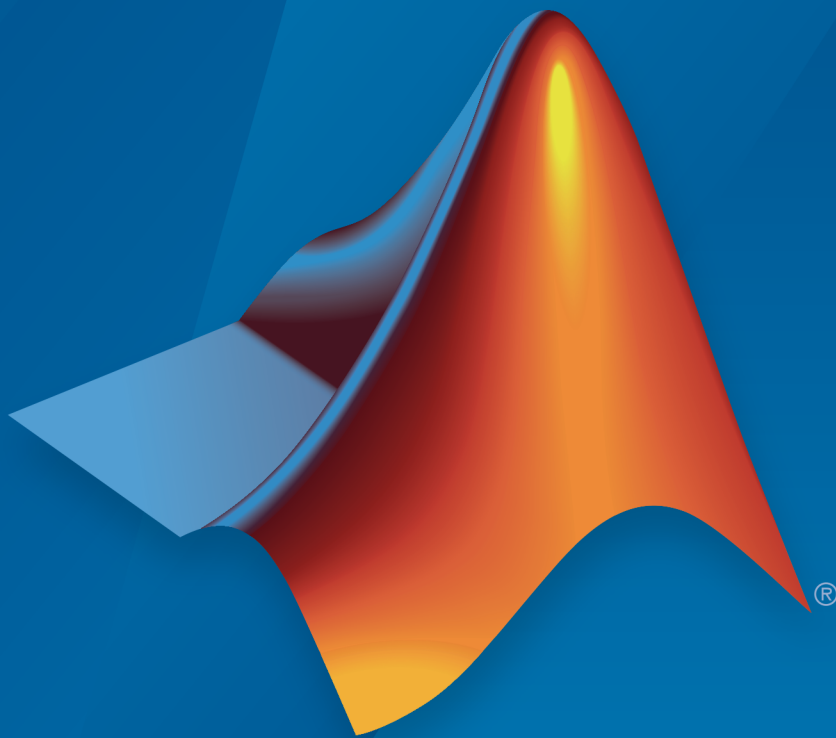


# Model-Based Calibration Toolbox™

## Getting Started Guide



MATLAB® & SIMULINK®

R2015a

 MathWorks®

## How to Contact MathWorks



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



Phone: 508-647-7000



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

*Model-Based Calibration Toolbox™ Getting Started Guide*

© COPYRIGHT 2005–2015 by The MathWorks, Inc.

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

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

### Trademarks

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

### Patents

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

## Revision History

|                |             |   |
|----------------|-------------|---|
| November 2005  | Online only | New for Version 3.0 (Release 14SP3+)      |
| September 2006 | Online only | Version 3.1 (Release 2006b)               |
| March 2007     | Online only | Version 3.2 (Release 2007a)               |
| September 2007 | Online only | Revised for Version 3.3 (Release 2007b)   |
| March 2008     | Online only | Revised for Version 3.4 (Release 2008a)   |
| October 2008   | Online only | Revised for Version 3.4.1(Release 2008a+) |
| October 2008   | Online only | Revised for Version 3.5 (Release 2008b)   |
| March 2009     | Online only | Revised for Version 3.6 (Release 2009a)   |
| September 2009 | Online only | Revised for Version 3.7 (Release 2009b)   |
| March 2010     | Online only | Revised for Version 4.0 (Release 2010a)   |
| September 2010 | Online only | Revised for Version 4.1 (Release 2010b)   |
| April 2011     | Online only | Revised for Version 4.2 (Release 2011a)   |
| September 2011 | Online only | Revised for Version 4.3 (Release 2011b)   |
| March 2012     | Online only | Revised for Version 4.4 (Release 2012a)   |
| September 2012 | Online only | Revised for Version 4.5 (Release 2012b)   |
| March 2013     | Online only | Revised for Version 4.6 (Release 2013a)   |
| September 2013 | Online only | Revised for Version 4.6.1 (Release 2013b) |
| March 2014     | Online only | Revised for Version 4.7 (Release 2014a)   |
| October 2014   | Online only | Revised for Version 4.8 (Release 2014b)   |
| March 2015     | Online only | Revised for Version 4.8.1 (Release 2015a) |



**Introduction**

**1**

|  |     |
|--|-----|
| <b>Model-Based Calibration Toolbox Product Description</b> . . . . | 1-2 |
| Key Features . . . . .   | 1-2 |
| <b>What Is Model-Based Calibration?</b> . . . . .                  | 1-3 |
| Designs and Modeling in the Model Browser . . . . .                | 1-3 |
| Calibration Generation in CAGE . . . . .                           | 1-4 |
| <b>Limitations</b> . . . . .                                       | 1-6 |
| Waitbar May Have Transparent Background . . . . .                  | 1-6 |

**Gasoline Engine Calibration Case Study**

**2**

|   |      |
|---|------|
| <b>Gasoline Case Study Overview</b> . . . . .               | 2-2  |
| Case Study Introduction . . . . .                           | 2-2  |
| Why Use Design of Experiment and Engine Modeling? . . . . . | 2-3  |
| Problem Definition . . . . .                                | 2-4  |
| Introduction to Two-Stage Modeling . . . . .                | 2-4  |
| <b>Designing the Experiment</b> . . . . .                   | 2-6  |
| Overview of Design Process . . . . .                        | 2-6  |
| Specifying Model Inputs . . . . .                           | 2-6  |
| Creating Designs . . . . .                                  | 2-7  |
| Data Source . . . . .                                       | 2-13 |
| <b>Selecting Data and Models to Fit</b> . . . . .           | 2-14 |
| <b>Viewing and Filtering Data</b> . . . . .                 | 2-19 |
| View Data . . . . .   | 2-19 |

|   |      |
|---|------|
| Filter Data .....   | 2-23 |
| <b>How Is a Two-Stage Model Constructed?</b> .....          | 2-26 |
| <b>Selecting Local Models</b> .....                         | 2-30 |
| <b>Viewing the Boundary Model</b> .....                     | 2-33 |
| <b>Selecting Global and Two-Stage Models</b> .....          | 2-38 |
| Inspect the Global Models .....                             | 2-38 |
| Create Multiple Models to Compare .....                     | 2-40 |
| Create a Two-Stage Model .....                              | 2-44 |
| Evaluate Other Response Models .....                        | 2-46 |
| <b>Using Validation Data</b> .....                          | 2-48 |
| <b>Exporting the Models</b> .....                           | 2-50 |
| <b>Optimized Calibration</b> .....                          | 2-51 |
| Problem Definition .....                                    | 2-51 |
| Benefits of Automated Calibration .....                     | 2-52 |
| <b>Importing Additional Models into CAGE</b> .....          | 2-54 |
| <b>Setting Up Calibration Tables to Fill</b> .....          | 2-57 |
| Altering Variable Ranges .....                              | 2-57 |
| Setting Up Tables .....                                     | 2-57 |
| <b>Setting Up the Optimization</b> .....                    | 2-60 |
| <b>Defining Variable Values</b> .....                       | 2-63 |
| <b>Running the Optimization</b> .....                       | 2-65 |
| <b>Setting Up the Sum Optimization</b> .....                | 2-70 |
| Setting Up the Optimization Initial Values and Objective .. | 2-70 |
| Creating Table Gradient Constraints .....                   | 2-71 |
| Running the Sum Optimization .....                          | 2-73 |
| <b>Filling Tables with Optimization Results</b> .....       | 2-75 |
| <b>MBT Spark Estimator Problem</b> .....                    | 2-76 |
| What Is an Estimator? .....                                 | 2-76 |

|                                      |      |
|--------------------------------------|------|
| View the Feature .....               | 2-76 |
| View Variables .....                 | 2-78 |
| Edit and Import Boundary Model ..... | 2-79 |
| Use the Feature Fill Wizard .....    | 2-80 |
| Inspect Results .....                | 2-84 |
| CAGE Import Tool .....               | 2-87 |

## Diesel Engine Calibration Case Study

# 3

|   |             |
|---|-------------|
| <b>Diesel Case Study Overview .....</b>                     | <b>3-2</b>  |
| Introduction .....  | 3-2         |
| Problem Definition .....                                    | 3-2         |
| <b>Design of Experiment .....</b>                           | <b>3-4</b>  |
| Introducing Design of Experiment .....                      | 3-4         |
| Constraining the Design .....                               | 3-6         |
| Creating Candidate Designs .....                            | 3-13        |
| Data Collection .....                                       | 3-14        |
| <b>Modeling .....</b>                                       | <b>3-15</b> |
| Overview of Modeling Process .....                          | 3-15        |
| Building Models .....                                       | 3-15        |
| Building and Evaluating Alternative Models .....            | 3-18        |
| Creating Boundary Models .....                              | 3-24        |
| Export to CAGE .....  | 3-27        |
| <b>Optimized Calibration .....</b>                          | <b>3-29</b> |
| Problem Definition .....                                    | 3-29        |
| Benefits of Automated Calibration .....                     | 3-30        |
| <b>Importing Models of Engine Responses into CAGE .....</b> | <b>3-32</b> |
| <b>Defining Additional Variables and Models .....</b>       | <b>3-33</b> |
| <b>Setting Up Calibration Tables to Fill .....</b>          | <b>3-35</b> |
| <b>Setting Up the Point Optimization .....</b>              | <b>3-36</b> |
| <b>Setting Up Constraints .....</b>                         | <b>3-39</b> |

|  |             |
|--|-------------|
| <b>Defining Variable Values</b> .....                        | <b>3-43</b> |
| <b>Running the Optimization</b> .....                        | <b>3-45</b> |
| <b>Setting Up the Sum Optimization</b> .....                 | <b>3-47</b> |
| Setting Up Initial Values and Sum Objective .....            | <b>3-47</b> |
| Creating the Brake Specific NOx Constraint .....             | <b>3-48</b> |
| Setting Weights for the Sum Objective and Constraint . . . . | <b>3-49</b> |
| Set Parameters and Run Optimization .....                    | <b>3-51</b> |
| <b>Filling Tables with Optimization Results</b> .....        | <b>3-52</b> |

## **4 Point-by-Point Diesel Engine Calibration Case Study**

### **4**

|  |             |
|--|-------------|
| <b>Point-by-Point Diesel Case Study Overview</b> .....       | <b>4-2</b>  |
| What Is Point-by-Point Modeling? .....                       | <b>4-2</b>  |
| Engine Calibration Specifications and Problem Description .. | <b>4-2</b>  |
| Required Tasks .....   | <b>4-4</b>  |
| <b>Create Designs and Models Programmatically</b> .....      | <b>4-5</b>  |
| Overview of Programmatic Design and Modeling .....           | <b>4-5</b>  |
| Creating Designs and Models Programmatically .....           | <b>4-5</b>  |
| <b>Verify and Refine Models</b> .....                        | <b>4-7</b>  |
| Open the Project and View Designs .....                      | <b>4-7</b>  |
| Analyze and Refine Local Fits .....                          | <b>4-7</b>  |
| <b>Point-by-Point Optimization Overview</b> .....            | <b>4-9</b>  |
| <b>Create Point Optimizations in CAGE</b> .....              | <b>4-10</b> |
| Introduction .....   | <b>4-10</b> |
| Load Models to Optimize .....                                | <b>4-10</b> |
| Create Part Load Point Optimization .....                    | <b>4-11</b> |
| Create Full-Load Point Optimization .....                    | <b>4-15</b> |
| <b>Create Multiregion Sum Optimization</b> .....             | <b>4-19</b> |
| Introduction .....   | <b>4-19</b> |
| Create Data Set from Point Optimization Results .....        | <b>4-19</b> |
| Create Sum Optimization .....                                | <b>4-20</b> |



|   |             |
|---|-------------|
| Add Application Point Sets . . . . .                              | 4-21        |
| Set Up Constraints . . . . .                                      | 4-23        |
| Define Weights and Fixed Variables and Run Optimization . . . . . | 4-25        |
| <b>Use Optimization Results . . . . .</b>                         | <b>4-27</b> |
| Introduction . . . . .  | 4-27        |
| Create Tables to Fill . . . . .                                   | 4-27        |
| Fill Tables from Optimization Results . . . . .                   | 4-28        |
| Export Tables . . . . .   | 4-29        |

## Composite Models and Modal Optimizations

# 5

|   |            |
|---|------------|
| <b>Introducing Composite Models and Modal Optimization . . . . .</b>      | <b>5-2</b> |
| <b>Composite Model and Modal Optimization Projects . . . . .</b>          | <b>5-3</b> |
| Gasoline Example with Four Cylinder and Eight Cylinder<br>Modes . . . . . | 5-3        |
| Diesel Example with Low and High EGR Modes . . . . .                      | 5-3        |
| Composite Model Example with Separate Tables for Each<br>Mode . . . . .   | 5-4        |

## Design and Modeling Scripts

# 6

|   |             |
|---|-------------|
| <b>Introduction to the Command-Line Interface . . . . .</b>         | <b>6-2</b>  |
| <b>Automate Design and Modeling With Scripts . . . . .</b>          | <b>6-3</b>  |
| Processes You Can Automate . . . . .                                | 6-3         |
| Engine Modeling Scripts . . . . .                                   | 6-5         |
| <b>Understanding Model Structure for Scripting . . . . .</b>        | <b>6-6</b>  |
| Projects and Test Plans for Model Scripting . . . . .               | 6-6         |
| Response Model Scripting . . . . .                                  | 6-6         |
| Boundary Model Scripting . . . . .                                  | 6-8         |
| <b>How the Model Tree Relates to Command-Line Objects . . . . .</b> | <b>6-10</b> |

|   |      |
|---|------|
| <b>Multi-Injection Diesel Calibration Workflow</b> .....                  | 7-2  |
| Multi-Injection Diesel Problem Definition .....                           | 7-2  |
| Engine Calibration Workflow .....   | 7-7  |
| Air-System Survey Testing .....   | 7-8  |
| Multi-Injection Testing .....   | 7-9  |
| Data Collection and Physical Modeling .....                               | 7-10 |
| Statistical Modeling .....  | 7-11 |
| Optimization Using Statistical Models .....                               | 7-12 |
| Case Study Example Files .....  | 7-20 |
| <b>Design of Experiment</b> .....   | 7-22 |
| Benefits of Design of Experiment .....                                    | 7-22 |
| Air-System Survey Testing .....   | 7-22 |
| Create Designs and Collect Data .....                                     | 7-23 |
| Fit a Boundary Model to Air Survey Data .....                             | 7-29 |
| Use the Air Survey and Boundary Model to Create the Final<br>Design ..... | 7-32 |
| Multi-Injection Testing .....   | 7-39 |
| <b>Statistical Modeling</b> .....   | 7-40 |
| Examine the Test Plans for Point-by-Point Models .....                    | 7-40 |
| Examine Response Models .....   | 7-43 |
| <b>Optimization</b> .....   | 7-46 |
| Optimization Overview .....   | 7-46 |
| Set Up Models and Tables for Optimization .....                           | 7-46 |
| Examine the Point Optimization Setup .....                                | 7-49 |
| Examine the Point Optimization Results .....                              | 7-53 |
| Create Sum Optimization from Point Optimization .....                     | 7-55 |
| Fill Tables from Optimization Results .....                               | 7-61 |
| Examine the Multiobjective Optimization .....                             | 7-71 |

|   |     |
|---|-----|
| <b>Empirical Engine Modelling</b> ..... | 8-2 |
|---|-----|

|   |      |
|---|------|
| <b>About Two-Stage Models</b> .....                     | 8-3  |
| <b>Start the Toolbox and Load Data</b> .....            | 8-5  |
| <b>Set Up the Model</b> .....                           | 8-7  |
| Specifying Model Inputs .....                           | 8-7  |
| Changing the Local Model Type .....                     | 8-10 |
| <b>Verify the Model</b> .....                           | 8-12 |
| Verifying the Local Model .....                         | 8-12 |
| Verifying the Global Model .....                        | 8-14 |
| Selecting the Two-Stage Model .....                     | 8-16 |
| Comparing the Local Model and the Two-Stage Model ..... | 8-21 |
| Maximum Likelihood Estimation .....                     | 8-22 |
| Response Node .....                                     | 8-24 |
| <b>Export the Model</b> .....                           | 8-27 |
| <b>Create Multiple Models to Compare</b> .....          | 8-29 |
| Methods For Creating More Models .....                  | 8-29 |
| Creating New Local Models .....                         | 8-29 |
| Adding New Response Features .....                      | 8-32 |
| Comparing Models .....                                  | 8-33 |
| Creating New Global Models .....                        | 8-34 |
| Creating Multiple Models Using Build Models .....       | 8-37 |

## Design of Experiment

# 9

|   |     |
|---|-----|
| <b>Design of Experiment</b> .....               | 9-2 |
| Why Use Design of Experiment? .....             | 9-2 |
| Design Styles .....                             | 9-3 |
| Get Started With The Design Editor .....        | 9-3 |
| <b>Set Up a Model and Create a Design</b> ..... | 9-5 |
| Setting Up Model Inputs .....                   | 9-5 |
| Open the Design Editor .....                    | 9-5 |
| Creating a New Design .....                     | 9-6 |

|  |      |
|--|------|
| <b>Create Optimal Designs</b> .....                    | 9-8  |
| Introducing Optimal Designs .....                      | 9-8  |
| Start Point Tab .....                                  | 9-9  |
| Candidate Set Tab .....                                | 9-10 |
| Algorithm Tab .....                                    | 9-13 |
| <b>View Design Displays</b> .....                      | 9-15 |
| <b>Use the Prediction Error Variance Viewer</b> .....  | 9-18 |
| Introducing the Prediction Error Variance Viewer ..... | 9-18 |
| Improving the Design .....                             | 9-21 |
| <b>Create Classical Designs</b> .....                  | 9-24 |
| Adding a Classical Design .....                        | 9-24 |
| Classical Design Browser .....                         | 9-26 |
| Setting Up and Viewing a Classical Design .....        | 9-27 |
| <b>Use the Design Evaluation Tool</b> .....            | 9-30 |
| <b>Create Space-Filling Designs</b> .....              | 9-33 |
| <b>Apply Constraints</b> .....                         | 9-35 |
| <b>Save Designs</b> .....                              | 9-41 |

## Data Editor for Modeling

# 10

|   |       |
|---|-------|
| <b>Data Manipulation for Modeling</b> ..... | 10-2  |
| <b>Load Data</b> .....                      | 10-3  |
| Entering the Data Editor .....              | 10-3  |
| Loading a Data File .....                   | 10-4  |
| <b>View and Edit the Data</b> .....         | 10-6  |
| Viewing Data .....                          | 10-6  |
| Using Notes to Sort Data for Plotting ..... | 10-8  |
| Removing Outliers and Problem Tests .....   | 10-9  |
| Reordering and Editing Data .....           | 10-10 |

|   |       |
|---|-------|
| <b>Create New Variables and Filters</b> .....               | 10-12 |
| Adding New Variables .....                                  | 10-12 |
| Applying a Filter .....                                     | 10-13 |
| Sequence of Variables .....                                 | 10-15 |
| Deleting and Editing Variables and Filters .....            | 10-16 |
| <b>Store Variables, Filters, and Plot Preferences</b> ..... | 10-17 |
| <b>Define Test Groupings</b> .....                          | 10-19 |
| <b>Match Data to Experimental Designs</b> .....             | 10-23 |
| Introducing Matching Data to Designs .....                  | 10-23 |
| Tolerances and Cluster Information .....                    | 10-26 |
| Understanding Clusters .....                                | 10-29 |

## Feature Calibration

# 11

|                                       |       |
|---------------------------------------|-------|
| <b>Feature Calibration</b> .....      | 11-2  |
| What Are Feature Calibrations? .....  | 11-2  |
| Start CAGE .....                      | 11-4  |
| Set Up Variables .....                | 11-4  |
| Set Up Models .....                   | 11-7  |
| Set Up a New Feature .....            | 11-9  |
| Set Up the Strategy .....             | 11-10 |
| Set Up the Tables .....               | 11-12 |
| Process For Feature Calibration ..... | 11-14 |
| Calibrate the Normalizers .....       | 11-15 |
| Calibrate the Tables .....            | 11-19 |
| Calibrate the Feature .....           | 11-25 |
| Export Calibrations .....             | 11-30 |

## Tradeoff Calibration

# 12

|                                       |      |
|---------------------------------------|------|
| <b>Tradeoff Calibration</b> .....     | 12-2 |
| What Is a Tradeoff Calibration? ..... | 12-2 |

|   |      |
|---|------|
| Setting Up a Tradeoff Calibration .....   | 12-2 |
| Performing the Tradeoff Calibration ..... | 12-7 |

## Data Sets

### 13

|   |       |
|---|-------|
| <b>Compare Calibrations To Data</b> ..... | 13-2  |
| Setting Up the Data Set .....             | 13-2  |
| Comparing the Items in a Data Set .....   | 13-7  |
| Reassigning Variables .....               | 13-14 |

## Filling Tables from Data

### 14

|  |       |
|--|-------|
| <b>Fill Tables from Data</b> .....                 | 14-2  |
| Setting Up a Table and Experimental Data .....     | 14-2  |
| Filling the Table from the Experimental Data ..... | 14-8  |
| Selecting Regions of the Data .....                | 14-11 |
| Exporting the Calibration .....                    | 14-13 |

## Optimization and Automated Tradeoff

### 15

|   |       |
|---|-------|
| <b>Optimization and Automated Tradeoff</b> .....      | 15-2  |
| Import Models to Optimize .....                       | 15-2  |
| Optimization Example Problems .....                   | 15-4  |
| <b>Single-Objective Optimization</b> .....            | 15-5  |
| Process Overview .....                                | 15-5  |
| Using the Create Optimization from Model Wizard ..... | 15-6  |
| Setting Constraints and Operating Points .....        | 15-9  |
| Running the Optimization .....                        | 15-12 |
| Using Optimization Results to Fill Tables .....       | 15-14 |
| Using a Custom Fill Routine to Fill Tables .....      | 15-16 |

|  |              |
|--|--------------|
| <b>Multiobjective Optimization</b> .....                 | <b>15-18</b> |
| Setting Up and Running the Multiobjective Optimization . | <b>15-18</b> |
| Optimization Output View .....                           | <b>15-23</b> |
| Selecting Best Solutions .....                           | <b>15-28</b> |
| <br>   |              |
| <b>Sum Optimization</b> .....                            | <b>15-30</b> |
| What Is a Sum Optimization? .....                        | <b>15-30</b> |
| Procedure .....  | <b>15-31</b> |
| <br>   |              |
| <b>Automated Tradeoff</b> .....                          | <b>15-35</b> |





# 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

## Calibrate complex powertrain systems

Model-Based Calibration Toolbox provides apps and design tools for optimally calibrating complex powertrain systems using statistical modeling and numeric optimization. You can define test plans, develop statistical models, and generate calibrations and lookup tables for complex high-degree-of-freedom engines that would require exhaustive testing using traditional methods. By using the toolbox with MATLAB<sup>®</sup> and Simulink<sup>®</sup>, you can develop a process for systematically identifying the optimal balance of engine performance, emissions, and fuel economy, and reuse statistical models for control design, hardware-in-the-loop testing, or powertrain simulation.

## Key Features

- MBC Model Fitting app for designing experiments, fitting statistical models to engine data, and producing optimal calibrations
- Classical, space-filling, and optimal designs, based on Design-of-Experiments methodology, for creating optimized test plans
- Techniques for developing high-fidelity nonlinear statistical models from test data
- Linear regression and radial basis function modeling techniques for creating accurate fits to data
- Built-in and user-definable libraries of empirical model forms
- Boundary modeling to keep optimization results within the engine operating envelope
- MBC Optimization app for solving calibration problems at individual operating points or over drive cycles
- Generation of lookup tables from models, optimization results, or test data
- Calibration import and export links 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.

# Gasoline Engine Calibration Case Study

---

This case study provides a step-by-step guide to using the Model-Based Calibration Toolbox product to solve a gasoline engine calibration problem. This section discusses the following topics:

- “Gasoline Case Study Overview” on page 2-2
- “Designing the Experiment” on page 2-6
- “Selecting Data and Models to Fit” on page 2-14
- “Viewing and Filtering Data” on page 2-19
- “How Is a Two-Stage Model Constructed?” on page 2-26
- “Selecting Local Models” on page 2-30
- “Viewing the Boundary Model” on page 2-33
- “Selecting Global and Two-Stage Models” on page 2-38
- “Using Validation Data” on page 2-48
- “Exporting the Models” on page 2-50
- “Optimized Calibration” on page 2-51
- “Importing Additional Models into CAGE” on page 2-54
- “Setting Up Calibration Tables to Fill” on page 2-57
- “Setting Up the Optimization” on page 2-60
- “Defining Variable Values” on page 2-63
- “Running the Optimization” on page 2-65
- “Setting Up the Sum Optimization” on page 2-70
- “Filling Tables with Optimization Results” on page 2-75
- “MBT Spark Estimator Problem” on page 2-76

# Gasoline Case Study Overview

| In this section...  |
|---|
| “Case Study Introduction” on page 2-2                           |
| “Why Use Design of Experiment and Engine Modeling?” on page 2-3 |
| “Problem Definition” on page 2-4                                |
| “Introduction to Two-Stage Modeling” on page 2-4                |

## Case Study Introduction

This case study demonstrates how to systematically develop a set of optimal steady-state engine calibration tables using the Model-Based Calibration Toolbox product. This case-study uses a 2.2L inline 4 cylinder, naturally aspirated DOHC (Dual Overhead Cams) 4 valve per cylinder spark ignition (SI) engine equipped with dual-independent variable cam-phasing (DIVCP) hardware and electronic throttle.

Optimal steady-state engine calibration tables for intake cam phase, exhaust cam phase, and spark advance are developed as part of the case study process.

This example takes you through the following steps:

- 1** Create a design for your experiment — see “Designing the Experiment” on page 2-6.
- 2** Import the resulting data (taken using the design) and choose responses to model — see “Selecting Data and Models to Fit” on page 2-14
- 3** Examine and filter the data in preparation for modeling, and create the models — see “Viewing and Filtering Data” on page 2-19.
- 4** Evaluate the statistical models based on the data.
  - a** “How Is a Two-Stage Model Constructed?” on page 2-26
  - b** “Selecting Local Models” on page 2-30
  - c** “Viewing the Boundary Model” on page 2-33
  - d** “Selecting Global and Two-Stage Models” on page 2-38
  - e** “Using Validation Data” on page 2-48
- 5** Export these models to the CAGE part of the toolbox to generate optimal calibration tables — see “Exporting the Models” on page 2-50.



- 6** The Model Browser section of the case study involves design of experiment, data handling, and model construction and export. In the CAGE browser section of the case study you use the models to complete the optimization of the calibration tables, see “Optimized Calibration” on page 2-51.

The following sections introduce the benefits of applying model-based calibration methods to solve this case study problem:

- “Why Use Design of Experiment and Engine Modeling?” on page 2-3
- “Problem Definition” on page 2-4
- “Introduction to Two-Stage Modeling” on page 2-4

## **Why Use Design of Experiment and Engine Modeling?**

These approaches can be used to ensure that you develop optimal engine calibrations for complex engines with many controllable variables (such as variable valve timing, variable valve lift, and cylinder deactivation) at minimum cost and time.

Test bed time is expensive, and Design of Experiments methodology can help you choose the most effective points to run to get the maximum information in the shortest time. You can break the exponential dependency between the complexity of the engine (number of inputs) and the cost of testing (number of tests). You can collect the most statistically useful data, and just enough of it to fit the models.

Experimental design test points can be constrained based on previous experience to avoid damaging expensive engine hardware prototypes at unrealistic operating points.

The act of statistically modeling engine data can help identify the effect of interactions between calibration settings and engine performance, which can be vital to understanding how to optimally meet emissions constraints.

Accurate statistical models of engine data can also be used to develop calibration tables that have smooth transitions between the operating range of the engine and the edge regions of calibration tables where the engine will not be operated.

Optimal calibrations can be generated from statistical engine models in a methodical, repeatable process to ensure that maximum performance is achieved subject to emissions, driveability, and material limit constraints.

### Problem Definition

The aim of this case study is to produce optimized tables for

- Intake Cam Phase
- Exhaust Cam Phase
- Spark Timing Schedules

as a function of Load and rpm, subject to the following constraints

- Constrain solutions to lie within the boundary constraint model (to keep the engine within its operating region)
- Constrain cam phase solutions so they do not change by more than  $10^\circ$  between table cells (that is, no more than  $10^\circ$  per 500 RPM change and per 0.1 load change).
- Constrain residual fraction  $\leq 25\%$  at each drive cycle point (to ensure stable combustion). Residual fraction is the percentage of burned gas mass in the cylinder at intake valve close, relative to the total mass in the cylinder at intake valve close. Constraining maximum residual fraction is a simple and reasonable way of ensuring stable combustion. Residual fraction =  $100 * \text{Burned Gas Mass from Last Cycle} / (\text{Burned Gas Mass From Last Cycle} + \text{Fresh Air Mass})$

To produce these tables, you need to make accurate models of the behavior of torque, exhaust temperature, and residual fraction at different values of speed, throttle area, spark, and cam timings. You need engine data to build these models, so the first step is constructing an experimental design to collect the most useful set of points.

Before you can design an experiment you need to set up a two-stage test plan and define your model inputs and model type.

### Introduction to Two-Stage Modeling

What is a two-stage test plan? You use a test plan to set up models in the Model Browser. The two stages refer to the way that engine data is often collected. For example, in each test, spark (the local variable) is swept while the other variables (such as speed and load) are held constant — these are referred to as global variables. Each test is taken at a different point in the global variables. Building the statistical models to take into account these individual sweeps makes it possible to incorporate engineering knowledge in the process. You can see plots of torque/spark sweeps, and use variables such as MBT (maximum brake torque) in modeling, rather than solely abstract mathematical

properties of curves. You can then apply previous knowledge about the expected behavior of these variables to help you select good models.

You can easily identify outliers when you can see the sweep in which they were taken. The Model Browser allows you to visualize the data in a way that can help you identify and investigate suspect sweeps, and decide what kind of models will produce the best fit to the shapes of the data. The more controllable variables there are in an engine the more useful it is to have these visual aids to investigate complex data. Constructing models to take into account the way the data is collected helps build good models that you can have more confidence in. Statistically, it is the correct thing to do as it allows you to partition the errors within sweeps and the errors between sweeps separately.

You use a two-stage test plan to build your models because this data is suited to it. Spark is varied as the other variables are held constant, so the data is collected in a hierarchical structure; your models attempt to capture this information. You come to more detail on how this two-stage model is constructed after creating a design and obtaining data.

# Designing the Experiment

### In this section...

“Overview of Design Process” on page 2-6

“Specifying Model Inputs” on page 2-6

“Creating Designs” on page 2-7

“Data Source” on page 2-13


## Overview of Design Process

Creating a design in the Model Browser comprises several steps. You need to open the tool, select a two-stage template, and enter the ranges and names of the input variables. Then you can create an initial design and set up the constraints on the input space. These constraints will be the same for all designs. From this constrained design, a series of child designs can be made with varying numbers of points added and slightly different models used. The final design can be chosen by comparing statistics of the various child designs and considering how many points you can afford to run. These steps are described next.

## Specifying Model Inputs

- 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.

The models you are building are intended to predict the torque, fuel flow, and manifold pressure of the engine as a function of spark angle at specified operating points defined by the engine's speed, load, and cam timings. The input to the local model is the spark angle.

- 4 Set up your local model input.

- a Set **Symbol** to S.
  - b Set **Signal** to SPARK. This is optional and matches the raw data.
  - c Set the range you want to model by changing **Max** to 50 (and leave **Min** at 0).
- 5 Set up your global inputs. The global inputs are the variables that are held constant at each operating point while spark is swept. In this case, these global variables are engine speed, scaled throttle area, intake cam angle, and exhaust cam angle.

By default, there is one input to the global model. Because this engine model has four input factors, you need to edit the input factors as follows:

- a Click the up arrow button to increase the **Number of factors** setting to four.
- b Edit the four factors to create the engine model input. In each case, change the symbols, signal names, and ranges to the following:

| Symbol | Signal  | Min  | Max  |
|--------|---------|------|------|
| N      | SPEED   | 500  | 6000 |
| L      | LOAD    | 0.05 | 0.95 |
| ICP    | INT_ADV | -5   | 50   |
| ECP    | EXH_RET | -5   | 50   |

Load = aircharge/maximum aircharge.

Cam angles are in units of degrees crankshaft, with intake values indicating advance from base timing, and exhaust values indicating retard from base timing.

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

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

## Creating Designs

Now you have set up the modeling test plan you can create an initial design and set up the constraints on the input space — these will be the same for all designs. From this constrained design, a series of child designs can be made with varying numbers of points added and slightly different models used. The final design can be chosen by comparing


statistics of the various child designs and considering how many points you can afford to run.

- 1 You will import a boundary model from an example file. In this way, you can use a boundary constraint from a previous investigation on a similar engine to constrain new designs. To locate example files, in MATLAB, change the current folder to `mbctraining`:

```
cd(fullfile(matlabroot, 'toolbox', 'mbc', 'mbctraining'))
```

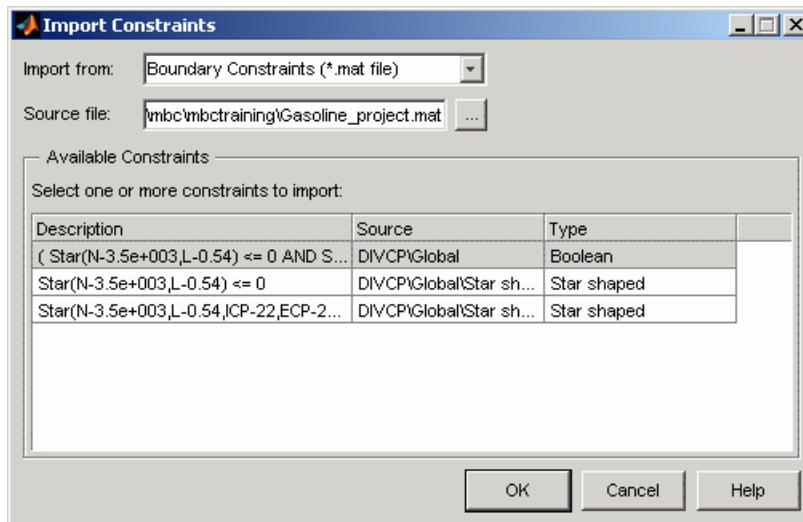
- 2 In the Model Browser test plan view, in the **Common Tasks** pane, click **Design experiment**.

The Design Editor appears.

- 3 Click the  button in the toolbar or select **File > New Design**. A new node called **Hybrid Radial Basis Function Design\_1** appears.

The new **Hybrid Radial Basis Function Design\_1** node is automatically selected. An empty Design Table (or any view you last used in the Design Editor) appears because you have not yet chosen a design.

- 4 Constrain the design space. Select **File > Import Constraints**. The Import Constraints dialog box appears.
- 5 In the **Import from** list, select **Boundary Constraints (.mat file)**.
- 6 Browse to the file `Gasoline_project.mat`, found in `matlab\toolbox\mbc\mbctraining`.
- 7 Click to select the **Boolean** type constraint as shown. This is the combination of both boundary constraint models.



- 8 Click **OK** to import the boundary constraint.

Click **OK** in the following data matching dialog as all the signal names are automatically selected in this case.

- 9 Examine the constrained design space by right-clicking the title bar of a Design Table view and selecting **Current View > 3D Constraints**.
- 10 Select **Design > Space Filling > Design Browser**, or click the **Space Filling Design** button on the toolbar.

The Space Filling Design Browser appears.

Space-filling designs are best 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. Space-filling designs are also best for radial basis function models. You can use a mix-and-match approach: start with a space-filling design to survey the space, then continue testing with an optimal design once you have more understanding of the response and constraints. Once you have an idea of what model type will fit the response best, you can optimally add points in the most efficient places for the most robust model fit.

The most important thing to decide is how many design points you want. Testing is expensive and time-consuming, so you need to bear in mind how many points you have time for. When you consider the number of points, you also need to remember that a sweep will be done at each point and this will take some time. Do you need to allow time to fix problems or redo experimental points that can't be achieved?

- 11 Enter **800** for the **Number of points** and press **Enter**. A space-filling design is constructed, using the latin hypercube sampling method. Click the 3-D and 2-D tabs to examine the plots of new design distribution.
- 12 Click **Generate** to create a different design, and repeat until you achieve approximately 200 points in the **Size of constrained design** reported above the preview. This iteration is necessary because the space-filling design uses the whole variable space and some of the points will be removed if they fall outside the constraint. The preview is identical to the final design.

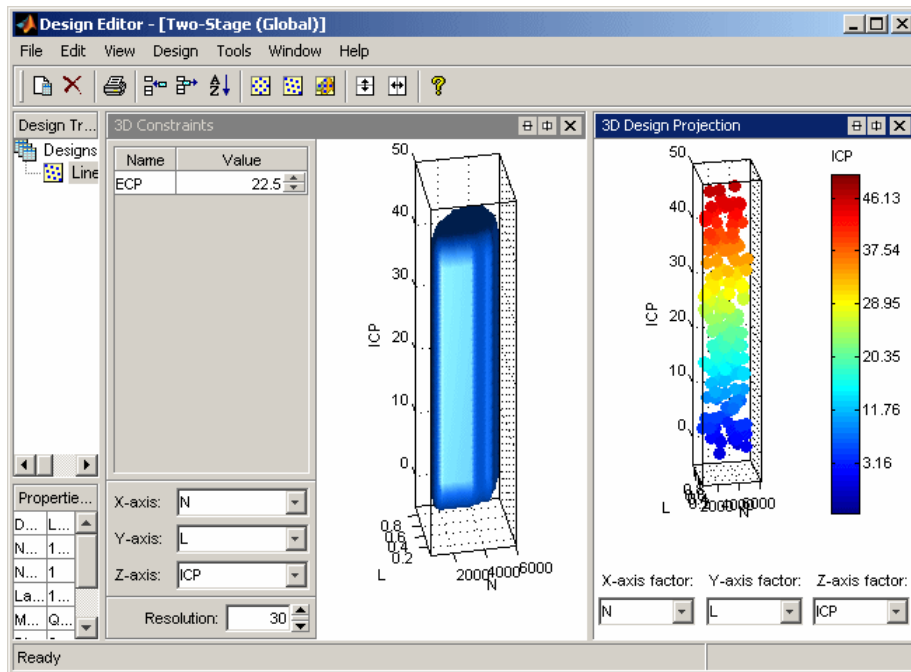
---

**Note:** You will import the example design for this case study in a later step, so don't worry about getting the exact number of points in the designs you create.

---

- 13 Click **OK** when you are satisfied with the design.
- 14 Right-click a view and select **Split View > 3D Design Projection** to view the design points.






- 15 Add another space-filling design for some points with parked cam phasers. These points are important because we need an accurate model when cams are parked.

- a Click the  button in the toolbar to add a new design. The view switches to the new design in the tree. Rename it CAM\_Parked. The design inherits the same constraints as the parent design.
- b Select **Design > Space Filling > Design Browser**, or click the **Space Filling Design** button on the toolbar.

In the dialog that appears, choose to replace the current points with a new design and click **OK**.

- c Enter a number of points (try 40) and click **Generate** until you achieve a constrained design of about 10 points, and click **OK**.
- d Display the design as a table, and edit the values of ICP and ECP in the new design to be zero.

- 16 Select **File > Merge Designs**. Select both your designs in the list, leave the **Merge to new design** option button selected, and click **OK**. Examine the merged design in the table view to confirm the parked cam points (ICP and ECP values of 0) are at the end of the list of design points.
- 17 Add another space-filling design for collecting validation data.
  - a Click the  button in the toolbar to add a new design. The view switches to the new design in the tree. You could rename it **Validation**. The design inherits the same constraints as the parent design.
  - b Select **Design > Space Filling > Design Browser**, or click the **Space Filling Design** button on the toolbar.

In the dialog that appears, choose to replace the current points with a new design and click **OK**.
  - c Enter a number of points (try 100) and click **Generate** until you achieve a constrained design of about 25 points, and click **OK**.
  - d To add a point where the cams are parked, select **Edit > Add Point**.

In the dialog that appears, select **User-specified** from the **Augment method** list, edit ICP and ECP to zero, and click **OK**.

You can use the Design Editor to make a selection of child designs to compare. When you have chosen the best design you can export it to file. In this case, you can import the example design for this case study. This design was used to collect the data for this case study, and you will later match these design points to data. Import the design as follows:

- 1 Select **File > Import Design**.
- 2 Leave the default **Design Editor file (.mvd)** in the **Import from** drop-down menu.
- 3 Browse to the design file **DIVCP.mvd**, found in **matlab\toolbox\mbc\mbctraining**, select it and click **Open**, then click **OK** to import the design. A new design node appears.
- 4 Rename the imported design (click, and press **F2**) **DIVCP**. You can right-click to change the Current View to examine the design points in 2D and 3D.

If you wanted to **Match selected data to design** when fitting models to data in later steps, you must select as design as best to use it. For this example, do not select any design as best.

5 Close the Design Editor.

## Data Source

The data was collected using a constrained space-filling design on speed, load, intake cam phase, and exhaust cam phase. The points specified in the design were measured using the GT-Power engine simulation tool from Gamma Technologies (see <http://www.gtisoft.com>).

Simulink and StateFlow<sup>®</sup> simulation tools controlled the GT-Power model, running on a cluster of 14 desktop PC machines. The simulation time for the testing was 20 hours. The GT-Power model used predictive combustion, which gives good realistic results but is computationally expensive. Ten cycles were run at each spark advance setting after attaining steady-state speed/load conditions.

For next steps, see “Selecting Data and Models to Fit” on page 2-14.

# Selecting Data and Models to Fit

Remember that the aim of this case study is to produce optimized tables for:

- Intake cam phase
- Exhaust cam phase
- Spark timing schedules

These tables are all functions of load and rpm, subject to constraints of operating region and residual fraction.

To produce these tables, you need to make accurate models of the behavior of torque, exhaust temperature, and residual fraction at different values of speed, load, spark, and cam timings. You have set the local model input as spark, and the global model inputs as engine speed, load, intake cam phase, and exhaust cam phase. Therefore, the responses you want to model are:

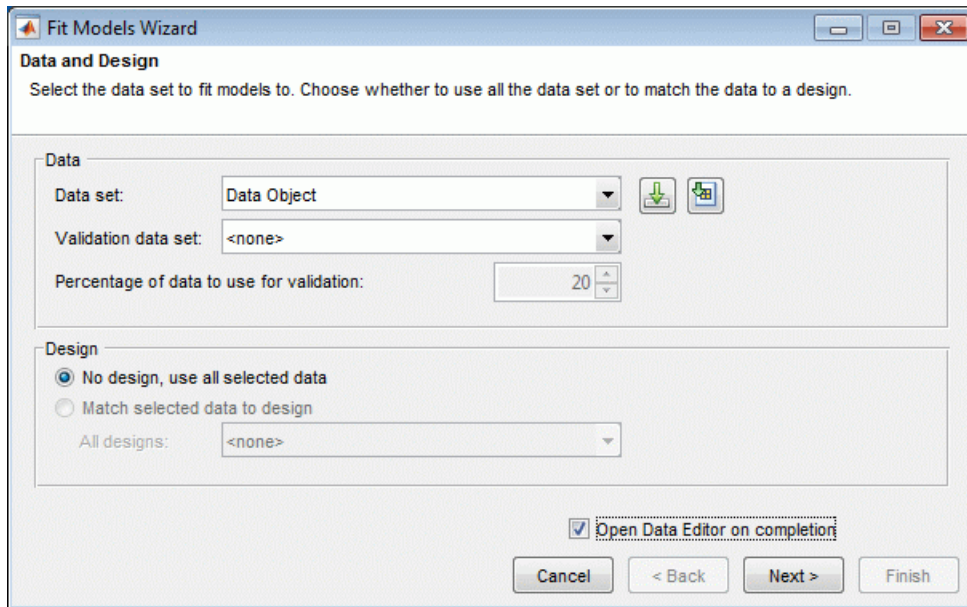
- Torque
- Exhaust Temperature
- Residual fraction

Follow these steps to select data and model types to fit.

- 1** In the Model Browser, click the test plan node, **Two -Stage**, in the **All Models** tree to go to the test plan view.
- 2** In the **Common Tasks** pane, click **Fit models**.

The Fit Models Wizard opens.

- 3** In the **Data** pane, next to **Data set**, use the browse button to find and select the **DIVCP\_Main\_DoE\_Data.xls** data file, found in `matlab\toolbox\mbc\mbctraining`. This example data file results from the example experimental design. Click **Open**.
- 4** In the Fit Models Wizard, select the **Open Data Editor on completion** check box.



The image shows a software dialog box titled "Fit Models Wizard" with a sub-header "Data and Design". The main instruction is "Select the data set to fit models to. Choose whether to use all the data set or to match the data to a design." The dialog is divided into two sections: "Data" and "Design".

**Data Section:**

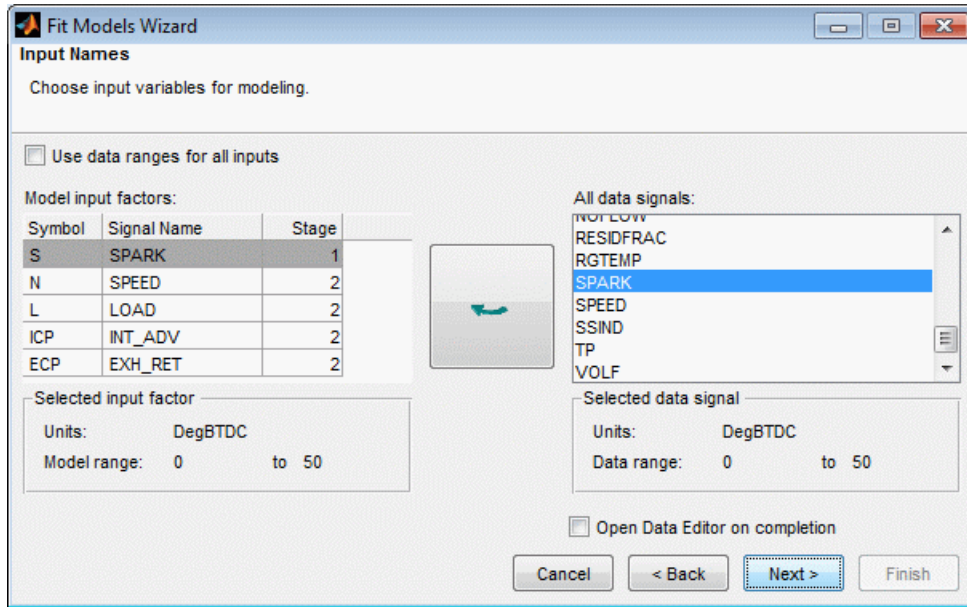
- "Data set:" dropdown menu is set to "Data Object".
- "Validation data set:" dropdown menu is set to "<none>".
- "Percentage of data to use for validation:" is a numeric spinner set to "20".

**Design Section:**

- Radio button "No design, use all selected data" is selected.
- Radio button "Match selected data to design" is unselected.
- "All designs:" dropdown menu is set to "<none>".

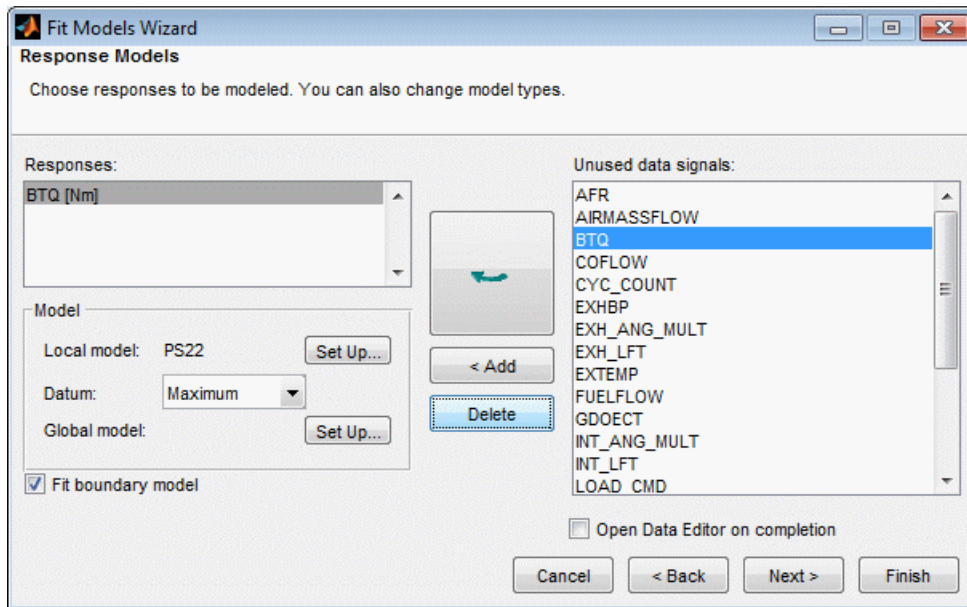
At the bottom right, there is a checked checkbox labeled "Open Data Editor on completion". Below this are four buttons: "Cancel", "< Back", "Next >", and "Finish".

- 5 Click **Next**.
- 6 The wizard tries to match each symbol to a data signal (shown as **Signal Name** in the model input factors list). Make sure **SPARK**, **SPEED**, **LOAD**, **INT\_ADV** and **EXH\_RET** are selected, as shown.



Click **Next**

- 7 On this screen you select the responses to model. Select BTQ as the first response you want to model and click **Add**.

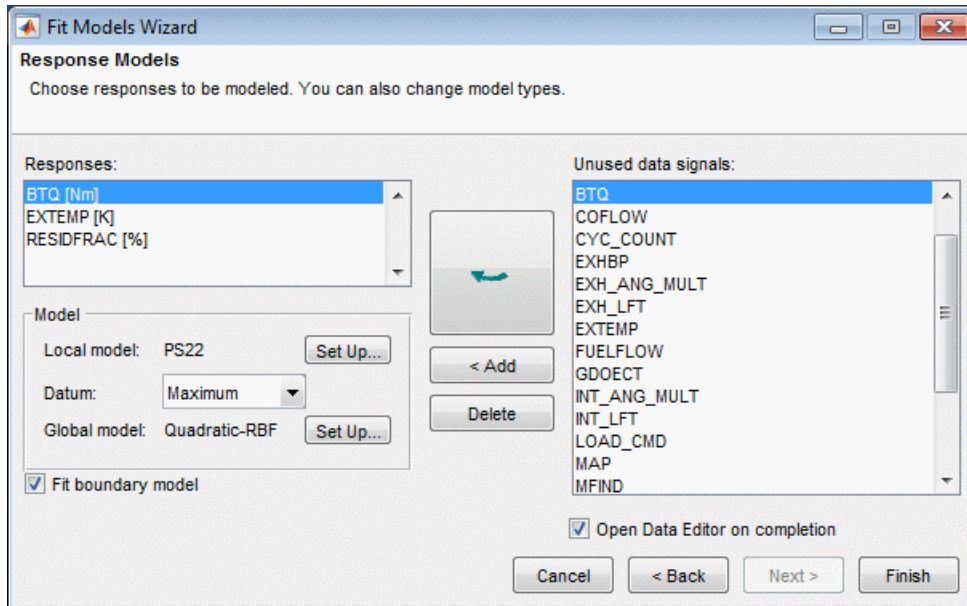


**8** Next to **Local model**, click **Set Up**.

The first response you want to model is torque against spark. The shape of torque/spark curves is well understood and you can examine them in the Data Editor in later steps. Polynomial spline curves are useful for fitting these shapes, where different curvature is required above and below the maximum. Therefore, you should set the local model type to polynomial spline. 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, at the maximum. The location of the knot in this case marks MBT.

- a** In the Local Model Setup dialog box, select **Polynomial Spline** from the **Local Model Class** list.
  - b** Set **Spline Order** to 2 below and 2 above knot.
  - c** Click **OK**.
- 9** In the Fit Models Wizard, select **Maximum** from the **Datum** drop-down menu. In this case, the maximum of the torque/spark curves is MBT (spark angle at maximum brake torque), so this can be a useful feature to model.

- 10 Add two more responses to model. Select EXTEMP and click **Add**, then select RESIDFRAC and click **Add**.



- 11 Click **Finish**.

The Test Groupings dialog box appears so you can select tests to model.

- 12 In the Test Groupings dialog box, edit the **Tolerance** for LOAD to 0.05. You should see 203 tests defined. Close the test Groupings dialog box.
- 13 Click **OK** to close the Define Test Groupings dialog box.

The Data Editor appears so you can inspect and filter data for modeling. The response models will be built when you close the Data Editor. For next steps, see “Viewing and Filtering Data” on page 2-19.



## Viewing and Filtering Data

| In this section...         |
|----------------------------|
| “View Data” on page 2-19   |
| “Filter Data” on page 2-23 |

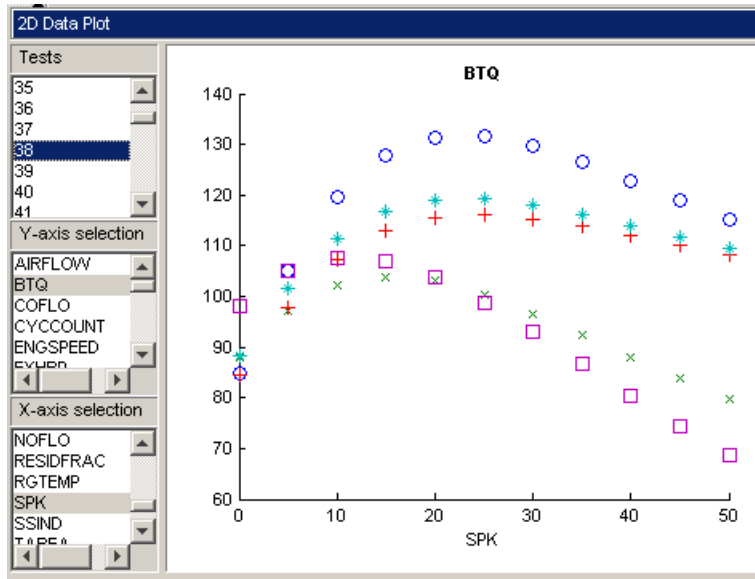
### View Data

- 1 Examine the data you imported by following the steps in “Selecting Data and Models to Fit” on page 2-14.

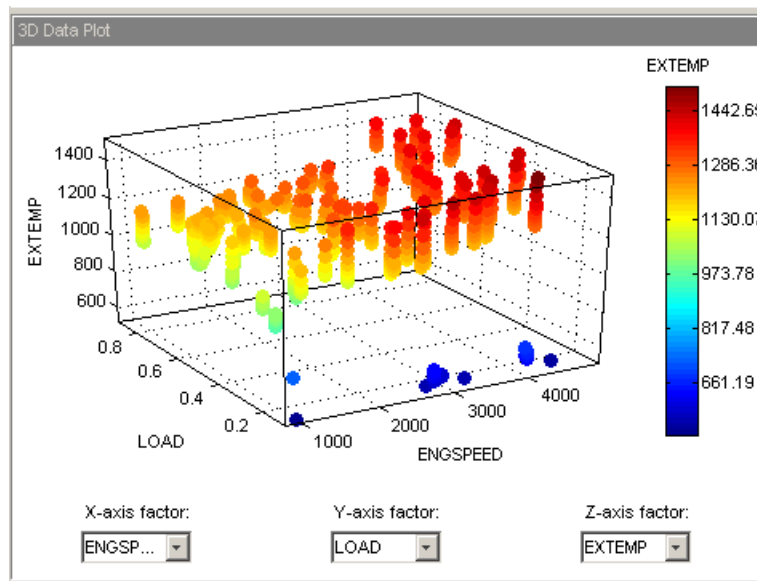
The Data Editor is a powerful tool for displaying and sorting your data. You can use the right-click context menu to split the views, or use the toolbar buttons or the **View** menu. You can choose 2-D plots, 3-D plots, multiple data plots, data tables, and list views of filters, variables, test filters, and test notes.

For example, if you do not already have a 2-D plot, right-click the title bar of any plot and select **Current View > 2-D Data Plot**.

- 2 Click in the left lists to plot torque (BTQ) against spark (SPARK), then select one or more tests to display.



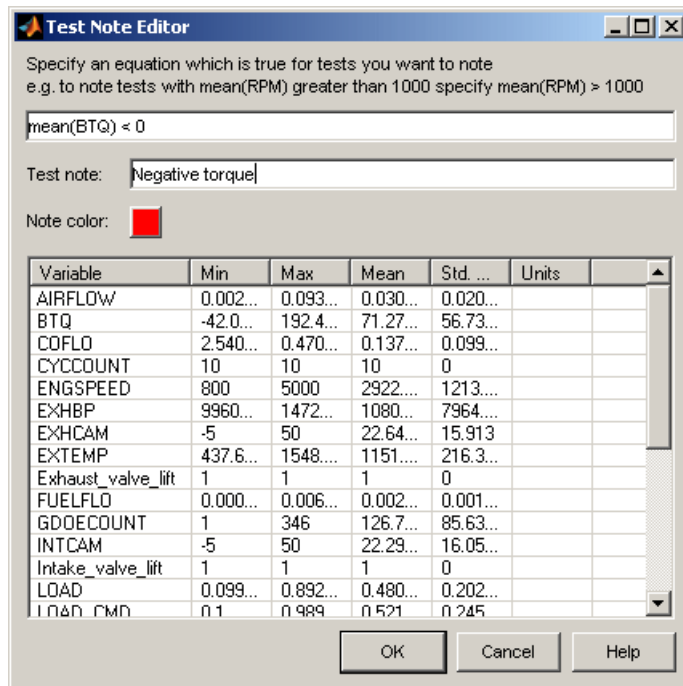
- 3 Right-click and select **Split View > 3-D Data Plot** to split the currently selected view and add a 3-D plot. Select one or more tests to display in the list at the left of the Data Editor, then choose three variables for the axes.



- 4 Right-click and select **Split View > Notes View** to split the currently selected view and add a test notes list view. This is empty until you add any test notes.
- 5 Select **Tools > Test Notes > Add**.

The Test Note editor appears.

- a Enter an expression that defines the tests you want to note; for example `mean(BTQ) < 0` will evaluate the mean torque for each test and note those tests where the value is less than zero.
- b Enter the text for this note in the edit box, e.g., **Negative Torque**.



- c Click **OK** to apply the test note.
- 6 Note that the new test note appears in the Notes list view.

| Notes |                 |
|-------|-----------------|
| Tests |                 |
| 1     | Negative torque |
| 12    | Negative torque |
| 15    | Negative torque |
| 16    | Negative torque |
| 30    | Negative torque |
| 78    | Negative torque |
| 83    | Negative torque |
| 94    | Negative torque |
| 98    | Negative torque |
| 108   | Negative torque |
| 110   | Negative torque |
| 149   | Negative torque |
| 184   | Negative torque |
| 192   | Negative torque |
| 2     |                 |

You can sort the column for the **Negative Torque** note by clicking the column header (once to sort ascending and once more to sort descending). This allows you to quickly identify which tests satisfy the note definition. Investigate these test points in the other views. Select a test in the notes view and that test is displayed in the table view, 3-D plot, and multiple data plots views (but not the 2-D plots, which have their own test selection controls). If you select multiple tests, they are all shown in the data plots, but only the first test in the list is highlighted in the table view.

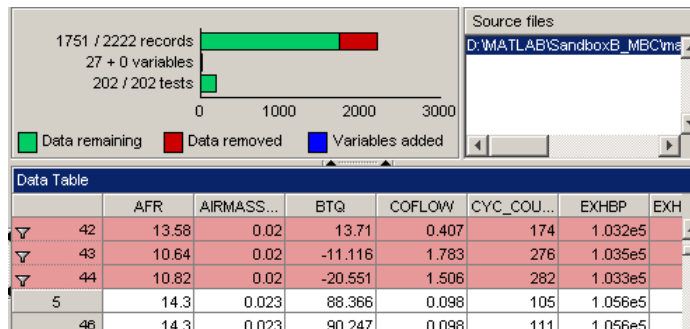
## Filter Data

- 1 If you decide certain operating points are unsuitable for modeling, for instance, unstable points on the edge of the engine's operating envelope, you can use the Data Editor tools to help you identify and remove them. You can remove suspect data in the following ways:
  - a You can remove individual points by selecting them as outliers in 2D and multiple data plots by clicking, then selecting **Tools > Filters > Remove Outliers**. You can always replace them again with the **Tools** menu.
  - b You can define filters to remove all data points that do not fulfil a certain expression. Select **Tools > Filters > Add**. The Filter Editor appears.

- c You can enter an expression that defines the records you want to keep. You can use any MATLAB function for filtering. Enter the following expression, which keeps records with AFR value greater than 14.25:

AFR>14.25

Click **OK**. Observe red records that have been filtered out in the top information bars. If you have turned on the option to **Allow Editing** in the data table view you can also see removed records in red.



- d Define another filter. Select **Tools > Filters > Add**. The Filter Editor appears.
- e Enter the following expression, which keeps records with RESIDFRAC value less than 35:
- RESIDFRAC<35
- f You can exclude whole tests by defining test filters to remove them. Select **Tools > Test Filter > Add**. The Test Filter Editor appears.

You want to keep only those tests with sufficient points to fit the model (at least 5 points).

Enter `length(BTQ)>4` and click **OK**.

- 2 You can add a Filter Definitions and Test Filter Definitions list view using the right-click menu to see whether the filters have been successfully applied and how many records or tests are removed by each filter.

| Filter List       |   |
|-------------------|---|
| Filter Expression | Results   |
| ✓ AFR > 14.25     | Filter successfully applied : 471 records excluded. |
| ✓ RESIDFRAC < 35  | Filter successfully applied : 131 records excluded. |

| Test Filter List       |  |
|------------------------|--|
| Test Filter Expression | Results  |
| ✓ length(BTQ) > 4      | Filter successfully applied : 2 tests excluded |

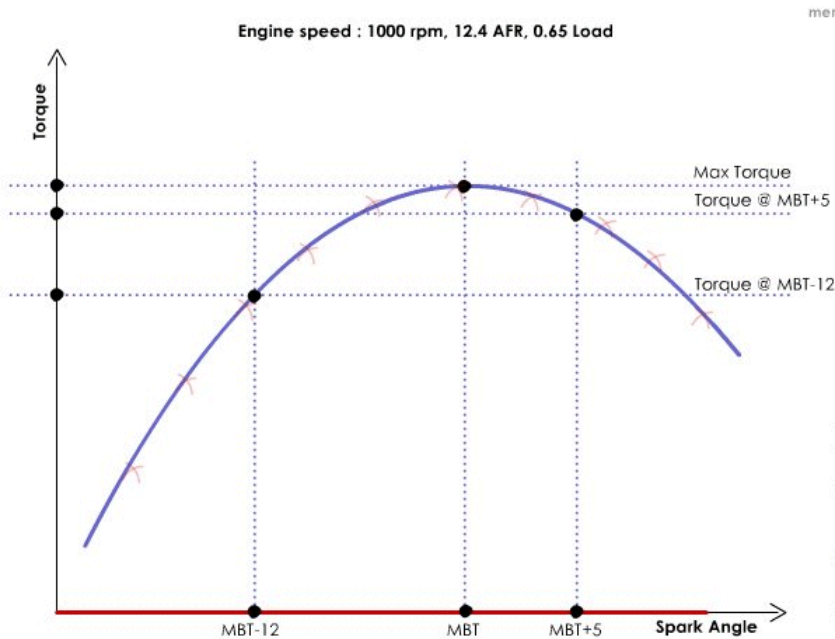
The bars at the top of the Data Editor always display the total numbers and proportion of removed data.

