Model Predictive Control - EL2700

Assignment 4: Model Predictive Control

Group 7: Christopher Biel, Davide Torrini

1 Q1 - Influence of Control Horizon

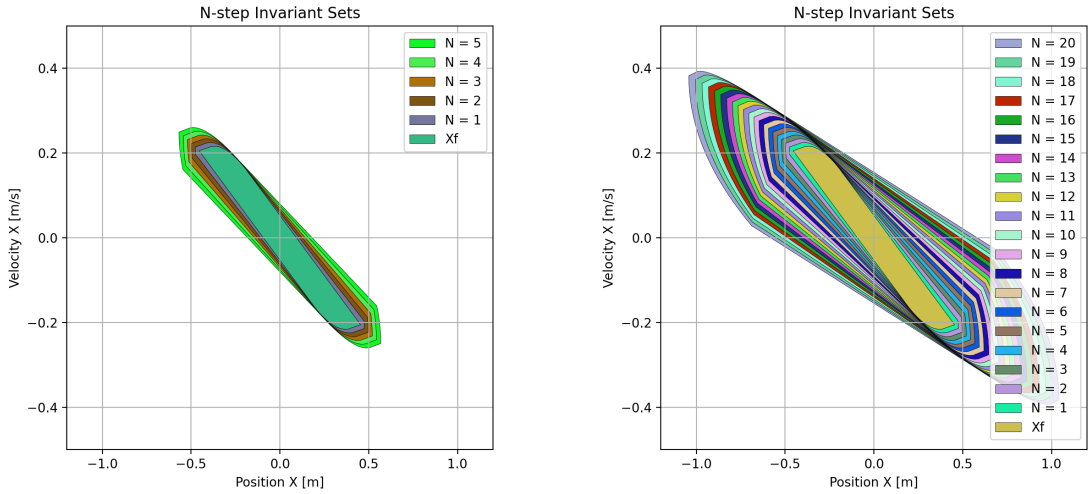
As expected, increasing the control horizon N allows for larger set of admissible states $x_{t|t} \in X_t$. This results from the simple fact, that given the control authority of our system, for an increased horizon N of the MPC controller we are able to reach the terminal set $x_{t+N|t} \in X_f$ (the control-invariant set we chose) from a larger set of initial states. This is displayed in fig. 1.
To increase the information content of the plots, we chose to plot the sets for p_x and v_x instead of p_x and p_y by changing the code in line 224 of `set_operations.py` from

```
Kni.project([1, 2]).plot(ax=ax, alpha=1, linestyle="-", linewidth=0.3)
```

to

```
Kni.project([1, 4]).plot(ax=ax, alpha=1, linestyle="-", linewidth=0.3)
```

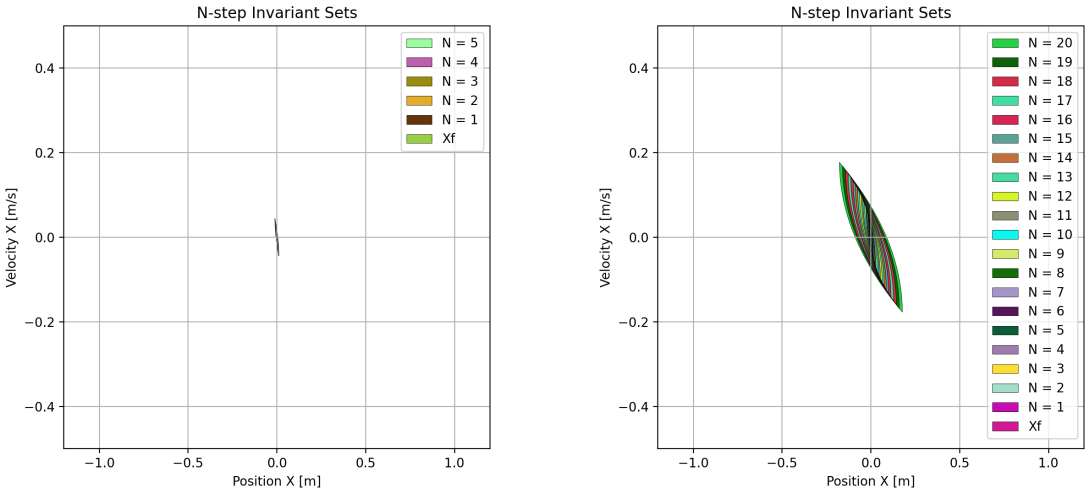
In comparison to p_x and p_y, the combination of states p_x and v_x is dependant on each other, which creates a much more interesting shape of the sets. Also, the terminal values of p_y are always constrained by the state constraint $-0.1 \leq p_y \leq 0.1$ and thus the values for p_y do not change with different horizons.



