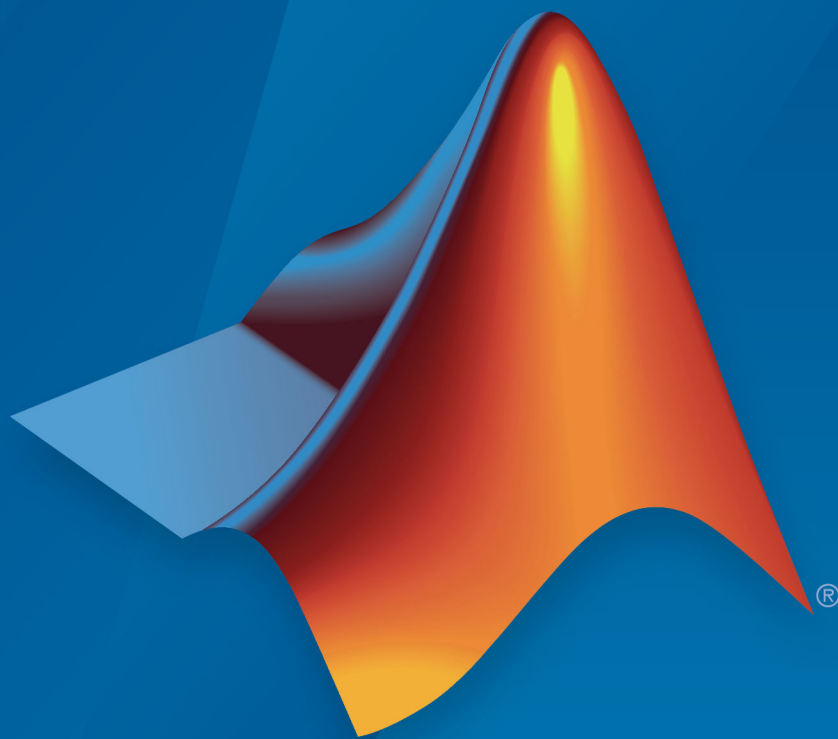


# Powertrain Blockset™

## User's Guide



MATLAB® & SIMULINK®

R2018b



# How to Contact MathWorks



Latest news: [www.mathworks.com](http://www.mathworks.com)  
Sales and services: [www.mathworks.com/sales\\_and\\_services](http://www.mathworks.com/sales_and_services)  
User community: [www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral)  
Technical support: [www.mathworks.com/support/contact\\_us](http://www.mathworks.com/support/contact_us)



Phone: 508-647-7000



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

*Powertrain Blockset™ User's Guide*

© COPYRIGHT 2016–2018 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

## Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [www.mathworks.com/trademarks](http://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

## Patents

MathWorks products are protected by one or more U.S. patents. Please see [www.mathworks.com/patents](http://www.mathworks.com/patents) for more information.

## Revision History

October 2016	Online only	New for Version 1.0 (Release 2016b+)
March 2017	Online only	Revised for Version 1.1 (Release 2017a)
September 2017	Online only	Revised for Version 1.2 (Release 2017b)
March 2018	Online only	Revised for Version 1.3 (Release 2018a)
September 2018	Online only	Revised for Version 1.4 (Release 2018b)

	<b>Getting Started</b>
<b>1</b>	
	<hr/>
	<b>Powertrain Blockset Product Description</b> . . . . . 1-2
	Key Features . . . . . 1-2
	<b>Required and Recommended Products</b> . . . . . 1-3
	Required Products . . . . . 1-3
	Recommended Products . . . . . 1-3
	<b>Getting Started with Powertrain Blockset</b> . . . . . 1-4
	Next Steps . . . . . 1-11
	<b>Conventional Vehicle Fuel Economy and Emissions</b> . . . . . 1-13

	<b>Workflows</b>
<b>2</b>	
	<hr/>
	<b>SI Core Engine Air Mass Flow and Torque Production</b> . . . . . 2-2
	Air Mass Flow Models . . . . . 2-3
	Torque Models . . . . . 2-3
	<b>SI Engine Dual-Independent Cam Phaser Air Mass Flow Model</b> . . . . . 2-5
	Collect Physical Measurements . . . . . 2-7
	Estimate Ideal Trapped Mass . . . . . 2-8
	Correct Trapped Mass . . . . . 2-9
	Calculate Air Mass Flow . . . . . 2-10
	<b>SI Engine Speed-Density Air Mass Flow Model</b> . . . . . 2-14
	<b>SI Engine Torque Structure Model</b> . . . . . 2-17

<b>SI Engine Simple Torque Model</b> .....	<b>2-26</b>
<b>CI Core Engine Air Mass Flow and Torque Production</b> .....	<b>2-27</b>
Air Mass Flow .....	<b>2-27</b>
Torque .....	<b>2-27</b>
<b>CI Engine Speed-Density Air Mass Flow Model</b> .....	<b>2-29</b>
<b>CI Engine Torque Structure Model</b> .....	<b>2-33</b>
Fuel Injection .....	<b>2-37</b>
Percent Oxygen .....	<b>2-37</b>
Exhaust Temperature .....	<b>2-37</b>
<b>CI Engine Simple Torque Model</b> .....	<b>2-40</b>
<b>Engine Calibration Maps</b> .....	<b>2-41</b>
Engine Plant Calibration Maps .....	<b>2-41</b>
Engine Controller Calibration Maps .....	<b>2-41</b>
Calibration Maps in Compression-Ignition (CI) Blocks .....	<b>2-42</b>
Calibration Maps in Spark-Ignition (SI) Blocks .....	<b>2-80</b>

## Reference Applications

### 3

<b>Internal Combustion Engine Reference Application Projects</b> .....	<b>3-2</b>
<b>Hybrid and Electric Vehicle Reference Application Projects</b> .....	<b>3-3</b>
<b>Explore the Conventional Vehicle Reference Application</b> .....	<b>3-4</b>
Drive Cycle Source .....	<b>3-5</b>
Longitudinal Driver .....	<b>3-5</b>
Controllers .....	<b>3-6</b>
Passenger Car .....	<b>3-7</b>
<b>Explore the CI Engine Dynamometer Reference Application</b> .....	<b>3-10</b>
Engine System .....	<b>3-12</b>
Performance Monitor .....	<b>3-13</b>

<b>Explore the SI Engine Dynamometer Reference</b>	
<b>Application</b> .....	<b>3-15</b>
Engine System .....	<b>3-17</b>
Performance Monitor .....	<b>3-18</b>
<b>Explore the Hybrid Electric Vehicle Multimode Reference</b>	
<b>Application</b> .....	<b>3-20</b>
Drive Cycle Source .....	<b>3-22</b>
Longitudinal Driver .....	<b>3-22</b>
Controllers .....	<b>3-23</b>
Passenger Car .....	<b>3-24</b>
<b>Explore the Electric Vehicle Reference Application</b> .....	<b>3-28</b>
Drive Cycle Source .....	<b>3-30</b>
Longitudinal Driver .....	<b>3-30</b>
Controllers .....	<b>3-31</b>
Passenger Car .....	<b>3-31</b>
<b>Explore the Hybrid Electric Vehicle Input Power-Split</b>	
<b>Reference Application</b> .....	<b>3-34</b>
Drive Cycle Source .....	<b>3-36</b>
Longitudinal Driver .....	<b>3-37</b>
Controllers .....	<b>3-37</b>
Passenger Car .....	<b>3-41</b>
<b>Explore the Hybrid Electric Vehicle P2 Reference</b>	
<b>Application</b> .....	<b>3-44</b>
Drive Cycle Source .....	<b>3-46</b>
Longitudinal Driver .....	<b>3-47</b>
Controllers .....	<b>3-47</b>
Passenger Car .....	<b>3-51</b>
<b>Resize the CI Engine</b> .....	<b>3-54</b>
Create CI Engine Models with Twice the Power .....	<b>3-54</b>
<b>Resize the SI Engine</b> .....	<b>3-63</b>
Create SI Engine Models with Twice the Power .....	<b>3-63</b>
<b>Generate Mapped CI Engine from a Spreadsheet</b> .....	<b>3-72</b>
Step 1: Generate Mapped Engine Calibration .....	<b>3-72</b>
Step 2: Apply Calibration to Mapped Engine Model .....	<b>3-76</b>

<b>Generate Mapped SI Engine from a Spreadsheet</b> .....	<b>3-78</b>
Step 1: Generate Mapped Engine Calibration .....	<b>3-78</b>
Step 2: Apply Calibration to Mapped Engine Model .....	<b>3-81</b>
<b>Internal Combustion Mapped and Dynamic Engine Models</b> .	<b>3-83</b>

## Project Templates

### 4

<b>CI Engine Project Template</b> .....	<b>4-2</b>
Controller .....	<b>4-2</b>
Plant .....	<b>4-2</b>
<b>SI Engine Project Template</b> .....	<b>4-5</b>
Controller .....	<b>4-5</b>
Plant .....	<b>4-5</b>

## Supporting Data

### 5

<b>Install Drive Cycle Data</b> .....	<b>5-2</b>
---------------------------------------	------------

## Calibration

### 6

<b>Generate Parameter Data for Datasheet Battery Block</b> .....	<b>6-2</b>
<b>Generate Parameter Data for Equivalent Circuit Battery Block</b> .....	<b>6-16</b>
Step 1: Load and Preprocess Data .....	<b>6-17</b>
Step 2: Determine the Number of RC Pairs .....	<b>6-20</b>
Step 3: Estimate Parameters .....	<b>6-21</b>
Step 4: Set Equivalent Circuit Battery Block Parameters .....	<b>6-27</b>

<b>Generate Parameters for Flux-Based Blocks</b> .....	<b>6-30</b>
<b>Generate Current Controller Parameters</b> .....	<b>6-33</b>
Collect and Post Process Motor Data .....	<b>6-34</b>
Model Motor Data .....	<b>6-35</b>
Generate Calibration .....	<b>6-40</b>
Set Block Parameters .....	<b>6-58</b>
<b>Generate Feed-Forward Flux Parameters</b> .....	<b>6-60</b>
Step 1: Load and Preprocess Data .....	<b>6-60</b>
Step 2: Generate Evenly Spaced Data .....	<b>6-61</b>
Step 3: Set Block Parameters .....	<b>6-63</b>
<b>Generate Parameters for Flux-Based PMSM Block</b> .....	<b>6-65</b>
Step 1: Load and Preprocess Data .....	<b>6-65</b>
Step 2: Generate Evenly Spaced Table Data From Scattered Data .....	<b>6-67</b>
Step 3: Set Block Parameters .....	<b>6-69</b>

## Powertrain Blockset Examples

# 7

<b>SI Engine Dynamometer Reference Application</b> .....	<b>7-2</b>
<b>CI Engine Dynamometer Reference Application</b> .....	<b>7-4</b>
<b>Electric Vehicle Reference Application</b> .....	<b>7-6</b>
<b>Hybrid Electric Vehicle Input Power-Split Reference Application</b> .....	<b>7-7</b>
<b>Hybrid Electric Vehicle P2 Reference Application</b> .....	<b>7-8</b>
<b>Hybrid Electric Vehicle Multimode Reference Application</b> ...	<b>7-9</b>
<b>Conventional Vehicle Reference Application</b> .....	<b>7-10</b>





# Getting Started

---

## Powertrain Blockset Product Description

### Model and simulate automotive powertrain systems

Powertrain Blockset provides fully assembled reference application models of automotive powertrains, including gasoline, diesel, hybrid, and electric systems. It includes a component library for simulating engine subsystems, transmission assemblies, traction motors, battery packs, and controller models. Powertrain Blockset also includes a dynamometer model for virtual testing. MDF file support provides a standards-based interface to calibration tools for data import.

Powertrain Blockset provides a standard model architecture that can be reused throughout the development process. You can use it for design tradeoff analysis and component sizing, control parameter optimization, and hardware-in-the-loop testing. You can customize models by parameterizing components in a reference application with your own data or by replacing a subsystem with your own model.

### Key Features

- Fully assembled models for gasoline, diesel, hybrid, and electric powertrains
- Libraries of engine, transmission, traction motor, and battery components
- Basic controllers for powertrain subsystems
- Standard drive cycle data, including FTP75, NEDC, and JC08
- Engine dynamometer model for virtual calibration and testing
- MDF file support for calibration data import

## Required and Recommended Products

### Required Products

Powertrain Blockset product requires current versions of these products:

- MATLAB
- Simulink

### Recommended Products

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

Goal	Recommended Product
Model events	Stateflow®
Use physical modeling blocks	Simscape and Simscape™ add-ons
Optimize powertrain performance and control parameters	Optimization Toolbox™
Generate reports	MATLAB® Report Generator™ Simulink® Report Generator
Optimize powertrain design	Simulink Design Optimization™
Parallel computing	MATLAB Distributed Computing Server™ Parallel Computing Toolbox™
Calibrate engine models	Model-Based Calibration Toolbox™

## Getting Started with Powertrain Blockset

The Powertrain Blockset provides reference application projects assembled from blocks and subsystems. Use the reference applications as a starting point to create your own powertrain models.

Objective	For	See
Design tradeoff analysis and component sizing, control parameter optimization, or hardware-in-the-loop (HIL) testing.	Full conventional vehicle with spark-ignition (SI) or combustion-ignition (CI)	“Explore the Conventional Vehicle Reference Application” on page 3-4
	Hybrid electric vehicle (HEV) — Multimode	“Explore the Hybrid Electric Vehicle Multimode Reference Application” on page 3-20
	HEV — Input power-split	“Explore the Hybrid Electric Vehicle Input Power-Split Reference Application” on page 3-34
	Full electric vehicle	“Explore the Electric Vehicle Reference Application” on page 3-28
Engine and controller calibration, validation, and optimization before integration with the vehicle model.	CI engine plant and controller	“Explore the CI Engine Dynamometer Reference Application” on page 3-10
	SI engine plant and controller	“Explore the SI Engine Dynamometer Reference Application” on page 3-15

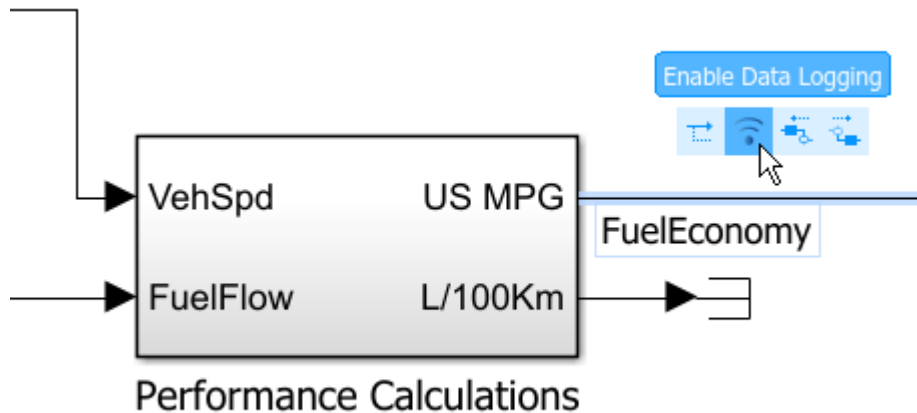
This example shows how to run the conventional vehicle reference application and examine the final drive gear ratio impact on fuel economy and tailpipe emissions.

Running this example requires a Stateflow license. You can install a Stateflow trial license using the Add-On Explorer.

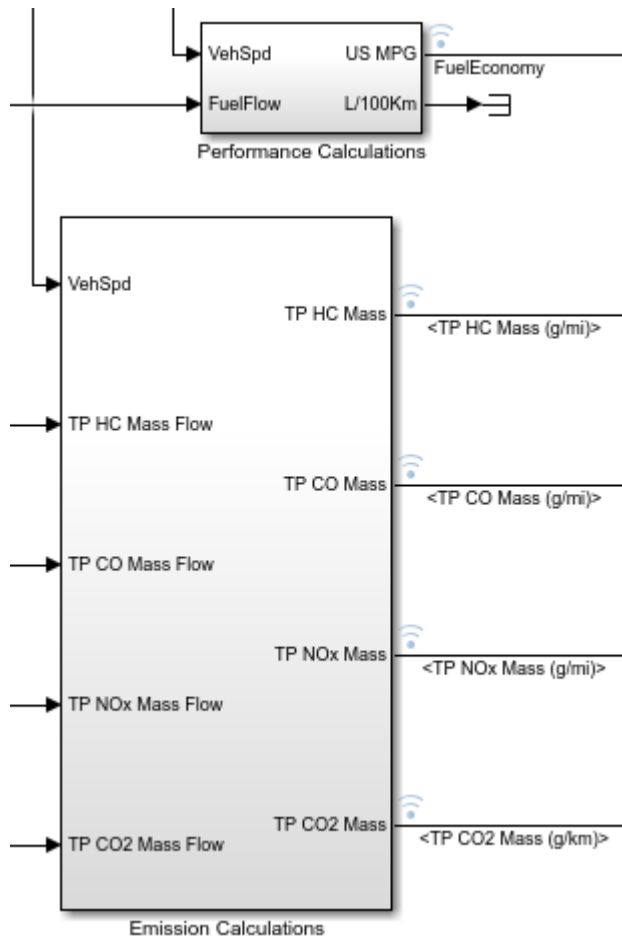
- 1 Open the conventional vehicle reference application project. By default, the application has a 1.5-L spark-ignition (SI) engine and a final drive gear ratio of 3.  
`autoblkConVehStart`

Project files open in a writable location.

- 2 Enable data logging for the fuel economy and tailpipe emissions signals.
  - a In the Visualization subsystem, select the FuelEconomy signal line and Enable Data Logging.

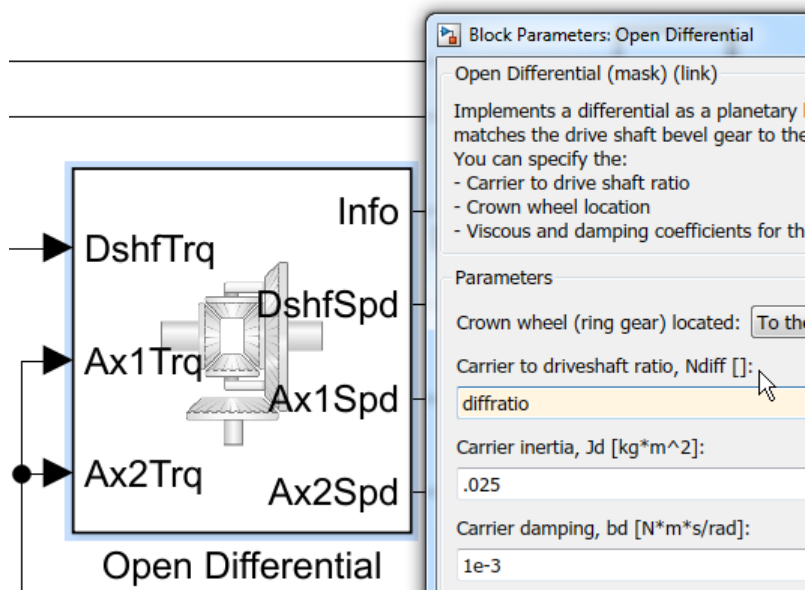


- b In the Visualization subsystem, enable data logging on the tailpipe emissions signals.



- c** Save the SiCiPtReferenceApplication model.
- 3** Parameterize the final drive gear ratio.
  - a** In the Passenger Car subsystem, navigate to the SiDrivetrain > Differential and Compliance > Front Wheel Drive subsystem. Open the Open Differential block.
  - b** In the Open Differential block mask:

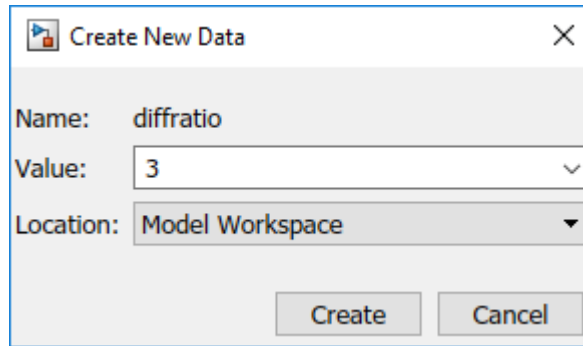
- Change the **Carrier to driveshaft ratio, Ndiff** parameter to the variable `diffratio`. The **Carrier to driveshaft ratio, Ndiff** parameter represents the final drive gear ratio.



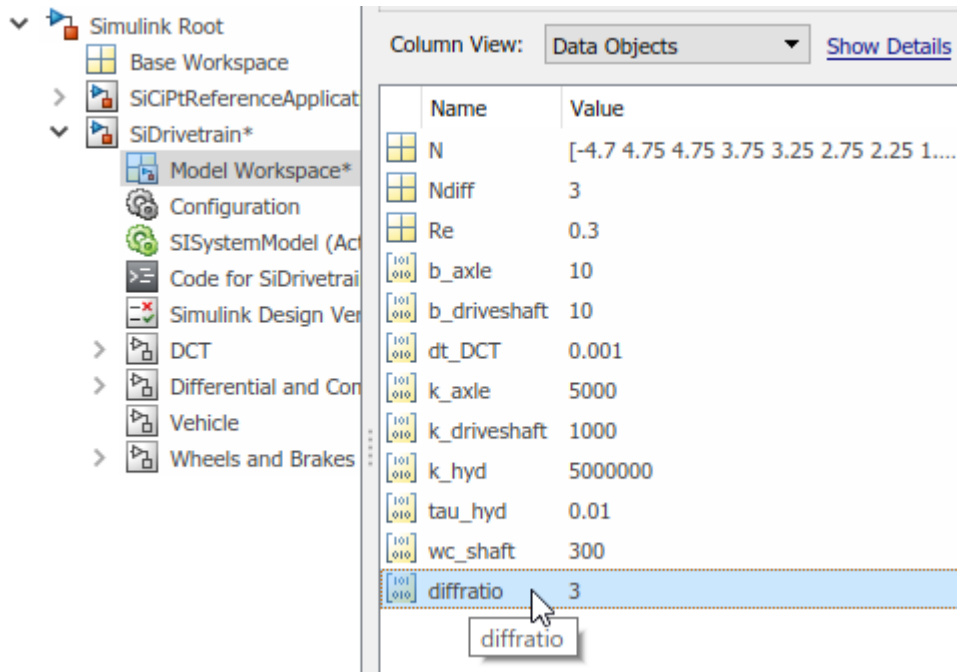
- Use the available actions to create new data.



- Use the Create New Data dialog box to create a Model Workspace parameter `diffratio` equal to a value of 3.




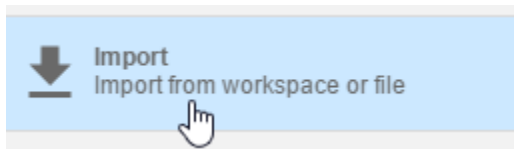
- In the Open Differential block mask, apply the change.
- c In the Model Explorer, for the SiDrivetrain model, confirm that the diffratio parameter is set to 3.



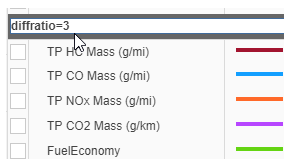
- d Save the SiDrivetrain and SiCiPtReferenceApplication models.



- 4 Run a baseline conventional vehicle simulation with a final drive gear ratio of 3. Import the results to the Simulation Data Inspector.
  - a In the `SiCiPtReferenceApplication` model, run the simulation for the default run time. The simulation can take time to run. View progress in the Simulink window.
  - b On the Simulink Editor toolbar, click the **Simulation Data Inspector** button  to open the Simulation Data Inspector.
    - i In the Simulation Data Inspector, select **Import**. In the Import dialog box, accept the defaults and select **Import**.

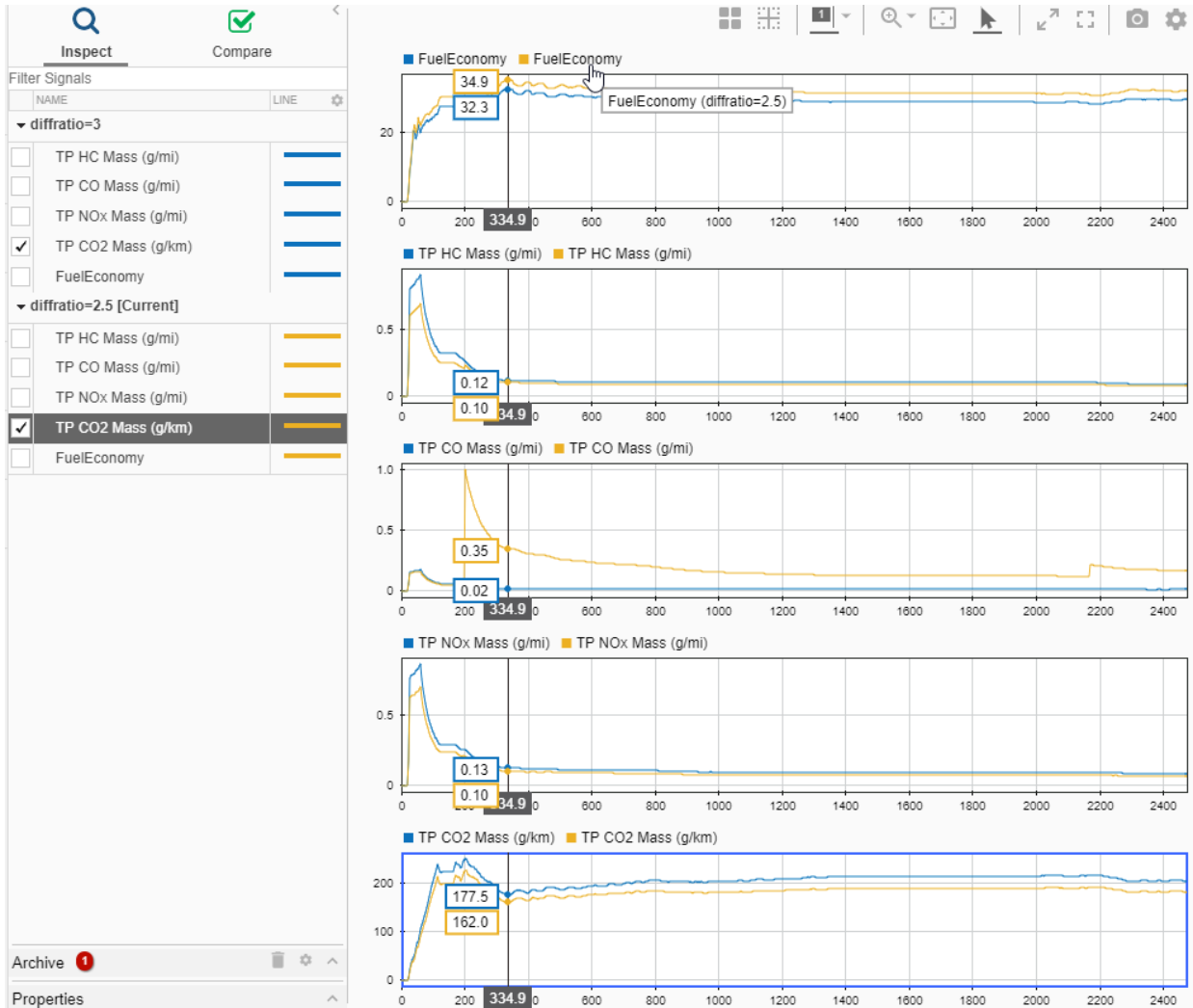


- ii In the results field for the run, right-click to rename the run `diffratio=3`.



- 5 Run a conventional vehicle simulation with a final drive gear ratio of 2.5. Import the results to the Simulation Data Inspector.
  - a In the Model Explorer, for the `SiDrivetrain` model, set the Model Workspace `diffratio` parameter to 2.5.
  - b Save the `SiDrivetrain` model.
  - c In the `SiCiPtReferenceApplication` model, run the simulation for the default run time.
  - d To import the results, on the toolbar, select the Simulation Data Inspector.
    - i In the Simulation Data Inspector, select **Import**. In the Import dialog box, accept the defaults and select **Import**.
    - ii In the Simulation Data Inspector, in the results field for the run, right-click to rename the run `diffratio=2.5`.

- 6 Use the Simulation Data Inspector to explore the results. To assess the impact of the final drive gear ratio on the fuel economy and tailpipe emissions, view the plots of the simulation results. For example, these simulation results indicate a better powertrain match when the final drive gear ratio is 2.5:
  - Fuel economy increases when the final drive gear ratio changes from 3 to 2.5.
  - Tailpipe emissions (HC, NO<sub>x</sub>, CO<sub>2</sub>) decrease when the final drive gear ratio changes from 3 to 2.5.



## Next Steps

Assess the impact of the final drive gear ratio on vehicle performance. Although the fuel economy and tailpipe emissions indicate a better powertrain match when the final drive gear ratio is 2.5, the ratio also impacts performance.

To assess the vehicle performance, examine 0 to 100 km/hr acceleration times for each axle setting. You can use the Drive Cycle Source block to output a constant velocity of (100/3.6) m/s.

## See Also

### Related Examples

- “Conventional Vehicle Fuel Economy and Emissions” on page 1-13

### More About

- “Explore the Conventional Vehicle Reference Application” on page 3-4
- Simulation Data Inspector

## Conventional Vehicle Fuel Economy and Emissions

This example shows how to obtain the city and highway fuel economy and tailpipe emissions for a conventional vehicle with a 1.5-L spark-ignition (SI) engine. To run this example, install the city (FTP75) and highway (HWFET) drive cycles. To install, see “Install Drive Cycle Data” on page 5-2.

To open a script that executes these commands, in the MATLAB command window, type:

ConVehMPGExample

- 1 Open the conventional vehicle reference application project.

```
autoblkConVehStart;
```

- 2 Prepare the reference application for simulation.

- Name the subsystems.

```
model = 'SiCiPtReferenceApplication';
dcs = [model, '/Drive Cycle Source'];
vis_sys = [model, '/Visualization'];
```

- Add signal loggers to the models using the function `pt_set_logging`.

```
pt_set_logging([vis_sys, '/Performance Calculations'], 'US MPG', ...
'Fuel Economy [mpg]', 'both')
pt_set_logging([vis_sys, '/Emission Calculations'], 'TP HC Mass (g/mi)', ...
'HC [g/mi]', 'both')
pt_set_logging([vis_sys, '/Emission Calculations'], 'TP CO Mass (g/mi)', ...
'CO [g/mi]', 'both')
pt_set_logging([vis_sys, '/Emission Calculations'], ...
'TP NOx Mass (g/mi)', 'NOx [g/mi]', 'both')
pt_set_logging([vis_sys, '/Emission Calculations'], ...
'TP CO2 Mass (g/km)', 'CO2 [g/km]', 'both')
```

- 3 Run a conventional vehicle simulation using the city drive cycle FTP75.

```
set_param(dcs, 'cycleVar', 'FTP75');
tfinal = get_param(dcs, 'tfinal');
tf = tfinal(1:strfind(tfinal, ' '));
simout1 = sim(model, 'ReturnWorkspaceOutputs', 'on', 'StopTime', tf);
```

- 4 Run a conventional vehicle simulation using the highway drive cycle HWFET.

```
set_param(dcs, 'cycleVar', 'HWFET');
tfinal = get_param(dcs, 'tfinal');
tf = tfinal(1:strfind(tfinal, ' '));
simout2 = sim(model, 'ReturnWorkspaceOutputs', 'on', 'StopTime', tf);
```

- 5 Extract the city and highway fuel economy. Calculate a combined fuel economy.

```
logouts1 = simout1.get('logouts');  
FE_urban = logouts1.get('Fuel Economy [mpg]').Values.Data(end);  
logouts2 = simout2.get('logouts');  
FE_hwy = logouts2.get('Fuel Economy [mpg]').Values.Data(end);  
  
FE_combined = 0.55*FE_urban + 0.45*FE_hwy;
```

- 6 Extract the tailpipe emissions from the city drive cycle.

```
HC = logouts1.get('HC [g/mi]').Values.Data(end);  
CO = logouts1.get('CO [g/mi]').Values.Data(end);  
NOx = logouts1.get('NOx [g/mi]').Values.Data(end);  
CO2 = logouts1.get('CO2 [g/km]').Values.Data(end);
```

- 7 Display the fuel economy and tailpipe emissions results in the command window.

```
fprintf('\n*****\n')  
fprintf('FUEL ECONOMY\n');  
fprintf('  City:      %4.2f mpg\n', FE_urban);  
fprintf('  Highway:   %4.2f mpg\n', FE_hwy);  
fprintf('  Combined:  %4.2f mpg\n', FE_combined);  
fprintf('\nTAILPIPE EMISSIONS\n');  
fprintf('  HC:      %4.3f [g/mi]\n',HC);  
fprintf('  CO:      %4.3f [g/mi]\n',CO);  
fprintf('  NOx:     %4.3f [g/mi]\n',NOx);  
fprintf('  CO2:     %4.1f [g/km]\n',CO2);  
fprintf('  NMOG:    %4.3f [g/mi]',HC+NOx);  
fprintf('\n*****\n')
```

- 8 Remove the temporary variables.

```
clear ans dcs model tf tfinal vis_sys
```

---

**Note** You cannot pause simulations started using the `sim` command.

---

## See Also

### Related Examples

- “Install Drive Cycle Data” on page 5-2
- “Getting Started with Powertrain Blockset” on page 1-4

## **More About**

- “Explore the Conventional Vehicle Reference Application” on page 3-4





# Workflows

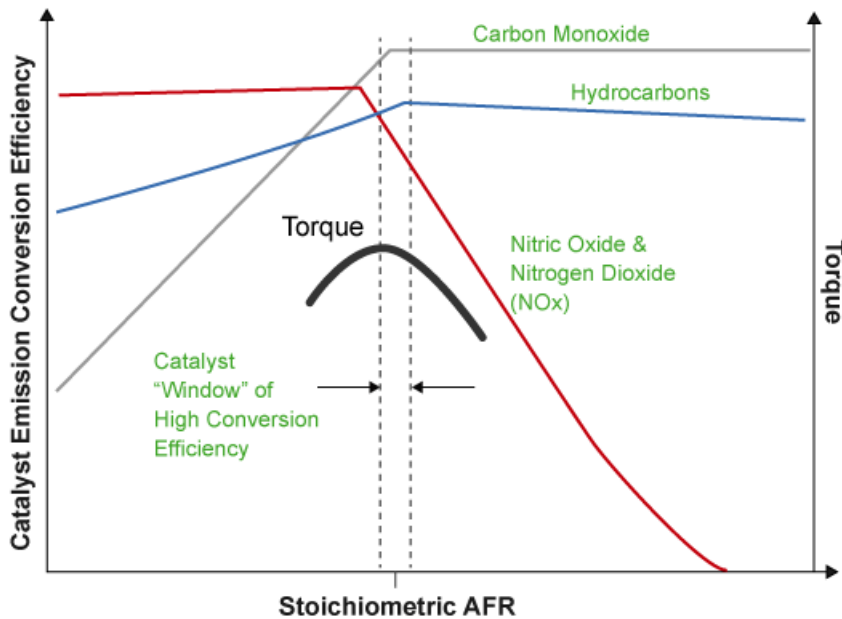
---

- “SI Core Engine Air Mass Flow and Torque Production” on page 2-2
- “SI Engine Dual-Independent Cam Phaser Air Mass Flow Model” on page 2-5
- “SI Engine Speed-Density Air Mass Flow Model” on page 2-14
- “SI Engine Torque Structure Model” on page 2-17
- “SI Engine Simple Torque Model” on page 2-26
- “CI Core Engine Air Mass Flow and Torque Production” on page 2-27
- “CI Engine Speed-Density Air Mass Flow Model” on page 2-29
- “CI Engine Torque Structure Model” on page 2-33
- “CI Engine Simple Torque Model” on page 2-40
- “Engine Calibration Maps” on page 2-41

## SI Core Engine Air Mass Flow and Torque Production

A spark-ignition (SI) engine produces torque by controlling the net airflow into the engine using throttle, turbocharger wastegate, and cam-phasing actuators.

While producing torque, the engine must comply with emission standards. To meet the tailpipe emission standards, the ECU operates a three-way-catalyst (TWC) at the stoichiometric air-fuel ratio (AFR).



In addition to emission controls, the ECU:

- Maximizes torque at middle speeds and high loads by operating rich of stoichiometry.
- Limits piston crown temperature at high speeds and high loads by running rich of stoichiometry.

## Air Mass Flow Models

To calculate engine air mass flow, configure the SI engine to use either of these air mass flow models.

Air Mass Flow Model	Description
"SI Engine Speed-Density Air Mass Flow Model" on page 2-14	Uses the speed-density equation to calculate the engine air mass flow, relating the engine air mass flow to the intake manifold pressure and engine speed. Consider using this air mass flow model in engines with fixed valvetrain designs.
"SI Engine Dual-Independent Cam Phaser Air Mass Flow Model" on page 2-5	<p>To calculate the engine air mass flow, the dual-independent cam phaser model uses:</p> <ul style="list-style-type: none"> <li>• Empirical calibration parameters developed from engine mapping measurements</li> <li>• Desktop calibration parameters derived from engine computer-aided design (CAD) data</li> </ul> <p>In contrast to typical embedded air mass flow calculations based on direct air mass flow measurement with an air mass flow (MAF) sensor, this air mass flow model offers:</p> <ul style="list-style-type: none"> <li>• Elimination of MAF sensors in dual cam-phased valvetrain applications</li> <li>• Reasonable accuracy with changes in altitude</li> <li>• Semiphysical modeling approach</li> <li>• Bounded behavior</li> <li>• Suitable execution time for electronic control unit (ECU) implementation</li> <li>• Systematic development of a relatively small number of calibration parameters</li> </ul>

## Torque Models

To calculate the brake torque, configure the SI engine to use either of these torque models.

<b>Brake Torque Model</b>	<b>Description</b>
"SI Engine Torque Structure Model" on page 2-17	For the structured brake torque calculation, the SI engine uses tables for the inner torque, friction torque, optimal spark, spark efficiency, and lambda efficiency.
"SI Engine Simple Torque Model" on page 2-26	For the simple brake torque calculation, the SI engine block uses a torque lookup table map that is a function of engine speed and load.

### See Also

SI Controller | SI Core Engine

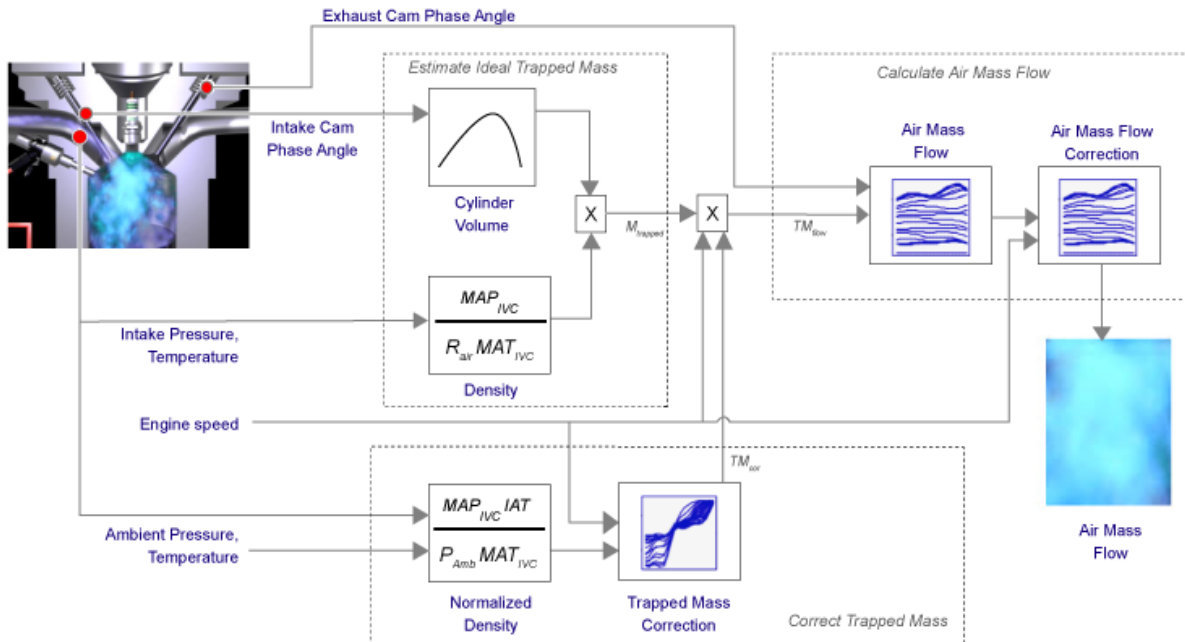
### More About

- "Engine Calibration Maps" on page 2-41

## SI Engine Dual-Independent Cam Phaser Air Mass Flow Model

To calculate intake air mass flow for an engine equipped with cam phasers, you can configure the spark-ignition (SI) engine with a dual-independent cam phaser intake air mass flow model. As illustrated, the spark-ignition (SI) engine intake air mass flow calculation consists of these steps:

- Collecting physical measurements
- Estimating the ideal trapped mass
- Correcting the trapped mass
- Calculating the intake air mass flow



The dual-independent cam phaser intake air mass flow model implements equations that use these variables.

$M_{trapped}$	Estimated ideal trapped mass
$TM_{corr}$	Trapped mass correction multiplier
$TM_{flow}$	Flow rate equivalent to corrected trapped mass at the current engine speed
$\dot{m}_{intkideal}$	Engine intake air mass flow at arbitrary cam phaser angles
$\dot{m}_{intkideal}$	Engine intake port mass flow at arbitrary cam phaser angles
$\dot{m}_{air}$	Engine intake air mass flow final correction at steady-state cam phaser angles
$\dot{m}_{intk}$	Engine intake port mass flow at steady-state cam phaser angles
$y_{intk,air}$	Engine intake manifold air mass fraction
$MAP_{IVC}$	Intake manifold pressure at IVC
$MAT_{IVC}$	Intake manifold temperature at IVC
$M_{Nom}$	Nominal engine cylinder intake air mass at standard temperature and pressure, piston at bottom dead center (BDC) maximum volume
$IAT$	Intake air temperature
$N$	Engine speed
$N_{cyl}$	Number of engine cylinders
$V_{IVC}$	Cylinder volume at IVC
$V_d$	Displaced volume
$R_{air}$	Ideal gas constant
$P_{Amb}$	Ambient pressure
$T_{std}$	Standard temperature
$P_{std}$	Standard pressure

$\rho_{norm}$	Normalized density
$\varphi_{ICP}$	Measured intake cam phaser angle
$\varphi_{ECP}$	Exhaust cam phaser angle
$L_{ideal}$	Engine load (normalized cylinder air mass) at arbitrary cam phaser angles, uncorrected for final steady-state cam phaser angles
$L$	Engine load (normalized cylinder air mass) at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles
$Cps$	Crankshaft revolutions per power stroke
$f_{Vivc}$	Cylinder volume at IVC table
$f_{TMcorr}$	Trapped mass correction table
$f_{airideal}$	Intake air mass flow table
$f_{aircorr}$	Intake air mass flow correction table

## Collect Physical Measurements

In the SI engine model, the dual-independent cam phaser intake air mass flow model requires these physical measurements:

- Intake manifold temperature and pressure at intake valve closing (IVC) condition
- Intake cam phase angle
- Exhaust cam phase angle
- Engine speed
- Ambient pressure and temperature
- Intake air mass flow, from one or more of the following
  - Tank air meter
  - Wide range air-fuel sensor and fuel-flow meter
  - Wide range air-fuel sensor and injector pulse-width

## Estimate Ideal Trapped Mass

The dual-independent cam phaser intake air mass flow model uses the Ideal Gas Law to estimate the ideal trapped mass at intake manifold conditions. The calculation assumes the cylinder pressure and temperature at IVC equal the intake manifold pressure and temperature.

$$M_{trapped} \cong \frac{MAP_{IVC} V_{IVC}}{R_{air} MAT_{IVC}}$$

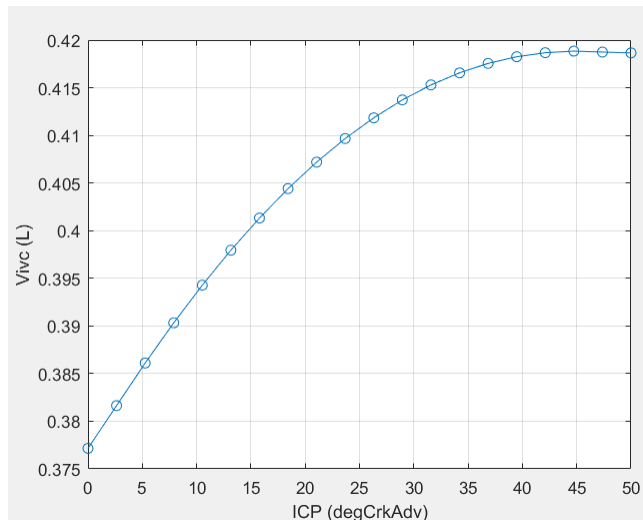
For engines with variable intake cam phasing, the trapped volume at IVC varies.

The cylinder volume at intake valve close table (IVC),  $f_{V_{ivc}}$  is a function of the intake cam phaser angle

$$V_{IVC} = f_{V_{ivc}}(\varphi_{ICP})$$

where:

- $V_{IVC}$  is cylinder volume at IVC, in L.
- $\varphi_{ICP}$  is intake cam phaser angle, in crank advance degrees.





## Correct Trapped Mass

The dual-independent cam phaser intake air mass flow model uses a correction factor to account for the difference between the ideal trapped mass in the cylinder and the actual trapped mass. The trapped mass correction factor is a lookup table that is a function of the normalized density and engine speed.

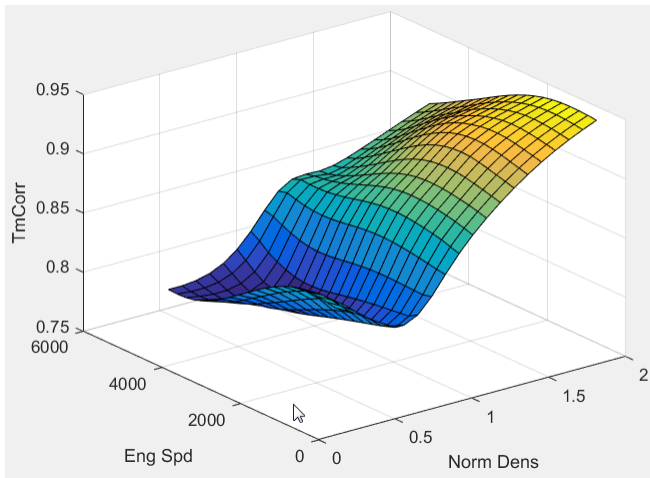
$$\rho_{norm} = \frac{MAP_{IVC} IAT}{P_{Amb} MAT_{IVC}}$$

The trapped mass correction factor table,  $f_{TM_{corr}}$ , is a function of the normalized density and engine speed

$$TM_{corr} = f_{TM_{corr}}(\rho_{norm}, N)$$

where:

- $TM_{corr}$ , is trapped mass correction multiplier, dimensionless.
- $\rho_{norm}$  is normalized density, dimensionless.
- $N$  is engine speed, in rpm.



- Normalized density accounts for the throttle position independent of a given altitude.
- Engine speed accounts for the pulsation effects of the piston movement.
- Ambient pressure is measured by a sensor on the electronic control unit (ECU) or estimated using an inverse throttle valve model.
- The ECU estimates or measures intake air temperature (IAT) upstream of the throttle.

Trapped mass flow is expressed as a flow rate in grams per second (g/s). The trapped mass flow is the maximum gas mass flow through the engine when no residual gases remain in the cylinder at the end of the exhaust stroke.

$$TM_{flow} = \frac{\left(1000 \frac{g}{kg}\right) N_{cyl} TM_{corr} M_{trapped} N}{\left(\frac{60s}{min}\right) Cps}$$

## Calculate Air Mass Flow

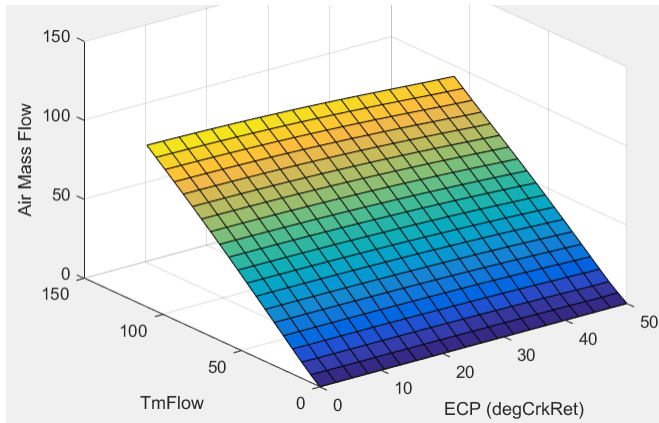
To determine the engine intake air mass flow at arbitrary cam phase angles, the dual-independent cam phaser air mass flow model uses a lookup table.

The phaser intake mass flow model lookup table is a function of exhaust cam phaser angles and trapped air mass flow

$$\dot{m}_{intkideal} = f_{intkideal}(\varphi_{ECP}, TM_{flow})$$

where:

- $\dot{m}_{intkideal}$  is engine intake port mass flow at arbitrary cam phaser angles, in g/s.
- $\varphi_{ECP}$  is exhaust cam phaser angle, in degrees crank retard.
- $TM_{flow}$  is flow rate equivalent to corrected trapped mass at the current engine speed, in g/s.



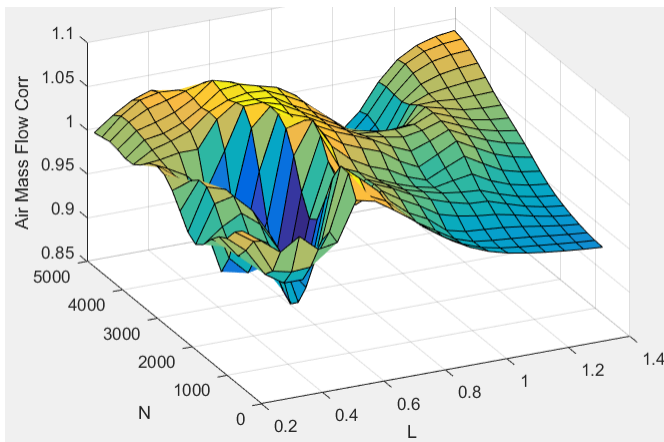
- The exhaust cam phasing has a significant effect on the fraction of burned gas. During the exhaust stroke, exhaust cam-phasing affects the exhaust valve position at exhaust valve closing (EVC) relative to the piston position. A retarded (late) exhaust cam phase angle moves EVC past piston top dead center (TDC), causing the exhaust gas to flow back from the manifold runner into the cylinder. This pull-back triggers the reburn of crevice volume gasses, reducing nitric oxide and nitrogen dioxide emissions (NO<sub>x</sub>) via charge temperature reduction and hydrocarbon (HC) emissions. Exhaust temperature and back pressure affect exhaust gas back-flow and exhaust cam phaser timing. Exhaust gas temperature and pressure correlate to trapped mass flow. Since at least 80% of trapped mass flow is unburned air, air mass flow is highly correlated to trapped mass flow.
- The unburned air mass flow determines the engine load and open-loop fuel control to achieve a target air-fuel ratio (AFR).
- The lookup table allows arbitrary cam phaser position combinations that can occur during transient engine operations when the phasers are moving from one target position to another.

The intake air mass flow correction lookup table,  $f_{aircorr}$ , is a function of ideal load and engine speed

$$\dot{m}_{air} = \dot{m}_{intkideal} f_{aircorr}(L_{ideal}, N)$$

where:

- $L_{ideal}$  is engine load (normalized cylinder air mass) at arbitrary cam phaser angles, uncorrected for final steady-state cam phaser angles, dimensionless.
- $N$  is engine speed, in rpm.
- $\dot{m}_{air}$  is engine intake air mass flow final correction at steady-state cam phaser angles, in g/s.
- $\dot{m}_{intkideal}$  is engine intake port mass flow at arbitrary cam phaser angles, in g/s.



- To calculate the engine intake port mass flow, the engine model uses this equation.

$$\dot{m}_{intk} = \frac{\dot{m}_{air}}{y_{intk,air}}$$

- Ideal load is the normalized engine cylinder unburned intake air mass before the final correction. To calculate ideal load, the model divides the unburned intake air mass by the nominal cylinder intake air mass. The nominal cylinder intake air mass is the intake air mass (kg) in a cylinder at piston bottom dead center (BDC) with air at standard temperature and pressure:

$$M_{Nom} = \frac{P_{std} V_d}{N_{cyl} R_{air} T_{std}}$$

$$L_{ideal} = \frac{\left(\frac{60s}{min}\right) C_p s \dot{m}_{intkideal} y_{intk,air}}{\left(\frac{1000g}{kg}\right) N_{cyl} N M_{Nom}}$$

- The final engine load is expressed by

$$L = \frac{\left(\frac{60s}{min}\right) C_p s \dot{m}_{air}}{\left(\frac{1000g}{Kg}\right) N_{cyl} N M_{Nom}}$$

## See Also

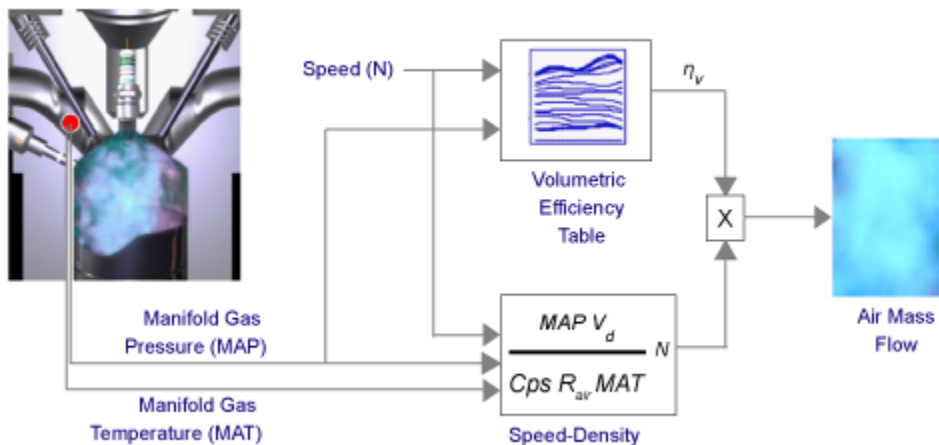
SI Controller | SI Core Engine

## More About

- “SI Core Engine Air Mass Flow and Torque Production” on page 2-2
- “SI Engine Speed-Density Air Mass Flow Model” on page 2-14
- “Engine Calibration Maps” on page 2-41

## SI Engine Speed-Density Air Mass Flow Model

To calculate the air mass flow in the spark-ignition (SI) engine, you can configure the Spark Ignition Core Engine block to use a speed-density air mass flow model. The speed-density model uses the speed-density equation to calculate the engine air mass flow. The equation relates the engine air mass flow to the intake manifold gas pressure, intake manifold gas temperature, and engine speed. Consider using this air mass flow model in simple conventional engine designs, where variable valvetrain technologies are not in use.



To determine the air mass flow, the speed-density air mass flow model applies these speed-density equations at the intake manifold gas pressure and gas temperature states.

$$\dot{m}_{intk} = \frac{MAP V_d N \left[ \frac{1 \text{ min}}{60 \text{ s}} \right]}{C_{ps} R_{air} MAT} \eta_v$$

$$\dot{m}_{air} = y_{intk,air} \dot{m}_{intk}$$

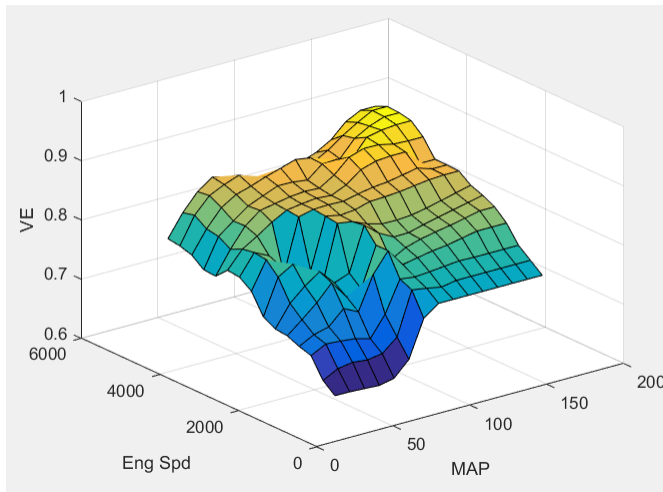
The speed-density air mass flow model uses a volumetric efficiency lookup table to correct the ideal air mass flow.

The engine volumetric efficiency lookup table,  $f_{\eta_v}$ , is a function of intake manifold absolute pressure and engine speed

$$\eta_v = f_{\eta_v}(MAP, N)$$

where:

- $\eta_v$  is engine volumetric efficiency, dimensionless.
- $MAP$  is intake manifold absolute pressure, in KPa.
- $N$  is engine speed, in rpm.



To develop the volumetric efficiency table, use the measured air mass flow rate, intake manifold gas pressure, intake manifold gas temperature, and engine speed from engine performance testing.

$$\eta_v = \frac{C_p R_{air} MAT}{MAP V_d N \left[ \frac{1 \text{ min}}{60 \text{ s}} \right]} \dot{m}_{air}$$

The air mass flow model implements equations that use these variables.

$MAP$	Cycle average intake manifold pressure
$\dot{m}_{intk}$	Engine intake port mass flow
$\dot{m}_{air}$	Engine intake air mass flow
$V_d$	Displaced volume
$N$	Engine speed
$C_{ps}$	Crankshaft revolutions per power stroke
$MAT$	Cycle average intake manifold gas absolute temperature
$R_{air}$	Ideal gas constant for air and burned gas mixture
$f_{\eta_v}$	Engine volumetric efficiency lookup table
$\eta_v$	Engine volumetric efficiency

## References

[1] Heywood, John B. *Internal Combustion Engine Fundamentals*. New York: McGraw-Hill, 1988.

## See Also

SI Controller | SI Core Engine

## More About

- “SI Core Engine Air Mass Flow and Torque Production” on page 2-2
- “SI Engine Dual-Independent Cam Phaser Air Mass Flow Model” on page 2-5
- “Engine Calibration Maps” on page 2-41

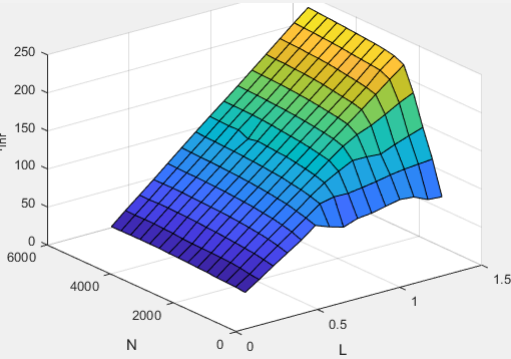


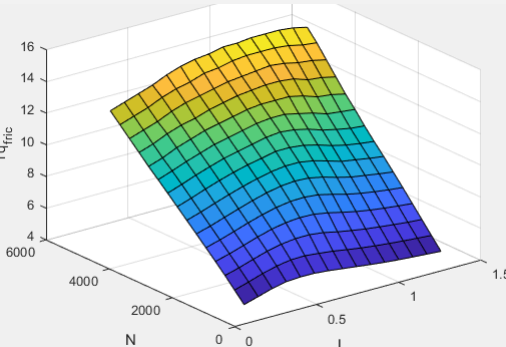
## SI Engine Torque Structure Model

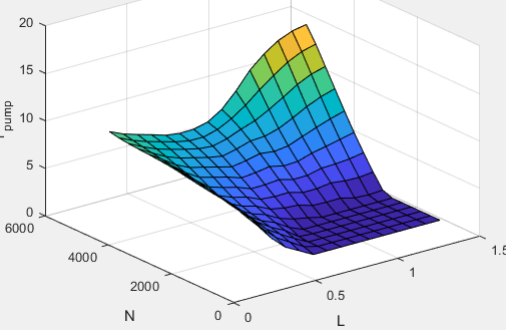
The spark-ignition (SI) engine implements a simplified version of the SI engine torque structure calculation used in a Bosch Engine Management System (EMS). For the torque structure estimation calculation, the block requires calibration tables for:

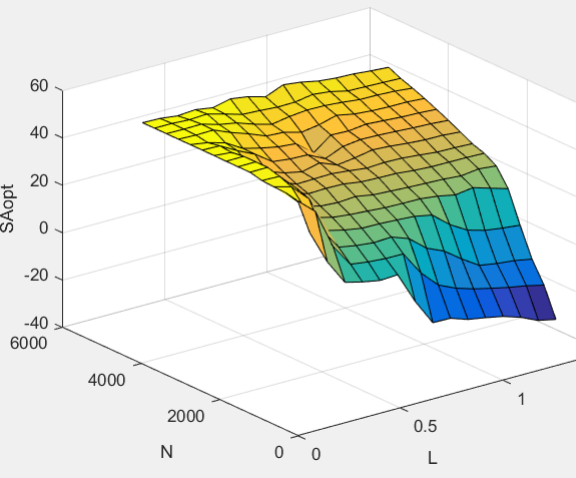
- Inner torque — Maximum torque potential of the engine at a given speed and load
- Friction torque — Torque losses due to friction
- Optimal spark — Spark advance for optimal inner torque
- Spark efficiency — Torque loss due to spark retard from optimal
- Lambda efficiency — Torque loss due to lambda change from optimal
- Pumping torque — Torque loss due to pumping

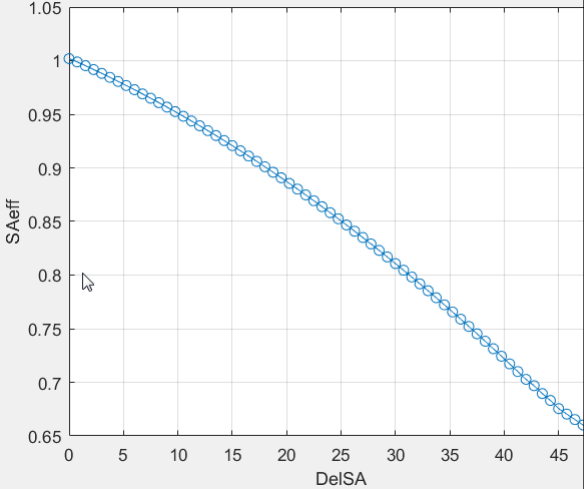
The tables available with Powertrain Blockset were developed with the Model-Based Calibration Toolbox.

