Design, Implementation, and Testing of Drone Controller Using Optimal Observer and Controller

Ash Miller

The purpose of this paper is to design, implement, and test a controller for a small scale drone, which utilizes an observer to take partial measurments of the drone system, and create a state space estimation which estimates the entire system state. The estimated system state will then be used alongside a controller to determine the neccesarry outputs for the system which brings it to equilibrium. The purpose of the observer/controller will be to guide the drone through a series of rings in a simulated 3D environment. Additionally, the controller and observer are designed in such a way so as to minimize the time to get to/through the ring(s). By solving the linear quadratic regulator problem (LQR) we can optimize the controller and observer to achieve this goal while also maintaning stability throughout flight.

I. Nomenclature

p_x	=	x position of drone (m)
p_y	=	y position of drone (m))
p_z	=	z position of drone (m)
ψ	=	yaw angle (rad)
θ	=	pitch angle (rad)
ϕ	=	roll angle (rad)
v_x	=	linear velocity along the body-fixed x axis (m/s)
v_y	=	linear velocity along the body-fixed y axis (m/s)
v_z	=	linear velocity along the body-fixed z axis (m/s)
ω_x	=	angular velocity about the body-fixed x axis (rad/s)
ω_y	=	angular velocity about the body-fixed y axis (rad/s)
ω_z	=	angular velocity about the body-fixed z axis (rad/s)
τ_x	=	net torque about the body-fixed x axis (N*m)
τ_y	=	net torque about the body-fixed y axis (N*m)
τ_z	=	net torque about the body-fixed z axis (N*m)
f_z	=	net force about the body-fixed z axis (N)
Qc,Qo	=	positive semidefinite matrices
Rc, Ro	=	positive definite matrices
K	=	controller gain matrix
W	=	controllability matrix
L	=	observer gain matrix
O	=	observability matrix
o	=	sensor model
$\hat{x}$	=	system state estimate

II. Introduction

Given a set of differential equations which describe the equations of motion for a drone, throughout this paper we will demonstrate the theory and then application of optimal observer and optimal controller design through solving the linear quadratic regulator problem. The model will first have to linearize the equations of motion, and then solve for the state-space system variables. Additionally, our output for our system, usually characterized by the variable, y, will represent our sensor measurments made at the front and back of the drone. Once we've linearized our equation around an equilibrium point and verified equilibrium, we need to determine whether the system is then controllable and observable. After that, we need to design our 4 weighted matrices, Qc, Rc, Qo, and Ro, which will be used in LQR to

determine the pole-placement of our K and L seperately. Once the model is designed we need to implement it into the acual drone in simulation using an our observer to estimate the system state, $\hat{x}$, and then controlling the torque and force outputs through our controller based on $\hat{x}$. Through the use of manipulating our equilibrium point we can determine the motion of the system, which is what allows the drone to fly through all of the rings on the entire course and reach the end quickly.

III. Model Design

Given the following equations of motion for the system:

$$\begin{bmatrix} \dot{p_x} \\ \dot{p_y} \\ \dot{p_z} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \\ \dot{\omega_x} \\ \dot{\omega_y} \\ \dot{\omega_z} \end{bmatrix} = f(p_x, p_y, p_z, \psi, \theta, \phi, v_x, v_y, v_z, \omega_x, \omega_y, \omega_z, \tau_x, \tau_y, \tau_z, f_z)$$

where f =

$$\begin{bmatrix} v_x \cos(\psi)\cos(\theta) + v_y(\sin(\phi)\sin(\theta)\cos(\psi) - \sin(\psi)\cos(\phi)) + v_z(\sin(\phi)\sin(\psi) + \sin(\theta)\cos(\phi)\cos(\psi)) \\ v_x \sin(\psi)\cos(\theta) + v_y(\sin(\phi)\sin(\psi)\sin(\theta) + \cos(\phi)\cos(\psi)) - v_z(\sin(\phi)\cos(\psi) - \sin(\psi)\sin(\theta)\cos(\phi)) \\ -v_x \sin(\theta) + v_y \sin(\phi)\cos(\theta) + v_z \cos(\phi)\cos(\theta) \\ \frac{w_y \sin(\phi) + w_z \cos(\phi)}{\cos(\theta)} \\ w_y \cos(\phi) - w_z \sin(\phi) \\ w_x + w_y \sin(\phi)\tan(\theta) + w_z \cos(\phi)\tan(\theta) \\ v_y w_z - v_z w_y + \frac{981 \sin(\theta)}{100} \\ -v_x w_z + v_z w_x - \frac{981 \sin(\phi)\cos(\theta)}{100} \\ 2f_z + v_x w_y - v_y w_x - \frac{981 \cos(\phi)\cos(\theta)}{100} \\ \frac{10000\tau_x}{23} - \frac{17 w_y w_z}{23} \\ \frac{10000\tau_y}{23} + \frac{17 w_x w_z}{23} \\ 250\tau_z \end{bmatrix}$$

