DP4: Control of a drone

Patrick Thornton and Krishna Modi
University of Illinois at Urbana-Champaign
Department of Aerospace Engineering

The goal of this report is to develop a controller that allows a quadrotor, which also serves as a drone, to race as fast as possible through rings from start to finish without crashing into the other drones. The report will derive the linear, closed-loop, and asymptotically stable system needed to model the quadrotor that will be compared to a numerical simulation. Our experiments will show that the controller and observer worked efficiently for a run time less than or equal to 30 seconds. To ensure that the drone is able to traverse through all the rings, failures that prevent the drone from reaching the finish line will also be identified, diagnosed, and eliminated.

I. Nomenclature

A, B	=	Matrices that describe the state-space model
C, D	=	Matrices that describe the observer
f_z	=	Net force along the body-fixed z axis (N)
K, L	=	Gain matrices for the controller and observer
LQR	=	Linear Quadratic Regulator
m_e, $\bar{m}_e$, n_e, $\bar{n}_e$	=	Original and new equilibrium values for the state and control input of the system
$\hat{\mathbf{p}}$	=	Vector of the current measured position of the drone (m)
ϕ, ψ, θ	=	Roll, Yaw, and pitch angle (rad)
$\mathbf{p}_{des}$	=	Vector of the desired position values (m)
$p_{ring,x}$, $p_{ring,y}$, $p_{ring,z}$	=	x, y, and z positions of the center of each ring (m)
p_x, p_y, p_z	=	x, y, and z positions of the drone (m)
$\hat{p}_x$, $\hat{p}_y$, $\hat{p}_z$	=	x, y, and z positions of the current measured position of the drone (m)
Q_c, R_c, Q_o, R_o	=	Weight matrices for the LQR controller and observer problems
r	=	Distance between the desired position and the current measured position (m)
τ_x, τ_y, τ_z	=	Net torques about the body-fixed x, y, and z axes (N $\cdot$ m)
u, n	=	Control inputs
v_x, v_y, v_z	=	Linear velocities along the body-fixed x, y, and z axes (m/s)
W_c, W_o	=	Controllability and observability matrices
w_x, w_y, w_z	=	Angular velocities about the body-fixed x, y, and z axes (rad/s)
x, m	=	State of the system
$\hat{x}$, $\dot{\hat{x}}$	=	State estimate and the time derivative of the state estimate of the system
x_{des}, u_{des}	=	Desired state and input of the system
y	=	Output of the sensor model of the system

II. Introduction

A quadrotor consists of four rotors on the drone that produce the net torques and net force, electric motors that drive each of the four rotors, and sensors that provide measurements of the position of the drone, the yaw angle, the position of the center of the next ring, and the position of all the other drones. The design of a quadrotor with sensors is used throughout many applications. For example, DJI is a company that produces commercially available drones for aerial photography and videography [1]. In addition, Ukrspecsystems, a Ukrainian drone maker, produced a quadrotor that is currently being used by the Armed Forces of Ukraine for intelligence, surveillance, reconnaissance, and search-and-rescue missions [2]. Understanding quadrotors will be crucial for future development in terms of parcel delivery, filmmaking, agriculture, and emergency response. This report will work to design a controller and observer that enables a drone to race as fast as possible through rings from start to finish without crashing into the other drones.

This will be done through the use of trajectory tracking, which will enable the drone to travel between the rings. A theoretical model will be developed to explore the system using a linear state-space model followed by showing that the linearized system is controllable and observable. The next steps highlighted in this report will be to design, implement, and test a stable controller and observer, and ensure that the specified requirement is met and validated. Any failures that prevent the drone from reaching the finish line will also be identified, diagnosed, and eliminated.

III. Theory

A. Equations of Motion for the Spacecraft

In the process of designing a proper controller and observer, the equations describing the dynamic system were first linearized so that an equation for the system may be written in state-space form. Given the equations of motion in the form of ODEs as a function, $f(m, n)$, of the system's variables, valid equilibrium points (i.e., the set of variables in $f(m_e, n_e)$ that make $f(m_e, n_e) = 0$) were determined analytically. In the above functions, m is the nonlinear state of the system while n is the nonlinear input of the system. The equilibrium points were chosen such that $f_{z_e} = 4.905$ N and all the other equilibrium variables were equal to 0. These equilibrium values were chosen such that they ensured that the drone was able to maintain stability while flying.

