RBE 502: Implementation of PID and LQR Controller

Simran Chauhan

December 16, 2024

1 Introduction

This report presents the implementation and comparison of PID and LQR controllers for trajectory tracking of a 3D quadrotor along circular and diamond-shaped helix trajectories. The study aims to evaluate the effectiveness of these control techniques in achieving accurate trajectory tracking, stability, and performance in real-world applications.This implicated no aggressive motion with large roll, pitch or yaw angle.It is important to note that the quadrotor's state was assumed to remain in an hovering condition during the entire trajectory.

2 Methodology

2.1 PD Controller

The PID controller was implemented using a simple proportional-derivative approach. The controller parameters were tuned manually to achieve a reasonable balance between the system's response time and accuracy. The PID controller adjusts the control input based on the error between the current state and the desired state, using a weighted sum of proportional, and derivative terms.

2.1.1 Quadrotor Dynamics

2.1.2 System Model

The quadrotor's state are described by the following vectors:

$$X = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ p \\ q \\ r \end{bmatrix}$$

where:

- x, y, z are the positions.

- ϕ, θ, ψ are the Euler angles (roll, pitch, yaw).
- $\dot{x}, \dot{y}, \dot{z}$ are the linear velocities.
- p, q, r are the angular velocities.

The system's control input vector is:

$$U = \begin{bmatrix} F \\ M_\phi \\ M_\theta \\ M_\psi \end{bmatrix}$$

where:

- F is the thrust.
- M_ϕ, M_θ, M_ψ are the moments (torques) along the roll, pitch, and yaw axes.

The quadrotor dynamics are linearized around the nominal hover state to simplify the control design. The equations of motion in the inertial frame are given by:

$$\dot{x} = Ax + Bu, \tag{1}$$

where x represents the state vector and u is the control input vector. The key parameters of the quadrotor are provided in below Table .

Parameter	Value	Description
l	0.046 m	Distance from center of mass to rotor
m	0.03 kg	Total mass of the quadrotor
I_{xx}, I_{yy}	1.43 kg·m^2	Moment of inertia *(roll, pitch)*
I_{zz}	2.89 kg·m^2	Moment of inertia *(yaw)*
g	9.81 m/s^2	Gravitational acceleration

Table 1: Quadrotor Parameters

2.1.3 Position Control

The errors in position and velocity are computed as follows:

$$\text{pos\_error} = \mathbf{pos}_{des} - \mathbf{pos}$$

$$\text{vel\_error} = \mathbf{vel}_{des} - \mathbf{vel}$$

where desired velocity and desired position we are getting from the trajectory generator.The system state in MATLAB as follows:

$$\text{Position: } \mathbf{pos} = [x, y, z]^T, \quad \text{Velocity: } \mathbf{vel} = [xd, yd, zd]^T$$

$$\text{Euler Angles } \mathbf{euler} = [\phi, \theta, \psi]^T, \quad \text{Angular Velocity: } \omega = [p, q, r]^T$$

$$\text{quadrotor state: } = [pos, vel, euler, \omega]^T$$

The command accelaration $\mathbf{acc}_{cmd}$ is then computed using PD Controllers ad below:

$$\mathbf{acc}_{cmd} = K_{p_{pos}} \cdot \text{pos\_error} + K_{d_{pos}} \cdot \text{vel\_error} + acc_{des}$$

where $K_{p_{pos}}$ and $K_{d_{pos}}$ are the proportional and derivative gains for position control, respectively. The proportional gain matrix$K_{p_{pos}}$ and derivative gain matrix $K_{d_{pos}}$ are defined as follows:

$$K_{p_{pos}} = \begin{bmatrix} K_{p1} & 0 & 0 \\ 0 & K_{p2} & 0 \\ 0 & 0 & K_{p3} \end{bmatrix}, \quad K_{d_{pos}} = \begin{bmatrix} K_{d1} & 0 & 0 \\ 0 & K_{d2} & 0 \\ 0 & 0 & K_{d3} \end{bmatrix}$$

The required thrust is then computed as:

$$F = m \cdot (g + \mathbf{acc}_{cmd}(3))$$

where m is the quadrotor's mass, and g is the gravitational constant. The value of the gains

