# DP3: Optimizing Control Dynamics for Amateur Drone Racing: A Study in Advanced Quadrotor Maneuverability

Jadyn Chowdhury[1]

*DP1-AE 353: Aerospace Control Systems,, Urbana, IL, 61820, United States of America*

**This study presents the design and analysis of a control system for a drone, intended for use in amateur drone racing. Emphasizing linearization, controller and observer design, along with controllability and observability, the report details the development of a stable, efficient control mechanism. The system's performance is evaluated through simulations, demonstrating the drone's ability to navigate through a pre-defined course with precision and speed. The project's success is anchored in its thorough theoretical foundation, rigorous mathematical modeling, and effective implementation of control strategies.**

## I. Nomenclature

$p_x$ = x position ($m$)
$p_y$ = y position ($m$)
$p_z$ = z position ($m$)
$\psi$ = yaw angle ($rad$)
$\theta$ = pitch angle ($rad$)
$\varphi$ = roll angle ($rad$
$v_x$ = linear velocity along the body-fixed x axis ($ms^{-1}$)
$v_y$ = linear velocity along the body-fixed y axis ($ms^{-1}$)
$v_z$ = linear velocity along the body-fixed z axis ($ms^{-1}$)
$\omega_x$ = angular velocity about the body-fixed x axis ($rads^{-1}$)
$\omega_y$ = angular velocity about the body-fixed y axis ($rads^{-1}$)
$\omega_z$ = angular velocity about the body-fixed z axis ($rads^{-1}$)
$\tau_x$ = net torque about the body-fixed x axis ($Nm$)
$\tau_y$ = net torque about the body-fixed y axis ($Nm$)
$\tau_z$ = net torque about the body-fixed z axis ($Nm$)
$f_z$ = net force along the body-fixed z axis wheel ($N$)

## II. Introduction

The increasing popularity of drone technology, particularly in amateur racing and critical applications like search and rescue, necessitates advanced control systems. This project aims to design, implement, and test a controller for a quadrotor drone, ensuring high-speed, agile flight. The project's scope includes linearizing the drone's motion equations, ensuring controllability and observability, and developing a stable controller and observer. The goal is to create a system capable of navigating complex courses swiftly while maintaining stability and responsiveness.

[1]Aerospace Engineering and Computer Science, Grainger School of Engineering at UIUC

## III. Theory and Model

### A. Linearization

The motion of each drone is governed by the by ordinary differential equations with the following form,

$$\begin{bmatrix} \\ \\ \\ \\ \\ \\ \dot{p_x} \\ \dot{p_y} \\ \dot{p_z} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\varphi} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \\ \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = f\left(p_x, p_y, p_z, \psi, \theta, \varphi, v_x, v_y, v_z \omega_x, \omega_y, \omega_z, \tau_x, \tau_y, \tau_z, f_z\right)$$

Where 'f' is defined as,

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

The equations of motion as derived from rotation, angular velocity, and angular rates with respect to applied forces and torques.

Linearization allows the simplification of analysis and control design for the drone's nonlinear dynamics. The goal is to approximate the nonlinear system around and equilibrium point with a linear model. The equilibrium point is a state where all derivatives of the state variables are zero. The equilibrium force along the z-axis is given by,

$$f_{z_e} = \begin{cases} controller, & \dfrac{v_{x_e}\omega_{y_e} - v_{y_e}\omega_{x_e} - 981\cos(\varphi_e)\cos(\theta_e)}{200} \\ otherwise, & \dfrac{1}{2}g \end{cases}$$

The sensor model relates the drone's states to its sensor outputs, the position of the markers on the drone's body, given by,

