ASEN 5519 Small UAS Guidance and Control

Homework 5 Assignment

Assigned: Monday, March 6, 2023

Due: 11:59 PM, Tuesday, March 14 2023

Submit all plots and code and through the course web site as a single pdf. Unlike past assignments, this assignment does not specify the exact form or decomposition of all additional functions you may create, nor is a specific control objective given.

Students are given a new version of `ttwistor.m` with all the inertia terms and terms for the linearized model, and the new function `PlotSimulationWithCommands.m` where the green dashed line corresponds to commanded values of the variables plotted based on the input color.

Problem 1

Create the function `[flight_angles] = FlightPathAnglesFromState(aircraft_state)` that returns the vector

$$\text{flight\_angles} = [V_g; \quad \chi; \quad \gamma] \tag{1}$$

where V_g is the ground speed, χ is the course angle, and γ is the flight path angle.

Problem 2

Complete the function
`[control_gains, linear_terms] = CalculateControlGainsSimpleSLC_Nondim_Ttwistor(aircraft_parameters, trim_definition, trim_variables)`.

Using the linear design models described in Lecture 5 and strategies described in Lecture 7 and Lecture 8, this function calculates the gains for the "Simple Successive Loop Closure" autpilot. Although the functions takes as inputs the aircraft parameter file, various aspects

of the design are specific to a given aircraft, thus this function is used only to design gains for the Ttwistor.

The gains are consolidated into the single `control_gains` structure which is used to pass the entire set of gains to the autopilot code.

Problem 3

Look over the function

`[control_input, x_command] = SimpleSLCAutopilot(time, aircraft_state, wind_angles, control_objectives, control_gain_struct)`

which implements the simple successive loop closure autopilot. Students will need to edit portions of the code (search for 'STUDENTS') in order to perform the control design in Problem 4.

Look over the function `RunHW6.m` which determines the aircraft trim, runs the control gain function, and then runs the autopilot. This function runs th e simulation on two separate times scales. The variable Ts specifies the ""sample time" of the control system. The control law calculates a new control input every Ts seconds, and then holds that control input constant for a short simulation of duration Ts. Then, a new control input is calculated and a new simulation is run using the output of the previous iteration as initial condition of the next iteration. The end result of each short simulation is stored as the state and control output. Hence, the final result is a simulation with state and control input at every Ts seconds. **To be clear, students should not modify any code within the main for loop that implements the control law and aircraft equations of motion.**

Problem 4

Tune the autopilot gains using the trim conditions $V_a = 18$ m/s, $\gamma = 0$, and $h = 1805$ m. Assume the Ttwistor aircraft has actuator limits of 45° for the elevator and 30° for the aileron and rudder. Use the steps provided in the lecture notes to tune the longitudinal gains first (holding aileron and rudder at zero), then tuning the lateral dynamics. Use your

own judgment on what you consider "good" performance. However, do not be too strict, i.e. oscillation and overshoot are expected in the responses. The figure below shows the results of one of my simulations, but students are not expected to match these results.

To demonstrate the behavior of your autpilot simulate the system with the following commands (all at once)

1. $V_{a,c} = 18$ m/s.
2. $\chi_c = 40^\circ$.
3. $h_c = 1805$ m.

Initialize the aircraft with the trim state with the exception of the initial height of 1655 m, e.g. on the ground here at Boulder. The variable `control_gains.takeoff_height` is set to 1675 m, which means the aircraft is in 'takeoff mode' beneath this height and 'climb' mode above it.

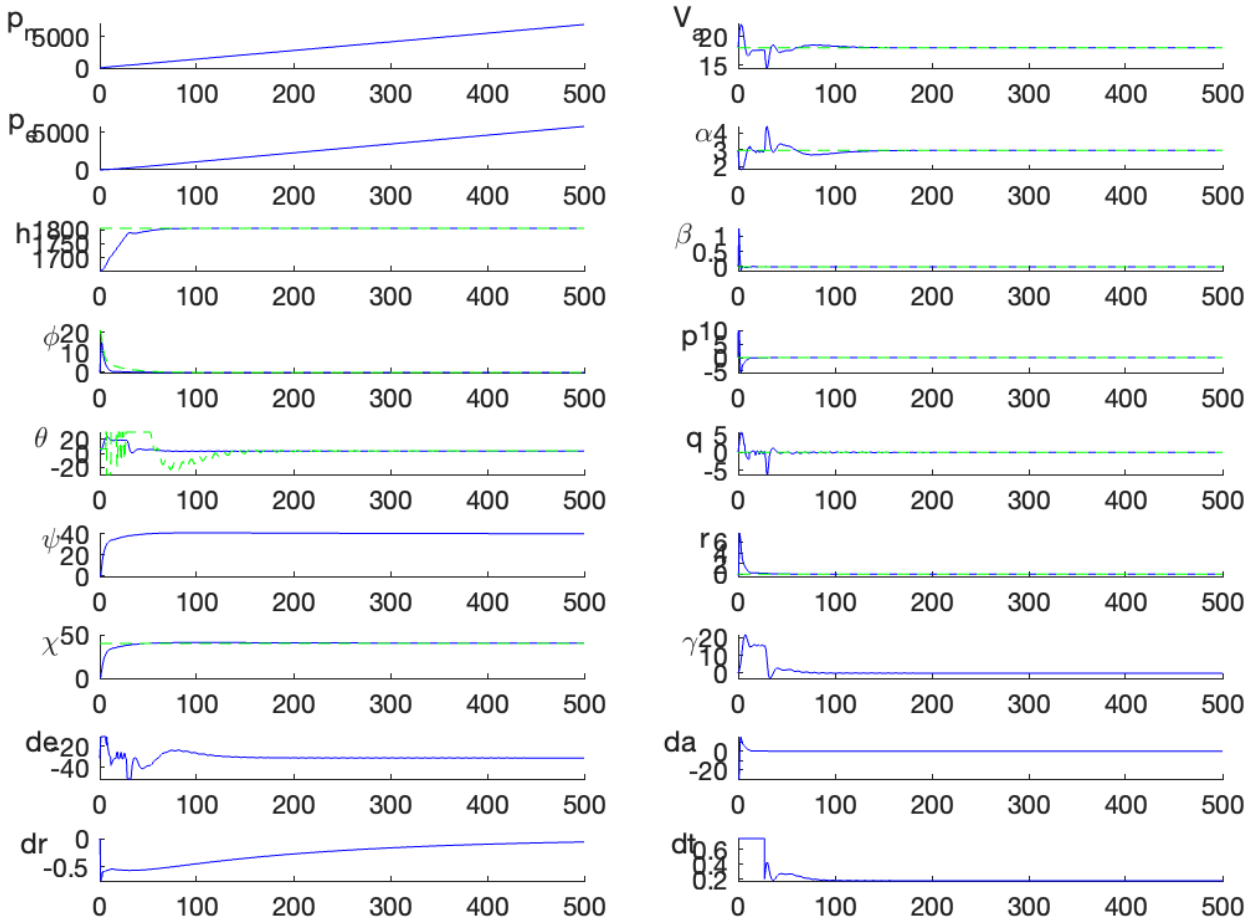

Figure 1: Result from my simulation.