EL2700 - Assignment 1: Discrete-time Linear Systems

Group 14: Chieh-Ju Wu and Charles Brinkley

September 7, 2021

1 Q1: Plant Dynamics

The first task as to implement the open-loop, continuous, plant model in the *one axis ground dynamics* function oksf *astrobee 1d*. This is a simple task of essentially copying the provided matrices and implementing with Casadi and Numpy library functions. The completed ODE has the form...

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ v_x \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m_g} \end{bmatrix} F_x \tag{1.1}$$

2 Q2: Discretized System Model

Question 2 asks to analytically discretize the continuous system and then compare it to that numerically generated solution given by the *casadi c2d* function. This is done by sample and hold method, and shown per the following equations (with sampling rate h=.1).

$$A = e^{A_c h} \tag{2.1}$$

$$B = \int_0^h e^{A_c s} B ds \tag{2.2}$$

Because this is a double integral, we take advantage of the Taylor theorem...

$$A = e^{A_c h} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} h + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} h^2 = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \tag{2.3}$$

$$A = \begin{bmatrix} 1 & .1 \\ 0 & 1 \end{bmatrix} \tag{2.4}$$

By knowing the discrete state space matrix, we can now plug this into the discrete B equation...

$$B = \int_0^h e^{A_c s} B ds = \int_0^h \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ \frac{1}{m_g} \end{bmatrix} ds = \begin{bmatrix} \frac{h^2}{2m_g} \\ \frac{h}{m_g} \end{bmatrix} \tag{2.5}$$

$$B = \begin{bmatrix} .00024 \\ .00478 \end{bmatrix} \tag{2.6}$$

By passing the continuous ODE matrices to the *casaadi c2d* function, we take advantage of the Casadi library functions to calculate the discretized model numerically. those solutions are seen below in the screenshot of Figure 1.

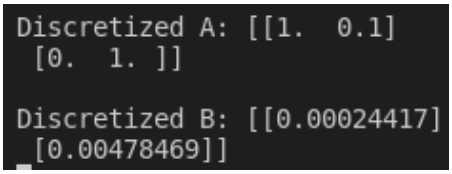

Figure 1: Ad and Bd terminal outputs from *casadi c2d* function

As can be seen, both our analytical solution and that of the numerical Casadi function approach show the same answer. This validates our methodolgy.

3 Q3: Continuous-Time Transfer Function

To determine the transfer function of the continuous time model, the numerical solution is given by equation3.2.

$$G(s) = C(SI - A)^{-1}B + D \tag{3.1}$$

The following output and feed-forward matrices are provided by the task...

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}; D = 0 \tag{3.2}$$

Rather than solving by hand, the *control* library was utilized to call the *ss2tf* transfer function which does the heavy lifting for us. The returned value from this function can be seen in figure 2 below. Results were also verified in MATLAB in a similar manner.

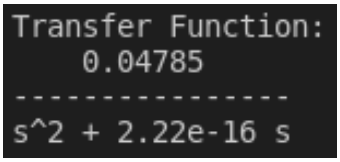

Figure 2: Terminal output from *ss2tf* function for continuous system model

By looking at the transfer function, we can discern that their will be no zeros due to the lack of an s variable in the numerator. The denominator is of second order, so we can expect two poles, but these will just be overlapping zeros given the lack of any roots. This is correctly expressed by the Pole-Zero map generated in figure 3 and describes a system that is unstable since the pole is placed right on the edge of the stability boundary.

