ECE 351 - Section #53

Lab #7 Report

Block Diagrams and System Stability

Submitted To:
Kate Antonov
University of Idaho
kantonov@uidaho.edu

Submitted By :
Macallyster S. Edmondson
University of Idaho
edmo7033@vandals.uidaho.edu
github.com/mac-edmondson

8th March, 2022

Contents

1 Introduction

The goal of this weeks lab was to become familiar with Laplace-domain block diagrams and use the factored form of the transfer function to determine system stability. These concepts have been covered in class and applying them using computational tools was a great exercise. Yet again, this lab was completed using *Python* through the *Spyder-IDE*. The packages used in the completion of this lab were `numpy` for definitions of mathematical functions, `matplotlib.pyplot` to plot outputs of functions, and `scipy.signal` to perform convolutions for reasons discussed throughout the report.

All code for this lab, including this report, can be found on my Github.

2 Equations

The equations used within this lab are shown in this section. The equations will be referenced by number throughout the rest of the report.

$$H_{OL}(s) = \frac{(s+9)(s+4)}{(s+2)(s+4)(s-8)(s+3)(s+1)} \tag{1}$$

$$H_{CL}(s) = \frac{A_{num}G_{num}}{A_{den}G_{den} + A_{den}B_{num}G_{num}} \tag{2}$$

3 Methodology

3.1 Lab: Part 1

In part 1 of this lab, we calculated the Open-Loop transfer function of the block-diagram system given in the lab handout by hand and using Python. The calculated equation can be seen in Equation 1. This was verified with Python using the `scipy.signal.tf2zpk()` function and the `scipy.signal.convolve()` function, giving the poles, zeros and expanded numerator and denominator coefficients of the total transfer function (Results seen in: Figure 1). As can be seen, the OL transfer function is unstable due to at least one pole being in the RHP. This can be obviously seen in the graph for this transfer function (Figure 2) since $h(t) \to \infty$ as $t \to 8$.

Below can be seen the code implementation of the tasks carried out in Part 1 of this lab.

```
  # PART 1
print("### PART 1 ###\n")

# 1/2
```

```python
zG, pG, kG = spsig.tf2zpk([1, 9], [1, -2, -40, -64])
zA, pA, kA = spsig.tf2zpk([1, 4], [1, 4, 3])
# zB, pB, kB = np.roots(1, 26, 168)

print("G(s): z = ", zG, "p = ", pG)
print("A(s): z = ", zA, "p = ", pA)

#3/4 (Open Loop HT = A*G)
num = spsig.convolve([1, 9], [1, 4])
den = spsig.convolve([1, -2, -40, -64], [1, 4, 3])
print("\nOpen Loop Numerator: ", num)
print("Open Loop Denominator: ", den)

#5
tout, y1 = spsig.step(spsig.lti(num, den))
plt.figure(figsize = (10, 11))
plt.subplot(1, 1, 1)
plt.plot(tout, y1, "b-")
plt.grid()
plt.ylabel('h(t) (Open Loop)')
plt.xlabel('t')
plt.title('Open Loop TF Step Response')
```

3.2 Lab: Part 2

In Part 2 of this lab, we again used the `scipy.signal.tf2zpk()` and `scipy.signal.convolve()` functions. This time, they were used to find the closed-loop transfer function of the block-diagram system given in the lab handout. Rather than doing a hand calculation to find the exact transfer function, the transfer function was found symbolically, seen in equation (2). This symbolic equation made finding the transfer function of the closed-loop system very easy and the expanded numerator and denominator coefficient matrices can be seen in Figure 1, while the graph of the transfer function can be seen in Figure 3. Both outputs tell us that the closed-loop transfer function is stable!

Below can be seen the code implementation of the tasks carried out in Part 2 of this lab.

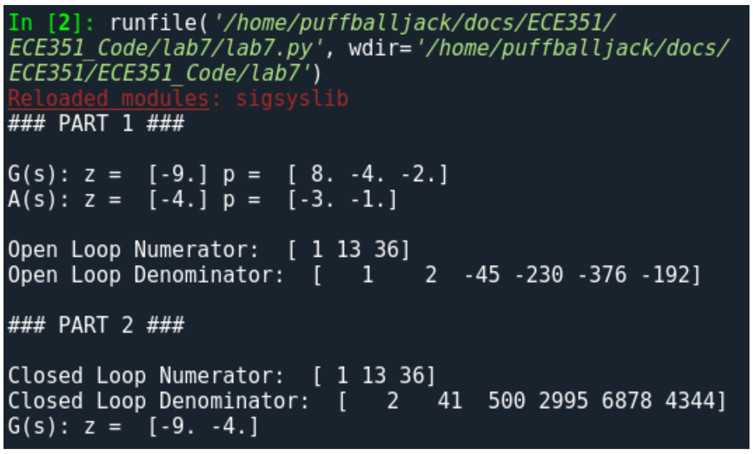
```python
 #PART 2
 print("\n### PART 2 ###\n")
 #1/2

 numG = [1, 9]
 denG = [1, -2, -40, -64]

 numA = [1, 4]
 denA = [1, 4, 3]

