Optimal Trajectory Control of a Quadcopter

Connor Reece (1205970957), Vishal Gupta (1226206568), Kaushik Iyer (1223696175)

I. Introduction

Trajectory control for unmanned aerial vehicles (UAVs) in general has been a topic of modern research. Quadcopters are a specific type of UAV, and have been used in numerous applications ranging from military and civil surveillance, logistics, and forest fire monitoring [1]. However, since they are airborne and are also used in areas where humans cannot reach, they are subject to multiple sources of disturbances and obstacles. This necessitates an effective way to control Quadcopter UAVs, in particular, their trajectories and movements. The feedback control methods used in the literature [2], [1], [3] are usually based on a state space modeling of the dynamic equations of motion that govern the motion of the Quadcopter.

This project aims to study two main optimal feedback control methods, namely, the proportional, integral, derivative (PID) and linear quadratic control (LQR) control, for trajectory control of the Quadcopter. An additional objective is also to study the cascading of these methods. Since we perform a comparative study of different methods, the main idea for the PID is taken from [4], along with some insights from [5] for implementation of an optimal tuning method. The main ideas for the cascaded LQR-PI and LQR have been taken from [1].

II. State Space Representation

The state space representation of a Quadcopter is derived from the non-linear equations of motion. However, the model is linearised about equilibrium points before forming a state space realisation. This work considers a 6 degree of freedom (DOF) model. The states of the system are the Euler angles (roll (ϕ), pitch (θ), yaw (ψ)), the linear position in (x, y, z), linear velocities $(\dot{x}, \dot{y}, \dot{z})$ and the angular rates $(\dot{\phi}, \dot{\theta}, \dot{\psi})$ [3], [6], [7]. The reader is referred to [3] for details on the dynamics and derivation of the equations of motion. For brevity, the dynamic equations of motions are skipped and the state space representation is presented here. For consistency and simplicity, we chose to use the realisation in [6].

A. 6-DOF State Space

Define $\mathbf{\Omega} = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^T$ as the Euler angle vector, $\dot{\mathbf{\Omega}} = \begin{bmatrix}\dot{\phi} & \dot{\theta} & \dot{\psi}\end{bmatrix}^T$ as the angular rate vector, $\mathbf{p} = \begin{bmatrix}x & y & z\end{bmatrix}^T$ as the linear position vector and $\mathbf{v} = \begin{bmatrix}\dot{x} & \dot{y} & \dot{z}\end{bmatrix}^T$ as the linear velocity vector. From this, the state vector for the 6-DOF system is given by

$$\mathbf{x} = \begin{bmatrix}\mathbf{p}^T & \mathbf{v}^T & \mathbf{\Omega}^T & \dot{\mathbf{\Omega}}^T\end{bmatrix}^T \tag{1}$$

The system model is given by:

$$\begin{aligned}\frac{d}{dt}\mathbf{x} &= \begin{bmatrix}\mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3}\\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{W}_{3\times3} & \mathbf{0}_{3\times3}\\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3}\\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3}\end{bmatrix}\mathbf{x} + \begin{bmatrix}\mathbf{0}_{4\times4}\\ \mathbf{F}_{1\times4}\\ \mathbf{0}_{4\times4}\\ \mathbf{G}_{3\times4}\end{bmatrix}\mathbf{u}\\ \frac{d}{dt}\mathbf{x} &= \mathbf{A}_{12\times12}\mathbf{x} + \mathbf{B}_{12\times4}\mathbf{u}\end{aligned} \tag{2}$$

where $\mathbf{W}$, $\mathbf{F}$ and $\mathbf{G}$ are defined as:

$$\mathbf{W} = \begin{bmatrix}0 & 0 & -g\\ 0 & g & 0\\ 0 & 0 & 0\end{bmatrix}\quad \mathbf{F} = \begin{bmatrix}\frac{-1}{m}\\ 0\\ 0\\ 0\end{bmatrix}^T\quad \mathbf{G} = \begin{bmatrix}0 & \frac{1}{I_x} & 0 & 0\\ 0 & 0 & \frac{1}{I_y} & 0\\ 0 & 0 & 0 & \frac{1}{I_z}\end{bmatrix} \tag{3}$$

The outputs of interest in this case are $\mathbf{y} = \begin{bmatrix}\mathbf{p} & \mathbf{\Omega}\end{bmatrix}^T$. Hence the output equation is given by:

$$\begin{aligned}\begin{bmatrix}\mathbf{p}\\ \mathbf{\Omega}\end{bmatrix} &= \begin{bmatrix}\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3}\\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3}\end{bmatrix}\mathbf{x} + \mathbf{0}_{6\times4}\mathbf{u}\\ \mathbf{y} &= \mathbf{C}_{6\times12}\mathbf{x} + \mathbf{D}_{6\times4}\mathbf{u}\end{aligned} \tag{4}$$

III. Trajectory Control

To solve the problem of trajectory control, multiple methods have been studied in the literature. This work considers the integral time absolute error PID (PID-ITAE), LQR and LQR-PI methods. This work considers the case of a 6-DOF Quadcopter.