$$o = g(p_x, p_y, p_z, \psi, \theta)$$

Where ‘g’ is defined as,

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

The matrices can now be defined to their respective Jacobians,

$$A = \frac{\delta f}{\delta x}$$

Given by,

$$\begin{bmatrix} 0 & 0 & 0 & -v_x \sin(\psi)\cos(\theta) + v_y(-\sin(\phi)\sin(\psi)\sin(\theta) - \cos(\phi)\cos(\psi)) + v_z(\sin(\phi)\cos(\psi) - \sin(\psi)\sin(\theta)\cos(\phi)) \\ 0 & 0 & 0 & v_x\cos(\psi)\cos(\theta) + v_y(\sin(\phi)\sin(\theta)\cos(\psi) - \sin(\psi)\cos(\phi)) - v_z(-\sin(\phi)\sin(\psi) - \sin(\theta)\cos(\phi)\cos(\psi)) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -v_x\sin(\theta)\cos(\psi) + v_y\sin(\phi)\cos(\psi)\cos(\theta) + v_z\cos(\phi)\cos(\psi)\cos(\theta) & v_y(\sin(\phi)\sin(\psi) + \sin(\theta)\cos(\phi)\cos(\psi)) + v_z(-\sin(\phi)\sin(\theta) \\ -v_x\sin(\psi)\sin(\theta) + v_y\sin(\phi)\sin(\psi)\cos(\theta) + v_z\sin(\psi)\cos(\phi)\cos(\theta) & v_y(-\sin(\phi)\cos(\psi) + \sin(\psi)\sin(\theta)\cos(\phi)) - v_z(\sin(\phi)\sin(\psi) \\ -v_x\cos(\theta) - v_y\sin(\phi)\sin(\theta) - v_z\sin(\theta)\cos(\phi) & v_y\cos(\phi)\cos(\theta) - v_z\sin(\phi)\cos(\theta) \\ \frac{(w_y\sin(\phi)+w_z\cos(\phi))\sin(\theta)}{\cos^2(\theta)} & \frac{w_y\cos(\phi)-w_z\sin(\phi)}{\cos(\theta)} \\ 0 & -w_y\sin(\phi) - w_z\cos(\phi) \\ w_y(\tan^2(\theta)+1)\sin(\phi) + w_z(\tan^2(\theta)+1)\cos(\phi) & w_y\cos(\phi)\tan(\theta) - w_z\sin(\phi)\tan(\theta) \\ \frac{981\cos(\theta)}{100} & 0 \\ \frac{981\sin(\phi)\sin(\theta)}{100} & -\frac{981\cos(\phi)\cos(\theta)}{100} \\ \frac{981\sin(\theta)\cos(\phi)}{100} & \frac{981\sin(\phi)\cos(\theta)}{100} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \cos(\psi)+\sin(\psi)\cos(\phi)) & \cos(\psi)\cos(\theta) & \sin(\phi)\sin(\theta)\cos(\psi) - \sin(\psi)\cos(\phi) & \sin(\phi)\sin(\psi) + \sin(\theta)\cos(\phi)\cos(\psi) & 0 \\ \sin(\theta)+\cos(\phi)\cos(\psi)) & \sin(\psi)\cos(\theta) & \sin(\phi)\sin(\psi)\sin(\theta) + \cos(\phi)\cos(\psi) & -\sin(\phi)\cos(\psi) + \sin(\psi)\sin(\theta)\cos(\phi) & 0 \\ & -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) & 0 \\ & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 1 \\ & 0 & w_z & -w_y & 0 \\ & -w_z & 0 & w_x & v_z \\ & w_y & -w_x & 0 & -v_y \\ & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & \frac{17w_z}{23} \\ & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \\ \cos(\phi) & -\sin(\phi) \\ \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ -v_z & v_y \\ 0 & -v_x \\ v_x & 0 \\ -\frac{17w_z}{23} & -\frac{17w_y}{23} \\ 0 & \frac{17w_x}{23} \\ 0 & 0 \end{bmatrix}$$

Evaluated at equilibrium point.

Similarly,

$$B = \frac{\delta f}{\delta u}$$

Evaluated at equilibrium point.

$$C = \frac{\delta g}{\delta x}$$

Evaluated at equilibrium point.

$$D = \frac{\delta g}{\delta u}$$

Evaluated at equilibrium point.

This yields the linearized model for the EOMs for the drone.

**B. Controller and Observer Design**

The control feedback gain matrix, 'K', determines control action based on system state. It computes the control input that aims to bring the state of the drone to the desired state or trajectory. It is a key component in the control law, influencing the performance and stability of the control system. In the controller, the control input, 'u', is calculated using the formula,

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

Where $\hat{x}$ is the estimated state and $x_{des}$ is the desired state. This formula represents a negative feedback loop, essential for stabilizing control systems. Using pole placement, K is calculated to place the poles (eigenvalues) of the closed-loop system (represented by $A - BK$) at desired locations in the complex plane, to achieve specific transient and stability characteristics.

