**Lab 2**

18 November 2024

Authors: C1C Lino De Ros & C1C Connor Emmons



Professor: Dr. Brown

Engr 443 T2

**Documentation:** Used course notes and previous lab reports for formatting purposes. ChatGPT used to help write the approach and conclusion sections, though final edits were made by us. All other sections of the report were generated solely by us. No unauthorized resources used. ChatGPT prompt can be found at <https://chatgpt.com/share/673aee5b-7fdc-8008-a2c5-b66a4a4a0a11>.

**Objective**

The objective of this lab is to design and evaluate a LQ regulator and a LQ controller with a given end time for a single-axis spacecraft attitude control system using optimal control theory.

**Approach**

The approach to this lab focuses on developing and analyzing linear quadratic (LQ) feedback systems for a single-axis spacecraft attitude control problem. Tasks are divided into implementing a state-space model, designing a linear quadratic regulator (LQR), and designing a linear quadratic controller (LQC) with a terminal time. The following methods are employed:

1. **State-Space Model Implementation**:  
   The spacecraft model is represented in state-space form using Simulink. This provides a baseline for system dynamics and set the stage for control design.
2. **LQR Design**:  
   The regulator minimizes a cost function to stabilize the spacecraft in a desired configuration indefinitely. Using weighting matrices and , the optimal feedback gain is calculated from the steady-state Riccati equation. Analytical solutions provide the constant gain matrix required for the control law.
3. **LQC Design**:  
   The controller aims to drive the system states to zero within a specified time (30 seconds). Weighting matrices , , and incorporate the final state constraints. The time-varying Riccati matrix is computed by integrating backward from the final condition ​, providing a dynamic gain matrix during simulation.
4. **Validation and Simulation**:  
   Simulink models and MATLAB functions are used to validate the control designs. Plots of system states, Riccati matrix components, and control inputs are used to verify that theoretical predictions match system behavior under both LQR and LQC designs.

**Assumptions**

The base system is assumed to be uncoupled (i.e., the inputs do not directly affect the outputs without affecting the states).

**Mathematical Techniques**

**Task 1:**

The spacecraft model is implemented in Simulink. The spacecraft model is given in state space, so

(1)

(2)

where

(3)

The full state linear quadratic feedback requirement of Tasks 2 and 3 combined with the assumption that the system is uncoupled (i.e., the inputs do not directly affect the output without affecting the states) give:

(4)

Note that for the system described by the specified values in equations (3) and (4), there are 2 states, one input, and two outputs (which are just the states).

**Task 2:**

A linear quadratic regulator is developed for the base spacecraft system. For a regulator, the cost function to be minimized is

(5)

and the control law takes the form

(6)

where

(7)

The Riccati Matrix, , is solved using the Matrix Riccati Equation:

(8)

where for the case of a regulator

(9)

Solving for with the base spacecraft system and

(10)

yields

(11)

This gives a value of:

(12)

**Task 3:**

A linear quadratic controller with an end time of 30 seconds is developed for the base spacecraft system. For a controller, the cost function to be minimized is:

The control law, gain matrix, Riccati Matrix, and Matrix Riccati Equation are the same between controllers and regulators, and are given by equations (6) through (8), respectively. However, in the case of a controller, the derivative of the Riccati Matrix is not defined to be zero. Thus, to find the Riccati Matrix, which changes with time, the Matrix Riccati Equation is integrated backwards with initial condition:

(13)

For this system, the initial condition for integration is defined:

The values of each component of the Riccati Matrix are shown as functions of time in Figure 1. Importantly, these values are calculated at discrete time steps as there is no analytical solution to the Matrix Riccati Equation for a controller. The values are shown as continuous functions for ease of viewing. Note that because the Riccati Matrix is symmetric across the main diagonal, only three components are shown.