- 3 Close the Data Editor. A dialog appears to check that you want to build the response models and update the **Actual Design** to include all data selected for modeling. Click **Yes**, and the models are created.

For next steps, see “Selecting Local Models” on page 2-30.

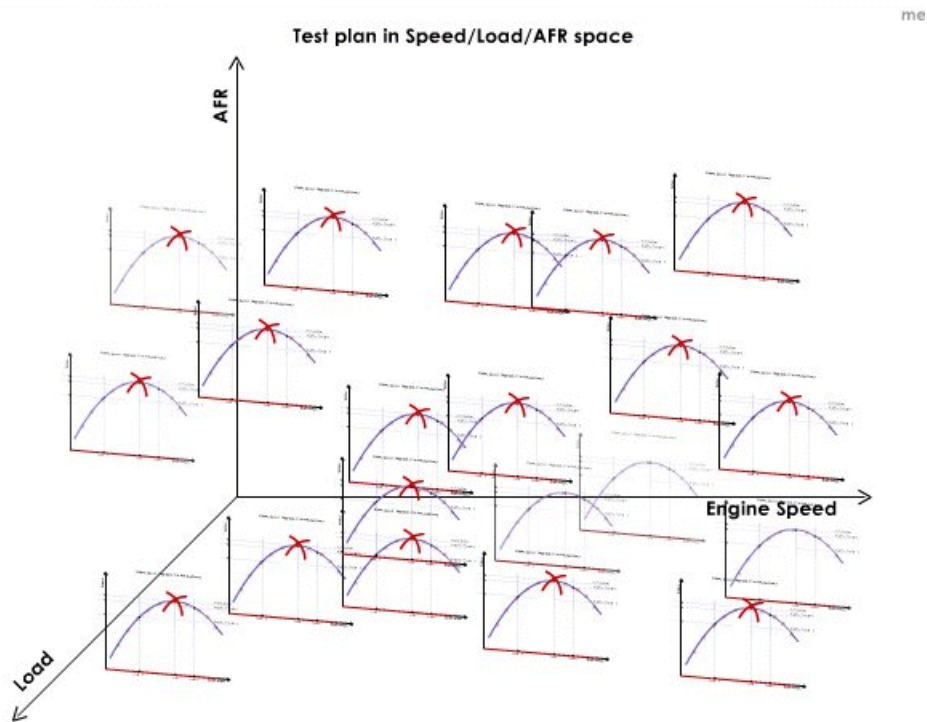
## How Is a Two-Stage Model Constructed?

Local models find the best fit of a curve to the data in each test. Each test in this case is a sweep of torque against spark angle, with speed, load, and cams held at a constant value for each sweep. The following illustrates a single sweep with a local model fitted.



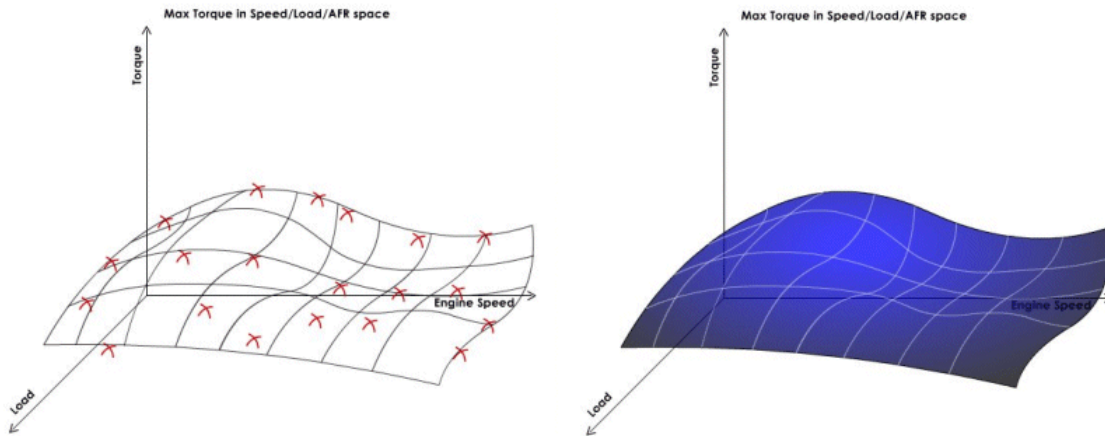
The local models provide the coefficients to generate global models. The equations describing those local model curves have certain coefficients such as max and knot, which for this data are peak torque and MBT spark (the spark angle that generates maximum brake torque).





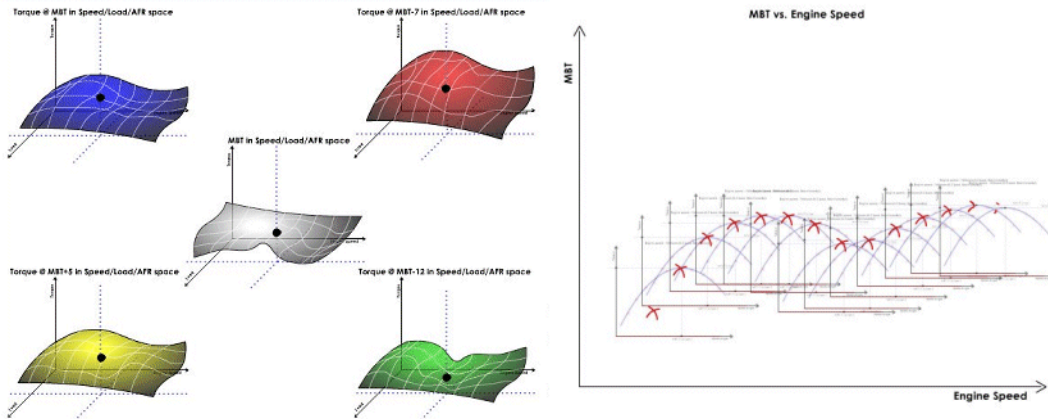
Local models are fitted to each test, in different places across the global space, as illustrated above. Each local model has coefficients for MBT and peak torque, etc. These coefficients become the data to which the global models are fitted. Coefficients such as peak torque and MBT are used to make the second stage of modeling more intuitive; an engineer will have a much better understanding of how a feature such as MBT spark varies through the global factor space than some esoteric curve fit parameter. Familiar variables like these are helpful to engineers trying to decide how well a model describes engine behavior. Better intuitive understanding allows much greater confidence in your models.

Global models are the best fit of a curve to the values of, for example, MBT for each test. This is repeated for each coefficient, producing several global models fitted to different coefficients of the local models. These coefficients are referred to as response features of the local models. The following example shows a global model for maximum torque across the speed/load global space.



The two-stage model is a surface fitted across all the global models, to describe the behavior across all global variables.

It can be useful to think of local and global models as a series of 2-D slices, while the two-stage model fits a 3-D surface across the curves of the global model slices. It is difficult to visualize more dimensions! The following example shows a variety of 3-D plots of global models for properties of the local torque/spark curves (such as MBT, peak torque, and torque a number of degrees before and after MBT), showing how these properties vary across the speed/load global space. The 2-D plot of the global MBT model (on the right) demonstrates how MBT varies with engine speed.



The two-stage model can take values of each coefficient at a certain value of, say, speed, to generate a new curve of torque against spark. This is a slice through the two-stage model surface.

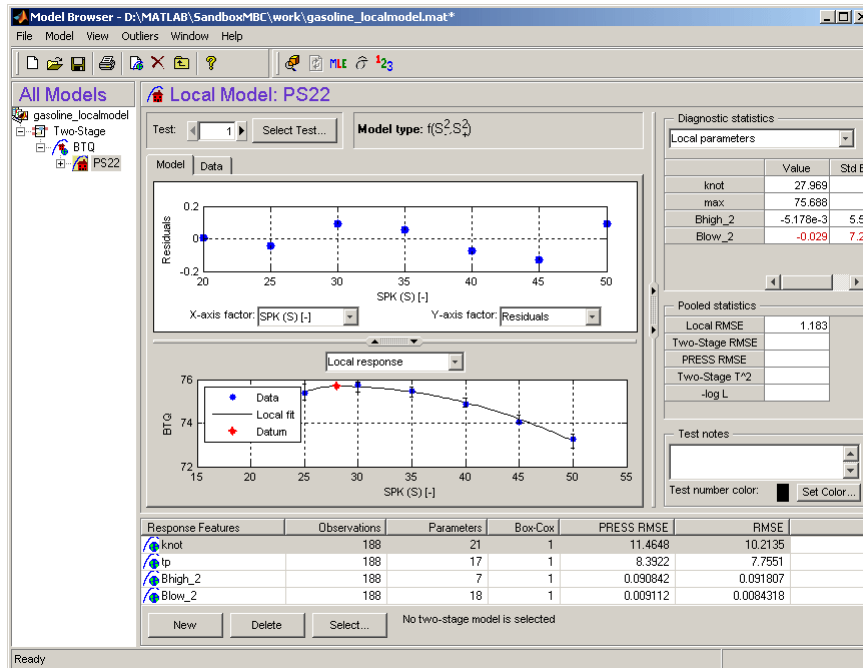
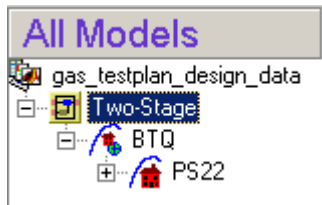
In other words, you can test your two-stage model by comparing it with the local fit and with the data. For example, you can reconstruct a local torque/spark curve at an operating point by taking the values of MBT and peak torque and the curvature from the two-stage model, and then validate this reconstructed curve against the original fit and the data. The two-stage model can also predict responses between tests, for new sweeps at intermediate values for which there is no data. If the two-stage model shows an accurate fit when compared to the local sweeps, this is a good sign that the engine behavior is well described by the model across the global variables.

For more details on two-stage modeling, see “About Two-Stage Models” on page 8-3, and see “Two-Stage Models for Engines” in the Model Browser documentation for more statistical depth. (In the Help Browser you can right-click and select **Back** to return to previous pages).

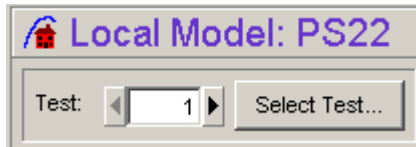
## Selecting Local Models

You should inspect the local and global models in turn, removing outliers if appropriate and trying different model types, before creating a two stage model. If the fit is good at the local and global levels you have the best chance of creating a two-stage model that accurately predicts the engine behavior. First, inspect the models.

- 1 Select the local model node PS22 in the model tree, in the **All Models** pane, and the local model view appears on the right.



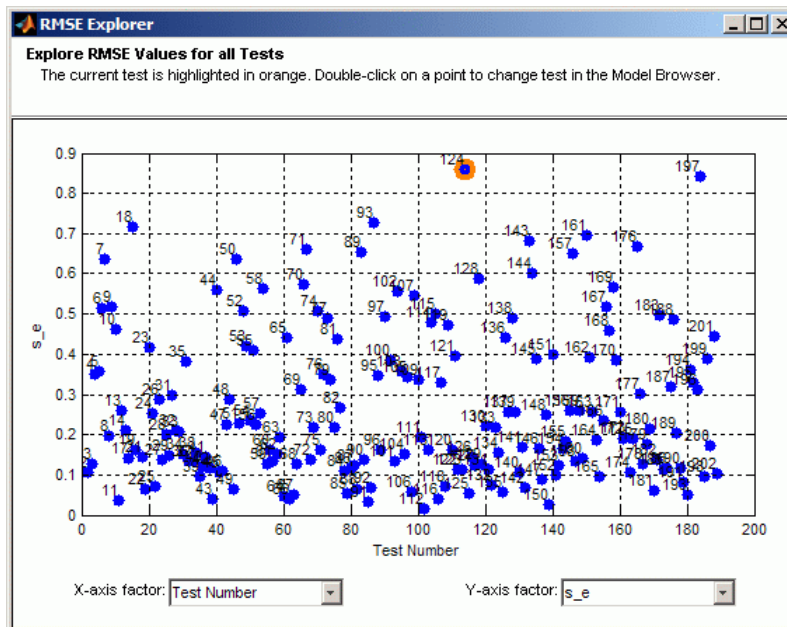
- 2 Look through the tests to inspect the fits. Use the **Test** controls.



3

To quickly identify problem tests, click RMSE Plots  in the toolbar (or **View** menu).

Inspect tests with high error values in the local model view. You can navigate to a test of interest from the RMSE Explorer by double-clicking a point in the plot to select the test in the Model Browser local model view. Plot all response features in the RMSE Explorer and investigate tests with extreme values.



4

Consider removing outliers to improve fits if some points are badly distorting the torque spark curve (use right-click or **Outliers** menu). Pay particular attention to end points. For example, for tests where the majority of points are at higher spark angles than the maximum (at MBT), it can improve the fit to remove some of these long “tails”. It can be useful to remove outliers in this region, because there is likely

to be knock at spark values much higher than MBT where the engine is less stable. Similarly, as there is no knock in simulation data, points can be collected far in advance of MBT, and it can improve the fit to remove these.

- 5 If removing some outliers does not bring MBT within the range of the data points, consider removing the whole test (use the **Outliers** menu).
- 6 After making changes, check the RMSE Explorer plots again for problem tests.
- 7 Check all tests before inspecting the global models. Try looking at the **Local diagnostics** (select from the list in the **Diagnostic Statistics** pane). Check for high values of **Cond(J)** (e.g.,  $> 10^8$ ). High values of this condition indicator can be a sign of numerical instability.

Optionally, see “Viewing the Boundary Model” on page 2-33, to inspect the boundary model.

For next steps, see “Selecting Global and Two-Stage Models” on page 2-38.

## Viewing the Boundary Model

---

**Note:** The Fit Models Wizard automatically created a boundary mode for you.

---

You can view and create boundary models at the test plan node. A model describing the limits of the operating envelope can be useful when you are evaluating optimization results and global models. The boundary model is useful for viewing global models, so you can see the model areas inside the boundary on plots.


To view the boundary model setup:

- 1 Select the test plan node in the model tree.
- 2 Select **TestPlan > Fit Boundary**.

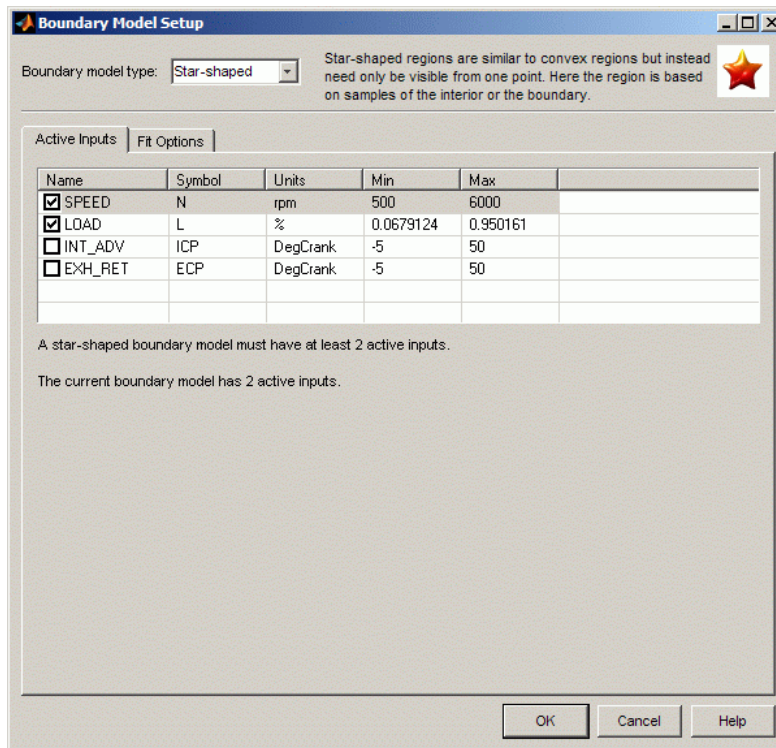
The Boundary Editor opens.

- 3 View the boundary models.

(Optional) The following steps show you how to build two boundary models and combine them, to map the envelope in speed and load, and also in all four inputs.

- 1 (Optional) Select the root Two-Stage node and click New Boundary Model  in the toolbar. A dialog opens where you can choose to build a boundary model of the response, local, or global inputs. In this case, you are interested only in making a model of the global boundary. You are not interested in the **Local** boundary (you know the range of spark already). Select **Global** and click **OK**.

A dialog opens where you can select constraint inputs.

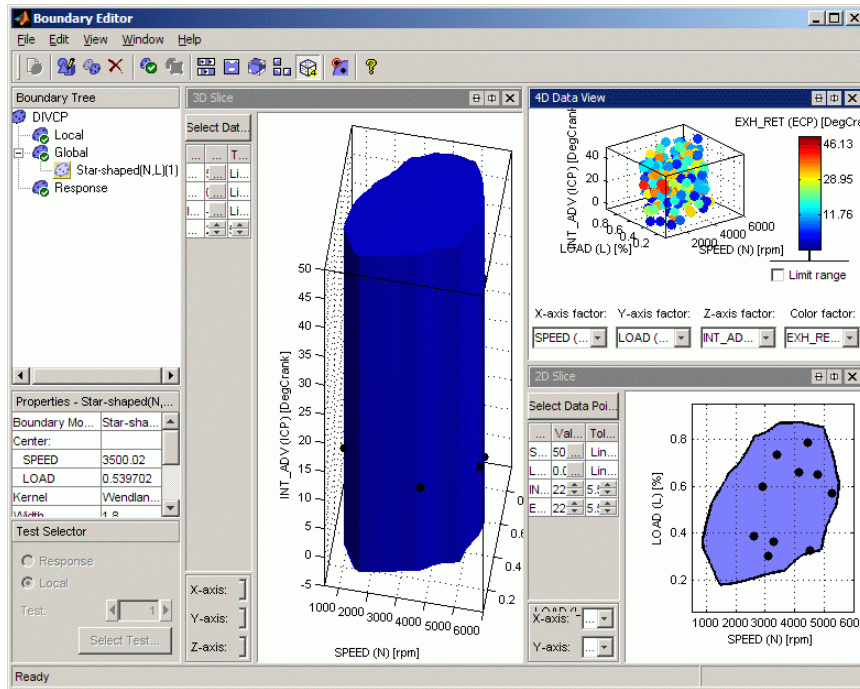


- 2 Leave the **Constraint type** set to **Star shaped** and leave only the **SPEED** and **LOAD** check boxes selected. Click **OK** and the boundary constraint is calculated.

A **Star shaped(N, L)** child node appears under the **Global** node and the view switches to the new model.

- 3 Select **View > Current View > 3D Slice** (or use the toolbar button) to examine the shape of the new boundary model. Drag the axes to rotate the plot. Use the drop-down menus to change the variables plotted. In any view, you can change the position of the plotted slice by altering the variable values in the edit boxes. You can split the views as in the Design and Data Editors using the buttons in the title bars, or the right-click or **View** menus.





4



Click Show Boundary Points in the toolbar. Boundary points are outlined in red. In some projections and at some plot resolutions, points can falsely appear to be outside the boundary. You can alter the number of points plotted in the Value edit boxes, but high resolutions can be time-consuming to plot. You can also check particular points: in the 3D, 2D, and 1D Slice views you can click (and hold) points to see the values of the global variables at that point and the Distance. A distance of 0 means the point is exactly on the boundary, and negative values show the distance inside the boundary. You can use this function to check that enough points are inside or close to the boundary.

- 5 In a 2D or 3D Slice view, click the button **Select Data Point**. A dialog appears where you can select a data point. When you click **OK** the slice is plotted at the location of the selected data point. You can also double-click points to move the slice to that location.


- 6 Select **View > Current View > Pairwise Projections** (or use the toolbar button). Here you can see pairwise projections to view the boundary across all combinations of factors.

---

**Note:** If some points appear to be outside the boundary, select **View > Set Resolution** to increase the resolution of the pairwise plots. Increasing the number of evaluation points displays more detail, but takes longer to calculate.

---

- 7 On one of the pairwise plots, click and drag to select a small area of points. The area you select is colored yellow across all plots, so you can view how those points are distributed across factors. With some detailed surfaces, areas in the pairwise plots can appear as discrete patches, so this feature is useful for tracking regions across factors. This can help you decide whether the boundary is capturing enough (or too much) detail of the surface.

- 8 Select the Global node, and click New Boundary Model  in the toolbar. You can build other boundary models to compare and combine for the most useful model.

From the Global node you will automatically get a global child node, so you do not need to select a level. The Boundary Model Setup dialog box appears.

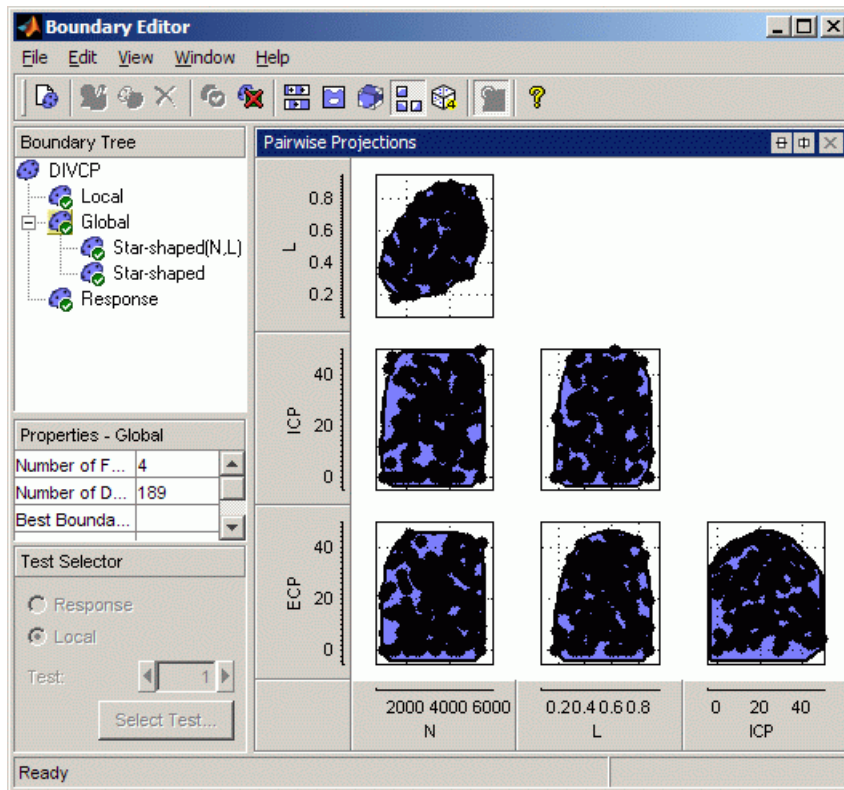
- 9 This time leave all four inputs selected and click **OK**. A new global boundary constraint is calculated.
- 10 Compare the two global boundary models by selecting the Pairwise view, and selecting each model in the tree in turn.
- 11 If you want to combine boundary models, you select **Edit > Add to Best** and select the parent node to see the combination.

---

**Note:** For this example, do not add any models to best.

---

For more information see “Combining Best Boundary Models”.



- 12** Close the Boundary Editor to return to the Model Browser. Once calculated, boundary models remain part of the test plan unless you delete them.

For next steps, see “Selecting Global and Two-Stage Models” on page 2-38

## Selecting Global and Two-Stage Models

### In this section...

“Inspect the Global Models” on page 2-38

“Create Multiple Models to Compare” on page 2-40

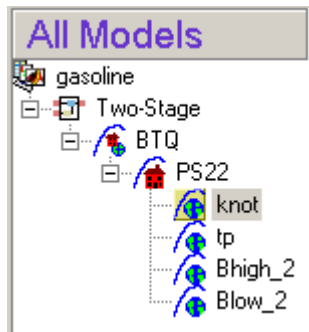
“Create a Two-Stage Model” on page 2-44

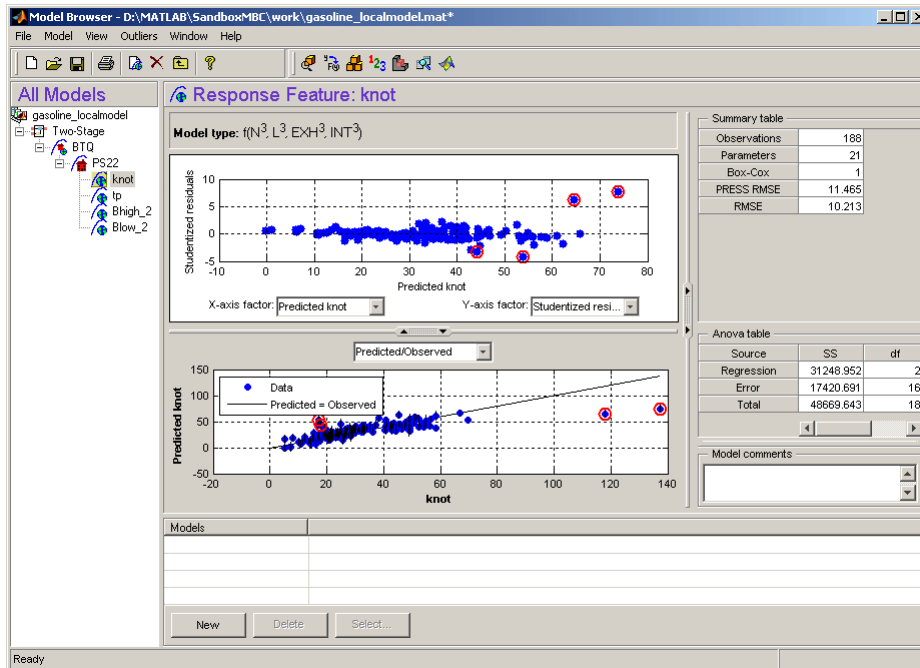
“Evaluate Other Response Models” on page 2-46

### Inspect the Global Models

When you are satisfied with the local fits, inspect the global models in turn. You should check trends of global models. Do the trends go in the right direction? Previous engineering knowledge can be applied. The following steps suggest useful plots for investigating trends.

- 1 Expand the local model node (PS22) in the model tree and click **knot**.





- 2 Right-click outliers (or any point) to see a plot of the test. You can inspect the shape of the torque/spark curve and see the values of the global variables. This can help you identify problem tests, perhaps on the edge of the stable operating region.

Also you can use the global model view scatter plots to plot predicted values against variables.

---

**Note:** After identifying problem tests at the global level, return to the local level to decide whether to remove outliers or the whole test.

---

- 3 Select **Model > Evaluate > Fit Data** to open the Model Evaluation window.
  - Select **View > Cross Section** to see the trends of your current model e.g., maximum torque (max model), or MBT (knot model). Try the **Response Surface** view. Apply engineering knowledge to check trends.
  - Be aware that Bhigh2 and Blow2 must be negative (to ensure maximum torque occurs at MBT). To check for this, select **View > Response Surface** and click the

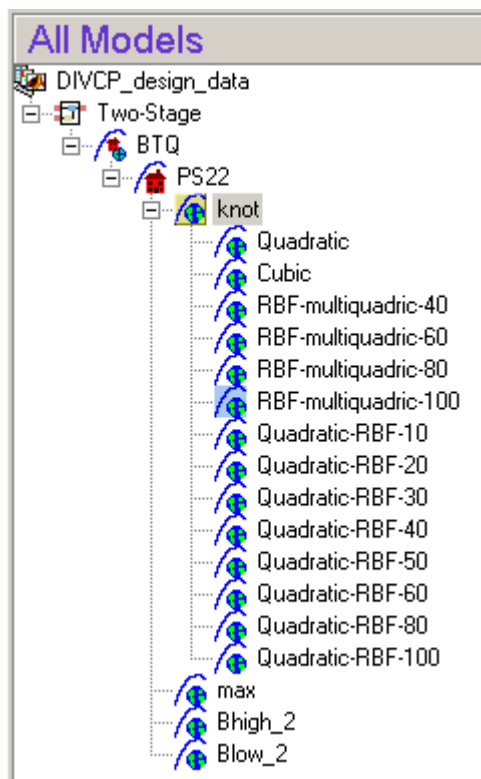
**Table Display type** — here cells outside the boundary model are highlighted yellow to help you focus on the area of interest.

- 4 Also you can use the global model view scatter plots to plot Predicted Bhigh2 or Predicted Blow2 against Speed to look for suspicious positive values.

### Create Multiple Models to Compare

- 1 Click the **Build Models** toolbar button.
- 2 Click **Browse**, locate the `matlab\toolbox\mbc\mbctraining` folder and click **OK**.
- 3 Select the DIVCP template and click **OK**.

Choose **PRESS RMSE** in the Model Selection dialog box and click **OK**, and a selection of child model types are built for `knot`. **PRESS RMSE** is the criterion used for automatically selecting the best model out of the child nodes.



You can save your own template of any collection of child models. From the parent model node (e.g., **knot**) select **Model > Make Template**.

- 4 Now **knot** has several child models. Inspect each model in turn. If you remove outliers that have an RBF center (marked with a star) be sure to refit the model (click the toolbar button Update Model Fit) to reselect widths and centers.
- 5 Return to the **knot** model node and look at the statistics reported in the list of child models at the bottom. From any parent model node you can see a list of statistical comparisons for all the child nodes in the lower list pane, along with information such as the number of parameters. Use this information to help you decide which model is best.

| Models               | Observations | Parameters | Box-Cox | PRESS RMSE | RMSE    |
|----------------------|--------------|------------|---------|------------|---------|
| Quadratic            | 187          | 10         | 1       | 7.337      | 6.9032  |
| Cubic                | 187          | 20         | 1       | 6.4821     | 5.9023  |
| rbf-multiquadric-40  | 187          | 32         | 1       | 3.4733     | 2.8414  |
| rbf-multiquadric-60  | 187          | 53         | 1       | 2.6373     | 2.0826  |
| rbf-multiquadric-80  | 187          | 68         | 1       | 2.3659     | 1.6809  |
| rbf-multiquadric-100 | 187          | 84         | 1       | 1.7372     | 1.1861  |
| Quadratic-RBF-10     | 187          | 19         | 1       | 3.6955     | 3.4357  |
| Quadratic-RBF-20     | 187          | 32         | 1       | 3.2752     | 2.772   |
| Quadratic-RBF-30     | 187          | 37         | 1       | 2.7291     | 2.4864  |
| Quadratic-RBF-40     | 187          | 44         | 1       | 2.8808     | 2.3218  |
| Quadratic-RBF-50     | 187          | 56         | 1       | 2.5455     | 1.8265  |
| Quadratic-RBF-60     | 187          | 65         | 1       | 2.3578     | 1.6425  |
| Quadratic-RBF-80     | 187          | 82         | 1       | 2.5026     | 1.48    |
| Quadratic-RBF-100    | 187          | 101        | 1       | 1.969      | 0.98542 |

New Delete Select... 'DIVCP\_design\_data/Two-Stage/BTQ/PS22/knot/rbf-multiquadric-100' is the best model for knot

- 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.

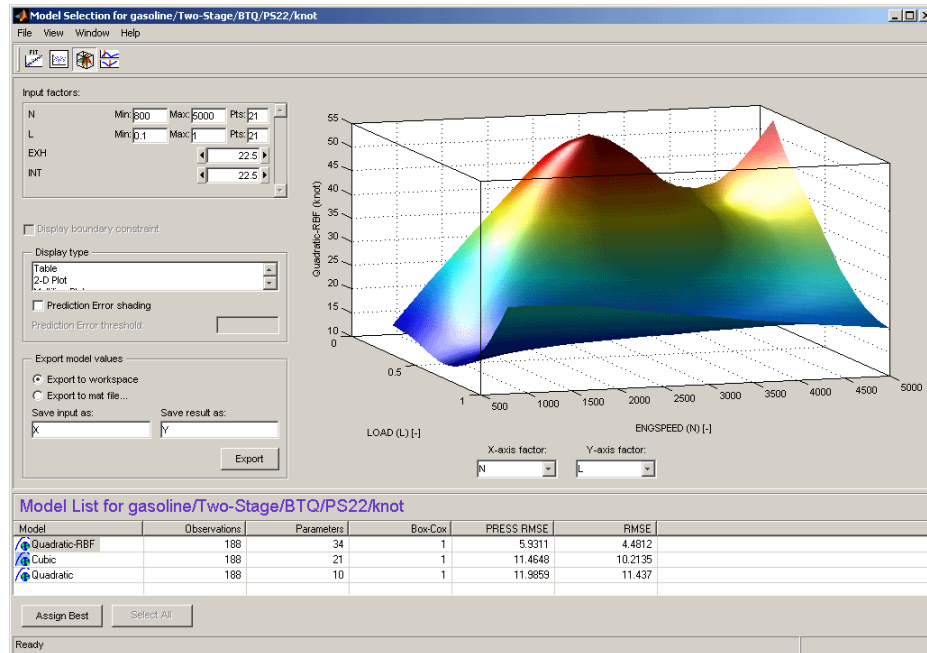
If the value of PRESS RMSE is much bigger than the RMSE then you are overfitting — the model is unnecessarily complex.

PRESS RMSE can be the most helpful single statistic you can use to search for the best fit relative to the number of terms in the model. However you should not rely on any single statistic, but use a variety of criteria and especially the graphical tools available for comparison of models in the Model Evaluation tool when you click Select. You can also use other diagnostic statistics to help you select models. For detailed guidance on how to understand the selection tools see “Create Multiple Models to Compare” on page 8-29 and the “Model Selection




Guide” in the Model Browser documentation. (In the Help Browser you can right-click and select **Back** to return to previous pages.)

- You can add other statistics here. Select **Model > Summary Statistics**. In the dialog, select the check box for **AICc** and click **OK**. A column of **AICc** values appears in the model list. This can be a useful statistic for comparison of models as it penalizes over-parameterized models. Over-fitting models can later cause problems with optimization when you use the models to create calibrations. See “Using Information Criteria to Compare Models”.
- 6 From the **knot** model node, click **Select** in the **Models** pane at the bottom to open the Model Selection window. Try the different views and compare the child models by selecting them in the **Model List** at the bottom.

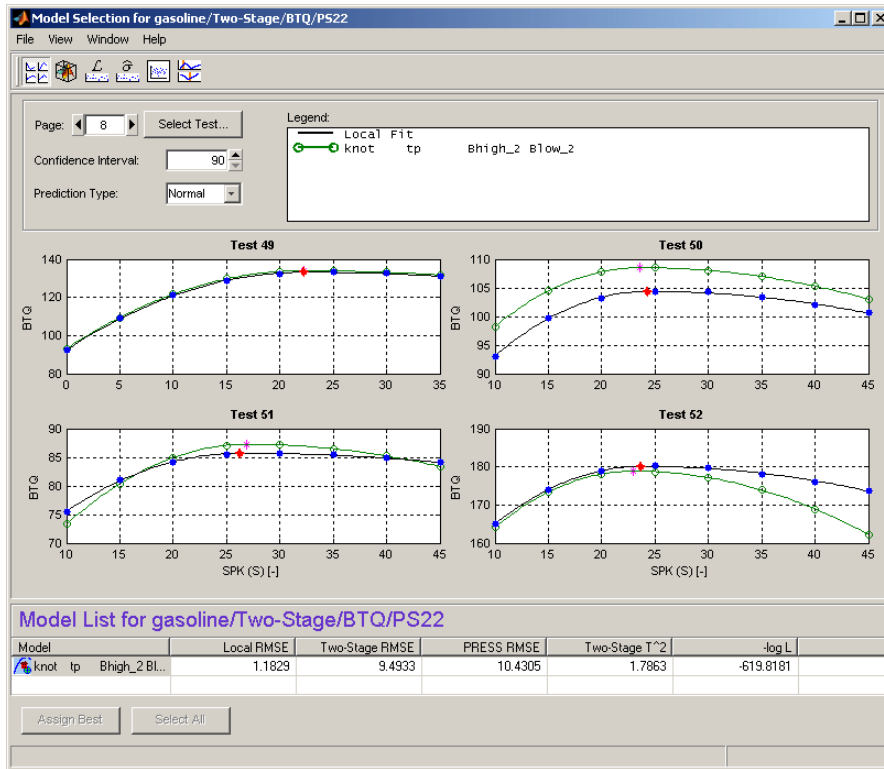


- 7 Select one of the child models as best (click the button **Assign Best**) and close the Model Selection window to return to the Model Browser. Try other model types if you are not satisfied with the quality of the fit. You could work through the modeling tutorial for more guided examples of how to select models, see “Empirical Engine Modelling” on page 8-2.

- 8 When you have selected a best child model, select **File > Clean-up Tree** to discard all but the model you chose as best. The process of searching for good model fits often results in a large selection of models, and cleaning up the tree reduces file size. It can also save time when saving the file, especially if you have made a change such as removing an outlier at local level, that causes all response feature models to be refitted.
- 9 Select another global model such as `Bhigh_2` and click Build Models  in the toolbar. Select the DIVCP template to automatically build the same selection of child model types for `Bhigh_2`. This can help you quickly build multiple models to compare. The Model Selection dialog appears, where you can choose a criterion such as PRESS RMSE or AICc for automatically selecting the best model out of the child nodes.
- 10 Repeat this process of searching for good global model fits for the other three global models.

### Create a Two-Stage Model

- 1 When you have selected best models for each global model, return to the local model node (PS22) in the model tree, and click **Select** in the model list pane at the bottom to calculate the two-stage model and open the Model Selection window.



Look through the plots of the new two-stage model against the local fits and the data.

When you close the Model Selection window and accept the new model as best, the two-stage model is copied to the BTQ response node in the model tree.

- 2 You can choose to calculate maximum likelihood estimation (MLE) at this point. This process refits, taking proper account of the correlation between different response features. Try calculating MLE, then select the BTQ node and return to the Model Selection window to compare the MLE model with the univariate model (click **Select All** in the Model List pane).

For more guidance on creating multiple local, global and two-stage models to search for the best fit, work through the step-by-step examples in “Empirical Engine Modelling”

on page 8-2, especially the section “Create Multiple Models to Compare” on page 8-29.

### Evaluate Other Response Models

To complete the model building you should follow the same process as for modeling torque to select good models of the other response models you created using the Fit Model Wizard: exhaust temperature and residual fraction. These models are required for the optimization problems. Remember you can look at the example finished project, `Gasoline_project.mat`, in the `matlab\toolbox\mbc\mbctraining` folder, to examine the example models. You will use these example models in CAGE for the optimization section.

#### Exhaust Temperature Model

- 1 Expand the EXTEMP model node in the model tree and select the local node POLY2.
- 2 You can copy the outliers you selected for the torque models. Select **Outliers > Copy Outliers From**. The Copy Outliers dialog box appears. Select the BTQ local model node in the tree (PS22, under BTQ), and click **OK** to copy the outlier selections to the EXTEMP local node.
- 3 Examine the fit in the same way as you did the torque fits.
  - Try different local models. Click **New** at the EXTEMP response node.
  - Try different global models as you did for the torque response features. Use **Build Models** and the DIVCP model template, or click **New** to add child nodes and try different model types one at a time. If you use **Build Models** at the local level you can apply a template to all response feature models at once.
  - Try a new response feature. Click **New** at one of the local nodes you have created and enter -10 for the value to use MBT minus 10 degrees of spark as a new response feature.

See “Empirical Engine Modelling” on page 8-2 for simple worked examples of adding new local, global, response feature and two-stage models.

- 4 When you are satisfied with the fits, return to the local model node and click **Select** to calculate the two-stage model. If you have added new response features, there will be more than one two-stage model to choose from in the Model Selection window.
- 5 Try calculating MLE and return to the Model Selection window to compare the MLE model with the univariate model and select the best.

### Residual Fraction Model

- 1 Repeat the steps to evaluate the model of residual fraction (RESIDFRAC).

You selected the MBT datum model in the Fit Models Wizard, so by default it also created the BTQ datum datum link model for the other response models. This means you can view MBT, and one of the response features, FX\_0, models residual fraction at MBT. You will use this in your optimizations in CAGE.

- 2 The process of searching for good model fits often results in a large selection of models. Remember to discard all but the models you chose as best, by selecting **File > Clean-up Tree**.

Look at the example finished project, `Gasoline_project.mat`, in the `matlab\toolbox\mbc\mbctraining` folder, to examine the example models. You will use these example models in CAGE for the optimization section.

### Using Validation Data

After you have created models, you can use the validation data.

To import and filter the validation data set,

- 1 Select the top project node in the Model Tree.
- 2 Double click the **Two-Stage: Data** Object in the Data Sets pane. The Data Editor appears.
- 3 Click the **Storage** button in the toolbar.

Click **Store current filters** and **Store current test filters** in the toolbar. This allows you to reuse these filters in another data object without having to recreate them.

- 4 In the Model Browser, select **Data > Import Data from File** (or use the toolbar button).
- 5 Use the Browse button to find and select the `DIVCP_Validation_DoE_Data.xls` data file, in the `matlab\toolbox\mbc\mbctraining` folder. Click **Open**.

The Data Editor opens.

- 6 You need to define test groupings as before.
  - a Select **Tools > Change Test Groupings** (or use the toolbar button)
  - b In the Define Test Groupings dialog box, clear the check box **One test/record**.
  - c Locate and double-click **LOAD** in the **Variables** list box.
  - d Edit the **Tolerance** to 0.05.

22 tests are defined.

- e Close the Define Test Groupings dialog box.
- 7 To apply the same filters as before, click the **Storage** button in the toolbar.

In the Storage dialog box, select the filter and test filter objects in turn and click **Append stored object** in the toolbar. This applies these filters to the current data object. Return to the Data Editor window and observe some data is removed by the filters in the summary at the top. You can check the filters in the Filter List and Test Filter List View.

- 8 Close the Data Editor and click **Yes** to accept data changes.

To attach the data set to your test plan for validation:

- 1 At the test plan level, select **TestPlan > Validation Data**. The Select Data for Validation wizard appears.
- 2 Select the validation data set (observe the smaller number of tests): **Data Object**, and click **Next**.
- 3 By default all tests are selected on the next screen, so click **Finish** to use all the tests to validate models in this test plan.

The validation data set appears in the information pane for the test plan. Validation RMSE is automatically added to the summary statistics for comparison in the bottom list view of response models in the test plan.

You can now use the validation data to validate all models except response features. You can see validation statistics in the following places:

- Model List — **Validation RMSE** appears in the summary statistics in the lower list of models at the test plan, response and one-stage nodes
- At the local node view:
  - Pooled Statistics — **Validation RMSE** — The root mean squared error between the two-stage model and the validation data for all tests
  - Diagnostic Statistics > Local Diagnostics — Local model **Validation RMSE** for the currently selected test (if validation data is available for the current test—global variables must match)
- Summary Table — **Validation RMSE** for one-stage models

View validation plots in the following places:

- Plots of **Validation residuals** — For local models
- From any model node except response features, you can select **Model > Evaluate > Validation Data** to open the Model Evaluation window and investigate the model with the selected validation data.

Check the model trends in the cross section view. Check the fit against the validation data. Is the fit acceptable? Do you need more data? Can you improve the fit with the existing data? Pay attention to the boundary model on the plots. Yellow areas are outside the boundary. Focus on the model trends only within the boundary model.

### Exporting the Models

You can export models to file, to CAGE, to generate calibrations.

- 1 To export all models in the test plan, select the test plan node.
- 2 In the **Common Tasks** pane, click **Generate calibration**.
- 3 In the Export to CAGE dialog box, click to **OK** to accept the default model names.

The CAGE Browser displays the models.

Alternatively you can use **File > Export Models** to choose to export to a file, to the workspace, or to Simulink. This capability makes models easy to share across engineering groups. The models exported depend on the model node you have selected in the tree.

For this example, you will use the models in the CAGE part of the toolbox to produce optimized calibration tables. For next steps, see “Optimized Calibration” on page 2-51.



# Optimized Calibration

**In this section...**

“Problem Definition” on page 2-51

“Benefits of Automated Calibration” on page 2-52

## Problem Definition

This section describes the creation and optimization of calibration tables for the gasoline case study. The Model Browser section of this case study covers creating the design for the experiment and creating and evaluating models from the resulting data. You can export your models directly to CAGE; or to Simulink software or to a file, ready to be imported into CAGE for model-based calibration generation. An example file is provided.

The aim of this case study is to produce optimized tables for

- Intake cam phase
- Exhaust cam phase
- Spark timing schedules

as a function of load and rpm, subject to the following constraints

- Constrain solutions to lie within the boundary constraint model
- Constrain cam phase solutions so they do not change by more than  $10^\circ$  between table cells (that is, no more than  $10^\circ$  per 500 RPM change and per 0.1 load change).
- Constrain residual fraction  $\leq 25\%$  at each drive cycle point (to ensure stable combustion). Residual fraction is the percentage of burned gas mass in the cylinder at intake valve close, relative to the total mass in the cylinder at intake valve close. Constraining maximum residual fraction is a simple and reasonable way of ensuring stable combustion. Residual fraction =  $100 * \text{Burned Gas Mass from Last Cycle} / (\text{Burned Gas Mass From Last Cycle} + \text{Fresh Air Mass})$

CAGE is intended for model-based calibration, although you can still create tables without reference to models if you want. For this case study, you use models produced in the Model Browser to generate calibrations in CAGE. You cover the following steps:

- 1 Load models of engine responses, decide on optimization strategy and define additional models. See “Importing Additional Models into CAGE” on page 2-54.

- 2 Set up tables. See “Setting Up Calibration Tables to Fill” on page 2-57.
- 3 Define optimization objective and constraints. See “Setting Up the Optimization” on page 2-60.
- 4 Set up an operating point set for the optimization. See “Defining Variable Values” on page 2-63.
- 5 Run the optimization and view the results. See “Running the Optimization” on page 2-65
- 6 Duplicate and modify the optimization to create a sum optimization using the previous results as starting points. See “Setting Up the Sum Optimization” on page 2-70
- 7 Fill tables from optimization results. See “Filling Tables with Optimization Results” on page 2-75.
- 8 Use models and optimized tables to fill a spark estimator table. See “MBT Spark Estimator Problem” on page 2-76.

For guidance, you can look at the example finished project:  
`Gasoline_optimization.cag`.

### **Benefits of Automated Calibration**

- You can move the table-filling process away from the test bed.
- You can regenerate calibrations when objectives, constraints, or calibration table layouts change, without additional testing.
- You can explore tradeoff possibilities interactively.
- You can produce initial calibrations using engine simulation software, before hardware is available.

CAGE can provide both automatic and interactive calibration optimization. You can trade off multiple objectives, deal with multiple constraints, and you can examine optimizations point-by-point or drive-cycle-based. You can use built-in optimization routines or write your own. You can fill groups of tables simultaneously, and optimize table values and breakpoint settings. CAGE can provide solutions for these example applications:

- Control problems
  - Injection timing and duration
  - EGR valve

- Spark timing
- Dual-independent variable valve timing
- Emissions-constrained BSFC optimization over drive cycles
- Estimation problems
  - Torque
  - Emissions
  - Air flow and manifold pressure
  - Intake valve temperature
  - Borderline spark

### Importing Additional Models into CAGE

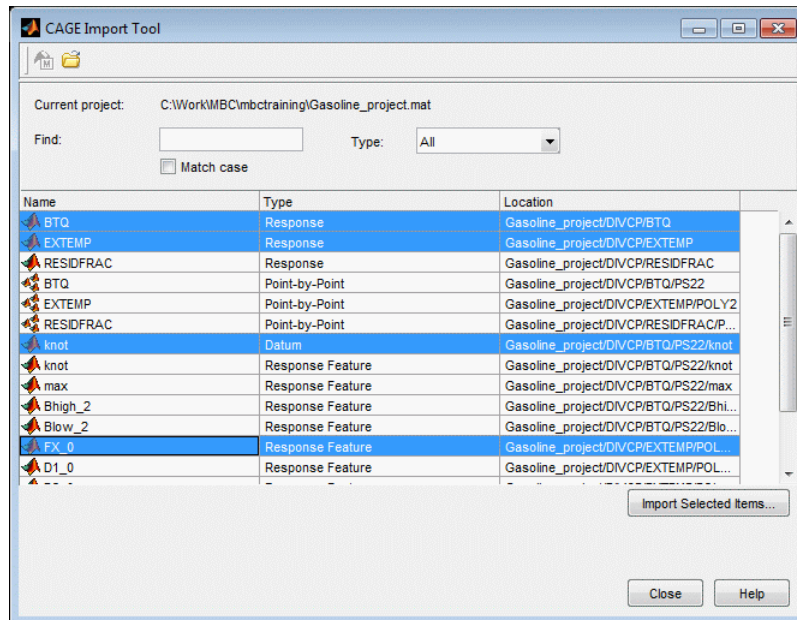
- 1 In the CAGE Browser select **File > Import From Project**. The CAGE Import Tool appears.
- 2 Click the **Import From Project File** button. Locate the example model file created in the Model Browser, `Gasoline_project.mat`, in the `matlab\toolbox\mbc\mbctraining` folder, and click **Open**.
- 3 You can select from a list of models in the file. Select these models by **Ctrl**+clicking in the list:
  - BTQ
  - EXTEMP
  - knot (**Type** = Datum)

In this case the `knot` model is duplicated because the datum model was used twice during modeling. The datum model tracked the maximum of the torque model, that is, MBT (the spark angle at maximum brake torque). This datum model was also used when modeling exhaust temperature and residual fraction, because it can be useful to see MBT on model plots for other factors. This is called a datum link model. You need only one copy of the `knot` model.

- `FX_0` (Check that the **Location** specifies `RESIDFRAC`, *not* `EXTEMP`)

The response feature model `FX_0` is `RESIDFRAC` at MBT. `RESIDFRAC` was constructed using a datum link (in this case `knot`, i.e., MBT) so `FX_0` is residual fraction evaluated at the datum point (MBT).

Check that your selection matches those shown.



- 4 Click the **Import Selected Items** button .
- 5 In the following Import dialog,
  - Double-click to edit the **CAGE Model Name** for the knot model to rename to MBT.
  - Double-click to edit the **CAGE Model Name** for the FX\_0 model to rename to RESIDFRACatMBT.
  - Leave the other settings to replace models already in CAGE with the models from the file.
  - Click **OK** to import the models.
- 6 Close the Import Tool and look at the Models view in CAGE. You should see the BTQ, EXTEMP, MBT, and RESIDFRACatMBT models in the list. The selected model is displayed in the other panes.

The objective is to produce optimized tables for spark and cam timings, subject to constraints on operating region, residual fraction, and the gradient of the cam phasers over the calibrated tables. You want the optimization to search for the timings that give the best torque and minimum fuel consumption, subject to the constraints.

You could fix the spark timing to be MBT spark (the spark angle that produces maximum brake torque) by using the MBT spark model as the spark input to the other models in the optimization. With spark fixed at MBT, you can set up an optimization that allows the two valve timings to vary, exploring the space via a gradient descent algorithm to locate the optimal timings. Remember, this is a simplified problem and running an engine at MBT is knock-limited and so is not possible at all operating points. For this example, you will not use the MBT model as you will use spark as a free variable in the optimization. You will return to the MBT model later.

Note you can also:

- Export models directly from the Model Browser to CAGE when both are open.
- Use the CAGE Import Tool to import models directly from the Model Browser.
- Use the CAGE Import Tool to import models and other calibration items (tables, optimizations, tradeoffs, data sets, features) from any project file created in CAGE or the Model Browser. This can help you use existing projects to speed up the setup of new sessions.
- Export models to a file which you can import to CAGE by selecting **File > Import > Model**

## Setting Up Calibration Tables to Fill

### In this section...

“Altering Variable Ranges” on page 2-57

“Setting Up Tables” on page 2-57

### Altering Variable Ranges

Editing variable ranges before creating tables will produce easy-to-read values (e.g., 0.1, 500, 0.2, 1000, etc.) for the table breakpoints when CAGE automatically spaces the normalizer values across the variable ranges.

- 1 Click the Variable Dictionary button to switch to the Variables view.
- 2 Select N, edit the range to 500 for the minimum, 5000 for the maximum, and press **Enter**.
- 3 Select L, edit the range to 0.1 for the minimum, 1 for the maximum, and press **Enter**.

Edit set points of variables as follows.

- 1 Click to select the variable ECP and edit the **Set Point** to 5.
- 2 Click to select the variable ICP and edit the **Set Point** to 40.

### Setting Up Tables

You can use the Create Tables From Model wizard helps you quickly create a set of tables with the same axes for all the inputs of a model, and the model response, and any other responses that share the same inputs. This wizard can be useful when creating tables for an optimization, to use when filling tables with optimization results, and for investigating results in the tradeoff views.

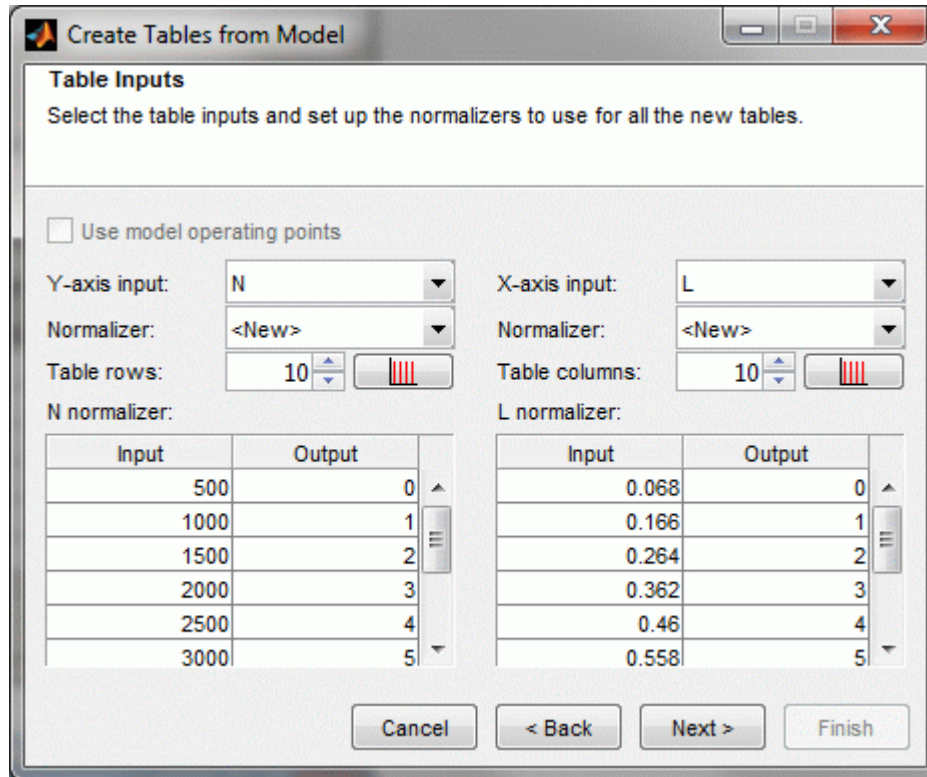
- 1 Select **Tools > Create Tables From Model** (or use the toolbar button).

The **Create Tables From Model** Wizard appears.

- 2 Select the BTQ model to base the new tables on.

Click **Next**.

- 3 Select table axes input variables and set up the normalizers to use for the new tables. Select N and L from the **X-** and **Y-axis input** drop-down menus. Leave the number of rows and columns at 10. CAGE automatically initializes the normalizers by spacing the breakpoints evenly across the range of the input variables N (speed) and L (load).



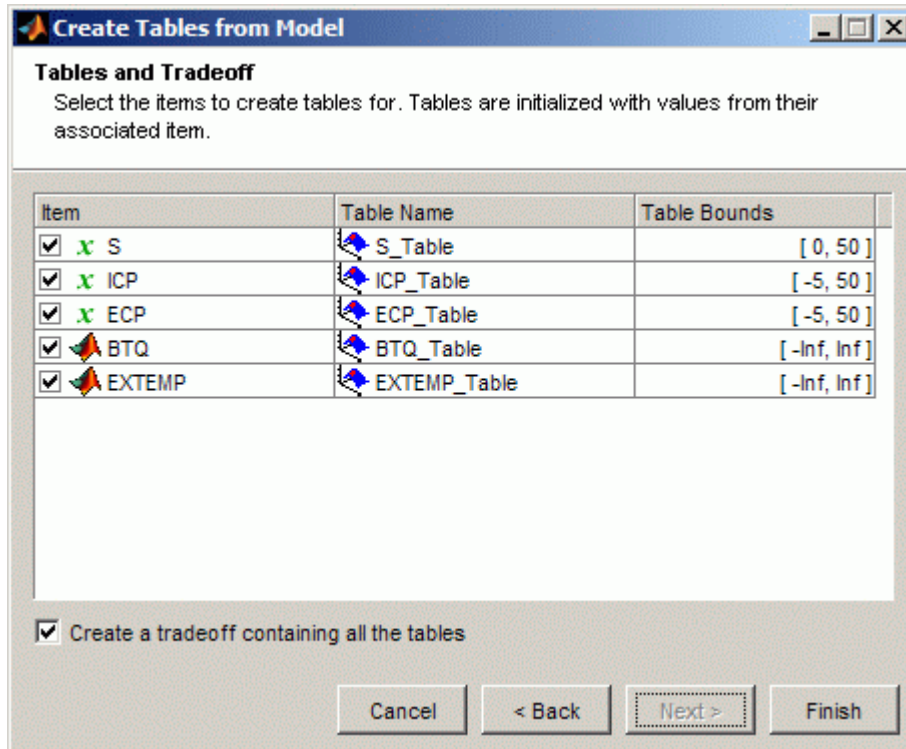
Click **Next**.

- 4 Select all check boxes to specify which variables and responses to create tables for. CAGE automatically selects the tables for all the models and the response You can create tables for other responses with exactly the same inputs as the primary model.

Note the wizard has defined **Table Bounds** matching the variable bounds for the S, ICP and ECP tables. You can also edit table bounds later in the Table Properties dialog box.



By default you will also create a tradeoff containing all of the new tables. The tradeoff can be very useful for investigating optimization results.



Click **Finish** to create the tables and tradeoff.

You see a dialog listing all the items you have created, and you can choose which (if any) of the items to view next.

Now you have tables for optimized spark and cam timings, ready to fill with optimization results.

### Setting Up the Optimization

CAGE provides a flexible optimization environment. You can define the objectives, the constraints, and the points where the optimization is carried out.

The objective is to maximize the weighted sum of torque over a set of [N, L] points subject to a set of constraints. This is a constrained single objective optimization problem.

You solve this problem in two parts:

- 1 Run an initial optimization to explore the problem point-by-point in order to obtain good starting points for the sum optimization.
- 2 Use the solutions from the initial optimization as the start points to run the sum optimization over the drive cycle to find optimal settings of spark (SPK) and cam (ICP, ECP) timing. Use these results to fill calibration tables.

CAGE has several built-in optimization routines and the capacity for you to write your own; in this case study you use `foptcon`. This is a modified version of `fmincon` from the Optimization Toolbox™ product. In CAGE, you can use the algorithm to minimize or maximize.

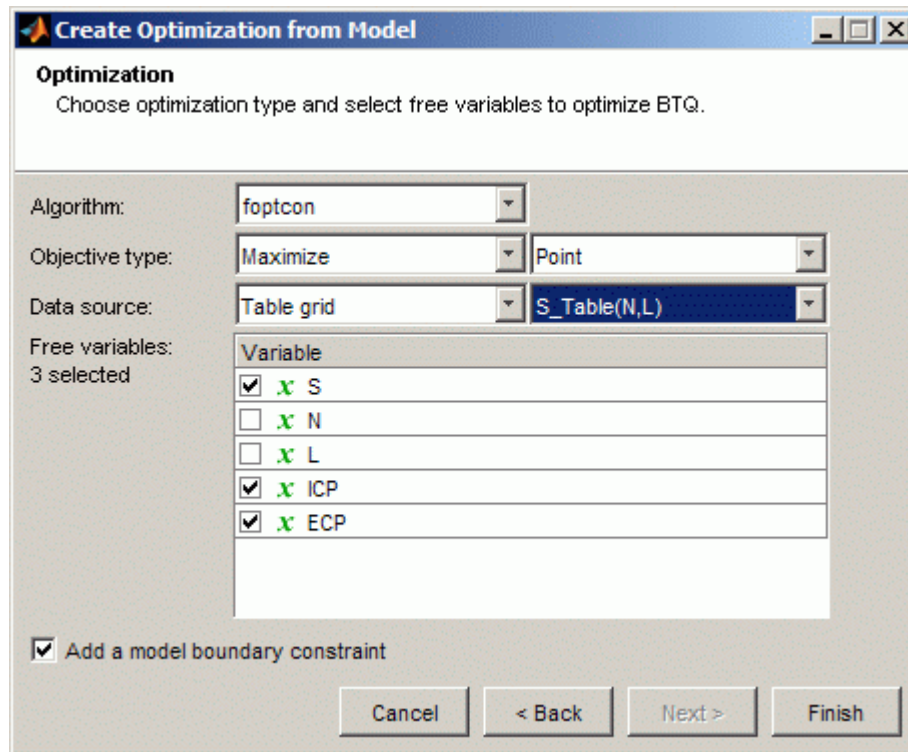
- 1 Select **Tools > Create Optimization From Model** (or use the toolbar button).

The **Create Optimization From Model** Wizard appears.

- 2 Select the BTQ model to maximize in the optimization.

Click **Next**.

