Automatic Transmission Controller

Name: Priya Gangadhar Dhore
Emp-ID: 214919
Track : Powertrain
Module: Model Based Design
Email: Priya.Dhore@kpit.com
Location: Pune

Table of Contents

Table of figures:

Introduction:

The system chosen is an Automatic transmission controller. Nonlinear ordinary differential equations model the engine, four-speed automatic transmission, and vehicle. The model for automatic transmission controller directly implements the blocks from below figure as modular Simulink subsystems. On the other hand, the logic and decisions made in the Transmission Control Unit (TCU) do not lend themselves to well-formulated equations. TCU is better suited for a Stateflow representation. Stateflow monitors the events which correspond to important relationships within the system and takes the appropriate action as they occur.

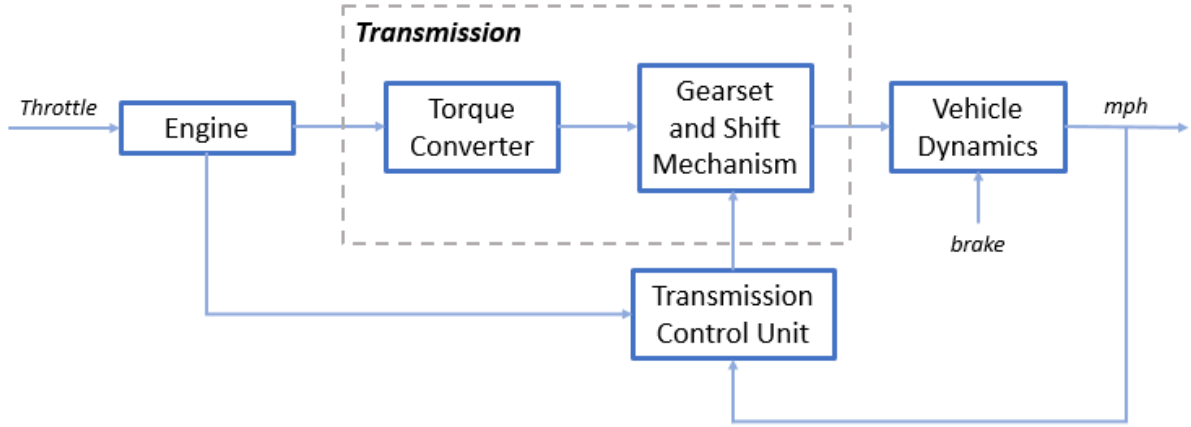


Figure 1 : Block diagram

1. Analysis and physics:

The throttle opening is one of the inputs to the engine. The engine is connected to the impeller of the torque converter which couples it to the transmission (see Equation 1).

Equation 1

$$I_{ei}\dot{N}_e = T_e - T_i$$

$N_e =$ engine speed (RPM)

$I_{ei} =$ moment of inertia of the engine and the impeller

$T_e, T_i =$ engine and impeller torque

The input-output characteristics of the torque converter can be expressed as functions of the engine speed and the turbine speed. In this example, the direction of power flow is always assumed to be from the impeller to the turbine (see Equation 2).

Equation 2

$$T_i = \frac{N_e^2}{K^2}$$

$$K = f_2\frac{N_{in}}{N_e} = \text{K-factor (capacity)}$$

$N_{in} =$ speed of turbine (torque converter output) = transmission input speed (RPM)

$$R_{TQ} = f_3\frac{N_{in}}{N_e} = \text{torque ratio}$$

The transmission model is implemented via static gear ratios, assuming small shift times (see Equation 3).

Equation 3

$$R_{TR} = f_4(gear) = \text{transmission ratio}$$

$$T_{out} = R_{TR}T_{in}$$

$$N_{in} = R_{TR}N_{out}$$

$T_{in}, T_{out} =$ transmission input and output torques

$N_{in}, N_{out} =$ transmission input and output speed (RPM)

The final drive, inertia, and a dynamically varying load constitute the vehicle dynamics (see Equation 4).

Equation 4

$$I_v\dot{N}_w = R_{fd}(T_{out} - T_{load})$$

$I_v =$ vehicle inertia

$N_w =$ wheel speed (RPM)

$R_{fd} =$ final drive ratio

$T_{load} = f_5(N_w) =$ load torque

The load torque includes both the road load and brake torque. The road load is the sum of frictional and aerodynamic losses (see Equation 5).

Equation 5

$$T_{load} = sgn(mph)(R_{load0} + R_{load2}mph^2 + T_{brake})$$

$R_{load0}, R_{load2} =$ friction and aerodynamic drag coefficients

$T_{load}, T_{brake} =$ load and brake torques

$mph =$ vehicle linear velocity

The model programs the shift points for the transmission according to the schedule shown in the figure below. For a given throttle in a given gear, there is a unique vehicle speed at which an upshift takes place. The simulation operates similarly for a downshift.

