Cairo University
Faculty of Engineering
Aerospace Department
Third Year

Flight Mechanics (AER 307A)

Group #7

Flight Simulator Using MATLAB and Simulink

Submitted to: Prof. Ayman Hamdy Kassem

Names :	Ahmed Raafat Abd el-Fattah	BN: 4	Sec: 1
	Ahmed Hisham Mohamed	BN: 7	Sec: 1
	Aya-T-Allah Ashraf	BN: 11	Sec: 1
	Amr Saber Hashem	BN: 26	Sec: 1
	Mohamed Hesham Amin	BN: 17	Sec: 2
	Mahmoud Sayed Ramadan	BN: 19	Sec: 2

Table of Contents

1. Introduction

Our group has successfully created the following:

- 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 actually fly/control the aircraft.

In this report, we will go through the process of creating the above-mentioned software for the DHC-2 'beaver' aircraft. A flexible environment for the analysis of aircraft dynamics and control will be developed for the DHC-2 'Beaver' aircraft. This environment uses the power and flexibility of the simulation and system analysis programs SIMULINK and MATLAB.

An understanding of the dynamic characteristics of an airplane is important in assessing its handling or flying qualities as well as for 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 airplane can be considered to be dynamically stable if after being disturbed from its equilibrium flight condition the ensuing motion diminishes with time.

Section 2 will give a problem definition and discuss the aircraft equations of motion and the approach taken to solve these equations. Section 3 will obtain results and solution of the equations to obtain the aircraft states. Section 4 will discuss the graphical simulator designed using SIMULINK and FlightGear.

2. Problem definition

Before developing the equations of motion, we need to define our system of axes, Figure 1 shows the two sets of coordinate systems used, the body axis system attached to the aircraft and the inertial axis system that is fixed to the Earth.

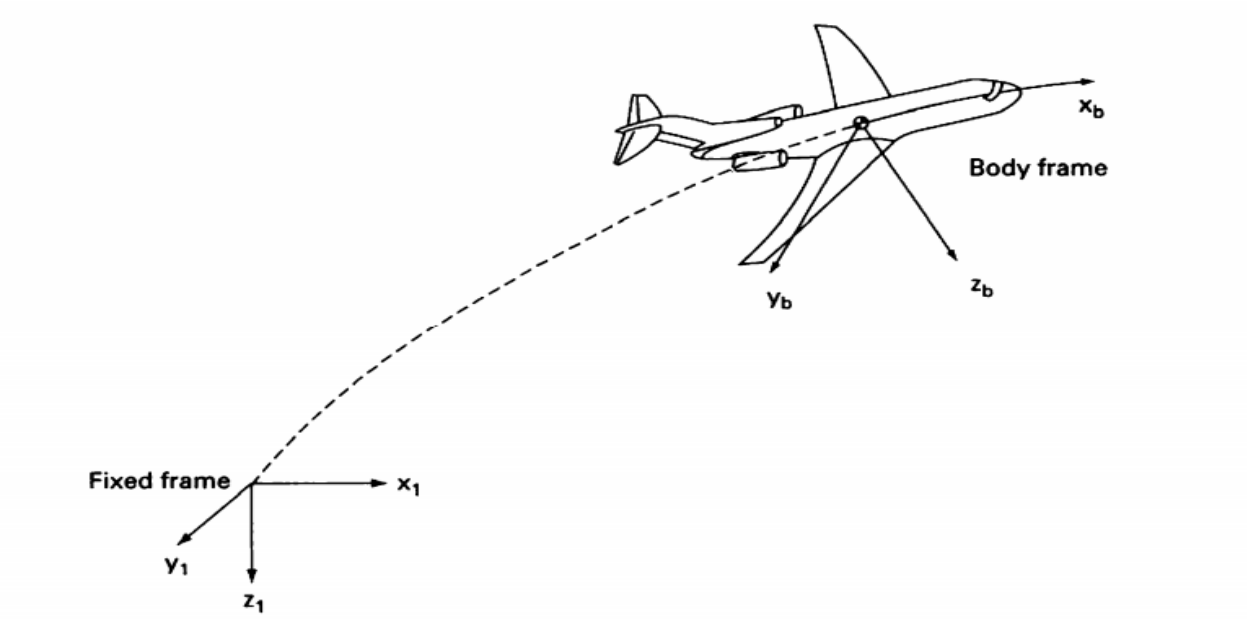


Figure 1 axis system

Using the rigid body motion equations to describe the motion of the aircraft, obtained from newton's second law:

$$\Sigma F = \frac{d}{dt}(mV)$$

$$\Sigma M = \frac{d}{dt}H$$

These are vector equations, where F_x, F_y, F_z, and u, v, w are the components of the force and velocity along the x, y, and z axes, respectively. The force components are composed of contributions due to the **aerodynamic, propulsive, and gravitational forces** acting on the airplane.