- 3 Select the optimization type (algorithm, maximize or minimize, point or sum), data source for optimization, free variables, and boundary constraint, as follows:



- **Algorithm:** Use the default `foptcon` for gradient-based single-objective optimizations.
- **Objective type:** Choose to `Maximize` your model, and leave the default `Point` objective type.
- **Data Source:** Select `Table grid` and `S_Table(N,L)` to define the points where you want to run the optimization from the spark table. You can also set up points in the Optimization view.
- **Free variables:** Leave the selected check boxes of `S`, `ICP`, and `ECP`, the variables you want to optimize from the set of model inputs.

In the preceding step when you selected the table grid as the **Data source**, CAGE automatically removed the table normalizer variables `N` and `L` from the selection of free variables.

- **Add a model boundary constraint:** Leave the check box selected to constrain the optimization within the boundary model associated with your model.

Click **Finish** to create the optimization.

CAGE switches to the Optimization view. Look at the information displayed. Here you can examine and edit the objective and constraint.

The Constraints pane shows the boundary constraint you specified in the wizard.

Add a constraint on the residual fraction response to constrain the optimization to stable combustion regions. Follow these steps:

- 1 Right-click the Constraints pane and select **Add Constraint**. The Edit Constraint dialog box appears.
- 2 Leave the **Constraint type** at the default, **Model**.
- 3 Edit the **Constraint name** to RESIDFRAC.
- 4 Select RESIDFRACatMBT from the list of models on the left and click to select it for the constraint.
- 5 With the RESIDFRACatMBT model selected, enter **25** in the **Constant** edit box and press **Enter**. CAGE closes the dialog box and displays the new constraint in the Optimization view.

Ensure that the constraint **Description** reads RESIDFRACatMBT <= 25

## Defining Variable Values

You need to define the set of points where you want the optimization to run. In this case, you run the optimization over the table breakpoints in the tables. You already defined this in the Create Optimization from Model wizard. In this section you will also set up initial values for spark.

You can edit values manually in the **Input Variable Values** pane in the Optimization view, or you can import values from existing optimization output values, tables or data sets.

To import table initial values to your optimization, you will first create a feature to fill the **S\_Table** from the MBT model. You can then use this table to initialize the starting values for your optimization.

To create the feature table,

- 1 Select **File > New > Feature**. CAGE switches to the Feature view.
- 2 Select **Feature > Graphical Strategy Editor**. Three Simulink windows appear.
- 3 In the CAGE project library window, double click the Tables block. The Tables library window opens.
- 4 Drag the **S\_Table** table block from the Tables library to the New Feature window.
- 5 Click to select **S\_Table**, then **Ctrl+click** the blue **New\_Feature** block to connect the two.
- 6 Double-click the blue **New\_Feature** block to import the strategy into CAGE.
- 7 In the Feature view, click the **Select Model** button.
- 8 In the Select Model dialog, select MBT and click **OK**.
- 9 Click **Fill Feature** in the toolbar. The Feature Fill Wizard appears.
- 10 Click **Next** three times, then click the **Fill Tables** button. Click **Finish** when the process has finished.
- 11 Expand the **New\_Feature** node in the tree and select the **S\_Table** node to view the filled table. In the comparison pane you can see where the table limits cause the values to diverge from the feature model at low load.
- 12 Click the **Optimization** button to return to the Optimization view.
- 13 Select **Optimization > Import from Table Values**.

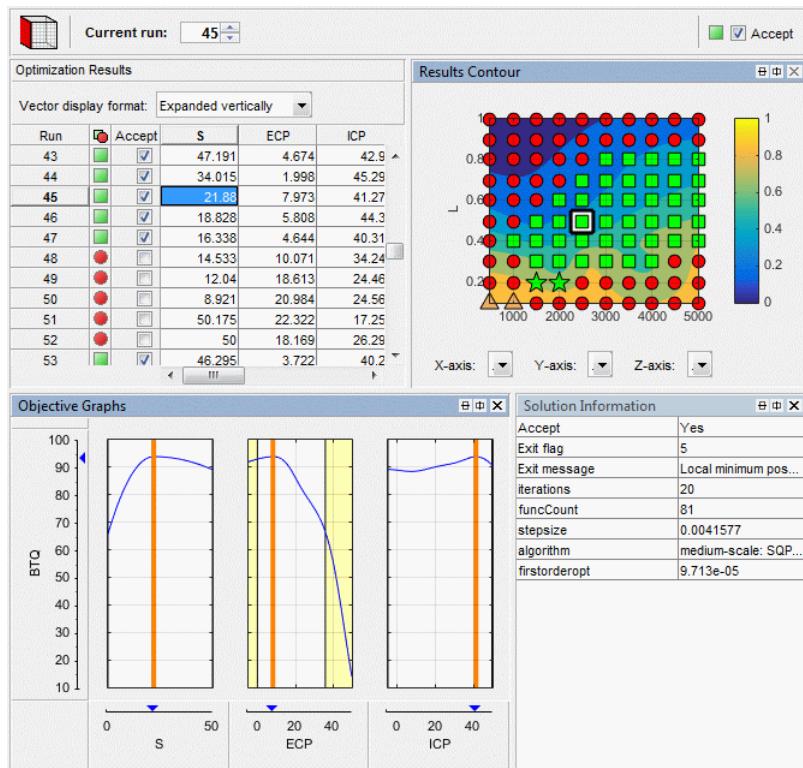
Select the check box to import values for **S**, and select **S\_Table** from the **Fill Input With** list. Click **OK** to import the table values of spark into the optimization initial values for **S**.

View the values in the **Input Variable Values** pane in the Optimization view. These values define the variable values at each point where you want the optimization to run. The fixed variable values are the table breakpoints, and the free variable initial values are the filled table values of spark, and the defaults (set point) for the other free variables.

## Running the Optimization

You have defined objectives, constraints and a set of operating points. Your optimization is ready to run.

- 1 Click Run Optimization in the toolbar.
- 2 When the optimization is complete, the view switches to the new child node, `BTQ_Optimization_Output`, in the Optimization tree under the `BTQ_Optimization` node. View the results.
- 3 Look through the solutions at different operating points by clicking cells in the Results table or contour plot. Regions that do not meet the constraint are yellow in the plots and tables.



Because this is a single-objective optimization, there is only one solution at each point.

- 4 Observe the colored icons that indicate the Accept status of each solution plotted in the Results Contour view, and of each corresponding row in the Optimization Output Values table.

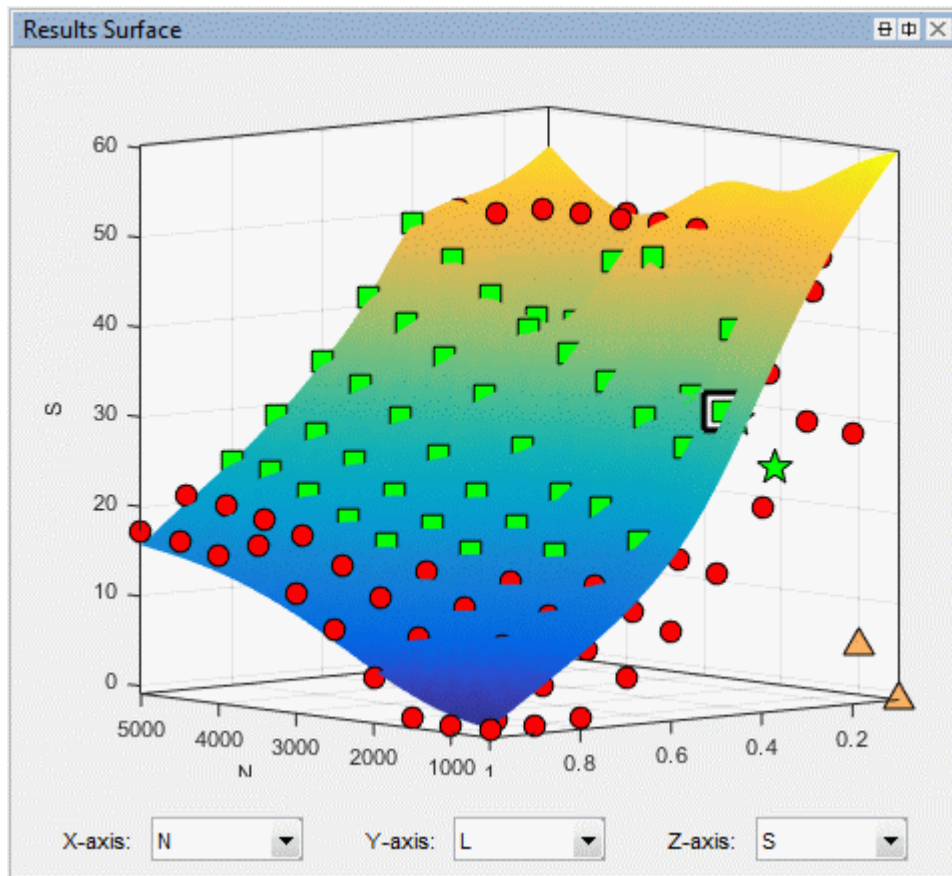
CAGE automatically selects the Accept check boxes in the table for solutions where the algorithm exit flag indicates success ( $>0$ ). These solutions show a green square icon in the table and surface. CAGE also highlights unsuccessful solutions for you to investigate.

|    |   |                                     |        |
|----|---|-------------------------------------|--------|
| 11 |  | <input type="checkbox"/>            | 4.666  |
| 12 |  | <input type="checkbox"/>            | 44.602 |
| 13 |  | <input type="checkbox"/>            | 35.963 |
| 14 |  | <input checked="" type="checkbox"/> | 32.128 |

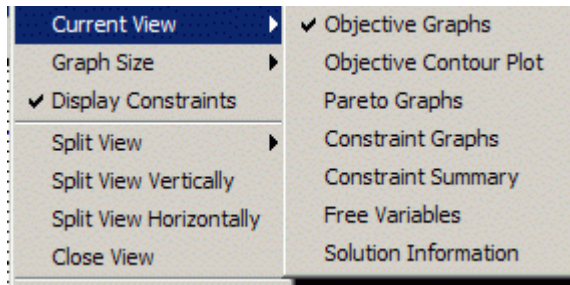
Click in the table or contour view to select solutions, and other plots (such as the Objective Contours and Objective Graphs) update.

- 5 Right-click the title bar of the Results Contour view and select **Current View > Results Surface**. Observe the Accept icons for the solutions.





- 6 Examine the orange triangle Accept status solutions. This indicates the algorithm exit flag is zero. This means the algorithm terminated because it exceeded limits on the amount of computation allowed (e.g., the algorithm ran out of iterations or function evaluations). You could decide to accept these solutions or you could try changing tolerances and optimizing again. *Do not edit any checkboxes yet.*
- 7 Use the plot title bar buttons to split views, or right-click the title bar and use the context menu to try all the available views for analyzing your results. The **Objective Contour** plot of ECP against ICP is useful for analyzing results. Also check the **Objective Graphs** for spark to ensure that maximum torque is obtained.



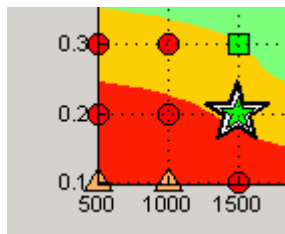
- 8 Export all acceptable results to tables to check surfaces. Click Fill Tables in the toolbar and use the wizard. If you need instructions, see “Filling Tables with Optimization Results” on page 2-75.

Check trends, and don't worry about smoothness. You will apply table gradients constraints to smooth the table surfaces in the next optimization.

- 9 You can change the Accept selections using the check boxes for each solution, or by right-clicking points in the surface view. You can use these Accept selections to choose solutions within the table for use in filling tables, exporting to data sets and importing to other optimization starting values.



Reject points 22 and 32 (clear the check boxes). The points are in the wrong region with respect to adjacent points in the table. Solutions where you have altered the check box status show an asterisk in the table and a star in the Results Surface or Contour plot.



- 10 If you are displaying the Objective Contours plot, you may want to turn off the display of constraints in the plot before running the next sum optimization or viewing the output node for the sum optimization. This will save time displaying

the output views. Right-click the contour plot and toggle **Display Constraints** off. Alternatively close the Objective Contours plot.

# Setting Up the Sum Optimization

### In this section...

“Setting Up the Optimization Initial Values and Objective” on page 2-70

“Creating Table Gradient Constraints” on page 2-71

“Running the Sum Optimization” on page 2-73

## Setting Up the Optimization Initial Values and Objective

The aim of this case study is to produce optimized tables for

- Intake cam phase
- Exhaust cam phase
- Spark timing schedules

as a function of load and RPM, subject to the following constraints:

- Constrain solutions to lie within the boundary constraint model
- Constrain cam phase solutions so they do not change by more than  $10^\circ$  between table cells (that is, no more than  $10^\circ$  per 500 RPM change and per 0.1 load change.)
- Constrain residual fraction  $\leq 25\%$  at each drive cycle point

The optimization objective is to maximize the weighted sum of torque over a set of [N, L] points subject to the constraints described.

The initial point-by-point optimization was designed to obtain good starting points for the sum optimization. Now, you can create a sum optimization from the results of that initial optimization to run the sum problem. You use the solutions from the initial optimization as the start points to run the sum optimization over the drive cycle to find optimal settings of spark and cam timings. You use these results to fill calibration tables.

- 1 To create a sum optimization from your point optimization output node, select **Solution > Create Sum Optimization**.

CAGE creates a new optimization named `Sum_BTQ_Optimization`. The optimization has these characteristics:

- The objective matches your original optimization but converted to a sum objective.
- The new optimization has identical constraints to your original optimization.
- Your original fixed and free variables are converted to a sum optimization (a single run with multiple values).
- The new optimization uses only accepted solutions from your original optimization output (all runs with a selected Accept check box). Therefore, the number of accepted solutions you had in the original optimization determines the number of values within the sum optimization run.
- The free variable initial values and fixed variable values are populated from your point optimal results (accepted solutions only).
- The fixed variables have a Weights column with every value set to 1.

In the Optimization view, observe the new sum objective, and the new values in the Input Variable Values pane.

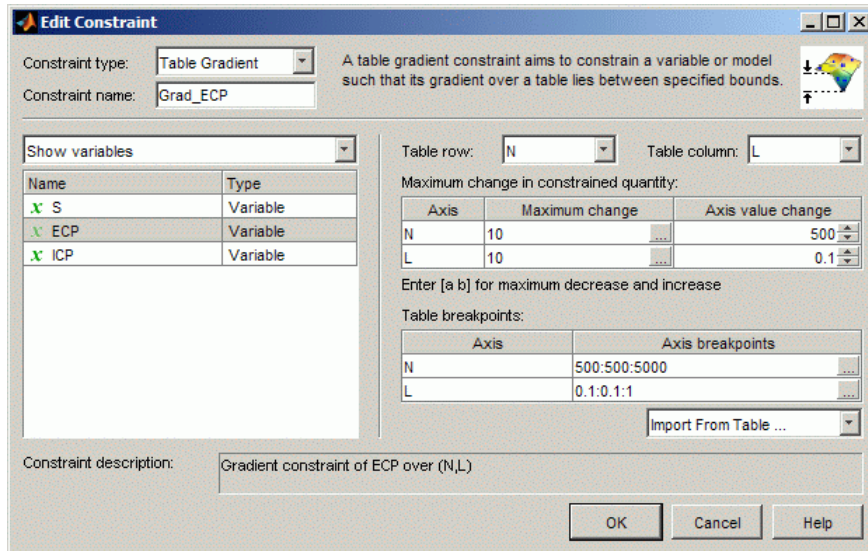
Now, your sum optimization is ready to run. Before you do this, see the next section to create table gradient constraints.

## Creating Table Gradient Constraints

To meet engineering constraints it is essential that some calibration tables meet a level of smoothness. For example, large cam phase changes between adjacent table cells cannot be used because the controls cannot physically move that fast. Achieving a level of smoothness is important for variable valve timing calibration problems. Unconstrained optimizations may find solutions where the cam timings change too rapidly across tables. You can apply table gradient constraints to make sure the resulting tables are smooth enough.

To apply a gradient constraint to restrict the change in EXH between adjacent table cells:

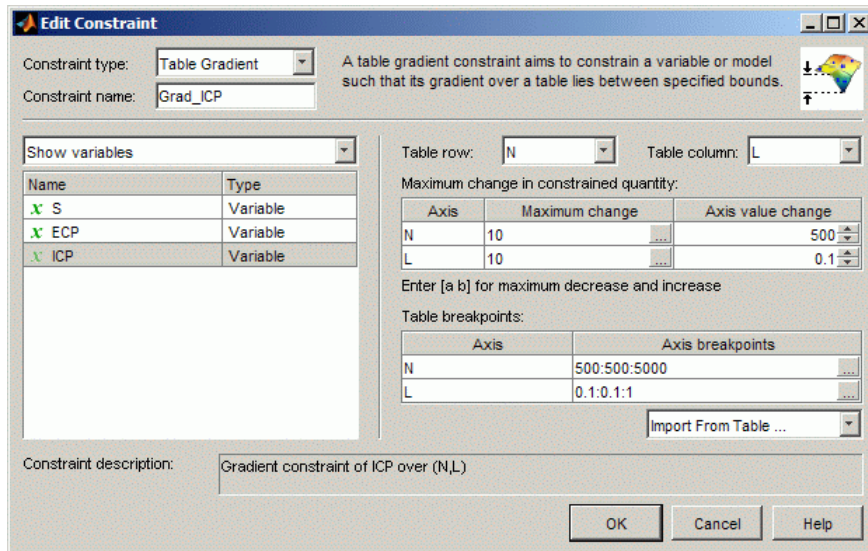
- 1 Right-click in the Constraints pane, and select **Add Constraint**. The Edit Constraint dialog box appears.



- 2 Select **Table Gradient** from the **Constraint type** drop-down menu.
- 3 Enter **Grad\_ECP** in the **Constraint name** edit box.
- 4 Select the variable **ECP** to constrain.
- 5 Select the table axes **N** (from the **Table row** drop-down) and **L** (from the **Table column** drop-down).
- 6 Select **ECP\_Table** from the **Import From Table** drop-down menu. The breakpoints of this table now appear in the breakpoints edit boxes: **500:500:5000** in **N**, and **0.1:0.1:1** in **L**.
- 7 Apply the gradient constraint across the breakpoints of the table. As shown in the figure, enter **10** in the **N axis Maximum change** edit box, and verify **500** is in the **Axis value change** edit box. This specification constrains the maximum change to 10 degrees **ECP** per 500 change in **N** (the difference between cells, as shown in the axis breakpoints edit box).
- 8 Next, enter **10** in the **L axis row Maximum change** edit box, and verify that **0.1** is shown in the **Axis value change**, to constrain the maximum change to 10 degrees per 0.1 change in **L**.
- 9 Click **OK** to close the **Edit Constraint** dialog box.

Similarly, add another gradient constraint to restrict the change in INT between adjacent table cells as follows:

- 1 Right-click the Grad\_ECP constraint in the Constraints pane and select **Duplicate Constraint**. The Edit Constraint dialog appears showing a copy of the settings for your previous constraint.
- 2 Change the **Constraint name** to Grad\_ICP.
- 3 Select the variable ICP to constrain.
- 4 Leave the other settings to match the ECP constraint: the table axes, the breakpoints, and the maximum change values. These settings constrain the maximum change to 10 degrees ICP per 500 change in N and per 0.1 change in L.



- 5 Click **OK** to dismiss the Edit Constraint dialog.

## Running the Sum Optimization

To run the optimization:

- 1 Click Run Optimization in the toolbar.
- 2 When the optimization is complete, the view switches to the child node, Sum\_BTQ\_Optimization\_Output, in the Optimization tree under the

`Sum_BTQ_Optimization` node. View the results. For guidance see “Interpreting Sum Optimization Output”.

Remember you can check the example project, `Gasoline_optimization.cag`, , found in `matlab\toolbox\mbc\mbctraining`, to view completed examples of the optimizations and filled tables.



## Filling Tables with Optimization Results

- 1 From the `Sum_BTQ_Optimization_Output` view, select **Solution > Fill Tables**, or use the toolbar button.

The Table Filling Wizard appears.

- 2 **Shift**-click to multi-select all the tables and click the button to add the tables to the filling list. Click **Next**.
- 3 Verify that the filling factors are correct. CAGE automatically selects factors in the optimization results for table filling, because you used the wizards to create your tables and optimization.

Note that you could also choose the MBT model to fill a table, for example if you wished to compare your optimization results with the spark timings specified by the MBT model. You will fill and optimize tables from models in more detail in the next section.

Click **Next**.

- 4 The last screen is for fill algorithm settings, such as filter rules to use subsets of results, and settings for repeated table filling such as locked cells and extrapolation masks. You can use the default settings.

Make sure the check box **Use acceptable solutions only** is selected, and click **Finish** to fill the tables.

A dialog appears with the message that the tables have been filled successfully. Click **OK**.

- 5 Switch to the **Tables** view to view the filled tables. Click **Tables** in the **Data Objects** pane, and select the filled tables in turn.

# MBT Spark Estimator Problem

### In this section...

- “What Is an Estimator?” on page 2-76
- “View the Feature” on page 2-76
- “View Variables” on page 2-78
- “Edit and Import Boundary Model” on page 2-79
- “Use the Feature Fill Wizard” on page 2-80
- “Inspect Results” on page 2-84
- “CAGE Import Tool” on page 2-87

## What Is an Estimator?

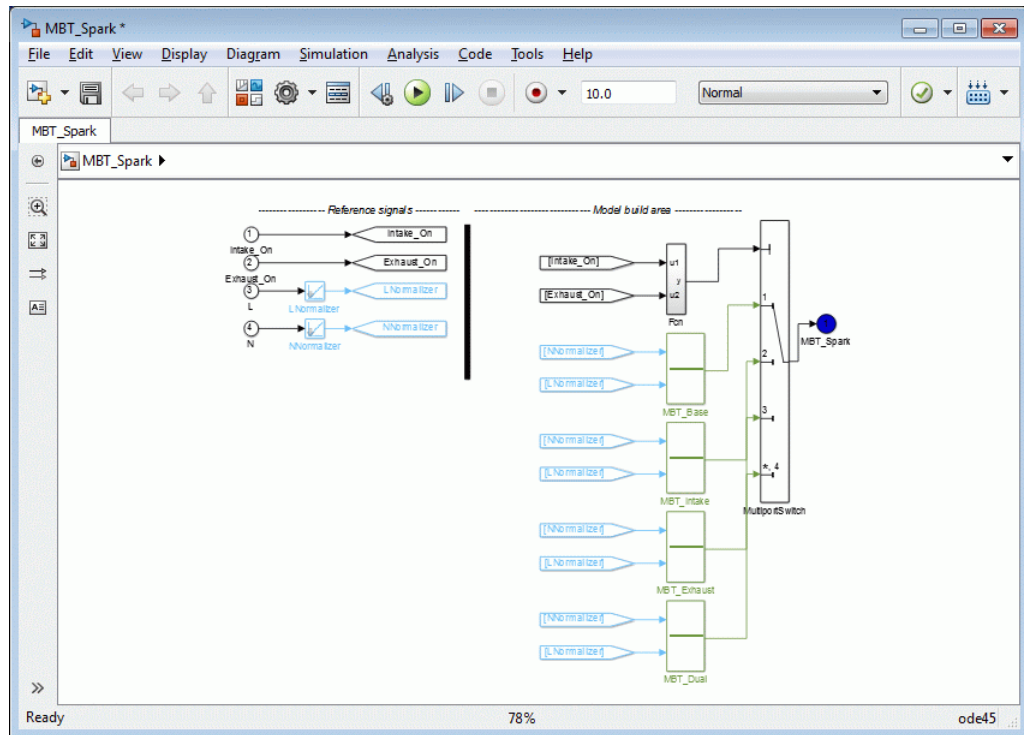
The cam timings optimization results solved a control problem. You can also use CAGE to calibrate estimator problems. Here you will use an MBT model to produce a spark estimator feature which estimates MBT spark when each cam is on or off, using the cam timings found by the optimization.

## View the Feature

- 1 Select **File > Open Project** and load the example project, `Gasoline_optimization.cag`, found in `matlab\toolbox\mbc\mbctraining`.
- 2 Click **Feature** in the **Processes** pane to go to the Feature view.

The features `Exhaust_CAM`, `Intake_CAM`, and `MBT_Spark` are in the Feature tree. The `MBT_Start_Feature` was used to provide initial values for the optimization, as described in “Defining Variable Values” on page 2-63.

- 3 Select `MBT_Spark` and view the strategy by selecting **Feature > Graphical Strategy Editor**.
- 4 Examine the strategy model. Observe the Multiport Switch block which switches between MBT tables depending on whether each cam is on or off. The variables `Intake_On` and `Exhaust_On` are used to define whether cams are parked or not. All the MBT tables (with and without cams) share the same speed (N) and load (L) normalizers.



If you wanted to import this strategy model, you would double-click the MBT\_Spark blue output to parse the strategy into CAGE. This strategy has already been imported into this project so just close the model window.

- 5 Expand the MBT\_Spark feature in the tree to see the MBT tables that have been created by importing the strategy: MBT\_Base, MBT\_Intake, MBT\_Exhaust, and MBT\_Dual. These tables share the same normalizers in speed and load.
- 6 Similarly view the strategies for the Exhaust and Intake CAM features. These features switch between parked cams and active cams depending on the threshold value. The CAM features define the cam inputs and switch between the optimal CAM tables (from the optimization results) and the parked values. You can use the linking functionality in the Feature Filling Wizard to connect features and tables to models.

### **View Variables**

You need some variables and constants to define when the cams are parked so the strategy can switch between optimal cam timings and parked cams at a threshold value.

View how this is done:

- 1 Click **Variable Dictionary** in the **Data Objects** pane to switch to the Variable Dictionary view.
- 2 Click to select the new variable `Exhaust_On`.
- 3 Observe the **Minimum** is 0 and the **Maximum** is 1.
- 4 The variable `Intake_On` has the same values.

Also two constants define the parked cam positions:

- 1 Select `Exhaust_Parked`.
- 2 Observe the **Set Point** of this constant is 0.
- 3 Select `Intake_Parked`; this has a **Set Point** of 0.

## Edit and Import Boundary Model

To edit the boundary model and import it to CAGE,

- 1 Open the Model Browser (enter `mbcmodel` at the MATLAB command line).
- 2 Load the example project. Select **File > Open Project**, locate and select `Gasoline_project.mat`, found in `matlab\toolbox\mbc\mbctraining`.
- 3 Select the DIVCP test plan node in the tree.
- 4 Select **TestPlan > Boundary Models** (or click **Edit boundary models** in the toolbar).

The Boundary Editor opens.

- 5 Select the **Star shaped** constraint in the tree, and click **Remove boundary model from best** in the toolbar. You need only **Star shaped(N, L)** in the collection of best boundary models.
- 6 Close the Boundary Editor. In the Model Browser, observe the single constraint **Global/Star shaped(N, L)** listed under Boundary Model in the right information pane.
- 7 Return to CAGE and select **File > Import From Project**.

The CAGE Import Tool appears. You can import directly from the Model Browser when it is open, and the CAGE Import Tool automatically displays the available items.

- 8 Select `knot` (the datum model) from the list.


- 9 Click the **Import Selected Items** button.

The Import dialog opens displaying the model you selected for import.

- 10 Double-click the **CAGE Model Name** column cell to edit the name to MBTwithSpeedLoadBoundary, and click **OK** to import the model.
- 11 Close the CAGE Import Tool.

### Use the Feature Fill Wizard

You can use the Feature Fill Wizard to fill and optimize the values in tables by reference to the model. You can fill multiple tables at once using the wizard, and you can Fill from the top feature node or from any table node in a feature.

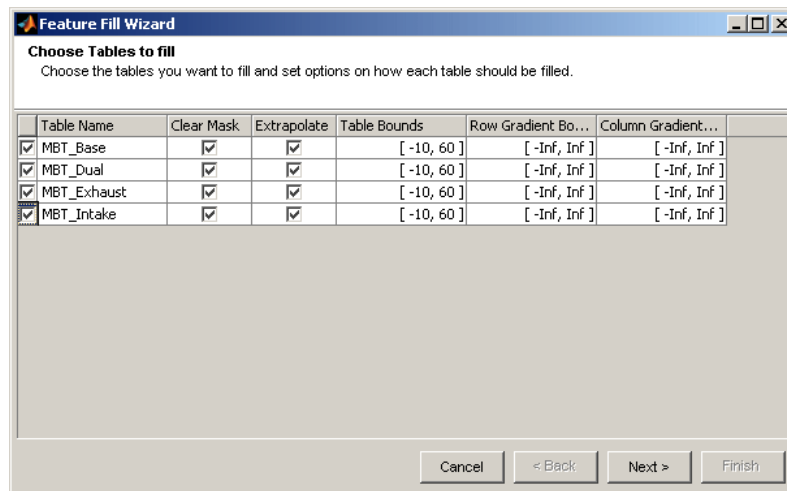
- 1 Click **Feature** in the **Processes** pane to return to the Feature view, then select the feature node MBT\_Spark.
- 2 Click  in the toolbar, or select **Feature > Fill**. This opens the Feature Fill Wizard.

---

**Note:** The Feature Fill Wizard is modal so you cannot scroll the help after opening the wizard. You might want to review the wizard steps before opening the wizard.

---

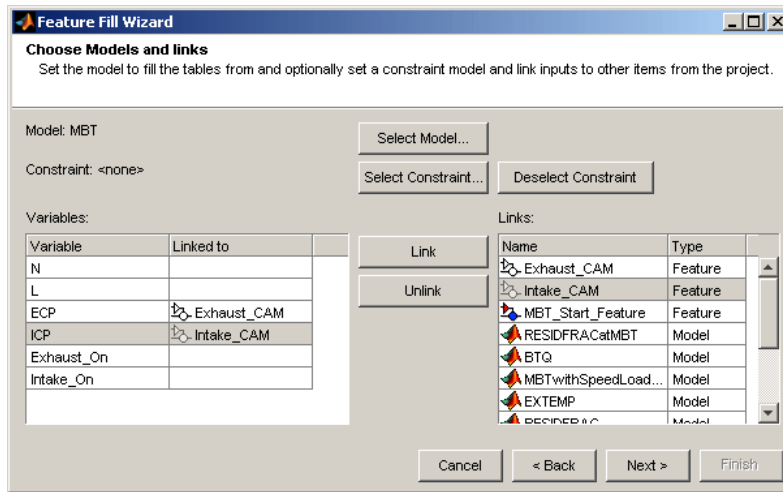
- 3 Select all four table check boxes to fill all tables.



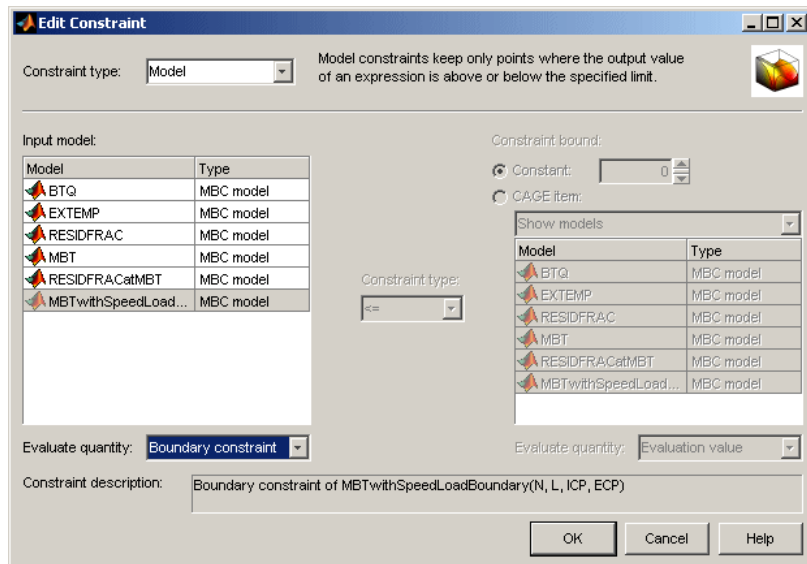
You could also explore setting gradient bounds to constrain table filling for smoothness.

This time leave the other settings at the defaults and click **Next**.

**4** Choose filling model, constraint, and links.

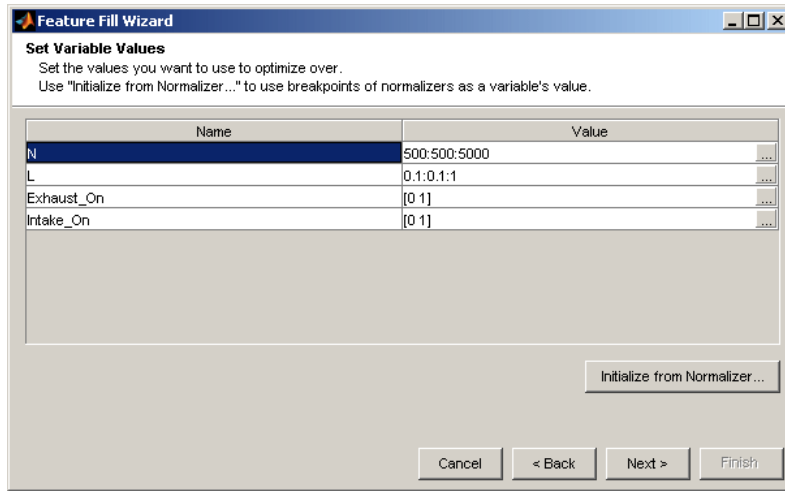


- a** Make sure MBT is the **Model** to fill the tables.
- b** Select ECP in the left **Variables** list and the Exhaust\_CAM feature in the right **Links** list and click **Link**.
- c** Select ICP in the left list and Intake\_CAM in the right list and click **Link**.
- d** Click **Select** next to Constraint. The Edit Constraint dialog box opens.



- e Select the model **MBTwithSpeedLoadBoundary**, select **Boundary constraint** as the **Evaluate quantity**, and click **OK**. This boundary model constraint ensures you only fill over speed/load points where the data was collected.
  - f When you return to the Feature Fill Wizard, click **Next**.
- 5 Set values to optimize over.





- Enter 0, 1 in the Exhaust\_On Value edit box, and press **Enter**.
- Enter 0, 1 in the Intake\_On Value edit box, and press **Enter**.

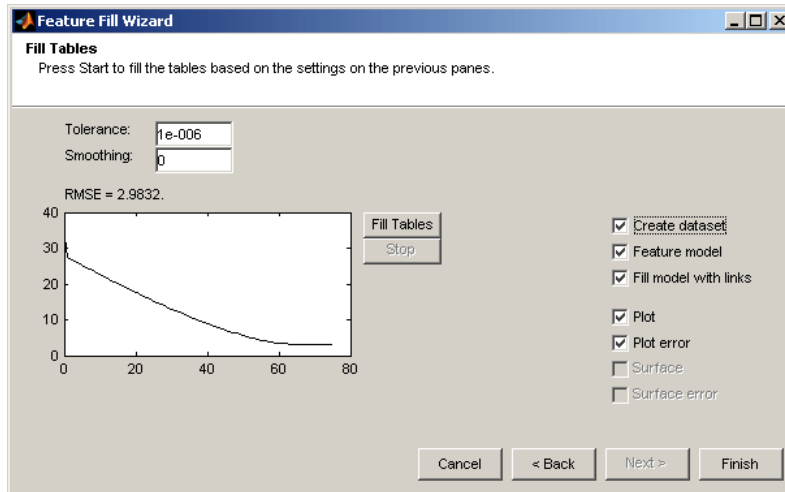
You use these values because the strategy includes the following tables.

| Table                  | Intake_On Value | Exhaust_On Value |
|------------------------|-----------------|------------------|
| MBT_Base (cams parked) | 0               | 0                |
| MBT_Exhaust            | 0               | 1                |
| MBT_Intake             | 1               | 0                |
| MBT_Dual               | 1               | 1                |

The N and L normalizer values are automatically selected. You can edit normalizers manually, or you can click the **Initialize From Normalizer** button here to reach a dialog where you can select normalizers and interleave values between breakpoints. Interleaving values can minimize interpolation error by adding values between each normalizer value. In this way, you can create a grid of more points than table cells to optimize over. Leave the setting alone for now.

Click **Next**.

- 6 Fill tables and generate plots. Click the **Fill Tables** button. Watch the graph as the optimization progresses.

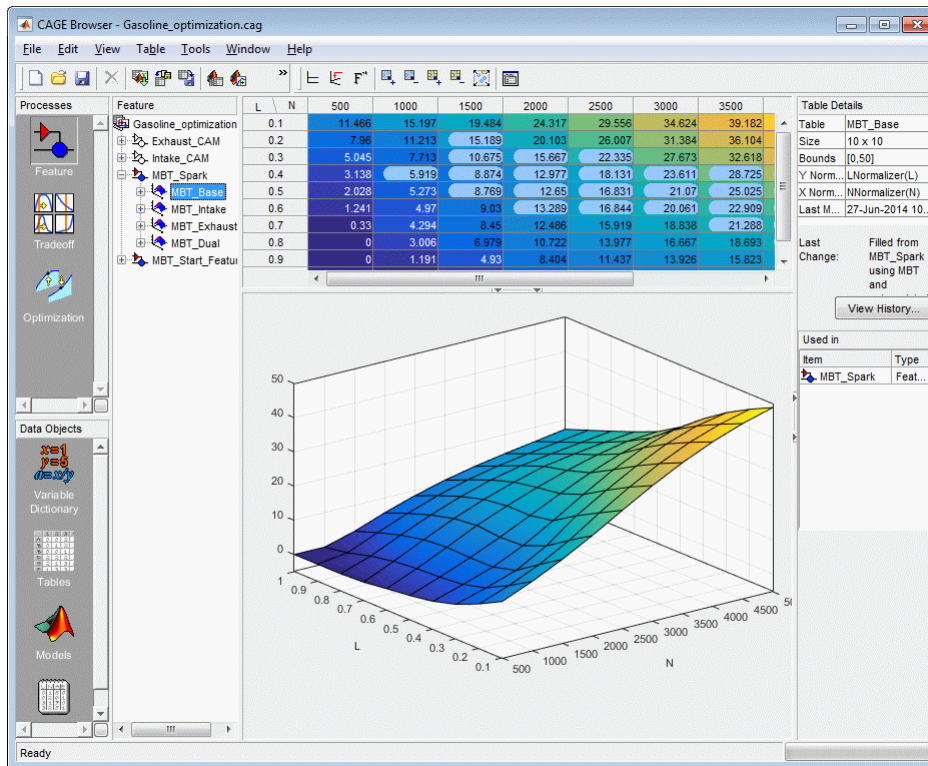


When it is finished, select all enabled check boxes, and click **Finish**. Plots appear summarizing the feature fill data.

### Inspect Results

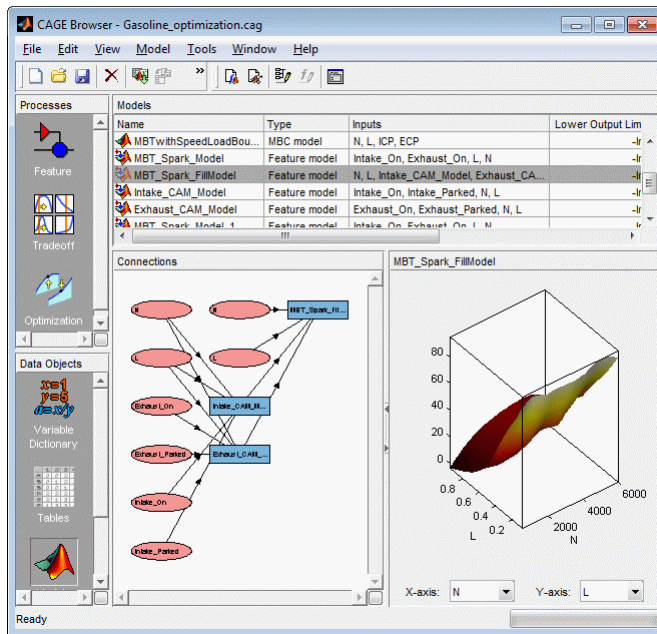
Look at the filled tables, linked models and exported data set.

- 1 In the Feature view, saved fill settings appear in the Feature Ffill Settings pane, and the Feature Tables pane shows the tables you filled.
- 2 Select in turn the tables MBT\_Base, MBT\_Intake, MBT\_Exhaust, and MBT\_Dual in the feature tree.




Observe the blue mask area of cells in each table — the mask is defined by the limits from the MBTwithSpeedLoadBoundary boundary constraint model you selected. The rest of the table values are extrapolated after the mask cells are filled by the Feature Fill Wizard (specified by the **Extrapolate** check box). Table values are limited to [-10 60] as specified in the **Table Bounds** in the Feature Fill Wizard.

- 3 Click **Models** to select the Models view. Look at the feature model and fill model MBT\_SPARK\_Model and MBT\_SPARK\_FillModel. If you can't view the whole Connections diagram of MBT\_SPARK\_FillModel, right-click and select **Zoom To Fit**.



The linking functionality in the Feature Fill Wizard allows features and tables to be connected to models and any expression to be connected to feature inputs. These links can be made permanent by creating feature models and fill models that are static snapshots of the table on finishing the feature fill wizard (specified by the **Feature model** and **Fill model with links** check boxes on screen 4 of the Feature Fill Wizard).

Notice that the CAM features are converted to feature models (Exhaust\_CAM\_Model and Intake\_CAM\_Model are new feature models) and they are connected to MBT\_SPARK\_FillModel.

- 4 Click **Data Sets** to select the Data Sets view. Select the dataset MBT\_SPARK\_FillResults to study the gridded data, the model and feature values for the feature fill. Click View Data  in the toolbar to see the data table view. All the links in the feature fill process are defined in this data set — try clicking column headers to see highlighted linked input columns.

The screenshot shows the CAGE Browser window titled "Gasoline\_optimization.cag". The interface includes a menu bar (File, Edit, View, Data, Tools, Window, Help), a toolbar, and several panels. The "Data Sets" panel displays a table with 17 rows and 10 columns. The "Project Expressions" panel at the bottom right shows a list of expressions with their types and information.

|    | N    | L   | ECP <sub>opt</sub> | ICP <sub>opt</sub> | Exhau... | In |
|----|------|-----|--------------------|--------------------|----------|----|
| 1  | 1500 | 0.2 | 0                  | 0                  | 0        | 0  |
| 2  | 1500 | 0.3 | 0                  | 0                  | 0        | 0  |
| 3  | 2000 | 0.3 | 0                  | 0                  | 0        | 0  |
| 4  | 2500 | 0.3 | 0                  | 0                  | 0        | 0  |
| 5  | 1000 | 0.4 | 0                  | 0                  | 0        | 0  |
| 6  | 1500 | 0.4 | 0                  | 0                  | 0        | 0  |
| 7  | 2000 | 0.4 | 0                  | 0                  | 0        | 0  |
| 8  | 2500 | 0.4 | 0                  | 0                  | 0        | 0  |
| 9  | 3000 | 0.4 | 0                  | 0                  | 0        | 0  |
| 10 | 3500 | 0.4 | 0                  | 0                  | 0        | 0  |
| 11 | 4000 | 0.4 | 0                  | 0                  | 0        | 0  |
| 12 | 4500 | 0.4 | 0                  | 0                  | 0        | 0  |
| 13 | 1500 | 0.5 | 0                  | 0                  | 0        | 0  |
| 14 | 2000 | 0.5 | 0                  | 0                  | 0        | 0  |
| 15 | 2500 | 0.5 | 0                  | 0                  | 0        | 0  |
| 16 | 3000 | 0.5 | 0                  | 0                  | 0        | 0  |
| 17 | 3500 | 0.5 | 0                  | 0                  | 0        | 0  |

| Expression          | Type                  | Information |
|---------------------|-----------------------|-------------|
| BTQ                 | MBC model             | Inputs req. |
| BTQ_Table           | 2D table              |             |
| ECP                 | Linked to Exhaust_CAM | In data set |
| ECP_Table           | 2D table              |             |
| Exhaust_CAM         | Feature               |             |
| Exhaust_CAM_Model   | Feature model         |             |
| Exhaust_CAM_Model_1 | Feature model         |             |

## CAGE Import Tool

Suppose that you are calibrating a similar engine at a later date. You would build new models for this engine and then want to solve the same problems in CAGE. The CAGE Import Tool allows you to reuse the setup from your old CAGE session. All that is necessary to import new models on top of the existing ones and rerun the optimizations and feature fill problems. For example you could import new BTQ (replace BTQ), knot (replace MBT and MBTwithSpeedLoadBoundary) and EXTEMP models from the example file `Gasoline_Project.mat` as follows:

- 1 Select **File > Import From Project**.

The CAGE Import Tool appears.

- 2 You can choose a project file or import directly from the Model Browser if it is open. If the Model Browser is open, the CAGE Import Tool automatically shows the items available. Use the **Import From Project File** button to select a file.

If you are choosing a project file, a file browser dialog opens. Locate `Gasoline_Project.mat` and click **Open**.

- 3 The CAGE Import Tool displays the available items. Select the items you want to import from the list: BTQ, EXTEMP, and knot.
- 4 Click the **Import Selected Items** button.
- 5 The Import dialog opens displaying the items you selected for import.
  - Double-click the **CAGE Model Name** column cells to edit item names.
  - Choose to replace BTQ, EXTEMP and MBT. For knot, select **Replace** from the **Action** list, then double-click the **CAGE Model Name** column cells to open a dialog to select the correct item, MBT, to replace.
  - Click **OK** to import the items.

Now you can run the optimization again to generate new optimal CAM timings with new models. Export the optimization results to the INTCAM and EXHCAM tables, and use the Feature Fill Wizard to fill the MBT\_SPARK strategy using the same steps as before.

# Diesel Engine Calibration Case Study

---

This case study provides a step-by-step guide to using the Model-Based Calibration Toolbox product to solve a diesel engine calibration problem. This section includes the following topics:

- “Diesel Case Study Overview” on page 3-2
- “Design of Experiment” on page 3-4
- “Modeling” on page 3-15
- “Optimized Calibration” on page 3-29
- “Importing Models of Engine Responses into CAGE” on page 3-32
- “Defining Additional Variables and Models” on page 3-33
- “Setting Up Calibration Tables to Fill” on page 3-35
- “Setting Up the Point Optimization” on page 3-36
- “Setting Up Constraints” on page 3-39
- “Defining Variable Values” on page 3-43
- “Running the Optimization” on page 3-45
- “Setting Up the Sum Optimization” on page 3-47
- “Filling Tables with Optimization Results” on page 3-52

## Diesel Case Study Overview

|                                  |
|----------------------------------|
| <b>In this section...</b>        |
| “Introduction” on page 3-2       |
| “Problem Definition” on page 3-2 |

### Introduction

This case study is an example of a diesel engine control calibration, for a six-cylinder 9.0 L common-rail diesel engine with VGT (variable geometry turbo) and cooled EGR (exhaust gas recirculation). It is applied in an off-road application with a very narrow engine speed range from 1600 to 2200 RPM. The aim of the case study is to produce optimal SOI (start of injection), base fuel, VGT, and EGR calibration schedules as a function of commanded torque and RPM. It involves models for torque, peak pressure, equivalence ratio, exhaust temperature, VGT speed, and EGR mass fraction. The optimization setup in CAGE is based on an 8-mode off-road emission test, approximated to 7 mode points by neglecting the idle operating point of the engine.

The Model Browser part of the example takes you through the following steps:

- 1 Create a design for your experiment “Design of Experiment” on page 3-4
- 2 Create models from the collected data “Modeling” on page 3-15

The Model Browser section of the case study covers design of experiment, data handling, and model construction and export. In the later CAGE browser section of the case study you use the models to complete the optimization of the calibration tables, see “Optimized Calibration” on page 3-29.

### Problem Definition

Produce optimal calibration tables in speed and torque for

|                       |              |
|-----------------------|--------------|
| Best injection timing | soi          |
| Best fuel quantity    | basefuelmass |
| Best fuel pressure    | fuelpress    |
| Best VGT              | grackmea     |



|          |               |
|----------|---------------|
| Best EGR | egrpos, egrmf |
|----------|---------------|

Minimize mode-weighted brake specific fuel consumption, subject to constraints on

- Turbo speed (vtgrpm)
- Cylinder pressure (pkpress)
- Exhaust equivalence ratio (exheqr)

To solve this problem, you must first use the Model Browser part of the Model-Based Calibration Toolbox product to design an experiment for collecting data, and then create models based on that data. You will use the resulting models in the CAGE Browser part of the toolbox to produce optimal calibration tables.

## Design of Experiment

**In this section...**

“Introducing Design of Experiment” on page 3-4

“Constraining the Design” on page 3-6

“Creating Candidate Designs” on page 3-13

“Data Collection” on page 3-14

### Introducing Design of Experiment

Creating a design in the Model-Based Calibration toolbox comprises several steps. First, you need to enter the ranges and names of the variables being used and choose a default model. Then you can create an initial design and set up the constraints on the space. These constraints will be the same for all designs. From this constrained design, you can create a series of child designs adding varying numbers of points and using different construction techniques. You can choose the final design by comparing the statistics of the various child designs, while considering how many test points you can afford to run.

Variables are

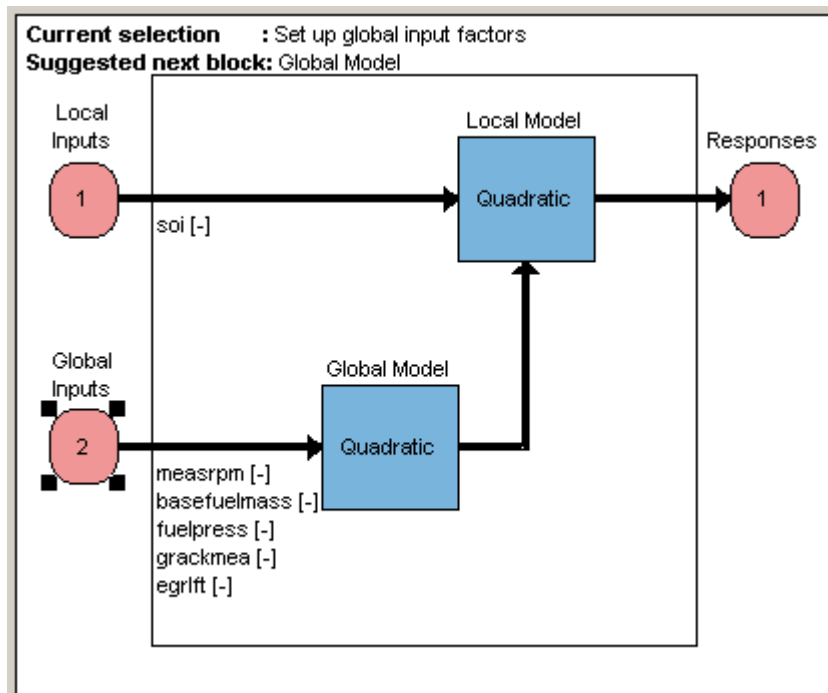
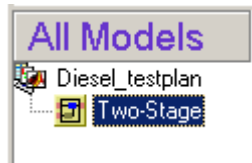
|              |                                  |
|--------------|----------------------------------|
| measrpm      | Engine speed (rpm)               |
| basefuelmass | Fuel quantity per injection (mg) |
| fuelpress    | Fuel pressure (MPa)              |
| grackmea     | VGT rack position (%)            |
| egr1ft       | EGR valve position (mm)          |
| soi          | Start of injection (deg ATDC)    |

You need to set up a test plan before you can make designs. This experiment is set up as a two-stage test plan with start-of-injection (SOI) sweeps at the local level and the other five variables at the global level.

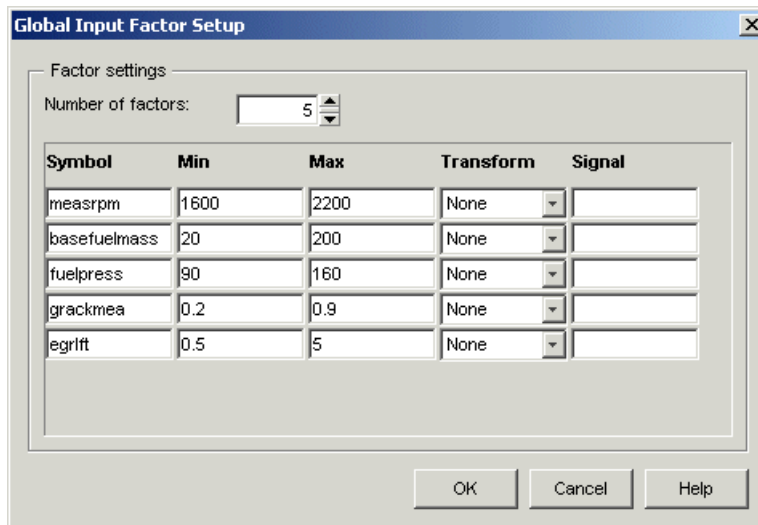
Open the example session with the test plan set up as follows:

- 1 Start the Model Browser by typing `mbcmodel` at the MATLAB command line.

- 2 Select **File > Open Project**. Locate the example session with the test plan set up, Diesel\_testplan.mat, in the mbctraining directory and double click to load the project.
- 3 Click the Two-Stage test plan node in the model tree to see the test plan diagram.



- 4 Double-click the Global Inputs block in the diagram to set the ranges of the inputs. You should set up the ranges before designing an experiment. You can enter the ranges in the min/max boxes to include the most extreme values you want to set for each variable. Check the ranges match those shown in the following example, then click **OK**.



- 5 Double-click the Global Model block in the test plan diagram to view the model type. For this exercise, leave the model type at the default, which is a quadratic in all factors. Click **OK** to dismiss the dialog.

Remember that the statistical usefulness of different designs depends on the model type. For example, if you think you need cubic instead of quadratic in EGR, the number of points required rises dramatically and this has a highly adverse effect on the statistical quality of the designs.

Some possible models are

- Cubic polynomial, quadratic in fuel pressure: 41 terms
- Cubic polynomial, quadratic in fuel pressure and EGR: 31 terms

However, you do need to bear in mind that the final model will probably not be either of the possibilities listed here, because some terms will have been removed, or it might even be an RBF (radial basis function). You choose the most suitable model you can in order to construct a design, then when you have collected the data you might find that a different model type produces the best fit.


## Constraining the Design

These are the constraints you want to apply to the design space:

- basefuelmass
  - Maximum 200 at 1600 rpm, 175 at 2200 rpm
- fuelpress
  - Range 90 - 110 at 1600 rpm
  - Range 120 - 160 at 2200 rpm
- grackmea
  - Range 0.2 - 0.6 at 1600 rpm
  - Range 0.4 - 0.9 at 2200 rpm

The tables here are very simple: one output value defined at the min and max settings of RPM. The final constraint is a cube within the base fuel mass-fuel pressure-VGT space that moves and changes size as RPM is altered.

To add a constraint to a design,

- 1 First open the Design Editor by right-clicking the Global Model block in the test plan diagram and selecting **Design Experiment**.
- 2 Click the New Design  button in the toolbar or select **File > New Design**. A new node called **Linear Model Design** appears.


The new **Linear Model Design** node is automatically selected. An empty Design Table appears because you have not yet chosen a design, unless you have previous design views from other sessions. The Design Editor retains memory of view settings.

- 3 Select **Edit > Constraints** from the Design Editor menus.
- 4 The Constraints Manager dialog appears. Click **Add**.
- 5 The Edit Constraint dialog with available constraints appears. Make sure the default **1D Table** is selected from the **Constraint Type** drop-down menu. These are easier to set up than linear constraints, although working out the linear constraint numbers might be worthwhile for larger problems as it is faster.
- 6 You can select the appropriate factors to use. For the first constraint, choose **measrpm**, **basefuelmass**, and the inequality **<=** from the menus.

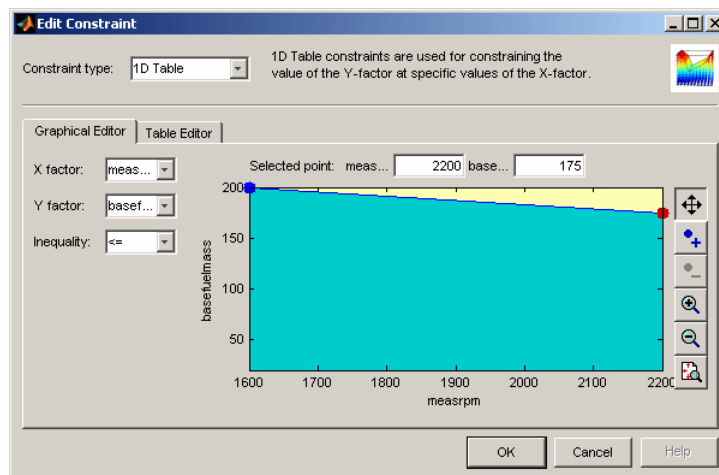
You can define the constraint by typing values in the edit boxes or by moving the large dots (clicking and dragging them) to define a boundary. For this constraint, you want to define two points.

- 7 Select the **Table Editor** tab and edit the **Number of breakpoints** to 2, and click **Span Factor Range**.

8

On the **Graphical Editor** tab, click Move Points , then click and drag the right point (where  $\text{measrpm} = 2200$ ) down to  $\text{basefuelmass} = 175$ . You can also enter the values in the **measrpm** and **basefuelmass** edit boxes, or in the table on the **Table Editor** tab.

- 9 Click to select the left point. Make sure the values are 1600 in the **measrpm** edit box and 200 in the **basefuelmass** edit box. The dialog should look like the following example.



This constraint defines the range of **basefuelmass** in terms of RPM to within these bounds: maximum 200 at 1600 rpm, 175 at 2200 rpm.

- 10 Click **OK** to return to the Constraints Manager.
- 11 In the Constraints Manager, click **Duplicate** four times. This saves you setting up tables with only two points for the next constraints. Click to select the first new constraint, then click **Edit**.

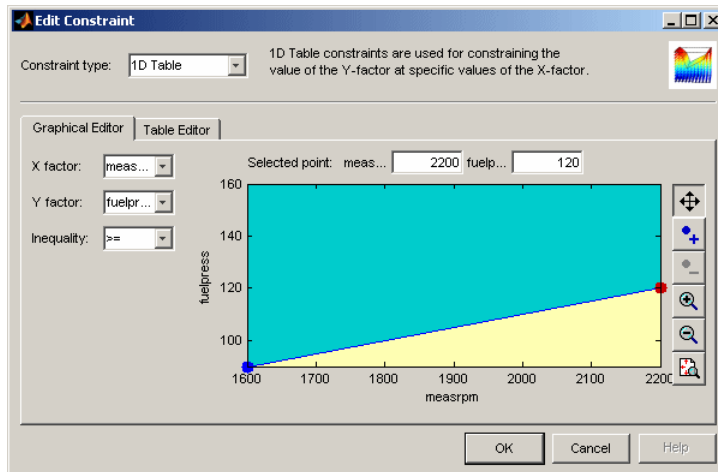
You need to add constraints that define each of the following:

## fuelpress

- Range 90 - 110 at 1600 rpm
- Range 120 - 160 at 2200 rpm

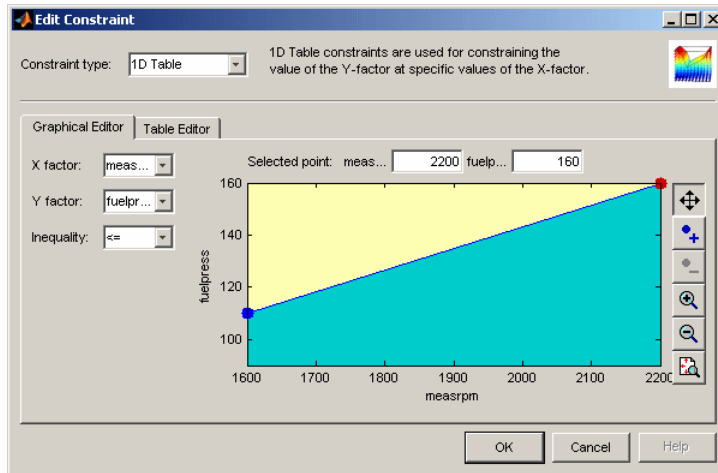
You achieve this by defining two constraints. In the first, the two table points define a **fuelpress** minimum of 90 at 1600 rpm and a minimum of 120 at 2200 rpm. In the second, the two table points define a **fuelpress** maximum of 110 at 1600 rpm and a maximum of 160 at 2200 rpm.

- 12** In the Edit Constraint dialog, change the Y factor to **fuelpress** and leave the X factor as **measrpm**.
- 13** Change the Inequality to  $\geq$ .
- 14** Select the left point (where **measrpm** = 1600) and enter **90** in the **fuelpress** edit box.
- 15** Select the right point (where **measrpm** = 2200) and enter **120** in the **fuelpress** edit box. The dialog should look like the following.



Click **OK** to return to the Constraint Manager.

- 16** Select the next constraint and click **Edit**. Edit the constraint to define a **fuelpress** maximum of 110 at 1600 rpm and a maximum of 160 at 2200 rpm, as shown.



Click **OK** to return to the Constraint Manager.

