**NATIONAL UNIVERSITY OF SINGAPORE**

**CEG5303: Intelligent Autonomous Robotic Systems**

**Lab: Guidance for Aerial Drone**

Instructions: Simple Example

In this part, the motion control problem is implemented for quadcopter with PID control.

You are required to:

* Familiar with the dynamics and kinematics of aerial drone systems, establish any one of the drone models, i.e., quadcopter, helicopter, multirotor drone, etc. In addition, explain the model that you have established.
* Establish the simulation model/environment using MATLAB/Simulink, which should clearly show the plant model part, controller part and input/output channel.
* Implement a PID control on the model you built. Show the control part in Simulink/MATLAB. Show the controller’s effectiveness in a trajectory following task. Explain how you tuned the PID parameters and discuss your observations.
* Modify anything in the system or controller. For example, use another control method, make a more complicated trajectory, change the dynamics, or add disturbances. Explain your observations.

In this lab, you need to use MATLAB and Simulink to realize software-in-the loop control. You need to familiar with basic language in MATLAB coding (M language), basic blocks, like gain, scope, etc.

Dynamics and control introduction

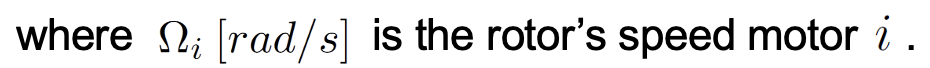
Consider a quadcopter actuated by 4 motors:

A diagram of a quadcopter

Description automatically generated

The actuation of the quadcopter can be modeled as

A math equations with numbers and symbols

Description automatically generated with medium confidence 

To simplify, we take the input to the quadcopter system as

A close-up of a number

Description automatically generated

The states of the quadcopter we observe are the x, y, z position in global reference frame and the roll, pitch, yaw angles.

A black rectangular object with letters and numbers in it

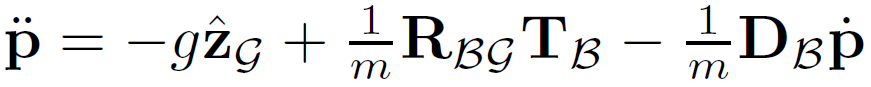
Description automatically generated

The rotational matrix from global inertial frame to quadcopter’s body frame is

A group of black letters

Description automatically generated with medium confidence

Using Euler-Lagrange equations, the dynamics of the quadcopter system can be written as



A black and white text

Description automatically generated

where

A black and white math equation

Description automatically generated with medium confidence

A black and white text

Description automatically generated

A close-up of a number

Description automatically generated

A group of black letters

Description automatically generated

A math equations with numbers and symbols

Description automatically generatedA number of letters and numbers

Description automatically generated with medium confidence

A math equations on a white background

Description automatically generated

The PID controller can be achieved as the following, there are 12 PID parameters to tune.

A diagram of a plane

Description automatically generated

Simulink and MATLAB implementation Example

Open “Quadcopter\_simulation.slx”, which is the Simulink simulation for the quadcopter. The area of the blue square is the 4 PID controller blocks. The red square is the Quadcopter system.

A diagram of a computer

Description automatically generated

Go into the *Quadcopter System* block, and go into the *Dynamics\_1* block, which calculates the acceleration in x, y, z.

Locate TODO\_1, which calculate the rotational matrix and the damping matrix. Complete the code.

Example:

%Calculate the rotational matrix R\_GB and R\_BG

R\_GB = [cos(phi)\*cos(psi) sin(psi)\*cos(phi) -sin(theta);

cos(psi)\*sin(theta)\*sin(phi)-sin(psi)\*cos(phi) sin(psi)\*sin(theta)\*sin(phi)+cos(phi)\*cos(psi) cos(theta)\*sin(phi);

cos(psi)\*sin(theta)\*cos(phi)+sin(phi)\*sin(psi) sin(psi)\*sin(theta)\*cos(phi)-cos(psi)\*sin(phi) cos(phi)\*cos(theta)];

