MECH 6316 – DIGITAL CONTROL OF AUTOMOTIVE POWERTRAIN SYSTEMS
BY DR. STEPHEN YURKOVICH

Idle Speed Control Problem in an Automotive Powertrain

Abishek Chandrasekhar

4/25/2017

In this project, different control methods are developed to maintain the idle speed of an Internal Combustion engine for which the simulator was given. Initially, the open loop dynamics of the simulator were studied. The nonlinear equations of the simulator were used to develop a discrete model of the engine. In the model developed, a SISO controller was implemented wherein the throttle angle was the input. The controller is further improved by adding another input, i.e., the spark advance angle. The different control techniques that were implemented were PID control, Root Locus technique and State Feedback design. A summary comparing the different control designs utilized is given.

1. Introduction:

The powertrain in an automobile includes the engine, the transmission, the drive shaft and the differentials. Automotive companies face various challenges such as maintaining performance and fuel efficiency. The objective of this project is to find a solution for the Idle Speed Control (ISC) problem. A simulator for a 2.4 liter 4-cylinder internal combustion engine is utilized as the model upon for which various control techniques will be implemented.

1.1 The Idle Speed Control Problem:

The Idle Speed Control problem is used to maintain the engine speed at a set level. This is important to keep the engine from stalling, minimize fuel consumption and reduce the pollutant emissions. The engine should be modeled such that it can maintain a steady speed when subjected to loads (such as when the air conditioning is turned on, servo motors for power steering or in fact any other devices that are powered by the engine are turned on. The control should be developed such that it can handle any unpredictability in the load change. The major consideration is to maintain the engine speed above 500 rpm, at which the engine stalls.

The parameters that have considerable effects on the idle speed control are the air flow and the ignition timing, which can be controlled. The other major parameter that can be controlled is the throttle angle.

1.2 Control Techniques:

The simplest solution to the ISC problem is implementing a form of feed-forward control with closed loop gain. The feedforward control utilizes look-up tables which anticipates the loads due to different components in a vehicle (e.g., when A/C is turned on).

Classical and modern control techniques can be utilized to maintain the idle speed of the engine.

- Proportional Integral Derivative (PID) control
  PID control can be used when information about the model is very limited. This control technique only depends on the output of the plant.
- Root Locus Technique
  The Root Locus is used to represent graphically the open loop and closed loop response of the system. The unstable poles are stabilized in the closed loop.

The various ways of implementing the control using modern control techniques are:

- SISO (Single Input Single Output) control where the controllable parameter is the throttle angle. The control is designed such that the throttle angle input (α) is within the allowable limit, in this case 6 – 20 deg.
- MIMO (Multiple Input Multiple Output) control where the controllable parameters are the throttle angle and the spark advance angle. The throttle angle input is maintained within the limits mentioned above. The spark advance angle () is maintained within the limit, in this case -30 – 0 deg.

The feedback gains for the SISO and MIMO control can be calculated using state feedback algorithms and placing the required poles or using a Linear Quadratic Regulator.

2. Structure of Engine model:

The Internal Combustion engine is a dynamic model, and their components are considered as dynamic systems. The physical variables of the engine become state, input and output signals. In this model of the engine, they are defined as:

Inputs – Throttle Position, Spark Timing and Load Disturbance.

Outputs – Intake Manifold Pressure (MAP), Indicated Torque and Engine Speed.

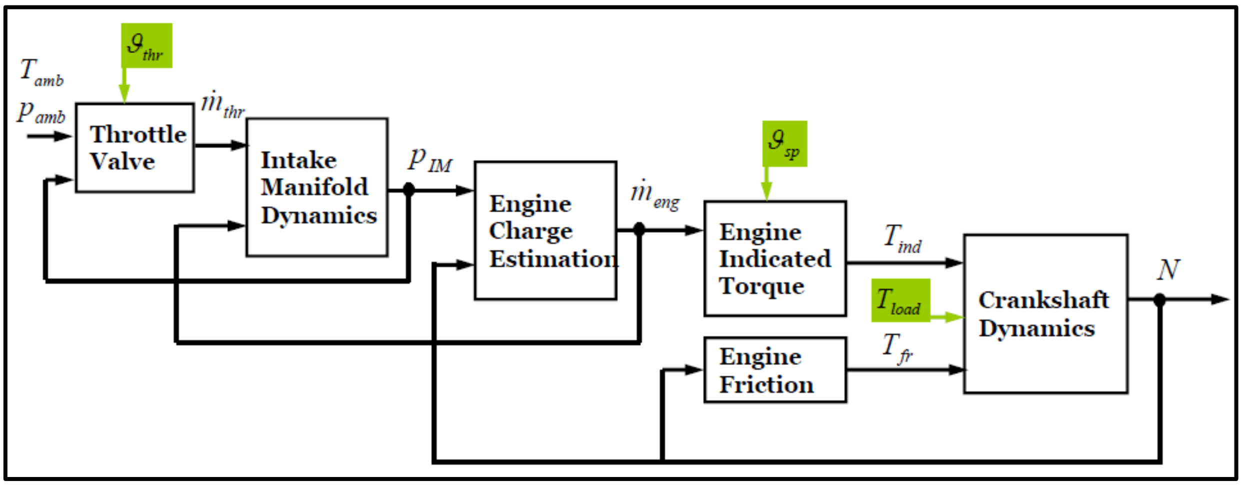

Figure 1 - Dynamic Structure of Engine model

2.1 Air Dynamics:

The air is drawn from the ambient through the throttle valve. The intake manifold distributes this air to the cylinders.

In the model, the flow of air into the intake manifold is determined by the angle of the throttle valve, i.e., the throttle angle is an input for the air dynamics. When loads are induced in the engine, the throttle valve is opened, more airflow is allowed into the cylinders of the engine. This is done in order to maintain the engine speed at the set point of 800rpm.

2.2 Torque Dynamics:

The engine speed is given by modeling the crankshaft dynamics. The spark advance angle is used as the input for the torque dynamics. The indicated torques and the friction torques are estimated with quasi-static models. The torque at the engine shaft is given by:

$$T_{brake} = T_{ind} - T_{fr}$$

The model structure is defined and the parameters are calibrated. The set of differential equations are used to create a Simulink model. In the model, the open loop system dynamics are studied by varying the throttle angle, the spark advance angle and different load torques. The manifold air pressure, the indicated torque and the engine speed are measured.

The dynamics of the engine are given by a set of equations:

$$\dot{m}_{thr} = \left[\frac{P_{amb}}{\sqrt{RT_{amb}}} \sqrt{\gamma} \sqrt{\left(\frac{2}{\gamma+1} \right)^{\frac{\gamma+1}{\gamma-1}}} \right] [a_2 \alpha_{thr}^2 + a_1 \alpha_{thr} + a_0]$$

$$\frac{dP_{IM}}{dt} = \frac{RT_{amb}}{V_{IM}} (\dot{m}_{thr} - \dot{m}_{eng})$$

$$\dot{m}_{eng} = \frac{(s_i P_{im} - y_i) V_d N}{120RT}$$

$$T_{ind} = \frac{V_d}{4\pi 10^{-5}} \{b_1 P_{IM}(t - t_d) - b_0\} \{c_2 \vartheta_{spark}^2 + c_1 \vartheta_{spark} + c_0\}$$

$$t_d = \frac{120}{ZN}$$

$$\frac{dN}{dt} = \frac{30}{\pi J} [T_{ind} - T_{load} - T_{fr}]$$

$$T_{fr} = d_2 N^2 + d_1 N + d_0$$

Based on these seven equations, the Simulink model of the engine is built as shown below:

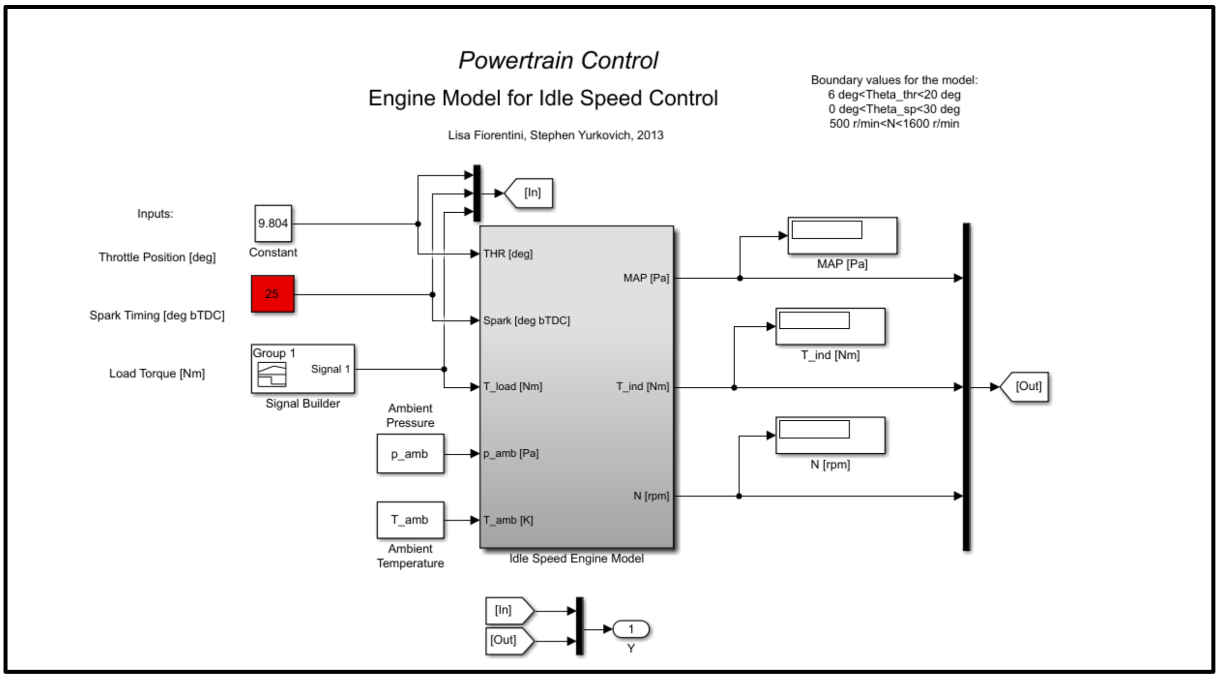

Figure 2 - Simulink model of the engine

3. Open Loop System Dynamics:

In the model created using the set of differential equations, the input parameters are varied in order to understand the dynamics. The limits of the parameters are as follows:

Throttle angle	- 6 deg<Theta_thr<20 deg
Spark advance angle	- 0 deg<Theta_sp<30 deg
Engine speed	- 500 r/min<N<1600 r/min

3.1 Variation of External torque:

The throttle angle and spark advance angle are set at constant values of 10 deg and 25 deg bTDC, respectively. The external torque is varied as 0, 5, 10, 15 and 20 Nm.

Constants:

Throttle Angle [deg]	: 10
Spark Advance Angle [deg bTDC]	: 20

Table 1 - Variation of External Torque

External Torque [N-m]	MAP[Pa]	Indicated Torque [N-m]	Engine Speed [RPM]
0	2.223e+04	11.62	1070
5	2.432e+04	16.48	934.7
10	2.643e+04	21.4	828.9
15	2.855e+04	26.34	744.2
20	3.068e+04	31.3	674.9

Inference:

- The **speed** of the engine (rpm) **decreases.** It can be inferred that the engine speed is inversely proportional to the external load. But it should also be noted that **as the load is increasing**, the **rate of speed decrease** also gradually **decreases**. Taking the case where the load is increased from 0 to 5Nm, speed decreases by roughly 136 rpm. But when the load is increased from 15 to 20Nm, speed decreases by 70 rpm.
- The intake **manifold pressure** (kPA) and **the indicated load torque** (Nm) **increases.** These parameters are directly proportional to the external load. The rate of increase of these parameters is also consistent for every 5Nm increase of external load.
- The engine speed is **always above 500 rpm** for all values of external torque. The **engine doesn't stall** for any value of Load.

3.2 Variation of throttle angle:

The external torque and spark advance angle are set at constant values of 10 Nm and 25 deg bTDC, respectively. The throttle angle is varied to take on values of 7, 8, 10, 12, 14 and 16 deg.

Constants:

External Torque [N-m]	: 10
Spark Advance Angle [deg bTDC]	: 25

Table 2 - Variation of Throttle Angle

Throttle Angle [deg]	MAP [Pa]	Indicated Torque [N-m]	Engine Speed [RPM]
7	2.64e+04	21.36	489
8	2.637e+04	21.26	576.6
10	2.643e+04	21.4	828.9
12	2.657e+04	21.73	1164
14	2.684e+04	22.35	1573
16	2.728e+04	23.37	2040

Inference:

- The increase in throttle angle **increases the speed** of the engine (rpm). As the throttle angle increases in steps of 2 degrees, the rpm of the engine increases. It is also noted that the **increase** in rpm **for bigger values** as the throttle angle increases from 14 to 16 is **much higher** than the increase in rpm for a throttle angle increase from 7 to 9 degrees.
- The intake manifold **pressure** also **increases** but has smaller increments. The rate of increase is consistent as the throttle angle increases.
- For the **throttle angle of 7 deg**, the engine speed drops to 489 rpm, which implies the **engine is stalling**.

3.3 Variation in spark advance angle

The throttle angle and external torque set at constant values of 10 deg and 10 Nm, respectively. The spark advance angle is set at different values of 30, 25, 20, 15, 10 and 5 deg bTDC.

Constants:

Throttle angle [deg]	: 10
External Torque [N-m]	: 10

Table 3 - Variation in Spark Advance Angle

Spark Advance Angle [deg bTDC]	MAP [Pa]	Indicated Torque [N-m]	Engine Speed [RPM]
30	3.094e+04	21.3	667.5
25	2.643e+04	21.4	828.9
20	2.464e+04	21.47	917.1
10	2.348e+04	21.53	984.7
5	2.353+04	21.53	981.8

Inference:

- As the spark advance angle is decreased from 30 to 5 degrees in decrements of 5, the engine speed increases. The engine speed (rpm) is inversely proportional to the spark advance angle.
- The intake manifold **pressure decreases** as the spark advance angle decreases.
- Whereas the **indicated torque increases** by a very small amount for **initial decrements** of the spark advance angle. But as the angle reaches 10 degrees and further reduces to 5 deg, the indicated torque **does not show any increment**.
- The engine speed is **above 500 rpm** for all values of spark advance angle. The **engine does not stall** at any point.

3.4 Ideal Throttle angle positions:

The throttle angle position which would result in 800 rpm for different values of external torques represented by values 0, 5, 10, 15 and 20 N-m are found out using the tables and the simulator. The spark advance angle is maintained at 25 deg bTDC throughout this process.

Constants:

Spark Advance Angle [deg bTDC] : 25

Table 4 - Throttle positions for maintaining engine speed

External Torque [N-m]	Engine Speed [RPM]	Throttle Angle [deg]
0	800	8.374
5	800	9.132
10	800	9.8015
15	800	10.4075
20	800	10.965

Implementation:

- In order to achieve an engine speed of 800 rpm for the increasing load torques, a constantly increasing step input was created using signal generator.
- The limits of the signal were between 6 and 20 deg (throttle angle bound). The increments in step size were given in the order of 0.1 (ex. 6, 6.1).
- An 'if else' condition block was created and connected to the output terminal that gave the engine speed (rpm). The condition was to check the speed for 800 rpm after a particular period of time (time condition was different for each load torque). If the engine reaches 800 rpm after the particular instant of time, the simulation is stopped.
- The throttle angle is measured once the simulation is stopped once the condition is satisfied.

Inference:

- The throttle angle is increased in steps of approximately 0.7 – 1.0 degrees in order to maintain the engine speed at 800 rpm for every 5 N-m increase in external torque.
- In this case, the spark advance angle is kept constant, so the throttle position is the only input to the engine model.
- This data will be useful for anticipatory control techniques.

3.5 External load acting on the engine:

A step function is invoked at 5 seconds for the external torque increasing from 10 Nm to 16 Nm. This can be used to represent the A/C turning on in the vehicle. Another step input decreasing from 16 Nm to 10 Nm at 15 sec (when the A/C cycles off), while maintaining the throttle opening angle at the value that was found previously in Table 4, for the value corresponding to 10 N-m. The spark advance is also constant at 25 deg bTDC. The response of the engine is as follows:

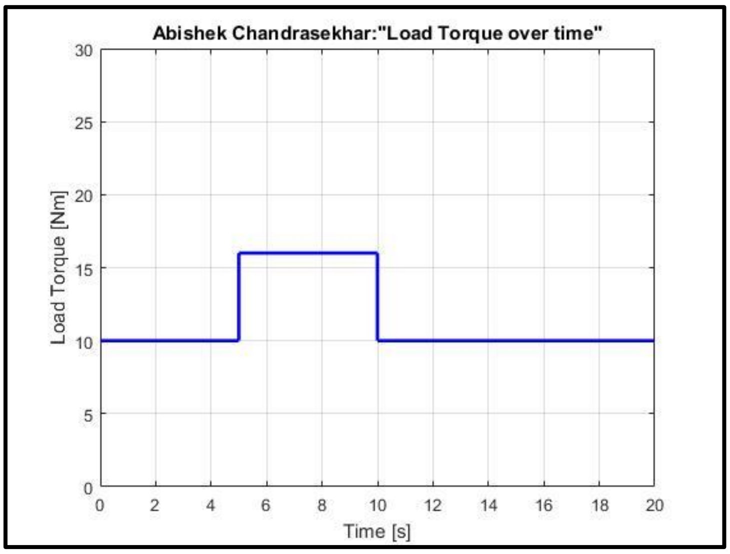

Figure 4 - Load torque that is invoked

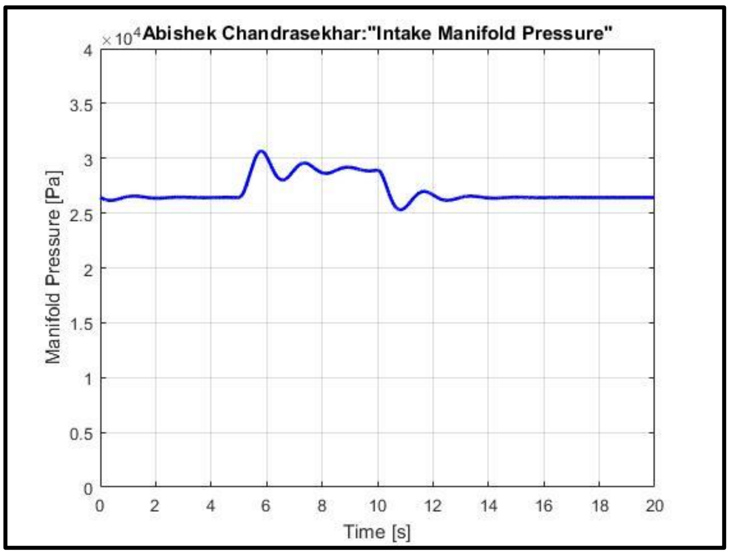

Figure 3 - Intake manifold pressure response to load torque

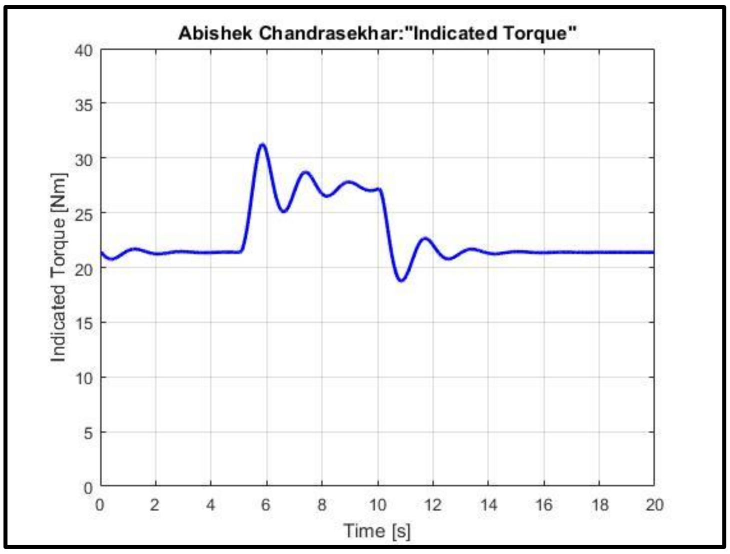

Figure 6 - Indicated torque

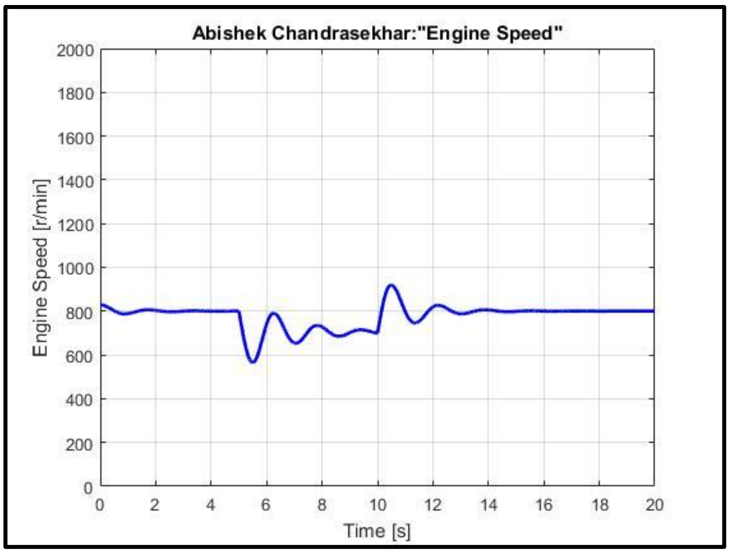

Figure 5 - Engine speed response to load torque

Inference:

- It can be seen from the above plots that the invoking of a load torque causes variations in the outputs. There are fluctuations in the engine speed. It varies between just under 600 and 950 rpm, but the engine does not stall in this case. (Does not drop below 500).