And given the following sensor model, o = $g(p_x, p_y, p_z, \psi, \theta)$:

where o =

$$\begin{bmatrix} p_x + \frac{7\cos(\psi)\cos(\theta)}{40} \\ p_y + \frac{7\sin(\psi)\cos(\theta)}{40} \\ p_z - \frac{7\sin(\theta)}{40} \\ p_x - \frac{7\cos(\psi)\cos(\theta)}{40} \\ p_y - \frac{7\sin(\psi)\cos(\theta)}{40} \\ p_z + \frac{7\sin(\theta)}{40} \end{bmatrix}$$

The first step in our model design is to linearize the equations of motion of the system and the sensor model to derive expressions for our state-space system, given by:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{1}$$

In order to linearize the system we first have to create two dummy matrices, one which represents our states and one which represents our inputs. these dummy matrices are ω and p respectively. Our set of first-order ODE's will then be $\dot{w} = f(w, p)$. Now we can find the taylor expansion of f(w,p) to approximate our linear system. Before performing our taylor expansion, we first must find an equilibrium point to linearize around. Our equilibrium point will resolve the equation $f(w_e, p_e) = 0$. Therefore, choosing the equilibrium points, $w_{1e}, w_{2e}, w_{3e}, ..., w_{12e} = 0$, $p_{1e}, p_{2e}, p_{3e} = 0$, and $p_{4e} = 9.81/2$ resolves the equilibrium equation. The state, and input are defined as follows: $x = w - w_e$, and $u = p - p_e$. We will also define the output equation, r = g(w,p), where g is our sensor model and find the equilibrium point of the output, $r_e = g(w_e, p_e)$. The output then is, $y = r - r_e$. Then the state-space space equation (1), will have A, B and C defined as, $A = \frac{\partial f}{\partial w}|_{(w_e,p_e)}$, $B = \frac{\partial f}{\partial p}|_{(w_e,p_e)}$, and $C = \frac{\partial g}{\partial w}|_{(w_e,p_e)}$, i.e. the jacobian of f or g with respect to w or p, evaluated at the equilibrium point. We thereby find A = $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{981}{100} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{981}{100} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$, B = $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 \\ \frac{10000}{23} & 0 & 0 & 0 \\ 0 & \frac{10000}{23} & 0 & 0 \\ 0 & 0 & 250 & 0 \end{bmatrix}$, and C = $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & \frac{7}{40} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -\frac{7}{40} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -\frac{7}{40} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \frac{7}{40} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$. Now that we've linearized the system around our chosen equilibrium point, the next step is to find whether or not the system is controllable and observable. If a system is controllable and observable, then the eigenvalues of the system can be chosen arbitrarily by performing pole placement. A system is asympytotically stable when all eigenvalues of (A-BK) have strictly negative real parts, so by designing our system with all negative eigenvalues we can assure asympytotic stability. The system is controllable and observable when our controllability and observability matrces are full rank. $W = [B, AB, A^2B, ..., A^{n-1}B]$ and $O = \begin{bmatrix} C \\ CA \\ CA^2 \\ ... \\ CA^{n-1} \end{bmatrix}$. Both W and O were calculated and were both found to have a rank of 12. Since there are 12 states in the system, the W and O are full rank and therefore the system is observable and controllable. Now to solve for the optimal controller and optimal observer we will solve the LQR problem for each. The solution to the LQR problem for our optimal controller is what's known as the Hamilton-Jacobi-Bellman equation. Solving for this equation in the infinite-horizon LQR problem requires two additional matrices, those being Qc and Rc. In LQR, Qc and Rc are parameters used to trade-off error with effort. If Q is much larger than R, then our controller will

