Simple Longitudinal Autopilot System

J-component report
ECE2010 - Control Systems

Adit V Kirtani - 19BEC0465
Vishwanath K - 19BEC0034
Samanyu Okade - 19BEC0451

Submitted to : Prof. Bagubali A

Index

1. Introduction...3

2. LiteratureReview..5

3. Objectives, Project Description and Goals, Tools used........8

4. Theory...9

5. Calculations...12

6. MATLAB Code...13

7. Results..15

8. Mathematical formulas used in the model...........................17

9. Simulink Model...19

10. Custom MATLAB function block.......................................21

11. Simulation screenshots...23

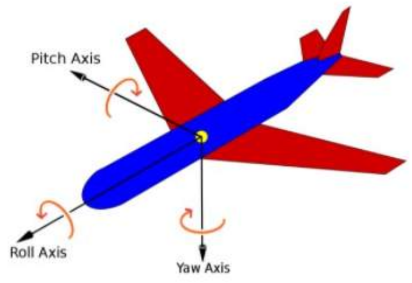
12. References..24

13. Conclusions...25

Introduction

An autopilot is a device or control system that allows an aircraft, a marine craft, or a spacecraft to navigate without the need for constant manual control by a human operator. Human operators are not replaced by autopilots. Instead, the autopilot aids the driver's management of the vehicle, allowing the driver to concentrate on more important areas of the job. The design of an autopilot is determined by the aircraft's control surfaces.

The longitudinal autopilot controls the pitch and altitude of the aircraft, while the lateral autopilot governs roll and yaw movements.We have restricted our model to longitudinal autopilot.



The pitch of an aircraft refers to the rotation of the aircraft around a side to side axis [1],[4]. It can be thought of as the "up and down" or "nodding" motion. Control of pitch is what most clearly differentiates operating an aircraft in the sky from any Earthbound vehicle. The image alongside clearly shows the various axes of motion control in an aircraft. In the case of longitudinal modes, pitch angle and altitude depend only on the elevator's contribution. We will control the elevator and ailerons of the aircraft to control its motion as a whole.

Autopilot is now a topic of great interest in research because of its newly increasing applications[2] in aircraft, missiles, space crafts, ships, UAVs and others. It has been evident that autopilot is essential to control air vehicles automatically and reduce human beings' workload.

This report is based on developing a flexible environment for aircraft dynamics and control analysis for the DHC-2 'Beaver' aircraft. This environment uses the power and flexibility of the simulation and system analysis programs SIMULINK and MATLAB[3].

An understanding of the dynamic characteristics of an aeroplane is vital in assessing its handling or flying qualities and designing autopilots. Static stability is a tendency of the aircraft to return to its equilibrium position. In addition to static stability, the aircraft also must be dynamically stable. An aeroplane can be considered dynamically stable if the ensuing motion diminishes with time after being disturbed from its equilibrium flight condition.

Literature Review

Sl. No.	Title	Year and Publication	Proposed Concept
1	A Low-Cost Altitude Control System for the Kadet Senior Radio-Controlled Airplane	2003, by Angel Abusleme in IEEE Transactions on Education	Application and tuning of 3 PID controllers to control the climb rate, altitude and longitudinal dynamics of a Kadet Senior model fitted with appropriate sensors for altitude sensing.
2	X-Plane-Simulink Simulation of a Pitch-Holding Automatic Control System for Boeing 747	2007, by Yazdi I. Jenie	This paper talks about simulating a pitch-hold control system on the Boeing 747 model using the classical method of linearized State Space model.
3	Self-tuning Fuzzy PID Controller Design for Aircraft Pitch Control	2012, by Nurbaiti Wahid in the Third International Conference on Intelligent Systems Modelling and Simulation	Design of a PID and instilling a self-tuning ability in it using fuzzy logic primarily for longitudinal dynamics, or, pitch.
4	Guidance Software-in-The-Loop Simulation Using X-Plane and Simulink for UAVs	2014, by Adriano Bittar in the 2014 International Conference on Unmanned Aircraft Systems (ICUAS)	This paper proposes an improved guidance algorithm using waypoint switching and reference generation to be able to autonomously find the most optimized path to the goal in the SITL.
5	State Space Model of an Aircraft Using Simulink	2017, by Lubna Moin in the International Journal of System Modeling and Simulation	The proposed model considers pitch rate and pitch angle for improved stability using PD controller.

Sl. No.	Title	Year and Publication	Proposed Concept
6	Imitation Learning for Neural Network Autopilot in Fixed-Wing Unmanned Aerial Systems	2020, by Daksh Shukla in the 2020 International Conference on Unmanned Aircraft Systems (ICUAS)	Here, an Artificial Neural Network (ANN) autopilot is trained along with supervised learning techniques that have an imitation learning framework called the data aggregation set (DAgger) algorithm to fly fixed-wing aircrafts.
7	Designing, Simulation and Control of Autopilot using PID Controller	2021, by Htet Soe Paing in the 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)	This paper represents the design and simulation of an aircraft's pitch motion using PID controller in an outdoor environment.
8	A Neural Autopilot Training Platform based on a Matlab and X-Plane co-simulation	2021, by Jérémy Pinguet in the 2021 International Conference on Unmanned Aircraft Systems (ICUAS)	The method proposed in this paper implements Imitation learning based Neural Network that learns by storing piloting skills as datasets and using the database to train the model.

Objectives

- A MATLAB script simulating the performance of the aircraft at the given conditions.

- A Simulink model linked with FlightGear to visualize the aircraft's performance in real-time – with the ability to change the pilot's command and fly/control the aircraft.

Project Description and Goals

The project has been completely modelled using Simulink from MATLAB. The simulation part has been done using FlightGear. The required curves have also been analysed for further analysis, visualisation and evaluation of the model's performance. We have successfully:

- Implemented the control system using Simulink and plotted the required data.
- Simulated the model in realtime using FlightGear.
- Understood the importance of Autopilot in keep the aircraft dynamically stable.

Tools Used

The following softwares were used to implement and simulate the project:
1. MATLAB
2. SIMULINK
3. FlightGear

Theory

Figure (a) shows the two sets of coordinate systems used, the body axis system attached to the aircraft and the inertial axis system fixed to the Earth.

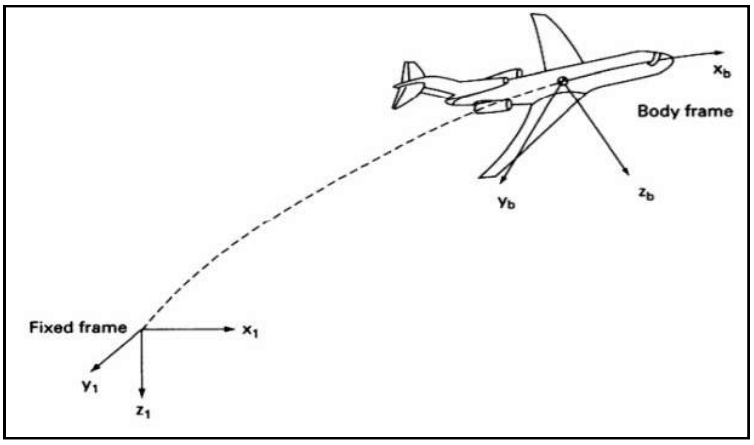


Figure (a)

The rigid body motion equations, derived from Newton's second law, are used to explain the motion of the aircraft:

$$\boxed{\begin{gathered}\Sigma F = \frac{d}{dt}(mV) \\ \Sigma M = \frac{d}{dt}H\end{gathered}}$$

These are vector equations, with Fx, Fy, Fz, and u, v, w denoting force and velocity components along the x, y, and z axes, respectively.
The force components are contributions from the aeroplane's aerodynamic, propulsive, and gravitational forces.

The moment and moment of momentum components along the x, y, and z axes are L, M, N and Hx, Hy, Hz, respectively.

The orientation of the airplane can be described by three consecutive rotations, whose order is important. The angular rotations are called the Euler angles.