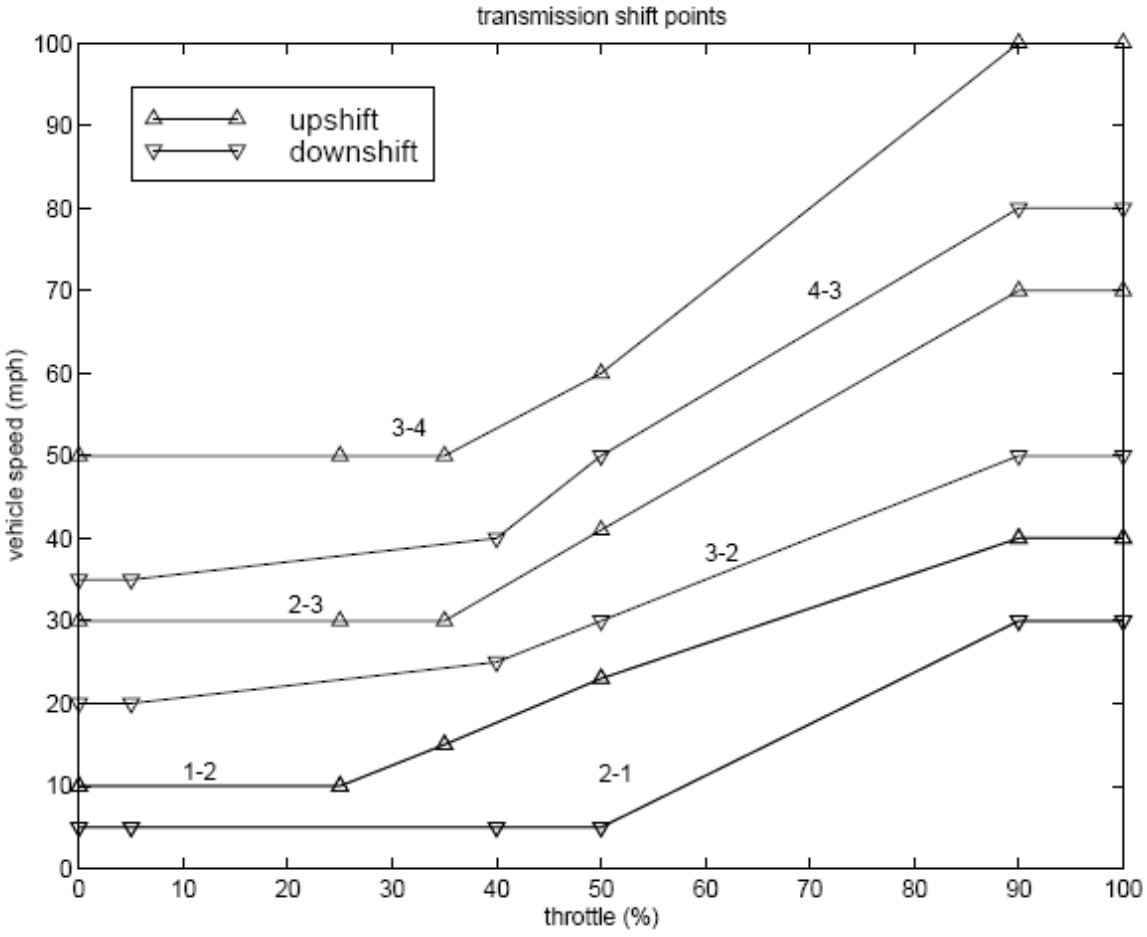


Figure 2 : Transmission shift points

2. Modeling:

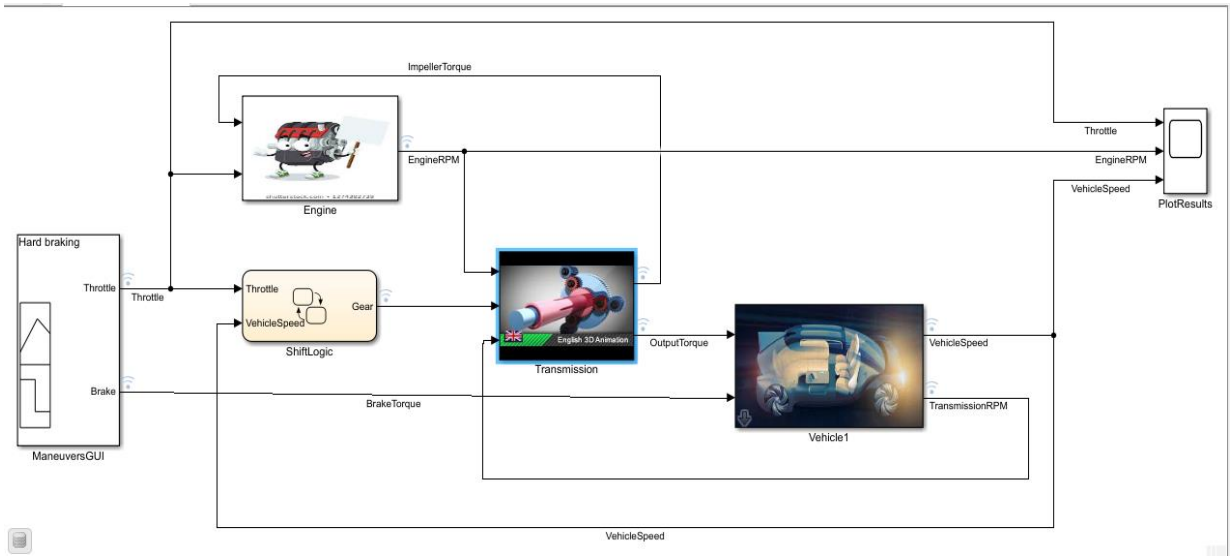
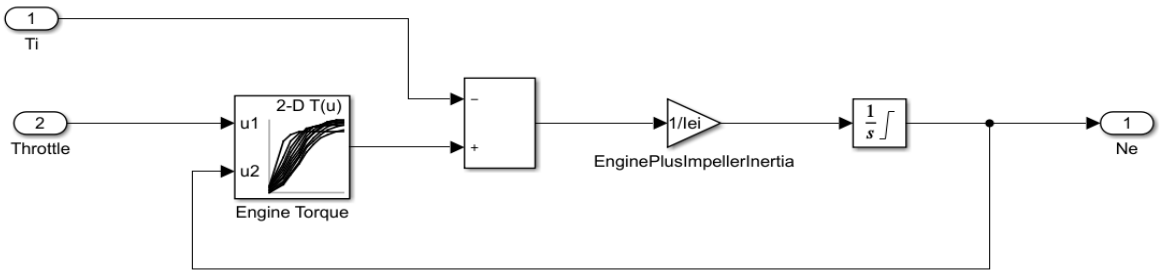


Figure 3 : Model (Automatic transmission controller)

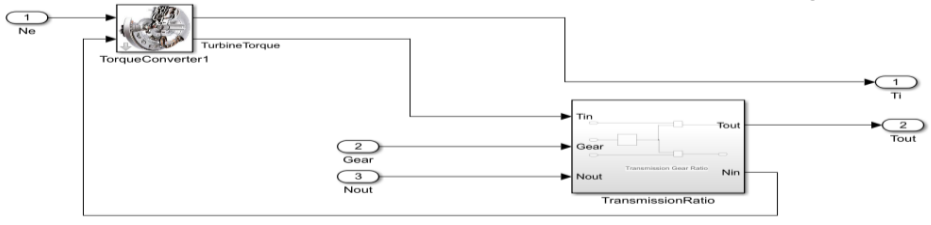
The Simulink model shown above is composed of modules which represent the engine, transmission, and the vehicle, with an additional shift logic block to control the transmission ratio. User inputs to the model are in the form of throttle (given in percent) and brake torque (given in ft-lb). The user inputs throttle and brake torques using the ManeuversGUI interface.

- The Engine subsystem consists of a two-dimensional table that interpolates engine torque versus throttle and engine speed. The figure below shows the composite Engine subsys



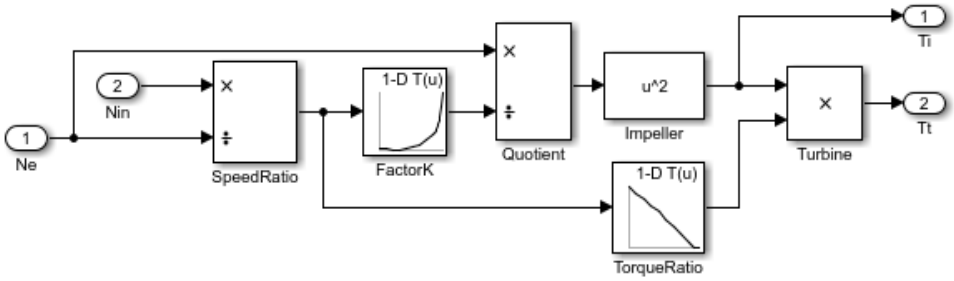
Engine

- The TorqueConverter and the TransmissionRatio blocks make up the Transmission subsystem, as shown in the figure below



Transmission

- The Torque Converter is a masked subsystem, which implements Equation 2. Below shows the implementation of the Torque Converter subsystem.

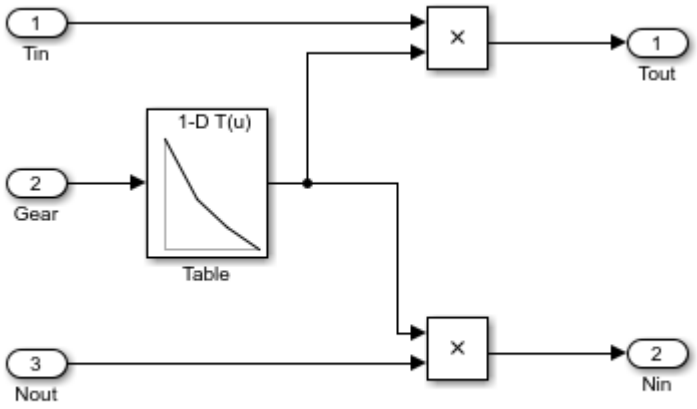


Torque Converter

- The transmission ratio block determines the ratio shown in Table 1 and computes the transmission output torque and input speed, as indicated in Equation 3. The figure that follows shows the block diagram for the subsystem that realizes this ratio in torque and speed.