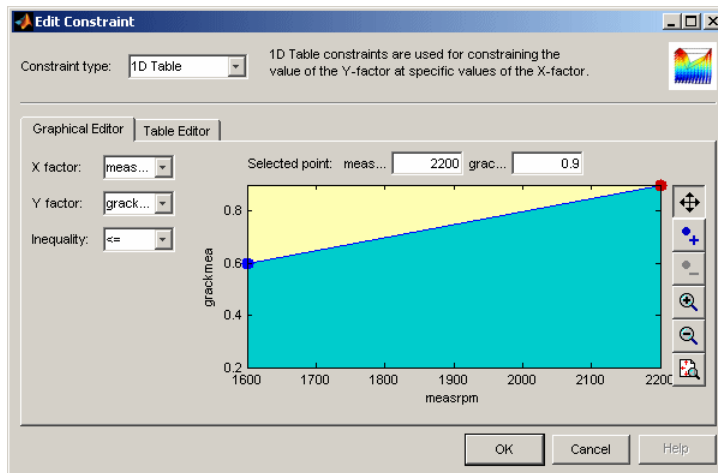
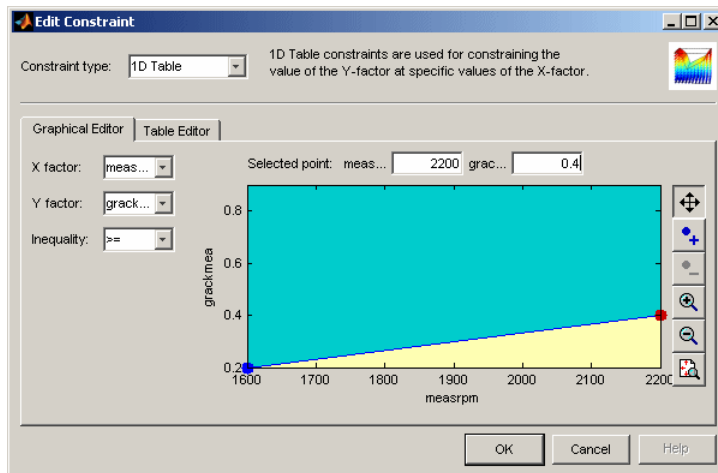
**17** Complete the other constraints in a similar way.

grackmea

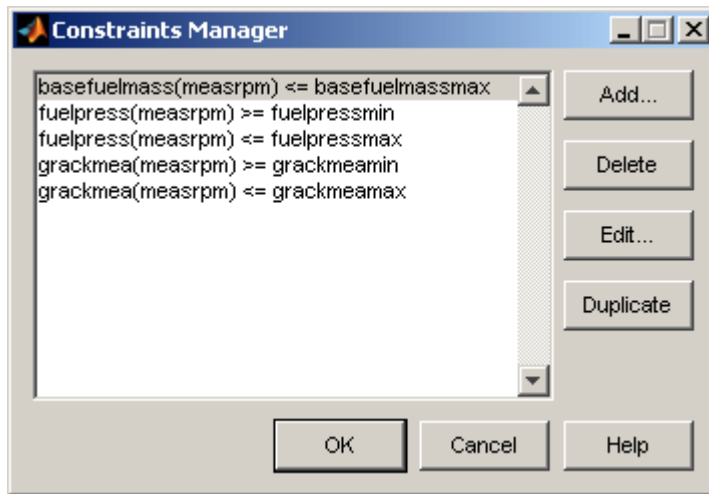
- Range 0.2 - 0.6 at 1600 rpm
- Range 0.4 - 0.9 at 2200 rpm

This is achieved as shown in the following two constraints.



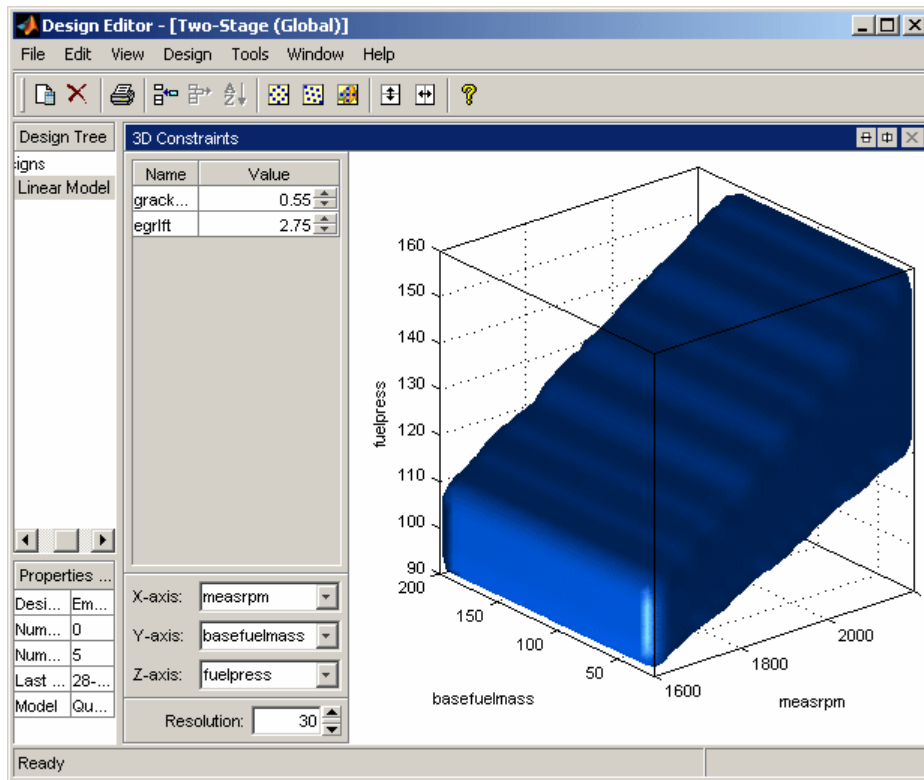


18 The Constraints Manager should contain all five constraints as shown.



Click **OK** to return to the Design Editor.

- 19 Right-click a Design Editor view and select **Current View > 3D Constraints** to view the constrained design space.



## Creating Candidate Designs

Number of points


- How many do you have time for? When you consider the number of points, you need to remember that a sweep will be done at each point, and this will take some time.
- Do you need to allow time to fix problems or redo experimental points that can't be achieved due to combustion stability constraints?

Design type

- V-optimal: reduces average prediction error

V-optimal designs are often the preferred choice for engine testing. Optimal designs tend to push points to the edge, so they should give good coverage of the 1600 and 2200 RPM points while also allowing good modeling of the entire experimental region.

Create an optimal design with 65 points to compare to the example design.

- 1 Click Optimal Design  in the toolbar. The Optimal Design dialog opens.
- 2 Enter 65 in the **Total number of points** edit box.
- 3 Select **V-Optimal** from the **Optimality criteria** drop-down menu and click **OK**.
- 4 The Optimizing Design dialog appears. Click **Accept** when iterations are not producing noticeable improvements; that is, the graph becomes very flat.
- 5 Examine the design points and compare to the constraint space by right-clicking the 3D Constraints view and selecting **Split View > 3D Design Projection**.

The final design used contained 65 points, for a quadratic in fuel pressure and EGR lift. V-optimal value = 0.302.

## Data Collection

Data was generated by a Ricardo WAVE model using the experimental design. MATLAB and Simulink simulation tools control WAVE. Simulation tools support multiple WAVE processes retrieving test points from a central store. Average simulation time was 8 points (30 engine cycles each) per hour using four processors in parallel. Transient test results were then processed to extract steady-state results.

You can use the toolbox to import test data, view it, sort it into tests, verify ranges, filter out unwanted points, and select data for modeling. For details on any of these processes, see the examples in the gasoline case study section “Viewing and Filtering Data” on page 2-19, and for comprehensive information on data handling in the toolbox, see “Data Import and Processing”. See also the examples in “Data Manipulation for Modeling” on page 10-2.

The example project provided (`Diesel_testplan.mat`) contains the filtered data attached to the test plan.

# Modeling

## In this section...

“Overview of Modeling Process” on page 3-15

“Building Models” on page 3-15

“Building and Evaluating Alternative Models” on page 3-18

“Creating Boundary Models” on page 3-24

“Export to CAGE” on page 3-27

## Overview of Modeling Process

Outline of modeling steps you will cover in this example:

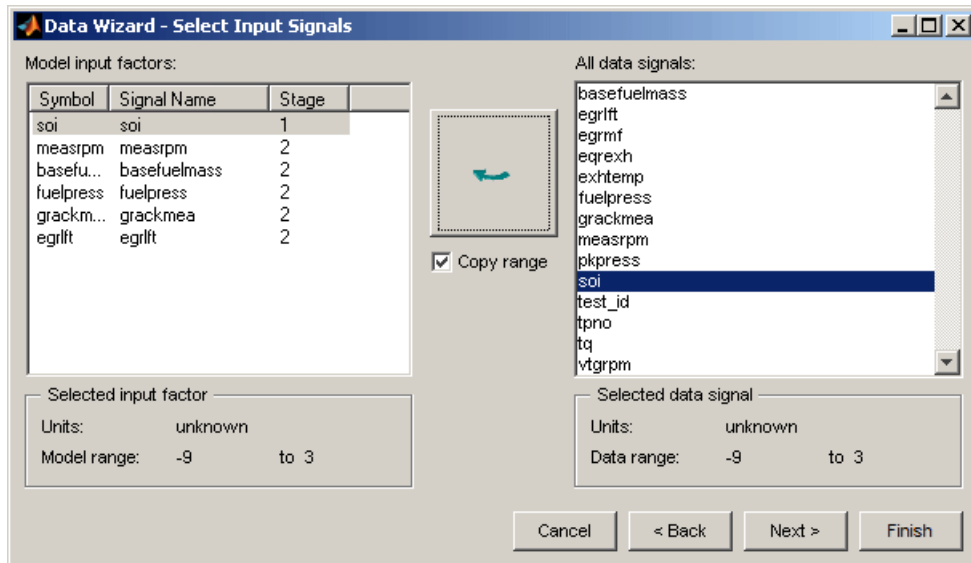
- 1 Build the models specified in the test plan.
- 2 Review local fits. Is the local model flexible enough?
- 3 Eliminate outliers.
- 4 Try alternative local models.
- 5 Review global fits. Is the global model flexible enough?
- 6 Eliminate outliers.
- 7 Try alternative global models.
- 8 Select best models.
- 9 Clean up tree.
- 10 Create boundary models.
- 11 Export models to file or CAGE.

See “Introduction to Two-Stage Modeling” on page 2-4 and “How Is a Two-Stage Model Constructed?” on page 2-26 for information about two-stage models (in the Help Browser you can right-click and select **Back** to return to this page).

## Building Models

The file `Diesel_testplan.mat` contains a test plan where the model inputs and types are set up, and the data is loaded. You have set up the input ranges before designing experiments. Now you can build models of the responses.

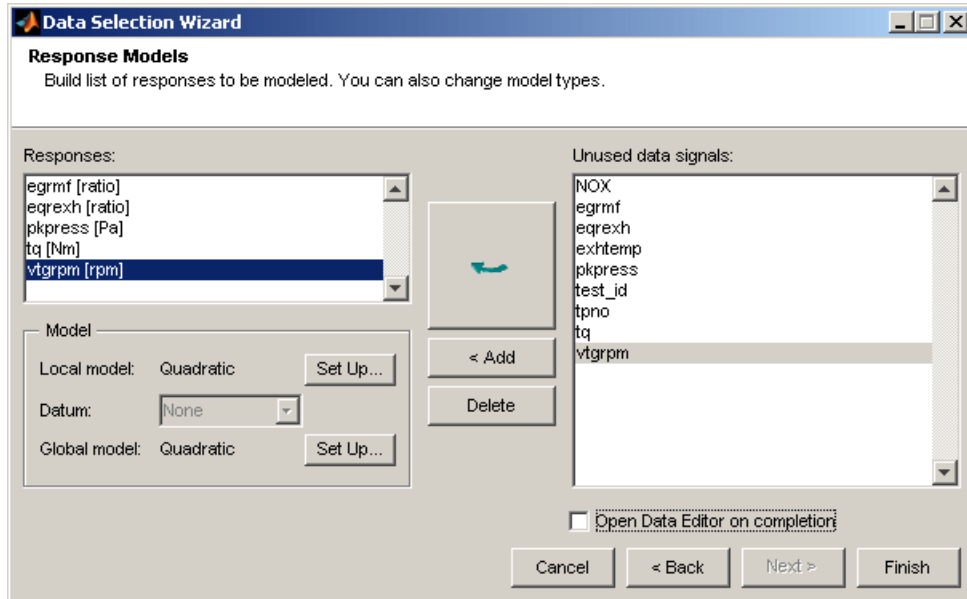
- 1 Double-click the Responses block to select data signals for modeling and build response models.
- 2 The Data Wizard appears. You want to use all selected data, so click **Next** on the Select Data screen.
- 3 Select the check box to **Copy range** of the signals, then match up model input names with data signal names one pair at a time and click the button to associate each pair. As the input signal names are set up to match the data signal names, each model input factor you select is matched automatically in the data signals list, so you can then click the button to select the data signal and copy the range. If the names did not match you would have to select data signals manually.



Click **Next**.

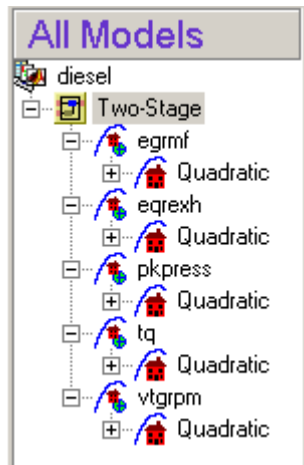
- 4 You can build one response at a time or set up several at once on the **Select Responses** screen of the Data Wizard. Click **Add** after selecting each of these responses:
  - egrmf
  - eqrexh
  - pkpress

- tq
- vtgrpm



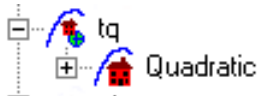
- 5 Clear the check box **Open Data Editor on completion**. If you left this check box selected you could choose to review the data selected for modeling in the Data Editor. Click **Finish**.

The models appear in the model tree in the All Models pane of the Model Browser.



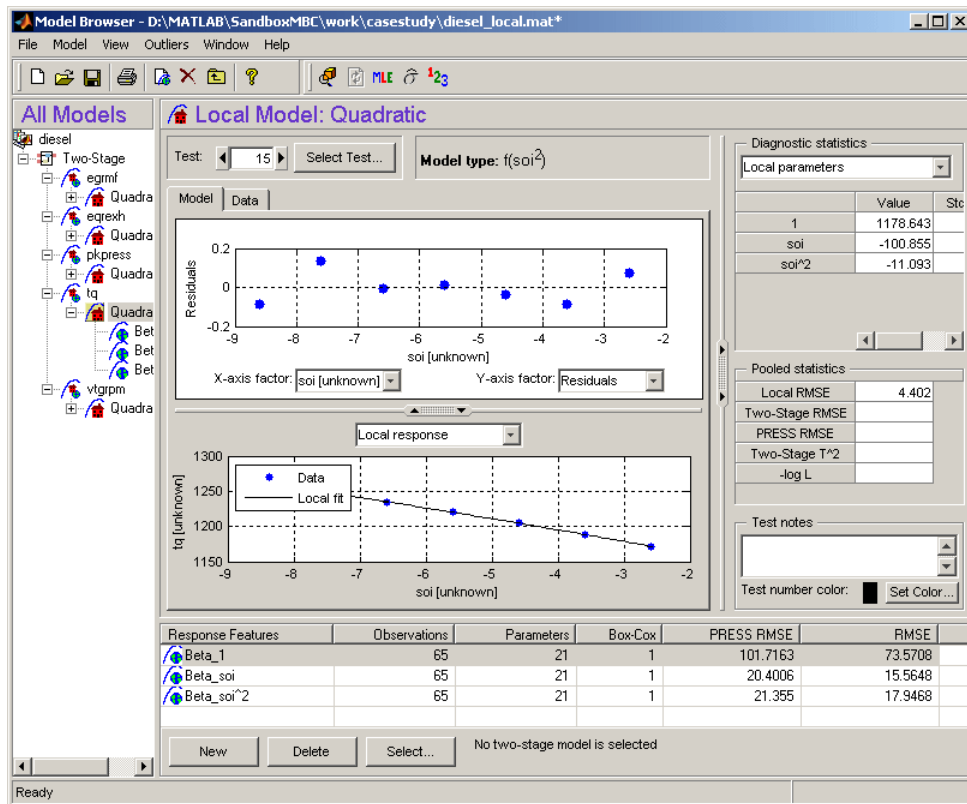
## Building and Evaluating Alternative Models

- 1 Review the local fits. Start by selecting the local model node (Quadratic) under the tq response.

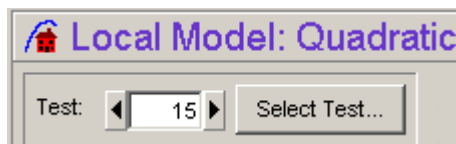


The local model view appears.






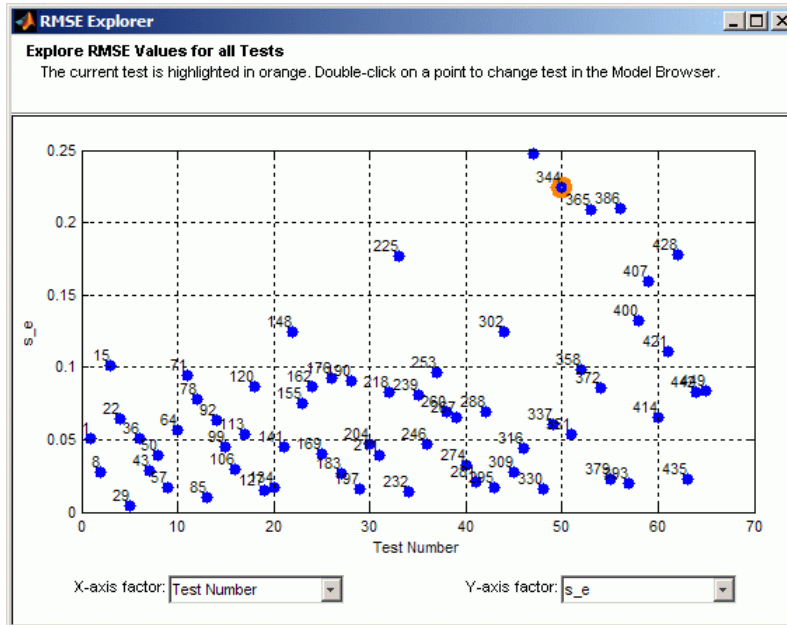
- Look through the tests to inspect the fits. Use the **Test** controls.



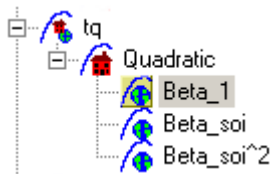
Look for trends in the residuals in the upper plot. Evenly distributed residuals are a good sign; trends in the residuals can indicate that the model is not flexible enough and you should try more flexible model types. A trend in magnitude of residuals indicates that transform or covariance modeling might be needed.

3

To quickly identify problem tests with high errors, click RMSE Plots  in the toolbar (or **View** menu). Double-click tests with high error values in the RMSE Explorer to inspect them in the local model view. Strongly outlying residuals should be investigated.



- 4 Consider removing outliers to improve fits if some points are badly distorting the fit. Engineering judgment is required to judge whether suspect data should be removed. Be careful not to remove outliers without good reason. If you keep removing points you can always get a better fit, but your aim is to achieve a model that predicts the data well. You can click the **Data** tab to inspect data variables for suspect tests.
- 5 Use the same principles to review the global fits. For example, expand the local model node (Quadratic) in the model tree and click **Beta\_1**.




You can right-click outliers (or any point) in the plots to see a plot of the test and inspect the variables. For information on how the local and global models relate to each other, see “How Is a Two-Stage Model Constructed?” on page 2-26





- 6 At both local and global level, create alternative model types to compare.
  - a From the response node, create alternative local models to compare.
  - b From the local nodes, create alternative response feature models to compare.
  - c From the global nodes, create alternative child global models to compare.

To create models one at a time, click **New** from the response node to create a new local model node, or from the global node to create new child global model nodes, etc. Choose different model types from the Model Setup dialog, or you can change any existing model by selecting **Model > Set Up**. There are step-by-step instructions for doing this in the gasoline case study, see “Create Multiple Models to Compare” on page 2-40. You could also work through the modeling tutorial for more guided examples of how to select models, see “Empirical Engine Modelling” on page 8-2.

When you have built a selection of different model types as child nodes of, say, a global model node, you can click the parent model node and select **Model > Make Template**. Save the template to a suitable directory, then you can use **Build**

**Models** (  in the toolbar) to automatically build the same selection of child model types for any other model.


Remember that from any parent node, you can see a list of statistical comparisons for all the child nodes in the lower list pane, along with information such as the number of parameters.

| Models   | Observatio... | Parameters | Box-Cox | PRESS RM... | RMSE    | AICc      | R <sup>2</sup> adj |
|--|---------------|------------|---------|-------------|---------|-----------|--------------------|
|  Quadratic-RBF(1) | 65            | 46         | 1       | 17.4758     | 2.6692  | 406.3691  | 0.99997            |
|  Linear           | 65            | 6          | 1       | 99.2332     | 92.7221 | 598.5186  | 0.96631            |
|  Quadratic        | 65            | 21         | 1       | 101.6145    | 73.5117 | 601.4002  | 0.97883            |
|  Cubic            | 65            | 56         | 1       | 171.9776    | 41.7926 | 1415.3095 | 0.99316            |

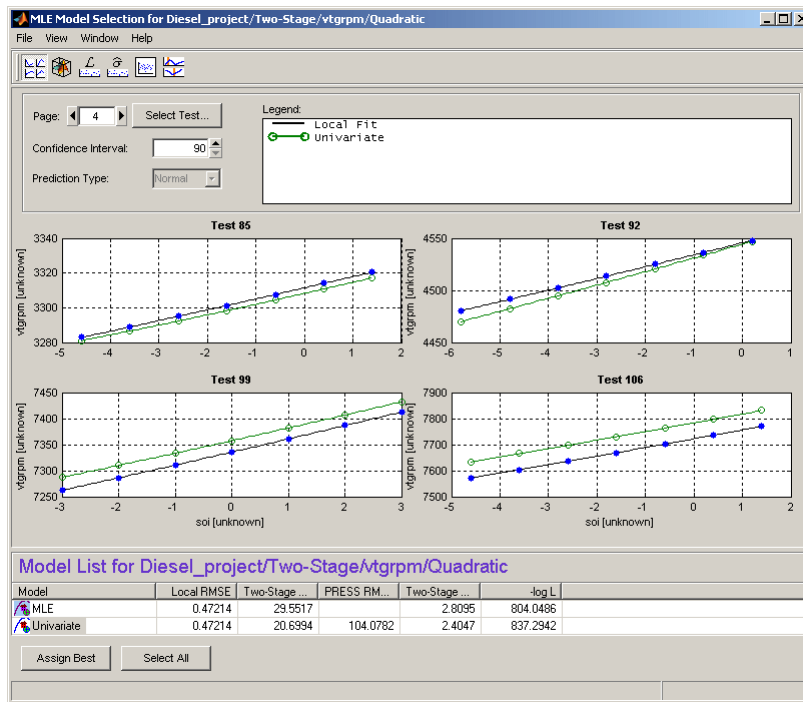
New Delete Select... No best model is selected

Review the fits graphically as well. Search for the best fit but be careful not to overfit. You can increase the number of terms in a model until every point is exactly on the line, but the predictive power of that model will be very low. PRESS RMSE can be the most helpful single statistic you can use to search for the best fit relative

to the number of terms in the model. However, you should not rely on any single statistic, but use a variety of criteria and especially the graphical tools available for comparison of models in the Model Evaluation tool when you click **Select**. For detailed guidance see “Create Multiple Models to Compare” on page 8-29.

Make use of the Stepwise tool ( in the toolbar for linear models) to automatically search for a good fit with the minimum number of useful model terms. You can set Stepwise to run automatically when you create models (for example, select **Min PRESS** from the **Stepwise** drop-down menu in the Model Setup dialog) or you can open the Stepwise window after the model is built. Remember that modeling is a tradeoff — too few parameters means the shape of the surface cannot be captured, while too many parameters gives a risk of overfitting.

- 7 Create two-stage models by clicking **Select** at the Local node. You must first select a best model for each response feature (global) model, if you have created alternatives. If you have created alternative response feature models, when you click **Select** you can choose the best combination of response features in the Model Selection window. There are many graphical tools available here, such as surface plots, contour plots, and movies.



- 8 Try MLE (Maximum Likelihood Estimation). You can choose to calculate MLE in the dialog that appears immediately after building a two-stage model, or you can click **Cancel** and choose to try MLE later. You can use the MLE toolbar button to calculate an MLE model any time. This process refits, taking proper account of the correlation between different response features. Once you have an MLE model, you can click **Select** from the local node to compare it with the univariate two-stage model. You can choose the univariate model as best here if you want; this is the way to "go back" from MLE.
- 9 Searching for a good fit often results in large numbers of models. To discard all but the models you chose as best, select **File > Clean-up Tree**.


For guidance, look at the models in the example finished project, `Diesel_project.mat`, found in the `mbctraining` directory.

## Creating Boundary Models

You can create a boundary model at the test plan node. A model describing the limits of the operating envelope can be useful when you are evaluating optimization results.

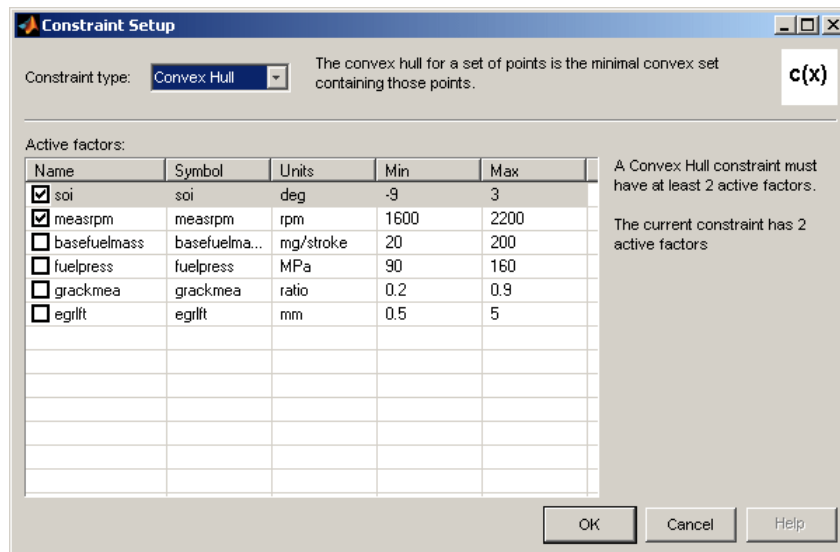
- 1 Select the test plan node in the model tree.
- 2 Select **TestPlan > Boundary Constraints**.

The Boundary Constraint Editor opens.

- 3 Click Make Boundary Constraint  in the toolbar. A dialog box opens where you can choose to build a boundary model of the response, local, or global inputs. First, make a model of the response boundary. Select **Response**, and click **OK**.

A dialog box opens where you can select constraint inputs.

- 4 Set the **Constraint type** to **Convex Hull**. Leave only the **soi** and **measrpm** check boxes selected, as shown in the following figure.



Click **OK**, and then click **OK** again in the next dialog box because there are no other parameters to set for this constraint.

A Convex Hull(soi, measrpm) child node appears under the Response node.

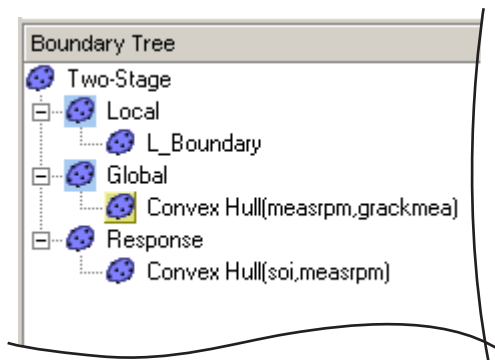
- 5 Select **View > Current View > Pairwise Projections** (or use the toolbar button). You can see pairwise projections to view the boundary across all combinations of factors.
- 6 Next, make a model of the global boundary. There is already a **Global** node with an empty child node under it. Select this node, **G\_Boundary**, then select **Constraint > Set Up**.

A dialog box opens where you can select constraint inputs.

- 7 Set the **Constraint type** to **Convex Hull**. Leave only the **measrpm** and **grackmea** check boxes selected. Click **OK**.

The name changes to **Convex Hull(measrpm, grackmea)**, but the constraint is not complete yet.

- 8 Select **Constraint > Full Constraint Fit**. Click **OK** in the next dialog box as there are no other parameters to set for this constraint. The constraint model appears in the pairwise view. The tree should look as shown in the following figure.

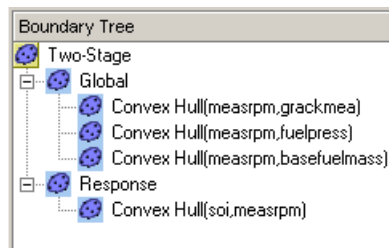


- 9 Click **Duplicate constraint** in the toolbar twice. Two new convex hull child nodes appear under the **Global** node.
- 10 Select the first new node, and select **Constraint > Set Up**. The Constraint Setup dialog box appears.
  - a Leave only the **measrpm** and **fuelpress** check boxes selected. Click **OK**.
  - b Select **Constraint > Full Constraint Fit**. Click **OK** in the next dialog box as there are no other parameters to set for this constraint.
- 11 Select the second new node, and select **Constraint > Set Up**. The Constraint Setup dialog box appears.

- a** Leave only the `measrpm` and `basefuelmass` check boxes selected. Click **OK**.
  - b** Select **Constraint > Full Constraint Fit**. Click **OK** in the next dialog box as there are no other parameters to set for this constraint.
- 12** You can select a combination of models as best. Select each of the **Global** and **Response** models in turn, and select **Constraint > Add to Best** (also in the toolbar). Make sure both the **Response** and **Global** nodes are also added to best.

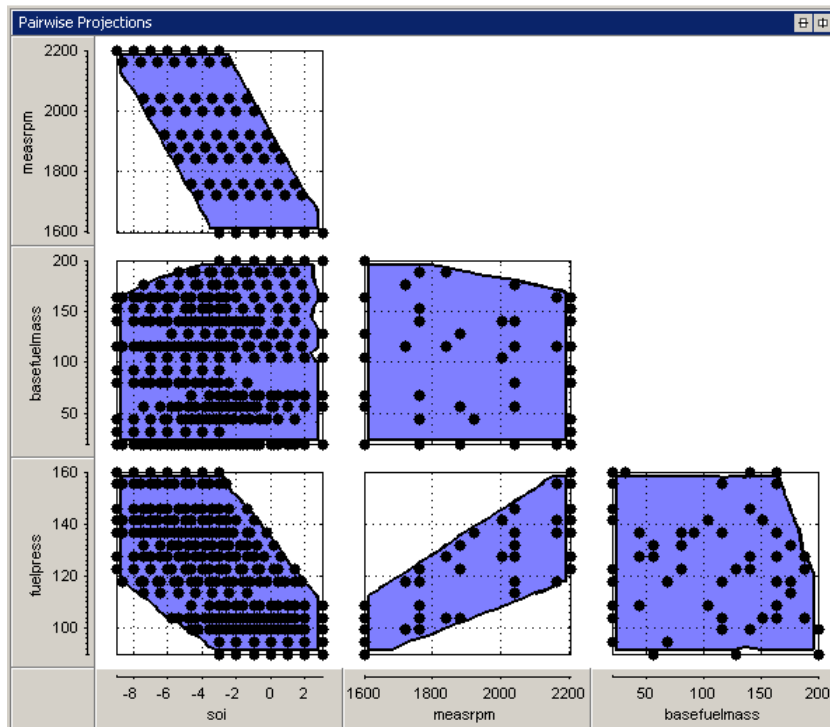
You can delete the **Local** node as you are not interested in the local boundary.

Make sure all remaining icons are blue, indicating they are included in the combination of best models, as shown in the following figure. For more information see “Combining Best Boundary Models”.



- 13** Now, select the root node **Two-Stage** to see the combination of all constraints in the pairwise view, as shown in the following figure.





- 14** Close the Boundary Constraint Editor to return to the Model Browser. After they are calculated, the boundary constraints remain part of the test plan unless you delete them.

## Export to CAGE

You can export your models to file, to CAGE, to the workspace or to Simulink. This capability makes models easy to share across engineering groups. You can do such exports in the Model Browser by selecting **File > Export**. The models exported depend on the model node you have selected in the tree. To export all models in the test plan, select the test plan node.

---

**Note:** For this example, you use CAGE to import the models.

---

You use the example models in the CAGE part of the Model-Based Calibration Toolbox product to produce optimized calibration tables. Proceed to the next section for instructions.

# Optimized Calibration

## In this section...

“Problem Definition” on page 3-29

“Benefits of Automated Calibration” on page 3-30

## Problem Definition

This section describes the creation and optimization of calibration tables for the diesel case study. You use the models created in the Model Browser section of this case study. An example file is provided.

The problem to solve is to produce tables in speed and torque for

|                       |              |
|-----------------------|--------------|
| Best injection timing | soi          |
| Best fuel quantity    | basefuelmass |
| Best fuel pressure    | fuelpress    |
| Best VTG              | grackmea     |
| Best EGR              | egr1ft       |

Also, to minimize brake specific fuel consumption, subject to constraints on

- Turbo speed (vtgrpm)
- Cylinder pressure (pkpress)
- Exhaust equivalence ratio (exheqr)

For this case study, you use models produced in the Model Browser to generate calibrations in CAGE. You cover the following steps:

- 1 Load models of engine responses. See “Importing Models of Engine Responses into CAGE” on page 3-32.
- 2 Define additional models and variables required by optimization strategy. See “Defining Additional Variables and Models” on page 3-33.
- 3 Set up tables. See “Setting Up Calibration Tables to Fill” on page 3-35.
- 4 Set up the optimization to provide feasible starting points for the sum problem. See “Setting Up the Point Optimization” on page 3-36.

- 5 Define constraints. See “Setting Up Constraints” on page 3-39.
- 6 Define optimization operating points. See “Defining Variable Values” on page 3-43.
- 7 Run the optimization and view results. See “Running the Optimization” on page 3-45
- 8 Create a sum optimization using the previous solutions as starting points. See “Setting Up the Sum Optimization” on page 3-47
- 9 Fill tables from optimization results. See “Filling Tables with Optimization Results” on page 3-52.

For guidance, you can look at the example finished project, `Diesel_optimization.cag`.

## Benefits of Automated Calibration

- You can move the table-filling process away from the test bed.
- You can regenerate calibrations when objectives, constraints, or calibration table layouts change, without additional testing.
- You can explore tradeoff possibilities interactively.
- You can produce initial calibrations using engine simulation software, before hardware is available.

CAGE can provide both automatic and interactive calibration optimization. You can trade off multiple objectives, deal with multiple constraints, and you can examine optimizations point-by-point or over a drive-cycle. CAGE can provide solutions for these example applications:

- Example control applications

Emissions-constrained BSFC optimization over drive cycles producing calibrations such as:

- Optimal fuel injection timing schedule
  - Optimal fuel injection quantity schedule
  - Optimal EGR valve position and EGR mass fraction schedule
  - Optimal spark timing schedule
  - Optimal dual-independent variable valve timing schedules
- Estimation problems
    - Torque
    - Emissions
    - Air flow and manifold pressure
    - Intake valve temperature
    - Borderline spark

## Importing Models of Engine Responses into CAGE

- 1 Start CAGE by typing `cage` at the MATLAB command line.
- 2 Select **File > Import > From Project**. The CAGE Import Tool appears.
- 3 To import from a file, click the **Import From Project File** button. Locate the example model file created in the Model Browser, `Diesel_project.mat`, in the `mbctraining` directory, and click **Open**. The import tool loads a list of models in the file.
- 4 Select **Response** from the **Type** drop-down list to filter the model list.
- 5 Select all the response models in the list by **Shift**+clicking.
- 6 Click the button **Import selected items**.
- 7 In the following Import dialog box, click **OK** to import the models.
- 8 Close the Import Tool and observe the Models view in CAGE. You should see the models in the list. The selected model is displayed in the other panes.

You can also:

- Export models directly from the Model Browser to CAGE when both are open.
- Use the CAGE Import Tool to import models directly from the Model Browser.
- Use the CAGE Import Tool to import models and other calibration items (tables, optimizations, tradeoffs, data sets, features) from any project file created in CAGE or the Model Browser. This capability can help you use existing projects to speed up the setup of new sessions.
- Export models to a file, which you can import to CAGE by selecting **File > Import > Model**.

## Defining Additional Variables and Models

Looking at the problem definition, you need to define some additional variables and models for this optimization. You want to minimize BSFC, but this is not a model output. You need to fill tables against speed and torque, but torque is not an input for the models. The solution to this is to create function models. You can then use a torque constraint and the BSFC function model in the optimization. To implement the torque constraint, you need to define a new variable for desired torque, `tq_desired`.

Follow these steps:

- 1 Select **File > New > Variable Item > Variable**. The Variable Dictionary view appears.
  - a Rename the new variable `tq_desired`.
  - b Set the range of this variable to be minimum 0 and maximum 1500.
  - c Edit the set point to 600 Nm
- 2 Create a new function model.
  - a Copy the following function model definition, ready to paste into the Function Model Wizard:
 
$$\text{bsfc} = 5400 / \text{pi} * \text{basefuelmass} / \text{tq}$$
  - b Select **File > New > Function Model**.
  - c Paste or enter `bsfc = 5400 / pi * basefuelmass / tq` and click **Next**.
 

Assuming: base fuel mass [mg/inj], Tq [Nm], 6 cylinders, 4 stroke
  - d Select **Automatically assign/create inputs**. It is important to check for typos or this step can create unintended new inputs. Click **Finish**.
 

The Models view appears.
- 3 This optimization requires a constraint on the air/fuel ratio (AFR). However, as you could not model AFR directly you need to create a function model that relates equivalence ratio to air/fuel ratio. Then you can constrain AFR. To do this, create another new function model.
  - a Copy the following function model definition, ready to paste into the Function Model Wizard:

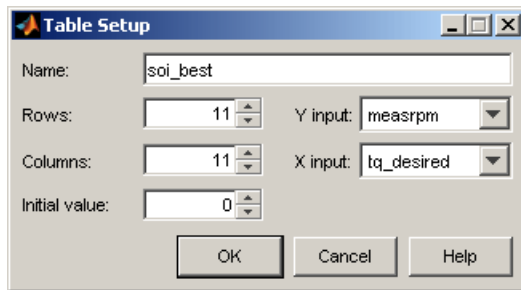
$\text{afr} = 14.46/\text{eqrexh}$

- b** Select **File > New > Function Model**.
  - c** Paste or enter  $\text{afr} = 14.46/\text{eqrexh}$  and click **Next**.
  - d** Select **Automatically assign/create inputs** and click **Finish**.
- 4** Select **File > New > Variable Item > Variable**. The Variable Dictionary view appears.
- a** Rename the new variable `afr_min`.
  - b** Set the range of this variable to be minimum 0 and maximum 100
  - c** Edit the set point to 50



## Setting Up Calibration Tables to Fill

- 1 Set up a new 2-D table. Select **File > New > 2D Table**.
- 2 Name the table `soi_best`.
- 3 Select `measrpm` and `tq_desired` from the **Y Input** and **X Input** drop-down menus.
- 4 Enter 11 rows and 11 columns.
- 5 Enter 0 for the initial value.



- 6 Click **OK** to create the table.

Look at the **Tables** tree on the left. Note that the normalizers appear as calibratable items in their own right, and as descendants (child nodes) of their tables.

- 7 Once you create a table, you can duplicate it to create more tables that share the same breakpoints. Click `soi_best` in the tree, then select **Edit > Duplicate soi\_best**. Repeat until you have four new tables, and rename them as follows:
  - a `basefuelmass_best`
  - b `fuelpress_best`
  - c `grackmea_best`
  - d `egr1ft_best`

## Setting Up the Point Optimization

CAGE provides a flexible optimization environment. You can define the objectives, the constraints, and the points where the optimization is carried out.

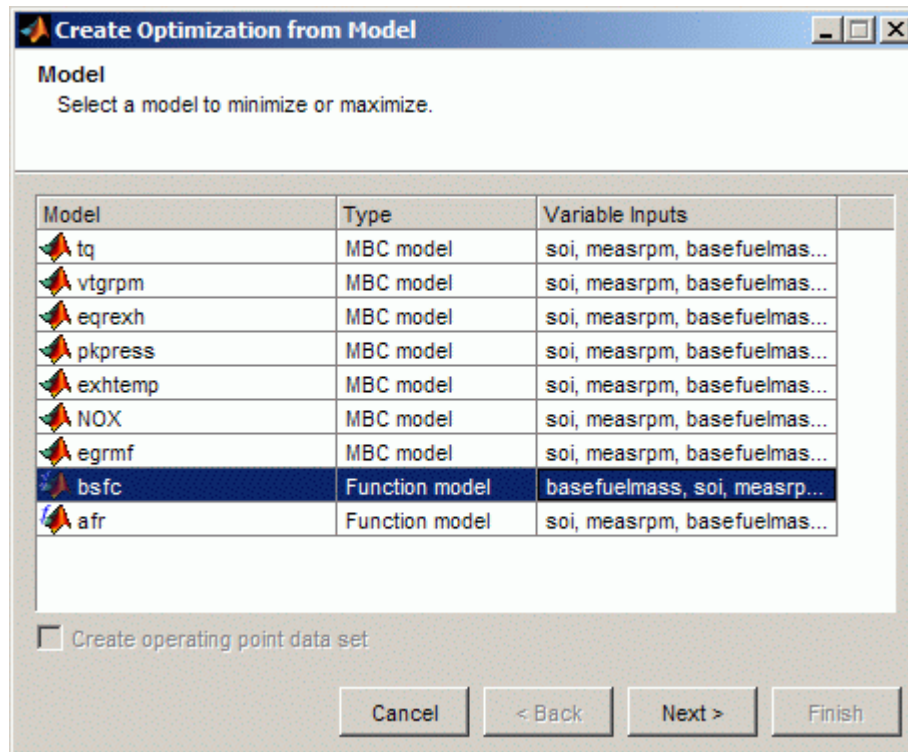
The objective is to minimize weighted BSFC over a set of points, subject to a set of constraints. This is a constrained single objective optimization problem.

You solve this problem in two parts:

- 1 Run an initial point optimization to explore the problem point-by-point in order to obtain good starting points for the sum optimization. You will create the optimization, set up constraints and values and run the optimization.
- 2 Next you can use the solutions from the initial optimization as the start points to run the sum optimization over the drive cycle to find optimal control settings. Use these results to fill calibration tables.

CAGE has several built-in optimization routines and the capacity for you to write your own. In this case study, you use `foptcon`. This algorithm is a modified version of `fmincon` from the Optimization Toolbox product. In CAGE, you can use the algorithm to minimize or maximize an objective function.

- 1 Select **Tools > Create Optimization from Model**, or click the toolbar button. The Create Optimization from Model Wizard appears.
- 2 Select the `bsfc` model to minimize in the optimization.

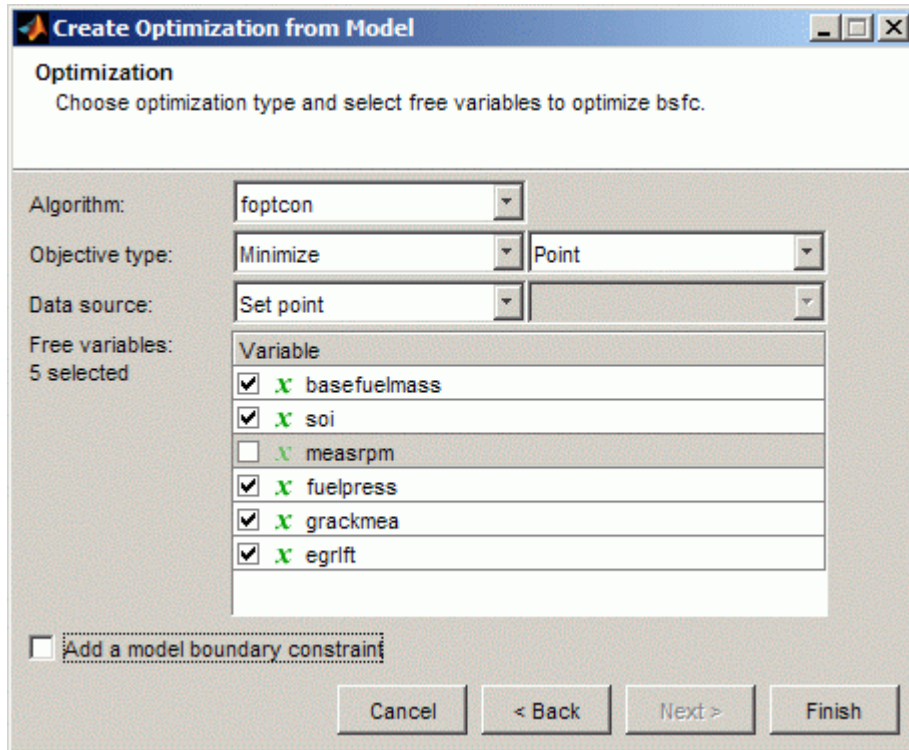


Click **Next**.

**3** Select the optimization settings as follows:

- **Free variables:** Clear the check box for `measrpm`. Leave the selected check boxes of the other free variables. These are the variables you want CAGE to optimize from the set of model inputs.
- **Add a model boundary constraint:** Clear the check box. You add another boundary constraint later in the Optimization view.
- Leave the defaults for all the other settings:
  - **Algorithm:** Use the default `foptcon` for gradient-based single-objective optimizations.

- **Objective type:** Leave the default set to Minimize your model, and leave the default Point objective type.
- **Data Source:** Leave the default Set point. You specify points later in the Optimization view.
- Click **Finish** to create the optimization.



CAGE displays the **Optimization** view with your new optimization.

You have not yet set up your constraints, so you can add them next.

## Setting Up Constraints

This case study problem has CAGE model constraints on the following quantities (which you will set up next):

- tq\_desired
- vtgrpm
- pkpress
- afr
- soi
- grackmea
- fuelpress
- basefuelmass

In the Optimization view, the constraints you will set up next will be shown as follows.

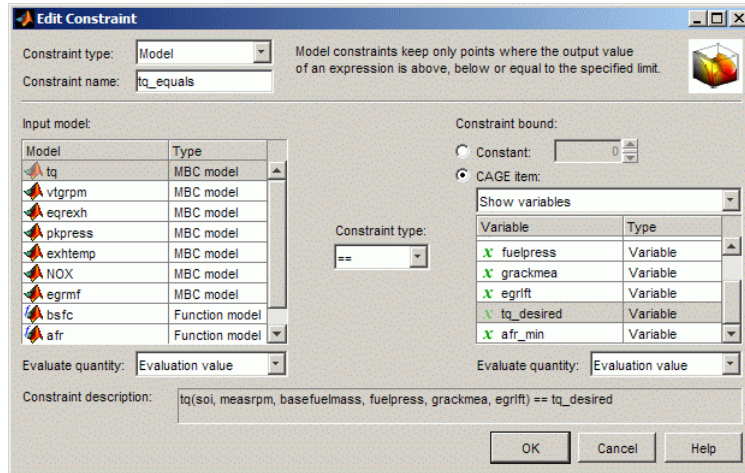
| Constraints |  |
|-------------|--|
| Name        | Description  |
| tq_equals   | tq(soi, measrpm, basefuelmass, fuelpress, grackmea, egrift) == tq_desired          |
| afr         | afr(soi, measrpm, basefuelmass, fuelpress, grackmea, egrift) >= afr_min            |
| vtgrpm      | vtgrpm(soi, measrpm, basefuelmass, fuelpress, grackmea, egrift) <= 128000          |
| pkpress     | pkpress(soi, measrpm, basefuelmass, fuelpress, grackmea, egrift) <= 18000000       |
| tq_Boundary | Boundary constraint of tq(soi, measrpm, basefuelmass, fuelpress, grackmea, egrift) |

You can edit constraint names to aid analysis in other Optimization views (right-click, or double-click to use the Edit Constraint dialog box).

Set up these constraints as follows:

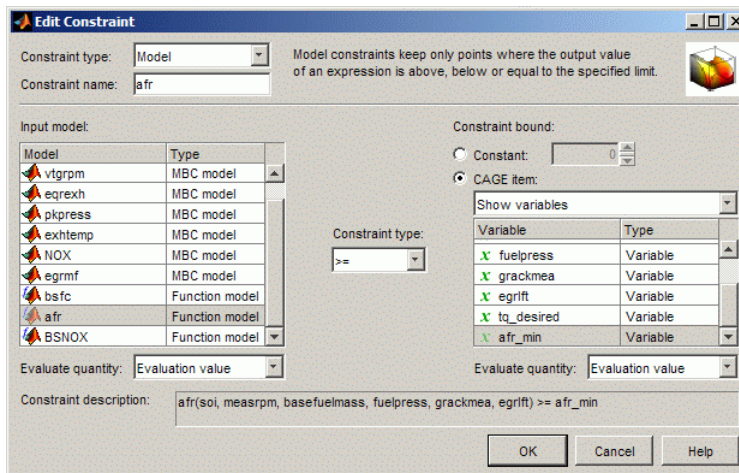
- 1 Right-click in the Constraints pane and select **Add Constraint**. The Constraint Editor appears. Set up the first constraint to `tq == tq_desired` as follows.
  - a Leave the Constraint type as **Model**.
  - b Edit the Constraint name to `tq_equals`.
  - c Select `tq` in the **Input model** list.
  - d Select `==` as the **Constraint type**.
  - e Select the **CAGE item** radio button.
  - f Select **Show variables** from the drop-down list.

- g Select `tq_desired` in the variable list. The dialog should look as shown in the following figure.

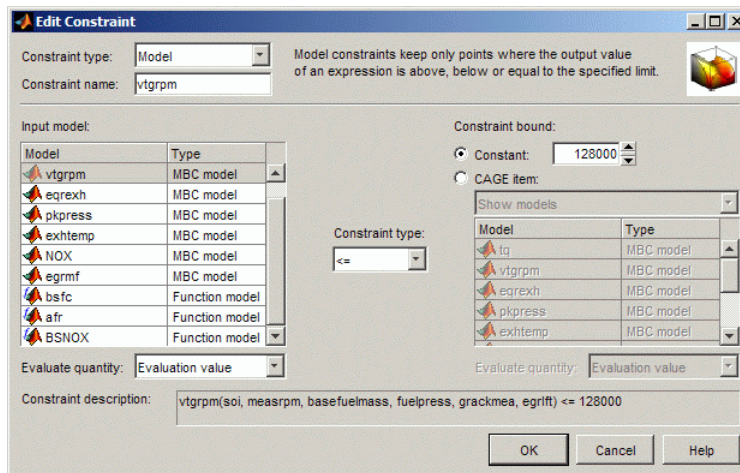


- h Click **OK** to return to the Optimization view.

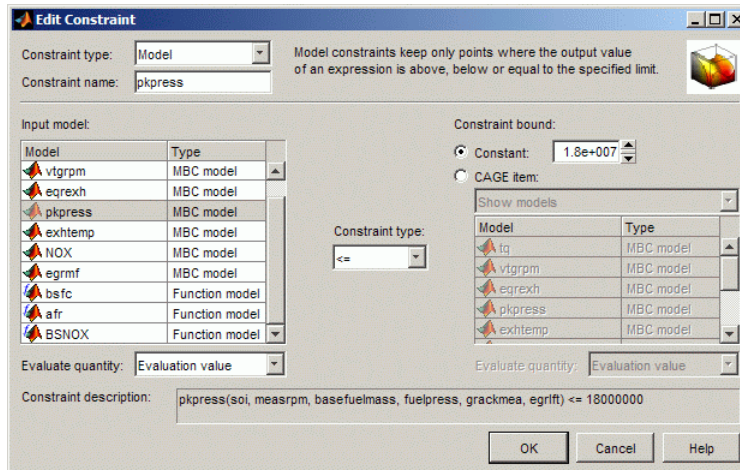
- 2 Similarly, right-click to add the next constraint as shown: `afr >= afr_min`.



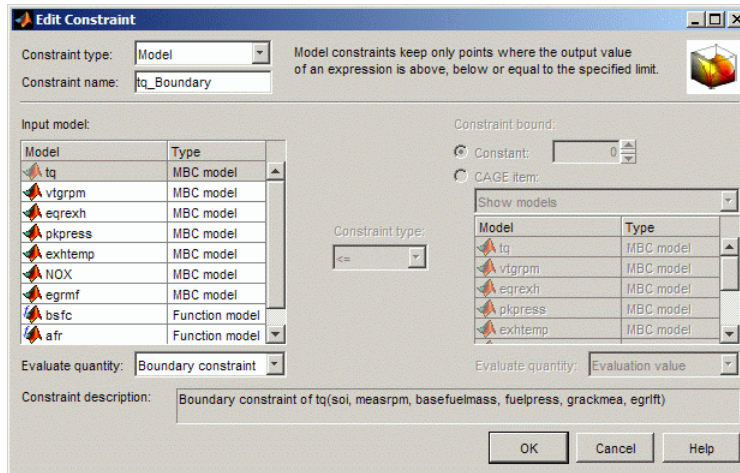
- 3 Right-click again and add the next constraint as shown: `vtgrpm <= 128000`. To do so, enter a value in the **Constant** edit box, rather than select a variable from the CAGE item list.



- 4 Right-click again to set up the constraint `pkpress <= 18000000`, as shown. This time, enter a value in the **Constant** edit box.



- 5 Repeat the action to set up a boundary model constraint, as shown in the following figure.



- a Leave the Constraint type as Model.
- b Edit the Constraint name to tq\_Boundary.
- c Select tq in the **Input model** list.
- d Select Boundary constraint in the **Evaluate quantity** drop-down list.
- e Click **OK** to create the constraint and return to the Optimization view.



## Defining Variable Values

You need to define the set of drive cycle points where you want the optimization to run. To do this you use the **Input Variable Values** pane in the Optimization view.

- 1 Increase the **Number of runs** to 7. Click the buttons or enter the value in the box. The number of rows in the fixed and free variables panes increases to 7. The default values are the set point of each variable. Leave the initial values for the free variables at the defaults.
- 2 You can enter or copy values into the **Fixed Variables** pane to define the fixed variable values at each point where you want the optimization to run. You can copy all the variable values from a text file, or from the Help Browser copy each column in turn. Copy and paste the following values into the `measrpm` column in the **Fixed Variables** pane.

```
measrpm
2200
2200
2200
2200
1600
1600
1600
```

- 3 Copy and paste the following values into the `tq_desired` column in the **Fixed Variables** pane.


```
tq_desired
1263
947
632
126
1550
1163
775
```

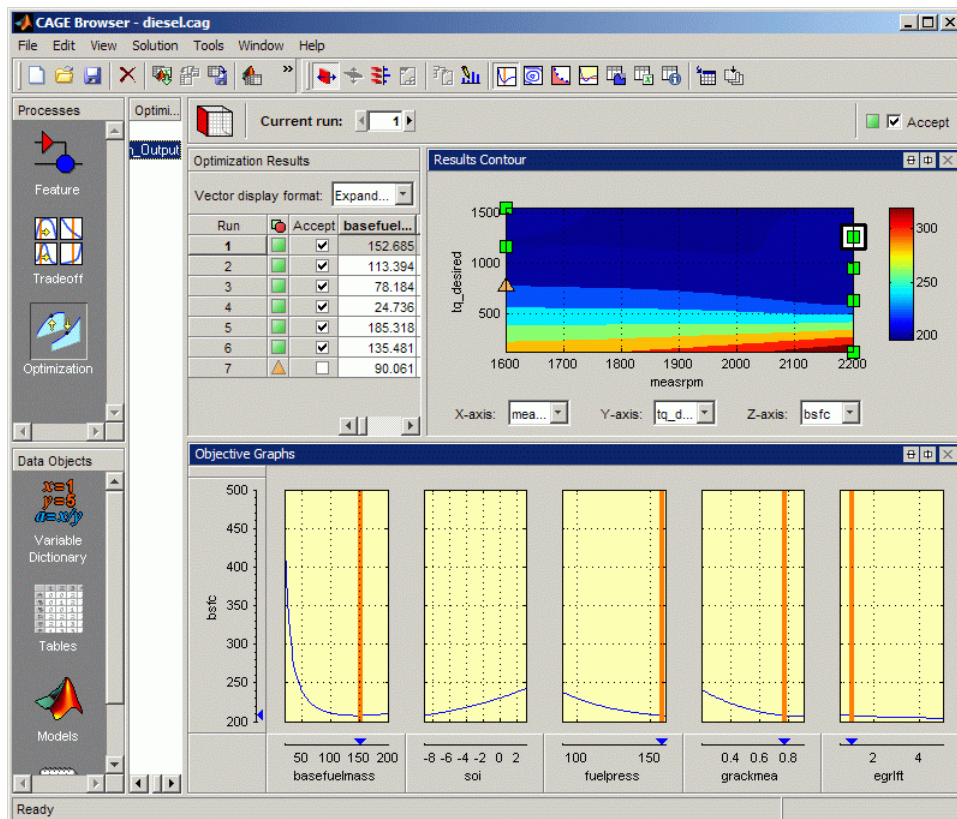
- 4** Copy and paste the following values into the `afr_min` column in the **Fixed Variables** pane.

```
afr_min  
25.5  
27.75  
30.0  
0  
22  
22.5  
23
```

## Running the Optimization

Now you can run the optimization. You have set up objectives, constraints and a set of operating points.

- 1 Click Run Optimization  in the toolbar.
- 2 When the optimization is complete, the view switches to the new child node, `bsfc_Optimization_Output`, in the Optimization tree under the `bsfc_Optimization` node. View the results.



- 3 Look through the solutions at different operating points by clicking cells in the output table. Compare with the gasoline example, where regions that do not meet the constraint are yellow in the graphs — in this case the torque equality constraint

causes all the graphs to be yellow, because this constraint produces a feasible contour in the free variable space, and any deviation off this contour is infeasible.

- 4 Split the view to display the constraint graphs, and scroll through them to see each constraint in relation to the solutions.

## Setting Up the Sum Optimization

### In this section...

“Setting Up Initial Values and Sum Objective” on page 3-47

“Creating the Brake Specific NOx Constraint” on page 3-48

“Setting Weights for the Sum Objective and Constraint” on page 3-49

“Set Parameters and Run Optimization” on page 3-51

### Setting Up Initial Values and Sum Objective

Previously, you created the initial point-by-point optimization to obtain good starting points for the sum optimization. Now, you can create the sum problem from that initial optimization. You then use the solutions from the initial optimization as the start points to run the sum optimization over the drive cycle to find optimal settings. Then, you can fill the calibration tables with these results.

- From the `bsfc_Optimization_Output` node of your first optimization, select **Solution > Create Sum Optimization**.

CAGE creates a new optimization called `Sum_bsfc_Optimization`. The optimization has these characteristics:

- The objective matches your original optimization but converted to a sum objective.
- The new optimization has identical constraints to your original optimization.
- Your original fixed and free variables have converted to a sum optimization (a single run with multiple values).
  - The new optimization uses only accepted solutions from your original optimization output (all runs with a selected **Accept** check box).
  - Therefore, the number of accepted solutions you had in the original optimization determines the number of values within the sum optimization run.
- The new optimization imports the free variable initial values and fixed variable values from your point optimal results (accepted solutions only).
- The fixed variables have a **Weights** column with each value set to 1.

In the Optimization view, observe the new sum objective and the new values in the Input Variable Values pane.

Now your sum optimization is ready to run. Before you run it, however, see the next section to create a brake specific NOx constraint.

## Creating the Brake Specific NOx Constraint

In this optimization, you will use brake specific NOx to constrain the results.

BSNOx is defined as:

$$BSNOx = \frac{\sum_{i=1}^N w_i NOx_i}{2\pi/60000 \sum_{i=1}^N w_i N_i TQ_i}$$

You implement this constraint as a weighted sum constraint in CAGE by creating a function model that represents the following term of the BSNOx sum:

$$BSNOx\_term = \frac{NOx}{2\pi/60000 \sum_{i=1}^N w_i N_i TQ_i}$$

The denominator of this term represents the weighted power (speed \* torque) over the drive cycle, which is constant for this optimization. You calculate this constant as follows:

$$\begin{aligned} & 2\pi/60000 \sum_{i=1}^N w_i N_i TQ_i \\ &= 2\pi/60000 \left( \begin{array}{l} 2200 \times 1263 + 2200 \times 947 + 2200 \times 632 + 2200 \times 126 + \\ 1600 \times 1550 + 1600 \times 1163 + 1600 \times 775 \end{array} \right) \\ &= 159.5573 \end{aligned}$$

Therefore, the function model to be created is

$$BSNOx\_term = \frac{NOx}{159.5573}$$

Create the function model as follows:

- 1 Copy the following function model definition, ready to paste into the Function Model Wizard:

$$\text{BSNOX} = 3600 * 1000 * \text{NOX} / 159.5573$$

- 2 Select **File > New > Function Model**.
- 3 Paste or enter  $\text{BSNOX} = 3600 * 1000 * \text{NOX} / 159.5573$  and click **Next**.

The NOx model is in kilograms per hour, and you want to calculate BSNOx in grams per second, so multiply NOx by 3600\*1000.

- 4 Select **Automatically assign/create inputs**. It is important to check for typos or this step can create unintended new inputs. Click **Finish**.

The Models view appears with the new function model. Now, you can use this function model to create a weighted sum constraint.

## Setting Weights for the Sum Objective and Constraint

To create a weighted sum constraint:

- 1 Click the Optimization button to return to the Optimization view.
- 2 Select your sum optimization.
- 3 Right-click in the Constraints pane, and select **Add Constraint**. The Edit Constraint dialog box appears.

**Edit Constraint**

Constraint type:  A sum constraint provides a constraint distance value for a weighted sum of a model with reference to a bound

Constraint name:

| Constraint | Type           |
|------------|----------------|
| !tq        | MBC model      |
| !vtgrpm    | MBC model      |
| !eqrexh    | MBC model      |
| !pkpress   | MBC model      |
| !exhtemp   | MBC model      |
| !NOX       | MBC model      |
| !egrmf     | MBC model      |
| !bsfc      | Function model |
| !afr       | Function model |
| !BSNOX     | Function model |

Constraint type:  Constraint bound:

Constraint description:

OK Cancel Help

- 4 Select **Sum Constraint** from the **Constraint type** drop-down menu.
- 5 Enter **BSNOX** in the **Constraint name** edit box.
- 6 Select **BSNOX** from the **Input model** list.
- 7 Enter **3** in the **Constraint bound** edit box, and press **Enter** to create the constraint, and return to the Optimization view. Verify that the new constraint description shows **Weighted sum of BSNOX <= 3**, to constrain the weighted sum to less than 3g/kWh.

Set the weights in the Optimization view as follows:

- 1 Enter **7** in the **Number of values** edit box for the `bsfc_weights` column.
- 2 Enter, or copy and paste, the following values into the `bsfc_weights` column in the **Fixed Variables** pane.

#### Objective 1 Weights

0.5  
0.15  
0.15  
0.05  
0.05  
0.05  
0.05

These values sum to 1.

- 3 Similarly, enter **7** in the **Number of values** edit box for the `BSNOX_weights` column.
- 4 Copy the following values into the `BSNOX_weights` column in the **Fixed Variables** pane.

#### Constraint 1 Weights

0.15  
0.15  
0.15  
0.1  
0.1



**Constraint 1 Weights**

0.1

0.1

These values sum to 0.85.

## Set Parameters and Run Optimization

To alter parameters and then run the optimization:

- 1 Click Set Up and Run Optimization in the toolbar.
- 2 Change the following parameters in the Optimization Parameters dialog box:
  - **Maximum iterations** = 500
  - **Maximum function evaluations** = 2000
  - **Variable tolerance** =  $1e-7$
  - **Function tolerance** =  $1e-7$

Click **OK** to close the dialog box, and the optimization runs.

