Toronto Metropolitan University

Department of Aerospace Engineering
Faculty of Engineering & Architectural Science

Semester (Term, Year): Winter 2024

Course Code: AER 404

Course Section: 01

Course Title: Introduction to Aerospace Engineering Design

Course Instructor: Dr. Reza Faieghi

Submission: Report

Submission No.: 1

Submission Due Date: 01/04/2024

Title: Flight Simulator Physics Engine

Submission by (Name):	Student ID (XXXX1234)	Signature
Anna Topacio	0648	
Mujtaba Tariq	9459	
Arman Suleman	0408	
Tasnim Fatema	8104	
Keith Ayok	5022	

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

Testing parts and full-scale models is not only costly but also inefficient and can be dangerous. For these reasons, simulation is used to model the project to gauge characteristics and performance. In this project, a new airliner is creating a competitor to the Boeing 737 and Airbus A320 families of aircraft. We, as engineers, will use a combination of Simulink and MATLAB to create the physics engine for a flight simulator to be used to train pilots. The following equations will be modeled in MATLAB; aerodynamic coefficients, forces and moments, Engine forces and moments, force equations, moment equations, kinematic equations, and geodetic coordinate equations. These will be made to accurately simulate the flight and will be placed into Simulink blocks for input, output, and iteration. The following report will revise the problem statement, create an objective tree, overview the design, and show all the equations explained above as well as show results and evaluations of the simulation.

2.0 OBJECTIVE

The objective of this project is to collaborate within teams by designing, implementing, and testing a specialized engineering software system. Through working on this project, we aim to achieve several key learning outcomes, such as gaining proficiency in using MATLAB and Simulink, applying theoretical concepts from calculus and dynamics courses to real-world scenarios like aircraft flight dynamics, and developing foundational programming skills for engineering applications. Initiated by an aircraft manufacturing company, this project involves creating a flight simulator for an aircraft that will be used for pilot training. As part of the Physics Engine group, we must implement equations to simulate the aircraft's flight behavior accurately. Our goal is to understand and implement the process of receiving input commands from a joystick and output this information into actions within the flight simulator.

3.0 PROBLEM STATEMENT

A flight simulation division has been created within a new airliner that is creating an aircraft to compete with the Boeing 737 and Airbus A320 family of aircraft. The primary task is to simulate the six degrees of freedom in an aircraft by creating equations to predict the following:

- Forces from the engine, aileron, rudder, elevator
- All states of the Euler angles (roll, pitch, yaw)
- Longitudinal, lateral, and vertical speed
- Geodetic coordinates

These will be made with flat earth equations. All aerodynamic characteristics and coefficients have been found and provided as well as response times and maximum outputs. The project must be completed in MATLAB and Simulink, flight gear will provide graphics and a joystick will be provided that matches what will be used in reality.

3.1 OBJECTIVE TREE

To better understand developing flight simulator software, we use an objective tree, breaking down goals into manageable parts. This visual aid clarifies objectives, identifies dependencies, and enhances communication with stakeholders. Ultimately, it summarizes the main goal: creating a flight simulator that accurately translates joystick commands into simulator actions.

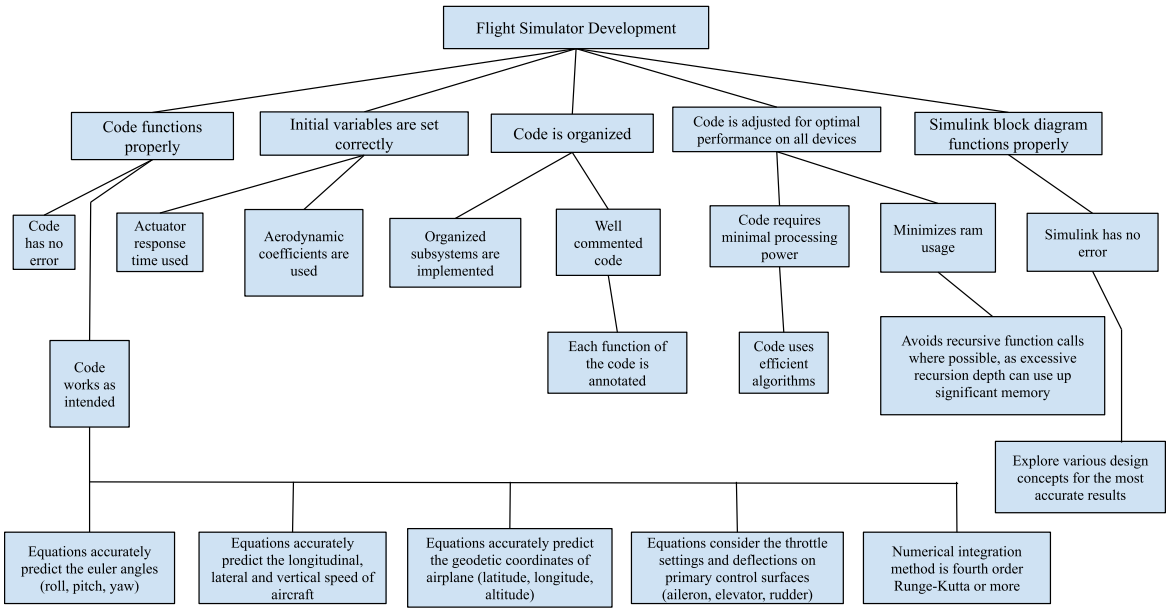

Figure 1: Objective Tree of Desired Attributes of Flight Simulator

The objective tree is clarified with a mission requirements and objectives (MRO) table, aligning project goals. It lists general objectives under "category" and specific approaches under "MR&O attribute." Client requirements are in the "requirements" column, and project goals are in the "objective" column. This table aids decision-making and tracks project progress.

Table 1: Mission Requirements and Objectives (MRO) of the Flight Simulator

Category	MR&O Attribute	Requirement (R)	Objective (O)
Code functions	Code has no error	N/A	Very minimal to no

Category	MR&O Attribute	Requirement (R)	Objective (O)
properly			error
	Code works as intended	Equations accurately predict the Euler angles (roll, pitch, yaw)	Plots accurately match the trajectories given for the roll, pitch and yaw angles
		Equations accurately predict the longitudinal, lateral, and vertical speed of aircraft	Plots accurately match the trajectories given for the longitudinal, lateral and vertical speeds
		Equations accurately predict the geodetic coordinates of the airplane (latitude, longitude, altitude)	Connect the geodetic coordinates using the Earth spheroid model
		Equations consider the throttle settings and deflections on primary control surfaces (aileron, elevator, rudder)	Equations can command inputs from the joystick with no error
		Flat-Earth equations are accurate and the numerical integration method is fourth-order Runge-Kutta	Equations can perform outputs with no error
Initial variables are set correctly	Actuator response time used	N/A	Use the variables given for the engine and actuator response time
	Aerodynamic coefficients are used	N/A	Use the variables given for the aircraft's aerodynamic coefficients
Code is organized	Organized subsystems are	N/A	Divide code into sections, allowing

Category	MR&O Attribute	Requirement (R)	Objective (O)
	implemented		you to run all sections or run each section individually
	Well commented code	N/A	Each function of the code is annotated
Code is adjusted for optimal performance on all devices	Code required minimal processing power	N/A	Code uses efficient algorithms
	Minimizes ram usage	N/A	Avoids recursive function calls when possible to use less space
Simulink block diagram functions properly	Simulink has no error	N/A	Very minimal to no error
			Explore various design concepts for the most accurate results
			Create subsystems

In this section, we define and document metrics for each objective in Table 1, scoring them from 0 to 2. This helps rank design concepts based on how well they meet criteria. A score of 0 means not meeting criteria, while 2 means excelling at it. We consider factors like functioning code, initial variables are set correctly, code is organized, code is adjusted for optimal performance on all devices, and functioning simulink block diagram. The goal is to systematically find the best design concept across these metrics.

