Engine-based powertrain (TTR-HEV) modeling and Simulation-PI Tuning Problem

Sajjad A.Anbaran

I. Introduction

This report intends to present problems and challenges on modeling and tuning of PIs for a TTRHEV powertrain , specifically front driveline of the TTRHEV. There are two drivelines in a TTRHEV that are, in concept, operating independently. However, they both are coordinated and dictated on by a supervisory control logic. The front driveline is engine-based with a multi-speed transmission and front wheels. As for rear driveline, there are two electric machines, battery, power electronic converter and rear wheels. Of course, all these components are accompanied by their associated controller to regulate their behavior in order to meet design performance and efficiency goals.

From control standpoint, there are two layers of controllers. First is the lower layer that regulates the TTRHEV components dynamic behavior such that the entire powertrain as whole, follows the dictated commands by supervisory control logic. At this layer, the controllers are realized through conventional PI controllers. On the higher layer, there is an event-driven decision making system to interpret driver's desire and then outputs set of commands to meet driver's wish. An event is any state or condition at which a car can be. How to perceive an event can vary in accordance to road condition, powertrain condition, driver's habit, and etc. There could be many possibilities on the number of events. One way to address this problem is to use engineering intuition and knowledge and professional driver's remarks to help the design engineer to define events and assign these events to a set of rules/ actions. This study, realized this task through stateflow toolbox of MATLAB/Simulink. System modeling and model fidelity is as equally important as the control system design. For the sake of control design for a TTR-HEV a system-level modeling is sufficient. Simscape toolbox and its libraries offer acausal modeling approach in contrast to causal modeling that is available in Simulink. Hence, SimDriveline, SimMechanics, SimElectronics libraries of Simscape used to build TTRHEV model. The focal point of this report, however, is to give a detailed account of hurdles on front driveline control design. You will be presented with front driveline models, and their description. Next, the control problem of front driveline such that vehicle speed tracks the reference speed profile is discussed.

II. Description of the system

Front driveline configuration and modeling with Simscape

There are three models of front driveline presented here; each model is simplified version of the previous model. For the sake of simplicity and clarity, these models are named A, B, and C. The rationale behind this simplification process is to alleviate the dynamics of the model, so to be able to ease the tuning task of PI controller. This part of report describes these three models and blocks that constitute them. And next section elaborates on control challenge(s) associated to these models. First, Model A, that is a fully-fledged driveline where you have Engine, Torque converter, Continuously Variable Transmission (CVT), Vehicle body dynamics, and Gear selection logic, Figure 1. This model is highly dynamic and nonlinear owing to presence of transmission system and gear selection logic; states of the system are constantly changing, Figure 2. Hence, the design and tune a linear controller (PI) for this system needs much effort and skill to deal with. To view the inside of subsystems in Model, refer to Appendix.

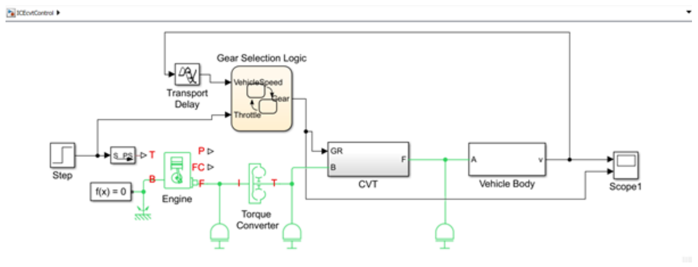


Figure 1: Model A, Complete model of front driveline

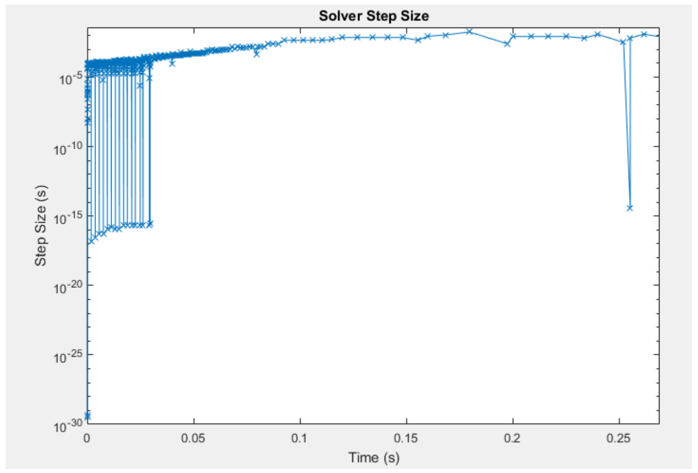


Figure 2: Solver step Size plot for Model A

Model A is further simplified by removing Gear Selection logic and replaced with a signal builder block, Figure 3 on the following page. The signal block imitates Gear Selection logic and sends out a continuously changing gear number to

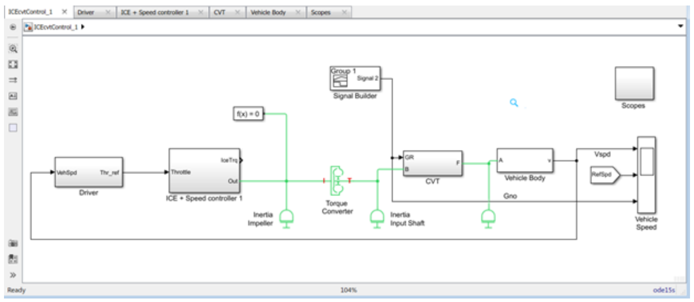


Figure 3: Model B, Front driveline

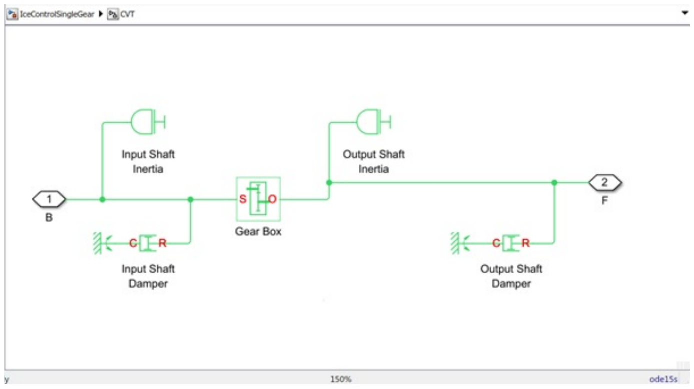


Figure 4: Model C, Single-speed gearbox

the CVT. Meanwhile, driver model is added to the system. Detailed view of the driver model and signal builder block is available in appendix.

However, attempts to obtain a valid linearized model of the Model B were not successful. Therefore, it is decided to further simplify model B which leads to Model C. At this stage, CVT is deleted and replaced with a single gear, Figure 4,nevertheless, the rest of driveline is retained.