A. PID-ITAE Control

The schematic for the PID-ITAE controller is shown in fig. 1. The flowchart shown at the top is the general approach to the controller tuning by ITAE performance index, and the diagram below is the Simulink model schematic. The methodology for the PID-ITAE controller is enumerated as under: First, the input/control-to-output transfer function $\mathbf{H}(s)$ is calculated as follows:

$$\begin{aligned}\mathbf{H}(s) &= \frac{\mathbf{Y}_0(s)}{\mathbf{U}(s)}\\ &= \mathbf{C}\left(s\mathbf{I}_{12\times12} - \mathbf{A}\right)^{-1}\mathbf{B} + \mathbf{D}\end{aligned} \tag{5}$$

where $\mathbf{Y}_0 = \begin{bmatrix}X(s) & Y(s) & Z(s) & \Phi(s) & \Theta(s) & \Psi(s)\end{bmatrix}^T$ is the Laplace transform of the outputs, and $\mathbf{U}(s) = \begin{bmatrix}U_1(s) & U_2(s) & U_3(s) & U_4(s)\end{bmatrix}^T$ are the Laplace transforms

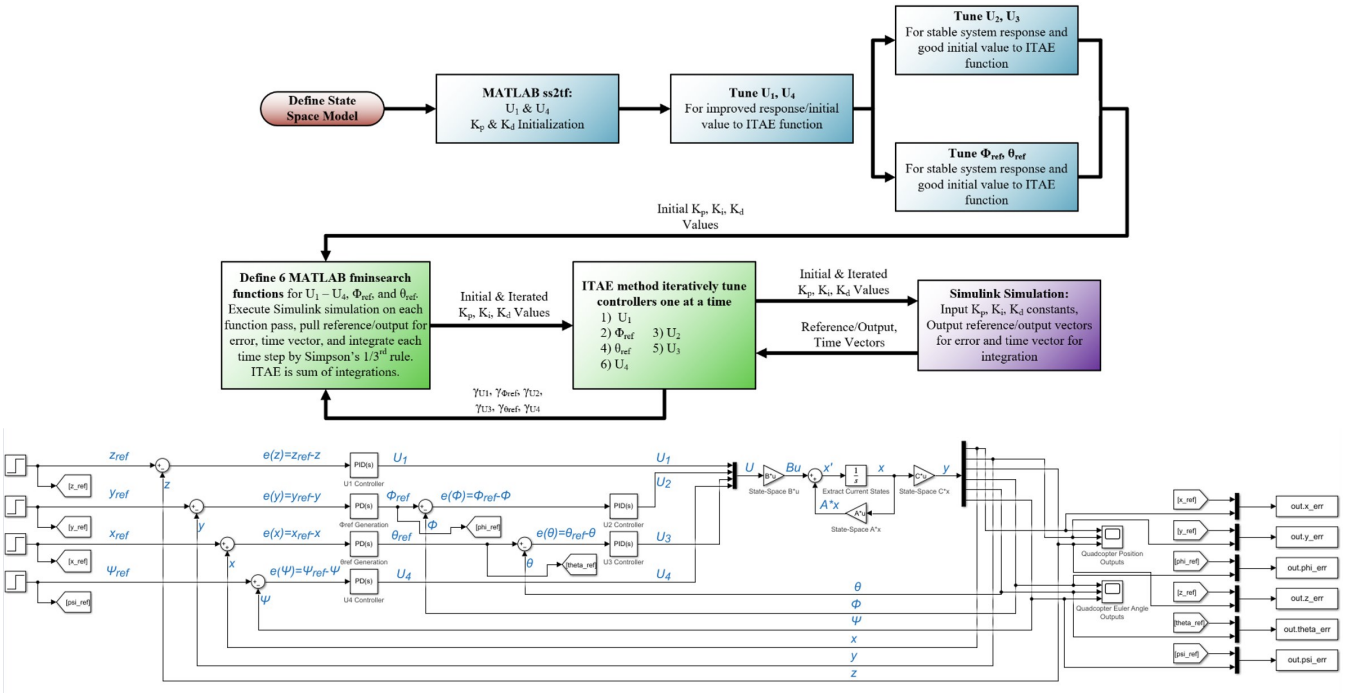

Fig. 1: Schematic for PID-ITAE.