Table 2: Evaluation Metrics for the Flight Simulator

Objective	Metric	Value
Code functions properly	No errors	2
	Few errors	1
	Many errors	0
Initial variables are set	Precise variables are used	2

Objective	Metric	Value
correctly	Adjust variable amount to ensure code functions	1
	Only aerodynamics coefficients are used	0
Code is organized	Includes sections and comments	2
	Includes no comments	1
	Does not include subsystems or comments	0
Code is adjusted for optimal performance on all devices	Uses very minimal space	2
	Uses minimal space	1
	Uses a significant amount of space	0
Simulink block diagram functions properly	No errors	2
	Few errors	1
	Many errors	0

3.2 HIERARCHICAL BLOCK DIAGRAM

A brief overview of how the simulink model works is illustrated in Figure 2. The inputs that are commanded from the joystick will be fed into the aerodynamic forces and moment equations. The outputs will be the 12 variables of the aircraft that describe the six-degrees-of the freedom flight equations and scope blocks will be used to visualize the software outputs before connecting to FlightGear [3].

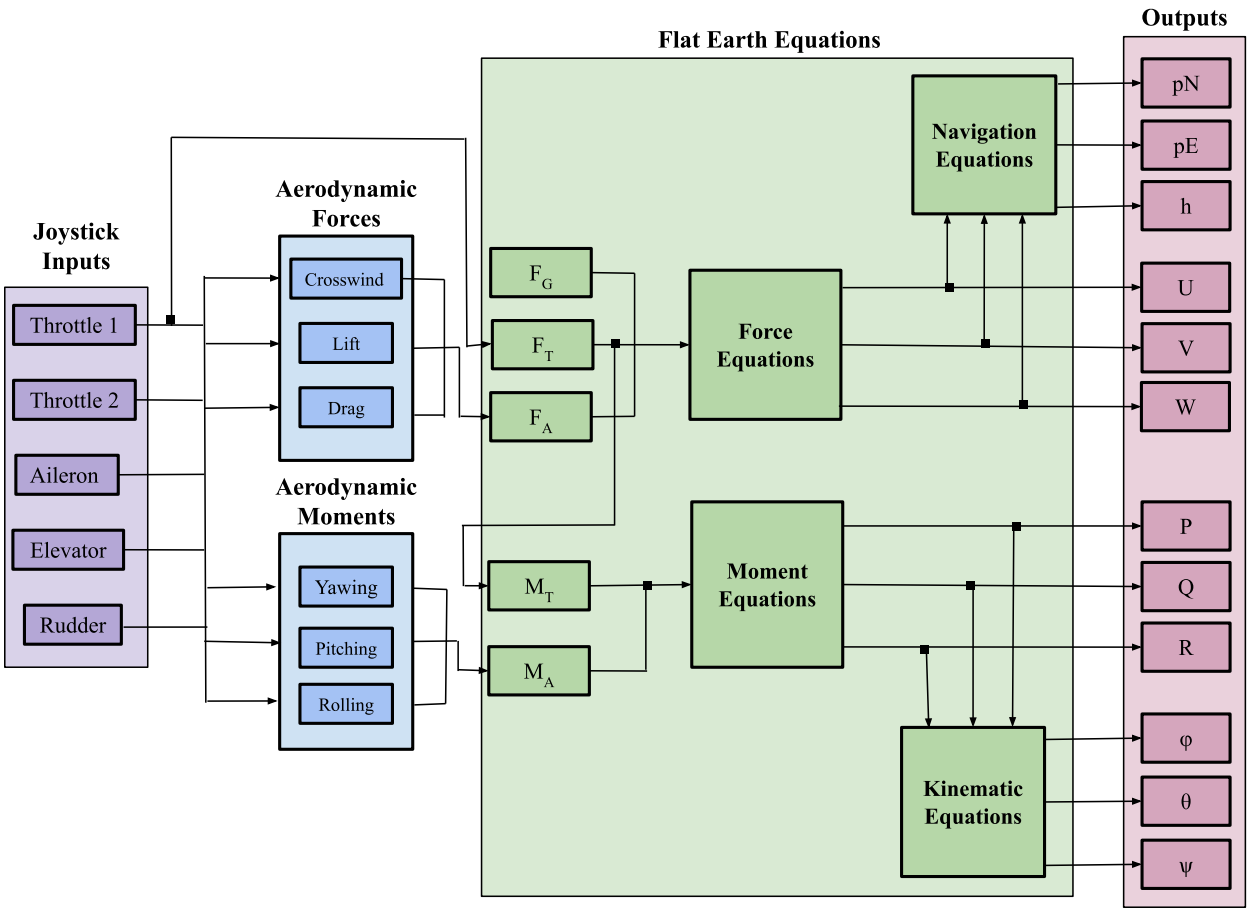

Figure 2: Data Flow in the Software

The system involves processing joystick inputs which include aileron, elevator, rudder deflection angles, engine throttle settings. Aerodynamic coefficients and forces, including lift, drag, crosswind, rolling moment, pitching moment, and yawing moment coefficients, are calculated. These coefficients further dictate the calculation of drag, crosswind, and lift forces. The core calculations are conducted within the Flat Earth Equations section, where force and moment equations are manipulated through navigation and kinematic equations. The output values, including longitude, latitude, altitude, angles, speeds, and rates, are visualized graphically and transmitted to the FlightGear simulator to dictate the movement of the aircraft model.

4.0 DESIGN DESCRIPTION

The objective of the physics engine is to utilize the mentioned five inputs to forecast the 12 aircraft variables that represent six degrees of freedom flight. Consequently, these variables serve as the outcomes generated by the physics engine and will be transmitted to FlightGear. The following equations were given from the *Project 2 Supplementary Material* [4].

4.1 EQUATIONS, INPUTS AND OUTPUTS

Force equations that calculate the translational velocity components:

- $U = longitudinal\ speed$
- $V = lateral\ speed$
- $W = vertical\ speed$

Moment equations to calculate body rate components:

- P = roll rate
- Q = pitch rate
- R = Yaw rate

Kinematic equations:

- $\phi = roll\ rate$
- θ = pitch angle
- ψ = Yaw angle

Navigation equations:

- $p_N = north\ component\ of\ the\ position\ vector$
- $p_E = east\ component\ of\ the\ position\ vector$
- $h = altitude$

CONSTANTS AND VARIABLES

Prior to incorporating the primary equations, it is necessary to establish numerous constants and fundamental variables essential for the flight equations. The aircraft in focus is identified as a narrow-body twinjet airliner, characterized by the following properties:

- Mass, m = 120000 kg
- Inertia matrix, $: J = m\begin{pmatrix} 40.07 & 0 & -2.0923 \\ 0 & 64 & 0 \\ -2.0923 & 0 & 99.92 \end{pmatrix}$ kg.m^2
- Coordinates of engine 1 thrust application point: $apt_1 = (0, \ -7.94, \ -1.9)^T$
- Coordinates of engine 2 thrust application point: $apt_2 = (0, \ -7.94, \ -1.9)^T$
- Wing planform area: $S = 260\ m^2$
- Wing planform area of tail: $S_t = 64\ m^2$

- Generalized length of wings: $l = 6.6\ m$
- Distance between the aircraft c_m and the a_c of tail: $l_t = 24.8\ m$
- Angle of attack when lift is zero, $\alpha_o = -11.5\ degree$
- Air density $\rho = 1.225\ kg/m^3$
- $g = 9.81\ m/s^2$

Alongside the constants mentioned earlier, various fundamental variables must be computed throughout the flight simulation. These variables play crucial roles in numerous equations, particularly in the calculation of aerodynamic coefficients. Below are these variables listed along with their corresponding formulas:

$$\text{Airspeed, } V_T = \sqrt{U^2 + V^2 + W^2} \textbf{ (Equation 1)}$$

$$\text{Dynamic Pressure, } q = 1/2\rho V_T^{\ 2} \textbf{ (Equation 2)}$$

$$\text{Angle of attack, } \alpha = tan^{-1}(\frac{W}{U}) \textbf{ (Equation 3)}$$

$$\text{Sideslip angle, } \beta = sin^{-1}(\frac{V}{V_T}) \textbf{ (Equation 4)}$$

