Assigment 8.

Root Locus Python Module Extension

Worked in group with Robin & David

Questions for the lab class portion:

I. Is there a function to find the gain from the damping ratio? or is matlab or trial and error really the only way to do it? Is there some formula or approximation method that matlab is using?
II. Is the frequency really at a point on the RLocus plot really just the distance to the origin? Why is that?
III. Does our code fail when there are two or fewer poles? We noticed sometimes the values were slightly off compared to MatLab in that case.

Test the homework8 root_locus_extras module

NOTE: from homework8 import root_locus_extras as rlx requires homework8 (or a symlink) to be in the ipython notebook directory, or added to python path.

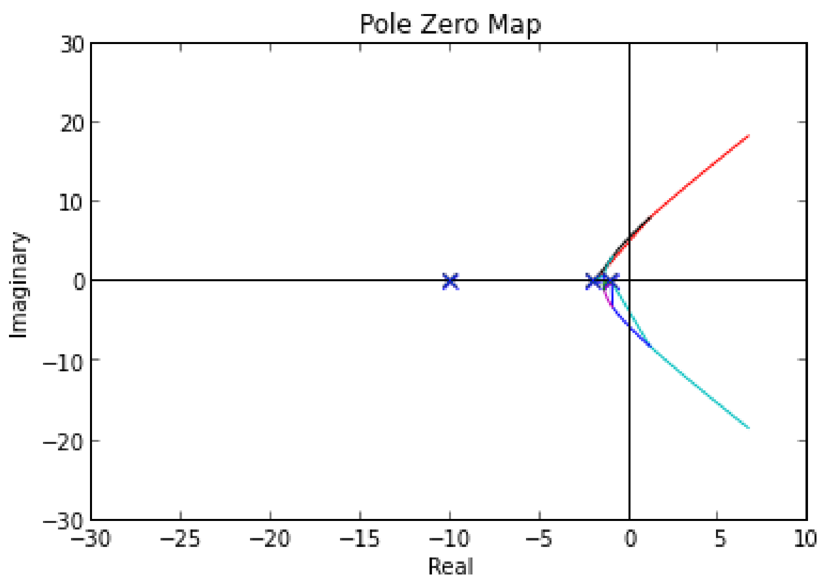
This week requested a python module. Which can't be written in iPython Notebook. The submission this week is in the following format:

```
hw8
├── MRC_Moriarty_Schriver_Ackerson_20130603.zip (the zip submitted: zip
9 -r --excludes ... hw8 )
├── MRC_AlexMoriarty_20130603.ipynb (this file)
├── MRC_AlexMoriarty_20130603.txt (time file)
├── doc ( pydoc generated)
│   ├── README.md
│   ├── homework8.root_locus_extras.html
│   └── hw8_test.html
├── src ( this weeks code )
│   ├── homework8 ( our package )
│   │   ├── __init__.py ( empty )
│   │   └── root_locus_extras.py ( our module )
│   └── hw8_test.py ( a test )
└── test
    └── RLX_Test.ipynb ( another test using ipynb )
```

In [8]:
```python
from control import matlab
import sympy
from homework8 import root_locus_extras as rlx


def main():
    print "This is an example of homework8 root locus extras"
    s = sympy.Symbol("s")
    S = 1 / ((s+1)*(s+2)*(s+10))

    numerS = map(float, sympy.Poly(S.as_numer_denom()[0], s).all_coeffs
    denomS = map(float, sympy.Poly(S.as_numer_denom()[1], s).all_coeffs
    tf = matlab.tf(numerS, denomS)

    gain = 164.5
    damping_ratio = 0.174

    print "provided test data :"
    print "system =", tf, "\ngain =", gain,"\ndamping ratio =",damping_


    print "Gain ", rlx.gainFromDampingRatio(tf, damping_ratio)
    print "Poles: ", rlx.polesFromTransferFunction(tf)
    print "Overshoot: ", rlx.overshootFromDampingRatio(tf, damping_rati
    print "Damping Ratio: ", rlx.dampingRatioFromGain(tf, gain)
    print "Overshoot: ", rlx.overshootFromGain(tf, gain)
    print "Frequency: ", rlx.frequencyFromGain(tf, gain)
```

In [3]:
```
main()
```

```
This is an example of homework8 root locus extras
provided test data :
system =
           1
------------------------
s^3 + 13 s^2 + 32 s + 20

gain = 0.11
damping ratio = 1.0
Gain  0.1
Poles:  [-10.  -2.  -1.]
Overshoot:  0
Damping Ratio:  0
Overshoot:  100.0
Frequency:  0
```



To view documentation use: ?rlx.functionName

In [6]:
```
#?rlx.gainFromDampingRatio()
#?rlx.polesFromTransferFunction()
#?rlx.overshootFromDampingRatio()
#?rlx.dampingRatioFromGain()
#?rlx.overshootFromGain()
#?rlx.frequencyFromGain()
```

In [ ]:
```
```