KPIT: VIRTUAL GENESIS 2020

REPORT ON

# ANTI-LOCK BRAKING SYSTEM

# Using MATLAB

MODEL BASED DESIGN MILESTONE

SUBMITTED BY:

SUDHANSHU KULSHRESHTHA

Unique ID: 2005005

SUBMITTED TO:

MR. PAWAN KUMAR FAKATKAR

# INTRODUCTION

An anti-lock braking system (ABS) is a safety anti-skid braking system used on aircraft and on land vehicles, such as cars, motorcycles, trucks, and buses. ABS operate by preventing the wheels from locking up during braking, thereby maintaining tractive contact with the road surface and allowing the driver to maintain more control over the vehicle.

ABS is an automated system that uses the principles of threshold braking and cadence braking, techniques which were once practiced by skilful drivers before ABS was widespread. ABS operate at a much faster rate and more effectively than most drivers could manage. Although ABS generally offer improved vehicle control and decreases stopping distances on dry and some slippery surfaces, on loose gravel or snow-covered surfaces ABS may significantly increase braking distance, while still improving steering control. Since ABS was introduced in production vehicles, such systems have become increasingly sophisticated and effective. Modern versions may not only prevent wheel lock under braking, but may also alter the front-to-rear brake bias. This latter function, depending on its specific capabilities and implementation, is known variously as electronic brakeforce distribution, traction control system, emergency brake assist, or electronic stability control (ESC). There are four main components of ABS: wheel speed sensors, valves, a pump, and a controller.

# MODELLING OF THE ABS

The model represents a single wheel, which may be replicated a number of times to create a model for a multi-wheel vehicle. In this model, the wheel speed is calculated in a separate model wheel speed calculator. This component is then referenced using a 'Model' block. Note that both the top model and the referenced model use a variable step solver, so Simulink will track zero-crossings in the referenced model.

## *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

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

Equation 1:

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

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

$\omega_v =$ vehicle speed divided by wheel radius

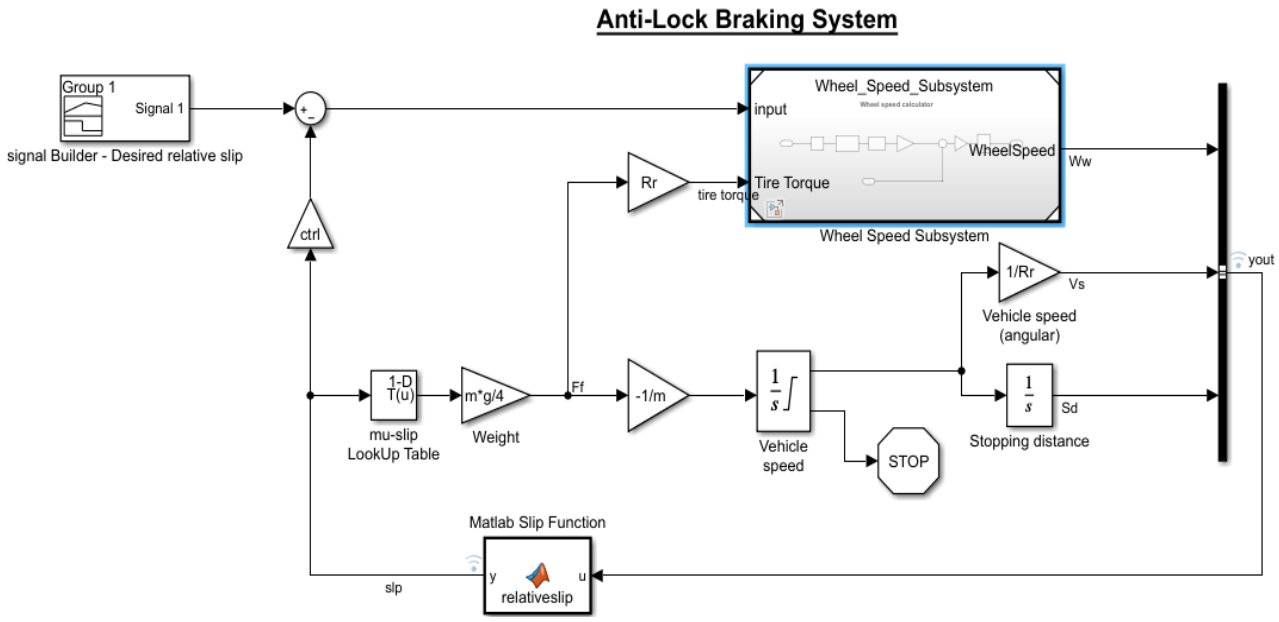
$V_v =$ vehicle linear velocity

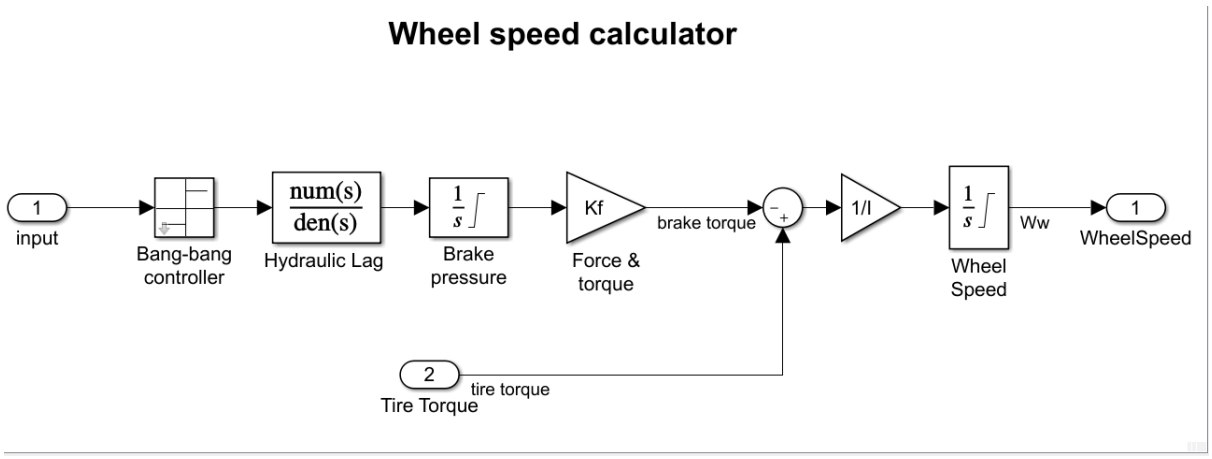
$R_r =$ wheel radius

$\omega_w =$ 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. This maximizes the adhesion between the tire and road and minimizes the stopping distance with the available friction.

