Submitted By: Group 15

Harsh Kadian (2017eeb1142)

Mrityunjay (2017eeb1153)

Pradeep (2017eeb1159)

SIMULATION AND CONTROL DESIGN ON SIMULINK AS A CAD INTERFACE TO MATLAB

OBJECTIVE

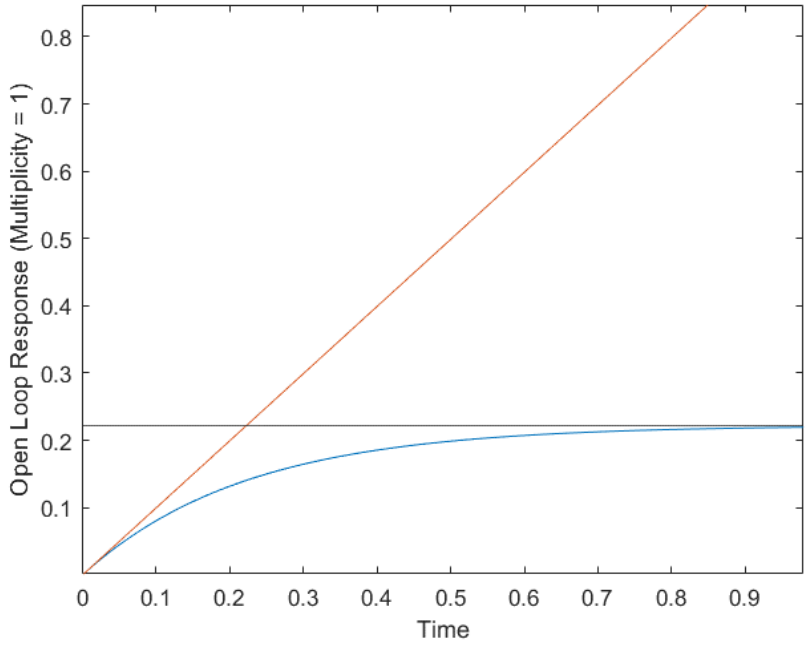
To analyse the effectiveness of Ziegler Nichols rules in the presence of poles with high multiplicity

TRANSFER FUNCTION AND ITS RESPONSE IN OLTF TO A UNIT STEP

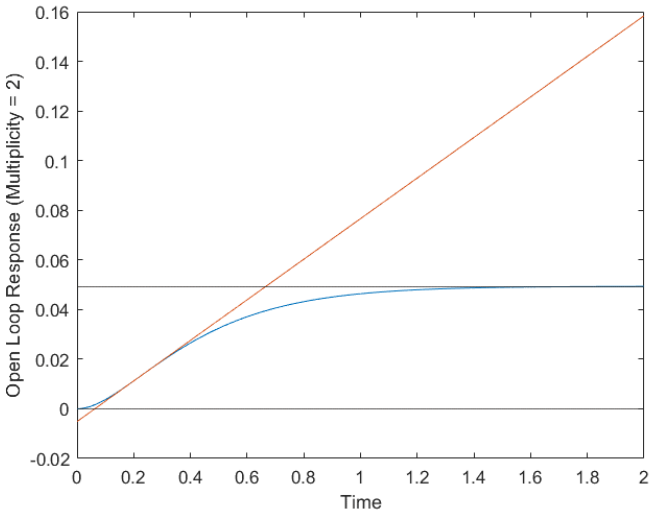
The transfer function under investigation is a first-order system expressed as: $\frac{1}{s+4.5}$