- 3 When the optimization is complete, the view switches to the updated child node, `Sum_bsfc_Optimization_Output`, in the Optimization tree under the `Sum_bsfc_Optimization` node. View the results. For guidance see “Interpreting Sum Optimization Output”.

## Filling Tables with Optimization Results

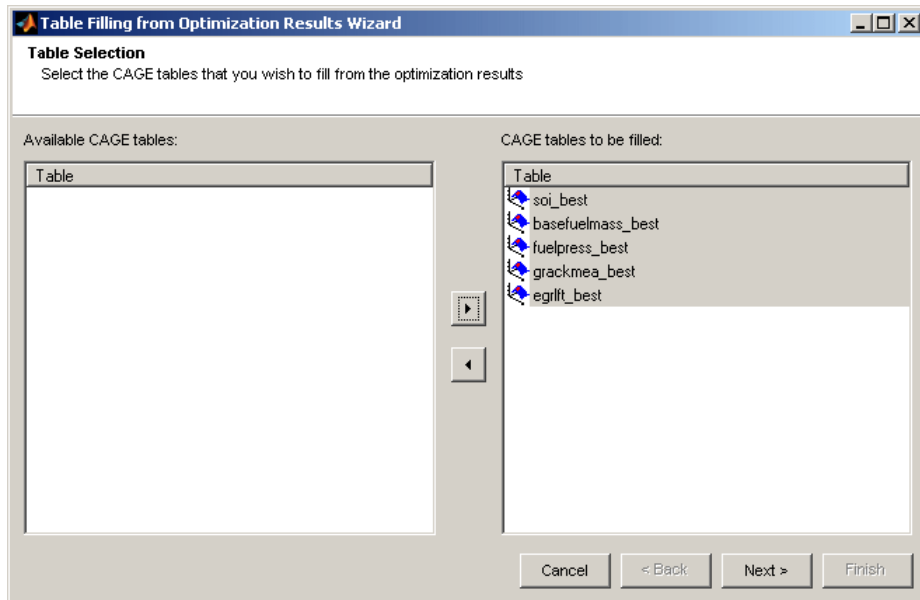
You can use the optimization results to fill the tables you created.

- 1 From the optimization output node, select **Solution > Fill Tables**.

The Table Filling Wizard appears.

- 2 **Shift**-click to multi-select all the following tables and click the button to add all your tables to the filling list:

- soi\_best
- basefuelmass\_best
- fuelpress\_best
- grackmea\_best
- egrlft\_best

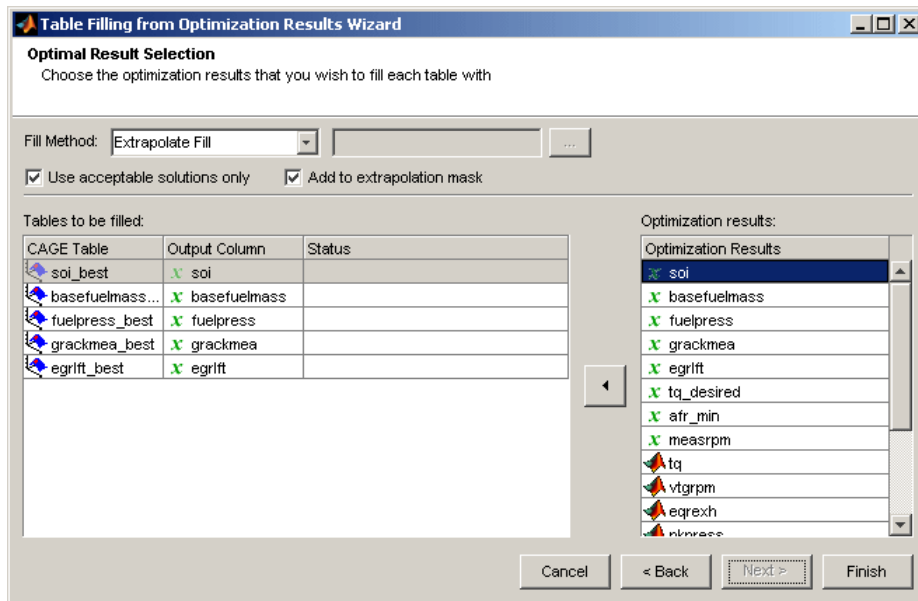


Click **Next**.

- 3 Match the following pairs of an output from the right list of optimization results with a table on the left.

- soi with soi\_best
- basefuelmass with basefuelmass\_best
- fuelpress with fuelpress\_best
- grackmea with grackmea\_best
- egrlft with egrlft\_best

You can double-click in the right list to select an item, and CAGE automatically moves focus on to the next table in the left list. For example, with the `soi_best` table selected in the left list, double-click `soi` in the right list, then you can double-click `basefuelmass` for the next table, and so on.



**4** Click **Finish**.

A dialog appears indicating whether the tables have been filled successfully. Click **Close**.

**5** Switch to the **Tables** view to view the filled tables. Click **Tables** in the **Data Objects** pane, and select the tables in turn to view the results.

Look at the example finished project, `Diesel_optimization.cag`.



# Point-by-Point Diesel Engine Calibration Case Study

---

- “Point-by-Point Diesel Case Study Overview” on page 4-2
- “Create Designs and Models Programmatically” on page 4-5
- “Verify and Refine Models” on page 4-7
- “Point-by-Point Optimization Overview” on page 4-9
- “Create Point Optimizations in CAGE” on page 4-10
- “Create Multiregion Sum Optimization” on page 4-19
- “Use Optimization Results” on page 4-27

# Point-by-Point Diesel Case Study Overview

|   |
|---|
| <b>In this section...</b>   |
| “What Is Point-by-Point Modeling?” on page 4-2                          |
| “Engine Calibration Specifications and Problem Description” on page 4-2 |
| “Required Tasks” on page 4-4  |

## What Is Point-by-Point Modeling?

This case study shows how to use the Model-Based Calibration Toolbox functionality for *point-by-point* engine calibration projects.

*Point-by-point* functionality allows you to build a model at each operating point of an engine with the necessary accuracy to produce an optimal calibration. You often need point-by-point models for multiple injection diesel engines and gasoline direct-injection engines. You use point-by-point command-line functionality to handle the complexity of developing designs for each operating point.

Point-by-point models can solve calibration problems with increasingly complex engines. Engineers add engine actuators and sensors to Engine Management Systems (EMS) to respond to emerging requirements for fuel economy, performance, and control of engine emissions. In some cases, optimal engine calibration development processes that rely on two-stage modeling can no longer model engine performance responses accurately enough.

---

**Note:** Point-by-point models can provide the necessary model accuracy at *measured* operating points. However, such models do not provide estimated responses at other operating points.

---

## Engine Calibration Specifications and Problem Description

The two-stage models in the “Diesel Engine Calibration” case study generated the point-by-point data for this example. The example engine is the same six-cylinder 9.0-L common-rail L common-rail diesel engine with VGT (variable geometry turbo) and cooled EGR (exhaust gas recirculation). The control calibration is for an off-road application with an engine speed range from 1600 rpm to 2200 rpm.

The objective of the case study is to produce optimal calibration schedules as a function of commanded torque and speed for:

- Optimal SOI (start of injection)
- Fuel pressure
- Base fuel
- VGT (variable geometry turbo)
- EGR (exhaust gas recirculation)

While speed and torque define the drive cycle, the engine is tested by controlling speed and total fuel. The engine test procedure adjusts the total fuel to achieve the required torque for each point.

The case study involves models for BSFC (brake-specific fuel consumption), BSNOX (brake-specific NO<sub>x</sub> emissions), AFR (air/fuel ratio), EGR mass fraction, peak pressure, VGTSPEED, and base fuel.

The optimization setup in CAGE uses an 8-mode off-road emission test as its basis. This setup approximates up to seven mode points by neglecting the idle operating point of the engine.

The engine calibration requires tables in speed and torque for the following variables:

|                       |           |
|-----------------------|-----------|
| Best injection timing | MAINSOI   |
| Best fuel quantity    | MAINFUEL  |
| Best fuel pressure    | FUELPRESS |
| Best VGT position     | VGTPPOS   |
| Best EGR position     | EGRPOS    |

To fill these tables, create optimizations to minimize mode-weighted brake-specific fuel consumption, subject to constraints on the following parameters:

- The boundary model defining the data boundary at each operating point (BSFC\_Boundary)
- Brake-specific NO<sub>x</sub> emissions (BSNOX)
- Cylinder pressure (PEAKPRESS) at full load points only
- Air/fuel ratio (AFR)

To solve this problem, follow the steps described in “Required Tasks” on page 4-4

### Required Tasks

This example requires you to complete the following tasks:

- Create designs, model, and boundary models at each operating point. See “Create Designs and Models Programmatically” on page 4-5.
- Analyze and refine models using the Model Browser. See “Verify and Refine Models” on page 4-7.
- Optimize the model. See “Point-by-Point Optimization Overview” on page 4-9.

Creating a point-by-point optimization for the model requires these tasks:

- 1 “Create Point Optimizations in CAGE” on page 4-10
- 2 “Create Multiregion Sum Optimization” on page 4-19
- 3 “Use Optimization Results” on page 4-27



# Create Designs and Models Programmatically

## In this section...

“Overview of Programmatic Design and Modeling” on page 4-5

“Creating Designs and Models Programmatically” on page 4-5

## Overview of Programmatic Design and Modeling

The programmatic part of the case study shows automated creation of designs, models, and boundary models at each operating point. Alternatively, you could use the Model Browser to create models interactively.

This example uses the two-stage models generated in the “Diesel Engine Calibration” case study as a surrogate for an engine dynamometer or engine simulation model. The models generate the point-by-point data for this example. The example shows you how to:

- 1 Generate local designs at each operating point.
- 2 Collect response data at the design points using the diesel case study models.
- 3 Augment the local design points using spacefilling Sobol sequences if you need more design points after initial data collection.
- 4 Create local multiple models to model each of the responses at each operating point.
- 5 Build a point-by-point boundary model to define the data boundary at each operating point for later use in calibration optimization.

## Creating Designs and Models Programmatically

To create designs and models in a project:

- 1 View the command-line example instructions for “Point-by-point Modeling for a Diesel Engine” by entering:

```
edit mbcPointByPointCmdLine
```

- 2 Run the example to create designs, a project, response models, and boundary models.
- 3 Save the project to a file before loading it into the Model Browser by entering:

```
project.Save('DieselPointByPoint.mat');
```

After you complete the programmatic design, visually inspect and refine the fitted models to verify that the model quality is acceptable. You can use the command line to plot diagnostics and remove outliers, but it is easier to use the graphical and statistical tools in the Model Browser. For instructions, see “Verify and Refine Models” on page 4-7.

# Verify and Refine Models

## In this section...

“Open the Project and View Designs” on page 4-7

“Analyze and Refine Local Fits” on page 4-7

## Open the Project and View Designs

Visually inspect your fitted models to verify that the model quality is acceptable. Identify and remove any outliers that affect quality. To verify the models, open the project you created and saved in the previous section (“Create Designs and Models Programmatically” on page 4-5). You can use the command line to plot diagnostics and remove outliers. However, as a best practice, use the graphical and statistical tools in the Model Browser to analyze and refine your models .

- 1 To open the project in the Model Browser, enter:

```
mbcmodel('DieselPointByPoint.mat')
```

- 2 To view the local DOE in the Design Editor:

- a Select the test plan node in the Model Tree
- b Right-click the local model block in the test plan diagram, and select **Design Experiment**.

The Design Editor opens.

- 3 View the designs. The local designs for each test are at the top level of the design tree.
- 4 Close the Design Editor.

## Analyze and Refine Local Fits

Analyze local fits for each response as follows:

- In the Model Browser, select each local node in turn in the Model Tree. Each response has a local node labeled **Multiple Models**.
- For each local node, look for problem tests and possible outliers. Use the following tools:

- 1 To check for tests that are not fitted, click **Select Test** to open the Test Selector dialog box.
- 2 To find poorly fitted tests, select **View > RMSE Plots** to open the RMSE Explorer dialog box.
- 3 Compare alternative model types for the currently selected operating point by selecting **Model > Utilities > Select Local Model**.

The Model Selection window opens and you can compare available model types. You can leave the Model Selection window open and it updates when you change tests or remove outliers in the Model Browser.

The example projects for this case study include verified models. To view the models, load the Model Browser project file `DieselPointByPoint.mat` from the `mbctraining` folder.

## Point-by-Point Optimization Overview

The CAGE Browser section of the case study covers using the models to create optimized calibration tables. The CAGE browser part of the example takes you through the following steps:

- 1** Create two separate point optimizations for use in combination in a drive cycle optimization. You create two optimizations so you can apply the peak pressure constraint only where it is relevant, at full load conditions.
  - a** Create a constrained point optimization to minimize BSFC (brake-specific fuel consumption) for each part load point.
  - b** Create a constrained point optimization to minimize BSFC for each full load point.
  - c** Export the results to a data set. You use the results from both point optimizations as start points for a sum optimization.
- 2** Create a multiregion constrained drive cycle sum optimization.
  - a** Import the results from both point optimizations as start points.
  - b** Apply the peak pressure constraint only at full load points.
  - c** Add a second drive cycle with different weights.

You can use the second drive cycle to evaluate optimization results at intermediate points. The second drive cycle for objective and constraints includes the measured points (that is, the model operating points) and intermediate speed points.

- 3** Create tables to fill with your optimization results using the Create Tables from Model wizard.
- 4** Fill the tables with your optimization results.
- 5** Export the tables to a calibration tool.

The toolbox provides an example project containing the models, optimizations, results, and filled tables. To view the finished project, load the CAGE browser project file `DieselPointByPoint.cag` from the `mbctraining` folder.

For the next task in solving this point-by-point optimization, see “Create Point Optimizations in CAGE” on page 4-10.

# Create Point Optimizations in CAGE

### In this section...

“Introduction” on page 4-10

“Load Models to Optimize” on page 4-10

“Create Part Load Point Optimization” on page 4-11

“Create Full-Load Point Optimization” on page 4-15

## Introduction

This section takes you through the steps to create two separate point optimizations for use in combination in a drive cycle optimization. You create two optimizations so you can apply the peak pressure constraint only where it is relevant, at full load conditions.

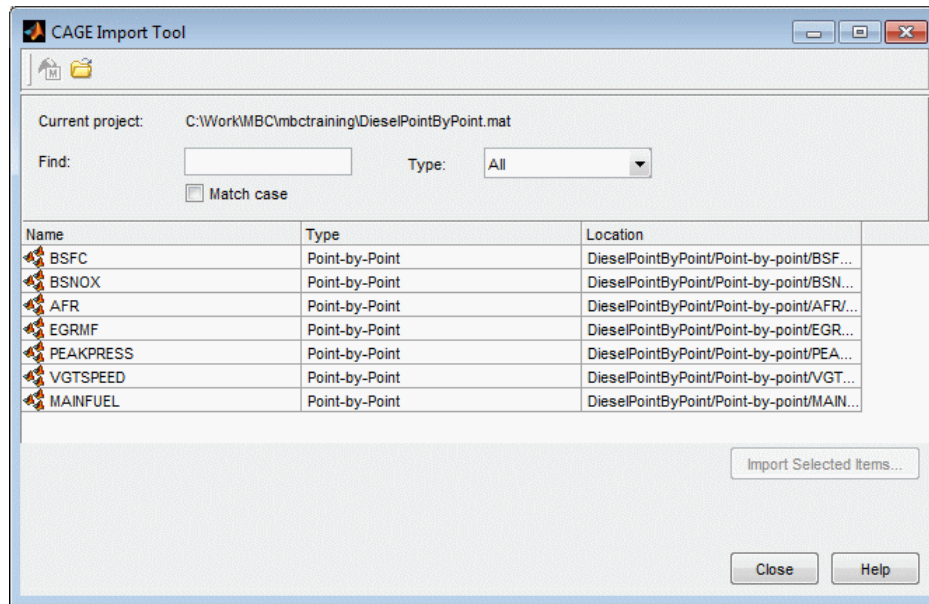
- 1 Create a constrained point optimization to minimize BSFC (brake-specific fuel consumption) for each part load point.
- 2 Create a constrained point optimization to minimize BSFC for each full load point.

You use the results from both point optimizations as start points for a sum optimization.

## Load Models to Optimize

To open CAGE and load the models for the optimization, follow these steps:

- 1 Enter `cage` to open the CAGE browser.
- 2 Load models to optimize. Select **File > Import From Project**. The CAGE Import Tool opens.
- 3 Click **Import From Project File**.
- 4 Locate the `mbctraining` folder and open the file `DieselPointByPoint.mat`.



- 5 Press **Shift+click** to select all models in the list, and click **Import Selected Items**. Then, click **OK** in the dialog box that appears next to import all the models. Then, close the Import Tool.

CAGE displays the variable connections for your imported point-by-point models.

## Create Part Load Point Optimization

To create the optimization, you perform the following tasks:

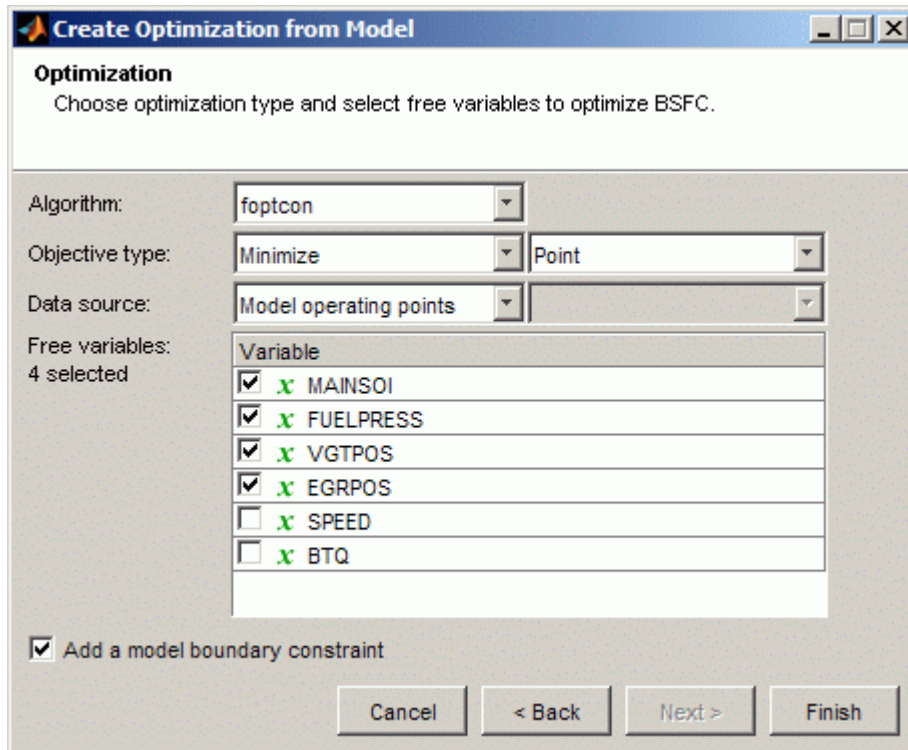
- “Select Model and Parameters for Optimization” on page 4-11
- “Specify Number of Runs” on page 4-13
- “Add a Constraint to the Optimization” on page 4-13
- “Run the Optimization” on page 4-15

### Select Model and Parameters for Optimization

- 1 Select **Tools > Create Optimization From Model**.

The Create Optimization From Model wizard opens.

- 2 Select **BSFC** as the model to optimize, and click **Next**.
- 3 Leave all the defaults as shown to create a point optimization that:
  - Minimizes **BSFC**
  - Using the **Model operating points** as the **Data source**
  - With fixed variables **SPEED** and **BTQ**
  - Selects all other variables as free variables for **CAGE** to optimize
  - Adds the boundary model as a constraint



- 4 Click **Finish** to create the optimization.

CAGE displays your optimization. Rename your optimization to **BSFC\_PartLoad** and continue to “Specify Number of Runs”.



### Specify Number of Runs

Observe the model operating points in the Optimization Point Set pane. Remove two of the runs. Later, your second optimization applies another constraint at the full load points.

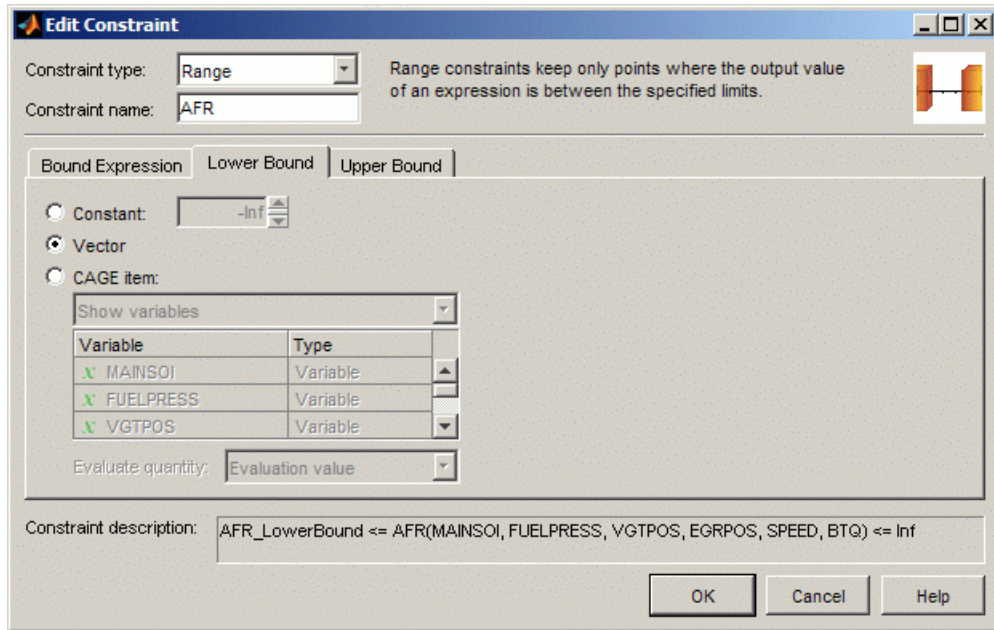
- 1 Right-click the first run, where SPEED = 2200 and BTQ = 1263, and select **Delete Runs**.
- 2 Right-click the fourth run, where SPEED = 1600 and BTQ = 1550, and select **Delete Runs**.

Your optimization now has five runs. Continue to “Add a Constraint to the Optimization”.

### Add a Constraint to the Optimization

Add a constraint for air/fuel ratio before running the optimization. Follow these steps:

- 1 Right-click the Constraints pane, and select **Add Constraint**.
- 2 Select **Constraint type Range**.
- 3 Select **Show models** from the drop-down menu on the **Bound Expression** tab.
- 4 Select AFR in the model list.
- 5 Edit the constraint name to AFR.
- 6 Click the **Lower Bound** tab, and select the **Vector** option button. Ensure the **Constraint description** matches the following example, and click **OK**.



- When the variable `AFR_LowerBound` appears in the Fixed Variables pane, copy and paste the following values into the `AFR_LowerBound` column.

| <b>AFR_LowerBound</b> |
|-----------------------|
| 27.5                  |
| 30.0                  |
| 0.0                   |
| 20.0                  |
| 22.5                  |

Verify that your fixed variable values look like those in the following image.

| Fixed Variables   |       |      |           |
|-------------------|-------|------|-----------|
| Variable:         | SPEED | BTQ  | AFR_Lo... |
| Number of values: | 1     | 1    | 1         |
| 1                 | 2200  | 947  | 27.5      |
| 2                 | 2200  | 632  | 30        |
| 3                 | 2200  | 126  | 0         |
| 4                 | 1600  | 1163 | 20        |
| 5                 | 1600  | 775  | 22.5      |

### Run the Optimization

You can now run your optimization:

- 1 Select **Optimization > Run**.
- 2 After CAGE calculates the optimization and displays the output, view the results.

### Create Full-Load Point Optimization

Creating the full-load point optimization requires you to set up the optimization, define runs at full load, add the peak pressure constraint, and define required variables.

- “Set Up the Optimization” on page 4-15
- “Import Variables to Define Constraint and Weights” on page 4-16
- “Add Peak Pressure Constraint and Define Fixed Variable Values” on page 4-17

### Set Up the Optimization

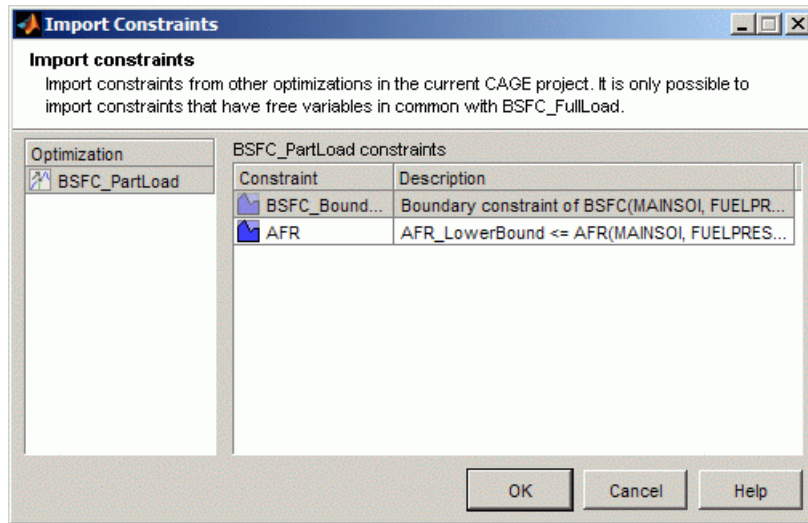
To set up the optimization, follow these steps:

- 1 Select **Tools > Create Optimization From Model**.

The Create Optimization From Model wizard opens.

- 2 Select **BSFC** as the model to optimize, and click **Next**.
- 3 Leave all the defaults, and click **Finish** to create the optimization.
- 4 Rename the new optimization to **BSFC\_FullLoad**.
- 5 Delete all but two runs in the Optimization Point Set pane, because this optimization uses only the full-load points. Right-click and delete runs to leave only these two:

- SPEED = 2200 and BTQ = 1263
  - SPEED = 1600, and BTQ = 1550
- 6 Import the AFR constraint from the previous optimization.
    - a Right-click the Constraints pane, and select **Import Constraints**.



- b Select the AFR constraint, and click **OK**.

### Import Variables to Define Constraint and Weights

You need some new variables to define a constraint and also to define some weights for the multiregion optimization. To save time you can import the variables and weights from the example project, rather than defining them manually, as follows:

- 1 Select **File > Import From Project**. The CAGE Import Tool opens.
- 2 Click **Import From Project File**.
- 3 Locate the `mbctraining` folder and open the file `DieselPointByPoint.cag`.
- 4 Select **Dataset** from the **Type** drop-down list.
- 5 Select `DriveCycle10` in the list, and click **Import Selected Items**. In the following dialog box, you also import the variables you need for later steps:
  - `BSFC_weights`

- BSNOX\_DriveCycle10\_weights

Click **OK** to import the data set and variables.

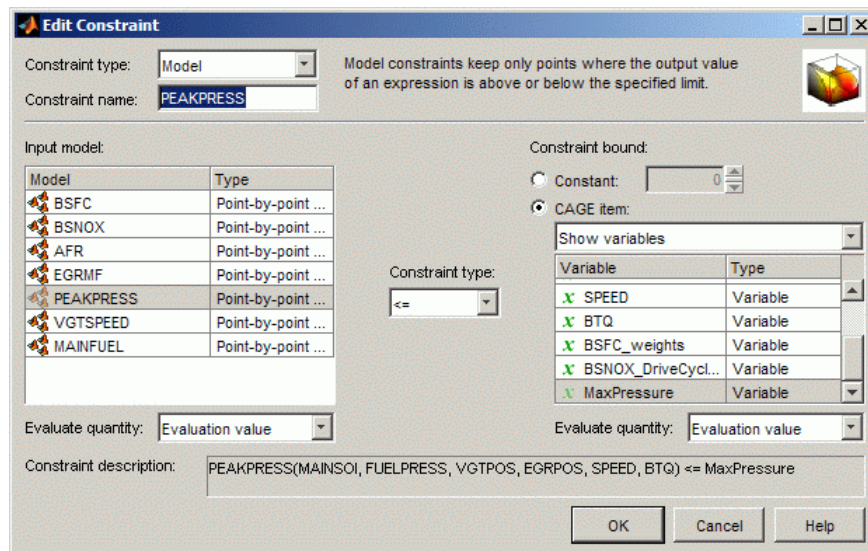
- 6 Select **Variable** from the **Type** drop-down list, select **MaxPressure** in the list, and click **Import Selected Items**. Click **OK** in the following dialog box to import the variable.

Close the Import Tool. You use all the items you imported in later steps.

### Add Peak Pressure Constraint and Define Fixed Variable Values

Return to the BSFC\_FullLoad optimization node, and add a peak pressure constraint, using the following steps:

- 1 Right-click the Constraints pane, and select **Add Constraint**.
- 2 Select the **PEAKPRESS** model.
- 3 Select the **CAGE item** option button, and then select **Show variables** from the drop-down menu.
- 4 Select **MaxPressure** in the variable list.
- 5 Rename the constraint to **PEAKPRESS**, and click **OK**.



- 6 Copy and paste the following values into the fixed variables columns for AFR\_LowerBound and MaxPressure.

| AFR_LowerBound | MaxPressure |
|----------------|-------------|
| 25.5           | 15          |
| 20             | 18          |

When you finish defining all required variables, select **Optimization > Run**. Then, continue next to “Create Multiregion Sum Optimization”.

# Create Multiregion Sum Optimization

## In this section...

“Introduction” on page 4-19

“Create Data Set from Point Optimization Results” on page 4-19

“Create Sum Optimization” on page 4-20

“Add Application Point Sets” on page 4-21

“Set Up Constraints” on page 4-23

“Define Weights and Fixed Variables and Run Optimization” on page 4-25

## Introduction

Creating a multiregion constrained drive cycle optimization requires the following tasks:

- 1 Create the optimization.
- 2 Import the results from both preliminary optimizations as start points.
- 3 Apply one of the constraints only at full load points.
- 4 Add a second drive cycle with different weights to evaluate optimization results at intermediate points.

## Create Data Set from Point Optimization Results

You have set up preliminary optimizations to generate suitable start points for the final multiregion sum optimization. To use the results from both previous optimizations as the start points, you can package the results from both optimizations into a single data set. Follow these steps:

- 1 Select the output node of your optimization, `BSFC_PartLoad_Output`.
- 2 Click **Export to data set** in the toolbar.

The **Export To Data Set** dialog box opens.

- 3 Enter `PointResults` for the new data set name, and click **OK**.
- 4 Return to the Optimization view, and select the output node of your second optimization, `BSFC_FullLoad_Output`.

- 5 Click **Export** to data set in the toolbar.

The **Export To Data Set** dialog box opens.

- 6 To add the full load results to the existing data set, select the **Modify existing** option button.
- 7 Select your data set **PointResults** in the list, leave the action set to **Append**, and click **OK**.

### Create Sum Optimization

To create the optimization, follow these steps:

- 1 Select **Tools > Create Optimization From Model**.

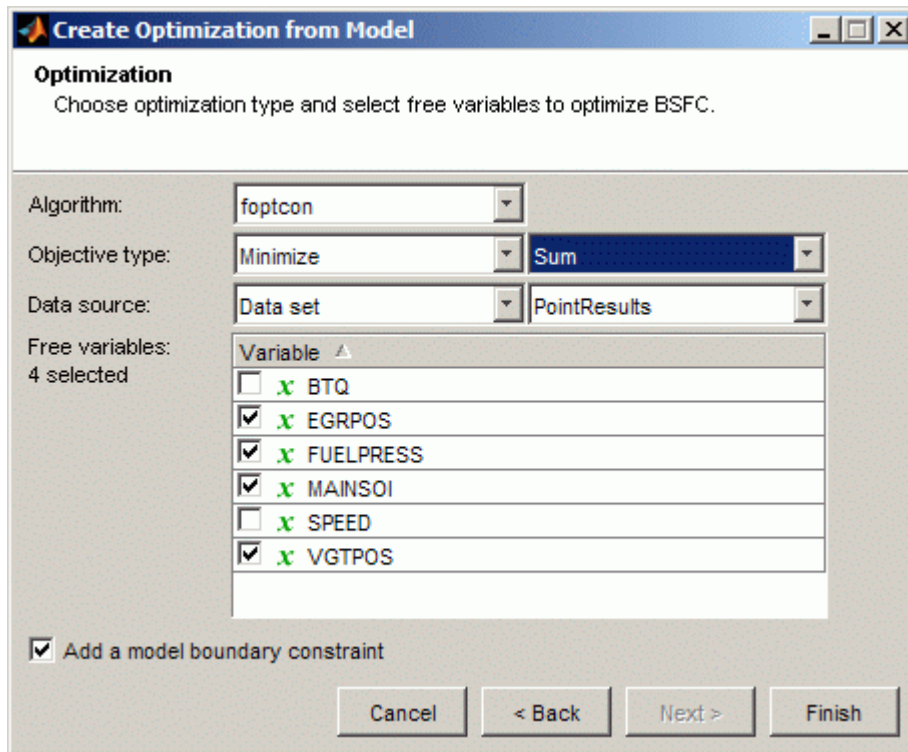
The **Create Optimization From Model** wizard opens.

- 2 Select **BSFC** as the model to optimize, and click **Next**.
- 3 Change the **Objective type** from **Point** to **Sum**.
- 4 To use your **PointResults** data set to define the optimization points, change the **Data source** to **Data set**.

Leave the rest of the defaults as shown to create a sum optimization that:

- Minimizes **BSFC**
- With fixed variables **SPEED** and **BTQ**
- Selects all other variables as free variables for **CAGE** to optimize
- Adds the boundary model as a constraint





- 5 Click **Finish** to create the optimization.

## Add Application Point Sets

You can use *application point sets* to evaluate constraints and objectives at different operating points than those points specified in the optimization. You can only use application point sets with sum optimizations.

In this case, you want to apply a peak pressure constraint only at full load. You can use an application point set to define the points where you want to apply this constraint.

You also want to evaluate the BSFC objective and a NO<sub>x</sub> constraint over a different drive cycle to the optimization points. You can use an application point set to define the secondary drive cycle points. This approach allows you to run an optimization at a subset of points, and evaluate interpolated results across an application point set.

To set up your application point sets, follow these steps:

- 1 To add an application point set to your optimization objective, right-click the objective **BSFC**, and select **Select Application Point Set**

The Select Operating Point Variables dialog box opens.

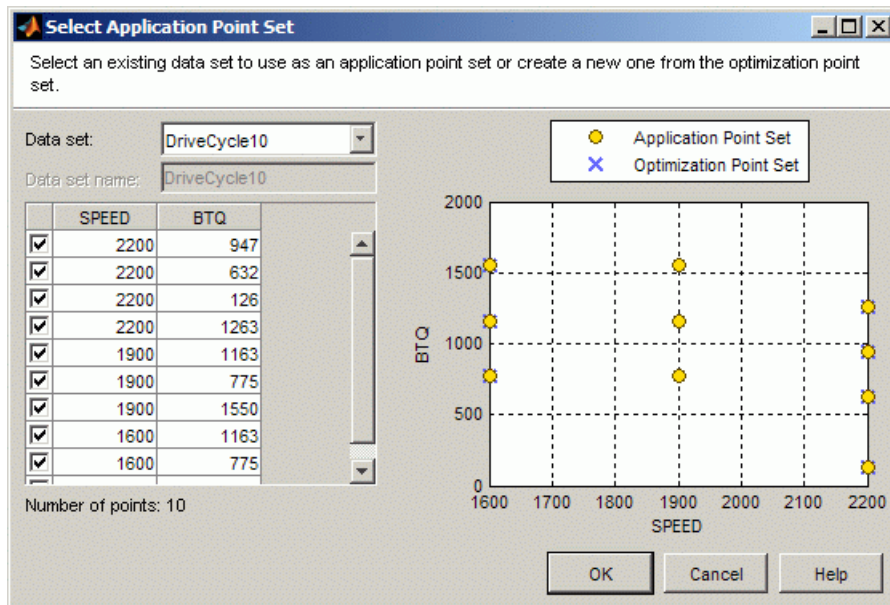
- 2 Define a pair of the fixed variables as the application point variables to use with this optimization. You only select variables once per optimization.

In this case, you only have two fixed variables available (**BTQ** and **SPEED**), so leave the defaults, and click **OK**.

The Select Application Point Set dialog box opens.

- 3 Select the **DriveCycle10** data set you imported earlier. View the plot displaying the application points and optimization points.

Some of the application points coincide with the optimization points, and others lie between them. CAGE extrapolates the optimization results to evaluate the objective at these intermediate points.



- 4 Click **OK** to apply the application point set to the objective.

## Set Up Constraints

- “Import Constraints from a Previous Optimization” on page 4-23
- “Apply the Peak Pressure Constraint at Full Load” on page 4-23
- “Add NOx Sum Constraints” on page 4-24

### Import Constraints from a Previous Optimization

Import the AFR and PEAKPRESS constraints from the previous optimization.

- 1 Right-click the Constraints pane, and select **Import Constraints**.
- 2 Select the AFR and PEAKPRESS constraints from the BSFC\_FullLoad optimization, and click **OK**.

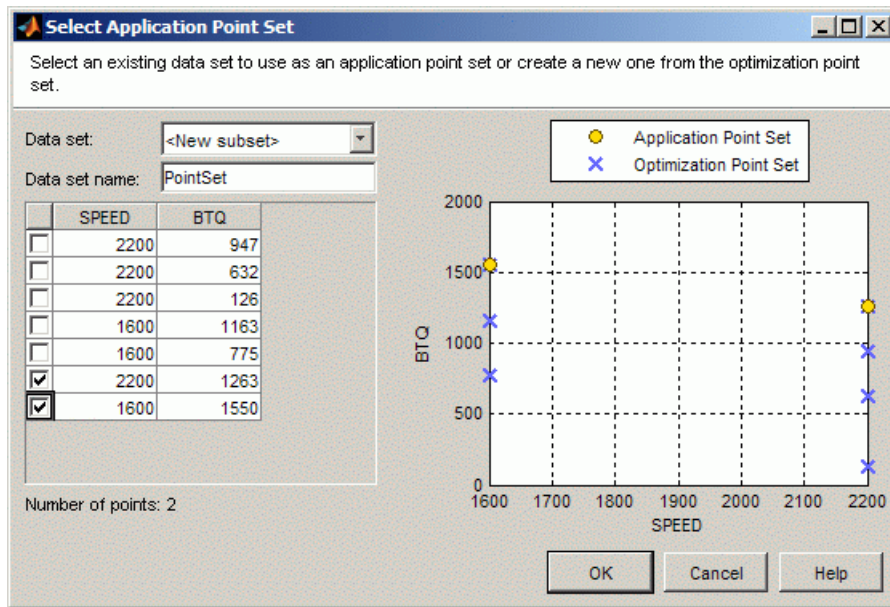
### Apply the Peak Pressure Constraint at Full Load

To apply the peak pressure constraint only at full load, add an application point set to the PEAKPRESS constraint.

- 1 Right-click the constraint PEAKPRESS, and select **Select Application Point Set**

The Select Application Point Set dialog box opens.

- 2 Select **New subset** from the **Data set** list. You can use this setting to select a subset of your optimization points for an objective or constraint.
- 3 Select the check boxes for the full load points only. To select points, you can use the check boxes or click points on the plot.



- 4 View the plot displaying the application points where the constraint applies, and click **OK**.

### Add NOx Sum Constraints

To add the NOx sum constraints,






- 1 Right-click the Constraints pane, and select **Add Constraint**.  
The Edit Constraint dialog box opens.
- 2 Select the **Sum Constraint Constraint type**.
- 3 Select the **BSNOX** model.
- 4 Enter 0.003 for the **Constraint bound** and press **Enter**. CAGE closes the dialog box.
- 5 Ensure that the constraint description shows the weighted sum  $\leq 0.003$ .
- 6 Right-click the BSNOX constraint, and select **Duplicate**.  
The Edit Constraint dialog box opens.
- 7 Rename the constraint to **BSNOX\_DriveCycle10**, and click **OK**.

- To evaluate the constraint at different operating points than the points specified in the optimization, right-click the constraint `BSNOX_DriveCycle10`, and select **Select Application Point Set**

The Select Application Point Set dialog box opens.

- Select the `DriveCycle10` data set. View the plot, and click **OK**.

Your sum optimization should contain the following constraints.

| Constraints  |                                  |  |
|--|----------------------------------|--|
| Name   | Description                      | Application Point Set                              |
|  BSFC_Boundary      | Boundary constraint of BSFC(M... |  |
|  AFR                | AFR_LowerBound <= AFR(MAI...     |  |
|  PEAKPRESS          | PEAKPRESS(MAINSOI, FUELPR...     | FullLoad(SPEED,BTQ)                                |
|  BSNOX              | Weighted sum of BSNOX(MAIN...    |  |
|  BSNOX_DriveCycle10 | Weighted sum of BSNOX(MAIN...    | DriveCycle10(SPEED,BTQ;BSNOX_DriveCycle10_weights) |

## Define Weights and Fixed Variables and Run Optimization

Next, review the Fixed Variables pane. To define new weights and fixed variables, change the default values for `BSFC_weights`, `AFR_LowerBound`, `MaxPressure`, `BSNOX_weights` and `BSNOX_DriveCycle10_weights` as described in the following steps:

- Increase the **Number of values** to 7 for `AFR_LowerBound`, `MaxPressure`, `BSNOX_weights` and `BSNOX_DriveCycle10_weights`.
- Copy and paste the following values into the fixed variables columns.

| BSFC_weights | SPEED  | BTQ    | AFR_LowerBound | Max Pressure | BSNOX_weights | BSNOX_DriveCycle10_weights |
|--------------|--------|--------|----------------|--------------|---------------|----------------------------|
| 0.15         | 2200.0 | 947.0  | 27.5           | 18           | 0.15          | 0.15                       |
| 0.15         | 2200.0 | 632.0  | 30.0           | 18           | 0.15          | 0.15                       |
| 0.15         | 2200.0 | 126.0  | 0.0            | 18           | 0.15          | 0.15                       |
| 0.1          | 1600.0 | 1163.0 | 20.0           | 18           | 0.1           | 0.1                        |
| 0.1          | 1600.0 | 775.0  | 22.5           | 18           | 0.1           | 0.1                        |

| BSFC_weights | SPEED  | BTQ    | AFR_LowerBound | Max Pressure | BSNOX_weights | BSNOX_DriveCycle10_weights |
|--------------|--------|--------|----------------|--------------|---------------|----------------------------|
| 0.15         | 2200.0 | 1263.0 | 25.5           | 15           | 0.15          | 0.15                       |
| 0.1          | 1600.0 | 1550.0 | 20.0           | 18           | 0.1           | 0.1                        |

---

**Note:** Weights in application point sets override weights in the Fixed Variables pane. If an objective or constraint has an application point set, CAGE applies the weights in the application point set if the name matches, for example “BSNOX\_weights”.

---

**3** Select **Optimization > Run**.

Inspect the optimization results before proceeding to the next task, “Use Optimization Results.”

## Use Optimization Results

### In this section...

“Introduction” on page 4-27

“Create Tables to Fill” on page 4-27

“Fill Tables from Optimization Results” on page 4-28

“Export Tables” on page 4-29

### Introduction

This section takes you through the following steps:

- 1 Create tables to fill with your optimization results using the Create Tables from Model wizard.
- 2 Fill the tables with your optimization results.
- 3 Export the tables to a calibration tool.

### Create Tables to Fill

You can use the Create Tables from Model wizard to create a set of tables with the same axes for all the inputs of a model, and the model response. You can add tables for any other responses that share the same inputs. You can choose which of these tables to create and select the values for the axes (or normalizers) that all tables share. You can also add all the new tables to a tradeoff.

- 1 Select **Tools > Create Tables From Model** (or use the toolbar button).

The **Create Tables From Model** Wizard opens.

- 2 Select the BSFC model to base the new tables on.

If you are viewing an optimization or an optimization output node, then the wizard automatically selects the model in the first objective. You can use this feature to create tables for the selected optimization.

Click **Next**.

- 3 As you have point-by-point models, CAGE automatically selects the check box to **Use model operating points** for the table normalizers. To edit the normalizers, clear the check box.

- 4 Enter 11 for the **Table rows** and **Table columns**, and click **Next**.
- 5 Select check boxes to specify which variables and responses to create tables for. CAGE automatically selects your free variables and the objective response (BSFC). You can add tables for other responses with the same the same inputs as the primary model (and the same operating points for point-by-point models). Select the check boxes to add AFR, BSNOX and MAINFUEL.
- 6 By default you also create a tradeoff containing all the new tables. The tradeoff can be useful for specifying fill factors for tables and for investigating optimization results. Leave the check box selected to **Create a tradeoff containing all the tables**.
- 7 Click **Finish** to create the tables and tradeoff.

### Fill Tables from Optimization Results

To fill the tables from your optimization results:

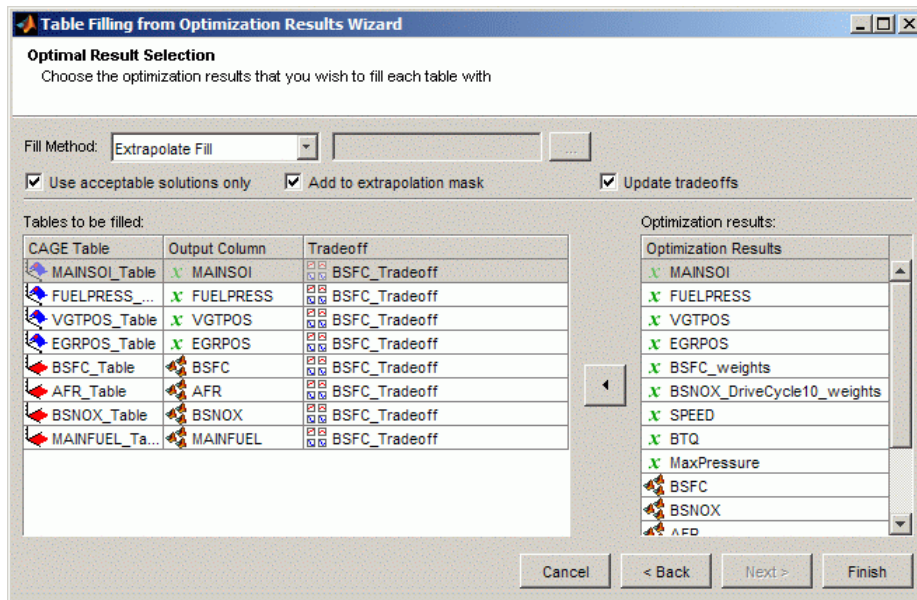
- 1 Select the output node of your multiregion sum optimization, `BSFC_Optimization_Output`.
- 2 Select **Solution > Fill Tables**, or click the toolbar button.

The Table Filling wizard opens.

- 3 **Shift**+click to select all the tables in the list, and click the button to move them to list of tables to fill. Click **Next**.



4



Check that all tables match to the correct filling item, and clear the check box to **Update tradeoffs**.

- 5 Click **Finish** to fill the tables.
- 6 View your optimization results in the Tables view.

## Export Tables

To export your calibration data to file or to a calibration tool,

- 1 Select **File > Export > Calibration > Selected Item** or **All Items**.

The Export Calibration Data dialog opens.

- 2 Select the check boxes of the calibration items you want to export.

All tables and normalizers in the project are in the list of calibration items.

- 3 Select the format you want to export to:

- ATI Vision
- ATI Vision MAT file

- INCA DCM file
- Simple CSV file
- Simple MAT file
- Simple MATLAB file

Click **OK** to export your selected items.

- 4 If you selected **ATI Vision**, the ATI Vision Connection Manager dialog opens.

If you select a file format, a file browser appears. Choose a location and filename, and click **Save**.

# Composite Models and Modal Optimizations

---

- “Introducing Composite Models and Modal Optimization” on page 5-2
- “Composite Model and Modal Optimization Projects” on page 5-3

## Introducing Composite Models and Modal Optimization

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

See the next section for example projects you can use to learn about modal optimizations, “Composite Model and Modal Optimization Projects” on page 5-3.

For more information, see:

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

## Composite Model and Modal Optimization Projects

### In this section...

“Gasoline Example with Four Cylinder and Eight Cylinder Modes” on page 5-3

“Diesel Example with Low and High EGR Modes” on page 5-3

“Composite Model Example with Separate Tables for Each Mode” on page 5-4

### 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.

For instructions showing examples from this project describing how to create and use composite models and modal optimizations, see:

- 
- 
- 
- 
- 

### Diesel Example with Low and High EGR Modes

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

- DieselComposite.mat
- DieselComposite.cag

This diesel example shows the interaction of composite models and point-by-point models. The `DieselComposite.cag` example shows optimizations created from a composite model imported from `DieselComposite.mat`. The composite model combines point-by-point response models for two engine operating modes: low and high exhaust gas recirculation (EGR).

Operating points are repeated in different modes. When using modal optimization to choose the best mode at an operating point, you need a single run for each operating point, with the mode as a free variable. Use the Create Optimization from Model wizard to remove repeated operating points as follows:

- 1 To specify an optimization run for each unique operating point, use the **Data Source** option **Unique operating points** in the Create Optimization from Model wizard. This option displays only for composite models created from point-by-point models.
- 2 The wizard automatically selects the mode variable as a free variable when you use the Create Optimization from Model wizard to create a modal optimization from a composite model.

In this example, both modes use the same operating points except for one point. Select **Model > Properties** to view which modes are available for each operating point. See .

If you want to run a separate optimization for each mode, use the **Model operating points** option in the Create Optimization from Model wizard. Then, delete the runs for the modes you do not want.

### Composite Model Example with Separate Tables for Each Mode

To see an example project where the strategy has separate tables for each mode, load the example project `CompositeWith2Tables.cag` found in `matlab\toolbox\mbc\mbctraining`.

In this example project there is a single table for each control variable which stores the value for the best mode. Composite calibration problems of this kind often involve separate optimizations (point and sum) with different free variables and constraints for each mode.

- 1 Each mode has a separate point optimization.

- 2** The results from each mode have been exported to the same data set, using the append option.
- 3** To finish off the calibration the sum optimization provides results for a multimodal drive cycle, using the selected mode at each point. The sum optimization was created as follows:
  - Select the combined point results data set for the optimization points.
  - Import point constraints from the point optimizations.
  - Add table gradient constraints for ICP and ECP. Separate table gradient constraints are required for different modes because the goal is to fill separate tables for each mode.

Add an application point set to restrict table gradient constraints to mode 1 points only. Use the Interpolate by mode option.

Duplicate the table gradient constraints and create an application point set to restrict them to mode 2 points only.
- 4** This example uses table-filling filter rules to fill tables using only the appropriate mode. For details see .





# Design and Modeling Scripts

---

- “Introduction to the Command-Line Interface” on page 6-2
- “Automate Design and Modeling With Scripts” on page 6-3
- “Understanding Model Structure for Scripting” on page 6-6
- “How the Model Tree Relates to Command-Line Objects” on page 6-10

## 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 Demos for command-line examples.

# Automate Design and Modeling With Scripts

## In this section...

“Processes You Can Automate” on page 6-3

“Engine Modeling Scripts” on page 6-5

## Processes You Can Automate

The following description illustrates an example engine modeling process you can automate with the command-line Model-Based Calibration Toolbox product. You can assemble the commands for these steps into an easy-to-use script or graphical interface. This is a guideline for some of the steps you can use to model engine data.

- 1 Create or load a project — `CreateProject`; `Load`
- 2 Create a new test plan for the project using a template set up in the Model Browser — `CreateTestplan`
- 3 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`.
- 4 Create or load a data object for the project and make it editable — `CreateData`;  
`BeginEdit`
- 5 Load data from a file or the workspace — `ImportFromFile`;  
`ImportFromMBCDataStructure`

You can instead specify the required data file when you call `CreateData`; you must still call `BeginEdit` before you can then make changes to the data.

- 6 Work with the data:
  - Examine data values — `Value`
  - Modify the data to remove unwanted records — `AddFilter`; `AddTestFilter`
  - Add user-defined variables to the data — `AddVariable`
  - Add new data — `Append`
  - Group your data for hierarchical modeling by applying rules — `DefineTestGroups`; `DefineNumberOfRecordsPerTest`
  - Export your data to the workspace — `ExportToMBCDataStructure`

- 7 Save your changes to the data, or discard them — `CommitEdit`; `RollbackEdit`
- 8 Designate which project data object to use for modeling in your test plan — `AttachData`
- 9 Create and evaluate boundary models, either in a project or standalone. You can use boundary models as design constraints. See “Boundary Model Scripting” on page 6-8.
- 10 Create models for the data; these can be one- or two-stage models and can include datum models — `CreateResponse`
- 11 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`
  - 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), then:
    - Fit to new data (`Fit`)
    - Create a copy of the model, change model type, properties and fit algorithm settings (`CreateModel`; `ModelSetup`; `Type` (for models); `Properties` (for models); `CreateAlgorithm`)
    - Include and exclude terms to improve the model (`StepwiseRegression`)
    - Examine regression matrices and coefficient values (`Jacobian`; `ParameterStatistics`)
    - If you change the model you need to use `UpdateResponse` to replace the new model back into the response in the project.
  - For two-stage test plans, once you are satisfied with the fit of the local and response feature models (and have selected best models from any alternatives you created), you can calculate the two-stage model — `MakeHierarchicalResponse`.
  - Now you can also examine the predicted values and PEV of the two-stage model.

- You can export any of these models to MATLAB or Simulink software — Export

This overview is not an exhaustive list of the commands available. For more information on all available functions, see “Automation Scripting”.

## Engine Modeling Scripts

See Model-Based Calibration Toolbox Examples for a selection of command-line script 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

## Understanding Model Structure for Scripting

| In this section...  |
|---|
| “Projects and Test Plans for Model Scripting” on page 6-6 |
| “Response Model Scripting” on page 6-6                    |
| “Boundary Model Scripting” on page 6-8                    |

### 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 6-10, 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 6-10, 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**.

To learn about all the available functions for constructing and working with models, see “Modeling”.

## 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.

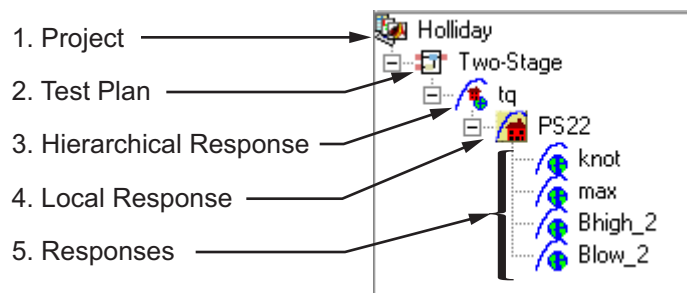
You can change the `ActiveInputs`, `Evaluate`, and use as a `designconstraint`.

For a complete list of boundary model classes, methods and properties, see “Boundary Models”.

## 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:

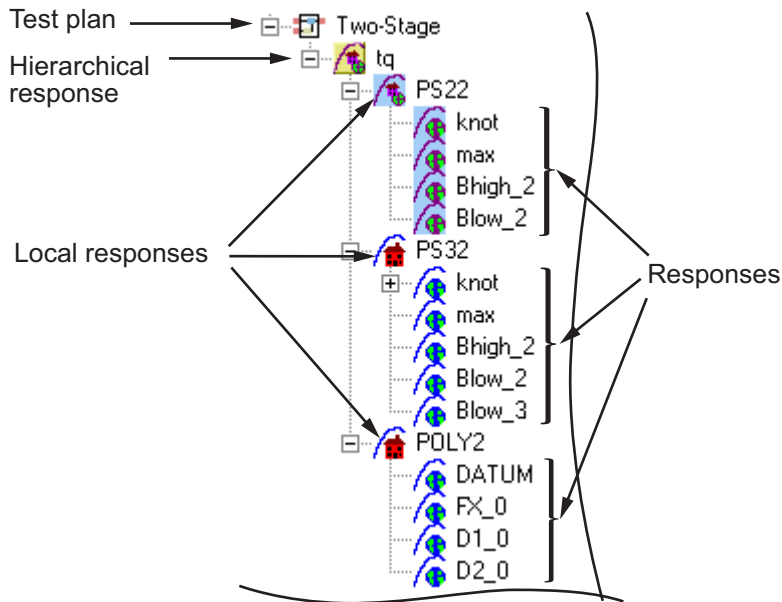
- 1 Project
- 2 Test Plan
- 3 Hierarchical Response
- 4 Local Response
- 5 Responses

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.

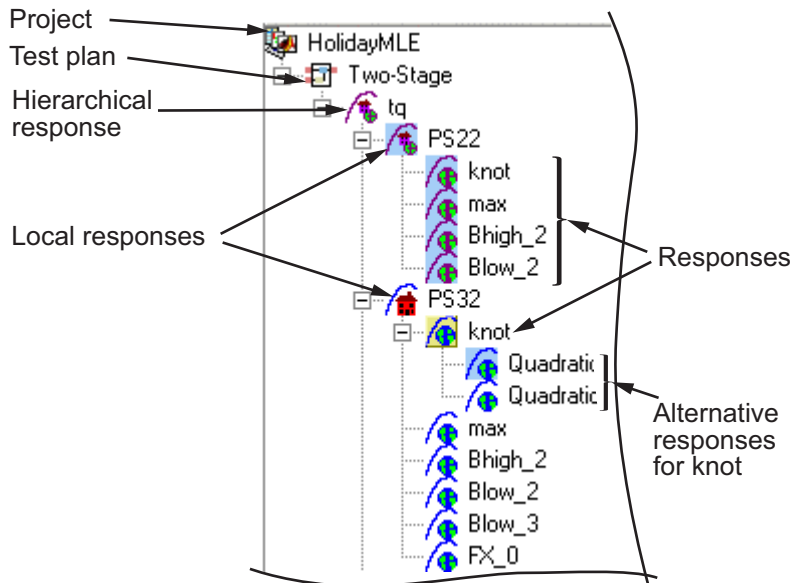


Hierarchical responses can have multiple local responses, as shown in the following example from the Model Browser. In the command-line interface these are accessible via the property `LocalResponses` for a hierarchical response object (`mbcmodel.hierarchicalresponse`). In this example, the local responses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

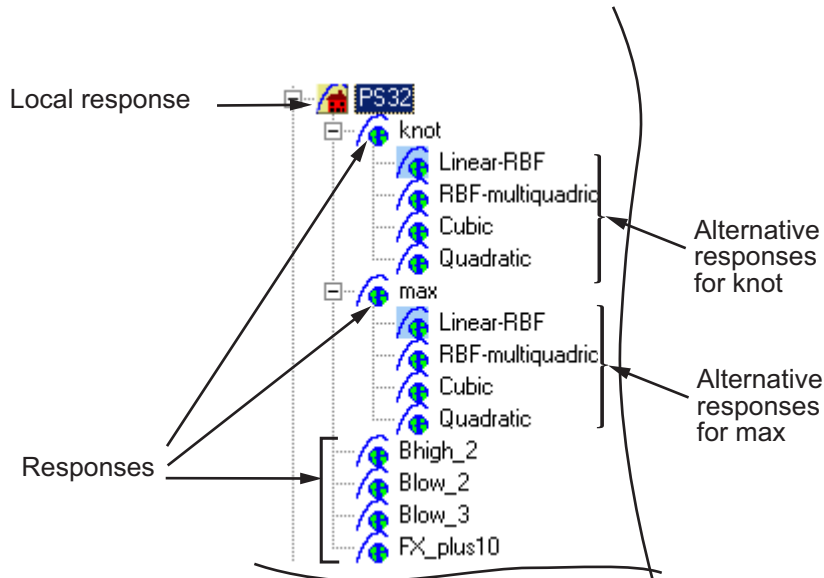


Responses can have zero or more alternative responses, as shown in the following model tree. You call the method `CreateAlternativeModels` on the command line to do the same.



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`.

# Multi-Injection Diesel Calibration

---

- “Multi-Injection Diesel Calibration Workflow” on page 7-2
- “Design of Experiment” on page 7-22
- “Statistical Modeling” on page 7-40
- “Optimization” on page 7-46

## Multi-Injection Diesel Calibration Workflow

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

### 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 NO<sub>x</sub> 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



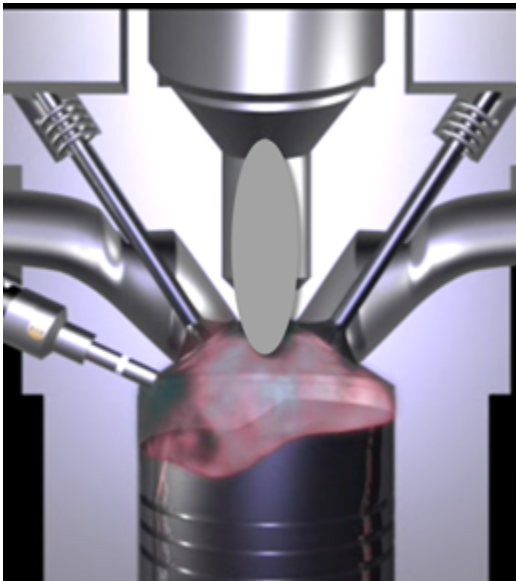
- Best variable geometry turbocharger (VGT) vane position
- Best fuel rail pressure relative to nominal pressure vs. engine speed

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

### **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 NO<sub>x</sub>. 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.

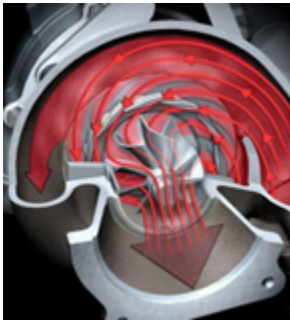
At low loads, you can use lower fuel pressure to decrease fuel pump power losses without much effect on emissions and power efficiency.

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 NO<sub>x</sub> 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.

