**Modeling an Anti-Lock Braking System**

This example shows how to model a simple model for an Anti-Lock Braking System (ABS). It simulates the dynamic behavior of a vehicle under hard braking conditions. The model represents a single wheel, which may be replicated a number of times to create a model for a multi-wheel vehicle.

**Analysis and Physics**

The wheel rotates with an initial angular speed that corresponds to the vehicle speed before the brakes are applied. We used separate integrators to compute wheel angular speed and vehicle speed. We use two speeds to calculate slip, which is determined by Equation 1. Note that we introduce vehicle speed expressed as an angular velocity.

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

**Equation 1**

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

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

$$\omega_v = \mbox{ vehicle speed divided by wheel radius}$$

$$ V_v = \mbox{ vehicle linear velocity}$$

$$ R_r = \mbox{ wheel radius}$$

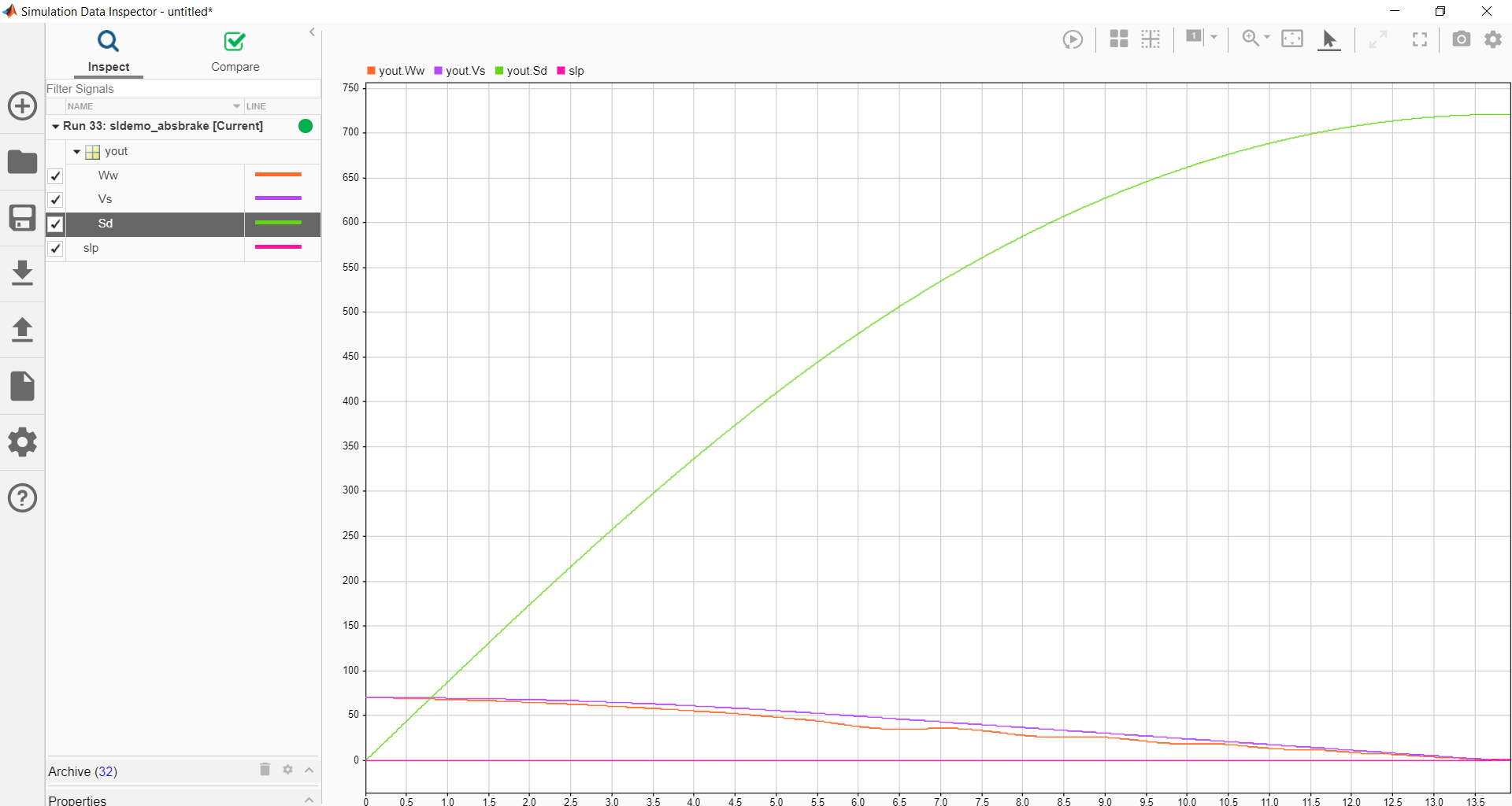
$$ \omega_w = \mbox{ wheel angular velocity}$$

