



# Model Predictive Control - EL2700

## Assignment 3: Linear Quadratic Regulator

Group 7: Christopher Biel, Davide Torrini

## Code Analysis

During inspection of the code we were confused by the implementation of the astrobee dynamics in the function `astrobee.astrobee_dynamics()`. The implementation differs from the dynamics described in the assignment pdf.

Code:

```
wdot = ca.mtimes(ca.inv(self.inertia), tau + ca.mtimes(skew(w), ca.mtimes(self.inertia, w)))
```

which equates to

$$\dot{\omega} = J^{-1}(\mathbf{t} + \omega \times J\omega)$$

Assignment PDF:

$$\dot{\omega} = J^{-1}(\mathbf{t} - \omega \times J\omega)$$

After consulting literature (scripts from other lectures and wikipedia), we believe there is a typo in the code. There should be a subtraction instead of an addition in the brackets. After simulating both versions we found, that it does not qualitatively change the results of the simulation, since the rotational velocities as well as inertia matrix are very low in our case ( $\omega \times J\omega \approx [3.724 * 10^{-5}, -1.691 * 10^{-5}, -2.675 * 10^{-5}]^T$ ). For simulations of different target positions this might not be the case.

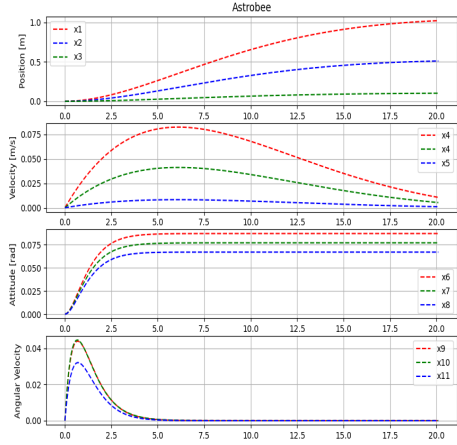
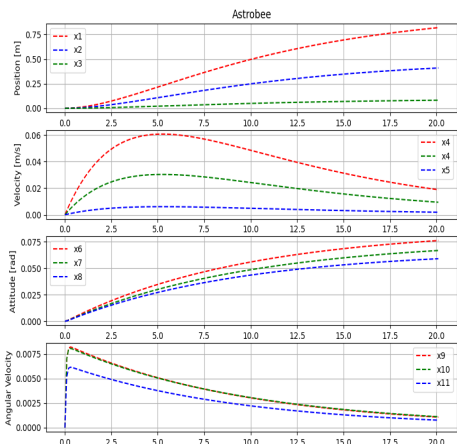
For the sake of comparability (even though the results do not change either way) we kept the current implementation in our code.

## 1 Q1

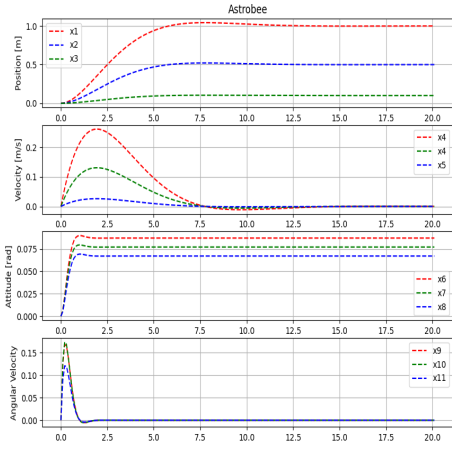
If the system is not stabilizable, the cost for the LQR function would not be finite. This means there is no valid solution for the problem. Since reachability is a sufficient condition for stabilizability (if the system is reachable, then there are no unreachable subsystems - lecture notes, theorem 1.2.4), we can prove that a solution for our LQR problem exists, if the system is reachable. By computing the left eigenvectors  $v_i$  and corresponding eigenvalues  $\lambda_i$  of the continuous, linearized A-matrix  $A_c$  in the linearization point  $\bar{x} = x_{ref}$ , we can evaluate reachability.

For all eigenvalues  $\lambda_i$  and corresponding eigenvectors, the result of  $v_i^T * B_d$  has to be non-zero. This is the case for our system. From this we can conclude that our system is reachable in the linearization point and the LQR problem has a solution.