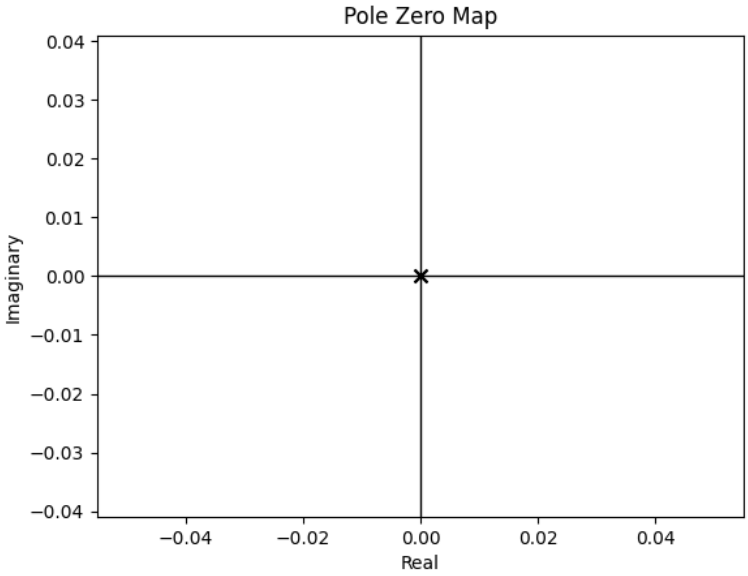

Figure 3: Pole-zero map of the continuous time model

Given that the poles/zeros in continuous time can be represented in discrete time via zero-order hold ($\lambda = e^{-\lambda_c h}$), we can assume that there should be two poles right on the stability boundary at one when discretized when using this method of pole conversion. This was found to be correct given the resulting pole-zero map in figure 4. What was surprising to see was the inclusion of a discrete zero despite no zeros being present on the continuous transfer function. This could be due to the specific method of conversion used by the pole zero function (i.e. ZOH, tustin, euler forward/backward, etc.).

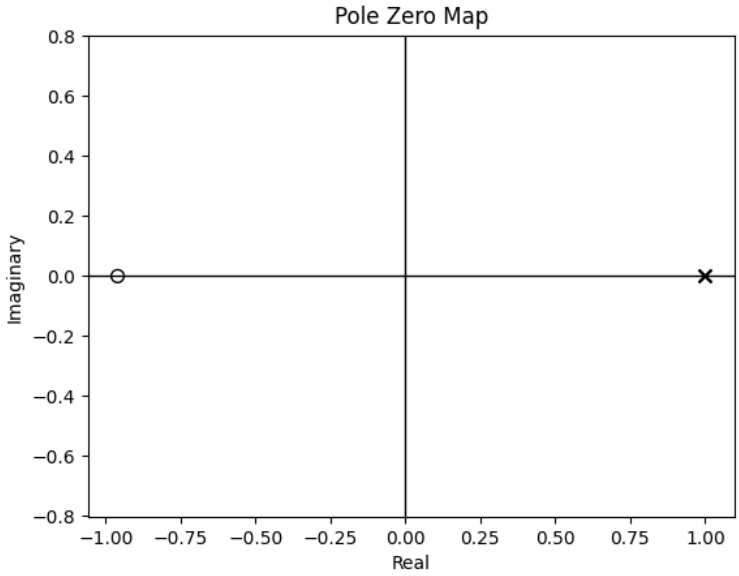

Figure 4: Pole-zero map of the discretized model

A quick check from the transfer function confirms what is shown in the pole-zero map (see figure 5). When the numerator is divided by the gain, we get a zero right at -.96. As theorized and shown by the map, we again have overlapping poles at 1.

```
Transfer Function:
0.0002442 s + 0.0002343
-----------------------
     s^2 - 2 s + 1
```

Figure 5: Transfer function for the discretized system

4 Q4: State Feedback Controller

To fit the performance requirements in question 4, we should make the system as slow as possible. In other words, we should try to put the discrete-time poles closer to 1. The resulting performance shown in figure 6 has its poles set at 0.98 and 0.981. At time 25 second, the distance between Astrobee and the docking station is smaller than 0.1 m, the maximum control input in the beginning is -0.8 N, and the steady state error is 0.002 m.

