Lab 5a: Quadrotor altitude/yaw control

1 Objectives

The goal of this week's lab is to study a quadrotor position control for a drone (model based on Parrot Mambo) as depicted in Figure 1. To do so, we want to design a controller such that we can control the altitude and yaw of such drone.

Figure 1: Parrot Mambo quadrotor. Note that this is NOT the quadrotor that we will be using in our experimental setting.

2 Theory

2.1 Dynamics of a quadcopter

We will consider an idealized quadcopter rigid body model, with translational and rotational dynamics, as described in the paper *Using Simulink Support Package for Parrot Minidrones in Nonlinear Control Education*, from Glaskov and Golubev (2019). The dynamics are given by:

$$m\ddot{x} = F(-\cos\gamma\cos\psi\sin\theta + \sin\gamma\sin\psi) \tag{1}$$

$$m\ddot{y} = F(-\cos\gamma\sin\psi\sin\theta - \cos\psi\sin\gamma) \tag{2}$$

$$m\ddot{z} = -mg + F\cos\theta\cos\gamma \tag{3}$$

with

$$\begin{bmatrix}\dot{\gamma}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & -\sin\gamma\tan\theta & -\cos\gamma\tan\theta\\ 0 & -\cos\gamma & \sin\gamma\\ 0 & \sin\gamma\sec\theta & \cos\gamma\sec\theta\end{bmatrix}\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix} \tag{4}$$

and

$$\begin{bmatrix}I_x & 0 & 0\\ 0 & I_y & 0\\ 0 & 0 & I_z\end{bmatrix}\begin{bmatrix}\dot{\omega}_x\\ \dot{\omega}_y\\ \dot{\omega}_z\end{bmatrix} = \begin{bmatrix}M_x\\ M_y\\ M_z\end{bmatrix} - \begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix} \times \begin{bmatrix}I_x & 0 & 0\\ 0 & I_y & 0\\ 0 & 0 & I_z\end{bmatrix}\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix} \tag{5}$$

$$= \begin{bmatrix}M_x\\ M_y\\ M_z\end{bmatrix} - \begin{bmatrix}(I_z - I_y)\omega_y\omega_z\\ (I_x - I_z)\omega_x\omega_z\\ (I_y - I_x)\omega_x\omega_y\end{bmatrix} \tag{6}$$

In here, (x, y, z) are the coordinates of the vehicle center of mass in the inertial frame, while (γ, θ, ψ) are roll, pitch and yaw angles respectively. Figure 2 depicts the angles in an aircraft.

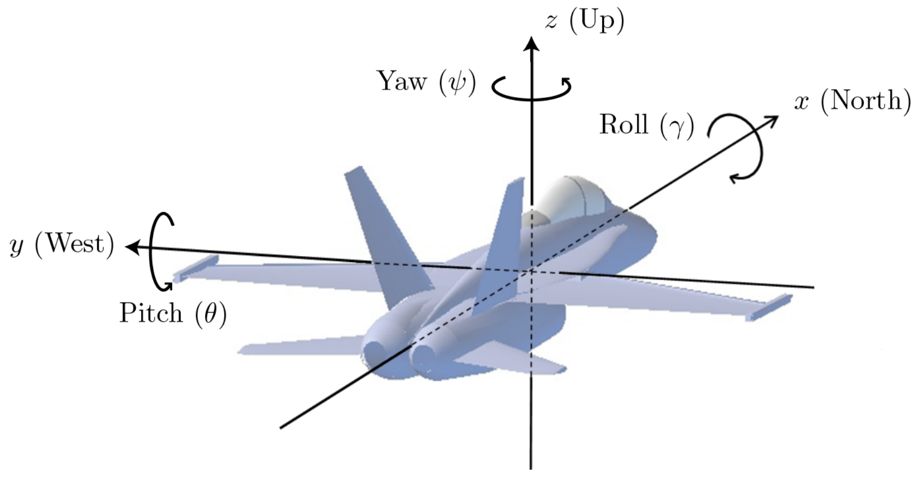

Figure 2: Pitch, roll and yaw in the inertial frame. Source: `http://www.chrobotics.com/library/understanding-euler-angles`

In the dynamics, F represents the thrust produced by the quadcopter rotors, g is the acceleration due to gravity, $M = (M_x, M_y, M_z)^\top$ are the vector of torques, $\omega = (\omega_x, \omega_y, \omega_z)^\top$ is the vector of angular velocities in the body-fixed frame and $I = \text{diag}(I_x, I_y, I_z)$ is the diagonal inertia matrix.

