Vehicle Dynamics Blockset™

User's Guide

MATLAB® & SIMULINK®

R2019b

MathWorks®

How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us

Phone: 508-647-7000

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

Vehicle Dynamics Blockset™ User's Guide

Revision History

March 2018	Online only	New for Version 1.0 (Release 2018a)
September 2018	Online only	Revised for Version 1.1 (Release 2018b)
March 2019	Online only	Revised for Version 1.2 (Release 2019a)
September 2019	Online only	Revised for Version 1.3 (Release 2019b)

Contents

1 Getting Started

2 Coordinate Systems

3 Reference Applications

4 Project Templates

5 Maneuver Standards

6 Supporting Data

7 Vehicle Dynamics Blockset Examples

1

Getting Started

Vehicle Dynamics Blockset Product Description

Model and simulate vehicle dynamics in a virtual 3D environment

Vehicle Dynamics Blockset™ provides fully assembled reference application models that simulate driving maneuvers in a 3D environment. You can use the prebuilt scenes to visualize roads, traffic signs, trees, buildings, and other objects around the vehicle. You can customize the reference models by using your own data or by replacing a subsystem with your own model. The blockset includes a library of components for modeling propulsion, steering, suspension, vehicle bodies, brakes, and tires.

Vehicle Dynamics Blockset provides a standard model architecture that can be used throughout the development process. It supports ride and handling analyses, chassis controls development, software integration testing, and hardware-in-the-loop testing. By integrating vehicle dynamics models with a 3D environment, you can test ADAS and automated driving perception, planning, and control software. These models let you test your vehicle with standard driving maneuvers such as a double lane change or with your own custom scenarios.

Key Features

- Preassembled vehicle dynamics models for passenger cars and trucks
- Preassembled maneuvers for common ride and handling tests, including a double-lane change
- 3D environment for visualizing simulations and communicating scene information to Simulink®
- Libraries of propulsion, steering, suspension, vehicle body, brake, and tire components
- Combined longitudinal and lateral slip dynamic tire models
- Predictive driver model for generating steering commands that track a predefined path
- Prebuilt 3D scenes, including straight roads, curved roads, and parking lots

Required and Recommended Products

Required Products

Vehicle Dynamics Blockset product requires current versions of these products:

- MATLAB
- Simulink

Recommended Products

You can extend the capabilities of the Vehicle Dynamics Blockset using the following recommended products.

Goal	Recommended Products
Model events	Stateflow®
Test closed-loop perception, planning, and control algorithms	Automated Driving Toolbox™
Test vehicle-level integration Optimize vehicle energy consumption, ride and handling	Powertrain Blockset™
Generate optimized suspension parameters	Model-Based Calibration Toolbox™ Simscape™ Multibody™

See Also

More About

- “3D Visualization Engine Requirements” on page 1-4

3D Visualization Engine Requirements

The 3D visualization engine requires:

- A Windows® 64-bit platform. If you do not enable the 3D visualization engine, Vehicle Dynamics Blockset runs on Windows, Mac, and Linux® 64-bit platforms.
- Microsoft® DirectX®. If it is not already installed on your machine, Vehicle Dynamics Blockset prompts you to install the software the first time you enable 3D visualization.

To use the Vehicle Dynamics Blockset 3D visualization engine, consider these minimum hardware requirements:

- Graphics card (GPU): Virtual Reality (VR) ready with 8-GB on-board RAM
- Processor (CPU): 2.60 GHz
- Memory (RAM): 12 GB

Limitations

The 3D visualization engine and blocks do not support:

- Code generation.
- Model reference.
- Multiple instances of the Simulation 3D Scene Configuration block.
- Multiple instances of the same actor tag. To refer to the same scene actor when you use the 3D block pairs (e.g. Simulation 3D Actor Transform Get and Simulation 3D Actor Transform Set), specify the same **Tag for actor in 3D scene, Actortag** parameter.
- Parallel simulations.
- Rapid accelerator mode.

See Also

Simulation 3D Scene Configuration

More About

- “Vehicle Dynamics Blockset Communication with 3D Visualization Software” on page 1-6

- “Scene Interrogation in 3D Environment” on page 3-14

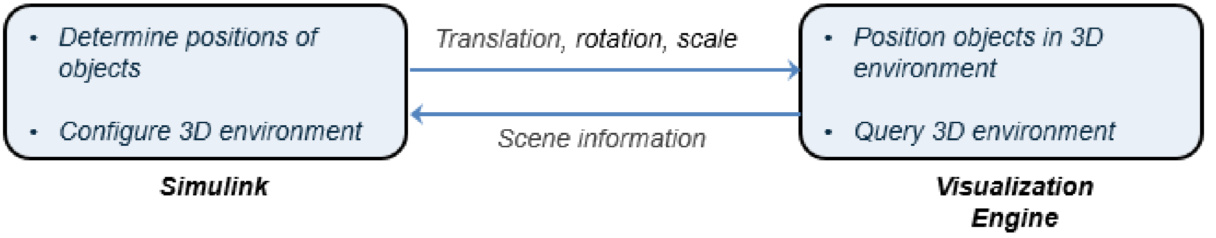
External Websites

- Unreal Engine

Vehicle Dynamics Blockset Communication with 3D Visualization Software

The vehicle dynamics models run programmable maneuvers in a photorealistic 3D visualization environment. Vehicle Dynamics Blockset integrates the 3D simulation environment with Simulink so that you can query the world around the vehicle for virtually testing perception, control, and planning algorithms. The Vehicle Dynamics Blockset visualization environment uses the Unreal Engine® by Epic Games®.

When you use Vehicle Dynamics Blockset to run a maneuver, Simulink can co-simulate with the visualization engine.



In the Simulink environment, Vehicle Dynamics Blockset:

- Determines the next position of objects by using 3D visualization environment feedback and vehicle dynamics models.
- Configures the 3D visualization environment, specifically:
  - Ray tracing
  - Scene capture cameras
  - Initial object positions

In the visualization engine environment, Vehicle Dynamics Blockset positions the objects and uses ray tracing to query the environment.

See Also

Related Examples

- “Send and Receive Double-Lane Change Scene Data” on page 3-65

More About

- “3D Visualization Engine Requirements” on page 1-4
- “Scene Interrogation in 3D Environment” on page 3-14

External Websites

- Unreal Engine

Engine Calibration Maps

Calibration maps are a key part of the Mapped CI Engine and Mapped SI Engine blocks available in the Vehicle Dynamics Blockset. Engine models use the maps to represent engine behavior and to store optimal control parameters. Using calibration maps in control design leads to flexible, efficient control algorithms and estimators that are suitable for electronic control unit (ECU) implementation.

To develop the calibration maps for engine plant models in the reference applications, MathWorks® developed and used processes to measure performance data from 1.5–L spark-ignition (SI) and compression-ignition (CI) engine models provided by Gamma Technologies LLC.

To represent the behavior of engine plants specific to your application, you can develop your own engine calibration maps. The data required for calibration typically comes from engine dynamometer tests or engine hardware design models.

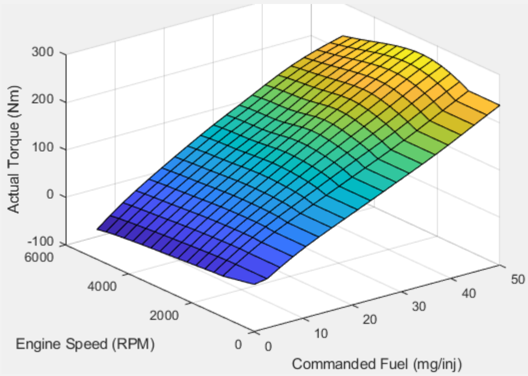
Engine Plant Calibration Maps

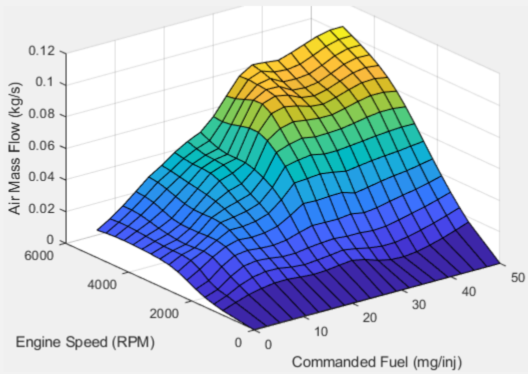
The engine plant model calibration maps in the Mapped CI Engine and Mapped SI Engine blocks affect the engine response to control inputs (for example, spark timing, throttle position, and cam phasing).

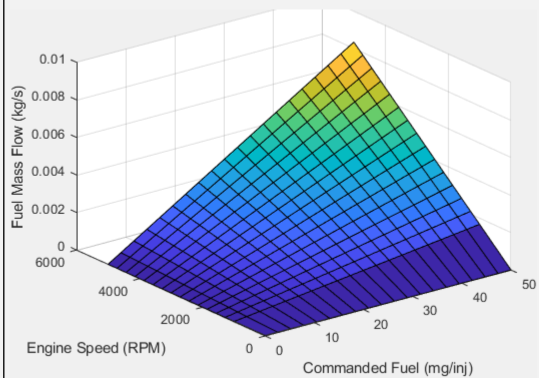
To develop the calibration maps in the engine plant models, MathWorks used GT-POWER models from the GT-SUITE modeling library in a Simulink-based virtual dynamometer. MathWorks used the Model-Based Calibration Toolbox to create design-of-experiment (DoE) test plans. The Simulink-based virtual dynamometer executed the DoE test plan on GT-POWER 1.5–L SI and CI reference engines. MathWorks used the Model-Based Calibration Toolbox to develop the engine plant model calibration maps from the GT-POWER.

Calibration Maps in the Mapped CI Engine Block

The Mapped CI Engine block implements these calibration maps.

Map	Used For	In	Description
Engine brake torque	Engine brake torque as a function of commanded fuel mass and engine speed	Mapped CI Engine	The engine brake torque lookup table is a function of commanded fuel mass and engine speed, $T_{brake} = f(F, N)$, where: • T_{brake} is engine torque, in N·m. • F is commanded fuel mass, in mg per injection. • N is engine speed, in rpm. 

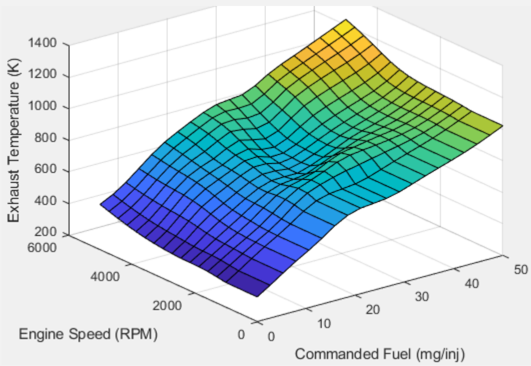
Map	Used For	In	Description
Engine air mass flow	Engine air mass flow as a function of commanded fuel mass and engine speed	Mapped CI Engine	The air mass flow lookup table is a function of commanded fuel mass and engine speed, $\dot{m}_{intk} = f(F_{max}, N)$, where: • $\dot{m}_{intk}$ is engine air mass flow, in kg/s. • F_{max} is commanded fuel mass, in mg per injection. • N is engine speed, in rpm. 

Map	Used For	In	Description
Engine fuel flow	Engine fuel flow as a function of commanded fuel mass and engine speed	Mapped CI Engine	The engine fuel flow lookup table is a function of commanded fuel mass and engine speed, *MassFlow*= ƒ(*F*, *N*), where: • *MassFlow* is engine fuel mass flow, in kg/s. • *F* is commanded fuel mass, in mg per injection. • *N* is engine speed, in rpm. 

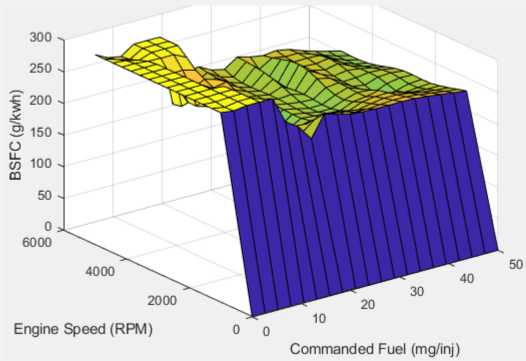
<table>
<tr><th>Map</th><th>Used For</th><th>In</th><th>Description</th></tr>
<tr>
<td>Engine exhaust temperature</td>
<td>Engine exhaust temperature as a function of commanded fuel mass and engine speed</td>
<td>Mapped CI Engine</td>
<td>

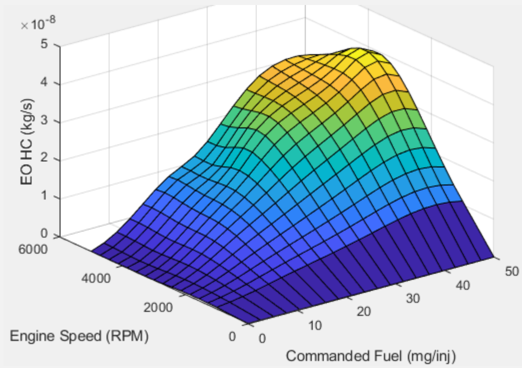
The engine exhaust temperature table is a function of commanded fuel mass and engine speed, $T_{exh} = f(F, N)$, where:

- T_{exh} is exhaust temperature, in K.
- F is commanded fuel mass, in mg per injection.
- N is engine speed, in rpm.



</td>
</tr>
</table>

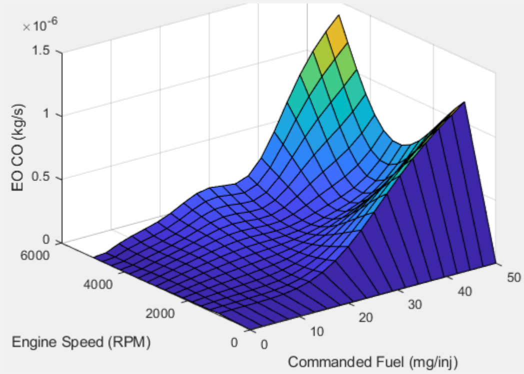
Map	Used For	In	Description
Brake-specific fuel consumption (BSFC) efficiency	BSFC efficiency as a function of commanded fuel mass and engine speed	Mapped CI Engine	The brake-specific fuel consumption (BSFC) efficiency is a function of commanded fuel mass and engine speed, $BSFC = f(F, N)$, where: • $BSFC$ is BSFC, in g/kWh. • F is commanded fuel mass, in mg per injection. • N is engine speed, in rpm. 

Map	Used For	In	Description
Engine-out (EO) hydrocarbon emissions	EO hydrocarbon emissions as a function of commanded fuel mass and engine speed	Mapped CI Engine	The engine-out hydrocarbon emissions are a function of commanded fuel mass and engine speed, $EO\ HC = f(F, N)$, where: • *EO HC* is engine-out hydrocarbon emissions, in kg/s. • *F* is commanded fuel mass, in mg per injection. • *N* is engine speed, in rpm. 

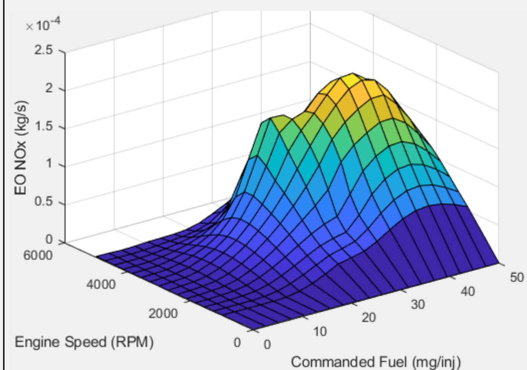
<table>
<tr><th>Map</th><th>Used For</th><th>In</th><th>Description</th></tr>
<tr>
<td>Engine-out (EO) carbon monoxide emissions</td>
<td>EO carbon monoxide emissions as a function of commanded fuel mass and engine speed</td>
<td>Mapped CI Engine</td>
<td>

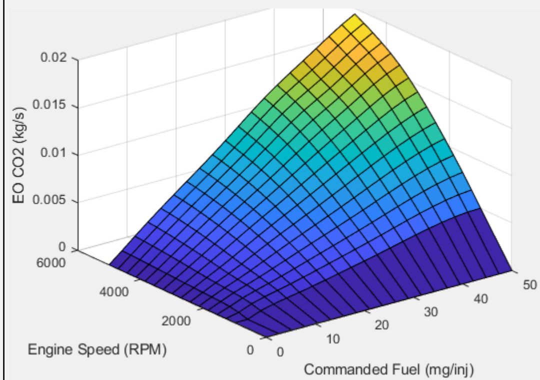
The engine-out carbon monoxide emissions are a function of commanded fuel mass and engine speed, $EO\ CO = f(F, N)$, where:

- $EO\ CO$ is engine-out carbon monoxide emissions, in kg/s.
- F is commanded fuel mass, in mg per injection.
- N is engine speed, in rpm.



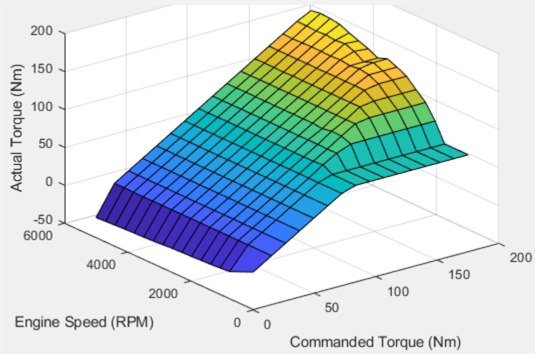
</td>
</tr>
</table>

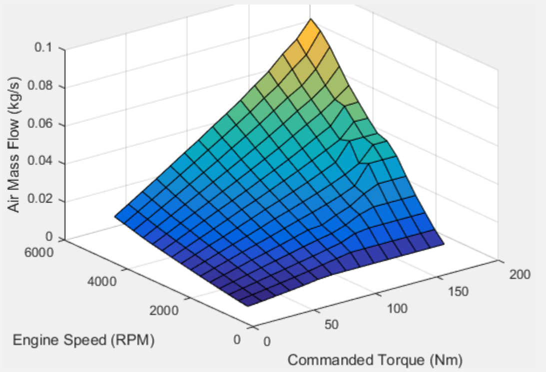
Map	Used For	In	Description
Engine-out (EO) nitric oxide and nitrogen dioxide	EO nitric oxide and nitrogen dioxide emissions as a function of commanded fuel mass and engine speed	Mapped CI Engine	The engine-out nitric oxide and nitrogen dioxide emissions are a function of commanded fuel mass and engine speed, $EO\ NOx = f(F, N)$, where: • $EO\ NOx$ is engine-out nitric oxide and nitrogen dioxide emissions, in kg/s. • F is commanded fuel mass, in mg per injection. • N is engine speed, in rpm. 

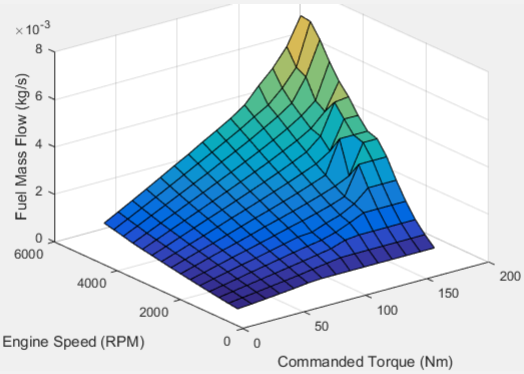
Map	Used For	In	Description
Engine-out (EO) carbon dioxide emissions	EO carbon dioxide emissions as a function of commanded fuel mass and engine speed	Mapped CI Engine	The engine-out carbon dioxide emissions are a function of commanded fuel mass and engine speed, *EO CO2*= ƒ(*F*, *N*), where: • *EO CO2* is engine-out carbon dioxide emissions, in kg/s. • *F* is commanded fuel mass, in mg per injection. • *N* is engine speed, in rpm. 

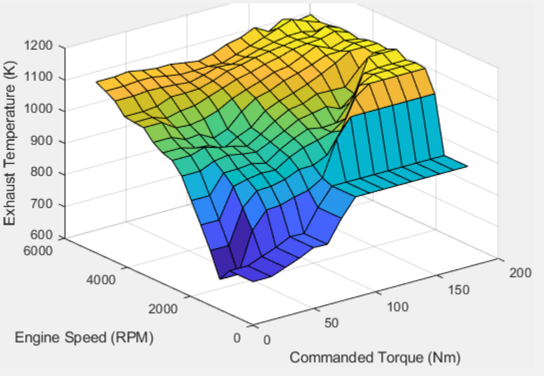
Calibration Maps in the Mapped SI Engine Block

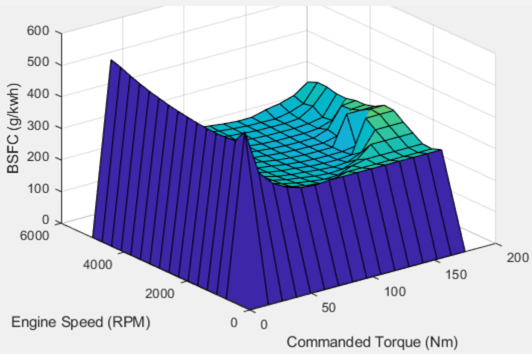
The Mapped SI Engine block implements these calibration maps.

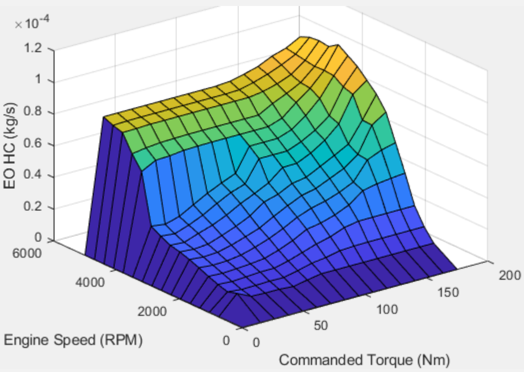
Map	Used For	In	Description
Engine torque	Engine brake torque as a function of commanded torque and engine speed	Mapped SI Engine	The engine torque lookup table is a function of commanded engine torque and engine speed, $T = f(T_{cmd}, N)$, where: • T is engine torque, in N·m. • T_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm. 

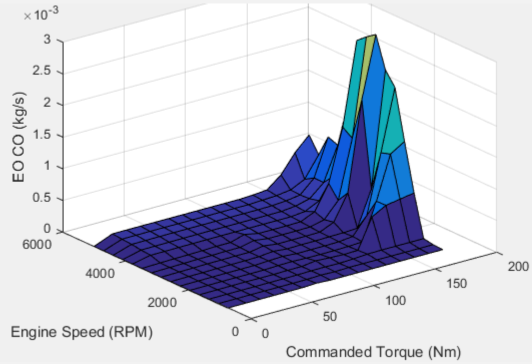
Map	Used For	In	Description
Engine air mass flow	Engine air mass flow as a function of commanded torque and engine speed	Mapped SI Engine	The engine air mass flow lookup table is a function of commanded engine torque and engine speed, $\dot{m}_{intk} = f(T_{cmd}, N)$, where: • $\dot{m}_{intk}$ is engine air mass flow, in kg/s. • T_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm. 

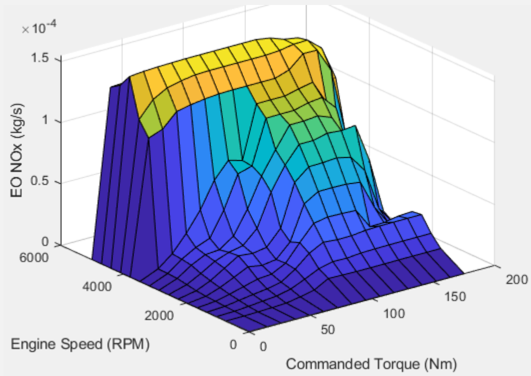
Map	Used For	In	Description
Engine fuel flow	Engine fuel flow as a function of commanded torque mass and engine speed	Mapped SI Engine	The engine fuel mass flow lookup table is a function of commanded engine torque and engine speed, $MassFlow = f(T_{cmd}, N)$, where: • $MassFlow$ is engine fuel mass flow, in kg/s. • T_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm. 

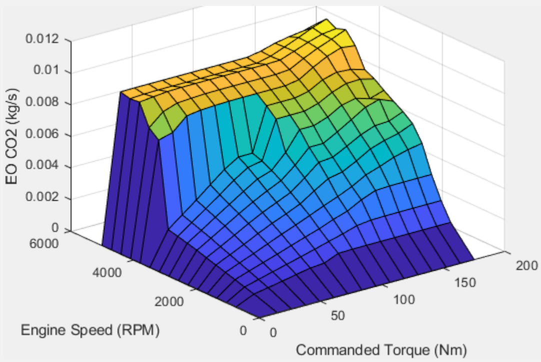
Map	Used For	In	Description
Engine exhaust temperature	Engine exhaust temperature as a function of commanded torque and engine speed	Mapped SI Engine	The engine exhaust temperature lookup table is a function of commanded engine torque and engine speed, $T_{exh} = f(T_{cmd}, N)$, where: • T_{exh} is exhaust temperature, in K. • T_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm. 

Map	Used For	In	Description
Brake-specific fuel consumption (BSFC) efficiency	Brake-specific fuel consumption (BSFC) as a function of commanded torque and engine speed	Mapped SI Engine	The brake-specific fuel consumption (BSFC) efficiency is a function of commanded engine torque and engine speed, $BSFC = f(T_{cmd}, N)$, where: • $BSFC$ is BSFC, in g/kWh. • T_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm. 

Map	Used For	In	Description
Engine-out (EO) hydrocarbon emissions	EO hydrocarbon emissions as a function of commanded torque and engine speed	Mapped SI Engine	The engine-out hydrocarbon emissions are a function of commanded engine torque and engine speed, $EO\ HC = f(T_{cmd}, N)$, where: • $EO\ HC$ is engine-out hydrocarbon emissions, in kg/s. • T_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm. 

Map	Used For	In	Description
Engine-out (EO) carbon monoxide emissions	EO carbon monoxide emissions as a function of commanded torque and engine speed	Mapped SI Engine	The engine-out carbon monoxide emissions are a function of commanded engine torque and engine speed, $EO\ CO = f(T_{cmd}, N)$, where: • $EO\ CO$ is engine-out carbon monoxide emissions, in kg/s. • T_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm. 

Map	Used For	In	Description
Engine-out (EO) nitric oxide and nitrogen dioxide emissions	EO nitric oxide and nitrogen dioxide emissions as a function of commanded torque and engine speed	Mapped SI Engine	The engine-out nitric oxide and nitrogen dioxide emissions are a function of commanded engine torque and engine speed, $EO\ NOx = f(T_{cmd}, N)$, where: • $EO\ NOx$ is engine-out nitric oxide and nitrogen dioxide emissions, in kg/s. • T_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm. 

Map	Used For	In	Description
Engine-out (EO) carbon dioxide emissions	EO carbon dioxide emissions as a function of commanded torque and engine speed	Mapped SI Engine	The engine-out carbon dioxide emissions are a function of commanded engine torque and engine speed, $EO\ CO2 = f(T_{cmd}, N)$, where: • $EO\ CO2$ is engine-out carbon dioxide emissions, in kg/s. • T_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm. 

See Also

Mapped CI Engine | Mapped SI Engine

External Websites

- Virtual Engine Calibration: Making Engine Calibration Part of the Engine Hardware Design Process

Yaw Stability on Varying Road Surfaces

This example shows how to run the vehicle dynamics double-lane change maneuver on different road surfaces, analyze the vehicle yaw stability, and determine the maneuver success.

ISO 3888-2[1] defines the double-lane change maneuver to test the obstacle avoidance performance of a vehicle. In the test, the driver:

- Accelerates until vehicle hits a target velocity
- Releases the accelerator pedal
- Turns steering wheel to follow path into the left lane
- Turns steering wheel to follow path back into the right lane

Typically, cones mark the lane boundaries. If the vehicle and driver can negotiate the maneuver without hitting a cone, the vehicle passes the test.

For more information about the reference application, see “Double-Lane Change Maneuver” on page 3-4.

Run a Double-Lane Change Maneuver

1. Create and open a working copy of the double-lane change reference application.

   ```
   vdynblksDblLaneChangeStart
   ```
2. Open the Lane Change Reference Generator block. By default, the maneuver is set with these parameters:

   - **Longitudinal entrance velocity setpoint** — 35 mph
   - **Vehicle width** — 2 m
   - **Lateral reference position breakpoints** and **Lateral reference data** — Values that specify the lateral reference trajectory as a function of the longitudinal distance
3. In the Visualization subsystem, open the 3D Engine block.

   - Position the vehicle in the recommended location for the double-lane change maneuver.

     a. Set these parameters.

- **Scene** to `Double lane change`
- Select **Recommended for scene**

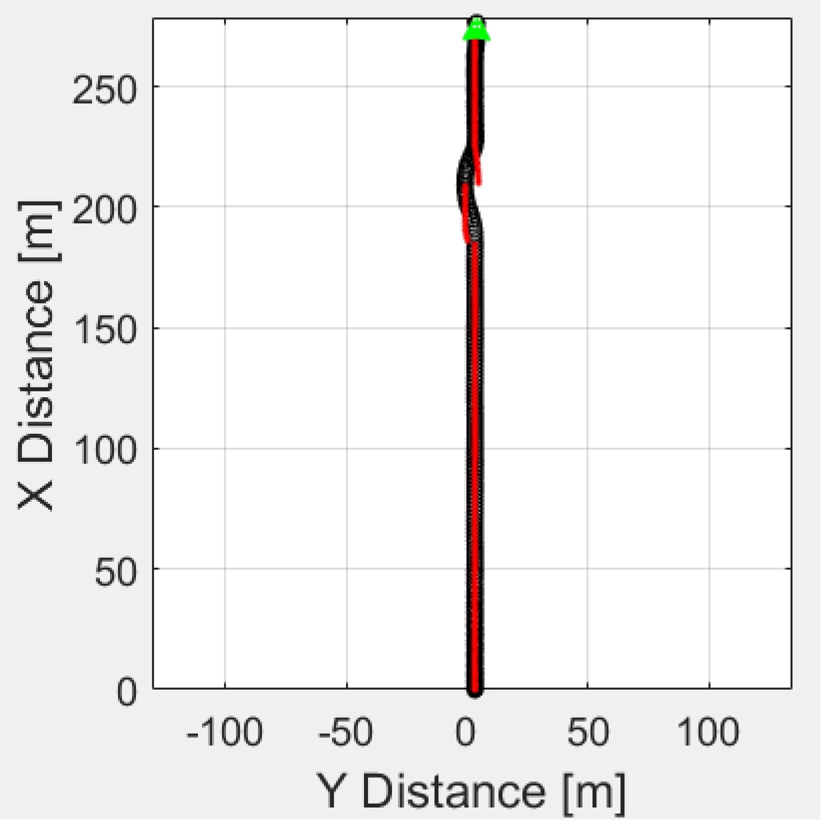
b Select **Apply** to modify the initial vehicle position parameters.

c Click **Update the model workspaces with the initial values** to overwrite the initial vehicle position in the model workspaces with the applied values.

- By default, **3D Engine** parameter is set to **Disabled**. For the 3D visualization engine platform requirements and hardware recommendations, see “3D Visualization Engine Requirements” on page 1-4.

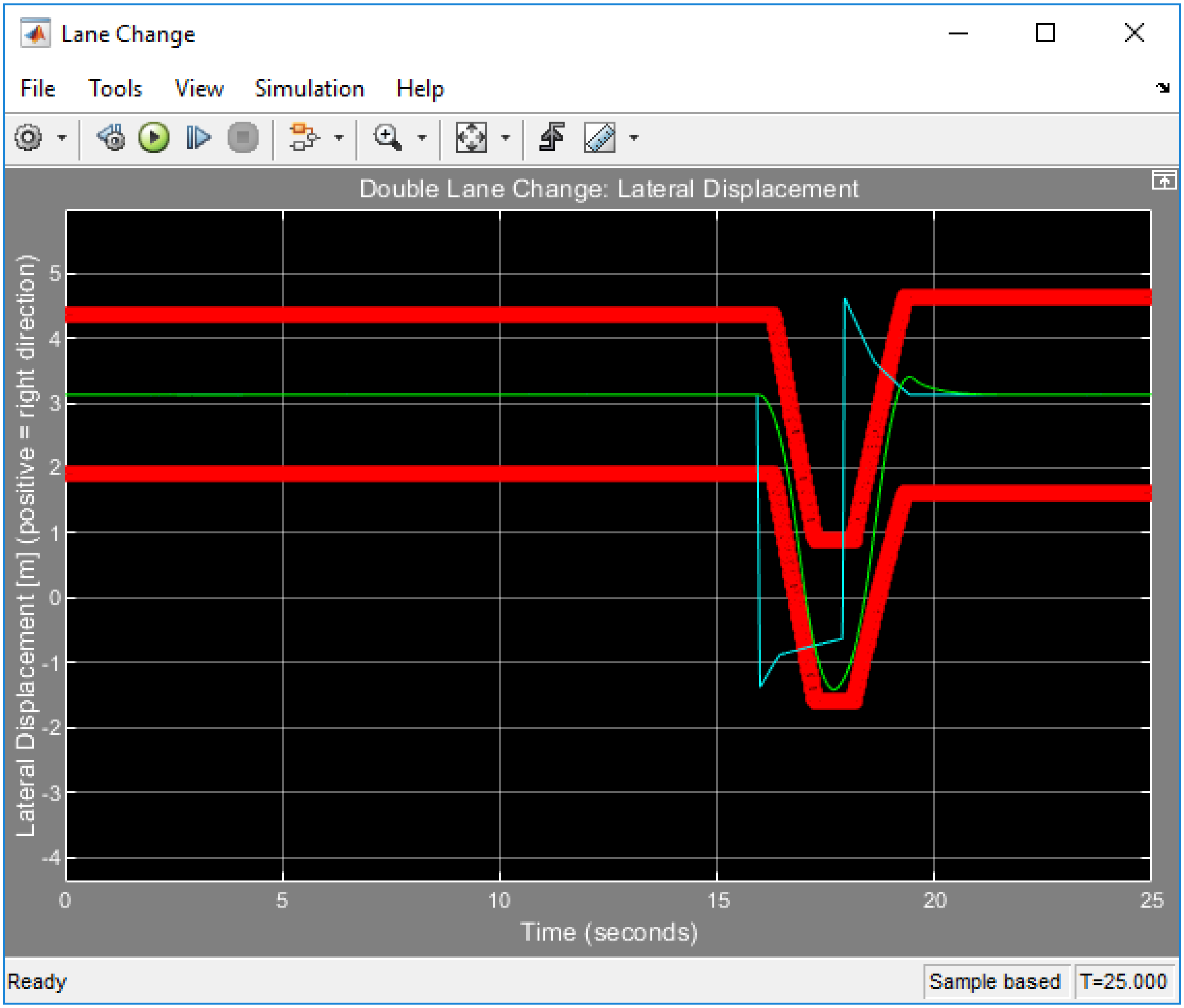
4 Run the maneuver. As the simulation runs, view vehicle information.

- In the Vehicle Position window, view the vehicle longitudinal distance as a function of lateral distance.



- In the Visualization subsystem, open the Lane Change scope block to display the lateral displacement as a function of time.
  - Red line — Cones marking lane boundary
  - Blue line — Reference trajectory
  - Green line — Actual trajectory

  The green line does come close to the red line that marks the cones.

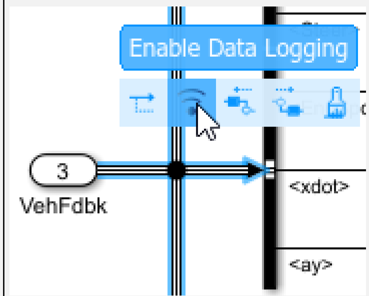
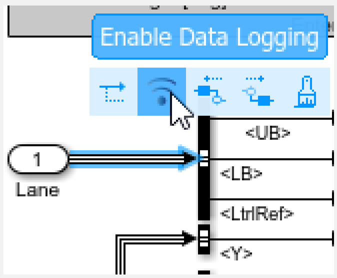
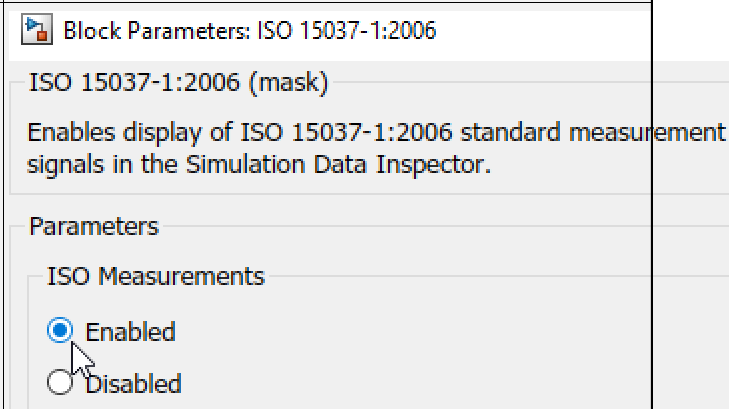


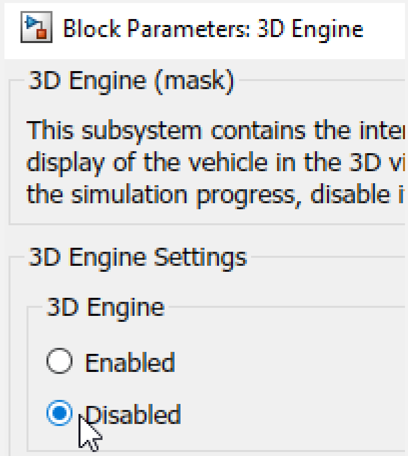
- In the Visualization subsystem, if you enable the 3D Engine block visualization environment, you can view the vehicle response in the `AutoVrtlEnv` window.

Sweep Surface Friction

Run the reference application on three road surfaces with different friction scaling coefficients. Use the results to analyze the yaw stability and help determine the success of the maneuver.

1. In the double-lane change reference application model `DLCReferenceApplication`, open the Environment subsystem. The Friction block parameter **Constant value** specifies the friction scaling coefficient. By default, the friction scaling coefficient is `1.0`. The reference application uses the coefficient to adjust the friction at every time step.
2. In the Visualization subsystem, enable signal logging. You can use the Simulink editor or, alternatively, MATLAB® commands. Save the model.