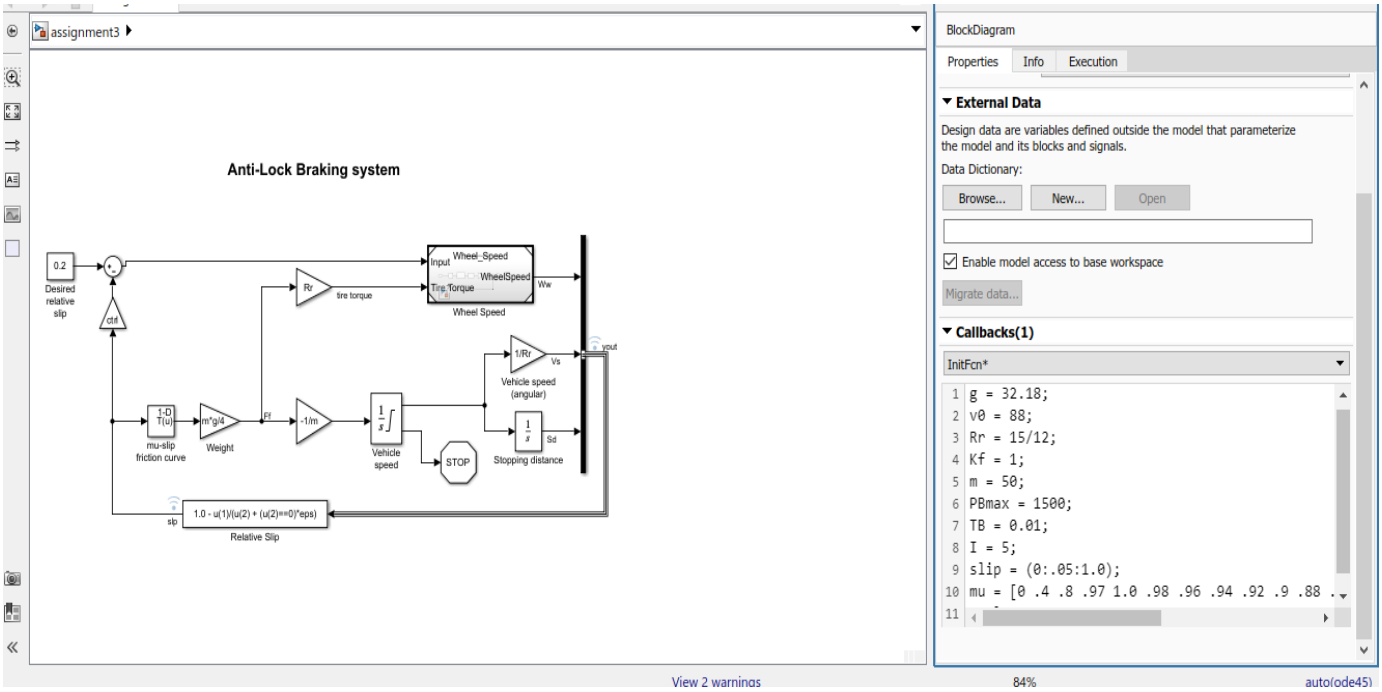
The friction coefficient between the tire and the road surface, mu, is an empirical function of slip, known as the mu-slip curve. We created mu-slip curves by passing MATLAB variables into the block diagram using a Simulink lookup table. The model multiplies the friction coefficient, mu, by the weight 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.



