KTH Royal Institute of Technology

EL2700 - Model Predictive Control

Assignment 3: Linear Quadratic and Gaussian Regulator

Division of Decision and Control Systems
School of Electrical Engineering and Computer Science

Authors:
Jacopo Dallafior,Stefano Tonini

September 16, 2024

Contents

Chapter 1

Design Task

1.1 Question 1: Stabilizability

The LQR (Linear Quadratic Regulator) problem aims to minimize a quadratic cost function where Q is the state weighting matrix (positive definite), and R is the control weighting matrix (also positive definite). These matrices must be positive definite to avoid unbounded behavior in the cost function.

For a solution to the infinite horizon LQR problem with a finite cost, the system must be stabilizable. Stabilizability ensures that the system's unstable modes, characterized by eigenvalues outside the unit circle, can be controlled. If the system is not stabilizable, the controller cannot correct these unstable modes, causing the state $x(t)$ to grow unbounded.

This unbounded growth in the state would result in an infinite cost, since the term $x(t)^T Qx(t)$ in the cost function J would continuously increase. Therefore, without stabilizability, the cost function would diverge, and a solution with a finite cost would not exist.

Stabilaizability is essential for the existence of a finite solution. Without it, the system cannot be controlled, leading to unbounded state trajectories and an infinite cost.
Since reachability implies stabilaizability, we want to prove reachability of the system, with the PBH test:

The system (A, B) is unreachable if and only if there exists $\lambda \in \mathbb{C}$ and $w \in \mathbb{C}^n$ with $w \neq 0$ such that

$$w^T A = \lambda w^T \quad \text{and} \quad w^T B = 0$$

To prove reachability, we need to show that no non-zero vector w exists such that $w^T A = \lambda w^T$ and $w^T B = 0$ for any eigenvalue λ. We have a system that is reachable since our matrix is:

$$\begin{bmatrix} 0 & 0 & 0 & 0.652 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.701 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.616 \\ 0.01 & -0.001 & 0.001 & 0 & 0 & 0 \\ 0.001 & 0.01 & -0.001 & 0 & 0 & 0 \\ -0.001 & 0.001 & 0.01 & 0 & 0 & 0 \\ -0.01 & 0.001 & -0.001 & 0 & 0 & 0 \\ -0.001 & -0.01 & 0.001 & 0 & 0 & 0 \\ 0.001 & -0.001 & -0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.65 & -0.005 & -0.047 \\ 0 & 0 & 0 & 0 & -0.698 & 0.054 \\ 0 & 0 & 0 & 0 & -0.061 & -0.614 \end{bmatrix}$$

1.2 Question 2: Tuning Parameters

In the first figure (Figure 1.1) we can see the inital plot obtained leaving all the values to one, this will be used as reference to compare it with the next results.

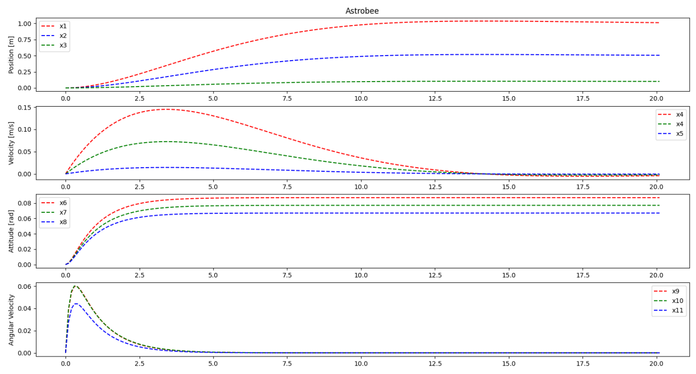
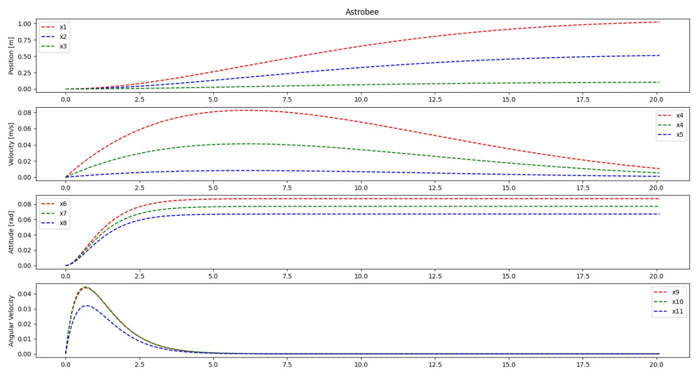
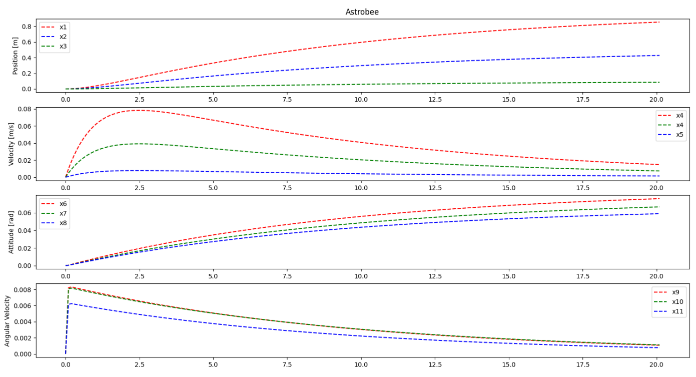


