University of Idaho

ECE 351 - Section #53

Lab #6 Report

Partial Fraction Expansion Using Python

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

3rd January, 2022

Contents

1 Introduction

The goal of this weeks lab was to use the `scipy.signal.residue()` function to perform partial fraction expansion within Python. I found this lab very useful and was able to apply the methods taught in this lab to double check homework submissions. Before the lab, a preliminary was completed which can be found attached to this report. The preliminary directly correlates to Part 1 (3.1) of this lab. 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 the step response and find the partial fraction expansion of a polynomial.

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.

$$y''(t) + 10y'(t) + 24y(t) = x''(t) + 6x'(t) + 12x(t) \tag{1}$$

$$y^{(5)}(t) + 18y^{(4)}(t) + 218y^{(3)}(t) + 2036y^{(2)}(t) + 9085y^{(1)}(t) + 25250y(t) = 25250x(t) \tag{2}$$

3 Methodology

3.1 Lab: Part 1

Part 1 of this lab was very simple. The code implemented for this part of the lab plots the step response based on the hand-calculated time-domain function based on the differential equation (1) found in the preliminary, seen in 8, as well as the step response found using the `scipy.signal.step()` function. Additionally, we used the `scipy.signal.residue()` function to find the R, P, & K Partial Fraction Expansion results of Y(s), again seen in 8.

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

```python
#PART 1
#1/2
 def y(t) :
    h = (1/2 - 1/2 * np.exp(-4*t) + np.exp(-6*t)) * ss.u(t)
    return h

```

```
system = ([1, 6, 12], [1, 10, 24])

t = np.arange(0, 2 + step_size, step_size)
y1 = y(t)

tout, y2 = spsig.step(system, T=t);

#Plot...

#3

b = [0, 1, 6, 12]
a = [1, 10, 24, 0]

R, P, K = spsig.residue(b, a)

print("Part 1:")
print("Residue of Poles (same order as presented below): \n", R, "\n\nPoles in ascending order of mag.:\n", P, "\n\nCoefficent of Direct Polynomial Term: \n", K)
```

3.2 Lab: Part 2

In Part 2 of this lab, we used the `scipy.signal.residue()` function to perform a PFE expansion on a function that would be difficult to analyze by hand, in this case Equation (2). Additionally, I made a function to implement the Cosine Method to plot the time-domain output of the solution to the differential equation. This was further verified using the `scipy.signal.step()` with the system based on Equation (2). The printed output of the PFE result was also printed.

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

```
#PART 2

#1

b = [0,   0,   0,    0,    0,     0, 25250]
a = [1, 18, 218, 2036, 9085, 25250, 0]

R, P, K = spsig.residue(b, a)

print("Part 2:")
print("Residue of Poles (same order as presented below): \n", R, "\n\nPoles in ascending order of mag.:\n", P, "\n\nCoefficent of Direct Polynomial Term: \n", K)


```

```python
#2
def cos_mthd(R, P, t):
    y = 0;
    for i in range(0, len(R)):
        k_mag = np.absolute(R[i])
        k_ang = np.angle(R[i])
        alpha = np.real(P[i])
        omega = np.imag(P[i])
        y += k_mag*np.exp(alpha*t)*np.cos(omega*t + k_ang)*ss.u(t)
    return y

def y_s1(t):
    y = (626.375/4)*np.exp(-3*t)*np.sin(4*t-np.deg2rad(156.615))*ss.u
  (t)
    return y

def y_s2(t):
    y = (186.75/10)*np.exp(-1*t)*np.sin(10*t - np.deg2rad(143.723))*
  ss.u(t)
    return y

def y_t(t):
    y = (1 * ss.u(t)) - (0.215*np.exp(-10*t)*ss.u(t)) + y_s1(t) +
  y_s2(t)
    return y

#3
system = ([0,   0,   0,    0,   0,  25250], [1, 18, 218, 2036, 9085,
  25250])

t = np.arange(0, 4.5 + step_size, step_size)
# y1 = y_t(t)
y1 = cos_mthd(R, P, t)

tout, y2 = spsig.step(system, T=t);

#Plot...
```

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 & 4, below.

