Design a complex automotive system

Problem Statements: Design a complex automotive system and implement required skills.

The model demonstrated : Anti-Lock Braking System

Requirements to be implemented:

1. Callbacks
2. Data Inspector
3. Solver selection strategy
4. MATLAB function block
5. Look-up table
6. Signal Builder to generate test signals. Demonstrate how your system is performing under various test conditions

Analysis and Equations:

The wheel rotates with an initial angular speed that corresponds to the vehicle speed before the brakes are applied.

$$\omega_v = \frac{V}{R} \text{ (equals the wheel angular speed if there is no slip)}$$

$$\omega_v = \frac{V_v}{R_r}$$

$$slip = 1 - \frac{\omega_w}{\omega_v}$$

$\omega_v =$ vehicle speed divided by wheel radius

$V_v =$ vehicle linear velocity

$R_r =$ wheel radius

$\omega_w =$ wheel angular velocity

From these expressions, we see that slip is zero when wheel speed and vehicle speed are equal, and slip equals one when the wheel is locked. A desirable slip value is 0.2, which means that the number of wheel revolutions equals 0.8 times the number of revolutions under non-braking conditions with the same vehicle velocity.

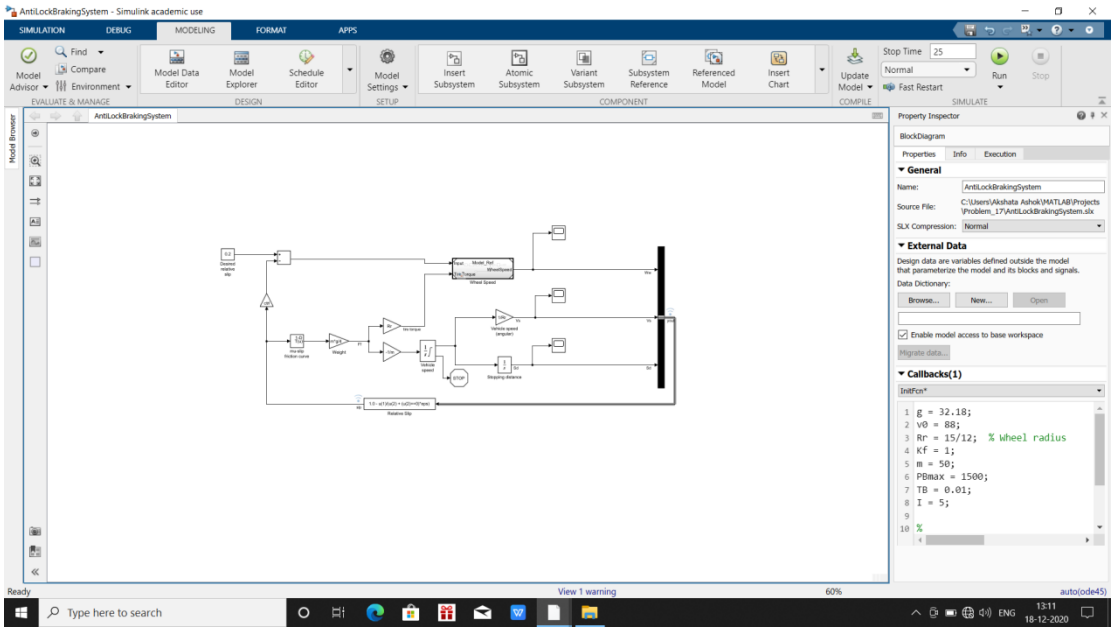
Modelling

- The friction coefficient between the tire and the road surface, mu, is an empirical function of slip, known as the mu-slip curve.
- We created mu-slip curves by passing MATLAB variables into the block diagram using a Simulink lookup table.
- The model multiplies the friction coefficient, mu, by the weight on the wheel, W, to yield the frictional force, Ff, acting on the circumference of the tire.
- Ff is divided by the vehicle mass to produce the vehicle deceleration, which the model integrates to obtain vehicle velocity.
- In this model, we used an ideal anti-lock braking controller, that uses 'bang-bang' control based upon the error between actual slip and desired slip.
- We set the desired slip to the value of slip at which the mu-slip curve reaches a peak value, this being the optimum value for minimum braking distance