Model Element	Simulink Editor
VehFdbk inport	
Lane inport	
ISO 15037-1:2006 block	

Model Element	Simulink Editor
3D Engine block	

Alternatively, use these commands to enable the signal logging and save the model.

```
% Open the model
mdl = 'DLCReferenceApplication';
open_system(mdl);

% Enable signal logging for VehFdbk
ph=get_param('DLCReferenceApplication/Visualization/VehFdbk','PortHandles');
set_param(ph.Outport,'DataLogging','on');

% Enable signal logging for Lane
ph=get_param('DLCReferenceApplication/Visualization/Lane','PortHandles');
set_param(ph.Outport,'DataLogging','on');

% Enable signal logging for ISO block
set_param([mdl '/Visualization/ISO 15037-1:2006'],'Measurement','Enable');

% Disable 3D environment
set_param([mdl '/Visualization/3D Engine'],'engine3D','Disabled');

save_system(mdl)
```

3. Set up a vector with the friction scaling coefficients, `lambdamu`, that you want to investigate. For example, to examine friction scaling coefficients equal to `0.9`, `0.95`, and `1.0`, at the command line enter:

```
mdl = 'DLCReferenceApplication';
open_system(mdl);
% Define the set of parameters to sweep
```

```
lambdamu = [0.9, 0.95, 1.0];
numExperiments = length(lambdamu);
```

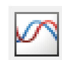
4 Create an array of simulation inputs that sets `lambdamu` equal to the Friction constant block parameter.

```
% Create an array of Simulink.SimulationInputs
for idx = numExperiments:-1:1
    in(idx) = Simulink.SimulationInput(mdl);
    in(idx) = in(idx).setBlockParameter([mdl '/Environment/Friction'],'Value',['ones(4,1).*',num2str(la
end
```

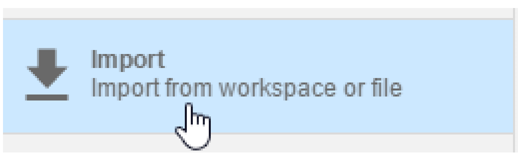
5 Set the simulation stop time at 30 s. Save the model and run the simulations. If available, use parallel computing.

```
set_param(mdl,'StopTime','30')
save_system(mdl)
tic;
simout = parsim(in,'ShowSimulationManager','on');
toc;
```

6 Import the simulation results to the Simulation Data Inspector.

   a On the Simulink Editor toolbar, click the **Data Inspector** button .

   b In the Simulation Data Inspector, select **Import**.



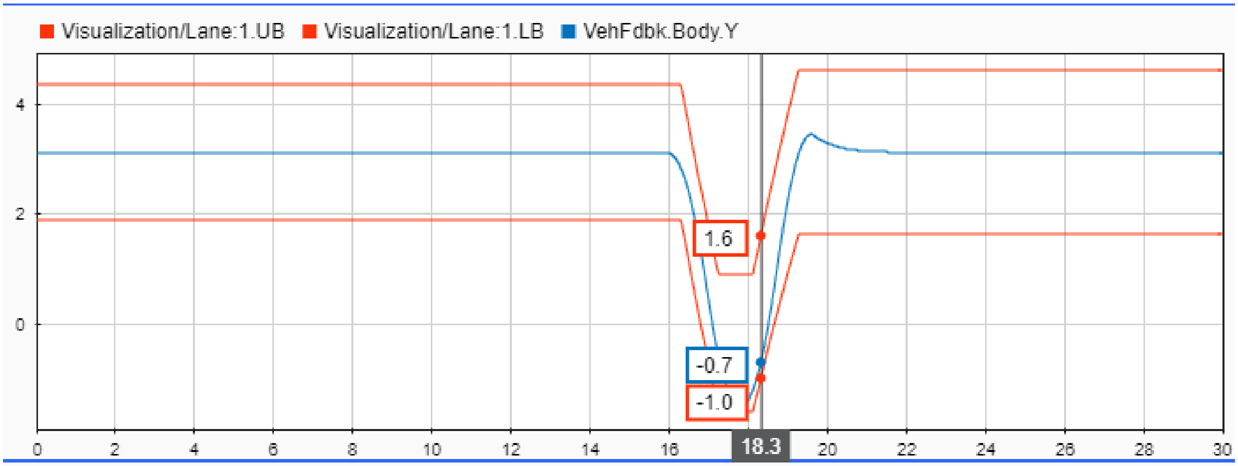
   c In the Import dialog box, clear `logsout`. Select `simout(1)`, `simout(2)`, and `simout(3)`. Select **Import**.

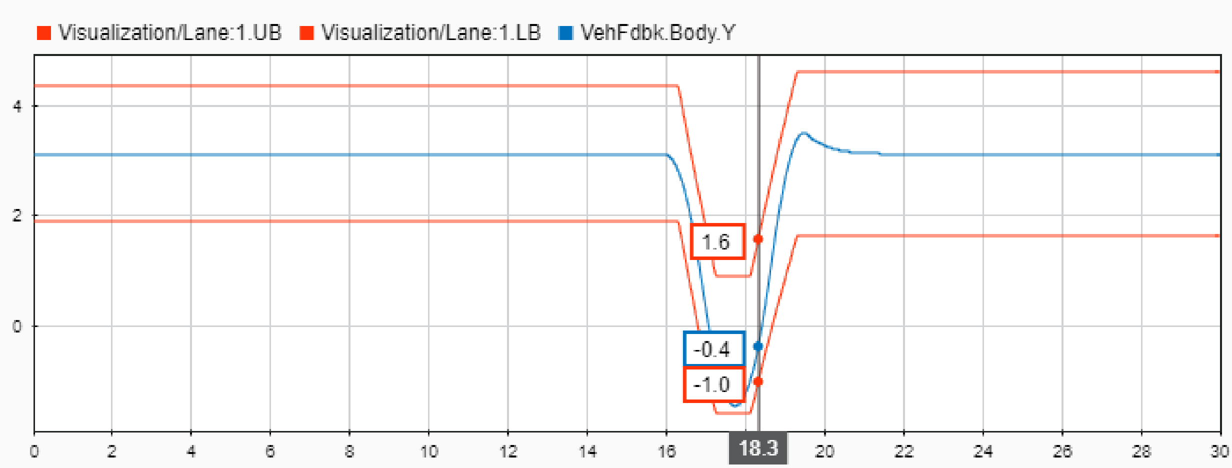
d Select each of the runs. For each run, right-click to rename the run to the friction scaling coefficient that corresponds to the simulation.

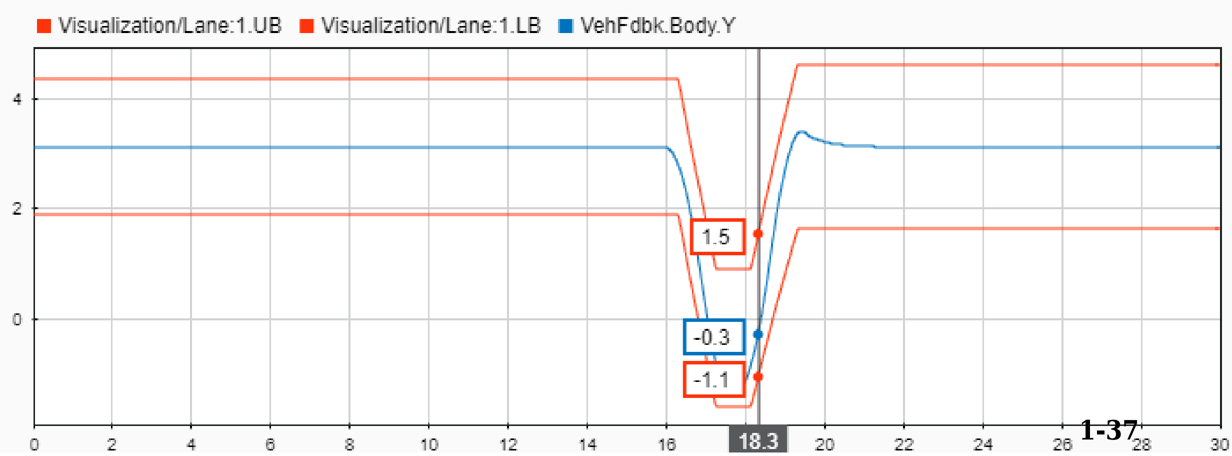
7 Explore the results in the Simulation Data Inspector.

- To assess the success of the maneuver test when `lambdamu` is equal to .9, .95, and 1.0, plot the upper lane boundary, `<UB>`, lower lane boundary, `<LB>`, and lateral vehicle distance, `Y`.

The results are similar to these plots, which show the results for the runs. The results indicate that the vehicle lateral position does come close to the lane boundaries.

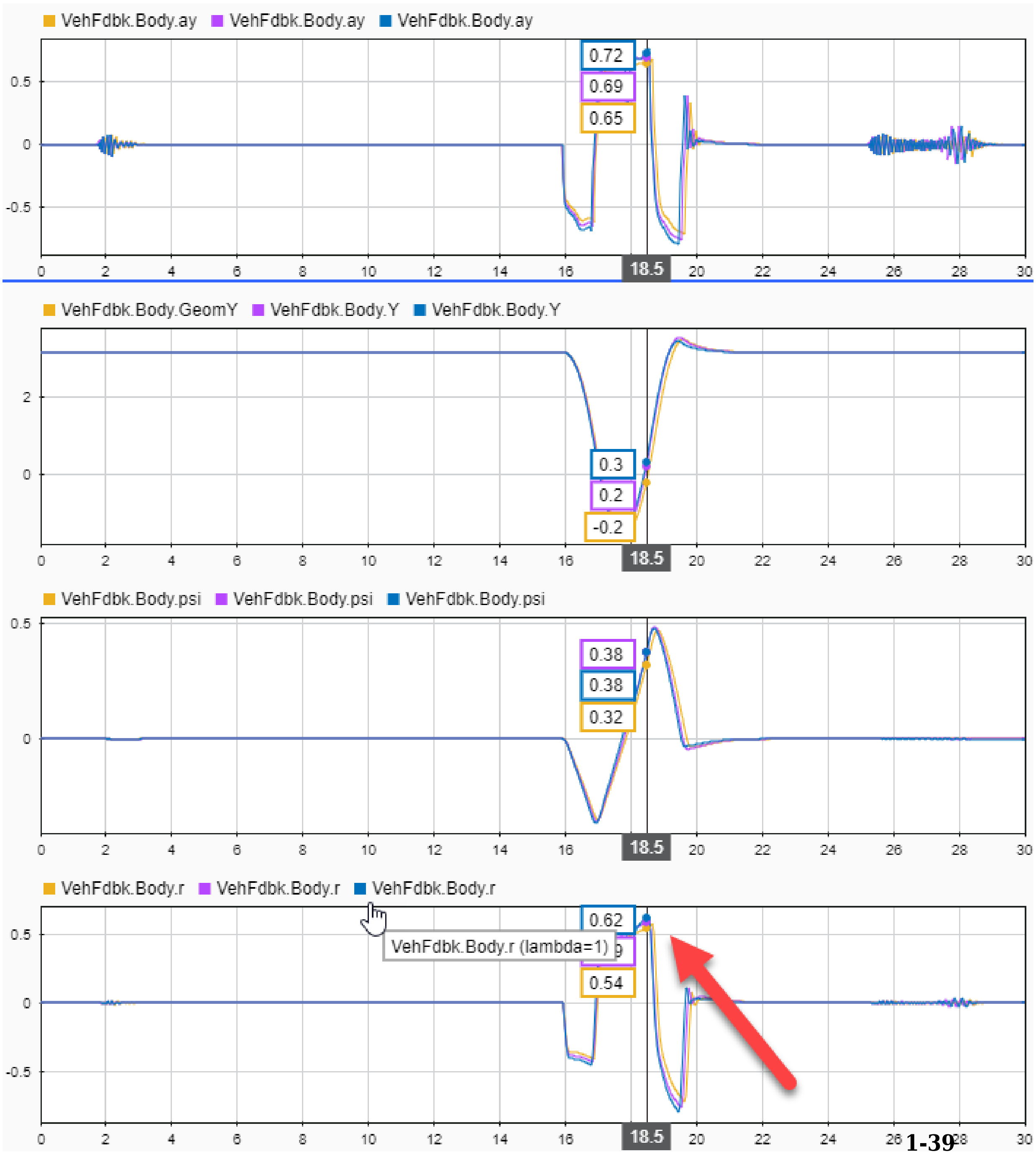

Visualization/Lane:1.UB
Visualization/Lane:1.LB
VehFdbk.Body.Y
1.6
-0.7
-1.0
18.3
0 2 4
0 2 4 6 8 10 12 14 16 20 22 24 26 28 30


Visualization/Lane:1.UB
Visualization/Lane:1.LB
VehFdbk.Body.Y
1.6
-0.4
-1.0
18.3
0 2 4
0 2 4 6 8 10 12 14 16 20 22 24 26 28 30


Visualization/Lane:1.UB
Visualization/Lane:1.LB
VehFdbk.Body.Y
1.5
-0.3
-1.1
18.3
0 2 4
0 2 4 6 8 10 12 14 16 20 22 24 26 28 30

- To assess the yaw stability for the road surfaces, plot the lateral acceleration, `ay`, lateral vehicle distance, `Y`, yaw angle, `psi`, and yaw rate, `r`.

  The results are similar to these plots. The results indicate that the vehicle has a yaw rate of about .62 rad/s when the friction scaling coefficient is equal to 1.


VehFdbk.Body.ay VehFdbk.Body.ay VehFdbk.Body.ay
0.72
0.69
0.65
18.5
VehFdbk.Body.GeomY VehFdbk.Body.Y VehFdbk.Body.Y
0.3
0.2
-0.2
18.5
VehFdbk.Body.psi VehFdbk.Body.psi VehFdbk.Body.psi
0.38
0.38
0.32
18.5
VehFdbk.Body.r VehFdbk.Body.r VehFdbk.Body.r
0.62
VehFdbk.Body.r (lambda=1)
0.54
18.5

8 To explore the results further, use these commands to extract the lateral acceleration, steering angle, and vehicle trajectory from the `simout` object.

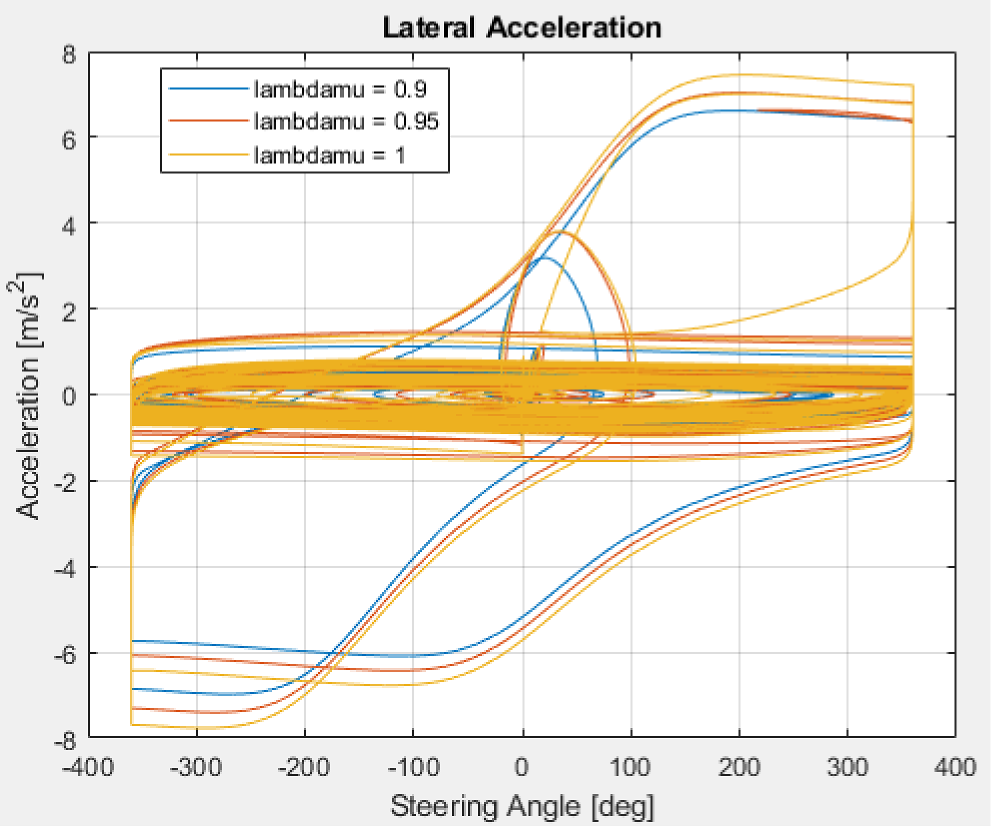
- Extract the lateral acceleration and steering angle. Plot the data.

```
% Plot results from simout object: lateral acceleration vs steering angle
figure
for idx = 1:numExperiments
    % Extract Data
    log = simout(idx).get('logsout');
    sa=log.get('Steering-wheel angle').Values;
    ay=log.get('Lateral acceleration').Values;

    legend_labels{idx} = ['lambdamu = ', num2str(lambdamu(idx))];

    % Plot steering angle vs. lateral acceleration
    plot(sa.Data,ay.Data)
    hold on
end
% Add labels to the plots
legend(legend_labels, 'Location', 'best');
title('Lateral Acceleration')
xlabel('Steering Angle [deg]')
ylabel('Acceleration [m/s^2]')
grid on
```

  The results are similar to this plot. They indicate that the greatest lateral acceleration occurs when the friction scaling coefficient is `1`.



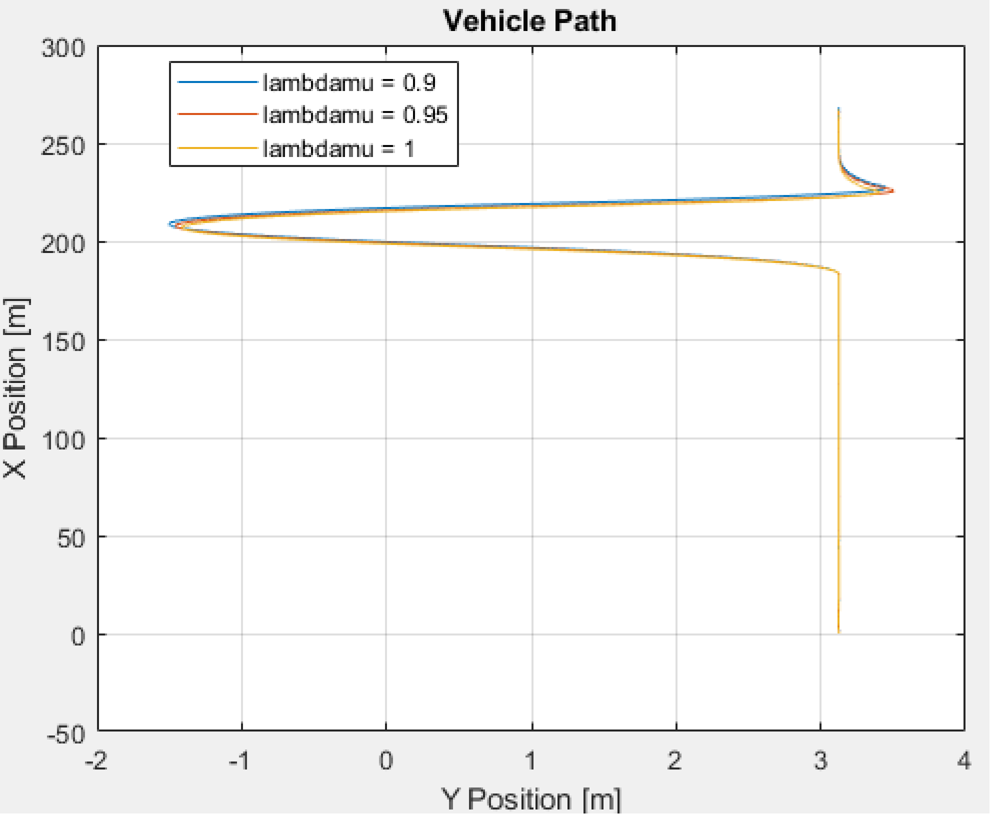
- Extract the vehicle path. Plot the data.

```
figure
for idx = 1:numExperiments
    % Extract Data
    log = simout(idx).get('logsout');
    VehFdbk = log.get('VehFdbk');
    x = VehFdbk.Values.Body.X;
    y = VehFdbk.Values.Body.Y;
    legend_labels{idx} = ['lambdamu = ', num2str(lambdamu(idx))];

    % Plot vehicle location
    plot(y.Data,x.Data)
    hold on
end
```

```
% Add labels to the plots
legend(legend_labels, 'Location', 'best');
title('Vehicle Path')
xlabel('Y Position [m]')
ylabel('X Position [m]')
grid on
```

The results are similar to this plot. They indicate that the greatest lateral vehicle position occurs when the friction scaling coefficient is `0.9`.



See Also

`Simulink.SimulationInput` | `Simulink.SimulationOutput`

References

[1] ISO 3888-2: 2011. *Passenger cars — Test track for a severe lane-change manoeuvre*.

See Also

Related Examples

- “Send and Receive Double-Lane Change Scene Data” on page 3-65

More About

- “Double-Lane Change Maneuver” on page 3-4
- “Vehicle Dynamics Blockset Communication with 3D Visualization Software” on page 1-6
- Simulation Data Inspector

Vehicle Steering Gain at Different Speeds

This example shows how to use the vehicle dynamics slowly increasing steering reference application to analyze the impact of the steering angle and speed on vehicle handling. Specifically, you can calculate the steering gain when you run the maneuver with different speed set points.

Based on the constant speed, variable steer test defined in SAE J266[1], the slowly increasing steering maneuver helps characterize the lateral dynamics of the vehicle. In the test, the driver:

- Accelerates until vehicle hits a target velocity.
- Maintains a target velocity.
- Linearly increases the steering wheel angle from 0 degrees to a maximum angle.
- Maintains the steering wheel angle for a specified time.
- Linearly decreases the steering wheel angle from maximum angle to 0 degrees.

For more information about the reference application, see "Slowly Increasing Steering Maneuver" on page 3-28.

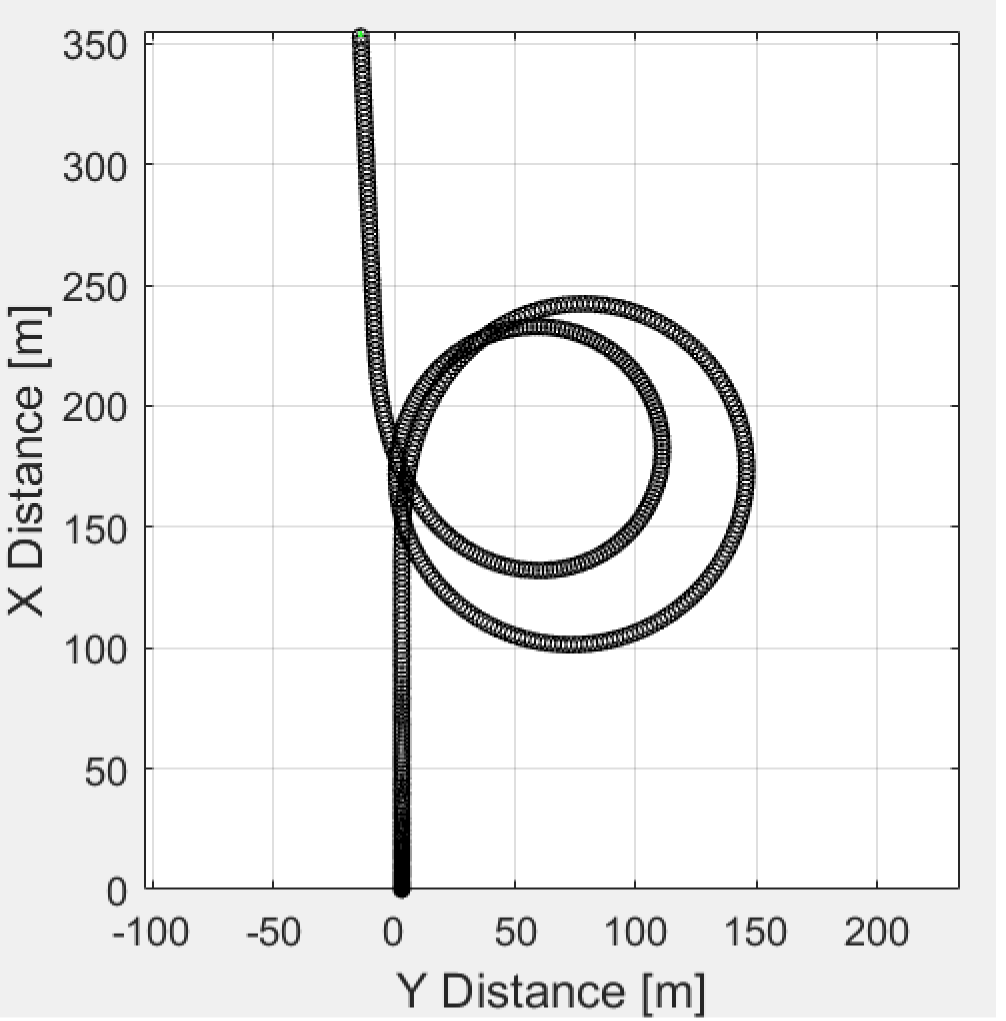
Run a Slowly Increasing Steering Maneuver

1. Create and open a working copy of the increasing steering reference application.

   ```
   vdynblksIncreasingSteeringStart
   ```
2. Open the Slowly Increasing Steer block. By default, the maneuver is set with these parameters:

   - **Longitudinal speed setpoint** — 50 mph
   - **Handwheel rate** — 13.5 deg
   - **Maximum handwheel angle** — 270 deg
3. Open the Visualization subsystem. By default, the 3D Engine is set with the 3D visualization engine disabled. For the 3D visualization engine platform requirements and hardware recommendations, see "3D Visualization Engine Requirements" on page 1-4.
4. Run the maneuver with the default settings. As the simulation runs, view vehicle information.

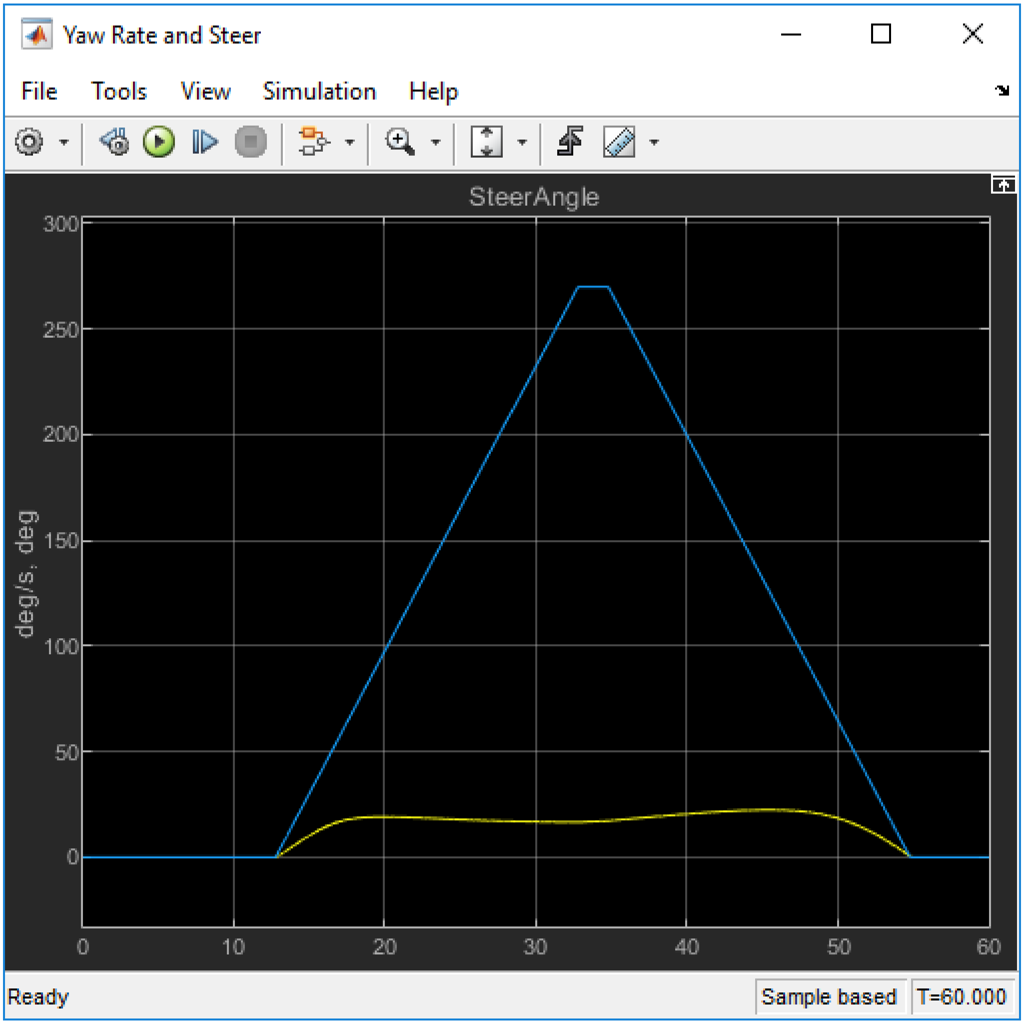
- In the Vehicle Position window, view the vehicle longitudinal distance as a function of lateral distance.



- In the Visualization subsystem, open the Yaw Rate and Steer Scope block to display the yaw rate and steering angle versus time:
  - Yellow line — Yaw rate

- Blue lines — Steering angle

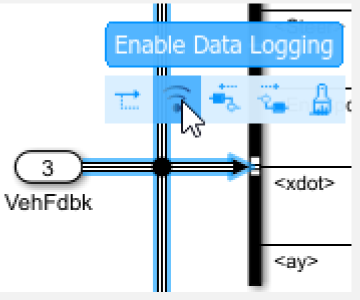
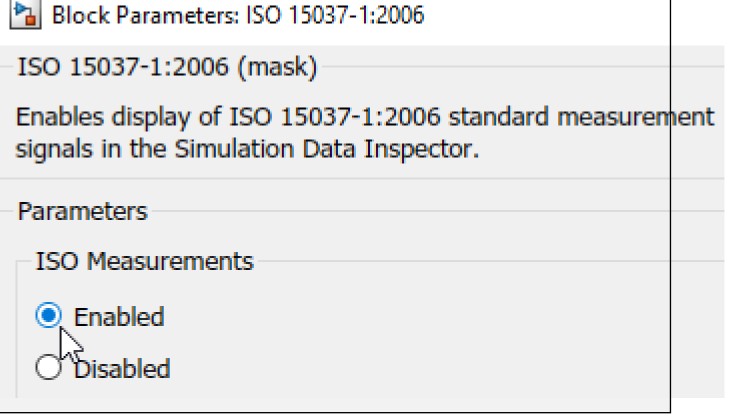
The blue line shows a linearly increasing and decreasing steering angle.

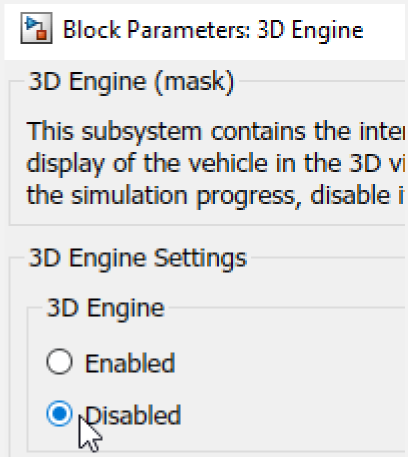


Sweep Speed Set Points

Run the slowly increasing steering angle reference application with three different speed set points.

1. In the slowly increasing steering reference application model `ISReferenceApplication`, open the Slowly Increasing Steer block. The **Longitudinal speed set point, xdot_r** block parameter sets the vehicle speed. By default, the speed is `50` mph.
2. In the Visualization subsystem, enable signal logging for these model elements. Disable the 3D visualization environment. You can use the Simulink editor or, alternatively, MATLAB commands. Save the model.

Model Element	Simulink Editor
VehFdbk inport	
ISO 15037-1:2006 block	

Model Element	Simulink Editor
3D Engine block	

Alternatively, use these commands to enable the signal logging, disable the 3D visualization environment, and save the model.

```
% Open the model
mdl = 'ISReferenceApplication';
open_system(mdl);

% Enable signal logging for VehFdbk
ph=get_param('ISReferenceApplication/Visualization/VehFdbk','PortHandles');
set_param(ph.Outport,'DataLogging','on');

% Enable signal logging for ISO block
set_param([mdl '/Visualization/ISO 15037-1:2006'],'Measurement','Enable');

% Disable 3D environment
set_param([mdl '/Visualization/3D Engine'],'engine3D','Disabled');

save_system(mdl)
```

3 Set up a speed set point vector, `xdot_r`, that you want to investigate. For example, at the command line, enter:

```
mdl = 'ISReferenceApplication';
open_system(mdl);
% Define the set of parameters to sweep
vmax = [40, 50, 60];
tfinal = [60, 60, 60];
numExperiments = length(vmax);
```

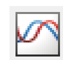
4. Create an array of simulation inputs that set `xdot_r` equal to the Slowly Increasing Steer block parameter.

```
for idx = numExperiments:-1:1
    in(idx) = Simulink.SimulationInput(mdl);
    in(idx) = in(idx).setBlockParameter([mdl '/Slowly Increasing Steer'], 'xdot_r', num2str(vmax(idx)))
    in(idx) = in(idx).setModelParameter('StopTime', num2str(tfinal(idx)));
end
```

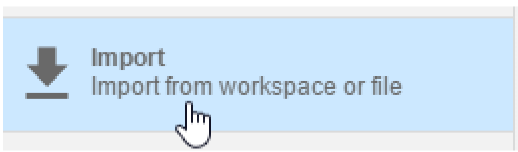
5. Save the model and run the simulations. If available, use parallel computing.

```
save_system(mdl);
tic;
simout = parsim(in,'ShowSimulationManager','on');
toc;
```

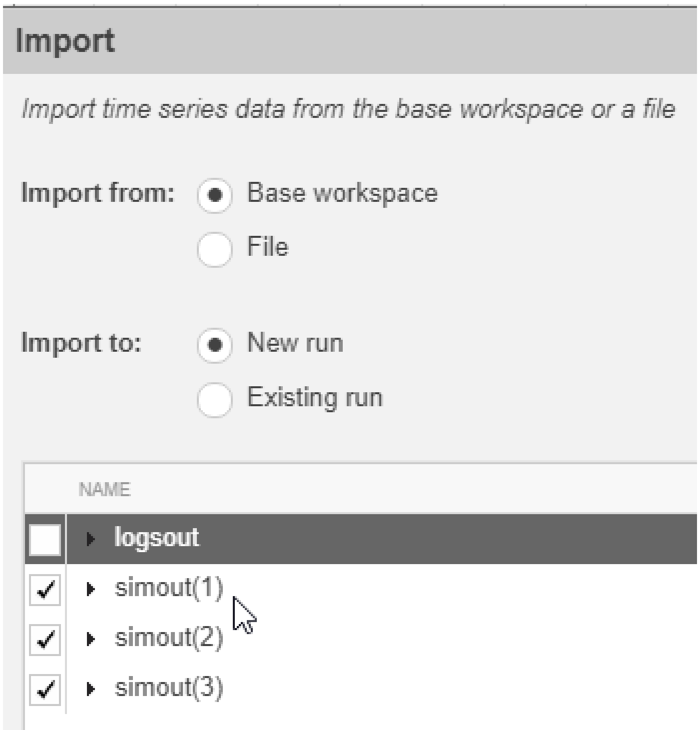
6. Import the simulation results to the Simulation Data Inspector.

   a. On the Simulink Editor toolbar, click the **Data Inspector** button .

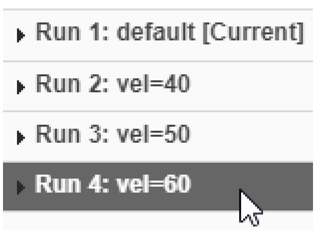
   b. In the Simulation Data Inspector, select **Import**. In the Import dialog box, accept the defaults and select **Import**.

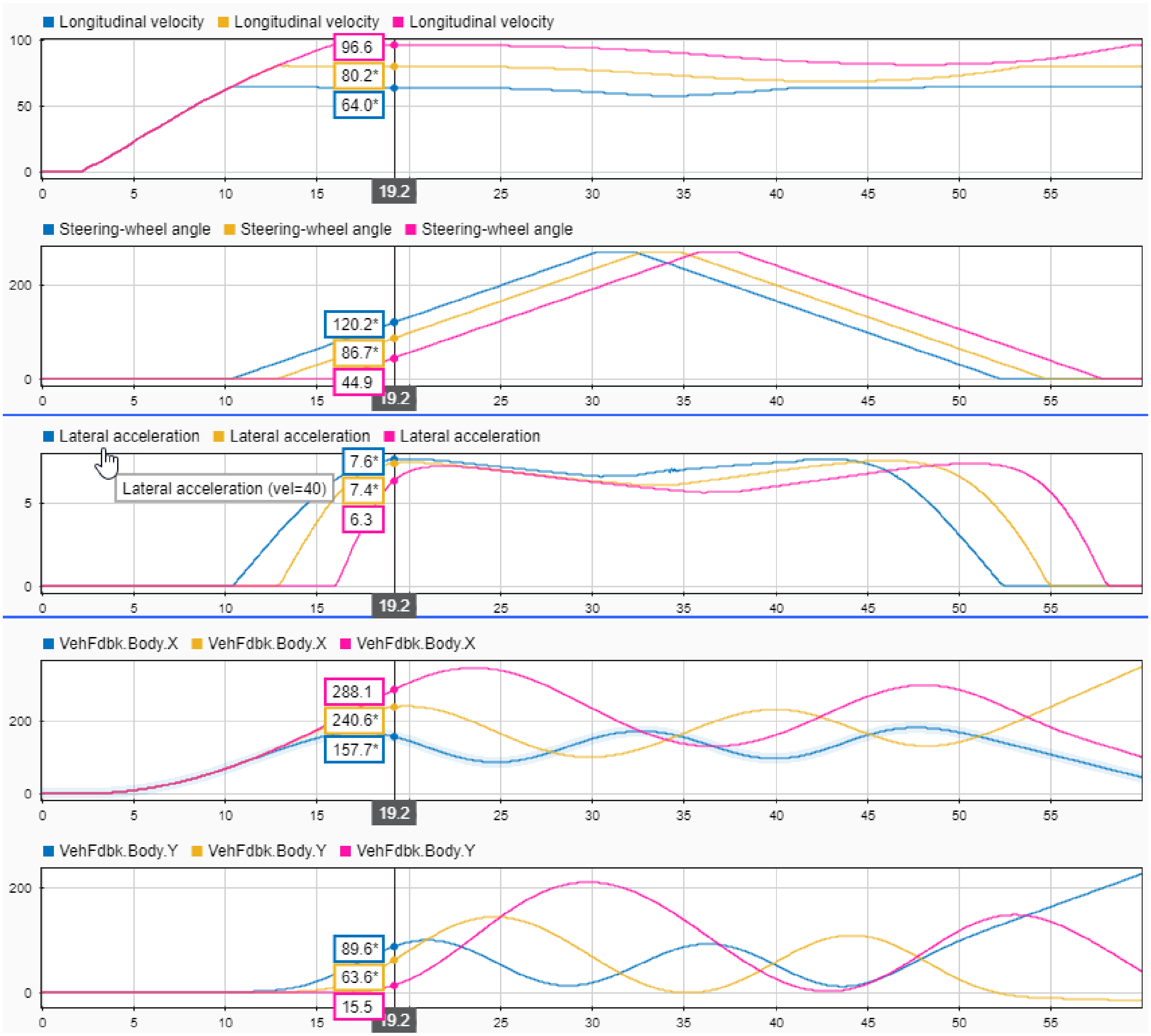
   c. In the Import dialog box, clear `logsout`. Select `simout(1)`, `simout(2)`, and `simout(3)`. Select **Import**.



d Select each of the runs. For each run, right-click to rename the results to the velocity that corresponds to the simulation. Run 1 corresponds to the simulation with the default settings.



