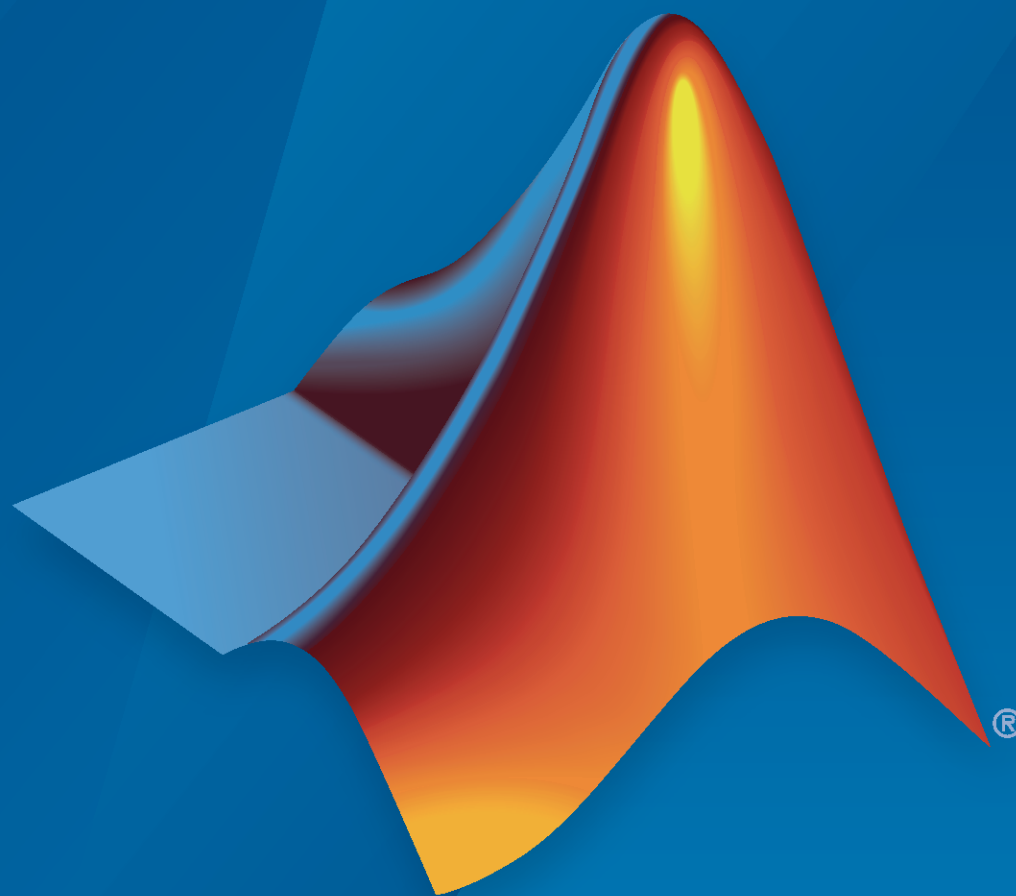


Powertrain Blockset™

User's Guide



MATLAB® & SIMULINK®

R2021a



How to Contact MathWorks



Latest news: www.mathworks.com
Sales and services: www.mathworks.com/sales_and_services
User community: www.mathworks.com/matlabcentral
Technical support: www.mathworks.com/support/contact_us



Phone: 508-647-7000



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

Powertrain Blockset™ User's Guide

© COPYRIGHT 2016–2021 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 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 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)
March 2019	Online only	Revised for Version 1.5 (Release 2019a)
September 2019	Online only	Revised for Version 1.6 (Release 2019b)
March 2020	Online only	Revised for Version 1.7 (Release 2020a)
September 2020	Online only	Revised for Version 1.8 (Release 2020b)
March 2021	Online only	Revised for Version 1.9 (Release 2021a)

1	Getting Started	
	Powertrain Blockset Product Description	1-2
	Key Features	1-2
	Required and Recommended Products	1-3
	Required Products	1-3
	Recommended Products	1-3
	Getting Started with Powertrain Blockset	1-4
	Next Steps	1-9
	Conventional Vehicle Spark-Ignition Engine Fuel Economy and Emissions	1-10
	Conventional Vehicle Powertrain Efficiency	1-15

	Workflows	
2	SI Core Engine Air Mass Flow and Torque Production	2-2
	Air Mass Flow Models	2-2
	Torque Models	2-3
	SI Engine Dual-Independent Cam Phaser Air Mass Flow Model	2-5
	Collect Physical Measurements	2-6
	Estimate Ideal Trapped Mass	2-7
	Correct Trapped Mass	2-7
	Calculate Air Mass Flow	2-8
	SI Engine Speed-Density Air Mass Flow Model	2-11
	SI Engine Torque Structure Model	2-14
	SI Engine Simple Torque Model	2-20
	CI Core Engine Air Mass Flow and Torque Production	2-21
	Air Mass Flow	2-21
	Torque	2-21
	CI Engine Speed-Density Air Mass Flow Model	2-22

CI Engine Torque Structure Model	2-25
Fuel Injection	2-27
Percent Oxygen	2-27
Exhaust Temperature	2-27
CI Engine Simple Torque Model	2-30
Engine Calibration Maps	2-31
Engine Plant Calibration Maps	2-31
Engine Controller Calibration Maps	2-31
Calibration Maps in Compression-Ignition (CI) Blocks	2-31
Calibration Maps in Spark-Ignition (SI) Blocks	2-55

Reference Applications

3

Internal Combustion Engine Reference Application Projects	3-2
Hybrid and Electric Vehicle Reference Application Projects	3-3
Explore the Conventional Vehicle Reference Application	3-4
Optimize Transmission Shift Maps	3-5
Evaluate and Report Power and Energy	3-6
Drive Cycle Source	3-6
Longitudinal Driver	3-6
Controllers	3-7
Passenger Car	3-8
Explore the CI Engine Dynamometer Reference Application	3-10
Engine System	3-11
Performance Monitor	3-12
Explore the SI Engine Dynamometer Reference Application	3-14
Engine System	3-15
Performance Monitor	3-16
Explore the Hybrid Electric Vehicle Multimode Reference Application	3-18
Evaluate and Report Power and Energy	3-20
Drive Cycle Source	3-20
Longitudinal Driver	3-20
Controllers	3-21
Passenger Car	3-23
Explore the Electric Vehicle Reference Application	3-25
Evaluate and Report Power and Energy	3-27
Drive Cycle Source	3-27
Longitudinal Driver	3-27
Controllers	3-28
Passenger Car	3-28

Explore the Hybrid Electric Vehicle Input Power-Split Reference Application	3-31
Evaluate and Report Power and Energy	3-33
Drive Cycle Source	3-33
Longitudinal Driver	3-33
Controllers	3-34
Passenger Car	3-37
Explore the Hybrid Electric Vehicle P0 Reference Application	3-40
Evaluate and Report Power and Energy	3-42
Drive Cycle Source	3-42
Longitudinal Driver	3-42
Controllers	3-43
Passenger Car	3-44
Limitations	3-45
Acknowledgment	3-45
Explore the Hybrid Electric Vehicle P1 Reference Application	3-47
Evaluate and Report Power and Energy	3-49
Drive Cycle Source	3-49
Longitudinal Driver	3-49
Controllers	3-50
Passenger Car	3-51
Limitations	3-52
Acknowledgment	3-52
Explore the Hybrid Electric Vehicle P2 Reference Application	3-54
Evaluate and Report Power and Energy	3-56
Drive Cycle Source	3-56
Longitudinal Driver	3-56
Controllers	3-57
Passenger Car	3-60
Limitations	3-61
Acknowledgment	3-61
Explore the Hybrid Electric Vehicle P3 Reference Application	3-63
Evaluate and Report Power and Energy	3-65
Drive Cycle Source	3-65
Longitudinal Driver	3-65
Controllers	3-66
Passenger Car	3-67
Limitations	3-68
Acknowledgment	3-68
Explore the Hybrid Electric Vehicle P4 Reference Application	3-70
Evaluate and Report Power and Energy	3-72
Drive Cycle Source	3-72
Longitudinal Driver	3-72
Controllers	3-73
Passenger Car	3-74
Limitations	3-75
Acknowledgment	3-75
Resize the CI Engine	3-77
Create CI Engine Models with Twice the Power	3-77

Resize the SI Engine	3-84
Create SI Engine Models with Twice the Power	3-84
Generate Mapped CI Engine from a Spreadsheet	3-91
Step 1: Generate Mapped Engine Calibration	3-91
Step 2: Apply Calibration to Mapped Engine Model	3-94
Generate Mapped SI Engine from a Spreadsheet	3-96
Step 1: Generate Mapped Engine Calibration	3-96
Step 2: Apply Calibration to Mapped Engine Model	3-98
Generate a Deep Learning SI Engine Model	3-100
Internal Combustion Mapped and Dynamic Engine Models	3-106
Analyze Power and Energy	3-107
Live Script	3-107
Power Signals	3-108

Project Templates

4

CI Engine Project Template	4-2
Controller	4-2
Plant	4-2
SI Engine Project Template	4-4
Controller	4-4
Plant	4-4

Supporting Data

5

Install Drive Cycle Data	5-2
Track Drive Cycle Errors	5-3

Calibration

6

Generate Parameter Data for Datasheet Battery Block	6-2
Generate Parameter Data for Equivalent Circuit Battery Block	6-14
Step 1: Load and Preprocess Data	6-15
Step 2: Determine the Number of RC Pairs	6-17
Step 3: Estimate Parameters	6-18

Step 4: Set Equivalent Circuit Battery Block Parameters	6-24
Generate Parameters for Flux-Based Blocks	6-26
Generate Current Controller Parameters	6-28
Collect and Post Process Motor Data	6-29
Model Motor Data	6-30
Generate Calibration	6-34
Set Block Parameters	6-47
Generate Feed-Forward Flux Parameters	6-49
Step 1: Load and Preprocess Data	6-49
Step 2: Generate Evenly Spaced Data	6-49
Step 3: Set Block Parameters	6-51
Generate Parameters for Flux-Based PMSM Block	6-53
Step 1: Load and Preprocess Data	6-53
Step 2: Generate Evenly Spaced Table Data From Scattered Data	6-54
Step 3: Set Block Parameters	6-56

Powertrain Blockset Examples

7

Conventional Vehicle Reference Application	7-2
HEV Multimode Reference Application	7-3
HEV Input Power-Split Reference Application	7-4
HEV P0 Reference Application	7-5
HEV P1 Reference Application	7-6
HEV P2 Reference Application	7-7
HEV P3 Reference Application	7-8
HEV P4 Reference Application	7-9
EV Reference Application	7-10
CI Engine Dynamometer Reference Application	7-11
SI Engine Dynamometer Reference Application	7-12
Optimize Transmission Control Module Shift Schedules	7-13
Calibrate ECMS Block	7-17
Use On-Board Diagnostics to Detect Misfire	7-28

Read and Write Block Parameters to Excel	7-31
---	-------------

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 Parallel 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 3-18
	HEV — Input power-split	“Explore the Hybrid Electric Vehicle Input Power-Split Reference Application” on page 3-31
	Full electric vehicle	“Explore the Electric Vehicle Reference Application” on page 3-25
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 3-14

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.

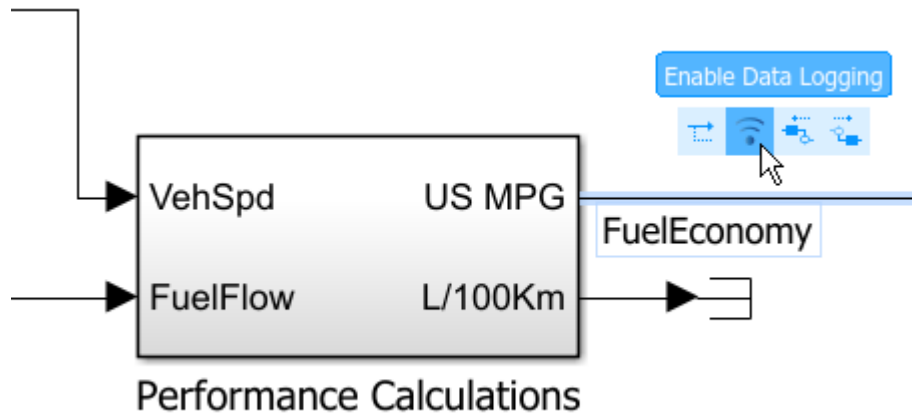
Running this example requires a Stateflow license. You can install a Stateflow trial license using the Add-On Explorer.

- 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.

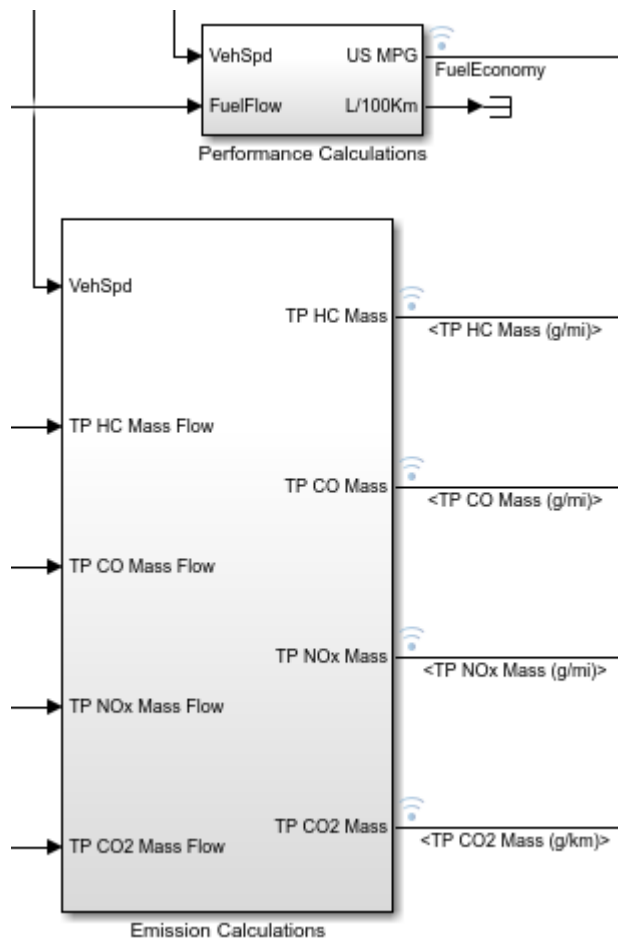
```
autoblckConVehStart
```

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.



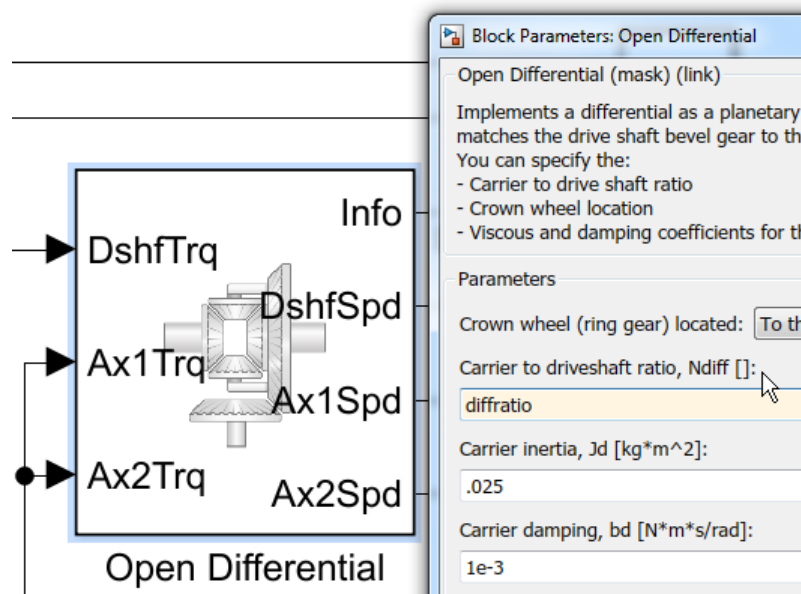
- b** In the Visualization subsystem, enable data logging on the tailpipe emissions signals.



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

b In the Open Differential block mask:

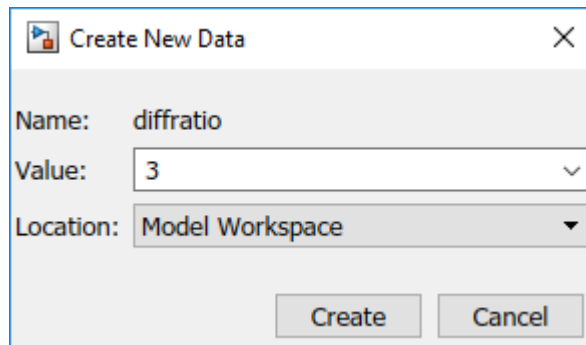
- Change the **Carrier to driveshaft ratio, Ndiff** parameter to the variable `diffratio`. The **Carrier to driveshaft ratio, Ndiff** parameter represents the final drive gear ratio.



- Use the available actions to create new data.

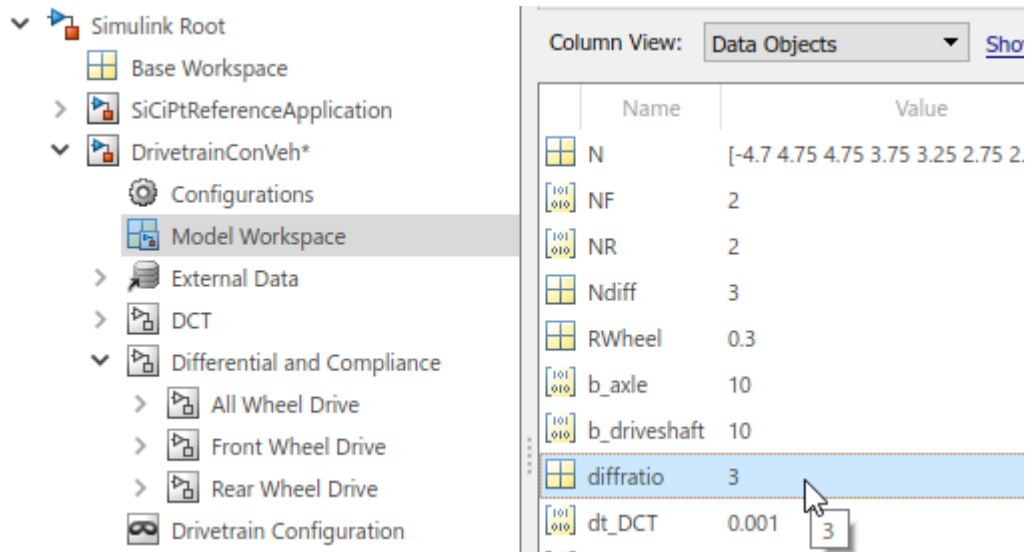



- Use the Create New Data dialog box to create a Model Workspace parameter `diffratio` equal to a value of 3.

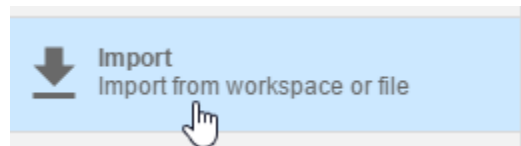


- In the Open Differential block mask, apply the change.

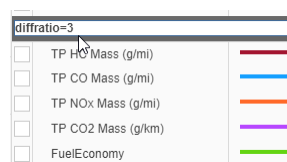
c In the Model Explorer, for the DrivetrainConVeh model, confirm that the `diffratio` parameter is set to 3.



- d Save the `DrivetrainConVeh` 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 **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 `DrivetrainConVeh` model, set the Model Workspace `diffratio` parameter to 2.5.
- b Save the `DrivetrainConVeh` 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_x, CO₂) 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 Spark-Ignition Engine Fuel Economy and Emissions” on page 1-10
- “Conventional Vehicle Powertrain Efficiency” on page 1-15

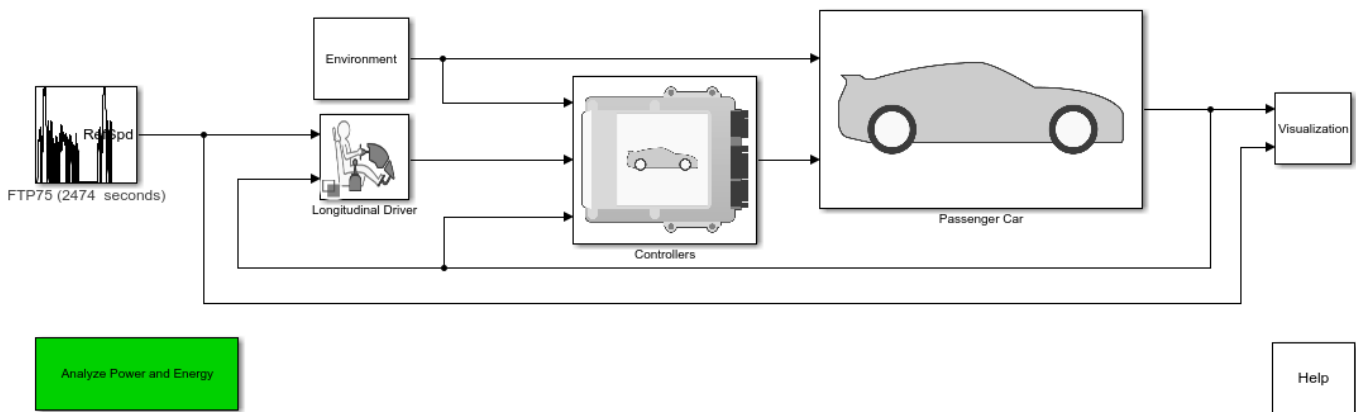
More About

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

Conventional Vehicle Spark-Ignition Engine Fuel Economy and Emissions

This example shows how to calculate the city and highway fuel economy and the emissions for a conventional vehicle with a 1.5-L spark-ignition (SI) engine. To run this example, make sure you have the city (FTP75) and the highway (HWFET) drive cycles installed. After you open the conventional vehicle reference application, open the Drive Cycle Source block and click **Install additional drive cycles**. For more information, see “Install Drive Cycle Data” on page 5-2.

```
setupconvehMPG;
```



Copyright 2015-2020 The MathWorks, Inc.

Prepare the Conventional Vehicle Reference Application For Simulation

Name the Drive Cycle Source block and Visualization subsystem.

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

In the Visualization subsystem, log the emissions signal data.

```
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');
```

Run City Drive Cycle Simulation

Configure the Drive Cycle Source block to run the city drive cycle (FTP75).

```
set_param(dcs, 'cycleVar', 'FTP75');
```

Run a simulation of the city drive cycle. View the results in the Performance and FE Scope.

```
tfinal = get_param(dcs, 'tfinal');
tf = tfinal(1:strfind(tfinal, ' '));
```

```
simout1 = sim(model, 'ReturnWorkspaceOutputs', 'on', 'StopTime', tf);
open_system('SiCiPtReferenceApplication/Visualization/Performance and FE Scope')
```

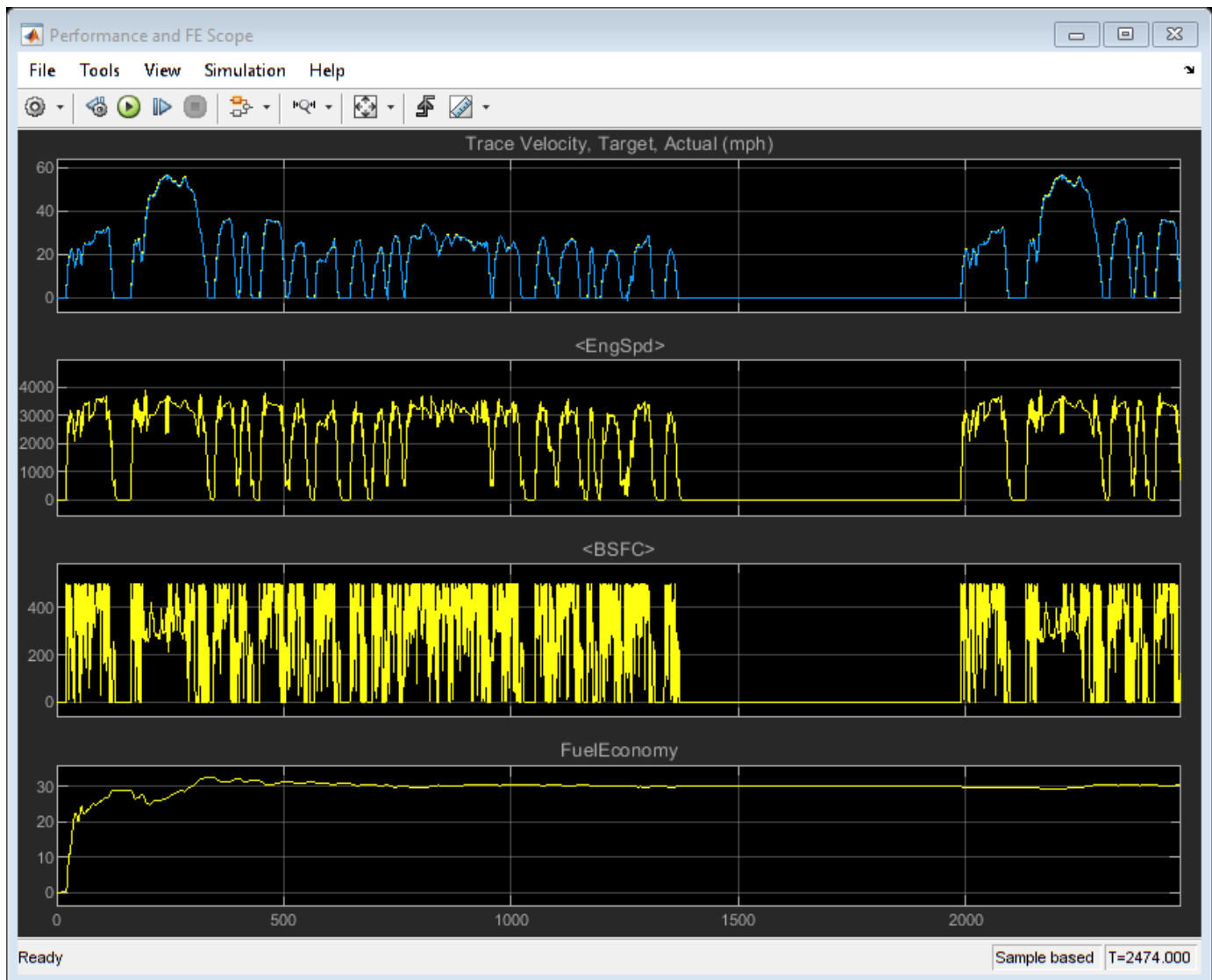
```
### Starting serial model reference simulation build
### Successfully updated the model reference simulation target for: DrivetrainConVeh
### Successfully updated the model reference simulation target for: PowertrainMaxPowerController
### Successfully updated the model reference simulation target for: SiEngineController
### Successfully updated the model reference simulation target for: SiMappedEngine
```

Build Summary

Simulation targets built:

Model	Action	Rebuild Reason
DrivetrainConVeh	Code generated and compiled	DrivetrainConVeh_msf.mexw64 does not exist
PowertrainMaxPowerController	Code generated and compiled	PowertrainMaxPowerController_msf.mexw64 does not exist
SiEngineController	Code generated and compiled	SiEngineController_msf.mexw64 does not exist
SiMappedEngine	Code generated and compiled	SiMappedEngine_msf.mexw64 does not exist

4 of 4 models built (0 models already up to date)
 Build duration: 0h 3m 47.506s



The results indicate that the fuel economy is approximately 30 mpg at the end of the drive cycle. The scope also provides the target velocity, engine speed, and brake specific fuel consumption (BSFC).

Run Highway Drive Cycle Simulation

Configure the Drive Cycle Source block to run the highway drive cycle (HWFET). Make sure that you have installed the highway drive cycle.

```
set_param(dcs, 'cycleVar', 'HWFET');
```

Run a simulation of the highway drive cycle. View the results in the Performance and FE Scope.

```
tfinal = get_param(dcs, 'tfinal');
tf = tfinal(1:strfind(tfinal, ' '));
simout2 = sim(model, 'ReturnWorkspaceOutputs', 'on', 'StopTime', tf);
open_system('SiCiPtReferenceApplication/Visualization/Performance and FE Scope')
```

```
### Starting serial model reference simulation build
### Model reference simulation target for DrivetrainConVeh is up to date.
```

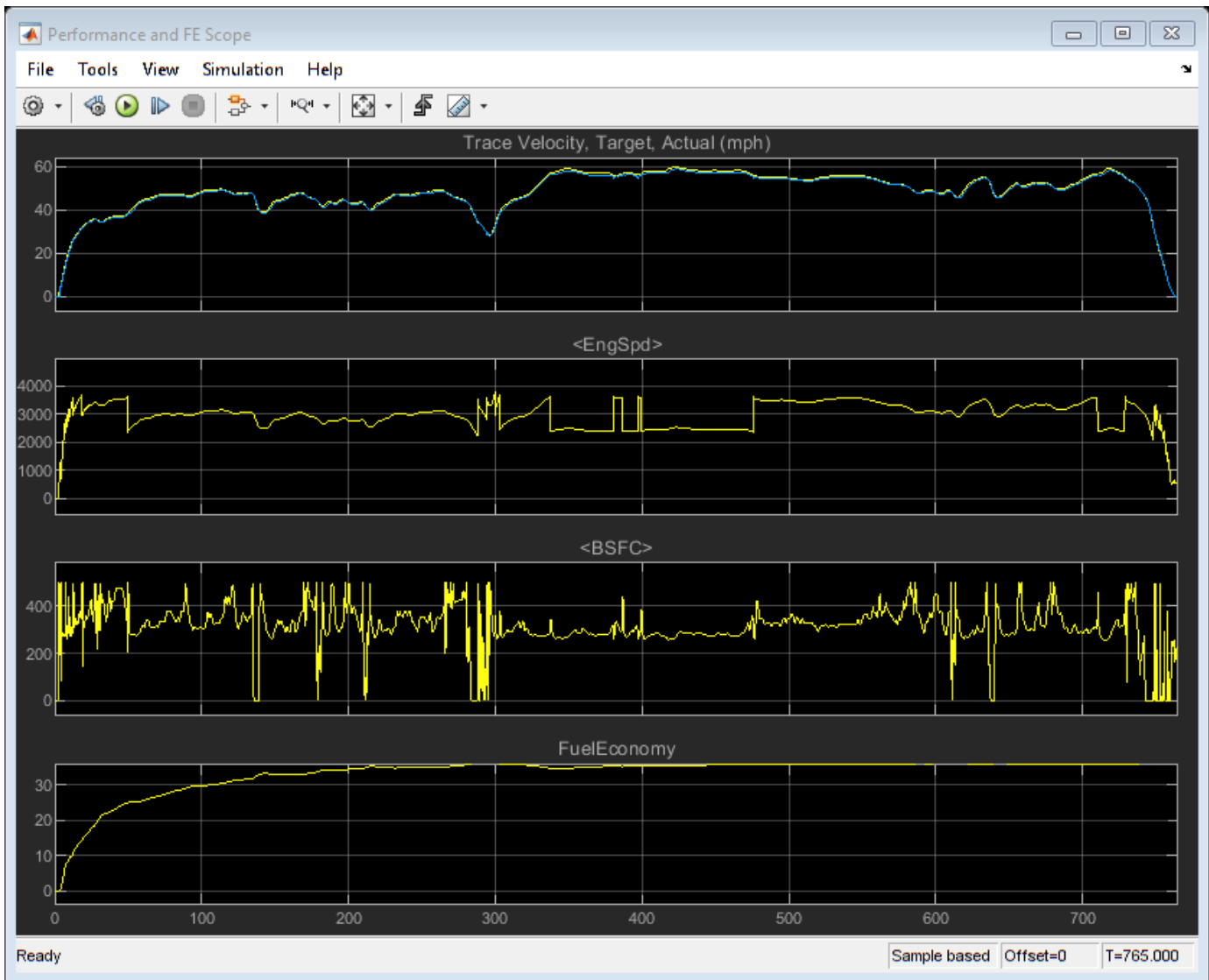
```

### Model reference simulation target for PowertrainMaxPowerController is up to date.
### Model reference simulation target for SiEngineController is up to date.
### Model reference simulation target for SiMappedEngine is up to date.
    
```

Build Summary

```

0 of 4 models built (4 models already up to date)
Build duration: 0h 0m 1.383s
    
```



The results indicate that the fuel economy is approximately 34 mpg at the end of the drive cycle. The scope also provides the target velocity, engine speed, and brake specific fuel consumption (BSFC).

Extract Results

Extract the city and highway fuel economy results for the city and highway drive cycles from the logged data.

```
logout1 = simout1.get('logout');
FE_urban = logout1.get('Fuel Economy [mpg]').Values.Data(end);
logout2 = simout2.get('logout');
FE_hwy = logout2.get('Fuel Economy [mpg]').Values.Data(end);
```

Use the city and highway fuel economy results to compute the combined sticker mpg.

```
FE_combined = 0.55*FE_urban + 0.45*FE_hwy;
```

Extract the tailpipe emissions from the city drive cycle.

```
HC = logout1.get('HC [g/mi]').Values.Data(end);
CO = logout1.get('CO [g/mi]').Values.Data(end);
NOx = logout1.get('NOx [g/mi]').Values.Data(end);
CO2 = logout1.get('CO2 [g/km]').Values.Data(end);
```

Display the fuel economy and city drive cycle 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');
```

```
*****
FUEL ECONOMY
  City:      30.51 mpg
  Highway:   36.63 mpg
  Combined:  33.26 mpg

