School of Engineering and Technology

# Digital Control Systems

Unit Code: ENEE 20004

Term 1

# PROJECT DESIGN REPORT

# Load Frequency Control


Team Members:

-Sandesh Chhetri (12122731)

-Anand Sebastian (12124306)

-Bhaumikkumar Vishnubhai Panchal (12122925)


Week of Submission: Week 11

Date of Submission: 29/05/2020

Unit Co-ordinator: Kianoush Emami

# Contents

## List of Figures:

# 1.Project Overview

The main objective of the project is to design a Load Frequency Control for a single synchronous generator which supplies power to the load of the CQUeGen. CQUeGen is operated by the CQ University Australia. This project contract is awarded to the DCS Team 1 for an estimation of 35,453 AUD. The project was started on 3rd April 2020 and Project Report is finalized and submitted on 29th May 2020.

For this project we will be modeling our continuous time system and design a continuous PID controller for the system and use the tuned parameters from the continuous PID control to develop a discrete PID control with Zero Order Hold.

## 1.1 Theory and Working of the LFC System

The quality of electric power relies on the steadiness of voltage and frequency of the AC supply irrespective of the change in both active and reactive load demand from the consumer. In real world power system operation, the active and reactive power demand changes randomly and continuously. The relation between active power and frequency (rotor angle) and between reactive power and voltage makes it impossible to keep these parameters constant.

Active Power Demand $\longrightarrow$ Frequency

Reactive Power Demand $\longrightarrow$ Voltage

The project is to design a control system which is required to cancel the effects of the changes from the demand side and to keep the frequency and voltage in a permissible range. The main use of the LFC that we design is to control the frequency within the nominal range 50Hz(±2Hz) by ensuring that the injected active power from the generator matches the active power demand from the consumer side. In other words, the LFC has to tackle two situations:

1. If there is too much electricity supply from generator, the frequency will increase
2. If there is too much electricity demand from the consumer side, the frequency will decrease

**LFC Mechanism**

The LFC will read the supply frequency via a frequency sensor and will directly control the governor which controls the speed of the turbine and hence the speed of the generator coupled with the turbine.

- When the active load demand decreases suddenly, the generator shaft will run faster and the frequency increases. The LFC system will immediately detect the load variation and command the governor to decrease speed of the turbine, which counteracts the load decrease and brings the shaft speed at the rated value which results in bringing back the frequency to its nominal range.
- When the active load demand increases suddenly, the generator shaft will run slower and the frequency decreases. The LFC system will detects this situation and command the governor to increase the speed of the turbine, which counteracts the load increase and brings the shaft speed at the rated value and hence bringing back the frequency to its nominal frequency.

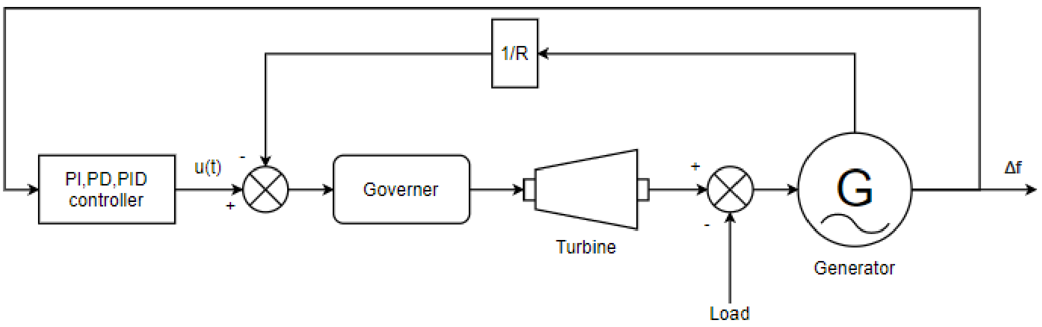



*Figure 1 Proposed System Model*

## 1.2 Task of DCS Team 1

- Find the Transfer Functions of Governor, Turbine and Generator
- Determine the gain $\frac{1}{R}$ that self-regulate the system without the controller
- Designing and Modelling of the system without the controller in MATLAB and Simulink with system performance check.
- Design of a digital controller (P, PI, PD, or PID) using the required parameters for the systems.
- Draft a Report detailing the design, calculations, process, and implementation of the system with the proposed controller.

## 2.Design Calculations

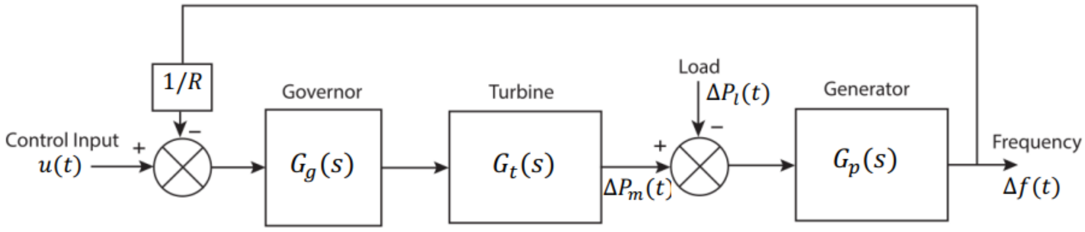



*Figure 2 System without Controller*

### 2.1 Transfer Function of Generator $G_P(s)$

In this project the set of value given by the client are:

Dumping Constant, $D=6*10^{-3}$ Inertia Constant, H= 21 frequency of nominal system=50 Hz

$\Delta P_m$ = mechanical power change acting on the shaft of the rotor

$\Delta P_L$ = load demand change (per-unit)

The swing equation of the generator is as follows:

$$\Delta f = \frac{f}{2*H}(\Delta P_m - \Delta P_L - D\Delta f) \qquad (1)$$

Here $\Delta P_m$ (t) - $\Delta P_L$ (t) is the input of the Generator and $\Delta f$ (t) is the output.

Then, $G_P (s) = \frac{\Delta F(s)}{\Delta Pm (s) - \Delta Pl (s)}$

Converting equation 1 to Laplace Transform

$$\Delta F (s) - \Delta f (0) = \frac{f}{2*H} (\Delta P_m (s) - \Delta P_L (s) - D\Delta F (s))$$

$$G_P (s) = \frac{\Delta F(s)}{\Delta Pm (s) - \Delta Pl (s)} = \frac{\frac{f}{2*H}}{s+\frac{D*f}{2*H}} = \frac{1.1905}{S+7.1428*10\text{^}-3}$$

The above given is the transfer function of the Generator.

## 2.2 Transfer Function of Turbine $G_t(S)$ And Governor $G_g(S)$

The step response of both Turbine and Governor is given by the client. To find the transfer function from the step response, we use System Identification Toolbox of MATLAB.

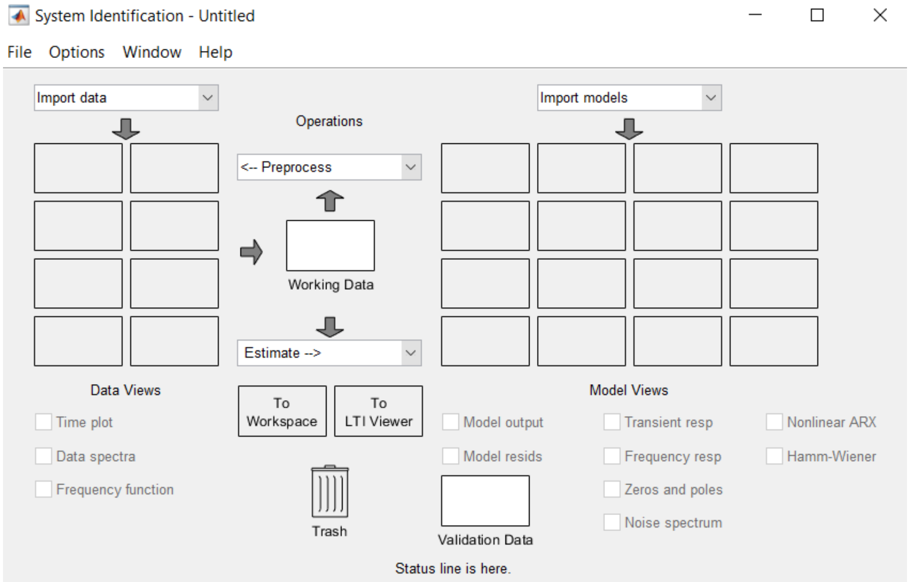



*Figure 3 System Identification Toolbox*

STEPS DONE:

- Enter the data given by the client in the MATLAB as a vector of output.
- Generate a vector of 1 in the MATLAB workspace and set it as system input.