When determining the aerodynamic coefficients, it is essential to separately account for the impact of the aircraft's tail. Consequently, two extra factors must be computed and incorporated into the calculation of the aerodynamic coefficients. These factors encompass:

$$\text{Downwash angle of the wings, } \varepsilon = 0.25(\alpha - \alpha_o) \textbf{ (Equation 5)}$$

$$\text{Angle of attack of the tail, } \alpha_t = \alpha - \varepsilon + \delta_E + 1.3Q\frac{l_t}{V_T} \textbf{ (Equation 6)}$$

4.2 AERODYNAMIC COEFFICIENTS

The lift coefficient is determined by adding together the lift coefficients of the wing and the tail as:

$$C_L = C_{L,Wb} + C_{L,t} \textbf{ (Equation 7)}$$

These coefficients are as follows:

$$C_{L,Wb} = 5.5\ (\alpha - \alpha_o) \textbf{ (Equation 8)}$$

$$C_{L,t} = 3.1\frac{S_t}{S}\alpha_t \textbf{ (Equation 9)}$$

The drag coefficient is calculated using the following equation,

$$C_D = 0.13 + 0.07(5.5\alpha + 0.654)^2$$ **(Equation 10)**

The cross-wind equation is calculated using the following equation,

$$C_c = -1.6\beta + 0.24\delta_R$$ **(Equation 11)**

Rolling moment coefficient

$$C_l = -1.4\beta - 11\frac{l}{V_T}P + 5\frac{l}{V_T}R - 0.6\delta_A + 0.22\delta_R$$ **(Equation 12)**

Pitching moment coefficient

$$C_m = -0.59 - 3.1\frac{S_t L_t}{Sl}(\alpha - \epsilon) - 4.03\frac{S_t L_t^2}{Sl^2}\frac{l}{V_T}Q - 3.1\frac{S_t L_t}{Sl}\delta_E$$ **(Equation 13)**

Yawing moment coefficient

$$C_n = (1 - 3.8179\alpha)\beta + 1.7\frac{l}{V_T}P - 11.5\frac{l}{V_T}R - 0.63\delta_R$$ **(Equation 14)**

4.3 AERODYNAMIC FORCES

To calculate drag, crosswind, and lift forces in the wind coordinates, the following equations were used:

$$D = qSC_D$$ **(Equation 15)**

$$C = qSC_c$$ **(Equation 16)**

$$L = qSC_L$$ **(Equation 17)**

The resultant forces will comprise the actual drag (X_A), crosswind (Y_A), and lift (Z_A) forces experienced during flight. These forces are essential inputs for the force equations.

$$X_A = -Dcos\alpha cos\beta + Ccos\alpha sin\beta + Lsin\alpha$$ **(Equation 18)**

$$Y_A = -Dsin\beta - Ccos\beta$$ **(Equation 19)**

$$Z_A = -Dsin\alpha cos\beta + Csin\alpha sin\beta - Lcos\alpha$$ **(Equation 20)**

4.4 AERODYNAMIC MOMENTS

The actual aerodynamic moments are calculated using the following vector operation:

$$\begin{pmatrix} l_A \\ m_A \\ n_A \end{pmatrix} = \begin{pmatrix} l'_A \\ m'_A \\ n'_A \end{pmatrix} + \begin{pmatrix} X_A \\ Y_A \\ Z_A \end{pmatrix} \times \begin{pmatrix} 0.11\overline{c} \\ 0 \\ 0.1\overline{c} \end{pmatrix}$$ **(Equation 20)**

4.5 ENGINE FORCES

As the engine forces only operate on the X-axis, The following is the X_T formula relation:

$$X_T = X_{T1} + X_{T2} = \delta_{T1} mg + \delta_{T2} mg \quad \textbf{(Equation 21)}$$

Where δ_{T1} and δ_{T2} represent the throttle settings of engine one and two. A rate limiter is used in Simulink to provide a delay in response rate.

4.6 ENGINE MOMENTS

A moment is produced by each engine in opposite directions to each other, below are the moment equations:

$$(l_{Ti}, m_{Ti}, n_{Ti})^T = (X_{Ti}, 0, 0)^T * (cm - apt_i) \quad \textbf{(Equation 22)}$$

This will provide total roll, pitch, and yaw moments generated by the engines on the aircraft.

4.7 FORCE EQUATIONS

Below are the force equations for the U, V, and W directions:

$$U = RV - QW - g\sin\theta + \frac{X_T + X_A}{m} \quad \textbf{(Equation 23)}$$

$$V = -RU + PW - g\sin\phi\cos\theta + \frac{Y_T + Y_A}{m} \quad \textbf{(Equation 24)}$$

$$W = QU - PV + g\cos\phi\cos\theta + \frac{Z_T + Z_A}{m} \quad \textbf{(Equation 25)}$$

These center around the center of mass of the aircraft. The terms with gravity "g" are related to the force from the weight of the aircraft, the terms "A" are from the aerodynamic forces, and the terms with "T" are the forces coming from the engine thrust. The thrust and aerodynamic forces are in their respective X, Y, and Z directions.

4.8 MOMENT EQUATIONS

The moment equations of an aircraft are as follows:

$$\Gamma\dot{P} = J_{xz}[J_x - J_y + J_z]PQ - [J_z(J_z - J_y) + J_{xz}^2]QR + J_z(l_A + l_T) + J_{xz}(n_A + n_T), \quad \textbf{(Equation 26)}$$

$$J_y\dot{Q} = (J_z - J_x)PR - J_{xz}(P^2 - R^2) + m_A + m_T, \quad \textbf{(Equation 27)}$$

$$\Gamma\dot{R} = [(J_x - J_y)J_x + J_{xz}^2]PQ - J_{xz}[J_x - J_y + J_z]QR + J_{xz}(l_A + l_T) + J_x(n_A + n_T). \quad \textbf{(Equation 28)}$$

It is worth mentioning that the moment's equations depict how moments influence flight dynamics, encompassing aircraft body rates, moments, and the aircraft's inertia matrix exclusively. The inertia matrix serves to represent how the mass of the aircraft is distributed across its X, Y, and Z axes.

4.9 KINEMATIC EQUATIONS

The kinematic equations are as follows:

$$\dot{\phi} = P + \tan\theta(Q\sin\phi + R\cos\phi) \quad \textbf{(Equation 29)}$$

$$\dot{\theta} = Q\cos\phi - R\sin\phi \quad \textbf{(Equation 30)}$$

$$\dot{\psi} = \frac{Q\sin\phi + R\cos\phi}{\cos\theta} \quad \textbf{(Equation 31)}$$

4.10 NAVIGATION EQUATIONS

The navigational equations are used to derive the GPS coordinates of the aircraft. These equations describe the positioning of the aircraft relative to a local origin. They use cartesian coordinates to describe the position as North (x) East (y) and Down (z) (NED). The three equations are obtained by multiplying a three-dimensional rotation matrix by the input U, V, and W values. Three separate equations are used to determine the x, y, and z coordinates but they each utilize the U, V, and W values and the angles relative to the three axes denoted as theta(θ), phi(ϕ), and psi(ψ).

$$\dot{p}_N = U\cos\theta\cos\psi + V(-\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi) + W(\sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi) \quad \textbf{(Equation 32)}$$

$$\dot{p}_E = U\cos\theta\sin\psi + V(\cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi) + W(-\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi), \quad \textbf{(Equation 33)}$$

$$\dot{h} = U\sin\theta - V\sin\phi\cos\theta - W\cos\phi\cos\theta. \quad \textbf{(Equation 34)}$$

4.11 GEODETIC COORDINATES EQUATIONS

Understanding where things are on Earth using the imaginary NED coordinate system is not useful on its own. However, by converting these coordinates to geodetic coordinates, one can utilize GPS to enable their flight simulator to work directly with GPS locations. The Earth's equatorial radius is measured at

a= 6378137 m

Earth's eccentricity (dimensionless unit)

e= 0.081819190842622

The meridian radius can be calculated using the following equation,

$$M = \frac{a(1-e^2)}{\sqrt{(1-e^2 sin^2 \phi)^3}}$$ **(Equation 35)**

The prime vertical radius can be calculated using the following equation,

$$N = \frac{a}{\sqrt{(1-e^2 sin^2 \phi)}}$$ **(Equation 36)**

Calculating these parameters will help to implement the equations that will convert the NED coordinates into GPS coordinates.

$$\dot{\mu} = \frac{\dot{p}_N}{M+h}$$ **(Equation 37)**

$$\dot{\lambda} = \frac{\dot{p}_E}{(N+h)\cos\mu}$$ **(Equation 38)**

4.12 SUBSYSTEMS

There are two subsystems within the simulink block. The initial block on the left represents the joystick input, while the subsequent block in the middle serves as the computational hub for all equations, referred to as "Aircraft" in Figure 3. Additionally, it includes a component named "Gen FG Run," responsible for creating a .bat file that initiates the flight simulator.