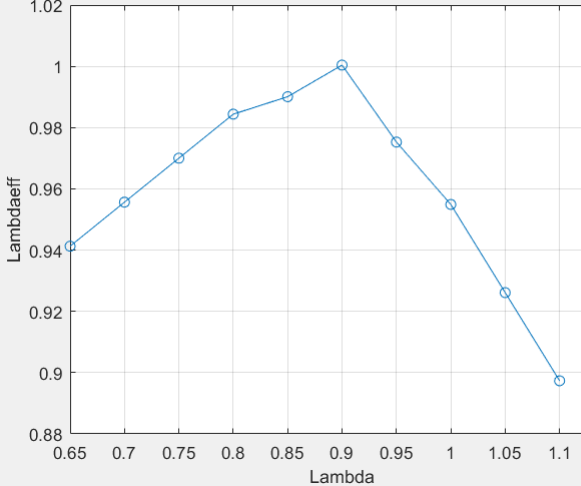
Lookup Table	Used to Determine	Plot
<p>Inner torque, <math>f_{Tq_{inr}}</math></p>	<p><math>Tq_{inr} = f_{Tq_{inr}}(L, N)</math></p>	<p>The inner torque lookup table, <math>f_{Tq_{inr}}</math>, is a function of engine speed and engine load, <math>Tq_{inr} = f_{Tq_{inr}}(L, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>Tq_{inr}</math> is inner torque based on gross indicated mean effective pressure, in N·m.</li> <li>• <math>L</math> is engine load at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles, dimensionless.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

Lookup Table	Used to Determine	Plot
Friction torque, $f_{T_{fric}}$	$T_{fric} = f_{T_{fric}}(L, N)$	<p>The friction torque lookup table, <math>f_{T_{fric}}</math>, is a function of engine speed and engine load,</p> <p><math>T_{fric} = f_{T_{fric}}(L, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>T_{fric}</math> is friction torque offset to inner torque, in N·m.</li> <li>• <math>L</math> is engine load at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles, dimensionless.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

Lookup Table	Used to Determine	Plot
Pumping torque, $f_{T_{pump}}$	$T_{pump} = f_{T_{pump}}(L, N)$	<p>The pumping torque lookup table, <math>f_{T_{pump}}</math>, is a function of engine speed and injected fuel mass, <math>T_{pump} = f_{T_{pump}}(L, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>T_{pump}</math> is pumping torque, in N·m.</li> <li>• <math>L</math> is engine load, as a normalized cylinder air mass, dimensionless.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

Lookup Table	Used to Determine	Plot
Optimal spark, $f_{SAopt}$	$SA_{opt} = f_{SAopt}(L, N)$	<p>The optimal spark lookup table, <math>f_{SAopt}</math>, is a function of engine speed and engine load,</p> $SA_{opt} = f_{SAopt}(L, N)$ , where: <ul style="list-style-type: none"> <li>• <math>SA_{opt}</math> is optimal spark advance timing for maximum inner torque at stoichiometric air-fuel ratio (AFR), in deg.</li> <li>• <math>L</math> is engine load at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles, dimensionless.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

Lookup Table	Used to Determine	Plot																								
Spark efficiency, $f_{Msa}$	$M_{sa} = f_{Msa}(\Delta SA)$ $\Delta SA = SA_{opt} - SA$	<p>The spark efficiency lookup table, <math>f_{Msa}</math>, is a function of the spark retard from optimal</p> $M_{sa} = f_{Msa}(\Delta SA)$ $\Delta SA = SA_{opt} - SA$ <p>where:</p> <ul style="list-style-type: none"> <li>• <math>M_{sa}</math> is the spark retard efficiency multiplier, dimensionless.</li> <li>• <math>\Delta SA</math> is the spark retard timing distance from optimal spark advance, in deg.</li> </ul>  <table border="1"> <caption>Approximate data points from the SAeff vs DelSA plot</caption> <thead> <tr> <th>DelSA (deg)</th> <th>SAeff</th> </tr> </thead> <tbody> <tr><td>0</td><td>1.00</td></tr> <tr><td>5</td><td>0.97</td></tr> <tr><td>10</td><td>0.94</td></tr> <tr><td>15</td><td>0.91</td></tr> <tr><td>20</td><td>0.88</td></tr> <tr><td>25</td><td>0.85</td></tr> <tr><td>30</td><td>0.82</td></tr> <tr><td>35</td><td>0.79</td></tr> <tr><td>40</td><td>0.76</td></tr> <tr><td>45</td><td>0.73</td></tr> <tr><td>50</td><td>0.70</td></tr> </tbody> </table>	DelSA (deg)	SAeff	0	1.00	5	0.97	10	0.94	15	0.91	20	0.88	25	0.85	30	0.82	35	0.79	40	0.76	45	0.73	50	0.70
DelSA (deg)	SAeff																									
0	1.00																									
5	0.97																									
10	0.94																									
15	0.91																									
20	0.88																									
25	0.85																									
30	0.82																									
35	0.79																									
40	0.76																									
45	0.73																									
50	0.70																									

Lookup Table	Used to Determine	Plot																						
Lambda efficiency, $f_{M\lambda}$	$M_\lambda = f_{M\lambda}(\lambda)$	<p>The lambda efficiency lookup table, <math>f_{M\lambda}</math>, is a function of lambda, <math>M_\lambda = f_{M\lambda}(\lambda)</math>, where:</p> <ul style="list-style-type: none"> <li><math>M_\lambda</math> is the lambda multiplier on inner torque to account for the air-fuel ratio (AFR) effect, dimensionless.</li> <li><math>\lambda</math> is lambda, AFR normalized to stoichiometric fuel AFR, dimensionless.</li> </ul>  <table border="1" data-bbox="758 690 1371 1177"> <caption>Data points for Lambda efficiency vs Lambda</caption> <thead> <tr> <th>Lambda</th> <th>Lambdaeff</th> </tr> </thead> <tbody> <tr><td>0.65</td><td>0.94</td></tr> <tr><td>0.70</td><td>0.955</td></tr> <tr><td>0.75</td><td>0.97</td></tr> <tr><td>0.80</td><td>0.985</td></tr> <tr><td>0.85</td><td>0.99</td></tr> <tr><td>0.90</td><td>1.00</td></tr> <tr><td>0.95</td><td>0.975</td></tr> <tr><td>1.00</td><td>0.955</td></tr> <tr><td>1.05</td><td>0.925</td></tr> <tr><td>1.10</td><td>0.895</td></tr> </tbody> </table>	Lambda	Lambdaeff	0.65	0.94	0.70	0.955	0.75	0.97	0.80	0.985	0.85	0.99	0.90	1.00	0.95	0.975	1.00	0.955	1.05	0.925	1.10	0.895
Lambda	Lambdaeff																							
0.65	0.94																							
0.70	0.955																							
0.75	0.97																							
0.80	0.985																							
0.85	0.99																							
0.90	1.00																							
0.95	0.975																							
1.00	0.955																							
1.05	0.925																							
1.10	0.895																							

The engine brake torque is based on inner torque with lambda efficiency, spark retard efficiency multipliers, pumping torque, and a friction torque offset

$$T_{brake} = M_\lambda M_{sa} T_{q_{inr}} - T_{fric} - T_{pump}$$

To account for thermal effects, the torque structure model corrects the friction torque calculation as a function of coolant temperature.

$$T_{fric} = M_{fric} f_{Tfric}(L, N)$$

$$M_{fric} = f_{fric,temp}(T_{coolant})$$

The pumping torque is a function of engine speed and engine speed.

$$T_{pump} = f_{T_{pump}}(L, N)$$

$SA_{opt}$	Optimal spark advance timing for maximum inner torque at stoichiometric air-fuel ratio (AFR)
$\Delta SA$	Spark retard timing distance from optimal spark advance
$SA$	Spark advance timing
$L$	Engine load at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles
$N$	Engine speed
$M_{\lambda}$	Lambda multiplier on inner torque to account for the AFR effect
$\lambda$	Lambda, AFR normalized to stoichiometric fuel AFR
$M_{sa}$	Spark retard efficiency multiplier
$f_{Msa}$	Spark efficiency lookup table to account for torque loss due to spark retard from optimal
$f_{Tfric}$	Friction torque lookup table to account for torque losses due to friction
$f_{M\lambda}$	Lambda efficiency lookup table to account for torque loss due to lambda change from optimal
$f_{SAopt}$	Optimal spark lookup table, for maximum inner torque as a function of engine speed and load
$f_{Tqinr}$	Inner torque lookup table, for maximum torque potential of the engine at a given speed and load
$T_{brake}$	Engine brake torque after accounting for spark advance, AFR, and friction effects
$T_{fric}$	Friction torque offset to inner torque



---

$T_{q_{inr}}$	Inner torque based on gross indicated mean effective pressure
$T_{pump}$	Pumping torque
$M_{fric}$	Friction torque modifier
$T_{coolant}$	Coolant temperature

## References

- [1] Gerhardt, J., Hönninger, H., and Bischof, H., *A New Approach to Functional and Software Structure for Engine Management Systems - BOSCH ME7*. SAE Technical Paper 980801, 1998.

## See Also

SI Controller | SI Core Engine

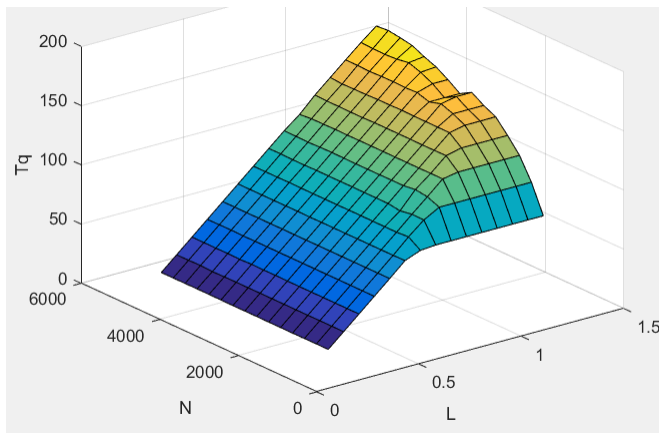
## More About

- “SI Core Engine Air Mass Flow and Torque Production” on page 2-2
- “SI Engine Simple Torque Model” on page 2-26

# SI Engine Simple Torque Model

For the simple torque lookup table model, the SI engine uses a lookup table map that is a function of engine speed and load,  $T_{brake} = f_{TL}(L, N)$ , where:

- $T_{brake}$  is engine brake torque after accounting for spark advance, AFR, and friction effects, in N·m.
- $L$  is engine load, as a normalized cylinder air mass, dimensionless.
- $N$  is engine speed, in rpm.



## See Also

SI Controller | SI Core Engine

## More About

- “SI Core Engine Air Mass Flow and Torque Production” on page 2-2
- “SI Engine Torque Structure Model” on page 2-17

## CI Core Engine Air Mass Flow and Torque Production

A compression-ignition (CI) engine produces mechanical power by injecting fuel into the combustion chamber near the end of the compression stroke. Since the combustion chamber pressure and temperature exceeds the fuel ignition limit, spontaneous ignition occurs after injection. Heat released during combustion increases the cylinder pressure. During the power stroke, the engine converts the pressure to mechanical torque.

Torque production relates to injected fuel mass, fuel injection timing, fuel pressure, and air system states. CI engines operate at lean air-fuel ratio (AFR) conditions, so the AFR is greater than the stoichiometric AFR. CI engines use exhaust gas recirculation (EGR). The exhaust gases recirculate back to the intake manifold, reducing engine-out nitric oxide and nitrogen dioxide (NO<sub>x</sub>) emissions.

### Air Mass Flow

To calculate the air mass flow, the compression-ignition (CI) engine uses the “CI Engine Speed-Density Air Mass Flow Model” on page 2-29. The speed-density model uses the speed-density equation to calculate the engine air mass flow, relating the engine intake port mass flow to the intake manifold pressure, intake manifold temperature, and engine speed.

### Torque

To calculate the engine torque, you can configure the block to use either of these torque models.

Brake Torque Model	Description
"CI Engine Torque Structure Model" on page 2-33	<p>The CI core engine torque structure model determines the engine torque by reducing the maximum engine torque potential as these engine conditions vary from nominal:</p> <ul style="list-style-type: none"><li>• Start of injection (SOI) timing</li><li>• Exhaust back-pressure</li><li>• Burned fuel mass</li><li>• Intake manifold gas pressure, temperature, and oxygen percentage</li><li>• Fuel rail pressure</li></ul> <p>To account for the effect of post-inject fuel on torque, the model uses a calibrated torque offset table.</p>
"CI Engine Simple Torque Model" on page 2-40	<p>For the simple engine torque calculation, the CI engine uses a torque lookup table map that is a function of engine speed and injected fuel mass.</p>

### See Also

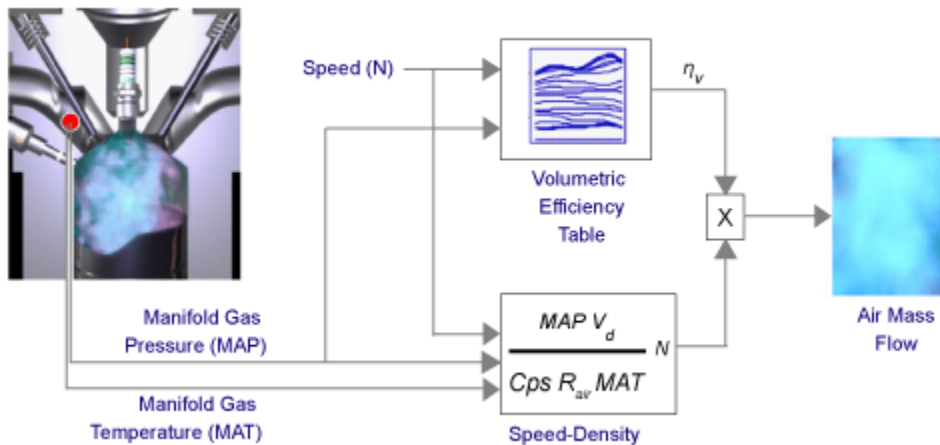
CI Controller | CI Core Engine

### More About

- "Engine Calibration Maps" on page 2-41

## CI Engine Speed-Density Air Mass Flow Model

To calculate the air mass flow in the compression-ignition (CI) engine, the CI Core Engine block uses a speed-density air mass flow model. The speed-density model uses the speed-density equation to calculate the engine air mass flow. The equation relates the engine air mass flow to the intake manifold gas pressure, intake manifold gas temperature, and engine speed. In the CI Core Engine block, the air mass flow and the cylinder air mass determine the engine load.



To determine the air mass flow, the speed-density air mass flow model uses this speed-density equation at the intake manifold and the volumetric efficiency. The model subtracts the exhaust gas recirculation (EGR) burned gas from the mass flow at the intake port.

$$\dot{m}_{port} = \frac{MAP V_d N \left[ \frac{1 \text{ min}}{60 \text{ s}} \right]}{Cps R_{air} MAT} \eta_v$$

$$\dot{m}_{air} = \dot{m}_{port} - \dot{m}_{egr}$$

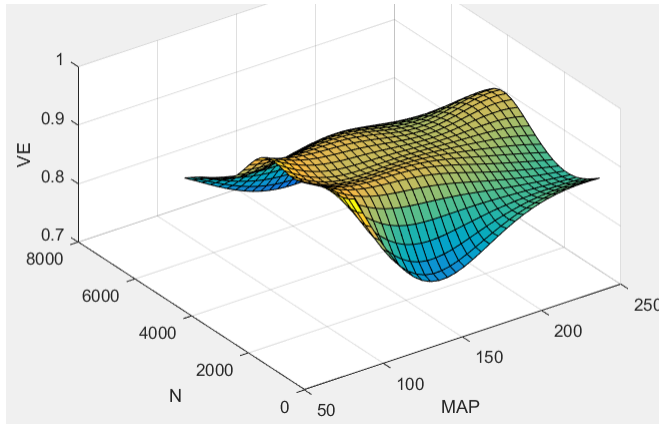
The speed-density air mass flow model uses a volumetric efficiency lookup table to determine the volumetric efficiency.

The volumetric efficiency lookup table is a function of the intake manifold absolute pressure at intake valve closing (IVC) and engine speed

$$\eta_v = f_{\eta_v}(MAP, N)$$

where:

- $\eta_v$  is engine volumetric efficiency, dimensionless.
- $MAP$  is intake manifold absolute pressure, in KPa.
- $N$  is engine speed, in rpm.



To create the volumetric efficiency table, use the air mass flow rate from measured engine performance data and the speed-density equation.

$$\eta_v = \frac{C_p R_{air} MAT}{MAP V_d N \left[ \frac{1 \text{ min}}{60 \text{ s}} \right]} \dot{m}_{air}$$

To calculate the engine load, the block divides the calculated unburned air mass by the nominal cylinder air mass. The nominal cylinder air mass is the mass of air (in kg) in a cylinder with the piston at bottom dead center (BDC), at standard air temperature and pressure.

$$M_{Nom} = \frac{P_{std} V_d}{N_{cyl} R_{air} T_{std}}$$

$$L = \frac{\left(\frac{60\text{s}}{\text{min}}\right) C_{ps} \dot{m}_{air}}{\left(\frac{1000\text{g}}{\text{kg}}\right) N_{cyl} N M_{Nom}}$$

The model implements equations that use these variables.

$\dot{m}_{air}$	Engine air mass flow
$MAP$	Cycle average intake manifold pressure
$\dot{m}_{port}$	Total engine air mass flow at intake ports, including EGR flow
$\dot{m}_{egr}$	Recirculated burned gas mass flow entering engine intake port
$V_d$	Displaced volume
$N$	Engine speed
$C_{ps}$	Crankshaft revolutions per power stroke
$R_{air}$	Ideal gas constant for air and burned gas mixture
$MAT$	Cycle average intake manifold gas absolute temperature
$\eta_v$	Engine volumetric efficiency
$f_{\eta_v}$	Engine volumetric efficiency lookup table
$L$	Engine load (normalized cylinder air mass) at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles
$M_{Nom}$	Nominal engine cylinder air mass at standard temperature and pressure; piston at bottom dead center (BDC) maximum volume
$P_{std}$	Standard pressure
$T_{std}$	Standard temperature

### References

[1] Heywood, John B. *Internal Combustion Engine Fundamentals*. New York: McGraw-Hill, 1988.

### See Also

CI Controller | CI Core Engine

### More About

- “CI Core Engine Air Mass Flow and Torque Production” on page 2-27
- “Engine Calibration Maps” on page 2-41



## CI Engine Torque Structure Model

The CI core engine torque structure model determines the engine torque by reducing the maximum engine torque potential as these engine conditions vary from nominal:

- Start of injection (SOI) timing
- Exhaust back-pressure
- Burned fuel mass
- Intake manifold gas pressure, temperature, and oxygen percentage
- Fuel rail pressure

To account for the effect of post-inject fuel on torque, the model uses a calibrated torque offset table.

To determine the engine torque, the CI core engine torque structure model implements the equations specified in these steps.

Step	Description
Step 1: Determine nominal engine inputs and states	<p>Model uses lookup tables to determine these nominal engine inputs and states as a function of compression stroke injected fuel mass, <math>F</math>, and engine speed, <math>N</math>:</p> <ul style="list-style-type: none"> <li>• Main start of injection timing, <math>SOI = f_{SOIc}(F,N)</math></li> <li>• Intake manifold gas temperature, <math>MAT = f_{MAT}(F,N)</math></li> <li>• Intake manifold gas pressure, <math>MAP = f_{MAP}(F,N)</math></li> <li>• Intake manifold oxygen percentage, <math>O2PCT = f_{O2}(F,N)</math></li> <li>• Fuel rail pressure, <math>FUELP = f_{fuelp}(F,N)</math></li> </ul>

Step	Description
<p>Step 2: Calculate relative engine states</p>	<p>To determine these relative engine states, the model calculates deviations from their nominal values.</p> <ul style="list-style-type: none"> <li>• Main start of injection timing delta, <math>\Delta SOI_c = f_{SOI}(F,N) - SOI</math></li> <li>• Intake manifold gas temperature delta, <math>\Delta MAT = f_{MAT}(F,N) - MAT</math></li> <li>• Intake manifold oxygen percentage delta, <math>\Delta O2PCT = f_{O2}(F,N) - O2PCT</math></li> <li>• Fuel rail pressure delta, <math>\Delta FUELP = f_{fuelp}(F,N) - FUELP</math></li> </ul> <p>For the intake manifold gas pressure, the block uses a pressure ratio to determine the relative state. The pressure ratio is the intake manifold gas pressure to the steady-state operating point gas pressure.</p> $MAP_{ratio} = \frac{MAP}{f_{MAP}(F,N)}$
<p>Step 3: Determine efficiency multipliers</p>	<p>Model uses gross indicated mean effective pressure (IMEPG)<sup>[1]</sup> efficiency multipliers to reduce the maximum average pressure potential of combustion. The efficiency multipliers are lookup tables that are functions of the relative engine states.</p> <ul style="list-style-type: none"> <li>• Main start of injection timing efficiency multiplier, <math>SOI_{eff} = f_{SOI_{eff}}(\Delta SOI, N)</math></li> <li>• Intake manifold gas temperature efficiency multiplier, <math>MAT_{eff} = f_{MAT_{eff}}(\Delta MAT, N)</math></li> <li>• Intake manifold gas pressure efficiency multiplier, <math>MAP_{eff} = f_{MAP_{eff}}(MAP_{ratio}, \lambda)</math></li> <li>• Intake manifold oxygen percentage efficiency multiplier, <math>O2P_{eff} = f_{O2P_{eff}}(\Delta O2P, N)</math></li> <li>• Fuel rail pressure efficiency multiplier, <math>FUELP_{eff} = f_{FUELP_{eff}}(\Delta FUELP, N)</math></li> </ul>

Step	Description
<p>Step 4: Determine indicated mean effective cylinder pressure (IMEP) available for torque production</p>	<p>To determine the IMEP available for torque production, the model implements these equations.</p> $IMEP = SOI_{eff} MAP_{eff} MAT_{eff} O2p_{eff} FUELP_{eff} IMEPG$ $IMEPG = f_{IMEP_g}(F, N)$ <p>The model multiplies the efficiency multipliers from step 3 by the IMEPG. The model implements IMEPG as lookup table that is a function of the of compression stroke injected fuel mass, <math>F</math>, and engine speed, <math>N</math>.</p>
<p>Step 5: Account for losses due to friction</p>	<p>To account for friction effects, the model uses the nominal friction mean effective pressure (FMEP)<sup>[1]</sup> to implement this equation.</p> $FMEP = f_{FMEP}(F, N) f_{fmod}(T_{oil}, N)$ <p>The model implements FMEP as lookup table that is a function of the of compression stroke injected fuel mass, <math>F</math>, and engine speed, <math>N</math>. To account for the temperature effect on friction, the model use a lookup table that is a function of oil temperature, <math>T_{oil}</math>, and <math>N</math>.</p>

Step	Description
<p>Step 6: Account for pressure loss due to pumping</p>	<p>To account for pressure losses due to pumping, the model uses the nominal pumping mean effective pressure (PMEP)<sup>[1]</sup> to implement these equations.</p> $\Delta MAP = f_{MAP}(F, N) - MAP$ $\Delta EMAP = f_{EMAP}(F, N) - EMAP$ $PMEP = f_{PMEP}(F, N) - \Delta MAP + \Delta EMAP$ <p>The model implements MAP and EMAP as lookup tables that are functions of the of compression stroke injected fuel mass, <math>F</math>, and engine speed, <math>N</math>. Under normal operating conditions, PMEP is negative, indicating a loss of cylinder pressure.</p>
<p>Step 7: Account for late fuel injection SOI timing on IMEP</p>	<p>To account for late fuel injection SOI timing on IMEP, <math>\Delta IMEP_{post}</math>, the model uses a lookup table that is a function of the effective pressure post inject SOI timing centroid, <math>SOI_{post}</math>, and the post inject mass sum, <math>F_{post}</math>.</p> $\Delta IMEP_{post} = f_{\Delta IMEP_{post}}(SOI_{post}, F_{post})$
<p>Step 8: Calculate engine brake torque</p>	<p>To calculate the engine brake torque, <math>T_{brake}</math>, the model converts the brake mean effective pressure (BMEP)<sup>[1]</sup> to engine brake torque using these equations. The BMEP calculation accounts for all gross mean effective pressure losses. <math>V_d</math> is displaced cylinder volume. <math>Cps</math> is the number of power strokes per revolution.</p> $BMEP = IMEPG + \Delta IMEP_{post} - FMPEP + PMEP$ $T_{brake} = \frac{V_d}{2\pi Cps} BMEP$

## Fuel Injection

In the CI Core Engine and CI Controller blocks, you can represent multiple injections with the start of injection (SOI) and fuel mass inputs to the model. To specify the type of injection, use the **Fuel mass injection type identifier** parameter.

Type of Injection	Parameter Value
Pilot	0
Main	1
Post	2
Passed	3

The model considers Passed fuel injections and fuel injected later than a threshold to be unburned fuel. Use the **Maximum start of injection angle for burned fuel,  $f\_tqs\_f\_burned\_soi\_limit$**  parameter to specify the threshold.

## Percent Oxygen

The model uses this equation to calculate the oxygen percent,  $O2p$ .  $y_{in,air}$  is the unburned air mass fraction.

$$O2p = 23.13y_{in,air}$$

## Exhaust Temperature

The exhaust temperature calculation depends on the torque model. For both torque models, the block implements lookup tables.

Torque Model	Description	Equations
Simple Torque Lookup	Exhaust temperature lookup table is a function of the injected fuel mass and engine speed.	$T_{exh} = f_{T_{exh}}(F, N)$

Torque Model	Description	Equations
Torque Structure	<p>The exhaust temperature is a product of these exhaust temperature efficiencies:</p> <ul style="list-style-type: none"> <li>• SOI timing</li> <li>• Intake manifold gas pressure</li> <li>• Intake manifold gas temperature</li> <li>• Intake manifold gas oxygen percentage</li> <li>• Fuel rail pressure</li> <li>• Optimal temperature</li> </ul> <p>To determine the efficiencies, the block uses lookup tables.</p>	$T_{exh} = SOI_{exhteff} MAP_{exhteff} MAT_{exhteff} O2p_{exhteff} FUELP_{exhteff}$ $SOI_{exhteff} = f_{SOI_{exhteff}}(\Delta SOI, N)$ $MAP_{exhteff} = f_{MAP_{exhteff}}(MAP_{ratio}, \lambda)$ $MAT_{exhteff} = f_{MAT_{exhteff}}(\Delta MAT, N)$ $O2p_{exhteff} = f_{O2p_{exhteff}}(\Delta O2p, N)$ $Texh_{opt} = f_{Texh_{opt}}(F, N)$

The equations use these variables.

$F$	Compression stroke injected fuel mass
$N$	Engine speed
$Texh$	Exhaust manifold gas temperature
$Texh_{opt}$	Optimal exhaust manifold gas temperature
$SOI_{exhteff}$	Main SOI exhaust temperature efficiency multiplier
$\Delta SOI$	Main SOI timing relative to optimal timing
$MAP_{exhteff}$	Intake manifold gas pressure exhaust temperature efficiency multiplier
$MAP_{ratio}$	Intake manifold gas pressure ratio relative to optimal pressure ratio
$\lambda$	Intake manifold gas lambda
$MAT_{exhteff}$	Intake manifold gas temperature exhaust temperature efficiency multiplier
$\Delta MAT$	Intake manifold gas temperature relative to optimal temperature
$O2P_{exhteff}$	Intake manifold gas oxygen exhaust temperature efficiency multiplier
$\Delta O2P$	Intake gas oxygen percent relative to optimal
$FUELP_{exhteff}$	Fuel rail pressure exhaust temperature efficiency multiplier

$\Delta FUEL_P$  Fuel rail pressure relative to optimal

## References

[1] Heywood, John B. *Internal Engine Combustion Fundamentals*. New York: McGraw-Hill, 1988.

## See Also

CI Controller | CI Core Engine

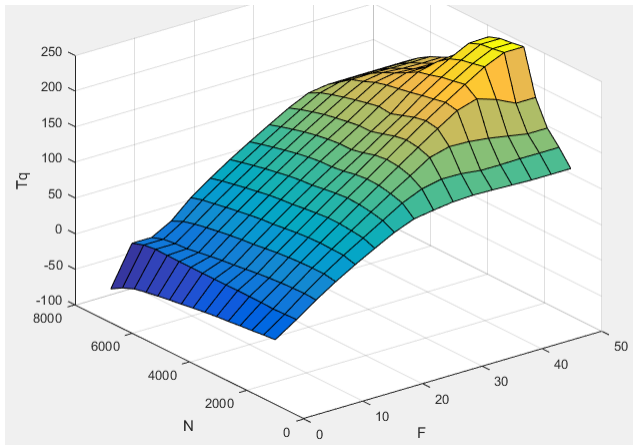
## More About

- “CI Core Engine Air Mass Flow and Torque Production” on page 2-27
- “CI Engine Simple Torque Model” on page 2-40

### CI Engine Simple Torque Model

For the simple torque lookup table model, the CI engine uses a lookup table as a function of engine speed and injected fuel mass,  $T_{brake} = f_{T_{nf}}(F, N)$ , where:

- $Tq = T_{brake}$  is engine brake torque after accounting for engine mechanical and pumping friction effects, in N·m.
- $F$  is injected fuel mass, in mg per injection.
- $N$  is engine speed, in rpm.



### See Also

CI Controller | CI Core Engine

### More About

- “CI Core Engine Air Mass Flow and Torque Production” on page 2-27
- “CI Engine Torque Structure Model” on page 2-33



## Engine Calibration Maps

Calibration maps are a key part of the engine plant and controller models available in the Powertrain 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 and controller 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 and controllers 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 Powertrain Blockset SI and CI reference applications affect the engine response to control inputs (for example, spark timing, throttle position, and cam phasing).

To develop the calibration maps in the Powertrain Blockset 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.

### Engine Controller Calibration Maps

The engine controller model calibration maps in the reference applications represent the optimal open-loop control commands for given engine operating points.

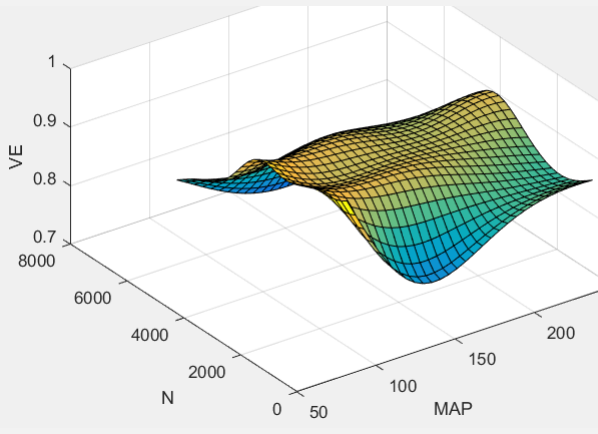
To develop the calibration maps for the SI engine controller, MathWorks used the GT-POWER reference engine models in a virtual engine calibration optimization (VECO) process. The process optimized the open-loop control commands for 1.5-L SI engine,

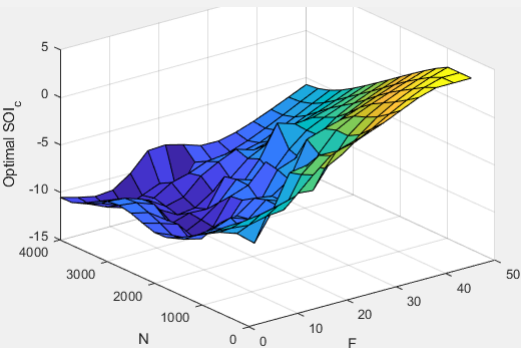
subject to engine operating constraints for knock, turbocharger speed, and exhaust temperature.

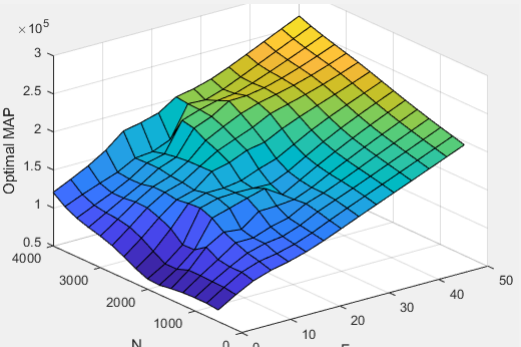
To develop the calibration maps for the CI engine controller, MathWorks used the DOE test data from the GT-POWER 1.5-L CI reference model operated at minimum brake-specific fuel consumption (BSFC).

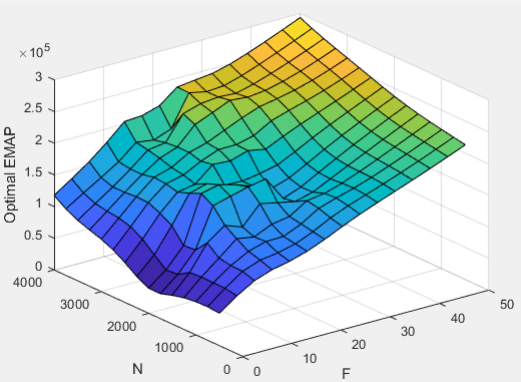
### **Calibration Maps in Compression-Ignition (CI) Blocks**

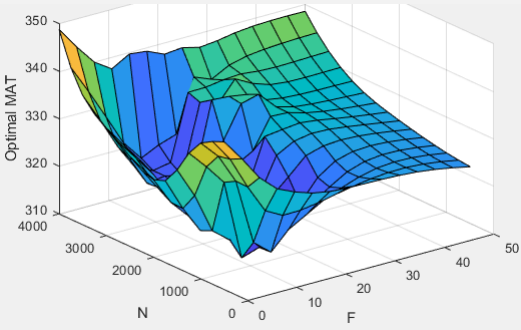
In the engine models, the Powertrain Blockset blocks implement these calibration maps.

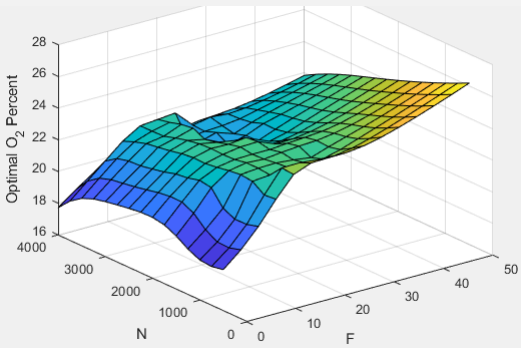
Map	Used For	In	Description
Volumetric efficiency	"CI Engine Speed-Density Air Mass Flow Model" on page 2-29	CI Core Engine CI Controller	<p>The volumetric efficiency lookup table is a function of the intake manifold absolute pressure at intake valve closing (IVC) and engine speed</p> $\eta_v = f_{\eta_v}(MAP, N)$ <p>where:</p> <ul style="list-style-type: none"> <li>• <math>\eta_v</math> is engine volumetric efficiency, dimensionless.</li> <li>• <math>MAP</math> is intake manifold absolute pressure, in KPa.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

Map	Used For	In	Description
Optimal main start of injection (SOI) timing	"CI Engine Torque Structure Model" on page 2-33	CI Core Engine  CI Controller	<p>The optimal main start of injection (SOI) timing lookup table, <math>f_{SOIc}</math>, is a function of the engine speed and injected fuel mass, <math>SOI_c = f_{SOIc}(F, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>SOI_c</math> is optimal SOI timing, in degATDC.</li> <li>• <math>F</math> is compression stroke injected fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

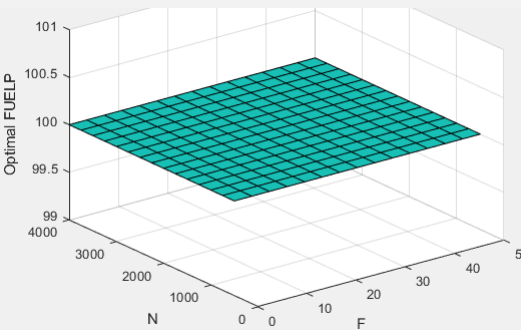
Map	Used For	In	Description
Optimal intake manifold gas pressure	"CI Engine Torque Structure Model" on page 2-33	CI Core Engine  CI Controller	<p>The optimal intake manifold gas pressure lookup table, <math>f_{MAP}</math>, is a function of the engine speed and injected fuel mass, <math>MAP = f_{MAP}(F, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>MAP</math> is optimal intake manifold gas pressure, in Pa.</li> <li>• <math>F</math> is compression stroke injected fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

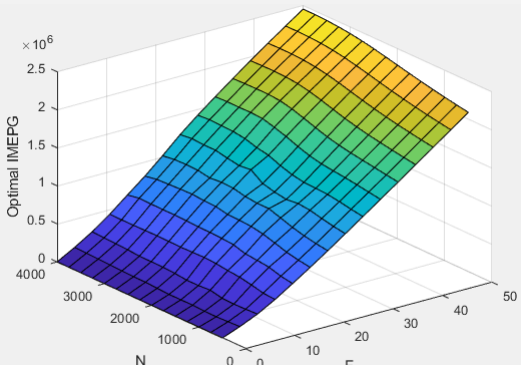
Map	Used For	In	Description
Optimal exhaust manifold gas pressure	"CI Engine Torque Structure Model" on page 2-33	CI Core Engine  CI Controller	<p>The optimal exhaust manifold gas pressure lookup table, <math>f_{EMAP}</math>, is a function of the engine speed and injected fuel mass, <math>EMAP = f_{EMAP}(F, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>EMAP</math> is optimal exhaust manifold gas pressure, in Pa.</li> <li>• <math>F</math> is compression stroke injected fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

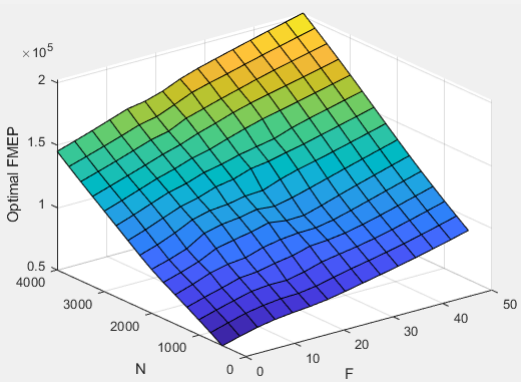
Map	Used For	In	Description
Optimal intake manifold gas temperature	"CI Engine Torque Structure Model" on page 2-33	CI Core Engine  CI Controller	<p>The optimal intake manifold gas temperature lookup table, <math>f_{MAT}</math>, is a function of the engine speed and injected fuel mass, <math>MAT = f_{MAT}(F, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>MAT</math> is optimal intake manifold gas temperature, in K.</li> <li>• <math>F</math> is compression stroke injected fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

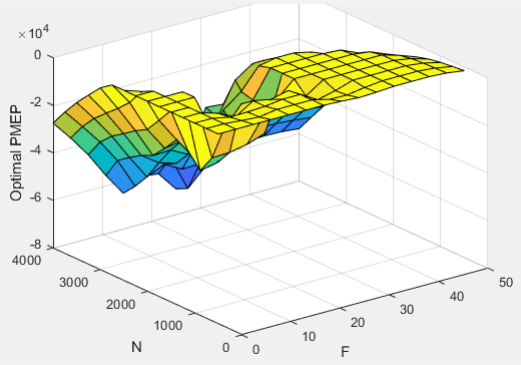
Map	Used For	In	Description
Optimal intake gas oxygen percent	"CI Engine Torque Structure Model" on page 2-33	CI Core Engine  CI Controller	<p>The optimal intake gas oxygen percent lookup table, <math>f_{O_2}</math>, is a function of the engine speed and injected fuel mass, <math>O2PCT = f_{O_2}(F,N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>O2PCT</math> is optimal intake gas oxygen, in percent.</li> <li>• <math>F</math> is compression stroke injected fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

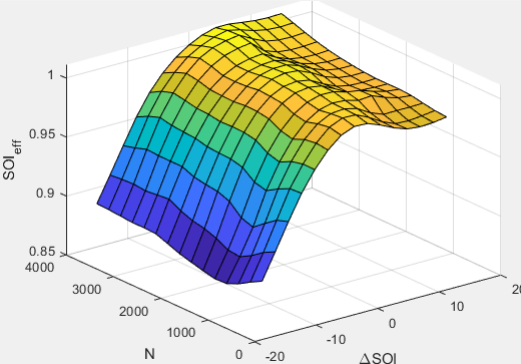


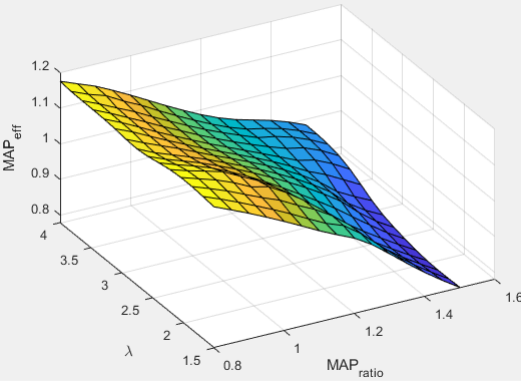
Map	Used For	In	Description
Optimal fuel rail pressure	"CI Engine Torque Structure Model" on page 2-33	CI Core Engine  CI Controller	<p>The optimal fuel rail pressure lookup table, <math>f_{fuelp}</math>, is a function of the engine speed and injected fuel mass, <math>FUELP = f_{fuelp}(F,N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>FUELP</math> is optimal fuel rail pressure, in MPa.</li> <li>• <math>F</math> is compression stroke injected fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul>  <p>The figure is a 3D surface plot showing the relationship between engine speed (N), injected fuel mass (F), and optimal fuel rail pressure (FUELP). The vertical axis (FUELP) ranges from 99 to 101. The horizontal axis (N) ranges from 0 to 4000. The depth axis (F) ranges from 0 to 50. The surface is a flat, teal-colored plane at a constant FUELP value of approximately 100 MPa across the entire range of N and F.</p>

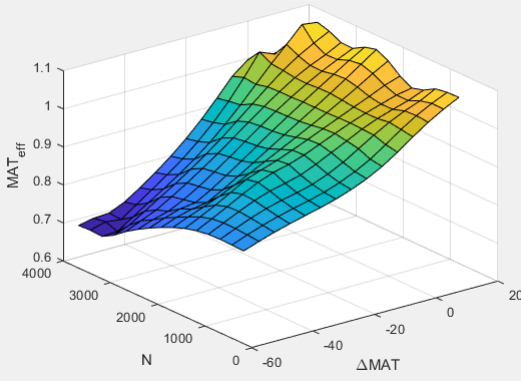
Map	Used For	In	Description
<p>Optimal gross indicated mean effective pressure</p>	<p>“CI Engine Torque Structure Model” on page 2-33</p>	<p>CI Core Engine CI Controller</p>	<p>The optimal gross indicated mean effective pressure lookup table, <math>f_{imepg}</math>, is a function of the engine speed and injected fuel mass, <math>IMEPG = f_{imepg}(F,N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>IMEPG</math> is optimal gross indicated mean effective pressure, in Pa.</li> <li>• <math>F</math> is compression stroke injected fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

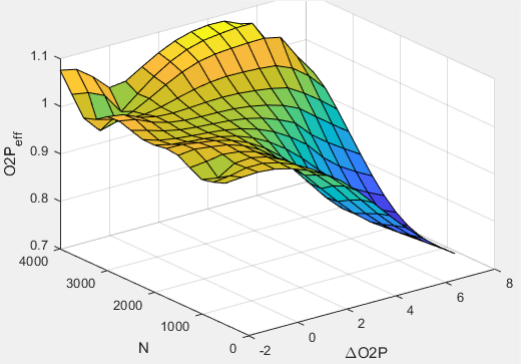
Map	Used For	In	Description
Optimal friction mean effective pressure	"CI Engine Torque Structure Model" on page 2-33	CI Core Engine  CI Controller	<p>The optimal friction mean effective pressure lookup table, <math>f_{f_{mep}}</math>, is a function of the engine speed and injected fuel mass, <math>FMEP = f_{f_{mep}}(F, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>FMEP</math> is optimal friction mean effective pressure, in Pa.</li> <li>• <math>F</math> is compression stroke injected fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

Map	Used For	In	Description
Optimal pumping mean effective pressure	"CI Engine Torque Structure Model" on page 2-33	CI Core Engine  CI Controller	<p>The optimal pumping mean effective pressure lookup table, <math>f_{pmeP}</math>, is a function of the engine speed and injected fuel mass, <math>PMEP = f_{pmeP}(F, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>PMEP</math> is optimal pumping mean effective pressure, in Pa.</li> <li>• <math>F</math> is compression stroke injected fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

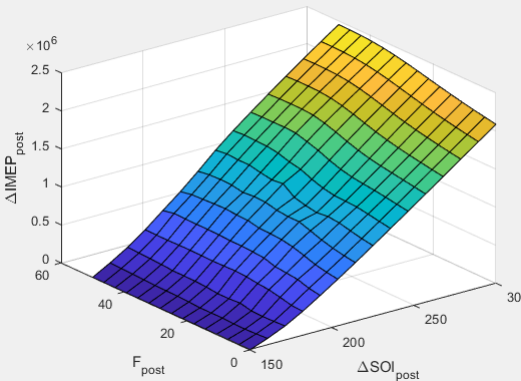
Map	Used For	In	Description
Main SOI timing efficiency multiplier	"CI Engine Torque Structure Model" on page 2-33	CI Core Engine  CI Controller	<p>The main start of injection (SOI) timing efficiency multiplier lookup table, <math>f_{SOI_{eff}}</math>, is a function of the engine speed and main SOI timing relative to optimal timing, <math>SOI_{eff} = f_{SOI_{eff}}(\Delta SOI, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>SOI_{eff}</math> is main SOI timing efficiency multiplier, dimensionless.</li> <li>• <math>\Delta SOI</math> is main SOI timing relative to optimal timing, in degBTDC.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul>  <p>The figure is a 3D surface plot showing the relationship between engine speed (N), main SOI timing relative to optimal timing (ΔSOI), and the main SOI timing efficiency multiplier (SOI_eff). The vertical axis (SOI_eff) ranges from 0.85 to 1.0. The horizontal axis (N) ranges from 0 to 3000 rpm. The depth axis (ΔSOI) ranges from -20 to 20 degBTDC. The surface shows that SOI_eff is highest (near 1.0) at low engine speeds and near-optimal timing, and decreases as engine speed increases and timing deviates from optimal.</p>

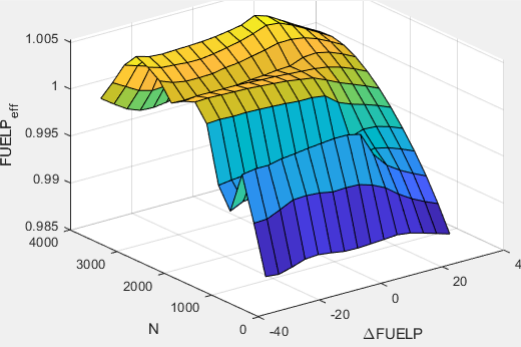
Map	Used For	In	Description
Intake manifold gas pressure efficiency multiplier	"CI Engine Torque Structure Model" on page 2-33	CI Core Engine  CI Controller	<p>The intake manifold gas pressure efficiency multiplier lookup table, <math>f_{MAP_{eff}}</math>, is a function of the intake manifold gas pressure ratio relative to optimal pressure ratio and lambda, <math>MAP_{eff} = f_{MAP_{eff}}(MAP_{ratio}, \lambda)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>MAP_{eff}</math> is intake manifold gas pressure efficiency multiplier, dimensionless.</li> <li>• <math>MAP_{ratio}</math> is intake manifold gas pressure ratio relative to optimal pressure ratio, dimensionless.</li> <li>• <math>\lambda</math> is intake manifold gas lambda, dimensionless.</li> </ul>  <p>A 3D surface plot showing the relationship between the intake manifold gas pressure ratio (MAP_ratio), the intake manifold gas lambda (λ), and the intake manifold gas pressure efficiency multiplier (MAP_eff). The vertical axis (MAP_eff) ranges from 0.8 to 1.2. The horizontal axis (MAP_ratio) ranges from 0.8 to 1.6. The depth axis (λ) ranges from 1 to 4. The surface is colored with a gradient from yellow (high efficiency) to blue (low efficiency), showing a peak efficiency of approximately 1.15 at a MAP_ratio of 1.0 and λ of 1.0, which decreases as λ increases and MAP_ratio deviates from 1.0.</p>

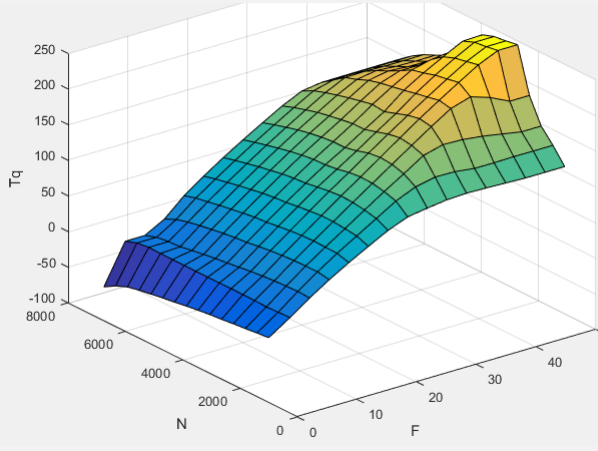
Map	Used For	In	Description
Intake manifold gas temperature efficiency multiplier	"CI Engine Torque Structure Model" on page 2-33	CI Core Engine  CI Controller	<p>The intake manifold gas temperature efficiency multiplier lookup table, <math>f_{MAT_{eff}}</math>, is a function of the engine speed and intake manifold gas temperature relative to optimal temperature, <math>MAT_{eff} = f_{MAT_{eff}}(\Delta MAT, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>MAT_{eff}</math> is intake manifold gas temperature efficiency multiplier, dimensionless.</li> <li>• <math>\Delta MAT</math> is intake manifold gas temperature relative to optimal temperature, in K.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

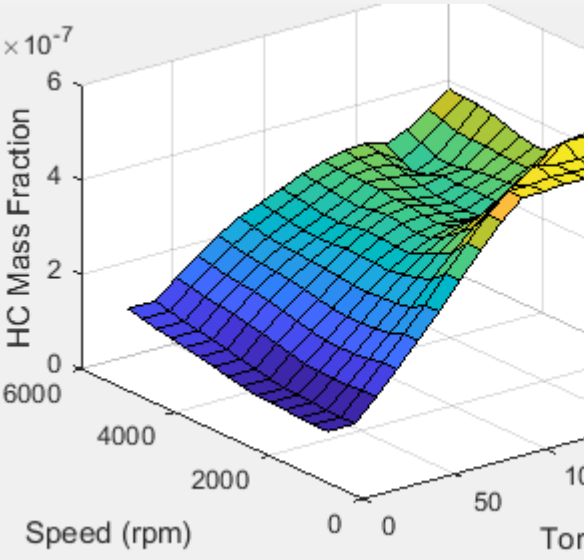
Map	Used For	In	Description
Intake manifold gas oxygen efficiency multiplier	"CI Engine Torque Structure Model" on page 2-33	CI Core Engine  CI Controller	<p>The intake manifold gas oxygen efficiency multiplier lookup table, <math>f_{O_2P_{eff}}</math>, is a function of the engine speed and intake manifold gas oxygen percent relative to optimal, <math>O_2P_{eff} = f_{O_2P_{eff}}(\Delta O_2P, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>O_2P_{eff}</math> is intake manifold gas oxygen efficiency multiplier, dimensionless.</li> <li>• <math>\Delta O_2P</math> is intake gas oxygen percent relative to optimal, in percent.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul>  <p>The figure is a 3D surface plot showing the relationship between engine speed (N), intake manifold gas oxygen percent relative to optimal (ΔO2P), and the intake manifold gas oxygen efficiency multiplier (O2P_eff). The vertical axis (O2P_eff) ranges from 0.7 to 1.1. The horizontal axis (N) ranges from 0 to 4000 rpm. The depth axis (ΔO2P) ranges from -2 to 8 percent. The surface shows a peak in efficiency around 2000 rpm and 0% ΔO2P, with efficiency decreasing as engine speed increases and ΔO2P deviates from 0.</p>

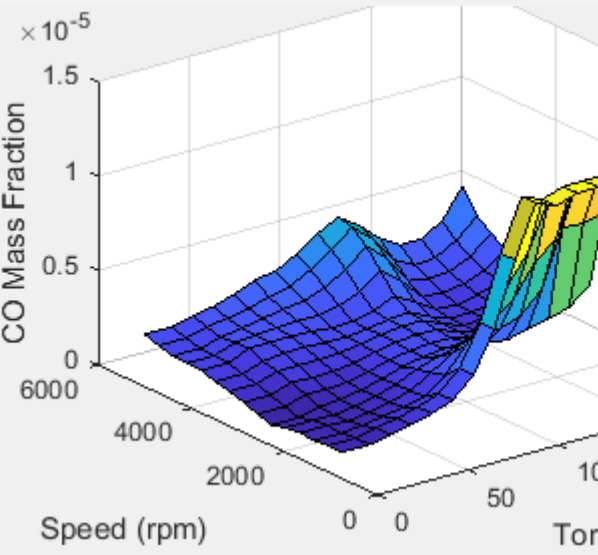


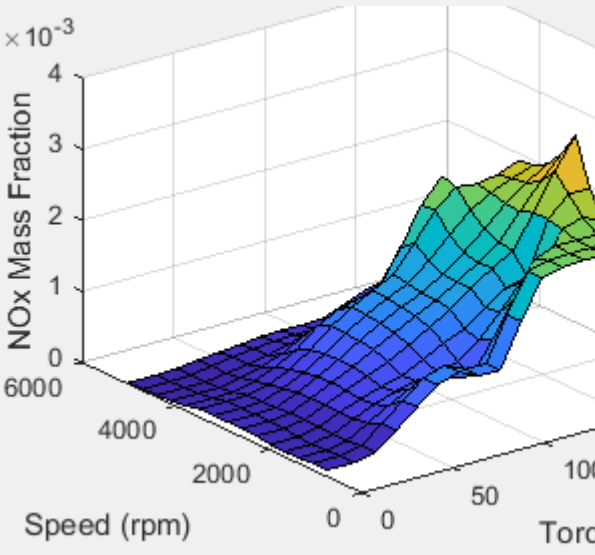
Map	Used For	In	Description
<p>Indicated mean effective pressure post inject correction</p>	<p>“CI Engine Torque Structure Model” on page 2-33</p>	<p>CI Core Engine  CI Controller</p>	<p>The indicated mean effective pressure post inject correction lookup table, <math>f_{IMEP_{post}}</math>, is a function of the engine speed and fuel rail pressure relative to optimal breakpoints, <math>\Delta IMEP_{post} = f_{IMEP_{post}}(\Delta SOI_{post}, F_{post})</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>\Delta IMEP_{post}</math> is indicated mean effective pressure post inject correction, in Pa.</li> <li>• <math>\Delta SOI_{post}</math> is indicated mean effective pressure post inject start of inject timing centroid, in degATDC.</li> <li>• <math>F_{post}</math> is indicated mean effective pressure post inject mass sum, in mg per injection.</li> </ul> 

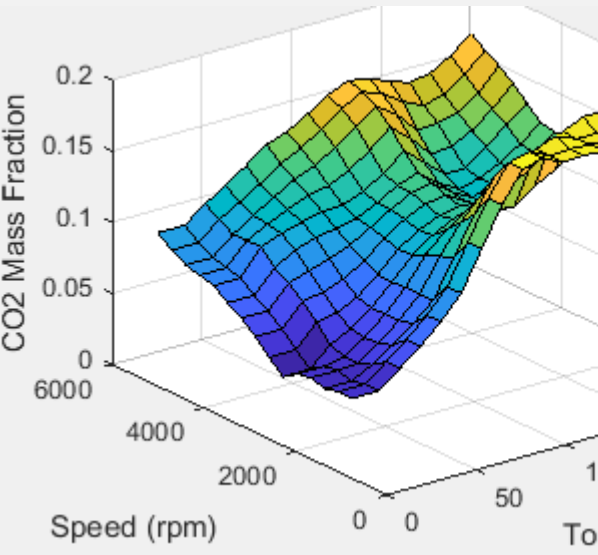
Map	Used For	In	Description
Fuel rail pressure efficiency multiplier	"CI Engine Torque Structure Model" on page 2-33	CI Core Engine  CI Controller	<p>The fuel rail pressure efficiency multiplier lookup table, <math>f_{FUELPeff}</math>, is a function of the engine speed and fuel rail pressure relative to optimal breakpoints, <math>FUELPeff = f_{FUELPeff}(\Delta FUEL, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>FUELPeff</math> is fuel rail pressure efficiency multiplier, dimensionless.</li> <li>• <math>\Delta FUEL</math> is fuel rail pressure relative to optimal, in MPa.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

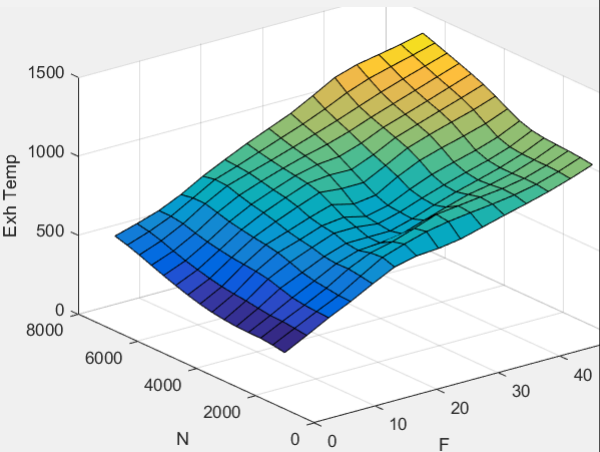
Map	Used For	In	Description
<p>Engine brake torque</p>	<p>“CI Engine Simple Torque Model” on page 2-40</p>	<p>CI Core Engine  CI Controller</p>	<p>For the simple torque lookup table model, the CI engine uses a lookup table is a function of engine speed and injected fuel mass,</p> <p><math>T_{brake} = f_{Tnf}(F, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>Tq = T_{brake}</math> is engine brake torque after accounting for engine mechanical and pumping friction effects, in N·m.</li> <li>• <math>F</math> is injected fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

Map	Used For	In	Description
Hydrocarbon (HC) mass fraction	HC emissions	CI Core Engine	<p>The CI Core Engine HC emission mass fraction lookup table is a function of engine torque and engine speed, <math>HC\ Mass\ Fraction = f(Speed, Torque)</math>, where:</p> <ul style="list-style-type: none"> <li>• <i>HC Mass Fraction</i> is the HC emission mass fraction, dimensionless.</li> <li>• <i>Speed</i> is engine speed, in rpm.</li> <li>• <i>Torque</i> is engine torque, in N·m.</li> </ul> 

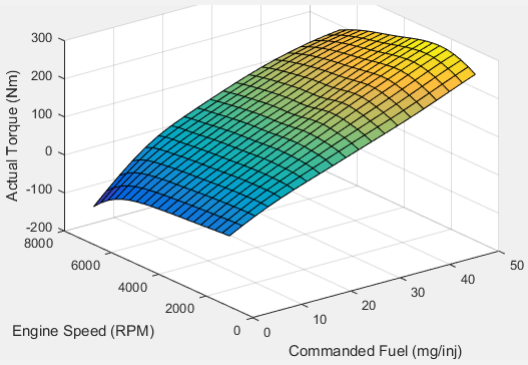
Map	Used For	In	Description
Carbon monoxide (CO) mass fraction	CO emissions	CI Core Engine	<p>The CI Core Engine CO emission mass fraction lookup table is a function of engine torque and engine speed, <math>CO\ Mass\ Fraction = f(Speed, Torque)</math>, where:</p> <ul style="list-style-type: none"> <li>• <i>CO Mass Fraction</i> is the CO emission mass fraction, dimensionless.</li> <li>• <i>Speed</i> is engine speed, in rpm.</li> <li>• <i>Torque</i> is engine torque, in N·m.</li> </ul> 

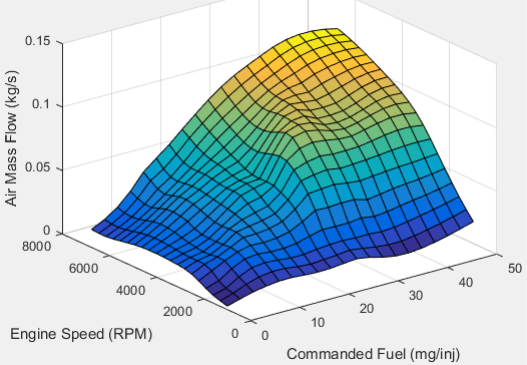
Map	Used For	In	Description
Nitric oxide and nitrogen dioxide (NOx) mass fraction	NOx emissions	CI Core Engine	<p>The CI Core Engine NOx emission mass fraction lookup table is a function of engine torque and engine speed, <math>NOx\ Mass\ Fraction = f(Speed, Torque)</math>, where:</p> <ul style="list-style-type: none"> <li>• <i>NOx Mass Fraction</i> is the NOx emission mass fraction, dimensionless.</li> <li>• <i>Speed</i> is engine speed, in rpm.</li> <li>• <i>Torque</i> is engine torque, in N·m.</li> </ul> 

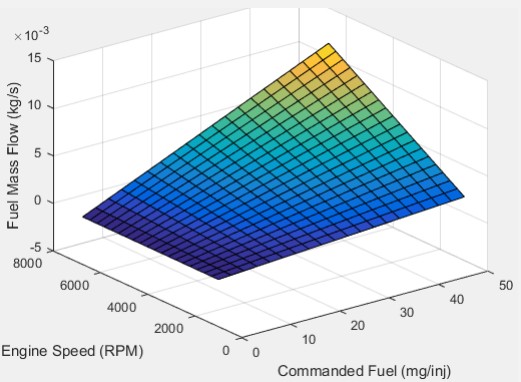
Map	Used For	In	Description
Carbon dioxide (CO <sub>2</sub> ) mass fraction	CO <sub>2</sub> emissions	CI Core Engine	<p>The CI Core Engine CO<sub>2</sub> emission mass fraction lookup table is a function of engine torque and engine speed, <math>CO_2 \text{ Mass Fraction} = f(\text{Speed}, \text{Torque})</math>, where:</p> <ul style="list-style-type: none"> <li>• <i>CO<sub>2</sub> Mass Fraction</i> is the CO<sub>2</sub> emission mass fraction, dimensionless.</li> <li>• <i>Speed</i> is engine speed, in rpm.</li> <li>• <i>Torque</i> is engine torque, in N·m.</li> </ul> 