## 2 Q2

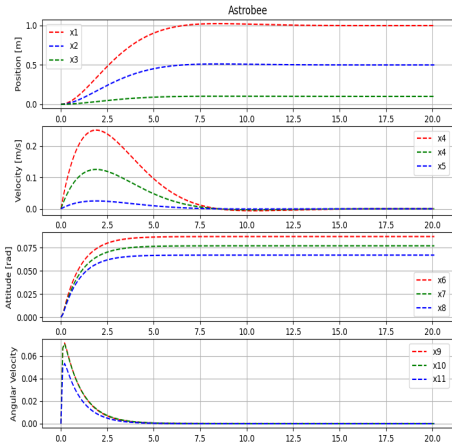
Multiply $R$ by 10	
	<p>By multiplying <math>R</math> by 10, the input must be reduced in order to keep the cost as minimum as possible, as a matter of fact the behaviour of the atrobee is slow and it will slowly reach it's final position. This is reflected in the plot on the left.</p>
Add 100 to $Q[3:6]$ and $Q[9:]$	
	<p>By adding 100 to the velocity and to the rotational velocity of the astrobee, the input has to balance for this penalty, thus it reduces. Indeed the peak angular velocity and the peak velocity happens to be smaller with respect to previous case. Anyhow since we need to reach the correct attitude, the velocity (both angular and translatational one) slowly decreases in order to let the astrobee reach its target position.</p>

Revert the velocity components to 1 and set 100 in  $Q[0:3]$  and  $Q[6:9]$



By increasing the cost on the position and on the attitude, we expect that the reference position and attitude are reached faster with respect to the previous case. This behaviour is reflected in the graph on the left, where also a small overshoot happens to be present. Due to this fact it is possible to conclude that this is not an ideal choice for the  $Q$  matrix.

Increase all elements of  $Q$  by 100



As expected considering the previous conclusion, the overshoot can still be noticed. The astrobee reaches fast its target position but the peak angular velocity is smaller than the previous case since the controller has to account for the penalty of the state.

### 3 Q3

After manually tuning the controller we were able to reach decent performance levels. All required performance targets were achieved, except for the maximum overshoot. To further improve our performance we decided to perform hyperparameter optimization with random search.

The hyperparameters for our LQR Controller are:

$$Q = \text{diag}([34, 78, 283, 171, 179, 161, 48, 48, 48, 3, 3, 3])$$

$$R = \text{diag}([49, 90, 31, 191, 191, 191])$$

With these hyperparameters we achieve the following performance metrics (KPI - Key Performance Indicators):

KPI - Our LQR design	
Max distance to reference:	0.0235 m
Max speed:	0.0238 $\frac{m}{s}$
Max position overshoot:	$[0.0194, 0.0151, 0.0009]^T$ m
Max forces:	$[0.8364, 0.4270, 0.3105]^T$ N
Max torques:	$[0.0369, 0.0361, 0.0278]^T$ Nm
Max Euler angle deviation:	$[2.82 * 10^{-9}, 2.01 * 10^{-9}, 2.80 * 10^{-9}]^T$ rad

Analyzing the values of the Q and R diagonals, we find that the value for the x-position cost is significantly lower than the value for the y- and z-position cost. The values for the velocity cost is also much higher than those of the position cost in general ( $Q[3:6] > Q[0:3]$ ). This allows the positional overshoot to be very low while still allowing a minimal distance to reference.

The explanation for this behaviour is, that the time optimal response to a step always includes an overshoot. So increasing the value for the x-position cost results in an increase in the positional overshoot in x-direction. We can prohibit that by increase the cost for the velocity in x-direction, which penalizes high velocities and thus reduces the overshoot behaviour of the x-position.

We can observe this behaviour especially for the x-direction, since the reference value for x is the highest.

The simulation results of the LQR controller can be seen in fig. 1 and fig. 2.

