Table of Contents

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  ASEN 3128 HW 4
%  Author: Caleb Bristol
%  Date: 10 March, 2022
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Clear Workspace

```
clc
clear
close all;
```

Problem 1

True, see explanation below

Explanation

The only real consideration for equilibrium is whether or not the aircraft is accelerating or not. If there is a state in which the aircraft has a non-zero angular velocity and isn't accelerating, then this statement is true, because there is a non-zero angular velocity and the craft would be in equilibrium. This is the case for a spinning quadrotor. If it has an angular momentum in the r direction then this rotation could persist forever without impacting the way gravity affects the craft, which could allow the rotors to perfectly counteract the effects of gravity. The rotors would also have to be designed such that the control moment perfectly counteracts the aerodynamic moment caused by the craft's spin, but if implemented, this would result in an aircraft with a non-zero angular velocity vector that is in equilibrium. Therefore, the statement is true.

Problem 2 a)

This was a simple case of utilizing equations of motion.

Establish Problem Constraints

```
x_ = [100; 100; -1600; 0.20; -0.21; 0; 10; 9.3; -2.9; 0; 0; 0];
x = x_(1);
y = x_(2);
z = 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);
f_m = [0.17;0.17;0.17;0.17];
I_x = 5.8E-5;   %[kg*m^2]
I_y = 7.2E-5;   %[kg*m^2]
I_z = 1.0E-4;  %[kg*m^2]

% As pulled from the lab document:
%
% For Parrot Mamba Minidrone
g = 9.81;       %[m/s^2]
m = 0.068;      %[kg]
R = 0.060;      %[m]
k_m = 0.0024;   %[Nm/N]
nu = 1E-3;      %[N/(m/s)^2]
mu = 2E-6;      %[Nm/(rad/s)^2]
```

Determine Forces and Moments

The methodology for this section was to calculate the control forces and moments first, based off the velocity and angular velocity of the craft, then utilize the individual motor forces to calculate the control forces and moments.

Gravitational Force

```
    f_g = m*g;
```

Aerodynamic Forces and Moments

```
        % Aerodynamic Drag Force
        airspeed = norm(x_(7:9));

        D = nu * airspeed^2;

        f_aero = -D * x_(7:9) / norm(x_(7:9));

        % Aerodynamic Moments
        m_aero = -mu * norm(x_(10:12)) * x_(10:12);
```

Control Forces and Moments

```
        % Control Forces
        f_c = [0;0;-f_m(1)-f_m(2)-f_m(3)-f_m(4)];

        % Control Moments
        M = [-R/sqrt(2) -R/sqrt(2) R/sqrt(2) R/sqrt(2); ...
            R/sqrt(2) -R/sqrt(2) -R/sqrt(2) R/sqrt(2); k_m -k_m k_m -
k_m];

        m_c = M*f_m;
```

Display Results

```
    fprintf('PROBLEM 2 a): \n \n')
    fprintf('The gravitational force acting on the craft [N] was: \n')
    disp(f_g)
    fprintf('The aerodynamic forces acting on the craft [N] were: \n')
    disp(f_aero)
    fprintf('The aerodynamic moments acting on the craft [Nm] were:
 \n')
    disp(m_aero)
    fprintf('The control forces acting on the craft [N] were: \n')
    disp(f_c)
    fprintf('The control moments acting on the craft [Nm] were: \n')
    disp(m_c)
    fprintf('The force exerted by motors 1 through 4 [N] were: \n')
    disp(f_m)
```

```
PROBLEM 2 a):

The gravitational force acting on the craft [N] was:
    0.6671

The aerodynamic forces acting on the craft [N] were:
   -0.1396
   -0.1298
    0.0405

The aerodynamic moments acting on the craft [Nm] were:
     0
     0
     0

The control forces acting on the craft [N] were:
         0
         0
   -0.6800

The control moments acting on the craft [Nm] were:
     0
     0
     0

The force exerted by motors 1 through 4 [N] were:
    0.1700
    0.1700
    0.1700
    0.1700
```

Problem 2 b)

The craft is not in equilibrium. For further explanation, see the explanation section below.