penalize the state more heaviliy, meaning it will put more effort into bringing the state to zero with larger control inputs. Conversly, if R is much larger than Q than the system will emphasize stability with small control inputs. As such, for our chosen controller, Qc and Rc were chosen such that our Rc will be larger than Qc, so as to emphasize stability. With our A,B,Qc, and Rc matrices we can solve for the control matrix K, by first finding P, which is found by solving the continuous-time algebraic Riccati equation. Once P is found, K can be expressed as $K = R^{-1}B^TP$. For our system, K = $\begin{bmatrix} 0 & -0.1 & 0 & 0 & 0 & 0.948 & 0 & -0.171 & 0 & 0.12 & 0 & 0 \\ 0.1 & 0 & 0 & 0 & 0.948 & 0 & 0.171 & 0 & 0 & 0 & 0.12 & 0 \\ 0 & 0 & 0 & 0.707 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.125 \\ 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0.332 & 0 & 0 & 0 \end{bmatrix}$. We find L, similarly, except instead our inputs for the LQR will be A^T, C^T, R_o^{-1}, and Q_o^{-1}. where $L = LQR(A^T, C^T, R_o^{-1}, Q_o^{-1})^T$. For our observer, if Ro is much larger than Qo, then our trajectory will correspond more to a linear model, and if our Qo is much larger than Ro, then our trajectory will correspond mor to the observations. As such we've chosen a large Ro such that our observer trajectory will be heavily impacted by our sensor measurments. L then is,

$$L = \begin{bmatrix} 11.277 & 0 & -0.72 & 11.277 & 0 & 0.72 \\ 0 & 12.205 & 0 & 0 & 12.205 & 0 \\ 0 & 0 & 10.488 & 0 & 0 & 10.488 \\ 0 & 7.573 & 0 & 0 & -7.573 & 0 \\ 4.114 & 0 & -6.036 & 4.114 & 0 & 6.036 \\ 0 & -10.0 & 0 & 0 & -10.0 & 0 \\ 27.687 & 0 & -17.118 & 27.687 & 0 & 17.118 \\ 0 & 48.954 & 0 & 0 & 48.954 & 0 \\ 0 & 0 & 10.0 & 0 & 0 & 10.0 \\ 0 & -10.0 & 0 & 0 & -10.0 & 0 \\ 3.666 & 0 & -9.304 & 3.666 & 0 & 9.304 \\ 0 & 10.0 & 0 & 0 & -10.0 & 0 \end{bmatrix}$$

Due to the fact that our sensors only provide measurments for less than half of the system's states, we also need to design a system state estimate which continually updates estimates of the current system states such that we can apply our controller to the system state estimate, $\hat{x}$, to determine our output torques and forces. our system state estimate is defined by the following state space equation:

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) - L(C\hat{x}(t) - y(t)) \tag{2}$$

where

$$u(t) = -K\hat{x}(t) \tag{3}$$

additionally, we can recursively calculate $\hat{x}$ using the approximate update rule

$$\hat{x}(t + \Delta t) \approx \Delta t(A\hat{x}(t) + Bu(t) - L(C\hat{x}(t) - y(t)) \tag{4}$$

choosing the time step, $\Delta t = 0.04$. Additionally, $\hat{x}(0) = x(0)$, as we are able to define our initial conditions. Now to implement trajectory tracking, which will allow the drone to fly through all the rings without colliding, I created a method for updating the equilibrium point of the system based on a linear combination of drone's current location, the vector which directs the drone to the center of the ring, and the rings normal vector. The thought process behind this design was based on the principle that the closer the equilibrium point is to the location of the drone, the less required effort for the controller to bring the system to equilibrium, aka bringing the drone to the new equilibrium point. In this way, by taking the position of the drone, and then adding some factor of the unit vector pointing from the drone's position to the center of the ring, the new equilibrium point will essentially be some slight distance along the line connecting the center of the drone to the center of the ring. I also decided to add a factor which weighs the importance of the rings normal vector. This was based on the idea that by targeting slighty in front of center of the ring in the direction of its normal vector, since throughout the course the rings come together in such a way so as to create relatively smooth curves, then this implies the normal vectors of rings which are close together will also have similar values. As such by adding the weighted normal vector towards the calculation of the targeted equilibrium point, the

drone will be "pushed" along the path of rings. This primarily increases the speed of the drone at a cost of stability. the equation of the updating equilibrium point is then given by,

