Avanti PID and LQR Altitude Control

Andres Pulido

November, 2021

1 Problem Statement

You are tasked with designing an autopilot for an Avanti S 80mm EDF. Specifically, you will focus on automating the change in altitude for obstacle avoidance. You must consider the flight dynamics of the aircraft along with the dynamics of the actuator that moves the elevator and generates the required moment. The scenario used to evaluate the autopilot begins with the Avanti S traveling straight-and- level at 50 miles-per-hour and an altitude of 100 m. The aircraft is part of a fleet that are tasked with surveillance of an area by flying over the same location but having different altitude. The Avanti S needs to increase altitude by 100 m to avoid a building; however, it must not increase by more than 120 m or it will enter the airspace of the aircraft flying above it.

- q : measured pitch rate (deg/s)
- qc : commanded pitch rate (deg/s)
- θ : measured pitch angle (deg/s)
- θ_c : commanded pitch angle (deg)
- h : measured altitude (m)
- hc : commanded altitude (m)
- δ : elevator deflection angle (deg)
- δ_c : commanded elevator deflection angle (deg)

2 Solution

2.1 Discussion of control design

To solve the tracking problem, a PID and a LQR controller were designed and tuned. The PID controller was implemented in Simulink (Figure 1 because it is relatively quick to implement and then tune based on the simulation plots. The LQR was implemented in normal scripting Matlab.

Even in the faster development environment of Simulink, the PID controller took way more time to tune and arrive to get to the desired performance. On the other hand, LQR was fairly quick since it I intuitively knew what I wanted to penalize and how the behavior of the response would change depending on the parameters I changed, which was not the case of the PID controller. Because of the sequential loop design, it was hard to figure out which gains I should start with and how would changing them would affect the behavior of the response.

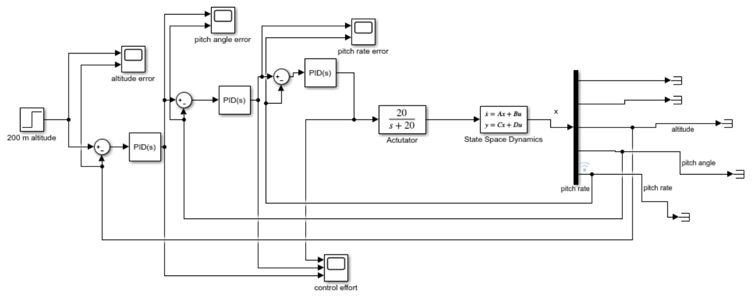

Figure 1: Block Diagram of PID controller in Simulink

2.2 Performance Plots

The PID controller was able to reach the desired performance. In Figure 9, the rise time can be seen to have a value of more than 0.8s. In Figure 6, the overshoot can be seen to be less than 220m, and in Figure 10, there can be seen that there is no steady state error.

The LQR controller was able to achieve a rise time of around 1s, and an overshoot barely more than 5%, which can be seen in Figure 2. Finally, in terms of performance, the steady state error can be seen to be minimal, Figure 5.

To be complete, the pitch angle of the PID and LQR controllers, respectively, are shown in Figure 8 and 3 as well as the pitch rate of the PID controller Figure 7, and the LQR controller, Figure 4.