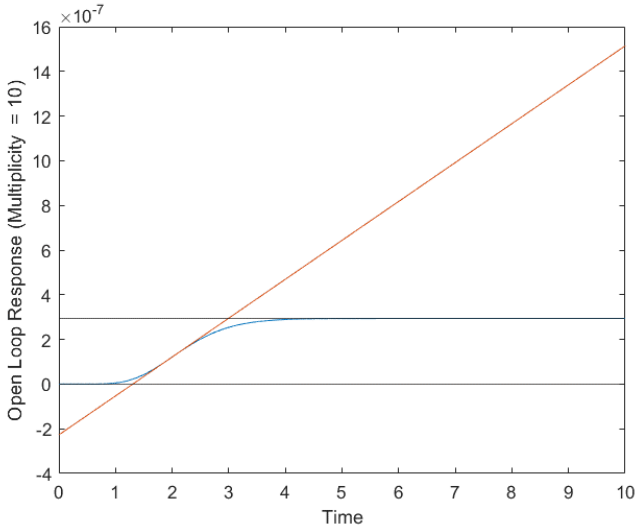
We would analyse the behaviour of this transfer function initially in Open Loop to find the required parameters for ZN rules while increasing the multiplicity of poles. Later on, to assess the performance metrics for the respective systems in Closed Loop controlled with P, PI and PID controllers designed based on Ziegler Nichols rules.

Beginning with multiplicity = 1, for a step input, the system would behave ideally (with no lag) with an exponential behaviour and eventually settles to 1 given the sufficient time.

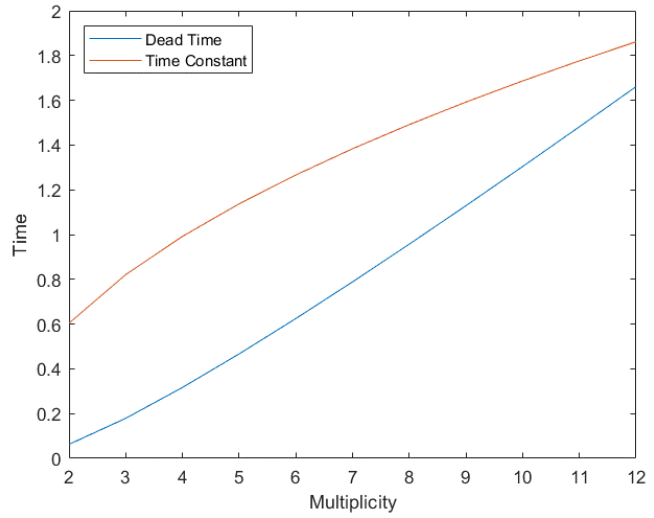


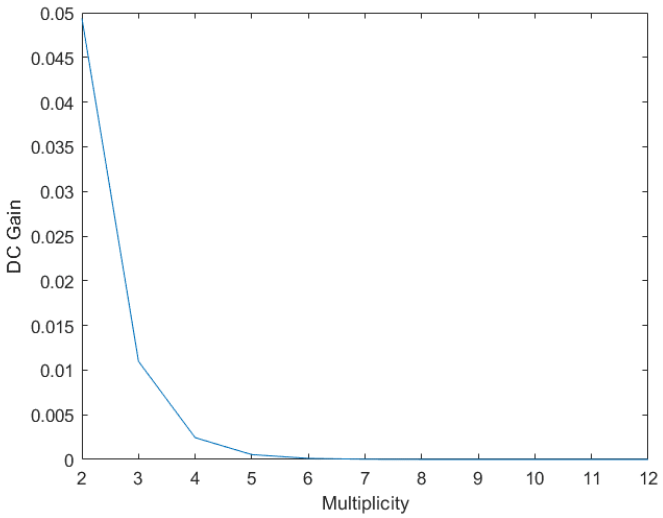
However, as one would keep on increasing the multiplicity, the dead time of the system would start to appear and would increase as the multiplicity increases. One of the reasons being as the number of poles increases, it tends to slow down the system.