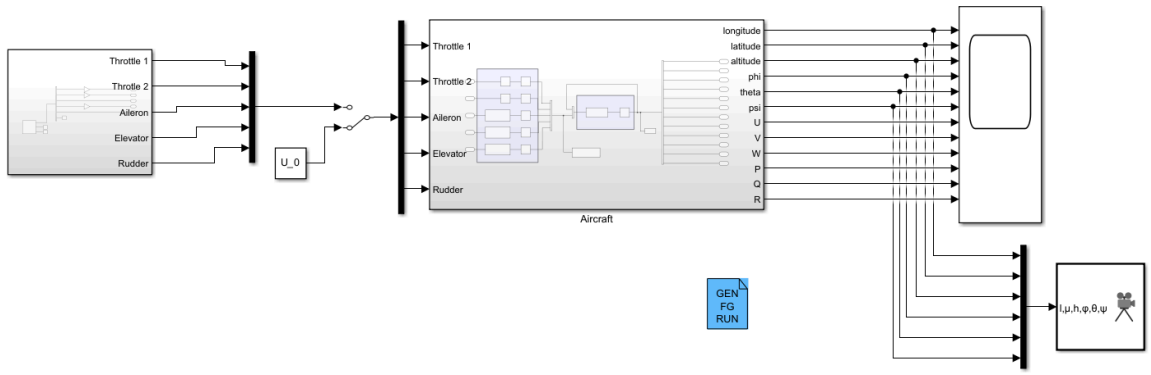

Figure 3: Simulink layout for the aircraft model

In the main subsystem (refer to Figure 4), joystick inputs are converted into radians for realistic representation. Saturation blocks mimic the delay between joystick movement and aircraft response. After processing, joystick input is saved to MATLAB Workspace and combined with initial aircraft conditions for parameter adjustments. The MATLAB Function performs all calculations, yielding "X_dot." This output is integrated, utilizing initial aircraft conditions, to produce x for use in the Flight Sim and subsequent calculations.

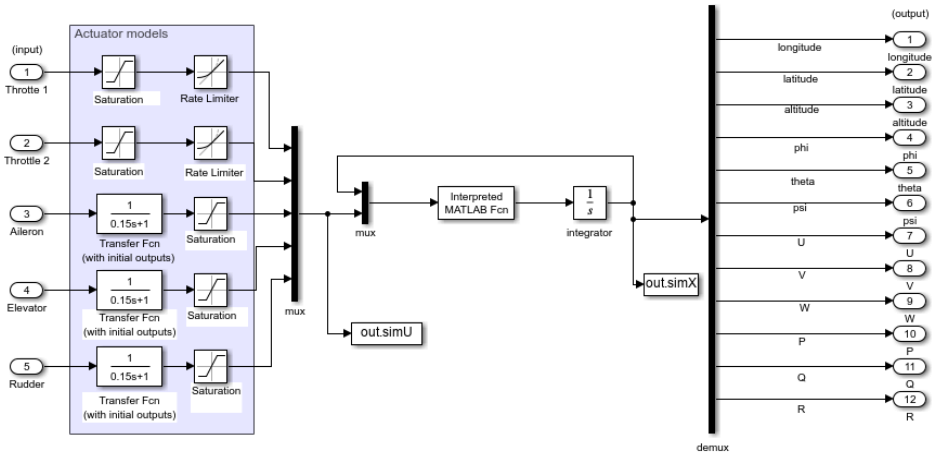

Figure 4: Simulink subsystem layout for the aircraft

The Joystick Block incorporates inputs from the joystick, a mux, and several gain components. The main source of these inputs is the subsystem Joystick block (refer to Figure 5). Irrelevant inputs such as buttons and POV signals are disregarded and channeled into a "Terminator" block. Analog signals, representing joystick movement, are passed through a mux and then adjusted using gain blocks. This adjustment ensures that deflection angles for aileron, elevator, and rudder are limited to a maximum value measured in radians.

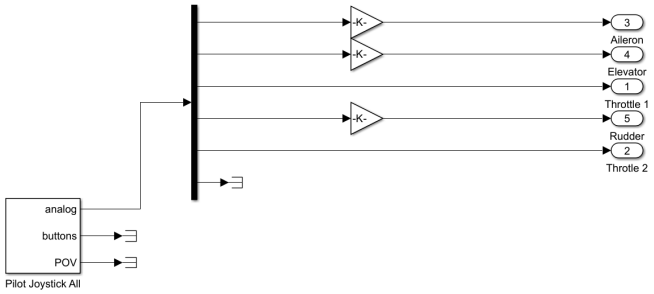

Figure 5: Input Subsystem Block for Joystick

5.0 DESIGN EVALUATION

Table 3: Name of Contributors to Each Part of the Design

Name	Contribution to design
Keith	Final MATLAB code and Simulink troubleshooting

Name	Contribution to design
Tasnim	MATLAB code for geodetic, kinematic equations
Mujtaba	MATLAB Code for Engine moments, Engine Forces, And force equations
Arman	MATLAB code for navigational equations, final code troubleshooting
Anna	Simulink block diagram, final MATLAB code troubleshooting

5.1 SIMULATION RESULTS

To run the simulation requires three files: two MATLAB files and one Simulink file. The Simulink file encompasses the necessary blocks for simulation, illustrated in Figures 3-5, along with all subsystems. The first MATLAB file holds the code utilized by the Simulink file for its calculations. Meanwhile, the second MATLAB file executes the Simulink file and utilizes its output data to generate the graphs depicted in Figures 9-20.

First, we would have to set up the Simulink in such a way that it takes the inputs and calculates the values in a function giving the outputs. To do this, we set up the Simulink as seen in Figure 4. In this subsystem, the main points of interest are the Interpreted MATLAB Function block and the integrator block as seen below:

Figure 6: Interpreted MATLAB Function block

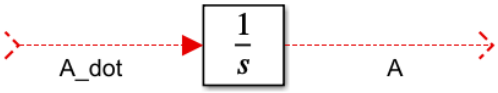

Figure 7: Integrator block

The Interpreted MATLAB Function block is where we will call in our MATLAB code which will do all the necessary calculations and output them to the integrator block. The integrator block is then used since we are dealing with rates in our code that change over time. The block integrates A_dot to A with the initial conditions set to the variable *init*. The output from the integrator is put through a demux giving the 12 outputs. It also looped back to the

Interpreted MATLAB Function block and what this does is update the variables of the aircraft in the air over time simulating its motion through the air. The out.simX block works by taking outputs from the integrator and putting them into the workspace to be plotted in graphs and for troubleshooting.

