LQR Control of a Nonlinear Quadrotor System

Keith Chester, Bob DeMont, Sean Hart

November 9th, 2021

Abstract

The purpose of the project is to simulate the control and stable flight of an unmanned four rotor flying vehicle known as a quadrotor. The problem is set into three goals: simulation and control of the quadrotor hover in a stable configuration, intercept a unknown drone's flight path, and return with the captured target quadrotor with randomized disturbance forces while maintaining stable flight.

Introduction

The mission is the defense of a designated airspace. Our quadrotor is assigned to guard a limited airspace, launch if an unidentified UAV enters that airspace, capture the target, and return to base. If the target leaves the airspace before being captured, our quadrotor should return to base. When captured, the target should be assumed to generate external forces and moments on our UAV.

Methodology

General Approach. We'll begin by generating the state space variables and defining the equations that define the state space. We'll prove controllability of the system and choose a controller methodology. Using the controller, we will simulate hovering as well as moving to a location and hovering. Finally we'll introduce external forces and moments that would be experienced by simulating a target capture in the problem statement. In order to simulate the quadrotor and its flight dynamics, the design will be coded into MATLAB, using an ODE solver to compute the state variables, with full state variables, A and B matrices, K gain matrix, $K_{capture}$ gain matrix (incorporating n and r forces and torques from the target), as well as characteristics for graphing.

Frames and Constants. Coordinate frames are identified for the reference/inertial frame $E = [e_1, e_2, e_3]$ at the center bottom of the airspace and the body frame $C = [c_1, c_2, c_3]$. All rotors are equidistant from the center of mass and in the same x-y plane as the body's center of mass. The external forces and moments on the system are represented by $\boldsymbol{r}$ and $\boldsymbol{n}$, where $\boldsymbol{r} = r_1c_1 + r_2c_2 + r_3c_3$ and $\boldsymbol{n} = n_1c_1 + n_2c_2 + n_3c_3$ are directly applied to the center of mass. We are assuming that the torque of the rotor is proportionally related to the input thrust via the constant $\sigma > 0$, for $\tau_i = \sigma u_i$. We will be utilizing $\boldsymbol{I}$ as our diagonal inertial matrix where diagonal elements I_x, I_y, and I_z represent the mass moments of inertia about c_1, c_2, and c_3 respectively. For the purpose of our analysis we will assume simplified aerodynamic effects and ignore drag, ground effects, translational lift, blade flapping, feathering and lagging, and other aerodynamic complexities. The parameters for our state are described in Table 1 of reference [2]

.

State Variable. Development of the quadrotor state equations are found in reference [3]. We'll establish the state variable, $\mathbf{z}$, to represent the quadrotor's xyz center-of-mass position in the earth frame $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$; the roll, pitch, yaw angles in the earth frame, $\boldsymbol{\alpha} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$; the xyz linear velocities in the earth frame, $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^T$; and the angular accelerations $\boldsymbol{\omega} = \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 \end{bmatrix}^T$ in the earth roll pitch yaw directions. Thus our state variable is $\mathbf{z} = \begin{bmatrix} \mathbf{x} & \boldsymbol{\alpha} & \mathbf{v} & \boldsymbol{\omega} \end{bmatrix}^T$. We define $\dot{\boldsymbol{z}}$ as $\begin{bmatrix} \dot{\mathbf{x}} & \dot{\boldsymbol{\alpha}} & \dot{\mathbf{v}} & \dot{\boldsymbol{\omega}} \end{bmatrix}^T$, with:

$$\dot{\mathbf{x}} = \mathbf{v} \tag{1}$$

$$\dot{\boldsymbol{\alpha}} = \mathbf{T}^{-1}\boldsymbol{\omega} \tag{2}$$

$$\begin{aligned} \dot{\mathbf{v}} = -ge_3 + \frac{1}{m}\mathbf{R_{C/E}}(u_1 + u_2 + u_3 + u_4)c_3 \\ + \frac{1}{m}\mathbf{R_{C/E}}r \end{aligned} \tag{3}$$

$$\begin{aligned} \dot{\boldsymbol{\omega}} = I^{-1}((u_2 - u_4)lc_1 + (u_3 - u_1)lc_2 \\ + (u_1 - u_2 + u_3 - u_4)\sigma c_3 + n - \omega \times I\omega) \end{aligned} \tag{4}$$

where $\mathbf{R_{C/E}}$ is defined in the problem statement as the Euler rotation matrix from quadrotor frame C to earth frame E and $\mathbf{T}^{-1}$, also defined in the problem statement, relates the angular rate of change of the Euler angles based on the angular velocity of the quadrotor.

