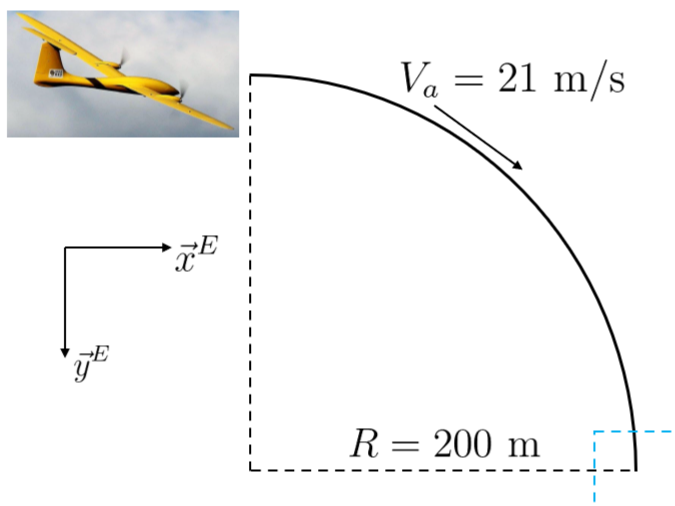
ASEN 3728 Aircraft Dynamics
Programming Homework 4

Due date listed on Gradescope.

In this assignment, you will design a lateral control system for the TTwistor aircraft so that the aircraft can execute a 90 degree turn, as shown below. You are given files for the aircraft dynamics and lateral/longitudinal forces and moments, which you have implemented previously. In order to accomplish this maneuver, you will implement the controller shown in Figure 18.26 without the washout filter W and with linear gains: $J_p = k_p$, $J_a = k_a$, and $J_r = k_r$. The aircraft parameters are given in the Matlab struct returned by the `ttwistor` function.



All files are available by cloning the git repository at `https://github.com/zsunberg/Aircraft-Dynamics-Materials` and navigating to the `assignments/P4` directory. A zip file is also available at `https://github.com/zsunberg/Aircraft-Dynamics-Materials/raw/main/zips/assignments/P4.zip`. It is possible that there will be bugfixes to the assignment after it is released. These will be announced on Piazza.

Task 1. Under the assumption that the 90 degree turn is a coordinated turn, calculate the roll angle required to complete the turn if the airspeed is 21 m/s and the turn radius is 200 m. Also, calculate the time needed to finish the turn if the turn is a perfect quarter-circle. This time, t_{max}, is needed in Task 3. **Indicate your calculated values into the designated location in the `report` pdf.**

Task 2. Design a roll-rate controller (inner loop of the roll controller). First use the `estimateAlat` function to estimate the lateral A matrix for the ttwistor. Then extract $\mathcal{L}_p$ and $\mathcal{L}_{\delta_a}$ from the matrix. Using the pure roll approximation, choose a gain so that the aircraft will achieve a commanded roll rate and the time constant for the roll mode is decreased by 10% from its open-loop value. **Record the value of the gain in the designated location in the `report` pdf.**

Task 3. Design a roll angle controller. **Plot the root locus** for a linear roll angle controller with the form $p_c = k_p(\phi_c - \phi)$ (J_p in Figure 8.26) using the dynamics in the estimated lateral A matrix and the roll-rate controller designed above. Choose a gain value that stabilizes the spiral mode and makes the spiral mode time constant less than 1 second. **Plot the step response** for the roll angle controller with the desired roll angle (ϕ_c in Figure 18.26) as input, and the measured roll angle as output. **Record the value of the gain k_p in the designated location in the `report` pdf.**

Task 4. Design a linear yaw damper with the form $\delta_r = -k_r r$. **Plot the root locus** for k_r and choose a gain that increases the damping ratio to 0.3 or greater and commands a rudder angle of less than 45° for a yaw rate of 0.1 rad/s. **Record the value of the gain k_r in the designated location in the `report` pdf.**

Task 5. Implement a roll controller and yaw damper in `controls.m` to calculate the aileron deflection δ_a and rudder deflection δ_r based on the aircraft state during the turn. You do not need to modify the elevator and thrust controllers that have already been implemented in the `controls` function.

Task 6. Use the existing code under Task 3 of `report.m` to simulate the aircraft's motion with the nonlinear dynamics. The goal for the simulated turn is to arrive inside a box (sketched in blue in the diagram above) defined by $(x^E, y^E, z^E) = (200 \pm 5 \text{ m}, 200 \pm 5 \text{ m}, -1800 \pm 5\text{m})$ after time t_{max}, which you calculated in Task 1. Tweak the values of the gains or the bank angle to achieve the goal. In your report, include the plot produced by `plotStateHistory` plus a plot showing the path of the aircraft in the x^E-y^E plane.

Task 7. Run the `evaluate` function to produce a `submission.json` file that certifies the aircraft arrives at the target with your implementation of `controls`.

Deliverables

In order to use the template files, rename them by removing `TEMPLATE_`. To produce the required report with plots, using the Matlab command `publish('report.m', 'pdf')` is highly recommended. Submit the following files to Gradescope:

- `submission.json` (make sure that the Gradescope autograder runs successfully when you submit!)
- `report.pdf` containing required output from the tasks.
- `controls.m`
- Any additional supporting functions you may have written.