Development of a Diesel Engine Fuel Quantity and Intake Manifold Pressure Controller – Term Project 1

Justin Hart
jh74789
ME 397 Graduate Student

Abstract

The objective of this report is to design and tune both a fuel injection quantity and intake manifold pressure controller to match the control performance of dSPACE ASM's Soft ECU. Results are compared using the vehicle speed, engine torque, and lambda signals. The scenario to be simulated that will be used to test the controller performance is a vehicle acceleration – deceleration phase. This simulates a driver entering a highway by onramp and accelerating past their desired cruising speed, then releasing the accelerator pedal to reach the desired cruising speed once on the highway. After the controllers are designed and tuned (using a PID design), the implementation within ASM is described and the results are presented and explained.

1 Introduction and Motivation

As the dominant internal combustion engine for medium and heavy-duty vehicles in the United States, diesel engines play an important part in the transportation of people, goods, and services. To manage everything from engine speed to emissions of these engines, controllers are implemented as part of the electronic control unit (ECU). This report will review the design of a fuel injection quantity and an intake manifold pressure controller for a diesel engine as well as their implementation within dSPACE ASM software. These controllers will be compared with ASM's Soft ECU signals and results of the comparison will be presented.

2 System Analysis and Design

The ASM model used for this analysis is a four-cylinder, 2.0 liter, turbocharged diesel engine. In order to evaluate the designed controllers vs. the Soft ECU within ASM, the vehicle speed, engine torque and exhaust lambda ratio simulated values are compared. The controllers are tested over an acceleration and deceleration phase in a highway speed scenario. To avoid the influence of the simulated spikes in data due to gear shifting, a segment of the acceleration phase where no gear shifting occurs is used for data collection and later comparison. The acceleration and deceleration scenario that is simulated is one in which a vehicle is starting from a red light and then entering a highway on-ramp and accelerating to highway speed. The driver accelerates above the desired cruising speed and then once on the highway, releases the accelerator to decelerate to their cruising speed. This is simulated within ASM as described in Section 3.

The first step in the system analysis is to collect the ASM Soft ECU data for the chosen acceleration and deceleration profile. This will be used to compare the designed controllers' performance in Section 4. The data collected in this step includes the exhaust gas lambda, intake manifold pressure, fuel injection quantity, vehicle speed, and engine torque. The baseline data is presented below in Figure 1.

