Model-Based Calibration Toolbox™

Getting Started Guide
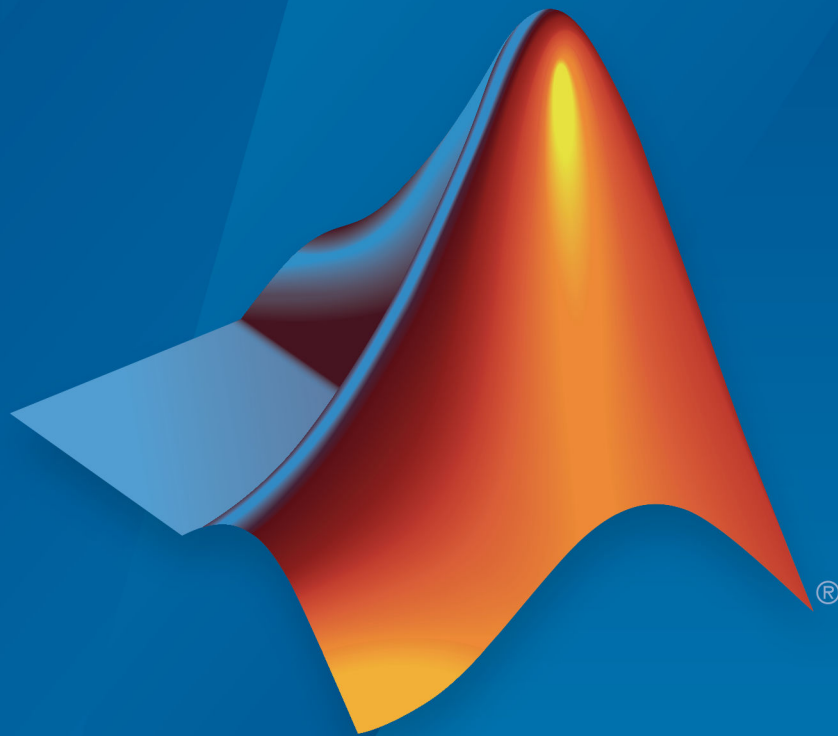
MATLAB®&SIMULINK®

MathWorks®

How to Contact MathWorks

	Latest news:	www.mathworks.com
	Sales and services:	www.mathworks.com/sales_and_services
	User community:	www.mathworks.com/matlabcentral
	Technical support:	www.mathworks.com/support/contact_us
	Phone:	508-647-7000

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

Contents

Composite Models and Modal Optimizations

4

Design and Modeling Scripts

5

Multi-Injection Diesel Calibration

6

Model Quickstart

7

Design of Experiment

8

Data Editor for Modeling

9

Feature Calibration

10

Tradeoff Calibration

11

Data Sets

12

Filling Tables from Data

13

Optimization and Automated Tradeoff

14

Introduction

The following sections introduce Model-Based Calibration Toolbox software.

- "Model-Based Calibration Toolbox Product Description" on page 1-2
- "What Is Model-Based Calibration?" on page 1-3
- "Limitations" on page 1-6

Model-Based Calibration Toolbox Product Description

Model and calibrate engines

Model-Based Calibration Toolbox provides apps and design tools for optimally calibrating complex engines and powertrain subsystems. You can define optimal test plans, automatically fit statistical models, and generate calibrations and lookup tables for complex high-degree-of-freedom engines that would otherwise require exhaustive testing using traditional methods. Calibrations can be optimized at individual operating points or over drive cycles to identify the optimal balance of engine fuel economy, performance, and emissions. Using apps or MATLAB® functions, you can automate the calibration process for similar engine types.

Models created with Model-Based Calibration Toolbox can be exported to Simulink® to support control design, hardware-in-the-loop testing, and powertrain simulation activities across the powertrain design team. Calibration tables can be exported to ETAS INCA and ATI Vision.

Key Features

- Apps that support the entire workflow: designing experiments, fitting statistical models to engine data, and producing optimal calibrations
- Design-of-Experiments methodology for reducing testing time through classical, space-filling, and optimal design techniques
- Accurate engine modeling with data fitting techniques including Gaussian process, radial basis function, and linear regression modeling
- Boundary modeling to keep optimization results within the engine operating envelope
- Generation of lookup tables from optimizations over drive cycles, models, or test data
- Export of performance-optimized models to Simulink for use in simulation and HIL testing
- Lookup table import and export to ETAS INCA and ATI Vision

What Is Model-Based Calibration?
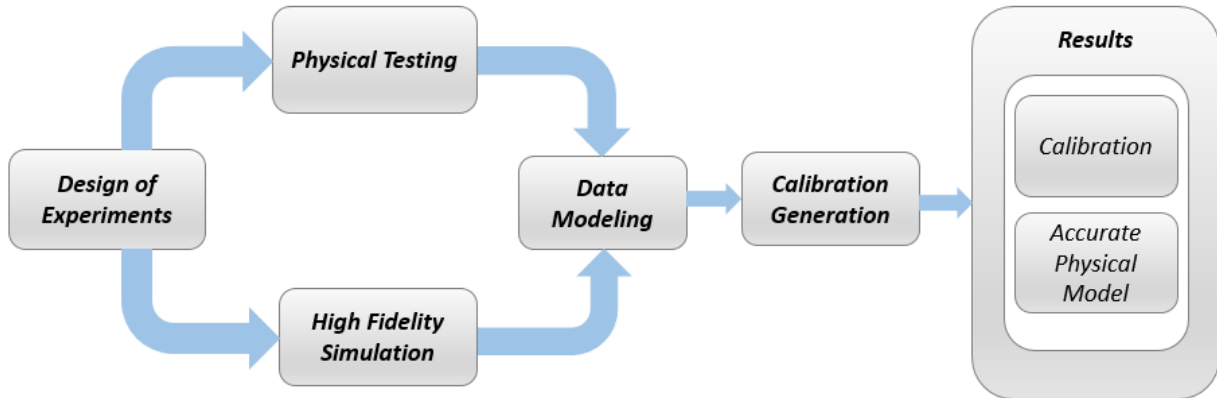
High accuracy engine models are a key component for reducing calibration effort and engine development time.

The time spent calibrating an engine control unit has been increasing, due to new control actuators. The new actuators give the potential for increased performance, reduced emissions, and improved fuel consumption. It is necessary to apply advanced modeling and optimization techniques to achieve the full benefits available from the addition of new actuators. Advanced modeling techniques offer increased understanding of the complex, nonlinear engine responses. High accuracy models can be used throughout the design process, including the calibration of base maps with the optimal settings for the control parameters, determined by constrained optimizations.

The toolbox has two main user interfaces for model-based calibration workflows:

- Model Browser for design of experiment and statistical modeling
- CAGE Browser for analytical calibration

The Model Browser part of the toolbox is a powerful tool for experimental design and statistical modeling. The models you build with the Model Browser can be imported into the CAGE Browser part of the toolbox to produce optimized calibration tables.

Designs and Modeling in the Model Browser

The Model Browser is a flexible, powerful, intuitive graphical interface for building and evaluating experimental designs and statistical models:

- Design of experiment tools can drastically reduce expensive data collection time.
- You can create and evaluate optimal, space-filling, and classical designs, and constraints can be designed or imported.
- Hierarchical statistical models can capture the nature of variability inherent in engine data, accounting for variation both within and between tests.
- The Model Browser has powerful, flexible tools for building, comparing, and evaluating statistical models and experimental designs.
- There is an extensive library of prebuilt model types and the capability to build user-defined models.
- You can export models to CAGE or to MATLAB or Simulink software.

Starting the Model Browser

To start the application, type

```
mbcmodel
```

at the MATLAB command prompt.

Calibration Generation in CAGE

CAGE (CAlibration GEneration) is an easy-to-use graphical interface for calibrating lookup tables for your electronic control unit (ECU).

As engines get more complicated, and models of engine behavior more intricate, it is increasingly difficult to rely on intuition alone to calibrate lookup tables. CAGE provides analytical methods for calibrating lookup tables.

CAGE uses models of the engine control subsystems to calibrate lookup tables. With CAGE, you fill and optimize lookup tables in existing ECU software using Model Browser models. From these models, CAGE builds steady-state ECU calibrations.

CAGE also compares lookup tables directly to experimental data for validation.

Starting the CAGE Browser

To start the application, type

```
cage
```

at the MATLAB command prompt.

Limitations

Waitbar May Have Transparent Background

The waitbar dialogs that are displayed to inform you of the progress of lengthy operations in the toolbox often display with transparent backgrounds. Instead of the normal window grey, they pick up their background from whatever is on the screen when they are displayed.

This bug is purely cosmetic and will not cause any data loss.

Advanced Topics

Advanced Topics

You can use the Model-Based Calibration Toolbox to calibrate tables, optimize, and manipulate data.

Goal	See
Calibrate an estimator for a control subsystem in an electronic control unit (ECU)	"Feature Calibration" on page 10-2
Balance calibration objectives	"Tradeoff Calibration" on page 11-2
Compare tables and models to experimental data	"Compare Calibrations To Data" on page 12-2
Import experimental data	"Fill Tables from Data" on page 13-2
Optimize and automate tradeoff calibration	"Optimization and Automated Tradeoff" on page 14-2

See Also

More About

- "What Is Model-Based Calibration?" on page 1-3

3

Gasoline Engine Calibration

Gasoline Case Study Overview

In this section...
"Gasoline Calibration Problem Definition" on page 3-2
"Case Study Example Files" on page 3-3

Gasoline Calibration Problem Definition

This case study demonstrates how to systematically develop a set of optimal steady-state engine calibration tables using the Model-Based Calibration Toolbox.

The engine to calibrate is a direct-injected 1.5L spark ignition turbocharged engine with dual cam-phasers and turbocharger wastegate.

The aim of the calibration is to maximize torque at specific speed/load values across the engine's operating range, and meet these constraints:

- Limit knock
- Limit residual fraction
- Limit exhaust temperature
- Limit calibration table gradients for smoothness.

The analysis must produce optimal engine calibration tables in speed and load for:

- Spark advance ignition timing (SA)
- Throttle position % (TPP)
- Turbo wastegate area % (WAP)
- Air/Fuel Ratio (Lambda: LAM)
- Intake cam phase (ICP)
- Exhaust cam phase (ECP)
- Torque (TQ)
- Brake-specific fuel consumption (BSFC)
- Boost (MAP)
- Exhaust temperature (TEXH)
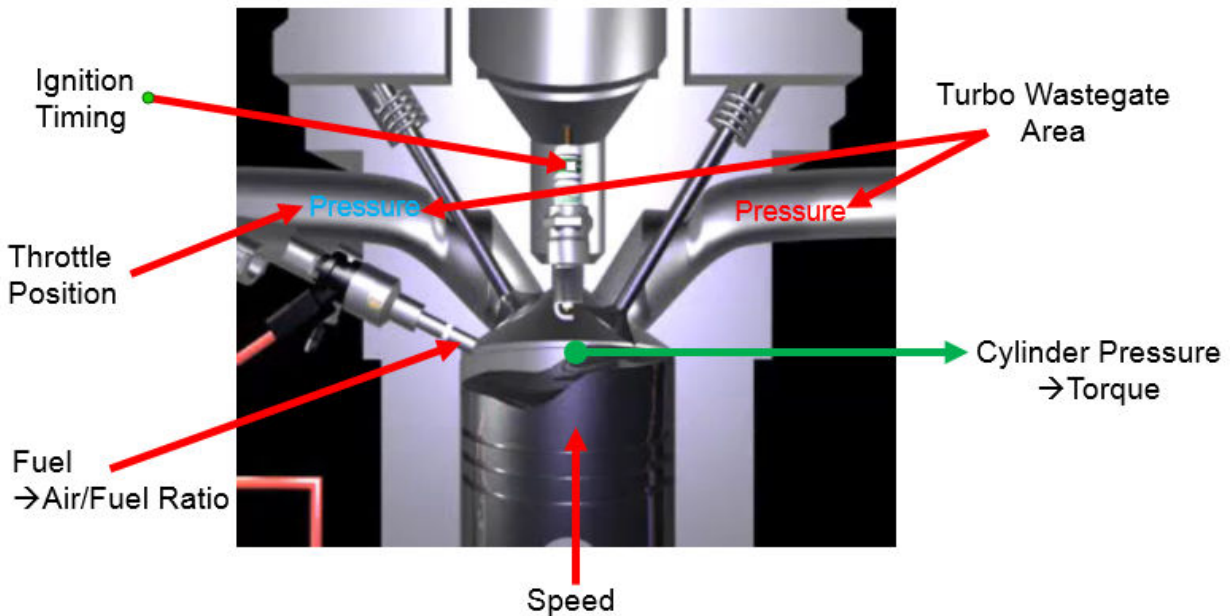
This case study illustrates the model-based calibration process.
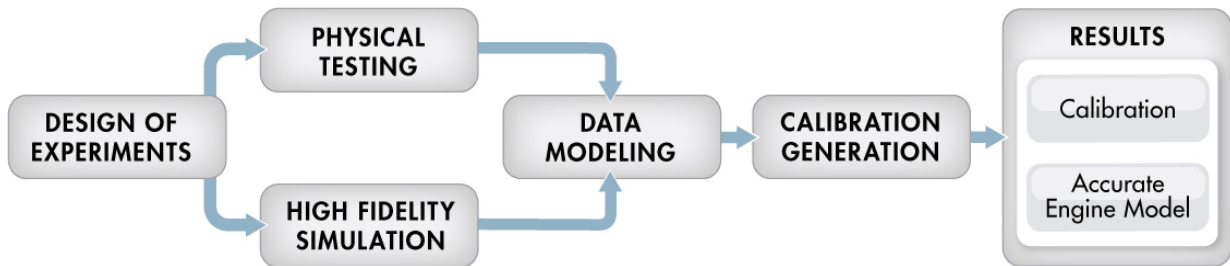
Case Study Example Files

The following sections guide you through opening example files to view each stage of the model-based calibration process. You can examine:

1. Designs, constraints, boundary model, and collected data, in topic "Design of Experiment" on page 3-5.

2 Finished statistical models, in topic "Empirical Engine Modeling" on page 3-15.

3 Optimization setup and results, and filled calibration tables, in topic "Optimization" on page 3-20.

Use these example files to understand how to set up systematic calibrations for similar problems. For next steps, see "Design of Experiment" on page 3-5.

Tip Learn how MathWorks® Consulting helps customers develop engine calibrations that optimally balance engine performance, fuel economy, and emissions requirements: see Optimal Engine Calibration.

Design of Experiment

In this section...
"Context" on page 3-5
"Benefits of Design of Experiment" on page 3-5
"Power Envelope Survey Testing" on page 3-5
"Create Designs and Collect Data" on page 3-6
"Data Collection and Physical Modeling" on page 3-13

Context

This topic describes design of experiments for the gasoline one-stage case study. To view the high-level workflow, see "Gasoline Case Study Overview" on page 3-2.

Benefits of Design of Experiment

You use design of experiment to efficiently collect engine data. Testing time (on a dyno cell, or as in this case, using high-fidelity simulation) is expensive, and the savings in time and money can be considerable when a careful experimental design takes only the most useful data. Dramatically reducing test time is increasingly important as the number of controllable variables in more complex engines is growing. With increasing engine complexity, the test time increases exponentially.

Power Envelope Survey Testing

The first stage to solve this calibration problem is to determine the boundaries of the feasible system settings. You need to generate the power envelope to constrain the design points. To do this, data was collected using simulation across a range of speed and torque. The initial survey determined boundaries that produce:

- Acceptable exhaust temperature (not too high to burn piston crowns)
- Achievable torque production
- Acceptable BSFC (not too high)
- Avoids knock

The envelope must include the idle region of low torque and speed, where the cams must be parked. The cam timings are set to zero in the idle design.

The initial survey design and test data provide information about the engine operating envelope. This information was used to create constraints for the final design, to collect detailed data about the engine behavior within those boundaries. You can then use this data to create response models for all the responses you need in order to create an optimal calibration for this engine.

The final design used 238 points for this engine with 4 inputs: speed, load, intake and exhaust cam. You can view the constraints defining the operating envelope by following the steps below.

Create Designs and Collect Data

You can use a space-filling design to maximize coverage of the factors' ranges as quickly as possible, to understand the operating envelope.

To create a design, you need to first specify the model inputs. Open the example file to see how to define your test plan.

1 Open MATLAB. On the **Apps** tab, in the **Automotive** group, click **MBC Model Fitting**.

2 In the Model Browser home page, in the **Case Studies** list, select **Dual CAM gasoline engine with spark optimized during testing**. Alternatively, select **File > Open Project** and browse to the example file `gasolineOneStage.mat`, found in `matlab\toolbox\mbc\mbctraining`.

3 To view how to define your test plan design inputs, in the **All Models** tree, click the top project node, `gasolineOneStage`. In the **Common Tasks** pane, click **Design experiment**. In the New Test Plan dialog box, observe the inputs pane, where you can change the number of model inputs and specify the input symbols, signals and ranges. This example project already has inputs defined, so click **Cancel**.

4 Click the first test plan node in the **All Models** tree, `gasolineOneStageDoE`. The test plan view appears.

5 Observe the inputs listed on the test plan diagram. Double-click the **Inputs** block to view the ranges and names (symbols) for variables in the Input Factor Set Up dialog box. The design inputs are torque, speed, intake cam, and exhaust cam. The dynamometer test setup is in speed/torque and so the design is in speed/torque.

   Close the dialog box by clicking **Cancel**.

6 After setting up inputs, you can create designs. In the **Common Tasks** pane, click **Design experiment**.

The Design Editor opens. Here, you can see how these designs are built.

7 Click the first design in the tree, `gasolineOneStageIdle`. If you do not see a 2D plot, select **View > Current View > 2D Design Projection**. Then select each design in the tree in turn.

- The first design in the tree, `gasolineOneStageIdle`, concentrates points in the idle area with cams parked. It uses the Sobol Sequence space-filling design type to maximize coverage. To park the cams, after creating the idle region space-filling design, the cam phaser positions are set to park position (0,0).
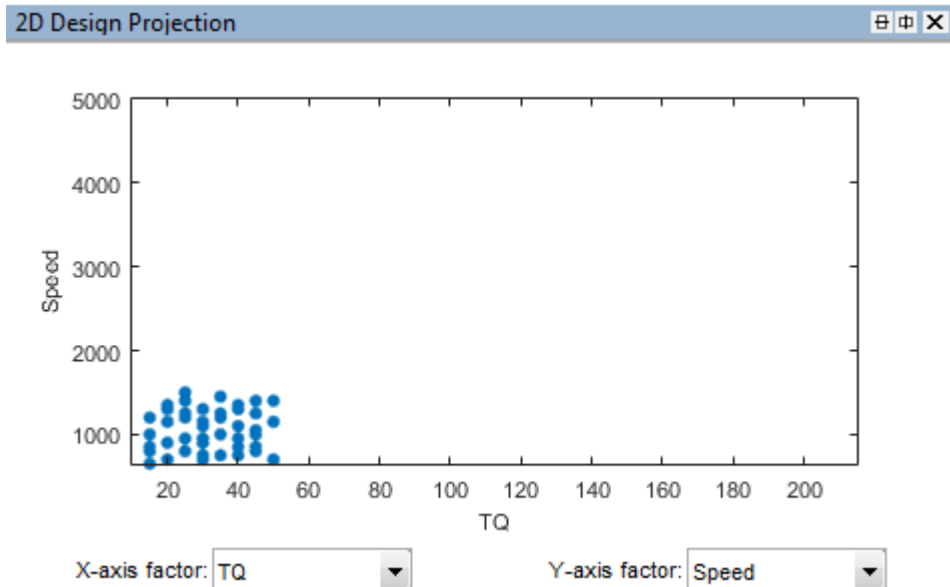
- The second design, `gasolineOneStageNonIdle`, is another Sobol Sequence space-filling design to span the non-idle engine operating envelope.
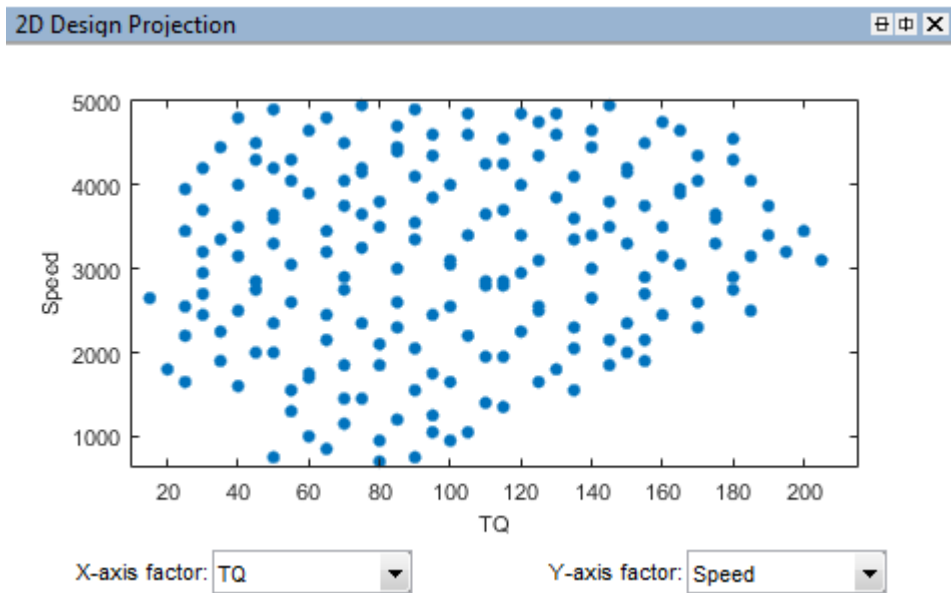
- The final design is called `gasolineOneStageMerged` because it contains the other two merged designs to cover the whole envelope.
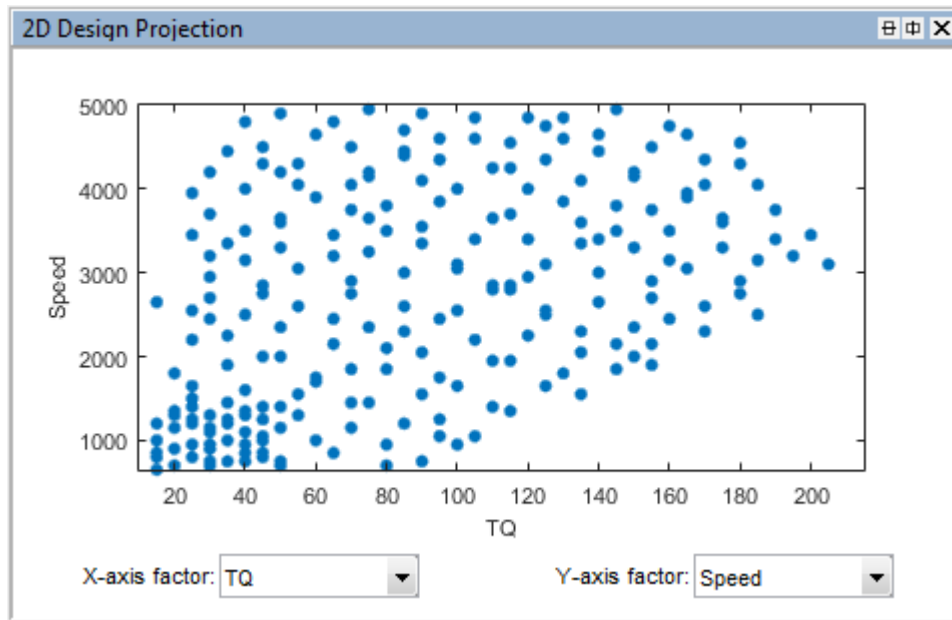
8   To see how the constraints are set up, select **Edit > Constraints**.

9   In the Constraints Manager dialog box, select each torque constraint in turn and click **Edit**. Observe the maximum and minimum torque constraints define the upper and lower boundary of the feasible operating envelope. These constraints were developed prior to this design in initial wide-open throttle and closed-throttle engine performance testing.
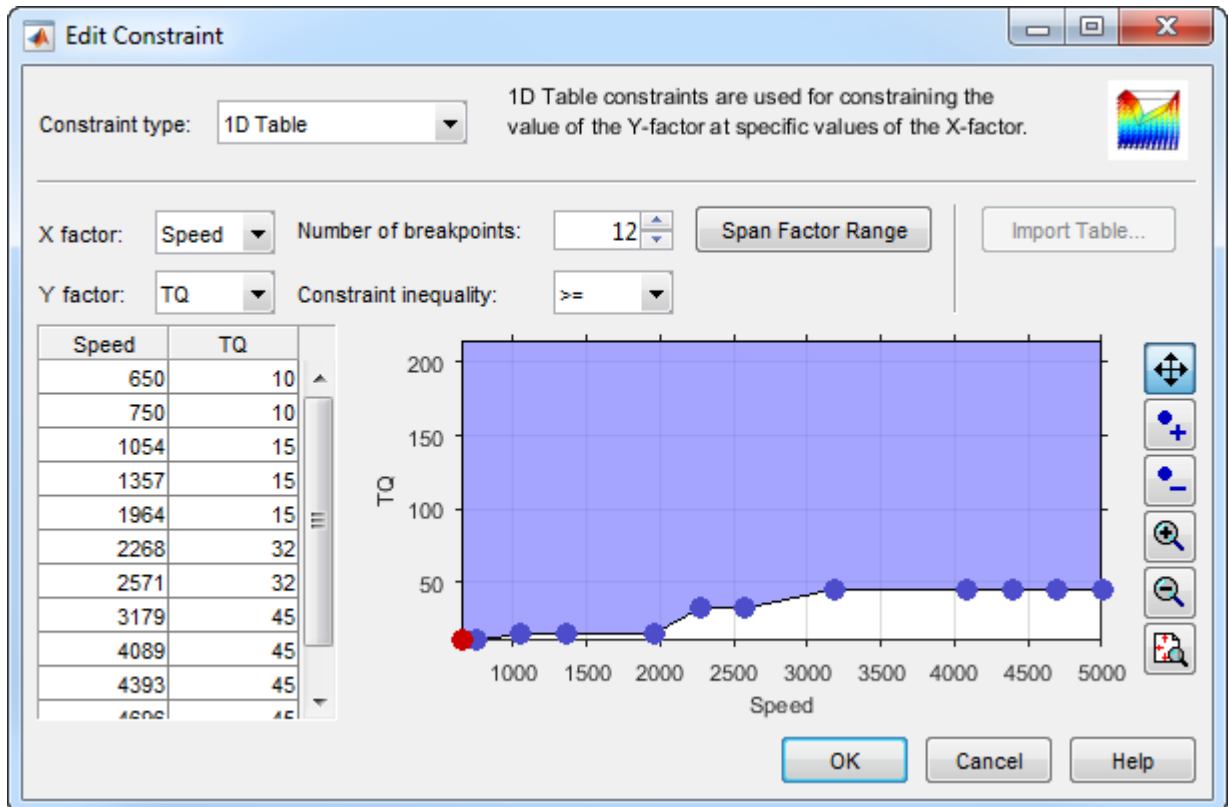
Observe that you can define areas to exclude by dragging points, typing in the edit boxes, or using the Table Editor tab.

To leave the constraint unchanged, click **Cancel**.

10 Observe the `Properties` of the selected `gasolineOneStageMerged` design under the tree, listing 2 constraints, and 238 points.

11 To experiment with a new design and avoid editing the prior designs, select the root `Designs` node and select **File > New Design**.

12 Add the constraints by selecting **Edit > Constraints**. In the Constraints Manager dialog box, click **Import**. Select the torque max and min constraints from the merged design and click **OK**. In the following dialogs, click **OK** to return to the Design Editor.

13 See how to construct a similar constrained space-filling design by selecting **Design > Space Filling > Design Browser**, or click the space-filling design toolbar button.

14 In the Space-Filling Design Browser, observe the design type is `Sobol Sequence`, and specify a **Number of points**. View the preview of design points. Click **OK**.
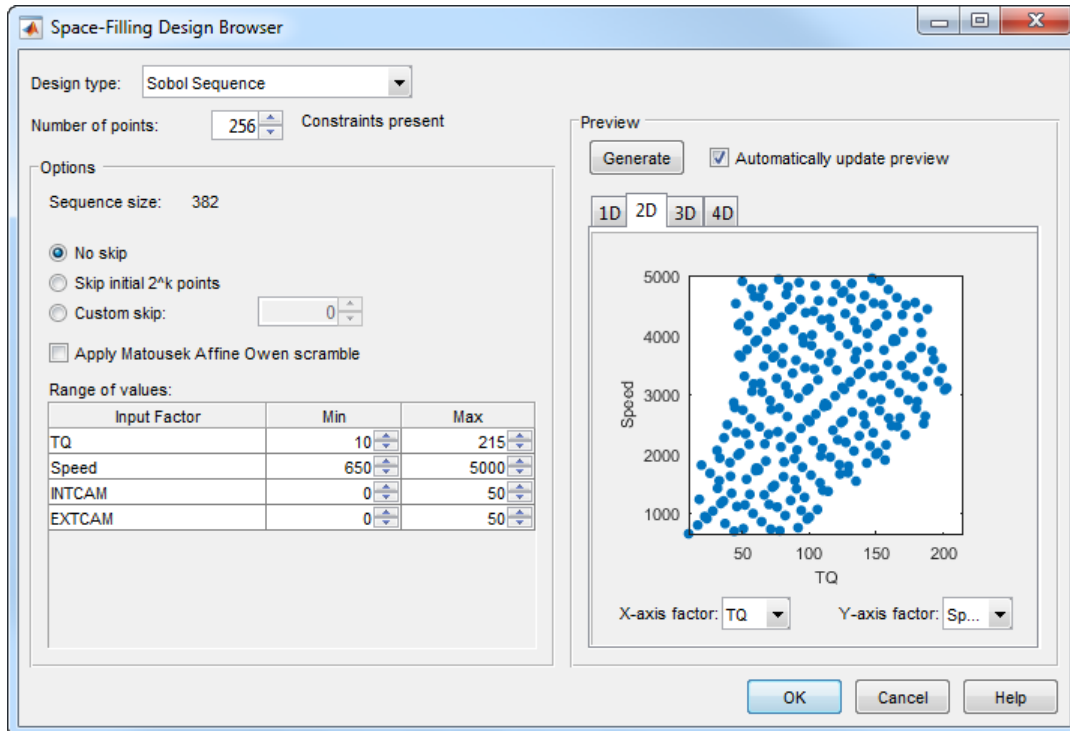
15 Click `gasolineOneStageMerged`. This design style is `Custom` because the points are rounded, using **Edit > Round Factor**. You might also sort design points to make data collection easier on a dyno. To preserve the space-filling sequence in case you want to add more points later, you can copy a design before rounding or sorting.

`gasolineOneStageMerged` is the final design used to collect data. To make it easier to collect the data, the points are rounded as follows:

- Intake and exhaust cam timings are rounded to 1 degree (2% of range)
- Speed is rounded to 50 RPM (1% of range)
- Torque is rounded to 5 Nm (3% of range)

You can export designs or copy and paste design points into other files to proceed to data collection.

16 Close the Design Editor.

The final `gasolineOneStageMerged` design was used to collect data from the GT-Power model with the Simulink and Simscape™ test harness. The example Model Browser project file `gasolineOneStage.mat` in the `mbctraining` folder contains this data, imported to the Model Browser after the data collection.

Data Collection and Physical Modeling

The toolbox provides the data in the projects for you to explore this calibration example.

MathWorks collected the data using simulation tools. Control and simulation models were constructed using a Simulink and Stateflow® test harness. Constrained experimental designs were constructed using Model-Based Calibration Toolbox. The points specified in the design were measured using the GT-Power engine simulation tool from Gamma Technologies (see http://www.gtisoft.com). The engine to calibrate is a direct-injected 1.5L spark ignition turbocharged engine with dual cam-phasers and turbocharger wastegate. This model is part of a GT-POWER engine library from Gamma Technologies.

To collect the data, Simulink and Stateflow controlled the GT-Power engine model to the desired Design of Experiments points.

Note Simulation time was reduced from days to minutes using Parallel Computing Toolbox™.

This simulation of 238 design points took 20 minutes to run in parallel on multiple machines in the cloud. Running on a single core, the same simulation takes 3 days. Parallel Computing Toolbox distributed the work to the 225 cores on a cloud computing cluster and showed that this problem scales linearly as you add workers.

The data was used in the next step of model-based-calibration, to create statistical models.
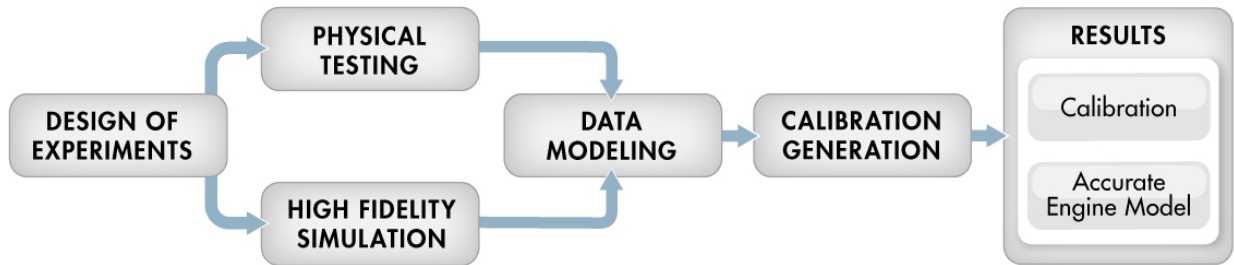
For next steps, see "Empirical Engine Modeling" on page 3-15.

Empirical Engine Modeling

In this section...
"Examine Response Models" on page 3-15
"Examine the Test Plan" on page 3-19

After designing the experiments and collecting the data, you can fit statistical models to the data. Use the toolbox to generate accurate, fast-running models from the measured engine data.

The dynamometer test setup is in speed/torque and so the design is in speed/torque. The model is in speed/load because the production engine controller implementation uses load (derived from air flow) instead of torque tables. The controller uses load because airflow sensors are presently less expensive in mass production than torque meters.

Examine Response Models

1   Open MATLAB. On the **Apps** tab, in the **Automotive** group, click **MBC Model Fitting**.

    In the Model Browser home page, in the **Case Studies** list, open **Dual CAM gasoline engine with spark optimized during testing**.

2   In the `gasolineOneStage.mat` project, click the second test plan node in the **All Models** tree, `gasolineOneStageModels`.

3   To assess high-level model trends, at the test plan node select the **Response Models** tab. After you fit models, the view at the test plan node displays the **Response Models** tab by default. View the cross-section plots of all the response models.
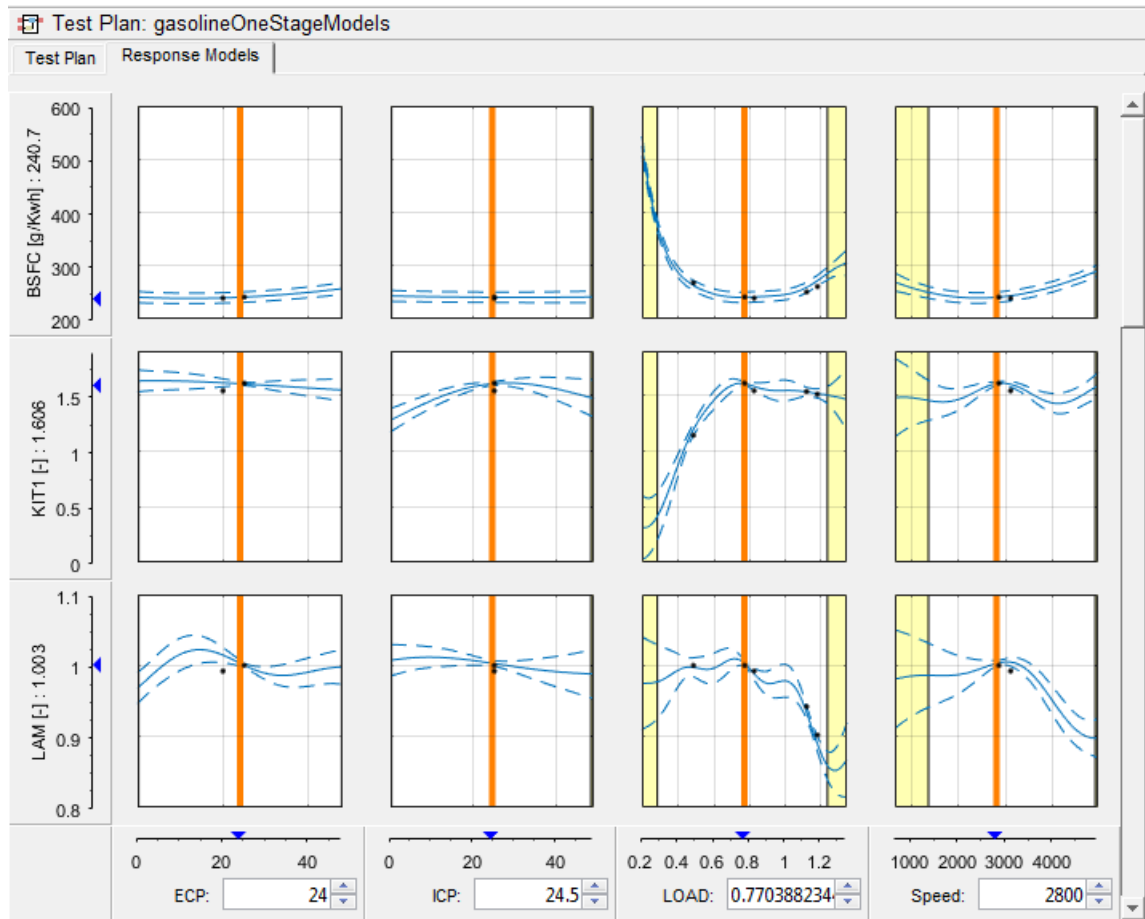
4 To view each response model in detail, expand `gasolineOneStageModels` test plan node in the **All Models** tree. Select the first response node under the test plan node, BSFC.
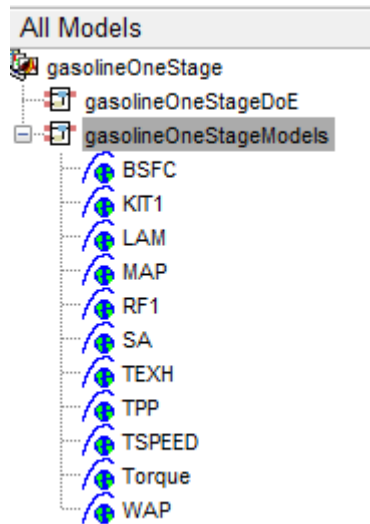
5 Examine the **Response Surface** plot and the **Diagnostic Statistics** plot.
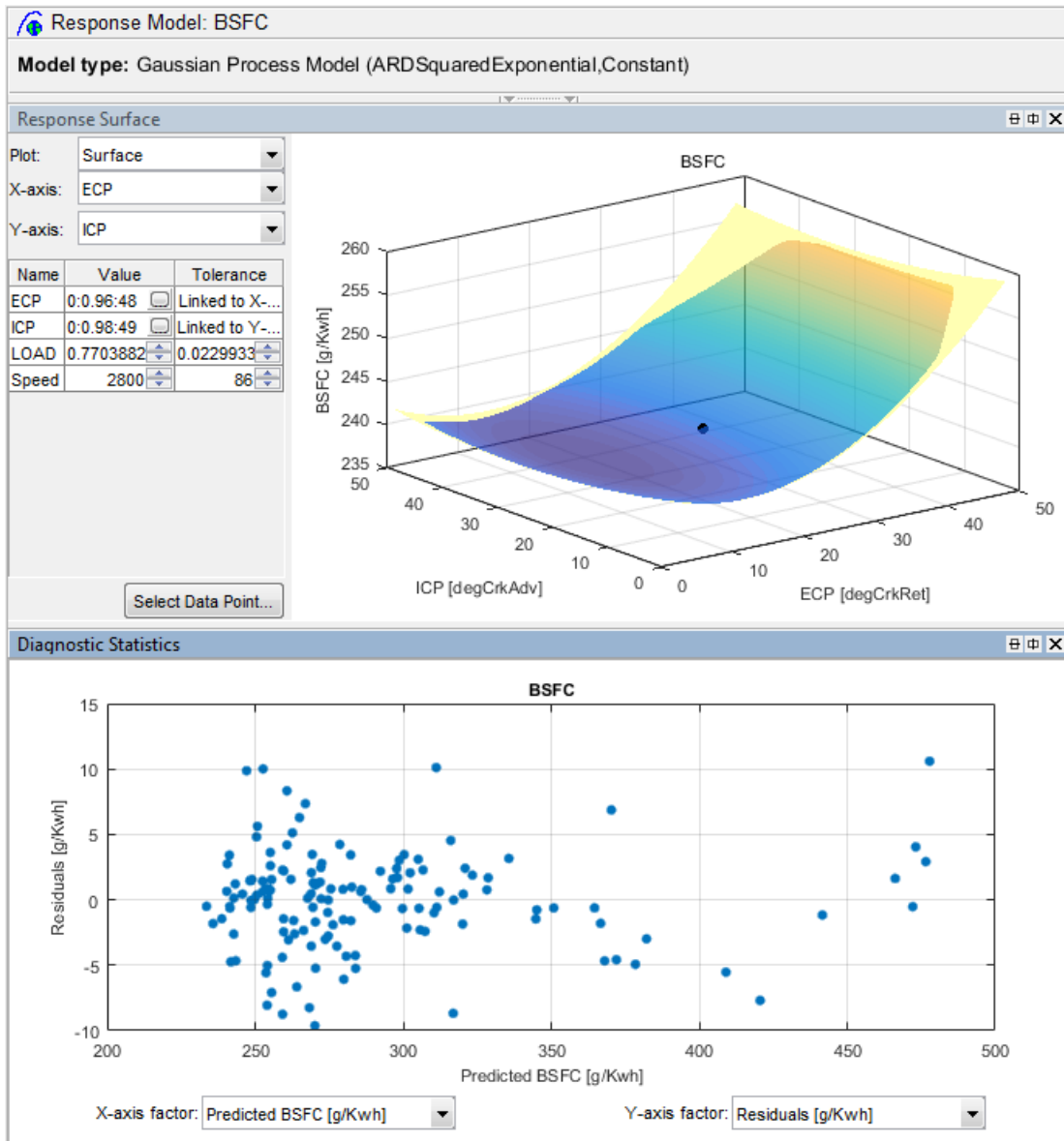
6 Examine the other responses in the All Models tree.

For details on using the plots and statistics to analyze models, see "Assess High-Level Model Trends" and "Assess One-Stage Models".

Examine the Test Plan

Examine the model setup.

1   At the `gasolineOneStageModels` test plan node, change to the test plan view if necessary by clicking the **Test Plan** tab. The Model Browser remembers selected views.

2   Observe the inputs and response model outputs listed on the test plan diagram.

3   Double-click the **Inputs** block to view the ranges and names (symbols) for variables on the Input Factor Set Up dialog box.

4   Double-click the **Model** block to view that the model class is the default for one-stage models, a `Gaussian Process Model`. When you use the **Fit models** button in the **Common Tasks** pane, and select a `One-stage` template, the toolbox sets the model type to a Gaussian Process Model.

   For details on setting up one-stage models, see "Fit a One-Stage Model".

5   Click **Cancel** to close the Model Setup dialog box without altering your example models.

For next steps, see "Optimization" on page 3-20.

Optimization

In this section...
"Optimization Overview" on page 3-20
"View Optimization Results" on page 3-21
"Set Up Optimization" on page 3-23
"Filling Tables From Optimization Results" on page 3-27

Optimization Overview

After creating statistical models to fit the data, you can use them in optimizations. You can use the accurate statistical engine model to replace the high-fidelity simulation and run much faster, enabling optimization to generate calibrations in feasible times. You use the statistical models to find the optimal configurations of the engine that meet the constraints.

The statistical models described in "Empirical Engine Modeling" on page 3-15 were used in the next step of model-based-calibration, to create optimized calibration tables.
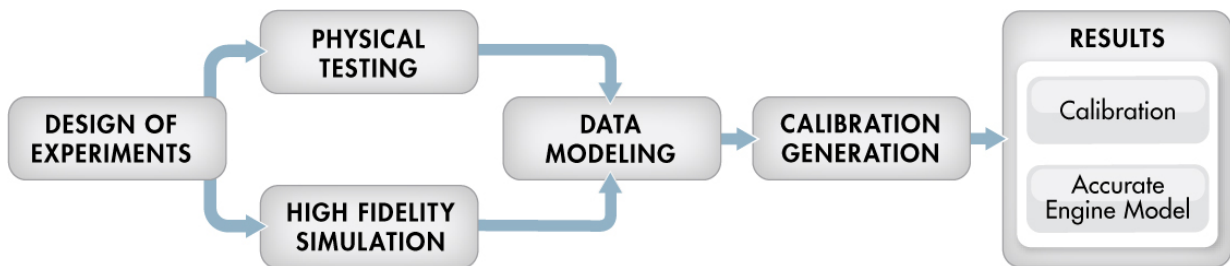
The aim of the calibration is to maximize torque at specific speed/load values across the engine's operating range, and meet these constraints:

- Limit knock (KIT1)
- Limit residual fraction
- Limit exhaust temperature
- Limit calibration table gradients for smoothness.

Turbocharger speed constraints are implicitly included in the boundary model which models the envelope.

The analysis must produce optimal engine calibration tables in speed and load for:

- Spark advance ignition timing (SA)
- Throttle position % (TPP)
- Turbo wastegate area % (WAP)
- Air/Fuel Ratio (Lambda: LAM)
- Intake cam phase (ICP)
- Exhaust cam phase (ECP)
- Torque (TQ)
- Brake-specific fuel consumption (BSFC)
- Boost (MAP)
- Exhaust temperature (TEXH)

View Optimization Results

1   Open MATLAB. On the **Apps** tab, in the **Automotive** group, click **MBC Optimization**.

2   In the CAGE Browser home page, in the **Case Studies** list, open **Dual CAM gasoline engine with spark optimized during testing**. Alternatively, select **File > Open Project** and browse to the example file gasolineOneStage.cag, found in matlab\toolbox\mbc\mbctraining.

3   In the **Processes** pane, click **Optimization**. Observe two optimizations, Torque_Optimization and Sum_ Torque_Optimization.

    Why are there two optimizations? To complete the calibration, you need to start with a point optimization to determine optimal values for the cam timings. You use the results of this point optimization to set up a sum optimization to optimize over the drive cycle and meet gradient constraints to keep the tables smooth.

4   In the Optimization pane, ensure Torque_Optimization is selected. In the Objectives pane, observe the description: maximize Torque as a function of ECP, ICP, LOAD and Speed.

5   In the Constraints pane, observe the constraints: the boundary model of Torque, knock, residual fraction, speed and exhaust temperature. If you want to see how these are set up, double-click a constraint and view settings in the Edit Constraint dialog box.

6 In the Optimization Point Set pane, observe the variable values defining the points at which to run the optimization.

7 To view the optimization results, in the Optimization pane expand the `Torque_Optimization` node and select `Torque_Optimization_Output`.
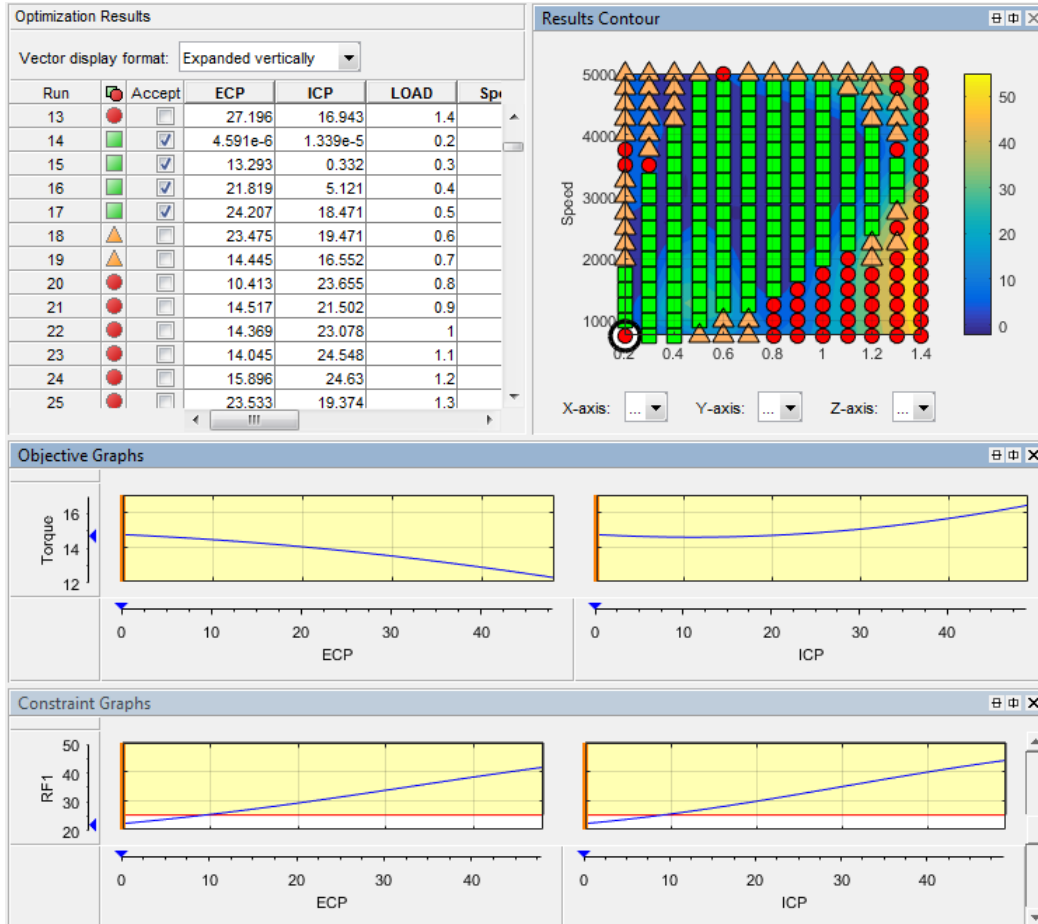
To learn about analyzing optimization results, see "Choosing Acceptable Solutions".

8 In the Optimization pane, select `Sum_Torque_Optimization` and compare with the previous point optimization setup. The sum optimization has additional gradient constraints to produce smooth tables, and a single run.

Set Up Optimization

Learn how to set up this optimization.

1   To perform an optimization, you need to import the statistical models created in the Model Browser. To see how to import models, in CAGE, select **File > Import From Project**.

The CAGE Import Tool opens. Here, you can select models to import from a model browser project file or direct from the Model Browser if it is open. However, the toolbox provides an example file with the models already imported, so click **Close** to close the CAGE Import Tool.

2   To view the models, click **Models** in the left **Data Objects** pane.

3   You need tables to fill with the results of your optimizations. To see all the tables, select the **Tables** view.

4   To see how to set up these tables, select **Tools > Create Tables from Model**. The Create Tables from Model wizard appears.

    a   Select the model `Torque` and click **Next**.

    b   On the next screen you see the controls for specifying the inputs and size of tables. Click the **Edit breakpoints** buttons to see how to set up row and column values. Click **Next**.
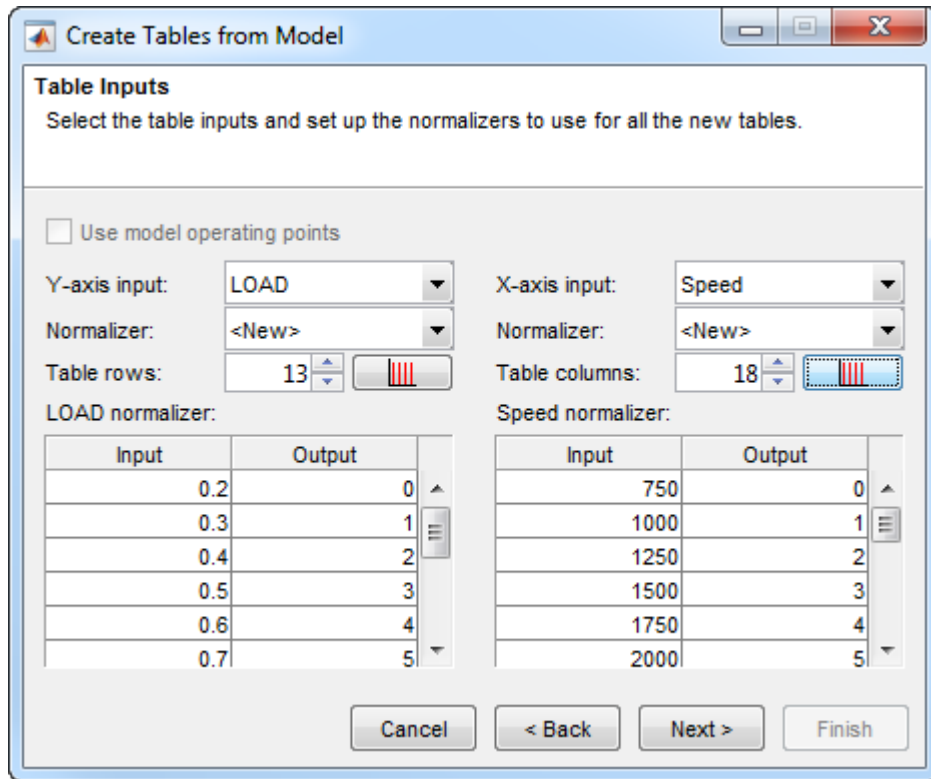
c   On the next screen you can set limits on table values. To edit, double-click **Table Bounds** values.

In this example, bounds on SA, ICP and ECP are set to [0,50], TPP and WAP are [0,100], and LAM is [0.8,1].
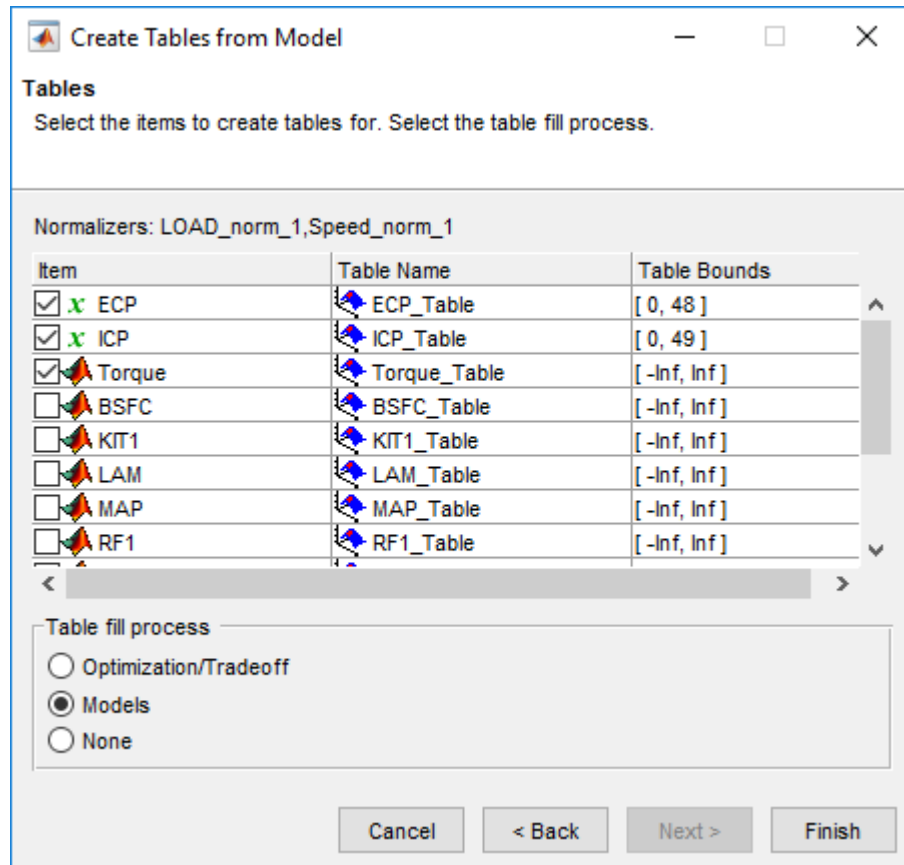
Click **Cancel** to avoid creating new unnecessary tables in your example file.

5   Learn the easiest way to set up an optimization by selecting **Tools > Create Optimization from Model**.

    a   Select the model `Torque` and click **Next**.

    b   Observe that the default settings will create a point optimization to minimize `Torque`, using 4 free variables, and constrained by a model boundary constraint. To create this example optimization, edit the settings to maximize `Torque`, use the BSFC table grid as a data source, and use only ECP and ICP as free variables. Click **OK** to create the optimization.
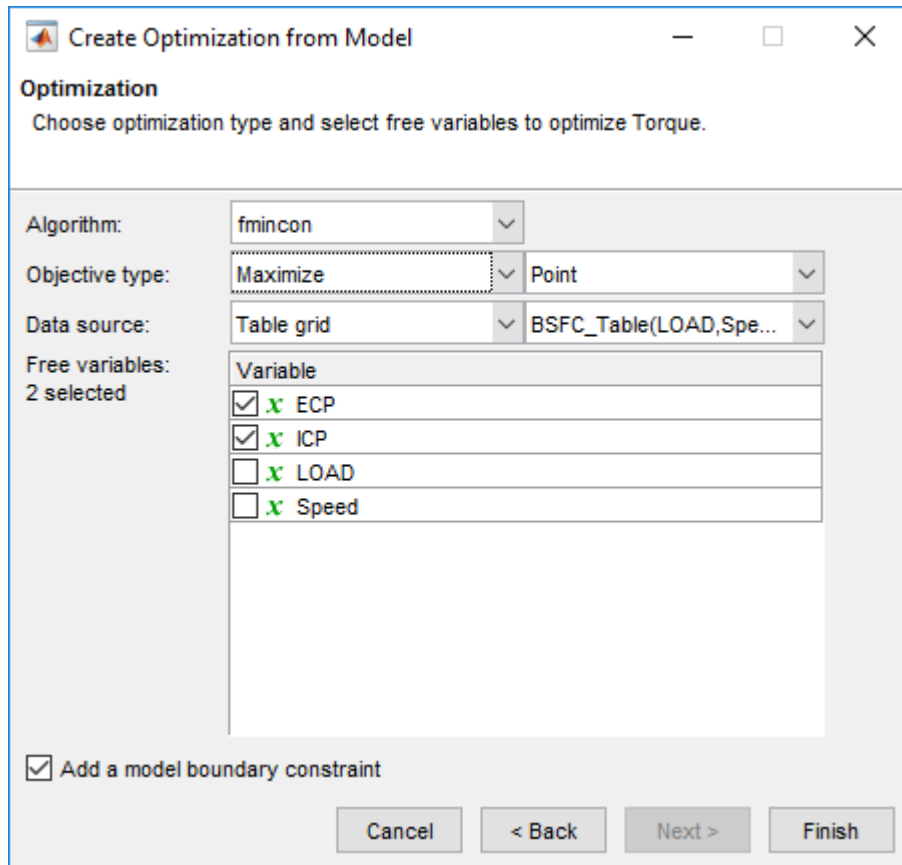
6. Compare your new optimization with the example `Torque_Optimization`. To finish the setup you need to add or import the rest of the constraints. In this case, select **Optimization > Constraints > Import Constraints**. Import the constraints from the `Torque_Optimization` optimization.

7. To learn how to set up the sum optimization, from the `Torque_Optimization_Output` node, select **Solution > Create Sum Optimization**. The toolbox creates a sum optimization for you.

8. Compare your new optimization with the example sum optimization, `Sum_Torque_Optimization`. The example shows you need to add the table smoothness constraints to complete the calibration. To see how to set up table gradient constraints, double-click the constraint `GradECP`.

To learn more about setting up optimizations and constraints, see:

- "Creating Tables from a Model"
- "Creating Optimizations from Models"
- "Edit Objectives and Constraints"
- "Create Sum Optimization from Point Optimization Output"

.

Filling Tables From Optimization Results

CAGE remembers table filling settings. To view how the example tables are filled:

1 Expand the example sum optimization node Sum_Torque_Optimization and select the Sum_Torque_Optimization_Output node.

2 Select **Solution** > **Fill Tables** (or use the toolbar button) to open the Table Filling from Optimization Results Wizard.

3 On the first screen, observe all the tables in the **CAGE tables to be filled list**. Note that the LAM (lambda) table is not filled from the optimization results, because this table was filled from the test data. Click **Next**.

4 On the second screen, observe all the tables are matched up with optimization results to fill them with. Click **Next**.

5 On the third screen, observe the table filling options. You can either click **Finish** to fill all the tables, or **Cancel** to leave the tables untouched. The example tables are already filled with these settings.

The following plot shows the calibration results for the ICP table. Observe that the idle region has locked cells to keep the cams parked at 0. If you want to lock values like this, to get smooth filled tables, lock the cells before filling from optimization results.
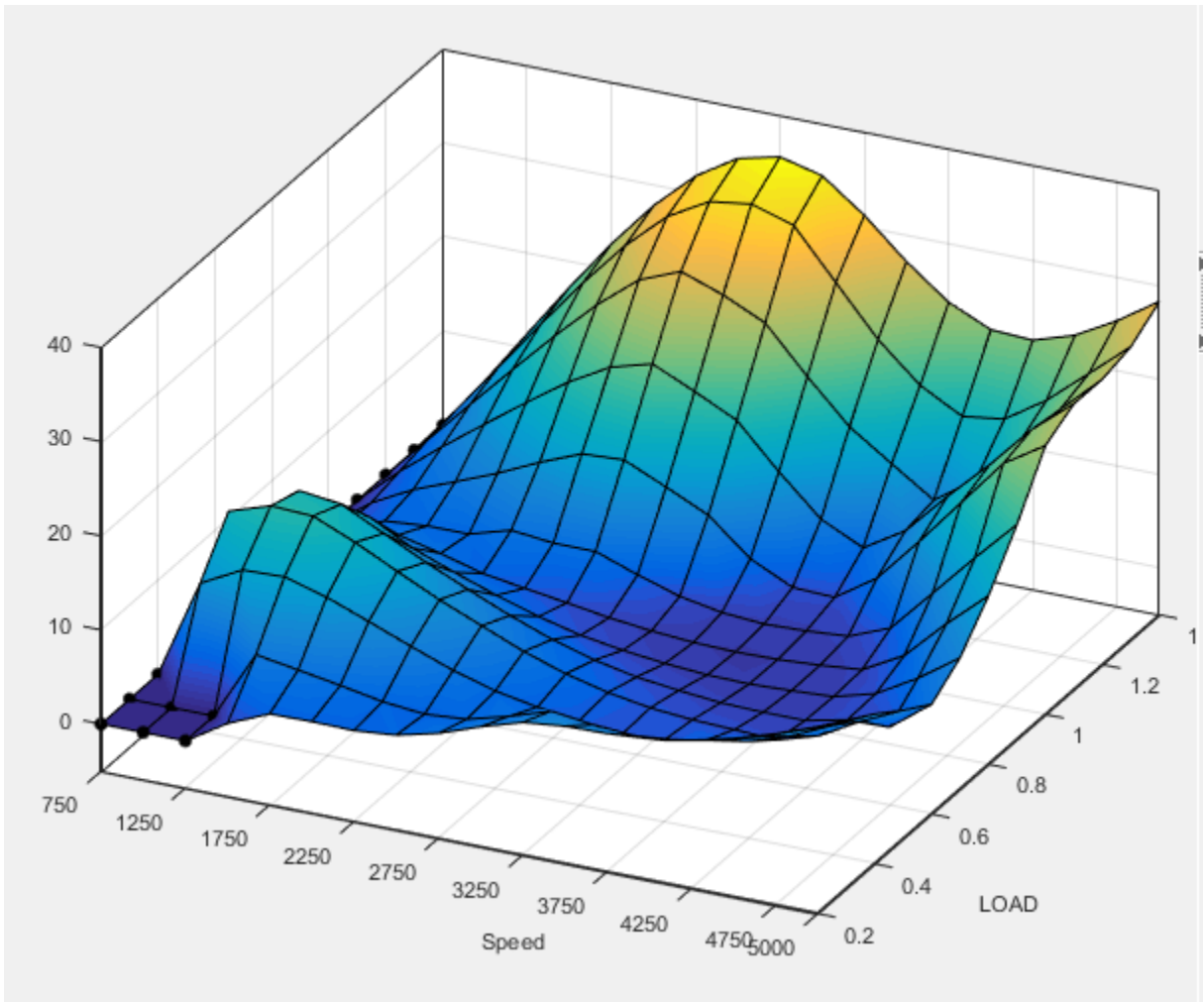
The following plot shows the calibration results for the ECP table.
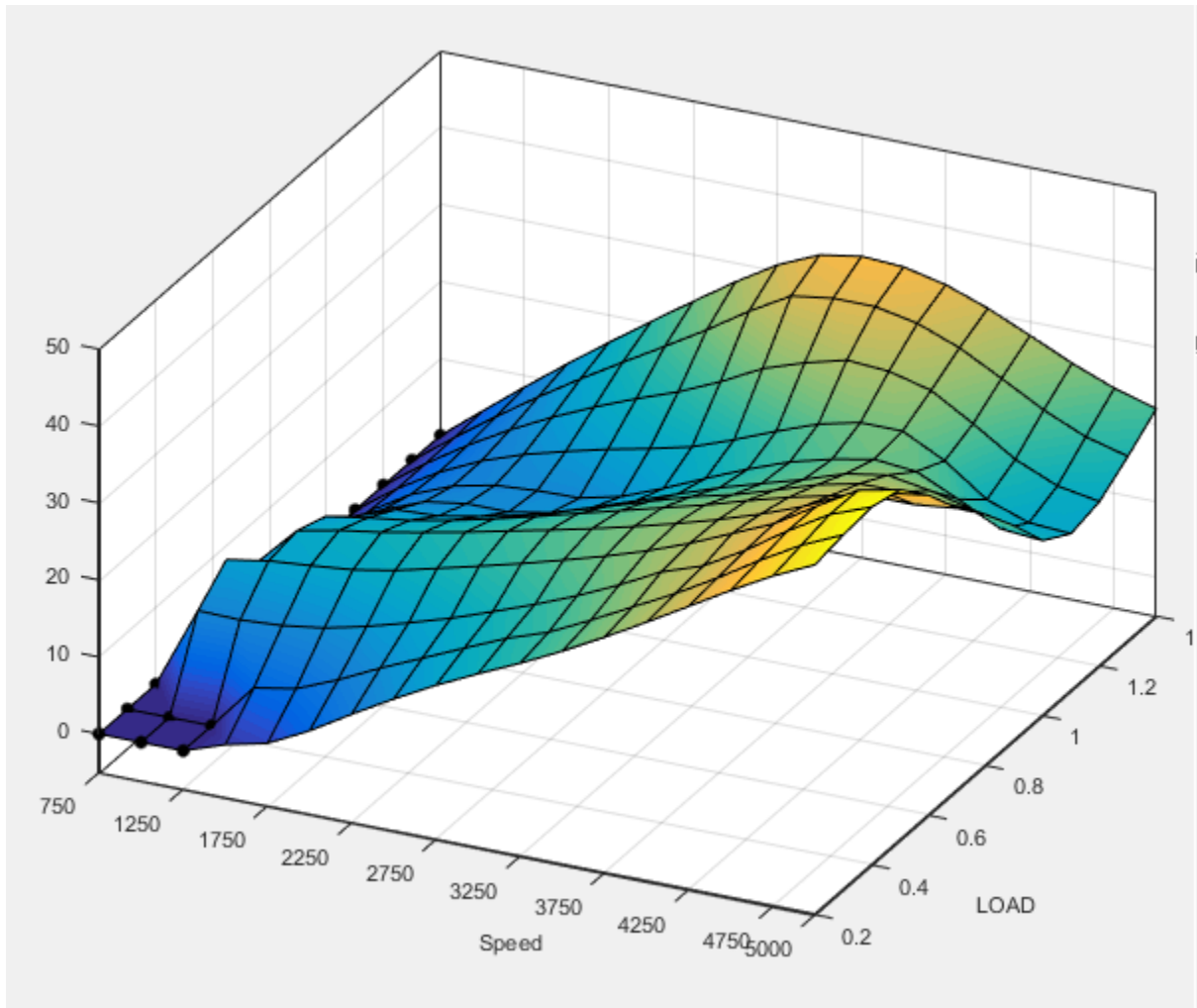
You can examine all the filled tables in the example project.

To learn more about analyzing and using optimization results, see "Optimization Analysis".

See Also

Related Examples

- "Creating Tables from a Model"
- "Creating Optimizations from Models"
- "Edit Objectives and Constraints"
- "Choosing Acceptable Solutions"
- "Create Sum Optimization from Point Optimization Output"
- "Filling Tables from Optimization Results"

Composite Models and Modal Optimizations

Composite Models and Modal Optimization

You can use composite models in CAGE to produce optimal calibrations for engines with multiple operating modes.

Gasoline Example with Four Cylinder and Eight Cylinder Modes

You can load these example projects found in `matlab\toolbox\mbc\mbctraining`, to view completed examples of composite models, optimizations and filled tables:

- `GasolineComposite.mat`
- `GasolineComposite.cag`

The `GasolineComposite.cag` example shows optimizations created from composite models imported from `GasolineComposite.mat`. The composite models combine responses for two engine operating modes: 4–cylinder mode and 8–cylinder mode. In this example, the modal optimization is a point optimization that selects the best mode for maximizing torque at each operating point.

In this example, the sum optimization has been created from the results of the point optimizations, using the selected best mode at each point. Table gradient constraints are added for ICP and ECP to ensure smooth control and engine response.

See Also

More About

- "Creating and Viewing Composite Models in CAGE"
- "Set Up Modal Optimizations"
- "Analyzing Modal Optimization Results"

5

Design and Modeling Scripts

Introduction to the Command-Line Interface

The Model-Based Calibration Toolbox product is a software tool for modelling and calibrating powertrain systems. The command-line interface to the Model-Based Calibration Toolbox product enables the design of experiments and modeling tools available in the toolbox to be accessible from the test bed.