of the control inputs. Since this is a multi-input-multi-output (MIMO) system, $\mathbf{H}(s)$ is a matrix, obtained as:

$$\mathbf{H}(s) = \begin{bmatrix} \mathbf{L}_{3\times2} & \mathbf{M}_{3\times2} \\ \mathbf{P}_{3\times2} & \mathbf{T}_{3\times2} \end{bmatrix}$$
$$\mathbf{L} = \begin{bmatrix} 0 & 0 \\ 0 & \frac{-g}{s^4 I_x} \\ \frac{-1}{s^2 m} & 0 \end{bmatrix} \quad \mathbf{M} = \begin{bmatrix} \frac{-g}{s^4 I_y} & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{6}$$
$$\mathbf{P} = \begin{bmatrix} 0 & \frac{1}{s^2 I_x} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \mathbf{T} = \begin{bmatrix} 0 & 0 \\ \frac{1}{s^2 I_x} & 0 \\ 0 & \frac{1}{s^2 I_z} \end{bmatrix}$$

Expanding (5) and substituting $\mathbf{H}(s)$ from (6) in (5), the following relations are established:

$$X(s) = \frac{-g}{s^4 I_y} U_3(s) \quad Y(s) = \frac{g}{s^4 I_x} U_2(s)$$
$$Z(s) = \frac{-1}{s^2 m} U_1(s) \quad \Phi(s) = \frac{1}{s^2 I_x} U_2(s) \tag{7}$$
$$\Theta(s) = \frac{1}{s^2 I_y} U_3 \quad \Psi(s) = \frac{1}{s^2 I_z} U_4(s)$$

From Equation (7), it is apparent that alteration of the MIMO system transfer function is necessary, as the nominal model is inherently unstable due to second and fourth order poles at the origin. Hence, based on the relations in (7), for coupled transfer functions the control loop is split into two loops for: The outer loop, the position control loop which sets a reference signal for the inner loop, which is the torque control loop. Non-coupled transfer functions implement no integral term, to ensure no added pole for stability. The control laws for the PID are as follows:

Inner Loop (Torque Control):

$$\phi_{\text{ref}} = K_{p,\phi}\left(e(y)\right) + K_{d,\phi}\left(\frac{d}{dt}e(y)\right)$$
$$\theta_{\text{ref}} = K_{p,\theta}\left(e(x)\right) + K_{d,\theta}\left(\frac{d}{dt}e(x)\right)$$

Outer Loop (Attitude/Position, Yaw Torque Control):

$$U_1 = K_{p,U_1}\left(e(z)\right) + K_{d,U_1}\left(\frac{d}{dt}(e(z))\right)$$
$$U_2 = K_{p,U_2}\left(e(\phi)\right) + K_{i,U_2}\left(\int e(\phi)dt\right) + K_{d,U_2}\left(\frac{d}{dt}(e(\phi))\right)$$
$$U_3 = K_{p,U_2}\left(e(\theta)\right) + K_{i,U_2}\left(\int e(\theta)dt\right) + K_{d,U_2}\left(\frac{d}{dt}(e(\theta))\right)$$
$$U_4 = K_{p,U_4}\left(e(\psi)\right) + K_{d,U_4}\left(\frac{d}{dt}(e(\psi))\right) \tag{8}$$

where the gains K_p, K_i, K_d are tuned using the ITAE method. This process is defined by defining the ITAE index γ, which is obtained using a minimisation process on the controllers as follows:

$$\gamma = \int_{t_0}^{t_f} |e(t)|dt \tag{9}$$

B. LQR Control

The LQR control method is based on finding an optimal control law that minimises the infinite horizon cost function $J(\mathbf{x}(\mathbf{t}), \mathbf{u}(t))$ defined as:

$$J(\mathbf{x}(\mathbf{t}), \mathbf{u}(t)) = \int_{t_0}^{\infty} \left(\mathbf{x}(t)^T \mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)^T \mathbf{R}\mathbf{u}(t)\right) dt \tag{10}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are matrices to be tuned. In this work, the method proposed in [1] will be presented to tune $\mathbf{Q}$. The optimal control given by the full state feedback $\mathbf{u}^*(t) = -\mathbf{K}_{\text{lqr}}\mathbf{x}^*(t)$, where the optimal LQR gain is given by:

$$\mathbf{K}_{\text{lqr}} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} \tag{11}$$

and $\mathbf{P}$ is the solution of the algebraic Riccati equation

$$\mathbf{PA} + \mathbf{A}^T\mathbf{S} - \mathbf{PBR}^{-1}\mathbf{B}^T\mathbf{P} + \mathbf{Q} = \mathbf{0} \tag{12}$$

As per [1], $\mathbf{Q}$ is initialised as:

$$\mathbf{Q} = \frac{1}{\alpha} diag\left(\begin{bmatrix}||\mathbf{A}(1,:)||_\infty & \dots & ||\mathbf{A}(n,:)||_\infty\end{bmatrix}\right) \tag{13}$$

where $||\cdot||_\infty$ denotes the L_∞ norm or the maximum value. This is taken along every row of $\mathbf{A}$. n denotes the dimension of $\mathbf{A}$, α is a parameter, chosen as α = 0.01, and $diag(\cdot)$ denotes a diagonal matrix. The schematic for the LQR controller is shown in fig. 2. Similar to fig. 1, fig. 2 shows the general flow of the controller design, and the design as implemented in Simulink. The notations in fig. 2 are as followed everywhere in the paper. PT_{ref}, RT_{ref}, YT_{ref} refer to the roll, pitch and yaw torques respectively. UF_{ref} refers to the upward force.

