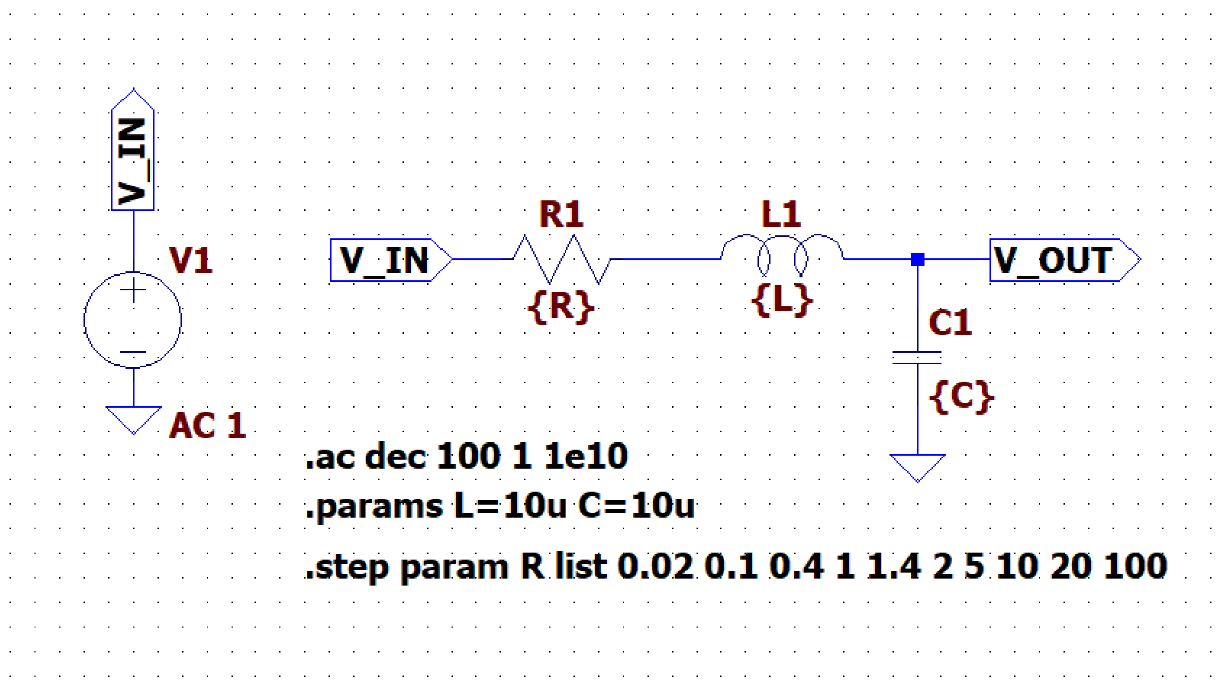
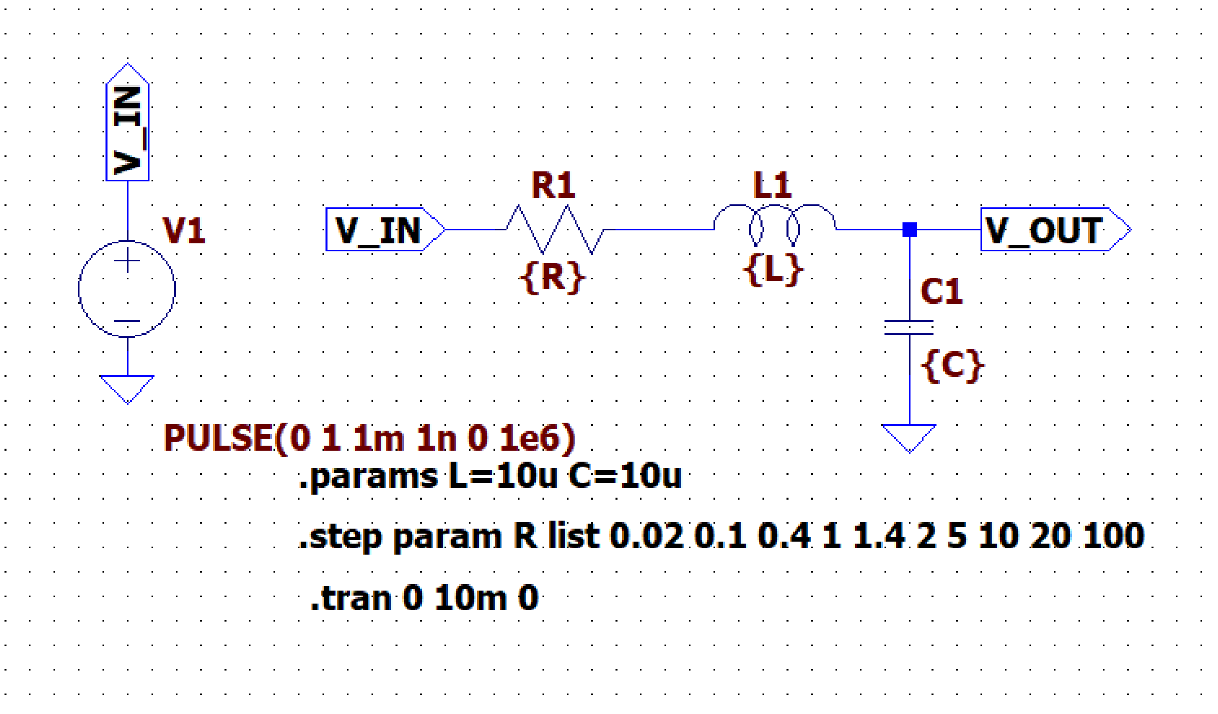
Understanding 2nd order RLC Low-Pass Filter and Two Stage Opamp

- Aadarsh Ramachandiran ee23b001

2nd order RLC Low-Pass Filter:

- The following LTSpice schematics detail the circuit diagram for analyzing Transfer Function and Step Response of a standard RLC low-pass filter.





- We analyze the differences in the plots for different values of R.

