MECH 6091 Flight Control Systems

Stability And Control Analysis For Modified Cessna T-37A

Dec 10th, 2024

Team Member

Prathmesh Deepak Gondkar	**40266289**
Phillip Arab	**40318128**
Radhouen Khmiri	**40311091**

Course Instructor

Dr. Jonathan Liscouët

Department of Mechanical, Industrial and Aerospace Engineering (MIAE)

Introduction

This report outlines an aircraft's design, trim, and stability analysis based on the Cessna T-37A, with minor simplifications. The model derived prioritizes the 3 flight control surfaces with the most significant impact on the aircrafts trim and stability, these being the rudder deflection angle (δ_r), the aileron deflection angle (δ_a), and the elevator deflection angle (δ_e). The study presents trim conditions, static stability, dynamic stability, and a flight dynamics simulation, with the objective of producing a stable and controllable dynamic model. The model is built and presented in MATLAB and Simulink. Most of the physical aircraft properties and system parameters and coefficients are taken from a solution of the analysis of the Cessna T-37A.

The first objective was to produce a solution of the deflection positions of the flight control surfaces for steady state trim condition of the designed aircraft, both in cruise and one engine inoperable (OEI) conditions. The second objective was to create a linearized state space model of the aircrafts nonlinear flight dynamics. The third objective was to use this linearized state space model in closed loop feedback control to test dynamic flight stability in the presence of disturbances (wind).

The model built had limitations, particularly in its control as the system is multiple-input-multiple-output (MIMO). The resulting model during simulation could maintain cruise flight with no disturbances but could not stably converge to the desired velocities in the presence of disturbances. Future work would be to study and implement a better MIMO controller for this system.

Design Description

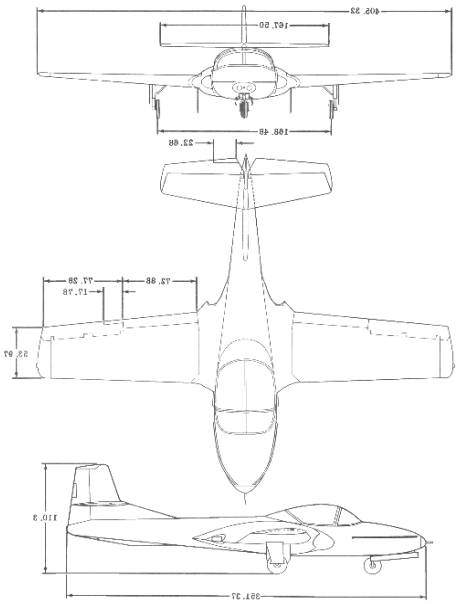


Figure: 1 Cessna T37-A

Specifications	Configuration
Length:8.92 m	**Wing:** Low-mounted, straight with moderate sweep.
Wingspan:10.29 m	**Tail**: Conventional design with vertical and horizontal stabilizers, rudder, and elevator
Height:2.79 m	**Fuselage:** Slender cylinder with cockpit nose and engine tail cones.
Wingarea:18.7 m^2	**Landing Gear:** Tricycle configuration.

Table (1):- Specifications and Configuration

Flight Envelope
Cruise Speed (v_max): 139.09 m/s
Altitude (h): 30,000 feet
Angle of Attack (α_1): 2 degrees
Mach Number (M): 0.459
Weight (W): 6360 lbs
Thrust (T1&T2):- 1,078 N
Minimum Stall Speed:- 30 to 36 m/s
Maximum Operating Speed (V_ne):- 154 m/s

Table (2):- Flight Envelope

Design Choice

This design was based on the Cessna T-37A with the main simplification being removal of any impact or control from the incidence angle of the horizontal tail (i_h). The design choice to omit the impact of the i_h was due to the redundancy in having control of both a i_h and an elevator, as they both affect mainly the pitch moment of the aircraft, and it is the elevator which is typically designed for active control and trim.

Primary Characteristics for Trim and Stability Analysis

Category	Parameter	Value
Flight Conditions	Velocity (V_∞)	139 m/s (cruise speed)
	Angle of Attack (α)	2° (0.0349 rad)
Mass & Gravity	Mass (m), Gravity (g)	2884.85 kg; 9.80665 m/s²
Wing	Wing Area (S)	55.47 m² (Trim); 16.7 m² (Stability)
	Wingspan (b)	1.524 m (Trim); 10.3 m (Stability)
	Mean Aerodynamic Chord (c)	1.66 m (Stability)
	Air Density (ρ)	0.3025 kg/m³
	Dynamic Pressure (Q)	4438 Pa (Stability)
Engine	Thrust (T)	1078 N/engine
	Engine Incidence Φ_T	5° (0.0873 rad)
	Engine Lateral Position (y_T)	2.5 m
	Drag Factor (k_{OEI})	1.15 (Inactive Engine)
	Engine Thrust Position (d_T)	0 m
Aerodynamic Coefficients	CD_0, CD_α	0.02(Drag) ;0.25(Drag)
	CL_0, CL_α	0.2 (Lift) ; 5.15 (Lift)
Pitching Moment	Cm_0, Cm_α, Cm_{de}	0.025 ; -0.7 ; -1.12
Side Force	$Cy_\beta, Cy_{da}, CY_{dr}$	-0.346 ; 0 ; 0.2 (Stability)
Roll	$Cl_\beta, Cl_{da}, Cl_{dr}$	-0.0944 ; 0.181 ; 0.015 (Stability)
Yaw	$Cn_\beta, Cn_{da}, Cn_{dr}$	0.1106 ; -0.0254 ; -0.0365 (Stability)
Elevator Deflection	δ_e	1.05° (-0.0183 rad)