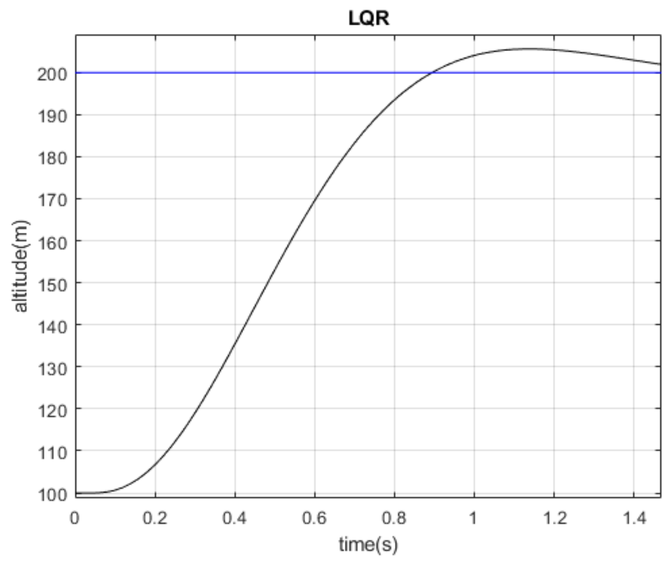

Figure 2: Rise time of LQR Controller

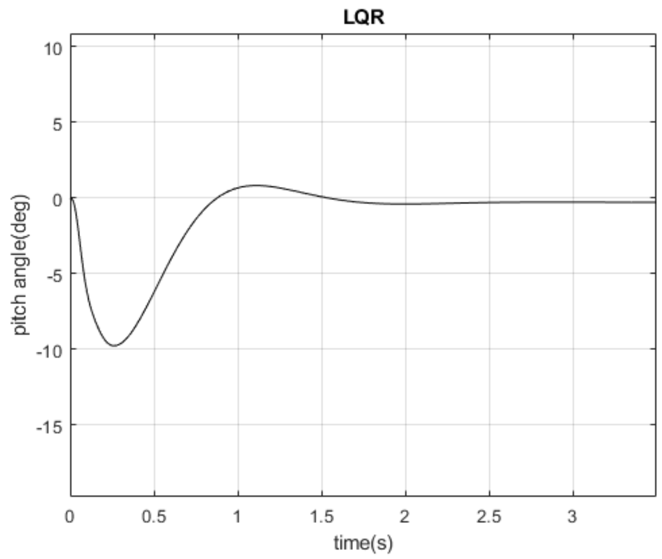

Figure 3: Pitch angle of LQR Controller

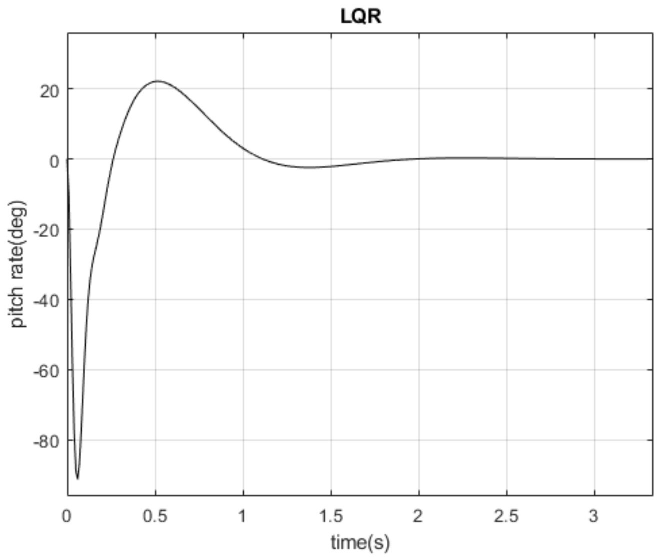

Figure 4: Pitch rate of LQR Controller

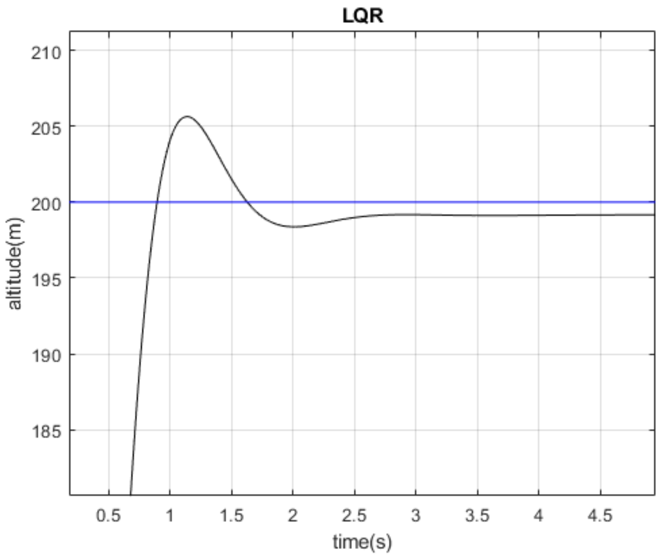

Figure 5: Steady State Error of LQR Controller

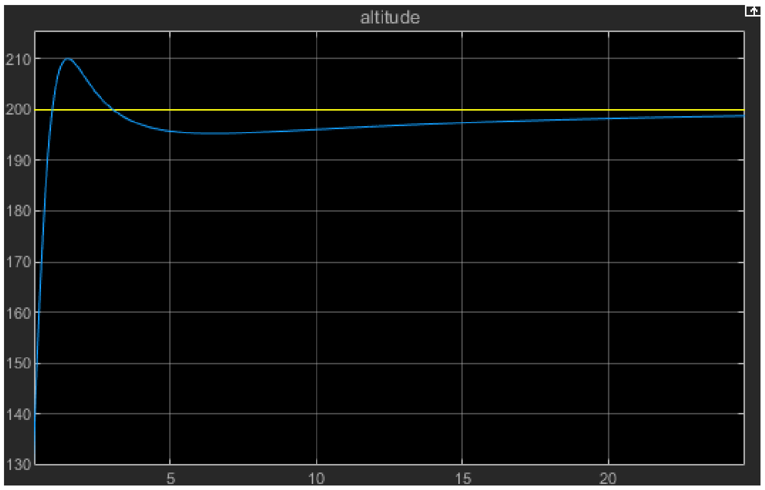