TAILPIPE EMISSIONS
  HC:    0.001 [g/mi]
  CO:    0.000 [g/mi]
  NOx:   0.002 [g/mi]
  CO2:   178.0 [g/km]
  NMOG:  0.003 [g/mi]
*****
```

See Also

Drive Cycle Source

Related Examples

- “Install Drive Cycle Data” on page 5-2

More About

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

Conventional Vehicle Powertrain Efficiency

The Powertrain Blockset vehicle reference applications include live scripts that you can run to evaluate and report energy and power losses at the component- and subsystem-level. This example shows how to examine the impact of the conventional vehicle transmission efficiency on the powertrain efficiency.

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 mapped 1.5-L spark-ignition (SI) engine and a dual clutch transmission.

autoblkConVehStart

Project files open in a writable location.

- 2 Double-click **Analyze Power and Energy** to open the live script. To generate the energy report, select **Run**.

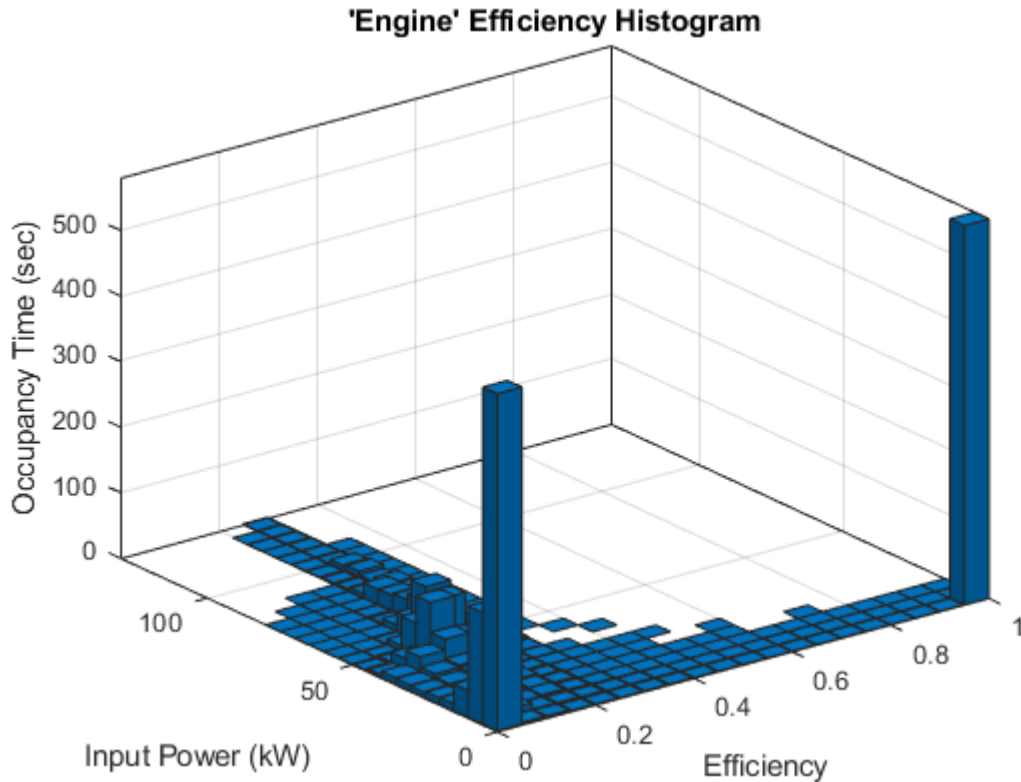
The live script provides:

- An overall energy summary and exported Excel® spreadsheet containing the data. For example, this is similar to the Overall Summary report for the conventional vehicle. The results indicate that the:
 - Overall powertrain input energy is 47.5 MJ
 - Dual clutch transmission average efficiency is 0.933

VehPwrAnalysis.dispSysSummary

System Name	Efficiency	Energy Loss (MJ)	Energy Input (MJ)	Energy Output (MJ)
SiCipReferenceApplication	0.0928	-47	47.5	0
Passenger Car	0.0928	-47	47.5	0
Drivetrain	0.407	-8.51	10.1	-1.04
DCT	0.932	-0.825	10.9	-10.1
Driveshaft Compliance	0.999	0	10.1	-10.1
Dual Clutch Transmission	0.933	-0.815	10.9	-10.1
Differential and Compliance	0.997	0	10.1	-10.1
Front Axle Compliance 1	1	0	5.04	-5.04
Front Axle Compliance 2	1	0	5.04	-5.04
Open Differential	0.997	0	10.1	-10.1
Vehicle	0.797	-2.26	7.15	-4.36
Vehicle Body 3 DOF Longitudinal	0.808	-2.26	7.15	-4.36
Wheels and Brakes	0.608	-5.39	13.6	-8.22
Longitudinal Wheel - Front 1	0.72	-1.62	5.72	-4.1
Longitudinal Wheel - Front 2	0.72	-1.62	5.72	-4.1
Longitudinal Wheel - Rear 1	0.0326	-1.08	1.08	0
Longitudinal Wheel - Rear 2	0.0326	-1.08	1.08	0
Engine	0.203	-38.5	48.3	-9.81
Accessory Load Model	0.937	-0.744	11.7	-11
Mapped SI Engine	0.207	-37.8	47.6	-9.84

- Engine and drivetrain efficiencies, including an engine histogram of time spent at the different engine efficiencies. For example, this is similar to the engine efficiency histogram for the conventional vehicle.



- Drivetrain plant summary that provides the average efficiency, energy input, output, loss, and stored. For example, this is similar to the Drivetrain Plant Summary for the conventional vehicle. The results indicate that the drivetrain input energy is 10.1 MJ.

[SiCiPtReferenceApplication/Passenger Car/Drivetrain](#)

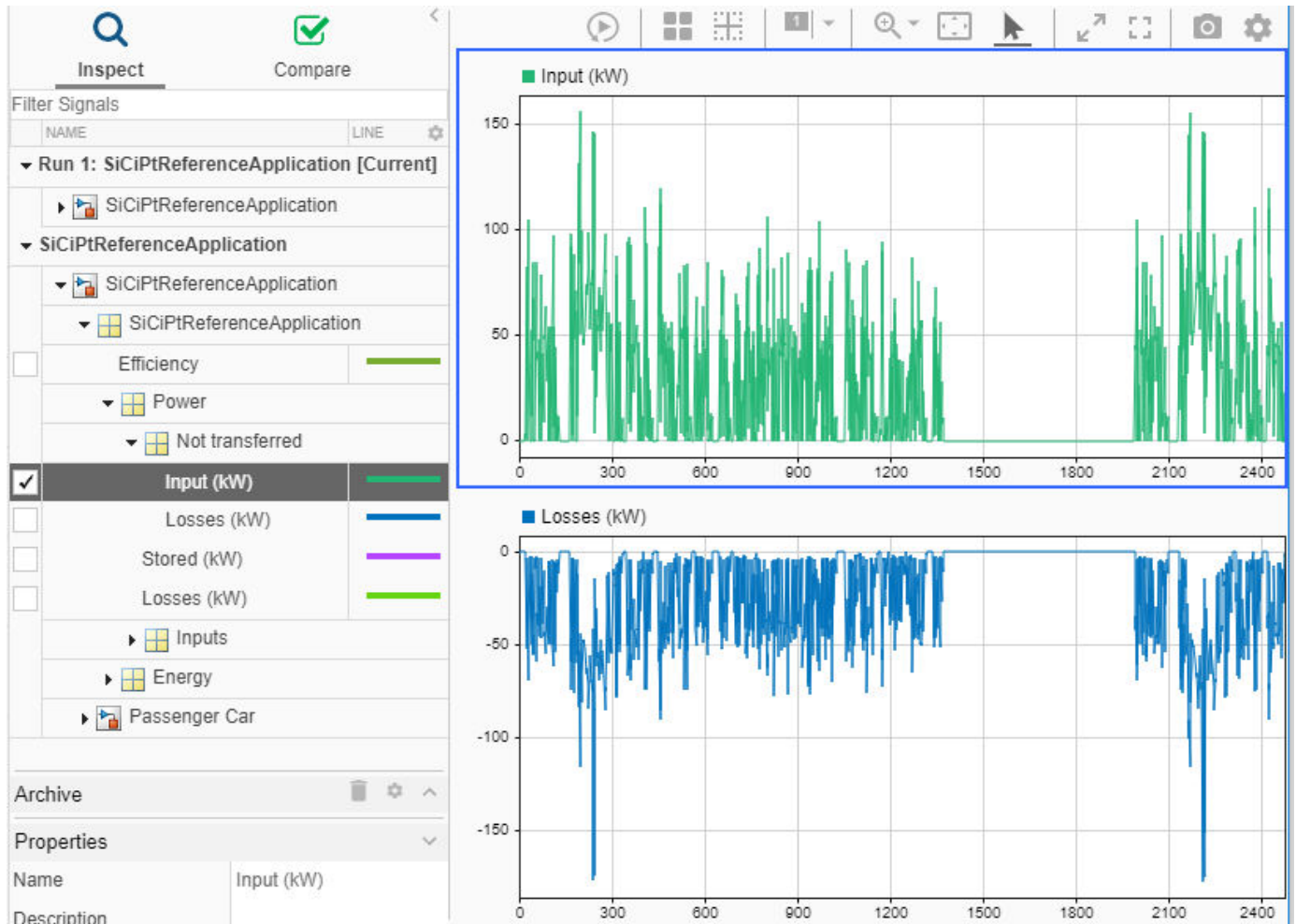
Average Efficiency = 0.41

Signal	Energy (MJ)

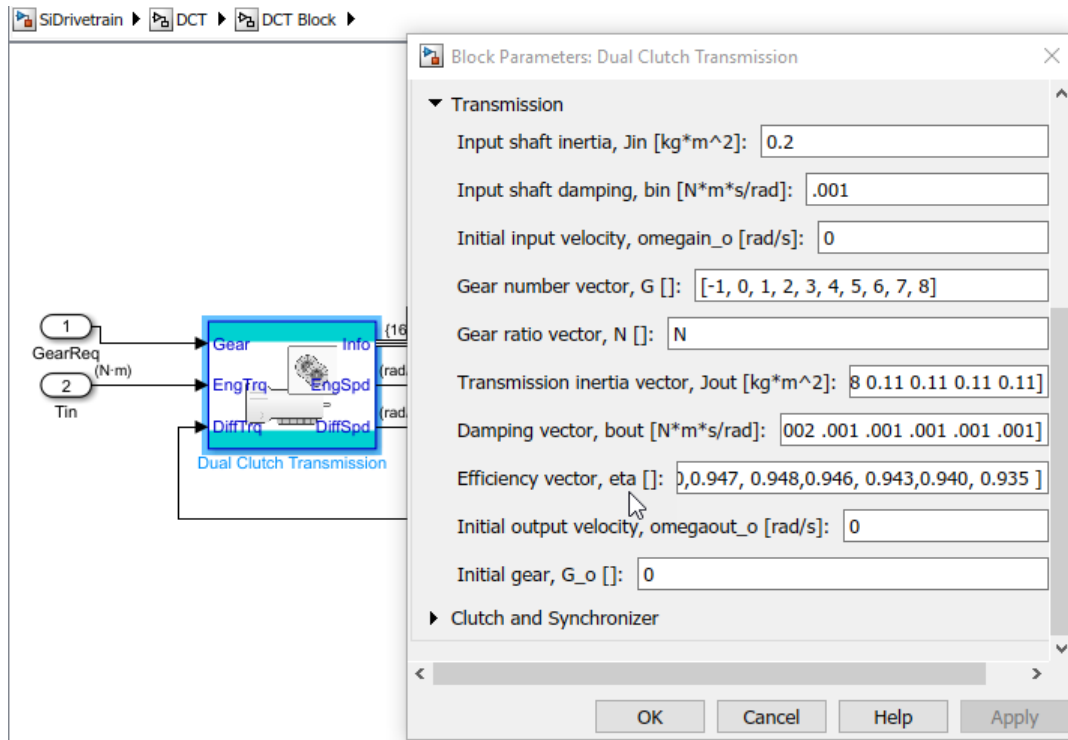
Inputs	10.1
Transferred	9.82
DCT: Engine	9.82
Not transferred	0.252
Outputs	-1.04
DCT: Engine	-1.04
Losses	-8.51
Stored	0.524



- Data logging so that you can use the Simulation Data Inspector to analyze the powertrain efficiency and energy transfer signals. For example, these are similar to the power input and loss plots for the conventional vehicle.



- 3 In the Overall Summary section of the report:
 - a Select Dual Clutch Transmission to open the DCT Block subsystem.
 - b Select the Dual Clutch Transmission block.
 - c In the block mask, open the **Transmission** parameters.



- 4 Change the dual clutch transmission so that it is less efficient. By default, the Dual Clutch Transmission block **Efficiency vector, η** parameter value is [0.930, 0.930, 0.930, 0.940, 0.947, 0.948, 0.946, 0.943, 0.940, 0.935].
 - a Set the **Efficiency vector, η** parameter to $.9*[0.930, 0.930, 0.930, 0.940, 0.947, 0.948, 0.946, 0.943, 0.940, 0.935]$.
 - b Save the DrivetrainConVeh model.
- 5 In the SiCIPtReferenceApplication model window, click **Analyze Power and Energy** to open the live script. To generate the energy report, select **Run**.
- 6 After you run the live script, in the Overall Summary, examine the efficiencies. For example, these results indicate that the:
 - Overall powertrain input energy is 50.6 MJ
 - Dual clutch transmission efficiency is 0.85

When the dual clutch transmission is less efficient, the powertrain requires more energy to complete the drive cycle.

VehPwrAnalysis.dispSysSummary

System Name	Efficiency	Energy Loss (MJ)	Energy Input (MJ)	Energy Output (MJ)
SiCiPtReferenceApplication	0.0867	-50.1	50.6	0
Passenger Car	0.0867	-50.1	50.6	0
Drivetrain	0.374	-9.61	11.1	-0.993
DCT	0.849	-2.01	12	-10
Driveshaft Compliance	0.999	0	10.2	-10.1
Dual Clutch Transmission	0.85	-2	12	-10
Differential and Compliance	0.997	0	10.1	-10.1
Front Axle Compliance 1	1	0	5.06	-5.06
Front Axle Compliance 2	1	0	5.06	-5.06
Open Differential	0.997	0	10.1	-10.1
Vehicle	0.796	-2.25	7.13	-4.34
Vehicle Body 3 DOF Longitudinal	0.808	-2.25	7.13	-4.34
Wheels and Brakes	0.612	-5.32	13.6	-8.25
Longitudinal Wheel - Front 1	0.723	-1.6	5.72	-4.12
Longitudinal Wheel - Front 2	0.723	-1.6	5.72	-4.12
Longitudinal Wheel - Rear 1	0.0328	-1.06	1.07	0
Longitudinal Wheel - Rear 2	0.0328	-1.06	1.07	0
Engine	0.212	-40.5	51.3	-10.9
Accessory Load Model	0.943	-0.721	12.7	-12
Mapped SI Engine	0.215	-39.7	50.6	-10.9

See Also

autoblks.pwr.PlantInfo

Related Examples

- “Getting Started with Powertrain Blockset” on page 1-4
- “Conventional Vehicle Spark-Ignition Engine Fuel Economy and Emissions” on page 1-10

More About

- “Analyze Power and Energy” on page 3-107
- “Explore the Conventional Vehicle Reference Application” on page 3-4
- Simulation Data Inspector

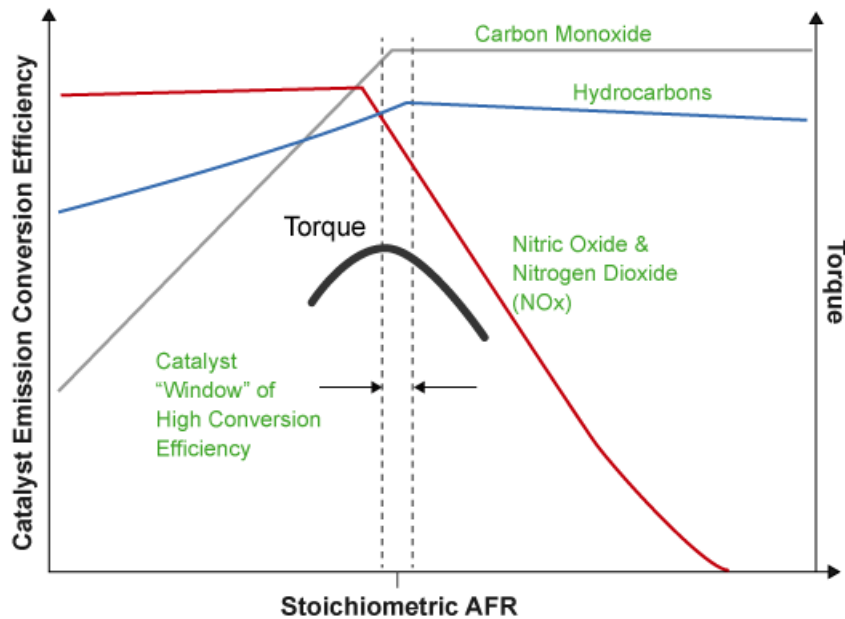
Workflows

- “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 2-11
- “SI Engine Torque Structure Model” on page 2-14
- “SI Engine Simple Torque Model” on page 2-20
- “CI Core Engine Air Mass Flow and Torque Production” on page 2-21
- “CI Engine Speed-Density Air Mass Flow Model” on page 2-22
- “CI Engine Torque Structure Model” on page 2-25
- “CI Engine Simple Torque Model” on page 2-30
- “Engine Calibration Maps” on page 2-31

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 2-11	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.

Air Mass Flow Model	Description
<p>“SI Engine Dual-Independent Cam Phaser Air Mass Flow Model” on page 2-5</p>	<p>To calculate the engine air mass flow, the dual-independent cam phaser model uses:</p> <ul style="list-style-type: none"> • Empirical calibration parameters developed from engine mapping measurements • Desktop calibration parameters derived from engine computer-aided design (CAD) data <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"> • Elimination of MAF sensors in dual cam-phased valvetrain applications • Reasonable accuracy with changes in altitude • Semiphysical modeling approach • Bounded behavior • Suitable execution time for electronic control unit (ECU) implementation • Systematic development of a relatively small number of calibration parameters

Torque Models

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

Brake Torque Model	Description
<p>“SI Engine Torque Structure Model” on page 2-14</p>	<p>For the structured brake torque calculation, the SI engine uses tables for the inner torque, friction torque, optimal spark, spark efficiency, and lambda efficiency.</p> <p>If you select Crank angle pressure and torque on the block Torque tab, you can:</p> <ul style="list-style-type: none"> • Simulate advanced closed-loop engine controls in desktop simulations and on HIL bench, based on cylinder pressure recorded from a model or laboratory test as a function of crank angle. • Simulate driveline vibrations downstream of the engine due to high-frequency crankshaft torsionals. • Simulate engine misfires due to lean operation or spark plug fouling by using the injector pulse width input. • Simulate cylinder deactivation effect (closed intake and exhaust valves, no injected fuel) on individual cylinder pressures, mean-value airflow, mean-value torque, and crank-angle-based torque. • Simulate the fuel-cut effect on individual cylinder pressure, mean-value torque, and crank-angle-based torque.
<p>“SI Engine Simple Torque Model” on page 2-20</p>	<p>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.</p>

See Also

SI Controller | SI Core Engine

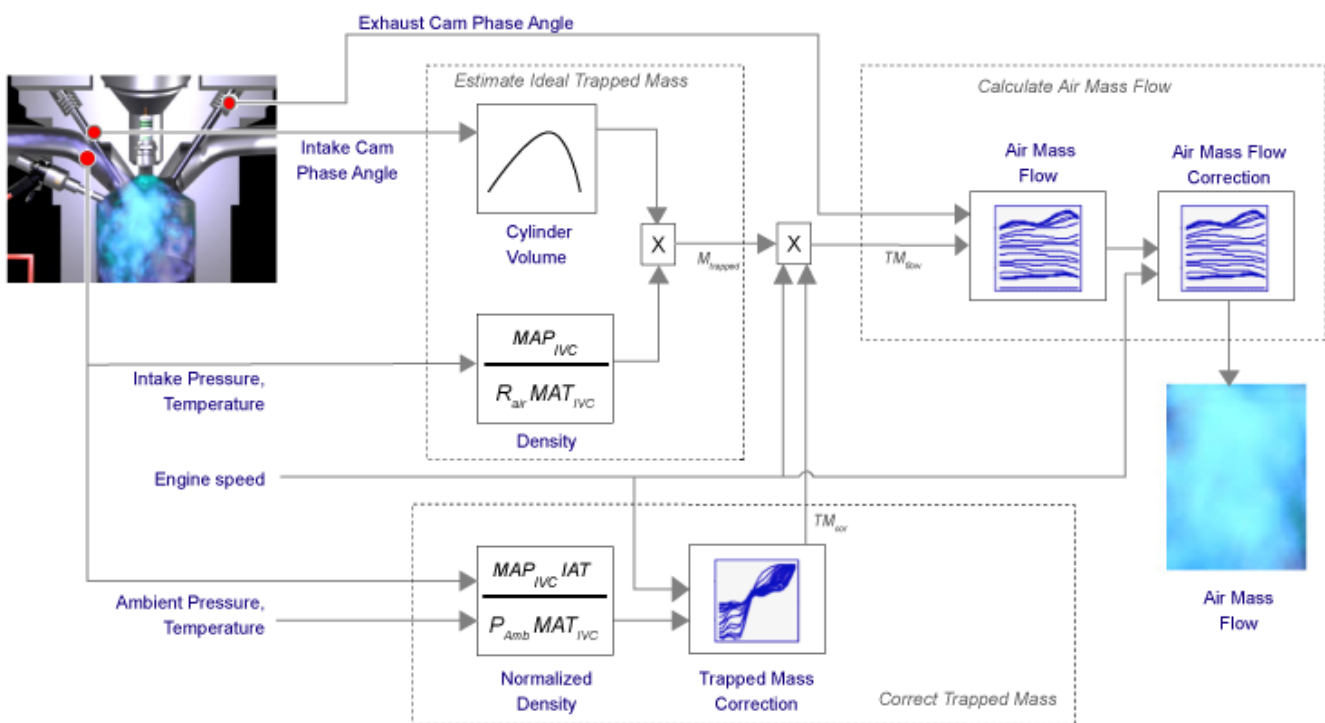
More About

- “Engine Calibration Maps” on page 2-31

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
ρ_{norm}	Normalized density
φ_{ICP}	Measured intake cam phaser angle
φ_{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.

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

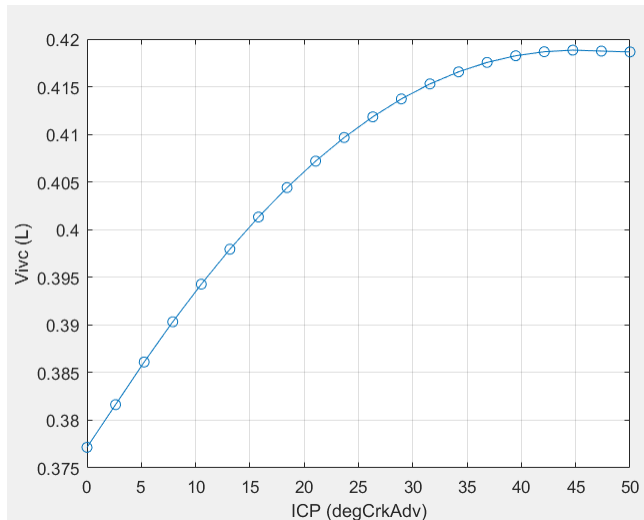
For engines with variable intake cam phasing, the trapped volume at IVC varies.

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.
- φ_{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.

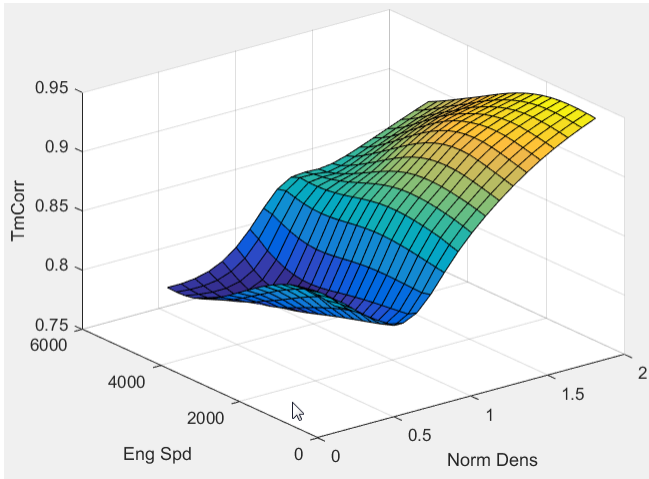
$$\rho_{norm} = \frac{MAP_{IVC}IAT}{P_{Amb}MAT_{IVC}}$$

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.
- ρ_{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.

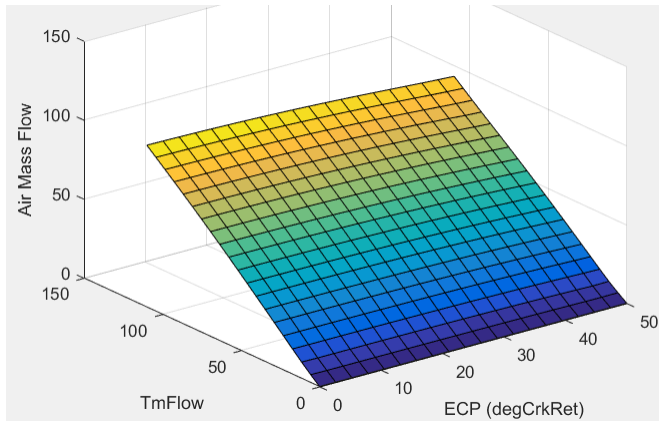
The phaser intake mass flow model lookup table is a function of exhaust cam phaser angles and trapped air mass flow

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

where:

- $\dot{m}_{intkideal}$ is engine intake port mass flow at arbitrary cam phaser angles, in g/s.

- φ_{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.



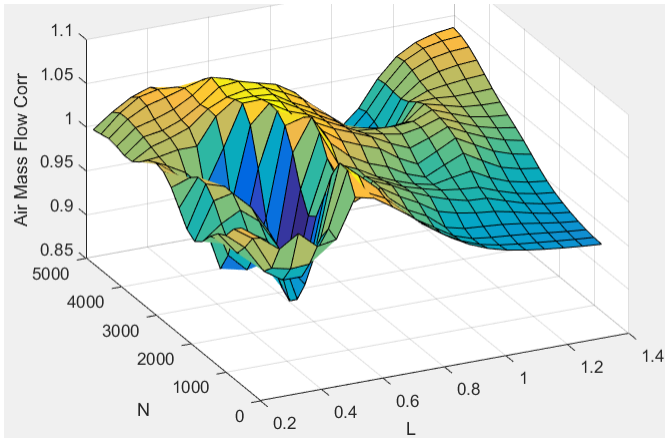
- The exhaust cam phasing has a significant effect on the fraction of burned gas. During the exhaust stroke, exhaust cam-phasing affects the exhaust valve position at exhaust valve closing (EVC) relative to the piston position. A retarded (late) exhaust cam phase angle moves EVC past piston top dead 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_x) 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:

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

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

- The final engine load is expressed by

$$L = \frac{\left(\frac{60s}{min}\right)Cps\dot{m}_{air}}{\left(\frac{1000g}{kg}\right)N_{cyl}NM_{Nom}}$$

See Also

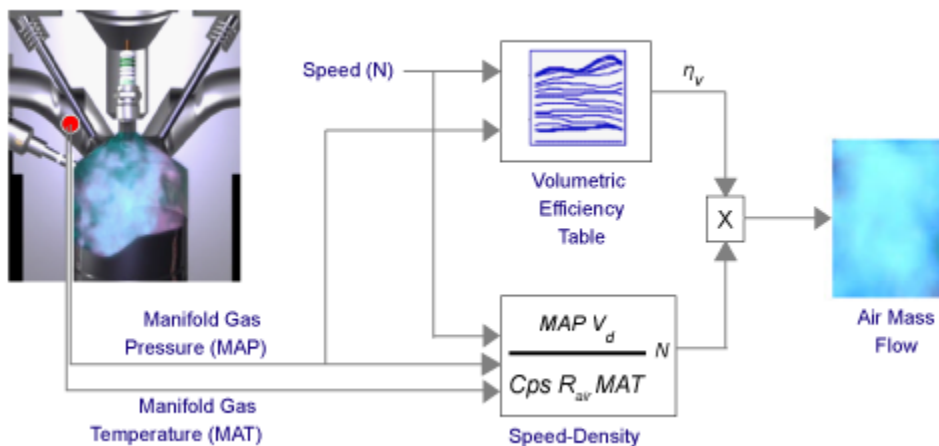
SI Controller | SI Core Engine

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 2-11
- “Engine Calibration Maps” on page 2-31

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.

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

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

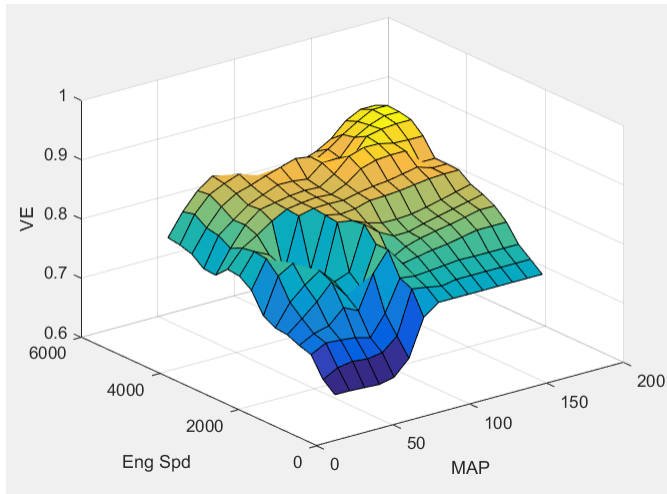
The speed-density air mass flow model uses a volumetric efficiency lookup table to correct the ideal air mass flow.

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

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

where:

- η_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.

$$\eta_v = \frac{CpsR_{air}MAT}{MAPV_dN\left[\frac{1 \text{ min}}{60s}\right]}\dot{m}_{air}$$

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
Cps	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_{η_v}	Engine volumetric efficiency lookup table
η_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

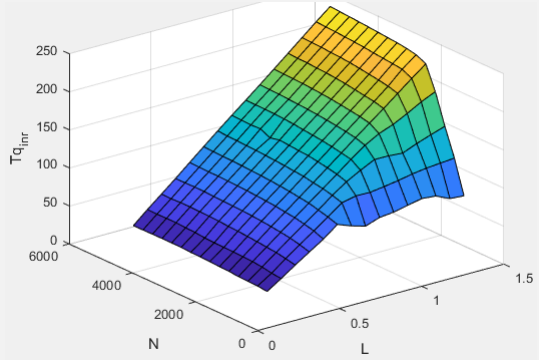
- “SI Engine Dual-Independent Cam Phaser Air Mass Flow Model” on page 2-5
- “Engine Calibration Maps” on page 2-31

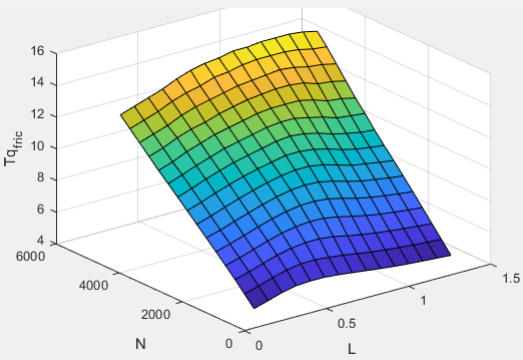
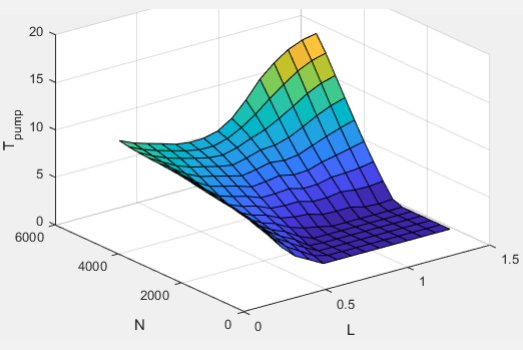
SI Engine Torque Structure Model

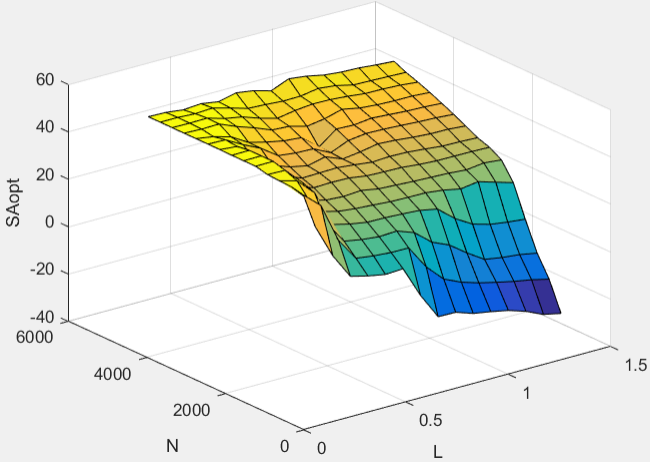
The spark-ignition (SI) engine implements a simplified version of the SI engine torque structure calculation used in a Bosch Engine Management System (EMS). For the torque structure estimation calculation, the block requires calibration tables for:

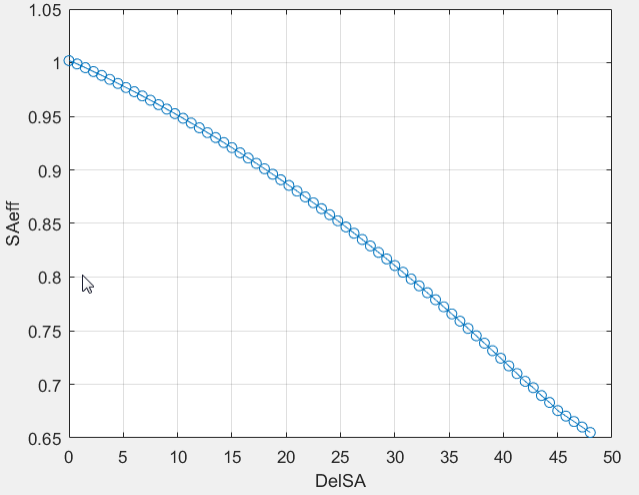
- Inner torque — Maximum torque potential of the engine at a given speed and load
- Friction torque — Torque losses due to friction
- Optimal spark — Spark advance for optimal inner torque
- Spark efficiency — Torque loss due to spark retard from optimal
- Lambda efficiency — Torque loss due to lambda 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
Inner torque, $f_{Tq_{inr}}$	$Tq_{inr} = f_{Tq_{inr}}(L, N)$	<p>The inner torque lookup table, $f_{Tq_{inr}}$, is a function of engine speed and engine load, $Tq_{inr} = f_{Tq_{inr}}(L, N)$, where:</p> <ul style="list-style-type: none"> • Tq_{inr} is inner torque based on gross indicated mean effective pressure, in N·m. • L is engine load at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles, dimensionless. • N is engine speed, in rpm. 

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

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

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, f_{Msa}, 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"> • M_{sa} is the spark retard efficiency multiplier, dimensionless. • ΔSA is the spark retard timing distance from optimal spark advance, in deg. 

Lookup Table	Used to Determine	Plot																						
Lambda efficiency, $f_{M\lambda}$	$M_\lambda = f_{M\lambda}(\lambda)$	<p>The lambda efficiency lookup table, $f_{M\lambda}$, is a function of lambda, $M_\lambda = f_{M\lambda}(\lambda)$, where:</p> <ul style="list-style-type: none"> M_λ is the lambda multiplier on inner torque to account for the air-fuel ratio (AFR) effect, dimensionless. λ is lambda, AFR normalized to stoichiometric fuel AFR, dimensionless. <table border="1"> <caption>Data points for Lambda efficiency vs Lambda</caption> <thead> <tr> <th>Lambda</th> <th>Lambda efficiency</th> </tr> </thead> <tbody> <tr><td>0.65</td><td>0.94</td></tr> <tr><td>0.7</td><td>0.955</td></tr> <tr><td>0.75</td><td>0.97</td></tr> <tr><td>0.8</td><td>0.985</td></tr> <tr><td>0.85</td><td>0.99</td></tr> <tr><td>0.9</td><td>1.00</td></tr> <tr><td>0.95</td><td>0.975</td></tr> <tr><td>1.0</td><td>0.955</td></tr> <tr><td>1.05</td><td>0.925</td></tr> <tr><td>1.1</td><td>0.89</td></tr> </tbody> </table>	Lambda	Lambda efficiency	0.65	0.94	0.7	0.955	0.75	0.97	0.8	0.985	0.85	0.99	0.9	1.00	0.95	0.975	1.0	0.955	1.05	0.925	1.1	0.89
Lambda	Lambda efficiency																							
0.65	0.94																							
0.7	0.955																							
0.75	0.97																							
0.8	0.985																							
0.85	0.99																							
0.9	1.00																							
0.95	0.975																							
1.0	0.955																							
1.05	0.925																							
1.1	0.89																							

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.

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

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

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)
ΔSA	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_λ	Lambda multiplier on inner torque to account for the AFR effect
λ	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_{qinr}	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

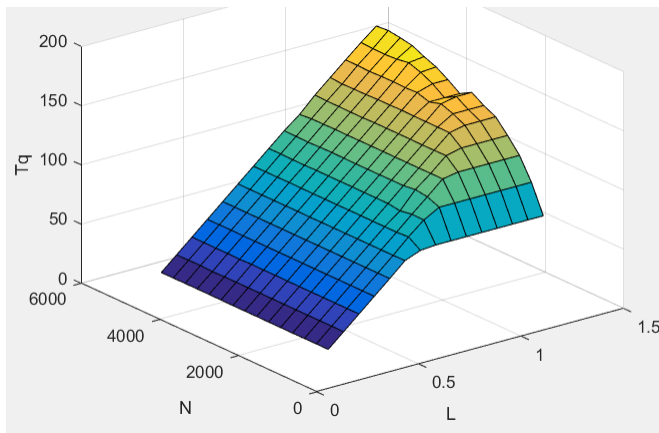
More About

- “SI Core Engine Air Mass Flow and Torque Production” on page 2-2
- “SI Engine Simple Torque Model” on page 2-20

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, $T_{brake} = f_{TnL}(L, N)$, 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 2-14

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 (NOx) 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 2-22. 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 2-25	<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"> • Start of injection (SOI) timing • Exhaust back-pressure • Burned fuel mass • Intake manifold gas pressure, temperature, and oxygen percentage • Fuel rail pressure <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 2-30	<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

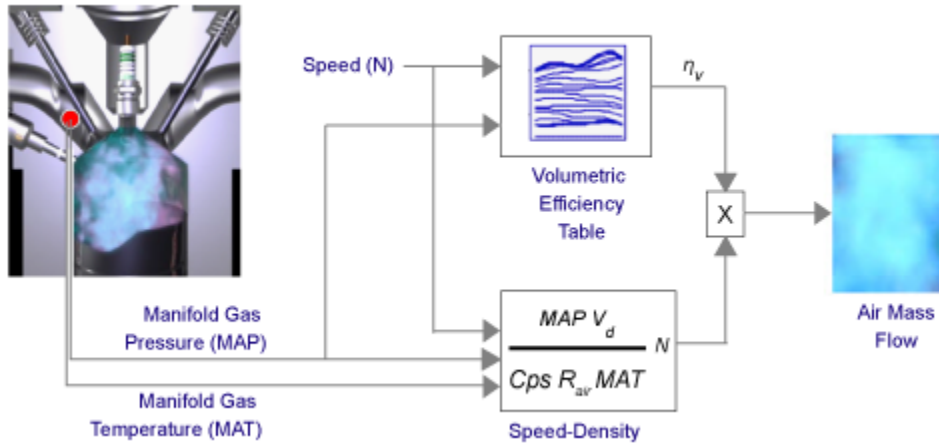
CI Controller | CI Core Engine

More About

- “Engine Calibration Maps” on page 2-31

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.

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

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

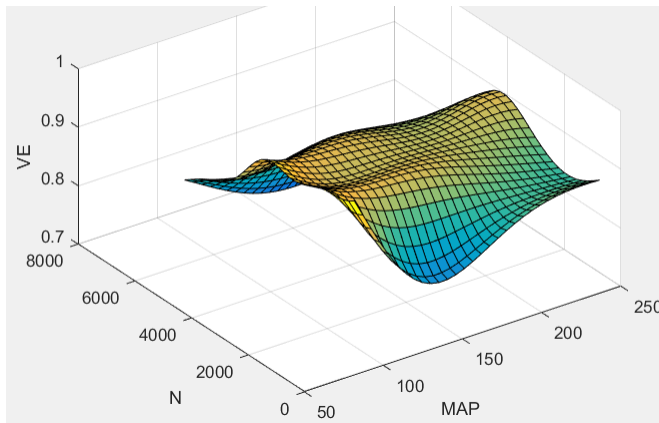
The speed-density air mass flow model uses a volumetric efficiency lookup table to determine the volumetric efficiency.

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:

- η_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.

$$\eta_v = \frac{CpsR_{air}MAT}{MAPV_dN\left[\frac{1}{60s}\right]}\dot{m}_{air}$$

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.

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

$$L = \frac{\left(\frac{60s}{min}\right)Cps\dot{m}_{air}}{\left(\frac{1000g}{kg}\right)N_{cyl}NM_{Nom}}$$

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
Cps	Crankshaft revolutions per power stroke
R_{air}	Ideal gas constant for air and burned gas mixture
MAT	Cycle average intake manifold gas absolute temperature
η_v	Engine volumetric efficiency
f_{η_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 2-21
- “Engine Calibration Maps” on page 2-31

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, F, and engine speed, N:</p> <ul style="list-style-type: none"> • Main start of injection timing, $SOI = f_{SOIc}(F,N)$ • Intake manifold gas temperature, $MAT = f_{MAT}(F,N)$ • Intake manifold gas pressure, $MAP = f_{MAP}(F,N)$ • Intake manifold oxygen percentage, $O2PCT = f_{O2}(F,N)$ • Fuel rail pressure, $FUELP = f_{fuelp}(F,N)$
Step 2: Calculate relative engine states	<p>To determine these relative engine states, the model calculates deviations from their nominal values.</p> <ul style="list-style-type: none"> • Main start of injection timing delta, $\Delta SOI_c = f_{SOI}(F,N) - SOI$ • Intake manifold gas temperature delta, $\Delta MAT = f_{MAT}(F,N) - MAT$ • Intake manifold oxygen percentage delta, $\Delta O2PCT = f_{O2}(F,N) - O2PCT$ • Fuel rail pressure delta, $\Delta FUELP = f_{fuelp}(F,N) - FUELP$ <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> $MAP_{ratio} = \frac{MAP}{f_{MAP}(F,N)}$

Step	Description
Step 3: Determine efficiency multipliers	<p>Model uses gross indicated mean effective pressure (IMEPG)^[1] 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"> • Main start of injection timing efficiency multiplier, $SOI_{eff} = f_{SOI_{eff}}(\Delta SOI, N)$ • Intake manifold gas temperature efficiency multiplier, $MAT_{eff} = f_{MAT_{eff}}(\Delta MAT, N)$ • Intake manifold gas pressure efficiency multiplier, $MAP_{eff} = f_{MAP_{eff}}(MAP_{ratio}, \lambda)$ • Intake manifold oxygen percentage efficiency multiplier, $O2P_{eff} = f_{O2P_{eff}}(\Delta O2P, N)$ • Fuel rail pressure efficiency multiplier, $FUELP_{eff} = f_{FUELP_{eff}}(\Delta FUELP, N)$
Step 4: Determine indicated mean effective cylinder pressure (IMEP) available for torque production	<p>To determine the IMEP available for torque production, the model implements these equations.</p> $IMEP = SOI_{eff} MAP_{eff} MAT_{eff} O2p_{eff} FUELP_{eff} IMEPG$ $IMEPG = f_{IMEPG}(F, N)$ <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, F, and engine speed, N.</p>
Step 5: Account for losses due to friction	<p>To account for friction effects, the model uses the nominal friction mean effective pressure (FMEP)^[1] 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, F, and engine speed, N. To account for the temperature effect on friction, the model use a lookup table that is a function of oil temperature, T_{oil}, and N.</p>
Step 6: Account for pressure loss due to pumping	<p>To account for pressure losses due to pumping, the model uses the nominal pumping mean effective pressure (PMEP)^[1] 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, F, and engine speed, N. Under normal operating conditions, PMEP is negative, indicating a loss of cylinder pressure.</p>

Step	Description
Step 7: Account for late fuel injection SOI timing on IMEP	To account for late fuel injection SOI timing on IMEP, $\Delta IMEP_{post}$, the model uses a lookup table that is a function of the effective pressure post inject SOI timing centroid, SOI_{post} , and the post inject mass sum, F_{post} . $\Delta IMEP_{post} = f_{\Delta IMEP_{post}}(SOI_{post}, F_{post})$
Step 8: Calculate engine brake torque	To calculate the engine brake torque, T_{brake} , the model converts the brake mean effective pressure (BMEP) ^[1] to engine brake torque using these equations. The BMEP calculation accounts for all gross mean effective pressure losses. V_d is displaced cylinder volume. Cps is the number of power strokes per revolution. $BMEP = IMEPG + \Delta IMEP_{post} - FMEP + PMEP$ $T_{brake} = \frac{V_d}{2\pi Cps} BMEP$

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 Structure	<p>The nominal exhaust temperature, $T_{exh_{nom}}$, is a product of these exhaust temperature efficiencies:</p> <ul style="list-style-type: none"> • SOI timing • Intake manifold gas pressure • Intake manifold gas temperature • Intake manifold gas oxygen percentage • Fuel rail pressure • Optimal temperature <p>The exhaust temperature, $T_{exh_{nom}}$, is offset by a post temperature effect, ΔT_{post}, that accounts for post and late injections during the expansion and exhaust strokes.</p>	$T_{exh_{nom}} = SOI_{exh_{teff}} MAP_{exh_{teff}} MAT_{exh_{teff}} O2p_{exh_{teff}} FUEL_{exh_{teff}}$ $T_{exh} = T_{exh_{nom}} + \Delta T_{post}$ $SOI_{exh_{teff}} = f_{SOI_{exh_{teff}}}(\Delta SOI, N)$ $MAP_{exh_{teff}} = f_{MAP_{exh_{teff}}}(MAP_{ratio}, \lambda)$ $MAT_{exh_{teff}} = f_{MAT_{exh_{teff}}}(\Delta MAT, N)$ $O2p_{exh_{teff}} = f_{O2p_{exh_{teff}}}(\Delta O2p, N)$ $T_{exh_{opt}} = f_{T_{exh}}(F, N)$

The equations use these variables.

F	Compression stroke injected fuel mass
N	Engine speed
T_{exh}	Exhaust manifold gas temperature
$T_{exh_{opt}}$	Optimal exhaust manifold gas temperature
ΔT_{post}	Post injection temperature effect
$T_{exh_{nom}}$	Nominal exhaust temperature
$SOI_{exh_{teff}}$	Main SOI exhaust temperature efficiency multiplier
ΔSOI	Main SOI timing relative to optimal timing
$MAP_{exh_{teff}}$	Intake manifold gas pressure exhaust temperature efficiency multiplier
MAP_{ratio}	Intake manifold gas pressure ratio relative to optimal pressure ratio
λ	Intake manifold gas lambda
$MAT_{exh_{teff}}$	Intake manifold gas temperature exhaust temperature efficiency multiplier
ΔMAT	Intake manifold gas temperature relative to optimal temperature
$O2P_{exh_{teff}}$	Intake manifold gas oxygen exhaust temperature efficiency multiplier
$\Delta O2P$	Intake gas oxygen percent relative to optimal
$FUEL_{exh_{teff}}$	Fuel rail pressure exhaust temperature efficiency multiplier
$\Delta FUEL$	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

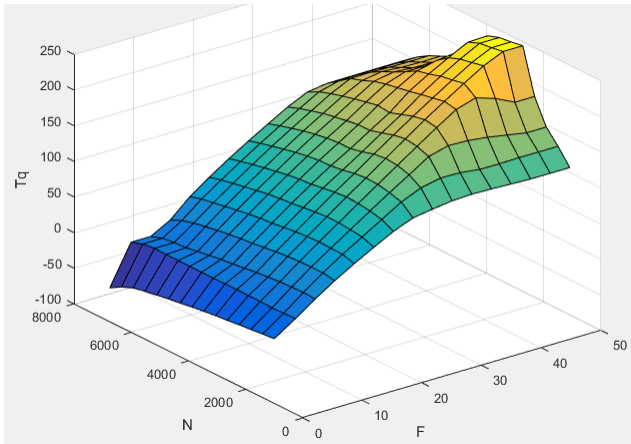
More About

- “CI Core Engine Air Mass Flow and Torque Production” on page 2-21
- “CI Engine Simple Torque Model” on page 2-30

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, $T_{brake} = f_{Tnf}(F, N)$, 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 2-21
- “CI Engine Torque Structure Model” on page 2-25

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

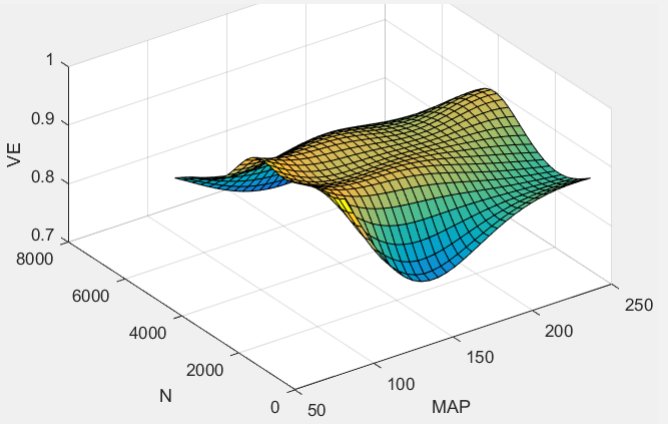
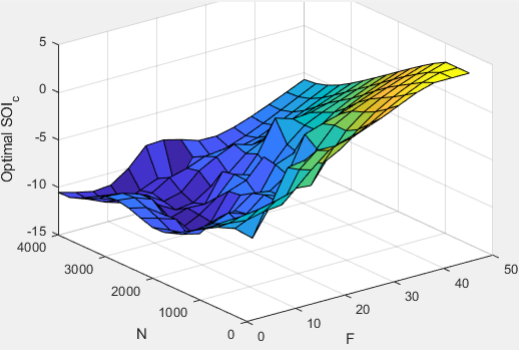
The engine controller model calibration maps in the reference applications represent the optimal open-loop control commands for given engine operating points.

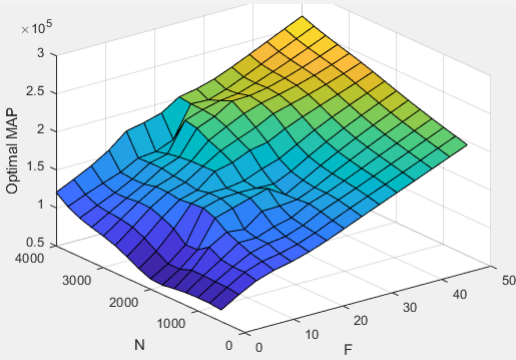
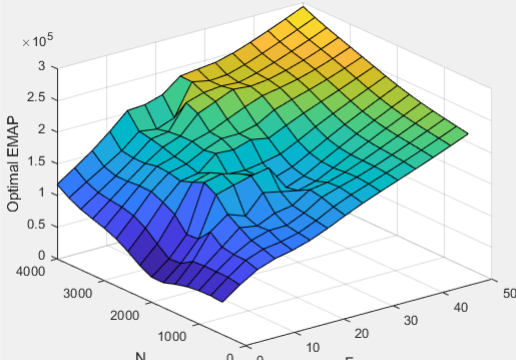
To develop the calibration maps for the SI engine controller, MathWorks used the GT-POWER reference engine models in a virtual engine calibration optimization (VECO) process. The process optimized the open-loop control commands for 1.5-L SI engine, subject to engine operating constraints for knock, turbocharger speed, and exhaust temperature.

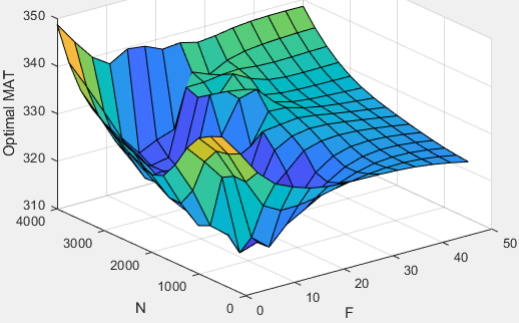
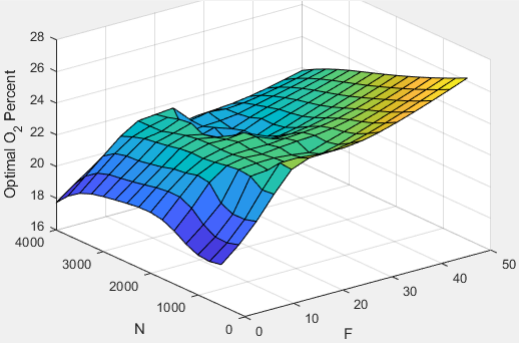
To develop the calibration maps for the CI engine controller, MathWorks used the DOE test data from the GT-POWER 1.5-L CI reference model operated at minimum brake-specific fuel consumption (BSFC).

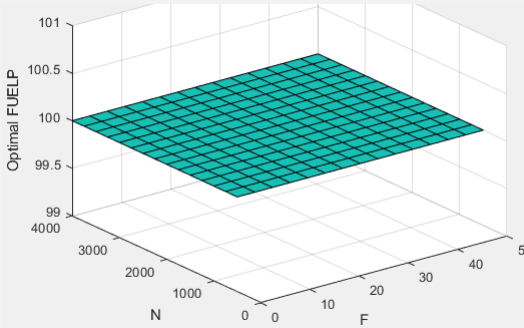
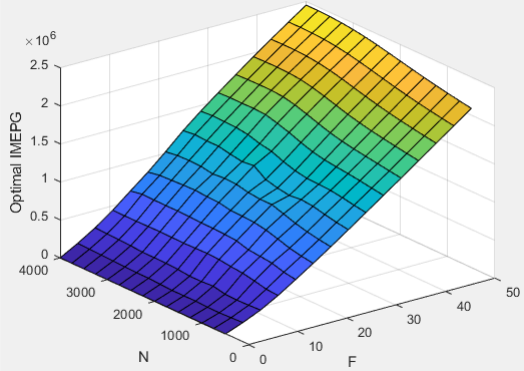
Calibration Maps in Compression-Ignition (CI) Blocks

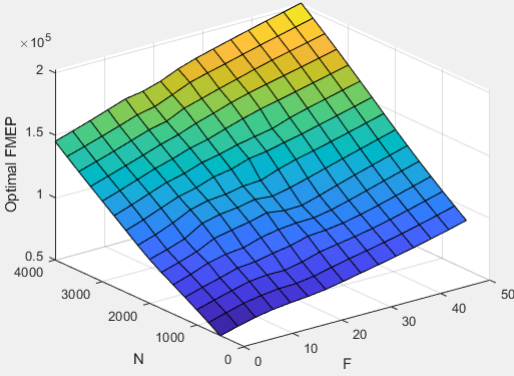
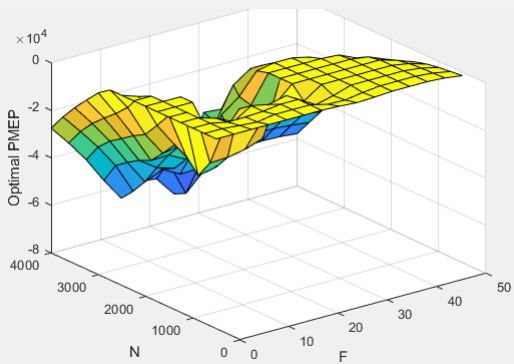
In the engine models, the Powertrain Blockset blocks implement these calibration maps.

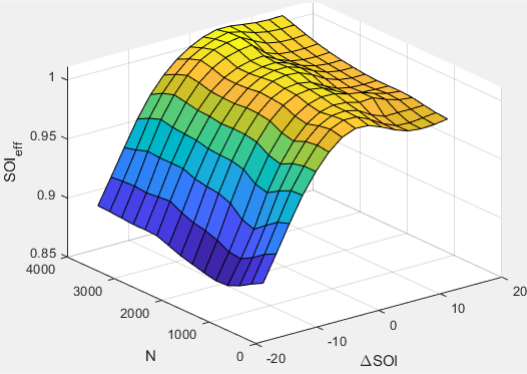
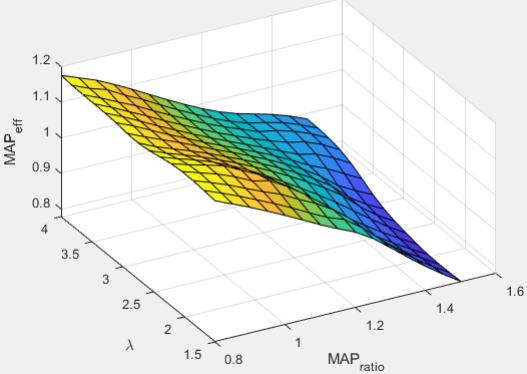
Map	Used For	In	Description
Volumetric efficiency	“CI Engine Speed-Density Air Mass Flow Model” on page 2-22	CI Core Engine CI Controller	The volumetric efficiency lookup table is a function of the intake manifold absolute pressure at intake valve closing (IVC) and engine speed $\eta_v = f_{\eta_v}(MAP, N)$ where: <ul style="list-style-type: none"> • η_v is engine volumetric efficiency, dimensionless. • MAP is intake manifold absolute pressure, in KPa. • N is engine speed, in rpm. 
Optimal main start of injection (SOI) timing	“CI Engine Torque Structure Model” on page 2-25	CI Core Engine CI Controller	The optimal main start of injection (SOI) timing lookup table, f_{SOI_c} , is a function of the engine speed and injected fuel mass, $SOI_c = f_{SOI_c}(FN)$, where: <ul style="list-style-type: none"> • SOI_c is optimal SOI timing, in degATDC. • F is compression stroke injected fuel mass, in mg per injection. • N is engine speed, in rpm. 

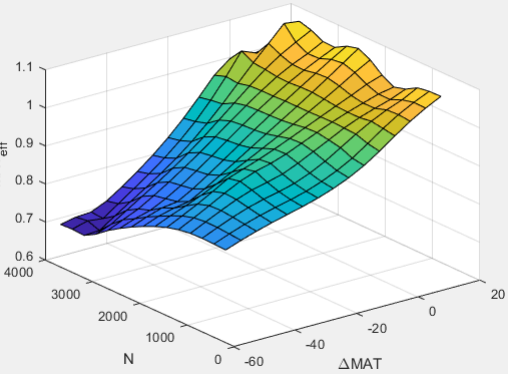
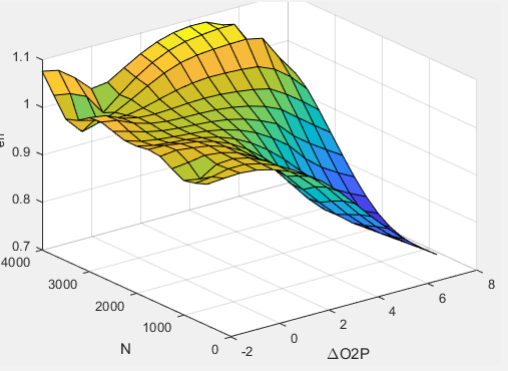
Map	Used For	In	Description
Optimal intake manifold gas pressure	"CI Engine Torque Structure Model" on page 2-25	CI Core Engine CI Controller	<p>The optimal intake manifold gas pressure lookup table, f_{MAP}, is a function of the engine speed and injected fuel mass, $MAP = f_{MAP}(F,N)$, where:</p> <ul style="list-style-type: none"> • MAP is optimal intake manifold gas pressure, in Pa. • F is compression stroke injected fuel mass, in mg per injection. • N is engine speed, in rpm. 
Optimal exhaust manifold gas pressure	"CI Engine Torque Structure Model" on page 2-25	CI Core Engine CI Controller	<p>The optimal exhaust manifold gas pressure lookup table, f_{EMAP}, is a function of the engine speed and injected fuel mass, $EMAP = f_{EMAP}(F,N)$, where:</p> <ul style="list-style-type: none"> • $EMAP$ is optimal exhaust manifold gas pressure, in Pa. • F is compression stroke injected fuel mass, in mg per injection. • N is engine speed, in rpm. 

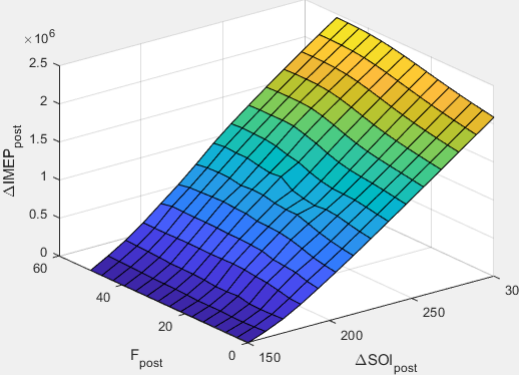
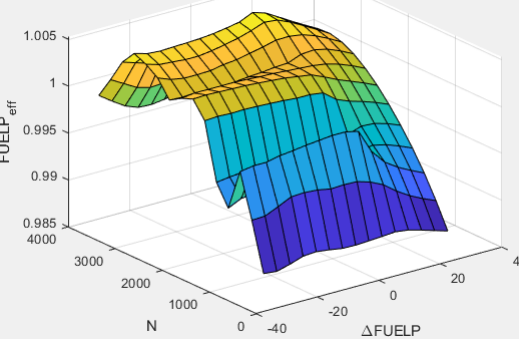
Map	Used For	In	Description
Optimal intake manifold gas temperature	"CI Engine Torque Structure Model" on page 2-25	CI Core Engine CI Controller	<p>The optimal intake manifold gas temperature lookup table, f_{MAT}, is a function of the engine speed and injected fuel mass, $MAT = f_{MAT}(F,N)$, where:</p> <ul style="list-style-type: none"> • MAT is optimal intake manifold gas temperature, in K. • F is compression stroke injected fuel mass, in mg per injection. • N is engine speed, in rpm. 
Optimal intake gas oxygen percent	"CI Engine Torque Structure Model" on page 2-25	CI Core Engine CI Controller	<p>The optimal intake gas oxygen percent lookup table, f_{O_2}, is a function of the engine speed and injected fuel mass, $O2PCT = f_{O_2}(F,N)$, where:</p> <ul style="list-style-type: none"> • $O2PCT$ is optimal intake gas oxygen, in percent. • F is compression stroke injected fuel mass, in mg per injection. • N is engine speed, in rpm. 

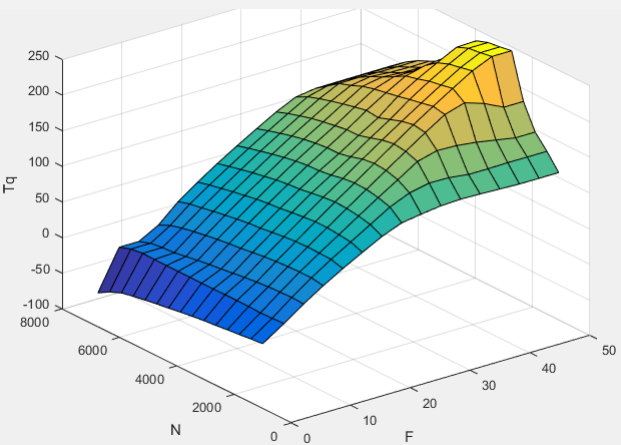
Map	Used For	In	Description
Optimal fuel rail pressure	"CI Engine Torque Structure Model" on page 2-25	CI Core Engine CI Controller	<p>The optimal fuel rail pressure lookup table, f_{fuelp}, is a function of the engine speed and injected fuel mass, $FUELP = f_{fuelp}(F,N)$, where:</p> <ul style="list-style-type: none"> • $FUELP$ is optimal fuel rail pressure, in MPa. • F is compression stroke injected fuel mass, in mg per injection. • N is engine speed, in rpm. 
Optimal gross indicated mean effective pressure	"CI Engine Torque Structure Model" on page 2-25	CI Core Engine CI Controller	<p>The optimal gross indicated mean effective pressure lookup table, f_{imepg}, is a function of the engine speed and injected fuel mass, $IMEPG = f_{imepg}(F,N)$, where:</p> <ul style="list-style-type: none"> • $IMEPG$ is optimal gross indicated mean effective pressure, in Pa. • F is compression stroke injected fuel mass, in mg per injection. • N is engine speed, in rpm. 

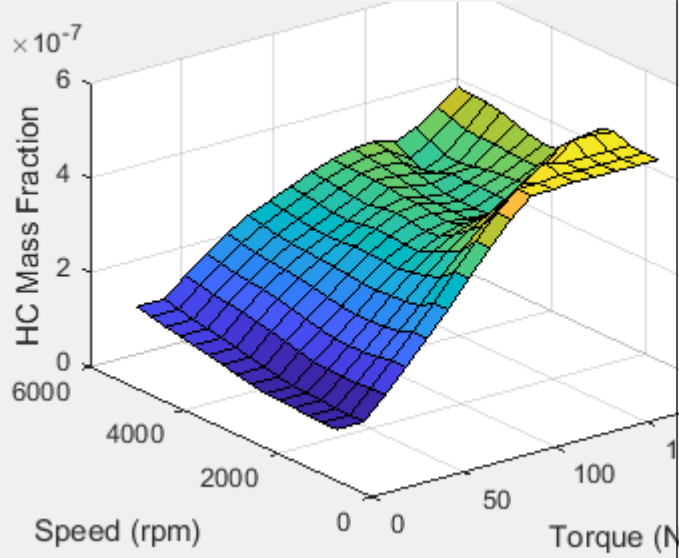
Map	Used For	In	Description
Optimal friction mean effective pressure	"CI Engine Torque Structure Model" on page 2-25	CI Core Engine CI Controller	<p>The optimal friction mean effective pressure lookup table, $f_{f_{mep}}$, is a function of the engine speed and injected fuel mass, $FMEP = f_{f_{mep}}(F, N)$, where:</p> <ul style="list-style-type: none"> • $FMEP$ is optimal friction mean effective pressure, in Pa. • F is compression stroke injected fuel mass, in mg per injection. • N is engine speed, in rpm. 
Optimal pumping mean effective pressure	"CI Engine Torque Structure Model" on page 2-25	CI Core Engine CI Controller	<p>The optimal pumping mean effective pressure lookup table, $f_{p_{mep}}$, is a function of the engine speed and injected fuel mass, $PMEP = f_{p_{mep}}(F, N)$, where:</p> <ul style="list-style-type: none"> • $PMEP$ is optimal pumping mean effective pressure, in Pa. • F is compression stroke injected fuel mass, in mg per injection. • N is engine speed, in rpm. 

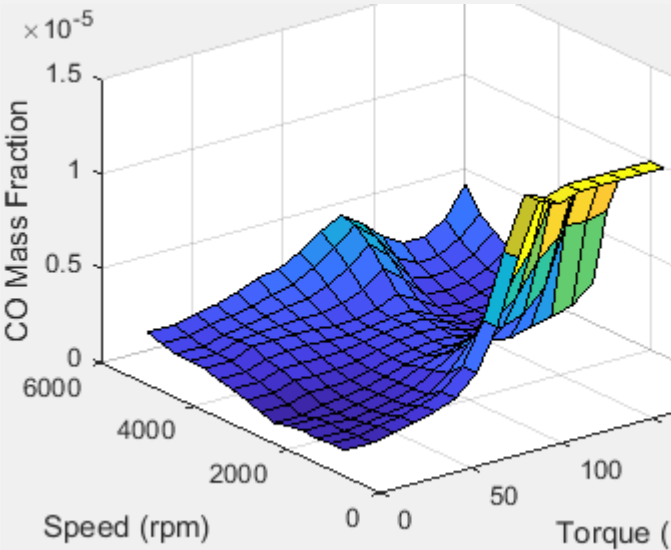
Map	Used For	In	Description
Main SOI timing efficiency multiplier	"CI Engine Torque Structure Model" on page 2-25	CI Core Engine CI Controller	<p>The main start of injection (SOI) timing efficiency multiplier lookup table, $f_{SOI_{eff}}$, is a function of the engine speed and main SOI timing relative to optimal timing, $SOI_{eff} = f_{SOI_{eff}}(\Delta SOI, N)$, where:</p> <ul style="list-style-type: none"> • SOI_{eff} is main SOI timing efficiency multiplier, dimensionless. • ΔSOI is main SOI timing relative to optimal timing, in degBTDC. • N is engine speed, in rpm. 
Intake manifold gas pressure efficiency multiplier	"CI Engine Torque Structure Model" on page 2-25	CI Core Engine CI Controller	<p>The intake manifold gas pressure efficiency multiplier lookup table, $f_{MAP_{eff}}$, is a function of the intake manifold gas pressure ratio relative to optimal pressure ratio and lambda, $MAP_{eff} = f_{MAP_{eff}}(MAP_{ratio}, \lambda)$, where:</p> <ul style="list-style-type: none"> • MAP_{eff} is intake manifold gas pressure efficiency multiplier, dimensionless. • MAP_{ratio} is intake manifold gas pressure ratio relative to optimal pressure ratio, dimensionless. • λ is intake manifold gas lambda, dimensionless. 

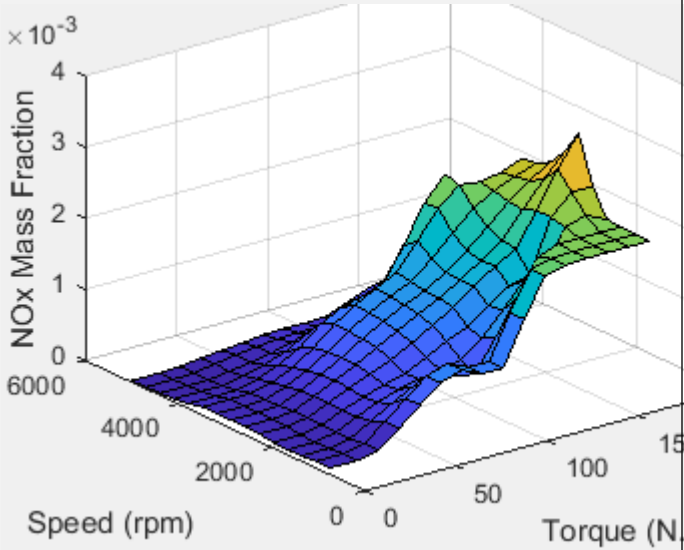
Map	Used For	In	Description
Intake manifold gas temperature efficiency multiplier	"CI Engine Torque Structure Model" on page 2-25	CI Core Engine CI Controller	<p>The intake manifold gas temperature efficiency multiplier lookup table, $f_{MAT_{eff}}$, is a function of the engine speed and intake manifold gas temperature relative to optimal temperature, $MAT_{eff} = f_{MAT_{eff}}(\Delta MAT, N)$, where:</p> <ul style="list-style-type: none"> • MAT_{eff} is intake manifold gas temperature efficiency multiplier, dimensionless. • ΔMAT is intake manifold gas temperature relative to optimal temperature, in K. • N is engine speed, in rpm. 
Intake manifold gas oxygen efficiency multiplier	"CI Engine Torque Structure Model" on page 2-25	CI Core Engine CI Controller	<p>The intake manifold gas oxygen efficiency multiplier lookup table, $f_{O2P_{eff}}$, is a function of the engine speed and intake manifold gas oxygen percent relative to optimal, $O2P_{eff} = f_{O2P_{eff}}(\Delta O2P, N)$, where:</p> <ul style="list-style-type: none"> • $O2P_{eff}$ is intake manifold gas oxygen efficiency multiplier, dimensionless. • $\Delta O2P$ is intake gas oxygen percent relative to optimal, in percent. • N is engine speed, in rpm. 

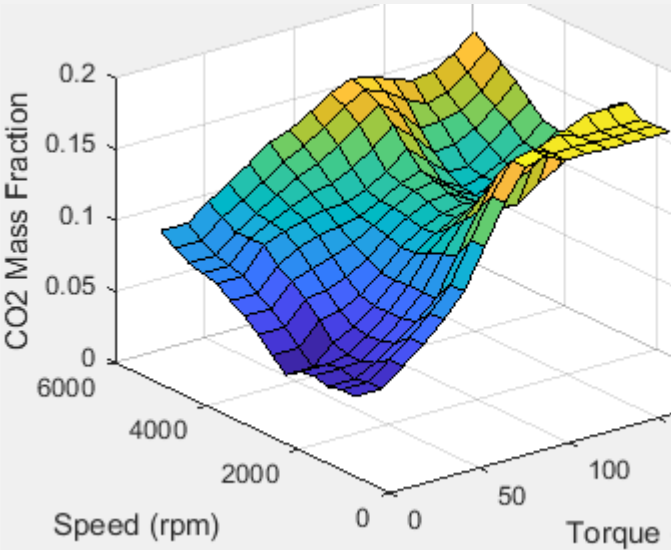
Map	Used For	In	Description
<p>Indicated mean effective pressure post inject correction</p>	<p>“CI Engine Torque Structure Model” on page 2-25</p>	<p>CI Core Engine CI Controller</p>	<p>The indicated mean effective pressure post inject correction lookup table, $f_{IMEP_{post}}$, is a function of the engine speed and fuel rail pressure relative to optimal breakpoints, $\Delta IMEP_{post} = f_{IMEP_{post}}(\Delta SOI_{post}, F_{post})$, where:</p> <ul style="list-style-type: none"> • $\Delta IMEP_{post}$ is indicated mean effective pressure post inject correction, in Pa. • ΔSOI_{post} is indicated mean effective pressure post inject start of inject timing centroid, in degATDC. • F_{post} is indicated mean effective pressure post inject mass sum, in mg per injection. 
<p>Fuel rail pressure efficiency multiplier</p>	<p>“CI Engine Torque Structure Model” on page 2-25</p>	<p>CI Core Engine CI Controller</p>	<p>The fuel rail pressure efficiency multiplier lookup table, $f_{FUELPeff}$, is a function of the engine speed and fuel rail pressure relative to optimal breakpoints, $FUELPeff = f_{FUELPeff}(\Delta FUELPeff, N)$, where:</p> <ul style="list-style-type: none"> • $FUELPeff$ is fuel rail pressure efficiency multiplier, dimensionless. • $\Delta FUELPeff$ is fuel rail pressure relative to optimal, in MPa. • N is engine speed, in rpm. 

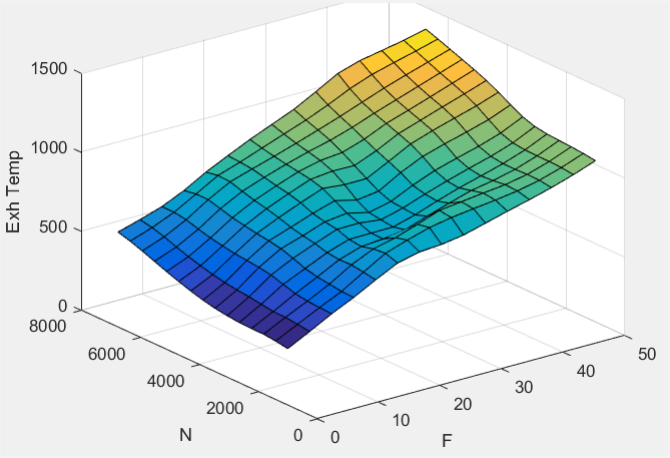
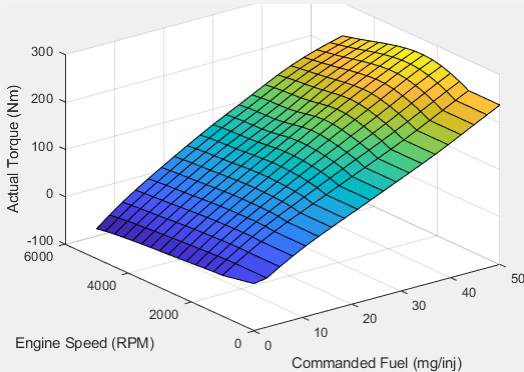
Map	Used For	In	Description
Engine brake torque	"CI Engine Simple Torque Model" on page 2-30	CI Core Engine CI Controller	<p>For the simple torque lookup table model, the CI engine uses a lookup table is a function of engine speed and injected fuel mass, $T_{brake} = f_{Tnf}(F, N)$, where:</p> <ul style="list-style-type: none"> • $Tq = T_{brake}$ 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. 

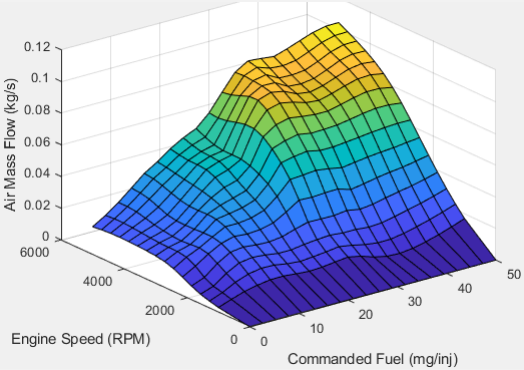
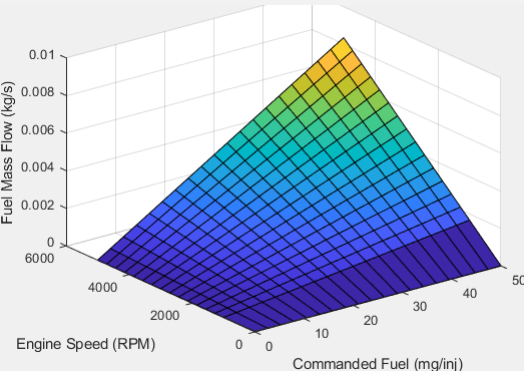
Map	Used For	In	Description
Hydrocarbon (HC) mass fraction	HC emissions	CI Core Engine	<p>The CI Core Engine HC emission mass fraction lookup table is a function of engine torque and engine speed, $HC\ Mass\ Fraction = f(Speed, Torque)$, where:</p> <ul style="list-style-type: none"> • <i>HC Mass Fraction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

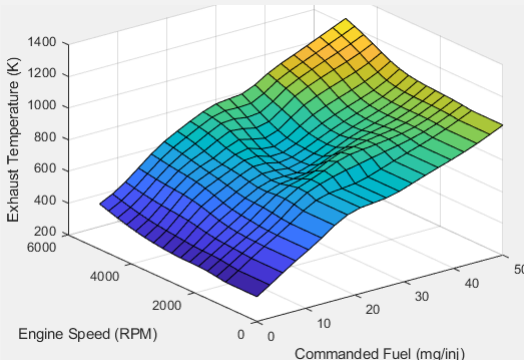
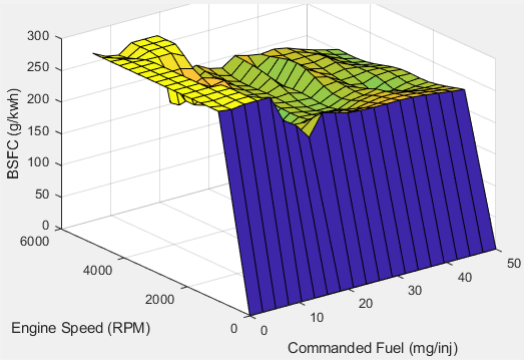
Map	Used For	In	Description
Carbon monoxide (CO) mass fraction	CO emissions	CI Core Engine	<p>The CI Core Engine CO emission mass fraction lookup table is a function of engine torque and engine speed, $CO\ Mass\ Fraction = f(Speed, Torque)$, where:</p> <ul style="list-style-type: none"> • <i>CO Mass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

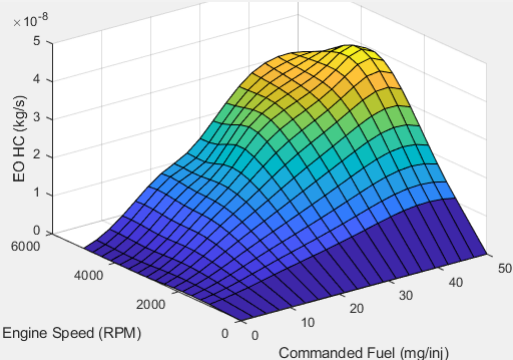
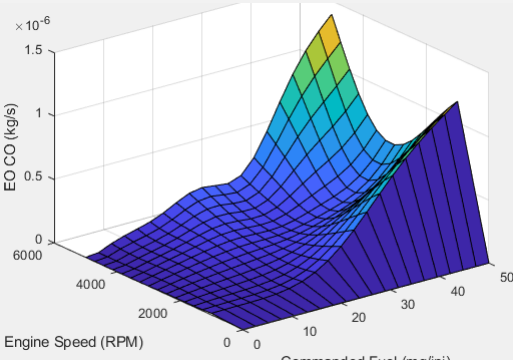
Map	Used For	In	Description
Nitric oxide and nitrogen dioxide (NOx) mass fraction	NOx emissions	CI Core Engine	<p>The CI Core Engine NOx emission mass fraction lookup table is a function of engine torque and engine speed, $NOx\ Mass\ Fraction = f(Speed, Torque)$, where:</p> <ul style="list-style-type: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

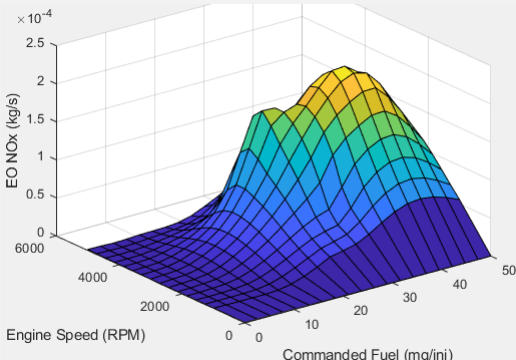
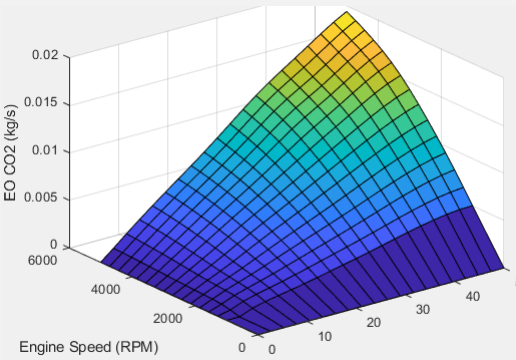
Map	Used For	In	Description
Carbon dioxide (CO ₂) mass fraction	CO ₂ emissions	CI Core Engine	<p>The CI Core Engine CO₂ emission mass fraction lookup table is a function of engine torque and engine speed, $CO_2 \text{ Mass Fraction} = f(\text{Speed}, \text{Torque})$, where:</p> <ul style="list-style-type: none"> • <i>CO₂ Mass Fraction</i> is the CO₂ emission mass fraction, dimensionless. • <i>Speed</i> is engine speed, in rpm. • <i>Torque</i> is engine torque, in N·m. 

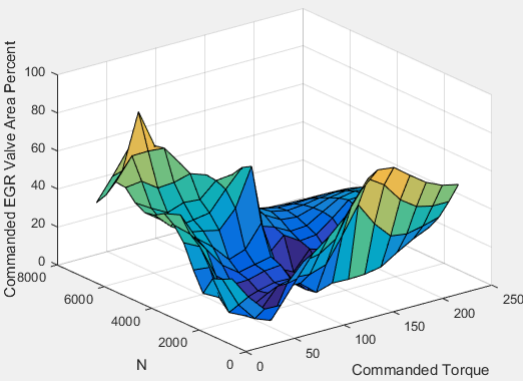
Map	Used For	In	Description
Exhaust temperature	Engine exhaust temperature as a function of injected fuel mass and engine speed	CI Core Engine CI Controller	<p>The lookup table for the exhaust temperature is a function of injected fuel mass and engine speed</p> $T_{exh} = f_{Texh}(F, N)$ <p>where:</p> <ul style="list-style-type: none"> • T_{exh} is exhaust temperature, in K. • F is injected fuel mass, in mg per injection. • N is engine speed, in rpm. 
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, $T_{brake} = f(F, N)$, where:</p> <ul style="list-style-type: none"> • T_{brake} is engine torque, in N·m. • F is commanded fuel mass, in mg per injection. • N is engine speed, in rpm. 

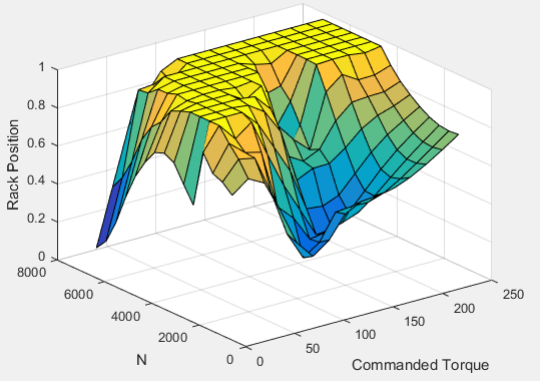
Map	Used For	In	Description
Engine air mass flow	Engine air mass flow as a function of commanded fuel mass and engine speed	Mapped CI Engine	<p>The air mass flow lookup table is a function of commanded fuel mass and engine speed, $\dot{m}_{intk} = f(F_{max}, N)$, where:</p> <ul style="list-style-type: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

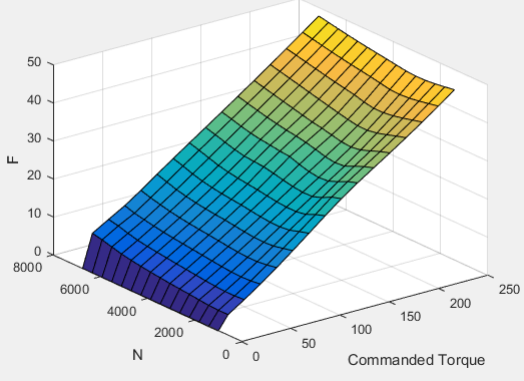
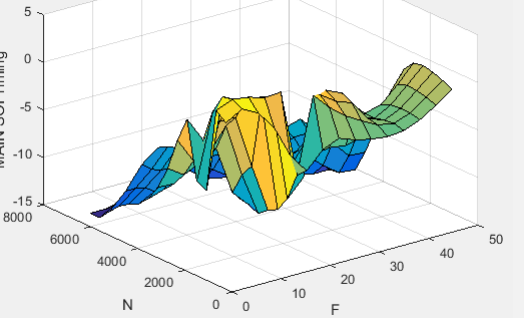
Map	Used For	In	Description
Engine exhaust temperature	Engine exhaust temperature as a function of commanded fuel mass and engine speed	Mapped CI Engine	<p>The engine exhaust temperature table is a function of commanded fuel mass and engine speed, $T_{exh} = f(F, N)$, where:</p> <ul style="list-style-type: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

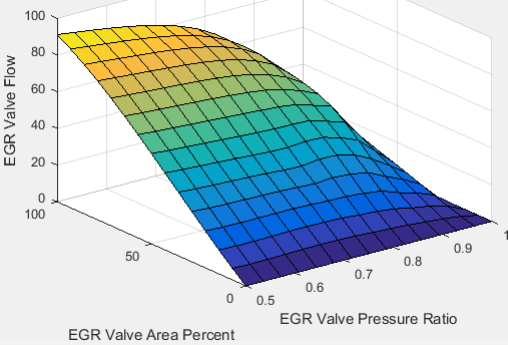
Map	Used For	In	Description
<p>Engine-out (EO) hydrocarbon emissions</p>	<p>EO hydrocarbon emissions as a function of commanded fuel mass and engine speed</p>	<p>Mapped CI Engine</p>	<p>The engine-out hydrocarbon emissions are a function of commanded fuel mass and engine speed, $EO\ HC = f(F, N)$, where: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

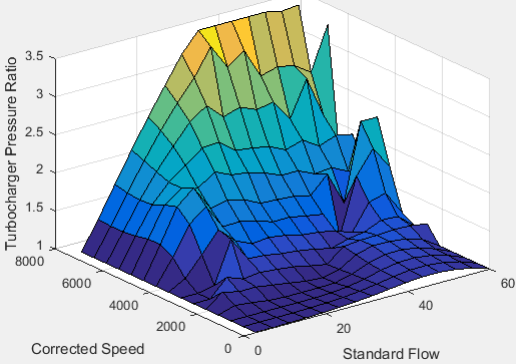
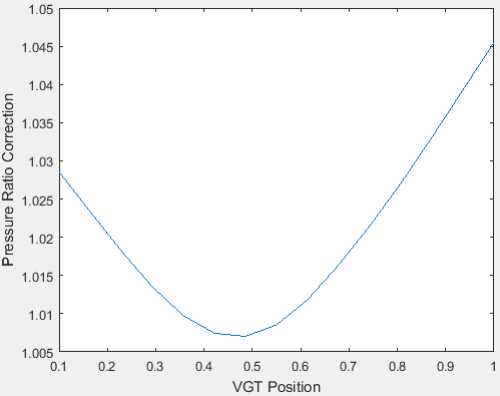
Map	Used For	In	Description
<p>Engine-out (EO) nitric oxide and nitrogen dioxide</p>	<p>EO nitric oxide and nitrogen dioxide emissions as a function of commanded fuel mass and engine speed</p>	<p>Mapped CI Engine</p>	<p>The engine-out nitric oxide and nitrogen dioxide emissions are a function of commanded fuel mass and engine speed, $EO NO_x = f(F, N)$, where:</p> <ul style="list-style-type: none"> • $EO NO_x$ is engine-out nitric oxide and nitrogen dioxide emissions, in kg/s. • F is commanded fuel mass, in mg per injection. • N is engine speed, in rpm.  <p>A 3D surface plot showing engine-out nitric oxide and nitrogen dioxide emissions (EO NO_x) in kg/s. The vertical axis is labeled 'EO NO_x (kg/s)' with a multiplier of 10⁻⁴ and ranges from 0 to 2.5. The horizontal axes are 'Engine Speed (RPM)' ranging from 0 to 6000 and 'Commanded Fuel (mg/inj)' ranging from 0 to 50. The surface shows a peak in emissions at approximately 2000 RPM and 20 mg/inj, with values decreasing as engine speed increases and fuel mass decreases.</p>
<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, $EO CO_2 = f(F, N)$, where:</p> <ul style="list-style-type: none"> • $EO CO_2$ is engine-out carbon dioxide emissions, in kg/s. • F is commanded fuel mass, in mg per injection. • N is engine speed, in rpm.  <p>A 3D surface plot showing engine-out carbon dioxide emissions (EO CO₂) in kg/s. The vertical axis is labeled 'EO CO₂ (kg/s)' and ranges from 0 to 0.02. The horizontal axes are 'Engine Speed (RPM)' ranging from 0 to 6000 and 'Commanded Fuel (mg/inj)' ranging from 0 to 50. The surface shows a peak in emissions at approximately 2000 RPM and 20 mg/inj, with values decreasing as engine speed increases and fuel mass decreases.</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"> • EGR_{cmd} is commanded EGR valve area percent, in percent. • Trq_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm. 

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> $RP_{cmd} = f_{RPcmd}(Trq_{cmd}, N)$ <p>where:</p> <ul style="list-style-type: none"> • RP_{cmd} is VGT rack position command, in percent. • Trq_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm.  <p>The 3D surface plot shows the VGT rack position (z-axis, 0 to 1) as a function of engine speed (N, x-axis, 0 to 8000 rpm) and commanded torque (y-axis, 0 to 250 N·m). The surface is highest (yellow, ~1.0) at low torque and low speed, and lowest (blue, ~0.0) at high torque and high speed.</p>

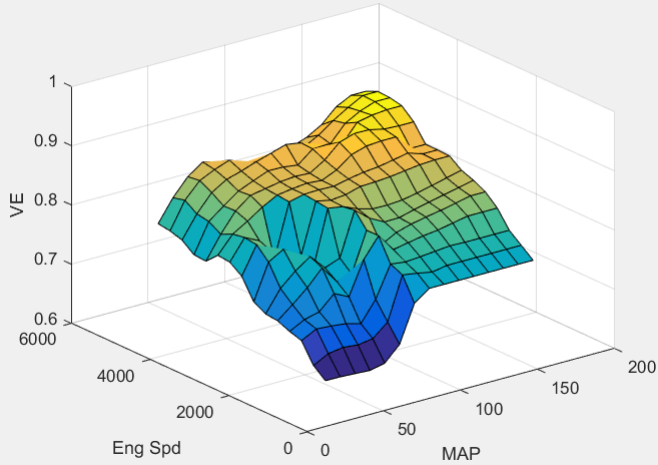
Map	Used For	In	Description
<p>Commanded total fuel mass per injection</p>	<p>Commanded total fuel mass per injection as a function of torque command and engine speed</p>	<p>CI Controller</p>	<p>The commanded total fuel mass per injection table is a function of the torque command and engine speed</p> $F_{cmd,tot} = f_{Fcmd,tot}(Trq_{cmd}, N)$ <p>where:</p> <ul style="list-style-type: none"> • $F_{cmd,tot} = F$ is commanded total fuel mass per injection, in mg per cylinder. • Trq_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm. 
<p>Main start-of-injection (SOI) timing</p>	<p>SOI timing as a function of commanded fuel mass and engine speed</p>	<p>CI Controller</p>	<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"> • $MAINSOI$ is the main start-of-injection timing, in degrees crank angle after top dead center (degATDC). • $F_{cmd,tot} = F$ is commanded fuel mass, in mg per injection. • N is engine speed, in rpm. 

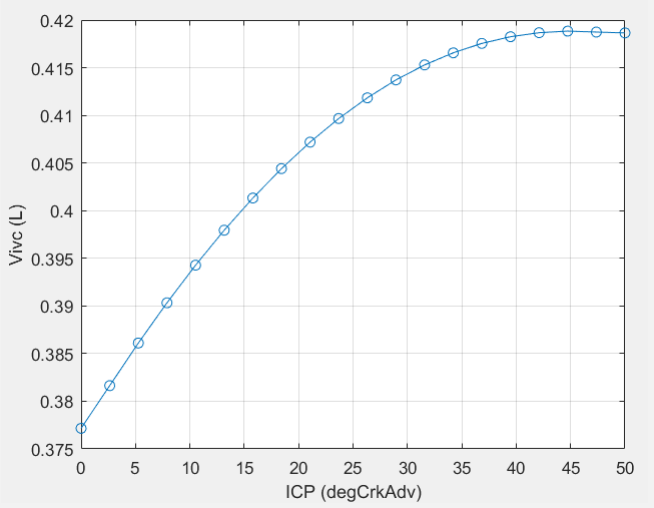
Map	Used For	In	Description
Standard exhaust gas recirculation (EGR) mass flow	EGR mass flow as a function of the standard flow pressure ratio and EGR valve flow area	CI Controller	<p>The standard exhaust gas recirculation (EGR) mass flow is a lookup table that is a function of the standard flow pressure 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"> • $\dot{m}_{egr, std}$ is the standard EGR valve mass flow, in g/s. • $P_{exh, est}$ is the estimated exhaust back-pressure, in Pa. • MAP is the cycle average intake manifold absolute pressure, in Pa. • $EGRap$ is the measured EGR valve area, in percent.  <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 colored with a gradient from blue (low flow) to yellow (high flow). The flow is highest at high area and low pressure ratio, and lowest at low area and high pressure ratio.</p>

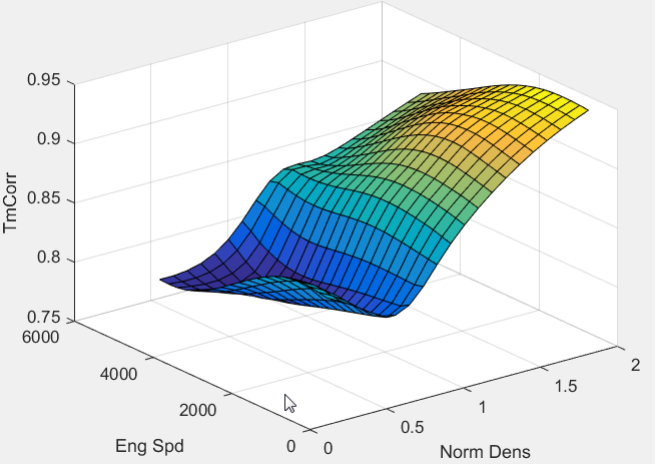
Map	Used For	In	Description
Turbocharger pressure ratio	Turbocharger pressure ratio as a function of the standard air mass flow and corrected turbocharger speed	CI Controller	<p>The turbocharger pressure ratio, corrected for variable geometry turbocharger (VGT) speed, is a lookup table that is a function of the standard air mass flow and corrected turbocharger speed, $Pr_{turbo} = f(\dot{m}_{airstd}, N_{vgtcrr})$, where:</p> <ul style="list-style-type: none"> • Pr_{turbo} is the turbocharger pressure ratio, corrected for VGT speed. • \dot{m}_{airstd} is the standard air mass flow, in g/s. • N_{vgtcrr} is the corrected turbocharger speed, in rpm/$K^{(1/2)}$.  <p>A 3D surface plot showing the Turbocharger Pressure Ratio (z-axis, ranging from 1 to 3.5) as a function of Corrected Speed (x-axis, ranging from 0 to 8000) and Standard Flow (y-axis, ranging from 0 to 60). The surface is colored with a gradient from blue (low pressure ratio) to yellow (high pressure ratio), showing a peak in pressure ratio at high corrected speeds and low standard flows.</p>
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, $Pr_{vgtcrr} = f(VGT_{pos})$, where:</p> <ul style="list-style-type: none"> • Pr_{vgtcrr} is the turbocharger pressure ratio correction. • VGT_{pos} is the variable geometry turbocharger (VGT) rack position.  <p>A 2D line graph showing the Pressure Ratio Correction (y-axis, ranging from 1.005 to 1.05) as a function of VGT Position (x-axis, ranging from 0.1 to 1). The curve is U-shaped, starting at approximately 1.028 at VGT Position 0.1, reaching a minimum of about 1.008 at VGT Position 0.5, and rising to approximately 1.045 at VGT Position 1.</p>

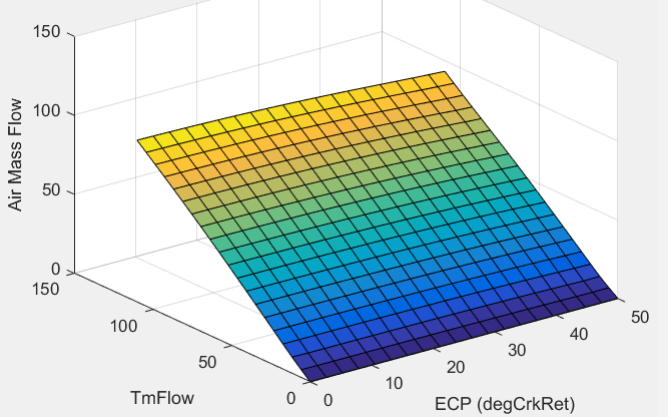
Calibration Maps in Spark-Ignition (SI) Blocks

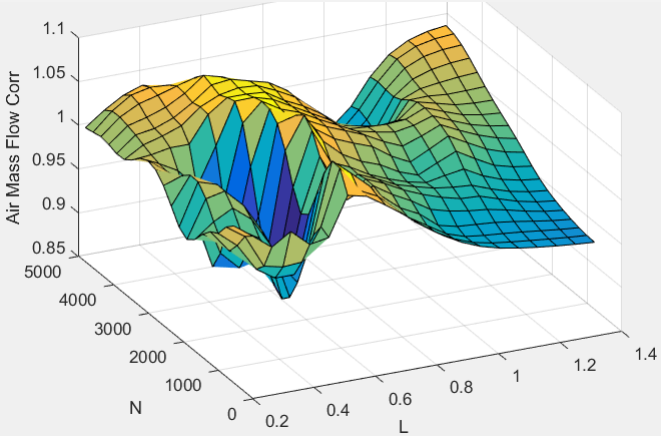
In the engine models, the Powertrain Blockset blocks implement these calibration maps.

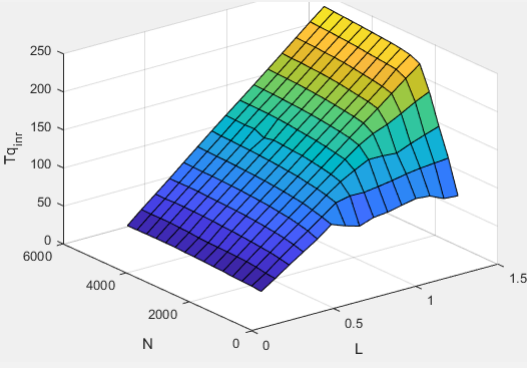
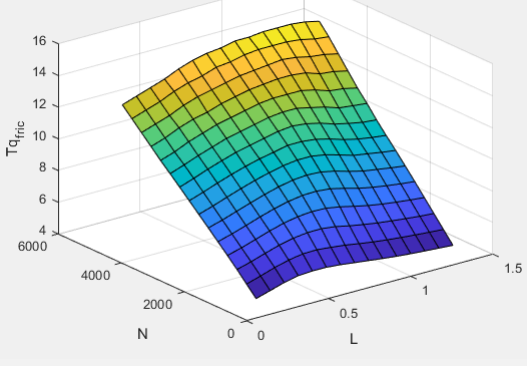
Map	Used for	In	Description
Engine volumetric efficiency	“SI Engine Speed-Density Air Mass Flow Model” on page 2-11	SI Core Engine SI Controller	<p>The engine volumetric efficiency lookup table, f_{η_v}, is a function of intake manifold absolute pressure and engine speed</p> $\eta_v = f_{\eta_v}(MAP, N)$ <p>where:</p> <ul style="list-style-type: none"> • η_v is engine volumetric efficiency, dimensionless. • MAP is intake manifold absolute pressure, in KPa. • N is engine speed, in rpm. 

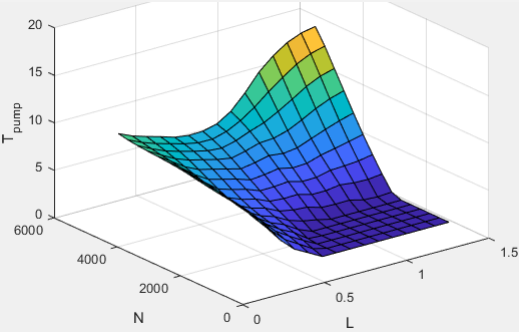
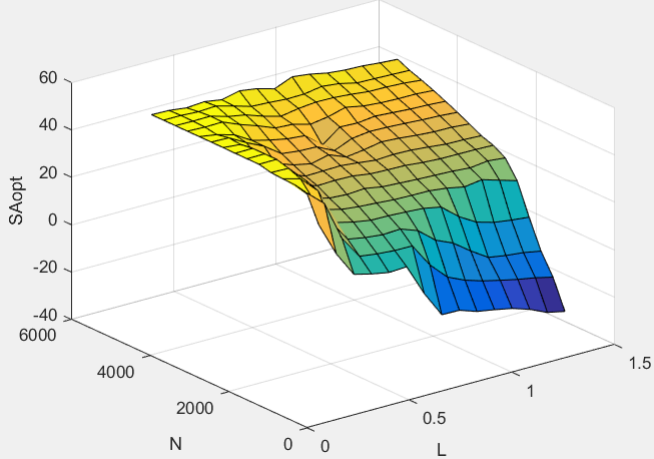
Map	Used for	In	Description																																																						
Cylinder volume at intake valve close table (IVC)	"SI Engine Dual-Independent Cam Phaser Air Mass Flow Model" on page 2-5	SI Core Engine SI Controller	<p>The cylinder volume at intake valve close table (IVC), $f_{V_{IVC}}$ 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"> • V_{IVC} is cylinder volume at IVC, in L. • φ_{ICP} is intake cam phaser angle, in crank advance degrees.  <table border="1" data-bbox="803 661 1453 1165"> <caption>Data points from the graph</caption> <thead> <tr> <th>ICP (degCrkAdv)</th> <th>V_{IVC} (L)</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.378</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>0.406</td></tr> <tr><td>16</td><td>0.410</td></tr> <tr><td>18</td><td>0.414</td></tr> <tr><td>20</td><td>0.418</td></tr> <tr><td>22</td><td>0.422</td></tr> <tr><td>24</td><td>0.426</td></tr> <tr><td>26</td><td>0.430</td></tr> <tr><td>28</td><td>0.434</td></tr> <tr><td>30</td><td>0.438</td></tr> <tr><td>32</td><td>0.442</td></tr> <tr><td>34</td><td>0.446</td></tr> <tr><td>36</td><td>0.448</td></tr> <tr><td>38</td><td>0.450</td></tr> <tr><td>40</td><td>0.451</td></tr> <tr><td>42</td><td>0.452</td></tr> <tr><td>44</td><td>0.452</td></tr> <tr><td>46</td><td>0.452</td></tr> <tr><td>48</td><td>0.452</td></tr> <tr><td>50</td><td>0.452</td></tr> </tbody> </table>	ICP (degCrkAdv)	V _{IVC} (L)	0	0.378	2	0.382	4	0.386	6	0.390	8	0.394	10	0.398	12	0.402	14	0.406	16	0.410	18	0.414	20	0.418	22	0.422	24	0.426	26	0.430	28	0.434	30	0.438	32	0.442	34	0.446	36	0.448	38	0.450	40	0.451	42	0.452	44	0.452	46	0.452	48	0.452	50	0.452
ICP (degCrkAdv)	V _{IVC} (L)																																																								
0	0.378																																																								
2	0.382																																																								
4	0.386																																																								
6	0.390																																																								
8	0.394																																																								
10	0.398																																																								
12	0.402																																																								
14	0.406																																																								
16	0.410																																																								
18	0.414																																																								
20	0.418																																																								
22	0.422																																																								
24	0.426																																																								
26	0.430																																																								
28	0.434																																																								
30	0.438																																																								
32	0.442																																																								
34	0.446																																																								
36	0.448																																																								
38	0.450																																																								
40	0.451																																																								
42	0.452																																																								
44	0.452																																																								
46	0.452																																																								
48	0.452																																																								
50	0.452																																																								

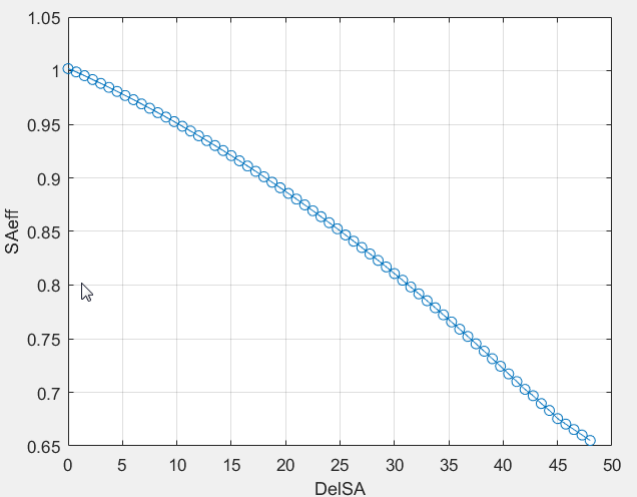
Map	Used for	In	Description
Trapped mass correction	"SI Engine Dual-Independent Cam Phaser Air Mass Flow Model" on page 2-5	SI Core Engine SI Controller	<p>The trapped mass correction factor table, $f_{TM_{corr}}$, is a function of the normalized density and engine speed</p> $TM_{corr} = f_{TM_{corr}}(\rho_{norm},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

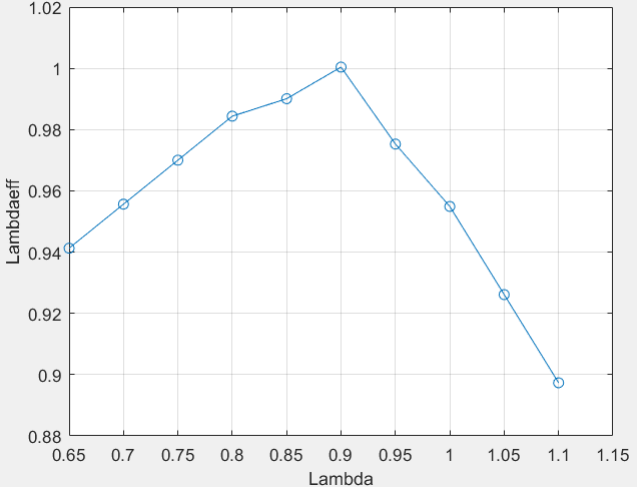
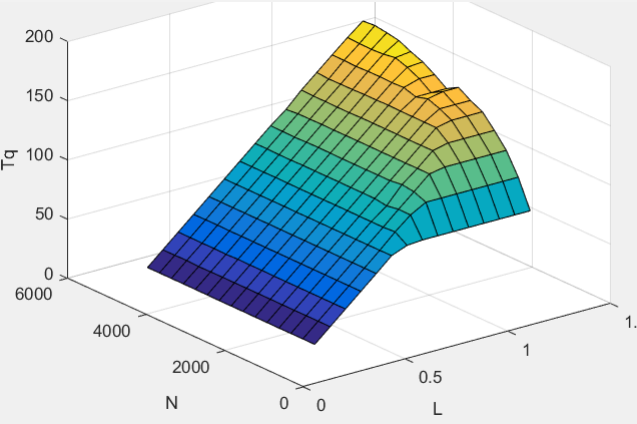
Map	Used for	In	Description
Air mass flow at cam phaser angles	"SI Engine Dual-Independent Cam Phaser Air Mass Flow Model" on page 2-5	SI Core Engine SI Controller	<p>The phaser intake mass flow model lookup table is a function of exhaust cam phaser angles and trapped air mass flow</p> $\dot{m}_{intkideal}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

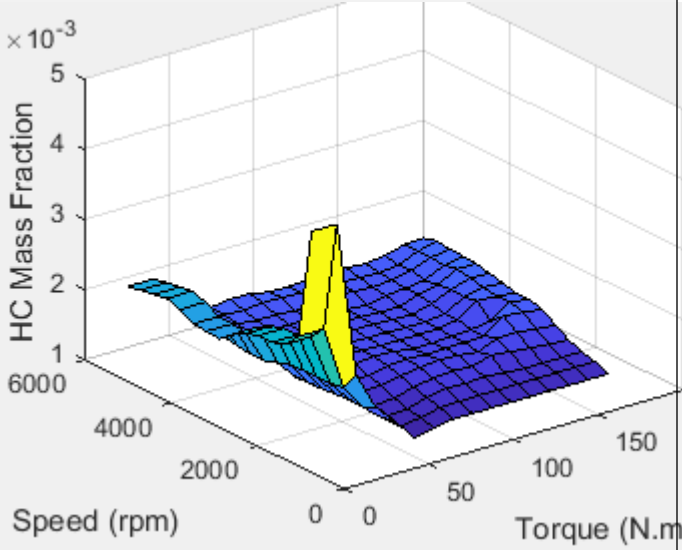
Map	Used for	In	Description
Air mass flow correction	“SI Engine Dual-Independent Cam Phaser Air Mass Flow Model” on page 2-5	SI Core Engine SI Controller	<p>The intake air mass flow correction lookup table, $f_{aircorr}$, is a function of ideal load and engine speed</p> $\dot{m}_{air} = \dot{m}_{intkideal} f_{aircorr}(L_{ideal}, N)$ <p>where:</p> <ul style="list-style-type: none"> • 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. 

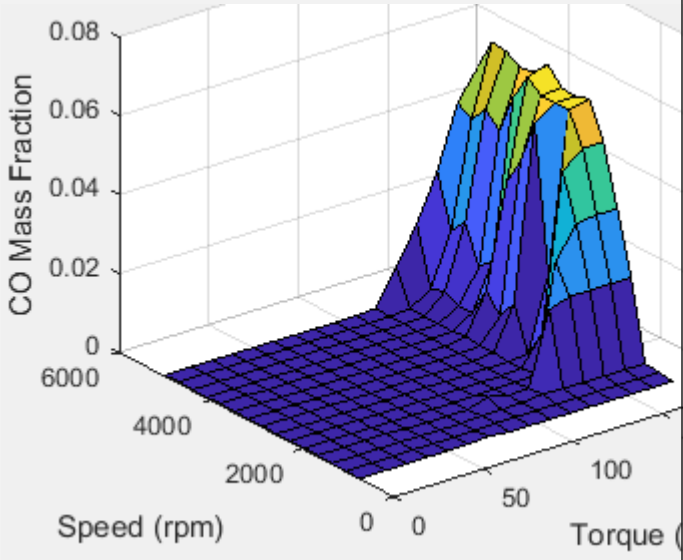
Map	Used for	In	Description
Inner torque	"SI Engine Torque Structure Model" on page 2-14	SI Core Engine SI Controller	<p>The inner torque lookup table, $f_{Tq_{inr}}$, is a function of engine speed and engine load, $Tq_{inr} = f_{Tq_{inr}}(L, N)$, where:</p> <ul style="list-style-type: none"> • Tq_{inr} is inner torque based on gross indicated mean effective pressure, in N·m. • L is engine load at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles, dimensionless. • N is engine speed, in rpm. 
Friction torque	"SI Engine Torque Structure Model" on page 2-14	SI Core Engine SI Controller	<p>The friction torque lookup table, $f_{Tf_{fric}}$, is a function of engine speed and engine load, $Tf_{fric} = f_{Tf_{fric}}(L, N)$, where:</p> <ul style="list-style-type: none"> • Tf_{fric} is friction torque offset to inner torque, in N·m. • L is engine load at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles, dimensionless. • N is engine speed, in rpm. 

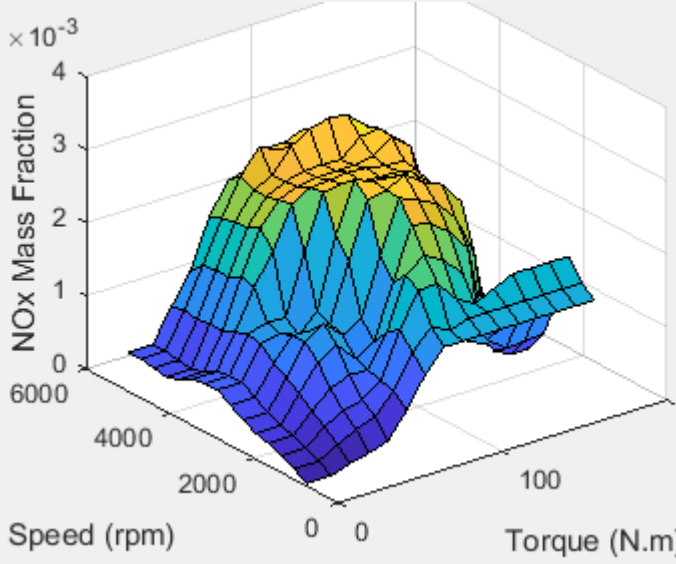
Map	Used for	In	Description
Pumping torque	"SI Engine Torque Structure Model" on page 2-14	SI Core Engine SI Controller	<p>The pumping work lookup table, $f_{T_{pump}}$, is a function of engine load and engine speed, $T_{pump} = f_{T_{pump}}(L, N)$, where:</p> <ul style="list-style-type: none"> T_{pump} is pumping work, in N·m. L is engine load, as a normalized cylinder air mass, dimensionless. N is engine speed, in rpm. 
Optimal spark advance	"SI Engine Torque Structure Model" on page 2-14	SI Core Engine SI Controller	<p>The optimal spark lookup table, $f_{SA_{opt}}$, is a function of engine speed and engine load, $SA_{opt} = f_{SA_{opt}}(L, N)$, where:</p> <ul style="list-style-type: none"> SA_{opt} is optimal spark advance timing for maximum inner torque at stoichiometric air-fuel ratio (AFR), in deg. L is engine load at arbitrary cam phaser angles, corrected for final steady-state cam phaser angles, dimensionless. N is engine speed, in rpm. 

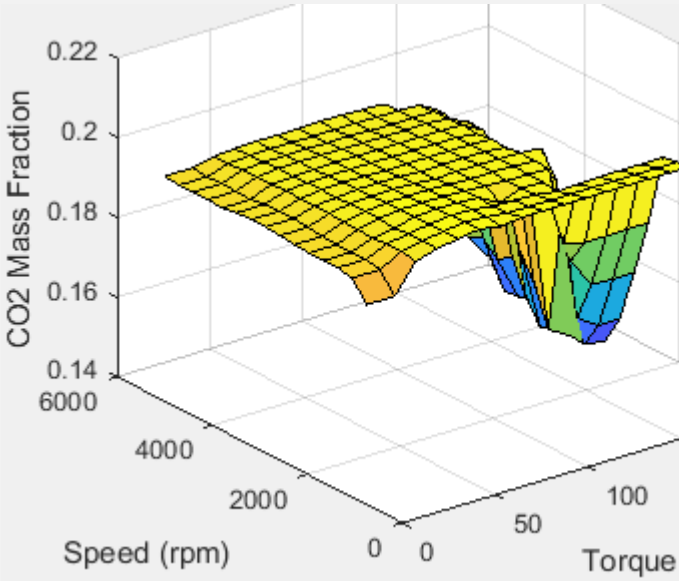
Map	Used for	In	Description
Spark efficiency	"SI Engine Torque Structure Model" on page 2-14	SI Core Engine SI Controller	<p>The spark efficiency lookup table, f_{Msa}, is a function of the spark retard from optimal</p> $M_{sa} = f_{Msa}(\Delta SA)$ $\Delta SA = SA_{opt} - SA$ <p>where:</p> <ul style="list-style-type: none"> • M_{sa} is the spark retard efficiency multiplier, dimensionless. • ΔSA is the spark retard timing distance from optimal spark advance, in deg. 

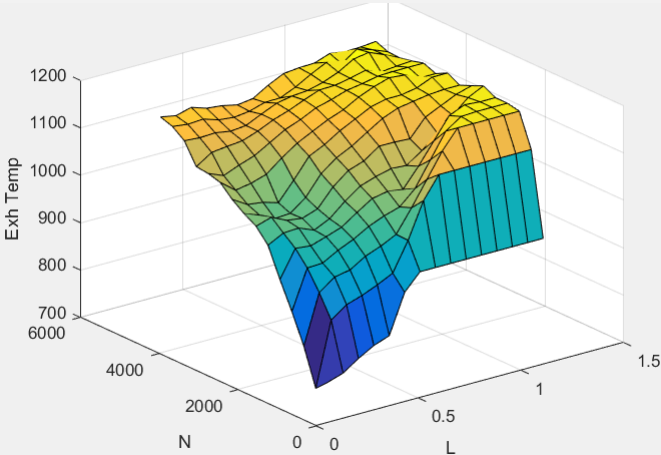
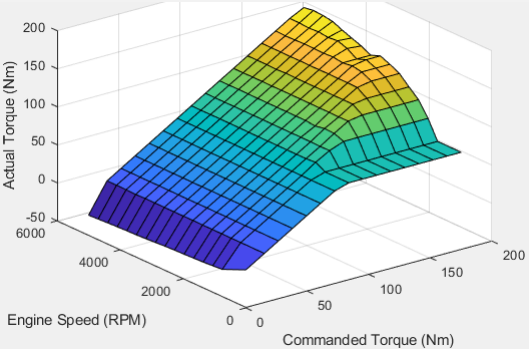
Map	Used for	In	Description
Lambda efficiency	"SI Engine Torque Structure Model" on page 2-14	SI Core Engine SI Controller	<p>The lambda efficiency lookup table, $f_{M\lambda}$, is a function of lambda, $M_\lambda = f_{M\lambda}(\lambda)$, where:</p> <ul style="list-style-type: none"> M_λ is the lambda multiplier on inner torque to account for the air-fuel ratio (AFR) effect, dimensionless. λ is lambda, AFR normalized to stoichiometric fuel AFR, dimensionless. 
Simple torque	"SI Engine Simple Torque Model" on page 2-20	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, $T_{brake} = f_{TnL}(L, N)$, where:</p> <ul style="list-style-type: none"> 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. 

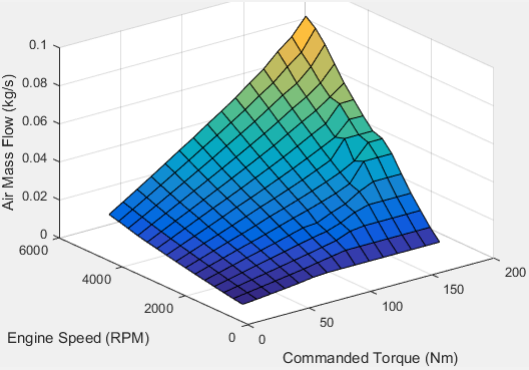
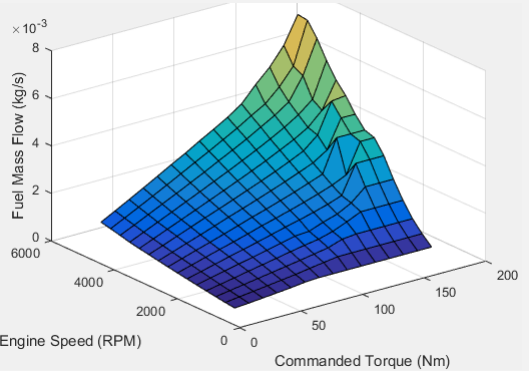
Map	Used for	In	Description
Hydrocarbon (HC) mass fraction	HC emissions	SI Core Engine	<p>The SI Core Engine HC emission mass fraction lookup table is a function of engine torque and engine speed, $HC\ Mass\ Fraction = f(Speed, Torque)$, where:</p> <ul style="list-style-type: none"> • <i>HC Mass Fraction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

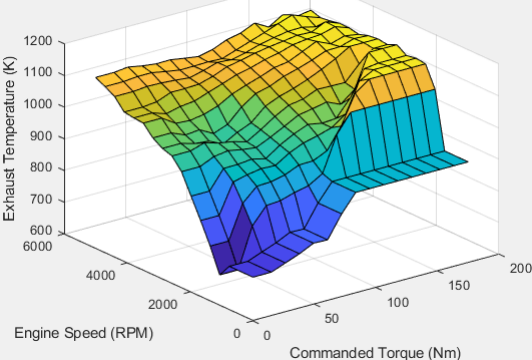
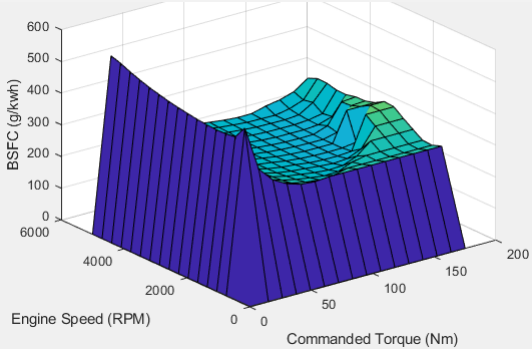
Map	Used for	In	Description
Carbon monoxide (CO) mass fraction	CO emissions	SI Core Engine	<p>The SI Core Engine CO emission mass fraction lookup table is a function of engine torque and engine speed, $CO\ Mass\ Fraction = f(Speed, Torque)$, where:</p> <ul style="list-style-type: none"> • <i>CO Mass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

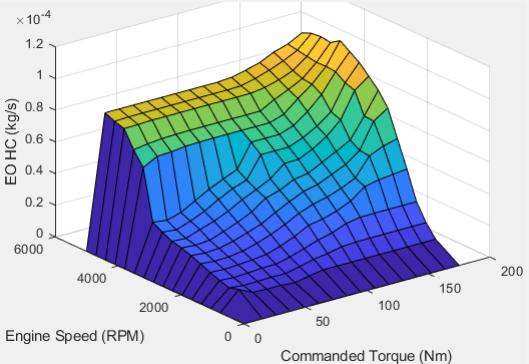
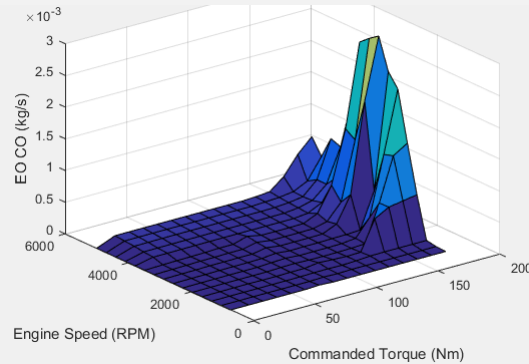
Map	Used for	In	Description
Nitric oxide and nitrogen dioxide (NOx) mass fraction	NOx emissions	SI Core Engine	<p>The SI Core Engine NOx emission mass fraction lookup table is a function of engine torque and engine speed, $NOx\ Mass\ Fraction = f(Speed, Torque)$, where:</p> <ul style="list-style-type: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

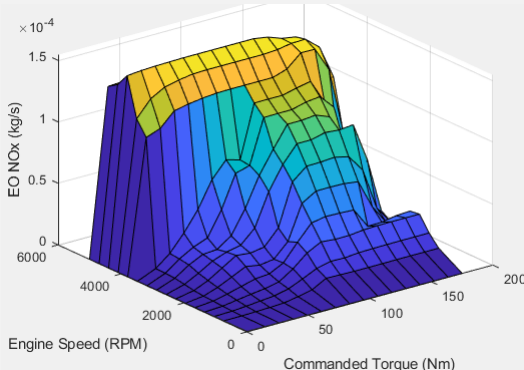
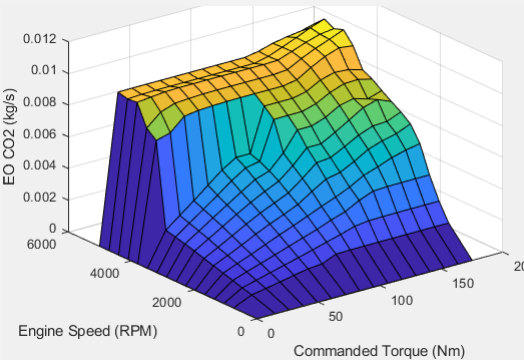
Map	Used for	In	Description
Carbon dioxide (CO ₂) mass fraction	CO ₂ emissions	SI Core Engine	<p>The SI Core Engine CO₂ emission mass fraction lookup table is a function of engine torque and engine speed, $CO_2 \text{ Mass Fraction} = f(\text{Speed}, \text{Torque})$, where:</p> <ul style="list-style-type: none"> • <i>CO₂ Mass Fraction</i> is the CO₂ emission mass fraction, dimensionless. • <i>Speed</i> is engine speed, in rpm. • <i>Torque</i> is engine torque, in N·m. 

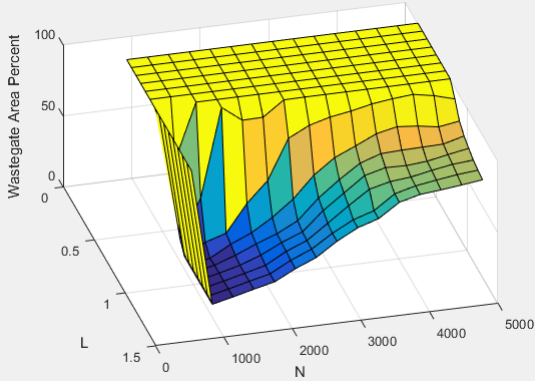
Map	Used for	In	Description
Exhaust temperature	Engine exhaust calculation as a function of engine speed and load	SI Core Engine SI Controller	<p>The exhaust temperature lookup table, f_{Texh}, is a function of engine load and engine speed</p> $T_{exh} = f_{Texh}(L, N)$ <p>where:</p> <ul style="list-style-type: none"> • T_{exh} is engine exhaust temperature, in K. • L is normalized cylinder air mass or engine load, dimensionless. • N is engine speed, in rpm. 
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, $T = f(T_{cmd}, N)$, where:</p> <ul style="list-style-type: none"> • T is engine torque, in N·m. • T_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm. 

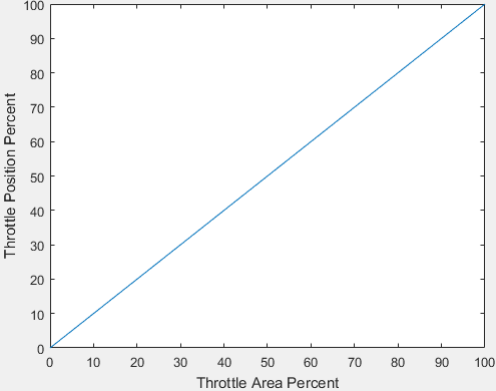
Map	Used for	In	Description
Engine air mass flow	Engine air mass flow as a function of commanded torque and engine speed	Mapped SI Engine	<p>The engine air mass flow lookup table is a function of commanded engine torque and engine speed, $\dot{m}_{intk} = f(T_{cmd}, N)$, where:</p> <ul style="list-style-type: none"> • \dot{m}_{intk} is engine air mass flow, in kg/s. • T_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm. 
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, $MassFlow = f(T_{cmd}, N)$, where:</p> <ul style="list-style-type: none"> • $MassFlow$ is engine fuel mass flow, in kg/s. • T_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm. 

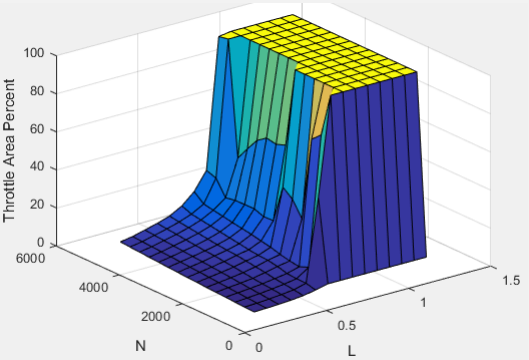
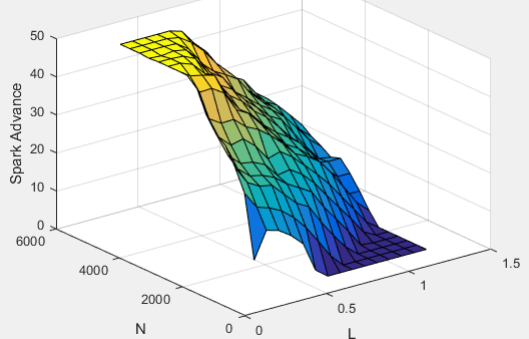
Map	Used for	In	Description
Engine exhaust temperature	Engine exhaust temperature as a function of commanded torque and engine speed	Mapped SI Engine	<p>The engine exhaust temperature lookup table is a function of commanded engine torque and engine speed, $T_{exh} = f(T_{cmd}, N)$, where:</p> <ul style="list-style-type: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

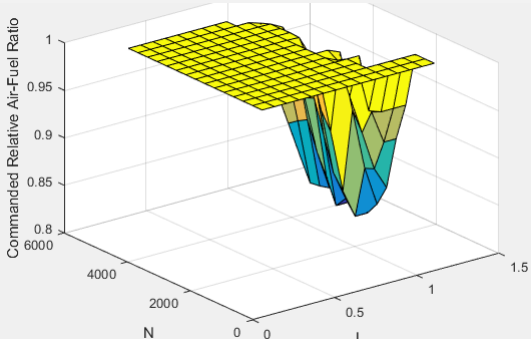
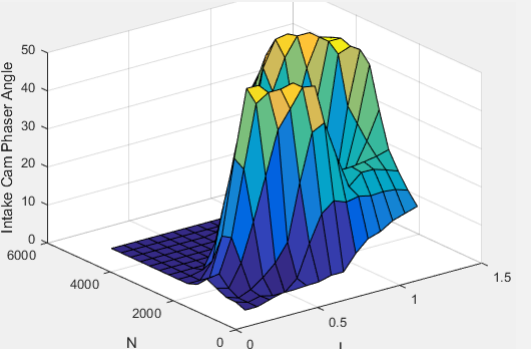
Map	Used for	In	Description
<p>Engine-out (EO) hydrocarbon emissions</p>	<p>EO hydrocarbon emissions as a function of commanded torque and engine speed</p>	<p>Mapped SI Engine</p>	<p>The engine-out hydrocarbon emissions are a function of commanded engine torque and engine speed, $EO HC = f(T_{cmd}, N)$, where:</p> <ul style="list-style-type: none"> • $EO HC$ is engine-out hydrocarbon emissions, in kg/s. • T_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm.  <p>A 3D surface plot showing engine-out hydrocarbon (EO HC) emissions in kg/s. The vertical axis is labeled 'EO HC (kg/s)' with a multiplier of $\times 10^{-4}$ and ranges from 0 to 1.2. The horizontal axes are 'Engine Speed (RPM)' ranging from 0 to 4000 and 'Commanded Torque (Nm)' ranging from 0 to 200. The surface shows a peak in emissions at high engine speeds and low torque, with values reaching approximately 1.2×10^{-4} kg/s.</p>
<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, $EO CO = f(T_{cmd}, N)$, where:</p> <ul style="list-style-type: none"> • $EO CO$ is engine-out carbon monoxide emissions, in kg/s. • T_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm.  <p>A 3D surface plot showing engine-out carbon monoxide (EO CO) emissions in kg/s. The vertical axis is labeled 'EO CO (kg/s)' with a multiplier of $\times 10^{-3}$ and ranges from 0 to 3. The horizontal axes are 'Engine Speed (RPM)' ranging from 0 to 4000 and 'Commanded Torque (Nm)' ranging from 0 to 200. The surface shows a sharp peak in emissions at high engine speeds and low torque, with values reaching approximately 3×10^{-3} kg/s.</p>

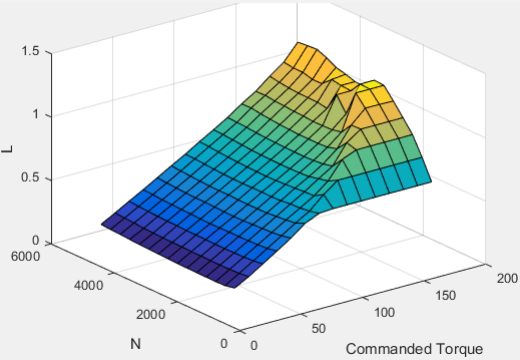
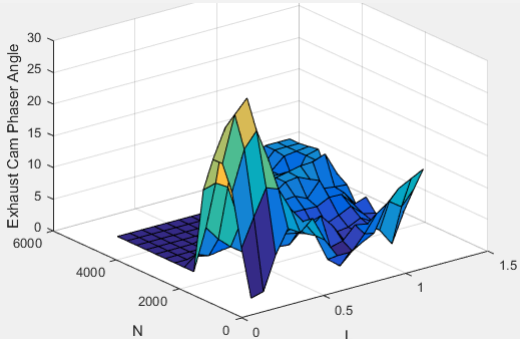
Map	Used for	In	Description
<p>Engine-out (EO) nitric oxide and nitrogen dioxide emissions</p>	<p>EO nitric oxide and nitrogen dioxide emissions as a function of commanded torque and engine speed</p>	<p>Mapped SI Engine</p>	<p>The engine-out nitric oxide and nitrogen dioxide emissions are a function of commanded engine torque and engine speed, $EO\ NO_x = f(T_{cmd}, N)$, where:</p> <ul style="list-style-type: none"> • $EO\ NO_x$ is engine-out nitric oxide and nitrogen dioxide emissions, in kg/s. • T_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm. 
<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, $EO\ CO_2 = f(T_{cmd}, N)$, where:</p> <ul style="list-style-type: none"> • $EO\ CO_2$ is engine-out carbon dioxide emissions, in kg/s. • T_{cmd} is commanded engine torque, in N·m. • N is engine speed, in rpm. 

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

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, $f_{TPP_{cmd}}$, 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"> • TPP_{cmd} is throttle position percentage command, in percent. • TAP_{cmd} is throttle area percentage command, in percent. 

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, f_{TAPcmd}, 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"> • TAP_{cmd} is throttle area percentage command, in percent. • $L_{cmd}=L$ is commanded engine load, dimensionless. • N is engine speed, in rpm. 
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"> • SA is spark advance, in crank advance degrees. • $L_{est}=L$ is estimated engine load, dimensionless. • N is engine speed, in rpm. 

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, λ_{cmd}, 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"> • λ_{cmd} is commanded relative AFR, dimensionless. • $L_{est}=L$ is estimated engine load, dimensionless. • N is engine speed, in rpm. 
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, f_{ICPCMD}, 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"> • φ_{ICPCMD} is commanded intake cam phaser angle, in degrees crank advance. • $L_{est}=L$ is estimated engine load, dimensionless. • N is engine speed, in rpm. 

Map	Used for	In	Description
Commanded engine load	Commanded engine load as a function of the commanded torque and engine speed	SI Controller	<p>The commanded engine load lookup table, f_{Lcmd}, 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"> • $L_{cmd}=L$ is commanded engine load, dimensionless. • T_{cmd} is commanded torque, in N·m. • N is engine speed, in rpm. 
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, f_{ECPCMD}, 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"> • φ_{ECPCMD} is commanded exhaust cam phaser angle, in degrees crank retard. • $L_{est}=L$ is estimated engine load, dimensionless. • N is engine speed, in rpm. 

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 3-14

See Also

Related Examples

- “Resize the CI Engine” on page 3-77
- “Resize the SI Engine” on page 3-84

More About

- “Analyze Power and Energy” on page 3-107
- “Hybrid and Electric Vehicle Reference Application Projects” on page 3-3
- “Internal Combustion Mapped and Dynamic Engine Models” on page 3-106

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 3-18
	HEV — Input power-split	“Explore the Hybrid Electric Vehicle Input Power-Split Reference Application” on page 3-31
	HEV — P0	“Explore the Hybrid Electric Vehicle P0 Reference Application” on page 3-40
	HEV — P1	“Explore the Hybrid Electric Vehicle P1 Reference Application” on page 3-47
	HEV — P2	“Explore the Hybrid Electric Vehicle P2 Reference Application” on page 3-54
	HEV — P3	“Explore the Hybrid Electric Vehicle P3 Reference Application” on page 3-63
	HEV — P4	“Explore the Hybrid Electric Vehicle P4 Reference Application” on page 3-70
	Electric vehicle	“Explore the Electric Vehicle Reference Application” on page 3-25

See Also

More About

- “Analyze Power and Energy” on page 3-107
- “Internal Combustion Engine Reference Application Projects” on page 3-2

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