- This code is used to tabulate the different parameter results as R varies.

```
def analyze_RLC(R, L, C):
    '''Find damping factor, quality factor, natural frequency and
poles for given RLC system'''
    from cmath import sqrt
    z = 0.5 * R * sqrt(C / L)
    Q = 1 / (2 * z)
    w = 1 / sqrt(L * C)
    p1 = w * (-z + sqrt(z**2 - 1))
    p2 = w * (-z - sqrt(z**2 - 1))
    return z, Q, w, p1, p2
```

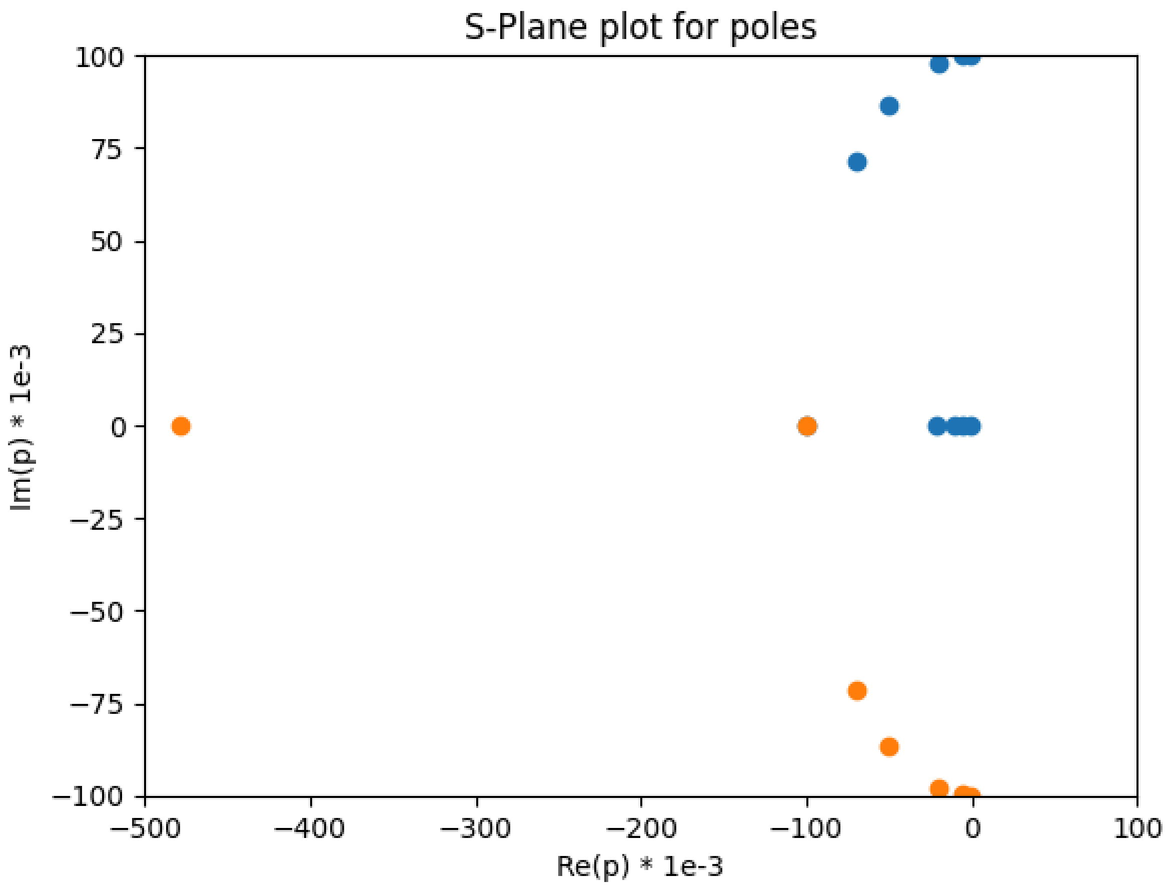
```python
from tabulate import tabulate
L = 10 * 10**(-6)
C = 10 * 10**(-6)
R_list = [0.02, 0.1, 0.4, 1, 1.4, 2, 5, 10, 20, 100]
results = []
for R in R_list:
    z, Q, w, p1, p2 = analyze_RLC(R, L, C)
    results.append([R, z, Q, w, p1, p2, p1.imag, p2.imag])
print(tabulate(results, headers = ["R", "z", "Q", "w", "p1", "p2", "wp1",
"wp2"]))
```

- The results are as follows:

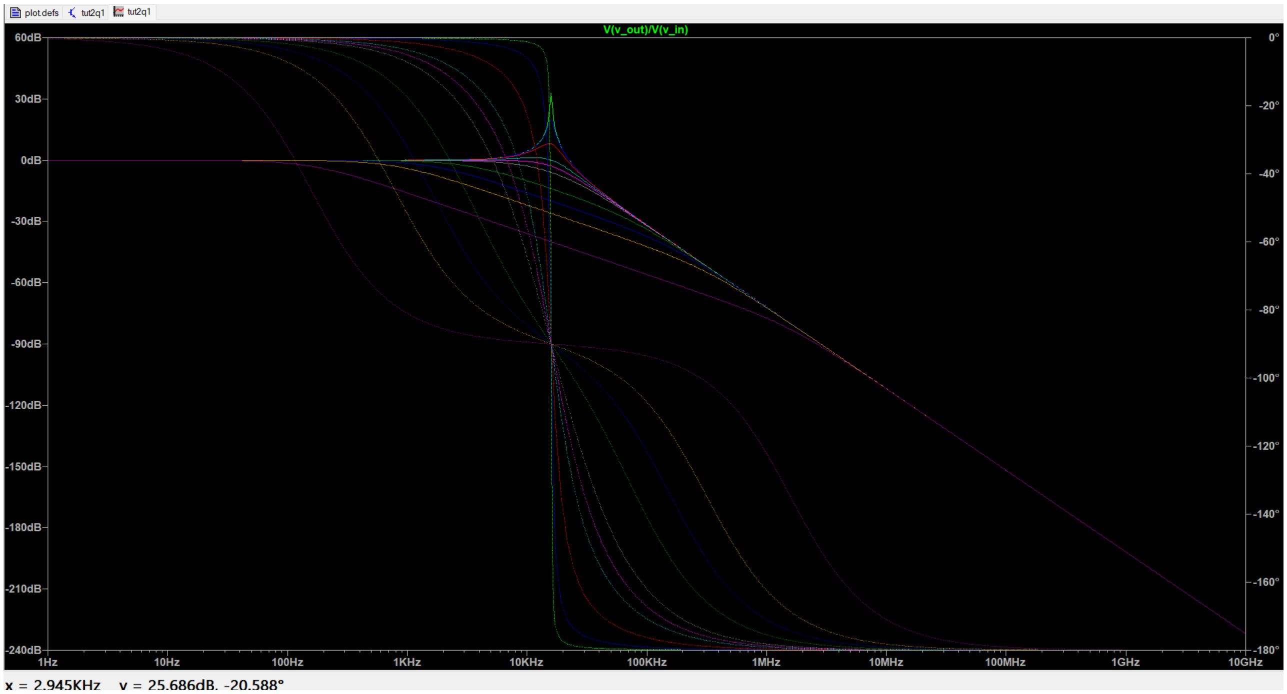
R	z	Q	w	p1	p2	wp1	wp2
0.02	(0.01+0j)	(50+0j)	(100000.00000000001+0j)	(-1000.0000000000001+99994.99987499377j)	(-1000.0000000000001-99994.99987499377j)	99995	-99995
0.1	(0.05+0j)	(10+0j)	(100000.00000000001+0j)	(-5000.000000000001+99874.9217771909j)	(-5000.000000000001-99874.9217771909j)	99874.9	-99874.9
0.4	(0.2+0j)	(2.5+0j)	(100000.00000000001+0j)	(-20000.000000000004+97979.58971132713j)	(-20000.000000000004-97979.58971132713j)	97979.6	-97979.6
1	(0.5+0j)	(1+0j)	(100000.00000000001+0j)	(-50000.00000000001+86602.54037844387j)	(-50000.00000000001-86602.54037844387j)	86602.5	-86602.5
1.4	(0.7+0j)	(0.7142857142857143+0j)	(100000.00000000001+0j)	(-70000+71414.28428542851j)	(-70000-71414.28428542851j)	71414.3	-71414.3
2	(1+0j)	(0.5+0j)	(100000.00000000001+0j)	(-100000.00000000001+0j)	(-100000.00000000001-0j)	0	-0
5	(2.5+0j)	(0.2+0j)	(100000.00000000001+0j)	(-20871.21525220801+0j)	(-479128.784747792-0j)	0	-0
10	(5+0j)	(0.1+0j)	(100000.00000000001+0j)	(-10102.051443364426+0j)	(-989897.9485566358-0j)	0	-0
20	(10+0j)	(0.05+0j)	(100000.00000000001+0j)	(-5012.562893380058+0j)	(-1994987.4371066205-0j)	0	-0
100	(50+0j)	(0.01+0j)	(100000.00000000001+0j)	(-1000.1000200048794+0j)	(-9998999.899979997-0j)	0	-0