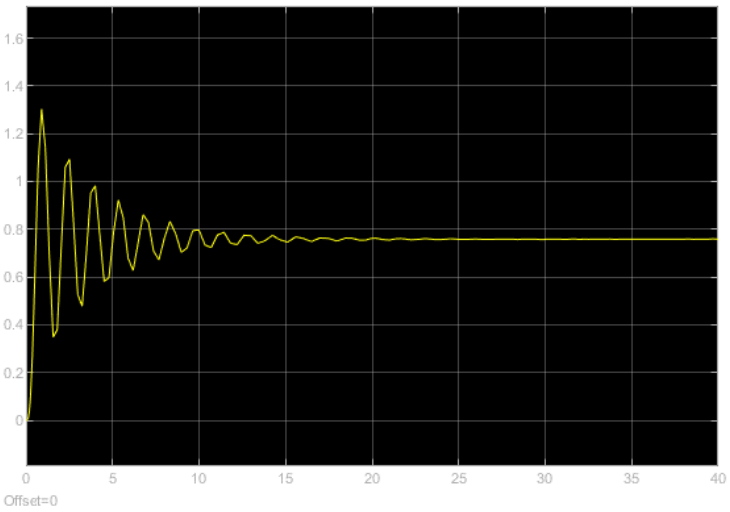
The variation of Dead Time and Time Constant as for Ziegler Rules is as shown below. Apart from that, on increasing the multiplicity of the pole, the DC gain decreases exponentially which is obvious. However this means while designing the controller using Ziegler Nichols rules, the proportional gain would increase exponentially too, which could limit the realizability of controllers to some extent as well as could render the system unstable.



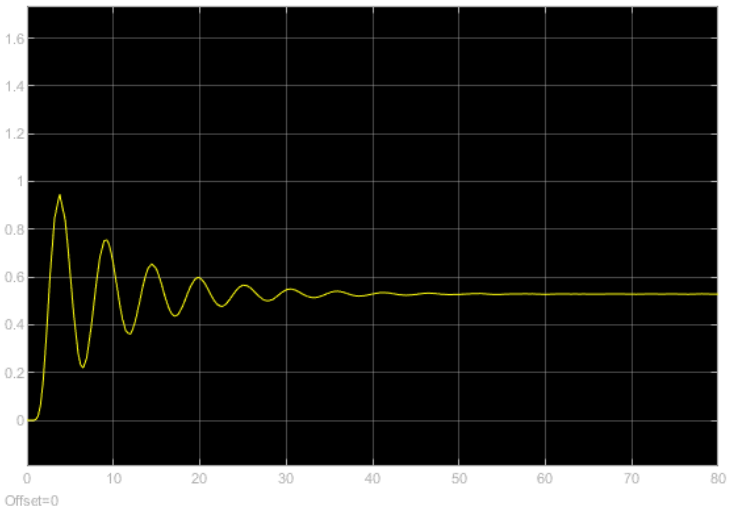


P CONTROLLER

- Oscillation Frequency: Decreases with the increase of multiplicity
- Steady State Error: Increases. The system has even shown a steady-state error of approximately 50% for multiplicity as high as 12
- Settling Time: Increases



