Quad-rotor P&S

Exercise 1: Simulation Development

Learning objectives relevant for this exercise sheet:

ii) Be able to simulate a general N-rotor vehicle for the purpose of testing and tuning controller and estimator designs,

iii) Be able to explain how flight performance is affected by changes in the parameters of the N-rotor vehicle, for example mass, centre of gravity location, propeller aerodynamics.

Pre-work Tasks

Complete the introduction to Simulink tutorial in Section 3.1 of the course script to familiarise with the Simulink tools required to complete this exercise sheet. Section 3.1.1 provides an introduction for those who have never used Simulink before, and Section 3.1.2 provides a brief explanation of the Simulink template provided for this exercise sheet. **You should also read through the comments in the file "`exercise01_simulation_parameters.m`" for hints and to understand what variables are available for use in the Simulink model.**

In-class Tasks

The following steps are a guideline for developing the template Simulink model provided into a high-fidelity N-rotor vehicle simulation that will allow you to assess the empirical-stability and performance of a controller without damaging your real-world vehicle.

A) Equations of motion

Add the non-linear, continuous-time, equations of motion to the template N-rotor vehicle Simulink model provided.

- Before working in Simulink, explain why there are seemingly 15 equations of motion despite the full state vector being defined as a 12-dimensional. For convenience, the full state definition and equations of motion are stated in the additional material at the end of this exercise sheet.
- Use the **"all-in-one-block"** approach described in **Section 3.1.3** of the course script. The file "`exercise01_fcn_eom_template.m`" provides a template for the code to write in the "MATLAB Function" block that is used for the equations of motion.
- All the variables needed to complete this task are specified and described by the comments in the file "`exercise01_get_vehicle_paramters.m`".
- Confirm the equations have been correctly added to the Simulink model by observing the reference tracking performance.

B) Sensitivity to model mis-match and feed-forward (equilibrium) thrust

In the file "`exercise01_get_vehicle_paramters.m`", adjust the mass used for the controller by $\{-4\%, -2\%, +2\%, +4\%\}$ and measure the steady-state offset from the reference, i.e., adjust the variable named "`nrotor_vehicle_mass_for_controller`". Is the offset linear in the mis-match?

Adjust also the propeller layout dimensions used for the controller by $\{-4\%, -2\%, +2\%, +4\%\}$ and measure the steady-state offset from the reference, i.e., adjust the variable named "`nrotor_vehicle_layout_for_controller`". Is the offset linear in the mis-match?

The goal of this task is to derive and implement a computation of the feed-forward (equilibrium) thrusts.

- The feed-forward architecture is briefly described in the additional material at the end of this exercise sheet.
- Recall the 4 actuators defined in the script, namely the total thrust force f_{total}, and the torque about each axis of the body frame $(\tau_x^{(\text{B})}, \tau_y^{(\text{B})}, \tau_z^{(\text{B})})$, see Figure 2.7 in the script.
- Derive all the possible state and input combinations that are equilibrium points of the non-linear, continuous-time equations of motions.
- For an N-rotor vehicle, the actuator and motor thrusts are related as:

$$\begin{bmatrix} f_{\text{total}} \\ \tau_x^{(\text{B})} \\ \tau_y^{(\text{B})} \\ \tau_z^{(\text{B})} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ y_1 & y_2 & \cdots & y_N \\ -x_1 & -x_2 & \cdots & -x_N \\ c_1 & c_2 & \cdots & c_N \end{bmatrix}}_{M_{\text{layout}}} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix},$$

  and thus the motor thrusts required to produce a set of actuations can be computed using the (pseudo) inverse of M_{layout}.
- Open the file "`exercise01_compute_equilibrium_thrusts.m`", delete the code that is already there, and implement the pseudo inverse of M_{layout} for computing the equilibrium thrusts.
- For a vehicle with $N \geq 5$ rotors, is the pseudo inverse of M_{layout} an appropriate choice? Explain your answer.
- Does the inverse always exists when $N = 4$ rotors? If not, find a simple quad-rotor where the inverse does not exist.

C) Discrete time controller

In reality, measurements of the vehicle's state are only available in discrete time, and new commands can only be sent to the propellers at discrete time instants. Convert your continuous-time model to a hybrid model where the non-linear dynamics of the vehicle are simulated in continuous time, the full-state measurements are taken at a particular frequency, and the controller computations are performed at a different frequency.