- As R increases damping increases.
- As damping increases poles go further apart.
- For damping factor < 1, complex conjugate poles exist. Causes ringing.
- System is unconditionally stable- real part of pole is always in left-half plane (<=0).

- The behaviour of poles as damping factor varies can be realised from the locus plot :

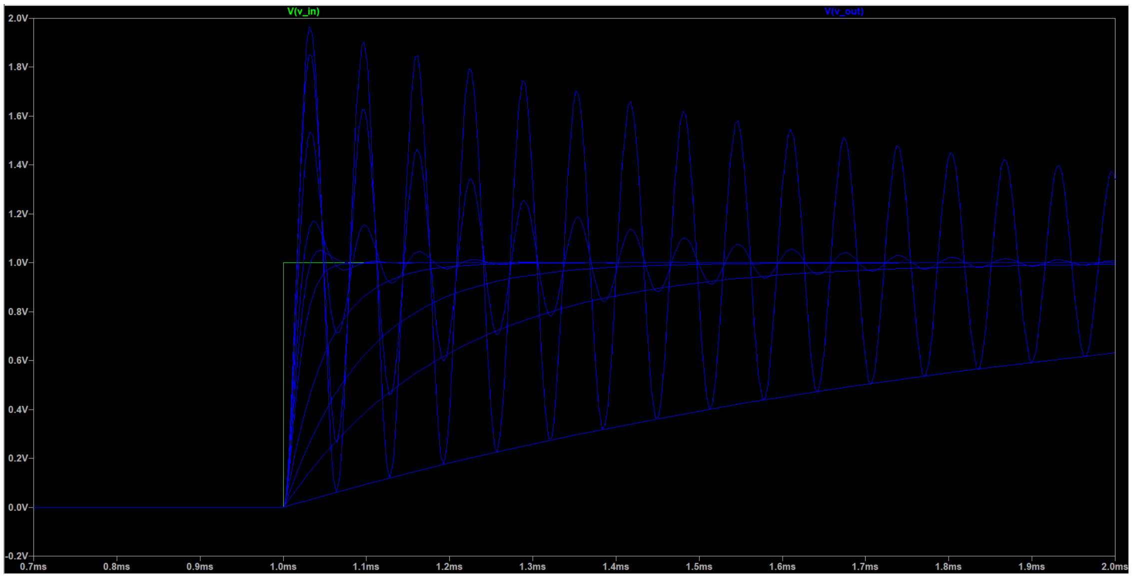


- Variation of AC magnitude and Phase response with damping.



- As damping factor increases the peak is reduced and leads to gradual roll-off. Phase are more abrupt for smaller damping factors.

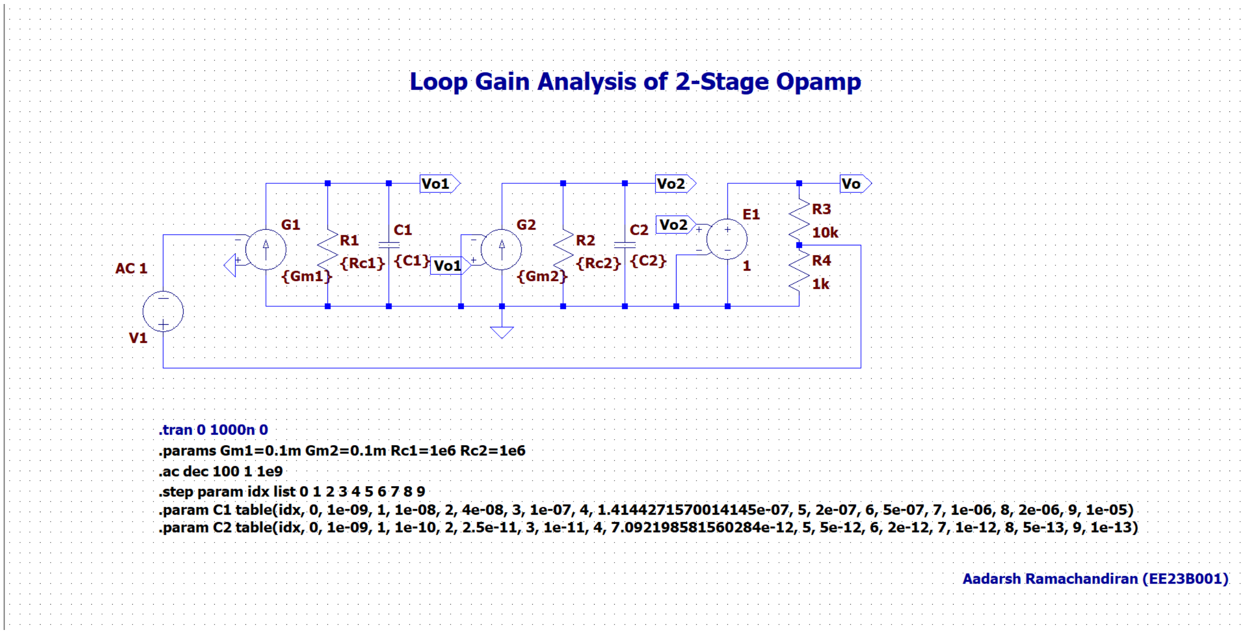
- The step response for varies damping factor values look as follows:

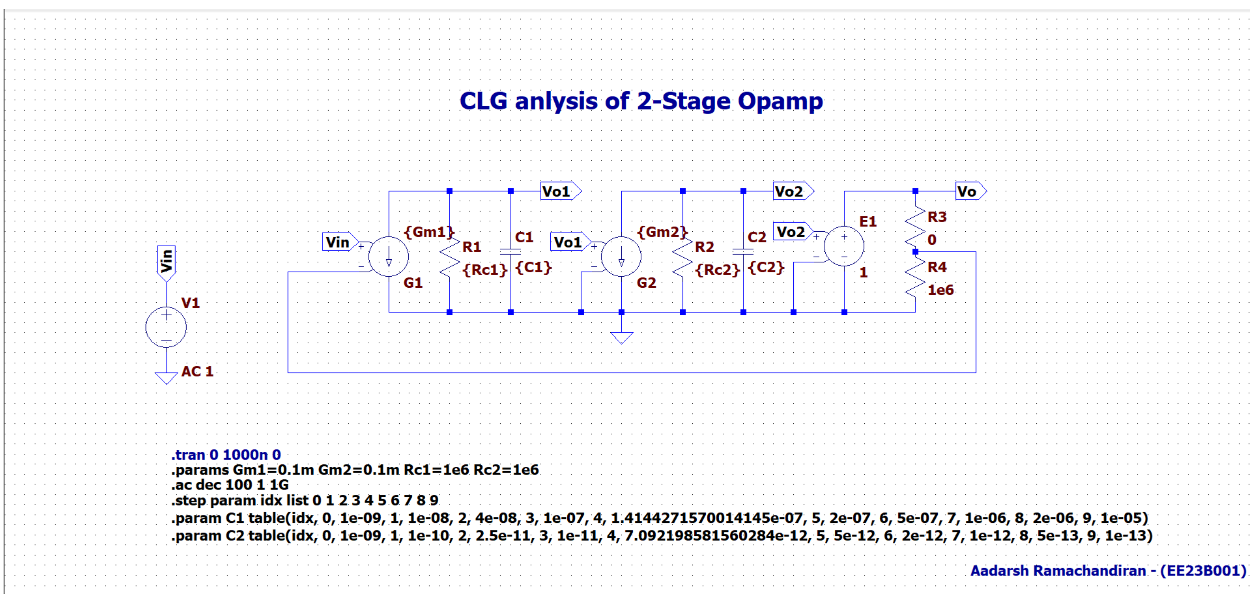


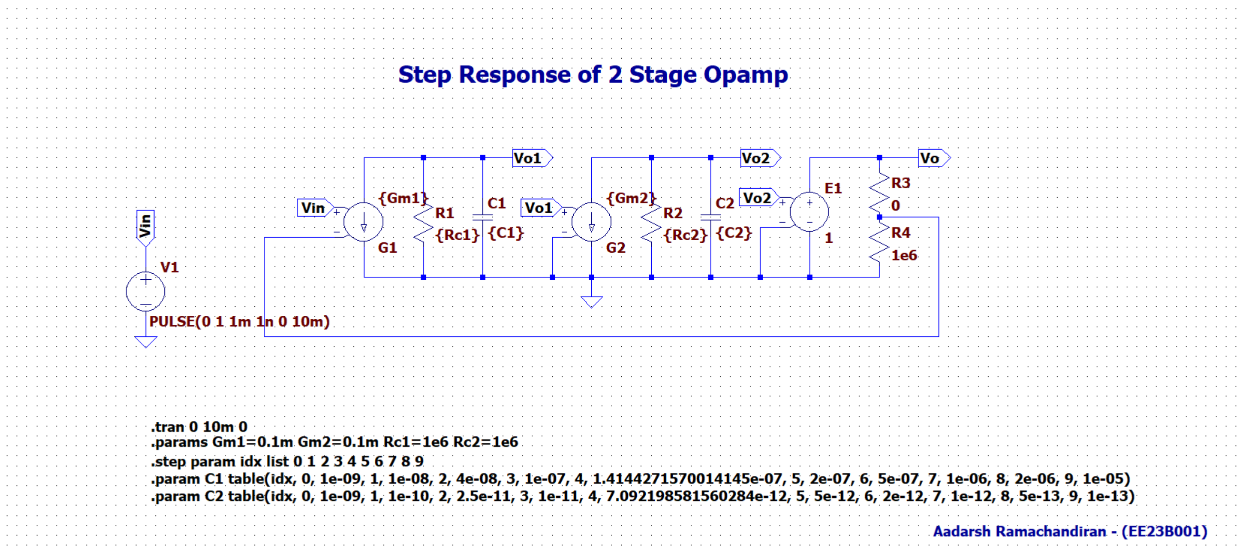
- (For some reason, my LTSpice software doesn't seem to differentiate these with different colours or provide a legend)
- As damping increases, there is less ringing.
- At critical damping (damping factor = 1), the settling time is the fastest.

2-stage Opamp

- The LTSpice schematics for loop gain, closed loop gain and step response analysis are detailed below:







- The following code is used to simulate values of various parameters of the opamp as poles of the gain vary.

```
import cmath

def analyze_opamp(wp1, wp2, A):
    '''Find loop gain and closed loop parameters of opamp'''
    w = cmath.sqrt(A * wp1 * wp2)
    z = (wp1 + wp2) / (2 * cmath.sqrt(A * wp1 * wp2))
    Q = 1/(2 * z)
    p1 = w * (-z + cmath.sqrt(z**2 - 1))
    p2 = w * (-z - cmath.sqrt(z**2 - 1))
    wp1cl = p1.imag
    wp2cl = p2.imag
    wugf = (0.5 * (-(wp1**2 + wp2**2) + cmath.sqrt((wp1**2-wp2**2)**2 + 4 * A**2 * wp1**2 * wp2**2)))**0.5
    pm = (cmath.atan(-wugf * (wp1 + wp2) / ((wp1*wp2) - (wugf**2))) * 180 / cmath.pi)

    return [wp1, wp2, wugf, pm, z, Q, wp1cl, wp2cl, p1, p2]
```