Gain	x	y	z
K_p	17	17	12
K_d	16.4	16.4	10

Table 2: Position Controller Gains of PD Controller in Trajectory Tracking

used in PD controller listed in below table:

2.1.4 Attitude Control

Attitude control is performed for roll, pitch, and yaw. The desired roll (ϕ_{des}) and pitch (θ_{des}) are computed based on the desired accelerations in the x and y axes, as well as the desired yaw angle. The orientation error is given by:

$$\text{att\_error} = [\phi_{des}; \theta_{des}; \text{yaw}_{des}] - \mathbf{euler}$$

The angular velocity error is computed as:

$$\text{ang\_vel\_error} = \mathbf{pqr}_{des} - \mathbf{omega}$$

The required moments to adjust the quadrotor's orientation are calculated as:

$$M = I \cdot (K_{p_{att}} \cdot \text{att\_error} + K_{d_{att}} \cdot \text{ang\_vel\_error})$$

where I is the inertia matrix, and $K_{p_{att}}$ and $K_{d_{att}}$ are the proportional and derivative gains for attitude control. The proportional gain matrix$K_{p_{att}}$ and derivative gain matrix $K_{d_{att}}$ for attitude control are defined as follows:

$$K_{p_{att}} = \begin{bmatrix} K_{p1} & 0 & 0 \\ 0 & K_{p2} & 0 \\ 0 & 0 & K_{p3} \end{bmatrix}, \quad K_{d_{att}} = \begin{bmatrix} K_{d1} & 0 & 0 \\ 0 & K_{d2} & 0 \\ 0 & 0 & K_{d3} \end{bmatrix}$$

The value of the attitude control gains used in my PD controller are listed in the following table:

Gain	x	y	z
$K_{p_{att}}$	30	30	20
$K_{d_{att}}$	15	15	18

Table 3: Atitude Controller Gains of PD Controller in Trajectory Tracking

The PD controller adjusts the control inputs (thrust and moments) based on the position, velocity, and orientation errors. The proportional terms drive the system towards the desired states, while the derivative terms help dampen oscillations and overshoot, ensuring smooth trajectory tracking.

2.2 LQR Controller

The LQR controller was implemented with the goal of minimizing a quadratic cost function that considers both the state deviation and control input deviation. The LQR controller computes the control inputs that minimize a quadratic cost function over time.

The state cost matrix Q and the control cost matrix R were chosen to balance state error minimization and control effort, with Q being proportional to the position error and R being proportional to the square of the control input.

2.2.1 LQR Controller Design

The Linear Quadratic Regulator (LQR) aims to minimize the following quadratic cost function:

$$J = \sum_{i=0}^{N-1} \big((X_{des,i} - X_i)^T Q (X_{des,i} - X_i) + (u_{des,i} - u_i)^T R (u_{des,i} - u_i) + (X_{des,N} - X_N)^T Q_N (X_{des,N} - X_N)\big)$$

subject to:

$$X_{i+1} = f(X_i, u_i)$$

where:

- Q is the state weighting matrix that penalizes deviations from the desired state.
- R is the control input weighting matrix that penalizes excessive control inputs.

We linearize around a nominal operating point $X_i = X_{i,\text{des}}$ and $U_i = U_{i,\text{des}}$. The linearized approximation is given as:

$$X_{i+1} \approx f(X_{i,\text{des}}, U_{i,\text{des}}) + \left.\frac{\partial f}{\partial X}\right|_{X_i = X_{i,\text{des}}, U_i = U_{i,\text{des}}} (X_i - X_{i,\text{des}}) + \left.\frac{\partial f}{\partial U}\right|_{X_i = X_{i,\text{des}}, U_i = U_{i,\text{des}}} (U_i - U_{i,\text{des}})$$

Where:

- X_i: State vector at time i
- U_i: Control input at time i
- f: Nonlinear function describing the system dynamics
- $A = \frac{\partial f}{\partial X}$: Jacobian of f with respect to X at $X_{i,\text{des}}$, $U_{i,\text{des}}$.
- $B = \frac{\partial f}{\partial U}$: Jacobian of f with respect to U at $X_{i,\text{des}}$, $U_{i,\text{des}}$.
- $X_{i,\text{des}}$: Desired or nominal state

$$X_{\text{des}} = \begin{bmatrix} \text{pos}_{\text{des}} \\ \text{euler}_{\text{des}} \\ \text{vel}_{\text{des}} \\ \text{w}_{\text{des}} \end{bmatrix}$$