3.6 Feedforward control using Throttle angle:

A simple open loop controller is created to attempt to correct for the external torque disturbances. An increasing step input followed by a decreasing step to the throttle angle command is given to the model to cancel out the speed decrease caused by the variation in the load torque disturbance. The spark advance is maintained at 25 deg bTDC.

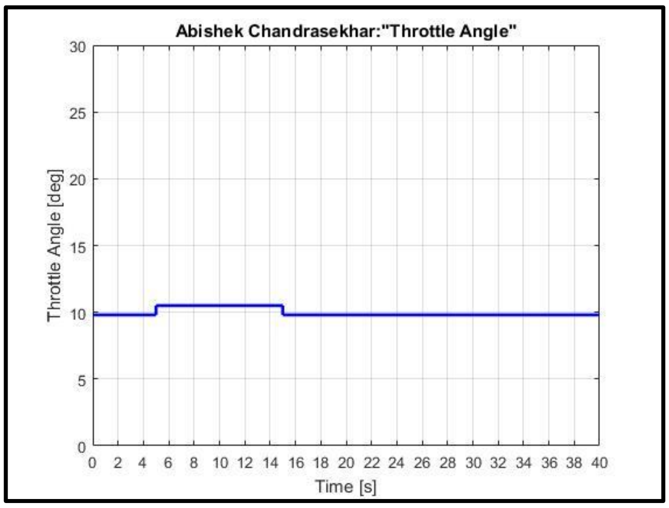

Figure 7 - Feedforward Throttle signal

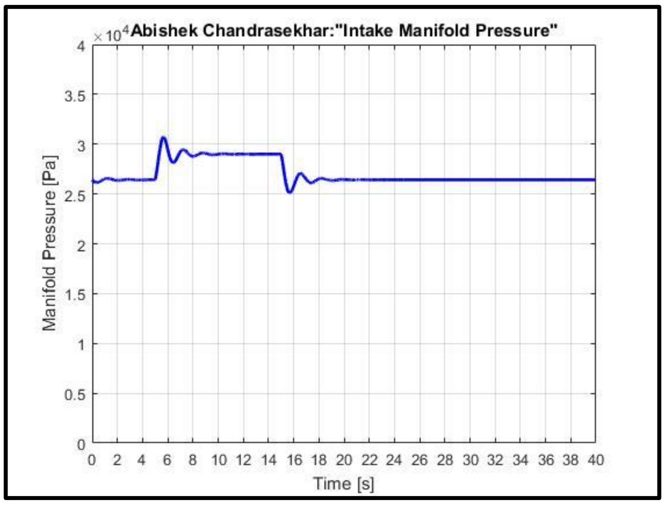

Figure 8 - MAP response to feedforward throttle signal

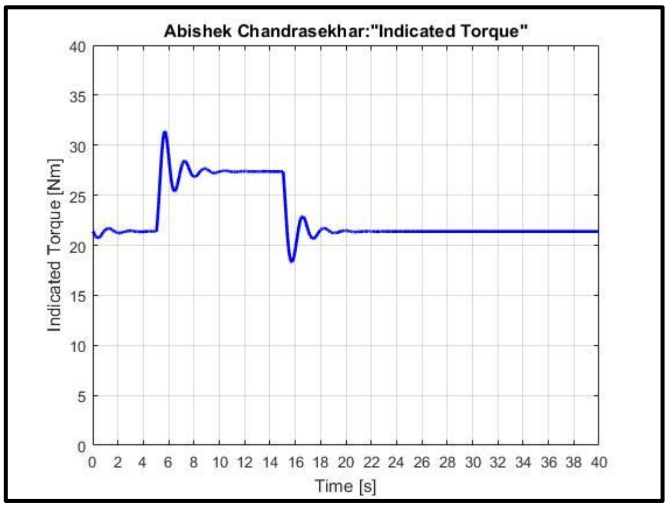

Figure 10 - Indicated torque for Throttle input

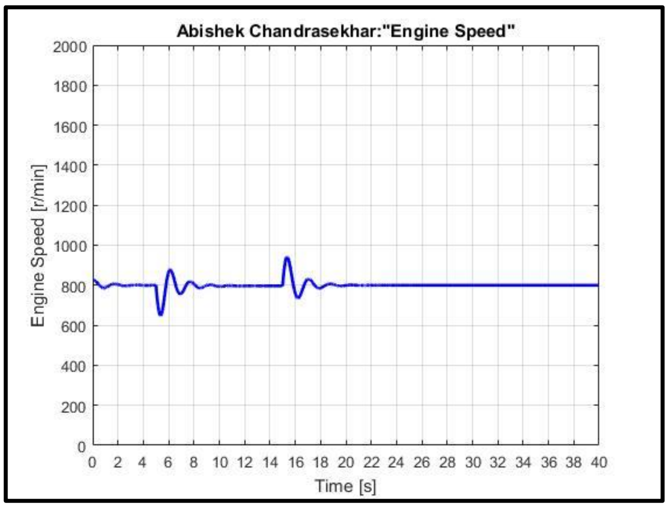

Figure 9 - Engine speed response to feedforward throttle signal

Inference:

- The **throttle angle** is increased from 9.8 to **10.5 degrees** in order to compensate for the decrease in engine speed caused by the change in external torque.
- This temporary increase in throttle angle attempts to **stabilize the engine speed,** the **intake manifold pressure** and the **indicated torque** during the load torque variation.
- The **engine speed** settles down at **800 rpm**, after fluctuating during change in external torque. The period where the load is 16 Nm, the engine speed is 800rpm.

3.7 Feedforward control using Spark Advance angle:

The same control technique is implemented here using the spark advance angle instead of the throttle angle. The throttle angle is kept at a constant corresponding to the third case in Table 4 where the load is 10 N-m.

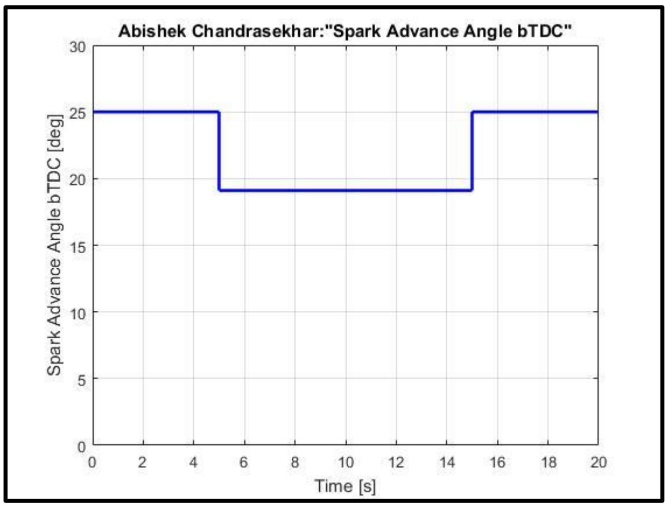

Figure 14 - Spark Advance angle feedforward signal

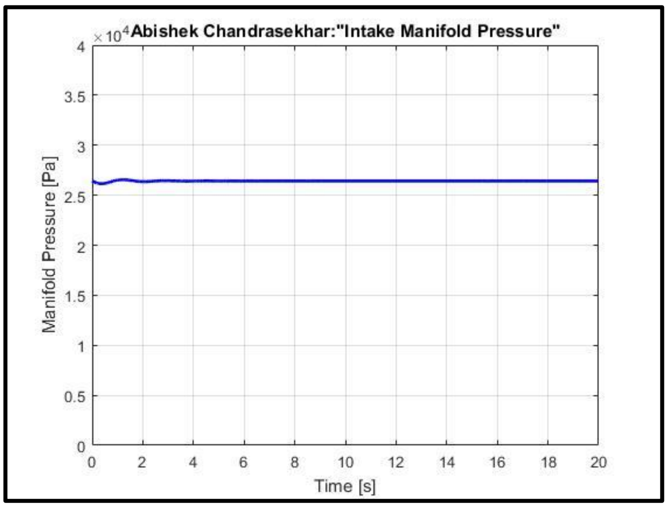

Figure 13 - MAP response to feedforward spark advance

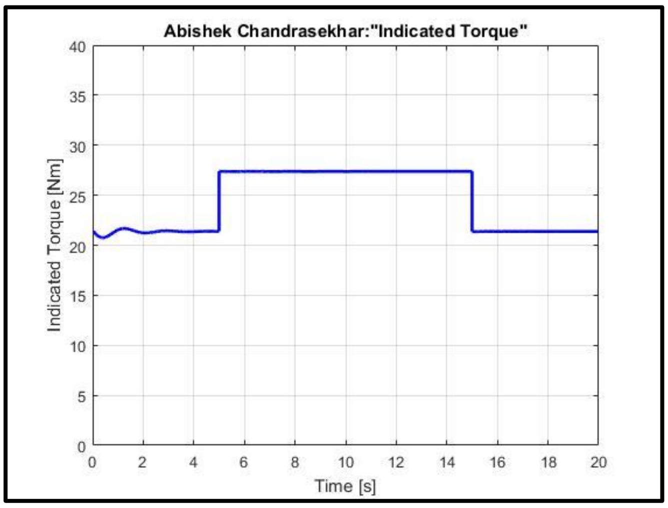

Figure 11 - Indicated torque for feedforward spark advance

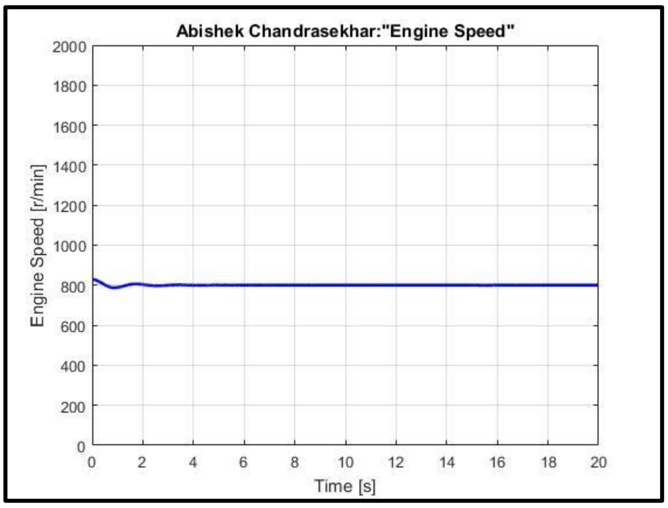

Figure 12 - Engine speed for feedforward spark advance signal

Inference:

- **The spark angle** is reduced from 25 to **19.11 degrees** in order to compensate for the disturbances due to the external torque.
- The intake **manifold pressure** remains **constant** more or less throughout.
- The **indicated torque** increases and decreases abruptly due to the changes in external loads and the spark angle. There are **no fluctuations** as in the previous problems.
- The **speed of the engine** (rpm) remains **constant** throughout the process, which implies that the change in **spark advance angle** has **successfully compensated** for the external torque disturbances.

Compared to the previous method of trying to compensate for the external torques, the spark advance angle changes result in the **most stable** simulation.

4. Modeling for Idle Speed Control:

For the Idle speed control problem, feedback control is normally the most efficient control technique. For feedback, it is significantly easier to design for linear discrete models. The engine Simulink model that was previously studied will be converted to a linear discrete model in the crank angle domain instead of time domain. The sampling time for this model will be $\tau_\theta = \pi/3$.

4.1 Equations of the Engine:

The seven equations of the engine that were used to create the Simulink model are linearized and discretized. The original equations are mentioned in Section 2.

4.2 Conversion to Crank Angle Domain:

The first step involved in discretizing the model is converting the idle speed dynamics to the crank- angle domain. This is done by 1) Applying chain rule and 2) converting revs/m to rad/sec.

Chain rule: $\frac{d}{dt} = \frac{d}{d\theta}\frac{d\theta}{dt}$, where $\frac{d}{d\theta} = \omega$. Which implies $\frac{d}{dt} = \frac{1}{\omega}\frac{d}{dt}$

Converting to radians per second: $\omega = N * \frac{\pi}{30}$.

The above calculations are applied to obtain a set of equations in the crank-angle domain. The manifold pressure, the indicated torque and the engine speed are given by:

$$\frac{dP_{IM}(\theta)}{d\theta} = \frac{K_{p1}}{\omega(\theta)} K_{thr}(a_2\alpha(\theta)_{thr}^2 + a_1\alpha(\theta)_{thr} + a_0) - K_{p2}P_{IM}(\theta) + K_{p3}$$

$$T_{ind}(\theta) = K_T\{b_1 P_{IM}(\theta - \theta_d) - b_0\}\{c_2\vartheta_{spark}^2 + c_1\vartheta_{spark} + c_0\}$$

$$\frac{d\omega(\theta)}{d\theta} = \frac{1}{J}\left[\frac{T_{ind}}{\omega(\theta)} - \frac{T_{load}}{\omega(\theta)} - \left(K_{fr1}\omega(\theta) + K_{fr2} + \frac{K_{fr3}}{\omega(\theta)}\right)\right]$$

where:

$$K_{thr} = \frac{p_{amb}}{\sqrt{RT_{amb}}} \cdot \sqrt{\gamma} \cdot \sqrt{\left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}}$$

$$K_{p1} = \frac{R \cdot T_{amb}}{V_{IM}}; \quad K_{p2} = \frac{s_i V_d}{4\pi V_{IM}}; \quad K_{p3} = \frac{y_i V_d}{4\pi V_{IM}}$$

$$K_T = \frac{10^5 V_d}{4\pi}; \quad \theta_d = \frac{4\pi}{Z}$$

$$K_{fr1} = \left(\frac{30}{\pi}\right)^2 d_2; \quad K_{fr2} = \frac{30}{\pi} d_1; \quad K_{fr3} = d_0$$

4.3 Linearization of the model:

The next step is to find a linear approximation of the system. This is done using the Taylor series approximation, for which, the equilibrium points are obtained. They are as follows:

Pressure P_0 = 26423.6 Pa; Indicated Torque T_{ind0} = 21.3766 Nm;

Speed N_0 = 801.205 rpm (or) ω_0 = 83.901 rad/sec.

The linearized system is as follows:

$$\frac{d\delta p}{d\theta} = -K_3\delta p - \frac{K_2}{\omega_0^2}\delta\omega + \frac{K_1}{\omega_0}\delta\alpha \qquad \textbf{\textit{Equation 1}}$$

$$\delta T = K_p\delta p(\theta - \theta_d) + K_\vartheta\delta\vartheta \qquad \textbf{\textit{Equation 2}}$$

$$\frac{d\delta\omega}{d\theta} = -\frac{K_f}{\omega_0^2}\delta\omega + \frac{1}{J\omega_0}\delta T - \frac{1}{J\omega_0}\delta T_l \qquad \textbf{\textit{Equation 3}}$$

where

$$K_1 = K_{pl}K_{thr}(2a_2\alpha_0 + a_1) \qquad K_2 = K_{pl}K_{thr}(a_2\alpha_0^2 + a_1\alpha_0 + a_0)$$

$$K_3 = K_{p2} \qquad \theta_d = \frac{4\pi}{Z}$$

$$K_p = K_T b_1(c_2\vartheta_0^2 + c_1\vartheta_0 + c_0) \qquad K_f = 2K_{fr2}\omega_0^2 + K_{fr1}\omega_0$$

$$K_\vartheta = K_T(b_1 p_0 - b_0)(2c_2\vartheta_0 + c_1)$$

4.4 Discretized model of the plant:

The linearly approximated system is then discretized using bilinear transformation in order to obtain the state space equations and the transfer function of the system.
Using the following bilinear transformation techniques:

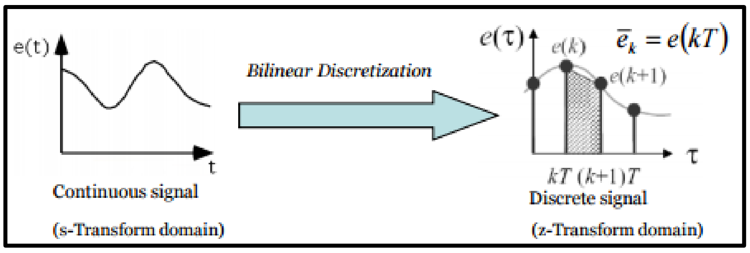

Figure 15 - Bilinear Transformation

$$\frac{d\delta x}{d\theta} = \frac{x(k\tau+\tau) - x(k\tau)}{\tau}\ ;\ \delta x = \frac{x(k\tau+\tau)+x(k\tau)}{2}$$

The system of equations become:

4.4.1 Engine breathing dynamics model equation:

$$\frac{d\delta p}{d\theta} = -K_3\delta p - \frac{K_2}{\omega_0^2}\delta\omega + \frac{K_1}{\omega_0}\delta\alpha$$

$$\frac{p(k\tau_\theta + \tau_\theta) - p(k\tau_\theta)}{\tau_\theta} = -K_3\frac{p(k\tau_\theta + \tau_\theta) + p(k\tau_\theta)}{2} - \frac{K_2}{\omega_0^2}\frac{\omega(k\tau_\theta + \tau_\theta) + \omega(k\tau_\theta)}{2} + \frac{K_1}{\omega_0}\frac{\alpha(k\tau_\theta + \tau_\theta) + \alpha(k\tau_\theta)}{2}$$

$$p(k+1) - p(k) = -\frac{\tau_\theta K_3}{2}(p(k+1) + p(k)) - \frac{\tau_\theta K_2}{2\omega_0^2}(\omega(k+1) + \omega(k)) + \frac{\tau_\theta K_1}{2\omega_0}(\alpha(k+1) + \alpha(k))$$

$$\left(1 + \frac{\tau_\theta K_3}{2}\right)(p(k+1) - p(k)) = -\frac{\tau_\theta K_2}{2\omega_0^2}(\omega(k+1) + \omega(k)) + \frac{\tau_\theta K_1}{2\omega_0}(\alpha(k+1) + \alpha(k))$$

- Equation 1 transforms to:

$$(C_{T1}z + C_{T2})p(z) = -C_{T3}(z+1)\omega(z) + C_{T4}(z+1)\alpha(z)$$ ***Equation 4***

$$\text{where: } C_{T1} = \left(1 + \frac{\tau_\theta K_3}{2}\right);\ C_{T2} = \left(\frac{\tau_\theta K_3}{2} - 1\right);$$

$$C_{T3} = \left(\frac{\tau_\theta K_2}{2\omega_0^2}\right);\ C_{T4} = \left(\frac{\tau_\theta K_3}{2\omega_0}\right);$$

4.4.2 Indicated torque equation:

The discretization of the indicated torque is immediate, since the sampling time is an integer multiple of the combustion delay.

$$\delta\mathrm{T} = \mathrm{K_p}\delta\mathrm{p}(\theta - \theta_\mathrm{d}) + \mathrm{K}_\vartheta\delta\vartheta$$

- Equation 2 transforms to:

$$T(z) = z^{-3}K_p p(z) + K_\vartheta \vartheta(z)$$ ***Equation 5***

Engine crankshaft dynamics model equation:

$$\frac{\mathrm{d}\delta\omega}{\mathrm{d}\theta} = -\frac{\mathrm{K_f}}{\omega_0^2}\delta\omega + \frac{1}{\mathrm{J}\omega_0}\delta\mathrm{T} - \frac{1}{\mathrm{J}\omega_0}\delta\mathrm{T_l}$$

$$\frac{\omega(k\tau_\theta+\tau_\theta)-\omega(k\tau_\theta)}{\tau_\theta} = -\frac{K_f}{J\omega_0^2}\frac{\omega(k\tau_\theta+\tau_\theta)+\omega(k\tau_\theta)}{2} + \frac{1}{J\omega_0}\frac{T(k\tau_\theta+\tau_\theta)+T(k\tau_\theta)}{2} - \frac{1}{J\omega_0}\frac{T_l(k\tau_\theta+\tau_\theta)+T_l(k\tau_\theta)}{2}$$

$$\omega(k+1)-\omega(k) = -\frac{\tau_\theta K_f}{2J\omega_0^2}(\omega(k+1)+\omega(k)) + \frac{\tau_\theta}{2J\omega_0}(T(k+1)+T(k)) - \frac{\tau_\theta}{2J\omega_0}(T_l(k+1)+T_l(k))$$

$$\left(1+\frac{\tau_\theta K_f}{2J\omega_0^2}\right)(\omega(k+1)-\omega(k)) = \frac{\tau_\theta}{2J\omega_0}(T(k+1)+T(k)) - \frac{\tau_\theta}{2J\omega_0}(T_l(k+1)+T_l(k))$$

- Equation 3 transforms to:

$$(C_{T5}z + C_{T6})\omega(z) = C_{T7}(z+1)T(z) - C_{T8}(z+1)T_l(z)$$ ***Equation 6***

$$\text{where: } C_{T5} = \left(1 + \frac{\tau_\theta K_f}{2J\omega_0^2}\right);\ C_{T6} = \left(\frac{\tau_\theta K_f}{2J\omega_0^2} - 1\right);$$

$$C_{T7} = \left(\frac{\tau_\theta}{2J\omega_0}\right);\ C_{T8} = \left(\frac{\tau_\theta}{2J\omega_0}\right);$$

The final form of the equations are **Equation 4, Equation 5 and Equation 6**.