Our system is nonlinear because it is underactuated; four actuators for six degrees of freedom. We can utilize linearization to simplify this model and approximate its behaviour for easier control. To do this, we approximate our system as $\dot{\boldsymbol{z}} = \boldsymbol{A}\boldsymbol{z} + \boldsymbol{B}\boldsymbol{u}$. We derive the $\boldsymbol{A}$ matrix as the Jacobian of our $\dot{\boldsymbol{z}}$ by $\boldsymbol{z}$, and our $\boldsymbol{B}$ as the Jacobian of our $\dot{\boldsymbol{z}}$ by $\boldsymbol{u}$.

For our approximation we model the quadrotor in a stable hovering position, where a given state is $\boldsymbol{z} = \begin{bmatrix} x & y & z & \mathbf{0} \end{bmatrix}^T$; no velocities, accelerations, and its orientation level. We assume a base output of $\frac{mg}{4}$, or each motor outputing the necessary force of a hover. We assign these values to our derived $\boldsymbol{A}$ and $\boldsymbol{B}$ matricies.

We then confirm that our system is controllable with these matricies. We create a controllability matrix $\boldsymbol{C}$, which we define as $\boldsymbol{C} = \begin{bmatrix} A^0B & A^1B & A^2B & \dots & A^{n-3}B & A^{n-2}B & A^{n-1}B \end{bmatrix}$ where n is the size of our input, 12. We check rank of the controllability matrix $\boldsymbol{C}$ to assure that we have rank equal to the number of states in our z vector. As expected, the rank of our derived controllability matrix is indeed 12.

LQR Controller

We selected the Linear-Quadratic Regulator (LQR) approach as an optimal control technique for generating the gain matrix, $\boldsymbol{K}$. LQR operates on our linearized system, utilizing both $\boldsymbol{A}$ and $\boldsymbol{B}$ matricies (states and inputs, respectively). LQR also utilizes two additional matrices - $\boldsymbol{Q}$ and $\boldsymbol{R}$, which correspond to $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively. $\boldsymbol{Q}$'s and $\boldsymbol{R}$'s purpose are to impose a weighted cost such that there will be a trade off between particular elements of the state and inputs in the optimization. Since we are not asked to weigh the cost of using our actuators (as if we were conserving fuel), we have set the 4x4 $\boldsymbol{R}$ matrix as an identity matrix. Our 12x12 $\boldsymbol{Q}$ matrix starts as an identity matrix, but we found through experimentation that our desired performance required adjustments we discuss later. There is an additional matrix, $\boldsymbol{N}$, which acts as a penalty to the interactions between state variables and inputs. We simplify by leaving $\boldsymbol{N}$ as an identiy matrix.

LQR acts on the linearized system of the form $\dot{x} = \boldsymbol{A}x + \boldsymbol{B}u$ and calculates a cost function based on optimization. This integrates the costs of both $\boldsymbol{Q}$ and $\boldsymbol{R}$ for comparison:

$$J = x_0^T F(0) x(t_0) + \int_{t_0}^{t_f} x^T Q x + u^T R u + 2x^T N u \, dt \quad (5)$$

The control input u that minimizes this cost function is $u = -Kx$ with the K value is given by

$$K = R^{-1}(B^T P(t) + N^T) \quad (6)$$

where P is created by solving Ricatti's continuous time differential equation:

$$A^T P(t) + P(t)A - (P(t)B + N)R^{-1}(B^T P(t) + N^T) + Q = \dot{P}(t) \quad (7)$$

and $P(t)$ is bounded by $P(0) = F(0)$. This process is repeated as we experimented with adjusting the weights within the $\boldsymbol{Q}$ matrix.

Once we have our calculated $\boldsymbol{K}$, we define z_d as the desired state of our quadrotor. If we are hovering at a given point in space, it will be $z_d = \begin{bmatrix} x_{1_d} & x_{2_d} & x_{3_d} & \mathbf{0} \end{bmatrix}^T$. When we are chasing a given target through protected airspace, we will set the desired state to $z_d(t) = \begin{bmatrix} x_t(t) & \mathbf{0} \end{bmatrix}^T$ where $x_t(t)$ is the given xyz position of the target at a given time t. We then calculate the error $e(t)$ as the difference between our desired state, or $z_d - z$. We utilize the error to determine the resulting inputs to our system, but we also must take note that since we assumed a linearized system around hovering, we must add an additional force of $\frac{mg}{4}$ to each of our input forces. Thus $\boldsymbol{u} = \boldsymbol{K}e(t) + \begin{bmatrix} \frac{mg}{4} & \frac{mg}{4} & \frac{mg}{4} & \frac{mg}{4} \end{bmatrix}^T$.

