Robot Dynamics: Fixed-wing UAVs
Exercise 4: Fixed-wing Simulation and Control

Thomas Stastny, David Rohr

2022.12.07

Figure 1: Techpod UAV.

Abstract

The goal of this exercise is familiarizing oneself with typical fixed-wing UAV dynamics (here, the Techpod UAV - see Fig. 1), how the model derived in the lecture may be simulated, and how one may control the various low- and high- level states of the vehicle. The exercise will be conducted in *MATLAB/Simulink*. All necessary files may be downloaded from the Robot Dynamics course on Moodle `https://moodle-app2.let.ethz.ch/`.

1 Setup/Organization

The simulation is contained within `fw_sim.slx`, see Fig. 2. The blocks within the model are color coded:

- the *red* block contains the vehicle dynamics with aerodynamic/thrusting forces and moments – model parameters specific to the Techpod UAV may be found in the `parameters.m` function. These values were identified from flight data for the nonlinear aircraft equations of motion – but note that the lift/drag curves are *not* defined in the post-stall regime, lateral-directional/longitudinal aerodynamics are decoupled, and, importantly, the model is formulated with Tait-Bryan angles, which will allow for singularities if extreme flight modes are experienced (be careful with your control settings!).

- *cyan* blocks represent low-level control structures.
- *orange* blocks represent high-level control structures.
- *blue* blocks are user-inputs, e.g. manual control deflections, airspeed setpoint, height reference, etc.
- *pink* switches may be double-clicked to change control modes.

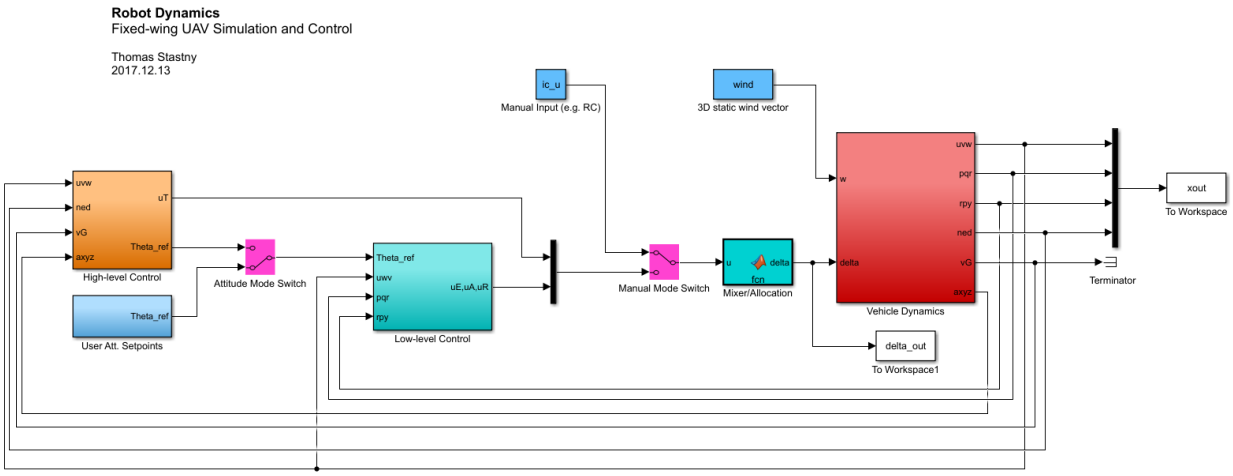

Figure 2: Simulink overview.

Take some time to browse the full model, and pay close attention to the inputs and outputs of each block. In particular – note the component build-up used for force/moment coefficients in the `parameters.m` file, the mixer/control allocation for converting normalized control signals to physical control surface deflections.

Before any simulation is run, the `init.m` script must be run with whatever desired initial conditions set. After simulating – the `plotting.m` script may be used to plot basic outputs from the simulation. Note any modification to this file is more than welcome, the script is only meant as a starting point for your exploration of the fixed-wing modeling/controlling process.