4.5 System of Linear Discrete Equations:

Manifold Absolute Pressure: $(C_{T1}z + C_{T2})p(z) = -C_{T3}(z+1)\omega(z) + C_{T4}(z+1)\alpha(z)$

Indicated Torque: $T(z) = z^{-3}K_p p(z) + K_\vartheta \vartheta(z)$

Engine Speed: $(C_{T5}z + C_{T6})\omega(z) = C_{T7}(z+1)T(z) - C_{T8}(z+1)T_l(z)$

where: $C_{T1} = \left(1 + \frac{\tau_\theta K_3}{2}\right)$; $C_{T2} = \left(\frac{\tau_\theta K_3}{2} - 1\right)$;$C_{T3} = \left(\frac{\tau_\theta K_2}{2\omega_0^2}\right)$; $C_{T4} = \left(\frac{\tau_\theta K_3}{2\omega_0}\right)$;

$$C_{T5} = \left(1 + \frac{\tau_\theta K_f}{2J\omega_0^2}\right) ; C_{T6} = \left(\frac{\tau_\theta K_f}{2J\omega_0^2} - 1\right) ;C_{T7} = \left(\frac{\tau_\theta}{2J\omega_0}\right); C_{T8} = \left(\frac{\tau_\theta}{2J\omega_0}\right) ;$$

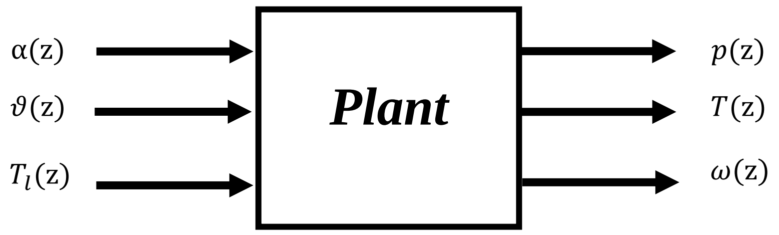

Figure 16 - Plant Model

4.6 System Transfer Function for SISO control:

The plant is reduced for a single input and single output (SISO) control. The input parameter is the throttle position and the output parameter is the engine speed.

The **spark angle input ϑ is omitted** from the system of equations. Equation 5 becomes

$$T(z) = z^{-3}K_p p(z)$$ ***Equation 7***

⇨ The load torque input T_l is omitted from the system as well. Equation 6 becomes

$$(C_{T5}z + C_{T6})\omega(z) = C_{T7}(z+1)T(z)$$ ***Equation 8***

The system transfer function G(z) is given as a ratio of the output to the input.

$$G(z) = \frac{speed}{throttle\ position} = \frac{\omega(z)}{\alpha(z)}$$

Equation 8 can be represented as $\omega(z) = \frac{-C_{T7}(z+1)T(z)}{(C_{T5}z+C_{T6})}$ ***Equation 9***

From Equation 7, we get $p(z) = \frac{(C_{T5}z+C_{T6})}{C_{T7}K_p z^{-3}(z+1)}\omega(z)$ ***Equation 10***

Substituting Equations 9 and Equation 10 in Equation 4, we solve for $\alpha(z)$.

Finally, we get the transfer function as:

$$\frac{\alpha(z)}{\omega(z)} = \frac{C_{T4}C_{T7}K_p(z^2 + 2z + 1)}{C_{T1}C_{T5}z^5 + (C_{T1}C_{T6} + C_{T2}C_{T5})z^4 + C_{T2}C_{T6}z^3 + C_{T3}C_{T7}K_p(z^2 + 2z + 1)}$$

4.7 Pole-Zero Map of the transfer function:

Once the SISO model equations and transfer function are defined, the pole-zero map of the system is plotted to understand the characteristics of the system i.e., to check whether there are unstable poles.

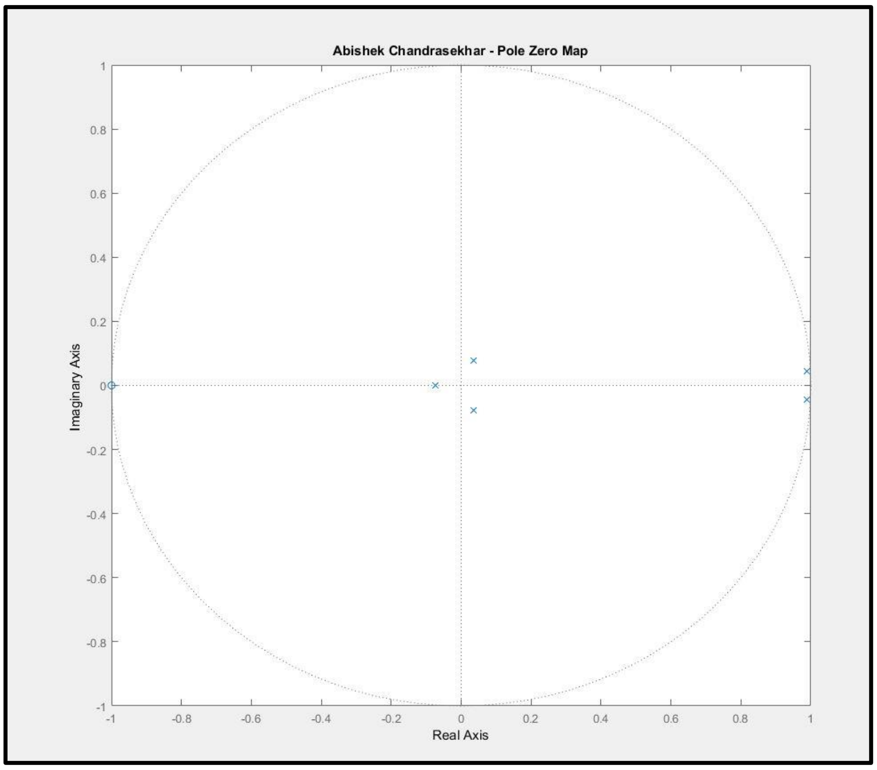

Figure 17 - Pole-Zero Map of the System

4.7.1 Poles:

The open loop poles of the system are:

```
0.988009115316149 +    0.0461119173915378i;
0.988009115316149 -    0.0461119173915378i;
0.035366751266058 +    0.0800193467122896i;
0.035366751266058 -    0.0800193467122896i;
-0.075407003787125 +                    0i;
```

It can be seen that all the poles lie within the unit circle. There is one pair of poles that is near the unit circle.

4.7.2 Zeros:

There is only one zero in the system, it lies at -1. On the unit circle.

4.7.3 Open Loop Transfer Function:

After the constants are defined, the open loop transfer function is as follows:

```
            0.008723 z^2 + 0.01745 z + 0.008723
  ----------------------------------------------------------------------
  1.015 z^5 - 2 z^4 + 0.9855 z^3 + 0.0005728 z^2 + 0.001146 z + 0.0005728
```

4.8 Simulink Model:

The open loop system block diagram is built using discrete blocks in Simulink. The diagram must be built so that a torque load disturbance can be applied; that is, instead of implementing the discretized and *Z*-transformed version of

$$\frac{d\delta\omega}{d\theta} = -\frac{K_f}{\omega_0^2}\delta\omega + \frac{1}{J\omega_0}\delta T$$

$$\frac{d\delta\omega}{d\theta} = -\frac{K_f}{\omega_0^2}\delta\omega + \frac{1}{J\omega_0}\delta T - \frac{1}{J\omega_0}\delta T_l$$

The equations to be implemented in Simulink are:

$$\mathrm{p(z)} = \frac{-\mathrm{C_{T3}(z+1)\omega(z)}}{\mathrm{(C_{T1}z + C_{T2})}} + \frac{\mathrm{C_{T4}(z+1)\alpha(z)}}{\mathrm{(C_{T1}z + C_{T2})}}$$

$$\mathrm{T(z) = z^{-3}K_p p(z)}$$

$$\omega(z) = \frac{C_{T7}(z+1)T(z)}{(C_{T5}z + C_{T6})} - \frac{C_{T8}(z+1)T_l(z)}{(C_{T5}z + C_{T6})}$$

- The Manifold pressure output is the output from the first block. It is the input to second transfer function. Similarly, the Indicated Torque is the output from the second block is given as input to the final transfer function which outputs Speed in rad/sec.
- As the output is required to be plotted in units of rpm, the output speed in rad/sec is converted into revs/min my multiplying by a constant= 9.55
- Discrete transfer function blocks are used to implement the transfer function in each equation.
- The sample time for the blocks are Tau-d = pi/3.
- The Simulink model that was built is shown below:

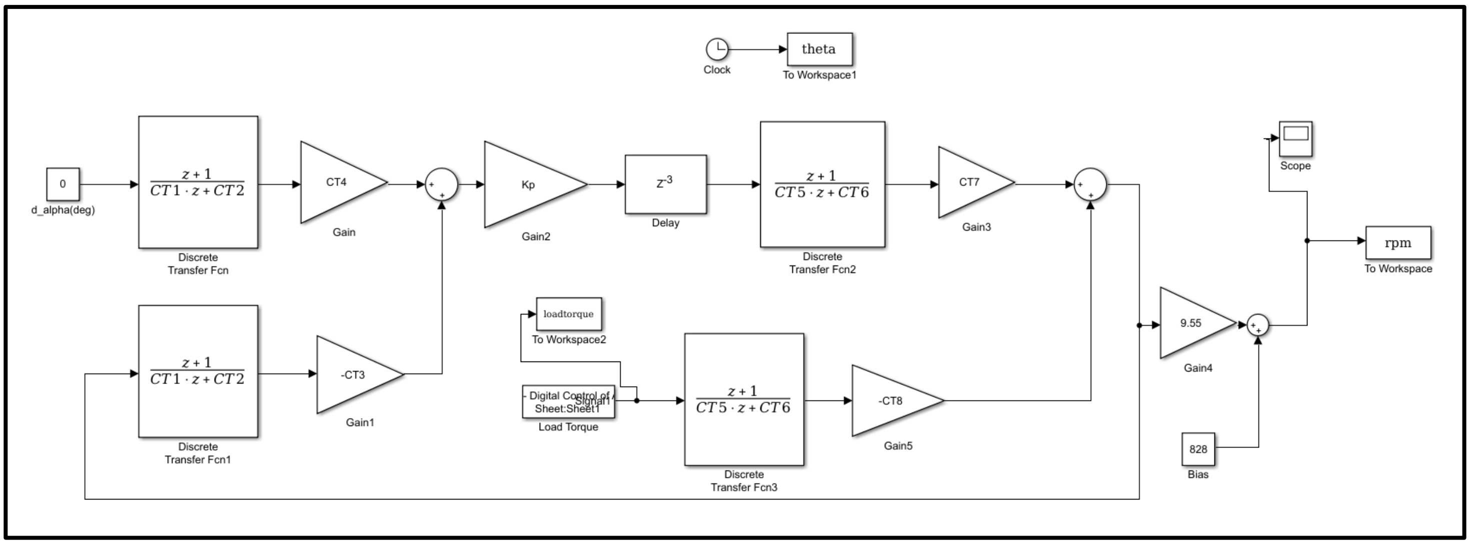

Figure 18 - Simulink Model built for SISO control

5. SISO Control:

The Simulink model is built in the previous section for SISO Control. In this, the control input is the throttle angle. The controller design runs on a discrete crank-angle domain (every π/3 radians of engine crankshaft rotation). A step disturbance of external torque as shown below is fed to the model while the spark advance angle is maintained at a constant value of 25 deg before TDC. The initial value (constant setpoint) of the throttle angle is set so that the engine speed is equal to 800 RPM without any disturbance applied.

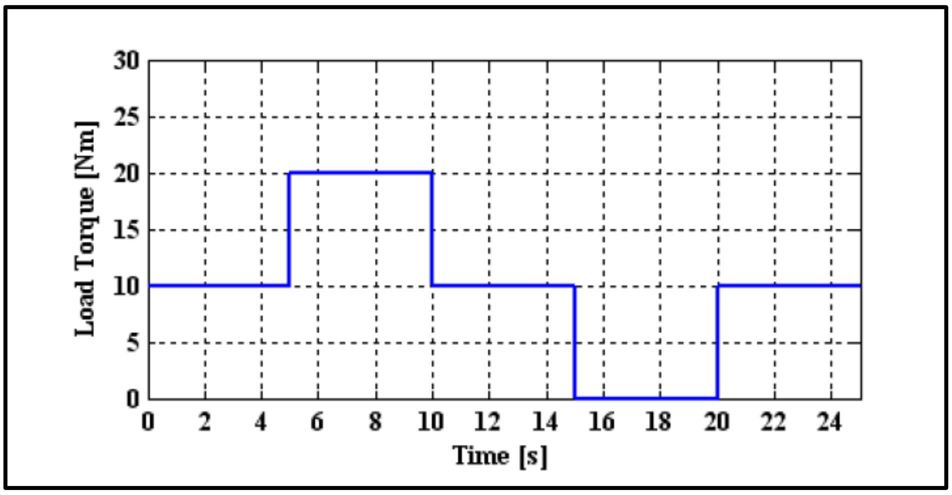

Figure 19 - Load torque to be applied to SISO model

5.1 PID Control:

- The PID controller for this problem is added to compensate for the load torque of signal above in Figure 19 that is fed to the engine model.
- The input to the PID controller is the change in speed of the engine [δω] (in rad/sec).
- The negative of that error is fed to the PID controller.
- The output of the PID controller is the change in throttle angle [δα] (in degrees), which is fed back to the input of the system.

5.1.1 Design of PID Control:

- The PID controller is designed and tuned using the frequency method.
- The Linearized Discrete Simulink model is used to tune the PID controller for a step load of 8Nm. This is done to obtain the gain values for the PID controller.
- This is done by implementing P, PI and then PID successively.

5.1.2 Proportional Control:

A Proportional Controller is directly proportional to the error from the output of the system.

$$u(t) = K\,e(t)$$

Where, u (t) is the used control; K is the controller gain; e (t) is the control error.
In this system, the error is the decrease in engine speed [δω] (in rad/sec).

- The value of K_p is increased marginally.
- After several increments, sustained oscillations were achieved at K = 0.216.
- Therefore, **K = 0.216 is the ultimate gain, K_{max}**.

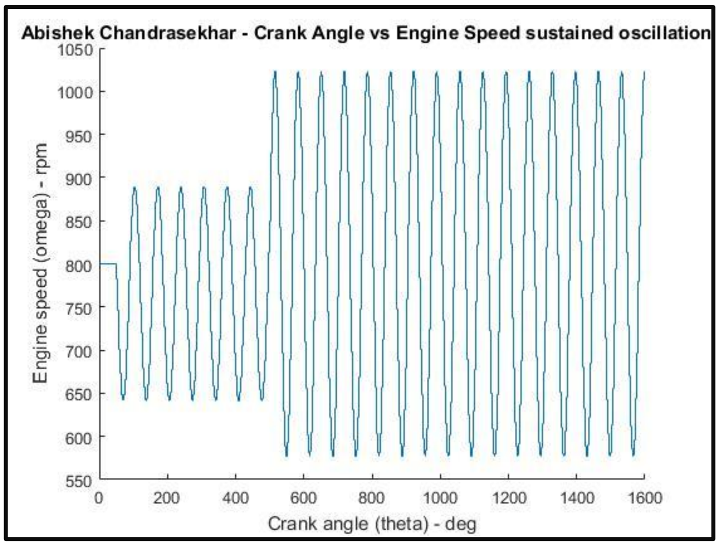

Figure 21 - Sustained oscillations using P controller

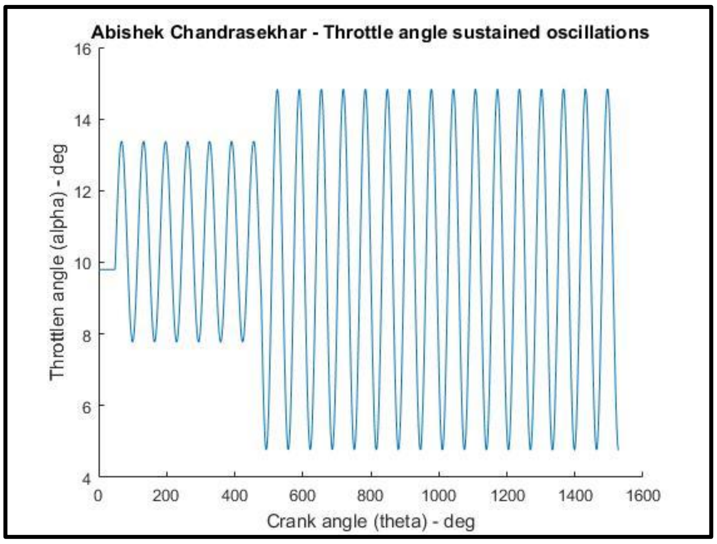

Figure 20 - Sustained oscillations of throttle angle

- Using the following formula from the frequency tuning method, K_p is found.

$$\mathbf{K_p = 0.5*K_{max} = 0.108}$$

- Implementing the Proportional control in the system, we get the following plots:

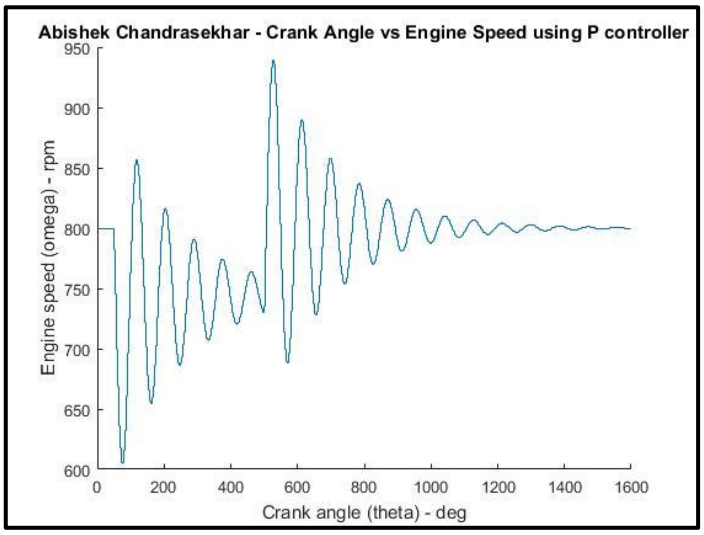

Figure 23 - Tuned P controller engine speed

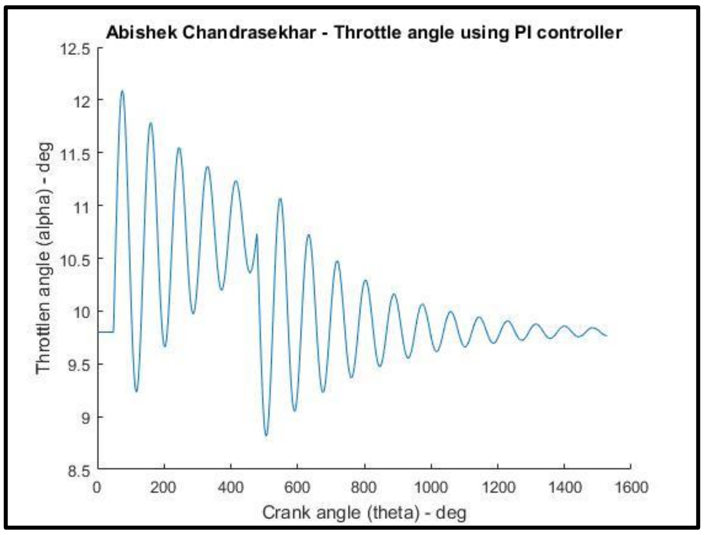

Figure 22 - Tuned P controller Throttle angle

- It can be seen from the above plots that the engine **speed does not fall below 500 rpm** and the **throttle angle remains between the 6 – 20-degree limit.**
- **Thus it can be seen that K_P = 0.108 is a valid gain for the Proportional Controller.**

5.1.3 PI controller:

The Integral part of the PI controller is used to obtain zero steady state error. Using the formula from frequency tuning method, the gain for the integrator, K_I is found out as follows:

$$K_p = 0.45K_{max};\ T_I = P_M/1.2$$

Where P_M = 60 (approx.) is the period of oscillation in the sustained oscillations.

Therefore, $\mathbf{K_P =}$ **0.0972** and $\mathbf{T_I}$ **= 129.6**; => $\mathbf{K_I}$ **= 0.00075**

Initial design of PI controller:

Using the values above, the PI controller is implemented in the system, which results in the following plots are obtained.