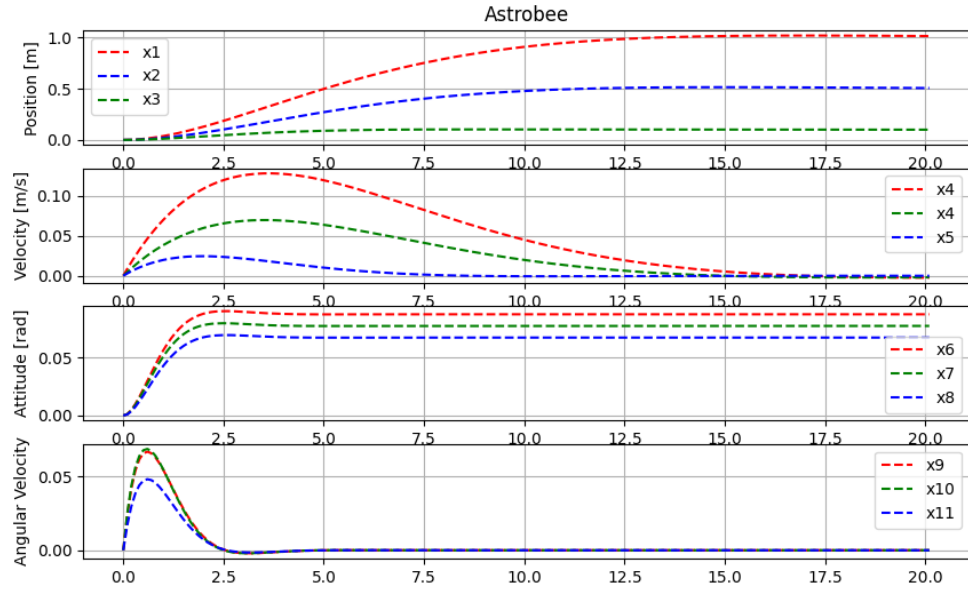


Figure 1: State for the astrobee with our LQR controller. All the performance metrics are achieved.

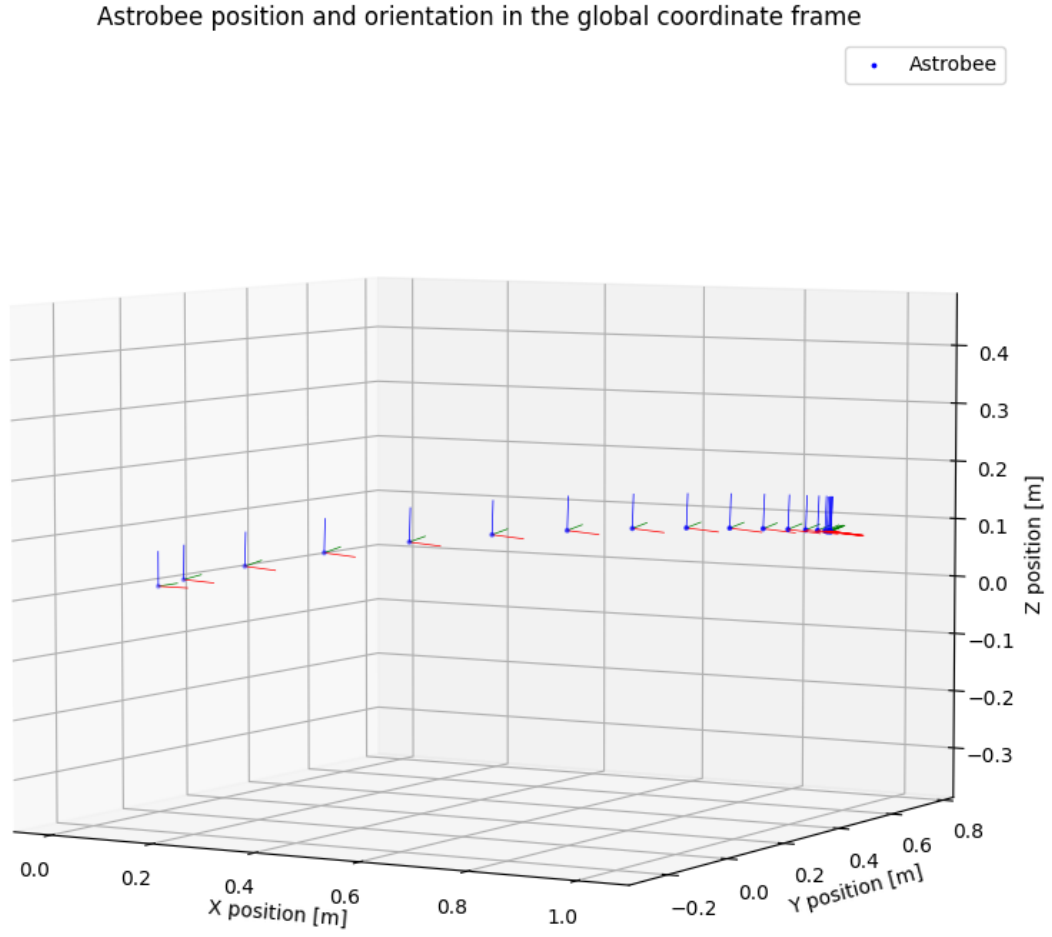


Figure 2: Trajectory and Orientation for the astrobee with our LQR controller from Q3, sampled every 1s. We can see, that the astrobee slightly tilts to the target orientation in the first few timesteps.

## 4 Q4

To see qualitative changes in the behaviour of our astrobee, we changed the minimum and maximum of the uniformly distributed measurement noise to  $v_t \in [-0.1, 0.1]$ .