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. This maximizes the adhesion between the tire and road and minimizes the stopping distance with the available friction.

**Callbacks**

* PreloadFn – For loading the initial data
* StopFcn- For stoping the function
* CloseFcn- For closing the plots which arer still open

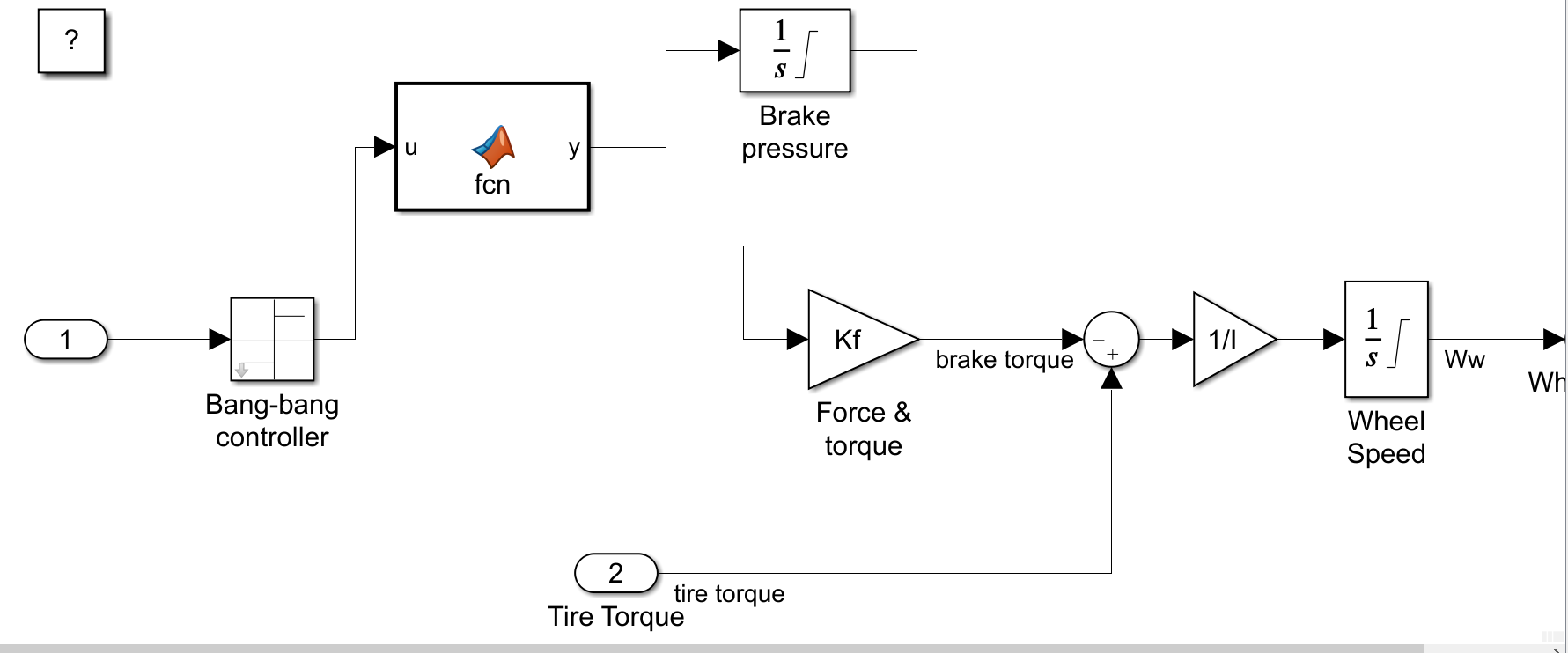