Once we have calculated our $\boldsymbol{u}$, we limit the resulting outputs to to $u \,\epsilon\, (0, 3)$ as that is the range of thrust our rotors can generate. We then utilize the MATLAB ODE45 solver, with our previously derived $\dot{\boldsymbol{z}}$ and calculated $\boldsymbol{u}$. At each stage we determine what state of behavior we desire from our quadrotor - hover, move, or intercept - and from this determine the z_d. We then solve for our state variable $\boldsymbol{z}$ at each time increment.

Simulation

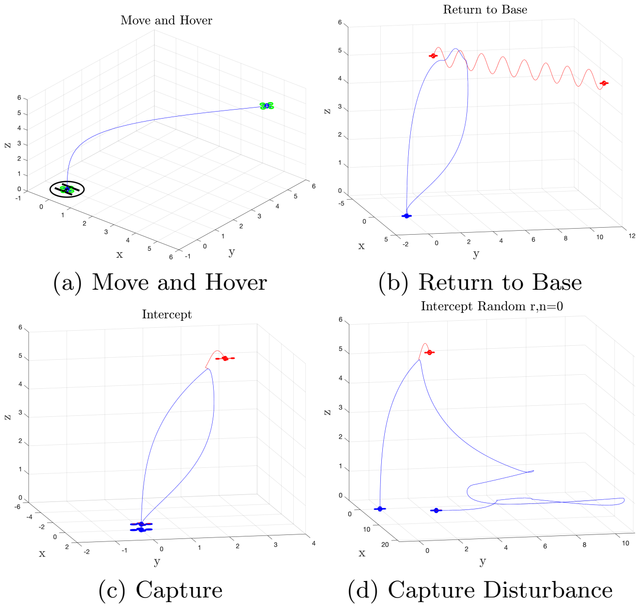
The foundation of simulating the controller is the solution of the system of differential equations of our original system with the added gain from our LQR controller. We have wrapped the changes in the environment into a separate function to serve as the ODE function in order to incorporate time-related changes to the target path as well as n and r torques and forces after capture.

Quadrotor Hovering. To demonstrate successful control of our quadrotor, we establish an initial state of $z = [\mathbf{0}]^T$ and a desired state $z_d = [5, 5, 5, \mathbf{0}]^T$ to demonstrate move and hover. Feeding these into our ODE45 solver, we obtain smooth movement and hovering at the given z_d. Note initial and final states as well as the trajectory plotted in Figure 1a.

Quadrotor Return to Base. A mission objective is to return to base if the target exits the envelope before capture. Figure 1b. demonstrates this performance. In this scenario the interceptor leaves the base to pursue the target, but the target exits the airspace envelope before the interceptor gets close enough to capture.

Quadrotor Interception of Target. For modeling the target UAV's state $z_t = \begin{bmatrix} \boldsymbol{x_t}(t) & \mathbf{0} \end{bmatrix}^T$, where $\boldsymbol{x_t}(t)$ is a continuous function for the target's given x, y, and z at time t. While its path is unknown to the inteceptor quadrotor's control system, we utilize this function to generate a realistic trajectory for the target UAV. It is assumed that the target is intercepted if the quadrotor is within a distance of 0.3 meters of the target. Our simulation demonstrates successful interception of the target after tuning the Q matrix to mitigate errors in the z direction (to enable smoother up and down movement) and penalize x-y errors.

Capture and Return to Base. To model the disturbance forces and torques post capture, we trigger a change in n and r within the code of the ODE function and re-evaluate K to recalculate the control term u for the ODE. Mission parameters asked to reach an $||n|| \leq 1$ (moment) and $||r|| \leq 2$ (force). We began by testing constant moments and forces in each direction before mixing or randomizing since the system must, at a minimum, handle a constant force of the required magnitude before being able to deal with a time-dependent randomized force.



(a) Move and Hover (b) Return to Base

(c) Capture (d) Capture Disturbance

Figure 1: Quadrotor Performance Tests

Results

As noted above, the LQR controller successfully handled move and hover, return to base, and intercept missions. We encountered less successful results with strong disturbances after capture.