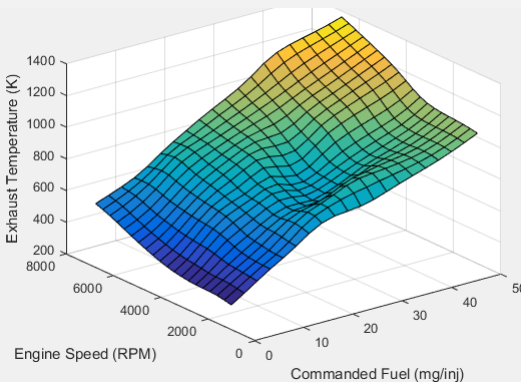
Map	Used For	In	Description
Exhaust temperature	Engine exhaust temperature as a function of injected fuel mass and engine speed	CI Core Engine CI Controller	<p>The lookup table for the exhaust temperature is a function of injected fuel mass and engine speed</p> $T_{exh} = f_{T_{exh}}(F, N)$ <p>where:</p> <ul style="list-style-type: none"> <li>• <math>T_{exh}</math> is exhaust temperature, in K.</li> <li>• <math>F</math> is injected fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul>  <p>The figure is a 3D surface plot showing the relationship between engine speed (N), injected fuel mass (F), and exhaust temperature (Exh Temp). The vertical axis (Exh Temp) ranges from 0 to 1500 K. The horizontal axis (N) ranges from 0 to 6000 rpm. The depth axis (F) ranges from 0 to 50 mg per injection. The surface shows that exhaust temperature increases as both engine speed and fuel mass increase, with the highest temperatures (yellow/orange) occurring at high engine speeds and high fuel masses, and the lowest temperatures (blue) occurring at low engine speeds and low fuel masses.</p>

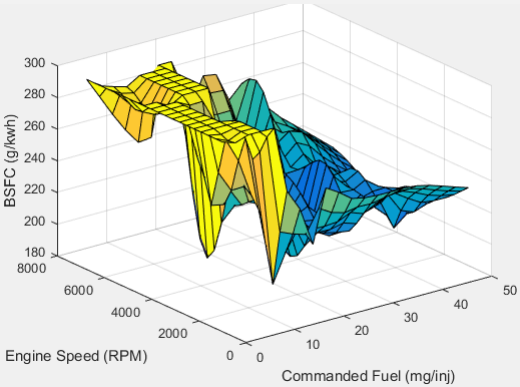


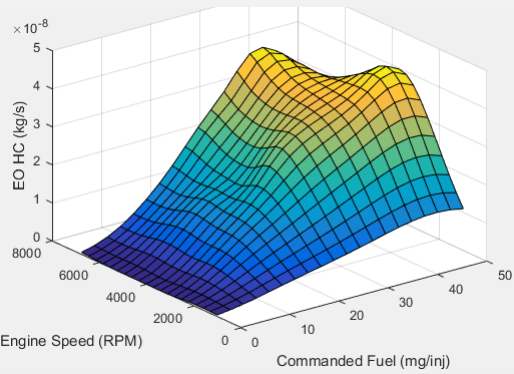
Map	Used For	In	Description
Engine brake torque	Engine brake torque as a function of commanded fuel mass and engine speed	Mapped CI Engine	<p>The engine brake torque lookup table is a function of commanded fuel mass and engine speed, <math>T_{brake} = f(F, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>T_{brake}</math> is engine torque, in N·m.</li> <li>• <math>F</math> is commanded fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

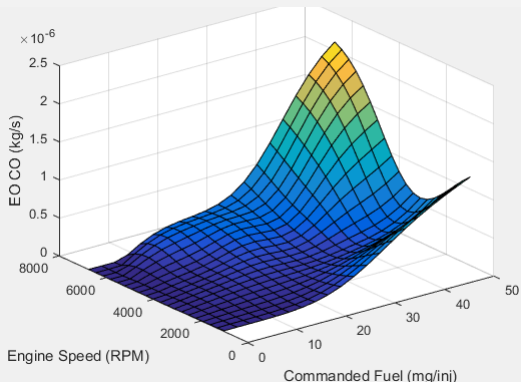
Map	Used For	In	Description
Engine air mass flow	Engine air mass flow as a function of commanded fuel mass and engine speed	Mapped CI Engine	<p>The air mass flow lookup table is a function of commanded fuel mass and engine speed, <math>\dot{m}_{intk} = f(F_{max}, N)</math>, where:</p> <ul style="list-style-type: none"> <li><math>\dot{m}_{intk}</math> is engine air mass flow, in kg/s.</li> <li><math>F_{max}</math> is commanded fuel mass, in mg per injection.</li> <li><math>N</math> is engine speed, in rpm.</li> </ul> 

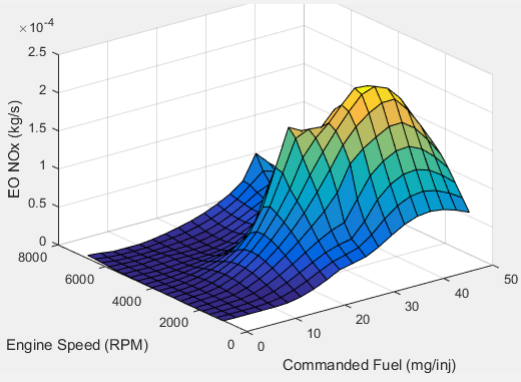
Map	Used For	In	Description
Engine fuel flow	Engine fuel flow as a function of commanded fuel mass and engine speed	Mapped CI Engine	<p>The engine fuel flow lookup table is a function of commanded fuel mass and engine speed, <math>MassFlow = f(F, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>MassFlow</math> is engine fuel mass flow, in kg/s.</li> <li>• <math>F</math> is commanded fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

Map	Used For	In	Description
Engine exhaust temperature	Engine exhaust temperature as a function of commanded fuel mass and engine speed	Mapped CI Engine	<p>The engine exhaust temperature table is a function of commanded fuel mass and engine speed, <math>T_{exh} = f(F, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>T_{exh}</math> is exhaust temperature, in K.</li> <li>• <math>F</math> is commanded fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

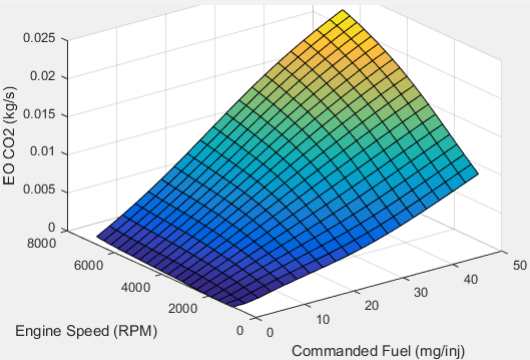
Map	Used For	In	Description
Brake-specific fuel consumption (BSFC) efficiency	BSFC efficiency as a function of commanded fuel mass and engine speed	Mapped CI Engine	<p>The brake-specific fuel consumption (BSFC) efficiency is a function of commanded fuel mass and engine speed, <math>BSFC = f(F, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>BSFC</math> is BSFC, in g/kWh.</li> <li>• <math>F</math> is commanded fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

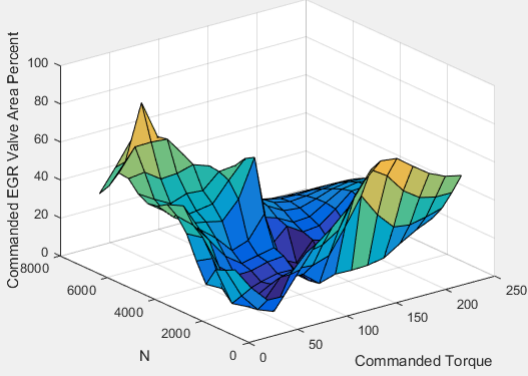
Map	Used For	In	Description
<p>Engine-out (EO) hydrocarbon emissions</p>	<p>EO hydrocarbon emissions as a function of commanded fuel mass and engine speed</p>	<p>Mapped CI Engine</p>	<p>The engine-out hydrocarbon emissions are a function of commanded fuel mass and engine speed, <math>EO\ HC = f(F, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>EO\ HC</math> is engine-out hydrocarbon emissions, in kg/s.</li> <li>• <math>F</math> is commanded fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

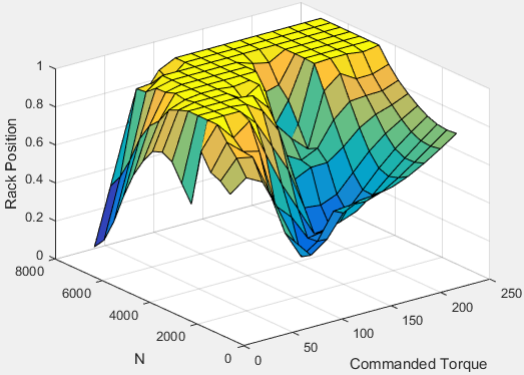
Map	Used For	In	Description
<p>Engine-out (EO) carbon monoxide emissions</p>	<p>EO carbon monoxide emissions as a function of commanded fuel mass and engine speed</p>	<p>Mapped CI Engine</p>	<p>The engine-out carbon monoxide emissions are a function of commanded fuel mass and engine speed, <math>EO\ CO = f(F, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>EO\ CO</math> is engine-out carbon monoxide emissions, in kg/s.</li> <li>• <math>F</math> is commanded fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul>  <p>The figure is a 3D surface plot showing the relationship between engine speed, commanded fuel mass, and engine-out carbon monoxide emissions. The vertical axis represents EO CO in kg/s, scaled by 10<sup>-6</sup>, with values from 0 to 2.5. The horizontal axes are Engine Speed (RPM) from 0 to 8000 and Commanded Fuel (mg/inj) from 0 to 50. The surface shows a peak in emissions at approximately 4000 RPM and 10 mg/inj, with values decreasing as engine speed increases further or fuel mass decreases.</p>

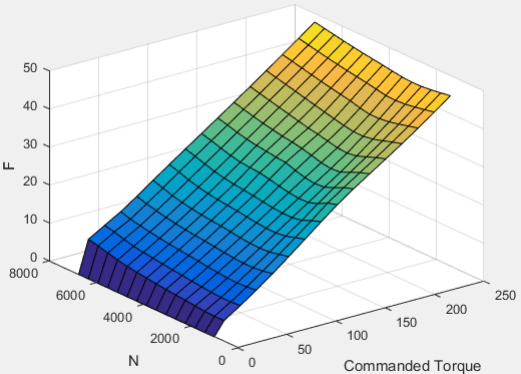
Map	Used For	In	Description
Engine-out (EO) nitric oxide and nitrogen dioxide	EO nitric oxide and nitrogen dioxide emissions as a function of commanded fuel mass and engine speed	Mapped CI Engine	<p>The engine-out nitric oxide and nitrogen dioxide emissions are a function of commanded fuel mass and engine speed, <math>EO\ NO_x = f(F, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>EO\ NO_x</math> is engine-out nitric oxide and nitrogen dioxide emissions, in kg/s.</li> <li>• <math>F</math> is commanded fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul>  <p>The figure is a 3D surface plot showing the relationship between engine speed, commanded fuel mass, and engine-out NOx emissions. The vertical axis represents EO NOx in kg/s, scaled by 10^-4, with values from 0 to 2.5. The horizontal axes are Engine Speed (RPM) from 0 to 6000 and Commanded Fuel (mg/inj) from 0 to 50. The surface shows a peak in emissions at approximately 4000 RPM and 30 mg/inj, with values decreasing as engine speed increases further or fuel mass decreases.</p>

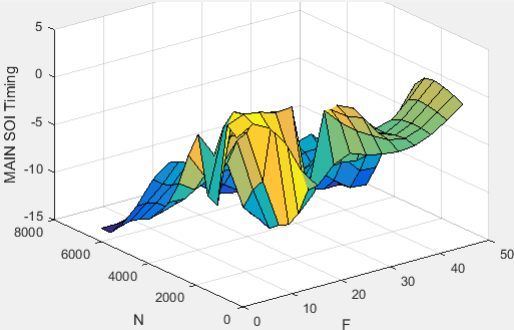


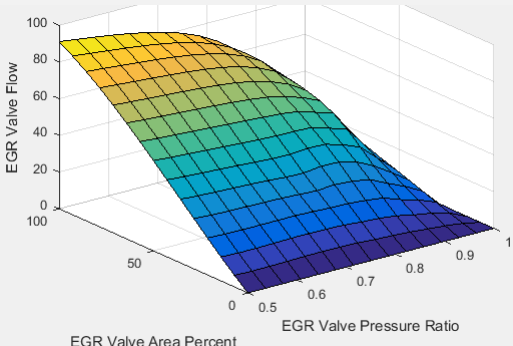
Map	Used For	In	Description
<p>Engine-out (EO) carbon dioxide emissions</p>	<p>EO carbon dioxide emissions as a function of commanded fuel mass and engine speed</p>	<p>Mapped CI Engine</p>	<p>The engine-out carbon dioxide emissions are a function of commanded fuel mass and engine speed, <math>EO\ CO_2 = f(F, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>EO\ CO_2</math> is engine-out carbon dioxide emissions, in kg/s.</li> <li>• <math>F</math> is commanded fuel mass, in mg per injection.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul>  <p>The figure is a 3D surface plot showing the relationship between engine speed, commanded fuel mass, and engine-out CO2 emissions. The vertical axis represents EO CO2 in kg/s, ranging from 0 to 0.025. The horizontal axes are Engine Speed (RPM) from 0 to 6000 and Commanded Fuel (mg/inj) from 0 to 50. The surface shows that CO2 emissions increase as both engine speed and commanded fuel mass increase, with the highest emissions occurring at high RPM and high fuel injection rates.</p>

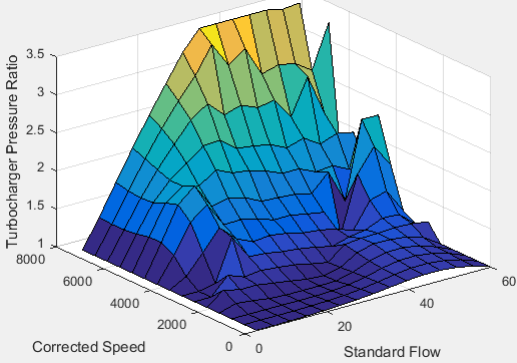
Map	Used For	In	Description
<p>Commanded exhaust gas recirculation (EGR) valve area percent</p>	<p>Commanded exhaust gas recirculation (EGR) valve area percent as a function of commanded torque and engine speed</p>	<p>CI Controller</p>	<p>The commanded exhaust gas recirculation (EGR) valve area percent lookup table is a function of commanded torque and engine speed</p> $EGR_{cmd} = f_{EGRcmd}(Trq_{cmd}, N)$ <p>where:</p> <ul style="list-style-type: none"> <li>• <math>EGR_{cmd}</math> is commanded EGR valve area percent, in percent.</li> <li>• <math>Trq_{cmd}</math> is commanded engine torque, in N·m.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

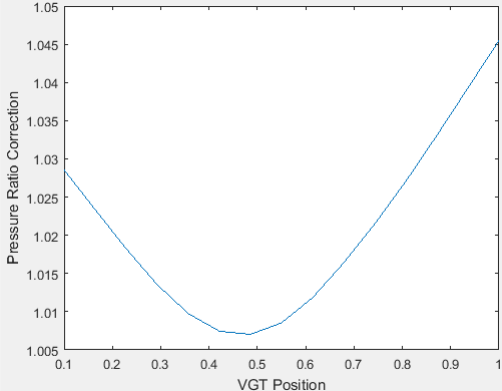
Map	Used For	In	Description
Variable geometry turbocharger (VGT) rack position	Variable geometry turbocharger (VGT) rack position as a function of commanded torque and engine speed	CI Controller	<p>The variable geometry turbocharger (VGT) rack position lookup table is a function of commanded torque and engine speed</p> $RP_{cmd} = f_{RP_{cmd}}(Trq_{cmd}, N)$ <p>where:</p> <ul style="list-style-type: none"> <li>• <math>RP_{cmd}</math> is VGT rack position command, in percent.</li> <li>• <math>Trq_{cmd}</math> is commanded engine torque, in N·m.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul>  <p>The 3D surface plot shows the VGT rack position (z-axis, 0 to 1) as a function of engine speed (y-axis, 0 to 8000 rpm) and commanded torque (x-axis, 0 to 250 N·m). The surface is highest (yellow, ~1.0) at low torque and low engine speed, and lowest (blue, ~0.0) at high torque and high engine speed. The surface shows a complex, non-linear relationship between the three variables.</p>

Map	Used For	In	Description
<p>Commanded total fuel mass per injection</p>	<p>Commanded total fuel mass per injection as a function of torque command and engine speed</p>	<p>CI Controller</p>	<p>The commanded total fuel mass per injection table is a function of the torque command and engine speed</p> $F_{cmd,tot} = f_{F_{cmd,tot}}(Trq_{cmd}, N)$ <p>where:</p> <ul style="list-style-type: none"> <li>• <math>F_{cmd,tot} = F</math> is commanded total fuel mass per injection, in mg per cylinder.</li> <li>• <math>Trq_{cmd}</math> is commanded engine torque, in N·m.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

Map	Used For	In	Description
Main start-of-injection (SOI) timing	SOI timing as a function of commanded fuel mass and engine speed	CI Controller	<p>The main start-of-injection (SOI) timing lookup table is a function of commanded fuel mass and engine speed</p> $MAINSOI = f(F_{cmd,tot}, N)$ <p>where:</p> <ul style="list-style-type: none"> <li>• <i>MAINSOI</i> is the main start-of-injection timing, in degrees crank angle after top dead center (degATDC).</li> <li>• <math>F_{cmd,tot} = F</math> is commanded fuel mass, in mg per injection.</li> <li>• <i>N</i> is engine speed, in rpm.</li> </ul>  <p>The figure is a 3D surface plot showing the relationship between engine speed (N), commanded fuel mass (F), and MAIN SOI Timing. The vertical axis represents MAIN SOI Timing, ranging from -15 to 5. The horizontal axes are N (engine speed) and F (commanded fuel mass). The N axis ranges from 0 to 6000 rpm, and the F axis ranges from 0 to 50 mg per injection. The surface shows that MAIN SOI Timing is generally negative (indicating injection before top dead center) and varies with both engine speed and fuel mass. The surface is colored with a gradient from blue (lower timing) to yellow (higher timing).</p>

Map	Used For	In	Description
Standard exhaust gas recirculation (EGR) mass flow	EGR mass flow as a function of the standard flow pressure ratio and EGR valve flow area	CI Controller	<p>The standard exhaust gas recirculation (EGR) mass flow is a lookup table that is a function of the standard flow pressure ratio and EGR valve flow area</p> $\dot{m}_{egr,std} = f\left(\frac{MAP}{P_{exh,est}}, EGRap\right)$ <p>where:</p> <ul style="list-style-type: none"> <li>• <math>\dot{m}_{egr,std}</math> is the standard EGR valve mass flow, in g/s.</li> <li>• <math>P_{exh,est}</math> is the estimated exhaust back-pressure, in Pa.</li> <li>• <math>MAP</math> is the cycle average intake manifold absolute pressure, in Pa.</li> <li>• <math>EGRap</math> is the measured EGR valve area, in percent.</li> </ul>  <p>The figure is a 3D surface plot showing the relationship between EGR Valve Flow (z-axis, 0 to 100), EGR Valve Area Percent (x-axis, 0 to 100), and EGR Valve Pressure Ratio (y-axis, 0 to 1). The surface is a smooth, downward-sloping plane that is highest at low pressure ratios and high valve areas, and lowest at high pressure ratios and low valve areas. The color gradient transitions from yellow at the top to dark blue at the bottom.</p>

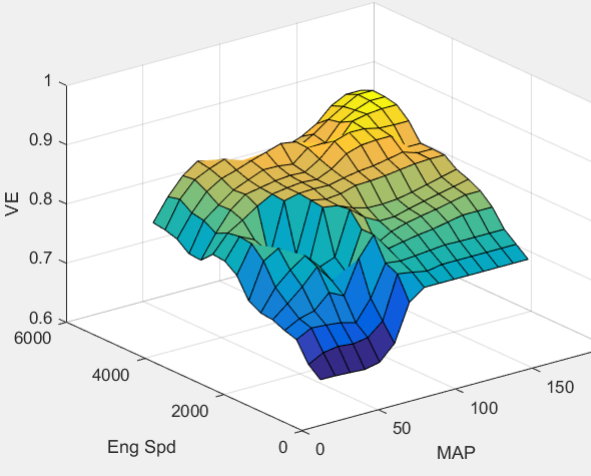
Map	Used For	In	Description
Turbocharger pressure ratio	Turbocharger pressure ratio as a function of the standard air mass flow and corrected turbocharger speed	CI Controller	<p>The turbocharger pressure ratio, corrected for variable geometry turbocharger (VGT) speed, is a lookup table that is a function of the standard air mass flow and corrected turbocharger speed, <math>Pr_{turbo} = f(\dot{m}_{airstd}, N_{vgtcrr})</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>Pr_{turbo}</math> is the turbocharger pressure ratio, corrected for VGT speed.</li> <li>• <math>\dot{m}_{airstd}</math> is the standard air mass flow, in g/s.</li> <li>• <math>N_{vgtcrr}</math> is the corrected turbocharger speed, in rpm/K<sup>(1/2)</sup>.</li> </ul>  <p>The figure is a 3D surface plot showing the Turbocharger Pressure Ratio as a function of Corrected Speed and Standard Flow. The vertical axis (z-axis) is labeled 'Turbocharger Pressure Ratio' and ranges from 1 to 3.5. The horizontal axis (x-axis) is labeled 'Corrected Speed' and ranges from 0 to 8000. The depth axis (y-axis) is labeled 'Standard Flow' and ranges from 0 to 60. The surface shows a peak pressure ratio of approximately 3.5 at a corrected speed of about 6000 and a standard flow of about 20. The pressure ratio decreases as the corrected speed decreases and the standard flow increases.</p>

Map	Used For	In	Description
Turbocharger pressure ratio correction	Turbocharger pressure ratio correction as a function of the rack position	CI Controller	<p>The variable geometry turbocharger pressure ratio correction is a function of the rack position, <math>Pr_{vgtcorr} = f(VGT_{pos})</math>, where:</p> <ul style="list-style-type: none"> <li><math>Pr_{vgtcorr}</math> is the turbocharger pressure ratio correction.</li> <li><math>VGT_{pos}</math> is the variable geometry turbocharger (VGT) rack position.</li> </ul> 

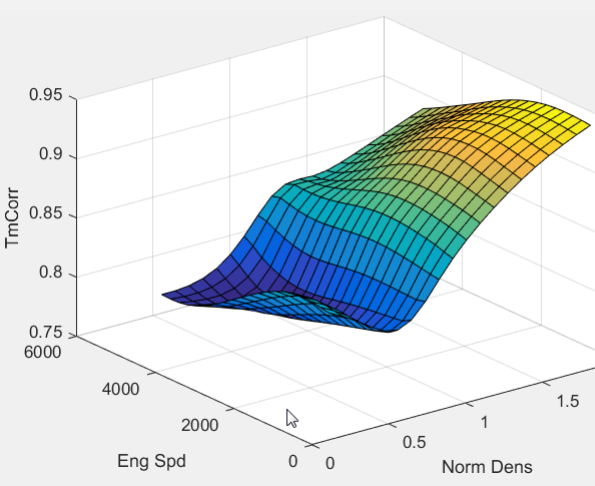
## Calibration Maps in Spark-Ignition (SI) Blocks

In the engine models, the Powertrain Blockset blocks implement these calibration maps.

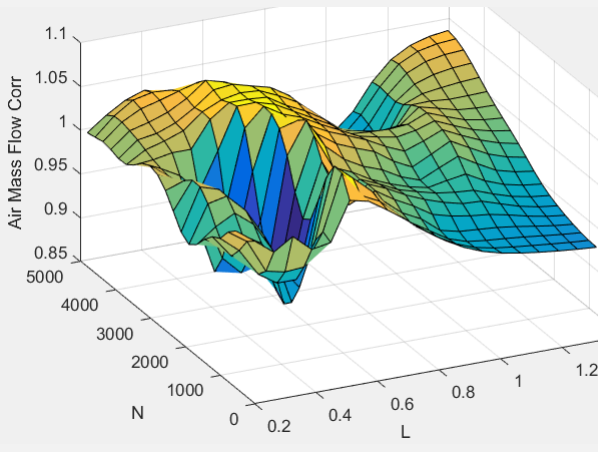


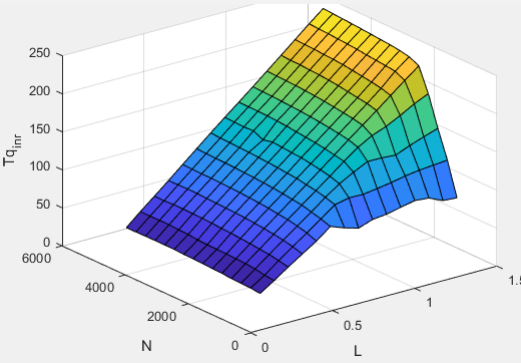
Map	Used for	In	Description
<p>Engine volumetric efficiency</p>	<p>“SI Engine Speed-Density Air Mass Flow Model” on page 2-14</p>	<p>SI Core Engine SI Controller</p>	<p>The engine volumetric efficiency lookup table, <math>f_{\eta_v}</math>, is a function of intake manifold absolute pressure and engine speed</p> $\eta_v = f_{\eta_v}(MAP, N)$ <p>where:</p> <ul style="list-style-type: none"> <li><math>\eta_v</math> is engine volumetric efficiency, dimensionless.</li> <li>MAP is intake manifold absolute pressure, in KPa.</li> <li>N is engine speed, in rpm.</li> </ul> 

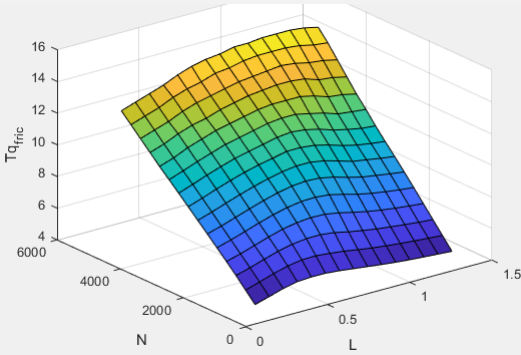
Map	Used for	In	Description																																																						
Cylinder volume at intake valve close table (IVC)	"SI Engine Dual-Independent Cam Phaser Air Mass Flow Model" on page 2-5	SI Core Engine SI Controller	<p>The cylinder volume at intake valve close table (IVC), <math>f_{V_{ivc}}</math> is a function of the intake cam phaser angle</p> $V_{IVC} = f_{V_{ivc}}(\varphi_{ICP})$ <p>where:</p> <ul style="list-style-type: none"> <li><math>V_{IVC}</math> is cylinder volume at IVC, in L.</li> <li><math>\varphi_{ICP}</math> is intake cam phaser angle, in crank advance degrees.</li> </ul> <table border="1"> <caption>Data points for the IVC Volume vs. ICP graph</caption> <thead> <tr> <th>ICP (degCrkAdv)</th> <th>Vivc (L)</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.377</td></tr> <tr><td>2</td><td>0.382</td></tr> <tr><td>4</td><td>0.386</td></tr> <tr><td>6</td><td>0.390</td></tr> <tr><td>8</td><td>0.394</td></tr> <tr><td>10</td><td>0.398</td></tr> <tr><td>12</td><td>0.402</td></tr> <tr><td>14</td><td>0.405</td></tr> <tr><td>16</td><td>0.408</td></tr> <tr><td>18</td><td>0.411</td></tr> <tr><td>20</td><td>0.413</td></tr> <tr><td>22</td><td>0.415</td></tr> <tr><td>24</td><td>0.417</td></tr> <tr><td>26</td><td>0.418</td></tr> <tr><td>28</td><td>0.419</td></tr> <tr><td>30</td><td>0.420</td></tr> <tr><td>32</td><td>0.421</td></tr> <tr><td>34</td><td>0.421</td></tr> <tr><td>36</td><td>0.421</td></tr> <tr><td>38</td><td>0.421</td></tr> <tr><td>40</td><td>0.421</td></tr> <tr><td>42</td><td>0.421</td></tr> <tr><td>44</td><td>0.421</td></tr> <tr><td>46</td><td>0.421</td></tr> <tr><td>48</td><td>0.421</td></tr> <tr><td>50</td><td>0.421</td></tr> </tbody> </table>	ICP (degCrkAdv)	Vivc (L)	0	0.377	2	0.382	4	0.386	6	0.390	8	0.394	10	0.398	12	0.402	14	0.405	16	0.408	18	0.411	20	0.413	22	0.415	24	0.417	26	0.418	28	0.419	30	0.420	32	0.421	34	0.421	36	0.421	38	0.421	40	0.421	42	0.421	44	0.421	46	0.421	48	0.421	50	0.421
ICP (degCrkAdv)	Vivc (L)																																																								
0	0.377																																																								
2	0.382																																																								
4	0.386																																																								
6	0.390																																																								
8	0.394																																																								
10	0.398																																																								
12	0.402																																																								
14	0.405																																																								
16	0.408																																																								
18	0.411																																																								
20	0.413																																																								
22	0.415																																																								
24	0.417																																																								
26	0.418																																																								
28	0.419																																																								
30	0.420																																																								
32	0.421																																																								
34	0.421																																																								
36	0.421																																																								
38	0.421																																																								
40	0.421																																																								
42	0.421																																																								
44	0.421																																																								
46	0.421																																																								
48	0.421																																																								
50	0.421																																																								

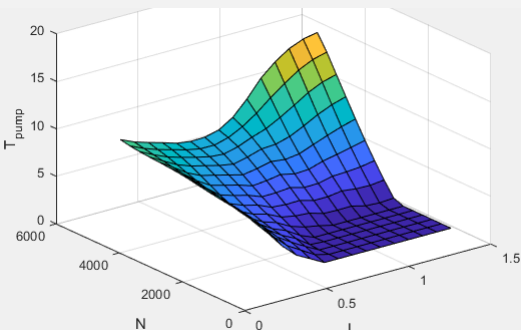
Map	Used for	In	Description
Trapped mass correction	"SI Engine Dual-Independent Cam Phaser Air Mass Flow Model" on page 2-5	SI Core Engine SI Controller	<p>The trapped mass correction factor table, <math>f_{TM_{corr}}</math>, is a function of the normalized density and engine speed</p> $TM_{corr} = f_{TM_{corr}}(\rho_{norm}, N)$ <p>where:</p> <ul style="list-style-type: none"> <li><math>TM_{corr}</math>, is trapped mass correction multiplier, dimensionless.</li> <li><math>\rho_{norm}</math> is normalized density, dimensionless.</li> <li><math>N</math> is engine speed, in rpm.</li> </ul> 

Map	Used for	In	Description
Air mass flow at cam phaser angles	“SI Engine Dual-Independent Cam Phaser Air Mass Flow Model” on page 2-5	SI Core Engine SI Controller	<p>The phaser intake mass flow model lookup table is a function of exhaust cam phaser angles and trapped air mass flow</p> $\dot{m}_{intkideal} = f_{intkideal}(\varphi_{ECP}, TM_{flow})$ <p>where:</p> <ul style="list-style-type: none"> <li><math>\dot{m}_{intkideal}</math> is engine intake port mass flow at arbitrary cam phaser angles, in g/s.</li> <li><math>\varphi_{ECP}</math> is exhaust cam phaser angle, in degrees crank retard.</li> <li><math>TM_{flow}</math> is flow rate equivalent to corrected trapped mass at the current engine speed, in g/s.</li> </ul>

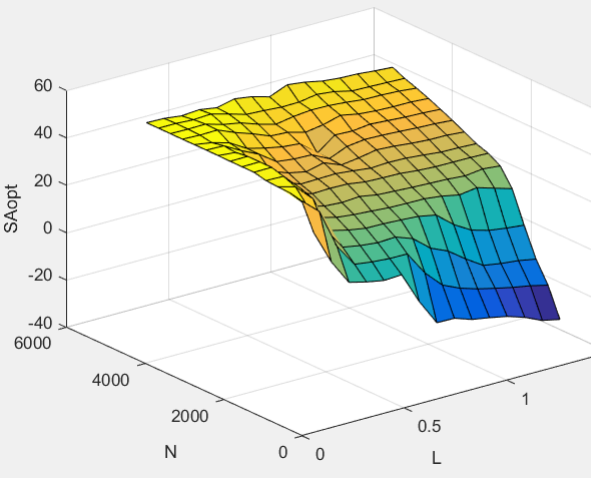
Map	Used for	In	Description
Air mass flow correction	"SI Engine Dual-Independent Cam Phaser Air Mass Flow Model" on page 2-5	SI Core Engine SI Controller	<p>The intake air mass flow correction lookup table, <math>f_{aircorr}</math>, is a function of ideal load and engine speed</p> $\dot{m}_{air} = \dot{m}_{inthideal} f_{aircorr}(L_{ideal}, N)$ <p>where:</p> <ul style="list-style-type: none"> <li>• <math>L_{ideal}</math> is engine load (normalized cylinder air mass) at arbitrary cam phaser angles, uncorrected for final steady-state cam phaser angles, dimensionless.</li> <li>• <math>N</math> is engine speed, in rpm.</li> <li>• <math>\dot{m}_{air}</math> is engine intake air mass flow final correction at steady-state cam phaser angles, in g/s.</li> <li>• <math>\dot{m}_{inthideal}</math> is engine intake port mass flow at arbitrary cam phaser angles, in g/s.</li> </ul> 

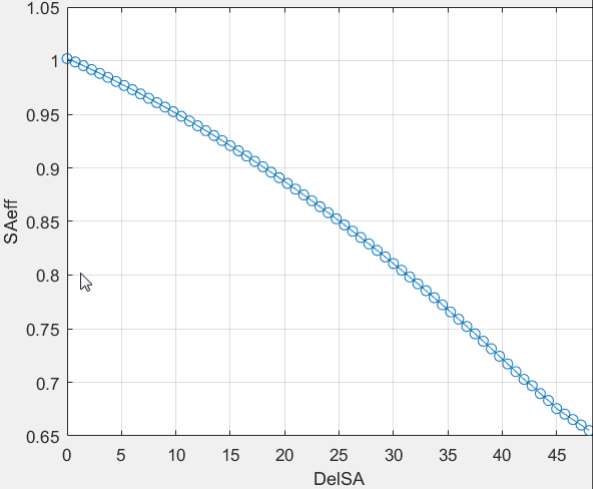
Map	Used for	In	Description
Inner torque	"SI Engine Torque Structure Model" on page 2-17	SI Core Engine  SI Controller	<p>The inner torque lookup table, <math>f_{Tq_{inr}}</math>, is a function of engine speed and engine load,</p> $Tq_{inr} = f_{Tq_{inr}}(L, N),$ where: <ul style="list-style-type: none"> <li>• <math>Tq_{inr}</math> is inner torque based on gross indicated mean effective pressure, in N·m.</li> <li>• <math>L</math> is engine load at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles, dimensionless.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

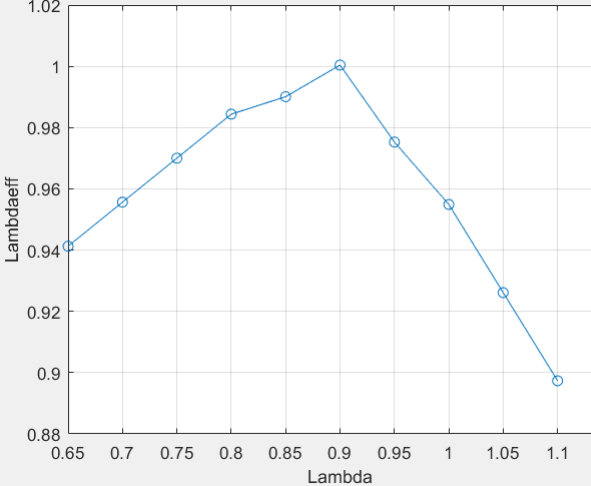
Map	Used for	In	Description
Friction torque	"SI Engine Torque Structure Model" on page 2-17	SI Core Engine  SI Controller	<p>The friction torque lookup table, <math>f_{T_{fric}}</math>, is a function of engine speed and engine load,</p> $T_{fric} = f_{T_{fric}}(L, N), \text{ where:}$ <ul style="list-style-type: none"> <li><math>T_{fric}</math> is friction torque offset to inner torque, in N·m.</li> <li><math>L</math> is engine load at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles, dimensionless.</li> <li><math>N</math> is engine speed, in rpm.</li> </ul> 

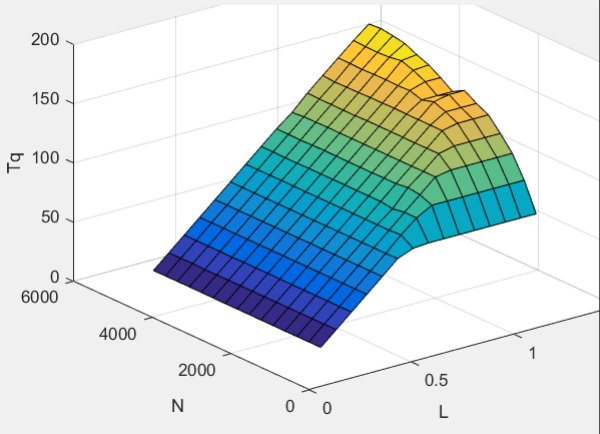
Map	Used for	In	Description
Pumping torque	"SI Engine Torque Structure Model" on page 2-17	SI Core Engine  SI Controller	<p>The pumping torque lookup table, <math>f_{T_{pump}}</math>, is a function of engine speed and injected fuel mass, <math>T_{pump} = f_{T_{pump}}(L, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>T_{pump}</math> is pumping torque, in N·m.</li> <li>• <math>L</math> is engine load, as a normalized cylinder air mass, dimensionless.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

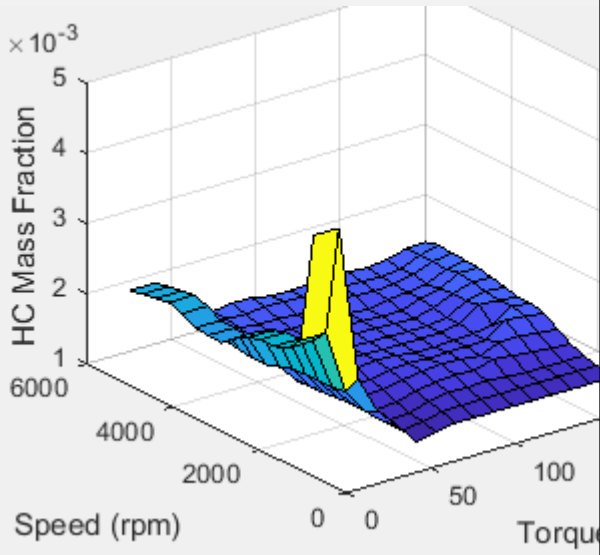


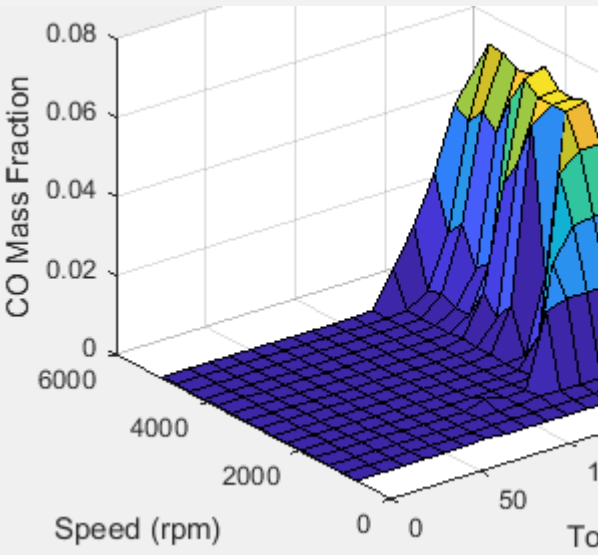
Map	Used for	In	Description
Optimal spark advance	"SI Engine Torque Structure Model" on page 2-17	SI Core Engine  SI Controller	<p>The optimal spark lookup table, <math>f_{SA_{opt}}</math>, is a function of engine speed and engine load,</p> $SA_{opt} = f_{SA_{opt}}(L, N), \text{ where:}$ <ul style="list-style-type: none"> <li>• <math>SA_{opt}</math> is optimal spark advance timing for maximum inner torque at stoichiometric air-fuel ratio (AFR), in deg.</li> <li>• <math>L</math> is engine load at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles, dimensionless.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

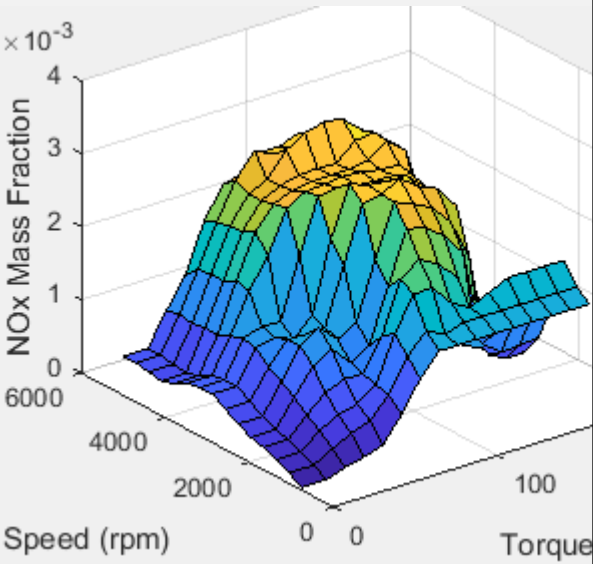
Map	Used for	In	Description																								
Spark efficiency	"SI Engine Torque Structure Model" on page 2-17	SI Core Engine  SI Controller	<p>The spark efficiency lookup table, <math>f_{M_{sa}}</math>, is a function of the spark retard from optimal</p> $M_{sa} = f_{M_{sa}}(\Delta SA)$ $\Delta SA = SA_{opt} - SA$ <p>where:</p> <ul style="list-style-type: none"> <li>• <math>M_{sa}</math> is the spark retard efficiency multiplier, dimensionless.</li> <li>• <math>\Delta SA</math> is the spark retard timing distance from optimal spark advance, in deg.</li> </ul>  <table border="1"> <caption>Approximate data points from the SAeff vs DelSA graph</caption> <thead> <tr> <th>DelSA (deg)</th> <th>SAeff</th> </tr> </thead> <tbody> <tr><td>0</td><td>1.00</td></tr> <tr><td>5</td><td>0.97</td></tr> <tr><td>10</td><td>0.94</td></tr> <tr><td>15</td><td>0.91</td></tr> <tr><td>20</td><td>0.88</td></tr> <tr><td>25</td><td>0.85</td></tr> <tr><td>30</td><td>0.82</td></tr> <tr><td>35</td><td>0.79</td></tr> <tr><td>40</td><td>0.76</td></tr> <tr><td>45</td><td>0.73</td></tr> <tr><td>48</td><td>0.65</td></tr> </tbody> </table>	DelSA (deg)	SAeff	0	1.00	5	0.97	10	0.94	15	0.91	20	0.88	25	0.85	30	0.82	35	0.79	40	0.76	45	0.73	48	0.65
DelSA (deg)	SAeff																										
0	1.00																										
5	0.97																										
10	0.94																										
15	0.91																										
20	0.88																										
25	0.85																										
30	0.82																										
35	0.79																										
40	0.76																										
45	0.73																										
48	0.65																										

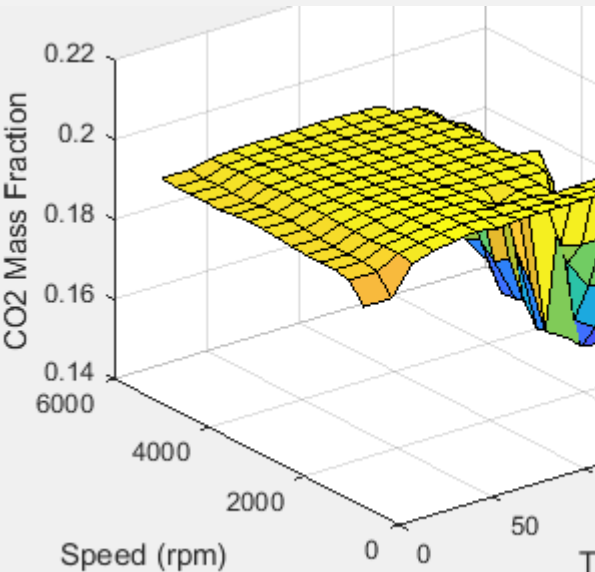
Map	Used for	In	Description
Lambda efficiency	"SI Engine Torque Structure Model" on page 2-17	SI Core Engine  SI Controller	<p>The lambda efficiency lookup table, <math>f_{M\lambda}</math>, is a function of lambda, <math>M_\lambda = f_{M\lambda}(\lambda)</math>, where:</p> <ul style="list-style-type: none"> <li><math>M_\lambda</math> is the lambda multiplier on inner torque to account for the air-fuel ratio (AFR) effect, dimensionless.</li> <li><math>\lambda</math> is lambda, AFR normalized to stoichiometric fuel AFR, dimensionless.</li> </ul> 

Map	Used for	In	Description
Simple torque	"SI Engine Simple Torque Model" on page 2-26	SI Core Engine  SI Controller	<p>For the simple torque lookup table model, the SI engine uses a lookup table map that is a function of engine speed and load,</p> $T_{brake} = f_{TnL}(L, N), \text{ where:}$ <ul style="list-style-type: none"> <li>• <math>T_{brake}</math> is engine brake torque after accounting for spark advance, AFR, and friction effects, in N·m.</li> <li>• <math>L</math> is engine load, as a normalized cylinder air mass, dimensionless.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

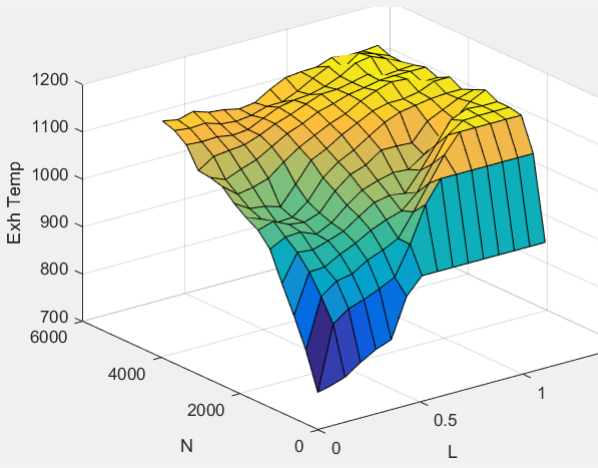
Map	Used for	In	Description
Hydrocarbon (HC) mass fraction	HC emissions	SI Core Engine	<p>The SI Core Engine HC emission mass fraction lookup table is a function of engine torque and engine speed, <math>HC\ Mass\ Fraction = f(Speed, Torque)</math>, where:</p> <ul style="list-style-type: none"> <li>• <i>HC Mass Fraction</i> is the HC emission mass fraction, dimensionless.</li> <li>• <i>Speed</i> is engine speed, in rpm.</li> <li>• <i>Torque</i> is engine torque, in N·m.</li> </ul> 

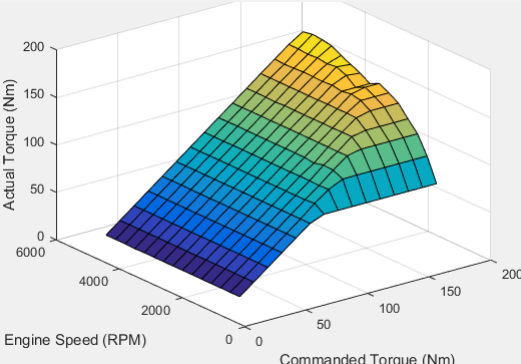
Map	Used for	In	Description
Carbon monoxide (CO) mass fraction	CO emissions	SI Core Engine	<p>The SI Core Engine CO emission mass fraction lookup table is a function of engine torque and engine speed, <math>CO\ Mass\ Fraction = f(Speed, Torque)</math>, where:</p> <ul style="list-style-type: none"> <li>• <i>CO Mass Fraction</i> is the CO emission mass fraction, dimensionless.</li> <li>• <i>Speed</i> is engine speed, in rpm.</li> <li>• <i>Torque</i> is engine torque, in N·m.</li> </ul> 

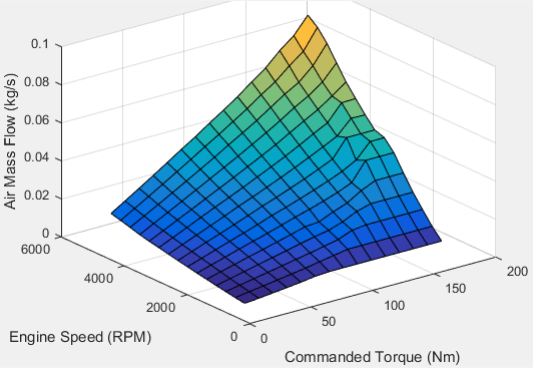
Map	Used for	In	Description
Nitric oxide and nitrogen dioxide (NOx) mass fraction	NOx emissions	SI Core Engine	<p>The SI Core Engine NOx emission mass fraction lookup table is a function of engine torque and engine speed, <math>NOx\ Mass\ Fraction = f(Speed, Torque)</math>, where:</p> <ul style="list-style-type: none"> <li>• <i>NOx Mass Fraction</i> is the NOx emission mass fraction, dimensionless.</li> <li>• <i>Speed</i> is engine speed, in rpm.</li> <li>• <i>Torque</i> is engine torque, in N·m.</li> </ul> 

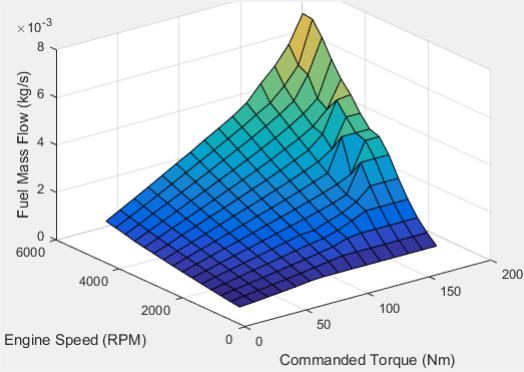
Map	Used for	In	Description
Carbon dioxide (CO <sub>2</sub> ) mass fraction	CO <sub>2</sub> emissions	SI Core Engine	<p>The SI Core Engine CO<sub>2</sub> emission mass fraction lookup table is a function of engine torque and engine speed, <math>CO_2 \text{ Mass Fraction} = f(\text{Speed}, \text{Torque})</math>, where:</p> <ul style="list-style-type: none"> <li>• <i>CO<sub>2</sub> Mass Fraction</i> is the CO<sub>2</sub> emission mass fraction, dimensionless.</li> <li>• <i>Speed</i> is engine speed, in rpm.</li> <li>• <i>Torque</i> is engine torque, in N·m.</li> </ul> 

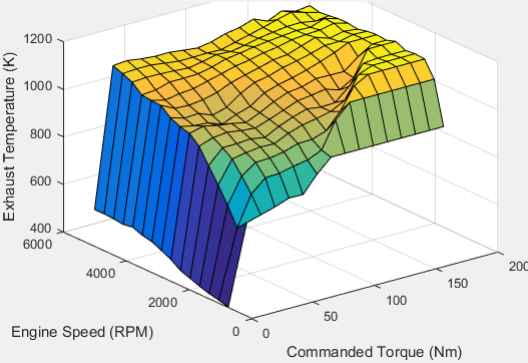


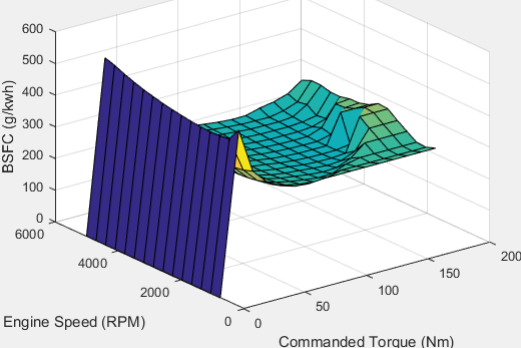
Map	Used for	In	Description
Exhaust temperature	Engine exhaust calculation as a function of engine speed and load	SI Core Engine SI Controller	<p>The exhaust temperature lookup table, <math>f_{Texh}</math>, is a function of engine load and engine speed</p> $T_{exh} = f_{Texh}(L, N)$ <p>where:</p> <ul style="list-style-type: none"> <li>• <math>T_{exh}</math> is engine exhaust temperature, in K.</li> <li>• <math>L</math> is normalized cylinder air mass or engine load, dimensionless.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul>  <p>The 3D surface plot shows Exhaust Temp (K) on the vertical axis, ranging from 600 to 1200. The horizontal axes are Engine Speed (N, rpm) ranging from 0 to 4000 and Normalized Cylinder Air Mass (L) ranging from 0 to 1.5. The surface shows that exhaust temperature increases with both engine speed and engine load, with the highest temperatures occurring at high speeds and high loads.</p>

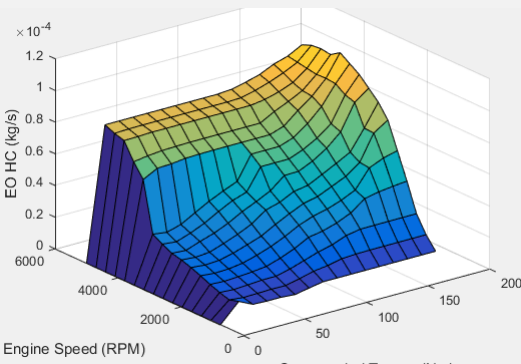
Map	Used for	In	Description
Engine torque	Engine brake torque as a function of commanded torque and engine speed	Mapped SI Engine	<p>The engine torque lookup table is a function of commanded engine torque and engine speed, <math>T = f(T_{cmd}, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>T</math> is engine torque, in N·m.</li> <li>• <math>T_{cmd}</math> is commanded engine torque, in N·m.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

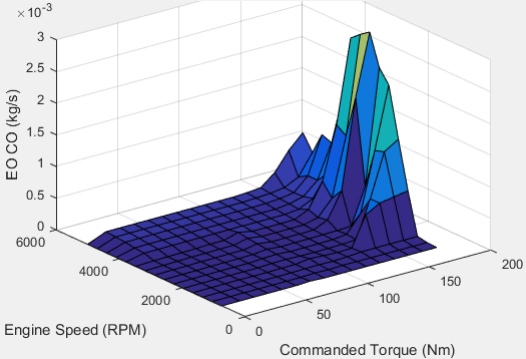
Map	Used for	In	Description
Engine air mass flow	Engine air mass flow as a function of commanded torque and engine speed	Mapped SI Engine	<p>The engine air mass flow lookup table is a function of commanded engine torque and engine speed, <math>\dot{m}_{intk} = f(T_{cmd}, N)</math>, where:</p> <ul style="list-style-type: none"> <li><math>\dot{m}_{intk}</math> is engine air mass flow, in kg/s.</li> <li><math>T_{cmd}</math> is commanded engine torque, in N·m.</li> <li><math>N</math> is engine speed, in rpm.</li> </ul>  <p>The 3D surface plot illustrates the relationship between engine speed, commanded torque, and air mass flow. The vertical axis represents Air Mass Flow in kg/s, ranging from 0 to 0.1. The horizontal axes are Engine Speed in RPM (0 to 6000) and Commanded Torque in Nm (0 to 200). The surface shows that air mass flow increases significantly with engine speed, particularly at low torque levels, reaching its maximum value of approximately 0.1 kg/s at 6000 RPM and 50 Nm.</p>

Map	Used for	In	Description
Engine fuel flow	Engine fuel flow as a function of commanded torque mass and engine speed	Mapped SI Engine	<p>The engine fuel mass flow lookup table is a function of commanded engine torque and engine speed, <math>MassFlow = f(T_{cmd}, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>MassFlow</math> is engine fuel mass flow, in kg/s.</li> <li>• <math>T_{cmd}</math> is commanded engine torque, in N·m.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul>  <p>The figure is a 3D surface plot showing the relationship between Engine Speed (RPM) on the x-axis, Commanded Torque (Nm) on the y-axis, and Fuel Mass Flow (kg/s) on the z-axis. The x-axis ranges from 0 to 6000 RPM, the y-axis from 0 to 200 Nm, and the z-axis from 0 to 8 x 10<sup>-3</sup> kg/s. The surface shows that fuel mass flow increases significantly as engine speed decreases and commanded torque increases, reaching its maximum value at approximately 6000 RPM and 200 Nm.</p>

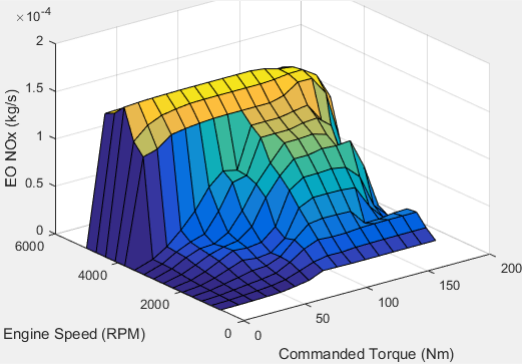
Map	Used for	In	Description
<p>Engine exhaust temperature</p>	<p>Engine exhaust temperature as a function of commanded torque and engine speed</p>	<p>Mapped SI Engine</p>	<p>The engine exhaust temperature lookup table is a function of commanded engine torque and engine speed, <math>T_{exh} = f(T_{cmd}, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>T_{exh}</math> is exhaust temperature, in K.</li> <li>• <math>T_{cmd}</math> is commanded engine torque, in N·m.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

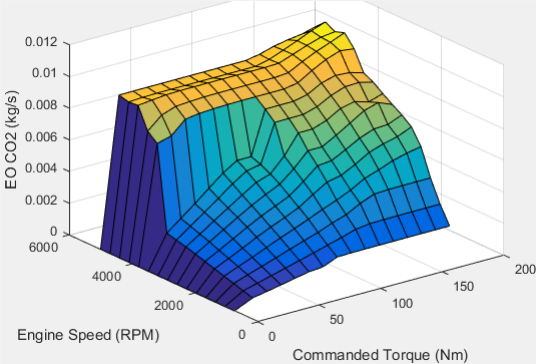
Map	Used for	In	Description
Brake-specific fuel consumption (BSFC) efficiency	Brake-specific fuel consumption (BSFC) as a function of commanded torque and engine speed	Mapped SI Engine	<p>The brake-specific fuel consumption (BSFC) efficiency is a function of commanded engine torque and engine speed, <math>BSFC = f(T_{cmd}, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>BSFC</math> is BSFC, in g/kWh.</li> <li>• <math>T_{cmd}</math> is commanded engine torque, in N·m.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

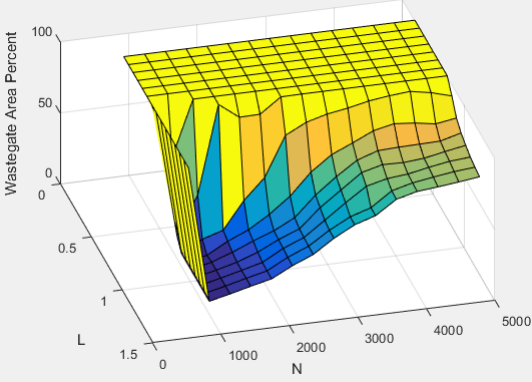
Map	Used for	In	Description
Engine-out (EO) hydrocarbon emissions	EO hydrocarbon emissions as a function of commanded torque and engine speed	Mapped SI Engine	<p>The engine-out hydrocarbon emissions are a function of commanded engine torque and engine speed, <math>EO\ HC = f(T_{cmd}, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>EO\ HC</math> is engine-out hydrocarbon emissions, in kg/s.</li> <li>• <math>T_{cmd}</math> is commanded engine torque, in N·m.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

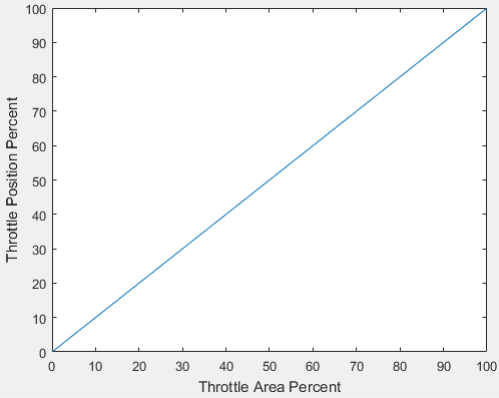
Map	Used for	In	Description
<p>Engine-out (EO) carbon monoxide emissions</p>	<p>EO carbon monoxide emissions as a function of commanded torque and engine speed</p>	<p>Mapped SI Engine</p>	<p>The engine-out carbon monoxide emissions are a function of commanded engine torque and engine speed, <math>EO\ CO = f(T_{cmd}, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>EO\ CO</math> is engine-out carbon monoxide emissions, in kg/s.</li> <li>• <math>T_{cmd}</math> is commanded engine torque, in N·m.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

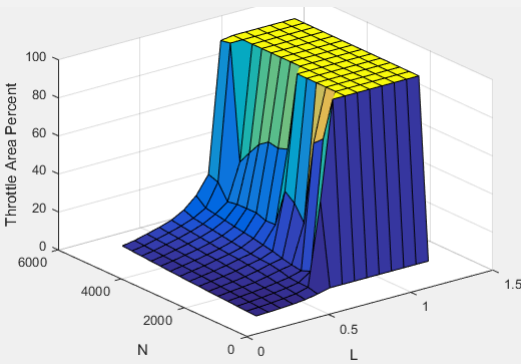


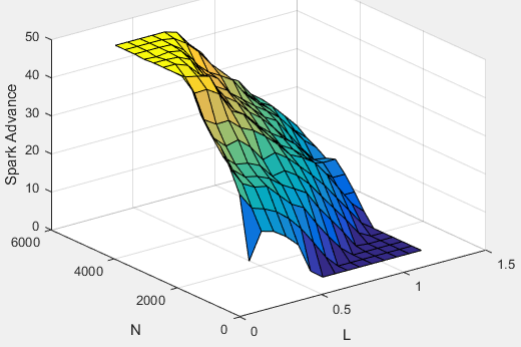
Map	Used for	In	Description
Engine-out (EO) nitric oxide and nitrogen dioxide emissions	EO nitric oxide and nitrogen dioxide emissions as a function of commanded torque and engine speed	Mapped SI Engine	<p>The engine-out nitric oxide and nitrogen dioxide emissions are a function of commanded engine torque and engine speed, <math>EO\ NO_x = f(T_{cmd}, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>EO\ NO_x</math> is engine-out nitric oxide and nitrogen dioxide emissions, in kg/s.</li> <li>• <math>T_{cmd}</math> is commanded engine torque, in N·m.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul>  <p>The figure is a 3D surface plot showing the relationship between engine speed, commanded torque, and engine-out nitric oxide (EO NOx) emissions. The vertical axis represents EO NOx in kg/s, scaled by 10<sup>-4</sup>, with values from 0 to 2. The horizontal axes are Engine Speed (RPM) from 0 to 6000 and Commanded Torque (Nm) from 0 to 200. The surface shows that emissions are highest at high engine speeds and high torques, reaching a peak of approximately 1.8 x 10<sup>-4</sup> kg/s at 6000 RPM and 150 Nm. Emissions decrease significantly as engine speed drops below 2000 RPM and as torque drops below 50 Nm.</p>

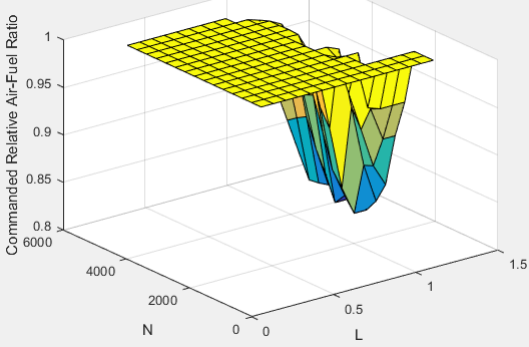
Map	Used for	In	Description
<p>Engine-out (EO) carbon dioxide emissions</p>	<p>EO carbon dioxide emissions as a function of commanded torque and engine speed</p>	<p>Mapped SI Engine</p>	<p>The engine-out carbon dioxide emissions are a function of commanded engine torque and engine speed, <math>EO\ CO_2 = f(T_{cmd}, N)</math>, where:</p> <ul style="list-style-type: none"> <li>• <math>EO\ CO_2</math> is engine-out carbon dioxide emissions, in kg/s.</li> <li>• <math>T_{cmd}</math> is commanded engine torque, in N·m.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul>  <p>The figure is a 3D surface plot showing the relationship between engine speed, commanded torque, and engine-out CO2 emissions. The vertical axis represents EO CO2 in kg/s, ranging from 0 to 0.012. The horizontal axes are Engine Speed (RPM) from 0 to 4000 and Commanded Torque (Nm) from 0 to 200. The surface shows that emissions increase significantly with both engine speed and torque, with the highest values occurring at high speeds and high torque levels.</p>

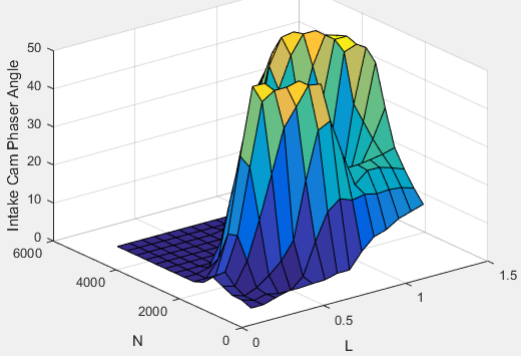
Map	Used for	In	Description
Wastegate area percent command	Wastegate area percent command as a function of the commanded engine load and engine speed	SI Controller	<p>The wastegate area percent command lookup table, <math>f_{WAPcmd}</math>, is a function of the commanded engine load and engine speed</p> $WAP_{cmd} = f_{WAPcmd}(L_{cmd}, N)$ <p>where:</p> <ul style="list-style-type: none"> <li>• <math>WAP_{cmd}</math> is wastegate area percentage command, in percent.</li> <li>• <math>L_{cmd}=L</math> is commanded engine load, dimensionless.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

Map	Used for	In	Description
Throttle position percent command	Throttle position percent command as a function of the throttle area percentage command	SI Controller	<p>The throttle position percent command lookup table, <math>f_{TPP_{cmd}}</math>, is a function of the throttle area percentage command</p> $TPP_{cmd} = f_{TPP_{cmd}}(TAP_{cmd})$ <p>where:</p> <ul style="list-style-type: none"> <li>• <math>TPP_{cmd}</math> is throttle position percentage command, in percent.</li> <li>• <math>TAP_{cmd}</math> is throttle area percentage command, in percent.</li> </ul> 

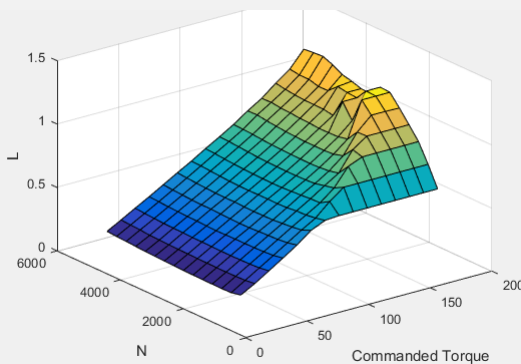
Map	Used for	In	Description
Throttle area percent command	Throttle area percent command as a function of commanded load and engine speed	SI Controller	<p>The throttle area percent command lookup table, <math>f_{TAPcmd}</math>, is a function of commanded load and engine speed</p> $TAP_{cmd} = f_{TAPcmd}(L_{cmd}, N)$ <p>where:</p> <ul style="list-style-type: none"> <li>• <math>TAP_{cmd}</math> is throttle area percentage command, in percent.</li> <li>• <math>L_{cmd}=L</math> is commanded engine load, dimensionless.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

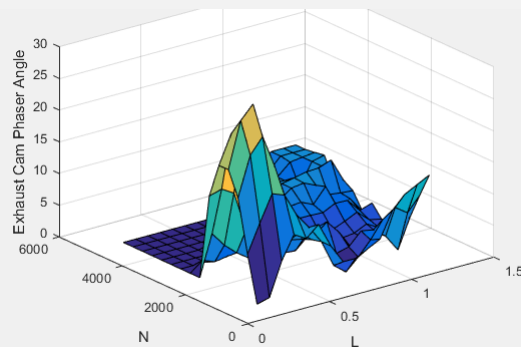
Map	Used for	In	Description
Spark advance	Spark advance as a function of estimated load and engine speed	SI Controller	<p>The spark advance lookup table is a function of estimated load and engine speed.</p> $SA = f_{SA}(L_{est}, N)$ <p>where:</p> <ul style="list-style-type: none"> <li>• SA is spark advance, in crank advance degrees.</li> <li>• <math>L_{est}=L</math> is estimated engine load, dimensionless.</li> <li>• N is engine speed, in rpm.</li> </ul> 

Map	Used for	In	Description
Commanded lambda	Commanded lambda as a function of estimated engine load and measured engine speed	SI Controller	<p>The commanded lambda, <math>\lambda_{cmd}</math>, lookup table is a function of estimated engine load and measured engine speed</p> $\lambda_{cmd} = f_{\lambda_{cmd}}(L_{est}, N)$ <p>where:</p> <ul style="list-style-type: none"> <li><math>\lambda_{cmd}</math> is commanded relative AFR, dimensionless.</li> <li><math>L_{est}=L</math> is estimated engine load, dimensionless.</li> <li><math>N</math> is engine speed, in rpm.</li> </ul> 

Map	Used for	In	Description
Intake cam phaser angle command	Intake cam phaser angle command as a function of the engine load and engine speed	SI Controller	<p>The intake cam phaser angle command lookup table, <math>f_{ICPCMD}</math>, is a function of the engine load and engine speed</p> $\varphi_{ICPCMD} = f_{ICPCMD}(L_{est}, N)$ <p>where:</p> <ul style="list-style-type: none"> <li>• <math>\varphi_{ICPCMD}</math> is commanded intake cam phaser angle, in degrees crank advance.</li> <li>• <math>L_{est}=L</math> is estimated engine load, dimensionless.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 



Map	Used for	In	Description
<p>Commanded engine load</p>	<p>Commanded engine load as a function of the commanded torque and engine speed</p>	<p>SI Controller</p>	<p>The commanded engine load lookup table, <math>f_{Lcmd}</math>, is a function of the commanded torque and engine speed</p> $L_{cmd} = f_{Lcmd}(T_{cmd}, N)$ <p>where:</p> <ul style="list-style-type: none"> <li>• <math>L_{cmd}=L</math> is commanded engine load, dimensionless.</li> <li>• <math>T_{cmd}</math> is commanded torque, in N·m.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

Map	Used for	In	Description
Exhaust cam phaser angle	Exhaust cam phaser angle as a function of the engine load and engine speed	SI Controller	<p>The exhaust cam phaser angle command lookup table, <math>f_{ECPCMD}</math>, is a function of the engine load and engine speed</p> $\varphi_{ECPCMD} = f_{ECPCMD}(L_{est}, N)$ <p>where:</p> <ul style="list-style-type: none"> <li>• <math>\varphi_{ECPCMD}</math> is commanded exhaust cam phaser angle, in degrees crank retard.</li> <li>• <math>L_{est}=L</math> is estimated engine load, dimensionless.</li> <li>• <math>N</math> is engine speed, in rpm.</li> </ul> 

## See Also

CI Controller | CI Core Engine | Mapped SI Engine | Mapped CI Engine | SI Controller | SI Core Engine

## External Websites

- [Virtual Engine Calibration: Making Engine Calibration Part of the Engine Hardware Design Process](#)

# Reference Applications

---

## Internal Combustion Engine Reference Application Projects

Use these reference application projects as a starting point for your own vehicle and internal combustion engine models.

