KTH Royal Institute of Technology

EL2700 - Model Predictive Control

Assignment 2: Finite-Time Optimal Control

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

Authors:
Jacopo Dallafior,Stefano Tonini

September 10, 2024

Contents

Chapter 1

Design Task

In this assignment, we will simulate rendezvous maneuvers using the Astrobee robots. Honey will be the debris, and Bumble will be our cleaning robot.

We will implement a finite-time optimal control algorithm using Python and the solver MOSEK together with CVXPY. The objective is to approach Honey optimally with Bumble and match its momentum for cleaning.

1.1 Question 1: Implement the Model

The linearized system model is provided in the assignment. We implemented the continuous-time model using the provided dynamics in the function ‘cartesian_ground_dynamics‘ of the Astrobee class. The linearized continuous-time model is given as:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

Where the matrices A and B are:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1/I_z \end{bmatrix}$$

where $m = 20.9\,\mathrm{kg}$ and $I_z = 0.2517\,\mathrm{kg\,m^2}$.

1.2 Question 2: Discrete-Time Dynamics

We used the ‘c2d‘ method to convert the continuous-time model to discrete-time dynamics. The discrete-time system matrices A_d and B_d are given by:

$$A_d = \begin{bmatrix} 1 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B_d = \begin{bmatrix} 2.3923 \times 10^{-4} & 0 & 0 \\ 0 & 2.3923 \times 10^{-4} & 0 \\ 4.7847 \times 10^{-3} & 0 & 0 \\ 0 & 4.7847 \times 10^{-3} & 0 \\ 0 & 0 & 2.0 \times 10^{-2} \\ 0 & 0 & 4.0 \times 10^{-1} \end{bmatrix}$$

These matrices were implemented in the 'set_discrete_dynamics' method in the Astrobee class. They define the dynamics of the Bumble robot in discrete time, enabling us to solve the finite-time optimal control problem.

The matrix A_d represents the state transition dynamics, while B_d describes how the control inputs affect the state in the discrete-time model.

$$x_{\mathrm{t}+1} = A_d \cdot x_{\mathrm{t}} + B_d \cdot u_{\mathrm{t}}$$

These matrices will be used in the optimization problem in the following tasks.

1.3 Question 3: Rendezvous Constraints

The following equation represents the cost function $J(u(t))$, where the control input $u(t)$ is optimized over a time horizon. The summation term in the cost function,

$$J(u(t)) = \sum_{t=0}^{t_f} u(t)^T R u(t),$$

indicates that the total cost is accumulated by the weighted control input over time, where R is a diagonal, positive definite weighting matrix that determines the importance of the control effort. The problem also introduces input constraints, where the upper and lower bounds of the control input are given by

$$\bar{u} = \begin{bmatrix} 0.85 \\ 0.42 \\ 0.04 \end{bmatrix} \quad \text{and} \quad \underline{u} = \begin{bmatrix} -0.85 \\ -0.42 \\ -0.04 \end{bmatrix},$$

respectively. These constraints ensure that the control input remains within specific limits during the optimization process.

The finite-time optimal control problem includes also constraints on position and attitude to ensure Bumble approaches Honey safely and accurately. In the solver initialization method, we added the following constraints to the list *con_ineq* to impose the necessary bounds for position and attitude. The constraints ensure that the position and attitude errors stay within the specified tolerances to ensure safe and accurate rendezvous.

The following Python code defines these constraints:

```python
con_ineq.append((x_ref[0:2] - x_t[0:2]) <= self.pos_tol)
con_ineq.append((x_ref[0:2] - x_t[0:2]) >= -self.pos_tol)

con_ineq.append((x_ref[2] - x_t[4]) <= self.att_tol)
con_ineq.append((x_ref[2] - x_t[4]) >= -self.att_tol)
```

- The first two constraints bound the position error in the X and Y axes to stay within *self.pos_tol*.

- The last two constraints bound the attitude error (rotation around the Z-axis) to stay within *self.att_tol*

These constraints are added to the list *con_ineq*, which holds the inequality constraints of the problem. The values *x_ref* and *x_t* represent the reference state and the current state, respectively.

1.4 Question 4: Solve the Optimization Problem

We solved the finite-time optimization problem using the given trajectory from Honey. The reference trajectory was extracted for the last 5 seconds, and Bumble tracked this trajectory to complete the rendezvous maneuver. The optimal state and control inputs, $x^\star$ and $u^\star$, were obtained from the solver and used to simulate the system.

