ECE 351 - Lab 7 - Block Diagrams and System Stability

Ethan Reeder

9 March 2021

Contents

1 Purpose

The purpose of this lab was to gain familiarity with Laplace-domain block diagrams and to use factored transform functions to determine system stability.

2 Procedure

This lab began by declaring 3 functions, G, A, and B, given in Equations 1 through 3 below, as well as a block diagram, shown in Figure 1.

$$G(s) = \frac{s+9}{(s^2-6s-16)(s+4)} \tag{1}$$

$$A(s) = \frac{s+4}{s^2+4s+3} \tag{2}$$

$$B(s) = s^2 + 26s + 168 \tag{3}$$

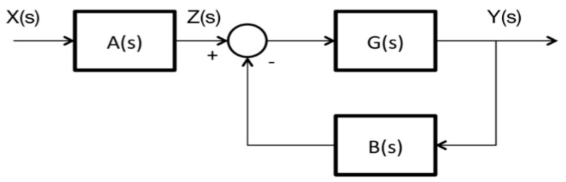

Figure 1: Given Block Diagram

Next, G(s), A(s), and B(s) were reduced to their factored forms, and the poles and zeroes were found, as seen in Equations 4-6 in the Equations section.

Then, the scipy.signal.tf2zpk() function was used to verify the zeroes and poles found by factoring the transfer functions. The output from the console is in the Results section of the report, and it verified the hand results.

Next, the open-loop transfer function was calculated, as seen in Equation 7 in the Equations section, and then expanded using the scipy.signal.convolve() function, as seen in Equation 8.

The step response of the open-loop transfer function was then graphed, as seen in Figure 2 in the Results section.

With analysis of the open-loop function completed, the analysis of the closed-loop function began. The closed-loop transfer function was calculated symbolically from the block diagram, as seen in Equations 9 and 10 of the Equations section. Using the scipy.signal functions, the expanded form of the transform function was then found, as seen in Equation 11.

The step response of the closed-loop transfer function was then graphed, as seen in Figure 3 in the Results Section.

3 Equations

$$G(s) = \frac{s+9}{(s-8)(s+2)(s+4)} \tag{4}$$

Zero at s = -9, poles at s = -4, -2, 8

$$A(s) = \frac{s+4}{(s+3)(s+1} \tag{5}$$

Zero at s = -4, poles at s = -3, -1

$$B(s) = (s+14)(s+12) \tag{6}$$

Zeroes at s = -14, 12, no poles

$$H_o(s) = A(s)G(s) = \frac{s+9}{(s+1)(s+2)(s+3)(s-8)} \tag{7}$$

The above transfer function is not stable because it has a positive pole at s = 8. Distributed, the transfer function reduces to:

$$H_o(s) = \frac{s+9}{s^4 - 2s^3 - 37s^2 - 82s - 48} \tag{8}$$

$$H_c(s) = A(s) * \frac{G(s)}{1 + B(s)G(s)} = \frac{numA}{denA} * \frac{\frac{numG}{denG}}{1 + \frac{numB*numG}{denG}} \tag{9}$$

Simplifying into a proper transfer function:

$$H_c(s) = \frac{numA * numG}{denA * (denG + numB * numG)} \tag{10}$$

$$H_c(s) = \frac{s^2 + 13s + 36}{2s^5 + 41s^4 + 500s^3 + 2995s^2 + 6878s + 4344} \quad (11)$$

The above transfer function is stable because all the poles are negative.

4 Results

Zeroes, Poles, and Amplification for G, A, and B:

```
G:  (array([-9.]), array([ 8., -4., -2.]), 1.0)
A:  (array([-4.]), array([-3., -1.]), 1.0)
B:  (array([-14., -12.]), array([], dtype=float64), 1.0)
```

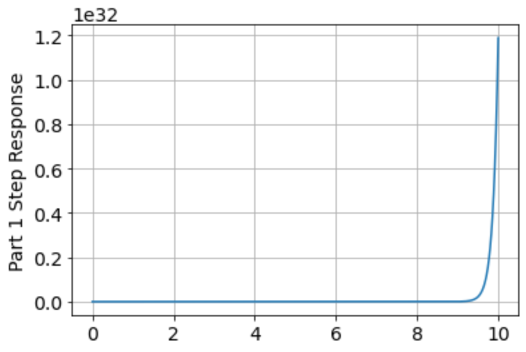

Figure 2: Open-Loop Step Response

This graph supports the idea that the open-loop response is unstable, due to the fact that the result increases exponentially as s increases.

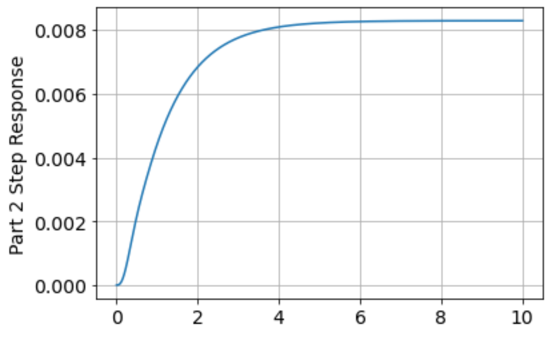

Figure 3: Closed-Loop Step Response

This graph supports the idea that the closed-loop response is stable, due to the fact that it stabilizes around s = 4 at a value of a little larger than 0.008.

5 Questions

In Part 1 Task 5, why does convolving the factored terms using scipy.signal.convolve() result in the expanded form of the numerator and denominator? Would this work with your user-defined convolution function from Lab 3? Why or why not?

The convolution works because convoluting in the t domain is the same as multiplying in the s domain. However, this doesn't work for the convolving function that was defined in Lab 3 because it's not aware of the difference between a function and an array, and is not capable of simply multiplying when it's already been transformed into the Laplace Domain.

Discuss the difference between the open- and closed-loop systems from Part 1 and Part 2. How does stability differ for each case, and why?

The open-loop function is unstable and the closed-loop function is stable. This is because a closed-loop function has the ability for feedback and to "self-stabilize." This ability is lost with no feedback loops.

What is the difference between scipy.signal.residue() used in Lab 6 and scipy.signal.tf2zpk() used in this lab?

The residue function expands a function into its partial fractions, finding the gains and roots for each term. However, the tf2zpk function simply factors the numerator and denominator of a function into its zeroes (roots in the numerator), poles (roots in the denominator), and gain (coefficient of the entire system).

Is it possible for an open-loop system to be stable? What about for a closed-loop system to be unstable? Explain how or how not for each.

Open-loop and closed-loop systems can both be both stable and unstable, since stability and instability are both caused by the polarity of the poles of a function. These could be positive or negative for both kinds loops, thus it depends on the system whether it is stable or unstable, and cannot simply be determined by the loop setup.

Leave any feedback on the clarity/usefulness of lab tasks, expectations, and deliverables.

The lab was clear and the expectations were understood.

6 GitHub Link

https://github.com/EReeder35

7 Conclusion

In conclusion, this lab continued education in how to use Python, and especially in some of the functions included in the scipy.signal library. Additionally, it introduced the concept of stability in a system, as well as clarifying the meaning and importance of poles and zeroes in a transfer function.