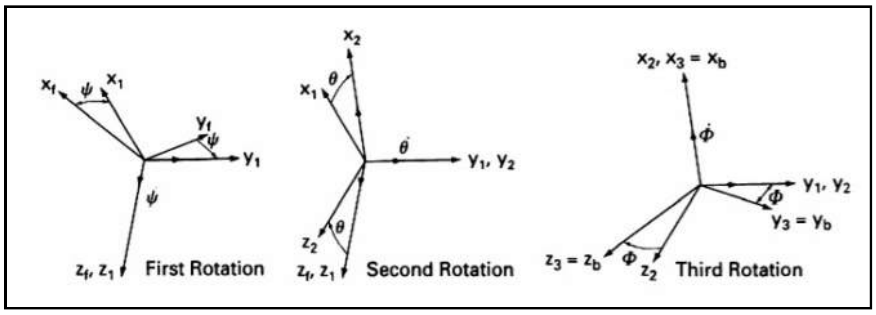


Figure (b) Euler Angles

It can be proven that, in addition to the Euler equations, these fundamental newton equations may be rearranged to yield the twelve governing equations of aircraft motion as follows:

$$L = I_x \dot{p}_i - I_{xz}\dot{r} + qr(I_z - I_y) - I_{xz}pq$$

$$\nwarrow \; M = I_y \dot{q} + qr(I_x - I_z) + I_{xz}(p^2 - r^2)$$

$$N = I_{xz}\dot{p}_i + I_z \dot{r} + qp(I_y - I_z) - I_{xz}rq$$

$$F_x - mg\sin(\theta) = m(\dot{u} + qw - rv)$$

$$F_y + mg\cos(\theta)\sin(\phi) = m(\dot{v} + ru - pw)$$

$$F_z + mg\cos(\theta)\cos(\phi) = m(\dot{w} + pv - qu)$$

Euler Angles

$$\begin{bmatrix}\dot{\Phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}=\begin{bmatrix}1 & S_{\Phi}\tan\theta & C_{\Phi}\tan\theta\\ 0 & C_{\Phi} & -S_{\Phi}\\ 0 & S_{\Phi}\sec\theta & C_{\Phi}\sec\theta\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix}$$

angular velocities in the body

Absolute Velocity Components along the fixed frame

Velocity Components along the body axes

$$\begin{bmatrix}\frac{dx}{dt}\\ \frac{dy}{dt}\\ \frac{dz}{dt}\end{bmatrix}=\begin{bmatrix}C_\theta C_\psi & S_\Phi S_\theta C_\psi - C_\Phi S_\psi & C_\Phi S_\theta C_\psi + S_\Phi S_\psi\\ C_\theta S_\psi & S_\Phi S_\theta S_\psi + C_\Phi C_\psi & C_\Phi S_\theta S_\psi - S_\Phi C_\psi\\ -S_\theta & S_\theta C_\theta & C_\Phi C_\theta\end{bmatrix}\begin{bmatrix}u\\ v\\ w\end{bmatrix}$$

Our aircraft state vector at the trimming point is stated in the following table:

state	value
V	35
α	0.211
β	-0.0206
p	0
q	0
r	0
ϕ	0.191
θ	0
ψ	0

state	value
x	0
y	0
z	0
δe	-0.093
δa	0.009624
δr	-0.0495
δf	0
n	1800
p_z	20

Where the velocity V in m/s, while all the angles are in radians.

The most significant aspects of the DHC-2 "Beaver" aircraft model will be implemented using the MATLAB code described in this report: the aerodynamic model, engine model, atmosphere, and air data model. To obtain the aircraft states and their divergence from the trimmed point over time, the non-linear state equations will be numerically solved using the Runge-Kutta method for solving ordinary differential equations[3]. We'll go over the steps we followed to solve the Equations using computations. The project's sequence is depicted in the flow chart below.

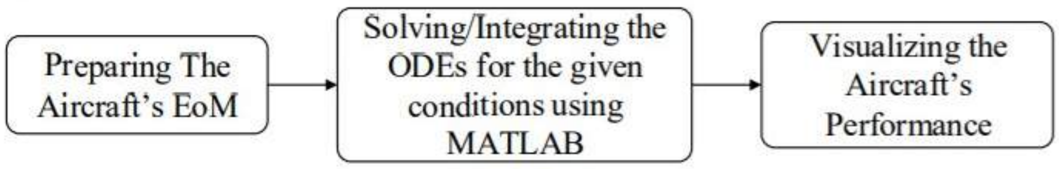

Preparing The Aircraft's EoM
Solving/Integrating the ODEs for the given conditions using MATLAB
Visualizing the Aircraft's Performance

Figure (c)

Calculations

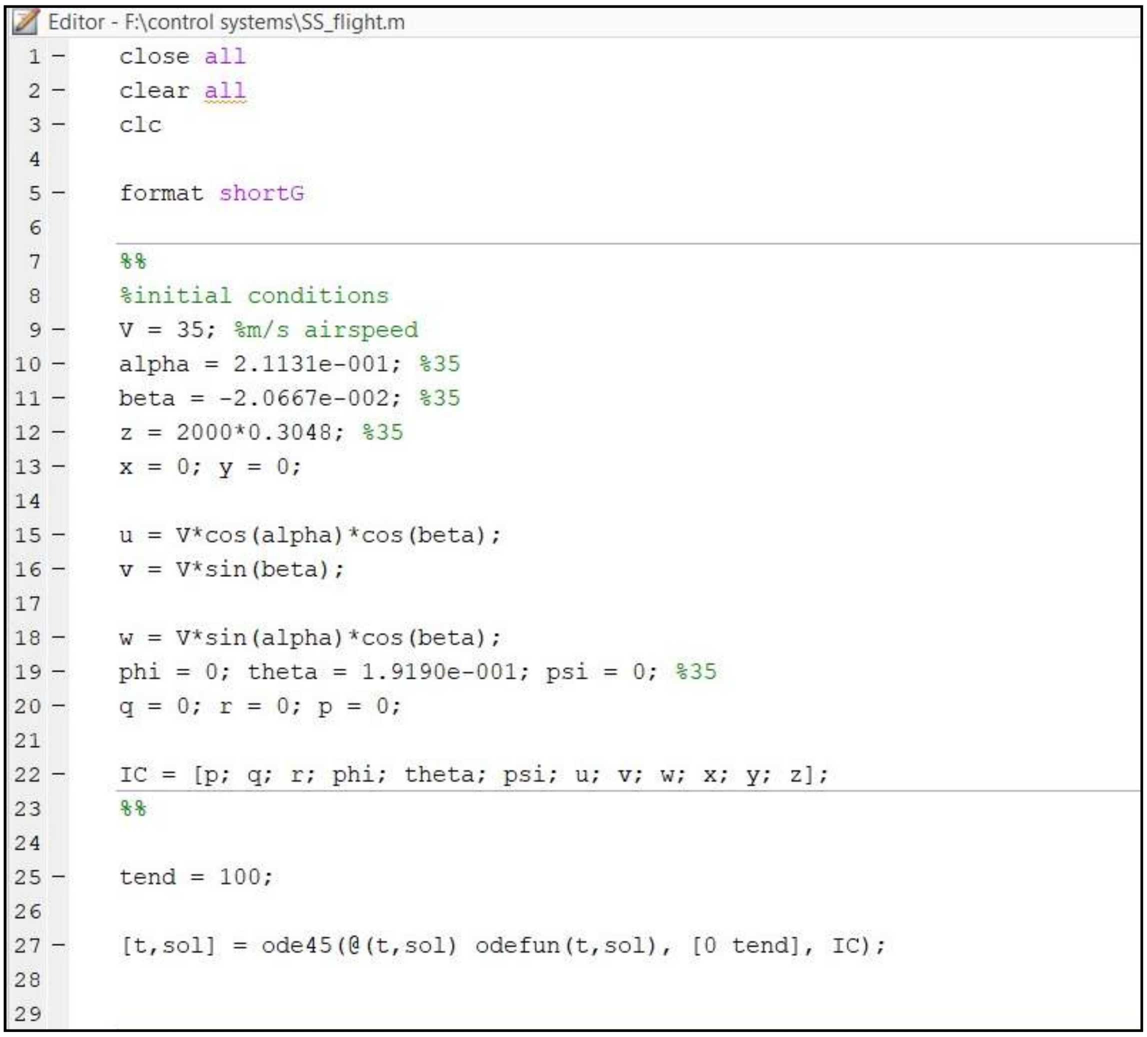
In order to solve the twelve ordinary differential equations using the Rung-Kutta method[5] in MATLAB by the “ode45” function, it is required that the time rate of each variable be explicitly defined, which means that we need to decouple any coupled equations to get each variable’s time rate defined in terms of the other
variables.
To do this, we used the MALTAB Symbolic Toolbox and solved the coupled $L \ and \ N$ moment equations for $\dot{p} \ and \ \dot{r}$, and we obtained the following decoupled equations.

$$\boxed{\begin{aligned}
\dot{p} &= \frac{I_z L + I_{xz} N + I_{xz}^2 qr - I_z^2 qr - I_{xz} I_y pq + 2 I_{xz} I_z pq + I_y I_z qr}{I_{xz}^2 + I_x I_z} \\
\dot{r} &= -\frac{I_{xz} L - I_x N + I_{xz}^2 pq + I_x I_y pq - I_x I_z pq - I_x I_{xz} qr + I_{xz} I_y qr - I_{xz} I_z qr}{I_{xz}^2 + I_x I_z}
\end{aligned}}$$