- $U_{i,\text{des}}$: Desired or nominal input

$$u_{\text{des}} = \begin{bmatrix} m \cdot g \\ M \end{bmatrix}$$

The optimization problem can be written as follows:

$$\min \sum_{i=0}^{N-1} e_i^T Q e_i + e_{ui}^T R e_{ui} + e_N^T Q_N e_N$$

where the error terms are defined as:

$$e_i = X_{\text{des},i} - X_i, \quad e_{ui} = U_{\text{des},i} - U_i$$

The system dynamics are given by:

$$X_{i+1} = f(X_i, U_i)$$

For the linearized system, the error dynamics can be expressed as:

$$e_{i+1} = A_i e_i + B_i e_{ui}$$

$$e_{ui} = u - u_{\text{des}} = -K e_i$$

Hence, the control input is given by

$$u = -K * e_i + u_{\text{des}}$$

where K solution is given by below equation $\mathrm{K}_i = \left(\mathbf{R}_i + \mathbf{B}_i^T \mathbf{P}_{i+1} \mathbf{B}_i\right)^{-1} \mathbf{B}_i^T \mathbf{P}_{i+1} \mathbf{A}_i$
$\mathbf{P}_i = \mathbf{Q}_i + \mathbf{A}_i^T \mathbf{P}_{i+1} \mathbf{A}_i - \mathbf{A}_i^T \mathbf{P}_{i+1} \mathbf{B}_i \mathbf{K}_i$

The controller is implemented in MATLAB using the 'lqr' function, which computes the optimal gain matrix K given the system matrices A, B, and the cost matrices Q and R.

$$[K, S] = \text{lqr}(A, B, Q, R)$$

where: - A and B are the system's state and control input matrices (linearized). - Q and R are the state and control input weighting matrices, respectively. - K is the optimal state feedback gain matrix Upon linearisation, the A and B matrices are defined as below:

$$A = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -p\sin\theta + r\cos\theta & 0 & 0 & 0 & 0 & \cos\theta & 0 & \sin\theta \\
0 & 0 & 0 & \sec^2\phi(p\sin\theta - r\cos\theta) & \tan\phi(p\cos\theta + r\sin\theta) & 0 & 0 & 0 & 0 & \sin\theta\tan\phi & 1 & -\cos\theta\tan\phi \\
0 & 0 & 0 & \tan\phi\sec\phi(r\cos\theta - p\sin\theta) & \sec\phi(-p\cos\theta - r\sin\theta) & 0 & 0 & 0 & 0 & -\frac{\sin\theta}{\cos\phi} & 0 & \frac{\cos\theta}{\cos\phi} \\
0 & 0 & 0 & g\sin\psi & g\cos\psi & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -g\cos\psi & g\sin\psi & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$B = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\frac{1}{m} & 0 & 0 & 0 \\
0 & \frac{1}{I_{xx}} & 0 & 0 \\
0 & 0 & \frac{1}{I_{yy}} & 0 \\
0 & 0 & 0 & \frac{1}{I_{zz}}
\end{bmatrix}$$

Here, the matrices A and B are computed through simulation based on the desired state (generated by the trajectory generator) and desired input.

$X_{i,\text{des}}$: Desired or nominal state

$$X_{\text{des}} = \begin{bmatrix} \text{pos}_{\text{des}} \\ \text{euler}_{\text{des}} \\ \text{vel}_{\text{des}} \\ \text{w}_{\text{des}} \end{bmatrix}$$

$U_{i,\text{des}}$: Desired or nominal input

$$u_{\text{des}} = \begin{bmatrix} m \cdot g \\ M \end{bmatrix}$$

The weight matrices Q and R used in the Linear Quadratic Regulator (LQR) simulation are defined as follows:

$$Q = \begin{bmatrix} 30 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 \\ 0 & 30 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 0 \\ 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 & 0 & 0 \\ 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 \\ 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 \end{bmatrix}$$

Here, Q is a 12x12 matrix used to penalize deviations from the desired state in the optimization problem.
The diagonal elements of Q represent the relative weightings assigned to the state variables, where the larger values correspond to higher penalties for deviation in those states.

$$R = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

On the other hand, R is a 4x4 diagonal matrix that penalizes the control effort. The elements of R correspond to the weights of the control inputs in the LQR cost.