Figure 1.1: Q and R are Identity

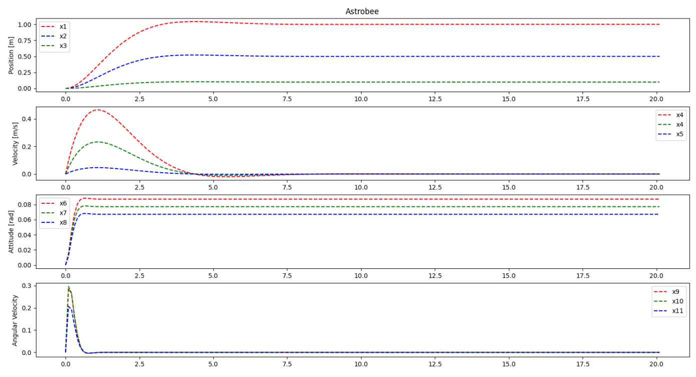
- **Multiply R by 10:** by increasing the value of matrix R, the system will prioritize minimizing energy consumption. This occurs to keep the control cost low. As a result, the reduced control effort is evident in Figure 1.2, where the convergence rate of the position is significantly slower compared to the initial case.



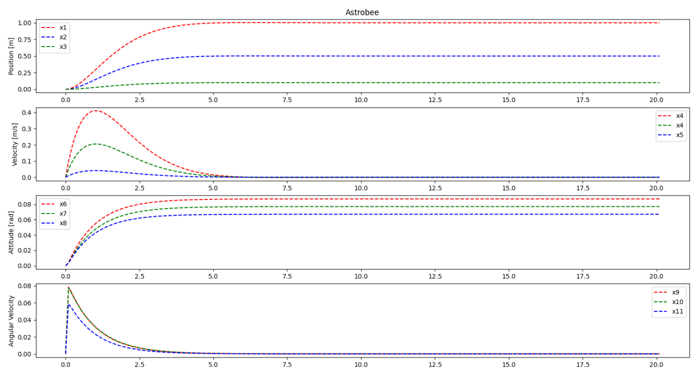
- **Add 100 to the velocity components:** in this second case we have a higher value for the matrix Q, but only for the velocity and the angular velocity. A higher value of Q will results in more precise and faster convergence in velocity, this can be seen by the peak in velocity and angular velocity that occurs at the beginnig of the simulation. Compared to the previous case, the peak reached by the angular velocity is smaller. This happens because the controller is now penalizing angular velocity more heavily, meaning that high angular velocity incurs a higher cost in the control strategy. After the peak the graph become smoother more sharply in order to reach the reference



- **Add 100 to the position and attitude components:** in the third case we have a higher value for the the cost on the position and on the attitude. In this case the reference position and attitude are reached in a faster way with a more aggressive peak consequent to an aggressive control. This can be seen in the following Figure.



