May 16, 2017

A CAD-Based Simulator to Compare Control Architectures for Quadrotors

by

Kevin Murray Jr., Robert B. Anderson, and Andrea L'Afflitto

1. Introduction

This Simulink toolbox allows to compare and contrast the performance of any two control architectures for quadrotors, such as the proportional-integral-derivative (PID) and a model reference adaptive control (MRAC) architecture. A *unique feature* of this simulator is that the the quadrotor's dynamics is not captured by a set of differential equations, but a CAD model generated in Solidworks and then imported in Simulink using the Simscape Multibody blockset; in this simulator, the CAD model of a DJI F450 quadrotor is utilized [1]. An additional feature of this simulator is that the efficiency of one of the quadrotor's propellers can be varied to simulate a *failure* in the control system. Finally, the quadrotors' trajectories obtained using two different control techniques can be compared with the ideal reference trajectory to better evaluate the controllers' performances.

This toolbox was first created by Robert B. Anderson, Riley Cotter, Jordan Logue, and Kevin Murray Jr. as part of an undergraduate students project for the AME 4513/5513 "Flight Controls" course at the School of Aerospace and Mechanical Engineering by coding the results discussed in [2]; this original version has been improved and finalized at the Advanced Control Systems Lab (ACSL) of the University of Oklahoma.

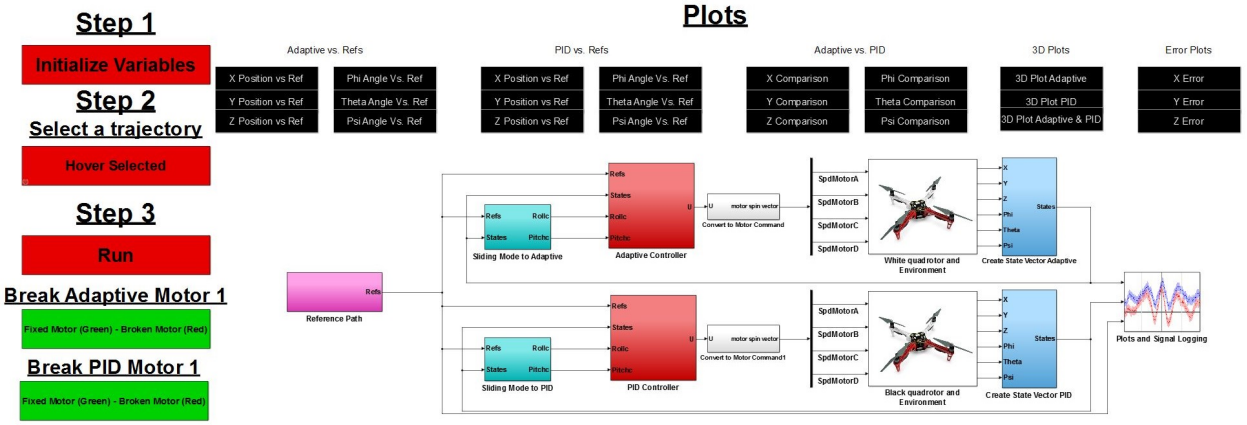

Figure 1.1: Simulator Screenshot

2. Performing a Simulation

To perform a simulation,

1. Double-click on 'Initialize Variables' to load the simulation variables stored in the file 'Simulator_Variables.m.'

2. Select a reference trajectory by double-clicking on the 'Step 2' button; in its original format, this simulator allows to track a square, a circular path, or hover at a given position. Existing paths can be altered and new paths can be added by modifying 'Simulator_Variables.m' and the 'Reference Path' block.

3. Double-click on 'Run' to start the simulation.

4. At any time during the simulation, the efficiency of a motor can be modified by double-clicking on the 'Break Adaptive Motor 1' or 'Break PID Motor 1' buttons. This way, the efficiency of one of the propellers for each respective quadrotor will be reduced to the amount specified by the variable 'changeToEff' in the file named 'Simulator_Variables.m' on line 76. By default, 'changeToEff' is set to 60%.