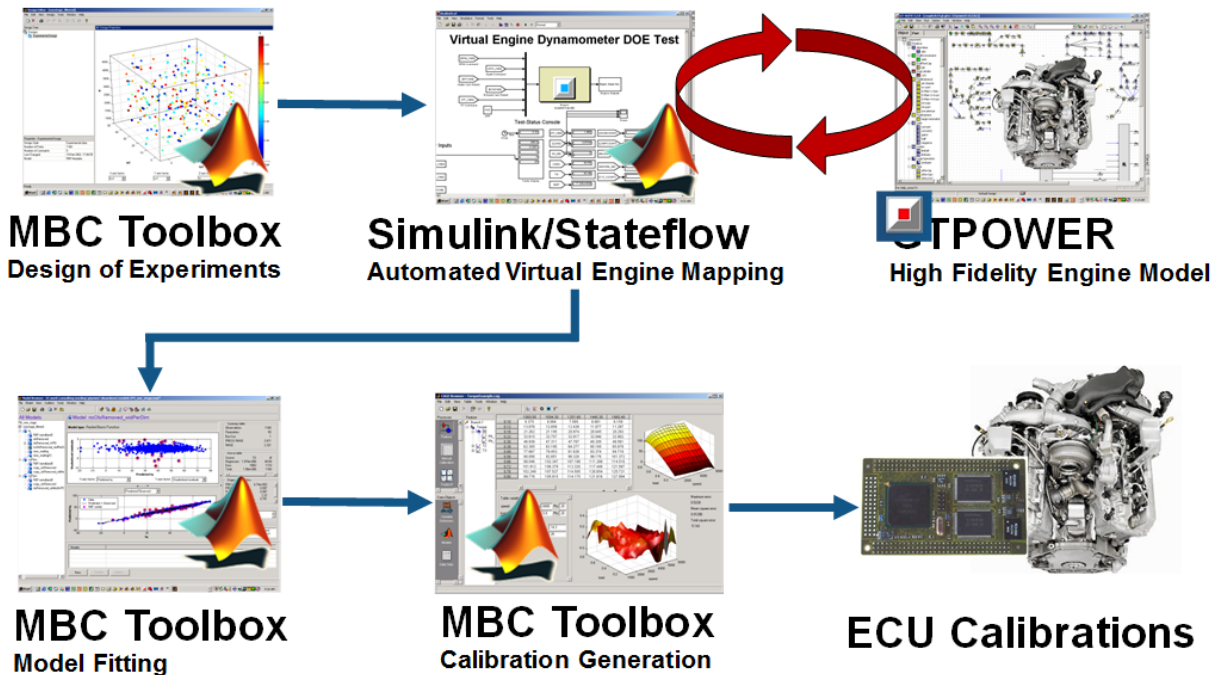
Reburning exhaust gases decreases in-cylinder temperature, resulting in a significant decrease in NOx emissions.

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.

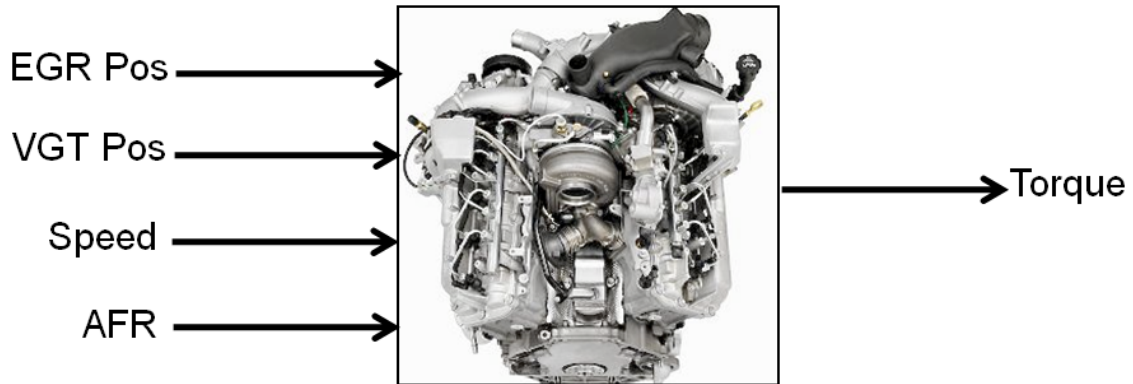
### **Engine Calibration Workflow**

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<sup>®</sup>, third-party high-fidelity simulation tools, and Hardware-in-the-Loop testing for fine tuning calibrations on ECUs.



### 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.



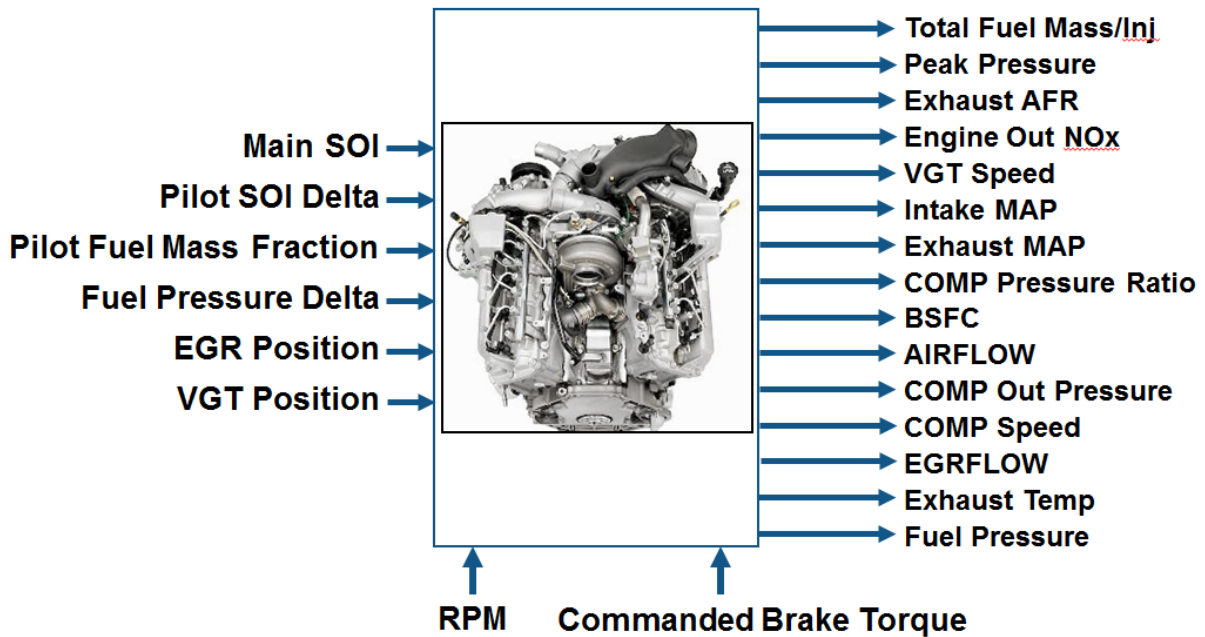
These simplifications were used to conduct the initial study:

- 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.



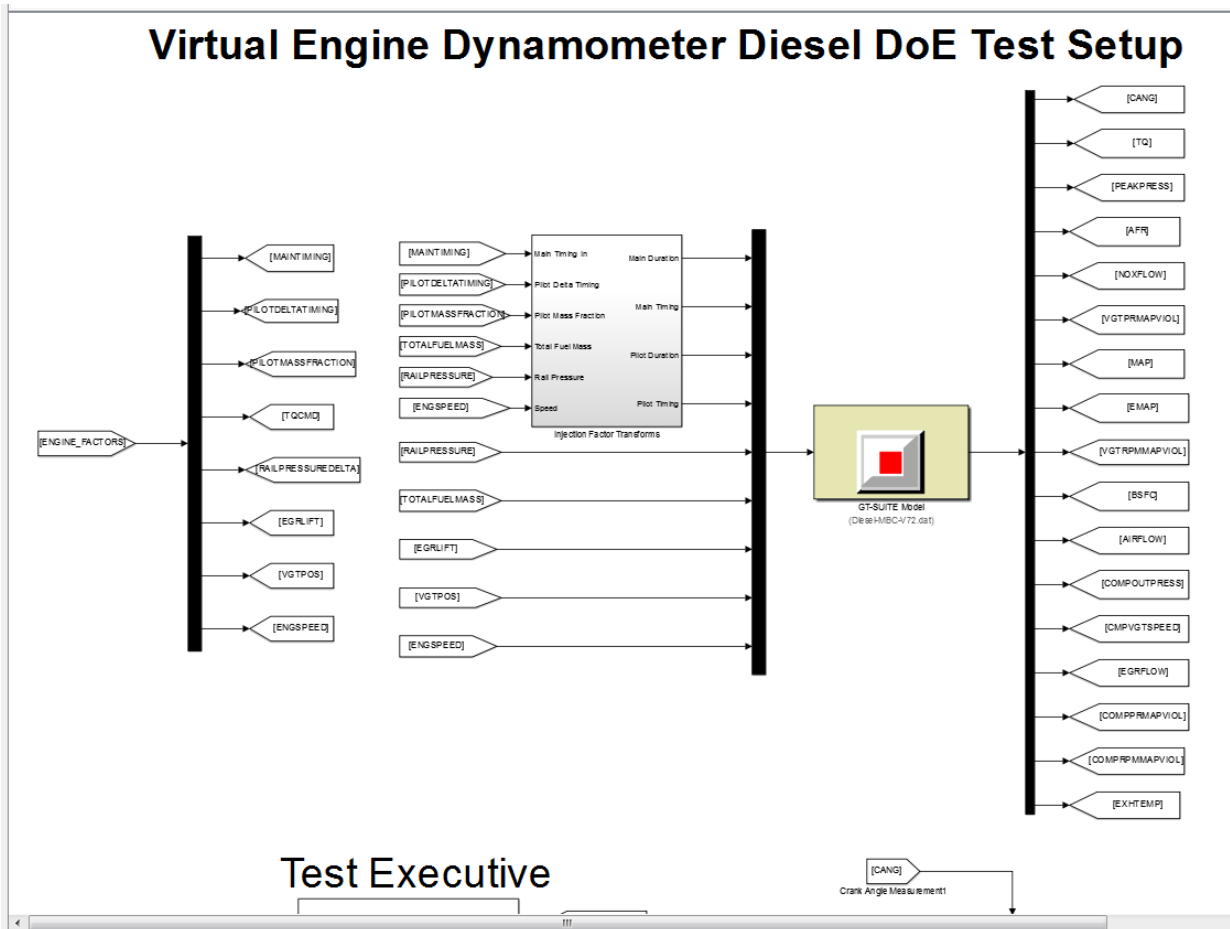
## Data Collection and Physical Modeling

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.



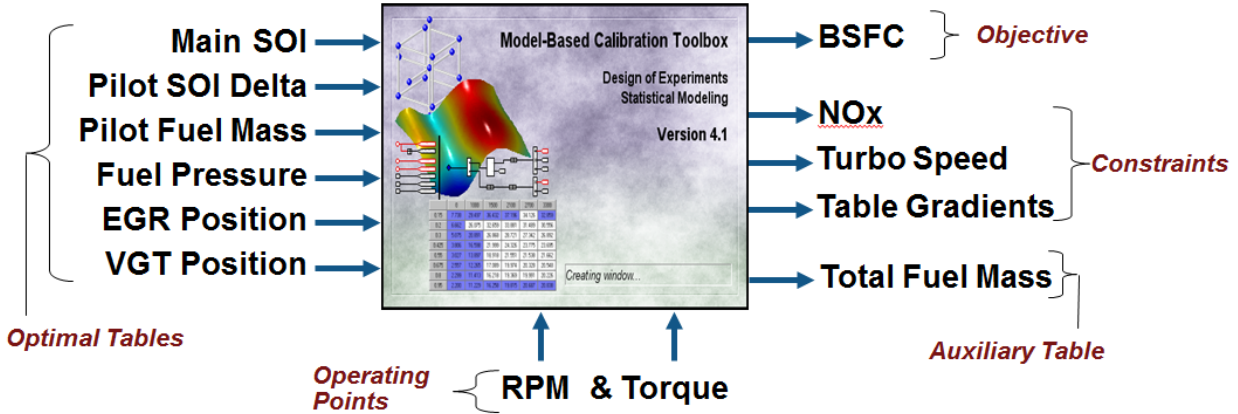


## Statistical Modeling

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**



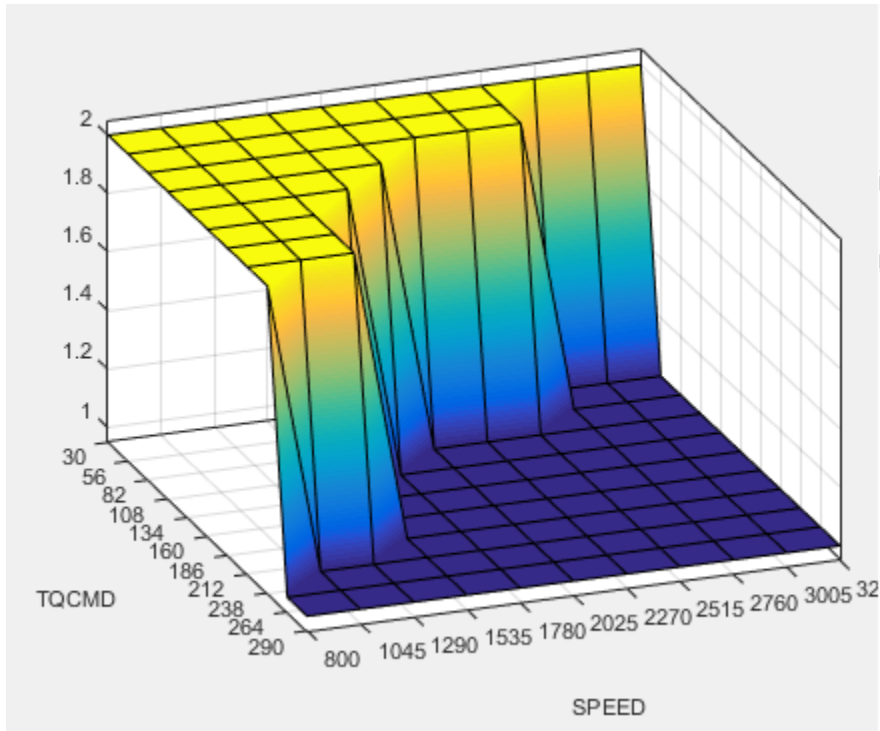
Goal: Minimize BSFC subject to NOx, turbocharger speed, and user-specified table gradient constraints

### 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.

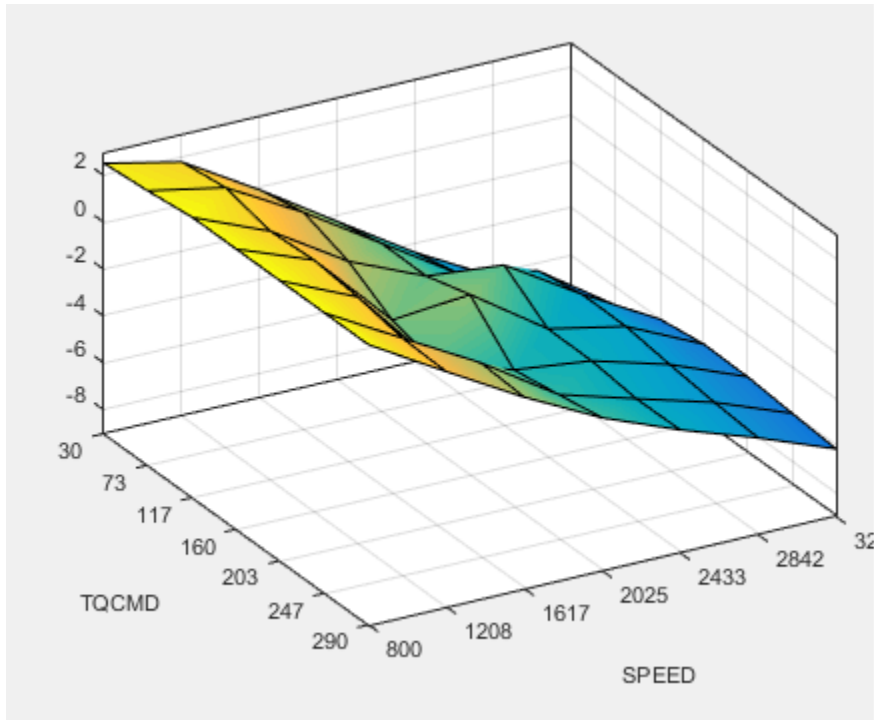


### Pilot Mode Table

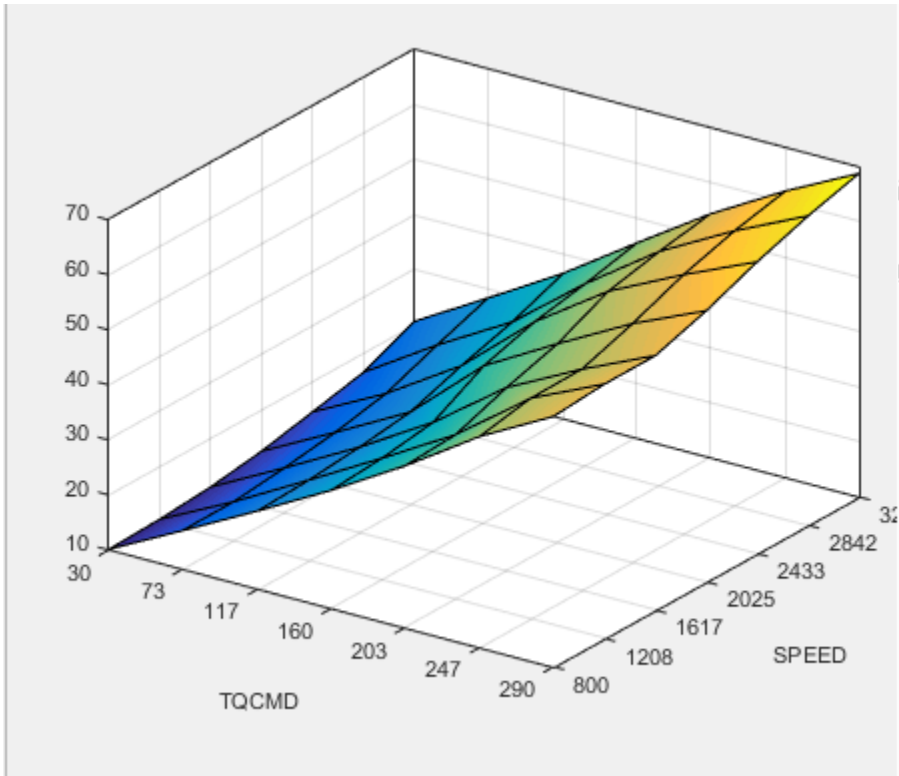
This graphic shows the plot of the table to select the active or inactive pilot mode depending on the speed and commanded torque

You need to fill calibration tables for each control variable described in “Multi-Injection Diesel Problem Definition” on page 7-2, in both pilot modes, active and inactive.

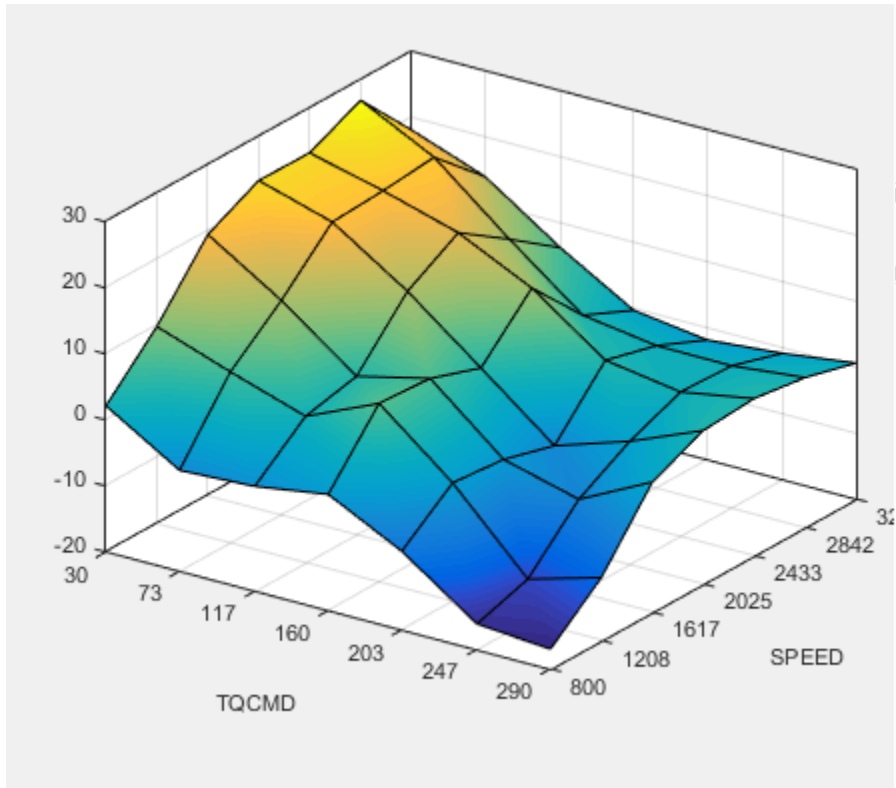
Following are all the pilot active tables.



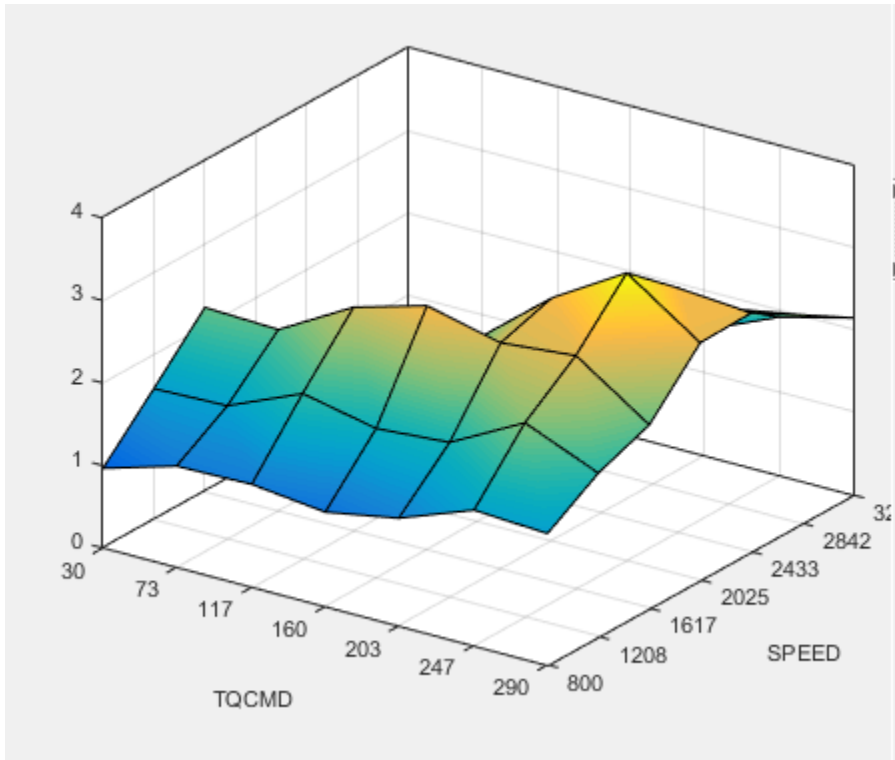
**Main Start of Injection (SOI) Timing Table**



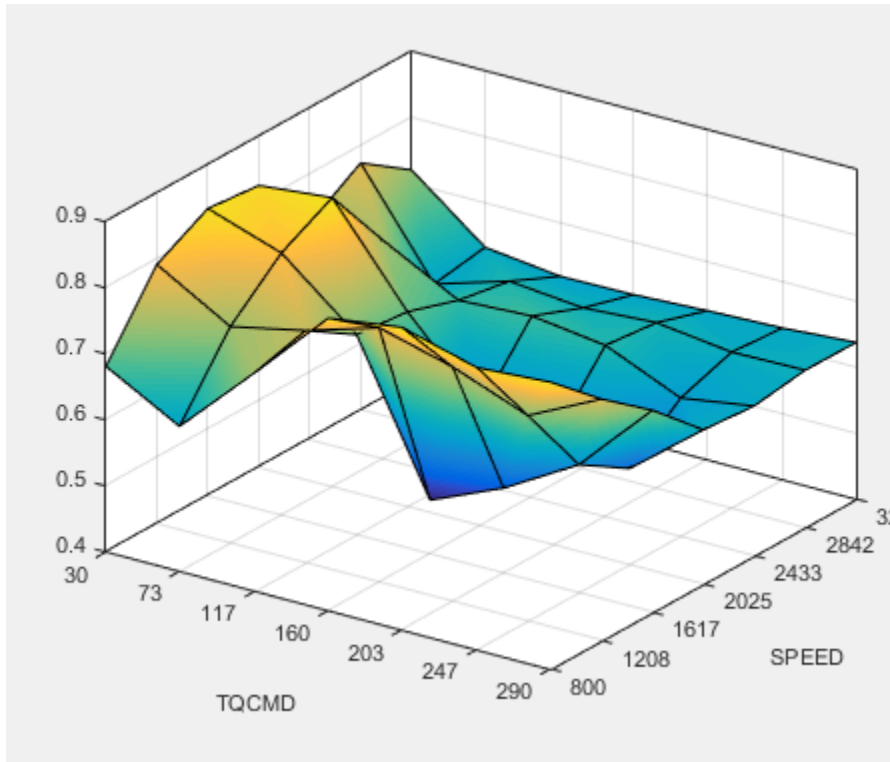
**Total Injected Fuel Mass Table**



Fuel Pressure Delta Table

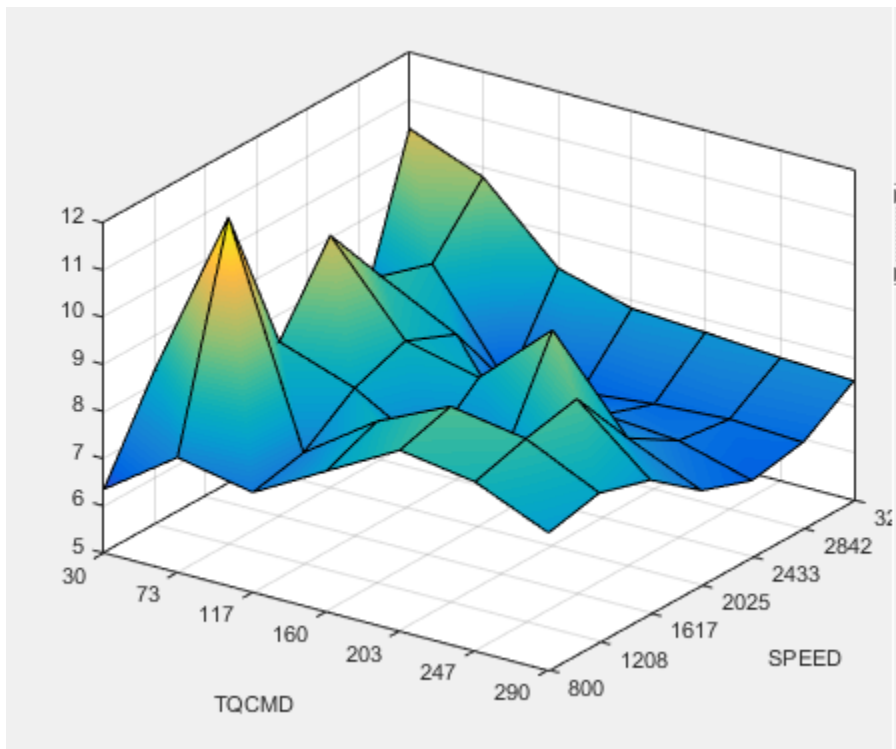


**Exhaust Gas Recirculation (EGR) Valve Position Table**

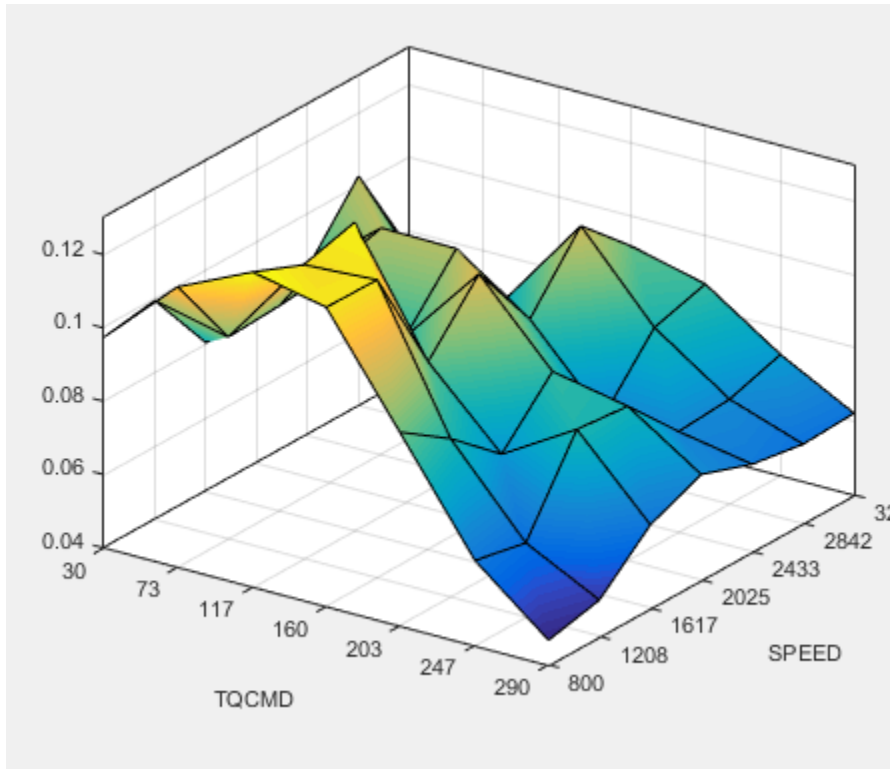


**Variable-Geometry Turbo (VGT) Position Table**





**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 7-22.
- 2 Finished statistical models, in topic “Statistical Modeling” on page 7-40.
- 3 Optimization setup and results, and filled calibration tables, in topic “Optimization” on page 7-46.

Use these example files to understand how to set up systematic calibrations for similar problems. For next steps, see “Design of Experiment” on page 7-22.

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.

## Design of Experiment

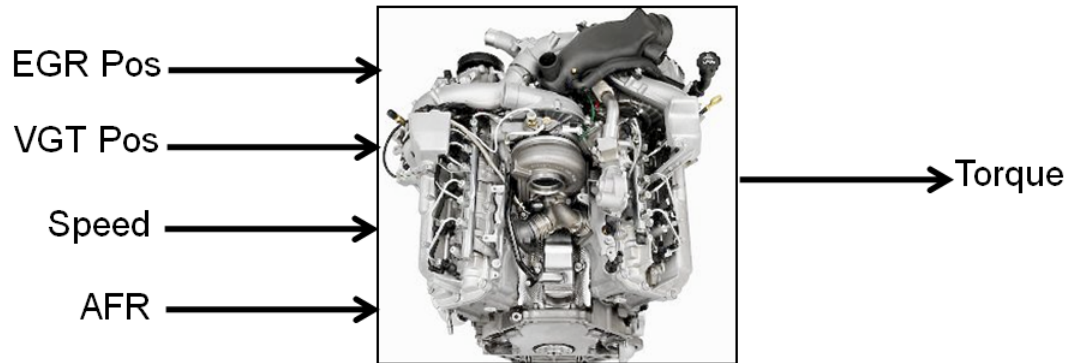
| In this section...  |
|---|
| “Benefits of Design of Experiment” on page 7-22                                 |
| “Air-System Survey Testing” on page 7-22  |
| “Create Designs and Collect Data” on page 7-23                                  |
| “Fit a Boundary Model to Air Survey Data” on page 7-29                          |
| “Use the Air Survey and Boundary Model to Create the Final Design” on page 7-32 |
| “Multi-Injection Testing” on page 7-39  |

### 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.

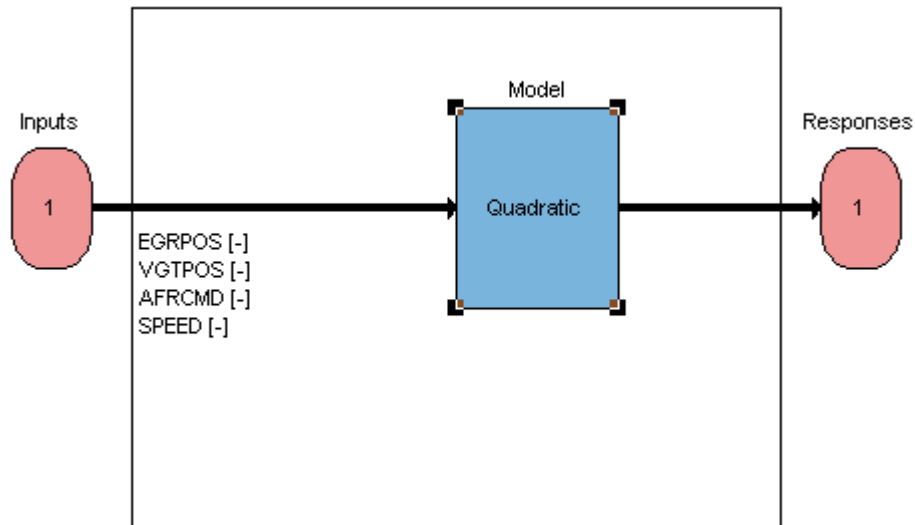
## Create Designs and Collect Data

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

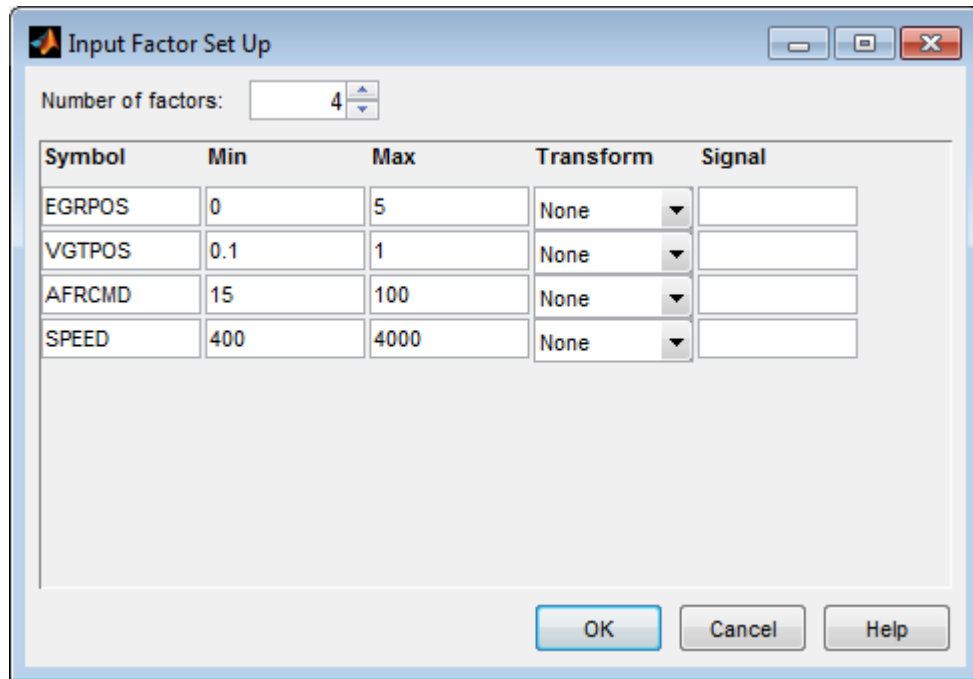
To create a design, you need to first specify the model inputs. Open the example file to see how to define your test plan.

- 1 Open MATLAB, and then open the Model Browser by entering:  

```
mbcmodel
```
- 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 opens and displays the top project mode in the **All Models** tree, `CI_Multiinject_AirSurvey`.
- 4 To see how to define your test plan, 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**.
- 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.

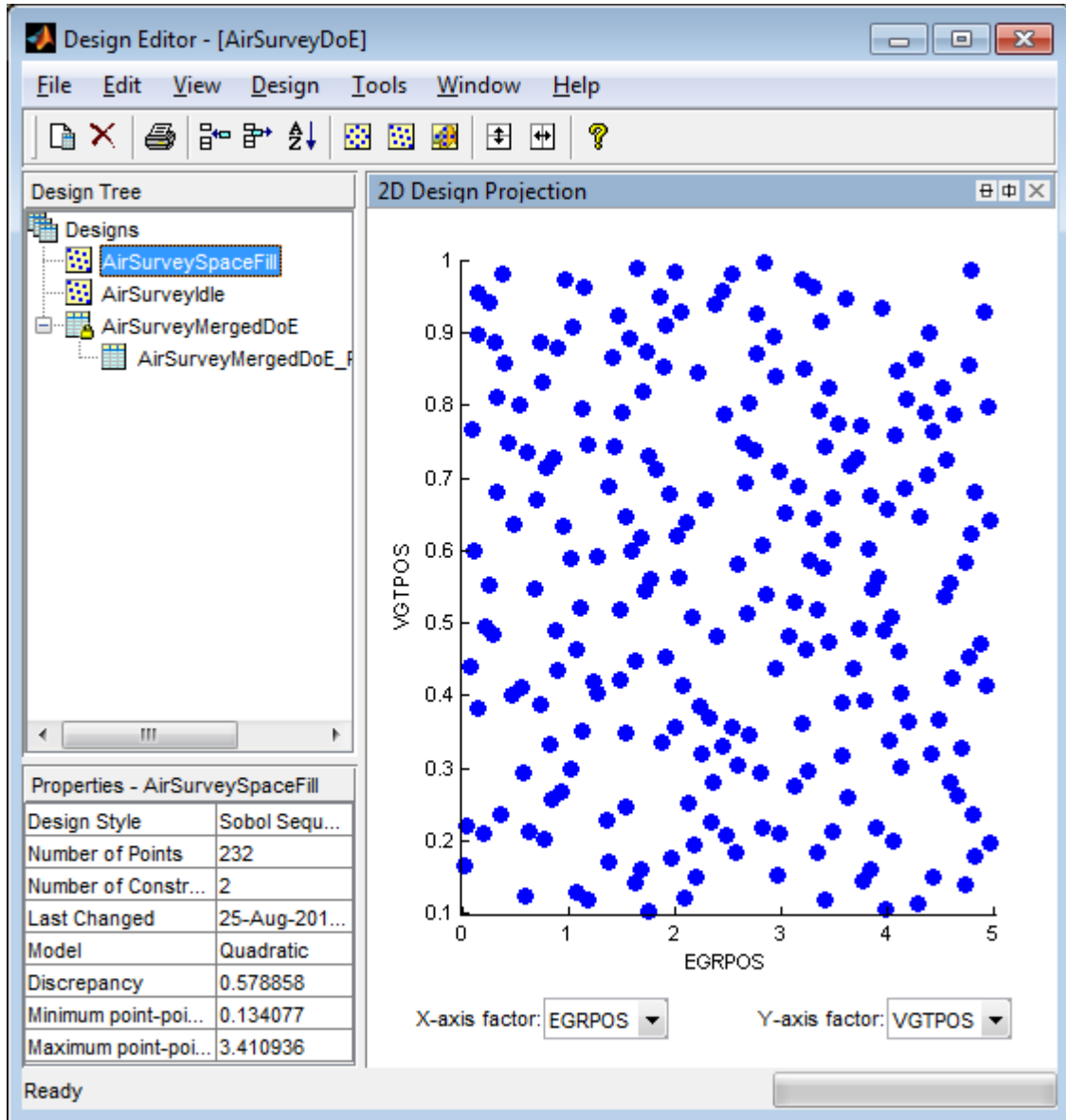


Close the dialog box.

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

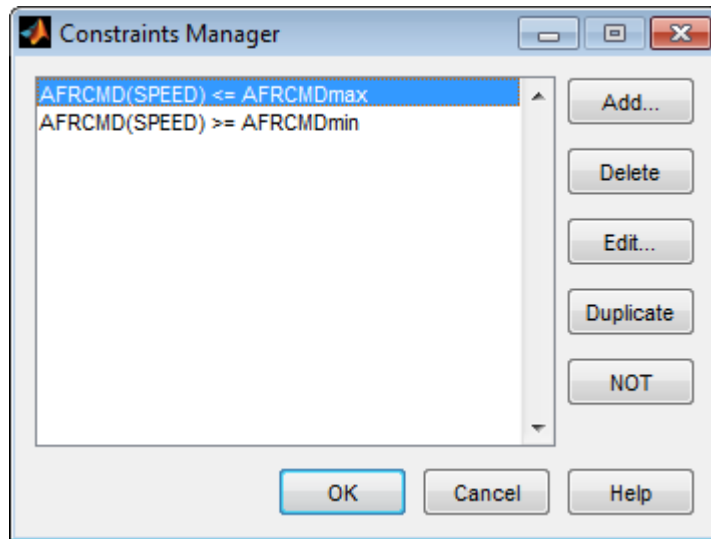
The Design Editor opens. Here, you can see how these designs are built:

- 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**.

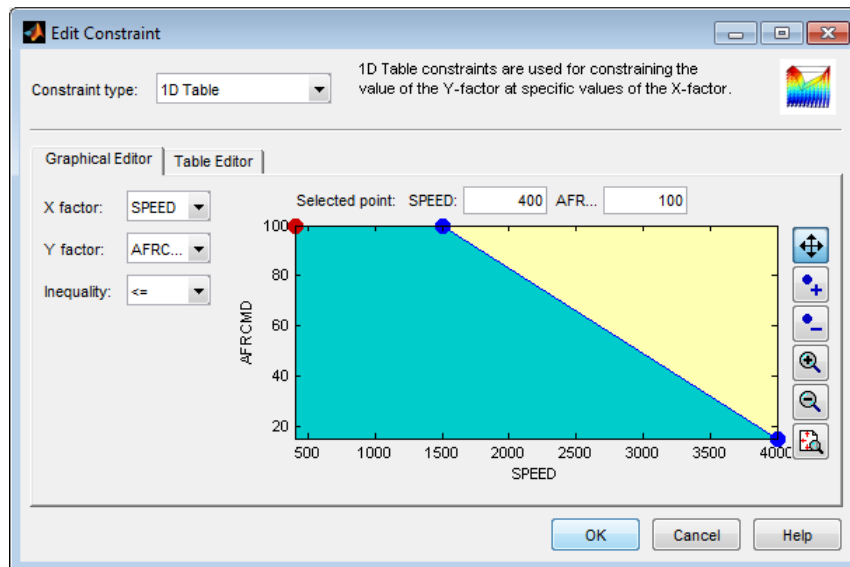


9 Select **Edit > Constraints** to see how the constraints are set up.



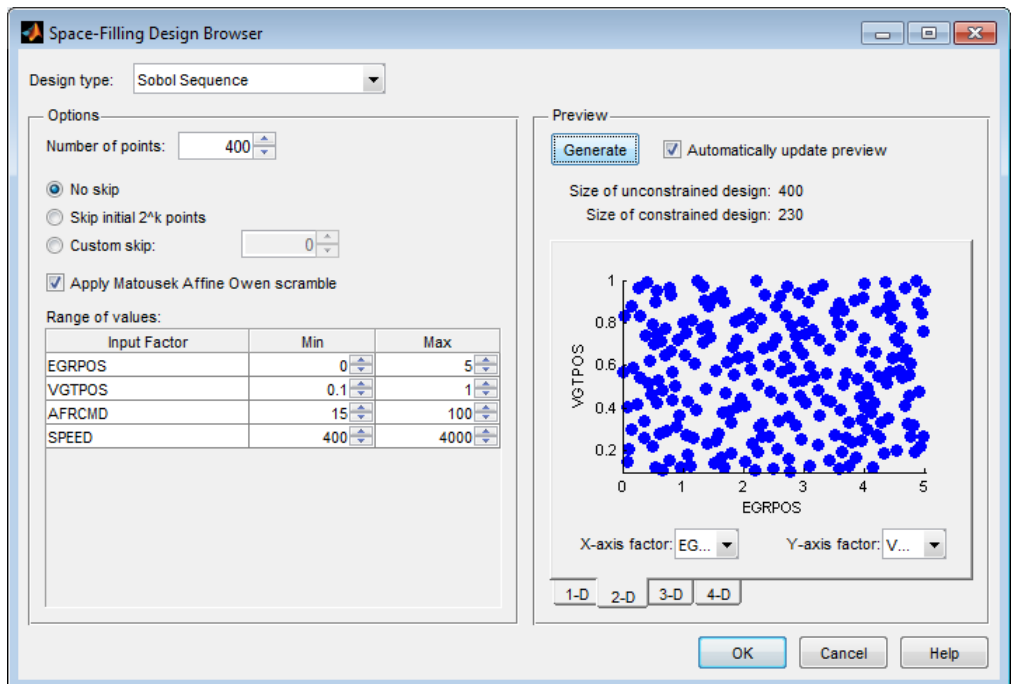


- 10 In the Constraints Manager dialog box, select a constraint and click **Edit**. Observe that you can define areas to exclude by dragging points or typing in the edit boxes.



Click **Cancel** to close the Edit Constraint dialog box and leave the constraint unchanged.

- 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**.
  - 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, you can select **Sobol Sequence** and **Number of points**, and view previews of design points. Click **Generate** several times to try different space-filling points.



- 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.

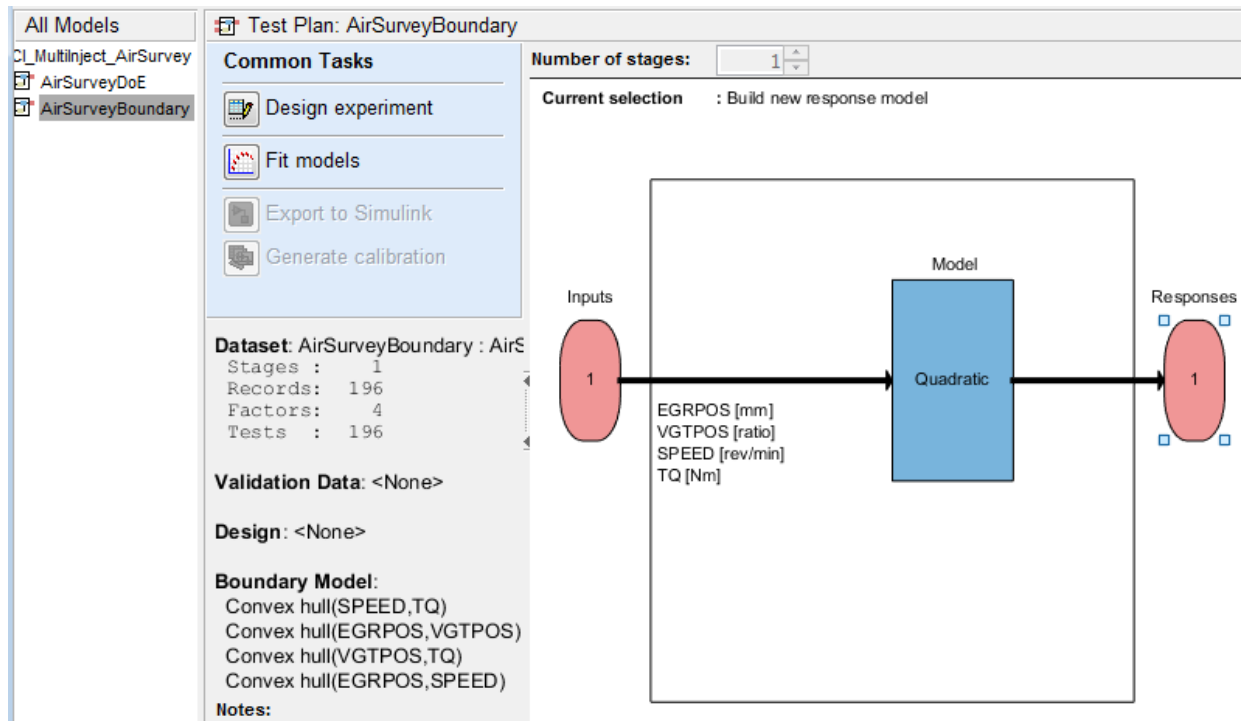
## Fit a Boundary Model to Air Survey Data

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.

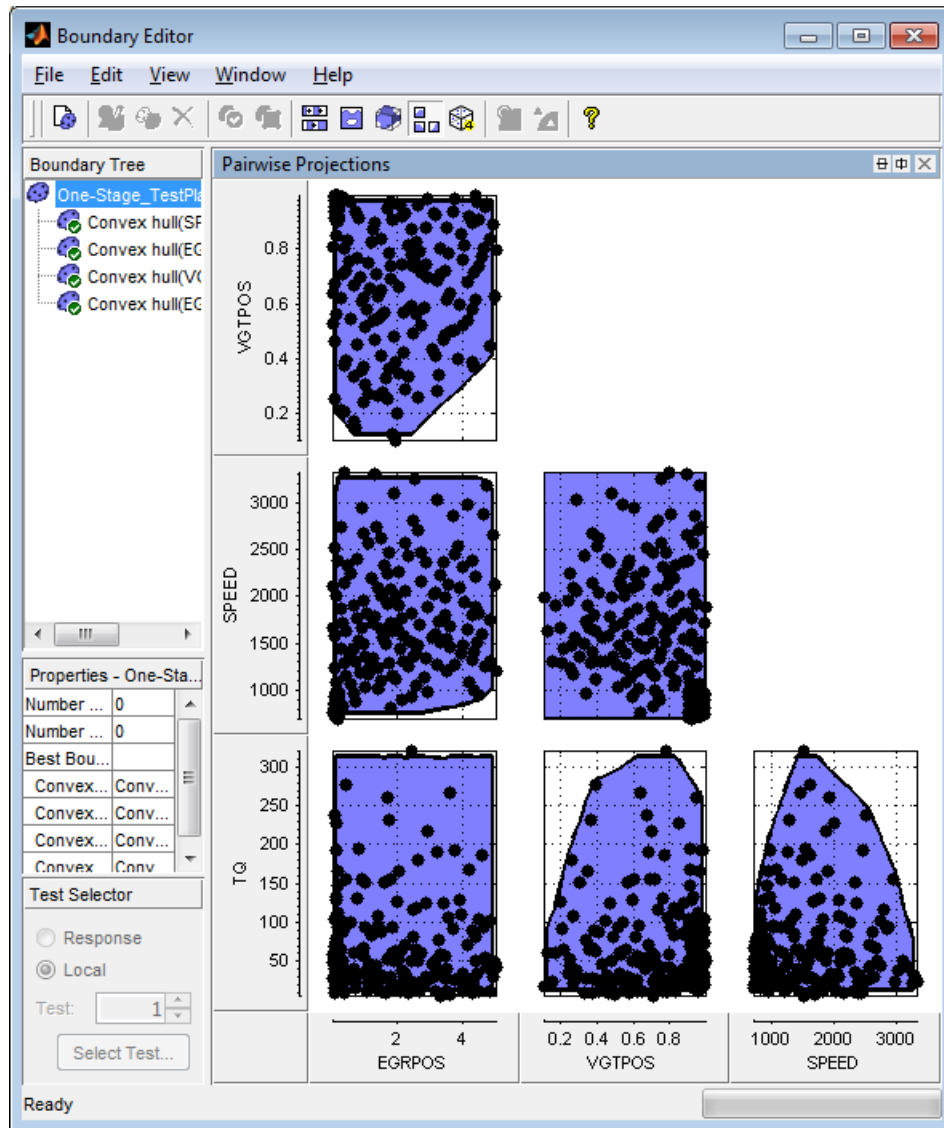
- 1 In the Model Browser, click the second test plan node in the **All Models** tree, `AirSurveyBoundary`.

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** > **Fit 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.

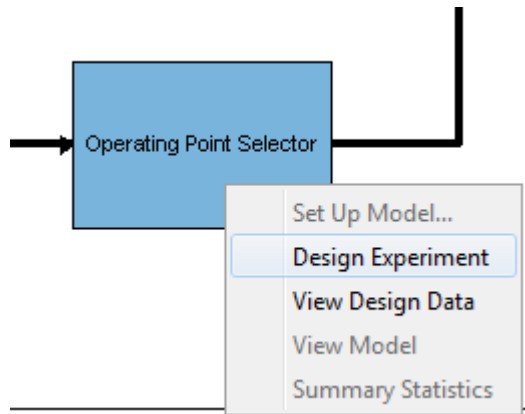
The final Point-by-Point Multi-Injection design (around 7000 points) was generated using a MATLAB script together with the Air Survey boundary model.

- 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.

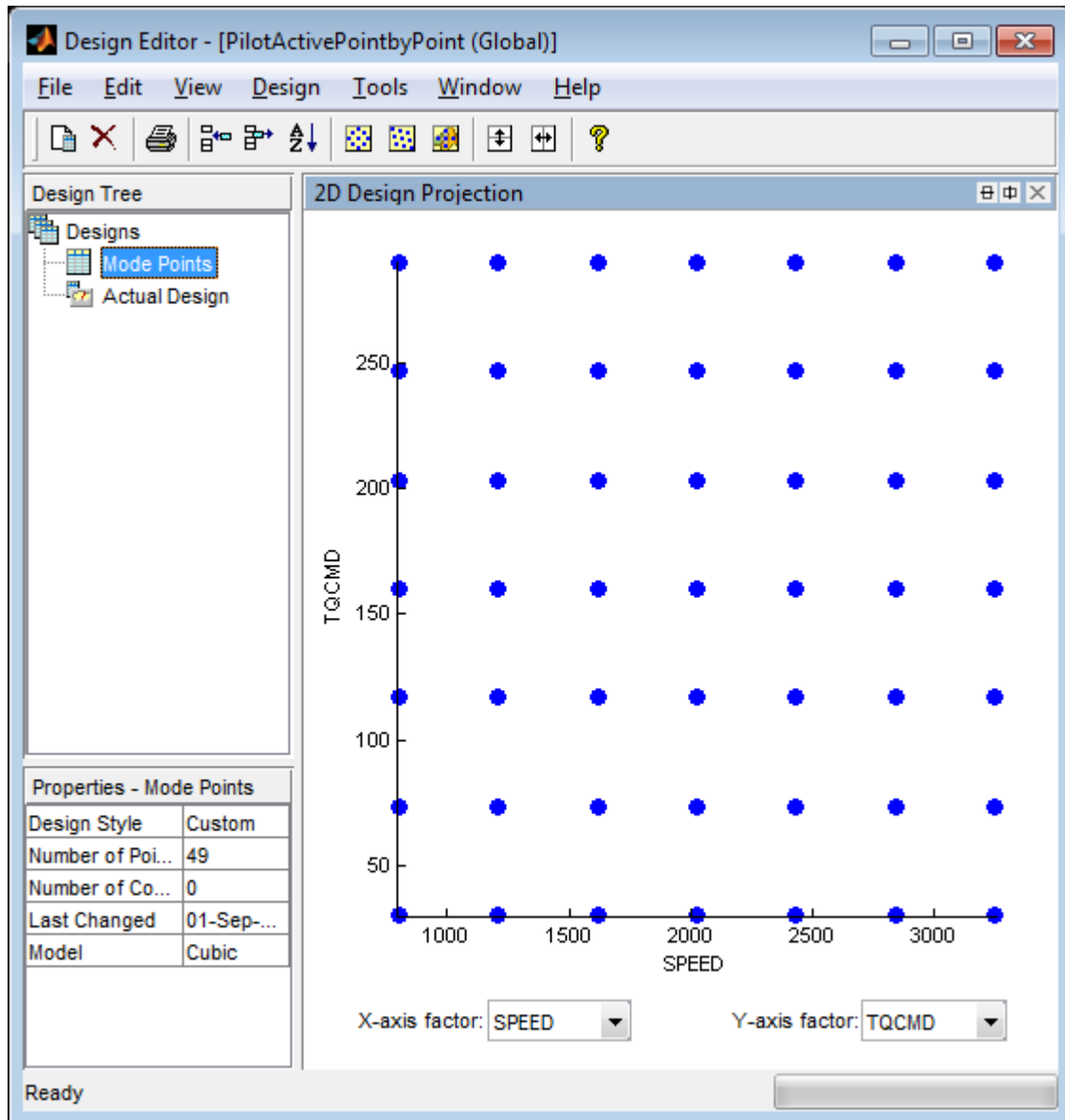
The script keeps only points that lie within the boundary model, and continues generating design points until it reaches the desired number of points. The Sobol space-filling design type keeps filling in spaces for good coverage of the design space.

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 first test plan view, then right-click the Operating Point Selector block, and select **Design Experiment** to view the designs created by the script.

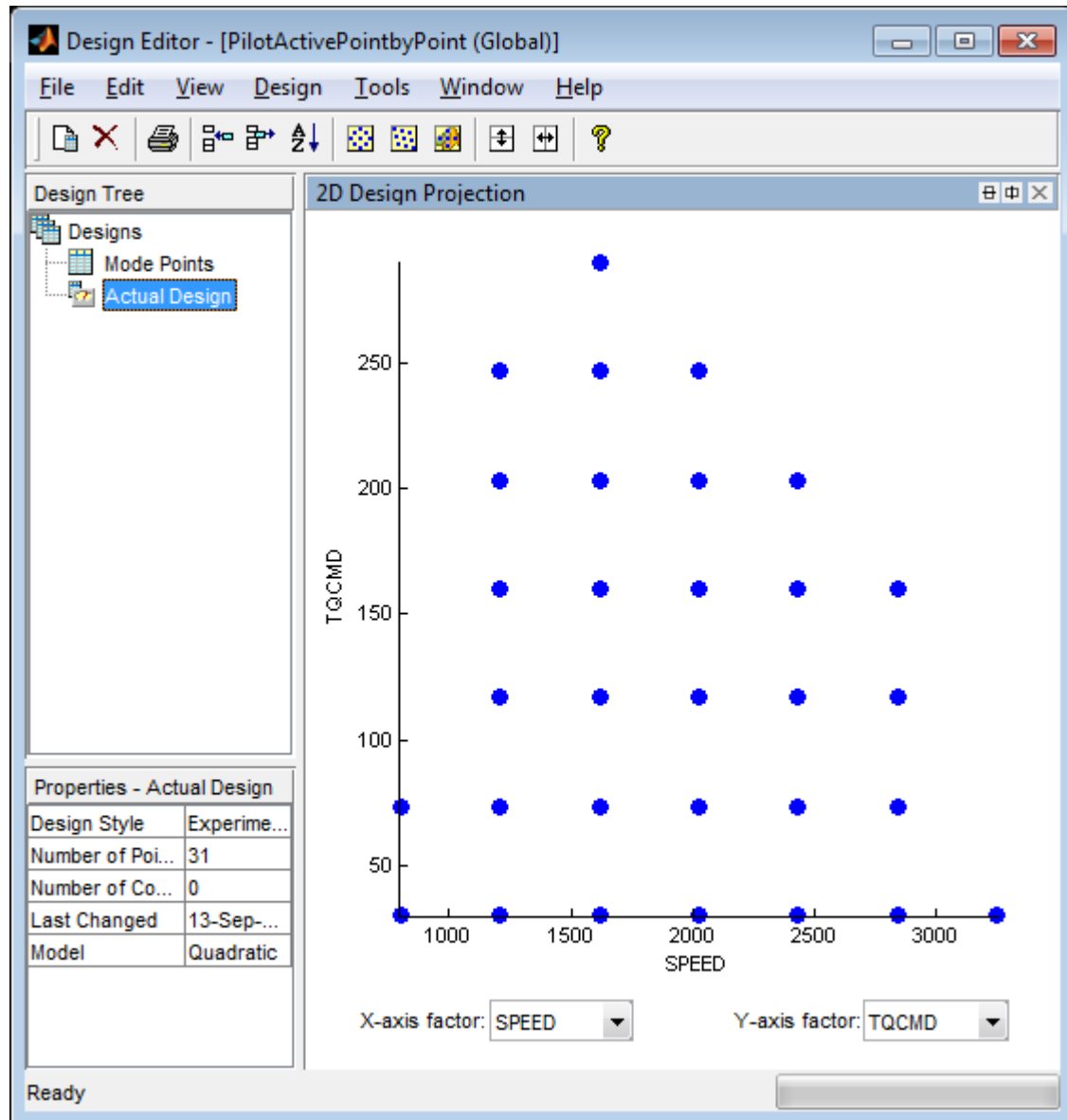


- 4 In the Design Editor, select **Mode Points** in the Design tree, and select **View > Current View > 2D Design Projection**.

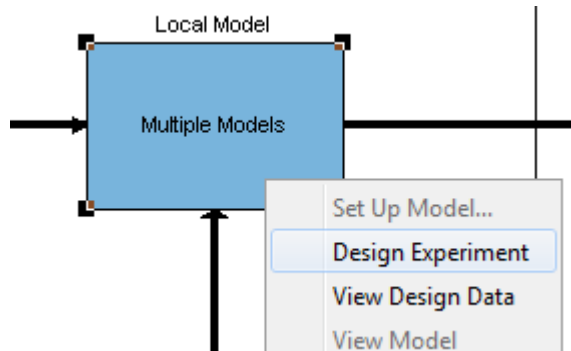




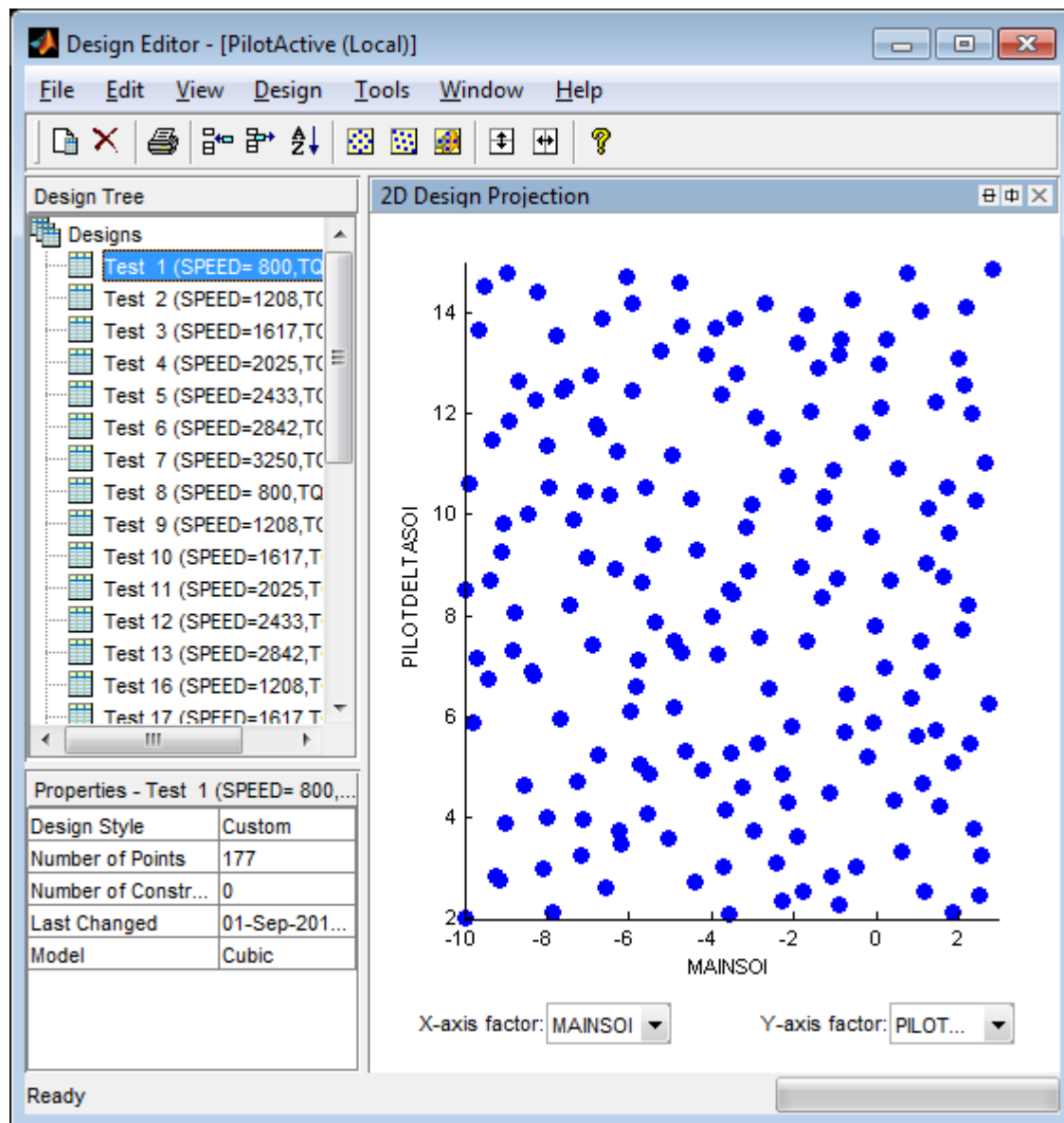
- 5 In the Design tree, select **Actual Design**. Observe that the boundary model constraint removed points in the corners, so that the actual design points collect data only in the feasible region determined by the initial air survey.