$$x_{goal} = p_{drone} + w1 * (p_{ring} \hat{-} p_{drone}) + w2 * \hat{n} \tag{5}$$

where p_{drone} is the location of the drone, (p_x, p_y, p_z), p_{ring} is the location of the center of the next ring, and n is the unit normal vector to the ring. w1 and w2 are both weights. In our system we chose w1 = 4 and w2 = 0.5. We can then determine our outputs using our controller alongside our state estimate and equilibrium point, solving equation (3) for

$$u = -K(\hat{x} - x_{goal}) \tag{6}$$

and our inputs are equal to u+p_e. we can thus extract the torques neccesarry to apply to the drone where $tau_x = u[0], tau_y = u[1], tau_z = u[2]$, and $f_z = u[3] + 9.81/2$

IV. Experimental Methods

In our implementation of the observer and controller into our drone system for the use of generating input torques and forces which guide the drone through a course of rings while minimizing time to travel through the rings and ensuring the drone does not collide with any rings. In order to test this system, I will run 5 seperate drone simulations on 5 randomly generated courses. I will then plot the trajectories of the drone as well as the rings to determine whether or not the drone collides with any rings, and I will record the end times each simulation to confirm it completes the course. As a rigid requirement I will say that if the drone can complete every course in under 35 seconds it successfully minimizes the time to travel through the rings. The following are the results of each trial, including the time to completion, a graph of the trajectory of the drone, and a graph of the trajectory of the drone alongside the rings, including the seed.

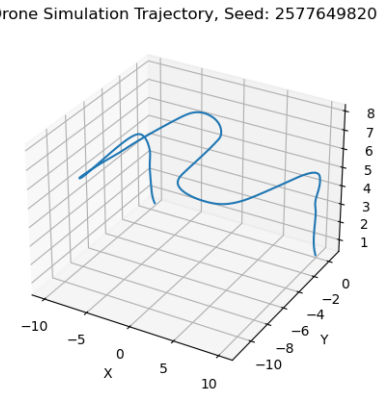
V. Results and Discussion

Below are the results of five trials

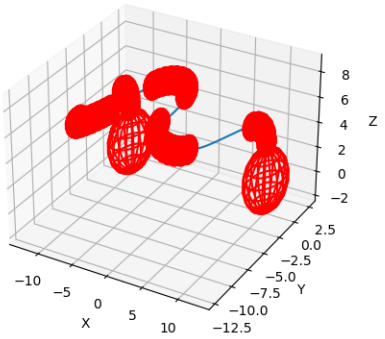
Trial 1

```
FINISHED: drone "template" at time 31.12
Simulated 779 time steps in 31.1712 seconds (24.9910 time steps per second)
```

Time to Complete Course



Drone Simulation Trajectory alongside Rings, Seed: 2577649820

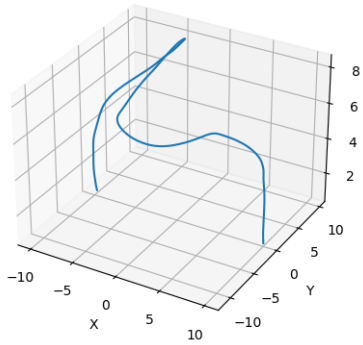


Trial 2

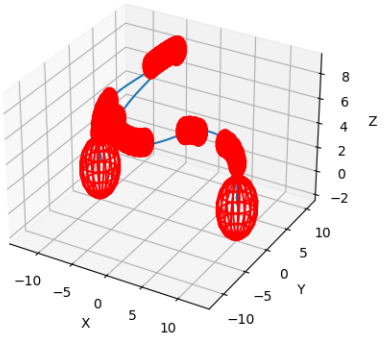
```
FINISHED: drone "template" at time 31.48
Simulated 788 time steps in 31.5262 seconds (24.9951 time steps per second)
```

