ECE 351 - Section 52

Lab 11

Z - Transform Operations

Submitted By :
Ben Bunce

Contents

1 Introduction

Using Pythons built in functions, and a function developed by Christopher Felton, a discrete system is analyzed and plots are generated.

2 Equations

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

Task 1 Derivation: $H(z) = \frac{2z(z-20)}{(z-2)(z-8)}$

Task 2 Derivation: $h[k] = 6 * 2^k - 4 * 8^k$

3 Methodology

First, the transfer function H(z) is found for the equation given. This transfer function is then broken apart into simpler terms and rewritten using partial fraction expansion. This was done to get the function in a form that can be transformed. Once in this form, the h[k] transform is found. Then, using the 'sig.residuez()' function, the transfer function is plugged in to verify that it matches the partial fraction expansion.

Next, the 'zplane' function written by Christopher Felton, is used to obtain the pole-zero plot for the transfer function H(z).

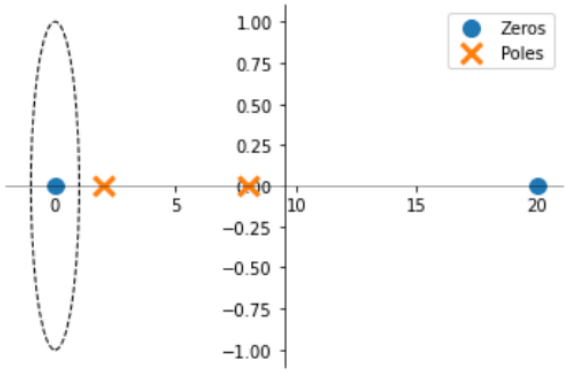
Finally, the 'sig.freqz()' function is used to plot the magnitude and phase responses of H(z).

4 Results

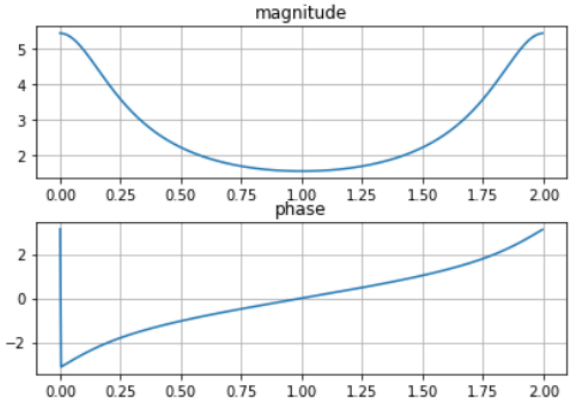
Part 1
Task 3 Output - PFD Verification:

```
[ 6. -4.]
[2. 8.]
```

Task 4 Output - zplane func:



Task 5 Output - freqz func:



5 Questions

Looking at the plot generated in Task 4, is H(z) stable? Explain why or why not.

The plot of the transfer function H(z) is unstable. This is because it goes off to infinity in both directions. Also, in the equation $A(p_1)^k + B(p_2)^k$, both p_1 and p_2 are greater than one.

6 Conclusion

In this lab, a transfer function was found for a given function, and then the z-transform was found for the transfer function. Then, a zplane function was used to obtain the pole-zero plot for the transfer function. The 'sig.freq' function was then also used to plot the magnitude and phase responses of H(z). These plots turned out as expected given the transfer function, and showed an unstable transfer function.

7 Appendix

7.1 zplane() function

```
#
# 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/ license s/ >.
#
# 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 (1 , 1 , 1)

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

# the coefficients are less than 1 , normalize the coefficients
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
```