ECE 351 - 52

Signals and Systems 1

Lab 11

Submitted By :
Owen Blair

Contents

1 Important Notes

It is important to note that plt.show() is needed at the end of the python code to properly show the plots of each function. It was not included in every section of code that plots a function(s) because it is assumed that its included at the end the code. It is also assumed that the following is included at the beginning of the program:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig
import scipy.fftpack as fft
import math as m

# The following package is not included with the Anaconda distribution
# and needed to be installed separately. The control package also has issues
# working on macs and a PC or a linux distribution is needed
import control as con
```

A note of caution for Mac users is that the control package has issues with the mac operating systems and needs to be run on a windows or Linux machine.

The web address to the GitHub where LaTeXcode is stored is here:
https://github.com/Blairis123/ECE351_Reports

The web address to the GitHub where the Python code is here:
https://github.com/Blairis123/ECE351_Code

2 Part 1 Description

2.1 Part 1

Part 1 is about analyzing a discrete system using python and a function developed by Christopher Felton and by hand. The function to be analyzed is assumed to be at rest and is:

$$y[k] = 2x[k] - 40x[k-1] + 10y[k-1] - 16y[-2]$$

The hand calculated derivation for the z-transform and solution can be found in the Results section with the subheading *H(z) Hand Derivation* and *h[k] Hand Derivation*. Then the *scipy.signal.residuez()* function was used to verify the partial fraction decomposition. The output can from this can be seen in the Results section under subheading *Residuez Function Output*. The code used to generate the output can be seen in *Listing 1: Residuez Function Output Code*. H(z) was also used in the *zplane()* function and the zeros and poles were plotted with respect to the unit circle. This can be seen in *Figure 1: H(z) Pole Plot* and the code used to do so can be seen in *Listing 2*.

The magnitude and phase responses of H(z) were then plotted using the *scipy.signal.freqz()* function. This function calculates the frequency response of a filter and returns an array of complex numbers. This array can then be used to calculate the phase and magnitude response of the filter. It is important that the input includes *whole = True* within the command. The frequency response plots can be found in *Figure 2: H(z) Frequency Response*. The code used to do this can be seen in *Listing 3: H(z) Frequency Response Code*.

3 Results

3.1 H(z) Hand Derivation

After transforming the function to the z-domain the result is:

$$Y(z) = 2X(z) - 40(z^{-1}X(z) - x[-1] + 10(z^{-1}Y(z) - y[-1]) - 16(Y(z)z^{-2} - z^{-1}y[-1] - y[-2])$$

Knowing that the system is initially at rest, the function can be further simplified to:

$$Y(z) = 2X(z) - 40z^{-1}X(z) + 10z^{-1}Y(z) - 16z^{-2}Y(z)$$

$$(40z^{-1} - 2)X(z) = (-16z^{-2} + 10z^{-1} - 1)Y(z)$$

$$\frac{Y(z)}{X(z)} = H(z) = \frac{40Z^{-1} - 2}{-16z^{-2} + 10z^{-1} - 1}$$

$$H(z) = \frac{2z^2 - 40z}{(z-8)(z-2)}$$

3.2 h[k] Hand Derivation

$$H(z) = \frac{2z^2 - 40z}{(z-8)(z-2)}$$

$$\frac{H(z)}{z} = \frac{2z - 40}{(z-8)(z-2)}$$

Using partial fraction decomposition it can be found that:

$$\frac{H(z)}{z} = \frac{-4}{z-8} + \frac{6}{z-2}$$

Multiplying by z the result is:

$$H(z) = 6\frac{z}{z-8} - 4\frac{z}{z-2}$$

The inverse Z transform of H(z) is:

$$h[k] = (6(2)^k - 4(8)^k)u[k]$$

3.3 Residuez Function Output