```
autoblConVehStart
```

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
Analyze Power and Energy	Double-click Analyze Power and Energy to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level. For more information about the live script, see "Analyze Power and Energy" on page 3-107.	NA
Drive Cycle Source block — FTP75 (2474 seconds)	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.	

Reference Application Element	Description	Variants
Longitudinal Driver subsystem	<p>Uses the Longitudinal Driver or Open Loop variant to generate normalized acceleration and braking commands.</p> <ul style="list-style-type: none"> Longitudinal Driver variant implements a driver model that uses vehicle target and reference velocities. Open Loop variant allows you to configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs. 	✓
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.	

Optimize Transmission Shift Maps

You can use the conventional vehicle reference application to optimize the transmission control module (TCM) shift schedules. Use the optimized shift schedules to:

- Design control algorithms.
- Assess the impact of powertrain changes, such as an engine or gear ratio, on performance, fuel economy, and emissions.

TCM shift schedule optimization requires Simulink Design Optimization, the Global Optimization Toolbox, and Stateflow. To increase the performance of the optimization, consider also using the Parallel Computing Toolbox.

To run the TCM shift schedule optimization, open a version of the conventional vehicle reference application that includes the option to optimize transmission shift maps by using this command:

```
autoblkConVehShftOptStart
```

Click **Optimize Transmission Shift Maps**. Optimizing the shift schedules can take time to run.

For more information, see “Optimize Transmission Control Module Shift Schedules” on page 7-13.

Evaluate and Report Power and Energy

Double-click **Analyze Power and Energy** to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level.

The script provides:

- An overall energy summary that you can export to an Excel spreadsheet.
- Engine plant and drivetrain efficiencies, including an engine plant histogram of time spent at the different engine efficiencies.
- Data logging so that you can use the Simulation Data Inspector to analyze the powertrain efficiency and energy transfer signals.

For more information about the live script, see “Analyze Power and Energy” on page 3-107.

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. The reference application has these variants.

Block Variants			Description
Longitudinal Driver (default)	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	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	No transmission.

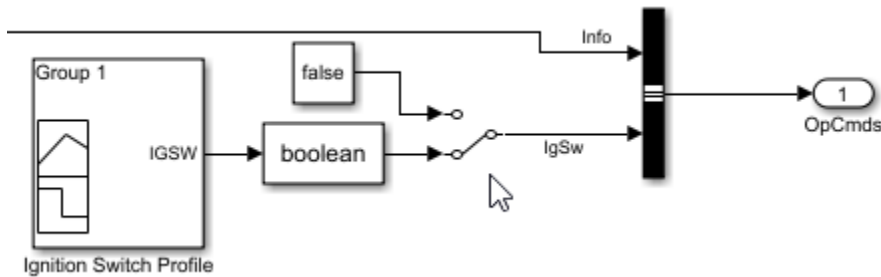
Block Variants		Description
	Scheduled	Stateflow chart models reverse, neutral, park, and N-speed gear shift scheduling.
Open Loop		Open-loop control subsystem. In the subsystem, you can configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs.

To idle the engine at the beginning of a drive cycle and simulate catalyst light-off before moving the vehicle with a pedal command, use the Longitudinal Driver variant. The Longitudinal Driver subsystem includes an ignition switch signal profile, `IgSw`. The engine controller uses the ignition switch signal to start both the engine and a catalyst light-off timer.

The catalyst light-off timer overrides the engine stop-start (ESS) stop function control while the catalyst light-off timer is counting up. During the simulation, after the `IgSw` down-edge time reaches the catalyst light-off time `CatLightOffTime`, normal ESS operation resumes. If there is no torque command before the simulation reaches the `EngStopTime`, the ESS shuts down the engine.

To control ESS and catalyst light-off:

- In the Longitudinal Driver Model subsystem, set the ignition switch profile `IgSw` to 'on'.



- In the engine controller model workspace, set these calibration parameters:
 - `EngStopStartEnable` – Enables ESS. To disable ESS, set the value to false.
 - `CatLightOffTime` – Engine idle time from engine start to catalyst light-off.
 - `EngStopTime` – ESS engine run time after driver model torque request cut-off.

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"> • Brake • Force calculation • Resistance calculation • Vertical motion <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	

Engine Subsystem	Variant	Description
Engine	SiEngineCore	Dynamic SI Core Engine with turbocharger
	SiEngineCoreNA	Dynamic naturally aspirated SI Core Engine
	SiEngineCoreV	Dynamic SI V Twin-Turbo Single-Intake Engine
	SiEngineCoreVNA	Dynamic SI V Engine
	SiEngineCoreVThr2	Dynamic SI V Twin-Turbo Twin-Intake Engine
	SiMappedEngine (default)	Mapped SI Engine with implicit turbocharger
	SiDLEngine	Deep learning SI engine
	CiEngine	Dynamic CI Core 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 Reference Application” on page 7-2
- “Conventional Vehicle Spark-Ignition Engine Fuel Economy and Emissions” on page 1-10
- “Conventional Vehicle Powertrain Efficiency” on page 1-15
- “Optimize Transmission Control Module Shift Schedules” on page 7-13
- “Track Drive Cycle Errors” on page 5-3

More About

- “Analyze Power and Energy” on page 3-107
- “Internal Combustion Engine Reference Application Projects” on page 3-2
- Simulation Data Inspector
- “Variant Systems”

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.	✓	✓
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 pseudo random 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.		✓

Test	Objective	Method	CI Engine Variant	
			Mapped	Dynamic
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. Also, the dynamometer recalibrates the controller and mapped engine model to match the resized dynamic engine.	✓	✓
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.	✓	

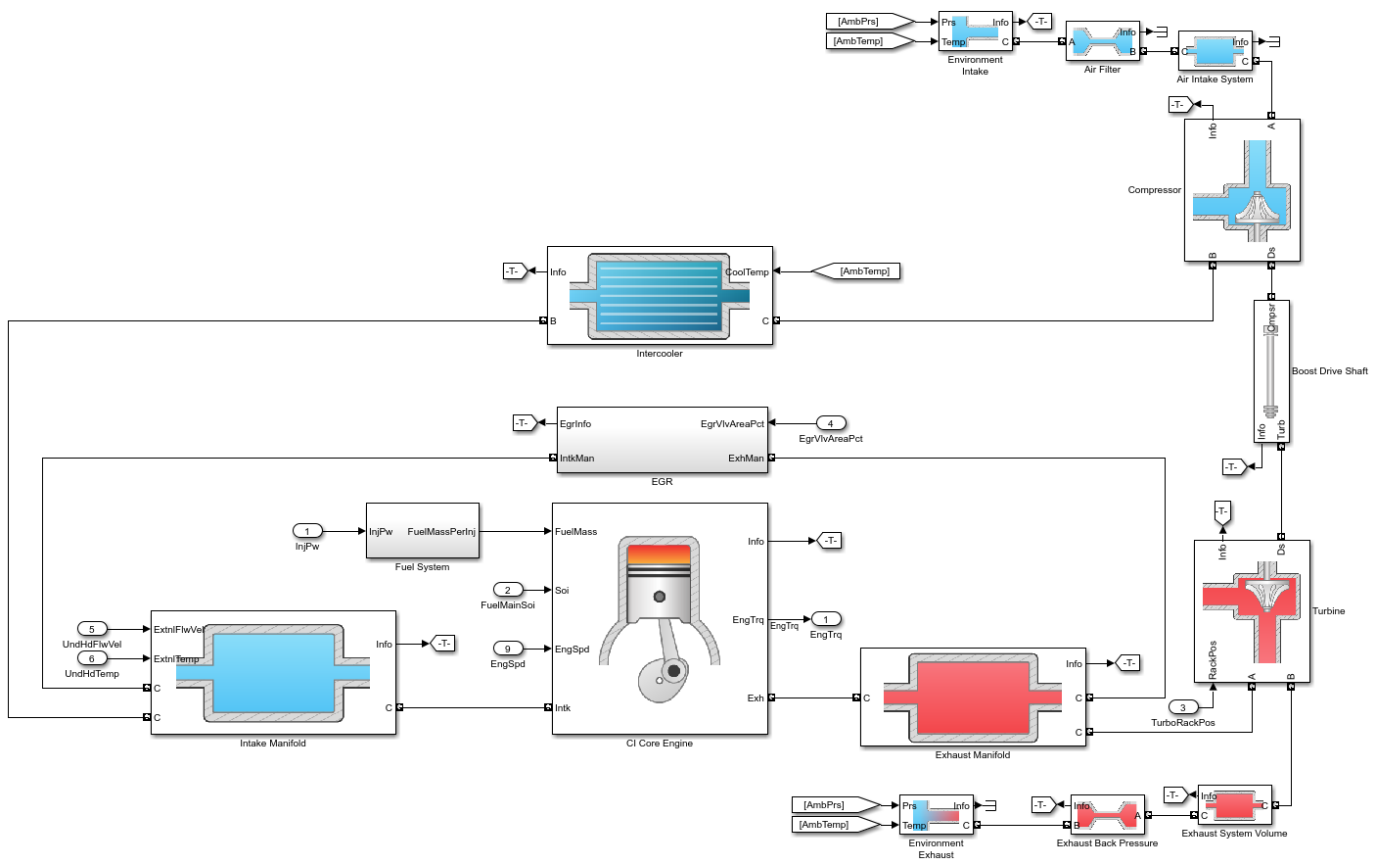
Engine System

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

Related Examples

- “CI Engine Dynamometer Reference Application” on page 7-11
- “Generate Mapped CI Engine from a Spreadsheet” on page 3-91
- “Resize the CI Engine” on page 3-77

More About

- “CI Engine Project Template” on page 4-2
- “Internal Combustion Mapped and Dynamic Engine Models” on page 3-106
- “Variant Systems”

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

```
autoblkSIDynamometerStart
```

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	Dynami c
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.	✓	✓
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 pseudo random 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.		✓

Test	Objective	Method	SI Engine Variant	
			Mapped	Dynamic
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. Also, the dynamometer recalibrates the controller and mapped engine model to match the resized dynamic engine. For an example, see “Resize the SI Engine” on page 3-84.	✓	✓
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. For an example, see “Generate Mapped SI Engine from a Spreadsheet” on page 3-96.	✓	
Generate Deep Learning Engine Model	Train a deep learning model of dynamic engine behavior from measured laboratory data or a high-fidelity engine model.	Dynamometer uses the Deep Learning Toolbox™ and Statistics and Machine Learning Toolbox™ to generate a dynamic deep learning engine model and update the mapped engine parameters. For an example, see “Generate a Deep Learning SI Engine Model” on page 3-100.	✓	

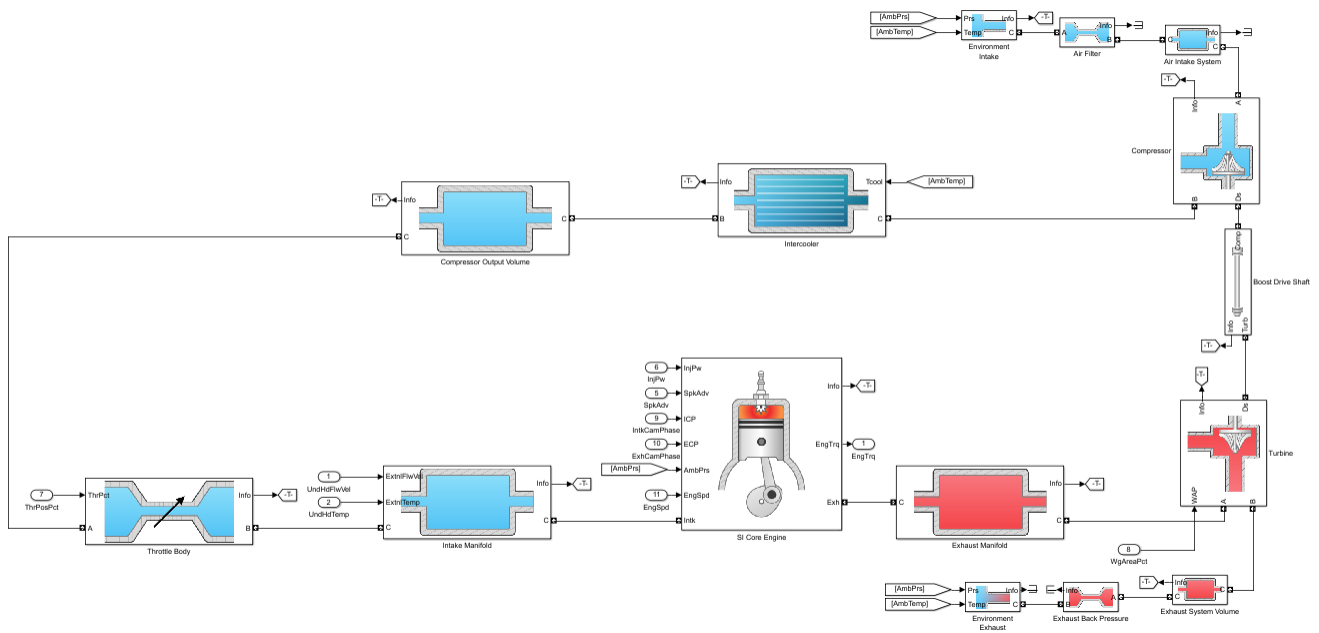
Engine System

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

Related Examples

- “SI Engine Dynamometer Reference Application” on page 7-12
- “Generate Mapped SI Engine from a Spreadsheet” on page 3-96
- “Generate a Deep Learning SI Engine Model” on page 3-100
- “Resize the SI Engine” on page 3-84

More About

- “SI Engine Project Template” on page 4-4
- “Internal Combustion Mapped and Dynamic Engine Models” on page 3-106

- “Variant Systems”

Explore the Hybrid Electric Vehicle Multimode Reference Application

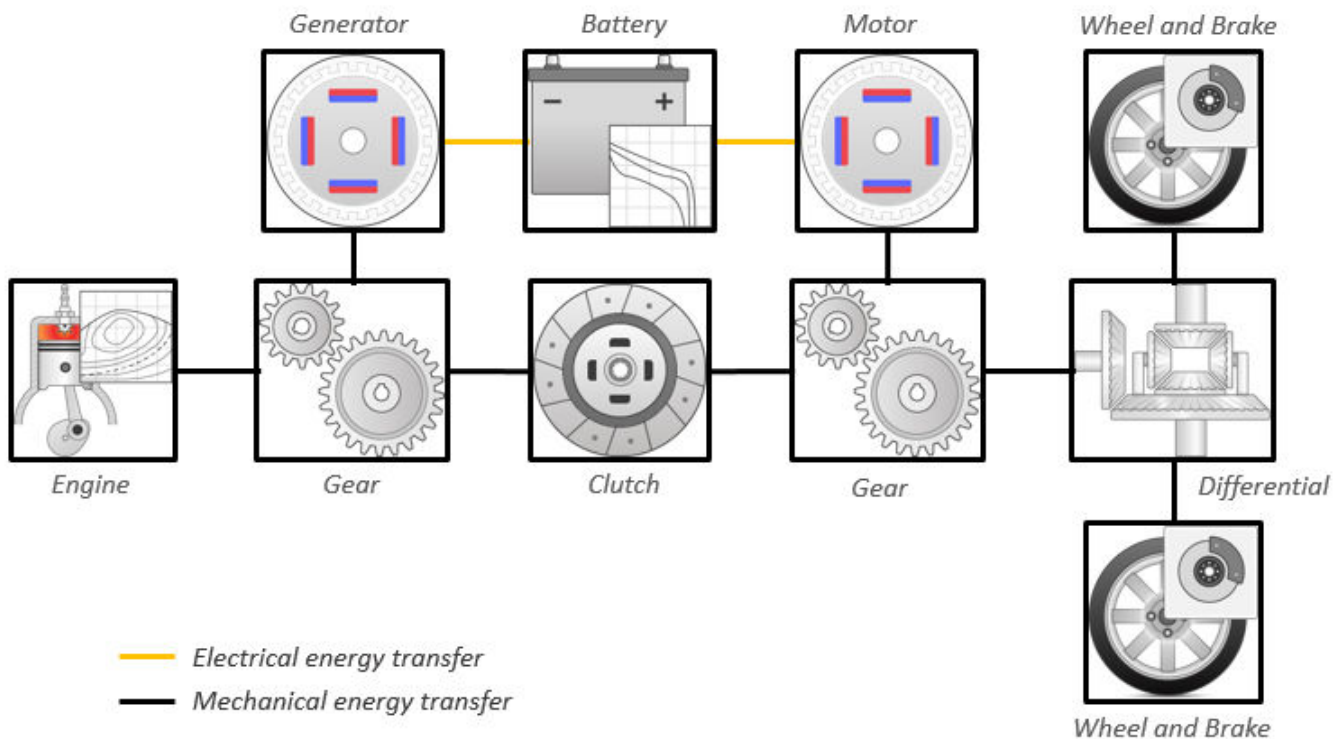
The hybrid electric vehicle reference application represents a full multimode hybrid electric vehicle (HEV) model with an internal combustion engine, transmission, battery, motor, generator, and associated powertrain control algorithms. Use the reference application for powertrain matching analysis and 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.

Reference Application Element	Description	Variants
Analyze Power and Energy	Double-click Analyze Power and Energy to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level. For more information about the live script, see "Analyze Power and Energy" on page 3-107.	NA
Drive Cycle Source block — FTP75 (2474 seconds)	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 or Open Loop variant to generate normalized acceleration and braking commands. <ul style="list-style-type: none"> Longitudinal Driver variant implements a driver model that uses vehicle target and reference velocities. Open Loop variant allows you to configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs. 	✓
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.	

Evaluate and Report Power and Energy

Double-click **Analyze Power and Energy** to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level. For more information about the live script, see “Analyze Power and Energy” on page 3-107.

The script provides:

- An overall energy summary that you can export to an Excel spreadsheet.
- Engine plant, electric plant, and drivetrain plant efficiencies, including an engine histogram of time spent at the different engine plant efficiencies.
- Data logging so that you can use the Simulation Data Inspector to analyze the powertrain efficiency and energy transfer signals.

For more information about the live script, see “Analyze Power and Energy” on page 3-107.

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. The reference application has these variants.

Block Variants			Description
Longitudinal Driver (default)	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	Proportional-integral (PI) control with tracking windup and feed-forward gains.
	Low-pass	LPF	Use an LPF on target velocity error for smoother driving.

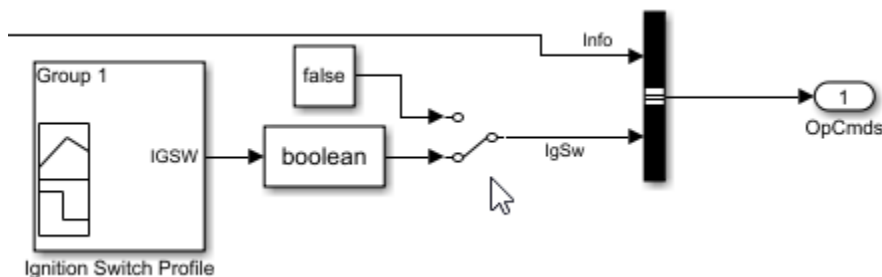
Block Variants		Description	
	filter (LPF)	pass	Do not use a filter on velocity error.
	Shift	Basic	Stateflow chart models reverse, neutral, and drive gear shift scheduling.
		External	Input gear, vehicle state, and velocity feedback generates acceleration and braking commands to track forward and reverse vehicle motion.
		None	No transmission.
		Scheduled	Stateflow chart models reverse, neutral, park, and N-speed gear shift scheduling.
Open Loop		Open-loop control subsystem. In the subsystem, you can configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs.	

To idle the engine at the beginning of a drive cycle and simulate catalyst light-off before moving the vehicle with a pedal command, use the Longitudinal Driver variant. The Longitudinal Driver subsystem includes an ignition switch signal profile, `IgSw`. The engine controller uses the ignition switch signal to start both the engine and a catalyst light-off timer.

The catalyst light-off timer overrides the engine stop-start (ESS) stop function control while the catalyst light-off timer is counting up. During the simulation, after the `IgSw` down-edge time reaches the catalyst light-off time `CatLightOffTime`, normal ESS operation resumes. If there is no torque command before the simulation reaches the `EngStopTime`, the ESS shuts down the engine.

To control ESS and catalyst light-off:

- In the Longitudinal Driver Model subsystem, set the ignition switch profile `IgSw` to 'on'.



- In the engine controller model workspace, set these calibration parameters:
 - `EngStopStartEnable` — Enables ESS. To disable ESS, set the value to false.
 - `CatLightOffTime` — Engine idle time from engine start to catalyst light-off.
 - `EngStopTime` — ESS engine run time after driver model torque request cut-off.

Controllers

The Controller subsystem has a PCM with an HCM and an ECM.

ECM

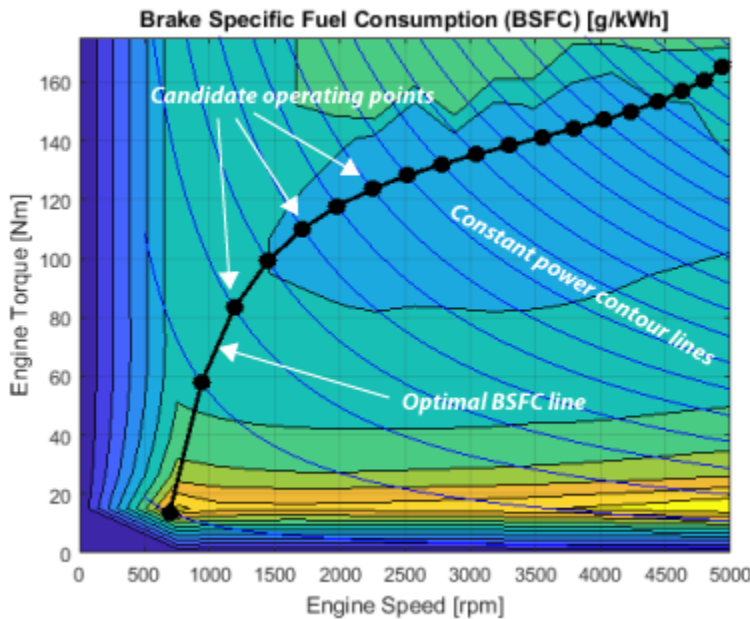
The reference application has these variants for the ECM.

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

HCM

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 subsystem variants.

Drivetrain

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	Longitudinal Wheel - Front 1	<p>For the wheels, you can configure the type of:</p> <ul style="list-style-type: none"> • Brake • Force calculation • Resistance calculation • Vertical motion <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>
	Longitudinal Wheel - Rear 1	

Electric Plant

Electric Plant Subsystem	Variant	Description
Battery	BattHevMm (default)	Configured with electric battery
Generator	GenMapped (default)	Mapped generator
	GenDynamic	Interior permanent magnet synchronous motor (PMSM) with controller

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

Engine

Engine Subsystem	Variant	Description
Engine	SiEngineCore	Dynamic SI Core Engine with turbocharger
	SiEngineCoreNA	Dynamic naturally aspirated SI Core Engine
	SiEngineCoreV	Dynamic SI V Twin-Turbo Single-Intake Engine
	SiEngineCoreVNA	Dynamic SI V Engine
	SiEngineCoreVThr2	Dynamic SI V Twin-Turbo Twin-Intake Engine
	SiMappedEngine (default)	Mapped SI Engine with implicit turbocharger
	SiDLEngine	Deep learning SI engine
	CiEngine	Dynamic CI Core 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

- “HEV Multimode Reference Application” on page 7-3

More About

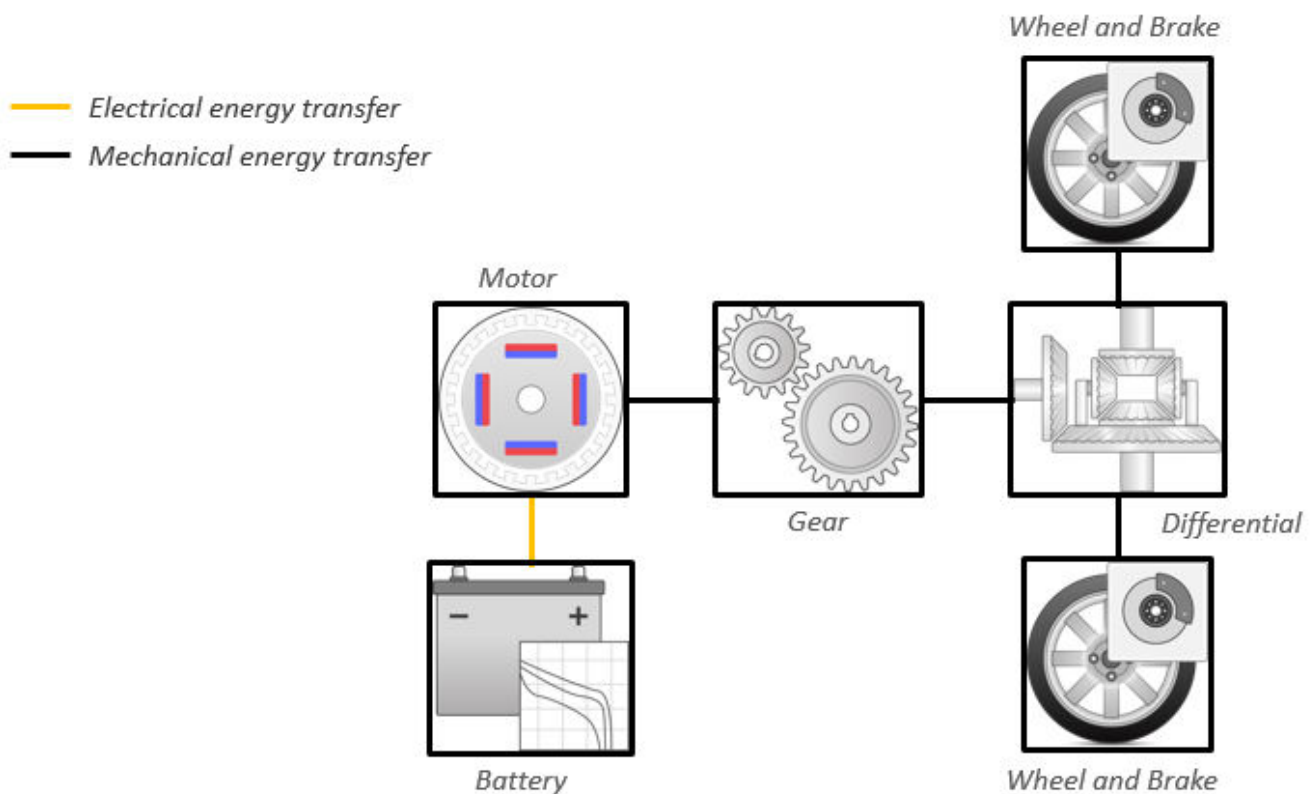
- “Analyze Power and Energy” on page 3-107
- “Hybrid and Electric Vehicle Reference Application Projects” on page 3-3
- “Variant Systems”

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
Analyze Power and Energy	Double-click Analyze Power and Energy to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level. For more information about the live script, see “Analyze Power and Energy” on page 3-107.	NA
Drive Cycle Source block — FTP75 (2474 seconds)	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	<p>Uses the Longitudinal Driver or Open Loop variant to generate normalized acceleration and braking commands.</p> <ul style="list-style-type: none"> • Longitudinal Driver variant implements a driver model that uses vehicle target and reference velocities. • Open Loop variant allows you to configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs. 	✓
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.	✓

Reference Application Element	Description	Variants
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.	

Evaluate and Report Power and Energy

Double-click **Analyze Power and Energy** to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level.

The script provides:

- An overall energy summary that you can export to an Excel spreadsheet.
- Electric plant and drivetrain plant efficiencies.
- Data logging so that you can use the Simulation Data Inspector to analyze the powertrain efficiency and energy transfer signals.

For more information about the live script, see “Analyze Power and Energy” on page 3-107.

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. The reference application has these variants.

Block Variants			Description
Longitudinal Driver (default)	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	LPF	Use an LPF on target velocity error for smoother driving.

Block Variants		Description	
	filter (LPF)	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.
Open Loop			Open-loop control subsystem. In the subsystem, you can configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs.

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

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	Longitudinal Wheel - Front 1	<p>For the wheels, you can configure the type of:</p> <ul style="list-style-type: none"> • Brake • Force calculation • Resistance calculation • Vertical motion <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>
	Longitudinal Wheel - Rear 1	

Electric Plant

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

- “EV Reference Application” on page 7-10

More About

- “Analyze Power and Energy” on page 3-107
- “Hybrid and Electric Vehicle Reference Application Projects” on page 3-3
- “Variant Systems”

Explore the Hybrid Electric Vehicle Input Power-Split Reference Application

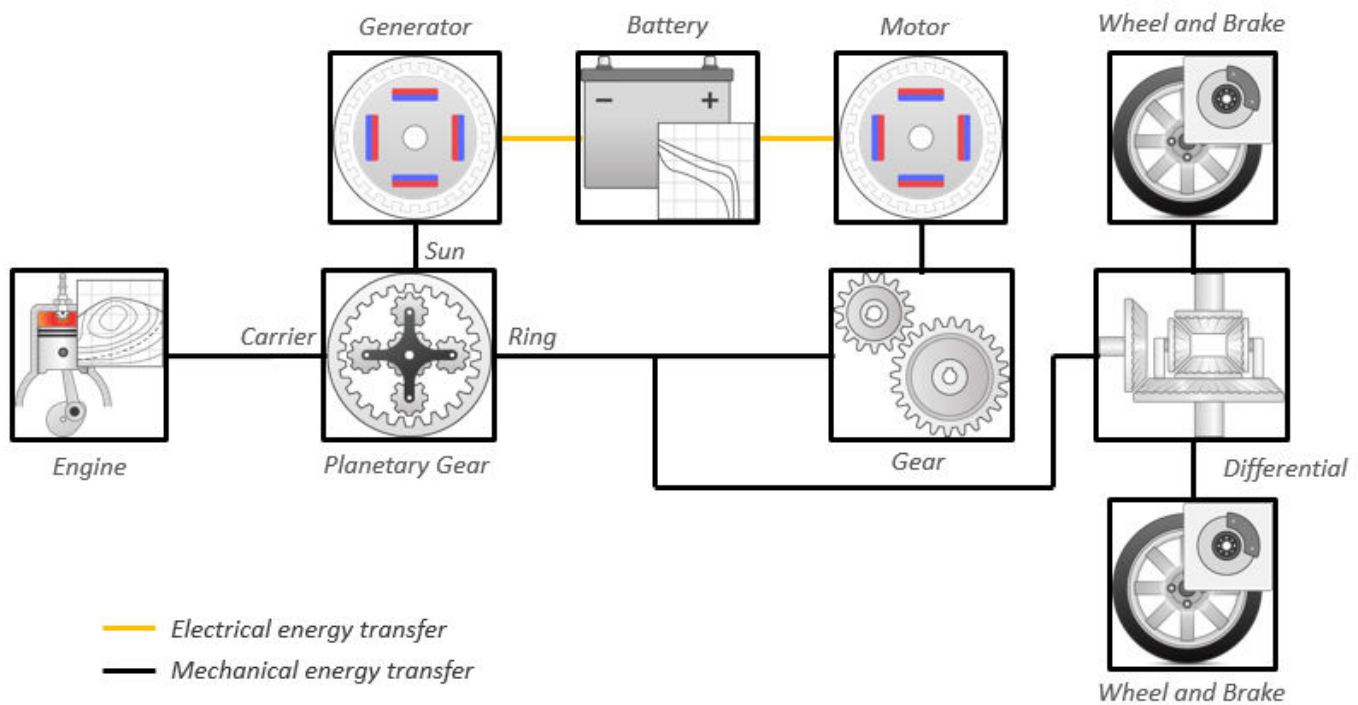
The hybrid electric vehicle (HEV) input power-split reference application represents a full HEV model with an internal combustion engine, transmission, battery, motor, generator, and associated powertrain control algorithms. Use the HEV input power-split reference application for HIL testing, tradeoff 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
Analyze Power and Energy	Double-click Analyze Power and Energy to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level. For more information about the live script, see “Analyze Power and Energy” on page 3-107.	NA
Drive Cycle Source block — FTP75 (2474 seconds)	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 or Open Loop variant to generate normalized acceleration and braking commands. <ul style="list-style-type: none"> • Longitudinal Driver variant implements a driver model that uses vehicle target and reference velocities. • Open Loop variant allows you to configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs. 	✓
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.	✓

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.	

Evaluate and Report Power and Energy

Double-click **Analyze Power and Energy** to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level. For more information about the live script, see “Analyze Power and Energy” on page 3-107.

The script provides:

- An overall energy summary that you can export to an Excel spreadsheet.
- Engine plant, electric plant, and drivetrain plant efficiencies, including an engine histogram of time spent at the different engine plant efficiencies.
- Data logging so that you can use the Simulation Data Inspector to analyze the powertrain efficiency and energy transfer signals.

For more information about the live script, see “Analyze Power and Energy” on page 3-107.

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. The reference application has these variants.

Block Variants			Description
Longitudinal Driver (default)	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	Proportional-integral (PI) control with tracking windup and feed-forward gains.
	Low-pass	LPF	Use an LPF on target velocity error for smoother driving.

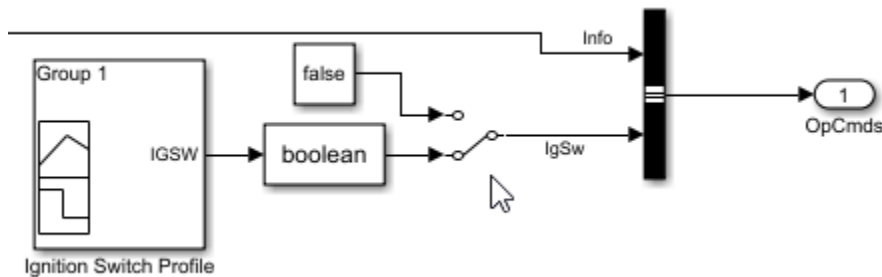
Block Variants		Description	
filter (LPF)	pass	Do not use a filter on velocity error.	
	Shift	Basic	Stateflow chart models reverse, neutral, and drive gear shift scheduling.
		External	Input gear, vehicle state, and velocity feedback generates acceleration and braking commands to track forward and reverse vehicle motion.
		None	No transmission.
		Scheduled	Stateflow chart models reverse, neutral, park, and N-speed gear shift scheduling.
Open Loop		Open-loop control subsystem. In the subsystem, you can configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs.	

To idle the engine at the beginning of a drive cycle and simulate catalyst light-off before moving the vehicle with a pedal command, use the Longitudinal Driver variant. The Longitudinal Driver subsystem includes an ignition switch signal profile, `IgSw`. The engine controller uses the ignition switch signal to start both the engine and a catalyst light-off timer.

The catalyst light-off timer overrides the engine stop-start (ESS) stop function control while the catalyst light-off timer is counting up. During the simulation, after the `IgSw` down-edge time reaches the catalyst light-off time `CatLightOffTime`, normal ESS operation resumes. If there is no torque command before the simulation reaches the `EngStopTime`, the ESS shuts down the engine.

To control ESS and catalyst light-off:

- In the Longitudinal Driver Model subsystem, set the ignition switch profile `IgSw` to 'on'.



- In the engine controller model workspace, set these calibration parameters:
 - `EngStopStartEnable` — Enables ESS. To disable ESS, set the value to false.
 - `CatLightOffTime` — Engine idle time from engine start to catalyst light-off.
 - `EngStopTime` — ESS engine run time after driver model torque request cut-off.

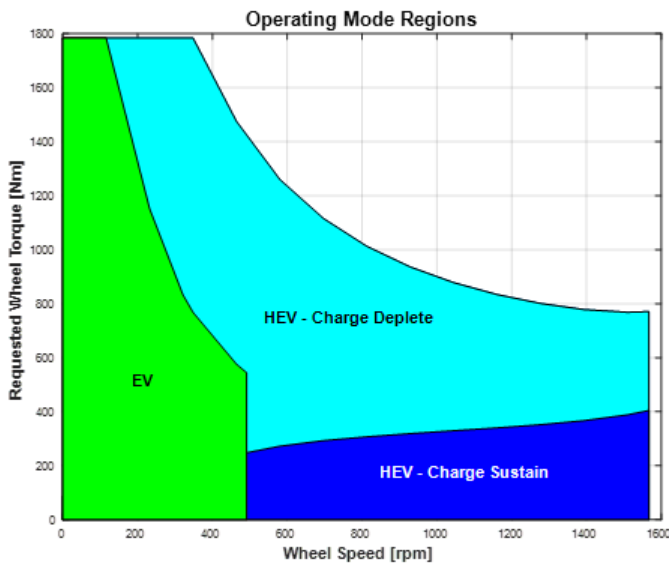
Controllers

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
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.

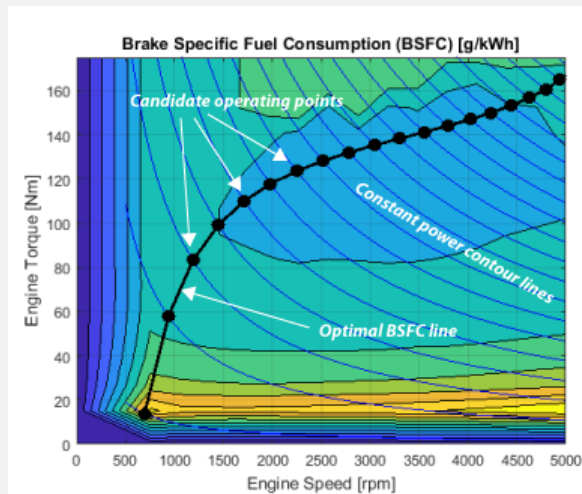


Mode	Description
EV	Traction motor provides the wheel torque request.

Mode	Description
HEV - Charge Sustaining (Low Power)	<ul style="list-style-type: none"> Engine provides the wheel torque request. 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. 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. 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.
HEV - Charge Depleting (High Power)	<ul style="list-style-type: none"> Engine provides the wheel power request up to its maximum output. 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.
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"> Decision logic determines the engine operation modes (off, start, run). In engine run mode, lookup tables determine the engine torque and engine speed that optimizes the break-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.



Control	Description
Generator	<ul style="list-style-type: none"> 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. A rule-based power management algorithm calculates a generator torque that does not exceed the dynamic power limits.
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 subsystem variants.

Drivetrain

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	
Gearbox	Ideal Fixed Gear Transmission	Configure gearbox efficiency with a constant (default) or 3D lookup table.
Vehicle	Vehicle Body 3 DOF Longitudinal	Configured for 3 degrees of freedom
Wheels and Brakes	Longitudinal Wheel - Front 1	<p>For the wheels, you can configure the type of:</p> <ul style="list-style-type: none"> Brake Force calculation Resistance calculation Vertical motion <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</p>

Drivetrain Subsystem	Variant	Description
	Longitudinal Wheel - Rear 1	blocks to model each wheel increases model complexity and computational cost.

Electric Plant

Electric Plant Subsystem	Variant	Description
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

Engine

Engine Subsystem	Variant	Description
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

- “HEV Input Power-Split Reference Application” on page 7-4

More About

- “Analyze Power and Energy” on page 3-107
- “Hybrid and Electric Vehicle Reference Application Projects” on page 3-3
- “Variant Systems”

Explore the Hybrid Electric Vehicle P0 Reference Application

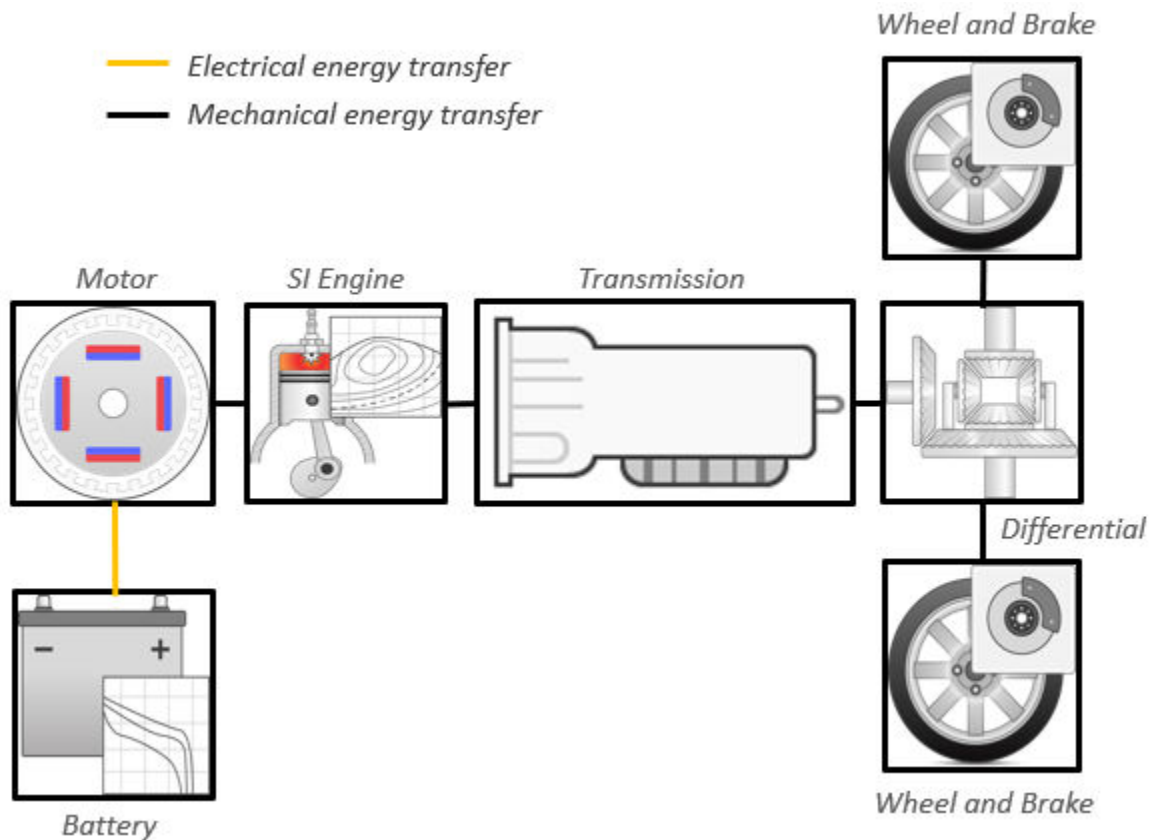
The hybrid electric vehicle (HEV) P0 reference application represents a full HEV model with an internal combustion engine, transmission, battery, motor, and associated powertrain control algorithms. Use the reference application for hardware-in-the-loop (HIL) testing, tradeoff analysis, and control parameter optimization of a HEV P0 hybrid. To create and open a working copy of the reference application project, enter