Objective	Model	Reference
Design tradeoff analysis and component sizing, control parameter optimization, or hardware-in-the-loop (HIL) testing.	Full conventional vehicle with spark-ignition (SI) or combustion-ignition (CI)	“Explore the Conventional Vehicle Reference Application” on page 3-4
Engine and controller calibration, validation, and optimization before integration with the vehicle model.	CI engine plant and controller	“Explore the CI Engine Dynamometer Reference Application” on page 3-10
	SI engine plant and controller	“Explore the SI Engine Dynamometer Reference Application” on page 3-15

### See Also

#### Related Examples

- “Resize the CI Engine” on page 3-54
- “Resize the SI Engine” on page 3-63

#### More About

- “Hybrid and Electric Vehicle Reference Application Projects” on page 3-3
- “Internal Combustion Mapped and Dynamic Engine Models” on page 3-83
- “CI Engine Project Template” on page 4-2
- “SI Engine Project Template” on page 4-5

## Hybrid and Electric Vehicle Reference Application Projects

Use these reference applications as a starting point for your own vehicle hybrid and electric vehicle models.

Objective	Model	Reference
Design tradeoff analysis and component sizing, control parameter optimization, or hardware-in-the-loop (HIL) testing.	Hybrid electric vehicle (HEV) — Multimode	“Explore the Hybrid Electric Vehicle Multimode Reference Application” on page 3-20
	HEV — Input power-split	“Explore the Hybrid Electric Vehicle Input Power-Split Reference Application” on page 3-34
	HEV — P2	“Explore the Hybrid Electric Vehicle P2 Reference Application” on page 3-44
	Electric vehicle	“Explore the Electric Vehicle Reference Application” on page 3-28

### See Also

#### More About

- “Internal Combustion Engine Reference Application Projects” on page 3-2
- “CI Engine Project Template” on page 4-2
- “SI Engine Project Template” on page 4-5

## Explore the Conventional Vehicle Reference Application

The conventional vehicle reference application represents a full vehicle model with an internal combustion engine, transmission, and associated powertrain control algorithms. Use the reference application for powertrain matching analysis and component selection, control and diagnostic algorithm design, and hardware-in-the-loop (HIL) testing. To create and open a working copy of the conventional vehicle reference application project, enter

`autoblkConVehStart`

By default, the conventional vehicle reference application is configured with these powertrain subsystem variants:

- 1.5-L spark-ignition (SI) dynamic engine
- Performance mode transmission controller

This table describes the blocks and subsystems in the reference application, indicating which subsystems contain variants. To implement the model variants, the reference application uses variant subsystems.

Reference Application Element	Description	Variants
Drive Cycle Source block	Generates a standard or user-specified drive cycle velocity versus time profile. Block output is the selected or specified vehicle longitudinal speed.	
Environment subsystem	Creates environment variables, including road grade, wind velocity, and ambient temperature and pressure.	
Longitudinal Driver subsystem	Uses the Longitudinal Driver block to generate normalized acceleration and braking commands based on vehicle target and feedback velocities.	

Reference Application Element	Description	Variants
Controllers subsystem	Implements a powertrain control module (PCM) containing a transmission control module (TCM) and engine control module (ECM).	✓
Passenger Car subsystem	Implements a passenger car that contains transmission drivetrain and engine plant model subsystems.	✓
Visualization subsystem	Displays vehicle-level performance, fuel economy, and emission results that are useful for powertrain matching and component selection analysis.	

## Drive Cycle Source

The Drive Cycle Source block generates a target vehicle velocity for a selected or specified drive cycle. The reference application has these options.

Timing	Variant	Description
Output sample time	Continuous (default)	Continuous operator commands
	Discrete	Discrete operator commands

## Longitudinal Driver

The Longitudinal Driver subsystem generates normalized acceleration and braking commands based on the target vehicle velocity. The reference application has these options.

Block Options		Description
Control	Mapped	PI control with tracking windup and feed-forward gains that are a function of vehicle velocity.
	Predictive	Optimal single-point preview (look ahead) control.
	Scalar (default)	Proportional-integral (PI) control with tracking windup and feed-forward gains.
Low-pass filter (LPF)	LPF	Use an LPF on target velocity error for smoother driving.
	pass	Do not use a filter on velocity error.
Shift	Basic	Stateflow chart models reverse, neutral, and drive gear shift scheduling.
	External	Input gear, vehicle state, and velocity feedback generates acceleration and braking commands to track forward and reverse vehicle motion.
	None (default)	No transmission.
	Scheduled	Stateflow chart models reverse, neutral, park, and N-speed gear shift scheduling.

## Controllers

To implement a powertrain control module (PCM), the `Controller` subsystem has a transmission control module (TCM) and an engine control module (ECM). The reference application has these variants.

Controller	Variants	Description
Engine controller — ECM	<code>SiEngineController</code> (default)	SI engine controller
	<code>CiEngineController</code>	CI engine controller
Transmission controller — TCM	<code>PowertrainMaxPowerController</code> (default)	Performance mode transmission controller



Controller	Variant	Description
	PowertrainBestFuelController	Fuel economy mode transmission controller

## Passenger Car

To implement a passenger car, the **Passenger Car** subsystem contains drivetrain and engine plant model subsystems. To create your own internal combustion engine variants for the reference application, use the CI and SI engine project templates. The reference application has these variants.

Drivetrain Subsystem	Variant	Description
Dual clutch transmission (DCT)	DCT Block (default)	Configure drivetrain with DCT block or DCT system. For the DCT system, you can configure the type of filter.
	DCT System	
Differential and Compliance	All Wheel Drive	Configure drivetrain for all wheel, front wheel, or rear wheel drive. For the all wheel drive variant, you can configure the type of coupling torque.
	Front Wheel Drive (default)	
	Rear Wheel Drive	
Vehicle	Vehicle Body 3 DOF Longitudinal	Vehicle configured for 3 degrees of freedom.
Wheels and Brakes	All Wheel Drive	Configure drivetrain for all wheel, front wheel, or rear wheel drive. For the wheels, you can configure the type of: <ul style="list-style-type: none"> <li>• Brake</li> <li>• Force calculation</li> <li>• Resistance calculation</li> <li>• Vertical motion</li> </ul> For performance and clarity, to determine the longitudinal force of each wheel, the variants implement the Longitudinal Wheel block. To determine the <i>total</i> longitudinal force of all wheels acting on the axle, the variants use a
	Front Wheel Drive (default)	

Drivetrain Subsystem	Variant	Description
	Rear Wheel Drive	scale factor to multiply the force of one wheel by the number of wheels on the axle. By using this approach to calculate the total force, the variants assume equal tire slip and loading at the front and rear axles, which is common for longitudinal powertrain studies. If this is not the case, for example when friction or loads differ on the left and right sides of the axles, use unique Longitudinal Wheel blocks to calculate independent forces. However, using unique blocks to model each wheel increases model complexity and computational cost.

Engine Subsystem	Variant	Description
Engine	SiEngine (default)	Dynamic SI engine with turbocharger
	SiMappedEngine	Mapped SI engine with implicit turbocharger
	CiEngine	Dynamic CI engine with turbocharger
	CiMappedEngine	Mapped CI engine with implicit turbocharger

## See Also

CI Controller | CI Core Engine | Drive Cycle Source | Longitudinal Driver | Mapped CI Engine | Mapped SI Engine | SI Controller | SI Core Engine

## Related Examples

- “Conventional Vehicle Fuel Economy and Emissions” on page 1-13
- “CI Engine Project Template” on page 4-2
- “SI Engine Project Template” on page 4-5

## More About

- “Internal Combustion Mapped and Dynamic Engine Models” on page 3-83

- “Variant Systems” (Simulink)

## Explore the CI Engine Dynamometer Reference Application

The compression-ignition (CI) engine dynamometer reference application represents a CI engine plant and controller connected to an AC dynamometer with a tailpipe emission analyzer. Using the reference application, you can calibrate, validate, and optimize the engine controller and plant model parameters before integrating the engine with the vehicle model. To create and open a working copy of the CI engine dynamometer reference application project, enter

```
autoblkcIDynamometerStart
```

By default, the reference application is configured with a 1.5-L CI dynamic engine.

You can configure the reference application project for different dynamometer control modes. To implement the operating modes, the reference application uses variant subsystems.

This table summarizes the dynamometer tests.

Test	Objective	Method	CI Engine Variant	
			Mapped	Dynamic
Execute Engine Mapping Experiment	Assess engine torque, fuel flow, and emission performance results using an existing engine controller calibration.	Dynamometer controller commands a series of engine speeds and torques to the engine controller. At each quasi-steady-state operating point, the experiment records the engine plant model output and the controller commands for the current calibration parameters.	✓	✓

Test	Objective	Method	CI Engine Variant	
			Mapped	Dynamic
Execute Model Predictive Control Plant Model Experiment	Generate transient engine datasets for linear plant models useful for model predictive controllers.	Dynamometer controller commands engine speed and torque dynamically as a function of time using a pseudorandom binary sequence. Experiment records the transient engine torque, temperature, airflow, and emission responses determined from linear dynamic plant model fitting via system identification.	✓	✓
Recalibrate Controller	Match measured engine torque to commanded engine torque across engine operating range.	Dynamometer controller generates a feedforward fuel command table by matching the measured engine torque to the commanded engine torque across the engine operating range.		✓
Resize Engine and Recalibrate Controller	Match engine torque to desired engine power and number of cylinders.	Dynamometer resizes the dynamic engine and engine calibration parameters. Additionally, the dynamometer recalibrates the controller and mapped engine model to match the resized dynamic engine.	✓	✓

Test	Objective	Method	CI Engine Variant	
			Mapped	Dynamic
Generate Mapped Engine from Spreadsheet	Generate a mapped engine calibration from a data spreadsheet. Update the mapped engine with the calibrated data.	Dynamometer uses the Model-Based Calibration Toolbox to fit data from a spreadsheet, generate calibrated tables, and update the mapped engine parameters.	✓	

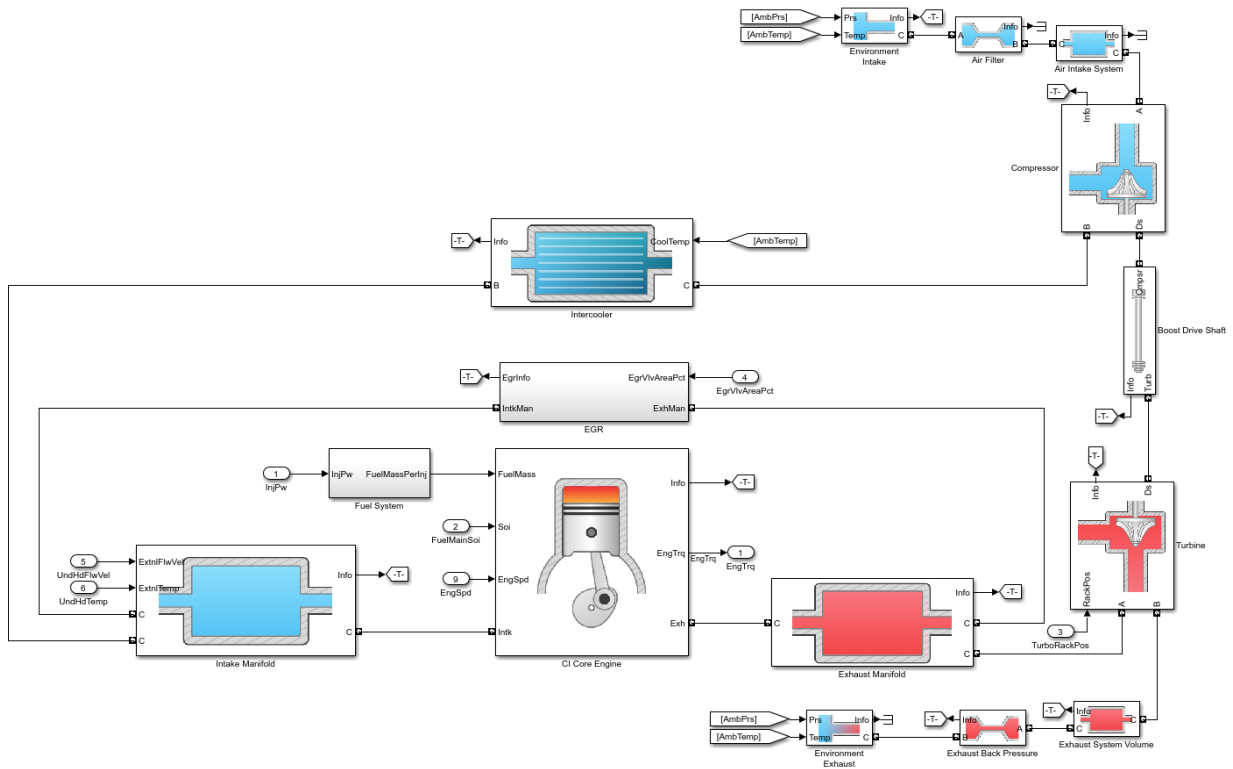
## Engine System

The reference application includes variant subsystems for mapped (steady-state) and dynamic 1.5-L CI engine systems with a variable geometry turbocharger (VGT). Using the CI engine project template, you can create your own CI engine variants.

Objective	Engine Variant
Dynamic analysis, including manifold and turbocharger dynamics	Dynamic
Faster execution	Mapped

### Dynamic

`CiEngineCore.slx` contains the engine intake system, exhaust system, exhaust gas recirculation (EGR), fuel system, core engine, and turbocharger subsystems.



## Mapped

CiMappedEngine.slx uses the Mapped CI Engine block to look up power, air mass flow, fuel flow, exhaust temperature, efficiency, and emission performance as functions of engine speed and injected fuel mass.

## Performance Monitor

The reference application contains a Performance Monitor block that you can use to plot steady-state and dynamic results. You can plot:

- Steady-state results as a function of one or two variables.
- Dynamic results using the Simulation Data Inspector.

### See Also

CI Controller | CI Core Engine | Mapped CI Engine

### More About

- “CI Engine Project Template” on page 4-2
- “Generate Mapped CI Engine from a Spreadsheet” on page 3-72
- “Resize the CI Engine” on page 3-54
- “Internal Combustion Mapped and Dynamic Engine Models” on page 3-83
- “Variant Systems” (Simulink)



## Explore the SI Engine Dynamometer Reference Application

The spark-ignition (SI) engine dynamometer reference application represents a SI engine plant and controller connected to an AC dynamometer with a tailpipe emission analyzer. Using the reference application, you can calibrate, validate, and optimize the engine controller and plant model parameters before integrating the engine with the vehicle model. To create and open a working copy of the SI engine dynamometer reference application project, enter

```
autobkSIDynamometerStart
```

By default, the reference application is configured with a 1.5-L SI dynamic engine.

You can configure the reference application project for different dynamometer control modes. To implement the operating modes, the reference application uses variant subsystems.

This table summarizes the dynamometer tests.

Test	Objective	Method	SI Engine Variant	
			Mapped	Dynamic
Execute Engine Mapping Experiment	Assess engine torque, fuel flow, and emission performance results using an existing engine controller calibration.	Dynamometer controller commands a series of engine speeds and torques to the engine controller. At each quasi-steady-state operating point, the experiment records the engine plant model output and the controller commands for the current calibration parameters.	✓	✓

Test	Objective	Method	SI Engine Variant	
			Mapped	Dynam ic
Execute Model Predictive Control Plant Model Experiment	Generate transient engine datasets for linear plant models useful for model predictive controllers.	Dynamometer controller commands engine speed and torque dynamically as a function of time using a pseudorandom binary sequence. Experiment records the transient engine torque, temperature, airflow, and emission responses determined from linear dynamic plant model fitting via system identification.	✓	✓
Recalibrate Controller	Match measured engine torque to commanded engine torque across engine operating range.	Dynamometer controller generates a feedforward throttle table by matching the measured engine torque to the commanded engine torque across the engine operating range.		✓
Resize Engine and Recalibrate Controller	Match engine torque to desired engine power and number of cylinders.	Dynamometer resizes the dynamic engine and engine calibration parameters. Additionally, the dynamometer recalibrates the controller and mapped engine model to match the resized dynamic engine.	✓	✓

Test	Objective	Method	SI Engine Variant	
			Mapped	Dynamic
Generate Mapped Engine from Spreadsheet	Generate a mapped engine calibration from a data spreadsheet. Update the mapped engine with the calibrated data.	Dynamometer uses the Model-Based Calibration Toolbox to fit data from a spreadsheet, generate calibrated tables, and update the mapped engine parameters.	✓	

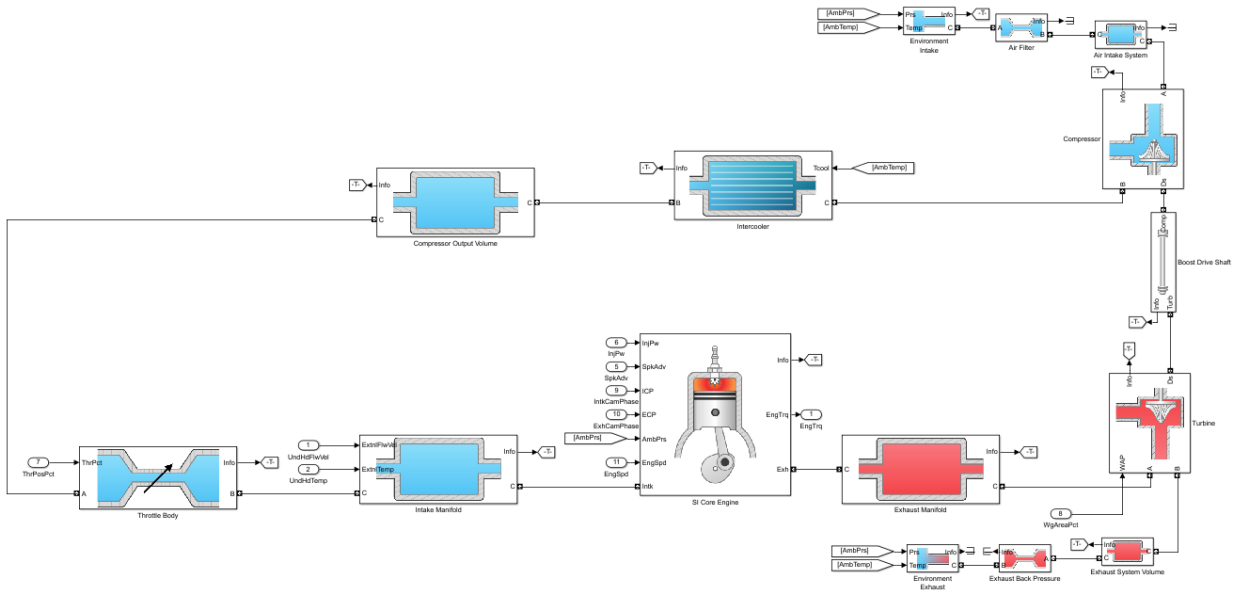
## Engine System

The reference application includes variant subsystems for mapped (steady-state) and dynamic turbocharged 1.5-L SI engine. Using the SI engine project template, you can create your own SI engine variants.

Objective	Engine Variant
Dynamic analysis, including manifold and turbocharger dynamics	Dynamic
Faster execution	Mapped

### Dynamic

SiEngineCore.slx contains the engine intake system, exhaust system, core engine, and turbocharger subsystems.



## Mapped

SiMappedEngine.slx uses the Mapped SI Engine block to look up power, air mass flow, fuel flow, exhaust temperature, efficiency, and emission performance as functions of engine speed and commanded torque.

## Performance Monitor

The reference application contains a Performance Monitor block that you can use to plot steady-state and dynamic results. You can plot:

- Steady-state results as a function of one or two variables.
- Dynamic results using the Simulation Data Inspector.

## See Also

Mapped SI Engine | SI Controller | SI Core Engine

## **More About**

- “SI Engine Project Template” on page 4-5
- “Generate Mapped SI Engine from a Spreadsheet” on page 3-78
- “Resize the SI Engine” on page 3-63
- “Internal Combustion Mapped and Dynamic Engine Models” on page 3-83
- “Variant Systems” (Simulink)

## Explore the Hybrid Electric Vehicle Multimode Reference Application

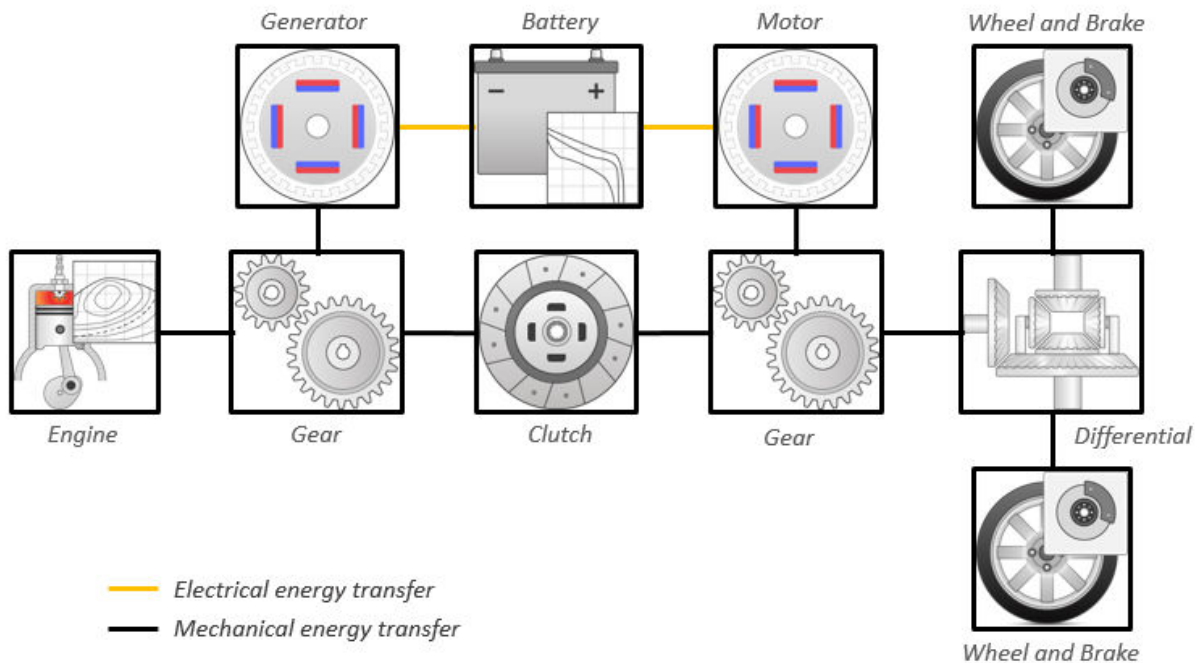
The hybrid electric vehicle reference application represents a full multimode hybrid electric vehicle (HEV) model with an internal combustion engine, transmission, battery, motor, generator, and associated powertrain control algorithms. Use the reference application for powertrain matching analysis and component selection, control and diagnostic algorithm design, and hardware-in-the-loop (HIL) testing. To create and open a working copy of the hybrid electric vehicle reference application project, enter

`autoblkHevStart`

By default, the HEV multimode reference application is configured with:

- Mapped motor and generator
- 1.5-L spark-ignition (SI) dynamic engine

This diagram shows the powertrain configuration.



This table describes the blocks and subsystems in the reference application, indicating which subsystems contain variants. To implement the model variants, the reference application uses variant subsystems.

<b>Reference Application Element</b>	<b>Description</b>	<b>Variants</b>
Drive Cycle Source block	Generates a standard or user-specified drive cycle velocity versus time profile. Block output is the selected or specified vehicle longitudinal speed.	
Environment subsystem	Creates environment variables, including road grade, wind velocity, and atmospheric temperature and pressure.	
Longitudinal Driver subsystem	Uses the Longitudinal Driver block to generate normalized acceleration and braking commands based on vehicle target and feedback velocities.	
Controllers subsystem	Implements a powertrain control module (PCM) containing a hybrid control module (HCM) and an engine control module (ECM).	✓
Passenger Car subsystem	Implements a hybrid passenger car that contains engine, electric plant, and drivetrain subsystems.	✓

Reference Application Element	Description	Variants
Visualization subsystem	Displays vehicle-level performance, battery state of charge (SOC), fuel economy, and emission results that are useful for powertrain matching and component selection analysis.	

## Drive Cycle Source

The Drive Cycle Source block generates a target vehicle velocity for a selected or specified drive cycle. The reference application has these options.

Timing	Variant	Description
Output sample time	Continuous (default)	Continuous operator commands
	Discrete	Discrete operator commands

## Longitudinal Driver

The Longitudinal Driver subsystem generates normalized acceleration and braking commands based on the target vehicle velocity. The reference application has these options.

Block Options	Description	
Control	Mapped	PI control with tracking windup and feed-forward gains that are a function of vehicle velocity.
	Predictive	Optimal single-point preview (look ahead) control.
	Scalar (default)	Proportional-integral (PI) control with tracking windup and feed-forward gains.
Low-pass filter (LPF)	LPF	Use an LPF on target velocity error for smoother driving.



Block Options		Description
	pass	Do not use a filter on velocity error.
Shift	Basic	Stateflow chart models reverse, neutral, and drive gear shift scheduling.
	External	Input gear, vehicle state, and velocity feedback generates acceleration and braking commands to track forward and reverse vehicle motion.
	None (default)	No transmission.
	Scheduled	Stateflow chart models reverse, neutral, park, and N-speed gear shift scheduling.

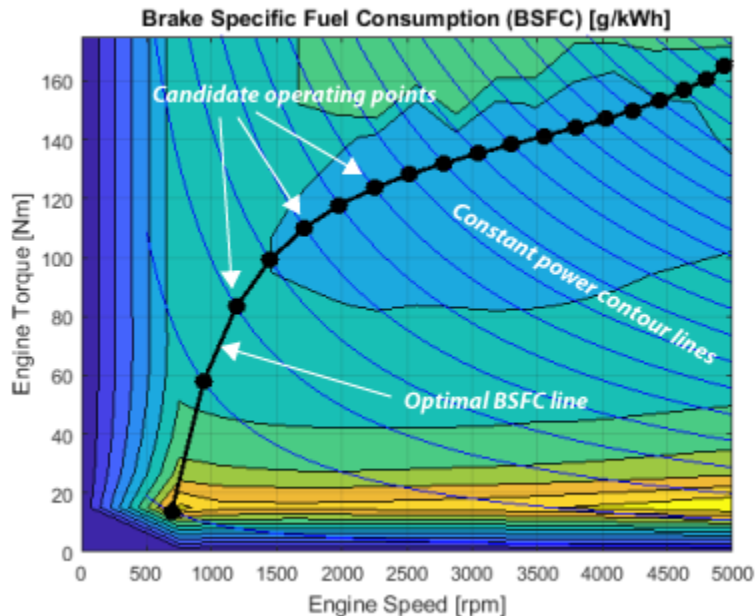
## Controllers

The Controller subsystem has a PCM with a HCM and an ECM. The reference application has these variants for the ECM.

Controller	Variant	Description
ECM	SiEngineController (default)	SI engine controller
	CiEngineController	CI engine controller

The HCM implements a dynamic embedded controller that directly determines the engine operating point that minimizes brake-specific fuel consumption (BSFC) while meeting or exceeding power required by the battery charging and vehicle propulsion subsystems.

To calculate the optimal engine operating point in speed and torque, the controller starts with a candidate set of discrete engine power levels. For each power level candidate, the block has a parameterized vector of torque and speed operating points that minimize BSFC.



The optimizer then removes power level candidates that are unacceptable for either of these reasons:

- Too much power sent through the generator to the battery.
- Too little power to meet charging and propulsion subsystem requirements.

Of the remaining power level candidates, the controller selects the one with the lowest BSFC. The controller then sends the associated torque / speed operating point command to the engine.

## Passenger Car

To implement a passenger car, the Passenger Car subsystem contains drivetrain, electric plant, and engine subsystems. To create your own engine variants for the reference application, use the CI and SI engine project templates. The reference application has these variants.

<b>Drivetrain Subsystem</b>	<b>Variant</b>	<b>Description</b>
Differential and Compliance	All Wheel Drive	Configure drivetrain for all wheel, front wheel, or rear wheel drive. For the all wheel drive variant, you can configure the type of coupling torque.
	Front Wheel Drive (default)	
	Rear Wheel Drive	
Vehicle	Vehicle Body 3 DOF Longitudinal	Configured for 3 degrees of freedom
Wheels and Brakes	All Wheel Drive	<p>For the wheels, you can configure the type of:</p> <ul style="list-style-type: none"> <li>• Brake</li> <li>• Force calculation</li> <li>• Resistance calculation</li> <li>• Vertical motion</li> </ul> <p>For performance and clarity, to determine the longitudinal force of each wheel, the variants implement the Longitudinal Wheel block. To determine the <i>total</i> longitudinal force of all wheels acting on the axle, the variants use a scale factor to multiply the force of one wheel by the number of wheels on the axle. By using this approach to calculate the total force, the variants assume equal tire slip and loading at the front and rear axles, which is common for longitudinal powertrain studies. If this is not the case, for example when friction or loads differ on the left and right sides of the axles, use unique Longitudinal Wheel blocks to calculate independent forces. However, using unique blocks to model each wheel increases model complexity and computational cost.</p>
	Front Wheel Drive (default)	
	Rear Wheel Drive	

<b>Electric Plant Subsystem</b>	<b>Variant</b>	<b>Description</b>
Battery	BattHevMm (default)	Configured with electric battery

Electric Plant Subsystem	Variant	Description
Generator	GenMapped (default)	Mapped generator
	GenDynamic	Interior permanent magnet synchronous motor (PMSM) with controller
Motor	MotMapped (default)	Mapped motor with implicit controller
	MotDynamic	Interior permanent magnet synchronous motor (PMSM) with controller

Engine Subsystem	Variant	Description
Engine	SiEngine (default)	Dynamic SI engine with turbocharger
	SiMappedEngine	Mapped SI engine with implicit turbocharger
	CiEngine	Dynamic CI engine with turbocharger
	CiMappedEngine	Mapped CI engine with implicit turbocharger

## References

- [1] Higuchi, N., Shimada, H., Sunaga, Y., and Tanaka, M., *Development of a New Two-Motor Plug-In Hybrid System*. SAE Technical Paper 2013-01-1476. Warrendale, PA: SAE International Journal of Alternative Powertrains, 2013.

## See Also

CI Controller | CI Core Engine | Datasheet Battery | Drive Cycle Source | Interior PMSM | Interior PM Controller | Longitudinal Driver | Mapped CI Engine | Mapped SI Engine | SI Controller | SI Core Engine

## Related Examples

- “Explore the Hybrid Electric Vehicle Input Power-Split Reference Application” on page 3-34
- “Explore the Hybrid Electric Vehicle P2 Reference Application” on page 3-44
- “Explore the Electric Vehicle Reference Application” on page 3-28

## **More About**

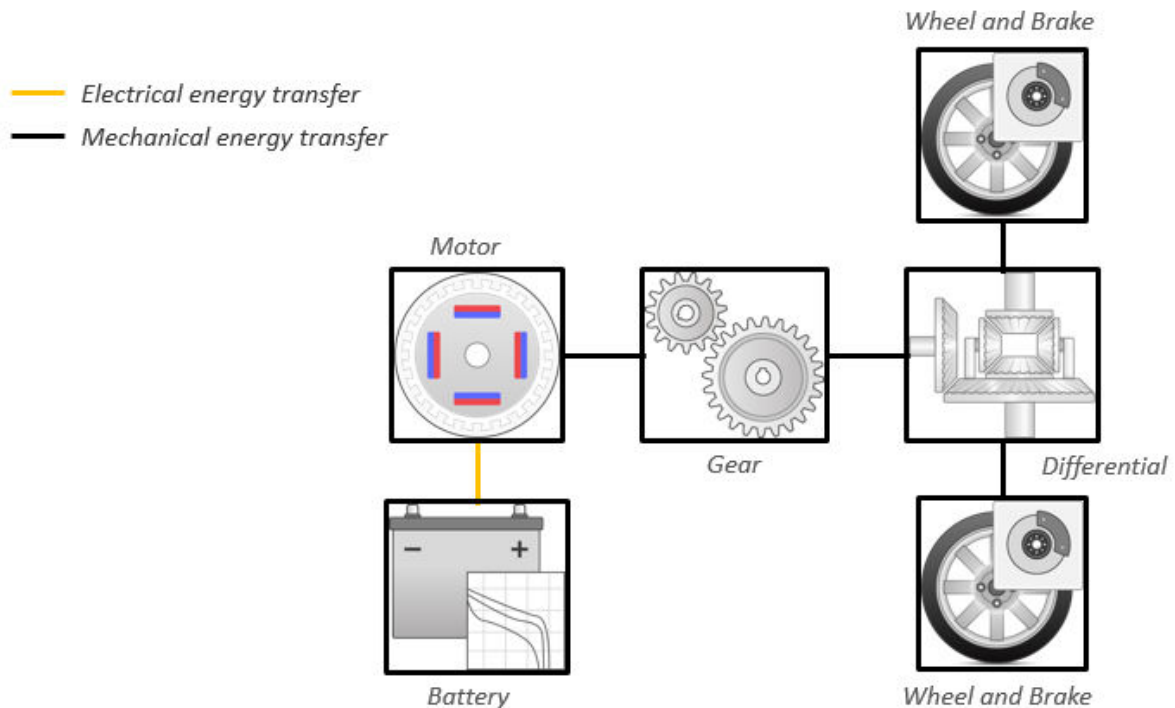
- “CI Engine Project Template” on page 4-2
- “SI Engine Project Template” on page 4-5
- “Variant Systems” (Simulink)

## Explore the Electric Vehicle Reference Application

The electric vehicle reference application represents a full electric vehicle model with a motor-generator, battery, direct-drive transmission, and associated powertrain control algorithms. Use the electric vehicle reference application for powertrain matching analysis and component selection, control and diagnostic algorithm design, and hardware-in-the-loop (HIL) testing. To create and open a working copy of the conventional vehicle reference application project, enter

`autoblkEvStart`

The electric vehicle reference application is configured with a mapped motor and battery. This diagram shows the powertrain configuration.



This table describes the blocks and subsystems in the reference application, indicating which subsystems contain variants. To implement the model variants, the reference application uses variant subsystems.

Reference Application Element	Description	Variants
Drive Cycle Source block	Generates a standard or user-specified drive cycle velocity versus time profile. Block output is the selected or specified vehicle longitudinal speed.	
Environment subsystem	Creates environment variables, including road grade, wind velocity, and atmospheric temperature and pressure.	
Longitudinal Driver subsystem	Uses the Longitudinal Driver block to generate normalized acceleration and braking commands based on vehicle target and feedback velocities.	
Controllers subsystem	Implements a powertrain control module (PCM) with regenerative braking, motor torque arbitration and power management.	✓
Passenger Car subsystem	Implements a passenger car that contains an electric plant and drivetrain subsystems.	✓
Visualization subsystem	Displays vehicle-level performance, battery state of charge (SOC), and equivalent fuel economy results that are useful for powertrain matching and component selection analysis.	

## Drive Cycle Source

The Drive Cycle Source block generates a target vehicle velocity for a selected or specified drive cycle. The reference application has these options.

Timing	Variant	Description
Output sample time	Continuous (default)	Continuous operator commands
	Discrete	Discrete operator commands

## Longitudinal Driver

The Longitudinal Driver subsystem generates normalized acceleration and braking commands based on the target vehicle velocity. The reference application has these options.

Block Options		Description
Control	Mapped	PI control with tracking windup and feed-forward gains that are a function of vehicle velocity.
	Predictive	Optimal single-point preview (look ahead) control.
	Scalar (default)	Proportional-integral (PI) control with tracking windup and feed-forward gains.
Low-pass filter (LPF)	LPF	Use an LPF on target velocity error for smoother driving.
	pass	Do not use a filter on velocity error.
Shift	Basic	Stateflow chart models reverse, neutral, and drive gear shift scheduling.
	External	Input gear, vehicle state, and velocity feedback generates acceleration and braking commands to track forward and reverse vehicle motion.
	None (default)	No transmission.
	Scheduled	Stateflow chart models reverse, neutral, park, and N-speed gear shift scheduling.



## Controllers

To determine the motor torque and brake pressure commands, the reference application implements a supervisory controller. Specifically, the controller subsystem includes a powertrain control module (PCM) with:

- Regenerative braking control
- Motor torque arbitration and power management
  - Converts the driver accelerator pedal signal to a torque request.
  - Converts the driver brake pedal signal to a brake pressure request. The algorithm multiplies the brake pedal signal by a maximum brake pressure.
  - Implements a regenerative braking algorithm for the traction motor to recover the maximum amount of kinetic energy from the vehicle.
  - Implements a virtual battery management system. The algorithm outputs the dynamic discharge and charge power limits as functions of battery state of charge (SOC).
  - Implements a power management algorithm that ensures the battery dynamic discharge and charge power limits are not exceeded.

Regen Braking Control has these variants.

Controller	Variant	Description
Regen Braking Control	Series Regen Brake (default)	Friction braking provides the torque not supplied by regenerative motor braking.
	Parallel Regen Braking	Friction braking and regenerative motor braking independently provide the torque.

## Passenger Car

To implement a passenger car, the `Passenger Car` subsystem contains a drivetrain and electric plant subsystem. The reference application has these variants.

Drivetrain Subsystem	Variant	Description
Differential and Compliance	All Wheel Drive	Configure drivetrain for all wheel, front wheel, or rear wheel drive. For the all wheel drive variant, you can configure the type of coupling torque.
	Front Wheel Drive (default)	
	Rear Wheel Drive	
Vehicle	Vehicle Body 3 DOF Longitudinal	Configured for 3 degrees of freedom
Wheels and Brakes	All Wheel Drive	<p>For the wheels, you can configure the type of:</p> <ul style="list-style-type: none"> <li>• Brake</li> <li>• Force calculation</li> <li>• Resistance calculation</li> <li>• Vertical motion</li> </ul> <p>For performance and clarity, to determine the longitudinal force of each wheel, the variants implement the Longitudinal Wheel block. To determine the <i>total</i> longitudinal force of all wheels acting on the axle, the variants use a scale factor to multiply the force of one wheel by the number of wheels on the axle. By using this approach to calculate the total force, the variants assume equal tire slip and loading at the front and rear axles, which is common for longitudinal powertrain studies. If this is not the case, for example when friction or loads differ on the left and right sides of the axles, use unique Longitudinal Wheel blocks to calculate independent forces. However, using unique blocks to model each wheel increases model complexity and computational cost.</p>
	Front Wheel Drive (default)	
	Rear Wheel Drive	

Electric Plant Subsystem	Variant	Description
Battery	BattEv (default)	Configured with electric battery

<b>Electric Plant Subsystem</b>	<b>Variant</b>	<b>Description</b>
Motor	MotGenEvMapped (default)	Mapped motor with implicit controller
	MotGenEvDynamic	Interior permanent magnet synchronous motor (PMSM) with controller

## See Also

Datasheet Battery | Drive Cycle Source | Interior PM Controller | Interior PMSM | Longitudinal Driver | Mapped Motor

## Related Examples

- “Explore the Hybrid Electric Vehicle Multimode Reference Application” on page 3-20
- “Explore the Hybrid Electric Vehicle Input Power-Split Reference Application” on page 3-34
- “Explore the Hybrid Electric Vehicle P2 Reference Application” on page 3-44

## More About

- “Variant Systems” (Simulink)

# Explore the Hybrid Electric Vehicle Input Power-Split Reference Application

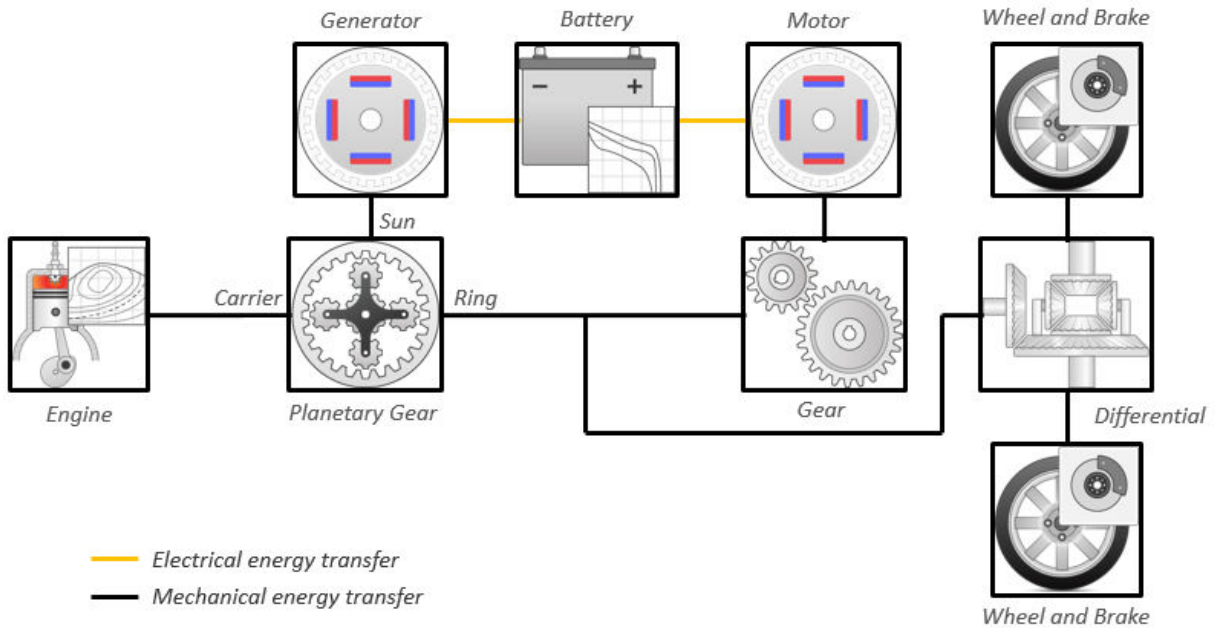
The hybrid electric vehicle (HEV) input power-split reference application represents a full HEV model with an internal combustion engine, transmission, battery, motor, generator, and associated powertrain control algorithms. Use the HEV input power-split reference application for HIL testing, tradeoff analysis, and control parameter optimization of a power-split hybrid like the Toyota® Prius®. To create and open a working copy of the HEV input power-split reference application project, enter

```
autoblkHevIpsStart
```

By default, the HEV input power-split reference application is configured with:

- Nickel-metal hydride (NiMH) battery pack
- Mapped electric motors
- Mapped spark-ignition (SI) engine

This diagram shows the powertrain configuration.



This table describes the blocks and subsystems in the reference application, indicating which subsystems contain variants. To implement the model variants, the reference application uses variant subsystems.

Reference Application Element	Description	Variants
Drive Cycle Source block	Generates a standard or user-specified drive cycle velocity versus time profile. Block output is the selected or specified vehicle longitudinal speed.	
Environment subsystem	Creates environment variables, including road grade, wind velocity, and atmospheric temperature and pressure.	

Reference Application Element	Description	Variants
Longitudinal Driver subsystem	Uses the Longitudinal Driver block to generate normalized acceleration and braking commands based on vehicle target and feedback velocities.	
Controllers subsystem	Implements a powertrain control module (PCM) containing an input power-split hybrid control module (HCM) and an engine control module (ECM).	✓
Passenger Car subsystem	Implements a hybrid passenger car that contains drivetrain, electric plant, and engine subsystems.	✓
Visualization subsystem	Displays vehicle-level performance, battery state of charge (SOC), fuel economy, and emission results that are useful for powertrain matching and component selection analysis.	

## Drive Cycle Source

The Drive Cycle Source block generates a target vehicle velocity for a selected or specified drive cycle. The reference application has these options.

Timing	Variant	Description
Output sample time	Continuous (default)	Continuous operator commands
	Discrete	Discrete operator commands

## Longitudinal Driver

The Longitudinal Driver subsystem generates normalized acceleration and braking commands based on the target vehicle velocity. The reference application has these options.

Block Options		Description
Control	Mapped	PI control with tracking windup and feed-forward gains that are a function of vehicle velocity.
	Predictive	Optimal single-point preview (look ahead) control.
	Scalar (default)	Proportional-integral (PI) control with tracking windup and feed-forward gains.
Low-pass filter (LPF)	LPF	Use an LPF on target velocity error for smoother driving.
	pass (default)	Do not use a filter on velocity error.
Shift	Basic	Stateflow chart models reverse, neutral, and drive gear shift scheduling.
	External	Input gear, vehicle state, and velocity feedback generates acceleration and braking commands to track forward and reverse vehicle motion.
	None (default)	No transmission.
	Scheduled	Stateflow chart models reverse, neutral, park, and N-speed gear shift scheduling.

## Controllers

The Controller subsystem has a PCM containing an input power-split HCM and an ECM. The controller has these variants.

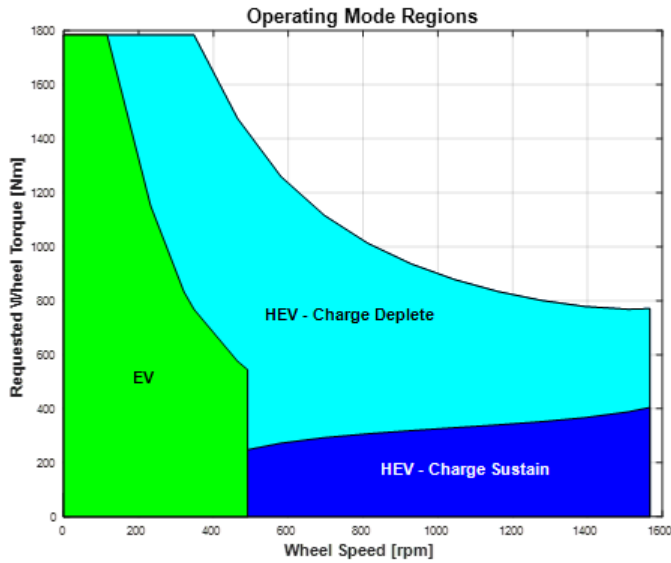
Controller	Variant	Description
ECM	SiEngineController (default)	SI engine controller

<b>Controller</b>	<b>Variant</b>	<b>Description</b>
Input power split HCM	Series Regen Brake (default)	Friction braking provides the torque not supplied by regenerative motor braking.
	Parallel Regen Braking	Friction braking and regenerative motor braking independently provide the torque.

The input-power split HCM implements a dynamic supervisory controller that determines the engine torque, generator torque, motor torque, and brake pressure commands. Specifically, the input power-split HCM:

- Converts the driver accelerator pedal signal to a wheel torque request. The algorithm uses the optimal engine torque and maximum motor torque curves to calculate the total powertrain torque at the wheels.
- Converts the driver brake pedal signal to a brake pressure request. The algorithm multiplies the brake pedal signal by a maximum brake pressure.
- Implements a regenerative braking algorithm for the traction motor to recover the maximum amount of kinetic energy from the vehicle.
- Implements a virtual battery management system. The algorithm outputs the dynamic discharge and charge power limits as functions of battery SOC.
- Determines the vehicle operating mode through a set of rules and decision logic implemented in Stateflow. The operating modes are functions of wheel speed and requested wheel torque. The algorithm uses the wheel power request, accelerator pedal, battery SOC, and vehicle speed rules to transition between electric vehicle (EV) and HEV modes.



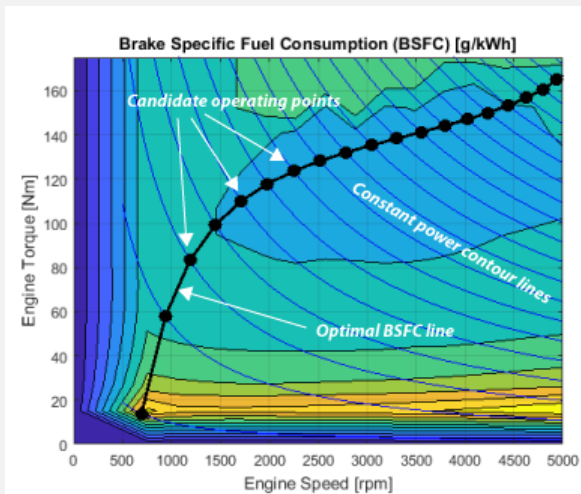


Mode	Description
EV	Traction motor provides the wheel torque request.
HEV - Charge Sustaining (Low Power)	<ul style="list-style-type: none"> <li>• Engine provides the wheel torque request.</li> <li>• Torque blending algorithm transitions the torque production from the EV motor to the HEV engine. The algorithm allows the motor to ramp down the torque while the engine torque ramps up. Once the blending is complete, the motor can start sustaining the charge (negative torque), if needed.</li> <li>• Based on the target battery SOC and available kinetic energy, the HEV mode determines a charge sustain power level. The mode includes the additional charge power in the engine power command. To provide the desired charge power, the traction motor acts as a generator.</li> <li>• Depending on the instantaneous speeds of the engine and motor, the generator may consume energy while regulating the engine speed. In this case, the motor provides the additional charge sustaining power.</li> </ul>

Mode	Description
HEV - Charge Depleting (High Power)	<ul style="list-style-type: none"> <li>Engine provides the wheel power request up to its maximum output.</li> <li>If the wheel torque request is greater than the engine torque output at the wheels, the traction motor provides the remainder of the wheel torque request.</li> </ul>
Stationary	While the vehicle is at rest, the engine and generator can provide optional charging if battery SOC is below a minimum SOC value.

- Controls the motor, generator, and engine through a set of rules and decision logic implemented in Stateflow.

Control	Description
Engine	<ul style="list-style-type: none"> <li>Decision logic determines the engine operation modes (off, start, run).</li> <li>In engine run mode, lookup tables determine the engine torque and engine speed that optimizes the brake specific fuel consumption (BSFC) for a given engine power request. The ECM uses the optimal engine torque command. The generator control uses the optimal engine speed command.</li> </ul>



Control	Description
Generator	<ul style="list-style-type: none"> <li>As determined by the HCM, the generator either starts the engine or regulates the engine speed. To regulate the engine speed, the generator uses a PI controller.</li> <li>A rule-based power management algorithm calculates a generator torque that does not exceed the dynamic power limits.</li> </ul>
Motor	A rule-based power management algorithm calculates a motor torque that does not exceed the dynamic power limits.

## Passenger Car

To implement a passenger car, the **Passenger Car** subsystem contains drivetrain, electric plant, and engine subsystems. To create your own engine variants for the reference application, use the CI and SI engine project templates. The reference application has these variants.

Drivetrain Subsystem	Variant	Description
Differential and Compliance	All Wheel Drive	Configure drivetrain for all wheel, front wheel, or rear wheel drive. For the all wheel drive variant, you can configure the type of coupling torque.
	Front Wheel Drive (default)	
	Rear Wheel Drive	
Vehicle	Vehicle Body 3 DOF Longitudinal	Configured for 3 degrees of freedom
Wheels and Brakes	All Wheel Drive	For the wheels, you can configure the type of: <ul style="list-style-type: none"> <li>Brake</li> <li>Force calculation</li> <li>Resistance calculation</li> <li>Vertical motion</li> </ul> For performance and clarity, to determine the longitudinal force of each wheel, the variants implement the Longitudinal Wheel block. To determine the <i>total</i> longitudinal force of all wheels acting on the axle, the variants use a
	Front Wheel Drive (default)	

<b>Drivetrain Subsystem</b>	<b>Variant</b>	<b>Description</b>
	Rear Wheel Drive	scale factor to multiply the force of one wheel by the number of wheels on the axle. By using this approach to calculate the total force, the variants assume equal tire slip and loading at the front and rear axles, which is common for longitudinal powertrain studies. If this is not the case, for example when friction or loads differ on the left and right sides of the axles, use unique Longitudinal Wheel blocks to calculate independent forces. However, using unique blocks to model each wheel increases model complexity and computational cost.

<b>Electric Plant Subsystem</b>	<b>Variant</b>	<b>Description</b>
Battery and DC-DC Converter	BattHevIps	Configured with NiMH battery
Generator	GenMapped (default)	Mapped generator with implicit controller
	GenDynamic	Interior permanent magnet synchronous motor (PMSM) with controller
Motor	MotMapped (default)	Mapped motor with implicit controller
	MotDynamic	Interior permanent magnet synchronous motor (PMSM) with controller

<b>Engine Subsystem</b>	<b>Variant</b>	<b>Description</b>
Engine	SiMappedEngine (default)	Mapped SI engine

## References

- [1] Balazs, A., Morra, E., and Pischinger, S., *Optimization of Electrified Powertrains for City Cars*. SAE Technical Paper 2011-01-2451. Warrendale, PA: SAE International Journal of Alternative Powertrains, 2012.

- [2] Burress, T. A. et al, *Evaluation of the 2010 Toyota Prius Hybrid Synergy Drive System*. Technical Report ORNL/TM-2010/253. U.S. Department of Energy, Oak Ridge National Laboratory, March 2011.
- [3] Rask, E., Duoba, M., Loshse-Busch, H., and Bocci, D., *Model Year 2010 (Gen 3) Toyota Prius Level-1 Testing Report*. Technical Report ANL/ES/RP-67317. U.S. Department of Energy, Argonne National Laboratory, September 2010.