   Remark: By breaking a motor in the PID Controller, the corresponding quadrotor may turn upside down and the simulation will not be completed due to a failure in the Simspace Multibody blockset; in practice, the quadrotor will crash. However, the adaptive controller proves to be resilient to this kind of failures. If the variable 'changeToEff' is set to a value close to zero, such as 0.01, then the adaptive controller is still able to fly the quadrotor for all the trajectories pre-loaded in this simulator.

5. A 3D animation shows the simulation results in a $40 \times 20 \times 20$ cubic-foot environment. At the end of the simulation, the quadrotor's reference and actual trajectories can be compared by double-clicking on the plot buttons at the top of this toolbox.

The simulation time can be varied by changing the variable 'simulation_time' in the 'Simulator_Variables.m' file.

3. Brief Description of the Simulink Model

In this section, we briefly present the model reference adaptive control algorithm coded in this simulator. For details, see [2] and [3].

3.1. Notation

In the following, we briefly summarize the mathematical notation used through this user manual. The current position of the quadrotor's center of mass is denoted by $[x, y, z]^{\mathrm{T}}$ and its current attitude is captured by the 3-2-1 Euler angles $[\psi, \theta, \phi]^{\mathrm{T}}$. The reference position and attitude are denoted by $[x_c, y_c, z_c]^{\mathrm{T}}$ and $[\psi_c, \theta_c, \phi_c]^{\mathrm{T}}$, respectively. In order to implement an MRAC controller, we define an ideal quadrotor [3, Ch. 9], whose position and attitude are denoted by $[x_d, y_d, z_d]^{\mathrm{T}}$ and $[\psi_d, \theta_d, \phi_d]^{\mathrm{T}}$, respectively.

3.2. The 'Reference Path' Block

The 'Reference Path' block shown in Figure 3.1 generates the trajectory for the quadrotor to follow. In its original version, this simulator allows to hover at a given position or track a square or a circular path. The output of this block is the vector $[x_c, y_c, z_c, \psi_c]^{\mathrm{T}}$.

Figure 3.1: 'Reference Path' block

3.3. The 'Sliding Mode' Blocks

The 'Sliding Mode' block shown in Figure 3.2 generates the roll and pitch angles ϕ_c, θ_c [4] needed to achieve the desired quadrotor's position in the x-y plane using the relation

$$\begin{bmatrix} \phi_c \\ \theta_c \\ F_{zc} \end{bmatrix} = \begin{bmatrix} \big(\ddot{x}\sin\psi - \ddot{y}\cos\psi\big)m/F_z \\ \big(\ddot{x}\cos\psi + \ddot{y}\sin\psi\big)m/F_z \\ \big(\ddot{z}+g\big)m + \hat{\Delta}_1 \end{bmatrix}, \tag{1}$$

where g denotes the gravitational acceleration, m denotes the mass of the vehicle, F_z denotes the total current thrust of the vehicle, and $\hat{\Delta}_1$ is an adaptive estimation to account for differences from the simplified model in [4]. The sliding surface is defined as

$$s_1 \triangleq \ddot{z}, \tag{2}$$

and is such that

$$\dot{\hat{\Delta}}_1 = \lambda_1 s_1. \tag{3}$$

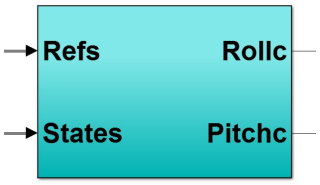

Figure 3.2: 'Sliding Mode' block

3.4. The 'Adaptive Controller' Block

Let f_1, f_2, f_3, and f_4 denote the forces generated by the four propellers of a quadrotor, let L denote the distance from the propellers' shaft to the quadrotor's center of mass, and define the quadrotor's *control vector* as

$$u \triangleq \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & L & 0 & -L \\ -L & 0 & L & 0 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}. \tag{4}$$