```
autoblkHevP0Start
```

By default, the HEV P0 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
Analyze Power and Energy	Double-click Analyze Power and Energy to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level. For more information about the live script, see “Analyze Power and Energy” on page 3-107.	NA
Drive Cycle Source block – FTP75 (2474 seconds)	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	<p>Uses the Longitudinal Driver or Open Loop variant to generate normalized acceleration and braking commands.</p> <ul style="list-style-type: none"> • Longitudinal Driver variant implements a driver model that uses vehicle target and reference velocities. • Open Loop variant allows you to configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs. 	✓
Controllers subsystem	Implements a powertrain control module (PCM) containing a P0 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.	

Evaluate and Report Power and Energy

Double-click **Analyze Power and Energy** to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level. For more information about the live script, see “Analyze Power and Energy” on page 3-107.

The script provides:

- An overall energy summary that you can export to an Excel spreadsheet.
- Engine plant, electric plant, and drivetrain plant efficiencies, including an engine histogram of time spent at the different engine plant efficiencies.
- Data logging so that you can use the Simulation Data Inspector to analyze the powertrain efficiency and energy transfer signals.

For more information about the live script, see “Analyze Power and Energy” on page 3-107.

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. The reference application has these variants.

Block Variants			Description
Longitudinal Driver (default)	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	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	No transmission.

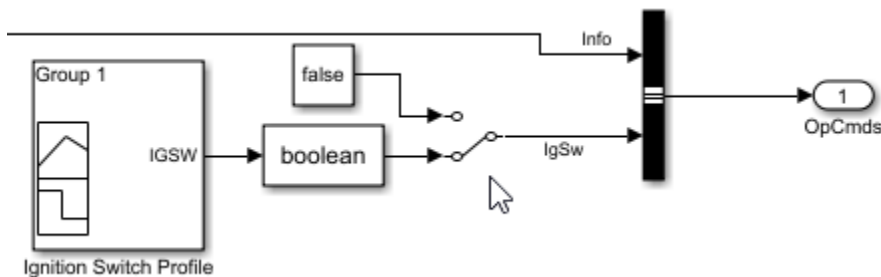
Block Variants		Description
	Scheduled	Stateflow chart models reverse, neutral, park, and N-speed gear shift scheduling.
Open Loop		Open-loop control subsystem. In the subsystem, you can configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs.

To idle the engine at the beginning of a drive cycle and simulate catalyst light-off before moving the vehicle with a pedal command, use the Longitudinal Driver variant. The Longitudinal Driver subsystem includes an ignition switch signal profile, IgSw. The engine controller uses the ignition switch signal to start both the engine and a catalyst light-off timer.

The catalyst light-off timer overrides the engine stop-start (ESS) stop function control while the catalyst light-off timer is counting up. During the simulation, after the IgSw down-edge time reaches the catalyst light-off time CatLightOffTime, normal ESS operation resumes. If there is no torque command before the simulation reaches the EngStopTime, the ESS shuts down the engine.

To control ESS and catalyst light-off:

- In the Longitudinal Driver Model subsystem, set the ignition switch profile IgSw to 'on'.



- In the engine controller model workspace, set these calibration parameters:
 - EngStopStartEnable – Enables ESS. To disable ESS, set the value to false.
 - CatLightOffTime – Engine idle time from engine start to catalyst light-off.
 - EngStopTime – ESS engine run time after driver model torque request cut-off.

Controllers

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

Controller	Variant	Description
ECM	SiEngineController (default)	Implements the SI Controller
	CiEngineController	Implements the CI Controller
HCM	ECMS	Implements the Equivalent Consumption Minimization Strategy
TCM	TransmissionController	Implements the transmission controller

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 subsystem variants.

Drivetrain

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	
Torque Converter Automatic Transmission	Ideal Fixed Gear Transmission	Configure locked and unlocked transmission efficiency with either a 1D or 4D (default) lookup table.
	Torque Converter	Configure for external, internal (default), or no lockup.
Vehicle	Vehicle Body 1 DOF Longitudinal	Configured for 1 degrees of freedom
Wheels and Brakes	Longitudinal Wheel - Front 1	<p>For the wheels, you can configure the type of:</p> <ul style="list-style-type: none"> • Brake • Force calculation • Resistance calculation • Vertical motion <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>
	Longitudinal Wheel - Rear 1	

Simscape Drivetrain. Another way to customize the drivetrain is to select a Simscape variant. This variant incorporates physical connections to provide a flexible way to assemble components.

Use the button in the top level of the reference application to toggle between Simscape and Powertrain Blockset variants of the drivetrain subsystem.

The reference application sets the appropriate solvers to optimize performance for each engine and drivetrain combination. Select the engine variant first, then select the drivetrain using the toggle button. If you select the drivetrain before changing the engine, you may encounter a solver error.

Electric Plant

Electric Plant Subsystem	Variant	Description
Battery	BattHevP0	Configured with Lithium Ion battery
Electric Machine	MotMapped	Mapped Motor with implicit controller

Engine

Engine Subsystem	Variant	Description
Engine	SiEngineCore	Dynamic SI Core Engine with turbocharger
	SiMappedEngine (default)	Mapped SI Engine with implicit turbocharger
	SiEngineCoreNA	Dynamic naturally aspirated SI Core Engine

Limitations

MathWorks used the SI Core Engine and SI Controller to calibrate the hybrid control module (HCM). If you use the CI Core Engine and CI Controller variants, the simulation may error because the HCM does not use calibrated results.

Acknowledgment

MathWorks would like to acknowledge the contribution of Dr. Simona Onori to the ECMS optimal control algorithm implemented in this reference application. Dr. Onori is a Professor of Energy Resources Engineering at Stanford University. Her research interests include electrochemical modeling, estimation and optimization of energy storage devices for automotive and grid-level applications, hybrid and electric vehicles modeling and control, PDE modeling, and model-order reduction and estimation of emission mitigation systems. She is a senior member of IEEE®.

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] Onori, S., Serrao, L., and Rizzoni, G., *Hybrid Electric Vehicles Energy Management Systems*. New York: Springer, 2016.

See Also

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

Related Examples

- “HEV P0 Reference Application” on page 7-5

More About

- “Analyze Power and Energy” on page 3-107
- “Hybrid and Electric Vehicle Reference Application Projects” on page 3-3
- “Variant Systems”

Explore the Hybrid Electric Vehicle P1 Reference Application

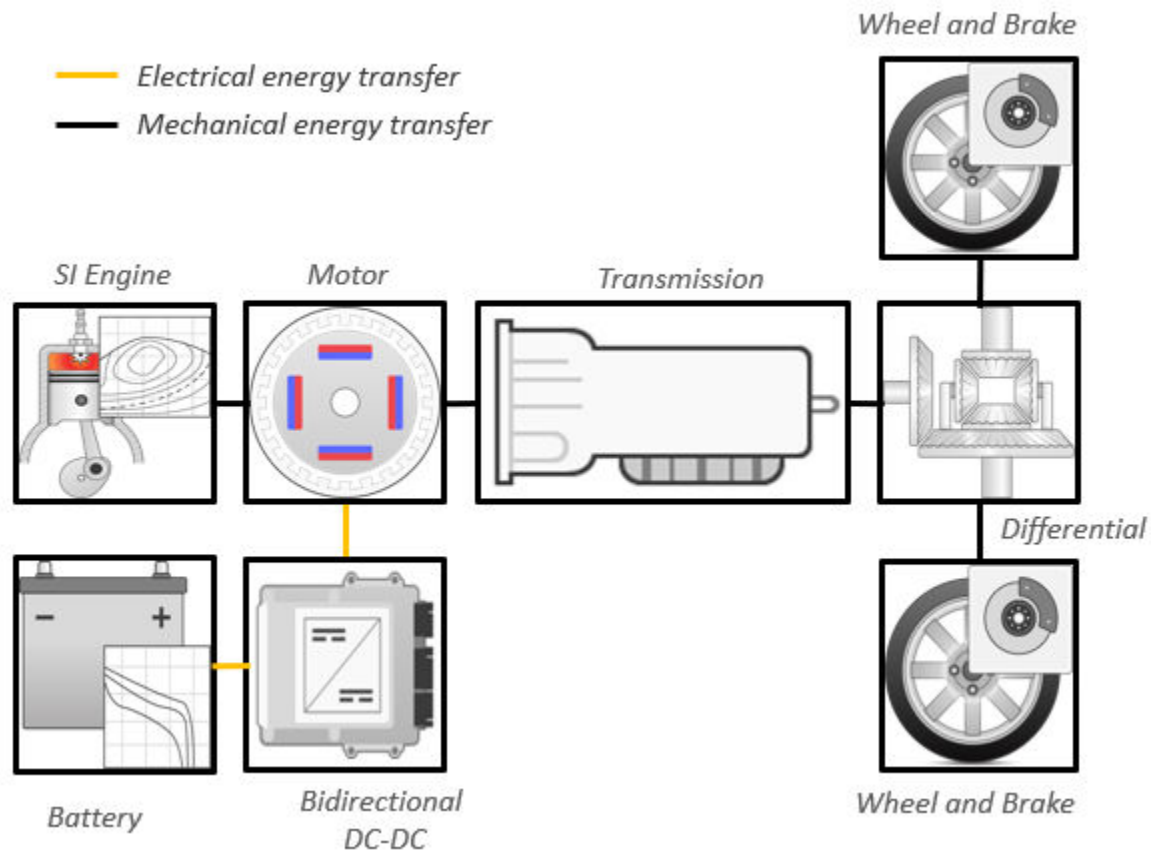
The hybrid electric vehicle (HEV) P1 reference application represents a full HEV model with an internal combustion engine, transmission, battery, motor, and associated powertrain control algorithms. Use the reference application for hardware-in-the-loop (HIL) testing, tradeoff analysis, and control parameter optimization of a HEV P1 hybrid. To create and open a working copy of the reference application project, enter

autoblkHevP1Start

By default, the HEV P1 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
Analyze Power and Energy	Double-click Analyze Power and Energy to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level. For more information about the live script, see “Analyze Power and Energy” on page 3-107.	NA
Drive Cycle Source block — FTP75 (2474 seconds)	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	<p>Uses the Longitudinal Driver or Open Loop variant to generate normalized acceleration and braking commands.</p> <ul style="list-style-type: none"> • Longitudinal Driver variant implements a driver model that uses vehicle target and reference velocities. • Open Loop variant allows you to configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs. 	✓
Controllers subsystem	Implements a powertrain control module (PCM) containing a P1 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.	

Evaluate and Report Power and Energy

Double-click **Analyze Power and Energy** to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level. For more information about the live script, see “Analyze Power and Energy” on page 3-107.

The script provides:

- An overall energy summary that you can export to an Excel spreadsheet.
- Engine plant, electric plant, and drivetrain plant efficiencies, including an engine histogram of time spent at the different engine plant efficiencies.
- Data logging so that you can use the Simulation Data Inspector to analyze the powertrain efficiency and energy transfer signals.

For more information about the live script, see “Analyze Power and Energy” on page 3-107.

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. The reference application has these variants.

Block Variants			Description
Longitudinal Driver (default)	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	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	No transmission.

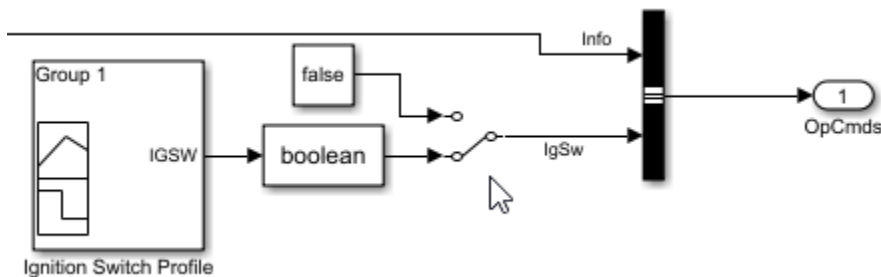
Block Variants		Description
	Scheduled	Stateflow chart models reverse, neutral, park, and N-speed gear shift scheduling.
Open Loop		Open-loop control subsystem. In the subsystem, you can configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs.

To idle the engine at the beginning of a drive cycle and simulate catalyst light-off before moving the vehicle with a pedal command, use the Longitudinal Driver variant. The Longitudinal Driver subsystem includes an ignition switch signal profile, `IgSw`. The engine controller uses the ignition switch signal to start both the engine and a catalyst light-off timer.

The catalyst light-off timer overrides the engine stop-start (ESS) stop function control while the catalyst light-off timer is counting up. During the simulation, after the `IgSw` down-edge time reaches the catalyst light-off time `CatLightOffTime`, normal ESS operation resumes. If there is no torque command before the simulation reaches the `EngStopTime`, the ESS shuts down the engine.

To control ESS and catalyst light-off:

- In the Longitudinal Driver Model subsystem, set the ignition switch profile `IgSw` to 'on'.



- In the engine controller model workspace, set these calibration parameters:
 - `EngStopStartEnable` — Enables ESS. To disable ESS, set the value to false.
 - `CatLightOffTime` — Engine idle time from engine start to catalyst light-off.
 - `EngStopTime` — ESS engine run time after driver model torque request cut-off.

Controllers

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

Controller	Variant	Description
ECM	<code>SiEngineController</code> (default)	Implements the SI Controller
	<code>CiEngineController</code>	Implements the CI Controller
HCM	ECMS	Implements the Equivalent Consumption Minimization Strategy
TCM	<code>TransmissionController</code>	Implements the transmission controller

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 subsystem variants.

Drivetrain

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	
Torque Converter Automatic Transmission	Ideal Fixed Gear Transmission	Configure locked and unlocked transmission efficiency with either a 1D or 4D (default) lookup table.
	Torque Converter	Configure for external, internal (default), or no lockup.
Vehicle	Vehicle Body 1 DOF Longitudinal	Configured for 1 degrees of freedom
Wheels and Brakes	Longitudinal Wheel - Front 1	<p>For the wheels, you can configure the type of:</p> <ul style="list-style-type: none"> • Brake • Force calculation • Resistance calculation • Vertical motion <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>
	Longitudinal Wheel - Rear 1	

Simscape Drivetrain. Another way to customize the drivetrain is to select a Simscape variant. This variant incorporates physical connections to provide a flexible way to assemble components.

Use the button in the top level of the reference application to toggle between Simscape and Powertrain Blockset variants of the drivetrain subsystem.

The reference application sets the appropriate solvers to optimize performance for each engine and drivetrain combination. Select the engine variant first, then select the drivetrain using the toggle button. If you select the drivetrain before changing the engine, you may encounter a solver error.

Electric Plant

Electric Plant Subsystem	Variant	Description
Battery	BattHevP1	Configured with Lithium Ion battery
Electric Machine	MotMapped	Mapped Motor with implicit controller

Engine

Engine Subsystem	Variant	Description
Engine	SiEngineCore	Dynamic SI Core Engine with turbocharger
	SiMappedEngine (default)	Mapped SI Engine with implicit turbocharger
	SiEngineCoreNA	Dynamic naturally aspirated SI Core Engine

Limitations

MathWorks used the SI Core Engine and SI Controller to calibrate the hybrid control module (HCM). If you use the CI Core Engine and CI Controller variants, the simulation may error because the HCM does not use calibrated results.

Acknowledgment

MathWorks would like to acknowledge the contribution of Dr. Simona Onori to the ECMS optimal control algorithm implemented in this reference application. Dr. Onori is a Professor of Energy Resources Engineering at Stanford University. Her research interests include electrochemical modeling, estimation and optimization of energy storage devices for automotive and grid-level applications, hybrid and electric vehicles modeling and control, PDE modeling, and model-order reduction and estimation of emission mitigation systems. She is a senior member of IEEE.

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] Onori, S., Serrao, L., and Rizzoni, G., *Hybrid Electric Vehicles Energy Management Systems*. New York: Springer, 2016.

See Also

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

Related Examples

- “HEV P1 Reference Application” on page 7-6

More About

- “Analyze Power and Energy” on page 3-107
- “Hybrid and Electric Vehicle Reference Application Projects” on page 3-3
- “Variant Systems”

Explore the Hybrid Electric Vehicle P2 Reference Application

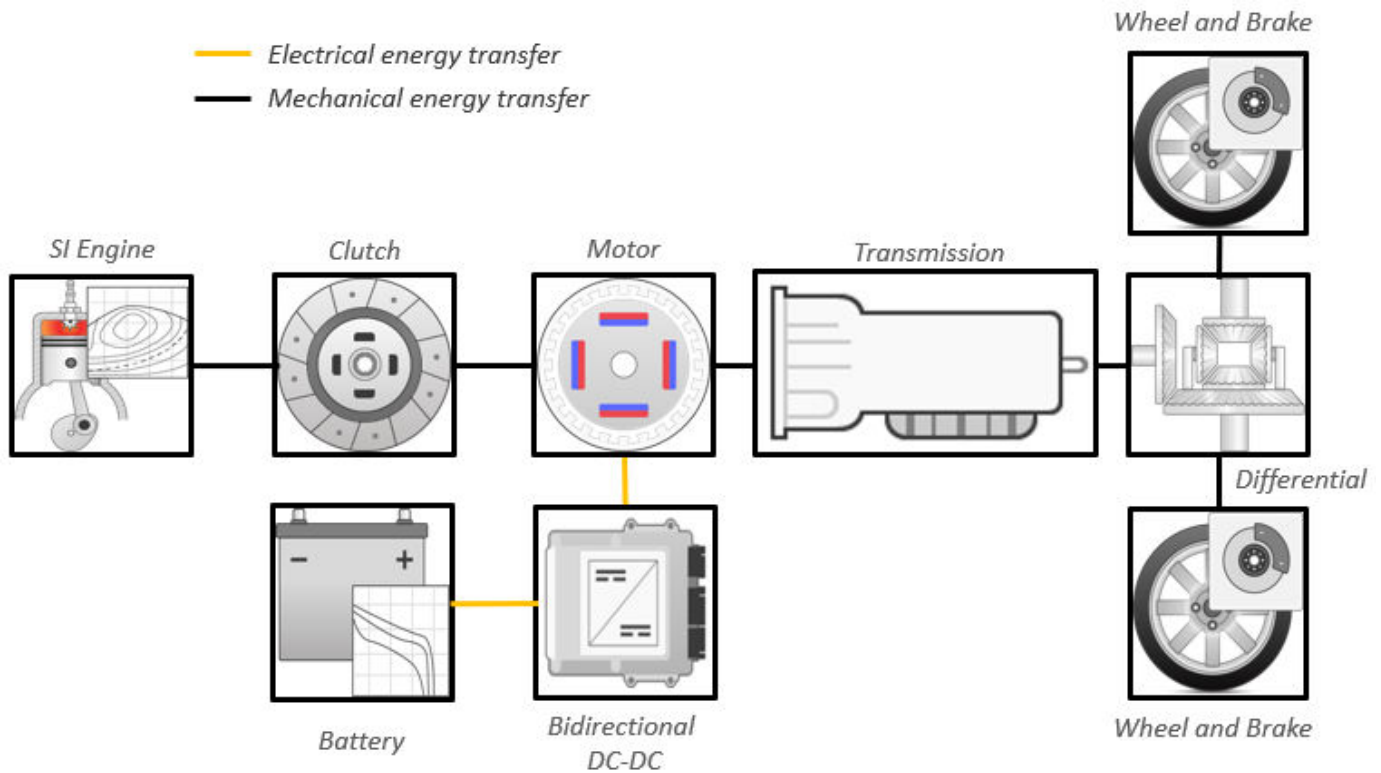
The hybrid electric vehicle (HEV) P2 reference application represents a full HEV model with an internal combustion engine, transmission, battery, motor, and associated powertrain control algorithms. Use the reference application for hardware-in-the-loop (HIL) testing, tradeoff analysis, and control parameter 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
Analyze Power and Energy	Double-click Analyze Power and Energy to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level. For more information about the live script, see “Analyze Power and Energy” on page 3-107.	NA
Drive Cycle Source block – FTP75 (2474 seconds)	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	<p>Uses the Longitudinal Driver or Open Loop variant to generate normalized acceleration and braking commands.</p> <ul style="list-style-type: none"> • Longitudinal Driver variant implements a driver model that uses vehicle target and reference velocities. • Open Loop variant allows you to configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs. 	✓
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.	

Evaluate and Report Power and Energy

Double-click **Analyze Power and Energy** to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level. For more information about the live script, see “Analyze Power and Energy” on page 3-107.

The script provides:

- An overall energy summary that you can export to an Excel spreadsheet.
- Engine plant, electric plant, and drivetrain plant efficiencies, including an engine histogram of time spent at the different engine plant efficiencies.
- Data logging so that you can use the Simulation Data Inspector to analyze the powertrain efficiency and energy transfer signals.

For more information about the live script, see “Analyze Power and Energy” on page 3-107.

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. The reference application has these variants.

Block Variants			Description
Longitudinal Driver (default)	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	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	No transmission.

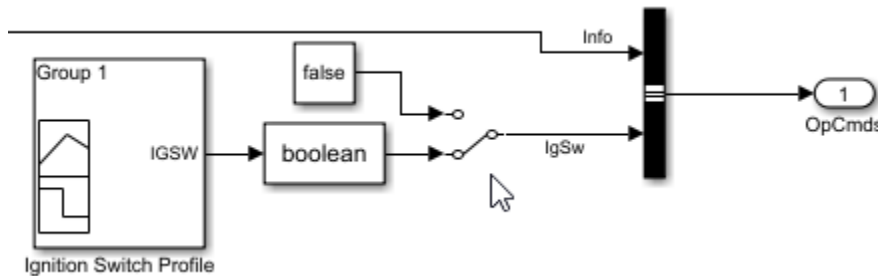
Block Variants		Description
	Scheduled	Stateflow chart models reverse, neutral, park, and N-speed gear shift scheduling.
Open Loop		Open-loop control subsystem. In the subsystem, you can configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs.

To idle the engine at the beginning of a drive cycle and simulate catalyst light-off before moving the vehicle with a pedal command, use the Longitudinal Driver variant. The Longitudinal Driver subsystem includes an ignition switch signal profile, `IgSw`. The engine controller uses the ignition switch signal to start both the engine and a catalyst light-off timer.

The catalyst light-off timer overrides the engine stop-start (ESS) stop function control while the catalyst light-off timer is counting up. During the simulation, after the `IgSw` down-edge time reaches the catalyst light-off time `CatLightOffTime`, normal ESS operation resumes. If there is no torque command before the simulation reaches the `EngStopTime`, the ESS shuts down the engine.

To control ESS and catalyst light-off:

- In the Longitudinal Driver Model subsystem, set the ignition switch profile `IgSw` to 'on'.



- In the engine controller model workspace, set these calibration parameters:
 - `EngStopStartEnable` — Enables ESS. To disable ESS, set the value to false.
 - `CatLightOffTime` — Engine idle time from engine start to catalyst light-off.
 - `EngStopTime` — ESS engine run time after driver model torque request cut-off.

Controllers

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

Controller	Variant	Description
ECM	<code>SiEngineController</code> (default)	Implements the SI Controller
	<code>CiEngineController</code>	Implements the CI Controller
TCM	<code>TransmissionController</code>	Implements the transmission controller

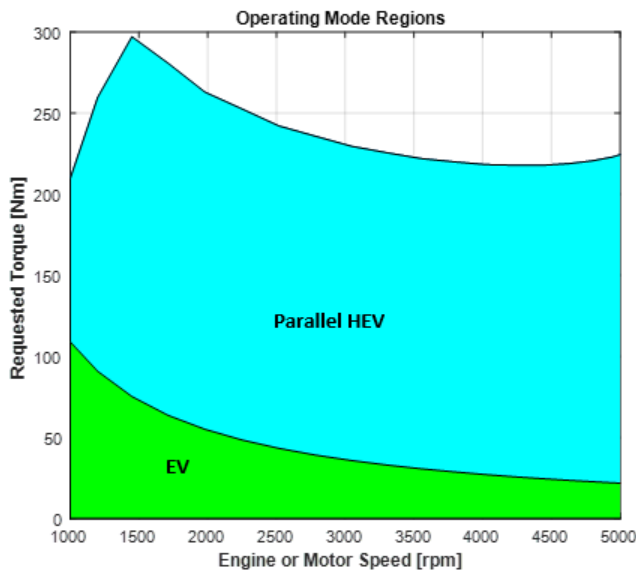
Controller	Variant		Description
HCM	Optimal Control (default)	Energy Management System	Implements the Equivalent Consumption Minimization Strategy
	Rule-Based Control	P2 Supervisory Control	Implements a dynamic supervisory controller that determines the engine torque, motor torque, starter, clutch, and brake pressure commands.
		Regen Braking Control	Implements a parallel or series regenerative braking controller during rule-based control.

Rule-Based Control

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

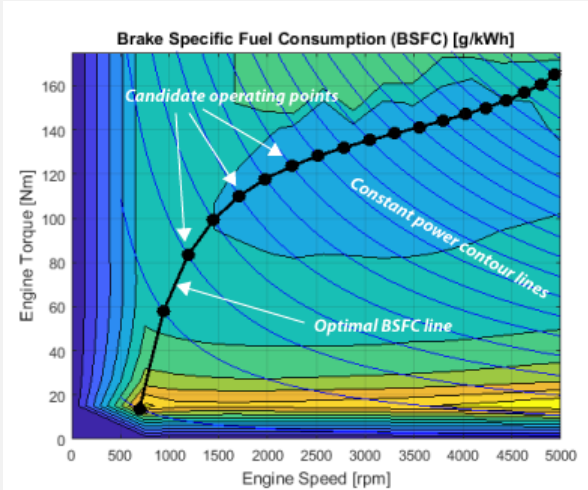
- Converts the driver accelerator pedal signal to a torque request. The algorithm uses the optimal engine torque and maximum motor torque curves to calculate the total powertrain torque.
- Converts the driver brake pedal signal to a brake pressure request. The algorithm multiplies the brake pedal signal 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 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 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"> Decision logic determines the engine operation modes (off, start, on). To start the engine in engine start (stationary) mode, the 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. 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. 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.  <ul style="list-style-type: none"> 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.
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

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	
Torque Converter Automatic Transmission	Ideal Fixed Gear Transmission	Configure locked and unlocked transmission efficiency with either a 1D or 4D (default) lookup table.
	Torque Converter	Configure for external, internal (default), or no lockup.
Vehicle	Vehicle Body 1 DOF Longitudinal	Configured for 1 degrees of freedom
Wheels and Brakes	Longitudinal Wheel - Front 1	<p>For the wheels, you can configure the type of:</p> <ul style="list-style-type: none"> • Brake • Force calculation • Resistance calculation • Vertical motion <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>
	Longitudinal Wheel - Rear 1	

Simscape Drivetrain. Another way to customize the drivetrain is to select a Simscape variant. This variant incorporates physical connections to provide a flexible way to assemble components.

Use the button in the top level of the reference application to toggle between Simscape and Powertrain Blockset variants of the drivetrain subsystem.

The reference application sets the appropriate solvers to optimize performance for each engine and drivetrain combination. Select the engine variant first, then select the drivetrain using the toggle button. If you select the drivetrain before changing the engine, you may encounter a solver error.

Electric Plant

Electric Plant Subsystem	Variant	Description
Battery	BattHevP2	Configured with Lithium Ion battery and DC-DC converter
Low Voltage Starting System	StarterSystemP2	Configured with a low voltage starting system
Motor	MotMapped (default)	Mapped Motor with implicit controller
	MotDynamic	Interior permanent magnet synchronous motor (PMSM) with controller

Engine

Engine Subsystem	Variant	Description
Engine	SiEngineCore	Dynamic SI Core Engine with turbocharger
	SiMappedEngine (default)	Mapped SI Engine with implicit turbocharger
	SiEngineCoreNA	Dynamic naturally aspirated SI Core Engine

Limitations

MathWorks used the SI Core Engine and SI Controller to calibrate the hybrid control module (HCM). If you use the CI Core Engine and CI Controller variants, the simulation may error because the HCM does not use calibrated results.

Acknowledgment

MathWorks would like to acknowledge the contribution of Dr. Simona Onori to the ECMS optimal control algorithm implemented in this reference application. Dr. Onori is a Professor of Energy Resources Engineering at Stanford University. Her research interests include electrochemical modeling, estimation and optimization of energy storage devices for automotive and grid-level applications, hybrid and electric vehicles modeling and control, PDE modeling, and model-order reduction and estimation of emission mitigation systems. She is a senior member of IEEE.

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] Onori, S., Serrao, L., and Rizzoni, G., *Hybrid Electric Vehicles Energy Management Systems*. New York: Springer, 2016.

See Also

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

Related Examples

- “HEV P2 Reference Application” on page 7-7

More About

- “Analyze Power and Energy” on page 3-107
- “Hybrid and Electric Vehicle Reference Application Projects” on page 3-3
- “Variant Systems”

Explore the Hybrid Electric Vehicle P3 Reference Application

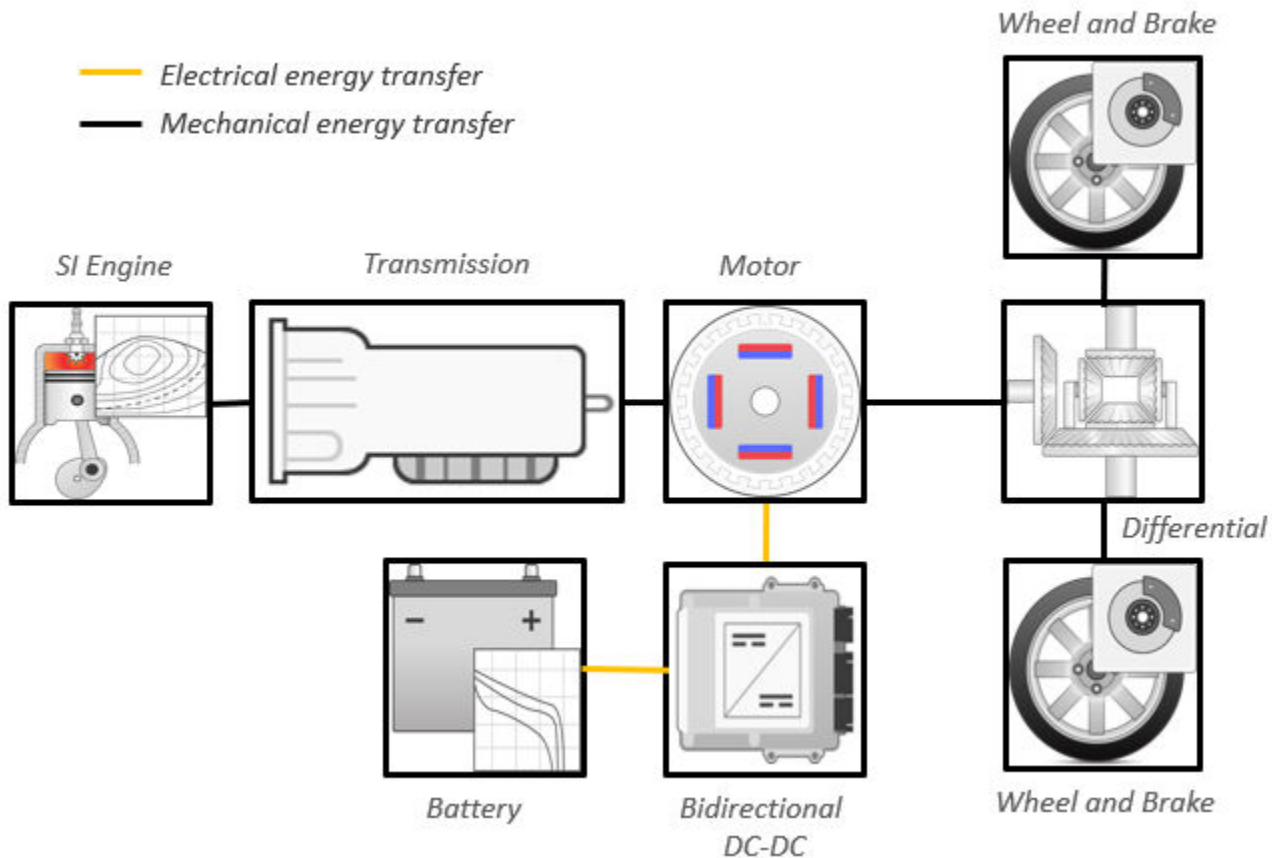
The hybrid electric vehicle (HEV) P3 reference application represents a full HEV model with an internal combustion engine, transmission, battery, motor, and associated powertrain control algorithms. Use the reference application for hardware-in-the-loop (HIL) testing, tradeoff analysis, and control parameter optimization of a HEV P3 hybrid. To create and open a working copy of the reference application project, enter