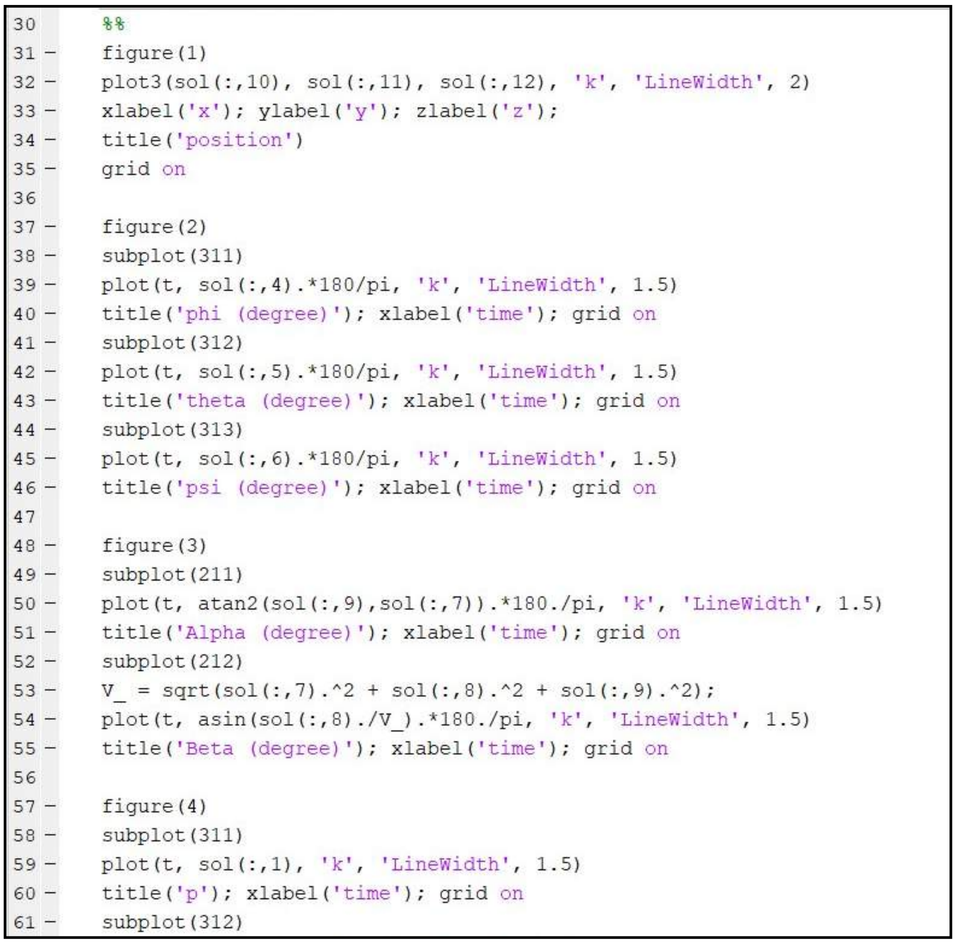
We chose the ode45 solver after analysing the various ODE solvers in MATLAB because it is based on an explicit Runge-Kutta formula. It is a one-step solver (in computing $y(tn)$, it only requires the solution at the previous time point $y(tn - 1)$.

MATLAB Code

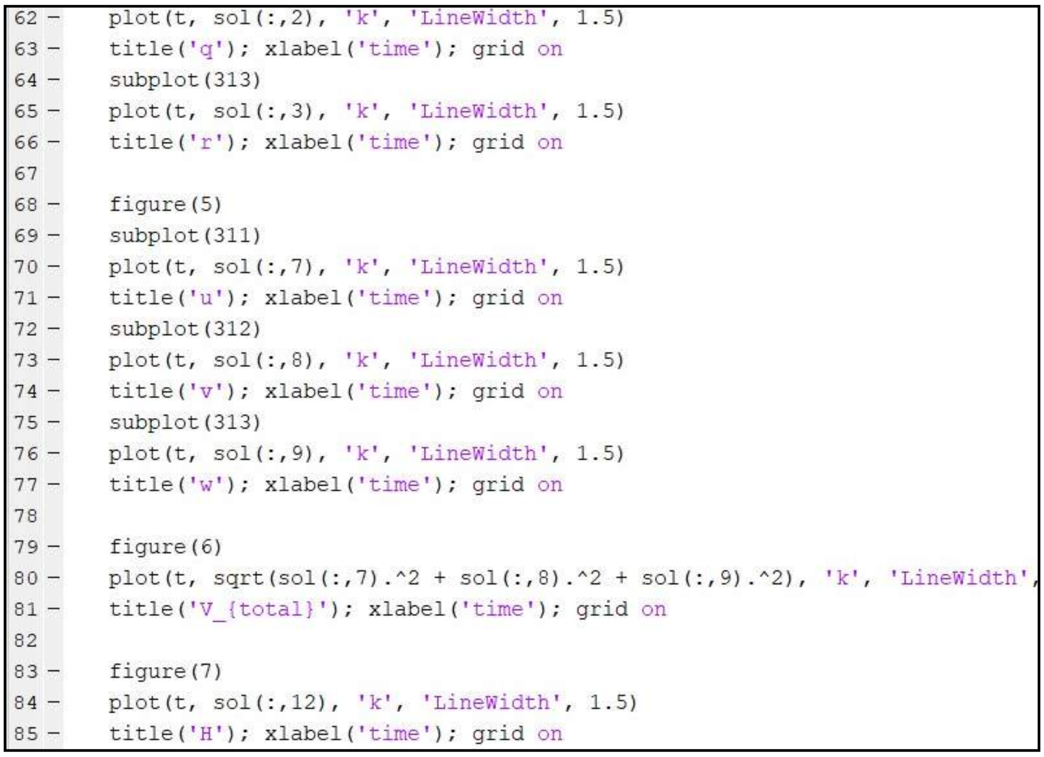
Solving the 12 non-linear ordinary differential equations at the given trim conditions for a time span of 100 seconds.



```
close all
clear all
clc

format shortG

%%
%initial conditions
V = 35; %m/s airspeed
alpha = 2.1131e-001; %35
beta = -2.0667e-002; %35
z = 2000*0.3048; %35
x = 0; y = 0;

u = V*cos(alpha)*cos(beta);
v = V*sin(beta);

w = V*sin(alpha)*cos(beta);
phi = 0; theta = 1.9190e-001; psi = 0; %35
q = 0; r = 0; p = 0;

IC = [p; q; r; phi; theta; psi; u; v; w; x; y; z];
%%

tend = 100;

[t,sol] = ode45(@(t,sol) odefun(t,sol), [0 tend], IC);
```

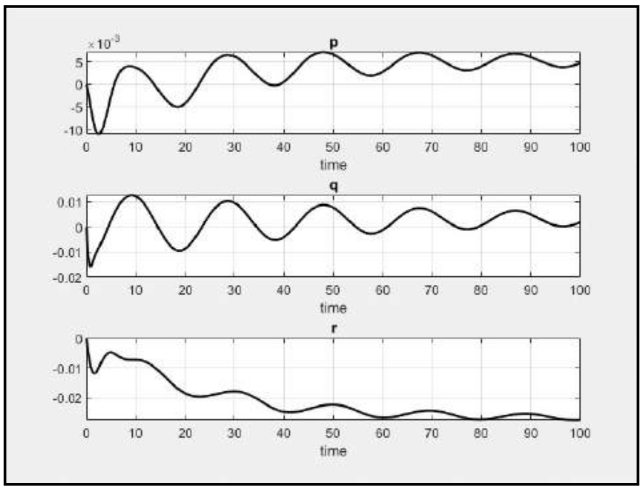


```matlab
%%
figure(1)
plot3(sol(:,10), sol(:,11), sol(:,12), 'k', 'LineWidth', 2)
xlabel('x'); ylabel('y'); zlabel('z');
title('position')
grid on

figure(2)
subplot(311)
plot(t, sol(:,4).*180/pi, 'k', 'LineWidth', 1.5)
title('phi (degree)'); xlabel('time'); grid on
subplot(312)
plot(t, sol(:,5).*180/pi, 'k', 'LineWidth', 1.5)
title('theta (degree)'); xlabel('time'); grid on
subplot(313)
plot(t, sol(:,6).*180/pi, 'k', 'LineWidth', 1.5)
title('psi (degree)'); xlabel('time'); grid on

figure(3)
subplot(211)
plot(t, atan2(sol(:,9),sol(:,7)).*180./pi, 'k', 'LineWidth', 1.5)
title('Alpha (degree)'); xlabel('time'); grid on
subplot(212)
V_ = sqrt(sol(:,7).^2 + sol(:,8).^2 + sol(:,9).^2);
plot(t, asin(sol(:,8)./V_).*180./pi, 'k', 'LineWidth', 1.5)
title('Beta (degree)'); xlabel('time'); grid on

figure(4)
subplot(311)
plot(t, sol(:,1), 'k', 'LineWidth', 1.5)
title('p'); xlabel('time'); grid on
subplot(312)
```