The 'Adaptive Controller' block shown in Figure 3.3 employs a MRAC architecture [2,3] to compute $u(\cdot)$, such that the quadrotor mimics the ideal quadrotor's dynamics and tracks the reference trajectory. Specifically, 'Adaptive Controller' block computes the control law as

$$u(t) = \overline{\Phi}(z(t), \psi(t), \theta(t), \phi(t), g)\hat{\Theta}(t) - K_2 s_2, \quad t \geq 0, \tag{5}$$

where

$$\overline{\Phi}(z,\psi,\theta,\phi,g) \triangleq Q(z,\psi,\theta,\phi) + F(\psi,\theta,\phi,g), \tag{6}$$

$$Q(z,\psi,\theta,\phi) \triangleq \begin{bmatrix} \ddot{z} & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddot{\phi} & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddot{\theta} & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddot{z} & 0 & 0 \end{bmatrix}, \tag{7}$$

$$F(\psi,\theta,\phi,g) \triangleq \begin{bmatrix} g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\dot{\theta}\dot{\psi} & 0 & \dot{\phi}\dot{\psi} & 0 \\ 0 & \dot{\phi}\dot{\psi} & 0 & 0 & -\dot{\phi}\dot{\psi} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\dot{\phi}\dot{\theta} \end{bmatrix}, \tag{8}$$

$\hat{\Theta} : [0,\infty) \to \mathbb{R}^6$ denotes the solution of the differential equation

$$\dot{\hat{\Theta}}(t) = -P\bar{\Phi}^{\mathrm{T}}(z(t),\psi(t),\theta(t),\phi(t),g)s_2(t), \qquad \hat{\Theta}(0) = 0, \qquad t \geq 0, \tag{9}$$

$P \in \mathbb{R}^{6\times 6}$ is a positive-definite gain matrix, $s_2 \triangleq \dot{e}_2 + \Lambda_2 e_2$, $e_2 \triangleq q_2 - q_{2d}$, $\Lambda_2 \in \mathbb{R}^{4\times 4}$ is a positive-definite gain matrix, $q_2 \triangleq [z,\phi,\theta,\psi]^{\mathrm{T}}$, $q_{2d}(\cdot)$ denotes the solution of the differential equation

$$\frac{d}{dt}\begin{bmatrix} q_{2d}(t) \\ \dot{q}_{2d}(t) \end{bmatrix} = A_{m2}\begin{bmatrix} q_{2d}(t) \\ \dot{q}_{2d}(t) \end{bmatrix} + B_{m2}q_{2c}(t), \qquad \begin{bmatrix} q_{2d}(0) \\ \dot{q}_{2d}(0) \end{bmatrix} = \begin{bmatrix} q_{2d0} \\ q_{2dv0} \end{bmatrix}, \tag{10}$$

$$A_{m2} = \begin{bmatrix} 0_{4\times 4} & I_{4\times 4} \\ a_{11} & a_{22} \end{bmatrix}, \; B_{m2} = \begin{bmatrix} 0_{4\times 4} \\ -a_{22} \end{bmatrix},$$

$$a_{11} = \begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & -18 & 0 & 0 \\ 0 & 0 & -18 & 0 \\ 0 & 0 & 0 & -40 \end{bmatrix}, \qquad a_{22} = \begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & -4 & 0 & 0 \\ 0 & 0 & -4 & 0 \\ 0 & 0 & 0 & -40 \end{bmatrix},$$

and K_2 is a postive-definite gain matrix.

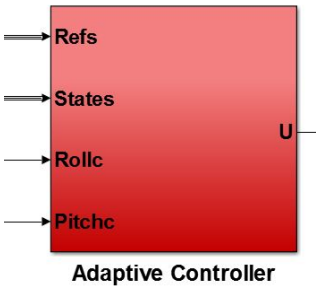

Figure 3.3: 'Adaptive Controller' block

3.5. 'PID Controller' Block

The 'PID Controller' block shown in Figure 3.4 applies the classic PID control to the error signals for both z and ψ while applying a PD control law to both ϕ and θ.