```
runfile('D:/U_of_I/Fall_2021/ECE-351/Lab11/Lab11_OwenBlair.py',
    wdir='D:/U_of_I/Fall_2021/ECE-351/Lab11')
[ 6. -4.]
[2. 8.]
[]
```

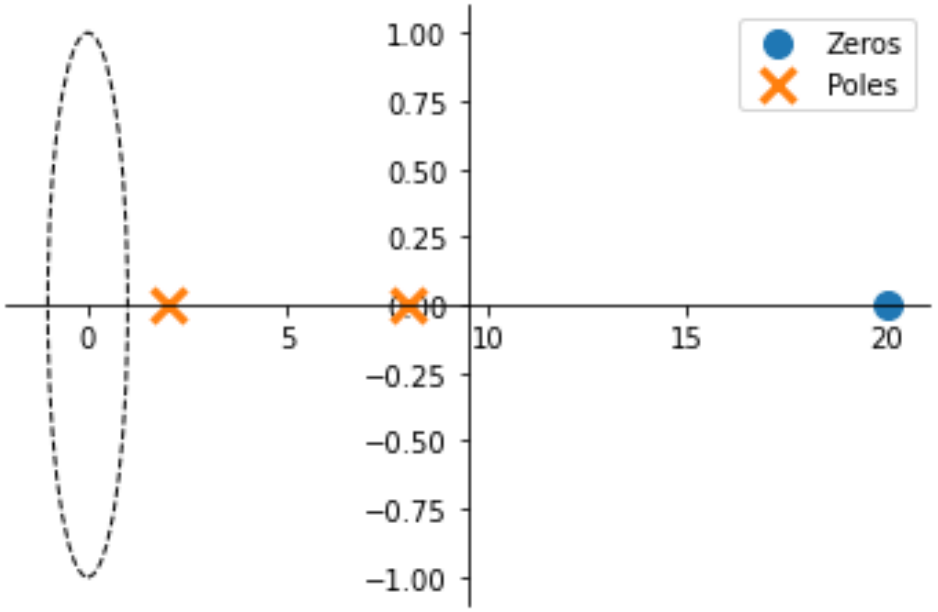
4 Questions

4.1 Question 1

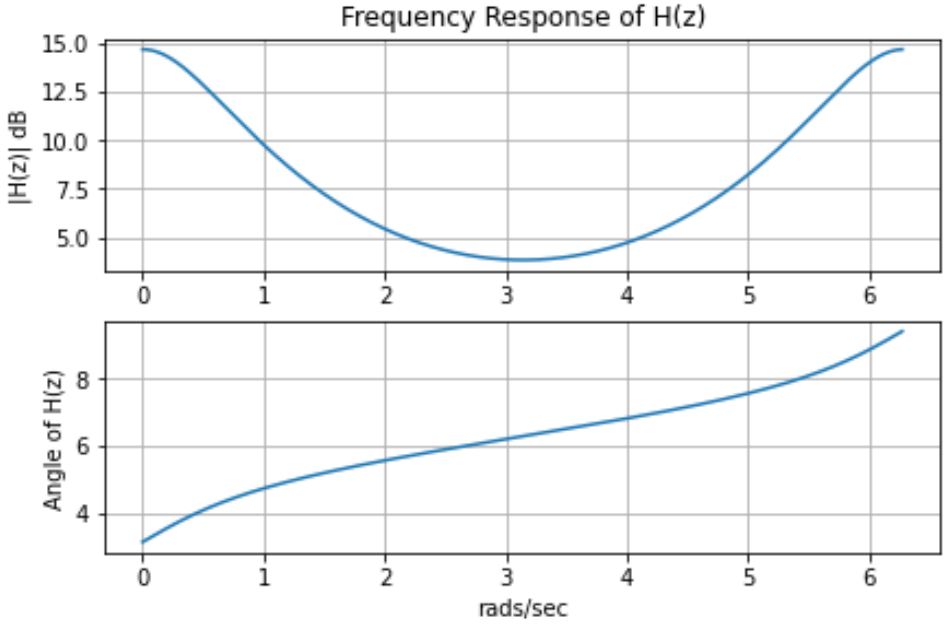
Looking at the plot generated by the zplane() function it can be determined that H(z) is not stable. This is because the poles of H(z) do not reside within the unit circle. The unit circle is the dashed oval on the right side of the plot in *Figure 1.*

5 Figures

5.1 Figure 1: H(z) Pole Plot



5.2 Figure 2: H(z) Frequency Response



6 Listings

6.1 Listing 1:

