Modelling an Automatic Transmission Controller

Name: Gorthy Venkata Sreeram Kumar
Id: 214865

Contents

Name: Gorthy Venkata Sreeram Kumar 1
1. Introduction 3
2. Analysis and Physics 3
3. Equations and Derivations 4
4. Modelling 6
5. Callbacks 6
6. Data Inspector 7
7. Solver Selection Strategy 8
8. MATLAB Function Block 9
9. Look-up Table 10
10. Signal Builder 11

1. Introduction

This Simulink design shows how to model an automotive drivetrain with Simulink. Stateflow enhances the Simulink model with its representation of the transmission control logic. Simulink provides a powerful environment for the modeling and simulation of dynamic systems and processes. In many systems, though, supervisory functions like changing modes or invoking new gain schedules must respond to events that may occur and conditions that develop over time. As a result, the environment requires a language capable of managing these multiple modes and developing conditions. In the following example, Stateflow shows its strength in this capacity by performing the function of gear selection in an automatic transmission. This function is combined with the drivetrain dynamics in a natural and intuitive manner by incorporating a Stateflow block in the Simulink block diagram.

2. Analysis and Physics

Figure 1 shows the power flow in a typical automotive drivetrain. Nonlinear ordinary differential equations model the engine, four-speed automatic transmission, and vehicle. The model discussed in this example directly implements the blocks from Figure 1 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.

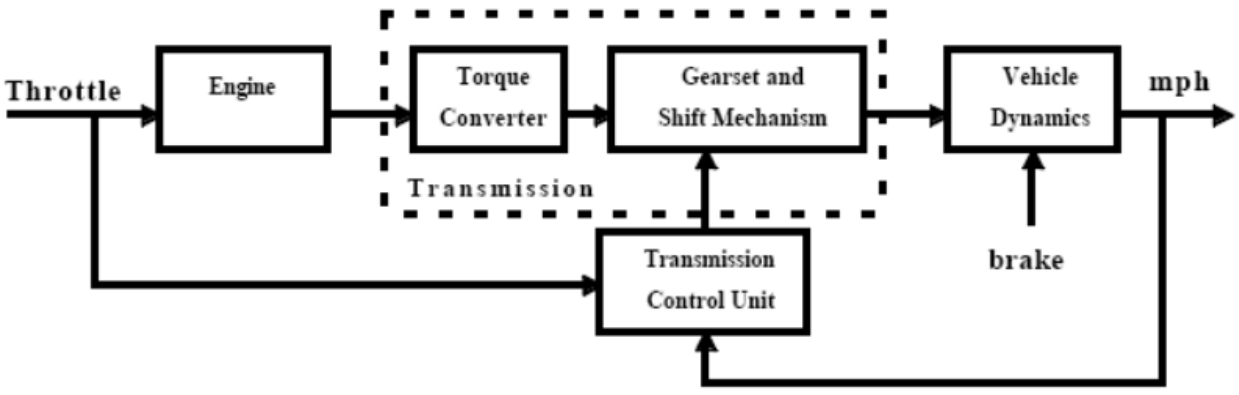

Figure 1 - Generic block diagram for a drivetrain system.

3. Equations and Derivations

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

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

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

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

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

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

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

Equation 4

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

$$I_v = \text{ vehicle inertia}$$

$$N_w = \text{ wheel speed (RPM)}$$

$$R_{fd} = \text{ final drive ratio}$$

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

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

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

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

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

$$mph = \text{ vehicle linear velocity}$$

4. Modelling

The top-level diagram of the model is shown in Figure 2. To run the simulation, press the Play button on the toolbar in the model window. Note that the model logs relevant data to MATLAB Workspace in a data structure called sldemo_autotrans_output. Logged signals have a blue indicator (see Figure 2). After you run the simulation, you can view the components of the data structure by typing sldemo_autotrans_output in MATLAB Command Window. Also note that the units appear on the subsystem icons and signal lines.