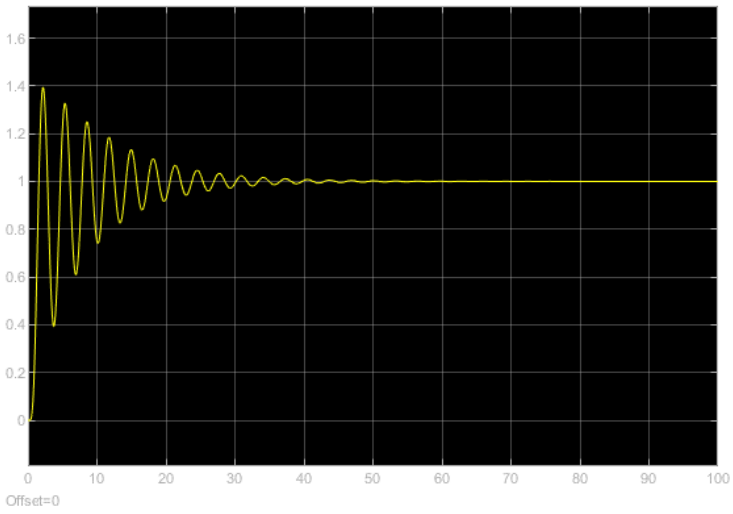
CL response for Multiplicity = 4



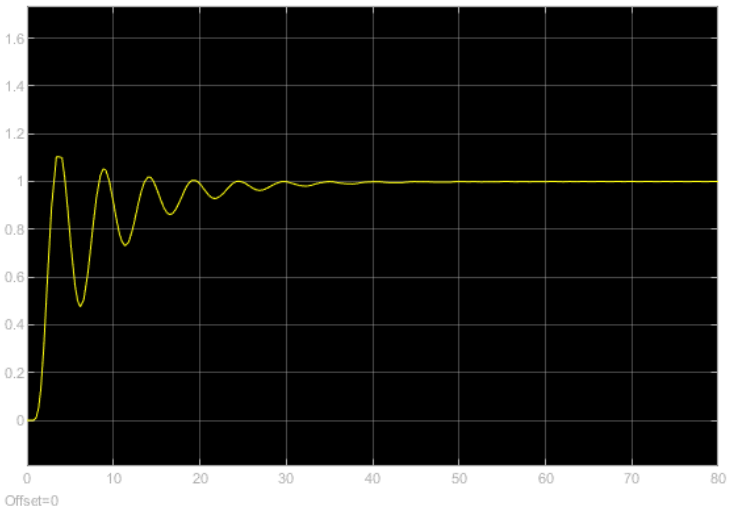
CL response for Multiplicity = 12

PI CONTROLLER

- Oscillation Frequency: It decreases with the increase of multiplicity
- Overshoot: Decreases as multiplicity increases



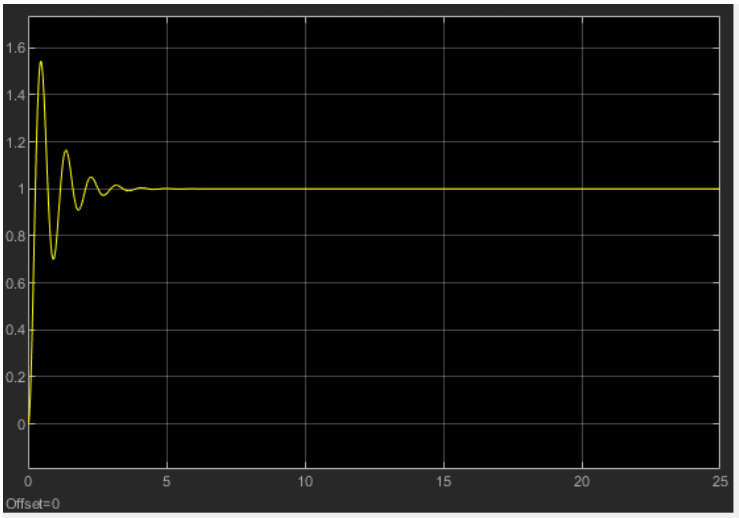
CL response for Multiplicity = 7



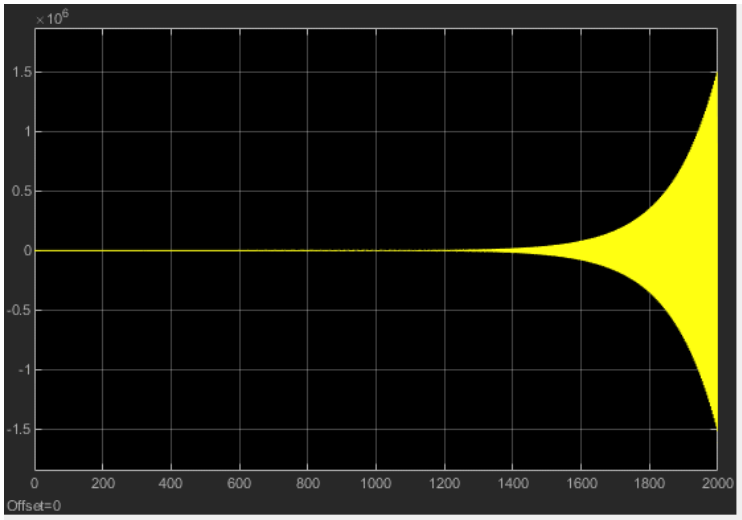
CL response for Multiplicity = 11

PID CONTROLLER

- Settling Time: It increases exponentially with increasing multiplicity
- Stability: The system becomes unstable and keeps on oscillating when the multiplicity> 9
- Oscillation frequency: Increases.