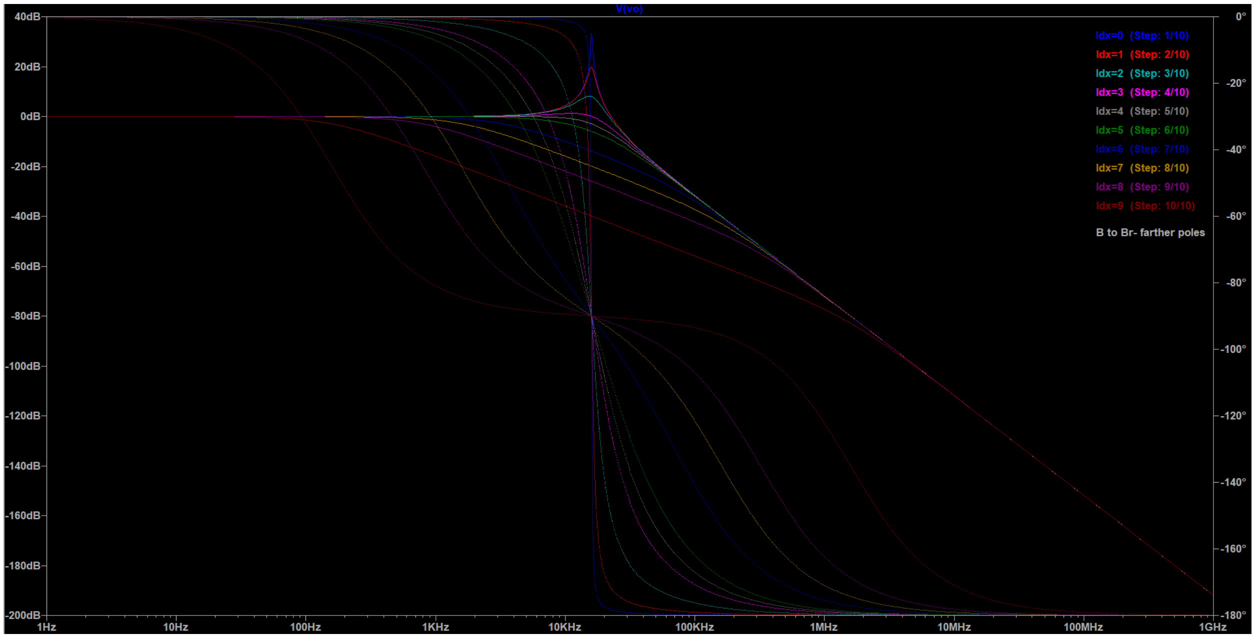
```python
from tabulate import tabulate

wp1_list = [1e3, 1e2, 2.5e1, 1e1, 7.07e0, 5e0, 2e0,1e0,5e-1,1e-1]
wp2_list = [1e3,1e4,4e4,1e5,1.41e5,2e5,5e5,1e6,2e6,1e7]
A = 1e4
results = []

for i in range(len(wp1_list)):
    wp1 = wp1_list[i]
    wp2 = wp2_list[i]
    results.append(analyze_opamp(wp1, wp2, A))
print(tabulate(results, headers = ["wp1", "wp2", "wugf", "PM", "z", "Q",
"wp1cl", "wp2cl", "p1cl", "p2cl"]))
```

- The results are as follows:

wp1	wp2	wugf	PM	z	Q	wp1cl	wp2cl	p1cl	p2cl
1000	1000	(99994.99987499375+0j)	(1.1459346897143052+0j)	(0.01+0j)	(50+0j)	99995	-99995	(-1000+99994.99987499375j)	(-1000-99994.99987499375j)
100	10000	(99750.2880909139+0j)	(5.7822332209242+0j)	(0.0505+0j)	(9.900990099009901+0j)	99872.4	-99872.4	(-5050+99872.40609898212j)	(-5050-99872.40609898212j)
25	40000	(96083.02985318055+0j)	(22.617192368618888+0j)	(0.200125+0j)	(2.4984384759525295+0j)	97977	-97977	(-20012.5+97977.03732890681j)	(-20012.5-97977.03732890681j)
10	100000	(78615.1373155213+0j)	(51.834580671696266+0j)	(0.50005+0j)	(0.9999000099990001+0j)	86599.7	-86599.7	(-50005+86599.65343464141j)	(-50005-86599.65343464141j)
7.07	141000	(64322.96951678045+0j)	(65.48423984049565+0j)	(0.7061413272826799+0j)	(0.7080735550828933+0j)	70696.2	-70696.2	(-70503.535+70696.1919236374j)	(-70503.535-70696.1919236374j)
5	200000	(48586.826931973636+0j)	(76.351311545624+0j)	(1.000025+0j)	(0.4999875003124922 3+0j)	0	-0	(-99295.38879941033+0j)	(-100709.61120058966-0j)
2	500000	(19984.04453184818+0j)	(87.71694955343823+0j)	(2.50001+0j)	(0.19999920000319998+0j)	0	-0	(-20871.12416317249 5+0j)	(-479130.8758368275 6-0j)
1	1e+06	(9999.500037483564+0j)	(89.43281980959843+0j)	(5.000005+0j)	(0.0999999000001+0j)	0	-0	(-10102.04113301194+0j)	(-989898.958866988 1-0j)
0.5	2e+06	(4999.984350170821+0j)	(89.86249089382912+0j)	(10.0000025+0j)	(0.04999998750000312+0j)	0	-0	(-5012.56163392653 8+0j)	(-1994987.9383660736-0j)
0.1	1e+07	(999.9999882812499+0j)	(-89.99999999986571+0j)	(50.0000005+0j)	(0.0099999999+0j)	0	-0	(-1000.100010001858 7+0j)	(-9998999.99989998-0j)

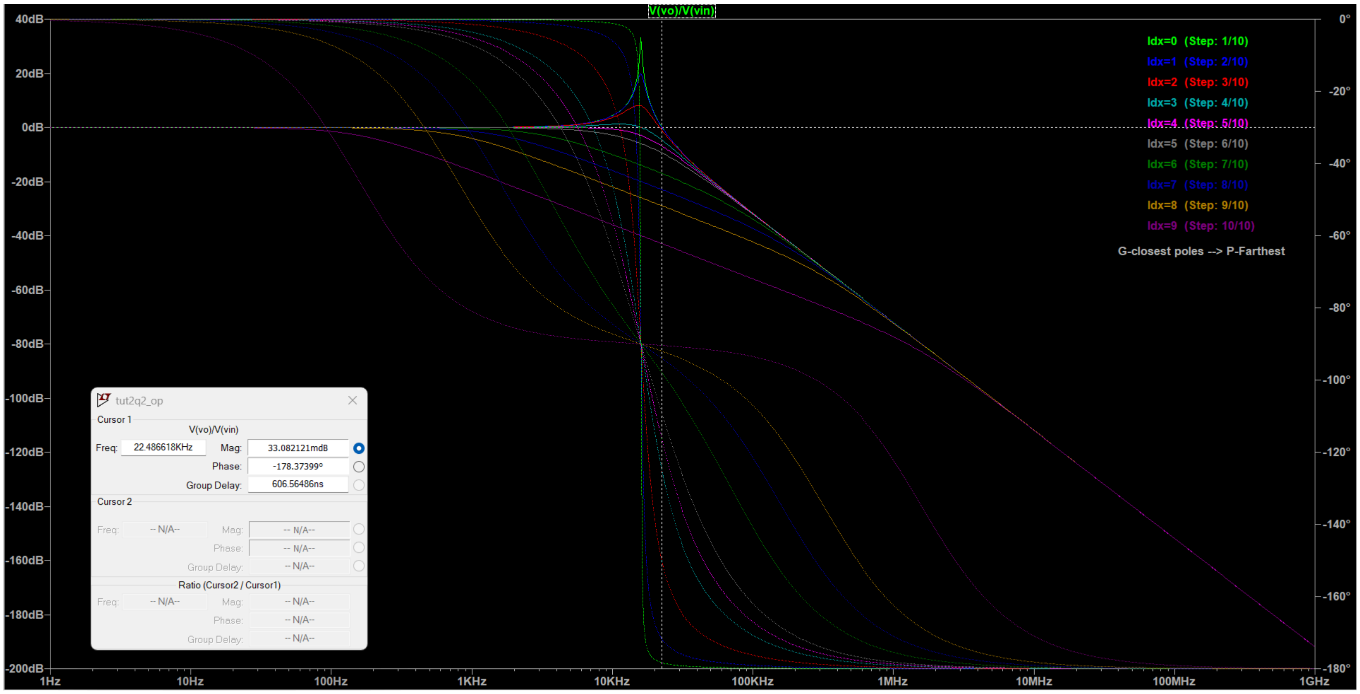
- The characteristics are similar to the previous RLC circuit.
- As poles go further apart, damping factor increases.
- For damping factor < 1, complex conjugate poles exist. Causes ringing.
- System is unconditionally stable- real part of pole is always in left-half plane (<=0).