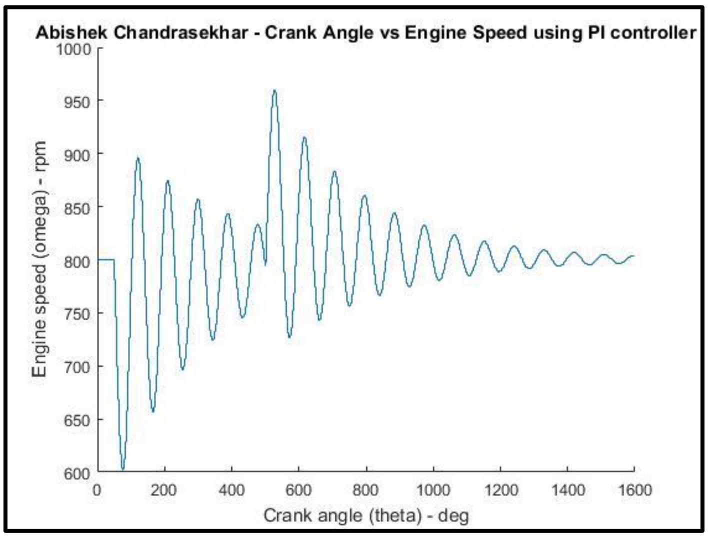

Figure 25 - Engine speed using PI control

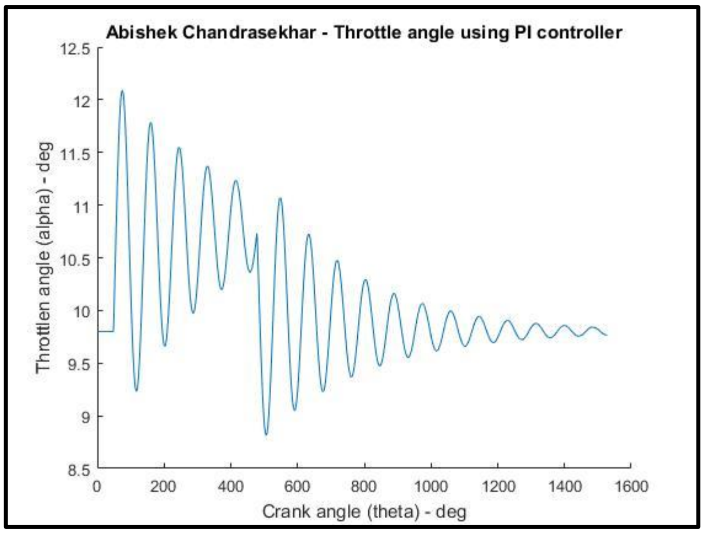

Figure 24 - Throttle angle using PI control

- It can be seen from the above plots that the engine **speed does not fall below 500 rpm** and the **throttle angle remains between the 6 – 20-degree limit.**
- **Thus it can be seen that K_P = 0.0972 and K_I = 0.00075 are valid gains for the Proportional Integral Controller.**

5.1.4 PID controller:

The derivative part of the PID controller predicts the change in error. It improves the stability of the system. The gains for the Derivative part K_D is found using the frequency tuning method.

$$K_p = 0.6*K_{max};\ T_I = P_M/2\ ;\ T_I = P_M/8$$

Therefore, **K_P = 0.1296** and **T_I = 43.2**; => **K_I = 0.003; T_D = 11.574; => K_D = 1.49972**

Initial design of PI controller:

Using the values above, the PI controller is implemented in the system, which results in the following plots are obtained.

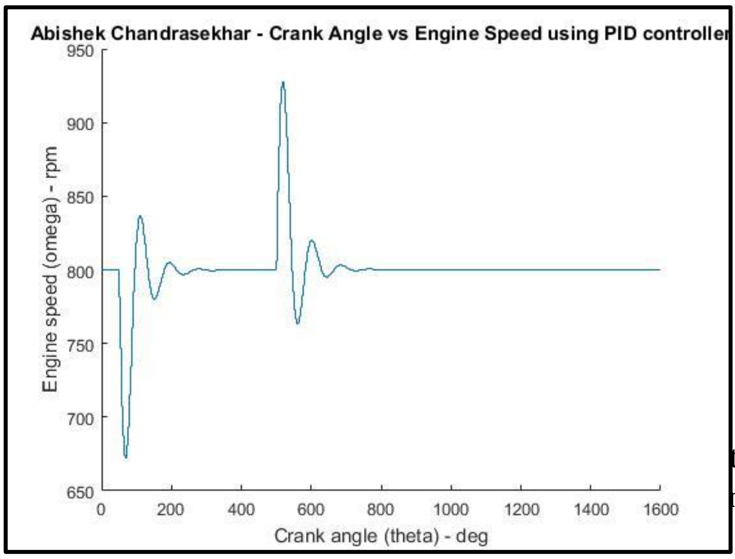

Figure 27 - Initial PID design -Engine speed

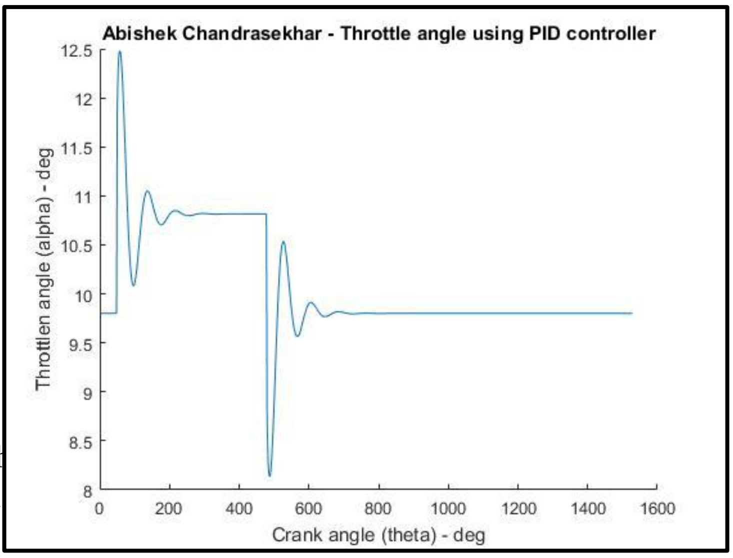

Figure 26 - Initial PID design - Throttle angle

- The steady state error is zero (i.e., the engine speed reaches 800 rpm) and settling time is 150 rpm (approx.)

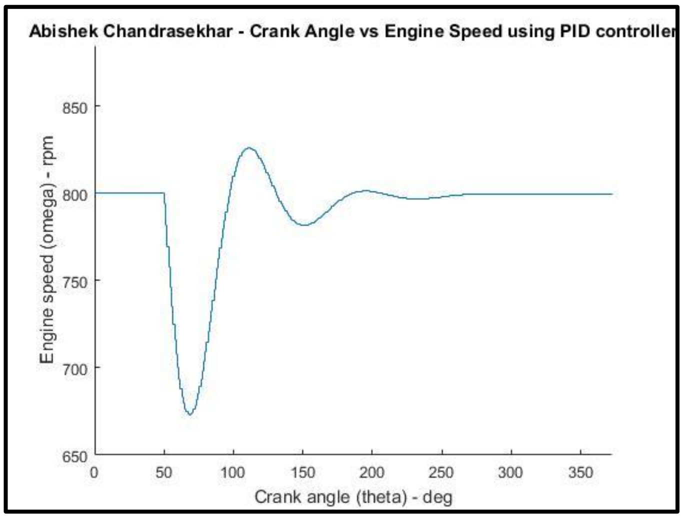

Figure 28 - Tuning PID to reduce settling time, steady state error

- **Thus it can be seen that K_p = 0.0972, K_I = 0.00075 and K_D = 1.49972 are valid gains for the Proportional Integral Derivative Controller.**

Tuned PID controller:
After tuning the values of the gains, the responses of the PID controller as follows:
The gains are:
T_I = 45; T_D = 13.5; Kp = 0.0972; K_I = 0.00288; K_D = 1.7496;

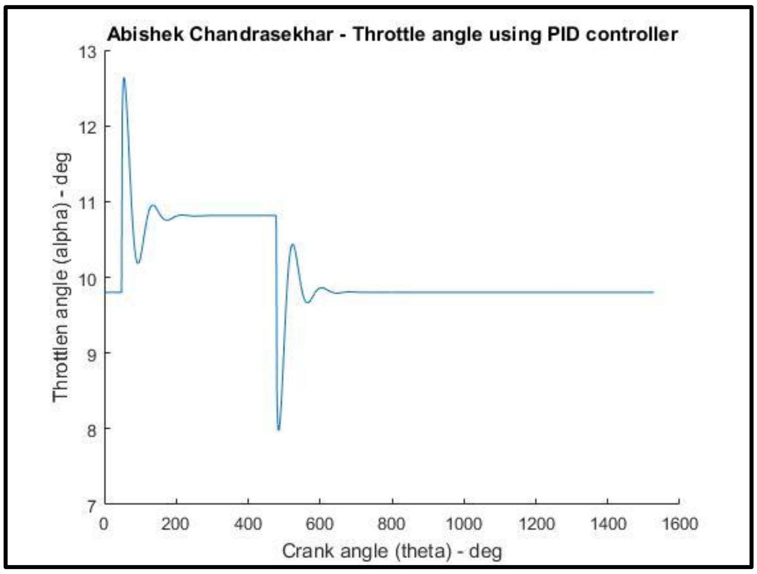

Figure 30 - Throttle angle using PID

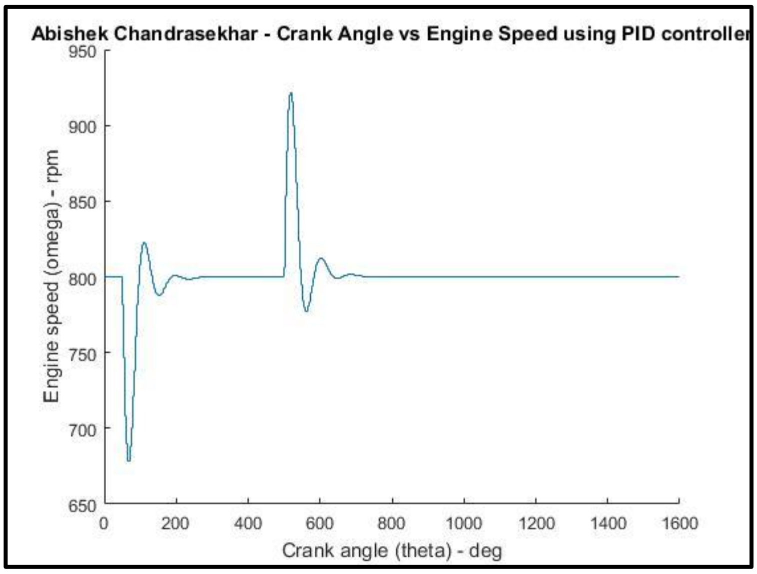

Figure 29 - Engine speed using PID

Final values of gains:
- Thus it can be seen that K_p = 0.0972, K_I = 0.00288, K_D = 1.7496; are final for the Proportional Integral Derivative Controller.
- This reduces the settling time to 130.

5.1.5 PID implementation for Linearized Discrete model:

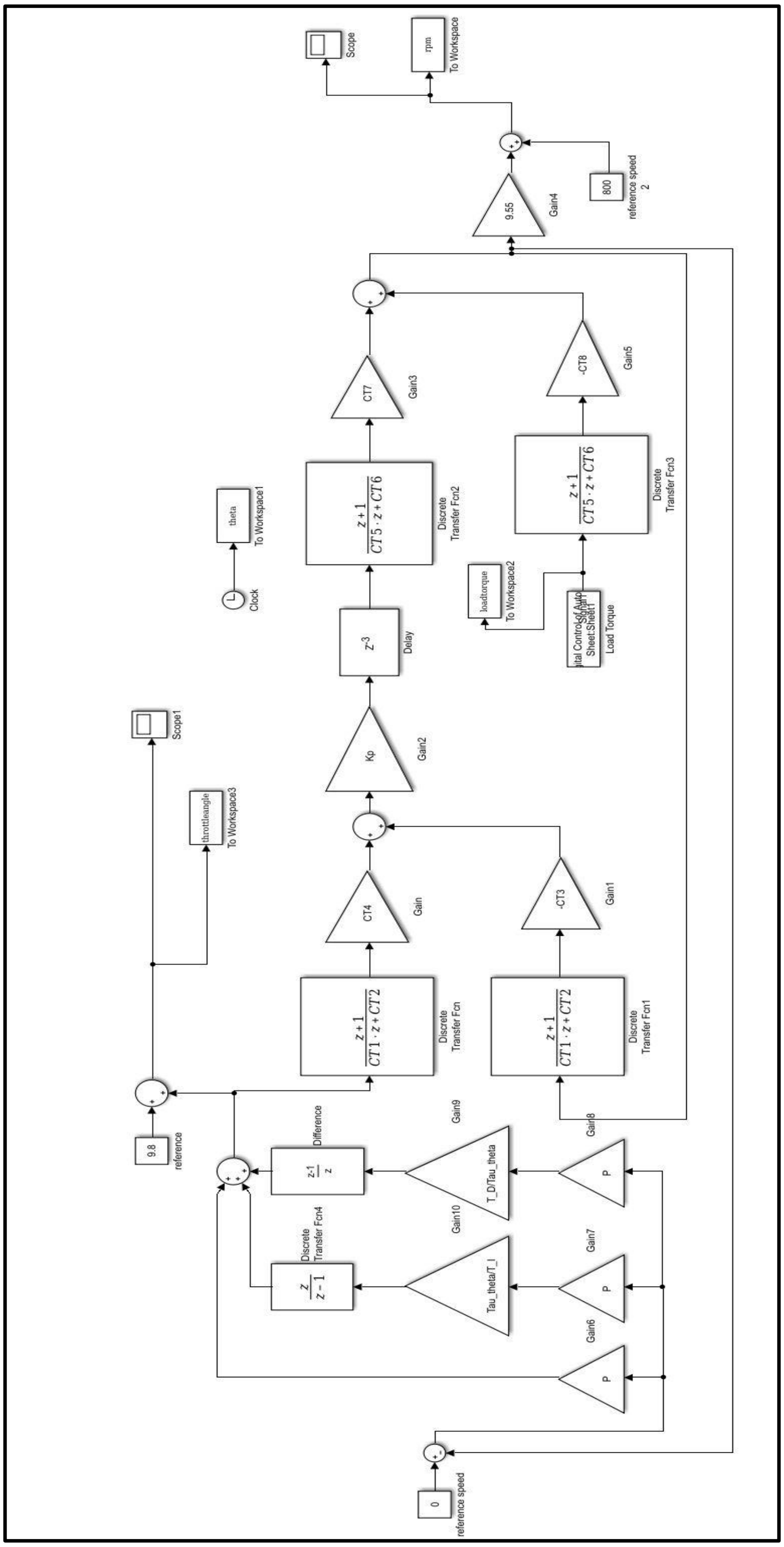

Figure 31 - PID implementation in linear discrete model

5.1.6 Implementation of PID in Engine Simulator:

The PID controller that is built and tuned for the linear discrete engine model is now implemented in the continuous model. The same load of 8N-m is given to the Simulator to see the response of the engine.

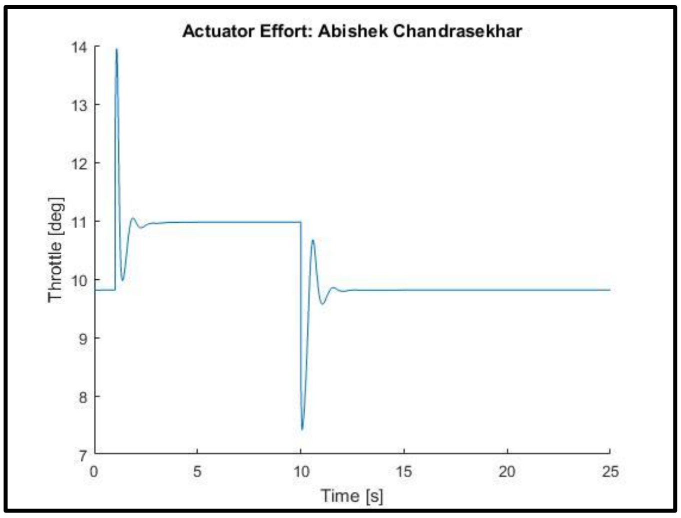

Figure 33 – Actuator effort using PID in simulator

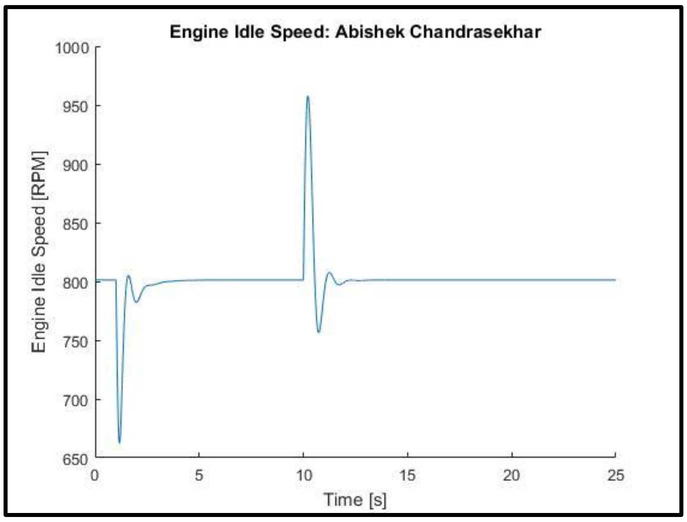

Figure 32 - Engine speed in Discrete model

- It can be seen that the excursions in the continuous model are more when the discrete controller is implemented compared to the discrete model.
- This is due to the discretization; the controller is built for a discrete model.

5.1.7 PID control for the load given:

- The load given in Figure 19 is given to the simulator.
- The initial gains used are those from the discrete model. They are given below:

  K_p= 0.0972; T_I= 45, K_i= 0.00288; T_D= 13.5, K_d= 1.7496
- The gains are further tuned in order to suit the continuous model. The final values of the gains are: **K_p= 0.1296;T_I= 45.2, K_i= 0.00286; T_D= 14.57, K_d= 1.888272**
- The response of the engine is given below.

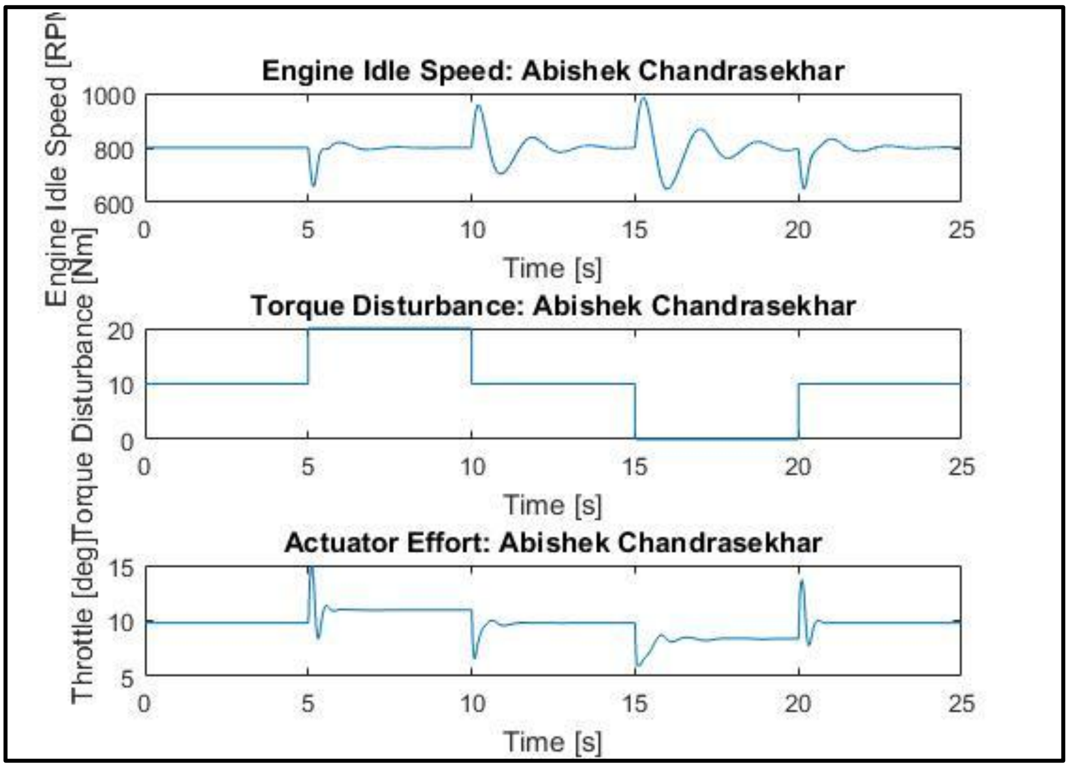

Figure 34 - Final PID design implementation in Simulator

5.2 Root Locus Design:

A controller is designed for the Linearized Discrete model using Root Locus Techniques. The Root Locus control is then implemented in the engine simulator.