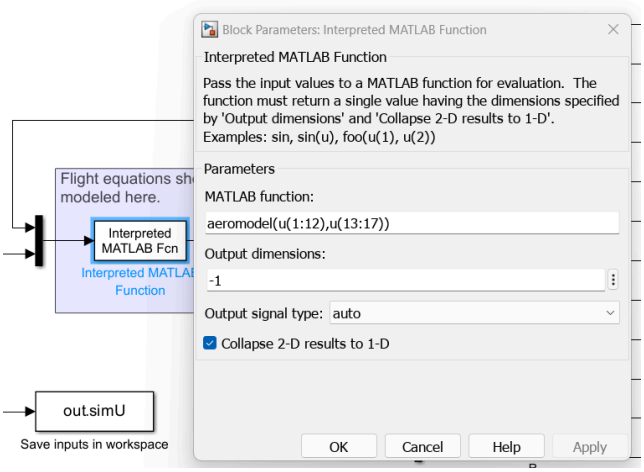

Figure 8: Calling the A_dot function into the Interpreted MATLAB Function block

The MATLAB code utilized by the Simulink file features a function named A_dot. This function initializes and employs all the elements from Sections 4.1 to 4.11 to compute the 12 outputs directed to the integrator block. Inputs are sourced from the data within the out.SimU block in the workspace. Each output of the function is meticulously indexed within the code to ensure Simulink effectively monitors their variations throughout the simulation loop. The function is saved as its own MATLAB file and called into the function block as seen in Figure 8.

The file *Runaeromodel* is used to run the simulation for a time *TF* and output the graphs seen in Figures 9-20. The code can be seen in Figures 33-35. The values are sent to the workspace via the out.simX block is what is used to plot the graphs vs time. This file also contains the variables *init* and *U_0*. The Variable *U_0* is used in a constant block as seen in Figure 3 for the initial state inputs. The variable init is initialized and input into the simulink file as mentioned earlier.

