Problem 1

```
disp('');

function [aero_force, aero_moment] =
AeroForcesAndMoments_BodyState_WindCoeffs(aircraft_state, aircraft_surfaces,
wind_inertial, density, aircraft_parameters)
% Takes as input the aircraft state, the control input vector,
% the inertial wind velocity in inertial coordinates, the air density, and
the aircraft parameters
% structure and returns the aerodynamic force and moment acting on the
aircraft expressed in body coordinates
%
% Inputs:
%   aircraft_state -> full 12x1 aircraft state
%   aircraft_surfaces -> control input vector [de, da, dr, dt]
%   wind_inertial -> inertial wind velocity in inertial coordinates
%   density -> air density
%   aircraft_parameters -> aircraft parameter structure
% Output:
%   aero_force -> aerodynamic force in body coordinates, includes
%   propulsion
%   aero_moment -> aerodynamic moment in body coordinates
%
% Author: Thomas Dunnington
% Date Modified: 9/10/2024

% State parameters
position = aircraft_state(1:3);
euler_angles = aircraft_state(4:6);
velocity_inertial = aircraft_state(7:9);
angular_velocity = aircraft_state(10:12);

% Input parameters
de = aircraft_surfaces(1);
da = aircraft_surfaces(2);
dr = aircraft_surfaces(3);
dt = aircraft_surfaces(4);

% Wind angles
velocity_air_relative = velocity_inertial -
TransformFromInertialToBody(wind_inertial, euler_angles);
wind_angles = AirRelativeVelocityVectorToWindAngles(velocity_air_relative);
airspeed = wind_angles(1);
beta = wind_angles(2);
alpha = wind_angles(3);

% Dynamic Pressure
rho = density;
Q = 0.5*rho*airspeed^2;

% Nondimensional rates
```

```
phat = (angular_velocity(1)*aircraft_parameters.b)/(2*airspeed);
qhat = (angular_velocity(2)*aircraft_parameters.c)/(2*airspeed);
rhat = (angular_velocity(3)*aircraft_parameters.b)/(2*airspeed);

% Lift and Drag calculations
CL = aircraft_parameters.CL0 + aircraft_parameters.CLalpha*alpha +
aircraft_parameters.CLq*qhat + aircraft_parameters.CLde*de;
CD = aircraft_parameters.CDmin + aircraft_parameters.K*(CL -
aircraft_parameters.CLmin)^2;

% Aerodynamic Force Coefficients
CX = -CD*cos(alpha) + CL*sin(alpha);
CY = aircraft_parameters.CYbeta*beta + aircraft_parameters.CYp*phat +
aircraft_parameters.CYr*rhat + aircraft_parameters.CYda*da +
aircraft_parameters.CYdr*dr;
CZ = -CD*sin(alpha) - CL*cos(alpha);

% Propulsive Force Coefficient
CT = 2*aircraft_parameters.Sprop/aircraft_parameters.S *
aircraft_parameters.Cprop * ...
    dt/airspeed^2 * (airspeed + dt*(aircraft_parameters.kmotor -
airspeed))*(aircraft_parameters.kmotor - airspeed);

% Moment Coefficients
Cl = aircraft_parameters.Clbeta*beta + aircraft_parameters.Clp*phat +
aircraft_parameters.Clr*rhat + aircraft_parameters.Clda*da +
aircraft_parameters.Cldr*dr;
Cm = aircraft_parameters.Cm0 + aircraft_parameters.Cmalpha*alpha +
aircraft_parameters.Cmq*qhat + aircraft_parameters.Cmde*de;
Cn = aircraft_parameters.Cnbeta*beta + aircraft_parameters.Cnp*phat +
aircraft_parameters.Cnr*rhat + aircraft_parameters.Cnda*da +
aircraft_parameters.Cndr*dr;

% Aero Forces
X = aircraft_parameters.S*Q*CX;
Y = aircraft_parameters.S*Q*CY;
Z = aircraft_parameters.S*Q*CZ;

% Propulsive Force
T = aircraft_parameters.S*Q*CT;

% Aero Moments
L = aircraft_parameters.b*aircraft_parameters.S*Q*Cl;
M = aircraft_parameters.c*aircraft_parameters.S*Q*Cm;
N = aircraft_parameters.b*aircraft_parameters.S*Q*Cn;

% Return force and moments
aero_force = [X + T; Y; Z];
aero_moment = [L; M; N];