- **Increase all elements of Q by 100:** In the final case, we increased all the elements of the Q matrix. As a result, the system reaches its target position more quickly, which aligns with our expectations. By heavily penalizing all state deviations, the controller prioritizes rapid convergence to minimize the cost associated with deviations from the desired state. As in the second case we can see that the peak of the angular velocity is lower compared to the initial case.



1.3 Question 3: Tune the LQR

We were asked to tune the LQR in order to follow this specifics:

1. It moves the astrobee from the initial state $x_0 = 0$ to the reference state
$$p^{\text{ref}} = \begin{pmatrix} 1 \\ 0.5 \\ 0.1 \end{pmatrix}, \quad v^{\text{ref}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \theta^{\text{ref}} = \begin{pmatrix} 0.087 \\ 0.077 \\ 0.067 \end{pmatrix}, \quad \omega^{\text{ref}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix};$$
2. The first three inputs (the forces) may not exceed 0.85 N;
3. The last three inputs (the torques) may not exceed 0.04 Nm;
4. From 12 seconds until the simulation ends (20 seconds):
    - the distance to the reference must be less than 0.06 m;
    - the speed of the astrobee must be less than 0.03 m/s;
    - the Euler angles must differ from their targets by less than 10^{-7} radians;
    - the position overshoot must be less than 2 cm for a safe docking maneuver.

To tune our LQR controller, we initially followed the methodology presented during the class, starting with the application of Bryson's rule to normalize the weights. Specifically, we used the weight normalization approach to ensure comparable deviations in the different signals by setting the elements of the matrices Q and R as:

$$Q_{ii} = \frac{1}{(z_{\max}^{(i)})^2}, \quad R_{ii} = \frac{1}{(u_{\max}^{(i)})^2}.$$

However, despite using these normalized weights, the system was unable to meet the performance targets outlined.

To meet the specified requirements, we decided to deviate from Bryson's rule and introduce additional degrees of freedom using the following formula:

$$\tilde{\mathbf{Q}} = \begin{pmatrix} \left(\frac{q_1}{z_{\max}^{(1)}}\right)^2 & 0 & \cdots & 0 \\ 0 & \left(\frac{q_2}{z_{\max}^{(2)}}\right)^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \left(\frac{q_{n_z}}{z_{\max}^{(n_z)}}\right)^2 \end{pmatrix}, \quad \tilde{\mathbf{R}} = \begin{pmatrix} \left(\frac{r_1}{u_{\max}^{(1)}}\right)^2 & 0 & \cdots & 0 \\ 0 & \left(\frac{r_2}{u_{\max}^{(2)}}\right)^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \left(\frac{r_m}{u_{\max}^{(m)}}\right)^2 \end{pmatrix}.$$

The final value choosen t tune LQR are:

Q = $diag([81, 81, 81, 711.11, 711.11, 711.11, 100, 100, 100, 0, 0, 0])$

R = $diag([111.4, 111.4, 111.4, 625, 625, 625])$

These adjustments allowed us to meet all the specified requirements, yielding the following results:

Parameter	**Value**
Max distance to reference	0.05413619231805008 m
Max speed	0.027464808984232866 m/s
Max forces	
Fx	0.8461688278518482 N
Fy	0.36969240351631716 N
Fz	0.11676788159245619 N
Max torques	
Tx	0.030071835557679307 Nm
Ty	0.029428087172595842 Nm
Tz	0.022571535958294664 Nm
Max Euler angle deviations	
roll	$7.733895895922771e^{-8}$ radians
pitch	$2.914759525970876e^{-8}$ radians
yaw	$8.304131141056992e^{-8}$ radians

Table 1.1: Results obtained from the simulation.

Below is the graph showing the results obtained by running the simulation with the LQR tuned using our values.