5.2 MATLAB GRAPHS

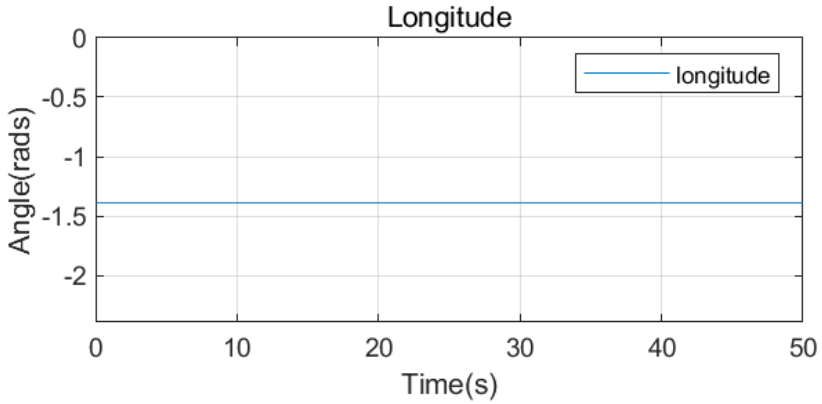

Figure 9: MATLAB graph for Longitude vs Time

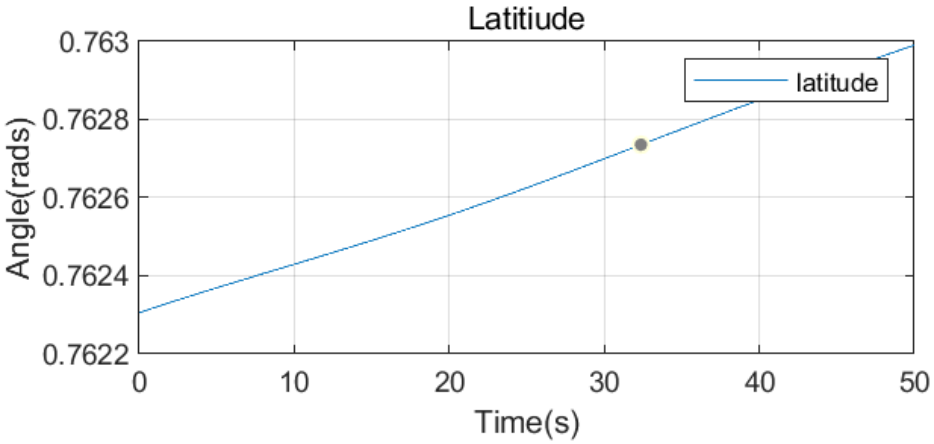

Figure 10: MATLAB graph for Latitude vs Time

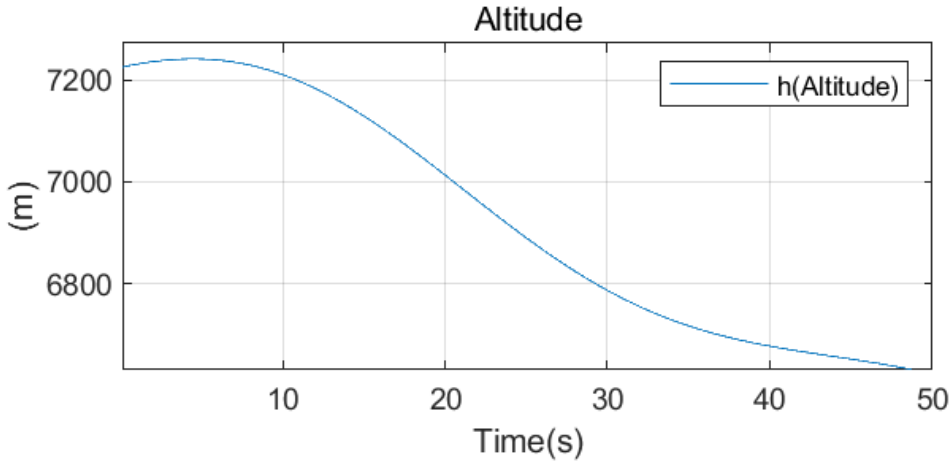

Figure 11: MATLAB graph for Altitude vs Time

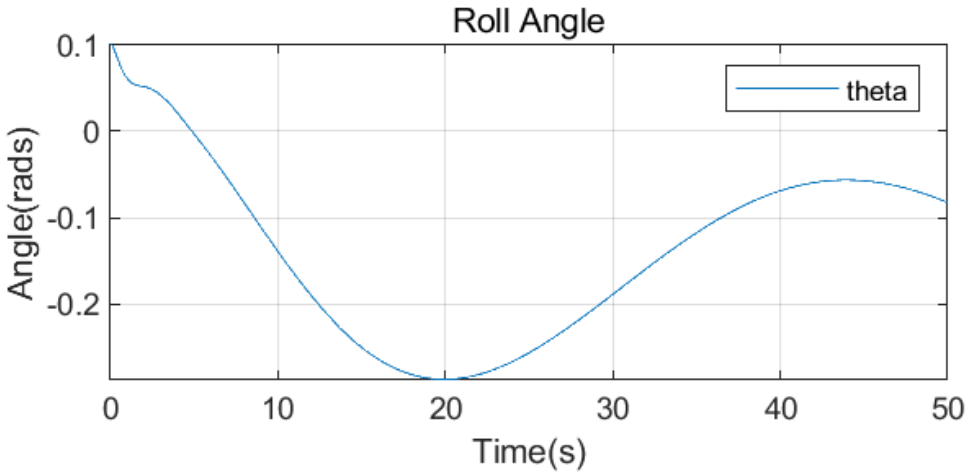

Figure 12: MATLAB graph for Roll Angle vs Time

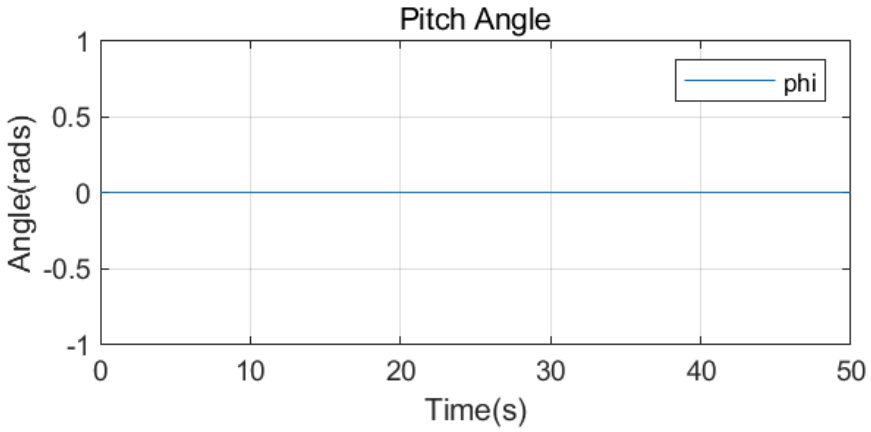

Figure 13: MATLAB graph for Pitch Angle vs Time

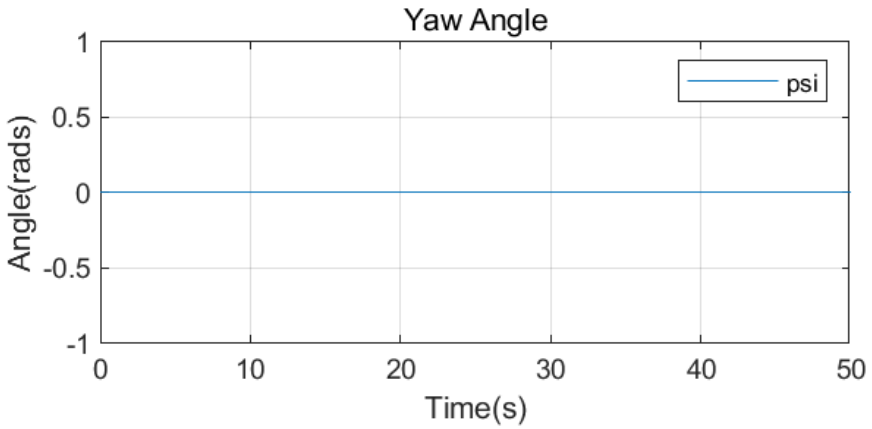

Figure 14: MATLAB graph for Yaw angle vs Time

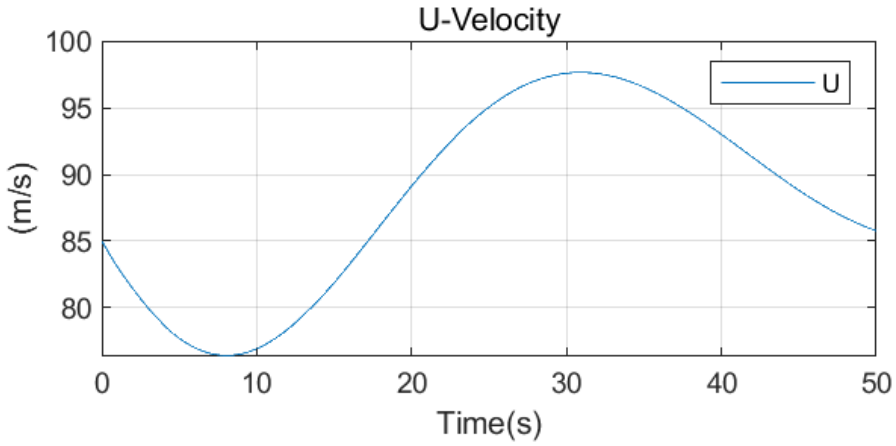

Figure 15: MATLAB graph for Velocity of the Longitudinal Speed vs Time

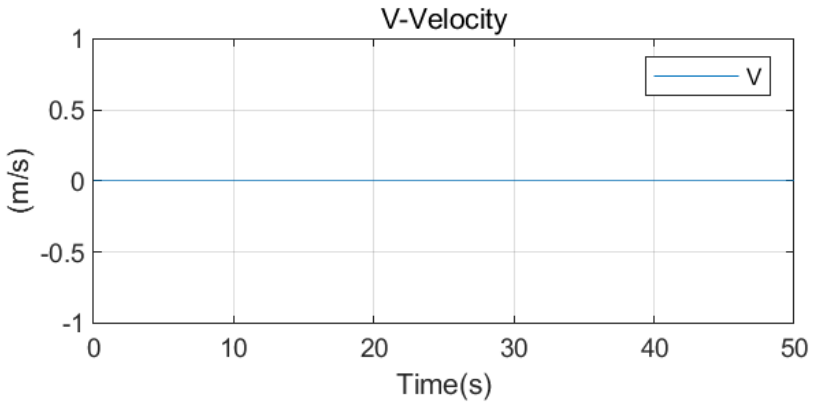

Figure 16: MATLAB graph for Velocity of the Lateral Speed vs Time

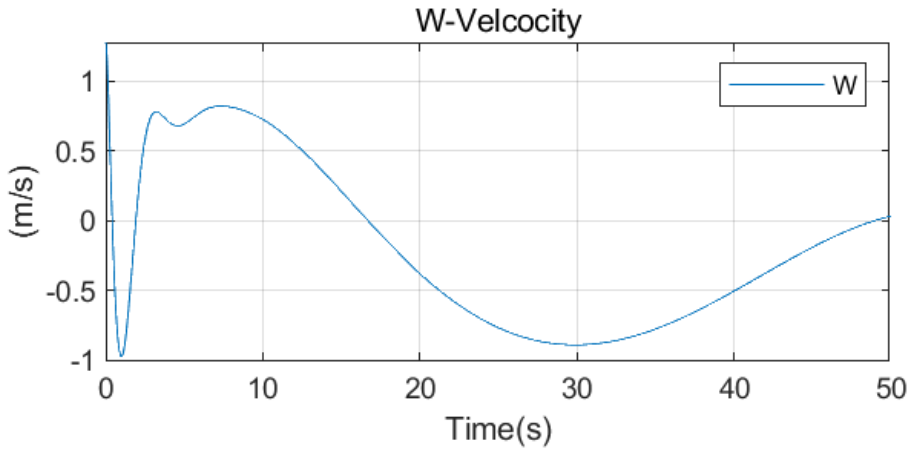

Figure 17: MATLAB graph for Velocity of the Vertical Speed vs Time

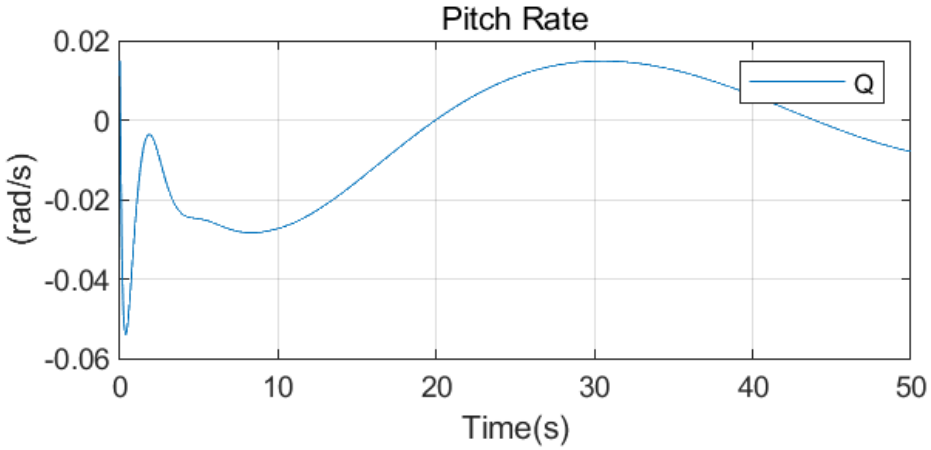

Figure 18: MATLAB graph for Pitch Rate vs Time

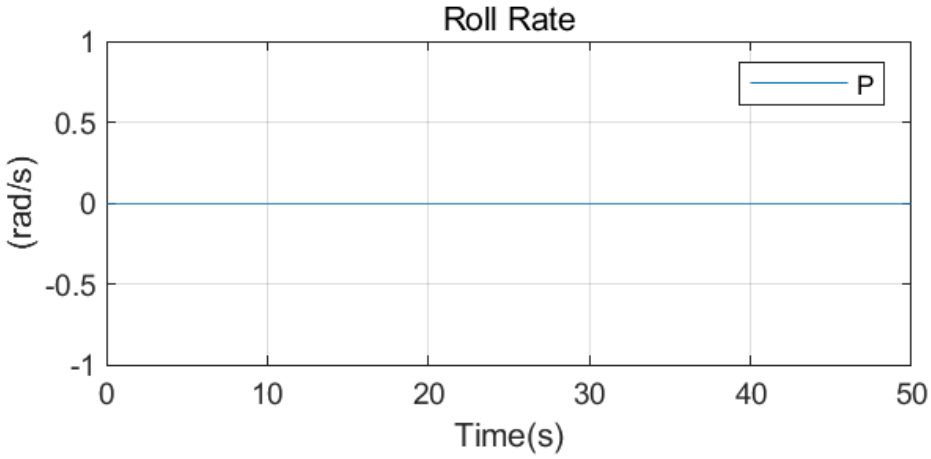

Figure 19: MATLAB graph for Roll Rate vs Time

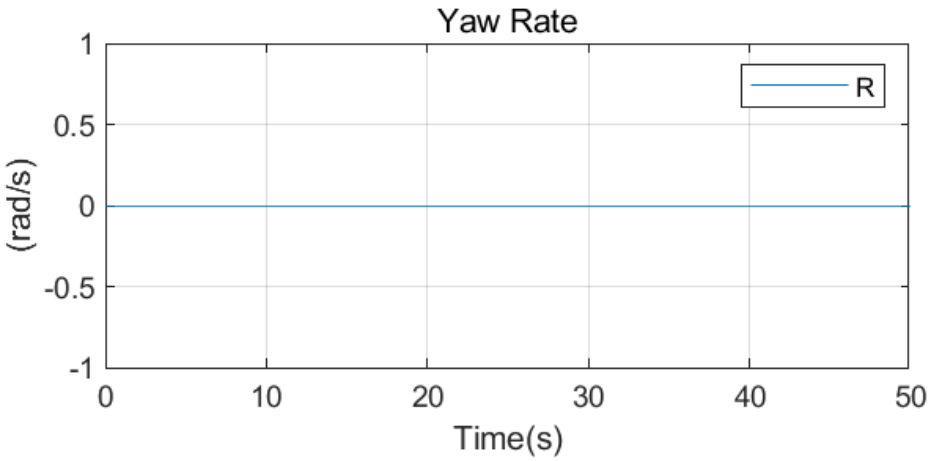

Figure 20: MATLAB graph for Yaw Rate vs Time

6.0 CONCLUSION

In summary, the report details the design process of a flight simulator resembling Boeing 737 and Airbus A320 aircraft. It covers objectives, execution steps, and outcomes. The simulator employs flat-earth equations processed through MATLAB and SIMULINK to simulate aircraft motion and joystick input. This software offers safe learning of aircraft behavior and pilot training, minimizing risk. It serves as a cost-effective, data-gathering tool for prolonged training compared to full-scale aircraft development.

The majority of the advantages of the software lie in the training of pilots. The technology allows pilots to familiarize themselves with the new aircraft controls, building muscle memory and learning the aircraft's behaviour. It also allows them to train for emergency situations and special case procedures without the pressure of risking lives or property. This in turn allows pilots to be comfortable with the aircraft once it comes into production as they are already familiar with the functions of the aircraft and how to deal with various scenarios. Unfortunately, the current simulator design comes with a few drawbacks. As stated earlier this design uses flat earth equations which does not simulate the environment physical aircrafts fly in. This means there may be a slight variation in certain portions of the simulator to the real world. Examples of this would be instrument readings, especially those of navigation and positioning. Reading the altitude or navigating with instruments may be slightly different in this simulator as compared to when flying on Earth. This drawback, however, is an opportunity to notice areas of improvement. The main one is modifying the equations to account for the earth's ellipsoid shape. This would produce more accurate data for the flight instruments and the aircraft's behavior in its environment. Further improvements could be made in the aircraft's controls, such as the introduction of flaps, reverse thrusters, and other controls that are common in an aircraft of this

size. Overall, this simulator provides a means to form a basic understanding of the aircraft's performance but can still be improved to be more accurate and detailed.