end

function [aircraft_forces, aircraft_moments] =
AircraftForcesAndMoments(aircraft_state, aircraft_surfaces, wind_inertial,
density, aircraft_parameters)
```

```
% Takes as input the aircraft state, the control input vector,
% the inertial wind velocity in inertial coordinates, the air density, and
the aircraft parameters
% structure and returns the total force and moment acting on the aircraft
expressed in body coordinates
%
% Inputs:
%   aircraft_state -> full 12x1 aircraft state
%   aircraft_surfaces -> control input vector [de, da, dr, dt]
%   wind_inertial -> inertial wind velocity in inertial coordinates
%   density -> air density
%   aircraft_parameters -> aircraft parameter structure
% Output:
%   aircraft_forces -> total aircraft force in body coordinates
%   aircraft_moments -> total aircraft moment in body coordinates
%
% Author: Thomas Dunnington
% Date Modified: 9/10/2024

% Euler Angles
euler_angles = aircraft_state(4:6);

% Aerodynamic Forces and Moments
[aero_force, aero_moment] =
AeroForcesAndMoments_BodyState_WindCoeffs(aircraft_state, aircraft_surfaces,
wind_inertial, density, aircraft_parameters);

% Gravity
fgE = [0; 0; aircraft_parameters.m * 9.81];
fgB = TransformFromInertialToBody(fgE, euler_angles);

% Total Forces and Moments
aircraft_forces = aero_force + fgB;
aircraft_moments = aero_moment;
end

function [xdot] =
AircraftEOM(time,aircraft_state,aircraft_surfaces,wind_inertial,aircraft_para
meters)
% Calculates the full nonlinear equations of motion of the aircraft given a
% current state
% Inputs:
%   time -> Time during simulation
%   aircraft_state -> full 12x1 aircraft state
%   aircraft_surfaces -> control input vector [de, da, dr, dt]
%   wind_inertial -> inertial wind velocity in inertial coordinates
%   aircraft_parameters -> aircraft parameter structure
% Output:
%   xdot -> Rate of change of the aircraft state
%
% Author: Thomas Dunnington
% Date Modified: 9/10/2024

% Allocation
```

```
xdot = ones(12,1);

% State parameters
position = aircraft_state(1:3);
euler_angles = aircraft_state(4:6);
velocity_inertial = aircraft_state(7:9);
angular_velocity = aircraft_state(10:12);

% Velocity
vel_inertial = TransformFromBodyToInertial(velocity_inertial, euler_angles);

% Euler rates
phi = euler_angles(1);
theta = euler_angles(2);
psi = euler_angles(3);

ang_vel_mat = [1, sin(phi)*tan(theta), cos(phi)*tan(theta);
    0, cos(phi), -sin(phi);
    0, sin(phi)*sec(theta), cos(phi)*sec(theta)];

euler_rates = ang_vel_mat*angular_velocity;

% Velocity components
p = angular_velocity(1);
q = angular_velocity(2);
r = angular_velocity(3);

% Define skew semetric matrix
omega_b = [0, -r, q; r, 0, -p; -q, p, 0];

% Define inertial matrix
IB = aircraft_parameters.inertia_matrix;

% Aircraft Forces
rho = stdatmo(-1*aircraft_state(3));
[aircraft_forces, aircraft_moments] =
AircraftForcesAndMoments(aircraft_state, aircraft_surfaces, wind_inertial,
rho, aircraft_parameters);

% Acceleration
vel_body_dot = -omega_b*velocity_inertial + 1/aircraft_parameters.m *
aircraft_forces;

% Time rate of change of angular velocity
omega_body_dot = inv(IB)*(-1*omega_b*(IB*angular_velocity) +
aircraft_moments);

% Full state derivatives
xdot = [vel_inertial; euler_rates; vel_body_dot; omega_body_dot];

end


function PlotSimulation(time, aircraft_state_array, control_input_array, col)
```

```
% Creates 6 plots from an aircraft simulation
% Inputs:
%   time -> Time during simulation