  >> Governor_in=ones(size(Governor_out));

  >> Turbine_in=ones(size(Turbine_out));
- Import input and output data to the System Identification Toolbox by opening "Import data" drop down menu and select the "Time domain data". Set the starting time to 0 and sampling time which is given in the client data. A blue curve will be shown on the left side of the toolbox.

- Open the “Estimate” drop down menu and Select “Transfer Function models”.

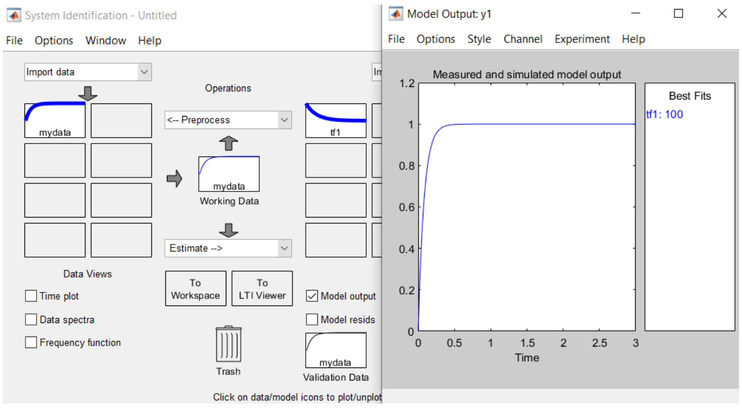



*Figure 4 Estimation of the Transfer Function using System Identification Toolbox*

- Choose the number of poles and zeros in a way that gives the best accuracy.

  Hence by using the System Identification Toolbox we generated the Transfer function of the Governor, $G_g$ (s) and Turbine $G_t$ (s)

Transfer function of the Governor, $G_g$ (s) = $\frac{12.5}{s + 12.5}$

Transfer function of the Turbine, $G_t$ (s) = $\frac{1.667\, s + 0.3334}{s^2 + 3.433\, s + 0.3334}$

We will be using these transfer function of generator, governor and turbine for our project ahead.

# 3.Droop Characteristics (1/R)

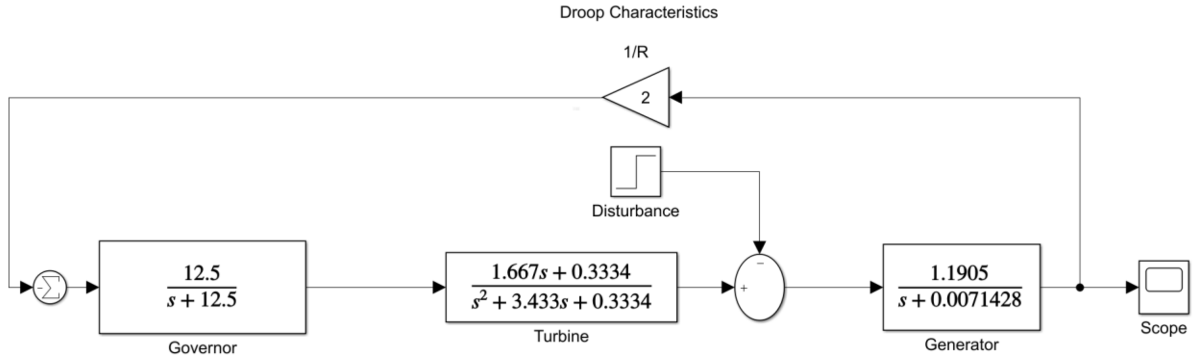



*Figure 5 Approximating the Droop characteristics*

Droop Characteristics in a synchronous AC generator system is used to control the speed of the prime mover that is governor in our system according to load demand.

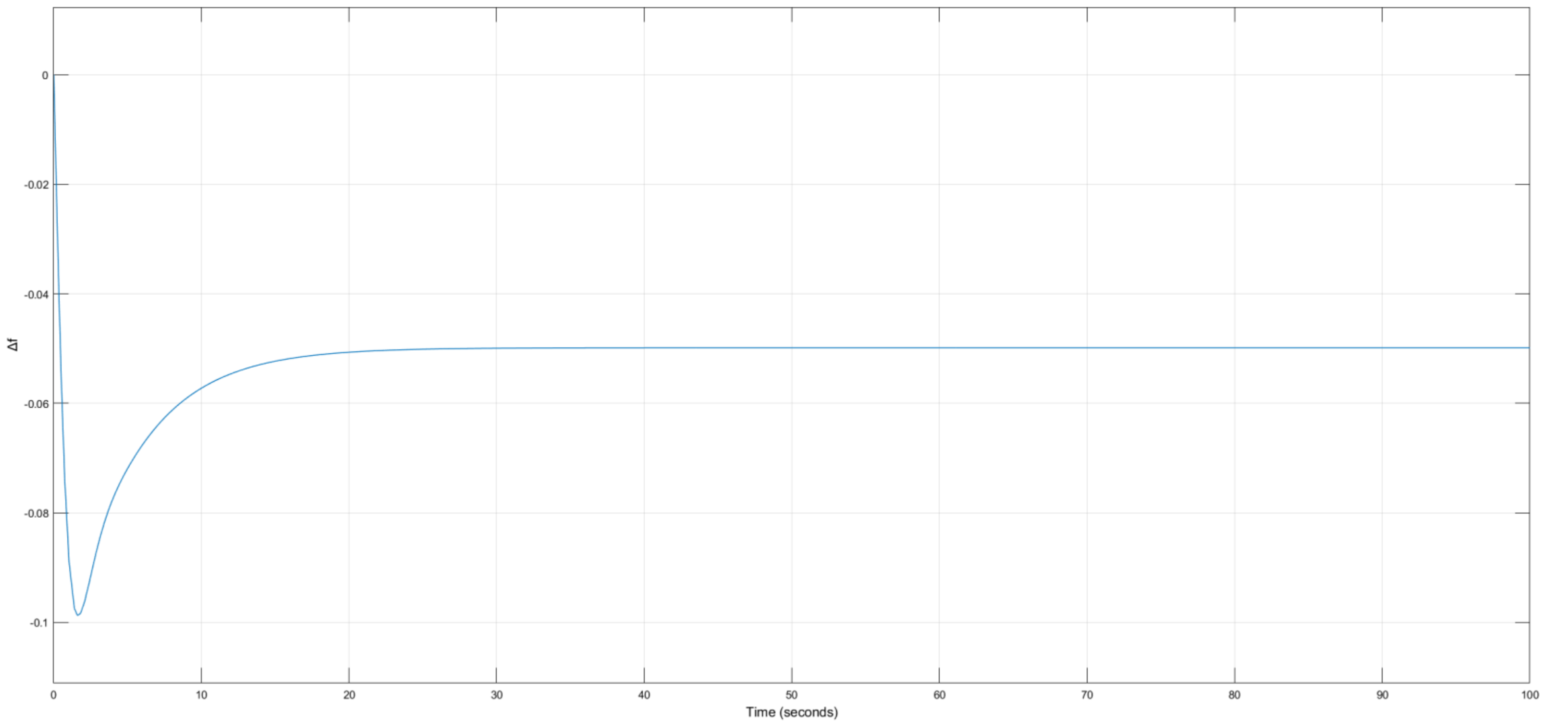



*Figure 6 Effect of introducing Droop Characteristics*

The droop characteristics somehow helps in compensating the load, but still there is some steady state error, as steady value is zero in our case, because we don't want any frequency deviation.