7 Explore the results in the Simulation Data Inspector. To characterize the steering, view the plots of the simulation results. For example, plot longitudinal velocity, steering wheel angle, lateral acceleration, longitudinal position, X, and lateral position, Y. The results are similar to these plots, which show the results for runs 2, 3, and 4. The results indicate that the greatest lateral acceleration occurs when the vehicle velocity is `40` mph.



8 To explore the results further, use these commands to extract the lateral acceleration, steering angle, and vehicle trajectory from the `simout` object.

- Extract the lateral acceleration and steering angle. Plot the data. To calculate the steering gain, fit a first order polynomial to the data.

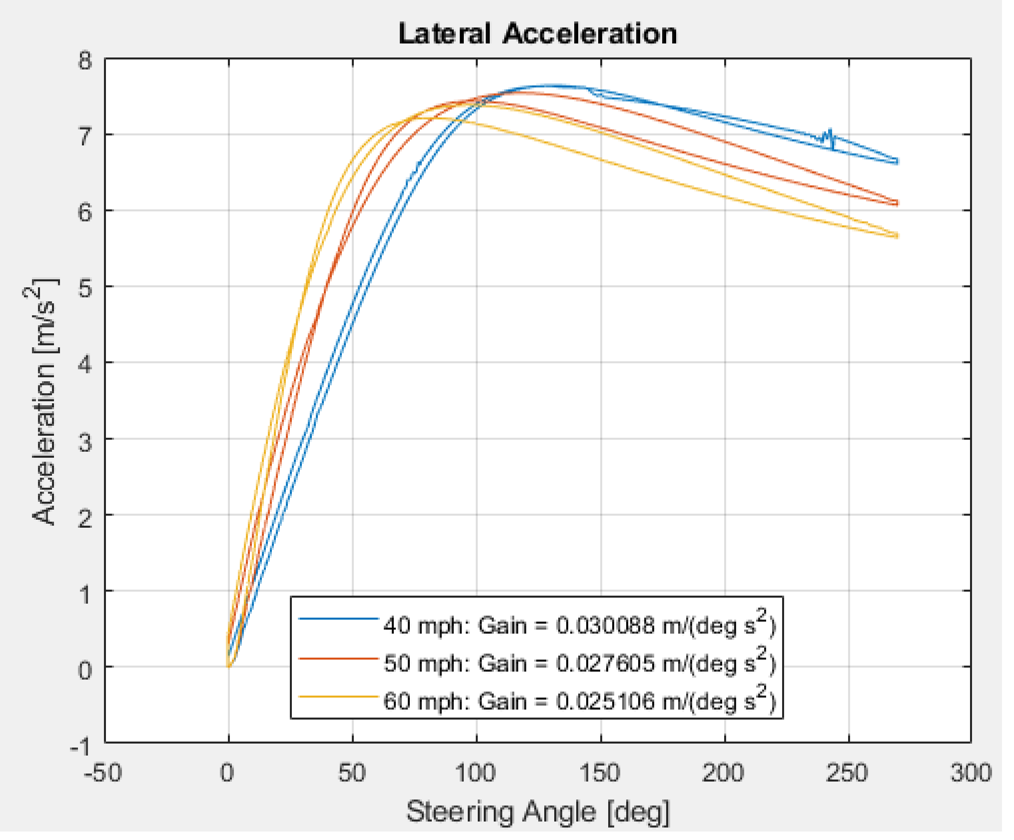
  ```
  % Plot results from simout object: lateral acceleration vs steering angle
  figure
  for idx = 1:numExperiments
      % Extract Data
      log = simout(idx).get('logsout');
      sa=log.get('Steering-wheel angle').Values;
      ay=log.get('Lateral acceleration').Values;

      firstorderfit = polyfit(sa.Data,ay.Data,1);
      gain(idx)=firstorderfit(1);

      legend_labels{idx} = [num2str(vmax(idx)), ' mph: Gain = ',num2str(gain(idx)), ' m/(deg s^2)'];

      % Plot steering angle vs. lateral acceleration
      plot(sa.Data,ay.Data)
      hold on
  end
  % Add labels to the plots
  legend(legend_labels, 'Location', 'best');
  title('Lateral Acceleration')
  xlabel('Steering Angle [deg]')
  ylabel('Acceleration [m/s^2]')
  grid on
  ```

  The results are similar to this plot.



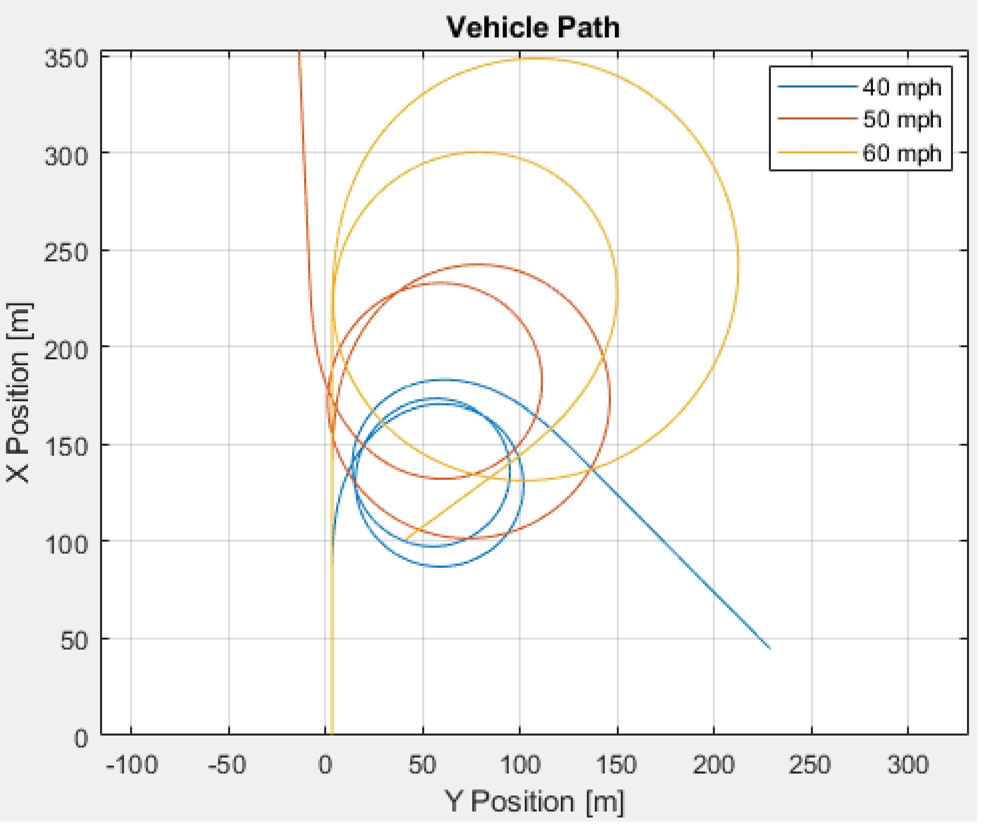
- Extract the vehicle path. Plot the data.

```
% Plot vehicle path
figure
for idx = 1:numExperiments
    % Extract Data
    log = simout(idx).get('logsout');
    VehFdbk = log.get('VehFdbk');
    x = VehFdbk.Values.Body.X;
    y = VehFdbk.Values.Body.Y;
    legend_labels{idx} = [num2str(vmax(idx)), ' mph'];

    % Plot vehicle location
    axis('equal')
    plot(y.Data,x.Data)
```

```
    hold on
end
% Add labels to the plots
legend(legend_labels, 'Location', 'best');
title('Vehicle Path')
xlabel('Y Position [m]')
ylabel('X Position [m]')
grid on
```

The results are similar to this plot.



References

[1] SAE J266. *Steady-State Directional Control Test Procedures For Passenger Cars and Light Trucks*. Warrendale, PA: SAE International, 1996.

See Also

`Simulink.SimulationInput` | `Simulink.SimulationOutput` | `polyfit`

More About

- “Slowly Increasing Steering Maneuver” on page 3-28
- “Vehicle Dynamics Blockset Communication with 3D Visualization Software” on page 1-6
- Simulation Data Inspector

Vehicle Lateral Acceleration at Different Speeds

This example shows how to use the vehicle dynamics constant radius reference application to analyze the impact of speed on the vehicle lateral dynamics. Specifically, you can examine the lateral acceleration when you run the maneuver with different speeds. For information about similar maneuvers, see standards SAE J266_199601[1] and ISO 4138:2012[2].

During the maneuver, the vehicle uses a predictive driver model to maintain a pre-specified turn radius at a set velocity.

For more information about the reference application, see "Constant Radius Maneuver" on page 3-37.

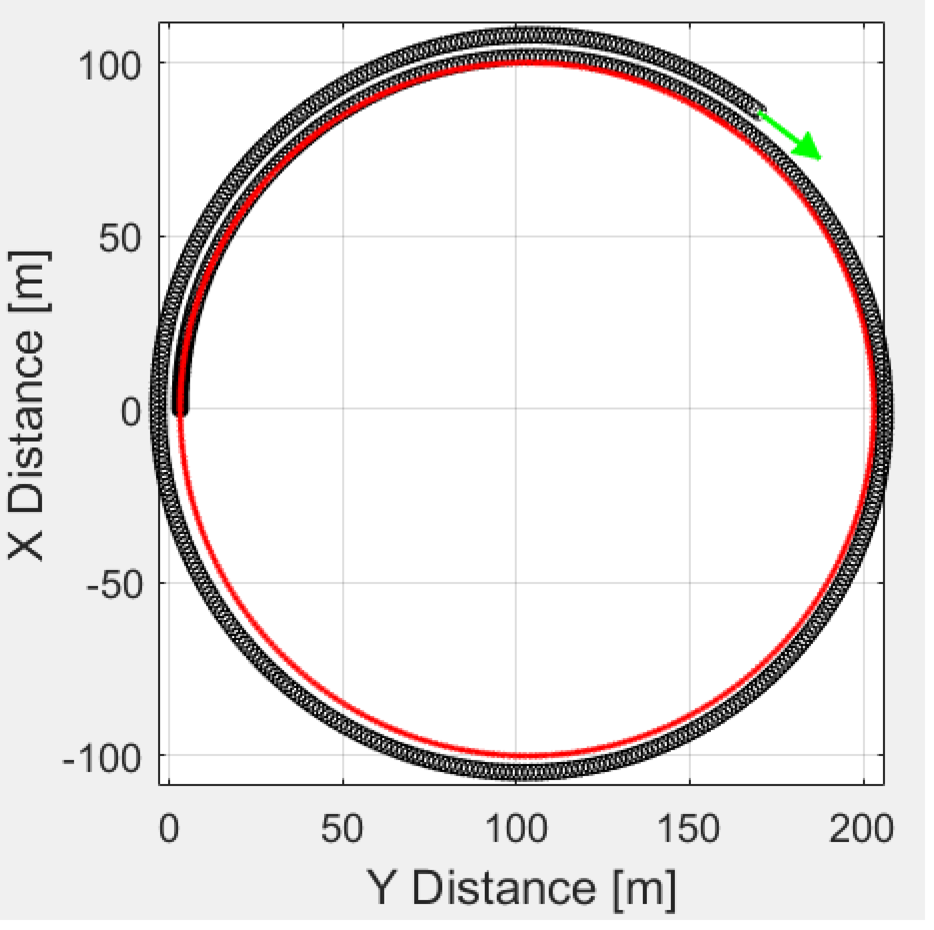
Run a Constant Radius Maneuver

1. Create and open a working copy of the constant radius reference application.

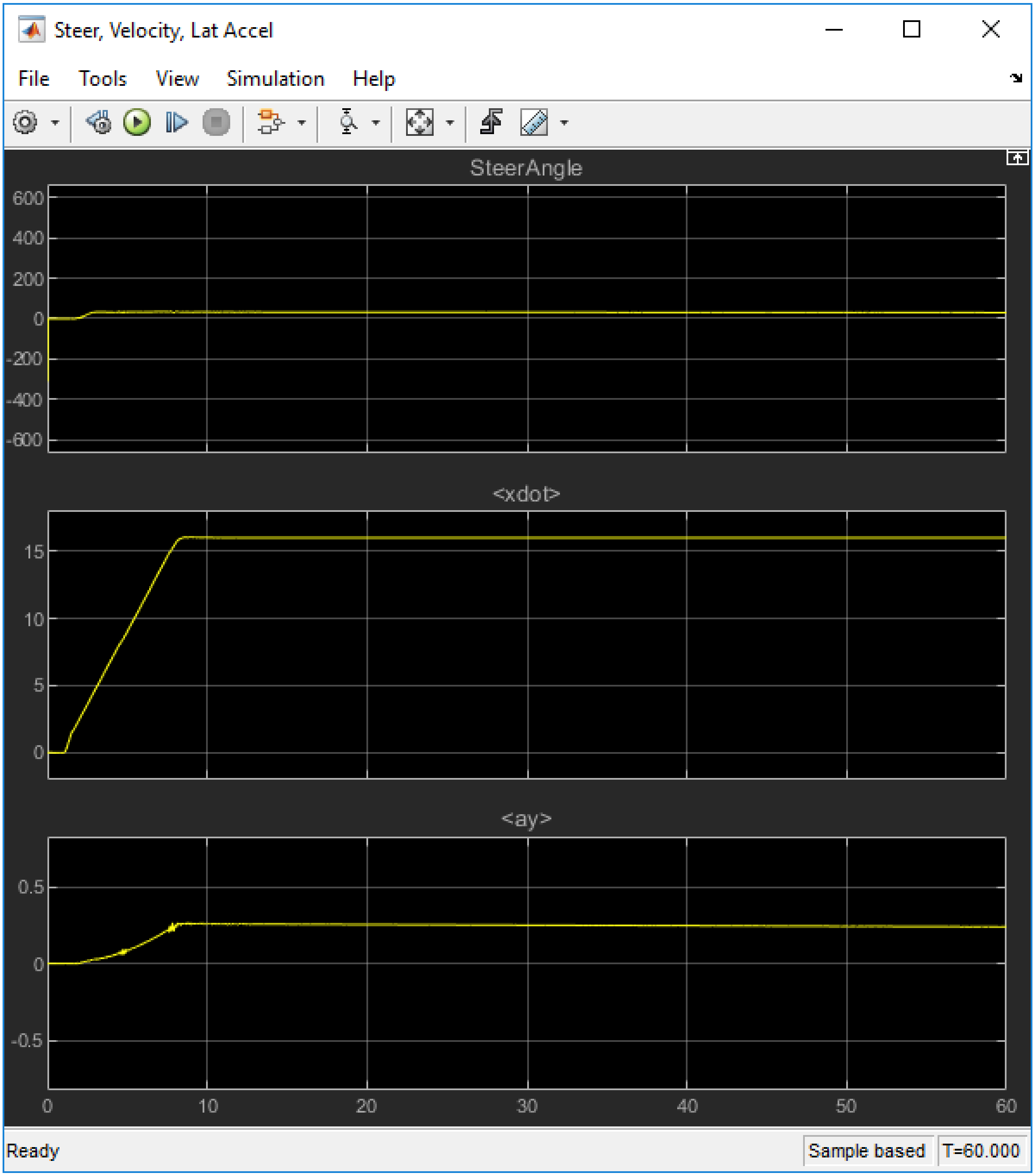
   ```
   vdynblksConstRadiusStart
   ```
2. Select the Reference Generator block. By default, the reference application uses the Predictive Driver block to maintain a 100 m right turn radius at 30 mph.

   - **Maneuver** — `Constant radius`
   - **Use maneuver-specific driver, initial position, and scene** — `on`
   - **Longitudinal velocity** — 30 mph
   - **Radius value** — 100 m
3. Select the Reference Generator block **3D Engine** tab. By default, the 3D Engine parameter is **Disabled**. For the 3D visualization engine platform requirements and hardware recommendations, see "3D Visualization Engine Requirements" on page 1-4.
4. Run the maneuver with the default settings. As the simulation runs, view vehicle information.

   - In the Vehicle Position window, view the vehicle longitudinal distance as a function of lateral distance.



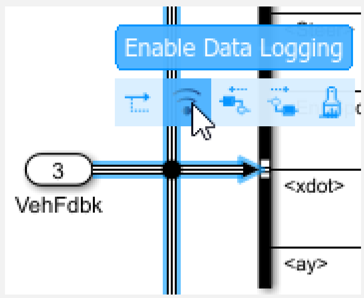
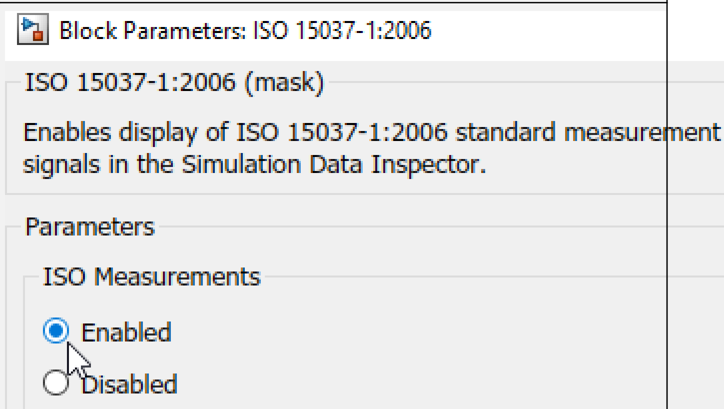
- In the Visualization subsystem, open the Steer, Velocity, Lat Accel Scope block to display the steering angle, velocity, and lateral acceleration versus time.

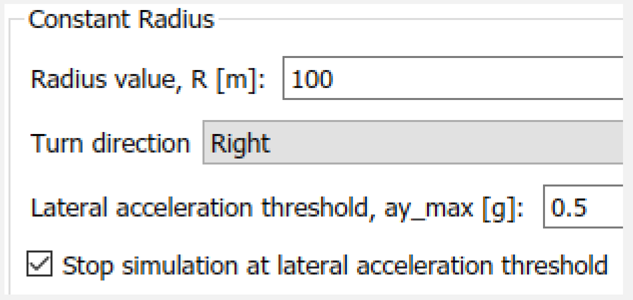

Steer, Velocity, Lat Accel
File
Tools
View
Simulation
Help
SteerAngle
600
400
200
0
-200
-400
-600
<xdot>
15
10
5
0
<ay>
0.5
0
-0.5
0
10
20
30
40
50
60
Ready
Sample based
T=60.000

Sweep Speed

Run the constant radius reference application with three different speeds. Stop the simulation if the vehicle exceeds a lateral acceleration threshold of .5 g.

1. In the constant radius reference application model `CRReferenceApplication`, open the Reference Generator block. The **Longitudinal velocity reference, xdot_r** block parameter sets the vehicle speed. By default, the speed is 30 mph.
2. Enable signal logging. Select the Reference Generator block **Stop simulation at lateral acceleration threshold** parameter. You can use the Simulink editor or, alternatively, MATLAB commands. Save the model.

Model Element	Simulink Editor
Visualization subsystem — VehFdbk inport	
Visualization subsystem — ISO 15037-1:2006 block	

Model Element	Simulink Editor
Reference Generator block	

Alternatively, use these commands to enable the signal logging and stop the simulation if the vehicle exceeds a lateral acceleration limit. Save the model.

```
% Open the model
mdl = 'CRReferenceApplication';
open_system(mdl);

% Enable signal logging for VehFdbk
ph=get_param('CRReferenceApplication/Visualization/VehFdbk','PortHandles');
set_param(ph.Outport,'DataLogging','on');

% Enable signal logging for ISO block
set_param([mdl '/Visualization/ISO 15037-1:2006'],'Measurement','Enable');

% Set parameter to stop simulation at lateral acceleration threshold
set_param([mdl '/Reference Generator'],'cr_ay_stop','on');
save_system(mdl)
```

3 Set up a speed set point vector, `xdot_r`, that you want to investigate. For example, at the command line, enter:

```
mdl = 'CRReferenceApplication';
open_system(mdl);
% Define the set of parameters to sweep
vmax = [35, 40, 45];
tfinal = [60, 60, 60];
numExperiments = length(vmax);
```

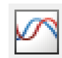
4 Create an array of simulation inputs that set `xdot_r` equal to the Reference Generator block parameter.

```
for idx = numExperiments:-1:1
    in(idx) = Simulink.SimulationInput(mdl);
    in(idx) = in(idx).setBlockParameter([mdl '/Reference Generator'], 'xdot_r', num2str(vmax(idx)));
    in(idx) = in(idx).setModelParameter('StopTime', num2str(tfinal(idx)));
end
```

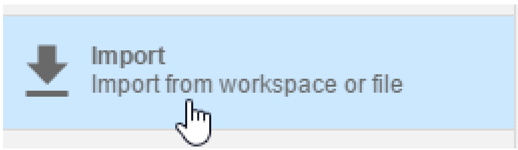
5 Save the model and run the simulations. If available, use parallel computing.

```
save_system(mdl);
tic;
simout = parsim(in,'ShowSimulationManager','on');
toc;
```

6 Import the simulation results to the Simulation Data Inspector.

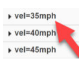
   a On the Simulink Editor toolbar, click the **Data Inspector** button .

   b In the Simulation Data Inspector, select **Import**. In the Import dialog box, accept the defaults and select **Import**.

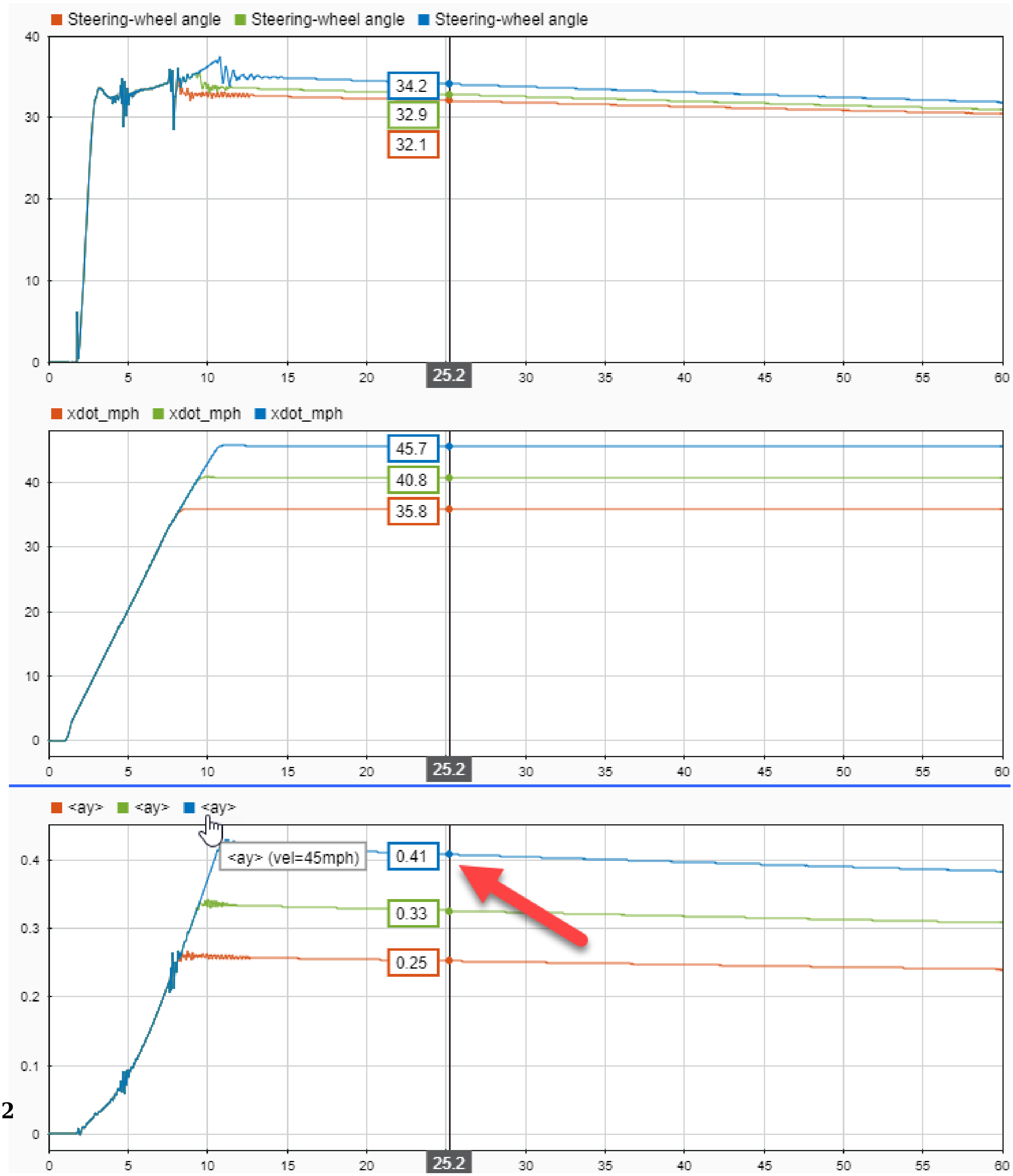


   c In the Import dialog box, clear `logsout`. Select `simout(1)`, `simout(2)`, and `simout(3)`. Select **Import**.

   d Select each of the runs. For each run, right-click to rename the results to the velocity that corresponds to the simulation.



7 Explore the results in the Simulation Data Inspector. To characterize the lateral acceleration and steering, view the plots of the simulation results. For example, plot longitudinal velocity, lateral acceleration, and the steering wheel angle. The results are similar to these plots, which show the results for the three runs. The results indicate that the greatest lateral acceleration occurs when the vehicle velocity is 45 mph.


Steering-wheel angle
Steering-wheel angle
Steering-wheel angle
34.2
32.9
32.1
25.2
xdot_mph
xdot_mph
xdot_mph
45.7
40.8
35.8
25.2
<ay>
<ay>
<ay>
<ay> (vel=45mph)
0.41
0.33
0.25
25.2

8 To explore the results further, use these commands to extract the lateral acceleration, steering angle, and vehicle trajectory from the `simout` object.

- Extract the lateral acceleration and steering angle. Plot the data. To calculate the steering gain, fit a first order polynomial to the data.

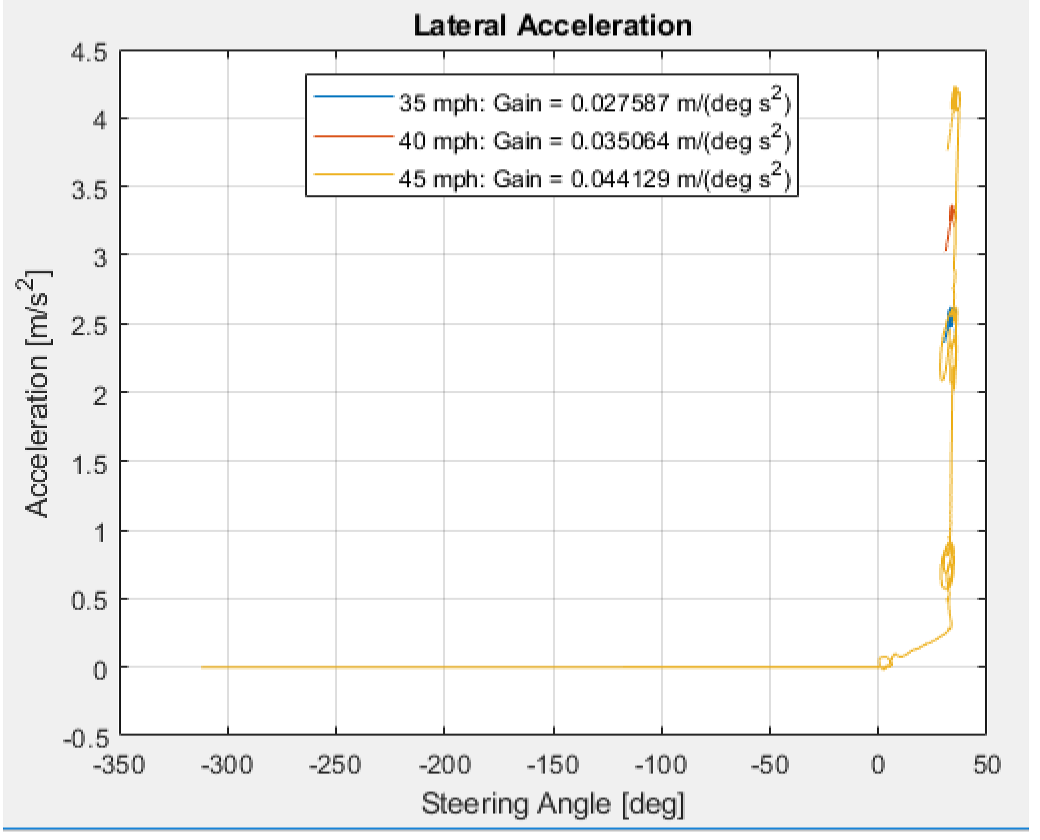
```
% Plot results from simout object: lateral acceleration vs steering angle
figure
for idx = 1:numExperiments
    % Extract Data
    log = simout(idx).get('logsout');
    sa=log.get('Steering-wheel angle').Values;
    ay=log.get('Lateral acceleration').Values;

    firstorderfit = polyfit(sa.Data,ay.Data,1);
    gain(idx)=firstorderfit(1);

    legend_labels{idx} = [num2str(vmax(idx)), ' mph: Gain = ',num2str(gain(idx)), ' m/(deg s^2)'];

    % Plot steering angle vs. lateral acceleration
    plot(sa.Data,ay.Data)
    hold on
end
% Add labels to the plots
legend(legend_labels, 'Location', 'best');
title('Lateral Acceleration');
xlabel('Steering Angle [deg]');
ylabel('Acceleration [m/s^2]');
grid on;
```

  The results are similar to this plot.



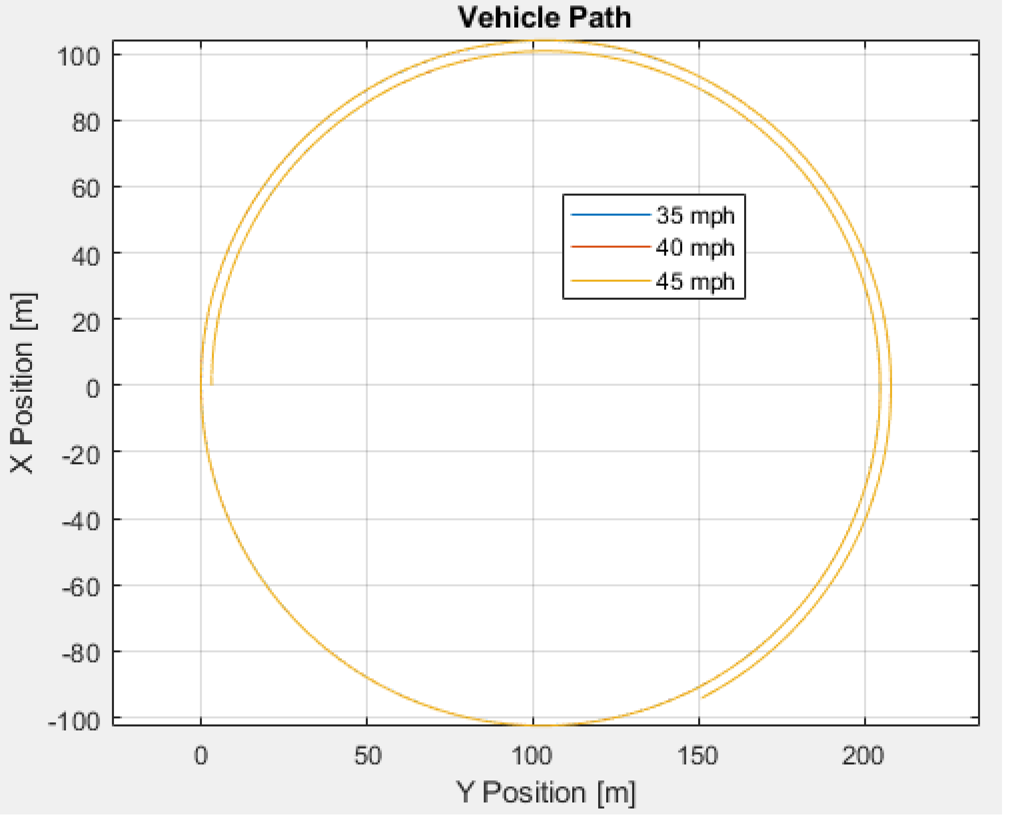
- Extract the vehicle path. Plot the data.

```
% Plot vehicle path
figure
for idx = 1:numExperiments
    % Extract Data
    log = simout(idx).get('logsout');
    VehFdbk = log.get('VehFdbk');
    x = VehFdbk.Values.Body.X;
    y = VehFdbk.Values.Body.Y;
    legend_labels{idx} = [num2str(vmax(idx)), ' mph'];

    % Plot vehicle location
    axis('equal')
    plot(y.Data,x.Data)
```

```
    hold on
end
% Add labels to the plots
legend(legend_labels, 'Location', 'best');
title('Vehicle Path')
xlabel('Y Position [m]')
ylabel('X Position [m]')
grid on
```

The results are similar to this plot.



References

[1] J266_199601. *Steady-State Directional Control Test Procedures for Passenger Cars and Light Trucks*. Warrendale, PA: SAE International, 1996.

[2] ISO 4138:2012. *Passenger cars -- Steady-state circular driving behaviour -- Open-loop test methods*. ISO (International Organization for Standardization), 2012.

See Also

`Simulink.SimulationInput` | `Simulink.SimulationOutput` | `polyfit`

More About

- "Constant Radius Maneuver" on page 3-37
- "Vehicle Dynamics Blockset Communication with 3D Visualization Software" on page 1-6
- Simulation Data Inspector

Frequency Response to Steering Angle Input

This example shows how to use the vehicle dynamics swept-sine steering reference application to analyze the dynamic steering response to steering inputs. Specifically, you can examine the vehicle frequency response and lateral acceleration when you run the maneuver with different sinusoidal wave steering amplitudes.

The swept-sine steering maneuver tests the vehicle frequency response to steering inputs. In the test, the driver:

- Accelerates until the vehicle hits a target velocity.
- Commands a sinusoidal steering wheel input.
- Linearly increase the frequency of the sinusoidal wave.

For more information about the reference application, see “Swept-Sine Steering Maneuver” on page 3-19.

Run a Swept-Sine Steering Maneuver

1 Create and open a working copy of the increasing steering reference application.

```
vdynblksSweptSineSteeringStart
```

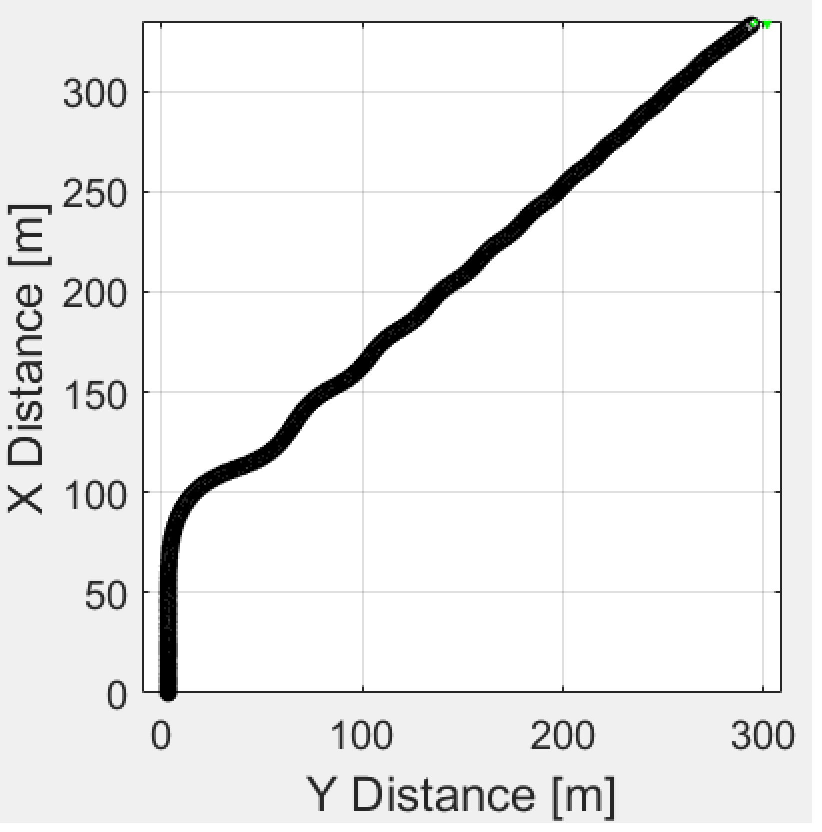
2 Open the Swept Sine Reference Generator block. By default, the maneuver is set with these parameters:

   - **Longitudinal velocity setpoint** — 30 mph
   - **Steering amplitude** — 90 deg
   - **Final frequency** — 0.7 Hz

3 Open the Visualization subsystem. By default, the 3D Engine is set with the 3D visualization engine disabled. For the 3D visualization engine platform requirements and hardware recommendations, see “3D Visualization Engine Requirements” on page 1-4.