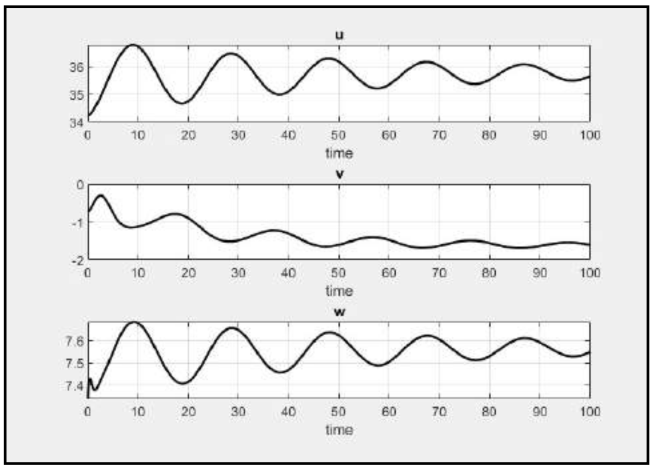


```matlab
plot(t, sol(:,2), 'k', 'LineWidth', 1.5)
title('q'); xlabel('time'); grid on
subplot(313)
plot(t, sol(:,3), 'k', 'LineWidth', 1.5)
title('r'); xlabel('time'); grid on

figure(5)
subplot(311)
plot(t, sol(:,7), 'k', 'LineWidth', 1.5)
title('u'); xlabel('time'); grid on
subplot(312)
plot(t, sol(:,8), 'k', 'LineWidth', 1.5)
title('v'); xlabel('time'); grid on
subplot(313)
plot(t, sol(:,9), 'k', 'LineWidth', 1.5)
title('w'); xlabel('time'); grid on

figure(6)
plot(t, sqrt(sol(:,7).^2 + sol(:,8).^2 + sol(:,9).^2), 'k', 'LineWidth',
title('V_{total}'); xlabel('time'); grid on

figure(7)
plot(t, sol(:,12), 'k', 'LineWidth', 1.5)
title('H'); xlabel('time'); grid on
```

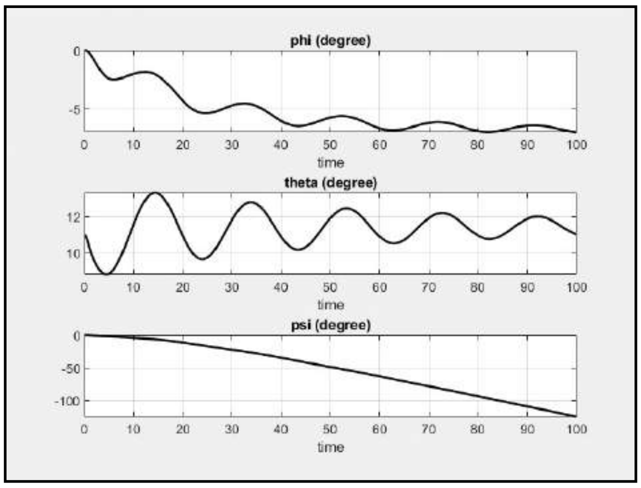
Results



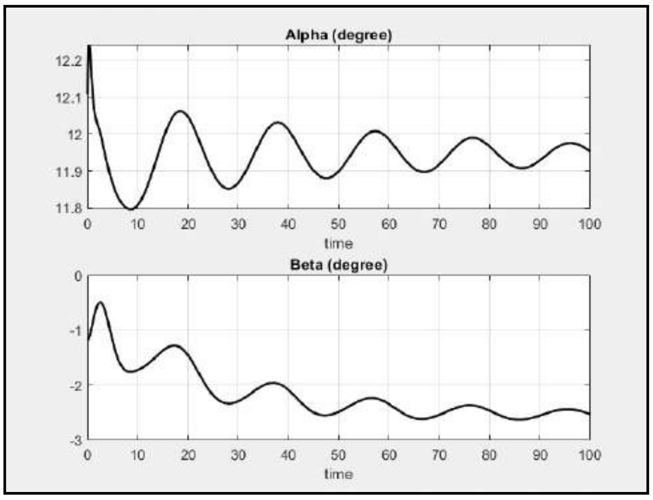
p
q
r
time



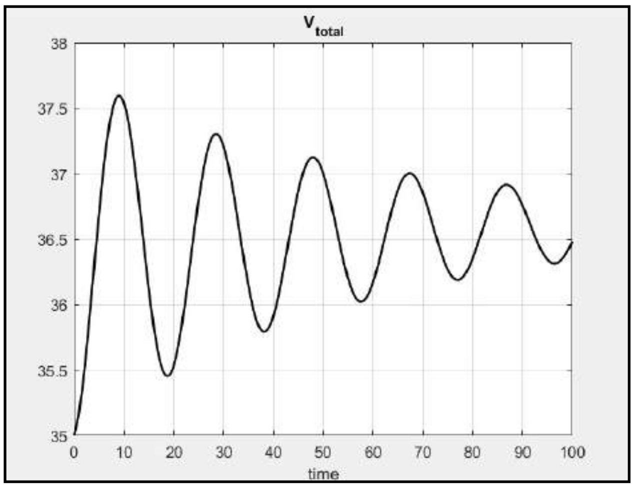
u
v
w
time



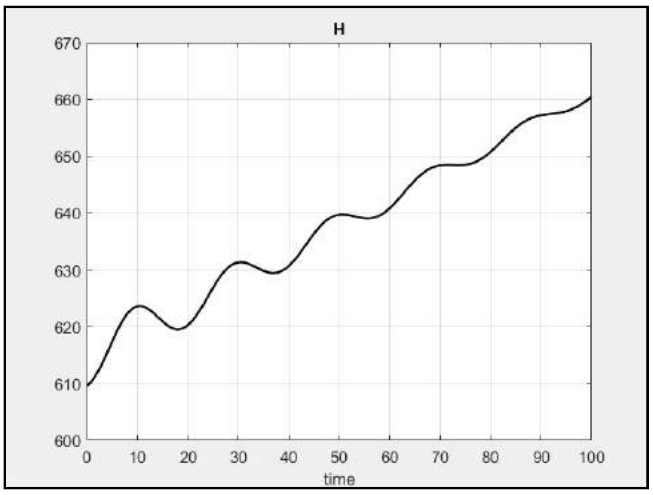
phi (degree)
theta (degree)
psi (degree)
time



Alpha (degree)
Beta (degree)
time

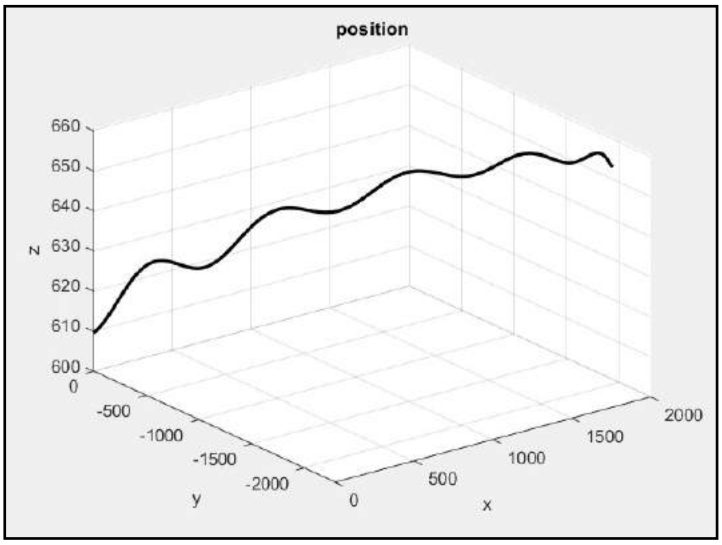


V_total
time



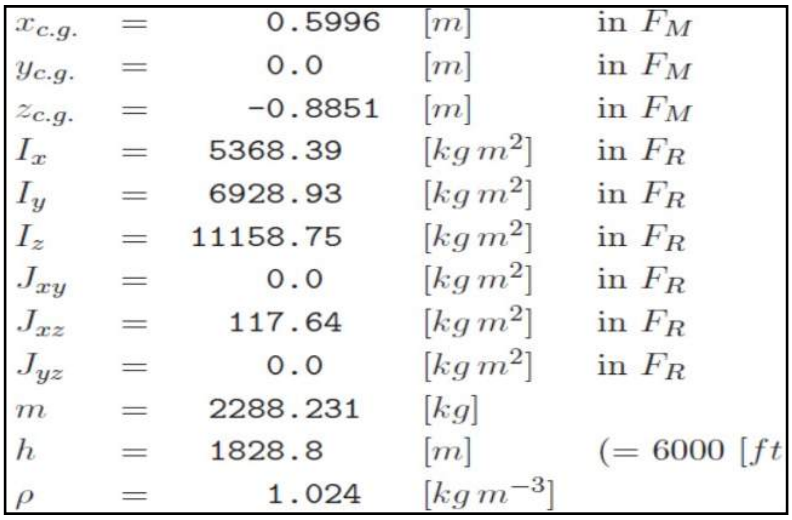
H
time

The aircraft appears to be oscillating about its trim state, and the results appear to be reasonable. The order of magnitude of the velocities is as expected. As for our case, the velocity component must have the most significant value, while the y-component has the lowest value change with time. In addition, the $p\ q,\ and\ r$ values appear to be oscillating about zero, which is expected during the trim state. The open-loop response of the position state in the three directions is obtained by integrating the velocity components and plotting to show the aircraft motion during the time. The aircraft's position plot clearly suggests that the results are valid, as it shows a typical aircraft's motion for a trim state.