For the first experiment we change the covariance matrices to:

$$Q_n = \text{diag}([100, 100, 100, 0, 0, 0])$$

$$R_n = \text{diag}([0.001, 0.001, 0.001])$$

This results in the behaviour seen in fig. 3. The large values in the  $Q_n$  matrix result in a large Kalman gain  $\mathbf{K}_k$  which leads to the result being more heavily influence by the measurement than the model prediction. The estimated state is impacted more heavily by the measurement noise and the resulting control inputs are also noisy. Interestingly enough, the performance of the controller nearly achieves the requirements, only failing in x- and y-overshoot and maximum force in x-direction by a small margin.

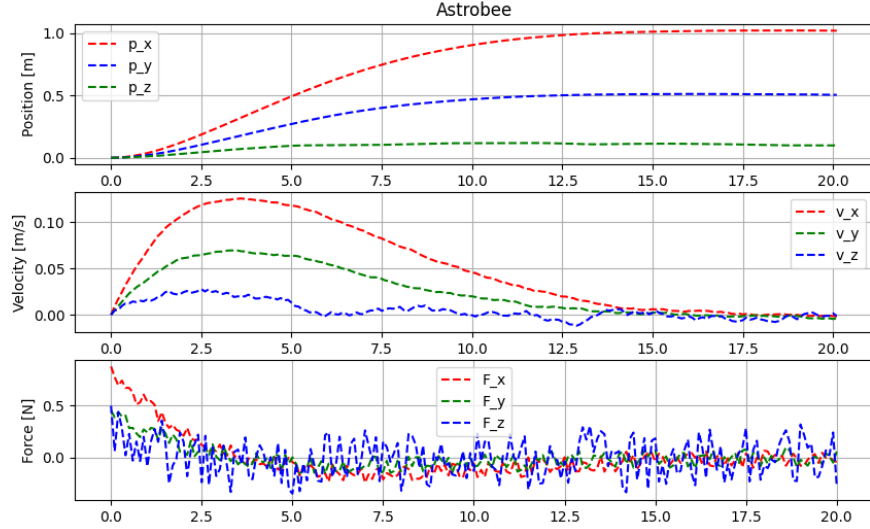


Figure 3: High  $Q_n$  matrix values lead to a large influence of the measurement noise on the state estimation. The actions are very noisy.

As an extreme counter-example, we change the covariance matrices to:

$$Q_n = \text{diag}([0.001, 0.001, 0.001, 0, 0, 0])$$

$$R_n = \text{diag}([100, 100, 100])$$

Now the model prediction has a greatly increased influence on the state estimation, while the influence of the measurement is decreased. Thus the introduced measurement noise has less of an effect on the performed actions (see fig. 4). Subsequently the controller performs better.



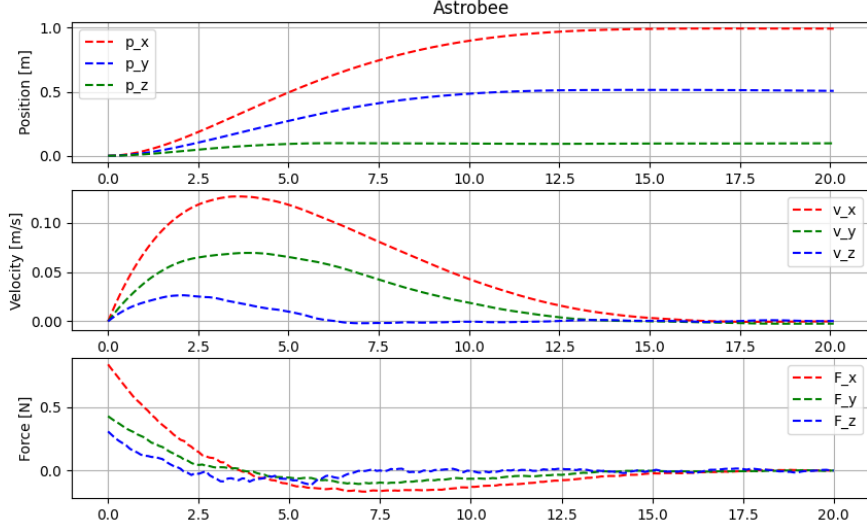


Figure 4: Low  $Q_n$  matrix values lead to a small influence of the measurement noise on the state estimation. The actions are less noisy.

## 5 Q5

Again, to see qualitative changes in the behaviour of our astrobee, we changed the minimum and maximum of the uniformly distributed process noise to  $w_t \in [-0.1, 0.1]$  and keep the measurement noise at  $v_t \in [-0.1, 0.1]$ .