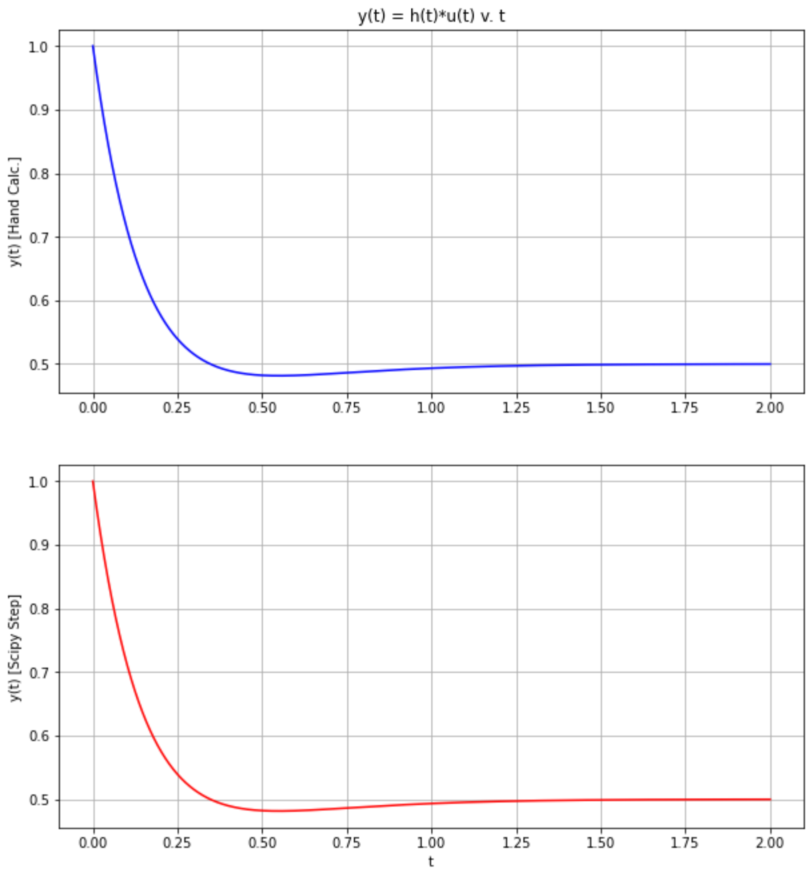
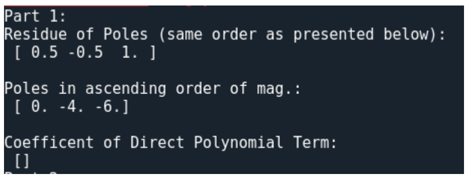


Figure 1: Part 1, Task 1



```
Part 1:
Residue of Poles (same order as presented below):
 [ 0.5 -0.5  1. ]

Poles in ascending order of mag.:
 [ 0. -4. -6.]

Coefficent of Direct Polynomial Term:
 []
```

Figure 2: Part 1, Task 2

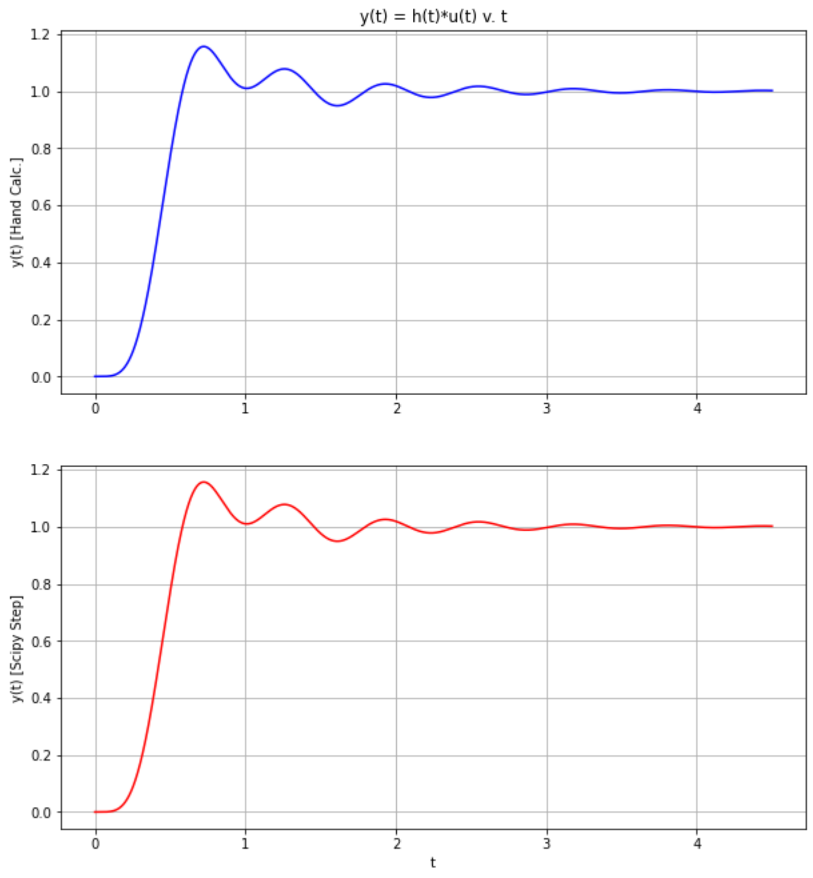
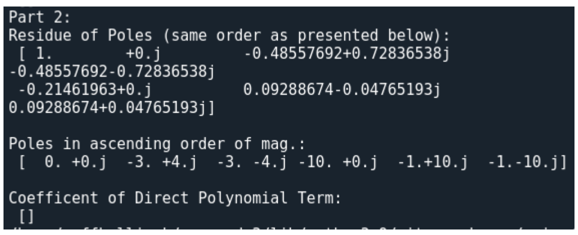


