Assignment 1

Lorenzo Fici, Hannes Gustafsson, Davide Nascivera

August 2024

1 Introduction to the Design Task

In this assignment, we will focus on the docking maneuver with the docking-station, which the robot uses to upload/download data and charge its batteries. Since this is the first task we are running on our freshly built system, we will focus the movement along one axis of the robot, assuming the robot was previously aligned with its docking port.
For designing the state feedback controller, we will use a linearized model that describes the motion of the Astrobee along one translation axis

$$\dot{x} = A_c x + B_c u$$

where the state $x = \begin{bmatrix} p_X & v_X \end{bmatrix}^T$ concatenates the position on the X-axis, and $u = F_x$ is a force along the body X-axis of the robot. The matrices A_c and B_c are given by

$$A_c = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad B_c = \begin{bmatrix} 0 \\ \frac{1}{m_G} \end{bmatrix}$$

where m_G is the mass of the Astrobee and air-carriage combined. Accordingly:

$$\begin{aligned} \text{Astrobee mass:} \quad & m_A = 9.6\,[\text{kg}] \\ \text{Air-carriage mass:} \quad & m_C = 11.3\,[\text{kg}] \\ \text{Combined ground test mass:} \quad & m_G = m_A + m_C = 20.9\,[\text{kg}] \end{aligned}$$

2 Questions

2.1 Q1

We implemented the model in the function `one_axis_ground_dynamics` as following:

Listing 1: Model implementation

```
def one_axis_ground_dynamics(self):

    Ac = np.zeros((2, 2))
    Bc = np.zeros((2, 1))

    Ac = np.array([[0, 1],[0,0]])
    Bc = np.array([[0],[1/self.mass]])

    self.Ac = Ac
    self.Bc = Bc

    return self.Ac, self.Bc
```

2.2 Q2

We start with the continuous-time system and we want to discretize it obtaining the following discrete-time system:

$$x_{k+1} = Ax_k + Bu_k$$
$$y_k = Cx_k + Du_k$$

where

$$A = e^{A_c h} = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}$$
$$B = \int_0^h e^{A_c s} B_c \, ds = \frac{h}{m_g} \cdot \begin{bmatrix} \frac{h}{2} \\ 1 \end{bmatrix}$$

the constant m_g is provided as well as the sampling time $h = 0.1s$.
We obtain the same numerical result from the simulation.

$$A = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 0.00024 \\ 0.0048 \end{bmatrix}$$

2.3 Q3

Here, we are tasked with computing the transfer function of the continuous-time model from the control input u to the output y, considering $C = [1\ 0]$ and $D = 0$. After Laplace transforming, the general solution for $G_{u\to y}(s)$ is given by

$$G_{u\to y}(s) = \frac{Y(s)}{U(s)} = C(sI - A)^{-1}B + D \tag{1}$$

Inserting our matrices, we obtain

$$
\begin{aligned}
G_{u\to y}(s) &= \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}s & -1\\ 0 & s\end{bmatrix}^{-1}\begin{bmatrix}0\\ \frac{1}{M_G}\end{bmatrix}\\
&= \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\frac{1}{s} & \frac{1}{s^2}\\ 0 & \frac{1}{s}\end{bmatrix}\begin{bmatrix}0\\ \frac{1}{m_G}\end{bmatrix}\\
&= \frac{1}{m_G s^2}
\end{aligned}
$$