3 Experiment Results

3.1 Simulation Setup

The quadrotor was simulated to follow a circular trajectory and diamond trajectory. The simulation was run for both PID and LQR controllers, and the results were compared to analyze the performance.

- **Trajectory Generator 1: circle.m**

- A helix in the xy-plane of radius 1 meter, centered at the point $(0.5, 0, 0)$, starting at the point $(1, 0, 0)$.
- The z-coordinate should start at 0 and end at 0.5.
- The helix should go counter-clockwise.

- **Trajectory Generator 2: diamond.m**
  - A "diamond helix" with corners at the following points when projected into the yz-plane:
$$(0,0,0),(0,\sqrt{2},\sqrt{2}),(0,0,2\sqrt{2}),(0,-\sqrt{2},\sqrt{2})$$
  - The x-coordinate should start at 0 and end at 1.
  - The quadrotor should start at $(0, 0, 0)$ and end at $(1, 0, 0)$.

3.2 PID Controller Results

The quadrotor controlled by the PID controller was able to follow the trajectory with reasonable accuracy, though it experienced some oscillations and delays in settling time. The proportional and derivative gains were crucial in determining the system's stability and responsiveness.The results are verified at the end of report.

3.3 LQR Controller Results

The quadrotor controlled by the LQR controller exhibited smoother trajectory following and faster convergence compared to the PID-controlled system. The optimal gains from LQR helped in reducing the control effort while maintaining accurate trajectory tracking.The results are verified at the end of report.

4 Comparison between LQR Controller and PID Controller

The performance of the PID and LQR controllers was compared based on response time, stability, and control effort. The PID controller offered simplicity in implementation and tuning, but it was prone to oscillations and had slower convergence, which impacted its efficiency. On the other hand, the LQR controller demonstrated smoother control and greater accuracy, but it was computationally intensive, which made it slower.

4.1 Pros and Cons

- **PID Controller:**
  - Pros: Simple to implement, easy to tune, suitable for many applications.
  - Cons: Struggles with complex dynamics, prone to oscillations, tuning can be time-consuming.
  - Does not optimize control effort versus tracking error, leading to less smooth trajectories compared to LQR.
- **LQR Controller:**
  - Pros: Optimal control, smoother trajectory following, faster convergence.