Figure 1.1 shows the tracking error in the position (x1 and x2) and attitude (x5) over time, along with the control inputs (u1, u2, u3) applied to the system. As seen in the figure, the tracking error decreases over time as Bumble approaches the desired trajectory for Honey, successfully completing the rendezvous maneuver.

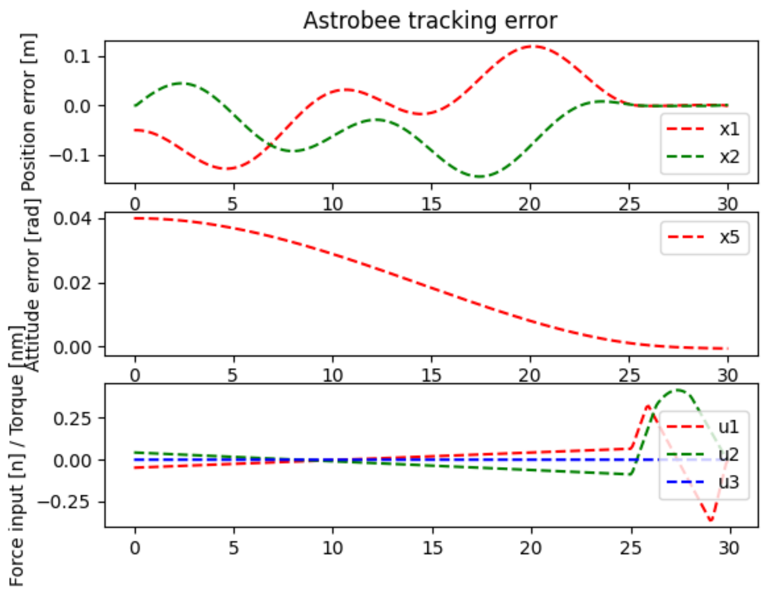

Figure 1.1: Astrobee tracking error and control inputs for the initial simulation.

In the Figure 1.2, we observe a comparison between the predicted state, obtained through optimization, and the reference trajectory (represented by dashed lines). The graph illustrate the rendezvous process between Bumble and Honey.

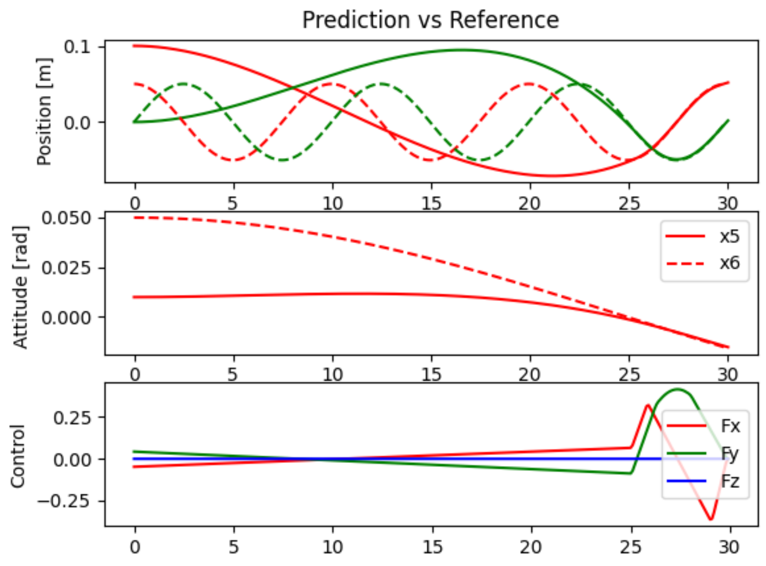

Figure 1.2: Prediction vs. Reference for the second simulation.

The first subplot shows the predicted positions (solid lines) compared to the reference positions (dashed lines). The red and green lines represent different position components. During the initial stages, the predicted positions deviate noticeably from the reference.

However, as we approach the last 5 seconds of the simulation, the solid and dashed lines begin to converge, indicating that Bumble is closely following the reference trajectory as it prepares to rendezvous with Honey.
The second subplot depicts the attitude of the system, showing how Bumble's orientation changes over time. The solid red line represents the predicted attitude, while the dashed red line is the reference attitude. Similar to the position plot, we see that in the final moments of the simulation, Bumble's predicted attitude closely matches the reference.
The control forces applied during the rendezvous process. The red, green, and blue lines correspond to forces in the x, y, and z directions, respectively. Notice the sudden changes in control inputs around the 25-second mark, where Bumble applies corrective forces to align its position and attitude with the reference.