## See Also

CI Controller | CI Core Engine | Datasheet Battery | Drive Cycle Source | Interior PM Controller | Interior PMSM | Longitudinal Driver | Mapped CI Engine | Mapped SI Engine | SI Controller | SI Core Engine

## Related Examples

- “Explore the Hybrid Electric Vehicle Multimode Reference Application” on page 3-20
- “Explore the Hybrid Electric Vehicle P2 Reference Application” on page 3-44
- “Explore the Electric Vehicle Reference Application” on page 3-28

## More About

- “SI Engine Project Template” on page 4-5
- “Variant Systems” (Simulink)

# Explore the Hybrid Electric Vehicle P2 Reference Application

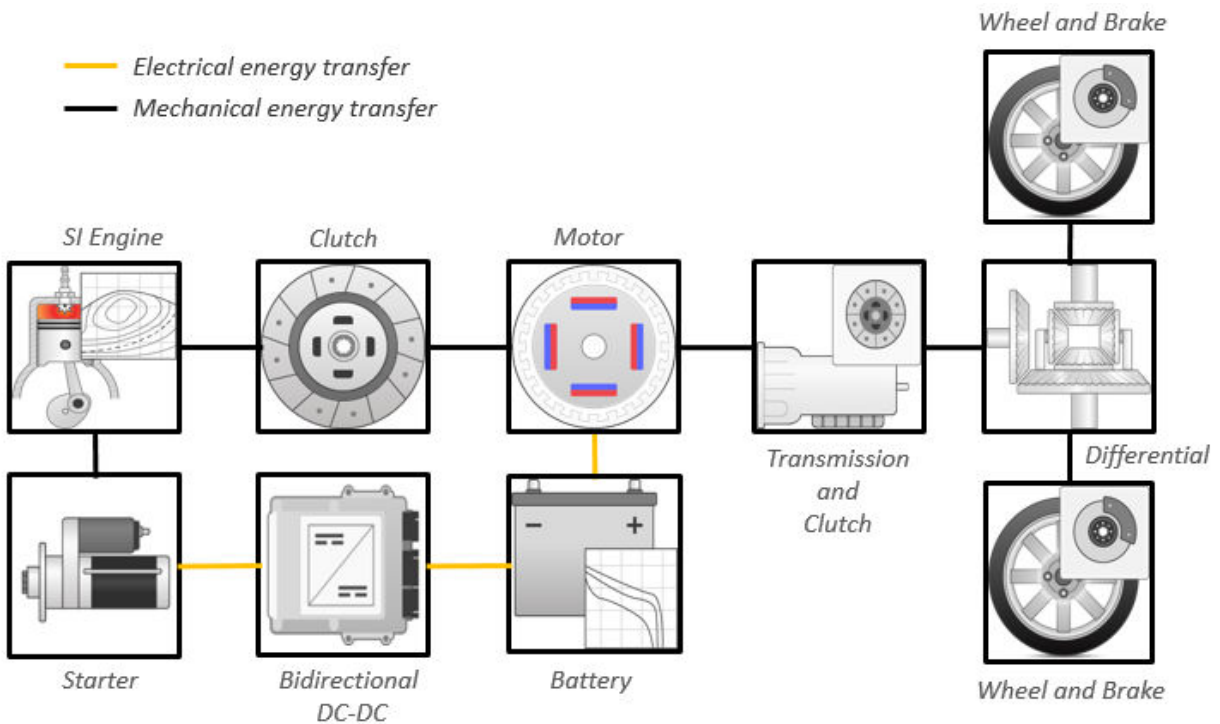
The hybrid electric vehicle (HEV) P2 reference application represents a full HEV model with an internal combustion engine, transmission, battery, motor, and associated powertrain control algorithms. Use the reference application for hardware-in-the-loop (HIL) testing, tradeoff analysis, and control parameter optimization of a HEV P2 hybrid. To create and open a working copy of the reference application project, enter

```
autoblkHevP2Start
```

By default, the HEV P2 reference application is configured with:

- Lithium-ion battery pack
- Mapped electric motor
- Mapped spark-ignition (SI) engine

This diagram shows the powertrain configuration.



This table describes the blocks and subsystems in the reference application, indicating which subsystems contain variants. To implement the model variants, the reference application uses variant subsystems.

Reference Application Element	Description	Variants
Drive Cycle Source block	Generates a standard or user-specified drive cycle velocity versus time profile. Block output is the selected or specified vehicle longitudinal speed.	

Reference Application Element	Description	Variants
Environment subsystem	Creates environment variables, including road grade, wind velocity, and atmospheric temperature and pressure.	
Longitudinal Driver subsystem	Uses the Longitudinal Driver block to generate normalized acceleration and braking commands based on vehicle target and feedback velocities.	
Controllers subsystem	Implements a powertrain control module (PCM) containing a P2 hybrid control module (HCM), an engine control module (ECM), and a transmission control module (TCM).	✓
Passenger Car subsystem	Implements a hybrid passenger car that contains drivetrain, electric plant, and engine subsystems.	✓
Visualization subsystem	Displays vehicle-level performance, battery state of charge (SOC), fuel economy, and emission results that are useful for powertrain matching and component selection analysis.	

## Drive Cycle Source

The `Drive Cycle Source` block generates a target vehicle velocity for a selected or specified drive cycle. The reference application has these options.

Timing	Variant	Description
Output sample time	Continuous (default)	Continuous operator commands



Timing	Variant	Description
	Discrete	Discrete operator commands

## Longitudinal Driver

The Longitudinal Driver subsystem generates normalized acceleration and braking commands based on the target vehicle velocity. The reference application has these options.

Block Options		Description
Control	Mapped	PI control with tracking windup and feed-forward gains that are a function of vehicle velocity.
	Predictive	Optimal single-point preview (look ahead) control.
	Scalar (default)	Proportional-integral (PI) control with tracking windup and feed-forward gains.
Low-pass filter (LPF)	LPF	Use an LPF on target velocity error for smoother driving.
	pass	Do not use a filter on velocity error.
Shift	Basic	Stateflow chart models reverse, neutral, and drive gear shift scheduling.
	External	Input gear, vehicle state, and velocity feedback generates acceleration and braking commands to track forward and reverse vehicle motion.
	None (default)	No transmission.
	Scheduled	Stateflow chart models reverse, neutral, park, and N-speed gear shift scheduling.

## Controllers

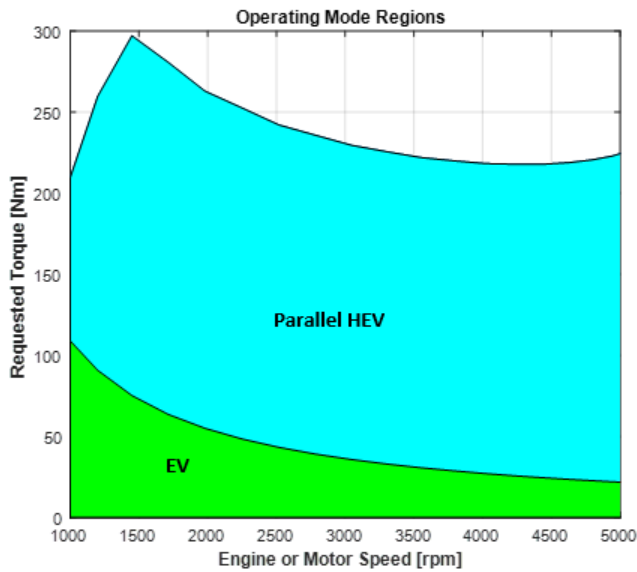
The Controller subsystem has a PCM containing an ECM, HCM, and TCM. The controller has these variants.

<b>Controller</b>	<b>Variant</b>	<b>Description</b>
ECM	SiEngineController (default)	SI engine controller
	CiEngineController	CI engine controller
HCM	Series Regen Brake (default)	Friction braking provides the torque not supplied by regenerative motor braking.
	Parallel Regen Braking	Friction braking and regenerative motor braking independently provide the torque.

The P2 HCM implements a dynamic supervisory controller that determines the engine torque, motor torque, starter, clutch, and brake pressure commands. Specifically, the P2 HCM:

- Converts the driver accelerator pedal signal to a torque request. The algorithm uses the optimal engine torque and maximum motor torque curves to calculate the total powertrain torque.
- Converts the driver brake pedal signal to a brake pressure request. The algorithm multiplies the brake pedal signal by a maximum brake pressure.
- Implements a regenerative braking algorithm for the traction motor to recover the maximum amount of kinetic energy from the vehicle.
- Implements a virtual battery management system. The algorithm outputs the dynamic discharge and charge power limits as functions of battery SOC.

The P2 HCM determines the vehicle operating mode through a set of rules and decision logic implemented in Stateflow. The operating modes are functions of motor speed and requested torque. The algorithm uses the calculated power request, accelerator pedal, battery SOC, and vehicle speed rules to transition between electric vehicle (EV) and parallel HEV modes.



Mode	Description
EV	Traction motor provides the torque request.
Parallel HEV	The engine and the motor split the power request. Based on the target battery SOC and available kinetic energy, the HEV mode determines a charge sustain power level. The parallel HEV mode adds the charge sustain power to the engine power command. To provide the desired charge sustain power, the traction motor acts as a generator if charging is needed, and as a motor if discharging is needed. If the power request is greater than the engine power, the traction motor provides the remainder of the power request.
Stationary	While the vehicle is at rest, the engine and generator can provide optional charging if battery SOC is below a minimum SOC value.

The P2 HCM controls the motor, and engine through a set of rules and decision logic implemented in Stateflow.

Control	Description
Engine	<ul style="list-style-type: none"> <li>Decision logic determines the engine operation modes (off, start, on).</li> <li>To start the engine in engine start (stationary) mode, the P2 motor closes clutch 1 and puts the transmission in neutral. If the high-voltage battery SOC is low, the mode uses the low-voltage starter motor.</li> <li>To start the engine in engine start (driving) mode, the mode uses the low-voltage starter motor with clutch 1 open. To connect the driveline, the engine controller matches the engine and motor speeds and closes clutch 1.</li> <li>In engine on (stationary) mode, lookup tables determine the engine torque and engine speed that optimizes the brake-specific fuel consumption (BSFC) for a given engine power request. The ECM uses the optimal engine torque command. The motor control uses the optimal engine speed command.</li> </ul> <div data-bbox="394 748 980 1239" data-label="Figure"> </div> <ul style="list-style-type: none"> <li>In engine on (parallel HEV) mode, a lookup table determines the engine torque for a given engine power. However, because the drivetrain couples the engine and wheel speeds, engine on mode might not operate at speeds that minimize BSFC.</li> </ul>
Motor	<p>A rule-based power management algorithm calculates a motor torque that does not exceed the dynamic power limits.</p>

## Passenger Car

To implement a passenger car, the **Passenger Car** subsystem contains drivetrain, electric plant, and engine subsystems. To create your own engine variants for the reference application, use the CI and SI engine project templates. The reference application has these variants.

Drivetrain Subsystem	Variant	Description
Differential and Compliance	All Wheel Drive	Configure drivetrain for all wheel, front wheel, or rear wheel drive. For the all wheel drive variant, you can configure the type of coupling torque.
	Front Wheel Drive (default)	
	Rear Wheel Drive	
Vehicle	Vehicle Body 3 DOF Longitudinal	Configured for 3 degrees of freedom
Wheels and Brakes	All Wheel Drive	<p>For the wheels, you can configure the type of:</p> <ul style="list-style-type: none"> <li>• Brake</li> <li>• Force calculation</li> <li>• Resistance calculation</li> <li>• Vertical motion</li> </ul> <p>For performance and clarity, to determine the longitudinal force of each wheel, the variants implement the Longitudinal Wheel block. To determine the <i>total</i> longitudinal force of all wheels acting on the axle, the variants use a scale factor to multiply the force of one wheel by the number of wheels on the axle. By using this approach to calculate the total force, the variants assume equal tire slip and loading at the front and rear axles, which is common for longitudinal powertrain studies. If this is not the case, for example when friction or loads differ on the left and right sides of the axles, use unique Longitudinal Wheel blocks to calculate independent forces. However, using</p>
	Front Wheel Drive (default)	

Drivetrain Subsystem	Variant	Description
	Rear Wheel Drive	unique blocks to model each wheel increases model complexity and computational cost.

Electric Plant Subsystem	Variant	Description
Battery	BattHevP2	Configured with Lithium Ion battery
Motor	MotMapped (default)	Mapped motor with implicit controller
	MotDynamic	Interior permanent magnet synchronous motor (PMSM) with controller
Starter Motor and DC-DC Converter	StarterSystemP2	Low voltage starter motor and DC-DC converter

Engine Subsystem	Variant	Description
Engine	SiEngine	Dynamic SI engine
	SiMappedEngine (default)	Mapped SI engine
	CiEngine	Dynamic CI engine with turbocharger
	CiMappedEngine	Mapped CI engine with implicit turbocharger

## References

- [1] Balazs, A., Morra, E., and Pischinger, S., *Optimization of Electrified Powertrains for City Cars*. SAE Technical Paper 2011-01-2451. Warrendale, PA: SAE International Journal of Alternative Powertrains, 2012.

## See Also

CI Controller | CI Core Engine | Datasheet Battery | Drive Cycle Source | Interior PM Controller | Interior PMSM | Longitudinal Driver | Mapped CI Engine | Mapped SI Engine | SI Controller | SI Core Engine

## **Related Examples**

- “Explore the Hybrid Electric Vehicle Input Power-Split Reference Application” on page 3-34
- “Explore the Hybrid Electric Vehicle Multimode Reference Application” on page 3-20
- “Explore the Electric Vehicle Reference Application” on page 3-28

## **More About**

- “CI Engine Project Template” on page 4-2
- “SI Engine Project Template” on page 4-5
- “Variant Systems” (Simulink)

# Resize the CI Engine

By default, the compression-ignition (CI) engine dynamometer reference application engine is configured with a dynamic 1.5-L turbocharged diesel engine. Based on a desired number of cylinders and maximum engine power or engine displacement, you can resize the dynamic engine (CiEngine) for different vehicle applications.

To resize the engine, use the dynamometer reference application. After you open the reference application, click **Resize Engine and Recalibrate Controller**. In the dialog box, set **Power or displacement** to either:

- Power - Enter a **Desired maximum power** value
- Displacement - Enter a **Desired displacement** value

For either power or displacement, enter a **Desired number of cylinders** value.

After you apply the changes, the reference application:

- Resizes the dynamic engine and engine calibration parameters. The **Recalibrate Engine** dialog box provides the updated engine performance characteristics based on the resized calibration parameters.
- Recalibrates the controller and mapped engine model to match the resized dynamic engine.

You can use the variants in other applications, for example, in vehicle projects that require a larger engine model.

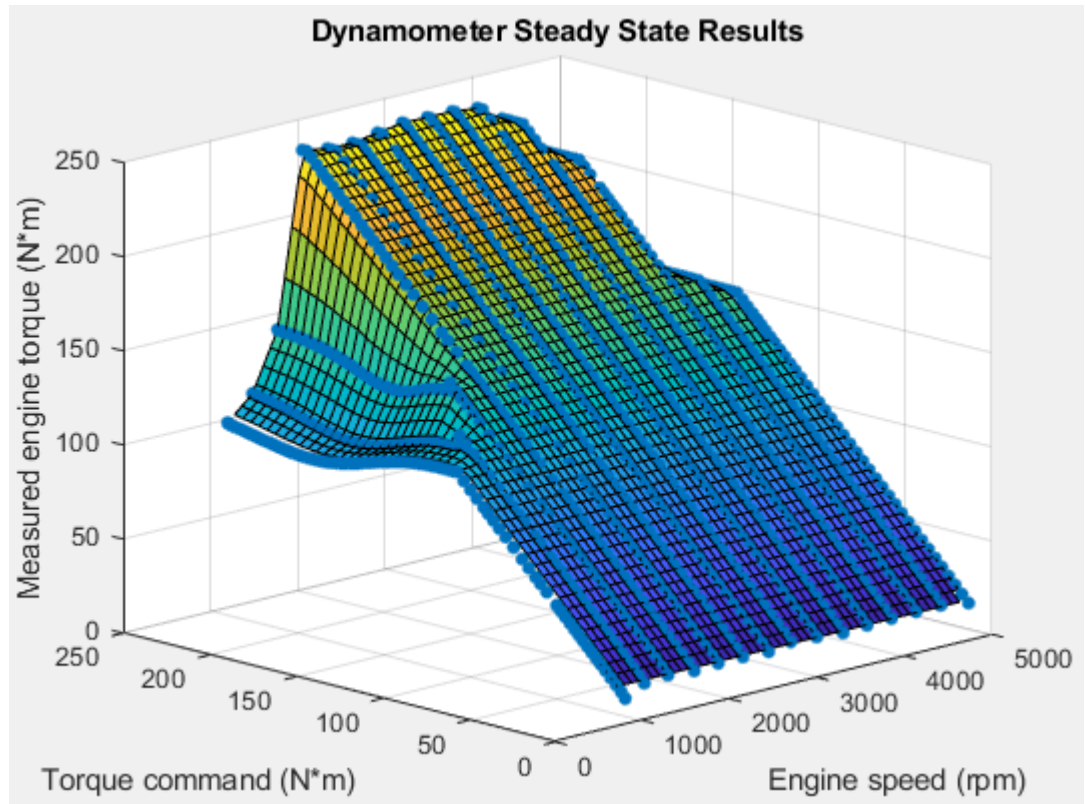
## Create CI Engine Models with Twice the Power

- 1 If it is not already open, open a copy of the CI engine reference application project by entering  

```
autoblkCIDynamometerStart
```
- 2 In the CiDynReferenceApplication model window, click **Recalibrate Controller**.

The reference application performs a dynamometer test to calibrate the engine controller for the default 1.5-L turbocharged diesel engine. For engine speeds 2000–5000 rpm, the measured engine torque approaches 240 N·m. The steady-state results for measured engine torque as a function of torque command and engine speed are similar to this plot.





- 3 In the CiDynReferenceApplication model window, click **Resize Engine and Recalibrate Controller**.

The dialog box opens with default values for **Desired maximum power** and **Desired number of cylinders**. These values represent the calibration parameters for the default 1.5-L dynamic engine.

The dialog box provides the calibration parameters for the current engine design. The parameters are similar to these.

Current Engine Design	
Maximum power [kW]:	109.307
Number of cylinders:	4
Engine displacement [L]:	1.5
Idle speed [rpm]:	750
Speed for maximum torque [rpm]:	3250
Maximum torque [Nm]:	264
Power for best fuel [kW]:	71.2
Speed for best fuel [rpm]:	3250
Torque for best fuel [Nm]:	209.2
BSFC for best fuel [g/kWh]:	215.2
Speed for maximum power [rpm]:	4000
Torque for maximum power [Nm]:	261
Throttle bore diameter [mm]:	50
Intake manifold volume [L]:	2.86
Exhaust manifold volume [L]:	0.7
Compressor out volume [L]:	2.6
Maximum turbo speed [rpm]:	237952.67
Turbo rotor inertia [kg*m <sup>2</sup> ]:	0.006
Fuel injector slope [mg/ms]:	6.45

- 4 In the **Resize Engine and Recalibrate Controller** dialog box, enter values that represent approximately twice the maximum power and number of cylinders. For example, set:
- **Desired maximum power** to 220.
  - **Desired number of cylinders** to 8.

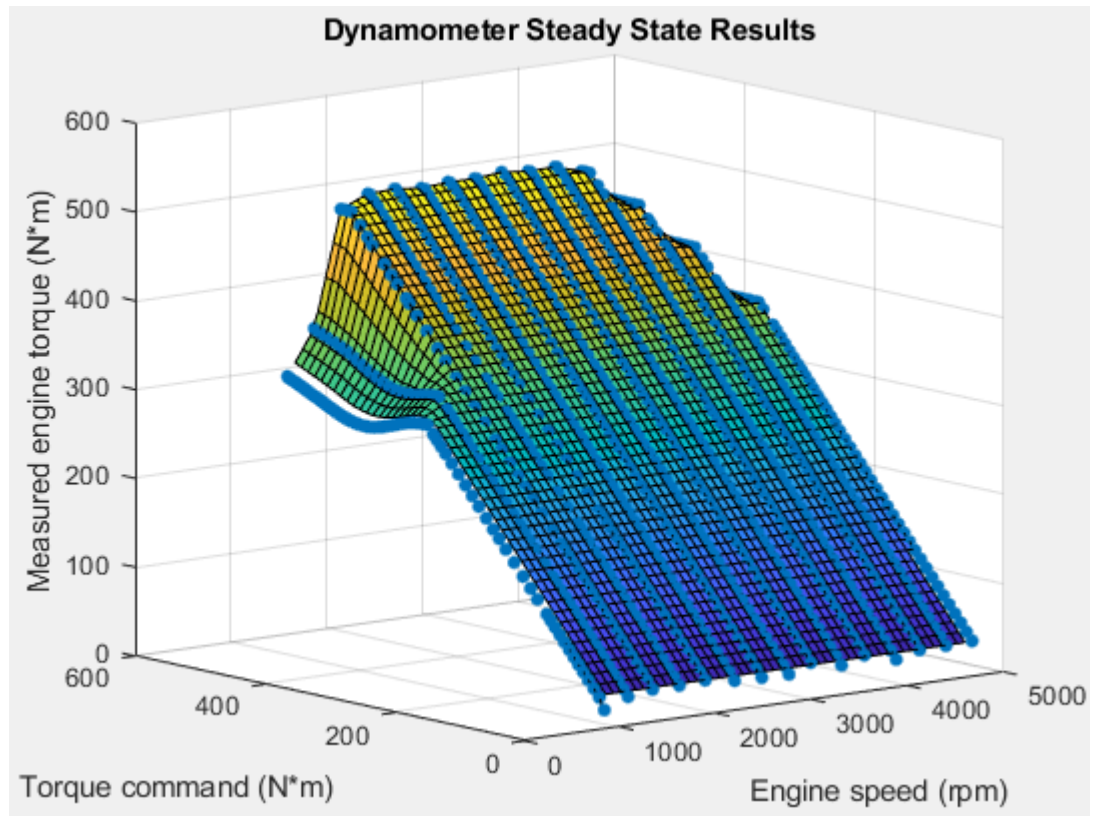
Click **Resize Engine**. The reference application:

- Resizes the dynamic engine (CiEngineCore) and engine calibration parameters. The dialog box provides the updated engine performance characteristics based on the resized calibration parameters.
- Recalibrates the controller (CiEngineController) and mapped engine model (CiMappedEngine) to match the resized dynamic engine (CiEngineCore).

After resizing and recalibration, the dialog box provides the calibration parameters for the resized engine. The parameters are similar to these.

Current Engine Design	
Maximum power [kW]:	220.0001
Number of cylinders:	8
Engine displacement [L]:	3.03
Idle speed [rpm]:	748
Speed for maximum torque [rpm]:	3240
Maximum torque [Nm]:	533
Power for best fuel [kW]:	143.3
Speed for best fuel [rpm]:	3240
Torque for best fuel [Nm]:	422.4
BSFC for best fuel [g/kWh]:	215.2
Speed for maximum power [rpm]:	3987
Torque for maximum power [Nm]:	526.9
Throttle bore diameter [mm]:	50
Intake manifold volume [L]:	5.77
Exhaust manifold volume [L]:	1.41
Compressor out volume [L]:	2.6
Maximum turbo speed [rpm]:	167727.1
Turbo rotor inertia [kg*m <sup>2</sup> ]:	0.012
Fuel injector slope [mg/ms]:	6.51

- 5 Examine the dynamometer steady-state results. For engine speeds 2000–5000 rpm, the measured engine torque approaches 500 N·m. This result is approximately twice the power of the default dynamic engine. The steady-state results for measured engine torque as a function of torque command and engine speed are similar to this plot.



- 6 To save the engine controller, resized engine mapped variant, and resized dynamic engine variant, in the CiDynReferenceApplication model window, save the reference application.

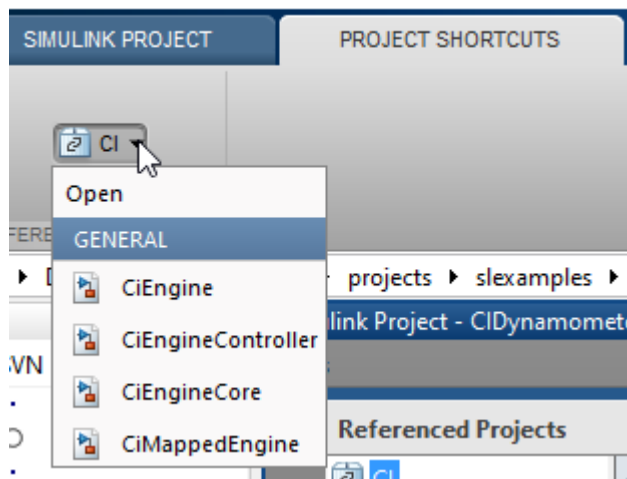
By default, this process creates:

- An updated CI engine controller
- Two engine variants — mapped and dynamic

To see the parameters associated with the controller and engine variants:

1 In MATLAB, use the **PROJECT SHORTCUTS** tab to open these models:

- CiEngineController
- CiMappedEngine
- CiEngineCore

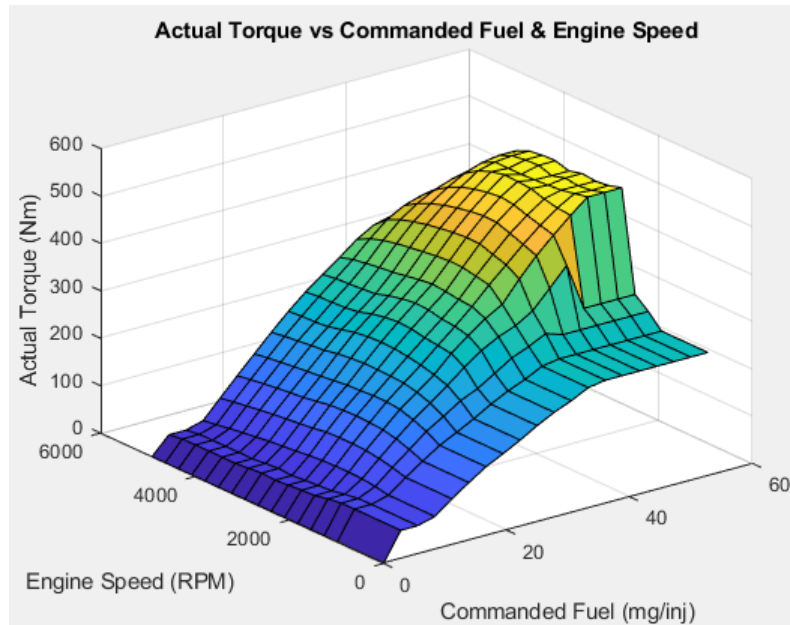


2 Use the Model Explorer to view the resized parameters:

Engine Model	Model Explorer														
Controller — CiEngineController	<div style="display: flex; justify-content: space-between;"> <div style="width: 60%;"> <p>Simulink Root</p> <ul style="list-style-type: none"> <li>Base Workspace</li> <li>CiDynoReferenceApplication*</li> <li>CiEngineCore</li> <li>CiEngineController                             <ul style="list-style-type: none"> <li>Model Workspace</li> <li>Configuration (Active)</li> <li>Reference</li> <li>Start Stop Logic</li> </ul> </li> <li>CiMappedEngine</li> </ul> </div> <div style="width: 35%;"> <p>Column View: Data Objects <a href="#">Show Details</a></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td><input checked="" type="checkbox"/> EngStopStartEnable</td> <td>true</td> </tr> <tr> <td><input type="checkbox"/> EngStopTime</td> <td>5</td> </tr> <tr> <td><input type="checkbox"/> NCyl</td> <td>8</td> </tr> <tr> <td><input type="checkbox"/> Pstd</td> <td>101325</td> </tr> <tr> <td><input type="checkbox"/> Rair</td> <td>287.05</td> </tr> <tr> <td><input type="checkbox"/> Sinj</td> <td>6.513064320203665</td> </tr> </tbody> </table> </div> </div>	Name	Value	<input checked="" type="checkbox"/> EngStopStartEnable	true	<input type="checkbox"/> EngStopTime	5	<input type="checkbox"/> NCyl	8	<input type="checkbox"/> Pstd	101325	<input type="checkbox"/> Rair	287.05	<input type="checkbox"/> Sinj	6.513064320203665
Name	Value														
<input checked="" type="checkbox"/> EngStopStartEnable	true														
<input type="checkbox"/> EngStopTime	5														
<input type="checkbox"/> NCyl	8														
<input type="checkbox"/> Pstd	101325														
<input type="checkbox"/> Rair	287.05														
<input type="checkbox"/> Sinj	6.513064320203665														

Engine Model	Model Explorer																		
Mapped — CiMappedEngine	<table border="1"> <caption>Data Objects</caption> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>AccPwrbp</td> <td>[0 4.025359766529133]</td> </tr> <tr> <td>AccSpdbp</td> <td>[299.0535069180764 747.633767]</td> </tr> <tr> <td>Cps</td> <td>2</td> </tr> <tr> <td>NCyl</td> <td>8</td> </tr> <tr> <td>Pstd</td> <td>101325</td> </tr> <tr> <td>Rair</td> <td>287.05</td> </tr> <tr> <td>Sinj</td> <td>6.451612903225807</td> </tr> </tbody> </table>	Name	Value	AccPwrbp	[0 4.025359766529133]	AccSpdbp	[299.0535069180764 747.633767]	Cps	2	NCyl	8	Pstd	101325	Rair	287.05	Sinj	6.451612903225807		
Name	Value																		
AccPwrbp	[0 4.025359766529133]																		
AccSpdbp	[299.0535069180764 747.633767]																		
Cps	2																		
NCyl	8																		
Pstd	101325																		
Rair	287.05																		
Sinj	6.451612903225807																		
Dynamic — CiEngineCore	<table border="1"> <caption>Data Objects</caption> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>AirFilterArea</td> <td>0.00193642497768315</td> </tr> <tr> <td>AirIntakeVol</td> <td>0.00363428989067364</td> </tr> <tr> <td>C_eng</td> <td>40000</td> </tr> <tr> <td>CompEff</td> <td>&lt;60x100 double&gt;</td> </tr> <tr> <td>CompMassFlwRate</td> <td>&lt;60x100 double&gt;</td> </tr> <tr> <td>CompPrsRatioBreakPoints</td> <td>[1 1.028080808080808 1.056161616...]</td> </tr> <tr> <td>CompRefPrs</td> <td>101325</td> </tr> <tr> <td>CompRefTemp</td> <td>300</td> </tr> </tbody> </table>	Name	Value	AirFilterArea	0.00193642497768315	AirIntakeVol	0.00363428989067364	C_eng	40000	CompEff	<60x100 double>	CompMassFlwRate	<60x100 double>	CompPrsRatioBreakPoints	[1 1.028080808080808 1.056161616...]	CompRefPrs	101325	CompRefTemp	300
Name	Value																		
AirFilterArea	0.00193642497768315																		
AirIntakeVol	0.00363428989067364																		
C_eng	40000																		
CompEff	<60x100 double>																		
CompMassFlwRate	<60x100 double>																		
CompPrsRatioBreakPoints	[1 1.028080808080808 1.056161616...]																		
CompRefPrs	101325																		
CompRefTemp	300																		

- 3 In the CiDynoReferencApplication > Engine System > Engine Plant > Engine > CIMappedEngine subsystem, open the Mapped CI Engine block. On the Power tab, plot the actual torque as a function of engine speed and commanded fuel.



## See Also

CI Core Engine | Mapped CI Engine

## More About

- "Explore the CI Engine Dynamometer Reference Application" on page 3-10



## Resize the SI Engine

By default, the spark-ignition (SI) engine dynamometer reference application engine is configured with a 1.5-L dynamic gasoline engine. Based on a desired number of cylinders and maximum engine power or engine displacement, you can resize the dynamic engine (SiEngineCore) for different vehicle applications.

To resize the engine, use the dynamometer reference application. After you open the reference application, click **Resize Engine and Recalibrate Controller**. In the dialog box, set **Power or displacement** to either:

- Power - Enter a **Desired maximum power** value
- Displacement - Enter a **Desired displacement** value

For either power or displacement, enter a **Desired number of cylinders** value.

After you apply the changes, the reference application:

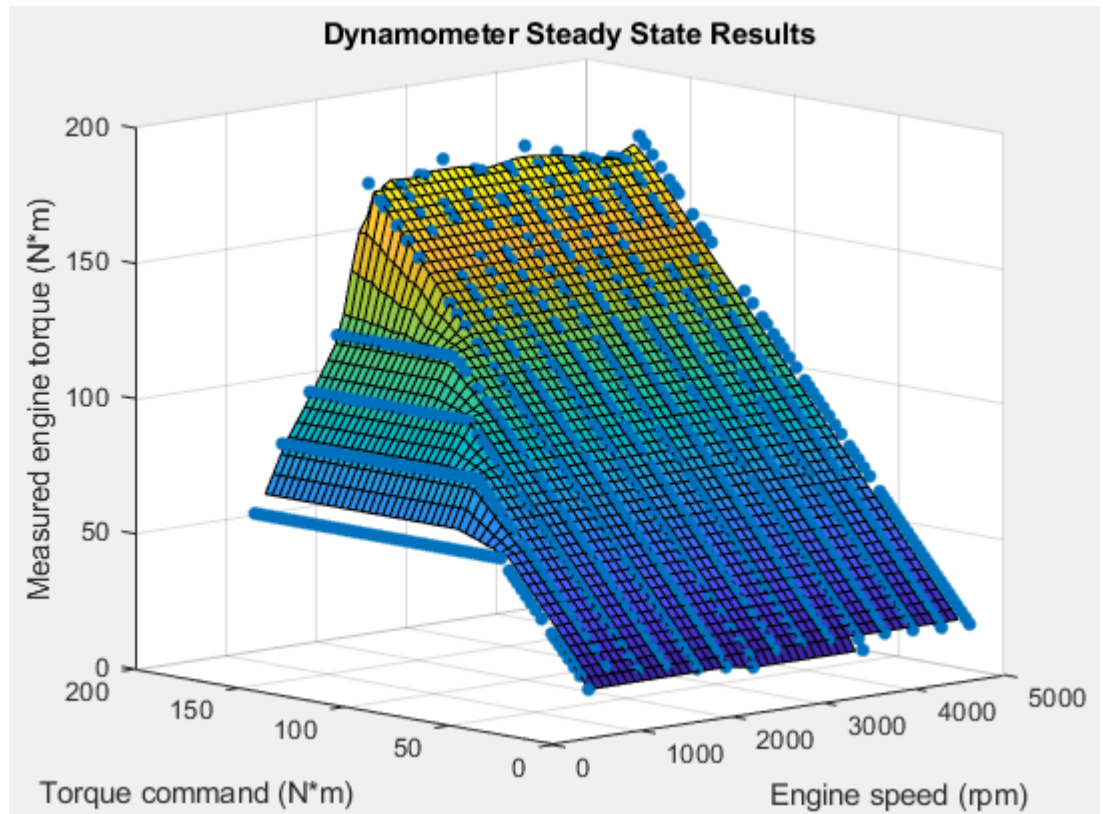
- Resizes the dynamic engine and engine calibration parameters. The **Recalibrate Engine** dialog box provides the updated engine performance characteristics based on the resized calibration parameters.
- Recalibrates the controller and mapped engine model to match the resized dynamic engine.

You can use the variants in other applications, for example, in vehicle projects that require a larger engine model.

### Create SI Engine Models with Twice the Power

- 1 If it is not already open, open a copy of the SI engine reference application project by entering  
`autoblksIDynamometerStart`
- 2 In the `SiDynReferenceApplication` model window, click **Recalibrate Controller**.

The reference application performs a dynamometer test to calibrate the engine controller for the default 1.5-L dynamic engine. For engine speeds 2000–5000 rpm, the measured engine torque approaches 180 N·m. The steady-state results for measured engine torque as a function of torque command and engine speed are similar to this plot.



- 3 In the SiDynReferenceApplication model window, click **Resize Engine and Recalibrate Controller**.

The dialog box opens with default values for **Desired maximum power** and **Desired number of cylinders**. These values represent the calibration parameters for the default 1.5-L dynamic engine.

The dialog box provides the calibration parameters for the current engine design. The parameters are similar to these.

Current Engine Design	
Maximum power [kW]:	115.0917
Number of cylinders:	4
Engine displacement [L]:	1.5
Idle speed [rpm]:	750
Speed for maximum torque [rpm]:	2571
Maximum torque [Nm]:	230.6
Power for best fuel [kW]:	41.1
Speed for best fuel [rpm]:	2571
Torque for best fuel [Nm]:	152.7
BSFC for best fuel [g/kWh]:	225.9
Speed for maximum power [rpm]:	5000
Torque for maximum power [Nm]:	219.8
Throttle bore diameter [mm]:	50
Intake manifold volume [L]:	2.86
Exhaust manifold volume [L]:	1.6
Compressor out volume [L]:	2.6
Maximum turbo speed [rpm]:	232000
Turbo rotor inertia [kg*m <sup>2</sup> ]:	0.016
Fuel injector slope [mg/ms]:	6.45

- 4 In the **Resize Engine and Recalibrate Controller** dialog box, enter values that represent approximately twice the maximum power and number of cylinders. For example, set:
- **Desired maximum power** to 230.
  - **Desired number of cylinders** to 8.

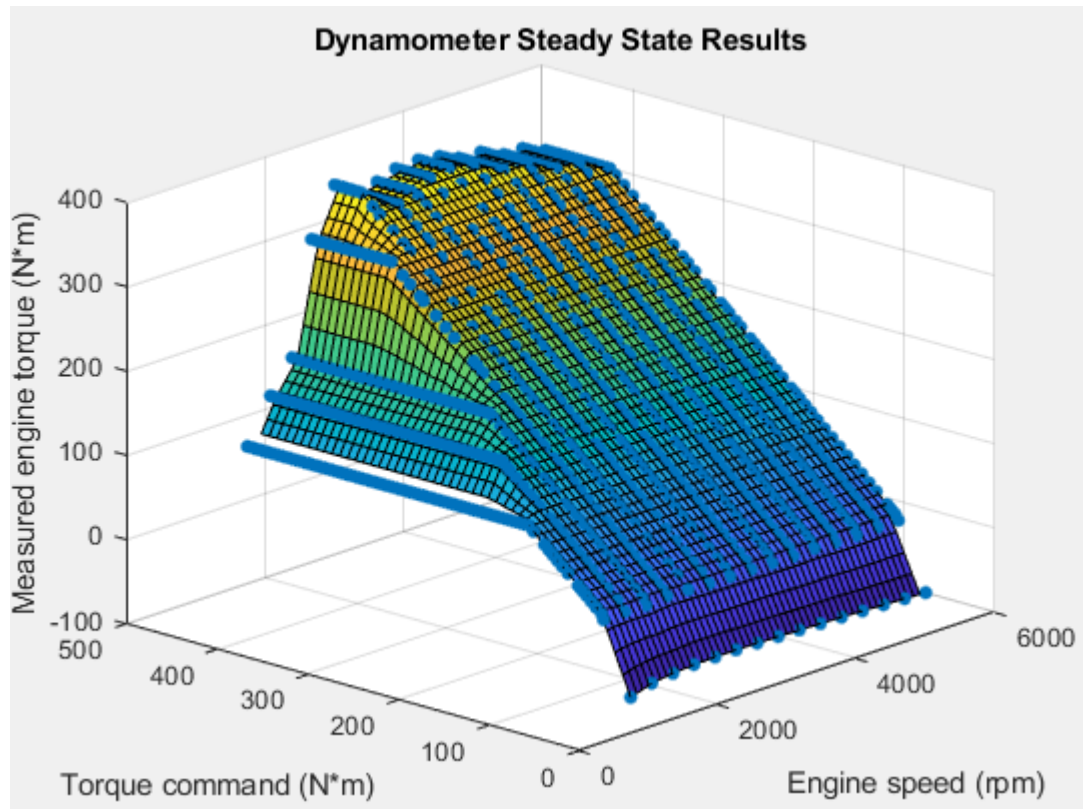
Click **Resize Engine**. The reference application:

- Resizes the dynamic engine (SiEngineCore) and engine calibration parameters. The **Recalibrate Engine** dialog box provides the updated engine performance characteristics based on the resized calibration parameters.
- Recalibrates the controller (SiEngineController) and mapped engine model (SiMappedEngine) to match the resized dynamic engine (SiEngineCore).

After resizing and recalibration, the dialog box provides the calibration parameters for the resized engine. The parameters are similar to these.

Current Engine Design	
Maximum power [kW]:	230
Number of cylinders:	8
Engine displacement [L]:	3
Idle speed [rpm]:	750
Speed for maximum torque [rpm]:	2572
Maximum torque [Nm]:	460.6
Power for best fuel [kW]:	82.1
Speed for best fuel [rpm]:	2572
Torque for best fuel [Nm]:	304.9
BSFC for best fuel [g/kWh]:	225.9
Speed for maximum power [rpm]:	5002
Torque for maximum power [Nm]:	439.1
Throttle bore diameter [mm]:	70.7
Intake manifold volume [L]:	5.71
Exhaust manifold volume [L]:	3.2
Compressor out volume [L]:	5.19
Maximum turbo speed [rpm]:	164114.17
Turbo rotor inertia [kg*m <sup>2</sup> ]:	0.031
Fuel injector slope [mg/ms]:	6.44

- 5 Examine the dynamometer steady-state results. For engine speeds 2000–5000 rpm, the measured engine torque approaches 350 N·m. This result is approximately twice the power of the default dynamic engine. The steady-state results for measured engine torque as a function of torque command and engine speed are similar to this plot.



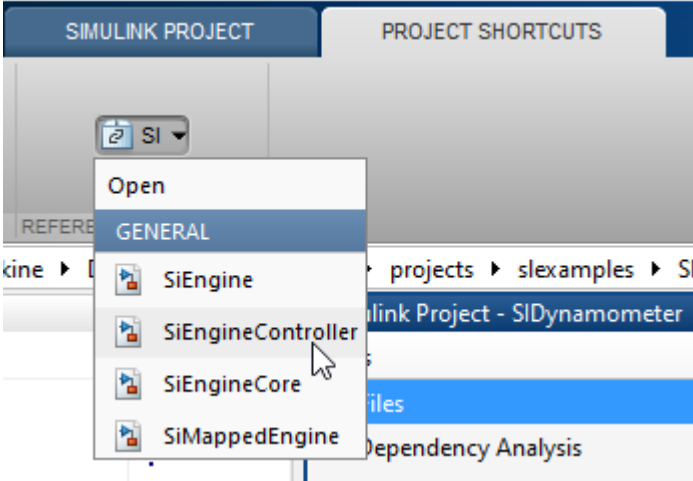
- 6 To save the engine controller, resized engine mapped variant, and resized dynamic engine variant, in the SiDynReferenceApplication model window, save the reference application.

By default, this process creates:

- An updated SI engine controller
- Two engine variants — mapped and dynamic

To see the parameters associated with the controller and engine variants:

- 1 In MATLAB, use the **PROJECT SHORTCUTS** tab to open these models:
  - SiEngineController
  - SiMappedEngine
  - SiEngineCore



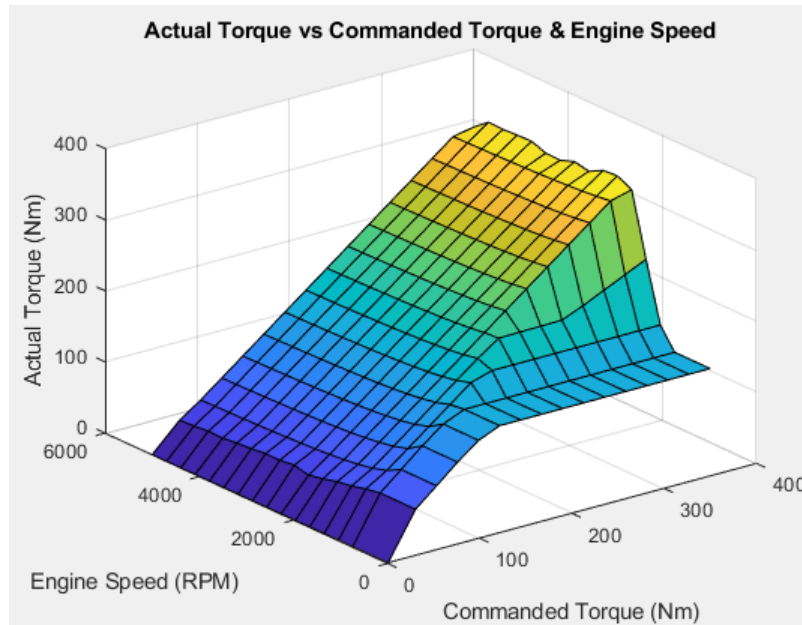
- 2 Use the Model Explorer to view the resized parameters:

Engine Model	Model Explorer														
Controller — SiEngineController	<div style="display: flex;"><div style="flex: 1;"><ul style="list-style-type: none"><li>Simulink Root<ul style="list-style-type: none"><li>Base Workspace</li><li>SiDynoReferenceApplication*</li><li>SiEngineCore</li><li>SiEngineController<ul style="list-style-type: none"><li>Model Workspace</li><li>Configuration (Active)</li><li>Reference</li><li>Start Stop Logic</li></ul></li><li>SiMappedEngine</li></ul></li></ul></div><div style="flex: 2;"><p>Column View: Data Objects <a href="#">Show Details</a></p><table border="1"><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td><input checked="" type="checkbox"/> EngStopStartEnable</td><td>true</td></tr><tr><td><input type="checkbox"/> EngStopTime</td><td>5</td></tr><tr><td><input type="checkbox"/> NCyl</td><td>8</td></tr><tr><td><input type="checkbox"/> N_idle</td><td>750.298962153551</td></tr><tr><td><input type="checkbox"/> Pstd</td><td>101325</td></tr><tr><td><input type="checkbox"/> Rair</td><td>287</td></tr></tbody></table></div></div>	Name	Value	<input checked="" type="checkbox"/> EngStopStartEnable	true	<input type="checkbox"/> EngStopTime	5	<input type="checkbox"/> NCyl	8	<input type="checkbox"/> N_idle	750.298962153551	<input type="checkbox"/> Pstd	101325	<input type="checkbox"/> Rair	287
Name	Value														
<input checked="" type="checkbox"/> EngStopStartEnable	true														
<input type="checkbox"/> EngStopTime	5														
<input type="checkbox"/> NCyl	8														
<input type="checkbox"/> N_idle	750.298962153551														
<input type="checkbox"/> Pstd	101325														
<input type="checkbox"/> Rair	287														

Engine Model	Model Explorer																	
Mapped — SiMappedEngine		<p>Column View: Data Objects <a href="#">Show Details</a></p> <table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>AccPwrbp</td> <td>[0 2.997609732065822]</td> </tr> <tr> <td>AccSpdbp</td> <td>[300.1195848614204 750.2989</td> </tr> <tr> <td>AfrStoich</td> <td>14.6</td> </tr> <tr> <td>Cps</td> <td>2</td> </tr> <tr> <td>NCyl</td> <td>8</td> </tr> <tr> <td>Pstd</td> <td>101325</td> </tr> <tr> <td>Rair</td> <td>287</td> </tr> </tbody> </table>	Name	Value	AccPwrbp	[0 2.997609732065822]	AccSpdbp	[300.1195848614204 750.2989	AfrStoich	14.6	Cps	2	NCyl	8	Pstd	101325	Rair	287
Name	Value																	
AccPwrbp	[0 2.997609732065822]																	
AccSpdbp	[300.1195848614204 750.2989																	
AfrStoich	14.6																	
Cps	2																	
NCyl	8																	
Pstd	101325																	
Rair	287																	
Dynamic — SiEngineCore		<p>Column View: Data Objects <a href="#">Show Details</a></p> <table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>CompRefPrs</td> <td>101325</td> </tr> <tr> <td>CompRefTemp</td> <td>300</td> </tr> <tr> <td>CompSpdBreakPoints</td> <td>[3.536943227040031 354.199357</td> </tr> <tr> <td>CoolantTemp</td> <td>310</td> </tr> <tr> <td>Cps</td> <td>2</td> </tr> <tr> <td>EgrArea</td> <td>0.000627817914172484</td> </tr> <tr> <td>EgrCoolantTemp</td> <td>360</td> </tr> </tbody> </table>	Name	Value	CompRefPrs	101325	CompRefTemp	300	CompSpdBreakPoints	[3.536943227040031 354.199357	CoolantTemp	310	Cps	2	EgrArea	0.000627817914172484	EgrCoolantTemp	360
Name	Value																	
CompRefPrs	101325																	
CompRefTemp	300																	
CompSpdBreakPoints	[3.536943227040031 354.199357																	
CoolantTemp	310																	
Cps	2																	
EgrArea	0.000627817914172484																	
EgrCoolantTemp	360																	

- 3 In the SiDynoReferencApplication > Engine System > Engine Plant > Engine > SIMappedEngine subsystem, open the Mapped SI Engine block. On the Power tab, plot the actual torque as a function of engine speed and commanded torque.





## See Also

Mapped SI Engine | SI Core Engine

## More About

- "Explore the SI Engine Dynamometer Reference Application" on page 3-15

## Generate Mapped CI Engine from a Spreadsheet

If you have Model-Based Calibration Toolbox and Stateflow, you can use the engine dynamometer reference application to generate lookup tables for the Mapped CI Engine block. The reference application uses engine data to calibrate the engine and generate the tables.

- 1 If it is not opened, open the reference application.

autoblkCIDynamometerStart

- 2 Click **Generate Mapped Engine from Spreadsheet**.

### Step 1: Generate Mapped Engine Calibration

- 1 Use the **Spreadsheet file** field to provide a data file. By default, the reference application has `CiEngineData.xlsx` containing required and optional data. The tables summarize the data file requirements for generating calibrated tables that are functions of either injected fuel mass or engine torque and engine speed.

---

**Note** To specify the lookup table type, in the Mapped CI Engine block, set the **Input command** parameter to `Fuel mass` or `Torque`.

---

Firing data contains data collected at different engine torques and speeds.

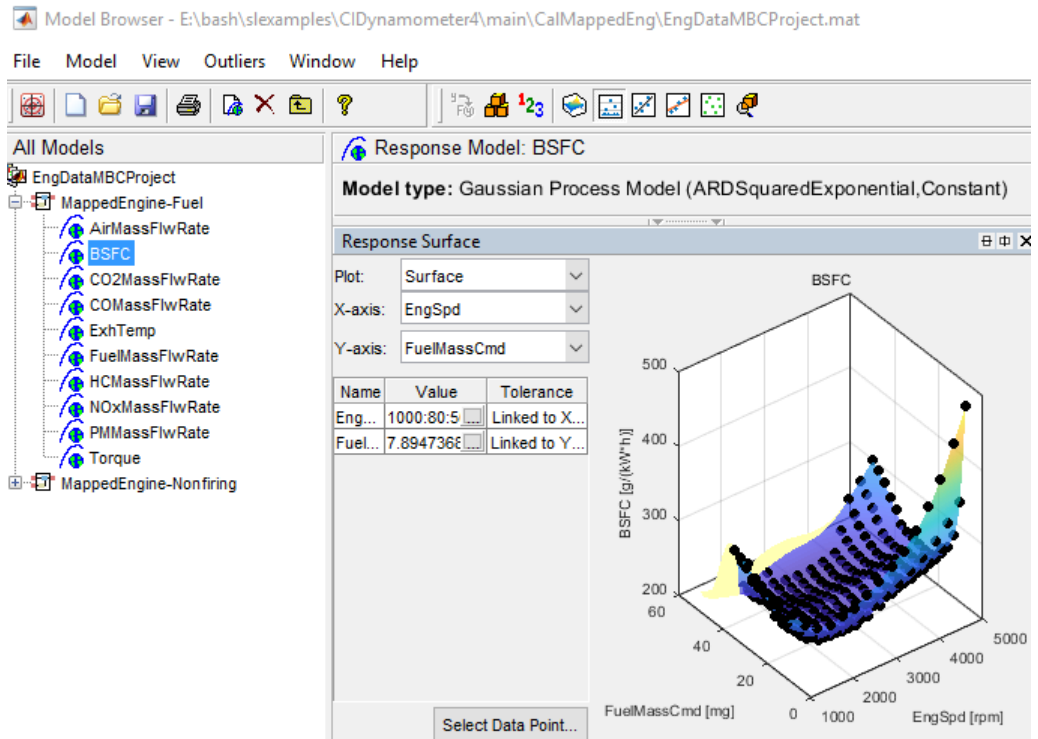
Firing Data	Description	Data Requirements for Generating Mapped Engine Tables	
		Function of Fuel Mass and Engine Speed	Function of Torque and Engine Speed
FuelMassCmd	Injected fuel mass, in mg per injection	<i>Required</i>	<i>Not used</i>
Torque	Engine torque command, in N·m	<i>Required</i>	<i>Required</i>
EngSpd	Engine speed, in rpm	<i>Required</i>	<i>Required</i>
AirMassFlowRate	Air mass flow, in kg/s	<i>Optional</i>	<i>Optional</i>

Firing Data	Description	Data Requirements for Generating Mapped Engine Tables	
		Function of Fuel Mass and Engine Speed	Function of Torque and Engine Speed
FuelMassFlwRate	Fuel mass flow, in kg/s	<i>Optional</i>	<i>Optional</i>
ExhTemp	Exhaust temperature, in K	<i>Optional</i>	<i>Optional</i>
BSFC	Engine brake-specific fuel consumption (BSFC), in g/kWh	<i>Optional</i>	<i>Optional</i>
HCMassFlwRate	Hydrocarbon emission mass flow, in kg/s	<i>Optional</i>	<i>Optional</i>
COMassFlwRate	Carbon monoxide emission mass flow, in kg/s	<i>Optional</i>	<i>Optional</i>
NOxMassFlwRate	Nitric oxide and nitrogen dioxide emissions mass flow, in kg/s	<i>Optional</i>	<i>Optional</i>
CO2MassFlwRate	Carbon dioxide emission mass flow, in kg/s	<i>Optional</i>	<i>Optional</i>
PMMassFlwRate	Particulate matter emission mass flow, in kg/s	<i>Optional</i>	<i>Optional</i>

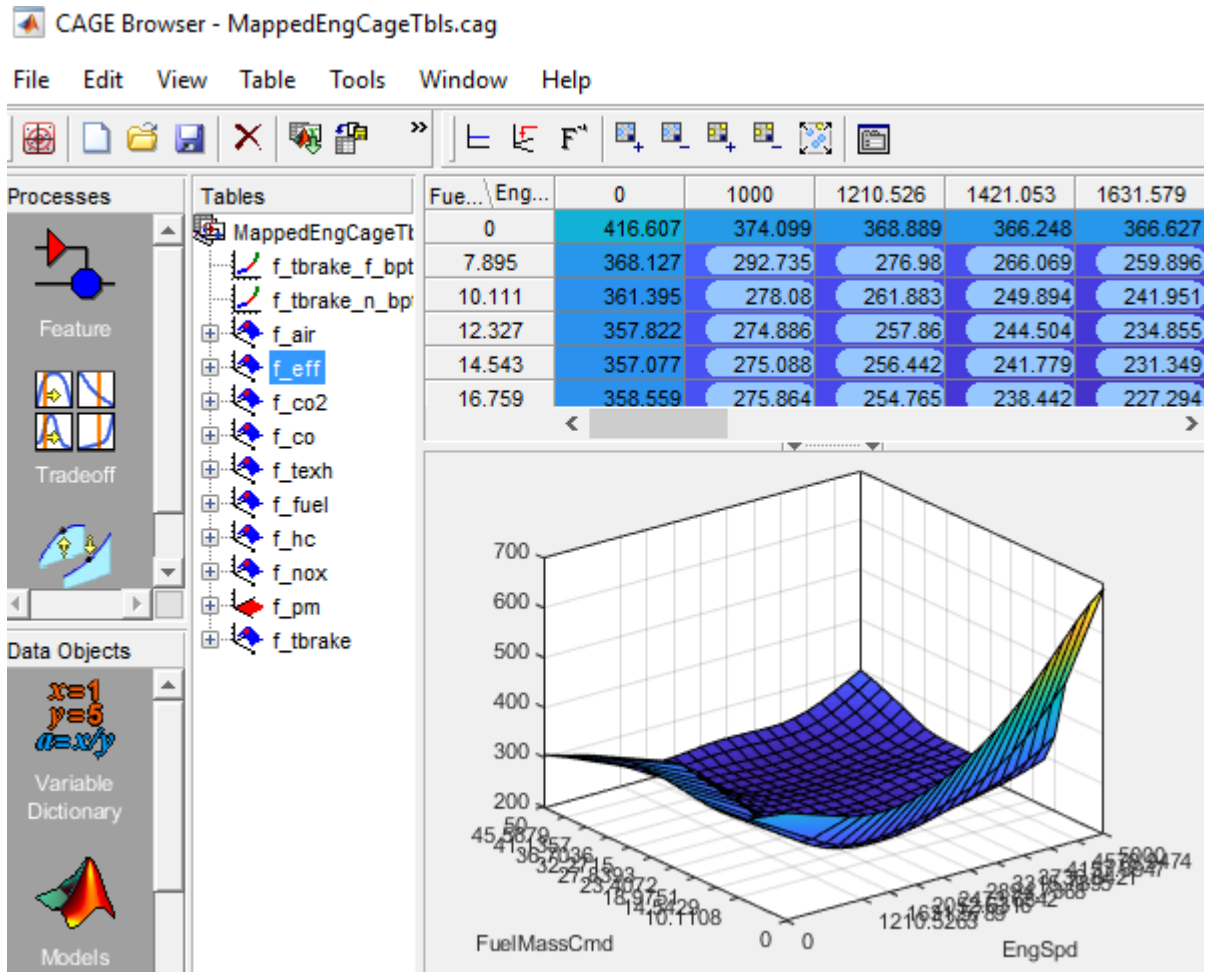
Nonfiring data contains data collected at different engine speeds without fuel consumption.

Nonfiring Data	Description	Data Requirements for Generating Mapped Engine Tables	
		Function of Fuel Mass and Engine Speed	Function of Torque and Engine Speed
FuelMassCmd	Injected fuel mass, in mg per injection	<i>Not used</i>	<i>Not used</i>
Torque	Engine torque command, in N·m	<i>Required</i>	<i>Required</i>
EngSpd	Engine speed, in rpm	<i>Required</i>	<i>Required</i>
AirMassFlwRate	Air mass flow, in kg/s	<i>Optional</i>	<i>Optional</i>

- 2 Click **Generate mapped engine calibration** to generate response surface models in the Model-Based Calibration Toolbox and calibration in CAGE (CALibration GEneration). CAGE and the model browser open when the process completes. To calibrate the data, Model-Based Calibration Toolbox uses templates.
  - The Model Browser provides the response model fits for the data contained in the data file, for example:



- The CAGE Browser provides the calibrated data, for example:



## Step 2: Apply Calibration to Mapped Engine Model

When you click **Apply calibration to mapped engine model**, Powertrain Blockset:

- Updates the Mapped CI Engine block parameters with the calibrated table and breakpoint data.
- Updates the CI Controller with the fuel mass per injection table if the Mapped CI Engine block tables are functions of fuel mass and engine speed.

- Sets the Mapped CI Engine as the active variant.
- Executes the engine mapping experiment.

When the dynamometer engine mapping completes, use the Simulation Data Inspector to verify the results.

## See Also

CI Controller | CI Core Engine | Mapped CI Engine

## More About

- “Explore the CI Engine Dynamometer Reference Application” on page 3-10
- “What Is CAGE?” (Model-Based Calibration Toolbox)
- “Mapped CI Lookup Tables as Functions of Fuel Mass and Engine Speed” (Model-Based Calibration Toolbox)
- “Mapped CI Lookup Tables as Functions of Engine Torque and Speed” (Model-Based Calibration Toolbox)

## Generate Mapped SI Engine from a Spreadsheet

If you have Model-Based Calibration Toolbox and Stateflow, you can use the engine dynamometer reference application to generate lookup tables for the Mapped SI Engine block. The reference application uses engine data to calibrate the engine and generate the tables.

- 1 If it is not opened, open the reference application.

autoblkSIDynamometerStart

- 2 Click **Generate Mapped Engine from Spreadsheet**.

### Step 1: Generate Mapped Engine Calibration

- 1 Use the **Spreadsheet file** field to provide a data file. By default, the reference application has `SiEngineData.xlsx` containing required and optional data. The tables summarize the data file requirements for generating calibrated tables that are functions of either injected fuel mass or engine torque and engine speed.

Firing data contains data collected at different engine torques and speeds.