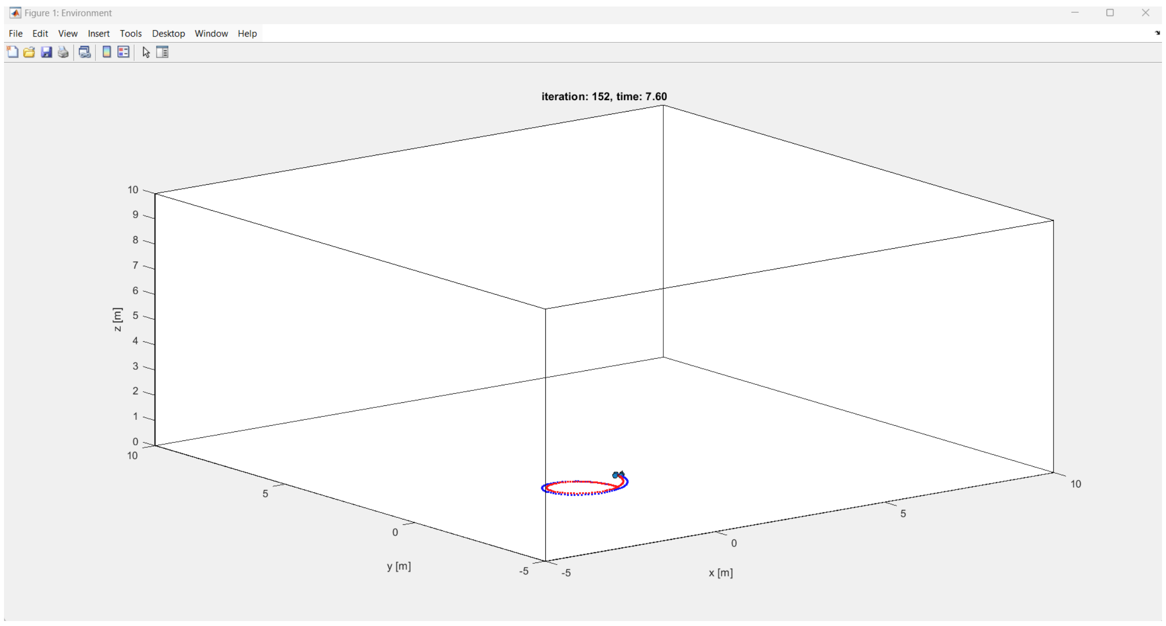

Figure 1: PD Circle Trajectory

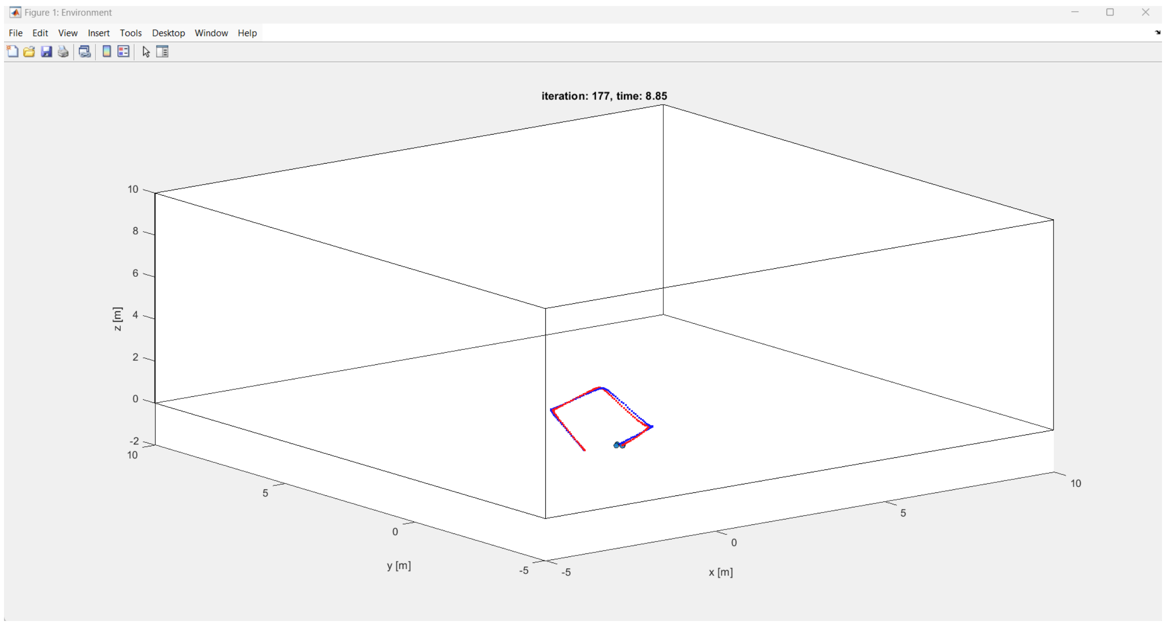

Figure 2: PD Diamond Trajectory

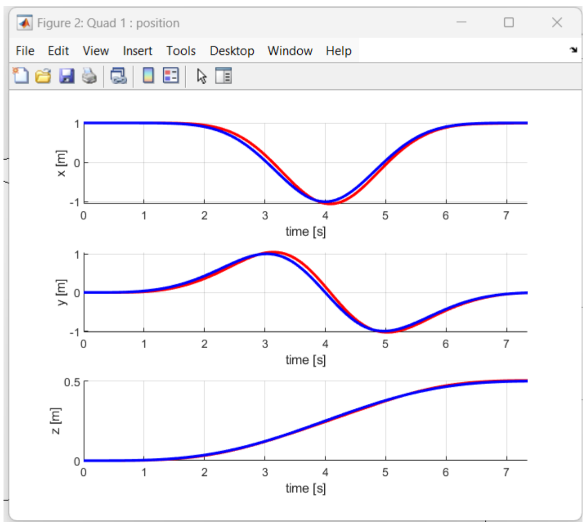

Figure 3: Circle Position Graph

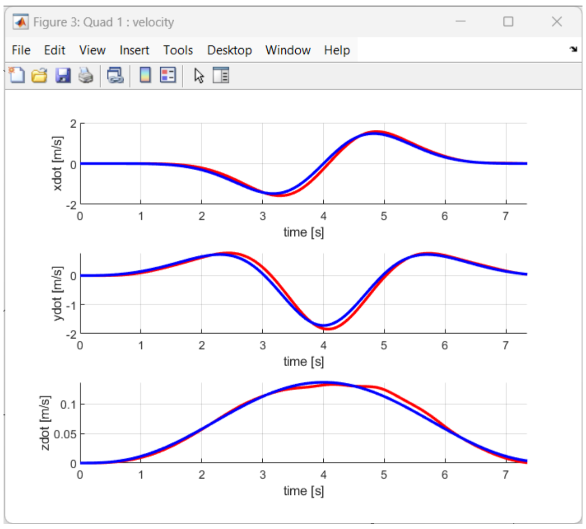