**Data Inspector**

For data inspector we logged slip signal and output signal

**Solver selection strategy**

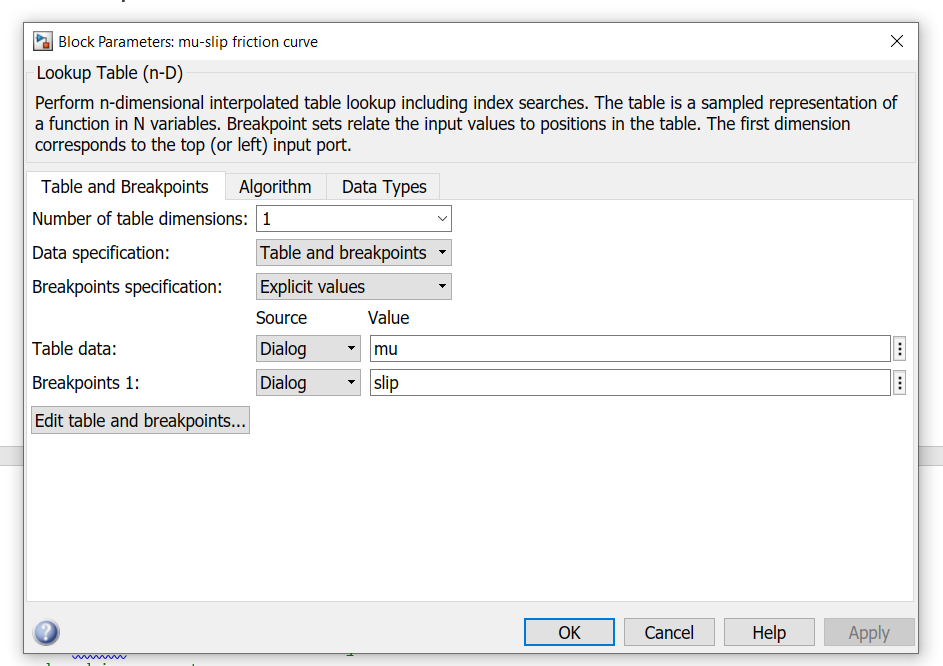
We are using ODE45 for solver because it is more fast and robust in this situation as we are taking 25 sec runtime.

**Matlab Function Block**

We used matlab function block inside wheel speed substem for implementing function