While, L, M, N and H_x, H_y, H_z, are the components of the moment and moment of momentum along the x, y, and z axes, respectively.

The orientation and position of the airplane can be defined in terms of a fixed frame of reference as shown in Figure 3.3, The orientation of the airplane can be described by three consecutive rotations, whose order is important. The angular rotations are called the ***Euler angles***.

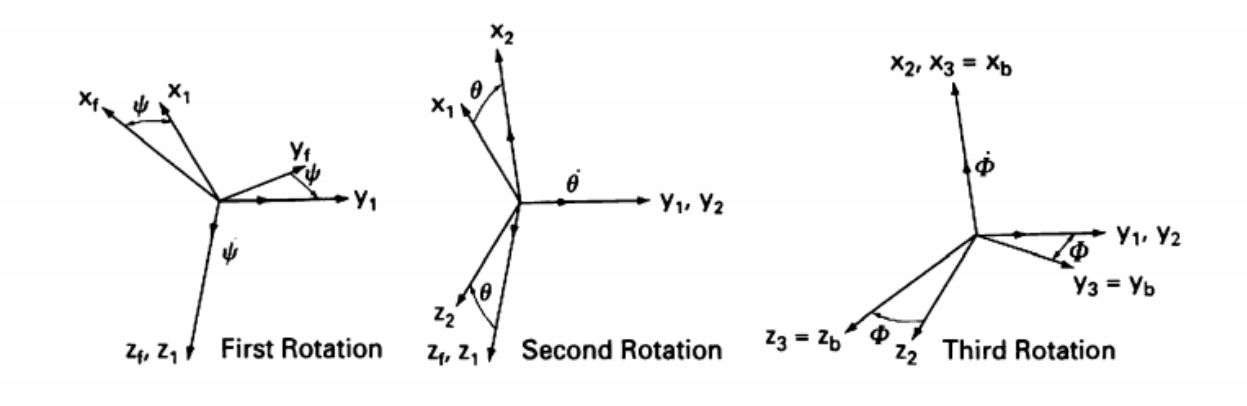


Figure 2 Euler angles

It can be shown that these fundamental newton equations in addition to the Euler equations can be manipulated to obtain the twelve governing equations of the motion of the aircraft as follows:

$$L = I_x \dot{p}_i - I_{xz}\dot{r} + qr(I_z - I_y) - I_{xz}pq$$
$$M = I_y \dot{q} + qr(I_x - I_z) + I_{xz}(p^2 - r^2)$$
$$N = I_{xz}\dot{p}_i + I_z \dot{r} + qp(I_y - I_z) - I_{xz}rq$$

$$F_x - mg\sin(\theta) = m(\dot{u} + qw - rv)$$
$$F_y + mg\cos(\theta)\sin(\phi) = m(\dot{v} + ru - pw)$$
$$F_z + mg\cos(\theta)\cos(\phi) = m(\dot{w} + pv - qu)$$

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

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

These are twelve non-linear ordinary differential equations in twelve variables, which represent the state vector of the aircraft $[p, q, r, u, v, w, \varphi, \theta, \psi, x, y, z]$.

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

state	**value**	**state**	**value**
V	35	x	0
α	0.211	y	0
β	-0.0206	z	0
p	0	δe	-0.093
q	0	δa	0.009624
r	0	δr	-0.0495
ϕ	0.191	δf	0
θ	0	n	1800
ψ	0	p_z	20

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

3. Calculations and Results

Our MATLAB code presented in this report, will implement the most critical parts of the DHC-2 "Beaver" aircraft model, which are the aerodynamic model, engine model, atmosphere and air data model. The non-linear state equations themselves will be solved numerically using **Rung-Kutta** method for solving ordinary differential equations, to obtain the aircraft states and their deviation from the trimmed point during an interval of time. We will walk through the calculations steps we toke to solve the Equations. The following flow chart describes the sequence of the project.