Figure 5 and 6, respectively, show the step size plot of Model B and Model C. The discontinuities and rapid changes have slightly decreased in Model B and we have further decrease in Model C.

III. Description of Front driveline control problem

At this stage of modeling and simulation, the control problem is to control engine torque such that vehicle tracks the reference speed profile. The type of the controller is a conventional PI controller and the objective of the control design is ‘reference-tracking’.

A. Driver Model

The reference speed profile could be any of standard drive cycles or a custom-made speed profile of the vehicle. To begin with control design, the reference signal constructed using signal builder block whereby the vehicle speeds up to 50 Kilometer per hour (kph) in 10 seconds at rate of +5 kph/s, then cruises at 50 kph for the next 10 seconds, ultimately, it slows down to zero speed in 10 seconds with the rate of -5 kph/s, Figure 7.

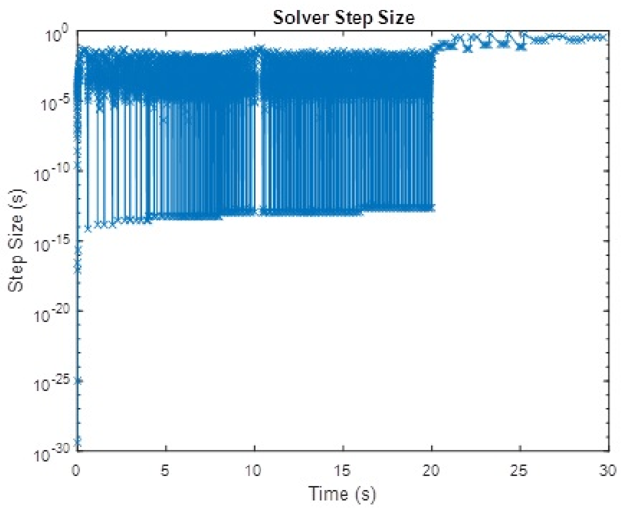


Figure 5: Step size plot, Model B

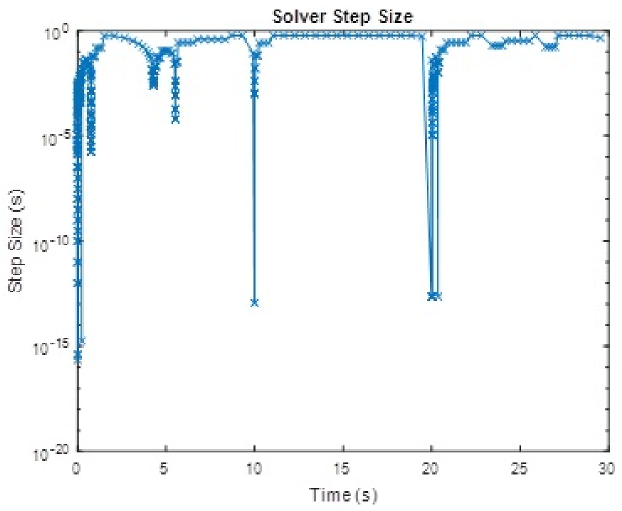


Figure 6: Step Size plot, Model C

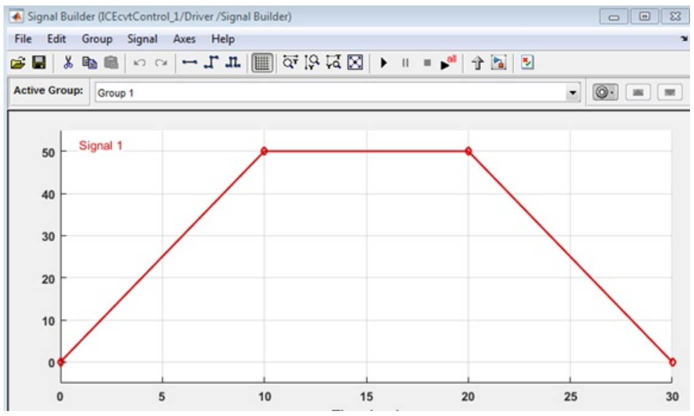


Figure 7: Test Drive cycle

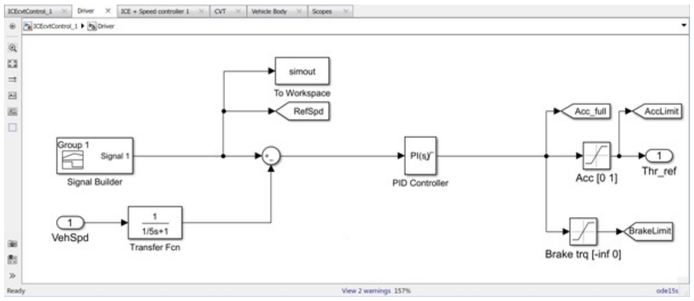


Figure 8: Driver Model

B. *Engine Torque Control*

Engine torque is controlled through Driver model. Figure. shows the inside view of driver subsystem. Outputs of driver subsystem are Accelerator and Brake pedal signals. The Accelerator pedal signal equates to demanded torque that the engine must supply to meet the control objective. This signal varies between 0 and 1 [0= closed throttle and 1= opened throttle] and fed to engine block. To compute the demanded torque, the reference speed is compared to the car actual speed and the difference is fed to a PI controller block. MATLAB/ Simulink offers several methods to design and tune PIDs. The common principle among all these methods is that they all require a linearized model of the system. Nevertheless, front driveline model with engine, multi-speed transmission, and gear selection logic is highly nonlinear. This non-linearity should be linearized before any attempts made to tune PI controller. Once the linearized model is obtained, it should be validated then one can proceed to use linearized model to tune PI controller(s).

IV. Linearization with Matlab / Simulink

The important motive to linearize a model is to obtain an approximated linear model of a system, so that it could be used to design and tune conventional PI controller. Linearization is the process of finding the linear approximation of a model at the vicinity of the equilibrium point(s). Equilibrium points are operating points of the model at which the model is at steady-state. The steady-state operating points include variables that do not change with time aka steady-state variables. The question, in turn, rises is that what states should be at steady-state, and how to identify them? MATLAB/simscape lays out the entire states of the model as a state vector x which contains Simulink components (both continuous and discrete) and Simscape components (continuous). This makes it difficult to analyze operating points of the model and impossible to identify state variables of the driveline. However, the number of the state variables for a driveline can be counted, and it is equal to number of independent Degree of freedom of the driveline, provided that all clutches are unlocked. Not all states are required to be at steady-state (or equilibrium). For instance, the variables in cruise control problem of an HEV are vehicle position and velocity, fuel and air rate into Engine. Since the car is moving, the position is changing and not having it as steady-state variable in linearization would not cause invalid linearized model. In a further complex system, good knowledge of the system dynamic and behavior over the course of simulation can greatly help on choosing variables that are needed to be at steady-state for the linearization. This freedom of choosing steady-state variable is available by MATLAB 'trim' command and its graphical equivalent of Simulink control design Toolbox at 'Linear Analysis Tool'. Nevertheless, using 'trim' command is not supported if the model is a Simscape model. However, 'Linear Analysis Tool' is supported by Simscape models whereby a model can be linearized either at a certain time (simulation snapshot) or at model default operating points, or at selected states. MATLAB/ Simulink offers several ways to linearize a Simscape model and they are listed in this part of report. Moreover, Simulink control design toolbox allows PI controller tune and design without necessarily having the knowledge about the model dynamics and linearization. This, nevertheless, does not often yields to a good design for a model developed in simscape environment. To be able to successfully design and tune PI controllers for a simscape model, having a good understanding of how model works and behaviors is tremendously important.

