ASEN 3728 Aircraft Dynamics
Programming Homework 2

Due date listed on Gradescope.

In this assignment, you will simulate the motion of a quadrotor in the inertial x-z plane and implement a control law to guide the quadrotor toward a goal state. You will follow the linear control design process outlined in lecture to first choose control gains for the linearized equations of motion and then create a controller for the full nonlinear dynamics.

The quadrotor dynamics, including physical parameters such as mass, moment of inertia, aerodynamic coefficients, and control limits are implemented in the `quadrotorDynamics` function. For the two-dimensional motion, the state vector is $\mathbf{x} = [x_E, z_E, u^E, w^E, \theta, q]^T$. Use $g = 9.81$ m/s^2 for gravity. The initial, trimmed quadrotor state is $\mathbf{x}_0 = [0\text{ m}, 0\text{ m}, 0\text{ m/s}, 0\text{ m/s}, 0\text{ rad}, 0\text{ rad/s}]^T$, and the goal reference state is

$$\mathbf{x}_r = [x_{E,r}, z_{E,r}, u_r^E, w_r^E, \theta_r, q_r]^T = [10\text{ m}, -10\text{ m}, 0\text{ m/s}, 0\text{ m/s}, 0\text{ rad}, 0\text{ rad/s}]^T.$$

To reach this goal, the quadrotor can apply the controls $\mathbf{u} = [Z_c, M_c]^T$, where Z_c is the control force along the body z-axis, and M_c is the control moment about the body y-axis.

The linearized equations of motion are

$$\begin{pmatrix} \Delta\dot{x}_E \\ \Delta\dot{z}_E \\ \Delta\dot{u} \\ \Delta\dot{w} \\ \Delta\dot{\theta} \\ \Delta\dot{q} \end{pmatrix} = \begin{pmatrix} \Delta u \\ \Delta w \\ -g\Delta\theta \\ \Delta Z_c/m \\ \Delta q \\ \Delta M_c/I_y \end{pmatrix}$$

The linear control law that you will implement follows the architecture discussed in class:

$$Z_c = -k_5\Delta w^E - k_6(\Delta z_E - z_{E,r}) - mg$$
$$M_c = -k_1\Delta q - k_2\Delta\theta + k_3\Delta u^E + k_3k_4(\Delta x_E - x_{E,r}).$$

After implementing this linear control law, you will improve the control law to achieve high performance on the nonlinear system.

All files are available by cloning the git repository at `https://github.com/zsunberg/Aircraft-Dynamics-Materials` and navigating to the `assignments/P2` directory. A zip file is also available at `https://github.com/zsunberg/Aircraft-Dynamics-Materials/raw/main/zips/assignments/P2.zip`. It is possible that there will be bugfixes to the assignment after it is released. These will be announced on Piazza.

Task 1. Choose the gains k_5 and k_6 for the linear control law for Z_c. A recommended method for this is the pole assignment strategy used on the homework and exams, but you may use any method. The deliverables for this task are the gain values themselves along with a 1-3 sentence description of how you chose them.

Task 2. Choose the gains k_1, k_2, k_3, and k_4 for the linear control law for M_c. A recommended method for this is the process discussed in the quadrotor control lecture: using pole assignment for k_1 and k_2, selecting

k_4 so that the $x-dynamics$ are an order of magnitude slower than the ϕ dynamics, and then using a root locus to select k_3. The deliverables for this task are the gain values themselves, a plot of the root locus for k_3 (include this regardless of whether you used it to determine the gains), and a 1-3 sentence description of how you chose the gains.

Task 3. Define the A^{cl} and B^{cl} matrices for the closed loop linear system

$$\Delta\dot{\mathbf{x}} = A^{\text{cl}}\,\Delta\mathbf{x} + B^{\text{cl}}\,\mathbf{x}_r.$$

Then, use the `lsim` function to plot the response of the closed loop system. The code in `TEMPLATE_report.m` gives the C and D matrices to create two plots, one that outputs the state values and one that outputs the control values. Verify that the gains that you chose produce acceptable performance (i.e. $\mathbf{x}_r$ is reached in a reasonable time. You may wish to revise the gains to improve performance. The deliverables for this task are the A^{cl} and B^{cl} matrices along with the two plots.

Task 4. Implement this control law in the `quadrotorLinearControls` function and use the `simulateQuadrotor` and `displayTrajectory` functions to visualize the motion of the quadrotor.

The deliverables for this task are the plot produced by `displayTrajectory` and a 1-3 sentence description of why the linear control law performs differently in the nonlinear simulation compared to `lsim`. You do not need to achieve good performance with the linear control law. Optionally, you can animate the simulation with `animateQuadrotor`, but this is not a deliverable.

Task 5. Modify the control law to perform well on the full nonlinear dynamics, and implement the new control law in the `quadrotorControls` function. Modifications could involve changing the control gains or the structure of the control law itself. Use the `simulateQuadrotor` and `displayTrajectory` functions to visualize the motion of the quadrotor. You may wish to additionally use the `plotStateHistory` from assignment P1 to help debug your control law. Below is an example result from `displayTrajectory` showing that the goal is reached in about 16 seconds.

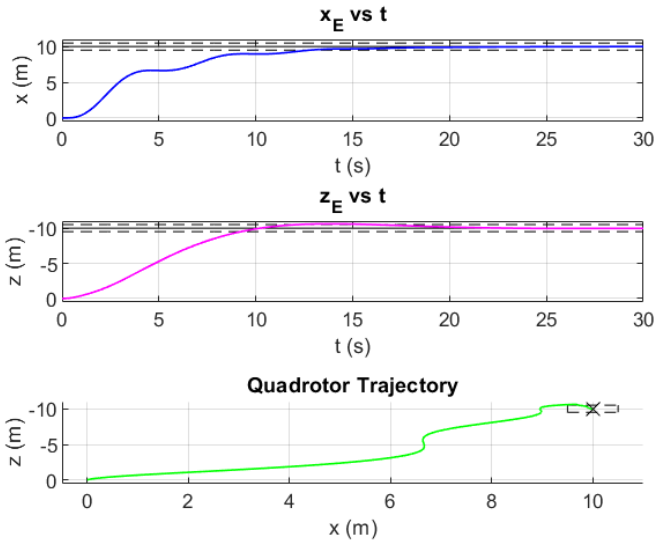

Figure 1: $\mathbf{x}_r$ reached successfully

The deliverables for this task are the plot produced by `displayTrajectory` and 1-5 sentences describing the modifications you made to the linear control law. Optionally, you can animate the simulation with `animateQuadrotor`, but this is not a deliverable.

Task 6. Run the `evaluate` function on your control law. This will give you a score based on how fast the quadrotor reaches the goal and stays within the vicinity of the goal. Full credit is earned if the quadrotor reaches the goal region within 20 seconds. The score then decreases from 100 to 0 as your goal time increases from 20 seconds to 40 seconds. The `evaluate` function will produce a `submission.json` to certify your score. You will upload this file to Gradescope to receive credit for this assignment.

Deliverables

In order to use the template files, rename them by removing `TEMPLATE_`. To produce the 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.
- `quadrotorLinearControls.m`
- `quadrotorControls.m`