Figure 4: Circle Velocity Graph

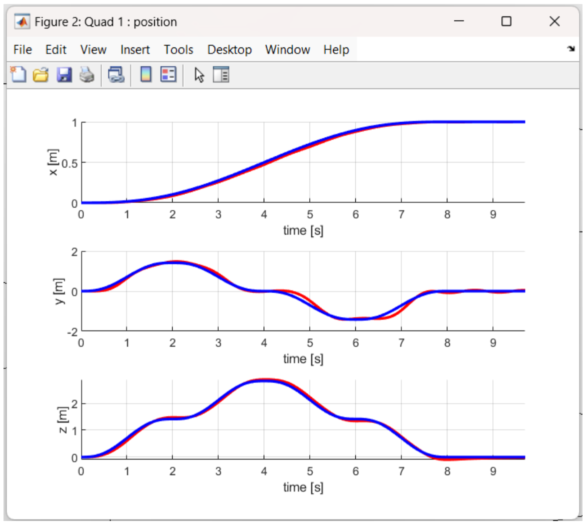

Figure 5: Diamond Position Graph

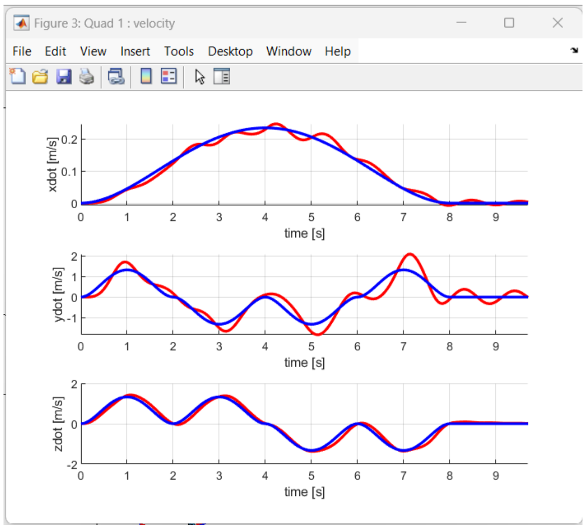

Figure 6: Diamond Velocity Graph

Figure 7: PD Controller Results for Circle and Diamond Trajectories.