1.5 Question 5: Broken Thruster Behavior

When the broken thruster simulation is activated using *sim_env.broken_thruster()*, we observed that the generated control inputs are no longer able to maintain the desired trajectory in the rendezvous manouvre, causing a position error in the y-axis diffrent from zero. This is likely due to the erratic force output from the broken thruster, which causes Bumble to deviate from the optimal trajectory.

In the figure (Figure 1.3), we simulated the behavior of the system when one of Bumble's thrusters malfunctioned. The figure shows the incorrectly behavior of the control input u2 which results in the position error for state x2 not converging to zero during the rendezvous maneuver. The erratic behaviour of only one state is due to the fact that the dynamic is decoupled and so the other state remain unchanged.

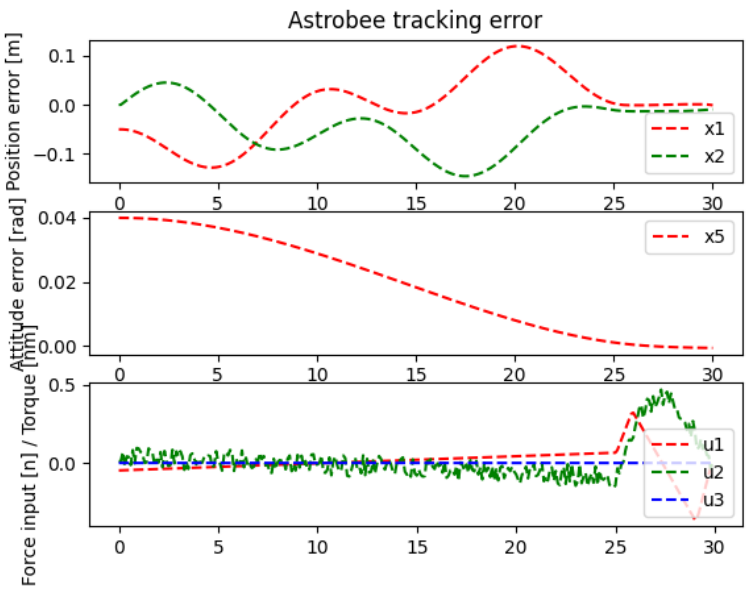

Figure 1.3: Astrobee tracking error and control inputs with a broken thruster.

In this case we cannot really appriciate the diffrence, but every simulation give us diffrents result (as can be seen in Figure 1.4) this is due to the fact that the broken thruster have randomic noise added after the optimization.
This suggests that the final control inputs are no longer optimal, and the system cannot maintain the desired trajectory in the y-direction.

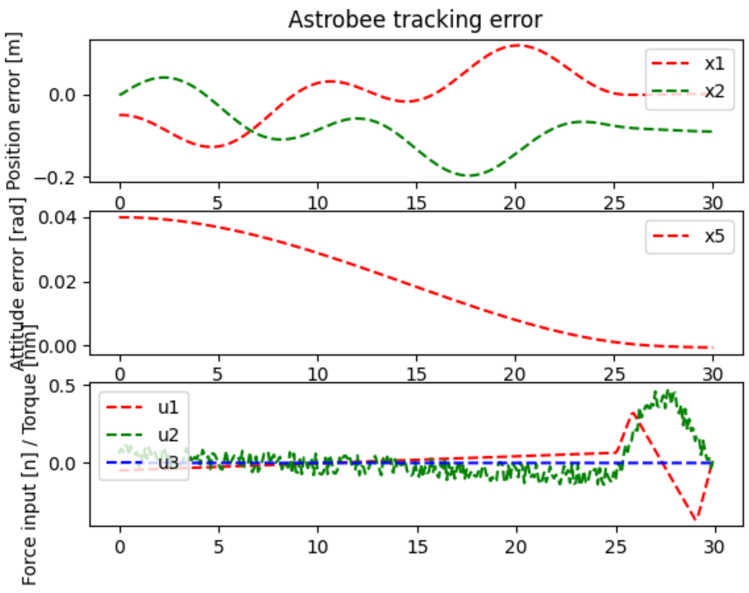

Figure 1.4: Astrobee tracking error

The issue in our case is that the optimization is performed assuming ideal inputs, without accounting for the presence of noise. As a result, the optimizer does not consider the effects of random noise, leading to errors during the rendezvous maneuver.

1.6 Question 6: Improving Performance with a Faulty Thruster

To address the faulty thruster issue, we can improve performance by incorporating a feedback-based approach similar to Model Predictive Control. Currently, our system optimizes the entire trajectory once, treating it as open-loop, without adapting to real-time changes. This approach does not account for disturbances like a malfunctioning thruster.