R\_BG = transpose(R\_GB); %Rotation matrix from body frame to global frame

%Calculate the damping matrix

D = [Ax 0 0;

0 Ay 0;

0 0 Az];

Locate TODO\_2, and calculate the accelerations using the dynamics above:

Example:

%Calculate the acceleration

p\_ddot = [0;0;-g] + (1/m)\*R\_BG\*[0;0;T] - (1/m)\*D\*[x\_dot;y\_dot;z\_dot];

x\_ddot = p\_ddot(1);

y\_ddot = p\_ddot(2);

z\_ddot = p\_ddot(3);

Go back to the *Quadcopter System* block, and go into the *Dynamics\_2* block, which calculates the acceleration in roll, pitch and yaw.

Locate TODO\_3, and calculate the coriolis matrix:

Example:

%Calculate the coriolis matrix

c11 = 0;

c12 = (Iyy-Izz)\*(theta\_dot\*cos(phi)\*sin(phi) + psi\_dot\*cos(theta)\*(sin(phi)^2-cos(phi)^2)) - Ixx\*psi\_dot\*cos(theta);

c13 = (Izz-Iyy)\*psi\_dot\*cos(phi)\*sin(phi)\*cos(theta)^2;

c21 = (Izz-Iyy)\*(theta\_dot\*cos(phi)\*sin(phi) + psi\_dot\*cos(theta)\*(sin(phi)^2-cos(phi)^2)) + Ixx\*psi\_dot\*cos(theta);

c22 = (Izz-Iyy)\*phi\_dot\*cos(phi)\*sin(phi);

c23 = (-Ixx\*psi\_dot + Iyy\*psi\_dot\*sin(phi)^2 + Izz\*psi\_dot\*cos(phi)^2)\*sin(theta)\*cos(theta);

c31 = (Iyy-Izz)\*psi\_dot\*cos(theta)^2\*sin(phi)\*cos(phi) - Ixx\*theta\_dot\*cos(theta);

c32 = (Izz-Iyy)\*(theta\_dot\*cos(phi)\*sin(phi)\*sin(theta)+phi\_dot\*cos(theta)\*(sin(phi)^2-cos(phi)^2)) + (Ixx-Iyy\*sin(phi)^2-Izz\*cos(phi)^2)\*psi\_dot\*sin(theta)\*cos(theta);

c33 = (Iyy-Izz)\*phi\_dot\*cos(phi)\*sin(phi)\*cos(theta)^2 + (Ixx - Iyy\*sin(phi)^2 - Izz\*cos(phi)^2)\*theta\_dot\*cos(theta)\*sin(theta);

C = [c11 c12 c13; c21 c22 c23; c31 c32 c33];

Locate TODO\_4, and calculate the angular accelerations:

Example:

%Calculate the acceleration

term = tau - (C\*n\_dot);

W = [1 0 -sin(theta); 0 cos(phi) cos(theta)\*sin(phi); 0 -sin(phi) cos(theta)\*cos(phi)];

I\_B = [Ixx 0 0; 0 Iyy 0; 0 0 Izz];

J = transpose(W)\* I\_B \* W;

output = inv(J)\*term;

phi\_ddot = output(1);

theta\_ddot = output(2);

psi\_ddot = output(3);

Then, open “Quadcopter.m” and locate TODO\_5, set the 12 PID gains. Go to each of the PID controllers blocks in Simulink to understand the PID control structures and tune the PID gains to reach good performance.

A good set of PID gains to start with are:

Kp\_z = 1;

Ki\_z = 0.1;

Kd\_z = 0.6;

Kp\_pos = 3;

Ki\_pos = 0.2;

Kd\_pos = 3;

Kp\_att = 2;

Ki\_att = 0.05;

Kd\_att = 1;

Kp\_psi = 1;

Ki\_psi = 0.1;

Kd\_psi = 1.5;

After that, you can run the “Quadcopter.m” file to view the results. Also take a look at the rest of the code.