Plug Into EOM

This is to verify the validity of part b), whether or not the craft is in equilibrium. If the craft is in equilibrium, the second half of the state vector should be zeros, as that would imply the craft is non-accelerating. This was done using a function from lab 3, which includes the equations of motion. While this function was originally intended for use with ODE45, it returns the state derivative vector so it is a perfect application in this problem.

Run State Derivative Function

```
dx_dt = nonlinearEOM(0,x_,g,m,nu,mu,f_c,m_c);
```

Display Results

```
fprintf('PROBLEM 2 b): \n \n')
fprintf('The full derivative state vector: \n')
disp(dx_dt)
```

```
PROBLEM 2 b):

The full derivative state vector:
    9.9876
    9.6908
    1.1119
         0
         0
         0
```

```
   -0.0080
   -0.0032
   -0.0014
         0
         0
         0
```

Explanation

Note that once plugged into the equations of motion, the dynamics of the craft suggest that it is seeing no change in its rotation, which is to be expected as there was no initial angular velocity and the rotor forces were balanced. However, due to the initial orientation of the craft, its translational velocity IS accelerating as can be seen from members 7 through 9 of the derivative state vector. Because this craft is accelerating, it is NOT in a state of equilibrium.

Problem 3

Solved on paper, see physical solutions

Problem 4

This involves the analysis of the quadrotor, with the inclusion of feedback control laws. The state space model and the poles of the system are to be determined. The natural frequency and damping ratios are to be found for the eigenvalues with imaginary poles, while the time constant is to be found for the eigenvalues with real poles. The state space model is shown on paper, see physical solutions.

Establish Variables

We know how the equation presented in the problem relates to the control gains based on the state space model. Therefore, we utilize them directly in solving for the eigenvalues

```
k_1 = 0.01;
k_2 = 0.05;
k_3 = 0.01;
k_4 = 0.01/k_3;
```

Calculate Eigenvalues

```
    A = [0 1 0 0;0 0 g 0;0 0 0 1;-k_4*k_3/I_x -k_3/I_x -k_2/I_x -k_1/
I_x];

    Eig = eig(A);
```

Compute Frequency, Damping, and Time Constant

We know which poles require which calculations simply by looking at if they contain imaginary values.

```
% Eigenvalues 1 and 2: Real
```

```
sigma_1 = real(Eig(1));
sigma_2 = real(Eig(2));

tau_1 = -1/sigma_1;
tau_2 = -1/sigma_2;

% Eigenvalues 3 and 4: Imaginary
sigma_3 = real(Eig(3));
sigma_4 = real(Eig(4));

d_3 = imag(Eig(3));
d_4 = imag(Eig(4));

omega_n_3 = sqrt(sigma_3^2 + d_3^2);
omega_n_4 = sqrt(sigma_4^2 + d_4^2);

chi_3 = sigma_3/omega_n_3;
chi_4 = sigma_4/omega_n_4;
```

Display Results

```
fprintf('PROBLEM 4: \n \n')
fprintf('The full set of eigenvalues 1-4: \n')
disp(Eig)
fprintf('Time Constant for eigenvalue 1 [s]: \n')
disp(tau_1)
fprintf('Time Constant for eigenvalue 2 [s]: \n')
disp(tau_2)
fprintf('Natural Frequency for eigenvalue 3 [rad/s]: \n')
disp(omega_n_3)
fprintf('Damping Coefficient for eigenvalue 3: \n')
disp(chi_3)
fprintf('Natural Frequency for eigenvalue 4 [rad/s]: \n')
disp(omega_n_4)
fprintf('Damping Coefficient for eigenvalue 4: \n')
disp(chi_4)
```

```
PROBLEM 4:

The full set of eigenvalues 1-4:
   1.0e+02 *

  -1.6732 + 0.0000i
  -0.0279 + 0.0000i
  -0.0115 + 0.0152i
  -0.0115 - 0.0152i

Time Constant for eigenvalue 1 [s]:
    0.0060

Time Constant for eigenvalue 2 [s]:
    0.3586
```

```
Natural Frequency for eigenvalue 3 [rad/s]:
    1.9039

Damping Coefficient for eigenvalue 3:
   -0.6049

Natural Frequency for eigenvalue 4 [rad/s]:
    1.9039

Damping Coefficient for eigenvalue 4:
   -0.6049
```