CL response for Multiplicity = 3

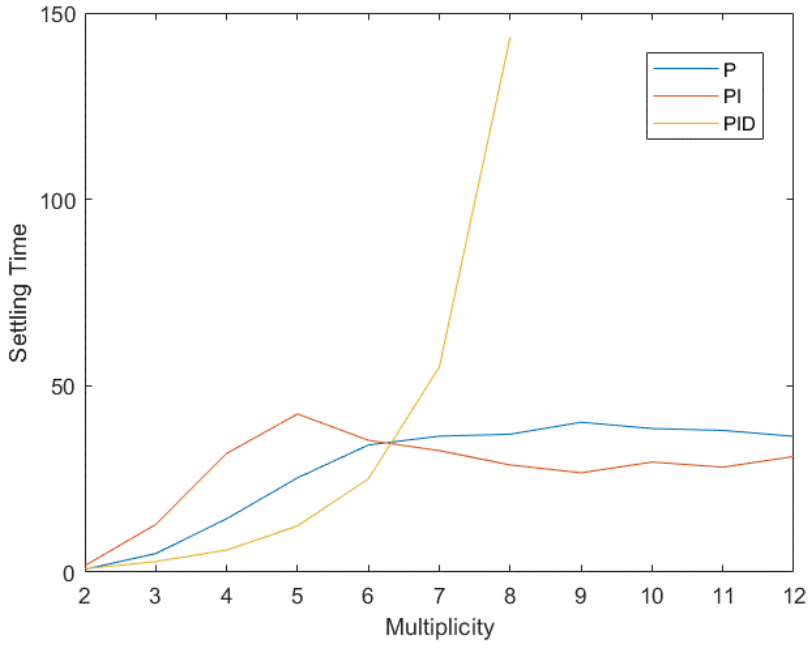


CL response for Multiplicity = 10

PERFORMANCE ANALYSIS OF CONTROLLERS

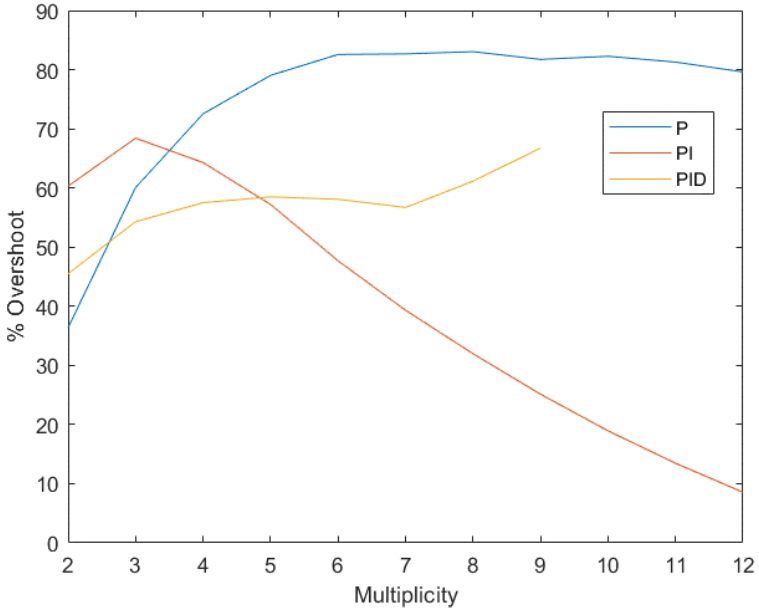
SETTLING TIME

As could be seen, PID works best in terms of settling time among all three controllers for low multiplicities however as the multiplicity of poles goes beyond 6, its performance deteriorates, and after a few more it becomes unstable too. The settling time for PID increases exponentially.