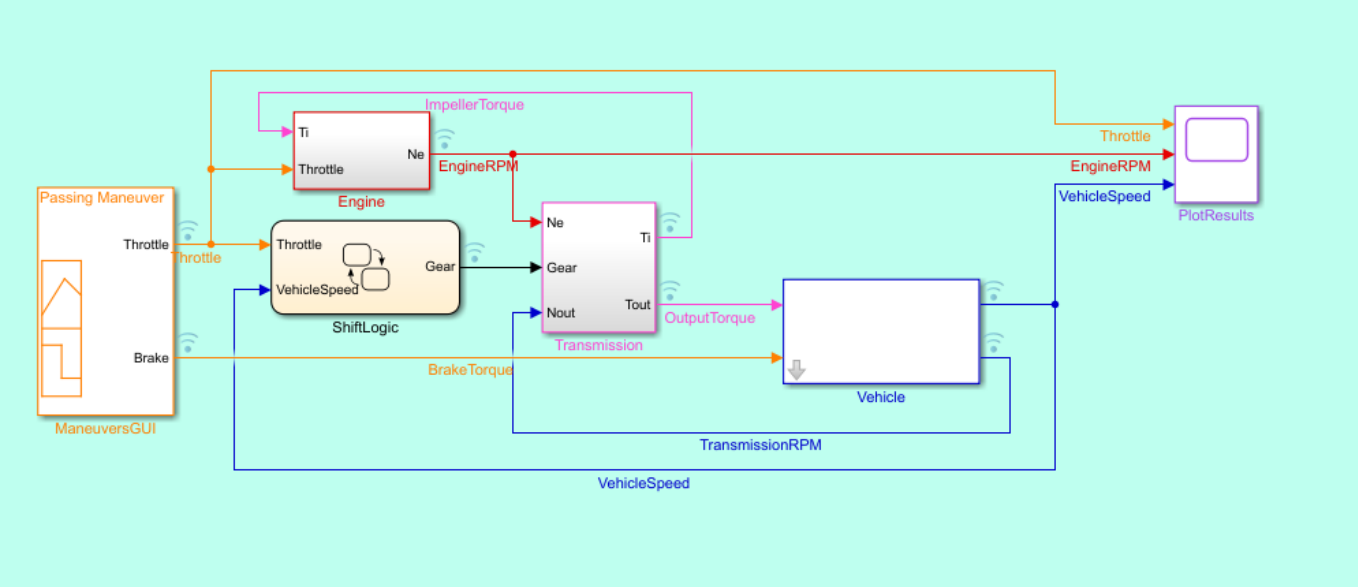

Figure 2 – Model Diagram

5. Callbacks

Model callbacks execute at specified action points, for example after you load or save the model.

You can set most of the same callbacks for libraries. Only the callbacks that can execute for a library are available to set for a library.