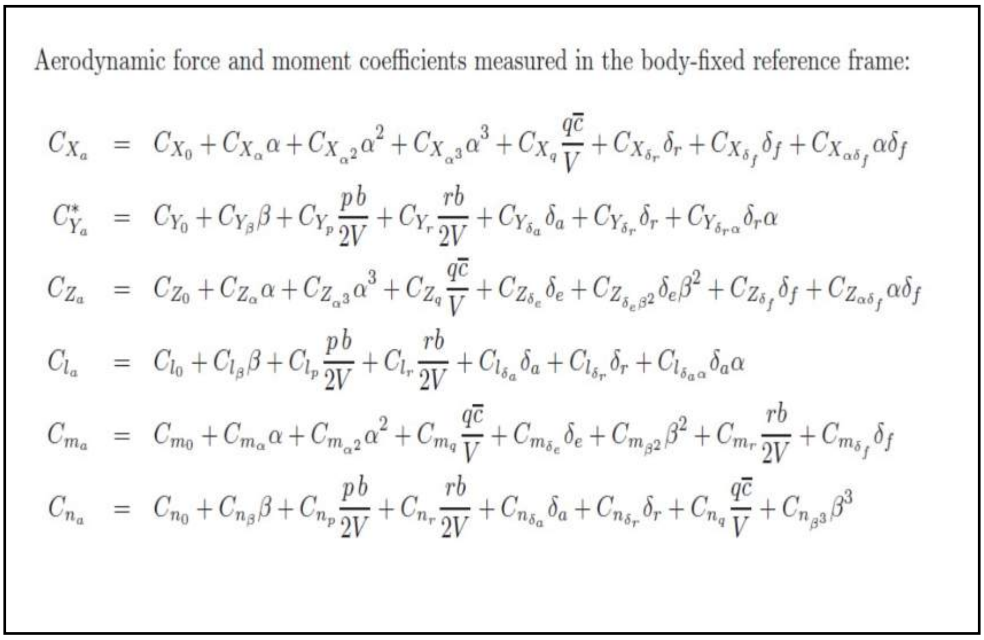


Mathematical formulas used in the model

Data for DHC-2 "Beaver" aircraft:



$x_{c.g.}$	=	0.5996	$[m]$	in F_M
$y_{c.g.}$	=	0.0	$[m]$	in F_M
$z_{c.g.}$	=	-0.8851	$[m]$	in F_M
I_x	=	5368.39	$[kg\,m^2]$	in F_R
I_y	=	6928.93	$[kg\,m^2]$	in F_R
I_z	=	11158.75	$[kg\,m^2]$	in F_R
J_{xy}	=	0.0	$[kg\,m^2]$	in F_R
J_{xz}	=	117.64	$[kg\,m^2]$	in F_R
J_{yz}	=	0.0	$[kg\,m^2]$	in F_R
m	=	2288.231	$[kg]$	
h	=	1828.8	$[m]$	$(= 6000\ [ft$
ρ	=	1.024	$[kg\,m^{-3}]$	



Aerodynamic force and moment coefficients measured in the body-fixed reference frame:

$$
\begin{aligned}
C_{X_a} &= C_{X_0} + C_{X_\alpha}\alpha + C_{X_{\alpha^2}}\alpha^2 + C_{X_{\alpha^3}}\alpha^3 + C_{X_q}\frac{q\bar{c}}{V} + C_{X_{\delta_r}}\delta_r + C_{X_{\delta_f}}\delta_f + C_{X_{\alpha\delta_f}}\alpha\delta_f \\
C^*_{Y_a} &= C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{pb}{2V} + C_{Y_r}\frac{rb}{2V} + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r + C_{Y_{\delta_r\alpha}}\delta_r\alpha \\
C_{Z_a} &= C_{Z_0} + C_{Z_\alpha}\alpha + C_{Z_{\alpha^3}}\alpha^3 + C_{Z_q}\frac{q\bar{c}}{V} + C_{Z_{\delta_e}}\delta_e + C_{Z_{\delta_e\beta^2}}\delta_e\beta^2 + C_{Z_{\delta_f}}\delta_f + C_{Z_{\alpha\delta_f}}\alpha\delta_f \\
C_{l_a} &= C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{pb}{2V} + C_{l_r}\frac{rb}{2V} + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r + C_{l_{\delta_a\alpha}}\delta_a\alpha \\
C_{m_a} &= C_{m_0} + C_{m_\alpha}\alpha + C_{m_{\alpha^2}}\alpha^2 + C_{m_q}\frac{q\bar{c}}{V} + C_{m_{\delta_e}}\delta_e + C_{m_{\beta^2}}\beta^2 + C_{m_r}\frac{rb}{2V} + C_{m_{\delta_f}}\delta_f \\
C_{n_a} &= C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{pb}{2V} + C_{n_r}\frac{rb}{2V} + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r + C_{n_q}\frac{q\bar{c}}{V} + C_{n_{\beta^3}}\beta^3
\end{aligned}
$$

C_X		C_Y		C_Z	
parameter	value	parameter	value	parameter	value
0	-0.03554	0	-0.002226	0	-0.05504
α	0.002920	β	-0.7678	α	-5.578
α^2	5.459	$\frac{pb}{2V}$	-0.1240	α^3	3.442
α^3	-5.162	$\frac{rb}{2V}$	0.3666	$\frac{q\bar{c}}{V}$	-2.988
$\frac{q\bar{c}}{V}$	-0.6748	δ_a	-0.02956	δ_e	-0.3980
δ_r	0.03412	δ_r	0.1158	$\delta_e \beta^2$	-15.93
δ_f	-0.09447	$\delta_r \alpha$	0.5238	δ_f	-1.377
$\alpha \delta_f$	1.106	$\frac{\dot{\beta} b}{2V}$	-0.1600	$\alpha \delta_f$	-1.261

C_l		C_m		C_n	
parameter	value	parameter	value	parameter	value
0	0.0005910	0	0.09448	0	-0.003117
β	-0.06180	α	-0.6028	β	0.006719
$\frac{pb}{2V}$	-0.5045	α^2	-2.140	$\frac{pb}{2V}$	-0.1585
$\frac{rb}{2V}$	0.1695	$\frac{q\bar{c}}{V}$	-15.56	$\frac{rb}{2V}$	-0.1112
δ_a	-0.09917	δ_e	-1.921	δ_a	-0.003872
δ_r	0.006934	β^2	0.6921	δ_r	-0.08265
$\delta_a \alpha$	-0.08269	$\frac{rb}{2V}$	-0.3118	$\frac{q\bar{c}}{V}$	0.1595
		δ_f	0.4072	β^3	0.1373

C_X		C_Y		C_Z	
parameter	value	parameter	value	parameter	value
dpt	0.1161	–	–	dpt	-0.1563
$\alpha \cdot dpt^2$	0.1453				

C_l		C_m		C_n	
parameter	value	parameter	value	parameter	value
$\alpha^2 \cdot dpt$	-0.01406	dpt	-0.07895	dpt^3	-0.003026

$$dpt \equiv \frac{\Delta p_t}{\frac{1}{2}\rho V^2} = C_1 + C_2 \frac{P}{\frac{1}{2}\rho V^3} \quad \text{with:} \quad \begin{cases} C_1 &= 0.08696 \\ C_2 &= 191.18 \end{cases}$$

- Engine power P, $[N m\, s^{-1}]$:

$$P = 0.7355 \left\{ -326.5 + \left(0.00412(p_z + 7.4)(n + 2010) + (408.0 - 0.0965n)\left(1.0 - \frac{\rho}{\rho_0}\right)\right)\right\}$$

Equations

- Dimensional forces, $[N]$:

$$\begin{aligned} X_a &= C_{X_a}\, q_{dyn}\, S \\ Y_a &= C_{Y_a}\, q_{dyn}\, S \\ Z_a &= C_{Z_a}\, q_{dyn}\, S \end{aligned}$$

- Dimensional moments, $[Nm]$:

$$\begin{aligned} L_a &= C_{l_a}\, q_{dyn}\, S\, b \\ M_a &= C_{m_a}\, q_{dyn}\, S\, \overline{c} \\ N_a &= C_{n_a}\, q_{dyn}\, S\, b \end{aligned}$$

ISA Atmospheric Model

For the atmospheric data an approximation of the International Standard Atmosphere (ISA) is used.