Table (3):-Primary Characteristics for Trim and Stability Analysis

Aerodynamic Coefficients and Derivatives:

The aerodynamic coefficients and derivatives were sourced from reference [1]. For missing coefficients, assumptions were made based on typical aircraft behaviour. For instance, the drag coefficient $C_{D_{ih}}$ was considered negligible due to its minimal impact. Control surface derivatives were assumed to follow typical aircraft responses for small deflections, and the side force coefficient C_{y_β} was taken from reference [2], as side forces from sideslip are crucial for stability analysis.

Trim Analysis

Objectives:

This project aimed to determine the trim conditions and control deflections needed for stable flight during normal cruise and one-engine inoperative scenarios. This analysis aimed to calculate the required aileron and rudder deflections to maintain steady flight in both scenarios, ensuring lateral trim and stable flight under different operating conditions.

Methodology:

The trim analysis calculates control inputs needed for steady flight on a normal cruise and during one engine failure. Elevator deflection (δ_e) is determined using the equation , where $C_{m_{\delta_e}}$ is the elevator moment, and C_{m_0} and C_{m_α} are baseline moment coefficients. For engine failure, the thrust differential (ΔT) accounts for extra drag.

An aerodynamic coefficient matrix models the effects of aileron (δ_a) and rudder (δ_r) deflections on lateral forces and moments. The solution for sideslip angle (β), aileron, and rudder deflections is obtained using either Cramer's Rule or matrix inversion. Results are presented in degrees to provide a clear understanding of control requirements for trim in these scenarios. For further details, refer to the code provided in *Appendix A*.

Assumptions:

- Steady-state conditions were considered. In steady-state cruise, the aircraft was assumed to fly at a constant velocity with an angle of attack of 2°.
- A 15% drag increase was assumed based on industry standards and prior examples of the one-engine failure condition.
- Consideration of symmetric cruise conditions and steady asymmetric thrust during engine failure.

Results:

```
>> Trim_analysis
Elevator Deflection (delta_e) in Cruise Condition:
delta_e (deg): 0.028924
Trim needed to counteract the engine failure [beta; delta_a; delta_r] (deg) =
   18.3519
    6.4921
   31.7487
```

Figure 2:- Trim Analysis Results

Cruise Condition Results:

In steady cruise, the required elevator deflection (δ_e) for trim was found to be 0.0289°. This small deflection indicates that the aircraft is flying in a stable condition and requires minimal trim adjustments, as expected for level and steady flight.

One Engine Inoperable Results:

Under OEI conditions, significant control inputs were necessary to maintain lateral and directional trim. The results show a sideslip angle (β) of 18.35°, an aileron deflection (δ_a) of 6.49°, and a rudder deflection (δ_r) of 31.75°. These values demonstrate the need for coordinated rudder and aileron inputs to counteract the yaw and roll caused by asymmetric thrust.

Simulation Model and Static/Dynamic Stability

A MATLAB script (see *Appendix B1*) was employed to set up the state space model of the designed aircrafts dynamics. This script defines aircraft properties, system parameters, state output variables and inputs, functions describing forces and moments, and nonlinear equations of motion (EOM). The script then linearizes the system, in state space form, determines the transfer functions, and lastly defines the PID controller. All of this is connected to the Simulink Block Diagram of the system, which is configured as a closed loop feedback PID control model (*Appendix C1-C3*).