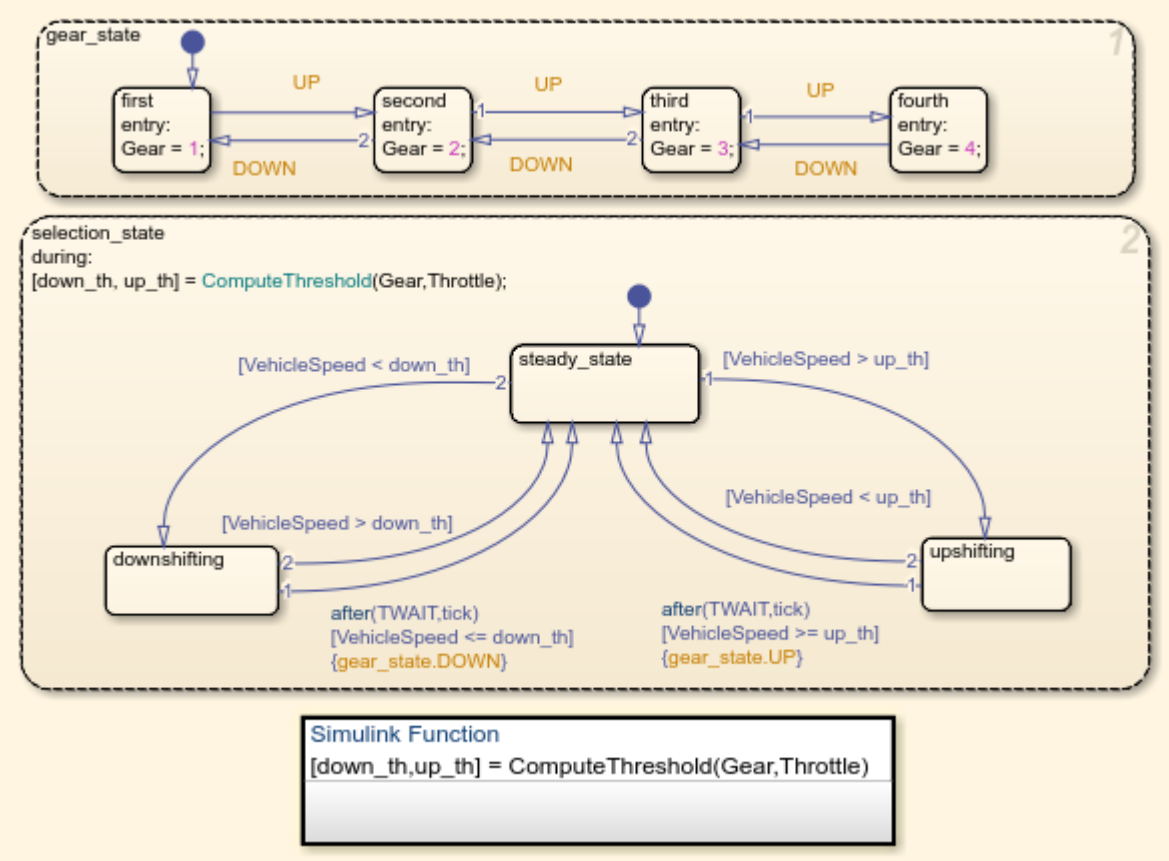
Table 1: Transmission gear ratios

gear	Rtr = Nin/Ne
1	2.393
2	1.450
3	1.000
4	0.677



Transmission Gear Ratio

- The Stateflow block labeled ShiftLogic implements gear selection for the transmission.The Model Explorer is utilized to define the inputs as throttle and vehicle speed and the output as the desired gear number. Two dashed AND states keep track of the gear state and the state of the gear selection process. The overall chart is executed as a discrete-time system, sampled every 40 milliseconds. The Stateflow diagram shown below illustrates the functionality of the block.



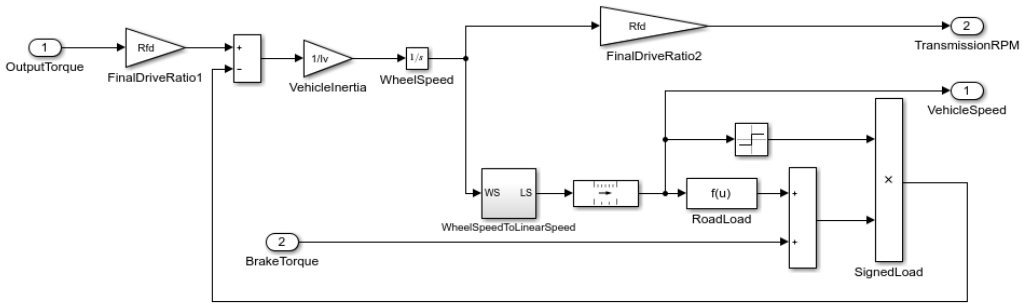
- The shift logic behaviour can be observed during simulation by enabling animation in the Stateflow debugger. The `selection_state` (always active) begins by performing the computations indicated in its `during` function. The model computes the upshift and downshift speed thresholds as a function of the instantaneous values of gear and throttle. While in steady_state, the model compares these values to the present vehicle speed to determine if a shift is required. If so, it enters one of the confirm states (`upshifting` or `downshifting`), which records the time of entry.

- If the vehicle speed no longer satisfies the shift condition, while in the confirm state, the model ignores the shift and it transitions back to `steady_state`. This prevents extraneous shifts due to noise conditions. If the shift condition remains valid for a duration of `TWAIT` ticks, the model transitions through the lower junction and, depending on the current gear, it broadcasts one of the shift events. Subsequently, the model again activates `steady_state` after a transition through one of the central junctions. The shift event, which is broadcast to the `gear_selection` state, activates a transition to the appropriate new gear.

- For example, if the vehicle is moving along in second gear with 25% throttle, the state `second` is active within `gear_state`, and `steady_state` is active in the `selection_state`. The `during` function of the latter, finds that an upshift should take

place when the vehicle exceeds 30 mph. At the moment this becomes true, the model enters the `upshifting` state. While in this state, if the vehicle speed remains above 30 mph for `TWAIT` ticks, the model satisfies the transition condition leading down to the lower right junction. This also satisfies the condition [|gear == 2|] on the transition leading from here to `steady_state`, so the model now takes the overall transition from `upshifting` to `steady_state` and broadcasts the event `UP` as a transition action. Consequently, the transition from second to third is taken in `gear_state` which completes the shift logic.

- The Vehicle subsystem uses the net torque to compute the acceleration and integrate it to compute the vehicle speed, per Equation 4 and Equation 5. The Vehicle subsystem is masked. The parameters entered in the mask menu are the final drive ratio, the polynomial coefficients for drag friction and aerodynamic drag, the wheel radius, vehicle inertia, and initial transmission output speed.



Vehicle

3. Callbacks:

Model callbacks execute at specified action points.

