
dndr_value = n_r( alpha, beta );
cnt = c_n( alpha, beta ) + dnda_value * dail + dndr_value * drdr;
%% ---- Add damping derivatives ------------
tvt = 0.5 / vt;
```

```
b2v = b * tvt;
cq = cbar * q * tvt;
d = damping( alpha );
cxt = cxt + cq * d(1);
cyt = cyt + b2v * ( d(2) * r + d(3) * p );
czt = czt + cq * d(4);
clt = clt + b2v * ( d(5) * r + d(6) * p );
cmt = cmt + cq * d(7) + czt * ( xcgr - xcg );
cnt = cnt + b2v * ( d(8)* r + d(9) * p ) - cyt * ( xcgr - xcg ) * cbar / b;

%% ---- Get ready for state equations ------------
cbta = cos( x(3) );
u = vt * cos( x(2) ) * cbta;
v = vt * sin( x(3) );
w = vt * sin( x(2) ) * cbta;
sth = sin( theta );
cth = cos( theta );
sph = sin( phi );
cph = cos( phi );
spsi = sin( psi );
cpsi = cos( psi );
qs = qbar * s;
qsb = qs * b;
rmqs = rm *qs;
gcth = g * cth;
qsph = q * sph;
ay = rmqs * cyt;
az = rmqs * czt;

%% ---- Force equations ------------
udot = r * v - q * w - g * sth + rm * ( qs * cxt + t );
vdot = p * w - r * u + gcth * sph + ay;
wdot = q * u - p * v + gcth * cph + az;
dum = ( u * u + w * w );
x_dot(1) = ( u * udot + v * vdot + w * wdot ) / vt;
x_dot(2) = (u * wdot - w * udot ) / dum;
x_dot(3) = (vt * vdot - v * x_dot(1) ) * cbta / dum;

%% ---- Kinematics ------------
x_dot(4) = p + ( sth / cth ) * (qsph + r * cph );
x_dot(5) = q * cph - r * sph;
x_dot(6) = ( qsph + r * cph ) / cth;

%% ---- Moments ------------
x_dot(7) = ( c2 * p + c1 * r + c4 * he ) * q + qsb * ( c3 * clt + c4 * cnt );
x_dot(8) = ( c5 * p -c7 * he )* r + c6 * ( r^2 - p^2 ) + qs * cbar * c7 * cmt;
x_dot(9) = ( c8 * p - c2 * r + c9 * he ) * q + qsb * ( c4 * clt + c9 * cnt );

%% ---- Navigation ------------
t1 = sph * cpsi;
t2 = cph * sth;
t3 = sph * spsi;
s1 = cth * cpsi;
s2 = cth * spsi;
s3 = t1 * sth - cph * spsi;
s4 = t3 * sth + cph * cpsi;
s5 = sph * cth;
s6 = t2 * cpsi + t3;
s7 = t2 * spsi - t1;
s8 = cph * cth;

x_dot(10) = u * s1 + v * s3 + w * s6;  %% north speed
x_dot(11) = u * s2 + v * s4 + w * s7;  %% east speed
x_dot(12) = u * sth - v * s5 - w * s8; %% vertical speed

x_dot = x_dot'; % transfer the vector to a column vector
```

```
%% ---- Outputs ------------
an = (-1) * az / g;
alat = ay / g;
```

References

1. Aircraft Flight Dynamics and Control Book by Wayne Durham
2. Flight Dynamics Principles Book by M. V. Cook
3. Lecture notes by Prof. Youmin Zhang
4. http://www.wright-brothers.org/Information_Desk/Just_the_Facts/Airplanes/Flyer_I.htm
5. http://www.faa-aircraft-certification.com/faa-definitions.html
6. http://www.mcclatchydc.com/news/nation-world/national/article24727069.html
7. http://www.mcclatchydc.com/news/nation-world/national/article24724569.html
8. New Aircraft II Color
9. Hillaker, Harry J. "Technology and the F-16 Fighting Falcon Jet Fighter." nae.edu. Retrieved: 25 October 2009
10. Non-Linear F-16 model by Richard S. Russel
11. Introduction to Aircraft Stability and Control Course Notes for M&AE 5070 David A. Caughey
12. https://www.lockheedmartin.com/us/products/f16.html
13. Lockheed Martin F-16 Fighting Falcon." Jane's All the World's Aircraft, updated 21 January 2008. Retrieved: 30 May 2008