For this configuration we can best see the effects with the following covariance matrices:

1. No process covariance and large measurement covariance

$$Q_n = \text{diag}([0, 0, 0, 0, 0, 0])$$

$$R_n = \text{diag}([100000, 100000, 100000])$$

2. Equal factors in  $Q_n$  and  $R_n$

$$Q_n = \text{diag}([1, 1, 1, 0, 0, 0])$$

$$R_n = \text{diag}([1, 1, 1])$$

The different behaviour is displayed in fig. 5 and fig. 6. Since the prediction step of the Kalman Filter calculates with the mean of the standard distribution, it performs the prediction identical to the true model without process noise. If there were no measurement input (which we try to show by setting the measurement covariance to a high value) the estimated position and velocity would be equal to the system behaviour without process noise. The performed actions are also the same. In reality however the Kalman Filter completely mispredicts the behaviour of the true system

(which now contains process noise) and only when adding back the measurement by reducing the measurement covariance, does the Kalman Filter react to the influence of the process noise. For this simulation, the process and measurement noise are too high to achieve any reasonable performance with our LQR controller.

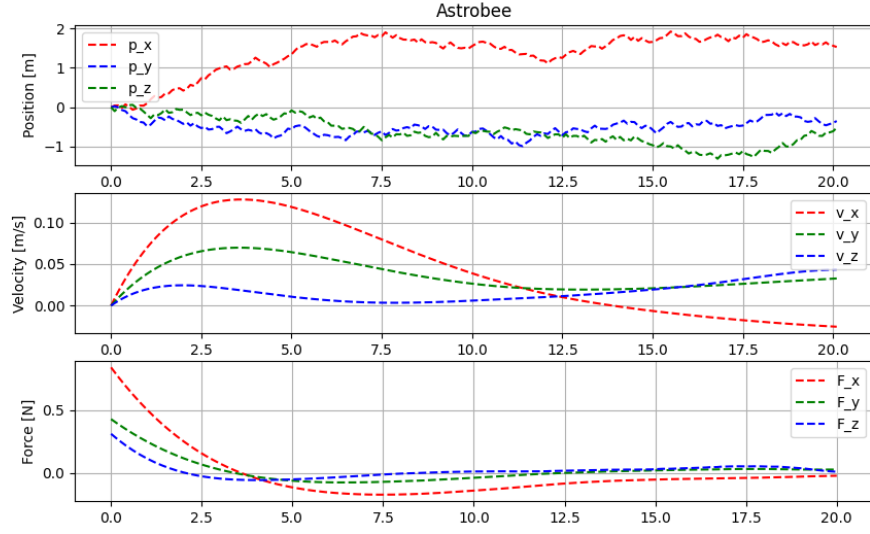


Figure 5: No process covariance and large measurement covariance. The Kalman Filter estimates mainly from the model prediction.

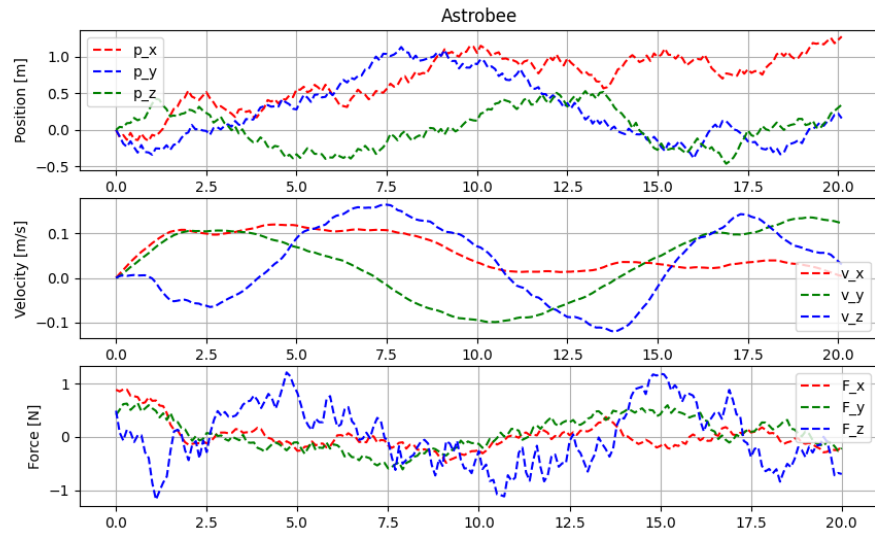


Figure 6: Equal factors in  $Q_n$  and  $R_n$ . The Kalman Filter reacts to the noise in the process.