Forces. To explore, we began simulating constant force in the c_1 direction to understand how our $Q_{(4,4)}$ term for roll would be impacted. At $Q_{(4,4)} = 1$ and $r = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ the system did not converge on an equilibrium and began a catastrophic tumbling beginning with pitch angle. Due to the tumbling we hypothesized that near level flight, i.e. preventing roll and pitch from exceeding $\frac{\pi}{4}$, would benefit control especially since the forces are acting in the c frame which causes an outsize effect in e. Compensating for the tumbling by increasing $Q_{(4,4)}$ and $Q_{(5,5)}$ to 10 allowed the system to converge but not at our expected home; but rather at $\begin{bmatrix} x & y & z \end{bmatrix} = \begin{bmatrix} 10.8 & 0.023 & 0.30 \end{bmatrix}$. We progressively increased $r = \begin{bmatrix} r_1 & 0 & 0 \end{bmatrix}$ for $r_1 \leq 2$ to attempt to meet mission parameters. None can return to $\begin{bmatrix} x & y & z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$ but some find stable solutions.

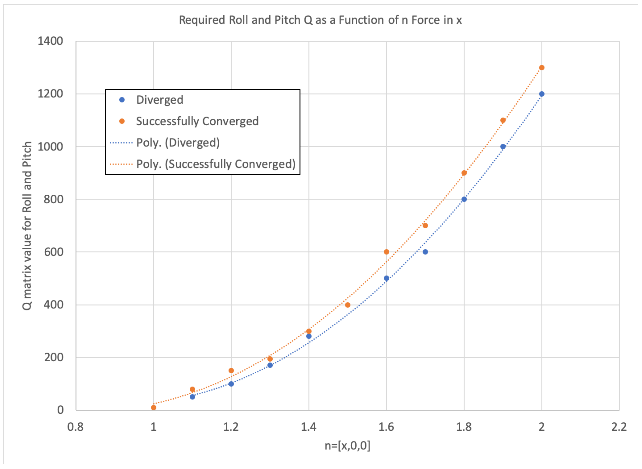


Figure 2: Q as Function of n in x

Figure 2 displays the results as well as approximates a control horizon (between the two lines) below which the system will not converge and above which it will. Solutions close to the horizon exhibit often erratic flight prior to converging. In the $r = (-r_1, 0, 0)$, the results were similar with convergence states in the opposite x direction (as we would expect). Forces, both positive and negative in the r_2 direction, result in similar convergence states with the x and y positions transposed as seen in Table 1.

r	$Q_{(3,4/5)}$	z_{1-6}
$[2, 0, 0]$	[1,1400]	[50.3,-0.2,1.5,0,-0.8,-0.6]
$[-2, 0, 0]$	[1,1400]	[-50.0, 0.4. 3.3,0,1.0,0.1]
$[0, 2, 0]$	[1,1400]	[-0.5, 50.0, 1.5,0.9,0,-0.2,]
$[0, -2, 0]$	[1,1400]	[0.6, -50.0,3.3,-0.9,0,-0.3]

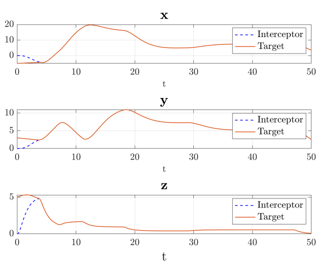
Table 1: State Variables from Constant Forces

Convergences under forces in the r_3 direction benefit from strong $Q_{(4,4)}$ and $Q_{(5,5)}$ but can have far lower penalties before not converging. Note in Table 2 the proximity to home in x and y in the early scenarios (still z=2.8), while the state diverges at $Q_{(4,4)} = Q_{(5,5)} = 145$.

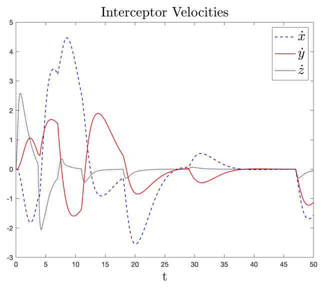
r	$Q_{(3,4/5)}$	z_{1-6}
[0, 0, 2]	[1,1400]	[0.0, 0.0, 2.8,0,0,0]
[0, 0, 2]	[1,1000]	[0.0, 0.0, 2.8,0,0,0]
[0, 0, 2]	[1,150]	[0.0, 0.0, 2.8,0,0,0.1]
[0, 0, 2]	[1,147]	[0.02, 0.01, 2.9,0,0,0.1]
[0, 0, 2]	[1,145]	[1.8, 0.5, 11.9, 0.2,-0.1,0.06]