C. Augmented LQR-PI (Observer) Control

The LQR-PI control, as defined in [1] is essentially an observer based control. It relies on creating an augmented state space system, which is defined using output feedback. The additional states are defined as error states with respect to the ideal/reference values. However, since in our case the outputs are the states of the system, it is very similar to an observer based control, where the reference can be thought of as the observer output. The augmented LQR-PI Observer model is defined by the state space model given by:

$$\begin{bmatrix}\dot{\mathbf{x}} \\ \dot{\mathbf{x}_e}\end{bmatrix} = \begin{bmatrix}\mathbf{A} - \mathbf{BK}_{\text{lqr}} & \mathbf{BK}_{\text{lqr}} \\ \mathbf{0} & \mathbf{A} - \mathbf{LC}\end{bmatrix}\begin{bmatrix}\mathbf{x} \\ \mathbf{x}_e\end{bmatrix} \tag{14}$$

where the gain $\mathbf{L}$ can be obtained by pole placement. Generally, the poles are placed farther away from those of the LQR-based feedback. This helps achieve a faster response. The augmented LQR-PI controller schematic is shown in fig. 3. As seen in fig. 3, there are two feedback blocks, one for the original system, and the other for the observer model. The feedback is defined with respect to the observer.

IV. Implementation, Results and Discussion

All the simulation is to be done in MATLAB and Simulink. The analysis has been divided into two broad categories:

1) **CASE 1: Stability Analysis** - These results have been replicated from [1]. The Quadcopter is made to hover at a stable reference point with fixed yaw angle. The position values are set to $[x, y, z] = [0, 0, 8]$ m. The angles are set to $[\phi, \theta, \psi] = [0, 0, 5]$ rads. This has been divided into 5 scenarios:
   a) Scenario 1: Nominal State-Space System
   b) Scenario 2: 30% Mass Uncertainty, With Angle Disturbance
   c) Scenario 3: 50% Mass Uncertainty, With Angle Disturbance
   d) Scenario 4: 50% Loss of Quadcopter Actuator Efficiency
   e) Scenario 5: 90% Loss of Quadcopter Actuator Efficiency

   where the mass uncertainty is analogous to the Quadcopter picking a payload, and is modeled as follows:

$$m_{\text{uc}} = m(1 + k); \quad k \in \{0.3, 0.5\} \tag{15}$$

   m_{uc} is the mass after uncertainty addition and m is the nominal Quadcopter mass.
   The angle disturbance is modeled as an exponentially decaying sinusoid as follows:

$$d(t) = 0.2 + 0.4e^{-t}\sin(62.8316t) \tag{16}$$

   For the last two scenarios, we also consider the case of a temporary malfunction in the Quadcopter actuators (e.g. rotors not spinning fast enough), which is manifested as a loss of actuator efficiency. This has been mathematically formulated as [1]:

$$\begin{aligned}\mathbf{u}_{\text{new}} &= \rho\mathbf{u} \\ \rho &= \begin{bmatrix}1 & 0 & 0 & 0 \\ 0 & \eta_{\text{act}} & 0 & 0 \\ 0 & 0 & \eta_{\text{act}} & 0 \\ 0 & 0 & 0 & \eta_{\text{act}}\end{bmatrix} \\ \eta_{\text{act}} &= 1 - n; \quad n \in \{0.3, 0.9\}\end{aligned} \tag{17}$$

2) **CASE 2: Circular Trajectory Following with Disturbance**- For our defined trajectory, we first raise the Quadcopter height to 8 meters, set the yaw angle to 0.25 radians, then follow a circular trajectory in the x and y axes with amplitude of 5 meters for 4 rotations with period of 20 seconds prior to returning to resting 0,0,0 positions with 0,0,0 Euler angles. The same Euler angle disturbance model used in stability simulations is injected at t = 30 seconds, t = 25 seconds, and t = 7 seconds for roll, pitch, and yaw angles respectively. The reference trajectory is mathematically governed by the following equations:

$$\begin{aligned}z_{\text{ref}} &= 8 + 0.4\left[u(t - 78.46)(t - 78.46)\right] \\ &\quad - 0.4\left[u(t - 58.46)(t - 58.46)\right] \\ RT_{\text{ref}} &= 5\sin(0.3142t + \pi/2)(u(t - 12.5) - u(t - 99.285)) \\ PT_{\text{ref}} &= 5\sin(0.3142t)(u(t - 12.5) - u(t - 94.2)) \\ YT_{\text{ref}} &= 0.25\left[u(t - 1) - u(t - 98.46)\right]\end{aligned} \tag{18}$$

   where $u(t)$ is the step function and $u(t)(t)$ is the ramp function.