4. Data Inspector:

The Simulation Data Inspector visualizes and compares multiple kinds of data. Simulation Data Inspector inspects the multiple stages of workflow as shown in the figure below :

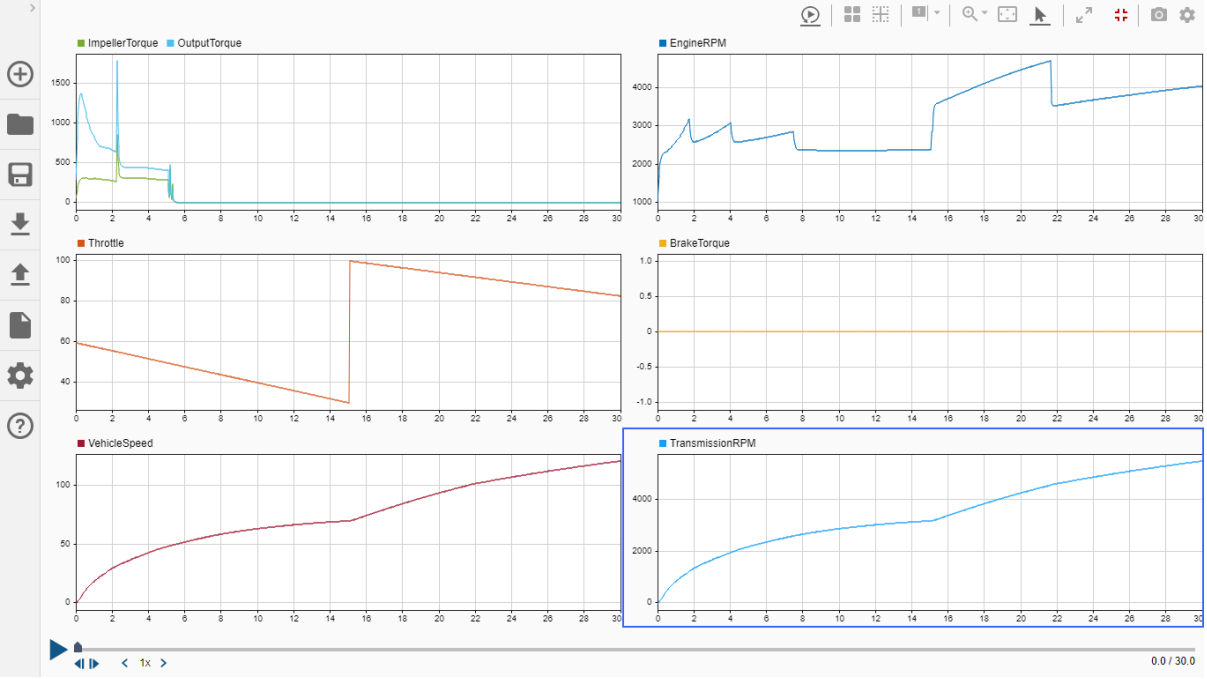


Figure 4 : Passive Maneuver

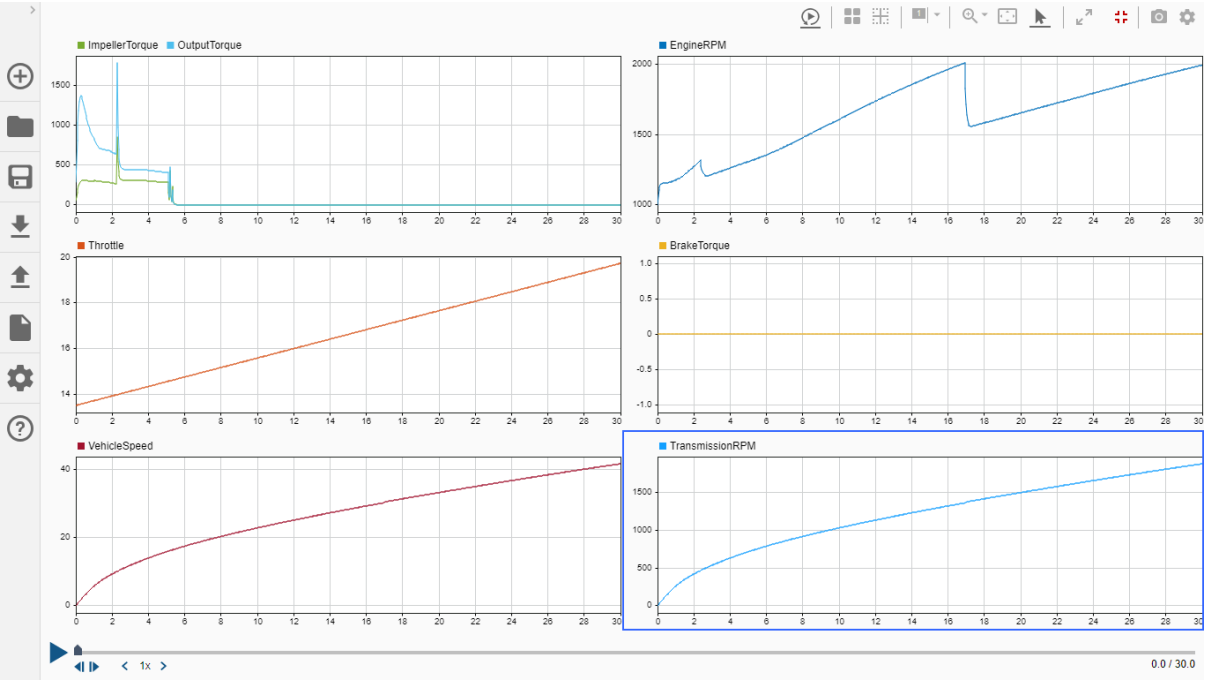


Figure 5 : Gradual Acceleration

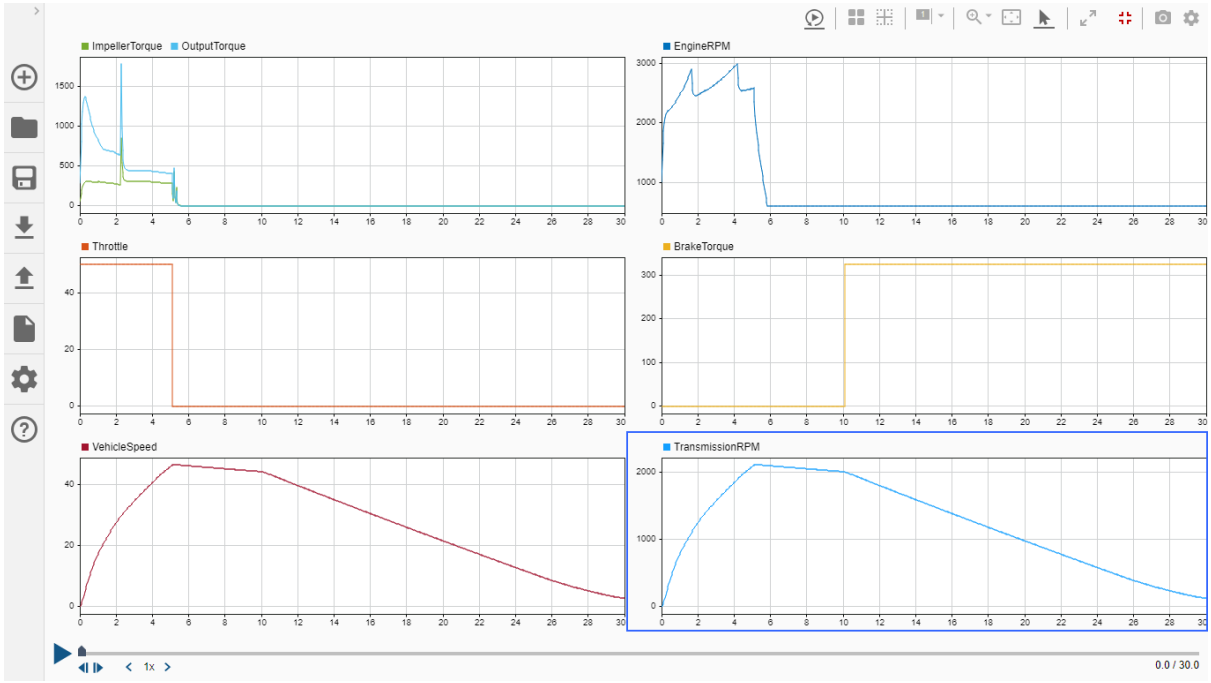


Figure 6 : Hard Braking

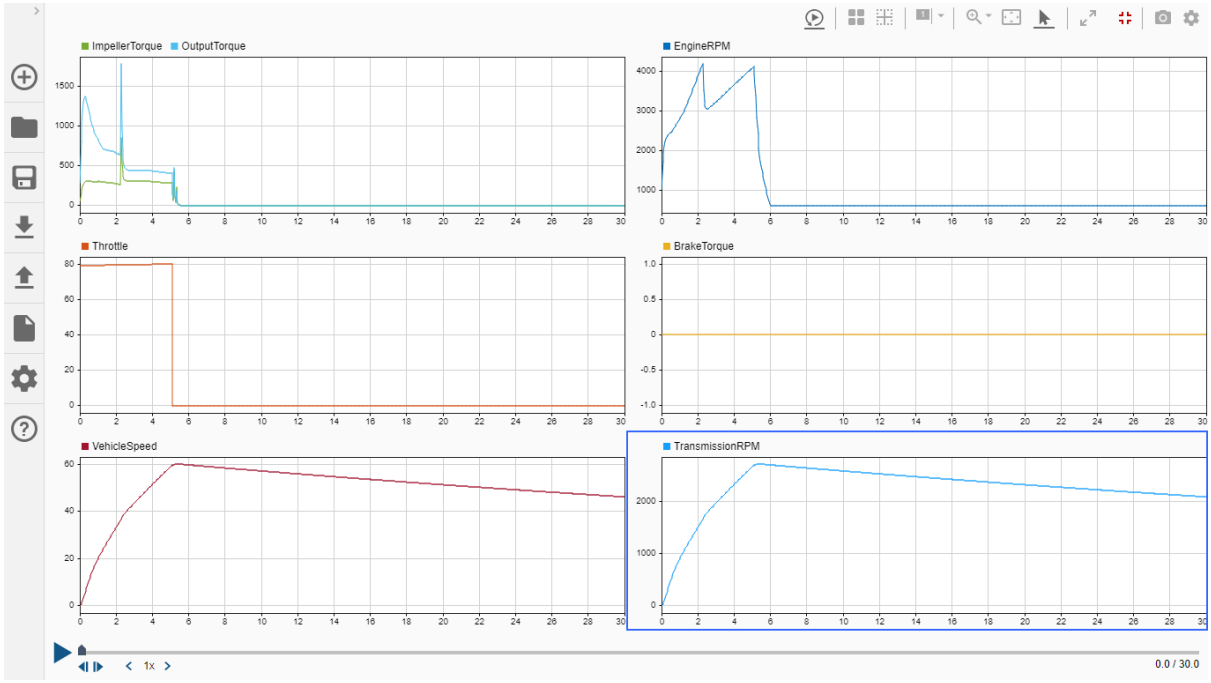


Figure 7 : Coasting

5. Solver selection strategy:

To simulate a dynamic system, you compute its states at successive time steps over a specified time span. This computation uses information provided by a model of the system. *Time steps* are time intervals when the computation happens. The size of this time interval is called *step size*. The process of computing the states of a model in this manner is known as *solving* the model. No single method of solving a model applies to all systems. Simulink provides a set of programs called *solvers*. Each solver embodies a particular approach to solving a model.