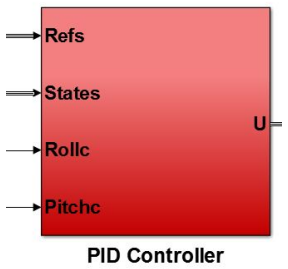

Figure 3.4: 'PID Controller' block

3.6. 'Convert to Motor Command' Block

The 'Convert to Motor Command' block shown in Figure 3.5 transforms the control vector computed using the 'Adaptive Controller' block or the 'PID Controller' block into the force applied by each propeller according to the relation (4). Then the voltage required to obtain the desired force in each propeller is computer by taking the the square root of each propeller's force and scaling the result by a constant gain. This gain has been computed by creating a calibration curve for the quadrotor's CAD model used in this toolbox.

Figure 3.5: 'Convert to Motor Command' block

3.7. 'Quadrotor and Environment Model' Blocks

The 'Quadrotor and Environment Model' blocks shown in Figure 3.6 merge the quadrotors' and the and environment's CAD models to perform numerical simulations and produce 3D animations; the DJI F450 quadroto quadrotor model available at [1] has been used in this toolbox.

3.8. 'Create State Vector' Blocks

The 'Create State Vector' blocks shown in Figure 3.7 capture the quadrotor's position and attitude computed by the 'Quadrotor and Environment' block; see Section 3.7.

Figure 3.6: 'Quadrotor and Environment' block

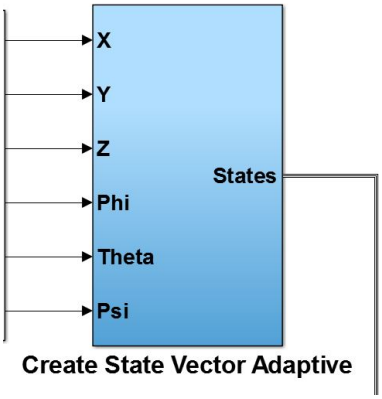

Figure 3.7: 'Create State Vector' block

3.9. 'Plots and Signal Logging' Block

The 'Plots and Signal Logging' block shown in Figure 3.8 allows storing the simulation data and displaying simulation plots, such as

- the trajectory and attitude of the quadrotor controlled by the adaptive control law vs. the trajectory and attitude of the quadrotor controlled by the PID controller;
- the trajectory and attitude of the quadrotor controlled by the adaptive control law vs. the reference trajectory and attitude;
- the trajectory and attitude of the quadrotor controlled by the PID control law vs. the reference trajectory and attitude;
- the trajectory tracking error;
- 3D plots of the quadrotors' trajectories.

These plots can be visualized by double-clicking on the plot buttons at the top of the toolbox.

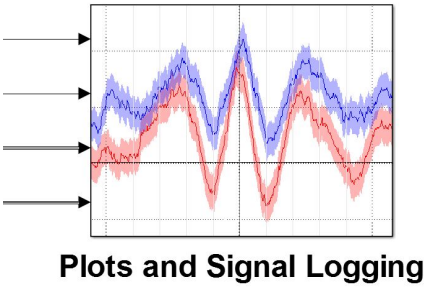

Figure 3.8: 'Plots an Signal Logging' block

References

[1] "Simulate Quadrotor in Simulink with SimMechanics," http://www.mathworks.com/matlabcentral/fileexchange/48052-simulatequadrotor-in-simulink-with-simmechanics, accessed: 2016-12-07.

[2] B. J. Emran, "Nonlinear Adaptive Control of Quadrotor," Master's thesis, American University of Sharjah, United Arab Emirates, 2014.

[3] E. Lavretsky and K. Wise, *Robust and Adaptive Control: With Aerospace Applications*. London, UK: Springer, 2012.

[4] L. Wang, C. He, and P. Zhu, "Adaptive sliding mode control for quadrotor aerial robot with I type configuration," *International Journal of Automation and Control Engineering*, vol. 3, no. 1, 2014.