2 Model Dynamics / Open-loop Simulation

Task 1: Set the simulation time length to 15-20 seconds, and run the simulation with all zeros in the manual actuator commands (default configuration when downloading the files). Choose an initial pitch angle of e.g. 5° and airspeed of $14\,\mathrm{m\,s^{-1}}$. Does the aircraft self stabilize in the longitudinal axis? What kind of mode is getting excited during this brief open-loop simulation? (***hint:*** check the extra-slides from the lecture).

Spiral divergence

Task 2: Is it possible to choose one set of manual control deflections (and throttle setting) in order to stabilize *straight*, *level*, and *steady* flight in open loop for the body-x-axis airspeed component $u = 13\,\mathrm{m\,s^{-1}}$? If so, record the control signal (normalized) settings as **trims**, both in the initial condition vector as well as in the low-level control block (u_E, u_A, u_R) and high-level control block (u_T). These will serve as trim deflections for the controller design in the next steps. The blue user-input blocks for the trims take the *normalized* values, i.e. $\in [-1, 1]$. Don't worry to get the perfect trims - as we may refine these later. Further record the steady-state pitch angle θ – input this quantity into the high-level control block's 'θ_{trim}' user input source as well as the "User Att. Setpoints" block just the left of the "attitude mode switch".

3 Control

A general control architecture for fixed-wing UAS is shown in Fig. 3 (borrowed from the *AtlantikSolar*), though not all details are shown in the figure, and not all components of the figure are included in the simulink structure (e.g. G-Load protection). The control structures within this exercise will adhere to this general architecture.

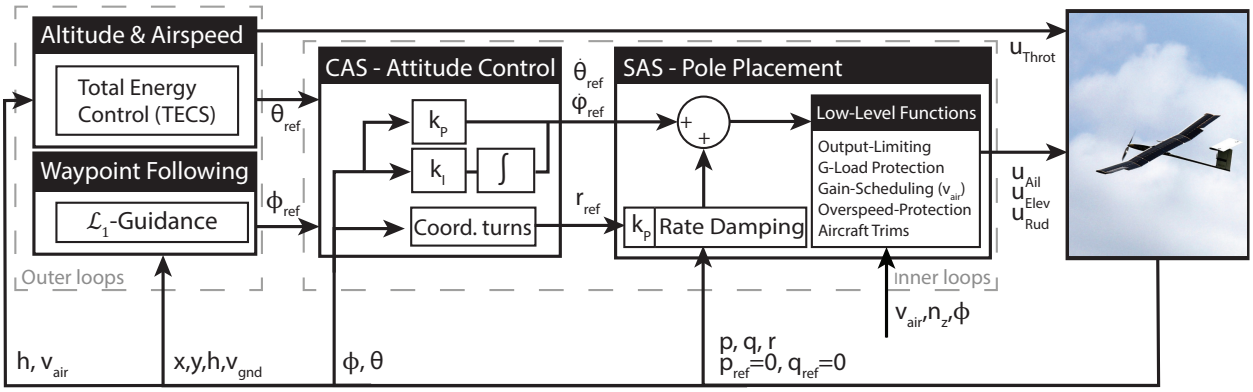

Figure 3: Control Architecture.

3.1 Attitude Stabilization

As seen in the block diagram above, we will use a cascaded-PID control scheme to track attitude setpoints and damp attitude rates. First, explore the low-level control blocks. Note the slight difference in formulation as compared to the cascaded structure shown in the lecture slides.

Task 1: What primary differences in the control structure are present in this architecture when compared to that showcased in the lecture slides? No Low-Level Functions are adopted

Task 2: After exploring the details of the low-level block, please write out the input/output relationships for full the attitude controller (in equations). Pay close attention in the simulink model to the *green* gain blocks – these will be gains which need to be tuned throughout this exercise.