A. *Simscape model linearization*

Standard procedure to design and tune PI controllers in Simulinik involves following steps:

1) Find steady-state operating point(s)
2) Linearize the model at steady-state operating point(s)
3) Model validation with 'Frequency Response Estimation'
4) Tune PI for the linearized model

MATLAB help documentation advises to use the following methods to find steady-state operating points for a Simscape model

1) Simulation in time to search for an operating point
2) Using Simscape initial condition solver
3) Using Simulink control design techniques to find operating points

It must be noted that MATLAB help does not recommend using sources to find operating points, and as it was mentioned earlier, 'trim' command is not supported with Simscape models. Finding operating point and linearization is a critical step toward designing PI controllers. There is no guarantee that there is any operating point for the model or the operating point is suitable for linearization. Therefore, it is critical to analyze the operating point by full simulation of the model, away from any discontinuity, and varying parameters, input, initial condition. Once a good operating point is found, the model can be linearized at the neighborhood of that operating point. MATLAB help advises to use following methods to linearize models containing Simscape components.

1) Linearize with Simulink 'linmod' and 'dlinmod' functions
2) Linearize with Simulink control design software
3) Linearize with Simulink Linear Analysis Tool

There is, however, another method to deal with control problem beside above methods. This method does not require linearization. Here 'System Identification Toolbox' is used to identify best suit approximate of the non-linear model. Next,

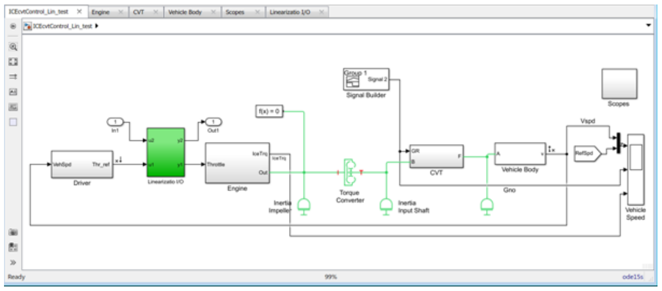


Figure 9: Model with 'linmode'

the control problem studied in this report is presented and then all above methods to search for a suitable operating point to linearize the model and tune PI controllers are explored and discussed.

B. Model Linearization with 'linmod' function

Simscape states are by default continuous, therefore, 'linmod' function is a good choice for the purpose of the linearization of the model under study in this report. There are several ways to use Simulink function 'linmod' to perform linearization and the result can be different depending on the method selected. these methods are

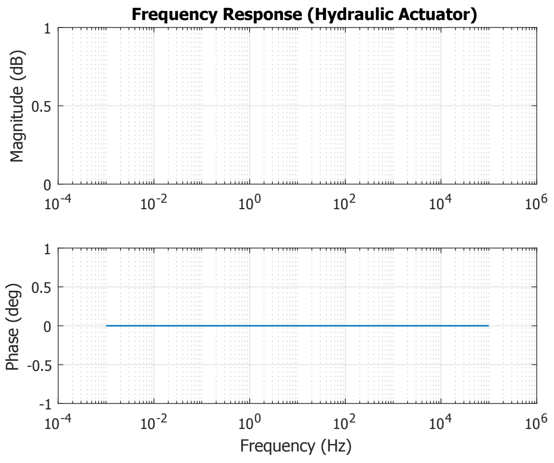
1) Linearize with default input and states
2) Linearize with the steady-state solver at an initial steady state
3) Linearization with specified state and input

First and the second method are ruled out, simply because of the fact that they do not produce acceptable simulation result, Figure 10. In particular, linearizing the model with steady-state solver at steady state distorts the output signal shape. On the other hand, linearization using the 'linmod' function with specific state and input does not demonstrate the same complications there are with method one and two. Thus, this report explores linearization with 'linmod' function at specific state and output. For this purpose, model-level input and output are added to the part of model (Plant) that are to be linearized , Figure 9.

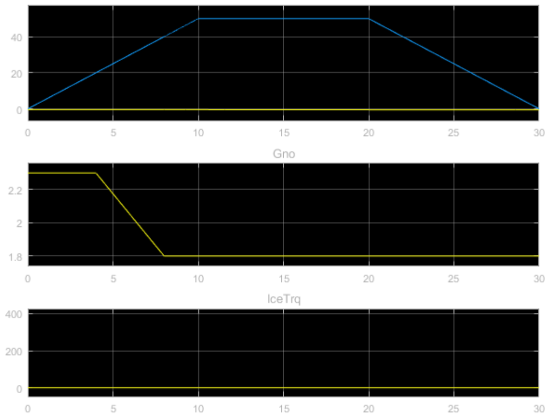
First, the model is simulated with feedback loop closed in order to analyze simulation behavior at the output (vehicle velocity mph). It is observered that vehicle actual velocity is oscillating however it is linear for very short period of time in a periodic manner, Figure 11. It is arbitrary to pick any of the windows of time at which model behaves linearly to linearize the plant, so that $t = 6.05$ is selected. Figure 12 on the following page is the result of linearization with 'linmod' function at a specific state and input.