Firing Data	Description	Data Requirements for Generating Mapped Engine Tables
FuelMassCmd	Injected fuel mass, in mg per injection	<i>Not Used</i>
Torque	Engine torque command, in N·m	<i>Required</i>
EngSpd	Engine speed, in rpm	<i>Required</i>
AirMassFlwRate	Air mass flow, in kg/s	<i>Optional</i>
FuelMassFlwRate	Fuel mass flow, in kg/s	<i>Optional</i>
ExhTemp	Exhaust temperature, in K	<i>Optional</i>
BSFC	Engine brake-specific fuel consumption (BSFC), in g/kWh	<i>Optional</i>

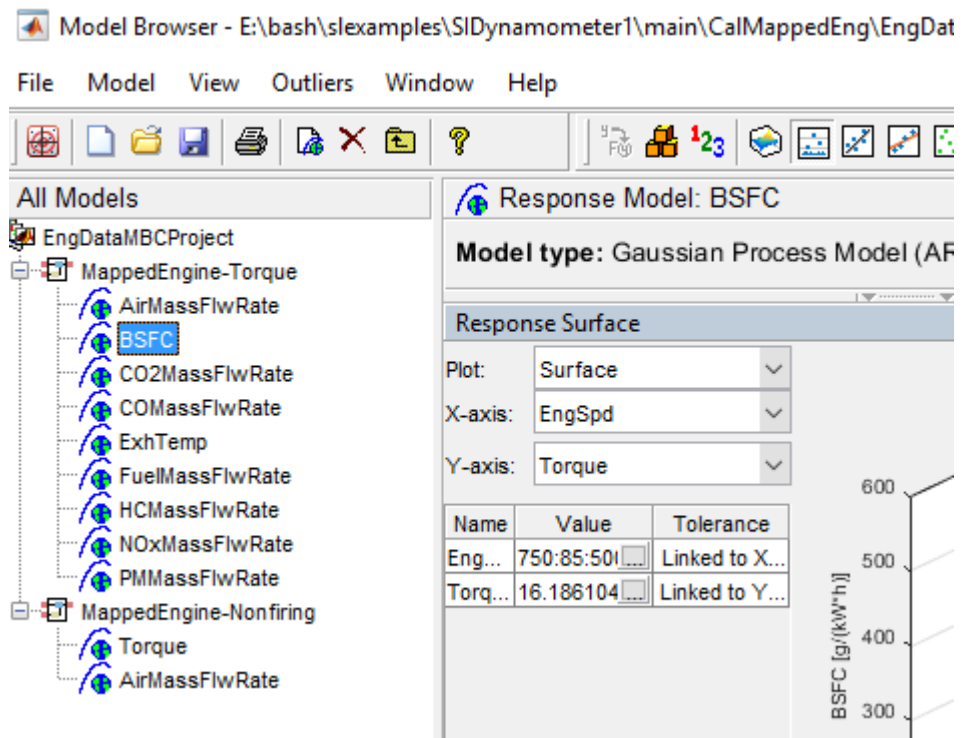


<b>Firing Data</b>	<b>Description</b>	<b>Data Requirements for Generating Mapped Engine Tables</b>
HCMassFlwRate	Hydrocarbon emission mass flow, in kg/s	<i>Optional</i>
COMassFlwRate	Carbon monoxide emission mass flow, in kg/s	<i>Optional</i>
NOxMassFlwRate	Nitric oxide and nitrogen dioxide emissions mass flow, in kg/s	<i>Optional</i>
CO2MassFlwRate	Carbon dioxide emission mass flow, in kg/s	<i>Optional</i>
PMMassFlwRate	Particulate matter emission mass flow, in kg/s	<i>Optional</i>

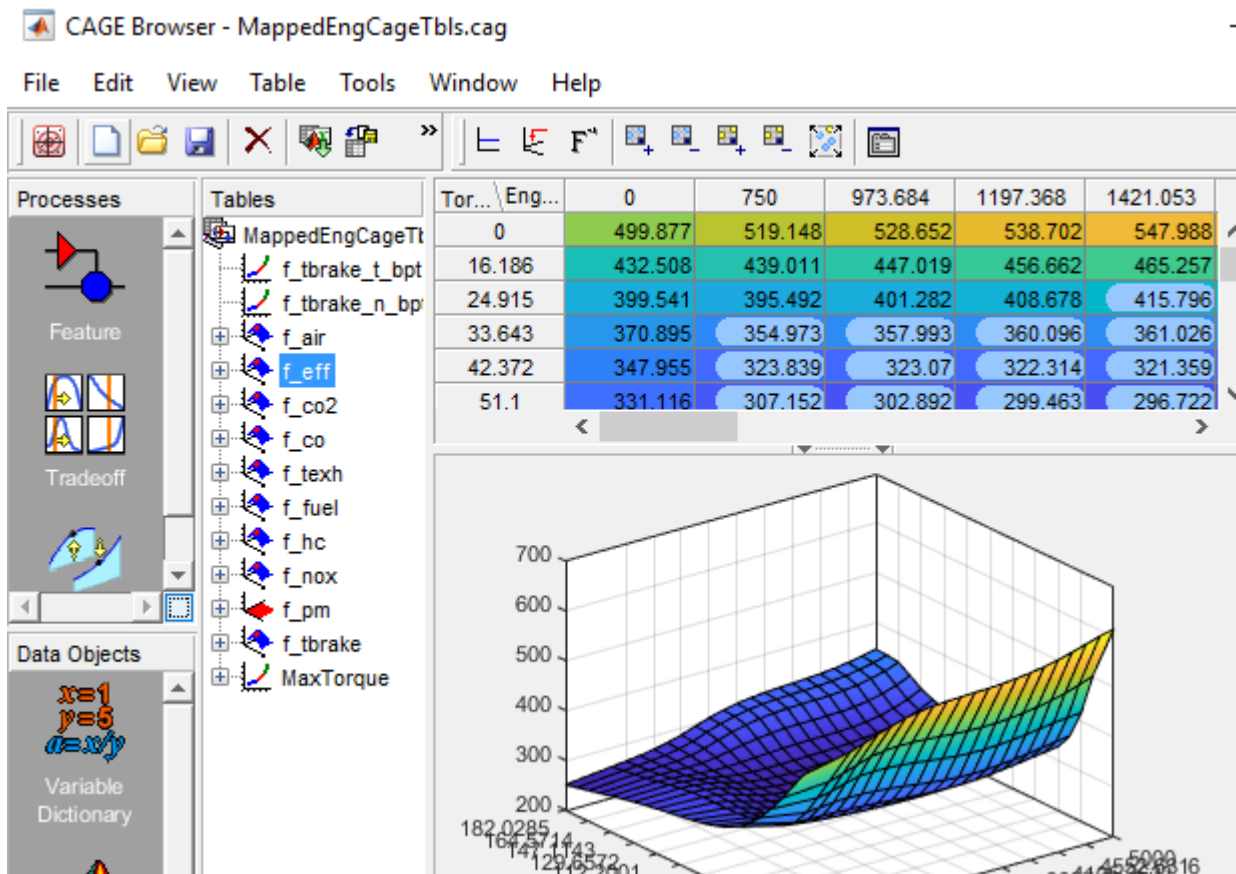
Nonfiring data contains data collected at different engine speeds without fuel consumption.

<b>Nonfiring Data</b>	<b>Description</b>	<b>Data Requirements for Generating Mapped Engine Tables</b>
FuelMassCmd	Injected fuel mass, in mg per injection	<i>Not used</i>
Torque	Engine torque command, in N·m	<i>Required</i>
EngSpd	Engine speed, in rpm	<i>Required</i>
AirMassFlwRate	Air mass flow, in kg/s	<i>Optional</i>

- 2 Click **Generate mapped engine calibration** to generate response surface models in the Model-Based Calibration Toolbox and calibration in CAGE (CALibration GEneration). CAGE and the model browser open when the process completes. To calibrate the data, Model-Based Calibration Toolbox uses templates.
  - The Model Browser provides the response model fits for the data contained in the data file, for example:



- The CAGE Browser provides the calibrated data, for example:



## Step 2: Apply Calibration to Mapped Engine Model

When you click **Apply calibration to mapped engine model**, Powertrain Blockset:

- Updates the Mapped SI Engine block parameters with the calibrated table and breakpoint data.
- Sets the Mapped SI Engine as the active variant.
- Executes the engine mapping experiment.

When the dynamometer engine mapping completes, use the Simulation Data Inspector to verify the results.

### See Also

Mapped SI Engine | SI Core Engine

### More About

- “Explore the SI Engine Dynamometer Reference Application” on page 3-15
- “What Is CAGE?” (Model-Based Calibration Toolbox)
- “Mapped SI Lookup Tables as Functions of Engine Torque and Speed” (Model-Based Calibration Toolbox)

## Internal Combustion Mapped and Dynamic Engine Models

When you customize a SI or CI reference application, you can use either a dynamic or mapped engine model. The table provides considerations for using either implementation.

Type		Implementation	When to Use
Mapped	CiMappedEngine	Model uses a set of steady-state lookup tables to characterize engine performance.	<ul style="list-style-type: none"> <li>• If you have engine data from a dynamometer or a design tool like GT-POWER.</li> <li>• For quasi steady-state engine simulations.</li> </ul>
	SiMappedEngine	The tables provide overall engine characteristics, including actual torque, fuel flow rate, BSFC, and engine-out exhaust emissions.	
Dynamic	CiEngine	Model decomposes the engine behavior into engine characteristics that are separated into lower-level components. By combining components in this way, the models capture the dynamic effects.	<ul style="list-style-type: none"> <li>• If you need a more detailed dynamic model and have component-level data.</li> <li>• To analyze the impact of individual engine components on the overall performance.</li> </ul>
	SiEngine		

### See Also

#### More About

- Mapped CI Engine
- Mapped SI Engine
- CI Core Engine
- SI Core Engine
- “Engine Calibration Maps” on page 2-41



# Project Templates

---

# CI Engine Project Template

The Powertrain Blockset software provides a project template for compression-ignition (CI) engines. Use the template to create engine variants that you can use with the internal combustion engine reference application projects. The project template contains CI engine controller and plant models.

Use the project template to create CI engine variants for these reference applications:

- Conventional vehicle
- Hybrid electric vehicles
- CI engine dynamometer

To open the CI engine project template:

- 1 In Simulink, select **File > New > Project**.

In the Simulink start page, browse to Powertrain Blockset and select **CI Engine**.

- 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 CI or SI engine model.

## Controller

The Controller folder contains the `CiEngineController.slx` model. The model uses the CI Controller block and a Start Stop Logic subsystem to control the CI engine plant model.

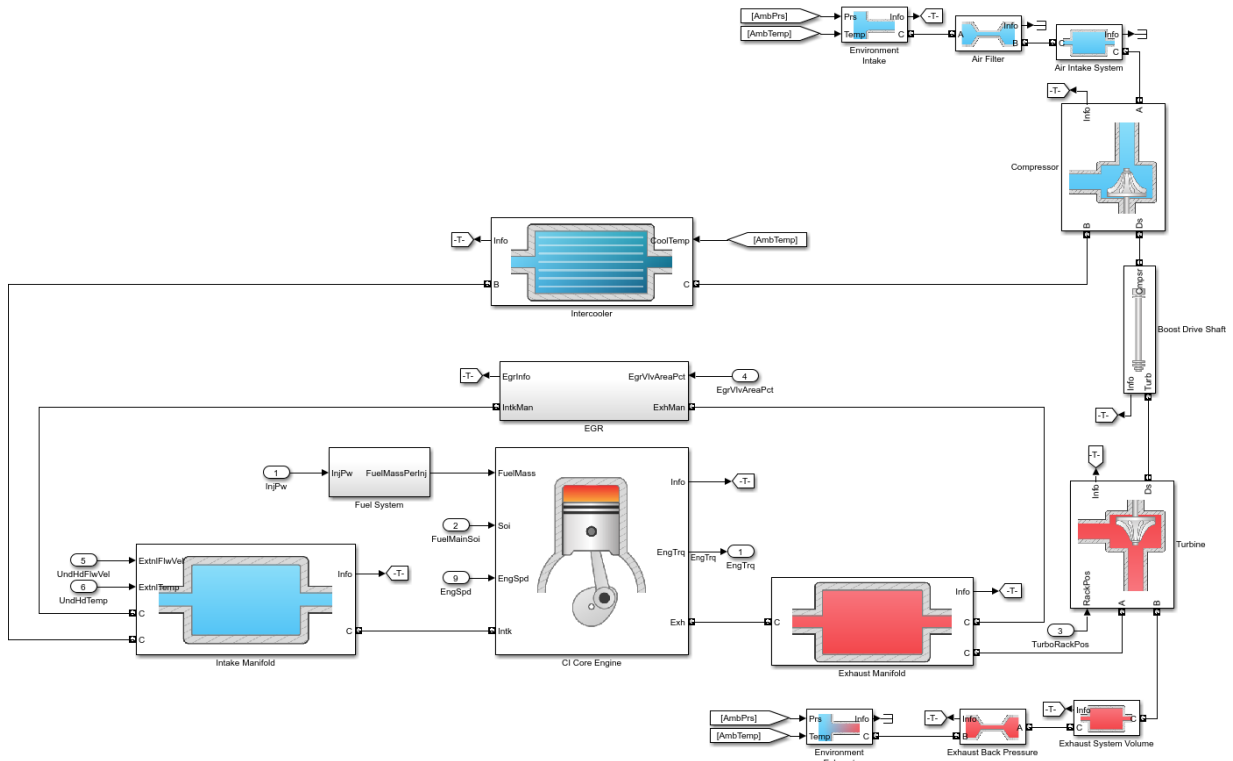
## Plant

The Plant folder contains models that represent dynamic and mapped CI engines. By default, the dynamic and mapped engines are configured for a 1.5-L engine with a variable geometry turbocharger (VGT).



## Dynamic

CiEngineCore.slx contains the engine intake system, exhaust system, exhaust gas recirculation (EGR), fuel system, core engine, and turbocharger subsystems.



## Mapped

CiMappedEngine.slx uses the Mapped CI Engine block to look up power, air mass flow, fuel flow, exhaust temperature, efficiency, and emission performance as functions of engine speed and injected fuel mass.

## See Also

CI Controller | CI Core Engine | Mapped CI Engine

### **More About**

- “Internal Combustion Engine Reference Application Projects” on page 3-2
- Simulink Projects (Simulink)
- “Variant Systems” (Simulink)

## SI Engine Project Template

The Powertrain Blockset software provides a project template for spark-ignition (SI) engines. Use the template to create engine variants that you can use with the internal combustion engine reference application projects. The project template contains SI engine controller and plant models.

Use the project template to create CI engine variants for these reference applications:

- Conventional vehicle
- Hybrid electric vehicles
- SI engine dynamometer

To open the SI engine project template:

- 1 In Simulink, select **File > New > Project**.

In the Simulink start page, browse to Powertrain Blockset and select **SI Engine**.

- 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 CI or SI engine model.

### Controller

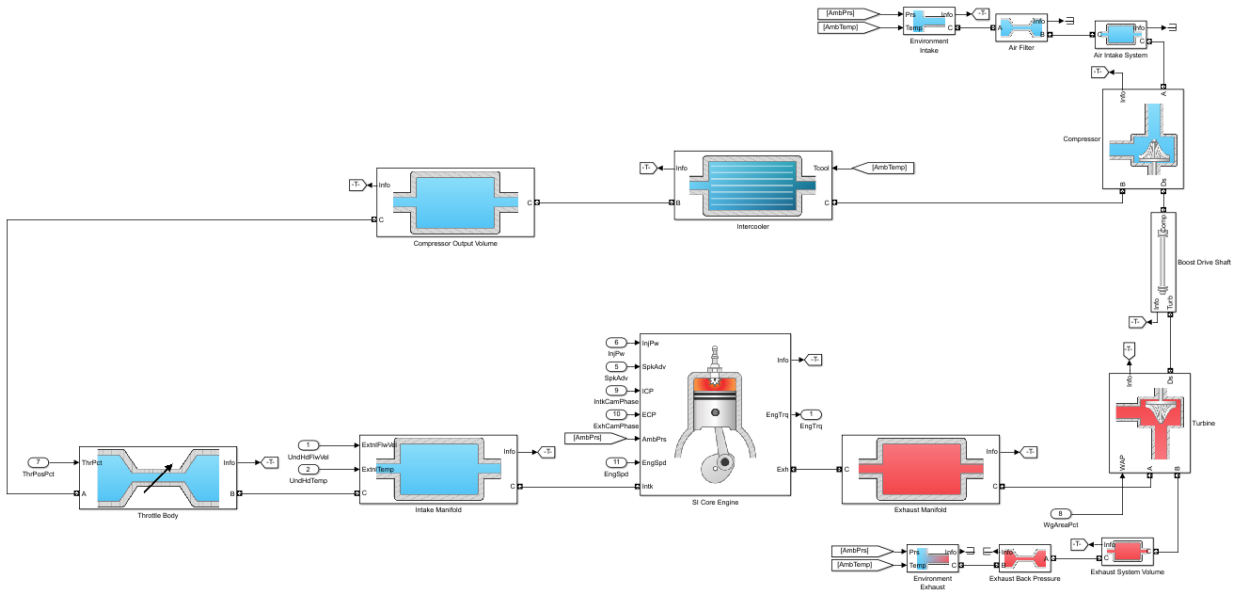
The **Controller** folder contains the `SiEngineController.slx` model. The model uses the SI Controller block and a **Start Stop Logic** subsystem to control the SI engine plant model.

### Plant

The **Plant** folder contains models that represent dynamic and mapped SI engines. By default, the dynamic and mapped engines are configured for a 1.5-L turbocharged engine.

## Dynamic

SiEngineCore.slx contains the engine intake system, exhaust system, core engine, and turbocharger subsystems.



## Mapped

SiMappedEngine.slx uses the Mapped SI Engine block to look up power, air mass flow, fuel flow, exhaust temperature, efficiency, and emission performance as functions of engine speed and commanded torque.

## See Also

Mapped SI Engine | SI Controller | SI Core Engine

## More About

- “Internal Combustion Engine Reference Application Projects” on page 3-2
- Simulink Projects (Simulink)

- “Variant Systems” (Simulink)



# Supporting Data

---

### Install Drive Cycle Data

This example shows how to install additional drive cycle data for the Drive Cycle Source block. By default, the 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 privileges for the Installation folder.

---

### See Also

Drive Cycle Source



# Calibration

---

## Generate Parameter Data for Datasheet Battery Block

This example shows how to import lithium-ion battery sheet data and generate parameters for the Datasheet Battery block. To run the example, you need the Curve Fitting Toolbox™.

In step 1, you import the datasheet data. Steps 2-5 show how to use curve-fitting techniques to obtain the open circuit voltage and battery resistance from the datasheet data. In steps 6-8, you validate the curve-fit voltage and battery values by comparing them to the Arrhenius behavior and the datasheet data. Finally, in step 9, you specify these Datasheet Battery block parameters:

- Rated capacity at nominal temperature
- Open circuit voltage table data
- Open circuit voltage breakpoints 1
- Internal resistance table data
- Battery temperature breakpoints 1
- Battery capacity breakpoints 2
- Initial battery charge

### Step 1: Import Battery Datasheet Data

Import the battery discharge and temperature datasheet into MATLAB. Ensure that each dataset in the datasheet includes a starting battery cell output voltage. Typically, data collected at different temperatures has the same reference current. Data collected at different currents has the same reference temperature.

For this example, load the battery datasheet discharge and temperature data for a lithium-ion battery from a file that contains 12 data sets. Each data set corresponds to battery data for a specific current and temperature. The data sets each have two columns. The first column contains the discharge capacity, in percent. The second column contains the corresponding battery cell voltage.

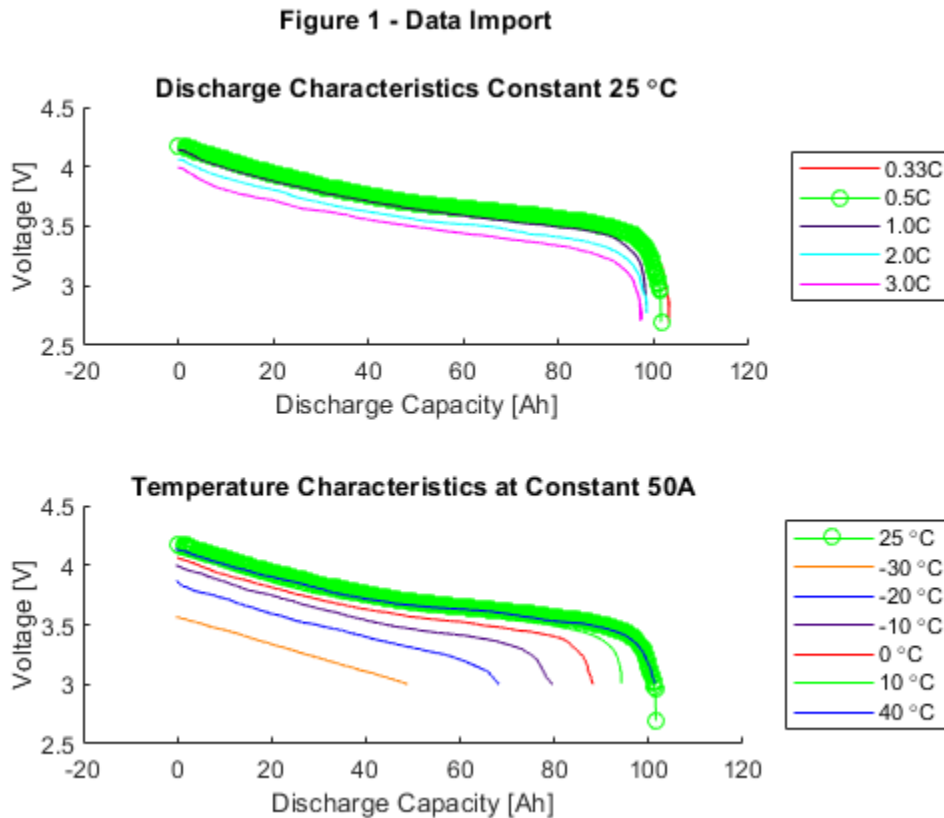
```
exp_data=load(fullfile(matlabroot,'examples','autoblks','ex_datasheetbattery_liion_1000'))
```

The example does not use the data set that corresponds to a current of 500 A at 25 °C.

Plot the discharge and temperature curves. Figure 1 shows the lithium-ion battery discharge characteristics at constant temperature (at five levels of current, shown as C-

rate) and constant current (at six temperatures). Figure 1 indicates the curve that corresponds to the reference temperature of 25 °C and the reference current of 50 A.

ex\_datasheetbattery\_plot\_data



## Step 2: Normalize State-of-Charge (SOC) Data

To represent 1-SOC capacity at constant temperature, normalize the relative discharge capacity with values between 0 and 1. Let 1 represent a fully discharged battery.

Set `ref_exp` to the dataset that corresponds to the reference temperature of 25 °C and the reference current of 50 A. Typically, the reference temperature is room temperature.

```
ref_exp = 2;
```

If you have several data sets, use a few for validation. Do not include them as part of the estimation dataset.

For this example, use `val_exp` to set up the validation and estimation data sets. Let 1 represent a validation dataset and 0 represent an estimation dataset.

```
val_exp = logical([1 0 0 0 1 0 0 0 0 1 0]);
```

Define reference current and temperature. For this example, the reference temperature is 25 °C and the reference current is 50 A.

```
ref_curr = current == current(ref_exp);  
ref_temp = temperature == temperature(ref_exp);
```

```
[sort_current, sort_index_current] = sort(current(ref_temp));  
[sort_temp, sort_index_temp] = sort(temperature(ref_curr));  
N = length(current); % Number of experiments
```

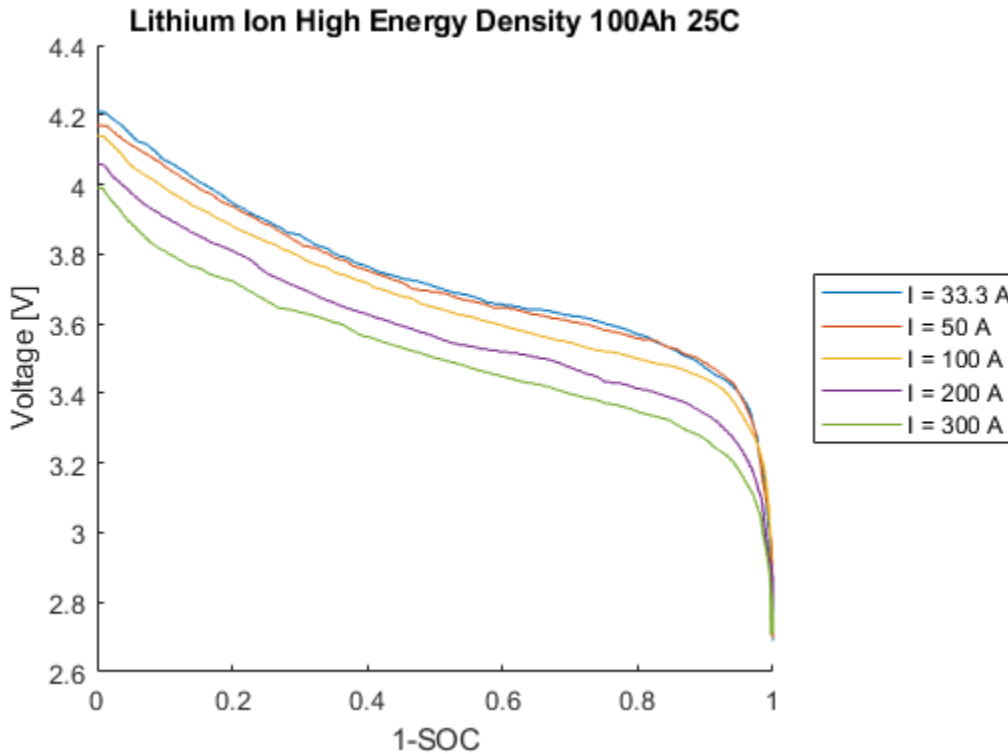
Prepare normalized x axes for each data set and find the actual capacity. x is a structure with as many fields as data sets and values between 0 and 1.

```
for i=1:N  
    x.(['curr' current_label{i} '_temp' temperature_label{i}]) = ...  
        exp_data.([label '_' current_label{i} '_' temperature_label{i}])(:,1)/...  
        exp_data.([label '_' current_label{i} '_' temperature_label{i}])(end,1);  
    % Calculate actual capacity for each datasheet  
    correct_cap.(['curr' current_label{i} '_temp' temperature_label{i}]) = ...  
        exp_data.([label '_' current_label{i} '_' temperature_label{i}])(end,1);  
end
```

Plot the normalized SOC data.

```
ex_datasheetbattery_plot_soc
```

Figure 2 - Normalized SOC Data



### Step 3: Fit Curves

Create `fitObj` curves for constant temperatures at different discharge rates and constant discharge rates at different temperatures. Use the `fitObj` curves to create a matrix of cell/module voltage versus discharge current at varying levels of SOC.

`fitObj` is a structure of fit objects that contains as many fields as data sets. The structure fits a discharge voltage to the normalized  $([0, 1])$  extracted Ah. This allows the discharge curves to be algebraically combined to calculate resistance at each SOC level.

Define state of charge vector and breakpoints.

```
SOC_LUT = (0:.01:1)';  
SOCbkpts = 0:.2:1;
```

Fit the discharge curves at different currents for reference temperature.

```
for i=find(ref_temp)  
    fitObj.(['fit' current_label{i}]) = ...  
        fit(x.(['curr' current_label{i} '_temp' temperature_label{i}]),...  
            exp_data.([label '_' current_label{i} '_' temperature_label{ref_exp}])(:,2),'sr'  
end
```

Fit the discharge curves at different temperatures for reference current.

```
for i=find(ref_curr)  
    fitObj.(['fit' temperature_label{i}]) = ...  
        fit(x.(['curr' current_label{i} '_temp' temperature_label{i}]),...  
            exp_data.([label '_' current_label{ref_exp} '_' temperature_label{i}])(:,2),'sr'  
end
```

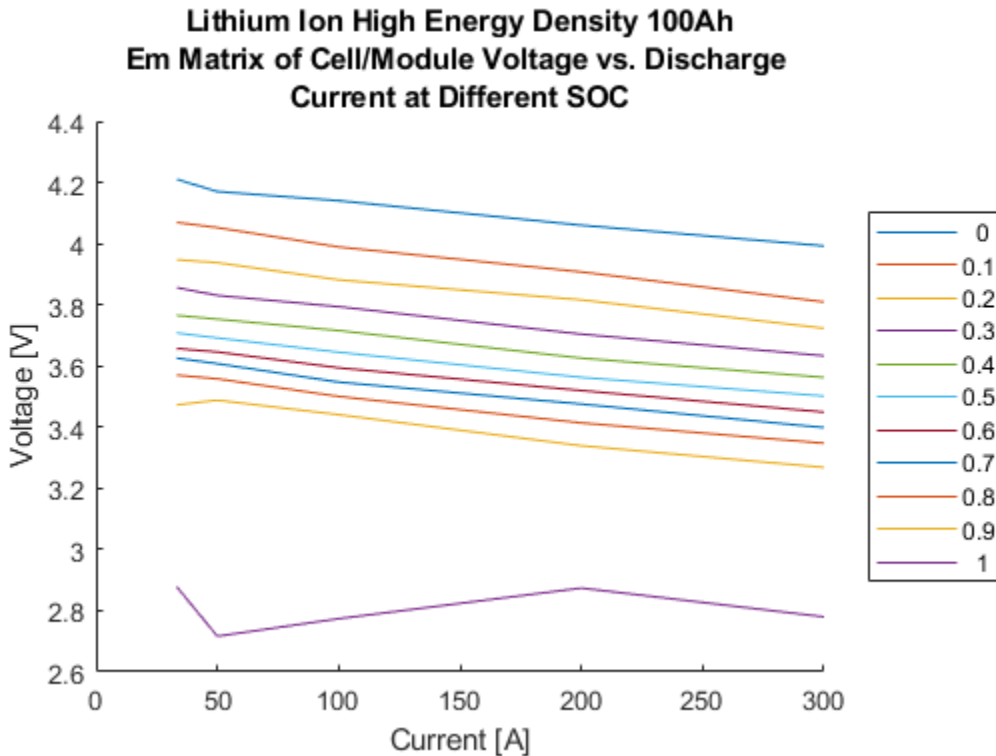
Construct the voltage versus discharge current for different SOC levels. Em\_MAT is a matrix with the SOC in rows and the current in columns.

```
Em_MAT = [];  
for i=find(ref_temp)  
    Em_MAT = [Em_MAT fitObj.(['fit' current_label{i}]) (SOC_LUT)];  
end
```

Figure 3 shows the voltage versus current at different SOC levels.

ex\_datasheetbattery\_plot\_curves

Figure 3 - Curve Fitting



#### Step 4: Extrapolate Open Circuit Voltage

To obtain the open circuit voltage,  $E_m$ , fit a line to the voltage versus current curve and extrapolate to  $i=0$ .

```
R0_refTemp = [];
for i=1:length(SOC_LUT)
    % Fit a line to V=f(I)
    fitSOC(['SOC' num2str(i)]) = fit(sort_current',Em_MAT(i,sort_index_current)', 'poly')
end
```

To estimate open circuit voltage,  $E_m$ , at all SOC levels, extrapolate the values of voltage to  $i=0$ .

```

Em = [];
for i=1:length(SOC_LUT)
    % Em = f(0)
    Em = [Em fitSOC.(['SOC' num2str(i)])(0)];
end
Em = Em';

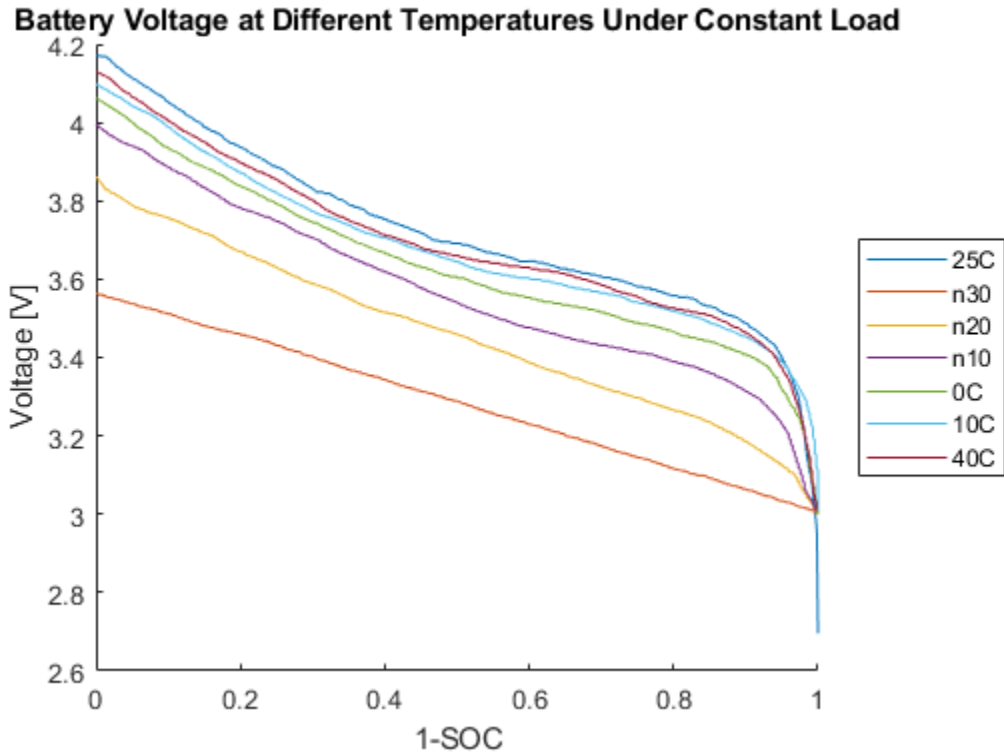
```

### Step 5: Determine Battery Voltage and Resistance at Different Temperatures

Use the discharge and temperature data to determine the battery resistance as a function of current and SOC at varying temperatures. The validation data is not included. Figure 4 shows the battery voltage at different temperatures.

ex\_datasheetbattery\_plot\_voltage

Figure 4 - Battery Voltage





Calculate the resistance at different temperatures using the reference current data set.

```
R0_LUT = [];
for i=find(ref_curr & ~val_exp)
    % Create fit object for V vs. SOC
    voltVsSOC(['temp' temperature_label{i}]) = fitObj(['fit' temperature_label{i}]);
    % Calculate R0(SOC,T) assuming linear behavior R0 = DeltaV / I
    R0(['temp' temperature_label{i}]) = (Em - voltVsSOC(['temp' temperature_label{i}]) / I);
    % Construct LUT
    R0_LUT = [R0_LUT R0(['temp' temperature_label{i}])];
end
```

To avoid the abrupt R change close to SOC=0 , extend R(0.9) all the way up to R(1). This is needed because of the way R is calculated. Make algorithm robust in case 0.9 is not an actual breakpoint

```
if ~isempty(find(SOC_LUT==0.9, 1))
    R0_LUT(SOC_LUT>0.9,:) = repmat(R0_LUT(SOC_LUT == 0.9,:),length(R0_LUT(SOC_LUT>0.9),1));
else
    [closestTo0p9, locClosestTo0p9] = min(abs(SOC_LUT-0.9));
    R0_LUT(SOC_LUT>closestTo0p9,:) = repmat(R0_LUT(locClosestTo0p9,:),...
        length(R0_LUT(SOC_LUT>closestTo0p9,:),1));
end
```

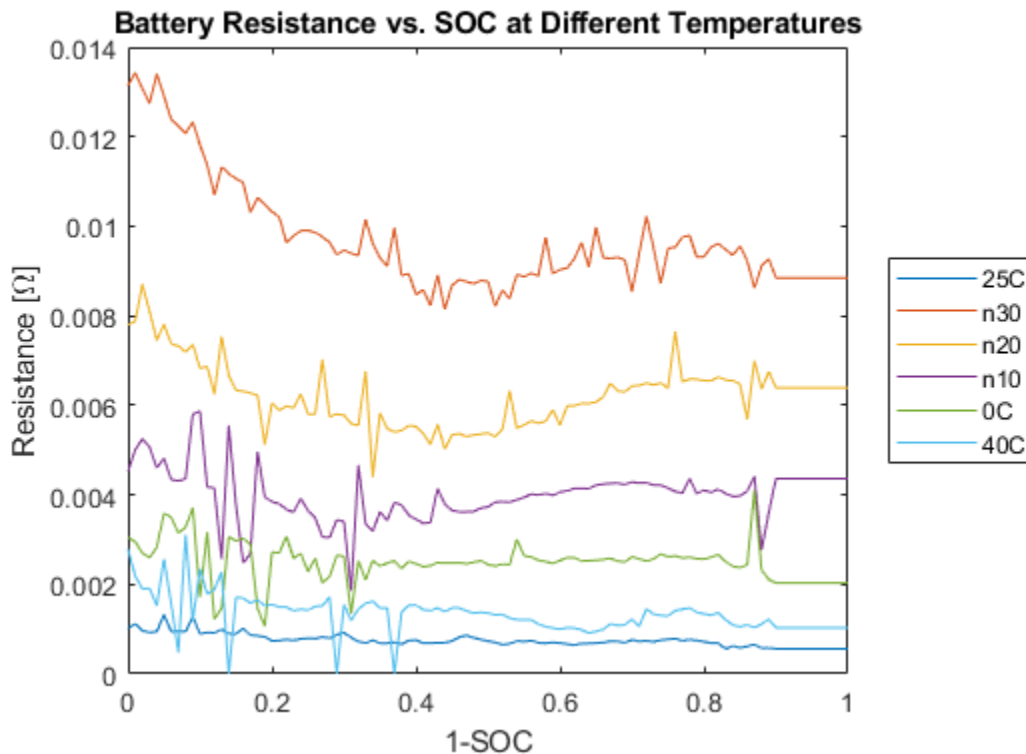
Determine the battery resistance at different temperatures.

```
R0_LUT = max(R0_LUT,0);
T_LUT = 273.15 + temperature(ref_curr & ~val_exp);
[T_LUT1,idx] = sort(T_LUT);
xtmp=R0_LUT';
R0_LUT1(1:length(T_LUT),:) = xtmp(idx,:);
```

Figure 5 shows the battery resistance at different temperatures.

ex\_datasheetbattery\_plot\_resistance

Figure 5 - Battery Resistance



### Step 6: Compare to Arrhenius Behavior

Since the temperature-dependent reaction rate for the lithium-ion battery follows an Arrhenius behavior, you can use a comparison to validate the curve fit.

To determine the curve-fit prediction for the Arrhenius behavior, examine the activation energy,  $E_a$ . Obtain the activation energy via the slope of the internal resistance,  $R_o$ , versus  $1000/T$  curve for different SOC. The slope equals the activation energy,  $E_a$ , divided by the universal gas constant,  $R_g$ .

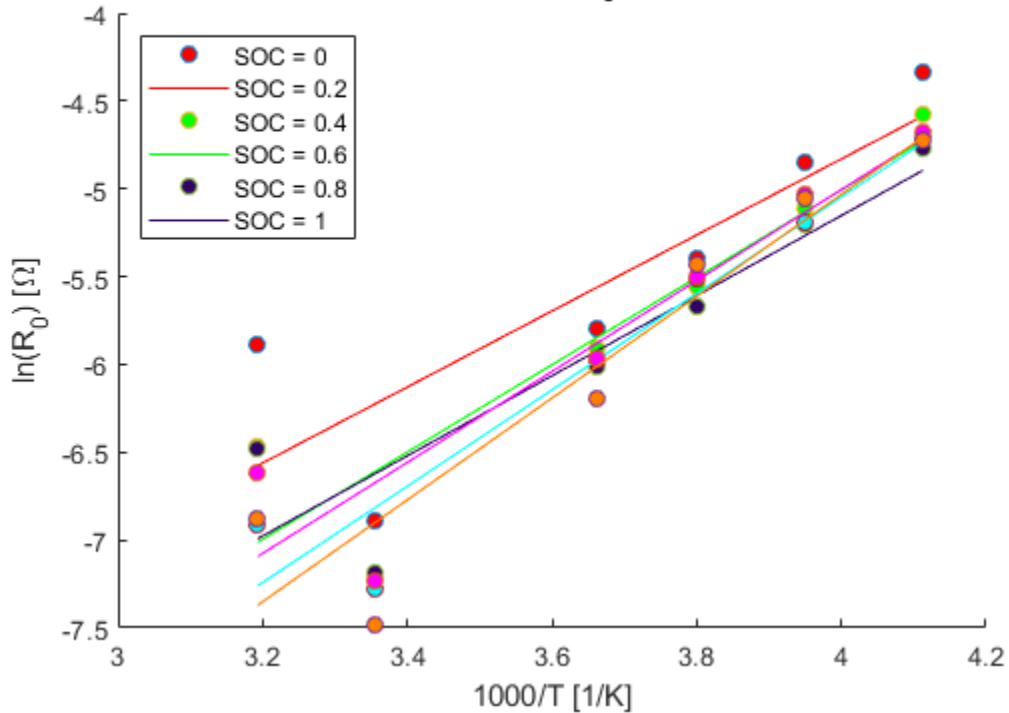
For a lithium-ion battery, a typical value of  $E_a$  is 20 kJ/mol[2]. Figure 6 indicates that the activation energy,  $E_a$ , obtained via the slope compares closely with 20 kJ/mol.

ex\_datasheetbattery\_plot\_arrhenius

Activation energy for Li ion conduction  
 $E_a = 17.9958 \quad 20.669 \quad 18.9557 \quad 22.8107 \quad 21.5289 \quad 24.0987$  kJ/mol  
 $E_a$  for electrolyte transport in Li ion battery = 20 kJ/mol

**Figure 6 - Arrhenius Behavior**

**Arrhenius plot of  $R_0$  vs.  $1000/T$**



### Step 7: Fit Battery Resistance

Fit the battery resistance to the validated temperature data as a function of SOC and temperature.

```
R0_LUT_bkpts = [];
counter = 1;
for i=find(ref_curr & ~val_exp)
    R0_LUT_bkpts = [R0_LUT_bkpts R0_LUT(idx',counter)];
```

```

        counter = counter+1;
    end

    [xx,yy,zz] = prepareSurfaceData(1000./T_LUT,SOCbkpts,log(R0_LUT_bkpts));
    [R0_vs_T_SOC_fit, gof] = fit([xx,yy],zz,'linearinterp');
    % [R0_vs_T_SOC_fit, gof] = fit([xx,yy],zz,'poly12');
    [xx1,yy1,zz1] = prepareSurfaceData(T_LUT,SOCbkpts,R0_LUT_bkpts);
    [R0_vs_T_SOC_fit1, gof] = fit([xx1,yy1],zz1,'linearinterp');

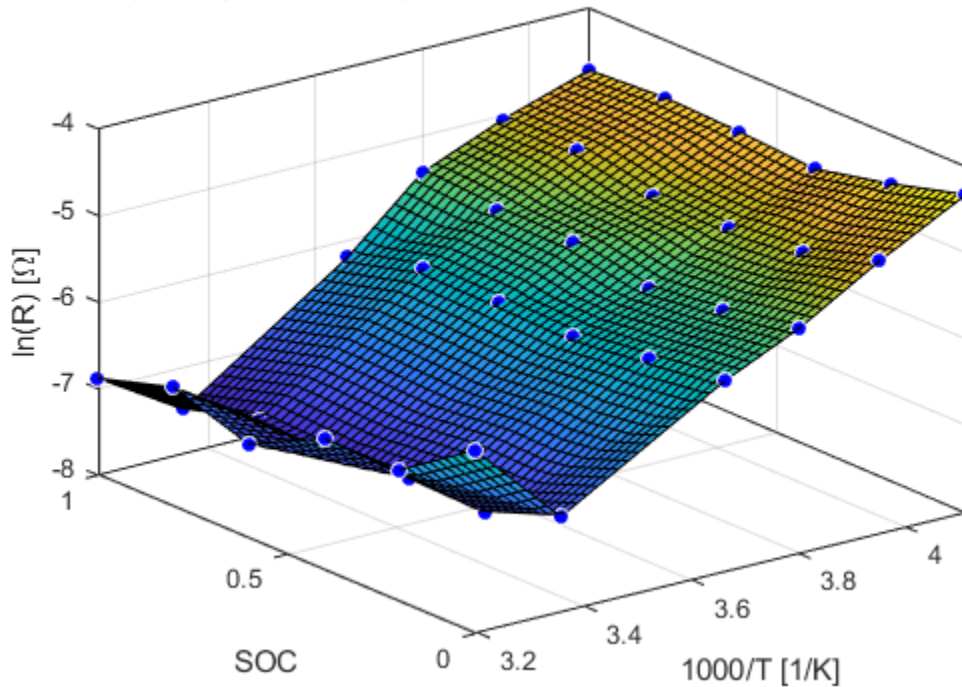
```

Figures 7 and 8 show the surface plots of the battery resistance as a function of SOC and temperature.

ex\_datasheetbattery\_plot\_surface

**Figure 7 - Surface Fit of Battery Resistance**

**ln(Battery Resistance) as a Function of SOC and Temperature**



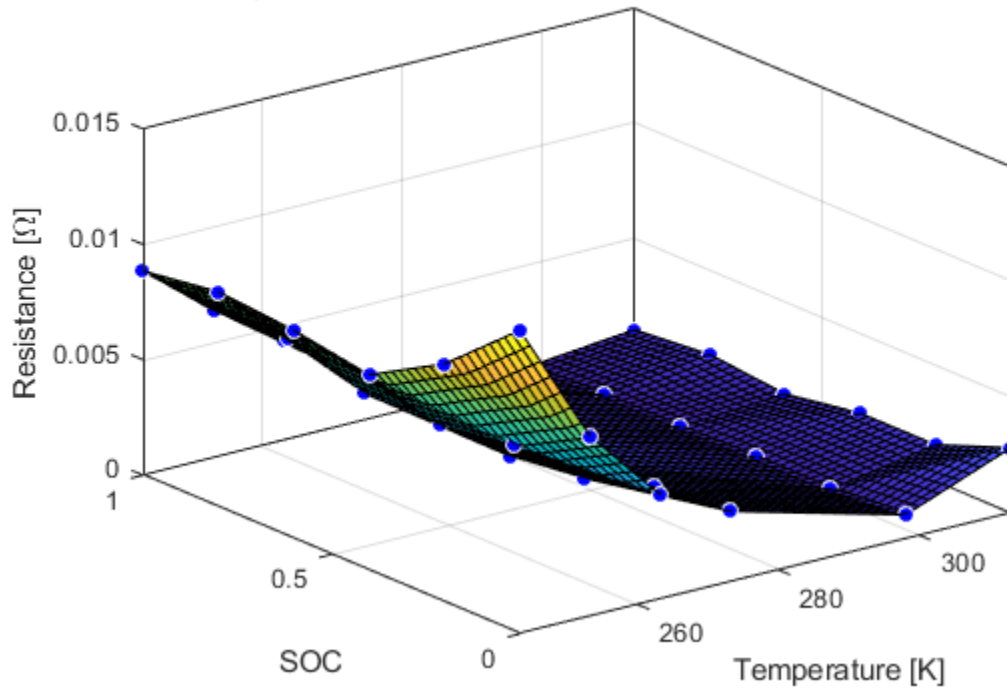
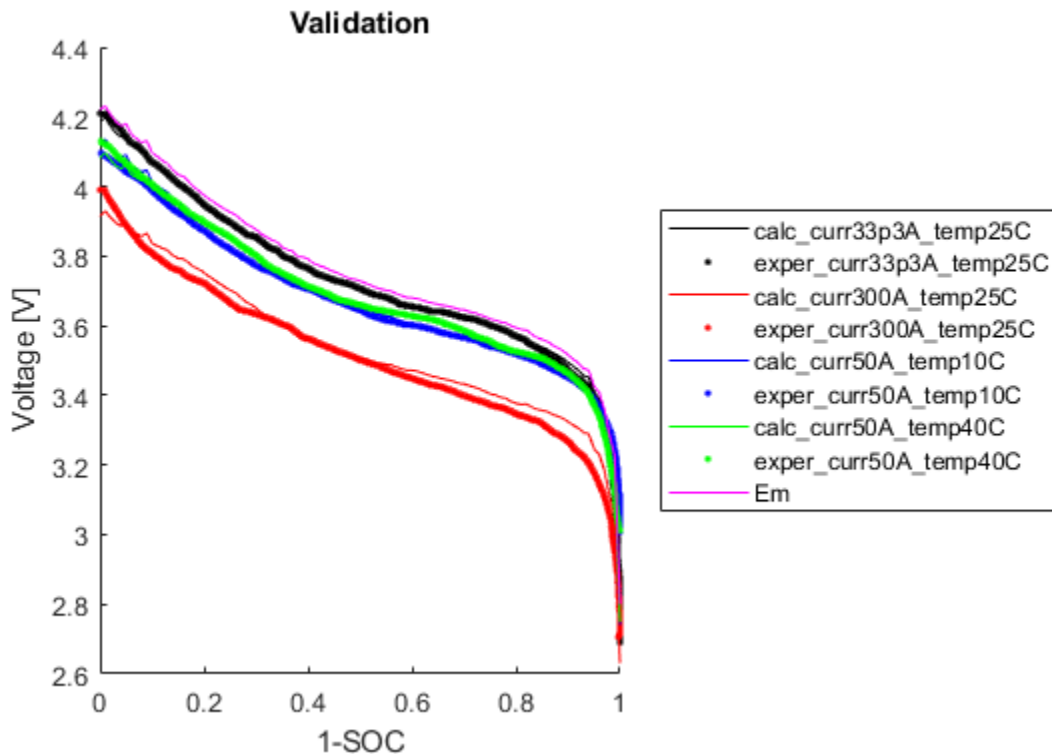
**Figure 8 - Surface Fit of Battery Resistance****Battery Resistance as a Function of SOC and Temperature****Step 8: Validate Battery Model Fit**

Figure 9 shows the calculated data and the experimental data set data.

ex\_datasheetbattery\_plot\_validation

Figure 9 - Validation of Battery Model Fit

**Step 9: Set the Datasheet Battery Block Parameters**

Set the **Rated capacity at nominal temperature** parameter to the capacity provided by the datasheet.

```
BattChargeMax = 100; % Ah Capacity from datasheet
```

Set the **Open circuit voltage table data** parameter to Em.

```
Em=Em;
```

Set the **Open circuit voltage breakpoints 1** parameter to the state of charge vector.

```
CapLUTBp=SOC_LUT;
```

Set the **Internal resistance table data** parameter to the fitted battery resistance data as a function of SOC and temperature.

```
RInt=R0_LUT_bkpts;
```

Set the **Battery temperature breakpoints 1** parameter to the temperature vector.

```
BattTempBp=T_LUT1;
```

Set the **Battery capacity breakpoints 2** parameter to the SOC vector.

```
CapSOCBp=SOCbkpts;
```

Set the **Initial battery charge** parameter to the value provided by the datasheet.

```
BattCapInit=100;
```

Clean up.

```
clear x xx xx1 yy yy1 zz zz1;
clear batt_id col correct cap count counter current;
clear correct_cap current_label data exp_data fitObj fitSOC gof;
clear i I idx indicot j k label leg line_colors;
clear indigo N orange p1 p2 purple ref_curr ref_exp ref_temp row;
clear sort_current sort_index_current sort_index_temp sort_temp;
clear temperature temperature_label V val_exp valIdx voltVsSOC xtmp temperature_label;
clear Ea Em_MAT markerType1 R0 R0_LUT R0_LUT1 R0_LUT_bkpts R0_refTemp R0_vs_T_fit;
clear T R R0_vs_T_SOC_fit R0_vs_T_SOC_fit1 SOC_LUT SOCbkpts T_LUT T_LUT1;
```

## References

[1] Jackey, Robyn, Tarun Huria, Massimo Ceraolo, and Javier Gazzarri. "High fidelity electrical model with thermal dependence for characterization and simulation of high power lithium battery cells." *IEEE International Electric Vehicle Conference*. March 2012, pp. 1-8.

[2] Ji, Yan, Yancheng Zhang, and Chao-Yang Wang. *Journal of the Electrochemical Society*. Volume 160, Issue 4 (2013), A636-A649.

## See Also

Datasheet Battery

## Generate Parameter Data for Equivalent Circuit Battery Block

Using MathWorks tools, estimation techniques, and measured lithium-ion or lead acid battery data, you can generate parameters for the Equivalent Circuit Battery block. The Equivalent Circuit Battery block implements a resistor-capacitor (RC) circuit battery with open circuit voltage, series resistance, and 1 through N RC pairs. The number of RC pairs reflects the number of time constants that characterize the battery transients. Typically, the number of RC pairs ranges from 1 through 5.

To create parameter data for the Equivalent Circuit Battery block, follow these workflow steps. The steps use numerical optimization techniques to determine the number of recommended RC pairs, provide initial estimates for the battery model circuit parameters, and estimate parameters to fit a model to experimental pulse discharge data. The results provide the open circuit voltage, series resistance, and RC pair parameter data for the Equivalent Circuit Battery block.

The workflow steps use this example script and models for a lithium-ion polymer (LiPo) battery:

- Estimate battery discharge script `Example_DischargePulseEstimation`.
- Model `BatteryEstim3RC_PTBS`.
- Model `BatteryEstim3RC_PTBS_EQ`.

The example battery discharge script uses a battery class to control the parameter estimation workflow.

Workflow	Description	Additional MathWorks Tooling
"Step 1: Load and Preprocess Data" on page 6-17	Load and preprocess time series battery discharge voltage and current data.	None
"Step 2: Determine the Number of RC Pairs" on page 6-20	Determine the number of necessary time constants (TC) for estimation.	Curve Fitting Toolbox



Workflow	Description	Additional MathWorks Tooling
<p>“Step 3: Estimate Parameters” on page 6-21</p>	<p>For battery discharge data, estimate and optimize:</p> <ul style="list-style-type: none"> <li>• Open-circuit voltage, <math>E_m</math></li> <li>• Series resistance, <math>R_0</math></li> <li>• RC pair(s) time constant(s), <math>\tau</math></li> <li>• RC pair(s) resistance(s), <math>R_x</math></li> </ul> <p>Use a model that exercises the Estimation Equivalent Circuit Battery block.</p>	<p>Curve Fitting Toolbox, Parallel Computing Toolbox, Optimization Toolbox, and Simulink Design Optimization</p>
<p>“Step 4: Set Equivalent Circuit Battery Block Parameters” on page 6-27</p>	<p>Set these block parameters:</p> <ul style="list-style-type: none"> <li>• <b>Open circuit voltage table data</b></li> <li>• <b>Series resistance table data</b></li> <li>• <b>State of charge breakpoints</b></li> <li>• <b>Temperature breakpoints</b></li> <li>• <b>Battery capacity table</b></li> <li>• <b>Network resistance table data</b></li> <li>• <b>Network capacitance table data</b></li> </ul>	<p>None</p>

## Step 1: Load and Preprocess Data

### Data Format and Requirements

The workflow supports pulse discharge sequences from 100% to 0% state-of-charge (SOC).

Data requirements include:

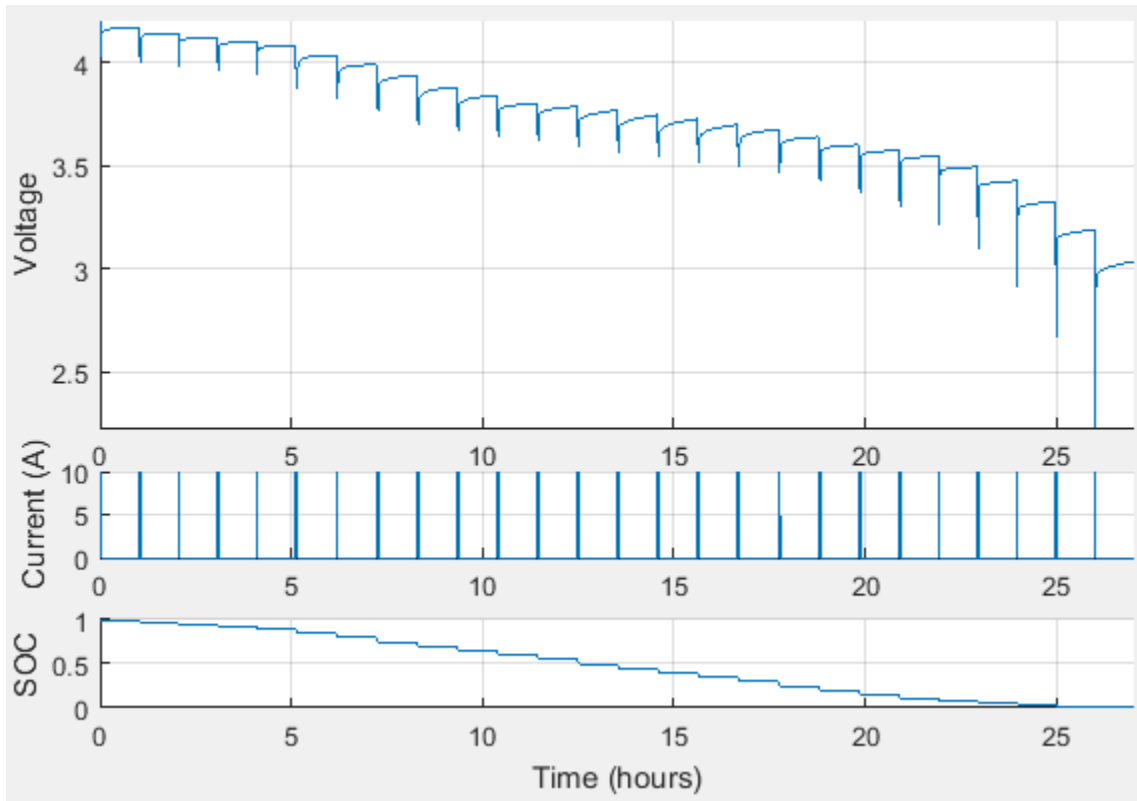
- Time series consisting of current and voltage from an experimental pulse discharge. For each experimental data set, the temperature is constant. The sample rate should be a minimum of 1 Hz, with an ideal rate at 10 Hz. This table summarizes the accuracy requirements.

<b>Measurement</b>	<b>Accuracy</b>	<b>Ideal</b>
Voltage	$\pm 5$ mV	$\pm 1$ mV
Current	$\pm 100$ mA	$\pm 10$ mA
Temperature	$\pm 1$ °C	$\pm 1$ °C

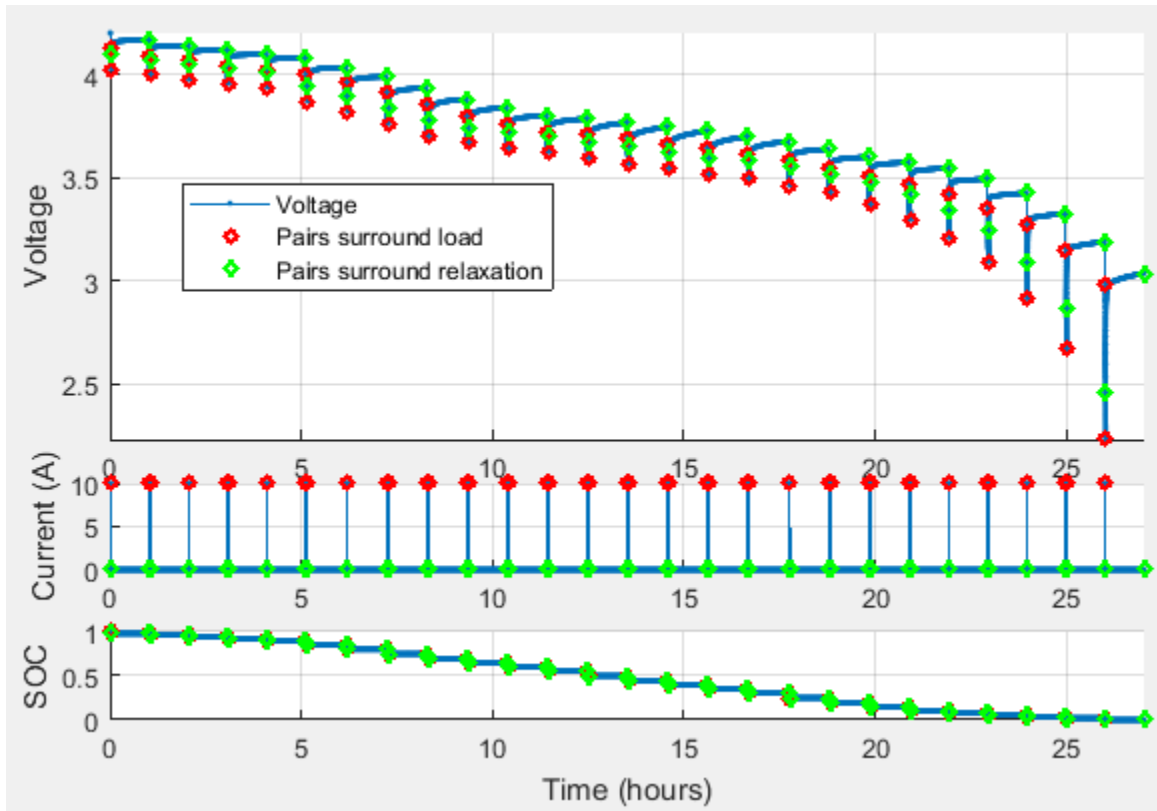
- Change in SOC for each pulse should not be greater than 5%.
- Data collection at high or low SOC might need modification to ensure safety.
- Sufficient relaxation time after each pulse to ensure battery approaches steady-state voltage.

### **Load and Preprocess Data**

Load the battery time, voltage, and discharge data. Break up the data into `Battery.Pulse` objects. For example, load and preprocess the discharge data for a lithium-ion polymer (LiPo) battery using the `Step1: Load and Preprocess Data` commands in the `Example_DischargePulseEstimation` script.



**Pulse Sequence**



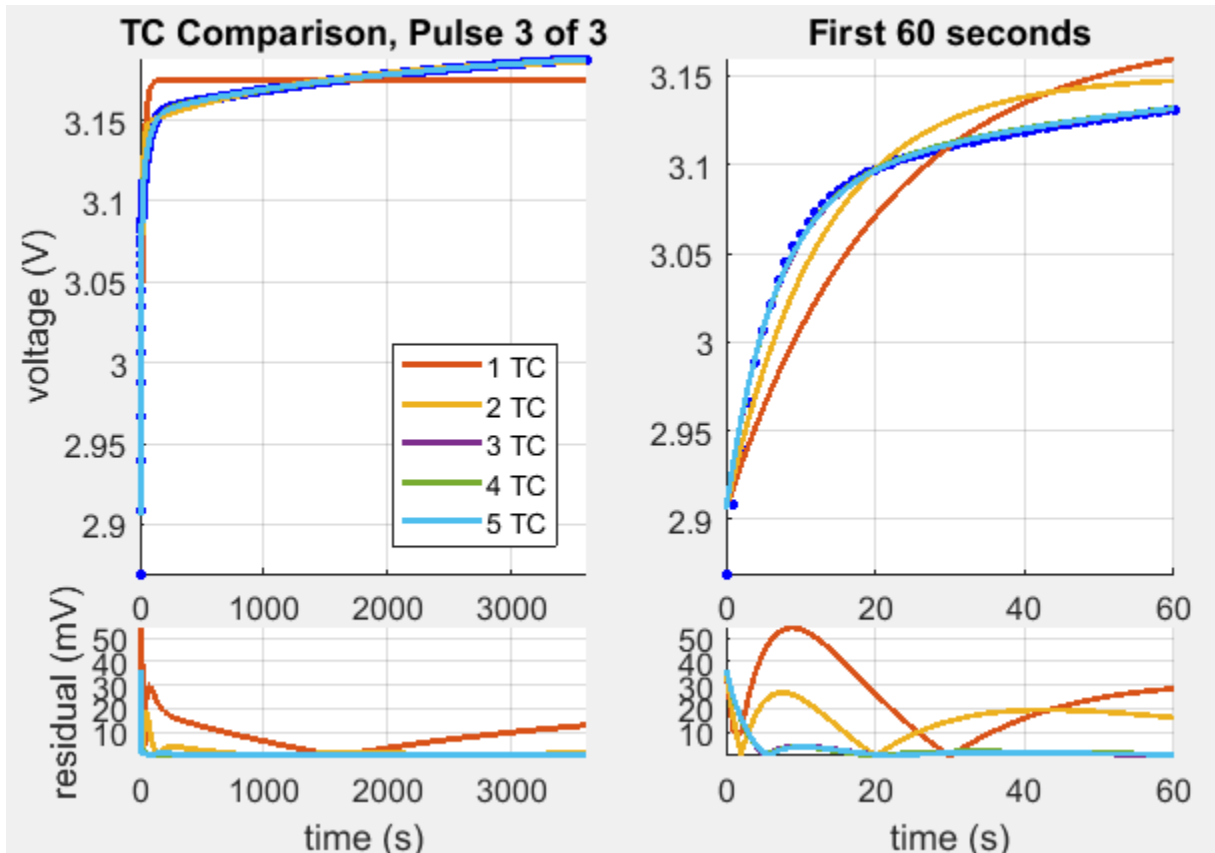
### Pulse Identification

## Step 2: Determine the Number of RC Pairs

Determine how many RC pairs to use in the model. You can investigate how many RC pairs to use by executing the Step 2: Determine the Number of RC Pairs commands in the `Example_DischargePulseEstimation` script. The example script uses the `BatteryEstim3RC_PTBS` model.