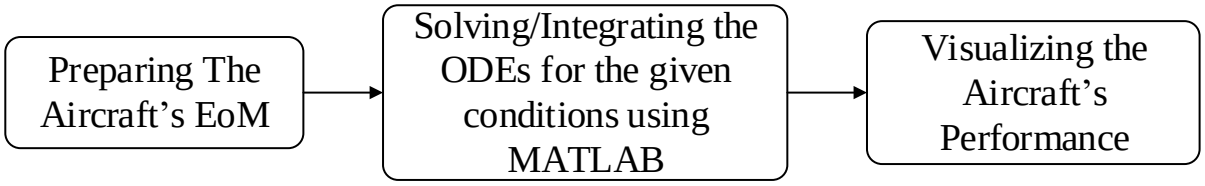


Fig. 3.1: Project Sequence

3.1 Decoupling the Equations of Motion

In order to solve the twelve ordinary differential equations using the Rung-Kutta method 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.

$$\dot{p} = \frac{I_z L + I_{xz} N + I_{xz}^2 qr - I_z^2 qr - I_{xz} I_y pq + 2 I_{xz} I_z pq + I_y I_z qr}{I_{xz}^2 + I_x I_z}$$

$$\dot{r} = -\frac{I_{xz} L - I_x N + I_{xz}^2 pq + I_x I_y pq - I_x I_z pq - I_x I_{xz} qr + I_{xz} I_y qr - I_{xz} I_z qr}{I_{xz}^2 + I_x I_z}$$

3.2 Solving the Equations of Motion

Now that the rates are well defined and the equations are ready, we can easily solve them by any of the available MATLAB solvers. Table 3.2 discussed some of the available solvers.

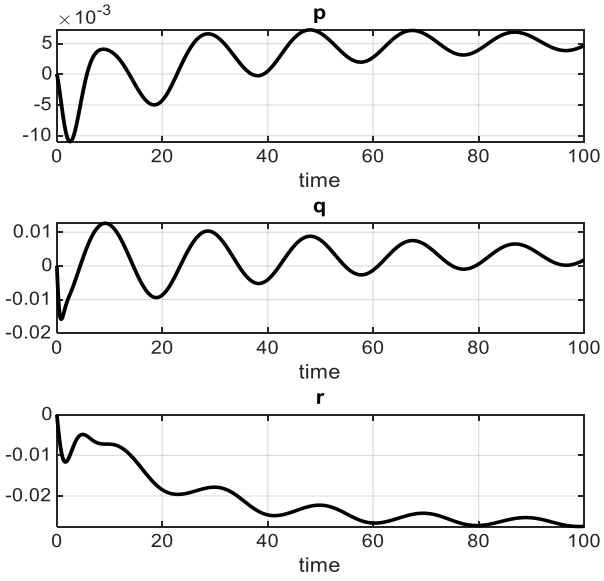
Solver	Problem Type	Order of Accuracy
Ode113	Nonstiff	Low to high
Ode15s	Stiff	Low to medium
Ode45	Nonstiff	Medium
Ode23	Nonstiff	Low
Ode23tb	Stiff	Low