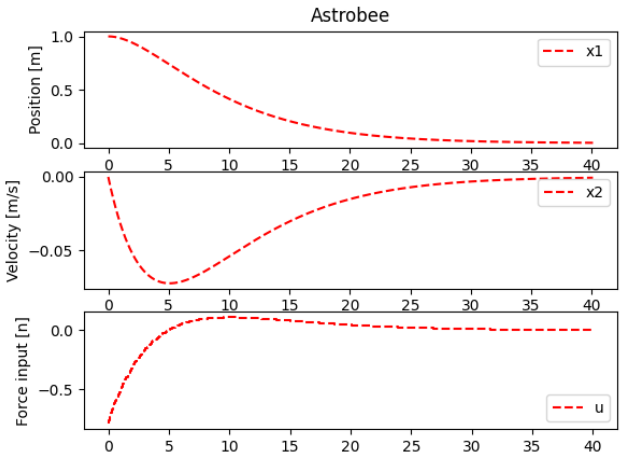
The system has a double pole at $s = 0$, and no zeros. Based on this, it was expected that the discrete system would have two poles, and no zeros as well. However, the intuition was wrong since the discrete system had one zero.

2.4 Q4

Based on the discretized linear model, we derived a state feedback controller denoted as L. The control law is given by:

$$u_k = -L(\mathbf{x}_k - \mathbf{x}_{\text{ref}})$$

The matrix L was computed using the function `place`, with the poles of the matrix $A - BL$ set at 0.98 and 0.981. The results, presented in the figure below, confirm that the system successfully meets the design criteria: reaching 90% of the final value in under 25 seconds while keeping the maximum control input below 0.85 N. Specifically, at 25 seconds the Astrobee's position is 0.043 meters, achieving 96% of the required distance.

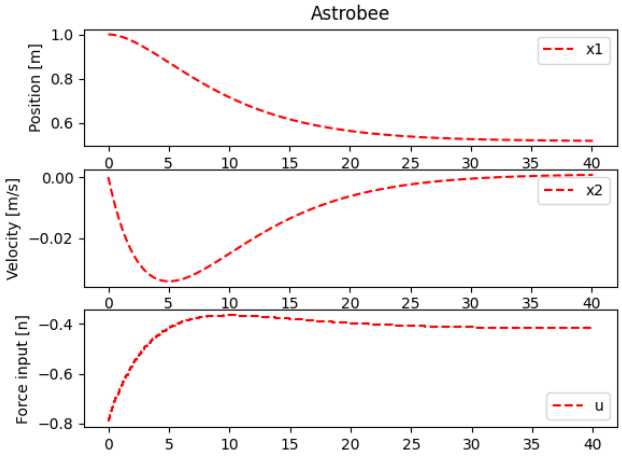


The resulting gain matrix L is given by:

$$L = \begin{bmatrix}0.794 & 8.111\end{bmatrix}$$

2.5 Q5

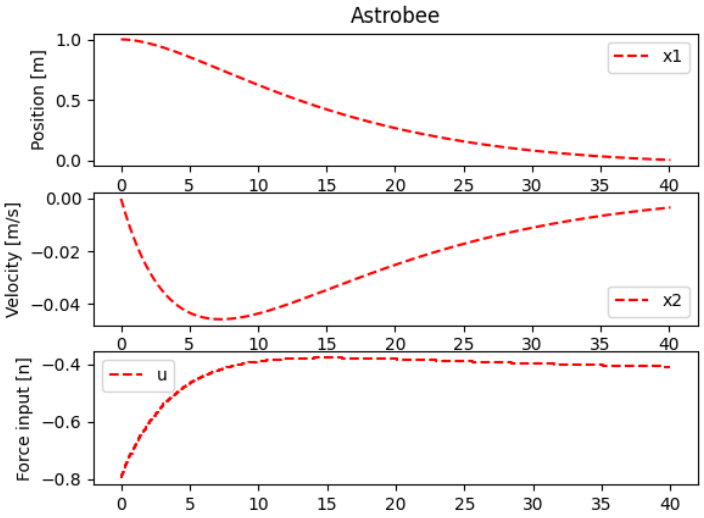
Upon introducing a disturbance into our system, the resulting plot reveals that Astrobee fails to reach the intended target position at (0,0). Instead, it stabilizes prematurely, settling at a final position of 0.518, rather than the desired 0.