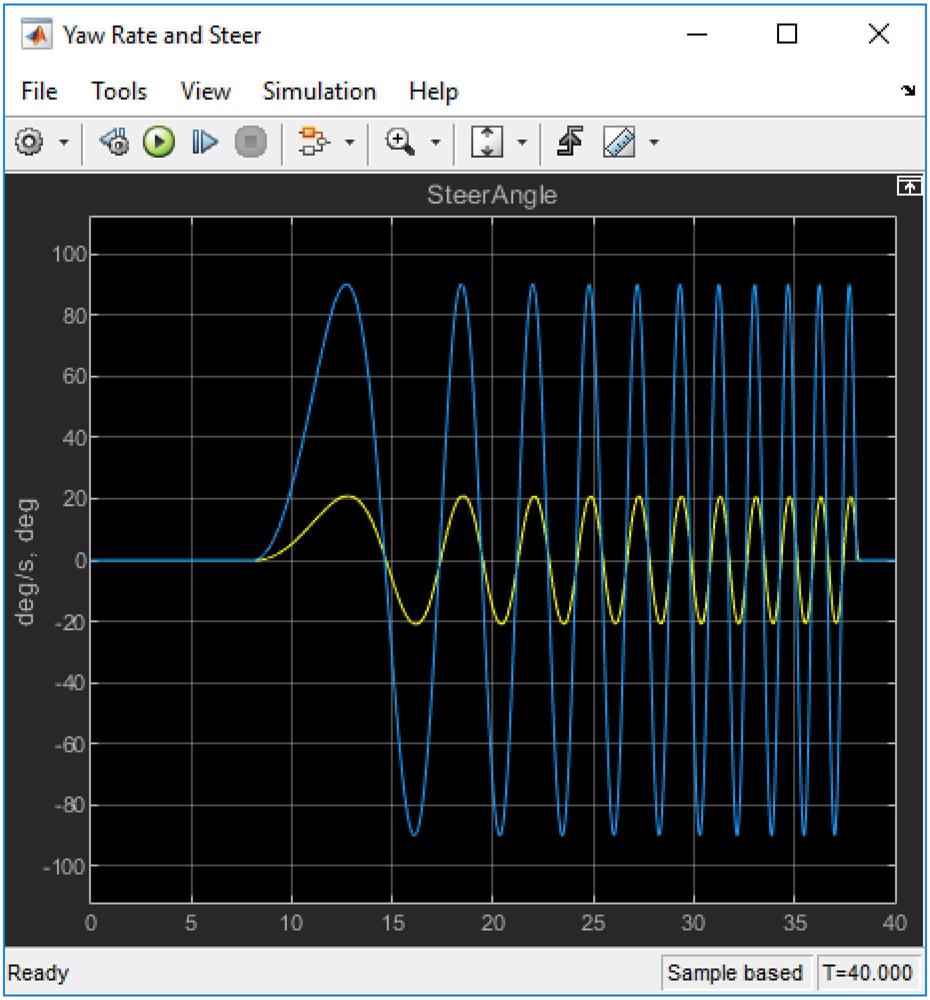
4 Run the maneuver with the default settings. As the simulation runs, view vehicle information.

   - In the Vehicle Position window, view the vehicle longitudinal distance as a function of lateral distance.



- In the Visualization subsystem, open the Yaw Rate and Steer Scope block to display the yaw rate and steering angle versus time:
  - Yellow line — Yaw rate
  - Blue lines — Steering angle

  The blue line shows a `90` deg amplitude sinusoidal steering angle with an increasing frequency.

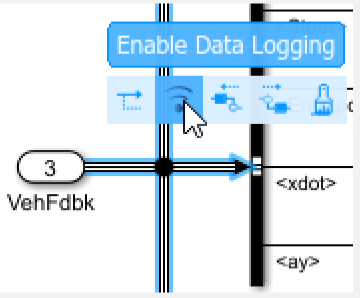
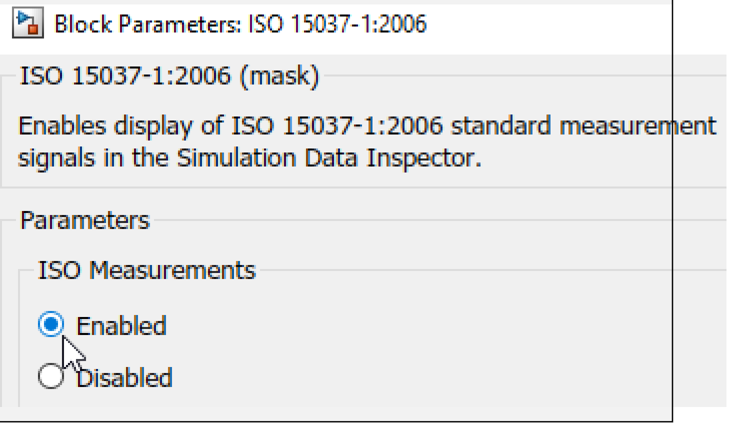


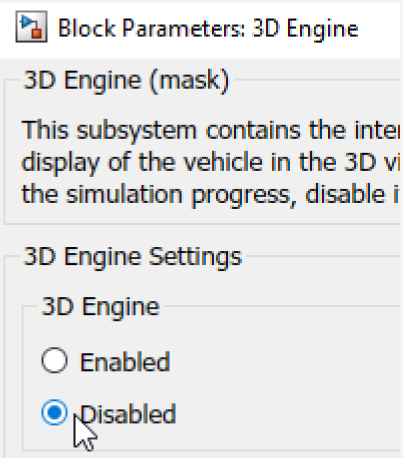
Sweep Steering

Run the reference application with three different sinusoidal wave steering amplitudes.

1 In the swept-sine steering reference application model `SSSReferenceApplication`, open the Swept Sine Reference Generator block. The **Steering amplitude, theta_hw** block parameter sets the amplitude. By default, the amplitude is `90` deg.

2 In the Visualization subsystem, enable signal logging for these model elements. Disable the 3D visualization environment. You can use the Simulink editor or, alternatively, MATLAB commands. Save the model.

Model Element	Simulink Editor
VehFdbk inport	
ISO 15037-1:2006 block	

Model Element	Simulink Editor
3D Engine block	

Alternatively, use these commands to enable the signal logging, disable the 3D visualization environment, and save the model.

```
% Open the model
mdl = 'SSSReferenceApplication';
open_system(mdl);

% Enable signal logging for VehFdbk
ph=get_param('SSSReferenceApplication/Visualization/VehFdbk','PortHandles');
set_param(ph.Outport,'DataLogging','on');

% Enable signal logging for ISO block
set_param([mdl '/Visualization/ISO 15037-1:2006'],'Measurement','Enable');

% Disable 3D environment
set_param([mdl '/Visualization/3D Engine'],'engine3D','Disabled');

save_system(mdl)
```

3. Set up a steering amplitude vector, `amp`, that you want to investigate. For example, at the command line, type:

```
mdl = 'SSSReferenceApplication';
open_system(mdl);
% Define the set of amplitudes to sweep
```

```
amp = [60, 90, 120];
numExperiments = length(amp);
```

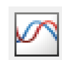
4 Create an array of simulation inputs that set the Swept Sine Reference Generator block parameter **Steering amplitude, theta_hw** equal `amp`.

```
for idx = numExperiments:-1:1
    in(idx) = Simulink.SimulationInput(mdl);
    in(idx) = in(idx).setBlockParameter([mdl '/Swept Sine Reference Generator'],'th
end
```

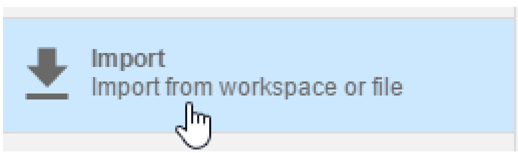
5 Save the model and run the simulations. If available, use parallel computing.

```
tic;
simout = parsim(in,'ShowSimulationManager','on');
toc;
```

6 Import the simulation results to the Simulation Data Inspector.

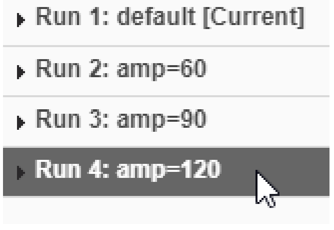
   a On the Simulink Editor toolbar, click the **Data Inspector** button .

   b In the Simulation Data Inspector, select **Import**. In the Import dialog box, accept the defaults and select **Import**.

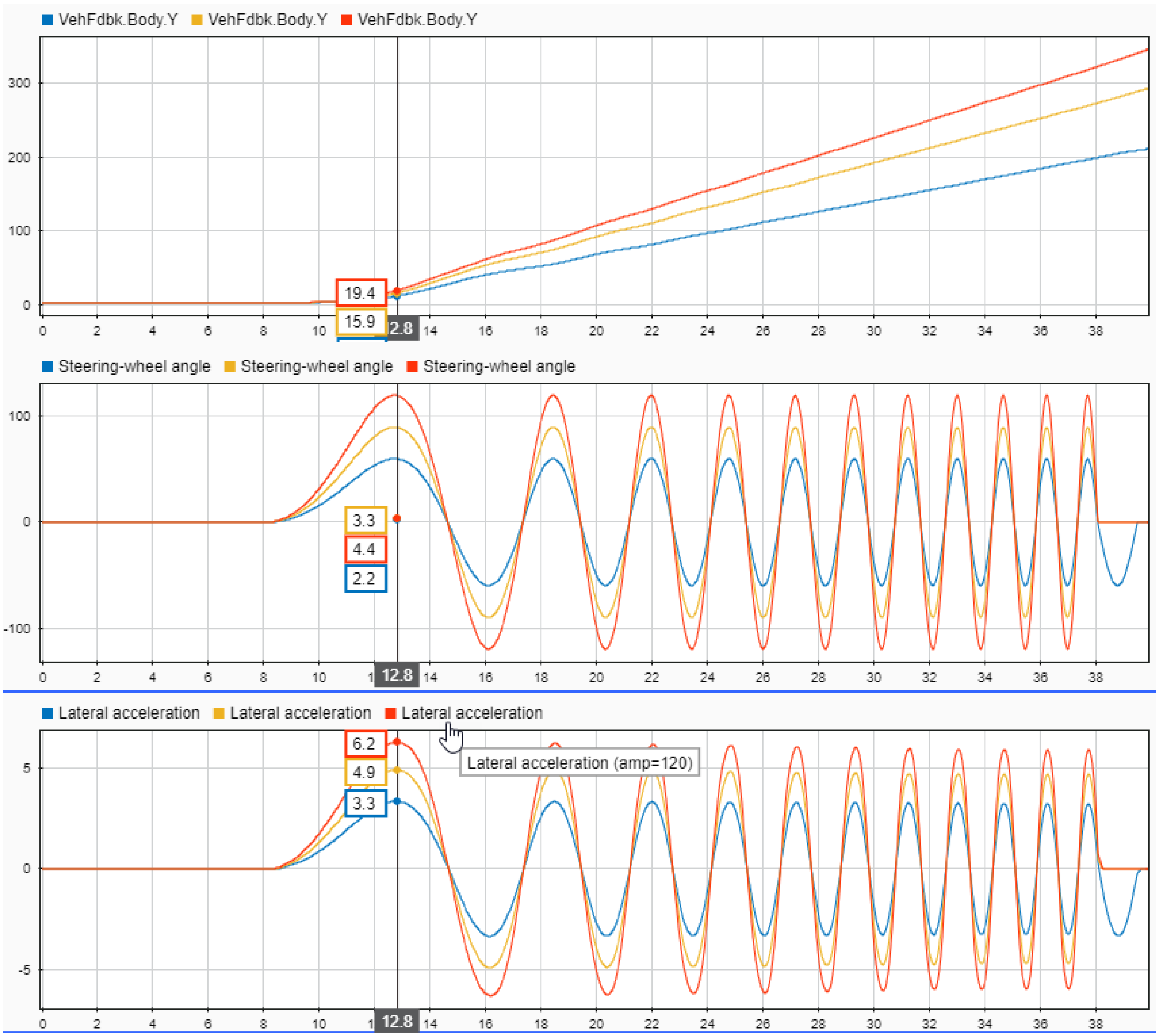


   c In the Import dialog box, clear `logsout`. Select `simout(1)`, `simout(2)`, and `simout(3)`. Select **Import**.

d Select each of the runs. For each run, right-click to rename the results to the amplitude that corresponds to the simulation. Run 1 corresponds to the simulation with the default settings.



7 Explore the results in the Simulation Data Inspector. To characterize the steering, view the plots of the simulation results. For example, plot lateral position, `Y`, steering wheel angle, and lateral acceleration. The results are similar to these plots, which show the results for runs 2, 3, and 4. The results indicate that the greatest lateral acceleration occurs when the steering amplitude is `120` deg.



8 To explore the results further, use these commands to extract the lateral acceleration, steering angle, and vehicle trajectory from the `simout` object.

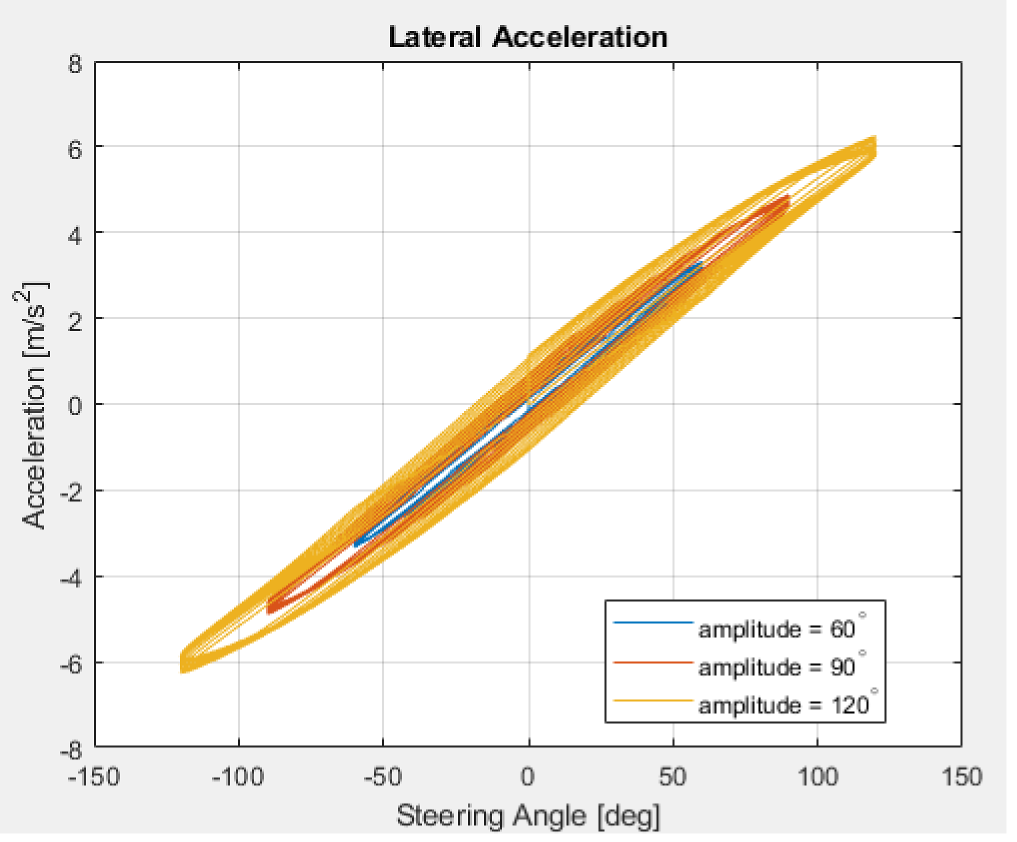
- Extract the lateral acceleration and steering angle. Plot the data.

```
% Plot results from simout object: lateral acceleration vs steering angle
figure
for idx = 1:numExperiments
    % Extract Data
    log = simout(idx).get('logsout');
    sa=log.get('Steering-wheel angle').Values;
    ay=log.get('Lateral acceleration').Values;

    legend_labels{idx} = ['amplitude = ', num2str(amp(idx)), '^{\circ}'];

    % Plot steering angle vs. lateral acceleration
    plot(sa.Data,ay.Data)
    hold on
end
% Add labels to the plots
legend(legend_labels, 'Location', 'best');
title('Lateral Acceleration')
xlabel('Steering Angle [deg]')
ylabel('Acceleration [m/s^2]')
grid on
```

The results are similar to this plot.



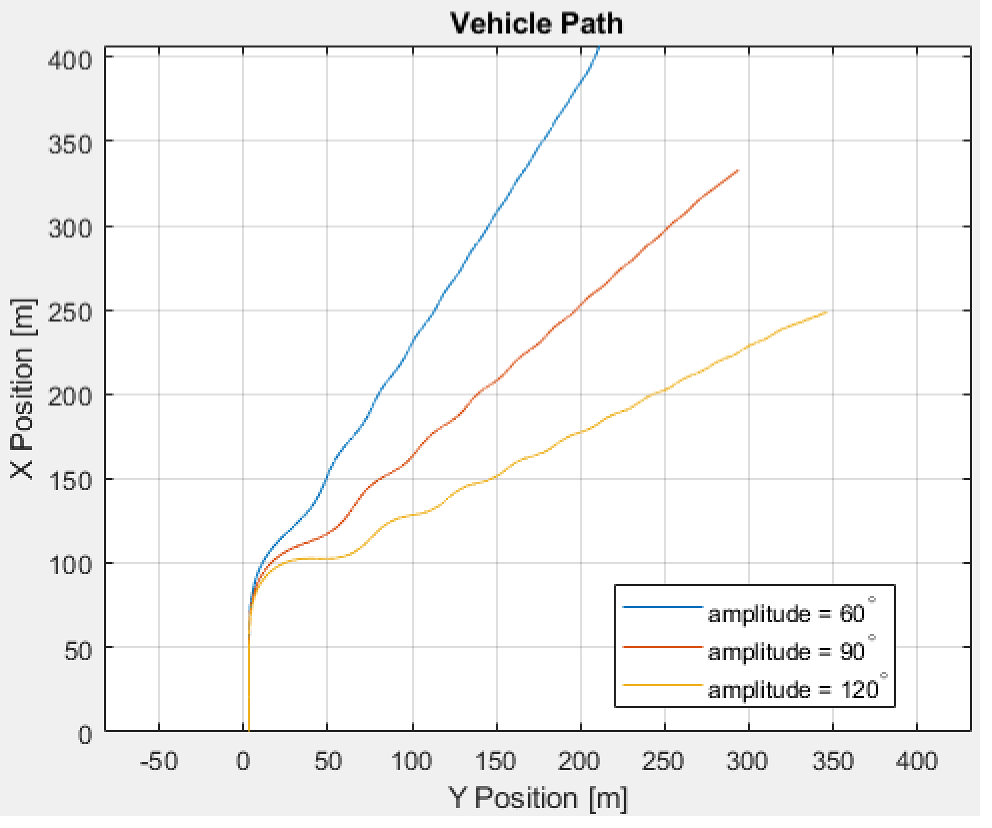
- Extract the vehicle path. Plot the data.

```
% Plot results from simout object
figure
for idx = 1:numExperiments
    % Extract Data
    log = simout(idx).get('logsout');
    VehFdbk = log.get('VehFdbk');
    x = VehFdbk.Values.Body.X;
    y = VehFdbk.Values.Body.Y;
    legend_labels{idx} = ['amplitude = ', num2str(amp(idx)), '^{\circ}'];

    % Plot vehicle location
    axis('equal')
    plot(y.Data,x.Data)
```

```
    hold on
end
% Add labels to the plots
legend(legend_labels, 'Location', 'best');
title('Vehicle Path')
xlabel('Y Position [m]')
ylabel('X Position [m]')
grid on
```

The results are similar to this plot.



9 For the next steps, use a fast Fourier transform (FFT) to examine the steering response in the frequency domain.

See Also

`Simulink.SimulationInput` | `Simulink.SimulationOutput` | `fft`

More About

- “Fourier Analysis and Filtering” (MATLAB)
- Simulation Data Inspector
- “Swept-Sine Steering Maneuver” on page 3-19
- “Vehicle Dynamics Blockset Communication with 3D Visualization Software” on page 1-6

2

Coordinate Systems

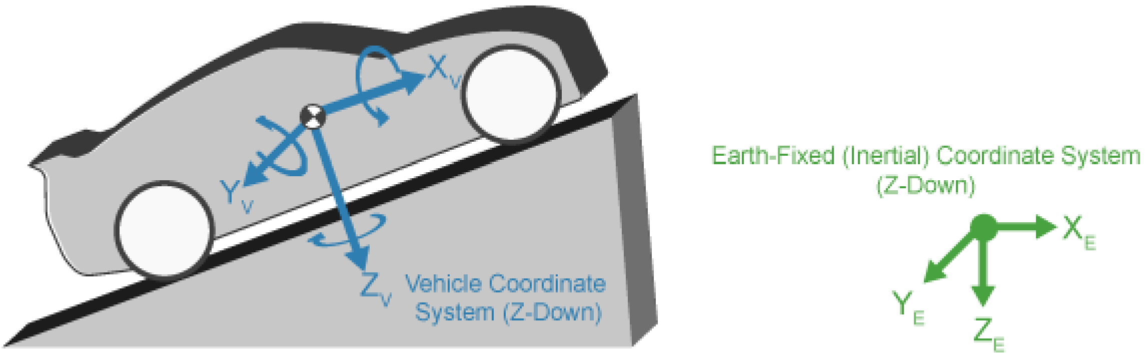
Coordinate Systems in Vehicle Dynamics Blockset

Vehicle Dynamics Blockset uses these coordinate systems to calculate the vehicle dynamics and position objects in the 3D visualization environment.

Environment	Description	Coordinate Systems
Vehicle dynamics in Simulink	The *right-hand rule* establishes the *X*-*Y*-*Z* sequence and rotation of the coordinate axes used to calculate the vehicle dynamics. The Vehicle Dynamics Blockset 3D simulation environment uses these *right-handed* (RH) *Cartesian* coordinate systems defined in the SAE J670[2] and ISO 8855[3] standards: • Earth-fixed (inertial) • Vehicle • Tire • Wheel The coordinate systems can have either orientation: • Z-down — Defined in SAE J670[2] • Z-up — Defined in SAE J670[2] and ISO 8855[3]	"Earth-Fixed (Inertial) Coordinate System" on page 2-2 "Vehicle Coordinate System" on page 2-3 "Tire and Wheel Coordinate Systems" on page 2-4
3D visualization engine	To position objects and query the 3D visualization environment, the Vehicle Dynamics Blockset uses a world coordinate system.	"World Coordinate System" on page 2-6

Earth-Fixed (Inertial) Coordinate System

The earth-fixed coordinate system (X_E, Y_E, Z_E) axes are fixed in an inertial reference frame. The inertial reference frame has zero linear and angular acceleration and zero angular velocity. In Newtonian physics, the earth is an inertial reference.

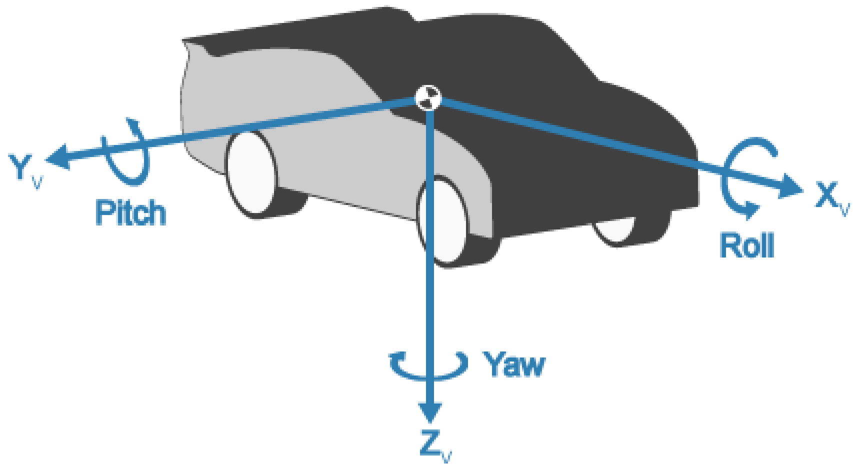


Axis	Description
X_E	The X_E axis is in the forward direction of the vehicle.
Y_E	The X_E and Y_E axes are parallel to the ground plane. The ground plane is a horizontal plane normal to the gravitational vector.
Z_E	In the Z-up orientation, the positive Z_E axis points upward. In the Z-down orientation, the positive Z_E axis points downward.

Vehicle Coordinate System

The vehicle coordinate system axes (X_V, Y_V, Z_V) are fixed in a reference frame attached to the vehicle. The origin is at the vehicle sprung mass.

Z-Down Orientation



<table>
<tr><th>Axis</th><th>Description</th></tr>
<tr><td>X_V</td><td>The X_V axis points forward and is parallel to the vehicle plane of symmetry.</td></tr>
<tr><td>Y_V</td><td rowspan="2">The Y_V axis is perpendicular to the vehicle plane of symmetry.

In the Z-down orientation:

• Y_V axis points to the right
• Z_V axis points downward</td></tr>
<tr><td>Z_V</td></tr>
</table>

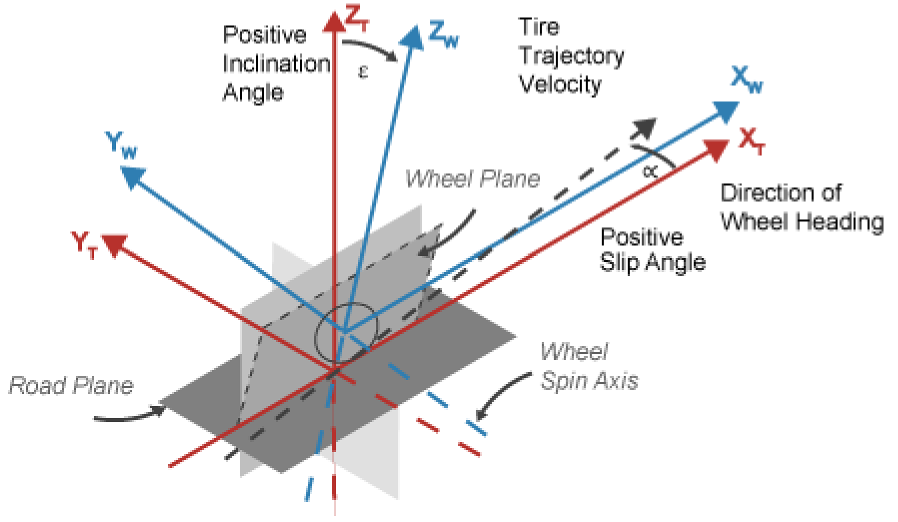
Tire and Wheel Coordinate Systems

The tire coordinate system axes (X_T, Y_T, Z_T) are fixed in a reference frame attached to the tire. The origin is at the tire contact with the ground.

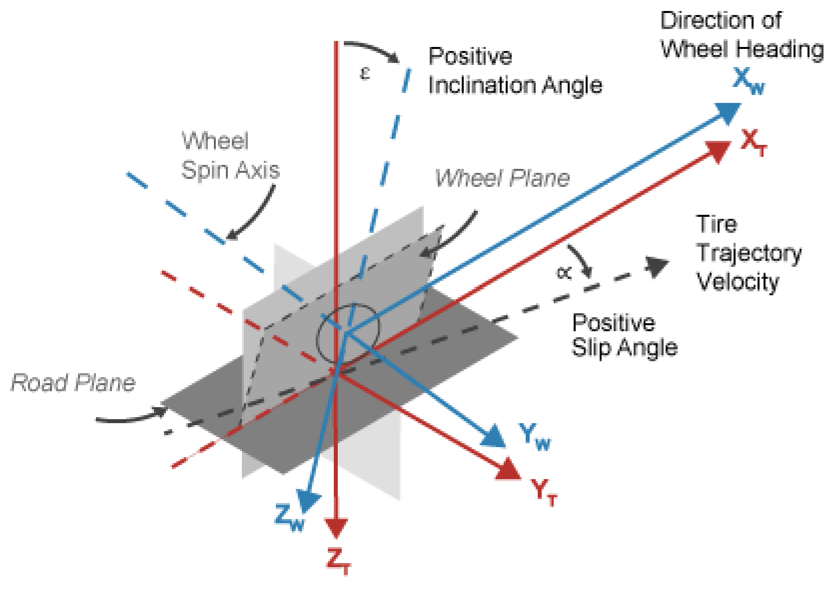
The wheel coordinate system axes (X_W, Y_W, Z_W) are fixed in a reference frame attached to the wheel. The origin is at the wheel center.

Z-Up Orientation[1]

1. Reprinted with permission Copyright © 2008 SAE International. Further distribution of this material is not permitted without prior permission from SAE.



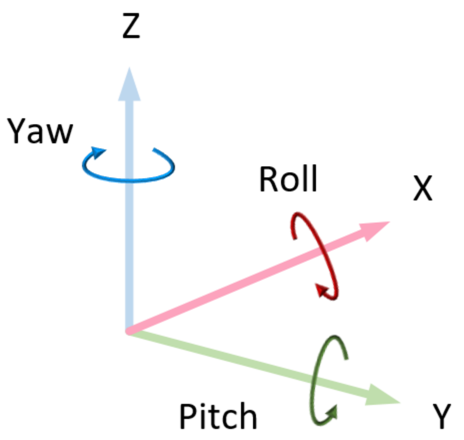
Z-Down Orientation



Axis	Description
X_T	X_T and Y_T are parallel to the road plane. The intersection of the wheel plane and the road plane define the orientation of the X_T axis.
Y_T	
Z_T	Z_T points: • Upward in the Z-up orientation • Downward in the Z-down orientation
X_W	X_W and Y_W are parallel to the wheel plane: • X_W is parallel to the local road plane. • Y_W is parallel to the wheel-spin axis.
Y_W	
Z_W	Z_W points: • Upward in the Z-up orientation • Downward in the Z-down orientation

World Coordinate System

The 3D visualization environment uses a world coordinate system with axes that are fixed in the inertial reference frame.



Axis	Description
X	Forward direction of the vehicle Roll — Right-handed rotation about *X*-axis
Y	Extends to the right of the vehicle, parallel to the ground plane Pitch — Right-handed rotation about *Y*-axis
Z	Extends upwards Yaw — Left-handed rotation about *Z*-axis

References

[1] Gillespie, Thomas. *Fundamentals of Vehicle Dynamics*. Warrendale, PA: Society of Automotive Engineers, 1992.

[2] Vehicle Dynamics Standards Committee. *Vehicle Dynamics Terminology*. SAE J670. Warrendale, PA: Society of Automotive Engineers, 2008.

[3] Technical Committee. *Road vehicles — Vehicle dynamics and road-holding ability — Vocabulary*. ISO 8855:2011. Geneva, Switzerland: International Organization for Standardization, 2011.

See Also

Combined Slip Wheel | Longitudinal Wheel | Simulation 3D Actor Transform Get | Simulation 3D Actor Transform Set | Simulation 3D Camera Get | Simulation 3D Scene Configuration | Vehicle Body 3DOF | Vehicle Body 6DOF | Vehicle Terrain Sensor

External Websites

- SAE International Standards
- ISO Standards

3

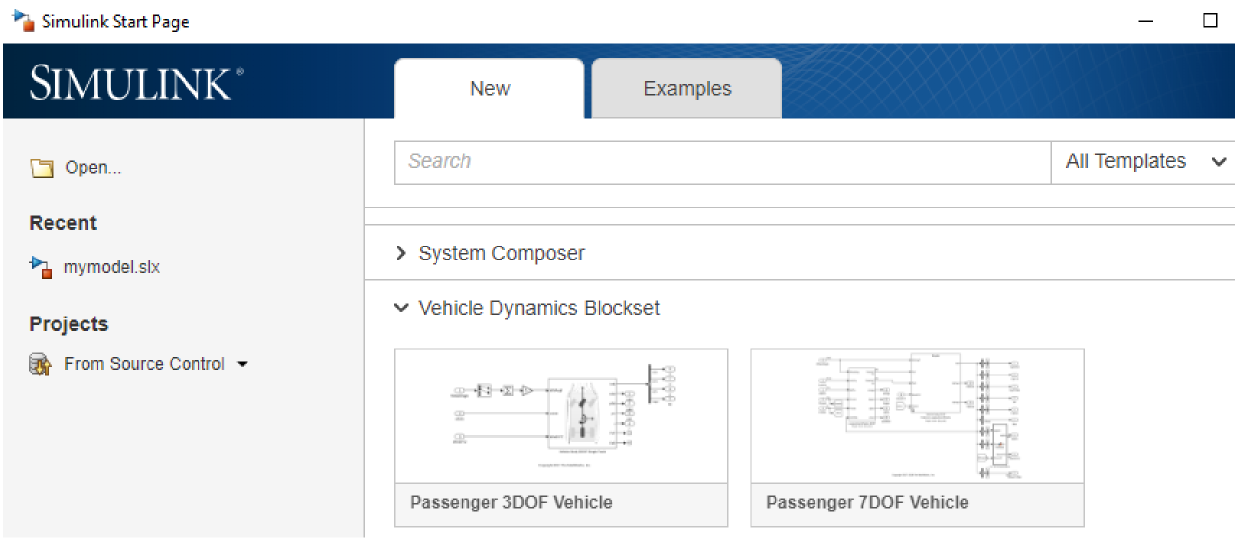
Reference Applications

Passenger Vehicle Dynamics Models

To analyze the dynamic system response in common ride and handling maneuvers, Vehicle Dynamics Blockset provides these pre-assembled vehicle dynamics models.

<table>
<tr><th>Vehicle Model</th><th>Description</th><th>Vehicle Body Degrees-of-Freedom (DOFs)</th><th>Wheel DOFs</th></tr>
<tr>
<td>Passenger 14DOF Vehicle</td>
<td>Vehicle with four wheels
Available as model variant in the maneuver reference applications</td>
<td>Six
<table>
<tr><th colspan="2">Translational</th><th colspan="2">Rotational</th></tr>
<tr><td>Longitudinal</td><td>✓</td><td>Pitch</td><td>✓</td></tr>
<tr><td>Lateral</td><td>✓</td><td>Yaw</td><td>✓</td></tr>
<tr><td>Vertical</td><td>✓</td><td>Roll</td><td>✓</td></tr>
</table></td>
<td>Two per wheel - eight total
<table>
<tr><th colspan="2">Translational</th><th colspan="2">Rotational</th></tr>
<tr><td>Vertical</td><td>✓</td><td>Rolling</td><td>✓</td></tr>
</table></td>
</tr>
<tr>
<td>Passenger 7DOF Vehicle</td>
<td>Vehicle with four wheels
Available as model variant in the maneuver reference applications</td>
<td>Three
<table>
<tr><th colspan="2">Translational</th><th colspan="2">Rotational</th></tr>
<tr><td>Longitudinal</td><td>✓</td><td>Pitch</td><td></td></tr>
<tr><td>Lateral</td><td>✓</td><td>Yaw</td><td>✓</td></tr>
<tr><td>Vertical</td><td></td><td>Roll</td><td></td></tr>
</table></td>
<td>One per wheel - four total
<table>
<tr><th colspan="2">Rotational</th></tr>
<tr><td>Rolling</td><td>✓</td></tr>
</table></td>
</tr>
<tr>
<td>Passenger 3DOF Vehicle</td>
<td>Vehicle with ideal tire</td>
<td>Three
<table>
<tr><th colspan="2">Translational</th><th colspan="2">Rotational</th></tr>
<tr><td>Longitudinal</td><td>✓</td><td>Pitch</td><td></td></tr>
<tr><td>Lateral</td><td>✓</td><td>Yaw</td><td>✓</td></tr>
<tr><td>Vertical</td><td></td><td>Roll</td><td></td></tr>
</table></td>
<td>None</td>
</tr>
</table>

From the Simulink start page, you can open project files that contain the vehicle models.



See Also

Vehicle Body 3DOF | Vehicle Body 6DOF

More About

- "Coordinate Systems in Vehicle Dynamics Blockset" on page 2-2
- "Vehicle Reference Applications"

Double-Lane Change Maneuver

This reference application represents a full vehicle dynamics model undergoing a double-lane change maneuver according to standard ISO 3888-2[1]. You can create your own versions, establishing a framework to test that your vehicle meets the design requirements under normal and extreme driving conditions. Use the reference application to analyze vehicle ride and handling and develop chassis controls. To perform vehicle studies, including yaw stability and lateral acceleration limits, use this reference application.

ISO 3888-2[1] defines the double-lane change maneuver to test the obstacle avoidance performance of a vehicle. In the test, the driver:

- Accelerates until vehicle hits a target velocity
- Releases the accelerator pedal
- Turns steering wheel to follow path into the left lane
- Turns steering wheel to follow path back into the right lane

Typically, cones mark the lane boundaries. If the vehicle and driver can negotiate the maneuver without hitting a cone, the vehicle passes the test.

To test advanced driver assistance systems (ADAS) and automated driving (AD) perception, planning, and control software, you can run the maneuver in a 3D environment. For the 3D visualization engine platform requirements and hardware recommendations, see “3D Visualization Engine Requirements” on page 1-4.

To create and open a working copy of the double-lane change reference application project, enter

```
vdynblksDblLaneChangeStart
```

This table summarizes the blocks and subsystems in the reference application. Some subsystems contain variants.

Reference Application Element	Description	Variants
Lane Change Reference Generator	Generates lane signals for the visualization subsystem and trajectory signals for the Predictive Driver block	

Reference Application Element	Description	Variants
Predictive Driver	Generates normalized steering, acceleration, and braking commands that track the reference trajectory	
Environment	Implements wind and ground forces	✓
Controllers	Implements controllers for engine control units (ECUs), transmissions, and brakes	✓
Passenger Vehicle	Implements the: • Engine • Steering, transmission, driveline, and brakes • Body, suspension, and wheels	✓
Visualization	Provides the vehicle trajectory, driver response, and 3D visualization	✓

To override the default variant, on the **Modeling** tab, in the **Design** section, click the drop-down. In the **General** section, select **Variant Manager**. In the Variant Manager, navigate to the variant that you want to use. Right-click and select **Override using this Choice**.

Lane Change Reference Generator

Use the Lane Change Reference Generator block to generate:

- Lane signals for the Visualization subsystem — The left and right lane boundaries are a function of the **Vehicle width** parameter.
- Velocity and lateral reference signals for the Predictive Driver block — Use the **Lateral reference position breakpoints** and **Lateral reference data** parameters to specify the lateral reference trajectory as a function of the longitudinal distance.

To specify the target velocity, use the **Longitudinal entrance velocity setpoint** parameter.

To start the maneuver a specified distance after the vehicle reaches the target speed, specify a **Distance after target speed to begin reference** parameter.

Predictive Driver

The reference application uses the Predictive Driver block to generate normalized steering, acceleration, and braking commands that track the reference trajectory.

The Predictive Driver block implements an optimal single-point preview (look ahead) control model developed by C. C. MacAdam[2], [3], [4]. The model represents driver steering control behavior during path-following and obstacle avoidance maneuvers. Drivers preview to follow a predefined path.

Environment

The Environment subsystem generates the wind and ground forces. The reference application has these environment variants.

Environment	Variant	Description
Ground Feedback	`3D Engine`	Uses Vehicle Terrain Sensor block to implement ray tracing in 3D environment
	`Constant` (default)	Implements a constant friction value

Controllers

The Controllers subsystem generates engine torque, transmission gear, and brake commands. The reference application has these brake variants.

Controller	Variant	Description
Brake Pressure Control	`Bang Bang ABS`	Anti-lock braking system (ABS) feedback controller that switches between two states
	`Open Loop` (default)	Open loop braking controller

Passenger Vehicle

The Passenger Vehicle subsystem has an engine, controllers, and a vehicle body with four wheels. Specifically, the vehicle contains these subsystems.