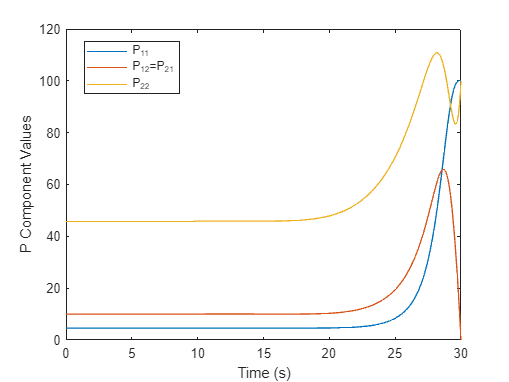


Figure 1. Riccati Matrix Components vs. Time

The gain matrix, is calculated real-time as the Simulink model runs according to equation (7).

The code supporting all calculations described in this section can be found in Appendix A, as well as on this GitHub Page:

<https://github.com/Connor-Lemons/Advanced-Controls-Labs/tree/main/Lab%202>

**Theoretical Predictions**

The goal of Task 2 was to implement a linear quadratic regulator for the base spacecraft system based on given weighting matrices. The purpose of this regulator was to keep the spacecraft within an acceptable deviation from a reference condition using acceptable amounts of control. In other words, the regulator should keep the spacecraft stable in a desired configuration (i.e., the states are within accepted values and are not changing) indefinitely. What determined the acceptable reference condition(s) and acceptable amounts of control were the weighting matrices, and .

Because a regulator functions indefinitely, it does not have an associated stop time. A consequence of this is that the Riccati Matrix, and thus the gain matrix used to determine the control, are constant with respect to time. This means that a solution to the Matrix Riccati Equation can be found analytically, and the gain matrix is easily found as a function of the Riccati Matrix.

For the regulated system, the states were expected to deviate from the initial conditions as a result of the step input being applied to the spacecraft, and then for the states to level off as the regulator worked to keep the spacecraft under control. The expected control input of the regulator was a relatively large initial value which approached a constant value as the states stopped changing.

The goal of Task 3 was to implement a linear quadratic controller for the base spacecraft system based on the same weighting matrices as were used for the linear quadratic regulator and a weighting matrix related to the final states. Unlike the regulator, the purpose of this controller was to drive the states of the spacecraft to zero in the allotted time (30 seconds). The cost function of the controller used the same and matrices as the regulator for similar purposes, but also took into account the matrix. This matrix determined how aggressively the controller attempted to reach the final states.

A controller, unlike a regulator, has a definite stop time and thus both the Riccati Matrix and the gain matrix are functions of time. Thus, the Matrix Riccati Equation cannot be solved analytically for a controller and a different technique is required. Because the end value of the Riccati Matrix is known to equal , a common way to find the Riccati Matrix for controllers is to integrate the Matrix Riccati Equation backwards using as in initial condition.

For the controlled system, the states were expected to deviate from the initial conditions as a result of the step input applied, and then for the states to be driven to zero (or close to zero, depending on the accuracy of the backwards integration) by the controller. Determining the expected control input without implementing the controller was almost impossible. The Riccati Matrix components were expected to initially remain constant, then to exhibit some change in behavior, ultimately taking on the values which would make the Riccati Matrix equal to the weighting matrix for the final states. Importantly, only three components will be plotted as the Riccati Matrix is known to be symmetric about the main diagonal.

**Experimental Results/Discussion**

Though not strictly necessary for the design of the regulator or the controller, or for the validation of either, modeling the base spacecraft system in response to the expected step input gives an interesting result and shows the need for some form of control. As a note, the initial condition used for the entirety of this lab is and the raw input is a unit step input. This is shown in Figure 2.

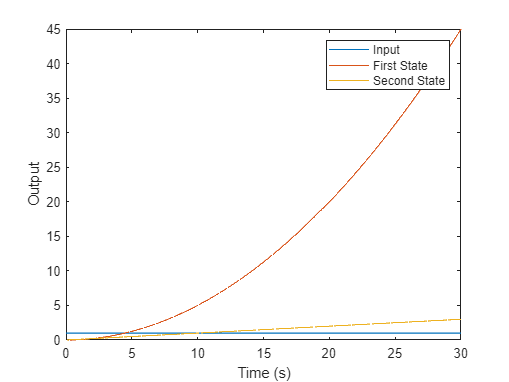


Figure 2. Base Spacecraft Response to Unit Step Input

Implementing the linear quadratic regulator shows a significant improvement over the base spacecraft system. The response of the spacecraft with the regulator given the same step input is shown in Figure 3.