You can use these commands to assemble your specific engine calibration processes into an easy to use script or graphical interface. Calibration technicians and engineers can use the custom interface without the need for extensive training. This system enables:

- Transfer of knowledge from the research and development engineers into the production environment
- Faster calibration
- Improved calibration quality
- Improved system understanding
- Reduced development time

See Model-Based Calibration Toolbox Examples for command-line examples.

Automate Design and Modeling With Scripts

In this section...
"Processes You Can Automate" on page 5-3
"Engine Modeling Scripts" on page 5-5

Processes You Can Automate

You can use these command-line functions to automate engine modeling processes.

Goal		Function
Create or load a project		• CreateProject • Load
Create a new test plan for the project using a template		CreateTestplan
Create designs that define data points to collect on the test bed		CreateDesign
Work with classical, space-filling or optimal designs		• CreateConstraint • CreateCandidateSet • Generate • FixPoints • Augment
Create or load a data object for the project and make it editable		• CreateData • BeginEdit
Load data from a file or the workspace		• ImportFromFile • ImportFromMBCDataStructure
Work with data	Examine	Value
	Modify	• AddFilter • AddTestFilter
	Add variables	AddVariable
	Add	Append

Goal		Function
	Group	• `DefineTestGroups` • `DefineNumberOfRecordsPerTest`
	Export	`ExportToMBCDataStructure`
Save your changes to the data, or discard them		• `CommitEdit` • `RollbackEdit`
Designate which project data object to use for modeling in your test plan		`AttachData`
Create and evaluate boundary models, either in a project or standalone		"Boundary Model Scripting" on page 5-9
Create models for the data; these can be one- or two-stage models and can include datum models		`CreateResponse`
Work with your models	Examine input data and response data	• `DoubleInputData` • `DoubleResponseData`
	Examine predicted values at specified inputs	• `PredictedValue` • `PredictedValueForTest`
	Examine Predicted Error Variance (PEV) at specified inputs	• `PEV` • `PEVForTest`
	Examine and remove outliers	• `OutlierIndices` • `OutlierIndicesForTest` • `RemoveOutliers` • `RemoveOutliersForTest` • `RestoreData`

Goal		Function
	Create a selection of alternative models	`CreateAlternativeModels`
	Choose the best model by using the diagnostic statistics	• `AlternativeModelStatistics` • `DiagnosticStatistics` • `SummaryStatistics`
	Extract a model object from any response object	• `Model Object` • `Fit` • `CreateModel` • `ModelSetup` • `Type (for models)` • `Properties (for models)` • `CreateAlgorithm` • `StepwiseRegression` • `Jacobian` • `ParameterStatistics` • `UpdateResponse`
For two-stage test plans, once you are satisfied with the fit of the local and response feature models, calculate the two-stage model		`MakeHierarchicalResponse`
Export models to MATLAB or Simulink		`Export`

Engine Modeling Scripts

For command-line script examples, see Model-Based Calibration Toolbox Examples. Run the scripts to learn about:

• Loading and Modifying Data

• Designing experiments and constraining designs

- Gasoline engine modeling script to automatically generate a project for the gasoline case study, including:

  - Grouping and filtering data
  - Boundary modeling
  - Response modeling
  - Removing outliers and copying outlier selections
  - Creating alternative models and selecting the best based on statistical results

- Point-by-point diesel engine modeling to automatically generate a project for the diesel case study, including:

  - Defining engine operating points
  - Creating designs for each operating point
  - Augmenting designs to collect more data
  - Building a point-by-point boundary model
  - Create response models using the local multiple model type

See Also

More About

- "Automation Scripting"

Understanding Model Structure for Scripting

In this section...
"Projects and Test Plans for Model Scripting" on page 5-7
"Response Model Scripting" on page 5-7
"Boundary Model Scripting" on page 5-9

Projects and Test Plans for Model Scripting

To use the Model Browser in the Model-Based Calibration Toolbox product, you must understand the structure and functions of the model tree to navigate the views. To use the command-line version of the toolbox, you must understand the same structure and functions, that is, how projects, test plans, and models fit together. The following sections describe the relationship between the different models that you can construct. The diagrams in the following section, "How the Model Tree Relates to Command-Line Objects" on page 5-11, illustrate these relationships.

- Projects can have one or more test plans.
- Projects can have one or more data objects.

- Test plans have no more than one data object.
- Test plans have response objects.
    - If a one-stage test plan, these are simply known as responses.
    - If two-stage test plan, these are hierarchical responses.
- Test plans have boundary tree objects.

Response Model Scripting

A response is a model fitted to some data. These are the types of responses:

- Hierarchical Response (Level 0)

    A hierarchical response (also known as a two-stage response) models a `ResponseSignalName` using a local response and one or more response features.

    A hierarchical response has one or more different local responses (accessible via the property `LocalResponses`) that provide different possible models of the

`ResponseSignalName`. One of these must be chosen as the best, and that will then be the local response used subsequently. The response features of each of the local responses are available directly from those local response objects.

- Local Response (Level 1)

  The local response consists of models of the `ResponseSignalName` as a function of the local input factors. The local input factors are accessible via the `InputSignalNames` property.

  A local response has one or more response features (accessible via the property `ResponseFeatures`) containing the models fitted to those response features of the local model.

- Response (Level 1 or 2)

  - For two-stage test plans, response objects model the response features of local responses (`ResponseSignalName` corresponds to the name of the response feature). In this case, the response has a level value of 2.

  - For one-stage test plans, response objects simply model the `ResponseSignalName` as a function of the input factors. In this case, the response will have a level value of 1.

  All responses can have zero or more alternative responses (accessible via the property `AlternativeResponses`) that provide different possible models of the `ResponseSignalName`. These all retain the same level as the response for which they are an alternative. One of these must be chosen as the best and that will then be the response used subsequently.

See the illustrations in the following section, "How the Model Tree Relates to Command-Line Objects" on page 5-11, for examples of different responses and how they relate to each other.

Note that each response contains a model object (`mbcmodel.model`) that can be extracted and manipulated independently of the project. You can change the model type and settings, fit to new data, examine coefficients, regression matrices and predicted values, and use stepwise functions to include or remove terms. You can change model type, properties and fit algorithm settings. To learn about what you do with a model object, see `Model Object`. If you change the model, you must use `UpdateResponse` to replace the new model type in the response object in the project. When you use `UpdateResponse` the new model is fitted to the response data. See `UpdateResponse`.

Boundary Model Scripting

You can create and evaluate boundary models either in a project or standalone. You can combine boundary models in the same way as when using the Boundary Editor GUI. You can use boundary models as design constraints.

In a project, the test plan has a `Boundary` property that can contain an `mbcboundary.Tree` object.

```
BoundaryTree = mbcmodel.testplan.Boundary
```

The `BoundaryTree` is a container for all the boundary models you create. The tree is empty until you create boundaries, and if you change the testplan data the toolbox deletes the boundaries.

You can fit boundary models in `mbcmodel` projects using the boundary tree class `mbcboundary.Tree`, or you can fit boundary models directly to data.

To create a boundary model outside of a project, you can either:

- Use the `CreateBoundary` package function:

  ```
  B = mbcboundary.CreateBoundary(Type,Inputs)
  ```
- Use the `Fit` method to create and fit a boundary to some data X:

  ```
  B = mbcboundary.Fit(X,Type)
  ```

To create a boundary model within a project, use the `CreateBoundary` method of the boundary tree:

```
B = CreateBoundary(Tree,Type)
```

This creates a new boundary model, B, from the `mbcboundary.Tree` object, `Tree`. The test plan inputs are used to define the boundary model inputs. The new boundary model is not added to the tree, you must call `Add`.

To create a new boundary model from an existing boundary model, you can use the `CreateBoundary` method of all boundary model types:

```
B = CreateBoundary(B,Type)
```

You can combine boundary models by using the `InBest` property of the boundary tree. This corresponds to combining boundary models in best in the Boundary Editor GUI, as

described in "Combining Best Boundary Models" in the Model Browser documentation. You can also combine boundary models with logical operators, for use as design constraints or outside projects.

You can change the `ActiveInputs`, `Evaluate`, and use as a `designconstraint`.

How the Model Tree Relates to Command-Line Objects

The tree in the Model Browser displays the hierarchical structure of models. This structure must be understood to use the command-line interface. The following examples illustrate the relationship between projects, test plans and responses in one-stage and two-stage models.

The following is an example of a two-stage model tree.
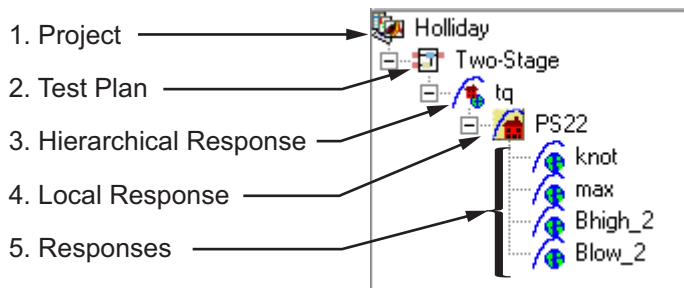
The elements of the tree correspond to the following objects in the command-line interface:

1 Project
2 Test Plan
3 Hierarchical Response
4 Local Response
5 Responses

The following example illustrates a project containing a one-stage test plan; in the command-line interface this corresponds to a project, one-stage test plan, and a response model.

Hierarchical responses can have multiple local responses, as shown in the following example from the Model Browser. In the command-line interface these are accessible via the property `LocalResponses` for a hierarchical response object (`mbcmodel.hierarchicalresponse`). In this example, the local responses are PS22, PS32, and POLY2.

Only one of these local responses can be chosen as best (in this example, PS22, indicated by the blue icon) and used to construct the hierarchical response, together with the associated response features of the local response. Each local response object has a set of responses, accessible by the property `ResponseFeatures(Local Response)`.
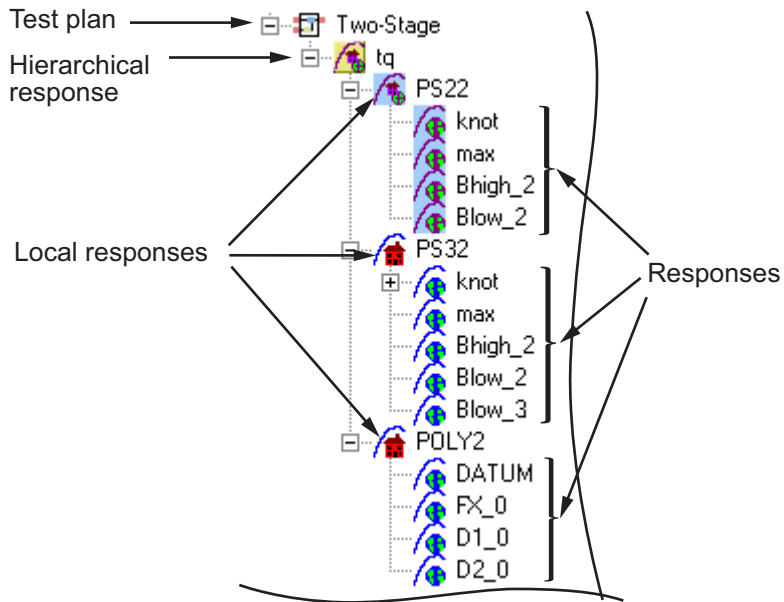
Responses can have zero or more alternative responses, as shown in the following model tree. You call the method `CreateAlternativeModels` on the command line to do the same.
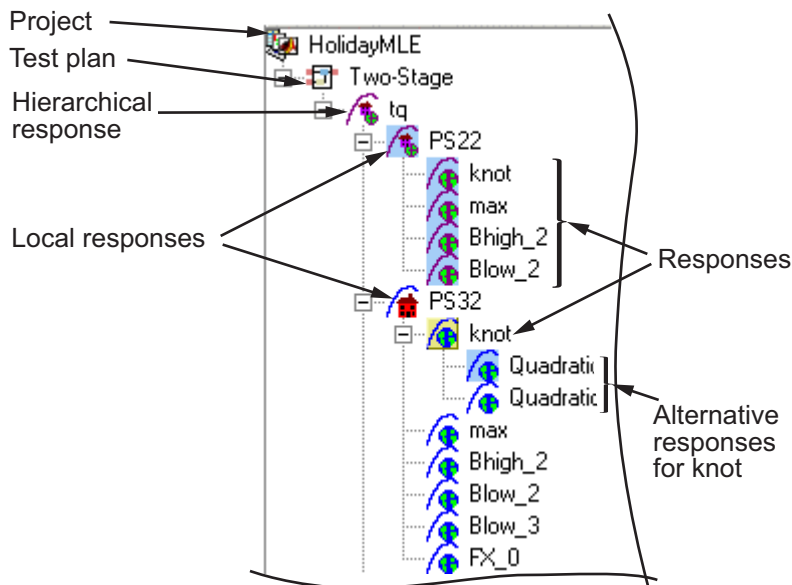
In this example, the alternative responses for the `knot` response are accessible via the property `AlternativeResponses`. You can create alternative responses for any response (including all one-stage responses).

You can use model templates to try alternative model types for several responses. The following example shows the results of using a model template for four alternative responses (`Linear-RBF`, `RBF-multiquadric`, `Cubic`, and `Quadratic`). The model template has been used to create alternative responses for the responses `knot` and `max`. You can call the method `CreateAlternativeModels` to do this in the command-line interface.
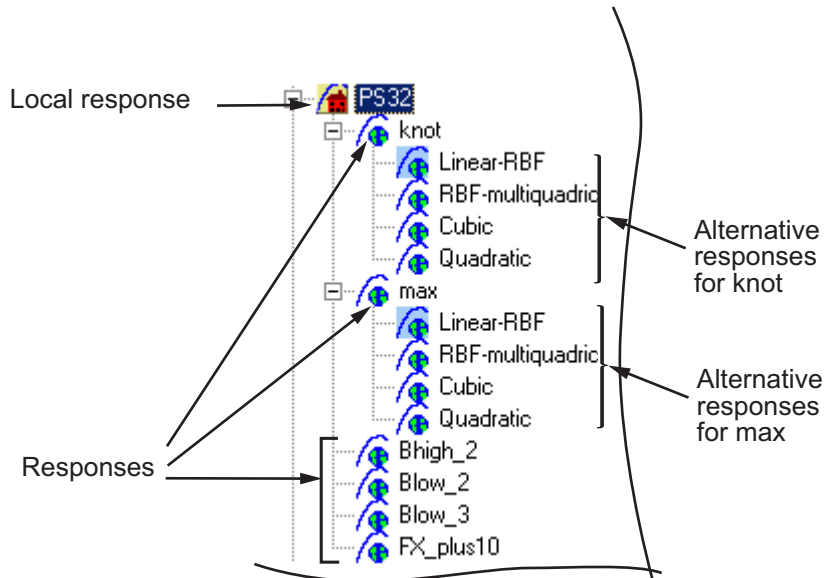
One of the alternative responses must be chosen as best for each response (call the method `ChooseAsBest`). In this example, when `Linear-RBF` is chosen as best from the alternatives for the `knot` response, then it is copied to `knot`.

6

Multi-Injection Diesel Calibration

Multi-Injection Diesel Calibration Workflow

In this section...
"Multi-Injection Diesel Problem Definition" on page 6-2
"Engine Calibration Workflow" on page 6-7
"Air-System Survey Testing" on page 6-8
"Multi-Injection Testing" on page 6-9
"Data Collection and Physical Modeling" on page 6-10
"Statistical Modeling" on page 6-11
"Optimization Using Statistical Models" on page 6-12
"Case Study Example Files" on page 6-20

Multi-Injection Diesel Problem Definition

This case study shows how to systematically develop a set of optimal steady-state engine calibration tables using Model-Based Calibration Toolbox.

The engine to be calibrated is a 3.1L multi-injection combustion ignition engine with common rail, variable-geometry turbocharger (VGT), and cooled exhaust gas recirculation (EGR).

The aim of the calibration is to minimize brake-specific fuel consumption (BSFC) at specific speed/load operating points across the engine's operating range, and meet these constraints:

- Limit total NOx emissions.
- Limit maximum turbocharger speed.
- Limit calibration table gradients for smoothness.

The analysis must produce optimal calibration tables in speed and torque for:

- Best main start of injection timing
- Best total injected fuel mass per cylinder per cycle
- Best pilot injection timing relative to main timing
- Best pilot injection fuel mass fraction of total injection mass

- Best exhaust gas recirculation (EGR) position
- Best variable geometry turbocharger (VGT) vane position
- Best fuel rail pressure relative to nominal pressure vs. engine speed

These sections explain the objectives of selecting best values for these calibration tables and the effects of these control variables on the engine:

- "Select Main Injection Timing for Efficiency" on page 6-3
- "Select Pilot Injection Timing to Control Noise" on page 6-4
- "Select Main Fuel Mass for Efficiency and Emissions" on page 6-5
- "Select Fuel Pressure for Efficiency and Emissions" on page 6-5
- "Select the Turbocharger Position to Control Air-Charge and EGR" on page 6-6
- "Select the EGR Valve Position to Control Air-Charge and Emissions" on page 6-7

Select Main Injection Timing for Efficiency

You select the injection timing of the main fuel injection to maximize engine efficiency. You aim to make peak cylinder pressure occur slightly after piston top center. You inject fuel just before top-center compression, as shown.
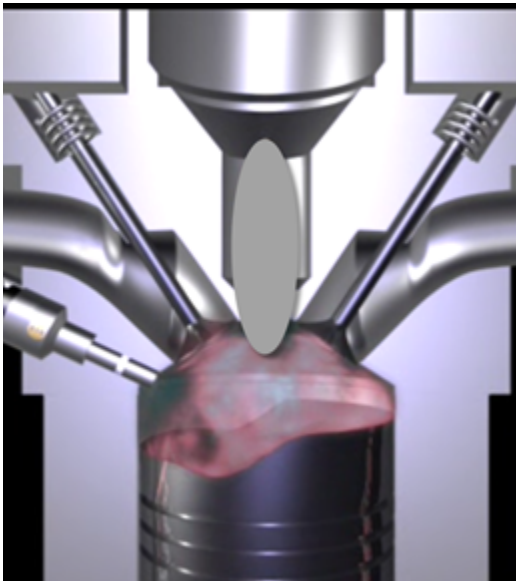
Then you can achieve peak combustion pressure just after top-center expansion.

You also need to adjust injection timing according to speed and other conditions.

- You need to advance (move earlier before piston top center) the start of injection timing with increasing speed and dilution (exhaust gas recirculation or EGR).

- You need to retard the start of injection timing with increased fresh air intake (load).

Select Pilot Injection Timing to Control Noise

You select the timing of the pilot fuel injection to start combustion early before the larger main fuel injection. The pilot fuel injection occurs well before top-center compression and before the main injection.

You can use pilot fuel injection to control combustion noise, because it affects the variability in cylinder pressure.

In this example, pilot fuel injection timing is defined as a crank-angle delta offset before the main injection, and is therefore a relative quantity.

Select Main Fuel Mass for Efficiency and Emissions

The air-fuel ratio (AFR) affects engine efficiency and emissions. A rich AFR causes high engine-out particulates and low engine-out NOx. You control AFR by changing the main fuel mass for optimal balance between power and emissions.

The AFR of the combustion mixture is determined by the main fuel injection mass for a given amount of fresh air. The amount of air results mainly from EGR valve position, VGT position, intake throttle position, and speed.

Select Fuel Pressure for Efficiency and Emissions

You can use fuel pressure to control fuel droplet size. Reduced fuel droplet size in the combustion chamber reduces particulates, releases more energy from the fuel, and achieves more stable combustion. High fuel pressure decreases fuel droplet size to improve efficiency and emissions.

At low loads, you can use lower fuel pressure to decrease fuel pump power losses without much effect on emissions and power efficiency.

In this example, fuel pressure is controlled relative to an engine-speed-dependent base level via a fuel pressure delta, and is therefore a relative quantity.

Select the Turbocharger Position to Control Air-Charge and EGR

You can use the variable-geometry turbocharger (VGT) position to balance fresh air and exhaust gas recirculation for optimal NOx control at a given power level.

You can change VGT vane position to increase cylinder fresh air due to the turbocharger speed increase. With the vanes closed, the turbocharger moves faster (high VGT speed) and sends a higher load (or boost) of air into the engine. Closing the vanes also increases exhaust gas recirculation (EGR) due to increased backpressure from the closed vanes.
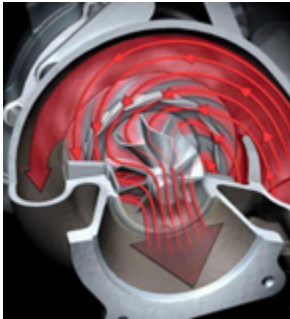
With the vanes open, VGT speed is low and passes through low load (or boost) to the engine.

Select the EGR Valve Position to Control Air-Charge and Emissions

You can use the EGR valve position to control the flow of burned exhaust gases back to the intake manifold.

Reburning exhaust gases decreases in-cylinder temperature, resulting in a significant decrease in NOx emissions.

If you select too much EGR for a given amount of injected fuel, then the air-fuel ratio will be rich, causing increased soot emissions. Therefore, you must balance these competing objectives.

In engines, timing is everything. Using the EGR valve and all the other control variables, controlling the engine's air flow is the key to optimizing fuel economy, reducing emissions, and increasing power density.

Engine Calibration Workflow

The following graphic illustrates the workflow for the model-based calibration process. The workflow can use a combination of tools: Model-Based Calibration Toolbox, Simulink, Stateflow, third-party high-fidelity simulation tools, and Hardware-in-the-Loop testing for fine tuning calibrations on ECUs.

MBC Toolbox
Design of Experiments

Simulink/Stateflow
Automated Virtual Engine Mapping

GTPOWER
High Fidelity Engine Model

MBC Toolbox
Model Fitting

MBC Toolbox
Calibration Generation
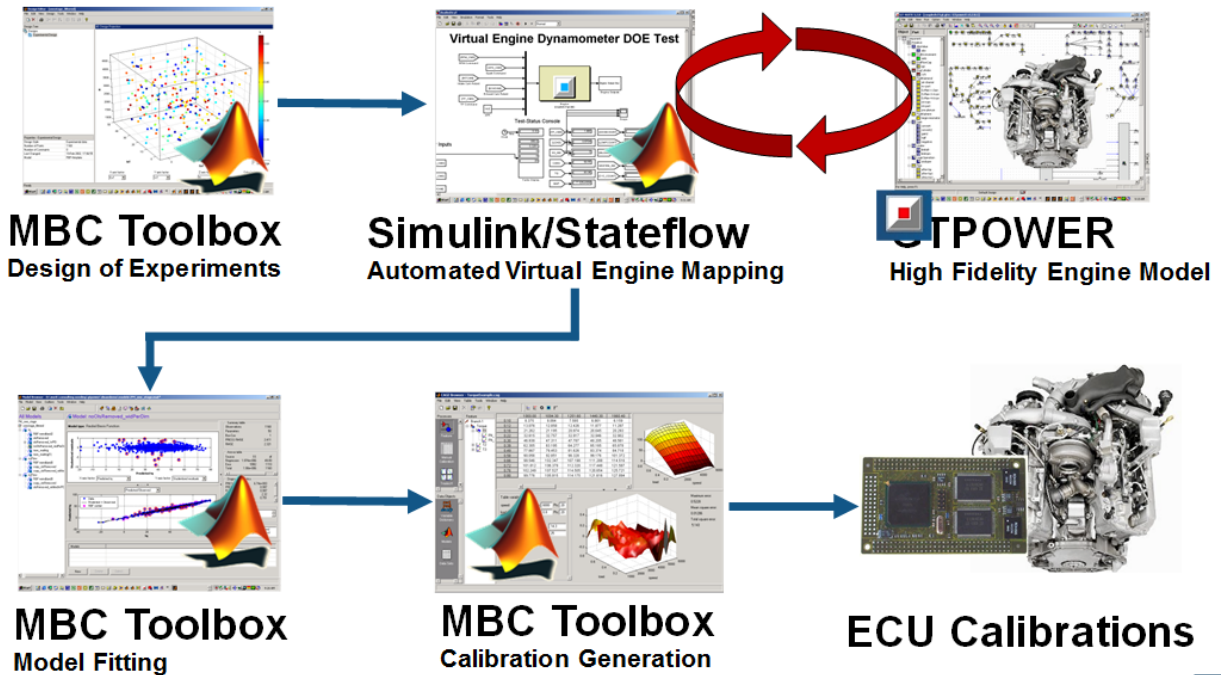
ECU Calibrations

Air-System Survey Testing

The first step to solve this calibration problem is to determine the boundaries of the feasible air-system settings. To do this, you create an experimental design and collect data to determine air-system setting boundaries that allow positive brake torque production in a feasible AFR range.
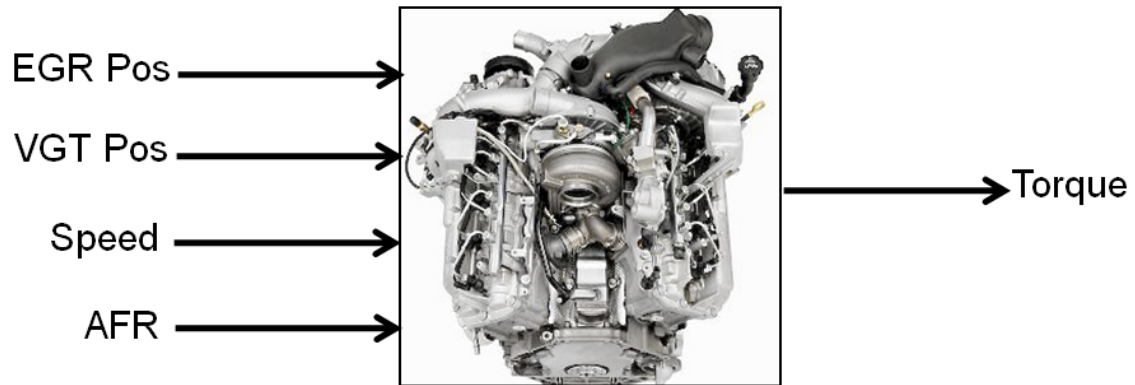
These simplifications were used to conduct the initial study:

- Pilot injection is inactive.
- Main timing is fixed.
- Nominal fuel pressure vs RPM.
- Main fuel mass is moved to match the AFR target.

Fit a boundary model to these design points.

Multi-Injection Testing

After the air-system survey, you have established the boundaries of positive brake torque air-system settings. Now, you can create an experimental design and collect data to gather fuel injection effects within those boundaries. You can then use this data to create response models for all the responses you need to create an optimal calibration for this engine.
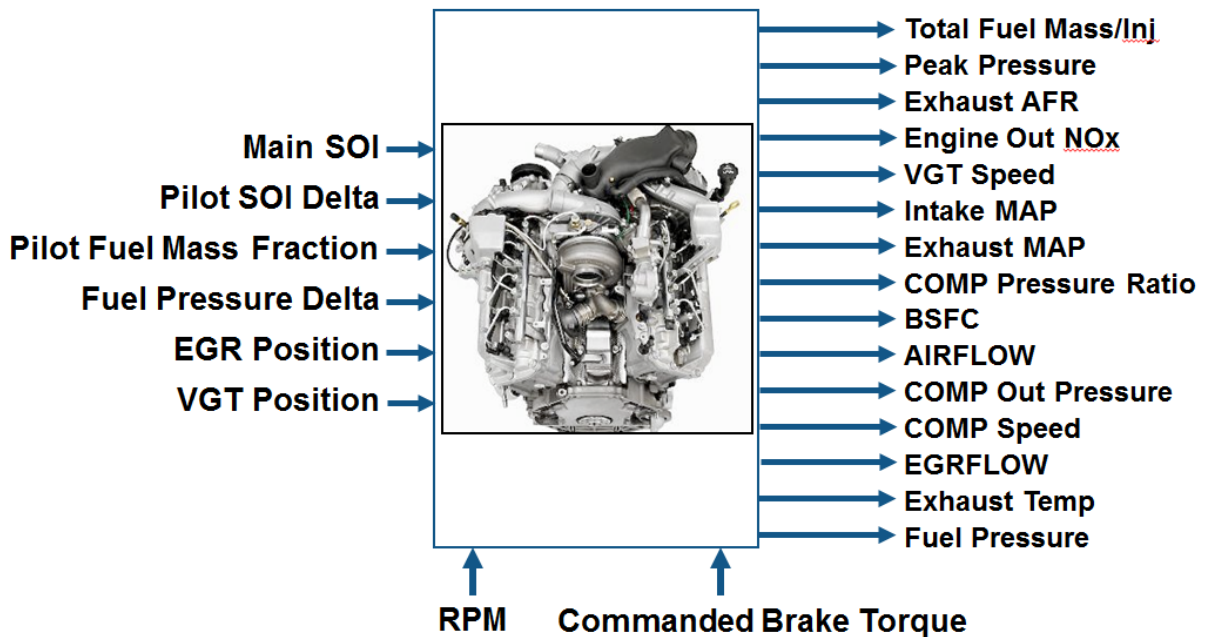
Data Collection and Physical Modeling

The toolbox provides the data for you to explore this calibration example.

MathWorks collected the data using simulation tools. Control and simulation models were constructed using Simulink and Stateflow. Constrained experimental designs were constructed using Model-Based Calibration Toolbox. The points specified in the design were measured using the GT-Power engine simulation tool from Gamma Technologies (see http://www.gtisoft.com).

To collect the data, Simulink and Stateflow controlled the torque output of the GT-Power engine model to the desired Design of Experiments points using total fuel mass. This graphic shows the virtual dynamometer test model.

Virtual Engine Dynamometer Diesel DoE Test Setup
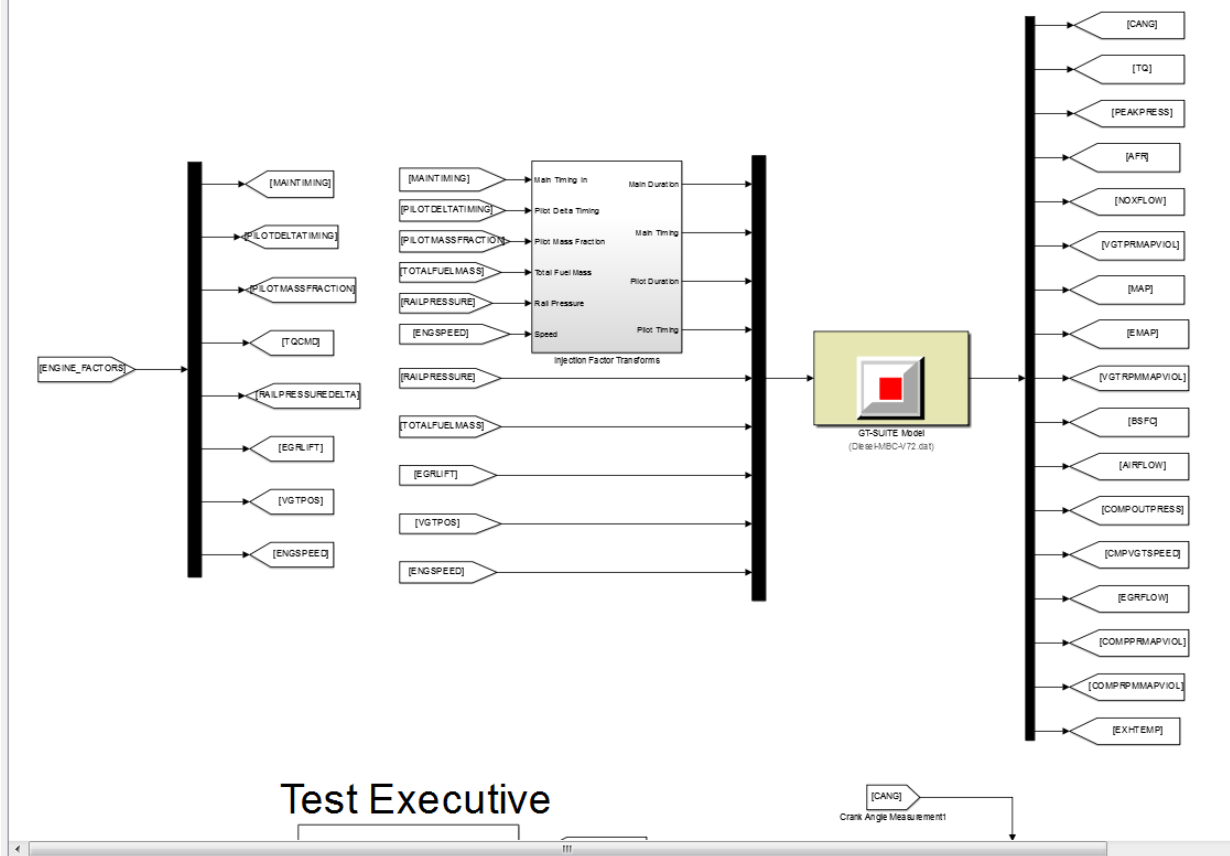
Statistical Modeling

After designing the experiments and collecting the data, you can fit statistical models to the data. You can use the toolbox to generate accurate, fast-running models from the measured engine data.

The following graphic shows the models to define in the toolbox to solve this calibration problem. The graphic shows how the model inputs and output relate to the optimal tables, optimization operating points, objectives and constraints you need to perform the optimization and create the calibration.

I/O of Multi-Inject 3.1L Common Rail Engine Model with Variable Geometry Turbocharger and Cooled EGR

Goal: Minimize BSFC subject to NOx, turbocharger speed, and user-specified table gradient constraints

Optimization Using Statistical Models

After creating statistical models to fit the data, you can use them in optimizations. You can use the accurate statistical engine model to replace the high-fidelity simulation and run much faster, enabling optimizations to generate calibrations.

1 Run an optimization to choose whether to use Pilot Injection at each operating point.

2 Optimize fuel consumption over the drive cycle, while meeting these constraints:

- Constrain total NOx
- Constrain turbocharger speed
- Constrain smoothness of tables

3 Fill lookup tables for all control inputs.

The following plots show a preview of the calibration results.
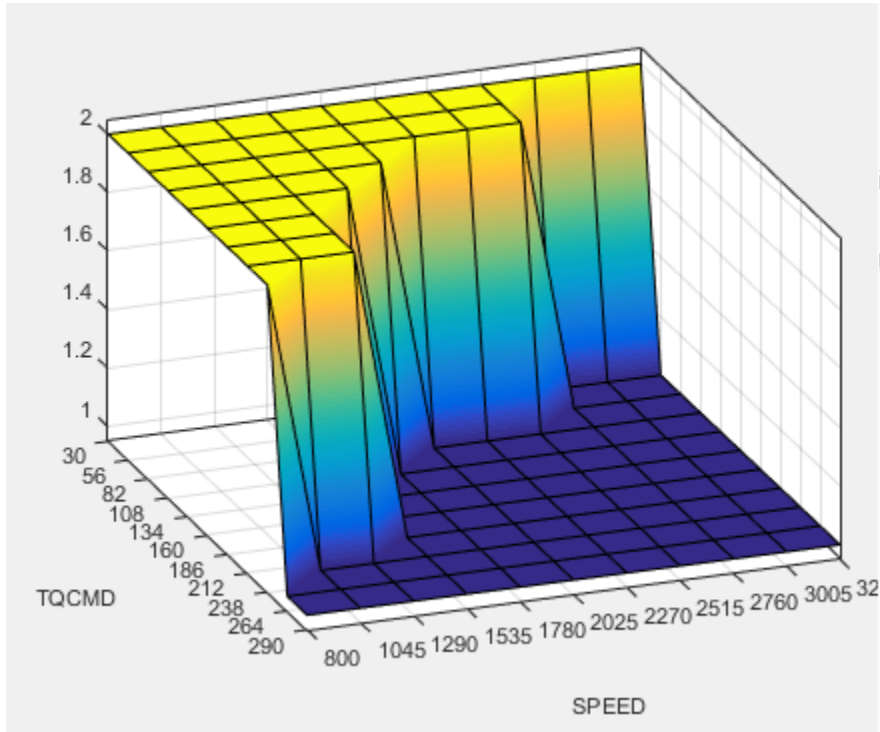
.

Pilot Mode Table

This graphic shows the plot of the table to select the active or inactive pilot mode depending on the speed and commanded torque

You need to fill calibration tables for each control variable described in "Multi-Injection Diesel Problem Definition" on page 6-2, in both pilot modes, active and inactive.

Following are all the pilot active tables.
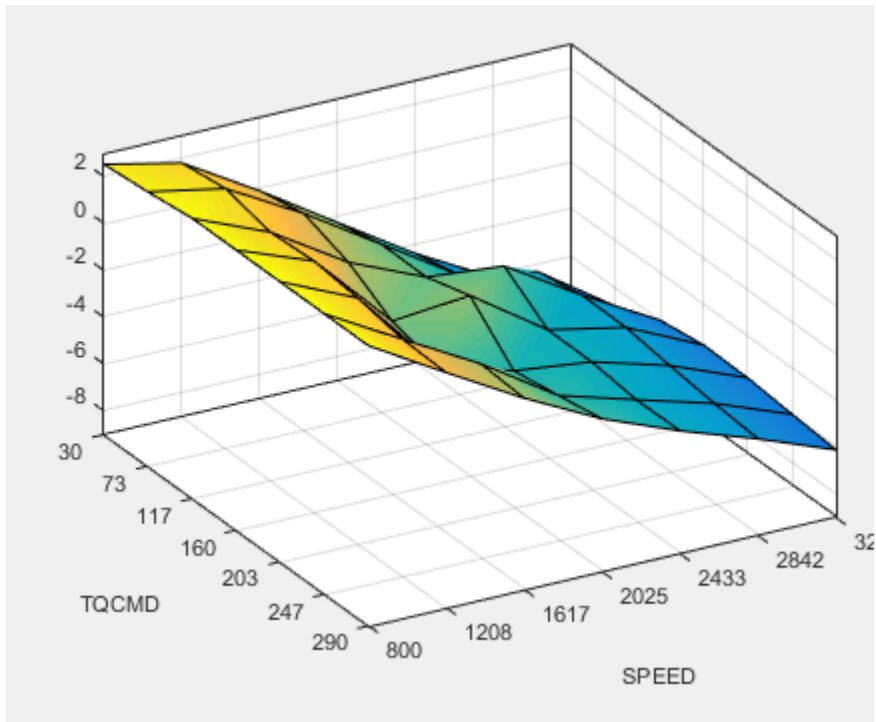
Main Start of Injection (SOI) Timing Table
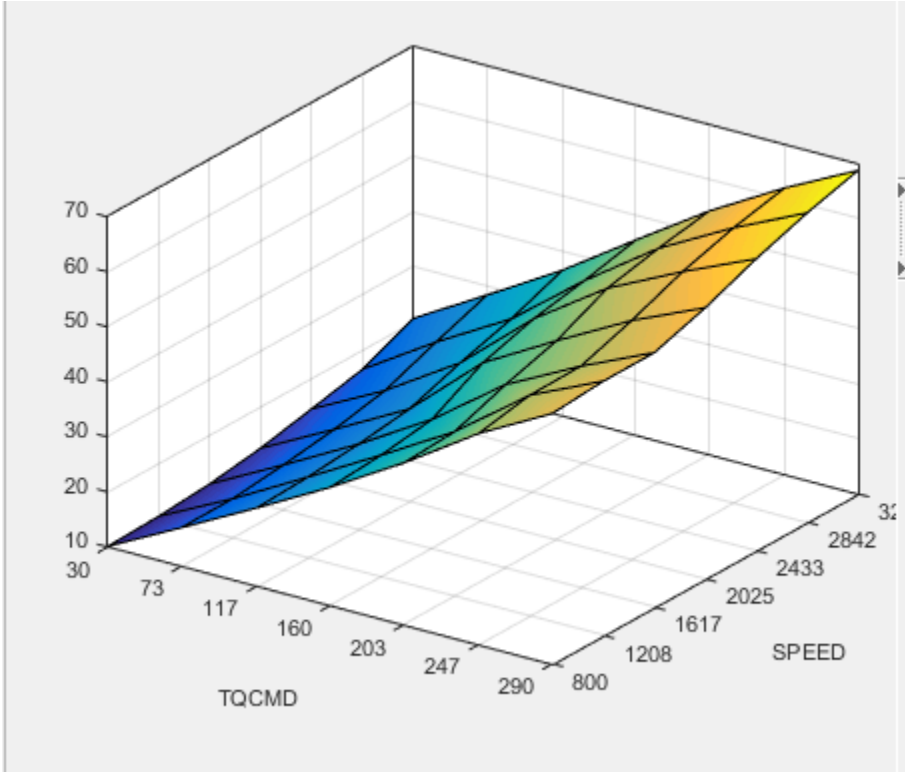
Total Injected Fuel Mass Table
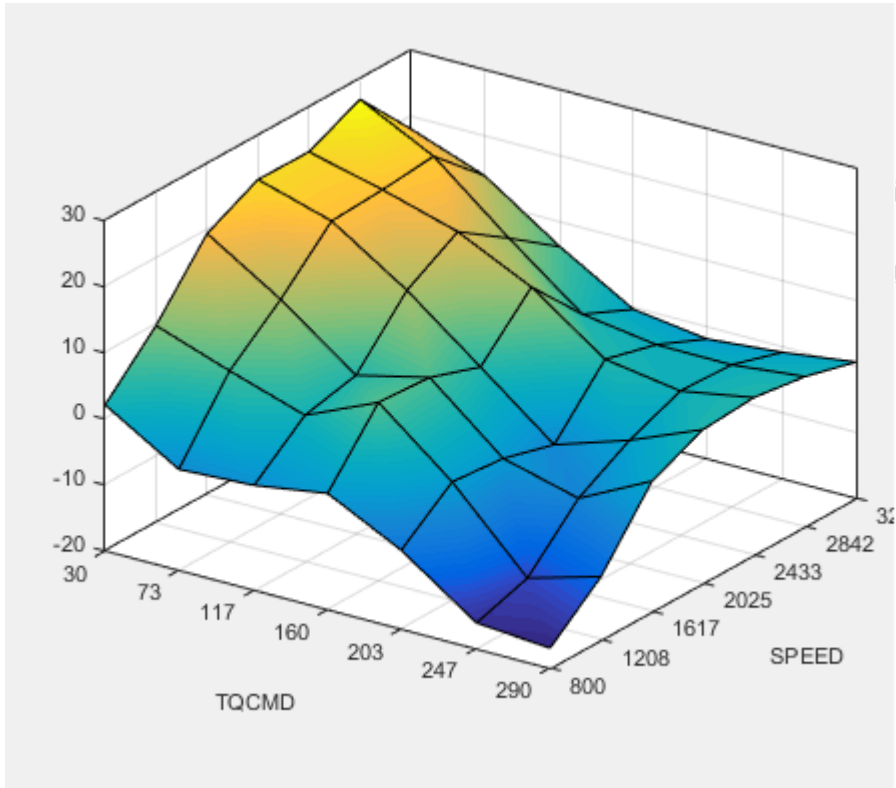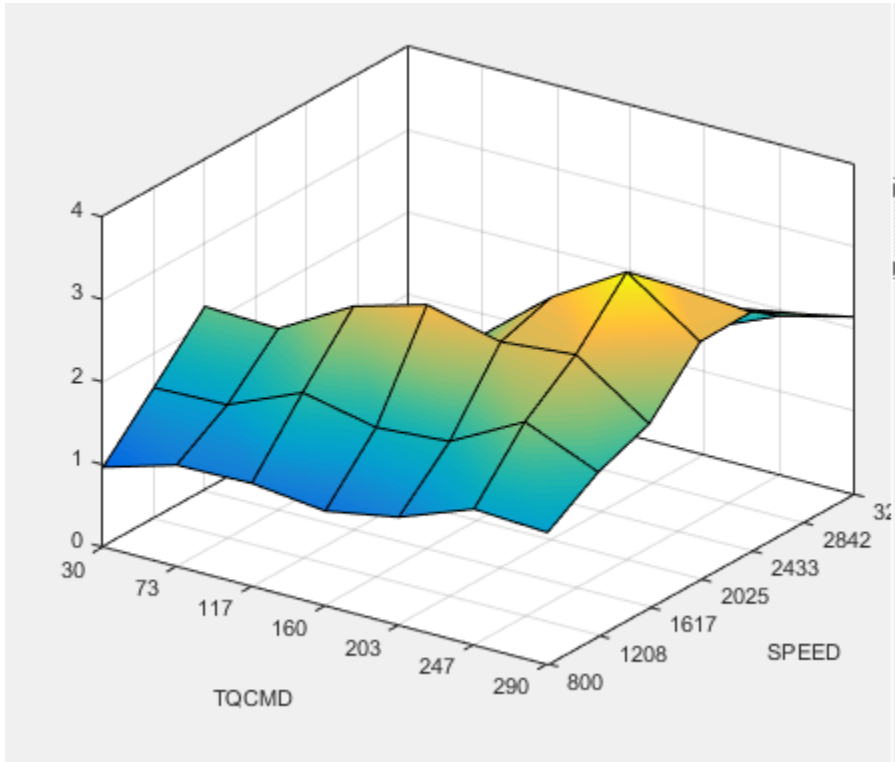
Fuel Pressure Delta Table

Exhaust Gas Recirculation (EGR) Valve Position Table
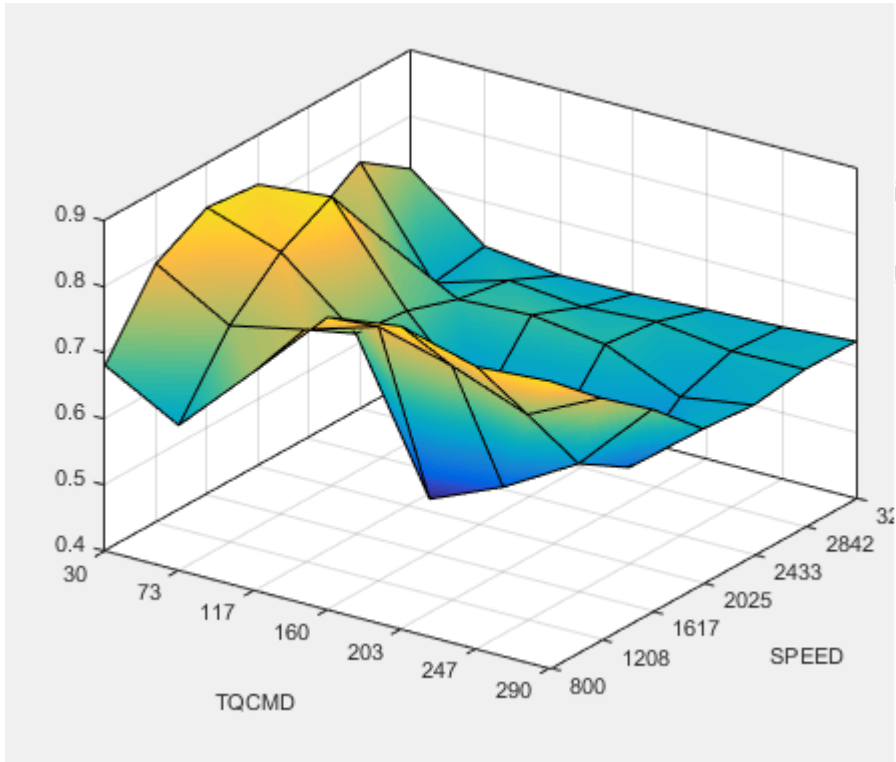
Variable-Geometry Turbo (VGT) Position Table
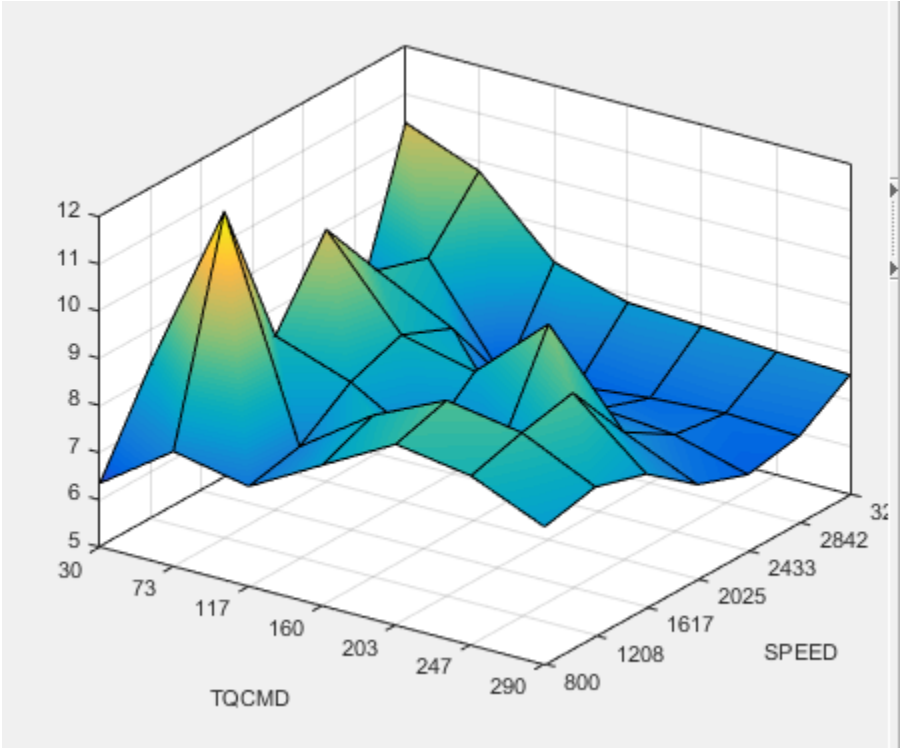
Pilot Injection Timing (Pilot SOI Delta) Table
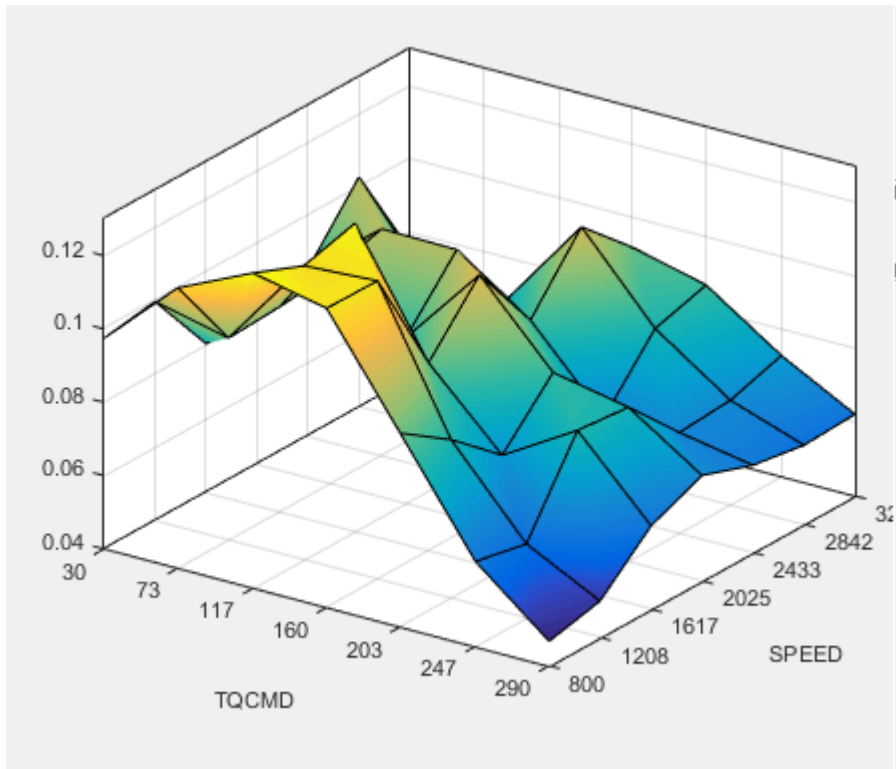
Pilot Fuel Mass Fraction Table

Case Study Example Files

The following sections guide you through opening example files to view each stage of the model-based calibration process. You can examine:

1   Designs, constraints, boundary model, and collected data, in topic "Design of Experiment" on page 6-22.

2   Finished statistical models, in topic "Statistical Modeling" on page 6-35.

3   Optimization setup and results, and filled calibration tables, in topic "Optimization" on page 6-42.

Use these example files to understand how to set up systematic calibrations for similar problems. For next steps, see "Design of Experiment" on page 6-22.

If you need more help on any stage of calibration, see the tutorials in "Getting Started with Model-Based Calibration Toolbox" for step-by-step examples.

Tip Learn how MathWorks Consulting helps customers develop engine calibrations that optimally balance engine performance, fuel economy, and emissions requirements: see Optimal Engine Calibration.

Design of Experiment

In this section...
"Benefits of Design of Experiment" on page 6-22
"Air-System Survey Testing" on page 6-22
"Create Designs and Collect Data" on page 6-23
"Fit a Boundary Model to Air Survey Data" on page 6-29
"Use the Air Survey and Boundary Model to Create the Final Design" on page 6-32
"Multi-Injection Testing" on page 6-34

Benefits of Design of Experiment

You use design of experiment to efficiently collect engine data. Testing time (on a dyno cell, or as in this case, using high-fidelity simulation) is expensive, and the savings in time and money can be considerable when a careful experimental design takes only the most useful data. Dramatically reducing test time is increasingly important as the number of controllable variables in more complex engines is growing. With increasing engine complexity, the test time increases exponentially.

Air-System Survey Testing

The first stage to solve this calibration problem is to determine the boundaries of the feasible air-system settings. To do this, create an experimental design and collect data to determine air-system setting boundaries that allow positive brake torque production in a feasible AFR range.
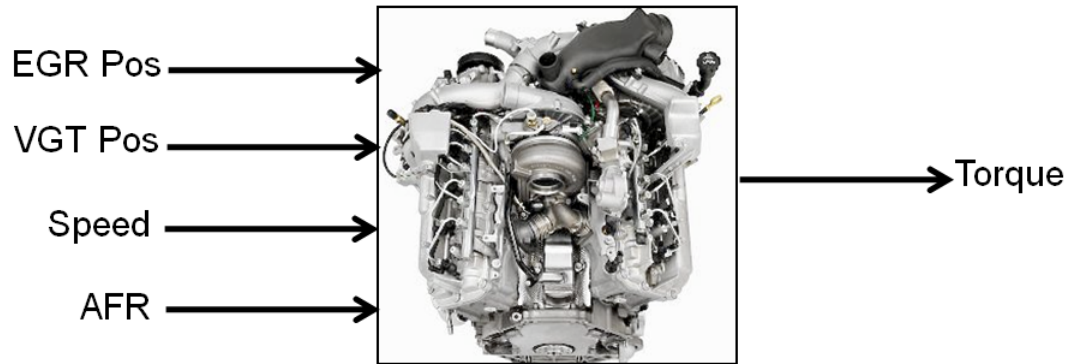
These simplifications were used to conduct the initial study:

- Pilot injection is inactive.
- Main timing is fixed.
- Nominal fuel pressure vs RPM.
- Main fuel mass is moved to match the AFR target.

The design process follows these steps:

1 Set up variable information for the experiment, to define the ranges and names of the variables in the design space.

2 Choose an initial model.

3 Create a base design that contains the correct constraints.

4 Create child designs using varying numbers of points and/or methods of construction.

5 Choose the design to run based on the statistics and how many points you can afford to run.

Create Designs and Collect Data

You can use a space-filling design to maximize coverage of the factors' ranges as quickly as possible, to understand the operating envelope.

To create a design, you need to first specify the model inputs. Open the example file to see how to define your test plan.

1   In MATLAB, on the **Apps** tab, in the **Automotive** group, click **MBC Model Fitting**.

2   Select **File > Open Project** and browse to the example file
    CI_MultiInject_AirSurvey.mat, found in matlab\toolbox\mbc
    \mbctraining.

3   The Model Browser displays the top project mode in the **All Models** tree,
    CI_Multiinject_AirSurvey.

4   To see how to define your test plan, click **Design Experiment**. In the new test plan
    dialog box, observe the inputs pane, where you can change the number of model
    inputs and specify the input symbols, signals and ranges. This example project
    already has inputs defined, so click **Cancel**.

5   Click the first test plan node in the **All Models** tree, AirSurveyDoE. The test plan
    view appears.
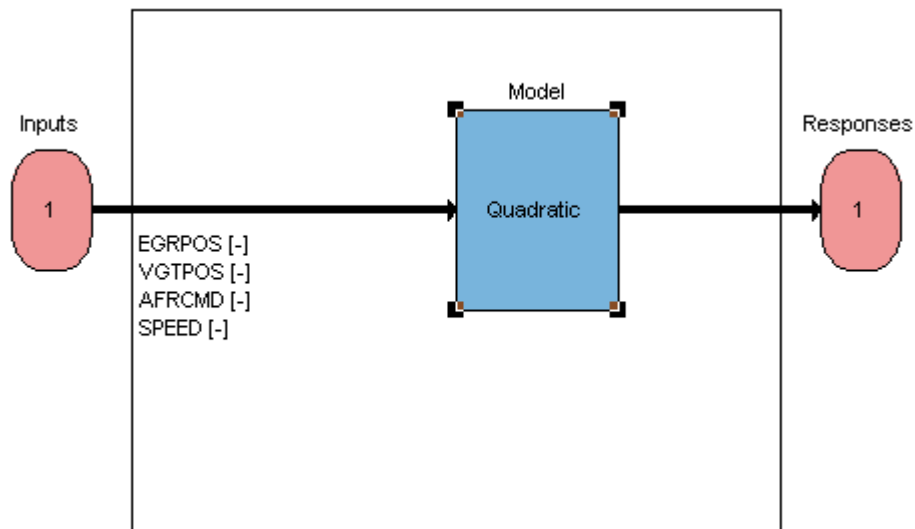
6   Observe the inputs listed on the test plan diagram. Double-click the **Inputs** block to
    view the ranges and names (symbols) for variables in the Input Factor Set Up dialog
    box.
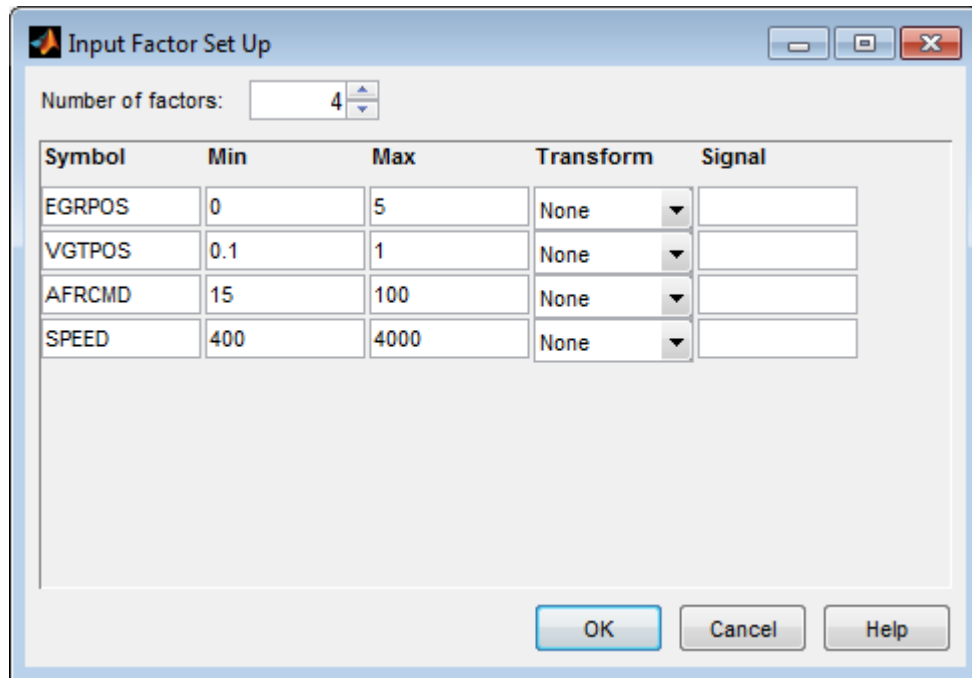
Close the dialog box.

7 After setting up inputs, you can create designs. In the **Common Tasks** pane, click **Design experiment**.

The Design Editor opens. Here, you can see how these designs are built:

- The final Air Survey design is a ~280 point Sobol Sequence design to span the engine operating space. The Sobol Sequence algorithm generated the space-filling design points. The final design is called `AirSurveyMergedDoE` because it contains two merged designs, one with a constraint and one unconstrained:

  - A 232-point design for overall engine operating range called `AirSurveySpaceFill`

  - A 50-point design low speed/load for an idle region called `AirSurveyIdle`

8 Select the `AirSurveySpaceFill` design in the Designs tree, and select **View > Current View > 2D Design Projection**.
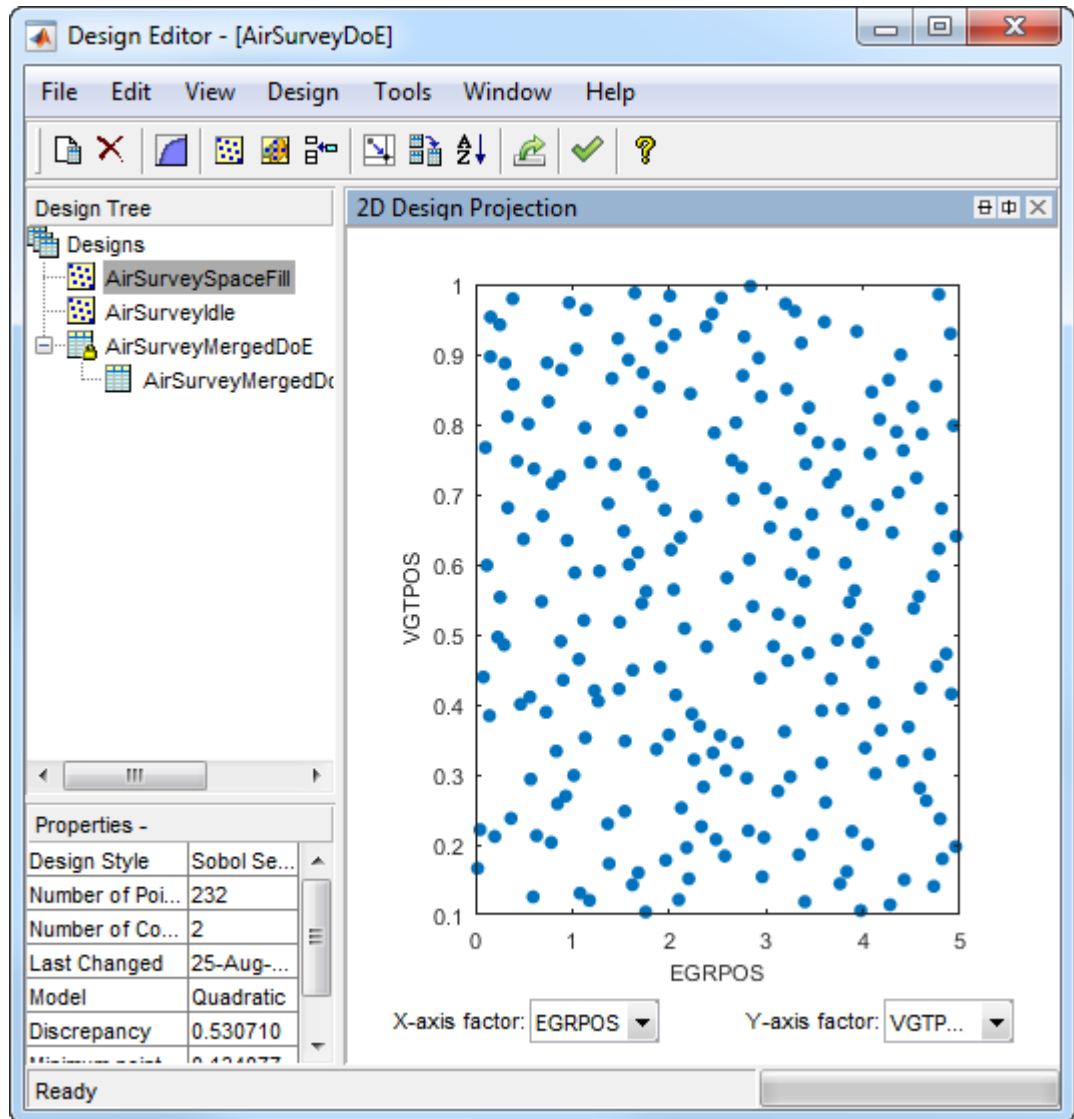
9 Select **Edit > Constraints** to see how the constraints are set up.
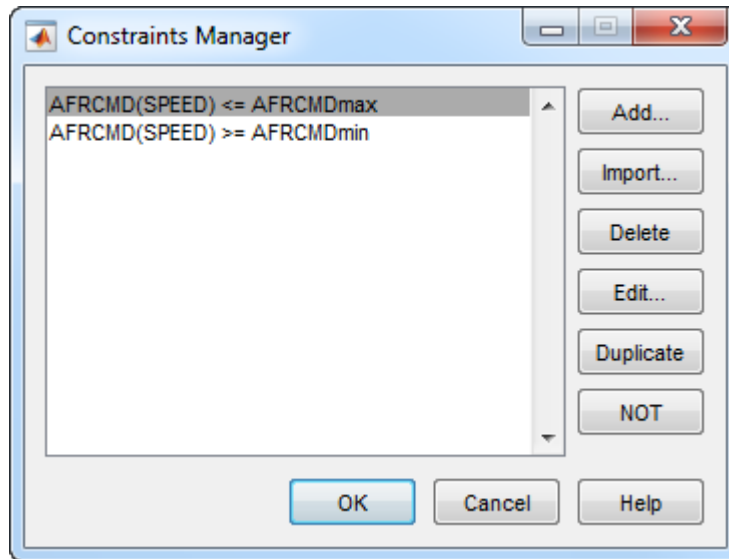
10 In the Constraints Manager dialog box, select a constraint and click **Edit**. Observe
    that you can define areas to exclude by dragging points or typing in the edit boxes.
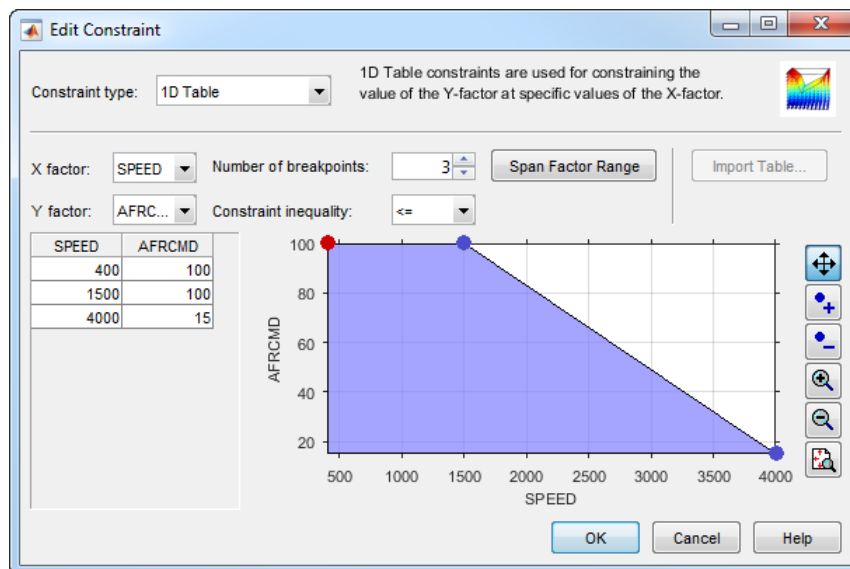
Click **Cancel** to close the Edit Constraint dialog box and leave the constraint unchanged. Close the Constraints Manager dialog box.

11 Observe the `Properties` of the selected `AirSurveySpaceFill` design under the tree, listing 2 constraints, 232 points, and a `Design Style` of `Sobol Sequence`.

12 You can see how to construct a design like this by selecting **Design** > **Space Filling** > **Design Browser**, or click the space-filling design toolbar button.

a   • A dialog box asks what you want to do with the existing design points. Click **OK** to replace the points with a new design.

•   In the Space-Filling Design Browser, click **Generate** several times to try different space-filling points. You can specify the **Number of points**, and view previews of design points. Observe the default design type is `Sobol Sequence`.
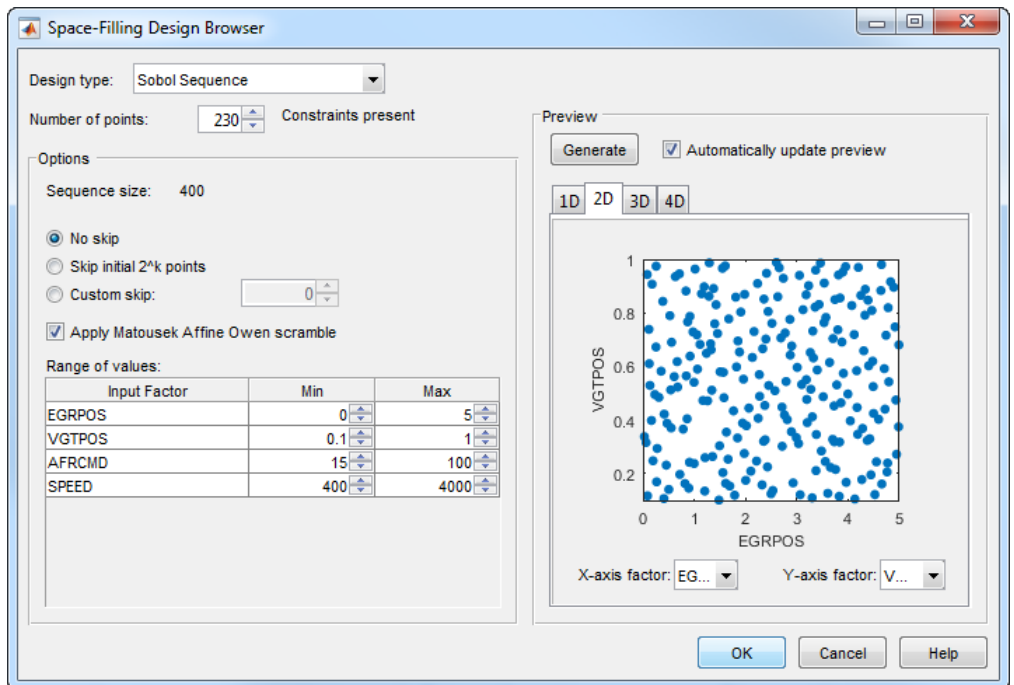
•   Click **Cancel** to close the Space-Filling Design Browser and leave the design unchanged.