The output state of interest, [X], a vector of length 12. The input state [U] is a vector of length 4. They are defined as follows:

State Variable and Inputs Legend:

X1	x_e: X-Position relative to earth
X2	y_e: Y-Position relative to earth
X3	z_e: Z-Position relative to earth.
X4	Φ: Roll angle
X5	Θ: Pitch angle
X6	Ψ: Yaw angle
X7	u: Velocity in the body X-axis
X8	v: Velocity in the body Y-axis

X9	w: Velocity in the body Z-axis
X10	p: Roll rate
X11	q: Pitch rate
X12	r: Yaw rate
U1	ih: Thrust control input
U2	d_e: Elevator deflection (controls pitch)
U3	d_a: Aileron deflection (controls roll)
U4	d_r: Rudder deflection (controls yaw)

Table (4): - State Variables

Linearization:

The nonlinear EOMs for this system were taken from [1] and are linearized using the first order Taylor series approximation. This is done using the Jacobian () function in MATLAB which takes the first order partial derivative of the set of EOMs with respect to the output vector X and input vector U. The linearization was calculated at an equilibrium, or operating, point equivalent to that of the cruise condition of this aircraft. The approximate cruise condition for this aircraft is $\alpha = 2°$, $\beta = 0°$, $u = 139\frac{m}{s}$, $z_e = -9146\,m$, $T_1 = T_2 = 1078\,N$, and all other states and flight control inputs are 0. This is the same point which was analysed in the trim analysis.

This produces the A and B matrices for the state space model representation of differential model $\dot{X} = [A]X + [B]U$. The script then determines the set of transfer functions for this model using the A, B, C, and D matrices. Where C, and D satisfy the counterpart of the differential model, $y = [C]X + [D]U$. In this model, where there is no additional output *y* of interest, C and D are identity and empty matrices respectively. It is worth noting that this linearized state space model is MIMO, with 12 inputs, 4 outputs, and consequently 48 transfer functions existing between the relationship of each input and output.

PID Control:

The last section of the script produces the PID controller. Due to the nature of the MIMO system, this PID controller must have a control matrix of size 4x12 to relate the error state (of size 12) to the control input vector of size 4. More than this, as there are 3 inputs terms of PID control (Proportional, Integral, Derivative) this results in 4x12x3 = 144 potential parameters to be tuned. This is where the model begins to fail. Several different control matrices, named "aero" in the script were tested. This was done such that one control matrix of size 4x12 would describe the relative weightings of each relationship between the inputs and outputs, and then one K_p, K_i, or K_d gain would be applied to that control matrix accordingly to tune the relative weight between the proportional, integral, and derivative terms. Many of the input-output relationships are 0, or negligible. For example, i_h has no effect on the yaw moment, and so that relative weighting can be 0. Optimal control of a MIMO system is quite complex, and beyond the scope of this project. The objective was to implement a simple PID controller to produce sub-optimal but satisfactory control.

Simulink Block Diagram:

The main block diagram loop (*Appendix C1*) contains the state space model block, error summation block, and PID controller. This is the main feedback loop which controls the system. *Appendix C2* shows the section of the block diagram which uses output state vector [X] to include the disturbance of wind. A wind vector W_e is defined in the earth reference frame, this is multiplied by a rotation matrix R_{e2b} which is a custom function (*Appendix B2*) which calculates the rotation matrix from the earth to body reference frame using the roll, pitch, and yaw

angles. After subtracting the wind disturbance, this state is then fed into the error block of the main loop. The last block diagram section (*Appendix C3*) shows the relationship between the output state and the desired state to control. Only the translational velocities (u, v, and w) are given desired references to maintain, the other 9 reference states are fed their output state as to negate any control effort on these 9 states. This is because positions are not directly controllable, and there is no desired angular velocity to maintain for cruise flight.

Results:

Due to the amount of control parameters to tune, unfortunately no configuration could be found which would maintain stability and converge to the reference point in the presence of disturbances. The system was, however, able to demonstrate maintaining its initial cruise travel condition, and even steady climb, when subjected to no wind or disturbances. The resulting positions with respect to time for constant climb ($u_{des} = 139\ \left(\frac{m}{s}\right), v_{des} = 0\ , w_{des} = 10\left(\frac{m}{s}\right)$) are as follows:

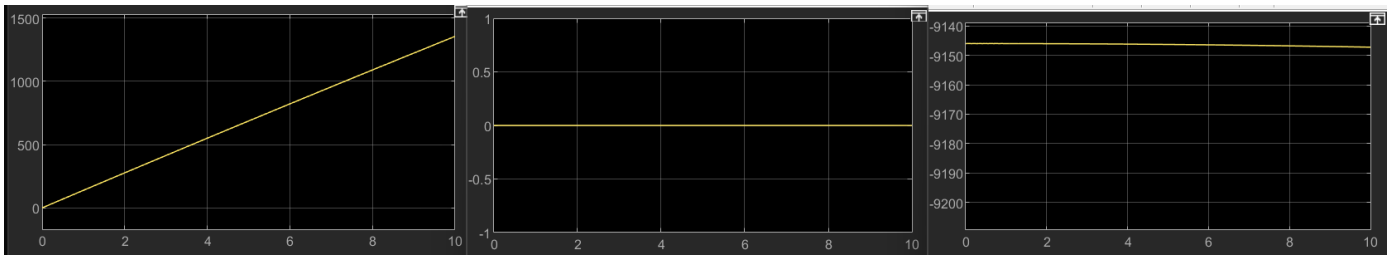


Figure 3: Steady Climb (0 Wind) xe(t), ye(t), ze(t)

Conclusion

The state space model representing flight dynamics of a modified Cessna T-37A were able to be linearized and analysed under trim and steady state conditions. Dynamic stability was not able to be conclusively shown due to limitation of manual PID tuning performance for MIMO systems.