The implementation details and initialisation of each of the methods described previously, are enumerated below.

1) ***ITAE-PID:*** The ITAE index γ is obtained by using the *fminsearch* function in MATLAB's Optimization Toolbox. The integration is done using Simpson's $1/3^{\text{rd}}$ rule, and the time and error vectors were obtained from Simulink.

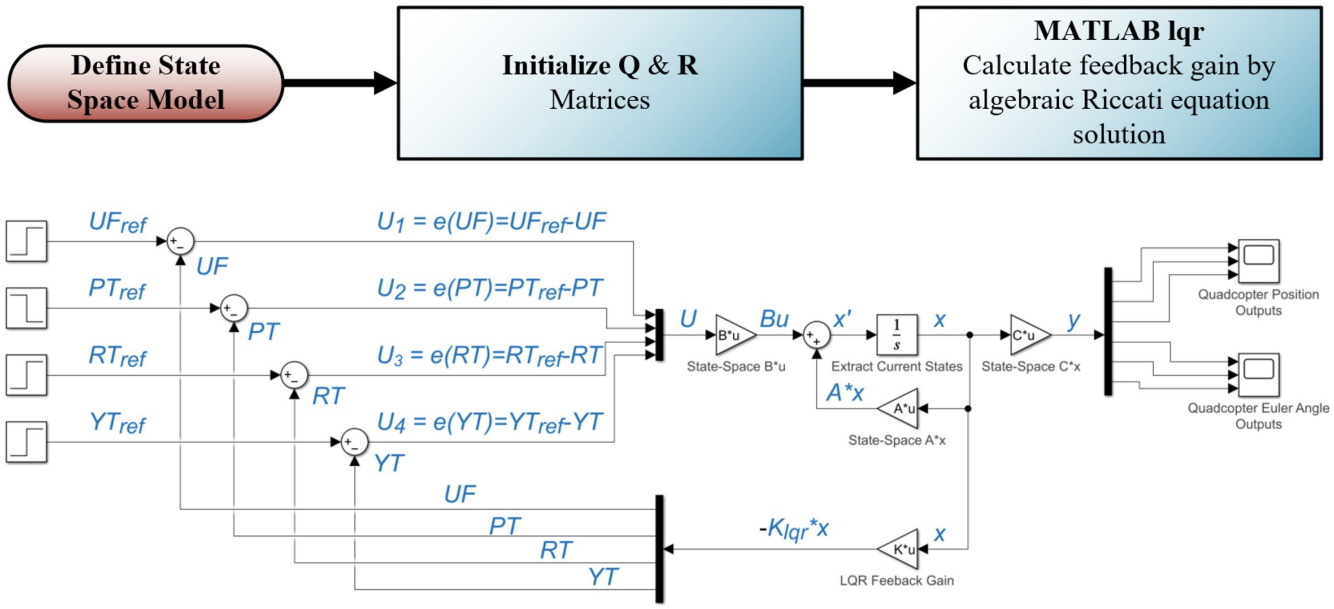

Fig. 2: Schematic for LQR.