By trial and error, we found the optimal value of 1/R to be 2, and we have used that value throughout out our whole project design.

As we can see above there is still some frequency deviation in our system, so now to compensate that deviation we will be designing continuous PID controller first and then discretise it to design digital/discrete PID controller.

# 4.Design of Continuous PID controller

A continuous time PID controller is approximated with the provided gains $K_p$, $K_i$ and $K_d$.

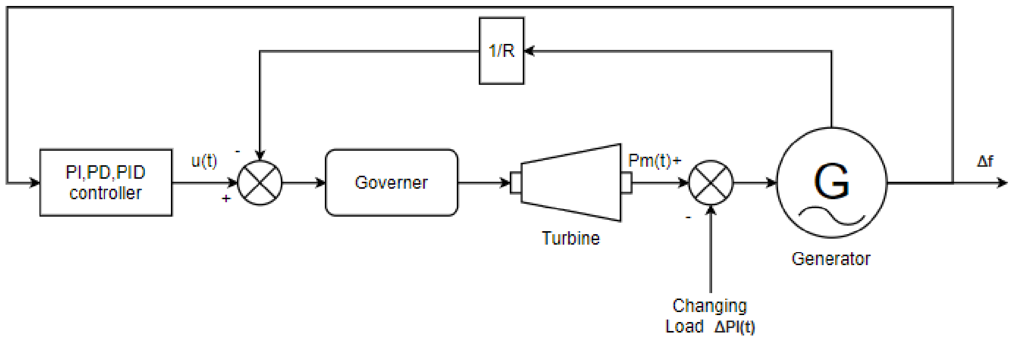



*Figure 7 Proposed LFC design Model with PID controller*

The transfer function of continuous-time PID controller is:

$$C(s) = K_p + \frac{K_i}{s} + K_d s$$

where;

$K_p$ = Proportional gain

$K_i$ = Integral gain

$K_d$ = Derivative gain

Each parameter has their own significance in the control design:

- Proportional gain helps in improved steady state offset and increased oscillations.
- Integral gain helps in achieving zero steady state offset and increased oscillations.
- Derivative gain helps in decreasing oscillations and improved stability.

In our project, with the help PID controller we provide control signal u(t) as input, and through that control signal we will be controlling our entire system and achieve zero frequency deviation as output. And, we will see how each controller has effects on system response.

## 4.1 Continuous PID Control Tuning

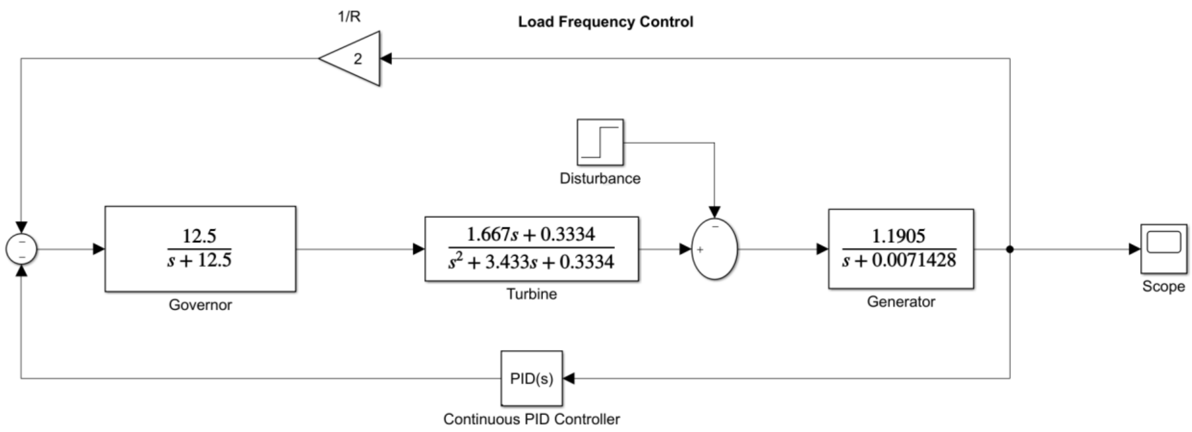
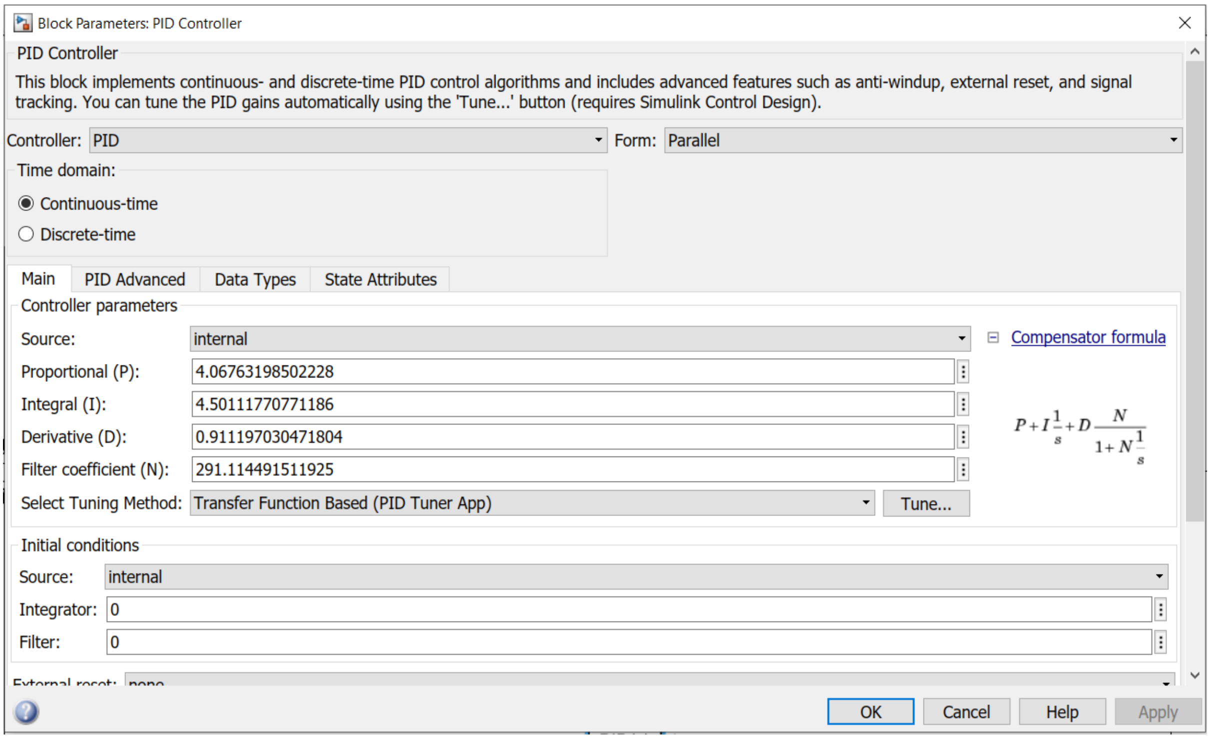



*Figure 8 Schematic of whole system with continuous PID*

As we can see above, we have introduced a continuous PID controller block in our schematic. And within that block we can choose different values for proportional, integral and derivative gain parameters.

With the help of Simulink Control Design Toolbox available in MATLAB we can automatically tune or PID controller for desired response time and transient behaviour of our whole control plant. As we can see below the Tune button inside PID parameter.



Unlike our theoretical continuous PID controller transfer function, MATLAB uses the following transfer function for continuous PID controller.

C(s) = $Kp + Ki\frac{1}{s} + Kd\ \frac{N}{1+N\frac{1}{s}}$

Where; Kp, Ki, Kd are the proportional, integral and derivative gain respectively.