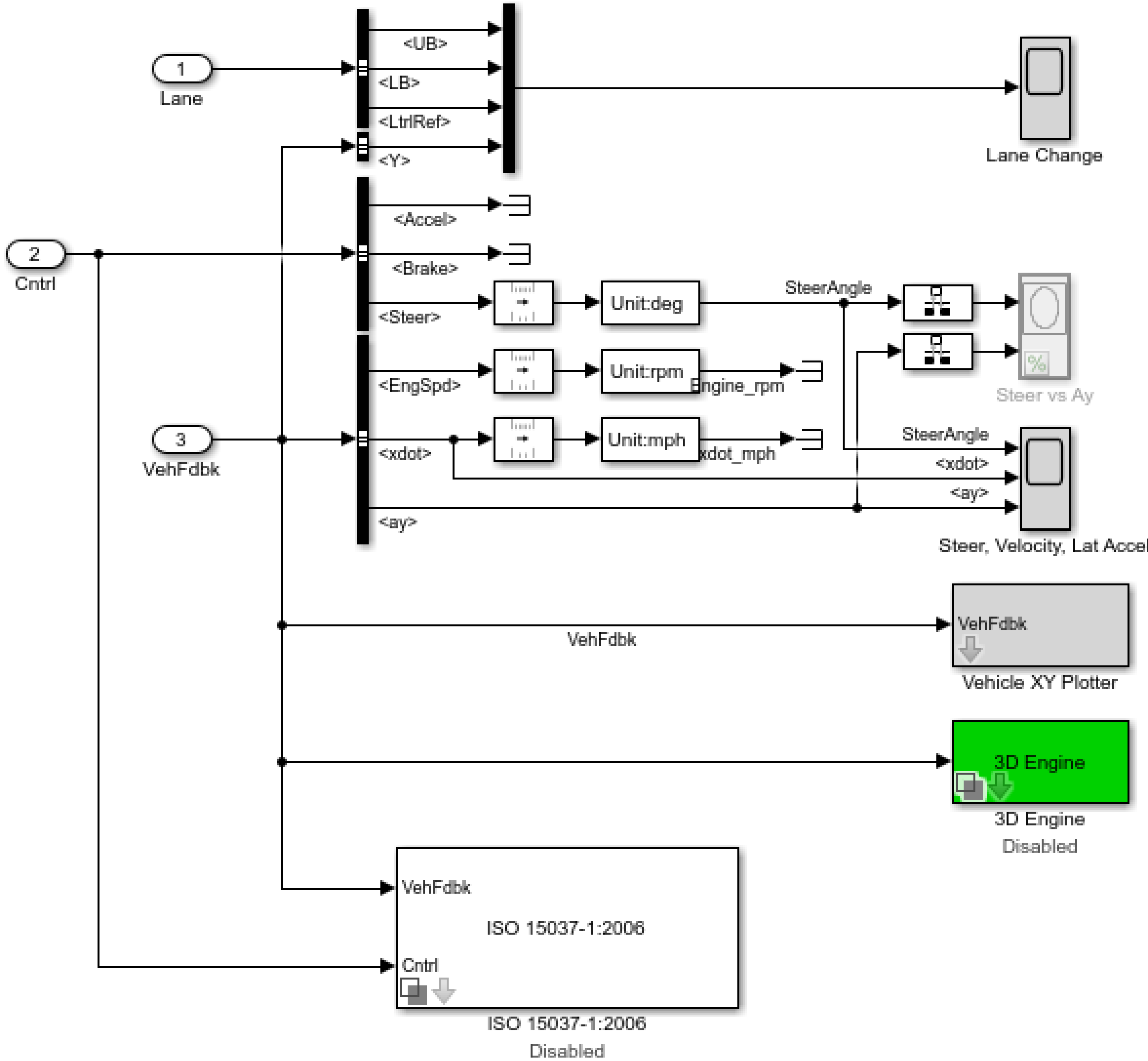
Body, Suspension, Wheels Subsystem	Variant	Description
PassVeh7DOF	`PassVeh7DOF` (default)	Vehicle with four wheels: • Vehicle body has three degrees-of-freedom (DOFs) — Longitudinal, lateral, and yaw • Each wheel has one DOF — Rolling
PassVeh14DOF	`PassVeh14DOF`	Vehicle with four wheels. • Vehicle body has six DOFs — Longitudinal, lateral, vertical and pitch, yaw, and roll • Each wheel has two DOFs — Vertical and rolling

Engine Subsystem	Variant	Description
Mapped Engine	`SiMappedEngine` (default)	Mapped spark-ignition (SI) engine

Steering, Transmission, Driveline, and Brakes Subsystem		Variant	Description
Driveline Ideal Fixed Gear	Driveline model	`All Wheel Drive`	Configure the driveline for all-wheel, front-wheel, or rear-wheel drive Specify the type of torque coupling
		`Front Wheel Drive`	
		`Rear Wheel Drive` (default)	
	Transmission	`Ideal` (default)	Ideal fixed gear transmission

Visualization

When you run the simulation, the Visualization subsystem provides driver, vehicle, and response information. The reference application logs vehicle signals during the maneuver, including steering, vehicle and engine speed, and lateral acceleration. You can use the Simulation Data Inspector to import the logged signals and examine the data.


1
Lane
2
Cntrl
3
VehFdbk
<UB>
<LB>
<LtrlRef>
<Y>
<Accel>
<Brake>
<Steer>
<EngSpd>
<xdot>
<ay>
Unit:deg
Unit:rpm
Unit:mph
SteerAngle
Engine_rpm
xdot_mph
Lane Change
Steer vs Ay
SteerAngle
<xdot>
<ay>
Steer, Velocity, Lat Accel
VehFdbk
VehFdbk
Vehicle XY Plotter
3D Engine
3D Engine
Disabled
VehFdbk
ISO 15037-1:2006
Cntrl
ISO 15037-1:2006
Disabled

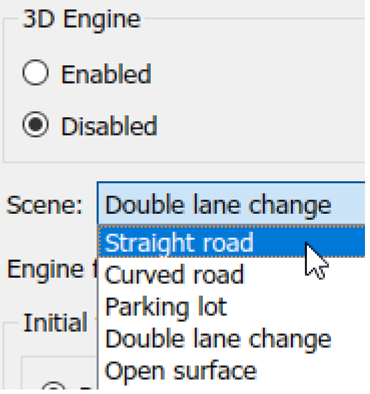
Element	Description
Driver Commands	Driver commands: • Handwheel angle • Acceleration command • Brake command
Vehicle Response	Vehicle response: • Engine speed • Vehicle speed • Acceleration command
Lane Change Scope block	Lateral vehicle displacement versus time: • Red line — Cones marking lane boundary • Blue line — Reference trajectory • Green line — Actual trajectory
Steer vs Ay Scope block	Steering angle versus lateral acceleration
Steer, Velocity, Lat Accel Scope block	• `SteerAngle` — Steering angle versus time • `<xdot>` — Longitudinal vehicle velocity versus time • `<ay>` — Lateral acceleration versus time
Vehicle XY Plotter	Vehicle longitudinal versus lateral distance
ISO 15037-1:2006 block	Display ISO standard measurement signals in the Simulation Data Inspector, including steering wheel angle and torque, longitudinal and lateral velocity, and sideslip angle

3D Visualization

Optionally, you can enable or disable the 3D visualization environment. For the 3D visualization engine platform requirements and hardware recommendations, see “3D Visualization Engine Requirements” on page 1-4. After you open the reference application, in the Visualization subsystem, open the 3D Engine block. Set these parameters.

- **3D Engine** to **Enabled**.

- **Scene** to one of the scenes, for example `Straight road`.



- To position the vehicle in the scene:

  1. Select the position initialization method:

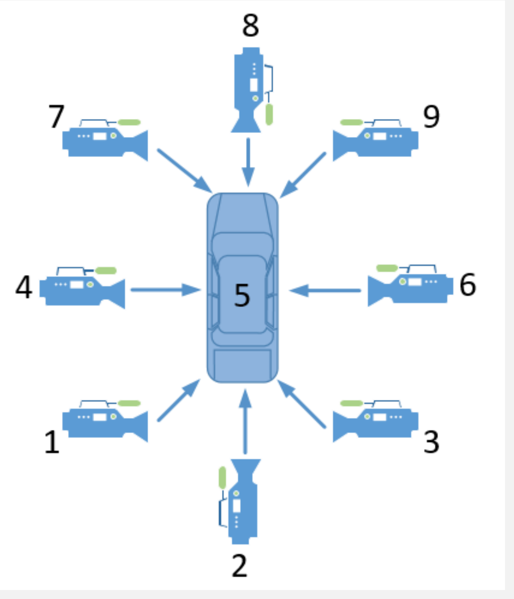
     - **Recommended for scene** — Set the initial vehicle position to values recommended for the scene
     - **User-specified** — Set your own initial vehicle position
  2. Select **Apply** to modify the initial vehicle position parameters.
  3. Click **Update the model workspaces with the initial values** to overwrite the initial vehicle position in the model workspaces with the applied values.

When you run the simulation, view the vehicle response in the `AutoVrtlEnv` window.

Note

- To open and close the `AutoVrtlEnv` window, use the Simulink Run and Stop buttons. If you manually close the `AutoVrtlEnv` window, Simulink stops the simulation with an error.
- When you enable the 3D visualization environment, you cannot step the simulation back.

To change the camera views in the AutoVrtlEnv window, use these key commands.

Key	Camera View	
1	Back left	
2	Back	
3	Back right	
4	Left	
5	Internal	
6	Right	
7	Front left	
8	Front	
9	Front right	
0	Overhead	

References

[1] ISO 3888-2: 2011. *Passenger cars — Test track for a severe lane-change manoeuvre*.

[2] MacAdam, C. C. "An Optimal Preview Control for Linear Systems". *Journal of Dynamic Systems, Measurement, and Control*. Vol. 102, Number 3, 1980.

[3] MacAdam, C. C. "Application of an Optimal Preview Control for Simulation of Closed-Loop Automobile Driving ". *IEEE Transactions on Systems, Man, and Cybernetics*. Vol. 11, Number 6, 1981.

[4] MacAdam, C. C. "Development of Driver/Vehicle Steering Interaction Models for Dynamic Analysis". *Final Technical Report UMTRI-88-53*. The University of Michigan Transportation Research Institute. 1988.

See Also

3D Engine | Mapped SI Engine | Predictive Driver | Vehicle Terrain Sensor

Related Examples

- "Send and Receive Double-Lane Change Scene Data" on page 3-65
- "Yaw Stability on Varying Road Surfaces" on page 1-27

More About

- "3D Visualization Engine Requirements" on page 1-4
- "Coordinate Systems in Vehicle Dynamics Blockset" on page 2-2
- "ISO 15037-1:2006 Standard Measurement Signals" on page 5-2
- "Passenger Vehicle Dynamics Models" on page 3-2
- "Send and Receive Double-Lane Change Scene Data" on page 3-65

- Simulation Data Inspector

Scene Interrogation in 3D Environment

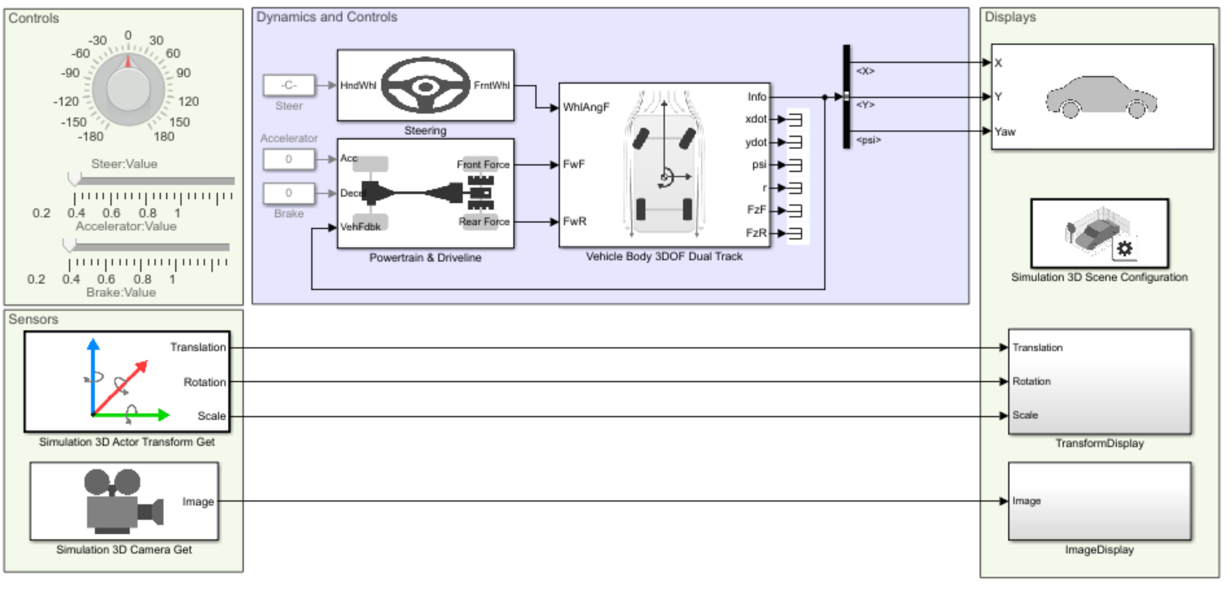
The scene interrogation with camera and ray tracing reference application provides the Simulink interface with the 3D visualization environment. For the minimum hardware required to run the reference application, see “3D Visualization Engine Requirements” on page 1-4.

The scene interrogation with camera and ray tracing reference application contains:

- One passenger vehicle with a simple driveline and a 3DOF vehicle dynamics model.
- One camera mounted on the passenger vehicle rearview mirror.
- Steering, acceleration, and braking control dials.
- 3D visualization environment configured for the Virtual Mcity scene.

Create and open a working copy of the camera and ray tracing reference application project.

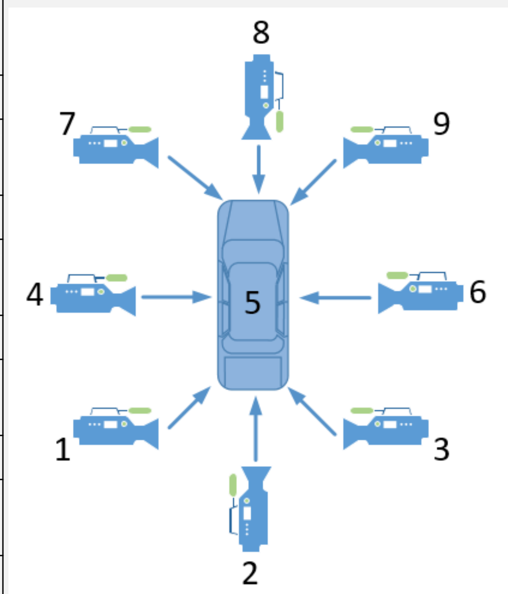
```
vdynblksSceneCameraRayStart
```



When you run the simulation, the reference application provides this vehicle and scene information.

<table>
<tr><th>Window</th><th>Description</th></tr>
<tr><td>AutoVrtlEnv</td><td>
Video output of the Unreal Engine 3D visualization environment image feedback. By default, the display shows the view from the Simulation 3D Scene Configuration block Scene view parameter <code>SimulinkVehicle1</code>.

To change the camera views in the AutoVrtlEnv window, use these key commands.

<table>
<tr><th>Key</th><th>Camera View</th><th></th></tr>
<tr><td>1</td><td>Back left</td><td rowspan="10">
</td></tr>
<tr><td>2</td><td>Back</td></tr>
<tr><td>3</td><td>Back right</td></tr>
<tr><td>4</td><td>Left</td></tr>
<tr><td>5</td><td>Internal</td></tr>
<tr><td>6</td><td>Right</td></tr>
<tr><td>7</td><td>Front left</td></tr>
<tr><td>8</td><td>Front</td></tr>
<tr><td>9</td><td>Front right</td></tr>
<tr><td>0</td><td>Overhead</td></tr>
</table>
</td></tr>
<tr><td>SDL Video Display</td><td>
Video image output of Simulation 3D Camera Get block. By default, the display shows the view specified by these parameter settings:

• Vehicle name — <code>SimulinkVehicle1</code>
• Vehicle mounting location — <code>Rearview mirror</code>
</td></tr>
</table>

This table summarizes the parts of the reference application.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Controls</td><td>Dials and gauges that control the vehicle steering, acceleration, and braking.</td></tr>
<tr><td>Sensors</td><td>The Simulation 3D Actor Transform Get block returns the translation, rotation, and scale for the vehicle passenger vehicle and four wheels from the 3D visualization environment.

The Simulation 3D Camera Get block returns the camera image from the 3D visualization environment. By default, the block returns image data for a camera location specified by these parameter settings:

• Vehicle name — <code>SimulinkVehicle1</code>
• Vehicle mounting location — <code>Rearview mirror</code></td></tr>
<tr><td>Dynamics and Controls</td><td>Interfaces with Simulink to calculate the dynamic response of the vehicle plant and controller. By default, the subsystem contains a simple driveline and the Vehicle 3DOF Dual Track block vehicle dynamics model.</td></tr>
<tr><td>Displays</td><td>The Simulation 3D Vehicle with Ground Following block implements a passenger vehicle in the 3D visualization environment. The block uses the vehicle position to adjust the vehicle elevation, roll, and pitch so that the vehicle follows the ground terrain. By default, the block has these parameter settings:

• Type — <code>Muscle car</code>
• Color — <code>Red</code>
• Name — <code>SimulinkVehicle1</code>

The Simulation 3D Scene Configuration block configures the Unreal Engine 3D visualization environment. By default, the block has these parameter settings:

• Scene description — <code>Virtual Mcity</code>
• Scene view — <code>SimulinkVehicle1</code>

The TransformDisplay subsystem displays the translation, rotation, and scale of the <code>SimulinkVehicle1</code> vehicle body and four wheels.

The ImageDisplay subsystem displays the video image output of Simulation 3D Camera Get block in the SDL Video Display window.</td></tr>
</table>

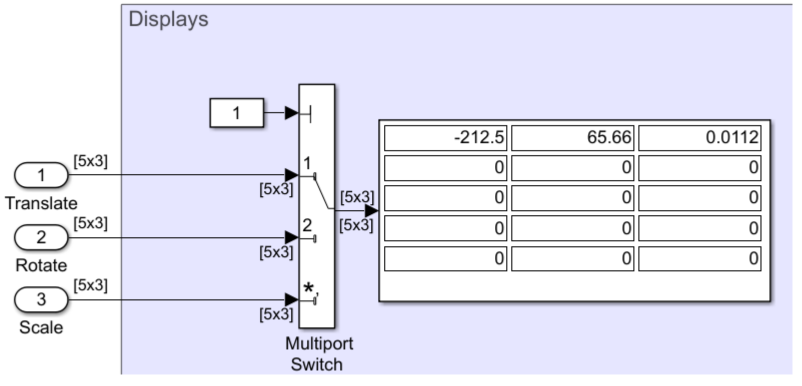
Displays Subsystems

TransformDisplay Subsystem

In the TransformDisplay subsystem, the Display block provides the translation, rotation, and scale of the vehicle body and four wheels. Use the Constant block value to control the display.

- `1` — Translation
- `2` — Rotation
- `3` — Scale

For example, to display translation information, set the value to `1`.



The display indicates that the:

- Vehicle body is at `-212.5` m, `65.66` m, and `0.0112` m along the world *X*-, *Y*-, and *Z*-axes, respectively.
- Wheels are at their initial positions along the world *X*-, *Y*-, and *Z*- axes, respectively.

The Display block provides an array of the vehicle and wheel locations.

$$\begin{bmatrix} Vehicle_X & Vehicle_Y & Vehicle_Z \\ FrontLeft_X & FrontLeft_Y & FrontLeft_Z \\ FrontRight_X & FrontRight_Y & FrontRight_Z \\ RearLeft_X & RearLeft_Y & RearLeft_Z \\ RearRear_X & RearRear_Y & RearRear_Z \end{bmatrix}$$

- Vehicle translation and rotation are along the world coordinate system axes.
- Wheel translations and rotations are with respect to their initial positions, along the world coordinate system axes.

ImageDisplay Subsystem

In the ImageDisplay subsystem, the Level-2 MATLAB S-Function block uses the `VideoDisplayMSfcnWin` function to display the video image output of Simulation 3D Camera Get block.

See Also

Simulation 3D Actor Transform Get | Simulation 3D Camera Get | Simulation 3D Scene Configuration | Simulation 3D Vehicle with Ground Following | **Virtual Mcity**

Related Examples

- “Send and Receive Double-Lane Change Scene Data” on page 3-65

More About

- “Coordinate Systems in Vehicle Dynamics Blockset” on page 2-2
- “Support Package for Customizing Scenes” on page 6-3
- “Vehicle Dynamics Blockset Communication with 3D Visualization Software” on page 1-6

External Websites

- Unreal Engine

Swept-Sine Steering Maneuver

This reference application represents a full vehicle dynamics model undergoing a swept-sine steering maneuver. You can create your own versions, providing a framework to test that your vehicle meets the design requirements under normal and extreme driving conditions. Use the reference application to analyze vehicle ride and handling and develop chassis controls. To analyze the dynamic steering response, use this reference application.

The swept-sine steering maneuver tests the vehicle frequency response to steering inputs. In the test, the driver:

- Accelerates until the vehicle hits a target velocity.
- Commands a sinusoidal steering wheel input.
- Linearly increase the frequency of the sinusoidal wave.

To test advanced driver assistance systems (ADAS) and automated driving (AD) perception, planning, and control software, you can run the maneuver in a 3D environment. For the 3D visualization engine platform requirements and hardware recommendations, see “3D Visualization Engine Requirements” on page 1-4.

To create and open a working copy of the swept-sine steering reference application project, enter

```
vdynblksSweptSineSteeringStart
```

This table summarizes the blocks and subsystems in the reference application. Some subsystems contain variants.

Reference Application Element	Description	Variants
Swept Sine Reference Generator block	Generate the sinusoidal steering commands for a swept-sine steering maneuver.	
Longitudinal Driver block	Generates normalized acceleration and braking commands to track speed.	
Environment	Implements wind and road forces.	✓

Reference Application Element	Description	Variants
Controllers	Implements controllers for engine control units (ECUs), transmissions, and brakes.	✓
Passenger Vehicle	Implements the: • Body, suspension, and wheels • Engine • Steering, transmission, driveline, and brakes	✓
Visualization	Provides the vehicle trajectory, driver response, and 3D visualization.	✓

To override the default variant, on the **Modeling** tab, in the **Design** section, click the drop-down. In the **General** section, select **Variant Manager**. In the Variant Manager, navigate to the variant that you want to use. Right-click and select **Override using this Choice**.

Swept Sine Reference Generator

Use the Swept Sine Reference block to generate the sinusoidal steering commands for a swept-sine steering maneuver.

- **Longitudinal velocity setpoint** — Target velocity
- **Steering amplitude** — Sinusoidal wave amplitude
- **Final frequency** — Cut off frequency to stop the maneuver

Longitudinal Driver

To track the vehicle speed, the Longitudinal Driver block implements an optimal single-point preview (look ahead) control model developed by C. C. MacAdam[1, 2, 3]. The model represents driver steering control behavior during path-following and obstacle avoidance maneuvers. Drivers preview (look ahead) to follow a predefined path.

Environment

The Environment subsystem generates the wind and ground forces. The reference application has these environment variants.

Environment	Variant	Description
Ground Feedback	`3D Engine`	Uses Vehicle Terrain Sensor block to implement ray tracing in 3D environment
	`Constant` (default)	Implements a constant friction value

Controllers

The Controllers subsystem generates engine torque, transmission gear, and brake commands. The reference application has these brake variants.

Controller	Variant	Description
Brake Pressure Control	`Bang Bang ABS`	Anti-lock braking system (ABS) feedback controller that switches between two states
	`Open Loop` (default)	Open loop braking controller

Passenger Vehicle

The Passenger Vehicle subsystem has an engine, controllers, and a vehicle body with four wheels. Specifically, the vehicle contains these subsystems.

Body, Suspension, Wheels Subsystem	Variant	Description
PassVeh7DOF	`PassVeh7DOF` (default)	Vehicle with four wheels: • Vehicle body has three degrees-of-freedom (DOFs) — Longitudinal, lateral, and yaw • Each wheel has one DOF — Rolling

Body, Suspension, Wheels Subsystem	Variant	Description
PassVeh14DOF	`PassVeh14DOF`	Vehicle with four wheels. • Vehicle body has six DOFs — Longitudinal, lateral, vertical and pitch, yaw, and roll • Each wheel has two DOFs — Vertical and rolling

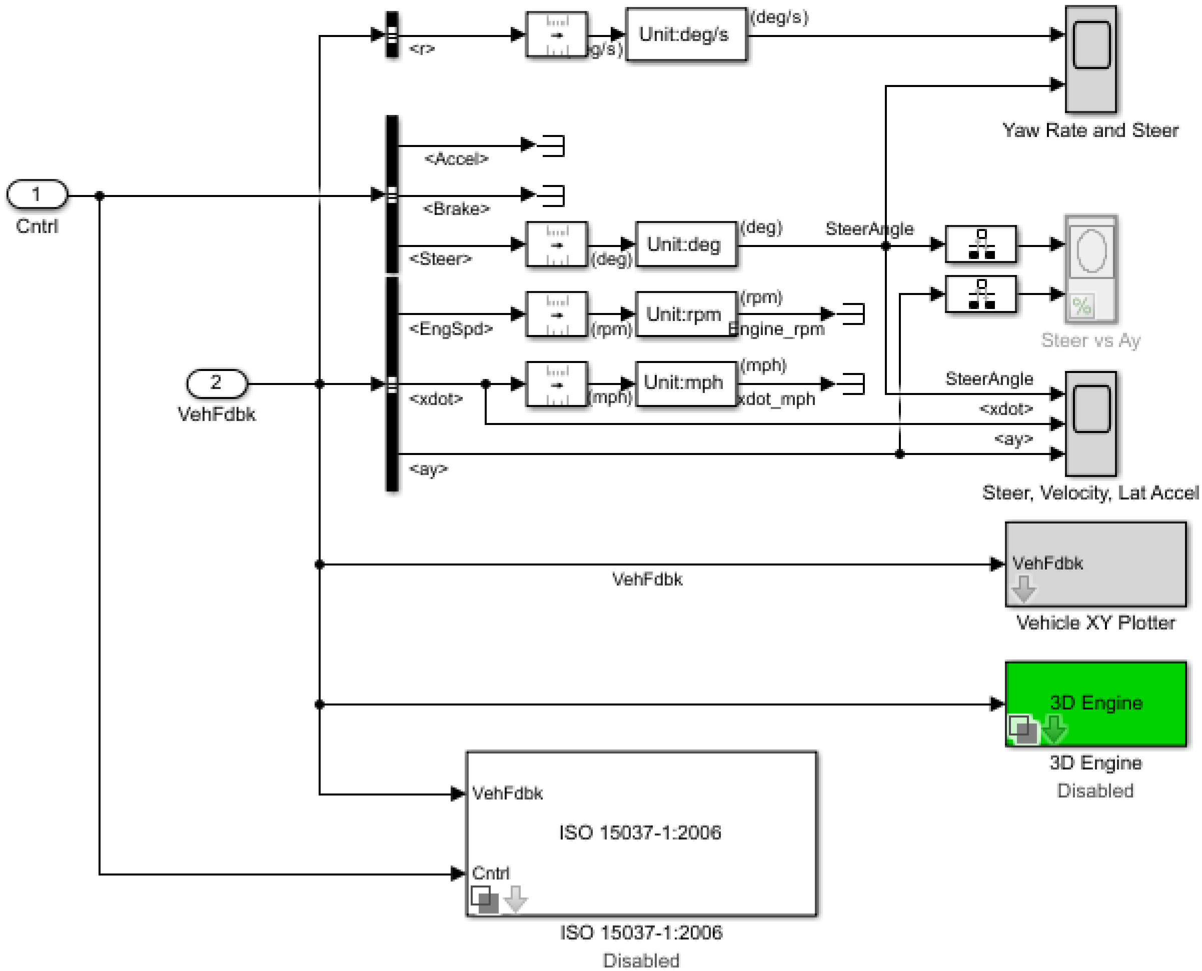
Engine Subsystem	Variant	Description
Mapped Engine	`SiMappedEngine` (default)	Mapped spark-ignition (SI) engine

<table>
<tr><th colspan="2">Steering, Transmission, Driveline, and Brakes Subsystem</th><th>Variant</th><th>Description</th></tr>
<tr><td rowspan="4">Driveline Ideal Fixed Gear</td><td rowspan="3">Driveline model</td><td><code>All Wheel Drive</code></td><td rowspan="3">Configure the driveline for all-wheel, front-wheel, or rear-wheel drive

Specify the type of torque coupling</td></tr>
<tr><td><code>Front Wheel Drive</code></td></tr>
<tr><td><code>Rear Wheel Drive</code> (default)</td></tr>
<tr><td>Transmission</td><td><code>Ideal</code> (default)</td><td>Ideal fixed gear transmission</td></tr>
</table>

Visualization Subsystem

When you run the simulation, the Visualization subsystem provides driver, vehicle, and response information. The reference application logs vehicle signals during the maneuver, including steering, vehicle and engine speed, and lateral acceleration. You can use the Simulation Data Inspector to import the logged signals and examine the data.



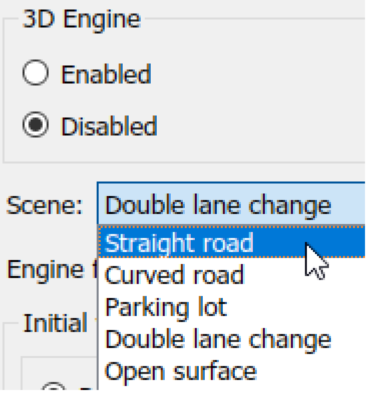
Element	Description
Driver Commands	Driver commands: • Handwheel angle • Acceleration command • Brake command

Element	Description
Vehicle Response	Vehicle response: • Engine speed • Vehicle speed • Acceleration command
Yaw Rate and Steer Scope block	Yaw rate and steering angle versus time: • Yellow line — Yaw rate • Blue lines — Steering angle
Steer vs Ay Scope block	Steering angle versus lateral acceleration
Steer, Velocity, Lat Accel Scope block	• `SteerAngle` — Steering angle versus time • `<xdot>` — Longitudinal vehicle velocity versus time • `<ay>` — Lateral acceleration versus time
Vehicle XY Plotter	Plot of vehicle longitudinal versus lateral distance
ISO 15037-1:2006 block	Display ISO standard measurement signals in the Simulation Data Inspector, including steering wheel angle and torque, longitudinal and lateral velocity, and sideslip angle

3D Visualization

Optionally, you can enable or disable the 3D visualization environment. For the 3D visualization engine platform requirements and hardware recommendations, see "3D Visualization Engine Requirements" on page 1-4. After you open the reference application, in the Visualization subsystem, open the 3D Engine block. Set these parameters.

- **3D Engine** to **Enabled**.
- **Scene** to one of the scenes, for example `Straight road`.



- To position the vehicle in the scene:

  1. Select the position initialization method:

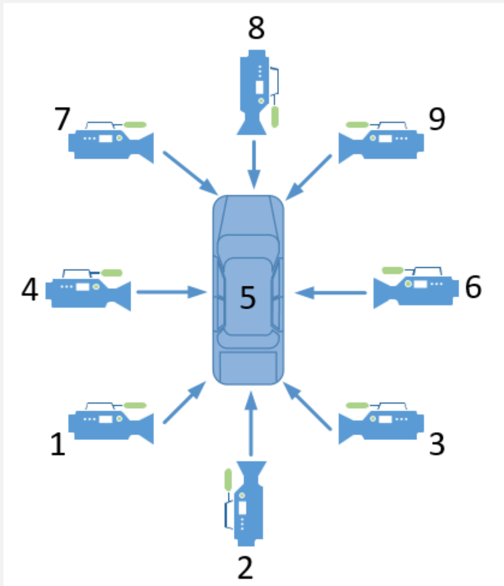
     - **Recommended for scene** — Set the initial vehicle position to values recommended for the scene
     - **User-specified** — Set your own initial vehicle position
  2. Select **Apply** to modify the initial vehicle position parameters.
  3. Click **Update the model workspaces with the initial values** to overwrite the initial vehicle position in the model workspaces with the applied values.

When you run the simulation, view the vehicle response in the `AutoVrtlEnv` window.

Note

- To open and close the `AutoVrtlEnv` window, use the Simulink Run and Stop buttons. If you manually close the `AutoVrtlEnv` window, Simulink stops the simulation with an error.
- When you enable the 3D visualization environment, you cannot step the simulation back.

To change the camera views in the AutoVrtlEnv window, use these key commands.

Key	Camera View	
1	Back left	
2	Back	
3	Back right	
4	Left	
5	Internal	
6	Right	
7	Front left	
8	Front	
9	Front right	
0	Overhead	

References

[1] MacAdam, C. C. "An Optimal Preview Control for Linear Systems". *Journal of Dynamic Systems, Measurement, and Control*. Vol. 102, Number 3, Sept. 1980.

[2] MacAdam, C. C. "Application of an Optimal Preview Control for Simulation of Closed-Loop Automobile Driving ". *IEEE Transactions on Systems, Man, and Cybernetics*. Vol. 11, Issue 6, June 1981.

[3] MacAdam, C. C. *Development of Driver/Vehicle Steering Interaction Models for Dynamic Analysis*. Final Technical Report UMTRI-88-53. Ann Arbor, Michigan: The University of Michigan Transportation Research Institute, Dec. 1988.

See Also

3D Engine | Longitudinal Driver | Mapped SI Engine | Vehicle Terrain Sensor

Related Examples

- “Frequency Response to Steering Angle Input” on page 1-67

More About

- “3D Visualization Engine Requirements” on page 1-4
- “Coordinate Systems in Vehicle Dynamics Blockset” on page 2-2
- “ISO 15037-1:2006 Standard Measurement Signals” on page 5-2
- “Passenger Vehicle Dynamics Models” on page 3-2

- Simulation Data Inspector

Slowly Increasing Steering Maneuver

This reference application represents a full vehicle dynamics model undergoing a slowly increasing steering maneuver according to standard SAE J266[1]. You can create your own versions, establishing a framework to test that your vehicle meets the design requirements under normal and extreme driving conditions. Use the reference application to analyze vehicle ride and handling and develop chassis controls. To characterize the steering and lateral vehicle dynamics, use this reference application.

Based on the constant speed, variable steer test defined in SAE J266[1], the slowly increasing steering maneuver helps characterize the lateral dynamics of the vehicle. In the test, the driver:

- Accelerates until vehicle hits a target velocity.
- Maintains a target velocity.
- Linearly increases the steering wheel angle from 0 degrees to a maximum angle.
- Maintains the steering wheel angle for a specified time.
- Linearly decreases the steering wheel angle from maximum angle to 0 degrees.

To test advanced driver assistance systems (ADAS) and automated driving (AD) perception, planning, and control software, you can run the maneuver in a 3D environment. For the 3D visualization engine platform requirements and hardware recommendations, see “3D Visualization Engine Requirements” on page 1-4.

To create and open a working copy of the increasing steering reference application project, enter

```
vdynblksIncreasingSteeringStart
```

This table summarizes the blocks and subsystems in the reference application. Some subsystems contain variants.

Reference Application Element	Description	Variants
Slowly Increasing Steer block	Generates steering, accelerator, and brake commands for the Longitudinal Driver block	

Reference Application Element	Description	Variant s
Longitudinal Driver block	Generates normalized acceleration and braking commands to track speed	
Environment	Implements wind and road forces.	✓
Controllers	Implements controllers for engine control units (ECUs), transmissions, and brakes	✓
Passenger Vehicle	Implements the: • Body, suspension, and wheels • Engine • Steering, transmission, driveline, and brakes	✓
Visualization	Provides the vehicle trajectory, driver response, and 3D visualization	✓

To override the default variant, on the **Modeling** tab, in the **Design** section, click the drop-down. In the **General** section, select **Variant Manager**. In the Variant Manager, navigate to the variant that you want to use. Right-click and select **Override using this Choice**.

Slowly Increasing Steer Block

Use the Slowly Increasing Steering block to generate steering, accelerator, and brake commands for a slowly increasing steering maneuver[1].

- **Longitudinal speed setpoint** — Target velocity setpoint
- **Handwheel rate** — Linear rate to increase steering wheel angle
- **Maximum handwheel angle** — Maximum steering wheel angle

Longitudinal Driver

To track the vehicle speed, the Longitudinal Driver block implements an optimal single-point preview (look ahead) control model developed by C. C. MacAdam[2, 3, 4]. The model represents driver steering control behavior during path-following and obstacle avoidance maneuvers. Drivers preview (look ahead) to follow a predefined path.

Environment

The Environment subsystem generates the wind and ground forces. The reference application has these environment variants.

Environment	Variant	Description
Ground Feedback	`3D Engine`	Uses Vehicle Terrain Sensor block to implement ray tracing in 3D environment
	`Constant` (default)	Implements a constant friction value

Controllers

The Controllers subsystem generates engine torque, transmission gear, and brake commands. The reference application has these brake variants.

Controller	Variant	Description
Brake Pressure Control	`Bang Bang ABS`	Anti-lock braking system (ABS) feedback controller that switches between two states
	`Open Loop` (default)	Open loop braking controller

Passenger Vehicle

The Passenger Vehicle subsystem has an engine, controllers, and a vehicle body with four wheels. Specifically, the vehicle contains these subsystems.