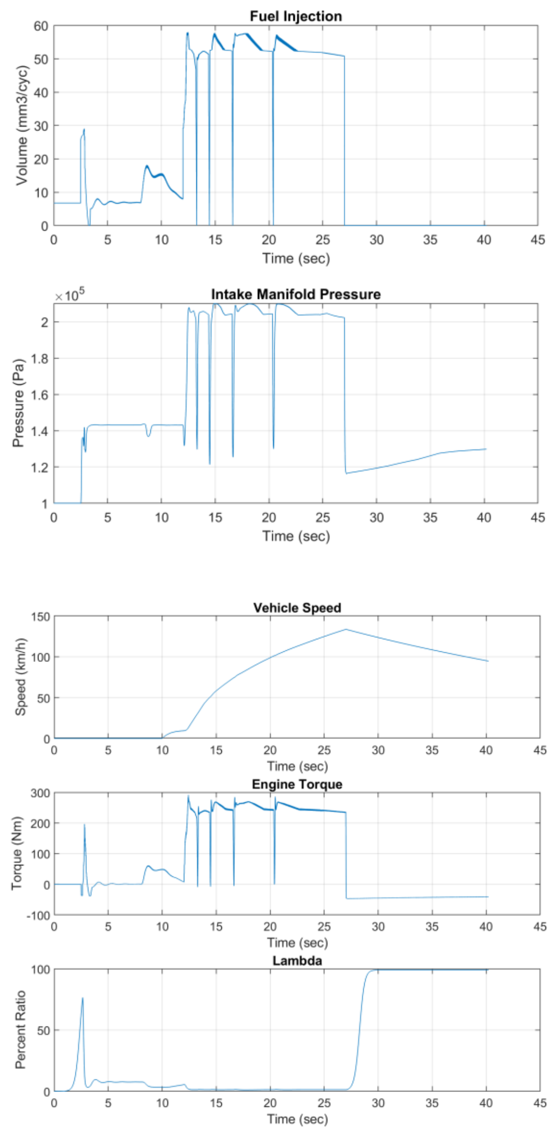

Figure 1: Baseline ASM Soft ECU Data

Looking at the vehicle speed plot, the simulation engine is started and then the vehicle is allowed to idle for a couple of seconds. Then, the acceleration pedal is ramped up to 60% input until the vehicle reaches a speed of about 130 km/h (80 mph). The acceleration pedal input is then dropped to 0% and the vehicle naturally decelerates to a cruising speed of about 100 km/h (65 mph). As you can see in the fuel injection, intake manifold pressure, and engine torque plots, there are large spikes in the data due to gear shifting. For this reason, the future comparison of controller data will over the time period from 20 to 27 seconds of the simulation as this is the longest set of data during acceleration that remains in a constant gear.

Next, the controller design is considered. Both the fuel injection and intake manifold pressure controllers are designed using PID controllers with a feedforward value calculated. For both controllers, the control reference values are the desired and measured engine torques. The engine torque is used as the reference value for both since the engine torque is closely proportional to the engine displacement (which can be boosted through a turbocharger) and is dominated by the fuel injection quantity. The vehicle speed and lambda ratio are related to the engine torque so that if the engine torque is what the controller is tracking, the vehicle speed and lambda ratio will remain close to their desired values as well. The fuel injection controller (1) and intake manifold controller (2) equations are listed below.

$$Fuel\ Inj\ Qty = Kp * e + Ki * \int edt + Kd * \frac{de}{dt} + FF \quad (1)$$

$$Int\ Man\ Pres = Kp * e + Ki * \int edt + Kd * \frac{de}{dt} + FF \quad (2)$$

Where 'Fuel Inj Qty' is the fuel injection quantity controller input, and 'Int Man Pres' is the intake manifold pressure controller input into the ASM model. Kp, Ki, and Kd are the proportional, integral and derivative gains respectively. The e term is the error. In this case, for both controllers, it is the difference between the desired and measured engine torque. The values used for the desired engine torque are the baseline engine torque values recorded from the ASM Soft ECU output. FF is the feedforward 1-D lookup table that is a function of the desired torque. The feedforward values for each controller are listed in Table 1.

Desired Engine Torque (Nm)	**Fuel Injection (mm^3/cyc)**	**Intake Manifold Pressure (Pa)**
234.65	50.7	202329
239.35	51.3	203428
242.67	51.87	204257
243.88	52	203944
246.6	52.17	203698
249.4	52.76	204619
255.2	53.5	207322
261	55	209002
266.8	56.26	209769
269.75	57	209757

Table 1: controller feedforward values as a function of desired engine torque

These feedforward values were gathered from the baseline ASM ECU data by looking at fuel injection and intake manifold values at different engine torques.

Next, the PID controllers are tuned using a trial-and-error method [2]. The tuning is used to achieve a desired response for engine torque, vehicle speed, and lambda ratio only for the period from 20-27 seconds when the simulation remains in a constant gear. First, the Ki and Kd gains are set to zero. The Kp gain is increased until the response is fast enough. Then the Ki gain is increased to reduce the oscillations and decrease the steady state error. Lastly, the Kd gain is increased in order to decrease the rise time and decrease overshoot. This is done slowly as the Kd makes the system sensitive to noise. Each controller was tuned separately using this method, with the other controller turned off while tuning. Then, once adequate gains were set, both controllers were turned on and the system was simulated. This required additional fine tuning of each of the gains in order to achieve an appropriate response of the system with both controllers active. The final gains used for each controller are shown in Table 2.

	Fuel Injection Qty	Intake Manifold Pressure
Kp	0.07	600
Ki	0.009	100
Kd	0.0002	1.5

Table 2: Final PID controller gains used

During the controller tuning process, it was discovered that both controllers were sensitive to integral windup. To counteract this, a clamping method of integrator anti-windup is implemented [1] and it's Simulink design is shown in Section 3. This method first adds a saturation limit on the controller input to the system. While the controller is operating, if the control input to the system is saturated, and the sign of the control input and error are the same, then the integral term of the PID controller is switched to zero. This prevents integral windup from affecting the system response.