Appendix

Appendix A: MATLAB Scripts for Trim Analysis

```
%parameters
phi = 0; % Roll angle (rad)
alpha = deg2rad(2); % Angle of attack (rad)
g = 9.80665; % Gravity (m/s^2)
m = 2884.8475; % MTOM (kg)
Vinf = 139; % Velocity (m/s)
rho = 0.3025; % Air density (kg/m^3)
S = 55.47; % Wing area (m^2)
b = 1.524; % Wingspan (m)
T = 1078; % Thrust per engine (N)
phi_T = deg2rad(5); % Engine incidence angle (rad)
k_OEI = 1.15; % Inactive engine drag factor (-)
y_T = 2.5; % Engine lateral position (m)
% Aerodynamic Coefficients
Cy_b = -0.346;
Cy_da = 0;
Cy_dr = 0.2;
Cl_b = -0.0944;
Cl_da = 0.18;
Cl_dr = 0.015;
Cn_b = 0.11;
Cn_da = -0.025;
Cn_dr = -0.036;
C_m0 =0.025;
Cm_alpha= -0.7;
Cm_delta_e= -1.12;
% ------------------delta_e for   Cruise Condition ------------------
delta_e = - (C_m0 + Cm_alpha * alpha) / Cm_delta_e;
delta_e_deg = rad2deg(delta_e);
disp('Elevator Deflection (delta_e) in Cruise Condition:');
%disp(['delta_e (rad): ', num2str(delta_e)]);
disp(['delta_e (deg): ', num2str(delta_e_deg)]);

% ------------------------- Engine failure -------------------------
% Dynamic Pressure
Q = rho * Vinf^2 / 2; % Dynamic pressure (Pa)
dT = 1.15 * T; % Thrust differential, Engine 1 inoperative (N)

% Aerodynamic Coefficient Matrix
Ac = [Cy_b Cy_da Cy_dr; Cl_b Cl_da Cl_dr; Cn_b Cn_da Cn_dr];

% Output vector for engine failure
Y = [0; -y_T * sin(phi_T + alpha) * dT / (Q * S * b);
       y_T * cos(phi_T + alpha) * dT / (Q * S * b)];

% Cramer's rule for engine failure
beta = det([Y(:), Ac(:,2:3)]) / det(Ac);
delta_a = det([Ac(:,1), Y(:), Ac(:,3)]) / det(Ac);
delta_r = det([Ac(:,1:2), Y(:)]) / det(Ac);
%disp('Cramer rule [beta; delta_a; delta_r] (deg) = ');
%disp(rad2deg([beta; delta_a; delta_r]));

% Inverse matrix method for engine failure
sol = inv(Ac) * Y;
disp('Trim needed to counteract the engine failure [beta; delta_a; delta_r] (deg) = ');
disp(rad2deg(sol));
```