By dividing the optimization into smaller blocks and updating control inputs based on real-time state feedback, we can re-optimize at each step, continuously adjusting the system to disturbances. This method mirrors MPC, where control inputs are recalculated iteratively, making the system more robust and minimizing errors caused by disturbances.

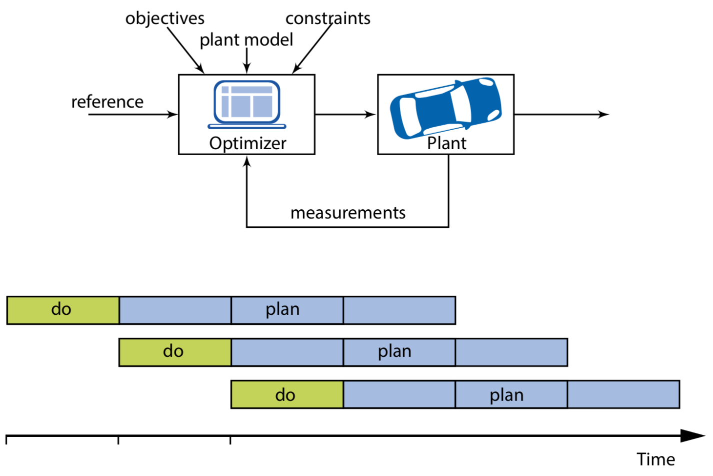

Figure 1.5: Proposed feedback control loop with real-time adjustments for faulty thruster compensation.

By transitioning to an MPC framework, we could improve the system's resilience to disturbances, such as the faulty thruster, by constantly re-optimizing the control sequence based on the actual state of the system.

1.7 Question 7: Extending to Three Dimensions

We extended the model to include translation along the Z-axis. The state vector for the 3D model is now:

$$x = \begin{bmatrix} p_X & p_Y & p_Z & v_X & v_Y & v_Z & \theta & \omega \end{bmatrix}^T$$

The control input vector is given by:

$$u = \begin{bmatrix} f_X & f_Y & f_Z & \tau_Z \end{bmatrix}^T$$

We implemented this model in the 'cartesian_3d_dynamics' function of the Astrobee class. After extending the model, the simulation was run to analyze the system's behavior in 3D space, tracking the desired reference trajectory in the X, Y, and Z directions. Figure

1.6 shows the tracking errors in the position (x1, x2, x3) and attitude (x5) over time, along with the corresponding control inputs (u1, u2, u3, u4) applied to the system.

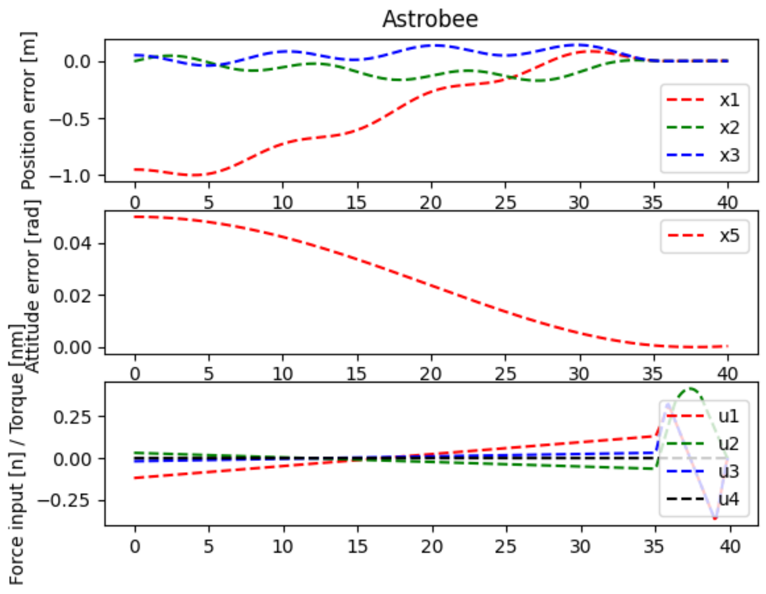

Figure 1.6: Astrobee tracking error and control inputs in the 3D model with Z-axis translation.

As seen in Figure 1.6, the system is able to track the desired trajectory in 3D space, reaching a position errors in the order of millimiters in the position along the X, Y, and Z axes. The attitude tracking also performs well, as the error decreases over time. The control inputs (forces and torque) remain within expected limits, with a noticeable adjustment to the control forces as the system approaches the final position.

The extension to 3D translation allows for more realistic simulation and control of the Astrobee in free space, improving the system's ability to execute complex maneuvers.