It is clear that Figure 12 on the next page and Figure 10a are identical. Both of the figures linearized at zero. MATLAB help documentation suggests few solutoin to deal with plants that cannot be linearized ot linearized to zero and there are as followings

1) linearize at a different operating point
2) import a linear model of the plant to PID tuner
3) Tune controller with model obtained from 'Estimated Frequency Response'

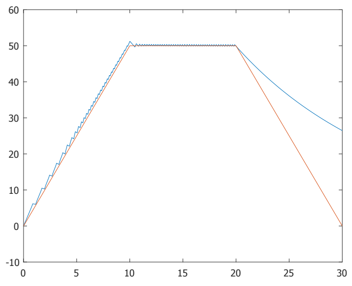


(a)

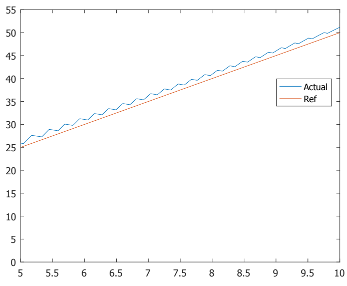


(b)

Figure 10: Bode response of the linearization a) with 'linmod' at default states and b) with the steady-state solver at an initial steady state



(a) Full scale



(b) Zoomed

Figure 11: Vehicle velocity

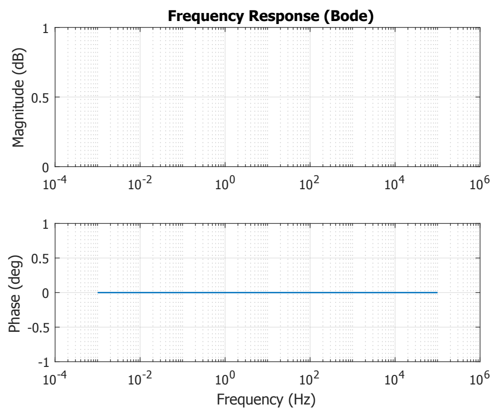


Figure 12: Bode diagram of linearization with 'linmod' at a specific state

4) Use 'System Identification Toolbox' to estimate a linear plant model

This section dicusses the first item in the list only and the rest are discussed in their repective sections.

The plant model has been linearized for several operating points and the result was identical to that of Figure 12, nevertheless, they are not presented here. To automatically linearize a model at multiple operating points without running simulation, MATLAB offeres 'Batch Linearization' technique but this report does not intend to explore that at the moment.

C. Model Linearization with 'Linear Analysis Tool'

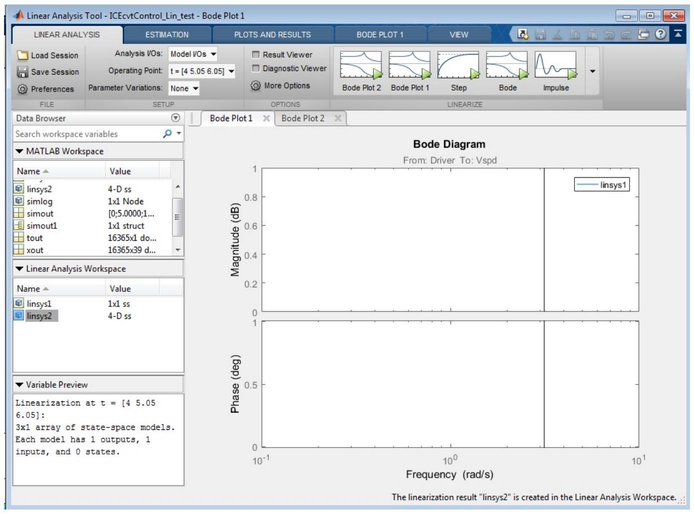
To linearize models, Linear Analysis Tool is the specific tool provided by MATLAB and it is the graphical equivalent of MATLAB 'linearize' function. It provides GUI environment to perform variety of tasks such as linearization, and frequency response estimation.

the linearization process begins with specifying input and outpu signals of open-loop plant , Figure 9 on the preceding page. It is expected the result to be similar to what obtained with 'linmod' function, because both techniques use the same approach to linearize a model. Figure 13 shows the Linear Analysis Tool front page. The model is linearized at its default operating point as well as at $t = [4, 5.05, 6.05]$. Figure 14 is the linearizatoin reult.

D. Model linearization with 'Frequency Response Estimation'

'Frequency Response Estimation' returns frequency response data serves several purposes. Amongst them are validation of fidelity of the linear model obtained from other linearization methods and an alternative to design PID controllers when the linearized plant model is invalid for PID design.

Before proceed to estimate frequency respose of the model, a test performed to understand the model behavior. The result from execution of follwing codes, Figure 15 on the next page,explains why the model under study is linearizing to zero. The model contains a static gain that causes the model linearizes to zero. This usually happens when the model contains some discontinuity caused by event-based blocks such



Analysis I/Os: Model I/Os
Operating Point: t = [4 5.05 6.05]
Parameter Variations: None

MATLAB Workspace

Name	Value
linsys2	4-D ss
simlog	1x1 Node
simout	[0;5.0000;1...
simout1	1x1 struct
tout	16365x1 do...
xout	16365x39 d...

Linear Analysis Workspace

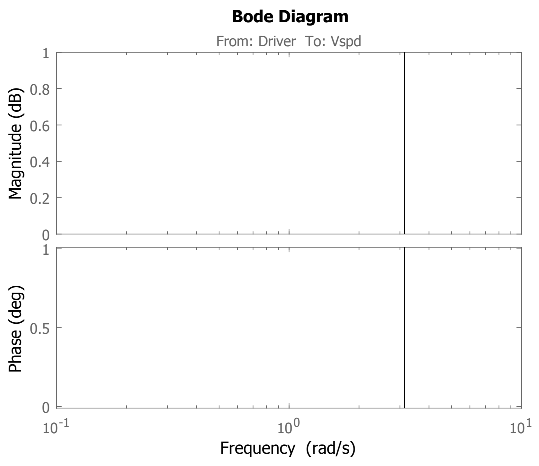
Name	Value
linsys1	1x1 ss
linsys2	4-D ss

Variable Preview

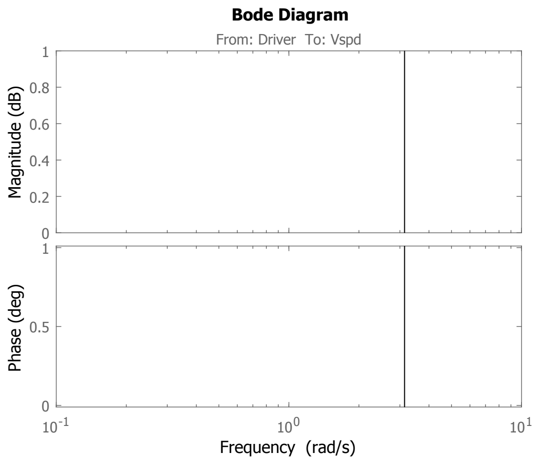
```
Linearization at t = [4 5.05
6.05]:
3x1 array of state-space models.
Each model has 1 outputs, 1
inputs, and 0 states.
```

The linearization result "linsys2" is created in the Linear Analysis Workspace.