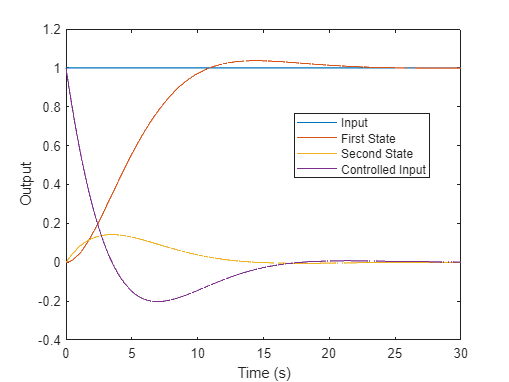


Figure 3. Spacecraft with Regulator Response to Unit Step Input

As predicted, the states deviate initially from the given initial condition as a result of the step input before settling to constant values as a result of the regulator. The controlled input (as a result of the regulator) initially had a relatively large value to get the spacecraft under control quickly before dipping negative—presumably to correct the small amount of overshoot seen in the first state—and then approaching a constant value like the states. The gain used for this regulator was determined by the process outlined in the Mathematical Techniques section, and had a value of:

(12)

The stop time of 30 seconds was chosen because it showed the eventual constant behavior of the states and the controlled input while not being incredibly computationally expensive.

In order to validate the Simulink model and the method for finding the gain matrix, plots of the states were generated using MatLab’s lqr(), step(), and feedback() functions. Each state plot given by Simulink was overlayed on the corresponding state plot generated using MatLab’s functions which can be seen in Figures 4 and 5.

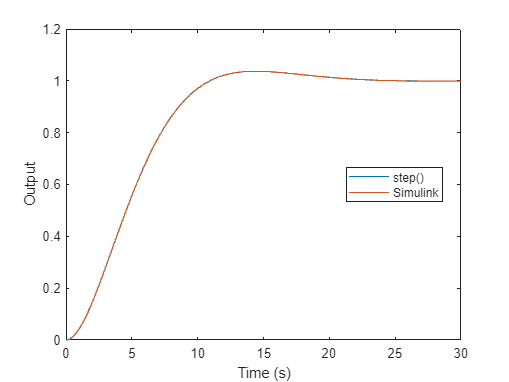


Figure 4. MatLab State and Simulink State 1 Overlay

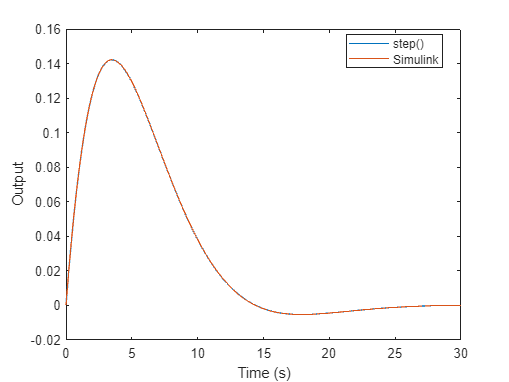


Figure 5. MatLab State and Simulink State 2 Overlay

These Figures clearly show that the Simulink model and the method of calculating the gain matrix described in the Mathematical Techniques section work very well in designing a linear quadratic regulator.

Implementing the linear quadratic controller not only kept the states in control, but actively sought to drive them back to zero. The response of the spacecraft with the controller given the same step input is shown in Figure 6.