5.2.1 Design Procedure:

- To design a controller for the system using Root Locus Design techniques, the Pole-Zero Map of the uncompensated system is first plotted:

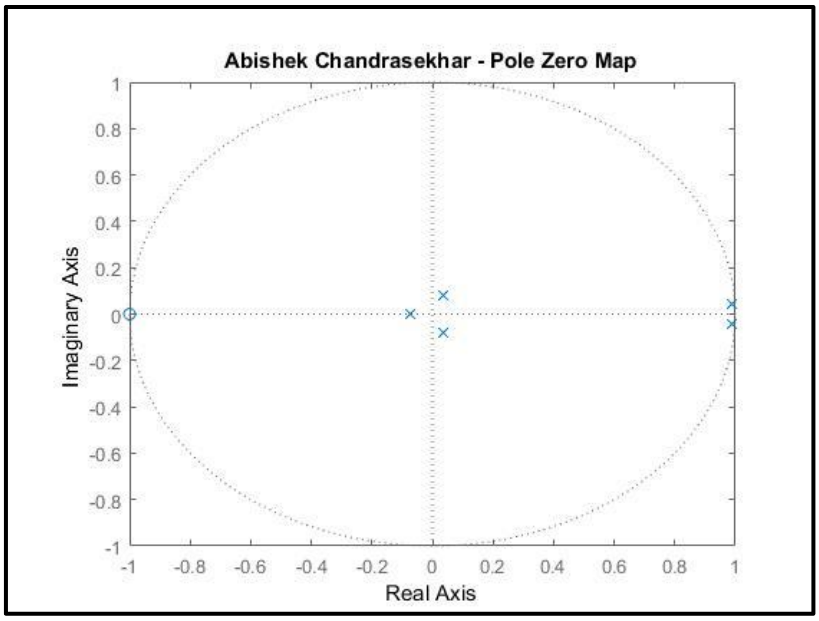

Figure 35 - Pole Zero Map

- It can be seen from the above plot that there are five poles and one zero for the uncompensated system. Out of which, there are two poles **at 0.988+0.0461i and 0.988-0.0461i and one zero at -1**.
- The root locus of the uncompensated system is given below:

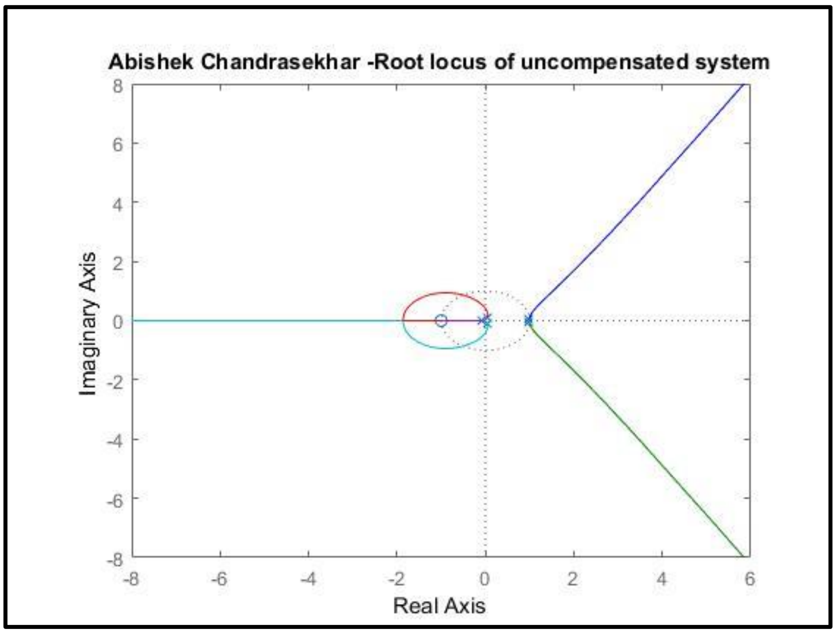

Figure 36 - Root Locus of uncompensated system

5.2.2 Initial Design of Root Locus:

- To stabilize the system, the poles **0.988+0.0461i and 0.988-0.0461i** are to be moved inwards, so in the compensator, **two zeros are added at or near the same location**.
- There is a zero at the unit circle. To move this zero inwards, in the compensator system a **pole 1** is added to the denominator of the compensator and another **pole of 0.5** is added to move the zero inwards.
- The compensator system

$$D(z) = \frac{(z - 0.988 + 0.0461i)(z - 0.988 - 0.0461i)}{(z + 1)(z + 0.6)}$$

- This compensator results in the following plots:

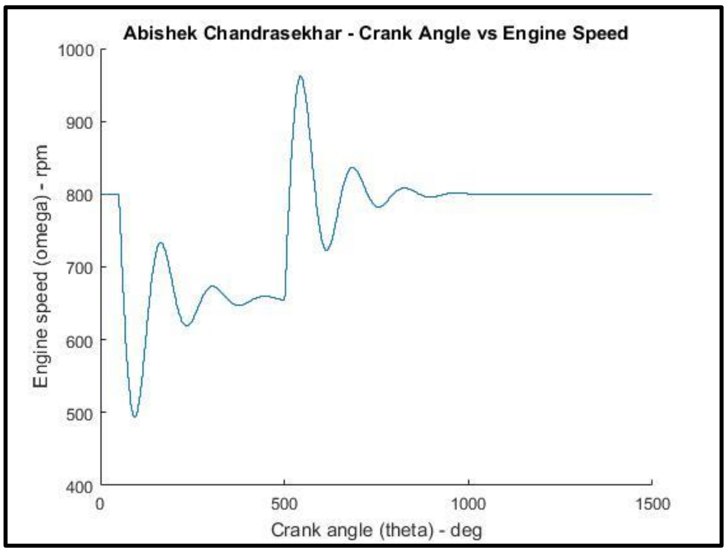

Figure 37 - Engine speed using initial root locus design

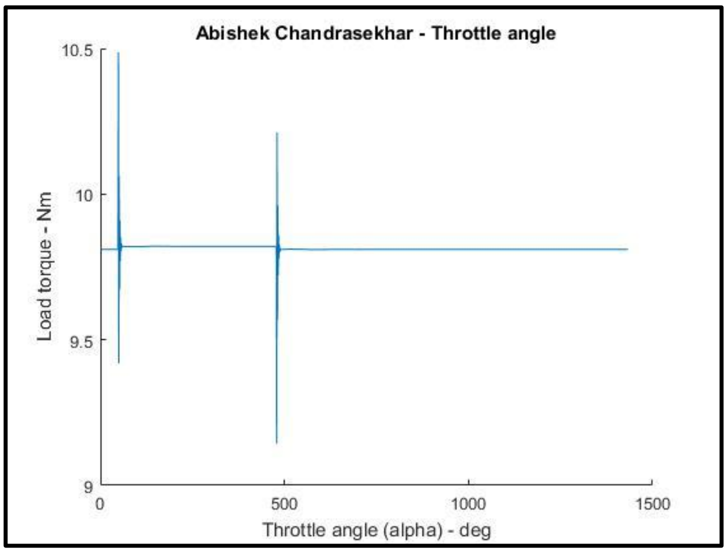

Figure 38 - Throttle angle using Initial Root Locus design

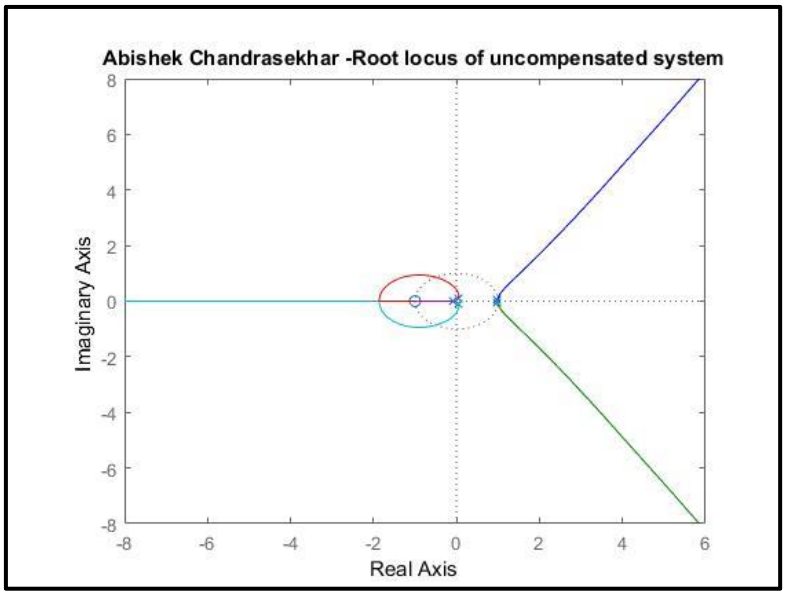

Figure 39 - Root locus of uncompensated system

- The throttle angle response is within the limits (between 6 and 20 degrees)
- There is a steady state error in the engine speed.
- So, **this compensated system is not valid.**

5.2.3 Root Locus tuning:

Using the rltool in MATLAB, the root locus that gives a proper stable response is acquired. The numerator (zeros) of the compensator are the same, but the denominator is different.

$$D(z) = \frac{(z - 0.988 + 0.0461i)(z - 0.988 - 0.0461i)}{(z + 0.9996)(z - 0.6733)}$$

Implementing this controller in the system, we obtain the following plots:

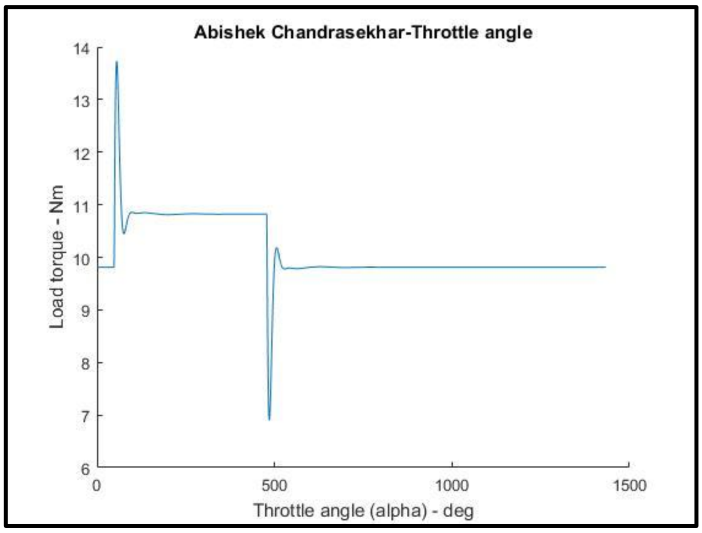

Figure 41 - Throttle angle using tuned Root Locus

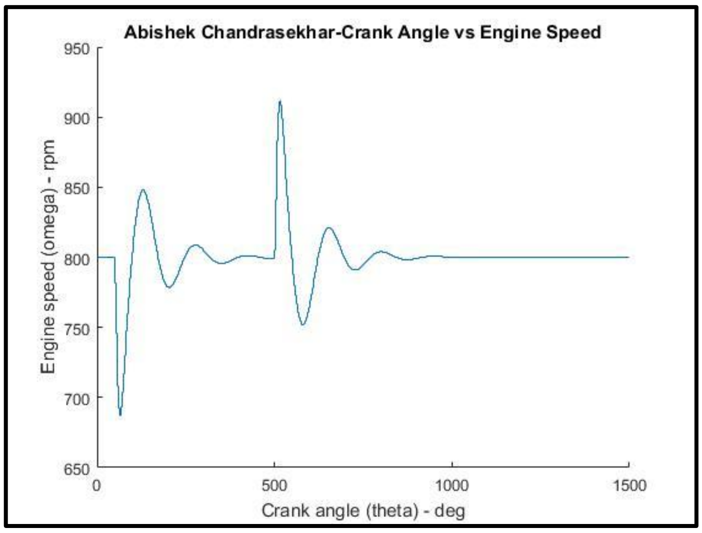

Figure 40 - Engine speed using tuned Root Locus

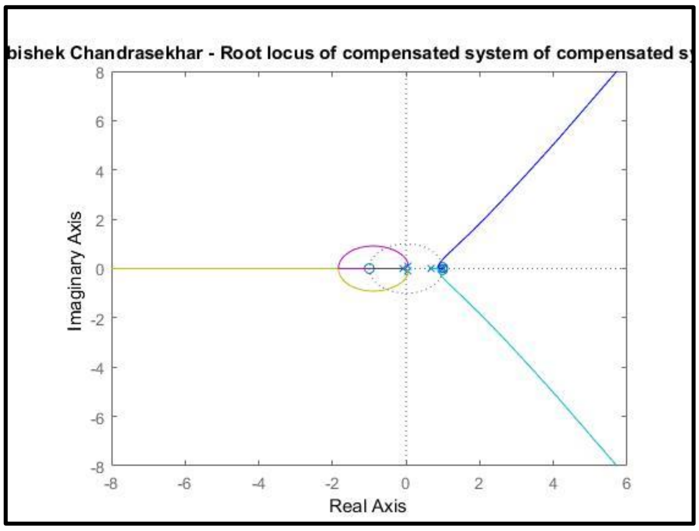

Figure 43 - Root locus of Compensated system

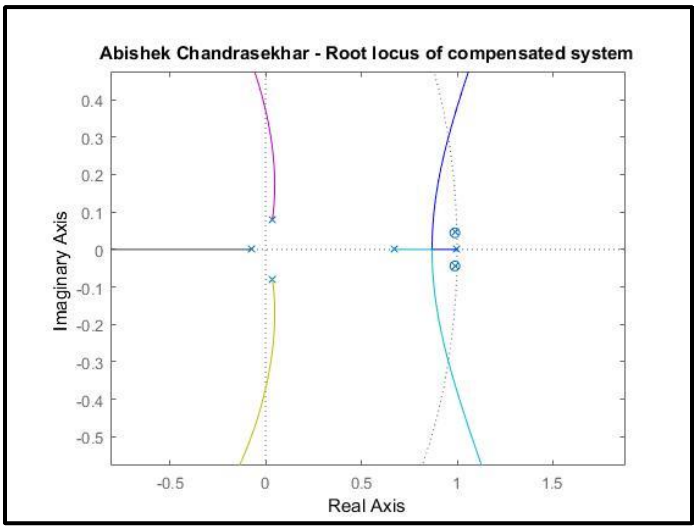

Figure 42 - Root locus of compensated system

- It can be seen clearly that the compensator satisfies the given conditions.
- The engine speed **settles at 5% of the original value (800 rpm).**
- The throttle **angle is between 6 and 20 degrees.**
- Therefore, this compensator

$$\boldsymbol{D}(z) = \frac{(z - 0.988 + 0.0461i)(z - 0.988 - 0.0461i)}{(z + 0.9996)(z - 0.6733)}$$

is a valid compensator for the engine idle speed control.

The gain constant for this system is taken as 1.

5.2.4 Implementation in Simulator:

The controller developed using Root Locus techniques is implemented in the Continuous model. The response of the engine is as follows:

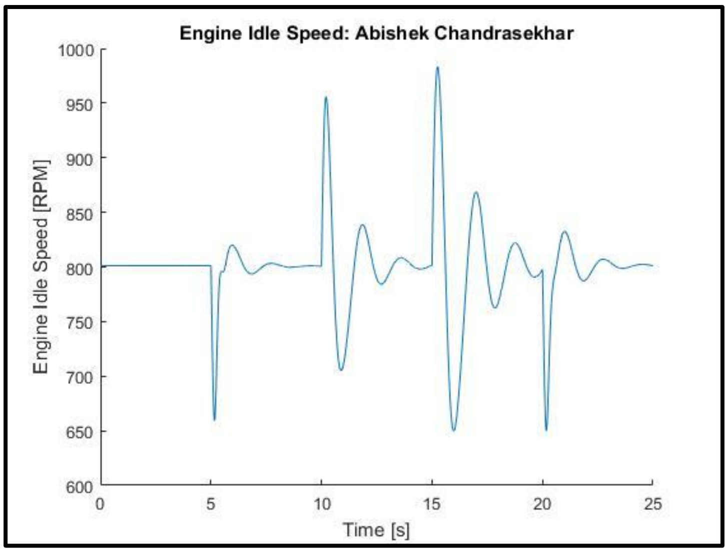

Figure 45 - Engine Speed using Root Locus Controller

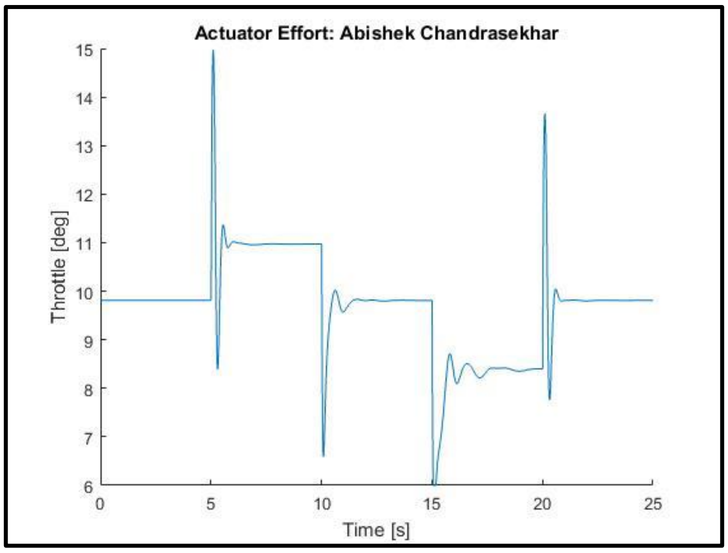

Figure 44 - Throttle angle using Root Locus Controller

- It can be seen from the plots that the root locus controller can stabilize the system.
- The idle speed of the engine is more or less maintained.
- The discrete model has lesser excursions as compared to the continuous model. There is large overshoot in the continuous model. The settling time is also larger for the continuous model.

5.3 SISO control using state feedback:

To implement SISO control using state feedback, the state space model of the system is built and then the desired closed loop poles are placed in the system either using the pole placement algorithm or using LQR.

5.3.1 Building the state space model of the system:

- Defining the state matrices for the discrete system as follows:

$$\Phi_T = \begin{bmatrix} (OM^{-1})_{11} & (OM^{-1})_{12} & 0 & 0 & Q_{12}K_p \\ (OM^{-1})_{21} & (OM^{-1})_{22} & 0 & 0 & Q_{22}K_p \\ (M^{-1})_{11} & (M^{-1})_{12} & 0 & 0 & -(M^{-1}N)_{12}K_p \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad \Gamma_T = \begin{bmatrix} Q_{11} \\ Q_{21} \\ -(M^{-1}N)_{11} \\ 0 \\ 0 \end{bmatrix}$$

$$H_T = \left[(M^{-1})_{21} \quad (M^{-1})_{22} \quad 0 \quad 0 \quad -(M^{-1}N)_{22}K_p\right]$$

$$D_T = \left[-(M^{-1}N)_{21}\right]$$

- The State Equations of the system are of the form:

$$x(k+1) = \Phi_T x(k) + \Gamma_T \alpha(k)$$
$$y(k) = H_T x(k) + D_T \alpha(k)$$

- The system is implemented in MATLAB
- The Eigen values of the Φ_T are
  -0.0754070037871251 + 0i ; 0.0353667512660581 + 0.0800193467122896i; 0.0353667512660581 - 0.0800193467122896i; 0.98800911531615 + 0.0461119173915335i; 0.98800911531615 - 0.0461119173915335i;
- The open loop poles from the previous section are:
  0.988009115316149 + 0.0461119173915378i; 0.988009115316149 - 0.0461119173915378i; 0.035366751266058 + 0.0800193467122896i; 0.035366751266058 - 0.0800193467122896i; -0.075407003787125 + 0i;
- The eigen values of Φ match the open loop poles of the system.

- The Transfer Function using the **ss2tf** command in MATLAB is:

```
                    0.008598 z^2 + 0.0172 z + 0.008598
  ----------------------------------------------------------------------
  z^5 - 1.971 z^4 + 0.9714 z^3 + 0.0005646 z^2 + 0.001129 z + 0.0005646
```

- The Transfer Function obtained from Assignment 2 is as follows:

```
                    0.008723 z^2 + 0.01745 z + 0.008723
  ------------------------------------------------------------------------
  1.015 z^5 - 2 z^4 + 0.9855 z^3 + 0.0005728 z^2 + 0.001146 z + 0.0005728
```

- It can be seen that the two Transfer functions are similar with the exception that the original transfer function is not normalized (coefficient of highest power z is not 1).
- The Controllability of the system is checked using the following code:
- The Controllability matrix of the system is

```
  -91.2330  -88.7137  -86.2640  -83.8305  -81.3111
        0         0         0    -0.0168   -0.0499
   46.6806   92.0721   89.5297   87.0311   84.4969
        0    46.6806   92.0721   89.5297   87.0311
        0         0    46.6806   92.0721   89.5297
```

- The rank of the controllability matrix of the system is 5 (full rank).
- The system is completely controllable

5.3.2 Simulink model of state feedback:

- A state feedback model is created in Simulink is created as shown below.
- The feedback is given by multiplying through a gain matrix K which is found by:
  - The controllable canonical form of the system and the controllability matrix are created (the coefficients of the characteristic equations are in one row).
  - The desired closed loop poles are first obtained.
  - The desired characteristic equation is formed and the coefficients of the polynomial are put in vector form.

- The corresponding coefficients from the open loop controllability matrices are also put in vector form.
- The corresponding desired coefficient vector terms are subtracted in order to get the the gain matrix K of the state feedback system.