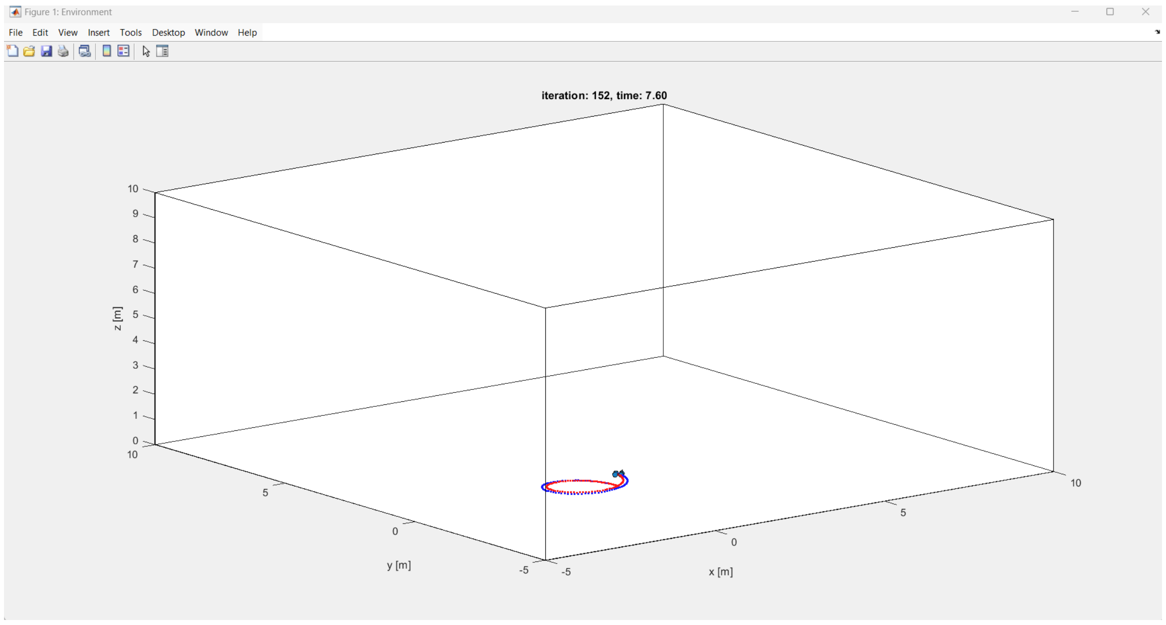

Figure 8: LQR Circle Trajectory

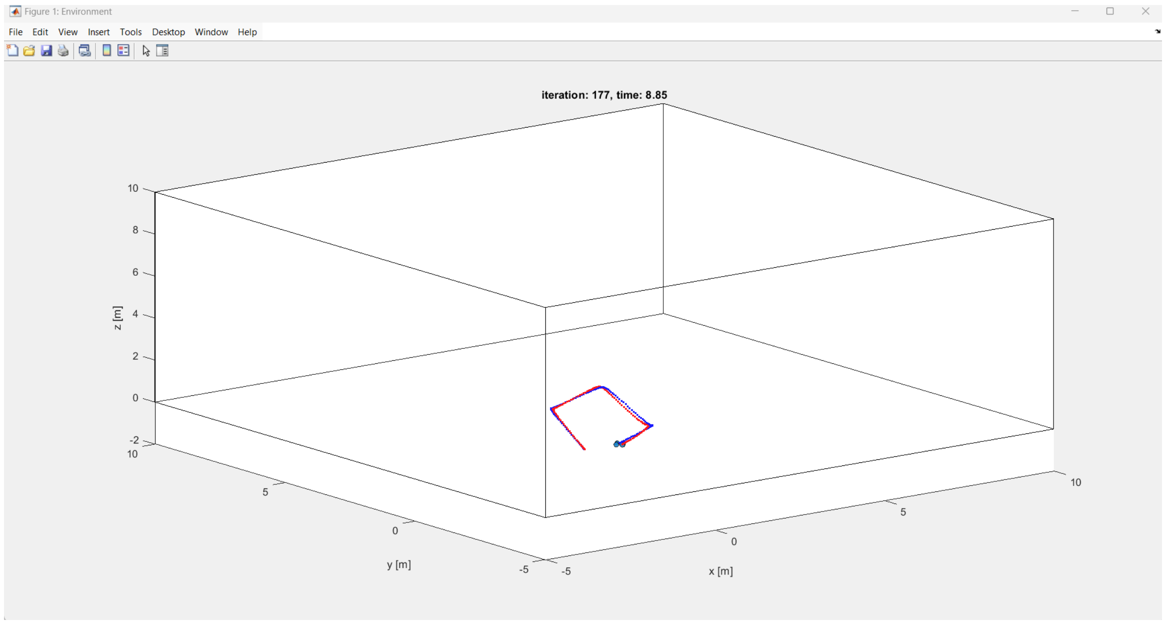

Figure 9: LQR Diamond Trajectory

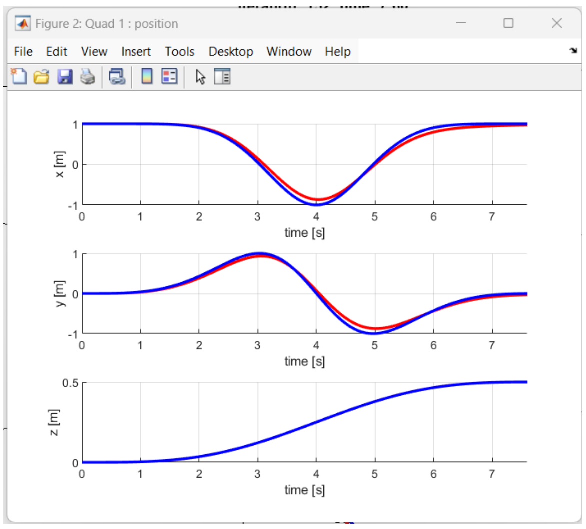

Figure 10: Circle Position Graph

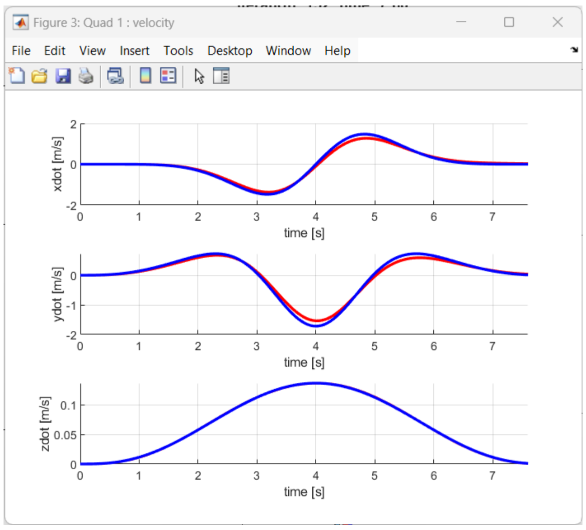

Figure 11: Circle Velocity Graph

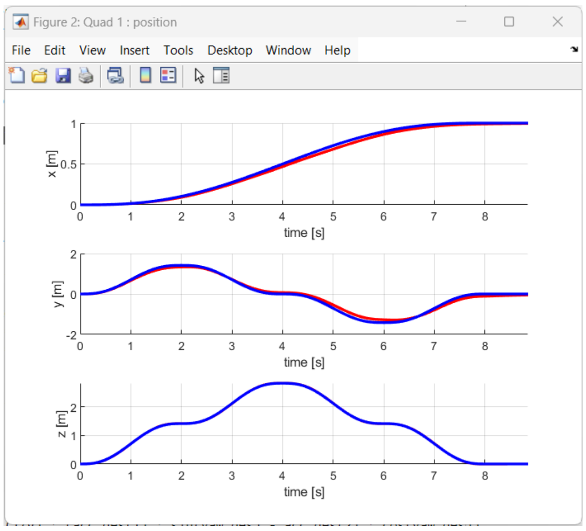

Figure 12: Diamond Position Graph

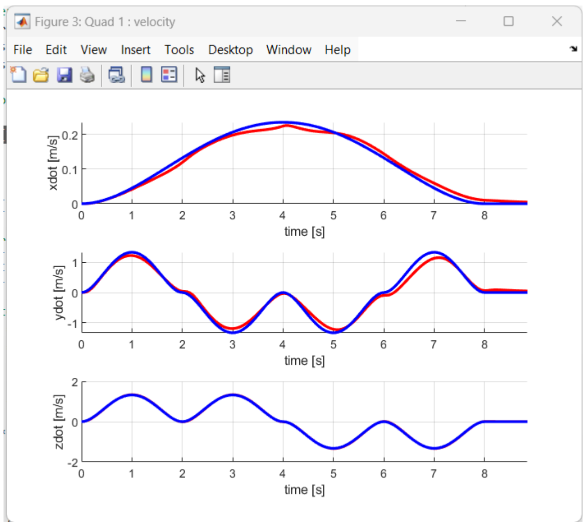

Figure 13: Diamond Velocity Graph

Figure 14: LQR Controller Results for Circle and Diamond Trajectories.

  - Cons: Requires precise system modeling, computationally intensive, requires tuning of weighting matrices.

5 Conclusion

In conclusion, the PID controller is simpler and more straightforward to implement, but it is not effective in systems with more complex dynamics. The LQR controller, on the other hand, provides better performance in terms of stability and convergence but requires more computational resources and tuning.