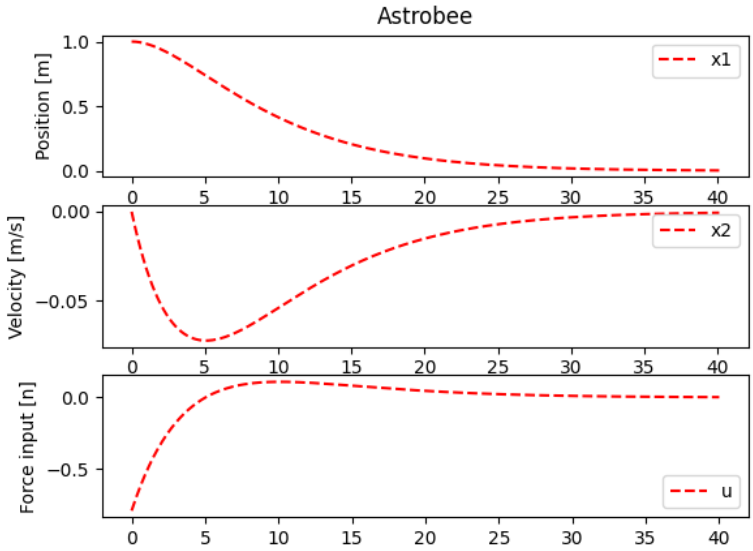

Figure 6: Position, velocity, and force plots for state feedback closed-loop control

5 Q5: Integrator Control

After adding disturbance into the system, we can see a significant increase of steady state error from 0.002 to 0.5 m as shown in figure 7. To minimise steady state error, an integrator is added into the controller. The new control signal thus includes a scaled integral error as shown in eq 5.1-5.2 where K_i is the gain of the integrator, i_k is the integral error, p_k is $p_X[k]$ and p_{ref} a desired position reference. To fit the question's requirements, we chose the integral gain K_i to be 0.027. This resulting steady state error are barely 0.001 as shown in figure 8.

$$u_k = -Lx_k - K_i i_k \tag{5.1}$$

$$i_{k+1} = i_k + h * (p_k - p_{ref}) \tag{5.2}$$

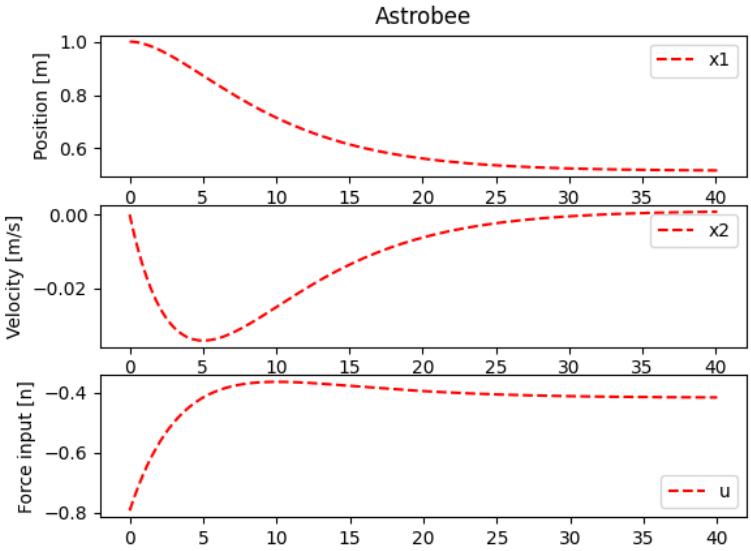

Figure 7: Plots describing closed-loop system response to disturbances

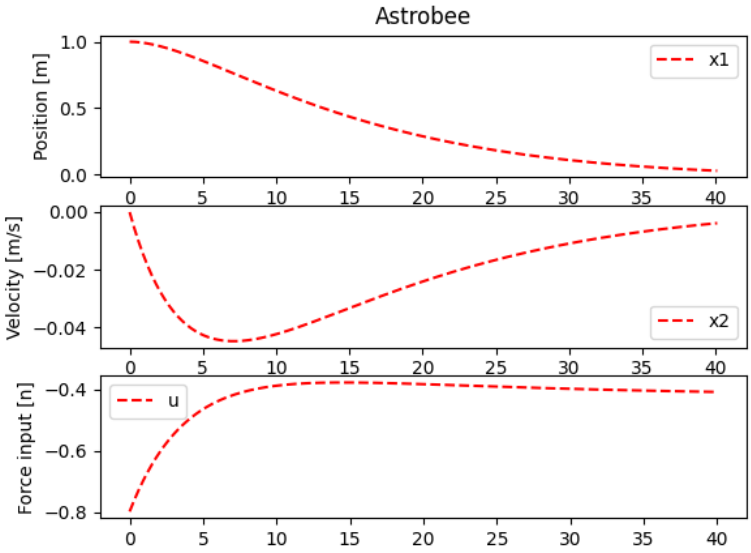

Figure 8: Plots describing closed-loop response with integrator