A solver applies a numerical method to solve the set of ordinary differential equations that represent the model. Through this computation, it determines the time of the next simulation step. In the process of solving this initial value problem, the solver also satisfies the accuracy requirements that you specify.

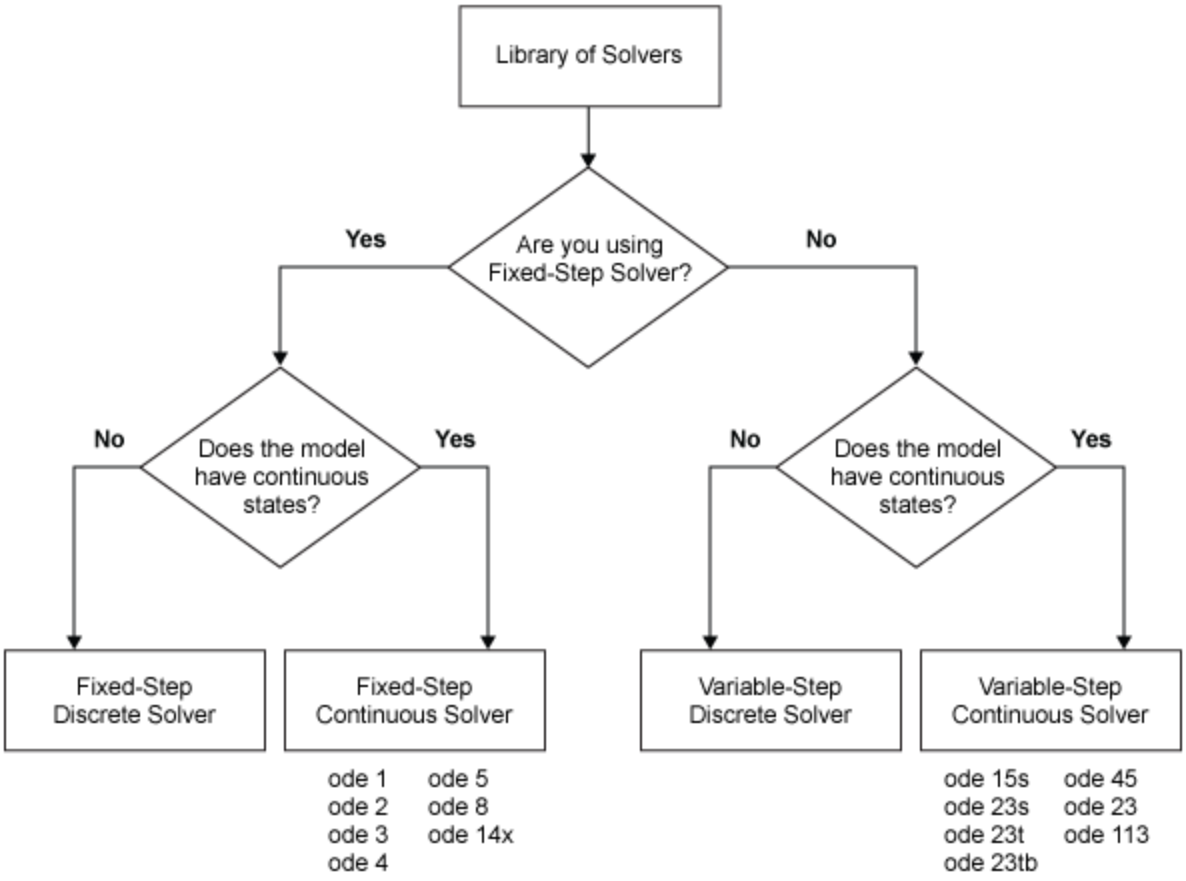
Mathematicians have developed a wide variety of numerical integration techniques for solving the ordinary differential equations (ODEs) that represent the continuous states of dynamic systems. An extensive set of fixed-step and variable-step continuous solvers are provided, each of which implements a specific ODE solution method. Select solvers in the **Solver** pane of model configuration parameters.

All solvers provided by MATLAB and Simulink follow a similar naming convention: `ode`, followed by two or three numerals indicating the orders of the solver. Some solvers can solve stiff differential equations and the methods used by them are expressed by the `s`, `t`, or `tb` suffixes.

Solver Selection Criteria

The appropriate solver for simulating a model depends on these characteristics:

- System dynamics
- Solution stability
- Computation speed
- Solver robustness
- This chart provides a broad classification of solvers in the Simulink library.



- Solver selected for the system is fixed step ode5.