Figure 3: Part 2, Task 1



```
Part 2:
Residue of Poles (same order as presented below):
 [ 1.        +0.j          -0.48557692+0.72836538j
-0.48557692-0.72836538j
 -0.21461963+0.j          0.09288674-0.04765193j
0.09288674+0.04765193j]

Poles in ascending order of mag.:
 [  0. +0.j  -3. +4.j  -3. -4.j -10. +0.j  -1.+10.j  -1.-10.j]

Coefficent of Direct Polynomial Term:
 []
```

Figure 4: Part 2, Task 2

5 Error Analysis

No sources of error were seen throughout this lab, though I did run into trouble while completing Part 2 of this lab. At first, I had decided to attempt to solve Equation (2) by hand using the sine method but was unsuccessful. I then followed the advice of the Lab TA and lab sheet and decided to implement a function which took the outputs of the `scipy.sig.residue` function to plot the time-domain response of the equation, as seen in 3.2.

6 Questions

1. The cosine method can still be used for a non-complex pole-residue term is due to the fact that for a non-complex pole-residue term, ω in p is 0 and you are just then left with the coefficient of a PFE term times $e^{\alpha t}$.
2. This lab and its tasks were very concise in what is expected for deliverables. The preliminary work fit perfectly with not only what the lab was about, but also what we have been covering in class.

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 accurate the `scipy.signal` functions were as compared to hand calculated functions. 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

1. Pre-Lab

Mac Edmundsen

Partial Fraction Expansion

Pre - Lab 6

Spring 2021

1 Purpose

To use Laplace transforms and partial fraction expansion to find the step response of a second-order differential equation.

2 Deliverables

Turn in your solution to the problem at the beginning of lab 6. Significant values and equations must be typed and properly formatted using LaTeX. Please attach any hand calculations to the back of the typed result.

3 Tasks

A system is described by the following differential equation:

$$y''(t) + 10y'(t) + 24y(t) = x''(t) + 6x'(t) + 12x(t) \tag{1}$$

1. By hand, find the transfer function. Assume all initial conditions are zero.
2. For the system $H(s)$, find $y(t)$ for a step input using partial fraction expansion and inverse Laplace transforms. Perform the calculation by hand and type your final answer and other significant results.

1. $y''(t) + 10y'(t) + 24y(t) = x''(t) + 6x'(t) + 12x(t)$

$$s^2Y(s) + 10sY(s) + 24Y(s) = s^2X(s) + 6sX(s) + 12X(s)$$

$$\Rightarrow Y(s)(s^2+10s+24) = X(s)(s^2+6s+12)$$

$$\Rightarrow H(s) = \frac{Y(s)}{X(s)} = \frac{s^2+6s+12}{s^2+10s+24}$$

$$\Rightarrow \boxed{H(s) = \frac{s^2+6s+12}{s^2+10s+24}}$$

2. Step-Response:

$$H(s)\cdot U(s) = H(s)\cdot\frac{1}{s}$$

$$= \frac{s^2+6s+12}{s(s+4)(s+6)} = \frac{A}{s} + \frac{B}{s+4} + \frac{C}{s+6}$$

$$\Rightarrow A = \frac{s^2+6s+12}{(s+4)(s+6)}\bigg|_{s=0} \Rightarrow A = \frac{12}{24} = \boxed{\frac{1}{2} = A}$$

$$\Rightarrow B = \frac{s^2+6s+12}{s(s+6)}\bigg|_{s=-4} \Rightarrow B = \frac{16+12-24}{-8} = \frac{4}{-8}$$

$$\Rightarrow \boxed{B = -\frac{1}{2}}$$

$$\Rightarrow C = \frac{s^2+6s+12}{s(s+4)}\bigg|_{s=-6} \Rightarrow C = \frac{12}{-2\cdot-6} = \frac{12}{12} = 1$$

$$\boxed{C = 1}$$

$$\Rightarrow H(s)\cdot U(s) = \frac{1/2}{s} - \frac{1/2}{s+4} + \frac{1}{s+6}$$

$$\Rightarrow \boxed{H(t) * u(t) = \left(\frac{1}{2} - \frac{1}{2} e^{-4t} + e^{-6t}\right) u(t)}$$