```
autoblkHevP3Start
```

By default, the HEV P3 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
Analyze Power and Energy	Double-click Analyze Power and Energy to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level. For more information about the live script, see “Analyze Power and Energy” on page 3-107.	NA
Drive Cycle Source block — FTP75 (2474 seconds)	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	<p>Uses the Longitudinal Driver or Open Loop variant to generate normalized acceleration and braking commands.</p> <ul style="list-style-type: none"> • Longitudinal Driver variant implements a driver model that uses vehicle target and reference velocities. • Open Loop variant allows you to configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs. 	✓
Controllers subsystem	Implements a powertrain control module (PCM) containing a P3 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.	

Evaluate and Report Power and Energy

Double-click **Analyze Power and Energy** to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level. For more information about the live script, see “Analyze Power and Energy” on page 3-107.

The script provides:

- An overall energy summary that you can export to an Excel spreadsheet.
- Engine plant, electric plant, and drivetrain plant efficiencies, including an engine histogram of time spent at the different engine plant efficiencies.
- Data logging so that you can use the Simulation Data Inspector to analyze the powertrain efficiency and energy transfer signals.

For more information about the live script, see “Analyze Power and Energy” on page 3-107.

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. The reference application has these variants.

Block Variants			Description
Longitudinal Driver (default)	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	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	No transmission.

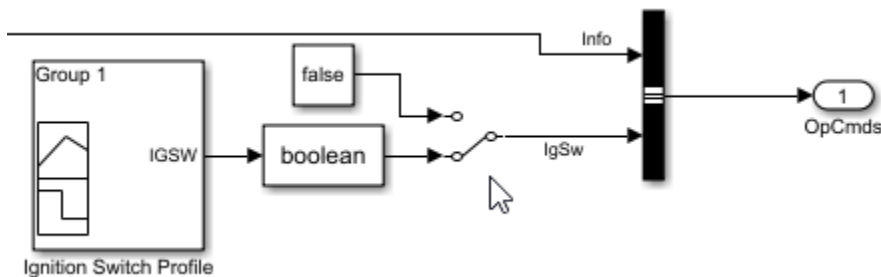
Block Variants		Description
	Scheduled	Stateflow chart models reverse, neutral, park, and N-speed gear shift scheduling.
Open Loop		Open-loop control subsystem. In the subsystem, you can configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs.

To idle the engine at the beginning of a drive cycle and simulate catalyst light-off before moving the vehicle with a pedal command, use the Longitudinal Driver variant. The Longitudinal Driver subsystem includes an ignition switch signal profile, `IgSw`. The engine controller uses the ignition switch signal to start both the engine and a catalyst light-off timer.

The catalyst light-off timer overrides the engine stop-start (ESS) stop function control while the catalyst light-off timer is counting up. During the simulation, after the `IgSw` down-edge time reaches the catalyst light-off time `CatLightOffTime`, normal ESS operation resumes. If there is no torque command before the simulation reaches the `EngStopTime`, the ESS shuts down the engine.

To control ESS and catalyst light-off:

- In the Longitudinal Driver Model subsystem, set the ignition switch profile `IgSw` to 'on'.



- In the engine controller model workspace, set these calibration parameters:
 - `EngStopStartEnable` — Enables ESS. To disable ESS, set the value to false.
 - `CatLightOffTime` — Engine idle time from engine start to catalyst light-off.
 - `EngStopTime` — ESS engine run time after driver model torque request cut-off.

Controllers

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

Controller	Variant	Description
ECM	<code>SiEngineController</code> (default)	Implements the SI Controller
	<code>CiEngineController</code>	Implements the CI Controller
HCM	ECMS	Implements the Equivalent Consumption Minimization Strategy
TCM	<code>TransmissionController</code>	Implements the transmission controller

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 subsystem variants.

Drivetrain

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	
Torque Converter Automatic Transmission	Ideal Fixed Gear Transmission	Configure locked and unlocked transmission efficiency with either a 1D or 4D (default) lookup table.
	Torque Converter	Configure for external, internal (default), or no lockup.
Vehicle	Vehicle Body 1 DOF Longitudinal	Configured for 1 degrees of freedom
Wheels and Brakes	Longitudinal Wheel - Front 1	<p>For the wheels, you can configure the type of:</p> <ul style="list-style-type: none"> • Brake • Force calculation • Resistance calculation • Vertical motion <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>
	Longitudinal Wheel - Rear 1	

Simscape Drivetrain. Another way to customize the drivetrain is to select a Simscape variant. This variant incorporates physical connections to provide a flexible way to assemble components.

Use the button in the top level of the reference application to toggle between Simscape and Powertrain Blockset variants of the drivetrain subsystem.

The reference application sets the appropriate solvers to optimize performance for each engine and drivetrain combination. Select the engine variant first, then select the drivetrain using the toggle button. If you select the drivetrain before changing the engine, you may encounter a solver error.

Electric Plant

Electric Plant Subsystem	Variant	Description
Battery	BattHevP3	Configured with Lithium Ion battery
Electric Machine	MotMapped	Mapped Motor with implicit controller

Engine

Engine Subsystem	Variant	Description
Engine	SiEngineCore	Dynamic SI Core Engine with turbocharger
	SiMappedEngine (default)	Mapped SI Engine with implicit turbocharger
	SiEngineCoreNA	Dynamic naturally aspirated SI Core Engine

Limitations

MathWorks used the SI Core Engine and SI Controller to calibrate the hybrid control module (HCM). If you use the CI Core Engine and CI Controller variants, the simulation may error because the HCM does not use calibrated results.

Acknowledgment

MathWorks would like to acknowledge the contribution of Dr. Simona Onori to the ECMS optimal control algorithm implemented in this reference application. Dr. Onori is a Professor of Energy Resources Engineering at Stanford University. Her research interests include electrochemical modeling, estimation and optimization of energy storage devices for automotive and grid-level applications, hybrid and electric vehicles modeling and control, PDE modeling, and model-order reduction and estimation of emission mitigation systems. She is a senior member of IEEE.

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] Onori, S., Serrao, L., and Rizzoni, G., *Hybrid Electric Vehicles Energy Management Systems*. New York: Springer, 2016.

See Also

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

Related Examples

- “HEV P3 Reference Application” on page 7-8

More About

- “Analyze Power and Energy” on page 3-107
- “Hybrid and Electric Vehicle Reference Application Projects” on page 3-3
- “Variant Systems”

Explore the Hybrid Electric Vehicle P4 Reference Application

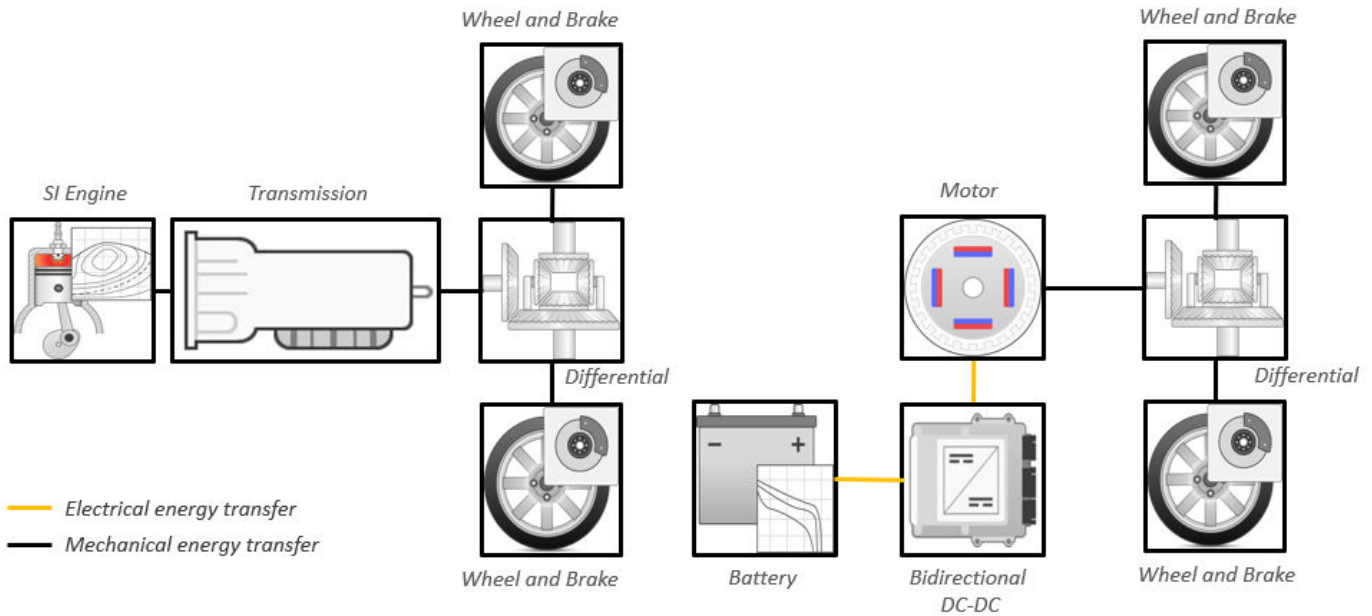
The hybrid electric vehicle (HEV) P4 reference application represents a full HEV model with an internal combustion engine, transmission, battery, motor, and associated powertrain control algorithms. Use the reference application for hardware-in-the-loop (HIL) testing, tradeoff analysis, and control parameter optimization of a HEV P4 hybrid. To create and open a working copy of the reference application project, enter

`autoblkHevP4Start`

By default, the HEV P4 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
Analyze Power and Energy	Double-click Analyze Power and Energy to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level. For more information about the live script, see “Analyze Power and Energy” on page 3-107.	NA
Drive Cycle Source block – FTP75 (2474 seconds)	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 or Open Loop variant to generate normalized acceleration and braking commands. <ul style="list-style-type: none"> Longitudinal Driver variant implements a driver model that uses vehicle target and reference velocities. Open Loop variant allows you to configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs. 	✓
Controllers subsystem	Implements a powertrain control module (PCM) containing a P4 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.	

Evaluate and Report Power and Energy

Double-click **Analyze Power and Energy** to open a live script. Run the script to evaluate and report power and energy consumption at the component- and system-level. For more information about the live script, see “Analyze Power and Energy” on page 3-107.

The script provides:

- An overall energy summary that you can export to an Excel spreadsheet.
- Engine plant, electric plant, and drivetrain plant efficiencies, including an engine histogram of time spent at the different engine plant efficiencies.
- Data logging so that you can use the Simulation Data Inspector to analyze the powertrain efficiency and energy transfer signals.

For more information about the live script, see “Analyze Power and Energy” on page 3-107.

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. The reference application has these variants.

Block Variants			Description
Longitudinal Driver (default)	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	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	No transmission.

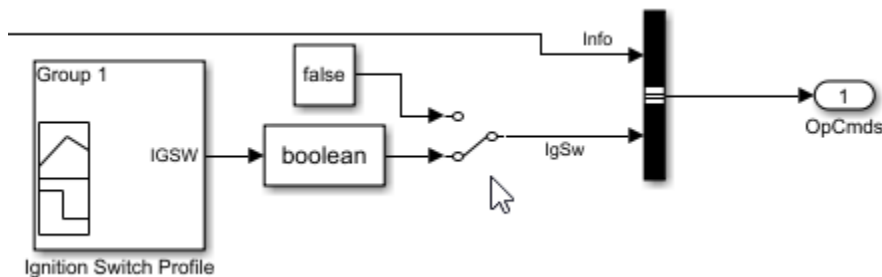
Block Variants		Description
	Scheduled	Stateflow chart models reverse, neutral, park, and N-speed gear shift scheduling.
Open Loop		Open-loop control subsystem. In the subsystem, you can configure the acceleration, deceleration, gear, and clutch commands with constant or signal-based inputs.

To idle the engine at the beginning of a drive cycle and simulate catalyst light-off before moving the vehicle with a pedal command, use the Longitudinal Driver variant. The Longitudinal Driver subsystem includes an ignition switch signal profile, `IgSw`. The engine controller uses the ignition switch signal to start both the engine and a catalyst light-off timer.

The catalyst light-off timer overrides the engine stop-start (ESS) stop function control while the catalyst light-off timer is counting up. During the simulation, after the `IgSw` down-edge time reaches the catalyst light-off time `CatLightOffTime`, normal ESS operation resumes. If there is no torque command before the simulation reaches the `EngStopTime`, the ESS shuts down the engine.

To control ESS and catalyst light-off:

- In the Longitudinal Driver Model subsystem, set the ignition switch profile `IgSw` to 'on'.



- In the engine controller model workspace, set these calibration parameters:
 - `EngStopStartEnable` — Enables ESS. To disable ESS, set the value to false.
 - `CatLightOffTime` — Engine idle time from engine start to catalyst light-off.
 - `EngStopTime` — ESS engine run time after driver model torque request cut-off.

Controllers

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

Controller	Variant	Description
ECM	<code>SiEngineController</code> (default)	Implements the SI Controller
	<code>CiEngineController</code>	Implements the CI Controller
HCM	ECMS	Implements the Equivalent Consumption Minimization Strategy
TCM	<code>TransmissionController</code>	Implements the transmission controller

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

Drivetrain Subsystem	Variant	Description
Differential and Compliance	Limited Slip Differential	You can vary the type of coupling torque and efficiency. By default, the differential is configured with an ideal wet clutch and constant efficiency.
	Open Differential	You can vary the type of differential efficiency. By default, the open differential is configured with a constant efficiency
Torque Converter Automatic Transmission	Ideal Fixed Gear Transmission	Configure locked and unlocked transmission efficiency with either a 1D or 4D (default) lookup table.
	Torque Converter	Configure for external, internal (default), or no lockup.
Vehicle	Vehicle Body 1 DOF Longitudinal	Configured for 1 degrees of freedom
Wheels and Brakes	Longitudinal Wheel - Front 1	<p>For the wheels, you can configure the type of:</p> <ul style="list-style-type: none"> • Brake • Force calculation • Resistance calculation • Vertical motion <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>
	Longitudinal Wheel - Rear 1	

Simscape Drivetrain. Another way to customize the drivetrain is to select a Simscape variant. This variant incorporates physical connections to provide a flexible way to assemble components.

Use the button in the top level of the reference application to toggle between Simscape and Powertrain Blockset variants of the drivetrain subsystem.

The reference application sets the appropriate solvers to optimize performance for each engine and drivetrain combination. Select the engine variant first, then select the drivetrain using the toggle button. If you select the drivetrain before changing the engine, you may encounter a solver error.

Electric Plant

Electric Plant Subsystem	Variant	Description
Battery	BattHevP4	Configured with Lithium Ion battery
Electric Machine	MotMapped	Mapped Motor with implicit controller

Engine

Engine Subsystem	Variant	Description
Engine	SiEngineCore	Dynamic SI Core Engine with turbocharger
	SiMappedEngine (default)	Mapped SI Engine with implicit turbocharger
	SiEngineCoreNA	Dynamic naturally aspirated SI Core Engine

Limitations

MathWorks used the SI Core Engine and SI Controller to calibrate the hybrid control module (HCM). If you use the CI Core Engine and CI Controller variants, the simulation may error because the HCM does not use calibrated results.

Acknowledgment

MathWorks would like to acknowledge the contribution of Dr. Simona Onori to the ECMS optimal control algorithm implemented in this reference application. Dr. Onori is a Professor of Energy Resources Engineering at Stanford University. Her research interests include electrochemical modeling, estimation and optimization of energy storage devices for automotive and grid-level applications, hybrid and electric vehicles modeling and control, PDE modeling, and model-order reduction and estimation of emission mitigation systems. She is a senior member of IEEE.

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] Onori, S., Serrao, L., and Rizzoni, G., *Hybrid Electric Vehicles Energy Management Systems*. New York: Springer, 2016.

See Also

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

Related Examples

- “HEV P4 Reference Application” on page 7-9

More About

- “Analyze Power and Energy” on page 3-107
- “Hybrid and Electric Vehicle Reference Application Projects” on page 3-3
- “Variant Systems”

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:

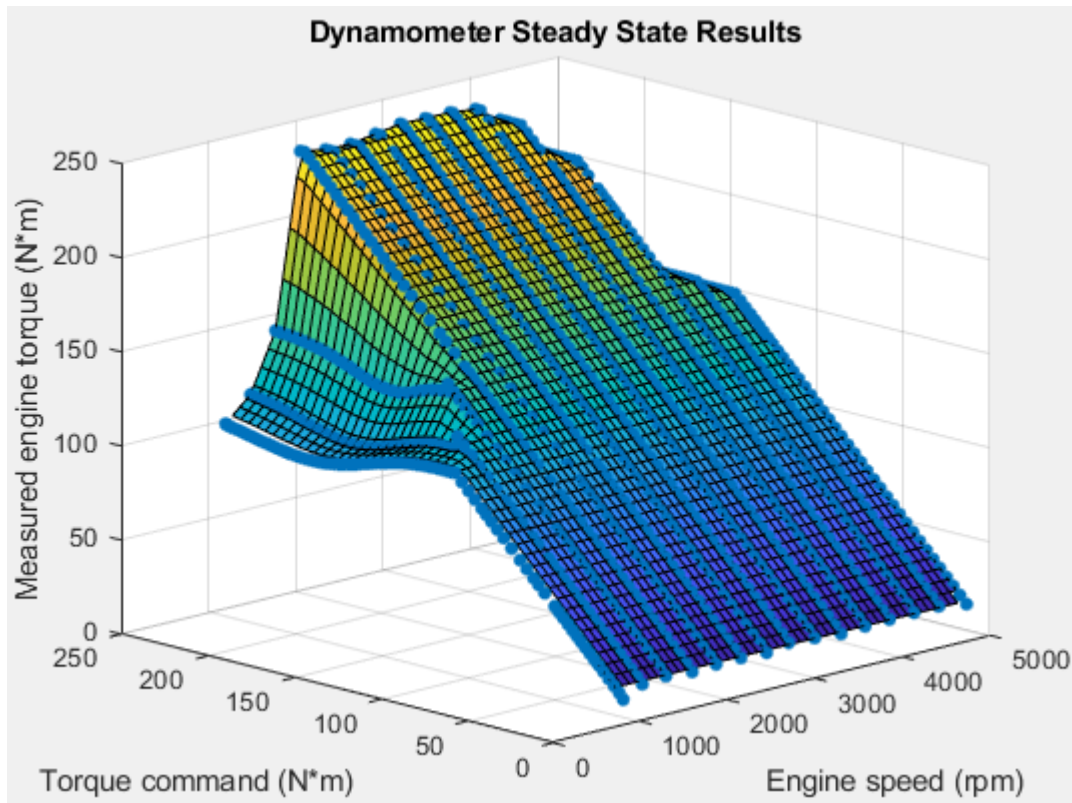
- Resizes the dynamic engine and engine calibration parameters. The **Resize Engine and Recalibrate Controller** block mask provides the updated engine performance characteristics based on the resized calibration parameters.
- Recalibrates the controller and mapped engine model to match the resized dynamic 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 `CiDynaReferenceApplication` 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 CiDynoReferenceApplication 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 ²]:	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:

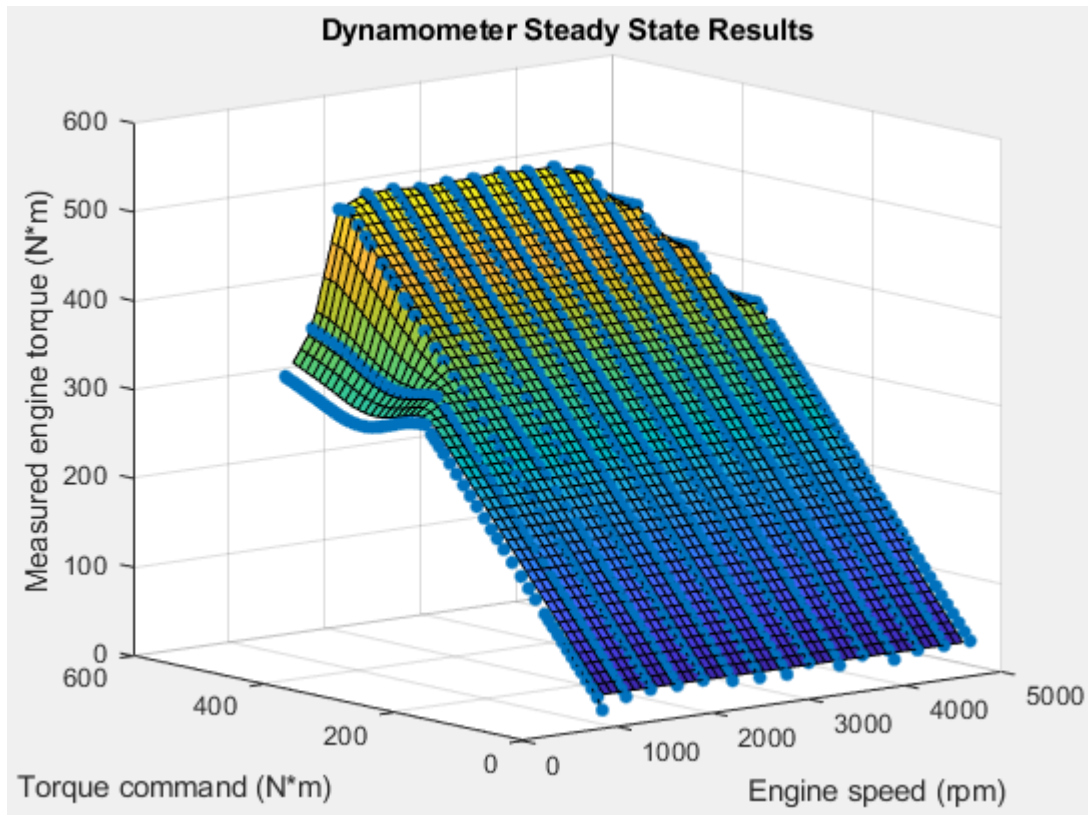
- Resizes the dynamic engine (CiEngineCore) and engine calibration parameters. The dialog box provides the updated engine performance characteristics based on the resized calibration parameters.

- 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 ²]:	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.



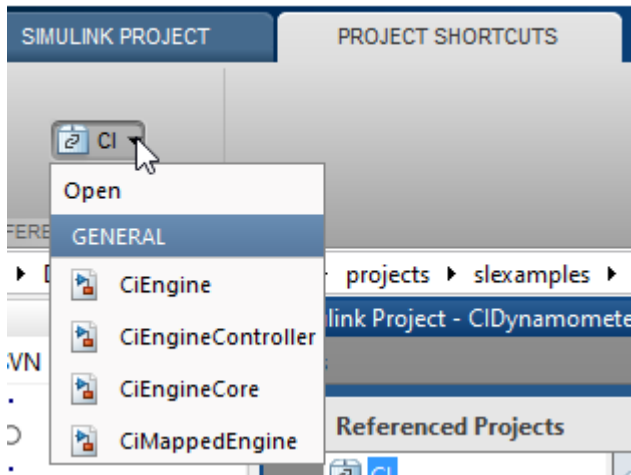
- 6 To save the engine controller, resized engine mapped variant, and resized dynamic engine variant, in the CiDynaReferenceApplication model window, save the reference application.

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:

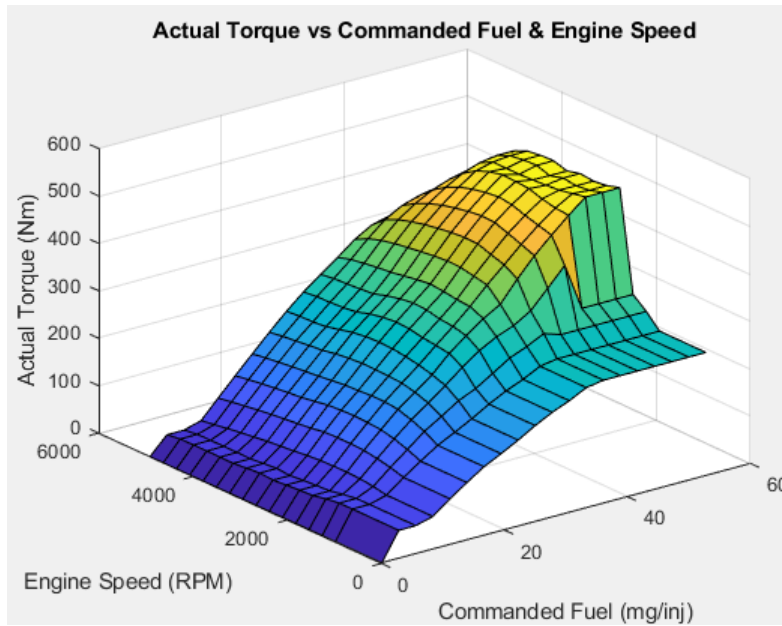
- 1 In MATLAB, use the **Project Shortcuts** tab to open these models:
 - CiEngineController
 - CiEngineCore
 - CiMappedEngine



2 Use the Model Explorer to view the resized parameters:

Engine Model	Model Explorer																		
Controller — CiEngineCont roller	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Simulink Root</p> <ul style="list-style-type: none"> Base Workspace CiDynoReferenceApplication* CiEngineCore CiEngineController <ul style="list-style-type: none"> Model Workspace Configuration (Active) Reference Start Stop Logic CiMappedEngine </div> <div style="width: 50%;"> <p>Column View: Data Objects Show Details</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 checked="" 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 checked="" 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 checked="" 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																		
Mapped — CiMappedEngi ne	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Simulink Root</p> <ul style="list-style-type: none"> Base Workspace CiDynoReferenceApplication* CiEngineCore CiEngineController CiMappedEngine <ul style="list-style-type: none"> Model Workspace Configuration (Active) Accessory Load Model Actuators </div> <div style="width: 50%;"> <p>Column View: Data Objects Show Details</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr><td><input type="checkbox"/> AccPwrpb</td><td>[0 4.025359766529133]</td></tr> <tr><td><input type="checkbox"/> AccSpdbp</td><td>[299.0535069180764 747.633767]</td></tr> <tr><td><input type="checkbox"/> Cps</td><td>2</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>6.451612903225807</td></tr> </tbody> </table> </div> </div>	Name	Value	<input type="checkbox"/> AccPwrpb	[0 4.025359766529133]	<input type="checkbox"/> AccSpdbp	[299.0535069180764 747.633767]	<input type="checkbox"/> Cps	2	<input type="checkbox"/> NCyl	8	<input type="checkbox"/> Pstd	101325	<input type="checkbox"/> Rair	287.05	<input type="checkbox"/> Sinj	6.451612903225807		
Name	Value																		
<input type="checkbox"/> AccPwrpb	[0 4.025359766529133]																		
<input type="checkbox"/> AccSpdbp	[299.0535069180764 747.633767]																		
<input type="checkbox"/> Cps	2																		
<input type="checkbox"/> NCyl	8																		
<input type="checkbox"/> Pstd	101325																		
<input type="checkbox"/> Rair	287.05																		
<input type="checkbox"/> Sinj	6.451612903225807																		
Dynamic — CiEngineCore	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Simulink Root</p> <ul style="list-style-type: none"> Base Workspace CiDynoReferenceApplication* CiEngineCore <ul style="list-style-type: none"> Model Workspace Configuration (Active) EGR Engine Coolant Temperature Fuel System Subsystem CiEngineController CiMappedEngine </div> <div style="width: 50%;"> <p>Column View: Data Objects Show Details</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr><td><input type="checkbox"/> AirFilterArea</td><td>0.00193642497768315</td></tr> <tr><td><input type="checkbox"/> AirIntakeVol</td><td>0.00363428989067364</td></tr> <tr><td><input type="checkbox"/> C_eng</td><td>40000</td></tr> <tr><td><input type="checkbox"/> CompEff</td><td><60x100 double></td></tr> <tr><td><input type="checkbox"/> CompMassFlwRate</td><td><60x100 double></td></tr> <tr><td><input type="checkbox"/> CompPrsRatioBreakPoints</td><td>[1 1.028080808080808 1.056161616...</td></tr> <tr><td><input type="checkbox"/> CompRefPrs</td><td>101325</td></tr> <tr><td><input type="checkbox"/> CompRefTemp</td><td>300</td></tr> </tbody> </table> </div> </div>	Name	Value	<input type="checkbox"/> AirFilterArea	0.00193642497768315	<input type="checkbox"/> AirIntakeVol	0.00363428989067364	<input type="checkbox"/> C_eng	40000	<input type="checkbox"/> CompEff	<60x100 double>	<input type="checkbox"/> CompMassFlwRate	<60x100 double>	<input type="checkbox"/> CompPrsRatioBreakPoints	[1 1.028080808080808 1.056161616...	<input type="checkbox"/> CompRefPrs	101325	<input type="checkbox"/> CompRefTemp	300
Name	Value																		
<input type="checkbox"/> AirFilterArea	0.00193642497768315																		
<input type="checkbox"/> AirIntakeVol	0.00363428989067364																		
<input type="checkbox"/> C_eng	40000																		
<input type="checkbox"/> CompEff	<60x100 double>																		
<input type="checkbox"/> CompMassFlwRate	<60x100 double>																		
<input type="checkbox"/> CompPrsRatioBreakPoints	[1 1.028080808080808 1.056161616...																		
<input type="checkbox"/> CompRefPrs	101325																		
<input type="checkbox"/> 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 turbocharged 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 variant 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 **Resize option** 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.

When in Displacement mode, you can define the maximum torque and the engine speed at which maximum torque occurs. Click the checkboxes to enable these entry fields.

You can choose the architecture, air path configuration (turbocharged or naturally aspirated), and presence or absence of cooled exhaust gas re-circulation (EGR) of your engine model. After making your selections, click **Resize Engine** to set the engine variant.

The available engine variants are:

Engine Subsystem	Variant	Description
Engine	SiEngineCore (default)	Dynamic Inline Turbo SI Core Engine
	SiEngineCoreNA	Dynamic SI Inline Naturally Aspirated Engine
	SiEngineCoreV	Dynamic SI V Twin-Turbo Single-Intake Engine
	SiEngineCoreVNA	Dynamic SI V Naturally Aspirated Engine
	SiEngineCoreVThr2	Dynamic SI V Twin-Turbo Twin-Intake Engine

After you apply the changes, the reference application:

- Resizes the dynamic engine and engine calibration parameters. The **Resize Engine and Recalibrate Controller** block mask 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.

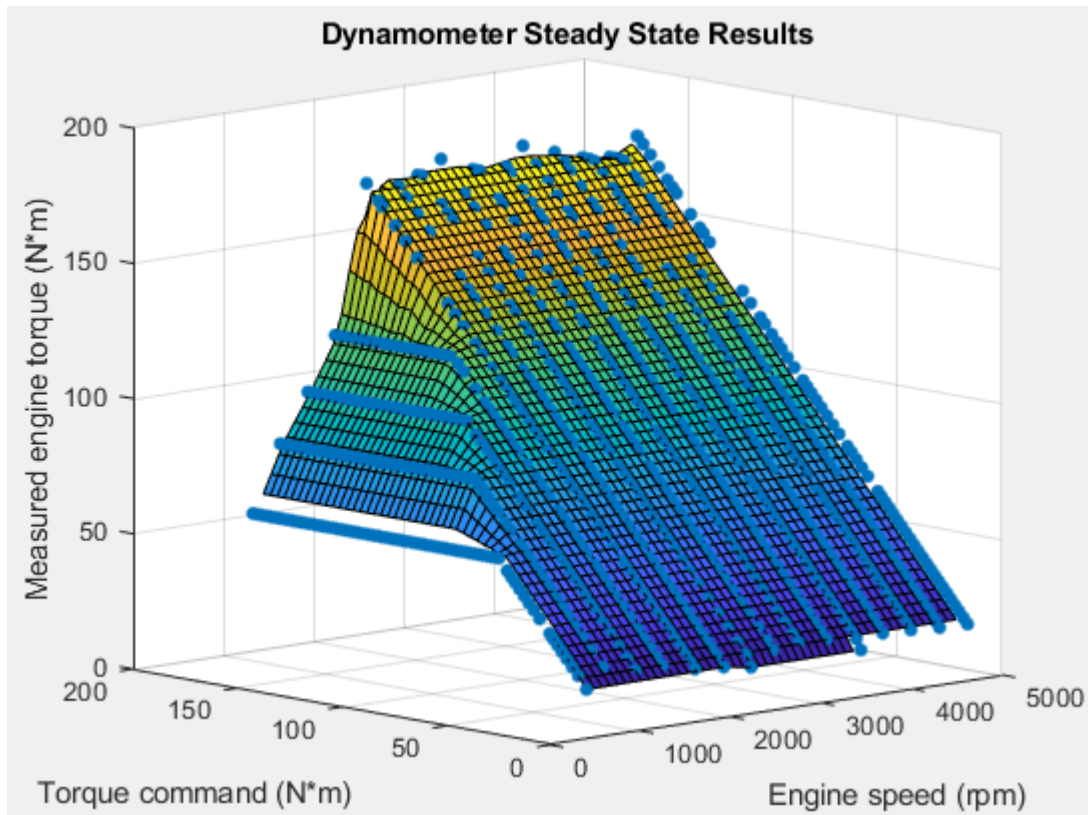
You can use the variants in other applications, for example, in vehicle projects that require a larger engine model.

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 `SiDynoReferenceApplication` 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 SiDynoReferenceApplication 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 ²]:	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:

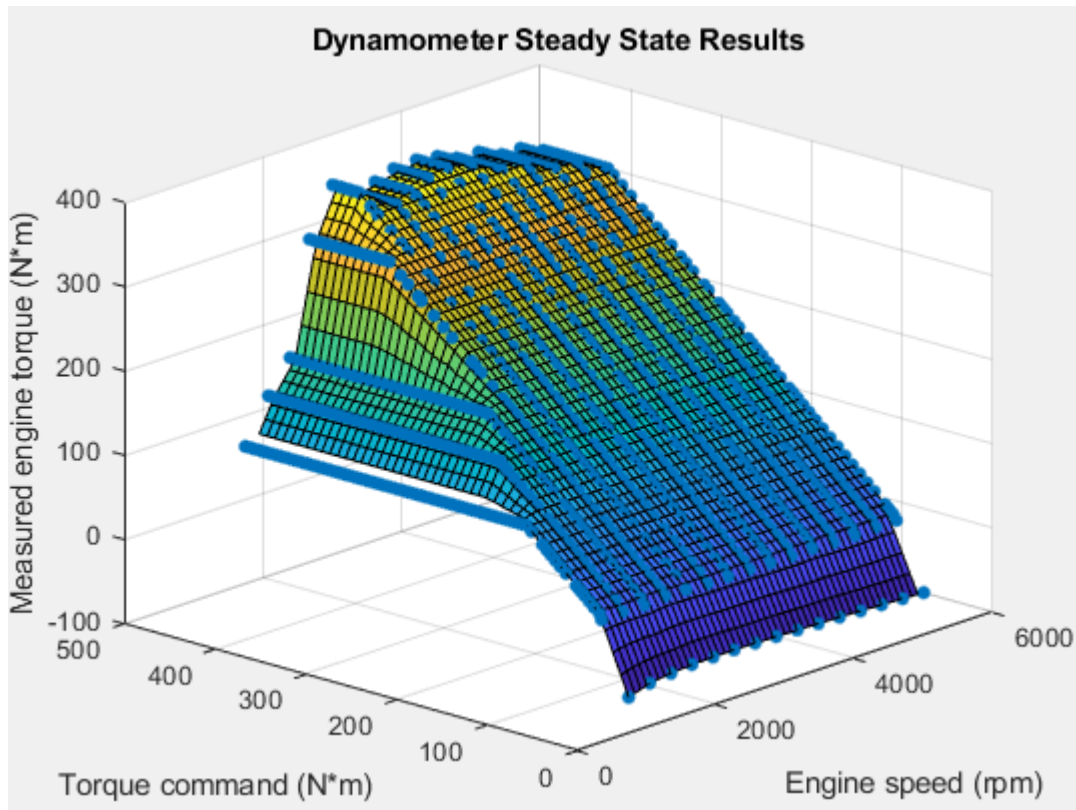
- Resizes the dynamic engine (SiEngineCore) and engine calibration parameters. The **Recalibrate Engine** dialog box provides the updated engine performance characteristics based on the resized calibration parameters.

- 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 ²]:	0.031
Fuel injector slope [mg/ms]:	6.44

- Examine the dynamometer steady-state results. For engine speeds 2000-5000 rpm, the measured engine torque approaches 400 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.



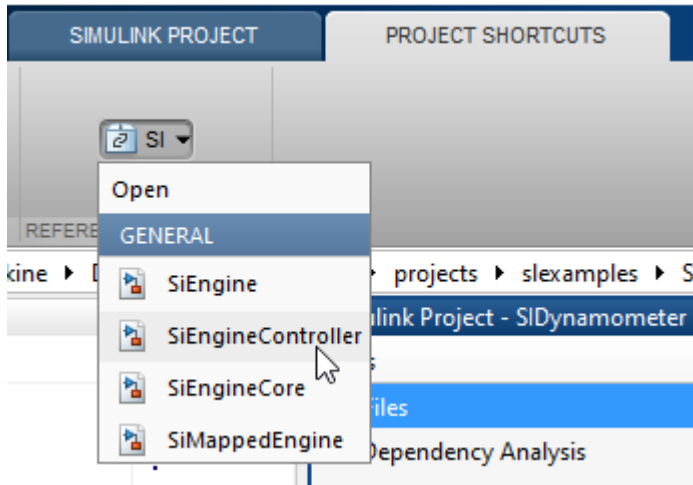
- 6 To save the engine controller, resized engine mapped variant, and resized dynamic engine variant, in the SiDynoReferenceApplication model window, save the reference application.

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:

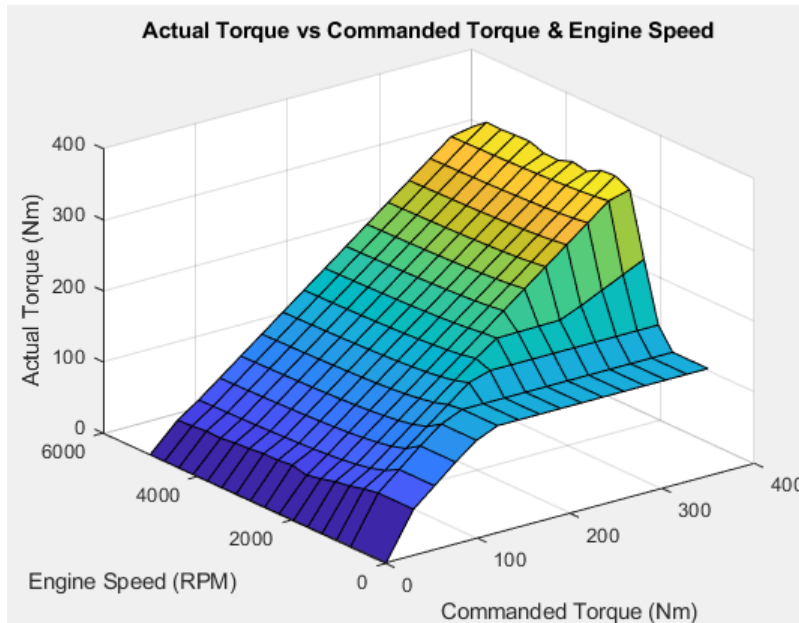
- 1 In MATLAB, use the **Project Shortcuts** tab to open these models:
 - SiEngineController
 - SiEngineCore
 - SiMappedEngine



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

Engine Model	Model Explorer																
Controller — SiEngineCont roller	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <ul style="list-style-type: none"> Simulink Root <ul style="list-style-type: none"> Base Workspace SiDynoReferenceApplication* SiEngineCore SiEngineController <ul style="list-style-type: none"> Model Workspace Configuration (Active) Reference Start Stop Logic SiMappedEngine </div> <div style="width: 50%;"> <p>Column View: Data Objects Show Details</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>EngStopStartEnable</td> <td>true</td> </tr> <tr> <td>EngStopTime</td> <td>5</td> </tr> <tr> <td>NCyl</td> <td>8</td> </tr> <tr> <td>N_idle</td> <td>750.298962153551</td> </tr> <tr> <td>Pstd</td> <td>101325</td> </tr> <tr> <td>Rair</td> <td>287</td> </tr> </tbody> </table> </div> </div>	Name	Value	EngStopStartEnable	true	EngStopTime	5	NCyl	8	N_idle	750.298962153551	Pstd	101325	Rair	287		
Name	Value																
EngStopStartEnable	true																
EngStopTime	5																
NCyl	8																
N_idle	750.298962153551																
Pstd	101325																
Rair	287																
Mapped — SiMappedEngi ne	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <ul style="list-style-type: none"> Simulink Root <ul style="list-style-type: none"> Base Workspace SiDynoReferenceApplication* SiEngineCore SiEngineController SiMappedEngine <ul style="list-style-type: none"> Model Workspace Configuration (Active) Accessory Load Model Actuators Three-Way Catalyst </div> <div style="width: 50%;"> <p>Column View: Data Objects Show Details</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <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> </div> </div>	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	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <ul style="list-style-type: none"> Simulink Root <ul style="list-style-type: none"> Base Workspace SiDynoReferenceApplication* SiEngineCore <ul style="list-style-type: none"> Model Workspace Configuration (Active) Engine Coolant Temperature LP EGR Subsystem SiEngineController SiMappedEngine </div> <div style="width: 50%;"> <p>Column View: Data Objects Show Details</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <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>CompSpdBkPts</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> </div> </div>	Name	Value	CompRefPrs	101325	CompRefTemp	300	CompSpdBkPts	[3.536943227040031 354.199357	CoolantTemp	310	Cps	2	EgrArea	0.000627817914172484	EgrCoolantTemp	360
Name	Value																
CompRefPrs	101325																
CompRefTemp	300																
CompSpdBkPts	[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 3-14

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.

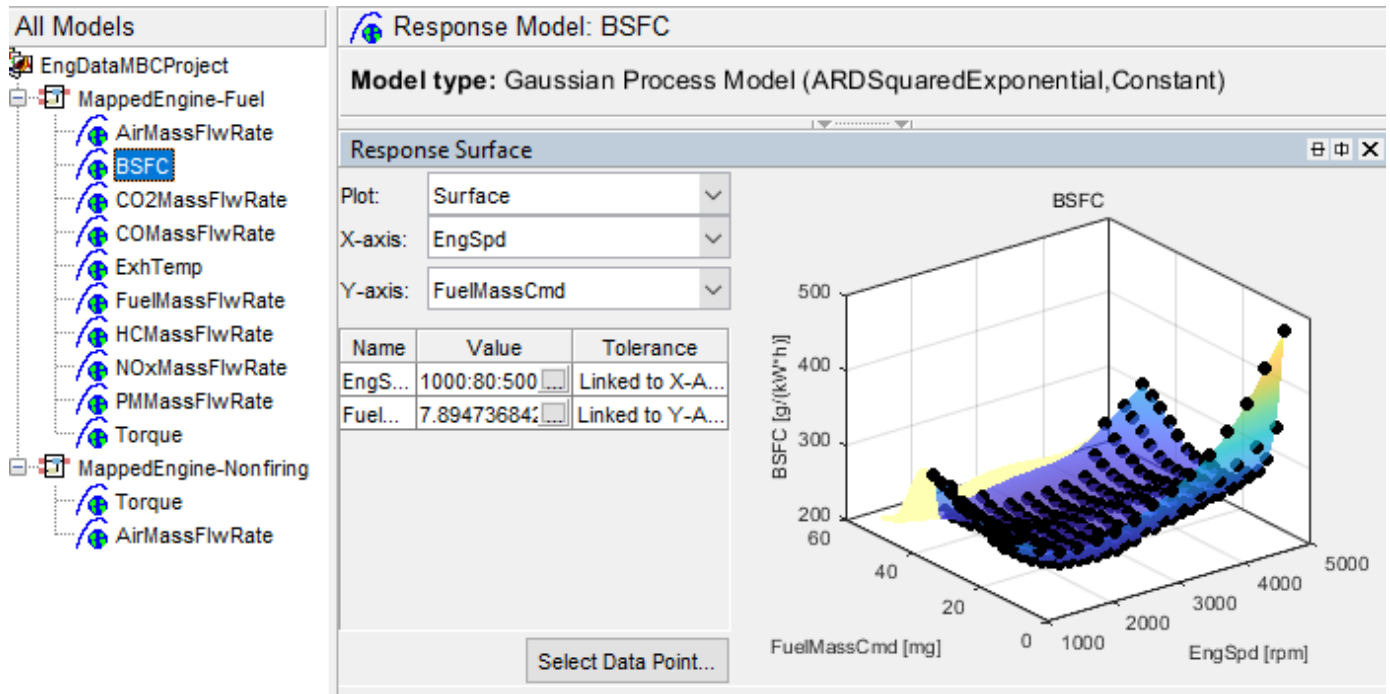
Firing Data	Description	Data Requirements for Generating Mapped Engine Tables	
		Function of Fuel Mass and Engine Speed	Function of Torque and Engine Speed
FuelMassCmd	Injected fuel mass, in mg per injection	<i>Required</i>	<i>Not used</i>
Torque	Engine torque command, in N·m	<i>Required</i>	<i>Required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

Firing Data	Description	Data Requirements for Generating Mapped Engine Tables	
		Function of Fuel Mass and Engine Speed	Function of Torque and Engine Speed
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>

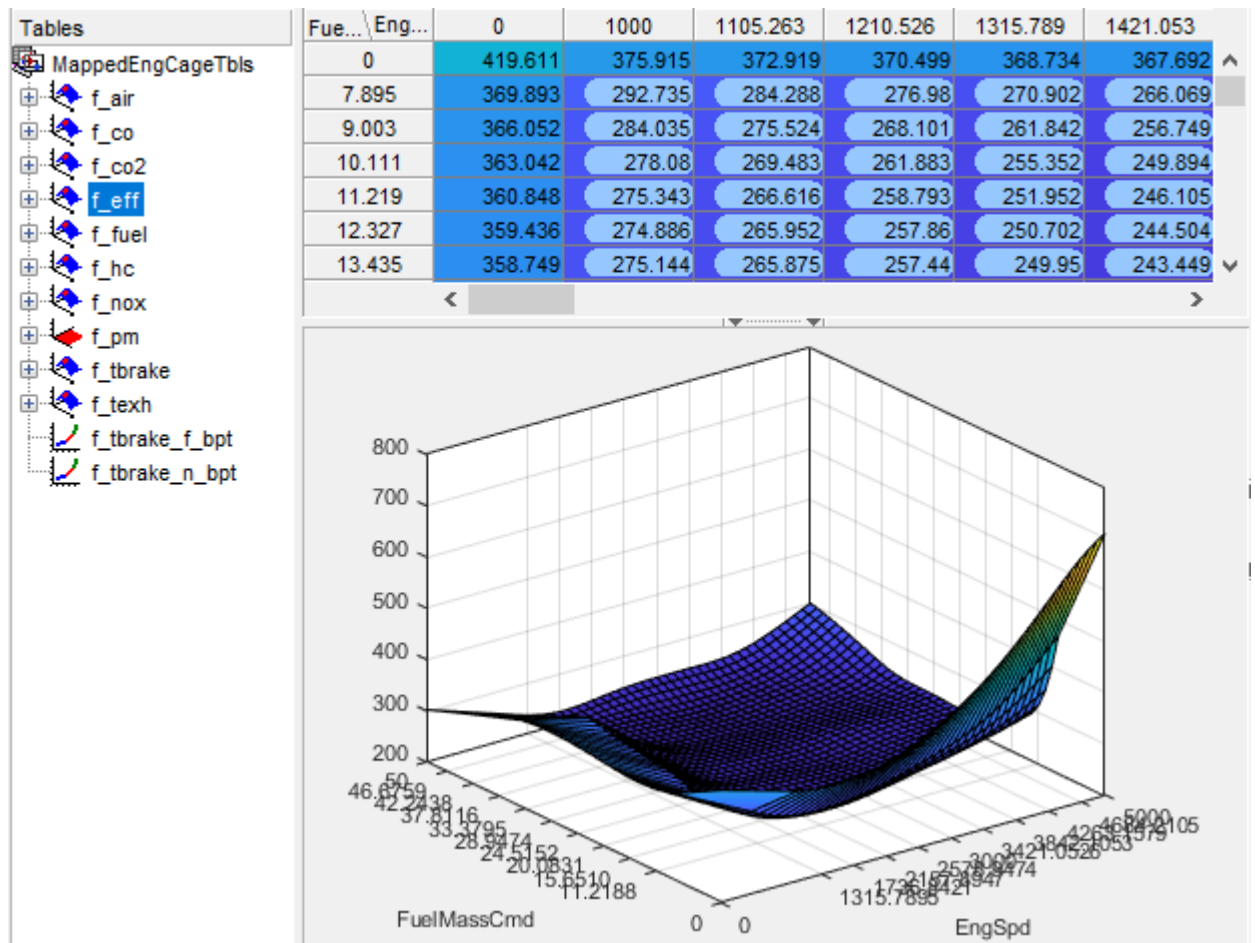
Nonfiring data contains data collected at different engine speeds without fuel consumption.

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.

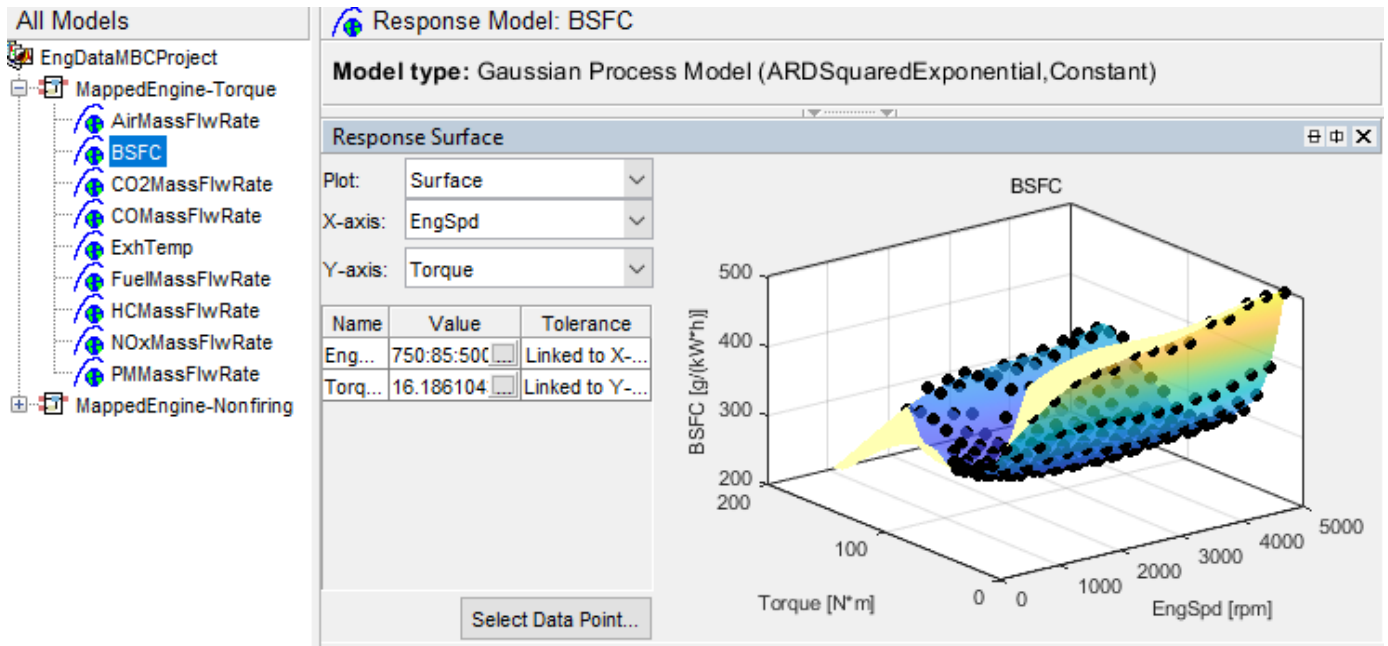
Firing Data	Description	Data Requirements for Generating Mapped Engine Tables
FuelMassCmd	Injected fuel mass, in mg per injection	<i>Not Used</i>
Torque	Engine torque command, in N·m	<i>Required</i>
EngSpd	Engine speed, in rpm	<i>Required</i>
AirMassFlwRate	Air mass flow, in kg/s	<i>Optional</i>
FuelMassFlwRate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

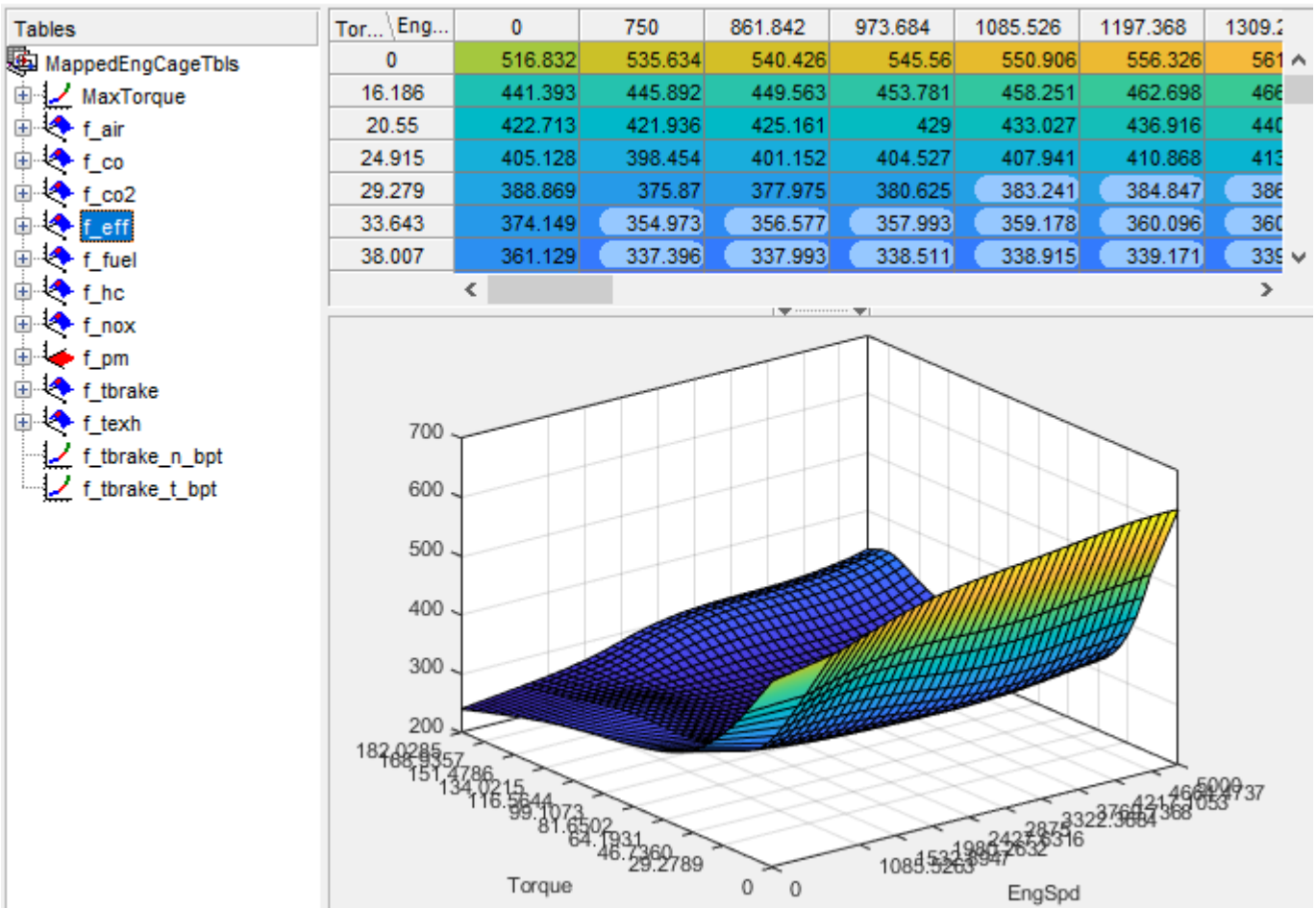
Nonfiring Data	Description	Data Requirements for Generating Mapped Engine Tables
FuelMassCmd	Injected fuel mass, in mg per injection	Not used
Torque	Engine torque command, in N·m	Required
EngSpd	Engine speed, in rpm	Required
AirMassFlwRate	Air mass flow, in kg/s	Optional

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 3-14
- “What Is CAGE?” (Model-Based Calibration Toolbox)

- “Mapped SI Lookup Tables as Functions of Engine Torque and Speed” (Model-Based Calibration Toolbox)

Generate a Deep Learning SI Engine Model

If you have the Deep Learning Toolbox and Statistics and Machine Learning Toolbox, you can generate a dynamic deep learning spark-ignition (SI) engine model to use for powertrain control, diagnostic, and estimator algorithm design. For example, fit a deep learning model to measured engine-out transient emissions data and use it for aftertreatment control and diagnostic algorithm development. The deep learning SI engine models the dynamic engine behavior from measured laboratory data or a high-fidelity engine model.

To train the deep learning SI engine model, Powertrain Blockset uses this SI engine data.

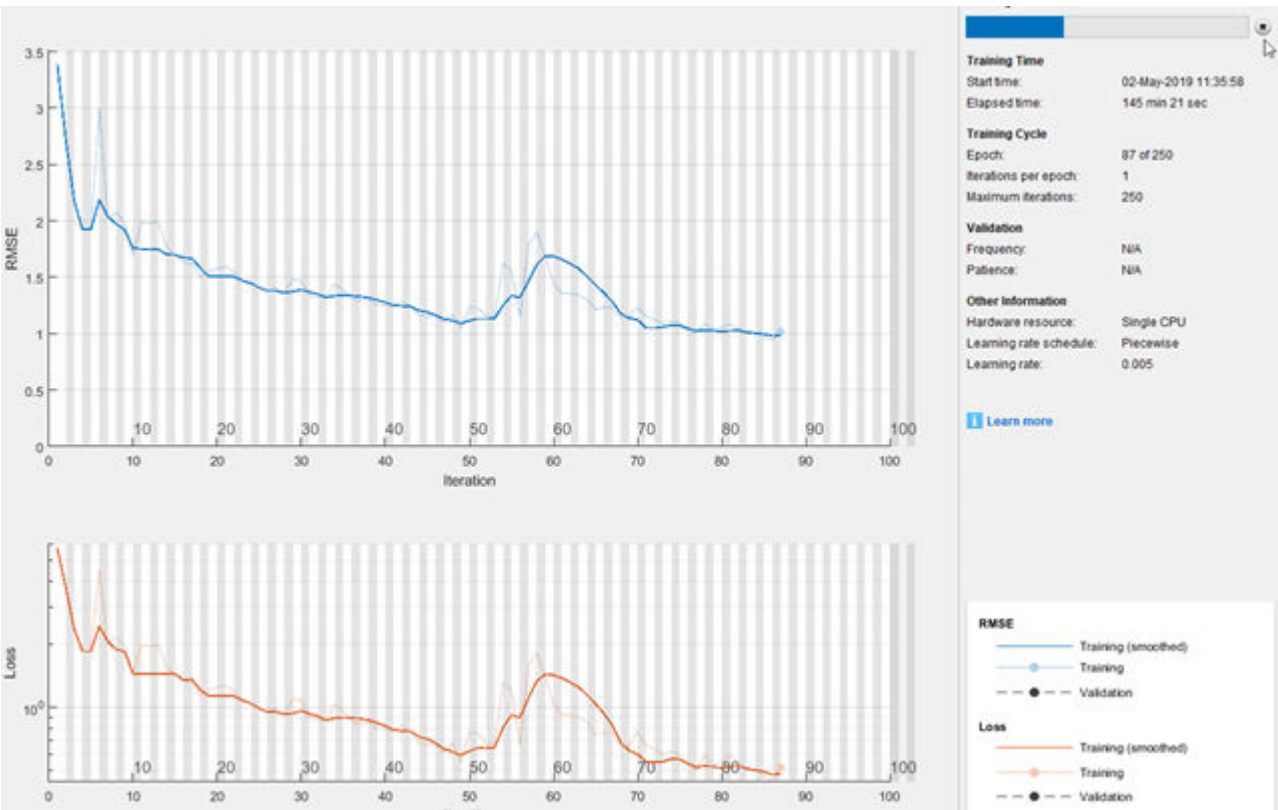
Input Data	Output Data
Engine speed	Brake torque
Commanded torque	Intake manifold gas pressure
	Intake manifold gas temperature
	Fuel flow
	Intake air mass flow
	Exhaust gas temperature at exhaust manifold inlet
	Turbocharger speed
	Engine out (EO) hydrocarbon (HC) emission mass flow
	EO carbon monoxide (CO) emission mass flow
	EO nitric oxide and nitrogen dioxide emissions (NOx) emission mass flow
	EO carbon dioxide (CO ₂) emission mass flow

To generate the deep learning engine model, follow these steps.

- 1 If it is not already opened, open the reference application.
`autoblkSIDynamometerStart`
- 2 Double-click **Generate Deep Learning Engine Model**. Generating the model can take several hours.

By default, to train the deep learning engine model, the reference application generates design of experiment (DoE) response data from the SI Core Engine block. Alternatively, you can use engine data generated by Powertrain Blockset from Gamma Technologies LLC engine models or other high-fidelity engine models.

- View the training progress window to see the iteration or stop the training.

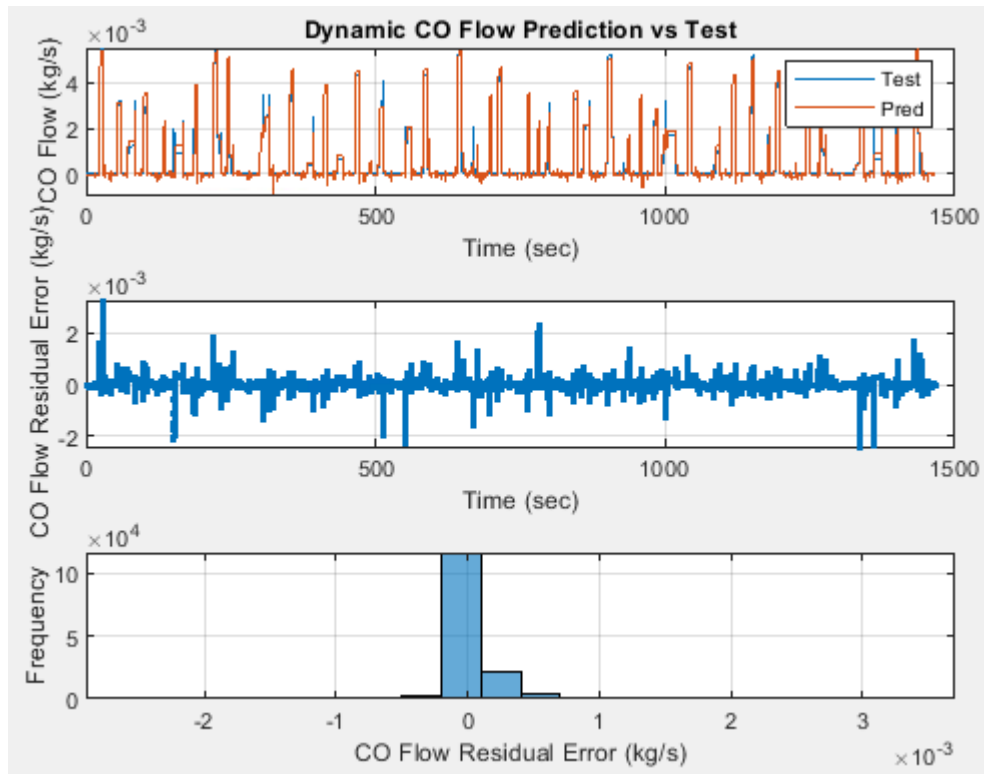


- As the training runs, Powertrain Blockset logs this data in the base workspace.
 - EngineInputs — m-by-2 array of engine inputs
 - EngineOutputs — m-by-11 array of engine outputs

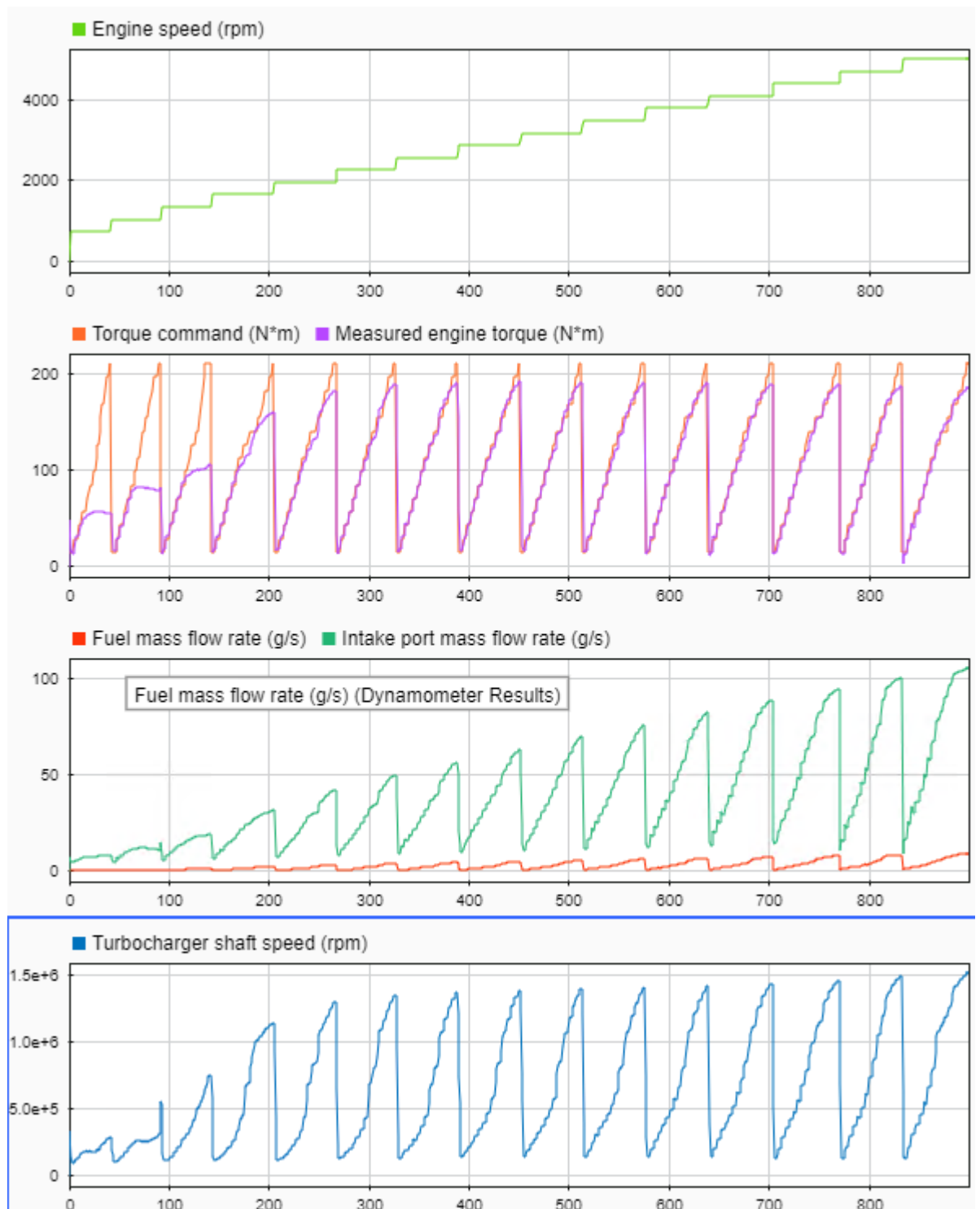
Powertrain Blockset uses half the data to train the model and half to test the model.

3 After you generate the deep learning SI model, view the results.

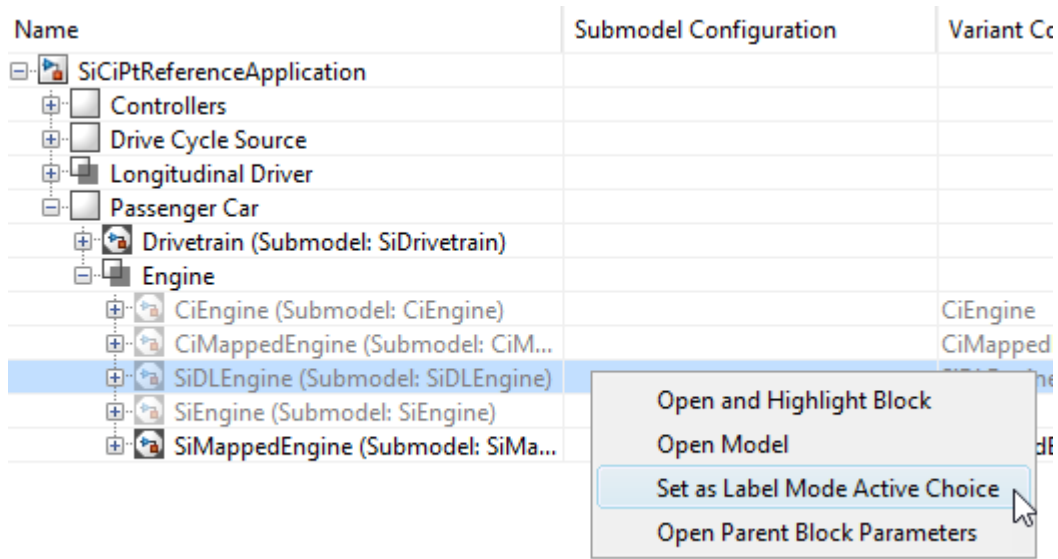
- For each engine output, a plot displays the SI engine deep learning model (Pred) and the test data (Test). For example, this plot shows the comparison for dynamic engine-out CO emissions mass flow.



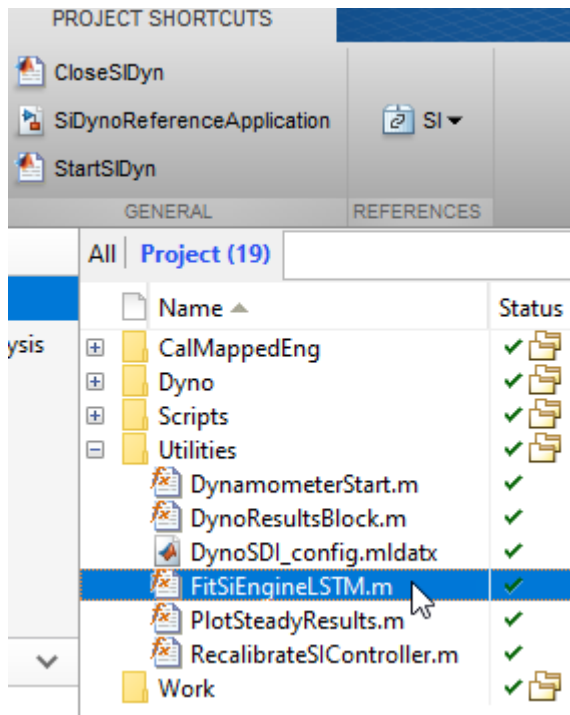
- The Simulation Data Inspector displays the SI engine deep learning model speed, torque commands, fuel mass flow rate, and shaft speed.



- 4 You can use the deep learning SI model, SiDLEngine, as an engine plant model variant in the conventional vehicle and hybrid electric vehicle (HEV) reference applications. For example, in the conventional vehicle reference application, on the **Modeling** tab, in the **Design** section, open the Variant Manager. Navigate to Passenger Car > Engine. Right-click to set SiDLEngine as the active choice.



- To fit your own deep learning SI engine model or adjust the deep learning training settings, use the `FitSiEngineLSTM.m` script in the reference application project folder.



See Also

Mapped SI Engine | SI Core Engine

More About

- “Explore the SI Engine Dynamometer Reference Application” on page 3-14

- “Deep Learning Toolbox”
- “Statistics and Machine Learning 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"> • If you have engine data from a dynamometer or a design tool like GT-POWER. • For quasi steady-state engine simulations.
	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"> • If you need a more detailed dynamic model and have component-level data. • To analyze the impact of individual engine components on the overall performance.
	SiEngine		

See Also

More About

- Mapped CI Engine
- Mapped SI Engine
- CI Core Engine
- SI Core Engine
- “Engine Calibration Maps” on page 2-31

Analyze Power and Energy

To assess powertrain efficiency, you can evaluate and report power and energy for component-level blocks and system-level reference applications.

These reference applications include live scripts that analyze the energy consumption. After you open the reference applications, double-click **Analyze Power and Energy** to open the live script. To generate the energy report, select **Run**.

- “Explore the Conventional Vehicle Reference Application” on page 3-4
- “Explore the Hybrid Electric Vehicle Multimode Reference Application” on page 3-18
- “Explore the Hybrid Electric Vehicle Input Power-Split Reference Application” on page 3-31
- “Explore the Hybrid Electric Vehicle P0 Reference Application” on page 3-40
- “Explore the Hybrid Electric Vehicle P1 Reference Application” on page 3-47
- “Explore the Hybrid Electric Vehicle P2 Reference Application” on page 3-54
- “Explore the Hybrid Electric Vehicle P3 Reference Application” on page 3-63
- “Explore the Hybrid Electric Vehicle P4 Reference Application” on page 3-70
- “Explore the Electric Vehicle Reference Application” on page 3-25

The plant model blocks calculate transferred, stored, and not transferred power. The blocks use the Power Accounting Bus Creator to log the power signals that the live script uses. If you use your own block in the reference application, add the Power Accounting Bus Creator to your subsystem to log the power signals.

The live script provides:

- An overall energy summary that the script exports to an Excel spreadsheet.
- Engine plant, electric plant, and drivetrain efficiencies, including an engine plant histogram of time spent at different efficiencies.
- Data logging so that you can use the Simulation Data Inspector to analyze the powertrain efficiency, power, and energy signals.

Live Script

The live script uses the `autoblks.pwr.PlantInfo` class to turn on data logging, run the simulation, and report power and energy results. Before running the simulation, the script finds all of the Power Accounting Bus Creator blocks in the model and turns on data logging. During the simulation, the model logs the transferred, not transferred, and stored power. The script uses the logged data to calculate efficiency, energy loss, energy input, and energy output for each component and subsystem. If the component does not conserve energy, the script issues warnings. Finally, the script provides an overall vehicle energy summary, a detailed subsystem summary, and Simulation Data Inspector time series plots.

Run Simulation

When you run the simulation, the script creates the `autoblks.pwr.PlantInfo` object to analyze the model energy and power consumption. Use these properties to set the units:

- `PwrUnits`

- **EnrgyUnits**

When the script creates the `autoblks.pwr.PlantInfo` object, the constructor searches the model for Power Accounting Bus Creator blocks. Starting at the top-level model, the constructor creates a child object for each subsystem that contains Power Accounting Bus Creator blocks. The constructor stops at the blocks that have a Power Accounting Bus Creator.

To track the power transferred between the components, the constructor uses the transferred power ports defined in the Power Accounting Bus Creator block mask.

To calculate the efficiency, the `autoblks.pwr.PlantInfo` class `Eff` property implements this equation.

$$\eta = \left| \frac{\sum P_{output} - \sum P_{store}(P_{store} > 0)}{\sum P_{input} - \sum P_{store}(P_{store} < 0)} \right|$$

To determine if the system conserves energy, the `isEnrgyBalanced` method checks the energy conservation at each time step. If the energy conservation error is within an error tolerance, the method returns true.

Overall Summary

The overall summary provides the efficiency, energy loss, energy input, energy output, and energy stored at the component- and system-level. The summary includes hyperlinks that you can use to investigate model blocks and subsystems.

The script uses the `autoblks.pwr.PlantInfo` class `xlsSysSummary` method to export the analysis to an Excel spreadsheet.

Plant Summary

The script provides engine plant, electric plant, and drivetrain efficiencies. Specifically, the script includes the signal energy, and an engine efficiency histogram.

Simulation Data Inspector Summary

The script includes the `autoblks.pwr.PlantInfo` class `sdiSummary` method to create Simulation Data Inspector power, energy, and efficiency signal plots.

Power Signals

The system-level power and energy accounting tests that the system satisfies the conservation of energy. If the component does not conserve energy, the live script issues warnings.

The Power Accounting Bus Creator for the plant blocks in the reference applications sort the signals into three power types.

Power Type		Description	Examples
P_{trans}	Transferred	Power transferred between blocks: <ul style="list-style-type: none"> Positive signals indicate flow into block Negative signals indicate flow out of block 	<ul style="list-style-type: none"> Crankshaft power transferred from mapped engine to transmission. Road load power transferred from wheel to vehicle. Rate of heat flow transferred from throttle to manifold volume.
$P_{nottrans}$	Not transferred	Power crossing the block boundary, but not transferred: <ul style="list-style-type: none"> Positive signals indicate an input Negative signals indicate a loss 	<ul style="list-style-type: none"> Rate of heat transfer with the environment. <ul style="list-style-type: none"> From environment is an input (positive signal) To environment is a loss (negative signal) Flow boundary with the environment. <ul style="list-style-type: none"> From environment is an input (positive signal) To environment is a loss (negative signal) Mapped engine fuel flow.
P_{store}	Stored	Stored energy rate of change: <ul style="list-style-type: none"> Positive signals indicate an increase Negative signals indicate a decrease 	Energy rate of change: <ul style="list-style-type: none"> Battery storage Kinetic energy in drivetrain components Vehicle potential energy Vehicle velocity

The power signals satisfy this equation.

$$\sum P_{trans} + \sum P_{nottrans} = \sum P_{store}$$

To conserve energy, sum of transferred power signals must be near zero.

The equations use these variables.

P_{trans}	Transferred power
$P_{nottrans}$	Not transferred power
P_{store}	Stored power
P_{input}, P_{output}	Input and output power logged by Power Accounting Bus Creator block

See Also

Power Accounting Bus Creator | [autoblks.pwr.PlantInfo](#)

Related Examples

- “Conventional Vehicle Powertrain Efficiency” on page 1-15

More About

- Simulation Data Inspector

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, on the **Simulation** tab, select **New > Project > New Project**.

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

- 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

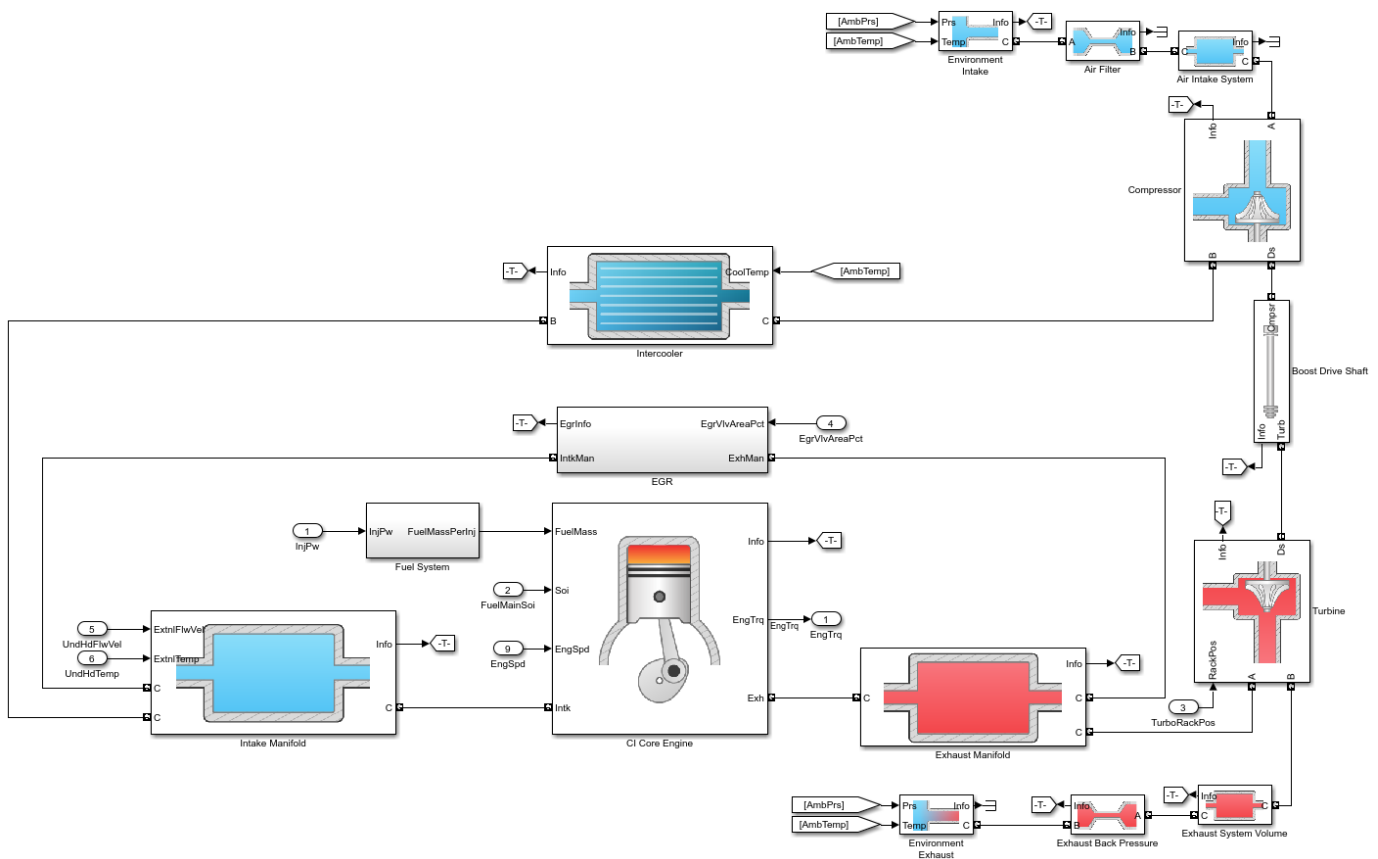
The Controller folder contains the `CiEngineController.slx` model. The model uses the CI Controller block and a `Start Stop Logic` subsystem to control the CI engine plant model.

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
- “Variant Systems”

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, on the **Simulation** tab, select **New > Project > New Project**.

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

- 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

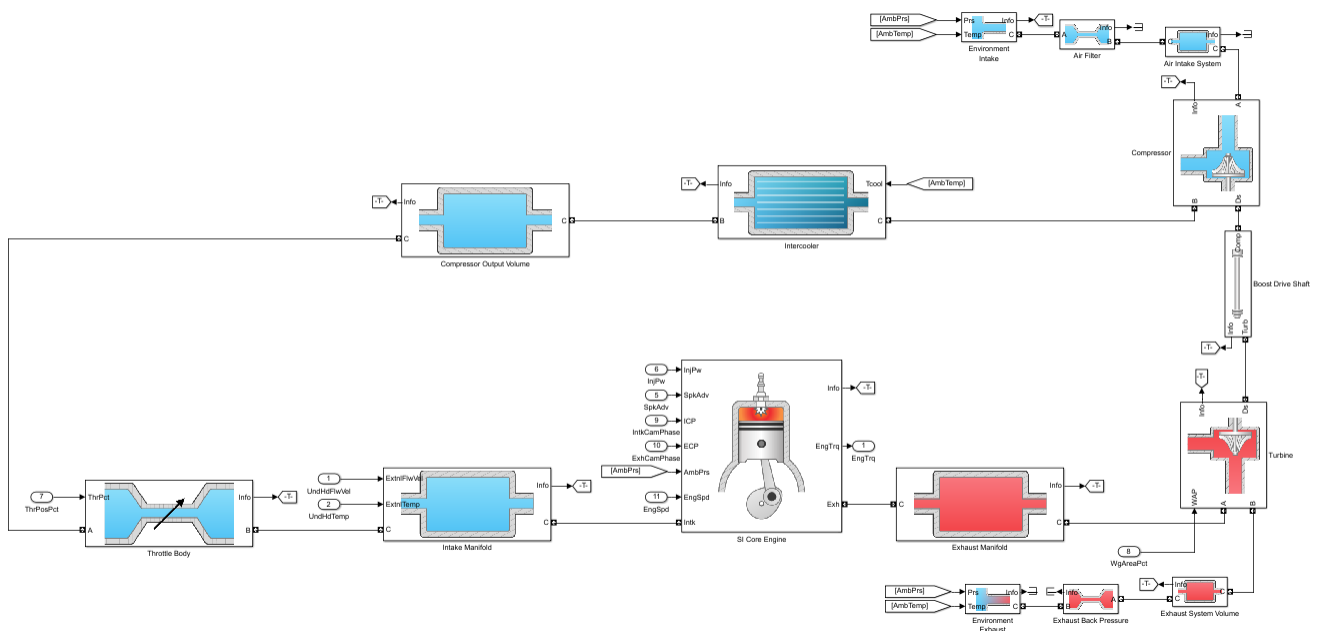
The Controller folder contains the `SiEngineController.slx` model. The model uses the SI Controller block and a `Start Stop Logic` subsystem to control the SI engine plant model.

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
- “Variant Systems”

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

More About

- “Track Drive Cycle Errors” on page 5-3

Track Drive Cycle Errors

This example shows how to use the Drive Cycle Source block to identify drive cycle faults when you run the conventional vehicle reference application with the FTP–75 drive cycle.

- 1 Open the conventional vehicle reference application project. By default, the application has a FTP–75 drive cycle with error tracking disabled.

autoblckConVehStart

Project files open in a writable location.

- 2 Open the Drive Cycle Source block. On the **Fault Tracking** tab, select these parameters:

- **Enable fault tracking**
- **Enable failure tracking**

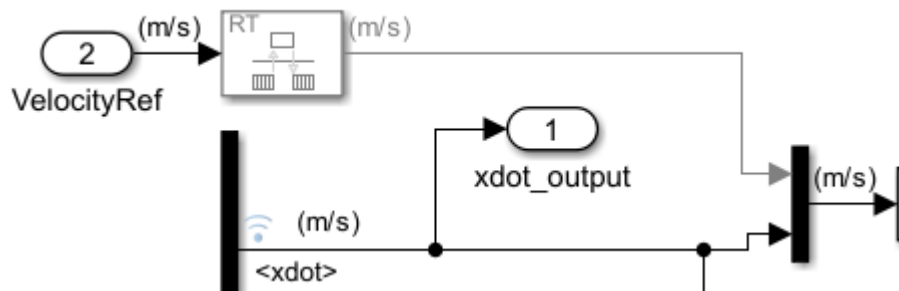
- 3 Review the parameters that specify the fault and failure conditions. If the vehicle speed is not within the allowable speed range during the time tolerance, the block sets a fault condition. Accept the default EPA dynamometer driving schedule parameter settings by clicking **OK**.

This table provides the settings for the EPA standard and the Worldwide Harmonised Light Vehicle Test Procedure (WLTP) laboratory tests.

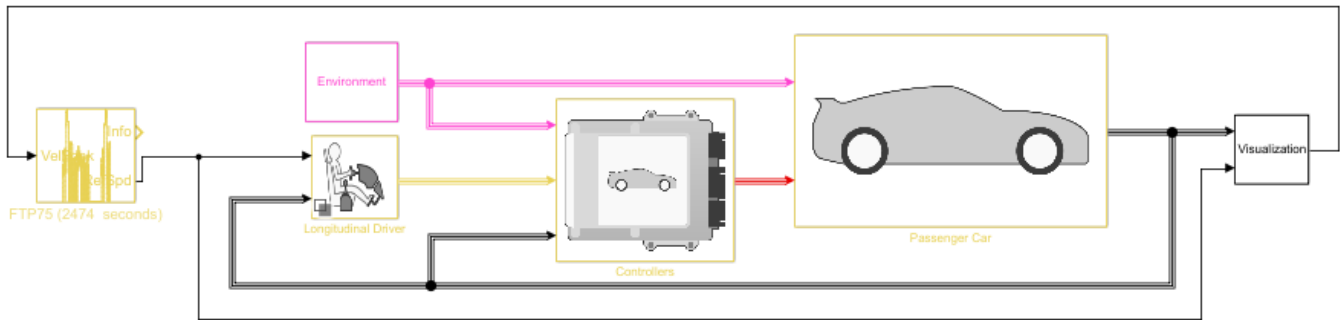
Parameter	Description	Setting	
		EPA Standard ¹	WLTP Tests ²
Speed tolerance	Speed tolerance above the highest point and below the lowest point of the drive cycle speed trace within the time tolerance.	2.0 mph	2.0 km/h
Time tolerance	Time that the block uses to determine the speed tolerance.	1.0 s	1.0 s
Maximum number of faults	Maximum number of faults during the drive cycle.	<i>Not specified</i>	10
Maximum single fault time	Maximum fault duration.	2.0 s	1.0 s
Maximum total fault time	Maximum accumulated time spent under fault condition.	<i>Not specified</i>	<i>Not specified</i>

- 4 Connect the vehicle longitudinal velocity signal to the Drive Cycle Source block VelFdbk input port.
 - a In the Visualization subsystem, connect the longitudinal velocity signal, \dot{x} , to an Outputport named \dot{x} _output.
 - b Determine the \dot{x} signal units. To display signal units, on the **Debug** tab, select **Information Overlays > Units**. The \dot{x} signal units are m/s.

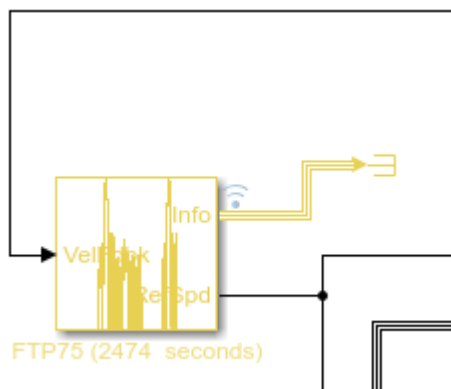
- c Select the `<xdot>` signal line and Enable Data Logging.



- d On the top level of the model, connect the Visualization output to the Drive Cycle Source block input.



- 5 Connect the Drive Cycle Source block Info output port to a Terminator port. Enable data logging.

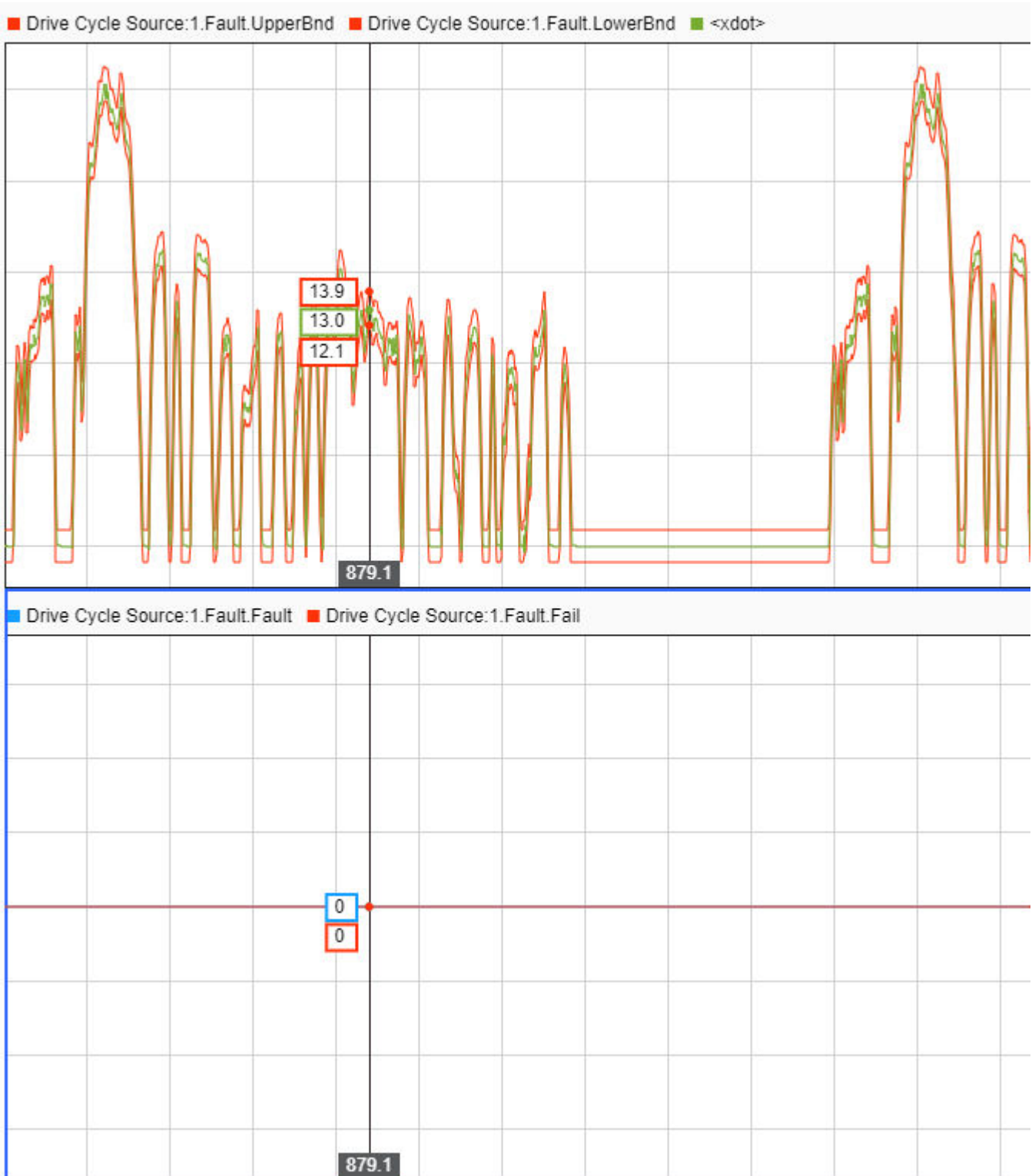


- 6 Save the model and run the simulation.
- 7 To inspect the results, use the Data Inspector. In the Simulink Toolstrip, on the **Simulation** tab, under **Review Results**, click **Data Inspector**.

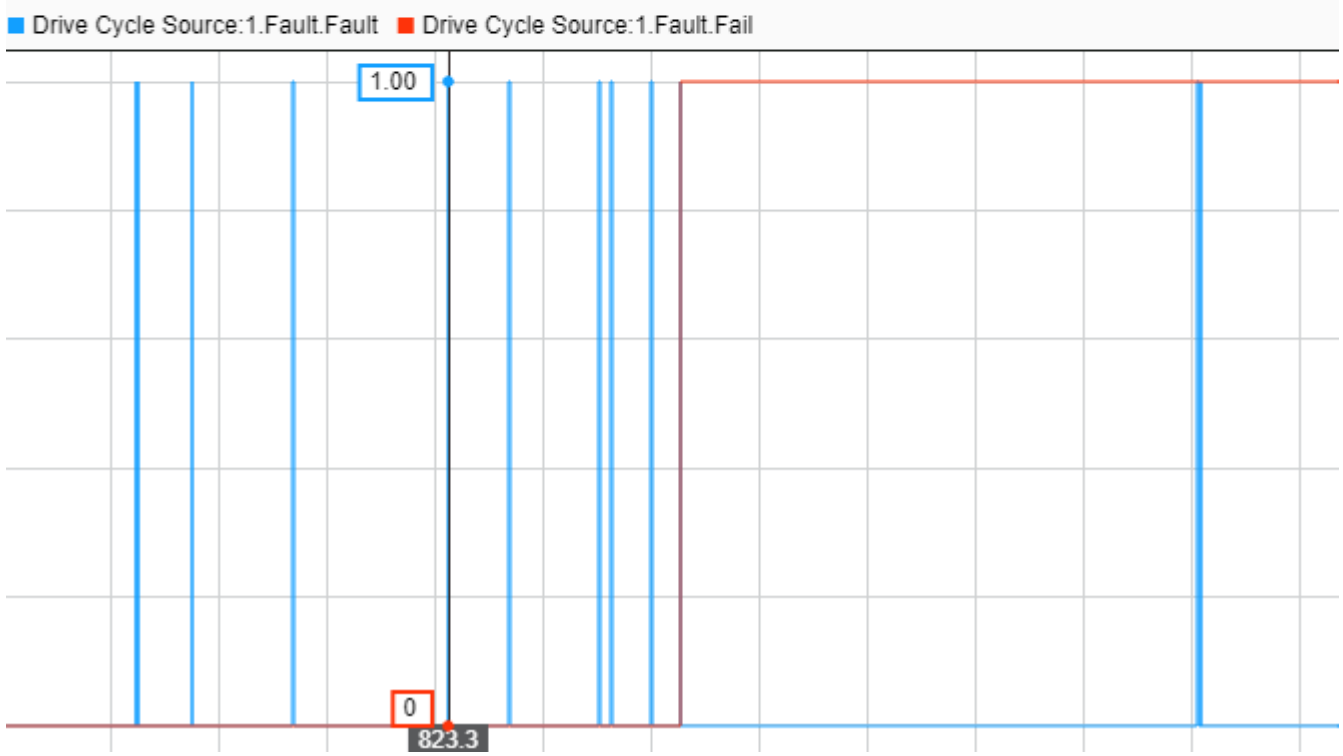
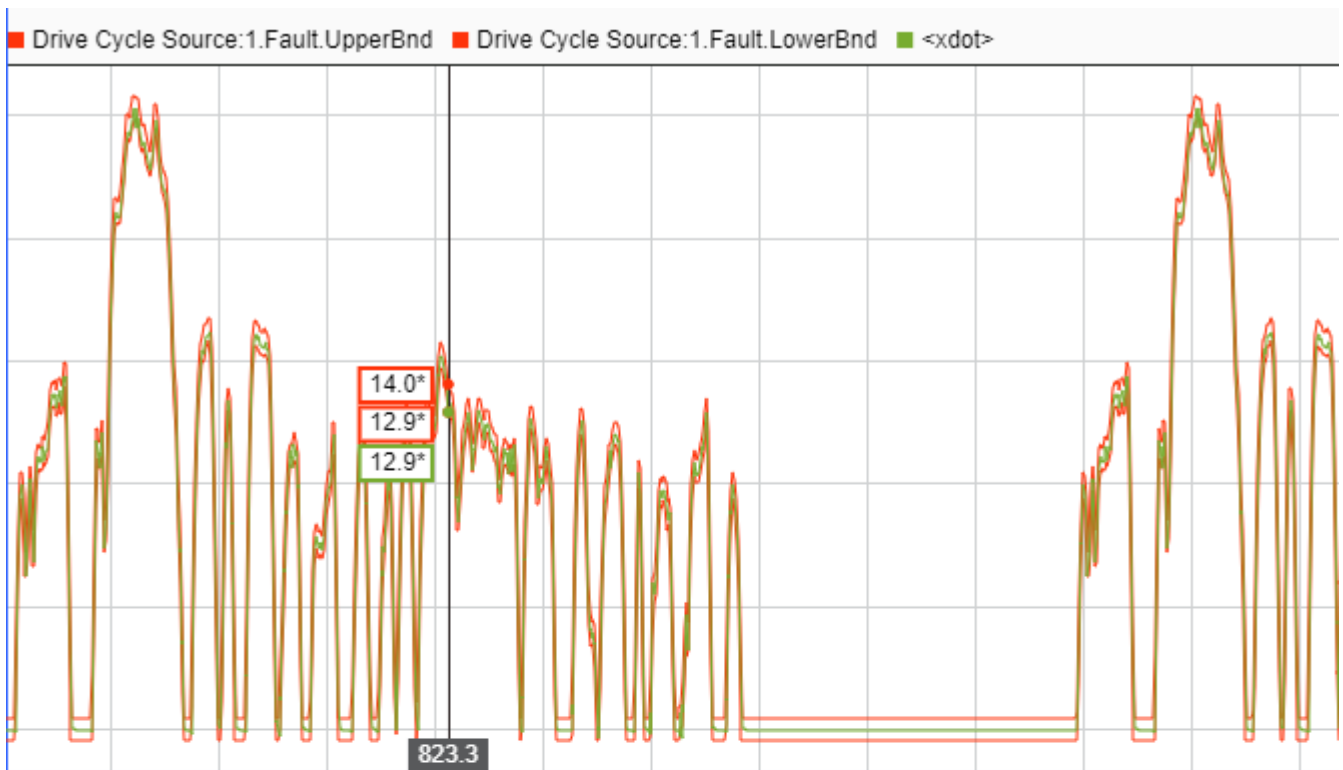
These results indicate that the Drive Cycle Source block did not detect faults or failures during the drive cycle.

- **Fault** — Vehicle speed, `<xdot>`, stayed within the upper and lower bounds of the allowable speed range.

- Fail — Fault conditions did not exceed the maximum number of faults, maximum single fault time, or maximum total fault time.



- 8 In the Drive Cycle Source block, set the **Speed tolerance** parameter to a tighter tolerance, for example 1 mph. The block calculates new error bounds for the speed.
- 9 Rerun the simulation.
- 10 To inspect the results, use the Data Inspector. These results indicate that the Drive Cycle Source block did detect failures and faults during the drive cycle.
 - **Fault** — Vehicle speed, \dot{x} , did not stay within the upper and lower bounds of the allowable speed range.
 - **Fail** — Fault conditions exceeded the maximum number of faults, maximum single fault time, or maximum total fault time.



References

- [1] Environmental Protection Agency (EPA). *EPA urban dynamometer driving schedule*. 40 CFR 86.115-78, July 1, 2001.
- [2] European Union Commission. "Speed trace tolerances". *European Union Commission Regulation*. 32017R1151, Sec 1.2.6.6, June 1, 2017.

See Also

Drive Cycle Source

More About

- "Explore the Conventional Vehicle Reference Application" on page 3-4
- "Install Drive Cycle Data" on page 5-2

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.

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.

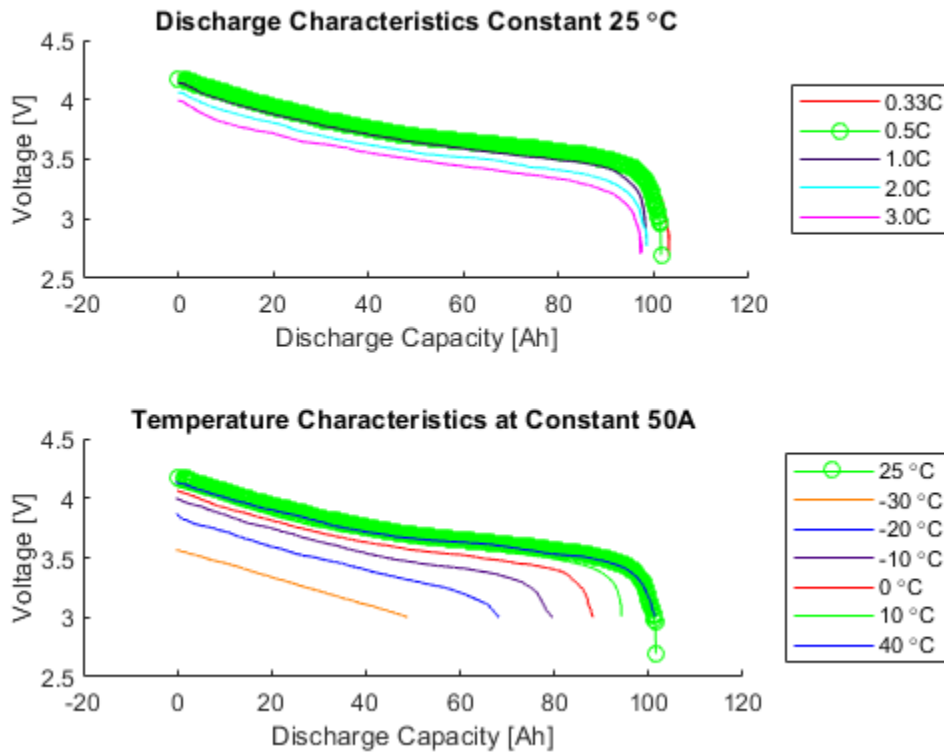
```
exp_data=load('ex_datasheetbattery_liion_100Ah.mat');
```

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
```


Figure 1 - Data Import



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
```

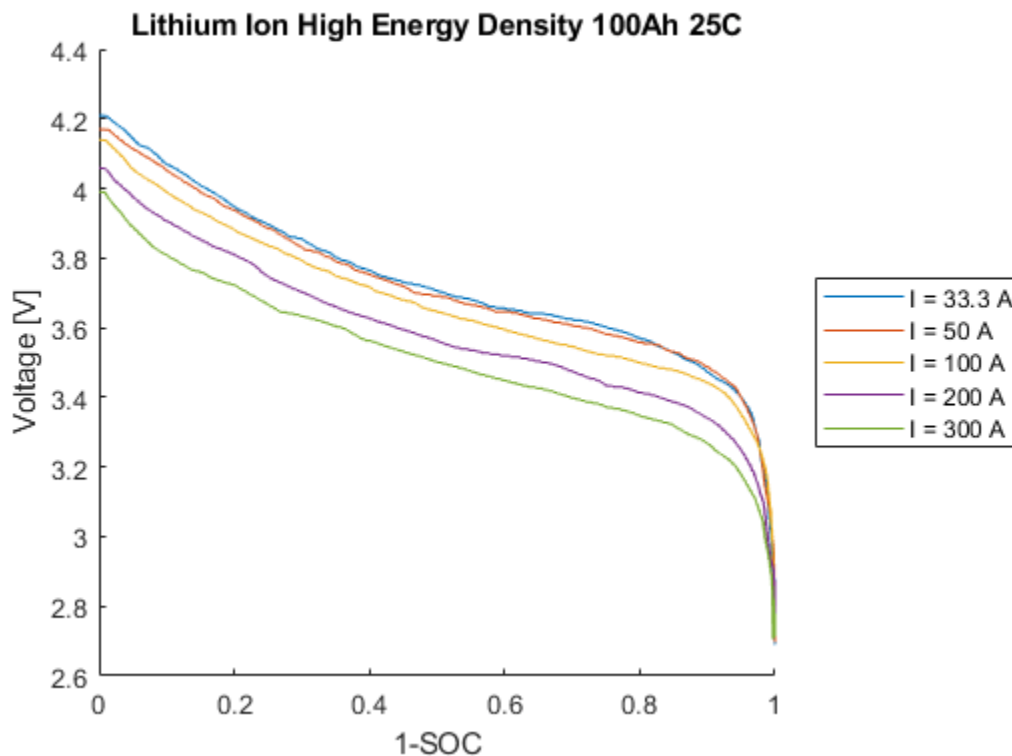
Prepare normalized x axes for each data set and find the actual capacity. x is a structure with as many fields as data sets and values between 0 and 1.