Figure 13: MATLAB Linear Analysis Tool

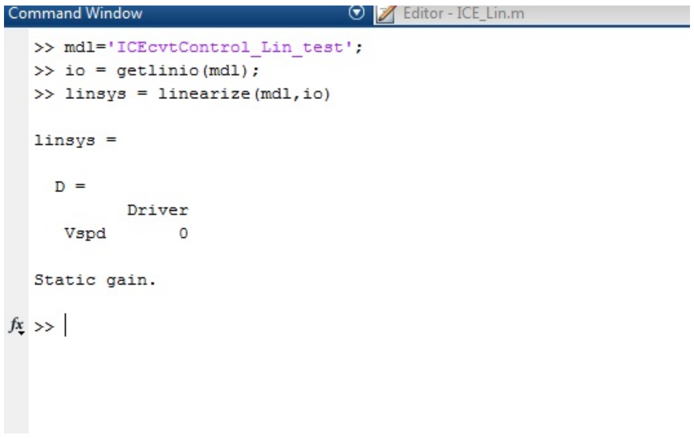


(a) Linearization at Model default operating points



(b) Linearizatoin at $t = [4, 5.05, 6.05]$

Figure 14: Model Linearization result



```
>> mdl='ICEcvtControl_Lin_test';
>> io = getlinio(mdl);
>> linsys = linearize(mdl,io)

linsys =

  D =
           Driver
   Vspd         0

Static gain.

>> |
```

Figure 15: Model Linearization test code

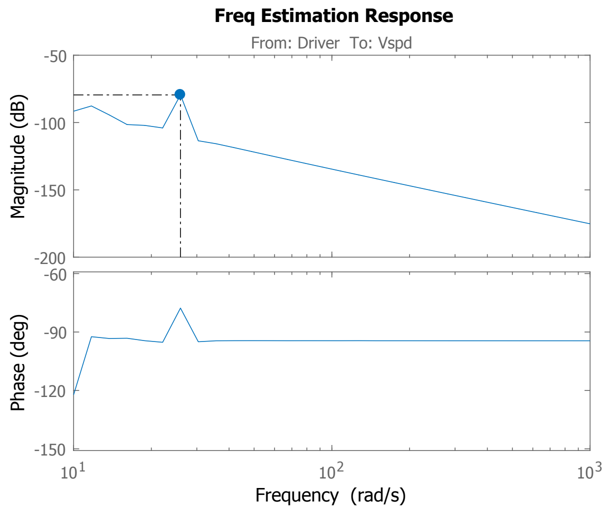


Figure 16: Frequency Response Estimation

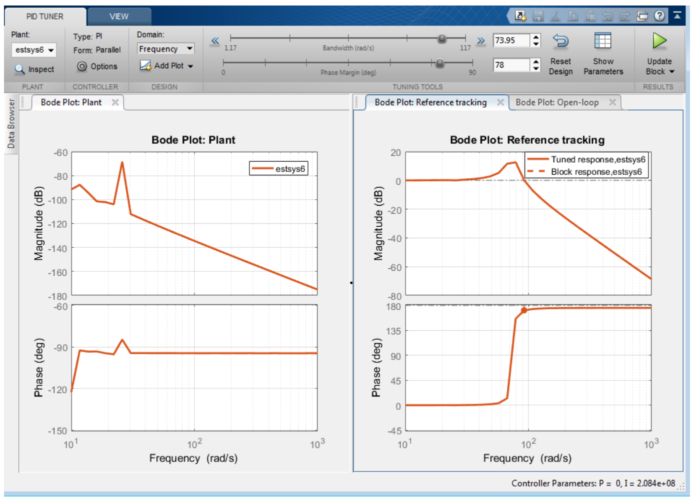


Figure 17: PID design with Frequency Response Estimation

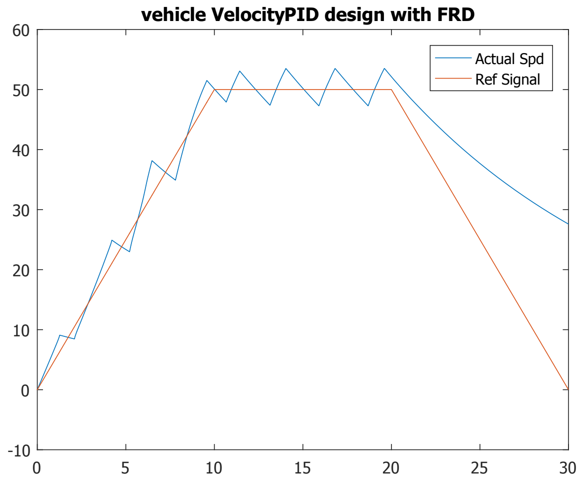


Figure 18: Vehicle velocity with PID design Frq Response Estimation

as stateflow block or triggered-based subsystems. However, this is not the case for the model in this study and the reason of this issue is unfathomable to the author at the moment.

'Frequency Response Estimation' is accessible through Simulink →Analysis → Control Design → Linear Analysis. This action opens the Linear Analysis Tool for the model. Next, in linear analysis tool, and finally click on Estimation Tab. In the Input Signal drop-down list, Sinestream signal selected in Frequency range of 10 to 1000 with Amplitude of 1. This returns frequency response data and plotted with Bode diagram, Figure 16.

To continue with PID design, the frequency response data should be moves to MATLAB workspace and then imported to PID Tuner block for further analysis and design, Figure 17.

The vehicle velocity (output), Figure 17, indicates that car fails to track reference signal and the response is slower that of obtained from previous methods.

E. Model Linearization/ PI Tune with Simulink Control Design software

Irrespective of type of the methods provided by'Simulink Control Design Software', all of them require model linearization before proceed to PI tuning. This is particularly true for Simscape Models. Several methods available at 'Simulink control Design software' are explored in this section and the objective is to tune $k_p$and k_i values to accomplish 'reference tracking' goal.

1) Control System Designer: Control System Designer is a tool for classical control design offering various graphical and automated tuning methods (Analysis →Control Design→Control System Designer). It is a good tool to tune compensatros in models that contain multi-loop or cascade feedback loops. An example is a full HEV powertrain model. However, the model under study here is a simple powertrain with single feedback loop. Design workflow with 'Control System Designer' is as following:

1) Select blocks to be tunes
2) Add analysis point to specify the portion of model to be linearized
3) Specify linearization options. Here the model chose to be linearized at $t = 6.05$.