3 System Implementation in ASM Description

To implement the system into ASM, a few settings are adjusted to allow for the desired operation. The transmission is set to automatic, the maneuver type is set to offline manual, and the engine EGR valve is turned off (see Figure 2).

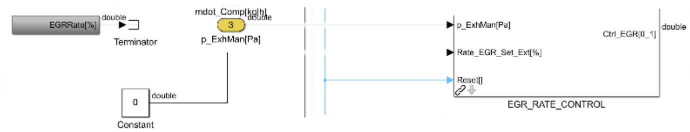

Figure 2: EGR valve input is set to external mode and 0 is input in order to turn off the EGR valve

The fuel injection quantity and intake manifold pressure set points are set to external mode so that the controller inputs are used in the system.

Then, the acceleration/deceleration phase is automated in the offline manual section of the ASM model as shown in Figure 3. The selector lever is first set to park, the accelerator pedal is set to 0% and the brake pedal is set to 100%.

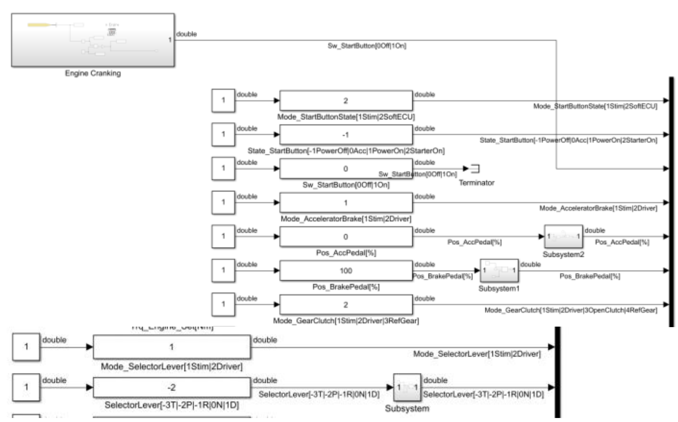

Figure 3: Acceleration/deceleration phase automation

The engine cranking block, and the three subsystems are shown in Figures 4-7.

