ECE 351

Signals and Systems

Lab 7

Block Diagrams and System Stability

Submitted By :
Zachary Pfaff
https://github.com/ZachPfaff

Contents

Block Diagrams and System Stability

Zachary Pfaff

March 1st 2022

1 Introduction

In ECE 350 we have learned about the significance of system block diagrams and how to visually represent them. Block diagrams can have three properties, being a series connection, parallel connection, and a negative feedback loop. With these three properties in mind, the goal of the following lab is to analyze block diagrams and determine if the system is stable through factoring the transfer function. All the functions implemented through code will also be calculated simultaneously in order to check solutions. The block diagram to be analyzed can be viewed below in figure 1.

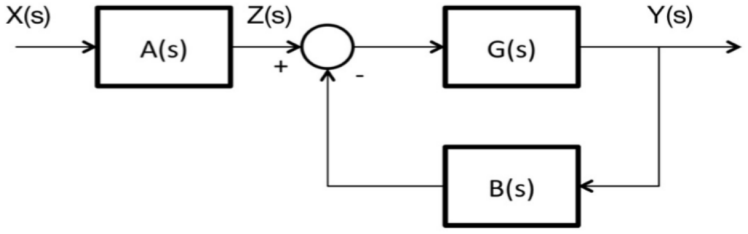

Figure 1: Lab Block Diagram

2 Equations

The block diagram supplied in the lab handout is first analyzed by hand. There are three functions in this system, all of which are listed below. The first task asks us to produce each function in factored form which lets us identify the poles and zeros of each function.

$$G(s) = \frac{s+9}{(s^2-6s-16)(s+4)} = \frac{s+9}{(s-8)(s+2)(s+4)}$$

$$A(s) = \frac{s+4}{s^2+4s+3} = \frac{s+4}{(s+1)(s+3)}$$

$$B(s) = s^2+26s+168 = (s+12)(s+14)$$

At first glance, it appears that the system will already be unstable due to the denominator in G(s). Before making this assumption for the overall system however, we should analyze the open loop and closed loop transfer functions. When the open loop transfer function was evaluated, the following equation was the result. Notice that this transfer function is not stable as expressed in factored form. This is due to a positive pole which will make the function rapidly increase. Code for implementing

this function, as well as its respected plot, are discussed in the methodology section of this report.

$$\frac{s+9}{(s-8)(s+2)(s+1)(s+3)} = \frac{s+9}{s^4-2s^3-37s^2-82s-48}$$

Next, the transfer function was evaluated for the closed loop which resulted in the equation below. Notice that this equation is stable due to there being two parts in the denominator.

$$\frac{(s+9)(s+4)}{(s+1)(s+3)(s-8)(s+2)(s+4)+(s+1)(s+3)(s+12)(s+9)(s+14)}$$

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

3 Methodology

The first task of lab was to use scipy.signal.tf2zpk() to check if G(s), A(s), and B(s) had been factored correctly by hand. The code used to check the poles of each function is shown below.

```
#for G(s)
b1 = [1, 9]
a1 = [1, -6, -16]
z1, p1, k1 = sig.tf2zpk(b1, a1)

print('G(s) numerator = ', z1)
print('G(s) denominator = ', p1)

#for A(s)
b2 = [1, 4]
a2 = [1, 4, 3]
z2, p2, k2 = sig.tf2zpk(b2, a2)

print('A(s) numerator = ', z2)
print('A(s) denominator = ', p2)

#for B(s)
x = [1, 26, 168]
v = np.roots(x)
print('B(s) factored = ' ,v)
```

Listing 1: Code for factoring polynomials

The output to this code can be seen below. Notice that the outputs match the factored forms as calculated in the equations section of this report.

```
G(s) numerator =  [-9.]
G(s) denominator =  [ 8. -2.]
A(s) numerator =  [-4.]
A(s) denominator =  [-3. -1.]
B(s) factored =  [-14. -12.]
```

Figure 2: Output from listing 1 code

The next step in lab was to plot the step response of the open loop transfer function and the closed loop transfer function. To do this in python however, we first needed to expand the numerator and denominator in both loop functions. This was done using the scipy.signal.convolve() command. My process here was to convolve two functions, then convolve the result with the next function until the every pole in the factored polynomial had been convolved. This required multiple convolve commands to be used, although there may have been a better way to implement it my code still worked. The listing below shows the factor expansion of the open loop transfer function. The output to this convolution is shown below in figure 3.

```
1 den3 = sig.convolve(sig.convolve(sig.convolve([1, -8], [1, 2]), [1,
      1]), [1, 3])
2 print('Open Loop Denominator = ', den3)
```