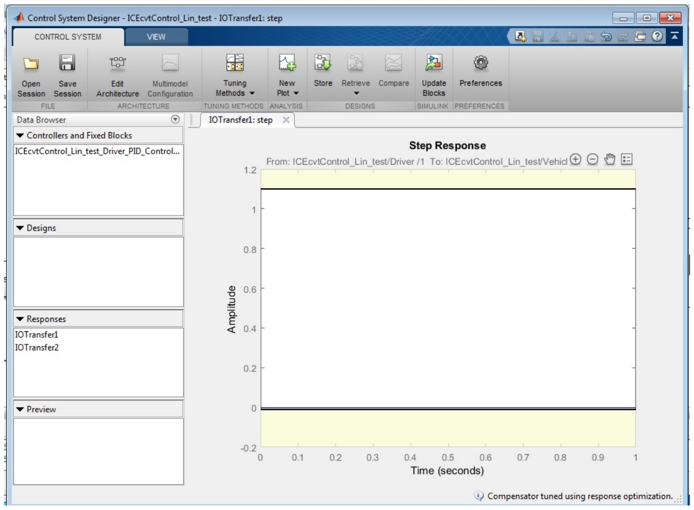


Figure 19: Control System Designer

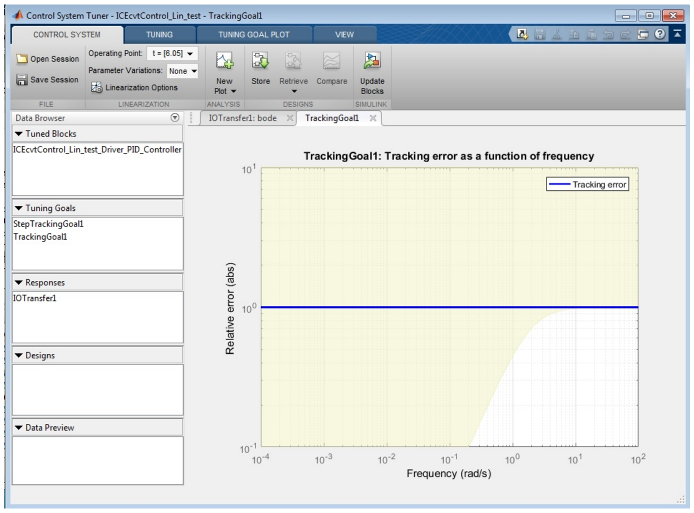


Figure 20: Control system Tuner result

4) select a tuning method

Several tuning methods are used to tune PI controller but none returned a good result. These methods are:

1) Bode Diagram Design
2) Root Locus Design
3) Nichols Plot Design

Ultimately, ’Optimization Based Tuning’ used, nevertheless, it did not produce any acceptable result, Figure 19. Likewise, different optimization techniques that are available as option in this method are examined.

2) Control system Tuner: Control System Tuner automatically tunes the controller parameters to satisfy the must-have requirements (design constraints) and to best meet the remaining requirements (objectives). Design workflow with this tool is

1) Model set-up
2) Tuning goals
3) Tuning, Analysis, and validation

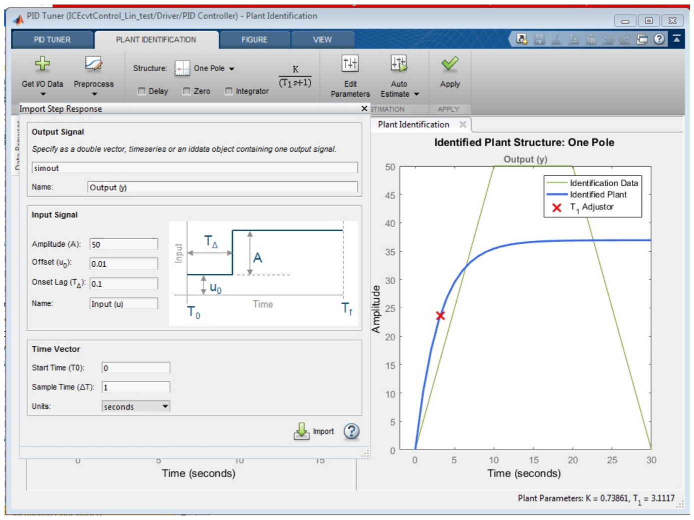
The Model set to be linearized at $t = 6.05$ and PI controller block is selected as tunable block. The Tuning goal is set to ’Reference Tracking’ and the auto tuning performed. Figure 20 is the result of Linearization/ PI tuning with ’Control System Tuner’.

The model linearized to zero and consequently the tuning result is unaccaptable.

F. Model identification with ‘System Identification Toolbox’

In many situations, like the system under study in this report, a dynamic representation of the system is not readily available. One solution to this problem is to obtain a dynamical model using identification techniques. One of the many methods MATLAB offers to tune PI controllers for a non-linear system is to use ’System Identification Toolbox’. The plant model can be estimated from measured or simulated response data. Measured data is not available, so instead simulated input/output data will be used.

PID Tuner block provides for both plant identification and controller design in a single interface. Input/output data and


PID Tuner (ICEcvtControl_Lin_test/Driver/PID Controller) - Plant Identification

Structure: One Pole $\frac{K}{(T_1 s+1)}$

Import Step Response

Output Signal

Specify as a double vector, timeseries or an iddata object containing one output signal.

simout

Name: Output (y)

Input Signal

Amplitude (A): 50

Offset (u_0): 0.01

Onset Lag (T_Δ): 0.1

Name: Input (u)

Time Vector

Start Time (T0): 0

Sample Time (ΔT): 1

Units: seconds

Plant Parameters: K = 0.73861, T_1 = 3.1117

Figure 21: System Identification simulated input/output

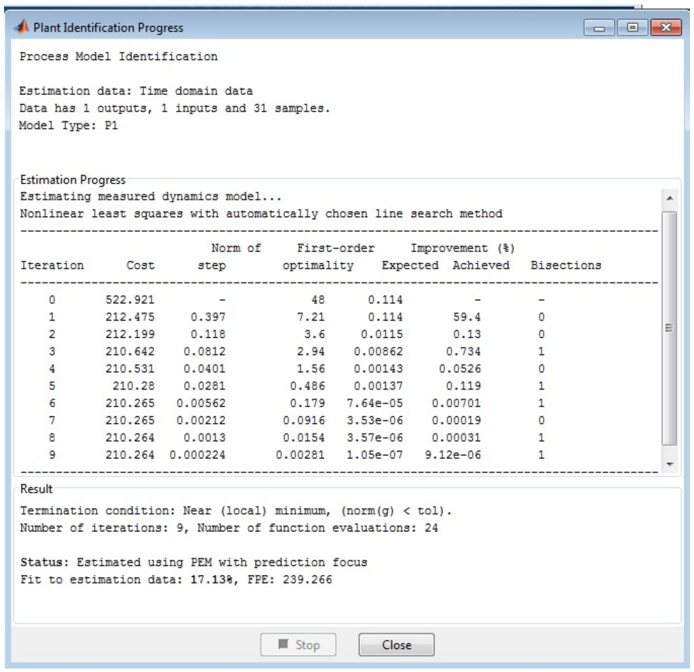
use it to identify one or more plant models., however, PID Tuner can only identify single-input, single output, continuous-time plant models. This poses no issues to coduct plant estimation.

The simulated input signal has the following specifications;

Amplitude = 50, Offset= 0.01 and Onset lag = 0.1.

As for the simulated output signal, the simulation output logged into workspace as timeseries and then fed as output signal, Figure 21.