(a) $N = 5$ (b) $N = 20$

Figure 1: The set of initial states X_0 increases with larger horizons N.

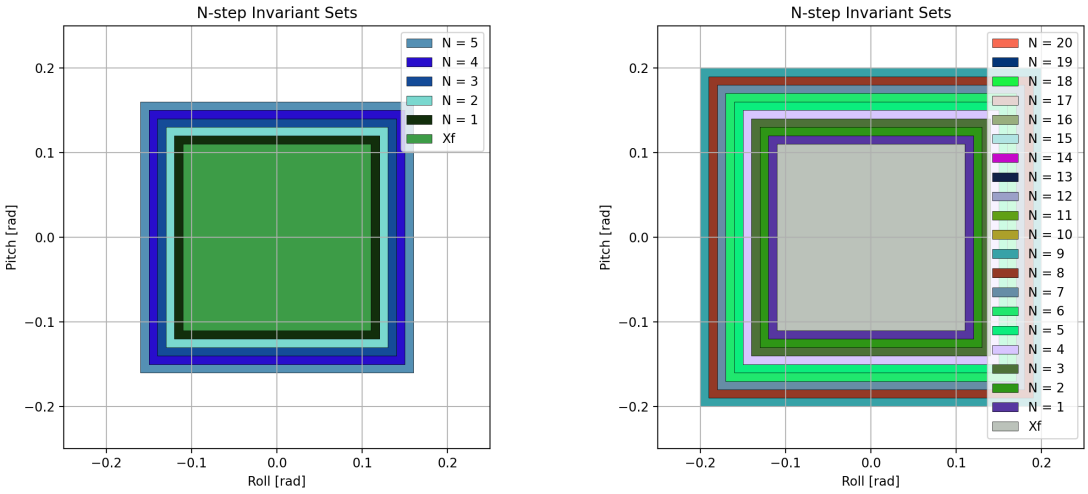
If the terminal set is $X_f = \{0\}$ the same holds true, as can be seen in fig. 2. In general however the set of initial states X_t is much smaller, since the terminal set is also smaller.



(a) $N = 5$ (b) $N = 20$

Figure 2: The set of initial states X_0 increases with larger horizons N.

For the 'attitude' plots, the behaviour is as expected, with the set increasing until the state boundaries for the plotted states are reached (fig. 3).

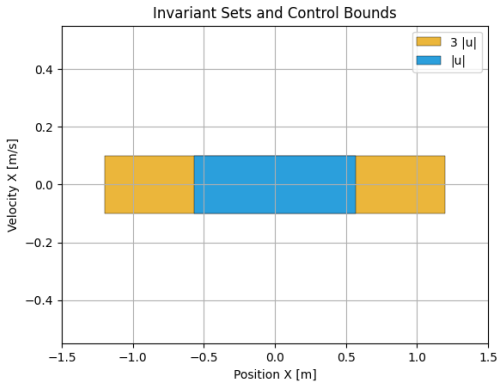


(a) $N = 5$ (b) $N = 20$

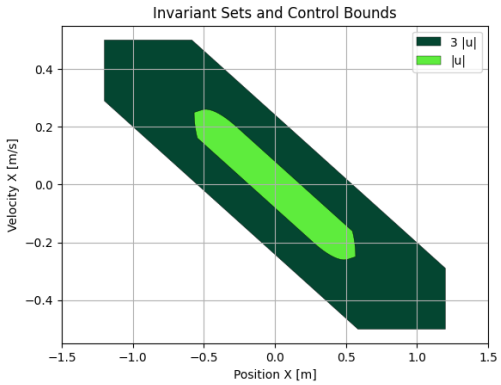
Figure 3: The set of initial states X_0 increases with larger horizons N up to $N = 9$, then the maximum value for the plotted states is reached.

2 Q2 - Influence of Control Constraints

By increasing the control constraints and by keeping $N = 5$ steps it is possible to see that the astrobbe would be able to reach it's target position by starting from far away with respect to where it should start by keeping the initial input value. This is due to the fact that increasing the input means increasing the force the astrobee is subjected to, thus it would be able to reach the target state faster.



(a) Input influence on X position and Y position



(b) Input influence on X position and X velocity

Figure 4: Input influence on position and velocity.