Listing 2: Open loop convolution code

```
Open Loop Denominator =  [  1  -2 -37 -82 -48]
```

Figure 3: Open loop convolution output

The next listing shows the factor expansion of the closed loop transfer function. Notice that two factor expansions were calculated and combined. The figure 4 below shows the output to the numerator factor expansion, and the two denominator factor expansions.

```
1 num = sig.convolve([1 , 9], [1, 4])
2 print('Numerator = ', num)
3
4 den1 = sig.convolve(sig.convolve(sig.convolve(sig.convolve([1, 1],
     [1, 3]), [1, -8]), [1, 2]),[1, 4])
5 print('Denominator1 = ', den1)
6
7 den2 = sig.convolve(sig.convolve(sig.convolve(sig.convolve([1, 1],
     [1, 3]), [1, 12]), [1, 9]),[1, 14])
```

```
print('Denominator2 = ', den2)
```

Listing 3: Closed loop convolution code

```
Numerator =  [ 1 13 36]
Denominator1 =  [   1    2  -45 -230 -376 -192]
Denominator2 =  [   1   39  545 3225 7254 4536]
```

Figure 4: Open loop convolution output

With the factor expansion now discovered, I was able to plot the open loop and closed loop transfer functions. Code for the open loop plot is shown below in listing 4. Code for the closed loop plot is shown below in listing 5.

```
numr2 = [1, 9]
denr2 = [1, -2, -37, -82, -48]
tout2, yout2 = sig.step((numr2, denr2), T = t)

plt.figure(figsize =(12 ,8))
plt.subplot (1,1,1)
plt.plot(tout2, yout2)
plt.title('Step Response Open Loop')
plt.ylabel('y')
plt.grid(True)
```

Listing 4: Step Response Open Loop Code

```
numr = [1, 13, 36]
denr = [2, 41, 500, 2995, 6878, 4344]

tout, yout = sig.step((numr, denr), T = t)

plt.figure(figsize =(12 ,8))
plt.subplot (1,1,1)
plt.plot(tout, yout)
plt.title('Step Response Closed Loop')
plt.ylabel('y')
plt.grid(True)
```

Listing 5: Step Response Closed Loop Code

The following two plots demonstrate the step response of the open loop and closed loop transfer functions. Notice that the open loop transfer function exponentially increases to infinity, which is a sign of an unstable system. The closed loop plot however evens out over time, meaning it is stable.