From there, the A and B matrices were determined using the following equations below. In the interest of brevity, the full A and B matrices were omitted from the report, but they are available in the finalized project code.

$$A = \left.\frac{\partial f}{\partial m}\right|_{(m_e, n_e)} = \text{(12 x 12 matrix)} \qquad B = \left.\frac{\partial f}{\partial n}\right|_{(m_e, n_e)} = \text{(12 x 4 matrix)} \tag{1}$$

This process resulted in a linear approximation of the dynamic system in state-space form:

$$\dot{x} = Ax + Bu \qquad x = [m - m_e]^T, \quad u = [n - n_e]^T \tag{2}$$

B. Verifying Controllability of the Spacecraft System

To determine if the spacecraft system is controllable, the controllability matrix W_c below was constructed and verified to be full rank. The rank is defined as the number of independent rows/columns of a matrix. W_c is defined as,

$$W_c = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix} \tag{3}$$

where n is the number of states (i.e., the number of rows in A which is 12 for this project). Through Python implementation, it was found that the rank of W_c is 12, which means that W_c is full rank, and thus the system is controllable.

C. Linearization of the Sensor Model

Just as in Subsection A, the sensor model was linearized by starting with the equations for the sensor model given in the form below.

$$o = g(p_x, p_y, p_z, \psi) = \begin{bmatrix} p_x & p_y & p_z & \psi \end{bmatrix}^T \tag{4}$$

The output was defined as the difference between what the sensor measurements are and what the measurements would be if the system were at equilibrium.

When describing a sensor model in state-space form, the D matrix is typically excluded as it is equal to zero. However, the equation for computing the C matrix is provided below. Again, the C matrix is available in the finalized project code.

$$C = \left.\frac{\partial g}{\partial m}\right|_{m_e, n_e)} = \text{(4 x 12 matrix)} \tag{5}$$

This process resulted in a linear approximation of the sensor model that may be written as shown below.

$$y = Cx = g(m, n) - g(m_e, n_e) \tag{6}$$

D. Verifying Observability of the Sensor Model

In the same manner as controllability, it was determined if the observability matrix, W_o, was full rank. To do so, the observability matrix was constructed as follows:

$$W_o = \begin{bmatrix} C^T & A^T C^T & (A^T)^2 C^T & \cdots & (A^T)^{n-1} C^T \end{bmatrix} \tag{7}$$

Again, n is the number of states, which is 12 for this project. Since the rank of W_o was found to be 12, the observability matrix is full rank ensuring that this system is observable.

E. Determining Controller Gains by LQR

An optimal controller for a system in state-space form is given by

$$u = u_{des} - K(\hat{x} - x_{des}) \qquad K = R_c^{-1} B^T P_c \tag{8}$$

Now, to determine a gain matrix, K, which produces an optimal controller that is stable in theory, the continuous-time infinite horizon Linear Quadratic Regulator (LQR) method was utilized. This method works to minimize a particular integral called the total cost, which is specified below along with the error and effort weights.

$$\underset{u_{[t_0,\infty)}}{\text{minimize}} \qquad \int_{t_0}^{\infty} \left(x(t)^T Q_c x(t) + u(t)^T R_c u(t) \right) dt \tag{9a}$$

$$\text{subject to} \qquad \dot{x}(t) = Ax(t) + Bu(t), \quad x(t_0) = x_0 \tag{9b}$$

$$Q_c = (12 \times 12) \qquad R_c = (4 \times 4) \tag{10}$$

By minimizing the above integral given the specified choice of weights, the best K was calculated to be a 4×12 matrix. To verify that the resulting gain matrix, K produced a theoretically stable controller, the eigenvalues of the $A - BK$ matrix were found and verified that they all had negative real components, which ensured that the system was asymptotically stable at the equilibrium conditions. The K matrix is available in the finalized project code.

F. Determining Observer Gains by LQR

An optimal observer for a system in state-space form is given by

$$\dot{\hat{x}} = A\hat{x} + Bu - L(C\hat{x} - y) \qquad L = P_o C^T Q_o \tag{11}$$

In the above equation, the P_o matrix was obtained through the same method that was used to find P_c.

To determine a gain matrix, L, which produces an optimal observer, the LQR method was used with the error and effort weights described below:

$$Q_o = (4 \times 4) \qquad R_o = (12 \times 12) \tag{12}$$

The best L was calculated to be a 12×4 matrix. Similarly, the eigenvalues of the $A - LC$ matrix were found and verified that they all had negative real components to confirm that the resulting gains produced a theoretically stable observer. The L matrix is available in the finalized project code.