Implementation:

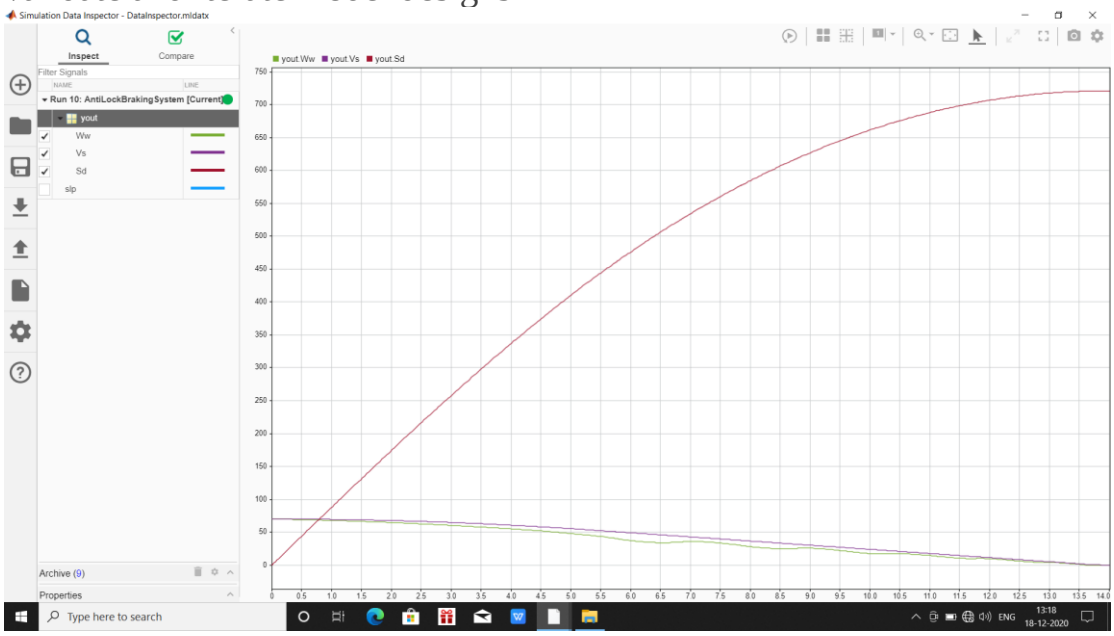
1. Callbacks

To implement this, we can eliminate the menthod of defining variable in scripts .m file. The perks of using callback function, there is no need to define variable everytime we run the model.
Callback used is : InitFcn



2. Data Inspector

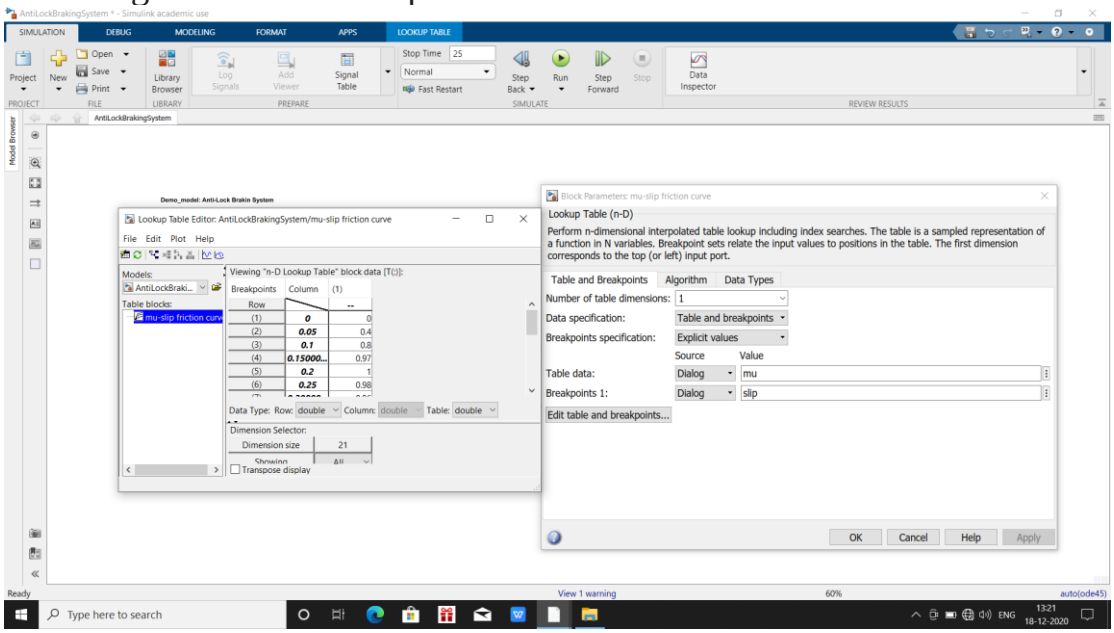
It is implemented to inspect and comapare data and simulation results to validate and iterate model designs