7.0 REFERENCES

[1] R. Faieghi, "L2 - Design Process, Establishing Objectives and Constraints," in *AER404 - Introduction to Aerospace Engineering Design*, 2024, https://courses.torontomu.ca/d2l/le/content/853745/viewContent/5611240/View**.**

[2] R. Faieghi, "L9 - Flight Dynamics in Simple Terms Annotated," in *AER404 - Introduction to Aerospace Engineering Design*, 2024, https://courses.torontomu.ca/d2l/le/content/853745/viewContent/5669171/View.

[3] R. Faieghi, "L10 - Simulink Pro Tips Annotated," in *AER404 - Introduction to Aerospace Engineering Design*, 2024, https://courses.torontomu.ca/d2l/le/content/853745/viewContent/5669996/View.

[4] "Project 2 Supplementary Material," in *AER404 - Introduction to Aerospace Engineering Design,* 2024, https://courses.torontomu.ca/d2l/le/content/853745/viewContent/5662808/View.

8.0 APPENDIX

MATLAB Code:

```
function A_dot = aeromodel(X,inputs)

%% VARIABLES
longitude = X(1);
latitude = X(2);
h = X(3);
phi = X(4);
theta = X(5);
psi = X(6);
U = X(7);
V = X(8);
W = X(9);
P = X(10);
Q = X(11);
R = X(12);

```

Figure 21: Matlab Aircraft Variables

```
%% INPUTS
delta_T_1 = inputs(1); % Engine 1 throttle angle
delta_T_2 = inputs(2); % Engine 2 throttle angle
delta_A = inputs(3); % Aileron deflection angle
delta_E = inputs(4); % Elevator deflection angle
delta_R = inputs(5); % Rudder deflection angle

```

Figure 22: Matlab Joystick Inputs

```
%% CONSTANTS
                % Mass
 m= 120000;
              % Inertia_Matrix
J = m*[40.07,0,-2.0923; 0, 64, 0; -2.0923, 0, 99.92];
            % Mean aerodynamic chord
c_bar = 6.6;
     % Coordinates of aircrafts center of mass
cm = [0.23*c_bar; 0; 0.1*c_bar];

    % Coordinates of aircrafts aerodynamic center
% ac = [0.12*c_bar; 0; 0];

     % Coordinates of engine 1 thrust application point
apt1 = [0;-7.94;-1.9];
      % Coordinates of engine 2 thrust application point
apt2 = [0;7.94;-1.9];
    % Wing platform area
S = 260;
    % Wing platform area of tail
S_t = 64;
   % Generalized length of wings
l = 6.6;
   % Distance between the aircraft cm and the ac of the tail
l_t = 24.8;
  % Angle of attack when lift is zero
alpha_0 = deg2rad(-11.5);
   % Air Density
Rho = 1.225;
   % Acceleration due to gravity
g = 9.81;
   % Earth equitorial radius
a = 6378137;
  % Earths eccentricity
e = 0.081819190842622;
```

Figure 23: Matlab Aircraft Constants

```
%% AERODYNAMCIS
% Airspeed
V_T = sqrt((U^2) + (V^2) + (W^2));
%Dynamic Pressure
q_bar = 0.5*Rho*(V_T^2);

%Angle of Attack
alpha = atan(W/U);
% Sideslip angle
beta_ = asin(V/V_T);

% Downwash angle of the wings
eppsilon = 0.25*(alpha-alpha_0);

% Angle of attack of the tail
alpha_t = alpha - eppsilon + delta_E + 1.3*Q*(l_t/V_T);

% Aerodynamic Coefficients
C_L_wb = 5.5*(alpha - alpha_0);
C_L_t = 3.1*(S_t/S)*alpha_t;
% Lift Coefficient
C_L = C_L_wb + C_L_t;

% Drag Coefficient
C_D = 0.13 + 0.07*((5.5*alpha + 0.654)^2);

% Crosswind Coefficient
C_C = -1.6*beta_ + 0.24*delta_R;

% Rolling moment Coefficient
C_l = -1.4*beta_ - 11*(l/V_T)*P + 5*(l/V_T)*R - 0.6*delta_A + 0.22*delta_R;

% Pitching moment Coefficient
C_m = -0.59 - 3.1*((S_t*l_t)/(S*l))*(alpha-eppsilon)-(4.03)*((S_t*(l_t^2))/(S*(l^2)))*(l/V_T)*(Q) - (3.1)*((S_t*l_t)/(S*l))*delta_E;

% Yawing moment coeficient
C n = (1 - 3.8179*alpha)*beta  + 1.7*(l/V T)*P - 11.5*(l/V T)*R - 0.63*delta R;
```