Time to Complete Course

Drone Simulation Trajectory, Seed: 1019664882



Drone Simulation Trajectory alongside Rings, Seed: 1019664882

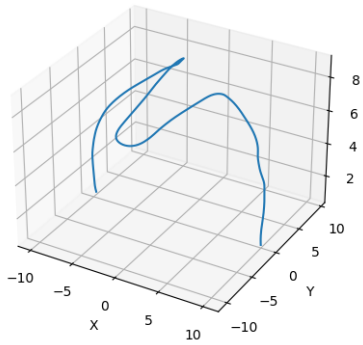


Trial 3

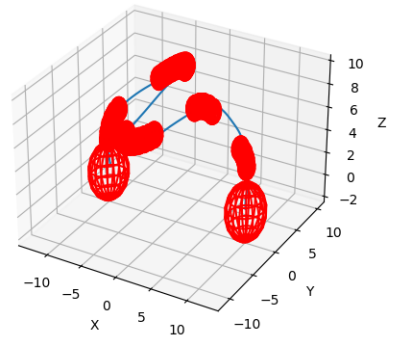
```
FINISHED: drone "template" at time 33.80
Simulated 846 time steps in 33.8510 seconds (24.9919 time steps per second)
```

Time to Complete Course

Drone Simulation Trajectory, Seed: 1809375842



Drone Simulation Trajectory alongside Rings, Seed: 1809375842

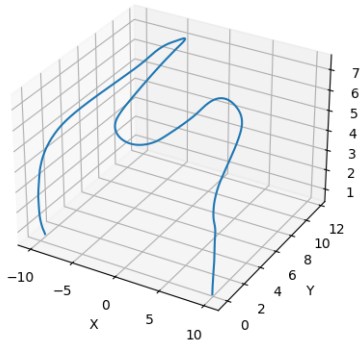


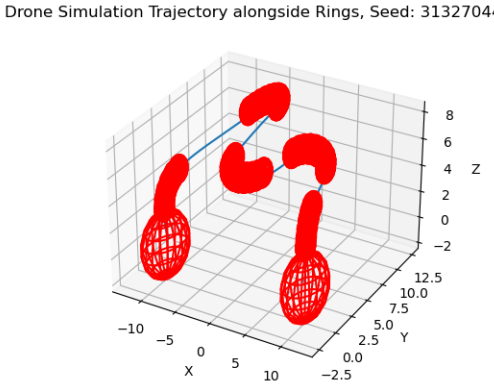
Trial 4

```
FINISHED: drone "template" at time 29.72
Simulated 744 time steps in 29.7669 seconds (24.9942 time steps per second)
```

Time to Complete Course

Drone Simulation Trajectory, Seed: 3132704445

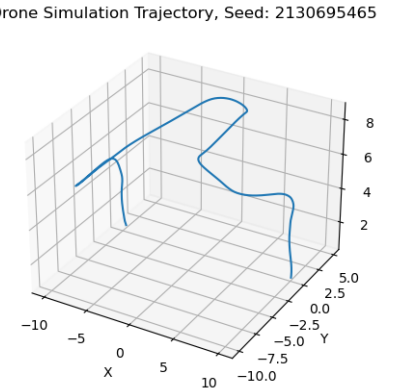


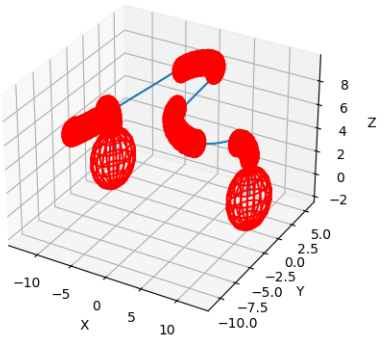


Trial 5

```
FINISHED: drone "template" at time 34.00
Simulated 851 time steps in 34.0559 seconds (24.9883 time steps per second)
```

Time to Complete Course





These results confirm our requirements, as the drone did not collide with any rings, and in all five trials the drone successfully navigated through the ring course in under 35 seconds.

VI. Conclusion

In conclusion, the design, implementation, and testing of a drone controller using an optimal observer and controller have proven successful in guiding the drone through a series of rings in a simulated 3D environment. The experimental results from the five trials demonstrate the effectivness of the controller/observer design alongside the implemented trajectory tracking method. Still however, there can doubtless be improvements to the design of the controller and observer alongside a more sophisticated trajectory tracking, perhaps one which varies the weights based on distance to the ring or one which attempts to predict the location of the next ring to improve routing. Trajectory tracking could

also incorporate information about the entire ring, as their radii are known ahead of time. This could again adjust the weights such that the closer the drone gets to the boundary of the ring, which could be calculated with the position of the ring, the rings normal vector, and the ring's radius, which are all known ahead of time, the more the weights value pushing the equilibrium point towards the center of the ring and less along its normal vector. Additionally, the weights of our Q and R matrices could likely be improved, making the drone more stable or possibly quicker.