13 Click the `AirSurveyIdle` design to observe it is also a `Sobol Sequence` design, containing 50 points and no constraints.

14 Click the `AirSurveyMergedDoE` design to observe it is of type `Custom`, and contains 282 points and 2 constraints. This design is formed by merging the previous 2 designs. You can find **Merge Designs** in the **File** menu.

15 Click `AirSurveyMergedDoE_RoundedSorted`, the child design of `AirSurveyMergedDoE`. This design contains the same points but rounded and sorted, using **Edit > Sort Points** and **Edit > Round Factor**. This is the final design used to collect data.

16 Close the Design Editor.

The final air survey design was used to collect data from the GT-Power model with the Simulink and Simscape test harness. The example Model Browser project file `CI_MultiInject_AirSurvey.mat` contains this data, imported to the Model Browser after the air survey. You can also view the data in spreadsheet form in the file `CI_AirSurvey_Data.xlsx` in the `mbctraining` folder.

Fit a Boundary Model to Air Survey Data

You need to fit a boundary model to the data collected at the air survey design points. The test data from the air survey was used to create a boundary model of the engine operating range. The example Model Browser project file `CI_MultiInject_AirSurvey.mat` contains this boundary model.

1 In the Model Browser, click the second test plan node in the **All Models** tree, `AirSurveyBoundary`.

Compare with the `AirSurveyDoE` test plan view. For the boundary modeling test plan,

- Observe an imported data set listed under the **Common Tasks** pane. The second test plan has imported the air survey data in order to fit a boundary model to the data.

- Observe the `Inputs` are different in the test plan diagram. Instead of the `AFRCMD` input in the DoE test plan, there is a `SPEED` input for boundary modeling. `AFRCMD` was used to constrain the design points to collect the air survey data. To model the boundary before creating the final design, you now need the `SPEED` input instead.

2   In the Model Browser, select **TestPlan > Edit Boundary** to open the Boundary Editor.

3   Examine the boundary model by selecting **View > Current View > Pairwise Projections**. The plots show the shape across the inputs.
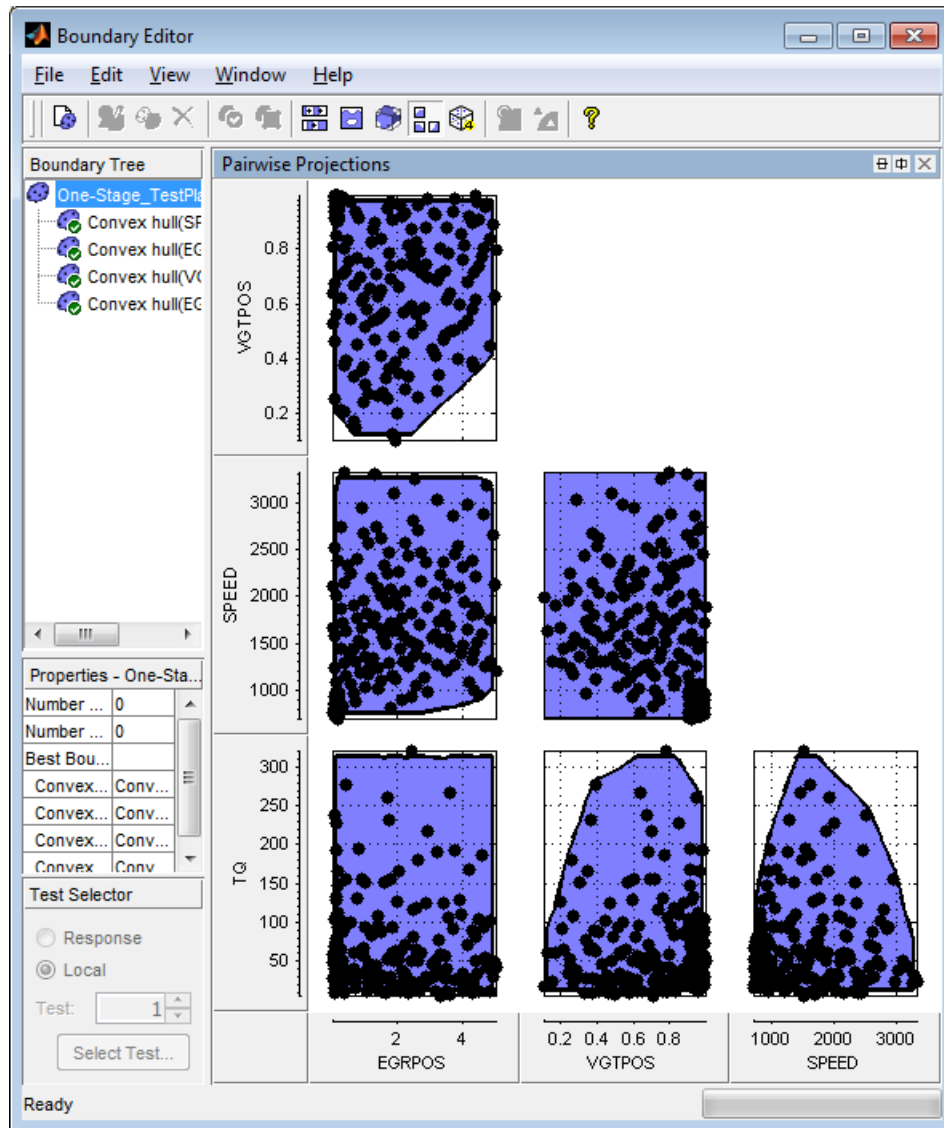
Use the Air Survey and Boundary Model to Create the Final Design

The initial air survey design and test data provide information about the engine operating envelope, where the feasible AFR range can produce positive brake torque. The resulting data lets you create a boundary model. You can use the boundary model to create the final design avoiding infeasible points, and in later steps to guide modeling and constrain optimizations.

Using the boundary model as a constraint, you can generate the final design to collect detailed data about fuel injection effects within those boundaries. You can then use this data to create response models for all the responses you need in order to create an optimal calibration for this engine.

The final Point-by-Point Multi-Injection design (around 7000 points) was generated using a MATLAB script together with the Air Survey boundary model.

1   Open the example files `CI_PointbyPointDoE.m` and `createCIPointbyPointDoEs.m` in the `mbctraining` folder to see the script commands that build the final design.

     The script keeps only points that lie within the boundary model, and continues generating design points until it reaches the desired number of points. The Sobol space-filling design type keeps filling in spaces for good coverage of the design space.

     After generating design points for each test, the script creates a Model Browser project with a point-by-point test plan, and attaches the point-by-point designs to the test plan.

2   Open the project `CI_MultiInject_PointbyPoint.mat` to view the project created by the script.

3   Select the second test plan node in the tree, and then click the Test Plan tab. In the test plan diagram, right-click the Operating Point Selector block, and select **Design Experiment** to view the designs created by the script.
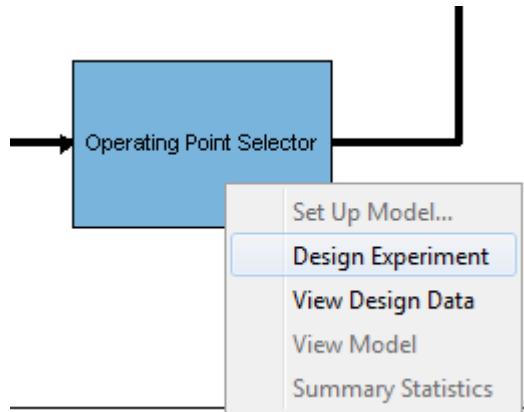
4   In the Design Editor, select **Mode Points** in the Design tree, and view the 2D Design Projection.

5   In the Design tree, select **Actual Design**. Observe that the boundary model constraint removed points in the corners, so that the actual design points collect data only in the feasible region determined by the initial air survey.

6   Return to the Model Browser test plan, right-click the Local Model block, and select **Design Experiment**.
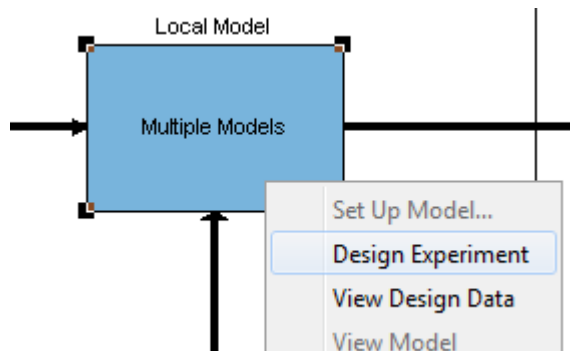
7   Click test numbers to view the local design points at each operating point. At each test, the values of SPEED and TQ are fixed and the space-filling design selects points across the space of the local inputs.

Multi-Injection Testing

The final design was used to collect data for the following inputs and responses.
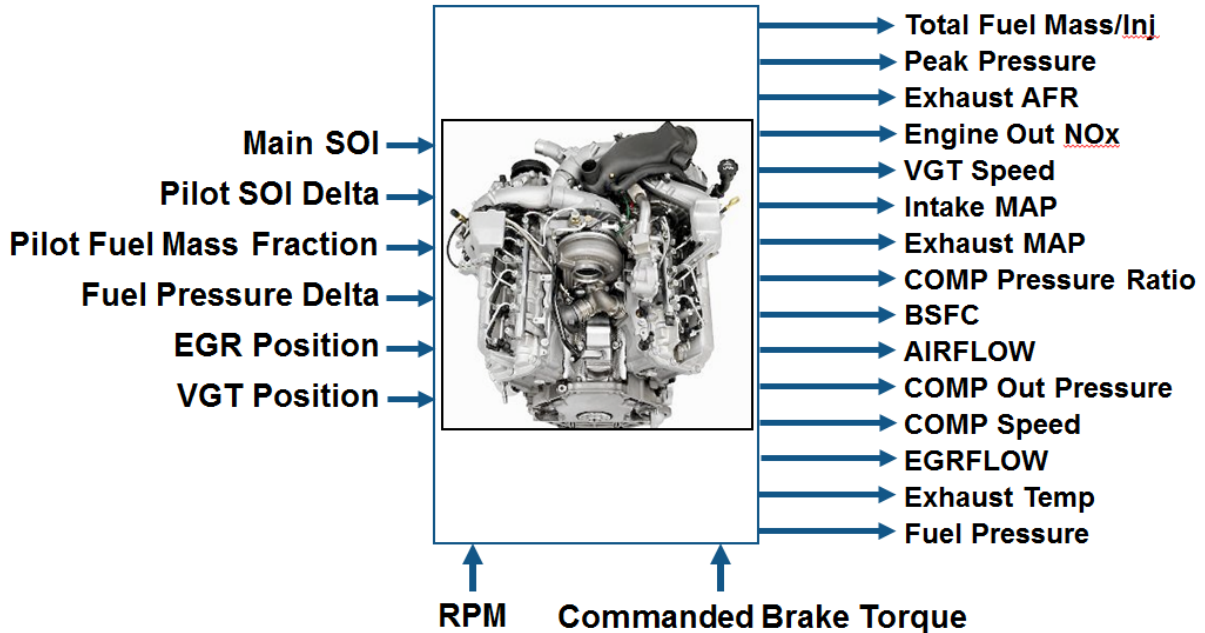
The toolbox provides the data files for you to explore this calibration example. You can view the data in spreadsheet form in the file `CI_MultiInject_PointByPoint_Data.xls`, and the data is imported to the Model Browser project file `CI_MultiInject_PointbyPoint.mat`.

For details on how the data was collected, see "Data Collection and Physical Modeling" on page 6-10.

For next steps, see "Statistical Modeling" on page 6-35.

Tip Learn how MathWorks Consulting helps customers develop engine calibrations that optimally balance engine performance, fuel economy, and emissions requirements: see Optimal Engine Calibration.

Statistical Modeling

In this section...

Examine the Test Plans for Point-by-Point Models

After designing the experiments and collecting the data, you can fit statistical models to the data. You can use the toolbox to generate accurate, fast-running models from the measured engine data.

The following graphic shows the models to define in the toolbox to solve this calibration problem. The graphic shows how the model inputs and output relate to the optimal tables, optimization operating points, objectives and constraints you need to perform the optimization and create the calibration.

I/O of Multi-Inject 3.1L Common Rail Engine Model with Variable Geometry Turbocharger and Cooled EGR

Goal: Minimize BSFC subject to NOx, turbocharger speed, and user-specified table gradient constraints

The toolbox provides the data for you to explore this calibration example. For details on how the data was collected, see "Data Collection and Physical Modeling" on page 6-10.

Examine the model setup.

1   In MATLAB, on the **Apps** tab, in the **Automotive** group, click **MBC Model Fitting**.

2   In the Model Browser home page, in the **Case Studies** list, select **Multi-injection diesel tested with pilot injection on and off**. Alternatively, select **File > Open Project** and browse to the example file CI_MultiInject_PointbyPoint.mat, found in matlab\toolbox\mbc\mbctraining.

3   The Model Browser remembers views, so change to the test plan view if necessary, by clicking the PilotInactivePointbyPoint test plan node in the **All Models** tree.

The **Response Models** tab shows the cross-section view of the responses. Here you can assess high-level model trends.
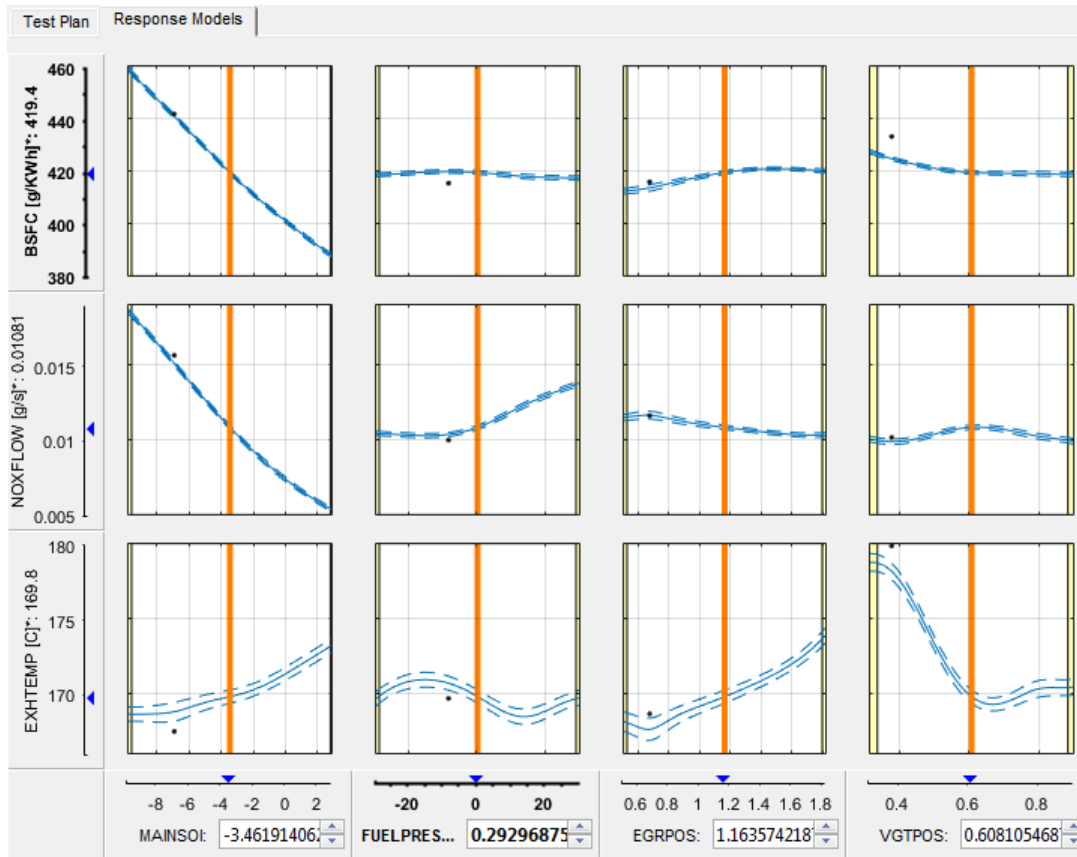
4 Select the **Test Plan** tab.
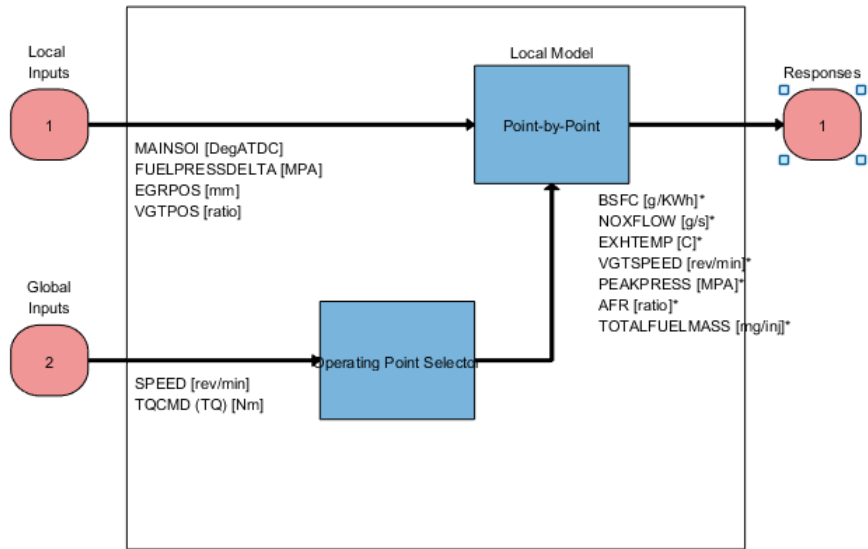
5. Observe the inputs and response model outputs listed on the test plan diagram. This is a Point-by-Point test plan. The Global Inputs SPEED and TQCMD select the operating points for each model.

6. Double-click the **Inputs** blocks to view the ranges and names (symbols) for variables on the Input Factor Set Up dialog box.

7. Double-click the **Local Model** block to view that the Point-by-Point Model Setup dialog. View the list of alternative models to fit at each operating point for a point-by-point test plan. When you use the **Fit models** button in the **Common Tasks** pane, and select a Point-by-Point template, the toolbox selects this list of models. This model setup fits four alternative model types at each operating point, and selects the best model based on the **Criteria** setting, in this case PRESS RMSE.
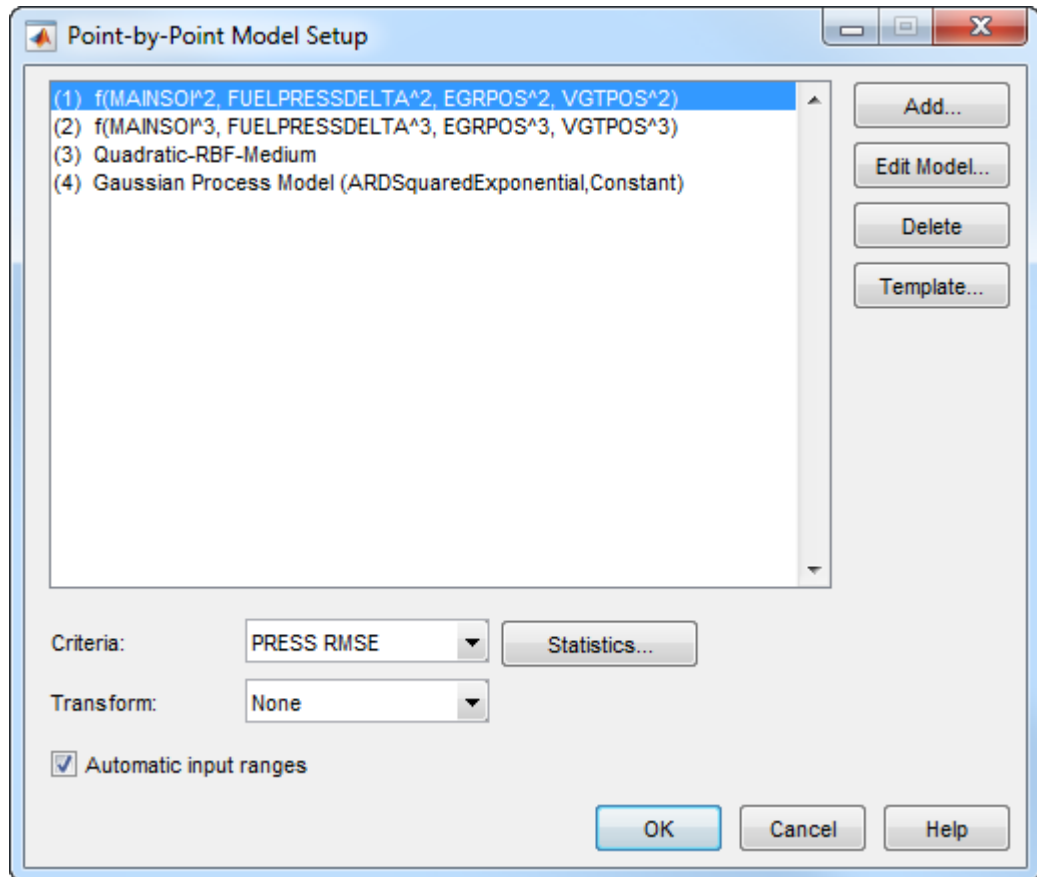
8 Click **Cancel** to close the Model Setup dialog box without altering your example models.

9 Similarly, examine the `PilotActivePointbyPoint` test plan. This test plan has the same response models and model type setup, but two more local inputs for the pilot injection timing and mass, `PILOTDELTASOI` and `PILOTFMF`. The data used to fit the models is also different, with the two additional factors. Observe the **Dataset** information under the **Common Task** pane in the test plan view.

For details on setting up point-by-point models, see "Fit a Point-by-Point Model".

Examine Response Models

1 Expand the `PilotInactivePointByPoint` test plan node in the **All Models** tree and select the nodes for each response name, e.g., `BSFC`.

2 Click through tests to see the model selected as best at each point.

The toolbox tries to select the best model for each operating point based on the selected statistical criteria. However, you should always verify the model choices. To search for the best fit you must examine each model, consider removing outliers, and compare with alternative fits. The example models show the results of this process.

3 Look at the RMSE Plots. These plots can help you identify problem tests to investigate. Double-click to view a test of interest (with larger RMSE) in the other plots.
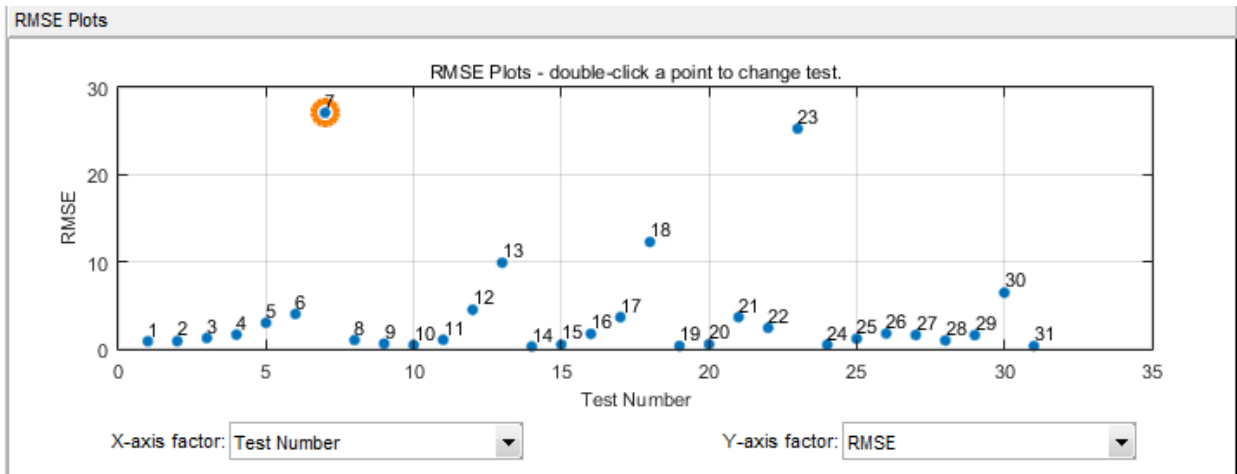
4 Choose a model to use at a particular operating point by selecting the **Best Model** check box in the list of **Alternative Local Models**.

For details on analyzing point-by-point models, see "Analyze Point-by-Point Models and Choose the Best".

For next steps, see "Optimization" on page 6-42.

Tip Learn how MathWorks Consulting helps customers develop engine calibrations that optimally balance engine performance, fuel economy, and emissions requirements: see Optimal Engine Calibration.

Optimization

In this section...

Optimization Overview

After creating the statistical models to fit the data, you can use them in optimizations. You can use the accurate statistical engine model to replace the high-fidelity simulation and run much faster, enabling optimization to generate calibrations in feasible times.

1   Run an optimization to choose whether to use Pilot Injection at each operating point.

2   Optimize fuel consumption over the drive cycle, and meet these constraints:

   - Constrain total NOx
   - Constrain turbocharger speed
   - Constrain smoothness of tables

3   Fill lookup tables for all control inputs.

Set Up Models and Tables for Optimization

To perform an optimization, you need to import the statistical models created earlier in the Model Browser.

1   In MATLAB, on the **Apps** tab, in the **Automotive** group, click **MBC Optimization**.

2   To see how to import models, on the home page, click **Import From Project**

   The CAGE Import Tool opens. Here, you can select models to import from a model browser project file or direct from the Model Browser if it is open. However, the

toolbox provides an example file with the models already imported, so click **Close** to close the CAGE Import Tool.

3 On the home page, in the Case Studies list, select the multi-injection example, or select **File > Open Project** and browse to the example file CI_MultiInject.cag, found in matlab\toolbox\mbc\mbctraining.

4 Click **Models** in the left **Data Objects** pane to view the models.

5 To see how to set up tables to fill with the results of your optimizations, select **Tools > Create Tables from Model**. The Create Tables from Model wizard appears.
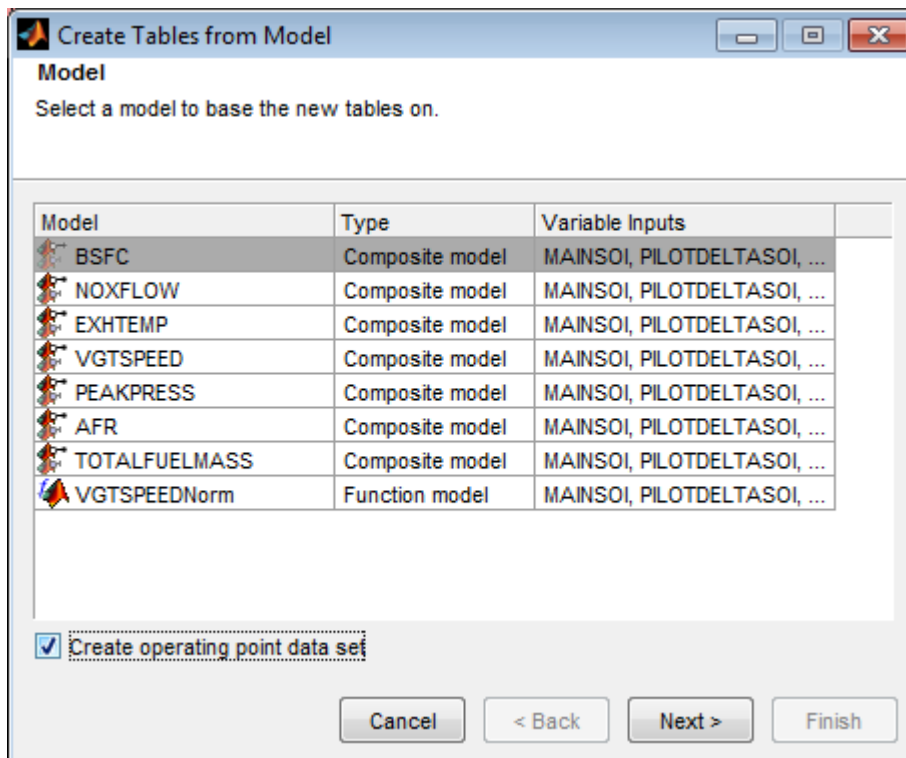
6 Select the model BSFC and the **Create operating point data set** check box, and click **Next**.

7 If you left the defaults, you would use the model operating points for the table breakpoints. However, you need to create different tables to the operating points.

   a Clear the **Use model operating points** check box.

    b   Select SPEED and TQCMD for the axis inputs.

    c   Enter 11 for the table rows and columns.
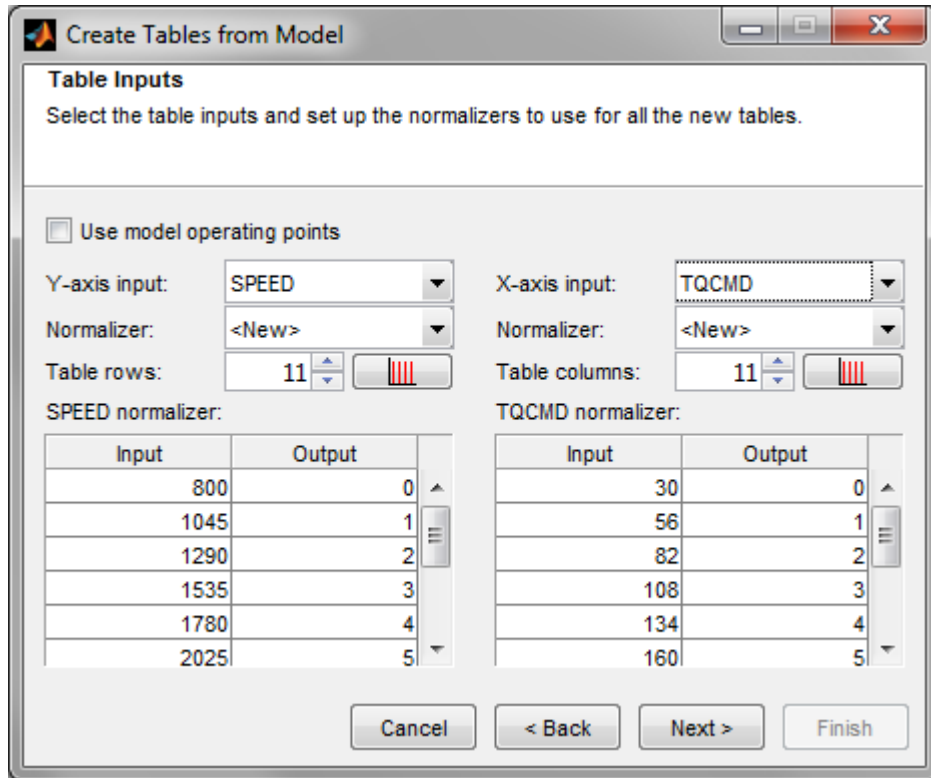
Click **Next**.

8   View the last screen to see how many tables CAGE will create for you if you finish the wizard.

Click **Cancel** to avoid creating new unnecessary tables in your example file. The example file already contains all the tables needed for your calibration results.
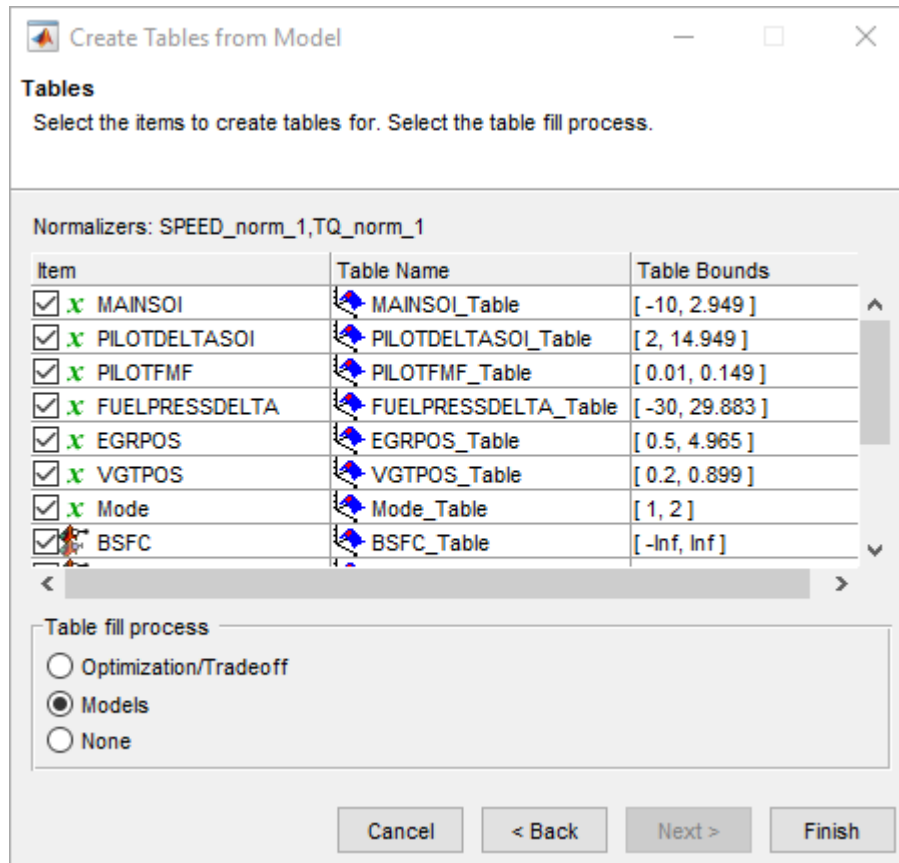
9 Select the **Tables** view to see all the tables. Observe there are tables named `_wPilot` and `_noPilot`, that will be filled by optimization results. The goal is to fill separate tables for each mode, pilot injection active and no pilot injection.

Examine the Point Optimization Setup

The example file `CI_MultiInject.cag` shows the results of the multistage process to finish this calibration.

1 Click **Optimization** in the left **Processes** pane to view the two optimizations.
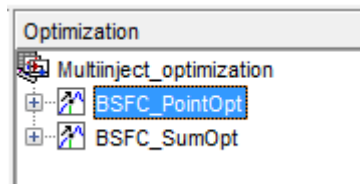
Why are there two optimizations? To complete the calibration, you need to start with a point optimization to determine whether to use Pilot Injection at each operating point. You use the results of this point optimization to set up the sum optimization to optimize fuel consumption over the drive cycle, and meet these constraints:

- Constrain total NOx
- Constrain turbocharger speed
- Constrain smoothness of tables

2 Select the `BSFC_PointOpt` optimization to view the setup.

3 View the two objectives in the Objectives pane. These are set up to minimize `BSFC` and `PEAKPRESS`. Double-click an objective to view its settings.

4 Observe that the Constraints pane shows a boundary model constraint.

5 Learn the easiest way to set up an optimization like this by selecting **Tools > Create Optimization from Model**.
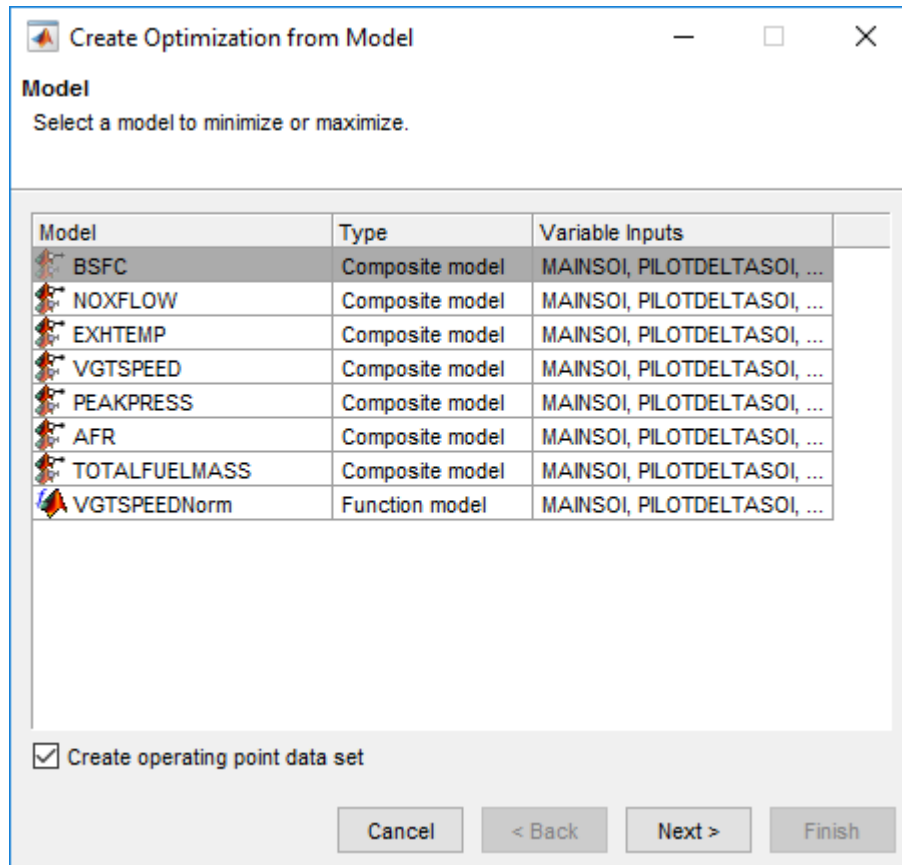
6   Select the model BSFC and the **Create operating point data set** check box, and click **Next**.

7   Observe that the default settings will create a point optimization to minimize BSFC at the model operating points, using six free variables and not SPEED, TQCMD, or PilotMode, and constrained by a model boundary constraint.
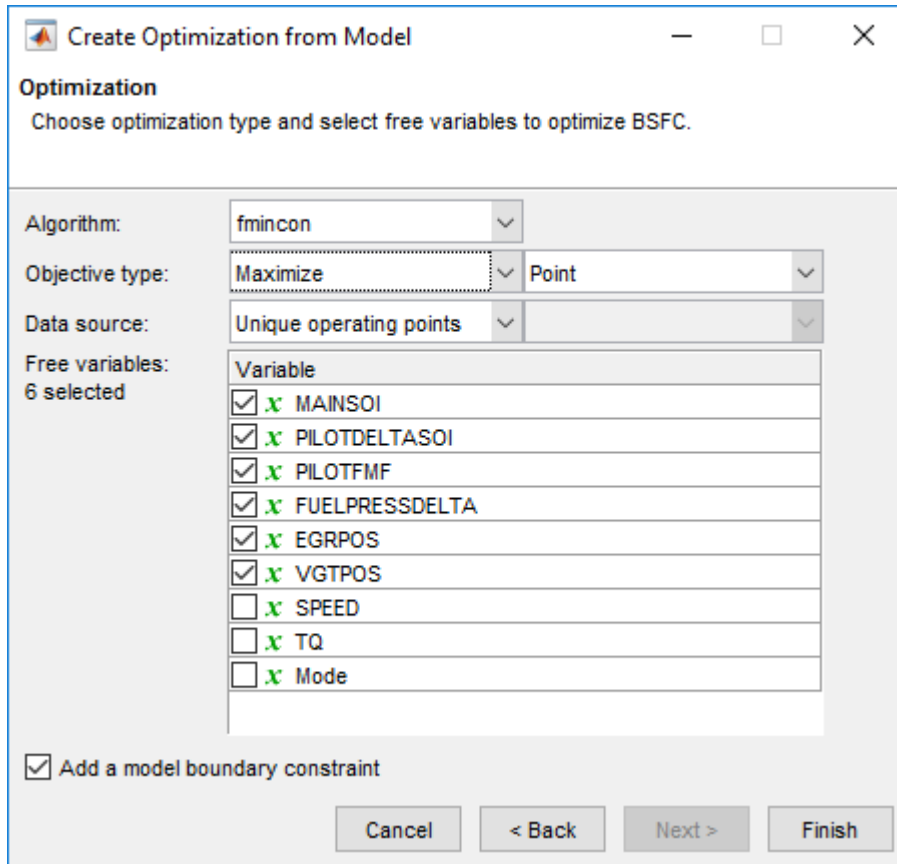
8   You want the optimization to identify which pilot mode (active pilot injection or no
    pilot injection) is best to use at each operating point. To do this, you must select
    `Modal optimization` from the **Algorithm** list.

9   You want to include `PilotMode` as a free variable, so the optimization can identify
    which mode is best to use at each operating point. From the **Free variables** list,
    select the check box to include `PilotMode`.
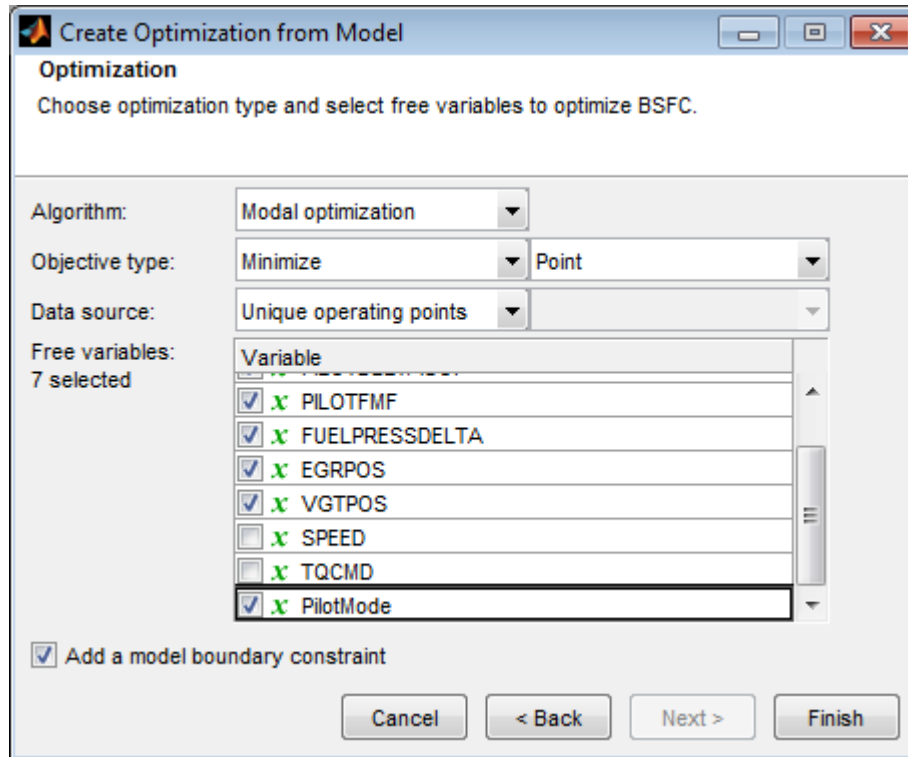
10 Click **Finish** to create the new optimization.

11 Compare your new optimization with the example BSFC_PointOpt. Notice the example has a second objective, to minimize PEAKPRESS. To see how to set this up, right-click the Objectives pane and select **Add Objective**.

Examine the Point Optimization Results

The example file CI_MultiInject.cag shows the results of the multistage process to finish this calibration.

1 Expand the first optimization node, BSFC_PointOpt, and select the output node underneath, BSFC_PointOpt_output.

2 Select **View > Selected Solution** (or the toolbar button) to view which pilot mode the optimization selected as best at each operating point. You can see the selected

value of 1 or 2 at each point in the **Results Contour** plot and the **PilotMode** column in the **Optimization Results** table.
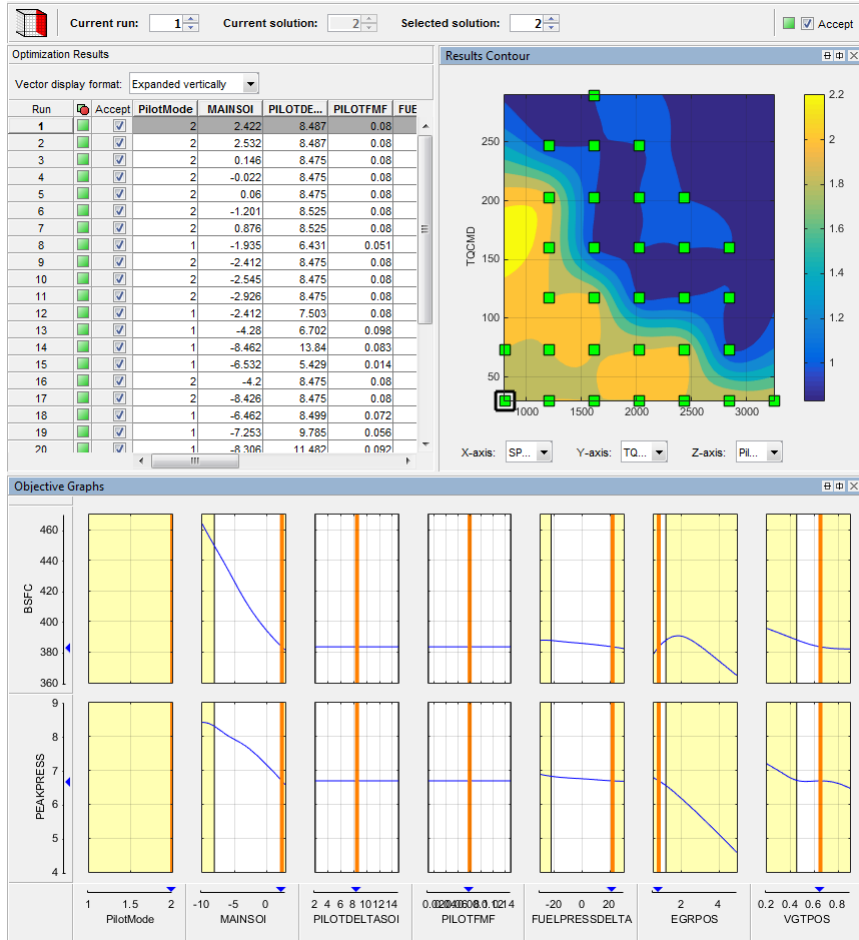
3. Review the Results Contour plot to see which mode has been selected across all operating points. Use this view to verify the distribution of mode selection.

4. Click to select a point in the table or Results Contour, and you can use the **Selected solution** controls at the top to alter which mode is selected at that point. You might want to change selected mode if another mode is also feasible at that point. For example, you can change the mode to make the table more smooth.

5 Use the other objectives to explore the results. For example, you might want to manually change the selected mode based on an extra objective value. It can be useful to view plots of the other objective values at your selected solutions.

    a To display another plot simultaneously, right-click the Results Contour title bar and select **Split View**.

    b Plot the objective BSFC on the Results Surface and observe the difference when you change a selected solution.

    c Plot PEAKPRESS on the Results Surface and observe the difference when you change a selected solution.
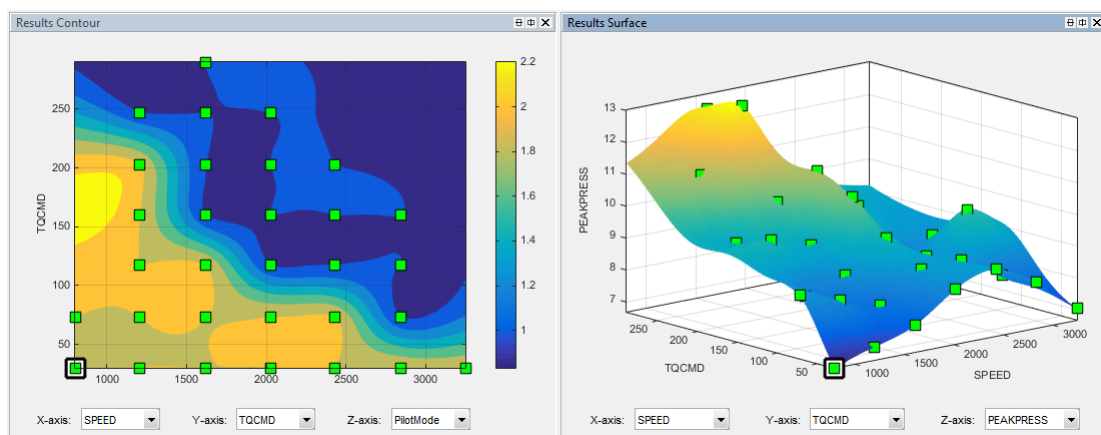
6 To see both solutions for a particular operating point, use the Pareto Slice view. You can inspect the objective value (and any extra objective values) for each solution. If needed, you can manually change the selected mode to meet other criteria, such as the mode in adjacent operating points, or the value of an extra objective.

For details on tools for choosing a mode at each operating point, see "Analyzing Modal Optimization Results".

Create Sum Optimization from Point Optimization

The point optimization results determine whether to use pilot injection at each operating point. When you are satisfied with all selected solutions for your modal optimization, you can make a sum optimization over all operating points. The pilot injection mode must be fixed in the sum optimization to avoid optimizing too many combinations of operating modes.

The results of the point optimization were used to set up the sum optimization to optimize fuel consumption over the drive cycle. To see how to do this:

1 From the point optimization output node, `BSFC_PointOpt_output`, select **Solution > Create Sum Optimization**.

   The toolbox automatically creates a sum optimization for you with your selected best mode for each operating point. The create sum optimization function converts the modal optimization to a standard single objective optimization (`fmincon` algorithm) and changes the `Mode Variable` to a fixed variable.

2 Compare your new optimization with the example sum optimization, `BSFC_SumOpt`. The example shows you need to set up more constraints to complete the calibration.

Constraints			
Name	Description	Application Point Set	Status
BSFC_Boundary	Boundary constraint of BSFC(M...		
NOxSum	Weighted sum of NOXFLOW(M...		
VGTSpeedMax	VGTSPEEDNorm(MAINSOI, PILO...		
MainSOI_wPilot	Gradient constraint of MAINSOI ...	PilotActive(SPEED,TQCMD;PilotMode)	
MainSOI_noPilot	Gradient constraint of MAINSOI ...	PilotInactive(SPEED,TQCMD;PilotMode)	
EGRPos_wPilot	Gradient constraint of EGRPOS ...	PilotActive(SPEED,TQCMD;PilotMode)	
EGRPos_noPilot	Gradient constraint of EGRPOS ...	PilotInactive(SPEED,TQCMD;PilotMode)	
FuelPressDelta_w...	Gradient constraint of FUELPRE...	PilotActive(SPEED,TQCMD;PilotMode)	
FuelPressDelta_n...	Gradient constraint of FUELPRE...	PilotInactive(SPEED,TQCMD;PilotMode)	
VGTPos_wPilot	Gradient constraint of VGTPOS ...	PilotActive(SPEED,TQCMD;PilotMode)	
VGTPos_noPilot	Gradient constraint of VGTPOS ...	PilotInactive(SPEED,TQCMD;PilotMode)	
PilotDeltaSOI_wPilot	Gradient constraint of PILOTDEL...	PilotActive(SPEED,TQCMD;PilotMode)	
PilotFMF_wPilot	Gradient constraint of PILOTFM...	PilotActive(SPEED,TQCMD;PilotMode)	

The additional constraints make the optimization meet these requirements:

- Constrain total NOx
- Constrain maximum turbocharger speed
- Constrain smoothness of tables with gradient constraints

3 Import all required constraints to your new optimization by selecting **Optimization > Constraints > Import Constraints**. Select the example sum optimization and import all but the first boundary model constraint.

4 Double-click the additional constraints to open the Edit Constraint dialog box and view the setup.

   This sum constraint controls the total NOx.
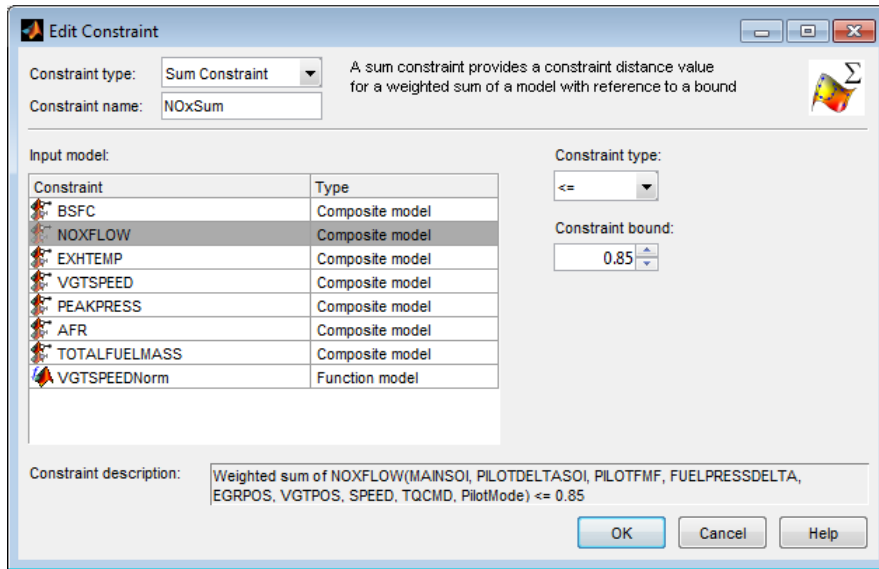
This constraint controls maximum turbocharger speed.
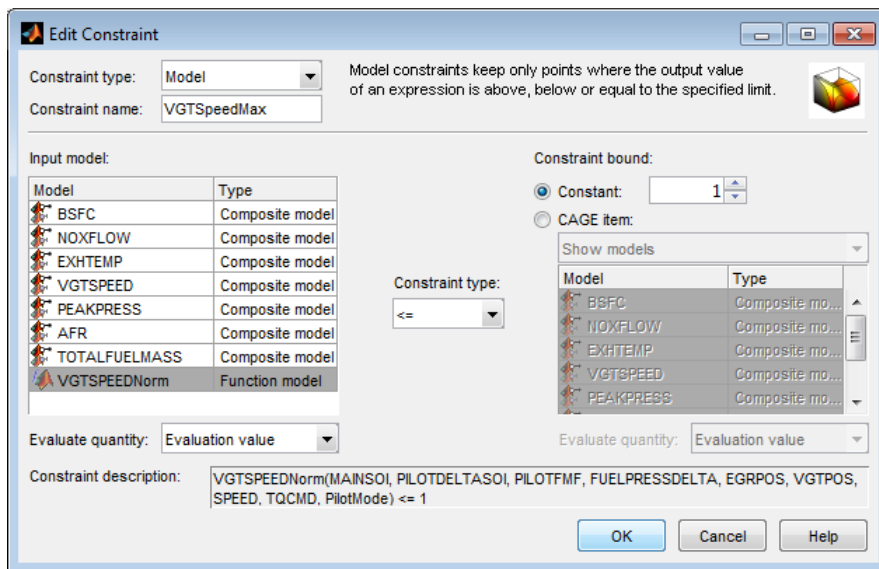
Observe that this constraint does not use the VGTSPEED model directly, but instead uses a function model VGTSPEEDNorm. You can examine this function model in the Models view. The function model scales the constraint to help the optimization routines, which have problems if constraints have very different sizes. VGTSPEED is of order of 165000, while the other constraints are of the order of 1, so the function model VGTSPEEDNorm normalizes VGTSPEED by dividing it by 165000.

The following constraint controls the gradient across the MAINSOI table.
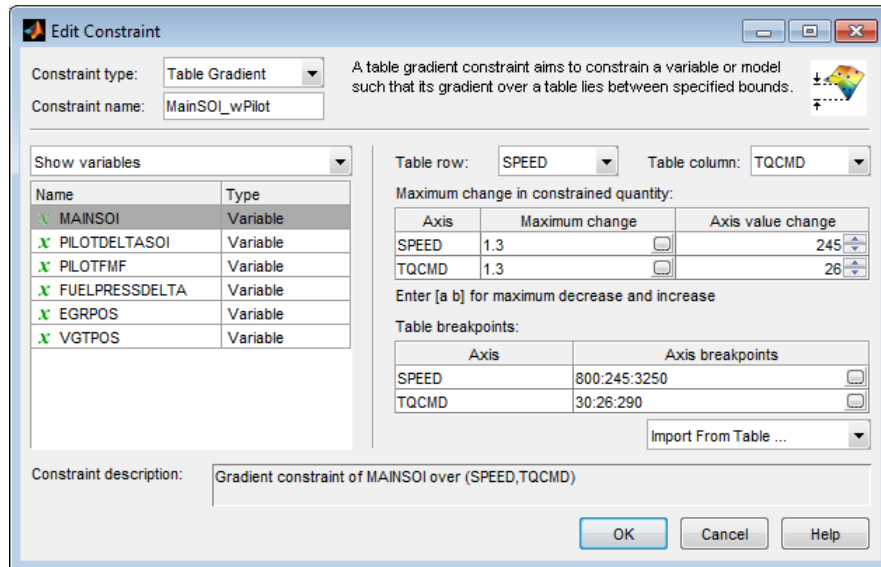
5 In the Optimization view, in the Constraints pane, observe that:

• The gradient constraints are in pairs, one with pilot and one with no pilot. Separate table gradient constraints are required for different modes because the goal is to fill separate tables for each mode.

• All the gradient constraints have an entry in the **Application Point Set** column.

6   Right-click the table gradient constraint `MainSOI_wPilot` and click **Select Application Point Set** to see how these are set up.
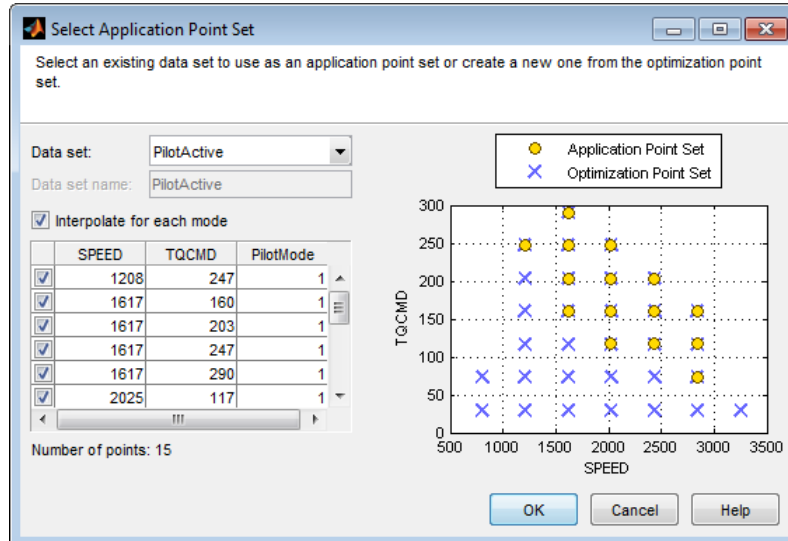
7   Observe the **Interpolate for each mode** option is selected. This application point set restricts the table gradient constraint to `PilotMode` 1 (active) points only. You can create an application point set like this by selecting `New subset` and then

choosing a subset of the optimization points by clicking in the plot or table. The application points here correspond to the operating points where the point optimization determined that the pilot mode should be active (`PilotMode` = 1). For comparison, here is the results contour plot for the point optimization results.
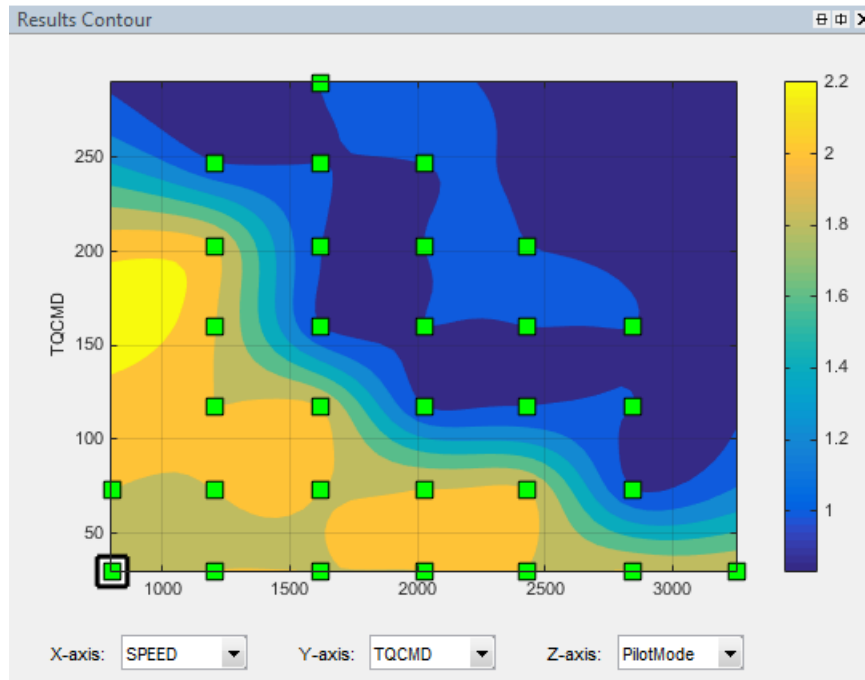
8 Compare the results contour plot with the application points for the next table gradient constraint, `MainSOI_noPilot`. These application points restrict the table gradient constraint to `PilotMode` 2 points only, where the pilot is inactive.
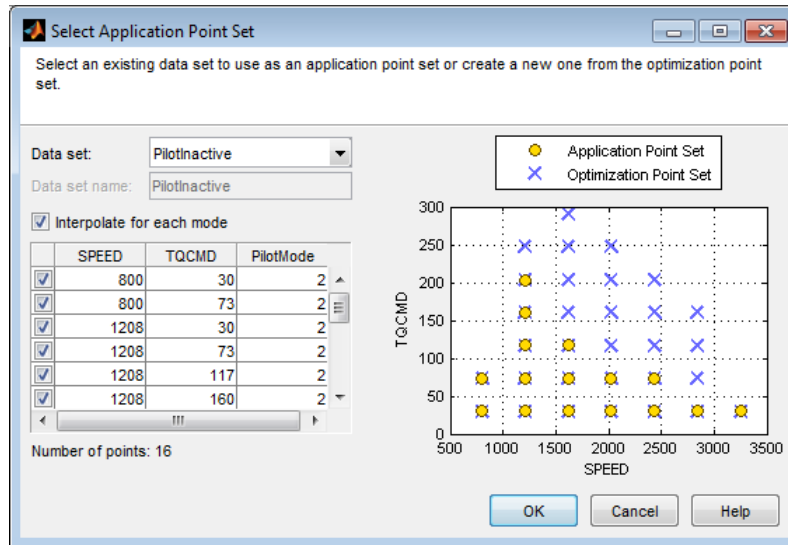
Your sum optimization now contains all the required constraints and is ready to run. Next, view the results in the example sum optimization.

Fill Tables from Optimization Results

CAGE remembers table filling settings. To view how the example tables are filled:

1   Expand the example sum optimization node BSFC_SumOpt and select the Output node.

2   Select **Solution > Fill Tables** (or use the toolbar button) to open the Table Filling from Optimization Results Wizard.

3   On the first screen, observe all the tables in the **CAGE tables to be filled list**. Click **Next**.

4   On the second screen, observe all the tables are matched up with optimization results to fill them with. Click **Next**.
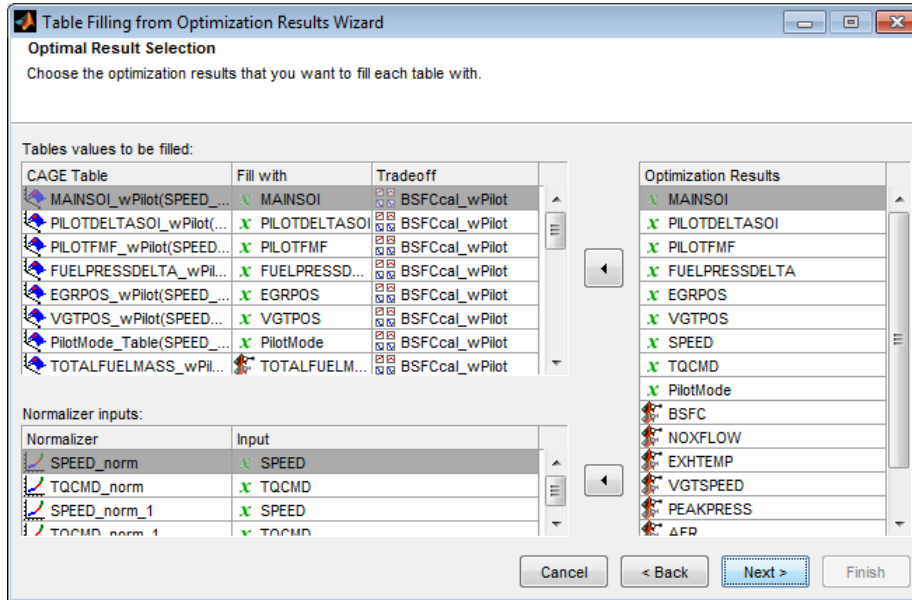
5   On the third screen, observe how some tables have a **Filter Rule** set up so that they are filled only with results where `PilotMode` is 1 or 2. You can create a filter rule like this by entering `PilotMode==1` in the **Filter Rule** column.
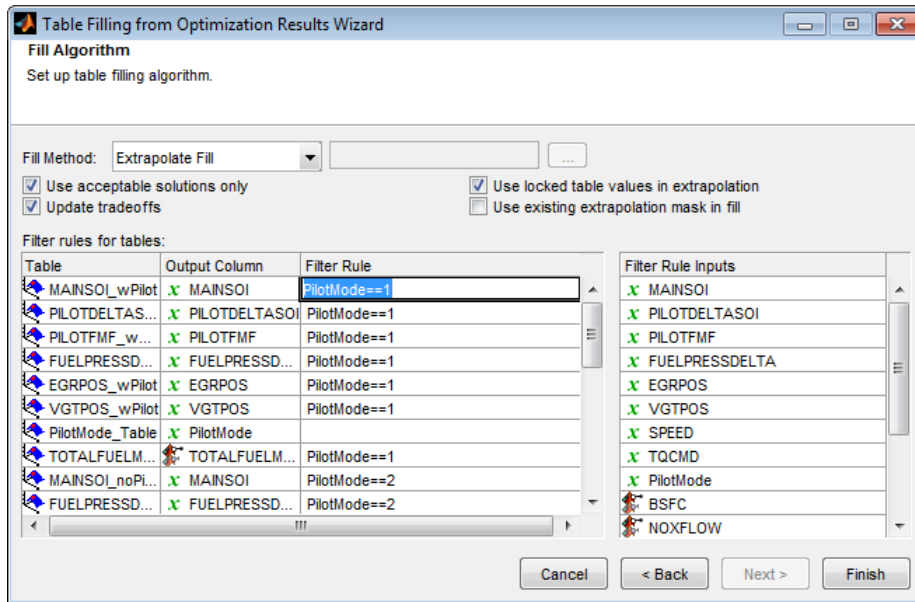
You can either click **Finish** to fill all the tables, or **Cancel** to leave the tables untouched. The example tables are already filled with these settings.

The following plots show the calibration results.
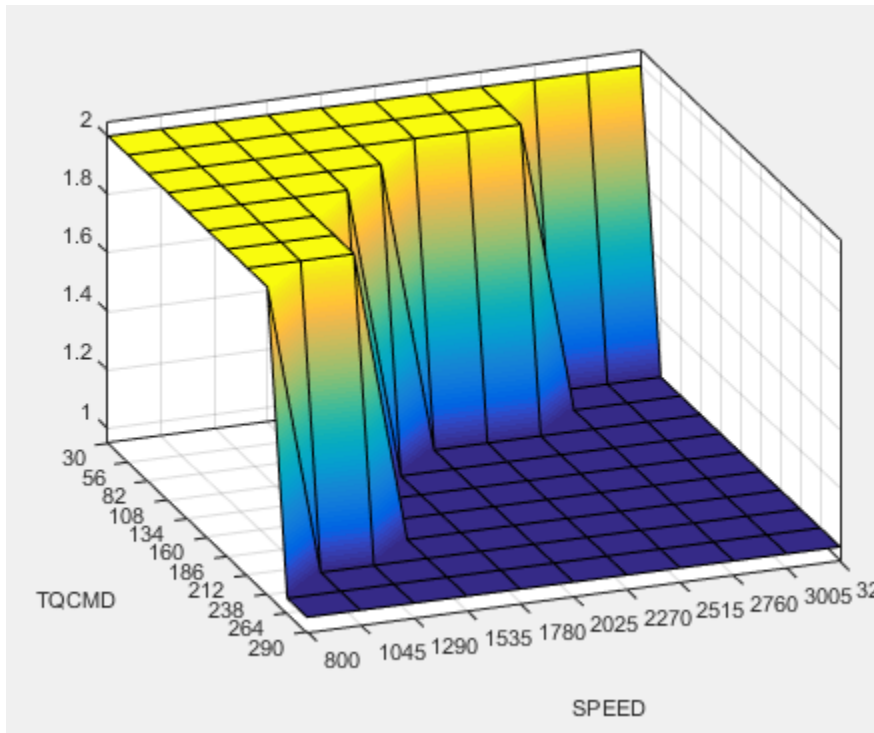
Pilot Mode Table

This graphic shows the plot of the table to select the active or inactive pilot mode depending on the speed and commanded torque

You need to fill calibration tables for each control variable described in "Multi-Injection Diesel Problem Definition" on page 6-2, in both pilot modes, active and inactive.