Body, Suspension, Wheels Subsystem	Variant	Description
PassVeh7DOF	`PassVeh7DOF` (default)	Vehicle with four wheels: • Vehicle body has three degrees-of-freedom (DOFs) — Longitudinal, lateral, and yaw • Each wheel has one DOF — Rolling

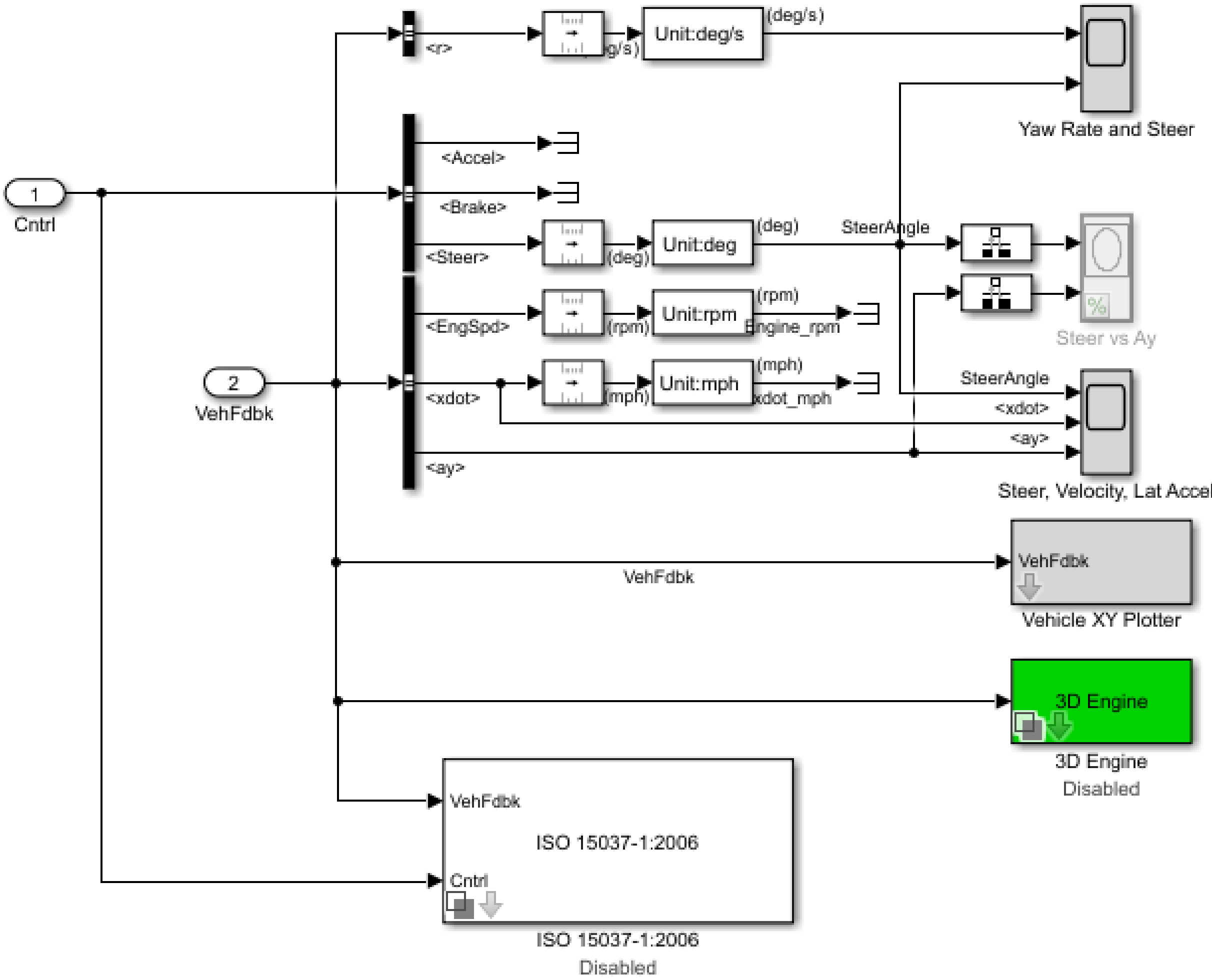
Body, Suspension, Wheels Subsystem	Variant	Description
PassVeh14DOF	`PassVeh14DOF`	Vehicle with four wheels. • Vehicle body has six DOFs — Longitudinal, lateral, vertical and pitch, yaw, and roll • Each wheel has two DOFs — Vertical and rolling

Engine Subsystem	Variant	Description
Mapped Engine	`SiMappedEngine` (default)	Mapped spark-ignition (SI) engine

Steering, Transmission, Driveline, and Brakes Subsystem		Variant	Description
Driveline Ideal Fixed Gear	Driveline model	`All Wheel Drive`	Configure the driveline for all-wheel, front-wheel, or rear-wheel drive Specify the type of torque coupling
		`Front Wheel Drive`	
		`Rear Wheel Drive` (default)	
	Transmission	`Ideal` (default)	Ideal fixed gear transmission

Visualization

When you run the simulation, the Visualization subsystem provides driver, vehicle, and response information. The reference application logs vehicle signals during the maneuver, including steering, vehicle and engine speed, and lateral acceleration. You can use the Simulation Data Inspector to import the logged signals and examine the data.



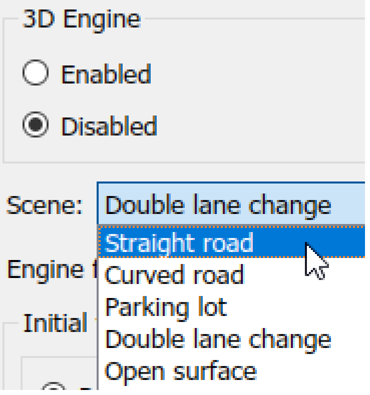
Element	Description
Driver Commands	Driver commands: • Handwheel angle • Acceleration command • Brake command

Element	Description
Vehicle Response	Vehicle response: • Engine speed • Vehicle speed • Acceleration command
Yaw Rate and Steer Scope block	Yaw rate and steering angle versus time: • Yellow line — Yaw rate • Blue lines — Steering angle
Steer vs Ay Scope block	Steering angle versus lateral acceleration
Steer, Velocity, Lat Accel Scope block	• SteerAngle — Steering angle versus time • <xdot> — Longitudinal vehicle velocity versus time • <ay> — Lateral acceleration versus time
Vehicle XY Plotter	Plot of vehicle longitudinal versus lateral distance
ISO 15037-1:2006 block	Display ISO standard measurement signals in the Simulation Data Inspector, including steering wheel angle and torque, longitudinal and lateral velocity, and sideslip angle

3D Visualization

Optionally, you can enable or disable the 3D visualization environment. For the 3D visualization engine platform requirements and hardware recommendations, see “3D Visualization Engine Requirements” on page 1-4. After you open the reference application, in the Visualization subsystem, open the 3D Engine block. Set these parameters.

- **3D Engine** to **Enabled**.
- **Scene** to one of the scenes, for example `Straight road`.



- To position the vehicle in the scene:

  1. Select the position initialization method:

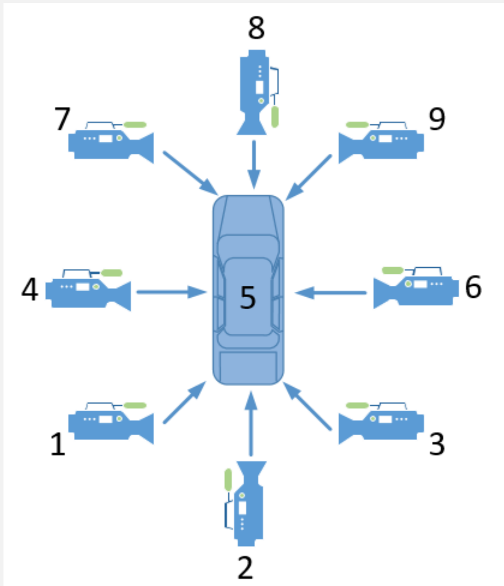
     - **Recommended for scene** — Set the initial vehicle position to values recommended for the scene
     - **User-specified** — Set your own initial vehicle position
  2. Select **Apply** to modify the initial vehicle position parameters.
  3. Click **Update the model workspaces with the initial values** to overwrite the initial vehicle position in the model workspaces with the applied values.

When you run the simulation, view the vehicle response in the `AutoVrtlEnv` window.

Note

- To open and close the `AutoVrtlEnv` window, use the Simulink Run and Stop buttons. If you manually close the `AutoVrtlEnv` window, Simulink stops the simulation with an error.
- When you enable the 3D visualization environment, you cannot step the simulation back.

To change the camera views in the AutoVrtlEnv window, use these key commands.

Key	Camera View	
1	Back left	
2	Back	
3	Back right	
4	Left	
5	Internal	
6	Right	
7	Front left	
8	Front	
9	Front right	
0	Overhead	

References

[1] SAE J266. *Steady-State Directional Control Test Procedures For Passenger Cars and Light Trucks*. Warrendale, PA: SAE International, 1996.

[2] MacAdam, C. C. "An Optimal Preview Control for Linear Systems". *Journal of Dynamic Systems, Measurement, and Control*. Vol. 102, Number 3, Sept. 1980.

[3] MacAdam, C. C. "Application of an Optimal Preview Control for Simulation of Closed-Loop Automobile Driving ". *IEEE Transactions on Systems, Man, and Cybernetics*. Vol. 11, Issue 6, June 1981.

[4] MacAdam, C. C. *Development of Driver/Vehicle Steering Interaction Models for Dynamic Analysis*. Final Technical Report UMTRI-88-53. Ann Arbor, Michigan: The University of Michigan Transportation Research Institute, Dec. 1988.

See Also

3D Engine | Longitudinal Driver | Mapped SI Engine | Vehicle Terrain Sensor

Related Examples

- “Vehicle Steering Gain at Different Speeds” on page 1-44

More About

- “3D Visualization Engine Requirements” on page 1-4
- “Coordinate Systems in Vehicle Dynamics Blockset” on page 2-2
- “ISO 15037-1:2006 Standard Measurement Signals” on page 5-2
- “Passenger Vehicle Dynamics Models” on page 3-2
- Simulation Data Inspector

Constant Radius Maneuver

This reference application represents a full vehicle dynamics model undergoing a constant radius test maneuver. For information about similar maneuvers, see standards SAE J266_199601[1] and ISO 4138:2012[2]. You can create your own versions, establishing a framework to test that your vehicle meets the design requirements under normal and extreme driving conditions. Use this reference application in ride and handling studies and chassis controls development to characterize the steering and lateral vehicle dynamics.

You can configure the reference application for open-loop and closed-loop tests:

- Open-loop — Maintain the target velocity and steering wheel angle to determine the lateral acceleration, side-slip characteristics, and steering angles for specific accelerations and subsequent test maneuvers. For the open-loop test, set the Reference Generator block **Maneuver** parameter to `Increasing Steer`.
- Closed-loop — Use the predictive driver to maintain a prespecified turn radius at different velocities for drivability and handling performance studies. For the closed-loop test, set the Reference Generator block **Maneuver** parameter to `Constant radius`.

To create and open a working copy of the constant radius reference application, enter

```
vdynblksConstRadiusStart
```

This table summarizes the blocks and subsystems in the reference application. Some subsystems contain variants.

Reference Application Element	Description	Variants
Reference Generator block	Sets the parameters that configure the maneuver and 3D visualization environment. By default, the block is set for the constant radius maneuver with the 3D simulation engine environment disabled. For the minimum 3D visualization environment hardware requirements, see "3D Visualization Engine Requirements" on page 1-4. To enable 3D visualization, on the **3D Engine** tab, select **Enabled**.	✓
Driver Commands block	Implements the driver model that the reference application uses to generate acceleration, braking, gear, and steering commands. By default, Driver Commands block variant is the Predictive Driver block.	✓
Environment	Implements wind and road forces.	✓
Controllers	Implements controllers for engine control units (ECUs), transmissions, and brakes	✓
Passenger Vehicle	Implements the: • Body, suspension, and wheels • Engine • Steering, transmission, driveline, and brakes	✓
Visualization	Provides the vehicle trajectory and driver response	✓

Reference Generator

The Reference Generator block sets the parameters that configure the maneuver and 3D simulation environment. By default, the block is set for the constant radius maneuver with the 3D simulation engine environment disabled.

Use the **Maneuver** parameter to specify the type of maneuver. You can specify the double lane change, swept sine, sine with dwell, and slowly increasing maneuvers.

If you select the **Use maneuver-specific driver, initial position, and scene** parameter, the reference application sets the driver, initial position, and scene for the maneuver that you specified.

For more information, see Reference Generator.

Driver Commands

The Driver Commands block implements the driver model that the reference application uses to generate acceleration, braking, gear, and steering commands. By default, if you select the Reference Generator block parameter **Use maneuver-specific driver, initial position, and scene**, the reference application selects the driver for the maneuver that you specified.

Vehicle Command Mode Setting	Implementation
`Longitudinal Driver`	Longitudinal Driver block — Longitudinal speed-tracking controller. Based on reference and feedback velocities, the block generates normalized acceleration and braking commands that can vary from 0 through 1. Use the block to model the dynamic response of a driver or to generate the commands necessary to track a longitudinal drive cycle.
`Predictive Driver`	Predictive Driver block — Controller that generates normalized steering, acceleration, and braking commands to track longitudinal velocity and a lateral reference displacement. The normalized commands can vary between -1 to 1. The controller uses a single-track (bicycle) model for optimal single-point preview control.
`Open Loop`	Implements an open-loop system so that you can configure the reference application for constant or signal-based steering, acceleration, braking, and gear command input.

Environment

The Environment subsystem generates the wind and ground forces. The reference application has these environment variants.

Environment	Variant	Description
Ground Feedback	`3D Engine`	Uses Vehicle Terrain Sensor block to implement ray tracing in 3D environment
	`Constant` (default)	Implements a constant friction value

Controllers

The Controllers subsystem generates engine torque, transmission gear, and brake commands. The reference application has these brake variants.

Controller	Variant	Description
Brake Pressure Control	`Bang Bang ABS`	Anti-lock braking system (ABS) feedback controller that switches between two states
	`Open Loop` (default)	Open loop braking controller

Passenger Vehicle

The Passenger Vehicle subsystem has an engine, controllers, and a vehicle body with four wheels. Specifically, the vehicle contains these subsystems.

Body, Suspension, Wheels Subsystem	Variant	Description
PassVeh7DOF	`PassVeh7DOF` (default)	Vehicle with four wheels: • Vehicle body has three degrees-of-freedom (DOFs) — Longitudinal, lateral, and yaw • Each wheel has one DOF — Rolling

Body, Suspension, Wheels Subsystem	Variant	Description
PassVeh14DOF	`PassVeh14DOF`	Vehicle with four wheels. • Vehicle body has six DOFs — Longitudinal, lateral, vertical and pitch, yaw, and roll • Each wheel has two DOFs — Vertical and rolling

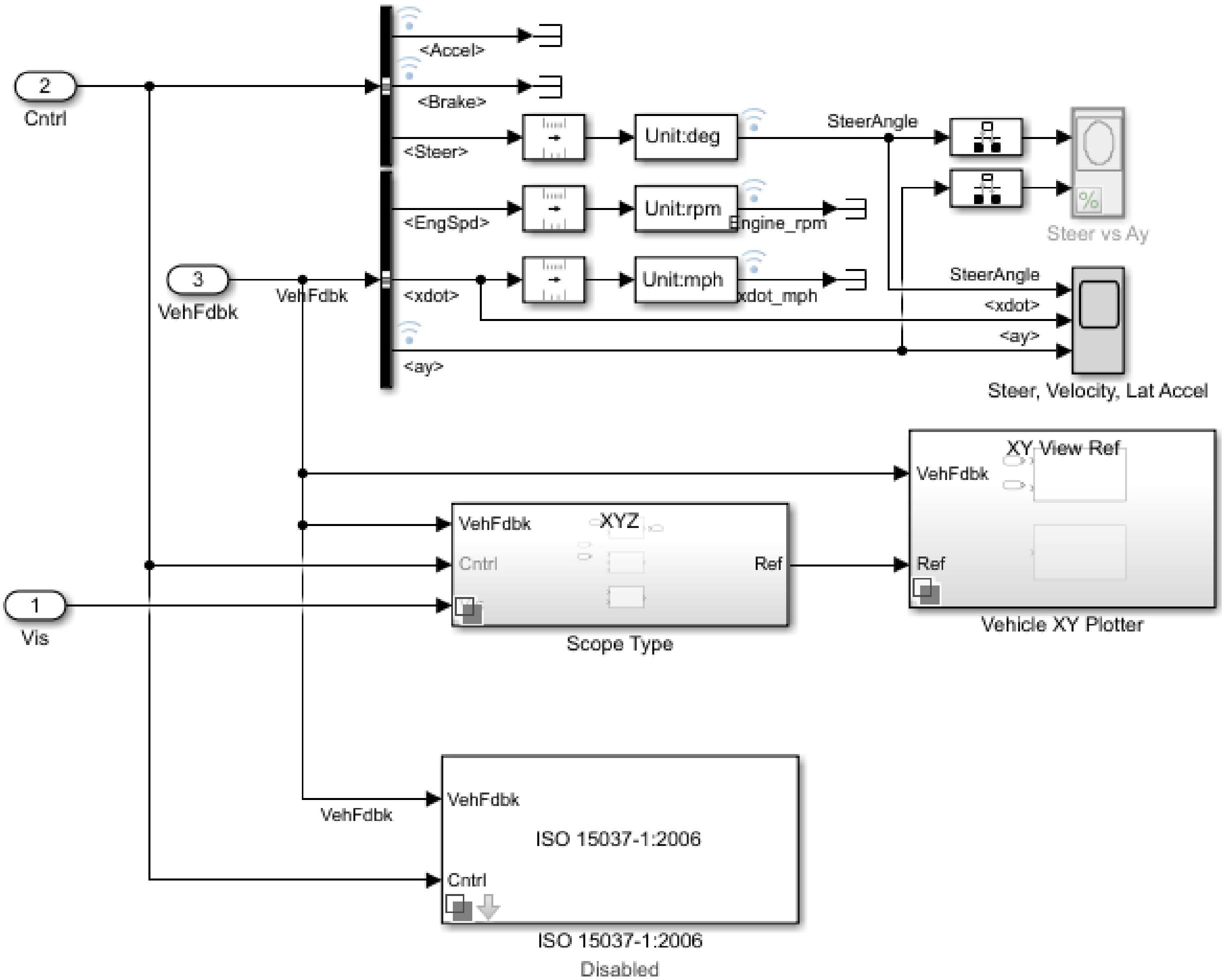
Engine Subsystem	Variant	Description
Mapped Engine	`SiMappedEngine` (default)	Mapped spark-ignition (SI) engine

<table>
<tr><th colspan="2">Steering, Transmission, Driveline, and Brakes Subsystem</th><th>Variant</th><th>Description</th></tr>
<tr><td rowspan="4">Driveline Ideal Fixed Gear</td><td rowspan="3">Driveline model</td><td><code>All Wheel Drive</code></td><td rowspan="3">Configure the driveline for all-wheel, front-wheel, or rear-wheel drive

Specify the type of torque coupling</td></tr>
<tr><td><code>Front Wheel Drive</code></td></tr>
<tr><td><code>Rear Wheel Drive</code> (default)</td></tr>
<tr><td>Transmission</td><td><code>Ideal</code> (default)</td><td>Ideal fixed gear transmission</td></tr>
</table>

Visualization

When you run the simulation, the Visualization subsystem provides driver, vehicle, and response information. The reference application logs vehicle signals during the maneuver, including steering, vehicle and engine speed, and lateral acceleration. You can use the Simulation Data Inspector to import the logged signals and examine the data.



Element	Description
Driver Commands	Driver commands: • Handwheel angle • Acceleration command • Brake command

Element	Description
Vehicle Response	Vehicle response: • Engine speed • Vehicle speed • Acceleration command
Steer, Velocity, Lat Accel Scope block	• `SteerAngle` — Steering angle versus time • `<xdot>` — Longitudinal vehicle velocity versus time • `<ay>` — Lateral acceleration versus time
Vehicle XY Plotter	Vehicle longitudinal versus lateral distance
ISO 15037-1:2006 block	Display ISO standard measurement signals in the Simulation Data Inspector, including steering wheel angle and torque, longitudinal and lateral velocity, and sideslip angle

References

[1] J266_199601. *Steady-State Directional Control Test Procedures for Passenger Cars and Light Trucks*. Warrendale, PA: SAE International, 1996.

[2] ISO 4138:2012. *Passenger cars — Steady-state circular driving behaviour — Open-loop test methods*. Geneva: ISO, 2012.

See Also

3D Engine | Driver Commands | Reference Generator

Related Examples

- “Vehicle Lateral Acceleration at Different Speeds” on page 1-56

More About

- “3D Visualization Engine Requirements” on page 1-4
- “Coordinate Systems in Vehicle Dynamics Blockset” on page 2-2
- “ISO 15037-1:2006 Standard Measurement Signals” on page 5-2

- Simulation Data Inspector
- "Slowly Increasing Steering Maneuver" on page 3-28

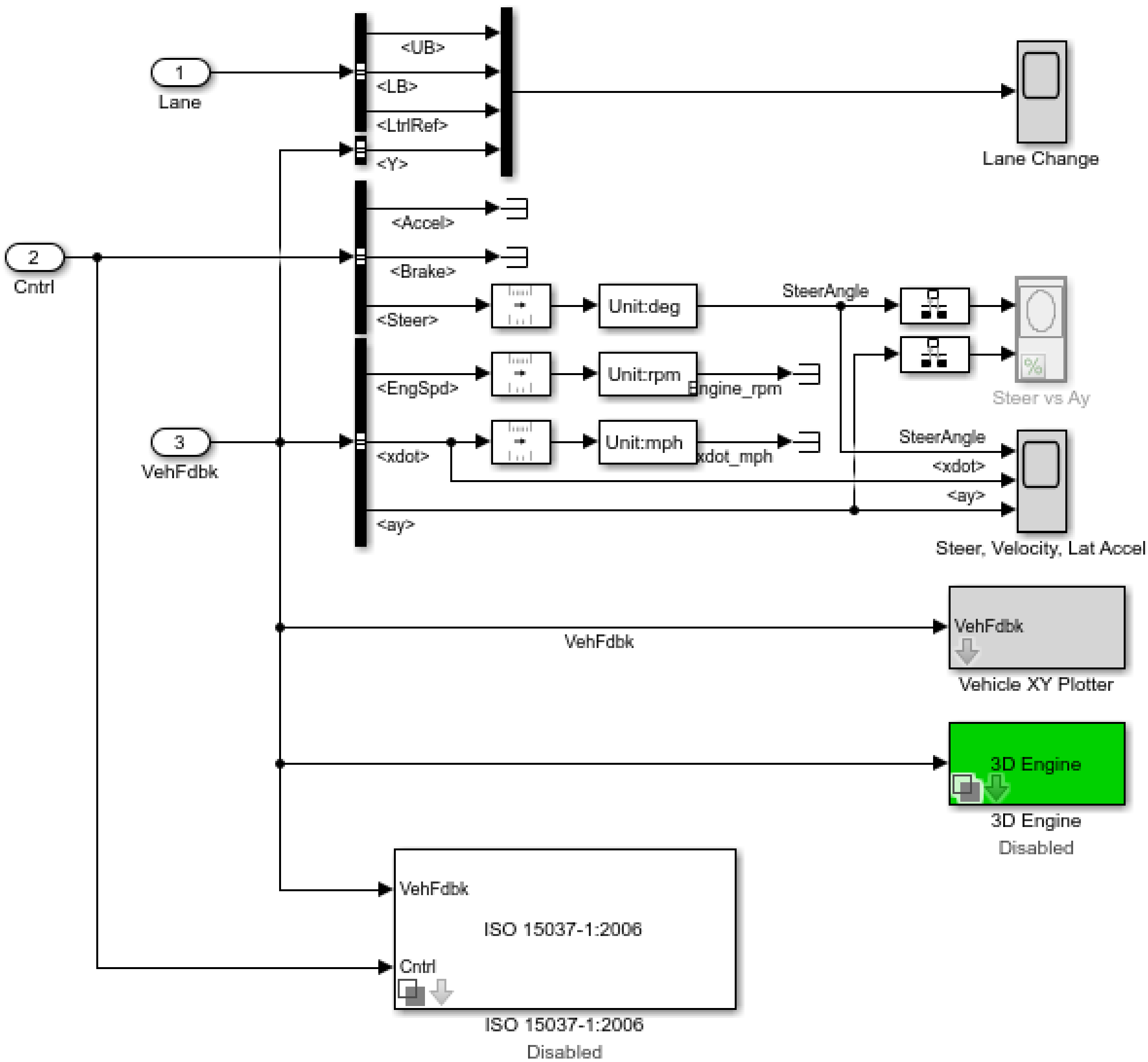
Run a Vehicle Dynamics Maneuver in 3D Environment

This example shows how to run a vehicle dynamics maneuver in a 3D environment. By integrating vehicle dynamics models with a 3D environment, you can test advanced driver assistance systems (ADAS) and automated driving (AD) perception, planning, and control software. For the 3D visualization engine platform requirements and hardware recommendations, see “3D Visualization Engine Requirements” on page 1-4.

1. Create and open a working copy of a maneuver reference application. For example, open the double-lane change reference application.

   ```
   vdynblksDblLaneChangeStart
   ```
2. Run the maneuver simulation. By default, the 3D environment is disabled.

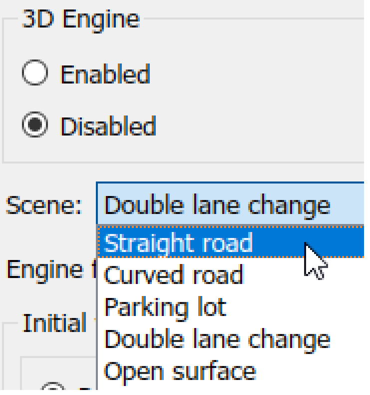
   When you run the simulation, the Visualization subsystem provides driver, vehicle, and response information. The reference application logs vehicle signals during the maneuver, including steering, vehicle and engine speed, and lateral acceleration. You can use the Simulation Data Inspector to import the logged signals and examine the data.


1
Lane
<UB>
<LB>
<LtrlRef>
<Y>
Lane Change
<Accel>
2
Cntrl
<Brake>
<Steer>
Unit:deg
SteerAngle
Steer vs Ay
<EngSpd>
Unit:rpm
Engine_rpm
3
VehFdbk
<xdot>
Unit:mph
xdot_mph
SteerAngle
<xdot>
<ay>
<ay>
Steer, Velocity, Lat Accel
VehFdbk
VehFdbk
Vehicle XY Plotter
3D Engine
3D Engine
Disabled
VehFdbk
ISO 15037-1:2006
Cntrl
ISO 15037-1:2006
Disabled

Element	Description
Driver Commands	Driver commands: • Handwheel angle • Acceleration command • Brake command
Vehicle Response	Vehicle response: • Engine speed • Vehicle speed • Acceleration command
Lane Change Scope block	Lateral vehicle displacement versus time: • Red line — Cones marking lane boundary • Blue line — Reference trajectory • Green line — Actual trajectory
Steer vs Ay Scope block	Steering angle versus lateral acceleration
Steer, Velocity, Lat Accel Scope block	• SteerAngle — Steering angle versus time • <xdot> — Longitudinal vehicle velocity versus time • <ay> — Lateral acceleration versus time
Vehicle XY Plotter	Vehicle longitudinal versus lateral distance
ISO 15037-1:2006 block	Display ISO standard measurement signals in the Simulation Data Inspector, including steering wheel angle and torque, longitudinal and lateral velocity, and sideslip angle

3. Enable the 3D visualization environment. In the Visualization subsystem, open the 3D Engine block. Set these parameters.

   - **3D Engine** to **Enabled**.
   - **Scene description** to one of the scenes, for example `Straight road`.



- To position the vehicle in the scene:

  a Select the position initialization method:

    - **Recommended for scene** — Set the initial vehicle position to values recommended for the scene
    - **User-specified** — Set your own initial vehicle position

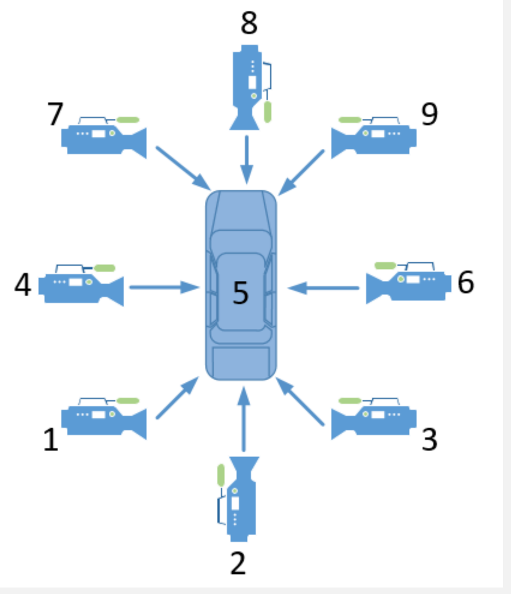
  b Select **Apply** to modify the initial vehicle position parameters.

  c Click **Update the model workspaces with the initial values** to overwrite the initial vehicle position in the model workspaces with the applied values.

To customize the scenes, use the Vehicle Dynamics Blockset Interface for Unreal Engine 4 Projects support package. For more information, see “Support Package for Customizing Scenes” on page 6-3.

4 Rerun the reference application. As the simulation runs, in the `AutoVrtlEnv` window, view the vehicle response.

To change the camera views in the AutoVrtlEnv window, use these key commands.

Key	Camera View	
1	Back left	
2	Back	
3	Back right	
4	Left	
5	Internal	
6	Right	
7	Front left	
8	Front	
9	Front right	
0	Overhead	

For example, when you run the double-lane change maneuver, use the cameras to visualize the vehicle as it changes lanes.

- Back

- Front left

- Internal

Note

- To open and close the `AutoVrtlEnv` window, use the Simulink Run and Stop buttons. If you manually close the `AutoVrtlEnv` window, Simulink stops the simulation with an error.
- When you enable the 3D visualization environment, you cannot step the simulation back.

See Also

More About

- “Double-Lane Change Maneuver” on page 3-4
- “Slowly Increasing Steering Maneuver” on page 3-28
- “Swept-Sine Steering Maneuver” on page 3-19
- Simulation Data Inspector
- “Support Package for Customizing Scenes” on page 6-3

Kinematics and Compliance Virtual Test Laboratory

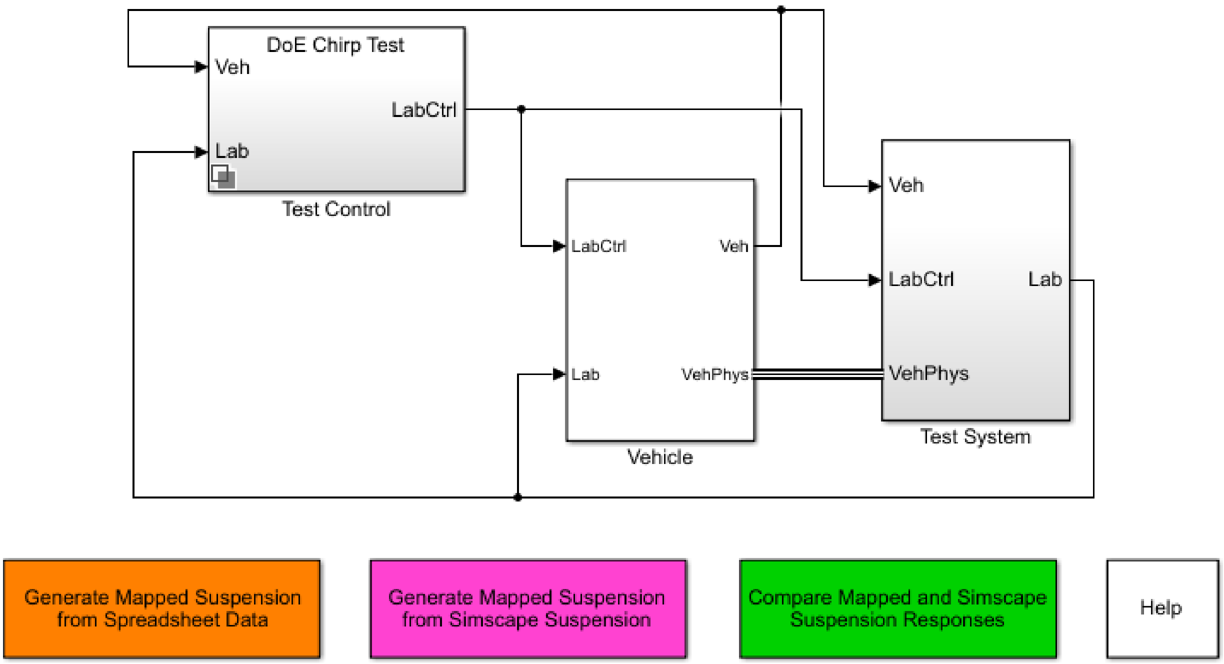
Model-Based Calibration Toolbox allows you to generate optimized suspension parameters for the Independent Suspension - Mapped and Solid Axle Suspension - Mapped blocks by using the kinematics (K) and compliance (C) virtual test laboratory.

To create and open a working copy of the K and C virtual test laboratory reference application, enter

```
vdynblksKandCTestLabStart
```

The K and C virtual test laboratory contains vehicle, test system, and test control subsystems. The vehicle system has two variants:

- `Simscape Multibody Vehicle` — Vehicle with a Simscape Multibody suspension system
- `VDBS Vehicle` — Vehicle with an Independent Suspension - Mapped block



This table summarizes the virtual test laboratory tests.

Test	Objective	Method
Generate Mapped Suspension from Spreadsheet Data	Use measured vertical force and suspension geometry data to generate calibrated suspension parameters for the mapped suspension blocks. **Note** You can use a third-party simulation model to generate the measured suspension data.	The virtual test lab uses Model-Based Calibration Toolbox to fit camber angle, toe angle, and vertical force response models for the data. The virtual test lab then uses the response models to generate suspension parameters for the suspension blocks.
Generate Mapped Suspension from Simscape Suspension	Use a Simscape Multibody suspension system to generate calibrated suspension parameters for the suspension mapped blocks.	The virtual test lab uses Model-Based Calibration Toolbox to perform a Sobol sequence design of experiments (DoE) on the suspension height and handwheel angle operating points. At each operating point, the reference application stimulates the Simscape Multibody suspension system with a chirp signal over a frequency range of 0.1 to 2 Hz. The virtual test lab then uses the data to fit the suspension vertical force, camber angle, and toe angle with a Gaussian process model (GPM) as a function of the suspension state. Finally, the reference application uses the GPM to generate suspension parameters for the suspension blocks.

Test	Objective	Method
Compare Mapped and Simscape Suspension Responses	Compare the mapped suspension with the Simscape Multibody suspension results.	The virtual test laboratory stimulates the Simscape Multibody suspension at one operating point and then compares the response to the mapped suspension.

Generate Mapped Suspension from Spreadsheet Data

The virtual test lab uses Model-Based Calibration Toolbox to fit camber angle, toe angle, and vertical force response models for the data. The virtual test lab then uses the response models to generate suspension parameters for the suspension blocks.

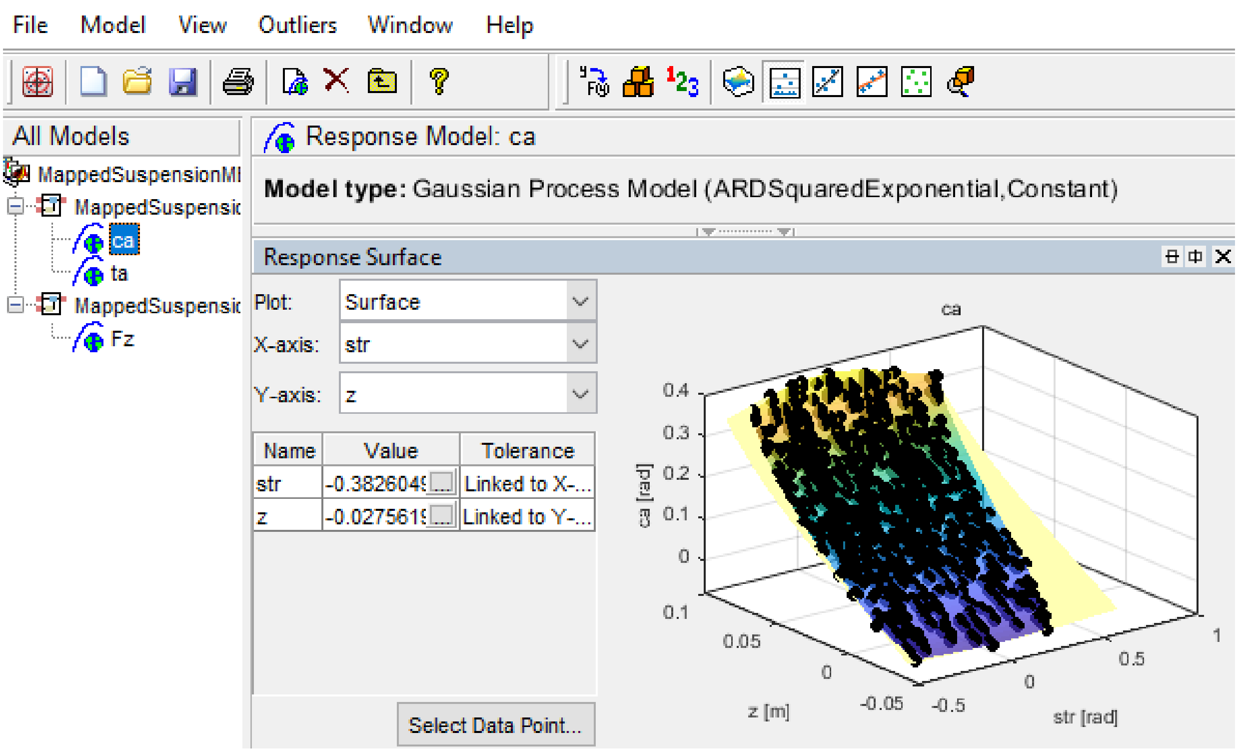
Generate Mapped Suspension Calibration

1. Use the **Spreadsheet file** field to provide a data file. By default, the reference application has `KandCTestData.xlsx` containing required data. The table summarizes the data file requirements for generating calibrated tables.

Data	Description	Data Requirements for Generating Mapped Suspension Tables
z	Vertical axis suspension height, in m	*Required*
zdot	Vertical axis suspension height velocity breakpoints, in m/s	*Required*
str	Steering angle, in rad	*Required*
Fz	Vertical axis suspension force, in N	*Required*
ca	Camber angle, in rad	*Required*
ta	Toe angle, in rad	*Required*

2. Click **Generate mapped suspension calibration** to generate response surface models in Model-Based Calibration Toolbox.

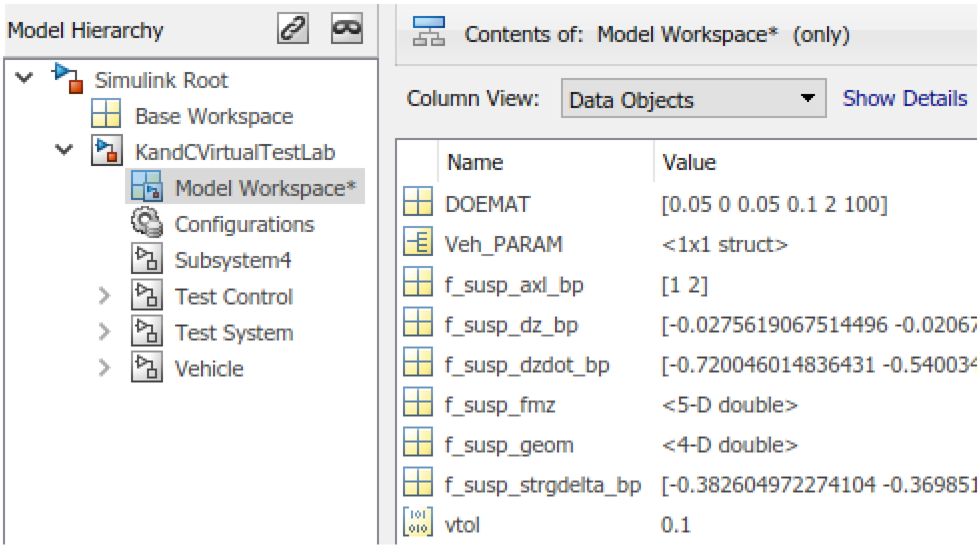
   The model browser opens when the process completes. You can view the camber angle, *ca*, toe angle, *ta*, and vertical force, *Fz*, response model fits for the data.