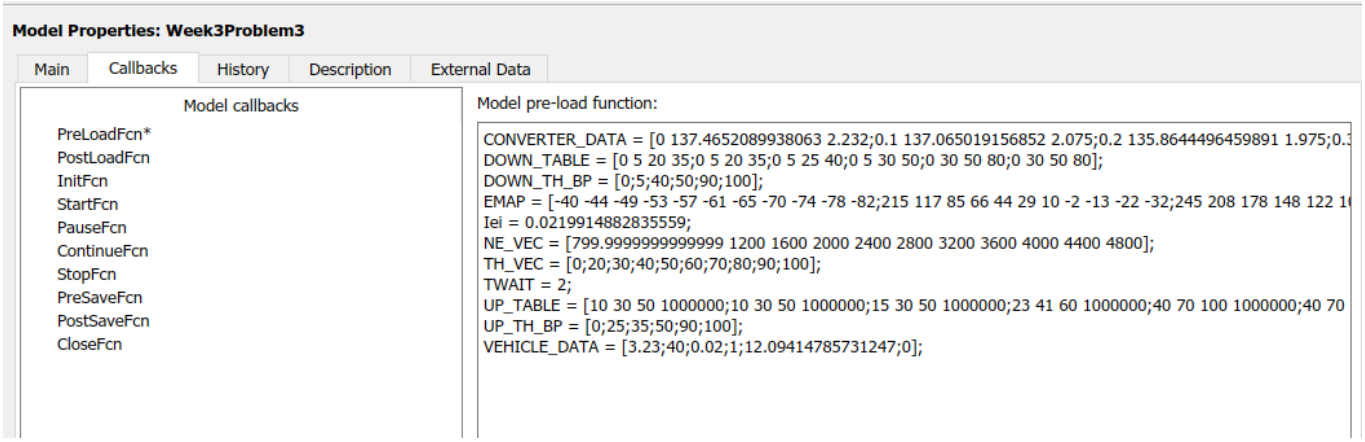

Figure 2 - Callbacks

6. Data Inspector

Inspect and compare data and simulation results to validate and iterate model designs.

The Simulation Data Inspector visualizes and compares multiple kinds of data.
Using the Simulation Data Inspector, you can inspect and compare time series data at multiple stages of your workflow.
This model workflow the Simulation Data Inspector supports all stages of the design cycle:

1. View Data in the Simulation Data Inspector.

   Run a simulation in a model configured to log data to the Simulation Data Inspector, or import data from the workspace or a MAT-file. You can view and verify model input data or inspect logged simulation data while iteratively modifying your model diagram, parameter values, or model configuration.

2. Inspect Simulation Data.

   Plot signals on multiple subplots, zoom in and out on specified plot axes, and use data cursors to understand and evaluate the data. Create Plots Using the Simulation Data Inspector to tell your story.

3. Compare Simulation Data

   Compare individual signals or simulation runs and analyse your comparison results with relative, absolute, and time tolerances. The compare tools in the Simulation Data Inspector facilitate iterative design and allow you to highlight signals that do not meet your tolerance requirements. For more information about the comparison operation, see How the Simulation Data Inspector Compares Data.

4. Save and Share Simulation Data Inspector Data and Views.

   Share your findings with others by saving Simulation Data Inspector data and views.

You can also harness the capabilities of the Simulation Data Inspector from the command line.