Plant Auto-estimated, Figure 22 on the following page and ready for fine tuning before saved and fed to the PID Tuner block. Figure shows the plant estimated with Underdamped pair next to the step plot for Reference Tracking. The result is the same if other plant structure for identification is used. However, There has been an slight improvment in the system output (Vehicle Velocity). In comparison to Figure 11 on page 4 the Velocity oscillatoin in Figure 24 on the following page is eliminated, particularly after $t = 4$sec but still the car does not track the reference as it supposed to.



```
Process Model Identification

Estimation data: Time domain data
Data has 1 outputs, 1 inputs and 31 samples.
Model Type: P1
```

Estimation Progress

```
Estimating measured dynamics model...
Nonlinear least squares with automatically chosen line search method
--------------------------------------------------------------------------------------
                         Norm of      First-order     Improvement (%)
Iteration       Cost       step        optimality   Expected  Achieved   Bisections
--------------------------------------------------------------------------------------
    0       522.921          -               48       0.114         -         -
    1       212.475      0.397             7.21       0.114      59.4         0
    2       212.199      0.118              3.6      0.0115      0.13         0
    3       210.642     0.0812             2.94     0.00862     0.734         1
    4       210.531     0.0401             1.56     0.00143    0.0526         0
    5        210.28     0.0281            0.486     0.00137     0.119         1
    6       210.265    0.00562            0.179    7.64e-05   0.00701         1
    7       210.265    0.00212           0.0916    3.53e-06   0.00019         0
    8       210.264     0.0013           0.0154    3.57e-06   0.00031         1
    9       210.264   0.000224          0.00281    1.05e-07  9.12e-06         1
--------------------------------------------------------------------------------------
```

Result

```
Termination condition: Near (local) minimum, (norm(g) < tol).
Number of iterations: 9, Number of function evaluations: 24

Status: Estimated using PEM with prediction focus
Fit to estimation data: 17.13%, FPE: 239.266
```

Figure 22: Plant Identification report

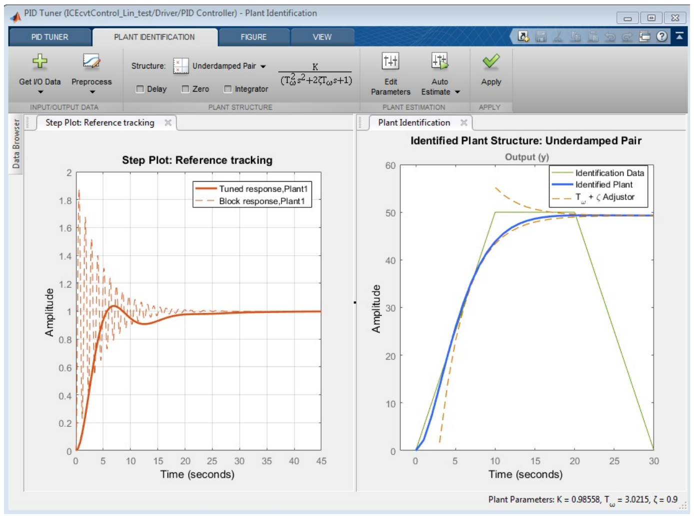


Figure 23: Identified Plant withUnderdamped pair

V. Conclucsion

This report is an attempt to describe the problem in hand at this stage of study. It basically comprised of three parts. First, the model and control problem is described, then proceeded to discussion over simscape model linearization and available method to perform linearization analysis and PID design. Ultimately, the third part of report explores each of the analysis and design methods and presents the result with some discussion.

A Simscape model of conventional car is developed. The car powertrain consists of Driver Model, Engine, CVT, and vehicle body. The objective of this study is to tune PI controller in the Driver model subsystem via classical PID design methods such that vehicle tracks the reference signal. MATLAB/ Simulink offeres several methods to design and tune PID controller.

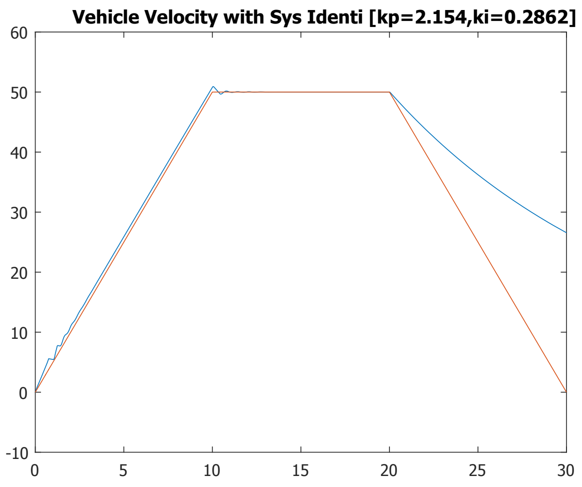


Figure 24: Vehicle velocity output with System Identification

However, they all need a linearized approximated of the model to be able to tune k_p and k_i values of PI controller, except for methods such as ’Frequency Response Estimation’ and ’System Identification Tool’. Moreover, not all of the available methods are applicable to Simscape models. MATLAB Help documentation is voluminous on the subject of Model linearizationa and PID design. Nevertheless, for a Simscape model, suitable methods can be jotted down as following

1) Linearize with Simulink ‘linmod’ and ‘dlinmod’ functions
2) Linearize with Simulink control design software
3) Linearize with Simulink Linear Analysis Tool

Model linearizatoin with a ’linmod’ function and ’Lineari Analaysis Tool’ both linearized to zero. These two methods, fundamentally, use same linearization approach to deal with a model. However, the reason why model linearizes to zero is not clear at this stage of study, although many attempts were made to linearize the model at different time and operating points with various optimation search methods. In these kind of cases, MATLAB recommends to use follwing methods

1) linearize at a different operating point
2) import a linear model of the plant to PID tuner
3) Tune controller with model obtained from ’Estimated Frequency Response’
4) Use ’System Identification Toolbox’ to estimate a linear plant model

It has already been tried to linearize and re-linearize the model at different operating point but it was fruitless. The second recommeded method is ruled out due to fact that no linear model of the plant is at disposal. On the other hand, third and forth recommended methods presented promising results. Although frequency response data of the model provides a linearized approximate, it does not yield to a good PI tuning. The vehicle velocity trajectory does not follow the reference signal,nonetheless, it contains less oscillacion but the amplitude of scillation is bigger than that of obtained from ’linmod’ function. A simple analysis of the model revealed the actual reason model linearizes to zero with ’linmod’ function and ’Linear Analysis Tool’. It is because the model contains ’Static Gain’ along the linearization path. This problem is common with models that made up of event-based blocks or trigger-based subsystem. But this is not the case in the model under study in this report, because of the fact that none of these blocks or subsystem are not present in the model and in linearization path. Ultimately, System Identification technique is used via PID Tuner block. The identification process was successful. A model was identified and fine tuned by adding ’Underdamped pair’ of poles and then fed to the PID Tuner block for further analysis and tuning. The output of simulatoin shows an improvement by significantly resducing oscillatoin at vehicle velocity.

