HW1: Modeling the Flying-Chardonnay

Work done by –

Akash Sharma

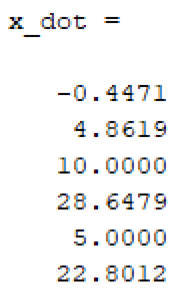
Yash Malandkar

The following MATLAB code generates the value of x_dot which is a solution of the function x_dot = f(x,u,w,drone).

Based on the proposed function signature, the elements of the x_dot vector are as follows.

x_dot = [vnd, vdd, thed, thedd, gamd, gamdd];

The final value of the x_dot vector is as follows (all values are in S.I. units).



```
x_dot =

   -0.4471
    4.8619
   10.0000
   28.6479
    5.0000
   22.8012
```

Based on the equations of motion for the Flying-Chardonnay System, the state-space model is created composed of the Π and **h** matrices. The final solution is obtained by vector multiplication of the inverse of Π and h matrices.

We neglect the reaction force on the drone from the cup and its lever system since it's not part of the state variables. The final MATLAB code is shown below.

```
clear all

clc

%% Creating a structure for the system's physical properties
drone = struct('drone_mass',1 , 'cup_mass',1  , 'cup_holder_length',1 ,
'propeller_arm',1 , 'moment_of_inertia',1 , ...
    'drag_coefficient',0.01 , 'gravity_acc',10 )

%% Initializing inputs

x = [1;0.1;10;10;5;5]; % State variables
u = [4.8;5.3]; % Thrust inputs
w = [2;-3]; % External factors

x_dot = drone_dynamics(x,u,w,drone);
```

```
function x_dot = drone_dynamics(x,u,w,drone)

% DRONE DYNAMICS
% All values are in S.I. units!!
%x = [ vn, vd, the, thed, gam, gamd ];
%x_dot = [vnd, vdd, thed, thedd, gamd, gamdd ];
%u = [ T1, T2 ];
%w = [ wn, wd ];

%% drone params
md = drone.drone_mass; % mass of drone [in S.I. units]
mc = drone.cup_mass; % mass of cup [in S.I. units]
l = drone.cup_holder_length; % Cup Holder Lever Arm [in S.I. units]
ld = drone.propeller_arm; % Propeller arm [in S.I. units]
J = drone.moment_of_inertia; % moment of inertia [in S.I. units]
Cd = drone.drag_coefficient; % Drag coefficient
g = drone.gravity_acc; % gravity [in S.I. units]

%% init variables

vn = x(1);
vd = x(2);
the = x(3);
thed = x(4);
gam = x(5);
gamd = x(6);
T1 = u(1);
T2 = u(2);
wn = w(1);
wd = w(2);
alpha = gam + the; % assuming alpha = theta + gamma

%% Defining Pi matrix

Pi = [ md 0 0 0 0 0 sind(alpha);
       0 md 0 0 0 0 cosd(alpha);
       mc 0 0 -mc*l*cosd(alpha) 0 -mc*l*cosd(alpha) -sind(alpha);
       0 mc 0 mc*l*sind(alpha) 0 mc*l*sind(alpha) -cosd(alpha);
       0 0 0 J 0 0 0;
       0 0 1 0 0 0 0;
       0 0 0 0 1 0 0 ]

%% Defining h matrix

h = [ -(T1+T2)*sind(the)-Cd*(vn-wn);
      md*g-(T1+T2)*cosd(the)-Cd*(vd-wd);
      -mc*l*(((thed+gamd)*pi/180)^2)*sind(alpha);
      mc*g-mc*l*(((thed+gamd)*pi/180)^2)*cosd(alpha);
      (T2-T1)*ld;
      thed;
      gamd ]
%% Calculating x_dot

x_dot = inv(Pi)*h;
x_dot(end)=[]; % to remove the value of F
x_dot(4)=x_dot(4)*180/pi;        % to report the final value of thetad in degrees
x_dot(6)=x_dot(6)*180/pi;        % to report the final value of gammad in degrees

end
```