6. MATLAB function block:

MATLAB Function blocks enable you to define custom functionality in Simulink models by using the MATLAB language. They are the easiest way to bring MATLAB code into Simulink. MATLAB Function blocks support C/C++ code generation from Simulink Code and Embedded Coder.

Use these blocks specifically when:

- You have an existing MATLAB function that models custom functionality, or it would be easy for you to create such a function.
- Your model requires custom functionality that is not or cannot be captured in the Simulink graphical language.
- You find it easier to model custom functionality by using a MATLAB function than by using a Simulink block diagram.
- The custom functionality that you want to model does not include continuous or discrete dynamic states. To model dynamic states, use S-functions.

7. Look-up table:

A *lookup table* is an array of data that maps input values to output values, thereby approximating a mathematical function. Given a set of input values, a lookup operation retrieves the corresponding output values from the table. If the lookup table does not explicitly define the input values, Simulink can estimate an output value using interpolation, extrapolation, or rounding, where:

- An interpolation is a process for estimating values that lie between known data points.
- An extrapolation is a process for estimating values that lie beyond the range of known data points.
- A rounding is a process for approximating a value by altering its digits according to a known rule. A lookup table block uses an array of data to map input values to output values, approximating a mathematical function. Given input values, Simulink performs a “lookup” operation to retrieve the corresponding output values from the table. If the lookup table does not define the input values, the block estimates the output values based on nearby table values.
- This this model 2-D and 1-D lookup tables are used in engine, transmission and torque converter subsystems.

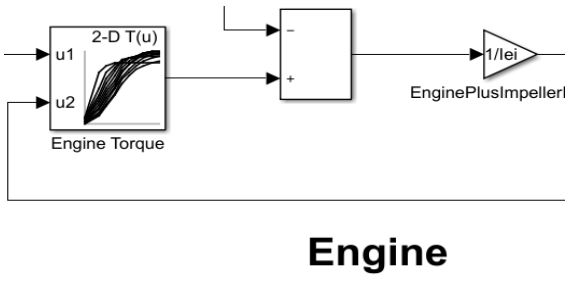


Figure 8 : 2-D lookup table

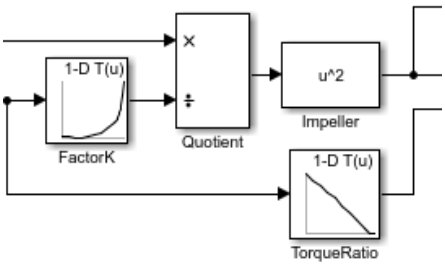


Figure 9 : 1-D lookup table

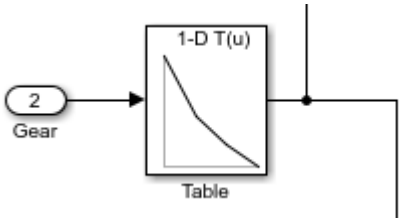


Figure 10 : 1-D lookup table

8. Signal Builder:

User inputs to the model are in the form of throttle (given in percent) and brake torque (given in ft-lb). The user inputs throttle and brake torques using the ManeuversGUI interface.

Test signals generated using signal builder are as shown below:

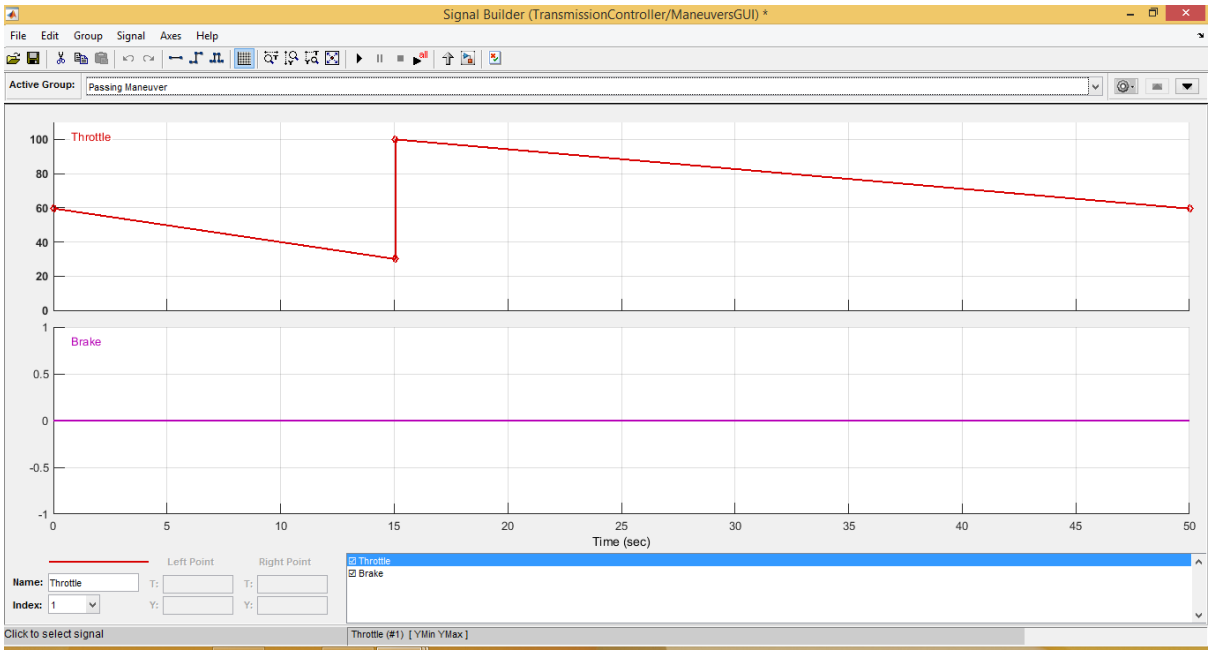


Figure 11 : Test signal (Passive Maneuver)

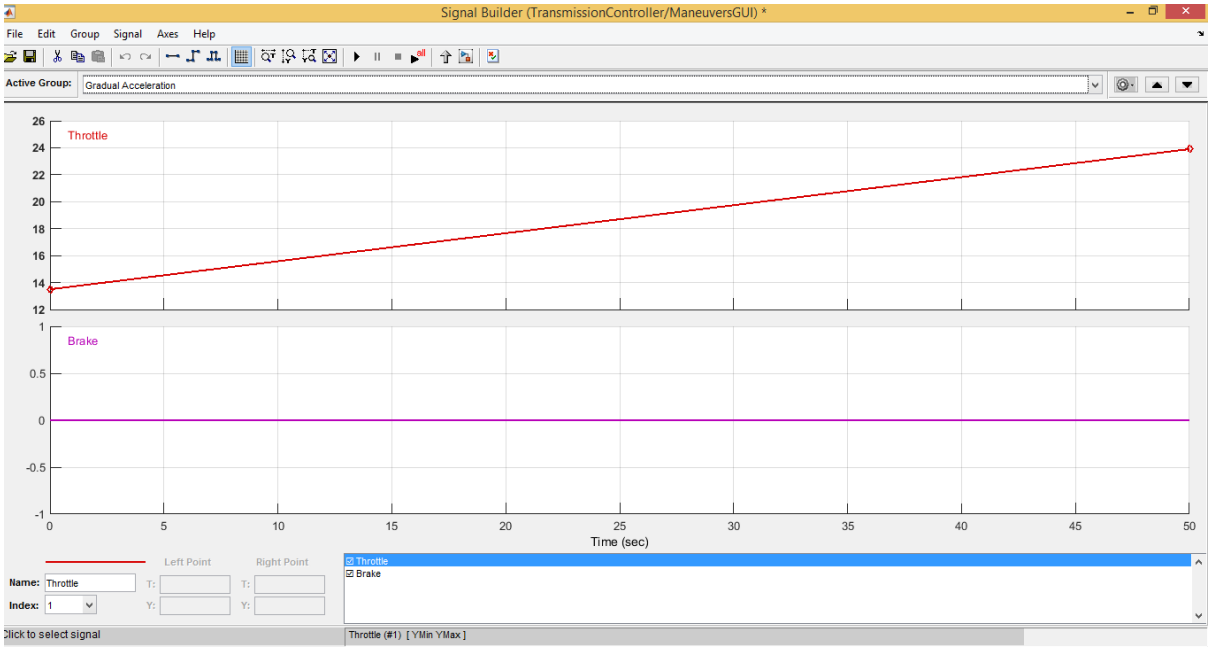


Figure 12 : Test signal (Gradual Acceleration)