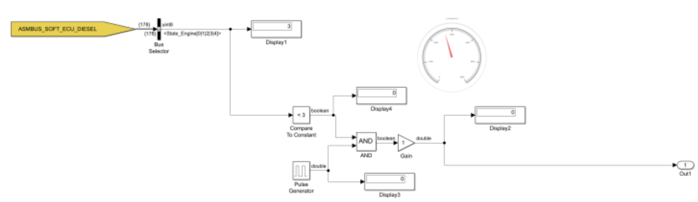

Figure 4: engine cranking subsystem as provided by Jin Ahn for ME 397 HW1

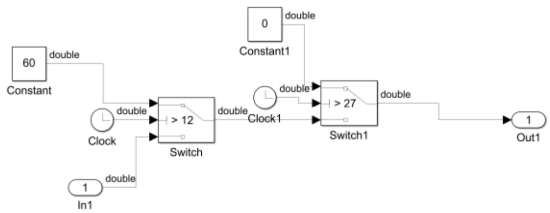

Figure 5: Accelerator pedal subsystem

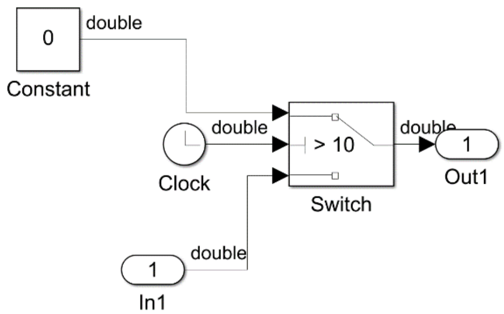

Figure 6: Brake pedal subsystem

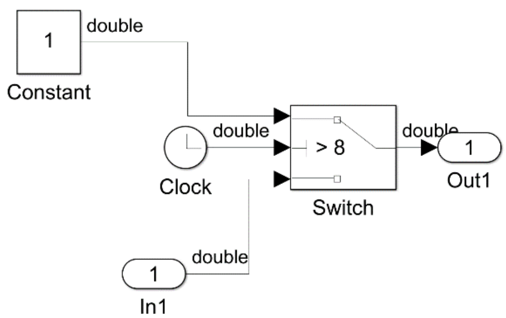

Figure 7: Selector lever subsystem

The engine cranking subsystem automates starting the engine when the simulation is run. The selector lever, brake pedal, and accelerator pedal subsystems automate the acceleration/deceleration phase by moving the selector lever to drive after 8 seconds of simulation (after the engine is started and reaches a steady state idle speed). Then the brake pedal is released after 10 seconds and the accelerator pedal is set to 60% after 12 seconds. Once the vehicle reaches the desired speed (after 27 seconds of total simulation run time), the accelerator pedal is set to 0% input so the vehicle speed will decelerate to the desired highway cruising speed.

The intake manifold pressure controller is setup in ASM as shown in Figure 8. This is located in the air path control block of the SoftECU_Diesel section of the model. Since we only want to look at control performance from the simulation time of 20-27 seconds, a switch is set to only use the controller inputs after 20 seconds. Prior to this, the baseline ASM Soft ECU data for intake manifold pressure is fed into the ASM turbo control block. The desired and current engine torques are fed into the controller block where the desired engine torque is the baseline ASM Soft ECU engine torque data.

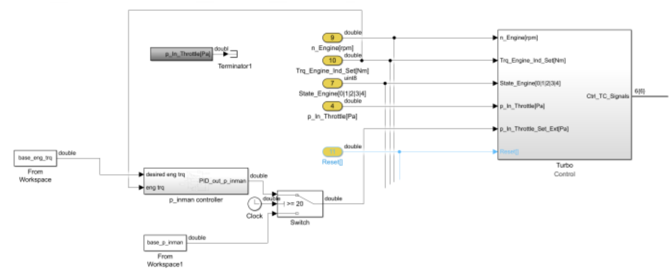

Figure 8: intake manifold pressure controller ASM implementation

The controller subsystem is shown in Figure 9. The integral anti-windup logic is shown and the 1-D lookup table values are given in Section 2, Table 1. The saturation low and high limits for the controller are 0 and 250 kPa respectively.

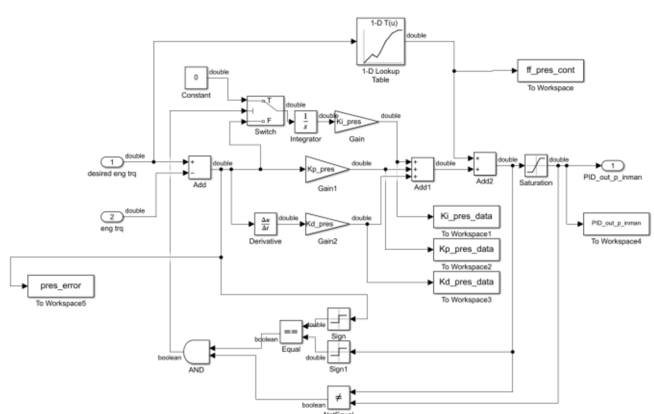

Figure 9: intake manifold pressure controller subsystem

The fuel injection quantity controller is implemented as shown in Figure 10. This is located in the fuel system control block of the SoftECU_Diesel section of the model. Similar to the intake manifold pressure controller, a switch is set to use the controller inputs after 20 seconds. Prior to this, the baseline ASM Soft ECU data for fuel injection quantity is fed into the ASM turbo control block. The desired and current engine torques are fed into the controller block where the desired engine torque is the baseline ASM Soft ECU engine torque data.

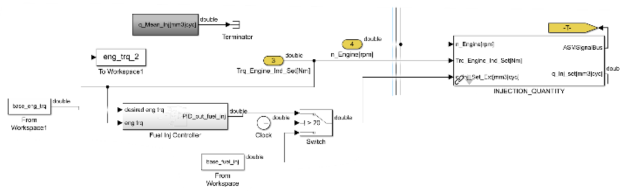

Figure 10: fuel injection quantity controller ASM implementation

The fuel injection quantity controller subsystem is shown in Figure 11. The integral anti-windup logic is present and the 1-D lookup table values are given in Section 2, Table 1. The saturation low and high limits for the controller are 0 and 58 mm^3/cyc respectively.