Appendix B: MATLAB Scripts for Stability and Dynamic Analysis

Appendix B1

```
% EOM Linearization for SS Model

% Aircraft Parameters
m = 2890;     % kg
g = 9.8;      % m/s^2
Ixx = 10820; % kg*m^2
Iyy = 4507; % kg*m^2
Izz = 15153; % kg*m^2
Ixz = 0; % kg*m^2
S = 16.7; % m^2
Q = 4438; % Pa
b = 10.3; % m
c = 1.66; % m
yT1 = b/4; % m
dT = 0; % m
phiT = 0; % rad

% Aerodyamic Coefficients
CD_0 = 0.02;
CD_alpha = 0.25;
CD_ih = 0;
CD_de = 0;
CY_beta = -0.346;
CY_da = 0;
CY_dr = 0.2;
CL_0 = 0.2;
CL_alpha = 5.15;
CL_ih = 0;
CL_de = 0.5;
Cl_beta = -0.0944;
Cl_da = 0.181;
Cl_dr = 0.015;
Cm_0 = 0.025;
Cm_alpha = -0.7;
Cm_ih = 0;
Cm_de = -1.12;
Cn_beta = 0.1106;
Cn_da = -0.0254;
Cn_dr = -0.0365;


% System Parameters
u_cruise = 139; % m/s
T1 = 1078; % N
T2 = T1; % N
alpha = deg2rad(2); % rad
beta = 0; % rad
z_cruise = -9146; % m

syms X [12 1] real
syms U [4 1] real

% State Variable and Inputs Legend
xe = X1;
ye = X2;
ze = X3;
phi = X4;
theta = X5;
psi = X6;
u = X7;
v = X8;
w = X9;
p = X10;
q = X11;
r = X12;

ih = U1;
de = U2;
da = U3;
dr = U4;

% Define external forces and moments
Fxb = -m*g*sin(theta+alpha) -Q*S*(CD_0+CD_alpha*alpha+CD_ih*ih+CD_de*de) + (T1+T2)*cos(phiT+alpha);
Fyb = m*g*cos(theta+alpha)*sin(phi) + Q*S*(CY_beta*beta+CY_da*da+CY_dr*dr) + 0;
Fzb = m*g*cos(phi)*cos(theta+alpha) -Q*S*(CL_0+CL_alpha*alpha+CL_ih*ih+CL_de*de) - (T1+T2)*sin(phiT+alpha);
Mxb = 0 + Q*S*b*(Cl_beta*beta+Cl_da*da+Cl_dr*dr) + yT1*sin(phiT+alpha)*(T2-T1);
Myb = 0 + Q*S*c*(Cm_0+Cm_alpha*alpha+Cm_ih*ih+Cm_de*de) + dT*(T1+T2);
Mzb = 0 + Q*S*b*(Cn_beta*beta+Cn_da*da+Cn_dr*dr) + yT1*cos(phiT+alpha)*(T2-T1);

% Nonlinear EOMs (Simplified for Ixz=Ixz=0)
xe_dot = u*cos(psi)*cos(theta)+v*(cos(psi)*sin(phi)*sin(theta)-cos(phi)*sin(psi))+w*(sin(phi)*sin(psi)+cos(phi)*cos(psi)*sin(theta));
ye_dot = u*cos(theta)*sin(phi)+v*(cos(phi)*cos(psi)+sin(phi)*sin(psi)*sin(theta))+w*(cos(phi)*sin(psi)*sin(theta)-cos(psi)*sin(phi));
ze_dot = -u*sin(theta)+v*cos(theta)*sin(phi)+w*cos(phi)*cos(theta);
phi_dot = p+q*sin(phi)*tan(theta)+r*cos(phi)*tan(theta);
theta_dot = q*cos(phi)-r*sin(phi);
psi_dot = q*(sin(phi)/cos(theta))+r*(cos(phi)/cos(theta));
u_dot = r*v-q*w + Fxb/m;
v_dot = p*w-r*u + Fyb/m;
w_dot = q*u-p*v + Fzb/m;
p_dot = -((Izz^2+Iyy*Izz)*q*r+Izz*Mxb)/(Ixx*Izz);
```

```
q_dot = ((Izz-Ixx)*p*r+Myb)/(Iyy);
r_dot = ((Ixx^2-Ixx*Iyy)*p*q+Ixx*Mzb)/(Ixx*Izz);


% Combine into 1 matrix
f_EOM = [xe_dot; ye_dot; ze_dot; phi_dot; theta_dot; psi_dot; u_dot; v_dot; w_dot; p_dot; q_dot; r_dot;];

% Equilibrium point
Xeq = [0; 0; 0; 0; 0; 0; u_cruise; 0; 0; 0; 0; 0;];
Ueq = [0; 0; 0; 0;];

% Compute Jacobians
A = jacobian(f_EOM, X);  % Partial derivative w.r.t. state variables X
B = jacobian(f_EOM, U);  % Partial derivative w.r.t. inputs U

% Evaluate at the equilibrium point
A_at_eq = subs(A, [X; U], [Xeq; Ueq]);
B_at_eq = subs(B, [X; U], [Xeq; Ueq]);

% Simplify/Evaluate
A_at_eq = double(A_at_eq);
B_at_eq = double(B_at_eq);

% Transfer Function
C_tf = eye(12);
D_tf = zeros(12, 4);
system = ss(A_at_eq, B_at_eq, C_tf, D_tf);
TF = tf(system);

% LQR Controller    (Couldnt get to work, didnt use)
% Q_lqr = eye(12);   % State cost matrix
% R_lqr = eye(4);    % Control cost matrix
% Calculate LQR gains
% [K_lqr,~,~] = lqr(A_at_eq, B_at_eq, Q_lqr, R_lqr);
% disp(K_lqr);

% PID Controller

aero = [CD_ih 0 CL_ih 0 Cm_ih 0 CD_ih 0 CL_ih 0 Cm_ih 0;    % U1 ih
        CD_de 0 CL_de 0 Cm_de 0 CD_de 0 CL_de 0 Cm_de 0;    % U2 de
        0 CY_da 0 Cl_da 0 Cn_da 0 CY_da 0 Cl_da 0 Cn_da;    % U3 da
        0 CY_dr 0 Cl_dr 0 Cn_dr 0 CY_dr 0 Cl_dr 0 Cn_dr];   % U4 dr
      % xe   ye  ze  phi  theta  psi  u  v  w  p  q  r

aero0 = -1*[0 0 0 0 0 0 CD_ih 0 CL_ih 0 Cm_ih 0;    % U1 ih
            0 0 0 0 0 0 CD_de 0 CL_de 0 Cm_de 0;    % U2 de
            0 0 0 0 0 0 0 CY_da 0 Cl_da 0 Cn_da;    % U3 da
            0 0 0 0 0 0 0 CY_dr 0 Cl_dr 0 Cn_dr];   % U4 dr
          % xe ye ze phi theta psi u v w p q r

aero1 = [1 1 1 1 1 1 1 1 1 1 1 1;   % U1 ih
         1 1 1 1 1 1 1 1 1 1 1 1;    % U2 de
         1 1 1 1 1 1 1 1 1 1 1 1;    % U3 da
         1 1 1 1 1 1 1 1 1 1 1 1;];  % U4 dr
        % xe ye ze phi theta psi u v w p q r

aero2 = [1 0 1 0 2 0 1 0 1 0 2 0;   % U1 ih
         1 0 1 0 2 0 1 0 1 0 2 0;    % U2 de
         0 1 0 3 0 0 1 0 0 3 0 0;    % U3 da
         2 2 0 0 0 3 1 1 0 0 0 3;];  % U4 dr
        % xe ye ze phi theta psi u v w p q r

aero3 = -1*[0 0 0 0 0 0 1 0 1 0 2 0;   % U1 ih
         0 0 0 0 0 0 1 0 1 0 2 0;    % U2 de
         0 0 0 0 0 0 1 0 0 3 0 0;    % U3 da
         0 0 0 0 0 0 1 1 0 0 0 3;];  % U4 dr
        % xe ye ze phi theta psi u v w p q r

Kp = 1*aero0;
Ki = 0.1*aero0;
Kd = -0.001*aero0;


% Simulation
X_ic = [0; 0; z_cruise; 0; 0; 0; u_cruise; 0; 0; 0; 0; 0];
ourSim = sim("AircraftControlSystem");



%{
% Plotting
% Get actual and reference arrays of x, y, z from simulink model
ourSim = sim("AircraftControlSystem");
xpos = ourSim.xposition;
ypos = ourSim.yposition;
zpos = ourSim.zposition;
xr = ourSim.xref;
yr = ourSim.yref;
zr = ourSim.zref;
```