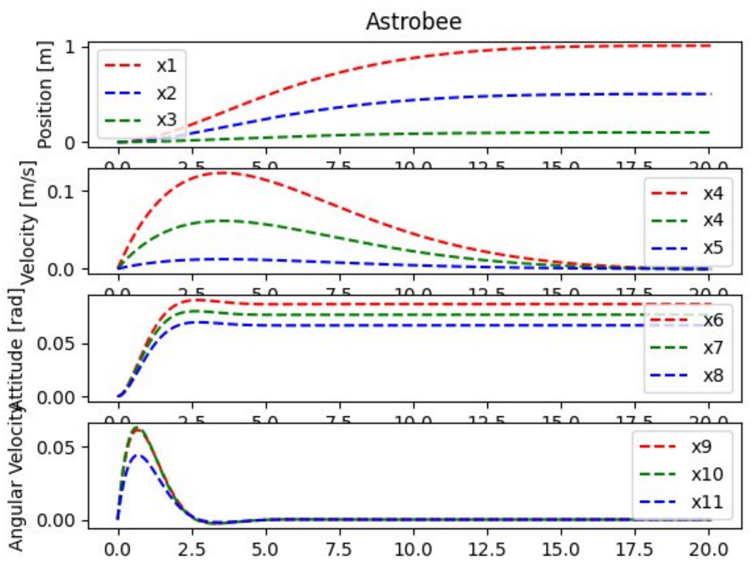


Figure 1.2: Simulation with our tunned parameters

1.4 Question 4: Effect of Tuning Kalman Filter Noise Covariance Matrices

In this question, we consider the effect of different values of the process noise covariance matrix Q_n and the measurement noise covariance matrix R_n on the performance of the controller when implementing an LQG (Linear-Quadratic-Gaussian) controller. To observe more significant changes in system behavior, we adjust the measurement noise within the range [-0.5, 0.5]. This allows us to better see the changes in Astrobee behaviour.

Performance with Different Values of Q_n and R_n:

1. $R_n = 1$ (Measurement Noise) and $Q_n = 1$ (Process Noise):
-this setteings implies that we are assigning equal levels of trust to both the measurement data and the system's predicted state. In practice, this indicates that we believe the measurement and the model's predictive ability to be equally reliable. However, this assumption might not always hold, as systems may have more accurate models or more reliable sensors, necessitating adjustments in these matrices to reflect the uncertainties and improve overall system performance.

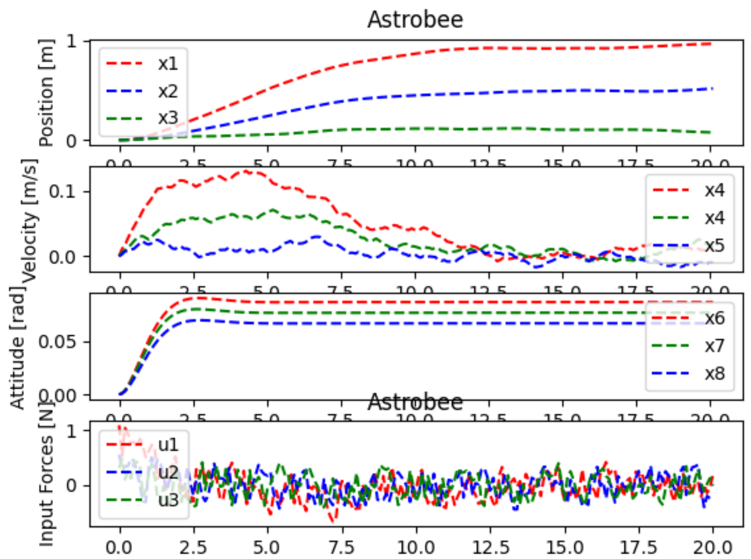


Figure 1.3: Kalman Filter performance with $Q_n = 1 \times I_6$ and $R_n = 1 \times I_3$.

2. Small R_n (Low Measurement Noise): - If R_n is small, the Kalman filter trusts the measurements more, leading to faster response. However, this could cause the state estimates to fluctuate with noisy measurements, impacting control stability.

Large Q_n (High Process Noise): - With large process noise, the Kalman filter assumes the system state is highly unpredictable and reacts quickly to changes. This could lead to noisier control inputs.