G. Tracking Implementation

In order for the drone to travel through each of the rings and reach a new desired position at every time step, a tracker was implemented within the observer. To start, the equation for the sensor model is given in the previous section in equation 11. In the equation, $\dot{\hat{x}}$ is the time derivative of the state of the system, the A, B, L, and C matrices are calculated in the previous parts, $\hat{x}$ is the state estimate of the system, and y is the output of the sensor model. The variable u is the control input, which is given by the following equation when trajectory tracking is implemented:

$$u = u_{des} - K(\hat{x} - x_{des}) \tag{13}$$

In the above equation, u_{des} and x_{des} are the desired input and state of the system, respectively, and K is calculated in the above sections. In order to find u_{des} and x_{des}, the following equations will be used:

$$u_{\text{des}} = \bar{n}_{\text{e}} - n_{\text{e}} \qquad x_{\text{des}} = \bar{m}_{\text{e}} - m_{\text{e}} \tag{14}$$

In this equation, $\bar{n}_{\text{e}}$ contains the new equilibrium values for the control input, which will be the same as n_{e}, which makes u_{des} equal to zero. Also, $\bar{m}_{\text{e}}$ contains the new equilibrium values for the state of the system, which will be provided by the desired position values in the x, y, and z coordinates while keeping the rest of the parameters at the original equilibrium values since the drone's attitude should remain unchanged for stability purposes. The desired position values that will form the first three elements of $\bar{m}_{\text{e}}$ will be given by $\mathbf{p}_{\text{des}}$, which is defined below:

$$\mathbf{p}_{\text{des}} = \hat{\mathbf{p}} + r\left(\frac{\mathbf{p}_{\text{ring}} - \hat{\mathbf{p}}}{||\mathbf{p}_{\text{ring}} - \hat{\mathbf{p}}||}\right) \tag{15}$$

In this equation, $\hat{\mathbf{p}}$ is the current measured position of the drone, r is the distance between the desired position and the current measured position, and $\mathbf{p}_{\text{ring}}$ is the location of the center of each ring. Now that the desired position has been defined, x_{des} will be defined as equivalent to x, but with the first three rows replaced with the respective entries in $\mathbf{p}_{\text{des}}$.

Now that the desired state and input are defined, these can be incorporated into equation 13 for the sensor model, which is then substituted into equation 11.

IV. Experimental Methods

A. Requirements

The next steps involve designing the controller and observer for the system in a Jupyter notebook using a Python programming language. Once both are implemented, tests will be conducted on the system in simulations run in the Pybullet environment [3]. Before conducting control design, it is first important to define what the goals of the controller and observer are when applied to the system, and how the achievement of this goal can be verified. The requirement placed on the design of the controller and observer is as follows:

> The controller and observer must be designed such that at least 70 out of the 100 simulations are able to maintain a completion time that is less than or equal to 30 seconds. This ensures that the quadrotor is able to race as fast as possible from the start ring to the goal ring, passing through all the rings in between despite the noisy sensor measurements.

B. Verification

To verify that the requirement is satisfied, the initial positions of the rings will be randomized to be at various positions based on the specific seed that is chosen. The seed is a value that allows the randomizer to obtain a set of initial conditions every time. The scope noise for the position and yaw measurements are 0.01 meters and 0.001 radians, respectively, and the distance between the desired position and the current measured position, r, was chosen to be 2.45 meters. To verify the requirement, a loop will be created to run 100 simulations with the maximum time for each simulation set to 45 seconds. For each simulation, the RMSE between the actual and desired positions, RMSE between the estimated and actual positions, the time at which the drone reaches the goal ring, and the time that it takes for the controller to run will be found, calculated, and stored in lists. Additionally, a counter will be initialized and will count the number of simulations that fulfill the requirement stated above. Data for the state estimate and the actual state in the x, y, and z positions of the drone will be found at each time step and plotted to see how accurate the controller and observer are in comparison to the actual data points. Finally, the data for the RMSE between the actual and desired positions, RMSE between the estimated and actual positions, the completion time, and the controller run time will be plotted in histograms for all the simulations. Information from the counter will be utilized to check if the controller and observer meet the requirement.