Following are all the pilot active tables.
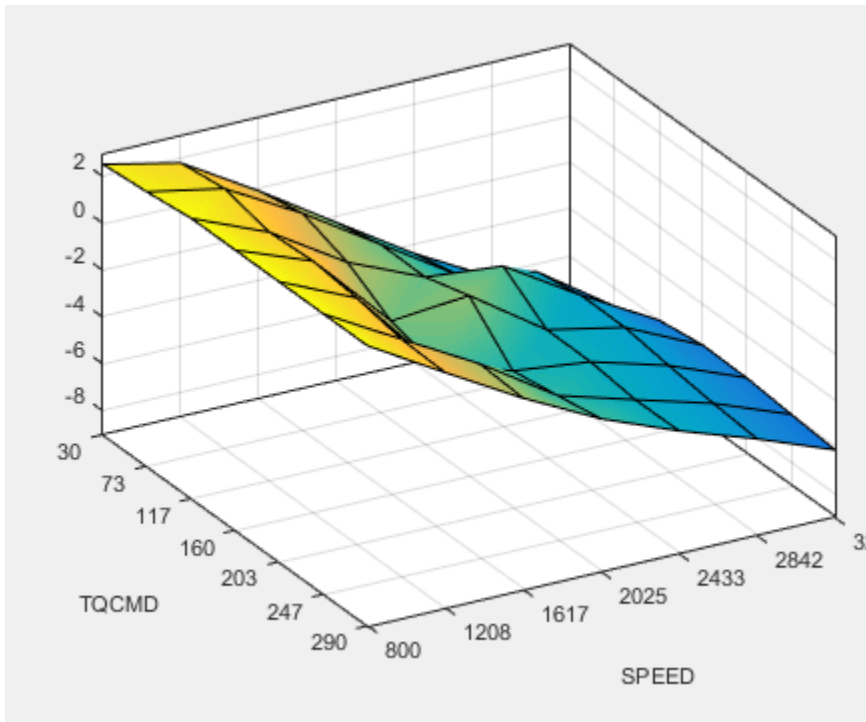
Main Injection Timing (SOI) Table
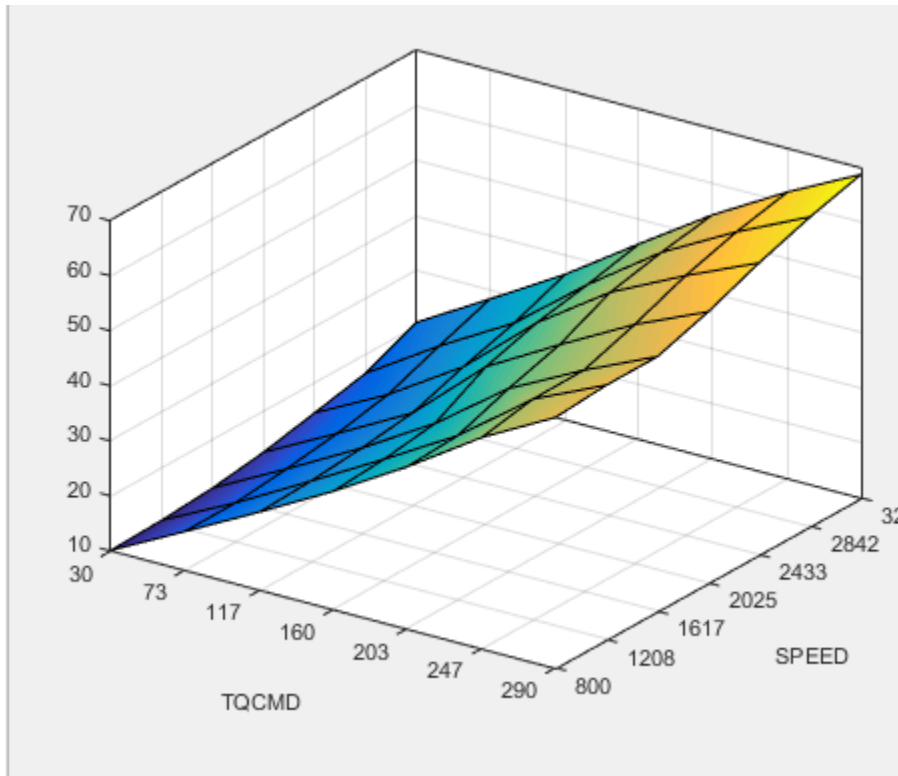
Total Injected Fuel Mass Table
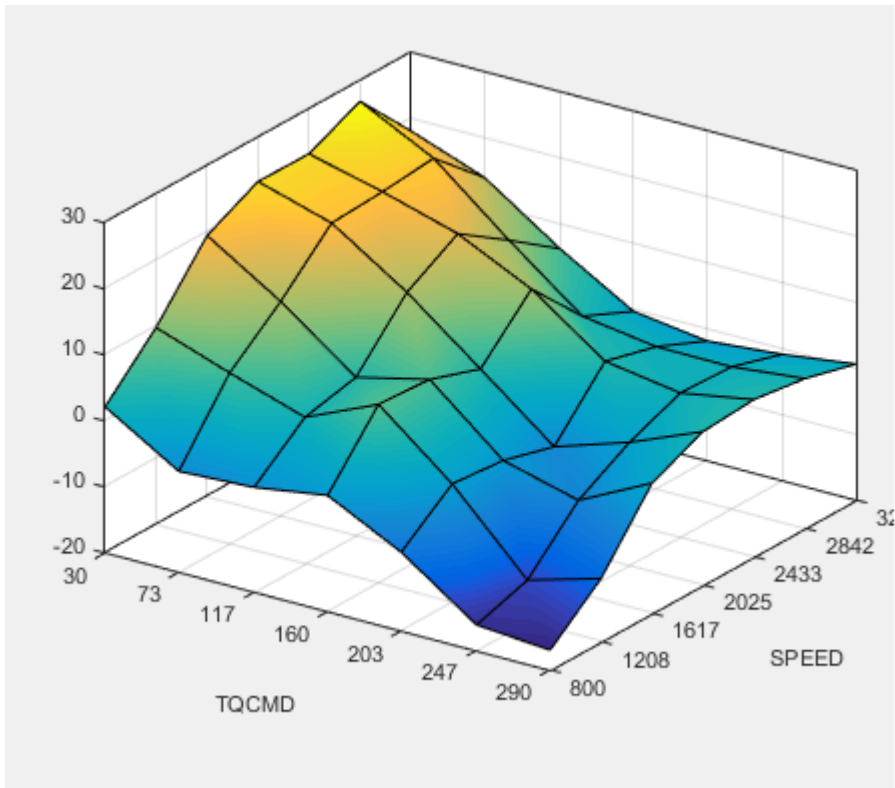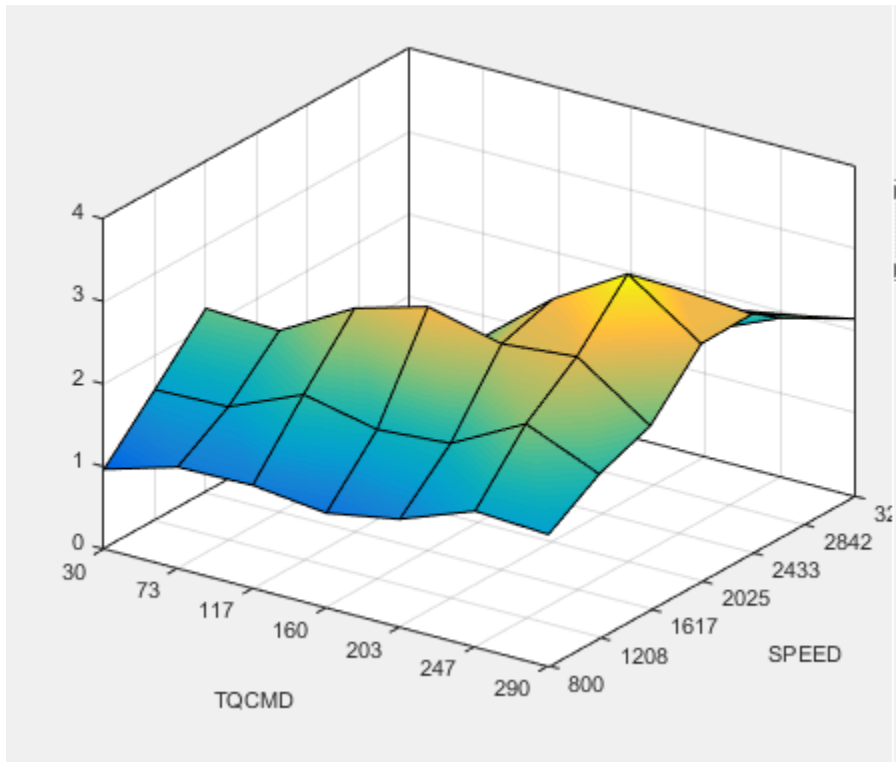
Fuel Pressure Delta Table

Exhaust Gas Recirculation (EGR) Valve Position Table
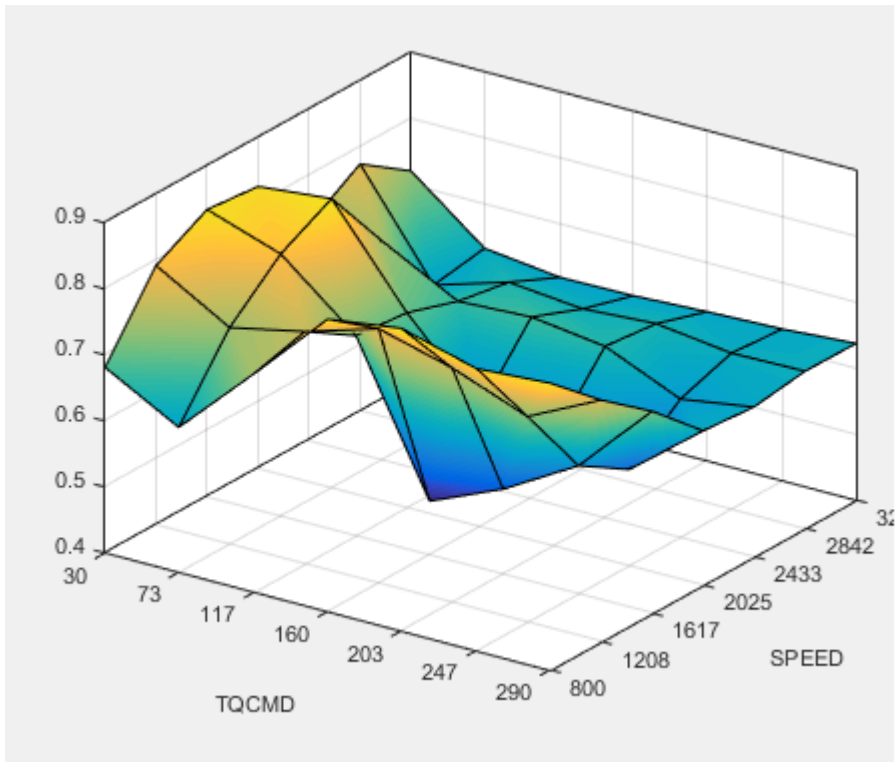
Variable-Geometry Turbo (VGT) Position Table
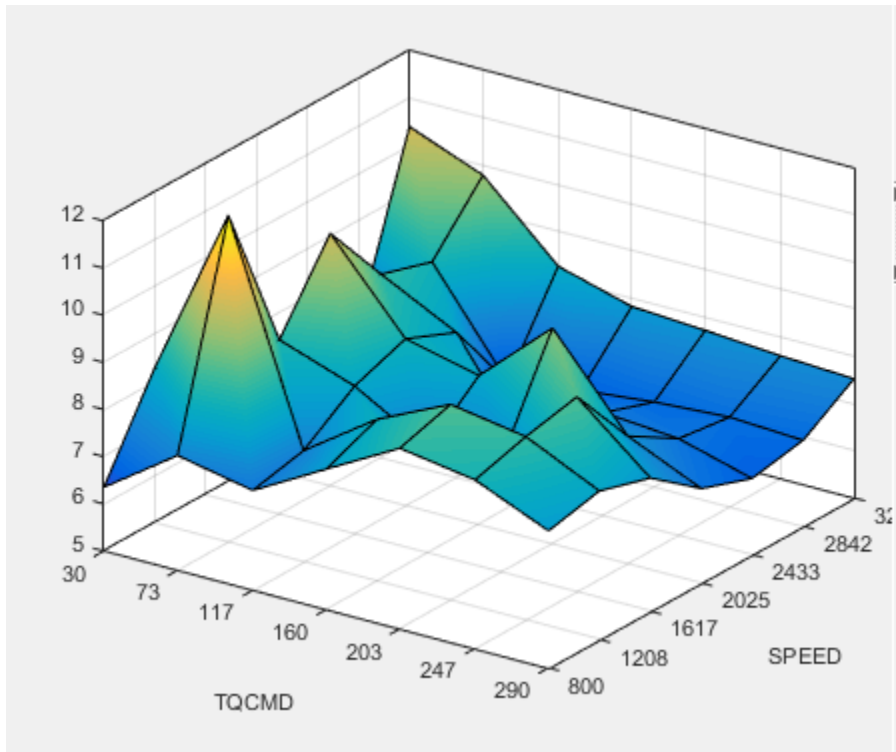
Pilot Injection Timing (Pilot SOI Delta) Table
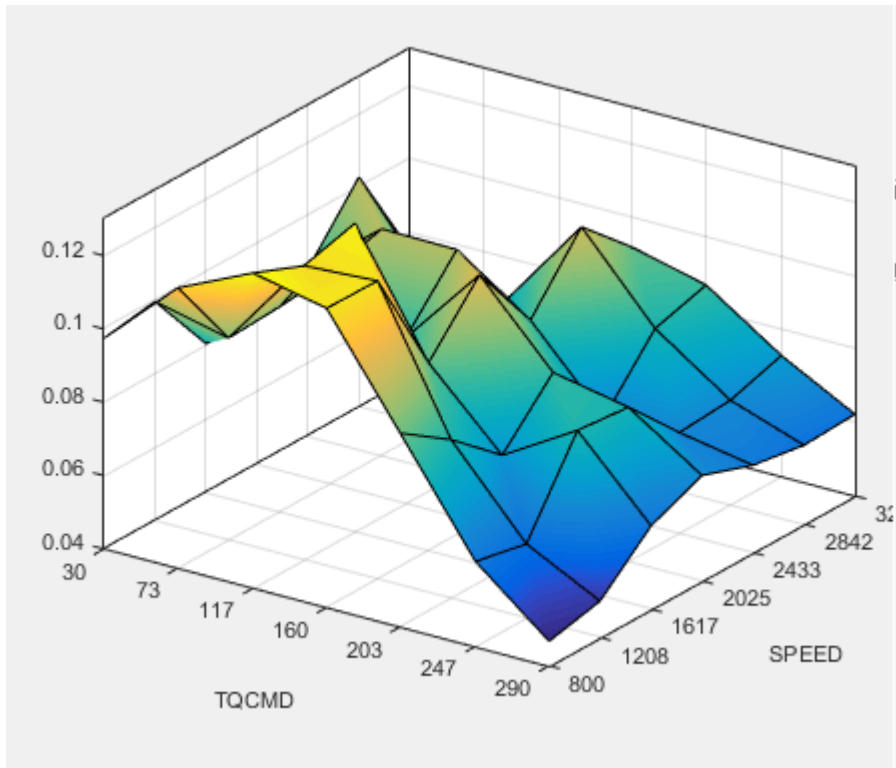
Pilot Fuel Mass Fraction Table

Examine the Multiobjective Optimization

The example file CI_MultiInject.cag also shows an example multiobjective optimization. This can useful for calibrations where you want to minimize more than one objective at a time, in this case, BSFC and NOX. The multiobjective optimization uses the gamultiobj algorithm.

1   Select the BSFC_NOX node to see how the optimization is set up. Observe the 2 objectives and the optimization information: Multiobjective optimization using a genetic algorithm.

2   Expand the BSFC_NOX node and select the BSFC_NOX_Output to view the results.

3   Examine the Pareto Graphs. It can be useful to display the Solution Information view at the same time to view information about a selected solution. You might want to

select a dominated solution (orange triangle) over a pareto solution (green square) to trade off desired properties.
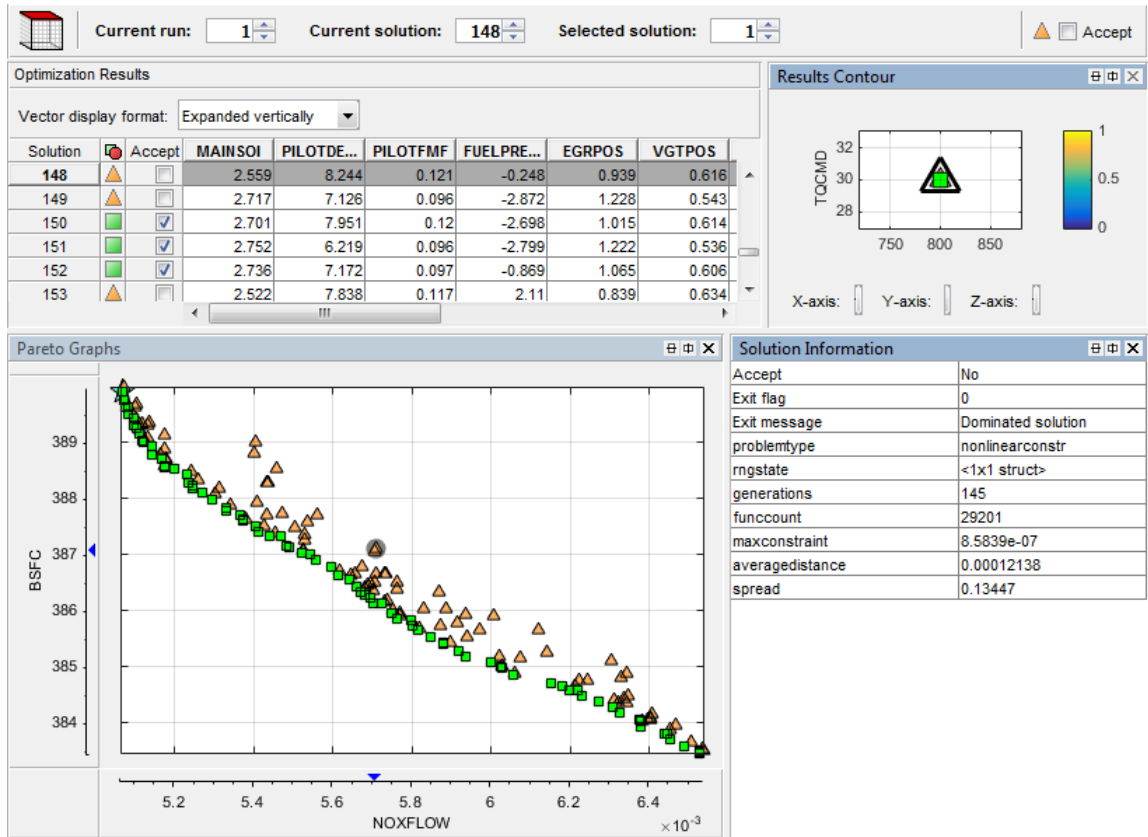
4   Select the Sum_BSFC_NOX node to see how the sum optimization is set up. The sum optimization was created from the point optimization results. Observe the 2 objectives and all the constraints.

5   Expand the Sum_BSFC_NOX node and select the Sum_BSFC_NOX_Output to view the results.

Tip Learn how MathWorks Consulting helps customers develop engine calibrations that optimally balance engine performance, fuel economy, and emissions requirements: see Optimal Engine Calibration.

Model Quickstart

This section discusses the following topics:

Empirical Engine Modeling

MBC Model Fitting provides tools for building, evaluating and comparing statistical models fitted to engine data.

- To get started with empirical engine modeling, follow the steps in this workflow: "Fit a One-Stage Model".
- To view an example project with engine data and finished models, see "Gasoline Engine Calibration".

To get started with two-stage modeling, follow the steps in this example: "Two-Stage Modeling Example" on page 7-3.

See Also

Related Examples
- "Fit a One-Stage Model"
- "Gasoline Engine Calibration"
- "Multi-Injection Diesel Calibration"

Two-Stage Modeling Example

About the Two-Stage Modeling Example

This two-stage modeling example shows you how to use MBC Model Fitting to create a statistical model of an automobile engine that predicts the torque generated by the engine as a function of spark angle and other variables. The toolbox provides example engine data. The instructions show you how to load data, fit a statistical model to the data, examine and verify the fit, and export the model.

In the normal modeling process, you would create many different models for one project and compare them to find the best solution. The tutorial also provides a quick guide to fitting and comparing multiple models.

To get started with two-stage modelling, follow the steps in these sections in order:

About Two Stage Models

Following is an explanation of how two-stage models are constructed and how they differ from one-stage models.

This tutorial is a step-by-step guide to constructing a single two-stage model for modeling engine brake torque as a function of spark, engine speed, load, and air/fuel ratio. One-stage modeling fits a model to all the data in one process, without accounting for the structure of the data. When data has an obvious hierarchical structure (as here), two-stage modeling is better suited to the task.

The usual way for collecting brake torque data is to fix engine speed, load, and air/fuel ratio within each test and sweep the spark angle across a range of angles. For this experimental setup, there are two sources of variation. The first source is variation within tests when the spark angle is changed. The second source of variation is between tests

when the engine speed, load, and air/fuel ratio are changed. The variation within a test is called local, and the variation between tests, global. Two-stage modeling estimates the local and global variation separately by fitting local and global models in two stages. A local model is fitted to each test independently. The results from all the local models are used to fit global models across all the global variables. Once the global models have been estimated, they can be used to estimate the local models' coefficients for any speed, load, and air/fuel ratio. The relationship between the local and global models is shown in the following block diagram, as you will see in the Model Browser.
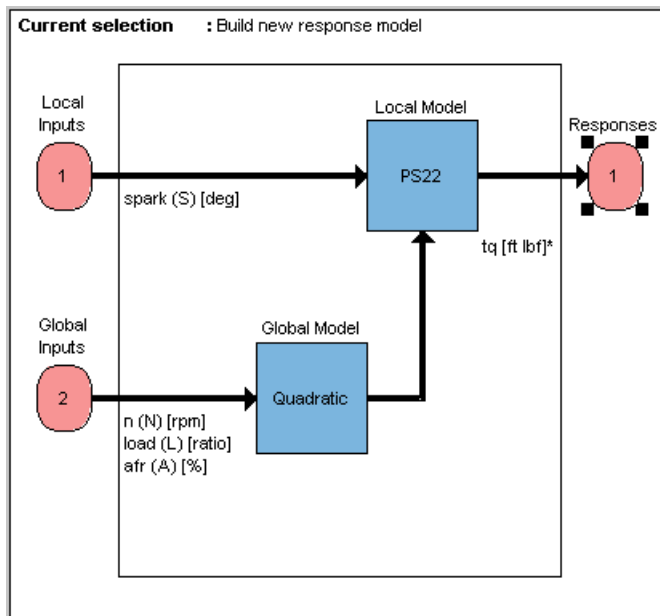
For next steps, see "Open the App and Load Data" on page 7-5.

Open the App and Load Data

1   In MATLAB, on the **Apps** tab, in the **Automotive** group, click **MBC Model Fitting**.

2   If you have never used the toolbox before, the User Information dialog appears. If you want, you can fill in any or all of the fields: your name, company, department, and contact information, or you can click **Cancel**. The user information is used to tag comments and actions so that you can track changes in your files (it does not collect information for MathWorks).

Note You can edit your user information at any time by selecting **File** > **Preferences**.

---

3   When you finish with the User Information dialog, click **OK**.

The Model Browser window appears.

In this window, the left pane, **All Models**, shows the hierarchy of the models currently built in a tree. At the start, only one node, the project, is in the tree. As you build models, they appear as child nodes of the project. The right panes change, depending on the tree node selected. You navigate to different views by selecting different nodes in the model tree.

Load the example data file `holliday.xls`:

1   From the startup project node view, in the **Common Tasks** pane, click **Import data**.

2   In the Select File to Import dialog box, browse to the file `holliday.xls` in the `mbctraining` directory. Click **Open** or double-click the file.

The Data Editor opens.

3   View plots of the data in the Data Editor by selecting variables and tests in the lists on the left side. Have a look through the data to get an idea of the shape of curve formed by plotting torque against spark.

Use the editor to prepare your data before model fitting. For more details on using the Data Editor, see "Data Manipulation for Modeling".

4   Close the Data Editor to accept the data and return to the Model Browser. Notice that the new data set appears in the **Data Sets** pane.

This data is from Holliday, T., "The Design and Analysis of Engine Mapping Experiments: A Two-Stage Approach," Ph.D. thesis, University of Birmingham, 1995.

For next steps, see "Set Up the Model" on page 7-7.

Set Up the Model

In this section...
"Specifying Model Inputs" on page 7-7
"Setting Up the Response Model" on page 7-9

Specifying Model Inputs

You can use the imported data to create a statistical model of an automobile engine that predicts the torque generated by the engine as a function of spark angle and other variables.

1 In the Model Browser project node view, in the **Common Tasks** pane, click **Fit models**.

2 In the Fit Models dialog box, observe that the `Data Object` you imported is selected in the **Data set** list.

3 Click the **Two-Stage** test plan icon in the **Template** pane.

4 In the **Inputs and Responses** pane, select data channels to use for the responses you want to model.

   The model you are building is intended to predict the torque generated by an engine as a function of spark angle at a specified operating point defined by the engine's speed, air/fuel ratio, and load. The input to the local model is therefore the spark angle, and the response is torque.

   The inputs to the global model are the variables that determine the operating point of the system being modeled. In this example, the operating point of the engine is determined by the engine's speed in revolutions per minute (rpm - often called N), load (L), and air/fuel ratio (afr).

   a Select `spark` in the **Data channels** list and click the button to add it to the **Local inputs** list.

   b Select n, `load` and `afr` in the **Data channels** list and click the button to add them to the **Global inputs** list.
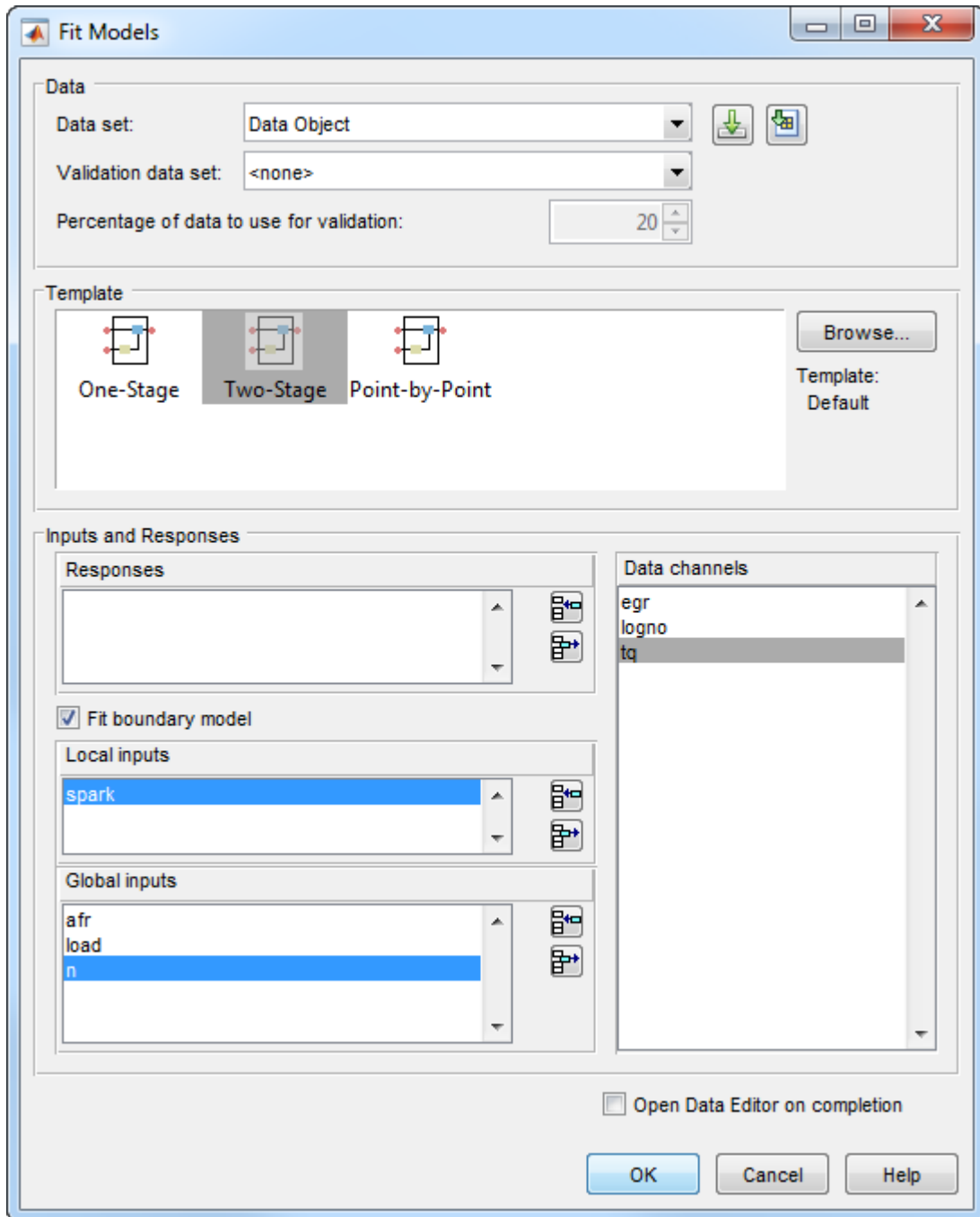
5 Leave the responses empty, and click **OK**. You will set up the response later.

The default name of the new test plan, `Two-Stage`, appears in the Model Browser tree, in the **All Models** pane.

Setting Up the Response Model

To achieve the best fit to torque/spark sweeps, you need to change the local model type from the default. The type of a local model is the shape of curve used to fit the test data, for example, quadratic, cubic, or polynomial spline curves. In this example, you use polynomial spline curves to fit the test data. A spline is a curve made up of pieces of polynomial, joined smoothly together. The points of the joins are called knots. In this case, there is only one knot. These polynomial spline curves are very useful for torque/spark models, where different curvature is required above and below the maximum.

To change from the default models and specify polynomial spline as the local model type,

1 In the Model Browser, select the test plan node `Two-Stage`, and in the **Common Tasks** pane, click **Fit models**. A dialog box asks if you want to change all the test plan models. Click **Yes**.

2 In the Fit Models Wizard, click **Next** to continue using the currently selected data.

3 The next screen shows the model inputs you already selected. Click **Next**

4 To choose the response, on the Response Models screen, select `tq` and click **Add**.

5 Edit the **Local model** type by clicking **Set Up**.

The Local Model Setup dialog box appears.

a Select `Polynomial Spline` from the **Local Model Class** list.

b Edit the **Spline Order Below knot** to 2, and leave **Above knot** set to 2.

c Click **OK** to dismiss the dialog box.

6 Select `Maximum` under **Datum**. Only certain model types with a clearly defined maximum or minimum can support datum models. See "Add Response Models and Datum Models".

7 Click **Finish**.

The Model Browser calculates local and global models using the test plan models you just set up.

Notice that the new name of the local model class, PS (for polynomial spline) 2,2 (for spline order above and below knot) now appears on a new node in the tree in the **All Models** pane, called PS22.

For next steps, see "Verify the Model" on page 7-11.

Verify the Model

In this section...

Verifying the Local Model

The first step is to check that the local models agree well with the data:

1   If necessary, select PS22 (the local node) on the Model Browser tree.

    The **Local Model** view appears, displaying the local model fitting the torque/spark data for the first test and diagnostic statistics that describe the fit. The display is flexible in that you can drag, open, and close the divider bars separating the regions of the screen to adjust the view.

2   View local model plots and statistics. The Sweep Plot shows the data being fitted by the model (blue dots) and the model itself (line). The red spot shows the position of the polynomial spline knot, at the datum (maximum) point.

3   Look for problem tests with the RMSE Plots. The plot shows the standard errors of all the tests, both overall and by response feature. Navigate to a test of interest by double-clicking a point in the plot to select the test in the other plots in the local model view.

4   In the Diagnostic Statistics plot pane, click the **Y-axis factor** pop-up menu and select Studentized residuals.

5   Scroll through local models test by test using the **Test** arrows at the top left, or by using the **Select Test** button.

6   Select Test 588. You see a data point outlined in red. This point has automatically been flagged as an outlier.

7   Right-click the plot and select **Remove Outliers**. Observe that the model is refitted without the outlier.

Both plots have right-click pop-up menus offering various options such as removing and restoring outliers and confidence intervals. Clicking any data point marks it in red as an outlier.

You can use the **Test Notes** pane to record information on particular tests. Each test has its own notes pane. The test numbers of data points with notes recorded against them are colored in the global model plots, and you can choose the color using the **Test Number Color** button in the **Test Notes** pane. Quickly locate tests with notes by clicking **Select Test**.

To learn more about the plots and statistics, see "Assess Local Models".

Verifying the Global Model

The next step is to check through the global models to see how well they fit the data:

1   Expand the PS22 local node on the Model Browser tree by clicking the plus sign (+) to the left of the icon. Under this node are four response features of the local model. Each of these is a feature of the local model of the response, which is torque.

2   Select the first of the global models, knot.

    You see a dialog box asking if you want to update fits, because you removed an outlier at the local level. Click **Yes**.

    The **Response Feature** view appears, showing the fit of the global model to the data for knot. Fitting the local model is the process of finding values for these coefficients or *response features*. The local models produce a value of knot for each test. These values are the data for the global model for knot. The data for each response feature come from the fit of the local model to each test.

    Use the plots to assess model fits.

3   Select the response feature Bhigh_2. One outlier is marked. Points with an absolute studentized residual value of more than 3 are automatically suggested as outliers (but included in the model unless you take action). You can use the right-click menu to remove suggested outliers (or any others you select) in the same way as from the Local Model plots. Leave this one. If you zoom in on the plot (**Shift**-click-drag or middle-click-drag) you can see the value of the studentized residual of this point more clearly. Double-click to return to the previous view.

> **Note** Never remove outliers as a matter of course. However, this tutorial shows you how the toolbox helps you to do this when required. The default outlier selection criterion is a studentized residual greater than 3, to bring your attention to possible outliers, but you should never remove data without good reasons. Remove enough points and the model will simply interpolate the data and become useless for prediction. You can customize the criteria for outlier selection.

4 Select the other response features in turn: `max` and `Blow 2`. You will see that `Blow 2` has a suggested outlier with a very large studentized residual; it is a good distance away from all the other data points for this response feature. All the other points are so clustered that removing this one could greatly improve the fit of the model to the remaining points, so remove it.

To learn more about the plots and statistics, see "Assess One-Stage Models".

Creating the Two-Stage Model

Recall how two-stage models are constructed: two-stage modeling partitions the variation separately between tests and within tests, by fitting local and global models separately. A model is fitted to each test independently (local models). These local models are used to generate global models that are fitted across all tests.

For each sweep (test) of spark against torque, you fit a local model. The local model in this case is a spline curve, which has the fitted response features of `knot`, `max`, `Bhigh_2`, and `Blow_2`. The result of fitting a local model is a value for `knot` (and the other coefficients) for each test. The global model for `knot` is fitted to these values (that is, the `knot` global model fits `knot` as a function of the global variables). The values of knot from the global model (along with the other global models) are then used to construct the two-stage model

The global models are used to reconstruct a model for the local response (in this case, torque) that spans all input factors. This is the two-stage model across the whole global space, derived from the global models.

After you are satisfied with the fit of the local and global models, it is time to construct a two-stage model from them.

1 Return to the Local Model view by clicking the local node `PS22` in the Model Browser tree.

2 To create a two-stage model, click **Create Two-Stage** in the Common Tasks pane.

Comparing the Local Model and the Two-Stage Model

1   Now the plots in the **Local Model** view show two lines fitted to the test data. Scroll though the tests using the left/right arrows or the **Select Test** button at the top left. The plots now show the fit of the two-stage model for each test (green circles and line), compared with the fit of the local model (blue line) and the data (blue dots). Zoom in on points of interest by **Shift**-click-dragging or middle-click-dragging. Double-click to return the plot to the original size.

Compare how close the two-stage model fit is to both the data and the local fit for each test.

2

Notice that the local model icon has changed (from the local   icon showing a

house, to a two-stage icon   showing a house and a globe) to indicate that a two-stage model has been calculated.

Response Node

Click the Response node (`tq`) in the Model Browser tree.

Now at the Response node in the Model Browser tree (`tq`), which was previously blank, you see plots showing you the fit of the two-stage model to the data. You can scroll through the tests, using the arrows at top left, to view the two-stage model against the data for groups of tests.

You have now completed setting up and verifying a two-stage model.

For next steps, see "Export the Model" on page 7-15 or "Create Multiple Models to Compare" on page 7-16.

Export the Model

All models created in the Model Browser are exported using the **File** menu. A model can be exported to the MATLAB workspace, to CAGE, or to a Simulink model.

1   Click the `tq` node in the model tree.
2   Choose **File > Export Models**. The Export Model dialog box appears.
3   For the **Export to** parameter, select the `Simulink`, `Workspace`, or `CAGE`.
4   Click **OK** to export the models.

    To import models into CAGE to create calibrations, use the CAGE Import Tool instead for more flexibility.

For next steps, see "Create Multiple Models to Compare" on page 7-16.

Create Multiple Models to Compare

In this section...
"Methods For Creating More Models" on page 7-16
"Creating New Local Models" on page 7-16
"Adding New Response Features" on page 7-18
"Comparing Models" on page 7-20
"Creating New Global Models" on page 7-21

Methods For Creating More Models

After you have fitted and examined a single model, you normally want to create more models to search for the best fit. You can create individual new models or use the **Create Alternatives** common task to create a selection of models at once, or create a template to save a variety of model settings for reuse.

To use the Model Template dialog box to quickly create a selection of different child nodes to compare, click **Create Alternatives** in the Common Tasks pane. The following exercises show you examples of these processes.

Note You need to complete the previous tutorial sections to have a complete two-stage model as a starting point. See "Empirical Engine Modeling" on page 7-2.

Creating New Local Models

To follow these examples, you need to create the initial models by following the previous sections in "Empirical Engine Modeling" on page 7-2.

1   As an example, select the `tq` response node and click **New Local Model** in the Common Tasks pane.

   The Local Model Setup dialog appears.

2   Select a `Polynomial Spline`, and edit the spline order to 3 below the knot and 2 above. Click **OK**.

   A new set of local models (and associated response feature models) is calculated.

3 Return to the parent `tq` response node , and click **New Local Model** again, in the Common Tasks pane.

4 Select a `Polynomial` with an order of `2` in the Local Model Setup dialog. Click **OK**.

A new set of local models and response feature models is calculated.

Now you have three alternative local models to compare: two polynomial splines (order 3,2 and order 2,2) and a polynomial (order 2), as shown.
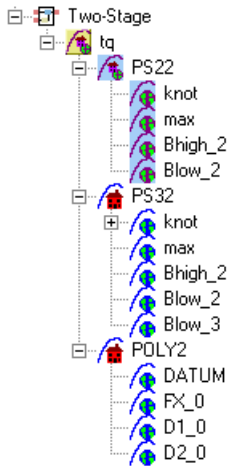
You can select the alternative local models in turn and compare their statistics. For an example, follow these steps:

1 Select the new local model node `PS32`.

2 Select test `587` in the **Test** edit box.

3 In the **Local statistics** pane, observe the value of RMSE (root mean squared error) for the current (i^{th}) test.

The RMSE value is our basic measure of how closely a model fits some data, which measures the average mismatch between each data point and the model. This is why you should look at the RMSE values as your first tool to inspect the quality of the fit — high RMSE values can indicate problems.

4 Now select the local model node `POLY2` and see how the value of `RMSE` changes.

Observe that the shape of the torque/spark sweep for this test is better suited to a polynomial spline model than a polynomial model. The curve is not symmetrical

because curvature differs above and below the maximum (marked by the red cross at the datum). This explains why the value of RMSE is much lower for PS32 (the polynomial spline) than for the POLY2 (polynomial) for this test. The polynomial spline is a better fit for the current test.

5   Look through some other tests and compare the values of RMSE for the different local models. To choose the most suitable local model you must decide which fits the majority of tests better, as there are likely to be differences among best fit for different tests.

6   To help you quickly identify which local models have the highest RMSE, indicating problems with the model fit, check the RMSE Plots.

   a   Use the plot to help you identify problem tests. Use the drop-down menus to change the display. For example, select s_knot to investigate the error values for knot (MBT), or RMSE to look at overall error.

   b   You can navigate to a test of interest from the RMSE Plots by double-clicking a point in the plot to select the test in the other plots.

7   Look at the value of Local RMSE reported in the **Pooled Statistics** pane on the right (this is pooled between all tests). Now switch between the POLY2 and the PS32 local models again and observe how this value changes.

8   You can compare these values directly by selecting the parent tq response node, when the Local RMSE is reported for each child local model in the list at the bottom.

   When all child models have a two-stage model calculated, you can also compare two-stage values of RMSE here. Remember, you can see statistics to compare the list of child models of the response node in this bottom list pane.

When comparing models, look for lower RMSE values to indicate better fits. However, remember that a model that interpolates between all the points can have an RMSE of zero but be useless for predicting between points. Always use the graphical displays to visually examine model fits and beware of "overfitting" — chasing points at the expense of prediction quality. You will return to the problem of overfitting in a later section when you have two-stage models to compare.

Adding New Response Features

Recall that two-stage models are made up of local models and global models. The global models are fitted to the response features of the local models. The response features available are specific to the type of local model. You can add different response features to see which combination of response features makes the best two-stage model as follows:

1 Select the local model node PS32.

2 Select **File > New Response Feature**.

A dialog appears with a list of available response features.

3 Select f(x+datum) from the list and enter -10 in the **Value** edit box. Click **OK**.

A new response feature called FX_less10 is added under the PS32 local model. Recall that the datum marks the maximum, in this case maximum torque. The spark angle at maximum torque is referred to as maximum brake torque (MBT). You have defined this response feature (f(x+datum)) to measure the value of the model (torque) at (-10 + MBT) for each test. It can be useful to use a response feature like this to track a value such as maximum brake torque (MBT) minus 10 degrees of spark angle. This response feature is not an abstract property of a curve, so engineering knowledge can then be applied to increase confidence in the models.

4 Select the local node PS32, and click **Create Two-Stage** in the Common Tasks pane. The Model Selection window opens, because you now need to choose 5 of the 6 response features to form the two-stage model.

In the Model Selection window, observe four possible two-stage models in the Model List. This is because you added a sixth response feature. Only five (which must include knot) are required for the two-stage model, so you can see the combinations available and compare them. Note that not all combinations of five response features can completely describe the shape of the curve for the two-stage model, so only the possible alternatives are shown.

5 Close the Model Selection window and click **Yes** to accept one of the models as best.

Notice that the response features chosen to calculate the two-stage model are highlighted in blue, and the unused response feature is not highlighted.

6 Select the tq response node to see a comparison of the statistics of both two-stage models (your original PS22 and the new PS32).

Remember that the POLY2 local model has no two-stage model yet; no two-stage statistics are reported for POLY2 in the bottom list pane. You cannot fully compare the two-stage models until every local model in the test plan has a two-stage model calculated.

7 To calculate the two-stage model for POLY2, in the Common Tasks pane, click **Create Two-Stage**.

Comparing Models

1 Now you have three two-stage models. Select the `tq` response node and look at the statistics, particularly Local RMSE and Two-Stage RMSE reported in the list of child models at the bottom.

- Look for lower RMSE values to indicate better fits.

- Look for lower PRESS RMSE values to indicate better fits without overfitting. PRESS RMSE is a measure of the predictive power of your models.

  It is useful to compare PRESS RMSE with RMSE as this may indicate problems with overfitting. RMSE is minimized when the model gets close to each data point; "chasing" the data will therefore improve RMSE. However, chasing the data can sometimes lead to strong oscillations in the model between the data points; this behavior can give good values of RMSE but is not representative of the data and will not give reliable prediction values where you do not already have data. The PRESS RMSE statistic guards against this by testing how well the current model would predict each of the points in the data set (in turn) if they were not included in the regression. To get a small PRESS RMSE usually indicates that the model is not overly sensitive to any single data point.

  If the value of PRESS RMSE is much bigger than the RMSE, then you are overfitting - the model is unnecessarily complex. For a fuller description of the meaning of overfitting, and how RMSE and PRESS can help you select good models, see "Choose the Best Model"."Choose the Best Model". As a rule of thumb, if you have about 100 data points, you should aim for a PRESS RMSE no more than 5% larger than the RMSE (remember here you have only 27 tests).

- Look for lower T^2 values. A large T^2 value indicates that there is a problem with the response feature models.

- Look for large negative log likelihood values to indicate better fits.

  See "Pooled Statistics" for more on T^2 and log likelihood.

2 To compare all three two-stage models simultaneously, select **Model > Selection Window**. Here you can see the same statistics to compare the models in the bottom list, but you can also make use of a variety of views to look for the best fit:

- You can plot the models simultaneously on the Tests, Residuals and Cross Section views (**Shift**- or **Ctrl**-click to select models in the list)

- You can view each model in the Response Surface view as a surface; movie, contour or multiline plot, and as a table

3   You can select a model and click **Assign Best** in the Model Selection window, or double-click a model to assign it as best.

4   When you close the Model Selection window and return to the Model Browser, the model you selected as best is copied to the parent response node, `tq`.

Creating New Global Models

In this example, you have not yet searched for the best global model types. You would normally do this before creating and comparing two-stage models. For the purpose of this tutorial, you have already created two-stage models and used RMSE to help you identify better models. The principle is the same at each level in the model tree: add new child models and choose the best. You can create any number of child nodes to search for the best global model fit for each response feature in your tree.

1   Select the local node `POLY2`.

2   To create a selection of alternatives for each response feature node, in the Common Tasks pane, click **Build Global Models**.

3   In the Model Template dialog box, click **New**, then click **OK**.

4   Observe the default list of a variety of model types, then click **OK**. It is worth trying the default model settings for a quick exploration of the trends in the data.

5   In the Model Selection dialog box, leave the default selection criterion for automatically choosing the best child node, and click **OK**.

     The toolbox builds the models and selects the best using your selection criteria.

Note The toolbox automatically builds models in parallel if you have Parallel Computing Toolbox.

---

6   Assess all the fits in the Alternative Models list in case you want to choose an alternative as a better fit.

7   Notice that the child node model assigned as best is highlighted in blue in the Alternative Models list and the model tree. The local node has changed from the two-stage icon back to the local model icon (a red house). This is because you have changed the response feature models, and so you need to recalculate the two-stage model using the new global models for the response features.

     When you have chosen best global models for all your response features, you need to recalculate the two-stage model.

8 When you have chosen a best model among alternatives, it can be useful to clean up the rejected models by selecting **Delete Alternatives** in the Common Tasks pane. You can also select **File > Clean Up Tree**. This deletes all rejected child models where best models have been chosen; only the child nodes selected as best remain.

You can use the Model Template dialog box to create and save templates of model types you often want to build. Creating a template containing a list of all the models you want is a very efficient way to quickly build a selection of alternative model child nodes for many global models. Use these techniques to find models well suited to the data for each of your global models. See "Create Alternative Models to Compare".

See Also

Related Examples
- "Fit a One-Stage Model"
- "Gasoline Engine Calibration"
- "Multi-Injection Diesel Calibration"

Generate Current Controller Calibration Tables for Flux-Based Motor Controllers

Using the Model-Based Calibration Toolbox, you can generate optimized calibration tables for flux-based motor controllers. This example shows how to import data, fit a model, and optimize the data based on objectives and constraints.

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

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

The calibration tables are d- and q- axis reference currents as functions of motor torque and motor speed.

To generate optimized current calibration tables, follow these workflow steps.

Workflow Steps	Description
"Collect and Post Process Motor Data" on page 7-24	Collect the nonlinear motor flux data from dynamometer testing or finite element analysis (FEA). For this example, file `ex_motor_data.xlsx` contains the data that you need: • d-axis current, I_d, in A • q-axis current, I_q, in A • Motor speed, n, in rpm • d-axis voltage, V_d, in V • q-axis voltage, V_q, in V • Electromagnetic motor torque, T_e, in N.m

Workflow Steps	Description
"Model Motor Data" on page 7-25	Use a one-stage model to fit the ex_motor_data.xlsx data. Specifically: • Import data • Filter data • Fit model
"Generate Calibration" on page 7-30	Calibrate and optimize the data using objectives and constraints. Specifically: • 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.

Collect and Post Process Motor Data

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

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

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

$$v_d = R_s i_d - \omega_e \lambda_q$$
$$v_q = R_s i_q + \omega_e \lambda_d$$
$$n = \frac{60\omega_e}{2\pi P}$$

The equations use these variables:

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

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

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

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

Model Motor Data

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

Import Data

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

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

- q-axis voltage, V_q, in V
- Electromagnetic motor torque, T_e, in N.m

1 In MATLAB, on the **Apps** tab, in the **Automotive** group, click **MBC Model Fitting**.

2 In the Model Browser home page, click **Import Data**. Click **OK** to open a data source file.

3 Navigate to the `matlab\toolbox\mbc\mbctraining` folder. Open data file `ex_motor_data.xlsx`. The Data Editor opens with your data.
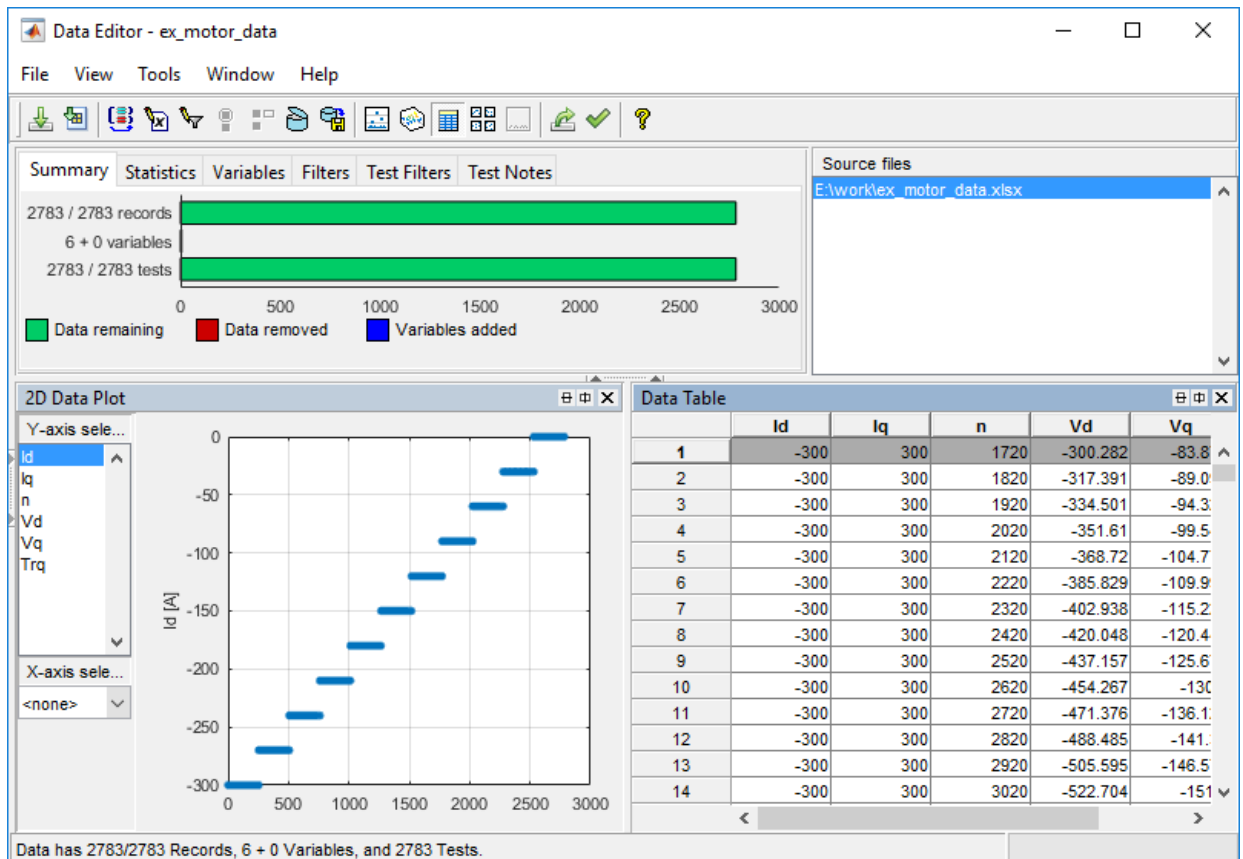
Filter Data

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

- Voltage magnitude, V_s, less than or equal to 300 V.

- Current magnitude, I_s, less than or equal to 350 A.

1   In the Data Editor, select **Tools** > **Variables** to open the **Variable Editor**. Create these variables. Add units.

- `Is = sqrt (Id.^2 + Iq.^2)`
- `Vs = sqrt (Vd.^2 + Vq.^2)`

Summary	Statistics	Variables (2)	Filters (2)	Test Filters	Test Notes
Variable Expression				**Units**	**Results**
✅ Is = sqrt (Id.^2 + Iq.^2)				A	Variable successfully added.
✅ Vs = sqrt (Vd.^2 + Vq.^2)				V	Variable successfully added.

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

- `Is <= 350`
- `Vs <= 300`

Summary	Statistics	Variables (2)	Filters (2)	Test Filters	Test Notes
Filter Expression				**Results**	
✅ Is <=350				Filter successfully applied : 230 records excluded.	
✅ Vs <= 300				Filter successfully applied : 2100 records excluded.	

Fit Model

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

- Responses

  - *d*-axis voltage, V_d, in V

- q-axis voltage, V_q, in V
- Inputs

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

1 In the Model Browser, select **Fit Models**.

2 In **Fit Models**, configure a One-Stage model with these responses and inputs.

Responses	Inputs
Vd	Id
Vq	Iq
	Trq
	n
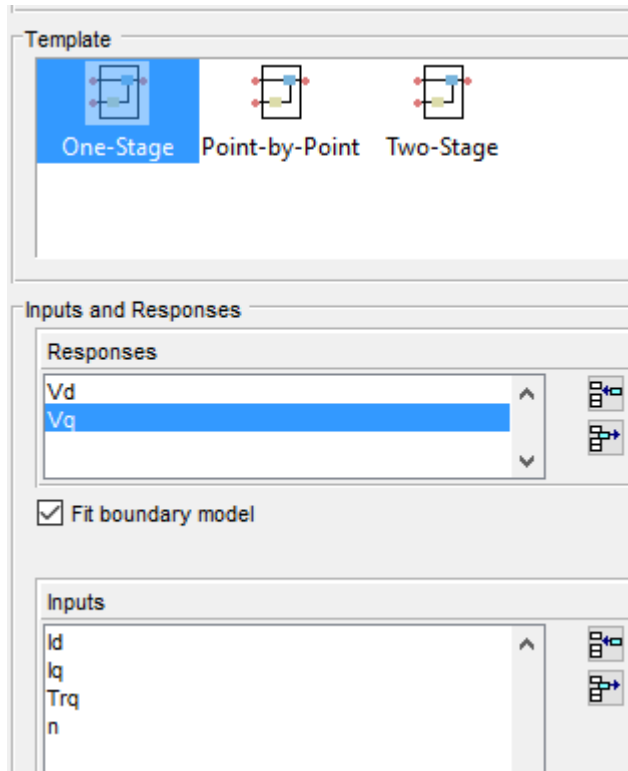
3. To fit the model, select **OK**. If prompted, accept changes to data. By default, the fit uses a Gaussian Process Model (GPM) to fit the data.

4. After the fit completes, examine the response models for V_d and V_q. The Model Browser displays information that you can use to determine the accuracy of the model fit.

   - In the Model Browser, select Vd. Examine the response surface and diagnostic statistics. In this example, the response surface indicates that Vd increases as Id approaches 0. The diagnostics indicate that the response residuals are mostly within ±1 V. These results indicate a reasonably accurate fit.
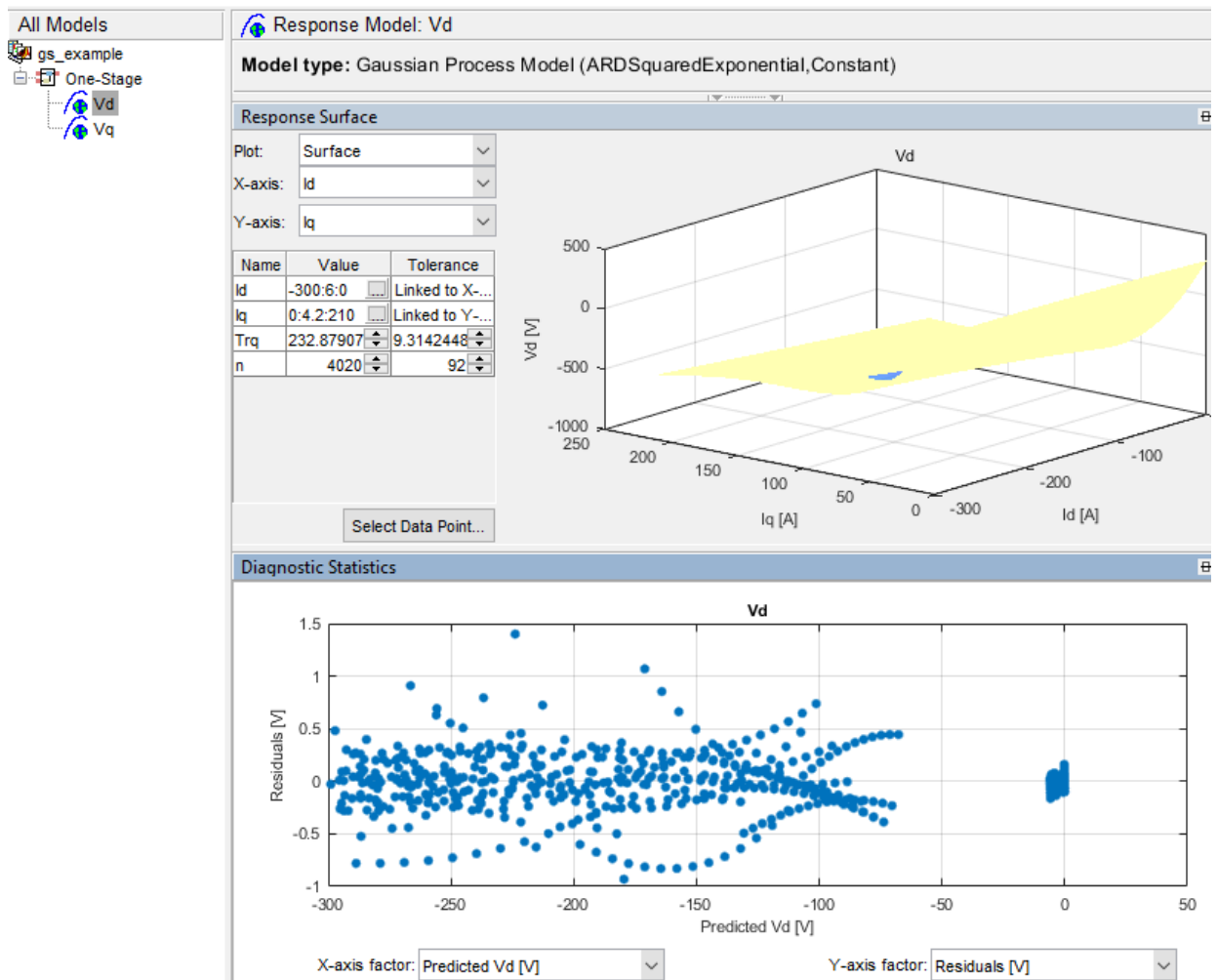
5 Save your project. For example, select **Files > Save Project**. Save gs_example.mat to work folder.

Generate Calibration

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

Create Functions

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

- Voltage magnitude, V_s
- Current magnitude, I_s
- Torque per amp, *TPA*

1   In MATLAB, on the **Apps** tab, in the **Automotive** group, click **MBC Optimization**.
2   In the Cage Browser, select **Import Models**. If it is not already opened, in the MBC Model Fitting browser, open the gs_example.mat project.

3   In Cage Import tool, select **Vd** and **Vq**. Click **Import Selected Items**.
4   In Import Models, click **OK**. Close the CAGE Import Tool.
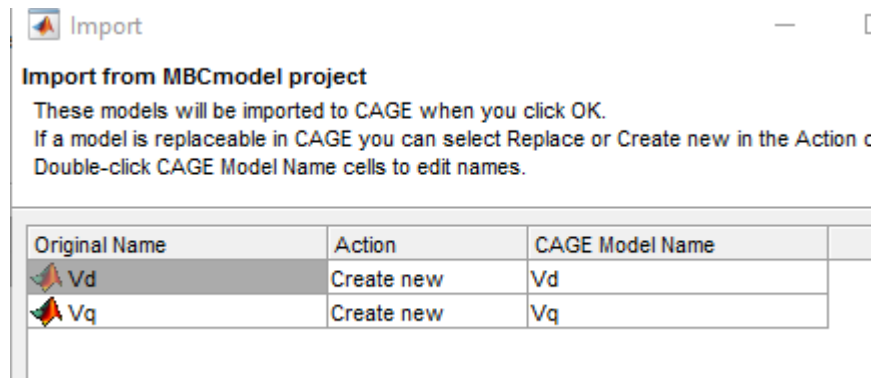
5 In the Cage Browser toolbar, use **New Function Model** wizard to create these functions:

- Is = sqrt (Id.^2 + Iq.^2)
- Vs = sqrt (Vd.^2 + Vq.^2)
- TPA = Trq./Is
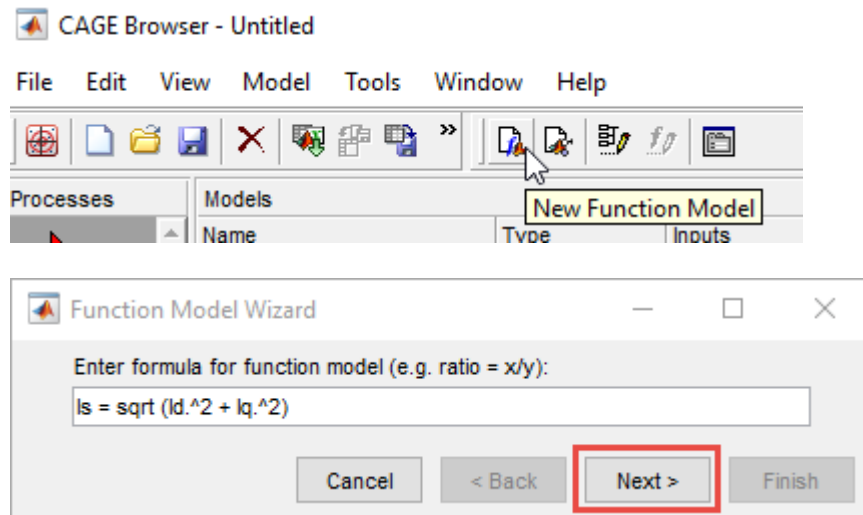
6   In the Cage Browser, verify that the function models for `Is`, `Vs`, and `TPA` have these descriptions.

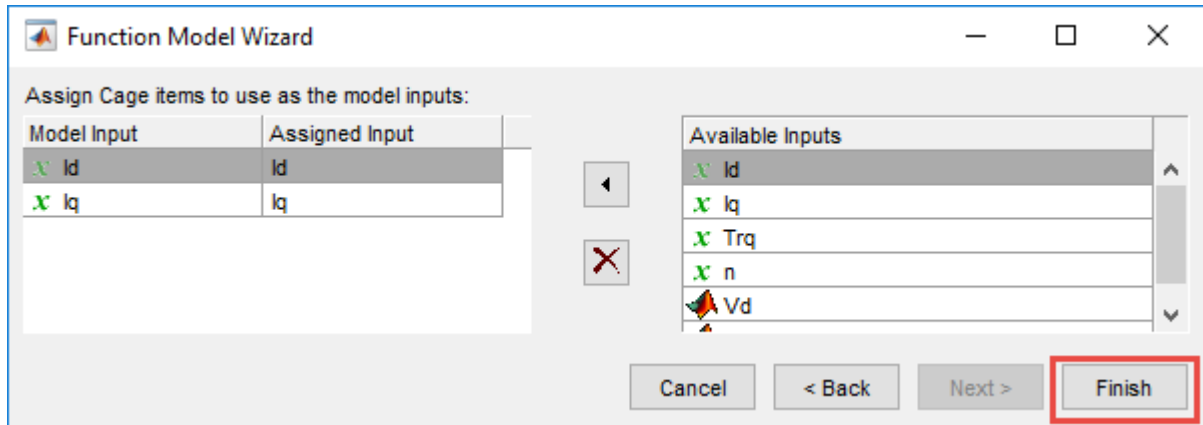| Models					
Name	Type	Inputs	Lower Output Limit	Upper Output Limit	Description
Vd	MBC model	Id, Iq, Trq, n	-Inf	Inf	Created by on 21-Apr-2017.
Vq	MBC model	Id, Iq, Trq, n	-Inf	Inf	Created by on 21-Apr-2017.
Is	Function model	Id, Iq	-Inf	Inf	sqrt (Id.^2 + Iq.^2)
Vs	Function model	Vd, Vq	-Inf	Inf	sqrt (Vd.^2 + Vq.^2)
TPA	Function model	Is, Trq	-Inf	Inf	Trq./Is

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

Create Tables from Model

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

- d-axis current, I_d, as a function of motor torque, `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 **Tables and Tradeoff**. In Create Tables from Model, select `Vd`. Click **Next**.