```
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),'smoothingsp')
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),'smoothingsp')
end
```

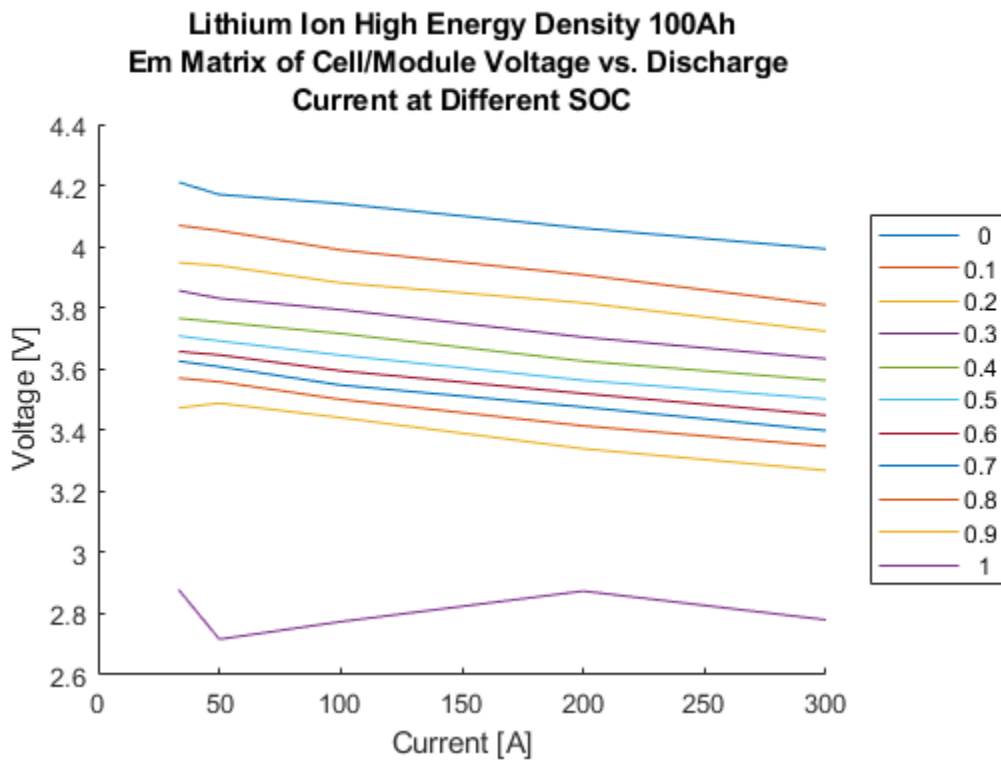
Construct the voltage versus discharge current for different SOC levels. Em_MAT is a matrix with the SOC in rows and the current in columns.

```
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)','poly1');
end
```

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

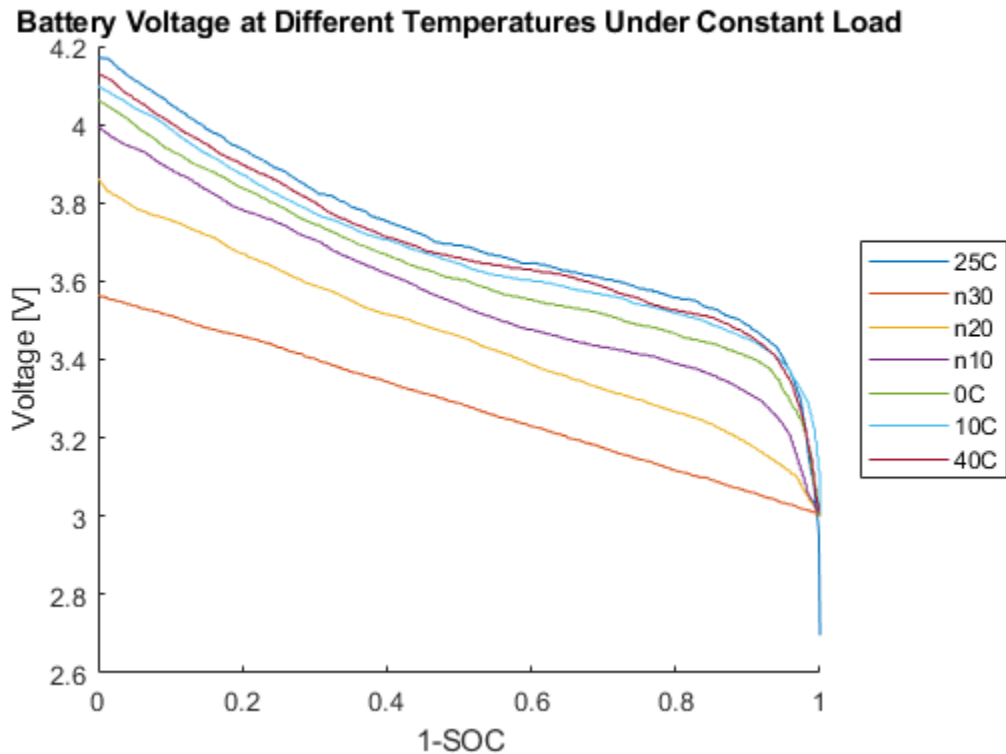
```
Em = [];
for i=1:length(SOC_LUT)
    % Em = f(0)
    Em = [Em fitSOC.(['SOC' num2str(i))](0)];
end
Em = Em';
```

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}](SOC_LUT);
    % Calculate R0(SOC,T) assuming linear behavior R0 = DeltaV / I
    R0(['temp' temperature_label{i}]) = (Em - voltVsSOC(['temp' temperature_label{i}]))./current;
    % 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.

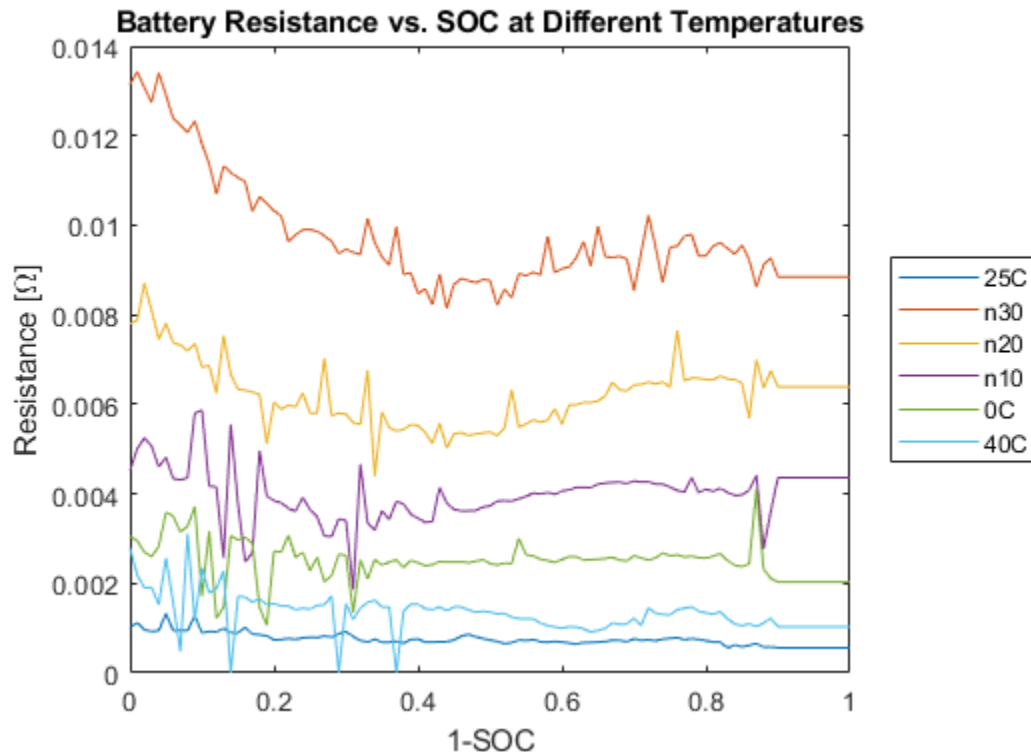
```
R0_LUT = max(R0_LUT,0);
T_LUT = 273.15 + temperature(ref_curr & ~val_exp);
[T_LUT1,idx] = sort(T_LUT);
```

```
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.

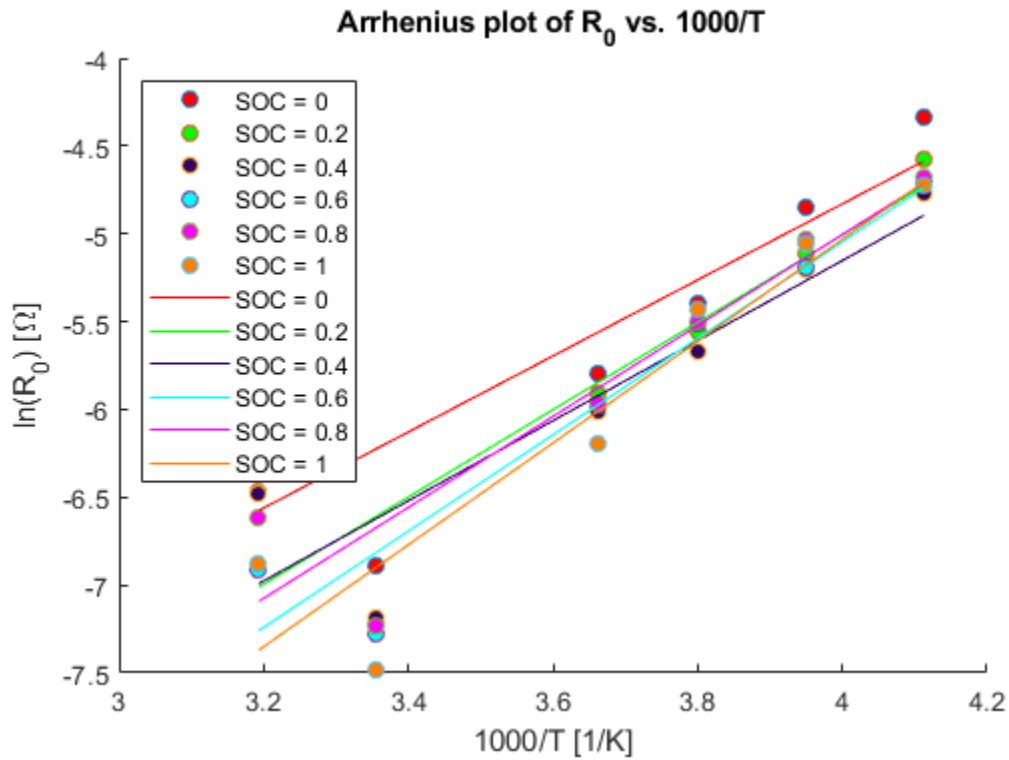
To determine the curve-fit prediction for the Arrhenius behavior, examine the activation energy, E_a . Obtain the activation energy via the slope of the internal resistance, R_o , versus $1000/T$ curve for different SOC. The slope equals the activation energy, E_a , divided by the universal gas constant, 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
Ea = 17.9958    20.669    18.9557    22.8107    21.5289    24.0987 kJ/mol
Ea for electrolyte transport in Li ion battery = 20 kJ/mol
```

Figure 6 - Arrhenius Behavior



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;
[SOC_LUT_index, ~] = find(abs(SOC_LUT-SOCbkpts)<0.001);

for i=find(ref_curr & ~val_exp)
    R0_LUT_bkpts = [R0_LUT_bkpts R0_LUT(SOC_LUT_index,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

$\ln(\text{Battery Resistance})$ as a Function of SOC and Temperature

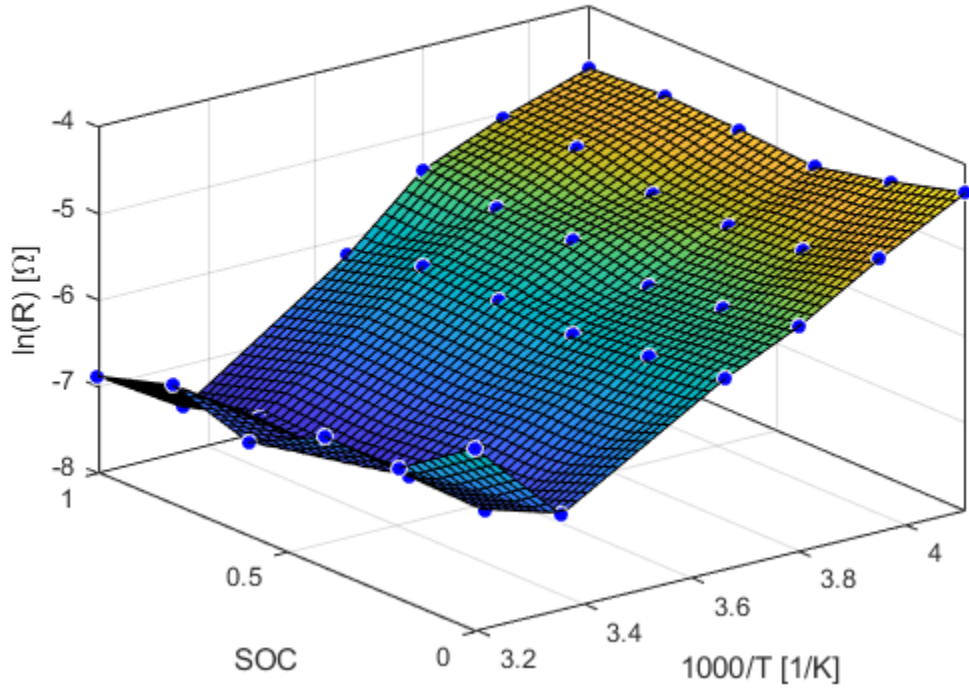


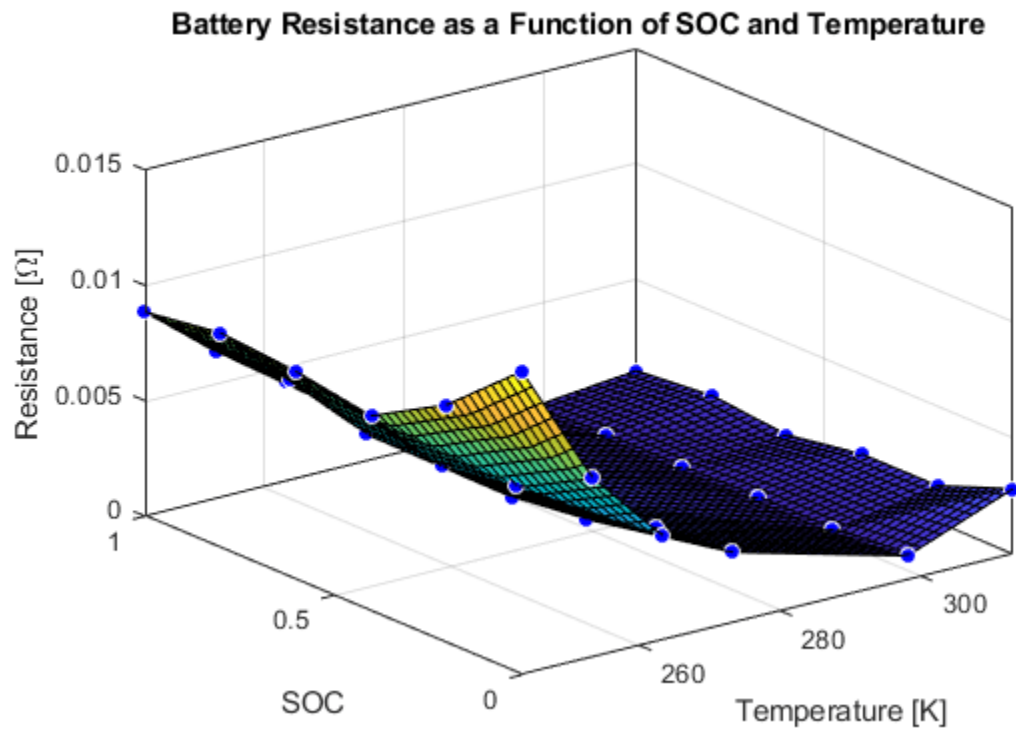
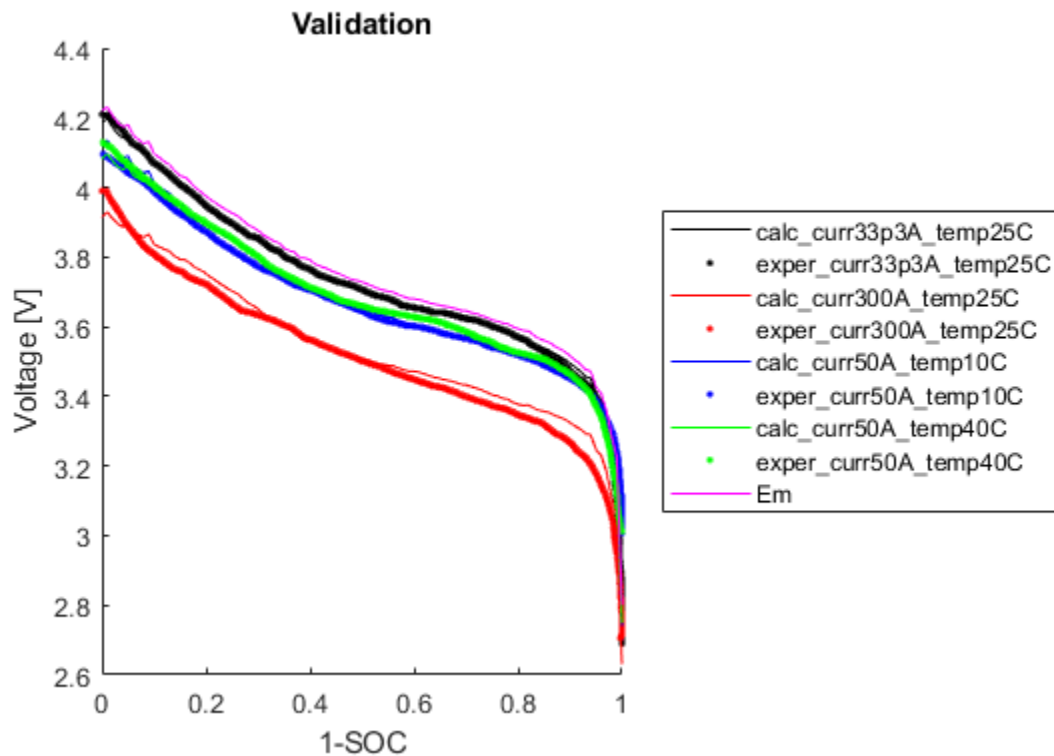
Figure 8 - Surface Fit of Battery Resistance**Step 8: Validate Battery Model Fit**

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=fLipud(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 colorV f9 p10 p9;  
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 SOC_LUT_index;
```

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

Battery.Metadata | Battery.Parameters | Battery.Pulse | Battery.PulseSequence | 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 6-15	Load and preprocess time series battery discharge voltage and current data.	None
“Step 2: Determine the Number of RC Pairs” on page 6-17	Determine the number of necessary time constants (TC) for estimation.	Curve Fitting Toolbox
“Step 3: Estimate Parameters” on page 6-18	<p>For battery discharge data, estimate and optimize:</p> <ul style="list-style-type: none"> • Open-circuit voltage, E_m • Series resistance, R_0 • RC pair(s) time constant(s), τ • RC pair(s) resistance(s), R_x <p>Use a model that exercises the Estimation Equivalent Circuit Battery block.</p>	Curve Fitting Toolbox, Parallel Computing Toolbox, Optimization Toolbox, and Simulink Design Optimization

Workflow	Description	Additional MathWorks Tooling
“Step 4: Set Equivalent Circuit Battery Block Parameters” on page 6-24	Set these block parameters: <ul style="list-style-type: none"> • Open circuit voltage table data • Series resistance table data • State of charge breakpoints • Temperature breakpoints • Battery capacity table • Network resistance table data • Network capacitance table data 	None

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:

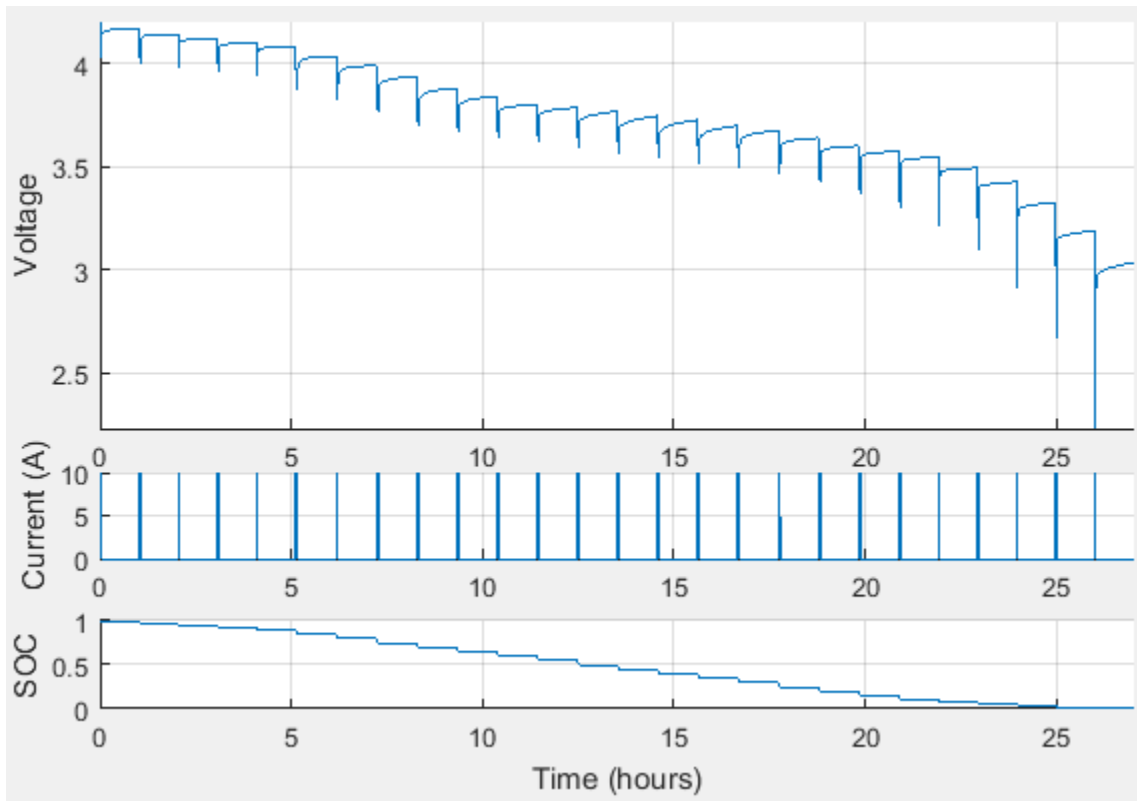
- Time series consisting of current and voltage from an experimental pulse discharge. For each experimental data set, the temperature is constant. The sample rate should be a minimum of 1 Hz, with an ideal rate at 10 Hz. This table summarizes the accuracy requirements.

Measurement	Accuracy	Ideal
Voltage	±5 mV	±1 mV
Current	±100 mA	±10 mA
Temperature	±1 °C	±1 °C

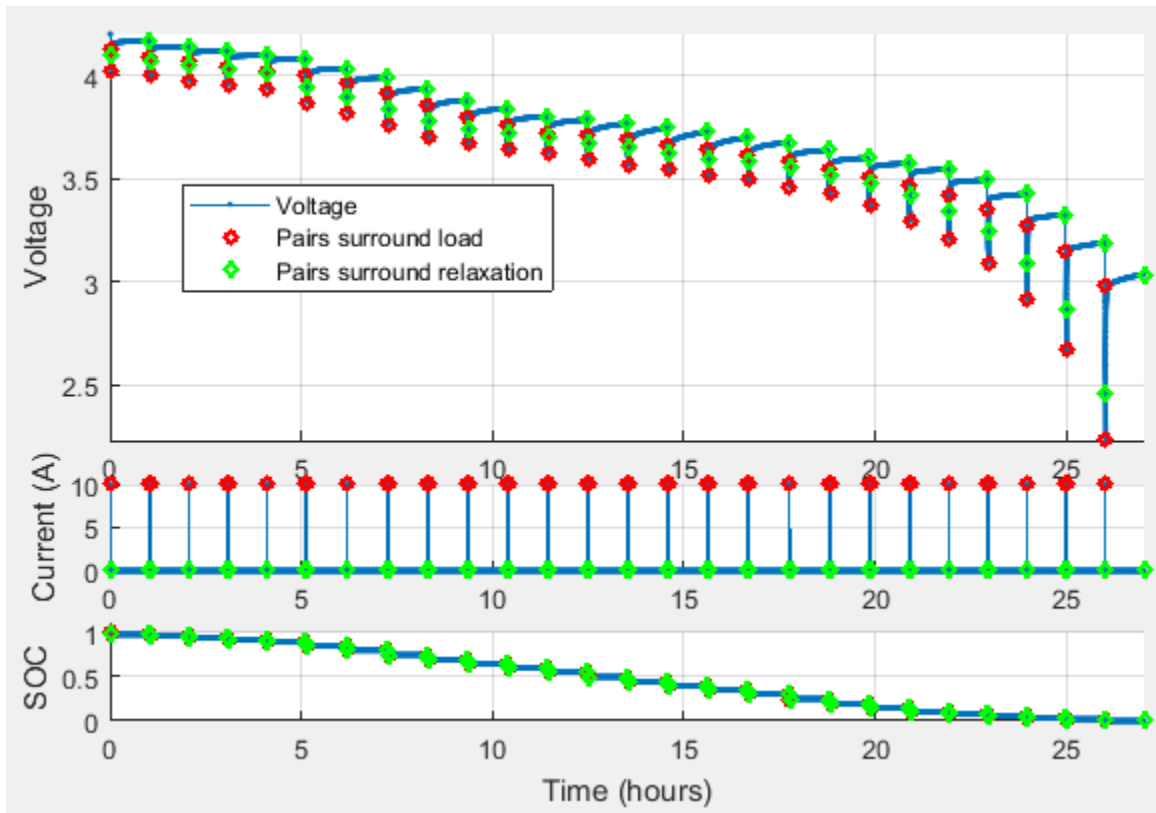
- Change in SOC for each pulse should not be greater than 5%.
- Data collection at high or low SOC might need modification to ensure safety.
- Sufficient relaxation time after each pulse to ensure battery approaches steady-state voltage.

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



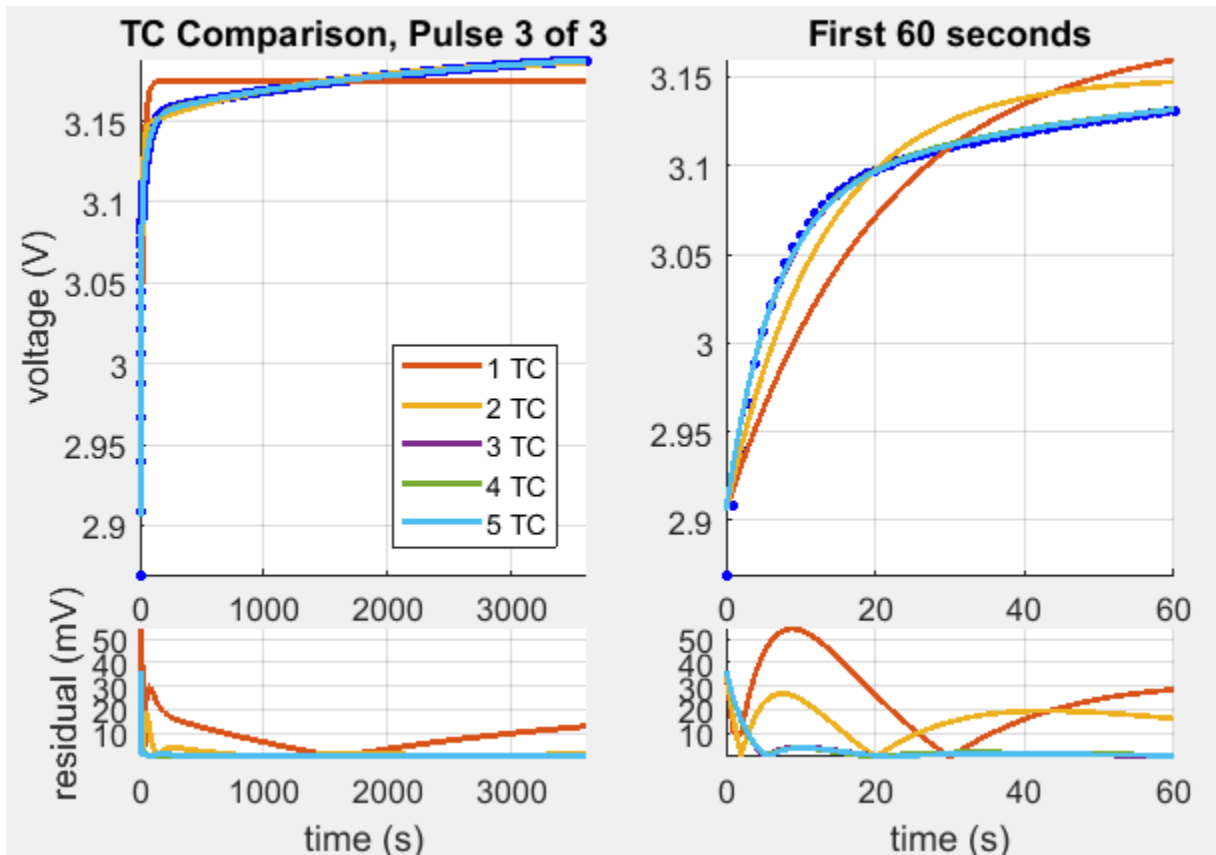
Pulse Identification

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.



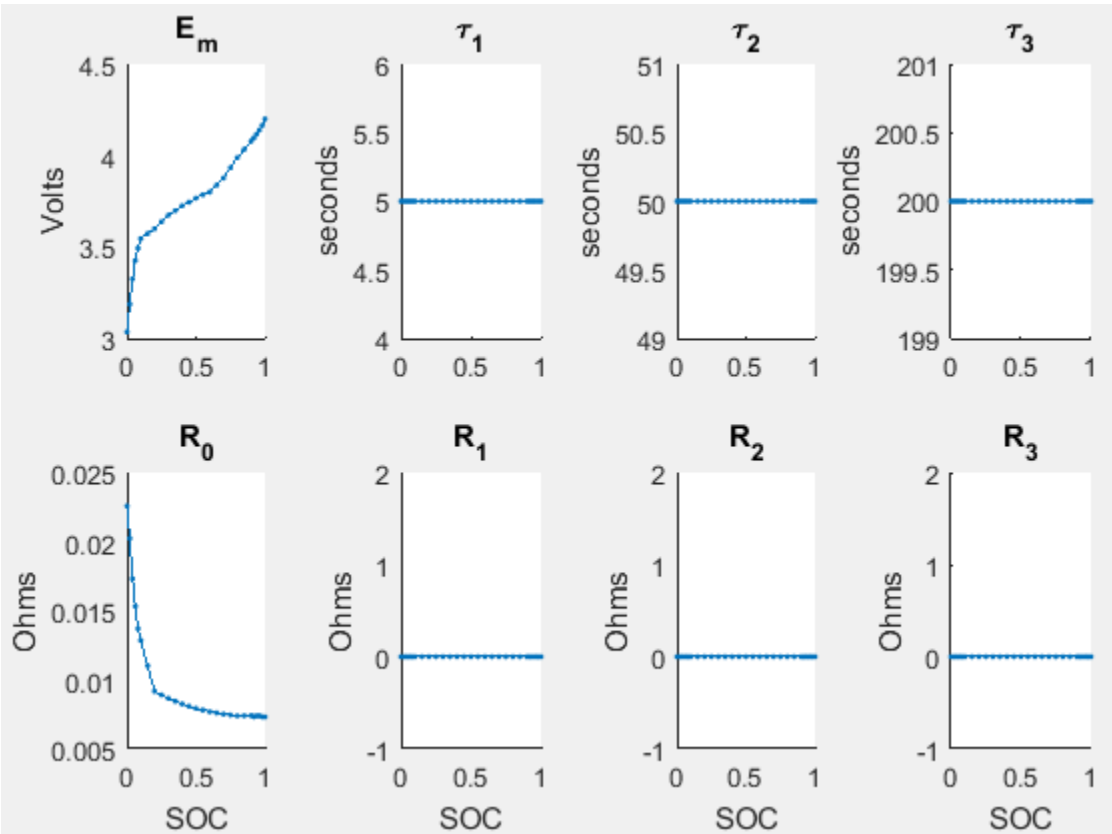
TC Comparison, Pulse 3 of 3

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 E_m and R_0

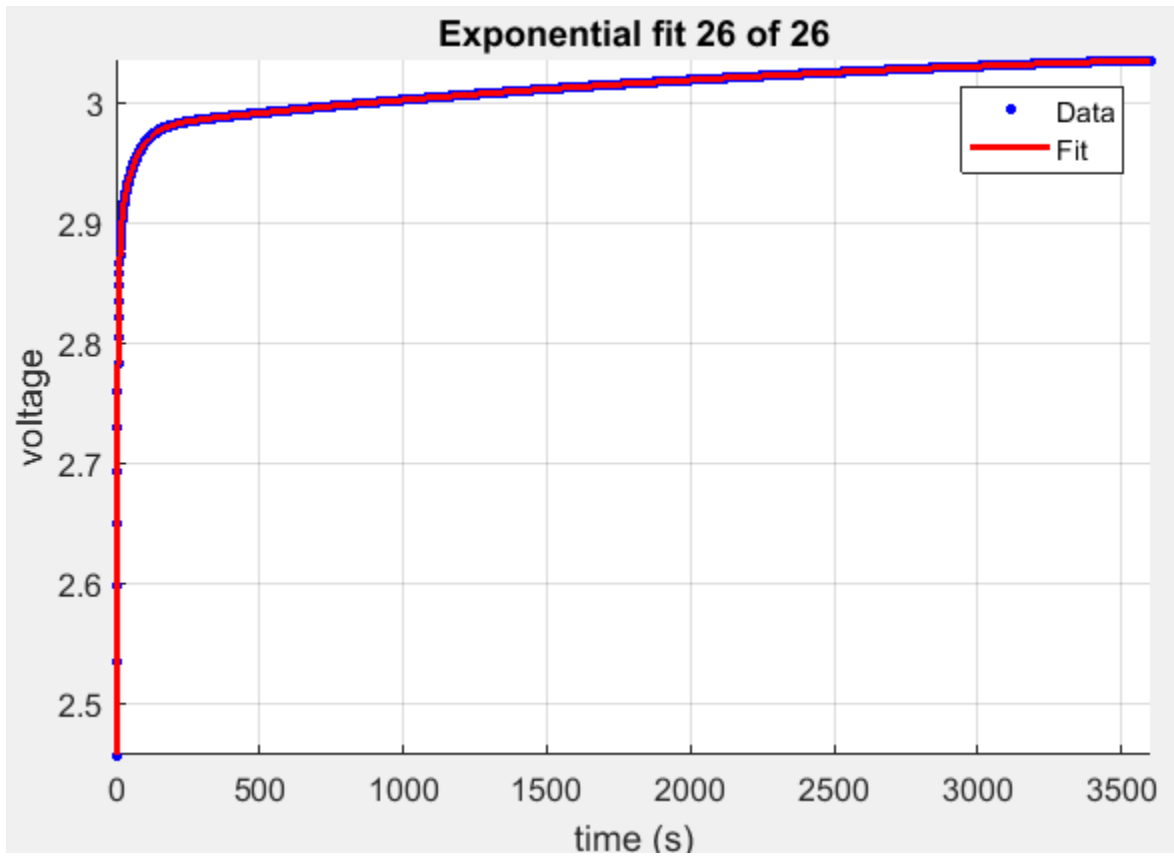
Inspect the voltage immediately before and after the current is applied and removed at the start and end of each pulse. The estimation technique uses the voltage for a raw calculation to estimate the open-circuit voltage (E_m) and the series resistance (R_0).



Parameter Tables

Estimate Tau

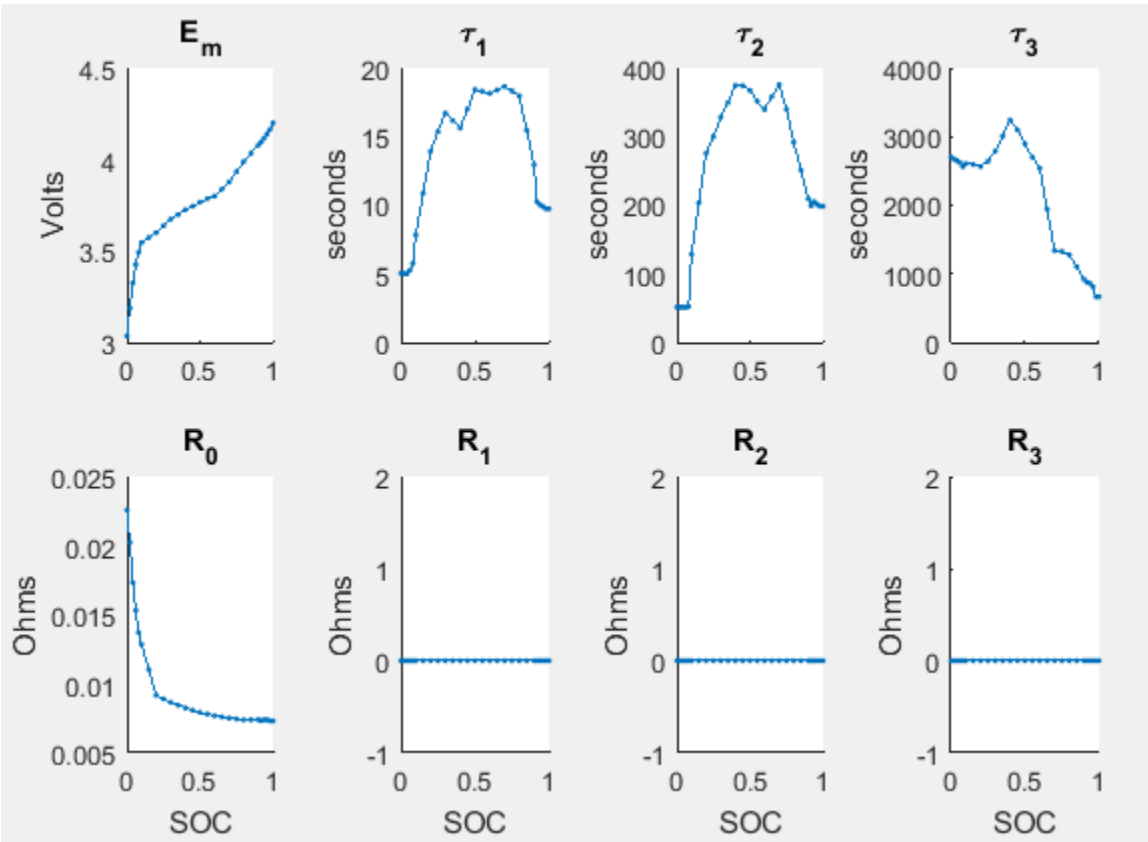
Use a curve-fitting technique on the pulse relaxation to estimate the RC time constant (Tau) at each SOC.



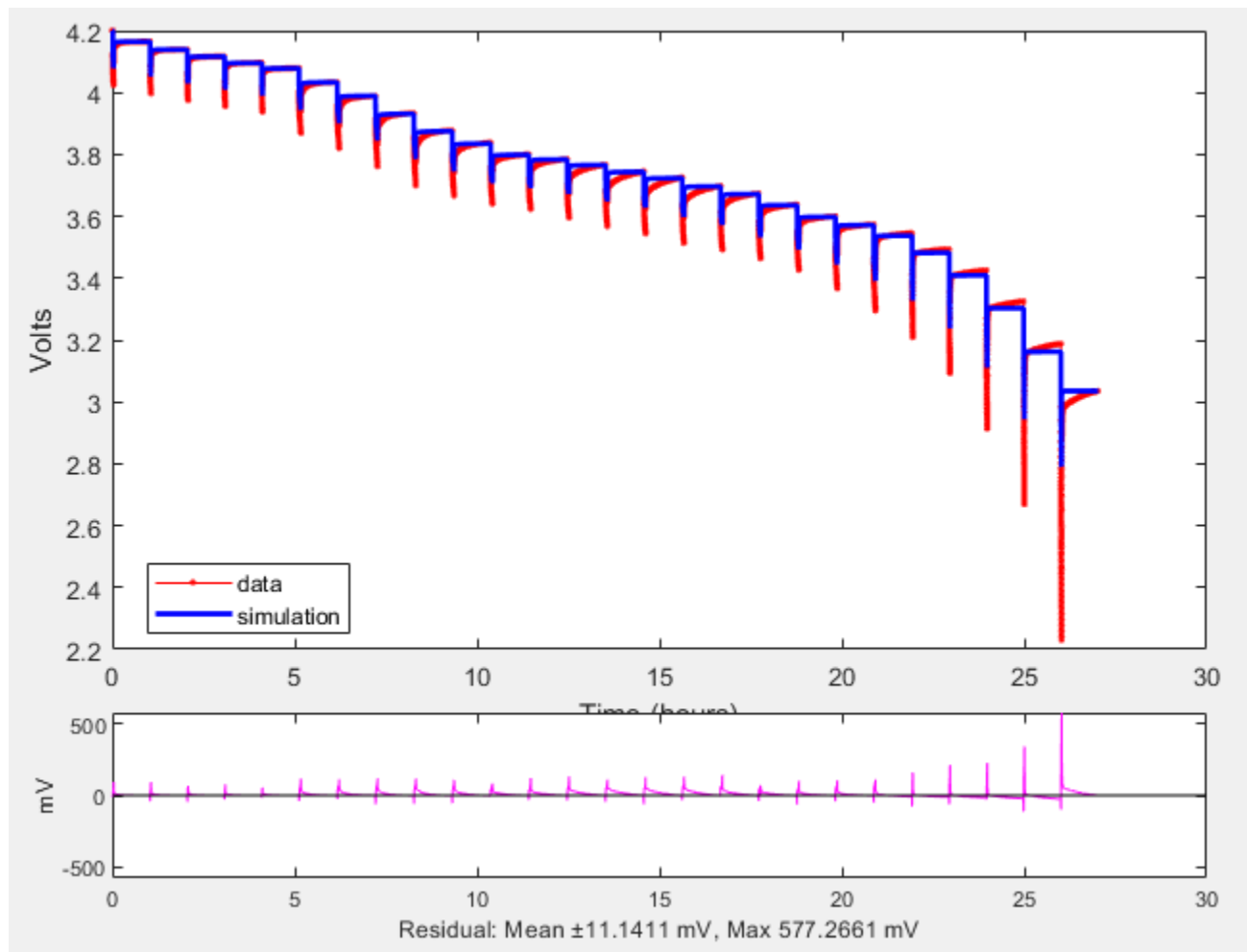
Relaxation Tau Fit

Plot Estimates

Plot the parameter and pulse sequence data and simulation comparison.



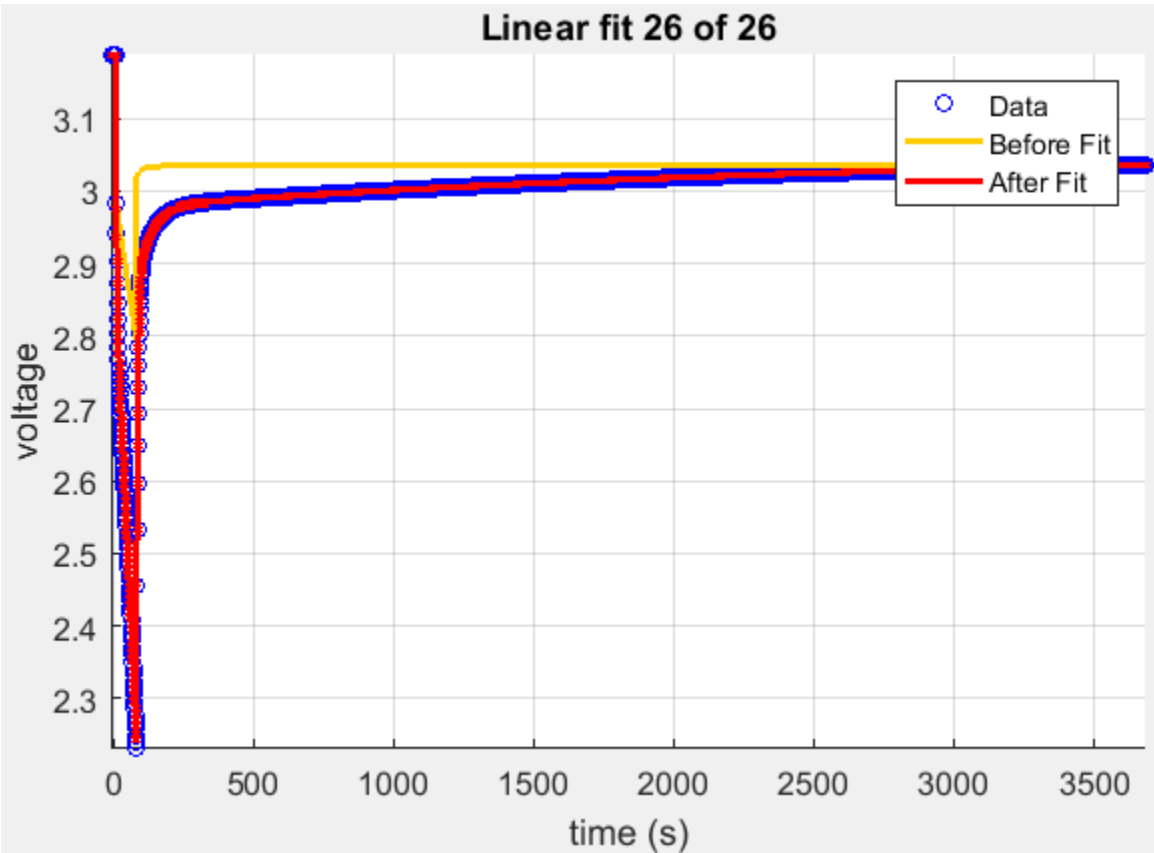
Parameter Tables



Pulse Sequence

Identify Parameters and Set Initial Values

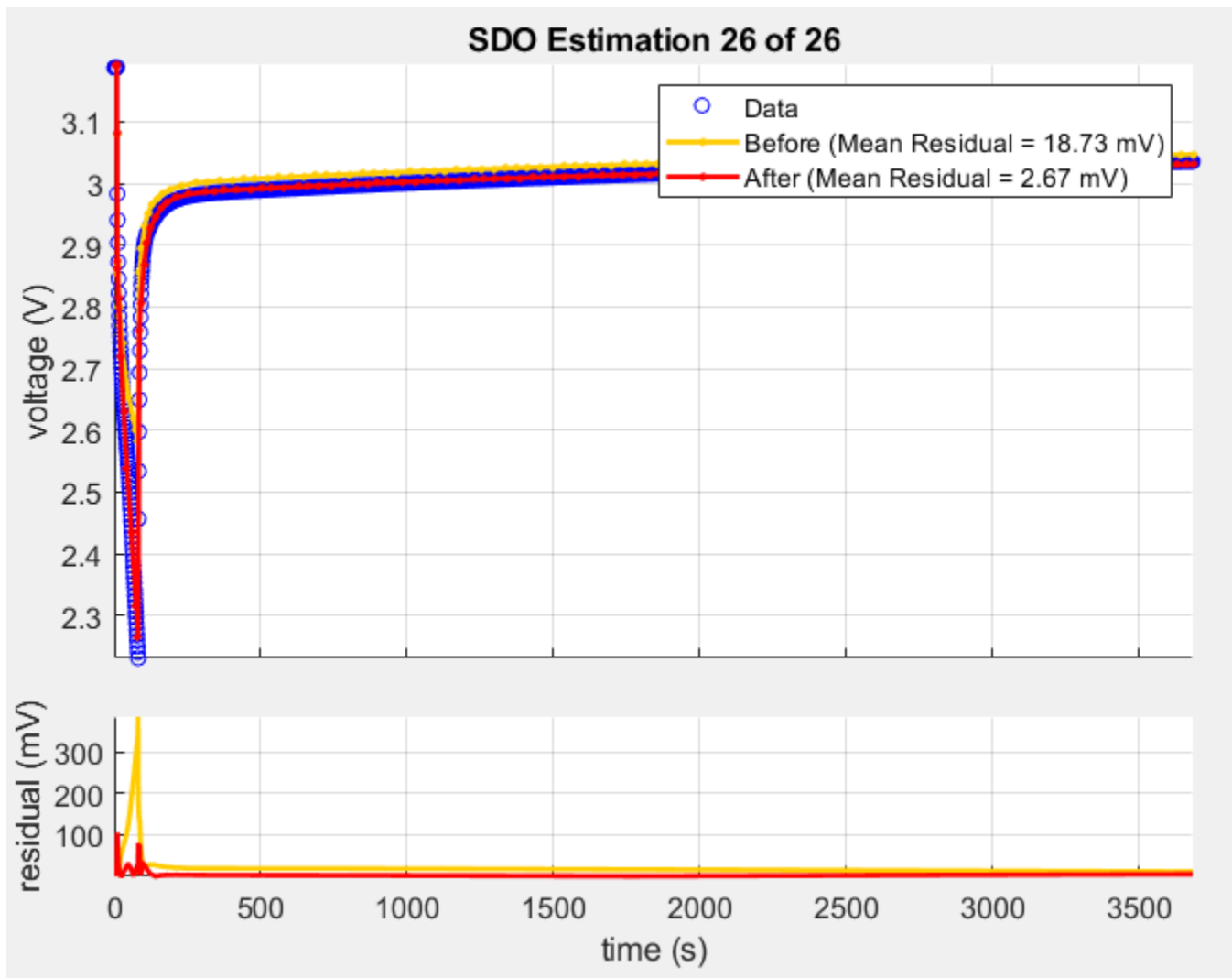
Identify parameters and set the initial values using a linear system approach, pulse-by-pulse.



Linear Fit

Optimize Estimates

Optimize the E_m , R_0 , R_x , and τ 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;

Parameter	Example Value
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

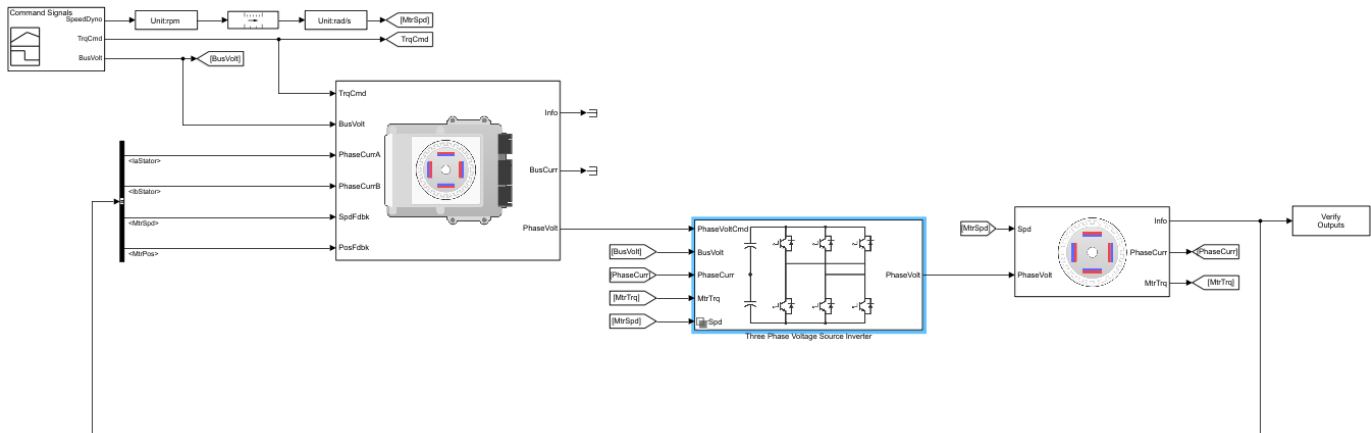
Equivalent Circuit Battery | Estimation Equivalent Circuit Battery

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"> • Corresponding d-axis current reference, id_ref • Corresponding q-axis current reference, iq_ref • Vector of speed breakpoints, wbp • Vector of torque breakpoints, tbp 	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"> • Motor efficiency • Maximum torque per ampere (MTPA) • Flux weakening 	“Generate Current Controller Parameters” on page 6-28
	Motor parameters: <ul style="list-style-type: none"> • Vector of d-axis current breakpoints, id_index • Vector of q-axis current breakpoints, iq_index • Corresponding d-axis flux, $lambda_d$ • Corresponding q-axis flux, $lambda_q$ 	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 6-49
Flux-Based PMSM	Parameters: <ul style="list-style-type: none"> • Vector of d-axis flux, $flux_d$ • Vector of q-axis flux, $flux_q$ • Corresponding d-axis current, id • Corresponding q-axis current, iq 	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 6-53

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.
- [3] Ottosson, J., M. Alakula. "A compact field weakening controller implementation." *International Symposium on Power Electronics, Electrical Drives, Automation and Motion*, July, 2006.

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 6-29	<p>Collect the nonlinear motor flux data from dynamometer testing or finite element analysis (FEA). For this example, file <code>PMSMEfficiencyData.xlsx</code> contains the data that you need:</p> <ul style="list-style-type: none"> • Total flux, Ψ_{total}, in Wb • Allowed flux, Ψ_{max}, in Wb • d-axis flux, Ψ_d, in Wb • q-axis flux, Ψ_q, in Wb • d-axis current, I_d, in A • q-axis current, I_q, in A • Current magnitude, I_s, in A • Motor torque, T_e, in N·m • Motor speed, n, in rpm 	N/A
“Model Motor Data” on page 6-30	<p>Use a one-stage model to fit the data. Specifically:</p> <ul style="list-style-type: none"> • Import data • Filter data • Fit model 	Model-Based Calibration Toolbox

Workflow Steps	Description	MathWorks Tooling
“Generate Calibration” on page 6-34	Calibrate and optimize the data using objectives and constraints. Specifically: <ul style="list-style-type: none"> • Create functions. • Create tables from model. • Run an optimization. • Generate and fill optimized current controller calibration tables that are functions of motor torque and motor speed. 	Model-Based Calibration Toolbox
“Set Block Parameters” on page 6-47	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 total flux, maximum flux, and current magnitude:

$$\Psi_{total} = \sqrt{\Psi_d^2 + \Psi_q^2}$$

$$i_s = \sqrt{i_d^2 + i_q^2}$$

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

$$\Psi_{max} = \frac{V_{dc}}{\sqrt{3}\omega_e}$$

The equations use these variables:

i_d, i_q	d - and q - axis current, respectively
i_s	Current magnitude
Ψ_d, Ψ_q	d - and q - axis flux linkage, respectively
Ψ_{total}, Ψ_{max}	Total and allowed flux, respectively
ω_e	Electrical motor angular speed, rad/s
n	Motor speed, rpm
V_{dc}	Inverter bus voltage
P	Number of pole pairs

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

- Total flux, Ψ_{total} , in Wb
- Allowed flux, Ψ_{max} , in Wb
- d -axis flux, Ψ_d , in Wb
- q -axis flux, Ψ_q , in Wb
- d -axis current, I_d , in A
- q -axis current, I_q , in A
- Current magnitude, I_s , in A
- Motor torque, T_e , in N·m
- Motor speed, n , in rpm

For this example:

- Pole pairs, P , is 4
- Inverter bus voltage, V_{dc} , is 500

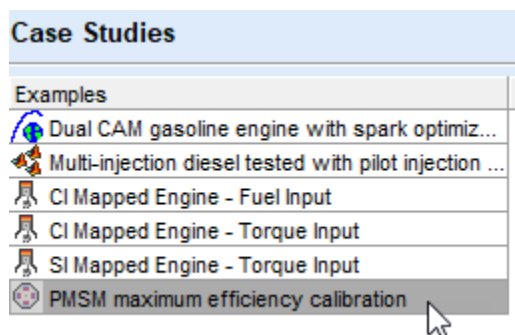
the data file `matlab\toolbox\mbc\mbctraining\PMSMEfficiencyData.xlsx` contains the motor flux data.

Model Motor Data

To model the motor data, use the **MBC Model Fitting** app to import, filter, and fit the data with a point-by-point model. For this example, the data file `PMSMEfficiencyData.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.

Since there is a simple relationship between the d - and q -axis currents for fixed torque-speed operating points, the point-by-point model provides an accurate fit.

For comparison, the PMSM maximum efficiency calibration case study contains the model fit.



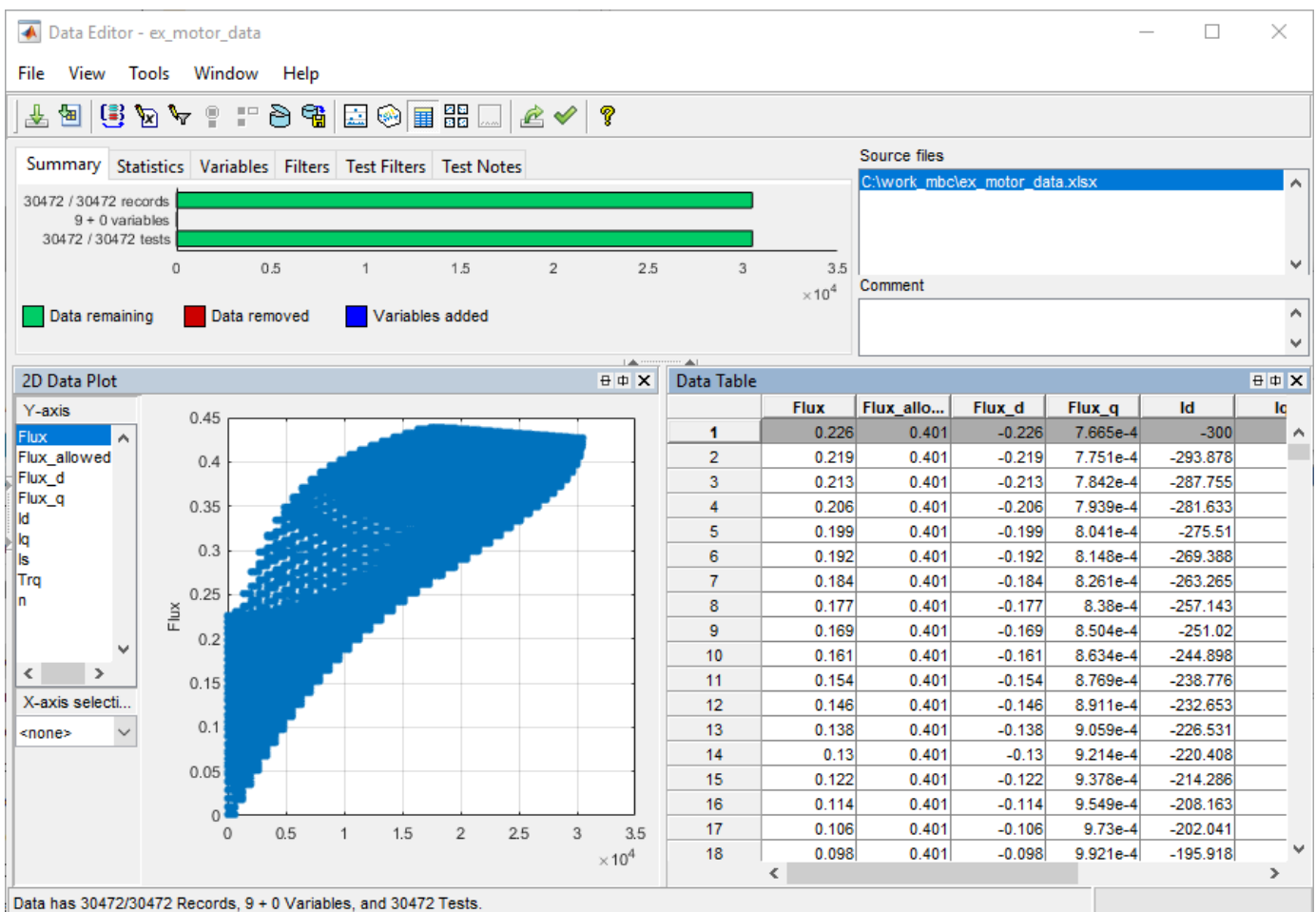
Import Data

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

- Total flux, Ψ_{total} , in Wb
- Allowed flux, Ψ_{max} , in Wb
- d -axis flux, Ψ_d , in Wb

- q -axis flux, Ψ_q , in Wb
- d -axis current, I_d , in A
- q -axis current, I_q , in A
- Current magnitude, I_s , in A
- Motor torque, T_e , in N·m
- Motor speed, n , in rpm

- 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 `PMSMEfficiencyData.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 include only flux and current magnitudes that are less than a specified threshold. Specifically:

- Current magnitude, I_s , less than or equal to 300 A.
- Total flux, Ψ_{total} , less than or equal to allowed flux Ψ_{max}

1 In the Data Editor, select **Tools > Filters** to open the **Filter Editor**. Create these filters:

- Is \leq 300
- Flux \leq Flux_allowed

Filter Expression	Results
Is \leq 300	Filter successfully applied : 4121 records excluded.
Flux \leq Flux_allowed	Filter successfully applied : 22621 records excluded.

Define Test Groupings

For point-by-point models, you need to define test groups. In the example, define groups for motor torque and speed. Set the tolerances to so that Model-Based Calibration Toolbox groups small variations in torque and speed at the same operating point.

- 1 In the Data Editor, select **Tools > Test Groups** to open the **Define Test Groupings** dialog box. Create groups for the motor torque and speed.
- 2 Set these tolerances:
 - Motor torque, Trq, to 1.000
 - Motor speed, n, to 10.000

3 In the Data Editor, select **File > Save & Close**. Accept the changes to the data.

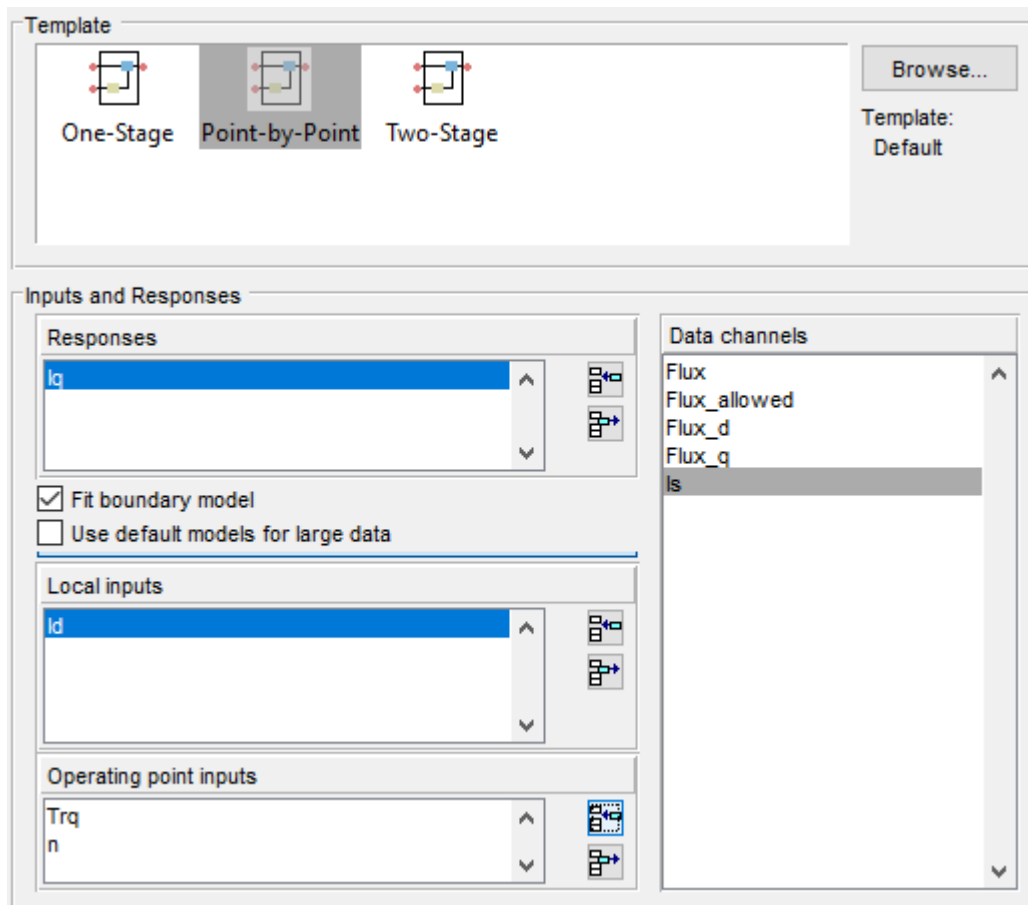
Fit Model

Fit the data to a point-by-point model with these responses, local inputs, and operating points:

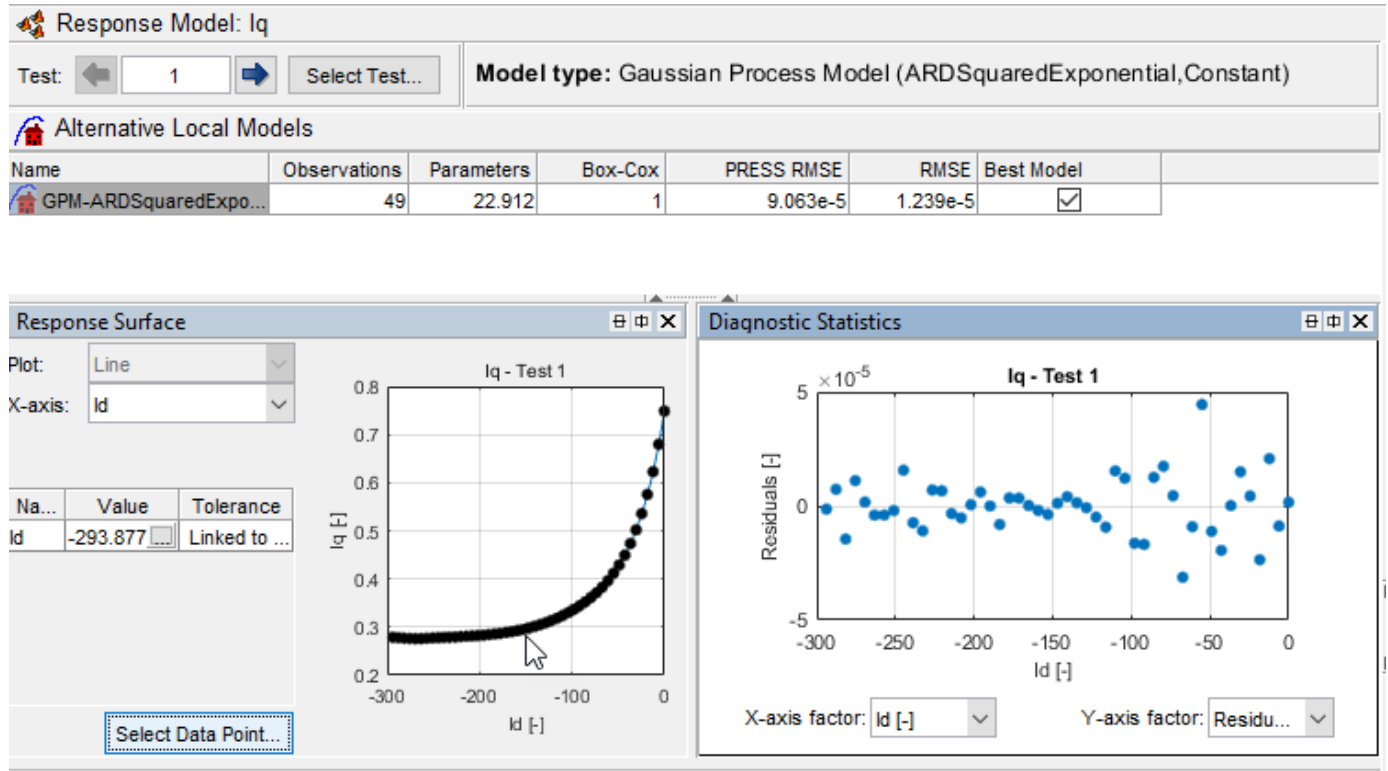
- Responses
 - q -axis current, I_q , in A
- Local inputs
 - d -axis current, I_d , in A
- Operating points
 - Motor speed, n , in rpm
 - Electromagnetic motor torque, T_e , in N·m

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

Responses	Local Inputs	Operating Points
Iq	Id	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 I_q . The Model Browser displays information that you can use to determine the accuracy of the model fit.
 - In the Model Browser, select I_q . Examine the response surface and diagnostic statistics. These results indicate a reasonably accurate fit. You can browse through each test to examine the response for each torque-speed operating point.