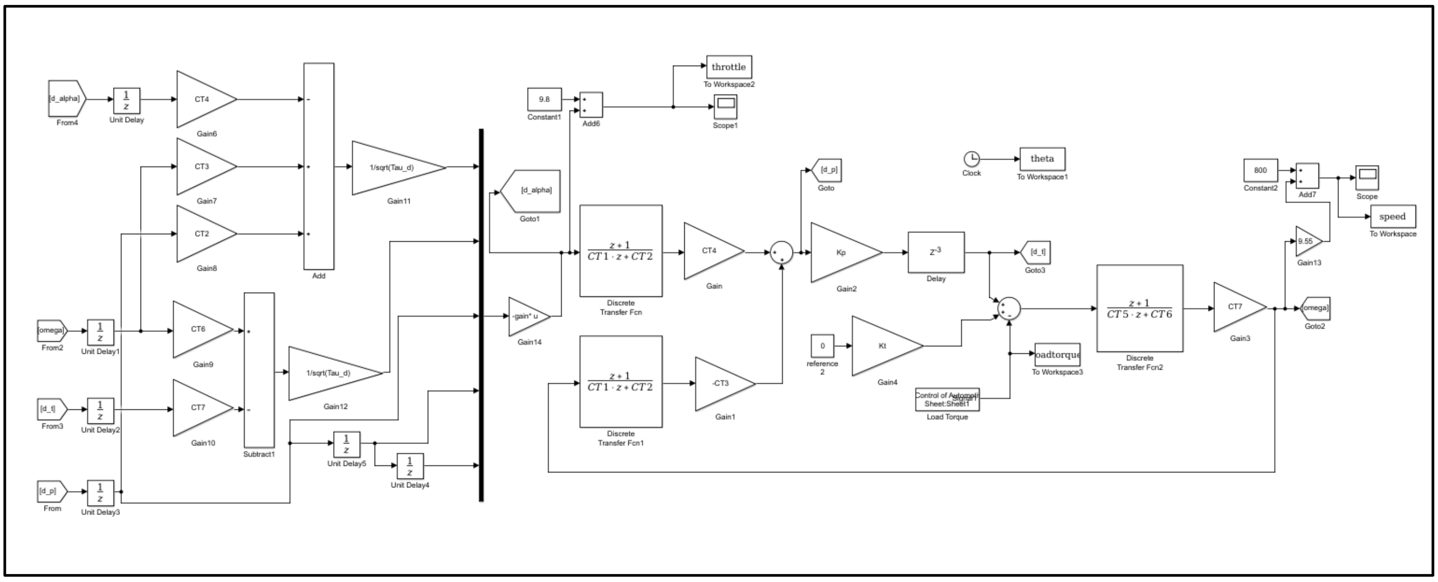

Figure 46 - State Feedback in Simulink

5.3.3 SISO control using pole placement:

- The desired eigen values for the system are:
  desp1 = -0.0754060550970808 + 0i;
  desp2 = 0.0353677601866212 + 0.0800164530756419i;
  desp3 = 0.0353677601866212 - 0.0800164530756419i;
  desp4 = 0.872475528559892 + 0.119136960397931i;
  desp5 = 0.872475528559892 - 0.119136960397931i;
- The gain matrix obtained from the above algorithm and desired eigen values is:
  Gain K = [-0.00237245997204207 -0.835597730470347 0.000313140739270608 0.000308679022174057 0.000303633292454355];
- The response of the linearized discrete model are:

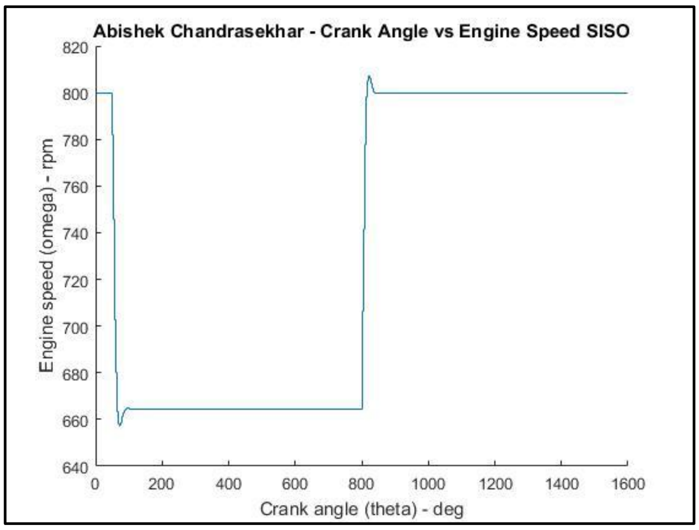

Figure 47 - Engine speed using SISO pole placement

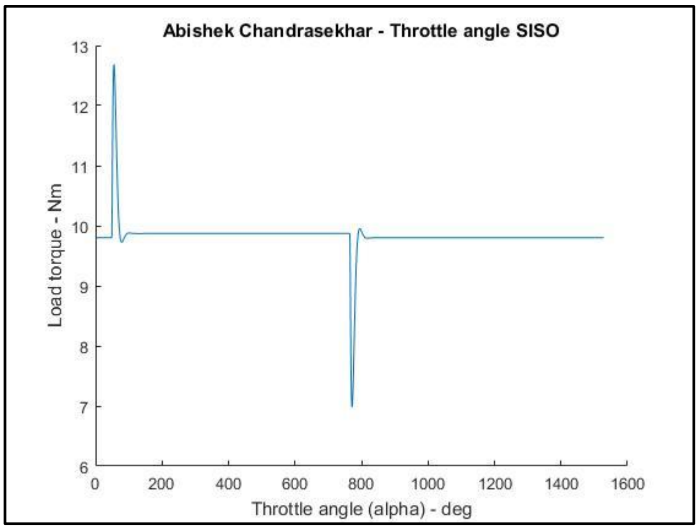

Figure 48 - Throttle response using SISO pole placement

5.3.4 SISO control using pole placement with Integrator:

- An integrator is added to the model that was created for Problem 2.
- The purpose of this integrator is to reduce the steady state error.
- The value of integrator gain K_i is taken from the previous assignment where PID control was implemented. ($K_i = 0.028$).
- The input to the integrator is the change in the output (omega).

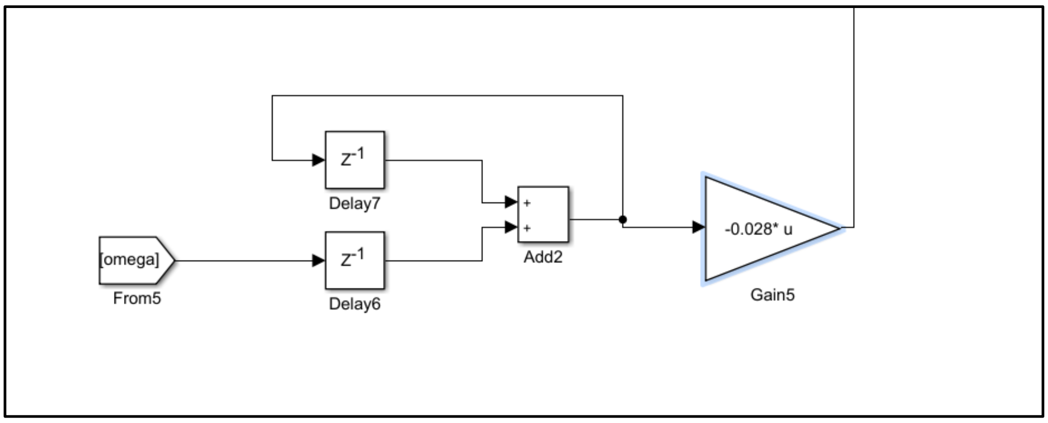

Figure 49 – Integrator for SISO pole placement

- The engine speed and the throttle angle for the SISO controlled system with an integrator:

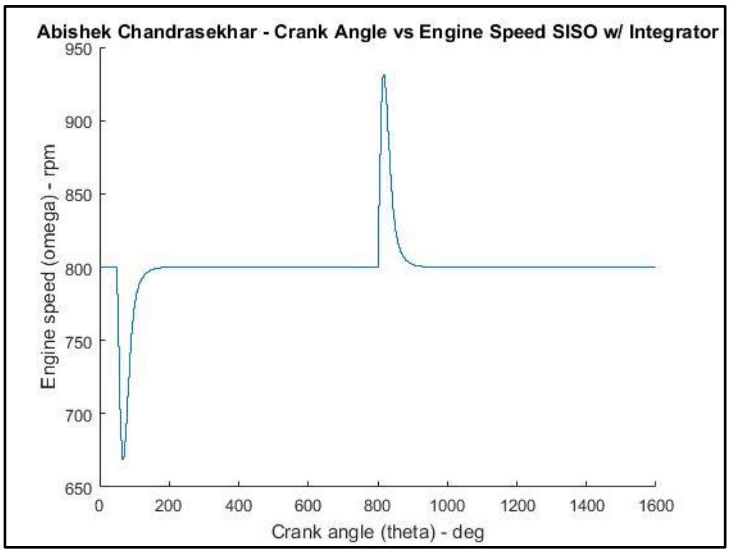

Figure 51 - Speed with SISO Control w/ Integrator

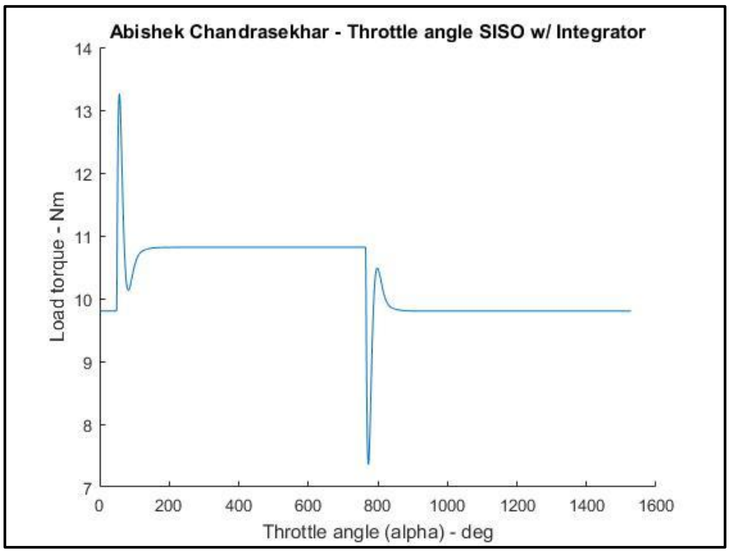

Figure 50 - Throttle angle with SISO control w/ Integrator

- From the above plots, the addition of the integrator eliminates the steady state error of the engine speed. It settles at 800 rpm. The settling time is 230 T theta.
- The throttle angle lies between 7 and 14 deg approx. and the engine speed does not fall below 500 rpm (does not stall) .

5.4 SISO – LQR

The SISO control is implemented again, but this time using a linear quadratic regulator.

Procedure:

- The Linear Quadratic Regulator is used to determine the state feedback gain matrix K.
- The LQR command in MATLAB finds the gain using the Phi, Gamma, Q and R matrices.
- If phi and gamma matrices are controllable, **then Q has to be positive semi definite.**
- Matrix **R has to be positive definite**.
- The Q and R matrices need to be defined in order to obtain an optimized gain.
- The elements of the matrix Q add weight to the respective states.
- The elements of the matrix R are weights to the inputs of the system. In this case, 1 input.
- In order to find the structure of the Q matrix, the following calculation is done:

  Q = H'*H;
- Once the elements of Q are calculated, a proper structure for altering Q is obtained
- For this particular system, it can be seen that Q(5,5) term is in the order of 10^8.
- After testing various Q matrices, the matrix below seems to give the best response.

  Q =[0 0 0 0 0; 0 1 0 0 0; 0 0 0 0 0; 0 0 0 0 0; 0 0 0 0 4e-08];
- The R matrix is a scalar quantity since there is only one input for the system.
- The weight given to the input of the system is 1. This is done in order to limit the throttle angle (input) from exceeding the allowable limits (6 deg to 20 deg).

  R = 1
- The gain matrix obtained by using the dlqr command in MATLAB is:

  K = [-0.00235356728354823 -0.815869415235006 0.000305934970387557 0.000301509008096398 0.000296505014277116]
- The closed loop poles of the system are:

  DP = [-0.0754088351057487+0i; 0.0353676254875392+0.0800222614145978i; 0.0353676254875392-0.0800222614145978i; 0.87350705621816+ 0.117446136833947i 0.873507056218163 - 0.117446136833947i]

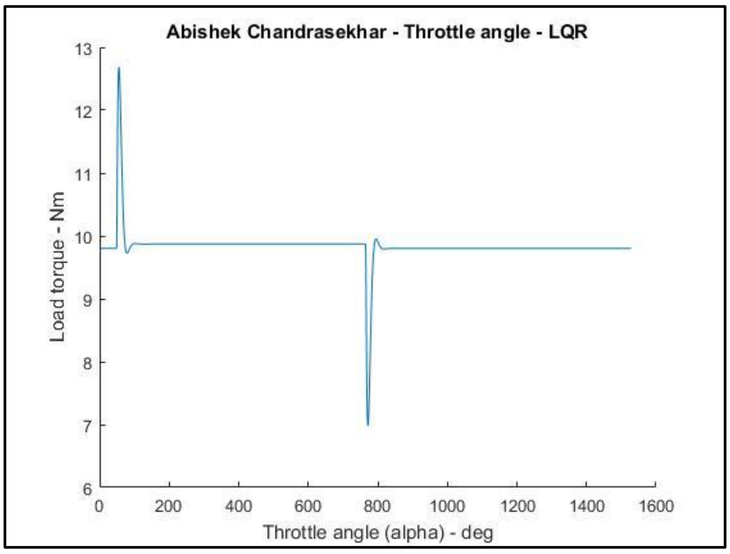

Figure 53 - Throttle angle using LQR for SISO

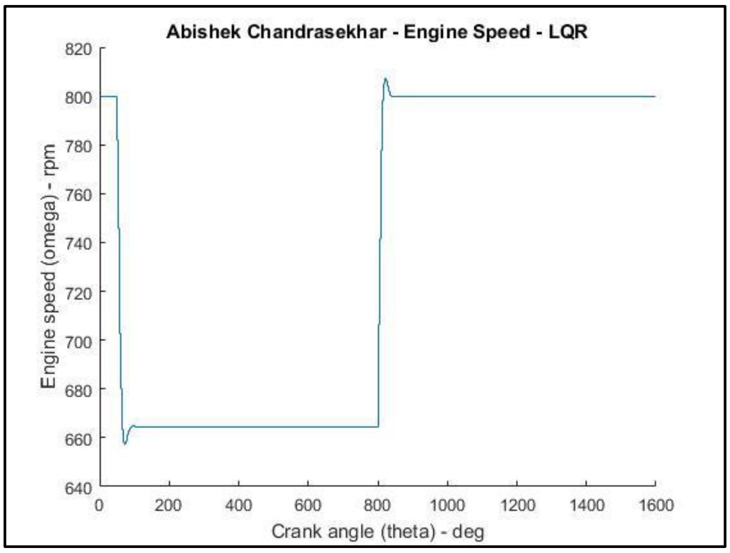

Figure 52 - Engine speed using LQR for SISO

- We can see that the **Q matrix is positive semidefinite and the R matrix is positive definite.**
- From the plots of the engine speed and the throttle angle given above, it can be seen that they are **within the allowable limits.**
- The throttle angle stays between 7 and 12 deg approx..
- The engine speed settles at 120 T theta approx. with a steady state error of 140 rpm.
- The engine speed does not drop below 500 rpm, hence does not stall.
- **Thus, the SISO control using the Linear Quadratic Regulator is valid.**

5.5 SISO - LQR - Augmented System:

- To the model created above, an integrator is added in order to reduce the steady state error of the system
- As in problem 3, The value of integrator gain K_i is taken from the previous assignment where PID control was implemented. ($K_i = 0.028$).
- The input to the integrator is the change in the output (omega).

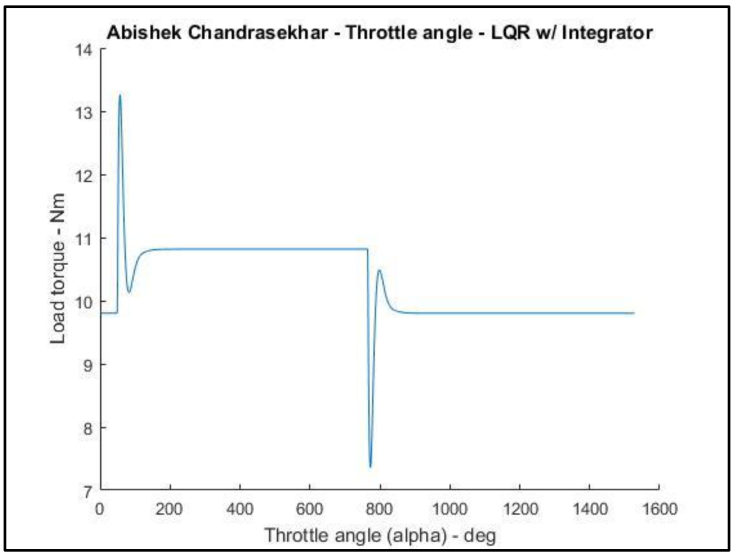

Figure 54 - Throttle angle using LQR w/ Integrator

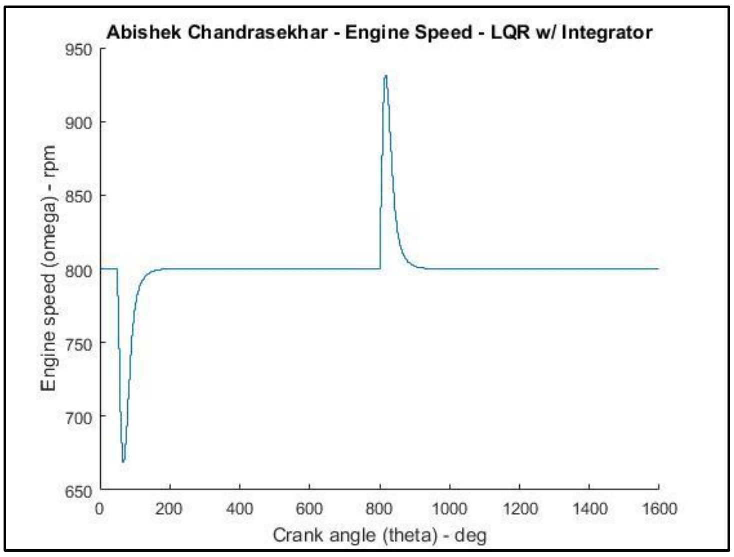

Figure 55 - Engine speed using LQR w/ Integrator

- As in problem 3, the addition of the **integrator eliminates the steady state error** of the engine speed. It settles at 800 rpm. Also, the engine does not stall (does not fall below 500).
- The engine speed settles at around 220 T theta.
- The throttle angle lies between 7 and 13.2 deg approx**.**

The SISO control implementation gives a good response for engine speed. The same procedure is now carried forward for MIMO control and will be implemented in the simulator

6. MIMO control:

The MIMO control for the engine utilizes both the throttle angle and the spark advance angle as inputs for the system.

6.1 State space equations for MIMO control:

- The MIMO model is built in Simulink and MATLAB where the two inputs are the throttle angle and the spark advance angle.
- The state matrices for the MIMO system are given below:

$$M^{-1} = \begin{bmatrix} (M^{-1})_{11} & (M^{-1})_{12} \\ (M^{-1})_{21} & (M^{-1})_{22} \end{bmatrix} \qquad OM^{-1} = \begin{bmatrix} (OM^{-1})_{11} & (OM^{-1})_{12} \\ (OM^{-1})_{21} & (OM^{-1})_{22} \end{bmatrix}$$

$$M^{-1}N = \begin{bmatrix} (M^{-1}N)_{11} & (M^{-1}N)_{12} \\ (M^{-1}N)_{21} & (M^{-1}N)_{22} \end{bmatrix} \qquad Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix}$$

$$\Phi_T = \begin{bmatrix} (OM^{-1})_{11} & (OM^{-1})_{12} & 0 & 0 & Q_{12}K_\tau \\ (OM^{-1})_{21} & (OM^{-1})_{22} & 0 & 0 & Q_{22}K_\tau \\ (M^{-1})_{11} & (M^{-1})_{12} & 0 & 0 & -(M^{-1}N)_{12}K_\tau \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad \Gamma_T = \begin{bmatrix} Q_{11} & Q_{12}K_\delta \\ Q_{21} & Q_{22}K_\delta \\ -(M^{-1}N)_{11} & -(M^{-1}N)_{12}K_\delta \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$H_T = [(M^{-1})_{21} \quad (M^{-1})_{22} \quad 0 \quad 0 \quad -(M^{-1}N)_{22}K_\tau]$$

$$D_T = [-(M^{-1}N)_{21} \qquad -(M^{-1}N)_{22}K_\delta]$$

- The state matrices are given by:

$$x(k+1) = \Phi_T x(k) + \Gamma_T \alpha(k)$$
$$y(k) = H_T x(k) + D_T \alpha(k)$$

- The Simulink implementation of the system is shown below

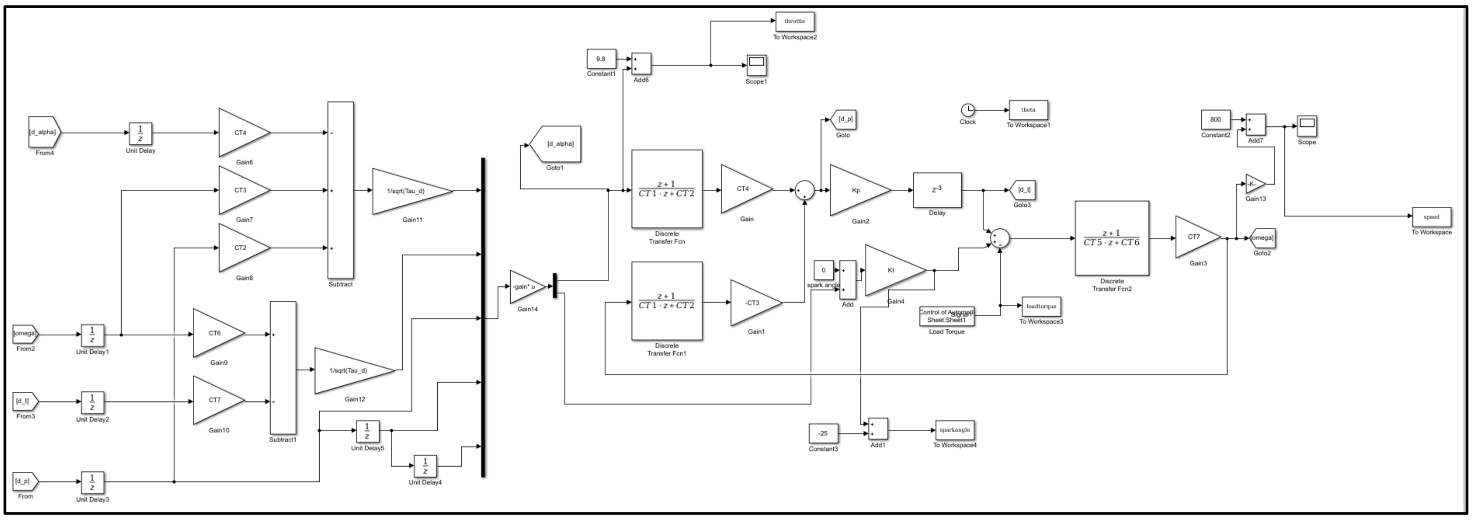

Figure 56 - Simulink block for MIMO control

6.2 MIMO Pole Placement:

- A Demultiplexer is used to split the gain feedback and send it to the throttle angle and the spark angle.
- The desired closed loop poles for this MIMO system are dp=[-0.0754060550970808+0i 0.0353677601866212+0.0800164530756419i 0.0353677601866212-0.0800164530756419i .872475528559892+0.119136960397931i 0.872475528559892-0.119136960397931i];
- As with the SISO case, the place command is used to find the gain matrix K
- The gain matrix obtained is:
  K = [-0.0272850613461539 0.0342808664069572 -0.0156738219245573 -0.00599721198882779 -0.00194109034549855;
  -26.1590516855793 -19.7230617692094 -18.8399250360446 -3.75762367166491 -1.15276732288923]
- The throttle angle, engine speed and spark angle are plotted with respect to the crank angle for this MIMO system.
- As can be seen from below, this **feedback leads to an unstable system.**
- The MATLAB place command does not return an ideal gain for the desired poles.