The matrix, L, is the observer gain matrix, integral to the observer's functionality used in the state estimator (observer) to update the estimate of the system's state. It plays a crucial role in determining the rate at which the estimated state converges to the true state. In the observer, the state estimate $\hat{x}$ is updated using the formula,

$$\hat{x} = \hat{x} + dt\big(A\hat{x} + Bu - L(C\hat{x} - y)\big)$$

Where 'y' is the measured output. The code calculates the observer gain L by applying the LQR method to the transposed system matrices (A.T and C.T). The matrices Qo and Ro represent the cost matrices for the observer design, balancing the trade-off between estimation accuracy and sensitivity to measurement noise.

**C. Controllability and Observability**

A system is said to be controllable if it's possible to move the state of the system from any initial state to any desired state in a finite time period using the control inputs. The controllability of the system can be checked by forming the controllability matrix and then assessing whether this matrix has full rank (i.e., its rank is equal to the number of states of the system). The controllability matrix for a linear system is formed using the system matrix (A) and the input matrix (B) as follows,

$$W = [B, AB, A^2B, \dots, A^{n-1}B]$$

Where n is the number of states.

A function in the code tests for this and outputs the result.

A system is said to be observable if the current state can be determined in finite time using only the outputs. The observability of the system is checked by forming the observability matrix and then assessing whether this matrix has full rank. The observability matrix for a linear system is formed using the system matrix (A) and the output matrix (C) as follows,

$$O = \begin{bmatrix} C \\ CA \\ CA^2 \\ \dots \\ CA^{n-1} \end{bmatrix}$$

A function in the code tests for this and outputs the result.

Stability checks are also performed on both matrices. If the gain matrix leads to all eigenvalues having negative real parts, it means the closed-loop system will settle over time, and any disturbance or initial condition will decay, leading to a stable system. Similarly, a stable observer implies that the estimated states will converge to the actual states, ensuring that the controller has accurate information to work with.

## IV. Experimental Methods

The requirement set is that the drone should complete the race within 30 seconds and pass through every ring in the course without colliding with them. In addition, it is expected that the drone takes a reasonable route, where reasonable is defined by a trajectory that does not deviate from a human based optimal trajectory by a factor deemed unnecessary by the author. The relevant code file, generate results, implements the above theory in a mathematical model using python. PyBullet will be used to simulate the drone. The data generated by this simulation will be imported into a Jupyter Notebook for analysis with Python. All relevant variables are predefined but can be altered. In the following section, these values are initially decided. The initial seed is needed to generate the same map. The data further includes the drone's position in 3D space at each time step that enables its trajectory to be plotted. The requirement can be verified based on terminal output tests and the plots produced by the data.

## V. Results and Discussion

Assuming initial conditions,

| Variable | Value | Unit |
|---|---|---|
| $m$ | 0.5 | $kg$ |
| $J_x$ | 0.0023 | $kgm^2$ |
| $J_y$ | 0. 0023 | $kgm^2$ |
| $J_z$ | 0.0040 | $kgm^2$ |
| $l$ | 0.175 | $m$ |
| $g$ | 9.81 | $ms^{-1}$ |
| ***Initial seed*** | 1167920979 | $NA$ |

**Table 1: Initial Assumed Conditions**

With equilibrium points,

| Variable | Value | Unit |
|---|---|---|
| $p_x, p_y, p_z$ | 0 | $m$ |
| $\psi, \theta, \phi$ | 0 | $rad$ |
| $v_x, v_y, v_z$ | 0 | $ms^{-1}$ |
| $\omega_x, \omega_y, \omega_z$ | 0 | $rads^{-1}$ |
| $\tau_x, \tau_y \tau_z$ | 0 | $Nm$ |
| $f_z$ | 4.905 = 0.5g | $N$ |