Table 2: State Variables from Constant Forces

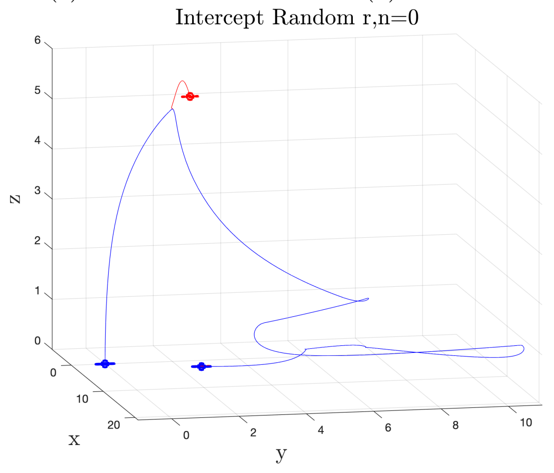
Moments. Similar to the above, we examined constant moments in each of the 3 directions. At $n_1 = .5$ (half mission parameter) we found near-convergence in the neighborhood of $Q_{(4,4)/(5,5)} = 950$ with slight upward drift ($v_z > 0$). At $n_1 = .5$ we could achieve convergence of the equations with $Q_{(4,4)/(5,5)} = 1200$, but again not at the desired state. Beyond, $n_1 = .51$ it was increasingly not possible to converge on any state. With n_2, the convergence horizon at $Q_{(4,4)} = Q_{(5,5)} = 1200$ is between $.4875 < n_2 < .4885$. Surprising to the team, the most sensitive was variation in n_3 where the convergence horizon for $Q_{(4,4)} = Q_{(5,5)} = Q_{(6,6)} = 1000$ is between $.0445 < n_3 < .045$, an order of magnitude smaller. Thus it is impossible to meet mission parameters with both $||n|| \leq 1$ and $||r|| \leq 2$, though it may be possible with only $r < 2$. Attempting this with a random, time dependent force at $Q_{(4,4)/(5,5)} = 1400$, results in no convergence. Though it gets close at times, the continued forces prevent velocity from converging enough for "shut down" as seen in Figure 3.



(a) Positions



(b) Velocities



(c) Trajectory

Figure 3: Random r Performance Tests

Conclusions

The approach utilized within this paper successfully navigates the simpler aspects of the assigned mission - move and hover at a desired location, and intercept an intruding target UAV. It fails, however, to deal with the resulting forces of the target UAV resisting capture. Since the behavior of the quadrotor fails catastrophically when its ϕ or θ increases to an angle above $\frac{\pi}{4}$, this suggests that it is due to our dependence upon linearization around a simplified system at hovering. While LQR allowed us to easily specify multiple weighting pairs of $\boldsymbol{Q}$ and $\boldsymbol{R}$ matricies for behaving differently in various states, it ultimately could not be adjusted to prevent reaching a catastrophic failure state.

Further research would tune the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices more closely. Various external research suggests simply trial and error while others suggest optimization methods. A common method is to weight by the inverse of the maximum allowable error squared [4]. We could also enhance the process to store target position to estimate the target velocity and acceleration vectors to anticipate next target position. This could improve time to intercept and allow use of $\boldsymbol{Q}$ weightings for velocity components. Similarly, attempting to predict target trajectory allows setting the z_d in our control function to a likely future state of the target, resulting in a quicker intercept. More advanced analysis could include utilizing the $\boldsymbol{R}$ matrix to penalize usage of motors to minimize power expenditure and maximize battery life and thus range of coverage and mission lifespan.

After this experiment, these authors would recommend not utilizing linearized systems when dealing with expected external and unpredictable perturbances.

References

[1] Jinho Kim, S. Andrew Gadsden, Stephen A . Wilkerson. "A Comprehensive Survey of Control Strategies for Autonomus Quadrotors". arXiv:2005.09858v1. 20 May 2020.

[2] Faal, Siamak. Project Statement. Class notes for RBE 502. Spring 2021.

[3] Faal, Siamak. Notes on Control of Quadrotors. Class notes for RBE 502. March 3, 2021.

[4] Murray, Richard M. Optimization Based Control. California Institute of Technology. January 4th, 2010.