V. Results and Discussion

A. Identification of Failures

While testing the performance of the drone and the controller, observer, and tracking implementation, several failures were encountered that led to changes in design which then yielded visible improvements in performance:

- Initially, the drone it was visually observed to not leave the starting circle. After examination of the tracking implementation, we determined this was due to the starting position of the drone in relation to the ring. To resolve this, the desired z-coordinate was set to a height of 0.6 m for any time where the drone measured a height less than 0.5 m.
- In early testing, we visually observed that the drone would travel much slower than desired, and would fail to meet the timing goals outlined by our requirement. This was determined to be a result of our low value of 1.0 m chosen for r used in calculating the desired position at each time step (eqn. 15). This was resolved by increasing the value of r.
- We then visually observed that the drone would veer off course and fail to pass through the rings, or would collide with the rings and fail to complete the race. This was due to our value of 3.0 m chosen for r being too high, and was resolved by determining an optimal r value of 2.45 m.

After the completion of system design, there are still some occasional failures that occur which are unlikely to be resolved after further testing.

- For some choices in randomized seeds, the drone occasionally collides with a ring and fails to complete the race. This was determined to occur when the ring placement for a given seed requires the drone to quickly change direction in the y-direction, thus issuing a large torque command that exceeds the maximum torque of the drone. We attempted to resolve this failure by adjusting our controller and observer gains, but these changes hindered performance in other areas. We concluded that this failure was negligible in that the drone would meet the requirements for the majority of randomized seeds.

B. Discussion of Results

Figure 1 shows the estimated and actual positions of the drone measured in the global x, y, and z coordinate frame resulting after the final simulation run. Visually, we observe that the consistent overlapping of the estimated and actual positions of the drone demonstrate that the observer successfully produces an accurate state estimate. In the z-position, however, we note that there is slight discrepancy when there is significant velocity in the z-direction. This is visible between times of 7 and 8 seconds, and between 21 and 22 seconds. The maximum error between these positions in the z-direction is 0.032 m, and this is explored further through RMSE in aggregate results below.

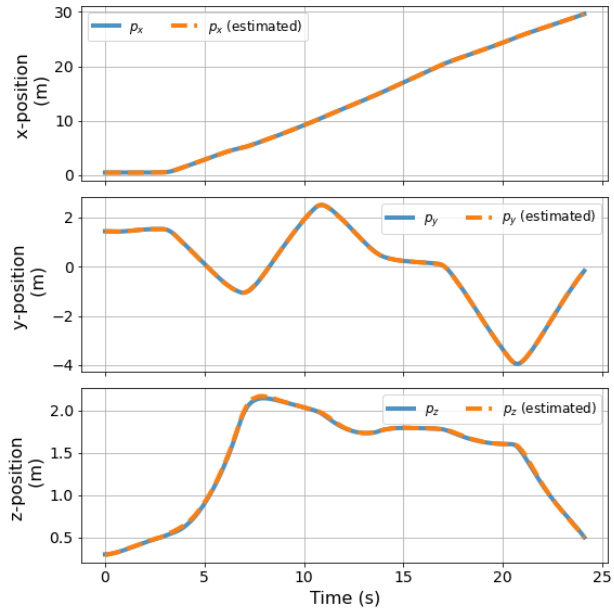


Fig. 1 Plots of actual x, y, z positions and estimated x, y, z positions of the drone for the final simulation ran

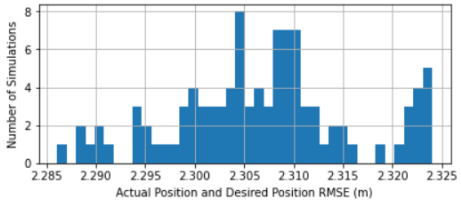
Shown below are the results from 100 simulations run in aggregate as specified in Section IV. In Figure 2a we see the RMSE between actual and desired positions of the drone. Because the controller updates a new desired position at every time step as detailed in Section III, it is reasonable that the RMSE falls between values of 2.285 m and 2.325 m. This

error was observed to vary depending on the choice of parameters in computing $\mathbf{p}_{\text{des}}$ during tracking implementation.

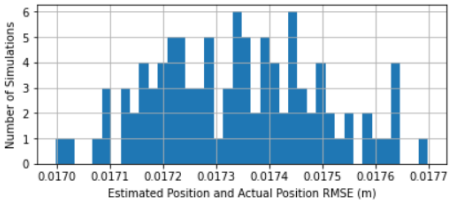
In Figure 2b we see the RMSE between estimated and actual positions of the drone. In this case, the goal is to minimize the RMSE, as lower errors indicate better observer performance.

