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

Giuseppe Silano, Pasquale Oppido and Luigi Iannelli

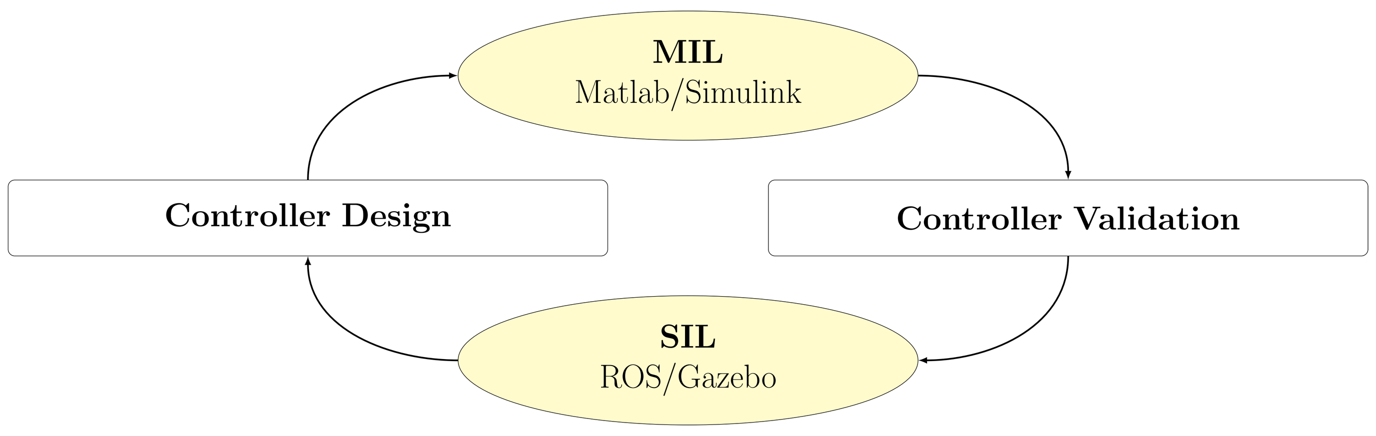
1. Motivation

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.

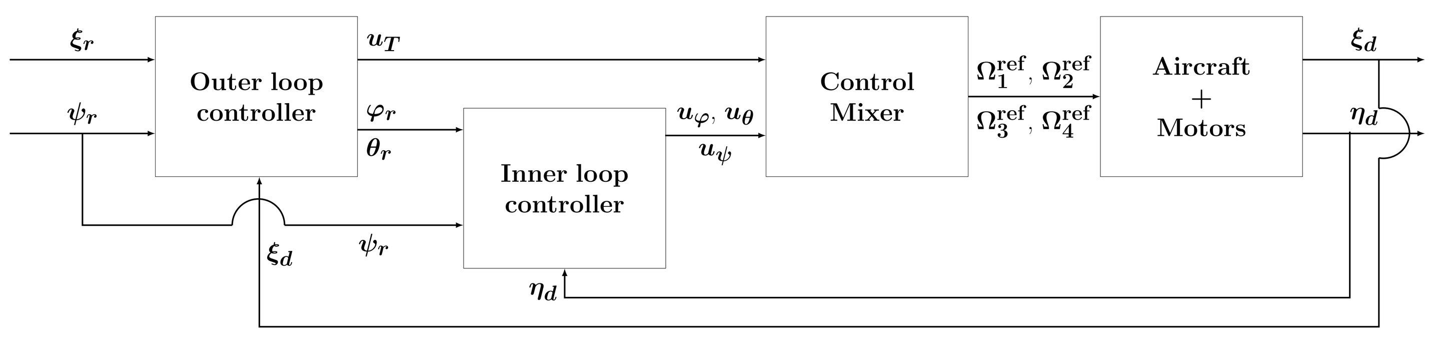


Figure 1: The control scheme. Subscript d indicates the drone variables and r indicates references to controllers.

The position controller (the outer loop controller) uses the measured drone position ξ_d to compute the thrust (u_T) and the attitude (φ_r and θ_r) that should have the drone in order to reach the desired position ξ_r with the desired heading (yaw angle) orientation ψ_r. The attitude controller (the inner loop controller) uses the measured drone attitude η_d to compute the model inputs u_φ, u_θ and u_ψ that should be actuated to achieve the desired attitude (φ_r, θ_r and ψ_r). The control mixer inverts the controller outputs obtaining the commanded motor velocities Ω_i^{ref}, later used as inputs for the aircraft dynamical model (it considers also the rotors dynamics).

Controller equations: The drone position in the world frame is controlled through the virtual inputs

$$u_q = \left(\frac{\alpha_q}{\mu_q}\,\dot{e}_q - \frac{\beta_q}{\mu_q^2}\,e_q\right)m, \quad q \in \{x, y, z\}$$

where $\mu_q > 0$, $\beta_q < 0$ and $\alpha_q = 1 - \beta_q > 0$ are controller parameters to be appropriately chosen and e_q are the components of the position tracking errors. While, the inner controller determines the virtual inputs u_φ, u_θ and u_ψ to regulate the drone attitude according to the following control laws:

$$u_\varphi = I_x\left(\frac{\alpha_\varphi}{\mu_\varphi}\,\dot{e}_\varphi - \frac{\beta_\varphi}{\mu_\varphi^2}\,e_\varphi - \frac{e_\theta e_\psi}{\mu_\theta\mu_\psi}\left(\frac{I_y - I_z}{I_x}\right)\right), \qquad u_\theta = I_y\left(\frac{\alpha_\theta}{\mu_\theta}\,\dot{e}_\theta - \frac{\beta_\theta}{\mu_\theta^2}\,e_\theta - \frac{e_\varphi e_\psi}{\mu_\varphi\mu_\psi}\left(\frac{I_z - I_x}{I_y}\right)\right),$$

$$u_\psi = I_z\left(\frac{\alpha_\psi}{\mu_\psi}\,\dot{e}_\psi - \frac{\beta_\psi}{\mu_\psi^2}\,e_\psi - \frac{e_\varphi e_\theta}{\mu_\varphi\mu_\theta}\left(\frac{I_x - I_y}{I_z}\right)\right),$$

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

$$\text{ISE}(\mu_k, \beta_k) \triangleq \frac{1}{t_f - t_i}\int_{t_i}^{t_f}\left(\|e_\eta(t)\|^2 + \|e_\xi(t)\|^2\right)\text{d}t$$

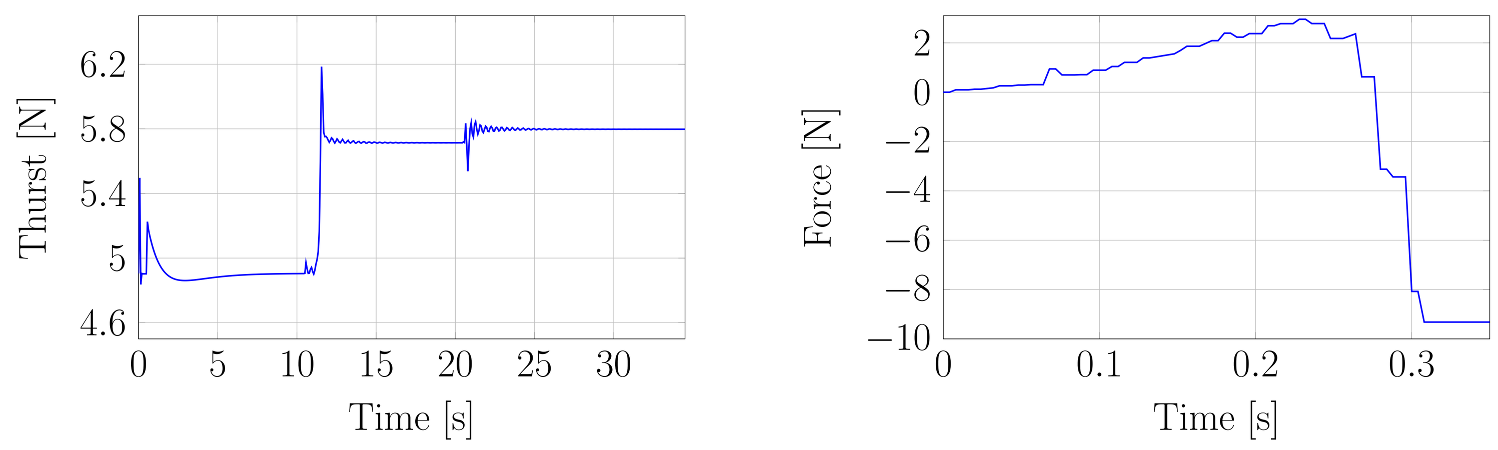
w.r.t. the control parameters μ_k, β_k, $k \in \{x, y, z, \varphi, \theta, \psi\}$ taking into account also motor dynamics, saturation constraints and controller discretization

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 $200\,\text{Hz}$ while the position controller runs at $100\,\text{Hz}$.
- Finally, the **hardware constraints and software implementation**. The controller has to clip the desired rotor velocity so that $0 \leq \Omega_i^{\text{ref}} \leq \Omega_{\text{max}}$. 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.



(a) The total thrust u_T requested by the controller: numerical experiment in MATLAB/Simulink.

(b) Signal u_Z: numerical experiment in Gazebo.

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

By looking at first tenths of second of simulations it comes out that the u_z control signal assumes negative values much below $-mg = -4.9\,\text{N}$, see Fig. 2(b). That is an important issue since $u_z < -mg \Rightarrow u_z + mg < 0$ but (see [3]), that means $u_z + mg = u_T c_{\theta_d} c_{\varphi_d} < 0$. 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 (see [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

[1] Bouzid, Y., Siguerdidjane, H. and Bestaoui, Y., "Nonlinear internal model control applied to VTOL multi-rotors UAV," *Mechatronics*, vol. 47, pp. 49–66, 2017.

[2] T. Nguyen, I. Prodan, and L. Lefèvre, "Flat trajectory design and tracking with saturation guarantees: a nano-drone application," *International Journal of Control*, pp. 1–14, 2018.

[3] G. Silano, P. Oppido, and L. Iannelli, "Software-in-the-loop simulation for improving flight control system design: a quadrotor case study", *IEEE International Conference on Systems, Man and Cybernetics*, 2019.

Grace
Group for Research on Automatic Control Engineering

Università degli Studi del Sannio
Dipartimento di Ingegneria, Benevento.
Web: https://www.ding.unisannio.it
E-mail: {giuseppe.silano, pasquale.oppido, luigi.iannelli}@unisannio.it