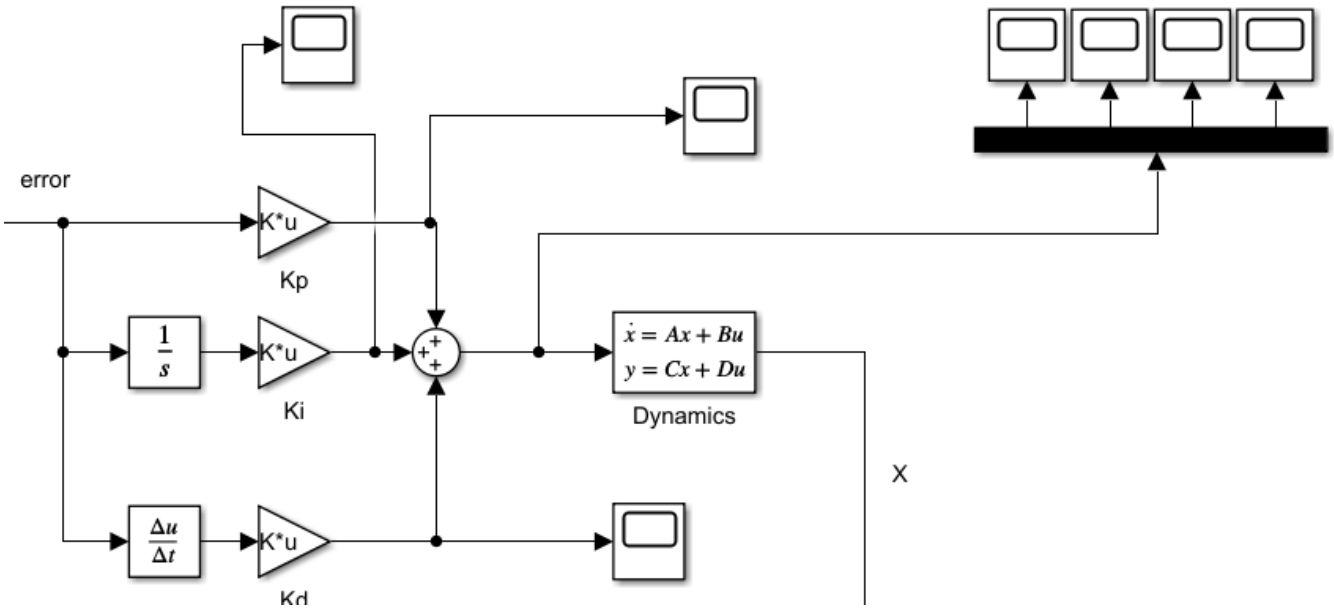
```
180
181   % Plot Results
182   figure;
183   plot3(xpos,ypos,zpos,'b-');
184   hold on;
185   plot3(xr,yr,zr,'r--');
186   grid on;
187   xlabel('X Position (m)');
188   ylabel('Y Position (m)');
189   zlabel('Z Position (m)');
190   title('3D Trajectory');
191   legend('Actual Trajectory','Desired Trajectory');
192   hold off;
193
194   %}
```