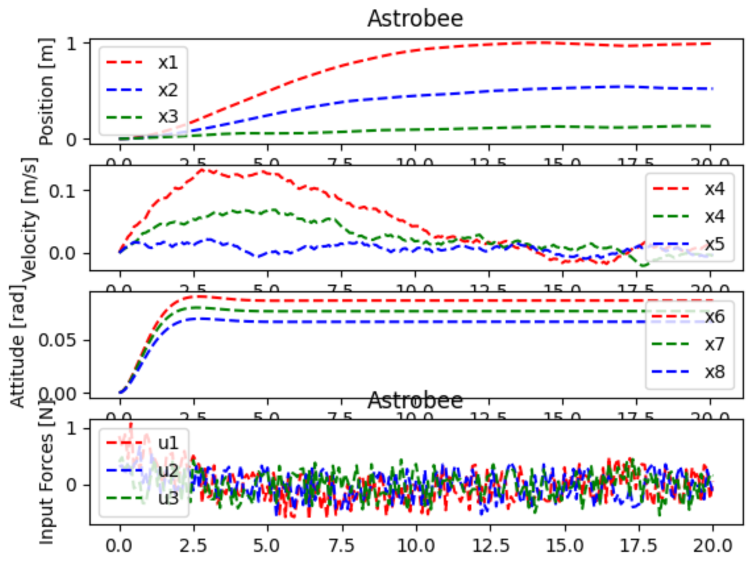


Figure 1.4: Kalman Filter performance with $Q_n = 100 \times I_6$ and $R_n = 0.001 \times I_3$.

3. Large R_n (High Measurement Noise): - When R_n is large, the Kalman filter relies more on model predictions rather than noisy measurements, leading to slower convergence but smoother state estimates. However, if the model is inaccurate, the state estimate may be biased.

Small Q_n (Low Process Noise): - If Q_n is small, the filter assumes the system model is accurate, resulting in smoother control actions. However, the filter may not respond fast enough to real disturbances.

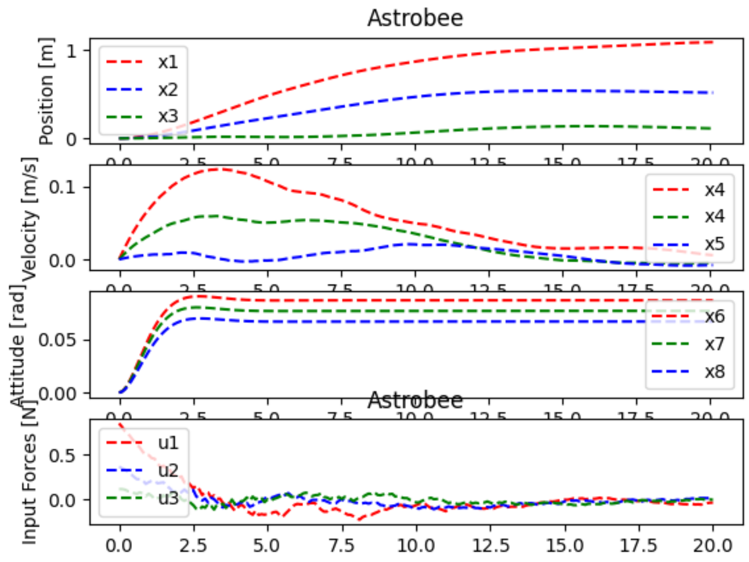


Figure 1.5: Kalman Filter performance with $Q_n = 0.01 \times I_6$ and $R_n = 100 \times I_3$.

In conclusion, tuning Q_n and R_n is crucial for achieving an optimal balance between fast response and noise sensitivity. The choice of these matrices depends on the system dynamics and noise levels.

1.5 Question 5: Controller Performance with Process Noise

In this section the process noise has been increased to [-0.5, 0.5]. The process noise is activated by enabling ‘self.kf_activated = True‘, we observe the performance changes in the following two scenario:

Effect of Process Noise on Control Performance:

1. Equal factors in Qn and Rn:

$$Q_n = diag([1, 1, 1, 0, 0, 0])$$

$$R_n = diag([1, 1, 1])$$

In this first simulation we can see how the filter behave with high measurement noise and process noise. Having that high noise results in our LQR that cannot achive any good performance, and so our system is not able to reach a stability. It can be observed that the position, velocity, and force estimates become noisier and less accurate compared to those in the second simulation, as the process noise starts to affect both the predictions and corrections of the filter. As a result, the system's behavior becomes more variable due to the uncertainty present in both the model and the measurements, making it harder for the Kalman Filter to achieve accurate estimates.