**Table 2: Eq Points**

This yields the following matrices,

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 9.81 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -9.81 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.0 \\ 434.782608695652 & 0 & 0 & 0 \\ 0 & 434.782608695652 & 0 & 0 \\ 0 & 0 & 250.0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0.175 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 & -0.175 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & -0.175 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 & 0.175 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Therefore K (unrounded matrix available in code) is,

$$\begin{bmatrix} -0.002 & -0.0082 & -1.654 \cdot 10^{-6} & -9.16 \cdot 10^{-6} & -0.01 & 0.098 & -0.0028 & -0.015 & -6.3 \cdot 10^{-7} & 0.025 & -0.0011 & -3.04 \cdot 10^{-6} \\ 0.012 & 0.0023 & -6.29 \cdot 10^{-6} & -1.73 \cdot 10^{-6} & 0.119 & -0.01 & 0.0213 & 0.002 & -1.82 \cdot 10^{-6} & -0.001 & 0.027 & -6.59 \cdot 10^{-7} \\ -6.11 \cdot 10^{-5} & 0.0004 & -0.006 & 0.029 & -0.0005 & -0.003 & -0.0001 & 0.0007 & -0.0017 & -0.0003 & -7.92 \cdot 10^{-5} & 0.023 \\ -0.046 & 0.0068 & 5.768 & -0.776 & -0.20 & -0.06 & -0.057 & 0.0143 & 3.476 & -0.009 & -0.022 & -0.213 \end{bmatrix}$$

And L,

$$\begin{bmatrix} 2.9271137918024 & 2.93223980190513 \cdot 10^{-16} & -0.219768163684976 & 2.9271137918024 & 1.82427389506807 \cdot 10^{-16} & 0.219768163684976 \\ 2.3782568484866 \cdot 10^{-16} & 2.97537601896218 & 4.13208457549389 \cdot 10^{-16} & 2.3782568484866 \cdot 10^{-16} & 2.97537601896218 & 3.73443240593517 \cdot 10^{-16} \\ -1.32897663063478 \cdot 10^{-16} & 8.62779647032773 \cdot 10^{-16} & 1.09868411346781 & -1.32897663063478 \cdot 10^{-16} & -7.6127948889867 \cdot 10^{-17} & 1.09868411346781 \\ 3.16561687667733 \cdot 10^{-16} & 2.13087075586691 & 2.59086474592405 \cdot 10^{-15} & 3.16561687667733 \cdot 10^{-16} & -2.13087075586692 & 2.77432151649104 \cdot 10^{-15} \\ 1.25581807819986 & -2.18865203047139 \cdot 10^{-17} & -0.304297380355707 & 1.25581807819986 & -2.05343290871697 \cdot 10^{-16} & 0.304297380355706 \\ -6.5334214135461 \cdot 10^{-16} & -1.30543173901126 & -5.10444599151935 \cdot 10^{-16} & -6.5334214135461 \cdot 10^{-16} & -1.30543173901126 & -6.24486275139007 \cdot 10^{-16} \\ 8.11629319592927 & -3.2008775027479 \cdot 10^{-16} & -1.19153356884774 & 8.11629319592927 & -8.92285376377905 \cdot 10^{-16} & 1.19153356884773 \\ 2.76681098987244 \cdot 10^{-15} & 8.35286245421523 & 1.90126003587643 \cdot 10^{-15} & 2.76681098987244 \cdot 10^{-15} & 8.35286245421523 & 2.21728292423291 \cdot 10^{-15} \\ -1.8921278731614 \cdot 10^{-16} & 1.4260687894301 \cdot 10^{-16} & 0.707106781186546 & -1.8921278731614 \cdot 10^{-16} & -6.13472947699331 \cdot 10^{-16} & 0.707106781186546 \\ -5.11209546812985 \cdot 10^{-16} & -0.70710678118655 & -8.61575568360618 \cdot 10^{-16} & -5.11209546812985 \cdot 10^{-16} & -0.707106781186551 & -9.60839295318142 \cdot 10^{-16} \\ 0.676831335818545 & -1.41814410763377 \cdot 10^{-16} & -0.204693289714364 & 0.676831335818545 & -3.4469712775926 \cdot 10^{-16} & 0.204693289714363 \\ -4.98207922158605 \cdot 10^{-17} & 0.707106781186543 & 1.0207455989306 \cdot 10^{-15} & -4.98207922158605 \cdot 10^{-17} & -0.707106781186549 & 1.04293537543721 \cdot 10^{-15} \end{bmatrix}$$