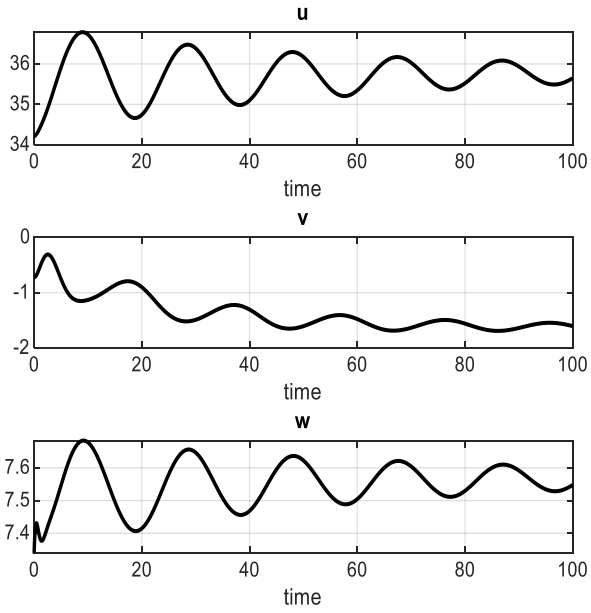
Table 3.2: MALTAB ODEs Solvers

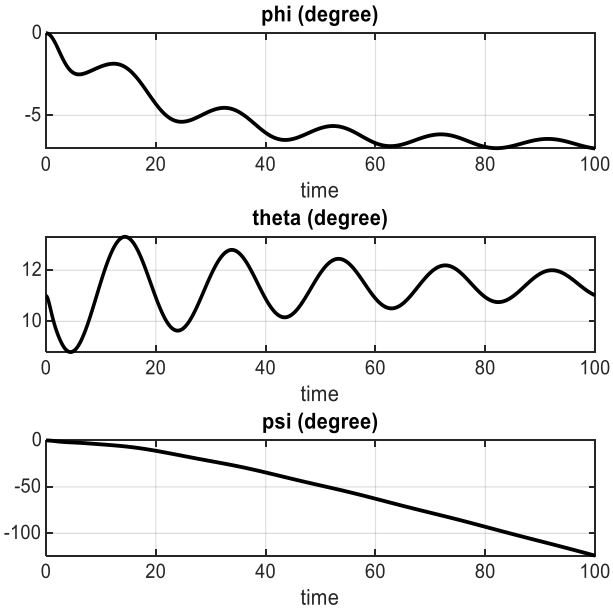
After reviewing the different ODEs solvers in MATLAB, we decided to use the ode45 solver, because ode45 is based on an explicit Runge-Kutta formula. It is a one-step solver (in computing $y(tn)$, it needs only the solution at the immediately preceding time point $y(tn-1)$.) In general, ode45 is the best function to apply as a "first try" for most problems.

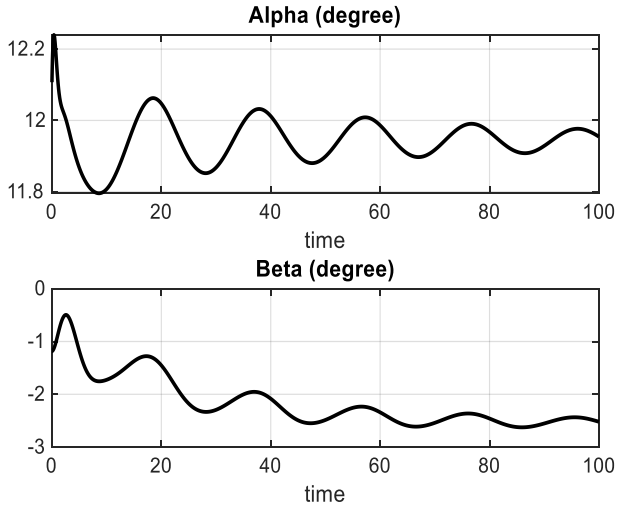
3.3 Results

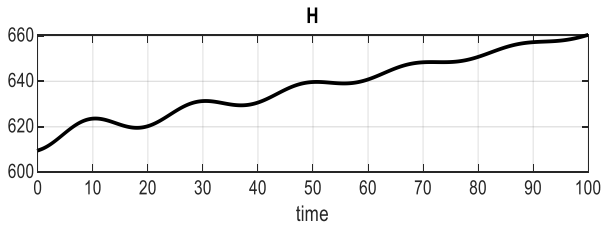
Solving the 12 non-linear ordinary differential equations at the given trim conditions for a time span of 100 *seconds* resulted the following results.

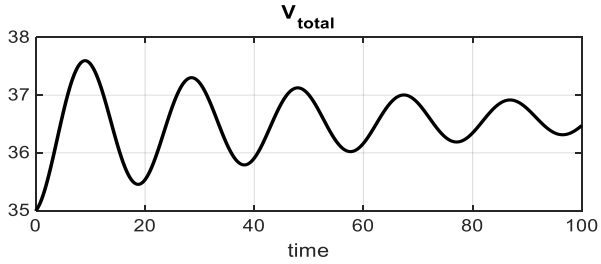












3.4 Comments on Results

The aircraft clearly 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 x-velocity component must have the largest value, while the y-component have the lowest change in value 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 plotted to show the aircraft motion during time. Figure 3.4 shows the position of the A/C in space. The aircraft's position plot also clearly suggests that the results are valid, as it shows a typical aircraft's motion for a trim state.