- Return to the Model Browser test plan, right-click the Local Model block, and select **Design Experiment**.

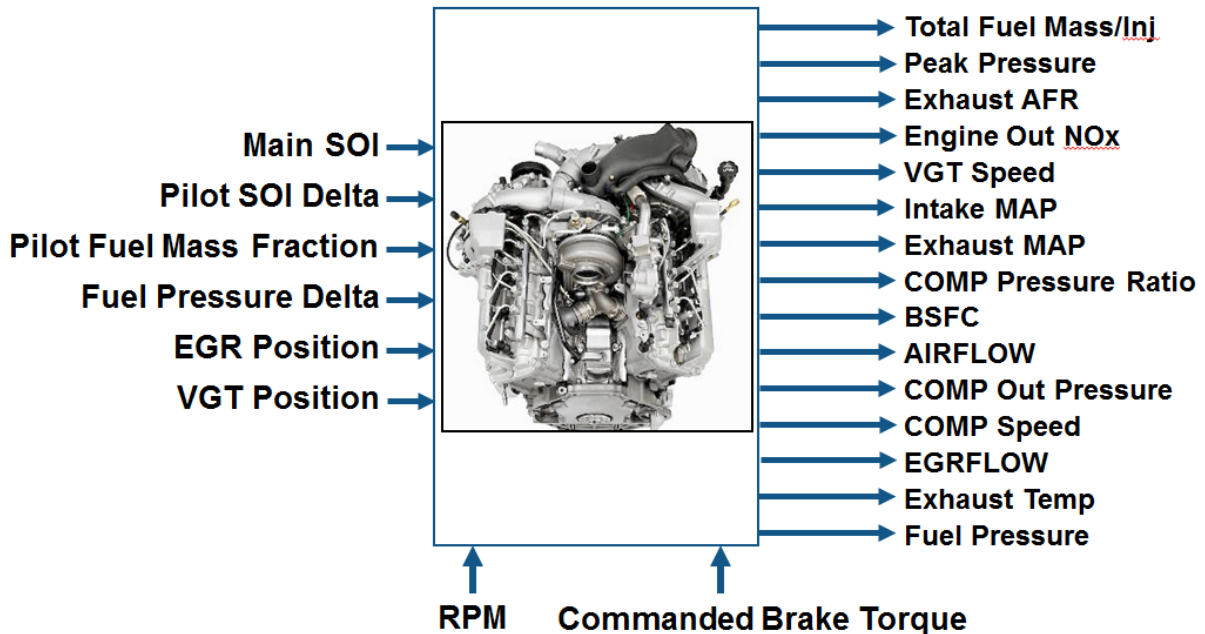


- Click test numbers to view the local design points at each operating point. At each test, the values of **SPEED** and **TQ** are fixed and the space-filling design selects points across the space of the local inputs.



## 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 7-10.

For next steps, see “Statistical Modeling” on page 7-40.

## Statistical Modeling

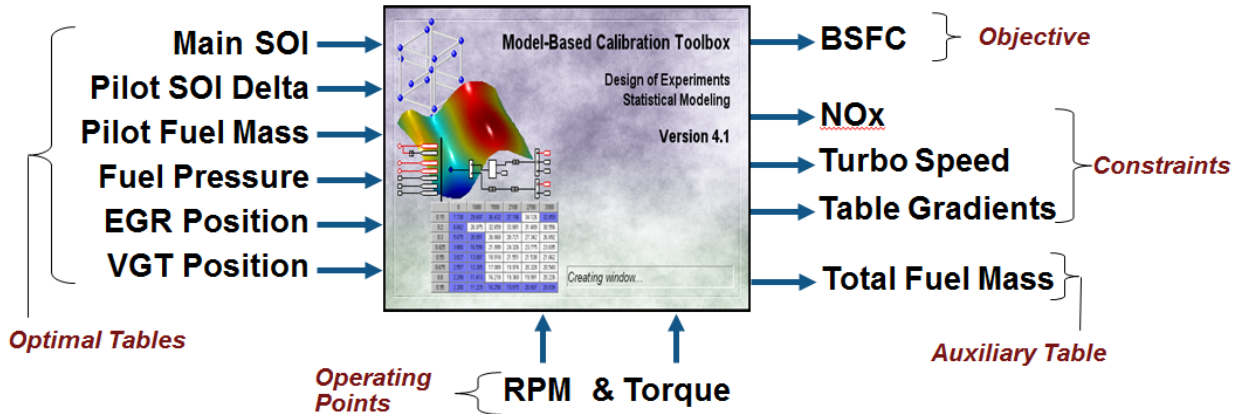
|   |
|---|
| <b>In this section...</b>                                       |
| “Examine the Test Plans for Point-by-Point Models” on page 7-40 |
| “Examine Response Models” on page 7-43                          |

### 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

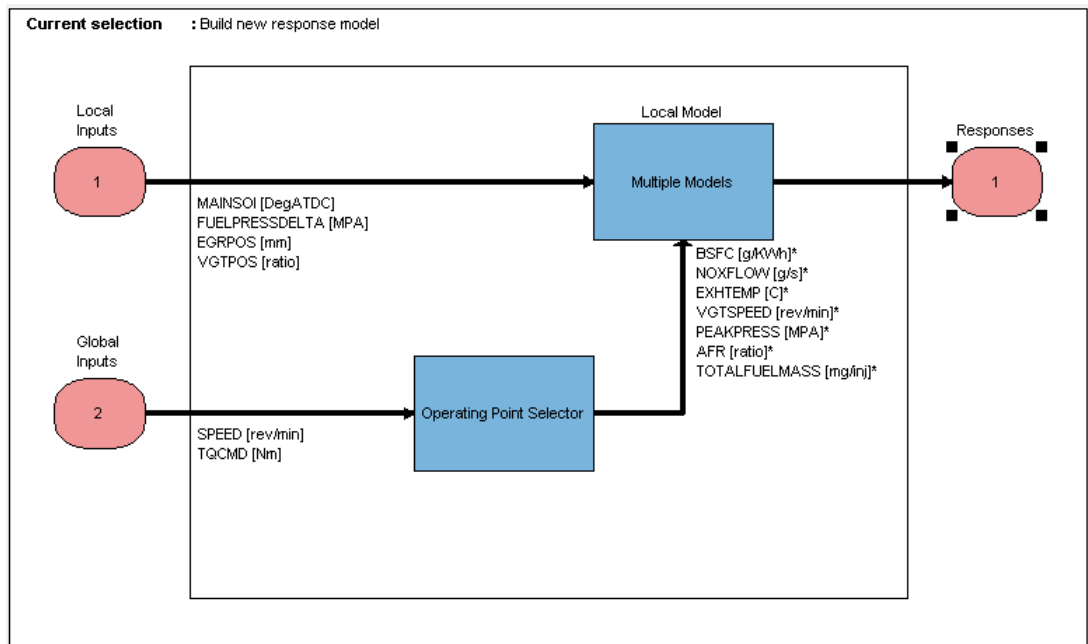
The toolbox provides the data for you to explore this calibration example. For details on how the data was collected, see “Data Collection and Physical Modeling” on page 7-10.

Examine the model setup.

- 1 Open MATLAB, and open the Model Browser by entering:

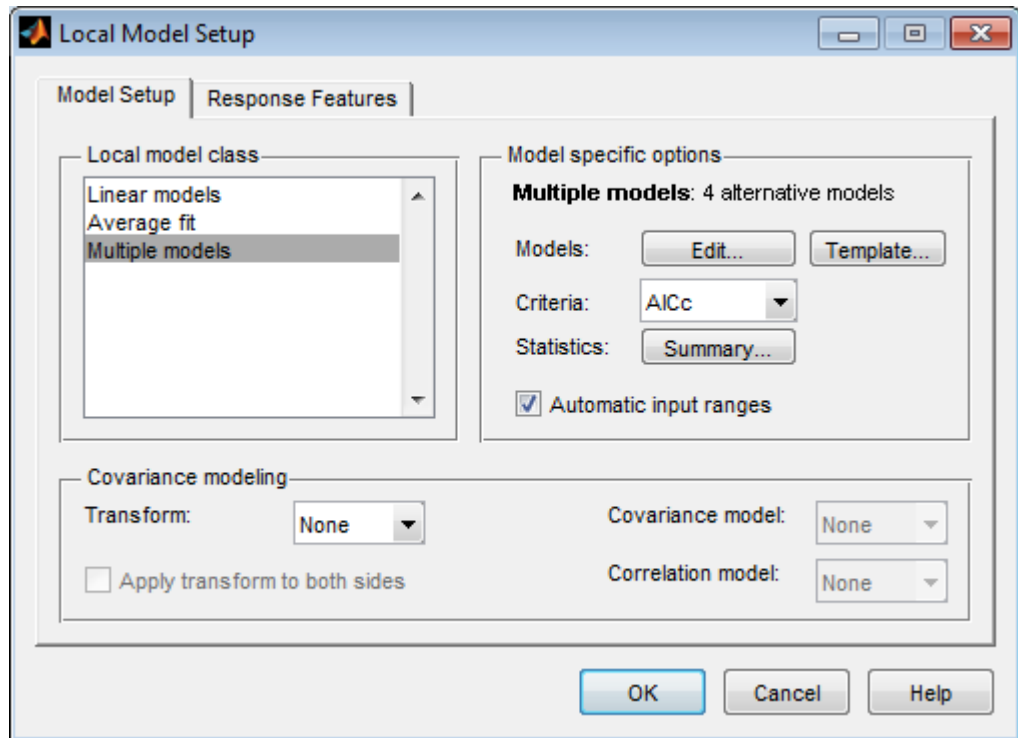
```
mbcmodel
```

- 2 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 test plan view appears.



- 4 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.
- 5 Double-click the **Inputs** blocks to view the ranges and names (symbols) for variables on the Input Factor Set Up dialog box.
- 6 Double-click the **Local Model** block to view that the **Local model class** is **Multiple models** 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 sets the local model type to build multiple 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 AICc.





- 7 Click **Edit** to view the alternative models to fit at each operating point.
- 8 Click **Cancel** twice to close the Model Setup dialog boxes 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”.

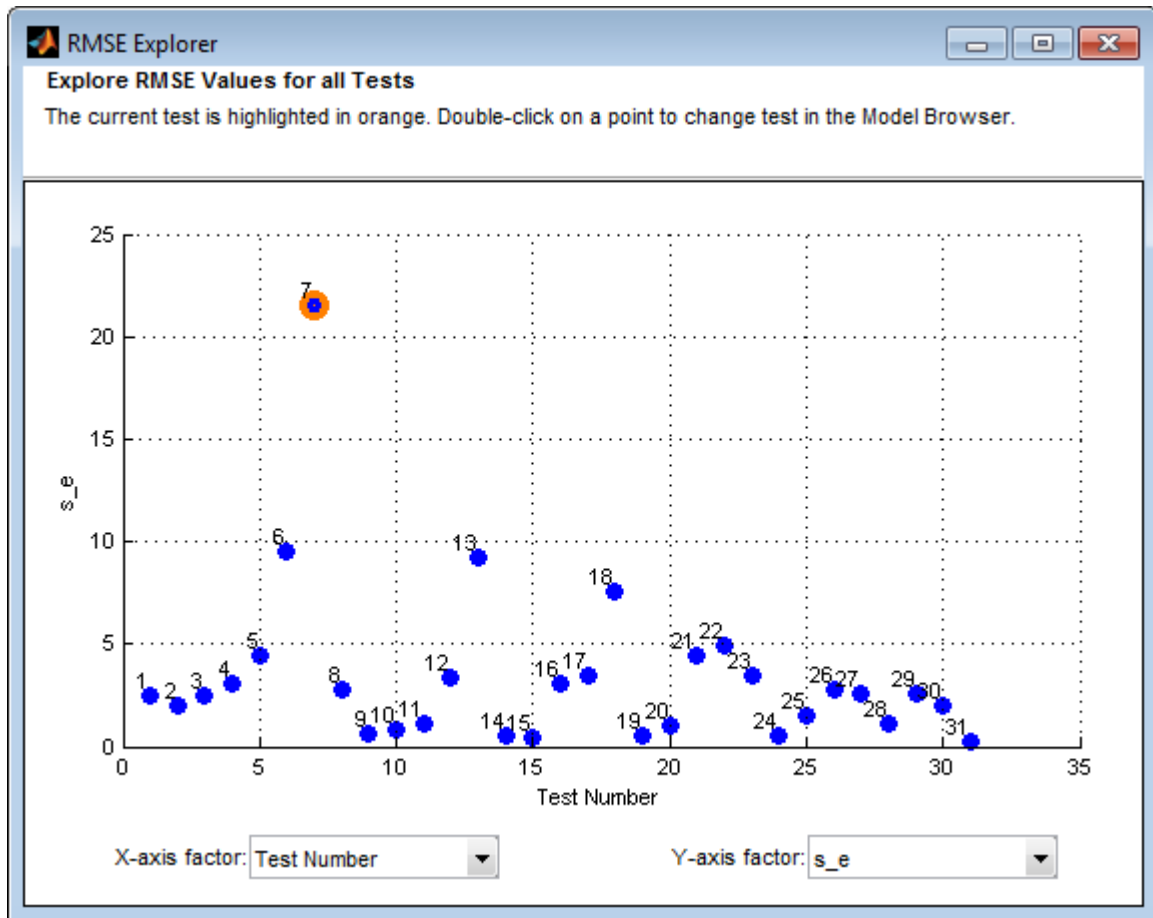
## Examine Response Models

- 1 Expand the `PilotInactivePointByPoint` test plan node in the **All Models** tree and select **Multiple Models** nodes under each response name.

- 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.

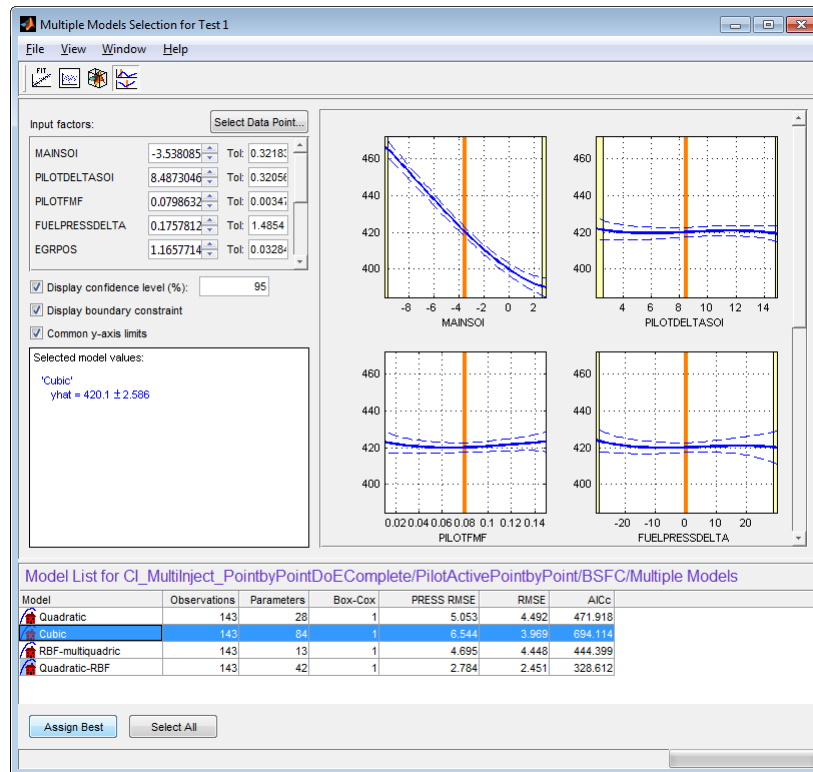
- Select **View > RMSE Plots** (or use the toolbar button) to open the RMSE Explorer. These plots can help you identify problem tests to investigate,



- Choose a model to use at a particular operating point by selecting **Model > Utilities > Select Local Model** to open the Model Selection window for the selected test. All

alternative models are refitted at this stage (as only the best model is stored) so this refitting can take time.

In the Model Selection window you can compare the fit of the local model types using various plot types and statistics, and select which model type to use for the selected test.



For details on analyzing point-by-point models, see “Analyze Point-by-Point Models”.

For next steps, see “Optimization” on page 7-46.

## Optimization

### In this section...

“Optimization Overview” on page 7-46

“Set Up Models and Tables for Optimization” on page 7-46

“Examine the Point Optimization Setup” on page 7-49

“Examine the Point Optimization Results” on page 7-53

“Create Sum Optimization from Point Optimization” on page 7-55

“Fill Tables from Optimization Results” on page 7-61

“Examine the Multiobjective Optimization” on page 7-71

### 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 NO<sub>x</sub>
  - Constrain turbocharger speed
  - Constrain smoothness of tables
- 3 Fill lookup tables for all control inputs.

### Set Up Models and Tables for Optimization

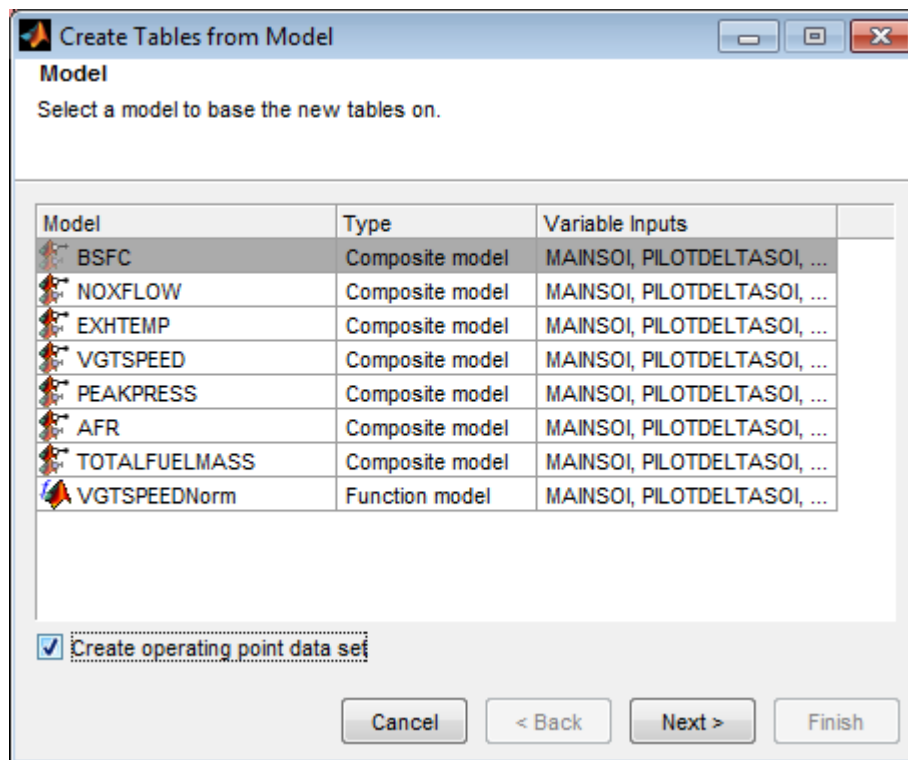
To perform an optimization, you need to import the statistical models created earlier in the Model Browser.

- 1 Open MATLAB, and open the CAGE Browser by entering:  
cage
- 2 To see how to import models, 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

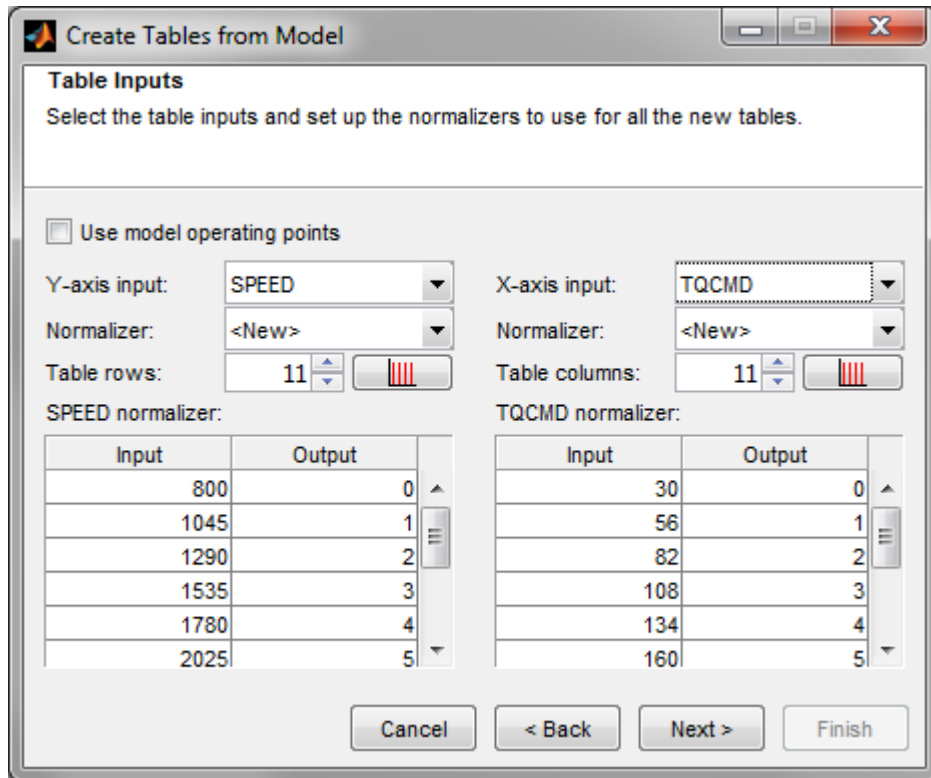
toolbox provides an example file with the models already imported, so click **Close** to close the CAGE Import Tool.

- 3 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.

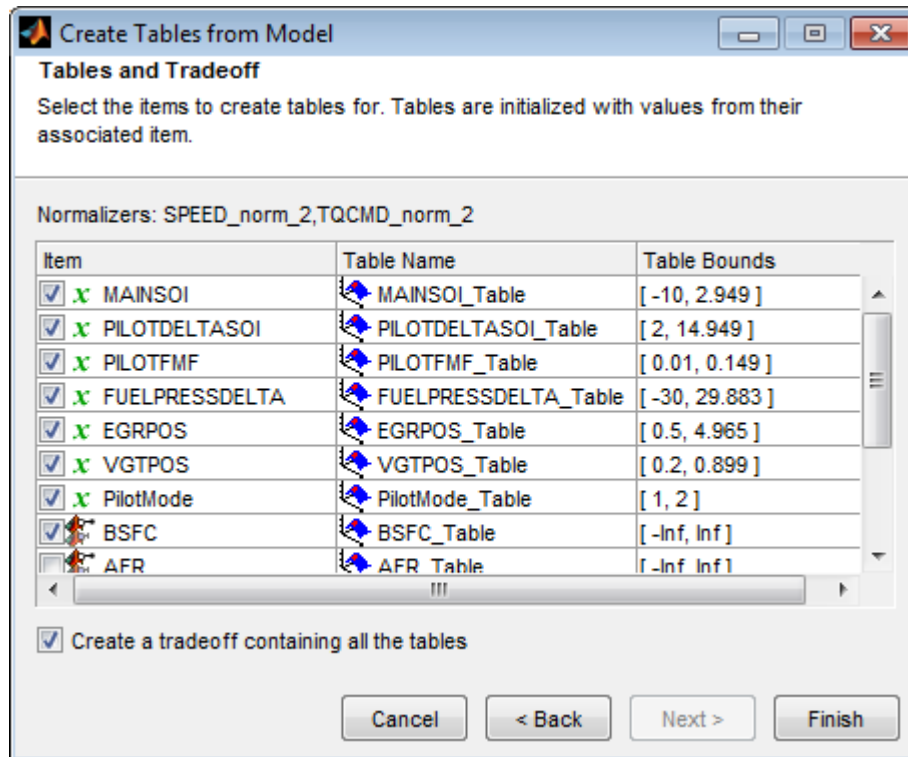
- b Select SPEED and TQCMD for the axis inputs.
- c Enter 11 for the table rows and columns.



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.

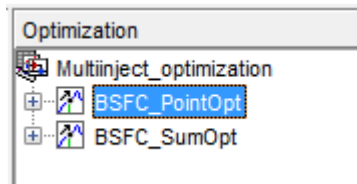


- 9 Select the **Tables** view to see all the tables. Observe there are tables named `_wPilot` and `_noPilot`, that will be filled by optimization results. The goal is to fill separate tables for each mode, pilot injection active and no pilot injection.

## 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 **Data Objects** pane to view the two optimizations.



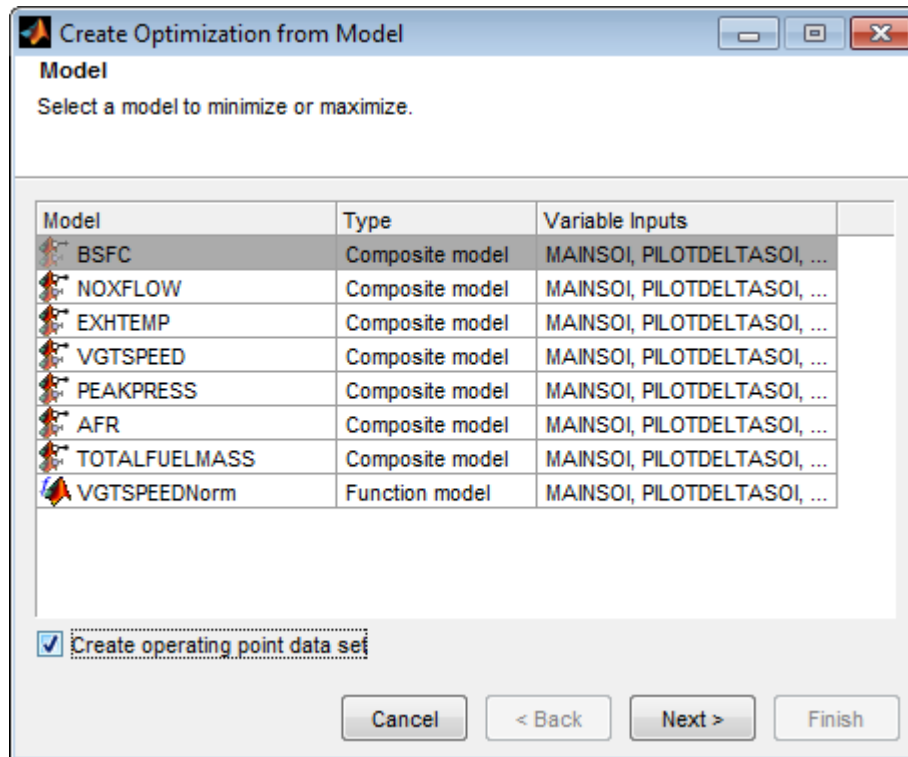
Why are there two optimizations? To complete the calibration, you need to start with a point optimization to determine whether to use Pilot Injection at each operating point. You use the results of this point optimization to set up the sum optimization to optimize fuel consumption over the drive cycle,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

- Constrain total NO<sub>x</sub>
  - 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.

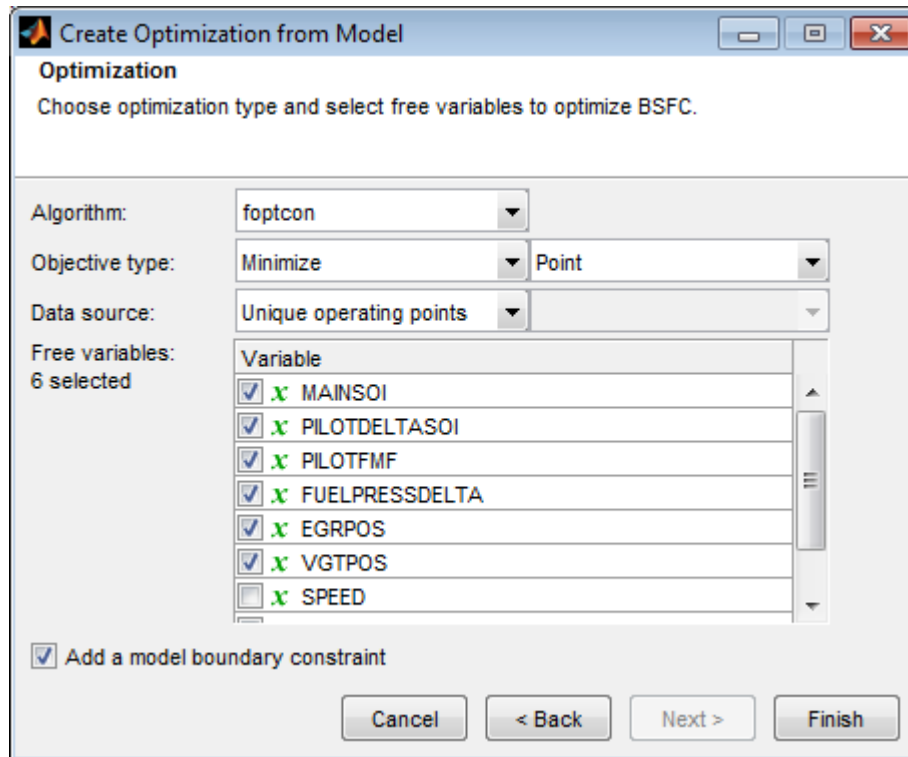
| Objectives  |   |          |
|-------------|---|----------|
| Name        | Description   | Type     |
| 🔥 BSFC      | BSFC(MAINSOI, PILOTDELTAOI, PILOTFMF, FUELPRESSDELTA, EGRPOS, VGTPOS, SPEED, TQCMD, PilotMode)      | Minimize |
| 🔥 PEAKPRESS | PEAKPRESS(MAINSOI, PILOTDELTAOI, PILOTFMF, FUELPRESSDELTA, EGRPOS, VGTPOS, SPEED, TQCMD, PilotMode) | Minimize |

- 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**.

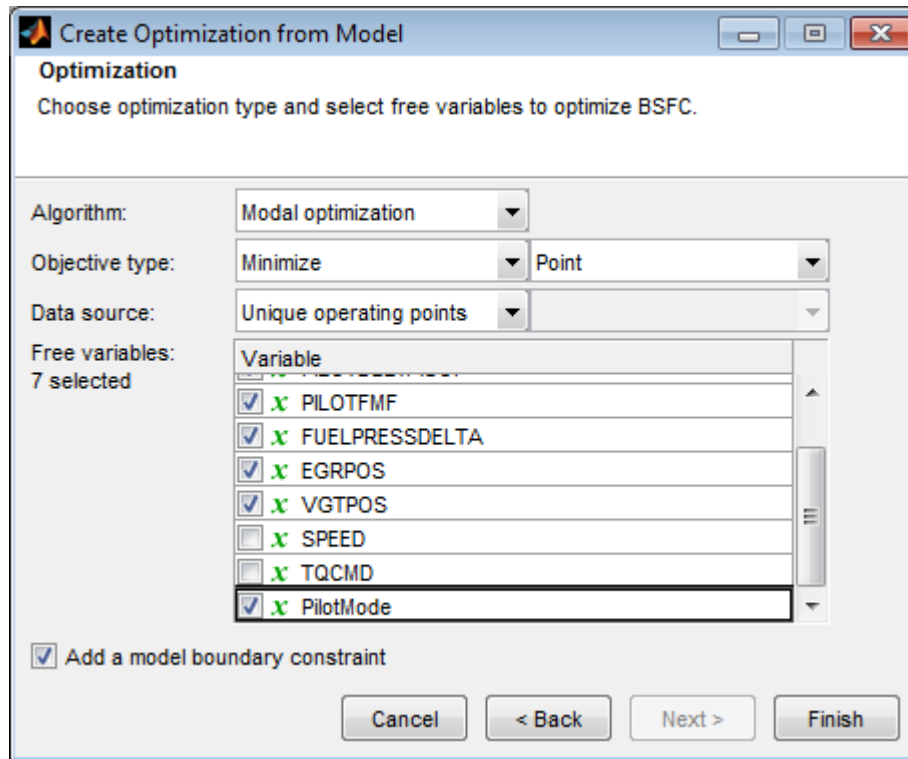




- 6 Select the model **BSFC** and the **Create operating point data set** check box, and click **Next**.
- 7 Observe that the default settings will create a point optimization to minimize **BSFC** at the model operating points, using six free variables and not **SPEED**, **TQCMD**, or **PilotMode**, 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**.



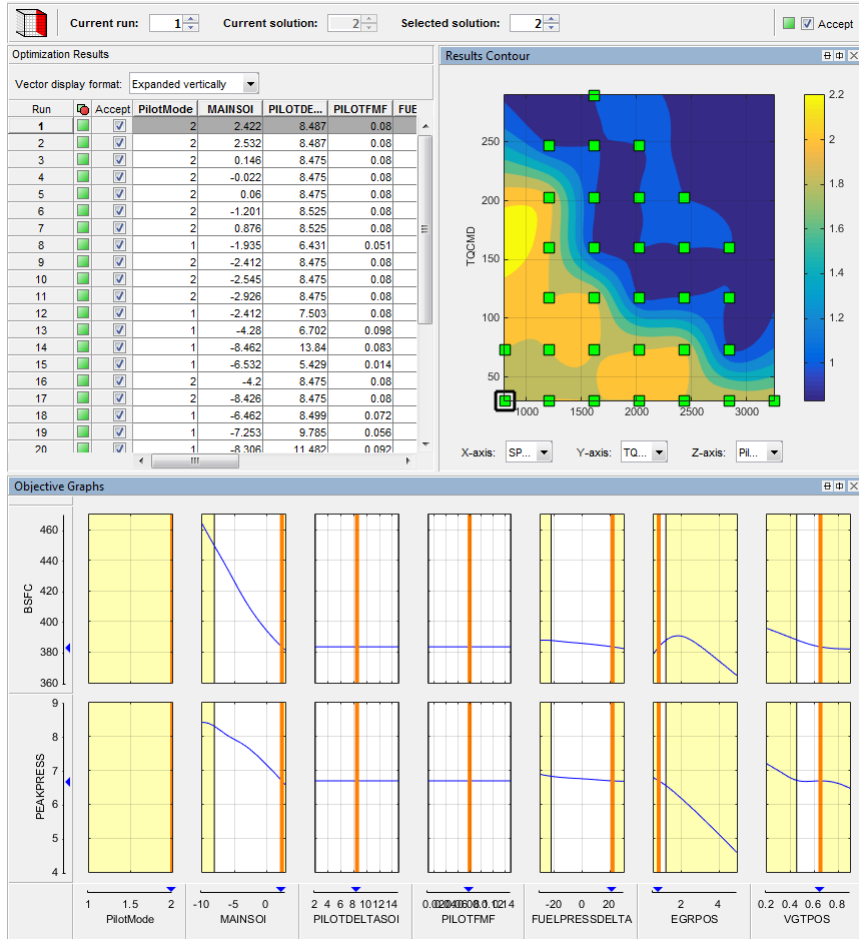
- 10 Click **Finish** to create the new optimization.
- 11 Compare your new optimization with the example BSFC\_PointOpt. Notice the example has a second objective, to minimize PEAKPRESS. To see how to set this up, right-click the Objectives pane and select **Add Objective**.

## Examine the Point Optimization Results

The example file CI\_MultiInject.cag shows the results of the multistage process to finish this calibration.

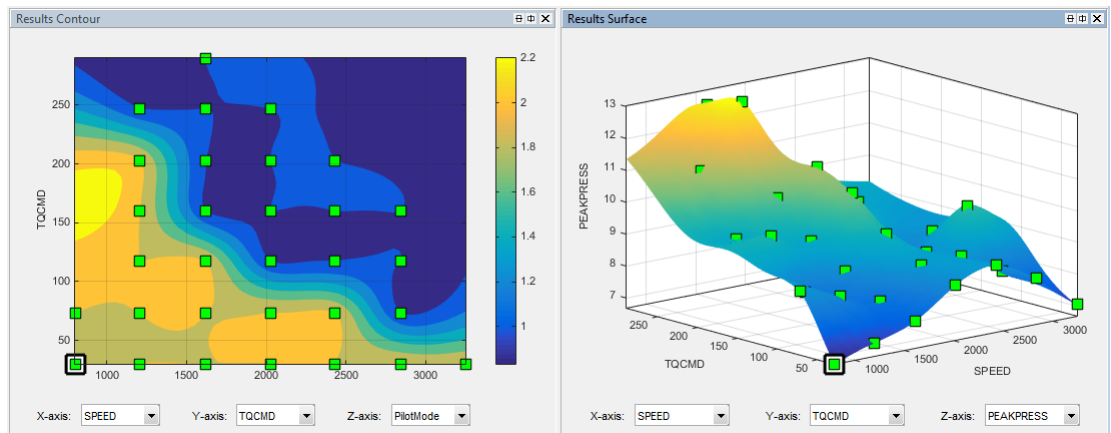
- 1 Expand the first optimization node, BSFC\_PointOpt, and select the output node underneath, BSFC\_PointOpt\_output.
- 2 Select **View > Selected Solution** (or the toolbar button) to view which pilot mode the optimization selected as best at each operating point. You can see the selected

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 (foptcon algorithm) and changes the `Mode Variable` to a fixed variable.

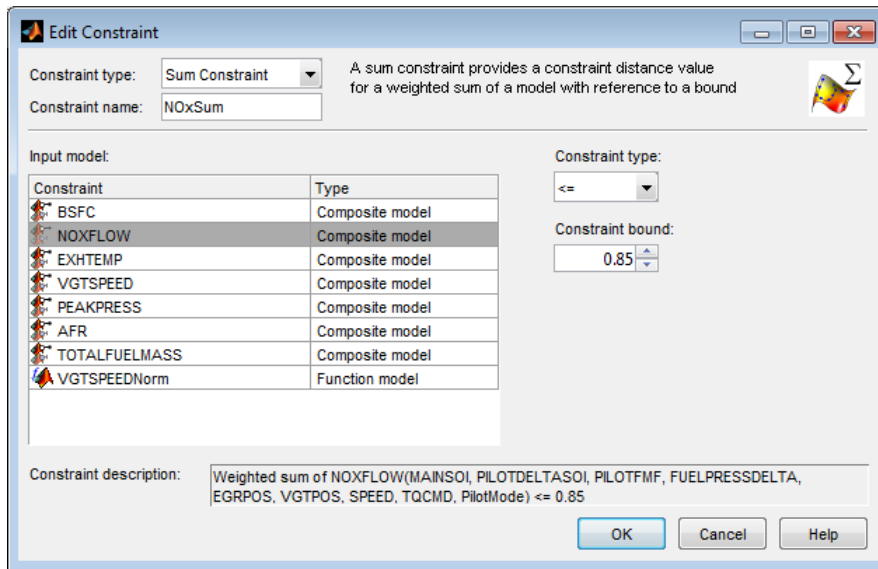
- 2 Compare your new optimization with the example sum optimization, `BSFC_SumOpt`. The example shows you need to set up more constraints to complete the calibration.

| 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)   |        |

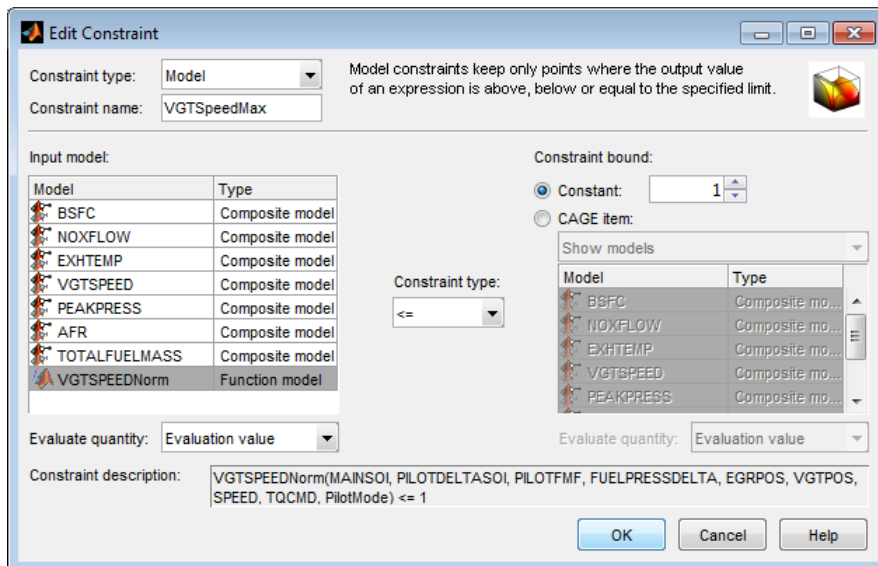
The additional constraints make the optimization meet these requirements:

- 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.

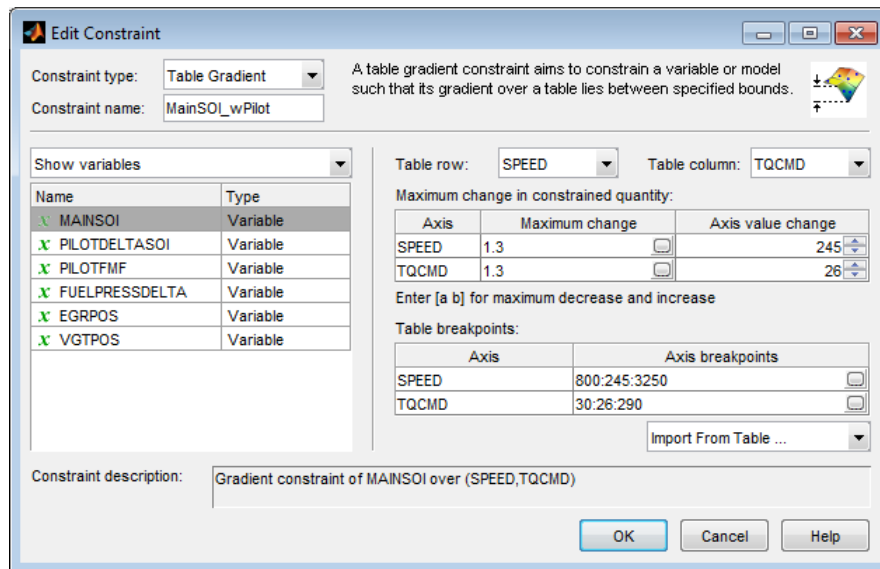


This constraint controls maximum turbocharger speed.



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.



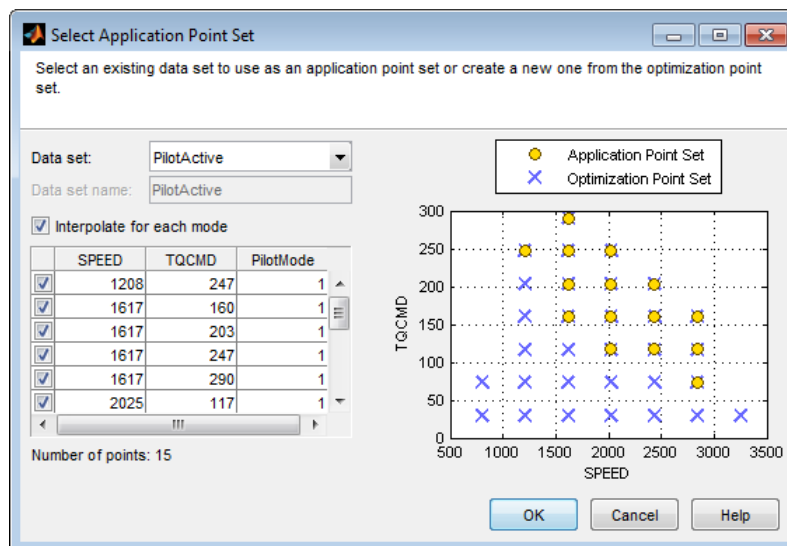
5 In the Optimization view, in the Constraints pane, observe that:

- 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.



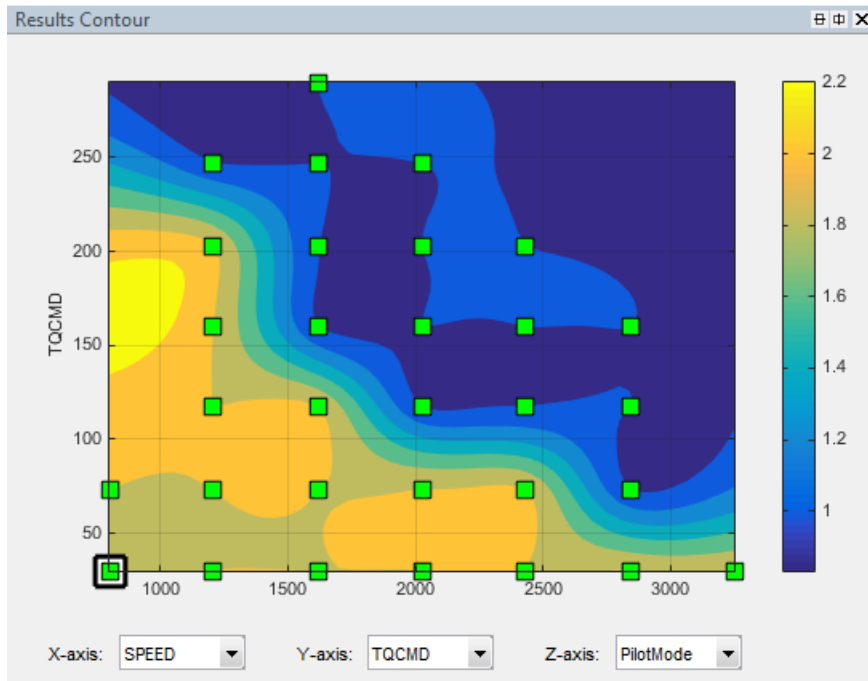
| Constraints            |                                    |                           |    |
|------------------------|------------------------------------|---------------------------|----|
| Name                   | Description                        | Application Point Set     | St |
| BSFC_Boundary          | Boundary constraint of BSFC(M...   |                           |    |
| NOxSum                 | Weighted sum of NOXFLOW(M...       |                           |    |
| VGTSpeedMax            | VGTSPEEDNorm(MAINSOI, PILO...      |                           |    |
| MainSOI_wPilot         | Gradient constraint of MAINSOI ... | PilotActive(SPEED,TQC...  |    |
| MainSOI_noPilot        | Gradient constraint of MAINSOI ... | PilotInactive(SPEED,TQ... |    |
| EGRPos_wPilot          | Gradient constraint of EGRPOS ...  | PilotActive(SPEED,TQC...  |    |
| EGRPos_noPilot         | Gradient constraint of EGRPOS ...  | PilotInactive(SPEED,TQ... |    |
| FuelPressDelta_w...    | Gradient constraint of FUELPRE...  | PilotActive(SPEED,TQC...  |    |
| FuelPressDelta_n...    | Gradient constraint of FUELPRE...  | PilotInactive(SPEED,TQ... |    |
| VGTPos_wPilot          | Gradient constraint of VGTPOS ...  | PilotActive(SPEED,TQC...  |    |
| VGTPos_noPilot         | Gradient constraint of VGTPOS ...  | PilotInactive(SPEED,TQ... |    |
| PilotActiveSOI_wPilot  | Gradient constraint of PILOTPE...  | PilotActive(SPEED,TQC...  |    |
| PilotActiveSOI_noPilot | Gradient constraint of PILOTPE...  | PilotInactive(SPEED,TQ... |    |

- 6 Right-click the table gradient constraint **MainSOI\_wPilot** and click **Select Application Point Set** to see how these are set up.

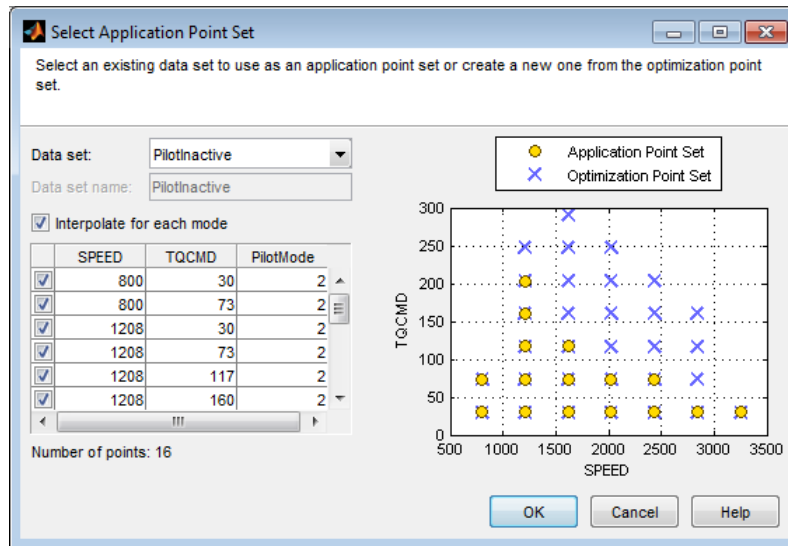


- 7 Observe the **Interpolate for each mode** option is selected. This application point set restricts the table gradient constraint to **PilotMode 1** (active) points only.

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.

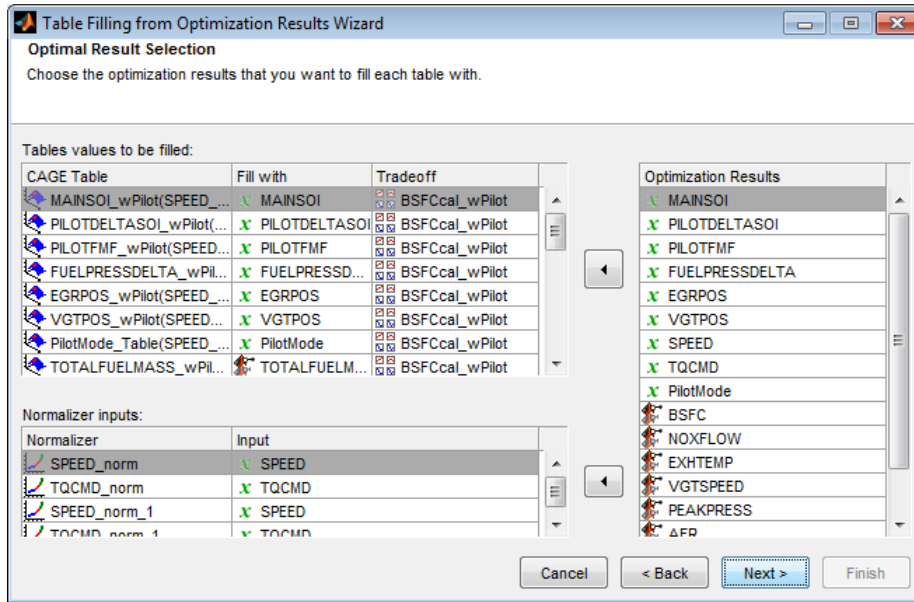


Your sum optimization now contains all the required constraints and is ready to run. Next, view the results in the example sum optimization.

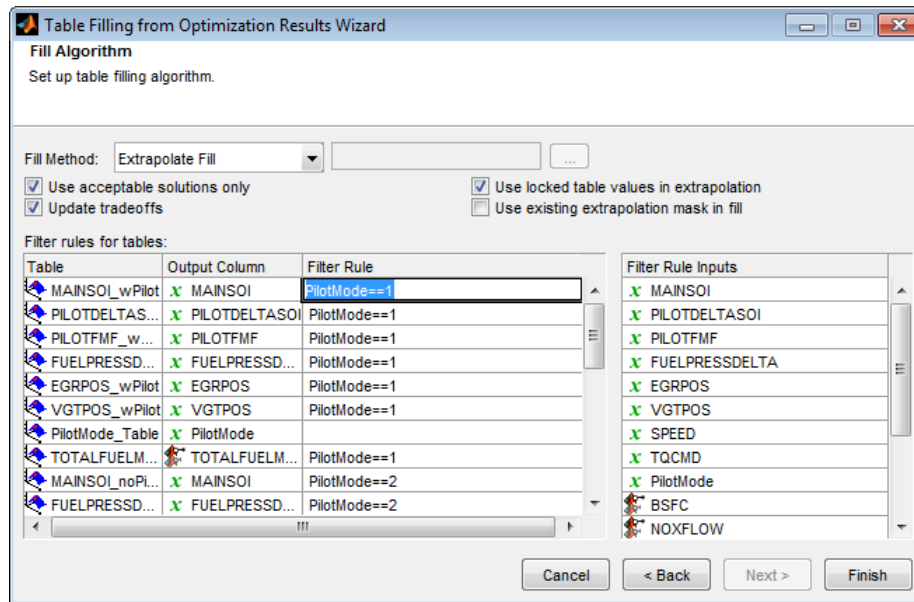
## Fill Tables from Optimization Results

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**.

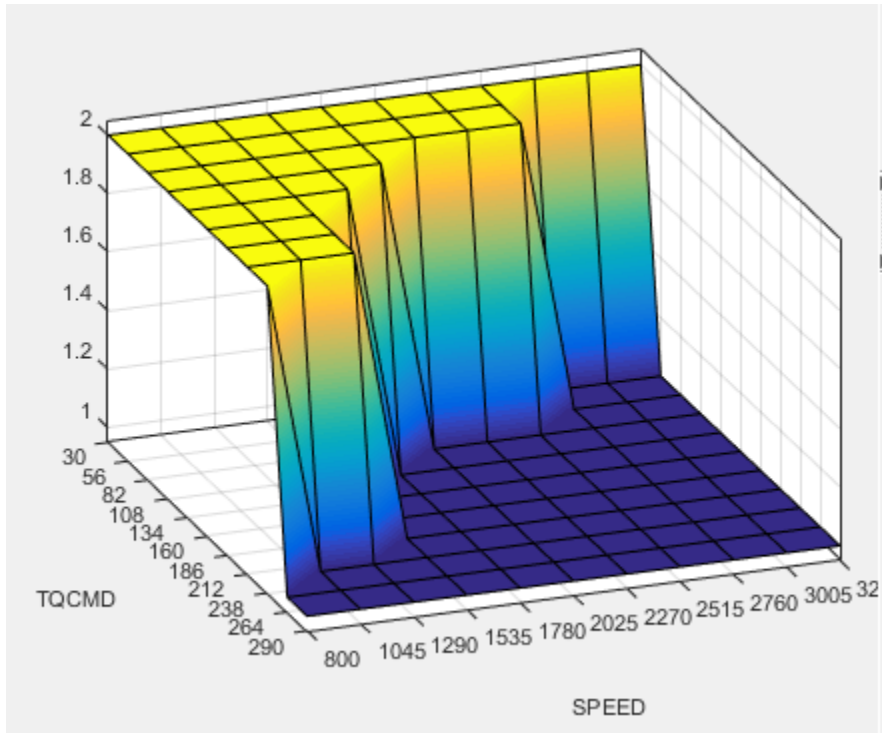


- On the third screen, observe how some tables have a **Filter Rule** set up so that they are filled only with results where PilotMode is 1 or 2. You can create a filter rule like this by entering PilotMode==1 in the **Filter Rule** column.



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.

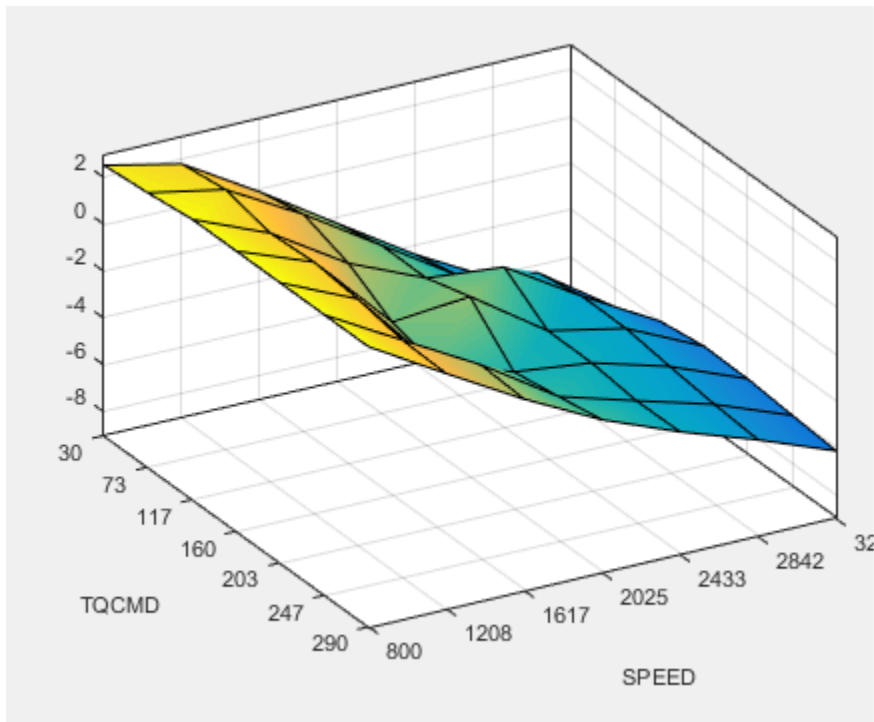


### Pilot Mode Table

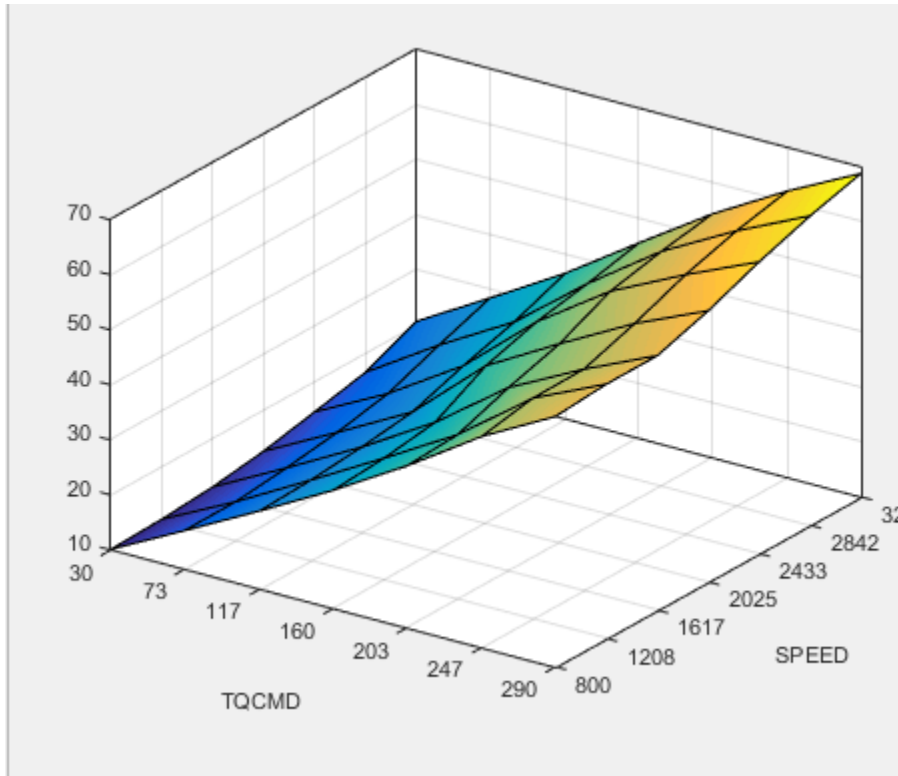
This graphic shows the plot of the table to select the active or inactive pilot mode depending on the speed and commanded torque

You need to fill calibration tables for each control variable described in “Multi-Injection Diesel Problem Definition” on page 7-2, in both pilot modes, active and inactive.

Following are all the pilot active tables.