```

```
B = [1, 26, 168]

numMain = spsig.convolve(numA, numG)
denT1 = spsig.convolve(denG, denA)
denT2 = spsig.convolve(denA, spsig.convolve(B, numG))
denMain = denT1 + denT2

#3
print("Closed Loop Numerator: ", numMain)
print("Closed Loop Denominator: ", denMain)
z,p,k = spsig.tf2zpk(numMain, denMain)
print("G(s): z = ", z)

#4
tout, y1 = spsig.step(spsig.lti(numMain, denMain))
plt.figure(figsize = (10, 11))
plt.subplot(1, 1, 1)
plt.plot(tout, y1, "b-")
plt.grid()
plt.ylabel('h(t) (Closed Loop)')
plt.xlabel('t')
plt.title('Closed Loop TF Step Response')
```

4 Results

The results of this lab are very straightforward. The implementation of all functions worked as expected and the results are as expected.

The deliverables for Parts 1 & 2 of this lab can be seen in Figures 1, 2, & 3, below.



```
In [2]: runfile('/home/puffballjack/docs/ECE351/
ECE351_Code/lab7/lab7.py', wdir='/home/puffballjack/docs/
ECE351/ECE351_Code/lab7')
Reloaded modules: sigsyslib
### PART 1 ###

G(s): z =  [-9.] p =  [ 8. -4. -2.]
A(s): z =  [-4.] p =  [-3. -1.]

Open Loop Numerator:  [ 1 13 36]
Open Loop Denominator:  [   1    2  -45 -230 -376 -192]

### PART 2 ###

Closed Loop Numerator:  [ 1 13 36]
Closed Loop Denominator:  [   2   41  500 2995 6878 4344]
G(s): z =  [-9. -4.]
```