And, N is the bandwidth of low pass filer coefficient. A pure derivative gain i.e. Kd*s amplifies the noise, so MATLAB uses the practical implementation with includes the filer coefficient.

The continuous PID block internally works like this with the filter coefficient and individual control blocks:

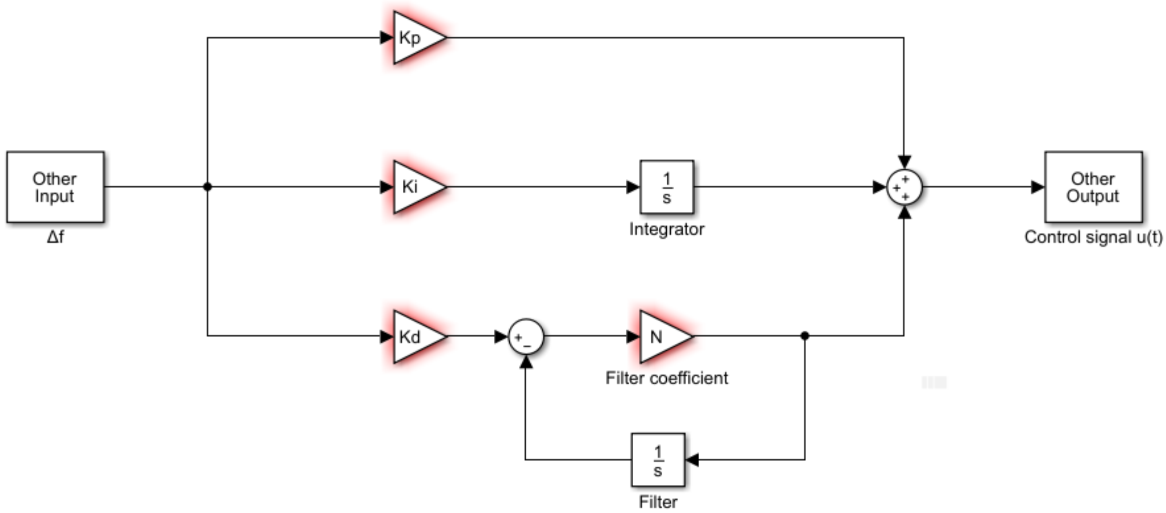



*Figure 9 Continuous PID internals*

The method we opted for tuning method is Transfer Function Base (PID Tuner App). This process linearizes our plant and helps in finding the optimal tuned parameters.

And, as we play through the desired Response Time and Transient Behaviour, we can see the plant response with the tuned parameters of Kp, Ki, Kd and N coefficient.

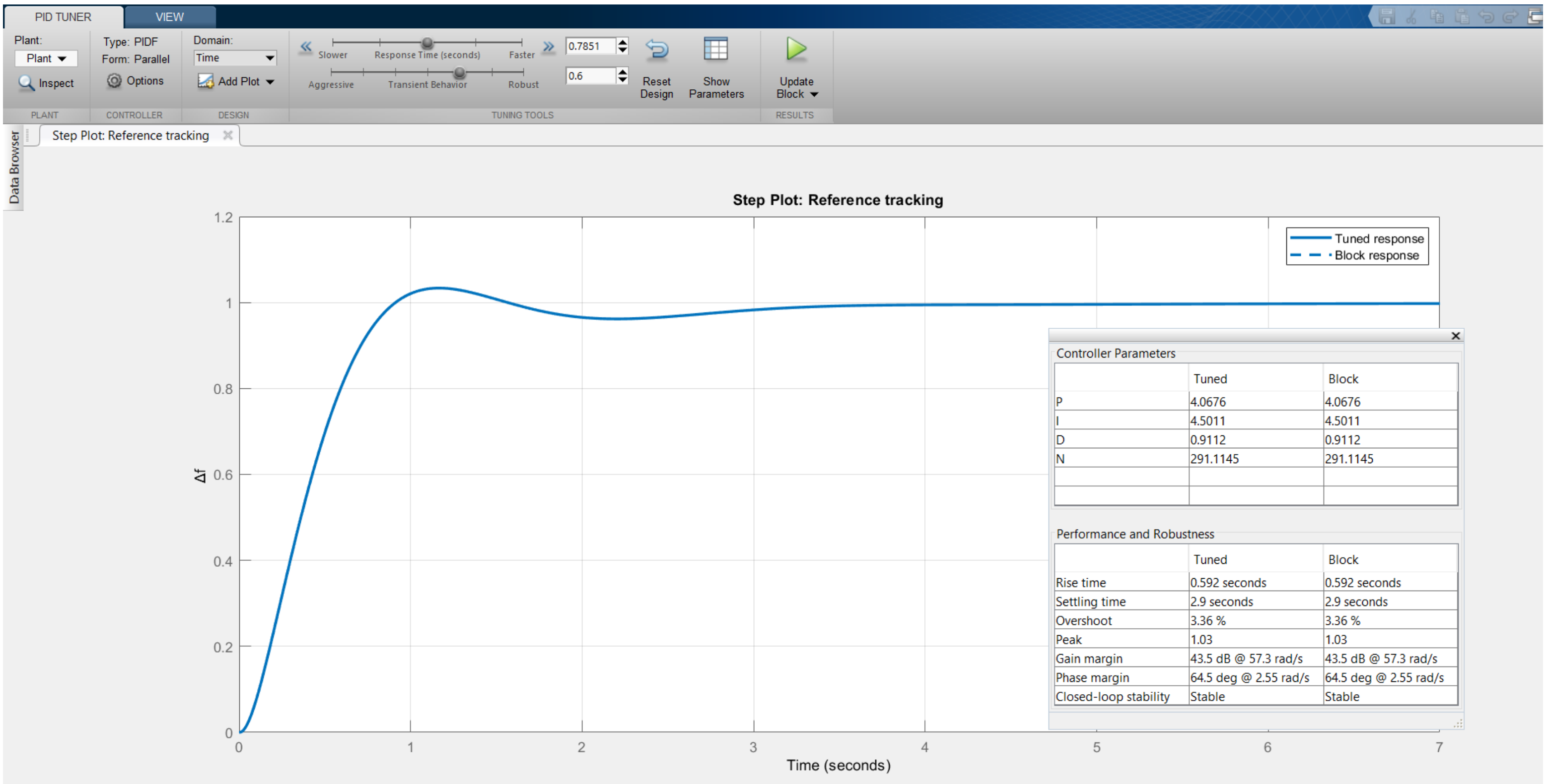



*Figure 10 Continuous PID tuning Process*

So, if we provide any parameters of Kp, Ki, Kd and N the tuning process will approximate the best tuned values for each controller gains and present them as Tuned gains, as we can see in the above figure, within the Controller Parameters. And, we can see the performance and robustness in the table too.

For Example, in our case we started with the following default Block parameters for each proportional, integral, derivative gain and filer coefficient, N, and we obtained the following tuned values for our desired response time and transient behaviour.

Controller Parameters

| | Tuned | Block |
|---|---|---|
| P | 4.0676 | 1.6514 |
| I | 4.5011 | 1.8505 |
| D | 0.9112 | 0.12704 |
| N | 291.1145 | 13.1574 |
| | | |
| | | |

*Figure 11 Default Block and Tuned parameters*

The performance and robustness of the plant:

Performance and Robustness

| | Tuned | Block |
|---|---|---|
| Rise time | 0.592 seconds | 1.21 seconds |
| Settling time | 2.9 seconds | 5.01 seconds |
| Overshoot | 3.36 % | 0 % |
| Peak | 1.03 | 0.997 |
| Gain margin | 43.5 dB @ 57.3 rad/s | 26.3 dB @ 8.55 rad/s |
| Phase margin | 64.5 deg @ 2.55 rad/s | 75.5 deg @ 1.26 rad/s |
| Closed-loop stability | Stable | Stable |

*Figure 12 System Performance*

The above performance and robustness is without the disturbance, because the step disturbance does not come under the linearization path in the PID tuner app.

The step load disturbance in our project design has a final value of 0.1 i.e. 10% of the unit step response.