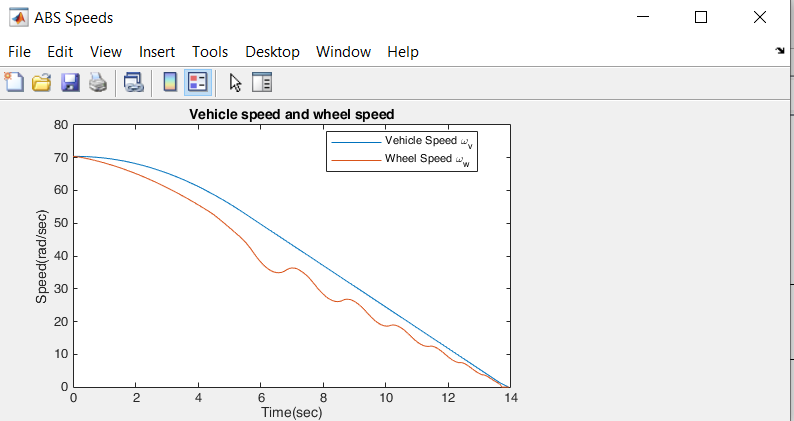
**Lookup Table**

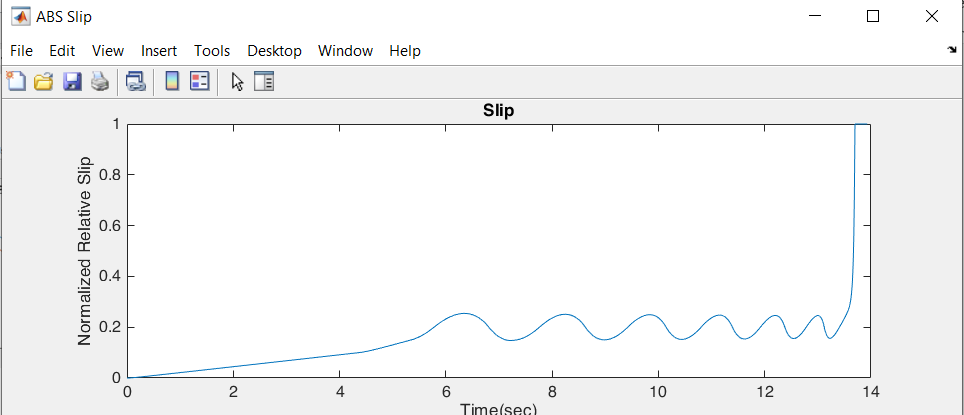
Created an one dimensional Lookup Table with data of mu and breakpoint at slip.



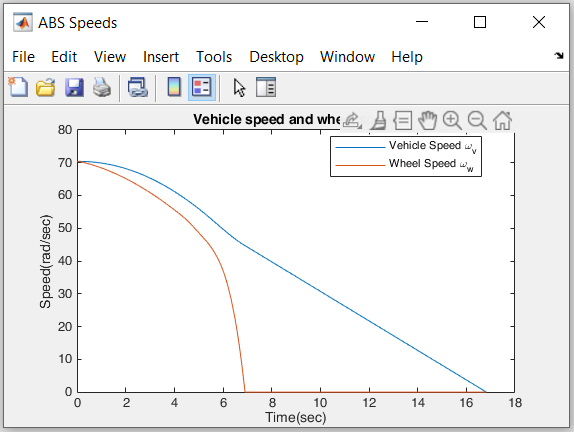
**System performance**

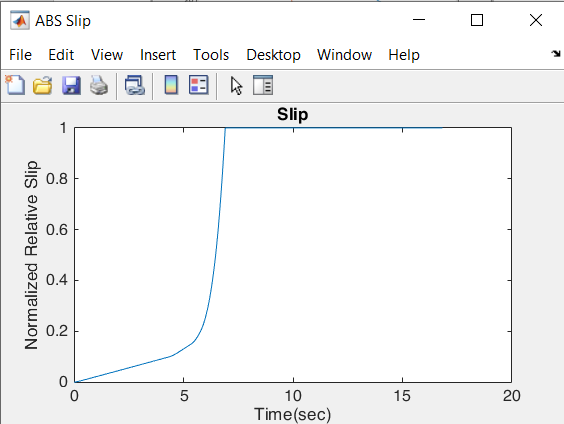
Simulation when abs is ON





Simulation when abs is OFF





# **Conclusions**

# This model shows how you can use Simulink to simulate a braking system

# under the action of an ABS controller. The controller in this example is

# idealized, but you can use any proposed control algorithm in its place to

# evaluate the system's performance. You can also use the Simulink(R)

# Coder(TM) with Simulink as a valuable tool for rapid prototyping of the

# proposed algorithm. C code is generated and compiled for the controller

# hardware to test the concept in a vehicle. This significantly reduces the

# time needed to prove new ideas by enabling actual testing early in the

# development cycle.

# For a hardware-in-the-loop braking system simulation, you can remove the

# 'bang-bang' controller and run the equations of motion on real-time

# hardware to emulate the wheel and vehicle dynamics. You can do this by

# generating real-time C code for this model using the Simulink Coder. You

# can then test an actual ABS controller by interfacing it to the real-time

# hardware, which runs the generated code. In this scenario, the real-time

# model would send the wheel speed to the controller, and the controller

# would send brake action to the model.