### Compare Pulse Time Constants

Compare the time constants (TC) for each pulse. This example compares three pulses.



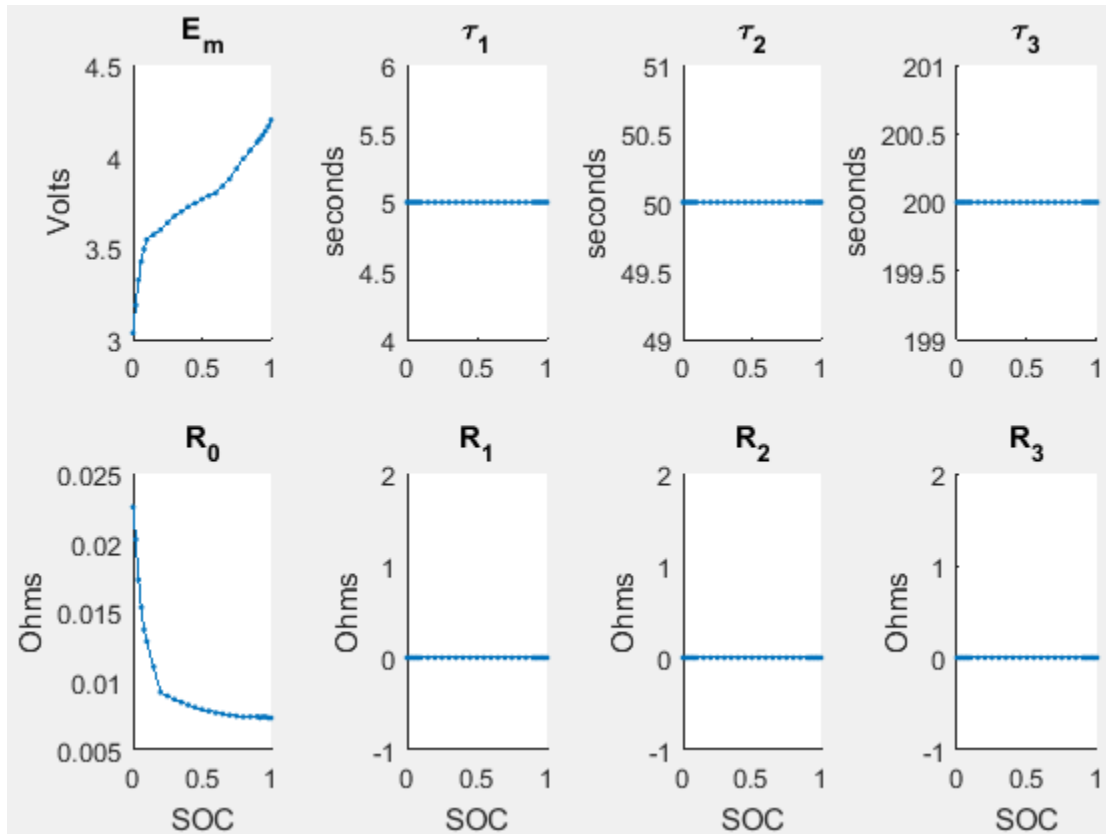
**TC Comparison, Pulse 3 of 3**

### Step 3: Estimate Parameters

Estimate the parameters. You can investigate parameter estimation by executing the Step 3: Estimate Parameters commands in the Example\_DischargePulseEstimation script.

#### Estimate Em and R0

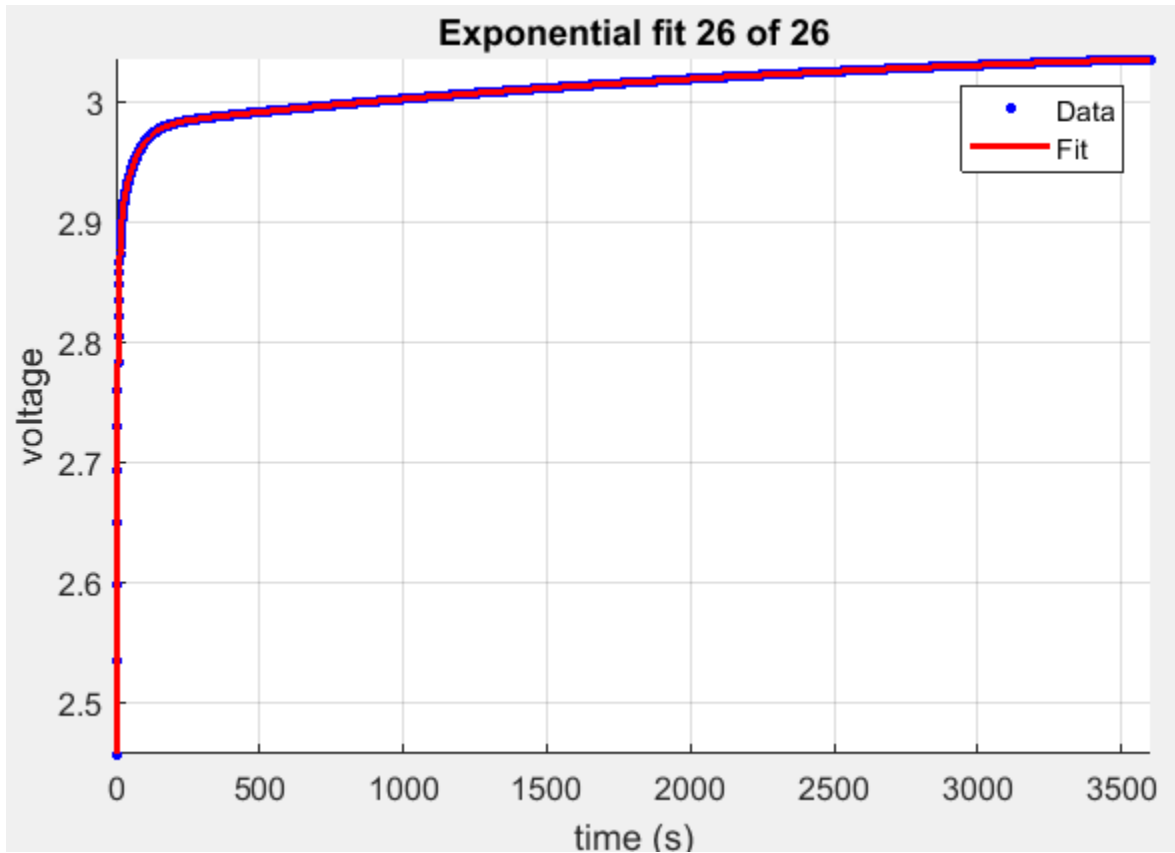
Inspect the voltage immediately before and after the current is applied and removed at the start and end of each pulse. The estimation technique uses the voltage for a raw calculation to estimate the open-circuit voltage ( $E_m$ ) and the series resistance ( $R_0$ ).



**Parameter Tables**

**Estimate Tau**

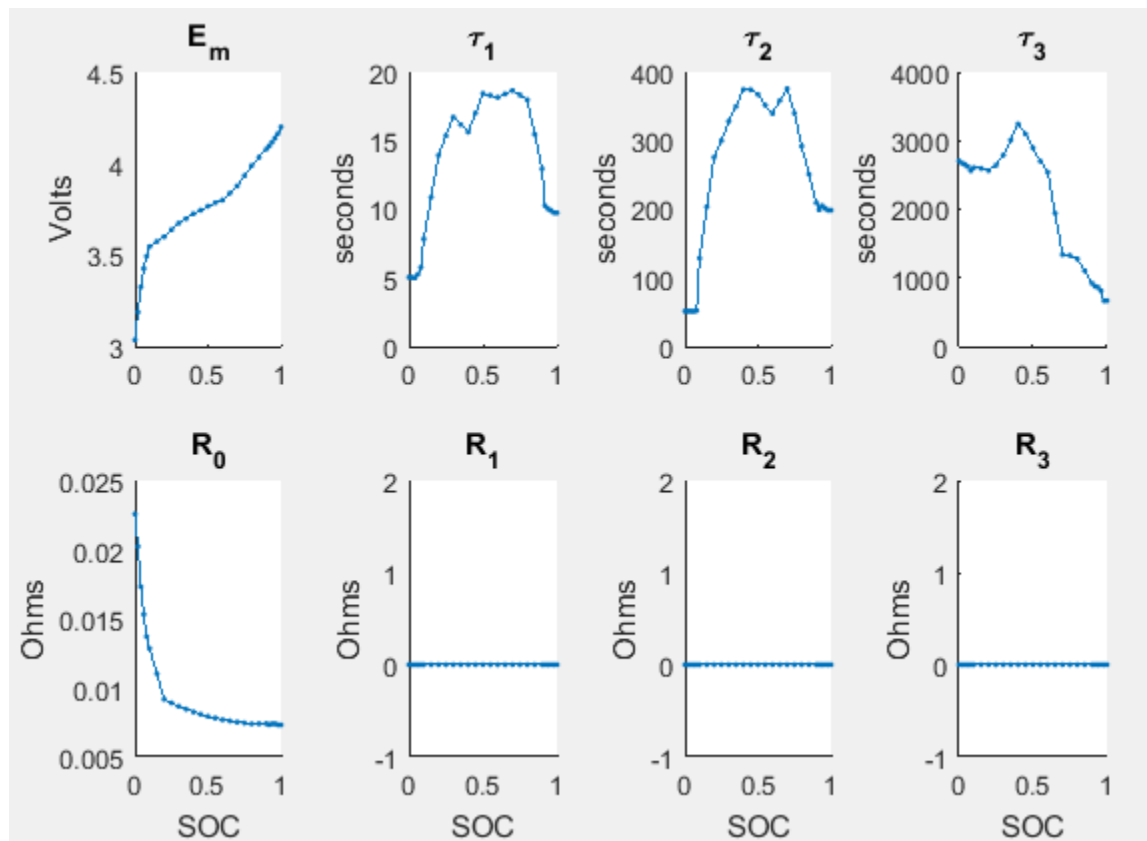
Use a curve-fitting technique on the pulse relaxation to estimate the RC time constant (Tau) at each SOC.



### Relaxation Tau Fit

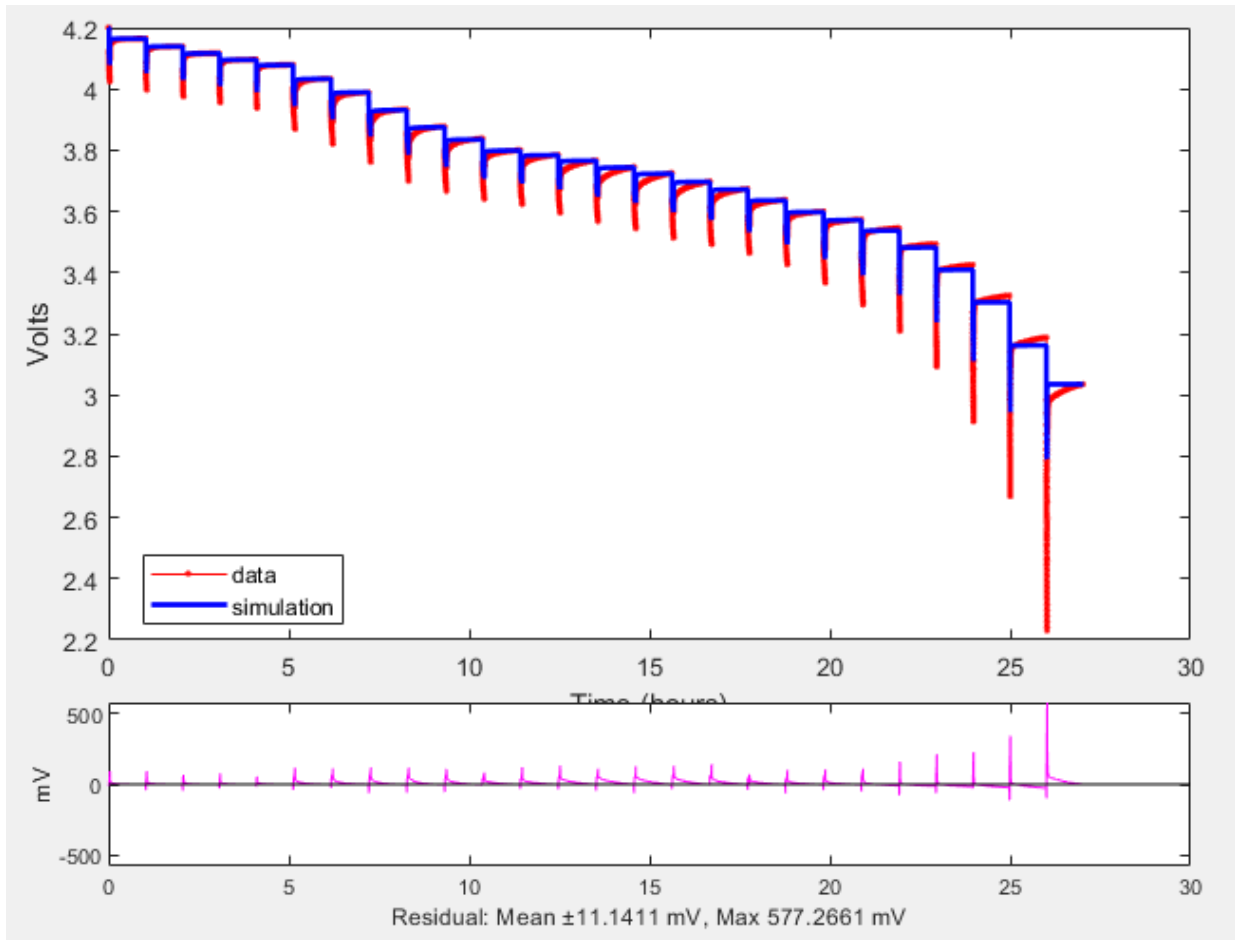
### Plot Estimates

Plot the parameter and pulse sequence data and simulation comparison.



Parameter Tables

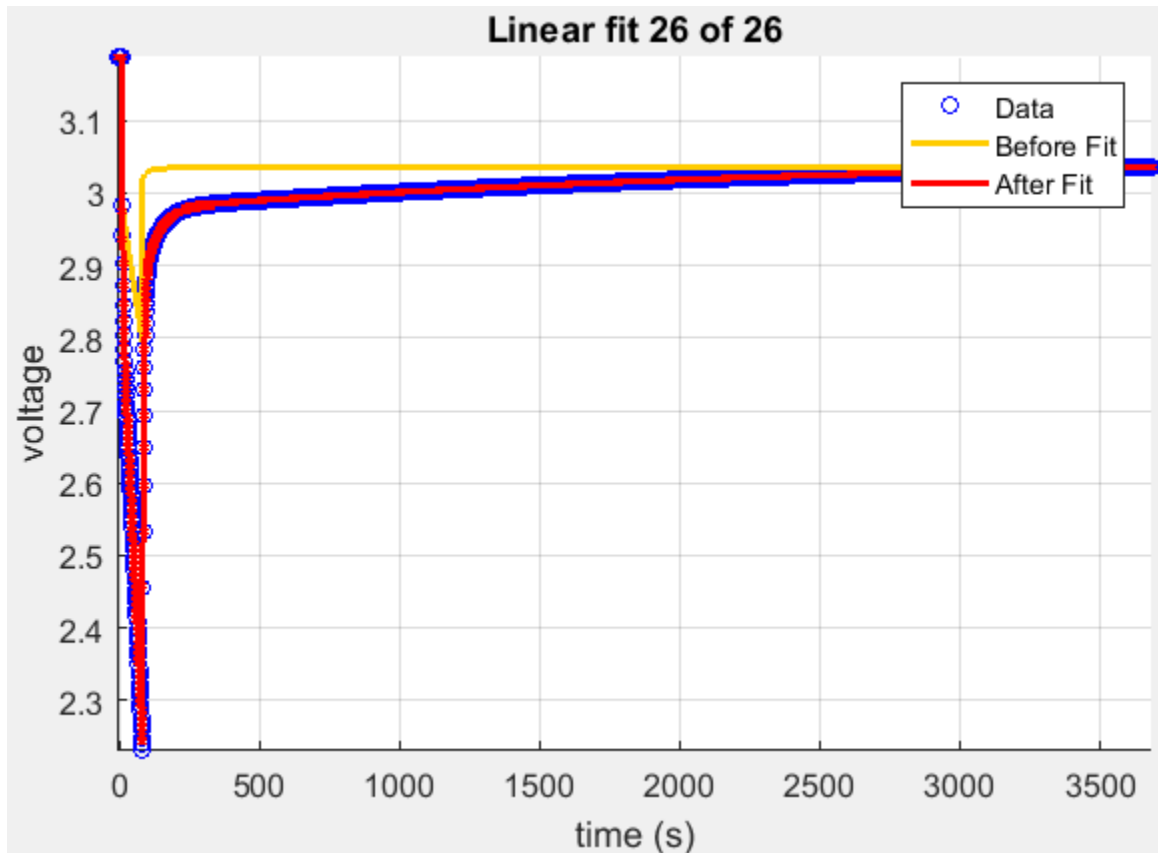




## Pulse Sequence

### Identify Parameters and Set Initial Values

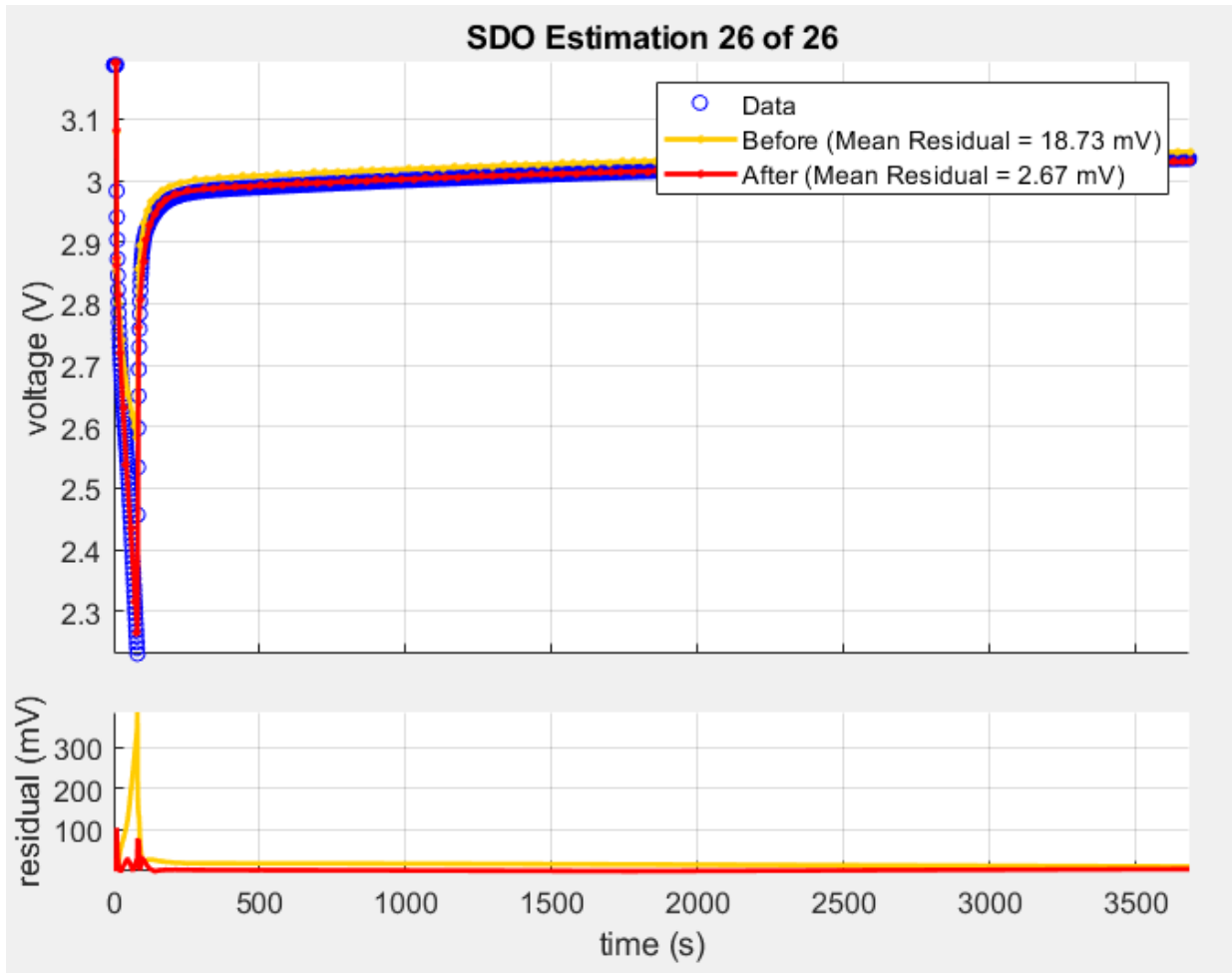
Identify parameters and set the initial values using a linear system approach, pulse-by-pulse.



**Linear Fit**

**Optimize Estimates**

Optimize the  $E_m$ ,  $R_0$ ,  $R_x$ , and  $\tau$  estimates using Simulink Design Optimization.



### Pulse Identification

## Step 4: Set Equivalent Circuit Battery Block Parameters

Set the Equivalent Circuit Battery block parameters to the values determined in step 3. To investigate setting the block parameters, execute the Step 4: Set Equivalent Circuit Battery Block Parameters commands in the Example\_DischargePulseEstimation script. The experiment ran at two constant

temperatures. There are three RC-pairs. The Equivalent Circuit Battery block parameter values are summarized in this table:

Parameter	Example Value
Number of series RC pairs	3
Open circuit voltage table data, EM	EmPrime = repmat(Em,2,1)';
Series resistance table data, R0	R0Prime = repmat(R0,2,1)';
State of charge breakpoints, SOC_BP	SOC_LUTPrime = SOC_LUT;
Temperature breakpoints, Temperature_BP	TempPrime = [303 315.15];
Battery capacity table	CapacityAhPrime = [CapacityAh CapacityAh];
Network resistance table data, R1	R1Prime = repmat(Rx(1,:),2,1)';
Network capacitance table data, C1	C1Prime = repmat(Tx(1,:)./Rx(1,:),2,1)';
Network resistance table data, R2	R2Prime = repmat(Rx(2,:),2,1)';
Network capacitance table data, C2	C2Prime = repmat(Tx(2,:)./Rx(2,:),2,1)';
Network resistance table data, R3	R3Prime = repmat(Rx(3,:),2,1)';
Network capacitance table data, C3	C3Prime = repmat(Tx(3,:)./Rx(3,:),2,1)';

## References

- [1] Ahmed, R., J. Gazzarri, R. Jackey, S. Onori, S. Habibi, et al. "Model-Based Parameter Identification of Healthy and Aged Li-ion Batteries for Electric Vehicle Applications." *SAE International Journal of Alternative Powertrains*. doi: 10.4271/2015-01-0252, 4(2):2015.
- [2] Gazzarri, J., N. Shrivastava, R. Jackey, and C. Borghesani. "Battery Pack Modeling, Simulation, and Deployment on a Multicore Real Time Target." *SAE International Journal of Aerospace*. doi:10.4271/2014-01-2217, 7(2):2014.
- [3] Huria, T., M. Ceraolo, J. Gazzarri, and R. Jackey. "High fidelity electrical model with thermal dependence for characterization and simulation of high power lithium battery cells." *IEEE® International Electric Vehicle Conference*. March 2012, pp. 1-8.

- [4] Huria, T., M. Ceraolo, J. Gazzarri, and R. Jackey. "Simplified Extended Kalman Filter Observer for SOC Estimation of Commercial Power-Oriented LFP Lithium Battery Cells." *SAE Technical Paper 2013-01-1544*. doi:10.4271/2013-01-1544, 2013.
- [5] Jackey, R. "A Simple, Effective Lead-Acid Battery Modeling Process for Electrical System Component Selection." *SAE Technical Paper 2007-01-0778*. doi: 10.4271/2007-01-0778, 2007.
- [6] Jackey, R., G. Plett, and M. Klein. "Parameterization of a Battery Simulation Model Using Numerical Optimization Methods." *SAE Technical Paper 2009-01-1381*. doi: 10.4271/2009-01-1381, 2009.
- [7] Jackey, R., M. Saginaw, T. Huria, M. Ceraolo, P. Sanghvi, and J. Gazzarri. "Battery Model Parameter Estimation Using a Layered Technique: An Example Using a Lithium Iron Phosphate Cell." *SAE Technical Paper 2013-01-1547*. Warrendale, PA: SAE International, 2013.

## See Also

Equivalent Circuit Battery | Estimation Equivalent Circuit Battery

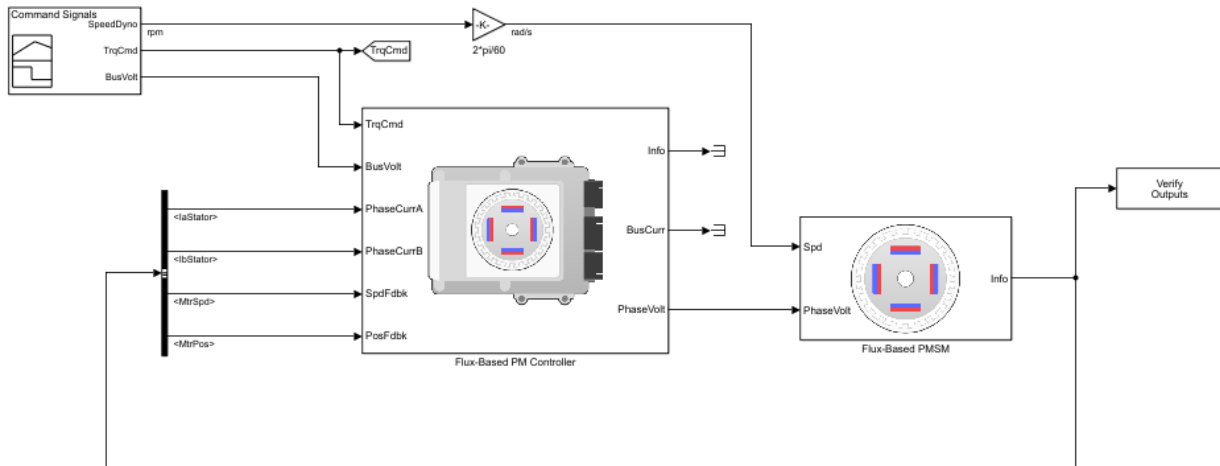
## Generate Parameters for Flux-Based Blocks

This table provides a description of the process to generate the parameters and links to examples.

For Block	To Generate	Description	Example
Flux-Based PM Controller	Current Controller parameters: <ul style="list-style-type: none"> <li>• <b>Corresponding d-axis current reference, <math>i_{d\_ref}</math></b></li> <li>• <b>Corresponding q-axis current reference, <math>i_{q\_ref}</math></b></li> <li>• <b>Vector of speed breakpoints, <math>wbp</math></b></li> <li>• <b>Vector of torque breakpoints, <math>tbp</math></b></li> </ul>	Use the Model-Based Calibration Toolbox to generate optimized current controller tables for flux-based motor controllers.  Based on nonlinear motor flux data, the calibration tables optimize: <ul style="list-style-type: none"> <li>• Motor efficiency</li> <li>• Maximum torque per ampere (MTPA)</li> <li>• Flux weakening</li> </ul>	“Generate Current Controller Parameters” on page 6-33

For Block	To Generate	Description	Example
	Motor parameters: <ul style="list-style-type: none"> <li>• <b>Vector of d-axis current breakpoints, id_index</b></li> <li>• <b>Vector of q-axis current breakpoints, iq_index</b></li> <li>• <b>Corresponding d-axis flux, lambda_d</b></li> <li>• <b>Corresponding q-axis flux, lambda_q</b></li> </ul>	Use MATLAB scripts available with Powertrain Blockset to load flux motor data, visualize the flux surface, and create plots of flux as a function of current.	“Generate Feed-Forward Flux Parameters” on page 6-60
Flux-Based PMSM	Parameters: <ul style="list-style-type: none"> <li>• <b>Vector of d-axis flux, flux_d</b></li> <li>• <b>Vector of q-axis flux, flux_q</b></li> <li>• <b>Corresponding d-axis current, id</b></li> <li>• <b>Corresponding q-axis current, iq</b></li> </ul>	Use MATLAB scripts available with Powertrain Blockset to load flux motor data, invert the flux, and create plots of current as a function of flux.	“Generate Parameters for Flux-Based PMSM Block” on page 6-65

To open a model with optimized parameters for the Flux-Based PM Controller and Flux-Based PMSM blocks, on the command-line, type `Flux_Based_PMSM_TestBench`.



## References

- [1] Hu, Dakai, Yazan Alsmadi, and Longya Xu. "High fidelity nonlinear IPM modeling based on measured stator winding flux linkage." *IEEE Transactions on Industry Applications*, Vol. 51, No. 4, July/August 2015.
- [2] Chen, Xiao, Jiabin Wang, Bhaskar Sen, Panagiotis Lasari, Tianfu Sun. "A High-Fidelity and Computationally Efficient Model for Interior Permanent-Magnet Machines Considering the Magnetic Saturation, Spatial Harmonics, and Iron Loss Effect." *IEEE Transactions on Industrial Electronics*, Vol. 62, No. 7, July 2015.

## See Also

Flux-Based PMSM | Flux-Based PM Controller



## Generate Current Controller Parameters

Using the Model-Based Calibration Toolbox, you can generate optimized current tables for flux-based motor controllers. Use the calibration tables for the Powertrain Blockset Flux-Based PM Controller current controller block parameters.

Based on nonlinear motor flux data, the calibration tables optimize:

- Motor efficiency
- Maximum torque per ampere (MTPA)
- Flux weakening

To generate optimized current tables, follow these workflow steps.

Workflow Steps	Description	MathWorks Tooling
“Collect and Post Process Motor Data” on page 6-34	<p>Collect the nonlinear motor flux data from dynamometer testing or finite element analysis (FEA). For this example, file <code>ex_motor_data.xlsx</code> contains the data that you need:</p> <ul style="list-style-type: none"> <li>• <math>d</math>-axis current, <math>I_d</math>, in A</li> <li>• <math>q</math>-axis current, <math>I_q</math>, in A</li> <li>• Motor speed, <math>n</math>, in rpm</li> <li>• <math>d</math>-axis voltage, <math>V_d</math>, in V</li> <li>• <math>q</math>-axis voltage, <math>V_q</math>, in V</li> <li>• Electromagnetic motor torque, <math>T_e</math>, in N·m</li> </ul>	N/A
“Model Motor Data” on page 6-35	<p>Use a one-stage model to fit the data. Specifically:</p> <ul style="list-style-type: none"> <li>• Import data</li> <li>• Filter data</li> <li>• Fit model</li> </ul>	Model-Based Calibration Toolbox

Workflow Steps	Description	MathWorks Tooling
“Generate Calibration” on page 6-40	Calibrate and optimize the data using objectives and constraints. Specifically: <ul style="list-style-type: none"> <li>• Create functions.</li> <li>• Create tables from model.</li> <li>• Run an optimization.</li> <li>• Generate and fill optimized current controller calibration tables that are functions of motor torque and motor speed.</li> </ul>	Model-Based Calibration Toolbox
“Set Block Parameters” on page 6-58	Use the optimized current controller calibration tables for the Flux-Based PM Controller block current controller parameters.	Powertrain Blockset

## Collect and Post Process Motor Data

Collect this nonlinear motor flux data from dynamometer testing or finite element analysis (FEA):

- $d$ - and  $q$ - axis current
- $d$ - and  $q$ - axis flux linkage
- Electromagnetic motor torque

Use the collected data and motor speed to calculate the  $d$ - and  $q$ -axis voltages:

$$v_d = R_s i_d - \omega_e \lambda_q$$

$$v_q = R_s i_q + \omega_e \lambda_d$$

$$n = \frac{60\omega_e}{2\pi P}$$

The equations use these variables:

$V_d, V_q$	$d$ - and $q$ - axis voltage, respectively
$i_d, i_q$	$d$ - and $q$ - axis current, respectively
$\lambda_d, \lambda_q$	$d$ - and $q$ - axis flux linkage, respectively
$R_S$	Stator resistance
$\omega_e$	Electrical motor angular speed, rad/s
$n$	Motor speed, rpm
$P$	Number of pole pairs

Finally, for each data point, create a file containing:

- $d$ -axis current,  $I_d$ , in A
- $q$ -axis current,  $I_q$ , in A
- Motor speed,  $n$ , in rpm
- $d$ -axis voltage,  $V_d$ , in V
- $q$ -axis voltage,  $V_q$ , in V
- Electromagnetic motor torque,  $T_e$ , in N.m

For this example, the data file `matlab\toolbox\mbc\mbctraining\ex_motor_data.xlsx` contains the motor flux data.

## Model Motor Data

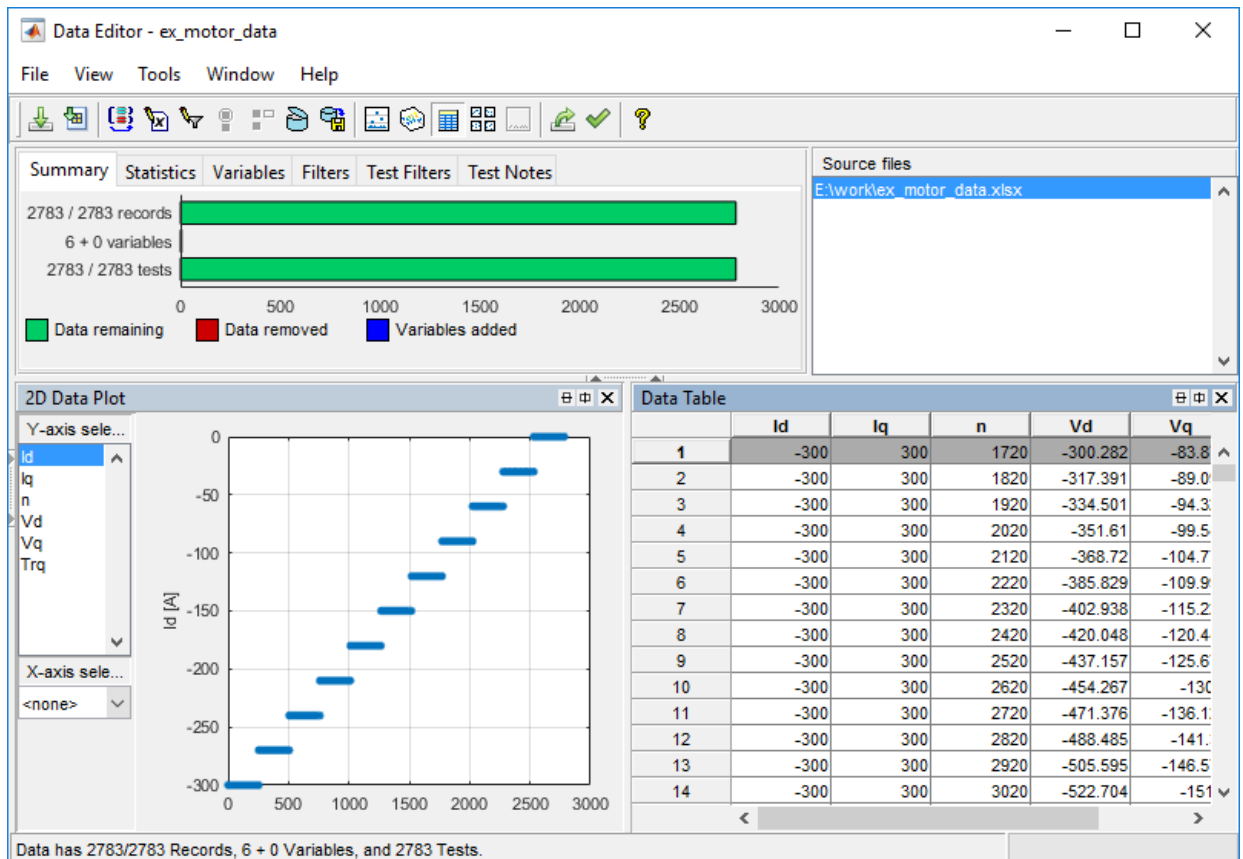
To model the motor data, use the **MBC Model Fitting** app to import, filter, and fit the data with a one-stage model. For this example, the data file `ex_motor_data.xlsx` contains a large data set. You could consider using a design of experiment (DOE) to limit the data. However, the data set represents typical FEA analysis results.

### Import Data

For this example, `ex_motor_data.xlsx` contains this motor controller data:

- $d$ -axis current,  $I_d$ , in A
- $q$ -axis current,  $I_q$ , in A
- Motor speed,  $n$ , in rpm
- $d$ -axis voltage,  $V_d$ , in V

- $q$ -axis voltage,  $V_q$ , in V
  - Electromagnetic motor torque,  $T_e$ , in N.m
- 1 In MATLAB, on the **Apps** tab, in the **Automotive** group, click **MBC Model Fitting**.
  - 2 In the Model Browser home page, click **Import Data**. Click **OK** to open a data source file.
  - 3 Navigate to the `matlab\toolbox\mbc\mbctraining` folder. Open data file `ex_motor_data.xlsx`. The Data Editor opens with your data.



## Filter Data

You can filter data to exclude records from the model fit. In this example, set up a filter to exclude voltage and current magnitudes that are less than a specified threshold. Specifically:

- Voltage magnitude,  $V_s$ , less than or equal to 300 V.
  - Current magnitude,  $I_s$ , less than or equal to 350 A.
- 1 In the Data Editor, select **Tools > Variables** to open the **Variable Editor**. Create these variables. Add units.
    - $I_s = \text{sqrt}(I_d.^2 + I_q.^2)$
    - $V_s = \text{sqrt}(V_d.^2 + V_q.^2)$

Summary	Statistics	Variables (2)	Filters (2)	Test Filters	Test Notes
Variable Expression			Units	Results	
<input checked="" type="checkbox"/>	$I_s = \text{sqrt}(I_d.^2 + I_q.^2)$		A	Variable successfully added.	
<input checked="" type="checkbox"/>	$V_s = \text{sqrt}(V_d.^2 + V_q.^2)$		V	Variable successfully added.	

- 2 In the Data Editor, select **Tools > Filters** to open the **Filter Editor**. Create these filters:
  - $I_s \leq 350$
  - $V_s \leq 300$

Summary	Statistics	Variables (2)	Filters (2)	Test Filters	Test Notes
Filter Expression			Results		
<input checked="" type="checkbox"/>	$I_s \leq 350$		Filter successfully applied : 230 records excluded.		
<input checked="" type="checkbox"/>	$V_s \leq 300$		Filter successfully applied : 2100 records excluded.		

## Fit Model

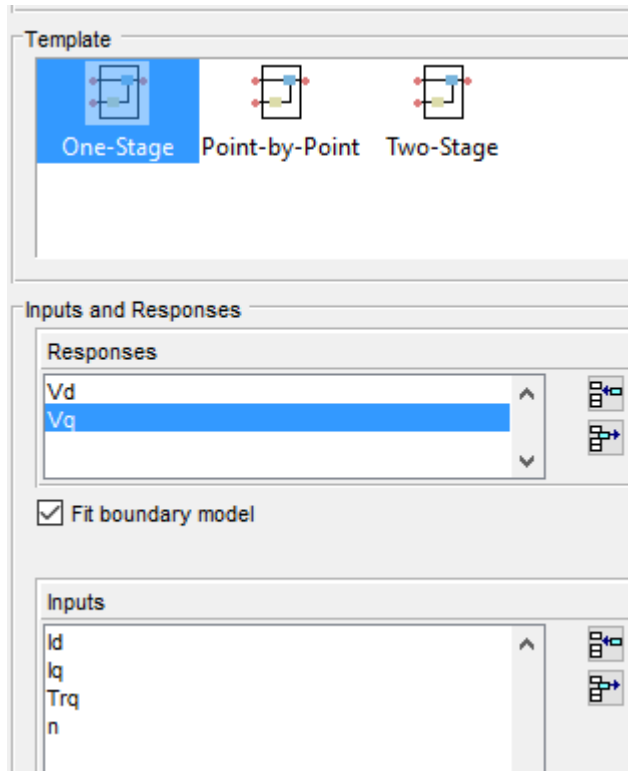
Fit the data to a one-stage 5-dimensional model with these responses and inputs:

- Responses
  - $d$ -axis voltage,  $V_d$ , in V

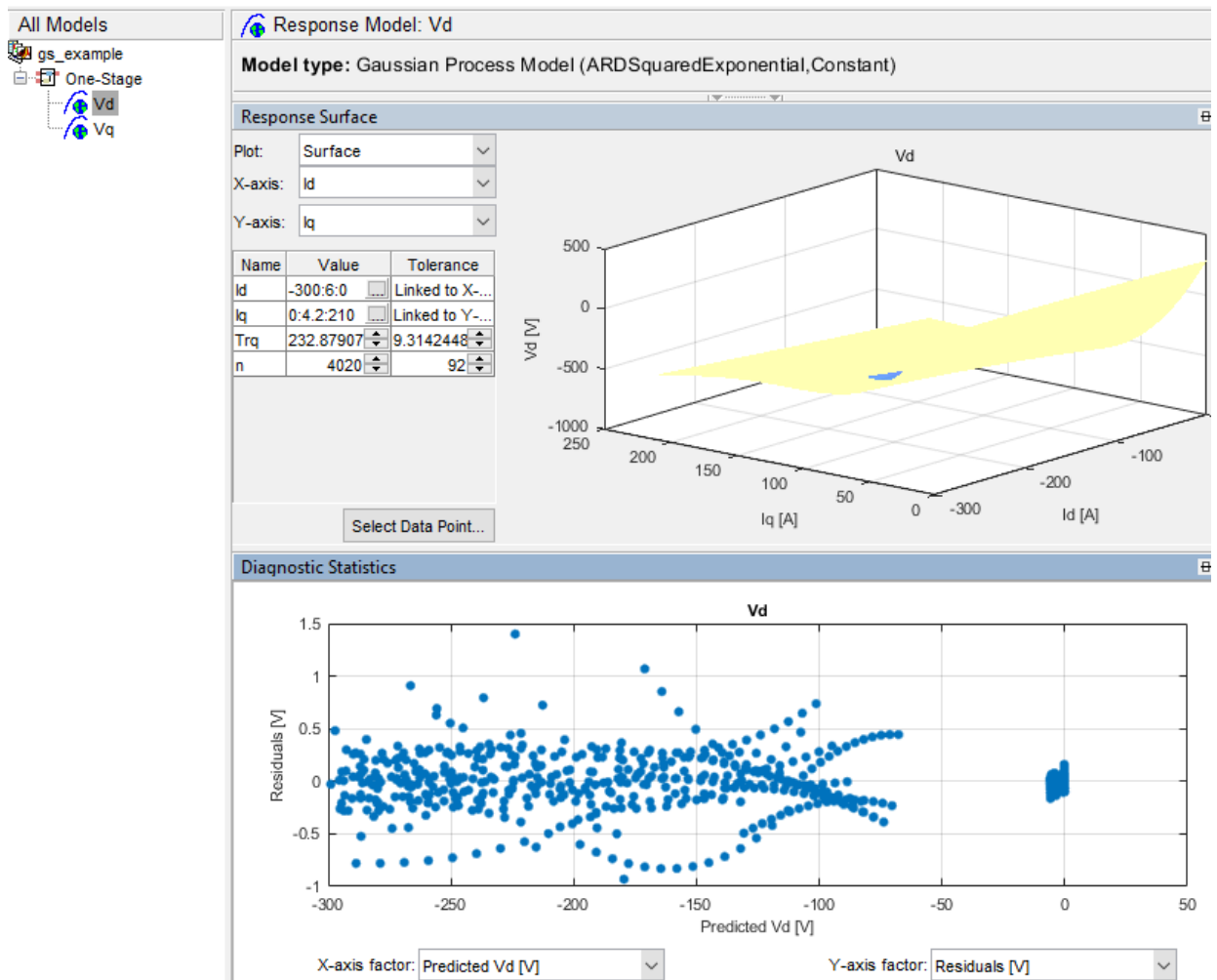
- $q$ -axis voltage,  $V_q$ , in V
- Inputs
  - $d$ -axis current,  $I_d$ , in A
  - $q$ -axis current,  $I_q$ , in A
  - Motor speed,  $n$ , in rpm
  - Electromagnetic motor torque,  $T_e$ , in N.m

- 1 In the Model Browser, select **Fit Models**.
- 2 In **Fit Models**, configure a One-Stage model with these responses and inputs.

Responses	Inputs
Vd	Id
Vq	Iq
	Trq
	n



- 3 To fit the model, select **OK**. If prompted, accept changes to data. By default, the fit uses a Gaussian Process Model (GPM) to fit the data.
- 4 After the fit completes, examine the response models for  $V_d$  and  $V_q$ . The Model Browser displays information that you can use to determine the accuracy of the model fit.
  - In the Model Browser, select  $V_d$ . Examine the response surface and diagnostic statistics. In this example, the response surface indicates that  $V_d$  increases as  $I_d$  approaches 0. The diagnostics indicate that the response residuals are mostly within  $\pm 1$  V. These results indicate a reasonably accurate fit.



- 5 Save your project. For example, select **Files > Save Project**. Save `gs_example.mat` to work folder.

## Generate Calibration

After you fit the model, create functions and tables, run the optimization, and fill the calibration tables.



## Create Functions

Create the functions to use when you optimize the calibration. In this example, set up functions for:

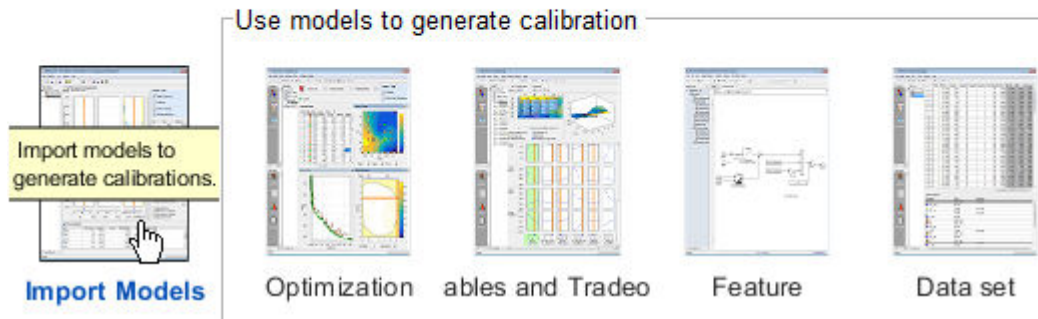
- Voltage magnitude,  $V_s$
  - Current magnitude,  $I_s$
  - Torque per amp,  $TPA$
- 1 In MATLAB, on the **Apps** tab, in the **Automotive** group, click **MBC Optimization**.
  - 2 In the Cage Browser, select **Import Models**. If it is not already opened, in the MBC Model Fitting browser, open the `gs_example.mat` project.



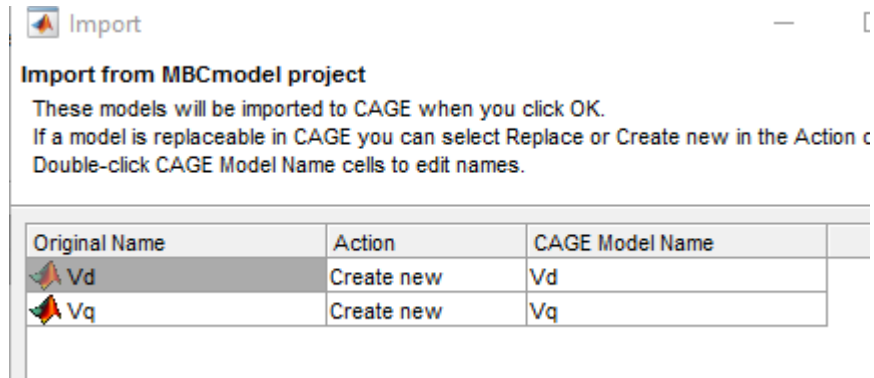
## MBC Model Optimization

Generate optimal look-up tables for model-based calibration.

You have an empty project. Import models to generate calibrations.

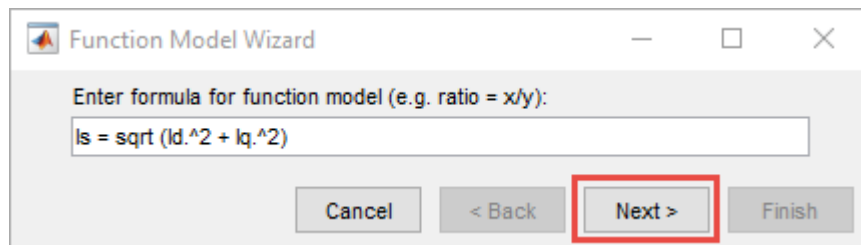


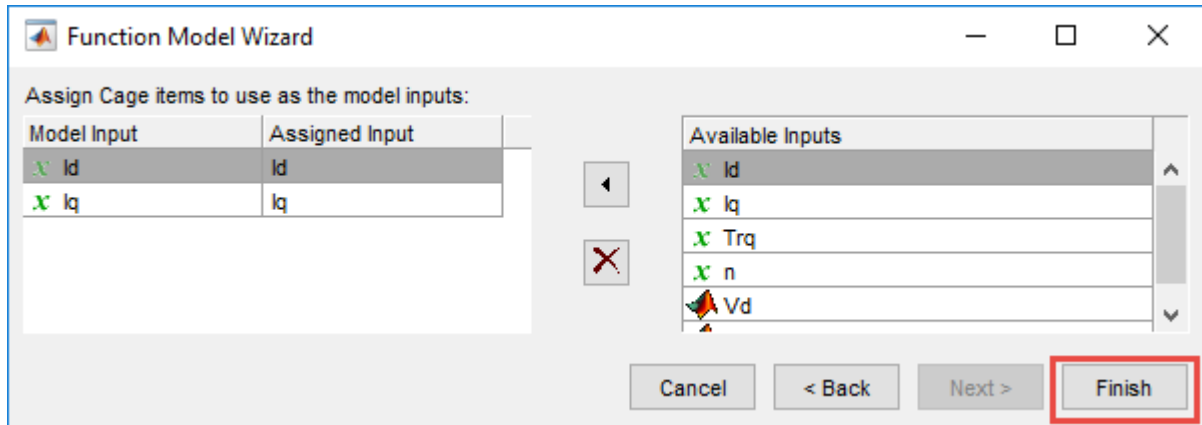
- 3 In Cage Import tool, select **Vd** and **Vq**. Click **Import Selected Items**.
- 4 In Import Models, click **OK**. Close the CAGE Import Tool.



- 5 In the Cage Browser toolbar, use **New Function Model** wizard to create these functions:

- $I_s = \sqrt{I_d.^2 + I_q.^2}$
- $V_s = \sqrt{V_d.^2 + V_q.^2}$
- $TPA = Trq./I_s$





- 6 In the Cage Browser, verify that the function models for  $I_s$ ,  $V_s$ , and TPA have these descriptions.

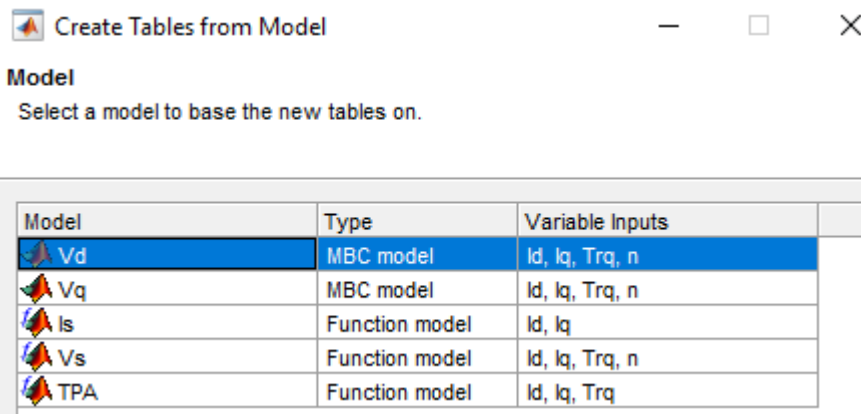
Models					
Name	Type	Inputs	Lower Output Limit	Upper Output Limit	Description
Vd	MBC model	ld, lq, Trq, n	-Inf	Inf	Created by on 21-Apr-2017.
Vq	MBC model	ld, lq, Trq, n	-Inf	Inf	Created by on 21-Apr-2017.
I <sub>s</sub>	Function model	ld, lq	-Inf	Inf	$\sqrt{ld.^2 + lq.^2}$
V <sub>s</sub>	Function model	Vd, Vq	-Inf	Inf	$\sqrt{Vd.^2 + Vq.^2}$
TPA	Function model	I <sub>s</sub> , Trq	-Inf	Inf	Trq/I <sub>s</sub>

- 7 Select **File > Save Project**. Save `gs_example.cag` to the work folder.

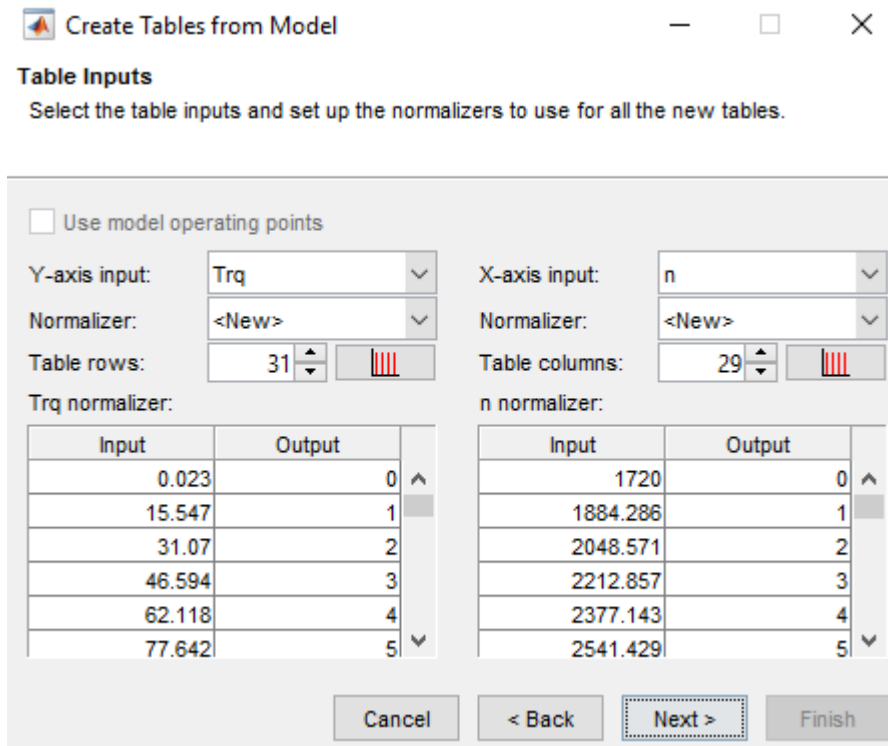
### Create Tables from Model

Create tables that the Model-Based Calibration Toolbox optimizer uses to store the optimized parameters. For this example, the tables are:

- $d$ -axis current,  $I_d$ , as a function of motor torque,  $T_{rq}$ , and motor speed,  $n$ .
  - $q$ -axis current,  $I_q$ , as a function of motor torque,  $T_{rq}$ , and motor speed,  $n$ .
- 1 In the Cage Browser, select **Tables and Tradeoff**. In Create Tables from Model, select Vd. Click **Next**.

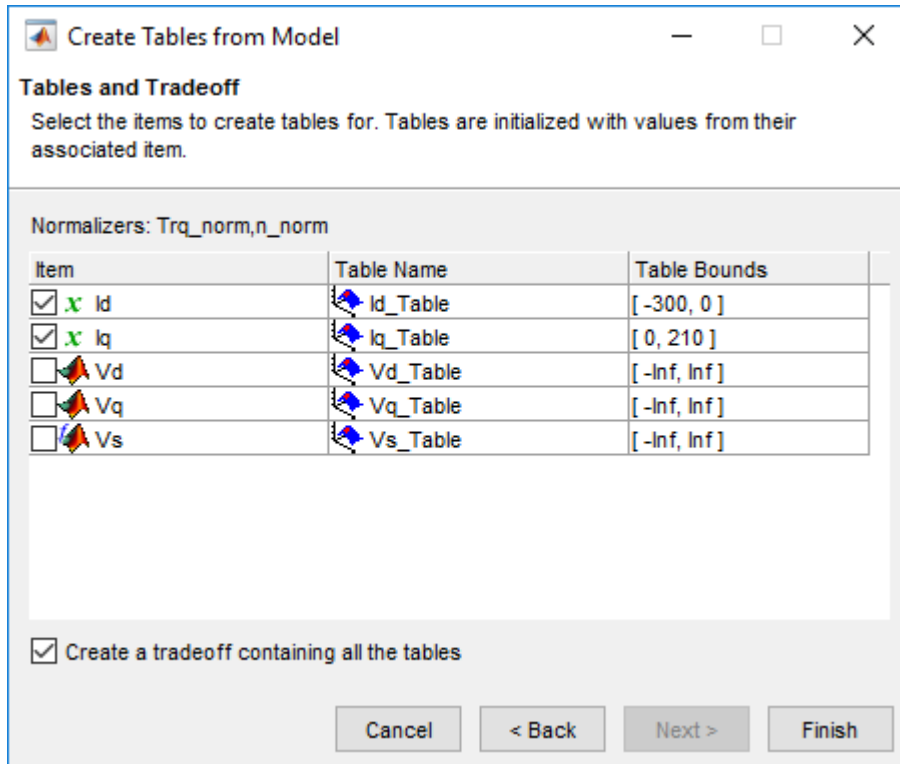


- 2 In **Table Inputs**, set the **Table rows** to 31. Set **Table columns** to 29. Click **Next**.

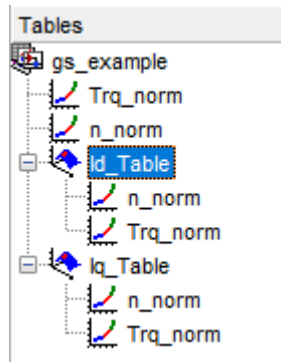


**3** In Create Tables from Model:

- Select Id and Iq.
- Clear Vd.
- Click **Finish**.



**4** In the CAGE Browser, examine the tables.

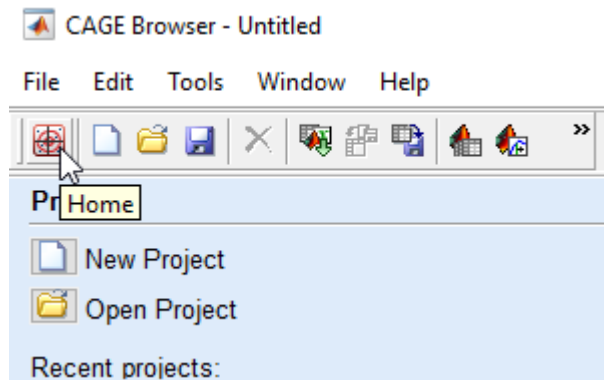


### Run Optimization

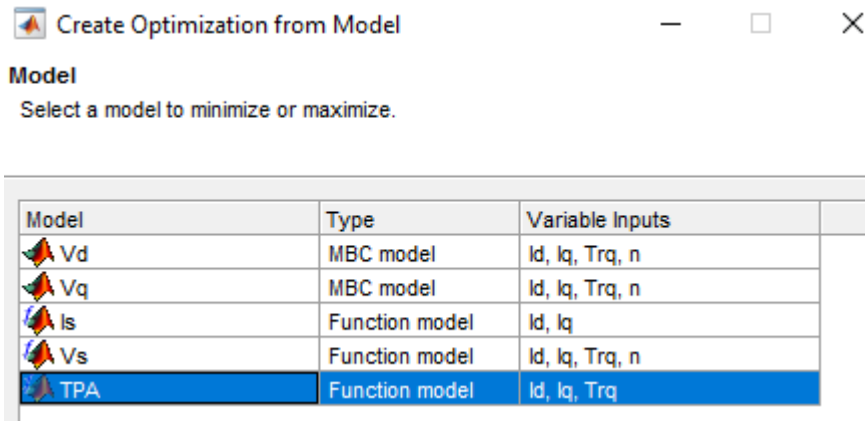
In this example, run a point optimization with these specifications:

- Voltage magnitude,  $V_s$ , less than or equal to 289 V.
- Current magnitude,  $I_s$ , less than or equal to 300 A.
- Maximizes torque per ampere,  $TPA$ .

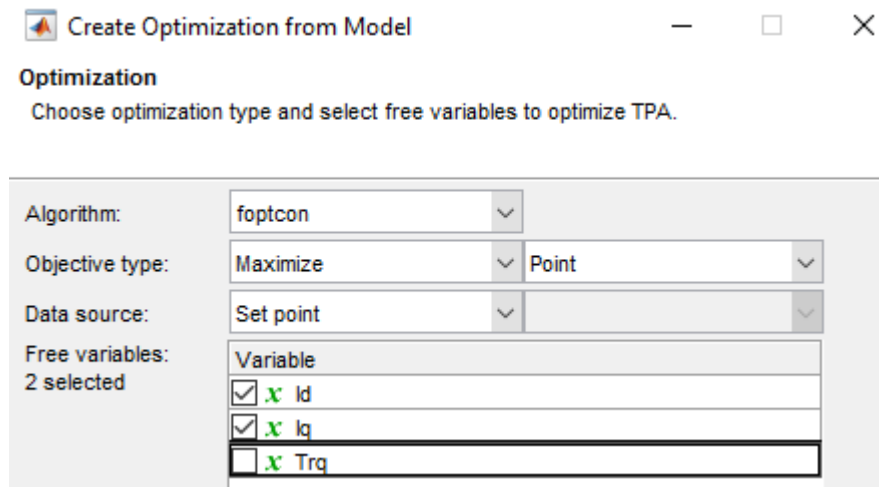
- 1 On the Cage Browser home, select **Optimization**.



- 2 In Create Optimization from Model, select TPA and **Next**.