Controllability, observability, and stability is tested as defined above,

$$W = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.0 \\ 434.782608695652 & 0 & 0 & 0 \\ 0 & 434.782608695652 & 0 & 0 \\ 0 & 0 & 250.0 & 0 \end{bmatrix}$$

$$O = \begin{bmatrix} 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0.175 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 & -0.175 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & -0.175 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 & 0.175 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

As per the rank tests, the results show,

```
drone = DroneDynamics()
drone.test_stable_K()
drone.test_stable_L()
# Check for controllability
if drone.check_controllability():
    print("The system is controllable.")
else:
    print("The system is not controllable.")

# Check for observability
if drone.check_observability():
    print("The system is observable.")
else:
    print("The system is not observable.")
```

✓ 1.5s

```
K matrix is stable
L matrix is stable
```

**Fig. 1 Stability, controllability and observability terminal output**

The simulation is then run with those conditions. The drone is able to complete the course without colliding with any of the rings, see the footage of the race attached for full route.

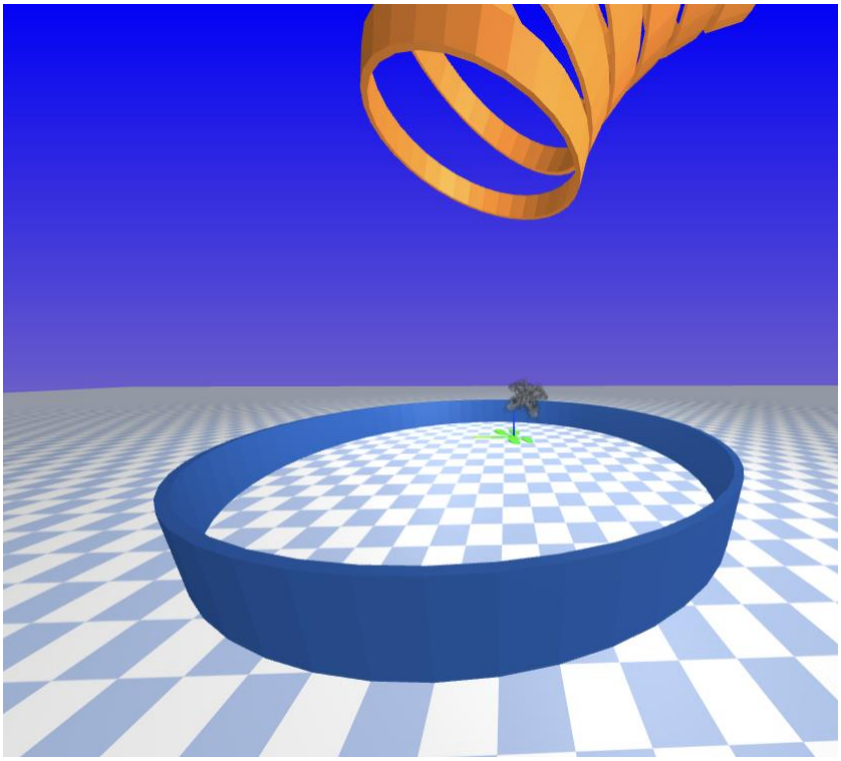

**Fig. 2 Drone at finish**

Confirmed by,

```
if did_it_fail:
    print('The drone failed before finishing.')
elif did_it_finish:
    print(f'The drone finished at time {what_time_did_it_finish}')
else:
    print('The drone did not finish (yet).')
```

```
The drone finished at time 29.12
```

**Fig. 3 Terminal output of drone completion**

The drone is confirmed to pass through all the rings as per the position data recorded,