```
    # Verify partial fraction expantion
num = [2, -40]
den = [1, -10, 16]

[res, poles, coeff] = sig.residuez( num, den)

print(res)
print(poles)
print(coeff)
```

6.2 Listing 2:

```
# Copyright (c) 2011 Christopher Felton
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU Lesser General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU Lesser General Public License for more details.
#
# You should have received a copy of the GNU Lesser General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>.
#
# The following is derived from the slides presented by
# Alexander Kain for CS506/606 "Special Topics: Speech Signal Processing"
# CSLU / OHSU, Spring Term 2011.
#
#
#
# Modified by Drew Owens in Fall 2018 for use in the University of Idaho's
# Department of Electrical and Computer Engineering Signals and Systems I Lab
# (ECE 351)
#
```

```
# Modified by Morteza Soltani in Spring 2019 for use in the ECE 351 of the U of
# I.
#
# Modified by Phillip Hagen in Fall 2019 for use in the University of Idaho's
# Department of Electrical and Computer Engineering Signals and Systems I Lab
# (ECE 351)

def zplane(b,a,filename=None):
    """Plot the complex z-plane given a transfer function.
    """

    import numpy as np
    import matplotlib.pyplot as plt
    from matplotlib import patches

    # get a figure/plot
    ax = plt.subplot(111)

    # create the unit circle
    uc = patches.Circle((0,0), radius=1, fill=False,
                        color='black', ls='dashed')
    ax.add_patch(uc)

    # The coefficients are less than 1, normalize the coeficients
    if np.max(b) > 1:
        kn = np.max(b)
        b = np.array(b)/float(kn)
    else:
        kn = 1

    if np.max(a) > 1:
        kd = np.max(a)
        a = np.array(a)/float(kd)
    else:
        kd = 1

    # Get the poles and zeros
    p = np.roots(a)
    z = np.roots(b)
    k = kn/float(kd)

    # Plot the zeros and set marker properties
    t1 = plt.plot(z.real, z.imag, 'o', ms=10,label='Zeros')
    plt.setp( t1, markersize=10.0, markeredgewidth=1.0)

    # Plot the poles and set marker properties
```

```
    t2 = plt.plot(p.real, p.imag, 'x', ms=10,label='Poles')
    plt.setp( t2, markersize=12.0, markeredgewidth=3.0)

    ax.spines['left'].set_position('center')
    ax.spines['bottom'].set_position('center')
    ax.spines['right'].set_visible(False)
    ax.spines['top'].set_visible(False)

    plt.legend()

    # set the ticks
    # r = 1.5; plt.axis('scaled'); plt.axis([-r, r, -r, r])
    # ticks = [-1, -.5, .5, 1]; plt.xticks(ticks); plt.yticks(ticks
   )

    if filename is None:
        plt.show()
    else:
        plt.savefig(filename)

    return z, p, k
```

6.3 Listing 3:

```
num = [2, -40]
den = [1, -10, 16]

    # Plot magnitude and phase response of H(z)
w, H = sig.freqz(num, den, whole = True)

    #This is for the angle!
angles = np.unwrap(np.angle(H))

    # Convert H into dB
H = 20 * np.log10(H)

    # Do some ploting!
plt.figure(figsize=(10,7))
plt.figure(constrained_layout=True)
plt.subplot(2,1,1)
plt.title("Frequency Response of H(z)")
plt.ylabel("|H(z)| dB")
plt.plot(w,H)
plt.grid()

plt.subplot(2,1,2)
plt.ylabel("Angle of H(z)")
plt.xlabel("rads/sec")
```

```
25 plt.plot(w,angles)
26 plt.grid()
```