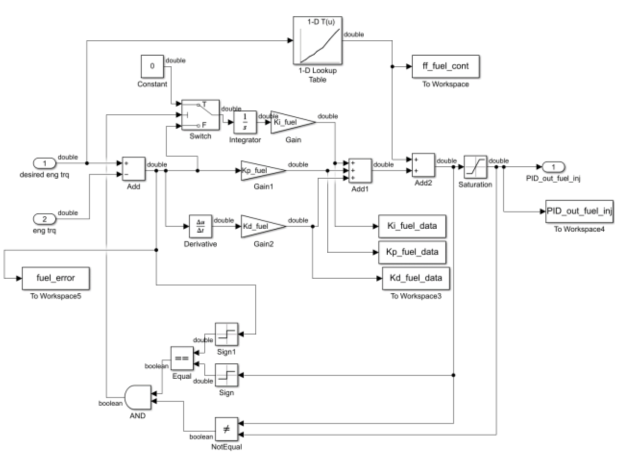

Figure 11: fuel injection quantity controller subsystem

4 Simulation Results and Discussion

As mentioned in Section 2, in order to avoid large spikes in data due to gear shifting, the designed controller comparison is conducted in the time interval of 20-27 seconds for the simulation. This is the longest stretch of the acceleration phase while in a single gear (a gearshift occurs roughly at 20.5 seconds).

In Figure 12, the comparison of the baseline data from the ASM Soft ECU (desired) and the designed controller (true) signals are plotted. The vehicle speed is chosen for comparison in order to gain insight from a user's perspective, the engine torque for an engine performance perspective, and lambda for an emissions perspective. The largest percent error between the two signals for vehicle speed, engine torque, and lambda are 1% at 27s, 4.6% at 26s and 7% at 27s respectively.

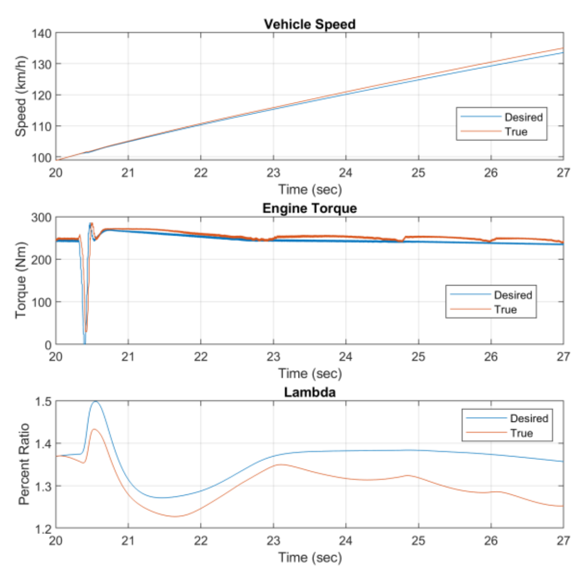

Figure 12: Signal comparison of vehicle speed, engine torque, and lambda between the baseline data (desired) and the designed controllers' results (true)

One cause for the difference in signals can be found in Figure 13. This shows the difference in the amount of fuel injected, and intake manifold pressures for the designed controller and the ASM Soft ECU. This means the designed controllers are not sending quite the same signals to the system model as the ASM Soft ECU.

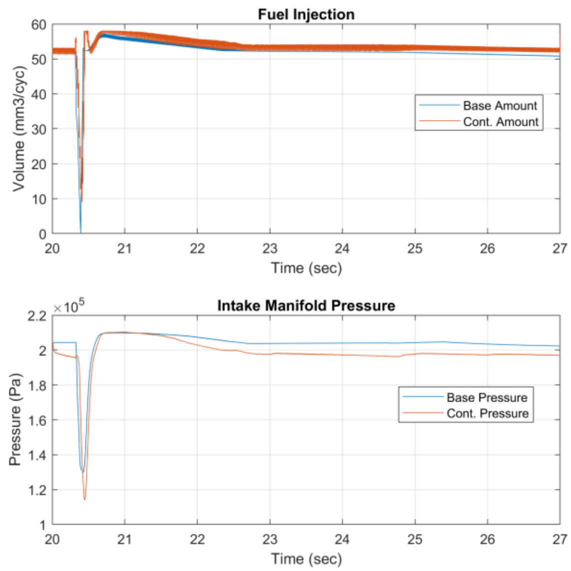

Figure 13: System inputs (fuel injection and intake manifold pressure) comparison between the ASM Soft ECU (base) and designed controllers' (cont.)