%   aircraft_state_array -> full nx12 aircraft state over time
%   control_input_array -> control input vector nx4
%   col -> plotting options
% Output:
%   6 plots of aircraft state and control variables over time
%
% Author: Thomas Dunnington
% Date Modified: 9/10/2024

%Plotting the inertial positions over time
fig = 1:6;
figure(fig(1));
subplot(3,1,1)
plot(time, aircraft_state_array(:,1), col, 'linewidth', 2);
hold on;
grid on;
ylabel('X position (m)');
title('Aircraft Positions');

subplot(3,1,2)
plot(time, aircraft_state_array(:,2), col, 'linewidth', 2);
hold on;
grid on;
ylabel('Y position (m)');

subplot(3,1,3)
plot(time, aircraft_state_array(:,3), col, 'linewidth', 2);
hold on;
grid on;
ylabel('Z position (m)');


xlabel('Time (s)');


%Plotting the euler angles over time
figure(fig(2));
subplot(3,1,1)
plot(time, aircraft_state_array(:,4), col, 'linewidth', 2);
hold on;
grid on;
ylabel('Roll (rad)');
title('Euler Angles');

subplot(3,1,2)
plot(time, aircraft_state_array(:,5), col, 'linewidth', 2);
hold on;
grid on;
ylabel('Pitch (rad)');

subplot(3,1,3)
```

```
plot(time, aircraft_state_array(:,6), col, 'linewidth', 2);
hold on;
grid on;
ylabel('Yaw (rad)');


xlabel('Time (s)');



%Plotting the inertial velocity in the body frame
figure(fig(3));
subplot(3,1,1)
plot(time, aircraft_state_array(:,7), col, 'linewidth', 2);
hold on;
grid on;
ylabel('uE (m/s)');
title('Inertial Velocities');

subplot(3,1,2)
plot(time, aircraft_state_array(:,8), col, 'linewidth', 2);
hold on;
grid on;
ylabel('vE (m/s)');

subplot(3,1,3)
plot(time, aircraft_state_array(:,9), col, 'linewidth', 2);
hold on;
grid on;
ylabel('wE (m/s)');


xlabel('Time (s)');


%Plotting the angular velocity
figure(fig(4));
subplot(3,1,1)
plot(time, aircraft_state_array(:,10), col, 'linewidth', 2);
hold on;
grid on;
ylabel('p (rad/s)');
title('Angular Velocities');

subplot(3,1,2)
plot(time, aircraft_state_array(:,11), col, 'linewidth', 2);
hold on;
grid on;
ylabel('q (rad/s)');

subplot(3,1,3)
plot(time, aircraft_state_array(:,12), col, 'linewidth', 2);
hold on;
grid on;
```

```
ylabel('r (rad/s)');


xlabel('Time (s)');



%Plotting each control input variable
figure(fig(5));
subplot(4,1,1)
plot(time, control_input_array(:,1), col, 'linewidth', 2);
hold on;
grid on;
ylabel('Elevator (rad)');
title('Control Inputs');

subplot(4,1,2)
plot(time, control_input_array(:,2), col, 'linewidth', 2);
hold on;
grid on;
ylabel('Aileron (rad)');

subplot(4,1,3)
plot(time, control_input_array(:,3), col, 'linewidth', 2);
hold on;
grid on;
ylabel('Rudder (rad)');

subplot(4,1,4)
plot(time, control_input_array(:,4), col, 'linewidth', 2);
hold on;
grid on;
ylabel('Throttle (frac)');

xlabel('Time (s)');


%Plotting the 3 Dimensional Path of the drone
figure(fig(6));
plot3(aircraft_state_array(:,1), aircraft_state_array(:,2),
-1*aircraft_state_array(:,3), col, 'linewidth', 2);
hold on;
grid on;

%Starting point
plot3(aircraft_state_array(1,1), aircraft_state_array(1,2),
-1*aircraft_state_array(1,3), 'p', 'markersize', 10, 'markerFaceColor',
'green', 'markerEdgeColor', 'green');

%Ending point
plot3(aircraft_state_array(end,1), aircraft_state_array(end,2),
-1*aircraft_state_array(end,3), 'p', 'markersize', 10, 'markerFaceColor',
'red', 'markerEdgeColor', 'red');
```

```
title('Aircraft Path');
xlabel('X Position (m)');
ylabel('Y Position (m)');
zlabel('Z Position (m)');


end
```

Published with MATLAB® R2023b