Appendix B2

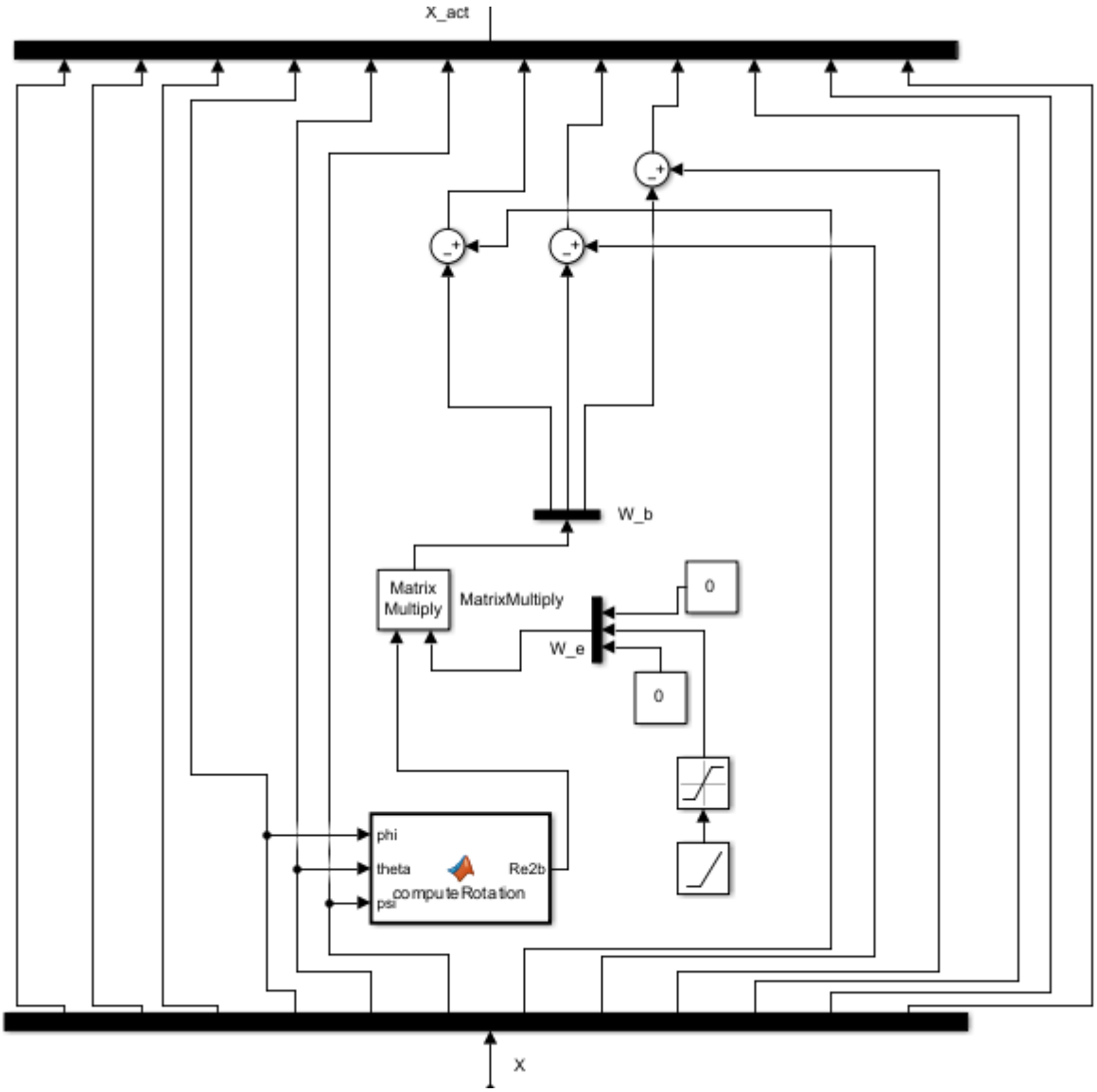
```
1   function Re2b = computeRotation(phi, theta, psi)
2       cphi = cos(phi);
3       sphi = sin(phi);
4       ctheta = cos(theta);
5       stheta = sin(theta);
6       cpsi = cos(psi);
7       spsi = sin(psi);
8
9       Re2b = [cpsi*ctheta, cpsi*stheta*sphi - spsi*cphi, cpsi*stheta*cphi + spsi*sphi;
10          spsi*ctheta, spsi*stheta*sphi + cpsi*cphi, spsi*stheta*cphi - cpsi*sphi;
11          -stheta,      ctheta*sphi,                 ctheta*cphi];
12  end
13
14
```