7-33
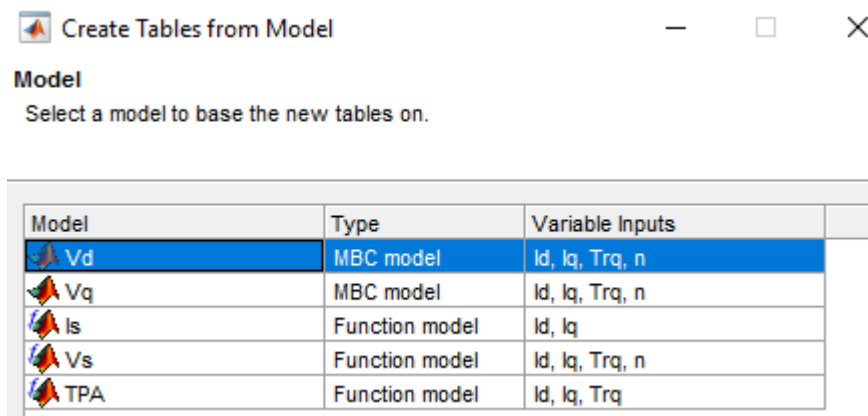
2   In **Table Inputs**, set the **Table rows** to 31. Set **Table columns** to 29. Click **Next**.
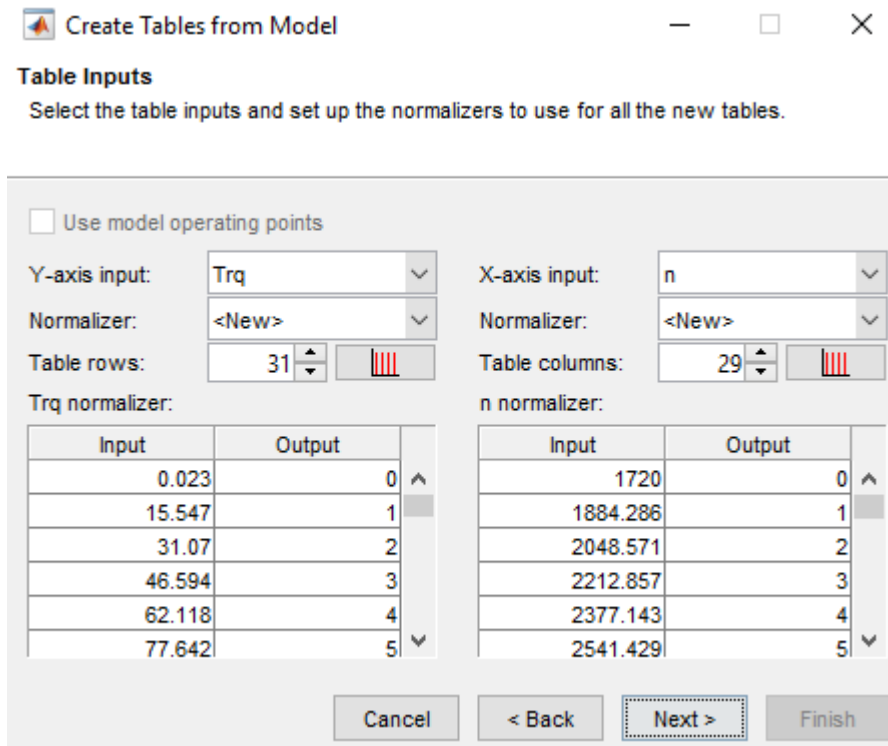
3 In Create Tables from Model:

- Select `Id` and `Iq`.
- Clear `Vd`.
- Click **Finish**.
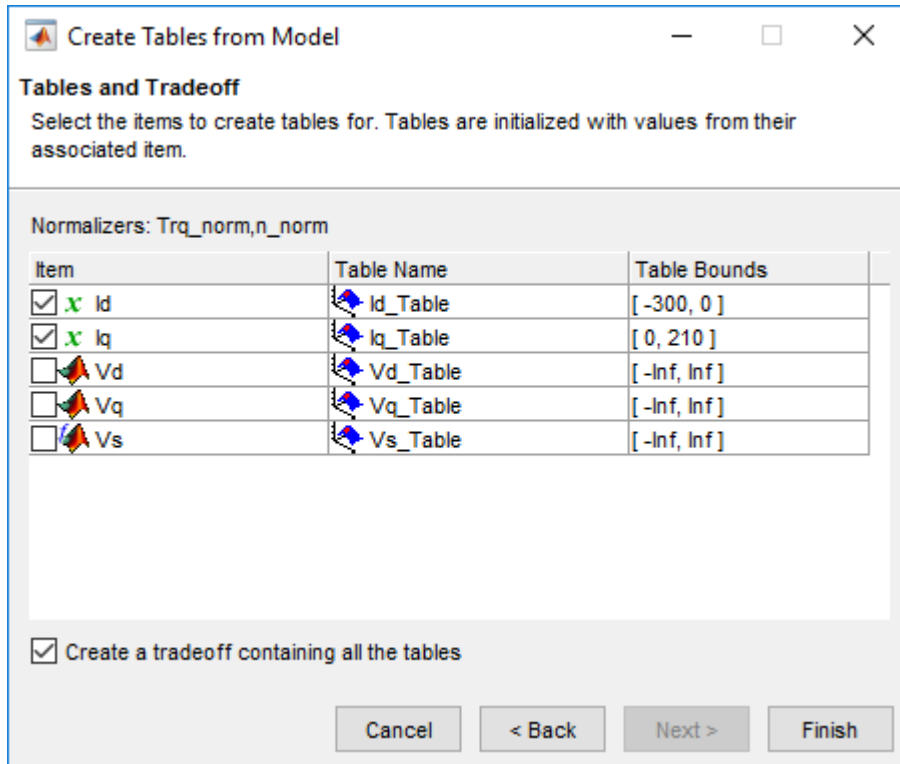
4 In the CAGE Browser, examine the tables.
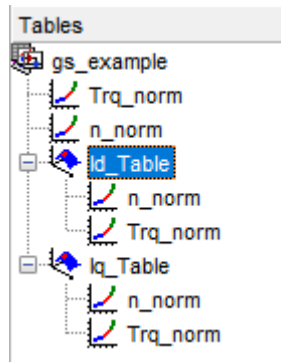
Run Optimization

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

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

1 On the Cage Browser home, select **Optimization**.
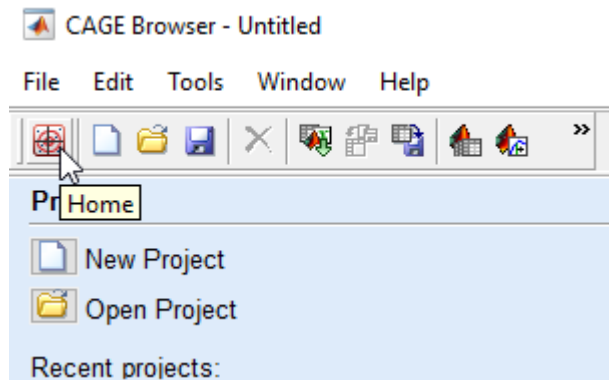
2 In Create Optimization from Model, select TPA and **Next**.
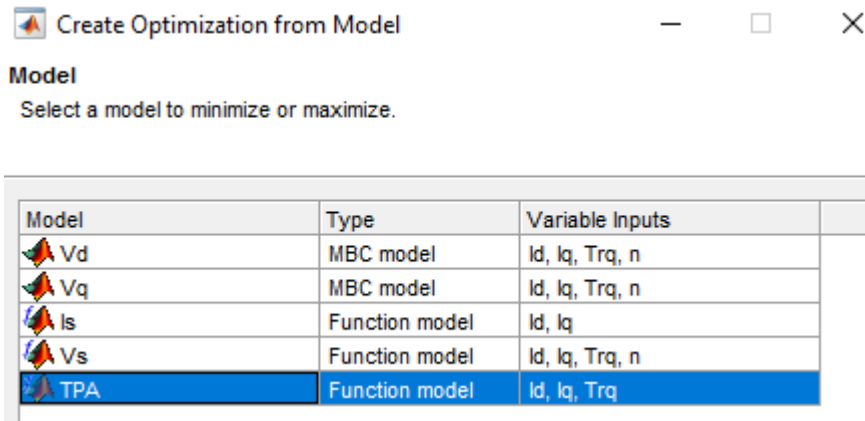
3 In Create Optimization from Model:

- Select `Id` and `Iq`. Clear `Trq`.
- Set **Objective type** to `Maximize`.
- Click **Finish**.
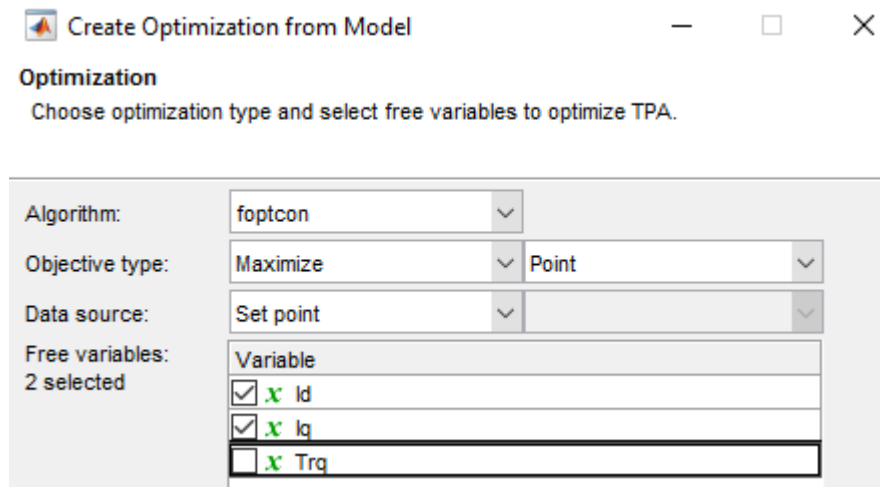
4 Add a boundary constraint for the Vd model. In the CAGE Browser, select **Optimization** > **Constraints** > **Add Constraint**. Set these parameters:

- **Constraint name**: `Vd_boundary`

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

Verify the settings. Click **OK**.
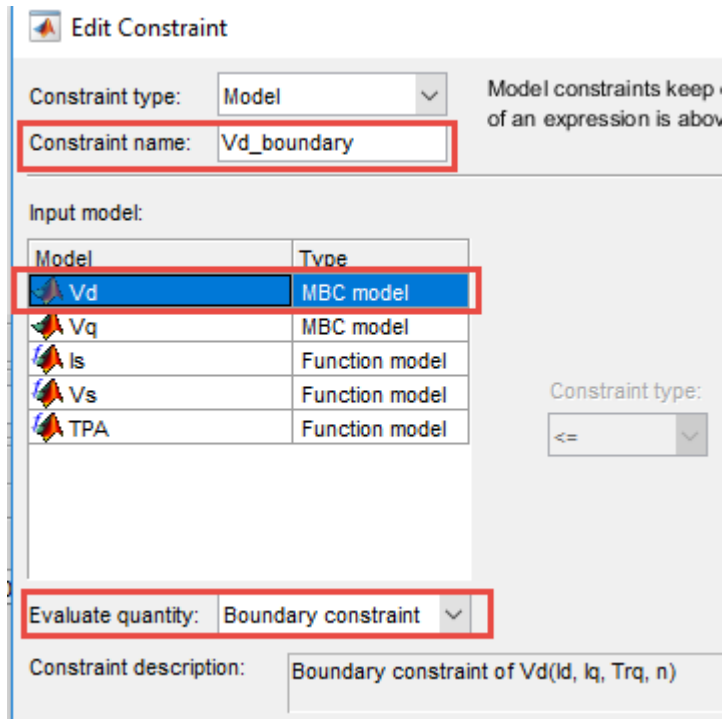
5   Add the optimization constraints. In the CAGE Browser, select **Optimization > Constraints > Add Constraints** to open Edit Constraint. Use the dialog box to create constraints on the current and voltage.

- `Is <= 300`
- `Vs <= 289`
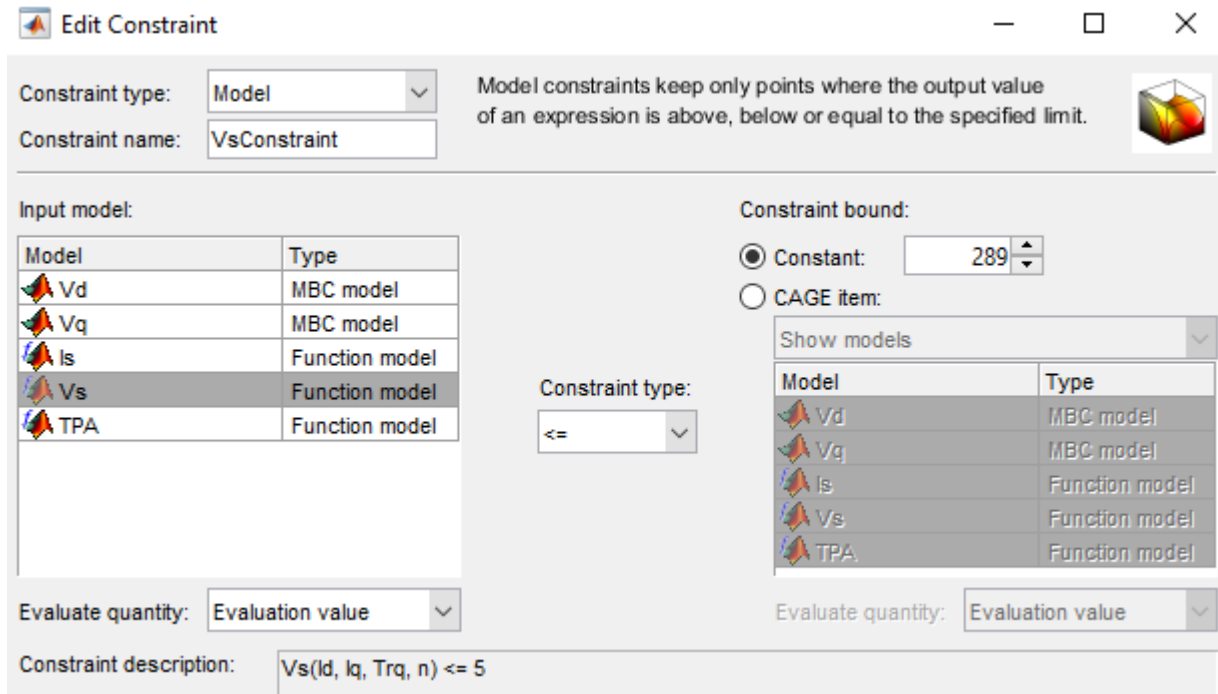
6 In the CAGE Browser, select **Import initial data from a table grid**.
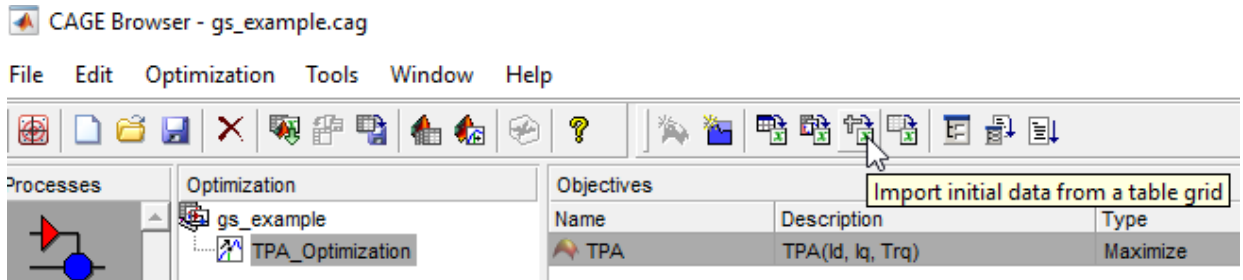
In Import From Table Grid, select `Id_Table`.

Import From Table Grid

Select table to import grid from:

Name	Size	Row Input	Column Input
Id_Table	31x29	Trq	n
Iq_Table	31x29	Trq	n
Trq_norm	31x1	Trq	
n_norm	29x1	n	

Import options

⦿ Use one table cell for each run (alter number of runs)

◯ Use all table cells for each run (alter length of each variable)
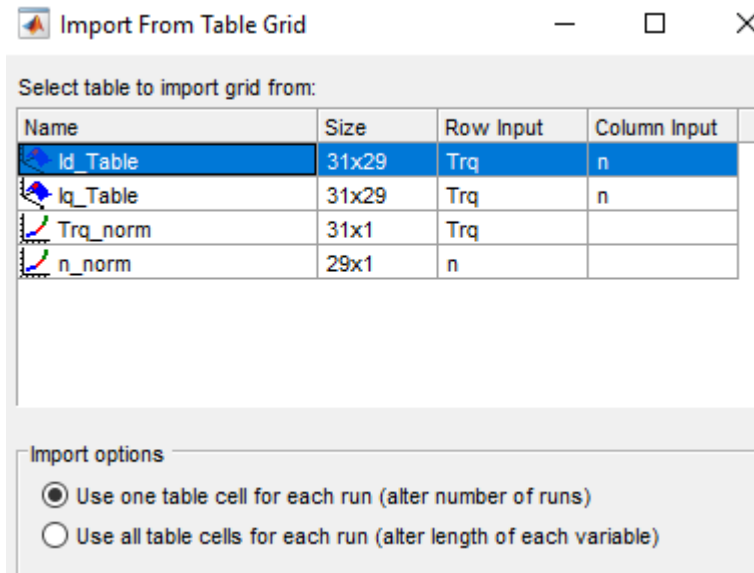
7   In the Cage Browser, *carefully* verify the Objectives and Constraints.

Objectives			
Name	Description	Type	Applicati
⚠ TPA	TPA(Id, Iq, Trq)	Maximize	

Common Tasks

🗀 Add Constraint...

📧 Set Up

📥 Run...

🖼 View Results

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

Optimization Information	
Algorithm name	mbcOSfmincon
Algorithm descri...	Single objective optimizati..
Free variables	Id, Iq
Operating point ...	None

Optimization Point Set

Number of runs: 899 ⏶⏷ Vector display format: Expanded vertically ⌄

Free Variables		
Variable:	Id	Iq
Number of values:	1 ⏶⏷	1 ⏶⏷
1	-150	105
2	-150	105
3	-150	105
4	-150	105

Fixed Variables		
Variable:	Trq	n
Number of values:	1 ⏶⏷	1 ⏶⏷
1	0.023	1720
2	15.547	1720
3	31.07	1720
4	46.594	1720

8   In the Cage Browser, select **Set Up**. Set **Number of start points** to 3. Click **Ok**.
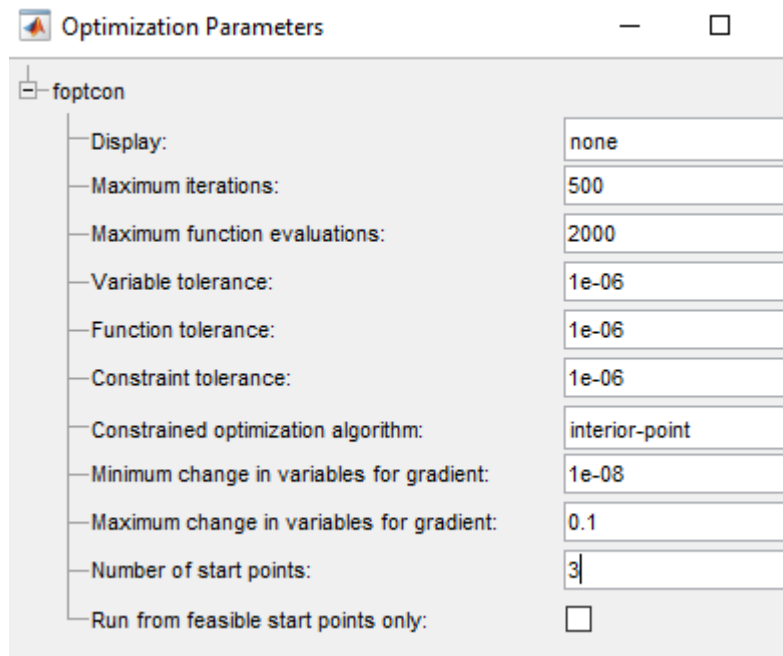
9   In the Cage Browser, select **Run**. The optimization can take time to run and slow other computer processes. View progress in **Running Optimization**.

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

The optimization results are similar to these.
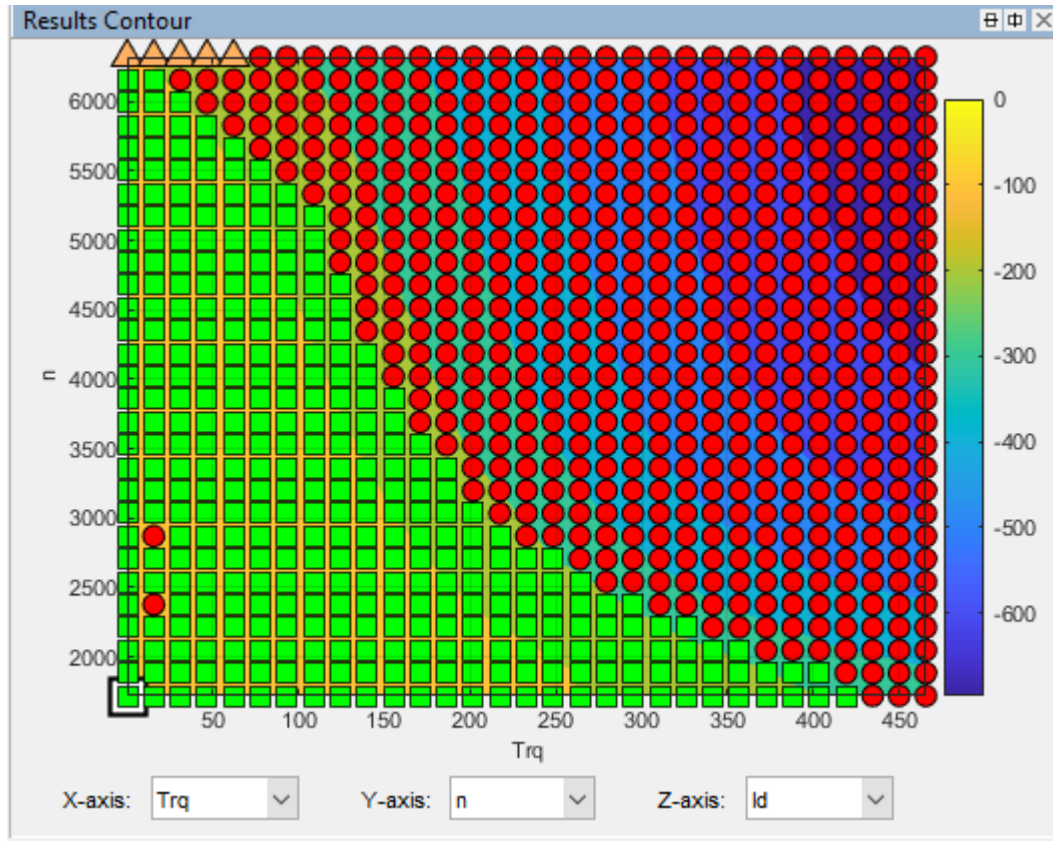
Fill Tables

1. In the CAGE Browser, select **Fill Tables**.
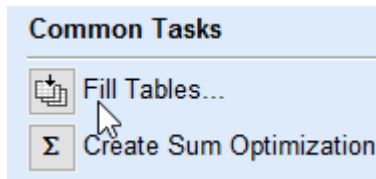
2. Use the Table Filling from Optimization Results Wizard to fill the Id_Table and Iq_Table tables.

Table Filling from Optimization Results Wizard

Table Selection
Select the CAGE tables that you wish to fill from the optimization results
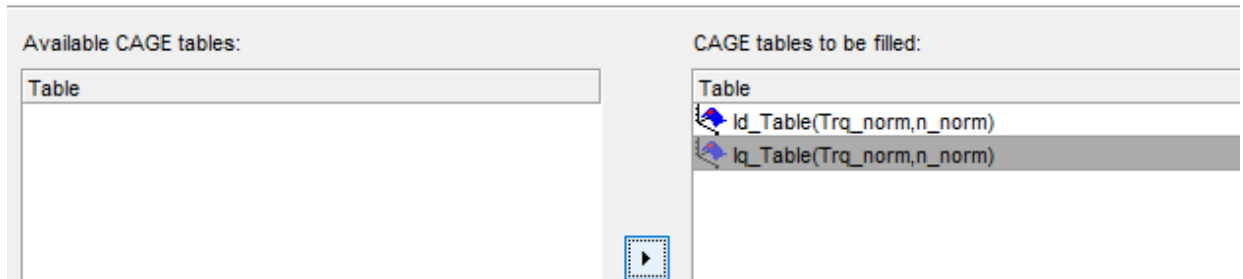
- For the `Id_Table`, fill with `Id`.
- For the `Iq_Table`, fill with `Iq`.
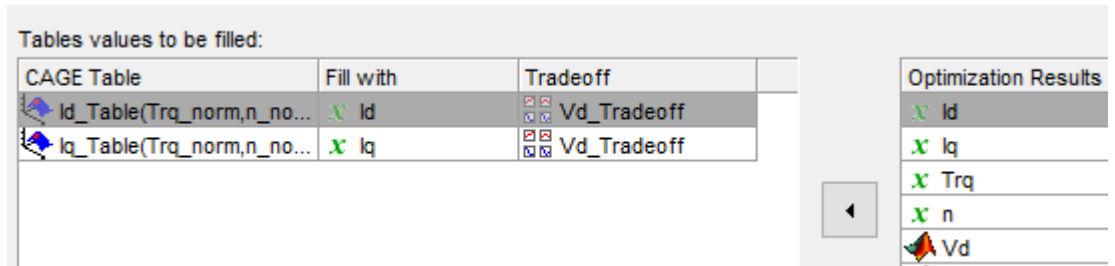
Accept the defaults. Click **Finish**.

3   Review results for `Iq_Table`. The results are similar to these.
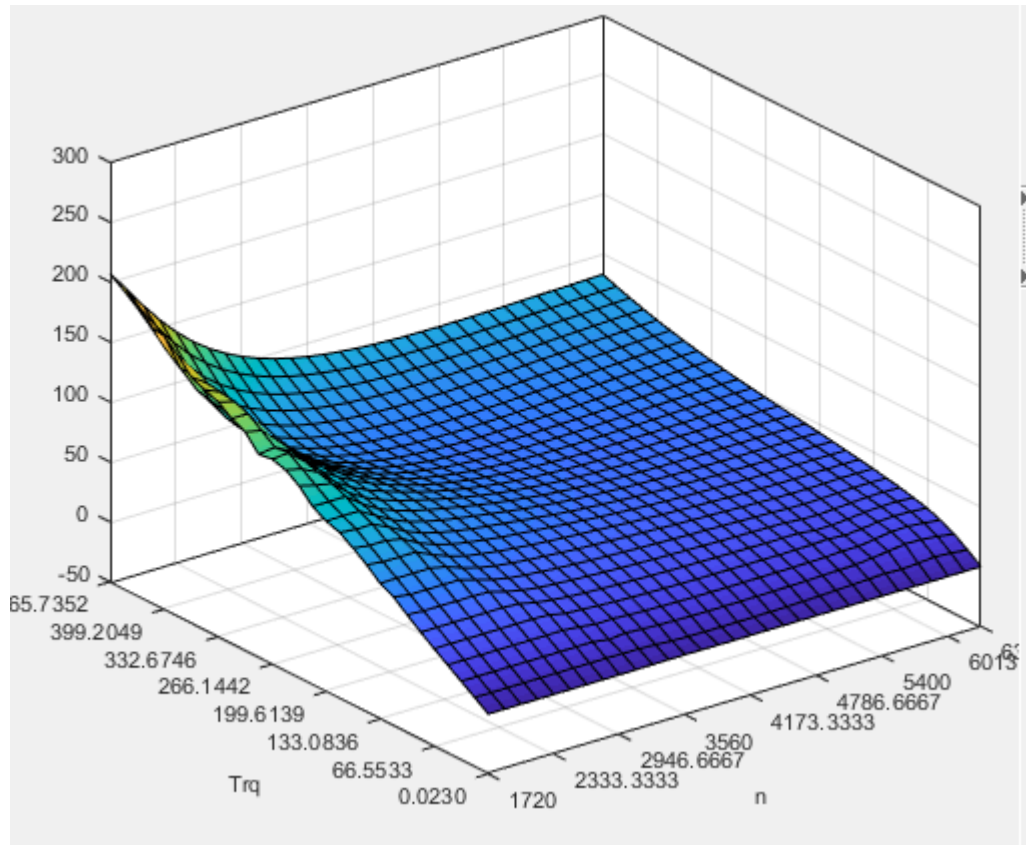
4 Review results for `Id_Table`. The results are similar to these.
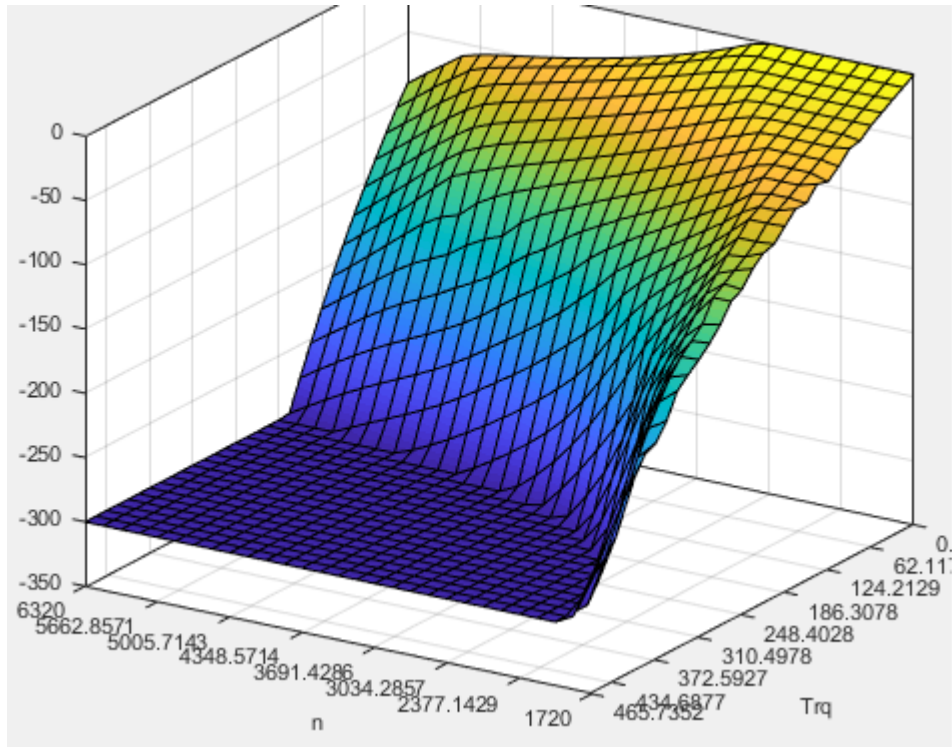
5 Select **File** > **Save Project**. Save gs_example.cag to work folder.

Export Results

1 Select **File** > **Export** > **Calibration** > **All Items**.

2 Use Export Calibration Data to select the items to export and format. For example, export the Id and Iq tables and breakpoints to MATLAB file gs_example.m.
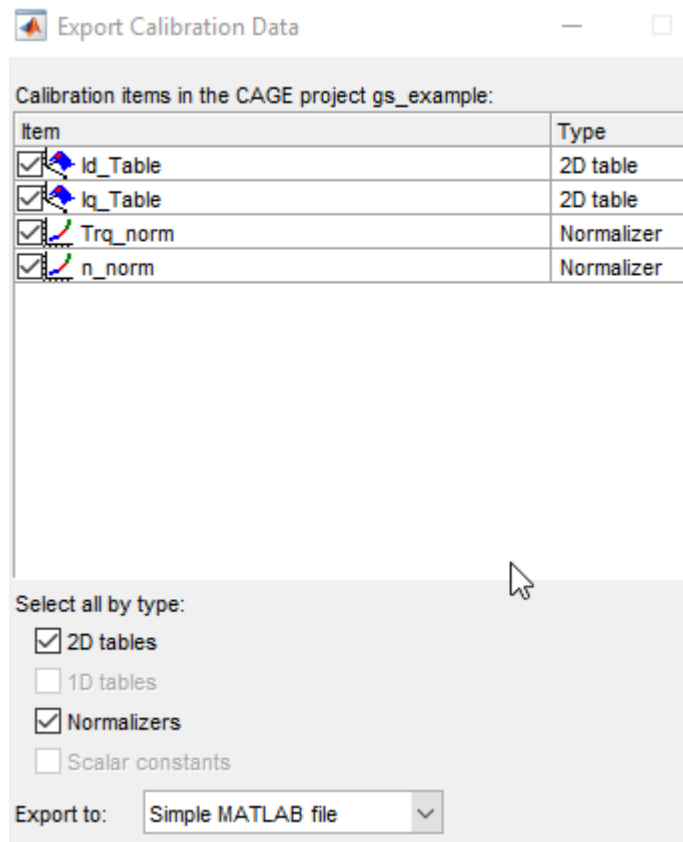
References

[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.

Mapped Engine Lookup Tables

The Model-Based Calibration Toolbox includes projects and templates that you can use to calibrate mapped engine lookup tables. Use the lookup tables in these Powertrain Blockset™ engine blocks:

- Mapped CI Engine — Implements a mapped compression-ignition (CI) engine model
- Mapped SI Engine — Implements a mapped spark-ignition (SI) engine model

To help you get started with the calibration, the CAGE browser and Model Explorer include these templates and examples.

Example	Template	Lookup Tables
"Mapped CI Lookup Tables as Functions of Fuel Mass and Engine Speed" on page 7-50	`CI Mapped Engine-Fuel Input`	• Power
"Mapped CI Lookup Tables as Functions of Engine Torque and Speed" on page 7-64	`CI Mapped Engine-Torque Input`	• Air • Fuel • Temperature • Efficiency • Hydrocarbon (HC) emissions • Carbon monoxide (CO) emissions
"Mapped SI Lookup Tables as Functions of Engine Torque and Speed" on page 7-79	`SI Mapped Engine-Torque Input`	• Nitric oxide and nitrogen dioxide (NOx) emissions • Carbon dioxide (CO2) emissions • Particulate matter (PM) emissions

See Also

Mapped CI Engine | Mapped SI Engine

Mapped CI Lookup Tables as Functions of Fuel Mass and Engine Speed

The Model-Based Calibration Toolbox includes projects and templates that you can use to generate calibrated compression-ignition (CI) lookup tables as a function of fuel mass and engine speed. Use the tables in the Powertrain Blockset Mapped CI Engine block.

Use Test Plan Template to Fit Models

1 In the Model Browser, to open the data, select **Import Data**. Navigate to the spreadsheet that contains the data.

   For example, open `matlab\toolbox\mbc\mbctraining\CiEngineData.xlsx`.

   The spreadsheet contains firing and motor data collected at different engine torques and speeds.

Firing Data	Description
FuelMassCmd	Commanded fuel mass, in mg
Torque	Engine torque, in Nm
EngSpd	Engine speed, in rpm
AirMassFlwRate	Air mass flow, in kg/s
BSFC	Engine brake-specific fuel consumption (BSFC), in g/kWh
CO2MassFlwRate	Carbon dioxide emission mass flow, in kg/s
COMassFlwRate	Carbon monoxide emission mass flow, in kg/s
ExhTemp	Exhaust temperature, in K
FuelMassFlwRate	Fuel mass flow, in kg/s
HCMassFlwRate	Hydrocarbon emission mass flow, in kg/s
NOxMassFlwRate	Nitric oxide and nitrogen dioxide emissions mass flow, in kg/s
PMMassFlwRate	Particulate matter emission mass flow, in kg/s

Nonfiring motor data is collected at different engine speeds, without fuel consumption.

Nonfiring Data	Description
Torque	Engine torque command, in Nm
EngSpd	Engine speed, in rpm
AirMassFlwRate	Air mass flow, in kg/s

2 In the Select Sheet dialog box, select the data that you want to calibrate. For example, select `Firing Data`.

3 Optionally, use the Data Editor filter the data. After you have filtered the data, close the Data Editor.

4 In the Model Browser, select **Fit Models**. In the Fit Models dialog box, in the Template pane, select the template.

For example, to fit the firing data in the spreadsheet, select `MappedEngine-Fuel`. Do not change the default responses and inputs.
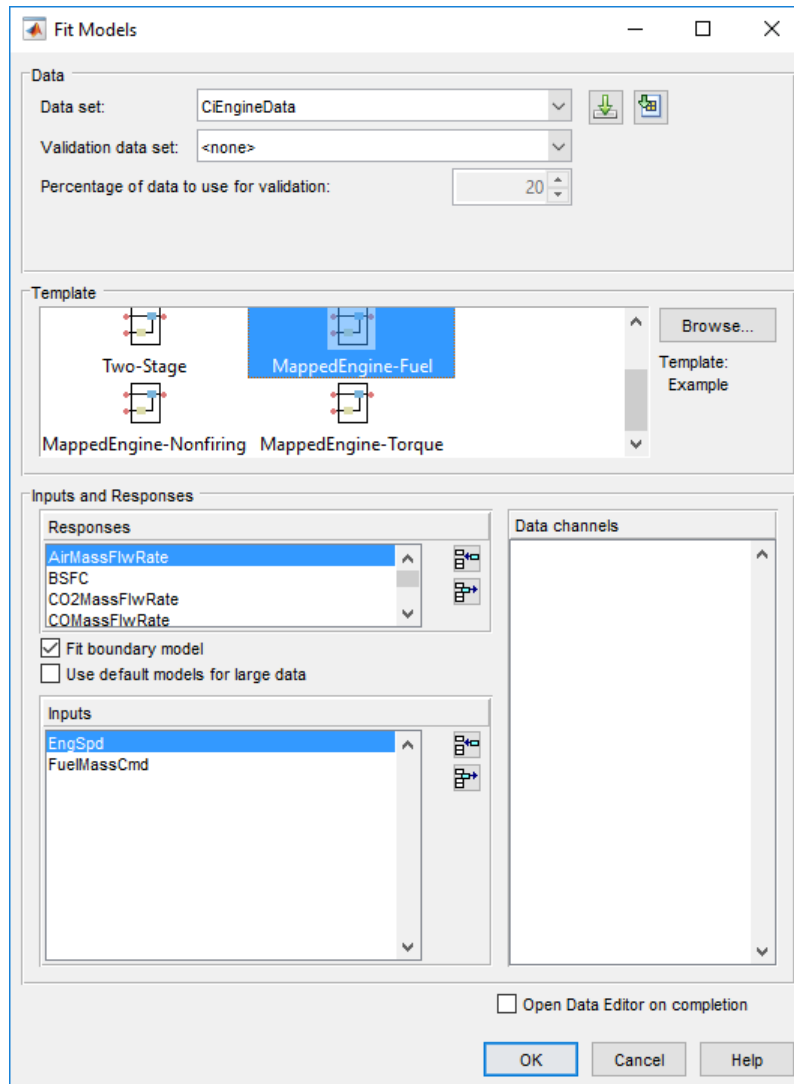
5 Review the model fits.

To review the response models, in the tree, select the top level.
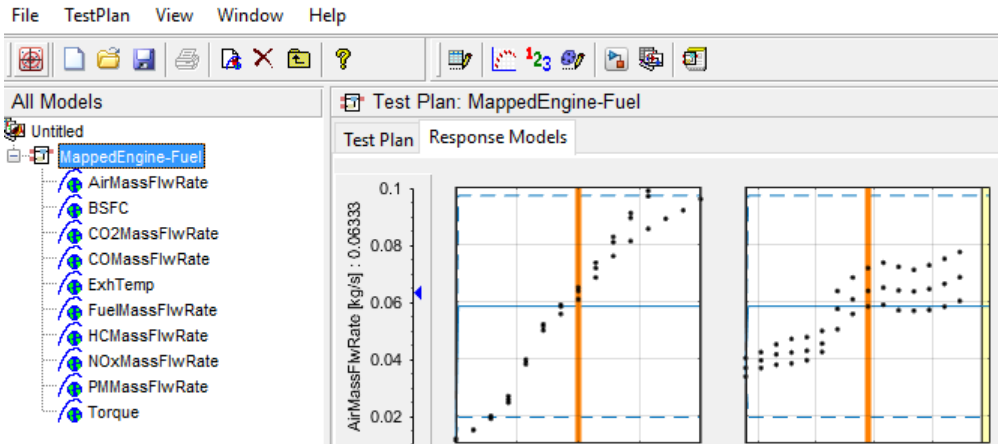
To review the response surfaces, in the tree, select the response.
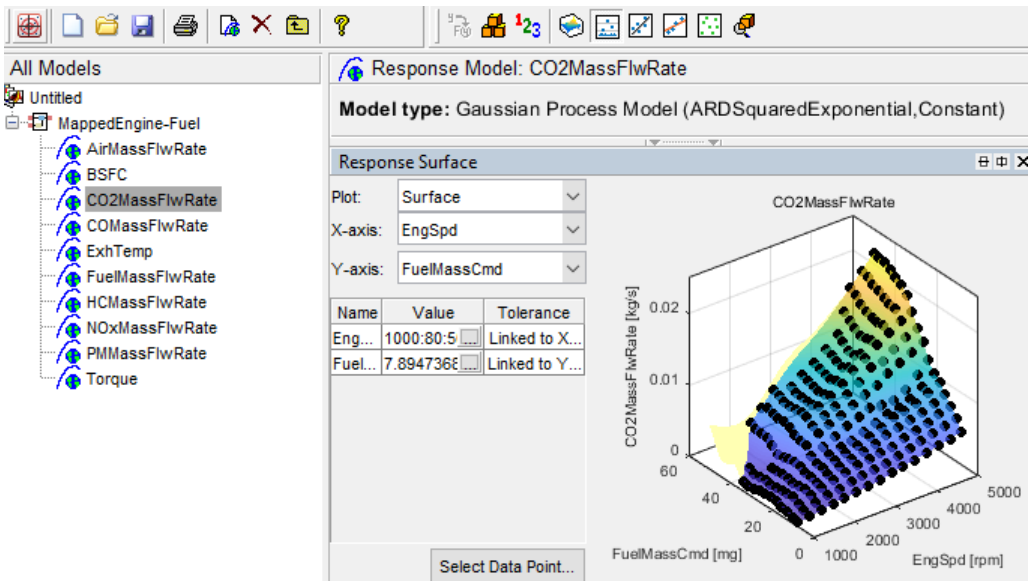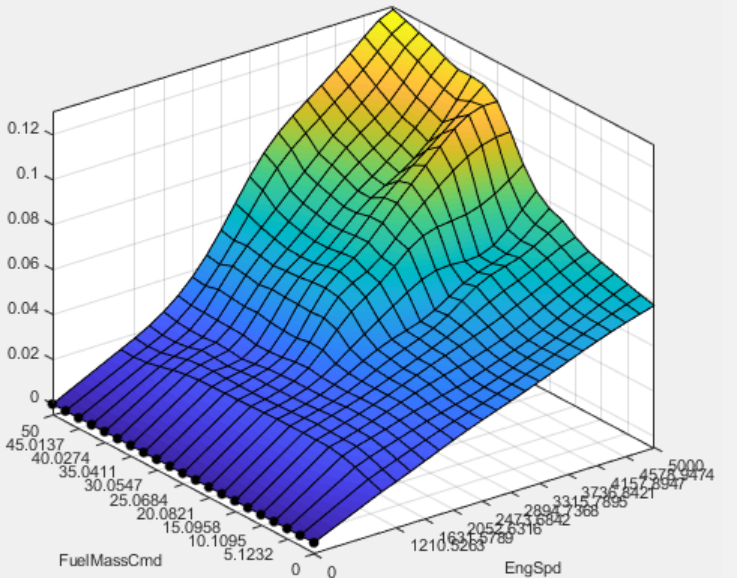
Open CAGE Project
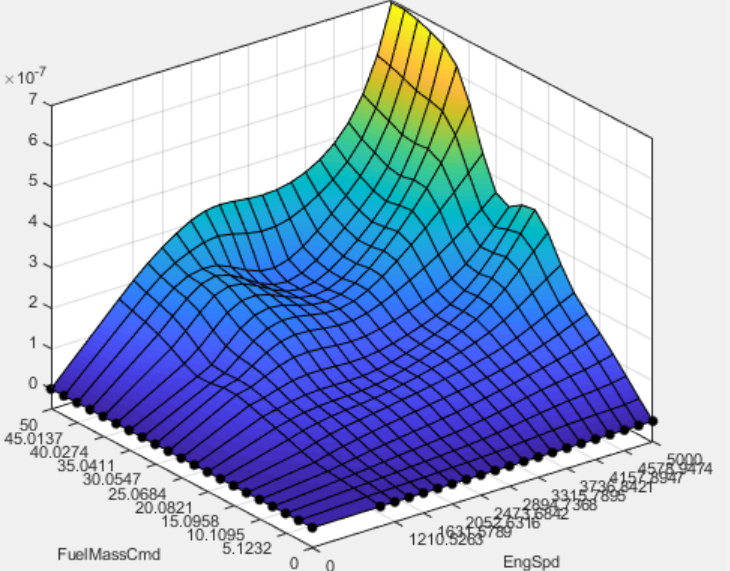
To open the project, in the CAGE **Case Studies** pane:

1   Select CI Mapped Engine - Fuel Input.

2   Click **Open Example**.
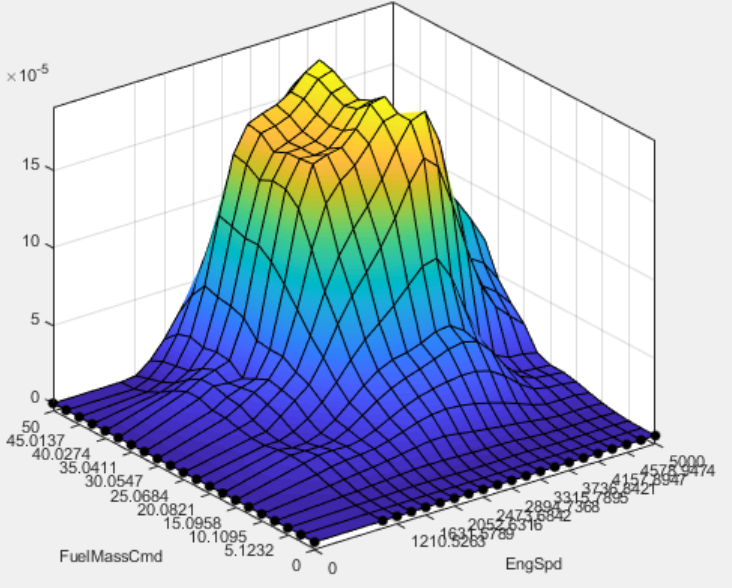
The project includes these tables.

Name	Description	Table
f_air	Air mass flow, in kg/s	
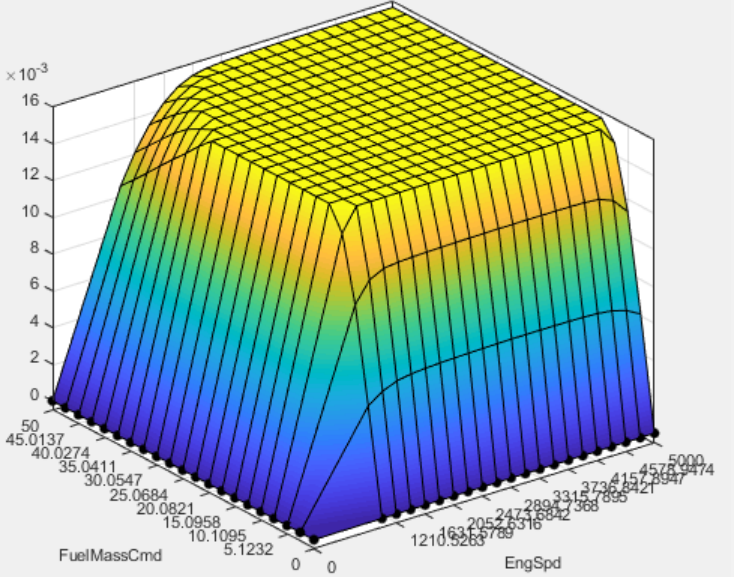
Name	Description	Table
f_eff	Engine brake-specific fuel consumption (BSFC), in g/kWh	
f_co2	Carbon dioxide emission mass flow, in kg/s	

Name	Description	Table
f_co	Carbon monoxide emission mass flow, in kg/s	

Name	Description	Table
f_texh	Exhaust temperature, in K	

Name	Description	Table
f_fuel	Fuel mass flow, in kg/s	
f_hc	Hydrocarbon emission mass flow, in kg/s	

Name	Description	Table
f_nox	Nitric oxide and nitrogen dioxide emissions mass flow, in kg/s	

Name	Description	Table
f_pm	Particulate matter emission mass flow, in kg/s	

Name	Description	Table
f_brake	Engine torque command, in Nm	

Use CAGE to Import and Replace Models

Use the project to import and replace existing models with new models.

1. In CAGE, select **File > Import > Model**. If you do not have a model open, the model browser opens. Select a model.

2. If your current project has two or more test plans, the Import Models dialog box prompts you to merge compatible models. Select **No**.

3. The Import Models dialog box prompts you to `Replace`, `Skip`, or `Create new` models. Select `Replace`.

Import Models —

Import from MBC Model Fitting
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 Acti
Double-click CAGE Model Name cells to edit names.

Original Name	Action	CAGE Model Name
AirMassFlwRate	Replace ∨	AirMassFlwRate
BSFC	Replace ∨	BSFC
CO2MassFlwRate	Replace ∨	CO2MassFlwRate

After you import and replace the existing models, the Import Models wizard opens the Update Tables dialog box. You can use the Update Table dialog box to rerun optimizations and feature fills to update the tables with the new models.

Update Tables — □ ✕

Rerun optimizations and feature fills to update tables using new models.

Item	Tables	Status
f_air_Feature	f_air	✓
f_co2_Feature	f_co2	✓
f_co_Feature	f_co	✓
f_eff_Feature	f_eff	✓
f_fuel_Feature	f_fuel	✓
f_hc_Feature	f_hc	✓
f_nox_Feature	f_nox	✓
f_pm_Feature	f_pm	✓
f_tbrake_Feature	f_tbrake	✓
f_texh_Feature	f_texh	✓

Complete.

Close

Review and Export Tables

1   In CAGE, review the calibrated tables.

2   To export the tables, select **File > Export > Calibration > All Items**. Use the
Export to parameter to specify the format. To export so that you can use the data for
the Powertrain Blockset mapped engine blocks, select `Simulink Model`
`Workspace`. The Model-Based Calibration Toolbox saves the mapped engine table
and breakpoint data to the model workspace.

See Also

Mapped CI Engine

More About

- "Data Manipulation"
- "Assess High-Level Model Trends"
- "Assess One-Stage Models"
- "Generate Mapped CI Engine from a Spreadsheet" (Powertrain Blockset)

Mapped CI Lookup Tables as Functions of Engine Torque and Speed

The Model-Based Calibration Toolbox includes projects and templates that you can use to generate calibrated compression-ignition (CI) lookup tables as a function of engine torque and speed. Use the tables in the Powertrain Blockset Mapped CI Engine block.

Use Test Plan Template to Fit Models

1   In the Model Browser, to open the data, select **Import Data**. Navigate to the spreadsheet that contains the data.

For example, open `matlab\toolbox\mbc\mbctraining\CiEngineData.xlsx`.

The spreadsheet contains firing and motor data collected at different engine torques and speeds.

Firing Data	Description
FuelMassCmd	Commanded fuel mass, in mg
Torque	Engine torque, in Nm
EngSpd	Engine speed, in rpm
AirMassFlwRate	Air mass flow, in kg/s
BSFC	Engine brake-specific fuel consumption (BSFC), in g/kWh
CO2MassFlwRate	Carbon dioxide emission mass flow, in kg/s
COMassFlwRate	Carbon monoxide emission mass flow, in kg/s
ExhTemp	Exhaust temperature, in K
FuelMassFlwRate	Fuel mass flow, in kg/s
HCMassFlwRate	Hydrocarbon emission mass flow, in kg/s
NOxMassFlwRate	Nitric oxide and nitrogen dioxide emissions mass flow, in kg/s
PMMassFlwRate	Particulate matter emission mass flow, in kg/s

Nonfiring motor data is collected at different engine speeds, without fuel consumption.

Nonfiring Data	Description
Torque	Engine torque command, in Nm
EngSpd	Engine speed, in rpm
AirMassFlwRate	Air mass flow, in kg/s

2   In the Select Sheet dialog box, select the data that you want to calibrate. For example, select `Firing Data`.

3   Optionally, use the Data Editor filter the data. After you have filtered the data, close the Data Editor.

4   In the Model Browser, select **Fit Models**. In the Fit Models dialog box, in the Template pane, select the template.

For example, to fit the firing data in the spreadsheet, select `MappedEngine-Torque`. Do not change the default responses and inputs.

5 Review the model fits.

To review the response models, in the tree, select the top level.

To review the response surfaces, in the tree, select the response.

Open CAGE Project

To open the project, in the CAGE **Case Studies** pane:

1   Select CI Mapped Engine - Torque Input.

2 Click **Open Example**.

The project includes these tables.

Name	Description	Table
f_air	Air mass flow, in kg/s	

Name	Description	Table
f_eff	Engine brake-specific fuel consumption (BSFC), in g/kWh	

Name	Description	Table
f_co2	Carbon dioxide emission mass flow, in kg/s	

Name	Description	Table
f_co	Carbon monoxide emission mass flow, in kg/s	

Name	Description	Table
f_texh	Exhaust temperature, in K	
f_fuel	Fuel mass flow, in kg/s	

Name	Description	Table
f_hc	Hydrocarbon emission mass flow, in kg/s	

Name	Description	Table
f_nox	Nitric oxide and nitrogen dioxide emissions mass flow, in kg/s	

Name	Description	Table
f_pm	Particulate matter emission mass flow, in kg/s	

Name	Description	Table
`f_brake`	Engine torque command, in Nm	

Use CAGE to Import and Replace Models

Use the project to import and replace existing models with new models.

1 In CAGE, select **File** > **Import** > **Model**. If you do not have a model open, the model browser opens. Select a model.

2 If your current project has two or more test plans, the Import Models dialog box prompts you to merge compatible models. Select **No**.

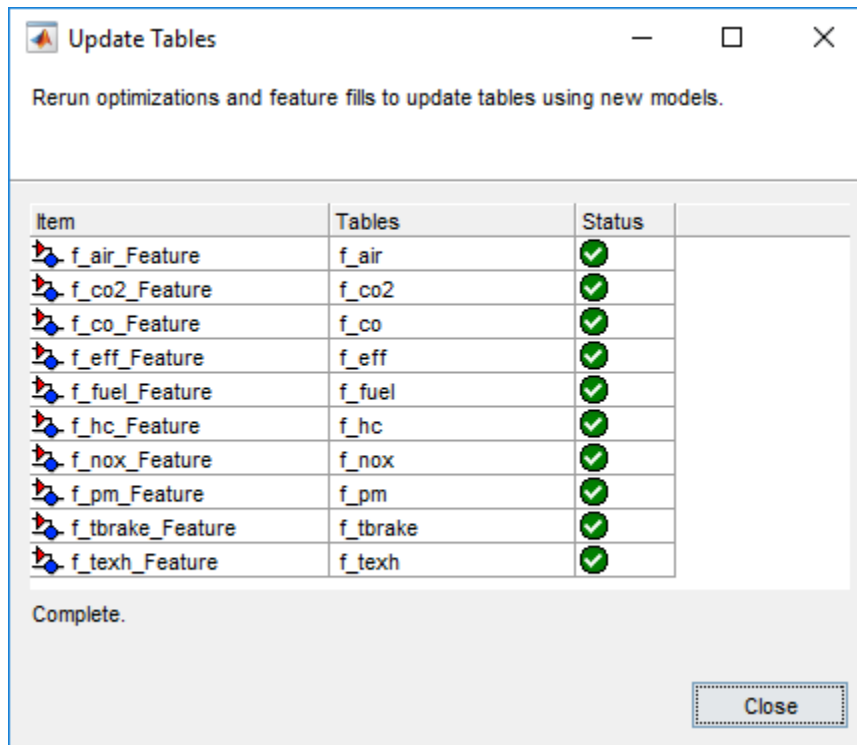3 The Import Models dialog box prompts you to `Replace`, `Skip`, or `Create new` models. Select `Replace`.

After you import and replace the existing models, the Import Models wizard opens the Update Tables dialog box. You can use the Update Table dialog box to rerun optimizations and feature fills to update the tables with the new models.

Review and Export Tables

1   In CAGE, review the calibrated tables.
2   To export the tables, select **File** > **Export** > **Calibration** > **All Items**. Use the **Export to** parameter to specify the format. To export so that you can use the data for the Powertrain Blockset mapped engine blocks, select `Simulink Model Workspace`. The Model-Based Calibration Toolbox saves the mapped engine table and breakpoint data to the model workspace.

See Also

Mapped CI Engine

More About

- "Data Manipulation"
- "Assess High-Level Model Trends"
- "Assess One-Stage Models"
- "Generate Mapped CI Engine from a Spreadsheet" (Powertrain Blockset)

Mapped SI Lookup Tables as Functions of Engine Torque and Speed

The Model-Based Calibration Toolbox includes a project and template that you can use to generate calibrated spark-ignition (SI) lookup tables as a function of engine torque and speed. Use the tables in the Powertrain Blockset Mapped SI Engine block.

Use Test Plan Template to Fit Models

1   In the Model Browser, to open the data, select **Import Data**. Navigate to the spreadsheet that contains the data.

For example, open `matlab\toolbox\mbc\mbctraining\SiEngineData.xlsx`.

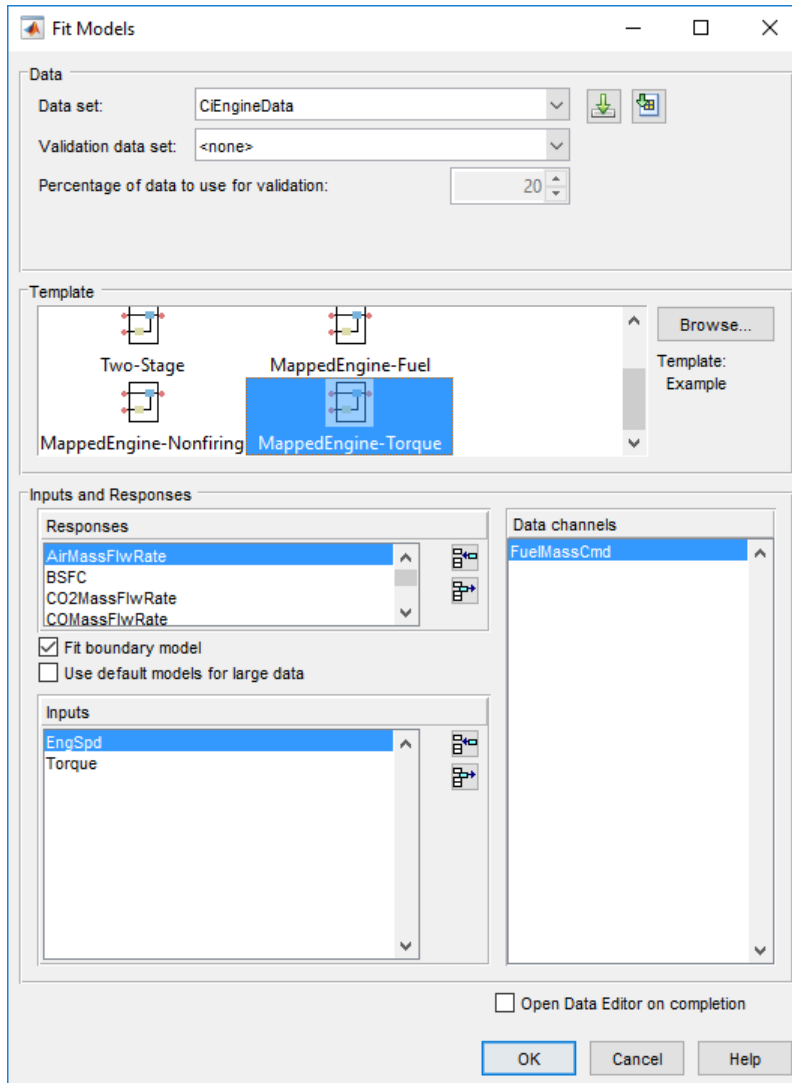The spreadsheet contains firing and motor data collected at different engine torques and speeds.

Firing Data	Description
Torque	Engine torque, in Nm
EngSpd	Engine speed, in rpm
AirMassFlwRate	Air mass flow, in kg/s.
BSFC	Engine brake-specific fuel consumption (BSFC), in g/kWh
CO2MassFlwRate	Carbon dioxide emission mass flow, in kg/s
COMassFlwRate	Carbon monoxide emission mass flow, in kg/s
ExhTemp	Exhaust temperature, in K
FuelMassFlwRate	Fuel mass flow, in kg/s
HCMassFlwRate	Hydrocarbon emission mass flow, in kg/s
NOxMassFlwRate	Nitric oxide and nitrogen dioxide emissions mass flow, in kg/s
PMMassFlwRate	Particulate matter emission mass flow, in kg/s

Nonfiring motor data is collected at different engine speeds, without fuel consumption.

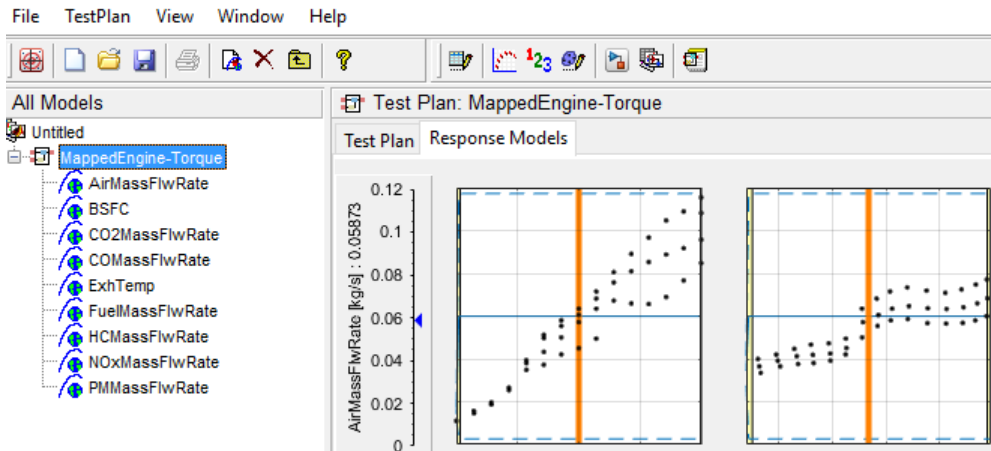| Nonfiring Data	Description
Torque	Engine torque command, in Nm
EngSpd	Engine speed, in rpm
AirMassFlwRate	Air mass flow, in kg/s

2 In the Select Sheet dialog box, select the data that you want to calibrate. For example, select `Firing Data`.

3 Optionally, use the Data Editor filter the data. After you have filtered the data, close the Data Editor.

4 In the Model Browser, select **Fit Models**. In the Fit Models dialog box, in the Template pane, select the template.

For example, to fit the firing data in the spreadsheet, select `MappedEngine-Torque`. Do not change the default responses and inputs.
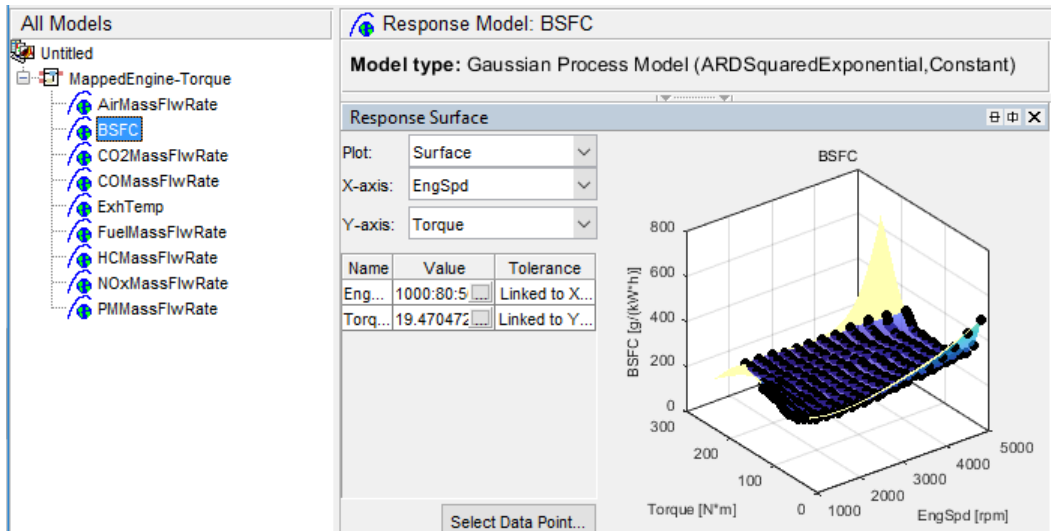
5 Review the model fits.

To review the response models, in the tree, select the top level.

To review the response surfaces, in the tree, select the response.
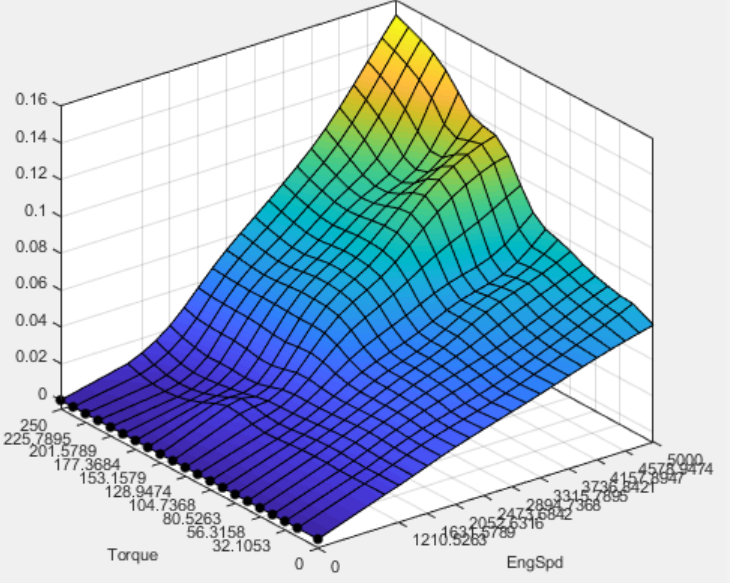

Open CAGE Project

To open the project, in the CAGE **Case Studies** pane:

1   Select SI Mapped Engine - Torque Input.

2   Click **Open Example**.
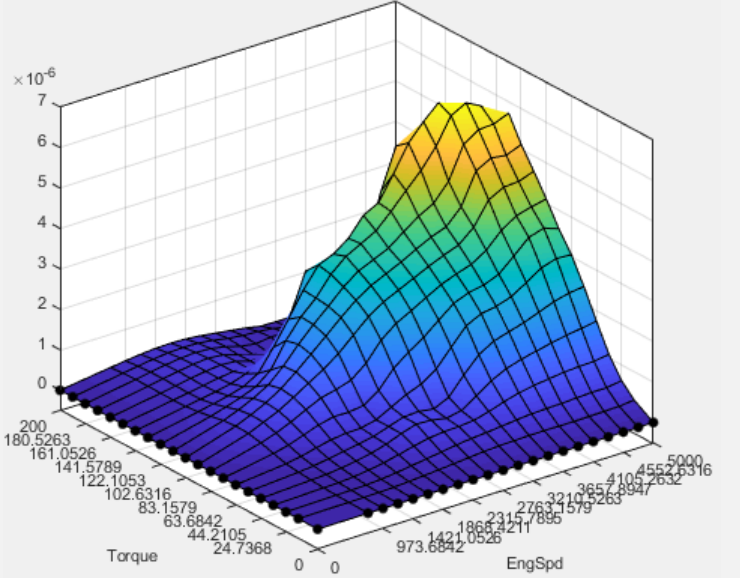
The project includes these tables.

Name	Description	Table
f_air	Air mass flow, in kg/s	
f_eff	Engine brake-specific fuel consumption (BSFC), in g/kWh	

Name	Description	Table
f_co2	Carbon dioxide emission mass flow, in kg/s	
f_co	Carbon monoxide emission mass flow, in kg/s	

Name	Description	Table
f_texh	Exhaust temperature, in K	

Name	Description	Table
f_fuel	Fuel mass flow, in kg/s	
f_hc	Hydrocarbon emission mass flow, in kg/s	

Name	Description	Table
f_nox	Nitric oxide and nitrogen dioxide emissions mass flow, in kg/s	

Name	Description	Table
f_pm	Particulate matter emission mass flow, in kg/s	

Name	Description	Table
f_brake	Engine torque command, in Nm	

Use CAGE to Import and Replace Models

Use the project to import and replace existing models with new models.

1   In CAGE, select **File** > **Import** > **Model**. If you do not have a model open, the model browser opens. Select a model.

2   If your current project has two or more test plans, the Import Models dialog box prompts you to merge compatible models. Select **No**.

3   The Import Models dialog box prompts you to `Replace`, `Skip`, or `Create new` models. Select `Replace`.

Import Models —

Import from MBC Model Fitting

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 Acti
Double-click CAGE Model Name cells to edit names.

Original Name	Action	CAGE Model Name
AirMassFlwRate	Replace	AirMassFlwRate
BSFC	Replace	BSFC
CO2MassFlwRate	Replace	CO2MassFlwRate

After you import and replace the existing models, the Import Models wizard opens the
Update Tables dialog box. You can use the Update Table dialog box to rerun optimizations
and feature fills to update the tables with the new models.

Update Tables — □ ✕

Rerun optimizations and feature fills to update tables using new models.

Item	Tables	Status
f_air_Feature	f_air	✓
f_co2_Feature	f_co2	✓
f_co_Feature	f_co	✓
f_eff_Feature	f_eff	✓
f_fuel_Feature	f_fuel	✓
f_hc_Feature	f_hc	✓
f_nox_Feature	f_nox	✓
f_pm_Feature	f_pm	✓
f_tbrake_Feature	f_tbrake	✓
f_texh_Feature	f_texh	✓

Complete.

Close

Review and Export Tables

1 In CAGE, review the calibrated tables.

2 To export the tables, select **File** > **Export** > **Calibration** > **All Items**. Use the **Export to** parameter to specify the format. To export so that you can use the data for the Powertrain Blockset mapped engine blocks, select `Simulink Model Workspace`. The Model-Based Calibration Toolbox saves the mapped engine table and breakpoint data to the model workspace.

See Also

Mapped SI Engine

More About

- "Data Manipulation"
- "Assess High-Level Model Trends"
- "Assess One-Stage Models"
- "Generate Mapped CI Engine from a Spreadsheet" (Powertrain Blockset)

8

Design of Experiment

This section discusses the following topics:

Design of Experiments

In this section...

Why Use Design of Experiment?

With today's ever-increasing complexity of models, design of experiment has become an essential part of the modeling process. The Design Editor within the Model-Based Calibration Toolbox product is crucial for the efficient collection of engine data. Dyno-cell time is expensive, and the savings in time and money can be considerable when a careful experimental design takes only the most useful data. Dramatically reducing test time is growing more and more important as the number of controllable variables in more complex engines is growing. With increasing engine complexity, the test time increases exponentially.

The traditional method of collecting large quantities of data by holding each factor constant in turn until all possibilities have been tested is an approach that quickly becomes impossible as the number of factors increases. A full factorial design (that is, testing for torque at every combination of speed, load, air/fuel ratio, and exhaust gas recirculation on a direct injection gasoline engine with stratified combustion capability) is not feasible for newer engines. Simple calculation estimates that, for recently developed engines, to calibrate in the traditional way would take 99 years!

With a five-factor experiment including a multiknot spline dimension and 20 levels in each factor, the number of points in a full factorial design quickly becomes thousands, making the experiment prohibitively expensive to run. The Design Editor solves this problem by choosing a set of experimental points that allow estimation of the model with the maximum confidence using just a fraction of the number of experimental runs; for the preceding example just 100 optimally chosen runs is more than enough to fit the model. Obviously, this approach can be advantageous for any complex experimental design, not just engine research.

The Design Editor offers a systematic, rigorous approach to the data collection stage. When you plan a sequence of tests to be run on an example engine, you can base your design on engineering expertise and existing physical and analytical models. During

testing, you can compare your design with the latest data and optimize the remaining tests to get maximum benefit.

The Design Editor provides prebuilt standard designs to allow a user with a minimal knowledge of the subject to quickly create experiments. You can apply engineering knowledge to define variable ranges and apply constraints to exclude impractical points. You can increase modeling sophistication by altering optimality criteria, forcing or removing specific design points, and optimally augmenting existing designs with additional points.

Design Styles

The Design Editor provides the interface for building experimental designs. You can make three different styles of design: classical, space-filling, and optimal.

Space-filling designs are better when there is low system knowledge. In cases where you are not sure what type of model is appropriate, and the constraints are uncertain, space-filling designs collect data in such as a way as to maximize coverage of the factors' ranges as quickly as possible. See "Create a Constrained Space-Filling Design" on page 8-8.

Optimal designs are best for cases with high system knowledge, where previous studies have given confidence on the best type of model to be fitted, and the constraints of the system are well understood. See "Create Optimal Designs" on page 8-14.

Classical designs (including full factorial) are very well researched and are suitable for simple regions (hypercube or sphere). Engines have complex constraints and models (high-order polynomials and splines). See "Create Classical Designs" on page 8-28.

You can augment any design by adding points. Working in this way allows new experiments to enhance the original, rather than simply being a second attempt to gain the necessary knowledge.

Create Examples Using the Design Editor

Follow the steps in the following sections in order. The instructions guide you through constructing space-filling, optimal, and classical designs; how to apply constraints to designs, and how to compare designs using the prediction error variance (PEV) viewer and Design Evaluation tool.

1   Choose a model to design an experiment for, enter the Design Editor, and construct an space-filling design. Then construct and apply two different constraints to this

design and view the results. Often you would design constraints before constructing a design, but for the purposes of this tutorial you make constraints last so you can view the effects on your design.

See

- "Set Up a Model and Create a Design" on page 8-6
- "Create a Constrained Space-Filling Design" on page 8-8

2 Create an optimal design. See "Create Optimal Designs" on page 8-14.

3 After you create designs, use the displays and tools to examine the properties of the design, save the design, and make changes.

- "View Design Displays" on page 8-20
- "Save Designs" on page 8-13
- "Use the Prediction Error Variance Viewer" on page 8-22

4 Create a classical design, and use the Prediction Error Variance Viewer to compare it with the optimal design. You can also use the Design Evaluation tool to view all details of any design.

See

- "Create Classical Designs" on page 8-28
- "Use the Design Evaluation Tool" on page 8-35

For more details on functionality in the Design Editor, see "Design of Experiments".