3 Q3 - Examine the MPC controller implementation and explain using your own words

3.1 How the state constrains are set

```
77 if xub is None:
78     xub = np.full((self.Nx), np.inf)
79 if xlb is None:
80     xlb = np.full((self.Nx), -np.inf)
81 if uub is None:
82     uub = np.full((self.Nu), np.inf)
83 if ulb is None:
84     ulb = np.full((self.Nu), -np.inf)
```

```
122 # State constraints
123 if xub is not None:
124     con_ineq.append(x_t)
125     con_ineq_ub.append(xub)
126     con_ineq_lb.append(np.full((self.Nx,), -ca.inf))
127 if xlb is not None:
128     con_ineq.append(x_t)
129     con_ineq_ub.append(np.full((self.Nx,), ca.inf))
130     con_ineq_lb.append(xlb)
```

For each time-step of x the inequalities are created by setting the upper and lower bound for the state vector respectively at line 125 and line 130. It is worth to point out that the upper bound is firstly set to plus infinite at line 78 by creating a vector with same size of the state vector.

3.2 How the object function is formed

```
142 # Objective Function / Cost Function
143 obj += self.running_cost((x_t - x0_ref), self.Q, u_t, self.R)
144 # Terminal Cost
145 obj += self.terminal_cost(opt_var['x', self.Nt] - x0_ref, self.P)
```

```
206 # Instantiate function
207 self.running_cost = ca.Function('Jstage', [x, Q, u, R],
208                                          [x.T @ Q @ x + u.T @ R @ u])
209
210 self.terminal_cost = ca.Function('Jtogo', [x, P],
211                                           [x.T @ P @ x])
```

The object function is formed at line 143 by summing up at each step the running cost that is obtained at line 206 for each step. The same happens for the terminal cost at line 172 and line 180. The MPC cost function is defined as eq.6a shows.

$$\min_{u_{t+\cdot|t}} \sum_{k=0}^{N-1} \mathbf{x}_{t+k|t}^T Q \mathbf{x}_{t+k|t} + \mathbf{u}_{t+k|t}^T R \mathbf{u}_{t+k|t} + \mathbf{x}_{t+N|t}^T P \mathbf{x}_{t+N|t} \tag{6a}$$

3.3 How the terminal constraint is set

```
148 # Terminal contraint
149 if terminal_constraint is not None:
150     # Should be a polytope
151     H_N = terminal_constraint.A
152     if H_N.shape[1] != self.Nx:
153         print("Terminal constraint with invalid dimensions.")
154         exit()
155     H_b = terminal_constraint.b
156     con_ineq.append(H_N @ opt_var['x', self.Nt])
157     con_ineq_lb.append(-ca.inf * ca.DM.ones(H_N.shape[0], 1))
158     con_ineq_ub.append(H_b)
```

The terminal constraint is set with the use of the polytope calculated earlier on. In our case this is either $X_f = 0$ or X_f^{LQR} and it is passed into the MPC class as `terminal_constraint`. We use the inequalities which define the polytope and pass them into our inequalities list. This is done by extracting the H_N and H_b matrices of the polytope. The inequality is defined similar to equation (4) of the Assignment_4.pdf:

$$X_f = \{H_N x \leq H_b\}$$

3.4 What the variable `param_s` holds

```
86 # Starting state parameters - add slack here
87 x0 = ca.MX.sym('x0', self.Nx)
88 x0_ref = ca.MX.sym('x0_ref', self.Nx)
89 u0 = ca.MX.sym('u0', self.Nu)
90 param_s = ca.vertcat(x0, x0_ref, u0)
```

The variable `param_s` is defined at line 90 and holds a vertical vector filled with symbols who represents x_0, x_{ref} and u_0. These variables will then be filled with the values for each MPC iteration.

3.5 Why we only select the first element

For MPC we recompute the problem at each timestep t. This means, that the full MPC problem over the finite-horizon is solved, but then only the first value of the calculated inputs

$$\{\hat{u}^\star_{t|t}, \hat{u}^\star_{t+1|t}, \hat{u}^\star_{t+2|t}, ..., \hat{u}^\star_{t+N|t}\}$$

is applied to the system. For the next timestep the input sequence is recomputed with the new value for $x_{t+1|t+1}$ and results in

$$\{\hat{u}^\star_{t+1|t+1}, \hat{u}^\star_{t+2|t+1}, ..., \hat{u}^\star_{t+1+N|t+1}\}$$