The plant response with the continuous PID controller and with the step load disturbance of final value 0.1 is given below:

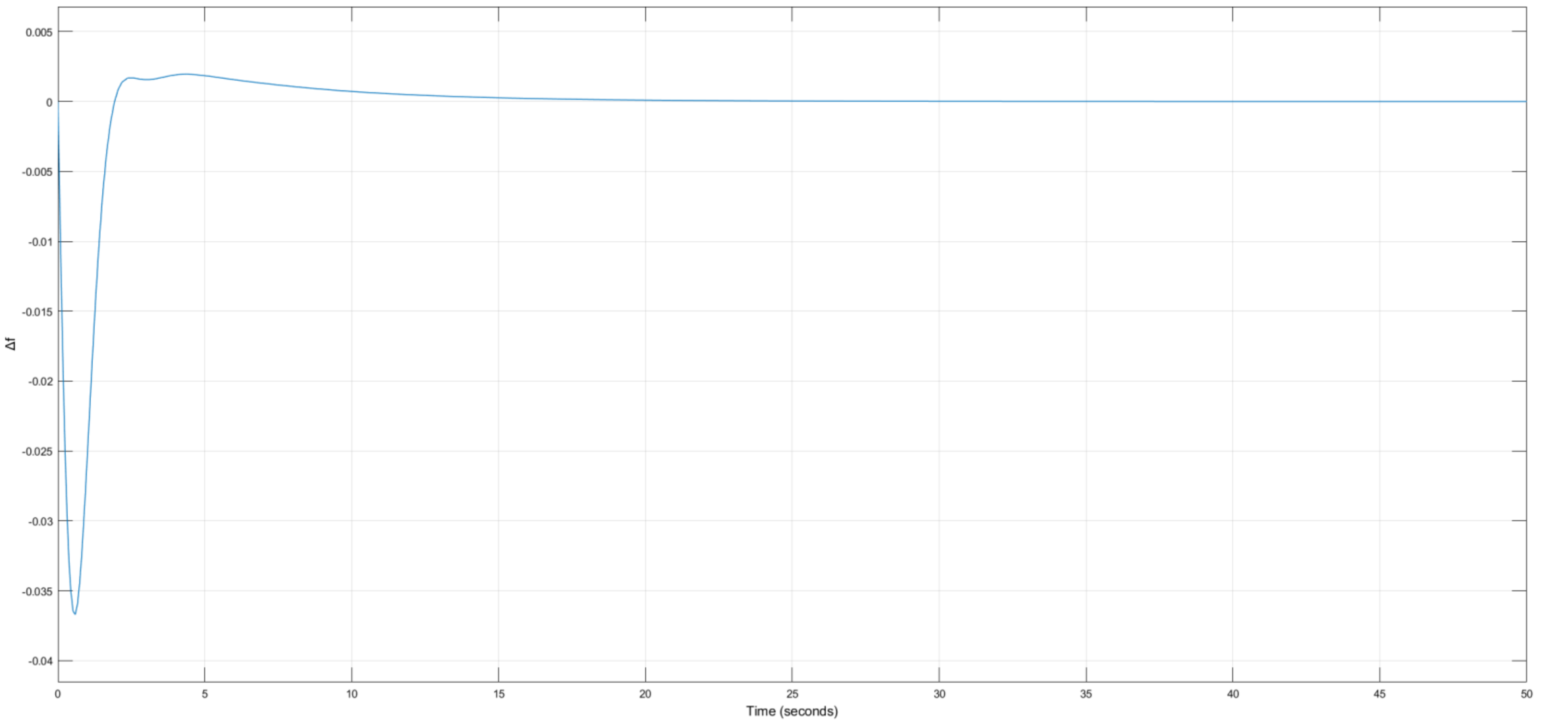



*Figure 13 Plant response with continuous PID control with step load disturbance of final value 0.1*

As we can see above the frequency dips at time t = 0 because we applied our step disturbance at time t = 0, and gradually the PID controller compensates the load disturbance. At final the frequency deviation is zero.

# 5.Design of Discrete PID Controller

A widely used method to discretise the controller is to convert the integral and derivative terms from the continuous PID controller transfer function to discrete-time counterpart. There are several methods like Bilinear transformation, Forward Euler, Backward Euler, and Trapezoidal methods.

The proportional gain Kp remains same as it is a constant.

For the integral controller term $\frac{Ki}{s}$, the discrete-time equivalent transfer functions according to different methods are:

Forward Euler (Difference):$\frac{Ki\,Ts}{z-1}$

Backward Euler (Difference): $\frac{Ki\,Ts\,z}{z-1}$

Trapezoidal Method: $\frac{Ki\,Ts\,(z+1)}{2\,(z-1)}$

For the derivative term $Kd\ *\frac{N}{1+N\frac{1}{s}}$, the discrete-time equivalent transfer functions according to different methods are:

Forward Euler (Difference): $\frac{N}{1+N\,Ts\,/\,(z-1)}$

Backward Euler (Difference): $\frac{N}{1+N\,Ts\,z/(z-1)}$

Trapezoidal Method: $\frac{N}{1\,+\,N\,Ts\,(z+1)/2(z-1)}$

Where, Ts is the sampling time.

## 5.1 Internal Schematic of the Discrete PID Controller in MATLAB

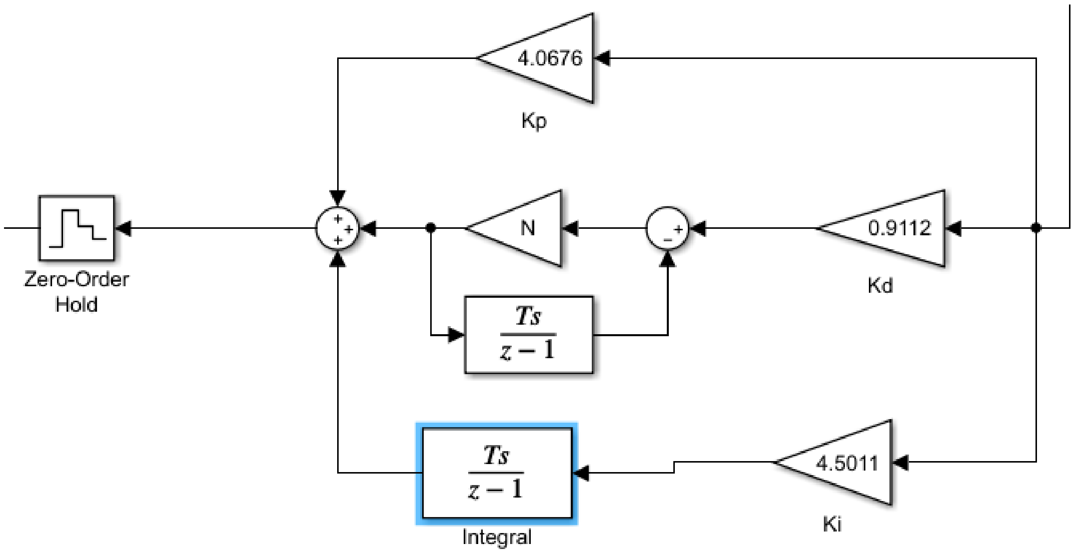



*Figure 14 Theoretical Internals of digital PID using Forward Euler (Difference)*

# 6.Selection of Sampling Time through ad-hoc Rule

One of the ad-hoc rules in selecting the sampling time is:

- Choose $\omega c\ Ts$ to be 0.15 to 0.5, where $\omega c$ is the crossover frequency and $Ts$ is the sampling time.

From the bode plot of the plant open loop the cross over frequency can be found to be 6.34 rad/s.