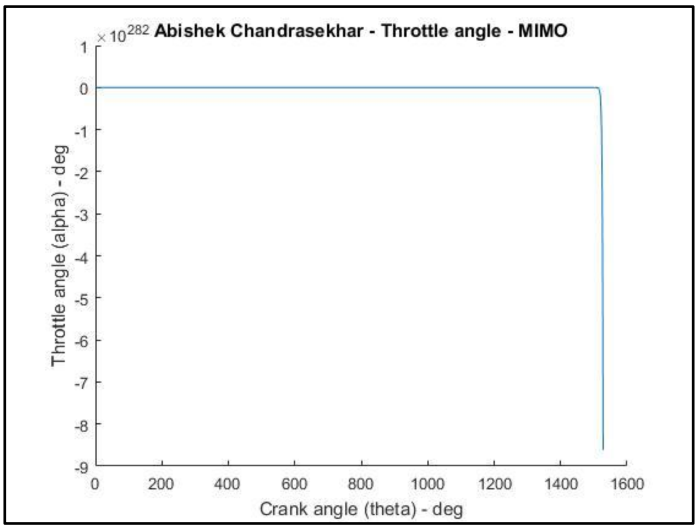

Figure 59 - MIMO throttle angle

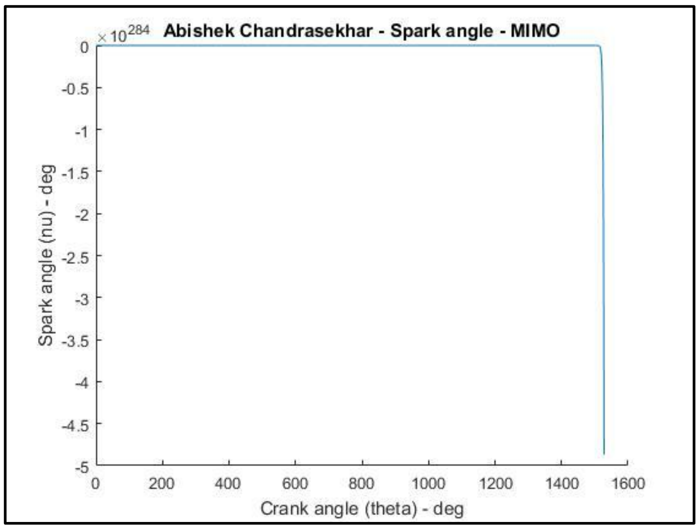

Figure 58 - MIMO spark advance angle

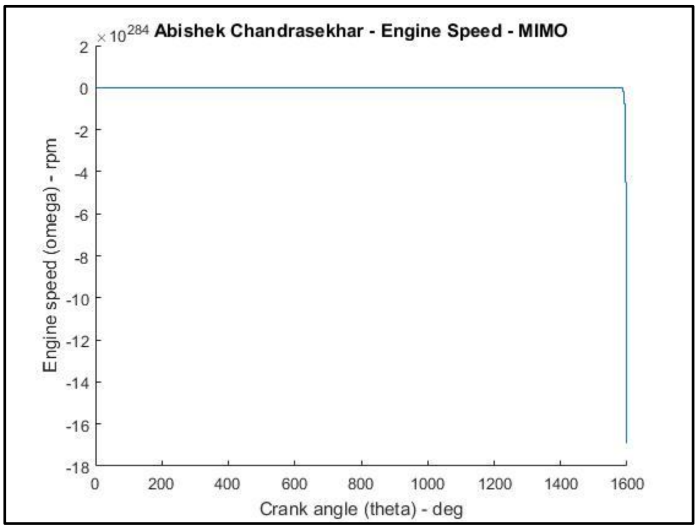

Figure 57 - Engine speed - MIMO

6.3 MIMO - Pole Placement - Augmented System:

- An integrator effect is added to the MIMO system created above to try and minimize the steady state.

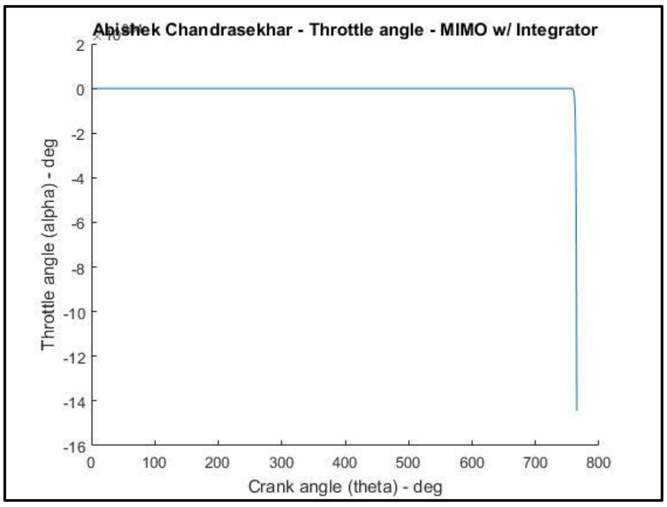

Figure 62 - Throttle angle - MIMO w/ Integrator

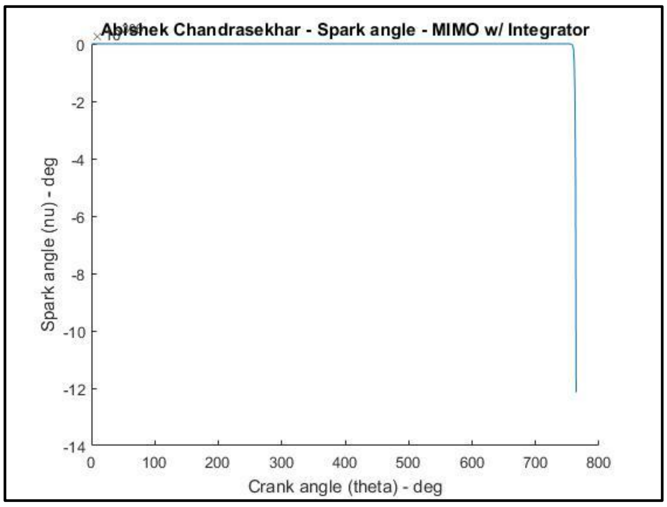

Figure 61 - Spark advance angle - MIMO w/ Integrator

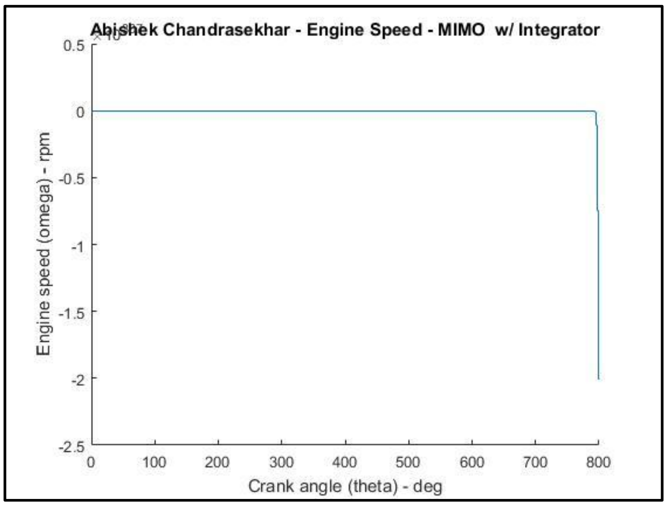

Figure 60 - Engine speed - MIMO w/ Integrator

- But as with the previous case, the gain matrix returned by MATLAB is not ideal and results in an unstable system. (The system blows up).
- MATLAB version used: 2017a;
- For this MATLAB version, the place command doesn’t seem to work for the MIMO case

6.4 MIMO – LQR:

- As in the SISO case, the Linear Quadratic Regulator is used to find an ideal gain that can be used to input

  Q =[0 0 0 0 0; 0 1 0 0 0; 0 0 0 0 0; 0 0 0 0 0; 0 0 0 0 1e-08];
- Since phi and gamma matrices are controllable, then Q has to be positive semi definite.
- Matrix **R has to be positive definite**.
- The Q and R matrices need to be defined in order to obtain an optimized gain.

- The elements of the matrix Q add weight to the respective states. Increasing it beyond a point leads to a less than optimum response

  R= [1 0 ; 0 1.8]
- The gain matrix obtained by using the DLQR command in MATLAB is:

  K = [-0.00224404573386889 -0.755098187834137 0.000284637301546551 0.000279643821350812 0.000274493110794325;
  -0.000256289482246264 -0.256936471419289 7.44007466532782e-05 8.36247046732109e-05 9.28446372649963e-05]
- The closed loop poles of the system are:
  dp=[-0.0754060550970808+0i 0.0353677601866212+0.0800164530756419i 0.0353677601866212-0.0800164530756419i 0.872475528559892+0.119136960397931i 0.872475528559892-0.119136960397931i];

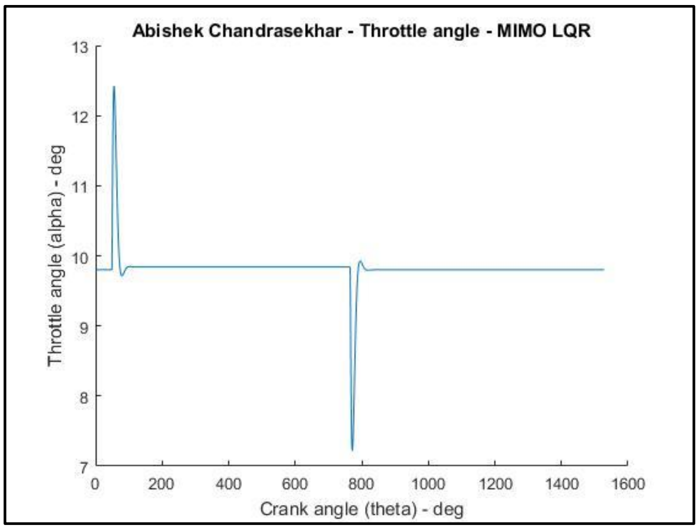

Figure 64 -Throttle angle MIMO LQR

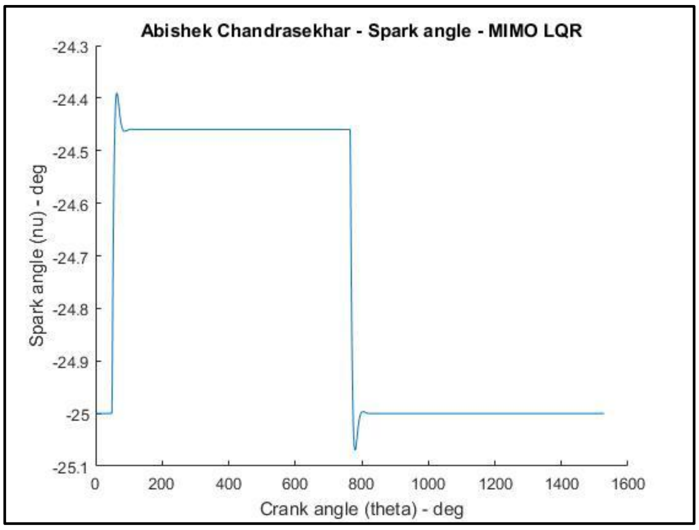

Figure 63 - Spark angle MIMO LQR

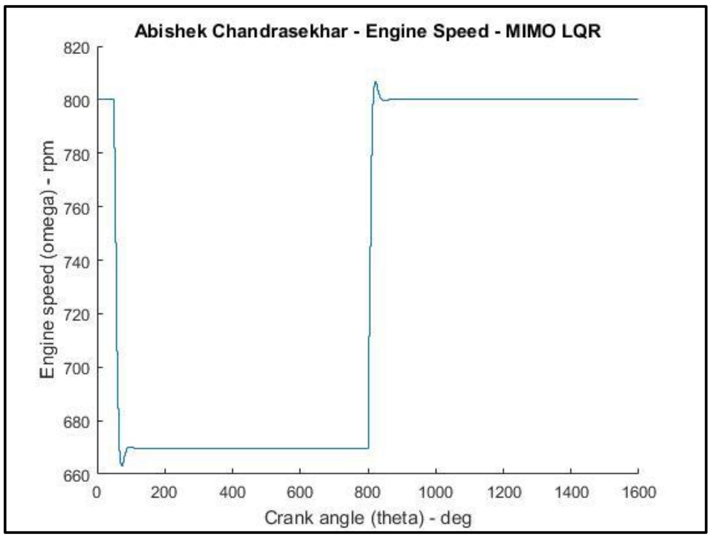

Figure 65 - Engine speed MIMO LQR

- The throttle angle (between 6 and 20 deg) , spark angle (between -40 and -5 deg)
- The engine speed does not drop below 500, (does not stall).

- The engine speed settles at 130 T theta (approx.) with a steady state error of 130.
- Using the MIMO control, it is easier to balance the performance and control effort of the two inputs.
- Less control effort is required by the throttle angle in the MIMO case.

6.5 MIMO – LQR Augmented system:

- To the model created above, an integrator is added in order to reduce the steady state error of the system
- As in problem 3, The value of integrator gain K_i is taken from the previous assignment where PID control was implemented. Ki = 0.028 was chosen as the base, and various values were tested, until optimum response was achieved. (K_i = 0.04).

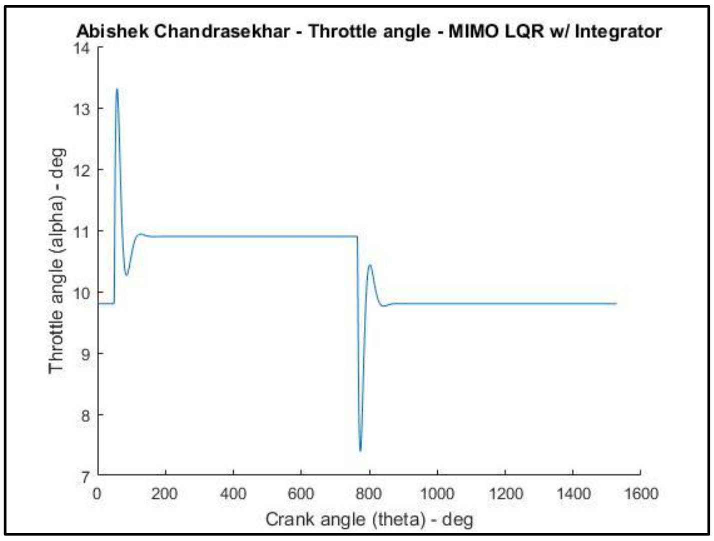

Figure 66 -Throttle angle MIMO LQR w/ Integrator

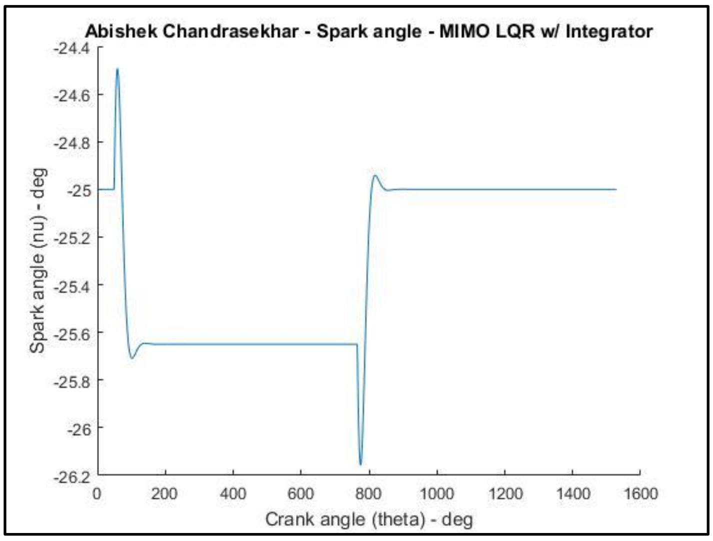

Figure 67 - Spark angle MIMO LQR w/ Integrator

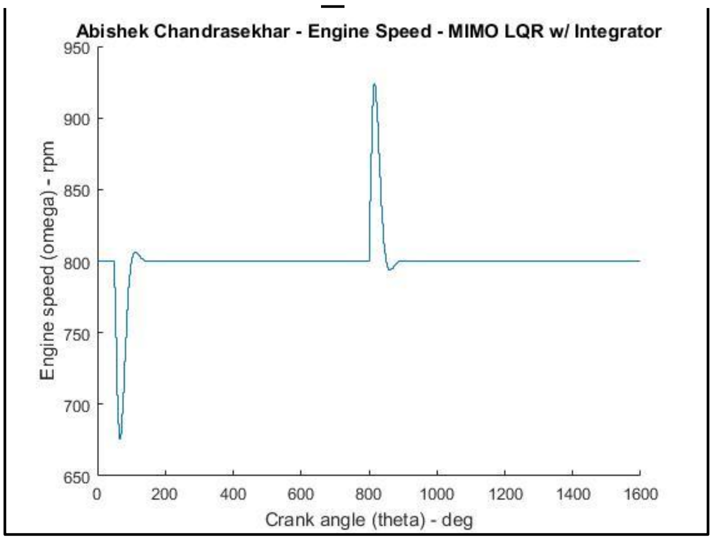

Figure 68 - Engine speed MIMO LQR w/ Integrator

- The integrator eliminates the steady state error of engine speed. It settles at 800 rpm.
- The throttle angle lies between 7 and 13 deg approx.
- The spark angle between -26 and -22
- The engine speed settles at 800 rpm at around 200 T theta, which is sooner than the SISO case and does not stall (does not fall below 500 rpm).