Apply Calibration to Mapped Suspension Model

1. Click **Apply calibration to mapped suspension model**. The virtual test lab uses the response models to generate calibrated suspension and breakpoint data.
2. Click **OK** to update the model workspace and suspension blocks.

In the Model Explorer, you can view the generated suspension parameters.



Parameter	Model Workspace Variable	Description
Axle breakpoints, f_susp_axl_bp	`f_susp_axl_bp`	Axle breakpoints, *P*, dimensionless.
Vertical axis suspension height breakpoints, f_susp_dz_bp	`f_susp_dz_bp`	Vertical axis suspension height breakpoints, *M*, in m.
Vertical axis suspension height velocity breakpoints, f_susp_dzdot_bp	`f_susp_dzdot_bp`	Vertical axis suspension height velocity breakpoints, *N*, in m/s.

Parameter	Model Workspace Variable	Description

Vertical axis suspension force and moment responses, f_susp_fmz	`f_susp_fmz`	`M`-by-`N`-by-`O`-by-`P`-by-`4` array of output values as a function of:
		• Vertical suspension height, *M*
		• Vertical suspension height velocity, *N*
		• Steering angle, *O*
		• Axle, *P*
		• 4 output types
		  • 1 — Vertical force, in N·m
		  • 2 — User-defined
		  • 3 — Stored energy, in J
		  • 4 — Absorbed power, in W

Suspension geometry responses, f_susp_geom	`f_susp_geom`	`M`-by-`O`-by-`P`-by-`3` array of geometric suspension values as a function of:
		• Vertical suspension height, *M*
		• Steering angle, *O*
		• Axle, *P*
		• 3 output types
		  • 1 — Camber angle, in rad
		  • 2 — Caster angle, in rad
		  • 3 — Toe angle, in rad

Steering angle breakpoints, f_susp_strgdelta_bp	`f_susp_strgdelta_bp`	Steering angle breakpoints, *O*, in rad.

Generate Mapped Suspension from Simscape Suspension

The virtual test lab uses Model-Based Calibration Toolbox to perform a Sobol sequence design of experiments (DoE) on the suspension height and handwheel angle operating points. At each operating point, the reference application stimulates the Simscape

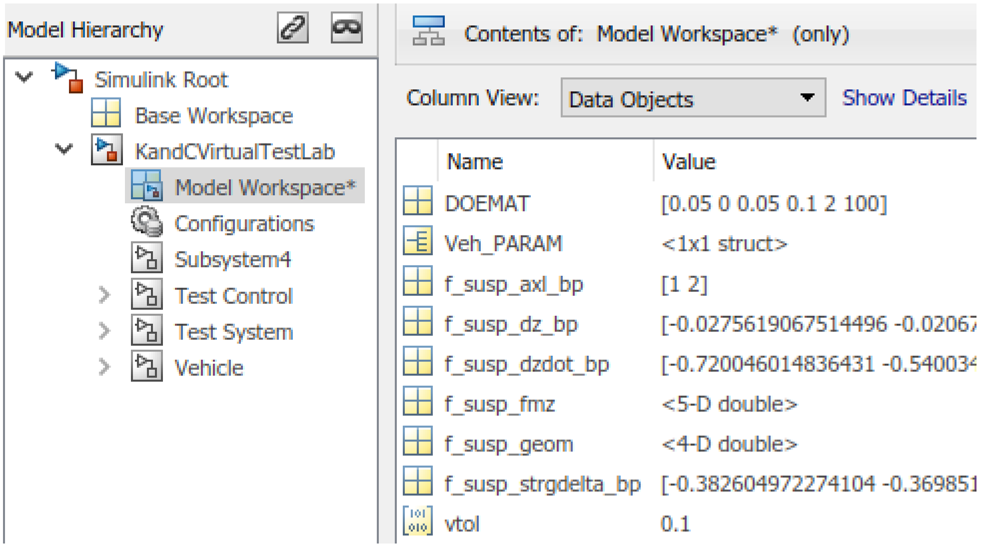
Multibody suspension system with a chirp signal over a frequency range of 0.1 to 2 Hz. The virtual test lab then uses the data to fit the suspension vertical force, camber angle, and toe angle with a Gaussian process model (GPM) as a function of the suspension state. Finally, the reference application uses the GPM to generate suspension parameters for the suspension blocks.

The test laboratory exercises the suspension system with the DOE settings contained in the `DOEMAT` array. To view the `DOEMAT` array values, open the Model Explorer.

Element	Description
`DOEMAT(1,1)`	Suspension height
`DOEMAT(1,2)`	Handwheel angle
`DOEMAT(1,3)`	Chirp signal amplitude
`DOEMAT(1,4)`	Starting chirp frequency
`DOEMAT(1,5)`	Ending chirp frequency
`DOEMAT(1,6)`	Simulation time to complete chirp signal frequency range

The generation can take time to run and slow other computer processes. View progress in the MATLAB window.

In the Model Explorer, you can view the generated suspension parameters.



Parameter	Model Workspace Variable	Description
Axle breakpoints, f_susp_axl_bp	`f_susp_axl_bp`	Axle breakpoints, *P*, dimensionless.
Vertical axis suspension height breakpoints, f_susp_dz_bp	`f_susp_dz_bp`	Vertical axis suspension height breakpoints, *M*, in m.
Vertical axis suspension height velocity breakpoints, f_susp_dzdot_bp	`f_susp_dzdot_bp`	Vertical axis suspension height velocity breakpoints, *N*, in m/s.

Vertical axis suspension force and moment responses, f_susp_fmz	`f_susp_fmz`	M-by-N-by-O-by-P-by-4 array of output values as a function of:
		• Vertical suspension height, *M*
		• Vertical suspension height velocity, *N*
		• Steering angle, *O*
		• Axle, *P*
		• 4 output types
		  • 1 — Vertical force, in N·m
		  • 2 — User-defined
		  • 3 — Stored energy, in J
		  • 4 — Absorbed power, in W

Parameter	Model Workspace Variable	Description

Suspension geometry responses, f_susp_geom	`f_susp_geom`	`M`-by-`O`-by-`P`-by-`3` array of geometric suspension values as a function of:
		• Vertical suspension height, *M*
		• Steering angle, *O*
		• Axle, *P*
		• 3 output types
		  • 1 — Camber angle, in rad
		  • 2 — Caster angle, in rad
		  • 3 — Toe angle, in rad

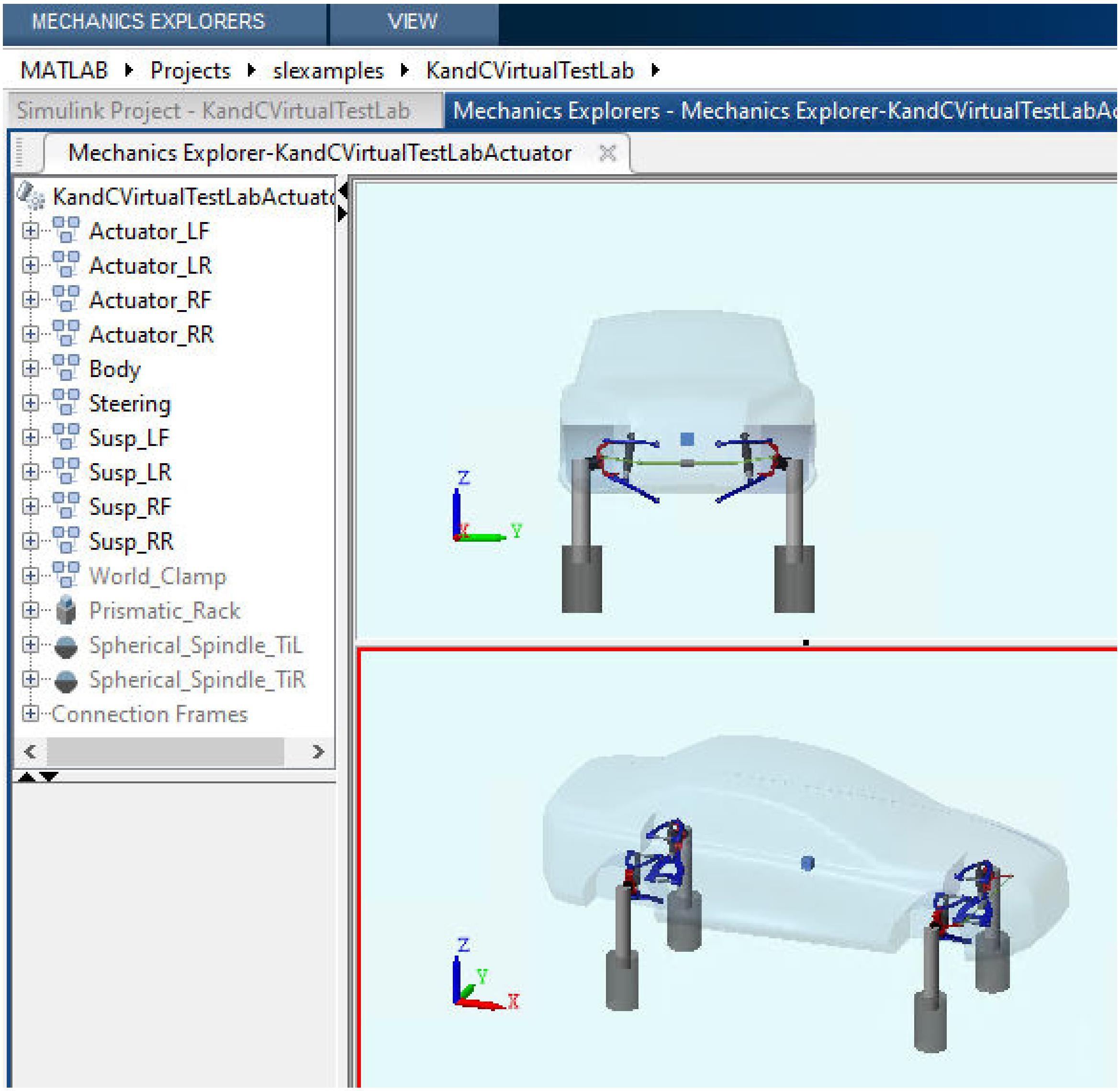
Steering angle breakpoints, f_susp_strgdelta_bp	`f_susp_strgdelta_bp`	Steering angle breakpoints, *O*, in rad.

Compare Mapped and Simscape Suspension Responses

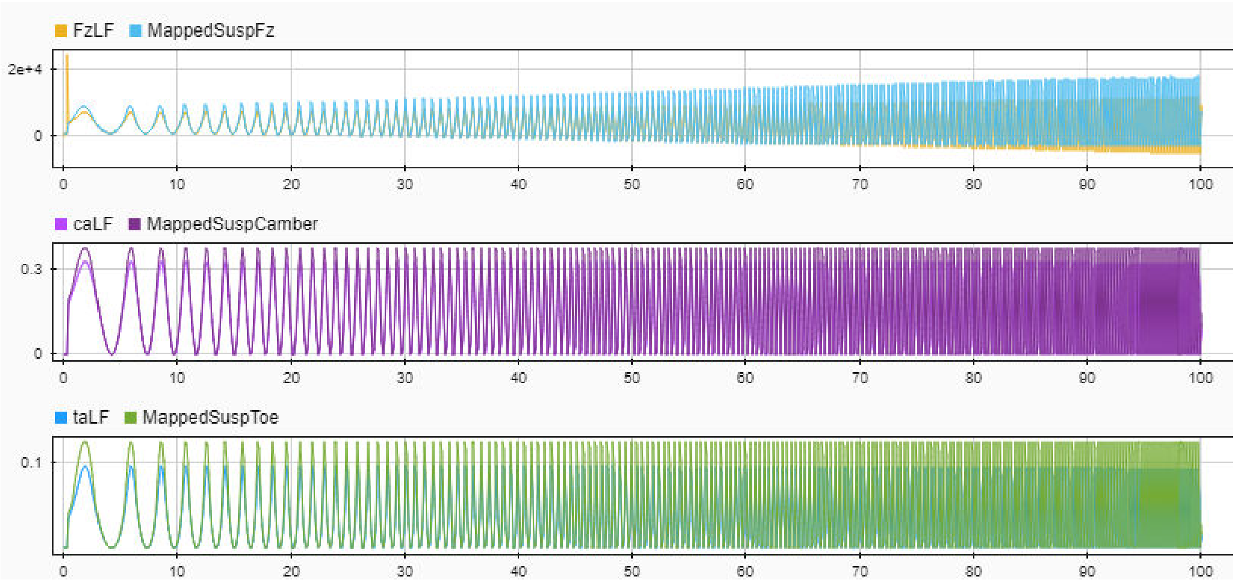
The virtual test laboratory stimulates the Simscape Multibody suspension at one operating point and then compares the response to the mapped suspension.

- To stimulate the Simscape Multibody suspension model, the test laboratory uses with the DOE settings contained in the `DOEMAT` array.

  During the simulation, to view the suspension system, select the **Mechanics Explorers** tab.



- After the simulation completes, use the Simulation Data Inspector to compare the suspension system response for the mapped suspension and Simscape Multibody suspension model. For example, compare the vertical force, camber angle, and toe angle responses.



See Also

Independent Suspension - Mapped | Solid Axle Suspension - Mapped

More About

- Simulation Data Inspector

Send and Receive Double-Lane Change Scene Data

This example shows you how to use the Simulation 3D Message Set and Simulation 3D Message Get blocks to communicate with the 3D visualization environment when you run the double-lane change maneuver. Specifically, you use the:

- Simulation 3D Message Get block to retrieve how many cones the vehicle hits during the maneuver.
- Simulation 3D Message Set block to control the traffic signal light.

For the minimum hardware required to run the example, see the “3D Visualization Engine Requirements” on page 1-4.

Run a Double-Lane Change Maneuver That Hits Cones

With the 3D visualization environment enabled, run a double-lane change maneuver that hits the cones.

1. Create and open a working copy of the double-lane change reference application project.

   ```
   vdynblksDblLaneChangeStart
   ```
2. Enable the 3D visualization environment. In the Visualization subsystem, open the 3D Engine block mask and select **Enabled**. Apply the changes and save the model.

   Alternatively, at the MATLAB command prompt, enter this code.

   See Code That Enables 3D Environment

   ```
   % Enable the 3D visualization environment
   mdl = 'DLCReferenceApplication';
   set_param([mdl '/Visualization/3D Engine'],'engine3D','Enabled');
   save_system(mdl)
   ```
3. In the top level of the model, set the Lane Change Reference Generator block parameters so that the vehicle does not successfully complete the maneuver. Set these block parameters, apply the changes, and save the model.

   - **Maneuver start time** to `5`.
   - **Longitudinal entrance velocity setpoint** to `50`.

   Alternatively, at the MATLAB command prompt, enter this code.

See Code That Sets Parameters

```
% Set Lane Change Reference Generator block parameters
mdl = 'DLCReferenceApplication';
set_param([mdl '/Lane Change Reference Generator'],'t_start','5');
set_param([mdl '/Lane Change Reference Generator'],'xdot_r','50');
save_system(mdl)
```

4. Run the maneuver. As the simulation runs, in the AutoVrtlEnv window, you can see the vehicle hitting the cones in the left lane.

Use Simulation 3D Message Get Block to Retrieve Cone Data

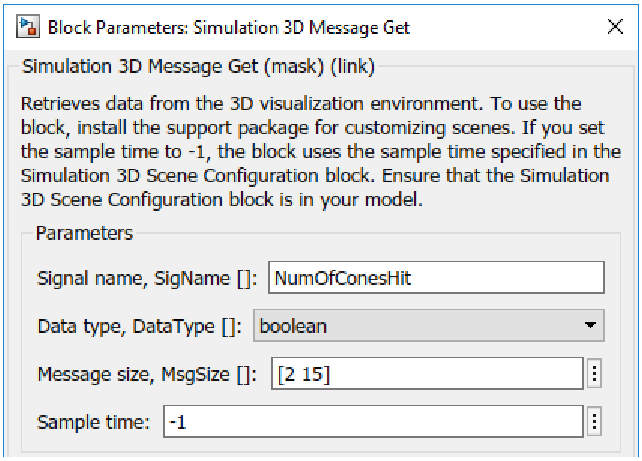
Use the Simulation 3D Message Get block to retrieve how many cones the vehicle hits during the maneuver. By default, the maneuver uses the double-lane change scene.

1. Navigate to the Visualization > 3D Engine subsystem. Right-click the 3D Engine block and select **Mask** > **Look Under Mask**. In the Visualization > 3D Engine > 3D Engine subsystem, insert these blocks:

   - Simulation 3D Message Get
   - Display

2 Set the Simulation 3D Message Get block parameters so that the block retrieves cone data from the double-lane change scene. Set these block parameters, apply the changes, and save the model.

- **Signal name, SigName** to `NumOfConesHit`
- **Data type, DataType** to `boolean`
- **Message size, MsgSize** to `[2 15]`
- **Sample time** to `-1`

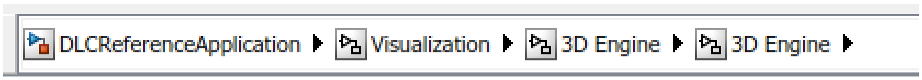


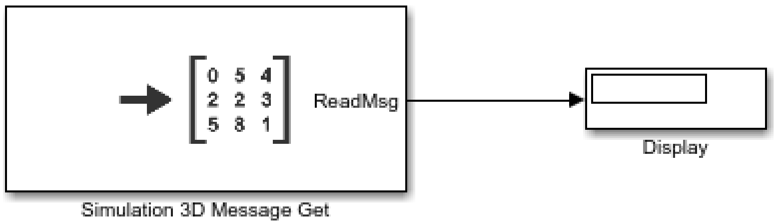
Alternatively, at the MATLAB command prompt, enter this code.

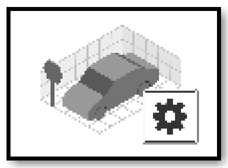
See Code That Sets Parameters

```
% Set these Simulation 3D Message Get block parameters
visualss='DLCReferenceApplication/Visualization/3D Engine/3D Engine';
set_param([visualss '/Simulation 3D Message Get'],'SigName','NumOfConesHit');
set_param([visualss '/Simulation 3D Message Get'],'DataType','boolean');
set_param([visualss '/Simulation 3D Message Get'],'MsgSize','[2 15]');
set_param([visualss '/Simulation 3D Message Get'],'Ts','-1');
save_system(mdl)
```

3 Connect the Simulation 3D Message Get output port `ReadMsg` to the Display block. Confirm the block parameters and signal connection. Save the model.





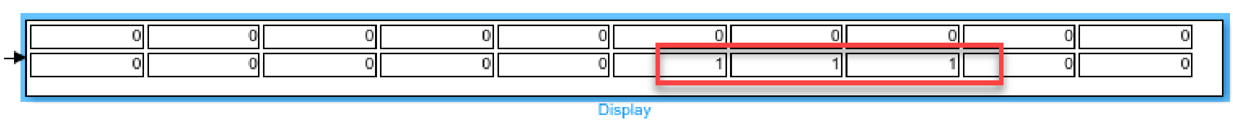


4 Verify that the Simulation 3D Scene Configuration block executes before the Simulation 3D Message Get block. That way, the Unreal Engine 3D visualization environment prepares the data before the Simulation 3D Message Get block receives it. To check the block execution order, right-click the blocks and select **Properties**. On the **General** tab, confirm these **Priority** settings:

- Simulation 3D Scene Configuration — `0`
- Simulation 3D Message Get — `1`

For more information about execution order, see “Control and Display the Execution Order” (Simulink).

5 Run the maneuver. As the simulation runs, the display block updates with the `ReadMsg` boolean value `1` when the vehicle hits the corresponding cone.



This table provides the Double Lane Change scene cone name that corresponds to the ReadMsg array element.

Simulation 3D Message Get Block ReadMsg Value	Unreal® Editor Cone Name	Simulation 3D Message Get Block Array Element	Unreal Editor Cone Name
ReadMsg(1,1)	SM_Cone5	ReadMsg(2,1)	SM_Cone10
ReadMsg(1,2)	SM_Cone4	ReadMsg(2,2)	SM_Cone09
ReadMsg(1,3)	SM_Cone3	ReadMsg(2,3)	SM_Cone08
ReadMsg(1,4)	SM_Cone2	ReadMsg(2,4)	SM_Cone07
ReadMsg(1,5)	SM_Cone01	ReadMsg(2,5)	SM_Cone06
ReadMsg(1,6)	SM_Cone15	ReadMsg(2,6)	SM_Cone20
ReadMsg(1,7)	SM_Cone14	ReadMsg(2,7)	SM_Cone19
ReadMsg(1,8)	SM_Cone13	ReadMsg(2,8)	SM_Cone18
ReadMsg(1,9)	SM_Cone12	ReadMsg(2,9)	SM_Cone17
ReadMsg(1,10)	SM_Cone11	ReadMsg(2,10)	SM_Cone16
ReadMsg(1,11)	SM_Cone25	ReadMsg(2,11)	SM_Cone30
ReadMsg(1,12)	SM_Cone24	ReadMsg(2,12)	SM_Cone29
ReadMsg(1,13)	SM_Cone23	ReadMsg(2,13)	SM_Cone28
ReadMsg(1,14)	SM_Cone22	ReadMsg(2,14)	SM_Cone27
ReadMsg(1,15)	SM_Cone21	ReadMsg(2,15)	SM_Cone26

The results indicate that the vehicle hits SM_Cone20, SM_Cone19, and SM_Cone18 during the maneuver.

Use Simulation 3D Message Set Block to Control Traffic Signal Light

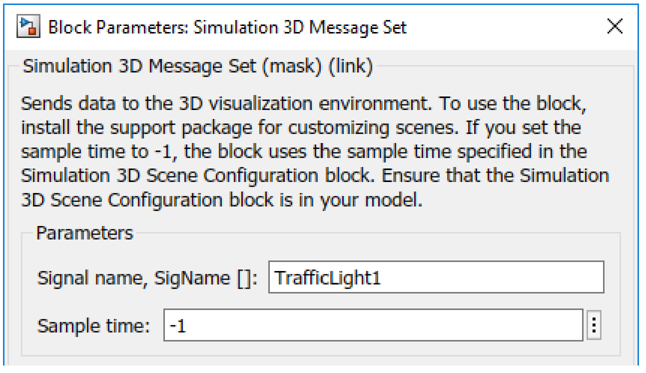
1. Navigate to the Visualization > 3D Engine subsystem. Right-click the 3D Engine block and select **Mask** > **Look Under Mask**. In the Visualization > 3D Engine > 3D Engine subsystem, insert these blocks:

   - Simulation 3D Message Set

- Data Type Conversion
- Stair Generator

2 Set the Simulation 3D Message Set block parameters so that the block sends traffic signal data to the double-lane change scene. Set these block parameters, apply the changes, and save the model.

   - **Signal name, SigName** to `TrafficLight1`
   - **Sample time** to `-1`



This table provides the scene traffic signal light color that corresponds to the `WriteMsg` value in the Double Lane Change scene.

Simulation 3D Message Set Block `WriteMsg` Value	`TrafficLight1` Color
`0`	`Red`
`1`	`Yellow`
`2`	`Green`

Alternatively, at the MATLAB command prompt, enter this code.

See Code That Sets Parameters

```
% Set Simulation 3D Message Set block parameters
visualss='DLCReferenceApplication/Visualization/3D Engine/3D Engine';
set_param([visualss  '/Simulation 3D Message Set'],'SigName','TrafficLight1');
set_param([visualss  '/Simulation 3D Message Set'],'Ts','-1');
save_system(mdl)
```

3 Set the Data Type Conversion block parameters to convert data to `int32`. Set these block parameters, apply the changes, and save the model.

- **Output data type** to `int32`

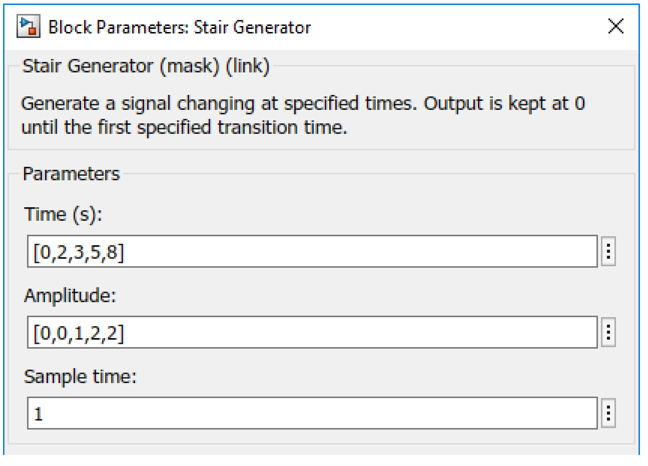
Alternatively, at the MATLAB command prompt, enter this code.

See Code That Sets Parameters

```
% Set Data Type Conversion block parameters
visualss='DLCReferenceApplication/Visualization/3D Engine/3D Engine';
set_param([visualss  '/Data Type Conversion'],'OutDataTypeStr','int32');
save_system(mdl)
```

4 Set the Stair Generator block parameters to send a command that corresponds to red, yellow, and green traffic light signals. Set these block parameters, apply the changes, and save the model.

- **Time** to `[0,2,3,5,8]`
- **Amplitude** to `[0,0,1,2,2]`
- **Sample time** to `1`



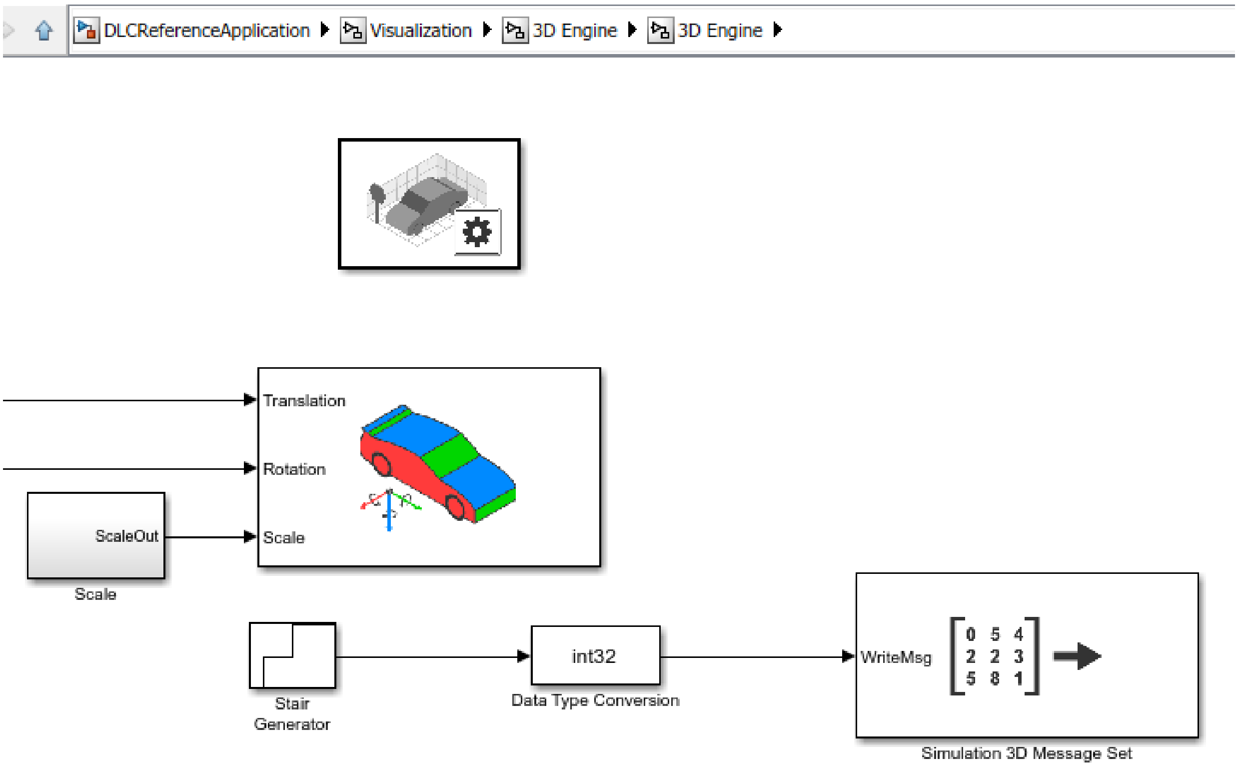
Alternatively, at the MATLAB command prompt, enter this code. Apply the block changes and save the model.

See Code That Sets Parameters

```
% Set Stair Generator block parameters
visualss='DLCReferenceApplication/Visualization/3D Engine/3D Engine';
```

```
open_system([visualss  '/Stair Generator']);
set_param([visualss  '/Stair Generator'],'e','[0,0,1,2,2]');
set_param([visualss  '/Stair Generator'],'t','[0,2,3,5,8]');
set_param([visualss  '/Stair Generator'],'Ts','1');
```

5 Connect the blocks as shown. Confirm the block parameters and signal connections. Save the model.

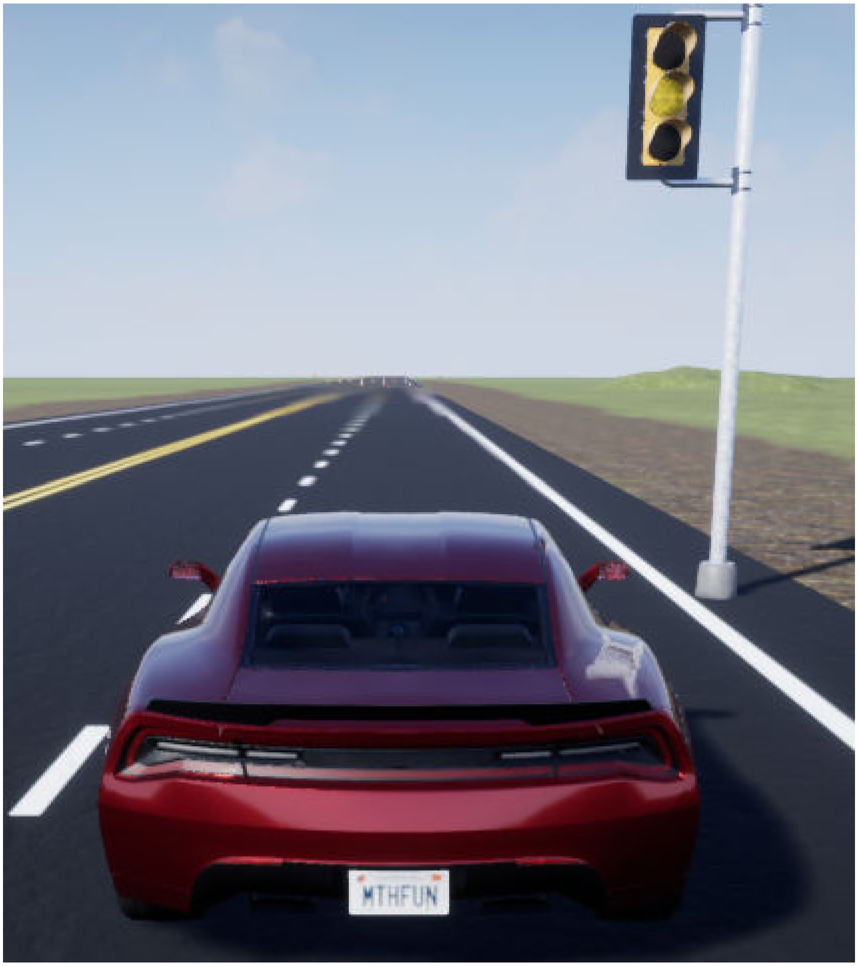


6 Verify that the Simulation 3D Message Set block executes before the Simulation 3D Scene Configuration block. That way, Simulation 3D Message Set prepares the signal data before the Unreal Engine 3D visualization environment receives it. To check the block execution order, right-click the blocks and select **Properties**. On the **General** tab, confirm these **Priority** settings:

- Simulation 3D Scene Configuration — `0`
- Simulation 3D Message Set — `-1`

For more information about execution order, see “Control and Display the Execution Order” (Simulink).

7 Run the maneuver. As the simulation runs, in the AutoVrtlEnv window, you can see the `TrafficLight1` light change from red to yellow to green.



Time Range (s)	`WriteMsg` Value	`TrafficLight1` Color
0–3	`0`	Red

Time Range (s)	`WriteMsg` Value	`TrafficLight1` Color
3–5	1	`Yellow`
5–8	2	`Green`

See Also

3D Engine | **Double Lane Change** | Simulation 3D Message Get | Simulation 3D Message Set | Simulation 3D Scene Configuration

Related Examples

- “Double Lane Change Reference Application” on page 7-4
- “Yaw Stability on Varying Road Surfaces” on page 1-27

More About

- “Support Package for Customizing Scenes” on page 6-3
- “3D Visualization Engine Requirements” on page 1-4

4

Project Templates

Vehicle Dynamics Blockset Project Templates

Vehicle Dynamics Blockset provides preassembled vehicle dynamics models that you can use to analyze the dynamic system response to common ride and handling tests. Use the templates to create vehicle dynamic model variants for the maneuver reference applications. Open project files that contain the vehicle models from the Simulink start page.

1. In Simulink, on the **Simulation** tab, select **File** > **New** > **Project** > **New Project**.

   In the Simulink start page, browse to Vehicle Dynamics Blockset and select **Passenger 3DOF Vehicle**, **Passenger 7DOF Vehicle**, or **Passenger 14DOF Vehicle**.
2. In the Create Project dialog box, in **Project name**, enter a project name.
3. In **Folder**, enter a project folder or browse to the folder to save the project.
4. Click **OK**.

   If the folder does not exist, the dialog box prompts you to create it. Click **Yes**.

   The software compiles the project and populates the project folders. All models and supporting files are in place for you to customize your vehicle dynamics model.

This table summarizes the vehicle dynamics project templates.

<table>
<tr><th>Vehicle Model</th><th>Description</th><th>Vehicle Body Degrees-of-Freedom (DOFs)</th><th>Wheel DOFs</th></tr>
<tr>
<td>Passenger 14DOF Vehicle</td>
<td>Vehicle with four wheels
Available as model variant in the maneuver reference applications</td>
<td>Six
<table>
<tr><th colspan="2">Translational</th><th colspan="2">Rotational</th></tr>
<tr><td>Longitudinal</td><td>✓</td><td>Pitch</td><td>✓</td></tr>
<tr><td>Lateral</td><td>✓</td><td>Yaw</td><td>✓</td></tr>
<tr><td>Vertical</td><td>✓</td><td>Roll</td><td>✓</td></tr>
</table></td>
<td>Two per wheel - eight total
<table>
<tr><th colspan="2">Translational</th><th colspan="2">Rotational</th></tr>
<tr><td>Vertical</td><td>✓</td><td>Rolling</td><td>✓</td></tr>
</table></td>
</tr>
</table>

<table>
<tr><th>Vehicle Model</th><th>Description</th><th>Vehicle Body Degrees-of-Freedom (DOFs)</th><th>Wheel DOFs</th></tr>
<tr><td>Passenger 7DOF Vehicle</td><td>Vehicle with four wheels
Available as model variant in the maneuver reference applications</td><td>Three<table><tr><th colspan="2">Translational</th><th colspan="2">Rotational</th></tr><tr><td>Longitudinal</td><td>✓</td><td>Pitch</td><td></td></tr><tr><td>Lateral</td><td>✓</td><td>Yaw</td><td>✓</td></tr><tr><td>Vertical</td><td></td><td>Roll</td><td></td></tr></table></td><td>One per wheel - four total<table><tr><th colspan="2">Rotational</th></tr><tr><td>Rolling</td><td>✓</td></tr></table></td></tr>
<tr><td>Passenger 3DOF Vehicle</td><td>Vehicle with ideal tire</td><td>Three<table><tr><th colspan="2">Translational</th><th colspan="2">Rotational</th></tr><tr><td>Longitudinal</td><td>✓</td><td>Pitch</td><td></td></tr><tr><td>Lateral</td><td>✓</td><td>Yaw</td><td>✓</td></tr><tr><td>Vertical</td><td></td><td>Roll</td><td></td></tr></table></td><td>None</td></tr>
</table>

See Also

More About

- “Double-Lane Change Maneuver” on page 3-4
- “Slowly Increasing Steering Maneuver” on page 3-28
- “Swept-Sine Steering Maneuver” on page 3-19

5

Maneuver Standards

ISO 15037-1:2006 Standard Measurement Signals

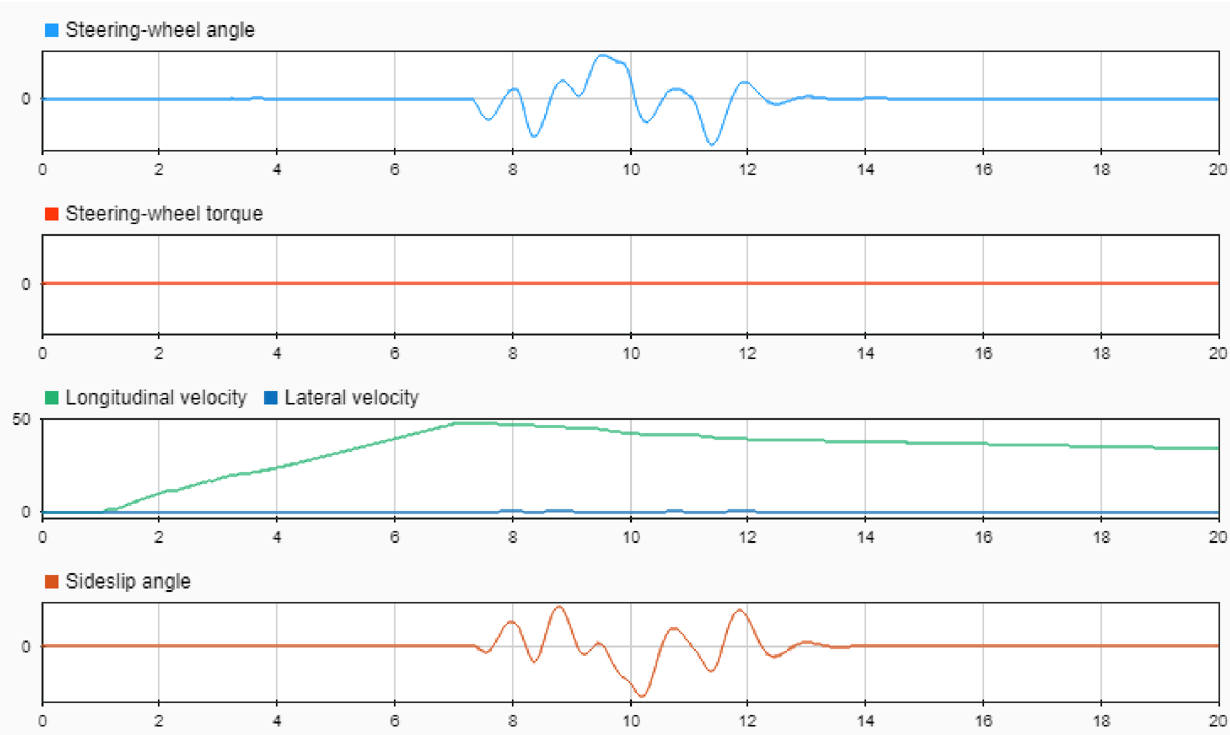
You can configure the maneuver reference applications to display ISO 15037-1:2006[1] standard measurement signals in the Simulation Data Inspector, including steering wheel angle and torque, longitudinal and lateral velocity, and sideslip angle.