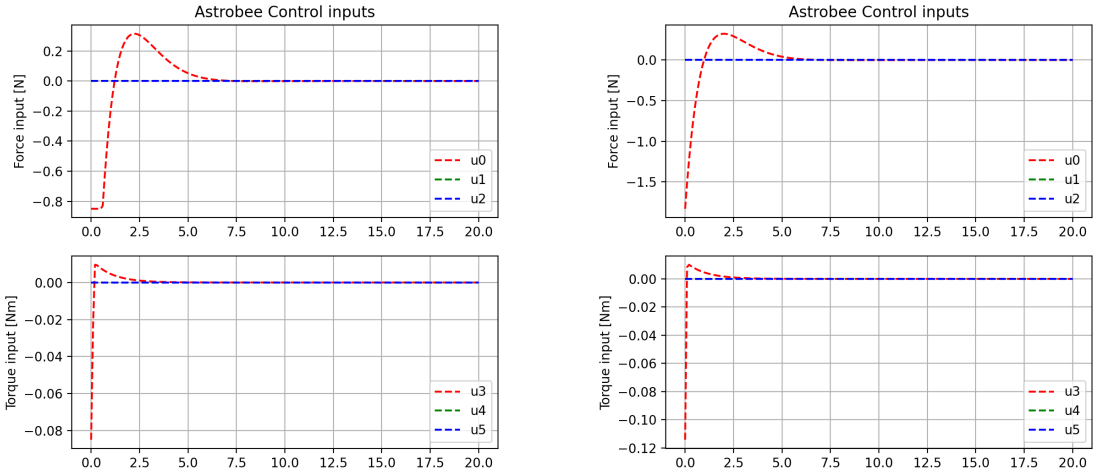
and again the first value is applied to the system. This method allows the controller to react to deviations of the state from the expected behaviour, which can happen due to model inaccuracies, noise or disturbances.

4 Q4

We compare (a) MPC with standard control bounds against (b) MPC with triple the control bounds.

	(a)	(b)
Used Energy	156.2	160.5
Position Integral Error	3.94	3.56
Attitude Integral Error	0.90	0.88

The increase in energy from (a) to (b) is marginal and explained with a larger maximum control input (as seen in 5). This also leads to a higher velocity in translation and rotation and thus to a faster convergence towards the target state. The position and attitude integral errors are lower for (b) than for (a). An increase in the control bounds allows for better performance while only marginally penalizing the used energy. It is thus always better to have larger control bounds (if those are not accompanied by other limitations, e.g. higher mass of the system).



(a) MPC with standard configuration

(b) MPC with increased control bounds

Figure 5: The increase in control bounds allows for a higher force and torque input. The system reaches its target faster.

5 Q5

Running the MPC with a terminal set $X_f \in \{0\}$ results in an infeasible problem: The controller can not calculate a solution for the optimization problem. When we recall the observations of 2 it becomes obvious why that is the case: with a starting point of $x_0 = [0.20^5 0.080^5]^T$ we start outside the set of feasible starting states: $x_0 \notin X_0$.

6 Q6

By increasing the horizon to $N = 50$ the starting set X_0 is big enough to include our chosen x_0 and we can find a feasible solution to the optimization problem. Since we now compute a much larger problem the time to solve increase drastically: While the inital problem with $N = 10$ and X_f^{LQR} took approx. 0.015 s to solve per iteration, the current parameters of $N = 50$ and $X_f = \{0\}$ result in a time of approx. 0.11 s per iteration.

7 Q7

We compare the following configurations of the MPC with X_f^{LQR} and $N = 10$

(a) `Q = np.eye(12)` and `R = np.eye(6)`

(b) `Q = np.eye(12)` and `R = np.eye(6)*10`

(c) `Q = np.diag([1, 1, 1, 101, 101, 101, 1, 1, 1, 101, 101, 101]])` and `R = np.eye(6)*10`

(d) `Q = np.diag([101, 101, 101, 1, 1, 1, 101, 101, 101, 1, 1, 1]])` and `R = np.eye(6)*10`

(e) `Q = np.diag([201, 201, 201, 101, 101, 101, 201, 201, 201, 101, 101, 101]])` and `R = np.eye(6)*10`

The results of the comparison are displayed in the table below and fig. 6.