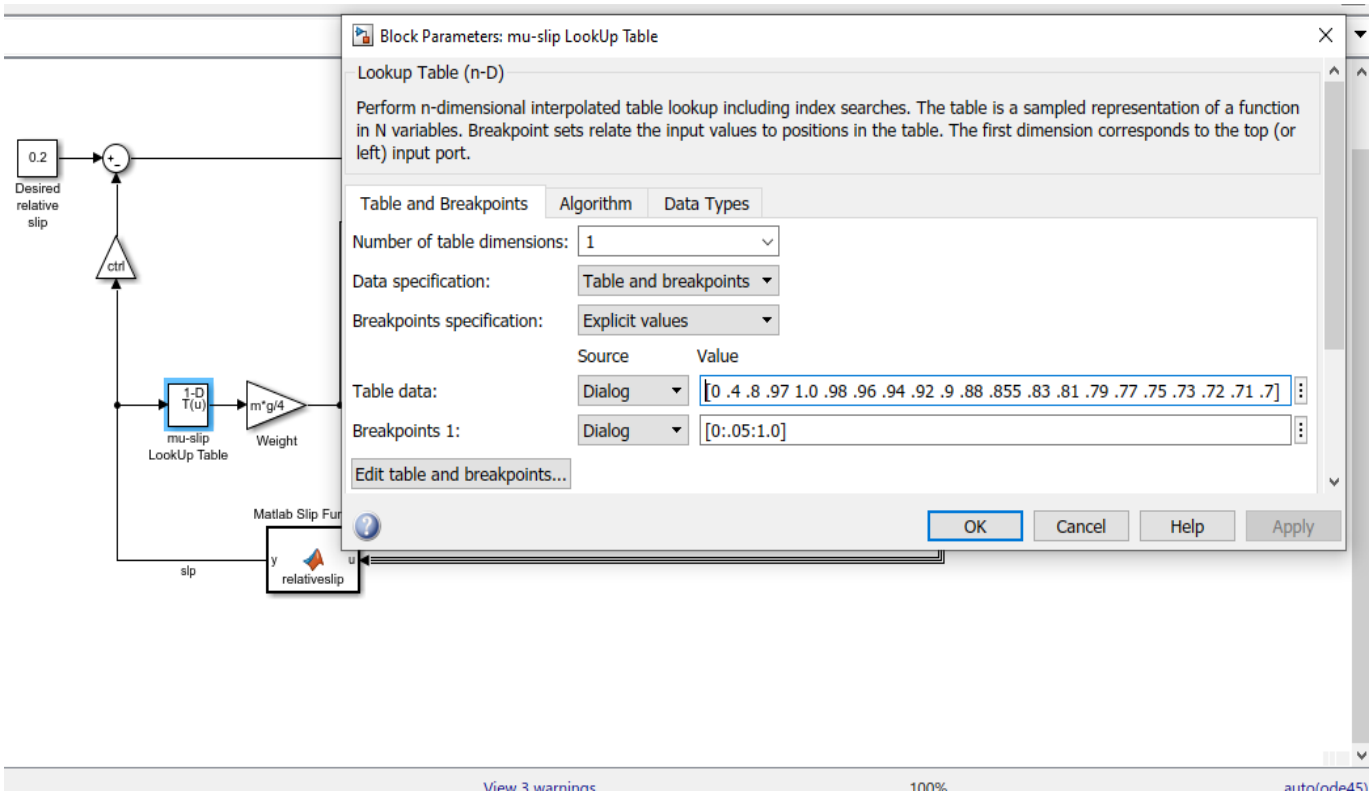



## Some Important Features included in the Model:

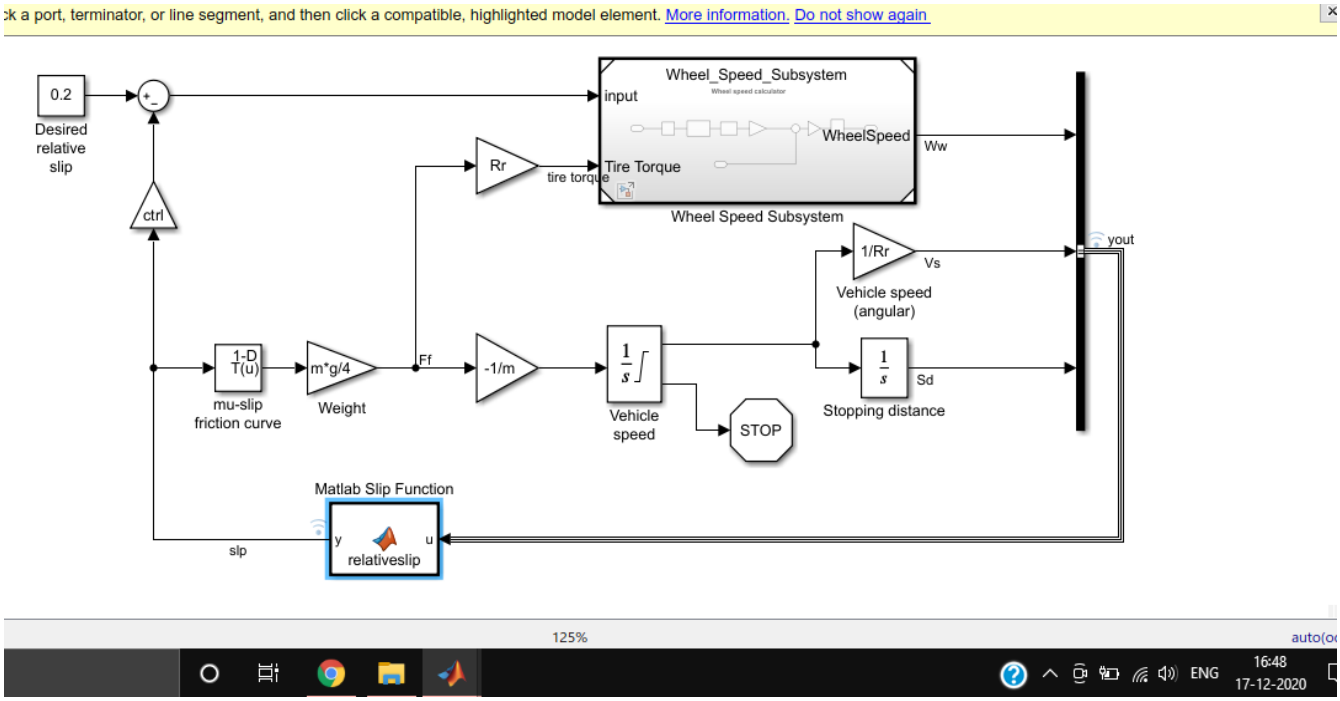
### 1. Call-back Function:

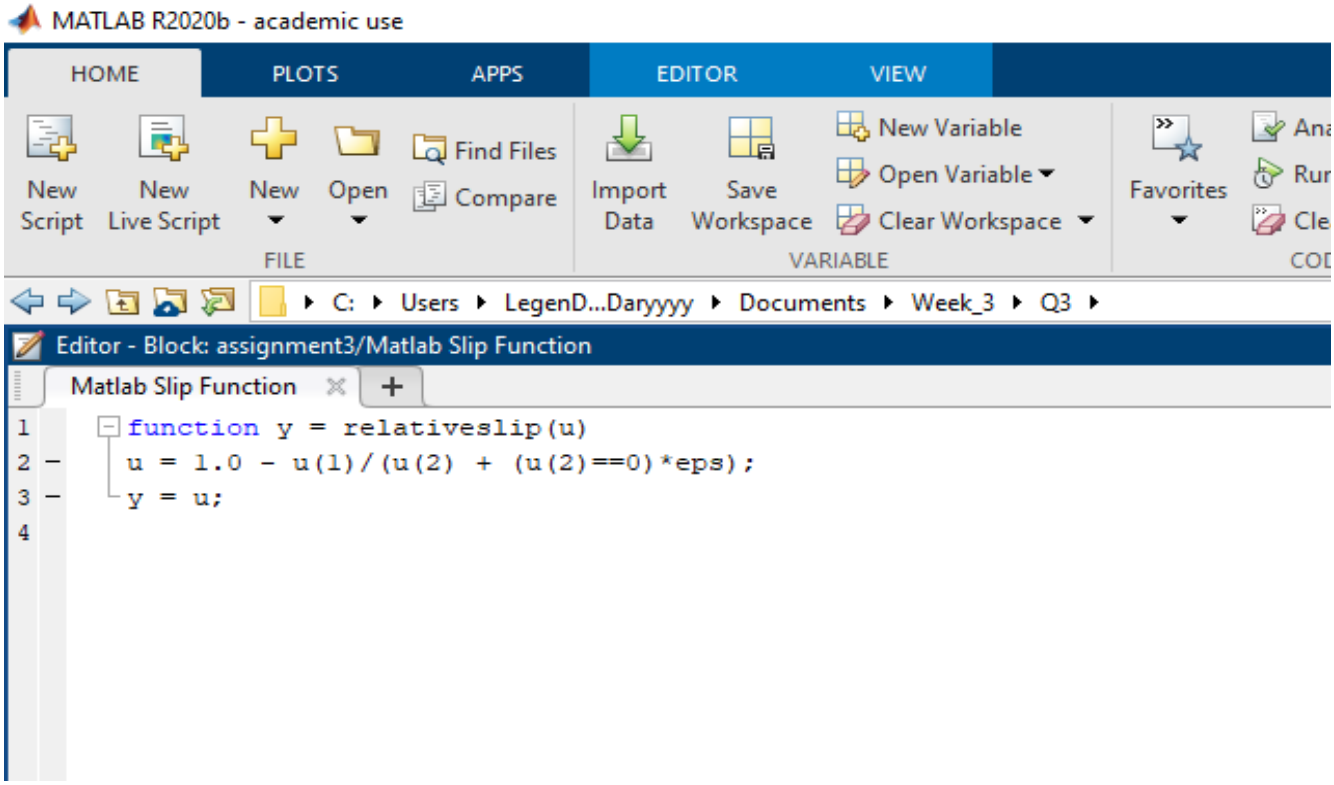



### 2. Look-up Table (Mu and Slip):

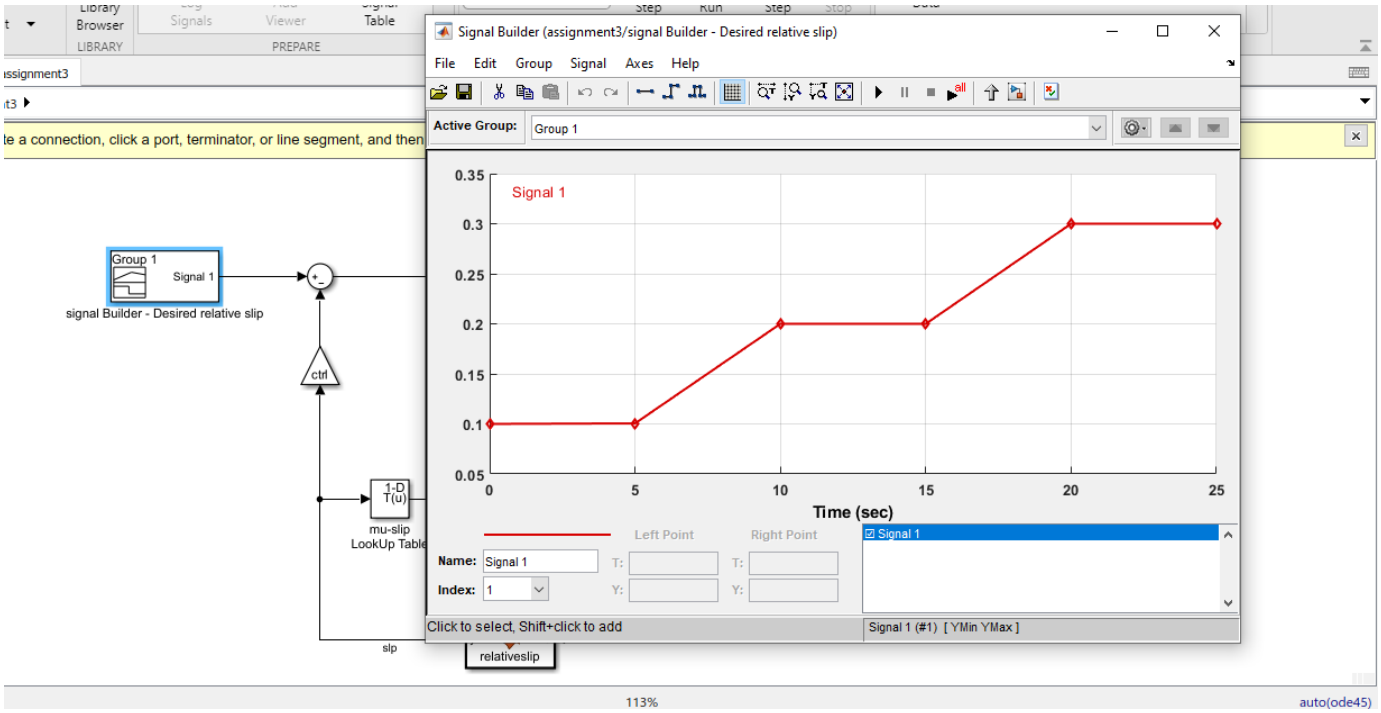


# 3. MATLAB Function Block with script:



## 4. Signal Builder:




## 5. Solver selection strategy:

ode45 performs well with most ODE problems and should generally be your first choice of solver. However, ode23 and ode113 can be more efficient than ode45 for problems with looser or tighter accuracy requirements.

Some ODE problems exhibit stiffness, or difficulty in evaluation. Stiffness is a term that defies a precise definition, but in general, stiffness occurs when there is a difference in scaling somewhere in the problem. For example, if an ODE has two solution components that vary on drastically different time scales, then the equation might be stiff. You can identify a problem as stiff if nonstiff solvers (such as ode45) are unable to solve the problem or are extremely slow. If you observe that a nonstiff solver is very slow, try using a stiff solver such as ode15s instead. When using a stiff solver, you can improve reliability and efficiency by supplying the Jacobian matrix or its sparsity pattern.

ode45 is a versatile ODE solver and is the first solver you should try for most problems. However, if the problem is stiff or requires high accuracy, then there are other ODE solvers that might be better suited to the problem.

**Results:** Wheel speed Ww vs Vehicle Speed Wv **AND** slip