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

where To = 288.15 is the temperature at sea level and po = 1.225 is the air density at sea level. This atmospheric model is only valid in the troposphere (h < 11000 m). Given the aircraft's altitude (h in meters) it returns the current temperature (T in Kelvin), the current air density (p in kg/m) and the speed of sound (a in m/s).

Simulink Model

The Simulink model is based on the MATLAB function, which calculates the state variables- where we used a simple “integrator” block to calculate the variables in real-time during simulation. Finally, the Simulink model is integrated with the FlightGear simulator to visualize the aircraft performance in real-time.

The following flowchart shows the main blocks/subsystems of our Simulink model.

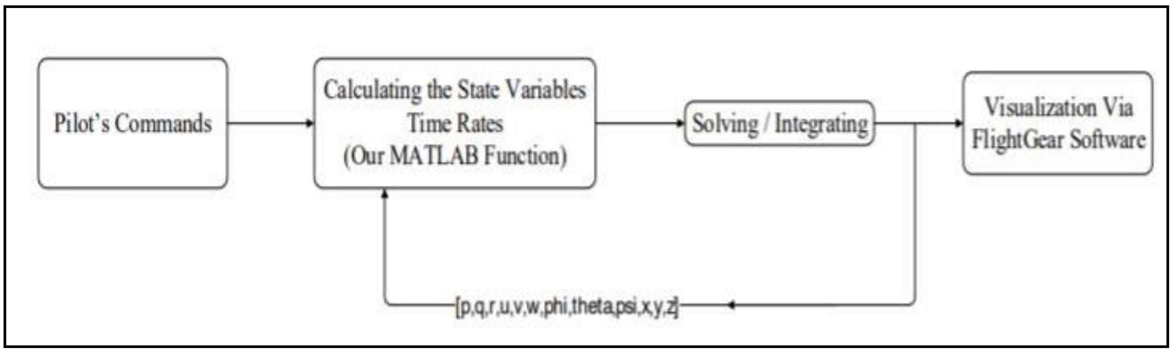

Pilot's Commands
Calculating the State Variables Time Rates (Our MATLAB Function)
Solving / Integrating
Visualization Via FlightGear Software
[p,q,r,u,v,w,phi,theta,psi,x,y,z]

Figure (d)

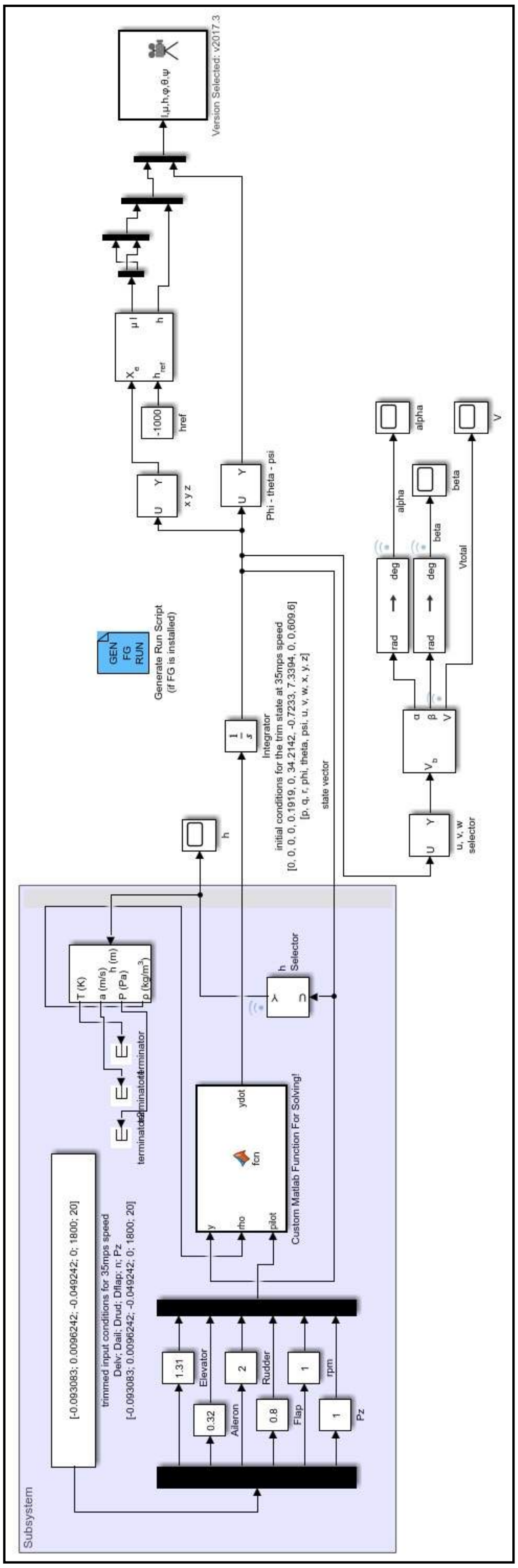
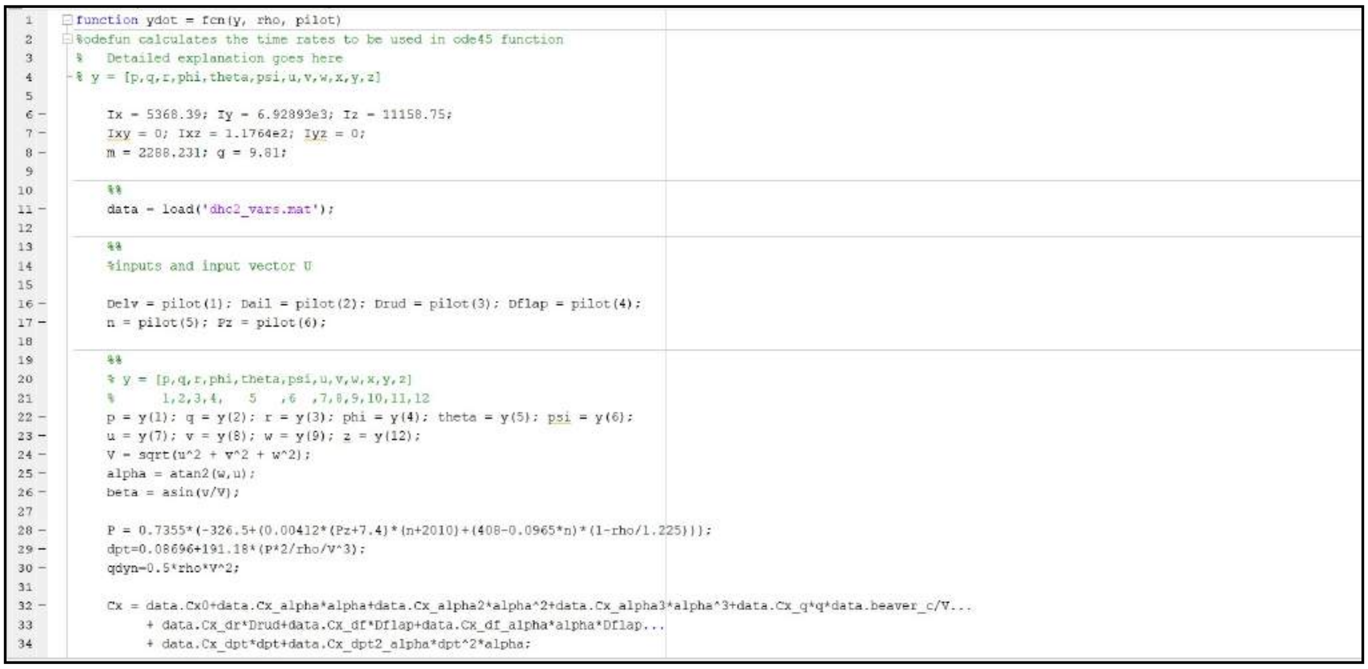
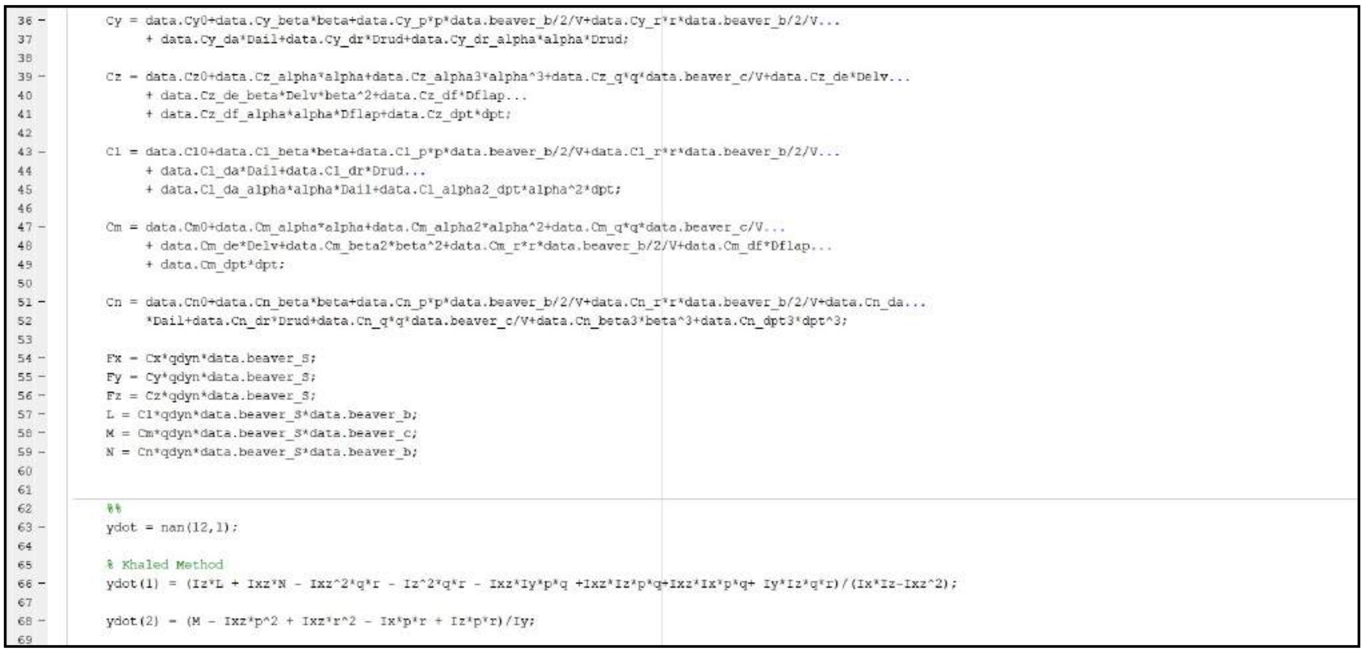