	(a)	(b)	(c)	(d)	(e)
Used Energy	58.688	30.889	19.844	108.350	122.159
Position Integral Error	10.074	16.844	22.082	5.705	4.875
Attitude Integral Error	0.961	1.165	7.045	0.455	0.691

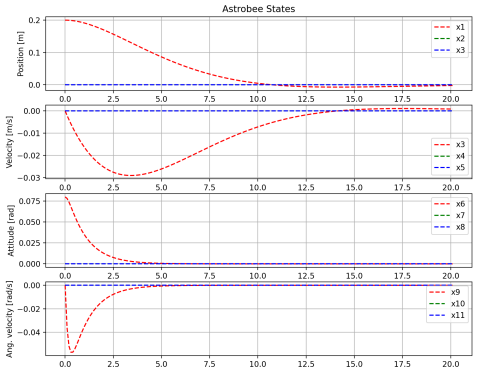
Increasing the values of Q penalizes high control inputs, which leads to a reduction in used energy from (a) to (b). Also the states converge slower (see fig. 6a vs 6b), the position and attitude integral errors increase. When the weights for the velocities increase, the control inputs are again reduced.

Configuration (c) thus has an ever lower used energy, with higher position and attitude integral errors. A significant change in behaviour can be seen in fig 6c, where the angular velocity has a lower, but more constant deviation. This results form the quadratic cost on the actions penalizing high absolute values.

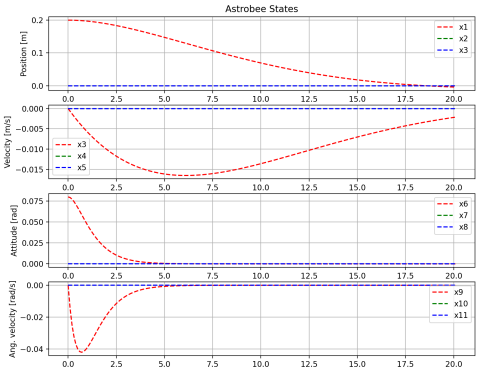
When observing configuration (d), it is obvious that an increase in cost for the position and attitude states should and in fact does result in a lower position and attitude integral error. The states converge faster at the cost of more used energy.

By increasing the costs for all states with configuration (e), this effect is slightly increased, since now the cost of the actions $u^t Ru$ has less of an effect. The used energy increases again, while the position integral error is decreased. The slight increase in the attitude integral error is an unexpected result. This behaviour probably stems from the change in relative weight of attitude and angular velocity, which now results in a lower peak of angular velocity. The same is not as significant for the position and velocity, since the velocity curve is not as sharp / pointy as that of the angular velocity.

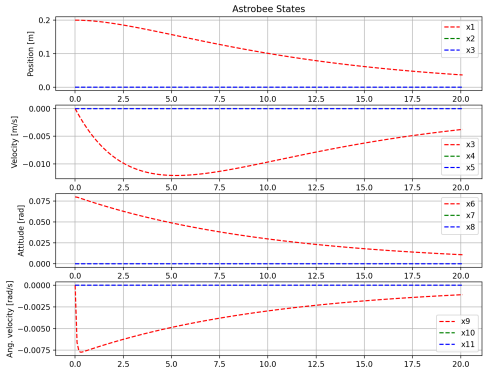
The observed behaviour of the MPC matches what we expected of the different configurations and what we observed when analyzing the behaviour of the LQR approach.



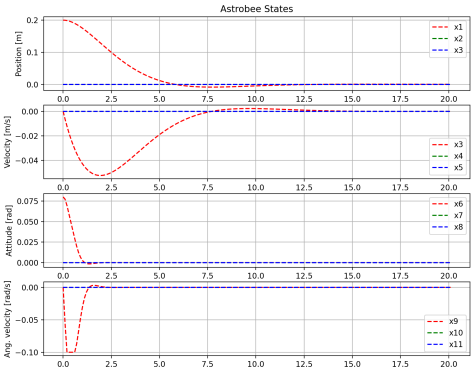
(a)



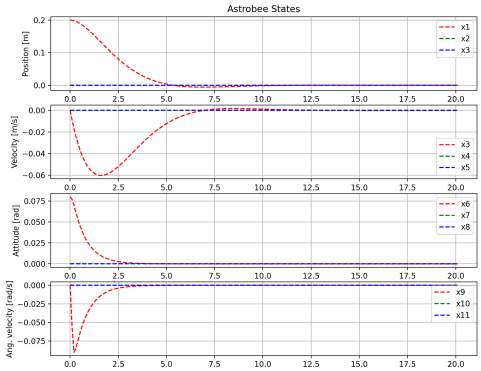
(b)



(c)



(d)



(e)

Figure 6: Effects of changes to Q and R on the states of the Astrobee.