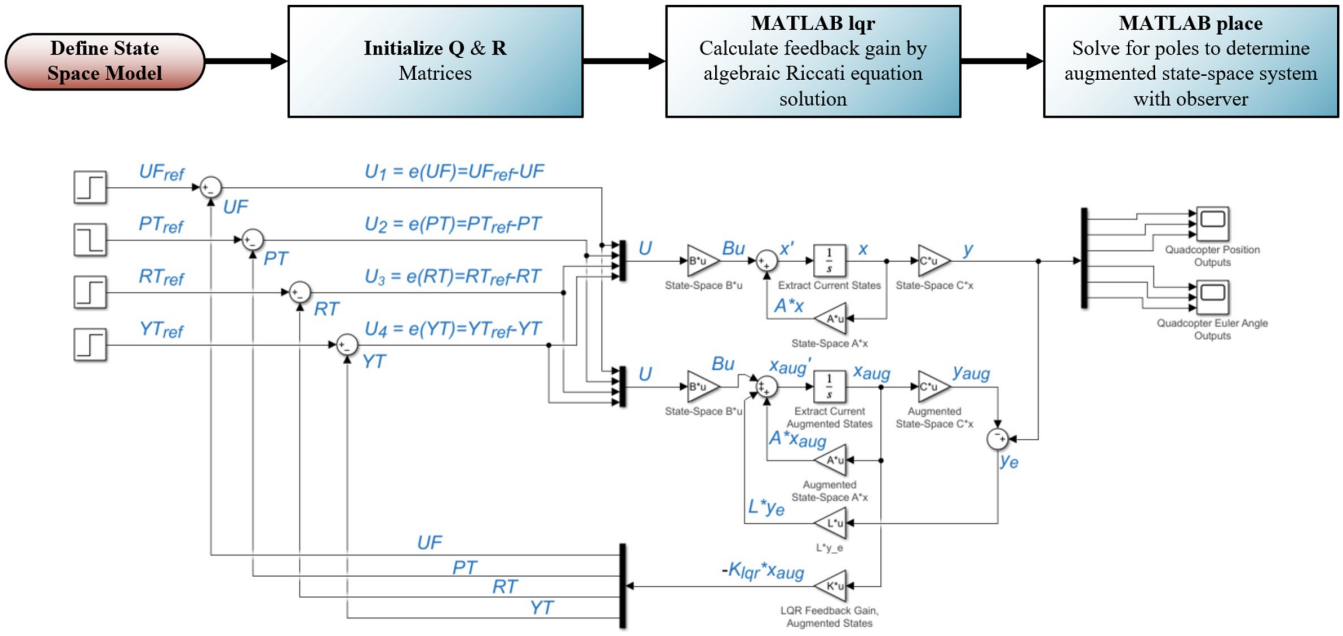

Fig. 3: Schematic for Augmented LQR-PI.

2) ***LQR:*** The $\mathbf{Q}$ and $\mathbf{R}$ matrices were initialised as: $\mathbf{R} = \mathbf{I}_4$, and $\mathbf{Q}$ was initialised using (13).
3) ***Augmented LQR-PI:*** The $\mathbf{Q}$ and $\mathbf{R}$ matrices were initialised similar to the LQR case. The pole placement was done using Ackermann's formula and the poles were placed 10x further away from the poles of the nominal LQR feedback.

Additionally, a cascaded PID-controller has also been implemented as a ROS node [8] for position-hold and trajectory-following within Gazebo simulator [9]. The Quadcopter model is derived from *RotorS* ROS-package [10], utilised for its attitude and angular velocity controller that integrates Geometric Methods on SE(3) [11].

Figure 4 shows the overview of ROS-Gazebo implementation. The input to the PID-controller is the error between the references and current pose (computed using an overhead camera localising a *Whycon marker* [12]).

Due to lack of space, only the results case 1 scenario 1 and case 2 are presented. However, all the codes, Simulink models used in this work, and the plots for all other scenarios, Gazebo

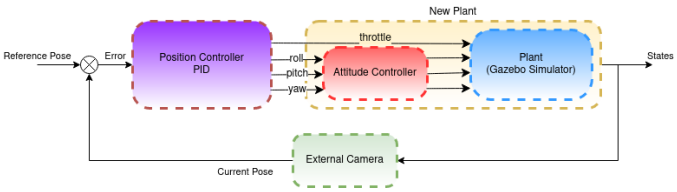

Fig. 4: Cascaded PID Controller for Position Hold and Trajectory Following.

setup and the corresponding results and videos, can be found at the Github link for the repository: [1]

A. Results

The simulation results for case 1, scenario 1, are plotted in fig. 5. The results obtained for the LQR and LQR-PI Observer are improved over results found in [1], because the error in the angles and the x and y positions are of the order of 10^{-16}. However, in [1], they were of the order of 10^{-12}. The PID-ITAE has a remarkable tracking ability when considering the x,y positions and the roll and pitch angles. However, there is a sharp impulse in the transient response of the PID-ITAE in the z position and the yaw angle. In this respect, the LQR and LQR-PI Observer perform better.

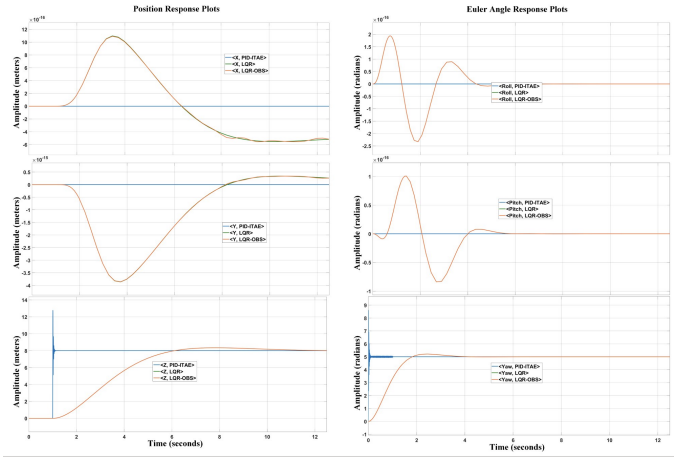

Fig. 5: Position and Attitude Tracking Results for Stability Analysis without External Disturbance or Uncertainty (Case 1, Scenario 1).