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

Generate Calibration

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

For comparison, the PMSM maximum efficiency calibration case study contains the calibration results.

Case Studies	
Examples	
	Dual CAM gasoline engine with spark optimiz...
	Multi-injection diesel tested with pilot injection ...
	CI Mapped Engine - Fuel Input
	CI Mapped Engine - Torque Input
	SI Mapped Engine - Torque Input
	PMSM maximum efficiency calibration

Import Models and Create Functions

Import models and create the functions to use when you optimize the calibration. In this example, set up functions for:

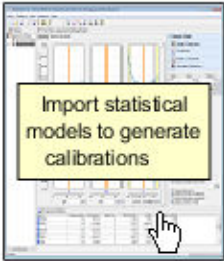
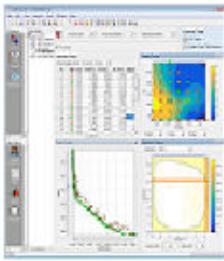
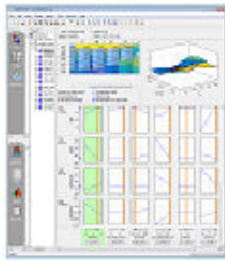
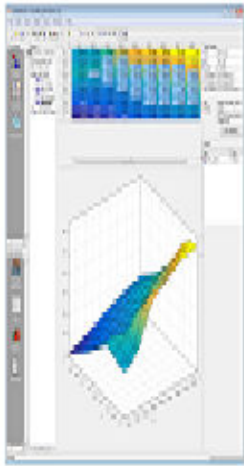

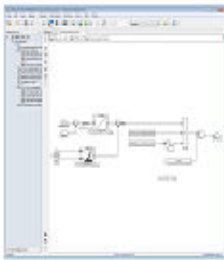
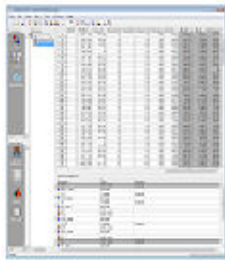
- Current magnitude, I_s
- Torque per amp, TPA

- 1 In MATLAB, on the **Apps** tab, in the **Automotive** group, click **MBC Optimization**.
- 2 In the Cage Browser, select **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.

Create an optimization for a model and use results to fill lookup tables

Import	Use models to generate calibration		Export
 <p>Models</p>	 <p>Optimization</p>	 <p>Lookup Tables and Tradeoff</p>	 <p>Lookup Tables</p>
 <p>Simulink Lookup Tables</p>	 <p>Feature Filling</p>	 <p>Data Set</p>	


- 3 In Import Models, click **OK**. Close the CAGE Import Tool.

Import Models to CAGE

These models will be imported to CAGE when you click OK.

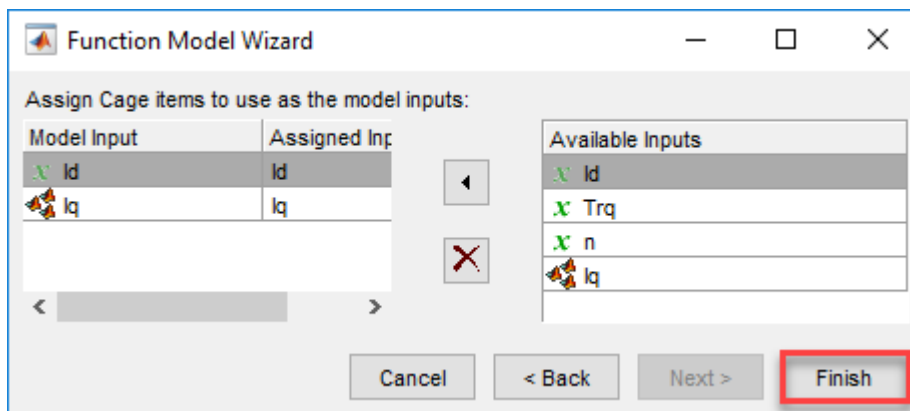
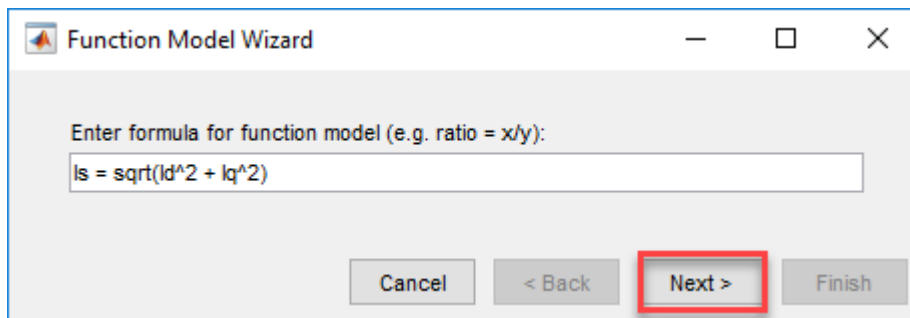
If a model is replaceable in CAGE you can select Replace or Create new in the Action column.

Double-click CAGE Model Name cells to edit names.

Original Name	Action	CAGE Model Name
 Iq	Create new	Iq

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

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



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

Models					
Name	Type	Inputs	Lower Output Limit	Upper Output Limit	Description
Iq	Point-by-point ...	Id, Trq, n	-Inf	Inf	Created by on 28-Mar-2019.
Is	Function model	Id, Iq	-Inf	Inf	$\sqrt{I_d^2 + I_q^2}$
TPA	Function model	Is, Trq	-Inf	Inf	Trq/Is

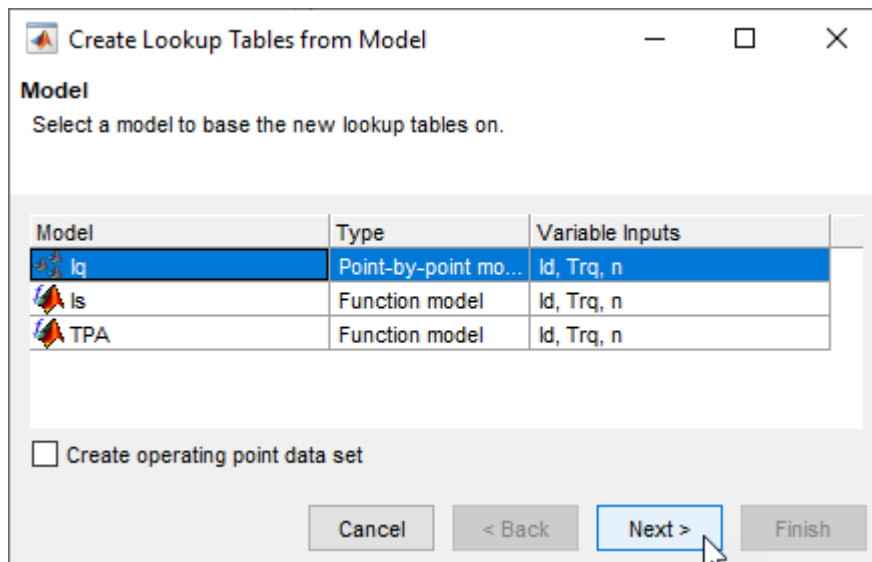
6 Select **File > Save Project**. Save `gs_example.cag` to the work folder.

Create Lookup Tables from Model

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

- d -axis current, I_d , as a function of motor torque, Trq , and motor speed, n .
- q -axis current, I_q , as a function of motor torque, Trq , and motor speed, n .

1 In the Cage Browser, select **Lookup Tables and Tradeoff**. In Create Lookup Tables from Model, select `Iq`. Click **Next**.



2 In the Create Lookup Tables from Model wizard:

- Clear **Use model operating points**.
- Set **Table rows** to 31.
- Set **Table columns** to 29.
- Click **Next**.

Create Lookup Tables from Model

Lookup Table Inputs
Select the lookup table inputs and set up the normalizers to use for all the new lookup tables.

Use model operating points

Rows (Y) input: Trq
Normalizer: <New>
Table rows: 31

Columns (X) input: n
Normalizer: <New>
Table columns: 29

Trq normalizer:

Input	Output
1	0
15.376	1
29.752	2
44.127	3
58.503	4

n normalizer:

Input	Output
1720	0
1884.286	1
2048.571	2
2212.857	3
2377.143	4

Cancel < Back Next > Finish

3 In Create Lookup Tables from Model:

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

Create Lookup Tables from Model

Lookup Tables
Select the items to create lookup tables for. Select the lookup table fill process.

Normalizers: Trq_norm_1,n_norm_1

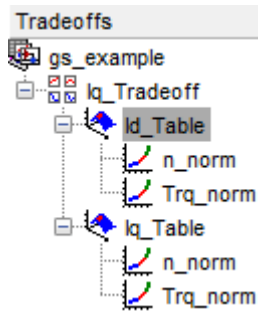
Item	Lookup Table Name	Table Bounds
<input checked="" type="checkbox"/> Id	Id_Table	[-293.878, 0]
<input checked="" type="checkbox"/> Iq	Iq_Table	[-Inf, Inf]

Lookup table fill process

Optimization/Tradeoff
 Models
 None

Cancel < Back Next > Finish

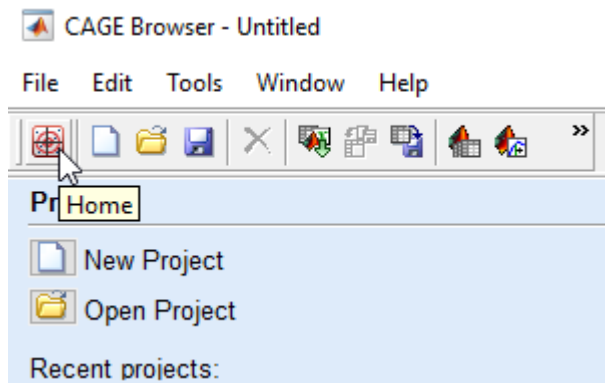
- 4 In the CAGE Browser, examine the tables.



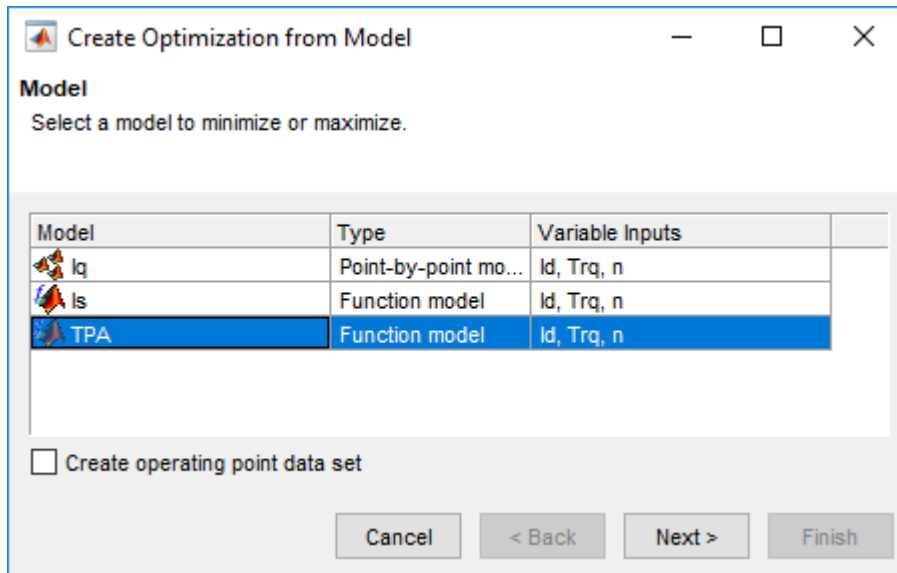
Run Optimization

In this example, run an optimization with these specifications:

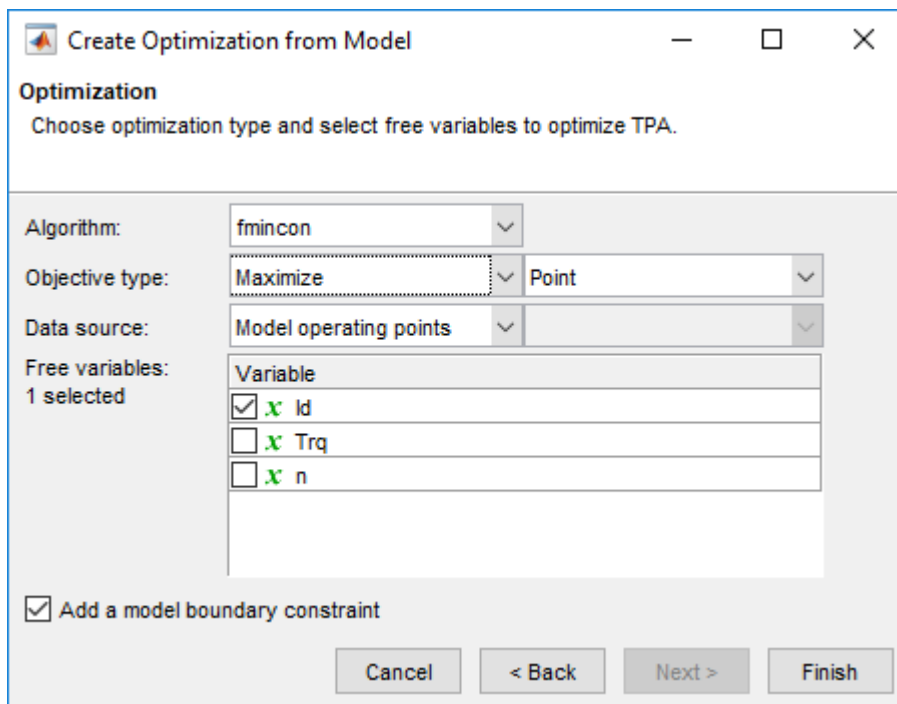
- Current magnitude, I_s , less than or equal to 300 A.
 - Maximizes torque per ampere, TPA .
- 1 On the Cage Browser home, select **Optimization**.



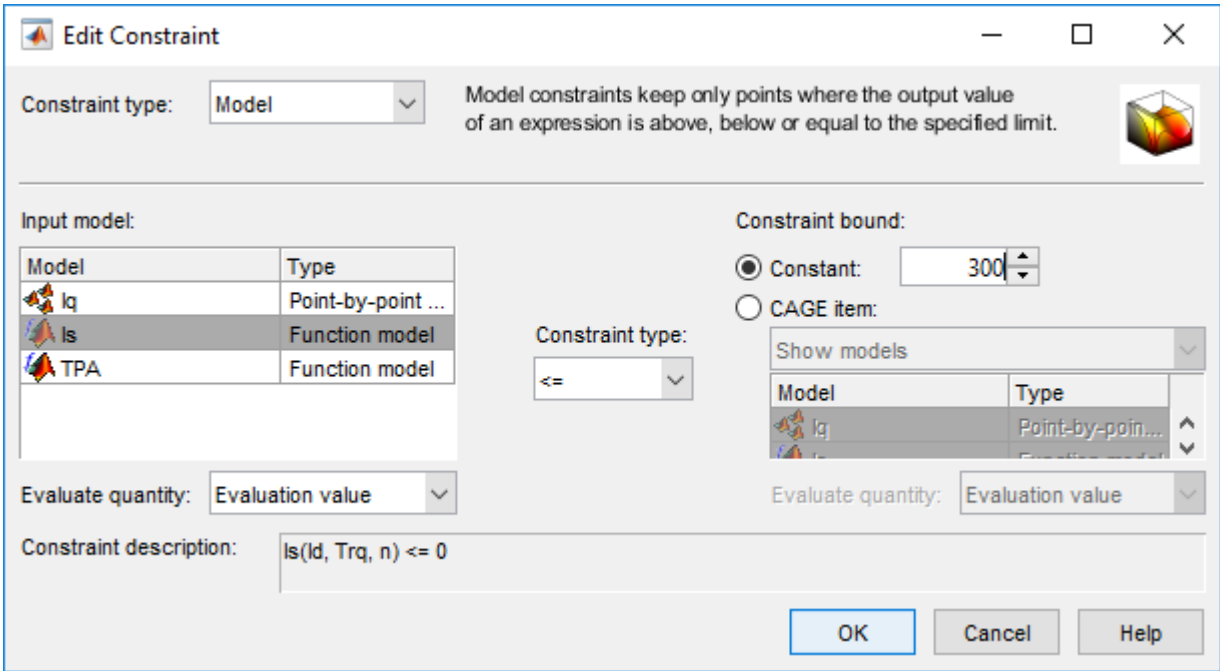
- 2 In Create Optimization from Model, select TPA and **Next**.



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



- 4 Add the optimization constraint for the current magnitude, I_s . In the CAGE Browser, select **Optimization > Constraints > Add Constraints** to open Edit Constraint. Use the dialog box to create a constraint on the current.
- $I_s \leq 300$



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

Objectives		
Name	Description	Type
TPA	TPA(ld, Trq, n)	Maximize

Constraints		
Name	Description	Type
TPA_Boundary	Boundary constraint of TPA(ld, Trq, n)	Model
ls	ls(ld, Trq, n) <= 300	Model

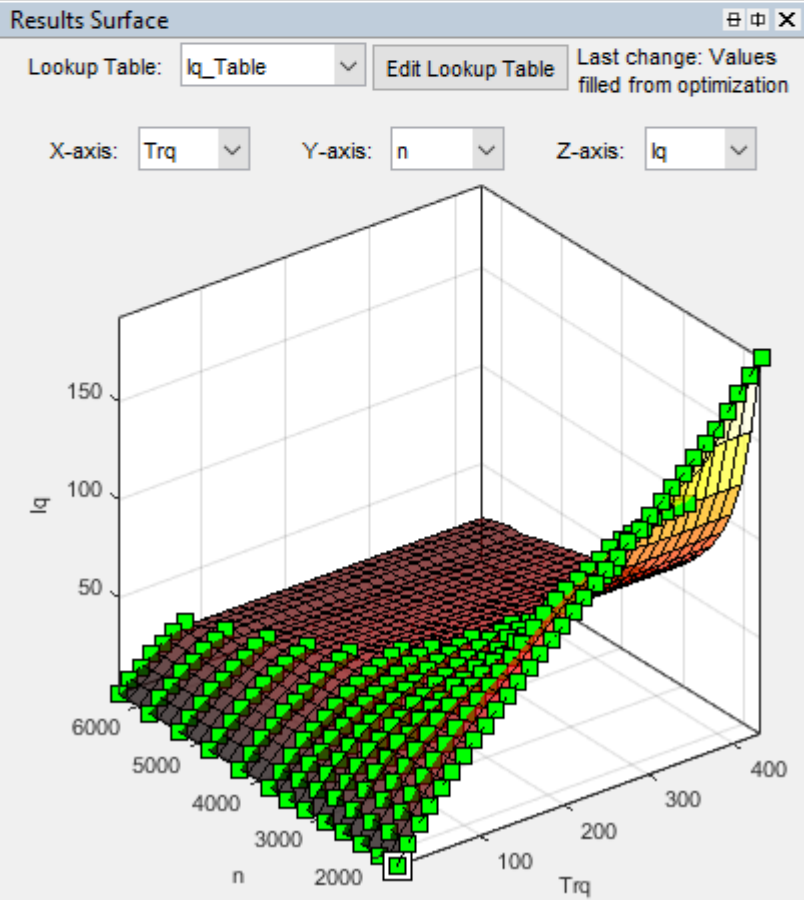
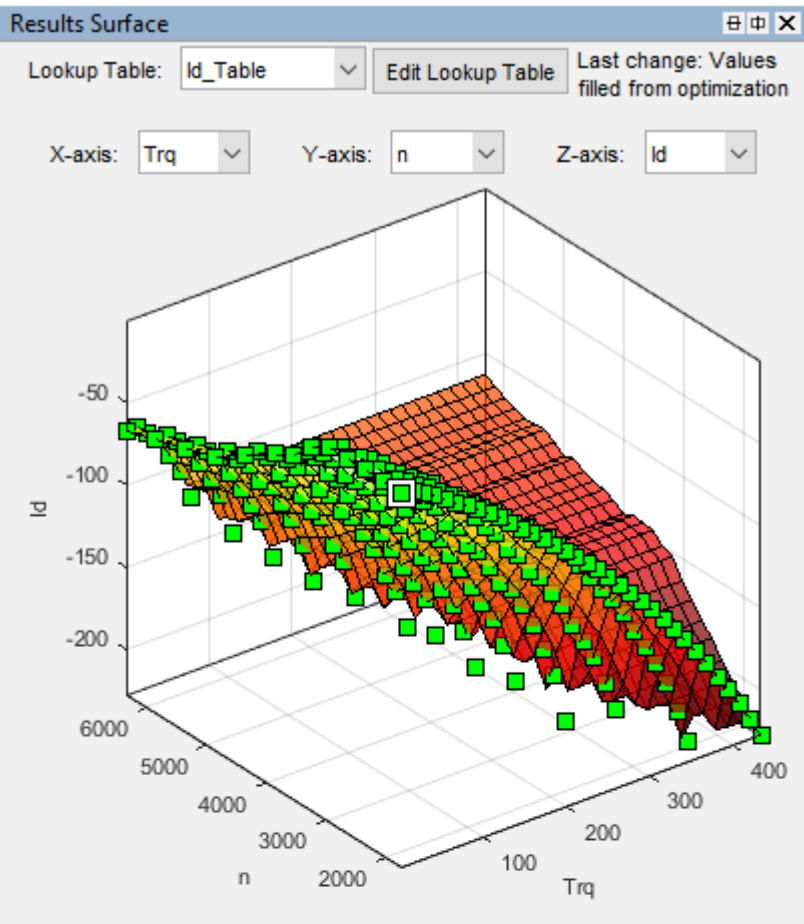
Optimization Information	
Algorithm name	mbcOSfmincon
Algorithm description	Single objective optimizati
Free variables	ld
Operating point vari...	None
Item scaling	off

Number of operating points: 214

Free Variables		Fixed Variables		
Variable:	ld	Variable:	Trq	n
1	-146.939	1	1	1720
2	-146.939	2	1	2020
3	-146.939	3	1	2320

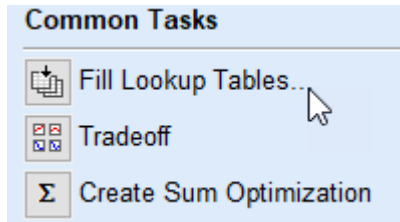
6 In the Cage Browser, select **Run**.

The optimization results are similar to these.

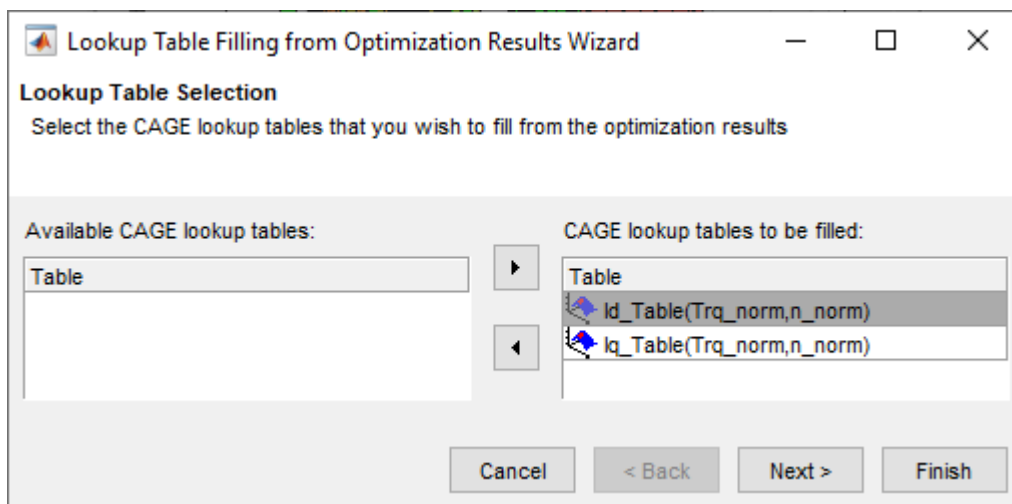


Fill Lookup Tables

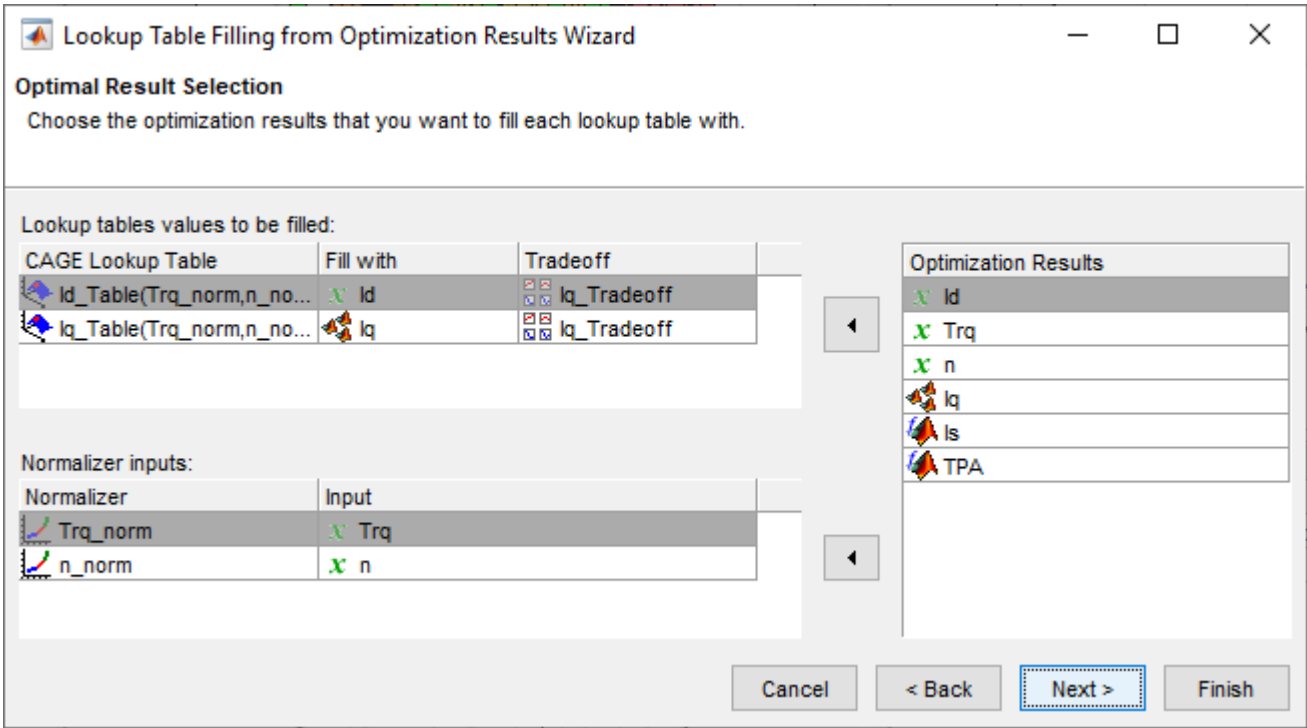
- 1 In the CAGE Browser, select **Fill Lookup Tables**.



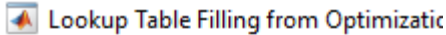
- 2 Use the Lookup Table Filling from Optimization Results Wizard to fill the Id_Table and Iq_Table tables.



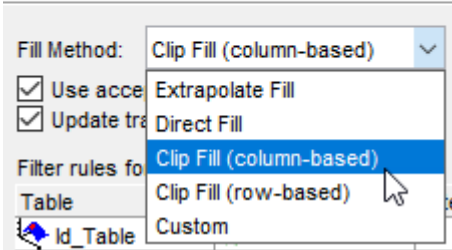
- For the Id_Table, fill with Id.
- For the Iq_Table, fill with Iq.



Click **Next**. For the **Fill Method**, select Clip Fill (column-based).

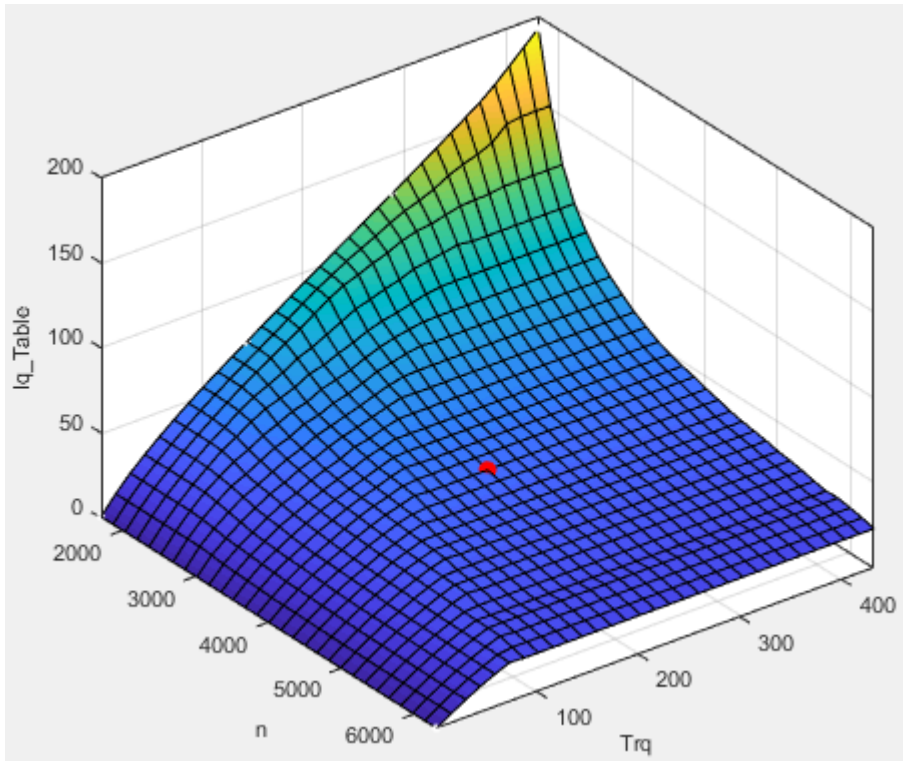


Fill Algorithm
Set up lookup table filling algorithm.

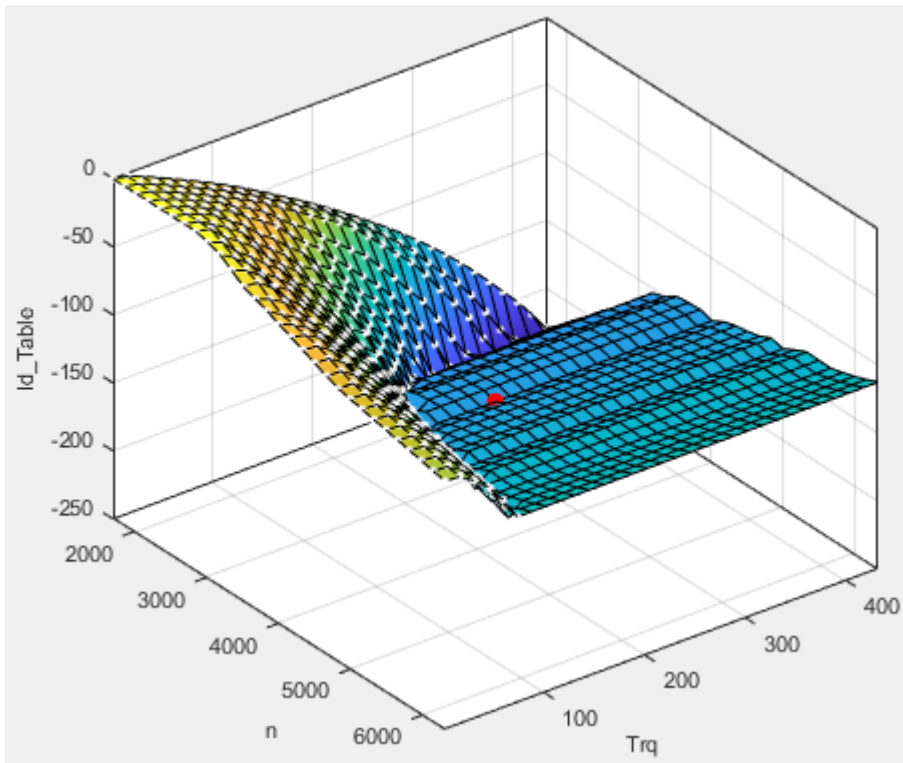


Click **Finish**.

- 3 Review results for Iq_Table. The results are similar to these.



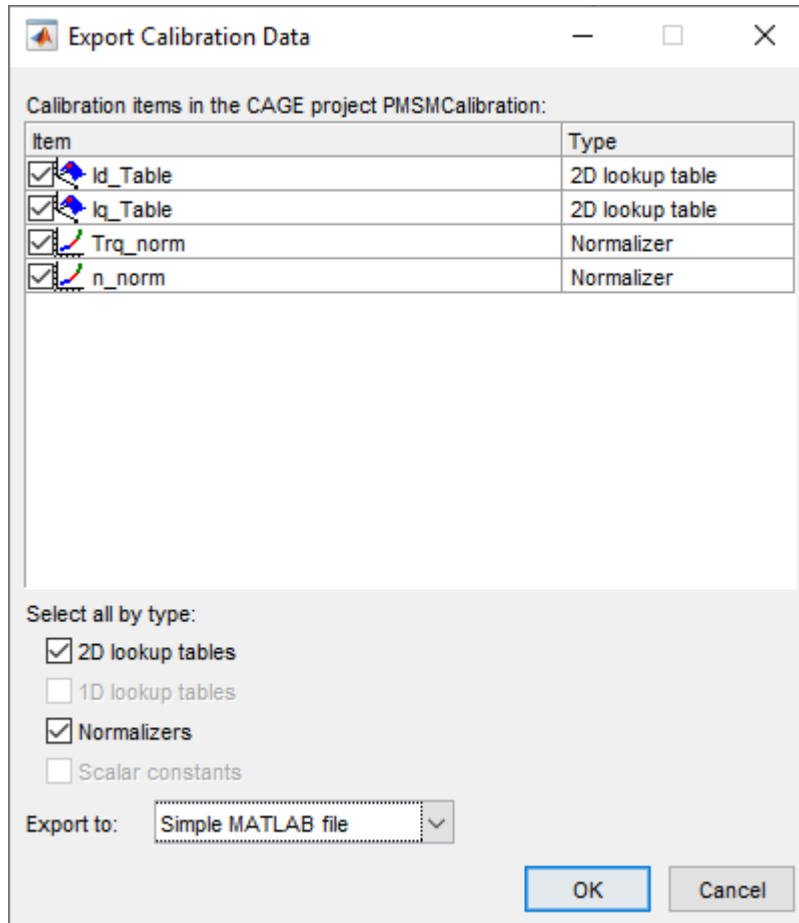
- 4 Review results for I_d_Table . The results are similar to these.



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

Export Results

- 1 Select **File > Export > Calibration**.
- 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
```

- 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, wbp	tbp=Trq_norm.X;
Vector of speed breakpoints, wbp	% 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;
Corresponding d-axis current reference, id_ref	id_table=Id_Table.Z; id_ref=id_table';
Corresponding q-axis current reference, iq_ref	iq_table=Iq_Table.Z; iq_ref=iq_table';

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 6-49	Load and preprocess this nonlinear motor flux data from dynamometer testing or finite element analysis (FEA): <ul style="list-style-type: none"> d- and q- axis current d- and q- axis flux Electromagnetic motor torque
“Step 2: Generate Evenly Spaced Data” on page 6-49	Use spline interpolation to generate evenly spaced data. Visualize the flux surface plots.
“Step 3: Set Block Parameters” on page 6-51	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 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 Use spline interpolation to get higher resolution.

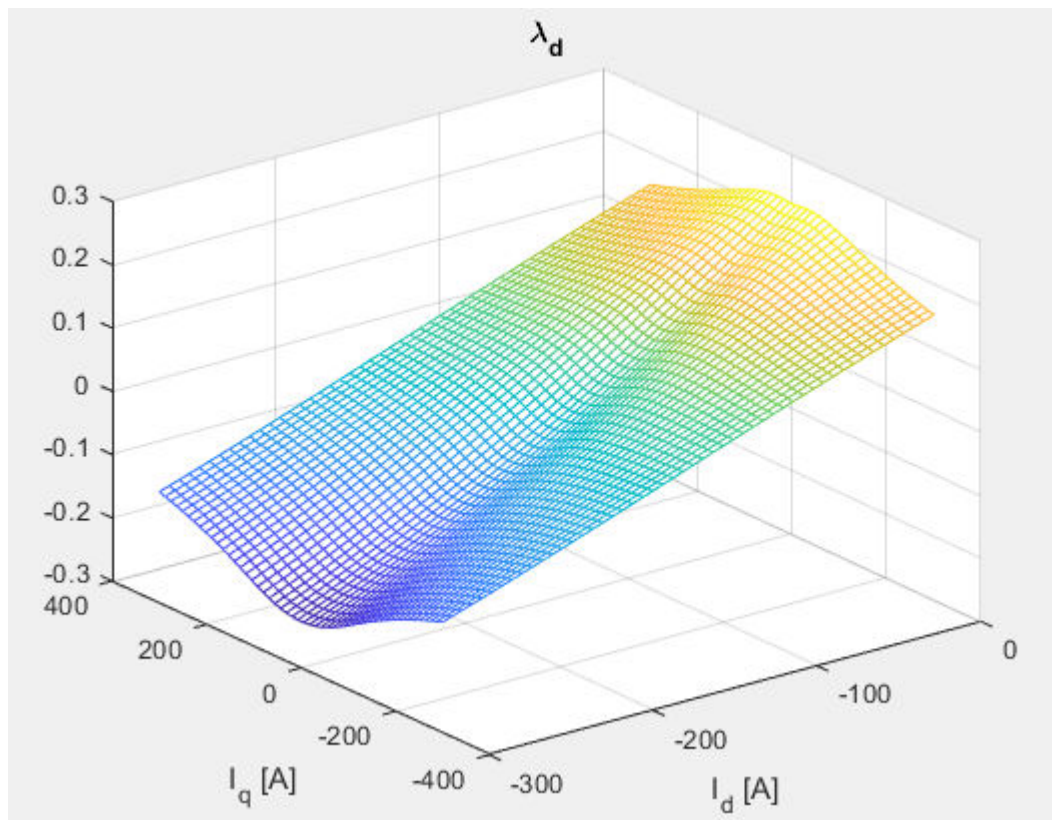
```
% Use spline interpolation to get higher resolution
id_new = linspace(min(id),max(id),flux_d_size);
iq_new = linspace(min(iq),max(iq),flux_q_size);
lambda_d_new = interp2(id',iq,lambda_d,id_new',iq_new,'spline');
lambda_q_new = interp2(id',iq,lambda_q,id_new',iq_new,'spline');
```

- 3 Visualize the flux surfaces.

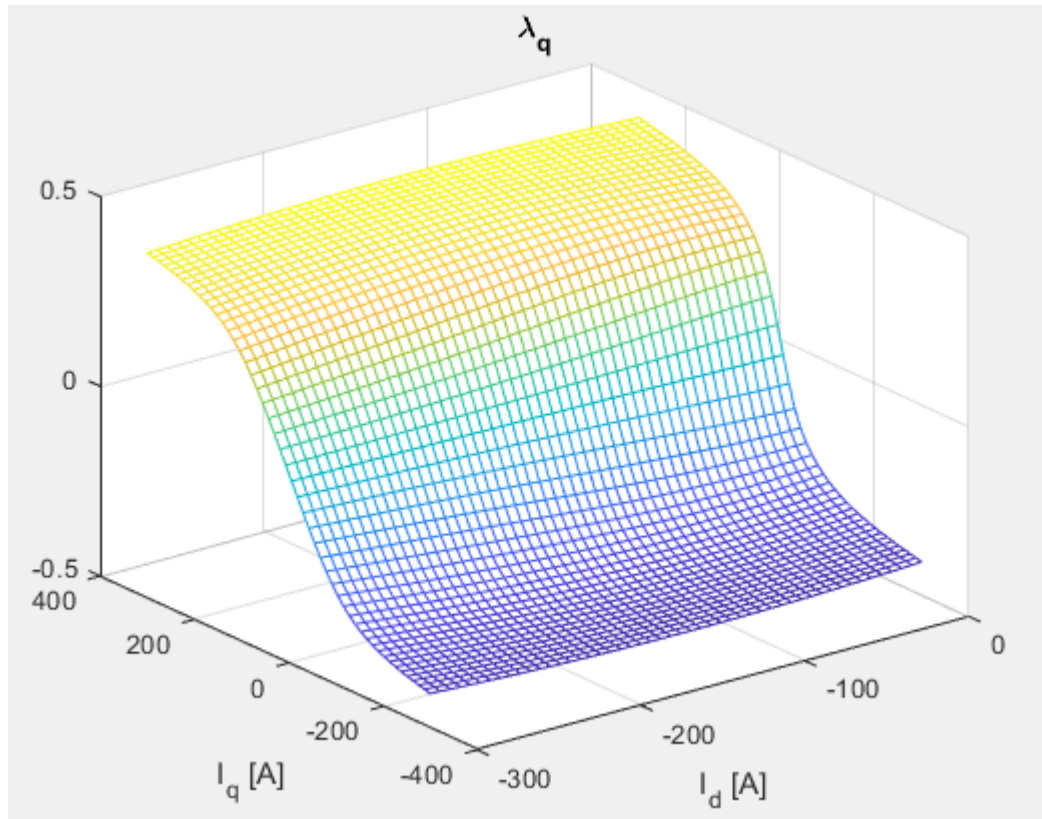
```
% Visualize the flux surface
figure;
mesh(id_new,iq_new,lambda_d_new);
xlabel('I_d [A]')
ylabel('I_q [A]')
title('\lambda_d'); grid on;
```

```
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, λ_d , as a function of d-axis current, I_d , and q-axis current, I_q .



- q-axis flux, λ_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	<code>lambda_d=lambda_d_new;</code>
Corresponding q-axis flux, lambda_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.

[3] Ottosson, J., M. Alakula. "A compact field weakening controller implementation." *International Symposium on Power Electronics, Electrical Drives, Automation and Motion*, July, 2006.

See Also

Flux-Based PMSM | Flux-Based PM Controller


```

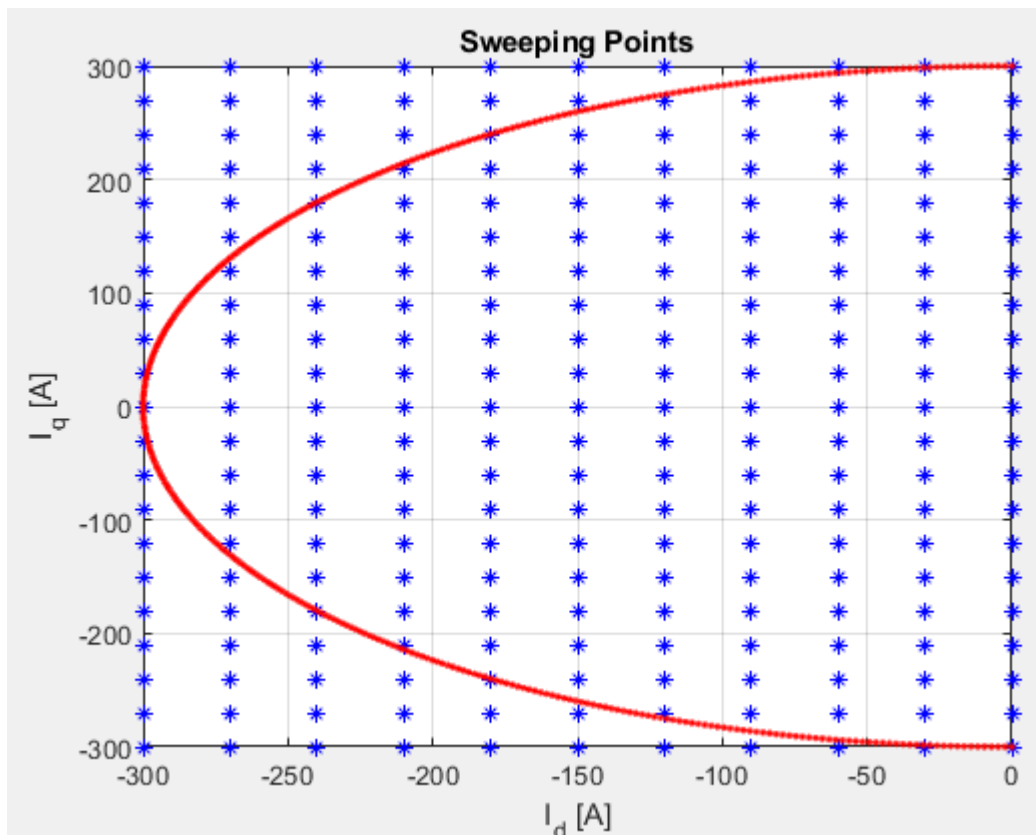
    end
end
5 Plot the current limit sweeping points and circle.

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);
```

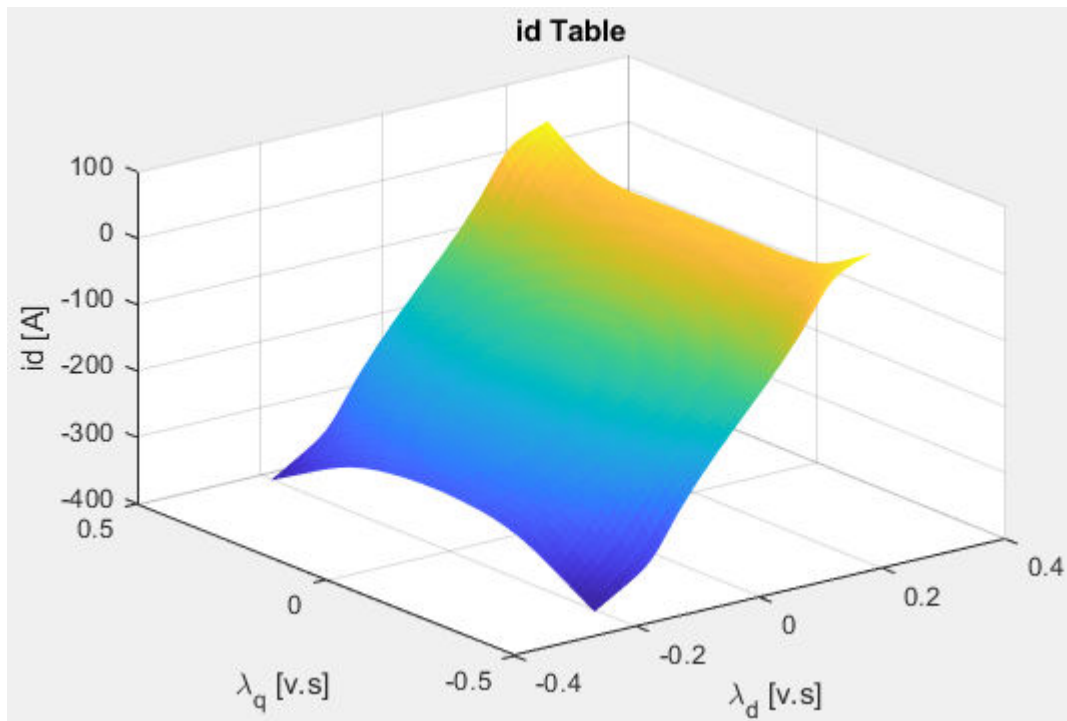
- 3 Create 2-D grid coordinates based on the d -axis and q -axis currents.

```
% 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'); grid on;
shading flat;
```

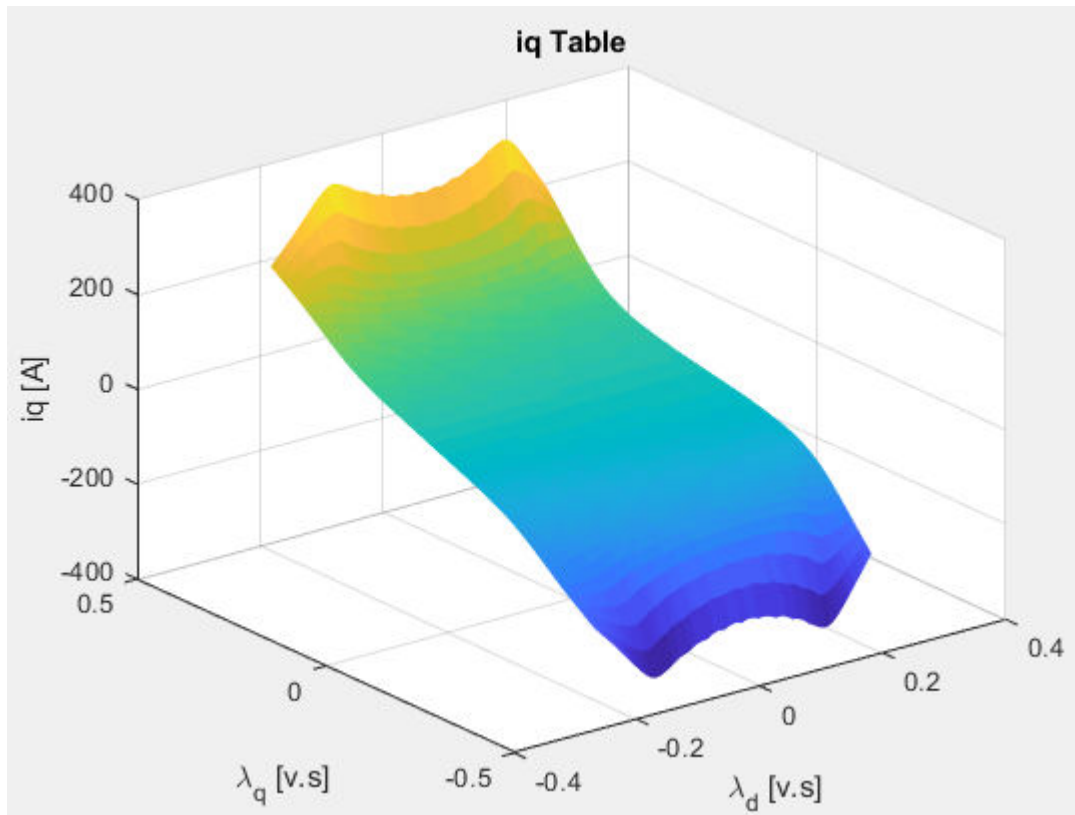
d -axis current, I_d , as a function of q -axis flux, λ_q , and d -axis flux, λ_d .