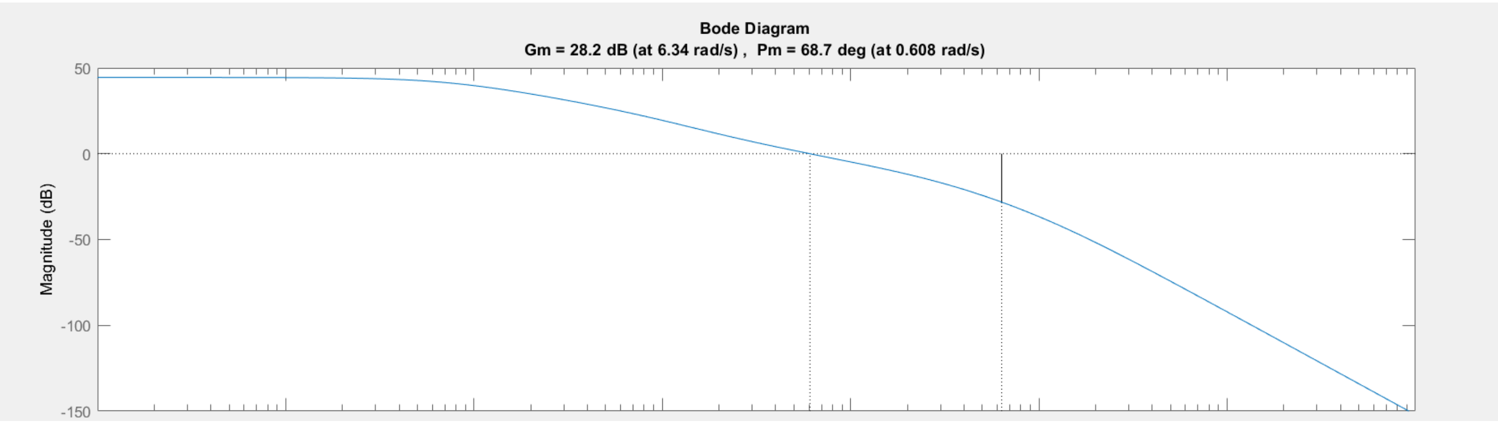



*Figure 15 Cross over frequency*

Now, suppose:

$$\omega c * Ts = 0.15$$

⇨ Ts = 0.15/6.34

⇨ Ts = 0.023 s

So, the optimal sampling time Ts can be 0.023 seconds for our digital controller.

The system response with the discrete PID controller implemented with forward difference (Euler) is:

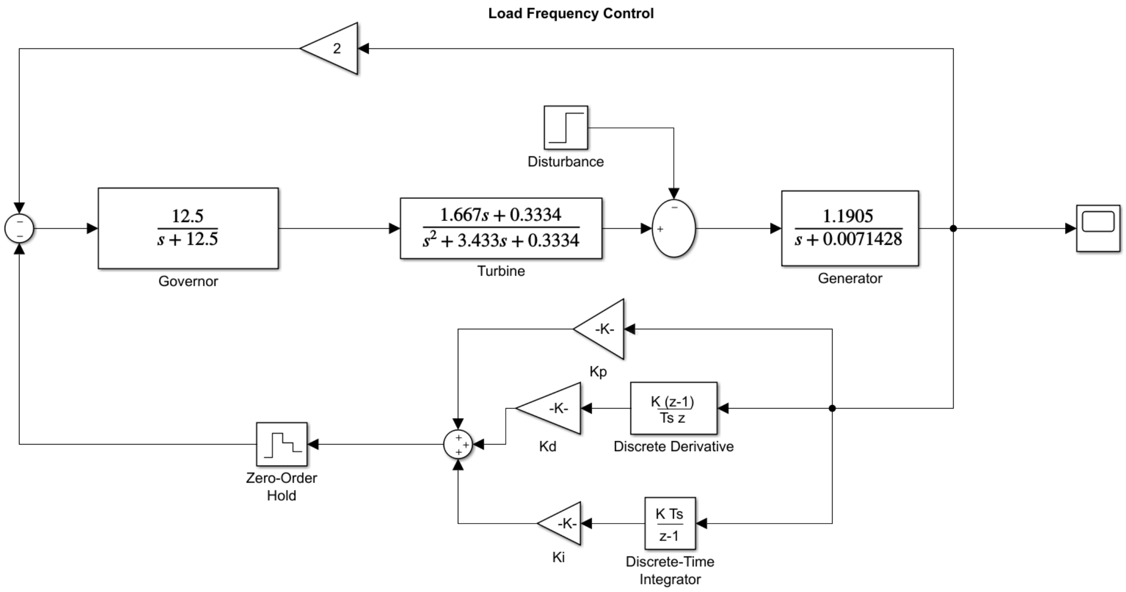



*Figure 16 Discrete PID controller Implemented forward difference (Euler)*

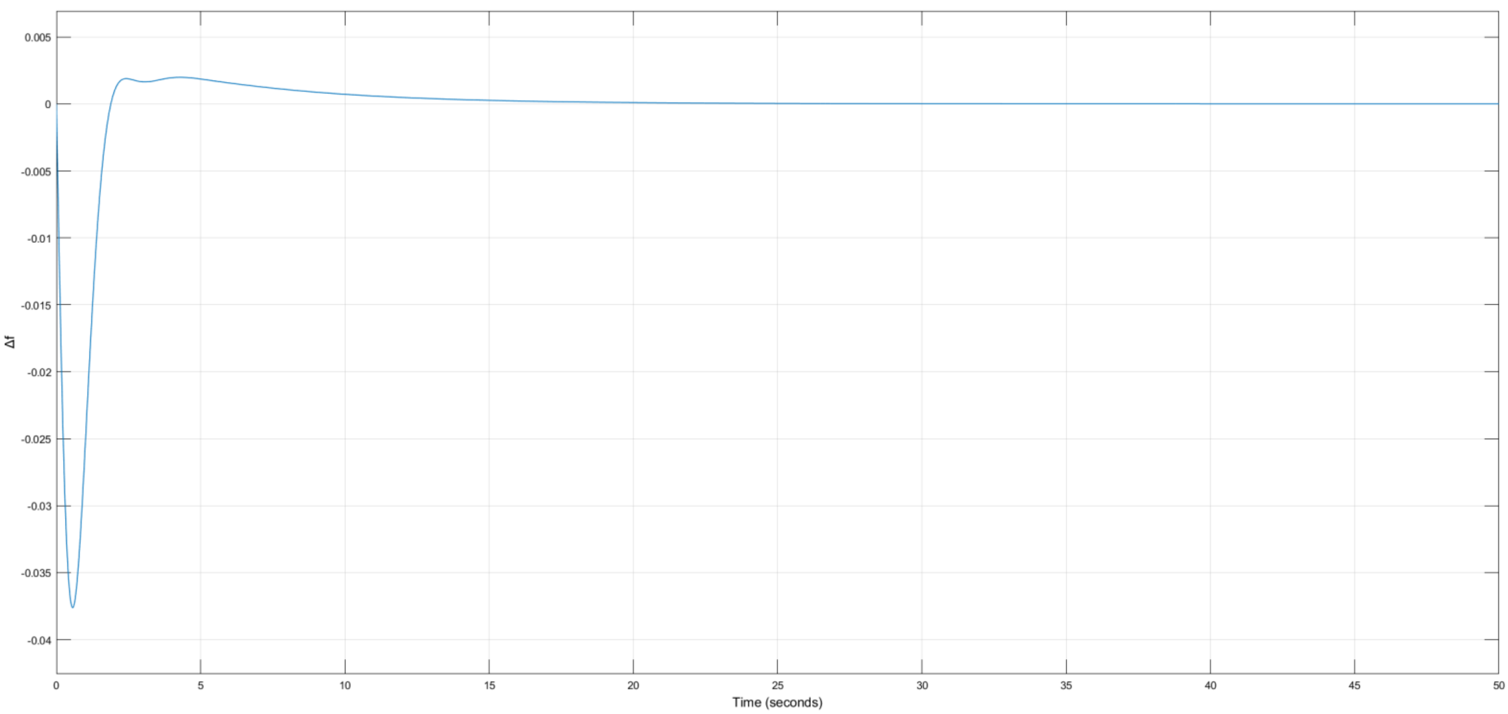



*Figure 17 System Response with discrete PID implemented*

## 6.1 System Response with only Proportional Control

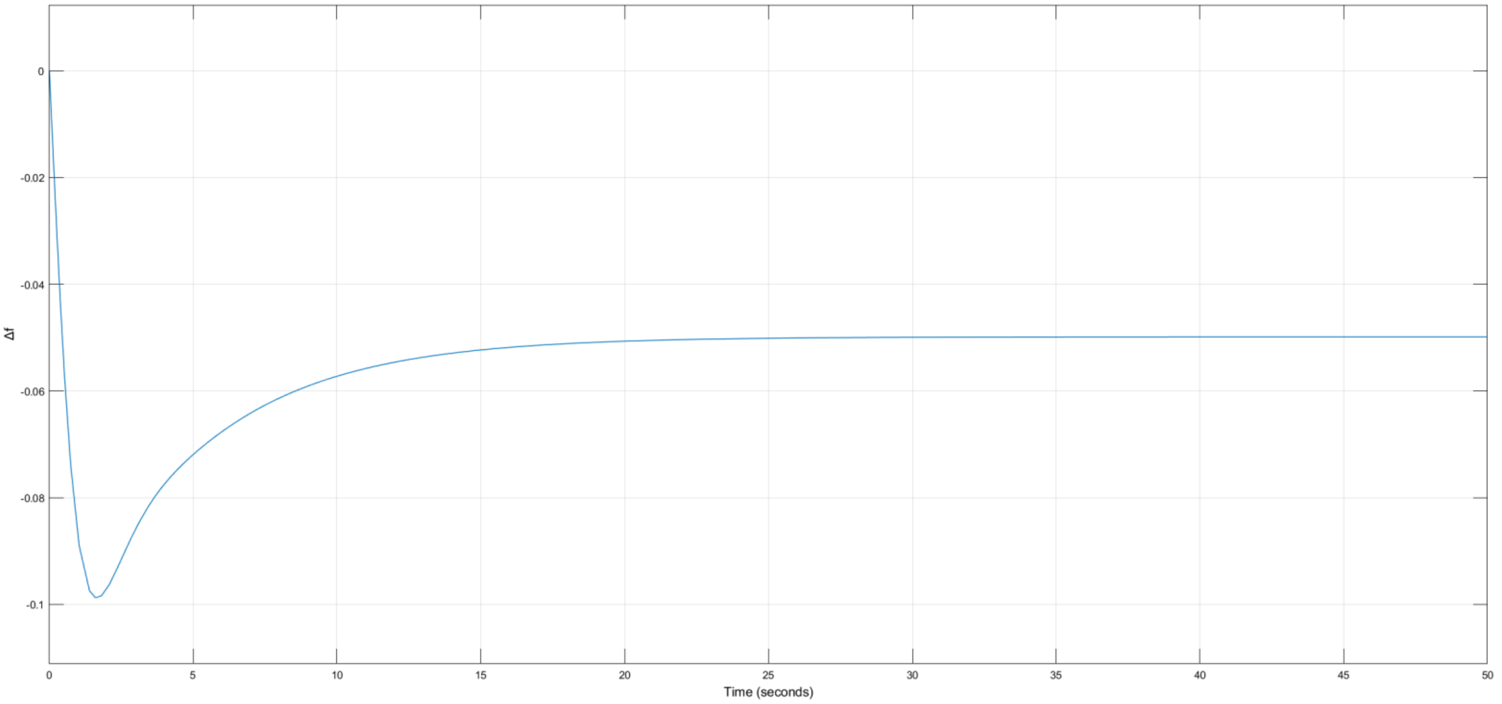



*Figure 18 System Response without any Controller*

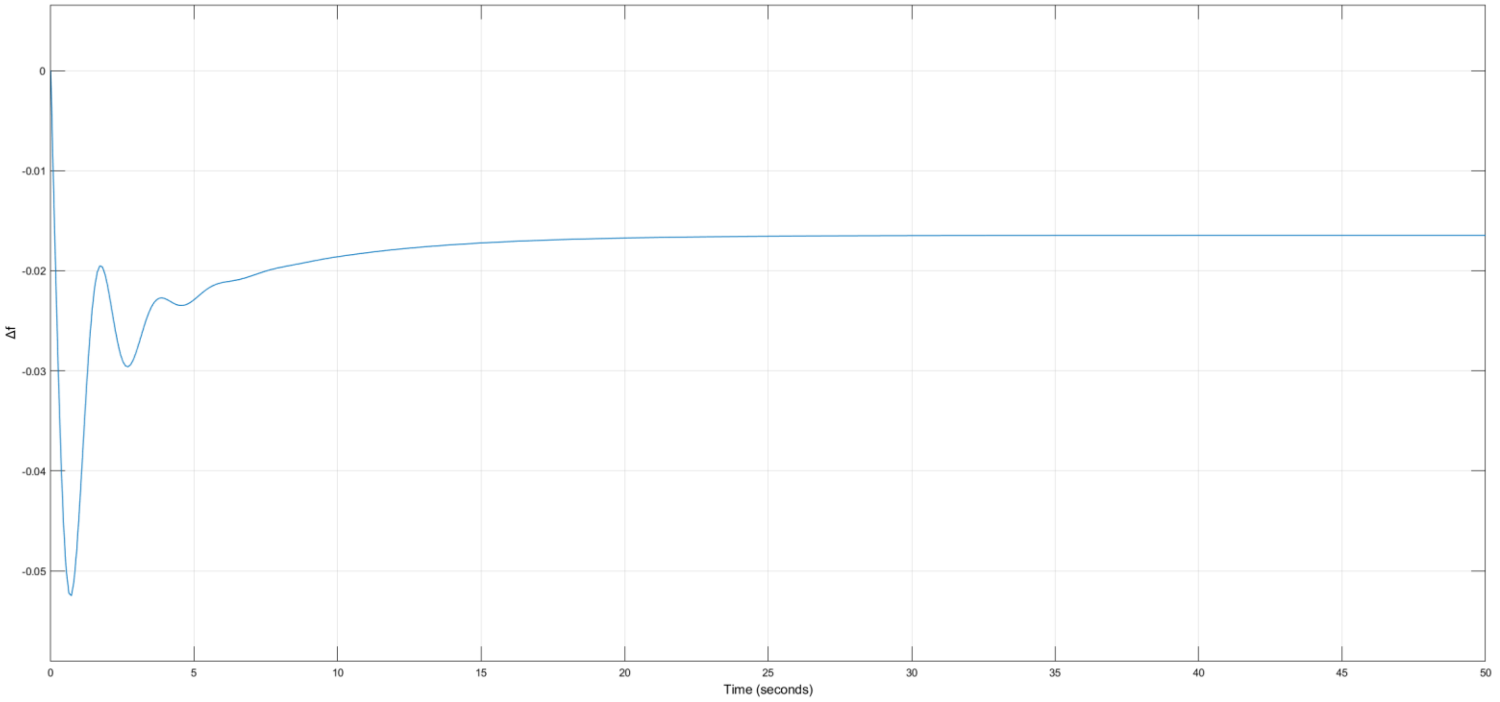



*Figure 19 System Response with only Proportional control*

The final value is 0 for our system. As we can see with proportional control steady error offset is improved.