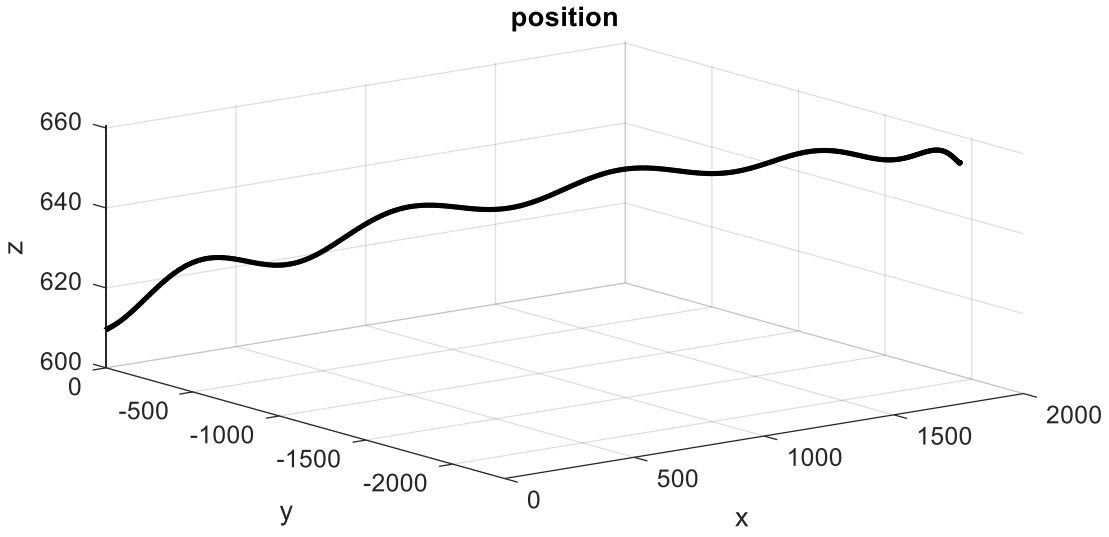


Fig. 3.4: Aircraft position plot

4. Simulink Model and FlightGear Integration

Having already done the "hard" part of the project which was solving the differential equations for the 12 state variables, we wanted to visualize and simulate the aircraft performance in real-time. Hence, we created a Simulink model based on the same MATLAB function we created earlier -which calculated the time rates of the state variables- where we used a simple "integrator" block to calculate the variables in real-time during simulation.

The following flow-chart shows the main blocks / subsystems of our Simulink model.

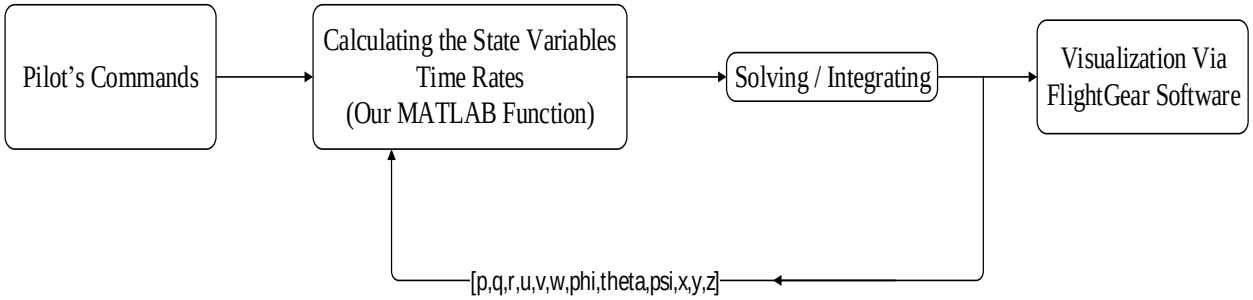


Fig 4.1: Simulink Model Main Blocks

The following picture in figure 4.2 is a screenshot from one of our PCs; it shows that all the simulation subsystems are up and working correctly.