- 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 Table'); grid on;
shading flat;
```

q -axis current, I_q , as a function of q -axis flux, λ_q , and d -axis flux, λ_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, flux_d	<code>flux_d=ParamFluxDIndex;</code>
Vector of q-axis flux, flux_q	<code>flux_q=ParamFluxQIndex;</code>
Corresponding d-axis current, id	<code>id=ParamIdLookupTable;</code>
Corresponding q-axis current, iq	<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.
- [3] Ottosson, J., M. Alakula. "A compact field weakening controller implementation." *International Symposium on Power Electronics, Electrical Drives, Automation and Motion*, July, 2006.

See Also

Flux-Based PMSM | Flux-Based PM Controller

External Websites

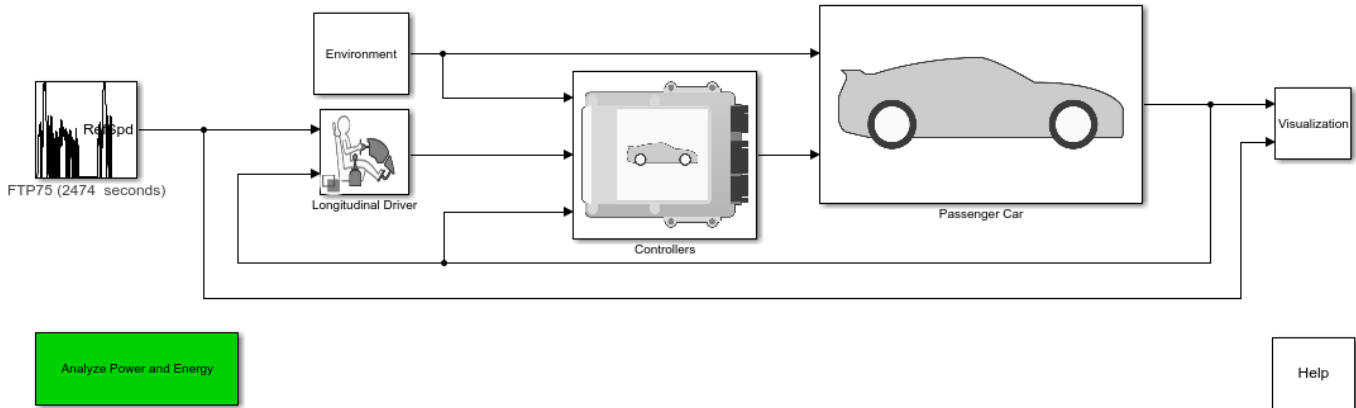
- [Surface Fitting using gridfit](#)

Powertrain Blockset Examples

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.

For more information, see “Explore the Conventional Vehicle Reference Application” on page 3-4.



Copyright 2015-2020 The MathWorks, Inc.

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 Spark-Ignition Engine Fuel Economy and Emissions” on page 1-10
- “Conventional Vehicle Powertrain Efficiency” on page 1-15
- “Generate a Deep Learning SI Engine Model” on page 3-100

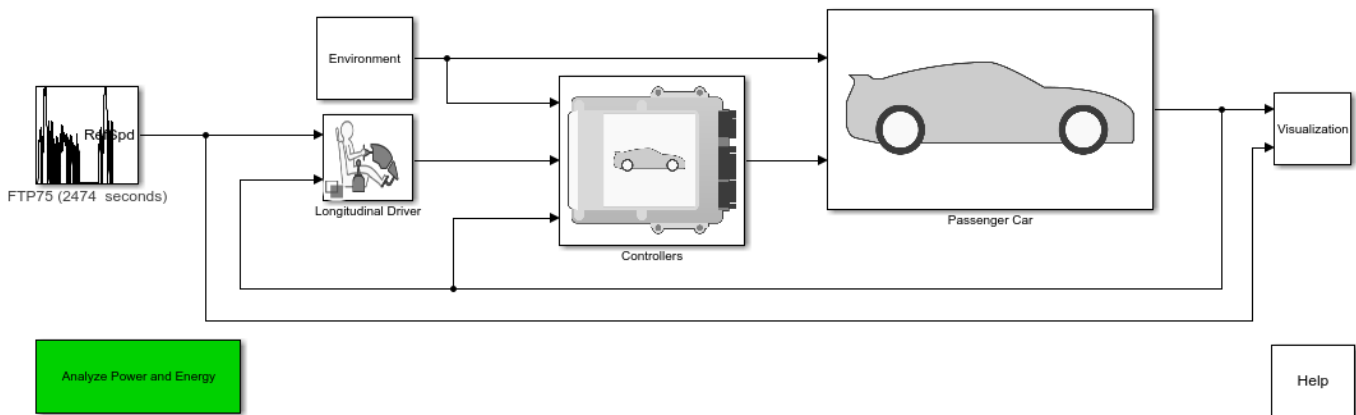
More About

- “Analyze Power and Energy” on page 3-107
- “Internal Combustion Mapped and Dynamic Engine Models” on page 3-106
- “Variant Systems”

HEV Multimode Reference Application

The hybrid electric vehicle (HEV) multimode reference application represents a full multimode 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.

For more information, see “Explore the Hybrid Electric Vehicle Multimode Reference Application” on page 3-18.



Copyright 2015-2020 The MathWorks, Inc.

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 3-31
- “Explore the Hybrid Electric Vehicle P2 Reference Application” on page 3-54
- “Explore the Electric Vehicle Reference Application” on page 3-25

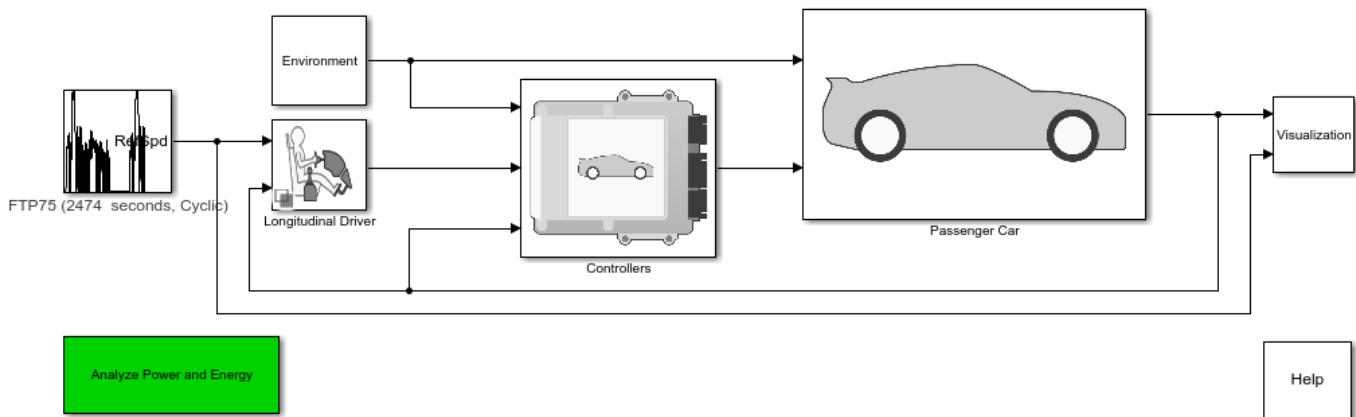
More About

- “Analyze Power and Energy” on page 3-107
- “Variant Systems”

HEV 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®.

For more information, see “Explore the Hybrid Electric Vehicle Input Power-Split Reference Application” on page 3-31.



Copyright 2017-2020 The MathWorks, Inc.

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 3-18
- “Explore the Hybrid Electric Vehicle P2 Reference Application” on page 3-54
- “Explore the Electric Vehicle Reference Application” on page 3-25

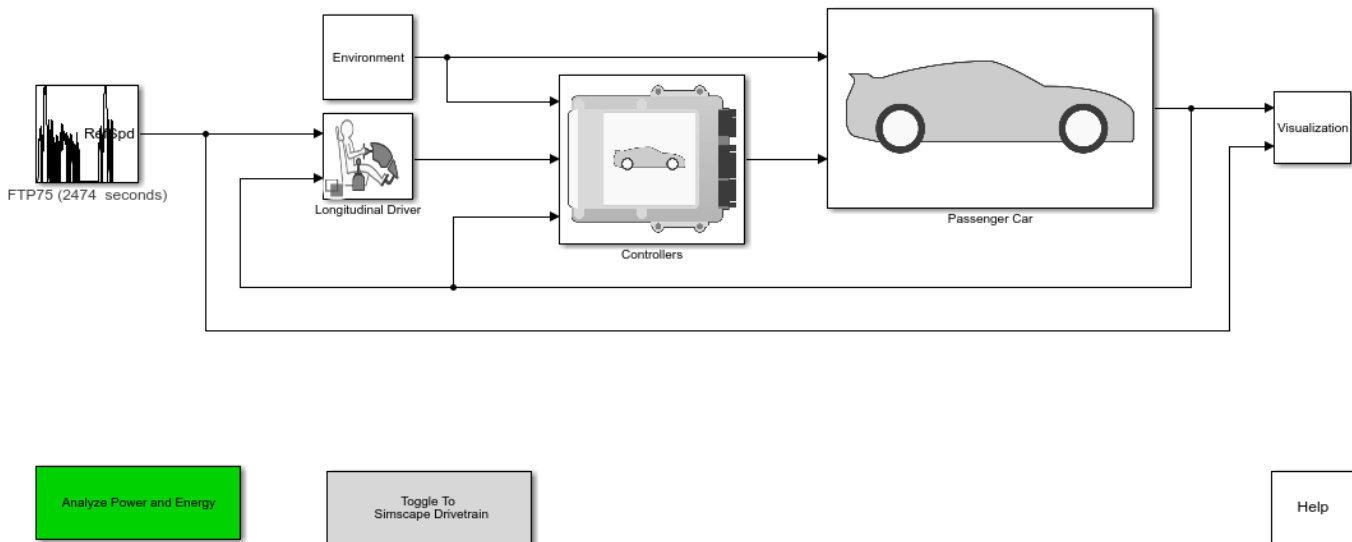
More About

- “Analyze Power and Energy” on page 3-107
- “Variant Systems”

HEV P0 Reference Application

The hybrid electric vehicle (HEV) P0 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 P0 hybrid.

For more information, see “Explore the Hybrid Electric Vehicle P0 Reference Application” on page 3-40.



Copyright 2018-2021 The MathWorks, Inc.

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 3-31
- “Explore the Hybrid Electric Vehicle Multimode Reference Application” on page 3-18
- “Explore the Electric Vehicle Reference Application” on page 3-25

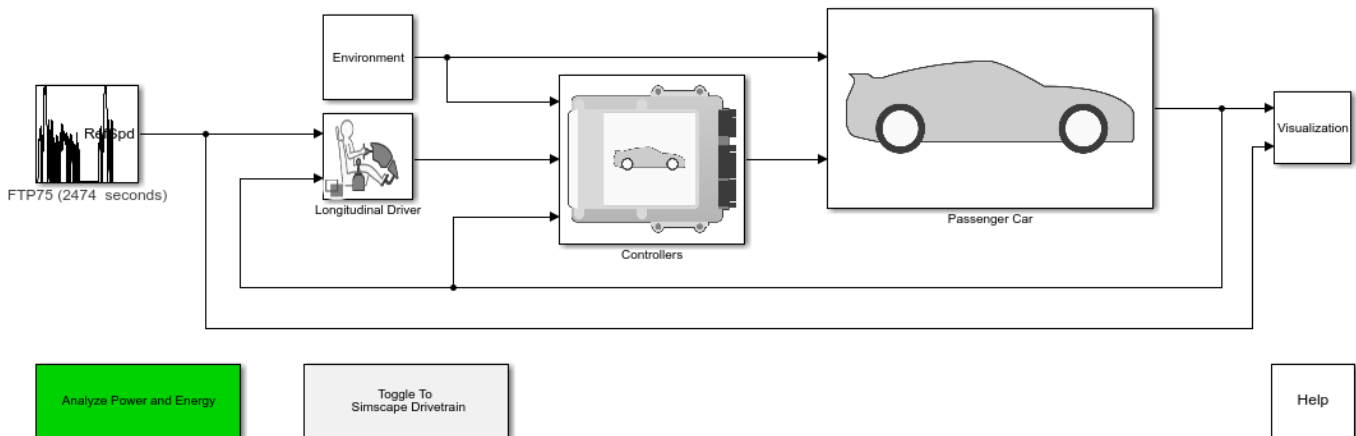
More About

- “Analyze Power and Energy” on page 3-107
- “Variant Systems”

HEV P1 Reference Application

The hybrid electric vehicle (HEV) P1 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 P1 hybrid.

For more information, see “Explore the Hybrid Electric Vehicle P1 Reference Application” on page 3-47.



Copyright 2018-2021 The MathWorks, Inc.

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 3-31
- “Explore the Hybrid Electric Vehicle Multimode Reference Application” on page 3-18
- “Explore the Electric Vehicle Reference Application” on page 3-25

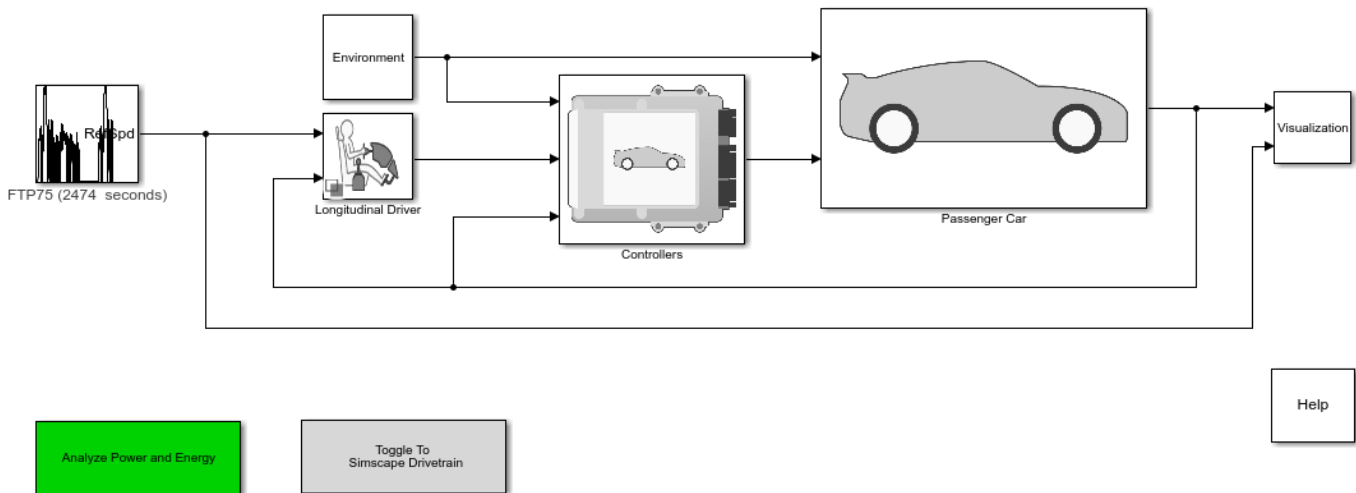
More About

- “Analyze Power and Energy” on page 3-107
- “Variant Systems”

HEV 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.

For more information, see “Explore the Hybrid Electric Vehicle P2 Reference Application” on page 3-54.



Copyright 2021 The MathWorks, Inc.

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 3-31
- “Explore the Hybrid Electric Vehicle Multimode Reference Application” on page 3-18
- “Explore the Electric Vehicle Reference Application” on page 3-25

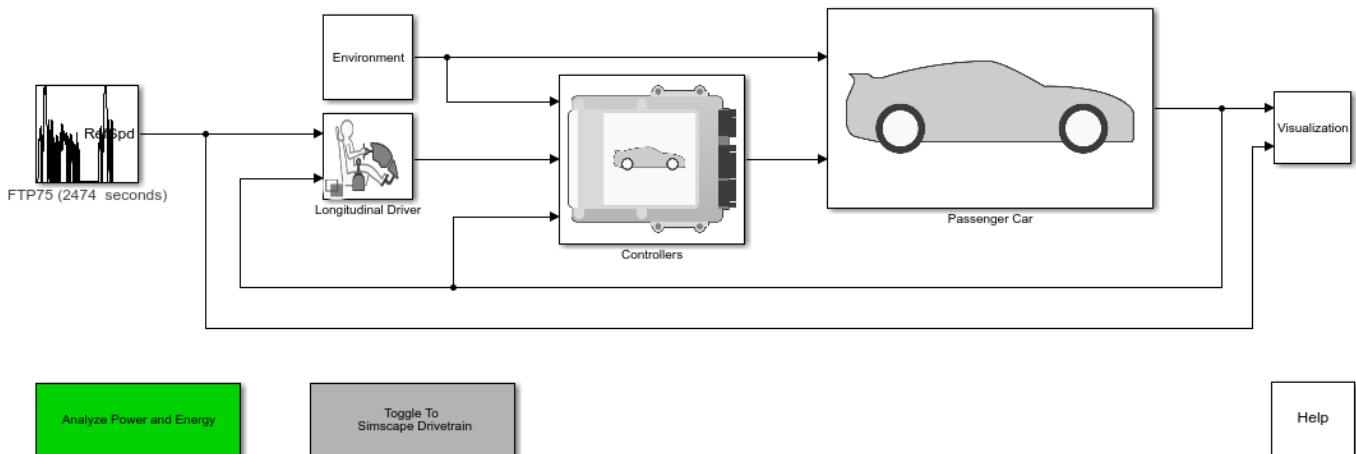
More About

- “Analyze Power and Energy” on page 3-107
- “Variant Systems”

HEV P3 Reference Application

The hybrid electric vehicle (HEV) P3 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 P3 hybrid.

For more information, see “Explore the Hybrid Electric Vehicle P3 Reference Application” on page 3-63.



Copyright 2018-2021 The MathWorks, Inc.

See Also

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

Related Examples

- “Explore the Hybrid Electric Vehicle Input Power-Split Reference Application” on page 3-31
- “Explore the Hybrid Electric Vehicle Multimode Reference Application” on page 3-18
- “Explore the Electric Vehicle Reference Application” on page 3-25

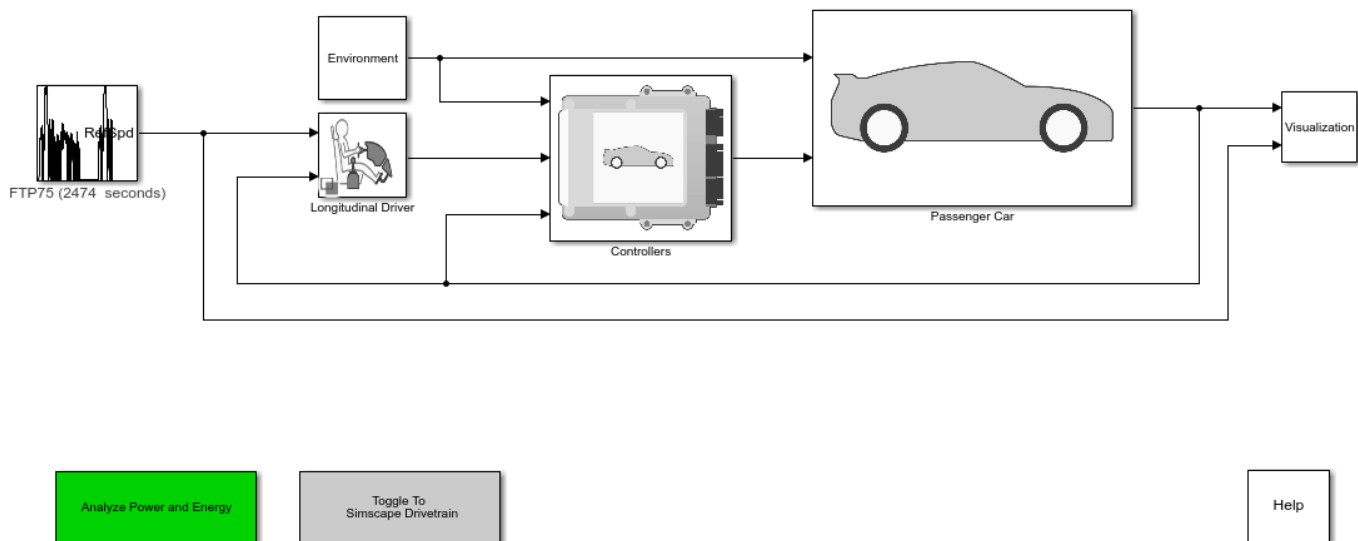
More About

- “Analyze Power and Energy” on page 3-107
- “Variant Systems”

HEV P4 Reference Application

The hybrid electric vehicle (HEV) P4 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 P4 hybrid.

For more information, see “Explore the Hybrid Electric Vehicle P4 Reference Application” on page 3-70.



Copyright 2018-2021 The MathWorks, Inc.

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 3-31
- “Explore the Hybrid Electric Vehicle Multimode Reference Application” on page 3-18
- “Explore the Electric Vehicle Reference Application” on page 3-25

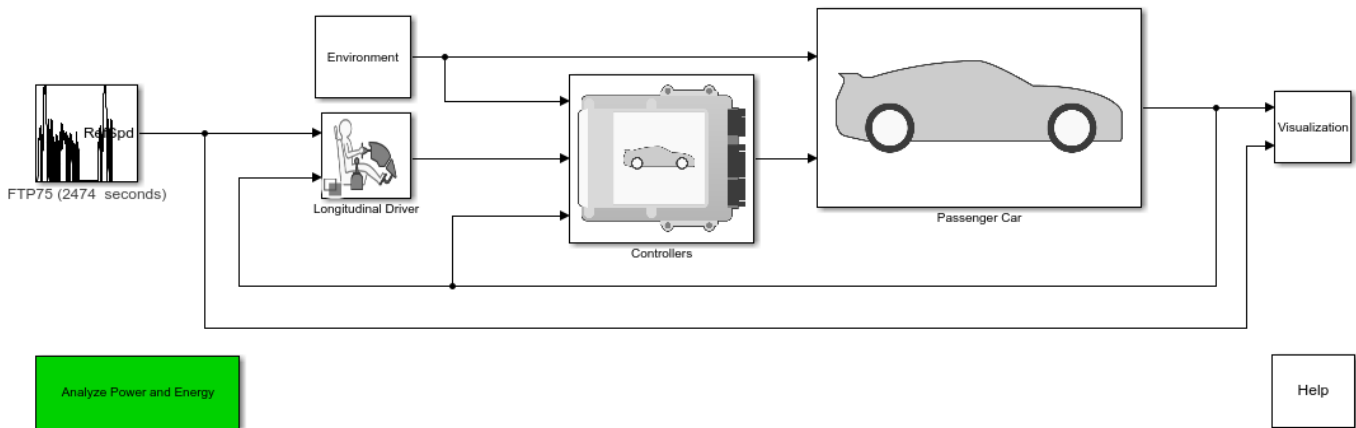
More About

- “Analyze Power and Energy” on page 3-107
- “Variant Systems”

EV Reference Application

The electric vehicle (EV) 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.

For more information, see “Explore the Electric Vehicle Reference Application” on page 3-25.



Copyright 2015-2020 The MathWorks, Inc.

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 3-18
- “Explore the Hybrid Electric Vehicle Input Power-Split Reference Application” on page 3-31
- “Explore the Hybrid Electric Vehicle P2 Reference Application” on page 3-54

More About

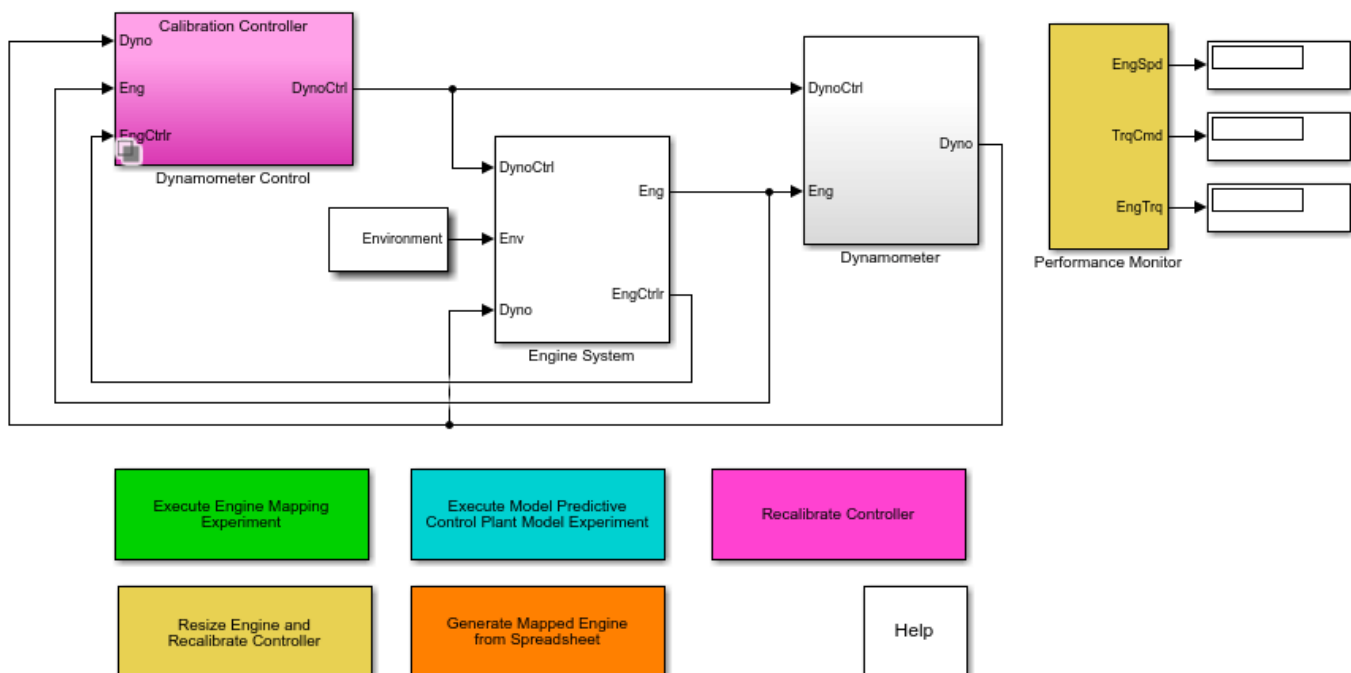
- “Analyze Power and Energy” on page 3-107
- “Variant Systems”

CI Engine Dynamometer Reference Application

The compression-ignition (CI) engine dynamometer reference application represents a CI engine plant and controller connected to a 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.

For more information, see “Explore the CI Engine Dynamometer Reference Application” on page 3-10.

Engine Dynamometer



Copyright 2015-2020 The MathWorks, Inc.

See Also

CI Controller | CI Core Engine | Mapped CI Engine

More About

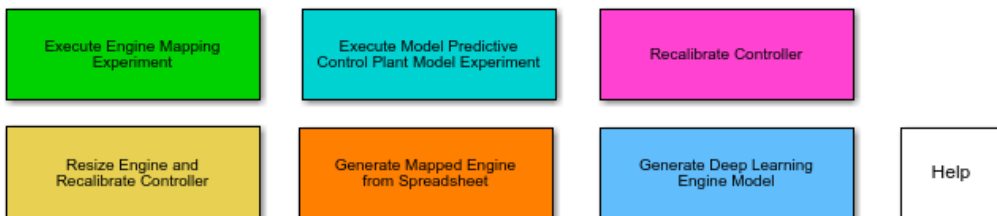
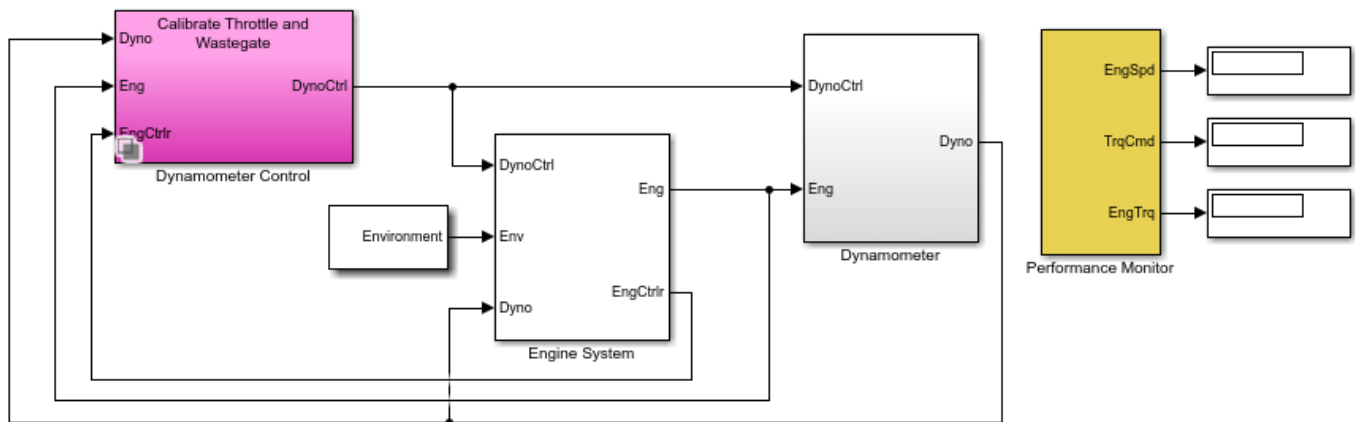
- “CI Engine Project Template” on page 4-2
- “Generate Mapped CI Engine from a Spreadsheet” on page 3-91
- “Resize the CI Engine” on page 3-77
- “Internal Combustion Mapped and Dynamic Engine Models” on page 3-106
- “Variant Systems”

SI Engine Dynamometer Reference Application

The spark-ignition (SI) engine dynamometer reference application represents a SI engine plant and controller connected to a 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.

For more information, see “Explore the SI Engine Dynamometer Reference Application” on page 3-14.

Engine Dynamometer



Copyright 2015-2020 The MathWorks, Inc.

See Also

Mapped SI Engine | SI Controller | SI Core Engine

More About

- “Generate Mapped SI Engine from a Spreadsheet” on page 3-96
- “Generate a Deep Learning SI Engine Model” on page 3-100
- “Resize the SI Engine” on page 3-84
- “Internal Combustion Mapped and Dynamic Engine Models” on page 3-106
- “Variant Systems”

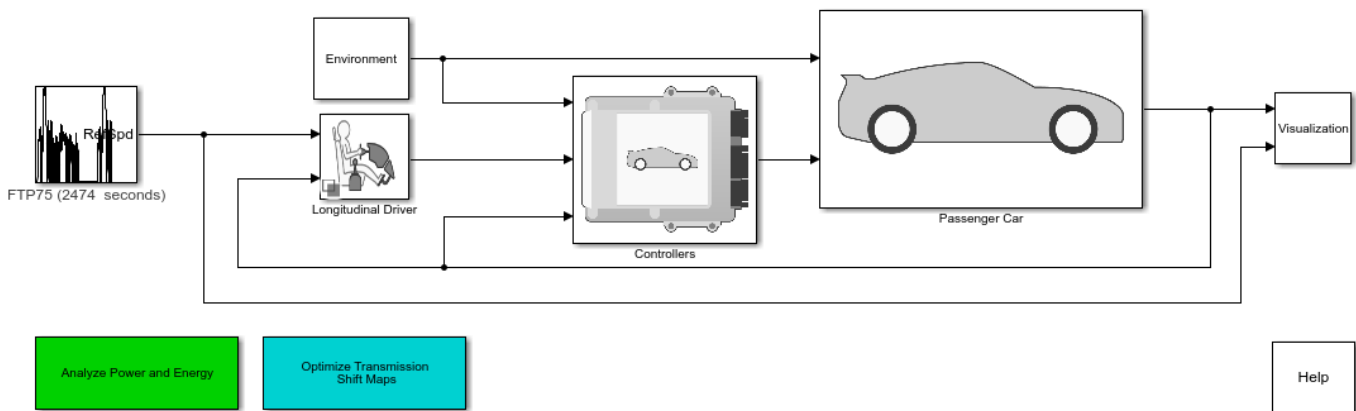
Optimize Transmission Control Module Shift Schedules

This example shows how to use the conventional vehicle reference application to optimize the transmission control module (TCM) shift schedules. Use the optimized shift schedules to:

- Design control algorithms.
- Assess the impact of powertrain changes, such as an engine or gear ratio, on performance, fuel economy, and emissions.

This example uses the Global Optimization Toolbox, Simulink® Design Optimization™, and Stateflow®. To increase optimization performance, consider using the Parallel Computing Toolbox™.

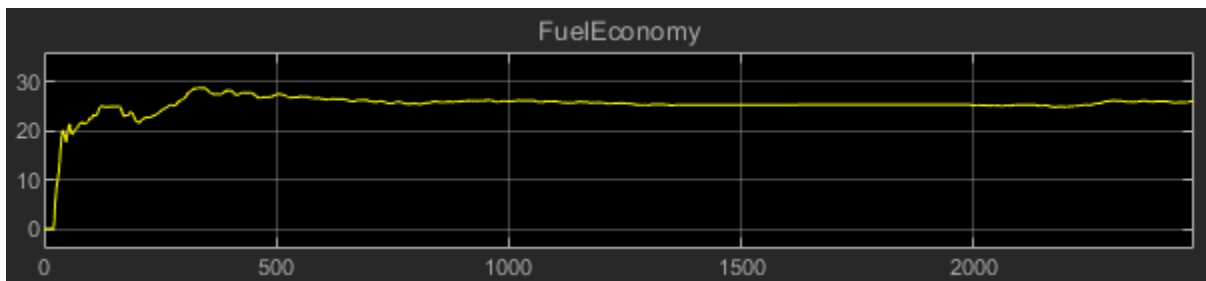
For more information about the reference application, see “Explore the Conventional Vehicle Reference Application” on page 3-4.



Copyright 2015-2019 The MathWorks, Inc.

Run Conventional Vehicle Reference Application

Click **Run** to simulate the conventional vehicle reference application with the default settings. The results indicate that the conventional vehicle has a fuel economy of approximately 26 mpg.



Optimize Transmission Shift Maps

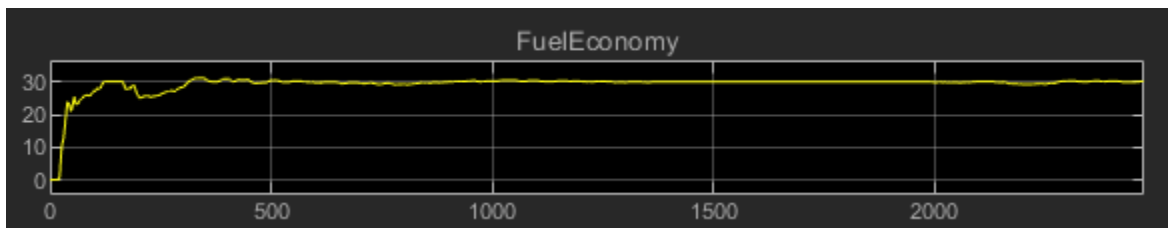
Click **Optimize Transmission Shift Maps**. Optimizing the shift schedules can take time to run. If you have the Parallel Computing Toolbox, the optimization uses parallel workers by default. View the optimization in the MATLAB® window.



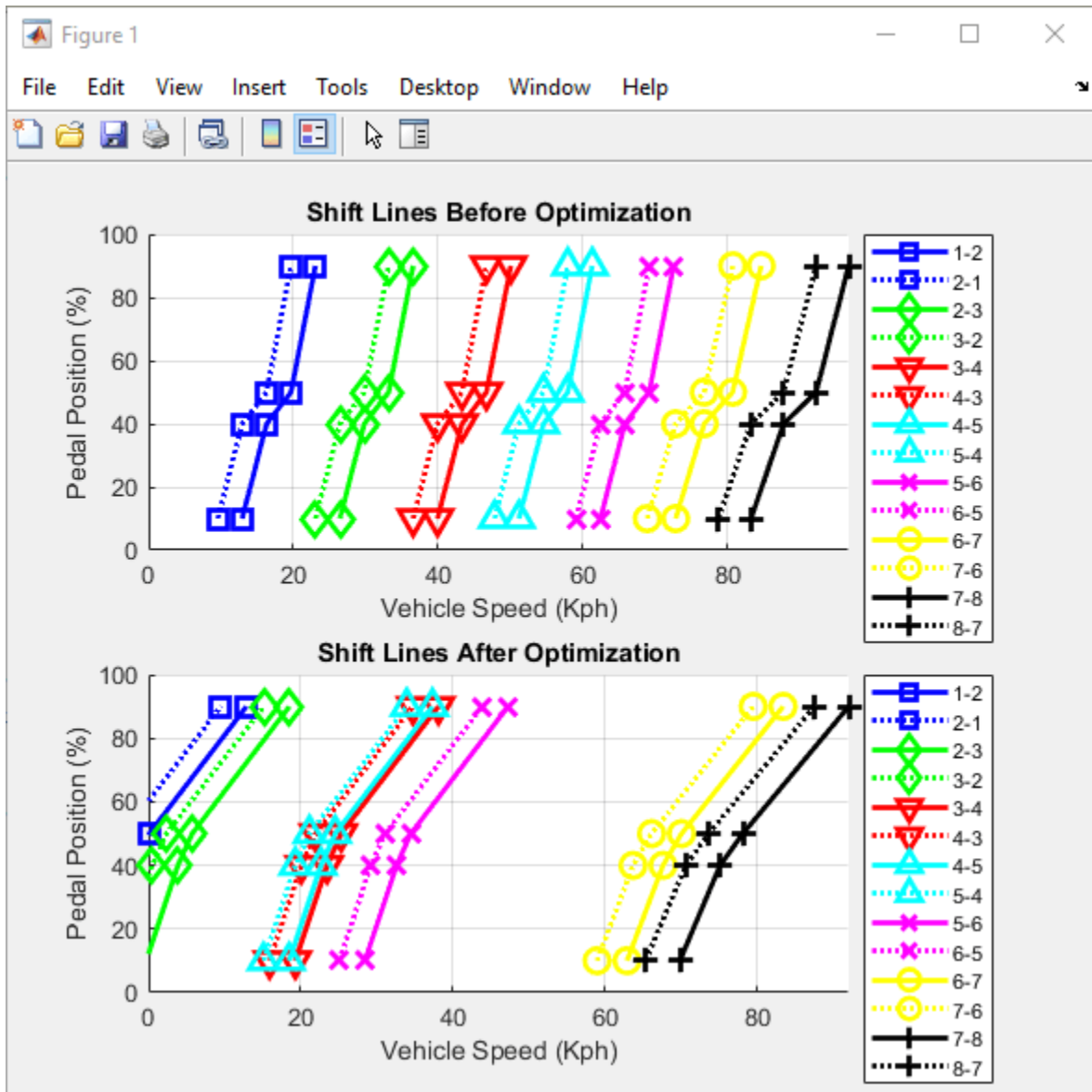
View Results

After you optimize the shift schedule, view the results.

The performance scope indicates that the conventional vehicle with an optimized TCM shift schedule has a fuel economy of approximately 30 mpg.



The figure shows the transmission shift schedule upshift and downshift calibration lines before and after optimization.



Open Reference Application From Command Line

Use this command to open a version of the conventional vehicle reference application that includes the option to optimize transmission shift maps.

```
autoblkConVehShftOptStart
```

See Also

More About

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

- “Simulink Design Optimization”
- “Stateflow”

Calibrate ECMS Block

This example shows how to calibrate the ECMS block in the HEV P2 reference application.

HEV P2 Reference Application

Open the P2 reference application.

```
autoblkHevP2Start
```

Set Path

Sets path to the calibration script.

```
path = fullfile(matlabroot, 'examples', 'autoblocks', 'main', 'internal');
addpath(path);
```

Execute Calibration Script

```
% calibration_ecms_script
close all;
pxName="P2"; % PT configuration type, P0,P1,P2,P3,P4
battMdl = "BattHev"+pxName; % battery model - for SOC sweep
mdlName="Hev"+pxName+"ReferenceApplication"; % Hev model
ctrlMdl="Hev"+pxName+"OptimalController"; % Controller model
maxIterat=4;% maximum iterations
SOCinit = 0.85; % initial SOC, unit is in [0, 1]
SOCEndTrg = SOCinit*100;

% Plot window size and position
x0=600;
y0=1040;
width=600;
height=280;

tic
load_system(mdlName);
load_system(ctrlMdl);
load_system(battMdl);

blk = mdlName + "/" + "Drive Cycle Source";
m = get_param(blk, 'MaskObject');
ECMS_CurrentCycle = m.Parameters(1,1).Value; % Current cycle setting.

mdlWks = get_param(ctrlMdl, 'ModelWorkspace'); % find model workspace
ECMS_CurrentValue = getVariable(mdlWks, 'ECMS_s'); % get current ECMS_s value

% extract initial value/name of ECMS tuning parameter
p = Simulink.Mask.get(ctrlMdl+"/ECMS");
baseParamName=p.getParameter("ECMS_s").Value;
battChrgMaxValue = getVariable(get_param(battMdl, 'modelworkspace'), 'BattChargeMax');

% set to a temp name
set_param(ctrlMdl+"/ECMS", 'ECMS_s', "ECMS_s_tune")
save_system(mdlName, [], 'SaveDirtyReferencedModels', 'on');
save_system(ctrlMdl, [], 'SaveDirtyReferencedModels', 'on');

% Enable SOC recording
```

```

x1_handles = get_param mdlName+"/Visualization/Rate Transition1", 'PortHandles');
x1 = x1_handles.Outputport(1);
Simulink.sdi.markSignalForStreaming(x1, 'on');

% arrays used to store ECMS_s and SOC_end values
xc = zeros (3,1);
yc = zeros (3,1);

% plot showing the ECMS_s and SOC_end
subplot (1,2,1);
set(gcf, 'position', [x0,y0,width,height])
xlabel({'ECMS_s', ' '})
ylabel('dSOC')
hold on;
subplot (1,2,2);
xlabel('ECMS_s')
ylabel('SOC')
sgtitle(ECMS_CurrentCycle);
% set model workspace variables
in = ModelSetVariable (mdlName, battChrgMaxValue, SOCinit,battMdl, SOCEndTrg, ctrlMdl);
% get initial 3 ECMS_s points and SOC_end values
[x, y, ECMS_s, SOC_end, solution_found] = dSOC_generate_starting_points (in, ECMS_CurrentValue,

### Starting serial model reference simulation build
### Successfully updated the model reference simulation target for: BattHevP2
### Successfully updated the model reference simulation target for: DrivetrainHevP2
### Successfully updated the model reference simulation target for: HevP2OptimalController
### Successfully updated the model reference simulation target for: HevP2TransmissionController
### Successfully updated the model reference simulation target for: MotMappedP2
### Successfully updated the model reference simulation target for: SiEngineController
### Successfully updated the model reference simulation target for: SiMappedEngine
### Successfully updated the model reference simulation target for: StarterSystemP2

Build Summary

Simulation targets built:

Model Action Rebuild Reason
=====
BattHevP2 Code generated and compiled BattHevP2_msf.mexw64 does not exist.
DrivetrainHevP2 Code generated and compiled DrivetrainHevP2_msf.mexw64 does not exist.
HevP2OptimalController Code generated and compiled HevP2OptimalController_msf.mexw64 does not exist.
HevP2TransmissionController Code generated and compiled HevP2TransmissionController_msf.mexw64 does not exist.
MotMappedP2 Code generated and compiled MotMappedP2_msf.mexw64 does not exist.
SiEngineController Code generated and compiled SiEngineController_msf.mexw64 does not exist.
SiMappedEngine Code generated and compiled SiMappedEngine_msf.mexw64 does not exist.
StarterSystemP2 Code generated and compiled StarterSystemP2_msf.mexw64 does not exist.

8 of 8 models built (0 models already up to date)
Build duration: 0h 7m 16.091s
x1 = 3.430000, y1 = 78.964430
### Starting serial model reference simulation build
### Model reference simulation target for DrivetrainHevP2 is up to date.
### Model reference simulation target for HevP2TransmissionController is up to date.
### Model reference simulation target for MotMappedP2 is up to date.
### Model reference simulation target for SiEngineController is up to date.
### Model reference simulation target for SiMappedEngine is up to date.
### Model reference simulation target for StarterSystemP2 is up to date.

```

Build Summary

```

0 of 8 models built (8 models already up to date)
Build duration: 0h 0m 14.237s
x2 = 3.676594, y2 = 79.652346
### Starting serial model reference simulation build
### Model reference simulation target for DrivetrainHevP2 is up to date.
### Model reference simulation target for HevP2TransmissionController is up to date.
### Model reference simulation target for MotMappedP2 is up to date.
### Model reference simulation target for SiEngineController is up to date.
### Model reference simulation target for SiMappedEngine is up to date.
### Model reference simulation target for StarterSystemP2 is up to date.

```

Build Summary

```

0 of 8 models built (8 models already up to date)
Build duration: 0h 0m 13.479s
x3 = 4.145992, y3 = 86.898529

```

```
if (solution_found == 0)
```

```
    for i = 1 : maxIterat
```

```

        [yy, II] = sort(y, 'ascend'); % sort the x, y array for plotting
        xx = x(II);
        subplot (1,2,2)
        if (i == 1); plot(xx,yy, 'b--o', 'LineWidth', 2); end
        if (i == 2); plot(xx,yy, 'g--o', 'LineWidth', 2); end
        if (i == 3); plot(xx,yy, 'r--o', 'LineWidth', 2); end
        if (i == 4); plot(xx,yy, 'y--o', 'LineWidth', 2); end
        xlabel('ECMS_s')
        ylabel('SOC')
        hold off
        % solve second order equation to get z = ECMS_s corresponds to y = SOCEndTrg

```

$SOC_{EndTrg} = SOC_{init} = ax^2 + bx + c$, where a, b, c satisfies $y_i = ax_i^2 + bx_i + c$, $1 \leq i \leq 3$. with

$x_1 = ECMS_s^{(1)}, y_1 = SOC_{end}(ECMS_s^{(1)}), x_2 = ECMS_s^{(2)}, y_2 = SOC_{end}(ECMS_s^{(2)}), x_3 = ECMS_s^{(3)}, y_3 = SOC_{end}(ECMS_s^{(3)})$

```

        [z] = Second_order_roots (x, y, SOCEndTrg);
        in = in.setVariable("ECMS_s_tune", z);
        [SOC_end, dSOC] = dSOCsim_v1(in);
        subplot (1,2,1);
        plot(z,dSOC, 'bs', 'MarkerSize', 15)
        hold on
        fprintf ('i= %u, ECMS_s = %f, SOC_end = %f \n', i, z, SOC_end);
        fprintf ('i= %u, x1 = %f, x2 = %f, x3 = %f, \n', i, x(1), x(2), x(3));
        fprintf ('i= %u, y1 = %f, y2 = %f, y3 = %f, \n', i, y(1), y(2), y(3));

```

```

        if (abs(SOC_end - SOCEndTrg) <= 1) % check if a solution is found
            ECMS_s = z;
            x(1) = ECMS_s;
            y(1) = SOC_end;
            solution_found = 1;
            break;
        end
    end
end

```

```

end

xc(1:3) = x (1:3); yc(1:3) = y (1:3);
x(1) = z; y(1) = SOC_end; % since z and SOC_end are new points, we use them and put into
[yb, II] = sort(yc, 'ascend'); % sort the array for processing
xb = xc(II);

% begin the process to pick other two points from the original 3 point
% xb(1:3), yb(1:3)

if (y(1) > SOCEndTrg) % y(1) > SOCEndTrg, we need to pick other points < SOCEndTrg
    if ((yb(2) < SOCEndTrg) && (SOCEndTrg < yb(3))) % yb(2) < SOCEndTrg, pick yb(2)
        x(2) = xb(2); y(2) = yb(2);
        if (abs(yb(1)-SOCEndTrg) < abs(yb(3)-SOCEndTrg)) % pick last point from xb(1), and
            x(3) = xb(1); y(3) = yb(1);
        else
            x(3) = xb(3); y(3) = yb(3);
        end
    end
    if ((yb(1) < SOCEndTrg) && (SOCEndTrg < yb(2))) % yb(1) < SOCEndTrg, pick yb(1)
        x(2) = xb(1); y(2) = yb(1);
        if (abs(yb(2)-SOCEndTrg) < abs(yb(3)-SOCEndTrg)) % pick last point from xb(2) and
            x(3) = xb(2); y(3) = yb(2);
        else
            x(3) = xb(3); y(3) = yb(3);
        end
    end
end
end
if (y(1) < SOCEndTrg) % y(1) < SOCEndTrg, we need to pick other points > SOCEndTrg
    if ((yb(2) < SOCEndTrg) && (SOCEndTrg < yb(3))) % yb(3) > SOCEndTrg, pick yb(3)
        x(2) = xb(3); y(2) = yb(3);
        if (abs(yb(1)-SOCEndTrg) < abs(yb(2)-SOCEndTrg)) % pick last point from xb(1) and
            x(3) = xb(1); y(3) = yb(1);
        else
            x(3) = xb(2); y(3) = yb(2);
        end
    end
    if ((yb(1) < SOCEndTrg) && (SOCEndTrg < yb(2))) % yb(2) > SOCEndTrg, pick this point
        x(2) = xb(2); y(2) = yb(2);
        if (abs(yb(1)-SOCEndTrg) < abs(yb(3)-SOCEndTrg)) % pick last point from xb(1) and
            x(3) = xb(1); y(3) = yb(1);
        else
            x(3) = xb(3); y(3) = yb(3);
        end
    end
end
end
end

y_abs = abs(y-SOCEndTrg);
[yb, II] = sort(y_abs, 'ascend');
xb = x(II);
ECMS_s = xb (1);

end

### Starting serial model reference simulation build
### Model reference simulation target for DrivetrainHevP2 is up to date.
### Model reference simulation target for HevP2TransmissionController is up to date.

```

```

### Model reference simulation target for MotMappedP2 is up to date.
### Model reference simulation target for SiEngineController is up to date.
### Model reference simulation target for SiMappedEngine is up to date.
### Model reference simulation target for StarterSystemP2 is up to date.

```

Build Summary

```

0 of 8 models built (8 models already up to date)
Build duration: 0h 0m 13.228s

```

```
i= 1, ECMS_s = 4.060548, SOC_end = 86.007364
```

```
i= 1, x1 = 3.430000, x2 = 3.676594, x3 = 4.145992,
```

```
i= 1, y1 = 78.964430, y2 = 79.652346, y3 = 86.898529,
```

```
### Starting serial model reference simulation build
```

```

### Model reference simulation target for DrivetrainHevP2 is up to date.
### Model reference simulation target for HevP2TransmissionController is up to date.
### Model reference simulation target for MotMappedP2 is up to date.
### Model reference simulation target for SiEngineController is up to date.
### Model reference simulation target for SiMappedEngine is up to date.
### Model reference simulation target for StarterSystemP2 is up to date.

```

Build Summary

```

0 of 8 models built (8 models already up to date)
Build duration: 0h 0m 13.42s

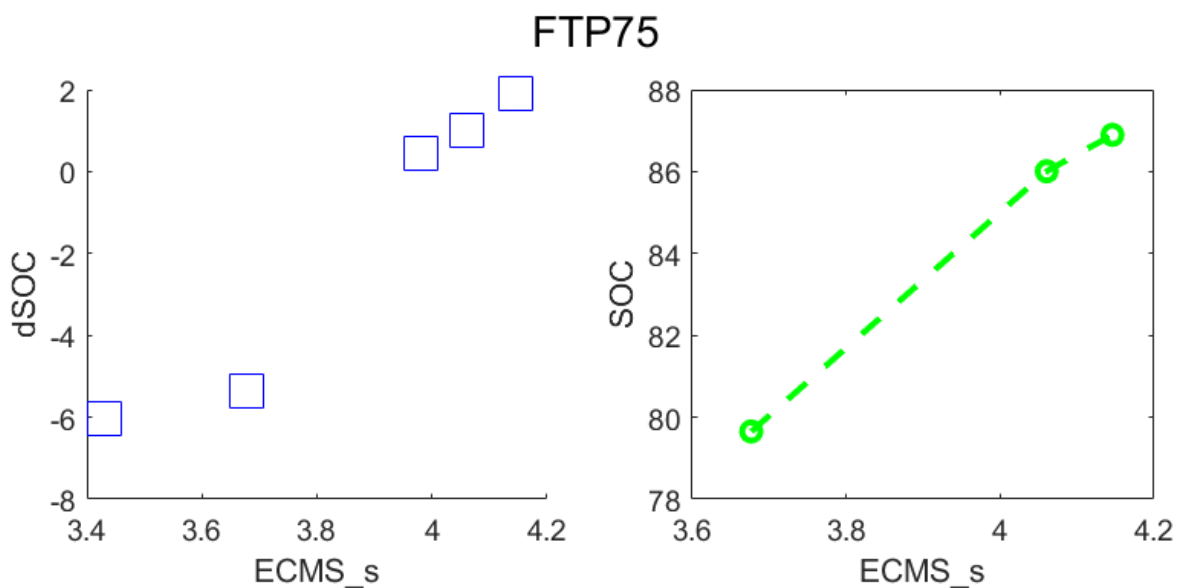
```

```
i= 2, ECMS_s = 3.980521, SOC_end = 85.445176
```

```
i= 2, x1 = 4.060548, x2 = 3.676594, x3 = 4.145992,
```

```
i= 2, y1 = 86.007364, y2 = 79.652346, y3 = 86.898529,
```

hold off;



```

toc;

Elapsed time is 957.897390 seconds.

if (solution_found == 1)
    fprintf ('Search converged. ECMS_s parameter updated in model. \n');
end

Search converged. ECMS_s parameter updated in model.

if (solution_found == 0)
    fprintf ('Search failed to converge. An approximate ECMS_s is updated in the model. \n');
    fprintf ('Refer to the Troubleshooting section of the example page for recommendations. \n');
end

ECMS_s_tune = ECMS_s;
% reset model
Simulink.sdi.markSignalForStreaming(x1,'off');
load_system(ctrlMdl)
set_param(ctrlMdl+ "/ECMS", 'ECMS_s', baseParamName)
save_system mdlName, [], 'SaveDirtyReferencedModels', 'on';

% update model and sim
hws = get_param(ctrlMdl, 'modelworkspace'); % get the workspace
hws.assignin('ECMS_s', ECMS_s);
save_system(ctrlMdl, [], 'SaveDirtyReferencedModels', 'on');
open_system(mdlName);
load_system(battMdl);
battChrgMaxValue = getVariable(get_param(battMdl, 'modelworkspace'), 'BattChargeMax');
in = in.setVariable('BattCapInit', battChrgMaxValue*SOCinit, 'Workspace', battMdl);
in = in.setVariable('SOCTrgt', SOCEndTrg, 'Workspace', ctrlMdl);
in = in.setVariable('SOCmin', max(SOCEndTrg-20, 20.5), 'Workspace', ctrlMdl);
in = in.setVariable('SOCmax', min(SOCEndTrg+20, 100), 'Workspace', ctrlMdl);
set_param(ctrlMdl+ "/ECMS", 'ECMS_s', "ECMS_s");
save_system(battMdl);
save_system(ctrlMdl);
% sim(in);
% open_system(mdlName+ "/Visualization/Performance and FE Scope");

function in = ModelSetVariable (mdlName, battChrgMaxValue, SOCinit, battMdl, SOCEndTrg, ctrlMdl)
in = Simulink.SimulationInput(mdlName);
in = in.setVariable('BattCapInit', battChrgMaxValue*SOCinit, 'Workspace', battMdl);
in = in.setVariable('SOCTrgt', SOCEndTrg, 'Workspace', ctrlMdl);
in = in.setVariable('SOCmin', max(SOCEndTrg-20, 20.5), 'Workspace', ctrlMdl);
in = in.setVariable('SOCmax', min(SOCEndTrg+20, 100), 'Workspace', ctrlMdl);
end

function [x_out, y_out, ECMS_s, SOC_end, solution_found] = dSOC_generate_starting_points (in, ECMS_s_tune)

x_out = zeros (3,1);
y_out = zeros (3,1);
x = zeros (4,1);
y = zeros (4,1);

solution_found = 0;
ECMS_s = ECMS_CurrentValue;

```

```

alpha = 0.8;

tol = 1;

x1 = ECMS_CurrentValue; % use current ECMS_s value as the starting point
in = in.setVariable("ECMS_s_tune", x1);
[SOc_end, dSOc] =dSOcSim_v1(in); %evaluate x1 results
y1 = SOc_end;
subplot (1,2,1);
plot(x1,dSOc,'bs','MarkerSize',15)
hold on
fprintf ('x1 = %f, y1 = %f \n',x1, y1);
if (abs(y1 - SOcEndTrg) <= tol) % check if a solution found
    ECMS_s = x1;
    SOc_end = y1;
    solution_found = 1;
end

if ((y1 > SOcEndTrg) && (solution_found == 0))
    x2 = x1 - ECMS_S_dis_up (x1, y1, SOcEndTrg); % use a decreased ECMS_s value as x2
    in = in.setVariable("ECMS_s_tune", x2);
    [SOc_end, dSOc] =dSOcSim_v1(in);
    y2 = SOc_end;
    subplot (1,2,1);
    plot(x2,dSOc,'bs','MarkerSize',15)
    hold on
    fprintf ('x2 = %f, y2 = %f \n',x2, y2);
    if (abs(y2 - SOcEndTrg) <= tol) % check if a solution found
        ECMS_s = x2;
        SOc_end = y2;
        solution_found = 1;
    end
    if ((y2 > SOcEndTrg) && (solution_found == 0)) % y2 is still too high
        y3_try = SOcEndTrg - 2; % set a lower SOc end trial to it will be easier to get y2 < SOc
        x3 = x1 + alpha*(y3_try - y1)*(x2-x1)/(y2-y1); % use x1, y1, x2, y2 to fit a linear func


$$x = x_1 + \alpha \frac{y - y_1}{y_2 - y_1} \times (x_2 - x_1)$$

        such that  $x = x_1$  at  $y = y_1$ , and  $x = \alpha x_2 + (1 - \alpha)x_1$  at  $y = y_2$ 

        dx = ECMS_S_dis_up (x2, y2, SOcEndTrg); % get a pre-defined decrease value
        x3 = min (x3, x2 - dx); % upper limit for x3
        x3 = max (x3, x2 - 2*dx); % lower limit for x3
        in = in.setVariable("ECMS_s_tune", x3); % set the x3 value as ECMS_s
        [SOc_end, dSOc] =dSOcSim_v1(in); % evaluate
        y3 = SOc_end;
        subplot (1,2,1);
        plot(x3,dSOc,'bs','MarkerSize',15)
        hold on
        fprintf ('x3 = %f, y3 = %f \n',x3, y3);
        if (abs(y3 - SOcEndTrg) <= tol) % check if a solution is found
            ECMS_s = x3;
            SOc_end = y3;
            solution_found = 1;
        end
    end
end
if (solution_found == 0)
    if ((y2 > SOcEndTrg) && (y3 >= SOcEndTrg)) % if y3 is still greater than SOcEndTrg, lower
        [x(4)] = Second_order_roots (x_out, y_out, SOcEndTrg - 2); % solve the second order

```

```

dx = ECMS_S_dis_up (x3, y3, SOCEndTrg);
x(4) = min (x(4), x3 - dx); % upper limit for x(4)
x(4) = max (x(4), x3 - 2*dx); % lower limit for x(4)
in = in.setVariable("ECMS_s_tune", x(4));
[SOCEnd, dSOC] =dSOCsim_v1(in);
ECMS_s = x(4);
y(4) = SOCEnd;
subplot (1,2,1);
plot(x(4),dSOC,'bs','MarkerSize',15)
hold on
fprintf ('x(4) = %f, y(4) = %f \n',x(4), y(4));
x_out (1) = x(4); y_out (1) = y(4);
x_out (2) = x3; y_out (2) = y3;
x_out (3) = x2; y_out (3) = y2;
if (abs(y(4) - SOCEndTrg) <= tol)
    ECMS_s = x(4);
    SOCEnd = y(4);
    solution_found = 1;
end
end
end

if ((y2 < SOCEndTrg) && (solution_found == 0)) % y2 is lower than SOCEndTrg while y1 is greater
    x_min = min (x1, x2); x_max = max (x1, x2); y_min = min(y1, y2); y_max = max (y1, y2); %
    w_min = abs (y_min - SOCEndTrg) / (abs(y_min - SOCEndTrg) + abs(y_max - SOCEndTrg)); % w
    x3 = (1-w_min) * x_min + w_min * x_max; % x3 is a weighted interpolation between x1 and x2
    in = in.setVariable("ECMS_s_tune", x3);
    [SOCEnd, dSOC] =dSOCsim_v1(in);
    y3 = SOCEnd;
    subplot (1,2,1);
    plot(x3,dSOC,'bs','MarkerSize',15)
    hold on
    fprintf ('x3 = %f, y3 = %f \n',x3, y3);
    if (abs(y3 - SOCEndTrg) <= tol)
        ECMS_s = x3;
        SOCEnd = y3;
        solution_found = 1;
    end
end
end
if (solution_found == 0)
    x_out(1) = x1; x_out(2) = x2; x_out(3) = x3; y_out(1) = y1; y_out(2) = y2; y_out(3) = y3
end
end

if ((y1 < SOCEndTrg) && (solution_found == 0)) % y1 < SOCEndTrg
    x2 = x1 + ECMS_S_dis_up (x1, y1, SOCEndTrg); % use a higher ECMS_s value as x2
    in = in.setVariable("ECMS_s_tune", x2);
    [SOCEnd, dSOC] =dSOCsim_v1(in);
    y2 = SOCEnd;
    subplot (1,2,1);
    plot(x2,dSOC,'bs','MarkerSize',15)
    fprintf ('x2 = %f, y2 = %f \n',x2, y2);
    if (abs(y2 - SOCEndTrg) <= tol)
        ECMS_s = x2;
        SOCEnd = y2;
        solution_found = 1;
    end
end
if ((y2 < SOCEndTrg) && (solution_found == 0)) % x2 is not high enough

```



```

y3_try = SOCEndTrg + 2; % set a higher SOC target

x = x1 + alpha*(y - y1)/(y2 - y1)*(x2 - x1), x = x1, at y = y1, x = alpha*x2 + (1 - alpha)*x1 at y = y2,

x3 = x1 + alpha*(y3_try - y1)*(x2-x1)/(y2-y1); % use x1, y1, x2, y2 as a linear prediction
dx = ECMS_S_dis_up (x2, y2, SOCEndTrg); % standard increase
x3 = max (x3, x2 + dx); % lower limit for x3
x3 = min (x3, x2 + 2*dx); % upper limit for x3
in = in.setVariable("ECMS_s_tune", x3);
[SOC_end, dSOC] = dSOCsim_v1(in);
y3 = SOC_end;
subplot (1,2,1);
plot(x3,dSOC,'bs','MarkerSize',15)
fprintf ('x3 = %f, y3 = %f \n',x3, y3);
if (abs(y3 - SOCEndTrg) <= tol)
    ECMS_s = x3;
    SOC_end = y3;
    solution_found = 1;
end
end
if (solution_found == 0)
    if ((y2 < SOCEndTrg) && (y3 <= SOCEndTrg)) % y3 is still not high enough
        [x(4)] = Second_order_roots (x_out, y_out, SOCEndTrg + 2); % increase again
        dx = ECMS_S_dis_up (x3, y3, SOCEndTrg); % standard increase
        x(4) = max (x(4), x3 + dx); % make it higher than the standard increase
        x(4) = min (x(4), x3 + 2*dx); % limit the increase to 2 times the standard increase
        in = in.setVariable("ECMS_s_tune", x(4));
        [SOC_end, dSOC] = dSOCsim_v1(in);
        ECMS_s = x(4);
        y(4) = SOC_end;
        subplot (1,2,1);
        plot(x(4),dSOC,'bs','MarkerSize',15)
        x_out (1) = x(4); y_out (1) = y(4); % output 3 points
        x_out (2) = x3; y_out (2) = y3;
        x_out (3) = x2; y_out (3) = y2;
        fprintf ('x(4) = %f, y(4) = %f \n',x(4), y(4));
        if (abs(y(4) - SOCEndTrg) <= tol)
            ECMS_s = x(4);
            SOC_end = y(4);
            solution_found = 1;
        end
    end
end
end
if ((y2 > SOCEndTrg) && (solution_found == 0)) % y2 is good
    x_min = min (x1, x2); x_max = max (x1, x2); y_min = min(y1, y2); y_max = max (y1, y2); %
    w_min = abs (y_min - SOCEndTrg) / (abs(y_min - SOCEndTrg) + abs(y_max - SOCEndTrg)); %
    x3 = (1-w_min) * x_min + w_min * x_max; % get weighted interpolation between x1, and x2
    in = in.setVariable("ECMS_s_tune", x3);
    [SOC_end, dSOC] = dSOCsim_v1(in);
    y3 = SOC_end;
    subplot (1,2,1);
    plot(x3,dSOC,'bs','MarkerSize',15)
    fprintf ('x3 = %f, y3 = %f \n',x3, y3);
    if (abs(y3 - SOCEndTrg) <= tol)
        ECMS_s = x3;
        SOC_end = y3;
    end
end

```

```

        solution_found = 1;
    end
end
if (solution_found == 0)
    x_out(1) = x1; x_out(2) = x2; x_out(3) = x3; y_out(1) = y1; y_out(2) = y2; y_out(3) = y3
end
end

function [SOC_end, dSOC]=dSOCsim_v1(in)

out=sim(in);
% for ii=1:length(out)
    if isempty(out.ErrorMessage)
        SOC=out.logsout.getElement('Battery SOC').Values.Data;
        dSOC=SOC(end)-SOC(1);
        SOC_end = SOC(end);
    else
        dSOC=NaN;
    end
end

function dx = ECMS_S_dis_up (ECMS_s, y, SOC_target)

% function computes standard dx = increase/decrease in ECMS_s
% the dx is proportional to the distance between SOC_end (y) and SOC_Target
% also, dx is proportional to ECMS_s, a, c, ff can be adjusted

ff = 0.04;
a = 0.01;
c = 3.5;
b = ECMS_s / c; % ECMS_s effect
if (ECMS_s < c-0.1); b = 1; end
dx = a + b * abs(y-SOC_target) * ff;

end

function [z1] = Second_order_roots (x, y, y_tar)

    solve the equation  $y_{tar} = az^2 + bz + c$ , with  $y_i = ax_i^2 + bx_i + c$ , for  $1 \leq i \leq 3$ .

d1 = y(1) / (x(1) - x(2)) / (x(1) - x(3));
d2 = y(2) / (x(2) - x(1)) / (x(2) - x(3));
d3 = y(3) / (x(3) - x(1)) / (x(3) - x(2));

AA = (d1 + d2 + d3);
BB = - (d1 * (x(2) + x(3)) + d2 * (x(1) + x(3)) + d3 * (x(1) + x(2)));
CC = d1 * x(2) * x(3) + d2 * x(1) * x(3) + d3 * x(1) * x(2) - y_tar;

z1 = (-BB + sqrt (BB*BB - 4 * AA * CC)) / 2 / AA;

```

end

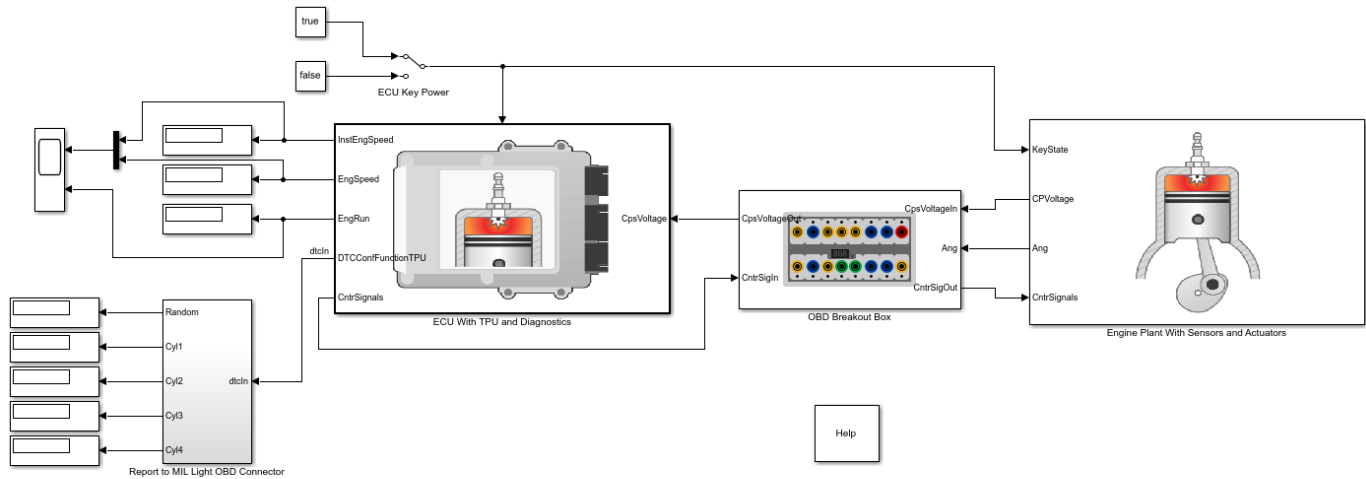
See Also

Equivalent Consumption Minimization Strategy

Use On-Board Diagnostics to Detect Misfire

This example shows how to introduce, detect, mature, and report misfire events from the SI Core Engine. The model demonstrates misfire on-board diagnostics (OBD) warnings.

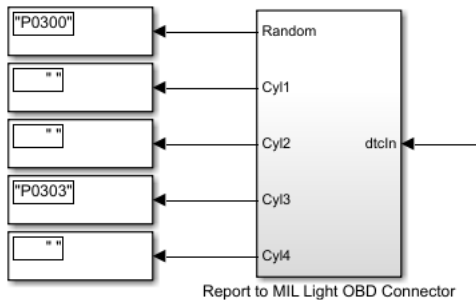
Model



Copyright 2020 The MathWorks, Inc.

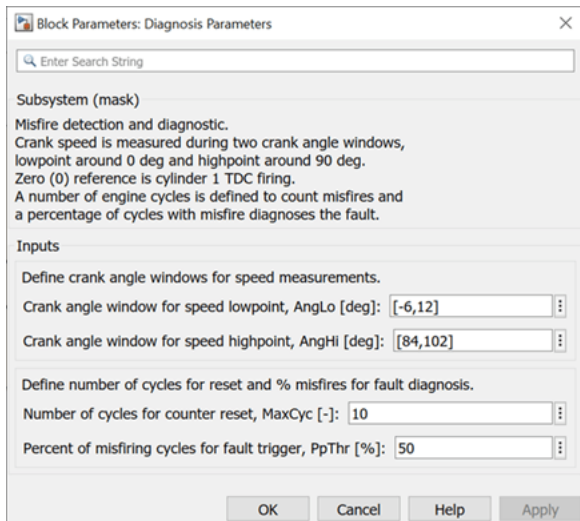
Run Simulation

On the Simulation tab, click **Run**. As the simulation runs, diagnostic trouble codes appear on the MIL Light display. P0300 is a general warning for a misfire issue in one or more cylinders. P0303 indicates a specific misfire issue with Cylinder 3.



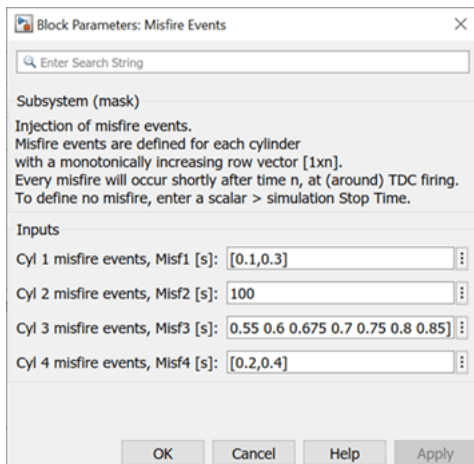
ECU With TPU and Diagnostics

This subsystem detects misfire events, using crankshaft acceleration as an indicator. Navigate to `autoblksICoreOBDMisfire > ECU With TPU and Diagnostics > TPU Level Misfire Detection`. Open the Diagnosis Parameters block. You can modify the OBD threshold calibrations for detection and maturation.



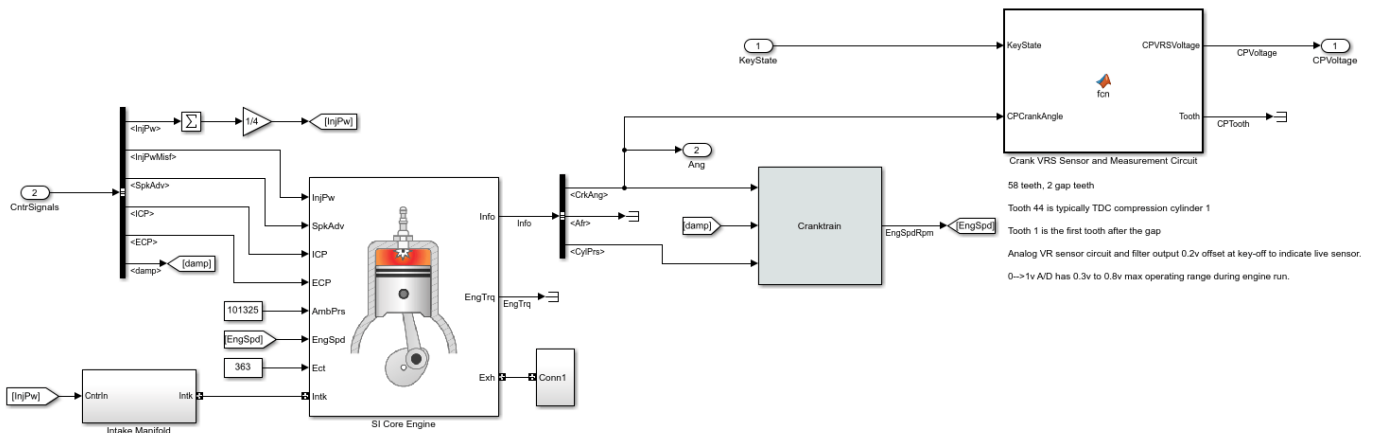
OBD Breakout Box

This subsystem defines misfire events to inject into the simulation. Navigate to `autoblkSICoreOBDMisfire > OBD Breakout Box > Injector Pulswidth Fault Injection`. Open the Misfire Events block. You can modify the misfire fault injection timing.



Engine Plant With Sensors and Actuators

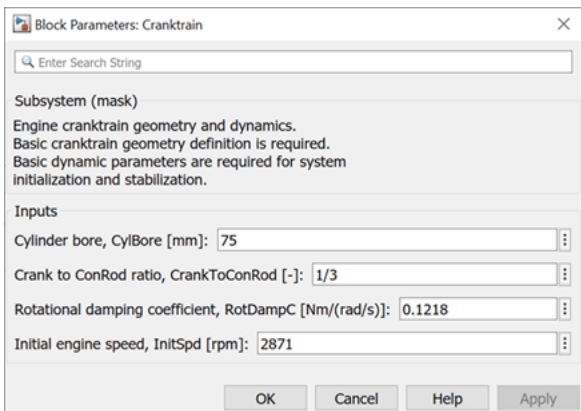
The SI Core Engine computes cylinder pressure as a function of crank angle. The Crank VRS Sensor and Measurement Circuit simulates the 58-tooth wheel that tracks the progression of the 4-stroke engine cycle.



Crank VRS Sensor and Measurement Circuit
 58 teeth, 2 gap teeth
 Tooth 44 is typically TDC compression cylinder 1
 Tooth 1 is the first tooth after the gap
 Analog VR sensor circuit and filter output 0.2v offset at key-off to indicate live sensor.
 0→1v A/D has 0.3v to 0.8v max operating range during engine run.

Cranktrain block

Open the Cranktrain block to modify the engine cranktrain geometry and dynamics.



See Also

SI Core Engine

Read and Write Block Parameters to Excel

If you manage model data in external files, you can use scripts to pass the data between the data file and a Simulink® model. This example shows you how to read block parameter data from and write parameter data to an Excel® data file. Specifically, the example provides functions that read and write Mapped SI Engine parameter data. You can adapt the functions to read and write parameters for additional blocks.

Open Mapped SI Engine Block

Open the Mapped SI Engine block in the hybrid electric vehicle P2 reference application.

Open the Hybrid Electric Vehicle P2 Reference Application

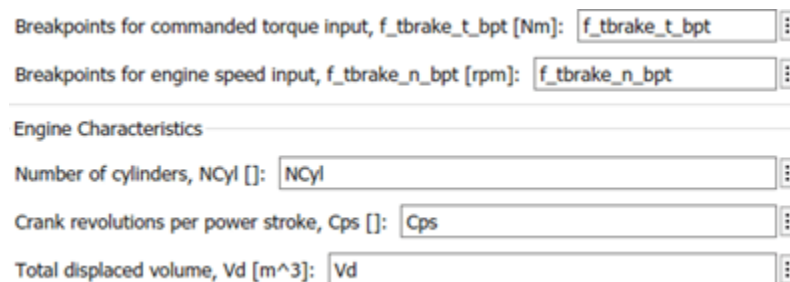
```
workDir = pwd;
autoblkHevP2Start;
cd(workDir);
```

Set a variable equal to the block path.

```
bp = 'SiMappedEngine/Mapped SI Engine'; % block path
```

Open Mapped SI Engine Block

In the HevP2ReferenceApplication model, navigate to Passenger Vehicle > Ideal Mapped Engine > SiMappedEngine . Open the Mapped SI Engine block. The **Breakpoints for commanded torque**, **Breakpoints for engine speed input**, **Number of cylinders**, **Crank revolutions per power stroke**, and **Total displaced volume** parameters are set to workspace variables.



The functions in the example overwrite the workspace variables with the values in the data file.

Specify Data File Configuration

First, specify the file name. This example file `SiEngineData.xlsx` contains three sheets. The first sheet contains scalar values for commanded torque breakpoints, breakpoints for engine speed input breakpoints, number of cylinders, crank revolutions, and total displaced volume. The second sheet contains a table values for the brake torque map. The third sheet contains table values for the fuel torque map.

```
fileName = 'SiEngineData.xlsx';
```

Note that the first sheet in the file specifies the **Number of cylinders**, `Ncyl` parameter as 6.

	MappedSIEngineBlock	VarName2	VarName3	VarName4
	Text	Text	Number	Number
1	Mapped SI Engine Block			
2	Input Configuration Parameters	Unit	Value	
3	Breakpoints for commanded torqu...	Nm	0	35
4	Breakpoints for engine speed inpu...	rpm	0	641.2000
5				
6	Engine Characteristics	Unit	Value	
7	Number of cylinders, Ncyl	-	6	
8	Crank revolutions per power strok...	-	2	
9	Total displaced volume, Vd	m^3	0.0036	

Next, define the configuration data for the engine subsystem. This example sets a configuration for double variables of size scalar, vector, or a 2D array.

- Scalar data structure specifies the data on the first sheet.
- Vector data structure specifies the data on the second sheet.
- Array data structure specifies the data on the third sheet.

```
engData = struct(); % engine parameter data
```

```
% Scalar data
```

```
engData.Ncyl = struct('xlSheet','Main', 'xlRange','C7:C7', 'slBlockPath',bp, 'slBlockParam','Ncyl');
engData.Cps = struct('xlSheet','Main', 'xlRange','C8:C8', 'slBlockPath',bp, 'slBlockParam','Cps');
engData.Vd = struct('xlSheet','Main', 'xlRange','C9:C9', 'slBlockPath',bp, 'slBlockParam','Vd');
```

```
% Vector data
```

```
engData.t_bpt = struct('xlSheet','Main', 'xlRange','C3:R3', 'slBlockPath',bp, 'slBlockParam','f_');
engData.n_bpt = struct('xlSheet','Main', 'xlRange','C4:R4', 'slBlockPath',bp, 'slBlockParam','f_');
```

```
% 2D array data
```

```
engData.torque = struct('xlSheet','Brake Torque', 'xlRange','B2:Q17', 'slBlockPath',bp, 'slBlockParam','torque');
engData.fuel = struct('xlSheet','Fuel Map', 'xlRange','B2:Q17', 'slBlockPath',bp, 'slBlockParam','fuel');
```

Read Mapped SI Engine Block Parameters

Update the Mapped SI Engine block to the values specified in the data file.

Read Data File and Update Parameters

Use this code to read the data file and update the Mapped SI Engine block parameters.

```
f = fields(engData);
for idx = 1:length(f)
    try
        var = getfield(engData, f{idx});
        % read value from Excel
        val = readmatrix(fileName, 'Sheet',var.xlsheet, 'Range',var.xlRange);
        % open Simulink model
        mdl = fileparts(var.slBlockPath);
        open_system(mdl);
        % set parameter value and save model
        set_param(var.slBlockPath, var.slBlockParam, mat2str(val));
        save_system(mdl);
    catch ME
        % return any error info
    end
end
```



```

        disp(getReport(ME, 'extended', 'hyperlinks', 'on'))
        fprintf('\nContinuing to next variable...\n\n');
    end
end
fprintf('Done writing values to Simulink\n')

```

Done writing values to Simulink

Open Mapped SI Engine Block

In the HevP2ReferenceApplication model, navigate to Passenger Vehicle > Ideal Mapped Engine > SiMappedEngine. Open the Mapped SI Engine block. The **Breakpoints for commanded torque**, **Breakpoints for engine speed input**, **Number of cylinders**, **Crank revolutions per power stroke**, and **Total displaced volume** parameters are set to the values specified in the data file. Confirm that the **Brake torque map** and **Fuel flow map parameters** are the same as the values specified in the data file.

Breakpoints for commanded torque input, f_tbrake_t_bpt [Nm]:

Breakpoints for engine speed input, f_tbrake_n_bpt [rpm]:

Engine Characteristics

Number of cylinders, NCyl []:

Crank revolutions per power stroke, Cps []:

Total displaced volume, Vd [m^3]:

Write Modified Parameters to Data File

In the Mapped SI Engine block, change the **Number of cylinders**, **NCyl** parameter from 6 to 8. Click **Apply**. Save the model.

Breakpoints for commanded torque input, f_tbrake_t_bpt [Nm]:

Breakpoints for engine speed input, f_tbrake_n_bpt [rpm]:

Engine Characteristics

Number of cylinders, NCyl []:

Crank revolutions per power stroke, Cps []:

Total displaced volume, Vd [m^3]:

Alternatively, use this code to update the parameter and save the model.

```

set_param(bp, 'Ncyl', '8');
save_system('SiMappedEngine');

```

Write Parameter Data to File

First, create a copy of the data file. Write the modified parameter data to the copy of the data file.

```

copyfile('SiEngineData.xlsx', 'SiEngineDataCopy.xlsx', 'f');
fileName = 'SiEngineDataCopy.xlsx';

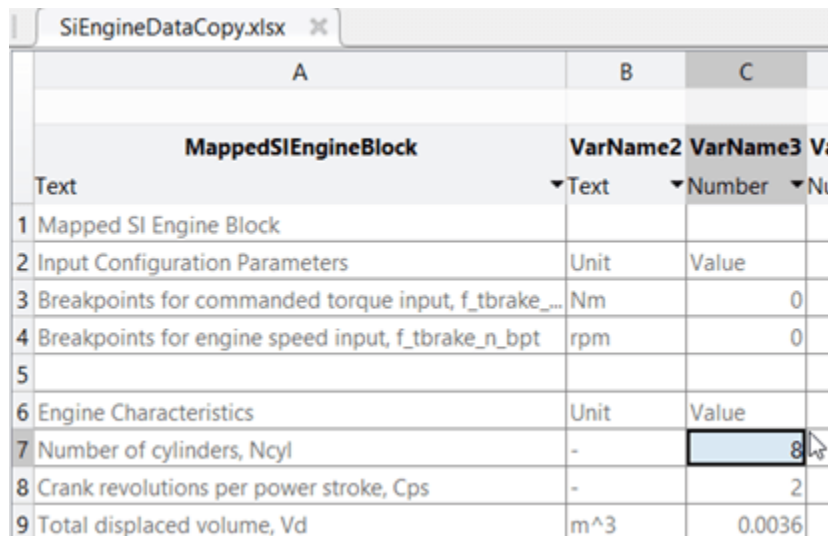
```

Next, use this code to write the Mapped SI Engine block **Breakpoints for commanded torque**, **Breakpoints for engine speed input**, **Number of cylinders**, **Crank revolutions per power stroke**, **Total displaced volume**, **Brake torque map**, and **Fuel flow map** parameters to the data file.

```
% Read data from Simulink model then write to Excel
f = fields(engData);
for idx = 1:length(f)
    try
        var = getfield(engData, f{idx});
        % open Simulink model
        mdl = fileparts(var.slBlockPath);
        open_system(mdl);
        % read value from Simulink
        val = str2num(get_param(var.slBlockPath, var.slBlockParam));
        % write value to Excel
        writematrix(val, fileName, 'Sheet', var.xlsheet, 'Range', var.xlRange);
    catch ME
        % return any error info
        disp(getReport(ME, 'extended', 'hyperlinks', 'on'))
        fprintf('\nContinuing to next variable...\n\n');
    end
end
fprintf('Done writing values to Excel\n')
```

Done writing values to Excel

Open the file with the modified data. Confirm that the number of cylinders in the data file is 8.



	A	B	C
	MappedSIEngineBlock	VarName2	VarName3
Text	Text	Number	Number
1	Mapped SI Engine Block		
2	Input Configuration Parameters	Unit	Value
3	Breakpoints for commanded torque input, f_tbrake_...	Nm	0
4	Breakpoints for engine speed input, f_tbrake_n_bpt	rpm	0
5			
6	Engine Characteristics	Unit	Value
7	Number of cylinders, Ncyl	-	8
8	Crank revolutions per power stroke, Cps	-	2
9	Total displaced volume, Vd	m^3	0.0036

See Also

Mapped SI Engine

Related Examples

- “HEV P2 Reference Application” on page 7-7