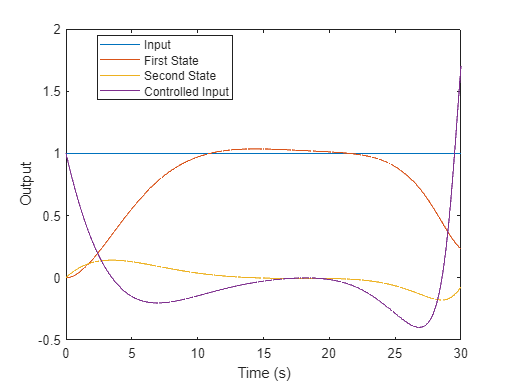


Figure 6. Spacecraft with Controller Response to Unit Step Input

Note that in comparison to the regulator, which held the states constant, the controller actively drives the states to zero. However, the cost of this is that the controlled input is much more varied and takes on more extreme values. This could put additional stress on the actuators used to control the spacecraft, though this can be tailored by changing the weighting functions.

Although the controller was designed to drive the states to zero in 30 seconds, the states do not actually achieve zero by the end of the time frame. This is likely due to a fundamental flaw in the backwards integration process used to determine the Riccati Matrix where integrating backwards only produces an approximation of the Riccati Matrix and is not an analytical solution. As a matter of fact, if the initial Riccati Matrix was used as and future Riccati Matrices were determined by using the full Matrix Riccati Equation at each time step, the final Riccati Matrix, which should be equal to , would be close but not exactly . In order to prevent this problem, the Riccati Matrices are stored during the backwards integration process and then used as the Riccati Matrices when the simulation is run. This ensures that the final Riccati Matrix has the correct value, but it also means that the Riccati Matrices, and thus the gain matrix, are not quite in line with the equations that govern them (i.e., the Matrix Riccati Equation). This can lead to errors such as the states not quite reaching zero by the end of the time frame. The graph of the components of the Riccati Matrices used is shown in Figure 1. Note that there are only three components, which are expressed as functions of time, because the Riccati Matrix is symmetric.

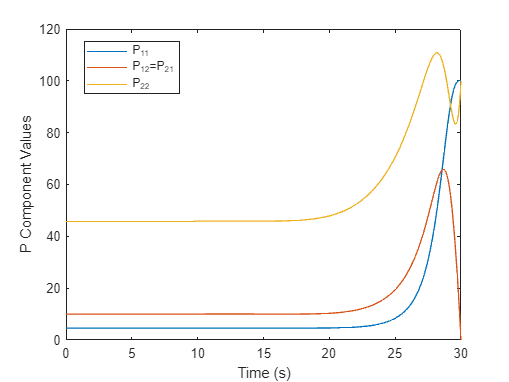


Figure 1. Riccati Matrix Components vs. Time

The code supporting all calculations described in this section can be found in Appendix A, as well as on this GitHub Page:

<https://github.com/Connor-Lemons/Advanced-Controls-Labs/tree/main/Lab%202>

The Simulink models used to generate these response graphs can be found in Appendix B.

**Conclusions and Recommendations**

This project aims to design and evaluate a linear quadratic (LQ) regulator and controller for a single-axis spacecraft attitude control system using optimal control theory. The regulator stabilizes the spacecraft in a desired configuration indefinitely, while the controller drives the spacecraft states to zero within a specified time frame (30 seconds). Using the spacecraft's state-space model, the linear quadratic regulator minimizes a steady-state cost function, yielding a constant feedback gain of . In contrast, because the Riccati Matrix associated with linear quadratic controller is a function of time, the Matrix Riccati Equation is integrated backward, using as the initial condition to ensure that the final value of the Riccati Matrix complies with the weighting function.

Simulation results validate the designs. For the regulator, the system stabilizes with steady-state values after an initial transient, while control inputs converge to consistent levels. Improvements to these results could include refining the backward integration technique for the Riccati equation to reduce numerical errors. Additionally, alternative weighting strategies to and could optimize controller performance for different mission requirements. Beyond spacecraft control, this methodology applies to systems requiring optimal stabilization or trajectory tracking, such as robotic manipulators or automotive cruise control systems.

These findings highlight the flexibility and effectiveness of linear quadratic feedback strategies. The regulator provides a robust, steady-state solution for maintaining stability, while the controller offers precise, time-constrained state optimization.

Appendix A: MATLAB

Appendix B: Simulink Models