Functions

```
function dvar_dt = nonlinearEOM(t,var,g,m,nu,mu,Fc,Gc)
```

nonlinearEOM

Includes the equations for the non-linearized equations of motion, utilized in problem 3

Group 37 Members: -Caleb Bristol -Adam Pillari -Devon Paris -Kushal Kedia

Credit to Group 24 of Lab 2 for helping write this code

```
% INPUTS: t is scalar time
%         var is a column vector of the aircraft state
%         g is scalar gravity
%         m is scalar mass
%         nu is the scalar aerodynamic force coefficient
%         mu is the scalar aerodynamic moment coefficient
%         Fc is a column vector of Body-Frame Control Forces
%         Gc is a column vector of Body-Frame Control Moments

pos = var(1:3);
ang = var(4:6);
vel = var(7:9);
ang_vel = var(10:12);

airspeed = norm(vel);
omega_mag = norm(ang_vel);

I_B = [6.8E-5; 9.2E-5; 1.35E-4];

% Calculate position dynamics
pos_DCM =   [cos(ang(2))*cos(ang(3))
 sin(ang(1))*sin(ang(2))*cos(ang(3))-cos(ang(1))*sin(ang(3))
 cos(ang(1))*sin(ang(2))*cos(ang(3))+sin(ang(1))*sin(ang(3));
             cos(ang(2))*sin(ang(3))
 sin(ang(1))*sin(ang(2))*sin(ang(3))+cos(ang(1))*cos(ang(3))
 cos(ang(1))*sin(ang(2))*sin(ang(3))-sin(ang(1))*cos(ang(3));
             -sin(ang(2)) sin(ang(1))*cos(ang(2))
 cos(ang(1))*cos(ang(2))];
```

```
dpos_dt = pos_DCM*vel;

% Calculate angular dynamics
ang_DCM =   [1 sin(ang(1))*tan(ang(2)) cos(ang(1))*tan(ang(2));
             0 cos(ang(1)) -sin(ang(1));
             0 sin(ang(1))*sec(ang(2)) cos(ang(1))*sec(ang(2))];

dang_dt = ang_DCM*ang_vel;

% Calculate velocity dynamics

vel_term_1 =    [ang_vel(3)*vel(2)-ang_vel(2)*vel(3);
             ang_vel(1)*vel(3)-ang_vel(3)*vel(1);
             ang_vel(2)*vel(1)-ang_vel(1)*vel(2)];

vel_term_2 =    g * [-sin(ang(2));
                 cos(ang(2))*sin(ang(1));
                 cos(ang(2))*cos(ang(1))];

vel_term_3 =    -nu*airspeed/m * vel;

vel_term_4 =    Fc/m;

dvel_dt = vel_term_1 + vel_term_2 + vel_term_3 + vel_term_4;

% Calculate angular velocity dynamics

ang_vel_term_1 =    [(I_B(2)-I_B(3))*ang_vel(2)*ang_vel(3)/I_B(1);
                     (I_B(3)-I_B(1))*ang_vel(1)*ang_vel(3)/I_B(2);
                     (I_B(1)-I_B(2))*ang_vel(1)*ang_vel(2)/I_B(3)];

ang_vel_term_2 =    -mu*omega_mag*ang_vel./I_B;

ang_vel_term_3 =    Gc./I_B;

dangvel_dt = ang_vel_term_1 + ang_vel_term_2 + ang_vel_term_3;

dvar_dt = [dpos_dt;dang_dt;dvel_dt;dangvel_dt];

end
```

Published with MATLAB® R2020a