See Also

Related Examples
- "Set Up a Model and Create a Design" on page 8-6
- "Create a Constrained Space-Filling Design" on page 8-8
- "Apply Constraints" on page 8-8
- "Create Optimal Designs" on page 8-14
- "Create Classical Designs" on page 8-28
- "Save Designs" on page 8-13

More About

- "Design of Experiments"

Set Up a Model and Create a Design

In this section...
"Set Up Model Inputs" on page 8-6
"Open the Design Editor" on page 8-6
"Create a New Design" on page 8-7

Set Up Model Inputs

You must first specify model inputs for which to design an experiment.

1   Start the Model Browser part of the toolbox by typing `mbcmodel` at the MATLAB command line.

2   From the startup project node view, in the **Common Tasks** pane, click **Design experiment**.

     The **New Test Plan** dialog box appears.

3
     Click the `Two-Stage` test plan icon in the Template pane. A two-stage model fits a model to data with a hierarchical structure.

4   There is only one input to the global model by default. Click the up arrow button to increase the **Number of factors** setting to 3.

5   Change the symbols of the three input factors to N, L, and A. This matches the global factors modeled in the tutorial "Empirical Engine Modeling" on page 7-2: speed (n), load (L), and air/fuel ratio (A).

6   Click **OK** to dismiss the dialog box.

     The Model Browser displays the test plan diagram with your specified model inputs.

Open the Design Editor

In the test plan view, in the **Common Tasks** pane, click **Design experiment**.

The Design Editor window appears.

Alternatively, to open the Design Editor, you can also use either of the following methods:

- Right-click the global model in the diagram and choose **Design Experiment**, as shown.
- You can also access the Design Editor by selecting the menu item **TestPlan > Design Experiment**.

Create a New Design

1. Click the ⬜ button in the toolbar or select **File > New**. A new node called appears.
2. The new node is automatically selected. An empty Design Table appears (or any view you last used in the Design Editor) because you have not yet chosen a design. For this example you create an optimal design for the default global model.

   You can change the model for which you are designing an experiment from within the Design Editor window by selecting **Edit > Model**.
3. Rename the new node Space Fill (you can edit the names by clicking again on a node when it is already selected, or by pressing **F2**, as when selecting to rename in Windows Explorer).

For next steps, see "Create a Constrained Space-Filling Design" on page 8-8.

Create a Constrained Space-Filling Design

Space-filling designs should be used when there is little or no information about the underlying effects of factors on responses. For example, they are most useful when you are faced with a new type of engine, with little knowledge of the operating envelope. These designs do not assume a particular model form. The aim is to spread the points as evenly as possible around the operating space. These designs literally fill out the *n*-dimensional space with points that are in some way regularly spaced. These designs can be especially useful with nonparametric models such as radial basis functions (a type of neural network).

1   In the Design Editor, with the new design selected (that you created in "Open the Design Editor" on page 8-6), select **Design > Space Filling > Design Browser**, or click the Space Filling Design button on the toolbar.

The Space Filling Design Browser appears.

1   Leave the default Sobol Sequence in the **Design type** list, and the default **Number of points**.
2   Use the Preview tabs to view 2-D and 3-D previews.
3   Click **OK** to calculate the space-filling design and return to the main Design Editor.

Apply Constraints

In many cases, designs might not coincide with the operating region of the system to be tested. For example, a conventional stoichiometric AFR automobile engine normally does not operate with high exhaust gas recirculation (EGR) in a region of low speed (n) and low load (l). You cannot run 15% EGR at 800 RPM idle with a homogeneous combustion process. There is no point selecting design points in impractical regions, so you can constrain the candidate set for test point generation. Only optimal designs have candidate sets of points; classical designs have set points, and space-filling designs distribute points between the coded values of (1, -1).

You would usually set up constraints *before* making designs. Applying constraints to classical and space-filling designs simply removes points outside the constraint. Constraining the candidate set for optimal designs ensures that design points are optimally chosen within the area of interest only.

Designs can have any number of geometric constraints placed upon them. Each constraint can be one of four types: an ellipsoid, a hyperplane, a 1-D lookup table, or a 2-D lookup table.

To add a constraint to your currently selected design:

1 Select **Edit > Constraints** from the Design Editor menus.

2 The Constraints Manager dialog appears. Click **Add**.

The Constraint Editor dialog with available constraints appears. Leave the default 1D Table in the **Constraint type** list.

3 You can select the appropriate factors to use. For this example, choose speed (N) and air/fuel ratio (A) for the X and Y factors.

4 Move the large dots (click and drag them) to define a boundary. The Constraint Editor should look something like the following.
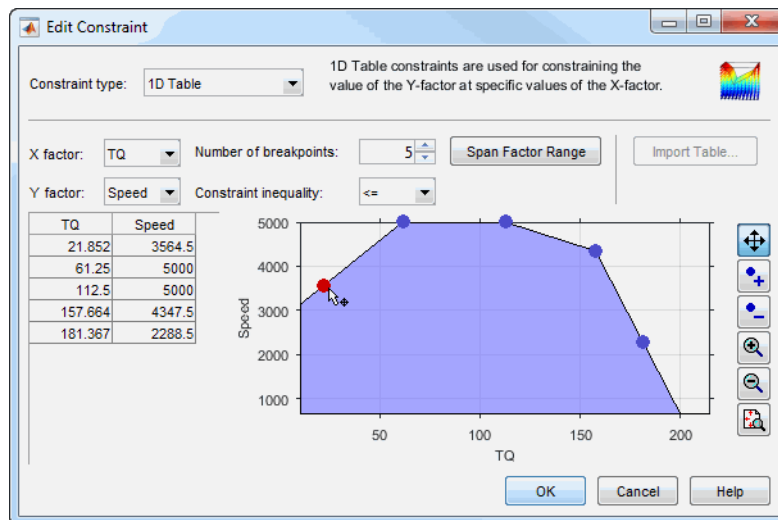
5 Click **OK**.

Your new constraint appears in the Constraint Manager list box. Click **OK** to return to the Design Editor. A dialog appears because there are points in the design that fall outside your newly constrained candidate set.

6 Click **Continue** to remove the points outside the constraint. Note that fixed points are not deleted by this process.
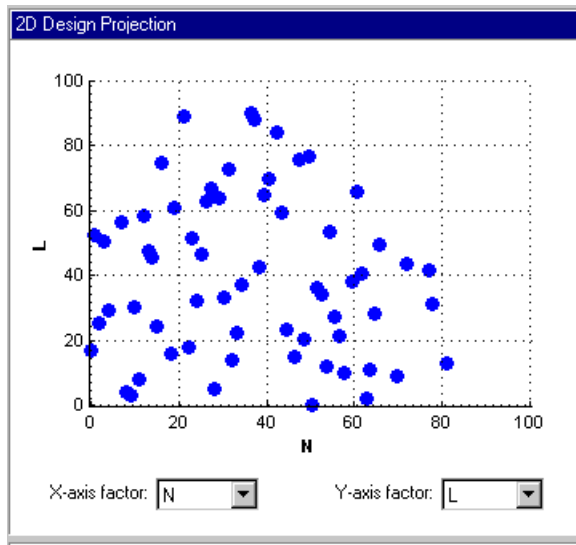
Plot the 2-D projection of the hypercube, and observe the effects of the new constraint on the shape of the design, as shown in the preceding example.

7   Right-click the display pane to reach the context menu, and select **Current View > 3D Constraints**. These views are intended to give some idea of the region of space that is currently available within the constraint boundaries.

8   Return to the Constraint Editor, choose **Edit > Constraint**, and click **Add** in the Constraint Manager.

9   Add an ellipsoid constraint. Choose Ellipsoid from the drop-down menu of constraint types.
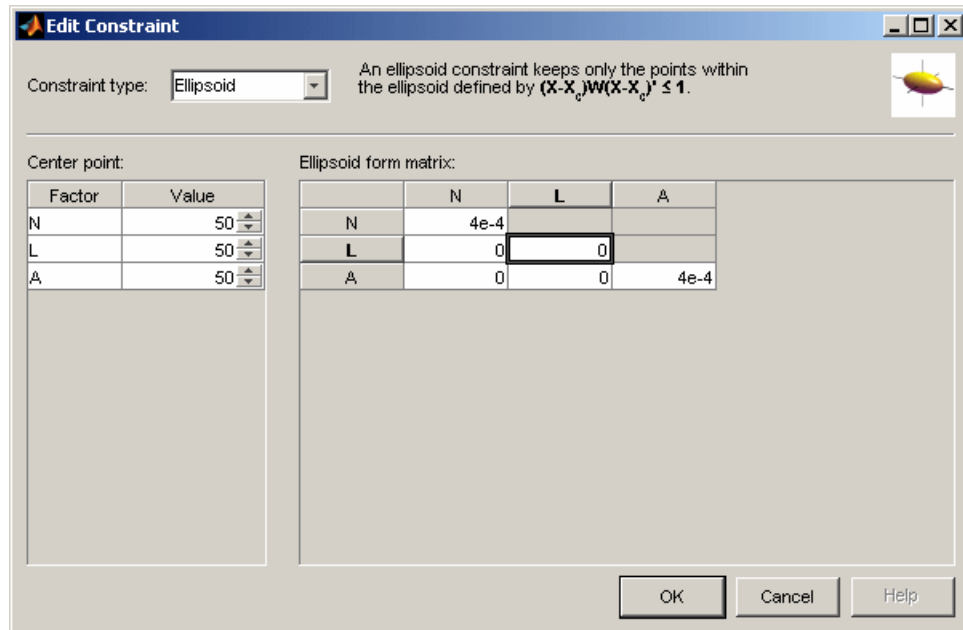
Enter 0 as the value for the **L** diagonal in the table, as shown. This will leave L unconstrained (a cylinder). The default ellipsoid constraint is a sphere. To constrain a factor, if you want a radius of r in a factor, enter $1/(r^2)$. For this example, leave the other values at the defaults. Click **OK** to apply the constraint.

10 Click **OK**, click **OK** again in the Constraint Manager, and click **Continue** to remove design points outside the new candidate set (or **Replace** if you are constraining an optimal design). Examine the new constraint 3-D constraints plot.

Both constraints are applied to this design.

For next steps, see "Save Designs" on page 8-13.

To try other design styles, see "Create Optimal Designs" on page 8-14 and "Create Classical Designs" on page 8-28.

See Also

Related Examples

- "Design of Experiments" on page 8-2
- "Save Designs" on page 8-13
- "Create Optimal Designs" on page 8-14
- "View Design Displays" on page 8-20
- "Create Classical Designs" on page 8-28

More About

- "Design of Experiments"

Save Designs

To save your designs with your project and close the Design Editor, select **File > Save and Close** or the toolbar button.

To export your design to a file:

1 Choose **File > Export Design**. The selected design *only* is exported.

Export to options:

- `Comma separated format file` exports the matrix of design points to a CSV (comma-separated values) file. You can include factor symbols and/or convert to coded values by selecting the check boxes.
- `Design Editor file` generates a Design Editor file (`.mvd`).
- `Workspace` exports the design matrix to the workspace. You can convert design points to a range of [-1, 1] by selecting the check box.

2 Choose a CSV file.

3 Name the destination file by typing `Designtutorial` in the edit box.

4 Click **OK** to save the file.

For more details on functionality in the Design Editor, see "Design of Experiments".

See Also

Related Examples
- "Design of Experiments" on page 8-2
- "Create a Constrained Space-Filling Design" on page 8-8
- "Create Optimal Designs" on page 8-14
- "View Design Displays" on page 8-20
- "Create Classical Designs" on page 8-28

More About
- "Design of Experiments"

Create Optimal Designs

In this section...

Introducing Optimal Designs

In the Design Editor, with a new design selected, create an optimal design by clicking

the button in the toolbar, or choose **Design > Optimal**.

Optimal designs are best for cases with high system knowledge, where previous studies have given confidence on the best type of model to be fitted, and the constraints of the system are well understood.

The optimal designs in the Design Editor are formed using the following process:

- An initial starting design is chosen at random from a set of defined candidate points.
- m additional points are added to the design, either optimally or at random. These points are chosen from the candidate set.
- m points are deleted from the design, either optimally or at random.
- If the resulting design is better than the original, it is kept.

This process is repeated until either (a) the maximum number of iterations is exceeded or (b) a certain number of iterations has occurred without an appreciable change in the optimality value for the design.

The Optimal Design dialog consists of several tabs that contain the settings for three main aspects of the design:

- Starting point and number of points in the design
- Candidate set of points from which the design points are chosen
- Options for the algorithm that is used to generate the points

Start Point Tab

The **Start Point** tab allows you to define the composition of the initial design: how many points to keep from the current design and how many extra to choose from the candidate set.

1 Leave the optimality criteria at the default to create a `V-Optimal` design.

2 Increase the total number of points to 30 by clicking the **Optional additional points** up/down buttons or by typing directly into the edit box. You can edit the additional points and/or the total number of points.
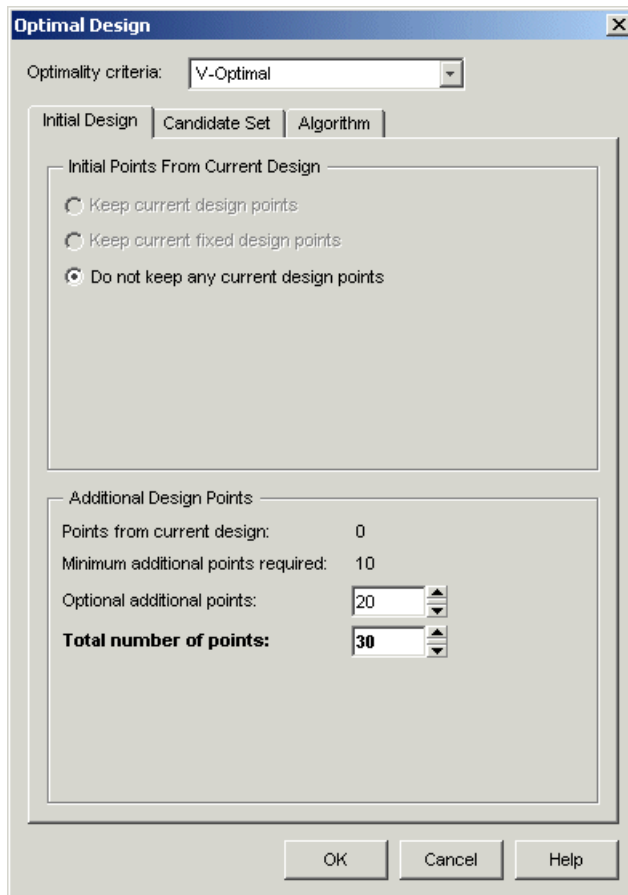
Candidate Set Tab

The **Candidate Set** tab allows you to set up a candidate set of potential test points. This typically ranges from a few hundred points to several hundred thousand.

1   Choose `Grid` for this example. Note that you could choose different schemes for different factors.

2   This tab also has buttons for creating plots of the candidate sets. Try them to preview the grid.

3   Notice that you can see 1-D, 2-D, 3-D, and 4-D displays (the fourth factor is color, but this example only uses three factors) at the same time as they appear in separate windows (see example following). Look at a display window while changing the number of levels for the different factors. See the effects of changing the number of levels on different factors, then return them all to the default of 21 levels.

Select variables in this list

Choose grid from this list

Open display windows with these buttons.

Change the number of levels of the selected variable here.

Algorithm Tab
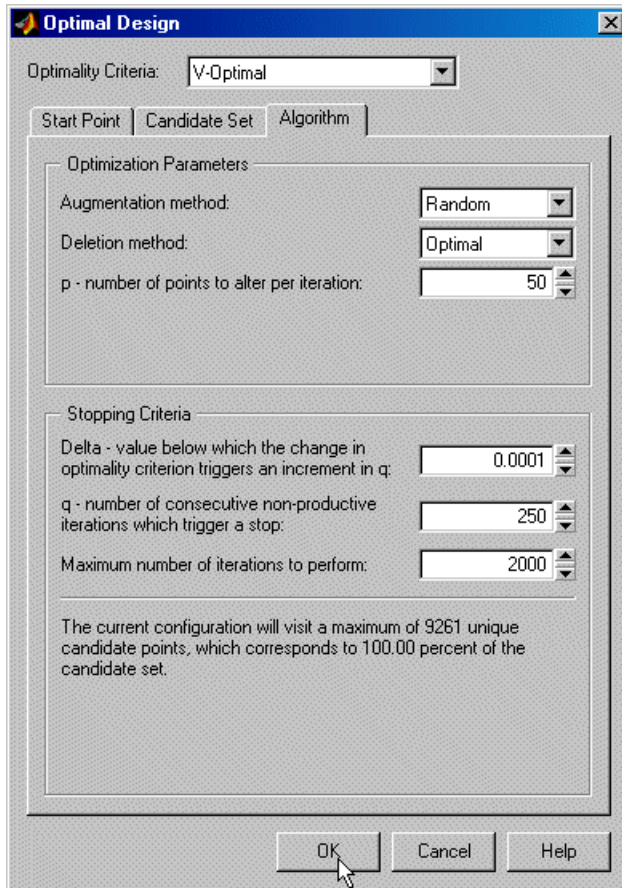
1   Leave the algorithm settings at the defaults and click **OK** to start optimizing the design.

When you click the **OK** button on the Optimal Design dialog, the Optimizing Design dialog appears, containing a graph. This dialog shows the progress of the optimization and has two buttons: **Accept** and **Cancel**. **Accept** stops the optimization early and takes the current design from it. **Cancel** stops the optimization and reverts to the original design.
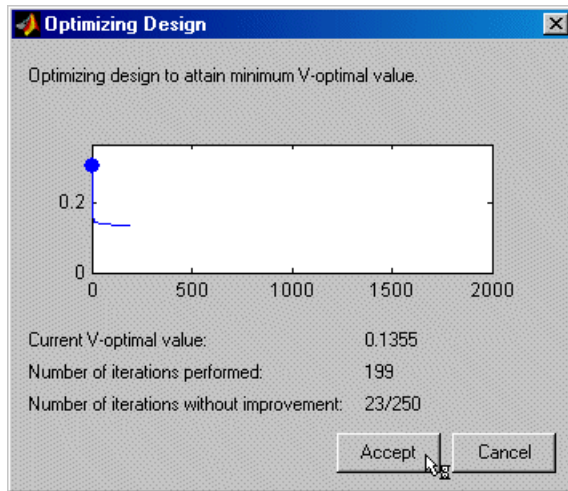
2 Click **Accept** when iterations are not producing noticeable improvements; that is, the graph becomes very flat.

For next steps, see "View Design Displays" on page 8-20, "Use the Prediction Error Variance Viewer" on page 8-22, and "Create Classical Designs" on page 8-28.

View Design Displays

When you click **Accept** button, you return .

In the Design Editor, after you create a design, it shows the **Design Table** view of the design or other views if you previously viewed designs. In the context menu, available by right-clicking on the title bar, you can change the view of the design to 1-D, 2-D, 3-D, 4-D, and pairwise design projections, 2-D and 3-D constraint views, and the table view (also under the **View** menu). This menu also allows you to split the display either horizontally or vertically so that you simultaneously have two different views on the current design. You can also use the toolbar buttons to do this. The split can be merged again. After splitting, each view has the same functionality; that is, you can continue to split views until you have as many as you want. When you click a view, its title bar becomes blue to show it is the active view.

The currently available designs are displayed on the left in a tree structure. For details, see "The Design Tree".

Display Options

The Design Editor can display multiple design views at once, so while working on a design you can keep a table of design points open in one corner of the window, a 3-D projection of the constraints below it and a 2-D or 3-D plot of the current design points as the main plot.

The current view and options for the current view are available either through the context menu or the **View** menu on the Design Editor window.

1   Change the main display to 3-D Projection view.
2   You can rotate the projection with click-drag mouse movement. View your design in several projections (singly, or simultaneously by dividing the pane) by using the right-click context menu in the display pane.

See Also

Related Examples

*   "Design of Experiments" on page 8-2

- "Create a Constrained Space-Filling Design" on page 8-8
- "Create Optimal Designs" on page 8-14
- "Create Classical Designs" on page 8-28
- "Use the Prediction Error Variance Viewer" on page 8-22
- "Use the Design Evaluation Tool" on page 8-35

More About

- "Design of Experiments"

Use the Prediction Error Variance Viewer

In this section...
"Introducing the Prediction Error Variance Viewer" on page 8-22
"Add Points Optimally" on page 8-25

Introducing the Prediction Error Variance Viewer

A useful measure of the quality of a design is its prediction error variance (PEV). The PEV hypersurface is an indicator of how capable the design is in estimating the response in the underlying model. A bad design is either not able to fit the chosen model or is very poor at predicting the response. The Prediction Error Variance Viewer is only available for linear models. The Prediction Error Variance Viewer is not available when designs are rank deficient; that is, they do not contain enough points to fit the model. Optimal designs attempt to minimize the average PEV over the design region.

With an optimal design selected (as constructed in "Create Optimal Designs" on page 8-14), select **Tools > Prediction Error Variance Viewer**.
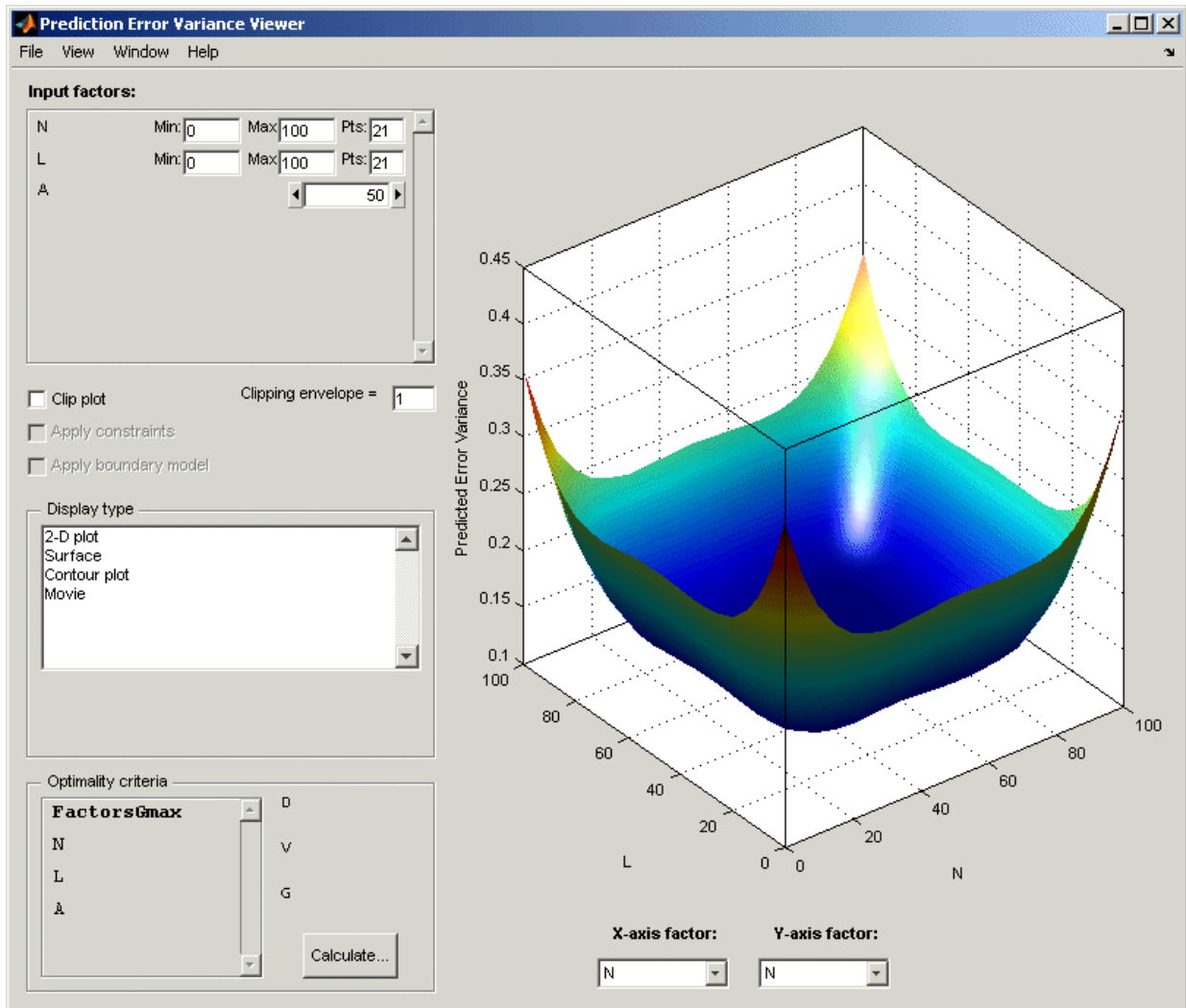
The default view is a 3-D plot of the PEV surface.

This shows where the response predictions are best. This example optimal design predicts well in the center and the middle of the faces (one factor high and the other midrange), but in the corners the design has the highest error. Look at the scale to see how much

difference there is between the areas of higher and lower error. For the best predictive power, you want low PEV (close to zero).

You can examine PEV for designs and models. The two are related in this way:

Accuracy of model predictions (model PEV)=Design PEV * MSE (Mean Square Error in measurements).

You can think of the design PEV as multiplying the errors in the data. The smaller the PEV, the greater the accuracy of your final model. You can read more about the calculation of PEV in "Prediction Error Variance".

Try the other display options.

- The **View** menu has many options to change the look of the plots.
- You can change the factors displayed in the 2-D and 3-D plots. The pop-up menus below the plot select the factors, while the unselected factors are held constant. You can change the values of the unselected factors using the buttons and edit boxes in the **Input factors** list, top left.
- The `Movie` option shows a sequence of surface plots as a third input factor's value is changed. You can change the factors, replay, and change the frame rate.
- You can change the number, position, and color of the contours on the contour plot with the **Contours** button, as shown.
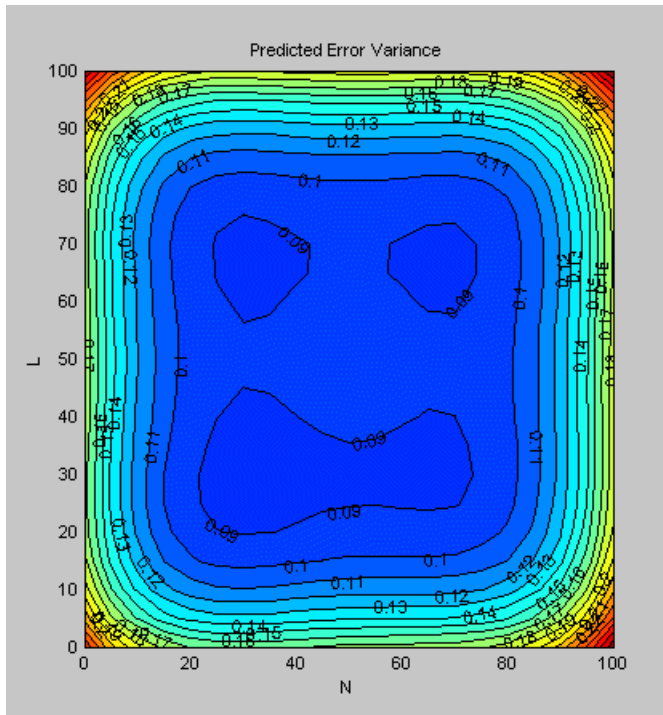
Add Points Optimally

You can further optimize the optimal design by returning to the Optimal Design dialog, where you can delete or add points optimally or at random. The most efficient way is to delete points *optimally* and add new points *randomly* — these are the default algorithm settings. Only the existing points need to be searched for the most optimal ones to delete (the least useful), but the entire candidate set has to be searched for points to add optimally.

To strengthen the current optimal design:

1   Return to the Design Editor window.

2   Click the Optimal Design button in the toolbar again to reenter the dialog, and add 60 more points. Keep the existing points (which is the default).

3   Click **OK** and watch the optimization progress, then click **Accept** when the number of iterations without improvement starts increasing.

4 View the improvements to the design in the main displays.

5 Once again select **Tools** > **Prediction Error Variance Viewer** and review the plots of prediction error variance and the new values of optimality criteria in the optimality frame (bottom left). The shape of the PEV projection might not change dramatically, but note the changes in the scales as the design improves. The values of D, V, and G optimality criteria will also change (you have to click **Calculate** to see the values).

To see more dramatic changes to the design, return to the Design Editor window (no need to close the Prediction Error Variance Viewer).

1 Split the display so you can see a 3-D projection at the same time as a Table view.

2 You can sort the points to make it easier to select points in one corner. For example, to pick points where N is 100 and L is 0,

   a Select **Edit** > **Sort Points**.

   b Choose to sort by N only (reduce the number of sort variables to one) and click **OK**.

3 Choose **Edit** > **Delete Point**.

4 Using the Table and 3-D views as a guide, in the Delete Points dialog, pick six points to remove along one corner. Add the relevant point numbers to the delete list by clicking the add (>) button.

5 Click **OK** to remove the points. See the changes in the main design displays and look at the new Surface plot in the Prediction Error Variance Viewer (see the example following).
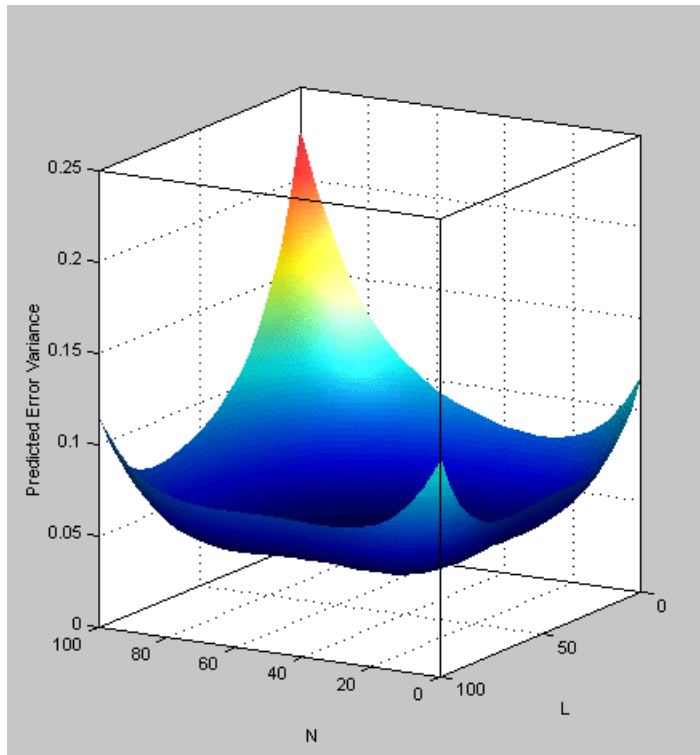
See Also

Related Examples
- "Design of Experiments" on page 8-2
- "Create a Constrained Space-Filling Design" on page 8-8
- "Create Optimal Designs" on page 8-14

More About
- "Design of Experiments"

Create Classical Designs

In this section...
"Adding a Classical Design" on page 8-28
"Classical Design Browser" on page 8-29
"Setting Up and Viewing a Classical Design" on page 8-30

Adding a Classical Design

Classical designs (including full factorial) are very well researched and are suitable for simple regions (hypercube or sphere).

1   In the Design Editor window, click to select the Optimal design in the design tree (as constructed in "Create Optimal Designs" on page 8-14).

2   Add a new design. Use the first toolbar button, or select **File > New**.

A new child node appears in the tree, called `Optimal_1`. Notice that the parent node now has a padlock on the icon. This indicates it is locked. This maintains the relationship between designs and their child nodes. The tree arrangement lets you try different operations starting from a basic design, then select the most appropriate one to use. The hierarchy allows clear viewing of the effects of changes on designs. The locking of parent designs also gives you the ability to easily reverse out of changes by retreating back up the tree.

3   Select the new design node in the tree. Notice that the display remains the same — all the points from the previous design remain, to be deleted or added to as necessary. The new design inherits all its initial settings from the currently selected design and becomes a child node of that design.

4   Rename the new node `Classical` by clicking again or by pressing **F2**.

5   Click the ▦ button in the toolbar or select **Design > Classical > Design Browser**.

Note In cases where the preferred type of classical design is known, you can go straight to one of the five options under **Design > Classical**. Choosing the **Design Browser** option allows you to see graphical previews of these same five options before making a choice.

A dialog appears because there are already points from the previous design. You must choose between replacing and adding to those points or keeping only fixed points from the design.

6 Choose the default, replace current points with a new design, and click **OK**.

The Classical Design Browser appears.

Classical Design Browser
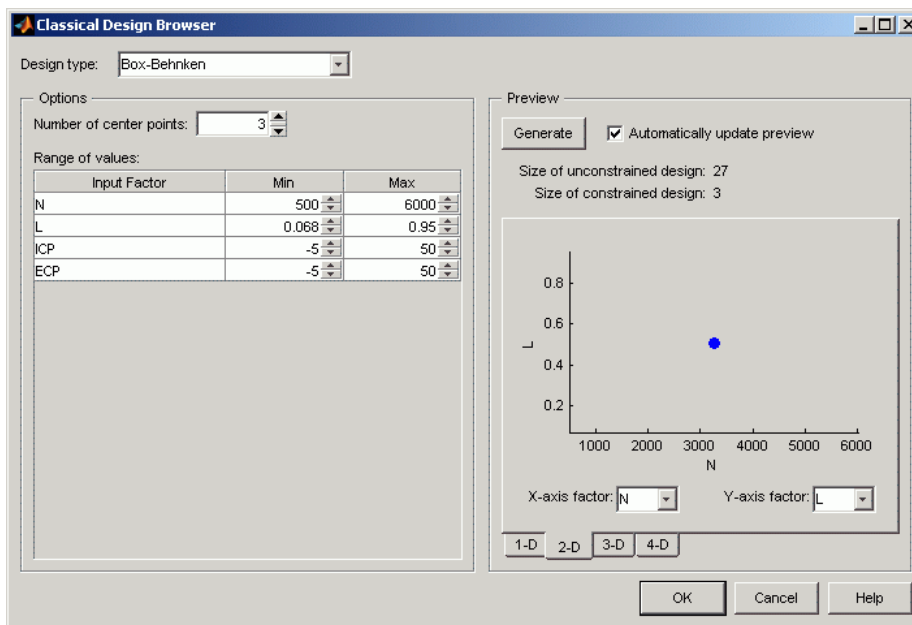
In the **Design Style** drop-down menu, there are five classical design options:

- `Central Composite`

    Generates a design that has a center point, a point at each of the design volume corners, and a point at the center of each of the design volume faces. You can choose a ratio value between the corner points and the face points for each factor and the number of center points to add. You can also specify a spherical design. Five levels are used for each factor.

- `Box-Behnken`

Similar to Central Composite designs, but only three levels per factor are required, and the design is always spherical in shape. All the design points (except the center point) lie on the same sphere, so there should be at least three to five runs at the center point. There are no face points. These designs are particularly suited to spherical regions, when prediction at the corners is not required.

- `Full Factorial`

  Generates an *n*-dimensional grid of points. You can choose the number of levels for each factor and the number of additional center points to add.

- `Plackett Burman`

  These are "screening" designs. They are two-level designs that are designed to allow you to work out which factors are contributing any effect to the model while using the minimum number of runs. For example, for a 30-factor problem this can be done with 32 runs.

- `Regular Simplex`

  These designs are generated by taking the vertices of a k-dimensional regular simplex (k = number of factors). For two factors a simplex is a triangle; for three it is a tetrahedron. Above that are hyperdimensional simplices. These are economical first-order designs that are a possible alternative to Plackett Burman or full factorials.

Setting Up and Viewing a Classical Design

1 Choose a `Box-Behnken` design.

2 Reduce the number of center points to 1.

3 View your design in different projections using the tabs under the display.

4 Click **OK** to return to the Design Editor.

5 Use the Prediction Error Variance Viewer to see how well this design performs compared to the optimal design created previously; see the following illustration.
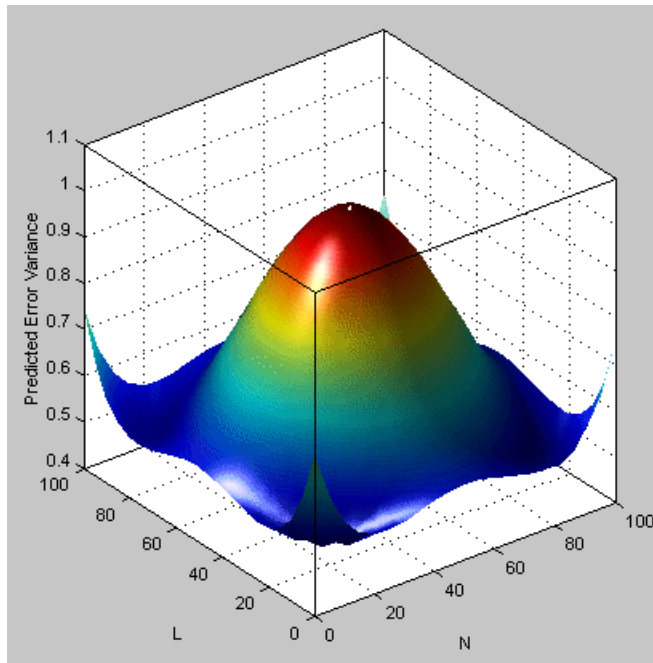
As you can see, this is not a realistic comparison, as this design has only 13 points (you can find this information in the bottom left of the main Design Editor display), whereas the previous optimal design had 100, but this is a good illustration of leverage. A single point in the center is very bad for the design, as illustrated in the Prediction Error Variance Viewer surface plot. This point is crucial and needs far more certainty for there to be any confidence in the design, as every other point lies on the edge of the space. This is also the case for Central Composite designs if you choose the spherical option. These are good designs for cases where you are not able to collect data points in the corners of the operating space.

If you look at the PEV surface plot, you should see a spot of white at the center. This is where the predicted error variance reaches 1. For surfaces that go above 1, the contour at 1 shows as a white line, as a useful visual guide to areas where prediction error is large.

1 Select Movie, and you see this white contour line as the surface moves through the plane of value 1.

2 Select the **Clip Plot** check box. Areas that move above the value of 1 are removed. The following example explains the controls.

Turn PEV value clipping on and off here

Change PEV clipping value here
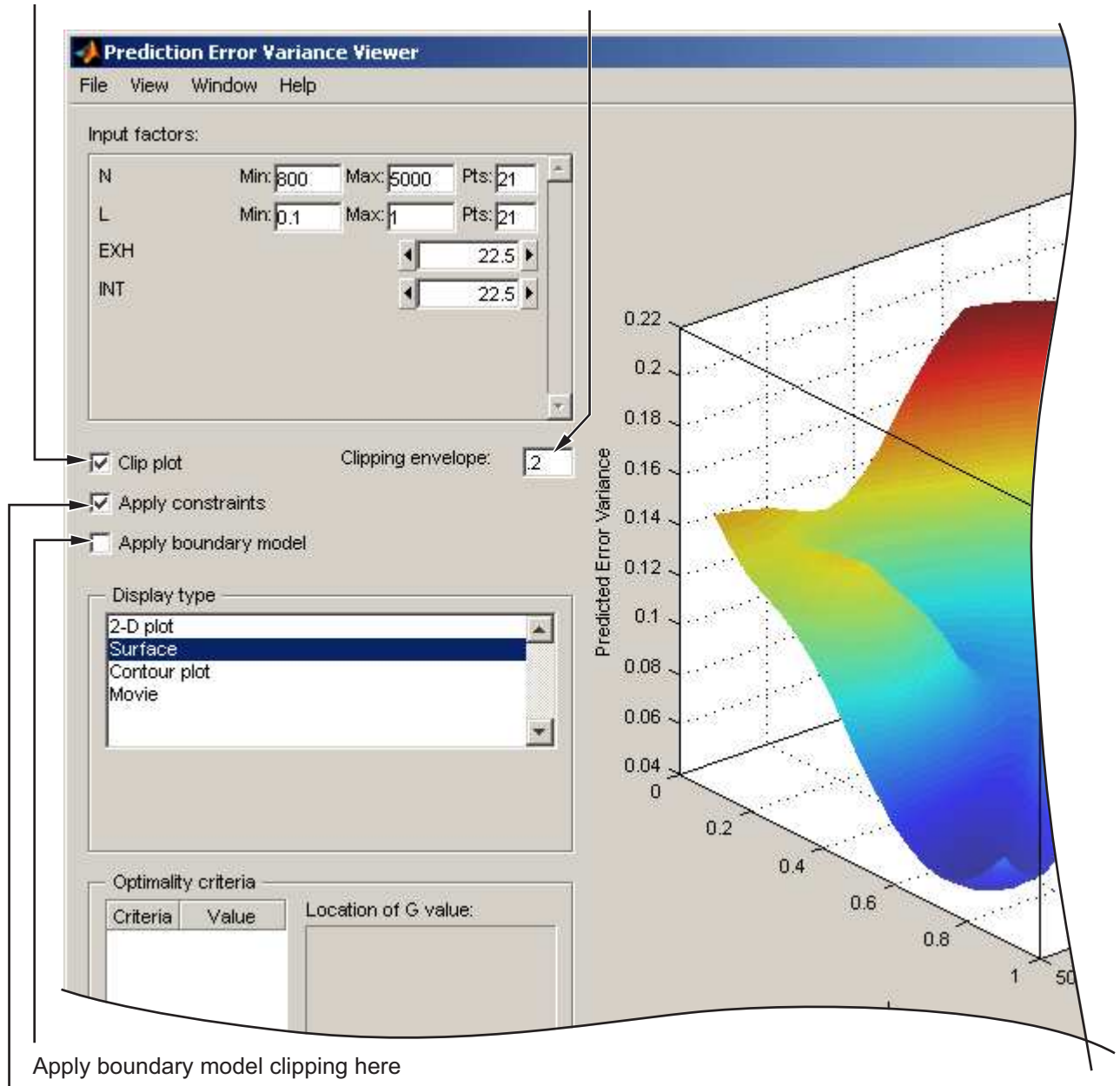
Apply boundary model clipping here

Apply design constraint clipping here

8-33

See Also

Related Examples

- "Design of Experiments" on page 8-2
- "Create a Constrained Space-Filling Design" on page 8-8
- "Create Optimal Designs" on page 8-14
- "View Design Displays" on page 8-20

More About

- "Design of Experiments"

Use the Design Evaluation Tool

The Design Evaluation Tool is available for linear models only. See also "Global Model Class: Multiple Linear Models".

1   In the Design Editor, after creating a classical design (in "Create Classical Designs" on page 8-28) select **Tools > Evaluate Designs**.

2   Choose the `Box-Behnken` design and click **OK** in the Select Designs dialog.

   The Design Evaluation Tool displays a large amount of statistical information about the design.

3   Select `Hat Matrix` from the list on the right.

4   Click the **Leverage Values** button.

   Note that the leverage of the central point is `1.00` (in red) and the leverage of all other points is less than this. The design would clearly be strengthened by the addition of more central points. Obviously, this is a special case, but for any kind of design, the Design Evaluation Tool is a powerful way to examine properties of designs.

5   Select `Design Matrix` from the list box.

6   Click the 3D Surface button in the toolbar.
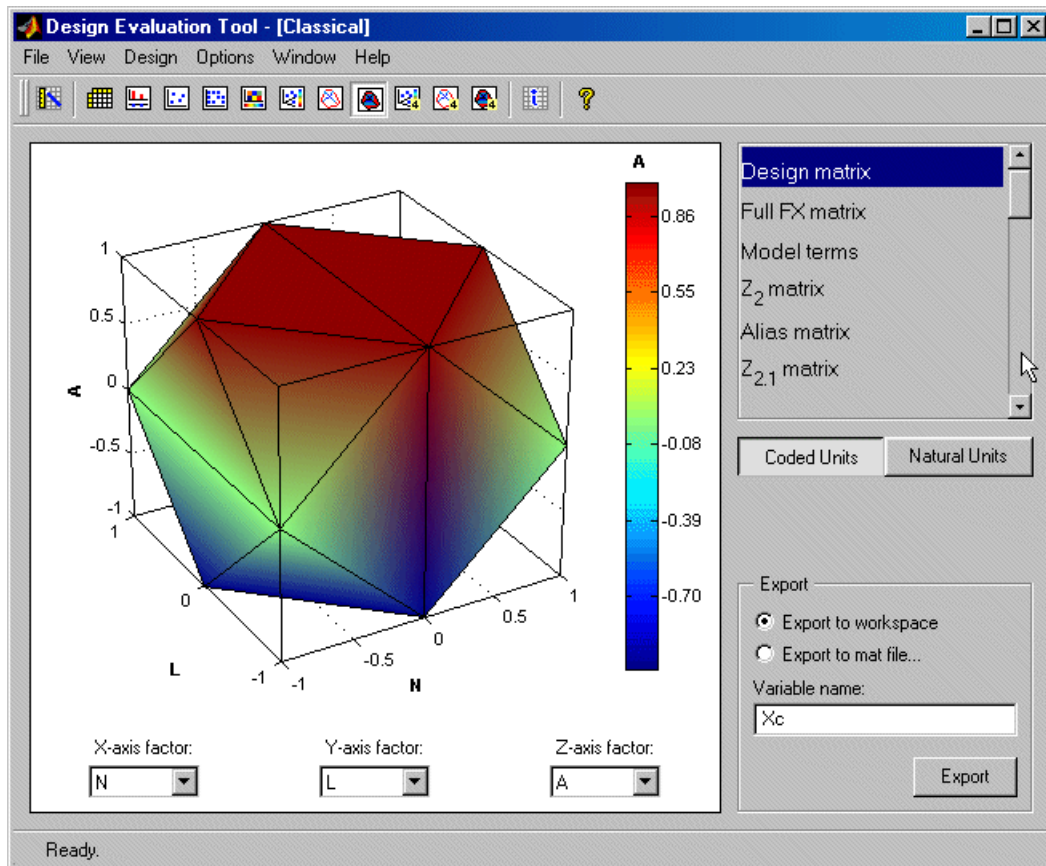
This illustrates the spherical nature of the current design. As usual, you can rotate the plot by clicking and dragging with the mouse.

There are many other display options to try in the toolbar, and in-depth details of the model terms and design matrices can all be viewed. You can export any of these to the workspace or a `.mat` file using the **Export** box.

For a description of all the information available here, see "Use the Design Evaluation Tool" on page 8-35.

Improving the Design

To strengthen the current Box-Behnken design near the center region:

1 Close the Design Evaluation Tool.

2 Return to the Design Editor window.

3 Select **Design > Classical > Box-Behnken**.

4 Click **OK** to replace the current points with a new design.

5 Increase the number of center points and click **OK**.

6 Once again select **Tools > Prediction Error Variance Viewer** and review the plots of prediction error variance and the new values of optimality criteria in the optimality frame (bottom left).

7 Review the leverage values of the center points. From the Design Editor window, use **Tools > Evaluate Design** and go to `Hat Matrix`.

8 Try other designs from the Classical Design Browser. Compare Full Factorial with Central Composite designs; try different options and use the Prediction Error Variance Viewer to choose the best design.

Note You cannot use the Prediction Error Variance Viewer if there are insufficient points in the design to fit the model. For example, you cannot fit a quadratic with less than three points, so the default Full Factorial design, with two levels for each factor, must be changed to three levels for every factor before you can use the Prediction Error Variance Viewer.

9 When you are satisfied, return to the Design Editor window and choose **Edit > Select as Best**. You will see that this design node is now highlighted in blue in the tree. This can be applied to any design.

When you are creating designs before you start modeling, the design that you select as best is the one used to collect data.

See Also

Related Examples

- "Design of Experiments" on page 8-2
- "Create a Constrained Space-Filling Design" on page 8-8
- "Create Optimal Designs" on page 8-14
- "View Design Displays" on page 8-20

•   "Create Classical Designs" on page 8-28

More About

•   "Design of Experiments"

Data Editor for Modeling

This section discusses the following topics:

- "Manipulate Data for Modeling" on page 9-2
- "Load Data" on page 9-3
- "View and Edit the Data" on page 9-5
- "Create New Variables and Filters" on page 9-9
- "Store and Import Variables, Filters, and Plot Preferences" on page 9-11
- "Define Test Groupings" on page 9-12
- "Match Data to Experimental Designs" on page 9-15

Manipulate Data for Modeling

For empirical engine modeling in the Model Browser, you first need to load, process and select data for modeling. This tutorial shows you how to use the Data Editor for loading data, creating new variables, and creating constraints for that data.

You can load data from files (Microsoft® Excel® files, MATLAB files, text files) and from the MATLAB workspace. You can merge data in any of these forms with previously loaded data sets (providing there is no conflict in the form of the data) to produce a new data set. Test plans can use only one data set, so the merging function allows you to combine records and variables from different files in one model.

You can define new variables, apply filters to remove unwanted data, and apply test notes to filtered tests. You can store and retrieve these user-defined variables and filters for any data set, and you can store plot settings. You can change and add records and apply test groupings, and you can match data to designs. You can also write your own data loading functions.

The following tutorial is a step-by-step guide. Follow the steps in these sections in order:

- "Load Data" on page 9-3
- "View and Edit the Data" on page 9-5
- "Create New Variables and Filters" on page 9-9
- "Store and Import Variables, Filters, and Plot Preferences" on page 9-11
- "Define Test Groupings" on page 9-12
- "Match Data to Experimental Designs" on page 9-15

For comprehensive help on using data in the Model Browser, see "Data Manipulation for Modeling".

Load Data

In this section...
"Opening the Data Editor" on page 9-3
"Loading a Data File" on page 9-3

Opening the Data Editor

You can create, copy, rename and delete data objects from the Project view in the Model Browser.

To enter the Data Editor and create a new data object, from the Project node, select **Data > New Data** (or click the New Data Object toolbar button).

The Data Editor appears.

There are no plots until some data has been loaded. The views shown depend on whether you have previously looked at the Data Editor, as it retains memory of your previous layout.

By default the new data object is called `Data Object`. Select **File > Rename Data** to enter a new name.

Loading a Data File

1   Click the import data from file button in the toolbar.

2   In the dialog box, find and select the `Holliday.xls` data file in the `matlab \toolbox\mbc\mbctraining` folder. Double-click to load the file, or select it and click **Open**.

   You can also enter file pathnames in the edit box. The list contains the file types recognized by the Model Browser (`Excel files`, `Delimited text files`, `MATLAB data files`, etc). Leave this at the default, and the toolbox tries to determine what type of file is selected by looking at the file extension.

3   In Data Variable Editor, edit the variables names and units if you want. Double-click or use the Edit or Remove buttons for selected variables. View the summary showing the total number of records and variables imported, and each variable's range, mean, standard deviation, and units. Click **OK**.

View the data you loaded in the Data Editor.

View and Edit the Data

In this section...
"Viewing Data" on page 9-5
"Using Notes to Sort Data for Plotting" on page 9-5
"Removing Outliers and Problem Tests" on page 9-6
"Reordering and Editing Data" on page 9-7

Viewing Data

You can split the views to display several plots at once. Use the right-click context menus, the toolbar buttons, or the **View** menu to split views. You can choose 2-D plots, 3-D plots, multiple data plots, data tables, and tabs showing summary, statistics, variables, filters, test filters, and test notes.

For more details about each view, see "View and Edit Data" in the *Model-Based Calibration Toolbox Model Browser User's Guide*.

You can use test notes to investigate problem data and decide whether some points should be removed before modeling. The following steps cover using notes and views to sort and investigate your data.

Using Notes to Sort Data for Plotting

1   Right-click a view and select **Current View > Multiple Data Plot**.

2   Right-click the new view and select **Add Plot**.

    The Plot Variables Setup dialog appears.

3   Select `spark` and click to add to the `X Variable` box, then select `tq` and click to add to the `Y Variable` box. Click **OK** to create the plot.

4   Click in the **Tests** list to select a test to plot (or **Shift**-click, **Ctrl**-click, or click and drag to select multiple tests).

5   Select **Tools > Test Notes**.

6   In The Test Note Editor, enter `mean(tq)<10` in the top edit box to define the tests to be noted, and enter `Low torque` in the Test Note edit box. Leave the note color at the default and click **OK**.

7   Click the Test Notes tab to view your note definition.

8   In the Test Selector pane on the left, observe that all the tests that satisfy the condition `mean(tq)<10` show `Low torque` next to them. Click the column header to sort the tests that meet the note condition to the top or bottom of the list.

9   Now create some more views.

- Right-click a view and select **Split View > Data Table**.

- Right-click a view and select **Split View > 3D Plot**.

10   In the Test Selector pane, click particular tests with the `Low torque` note.

Notice that when you select a test here, the same test is plotted in the multiple data plots, the 3D data plot, and highlighted in the data table. You can use the notes in this way to easily identify problem tests and decide whether you should remove them.

Removing Outliers and Problem Tests

1   Click a point on the **Multiple Data Plots** view.

The point is outlined in red on the plot, and highlighted in the data table. You can remove points you have selected as outliers by selecting **Tools > Remove Data** (or use the keyboard shortcut **Ctrl+A**). Select **Tools > Restore Data** (or use the keyboard shortcut **Ctrl+Z**) to open a dialog where you can choose to restore any or all removed points.

You can remove individual points as outliers, or you can remove records or entire tests with filters.

2   For example, after examining all the `Low torque` noted tests, you could decide they should be filtered out.

a   Select **Tools > Test Filters**.

b   In the Test Filter Editor, enter `mean(tq)>10` to keep all tests where the mean torque is greater than 10, and click **OK**.

c   Click the Test Filters tab and observe the new test filter results show it is successfully applied and the number of records removed.

Similarly, you can use filters to remove individual records rather than entire tests, which you will cover in a later section "Applying a Filter" on page 9-9.

3   To view removed data in the table view, right-click and select **Allow Editing**. Removed records are red. To view removed data in the 2-D and Multiple Data Plots, select **Properties** and select the box **Show removed data**.

Reordering and Editing Data

To change the display, right-click a 2-D plot and select **Properties**. You can alter grid and plot settings including lines to join the data points.

Reorder X Data in the Plot Properties dialog can be useful when record order does not produce a sensible line joining the data points. For an illustration of this:

1   Ensure you are displaying a 2-D plot. You can right-click on any plot and select **Current Plot > 2-D Plot**, or use the context menu split commands to add new views.

2   Right-click on a 2-D plot and select **Properties** and choose `solid` from the **Data Linestyle** drop-down menu, as shown below. Click **OK**.
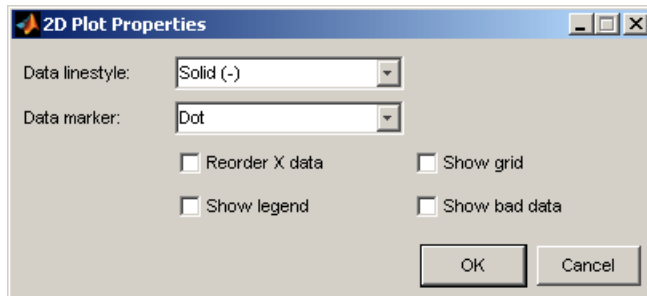
3   Choose `afr` for the *y*-axis.

4   Choose `Load` for the *x*-axis.

5   Select test 590. You must use the test controls contained within the 2-D plot. The **Tests** pane on the left applies to other views: tables and 3-D and multiple data plots.

9-7
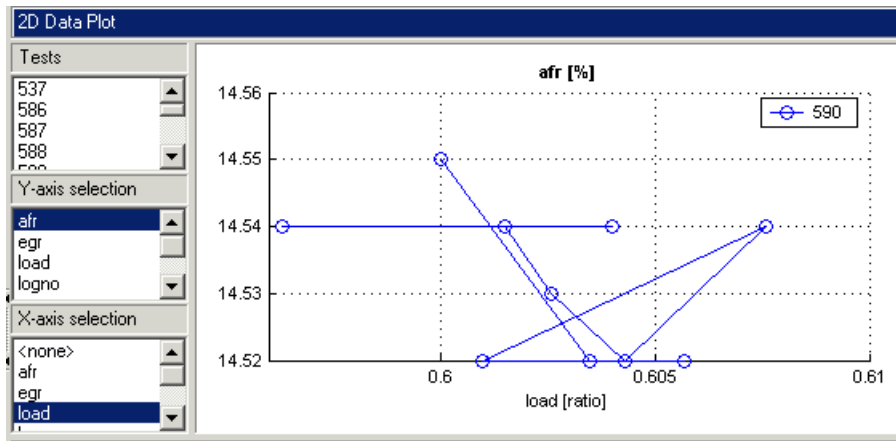
6   Right-click and select **Properties** and choose **Reorder X Data**. Click **OK**.

    This command replots the line from left to right instead of in the order of the records, as shown.
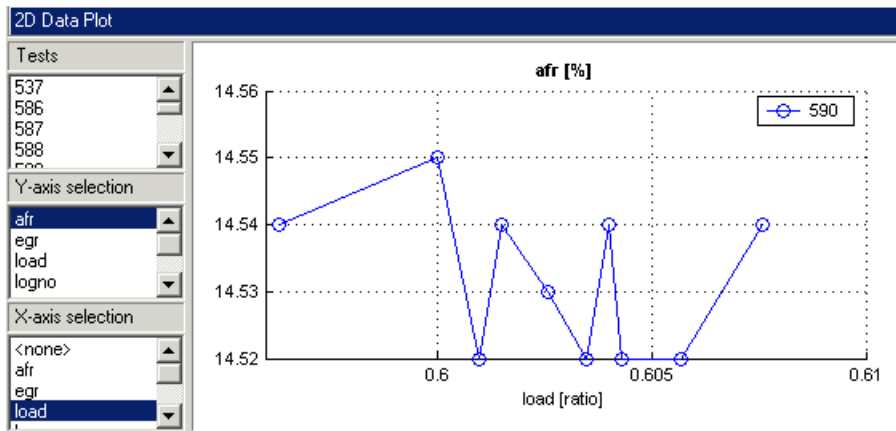
7   Right-click and select **Split Plot > Data Table** to split the currently selected view and add a table view. You can select particular test numbers in the **Tests** pane on the left of the Data Editor. You can right-click to select **Allow Editing**, and then you can double-click cells to edit them.

Create New Variables and Filters

In this section...

Adding New Variables

You can add new variables to the data set.

1   Select **Tools > Variables**, or click the toolbar button.

In the Variable Editor, you can define new variables in terms of existing variables. Define the new variable by writing an equation in the edit box at the top.

2   Define a new variable called `POWER` that is defined as the product of two existing variables, `tq` and `n`, by entering `POWER=tq*n`, as seen in the example following. You can also double-click variable names and operators to add them, which can be useful to avoid typing mistakes in variable names, which must be exact including case.

3   Click **OK** to add this variable to the current data set.

4   View the new variable in the Data Editor in the Variables tab at the top. You can also now see `7 + 1 variables` in the summary tab.

Applying a Filter

A filter is a constraint on the data set you can use to exclude some records. You use the Filter Editor to create new filters.

1   Choose **Tools > Filters**, or click the toolbar button.

In the Filter Editor, define the filter using logical operators on the existing variables.

2   Keep all records with speed (`n`) greater than 1000. Type `n` (or double-click on the variable `n`), then type `>1000`.

3   Click **OK** to impose this filter on the current data set.

4   View the new filter in the Data Editor in the Filters tab at the top. Here you can see a list of your user-defined filters and how many records are removed by the new filter.

You can also now see `141/270 records` in the summary tab and a red section in the bar illustrating the records removed by the filter.

Sequence of Variables

You can change the order of user-defined variables in the Variable Editor list using the up and down arrow buttons.

Select **Tools > Variables**.

Example:

1   Define two new variables, `New1` and `New2`. Note that you can use the buttons to add or remove a list item to create or delete variables in this view. Click the button to 'Add item' to add a new variable, and enter the definitions shown.

Notice that `New2` is defined in terms of `New1`. New variables are added to the data in turn and hence `New1` must appear in the list before `New2`, otherwise `New2` is not well defined.

2   Change the order by clicking the down arrow in the Variable Editor to produce this erroneous situation. Click **OK** to return to the Data Editor and in the Variables tab you can see the error message that there is a problem with the variable.

3   Use the arrows to order user-defined variables in legitimate sequence.

Deleting and Editing Variables and Filters

You can delete user-defined variables and filters.

Example:

1   To delete the added variable `New1`, select it in the Variables tab and press the **Delete** key.

2   You can also delete variables in the Variable Editor by clicking the Remove Item button.

Similarly, you can delete filters by selecting the unwanted filter in the Filters tab and using the **Delete** key.

Store and Import Variables, Filters, and Plot Preferences

You can store and import plot preferences, user-defined variables, filters, and test notes so they can be applied to other data sets loaded later in the session, and to other sessions.

- Select **Tools > Import Expressions**
- Click the toolbar button

The Data Editor remembers your plot type settings and when reopened will display the same types of views. You can also store your plot layouts to save the details of your Multiple Data Plots, such as which factors to display, line style, grid, etc.

In the Import Variables, Filters, and Editor Layout dialog box, use the toolbar buttons to import variables, filters and plot layouts. Import from other data sets in the current project, or from MBC project files, or from files exported from the Data Editor.

To use imported expressions in your current project, select items in the lists and click the toolbar button to apply in the data editor.

To store expressions in a file, in the Data Editor, select **Tools > Export Expressions** and select a file name.

For more, see "Using Data".

Define Test Groupings

The Define Test Groupings dialog collects records of the current data object into groups; these groups are referred to as *tests*.

The dialog is accessed from the Data Editor in either of these ways:

- Using the menu **Tools > Test Groups**

- Using the toolbar button

When you enter the dialog using the `holliday` data, a plot is displayed as the variable `logno` is automatically selected for grouping tests.

Select another variable to use in defining groups within the data.

1 Select `n` in the **Variables** list.

2 Click the button to add the variable (or double-click `n`).

   The variable `n` appears in the list view on the left. You can now use this variable to define groups in the data. The maximum and minimum values of `n` are displayed. The **Tolerance** is used to define groups: on reading through the data, when the value of `n` changes by more than the tolerance, a new group is defined. You change the **Tolerance** by typing directly in the edit box.

   You can define additional groups by selecting another variable and choosing a tolerance. Data records are then grouped by `n` or by this additional variable changing outside their tolerances.

3 Clear the box **Group by** for `logno`. Notice that variables can be plotted without being used to define groups.

4 Add `load` to the list by selecting it on the right and clicking .

5 Change the `load` tolerance to 0.01 and watch the test grouping change in the plot.

6 Clear the **Group By** check box for `load`. Now this variable is plotted without being used to define groups.

   The plot shows the scaled values of all variables in the list view (the color of the tolerance text corresponds to the color of data points in the plot). Vertical pink bars show the tests (groups). You can zoom the plot by **Shift**-click-dragging or middle-click-dragging the mouse; zoom out again by double-clicking.

7 Select `load` in the list view (it becomes highlighted in blue) and remove it from the list by clicking the ⊞ button.

8 Double-click to add `spark` to the list, and clear the **Group By** check box. Select `logno` as the only grouping variable.

It can be helpful to plot the local model variable (in this case spark) to check you have the correct test groupings. The plot shows the sweeps of spark values in each test while speed (n) is kept constant. Speed is only changed between tests, so it is a global variable. Try zooming in on the plot to inspect the test groups; double-click to reset.

One-stage data defines one test per record, regardless of any other grouping. This is required if the data is to be used in creating one-stage models.

Sort records before grouping allows you to reorder records in the data set. Otherwise, the groups are defined using the order of records in the original data object.

Show original test groups displays the original test groupings if any were defined.

Test number variable contains a list of all the variables in the current data set. Any of these could be selected to number the tests.

9 Make sure `logno` is selected for the **Test number variable**.

This changes how the tests are displayed in the rest of the Model Browser. Test number can be a useful variable for identifying individual tests in Model Browser and Data Editor views (instead of 1,2,3...) if the data was taken in numbered tests and you want access to that information during modeling.

If you chose `none` from the **Test number variable** list, the tests would be numbered 1,2,3 and so on in the order in which the records appear in the data file. With `logno` chosen, you will see tests in the Data Editor listed as 586, 587 etc.

Every record in a test must share the same test number to identify it, so when you are using a variable to number tests, the value of that variable is taken in the first record in each test.

Test numbers must be unique, so if any values in the chosen variable are the same, they are assigned new test numbers for the purposes of modeling (this does not change the underlying data, which retains the correct test number or other variable).

10 Click **OK** to accept the test groupings defined and close the dialog.

In the Data Editor summary tab, view the number of tests.

The number of records shows the number of values left (after filtration) of each variable in this data set, followed by the original number of records. The color coded bars also display the number of records removed as a proportion of the total number. The values are collected into a number of tests; this number is also displayed. The variables show the original number of variables plus user-defined variables.

Match Data to Experimental Designs

In this section...
"Introducing Matching Data to Designs" on page 9-15
"Tolerances and Cluster Information" on page 9-17
"Understanding Clusters" on page 9-19

Introducing Matching Data to Designs

You can use an example project to illustrate the process of matching experimental data to designs.

Experimental data is unlikely to be identical to the desired design points. You can use the Design Match view in the Data Editor to compare the actual data collected with your experimental design points. Here you can select data for modeling. If you are interested in collecting more data, you can update your experimental design by matching data to design points to reflect the actual data collected. You can then optimally augment your design (using the Design Editor) to decide which data points it would be most useful to collect, based on the data obtained so far.

You can use an iterative process: make a design, collect some data, match that data with your design points, modify your design accordingly, then collect more data, and so on. You can use this process to optimize your data collection process in order to obtain the most robust models possible with the minimum amount of data.

1   To see the data matching functions, in the Model Browser, select **File > Open Project** and browse to the file `Data_Matching.mat` in the in the `matlab\toolbox\mbc\mbctraining` directory.

2   Click the `Spark Sweeps` node in the model tree to change to the test plan view.

   Here you can see the two-stage test plan with model types and inputs set up. The global model has an associated experimental design (which you could view in the Design Editor). You are going to use the Data Editor to examine how closely the data collected so far matches to the experimental design.

3   Click the Edit Data button ( ) in the toolbar.

   The Data Editor appears.

4 You need a **Design Match** view to examine design and data points. Right-click a view in the Data Editor and select **Current View > Design Match**.

In the Design Match you can see colored areas containing points. These are "clusters" where closely matching design and data points have been selected by the matching algorithm.

Tolerance values (derived initially from a proportion of the ranges of the variables) are used to determine if any data points lie within tolerance of each design point. Data points that lie within tolerance of any design point are matched to that cluster. Data points that fall inside the tolerance of more than one design point form a single cluster containing all those design and data points. If no data points lie within tolerance of a design point, it remains unmatched and no cluster is plotted.
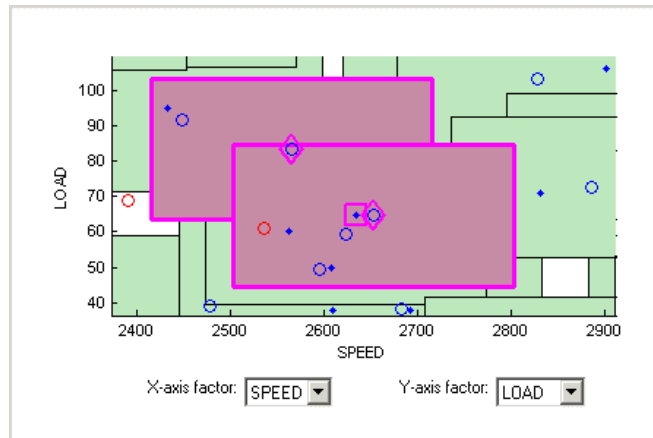
Notice the shape formed by overlapping clusters. The example shown outlined in pink is a single cluster formed where a data point lies within tolerance of two design points.

Note that on this plot you can see other unselected points that appear to be contained within this cluster. You need to track points through other factor dimensions using the axis controls to see where points are separated beyond tolerance. You will do this in a later step of this tutorial, "Understanding Clusters" on page 9-19.

Tolerances and Cluster Information

1 To edit tolerance values, select **Tolerances** in the context menu.

The Tolerance Editor appears. Here you can change the size of clusters in each dimension. Observe that the LOAD tolerance value is currently 100. This accounts for the elongated shape (in the LOAD dimension) of the clusters in the current plot, because this tolerance value is a high proportion of the total range of this variable.
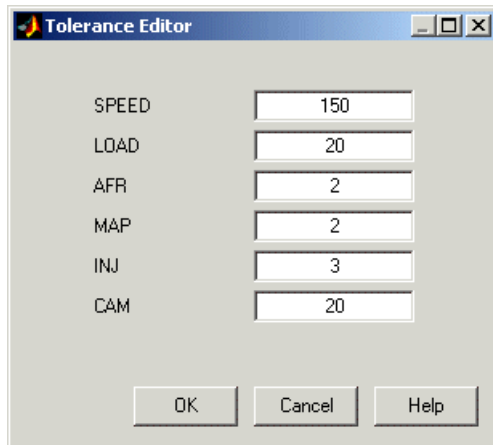
2 Click the **LOAD** edit box and enter 20, as shown. Click **OK**.