## 6.2 System Response with only Proportional and Integral Control

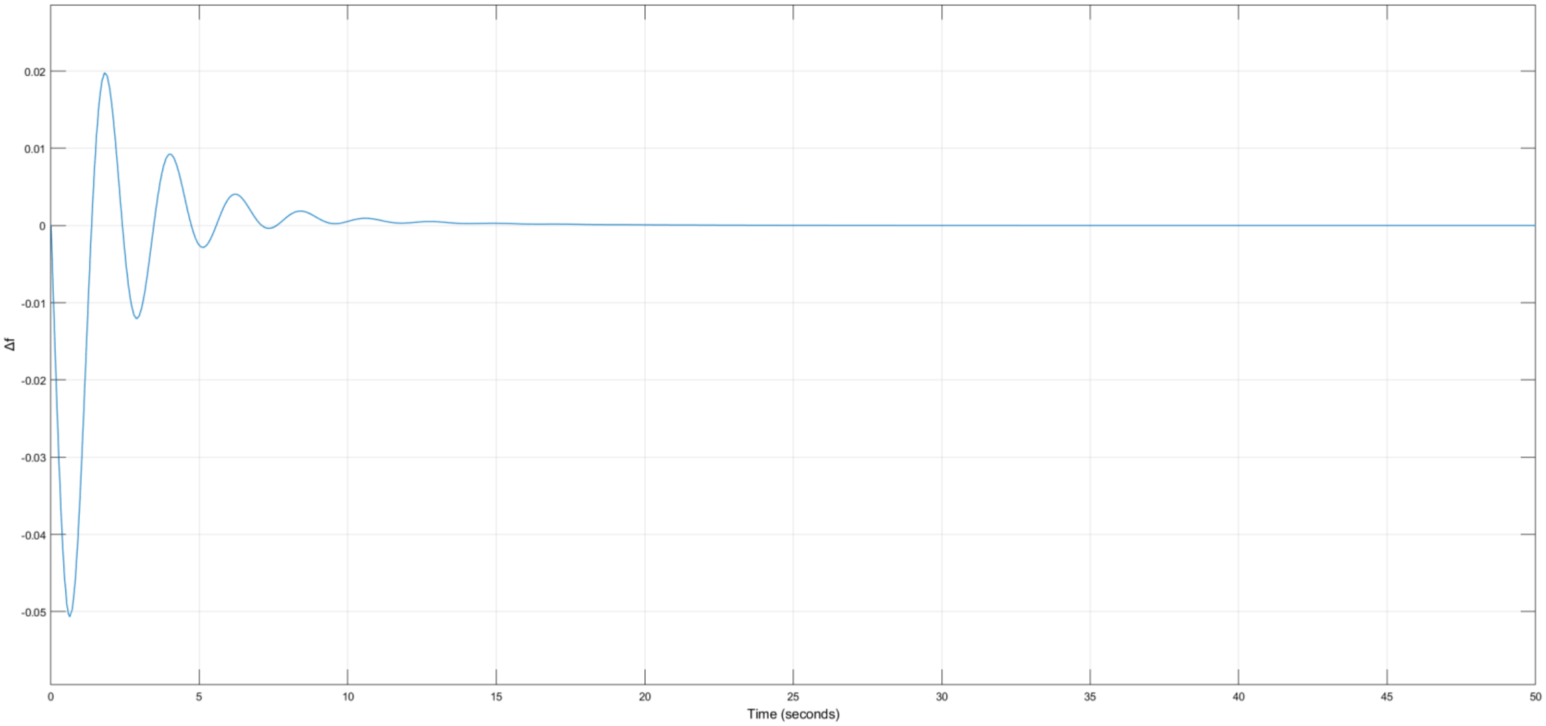



*Figure 18 System Response with PI controller*

As we can see above with introduction of Integral control, the oscillations have increased, but at the same time steady state error of zero is achieved.

## 6.3 System Response with PID controller

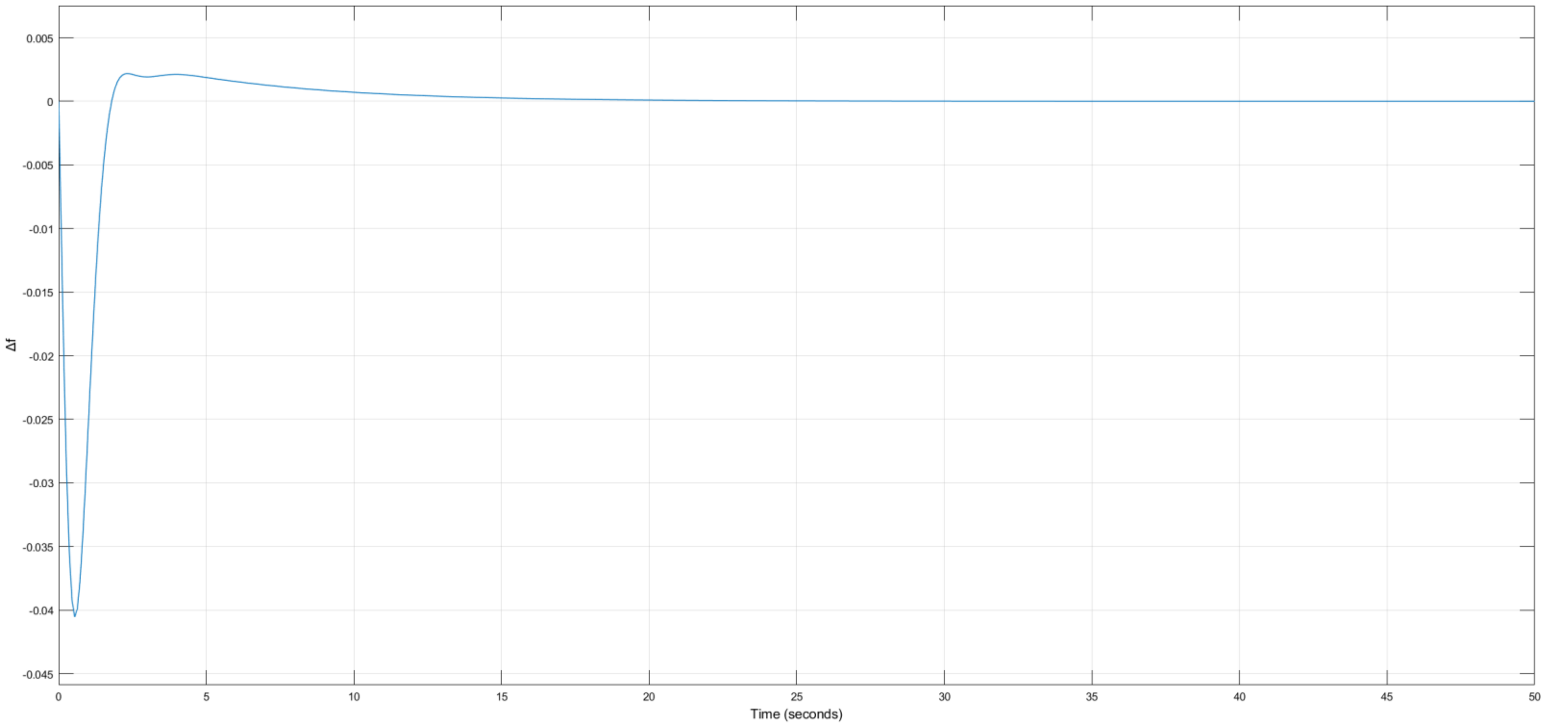



*Figure 19 System Response with PID controller*

With introduction of Derivative control in the PI controller, the oscillations have reduced as we had proposed earlier in the theoretical aspects of PID controller.

# 7.Conclusion

Hence, throughout the project we were introduced to MATLAB and it's tools like Simulink, PID tuner App, System identification Tool. This project helped us in learning the process of designing and tuning a continuous PID and develop a discrete PID controller accordingly with different transformation methods with the optimal sampling time.

# References

Chakrabortty, A., Nagle, H.T. and Phillips, C.L., 2015. *Digital Control System Analysis & Design: Global Edition.*

Ctms.engin.umich.edu. 2020. *Control Tutorials For MATLAB And Simulink - Introduction: Simulink Modeling*. [online] Available at: <http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=SimulinkModeling> [Accessed 29 May 2020].

Moudgalya, K.M., 2007. *Digital control*. New York: Wiley.

Scilab.org. 2020. *Discrete-Time PID Controller Implementation | www.Scilab.Org*. [online] Available at: <https://www.scilab.org/discrete-time-pid-controller-implementation> [Accessed 29 May 2020].