2.2 Control Mixer

Recall that in a quadcopter we actuate the four rotors, produced a resulting thrust F and torques M. Denote the rotor angular velocities as Ω_i. With this the relationship between the thrust force f_i of the rotors and F and M can be modeled by:

$$\begin{bmatrix} F \\ M_x \\ M_y \\ M_z \end{bmatrix} = Pf \tag{7}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & \ell & 0 & -\ell \\ -\ell & 0 & \ell & 0 \\ b/k & -b/k & b/k & -b/k \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{8}$$

on which $f_i = k\Omega_i^2$ are the i-th rotor thrust force, k are the rotors' lift aerodynamic coefficient, b are the rotors' drag aerodynamic coefficient and ℓ is the distance between the quadrotor center of mass and the rotors. The Control Mixer matrix P allows us to control directly F and M, since our motors commands are simply $f = P^{-1}(F, M^\top)^\top$. The signs on the matrix are related with the direction on which the rotors rotate to produce torque. See `https://en.wikipedia.org/wiki/Quadcopter` for an explanation of the design principles.

2.2.1 Altitude and Yaw Control Mixer

Observe from equation (8) that:

$$F = f_1 + f_2 + f_3 + f_4 \tag{9}$$

$$M_z = \frac{b}{k}(f_1 + f_3) - \frac{b}{k}(f_2 + f_4) \tag{10}$$

This implies that the thrust force of all motors will contribute to the thrusting force F. However, since rotors 1 and 3 are rotating in counter-clockwise direction, and rotors 2 and 4 in clockwise direction, an imbalance between these thrusts will produce a torque in the z direction, rotating the aircraft in the yaw axis. Similar analyses can be done for pitch and roll axes.

In any case, note that this system is highly coupled via the pitch and roll angles γ, θ. However, for this laboratory we will focus in the case that no disturbance and control is applied on pitch and roll. That is, $\gamma(t) = \theta(t) = 0$ for every time step, and $\omega_x(t) = \omega_y(t) = 0$ for every time step. We will see that this will highly simplify the dynamics of the quadrotor.

3 Pre-Lab

3.1 Equations of motion

In this section we will focus on obtaining the simplified equations when we assume that pitch and roll are fixed, that is $\gamma = \theta = \omega_x = \omega_y = 0$.

1. Obtain the dynamics for $x, y, \gamma, \theta, \dot{\omega}_x, \dot{\omega}_y$ when we fixed $\gamma = \theta = \omega_x = \omega_y = 0$. You can also assume that $M_x = M_y = 0$. That is, find the equations for:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \dot{\omega}_x \\ \dot{\omega}_y \end{bmatrix} = h(\boldsymbol{x}) \tag{11}$$

   In here $\boldsymbol{x}$ represents the vector of states and inputs. That is, the function could depend on any state $x, y, z, \dot{x}, \dot{y}, \dot{z}, \gamma, \theta, \psi, \dot{\gamma}, \dot{\theta}, \dot{\psi}$ and any input F, M_x, M_y, M_z, while h represents the function that you are looking for.
   Comment on this result. Is it possible to accelerate the aircraft in the x, y directions with pitch and roll being equal to zero at all times?

2. Obtain the dynamics for z and ψ when we fixed $\gamma = \theta = \omega_x = \omega_y = 0$. That is, find the dynamics:

$$m\ddot{z} = h_1(\boldsymbol{x}) \tag{12}$$

$$\dot{\psi} = h_2(\boldsymbol{x}) \tag{13}$$

$$I_z\dot{\omega}_z = h_3(\boldsymbol{x}) \tag{14}$$

3. Now we will consider a less idealized case, when we add drag (or air resistance) of the form $-c_1\dot{z}$ as a force for the altitude dynamics and $-c_2\omega_z$ as a torque for the yaw dynamics. First, re-define the force applied on the z-axis as $F_1 = F - mg$ to eliminate the constant term due to gravity. Obtain the transfer function for $Z(s)/F_1(s)$ and $\Psi(s)/M_z(s)$ (on which $Z(s) = \mathcal{L}\{z(t)\}$ and $\Psi(s) = \mathcal{L}\{\psi(t)\}$)

with the additional drag force and drag torque.