Notice the change in shape of the clusters in the **Design Match** view.
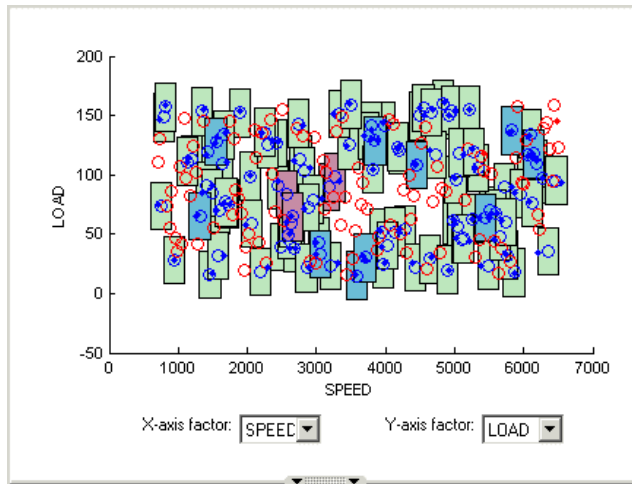
3 Shift click (center-click) and drag to zoom in on an area of the plot, as shown. You can double-click to return to the full size plot.
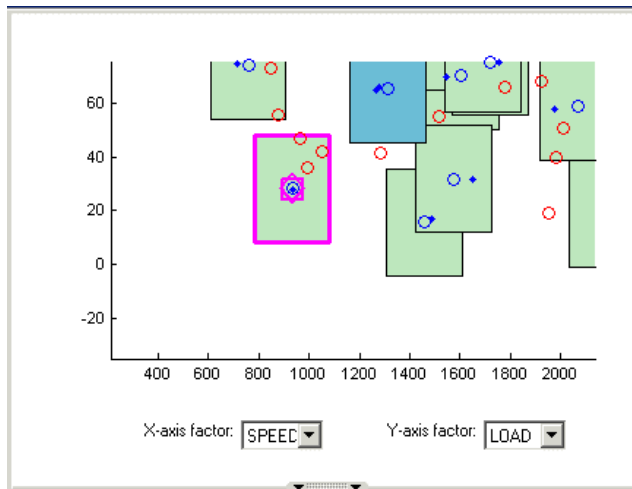
4 Click a cluster to select it. Selected points or clusters are outlined in pink. If you click and hold, you can inspect the values of global variables at the selected points (or for all data and design points if you click on a cluster). You can use this information to help you decide on suitable tolerance values if you are trying to match points.

5 Notice that the **Cluster Information** list shows the details of all data and design points contained in the selected cluster. You use the check boxes here to select or exclude data or design points. Click different clusters to see a variety of points. The list shows the values of global variables at each point, and which data and design points are within tolerance of each other. Your selections here determine which data will be used for modeling, and which design points will be replaced by actual data points.

> **Note** All data points with a selected check box will be used for modeling. All data points with a cleared check box will be removed from the data set, and not seen in any other views. This design match cluster view is the only place you can restore these excluded data to the data set.

Understanding Clusters

If you are not interested in collecting more data, then there is no need to make sure the design is modified to reflect the actual data. All data (except those you exclude by clearing the check boxes) will be used for modeling.

However, if you want your new design (called `Actual Design`) to accurately reflect what data has been obtained so far, for example to collect more data, then the cluster matching is important. All data points with a selected check box will be added to the new `Actual Design`, except those in red clusters. The color of clusters indicates what proportion of selected points it contains as follows:

- Green clusters have equal numbers of selected design and selected data points. The data points will replace the design points in the `Actual Design`.

  Note that the color of all clusters is determined by the proportion of *selected* points they contain; excluded points (with cleared check boxes) have no effect. Your check box selections can change cluster color.

- Blue clusters have more data points than design points. All the data points will replace the design points in the `Actual Design`.

- Red clusters have more design points than data points. These data points will not be added to your design as the algorithm cannot choose which design points to replace, so you must manually make selections to deal with red clusters if you want to use these data points in your design.

If you don't care about the `Actual Design` (for example, if you do not intend to collect more data) and you are just selecting data for modeling, then you can ignore red clusters. The data points in red clusters are selected for modeling.

1   Right-click the **Design Match** and select **Select Unmatched Data**. Notice that the remaining unmatched data points appear in the list. Here you can use the check boxes to select or exclude unmatched data in the same way as points within clusters.

2   Select a cluster, then use the drop-down menu to change the **Y-Axis factor** to `INJ`. Observe the selected cluster now plotted in the new factor dimensions of `SPEED` and `INJ`.

You can use this method to track points and clusters through the dimensions. This can give you a good idea of which tolerances to change in order to get points matched. Remember that points that do not form a cluster may appear to be perfectly matched when viewed in one pair of dimensions; you must view them in other dimensions to find out where they are separated beyond the tolerance value. You can use this tracking process to decide whether you want particular pairs of points to be matched, and then change the tolerances until they form part of a cluster.

3   Clear the **Equal Data and Design** check box in the **Design Match** view. You control what is plotted using these check boxes.
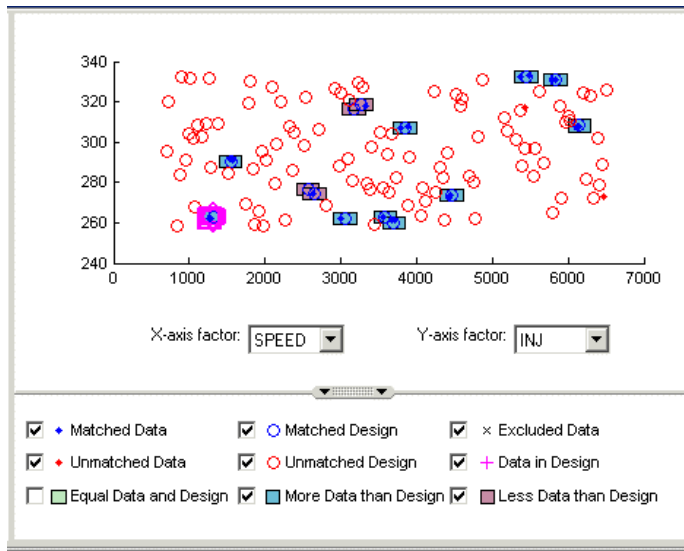
This removes the green clusters from view, as shown. These clusters are matched; you are more likely to be interested in unmatched points and clusters with uneven numbers of data and design points. Removing the green clusters allows you to focus on these points of interest. If you want your new `Actual Design` to accurately reflect your current data, your aim is to get as many data points matched up to design points as possible; that is, as few red clusters as possible.

4   Clear the check box for **More Data than Design**. You may also decide to ignore blue clusters, which contain more data points than design points. These design points will be replaced by all data points within the cluster. An excess of data points is unlikely to be a concern.

However, blue clusters may indicate that there was a problem with the data collection at that point, and you may want to investigate why more points than expected were collected.
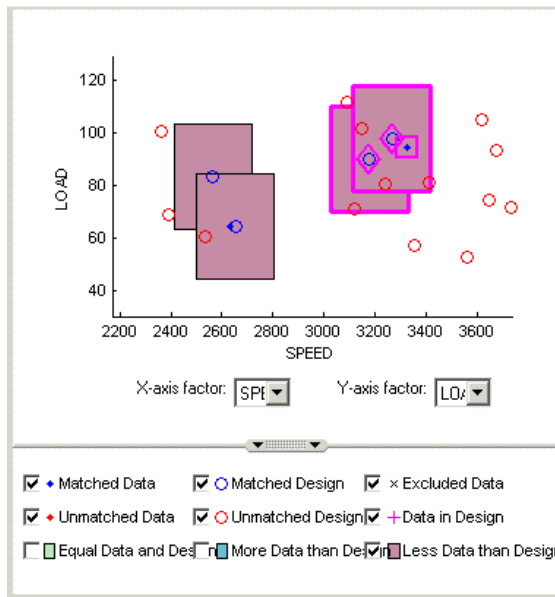
5   Select one of the remaining red clusters. Both of these have two design points within tolerance of a single data point.

6   Choose one of the design points to match to the data point, then clear the check box of the other design point. The cleared design point remains unchanged in the design. The selected design point will be replaced by the matched data point.

Notice that the red cluster disappears. This is because your selection results in a cluster with an equal number of selected data and design points (a green cluster) and your current plot does not display green clusters.

7 Repeat for the other red cluster.

Now all clusters are green or blue. There are two remaining unmatched data points.

8 Clear the **Unmatched Design** check box to locate the unmatched data points. Select **Unmatched Design** check box again — you need to see design points to decide if any are close enough to the data points that they should be matched.

9 Locate and zoom in on an unmatched data point. Select the unmatched data point and a nearby design point by clicking, then use the axis drop-down menus to track the candidate pair through the dimensions. Decide if any design points are close enough to warrant changing the tolerance values to match the point with a design point.

10 Recall that you can right-click the **Design Match** and select **Select Unmatched Data** to display the remaining unmatched data points in the **Cluster Information** list. Here you can use the check boxes to select or exclude these points. If you leave them selected, they will be added to the `Actual Design`.

These steps illustrate the process of matching data to designs, to select modeling data and to augment your design based on actual data obtained. Some trial and error is necessary to find useful tolerance values. You can select points and change plot dimensions to help you find suitable values. If you want your new `Actual Design` to accurately reflect your experimental data, you need to make choices to deal with red clusters. Select which design points in red clusters you want to replace with the data points. If you do not, then these data points will not be added to the new design.

When you are satisfied that you have selected all the data you want for modeling, close the Data Editor. At this point, your choices in the Design Match view will be applied to the data set and a new design called `Actual Design` will be created. All the changes are determined by your check box selections for data and design points.

All data points with a selected check box are selected for modeling. Data points with cleared check boxes are excluded from the data set. Changes are made to the existing design to produce the new `Actual Design`. All selected data will be added to your new design, except those in red clusters. Selected data points that have been matched to design points (in green and blue clusters) replace those design points.

All these selected data points become fixed design points (red in the Design Editor) and appear as `Data in Design` (pink crosses) when you reopen the Data Editor.

This means these points will not be included in clusters when matching again. These fixed points will also not be changed in the Design Editor when you add points, though you can unlock fixed points if you want. This can be very useful if you want to optimally augment a design, taking into account the data you have already obtained.

See "Match Data to Designs" for more information.

For more information on all aspects of data handling for modeling, see "Data Import and Processing".

10

Feature Calibration

This section includes the following topics:

Feature Calibration

What Are Feature Calibrations?

The feature calibration process within the Model-Based Calibration Toolbox product calibrates an estimator, or feature, for a control subsystem in an electronic control unit (ECU). These features are usually algebraic collections of one or more tables. You use the features to estimate signals in the engine that are unmeasurable, or expensive to measure, and are important for engine control. The toolbox can calibrate the ECU subsystem by directly comparing it with a plant model of the same feature.

There are advantages to feature calibration compared with simply calibrating using experimental data. Data is noisy (that is, there is measurement error) and this can be smoothed by modeling; also models can make predictions for areas where you have no data. This means you can calibrate more accurately while reducing the time and effort required for gathering experimental data.

An example of an ECU subsystem control feature estimates the value of torque, depending on the four inputs: speed, load, air/fuel ratio (AFR), and spark angle.

A diagram of this ECU subsystem example follows.

Plant Model for Torque = TQ(speed, load, AFR, spark)
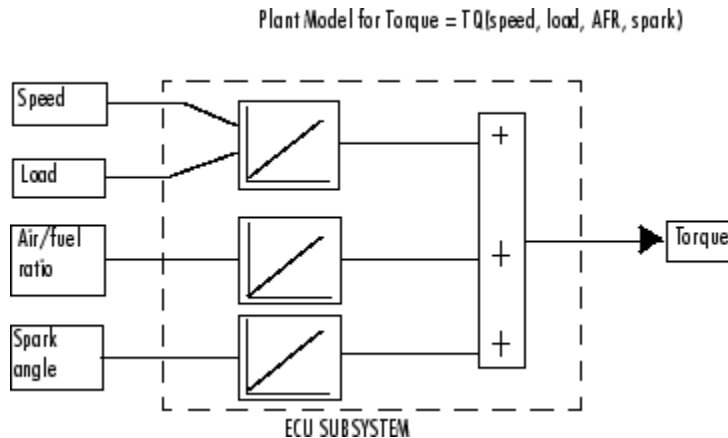
In this tutorial example, there are three lookup tables:

- A speed-load table
- A modifier, or table, for AFR
- A modifier for spark angle

This tutorial takes you through the various steps required to set up this feature and then calibrate it using CAGE. You will use CAGE to fill the tables by comparing them with a torque engine model.

The model is a copy of the torque model built in the Model Browser's Quick Start tutorial using engine data. This illustrates how you can use the Model-Based Calibration Toolbox product to map engine behavior and transfer this information to engine calibrations. You can construct a model using the Model Browser; then you can use CAGE to calibrate lookup tables by reference to the model.

Start CAGE

Start CAGE by typing

`cage`

at the MATLAB prompt.

Note If you have a CAGE session open, select **File > New > Project**.

Before you can perform a calibration, you must set up the variable dictionary and models that you want to use.

Set Up Variables

To set up the variables and constants that you want to use in your calibration,

1   Click **Variable Dictionary** in the **Data Objects** pane of CAGE.
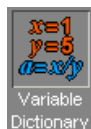
The **Variable Dictionary** view displays all the variables, constants, and formulas in a session. This is empty until you add some variable items.

There are two ways in which you can set up variables:

- Import a variable dictionary
- Add variables and constants to your session

After setting up your variables and constants, you can export the variable dictionary to use in other calibrations.

Importing a Variable Dictionary

To import a variable dictionary,

1   Select **File > Import > Variable Dictionary**.

2   Select the `tutorial.xml` file found in `matlab\toolbox\mbc\mbctraining` and click **Open**. CAGE automatically switches to the Variable Dictionary view.

This imports a set of variables and a constant. In this example, the variable dictionary contains

- The stoichiometric constant, `stoich`
- `N`, engine speed
- `L`, load
- `A`, AFR

Your display should resemble the following.
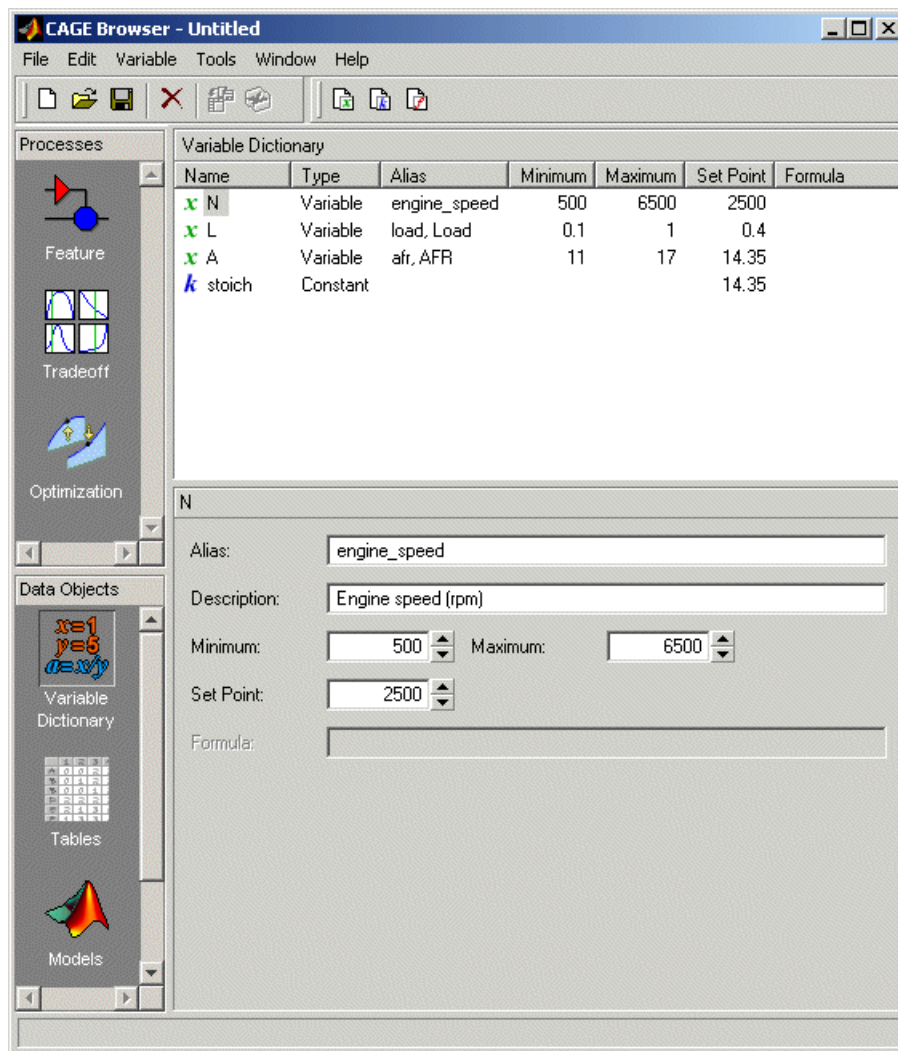
Adding and Editing Variables and Constants

To add a variable for the spark angle,

1 Click New Variable ⬚ in the toolbar. This adds a new variable to your dictionary.

2 Right-click the new variable and select **Rename** (or press **F2**) to rename the variable.

3 Enter SPK as the name.

4 Set the range of the variable by entering -5 as the **Minimum** and 50 as the **Maximum**.

The variable dictionary enables you to specify different names for the same variable, and also give descriptions of variables. For example, the variable spk might be referred to as S or spark in other models.

To ensure that CAGE recognizes an instance of S or spark as the same as spk, specify the aliases of SPK:

1 Enter S, spark in the **Alias** edit box.

2 Enter Spark advance (deg) in the **Description** edit box.

> **Note** The **Variable Dictionary** is case sensitive: s and S are different.

The variable dictionary enables you to specify a preferred value for a variable. For example, in the preferred value of the variable, AFR is set as the stoichiometric constant 14.35.

1 Select SPK and enter 25 in the **Set Point** edit box to specify the preferred value.

Set Up Models

A model in the Model-Based Calibration Toolbox product is a function of a set of variables. Typically, you construct a model using the Model Browser; then you can use CAGE to calibrate lookup tables by reference to the model.

The following example uses a model of the behavior of torque with varying spark angle, air/fuel ratio, engine speed, and load.

Importing a Model

To import a model built using the Model Browser,

1 Select **File** > **Import** > **Model**, which opens a file browser.

2   Browse to `matlab\toolbox\mbc\mbctraining`, select the `tutorial.exm` file (this is a copy of the torque model built in the Model Browser's Empirical Engine Modeling tutorial), and click **Open**. The Model Import Wizard appears.

3   There are two models stored in this file, `tq` and `knot`. Highlight `tq` and select the check box to **Automatically assign/create inputs**.

CAGE automatically assigns variables in the variable dictionary to the model input factors or their aliases (as long as names are exact). If names are not exact you can select variables manually using the wizard.

4   Click **Finish** to complete the wizard.

CAGE switches to the **Models** view, as shown following.
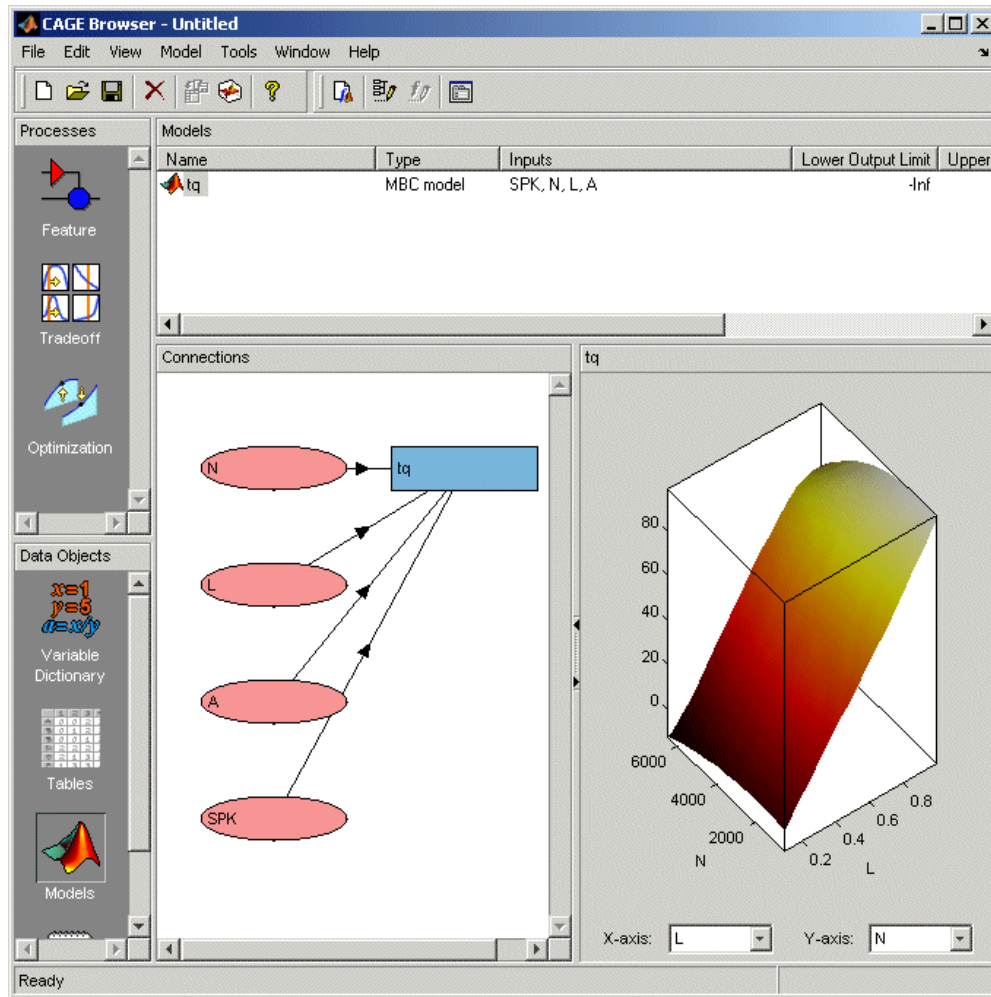
For more information about models, see "Setting Up Models" in the CAGE documentation.

Set Up a New Feature

The feature calibration process calibrates an algebraic collection of lookup tables, or *strategy*, by comparing the tables to the model.

When you have set up the variables and models, you can set up the feature as follows:

1   Select **File** > **New** > **Feature**.

CAGE automatically displays the **Feature** view and creates a new feature.

2   Select **Feature** > **Select Filling Item**. This opens the Select Filling Item dialog box. Select `tq` (currently the only model in your project) and click **OK**.

3   Create a strategy. For instructions, see the next section, "Set Up the Strategy" on page 10-9.

A strategy is a collection of tables. The Model-Based Calibration Toolbox product uses Simulink software to enable you to graphically specify the collection of tables for a feature.

4   After you have created a strategy, the next step is to set up your tables. For more information, see the section, "Set Up the Tables" on page 10-11.

Set Up the Strategy

The toolbox uses Simulink to graphically specify the strategy.

Importing a Strategy

To import a strategy,

1   Select **File** > **Import** > **Strategy**.

2   Select the model file called `tutorial`, found in `matlab\toolbox\mbc\mbctraining`, and click **Open**.

CAGE imports the strategy.

3   To view the strategy, select **Feature** > **Graphical Strategy Editor**.

This opens the `Torque_Output` strategy in a Simulink window, and also the libraries for your project and other blocks for use with CAGE.

View the `Torque_Output` strategy.
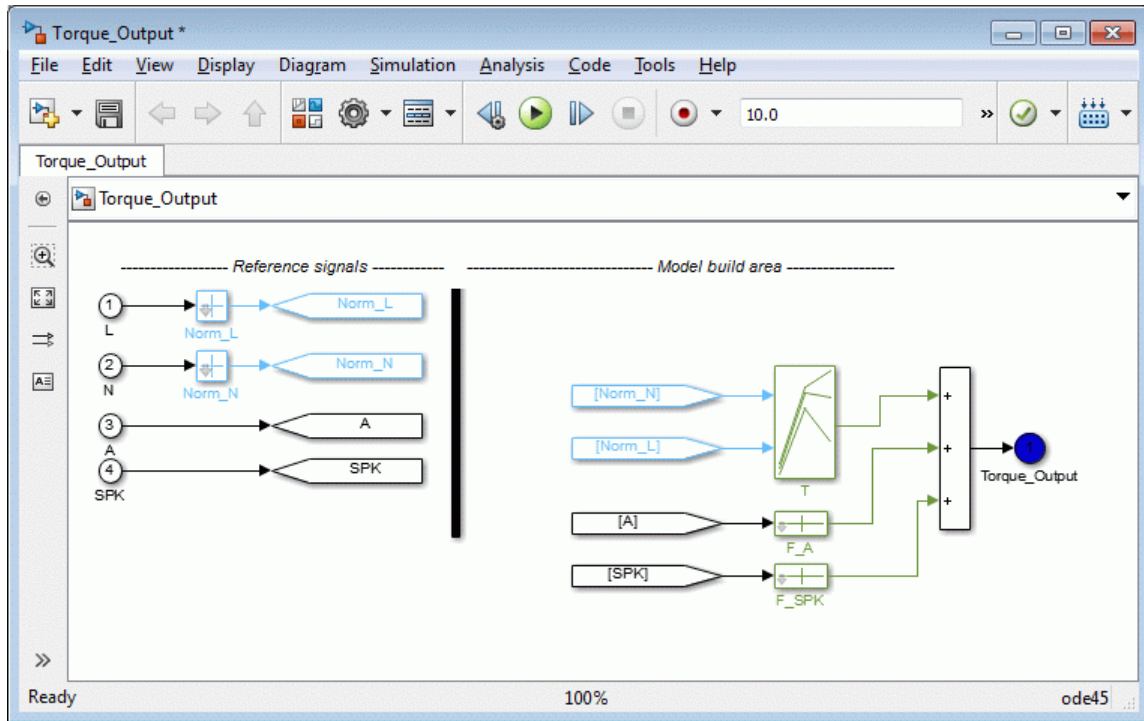
The blocks show how the strategy is built up.

4 Close the Simulink windows.

CAGE displays the new `Torque_Output` feature parsed from Simulink as the output of the algebraic equation of tables. You can see this parsed into the **Strategy** pane as follows:

```
Torque_Output = T(Norm_N(N),Norm_L(L)) + F_A(A) +
F_SPK(SPK)
```

5 Select **View > Full Strategy Display** to turn off the full description and see this simplified expression:

```
Torque_Output = T + F_A + F_SPK
```

This shows the collection of tables that makes up the new feature — a torque table T (with normalizers in speed N and load L) combined with modifier tables depending on

the values of air/fuel ratio and spark. You will fill these tables by using CAGE to compare them with the torque model.

6 Click the plus next to `Torque_Output` to expand the Feature tree and observe the tables created by importing the strategy: `T`, `F_A`, and `F_SPK`. Expand each table node in turn to view the normalizers of each. You will define the sizes of the tables next.

For a more detailed description of strategies, see "What Is a Strategy?" in the CAGE documentation.

Set Up the Tables

Currently, the lookup tables have neither rows nor columns, so you must set up the tables.

Click Calibration Manager 📇 or select **Tools > Calibration Manager**. The Calibration Manager dialog box opens, so you can specify the number of breakpoints for each axis.
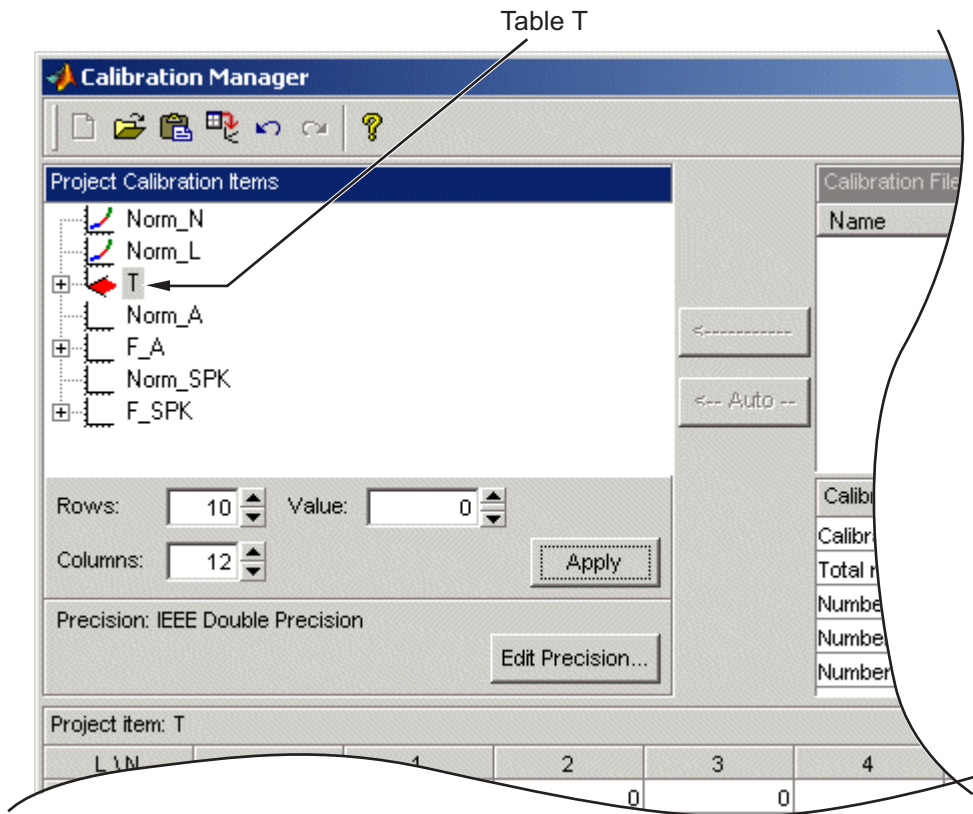
Table T

To set up table T,

1   Highlight the table T by clicking T in the tree hierarchy.

2   Enter 10 as the number of rows and 12 as the number of columns. This determines the size of each normalizer.

3   Set the Value for each cell set to 0.

4   Click **Apply**, and click **Continue** in the dialog to change the size of the table. The pane changes to show the table is set up.

5   Follow the same procedure for the F_A table:

   a   Highlight the F_A node.

   b   Set the number of rows to be 10 and press Enter.

    c   Leave the value for each cell set to 0.

    d   Click **Apply**.

6   Repeat step 5 for F_SPK.

Note The icons change as you initialize each table or function.

---

7   Click **Close** to leave the Calibration Manager.

After completing these steps, you can calibrate the lookup tables.

Process For Feature Calibration

The feature contains both a strategy (which is a collection of tables) and a model. You can use CAGE to fill the lookup tables using the model as a reference.

These are the three steps to calibrate a feature, described in these sections:

1   Calibrate the normalizers. on page 10-14

2   Calibrate the tables. on page 10-18

3   Calibrate the feature on page 10-23 as a whole.

Click the expand icon, ⊞, to expand the nodes and display all the tables and normalizers in the feature.
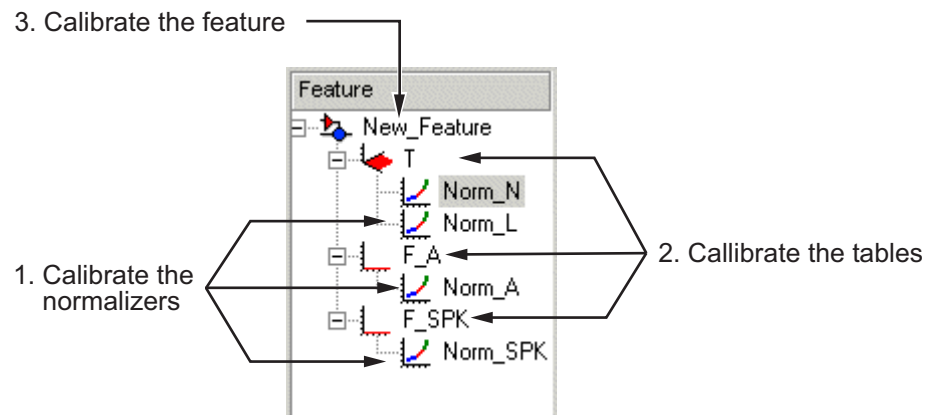
Each node in the display has a different view and different operations.

Calibrate the Normalizers

Normalizers are the axes for the lookup tables. Currently, `Norm_N` has 12 breakpoints; the other normalizers have 10 breakpoints each. This section describes how to set values for the normalizers `Norm_N` and `Norm_L`, based on the torque model, `tq`.

To display the Normalizer view, select the normalizer `Norm_N` in the branch display.

Input output display   Normalizer display   Breakpoint spacing display

Norm_N

Input	Output
0.00	0
1.00	1
2.00	2
3.00	3
4.00	4
5.00	5
6.00	6
7.00	7
8.00	8
9.00	9
10.00	10
11.00	11

Norm_L

Input	Output
0.00	0
1.00	1
2.00	2
3.00	3
4.00	4
5.00	5
6.00	6
7.00	7
8.00	8
9.00	9
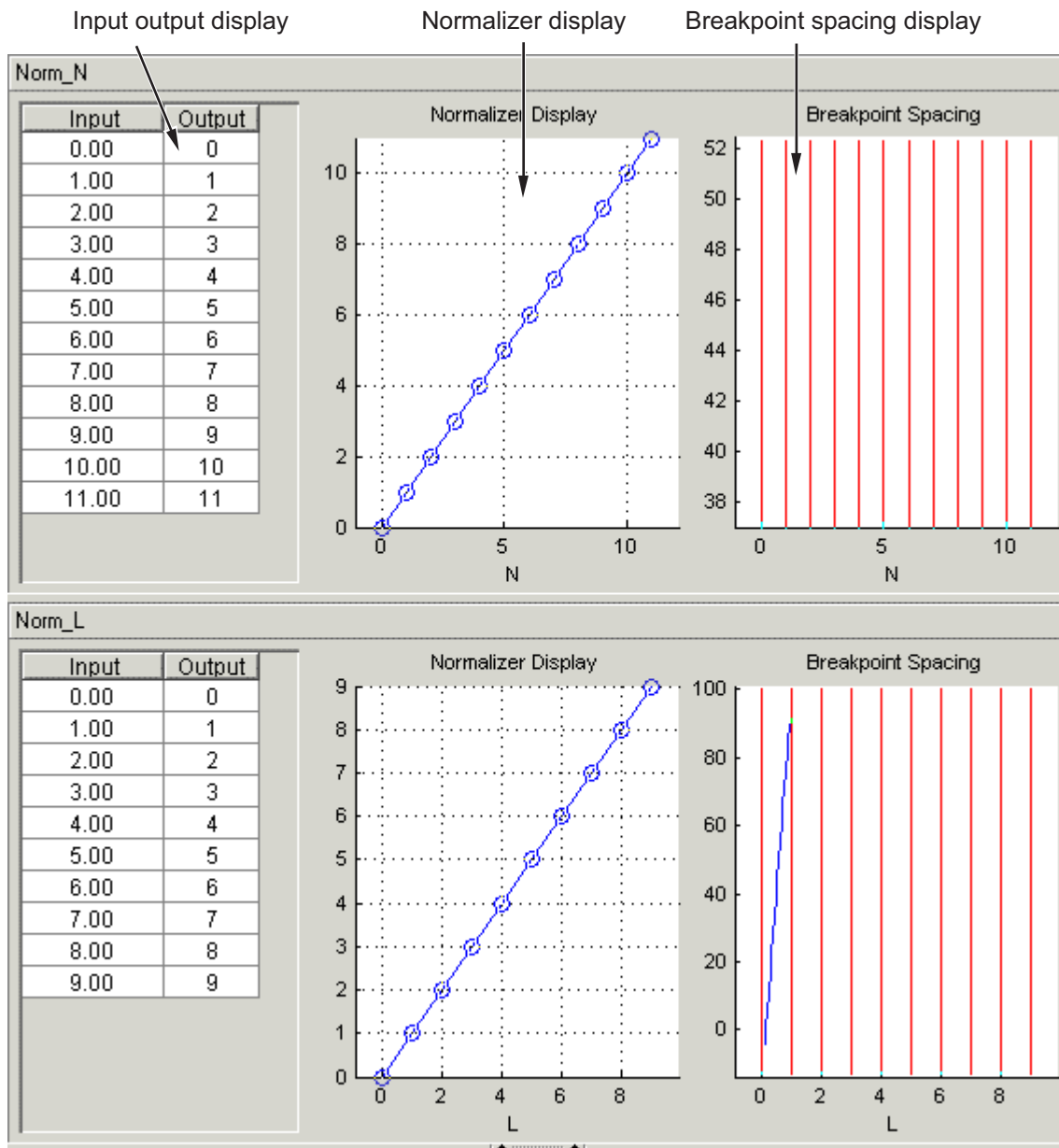
The Normalizer view has two panes, **Norm_N** and **Norm_L**.

In each pane, you see

- An input/output table
- A normalizer display
- A breakpoint spacing display

In both Normalizer panes, the Input Output table and the **Normalizer Display** show the position of the breakpoints.

The **Breakpoint Spacing** display shows a blue slice through the model with the break points overlaid as red lines.

For a more detailed description of the Normalizer view, see "Table Normalizers" in the CAGE documentation.

Placing the Breakpoints Automatically

You now must space the breakpoints across the range of each variable. For example, Norm_N takes values from 500 to 6500, the range of the engine speed.

To space the breakpoints evenly throughout the data values,

1. Click Initialize ▥ in the toolbar. Alternatively, select **Normalizer > Initialize**.

    This opens a dialog box that suggests ranges for Norm_N and Norm_L.

2. To accept the default ranges of values of the data, click **OK**.

Norm_N (N)

Input	Output
500	0
1045.455	1
1590.909	2
2136.364	3
2681.818	4
3227.273	5
3772.727	6
4318.182	7
4863.636	8
5409.091	9
5954.545	10
6500	11

Norm_L (L)

Input	Output
0.1	0
0.2	1
0.3	2
0.4	3
0.5	4
0.6	5
0.7	6
0.8	7
0.9	8
1	9
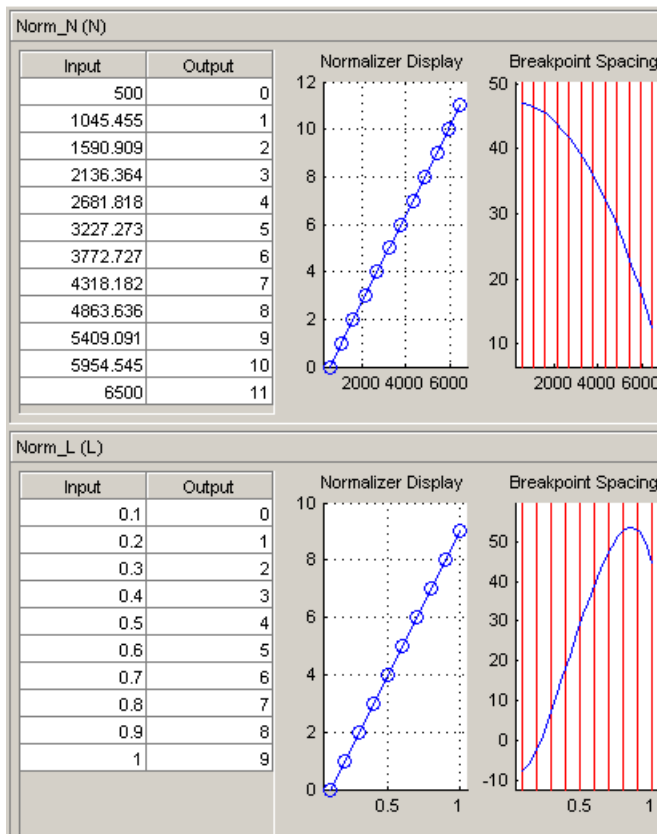
A better fit between model and table can often be achieved by spacing the breakpoints nonlinearly.

1   Click Fill in the toolbar. Alternatively, select **Normalizer > Fill**.

This opens a dialog box that suggests ranges for `Norm_N` and `Norm_L`. It also suggests values for `AFR` and `SPK`; these values are the set points for `AFR` and `SPK`.

2   To accept the values in the dialog box, click **OK**.

This ensures that the majority of the breakpoints are where the model is most curved. The table now has most values where the model changes most. So, with the same number of breakpoints, the table is a better match to the model.

10-17

For more information about calibrating the normalizers, see "About Normalizers" in the CAGE documentation.
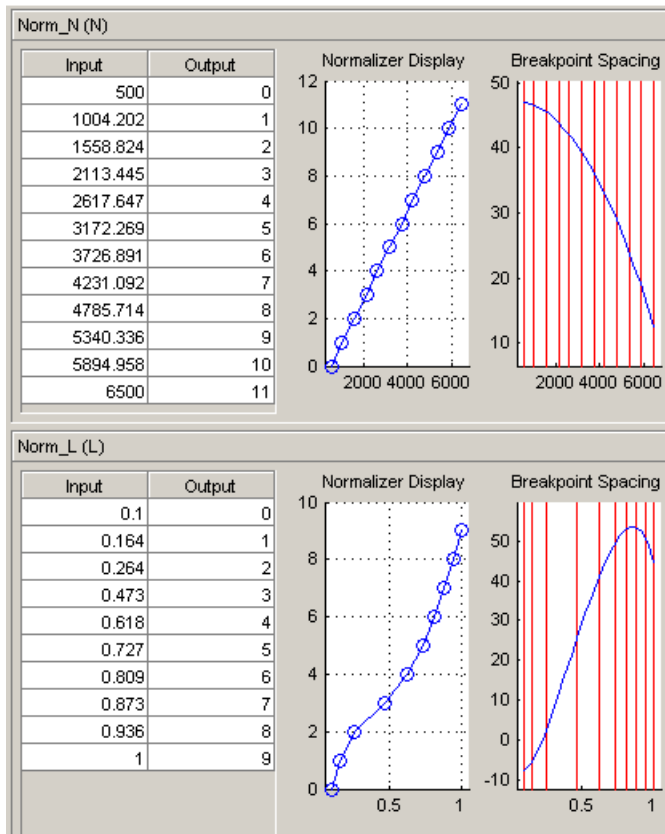
You can now calibrate the lookup tables; this is described in the next section.

Calibrate the Tables

The lookup tables currently have zero as the entry for each cell. This section demonstrates how to fill the table T with values of torque using the torque model, tq.

To view the Table display, click the T node.

This view has two panes: the table and the graph of the table.

To fill the table with values of the model at the appropriate operating points,

1   Click Fill on the toolbar.

This opens the Feature Fill Wizard.

2   Observe that the `tq` model is selected to fill the table. Here you could also set up constraints, for example using a boundary model to constrain filling to table areas where data was collected, and you can link other models or features to inputs.
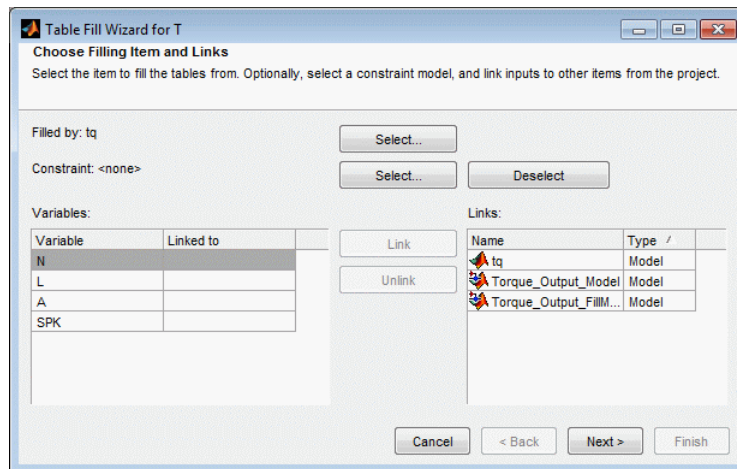
Leave the settings at the defaults and click **Next**.

3   On this screen you can set variable values for optimizing over.
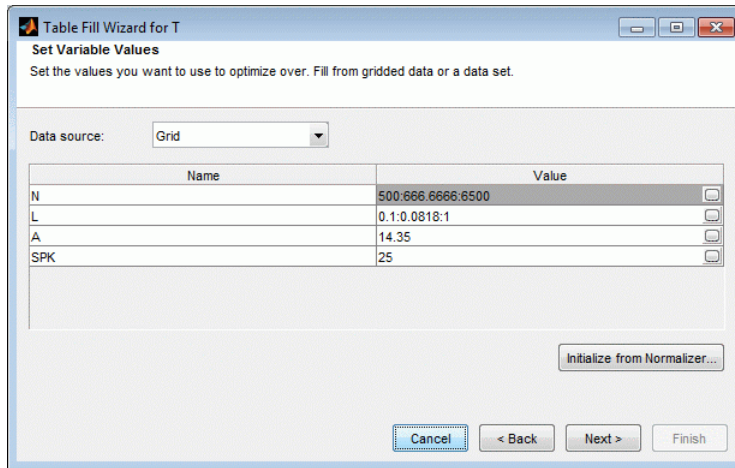
By default the table's normalizer breakpoints (here N and L) and the set points of the other variables (A and SPK) are selected. You can select different normalizers, and edit values to optimize over a range rather than at a single point. You can edit values directly in the **Value** edit box or click the button on the right in the **Value** box to open the Vector Editor dialog box. If you choose a range of values, CAGE fills the table using the average model value at each cell. If you choose **Initialize from Normalizer**, you can use the **Number of values between breakpoints** setting to add values between normalizer breakpoints to optimize over a finer grid than the number of table cells.

Leave the settings at the defaults and click **Next**.

4   Click **Fill Tables**. The graph shows the progress of the optimization.

5   Select all the check boxes.
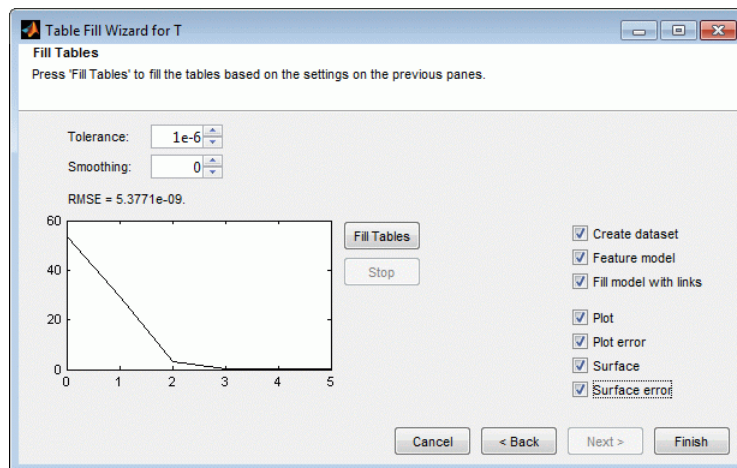
6   Click **Finish**.

CAGE creates plots of the filled table surface and the error between the table values and the model.

7   Click **OK**.

The following view shows the table filled with values of the model.

N \ L	0.1	0.182	0.264	0.345	0.4
500	-3.837	5.666	14.93	23.897	
1166.667	-3.099	6.399	15.843	24.949	
1833.333	-2.823	6.59	16.195	25.456	
2500	-2.995	6.253	15.95	25.428	
3166.667	-3.6	5.399	15.116	24.787	
3833.333	-4.624	4.041	13.703	23.502	
4500	-6.053	2.19	11.721	21.58	
5166.667	-7.872	-0.142	9.18	19.029	
5833.333	-10.068	-2.942	6.089	15.857	
6500	-12.624	-6.198	2.457	12.07	
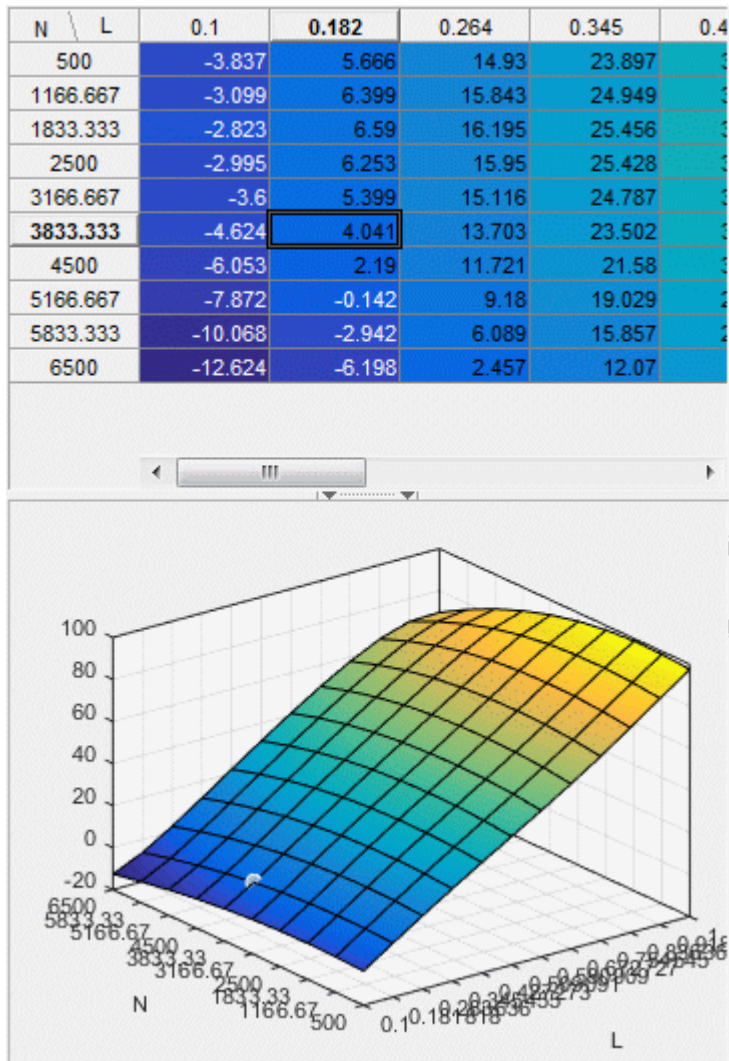
The table T is now filled with optimized values compared to the model at these operating points. CAGE runs an optimization routine over the feature to minimize the total square error between the model and the feature.

For more information about the process of filling tables, see "Optimize Table Values" in the CAGE documentation

Now you must fill the tables F_A and F_SPK and their normalizers. The tables are modifiers for AFR and the spark angle respectively. These steps are described in the next section.

Calibrate the Feature

A feature is a strategy (which is a collection of tables) and a model. Currently the torque table, T, is filled with optimized values compared to the torque model, tq. You must now calibrate the normalizers and tables for F_A and F_SPK.

You could calibrate the normalizers and then the tables for F_A and F_SPK in turn. However, CAGE enables you to calibrate the entire feature in one procedure.

To view the Feature view following, click the Torque_Output node.
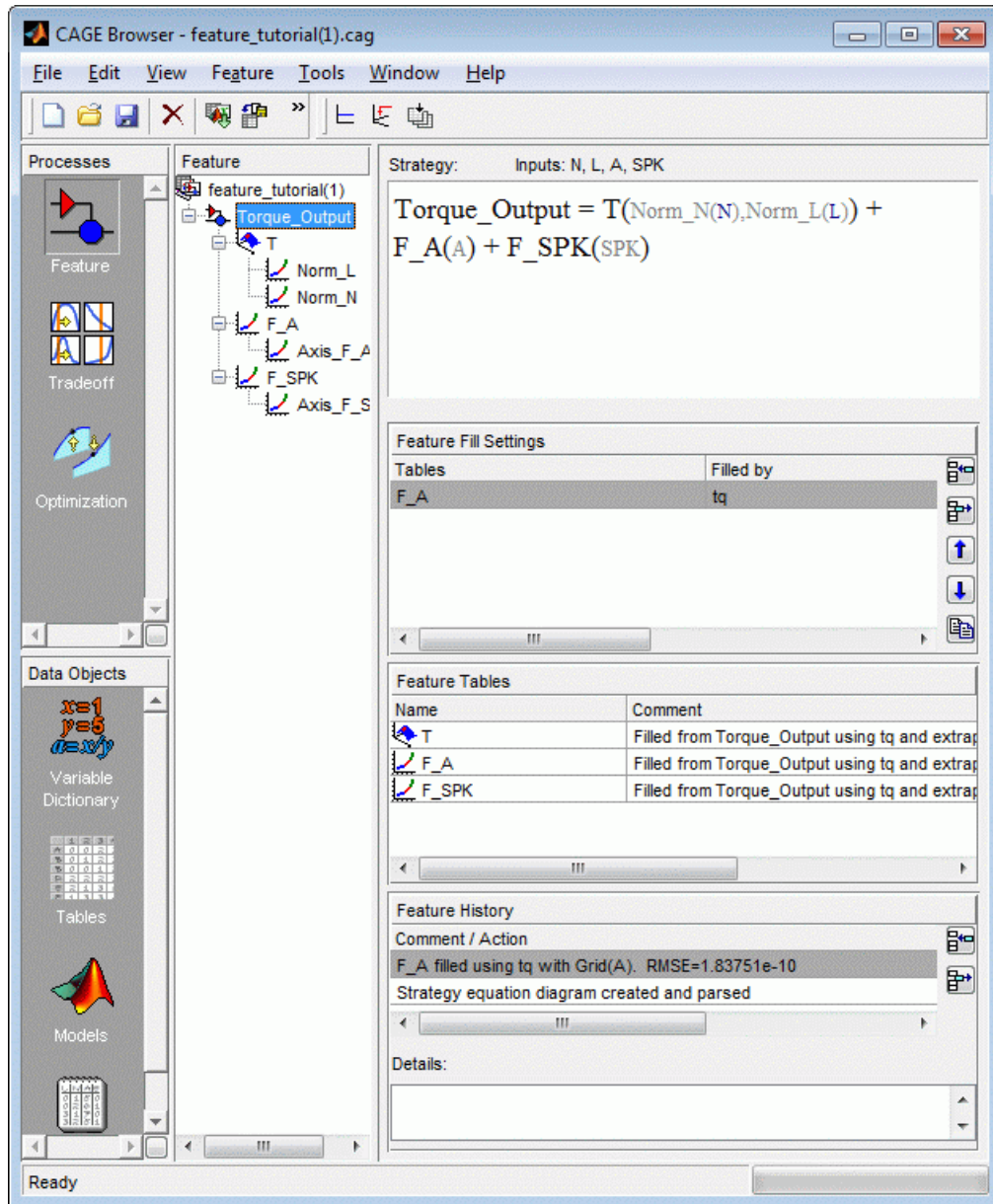
To calibrate all the tables and their normalizers,

1 Select **Feature** > **Initialize** (or use the Initialize toolbar button). The Feature Initialization Options dialog appears.

2 Clear the Enable check boxes for Breakpoints of T, and Values of T, as shown.
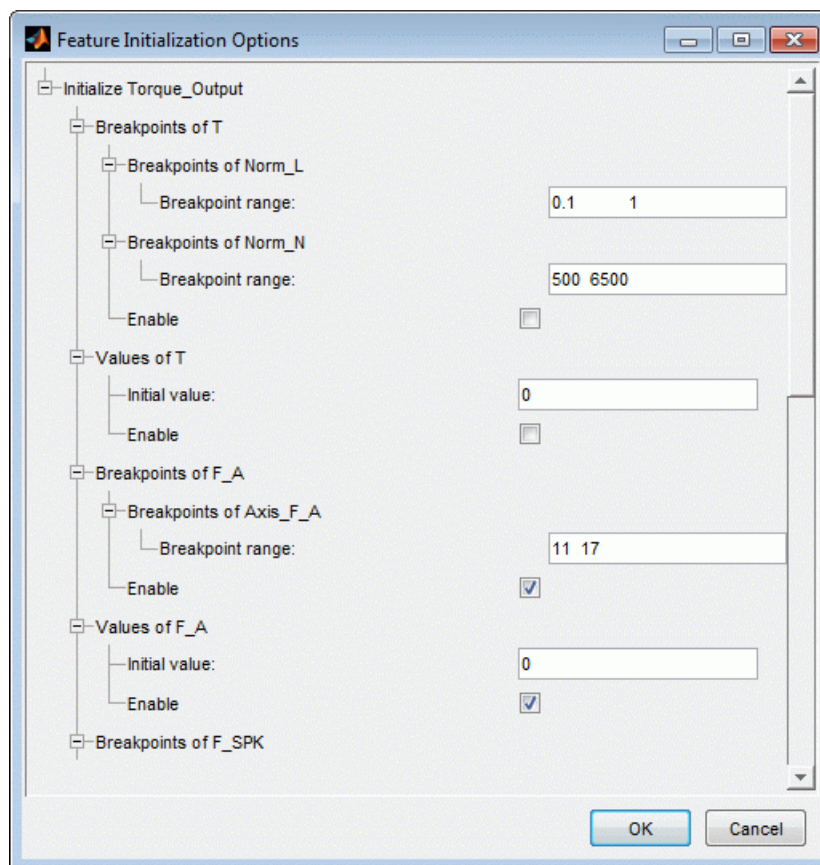
You have already optimized the breakpoints and table values for table T, so you only want to initialize the other tables F_A and F_SPK.

Click **OK**.

3 Select **Feature** > **Fill Feature** (or use the Fill toolbar button) to open the Feature Fill Wizard.

4 Select only the F_A table to fill, and follow the steps in the wizard to fill this table:

a   Click **Next** 3 times.

b   Click **Fill Tables**.

c   Click **Finish**.

5   Select the F_A table node to view the results.

6   Select F_SPK table node and click Fill . Repeat the wizard steps to fill the table.

All three tables and normalizers are filled.
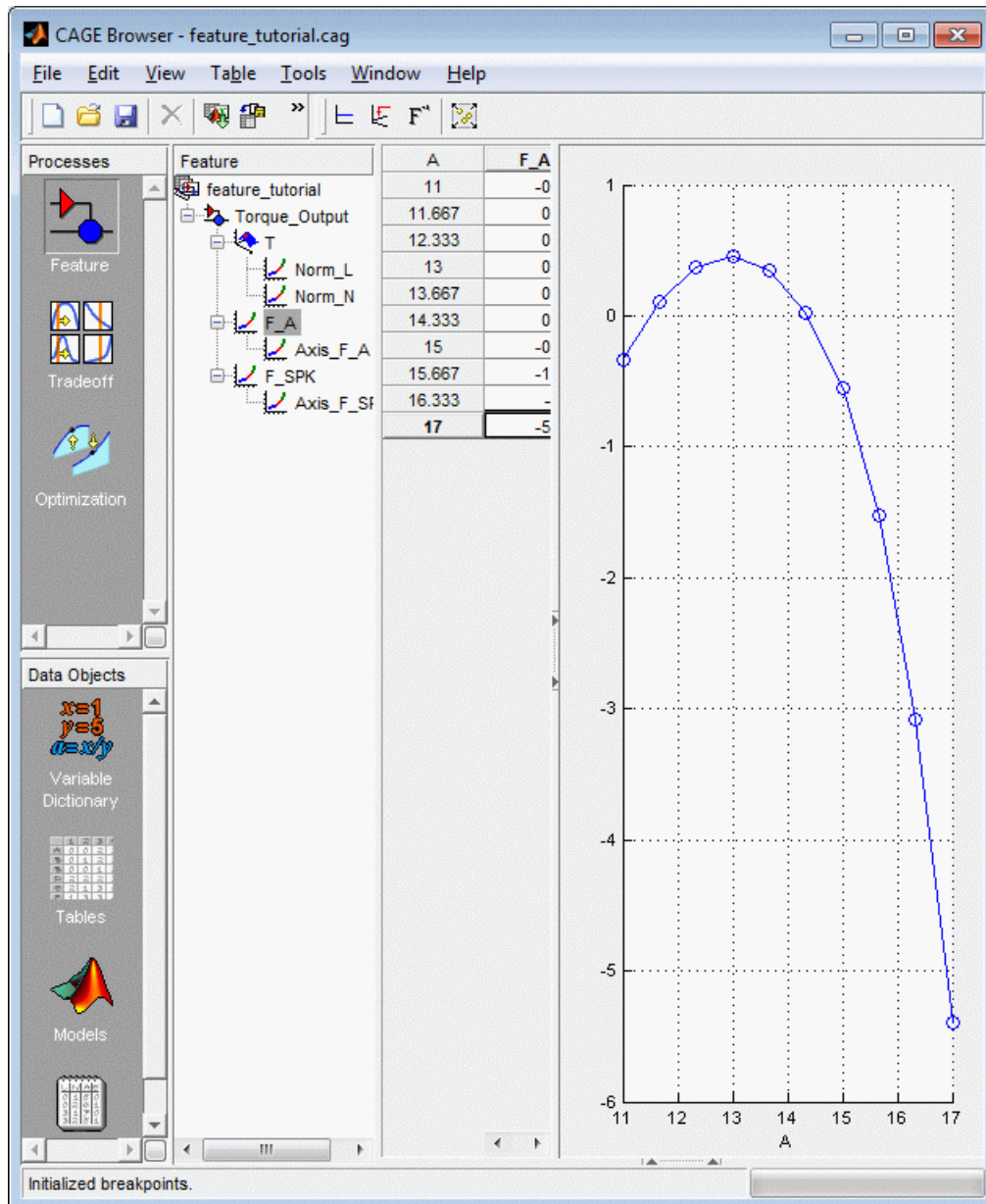
This display shows that the range of the normalizer for F_A is 11 to 17, the range of AFR.

This completes the calibration of the torque feature.

For more information about calibrating features, see "Optimize Table Values" in the CAGE documentation.

You can now export the calibration.

Export Calibrations

To export your feature,

1   Select the Torque_Output node in the branch display.
2   Select **File** > **Export** > **Calibration** > **Selected Item**.
3   Choose the type of file you want to save your calibrations as. For the purposes of this tutorial, in the **Export to** list, select Simple CSV file, a comma separated value file. Click **OK**.
4   Enter feature_tutorial.csv as the file name and click **Save**.

This exports the calibration.

Note that when you choose to export**Selected Item** rather than **All Items**, what you export depends on which node is highlighted:

- Selecting a normalizer node outputs the values of the normalizer.
- Selecting a table node outputs the values of the table and its normalizers.
- Selecting a feature outputs the whole feature (all tables and normalizers).
- Selecting a branch node outputs all the features under the branch.

You have now completed the feature calibration tutorial.

Tradeoff Calibration

This section includes the following topics:

Tradeoff Calibration

In this section...
"What Is a Tradeoff Calibration?" on page 11-2
"Setting Up a Tradeoff Calibration" on page 11-2
"Performing the Tradeoff Calibration" on page 11-7

What Is a Tradeoff Calibration?

A tradeoff calibration is the process of filling lookup tables by balancing different objectives.

Typically there are many different and conflicting objectives. For example, a calibrator might want to maximize torque while restricting nitrogen oxides (NOX) emissions. It is not possible to achieve maximum torque and minimum NOX together, but it is possible to trade off a slight reduction in torque for a reduction of NOX emissions. Thus, a calibrator chooses the values of the input variables that produce this slight loss in torque over the values that produce the maximum value of torque.

This tutorial takes you through the various steps required for you to set up this tradeoff, and then to calibrate the lookup table for it.

Setting Up a Tradeoff Calibration

- "Creating a Tradeoff" on page 11-2
- "Adding Tables to a Tradeoff Calibration" on page 11-5
- "Displaying the Models" on page 11-6

Creating a Tradeoff

Start CAGE by typing

cage

at the MATLAB prompt.

Before you can calibrate the lookup tables, you must set up the calibration.

1 Select **File** > **Open Project** (or the toolbar button) to choose the
`tradeoffInit.cag` file, found in the `matlab\toolbox\mbc\mbctraining`
directory, then click **OK**.

The `tradeoffInit.cag` project contains two models and all the variables necessary
for this tutorial. For information about how to set up models and variables, see
"Calibration Setup" in the CAGE documentation.

2 To create a tradeoff calibration, select **File** > **New** > **Tradeoff**.

This takes you to the **Tradeoff** view. You need to add tables and display models to the
tradeoff, which are described step by step in the following sections:

- "Adding Tables to a Tradeoff Calibration" on page 11-5.
- "Displaying the Models" on page 11-6 describes how you display the models of
  torque and NOX emissions.

1. Open the project file

2. Add a new tradeoff

3. Add a new table

3. Select item to fill table.

5. Display the models
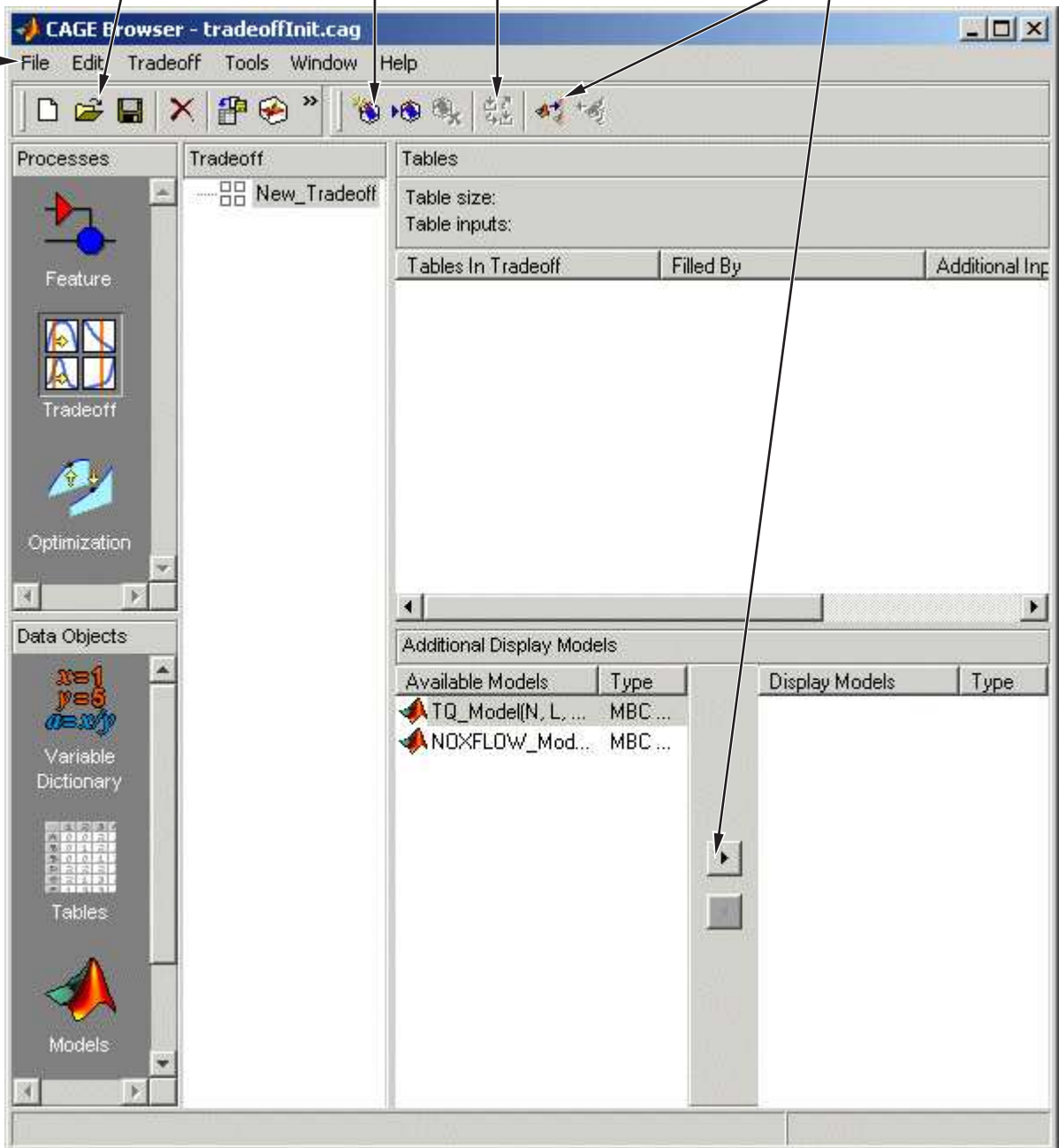
Adding Tables to a Tradeoff Calibration

The models of torque and NOX are in the current session. You must add the lookup table to calibrate.

Both models have five inputs. The inputs for the torque and NOX models are

- Exhaust gas recycling (EGR)
- Air/fuel ratio (AFR)
- Spark angle
- Speed
- Load

For this tutorial, you are interested in the spark angle over the range of speed and load.

To generate a lookup table for the spark angle,

1 Click (Add New Table) in the toolbar. This opens the Table Setup dialog.
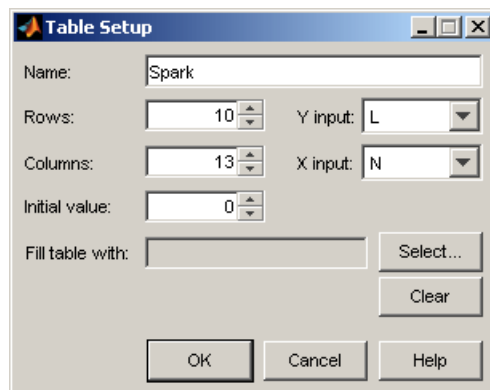
2 Enter Spark as the table **Name**.

3 Check that N is the **X input** and L is the **Y input** (these are selected automatically as the first two variables in the current Variable Dictionary).

4 Enter 10 as the size of the load axis (**Rows**).

5 Enter 13 as the size of the speed axis (**Columns**).

6 Click **Select** to open the dialog Select Filling Item.

11-5
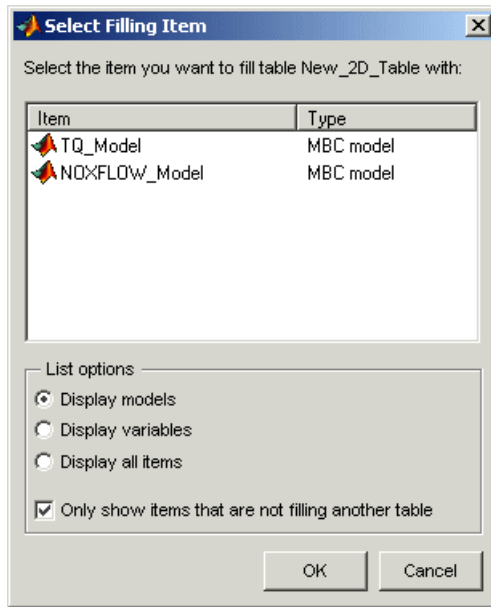
Select the radio button **Display variables**, then select SPK to fill the table and click **OK**.

7 Click **OK** to close the Table Setup dialog.

Before you can perform the calibration, you must display the models.

Displaying the Models

For this tutorial, you are comparing values of the torque and NOX models. Thus, you need to display these models.

To display both models,

*   Click ⬥ Add Model to Display List in the toolbar twice. This will move both available models into the Display list.

*   Alternatively, **Shift**-click to select both models in the **Available Models** list and click ⬲ to include both models in the current display. In this case you want to include all available models. You can click to select particular models in the list to display.