- Loop gain is analyzed for different damping factors. The following is the AC magnitude and phase plot.



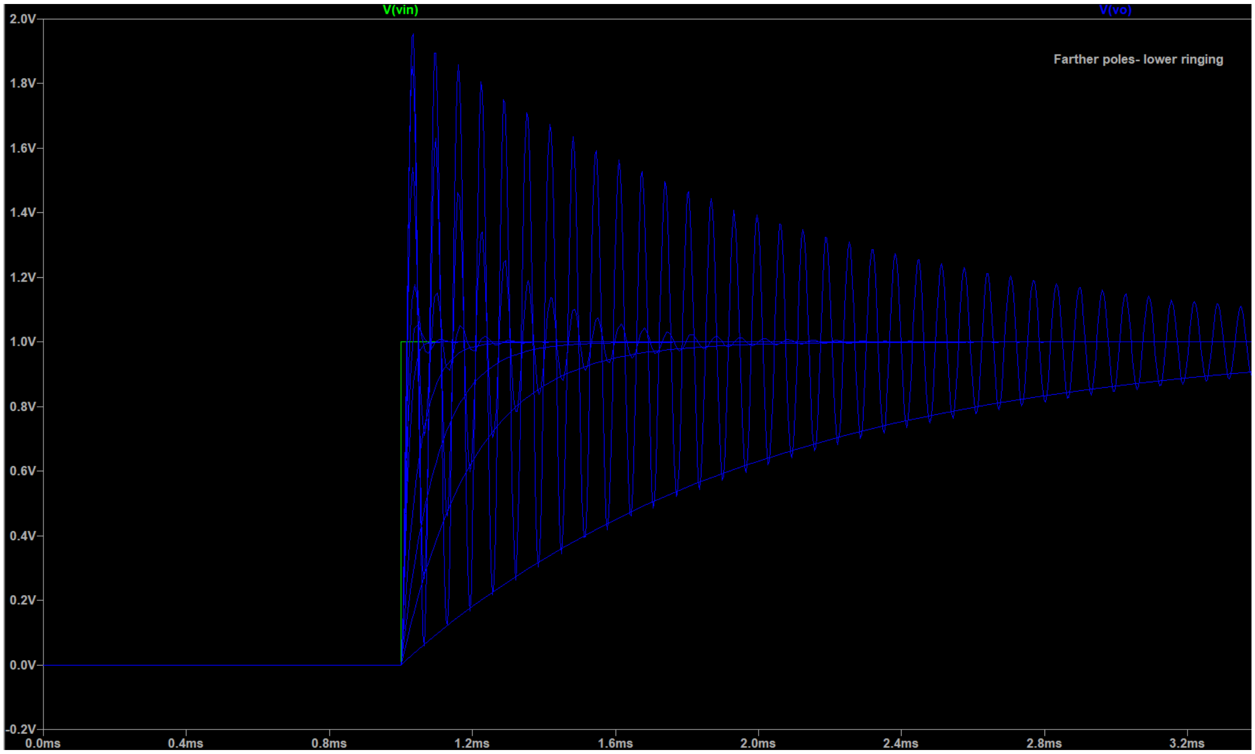
- As poles go farther, the peak is decreased and leads to gradual roll-off. Phase margin increases.

- The closed loop gain is analyzed for different damping factors.

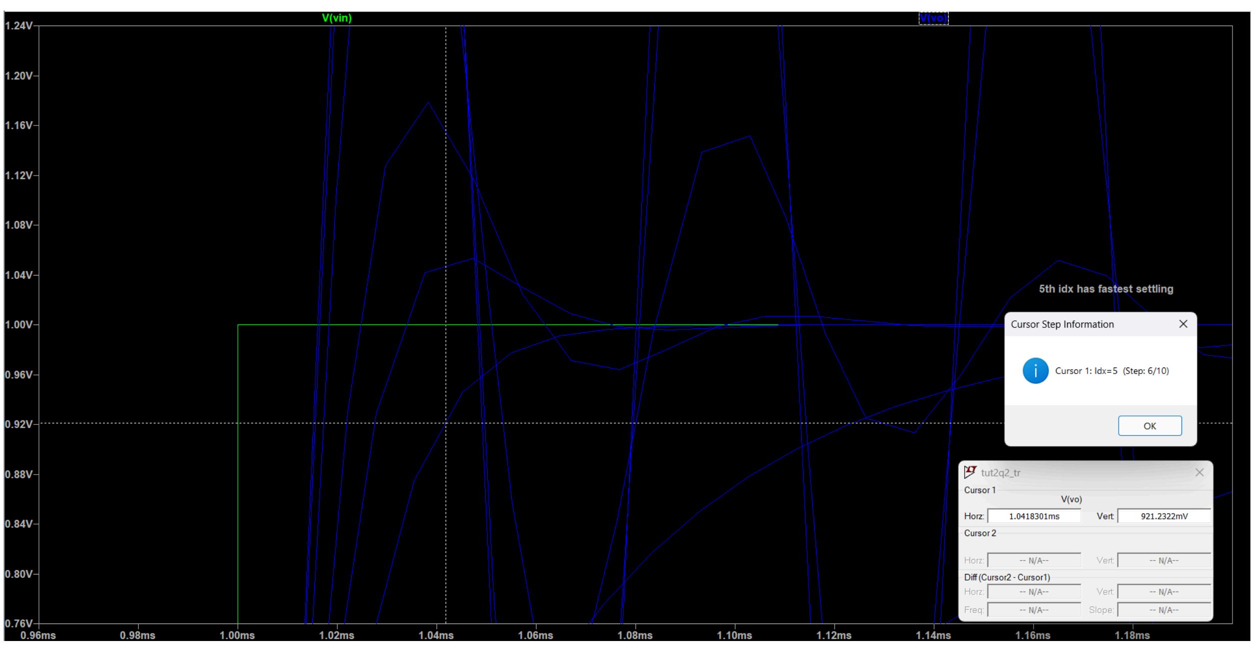


- As poles go farther, peak in magnitude plot decreases. Phase changes gradually.