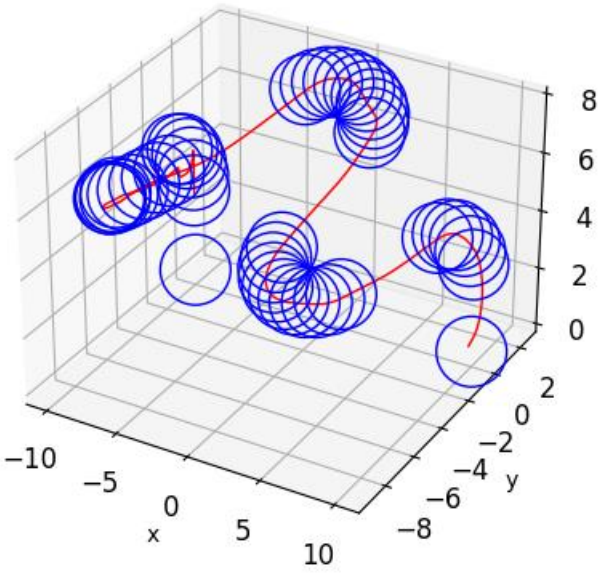


**Fig. 4 Drone trajectory**

To examine drone performance,

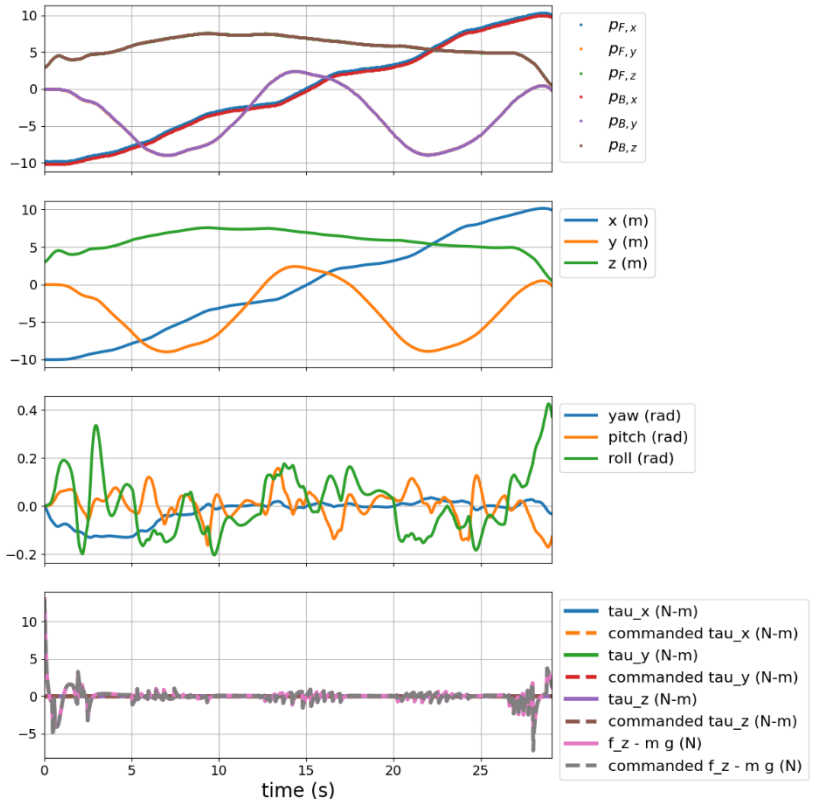


**Fig. 5 Drone parameters**

Based on this data, it can be concluded that at the provided initial conditions, the drone is able to, successfully and in a timely manner, complete the race. The requirement is satisfied as the drone went through every ring, took a reasonable trajectory, and finished in under 30 seconds. It is also observed that the position changes were relatively smooth indicating the ability to carry sensitive load and take smooth images and videos.

### A. Limitations

Initial parameters are altered to establish the limits of the designed controller.

```
if did_it_fail:
    print('The drone failed before finishing.')
elif did_it_finish:
    print(f'The drone finished at time {what_time_did_it_finish}')
else:
    print('The drone did not finish (yet).')
```