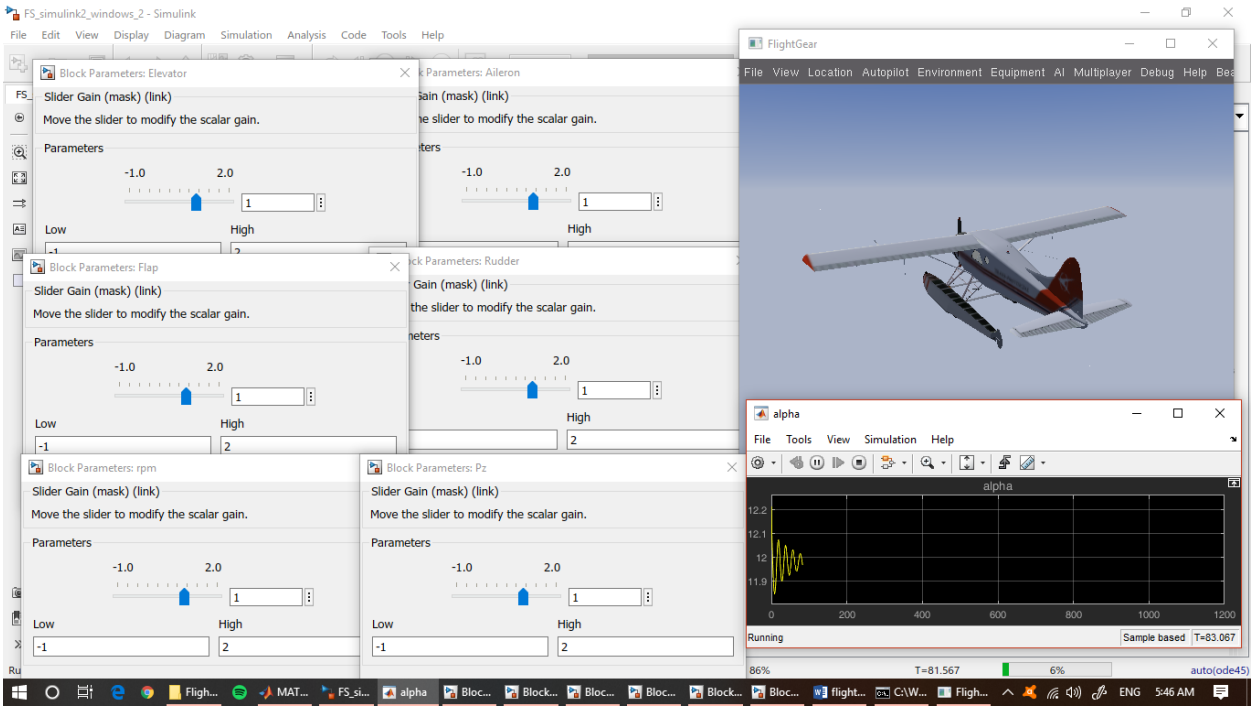


Fig 4.2: Simulation Screenshot

The several slider gains shown on the left of figure 4.2 are used to manipulate the pilot's commands, and hence control / fly the aircraft in real-time.

Regarding the Simulink model itself, a picture of it can be found in the appendix. Also, all our work can be found in a zip file we uploaded to google drive and can be downloaded from the following link.

https://drive.google.com/file/d/188Hx93oqAaJYVEsz58bNaqPuiV3TOCdN/view?usp=sharing

5. References

[1] Robert C. Nelson. *flight stability and automatic control*. Second edition, United States, McGraw-Hill, INC., 1998.

[2] MSc-thesis - A SIMULINK environment for flight dynamics and control analysis: Application to the DHC-2 Beaver

6. Appendix

6.1 MATLAB main script code

```
close all
clear all
clc
format longG
%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) odefun5(t,sol), [0 tend], IC);

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)
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', 1.5)
title('V_{total}'); xlabel('time'); grid on
figure(7)
subplot(311)
plot(t, sol(:,10), 'k', 'LineWidth', 1.5)
title('x'); xlabel('time'); grid on
subplot(312)
plot(t, sol(:,11), 'k', 'LineWidth', 1.5)
title('y'); xlabel('time'); grid on
subplot(313)
plot(t, sol(:,12), 'k', 'LineWidth', 1.5)
title('H'); xlabel('time'); grid on
```

6.2 odefun5.m code

```
function [ ydot ] = odefun5( t,y )
%odefun calculates the time rates to be used in ode45 function
    % 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
    load('dhc2_vars.mat')
    Ix = 5368.39; Iy = 6.92893e3; Iz = 11158.75;
    Ixy = 0; Ixz = 1.1764e2; Iyz = 0;
    m = 2288.231; g = 9.81;
    %inputs and input vector U
    Delv = -9.3083e-002; %35
    Dail = 9.6242e-003; %35
```

```
    Drud = -4.9242e-002; %35
    Dflap = 0;
    n = 1800; %rpm
    Pz = 20; %mainfold pressure ["Hg] 35mps
    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); xe = y(10); ye = y(11); z = y(12);

    V = sqrt(u^2 + v^2 + w^2);

    alpha = atan2(w,u);

    beta = asin(v/V);

    rho = 1.225*exp(-g*(z)/287.05/(288-0.0065*(z)));

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

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

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

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

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

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

    Fx = Cx*qdyn*beaver_S;
    Fy = Cy*qdyn*beaver_S;
    Fz = Cz*qdyn*beaver_S;
    L = Cl*qdyn*beaver_S*beaver_b;
```

```
    M = Cm*qdyn*beaver_S*beaver_c;
    N = Cn*qdyn*beaver_S*beaver_b;

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

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

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

    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
```

6.3 Simulink Model