To configure the ISO signal display, in the reference application Visualization subsystem, open the ISO 15037-1:2006 block. Select **Enabled**. After you run the maneuver, the Simulation Data Inspector opens with standard measurements.

For example, to display the ISO signals when you run the double lane change maneuver:

1. Create and open a working copy of the double-lane change reference application project.

   ```
   vdynblksDblLaneChangeStart
   ```
2. In the Visualization subsystem, open the ISO 15037-1:2006 block. Select **Enabled**. Save the reference application.
3. Run the maneuver. As the simulation runs, view the ISO standard measurement signals in the Simulation Data Inspector, including steering wheel angle and torque, longitudinal and lateral velocity, and sideslip angle.



References

[1] ISO 15037-1:2006. *Road vehicles -- Vehicle dynamics test methods -- Part 1: General conditions for passenger cars*. ISO (International Organization for Standardization), 2014.

See Also

More About

- "Double-Lane Change Maneuver" on page 3-4
- "Slowly Increasing Steering Maneuver" on page 3-28
- "Swept-Sine Steering Maneuver" on page 3-19

- Simulation Data Inspector

External Websites

- International Organization for Standardization

6

Supporting Data

Support Package For Maneuver and Drive Cycle Data

This example shows how to install additional maneuver and drive cycle data from a support package. By default, the Drive Cycle Source block has the FTP-75 drive cycle data. The support package has drive cycles that include the gear shift schedules, for example `JC08` and `CUEDC`.

1. In the Drive Cycle Source block, click **Install additional drive cycles** to start the installer.
2. Follow the instructions and default settings provided by the installer to complete the installation.
3. On the **Select a support package** screen, select the data you want to add:

   Accept or change the **Installation folder** and click **Next**.

   Note You must have write permission for the Installation folder.

See Also

Drive Cycle Source

Support Package for Customizing Scenes

To customize Vehicle Dynamics Blockset scenes, you can use the Vehicle Dynamics Blockset Interface for Unreal Engine 4 Projects support package. The support package contains an Unreal Engine project that allows you to customize these Vehicle Dynamics Blockset scenes:

- Curved Road
- Double Lane Change
- Open Surface
- Large Parking Lot
- Parking Lot
- Straight Road
- US City Block
- US Highway

Optionally, after you customize the scenes, you can create an Unreal Engine project executable file that can improve Simulink co-simulation performance with the Unreal Editor.

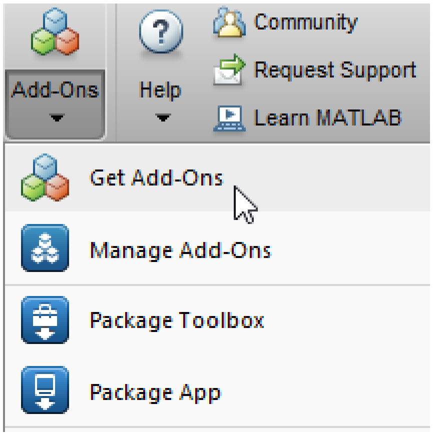
This table provides the setup steps for customizing the scenes.

Setup Step	Description
"Install Support Package" on page 6-4	Use the Add-On Explorer to install the support package that contains the Unreal Engine 4.19 project file.
"Install Unreal Engine" on page 6-5	To customize the scenes, you need Unreal Engine 4.19.
"Set Up Environment and Open Unreal Editor" on page 6-5	Set up the 3D simulation environment to use the project file.
"Configure Simulation 3D Scene Configuration Block for Unreal Editor Co-Simulation" on page 6-7	Configure co-simulation with the Unreal Editor.

Setup Step	Description
"Use Unreal Editor to Customize Scenes" on page 6-8	Use the Unreal Editor to customize the scenes in the project. You can use the Simulation 3D Message Set and Simulation 3D Message Get blocks to send and receive scene data from the Unreal Engine 3D visualization environment. You must configure the visualization environment to communicate with the Simulink model data. For information, see the block documentation and the support package file *QuickStart_Message_Get_Set*.
Optionally, "Configure Simulation 3D Scene Configuration Block to Run Project Executable" on page 6-9	Create an Unreal Engine project executable file that contains your updates. To improve co-simulation performance, consider configuring the Simulation 3D Scene Configuration block to co-simulate with the project executable.

Install Support Package

1 On the MATLAB **Home** tab, in the Environment section, select **Add-Ons** > **Get Add-Ons**.



2. In the Add-On Explorer window, search for the Vehicle Dynamics Blockset Interface for Unreal Engine 4 Projects support package. Click **Install**.

   Note You must have write permission for the Installation folder.

Install Unreal Engine

To customize the scenes, you need Unreal Engine 4.19.

For installation information, see Unreal Engine.

Set Up Environment and Open Unreal Editor

The support package includes the Unreal Engine 4.19 `AutoVrtlEnv.uproject` project file. You can use the project as a template for customizing a scene.

Follow one-time steps 1 through 4 to set up the 3D simulation environment to use the project files. To open the Unreal Editor, follow steps 5 and 6.

The following section provides code that sets up the environment and opens the Unreal Editor.

<table>
<tr><th colspan="2">Step</th><th>Description</th></tr>
<tr><td rowspan="4">One-time steps that set up the 3D simulation environment</td><td>1</td><td>Specify the location of the support package project files and a local folder destination.

Note You must have write permission for the local folder destination.</td></tr>
<tr><td>2</td><td>Specify the location of the Unreal Engine installation, for example <code>C:\Program Files\Epic Games\UE_4.xx</code>.</td></tr>
<tr><td>3</td><td>Copy the <code>MathWorksSimulation</code> plugin to the Unreal Engine plugin folder.</td></tr>
<tr><td>4</td><td>Copy the support package folder that contains the <code>AutoVrtlEnv.uproject</code> files to the local folder destination.</td></tr>
</table>

Step		Description
Step that opens the Unreal Editor	5	Create an instance of the `sim3d.Editor` class for the project. Open the project in the Unreal Editor. This might take some time. **Tip** Make sure that you associate `.uproject` files with the Unreal Editor so that the Unreal Editor opens when you select `.uproject` files.

Code That Sets Up Environment (Steps 1–4)

```
%% STEP1
% Specify the location of the support package project files and a local folder destination
% Note:  Only one path is supported. Select latest download path.
dest_root = "C:\Local";
src_root = fullfile(matlabshared.supportpkg.getSupportPackageRoot, ...
    "toolbox", "shared", "sim3dprojects", "automotive");

%% STEP2
% Specify the location of the Unreal Engine installation.
ueInstFolder = "C:\Program Files\Epic Games\UE_4.19";

%% STEP3
% Copy the MathWorksSimulation plugin to the Unreal Engine plugin folder.
mwPluginName = "MathWorksSimulation";
mwPluginFolder = fullfile(src_root, "PluginResources", "UE419"); % choose UE version
uePluginFolder = fullfile(ueInstFolder, "Engine", "Plugins");
uePluginDst = fullfile(uePluginFolder, "Marketplace", "MathWorks");

cd(uePluginFolder)
foundPlugins = dir("**/" + mwPluginName + ".uplugin");

if ~isempty(foundPlugins)
    numPlugins = size(foundPlugins, 1);
    msg2 = cell(1, numPlugins);
    pluginCell = struct2cell(foundPlugins);

    msg1 = "Plugin(s) already exist here:" + newline + newline;
    for n = 1:numPlugins
        msg2{n} = "    " + pluginCell{2,n} + newline;
    end
    msg3 = newline + "Please remove plugin folder(s) and try again.";
    msg  = msg1 + msg2 + msg3;
    warning(msg);
else
    copyfile(mwPluginFolder, uePluginDst);
    disp("Successfully copied MathWorksSimulation plugin to UE4 engine plugins!")
end

%% STEP4
% Copy the support package folder that contains the AutoVrtlEnv.uproject
```

```
% files to the local folder destination.
projFolderName = "AutoVrtlEnv";
projSrcFolder = fullfile(src_root, projFolderName);
projDstFolder = fullfile(dest_root, projFolderName);
if ~exist(projDstFolder, "dir")
    copyfile(projSrcFolder, projDstFolder);
end
```

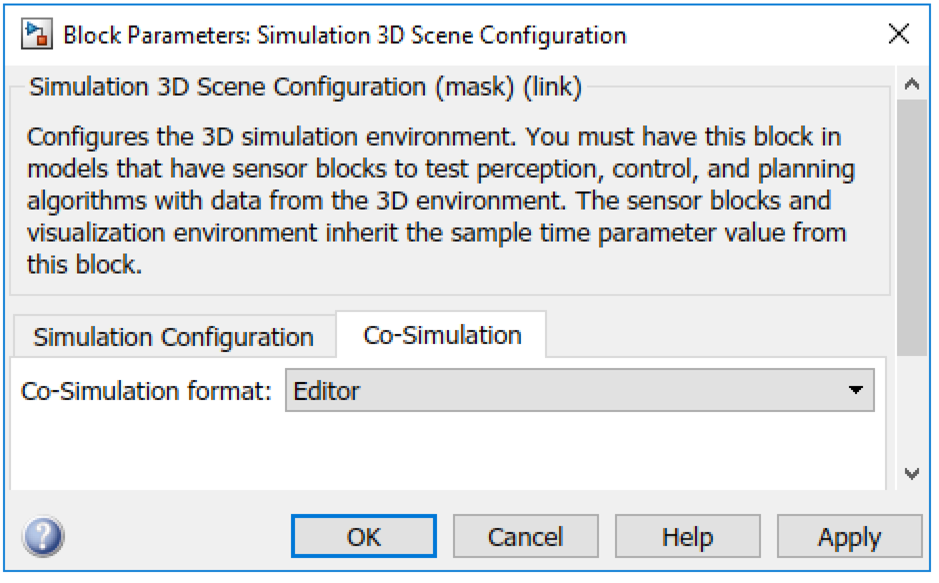
Code That Opens Unreal Editor (Step 5)

```
%% STEP5
% Create an instance the of sim3d.Editor class for the project.
% The project is located in the projDstFolder and has the projFolder name.
dest_root="C:\Local";
projFolderName = "AutoVrtlEnv";
projDstFolder = fullfile(dest_root, projFolderName);
editor = sim3d.Editor(fullfile(projDstFolder, projFolderName + ".uproject"));

% Open the project in the Unreal Editor.
editor.open();
```

Configure Simulation 3D Scene Configuration Block for Unreal Editor Co-Simulation

To set up Simulink co-simulation with the Unreal Editor, in your Simulink model, navigate to the Simulation 3D Scene Configuration block. Set **Co-simulation format** to `Editor`.



To run the simulation, in Simulink, click **Run**. Before you select **Play** in the Unreal Editor, wait until the Diagnostic Viewer window displays this confirmation message:

```
In the Simulation 3D Scene Configuration block, you set the co-simulation format to 'Editor'.
In Unreal Editor, select 'Play' to view the scene.
```

This message confirms that Simulink has instantiated the scene actors, including the vehicles and cameras, in the Unreal Engine 3D environment. If you select **Play** before the confirmation message appears, Simulink might not instantiate the actors in the Unreal Editor.

Use Unreal Editor to Customize Scenes

Use the Unreal Editor to customize the scenes in the `AutoVrtlEnv.uproject` project. This table provides the Vehicle Dynamics Blockset scenes included in the project and the equivalent Unreal Editor map.

Vehicle Dynamics Blockset Scene	Unreal Editor Map
Curved Road	`HwCurve`
Double Lane Change	`DblLnChng`
Open Surface	`BlackLake`
Large Parking Lot	`LargeParkingLot`
Parking Lot	`SimpleLot`
Straight Road	`HwStrght`
US City Block	`USCityBlock`
US Highway	`USHighway`

Note The `AutoVrtlEnv.uproject` project does not include the Virtual Mcity® scene.

For information about using the Unreal Editor, see the Unreal Engine 4 Documentation.

Configure Scenes to Send Data

The Simulation 3D Message Get block retrieves data from the Unreal Engine 3D visualization environment. To use the block, you must configure scenes in the Unreal Engine environment to send data to the Simulink model.

For detailed information about using the block to customizing the scenes, see the support package file `QuickStart_Message_Get_Set` and the Simulation 3D Message Get documentation.

Configure Scenes to Receive Data

The Simulation 3D Message Set block sends data to the Unreal Engine 3D visualization environment. To use the block, you must configure scenes in the Unreal Engine environment to receive data from the Simulink model.

For detailed information about using the block to customizing the scenes, see the support package file `QuickStart_Message_Get_Set` and the Simulation 3D Message Set documentation.

Create a New Scene

In Unreal Editor, to create a scene:

1. In the sources panel, select **Content/Maps**.
2. Inside **Maps**, right-click **BlackLake**. Select Duplicate.
3. Rename the new map.

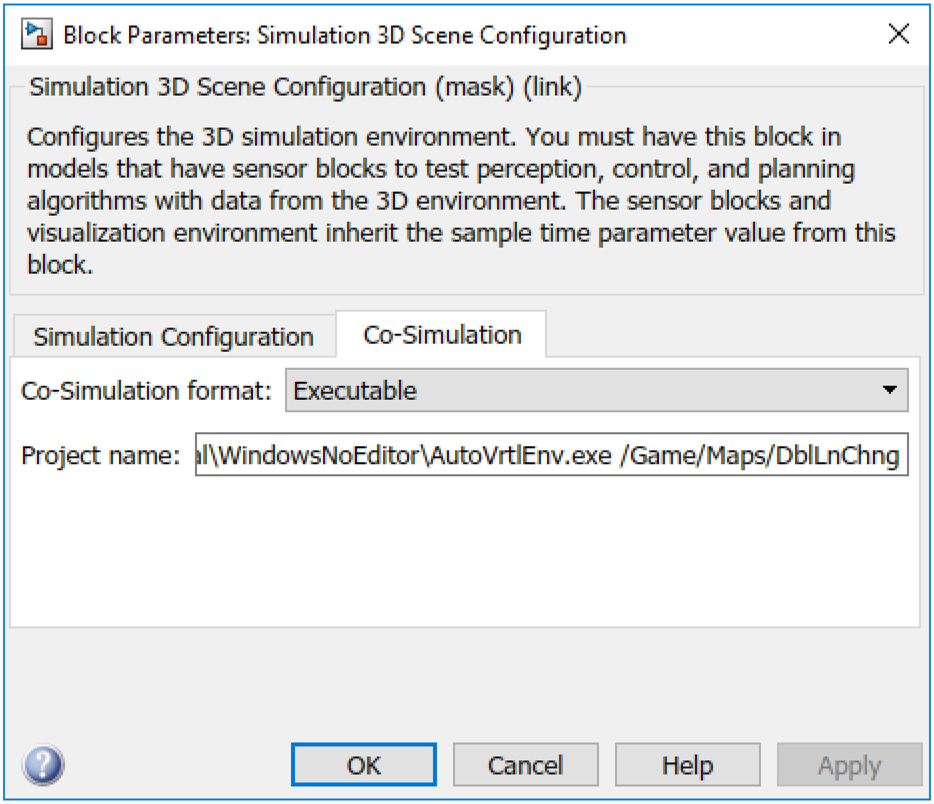
Configure Simulation 3D Scene Configuration Block to Run Project Executable

Optionally, after you customize the scenes, you can create an Unreal Engine project executable file that contains your updates. For information about creating the project executable, see Unreal Engine 4 Documentation.

To improve co-simulation performance, consider configuring the Simulation 3D Scene Configuration block to co-simulate with the project executable. In your Simulink model, navigate to the Simulation 3D Scene Configuration block.

- On the **Simulation Configuration** tab, set **Scene description** to `Custom`.
- On the **Co-Simulation** tab, set:
  - **Co-simulation format** to `Executable`
  - **Project name** — Path to the project executable followed by the path to the scene within the project

For example, to specify the `C:\Local\WindowsNoEditor\AutoVrtlEnv.exe` project executable and the double-lane change scene, set **Project name** to `C:\Local\WindowsNoEditor\AutoVrtlEnv.exe /Game/Maps/DblLnChng`.



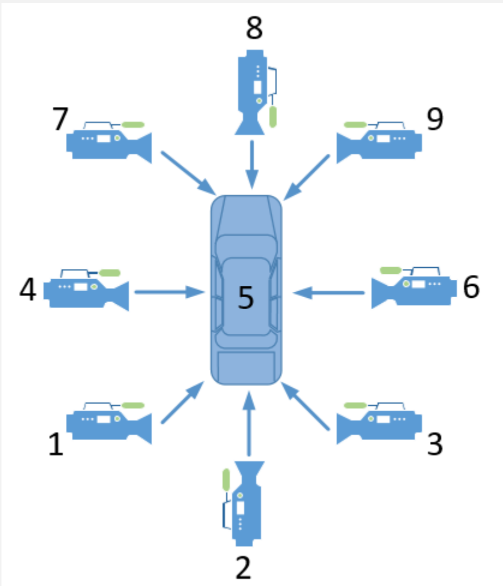
AutoVrtlEnv.uproject Keyboard Functions

The `AutoVrtlEnv.uproject` uses these key commands.

Key	Function
Tab	Switches focus to next vehicle actor

To change the camera views in the AutoVrtlEnv window, use these key commands.

Key	Camera View
1	Back left
2	Back
3	Back right
4	Left
5	Internal
6	Right
7	Front left
8	Front
9	Front right
0	Overhead



Troubleshooting

Startup Warning

If you start Unreal Engine without following the steps in “Set Up Environment and Open Unreal Editor” on page 6-5 and “Configure Simulation 3D Scene Configuration Block for Unreal Editor Co-Simulation” on page 6-7, you can get `Warning: Integration with MATLAB/Simulink is not active`. Unreal Engine might crash if you then continue working with the Unreal Editor.

Error Creating Project Executable

To create a project executable, the `MathWorksSimulation` plugin must be located in the Unreal Engine plugin folder. Check that the `MathWorksSimulation` plugin is not in the `AutoVrtlEnv` folder or subfolders. Set up the environment by following the steps in “Set Up Environment and Open Unreal Editor” on page 6-5.

See Also

Simulation 3D Message Get | Simulation 3D Message Set | Simulation 3D Scene Configuration | **Virtual Mcity** | `sim3d.Editor`

Related Examples

- “Send and Receive Double-Lane Change Scene Data” on page 3-65

More About

- “Vehicle Dynamics Blockset Communication with 3D Visualization Software” on page 1-6
- “3D Visualization Engine Requirements” on page 1-4

External Websites

- Unreal Engine
- Unreal Engine 4 Documentation

7

Vehicle Dynamics Blockset Examples

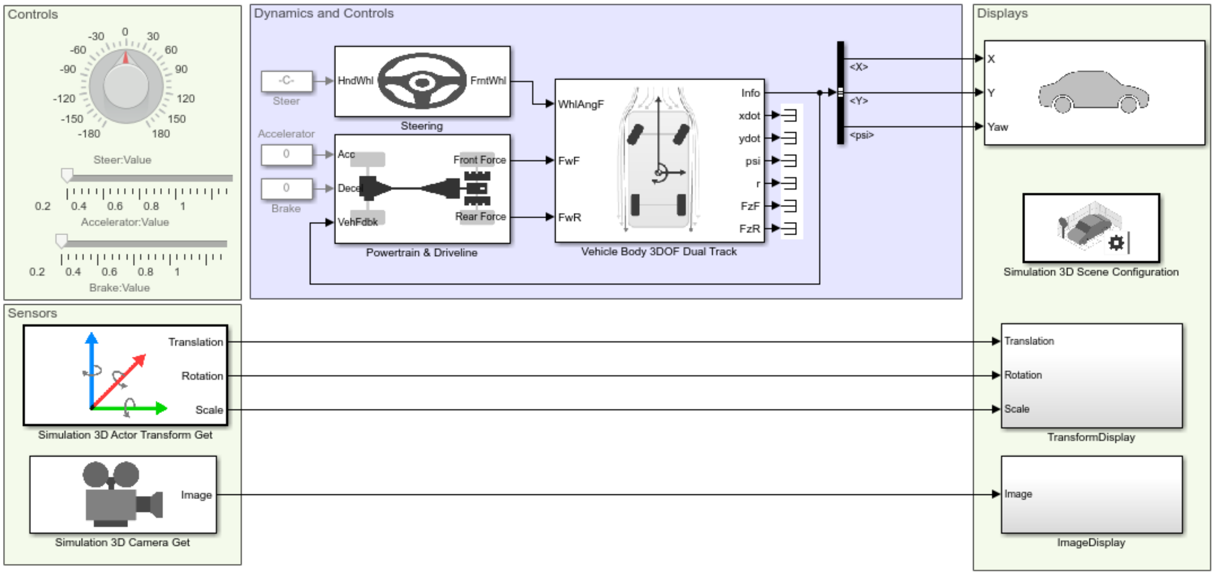
Scene Interrogation with Camera and Ray Tracing Reference Application

Interrogate a 3D scene with a vehicle dynamics model by using a camera and ray tracing reference application project.

To create or modify other scenes, you need the Vehicle Dynamics Blockset Interface for Unreal Engine 4 Projects support package. For more information, see "Support Package for Customizing Scenes" on page 6-3.

For the minimum hardware required to run the example, see "3D Visualization Engine Requirements" on page 1-4.

For more information about the reference application, see "Scene Interrogation in 3D Environment" on page 3-14.



See Also

Bicycle Model | Simulation 3D Camera Get | Simulation 3D Config | Simulation 3D Vehicle with Ground Following | **Straight Road**

More About

- "3D Visualization Engine Requirements" on page 1-4
- "Vehicle Dynamics Blockset Communication with 3D Visualization Software" on page 1-6

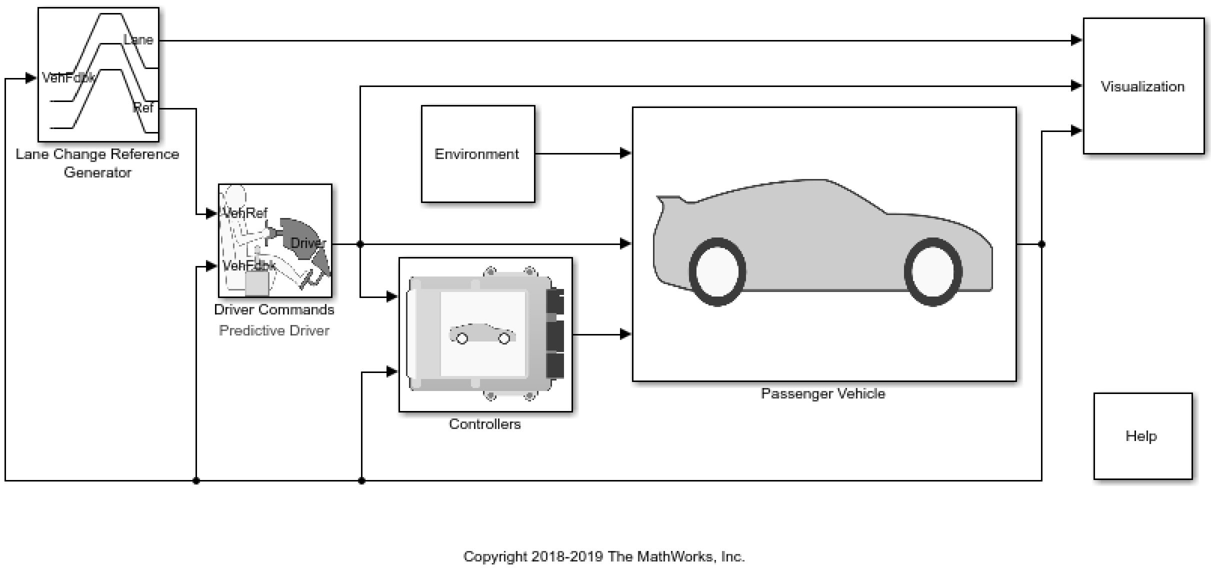
External Websites

- Unreal Engine

Double Lane Change Reference Application

Simulate a full vehicle dynamics model undergoing a double-lane change maneuver according to standard ISO 3888-2. You can create your own versions, establishing a framework to test that your vehicle meets the design requirements under normal and extreme driving conditions. Use the reference application for vehicle dynamics ride and handling analysis and chassis controls development, including yaw stability and lateral acceleration limits.

For more information about the reference application, see “Double-Lane Change Maneuver” on page 3-4.



See Also

3D Engine | Mapped SI Engine | Predictive Driver | Vehicle Terrain Sensor

Related Examples

- “Send and Receive Double-Lane Change Scene Data” on page 3-65

- “Yaw Stability on Varying Road Surfaces” on page 1-27

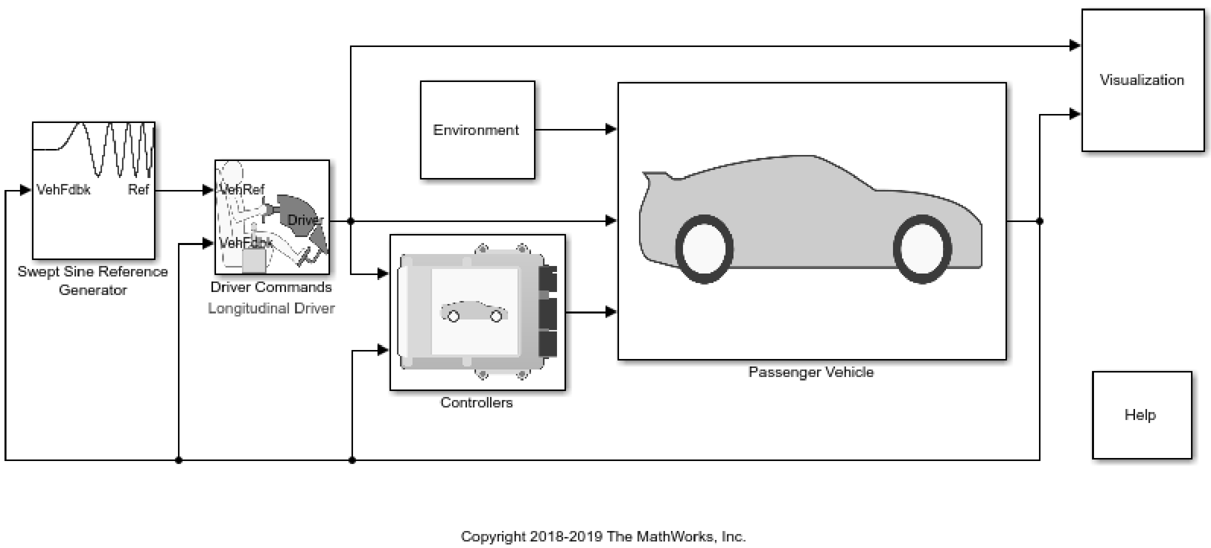
More About

- “Coordinate Systems in Vehicle Dynamics Blockset” on page 2-2
- Simulation Data Inspector

Swept Sine Steering Reference Application

Simulate a full vehicle dynamics model undergoing a swept-sine steering maneuver. You can create your own versions, providing a framework to test that your vehicle meets the design requirements under normal and extreme driving conditions. Use the reference application for vehicle dynamics ride and handling analysis and chassis controls development, including the dynamic steering response.

For more information about the reference application, see “Swept-Sine Steering Maneuver” on page 3-19.



See Also

3D Engine | Longitudinal Driver | Mapped SI Engine | Vehicle Terrain Sensor

Related Examples

- “Frequency Response to Steering Angle Input” on page 1-67

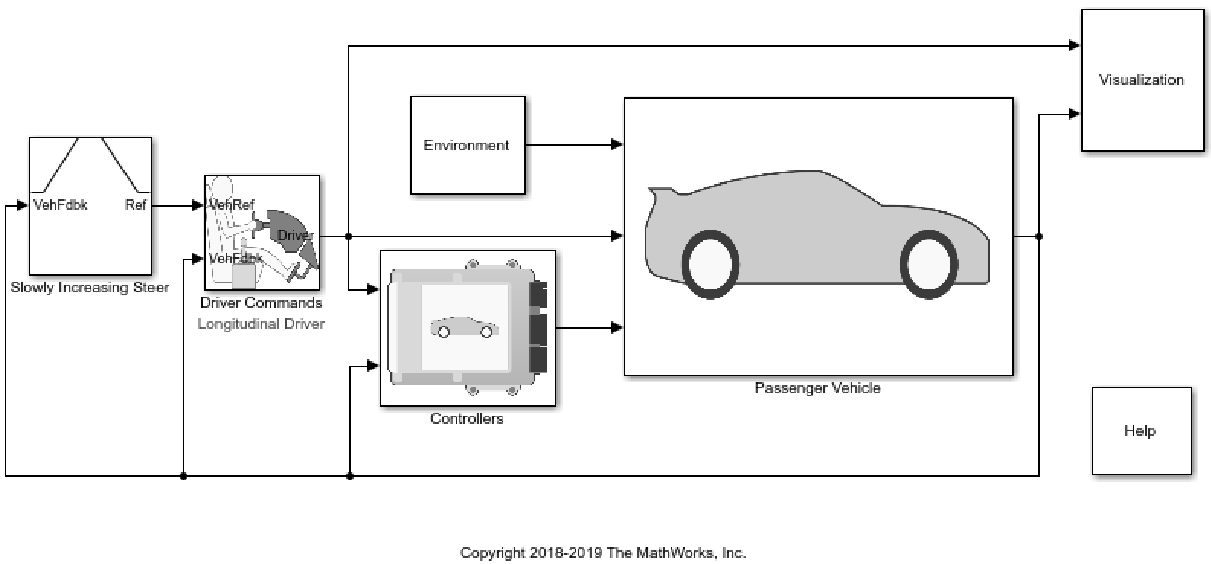
More About

- “Coordinate Systems in Vehicle Dynamics Blockset” on page 2-2
- Simulation Data Inspector

Increasing Steering Reference Application

Simulate a full vehicle dynamics model undergoing a slowly increasing steering maneuver according to standard SAE J266. You can create your own versions, establishing a framework to test that your vehicle meets the design requirements under normal and extreme driving conditions. Use the reference application for lateral vehicle dynamics ride and handling analysis and chassis controls development, including the steering response.

For more information about the reference application, see “Slowly Increasing Steering Maneuver” on page 3-28.



See Also

3D Engine | Longitudinal Driver | Mapped SI Engine | Vehicle Terrain Sensor

Related Examples

- “Vehicle Steering Gain at Different Speeds” on page 1-44

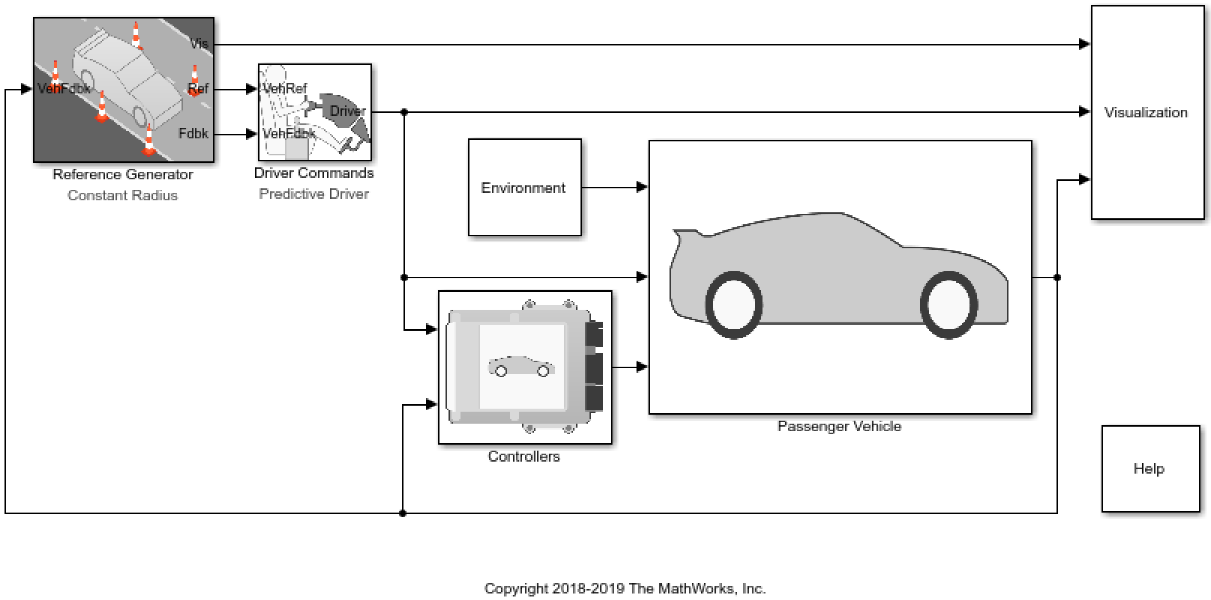
More About

- “Coordinate Systems in Vehicle Dynamics Blockset” on page 2-2
- Simulation Data Inspector

Constant Radius Reference Application

Simulate a full vehicle dynamics model undergoing a constant radius maneuver. You can create your own versions, providing a framework to test that your vehicle meets the design requirements under normal and extreme driving conditions. Use the reference application for vehicle dynamics ride and handling analysis and chassis controls development, including the dynamic steering response.

For more information about the reference application, see "Constant Radius Maneuver" on page 3-37.



References

[1] J266_199601. *Steady-State Directional Control Test Procedures for Passenger Cars and Light Trucks*. Warrendale, PA: SAE International, 1996.

[2] ISO 4138:2012. *Passenger cars — Steady-state circular driving behaviour — Open-loop test methods*. Geneva: ISO, 2012.

See Also

3D Engine | Driver Commands | Reference Generator

Related Examples

- “Vehicle Lateral Acceleration at Different Speeds” on page 1-56

More About

- “Coordinate Systems in Vehicle Dynamics Blockset” on page 2-2
- Simulation Data Inspector

Kinematics and Compliance Virtual Test Laboratory Reference Application

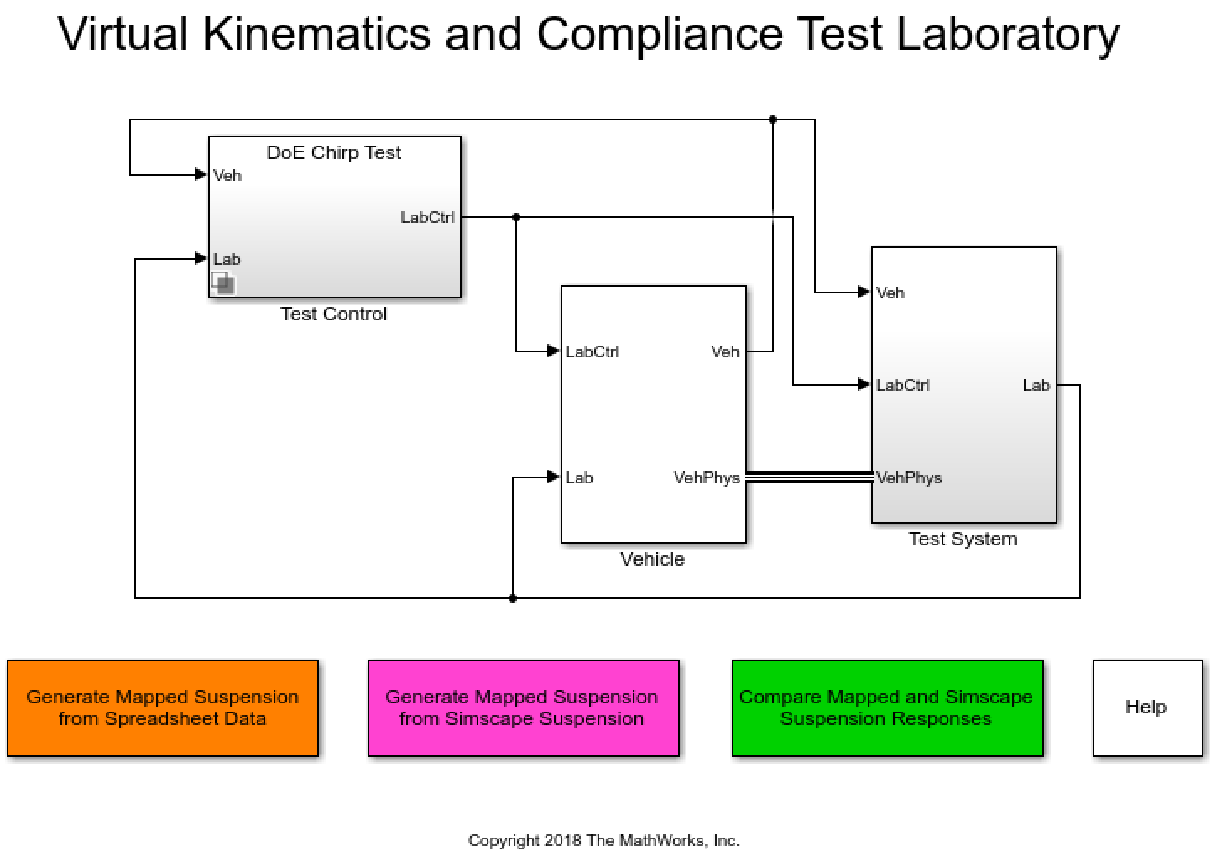
Generate optimized suspension parameters for the vehicle dynamics mapped suspension blocks.

Generate Mapped Suspension from Spreadsheet Data uses Model-Based Calibration Toolbox™ to generate calibrated suspension parameters from measured vertical force and suspension geometry data.

Generate Mapped Suspension from Simscape Suspension uses a Simscape™ Multibody™ suspension system to generate calibrated suspension parameters for the mapped suspension blocks.

Compare Mapped and Simscape Suspension Responses compares the mapped suspension with the Simscape Multibody suspension results.

For more information about the reference application, see “Kinematics and Compliance Virtual Test Laboratory” on page 3-54.



See Also

Independent Suspension - Mapped | Solid Axle Suspension - Mapped

More About

- Simulation Data Inspector