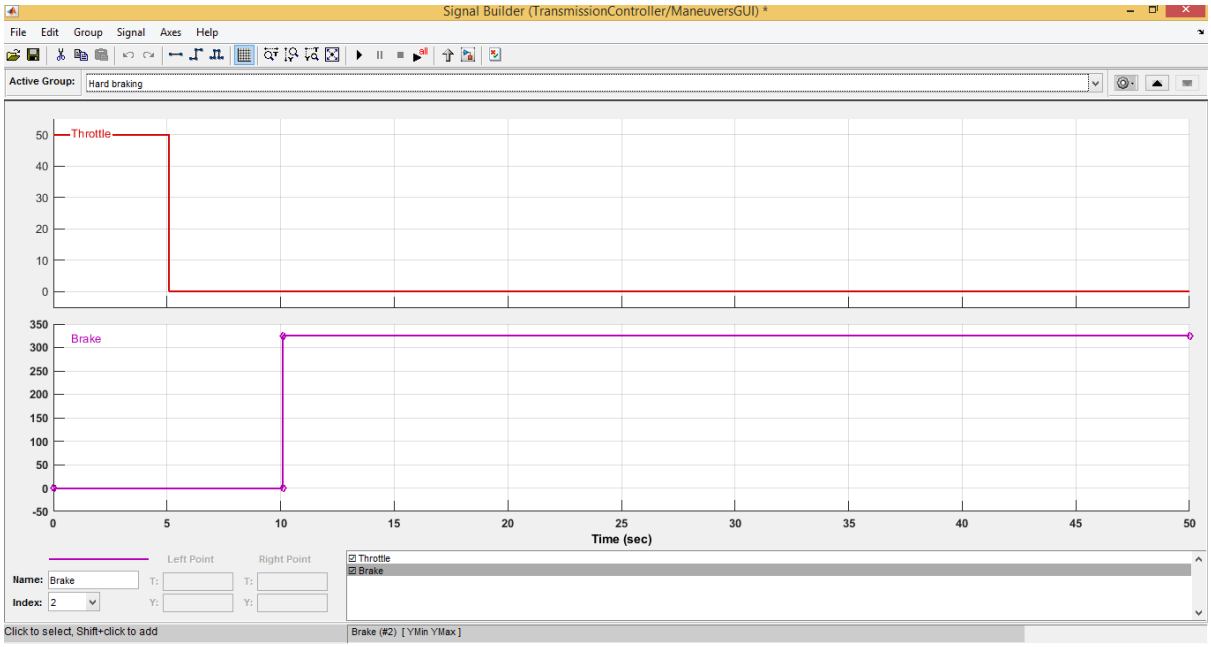


Figure 13 : Test signal (Hard Braking)

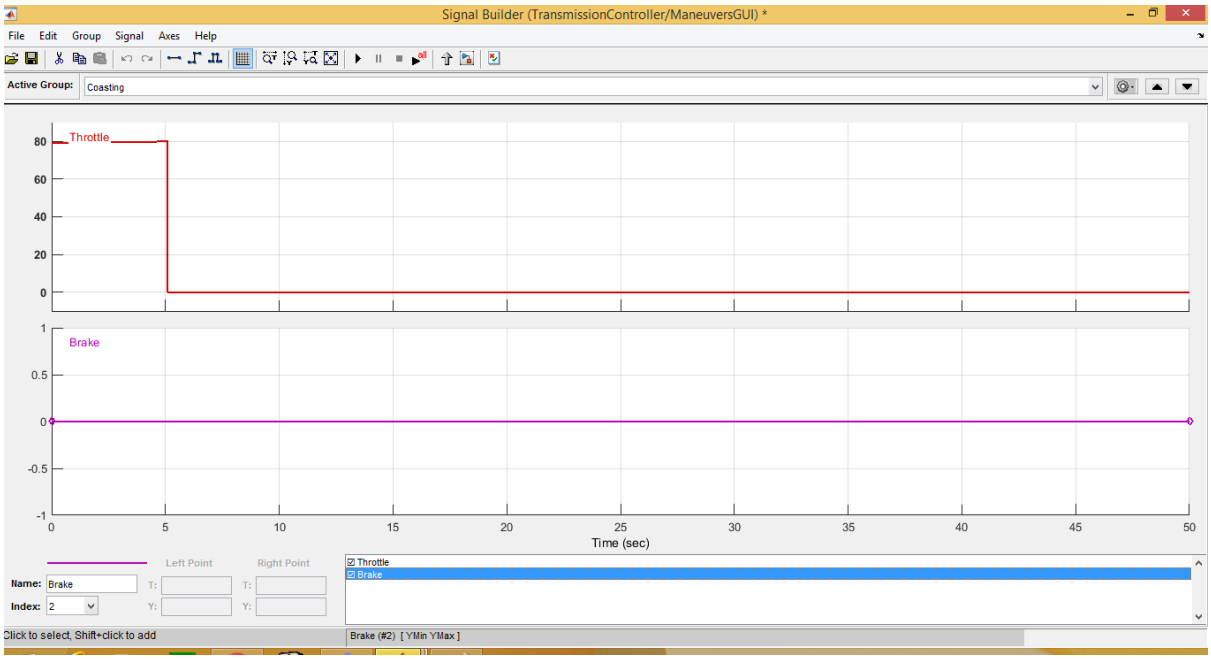


Figure 14 : Test signal (Coasting)

9. Outputs:

Test conditions:

1. Passive Maneuver:

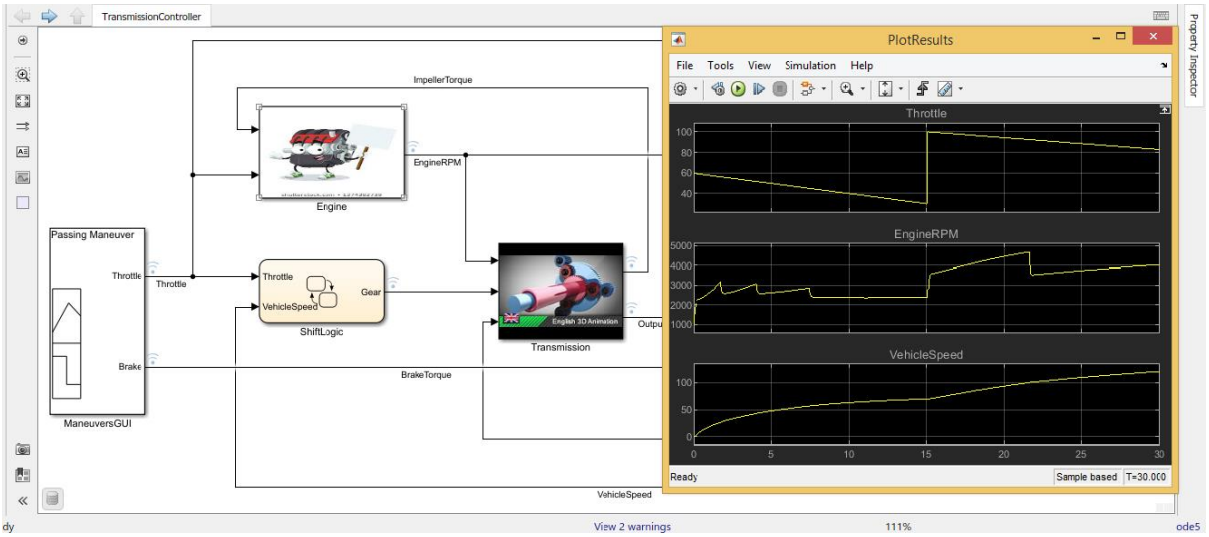


Figure 15 : Passive Maneuver

2. Gradual acceleration:

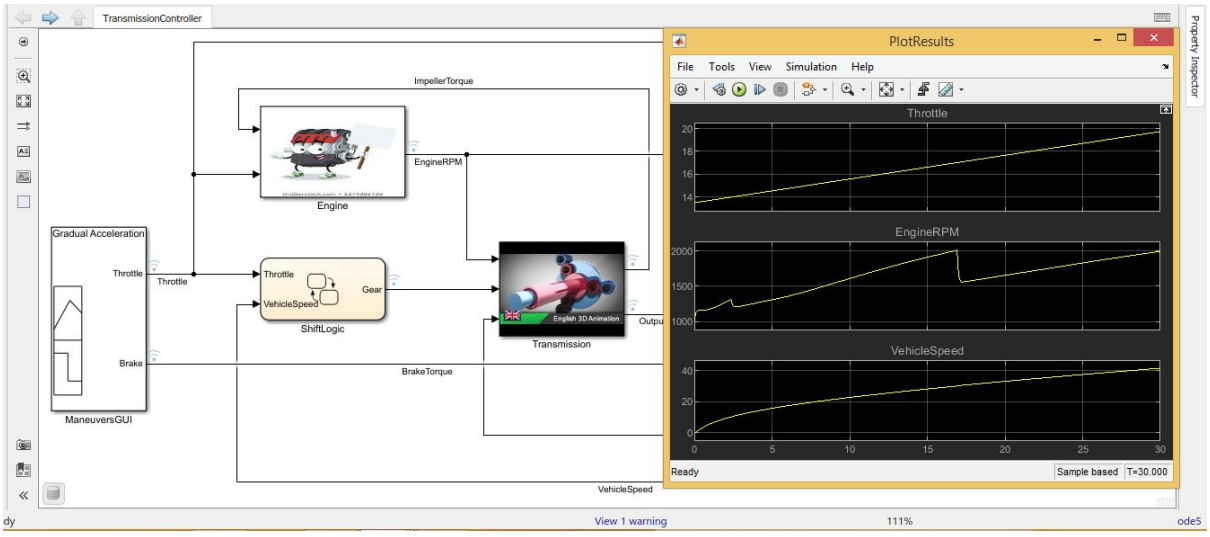


Figure 16 : Gradual Acceleration

3. Hard braking:

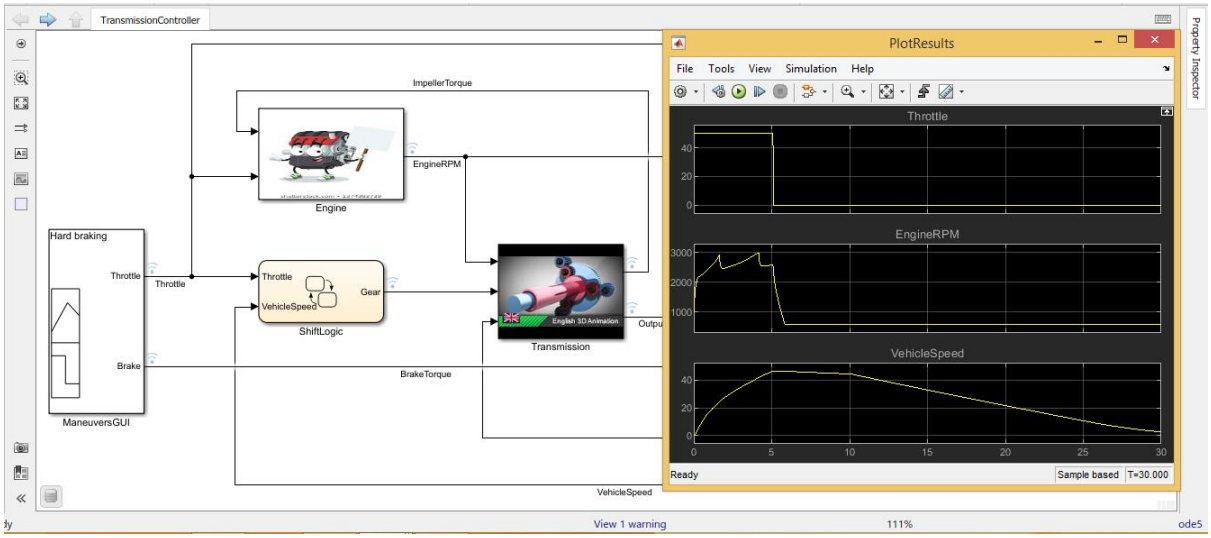


Figure 17 : Hard Braking

4. Coasting:

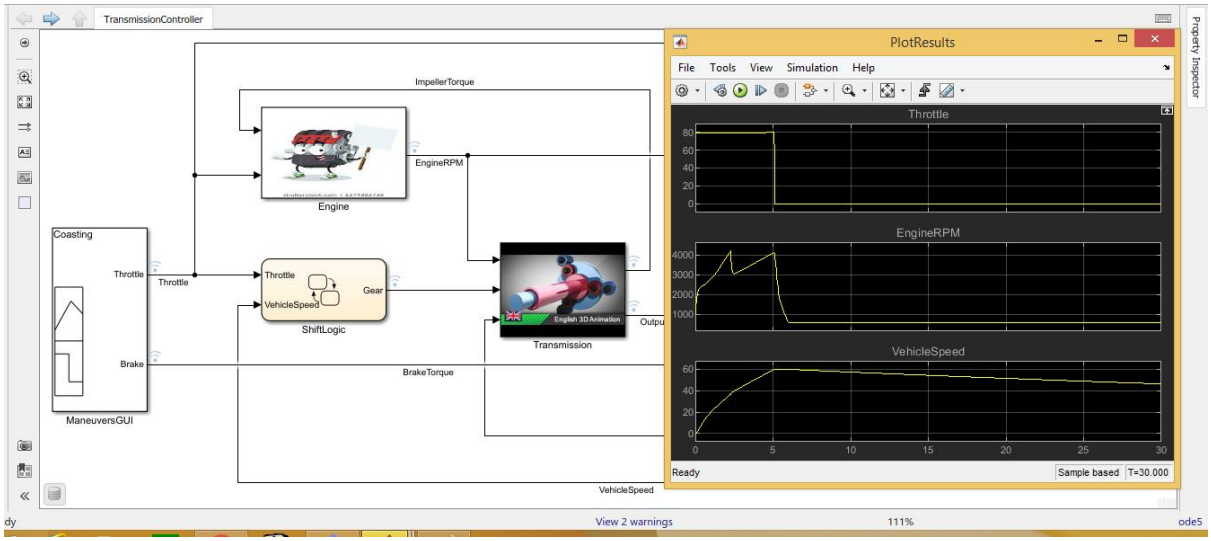


Figure 18 : Coasting

References:

https://www.mathworks.com/help/simulink/slref/modeling-an-automatic-transmission-controller.html