- This is implemented most easily with the "Zero-Order Hold" block in Simulink.
- The frequencies should be specified as parameters that are easy to change. When state estimation is not performed, these frequencies are typically the same, and for the experiments we will collect measurements at 200Hz.
- In the file "`exercise01_simulation_parameters.m`" you find the the variable named "`K_lqr_full_state`", and the comments there provide instructions for how to choose the controller feedback matrix appropriate for the frequency.
- How is the step change tracking performance affected by the frequency of control computations? Does the closed loop system become unstable at some frequency?
- Test out the following two combination: (i) set the controller computations to a frequency of 20Hz, but use the controller parameters designed for 200Hz, (ii) set the controller computations to a frequency of 200Hz, but use the controller parameters designed for 20Hz. Explain the results using intuitive arguments.

Additional Tasks

D) Additional Task: Measurement noise

Measurements of the vehicle's state are always corrupted by some level of noise. Add zero-mean Gaussian (white) noise to each of the states separately.

- This is implemented most easily by using the "Random Number" block (found in the "Sources" category) in Simulink to inject a vector with independent samples.
- The measurement noise should be added in a manner that allows you to quickly select between simulating the vehicle with and without noise.
- How is the step change tracking performance affected by the amplitude of the added noise? Is the vehicle's performance more sensitive to noise on particular measurements?
- Is zero-mean white noise a realistic assumption? What other types of noise might exist for particular measurements.

E) Additional Task: thrust-to-command conversion

Thus far the controller has requested a particular thrust from the propellers of our N-rotor vehicle. However, in reality on the Crazyflie 2.0 hardware we will use for the experiments, the micro-controller sends an integer command in the range $[0 - 65535]$, where 0 is zero-thrust, and 65535 is full-thrust. Add to the controller a conversion from force requested to this integer command, and add to the N-rotor vehicle model the conversion from this integer command to thrust produced by the respective propeller.

- For the propellers used on the Crazyflie 2.0, this static "integer command -to- thrust" conversion was identified as:

$$\text{thrust [N]} = \text{thrust}_{\text{max}} \left(1.3385 \cdot 10^{-10}\, \text{cmd}^2 + 6.4870 \cdot 10^{-6}\, \text{cmd} \right)$$

  where cmd is the integer command in the range $[0 - 65535]$, and $\text{thrust}_{\text{max}}$ is the maximum thrust in Newtons produced by the propeller at $\text{cmd} = 65535$.
- The purpose of including this conversion in the simulation is so that the simulated controller architecture and tuning matches that of the real-world system.

Additional info - equations of motions summary:

To save the trouble of flipping back-and-forth through the script, the following re-states the choice of the full state, the equations of motion, and the equation for the rotation and transformation matrix. The full state is defined as:

$$\begin{bmatrix} \vec{p} \\ \dot{\vec{p}} \\ \vec{\psi} \\ \dot{\vec{\psi}} \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{linear velocity} \\ \text{attitude} \\ \text{angular velocity} \end{bmatrix} , \quad \text{where } \vec{\psi} = \begin{bmatrix} \gamma \\ \beta \\ \alpha \end{bmatrix} = \begin{bmatrix} \text{roll} \\ \text{pitch} \\ \text{yaw} \end{bmatrix} , \tag{1}$$

for which the non-linear, continuous-time equations of motion are:

$$\dot{\vec{p}} = \frac{d}{dt}\left(\vec{p}\right) \tag{2a}$$

$$\ddot{\vec{p}} = \begin{bmatrix} \ddot{p}_x \\ \ddot{p}_y \\ \ddot{p}_z \end{bmatrix} = \frac{1}{m}\left({}_{(\mathrm{I})}R^{(\mathrm{B})}(\vec{\psi}) \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{N} f_i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \right) , \tag{2b}$$

$$\dot{\vec{\psi}} = \frac{d}{dt}\left(\vec{\psi}\right) \tag{2c}$$

$$\ddot{\vec{\psi}} = \begin{bmatrix} \ddot{\gamma} \\ \ddot{\beta} \\ \ddot{\alpha} \end{bmatrix} = T^{-1}(\vec{\psi}) \left(\dot{\vec{\omega}} - \dot{T}(\vec{\psi})\,\dot{\vec{\psi}} \right) , \tag{2d}$$

$$\dot{\vec{\omega}} = \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = J^{-1}\left(\begin{bmatrix} \sum_{i=1}^{N} f_i\, y_i \\ \sum_{i=1}^{N} -f_i\, x_i \\ \sum_{i=1}^{N} f_i\, c_i \end{bmatrix} - \left(\vec{\omega} \times J\,\vec{\omega}\right) \right) . \tag{2e}$$