To counteract the effect of the disturbance, we need to add an integral component to our controller. The control input then becomes:

$$u_k = -L\mathbf{x}_k - K_i i_k$$
$$i_{k+1} = i_k + h \cdot (p_k - p_{\text{ref}})$$

By including the integral component with $K_i = 0.029$, the system achieves the desired performance. The system meets all specified requirements, specifically Astrobee moves 90% of the distance from the initial position to the reference point within 30 seconds. Specifically, after 30 seconds the Astrobee's position is 0.084 meters, achieving 92% of the required distance.



2.6 Extra

For the extra challenge, we were tasked with extending the model to a 2-axis translation. This was done by considering the state variables p_Y, v_Y and the additional force f_Y, there by obtaining

$$\mathbf{x} = [p_X, v_X, p_Y, v_Y] \quad \mathbf{u} = [f_X, f_Y] \tag{2}$$

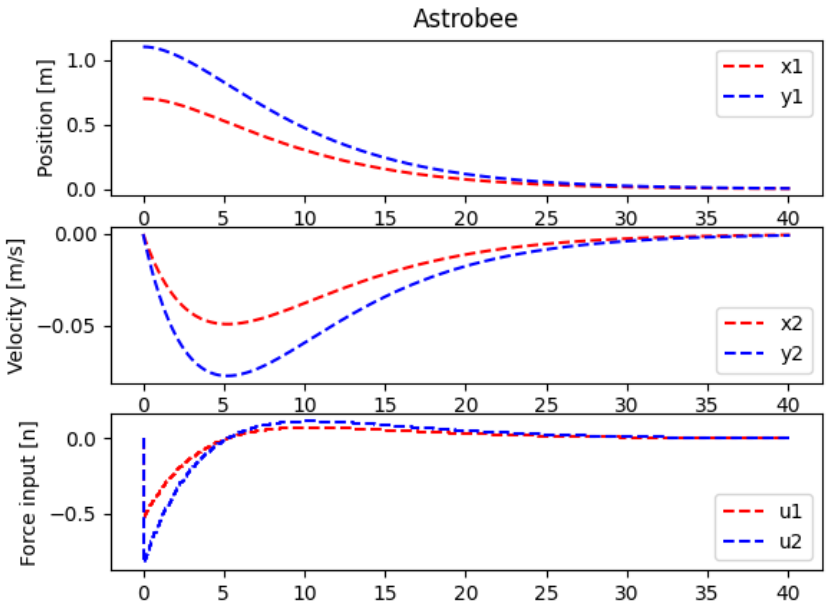
Since the translations in X and Y axis are independent of each other, the new system was merely the block diagonal extension of the system in a 1-axis translation - almost like a tensor product.

$$A_{c,2D} = \begin{bmatrix} A_c & \mathbf{0} \\ \mathbf{0} & A_c \end{bmatrix} \qquad B_{c,2D} = \begin{bmatrix} B_c & \mathbf{0} \\ \mathbf{0} & B_c \end{bmatrix}$$

$$C_{c,2D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad D_{c,2D} = \mathbf{0}$$

where $\mathbf{0}$ denotes a zero matrix of implicits rows and columns. For example, in $A_{c,2D}$, $\mathbf{0}$ is of dimensions $(2, 2)$.

After updating the code to handle this new model, we set $x_0 = [0.7\ 0\ 1.1\ 0]$ and ran the simulation without disturbances. The results are shown in the following figure. This can be generalized for disturbances and the integral component as well.



3 Conclusion

In this first assignment our group managed to design and test a controller for the Astrobee robot focusing on its docking maneuver with the docking-station along a single axis. With a linearized model we met all the performance requirements. We began by discretizing the continuous-time model and verifying the results through simulation. The discrete system's behavior was analyzed, revealing an unexpected zero that differed from the continuous model. The state feedback controller was designed using pole placement making sure to take care at every time of all the constraints. When disturbances were introduced in the model the controller needed an integral action to compensate them. At the end we extended the model to a 2 dimensional space.