- The step response is analyzed for different damping factors.



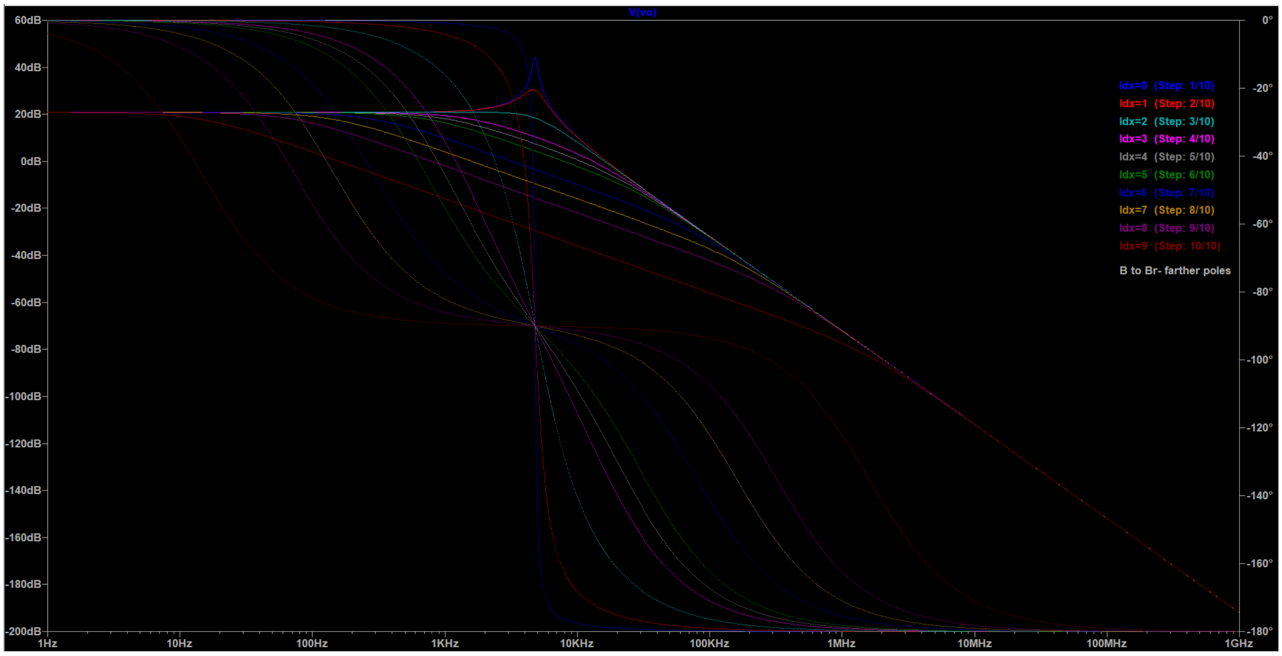
- As damping factor increases, less ringing.
- Fastest settling occurs when damping factor is close to 1



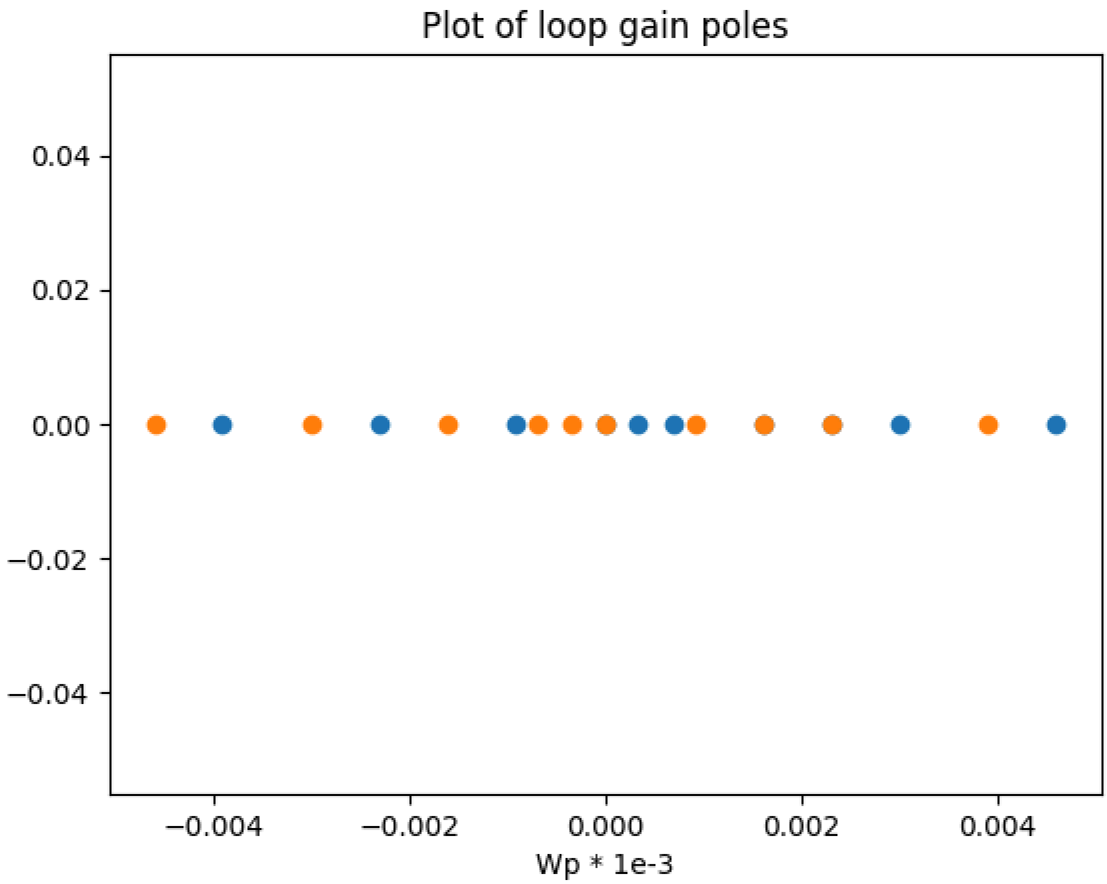
- We also analyze the effect of the gain of opamp on closed loop gain and loop gain.

- Increasing beta increases damping factor. Unity gain frequency decreases.