From Figure 2c, we can calculate that the aggregate simulations were completed with a mean time of 24.0027 s, a median time of 24.0250 s, and a standard deviation of 0.4180 s. The data follows a relatively uniform distribution without any significant outliers in completion time.

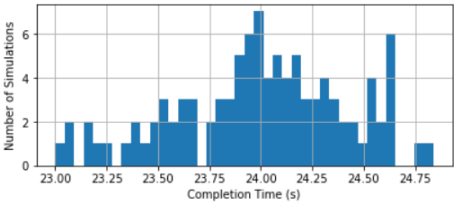
From the numerical data in Figure 2d, we can calculation that the computation time for each instance the controller class was called to have a minimum time of $4.26769 * 10^{-5}$ s, a maximum time of 0.09077 s, and a mean time of $7.40770 * 10^{-5}$ s. While the majority of trials and their respective instances satisfied the run time condition of being less than $1 * 10^{-3}$ s, some instances exceeded this value indicating that the controller implementation will need to be tuned further. Some factors that were observed to have an effect on the computation times include whether or not the simulation was set to display or not, the machine on which the code was run, the background processes running, and the display size of the browser running executing the Jupyter Notebook.



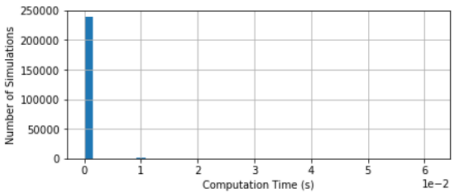
(a) Error between actual position and desired position



(b) Error between estimated position and actual position



(c) Completion times



(d) Computation times

Fig. 2 Histograms of actual and desired position error, estimated and actual position error, completion time, and computation time over all simulations

C. Verification of Requirements in Testing

For the 100 simulations run in aggregate, the final controller, observer, and tracking implementation produced 100 successful completions of the race, meaning that zero drones failed to reach the final ring. Further, as illustrated in Figure 2c and recorded in the implemented counter, 100 out of 100 of the simulations were able to complete the race in under 30 seconds. Thus, the system successfully passed verification for the requirement.

VI. Conclusion

Acknowledgements

References

[1] DJI, "About Us," , 2022. URL `https://www.dji.com/company?site=brandsite&from=footer`, last accessed 13 April 2022.

[2] Praveen, "PC-1 Multipurpose Quadcopter," , 2017. URL `https://www.airforce-technology.com/projects/pc-1-multipurpose-quadcopter/`, last accessed 13 April 2022.

[3] Bretl, T., "AE 353 Course GitHub," , 2022. URL `https://github.com/tbretl/ae353-sp22/tree/main/projects/03_spacecraft`, last accessed 21 April 2022.

Appendix

Day	Task	Person or People
04/12/2022	Worked on code for linearizing the equations of motion, establishing equilibrium points, calculating A, B, C, and D matrices, calculating W and W_{obs}, finding the rank of W and W_{obs}, finding K and L matrices using LQR, and finding the eigenvalues of the A-BK and A-LC matrices	Krishna Modi
04/12/2022	Wrote the Abstract and Nomenclature sections	Krishna Modi
04/12/2022	Started writing the Theory section	Patrick Thornton
04/13/2022	Edited Nomenclature section, and wrote the Introduction section	Krishna Modi
04/14/2022	Completed subsections A through F of Theory section	Patrick Thornton
04/14/2022	Edited Nomenclature section and completed section G of the Theory section	Krishna Modi
04/19/2022	Worked on writing code to implement the controller and observer within the Controller class	Patrick Thornton
04/19/2022	Worked on writing code to implement the controller and observer within the Controller class	Krishna Modi
04/20/2022	Edited the weights for the controller and observer to choose optimal values that ensured the drone moved quickly	Krishna Modi
04/20/2022	Worked on code to plot the histograms and state estimate plots	Patrick Thornton
04/20/2022	Worked on choosing an optimal r value to ensure the drone moved quickly through all the rings	Krishna Modi
04/21/2022	Worked on code for plotting histograms and state estimate plots	Krishna Modi
04/21/2022	Added on to code to consolidate all three coordinate positions in one plot rather than three	Patrick Thornton
04/22/2022	Worked on code for finding the RMSE values	Krishna Modi
04/22/2022	Wrote the Experimental Methods section	Krishna Modi
04/22/2022	Edited the Theory section based on the feedback provided in DP4 Draft 1	Krishna Modi
04/22/2022	Wrote the Results and Discussion section	Patrick Thornton