Figure 6: Overshoot of PID Controller

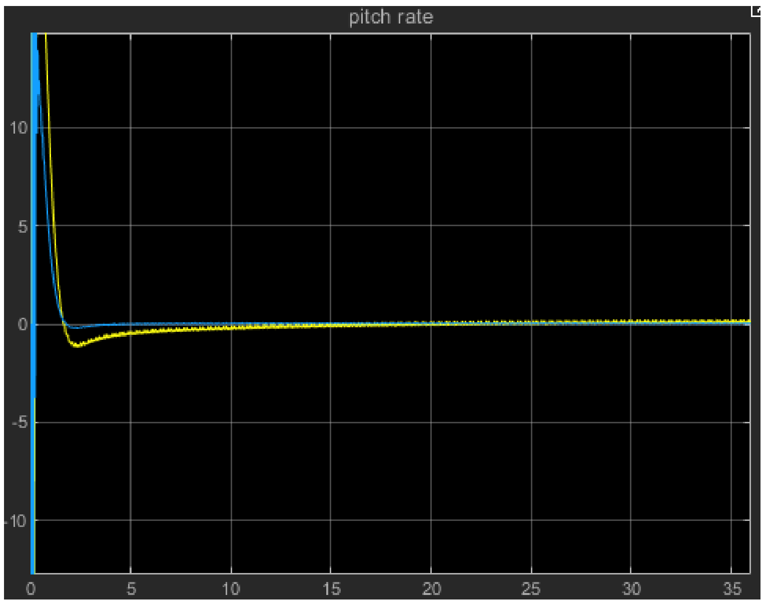

Figure 7: Pitch rate of PID Controller

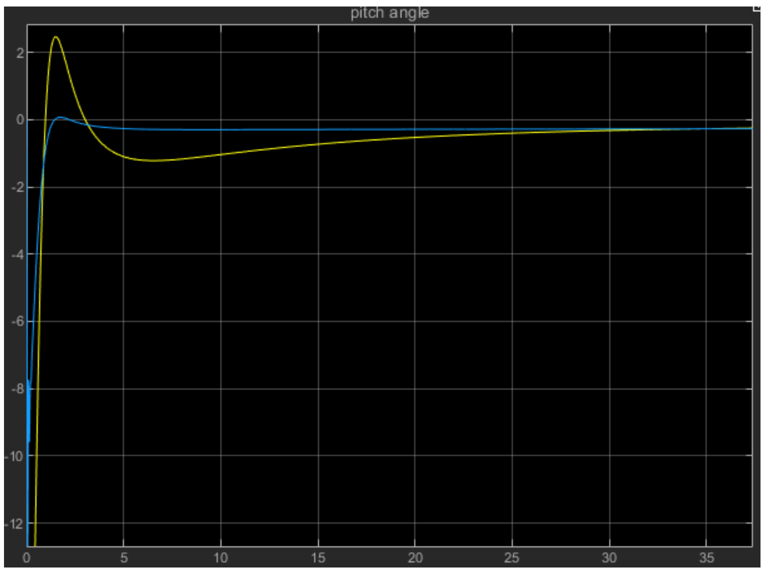

Figure 8: Pitch Angle of PID Controller

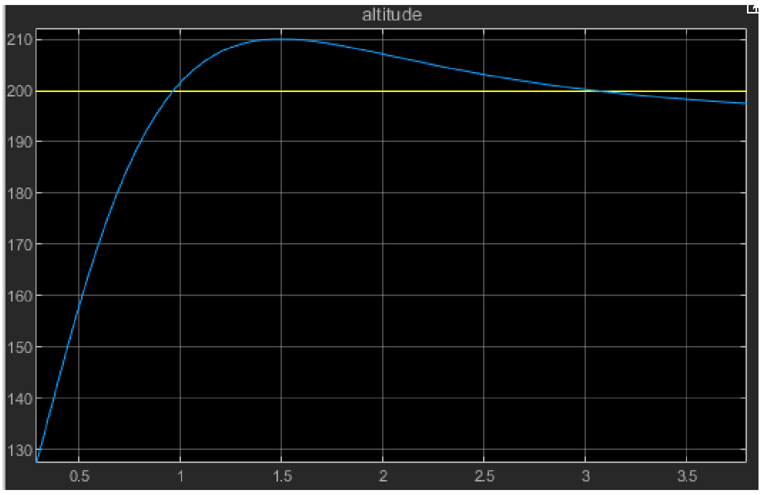

Figure 9: Rise time of PID Controller

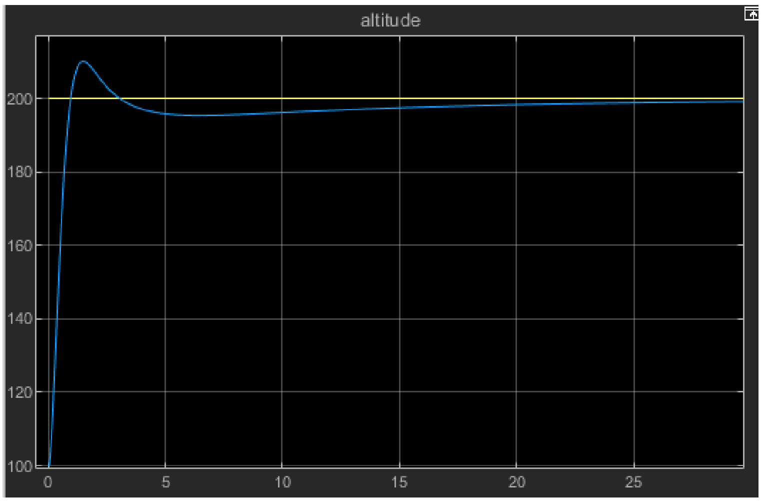

Figure 10: Steady State Error of PID Controller