3.2 Bode Plots

Using the following parameters:

Name	Value
Mass: m	0.063 kg
Inertia z-axis: I_z	10^{-4} kg m^2
Friction z-axis: c_1	0.5 kg/s
Friction ψ-axis: c_2	10^{-3} kg m^2/s

Table 1: Parameters for pre-lab.

- Using Matlab, present the Bode Plot for altitude z. Report gain and phase margins. Is the system closed-loop stable? (Hint: use `margin` function and check gain and phase margin being positive).
- Using Matlab, present the Bode Plot for yaw ψ. Report gain and phase margins. Is the system closed-loop stable? (Hint: use `margin` function and check gain and phase margin being positive).

3.3 Controller Design

- For the altitude controller, using root-locus, design a PD controller:

$$C_1(s) = K_1(s + z_1)$$

  that has a settling time of less than one second on the closed-loop system, without any overshoot. You should use `sisotool` or `rlocus` for this problem. Plot the step response of such system.

4 Lab

In this lab we will be using the `Parrot Minidrones Support from Simulink`, available at: `https://www.mathworks.com/hardware-support/parrot-minidrones.html`. They provide multiple alternatives to simulate and visualize that you can explore if you are interested.

4.1 Requirements

We provide a modification from the official project that you can download from the course website. To properly work you will need to install the following add-ons in Matlab:

- Aerospace Toolbox
- Aerospace Blockset
- Image Processing Toolbox
- Computer Vision Toolbox
- Signal Processing Toolbox
- Simulink Control Design

4.2 Instructions

1. Unzip the provided files in the course website in your Matlab folder that you will be working.

2. Open the file `asbQuadcopter.prj`. This will open all the necessary files and initialize the parameters and variables. You should see all check marks when opening. If everything works you can close the drone visualization and scope.
3. Open the file `lab5A_sim.slx`. This will be the Simulink on which we will work. We will run the simulations directly from Simulink (pressing `Run` in the Simulation window). You should export the results to your Workspace to create the required plots.

4.3 Description

The block diagram `Airframe` and `Environment` provides a non-linear non-ideal quadcopter model. The model is based on the model represented in the theory section, but includes additional rotor and environment interactions that impact the dynamics. In the laboratory we will explore such effects.

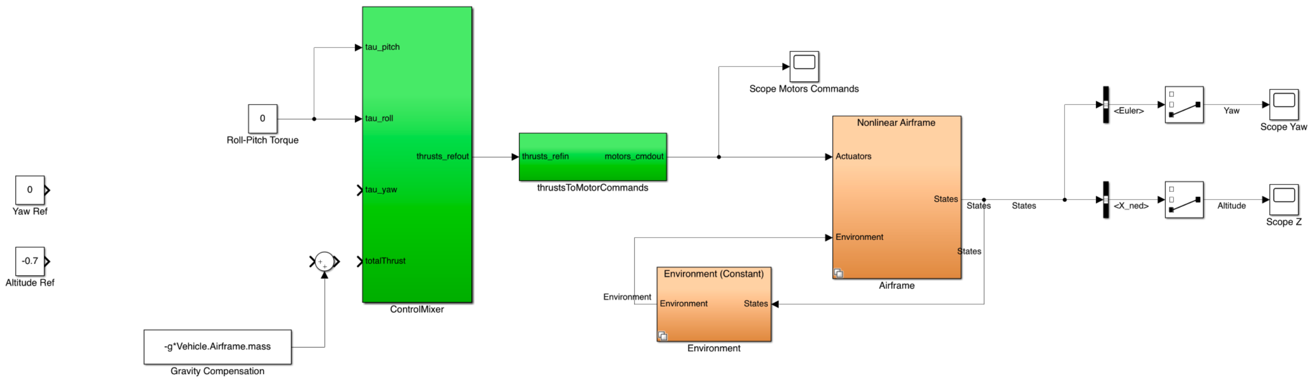

Figure 3: Simulink diagram.

4.4 Implementing controllers

4.4.1 Proportional controller

1. Our first attempt for this new model will be to include a proportional controller on both yaw and altitude. Create two P controller of the form:

$$\texttt{totalThrust} = u_z = K_z(r_z - z) - mg \qquad \texttt{tau\_yaw} = u_\psi = K_\psi(r_\psi - \psi)$$

   Remember to add the gravity compensation $-mg$ to the thrust applied to the Simulink block (`totalThrust`). The initial conditions of the system are: $\psi(0) = \theta(0) = \gamma(0) = 0$ and $z(0) = -0.7$.