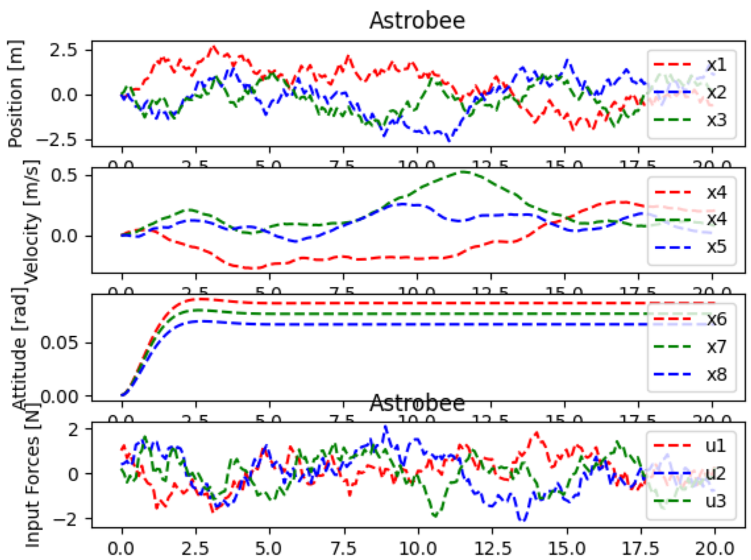


Figure 1.6: Kalman Filter performance with $Q_n = 1 \times I_6$ and $R_n = 1 \times I_3$.

2. No process covariance and large measurement covariance:

$$Q_n = diag([0, 0, 0, 0, 0, 0])$$

$$R_n = diag([12000, 12000, 12000])$$

In this case, the system is simulated with zero process noise covariance, meaning the system model is assumed to be perfect, and a very high measurement noise covariance, indicating that the measurements are unreliable or almost non-existent. In this scenario, the Kalman Filter relies entirely on the mathematical model for predictions, as the measurements are nearly unusable. From the graph, we can observe a better system performance, with more stable and less noisy estimates compared to scenarios where process noise is present. This is due to the complete reliance on the mathematical model, which, in the absence of noise, provides more accurate predictions.

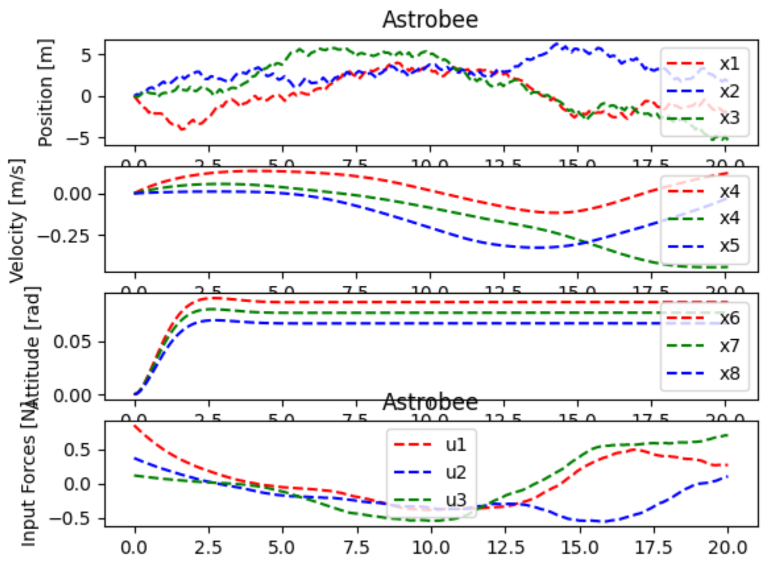


Figure 1.7: Kalman Filter performance with $Q_n = 0 \times I_6$ and $R_n = 12000 \times I_3$.