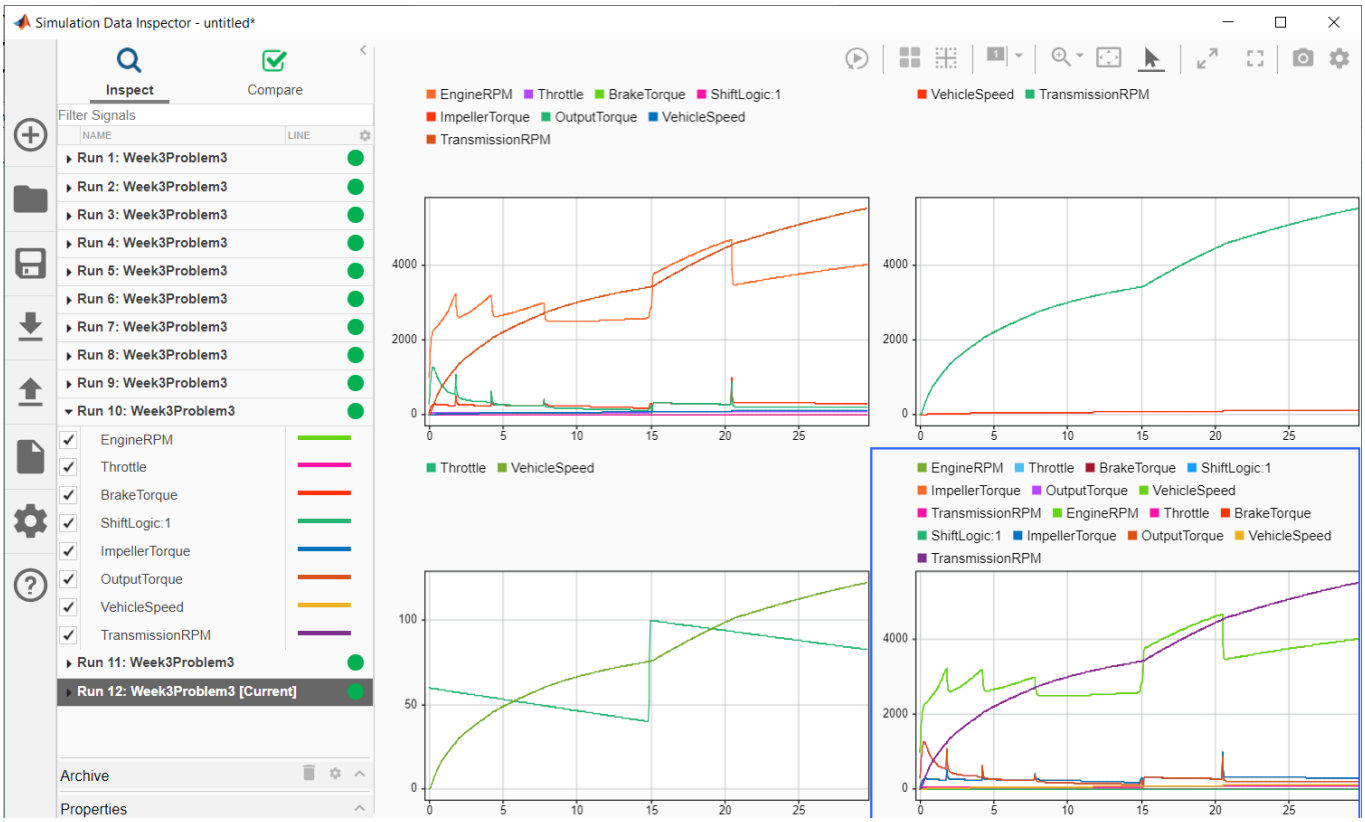

Figure 4 – Data Inspector

7. Solver Selection Strategy

Simulink selects a solver for simulating the model, and it selects an auto solver. Auto solver chooses a suitable solver and sets the maximum step size of the simulation.

For this Simulink model Fixed-step type is used. *Fixed-step solvers* solve the model at regular time intervals from the beginning to the end of the simulation. The size of the interval is known as the step size. You can

specify the step size or let the solver choose the step size. Generally, a smaller the step size increases the accuracy of the results but also increases the time required to simulate the system.

And the Solver used is ode45 (Dormand-Prince).
ODE45 Uses the fifth-order Dormand-Prince formula to compute the model state at the next time step as an explicit function of the current value of the state and the state derivatives approximated at intermediate points.