Another contribution to the difference in signals is due to the fact that the ASM Soft ECU uses a 2-D map which considers the engine torque and engine speed to determine the fuel injection quantity [3], whereas only engine torque was considered for the designed fuel injection quantity controller. The ASM Soft ECU turbo control block uses the same method, determining the pressure setpoint from the engine torque and engine speed [4].

The fuel injection and intake manifold pressure controller contributions from each term as well as error during the simulation are shown in Figure 14. The feedforward term contributes a large input to each controller. The proportional gain contributes the next highest amount, while the derivative term oscillates significantly, even at relatively low values.

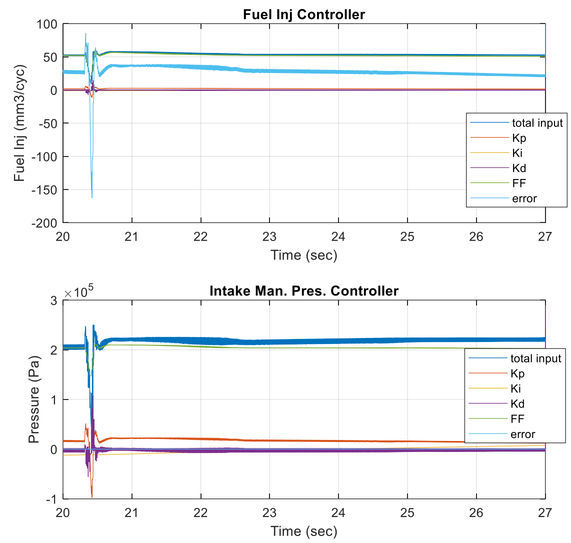

Figure 14: Total controller input into the system for each controller, as well as individual gain and feedforward contributions and the error signal

5 Conclusions and Future Work

This report presents the design, analysis, and implementation into ASM of both a fuel injection quantity and intake manifold pressure controller. The controller performance is compared to the ASM Soft ECU and discrepancies are explained.

In the future, additional work could be completed in order to further the results of this report. First, the controllers could be tuned to perform over the entire range of the simulation. Also, the EGR rate could be turned on and the controllers adjusted to work while the EGR rate is controlled by the ASM Soft ECU. Engine speed could be considered as a reference in addition to engine torque to better match the ASM Soft ECU results, and a filter could be applied to the derivative term of each controller to reduce noise.

Lastly, different control methods could be implemented including robust and optimal control, and digital control effects such as sampling time could be analyzed for hardware implementation.

References

[1] "Understanding PID Control, Part 2: Anti-Windup for PID Control." *MATLAB*, https://www.mathworks.com/videos/understanding-pid-control-part-2-expanding-beyond-a-simple-integral-1528310418260.html.

[2] "The PID Controller & Theory Explained." *NI*, National Instruments, 22 Sept. 2022, https://www.ni.com/en-us/innovations/white-papers/06/pid-theory-explained.html#:~:text=The%20basic%20idea%20behind%20a,components%20to%20compute%20the%20output.

[3] "ASM Diesel Engine Reference." dSPACE GmbH, May 2019, pp. 154. PDF file.

[4] "ASM Turbocharger Light Reference." dSPACE GmbH, May 2019, pp. 31. PDF file.