Apart from that, it can also be inferred that for higher multiplicities PI controller has a slightly better performance.

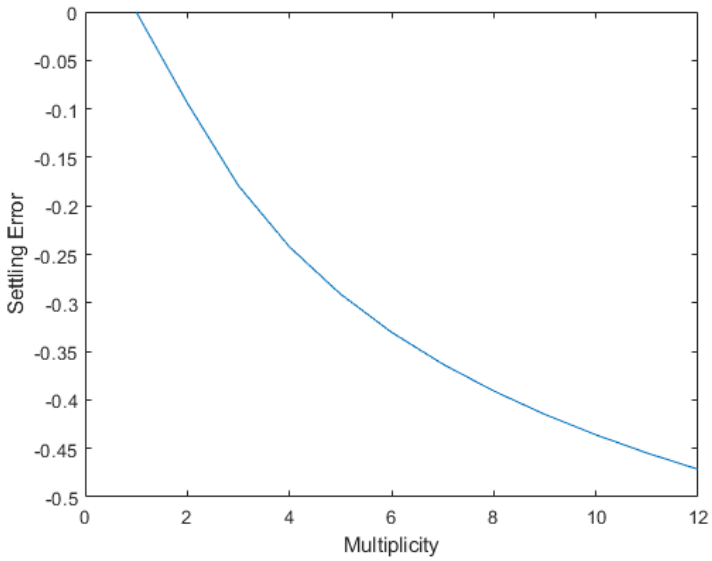
% Overshoot



Although PI controller has overshoots as high as 60% for low values, it keeps on decreasing for higher values whereas overshoot of the other two controllers keeps on increasing which points out their limitations.

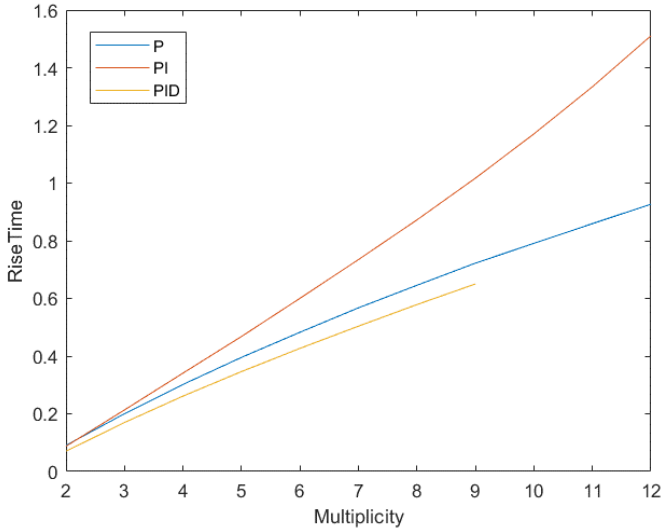
Contrarily P begins with lowest overshoot but increases and later on saturates to 80%, which is a high value in itself. Perhaps PID seems to work better on average, but we should not forget that at the multiplicity of 10, it becomes unstable.

Steady State Error



Of all the controllers, only P controller shows the Steady-state error, and that too keeps on increasing with multiplicity. The errors in case of other two controllers were as small as 10^{-6} and thus could be safely neglected.

Rise Time



Rise time governs how responsive the system is and how fast it reaches its maximum value. It is visible that PID controller is the fastest over the low values of multiplicities. Moreover, for high multiplicities, PI is approximately twice as slow as P controller.

Conclusion

As we observed over various parameters, the performance of PID controller is best for low multiplicities as it offers low rise time, comparably low overshoot, negligible steady state error and least settling time. But as one moves to the systems with higher multiplicities, more than 6, ZN rules fail for PID tuning. Therefore, ZN tuning incorporated PI and P controller could serve the purpose interchangeably keeping in mind the required set of properties you wish to incorporate in your system. Each has its pros and cons