Appendix C: Simulink Block Diagram

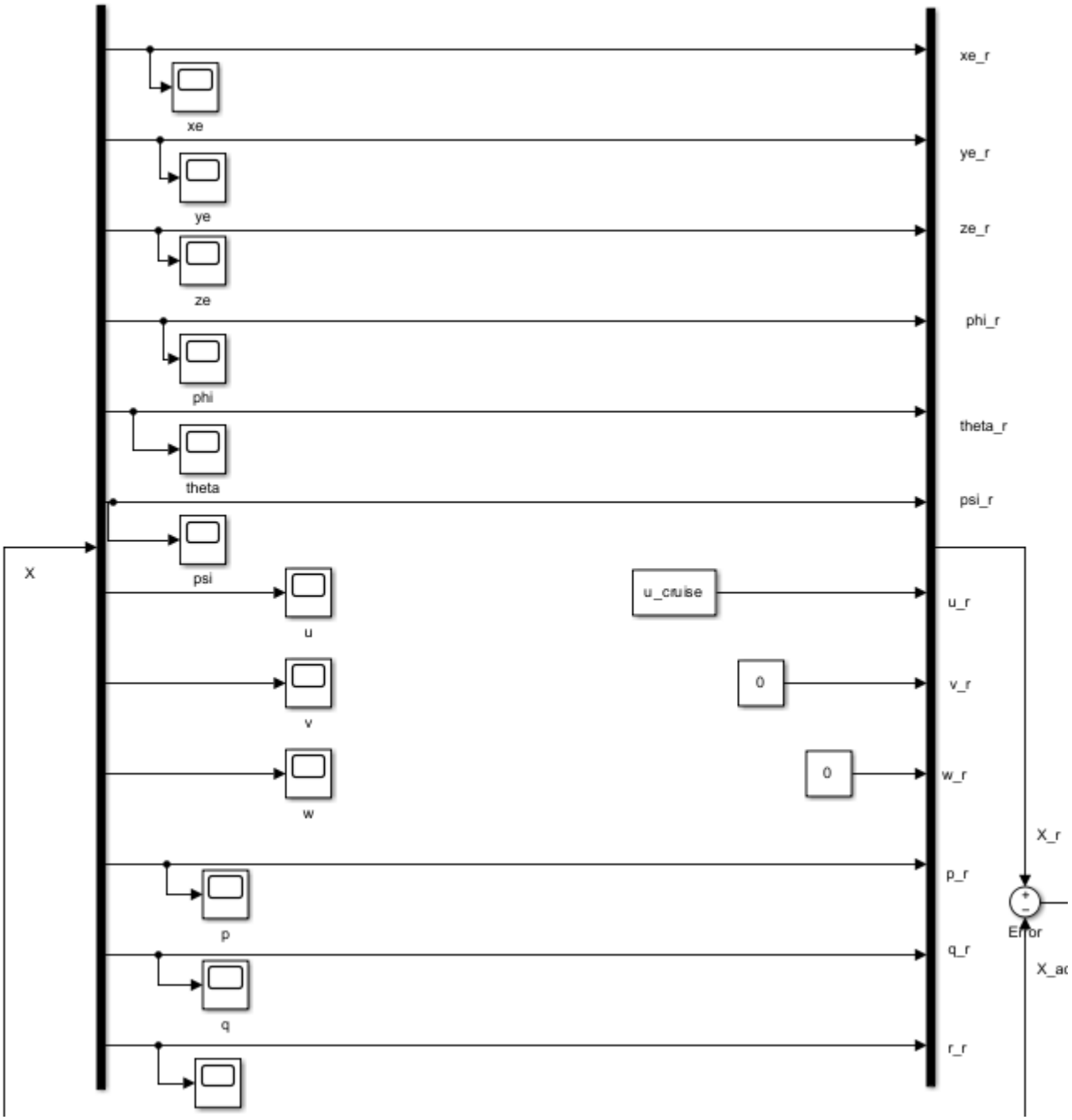
Appendix C1



Appendix C2


X_act
W_b
Matrix Multiply
MatrixMultiply
W_e
0
0
phi
theta
psi
computeRotation
Re2b
X

Appendix C3


xe_r
xe
ye_r
ye
ze_r
ze
phi_r
phi
theta_r
theta
psi_r
psi
X
u_cruise
u_r
u
0
v_r
v
0
w_r
w
X_r
p_r
p
Error
q_r
q
r_r

References

1. **J. Liscouët, "Equations of Motion," Lecture slides, Mech-6091, Concordia University, 2024.**
2. **J. Roskam, *Airplane Flight Dynamics and Automatic Flight Controls*, Part I and II. Lawrence, KS: DARcorporation, 2001.**