Figure (e) Simulink Model

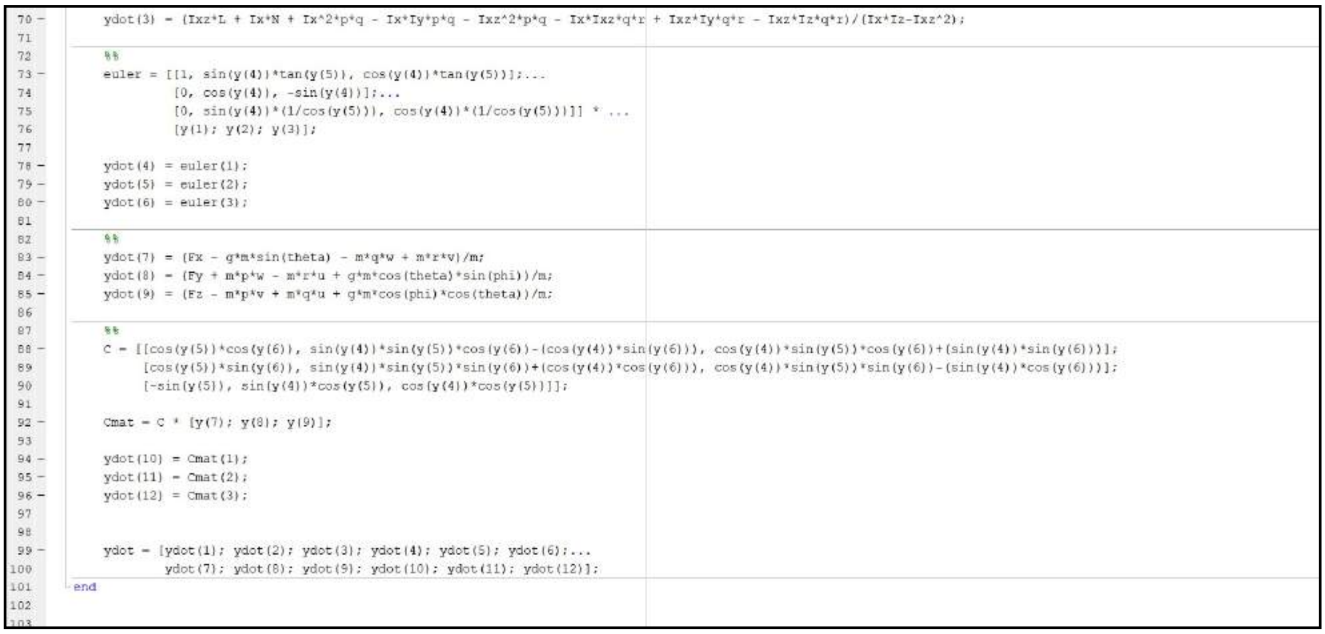
Custom MATLAB function block



```matlab
function ydot = fcn(y, rho, pilot)
%odefun calculates the time rates to be used in ode45 function
%   Detailed explanation goes here
% y = [p,q,r,phi,theta,psi,u,v,w,x,y,z]

    Ix = 5368.39; Iy = 6.92893e3; Iz = 11158.75;
    Ixy = 0; Ixz = 1.1764e2; Iyz = 0;
    m = 2288.231; g = 9.81;

    %%
    data = load('dhc2_vars.mat');

    %%
    %inputs and input vector U

    Delv = pilot(1); Dail = pilot(2); Drud = pilot(3); Dflap = pilot(4);
    n = pilot(5); Pz = pilot(6);

    %%
    % y = [p,q,r,phi,theta,psi,u,v,w,x,y,z]
    %      1,2,3,4,   5   ,6  ,7,8,9,10,11,12
    p = y(1); q = y(2); r = y(3); phi = y(4); theta = y(5); psi = y(6);
    u = y(7); v = y(8); w = y(9); z = y(12);
    V = sqrt(u^2 + v^2 + w^2);
    alpha = atan2(w,u);
    beta = asin(v/V);

    P = 0.7355*(-326.5+(0.00412*(Pz+7.4)*(n+2010)+(408-0.0965*n)*(1-rho/1.225)));
    dpt=0.08696+191.18*(P*2/rho/V^3);
    qdyn=0.5*rho*V^2;

    Cx = data.Cx0+data.Cx_alpha*alpha+data.Cx_alpha2*alpha^2+data.Cx_alpha3*alpha^3+data.Cx_q*q*data.beaver_c/V...
         + data.Cx_dr*Drud+data.Cx_df*Dflap+data.Cx_df_alpha*alpha*Dflap...
         + data.Cx_dpt*dpt+data.Cx_dpt2_alpha*dpt^2*alpha;
```