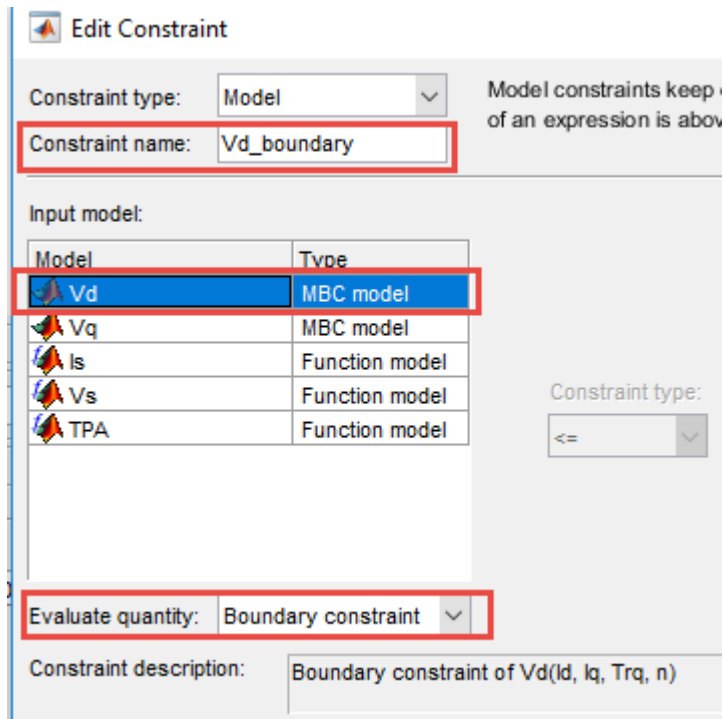
- 3 In Create Optimization from Model:
- Select Id and Iq. Clear Trq.
  - Set **Objective type** to Maximize.
  - Click **Finish**.



- 4 Add a boundary constraint for the Vd model. In the CAGE Browser, select **Optimization > Constraints > Add Constraint**. Set these parameters:
- **Constraint name:** Vd\_boundary

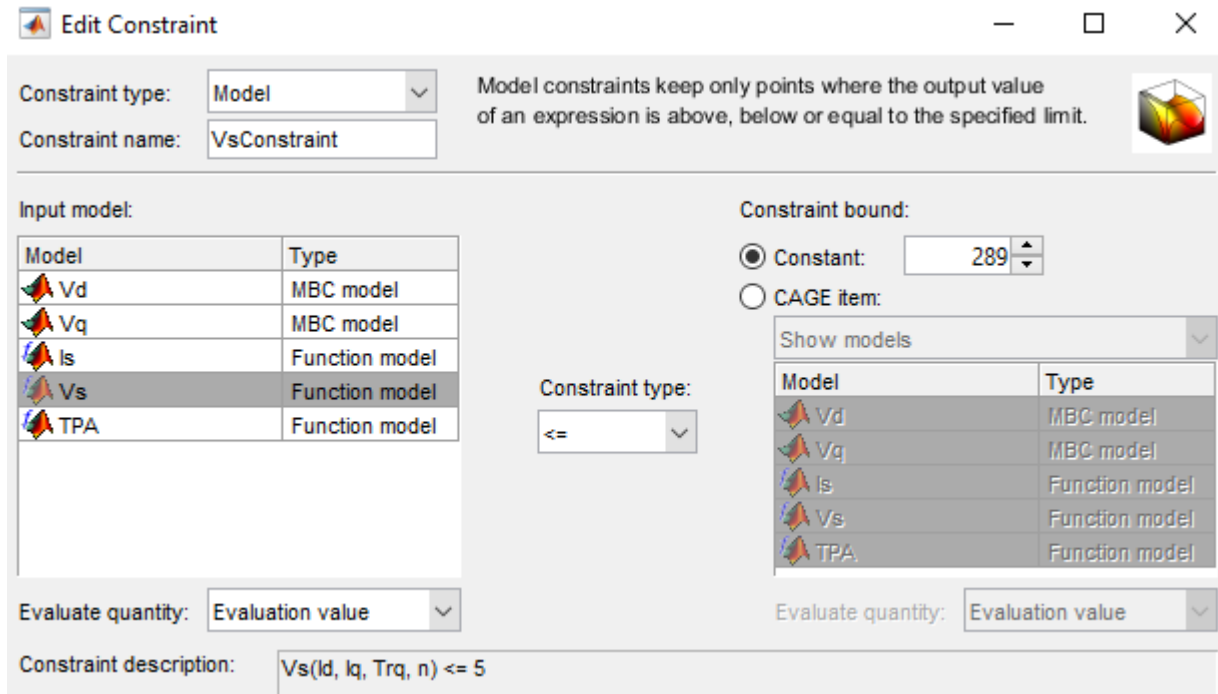
- **Input model:** Vd
- **Evaluate quantity:** Boundary constraint

Verify the settings. Click **OK**.

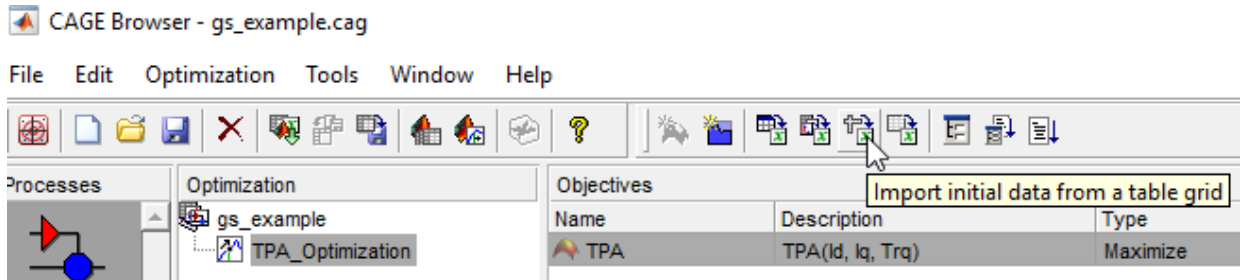


- 5 Add the optimization constraints. In the CAGE Browser, select **Optimization > Constraints > Add Constraints** to open Edit Constraint. Use the dialog box to create constraints on the current and voltage.
  - $I_s \leq 300$
  - $V_s \leq 289$

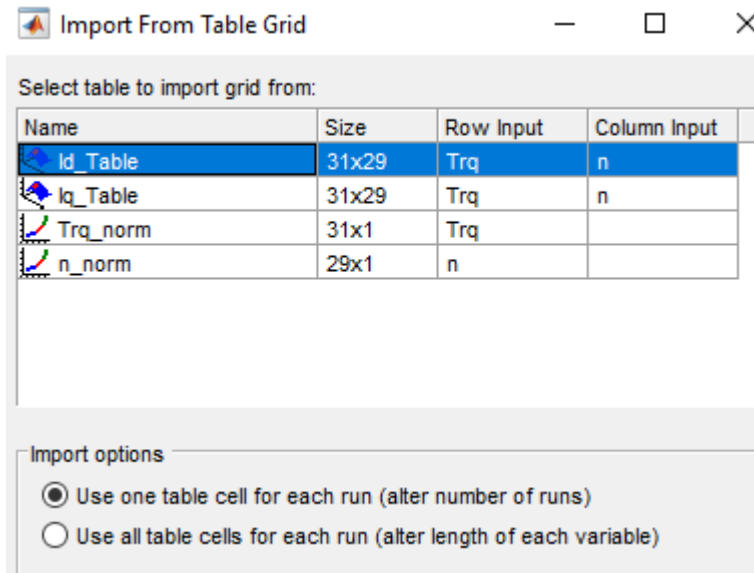




6 In the CAGE Browser, select **Import initial data from a table grid**.



In Import From Table Grid, select Id\_Table.



- 7 In the Cage Browser, *carefully* verify the Objectives and Constraints.

Objectives			
Name	Description	Type	Applicati
TPA	TPA(ld, lq, Trq)	Maximize	

Constraints			
Name	Description	Application Point Set	Status
Vd_boundary	Boundary constraint of Vd(ld, lq...		
IsConstraint	Is(ld, lq) <= 300		
VsConstraint	Vs(ld, lq, Trq, n) <= 289		





Optimization Point Set	
Number of runs:	899
Vector display format:	Expanded vertically

Free Variables			Fixed Variables		
Variable:	ld	lq	Variable:	Trq	n
Number of values:	1	1	Number of values:	1	1
1	-150	105	1	0.023	1720
2	-150	105	2	15.547	1720
3	-150	105	3	31.07	1720
4	-150	105	4	46.594	1720

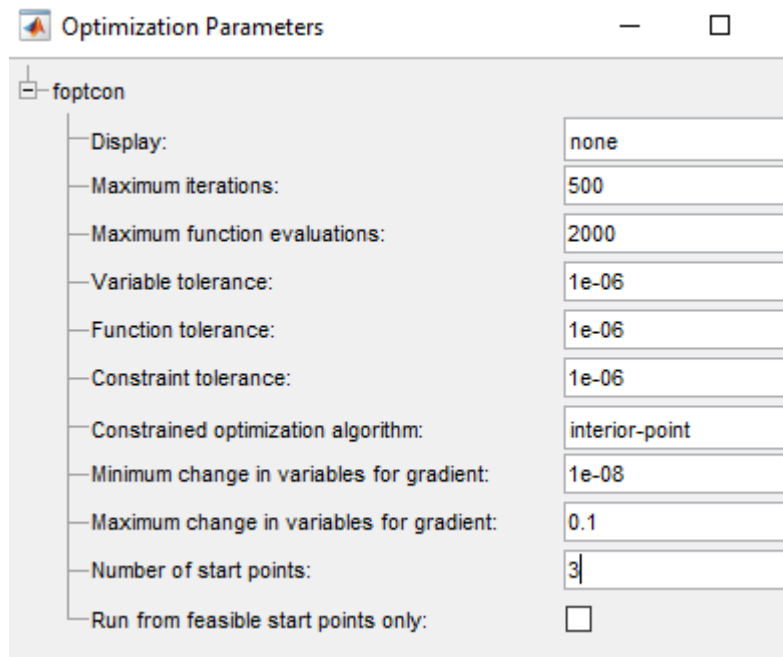
**Common Tasks**

-  Add Constraint...
-  Set Up
-  Run...
-  View Results

**Optimization Information**

Algorithm name	mbcOSfmincon
Algorithm descri...	Single objective optimizati..
Free variables	ld, lq
Operating point ...	None

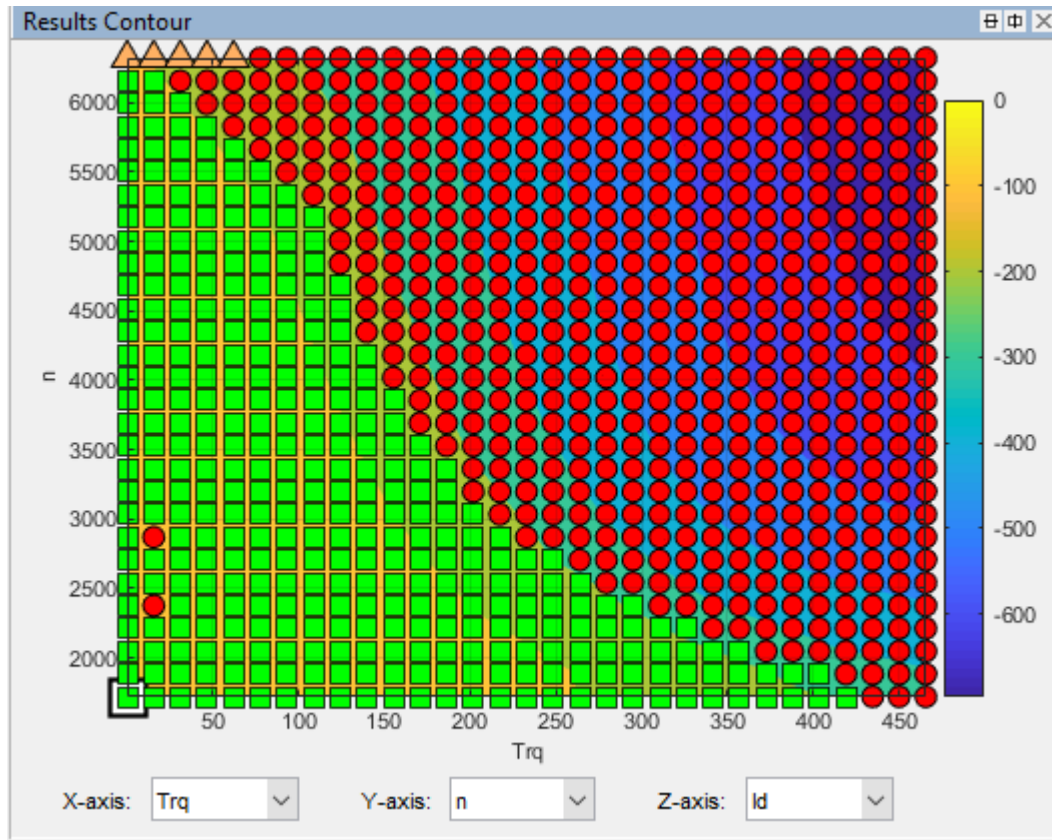
8 In the Cage Browser, select **Set Up**. Set **Number of start points** to 3. Click **Ok**.



- 9 In the Cage Browser, select **Run**. The optimization can take time to run and slow other computer processes. View progress in **Running Optimization**.

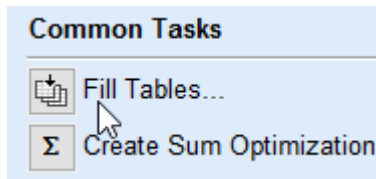
The optimization can take time to run and slow other computer processes. View progress in **Running Optimization**.

The optimization results are similar to these.



## Fill Tables

- 1 In the CAGE Browser, select **Fill Tables**.



- 2 Use the Table Filling from Optimization Results Wizard to fill the Id\_Table and Iq\_Table tables.

Table Filling from Optimization Results Wizard

**Table Selection**

Select the CAGE tables that you wish to fill from the optimization results

Available CAGE tables:

Table

CAGE tables to be filled:

Table
Id_Table(Trq_norm,n_norm)
Iq_Table(Trq_norm,n_norm)

▶

- For the Id\_Table, fill with Id.
- For the Iq\_Table, fill with Iq.

Tables values to be filled:


CAGE Table	Fill with	Tradeoff
Id_Table(Trq_norm,n_no...)	<input checked="" type="checkbox"/> Id	<input checked="" type="checkbox"/> Vd_Tradeoff
Iq_Table(Trq_norm,n_no...)	<input checked="" type="checkbox"/> Iq	<input checked="" type="checkbox"/> Vd_Tradeoff

◀

Optimization Results

- Id
- Iq
- Trq
- n
- Vd

Accept the defaults. Click **Finish**.








 Table Filling from Optimization Results Wizard
**Fill Algorithm**

Set up table filling algorithm.

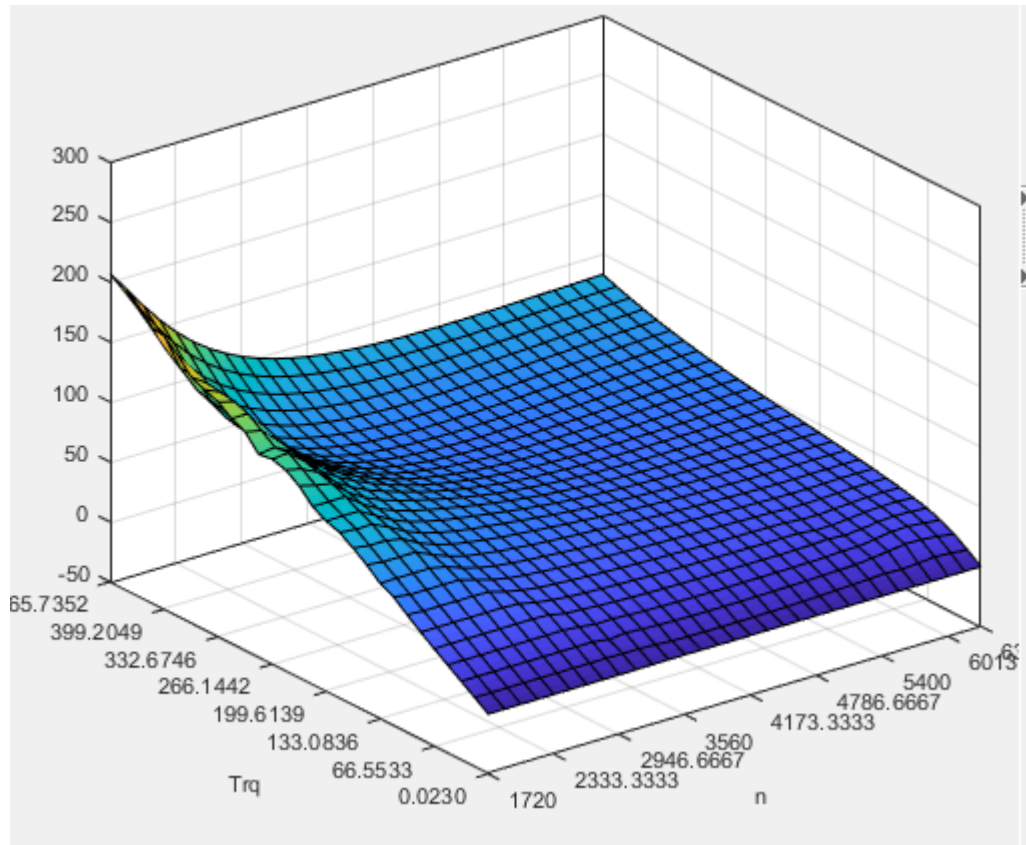
Fill Method:   ...

Use acceptable solutions only       Use locked table values in extrapolation  
 Update tradeoffs       Use existing extrapolation mask in fill

Filter rules for tables:

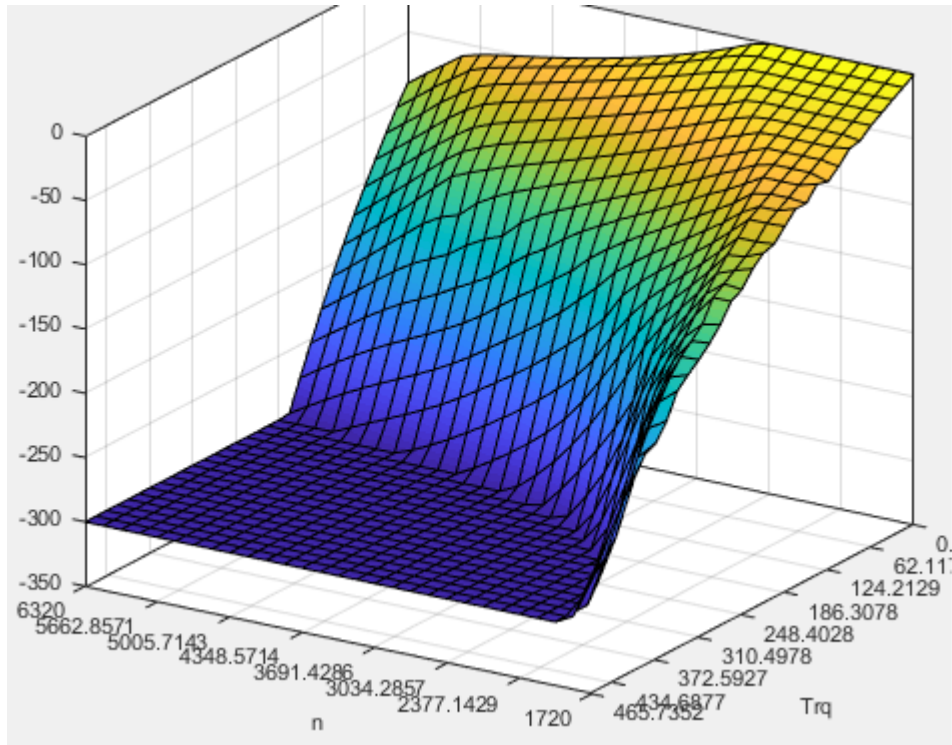
Table	Output Column	Filter Rule	Filter Rule Inputs
 Id_Table	 Id		 Id
 Iq_Table	 Iq		 Iq
			 Trq

- Review results for Iq\_Table. The results are similar to these.



- 4 Review results for `Id_Table`. The results are similar to these.

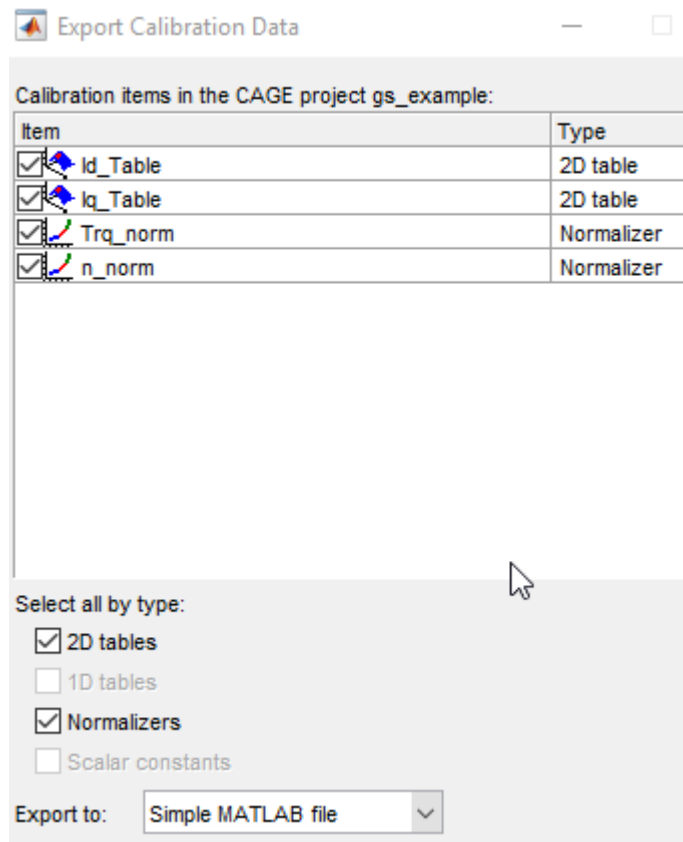




- 5 Select **File > Save Project**. Save `gs_example.cag` to work folder.

### Export Results

- 1 Select **File > Export > Calibration > All Items**.
- 2 Use Export Calibration Data to select the items to export and format. For example, export the Id and Iq tables and breakpoints to MATLAB file `gs_example.m`.



## Set Block Parameters

The optimized current controller calibration tables are functions of motor torque and motor speed. Use the tables for these Flux-Based PM Controller block parameters:

- **Corresponding d-axis current reference,  $id\_ref$**
- **Corresponding q-axis current reference,  $iq\_ref$**
- **Vector of speed breakpoints,  $wbp$**
- **Vector of torque breakpoints,  $tbp$**

To set the block parameters:

- 1 Run the .m file that contains the Model-Based Calibration Toolbox calibration results for the current controller. For example, in the MATLAB command line, run `gs_example.m`:

```
% Access data from MBC current controller calibration
gs_example
```

- 2 Assign the breakpoint parameters to the data contained in the .m file. In this example, the speed data is in rpm. To use the calibration data for the block parameters, convert the speed breakpoints from rpm to rad/s.

Parameter	MATLAB Commands
Vector of speed breakpoints, <code>wbp</code>	<code>tbp=Trq_norm.X;</code>
Vector of speed breakpoints, <code>wbp</code>	<code>% MBC data for speed is in rpm. % For the block parameter, use rad/s nbp=n_norm.X; conversion=(2*pi/60.); wbp=conversion.*nbp;</code>
Corresponding d-axis current reference, <code>id_ref</code>	<code>id_table=Id_Table.Z; id_ref=id_table';</code>
Corresponding q-axis current reference, <code>iq_ref</code>	<code>iq_table=Iq_Table.Z; iq_ref=iq_table';</code>

## Generate Feed-Forward Flux Parameters

Using MathWorks tools, you can create lookup tables for an interior permanent magnet synchronous motor (PMSM) controller that characterizes the  $d$ -axis and  $q$ -axis flux as a function of  $d$ -axis and  $q$ -axis currents.

To generate the flux parameters for the Flux-Based PM Controller block, follow these workflow steps. The steps use example script `VisualizeFluxSurface.m`.

Workflow	Description
“Step 1: Load and Preprocess Data” on page 6-60	Load and preprocess this nonlinear motor flux data from dynamometer testing or finite element analysis (FEA): <ul style="list-style-type: none"> <li>• <math>d</math>- and <math>q</math>- axis current</li> <li>• <math>d</math>- and <math>q</math>- axis flux</li> <li>• Electromagnetic motor torque</li> </ul>
“Step 2: Generate Evenly Spaced Data” on page 6-61	Use spline interpolation to generate evenly spaced data. Visualize the flux surface plots.
“Step 3: Set Block Parameters” on page 6-63	Set workspace variables that you can use for the Flux-Based PM Controller block parameters.

### Step 1: Load and Preprocess Data

Load and preprocess this nonlinear motor flux data from dynamometer testing or finite element analysis (FEA):

- $d$ - and  $q$ - axis current
- $d$ - and  $q$ - axis flux
- Electromagnetic motor torque

- 1 Open the example script `VisualizeFluxSurface.m`.
- 2 Load and preprocess the data.

```
%
% Load the data from a |mat| file captured from a dynamometer or
```

```

% another CAE tool.
load FEAdata.mat;

% Load the data matrix.
lambda_d = FEAdata.flux.d;
lambda_q = FEAdata.flux.q;
id = FEAdata.current.d;
iq = FEAdata.current.q;

```

## Step 2: Generate Evenly Spaced Data

The flux tables can have different step sizes for the currents. Evenly spacing the rows and columns helps improve interpolation accuracy. This script uses spline interpolation.

- 1 Set the spacing for the table rows and columns.

```

% Set the spacing for the table rows and columns
flux_d_size = 50;
flux_q_size = 50;

```

- 2 Use spline interpolation to get higher resolution.

```

% Use spline interpolation to get higher resolution
id_new = linspace(min(id),max(id),flux_d_size);
iq_new = linspace(min(iq),max(iq),flux_q_size);
lambda_d_new = interp2(id',iq,lambda_d,id_new',iq_new,'spline');
lambda_q_new = interp2(id',iq,lambda_q,id_new',iq_new,'spline');

```

- 3 Visualize the flux surfaces.

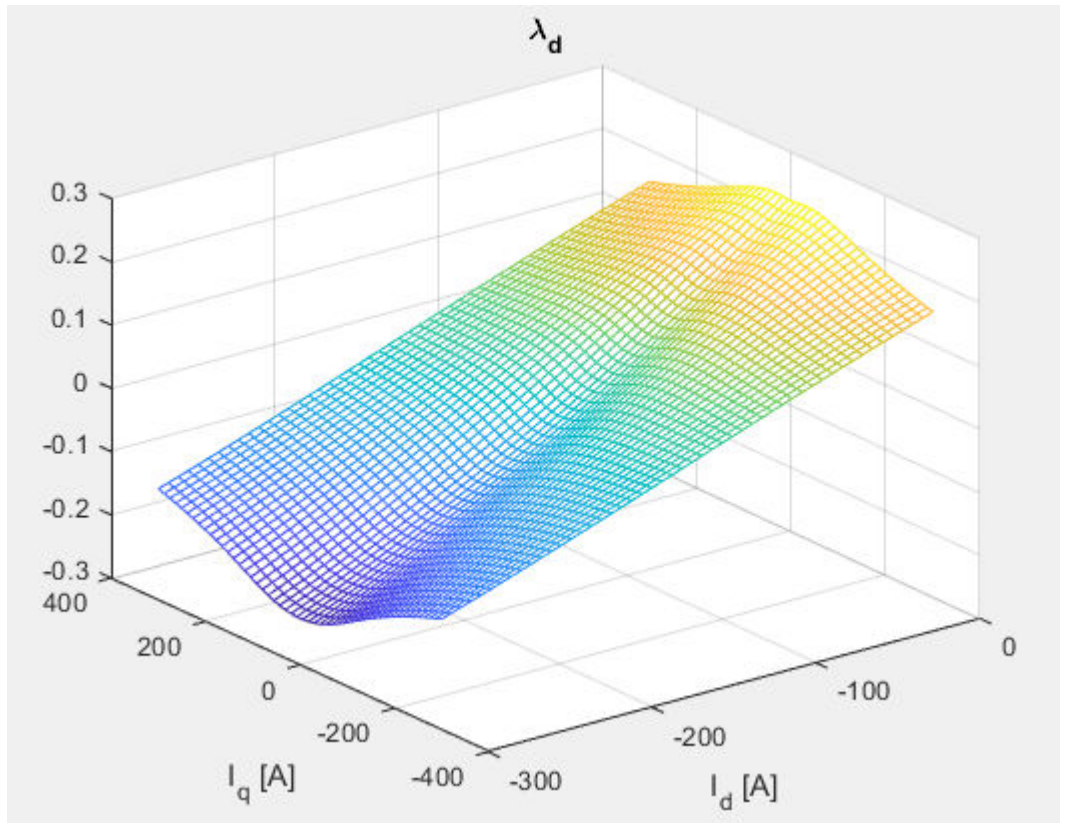
```

% Visualize the flux surface
figure;
mesh(id_new,iq_new,lambda_d_new);
xlabel('I_d [A]')
ylabel('I_q [A]')
title('\lambda_d'); grid on;

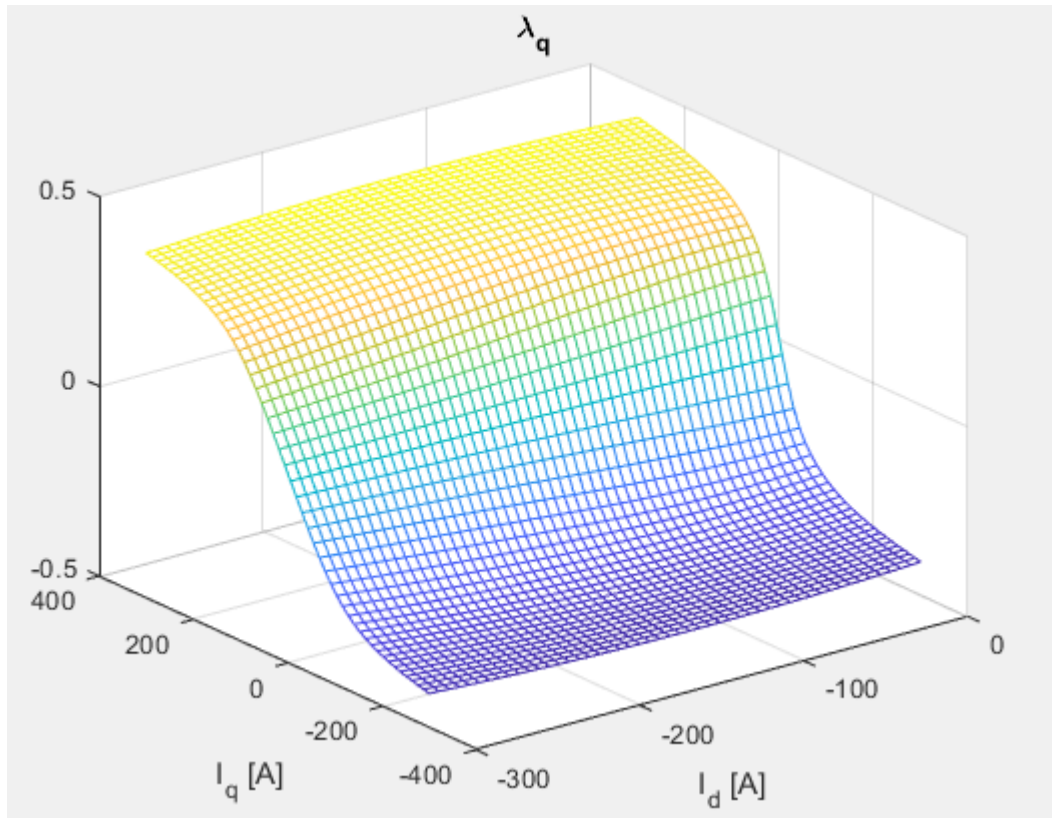
figure;
mesh(id_new,iq_new,lambda_q_new);
xlabel('I_d [A]')
ylabel('I_q [A]')
title('\lambda_q'); grid on;

```

- d-axis flux,  $\lambda_d$ , as a function of d-axis current,  $I_d$ , and q-axis current,  $I_q$ .



- q-axis flux,  $\lambda_q$ , as a function of d-axis current,  $I_d$ , and q-axis current,  $I_q$ .



### Step 3: Set Block Parameters

Set the block parameters to these values assigned in the example script.

Parameter	MATLAB Commands
Vector of d-axis current breakpoints, $id\_index$	<code>id_index=id_new;</code>
Vector of q-axis current breakpoints, $iq\_index$	<code>iq_index=iq_new;</code>
Corresponding d-axis flux, $\lambda_{d\_d}$	<code>lambda_d=lambda_d_new;</code>
Corresponding q-axis flux, $\lambda_{d\_q}$	<code>lambda_q=lambda_q_new;</code>

## References

- [1] Hu, Dakai, Yazan Alsmadi, and Longya Xu. "High fidelity nonlinear IPM modeling based on measured stator winding flux linkage." *IEEE Transactions on Industry Applications*, Vol. 51, No. 4, July/August 2015.
  
- [2] Chen, Xiao, Jiabin Wang, Bhaskar Sen, Panagiotis Lasari, Tianfu Sun. "A High-Fidelity and Computationally Efficient Model for Interior Permanent-Magnet Machines Considering the Magnetic Saturation, Spatial Harmonics, and Iron Loss Effect." *IEEE Transactions on Industrial Electronics*, Vol. 62, No. 7, July 2015.

## See Also

Flux-Based PMSM | Flux-Based PM Controller



## Generate Parameters for Flux-Based PMSM Block

Using MathWorks tools, you can create lookup tables for an interior permanent magnet synchronous motor (PMSM) controller that characterizes the  $d$ -axis and  $q$ -axis current as a function of  $d$ -axis and  $q$ -axis flux.

To generate the flux parameters for the Flux-Based PMSM block, follow these workflow steps. Example script `CreatingIdqTable.m` calls `gridfit` to model the current surface using scattered or semi-scattered flux data.

Workflow	Description
“Step 1: Load and Preprocess Data” on page 6-65	Load and preprocess this nonlinear motor flux data from dynamometer testing or finite element analysis (FEA): <ul style="list-style-type: none"> <li>• <math>d</math>- and <math>q</math>- axis current</li> <li>• <math>d</math>- and <math>q</math>- axis flux</li> <li>• Electromagnetic motor torque</li> </ul>
“Step 2: Generate Evenly Spaced Table Data From Scattered Data” on page 6-67	Use the <code>gridfit</code> function to generate evenly spaced data. Visualize the flux surface plots.
“Step 3: Set Block Parameters” on page 6-69	Set workspace variables that you can use for the Flux-Based PM Controller block parameters.

### Step 1: Load and Preprocess Data

Load and preprocess this nonlinear motor flux data from dynamometer testing or finite element analysis (FEA):

- $d$ - and  $q$ - axis current
- $d$ - and  $q$ - axis flux
- Electromagnetic motor torque

- 1 Open the example script `CreatingIdqTable.m`.
- 2 Load and preprocess the data.

```
% Load the data from a |mat| file captured from a dynamometer or  
% another CAE tool.
```

```
load FEAdata.mat;
```

- 3** Determine the minimum and maximum flux values.

```
flux_d_min = min(min(FEAdata.flux.d)) ;
```

```
flux_d_max = max(max(FEAdata.flux.d)) ;
```

```
flux_q_min = min(min(FEAdata.flux.q)) ;
```

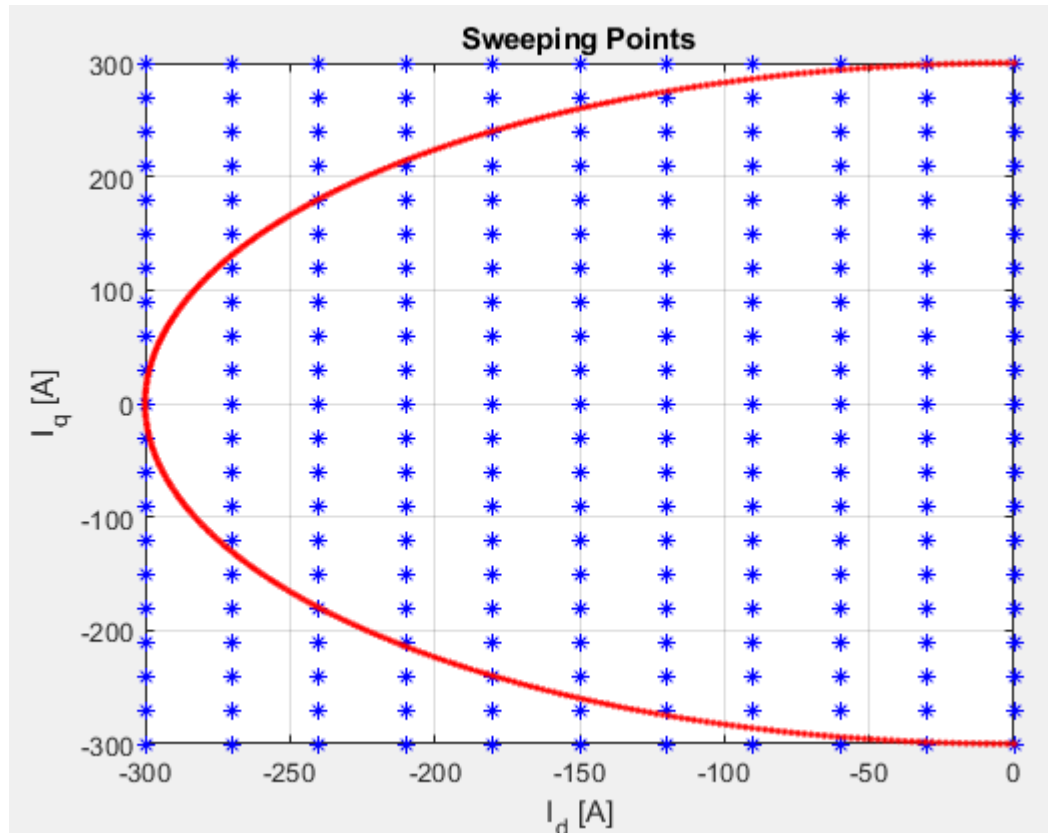
```
flux_q_max = max(max(FEAdata.flux.q)) ;
```

- 4** Plot the sweeping current points used to collect the data.

```
for i = 1:length(FEAdata.current.d)  
    for j = 1:1:length(FEAdata.current.q)  
        plot(FEAdata.current.d(i),FEAdata.current.q(j), 'b*');  
        hold on  
    end  
end
```

- 5** Plot the current limit sweeping points and circle.

```
for angle_theta = pi/2:(pi/2/200):(3*pi/2)  
    plot(300*cos(angle_theta),300*sin(angle_theta), 'r. ');  
    hold on  
end  
xlabel('I_d [A]')  
ylabel('I_q [A]')  
title('Sweeping Points'); grid on;  
xlim([-300,0]);  
ylim([-300,300]);  
hold off
```



## Step 2: Generate Evenly Spaced Table Data From Scattered Data

The flux tables and can have different step sizes for the currents. Evenly spacing the rows and columns helps improve interpolation accuracy. This script uses spline interpolation.

- 1 Set the spacing for the table rows and columns.

```
% Set the spacing for the table rows and columns
flux_d_size = 50;
flux_q_size = 50;
```

- 2 Generate linear spaced vectors for the breakpoints.

```

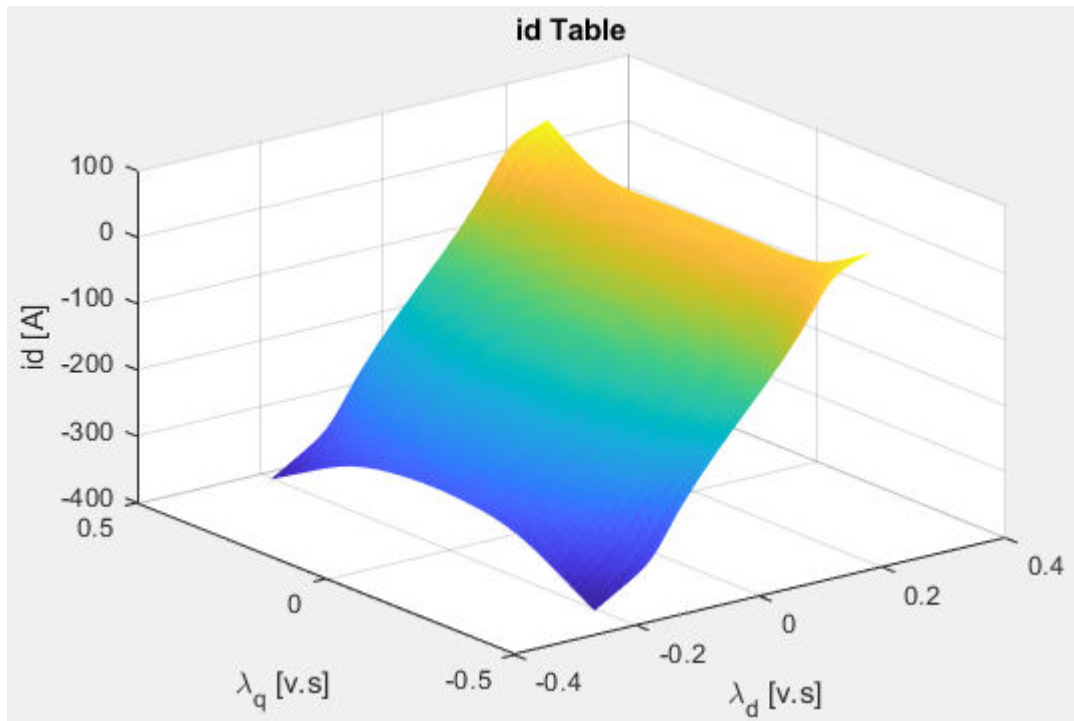
% Generate linear spaced vectors for the breakpoints
ParamFluxDIndex = linspace(flux_d_min,flux_d_max,flux_d_size);
ParamFluxQIndex = linspace(flux_q_min,flux_q_max,flux_q_size);
3 Create 2-D grid coordinates based on the d-axis and q-axis currents.

% Create 2-D grid coordinates based on the d-axis and q-axis currents
[id_m,iq_m] = meshgrid(FEAdata.current.d,FEAdata.current.q);
4 Create the table for the d-axis current.

% Create the table for the d-axis current
id_fit = gridfit(FEAdata.flux.d,FEAdata.flux.q,id_m,ParamFluxDIndex,ParamFluxQIndex);
ParamIdLookupTable = id_fit';
figure;
surf(ParamFluxDIndex,ParamFluxQIndex,ParamIdLookupTable');
xlabel('\lambda_d [v.s]');ylabel('\lambda_q [v.s]');zlabel('id [A]');title('id Table');
shading flat;

```

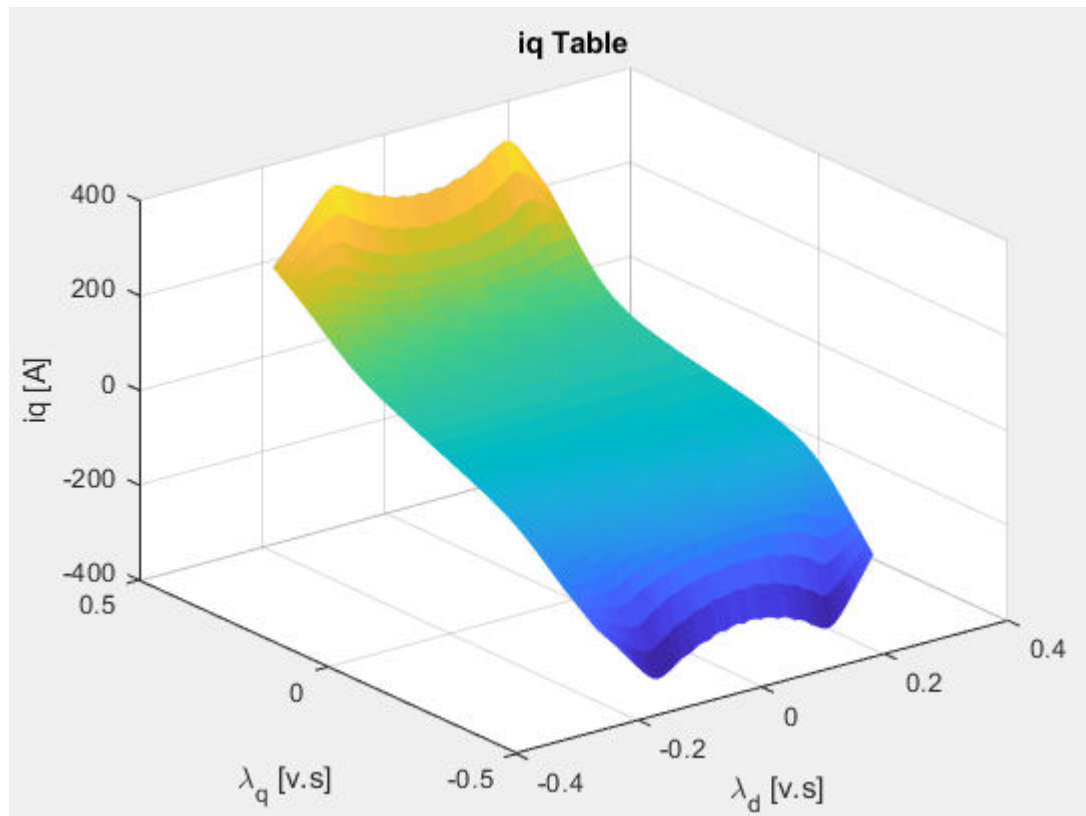
d-axis current,  $I_d$ , as a function of q-axis flux,  $\lambda_q$ , and d-axis flux,  $\lambda_d$ .



- 5 Create the table for the  $q$ -axis current.

```
% Create the table for the q-axis current
iq_fit = gridfit(FEAdata.flux.d,FEAdata.flux.q,iq_m,ParamFluxDIndex,ParamFluxQIndex);
ParamIqLookupTable = iq_fit';
figure;
surf(ParamFluxDIndex,ParamFluxQIndex,ParamIqLookupTable');
xlabel('\lambda_d [v.s]');ylabel('\lambda_q [v.s]');zlabel('iq [A]'); title('iq Tab');
shading flat;
```

$q$ -axis current,  $I_q$ , as a function of  $q$ -axis flux,  $\lambda_q$ , and  $d$ -axis flux,  $\lambda_d$ .



### Step 3: Set Block Parameters

Set the block parameters to these values assigned in the example script.

Parameter	MATLAB Commands
Vector of d-axis flux, <code>flux_d</code>	<code>flux_d=ParamFluxDIndex;</code>
Vector of q-axis flux, <code>flux_q</code>	<code>flux_q=ParamFluxQIndex;</code>
Corresponding d-axis current, <code>id</code>	<code>id=ParamIdLookupTable;</code>
Corresponding q-axis current, <code>iq</code>	<code>iq=ParamIqLookupTable;</code>

## References

- [1] Hu, Dakai, Yazan Alsmadi, and Longya Xu. "High fidelity nonlinear IPM modeling based on measured stator winding flux linkage." *IEEE Transactions on Industry Applications*, Vol. 51, No. 4, July/August 2015.
- [2] Chen, Xiao, Jiabin Wang, Bhaskar Sen, Panagiotis Lasari, Tianfu Sun. "A High-Fidelity and Computationally Efficient Model for Interior Permanent-Magnet Machines Considering the Magnetic Saturation, Spatial Harmonics, and Iron Loss Effect." *IEEE Transactions on Industrial Electronics*, Vol. 62, No. 7, July 2015.

## See Also

Flux-Based PMSM | Flux-Based PM Controller

## External Websites

- Surface Fitting using `gridfit`

# Powertrain Blockset Examples

---

## SI Engine Dynamometer Reference Application

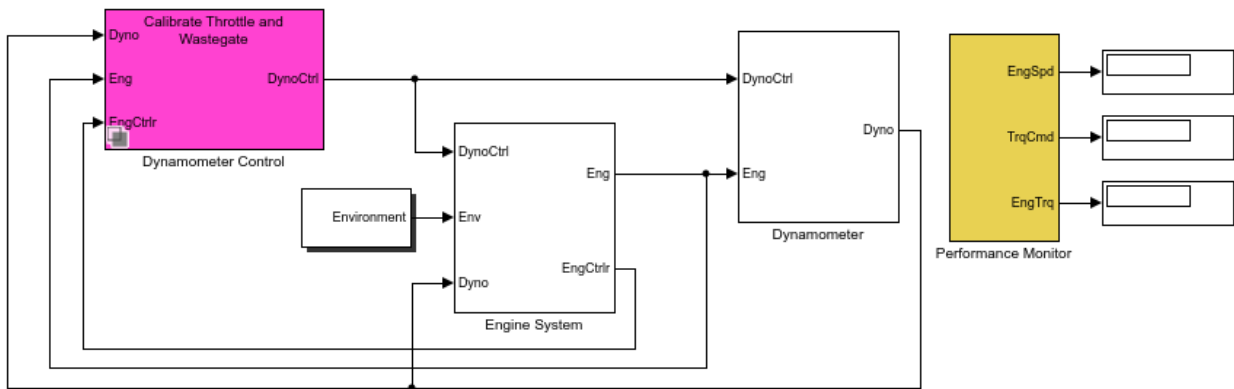
This example shows how to create a spark-ignition (SI) engine dynamometer project. You can use the project as a starting point for simulating a SI engine and controller under a dynamometer test harness.

### Open the SI Engine Dynamometer Reference Application Project

Run the following command to create and open a working copy of the project files:

```
autoblkSIDynamometerStart
```

## Engine Dynamometer



Execute Engine Mapping Experiment

Execute Model Predictive Control Plant Model Experiment

Recalibrate Controller

Resize Engine and Recalibrate Controller

Help

Generate Mapped Engine from Spreadsheet



For more information, see Explore the SI Engine Dynamometer Reference Application.

```
helpview('autoblks', 'siengine_dyno_refapp')
```

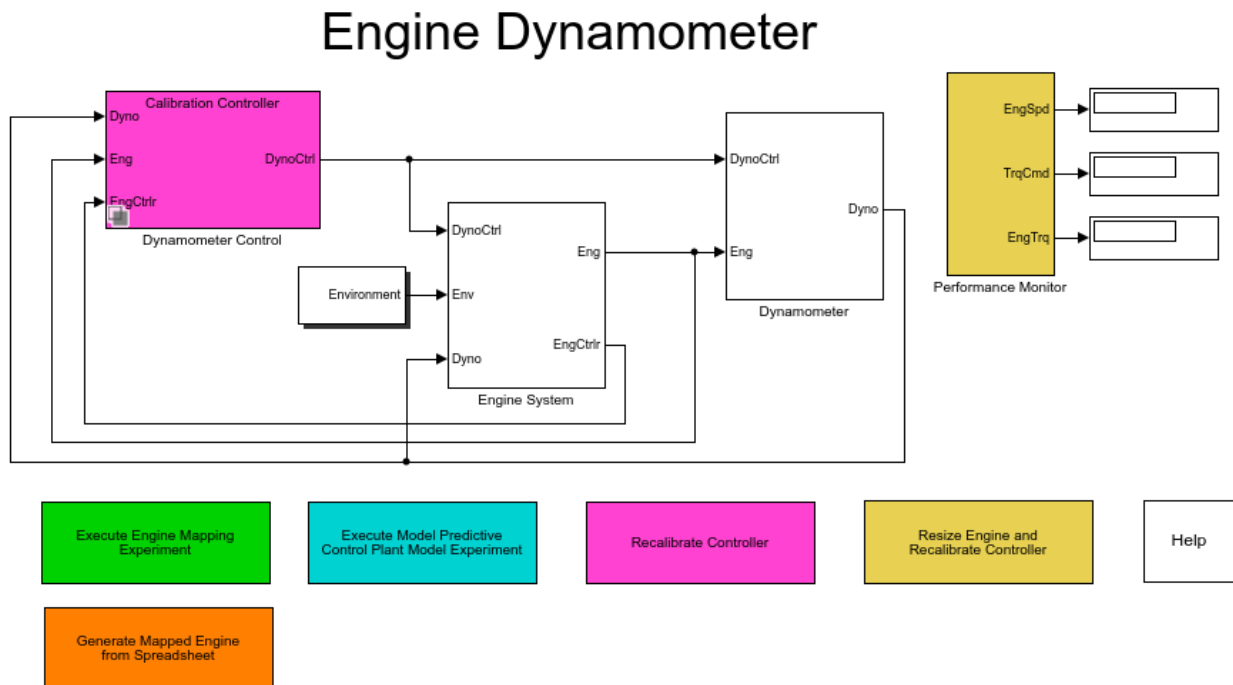
## CI Engine Dynamometer Reference Application

This example shows how to create a compression-ignition (CI) engine dynamometer project. You can use the project as a starting point for simulating a CI engine and controller under a dynamometer test harness.

### Open the CI Engine Dynamometer Reference Application Project

Run the following command to create and open a working copy of the project files:

```
autoblkCIDynamometerStart
```



For more information, see [Explore the CI Engine Dynamometer Reference Application](#).

```
helpview('autoblks', 'ciengine_dyno_refapp')
```

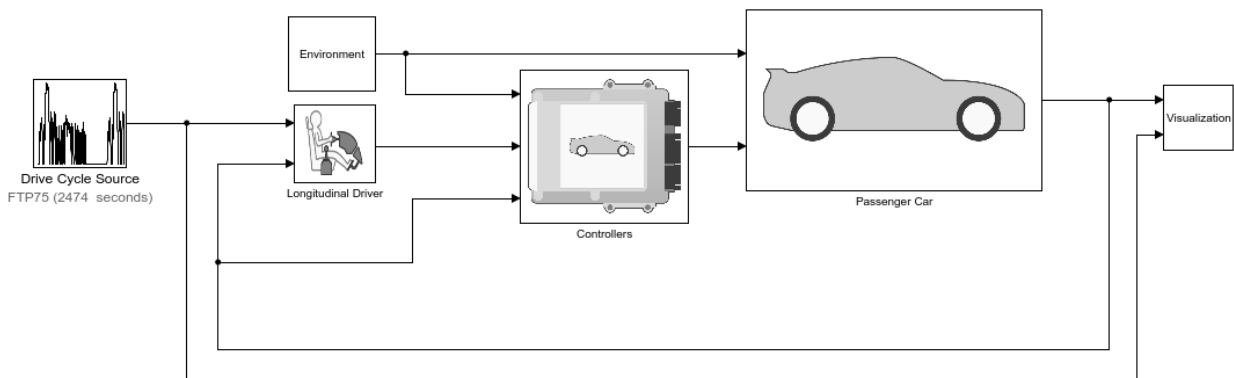
## Electric Vehicle Reference Application

This example shows how to create an electric vehicle reference application project.

### Open the Electric Vehicle Reference Application Project

Run the following command to create and open a working copy of the project files:

```
autoblkEvStart
```



Copyright 2015-2016 The MathWorks, Inc.

For more information, see [Explore the Electric Vehicle Reference Application](#).

```
helpview('autoblks', 'ev_refapp')
```

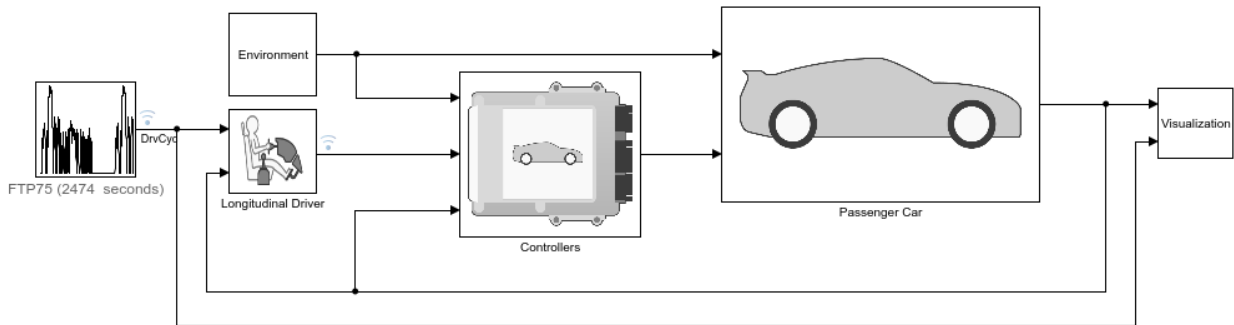
# Hybrid Electric Vehicle Input Power-Split Reference Application

This example shows how to create a hybrid electric vehicle input power-split reference application project.

## Open the Hybrid Electric Vehicle Input Power-Split Reference Application Project

Run the following command to create and open a working copy of the project files:

```
autoblkHevIpsStart
```



Copyright 2015-2017 The MathWorks, Inc.

For more information, see Explore the Hybrid Electric Vehicle Input Power-Split Reference Application.

```
helpview('autoblks', 'split_hybrid_refapp')
```

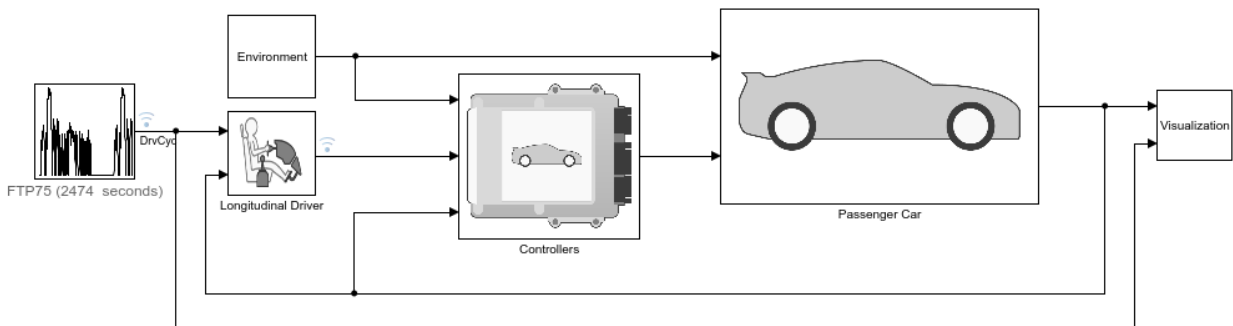
## Hybrid Electric Vehicle P2 Reference Application

This example shows how to create a hybrid electric vehicle P2 reference application project.

### Open the Hybrid Electric Vehicle P2 Reference Application Project

Run the following command to create and open a working copy of the project files:

```
autoblkHevP2Start
```



Copyright 2015-2017 The MathWorks, Inc.

For more information, see [Explore the Hybrid Electric Vehicle P2 Reference Application](#).

```
helpview('autoblks', 'p2_hybrid_refapp')
```

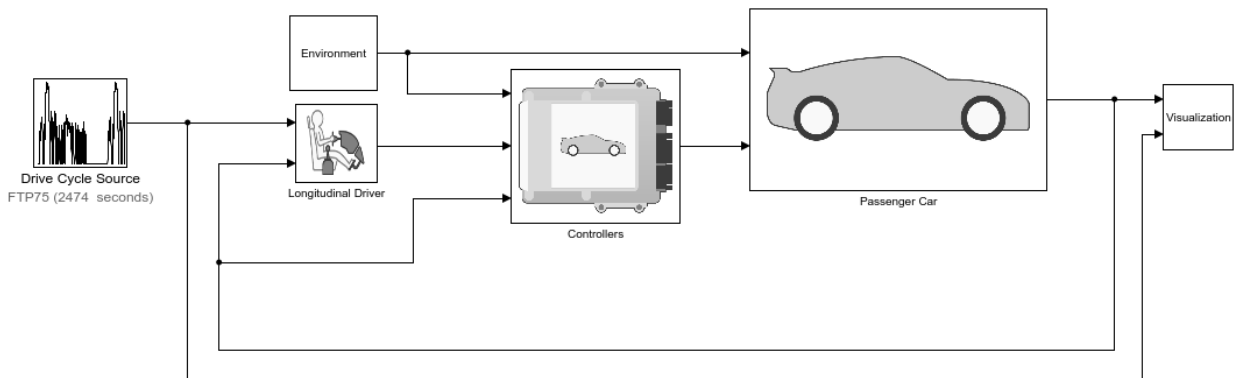
# Hybrid Electric Vehicle Multimode Reference Application

This example shows how to create a hybrid electric vehicle multimode reference application project.

## Open the Hybrid Electric Vehicle Multimode Reference Application Project

Run the following command to create and open a working copy of the project files:

```
autoblkHevStart
```



Copyright 2015-2016 The MathWorks, Inc.

For more information, see [Explore the Hybrid Electric Vehicle Multimode Reference Application](#).

```
helpview('autoblks', 'hybrid_refapp')
```

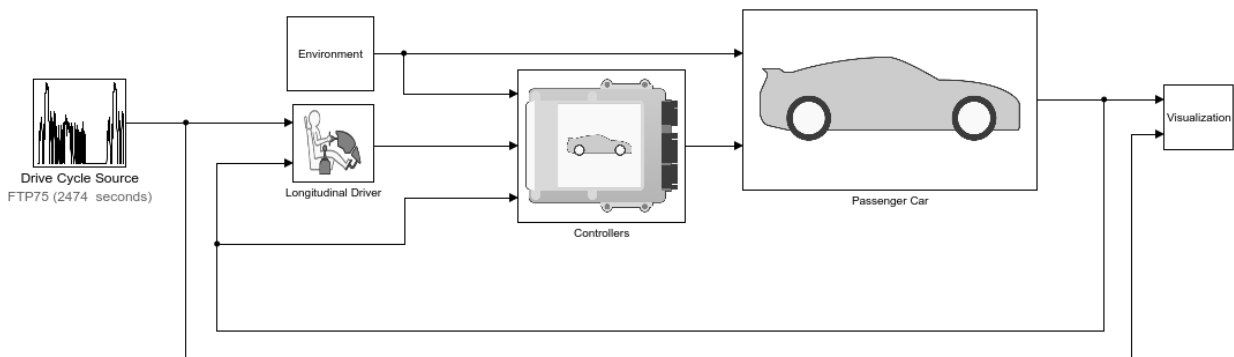
## Conventional Vehicle Reference Application

This example shows how to create a conventional vehicle reference application project.

### Open the Conventional Vehicle Reference Application Project

Run the following command to create and open a working copy of the project files:

```
autoblkConVehStart
```



Copyright 2015-2016 The MathWorks, Inc.

For more information, see [Explore the Conventional Vehicle Reference Application](#).

```
helpview('autoblks', 'conveh_refapp')
```