Figure 24: Matlab Aerodynamic Coefficients

```
% Aerodynamic Forces

D = q_bar*S*C_D;
C = q_bar*S*C_C;
L = q_bar*S*C_L;

% Resultant Forces
X_A = -D*cos(alpha)*cos(beta_) + C*cos(alpha)*sin(beta_)+L*sin(alpha);
Y_A = -D*sin(beta_) - C*cos(beta_);
Z_A = -D*sin(alpha)*cos(beta_) + C*sin(alpha)*sin(beta_)-L*cos(alpha);

Vector_forces = [X_A ; Y_A ; Z_A];
```

Figure 25: Matlab Aerodynamic and Resultant Forces

```


% Aerodynamic moments for simulink aswell
l_prime_A = q_bar*c_bar*S*C_l;
m_prime_A = q_bar*c_bar*S*C_m;
n_prime_A = q_bar*c_bar*S*C_n;

% Calculation
Vector_moment_prime = [l_prime_A; m_prime_A; n_prime_A];
Vector_constants = [0.11*c_bar; 0 ; 0.1*c_bar];

Actual_Aero_Moments  = Vector_moment_prime +  cross(Vector_forces,Vector_constants);

l_A = Actual_Aero_Moments(1);
m_A = Actual_Aero_Moments(2);
n_A = Actual_Aero_Moments(3);

```

Figure 26: Matlab Aerodynamic Moments and Calculations

```
%% ENGINE EQUATIONS
% Engine forces
XT1 = delta_T_1*m*g;
XT2 = delta_T_2*m*g;
YT = 0;
ZT = 0;
XT = XT1 + XT2;

% Engine moments
Engine1_moment1 = [XT1;0;0];
Engine1_moment2 = cm - apt1;
Engine2_moment1 = [XT2;0;0];
Engine2_moment2 = cm - apt2;

Engine1_total_moment = cross(Engine1_moment2,Engine1_moment1);
Engine2_total_moment = cross(Engine2_moment2,Engine2_moment1);

l_Ti = Engine1_total_moment(1) + Engine2_total_moment(1);
m_Ti = Engine1_total_moment(2) + Engine2_total_moment(2);
n_Ti = Engine1_total_moment(3) + Engine2_total_moment(3);

```

Figure 27: Matlab Engine Equations

```
%% FORCE EQUATIONS

U_dot = R*V - Q*W - g*sin(theta) + (XT+X_A)/m;
V_dot  = -R*U + P*W + g*sin(phi)*cos(theta) + (YT+Y_A)/m;
W_dot  = Q*U - P*V + g*cos(phi)*cos(theta) + (ZT+Z_A)/m;

```

Figure 28: Matlab Force Equations

```
%% MOMENT EQUATIONS

% Moment equations

J_x = J(1,1);
J_y = J(2,2);
J_z = J(3,3);
J_xz = J(3,1);

Gamma = J_x*J_z - (J_xz^2);

Pdot = (J_xz*(J_x - J_y + J_z)*P*Q-(J_z*(J_z-J_y) + (J_xz^2))*Q*R+J_z*(l_A+l_Ti)+J_xz*(n_A +n_Ti))/Gamma;
Qdot = ((J_z-J_x)*P*R-J_xz*((P^2)-(R^2))+ m_A + m_Ti)/J_y;
Rdot = (((J_x-J_y)*J_x+J_xz^2)*P*Q-J_xz*(J_x-J_y+J_z)*Q*R+J_xz*(l_A+l_Ti)+J_x*(n_A+n_Ti))/Gamma;

```

Figure 29: Matlab Moment Equations

```
%% KINEMATIC EQUATIONS

phi_dot = P + (tan(theta))*(Q*sin(phi) + R*cos(phi));
theta_dot = Q*cos(phi) - R*sin(phi);
psi_dot = (Q*sin(phi) + R*cos(phi))/cos(theta);

```

Figure 30: Matlab Kinematic Equations

```
%% NAVIGATION EQUATIONS

P_ndot = U*cos(theta)*cos(psi)+V*(-cos(phi)*sin(psi)+sin(phi)*sin(theta)*cos(psi))+W*(sin(phi)*sin(psi)+cos(phi)*sin(theta)*cos(psi));
P_edot = U*cos(theta)*sin(psi)+V*(cos(phi)*cos(psi)+sin(phi)*sin(theta)*sin(psi))+W*(-sin(phi)*cos(psi)+cos(phi)*sin(theta)*sin(psi));
h_dot = U*sin(theta)-V*sin(phi)*cos(theta)-W*cos(phi)*cos(theta);

```

Figure 31: Matlab Navigation Equations

```
%% GEODECTIC COORDINATES EQUATIONS

% Meridian radius
M = a*(1-(e^2))/sqrt((1-(e^2)*sin(phi)^2)^3);

% Prime vertical Radius
N = a/sqrt(1-(e^2)*sin(phi)^2);

latitude_dot = P_ndot/(M + h);
longitude_dot = P_edot/((N + h)*cos(latitude));

A_dot = [longitude_dot;latitude_dot;h_dot;phi_dot;theta_dot;psi_dot;U_dot;V_dot;W_dot;Pdot;Qdot;Rdot];
end
```

Figure 32: Matlab Geodetic Coordinates Equations

```
clc
close all
clear

% Intialise the intial condition values
longitude0 = deg2rad(-79.625335);
latitude0 = deg2rad(43.676856);
altitude0 = 7224;
phi0 = 0;
theta0 = 0.1;
psi0 = 0;
U0 = 84.9905;
V0 = 0;
W0 = 1.2713;
P0 = 0;
Q0 = 0.0149;
R0 = 0;

%Put the initial condition values into an matrix that can be input into the
%the integrator block in the simulink file
init = [longitude0;latitude0;altitude0;phi0;theta0;psi0;U0;V0;W0;P0;Q0;R0];
```

Figure 33: Runaeromodel MATLAB code initializing initial conditions

```
% Initialise the input values for the Throttle (1 and 2), Aileron, Rudder
% and Elevator. Put into the constant block
deltaT10 = 0.0821;
deltaT20 = 0.0821;
deltaA0 = 0;
deltaE0 = -0.1780;
deltaR0 = 0;
% Put into the constant block U_0
U_0 = [deltaT10;deltaT20;deltaA0;deltaE0;deltaR0];

% Time to run the sumulation
TF = 50;

% Set parameters that adjust Joystick input. Can be seen in Gain blocks in Input subsystem
maxAileronDeflection = 25*(pi/180);
maxElevatorDeflection = 10*(pi/180);
maxRudderDeflection = 30*(pi/180);

%Runs the simulation
out = sim("AircraftModel.slx");
```

Figure 34: Runaeromodel MATLAB code initializing inputs

```
%Access to the arrays of data after simulation is done
t = out.simX.Time;

x1 = out.simX.Data(:,1);
x2 = out.simX.Data(:,2);
x3 = out.simX.Data(:,3);
x4 = out.simX.Data(:,4);
x5 = out.simX.Data(:,5);
x6 = out.simX.Data(:,6);
x7 = out.simX.Data(:,7);
x8 = out.simX.Data(:,8);
x9 = out.simX.Data(:,9);
x10 = out.simX.Data(:,10);
x11 = out.simX.Data(:,11);
x12 = out.simX.Data(:,12);
```

Figure 34: Runaeromodel MATLAB code extracting data from the simulation

```
%plot results
figure
subplot(4,3,1)
plot(t,x1)
legend('longitude')
title(x1,'Longitude')
ylabel('Angle(rads)')
xlabel('Time(s)')

grid on
subplot(4,3,2)
plot(t,x2)
legend('latitude')
title(x2,'Latitiude')
ylabel('Angle(rads)')
xlabel('Time(s)')
```

Figure 35: Runaeromodel example MATLAB code plotting the extracted outputs vs time