The simulation results for case 2 are shown in fig. 6.

B. Discussion

Based on all the scenarios and cases that were analysed, the following observations were made:

1) The PID-ITAE controller demonstrates the fastest settling time (nearly negligible) of compared controllers with the smallest overall trajectory error. The controller, however, exhibits large oscillations with high overshoot and undershoot upon the step response. The Euler angle responses are infeasible in their step response, observing up to 300 radian peak overshoot, implying that the Quadcopter has rotated over itself numerous times to reach its final state so quickly.
2) The LQR and LQR-PI Observer controllers have slower settling time but exhibit very small overshoot with no undershoot in the positions, and small undershoot in the Euler angle response. These controllers, however, ensure that the Euler angle response is practical and well within π-radians (180 degrees) for all simulated step responses.
3) The LQR and LQR-PI Observer controllers have excellent disturbance response rejection, whereas the PID-ITAE has poor disturbance rejection even showing observable error in settled positions after disturbance.

[1] https://github.com/Optimal-Control-Project-Quadcopter

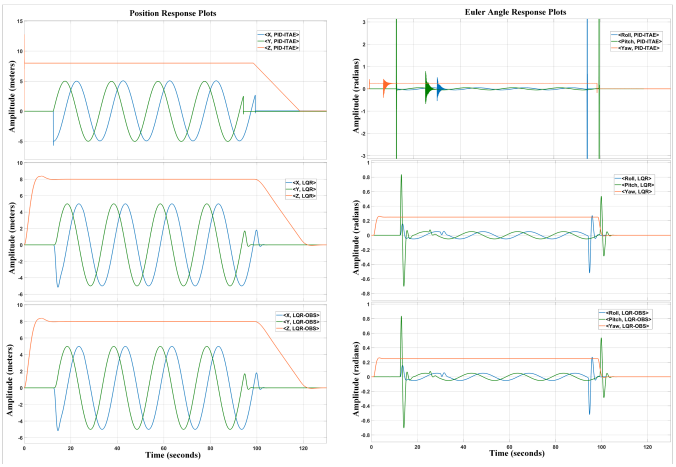

Fig. 6: Position and Attitude Tracking Results for Circular Trajectory Tracking with Disturbance (Case 2).

V. Project Experience and Learning Outcomes

This project was really exciting to work on. As a team, we really enjoyed the process of going through multiple resources to learn and implement new concepts. We learned a lot of things, not only in terms of optimal control methods, but also the process of research in general. Starting from performing an extensive literature survey to finalise a topic, to understanding the research paper theoretically and practically, to finally replicating it has taught us a lot. In terms of concepts, we got to understand how the equations of motion are translated to a state space model, the coupling effects of inputs-to-states and ways to decouple them. The ITAE tuning method was new and interesting to explore, since it is not as common as the Ziegler-Nichols tuning method, and optimizes PID gain constants for minimal trajectory error. The proposed initialisation of the $\mathbf{Q}$ matrix in [1] provided more insight on the regulation process. The LQR-PI Observer based control helped us understand the concept of observers in feedback control and the underlying concepts of pole-placement. In terms of implementation, we definitely improved our MATLAB and Simulink skills, which is beneficial for analyses in future. The implementation on Gazebo gave us a flavour of how the theoretical ideas are translated to hardware. The Gazebo simulation also helped us learn more about the separation of inner and outer feedback

loop, sensor fusion, and the practical importance of overshoot, undershoot and settling time.

VI. Lyapunov Stability Analysis

The stability of the LQR and Augmented LQR-PI Observer algorithms can be verified using the Lyapunov's theorem for stability. Lyapunov's theorem states that if a positive definite function $V(\mathbf{x}(t))$, has a negative semi-definite gradient, i.e. $\dot{V}(\mathbf{x}(t)) \leq \mathbf{0}$, then the system is asymptotically stable.

Let $V(\mathbf{x}(t))$ be chosen as follows:

$$V(\mathbf{x}(t)) = \mathbf{x}^T(t)\mathbf{P}(t)\mathbf{x}(t) \tag{19}$$

where $\mathbf{P}(t)$ is a positive definite matrix. For simplicity, the time dependence is not explicitly mentioned from now on.Taking the derivative of (19) gives the following expression for $V(\mathbf{x}(t))$:

$$\dot{V}(\mathbf{x}) = \dot{\mathbf{x}}^T\mathbf{P}\mathbf{x} + \mathbf{x}^T\dot{\mathbf{P}}\mathbf{x} + \mathbf{x}^T\mathbf{P}\dot{\mathbf{x}} \tag{20}$$

Substituting the state space equation in (20) along with the LQR feedback law, and rearranging terms, the final expression for $\dot{V}(\mathbf{x})$ is given by:

$$\dot{V}(\mathbf{x}) = -\mathbf{x}^T\Big[\mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} + \mathbf{Q}\Big]\mathbf{x} \tag{21}$$

which is negative semi-definite if $\mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} + \mathbf{Q}$ is positive semi-definite. For this to be true, $\mathbf{Q}$ must be positive semi-definite, and $\mathbf{R}$ must be positive definite. Given these conditions, the LQR and augmented LQR-PI Observer systems are asymptotically stable.