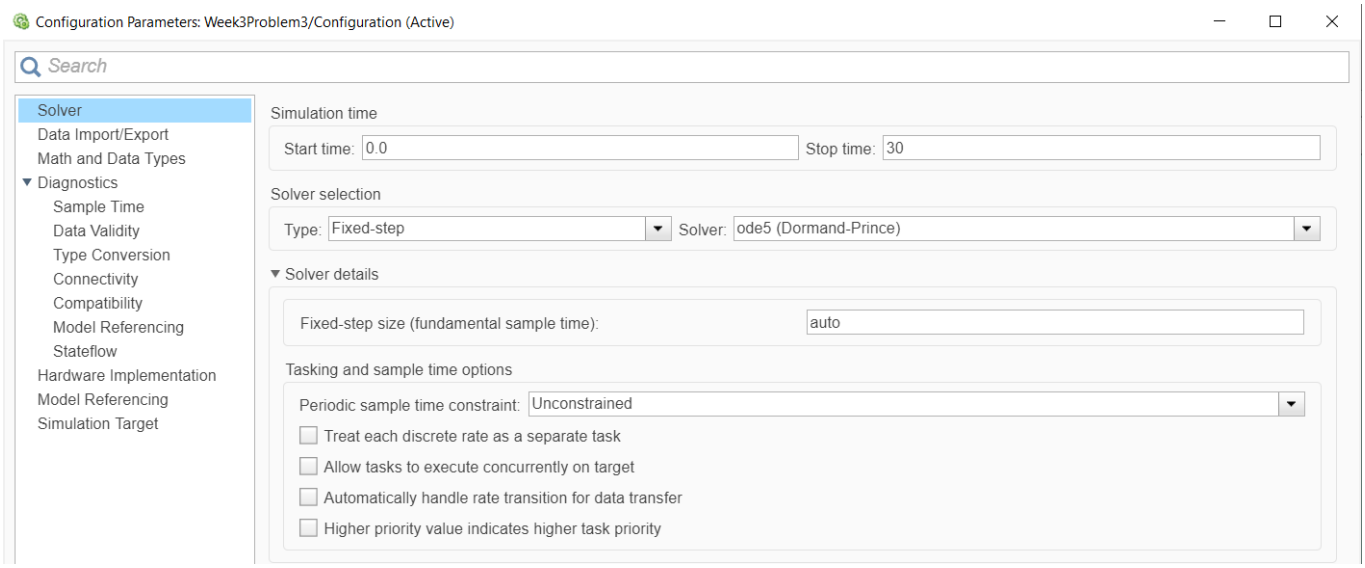

Figure 5 – Solver selection strategy

8. MATLAB Function Block

A MATLAB Function blocks enable you to define custom functionality in Simulink models by using the MATLAB language.

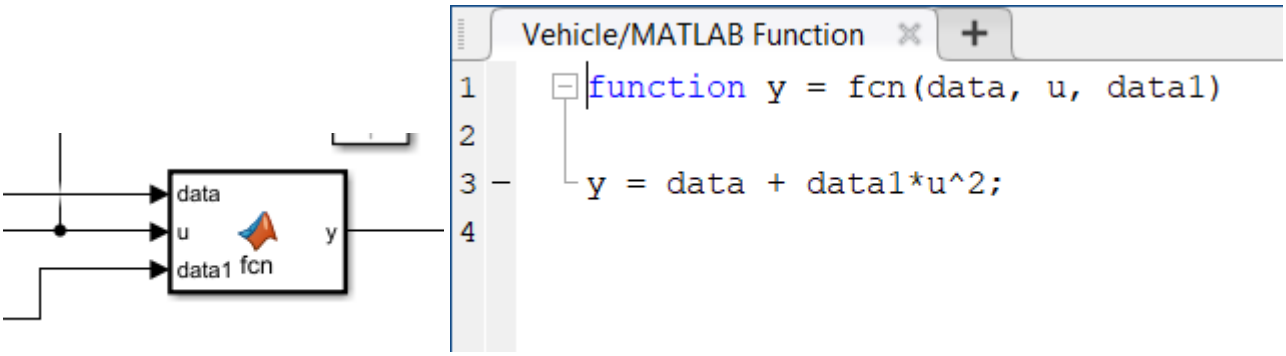

Figure 6 – MATLAB Function Block

MATLAB Function block to implement MATLAB functions to Simulink models to deploy code and embed code in processors. Using MATLAB Function block, you can generate readable, efficient, and compact C/C++ code for deployment to desktop and embedded applications.

In this model we used a MATLAB function block for defined equation. For this block there were three inputs in our model with single output. With the help of input value we calculated the output.

9. 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.

In this Simulink model there three look-up tables. Look-up tables are used in engine and transmission.

An interpolation is a process for estimating values that lie between known data points.