Task 3: Rate damping. Set some small initial guesses for the rate dampers, only the *P-Gains* (proportional gains) of p and q. Leave r for later. A value close to 0.1 is a decent starting point for the dampers. Switch the "Manual Mode Switch" to take inputs from the low-level controller. Freely increase the damper gain for q to observe the damping effect on pitch. A well-tuned damper will, in outdoor conditions, reject small turbulences or gusts, and increase the stability of the trim angle tracking. In the present simulation – note that the trim states for *steady, level, straight* flight are not being tracked, but damped/partially stabilized. This is the first step in tuning the low-level loops!

Task 4: Pitch control. Next, add a small gain (e.g. start with 0.1) to the pitch *P-Gain* within the Attitude Controller's PI block. Slowly increase this value to watch the pitch angle tracking response. In the User Attitude Setpoints block, you may also switch to the step inputs for pitch, this will give a reference for which we can attempt to evaluate the controller response. What final value showed the best tracking peformance without causing too much oscillation? How does the response differ for larger or smaller pitch step inputs?

Task 5: Roll control. Now repeat tasks 3 and 4 for the roll axis, i.e. tune the p damper and the ϕ tracking proportional gain (leave the integrators alone for now). What gains ended up showing the best behavior?

Task 6: Turn coordination. With pitch and roll axes tuned, we may return to the yaw damper. The best way to evaluate the effects of turn coordination is by switching the pitch reference back to a constant, and keeping the roll step inputs. With zero gain on the yaw damper, check the side velocity v plot (which corresponds to sideslip). If we have very small values (e.g. less than $0.5\,\mathrm{m\,s^{-1}}$), then our slip is already well regulated. However, if the aircraft is reaching higher values, the controller may benefit from rudder compensation through the yaw damper. Slowly increase the gain to observe the reduction in sideslip. Record the final value. Also watch the behavior of the angular rates and their respective setpoints.

3.2 Airspeed/Altitude Control

Task 1: TECS. Switch the Attitude Mode Switch to take reference signals from the High-level control block. Inside the block, switch the Airsp/Alt Ctrl Mode Switch to take Δ signals from the TECS block. We will not explore in detail tuning of the TECS block, but it is worth your time to investigate the connections and play with the respective gains to examine the control behavior. The preset gains (on download) are sufficient to marginally track a desired altitude and the given airspeed reference, **assuming the low-level loops are well tuned!** If increasing or decreasing various gains inside TECS, what trade-offs in performance can you observe?

3.3 Lateral-directional Position Control

Task 1: L1 position control. Finally, we will attempt to follow a path in the lateral-directional plane, defined in the *purple* block labeled "Path Definition". Here, you may set a loiter radius, center location (in NED frame), and loitering direction. There is, further, an externally defined L1 distance block, the default value of 150 meters is a conservative first choice for the given flight speed. These inputs will be sent to the L1 control block – a user-defined *MATLAB* function. Open this block, and you will notice the function is empty. The objective of this task is the programming of the L1 position controller defined within the lecture notes, and outputting a sufficiently mapped roll angle reference ϕ_{ref} to achieve the generated normal acceleration commands L1 provides. ***Hint:*** remember the coordinated turn! ***Hint 2:*** Pay careful attention to the L1 distance, and how far in the horizontal plane the aircraft is from the loiter radius. It is impossible to calculate the look-ahead point on the loiter circle if the L1 distance will not reach!

Task 2: L1 position control, part 2. Once the L1 algorithm is programmed into the block, attempt to track various circle radii starting from various initial conditions. What effect does the L1 distance have on the tracking performance? Further, input some wind in the horizontal axis, and evaluate its effect on the tracking performance. ***Reminder:*** Don't forget to switch the Lateral-directional Ctrl Mode Switch to take reference signals from the L1 block! ***Hint:*** There is a variable in the position plotting script that can be set equal to 1 in order to show the prescribed path also on the 3D position plot with the aircraft trajectory.