VII. Conclusions and Future Prospects

This work performed a comparative study of three optimal control methods, namely, the PID-ITAE, LQR and LQR-PI Observer based control, on a 6-DOF Quadcopter. The state-space model was presented, and the controllers were analysed under different scenarios of stability (with external disturbance and uncertainty modelling), and a case of trajectory tracking. Based on the analysis, we conclude that for the system in consideration, the LQR and LQR-PI Observer control are better methods. Additionally, the PID controller was implemented on Gazebo to study practical effects of the optimal controller. A Lyapunov stability analysis was also presented to validate the asymptotic convergence.

Possible directions of future work would be to implement the LQR and LQR-PI on Gazebo. Another work could be to investigate further on how to initialise the $\mathbf{R}$ matrix in the LQR problems. Further, the $\mathbf{Q}$ and $\mathbf{R}$ matrices could be adapted iteratively in the cost function to provide more robustness. Another possibility is to delve into the tuning of the PID gains using a different technology or modify the ITAE method.

VIII. Statement of Contributions

Connor Reece (ASU ID #1205970957) worked on the PID-ITAE and the Simulink models for all the controllers. He also contributed mostly in the making of the PPT. He also compiled the results, and worked on the disturbance modeling.

Connor Reece
Vishal Gupta
Kaushik Iyer

Vishal Gupta (ASU ID # 1226206568) worked on the Gazebo implementation and video demonstrations (extension of a previous work with Rishikesh Madan at the ERTS Lab, IIT Bombay) and contributed in writing all the relevant text for it in the IEEE-format document and the PPT.

Connor Reece
Vishal Gupta
Kaushik Iyer

Kaushik Iyer (ASU ID #1223696175) worked on the LQR and LQR-PI Observer control. He wrote the MATLAB codes for them. He also wrote IEEE-format document for the most part. He also performed the Lyapunov analysis.

Connor Reece
Vishal Gupta
Kaushik Iyer

References

[1] Aisha Sir Elkhatem and Seref Naci Engin. Robust LQR and LQR-PI control strategies based on adaptive weighting matrix selection for a UAV position and attitude tracking control. *Alexandria Engineering Journal*, 61(8):6275–6292, 2022.

[2] Lucas M Argentim, Willian C Rezende, Paulo E Santos, and Renato A Aguiar. PID, LQR and LQR-PID on a quadcopter platform. In *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*, pages 1–6. IEEE, 2013.

[3] Bouzgou Kamel, Bestaoui Yasmina, Benchikh Laredj, Ibari Benaoumeur, and Ahmed-Foitih Zoubir. Dynamic modeling, simulation and PID controller of unmanned aerial vehicle UAV. In *2017 Seventh International Conference on Innovative Computing Technology (INTECH)*, pages 64–69, 2017.

[4] Ala Eldin Abdallah Awouda and Rosbi Bin Mamat. Refine PID tuning rule using ITAE criteria. In *2010 The 2nd International conference on computer and automation engineering (ICCAE)*, volume 5, pages 171–176. IEEE, 2010.

[5] Fernando G Martins. Tuning PID controllers using the ITAE criterion. *International Journal of Engineering Education*, 21(5):867, 2005.

[6] Zaid Tahir, Waleed Tahir, and Saad Ali Liaqat. State space system modelling of a quad copter UAV. *arXiv preprint arXiv:1908.07401*, 2019.

[7] Pengcheng Wang, Zhihong Man, Zhenwei Cao, Jinchuan Zheng, and Yong Zhao. Dynamics modelling and linear control of quadcopter. In *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pages 498–503. IEEE, 2016.

[8] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[9] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[10] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. Rotors—a modular gazebo mav simulator framework. *Robot Operating System (ROS) The Complete Reference (Volume 1)*, pages 595–625, 2016.

[11] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Geometric tracking control of a quadrotor uav on se (3). In *49th IEEE conference on decision and control (CDC)*, pages 5420–5425. IEEE, 2010.

[12] Tomáš Krajník, Matías Nitsche, Jan Faigl, Petr Vaněk, Martin Saska, Libor Přeučil, Tom Duckett, and Marta Mejail. A practical multirobot localization system. *Journal of Intelligent & Robotic Systems*, 2014.