6.6 Implementation of MIMO -LQR with Integral control in Simulator:

- The MIMO controller that was developed above is implemented in the continuous model.
- The Q and R matrices from the discrete case are carried over to this model.
- The implementation of the MIMO control inside the triggered subsystem is shown

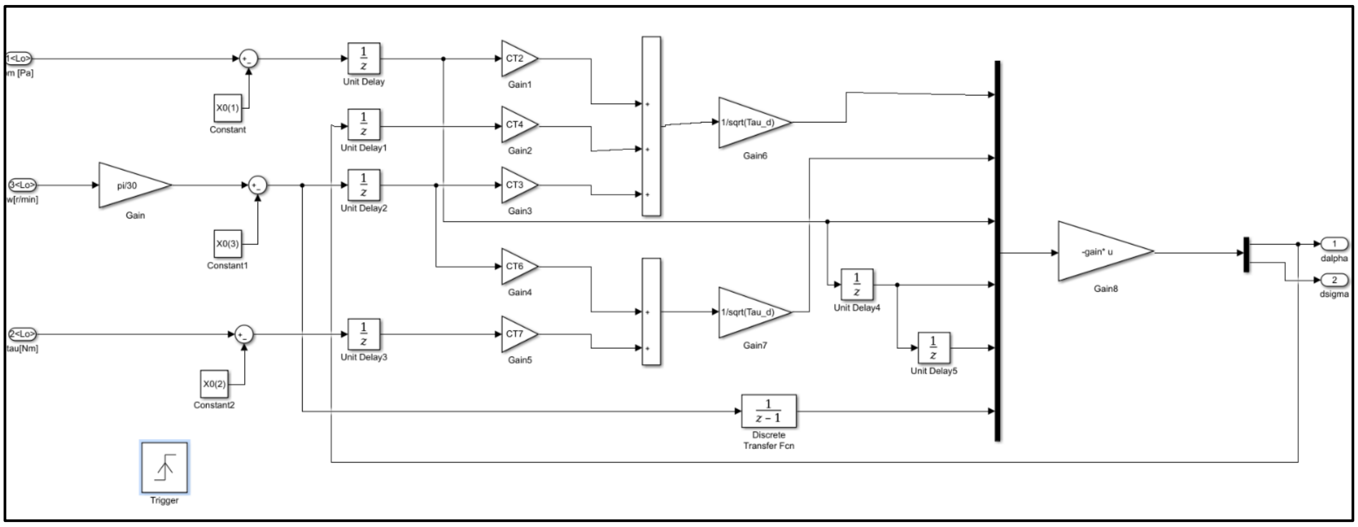

Figure 69 - MIMO control inside the Triggered subsystem

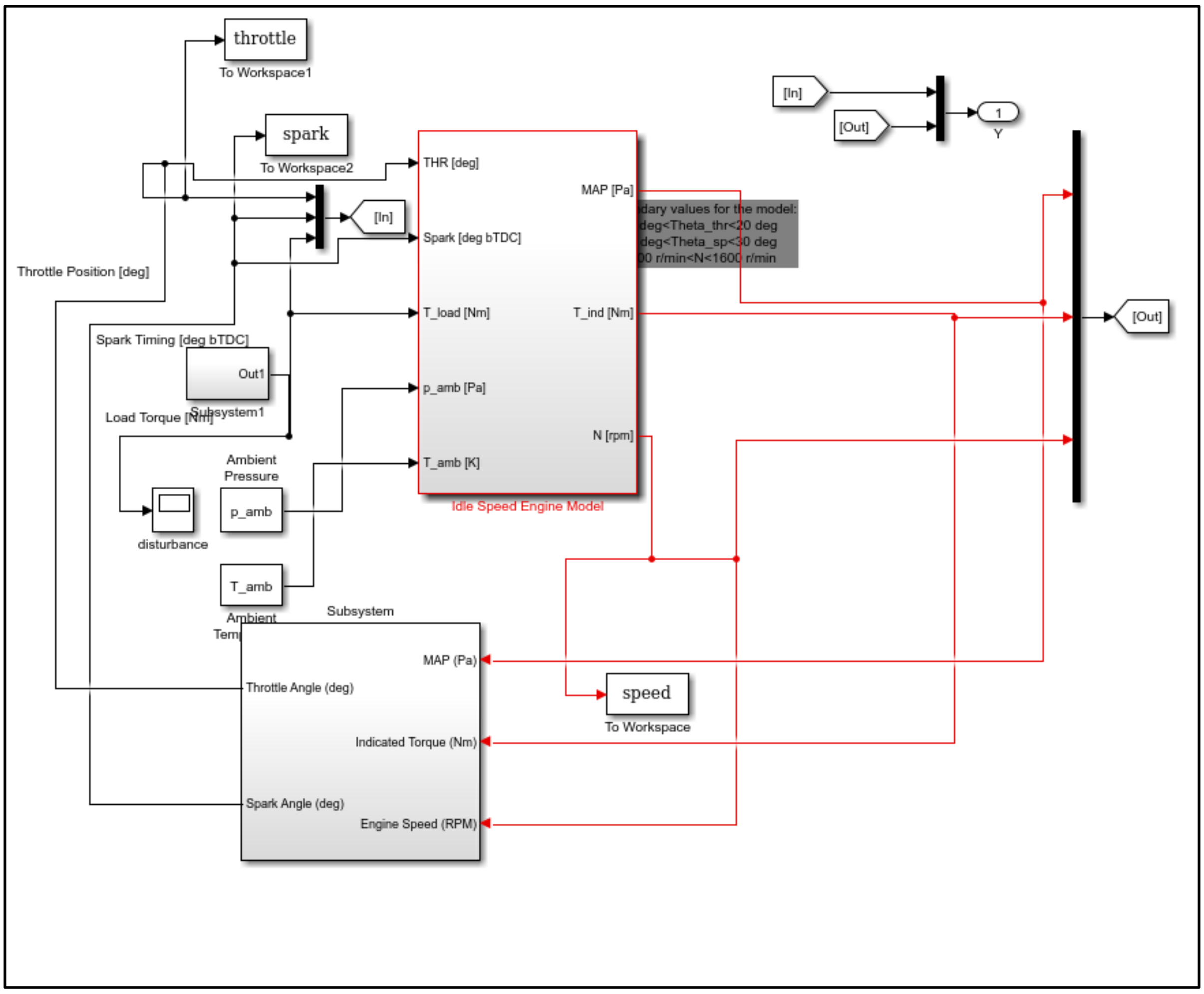

Figure 70 - Engine simulator with MIMO controller

- The response of the simulator once the MIMO LQR is added is given as follows:

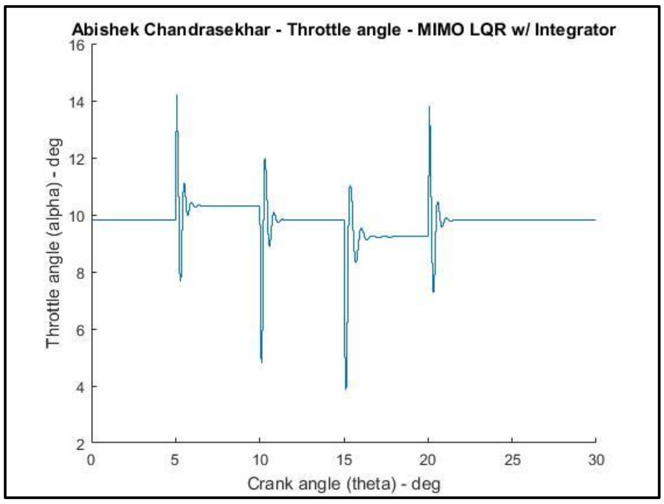

Figure 73 - Throttle angle using MIMO LQR in Simulator

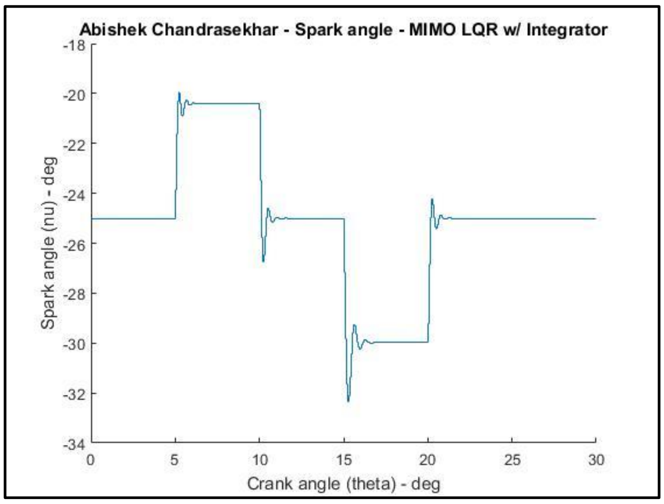

Figure 72 - Spark advance angle using MIMO LQR in Simulator

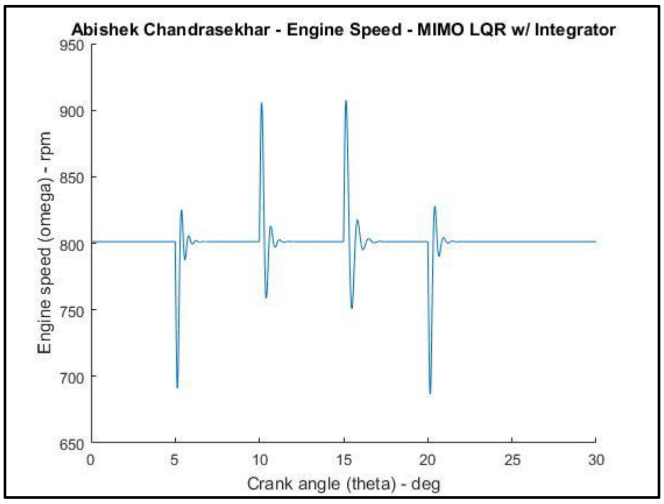

Figure 71 - Engine speed using MIMO LQR in Simulator

- From the above plots, it can be seen that though the engine speed stabilizes at 800 rpm with excursions, on the application of external load, **the throttle angle goes below 6 degrees**, which is below the allowable limit.
- Changes in Q and R matrices were attempted, but the MATLAB code **didn't solve** for the gain matrix for most values of Q and R.
- There were several errors that stated that the solver couldn't converge and a particular state was going to infinity.

6.7 Implementation of MIMO pole placement:

- With the pole placement method as well, the MATLAB code returned an invalid gain that could not be solved.
- The following poles were attempted to be placed:
- Desired poles =[0.0754060550970808+0i 0.0353677601866212+0.0800164530756419i 0.0353677601866212-0.0800164530756419i 0.872475528559892+0.119136960397931i 0.872475528559892-0.119136960397931i];
- The gain matrix that MATLAB returned using the place command is as follows:
- K = [-0.0272850613461539 0.0342808664069572 -0.0156738219245573 -0.00599721198882779 -0.00194109034549855 0.0400000000000000; -26.1590516855793 -19.7230617692094 -18.8399250360446 -3.75762367166491 -1.15276732288923 0.0400000000000000]

- For this gain matrix, the MATLAB code doesn’t solve and returns an error.

7. Conclusion:

The simulator model was built and different control methods were implemented. The open loop dynamics of the model was used to get data that is used to study the model and improve the control for the engine. Both classical and modern control techniques were implemented. The idle speed of the engine was attempted to be maintained using these techniques.

To implement control, a linearized discrete model of the engine was created. The time domain is converted to crank shaft domain. The sampling time for the linearized discrete model is taken as pi/3. Once the discrete model of the engine is created, it becomes easier to implement control such as state feedback.

SISO PID and Root Locus:

Initially, Proportional Integral Derivative control was implemented. The gains for the PID were obtained using the frequency tuning method. The final tuned PID controller resulted in a good response with a low settling time and zero steady state error. After PID control was implemented, another controller was designed using root locus technique. The pole zero map of the system was used to find the unstable poles and using the rltool in MATLAB, the compensator for the system was designed.

The PID control and the controller designed using root locus techniques were modeled and tuned in the linear discrete model of the engine. They were later implemented in the continuous engine simulator. Comparing the results of the discrete model to the continuous model, the continuous model has more excursions, i.e., both the controllers were more suited to the linear discretized model.

SISO State feedback:

To implement control using state feedback, the state space equations are created in MATLAB. The corresponding Simulink model is also built. For SISO control, the throttle angle is the input and the engine speed is the output. The state feedback gain in this case was a single row vector. The gains for feedback were calculated using the pole placement algorithm from Linear systems theory. The desired poles are placed inside the new system. The throttle angle and engine speed responses of the engine were stabilized But the SISO control resulted in a steady state error of around 150 rpm. An integrator is added to the system to remove the steady state error.

The same procedure is repeated for SISO control, but instead of calculating the gain matrix using either the algorithm or the pole placement method, the gain is calculated by MATLAB using the Linear Quadratic Regulator (LQR). The Q and R weighting matrices for the LQR were tuned and the response of the engine was far better that pole placement. The steady state error that resulted in this controller was eliminated as in the previous case with the addition of an integrator.

The above two SISO state feedback control methods were implemented only in the discrete model of the engine. They were not carried over to the continuous model. But for the discrete engine model, the LQR technique was the most efficient in controlling idle speed of the engine. The poles obtained from LQR were then used in pole placement to improve the response in the first method as well.

MIMO State feedback:

For MIMO control, the throttle angle and the spark advance angle were taken as the inputs to the systems. As with the SISO case, the state space equations are given in MATLAB and Simulink. But in this case, the gain matrix contained two row vectors. The gains were again calculated using LQR technique. An integrator was added to the state feedback so as to remove the steady state error. But while, using the pole placement technique, the gain matrix obtained resulted in a system that blew up. These two methods were implemented in the discrete model. The control effort for the throttle is reduced for this discrete model, as it allows control of both the throttle angle and the spark advance angle.

When they were carried over to the continuous model, the LQR feedback design resulted in a reasonable response for engine speed. But the throttle angle of the angle falls below the allowable limit (below 6 degrees). The Q and R matrices were tuned, but for this MIMO system, MATLAB couldn’t solve for most of the weighting matrices that were implemented in the discrete model. The pole placement command does not run in MATLAB version 2017a. It returned an error stating that the solver failed to converge, so the response plots were not obtained.

For no case, except the MIMO pole placement part, the engine speeds are within the limits. The engine does not stall. For all SISO controllers, the throttle angle is within the allowable limit. For the MIMO LQR part, the throttle angle goes out of the allowable limit.

Comparing the different techniques that were implemented for the simulator, the PID control returned a response that was fairly stable. The MIMO LQR control returned the best possible response, but the throttle angle was not within the limit. It can be concluded that the PID controller performs best for the Continuous engine simulator and the MIMO LQR controller performs best for the Discrete model.

8. Open Ended Design (Extra Credit):

To further improve the performance of the Idle speed controller, several attempts were made using the following techniques:

- Model Predictive control
- H infintity controller
- Feedforward control using lookup table

After attempting several methods, the final method chosen is described below:

MIMO LQR + Feedforward control using Lookup tables:

Although the idle speed controller is used to maintain idle speed of 800 rpm, using MIMO control, the throttle angle was beyond the allowable limits. So the following implementation improves the MIMO control using LQR

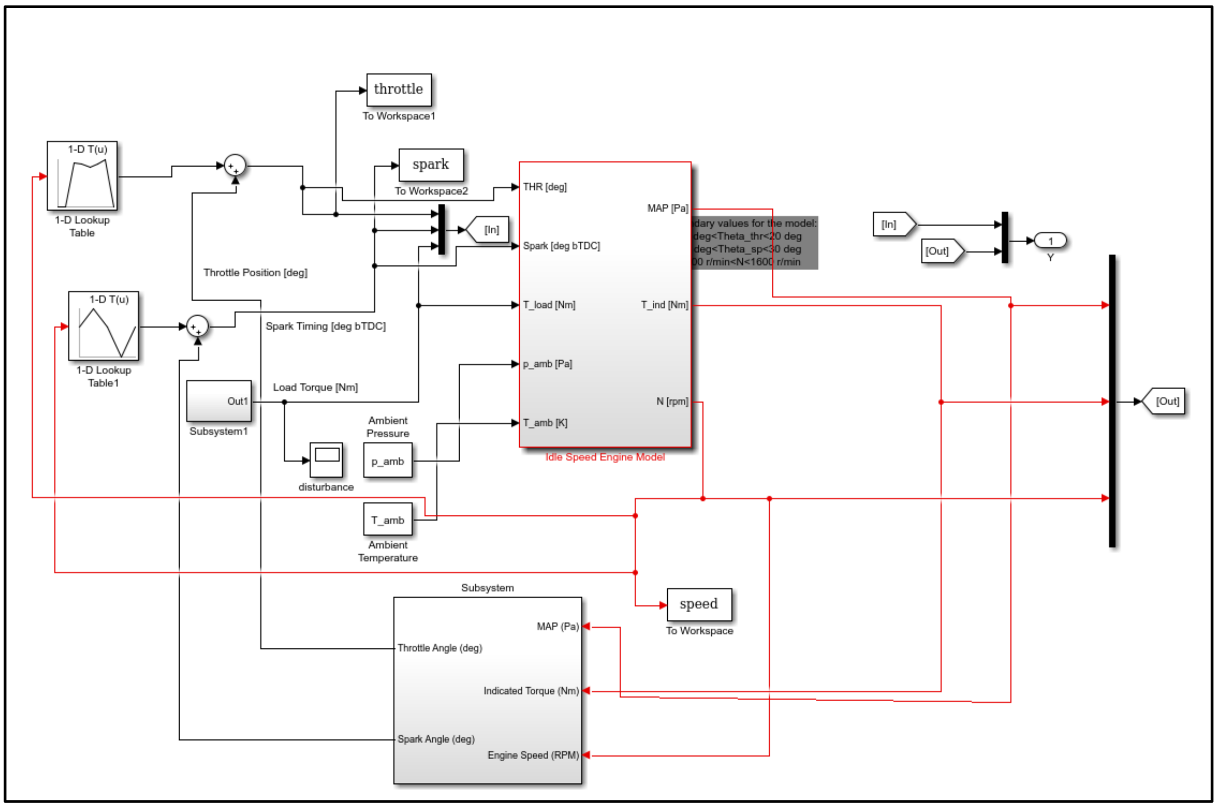

Figure 74 - MIMO control + feedforward using Lookup table

In addition to the idle speed controller, look up tables are added to the two inputs: throttle angle and the spark angle. The values of the throttle angle and the spark advance angle which compensate for the excursions are determined and are entered in the look up table.

The anticipatory signals are given based on the assumption that the time when the load acts on the engine is known. The responses of the engine with this new control are given below:

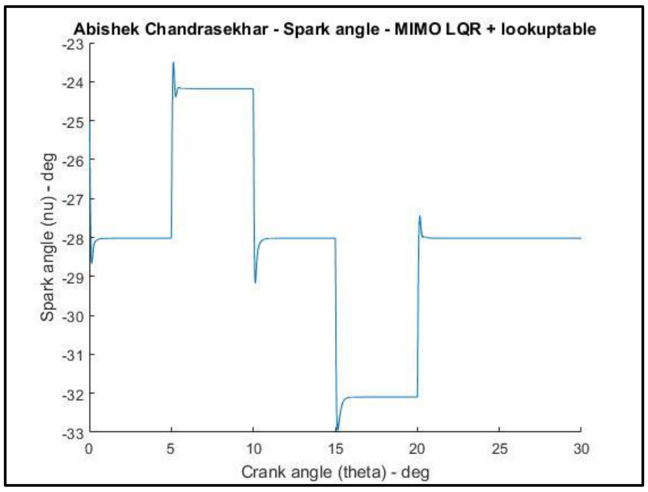

Figure 77 - Spark advance angle for implemented control

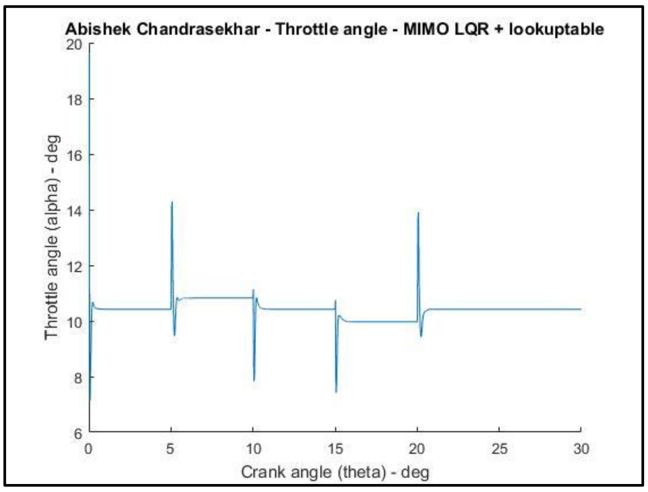

Figure 76 - throttle angle for implemented control

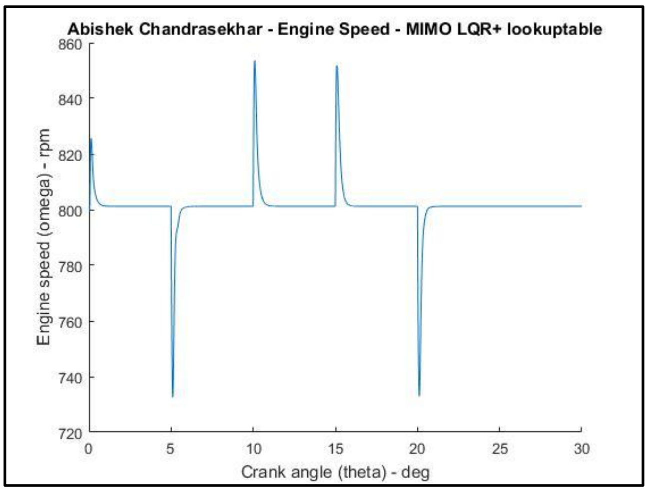

Figure 75 - Engine speed for implemented control

Inference:
From the above plots, it is clear that the new control improves the MIMO control using LQR. The throttle angle is between the allowable limits in this case, and the engine speed is controlled with excursions of only 70 rpm.

It should be noted that, although in Simulink, the control is successfully implemented, the throttle changes are too sharp for a real world implementation.

MATLAB version : 2017a

9. Reference:

[1] Digital Control of Automotive Powertrain Lecture notes by Dr. Stephen Yurkovich
[2] Y. Yildiz, A. Annaswamy, D. Yanakiev and I. Kolmanovsky, "Adaptive Idle Speed Control for Internal Combustion Engines," 2007 American Control Conference, New York, NY, 2007, pp. 3700-3705.
[3] Andrea Balluchi, Luca Benvenutix, Maria Domenica Di Benedettoz, Giovanni Girasole , Alberto L. Sangiovanni Vincentelli "IDLE SPEED CONTROL DESIGN AND VERIFICATION FOR AN AUTOMOTIVE ENGINE"