The rotation matrix, transformation matrix, its inverse, and its time derivative are given by the following expressions:

$${}_{(\mathrm{B})}R^{(\mathrm{I})}(\vec{\psi}) = \begin{bmatrix} c_\beta\, c_\alpha & c_\beta\, s_\alpha & -s_\beta \\ (-c_\gamma\, s_\alpha + s_\gamma\, s_\beta\, c_\alpha) & (c_\gamma\, c_\alpha + s_\gamma\, s_\beta\, s_\alpha) & s_\gamma\, c_\beta \\ (s_\gamma\, s_\alpha + c_\gamma\, s_\beta\, c_\alpha) & (-s_\gamma\, c_\alpha + c_\gamma\, s_\beta\, s_\alpha) & c_\gamma\, c_\beta \end{bmatrix} , \tag{3}$$

$$T(\vec{\psi}) = \begin{bmatrix} 1 & 0 & -\sin(\beta) \\ 0 & \cos(\gamma) & \sin(\gamma)\,\cos(\beta) \\ 0 & -\sin(\gamma) & \cos(\gamma)\,\cos(\beta) \end{bmatrix} , \tag{4}$$

$$T^{-1}(\vec{\psi}) = \begin{bmatrix} 1 & \sin(\gamma)\,\tan(\beta) & \cos(\gamma)\,\tan(\beta) \\ 0 & \cos(\gamma) & -\sin(\gamma) \\ 0 & \sin(\gamma)\,\sec(\beta) & \cos(\gamma)\,\sec(\beta) \end{bmatrix} , \tag{5}$$

$$\dot{T}(\vec{\psi}, \dot{\vec{\psi}}) = \begin{bmatrix} 0 & 0 & -\dot{\beta}\cos(\beta) \\ 0 & -\dot{\gamma}\sin(\gamma) & \dot{\gamma}\cos(\gamma)\,\cos(\beta) - \dot{\beta}\sin(\gamma)\,\sin(\beta) \\ 0 & -\dot{\gamma}\cos(\gamma) & -\dot{\gamma}\sin(\gamma)\,\cos(\beta) - \dot{\beta}\cos(\gamma)\,\sin(\beta) \end{bmatrix} . \tag{6}$$

Recall that the transformation matrix T relates the body rates and Euler angular rates as:

$$\vec{\omega} = T(\vec{\psi})\dot{\vec{\psi}} \tag{7}$$

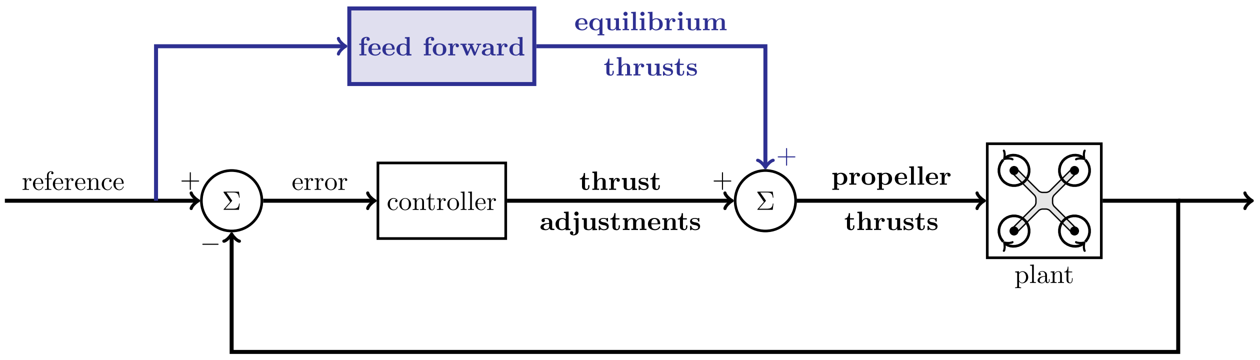

Figure 1: Schematic of the feed-forward architecture for an N-rotor vehicle.

Additional info - Feed forward architecture:

It is common to design a controller for making adjustments around a chosen equilibrium point. It is important to remember that the equilibrium point chosen is a combination of:

(i) an equilibrium state of the system, and

(ii) the equilibrium input that keep the system at that equilibrium state.

For an N-rotor vehicle, the inputs to the vehicle are the thrust for each propeller, and therefore a controller designed for a linearised N-rotor vehicle system is a function that:

- should receive the **error** between the current state of the system and the **reference** equilibrium state, and
- returns the **thrust adjustments** that should be made to the **equilibrium thrust** of each propeller.

This equilibrium input is commonly referred to as the feed-forward input, and the feed-forward architecture for an N-rotor vehicle is shown in Figure 1. The control function can therefore be described in words as:

$$\begin{array}{c}\text{propeller}\\\text{thrusts}\end{array} = \begin{array}{c}\text{equilibrium}\\\text{thrusts}\end{array} + \begin{array}{c}\text{thrust}\\\text{adjustments}\end{array}. \qquad (8)$$