Appendices – MATLAB Code

1 – Initial Settings

```
%make maneuver type offline manual

%make sure transmission Automatic
MDL.DrivetrainBasic.Sw_Transmission_Mode.v = 2;
MDL.DrivetrainBasic.Sw_Transmission_Mode

%turn off engine EGR valve - set to external mode
MDL.SoftECU.SoftECUDiesel.AirPathControl.EGRRateControl.Sw_EGR_Rate.v = 3;
MDL.SoftECU.SoftECUDiesel.AirPathControl.EGRRateControl.Sw_EGR_Rate

%set fuel injection quantity to external
MDL.SoftECU.SoftECUDiesel.FuelSystemControl.InjectionQuantity.Sw_q_Inj.v = 3;
MDL.SoftECU.SoftECUDiesel.FuelSystemControl.InjectionQuantity.Sw_q_Inj

%set intake manifold pressure to external
MDL.SoftECU.SoftECUDiesel.AirPathControl.TurboControl.Sw_p_In_Throttle.v = 3;
MDL.SoftECU.SoftECUDiesel.AirPathControl.TurboControl.Sw_p_In_Throttle

%start simulation in park, brake at 100%, acc pedal at 0%

%initialize controller gains

%%
%base data timeseries
% base_data = [time lambda fuel_inj p_inman v_veh n_eng eng_trq];
base_lambda = base_data(2);
base_fuel_inj = base_data(3);
base_p_inman = base_data(4);
base_v_veh = base_data(5);
base_n_eng = base_data(6);
base_time = base_data(1);
base_eng_trq = base_data(7);
```

2 – Controller Tuning

```
%set controller gains
%fuel inj controller
Ki_fuel = .009;
Kp_fuel = .07;
Kd_fuel = .0002;

%intake manifold pressure
Ki_pres = 100;
Kp_pres = 600;
Kd_pres = 1.5;

%%
%plots of all controller contributions
%fuel inj controller
subplot(2,1,1)
plot(PID_out_fuel_inj)
title('Fuel Inj Controller')
```

```
xlabel('Time (sec)')
ylabel('Fuel Inj (mm3/cyc)')
hold on
plot(Kp_fuel_data)
hold on
plot(Ki_fuel_data)
hold on
plot(Kd_fuel_data)
hold on
plot(ff_fuel_cont)
hold on
plot(fuel_error)
legend('total input','Kp','Ki','Kd','FF','error')
grid on
xlim([20 27]);

%intake manifold pressure controller
subplot(2,1,2)
plot(PID_out_p_inman)
title('Intake Man. Pres. Controller')
xlabel('Time (sec)')
ylabel('Pressure (Pa)')
hold on
plot(Kp_pres_data)
hold on
plot(Ki_pres_data)
hold on
plot(Kd_pres_data)
hold on
plot(ff_pres_cont)
hold on
plot(pres_error)
legend('total input','Kp','Ki','Kd','FF','error')
grid on
xlim([20 27]);
```

3 – Plots

```
%vehicle speed, engine torque, lambda comparison, engine speed
figure; clf
%vehicle speed
subplot(3,1,1)
plot(base_v_veh)
title('Vehicle Speed')
xlabel('Time (sec)')
ylabel('Speed (km/h)')
hold on
plot(v_veh)
legend('Desired','True')
grid on
xlim([20 27]);

%engine torque
subplot(3,1,2)
plot(base_eng_trq)
title('Engine Torque')
```

```
xlabel('Time (sec)')
ylabel('Torque (Nm)')
hold on
plot(eng_trq)
legend('Desired','True')
grid on
xlim([20 27]);

%lambda
subplot(3,1,3)
plot(base_lambda)
title('Lambda')
xlabel('Time (sec)')
ylabel('Percent Ratio')
hold on
plot(lambda)
legend('Desired','True')
grid on
xlim([20 27]);

%intake manifold pressure and fuel injection quantity
figure; clf
%fuel injection
subplot(2,1,1)
plot(base_fuel_inj)
title('Fuel Injection')
xlabel('Time (sec)')
ylabel('Volume (mm3/cyc)')
hold on
plot(fuel_inj)
legend('Base Amount','Cont. Amount')
grid on
xlim([20 27]);

%intake manifold pressure
subplot(2,1,2)
plot(base_p_inman)
title('Intake Manifold Pressure')
xlabel('Time (sec)')
ylabel('Pressure (Pa)')
hold on
plot(p_inman)
legend('Base Pressure','Cont. Pressure')
grid on
xlim([20 27]);
```