   Maintaining the yaw reference at $r_\psi = 0$, apply a step at $t = 1$ from $r_z = -0.7$ to $r_z = -1.0$ (in the Simulink, negative values of z represent above the ground).

   - Can you stabilize the system with only a P controller? You are free to explore different values of P. Report in a plot the step-response of z for a given K_z.

2. Now, maintaining $r_z = -0.7$, apply a step at $t = 1$ on r_ψ from 0 to 0.4. Plot the response of z and ψ. Try to increase the total time and see if its converging.

   - Is the system coupled in ψ and z? Explore some reasons of why this could happen. Explore the motor output torques in the Simulink `motor_cmdout`. Analyze equations (9) and (10) in a real system to explain such phenomenon. You can assume that the rotor thrusts are not exactly the same when it saturates, as occur in real life when rotating at fast speeds.

- Can you make the system stable with the P controller?

4.4.2 Lead compensator

- For the altitude controller of the **Prelab** (using Prelab transfer function $Z(s)/F(s)$), design a lead controller of the form:

$$C_2(s) = K_2 \frac{(s + z_2)}{(s + p_2)}$$

with $p_2 > z_2 > 0$ that also has a settling time $T_s < 1$ second and $\%OS < 10\%$ on the closed-loop system. You should use `sisotool` or `rlocus` for this problem. Plot the step response of such system.

- In your Simulink model, implement your lead-compensator for the altitude controller. In addition, implement a lead compensator for your yaw-controller too (you can start with the same one of the altitude controller) and repeat the yaw-step from section 4.4.1 (and maintain $r_z = -0.7$). You are free to explore other parameters for the zero/pole and gain (for both altitude and yaw controllers) if you want to try a better response. You can use the `Transfer Fcn` block located in `Simulink/Continuous` to construct your lead compensator.
  - Can you make your system stable with the lead-compensator?
  - Present the plot for both ψ and z. Does the system converge back? How is it response, under-damped or over-damped?
  - Is it coupled between z and ψ? Explore the motor commands output. You can assume that the rotor responses are not exactly the same when it saturates.

4.4.3 PD controller

Finally we will implement a PD controller independently for yaw and altitude. Use the `PID Controller` block located in `Simulink/Continuous` to construct your PD controller. You must use a filter coefficient of at least $N = 100$. The requirements are as follow:

- For the altitude controller apply a step from $r_z = -0.7$ to $r_z = -1.0$ (maintain $r_\psi = 0$). The settling time must satisfy $T_s < 3$ seconds (visually) and the overshoot must be such that the peak must be greater than -1.025 (i.e. z cannot go lower than -1.025). Present the plot for z of this step.
- For the yaw controller, apply a step from 0 to 0.4 (maintain $r_z = -0.7$). The settling time must satisfy $T_s < 1$ second (visually) and must not have overshoot (i.e. no oscillations). Present the plot of both ψ and z of this step.

4.5 Frequency tracking

In here we are interested in tracking a sine wave on the yaw command. You must use your PD controller from the previous section. Maintain in all simulations the altitude reference at $r_z = -0.7$.

Remark: Due to initialization consideration, ignore the changes occurring at the beginning of the simulation. These are transient and numerical effects due to the derivative that we will ignore. It is preferable that in your figures do not plot the first 2 seconds of the simulation.

- Use a `Sine Wave` block to provide a sine wave with amplitude 0.2 and frequency 0.1 Hz as a yaw reference r_ψ. Simulate at least 30 seconds. Present in a single plot the ψ and z response.

- Use a `Sine Wave` block to provide a sine wave with amplitude 0.2 and frequency 1 Hz as a yaw reference r_ψ. Simulate at least 5 seconds. Present in a single plot the ψ and z response.
- Use a `Sine Wave` block to provide a sine wave with amplitude 0.2 and frequency 10 Hz as a yaw reference r_ψ. Simulate at least 5 seconds. Present in a single plot the ψ and z response.

Analyze the three plots and compare the differences:

- Analyze the motor command responses and plot them if it's necessary. Check saturation
- Is the system coupled now? What can you say in comparison to the coupling between z and ψ when doing step responses in the previous sections?
- Can you make the 10 Hz system stable by modifying your yaw PD controller?