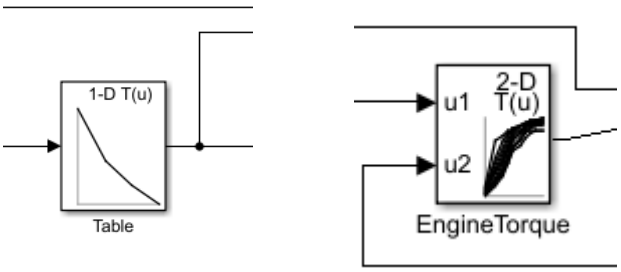

Figure 7 – Look-up Table

10. Signal Builder

Signal builder create and generate interchangeable groups of signals whose waveforms are piecewise linear.
You can quickly switch the signal groups into and out of a model to facilitate testing. In the Signal Builder window, create signals and define the output waveforms.

A Signal Builder also allows you to construct different kinds of test cases so that you are running all of them to study your model's behaviour. If you click on the "Start simulation" icon inside the Signal Builder, you will run the simulation with this signal as an output of the signal builder.You can modify the signal by clicking on one line or one point and moving it around. As you can see, this only modifies the location of the straight lines delimited by points or the location of the points.

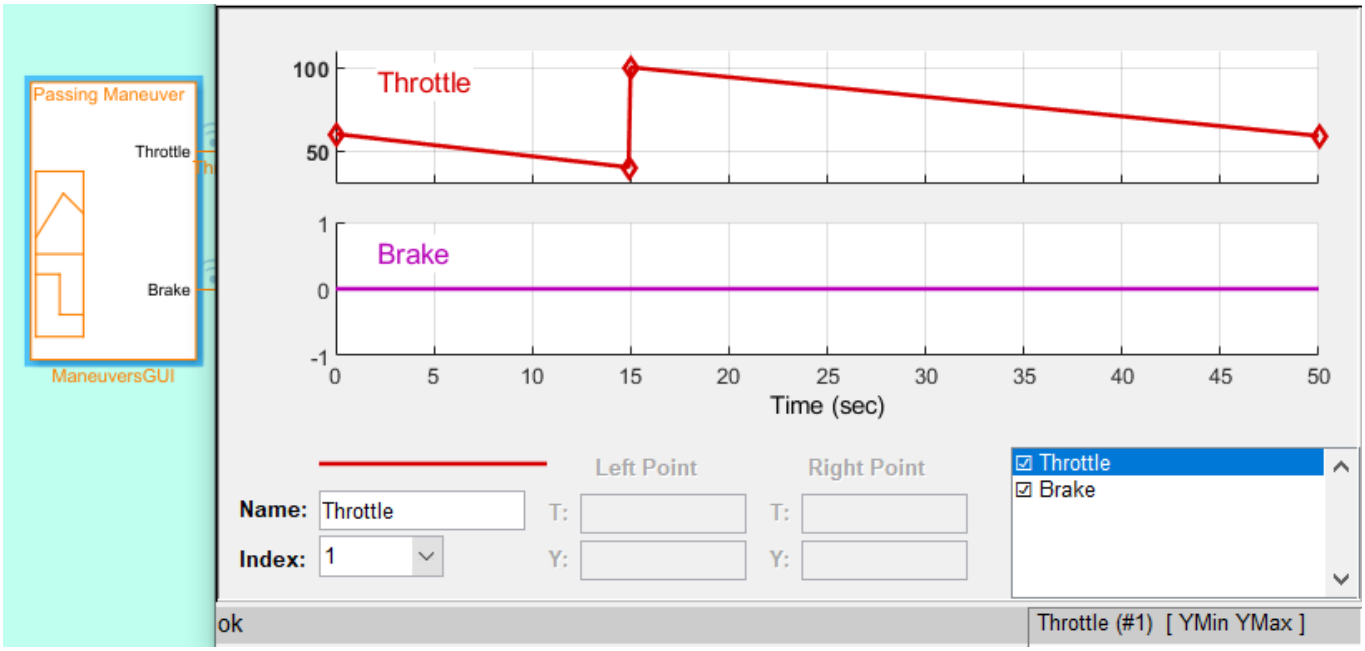

Figure 8 – Signal Builder