Software-in-the-loop simulation for improving flight control system design: a quadrotor case study

Giuseppe Silano, Pasquale Oppido and Luigi Iannelli

1. Motivation

Multi-rotor aircraft are a fast-growing field of robotics and nowadays are rapidly increasing in popularity also out of the scientific community. However, designing autopilots for UAVs (Unmanned Aerial Vehicles) is a challenging task, which involves multiple interconnected aspects. Therefore, having tools able to show what it happens when some new applications are going to be developed in unknown or critical situations is more and more important.

Goal: Showing how the use of the SIL (software-in-the-loop) techniques allows to understand the behavior of flight control systems discovering issues that model-in-the-loop (MIL) simulation approaches do not necessarily detect, even if carried out through a multi-physics co-simulation approach.

2. Case Study

The case study here considered is the stabilizing controller discussed in [1] that in our case study it has been designed by considering the Parrot Bebop 2 quadrotor.

A detailed aircraft model was used in a twofold way: firstly, for tuning the controller gains (obtained as the solution of an optimization problem), and then for validating the flight control system by comparing MIL simulation results (the flight control system implemented as a Simulink model) with those obtained through SIL simulations (the flight control system implemented as an executable object code in ROS/Gazebo).

3. Flight Control System

With the aim of illustrating a control design methodology exploiting the SIL simulation, a common cascade control architecture was used.

![Diagram of the control scheme](image)

Figure 1: The control scheme. Subscript \( d \) indicates the drone variables and \( r \) indicates references to controllers.

The position controller (the outer loop controller) uses the measured drone position \( \xi \) to compute the thrust \( u \) and the attitude \( \psi \) that should have the drone in order to reach the desired position \( \xi_d \) with the desired heading (yaw angle) \( \psi_d \). The attitude controller (the inner loop controller) uses the measured drone attitude \( \psi \) to compute the control inputs \( u_\psi \), \( u_\zeta \) and \( u_z \) that should be actuated to achieve the desired attitude \( \psi = \psi_d \) and \( \psi = \psi_d \). The controller mixers the controller outputs obtaining the commanded motor velocities \( \Omega_k \), later used as inputs for the aircraft dynamical model (it considers also the rotors dynamics).

Controller equations:

\[
\begin{align*}
\dot{v}_y &= \left( \alpha y \right) \Omega_1 - \beta y \Omega_2 + m g, \quad (x, y, z) \\
\dot{v}_z &= \left( \beta z \right) \Omega_1 - \alpha z \Omega_2 + m g
\end{align*}
\]

where \( \Omega_k \) are the components of the position tracking errors.

\[
\begin{align*}
\dot{v}_\psi &= \Omega_1 - \Omega_2 + \Omega_3 \\
\dot{v}_\zeta &= \Omega_2 - \Omega_3 + \Omega_4 \\
\dot{v}_z &= \Omega_3 - \Omega_4 + \Omega_5
\end{align*}
\]

where \( J = \text{diag}(I_x, I_y, I_z) \) is the inertia matrix of the vehicle w.r.t. its principal axis. Finally, according to [1, eqn. (40)-(41)], the MATLAB/Simulink platform was used to minimize in a numerical way the integral of the squared error (ISE)

\[
\text{ISE}(\mu_\zeta, \beta \psi) \triangleq \int_{t_f}^{t_i} \left[ (\xi_d(t) - \xi(t))^2 + (\psi_d(t) - \psi(t))^2 \right] dt
\]

w.r.t. the control parameters \( \mu_\zeta, \beta \psi, k \in \{x, y, z, \theta, \phi, \psi\} \) taking into account also motor dynamics, saturation constraints and controller discretization.

4. Numerical Results

When moving from the control design based on the nominal model to the actual implementation, several issues should be addressed:

- First, the control architecture. In our case it is based on control loops that are nothing but PD (Proportional-Derivative). For such class of controllers, a classical way of dealing with the time derivative of controller error is to differentiate only the output signal;
- Then, the controller discretization. As a common rule in cascade structures, the inner loops need to be regulated at a rate faster than the outer. In our case, the attitude controller runs at 200 Hz while the position controller runs at 100 Hz.
- Finally, the hardware constraints and software implementation. The controller has to clip the desired rotor velocity so that \( 0 \leq \Omega_k^\text{ref} \leq \Omega_{\text{max}} \). Furthermore, software aspects such as synchronization, overflow, tasks communication have to be managed paying attention to maintain the controller behavior.

SIL Simulation: The SIL simulation has been carried out by the ROS (Robot Operating System) middleware using Gazebo and the ROS package RotorS as robotic simulators. The considered architecture was used to understand how the system behaves when moving from MATLAB to Gazebo (see Fig. 2) and if the stability is still hold.

![Graph showing the results of the SIL simulation](image)

Figure 2: The \( \psi \) and \( u_z \) signals obtained doing the same experiment in MATLAB/Simulink and Gazebo, respectively.

By looking at first tenth of seconds of simulations it comes out that the \( u_z \) control signal assumes negative values much below \( -m g = -4.9 N \), see Fig. 2(b). That is an important issue since \( u_z < -m g \Rightarrow u_z + m g < 0 \) but [3], that means \( u_z + m g = u_z(e_u) < 0 \). Of course, it is not possible to apply a thrust that gives the desired \( u_z \) and such specific situation is well known in literature to bring the system to instability [2].

Conclusions: Thus, considering the approach proposed in [2] and the naive clipping of rotor velocities [3], it is possible to make the flight control work exploiting the advantages offered by the use of SIL methodologies in detecting instabilities that might not arise during classical MIL simulations. We published the software as open source available at the link https://github.com/gsilano/BebopS.

References


Università degli Studi del Sannio
Dipartimento di Ingegneria, Benevento.
E-mail: (giuseppe.silano, pasquale.oppido, luigi.iannelli)@unisannio.it