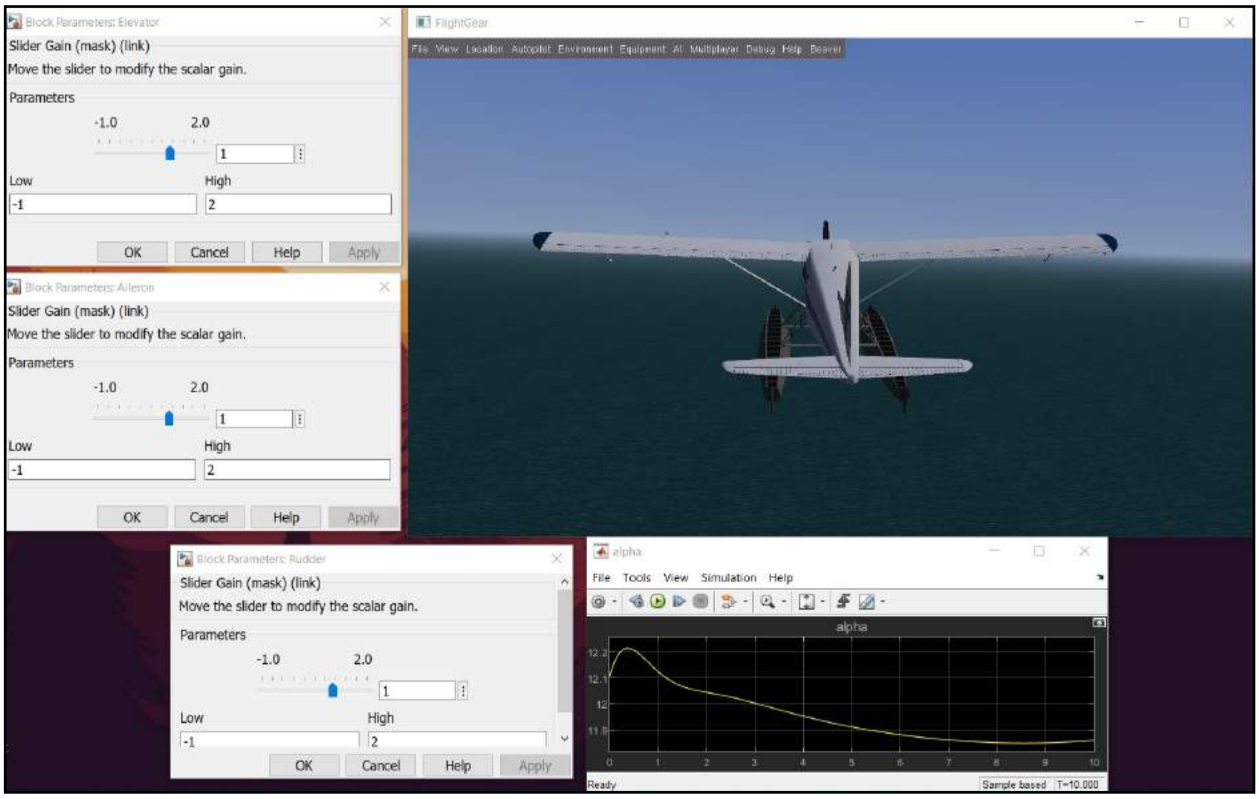


```matlab
    Cy = data.Cy0+data.Cy_beta*beta+data.Cy_p*p*data.beaver_b/2/V+data.Cy_r*r*data.beaver_b/2/V...
         + data.Cy_da*Dail+data.Cy_dr*Drud+data.Cy_dr_alpha*alpha*Drud;

    Cz = data.Cz0+data.Cz_alpha*alpha+data.Cz_alpha3*alpha^3+data.Cz_q*q*data.beaver_c/V+data.Cz_de*Delv...
         + data.Cz_de_beta*Delv*beta^2+data.Cz_df*Dflap...
         + data.Cz_df_alpha*alpha*Dflap+data.Cz_dpt*dpt;

    Cl = data.Cl0+data.Cl_beta*beta+data.Cl_p*p*data.beaver_b/2/V+data.Cl_r*r*data.beaver_b/2/V...
         + data.Cl_da*Dail+data.Cl_dr*Drud...
         + data.Cl_da_alpha*alpha*Dail+data.Cl_alpha2_dpt*alpha^2*dpt;

    Cm = data.Cm0+data.Cm_alpha*alpha+data.Cm_alpha2*alpha^2+data.Cm_q*q*data.beaver_c/V...
         + data.Cm_de*Delv+data.Cm_beta2*beta^2+data.Cm_r*r*data.beaver_b/2/V+data.Cm_df*Dflap...
         + data.Cm_dpt*dpt;

    Cn = data.Cn0+data.Cn_beta*beta+data.Cn_p*p*data.beaver_b/2/V+data.Cn_r*r*data.beaver_b/2/V+data.Cn_da...
         *Dail+data.Cn_dr*Drud+data.Cn_q*q*data.beaver_c/V+data.Cn_beta3*beta^3+data.Cn_dpt3*dpt^3;

    Fx = Cx*qdyn*data.beaver_S;
    Fy = Cy*qdyn*data.beaver_S;
    Fz = Cz*qdyn*data.beaver_S;
    L = Cl*qdyn*data.beaver_S*data.beaver_b;
    M = Cm*qdyn*data.beaver_S*data.beaver_c;
    N = Cn*qdyn*data.beaver_S*data.beaver_b;


    %%
    ydot = nan(12,1);

    % Khaled Method
    ydot(1) = (Iz*L + Ixz*N - Ixz^2*q*r - Iz^2*q*r - Ixz*Iy*p*q +Ixz*Iz*p*q+Ixz*Ix*p*q+ Iy*Iz*q*r)/(Ix*Iz-Ixz^2);

    ydot(2) = (M - Ixz*p^2 + Ixz*r^2 - Ix*p*r + Iz*p*r)/Iy;

```



```matlab
    ydot(3) = (Ixz*L + Ix*N + Ix^2*p*q - Ix*Iy*p*q - Ixz^2*p*q - Ix*Ixz*q*r + Ixz*Iy*q*r - Ixz*Iz*q*r)/(Ix*Iz-Ixz^2);

    %%
    euler = [[1, sin(y(4))*tan(y(5)), cos(y(4))*tan(y(5))];...
             [0, cos(y(4)), -sin(y(4))];...
             [0, sin(y(4))*(1/cos(y(5))), cos(y(4))*(1/cos(y(5)))]] * ...
             [y(1); y(2); y(3)];

    ydot(4) = euler(1);
    ydot(5) = euler(2);
    ydot(6) = euler(3);

    %%
    ydot(7) = (Fx - g*m*sin(theta) - m*q*w + m*r*v)/m;
    ydot(8) = (Fy + m*p*w - m*r*u + g*m*cos(theta)*sin(phi))/m;
    ydot(9) = (Fz - m*p*v + m*q*u + g*m*cos(phi)*cos(theta))/m;

    %%
    C = [[cos(y(5))*cos(y(6)), sin(y(4))*sin(y(5))*cos(y(6))-(cos(y(4))*sin(y(6))), cos(y(4))*sin(y(5))*cos(y(6))+(sin(y(4))*sin(y(6)))];
         [cos(y(5))*sin(y(6)), sin(y(4))*sin(y(5))*sin(y(6))+(cos(y(4))*cos(y(6))), cos(y(4))*sin(y(5))*sin(y(6))-(sin(y(4))*cos(y(6)))];
         [-sin(y(5)), sin(y(4))*cos(y(5)), cos(y(4))*cos(y(5))]];

    Cmat = C * [y(7); y(8); y(9)];

    ydot(10) = Cmat(1);
    ydot(11) = Cmat(2);
    ydot(12) = Cmat(3);


    ydot = [ydot(1); ydot(2); ydot(3); ydot(4); ydot(5); ydot(6);...
            ydot(7); ydot(8); ydot(9); ydot(10); ydot(11); ydot(12)];
end
```

Simulation screenshot


FlightGear
File View Location Autopilot Environment Equipment AI Multiplayer Debug Help Beaver
alpha
File Tools View Simulation Help
12.2
12.1
12
11.9
0 1 2 3 4 5 6 7 8 9 10
OK Cancel Help Apply

Block Parameters: Elevator
Slider Gain (mask) (link)
Move the slider to modify the scalar gain.
Parameters
-1.0 2.0
1
Low -1
High 2

Block Parameters: Aileron
Slider Gain (mask) (link)
Move the slider to modify the scalar gain.
Parameters
-1.0 2.0
1
Low -1
High 2

Block Parameters: Rudder
Slider Gain (mask) (link)
Move the slider to modify the scalar gain.
Parameters
-1.0 2.0
1
Low -1
High 2

Ready
Sample based T=10.000

The following picture is a screenshot that shows that all the simulation subsystems are up and working correctly.

The several slider gains shown in the picture are used to manipulate the pilot's elevator, rudder, and aileron values and control / fly the aircraft in real-time. The work of autopilot software is to decide the perfect gain values to achieve the desired aircraft motion.

References

[1] M. A. Ponrani and A. K. Godweena, "Aircraft Pitch Control using PID Controller," 2021 International Conference on System, Computation, Automation and Networking (ICSCAN), 2021, pp. 1-4, doi: 10.1109/ICSCAN53069.2021.9526373.

[2]J. Pinguet, P. Feyel and G. Sandou, "A Neural Autopilot Training Platform based on a Matlab and X-Plane co-simulation," 2021 International Conference on Unmanned Aircraft Systems (ICUAS), 2021, pp. 1200-1209, doi: 10.1109/ICUAS51884.2021.9476679.

[3]Rauw, M. O. (1993). A SIMULINK environment for Flight Dynamics and Control analysis-application to the DHC-2'Beaver'. Graduate's thesis. Delft Univ. of Thechnology, the Netherlands.

[4]NOVÁK, B. J. DEVELOPMENT OF AUTOPILOT AND FLIGHT DIRECTOR MODES INSIDE A SIMULINK ENVIRONMENT.

[5]Chauhan, V., & Srivastava, P. K. (2019). Computational techniques based on Runge-Kutta method of various order and type for solving differential equations. International Journal of Mathematical, Engineering and Management Sciences, 4(2), 375-386.Chauhan, V., & Srivastava, P. K. (2019). Computational techniques based on Runge-Kutta method of various order and type for solving differential equations. International Journal of Mathematical, Engineering and Management Sciences, 4(2), 375-386.

Conclusions

We have achieved our target of creating a simple autopilot system. The SIMULINK model communicates with a FlightGear simulation instance running at a particular UDP port.

The ISA model for atmospheric conditions has been implemented successfully, providing an accurate result.
For future scope this system can be used for basic structures of autopilot designs using neural networks. Autopilots which use neural networks for training will collect data from either real pilots or implemented autopilot systems like these. A neural controller then can be trained on this data and a final version can be implemented.

All our project files are uploaded on the following Github repository: https://github.com/vish2001/Aircraft_dynamics-n-control.git