Figure 1: Program Output

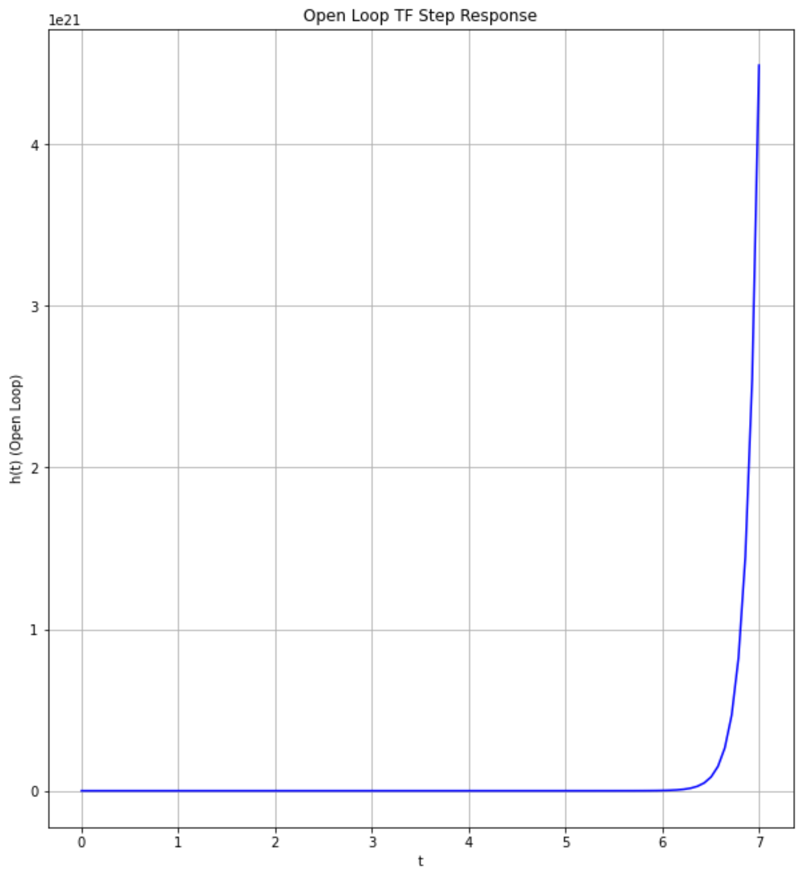


Figure 2: Part 1, Task 5

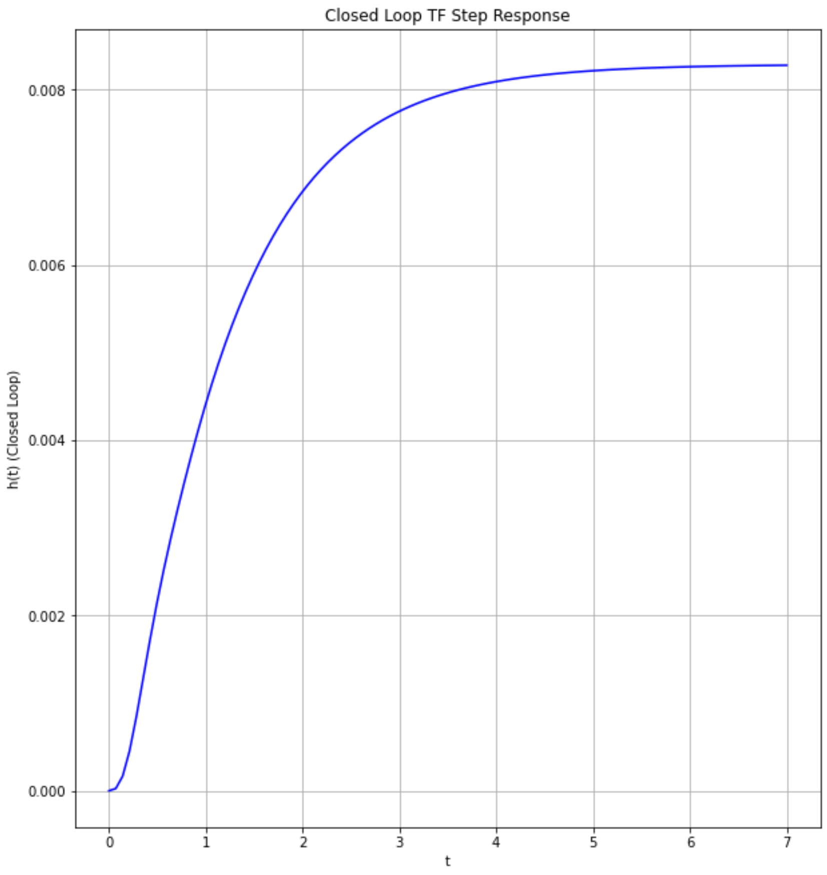


Figure 3: Part 2, Task 4

5 Error Analysis

No sources of error were seen throughout this lab, and I did not run into any real problems while implementing anything lab lab asked. Part 2 was time consuming and without a lab TA to tell me that my graph was correct, I'm not sure how I would know I did it right. I wish this was something included in this lab.

6 Questions

1. Using the `scipy.signal.convolve()` function to find the expanded form of the numerator or denominator works due to the fact that all coefficient matrices represent coefficients for s-domain functions. As we know, convolution in the time-domain is the same as multiplication in the s-domain. With that in mind, the convolution function we made in **Lab 3** would not have worked as it was made strictly based on the summation definition and only worked to graph convolutions discretely, not perform "symbolic" math as with the `scipy` function.
2. As seen throughout this lab, the open-loop transfer function is unstable and the closed-loop transfer function was stable. This may be the reason you choose to have feedback in a system. For any system to be implemented realistically, it should be stable. Feedback could be used to make an open-loop system stable.
3. The `scipy.signal.tf2zpk()` function is used to find the zero, pole, and gain of transfer function. The `scipy.signal.residue()` function takes the numerator and denominator coefficients of a function and performs partial fraction expansion, returning the residue (denominator term) and poles for each term in the PFE. (See scipy reference.)
4. It is possible for any system to be stable or unstable. It just depends on the transfer functions that go into making the overall transfer function. $\frac{Y(s)}{F(s)}$. This goes along with my answer to Question 3.
5. See the end of Error Analysis for some feedback on the lab. Other than that, purpose, deliverables, and expectations were very clear for this lab.

7 Conclusion

In conclusion, I feel this lab was very successful. The implementation of the code in this lab was quite simple and I really enjoyed seeing how you could find the total transfer function with some symbolic math and Python. All in all, I am very satisfied with what this lab has taught me and feel it was an excellent use of time.

8 Attachments

No attachments for this lab.