✓ 0.0s

```
The drone finished at time 31.32
```

**Fig. 6 Terminal output with different seed**

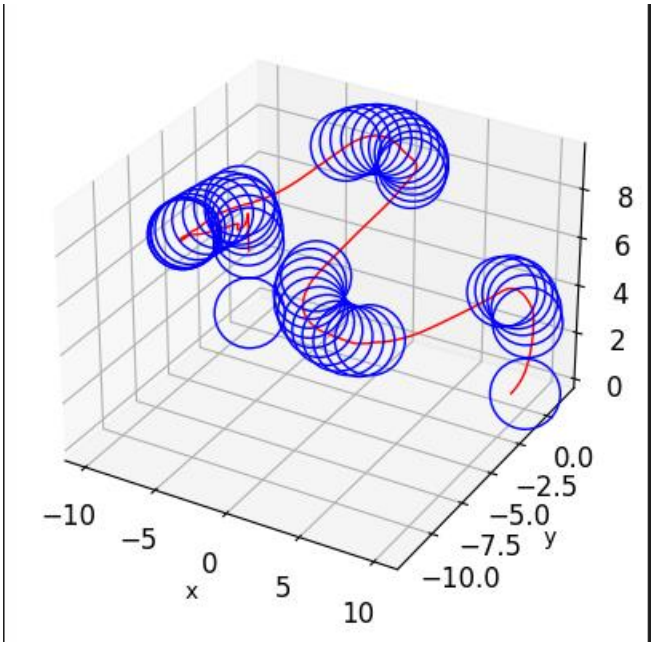


**Fig. 7 Trajectory with different seed**

The seed is changed to produce a different course. The drone is able to complete the course, however the requirement is not met as it took more than 30 seconds. However, the controller is functional in variable scenarios.

```
    if did_it_fail:
        print('The drone failed before finishing.')
    elif did_it_finish:
        print(f'The drone finished at time {what_time_did_it_finish}')
    else:
        print('The drone did not finish (yet).')
✓ 0.0s
The drone failed before finishing.
```

**Fig. 8 Terminal output at drone mass 5kg**

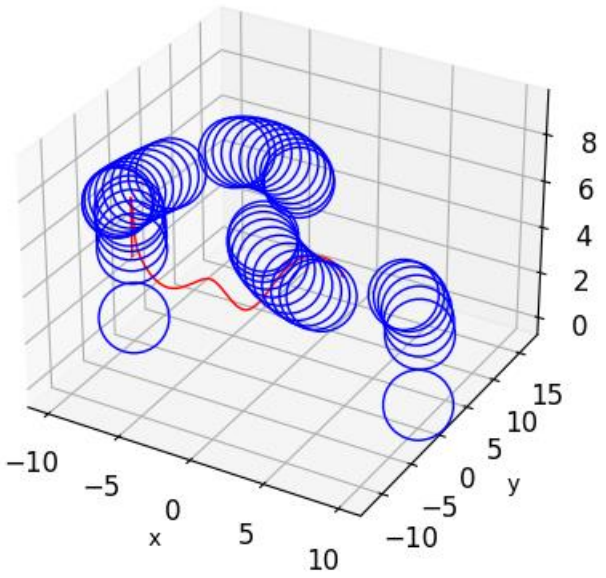


**Fig. 9 Trajectory at drone mass 5kg**

The mass was increased to 5kg to simulate the presence of a payload, the drone was unable to complete the race and quickly impacted the ground. This shows limitations in the carrying capacity of the controller for medical supplies or camera equipment for surveillance.

## VI. Conclusion

The designed control system successfully met the project objectives, demonstrating the drone's ability to complete a complex course within the stipulated time frame and without collision. The linearization of motion equations, coupled with the implementation of a stable controller and observer, proved effective in controlling the drone's trajectory. The system's robustness was further underscored by its performance under varied conditions. Future work could explore enhancements in payload capacity and adaptability to different drone models, broadening the system's applicability in diverse scenarios.

## VII. Acknowledgements

The author wishes to express their sincere gratitude to Professor Melkior Ornik for their invaluable guidance and insights throughout the course of this research and by imparting the required knowledge to complete it. Special thanks are also extended to Gokul Puthumanaillam for their significant contributions and tireless efforts to clarify and walk the author through the experimental and analytical aspects of this study.

## VIII. References

"Design Project 3," AE353 - Introduction to Control System Design, Fall 2023, accessed [1/12/2023], < https://github.com/uiuc-ae353/ae353-fa23/wiki/Design-Project-3>