Lastly, Simulink Control Design software is a package of several methods to design and tune PIDs with the use of graphical means of design such as bode diagram, root locus and Nichols diagrams. Even though having good knowledge of linear control system theory is not necessary sometimes, it is not valid when dealing with Simscape models. Due to the fact that the model is non-linear and classical PID tuning technique is being used, the model still need to be linearized when working with ’Simulink Control Design software’. It is found that when linearize with methods provided by ’Simulink Control Design software’, they all use the same technique as it is used by ’linmod’ function and ’Linear Analysis Tool’ to linearize the model. Hence, the PID design founded on invalid linearized model and leads to unsatisfactory results.

This report examined myriad of methods MATLAB offeres to analyze, design, validate PID controllers, particularly those are supported by Simscape model. They all come with their pros and cons, and few of them can be ruled out. However, what is certain is that deep grasp of the Model dynamic and behaviour would tremendeously ease the process of PID design. At the time of the writing this report, the author is not able to fathom many of issues in the model design and control problem. Thus, further study and investigation into the problem is a must.

VI. Future work

Over the course of report write-up, it was observed that Vehicle output with ’System Identification’ method, Figure 24 on the previous page has longer linear region in contrast to Figure 11 on page 4 and Figure 18 on page 6. A longer window of linear behavior of system could lead to a better linearization result. Therefore, the model was again linearized at $t = 6.05$sec once with ’linmod’ functoin and another time with ’dlinmod’ function, Figure 25 and Figure 26, respectively.

Admittedly, this time neither of ’linmod’ and ’dlinmod’ fucntions linearize the model to zero like what observed in Figure 12 on page 5. This reveals another reason for linearization at zero beside presence of ’static gain’. Obviously, the length of linearization region was inadequetly short. Next, these linearized models will be examined, validated and used to tune PI controller.

On another note, the simulation of Electric drivetrain is complete. Necessary preparations are being made to begin with Desktop Real-Time Simulation of Electric Drivetrain and then HIL.

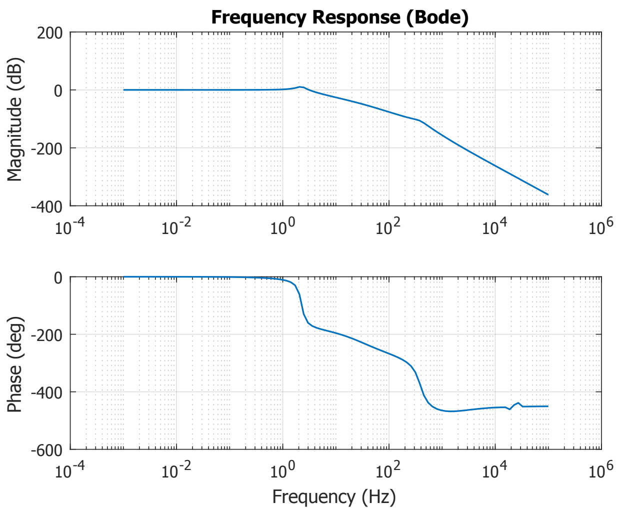


Figure 25: Linearizatoin with ’linmod’ function

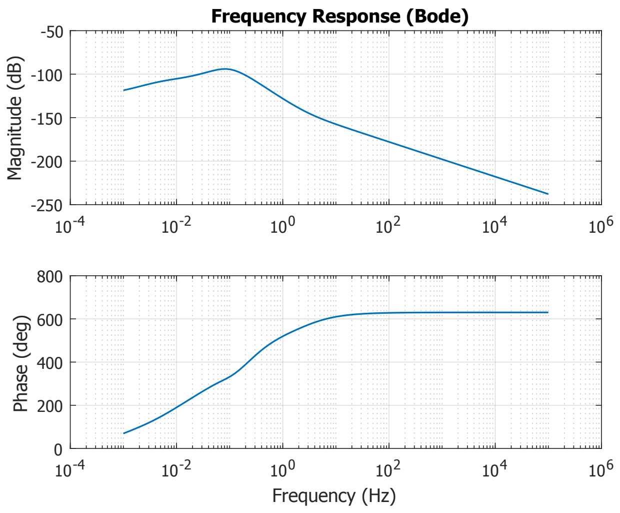


Figure 26: Linearization with ’dlinmod’

VII. Appendix

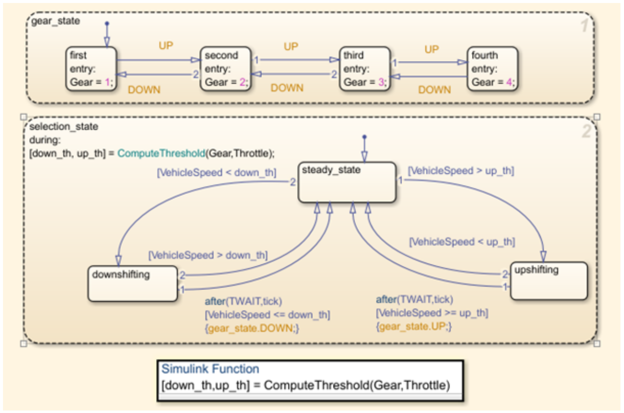


Figure 27: Inside view of Supervisory Control Logic

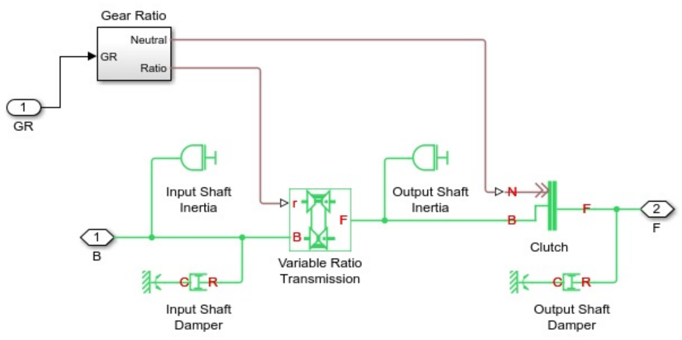


Figure 28: Inside view of CVT Subsystem

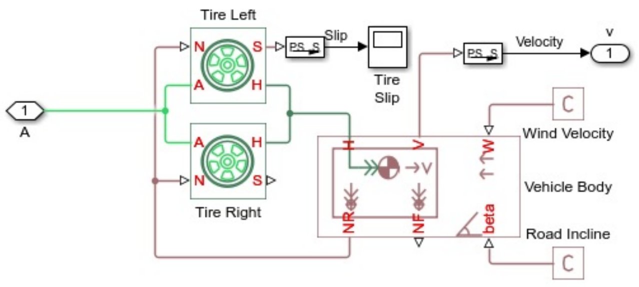


Figure 29: Inside View of Vehicle Body