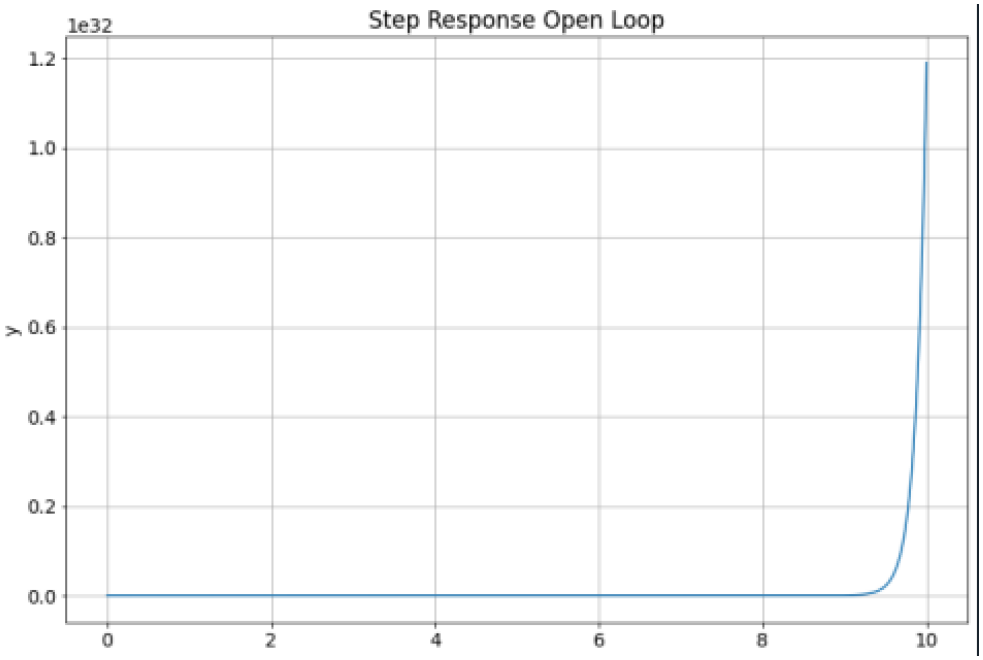

Figure 5: Open Loop Plot

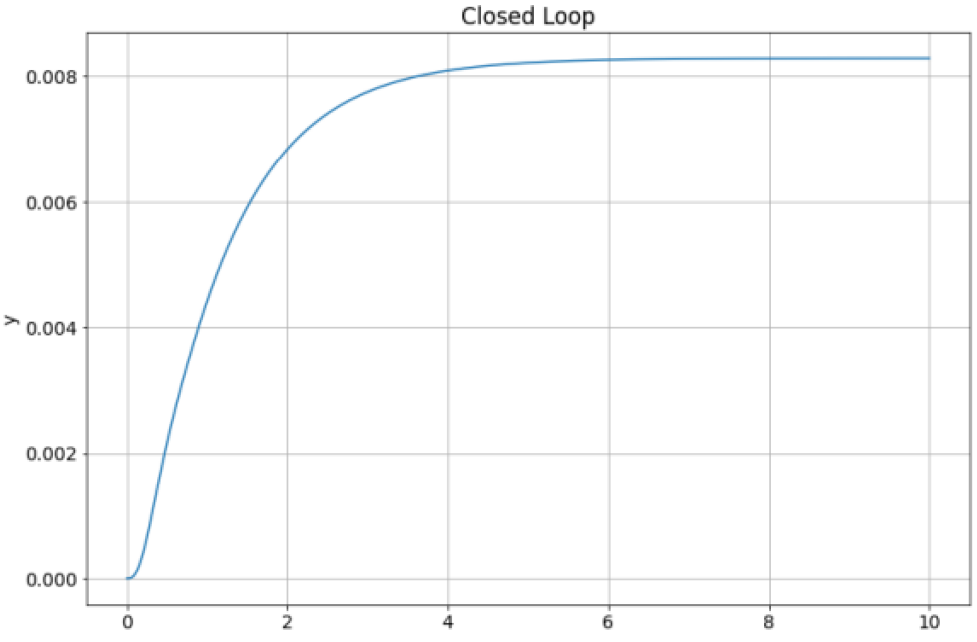

Figure 6: Closed Loop Plot

4 Conclusion

By the end of lab 7, I was able to successfully factor the functions G(s), A(s), and B(s) using python code. I was then able to plot the open loop and closed loop transfer functions after finding the expanded polynomials of each. After completing this lab, I feel that I have a better understanding of closed loop and open loop block diagram analysis, as well as how to implement a transfer function into python code.

5 Questions

1. 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 expanded factored from of the numerator and denominator is able to be found through convolution. This worked because, when working with laplace transforms, convolution becomes multiplication. This would only work for the python command however because the user defined convolution was not created for functions in the s domain.

2. 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 difference between a closed loop and open loop system is the open loop system is evaluated without taking into account the feedback loop, which for the case of our block diagram in lab 6 the system would only be evaluated with the series connection property. Closed loop however means the system is evaluated with the feedback loop in mind which, although will take longer, means the block diagram in lab 6 is evaluated as the series property with the feedback loop property. A good way to think of this is the open loop is the shortest route whereas the closed loop is the longest route.

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

scipy.signal.residue() was used to determine the result of partial fraction expansion whereas scipy.signal.tf2zpk was used to factor polynomials.

4. 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.

It is possible for the open loop system to be stable and the closed loop system to be unstable and vice versa. It ultimately depends on the poles of the factored polynomials. If there are any positive poles in the denominator, then the system is likely unstable unless it is added to another factored polynomial as demonstrated in this lab.

5. Leave any feedback on the clarity/usefulness of the purpose, deliverables, and expectations for this lab.

I did not have any problems following lab instructions.