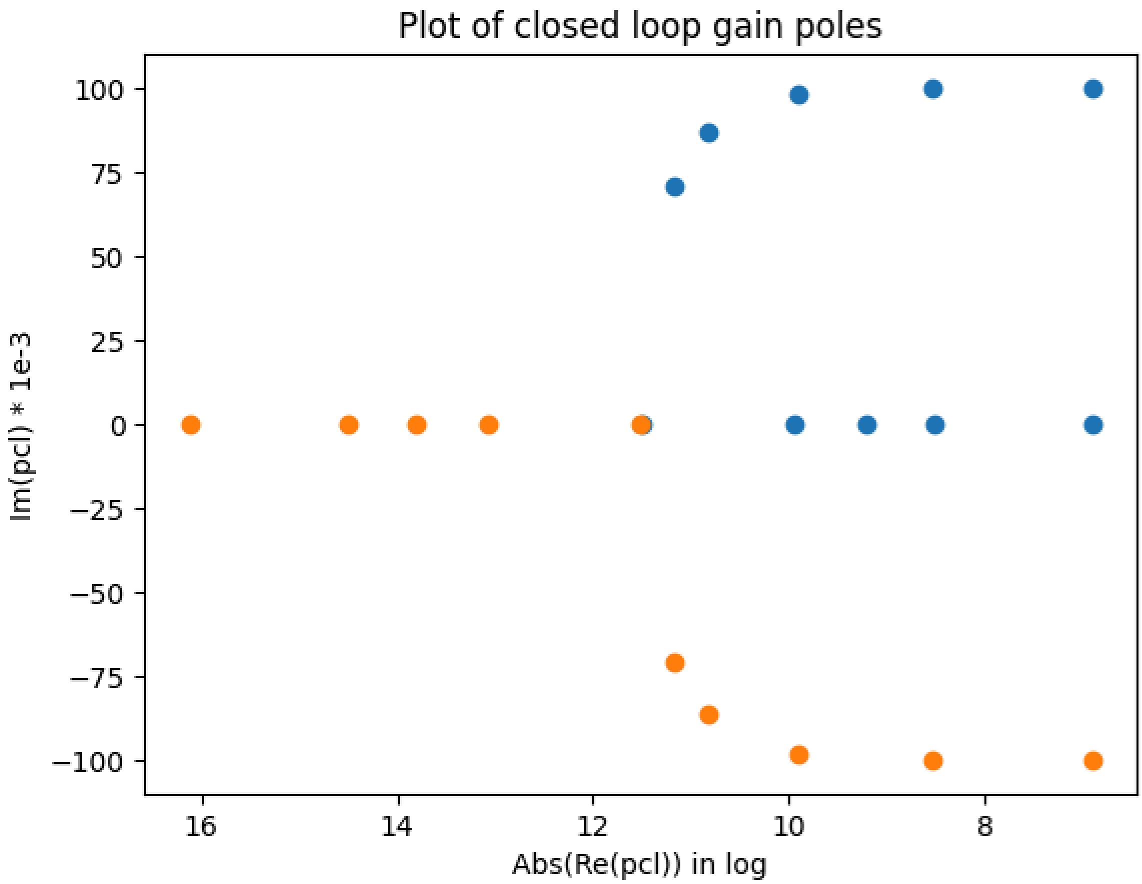
- As the following plot shows, the phase margin increases and unity gain frequency of loop gain also decreases.

- 

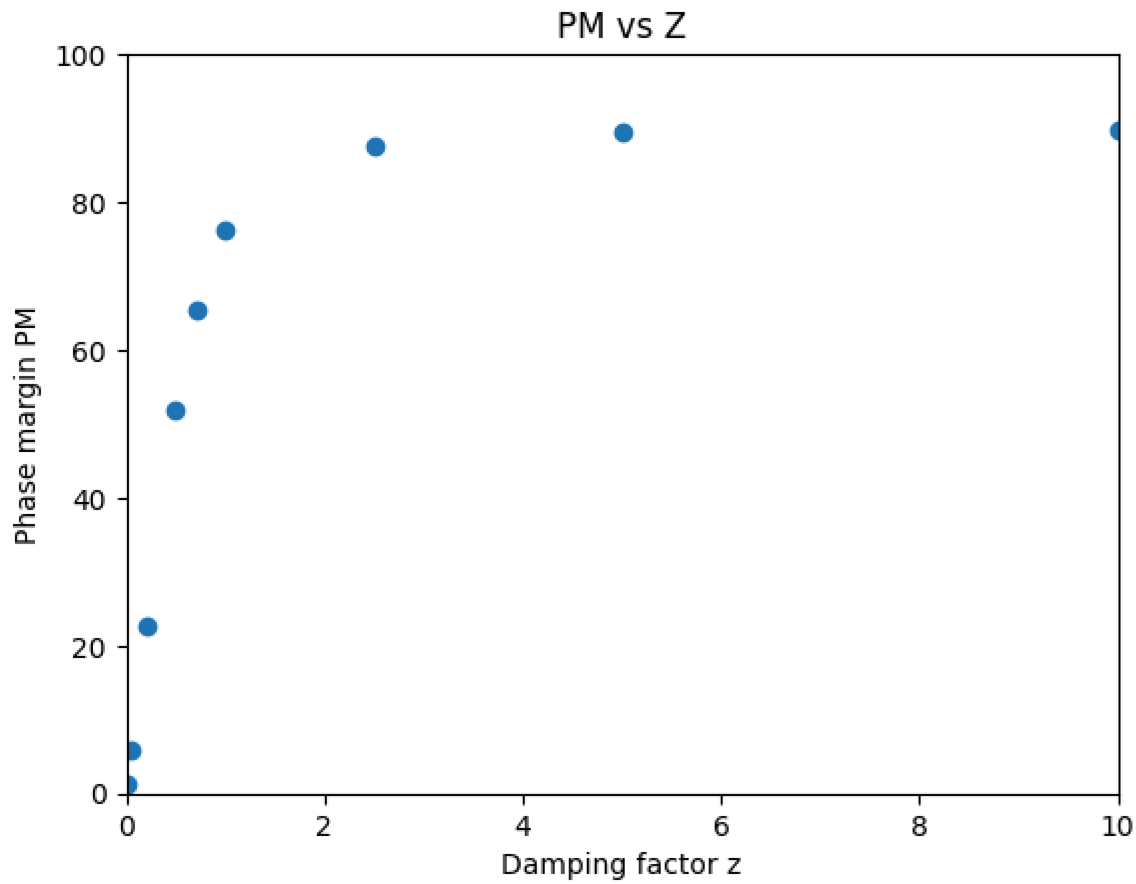
- To understand the table better we plot the poles of the loop gain and the closed loop gain.



- As damping factor decreases, the loop gain poles go closer together.



- As damping factor increases, loop gain poles go closer and same with closed loop gain poles which later split into complex conjugate pairs.



- The phase margin increases with the damping factor. Note that we can approximate a linear relationship for lower values of damping factor.

Conclusion

Hopefully, this helps in understanding the characteristics better through plots and simulation. More insights can be gained from these as understanding of the core concepts improve.