3. 1D Look up table

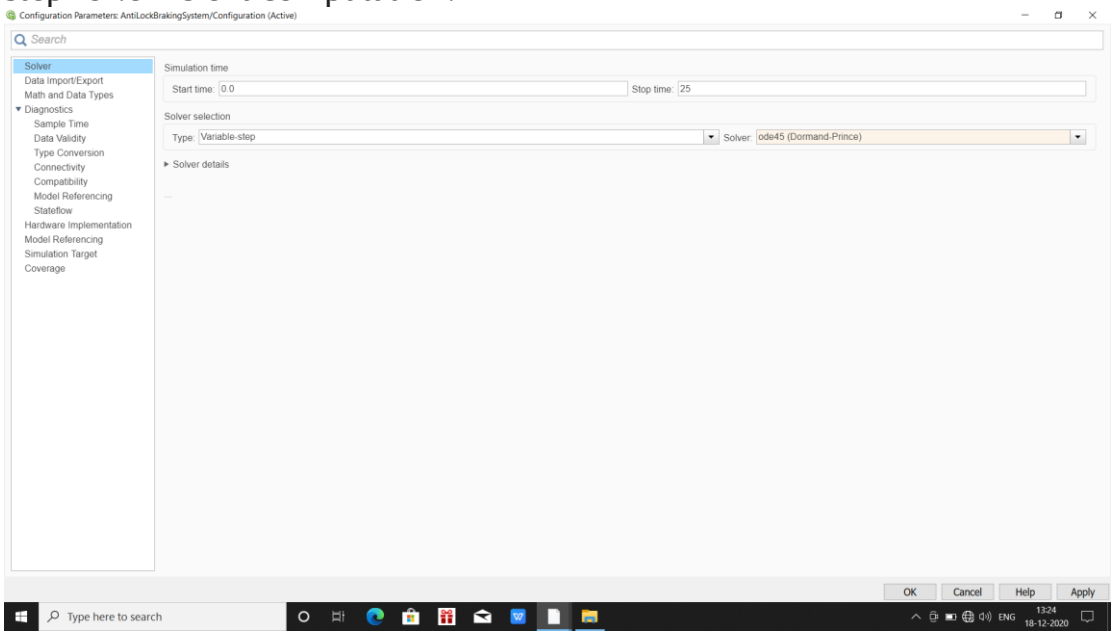
A **lookup table** is an array of data that maps input values to output values, thereby approximating a mathematical function. Given a set of input values, a **lookup** operation retrieves the corresponding output values from the **table**.

Here, the mu slip array values has been assigned to look up table retrieving from base workspace.



4. Solver

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.
Here, **ode45:** implements a Runge-Kutta **method** with a variable time step for.efficient computation.



5. .Signal builder

Create and generate interchangeable groups of signals whose waveforms are piecewise linea