The **Display Models** pane following shows both models selected for display.
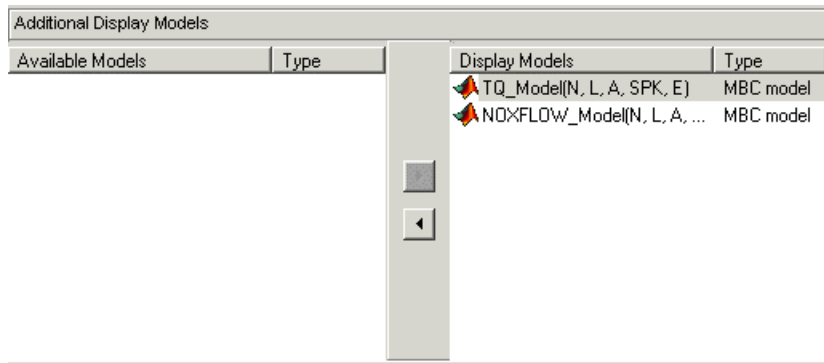
You can now calibrate the tradeoff.

Performing the Tradeoff Calibration

- "Process Overview" on page 11-7
- "Checking the Normalizers" on page 11-10
- "Setting Values for Other Variables" on page 11-11
- "Filling Key Operating Points" on page 11-13
- "Filling the Table by Extrapolation" on page 11-17
- "Exporting Calibrations" on page 11-19

Process Overview

You now fill the lookup table for spark angle by trading off gain in torque for reduction in NOX emissions.

The method that you use to fill the lookup table is

- Obtain the maximum possible torque.
- Restrict NOX to below 250 g/hr at any operating point.

To perform the tradeoff calibration, follow the instructions in the next four sections:

1 Check the normalizers. on page 11-10

2 Set values for the other variables, AFR and EGR. on page 11-11

3 Fill key operating points with values for spark angle on page 11-13.

4 Fill the table by extrapolation on page 11-17.

Once you have completed the calibration, you can export the calibration for use in the electronic control unit.

1. Check the normalizers.　　5. Export the calibration.　　3. Trade off torque and **NOX** to find values of spark to fill key operating points in the table.　　4. Fill the table by extrapolation.
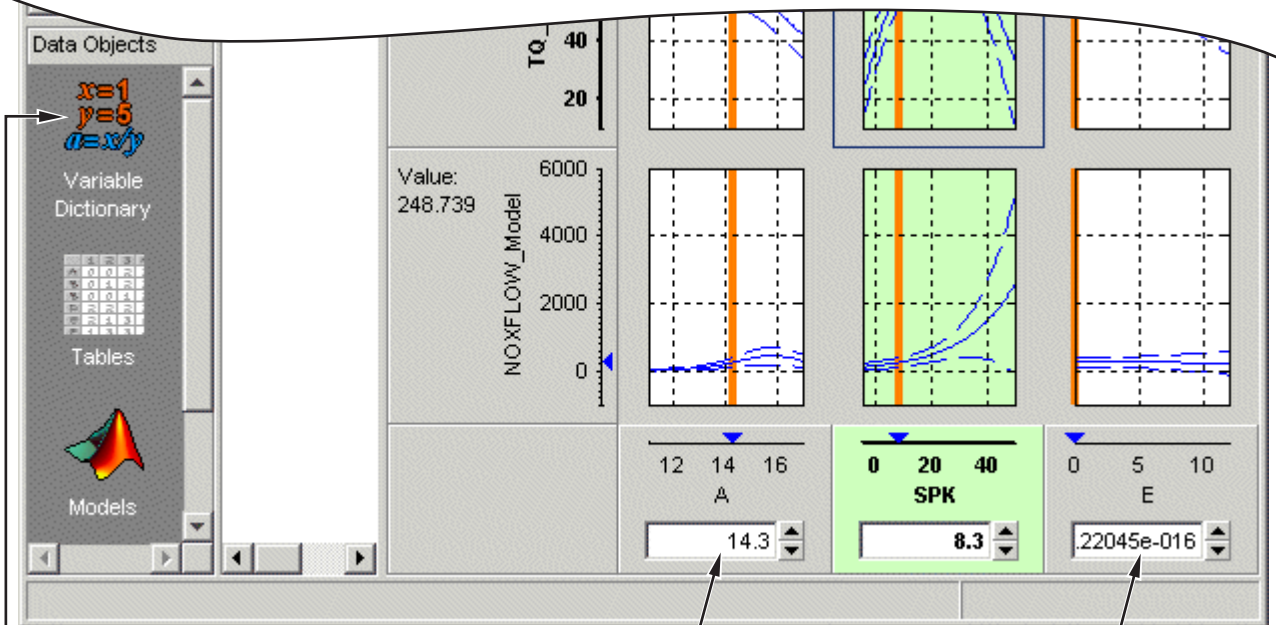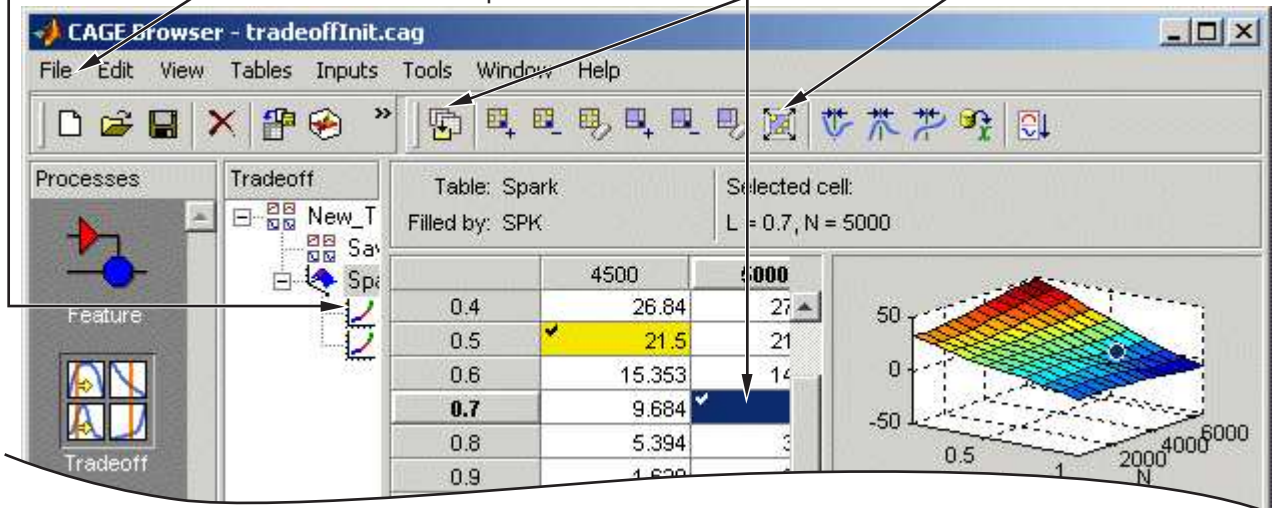
2. Set values for the other variables, either individually for each operating point or in the Variable Directory.

Checking the Normalizers

A normalizer is the axis of the lookup table (which is the collection of breakpoints). The breakpoints of the normalizers are automatically spaced over the ranges of speed and load. These define the operating points that form the cells of the tradeoff table.

Expand the Tradeoff tree by clicking the plus sign in the display, so you can see the `Spark` table and its normalizers `Speed` and `Load`. Click to highlight either normalizer to see the normalizer view. A tradeoff calibration does not compare the model and the table directly, so you cannot space the breakpoints by reference to the model.
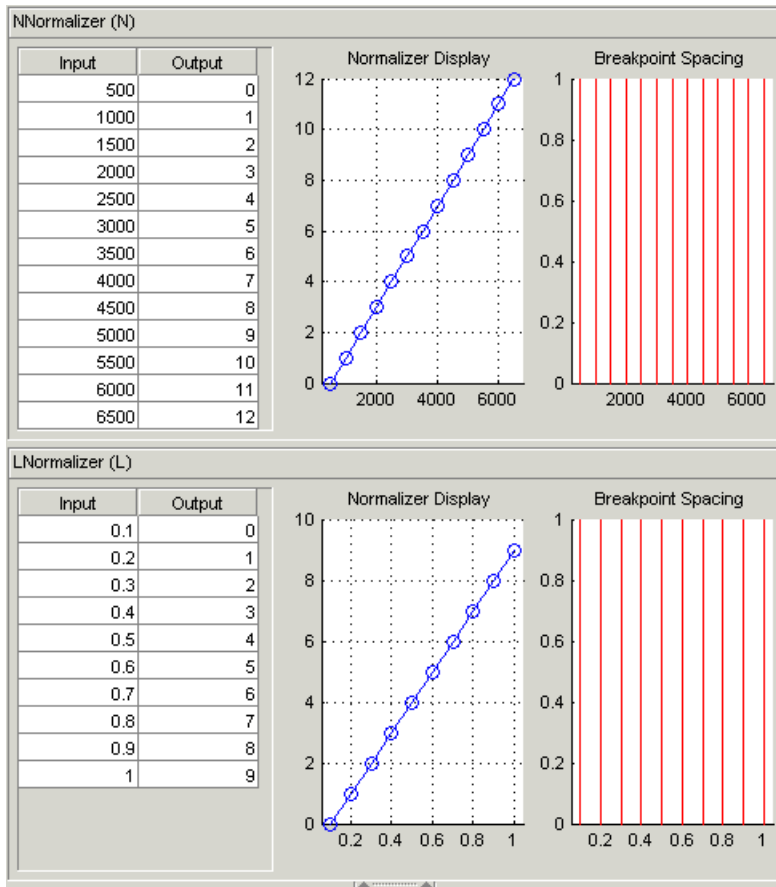
Setting Values for Other Variables

At each operating point, you must fill the values of the spark table. Both of the models depend on spark, AFR (labeled A, in the session), and EGR (labeled E in the session). You could set the values for AFR and EGR individually for each operating point in the table, but for simplicity you will set constant values for these model inputs.

To set constant values of AFR and EGR for all operating points,

1   Click **Variable Dictionary** in the **Data Objects** pane.

2   Click A and edit the **Set Point** to 14.3, the stoichiometric constant, and press **Enter**.

3   Click E and change the **Set Point** to 0 and press **Enter**.

   You have set these values for every operating point in your tradeoff table. You can now fill the spark angle lookup table. The process is described next.

4   Click **Tradeoff** in the **Processes** pane to return to the tradeoff view.

5   Highlight the Spark table node in the Tradeoff tree display.

6   In the lower pane, check that the value for A is 14.3, and the value for E is 0, as shown in the following example. You leave these values unchanged for each operating point.

For each operating point you change the values of spark to trade off the torque and NOX objectives; that is, you search for the best value of spark that gives acceptable torque within the emissions constraint. The following example illustrates the controls you use, and there are step-by-step instructions in the following section.

1. Highlight the **Spark** node.

2. Select operating points in the **Spark** tradeoff table.

3. Change values of **SPK** to trade off torque and NOX emissions at each operating point.

4. Leave A and E at the set point values.

Filling Key Operating Points

You now fill the key operating points in the lookup table for spark angle.

The upper pane displays the lookup table, and the lower pane displays the behavior of the torque and NOX emissions models with each variable.

The object is to maximize the torque and restrict NOX emissions to below 250 g/hr.

Determining the Value of Spark

At each operating point, the behavior of the model alters. The following display shows the behavior of the models over the range of the input variables at the operating point selected in the table, where speed (N) is 4500 and load (L) is 0.5. You can show confidence intervals by selecting **View > Display Confidence Intervals**.

Value of the torque model

Value of the **NOX** model
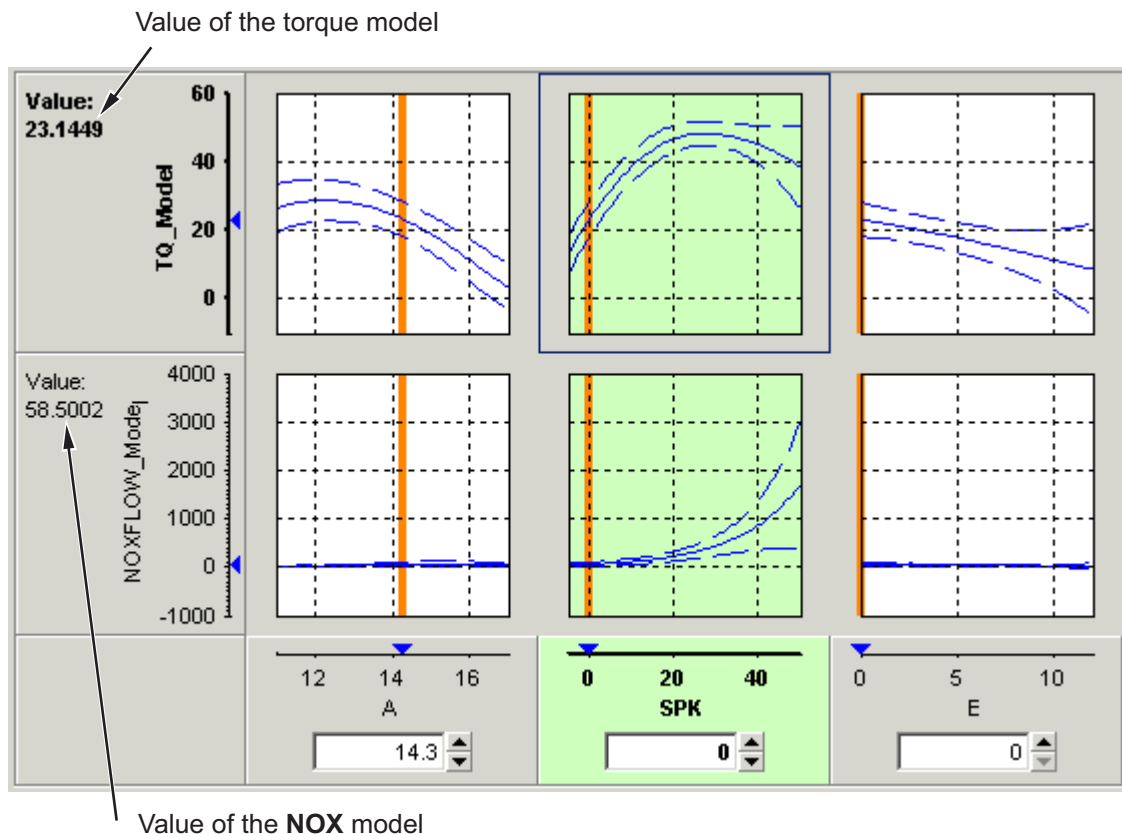
The top three graphs show how the torque model varies with the AFR (labeled A), the spark angle (SPK), and the EGR (E), respectively. The lower panes show how the NOX emissions model varies with these variables.

You are calibrating the Spark table, so the two spark (SPK) graphs are green, indicating that these graphs are directly linked to the currently selected lookup table.

1   Select the operating point N = 4500 and L = 0.5 in the lookup table.
2   Now try to find the spark angle that gives the maximum torque and restricts NOX emissions to below 250 g/hr. You can change the value of spark by clicking and dragging the orange line on the SPK graphs, or by typing values into the SPK edit box. You can change the values of any of the other tradeoff variables in the same way, but as you have already set constant values for A and E you should not change these.

Try different values of spark and look at the resulting values of the torque and NOX models.

3   Click to select the top SPK - TQ_Model graph (`TQ_Model` row, `SPK` column). When selected the graph is outlined as shown in the following example.

4   Now click 'Find maximum of output' (⚒) in the toolbar. This calculates the value of spark that gives the maximum value of torque. The following display shows the behavior of the two models when the spark angle is `26.4458`, which gives maximum torque output.

Torque model behavior

99% Confidence interval for the torque model

NOX model behavior
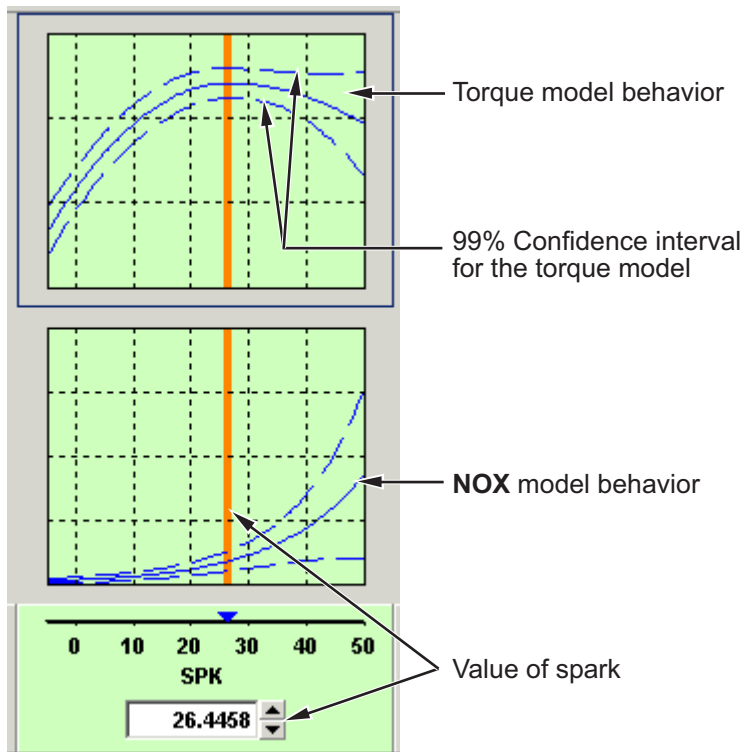
Value of spark

At this operating point, the maximum torque that is generated is `48.136` when the spark angle is `26.4989`. However, the value of NOX is `348.968`, which is greater than the restriction of 250 g/hr. Clearly you have to look at another value of spark angle.

5   Click and drag the orange bar to change to a lower value of spark. Notice the change in the resulting values of the torque and NOX models.

6   Enter `21.5` as the value of **SPK** in the edit box at the bottom of the SPK column.

The value of the NOX emissions model is now `249.154`. This is within the restriction, and the value of torque is `47.2478`.

At this operating point, this value of `21.5` degrees is acceptable for the spark angle lookup table, so you want to apply this point to your table.

7   Press **Ctrl**+T or click [icon] (Apply table filling values) in the toolbar to apply that value to the spark table.

This automatically adds the selected value of spark to the table and turns this cell yellow. It is blue when selected, yellow if you click elsewhere. Look at the table legend to see what this means: yellow cells have been added to the extrapolation mask, and the tick mark indicates you saved this input value by applying it from the tradeoff. You can use the **View** menu to choose whether to display the legend.

8   Now repeat this process of finding acceptable values of spark at four more operating points listed in the table following. In each case,

- Select the cell in the spark table at the specified values of speed and load.
- Enter the value of spark given in the table (the spark angles listed all satisfy the requirements).
- Press **Ctrl**+T or click [icon] (Apply table filling values) in the toolbar to apply that value to the spark table.

Speed, N	Load, L	Spark Angle, SPK
2500	0.3	25.75
3000	0.8	10.7
5000	0.7	8.2
6000	0.2	41.3

After you enter these key operating points, you can fill the table by extrapolation. This is described in the next section.

Filling the Table by Extrapolation

When you have calibrated several key operating points, you can produce a smooth extrapolation of these values across the whole table.

When you apply the value of the spark angle to the lookup table, the selected cell is automatically added to the extrapolation mask. This is why the cell is colored yellow. The extrapolation mask is the set of cells that are used as the basis for filling the table by extrapolation.

Click in the toolbar to fill the table by extrapolation.

The lookup table is filled with values of spark angle.

The following figure displays the view after extrapolation over the table.
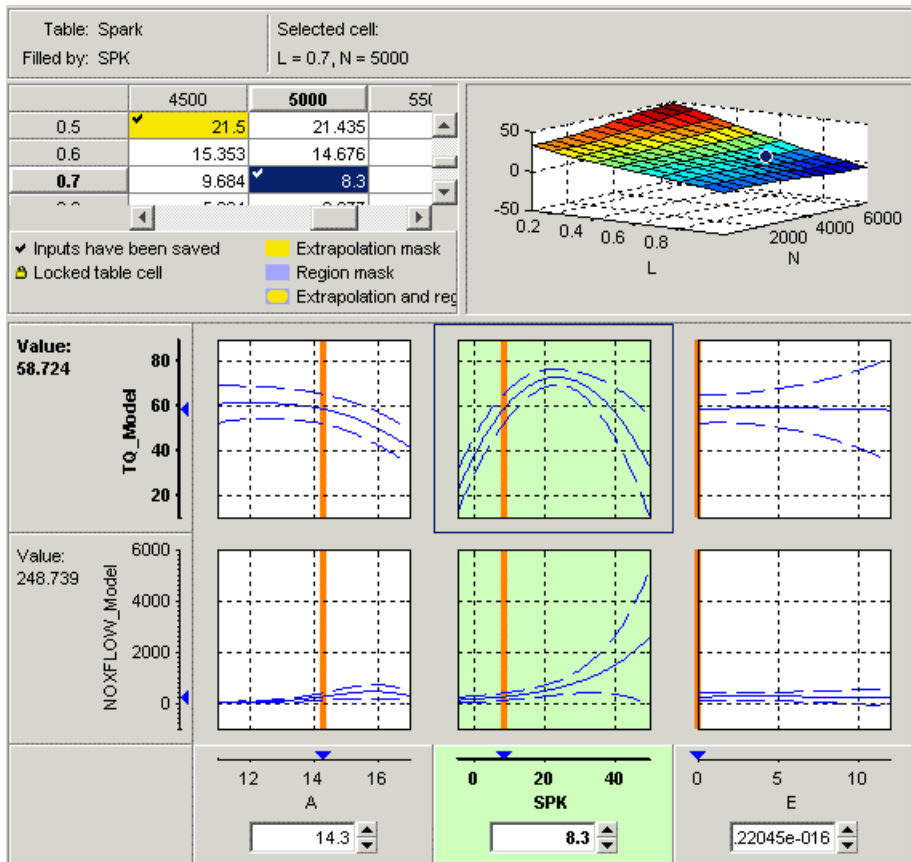
Note Not all the points in the lookup table will necessarily fulfill the requirements of maximizing torque and restricting the NOX emissions.

You could use these techniques to further improve the calibration and trade off torque and NOX to find the best values for each cell in the spark table.

For a more detailed description of tradeoff calibrations, see "Tradeoff Calibration".

You can now export this calibration to file.

Exporting Calibrations

To export your table and its normalizers,

1   Select the `Spark` node in the branch display.

2   Select **File** > **Export** > **Calibration**.

3   Choose the file type you want for your calibration. For the purposes of this tutorial, select `Comma Separated Value (.csv)`.

4   Enter `tradeoff.csv` as the file name and click **Save**.

This exports the spark angle table and the normalizers, `Speed` and `Load`.

You have now completed the tradeoff calibration tutorial.

Data Sets

This section includes the following topics:

Compare Calibrations To Data

Setting Up the Data Set

Tutorial Overview

You can use the **Data Sets** view in CAGE to compare features, tables, and models with experimental data. You can use data sets to plot the features, tables, etc., as tabular values or as plots on a graph.

Data sets enable you to view the data at a set of operating points. You can determine the set of operating points yourself, using Build Grid. Alternatively, you can import a set of experimental data taken at a series of operating points. These operating points are not the same as the breakpoints of your tables.

This tutorial takes you through the basic steps required to compare a completed feature calibration to a set of experimental data.

Start CAGE by typing

```
cage
```

at the MATLAB prompt.

To set up the data set tutorial, you need to

1   Open an existing calibration on page 12-3.
2   Import the experimental data on page 12-3.

3   Add on page 12-5 the `Torque` feature to the data set.

Your data set contains all the input factors and output factors required. As the imported data contains various operating points, this information is also included in the data set.

The next sections describe these processes in more detail.

Opening an Existing Calibration

For this tutorial, use the file `datasettut.cag`, found in the `matlab\toolbox\mbc\mbctraining` directory.

To open this file,

1   Select **File > Open Project**.

2   In the file browser, select `datasettut.cag` and click **Open**.

This opens a file that contains a complete calibrated feature with its associated models and variables. This particular feature is a torque calibration, using a torque table (labeled T1) and modifiers for spark (labeled T2) and air/fuel ratio (labeled T3).

For information about completing a feature calibration, see "Feature Calibration".

3   Select **File > New > Data Set** to add a new data set to your session.

This automatically switches you to the **Factor Information** pane of the data set display.

Importing Experimental Data into a Data Set

To import data into a data set,

1   Select **File > Import > Data > File**.

2   In the file browser, select `meas_tq_data.xls` from the `mbctraining` directory, and click **Open**.

This set of data includes six columns of data, the test cell settings for engine speed (RPM), and the measured values of torque (`tqmeas`), engine speed (`nmeas`), air/fuel ratio (`afrmeas`), spark angle (`spkmeas`), and load (`loadmeas`).

3   The Data Set Import Wizard asks which of the columns of data you would like to import. Click **Next** to import them all.

The following screen asks you to associate variables in your project with data columns in the data.

4 Highlight `afr` in the **Project Assignments** column and `afrmeas` in the **Data Column**, then click the assign button, shown.

5 Repeat this to associate `load` with `loadmeas`, `n` with `RPM`, and `spk` with `spkmeas`. The dialog box should be the same as shown.
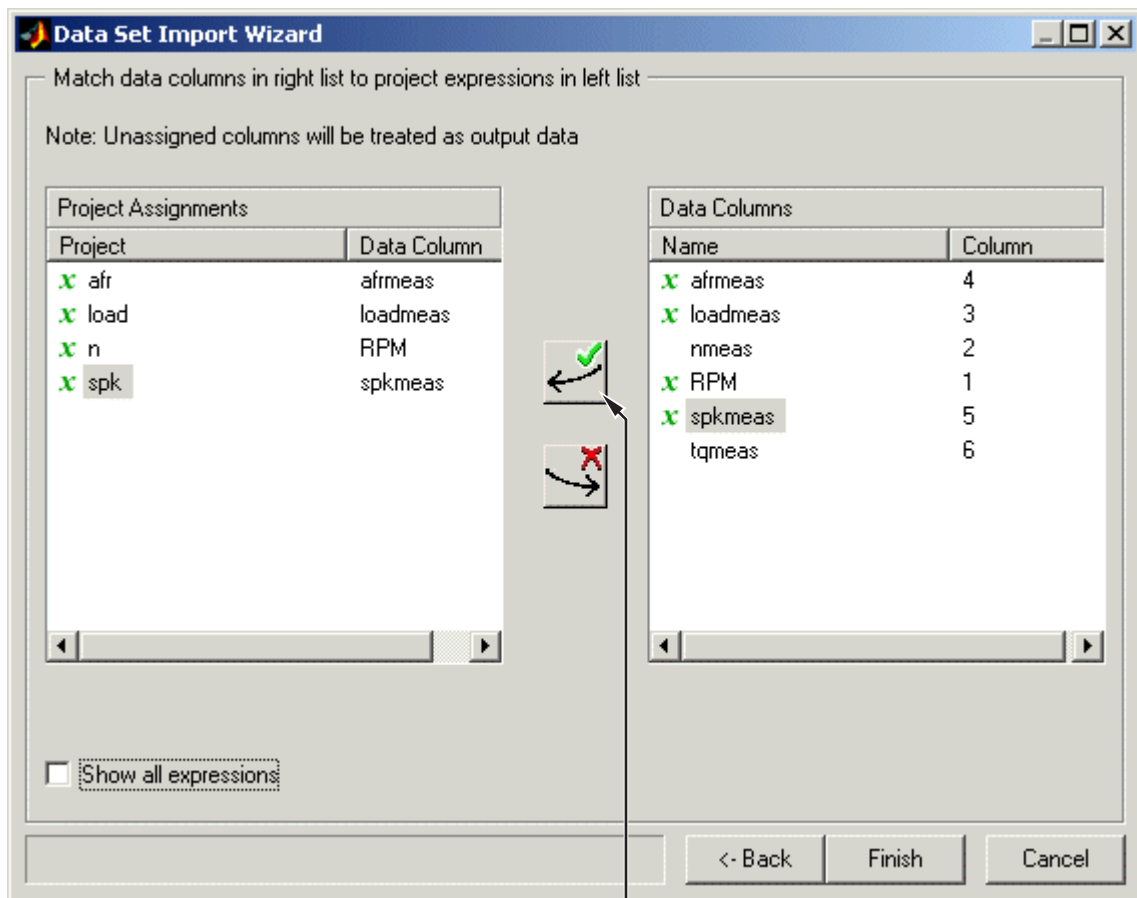
Assign button

6 Click **Finish** to close the dialog box.

> **Note** If you need to reassign any inputs after closing this dialog you can click ![icon] in the toolbar or select **Data** > **Assign**.

Adding an Item to a Data Set

To add the `Torque` feature to the data set,

1 Highlight the `Torque` feature in the lower list of **Project Expressions**.

2 Select **Data** > **Factors** > **Add to Data Set**.

This adds two objects to the data set: `Torque: Model` and `Torque: Strategy`. These two objects make up the `Torque` feature.

- `Torque: Model` is the model used as a reference point to calibrate the feature.
- `Torque: Strategy` is the values of the feature at these operating points.

When these steps are complete, the list of factors includes four input factors and four output factors, as shown.
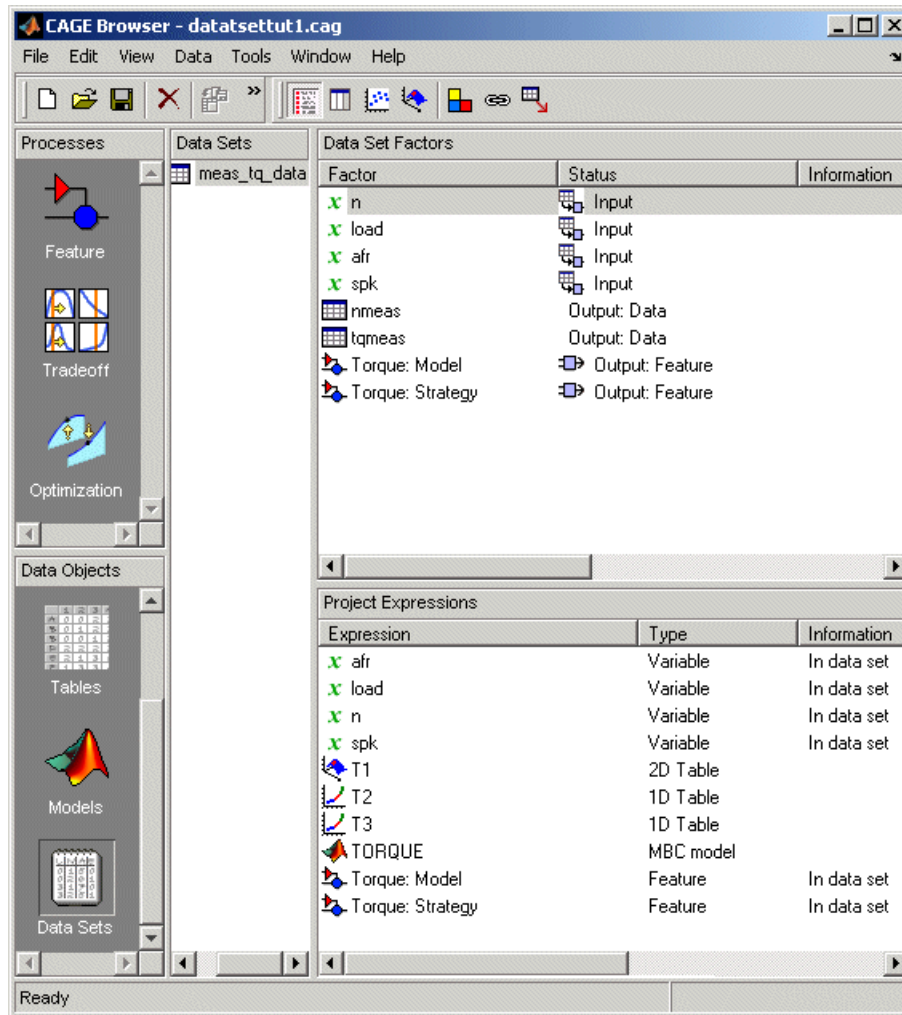
Comparing the Items in a Data Set

- "Coloring the Display" on page 12-11

Viewing the Data Set as a Table

By viewing the data set, you can compare experimental data with calibrations or models in your project.

Click in the toolbar to view the data set as a table of values.

	n	load	afr	spk	nmeas	tqmeas	Torque: Model	Torque: Strategy
1	2235	0.549	9.5	0.1	2247	66.7	71.666	66.079
2	3591	0.454	13.2	0.1	3613	54.1	47.163	46.891
3	4946	0.651	12	0.1	4974	73.7	47.573	79.256
4	881	0.648	11.9	5.7	881	75.8	99.23	80.211
5	2234	0.441	13.3	0.1	2247	55.9	51.256	45.152
6	3591	0.747	10.9	0.1	3612	90	92.837	105.586
7	4947	0.541	9.7	0.1	4973	62.8	57.76	57.587
8	881	0.622	9.9	0.1	884	72.1	76.198	60.926
9	1219	0.333	14	0.1	1224	41.8	33.226	21.318
10	1558	0.382	12	0.1	1567	49.4	40.487	31.957
11	1896	0.209	10.7	3.3	1906	28.5	3.492	4.197
12	2234	0.284	9.8	3.2	2245	36	23.063	19.891
13	2574	0.407	13.4	3	2588	49.9	49.629	44.794
14	2914	0.595	11.5	3.1	2929	70.5	84.68	82.229
15	3251	0.781	12.3	3.1	3268	90.5	117.424	117.259
16	3589	0.668	13.5	3	3608	77.1	87.987	96.408
17	3930	0.452	11.9	3.1	3952	52.7	46.511	51.722
18	4268	0.235	10.9	3	4293	27.7	5.253	3.085
19	4606	0.194	12	3.2	4633	21.3	-2.088	-5.771

In the table, the input cells are white and the output cells are grey. Select the `Torque: Strategy` column header to see the view shown. The selected column turns blue and the column headers of the strategy's inputs (`n`, `load`, `afr` and `spk`) turn cream. Column headers are always highlighted in this way when they are associated with the currently selected column (such as model inputs, strategy inputs or linked columns).

In addition to viewing the columns, you can use data sets to create a column that shows the difference between two columns:

1 Select the `tqmeas` and `Torque: Strategy` columns by using **Ctrl**+click.

2 Select **Create Error** from the right-click menu on either column header.

This creates another column that is the difference between `tqmeas` and `Torque: Strategy`. Note that all the columns that are inputs to this new column have highlighted headers.

	n	load	afr	spk	nmeas	tqmeas	Torque: Model	Torque: Strategy	tqmeas_minus_Torque
1	2235	0.549	9.5	0.1	2247	66.7	71.666	66.079	-0.621
2	3591	0.454	13.2	0.1	3613	54.1	47.163	46.891	-7.209
3	4946	0.651	12	0.1	4974	73.7	47.573	79.256	5.556
4	881	0.648	11.9	5.7	881	75.8	99.23	80.211	4.411
5	2234	0.441	13.3	0.1	2247	55.9	51.256	45.152	-10.748
6	3591	0.747	10.9	0.1	3612	90	92.837	105.586	15.586
7	4947	0.541	9.7	0.1	4973	62.8	57.76	57.587	-5.213
8	881	0.622	9.9	0.1	884	72.1	76.198	60.926	-11.174
9	1219	0.333	14	0.1	1224	41.8	33.226	21.318	-20.482
10	1558	0.382	12	0.1	1567	49.4	40.487	31.957	-17.443
11	1896	0.209	10.7	3.3	1906	28.5	3.492	4.197	-24.303
12	2234	0.284	9.8	3.2	2245	36	23.063	19.891	-16.109
13	2574	0.407	13.4	3	2588	49.9	49.629	44.794	-5.106
14	2914	0.595	11.5	3.1	2929	70.5	84.68	82.229	11.729
15	3251	0.781	12.3	3.1	3268	90.5	117.424	117.259	26.759
16	3589	0.668	13.5	3	3608	77.1	87.987	96.408	19.308
17	3930	0.452	11.9	3.1	3952	52.7	46.511	51.722	-0.978
18	4268	0.235	10.9	3	4293	27.7	5.253	3.085	-24.615
19	4606	0.194	12	3.2	4633	21.3	-2.088	-5.771	-27.071

The error column is simply the difference between `tqmeas` and `Torque: Strategy`. This provides a simple way of comparing the feature and the measured data.

Viewing the Data Set as a Plot

1   Click or select **View > Plot** to view the data set as a plot.

    The lower pane lists all the output expressions in the data set and in the project.

2   Use **Ctrl**+click to select `tqmeas` and `Torque: Strategy` from the lower list.
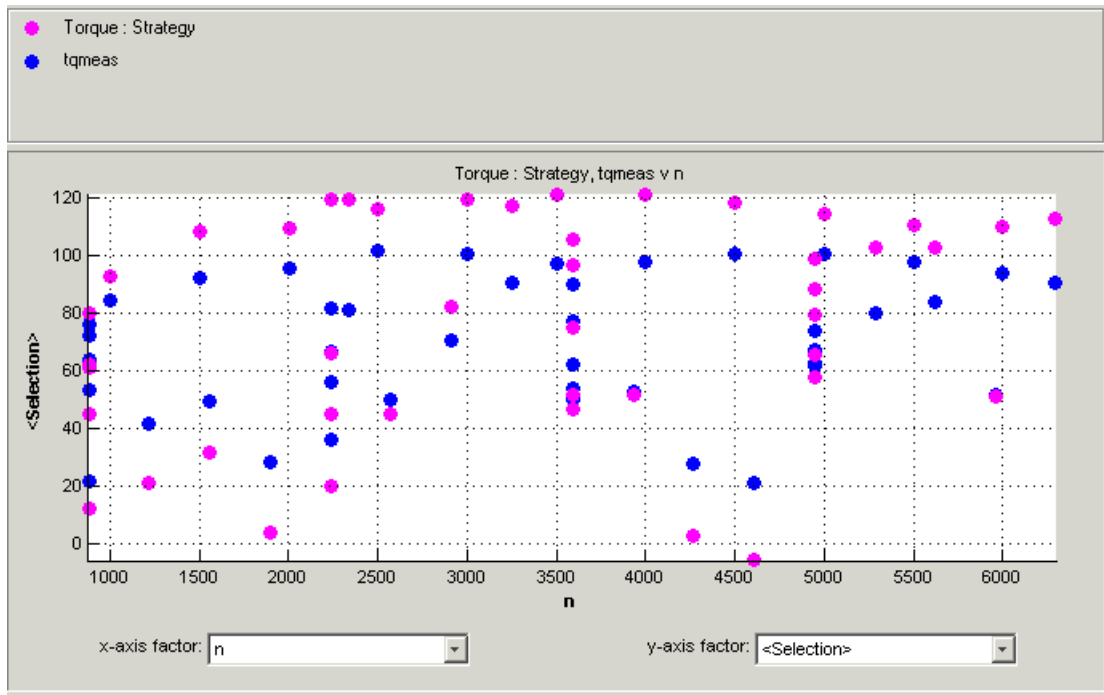
3 Change the **x-axis factor** to n from the drop-down menu.

This displays the calibrated values of torque from the feature, and the measured values of torque from the experimental data, against the test cell settings for engine speed.

Clearly there is some discrepancy between the two.

Displaying the Error

View the error between the calibrated and measured values of torque.

12-9
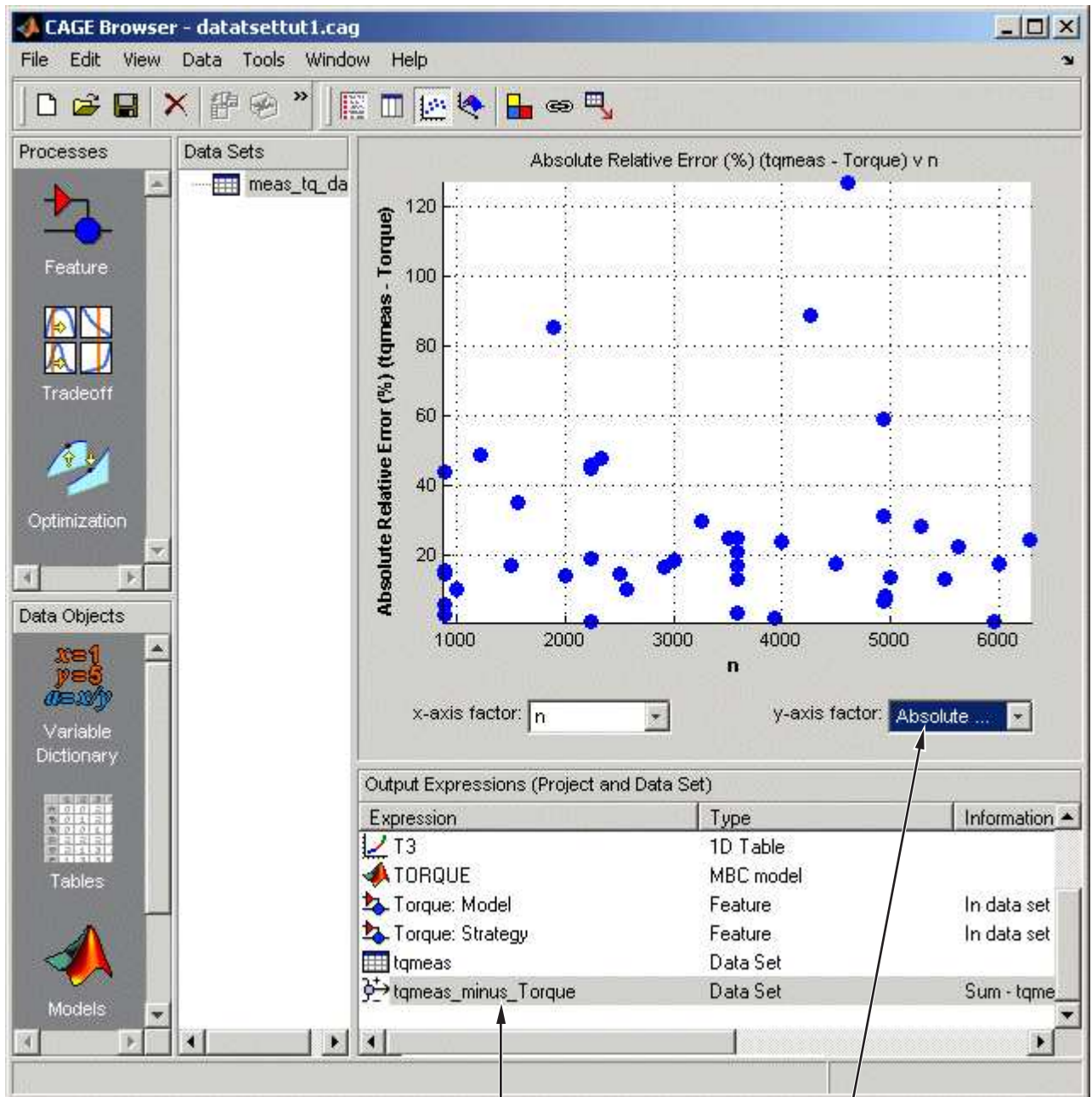
1. Select tqmeas_minus_Torque.

2. Select Absolute Relative Error (tqmeas - Torque).

1. Select tqmeas_minus_Torque from the lower list (**Output Expressions**).
2. For the **y-axis factor**, select Absolute Relative Error (tqmeas - Torque) from the drop-down menu.

As you can see, there seems to be no particular correlation between engine speed and the error in the calibration.

Coloring the Display

1. Select **Color by Value** from the right-click menu on the graph.
2. From the **Color by** drop-down menu, select load.

In this display, you can see that some of the low values of load display a high error.

Limiting the Range of the Colors
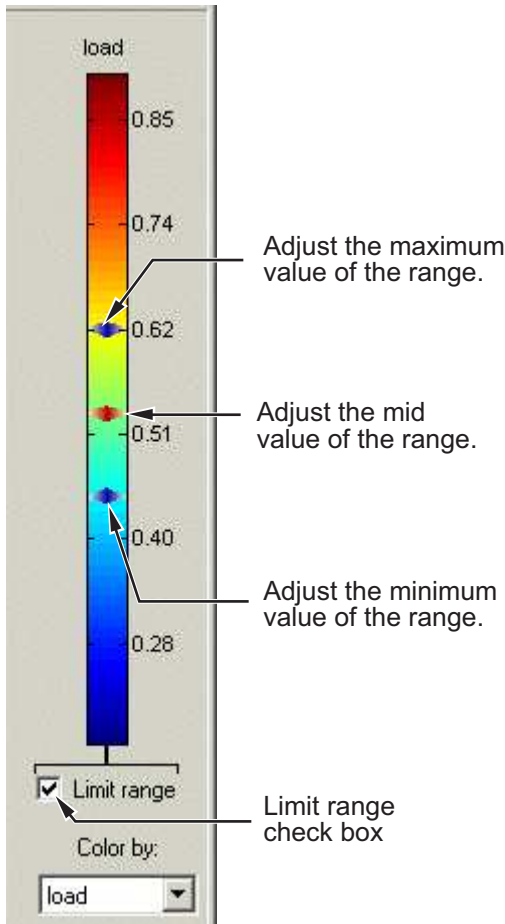
To view the colors in more detail, you can limit the range of the colors:

1 Select the **Limit range** box (or you could right-click the graph and select **Restrict Color to Limits**).

2 Set the minimum value of the color range to be as low as possible by dragging the minimum value down.

3 Set the maximum value of the color range to be around 0.4.

As the low values of load are causing large errors, it would be wise to reexamine the calibration, particularly at small values of `load`.

Reassigning Variables

You can alter the data set by changing which variables are used for project expressions.

Instead of using the test cell settings for the engine speed (RPM), you might want to use the measured values of engine speed (`nmeas`). So you have to reassign the variable `n` to `nmeas`.

To reassign `n`,

1 Click or select **Data > Assign**.

2 In the dialog that appears, select `n` from the **Project Assignments** pane and `nmeas` from the **Data Columns** pane.

3 Click the assign button.
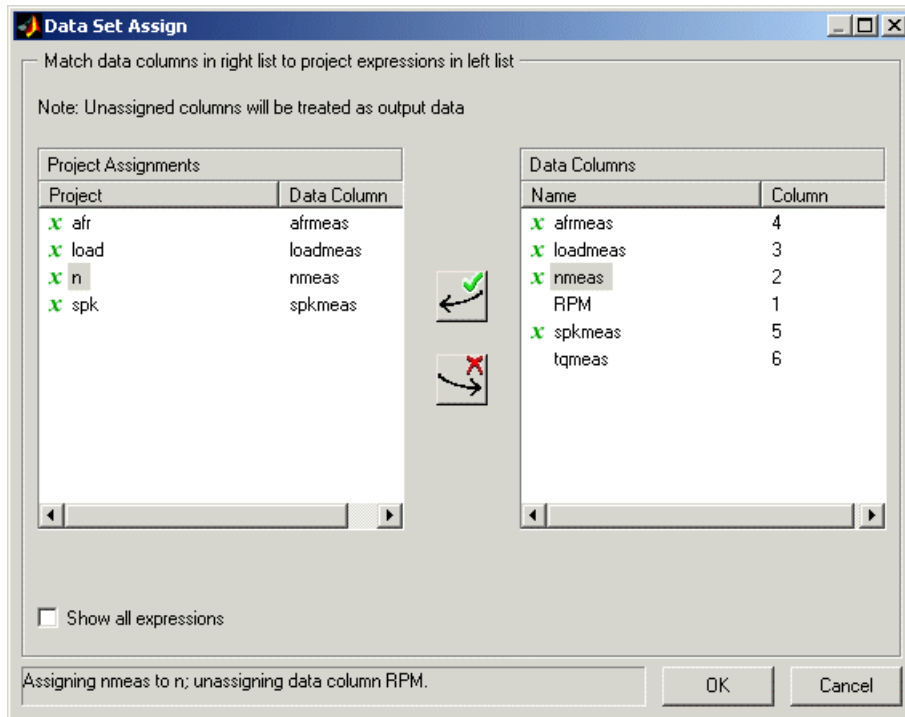
You can now compare your calibration with your experimental data again, using the techniques described.

For more information about the functionality of data sets, see "Data Sets in CAGE".

You have now completed the data sets tutorial.

Filling Tables from Data

This section includes the following topics:

Fill Tables from Data

Setting Up a Table and Experimental Data

Tutorial Overview

If you are considering a straightforward strategy, you might want to fill tables directly from experimental data. For example, a simple torque strategy fills a lookup table with values of torque over a range of speed and relative air charge, or load. You can use CAGE to fill this strategy (which is a set of tables) by referring to a set of experimental data. You can also fill tables with the output of optimizations.

This tutorial takes you through the steps of calibrating a lookup table for torque, based on experimental data.

- This section describes the steps required to set up CAGE in order to calibrate a table by reference to a set of data.
- "Filling the Table from the Experimental Data" on page 13-7 describes the process of filling the lookup table.
- "Selecting Regions of the Data" on page 13-11 describes how you can select some of the data for inclusion when you fill the table.
- "Exporting the Calibration" on page 13-13 describes how to export your completed calibration.

Start CAGE by typing

```
cage
```

at the MATLAB prompt.

If you already have a CAGE session open, select **File > New Project**.

First you will set up a blank table ready for filling using experimental data or optimization output.

The steps that you need to follow to set up the CAGE session are

1   Add the variables on page 13-3 for speed and load by importing a variable dictionary.
2   Add a new table on page 13-3 to your session.
3   Import your experimental data on page 13-5.

The next sections describe each of these processes in detail.

Adding Variables

Before you can add tables to your session, you must add variables to associate with the normalizers or axes.

To add a variable dictionary,

1   Select **File > Import > Variable Dictionary**.
2   Select `table_filling_tutorial.xml` from the `matlab\toolbox\mbc\mbctraining` directory.

This loads a variable dictionary into your session. The variable dictionary includes the following:

- `N`, the engine speed
- `L`, the relative air charge
- `A`, the air/fuel ratio (AFR)
- `stoich`, the stoichiometric constant

You can now add a table to your session.

Adding a New Table

You must add a table to fill.

To add a new table,

1   Select **File > New > 2D table**.

This opens a dialog box that asks you to specify the variable names for the normalizers. As you can see in the dialog controls, accepting the defaults will create a table with ten rows and ten columns with an initial value of 0 in each cell.

2   Change the number of columns to 7.

3   Select L as the variable for normalizer **Y** and N as the variable for normalizer **X**, then click **OK**.

Note   In CAGE, a 2-D table is defined as a table with two inputs.

CAGE takes you to the **Tables** view, where you can see the following.
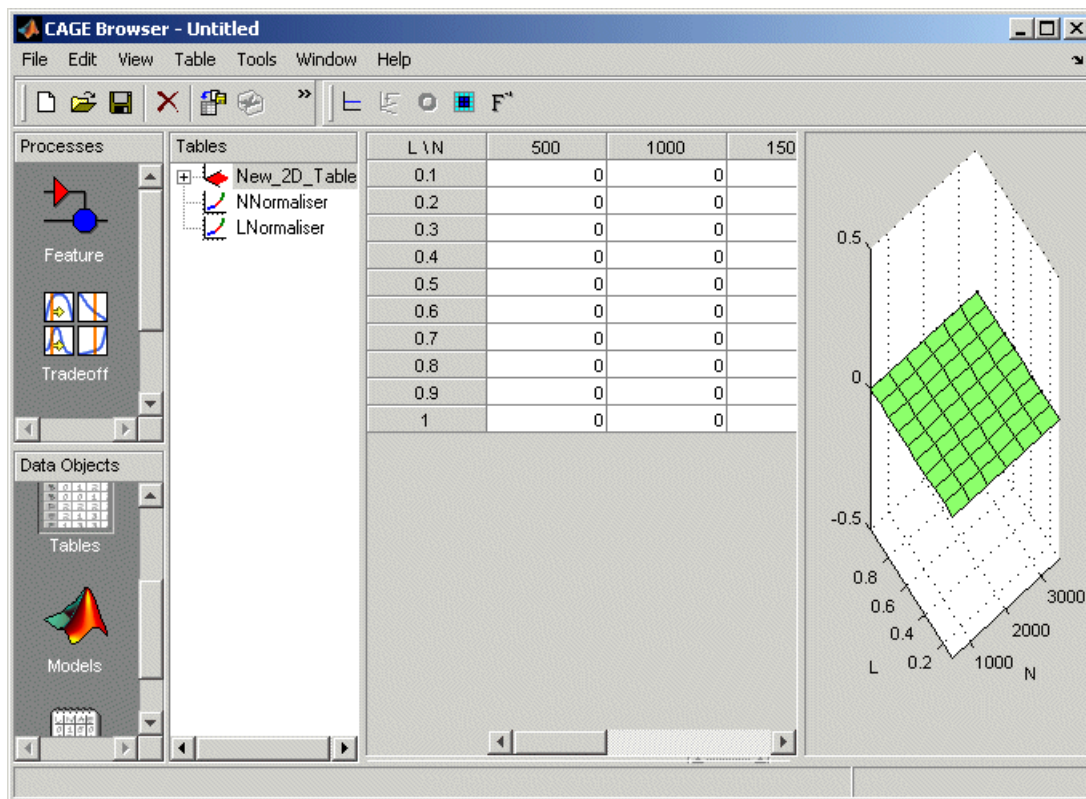
Inspecting the Values of the Normalizers

CAGE has automatically initialized the normalizers by spacing the breakpoints evenly across the range of values for the engine speed (N) and load (L). The variable ranges are found in the variable dictionary. Switch to the Normalizer view to inspect the normalizers.

Expand the table branch by clicking ⊞, and select `NNormalizer` as shown.
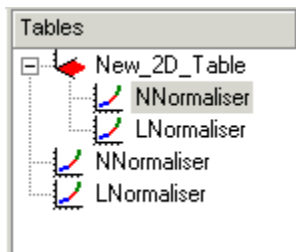
This displays the two normalizers for the table.

You have an empty table with breakpoints over the ranges of the engine speed and load, which you can fill with values based on experimental data.

Importing Experimental Data

To fill a table with values based on experimental data, you must add the data to your session. If you want to fill a table with the output of an optimization, the output appears automatically in the Data Sets view as a new data set called `Exported_Optimization_Data` when you select the Export to Data Set toolbar button. For this tutorial you need to import some experimental data.

CAGE uses the **Data Sets** view to store grids of data. Thus, you need to add a data set to your session as well.

Select **File > New > Data Set** to add a data set to your session. This changes the view to the **Data Set** view.

You can now import experimental data into the data set:

1   Select **File > Import > Data**.
2   In the file browser, select `meas_tq_data.csv` from the `matlab\toolbox\mbc` `\mbctraining` directory and click **Open**.

This set of data includes six columns of data: the test cell settings for engine speed (RPM), and the measured values of torque (`tqmeas`), engine speed (`nmeas`), air/fuel ratio (`afrmeas`), spark angle (`spkmeas`), and load (`loadmeas`).

3 This opens the Data Set Import Wizard. The first screen asks which of the columns of data you want to import. Click **Next** to import them all.

The following screen asks you to associate variables in your project with data columns in the data.

4 Highlight `N` in the **Project Assignments** column and `nmeas` in the **Data Column**, then click the assign button, shown.

5 Repeat this to associate `L` with `loadmeas`. The dialog box should be the same as the following.
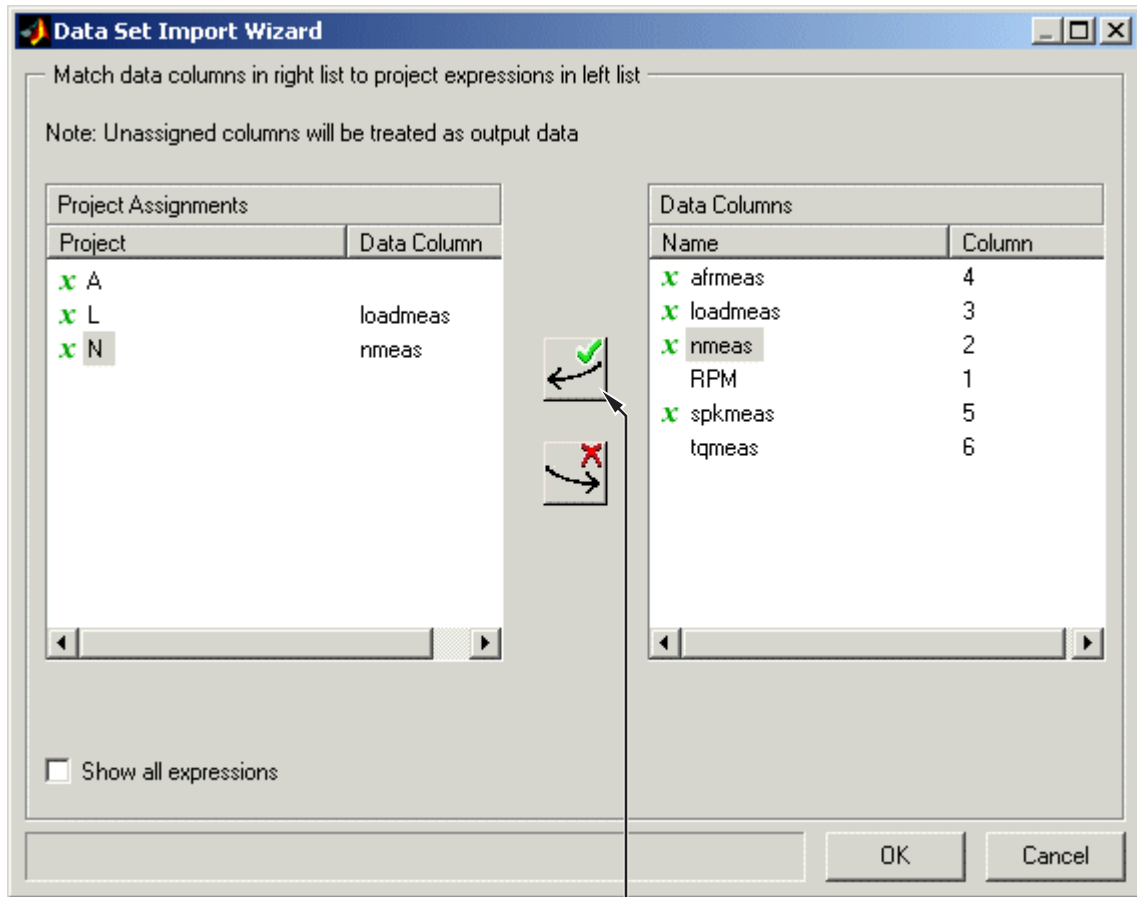
Assign button

6 Click **Finish** to close the dialog box.

You now have an empty table and some experimental data in your session. You are ready to fill the table with values based on this data.

Filling the Table from the Experimental Data

You have an empty table and the experimental data in your session. You can now fill the table with values based on your data.

The data that you have imported is a series of measured values of torque at a selection of different operating points. These operating points do not correspond to the values of the breakpoints that you have specified. The lookup table has a range of engine speed from 500 revolutions per minute (rpm) to 3500 rpm. The range of the experimental data is far greater.

CAGE extrapolates the values of the experimental data over the range of your table. Then it fills the table by selecting the torque values of the extrapolation at your breakpoints.

To fill the table with values based on the experimental data,

1   To view the **Table Filler** display, click ![icon] (Fill Table From Data Set) in the toolbar in the **Data Sets** view; or select **View > Table Filler**.

    You can use this display to specify the table you want to fill and the factor you want to use to fill it.

2   In the lower pane, select New_2D_Table from the **Table to fill** list.

3   Select tqmeas from the **Factor to fill table** list. This is the data that you want to use to fill the table.

4   Select N from the **x-axis factor** list and L from the **y-axis factor** list. Your session should be similar to the following display.

An operating point from the experimental data (a blue dot)

A breakpoint in your lookuptable (a cross)

Filling table New_2D_Table, from factor tqmeas

x-axis factor: N

y-axis factor: L

Filling table New_2D_Table with output factor tqmeas from meas_tq_data

Table to fill

Table	Inputs
New_2D_Table	N, L.

Factor to fill table

Factor	Information
RPM	
spkmeas	
tqmeas	

Table filling rules (optional)

Click and drag over Data Set plot to create rules

☑ Show table history after fill

Fill Table

The upper pane displays the breakpoints of your table as crosses and the operating points where there is data as blue dots. Data sets display the points in the

experimental data, not the values at the breakpoints. You can inspect the spread of the data compared to the breakpoints of your table before you fill the table.

5 To view the table after it is filled, ensure that the **Show table history after fill** box, at the bottom left, is selected.

6 To fill the table with values of `tqmeas` extrapolated over the range of the normalizers, click **Fill Table**.

This opens the History dialog box, shown.

L \ N	500	1000	1500	2000	2500	3000	3500
0.1	12.245	13.471	14.637	15.084	14.622	13.805	13.044
0.2	23.802	25.336	26.94	27.322	25.9	24.344	23.697
0.3	35.14	36.987	38.912	38.876	36.598	33.439	31.511
0.4	46.028	48.217	51.119	51.517	49.49	45.317	40.169
0.5	56.839	58.411	60.752	62.257	62.139	61.779	62.486
0.6	68.694	69.387	69.545	69.367	69.788	71.364	68.274
0.7	79.019	79.285	78.65	76.015	75.705	82.919	85.571
0.8	88.482	88.409	92.981	98.575	92.016	91.03	93.019
0.9	104.147	106.258	110.804	114.302	112.183	107.478	107.431
1	121.64	123.967	126.968	129.007	128.826	127.695	127.643

7 Click **Close** to close the History dialog box and return to the **Table Filler** display.

8 To view the graph of your table, as shown, select **Data > Plot > Surface**.

Filling table New2DTable, from factor tqmeas
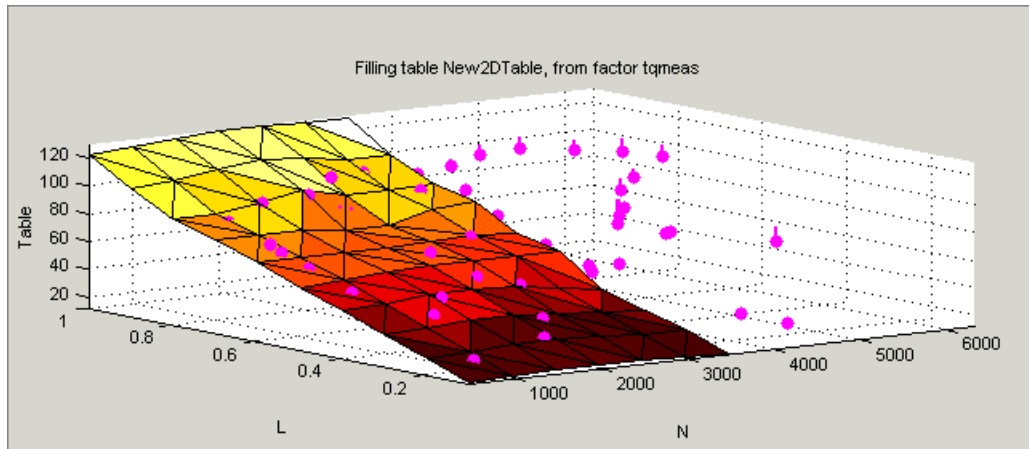
This display shows the table filled with the experimental points overlaid as purple dots.

The table has been calibrated by extrapolating over the values of your data and filling the values that the data predicts at your breakpoints.

Notice that the range of the table is smaller than the range of the data, as the table only has a range from 500 rpm to 3500 rpm.

The data outside the range of the table affects the values that the table is filled with. You can exclude the points outside the range of the table so that only points in the range that you are interested in affect the values in the table.

Selecting Regions of the Data

You can ignore points in the data set when you fill your lookup table.

For example, in this tutorial the experimental data ranges over values that are not included in the lookup table. You want to ignore the values of engine speed that are greater than the range of the table.

To ignore points in the data set,

1   Select **Data > Plot > Data Set**. This returns you to the view of where the breakpoints lie in relation to the experimental data.
2   To define the region that you want to include, left-click and drag the plot. Highlight all the points that are included in your table range, as shown.

Filling table New_2D_Table, from factor tqmeas
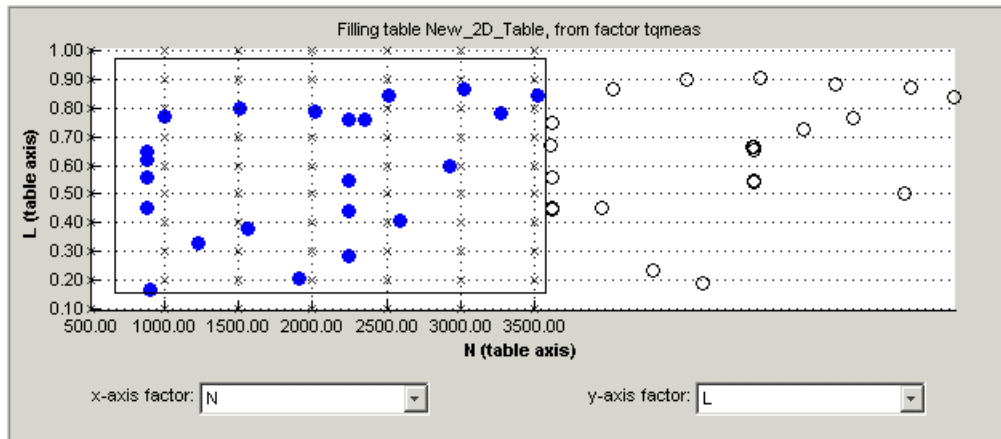
3   To fill the table based on an extrapolation over these data points only, click **Fill Table**. This opens the History display again.

4   In the History display, select version 2 and 3, using **Ctrl**+click. The following display shows a comparison between the table filled with two different extrapolations.

History for New_2D_Table

Version	Comment / Action	Date and Time
3	Values filled from data set meas_tq_data, factor tqmeas	16-Mar-2004 13:41:36
2	Values filled from data set meas_tq_data, factor tqmeas	16-Mar-2004 13:32:06
1	Initial configuration	16-Mar-2004 12:15:34

Reset

Add...

Remove

Edit...

L \ N	500	1000	1500	2000	2500	3000	3500
0.1	-0.368	0.203	0.454	-0.018	-0.622	-0.36	0.268
0.2	-0.525	0.02	0.324	-0.202	-0.437	0.626	1.227
0.3	-0.621	-0.16	0.465	0.312	0.583	3.236	5.139
0.4	-0.451	-0.262	-0.016	0.121	-0.04	3.294	8.275
0.5	-0.247	0.14	0.762	0.41	-1.059	-1.354	-2.413
0.6	-0.746	0.292	1.868	1.555	-0.748	-0.192	2.926
0.7	-0.706	0.468	2.908	4.145	0.084	-2.017	-3.697
0.8	-1.201	-1.266	-0.239	-0.649	0.424	1.39	-0.499
0.9	-6.319	-6.772	-4.776	-2.133	-0.709	-1.971	-3.44
1	-11.745	-11.121	-8.054	-4.653	-4.504	-8.048	-10.813

Close   Help

5   Click **Close** to close the History viewer.

6   Select **Data > Plot > Surface** to view the surface again.

The display of the surface now shows the table filled only by reference to the data points that are included in the range of the table.

You have filled a lookup table with values taken from experimental data.

Exporting the Calibration

To export the calibration,

1   To highlight the table that you want to export, you must first click **Tables**, shown.

2   Highlight the New_2D_Table.

3   Select **File** > **Export** > **Calibration** > **Selected Item**.

4   Choose the type of file you want to save your calibrations as. For the purposes of this tutorial, select Comma Separated Value (.csv).

5   Enter table_filling_tutorial.csv as the file name and click **Save**.

This exports the calibration.

You have now completed this tutorial.

Optimization and Automated Tradeoff

This section includes the following topics:

Optimization and Automated Tradeoff

Import Models to Optimize

To open the CAGE browser and set up your optimization:

1   Start the CAGE Browser part of the Model-Based Calibration Toolbox product by typing

    cage

    at the MATLAB prompt.

2   To reach the Optimization view, click the **Optimization** button in the **Processes** pane.

You can use the **Optimization** view to set up, run, view, and export optimizations. You must also set up optimizations here in order to use them for automated tradeoff.

When you first open the **Optimization** view both panes are blank until you create an optimization. After you set up your optimizations, the left **Optimization** pane shows a tree hierarchy of your optimizations, and the right hand panes display details of the optimization selected in the tree, as with other CAGE processes.

For any optimization, you need one or more models. You can run an optimization at a single point, or you can supply a set of points to optimize. The steps required are as follows:

1   Import a model or models.

2   Set up a new optimization.

The following tutorial guides you through this process to evaluate this optimization problem:

MaxTQ (SPK, N, L)

That is, find the maximum of the torque model (TQ) as a function of spark (SPK), engine speed (N), and load (L). You will use the NOXFLOW model to constrain these optimization problems.

For any optimization, you need to load one or more models. You can use the CAGE Import tool to import models from Model Browser projects (see "Import Models and Calibration Items Using CAGE Import Tool" in the CAGE documentation). For this tutorial you can load a CAGE project from the mbctraining directory that contains two models for the optimization problems. Load the project containing models to optimize as follows:

1   Select **File** > **Open Project** (or the toolbar button) to choose the tradeoffInit.cag file, found in the matlab\toolbox\mbc\mbctraining directory, then click **OK**.

    The tradeoffInit.cag project contains two models and all the variables necessary for this tutorial. For more information about how to set up models and variables, see "Calibration Setup" in the CAGE documentation.

2   CAGE displays the Models view. You can view your models at any time by clicking the **Models** button in the **Data Objects** pane.

    Observe that the project you have opened contains two models: TQ_Model and NOXFLOW_Model. In this tutorial you use these models to optimize torque values subject to emissions constraints.

3   To view the items in the Variable Dictionary, click the **Variable Dictionary** button in the **Data Objects** pane.
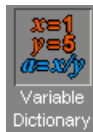
    The **Variable Dictionary** view appears, displaying the variables, constants, and formulas in the current project. The project already has the relevant variables defined, so you do not need to import a variable dictionary. Note that the variables have ranges and set points defined.