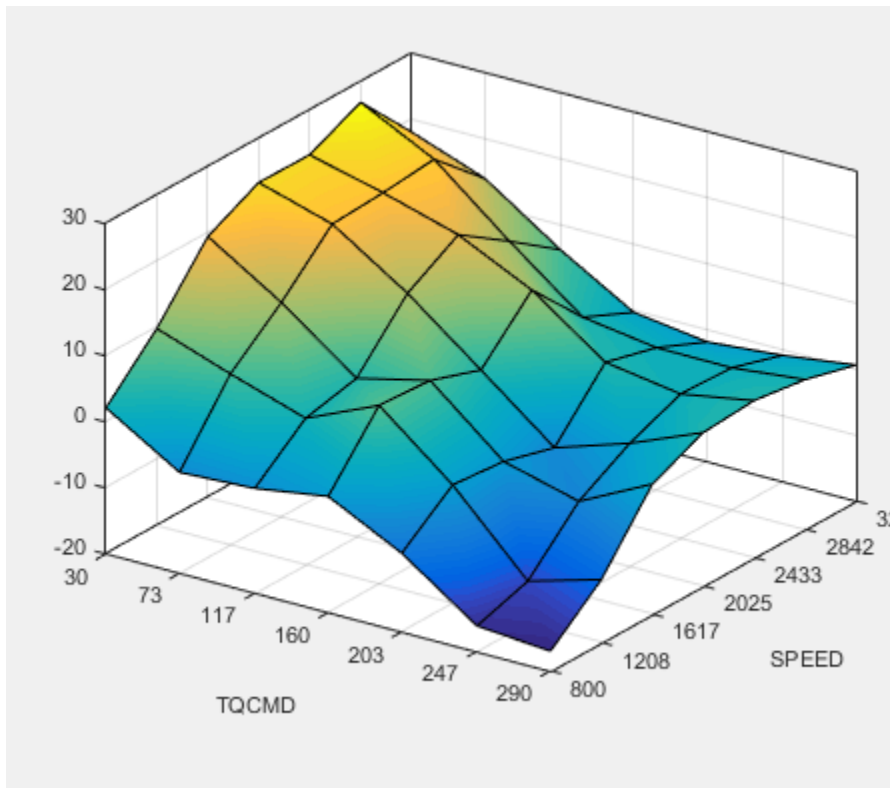


**Main Injection Timing (SOI) Table**

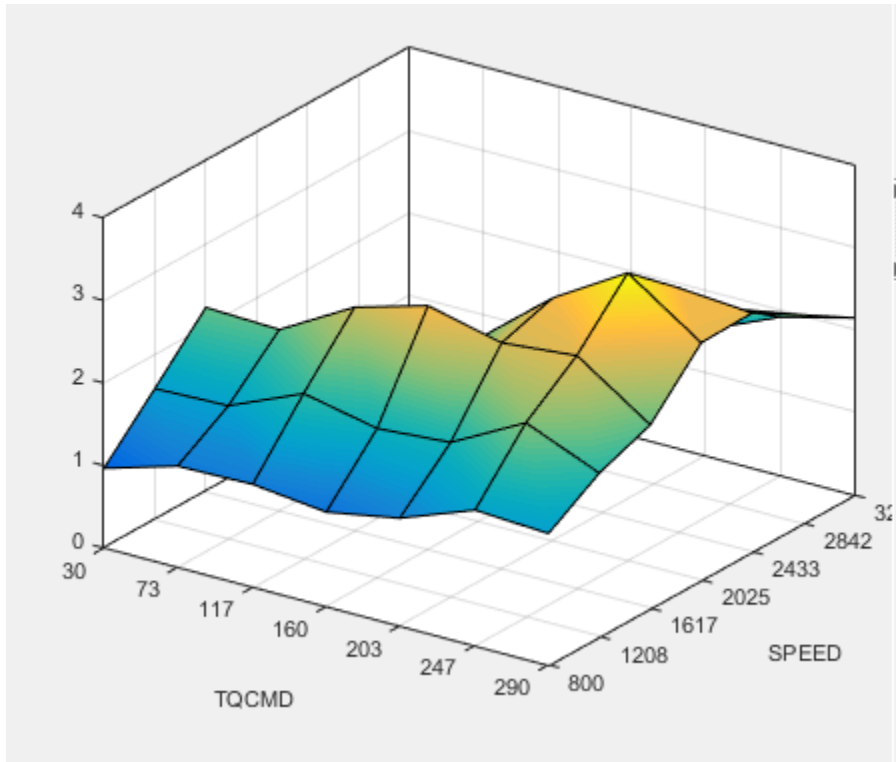


Total Injected Fuel Mass Table

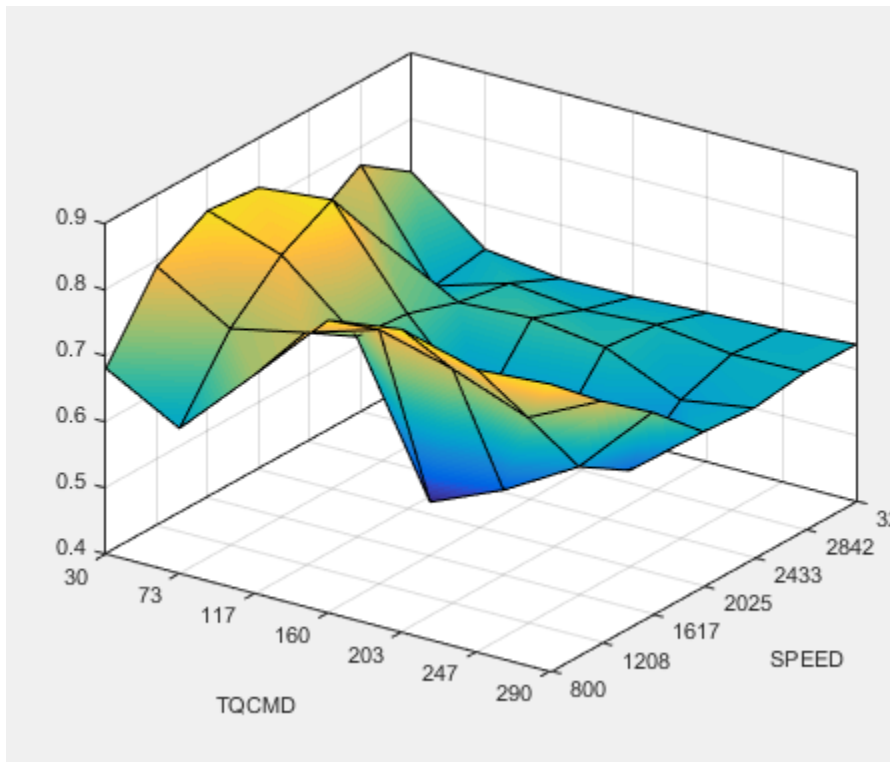




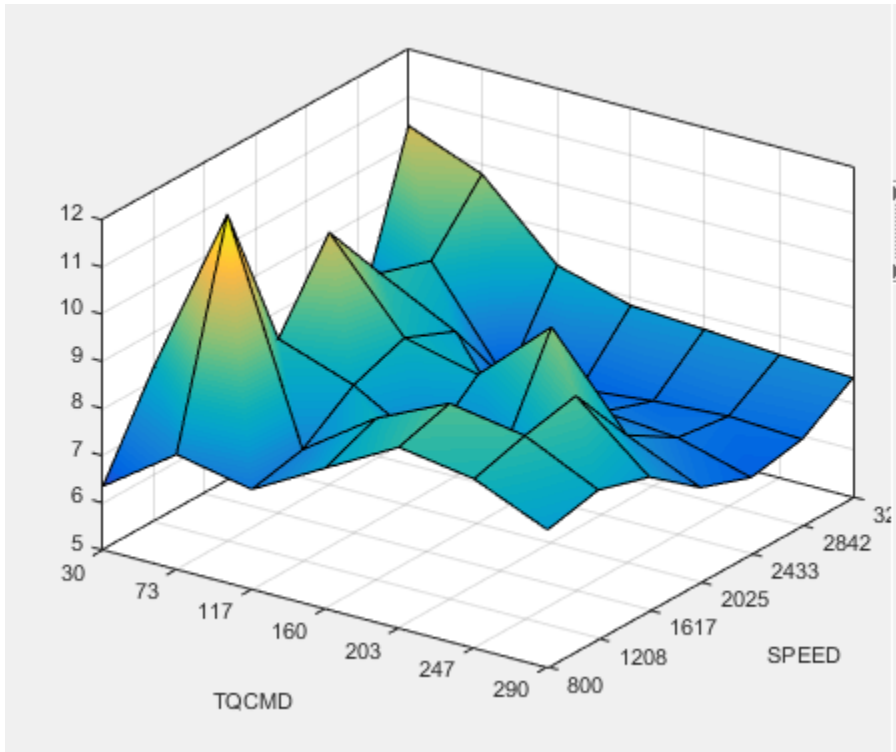
**Fuel Pressure Delta Table**



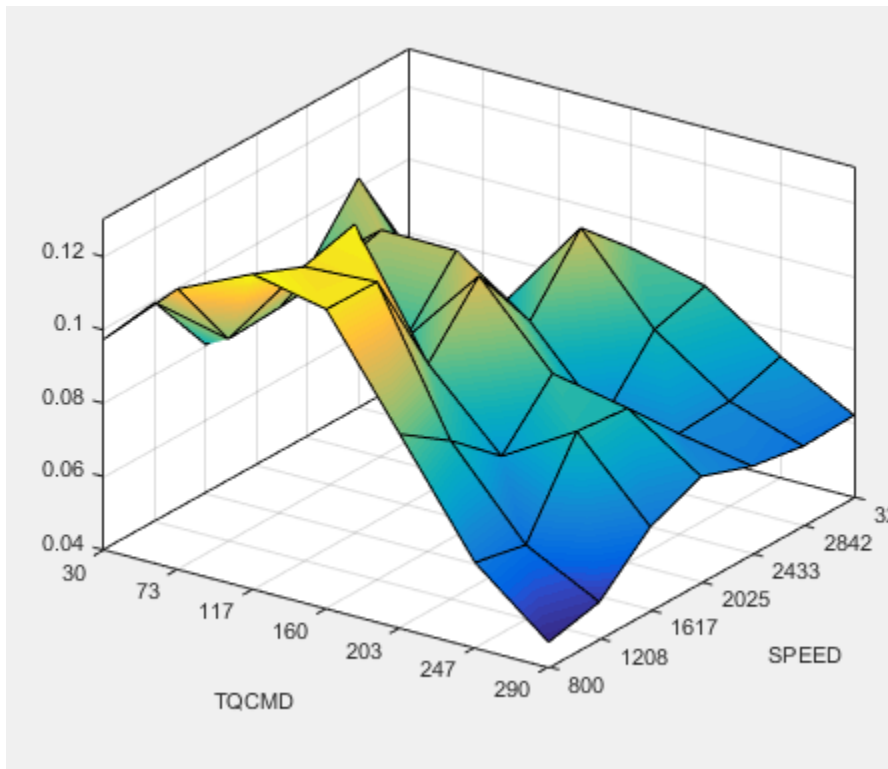
**Exhaust Gas Recirculation (EGR) Valve Position Table**



**Variable-Geometry Turbo (VGT) Position Table**



**Pilot Injection Timing (Pilot SOI Delta) Table**



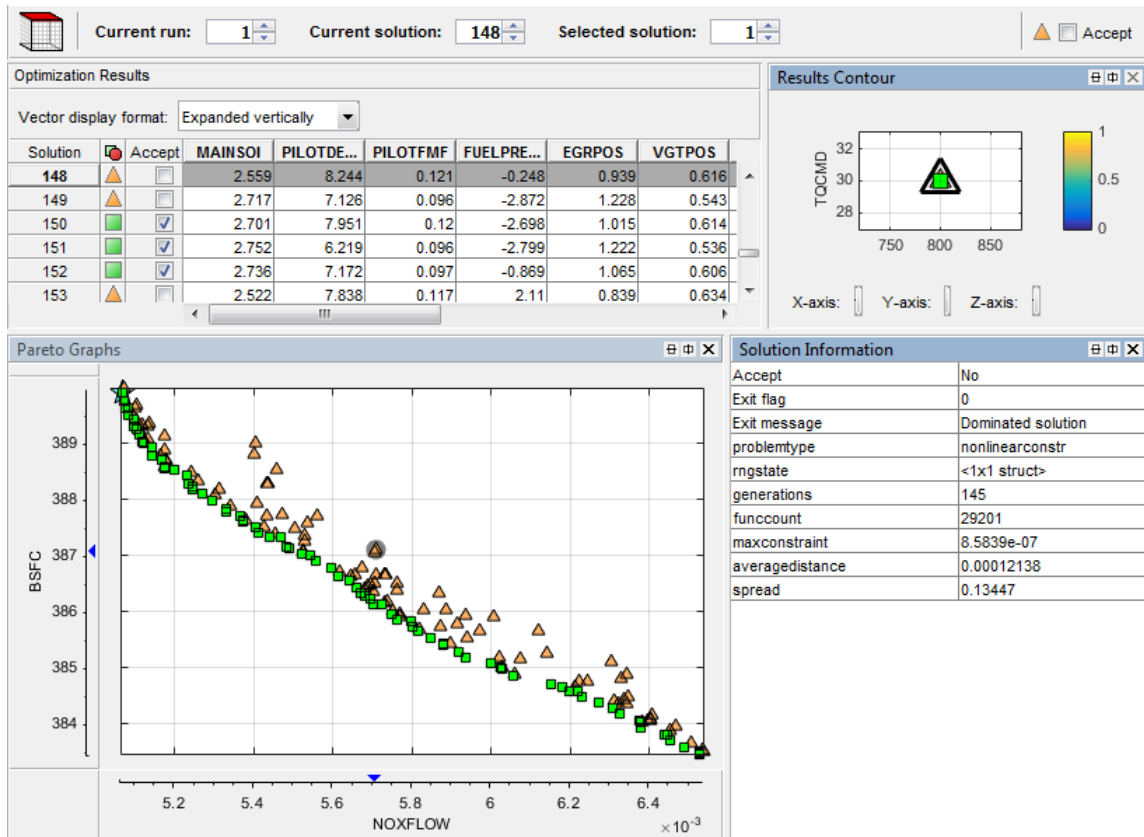
### Pilot Fuel Mass Fraction Table

## Examine the Multiobjective Optimization

The example file `CI_MultiInject.cag` also shows an example multiobjective optimization. This can be 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.

- 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.



- 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.
- Expand the Sum\_BSFC\_NOX node and select the Sum\_BSFC\_NOX\_Output to view the results.

# Model Quickstart

---

This section discusses the following topics:

- “Empirical Engine Modelling” on page 8-2
- “About Two-Stage Models” on page 8-3
- “Start the Toolbox and Load Data” on page 8-5
- “Set Up the Model” on page 8-7
- “Verify the Model” on page 8-12
- “Export the Model” on page 8-27
- “Create Multiple Models to Compare” on page 8-29

## Empirical Engine Modelling

This tutorial gives you a quick introduction to the modeling end of the Model-Based Calibration Toolbox product. The instructions show you how to use the toolbox 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 empirical engine modelling, follow the steps in these sections in order:

- 1** “About Two-Stage Models” on page 8-3
- 2** “Start the Toolbox and Load Data” on page 8-5
- 3** “Set Up the Model” on page 8-7
- 4** “Verify the Model” on page 8-12
- 5** “Export the Model” on page 8-27
- 6** “Create Multiple Models to Compare” on page 8-29

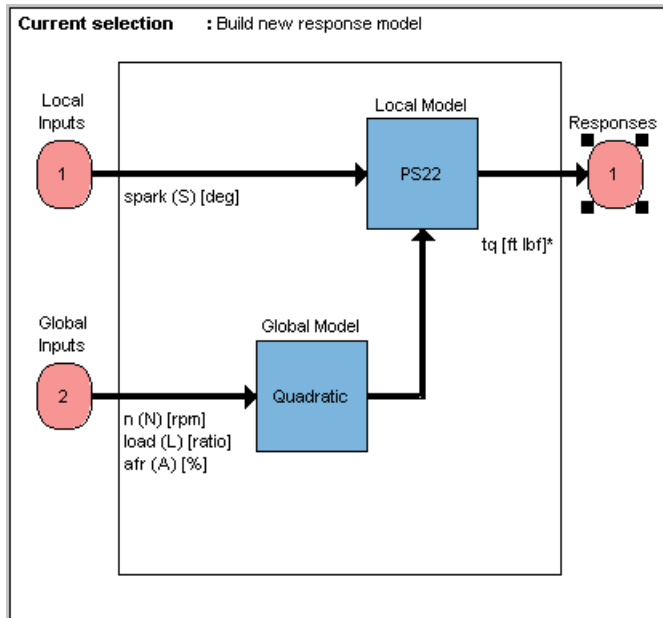


## About Two-Stage Models

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 “Start the Toolbox and Load Data” on page 8-5.

## Start the Toolbox and Load Data

- 1 To open the Model Browser, enter `mbcmode1` at the command prompt in MATLAB.
- 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.

For more details on functionality available within 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 8-7.

# Set Up the Model

## In this section...

“Specifying Model Inputs” on page 8-7

“Changing the Local Model Type” on page 8-10

## Specifying Model Inputs

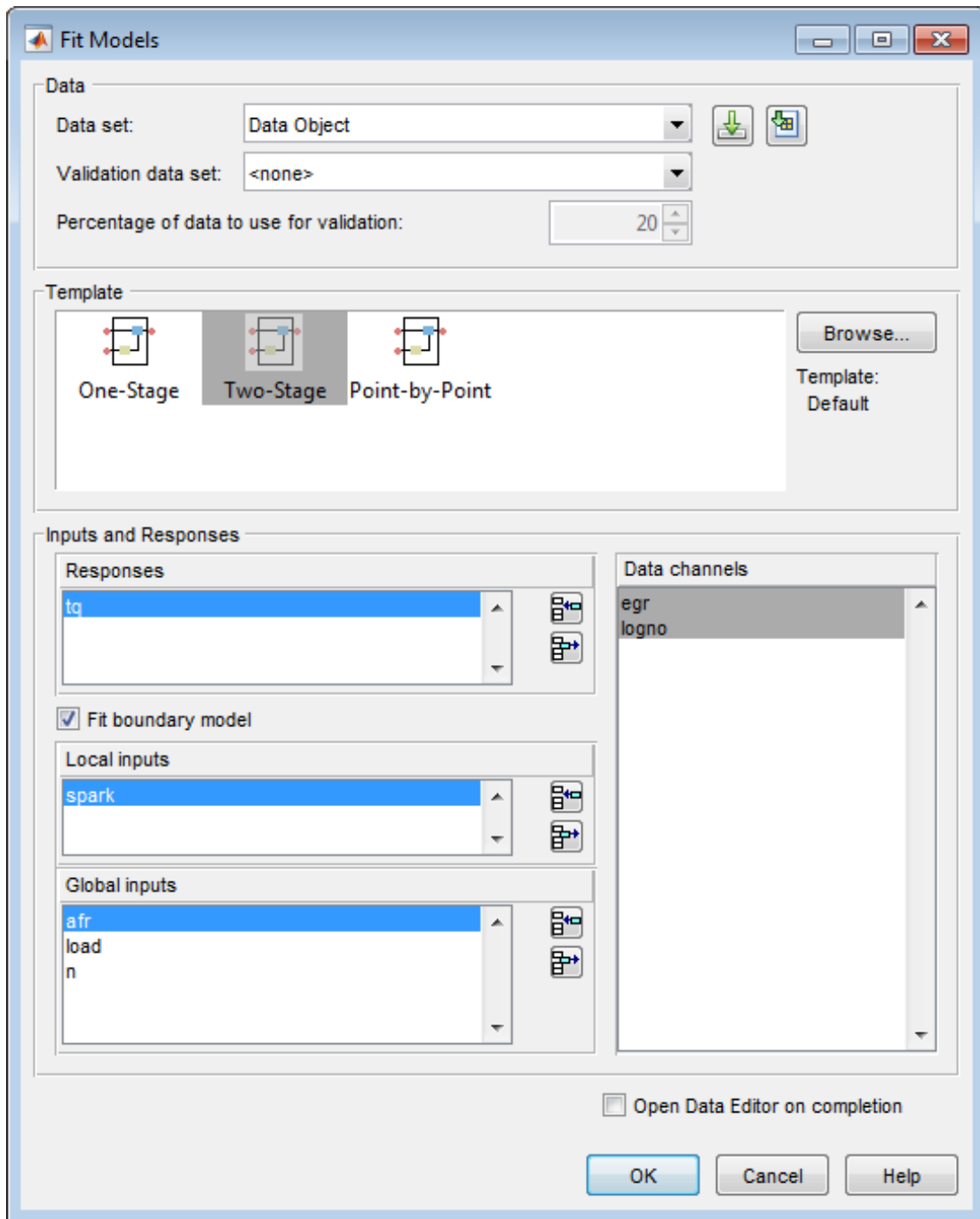
You can use the imported data to create a statistical model of an automobile engine that predicts the torque generated by the engine as a function of spark angle and other variables.

- 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 **tq** in the **Data channels** list and click the button to add it to the **Responses** list. Torque is the factor you want the model to predict.
- b Select **spark** in the **Data channels** list and click the button to add it to the **Local inputs** list.
- c Select **n**, **load** and **afr** in the **Data channels** list and click the button to add them to the **Global inputs** list.

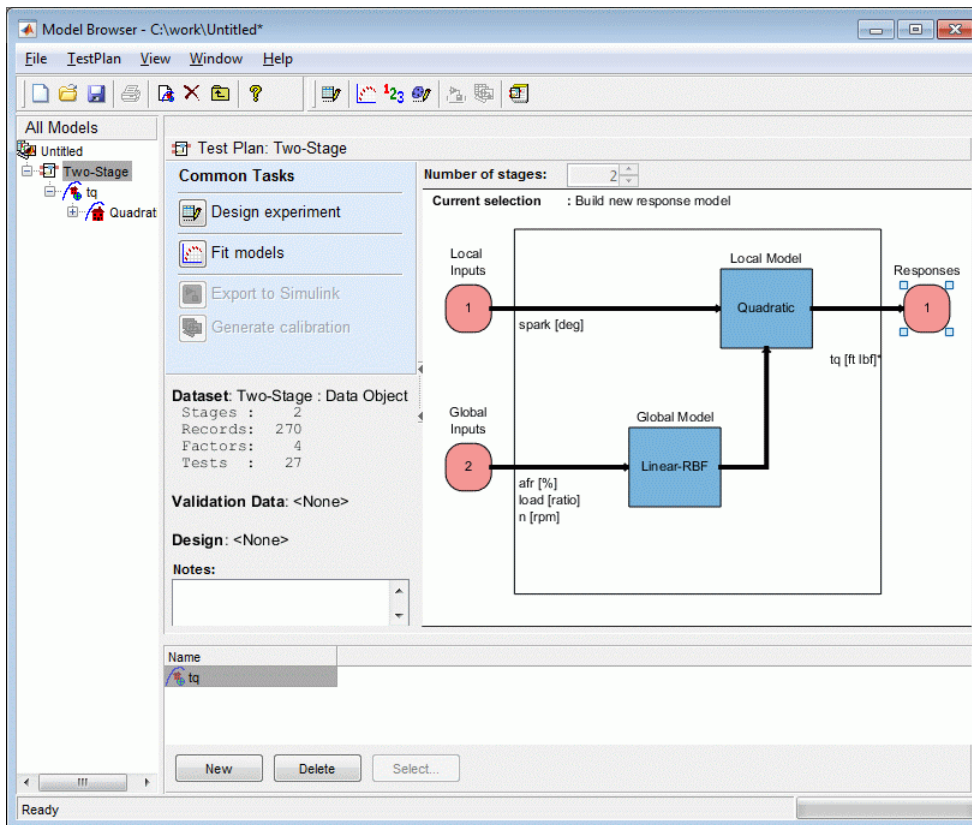


5 Click **OK** to fit the default model types to your selected data.

The default name of the new test plan, **Two-Stage**, appears in the Model Browser tree, in the **All Models** pane, with the response node **tq** underneath.

The view switches to the new test plan node, and the Model Browser window displays a diagram representing the two-stage model. Observe the inputs and response on the test plan diagram.

See also “Edit Test Plan Definition”.



## Changing the Local Model Type

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 polyspline curves. In this example, you use polyspline 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.

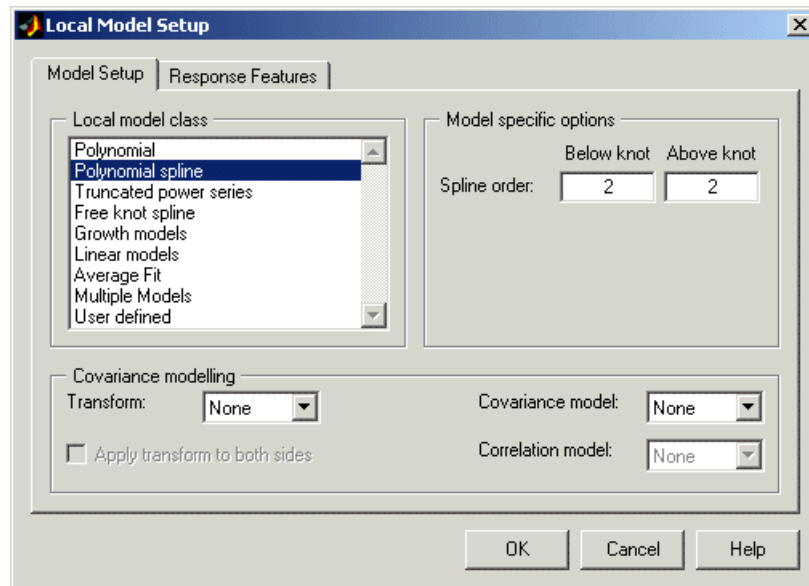
To change from the default models and specify polyspline as the local model type,

- 1 In the Model Browser test plan node view, 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 On the Response Models screen, 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 Set **Spline Order Below knot** to 2, and leave **Above knot** set to 2.





- c Click **OK** to dismiss the dialog box.
- 5 Select **Maximum** under **Datum**. Only certain model types with a clearly defined maximum or minimum can support datum models. See “New Response Models and Datum Models”.
- 6 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 polyspline) 2,2 (for spline order above and below knot) now appears on the two-stage model diagram. A new node appears on the tree in the **All Models** pane, called PS22.

For next steps, see “Verify the Model” on page 8-12.

## Verify the Model

### In this section...

“Verifying the Local Model” on page 8-12

“Verifying the Global Model” on page 8-14

“Selecting the Two-Stage Model” on page 8-16

“Comparing the Local Model and the Two-Stage Model” on page 8-21

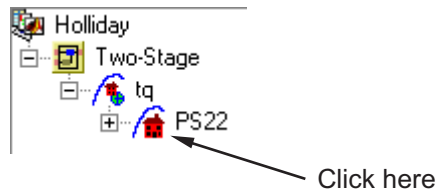
“Maximum Likelihood Estimation” on page 8-22

“Response Node” on page 8-24

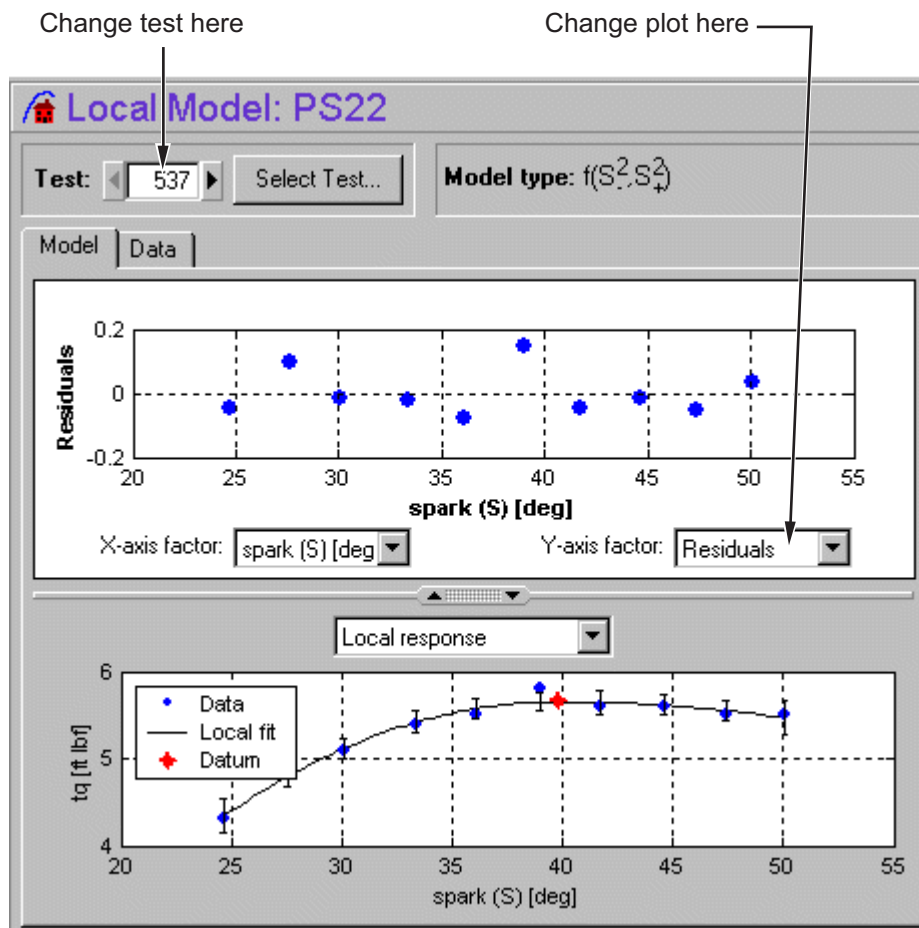
### Verifying the Local Model

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** pane 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.



The lower plot shows the data being fitted by the model (blue dots) and the model itself (line). The red spot shows the position of the polyspline knot, at the datum (maximum) point.

- 2 In the upper scatter plot pane, click the **Y-axis factor** pop-up menu and select Studentized residuals.
- 3 To display plots and statistics for the other test data, scroll through the tests using the **Test** arrows at the top left, or by using the **Select Test** button.

- 4 Select Test 588. You see a data point outlined in red. This point has automatically been flagged as an outlier.
- 5 Right-click the scatter plot and select **Remove Outliers**. Observe that the model is refitted without the outlier.

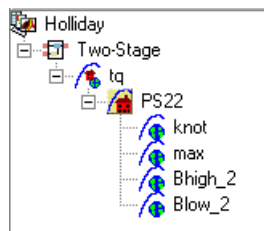
Both plots have right-click pop-up menus offering various options such as removing and restoring outliers and confidence intervals. Clicking any data point marks it in red as an outlier.

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. You can quickly locate tests with notes by clicking **Select Test**.

## Verifying the Global Model

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

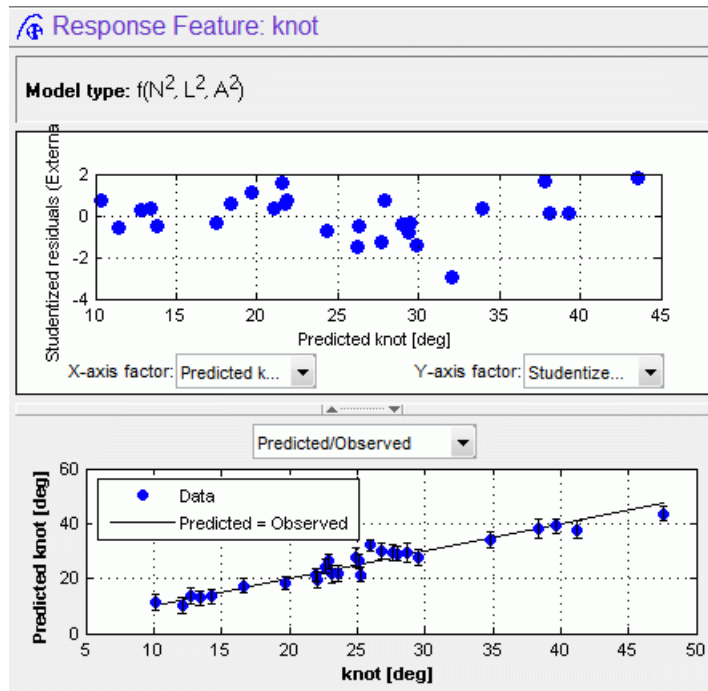
- 1 Expand the PS22 local node on the Model Browser tree by clicking the plus sign (+) to the left of the icon. Under this node are four response features of the local model. Each of these is a feature of the local model of the response, which is torque.



- 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** pane 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.



- 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 is designed to show 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. Use the plot of Cook's Distance to see the influence of each point on the model fit to help you decide whether to remove an outlier.

---

- 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.

Return to the **Local Model** pane by clicking the local node `PS22` in the Model Browser tree.

## Selecting 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.

Once 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. The Model Browser should look like the following example.

Model Browser - D:\Doc\Projects\Holiday2.mat\*

File Model View Outliers Window Help

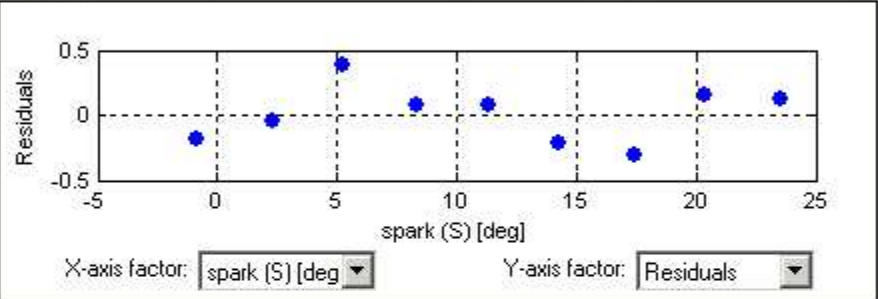
MLC

All Models

Local Model: PS22

Test: 588 Select Test... Model type:  $f(S^2, S^2)$

Model Data

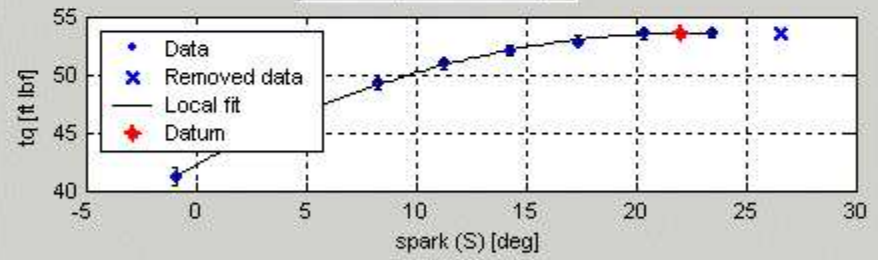


Residuals

spark (S) [deg]

X-axis factor: spark (S) [deg] Y-axis factor: Residuals

Local response



tq [ft lbf]

spark (S) [deg]

| Response Features | Observations | Parameters | Box-Cox | PRESS P |
|-------------------|--------------|------------|---------|---------|
| knot              | 27           | 7          | 1       | 3       |
| max               | 27           | 5          | 1       | 0.5     |
| Bhigh_2           | 27           | 3          | 1       | 0.003   |
| Blow_2            | 26           | 4          | 1       | 0.00    |

New Delete Select...

No two-stage model is selected

Ready

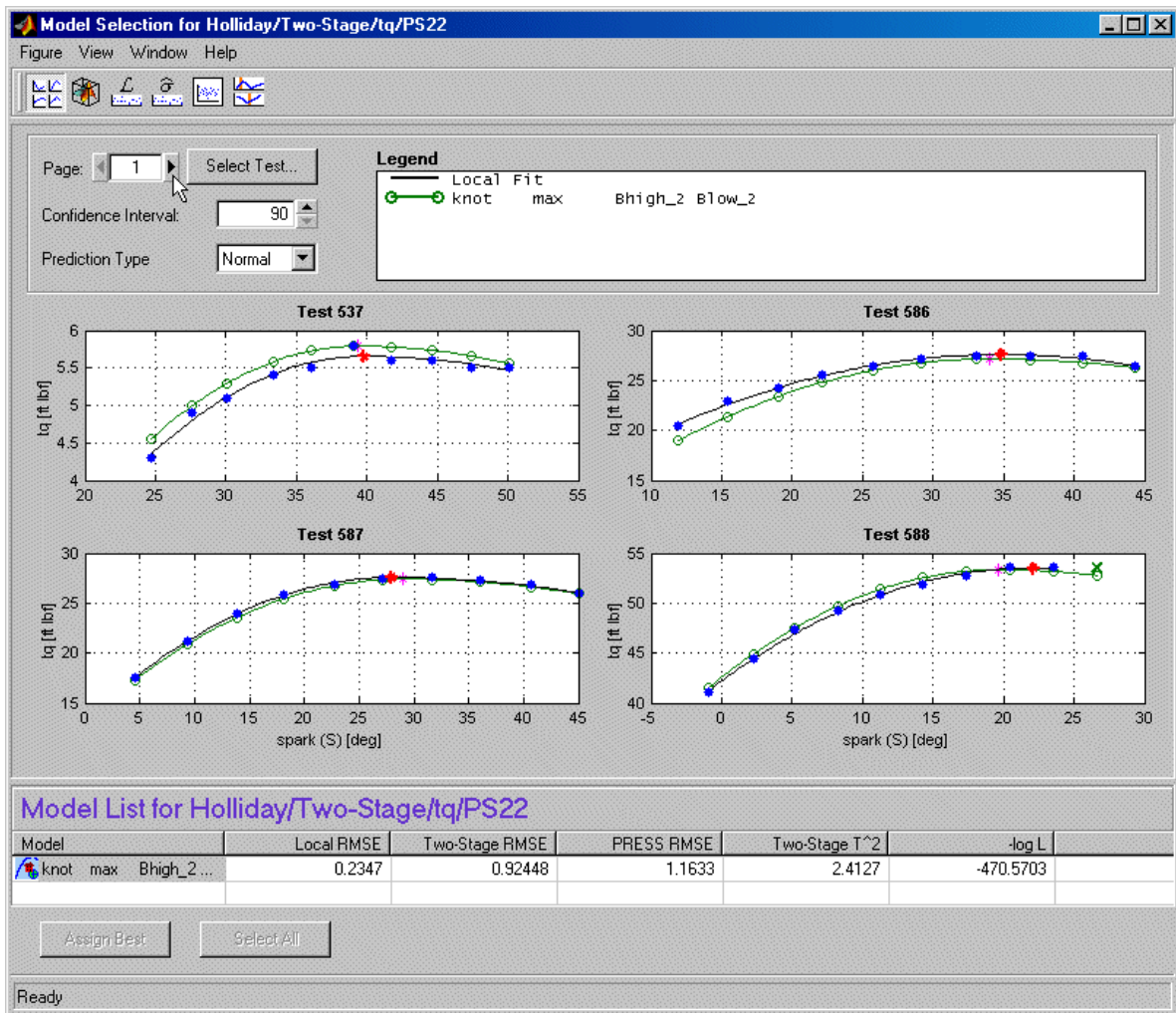
Open Model Selection here.

- 2 **Note** To construct a two-stage model from the local and global models, when viewing the local node in the model tree (with the house icon), click the **Select** button.
- 


Click **Select** in the **Response Features** list pane, and the Model Selection window appears. This window is intended to help you select a *best model* by comparing several candidate models.

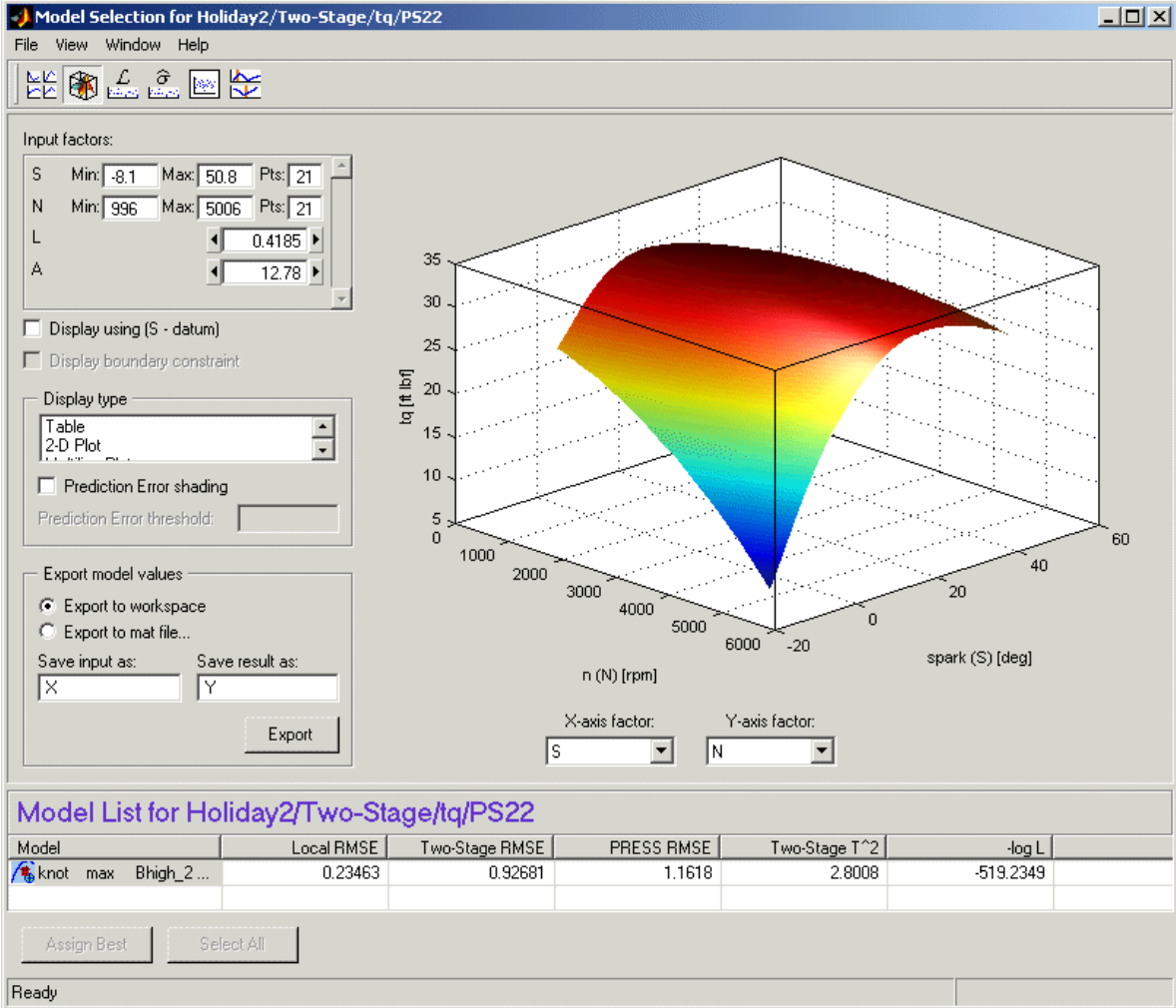
Now you can use the model selection features to view the fit of this two-stage model in various ways, to compare it with both the data and the local model fit. Toolbar buttons enable you to view the fit of the model in various ways. By default, the Tests view appears. These plots show how well the two-stage model agrees with the data.






- 3 Scroll through the tests using the left/right arrows or the **Select Test** button at the top left. The plots show the fit of the two-stage model for each test (green open circles and line), compared with the fit of the local model (black line) and the data (blue dots). You can left-click (and hold) to see information on each test or zoom in on points of interest by **Shift**-click-dragging or middle-click-dragging. Double-click to return the plot to the original size.

- 4 View the model as a surface by clicking the **Response Surface**  icon in the toolbar. You can rotate the plot by click-dragging it.
- 5 Click **Movie** in the **Display Type** list to see the surface (torque against spark and speed) vary through different values of load. Click **Replay** to see it again.
- 6 Take a look at some of the other display types.



The screenshot shows the 'Model Selection for Holiday2/Two-Stage/tq/PS22' window. On the left, the 'Input factors' section lists S (Min: -8.1, Max: 50.8, Pts: 21), N (Min: 996, Max: 5006, Pts: 21), L (0.4185), and A (12.78). Below this are checkboxes for 'Display using (S - datum)' and 'Display boundary constraint'. The 'Display type' dropdown is set to 'Table', with '2-D Plot' also visible. There are also checkboxes for 'Prediction Error shading' and a 'Prediction Error threshold' field. The 'Export model values' section has radio buttons for 'Export to workspace' (selected) and 'Export to mat file...'. Below are fields for 'Save input as:' (X) and 'Save result as:' (Y), and an 'Export' button. At the bottom left of the plot area, there are dropdowns for 'X-axis factor:' (S) and 'Y-axis factor:' (N). The 3D plot shows a surface of torque (Tq [ft lbf]) on the vertical axis (0 to 35) against spark (S) [deg] on the horizontal axis (-20 to 60) and speed (n (N) [rpm]) on the depth axis (1000 to 6000). The surface is colored with a gradient from blue at the bottom to red at the top. At the bottom of the window, a 'Model List for Holiday2/Two-Stage/tq/PS22' table is displayed.

| Model  | Local RMSE | Two-Stage RMSE | PRESS RMSE | Two-Stage T <sup>2</sup> | -log L    |
|--|------------|----------------|------------|--------------------------|-----------|
|  knot max Bhigh_2 ... | 0.23463    | 0.92681        | 1.1618     | 2.8008                   | -519.2349 |

Buttons for 'Assign Best' and 'Select All' are located below the table. The status bar at the bottom indicates 'Ready'.

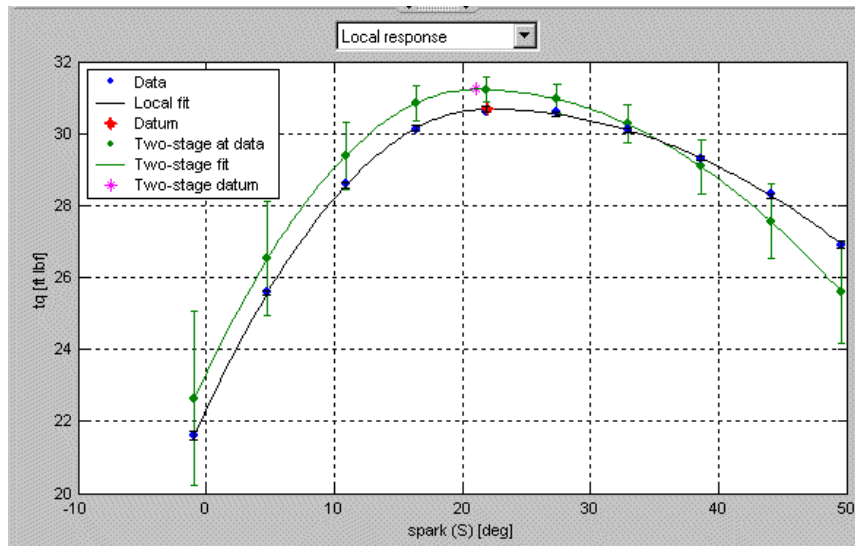
For more detailed help on all the views available in the Model Selection window, see “Compare and Select Best Models”.



- 7 Dismiss the Model Selection window, and accept the best model by clicking **Yes** in the Model Selection dialog (it is the only two-stage model so far).
- 8 The MLE dialog appears, prompting you to calculate the maximum likelihood estimate (MLE) for the two-stage model. Click **Cancel**. You can calculate MLE later.


## Comparing the Local Model and the Two-Stage Model

Now the lower plots in the **Local Model** pane show two lines fitted to the test data: the Local Model line (black), and the Two-Stage Model line (green). The plots also show the data (in blue), so you can compare how close the two-stage model fit is to both the data and the local fit for each test.

You can scroll through the various tests (using the arrows at the top left or the **Select Test** button) to compare the local and two-stage models for different tests.




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.

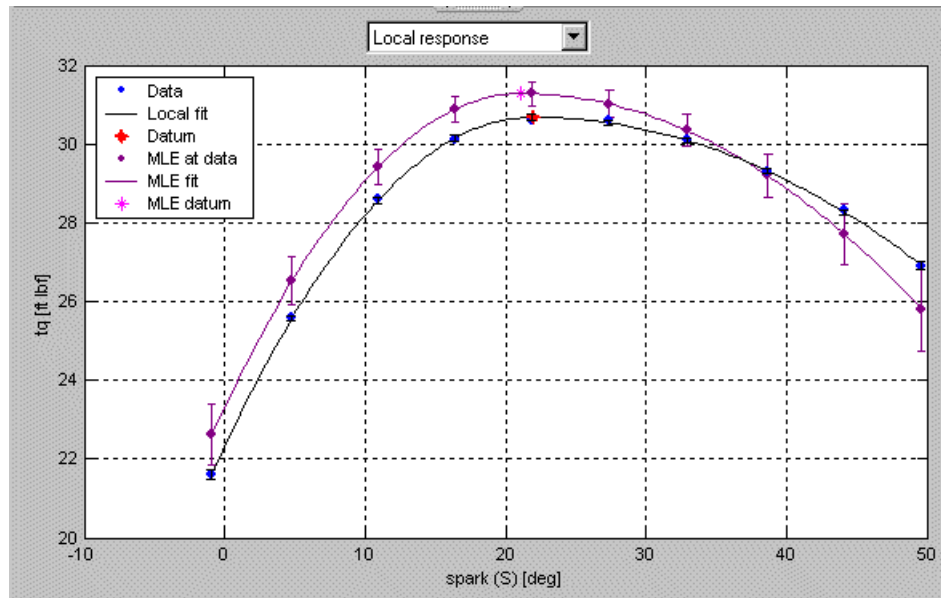
Click the  button in the toolbar to calculate the maximum likelihood estimate.

## Maximum Likelihood Estimation

The global models were created in isolation without accounting for any correlations between the response features. Using MLE (maximum likelihood estimation) to fit the two-stage model takes account of possible correlations between response features. In cases where such correlations occur, using MLE significantly improves the two-stage model.

- 1 You reach the MLE dialog from the local node (PS22 in this case) by
  - Clicking the  button in the toolbar
  - Or by choosing **Model > Calculate MLE**
- 2 Leave the algorithm default settings and click **Start** to calculate MLE.
- 3 Watch the progress indicators until the process finishes and a two-stage RMSE (root mean square error) value appears.
- 4 Click **OK** to leave the MLE dialog.

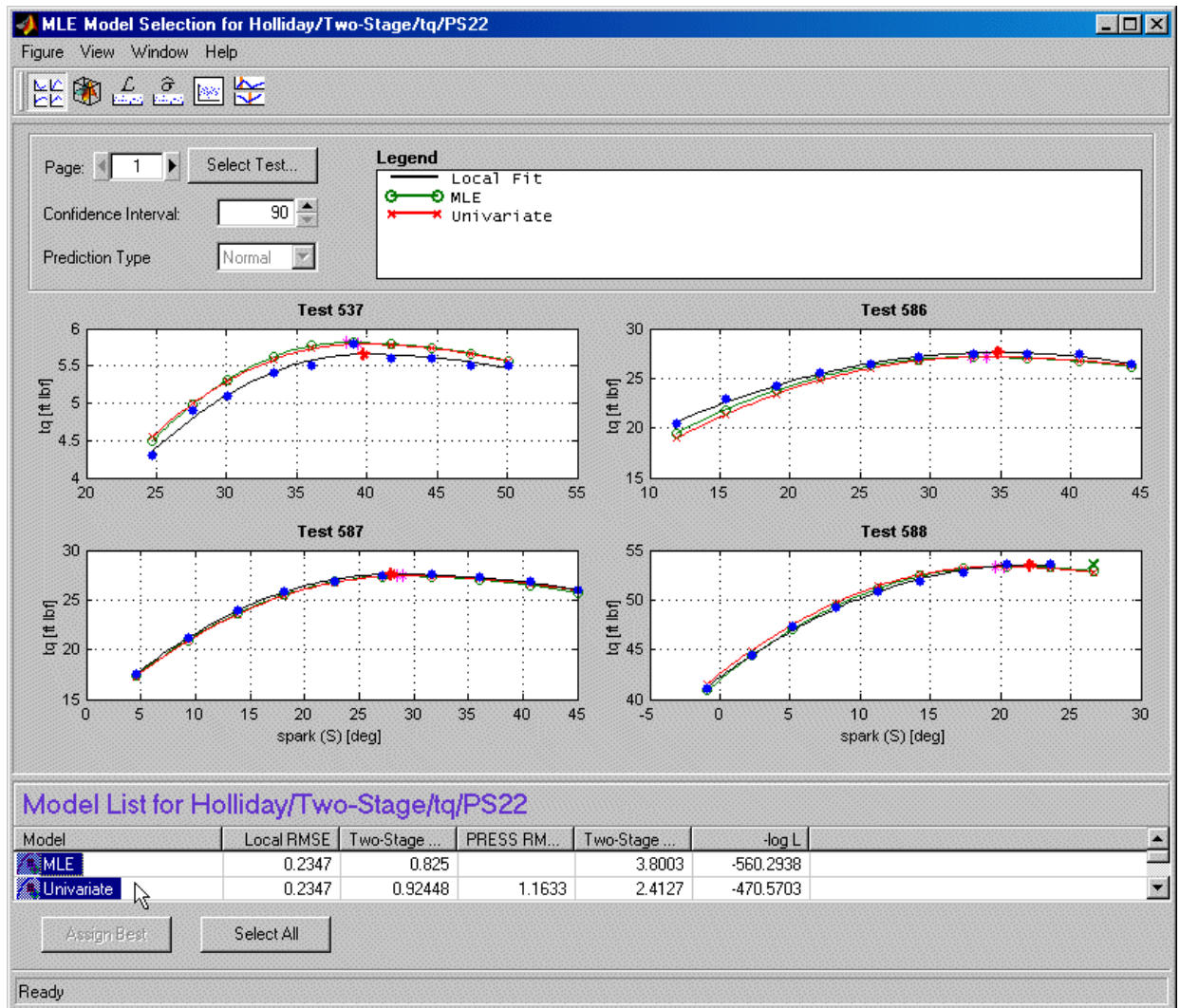
Now the plots on the **Local Model** pane all show the two-stage model in purple to indicate that it is an MLE model. This is also indicated in the legend. Notice that all the model icons in the tree (the response, the local model, and the response features) have also changed to purple to indicate that they are MLE models.



- 5 Click the **Select** button. This takes you to the Model Selection window.

Here you can compare MLE with the univariate model previously constructed (without correlations). By default, the local fit is plotted against the MLE model.

- 6 Select both MLE and the Univariate model for plotting by holding down **Shift** while you click the Univariate model in the **Model List** at the bottom of the view.



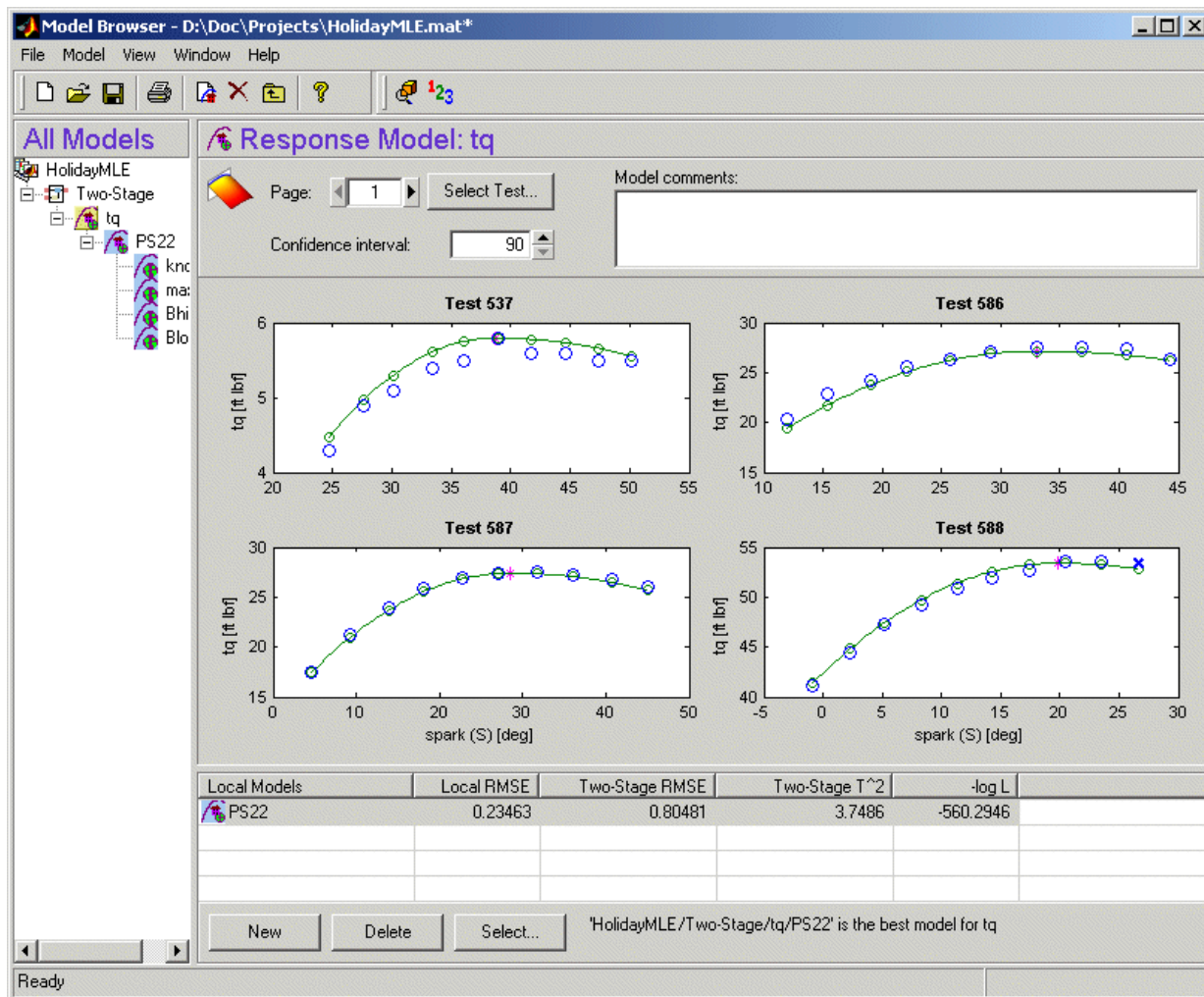
7 Close the Model Selection window. Click **Yes** to accept the MLE model as the best.

## 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 this:



This shows 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 MLE model (in green) against the data (in blue) for each test.

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

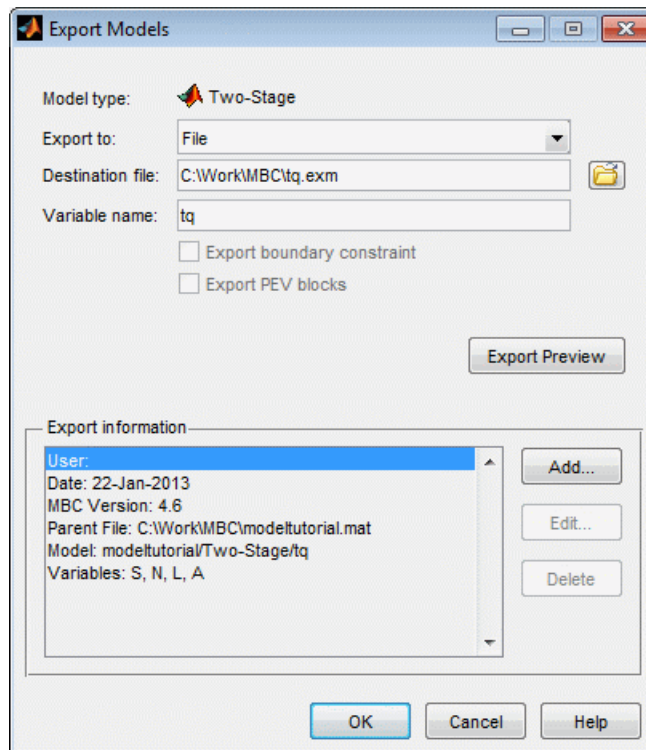
For next steps, see “Export the Model” on page 8-27 or “Create Multiple Models to Compare” on page 8-29.



## 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 a file, 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 Choose **File** from the **Export to** pop-up menu. This saves the work as a file for use within the Model-Based Calibration Toolbox product, for instance, to create calibrations in the CAGE Browser.
- 4 In the **Export to** edit box, select the destination of the file. You can do this by typing directly in the edit box, or using the Browse button if you want to locate a directory or use an existing file.



- 5 Click **OK** to export the models to file.

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 8-29.

## Create Multiple Models to Compare

### In this section...

“Methods For Creating More Models” on page 8-29

“Creating New Local Models” on page 8-29

“Adding New Response Features” on page 8-32

“Comparing Models” on page 8-33

“Creating New Global Models” on page 8-34

“Creating Multiple Models Using Build Models” on page 8-37

### Methods For Creating More Models

Once you have fitted and examined a single model, you will normally want to create more models to search for the best fit. You can create individual new models, use the **Build Models** function to create a selection of models at once, or create a template to save a variety of model settings for reuse.

You can create new child nodes by clicking the **New** button from any modeling node. The Model Setup dialog appears where you can change the type and settings, and when you close the dialog the view switches to the new child node on the tree. You can do this for multiple child nodes to create a selection of different model types fitted to the same data. You can also use the Build Models dialog to quickly create a selection of different child nodes to compare. The following exercises show you examples of these processes. Note that you need to complete the previous tutorial sections to have a complete two-stage model as a starting point.

### Creating New Local Models

To follow these examples, you need to create the initial models by following the previous sections in “Empirical Engine Modelling” on page 8-2.

- 1 As an example, select the tq response node and click **New** in the **Local Models** list pane.

The Local Model Setup dialog appears.

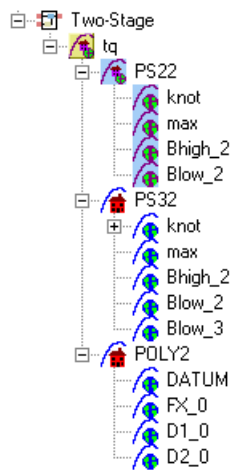
- 2 Select a **Polynomial Spline**, and leave the default spline order of 3 below the knot and 2 above. Click **OK**.

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

- 3 Return to the parent **tq** response node, and click **New** again, in the **Local Models** list 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 **Diagnostic statistics** pane, select **Local diagnostics** from the drop-down menu. Observe the value of **s<sub>i</sub>** in this pane. This is the value of RMSE (root mean squared error) for the current (*i*<sup>th</sup>) 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  $s_i$  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  $s_i$  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  $s_i$  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, click RMSE Plots () in the toolbar (or select **View > RMSE Plots**) to open the RMSE Explorer dialog.
  - a Use the plot to help you identify problem tests. Use the drop-down menus to change the display. For example, select  $s_{\text{knot}}$  to investigate the error values for knot (MBT), or  $s_e$  to look at overall error.
  - b You can navigate to a test of interest from the RMSE Explorer by double-clicking a point in the plot to select the test in the Model Browser local model view.
- 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 always see statistics for the list of child models of the currently selected 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 Click the **New** button under the list of response features.

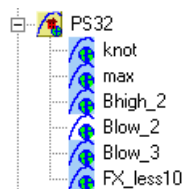
A dialog appears with a list of available response features.

- 3 Select  $f(x+\text{datum})$  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 ( $f(x+\text{datum})$ ) to measure the value of the model (torque) at  $(-10 + \text{MBT})$  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 **Select**. Notice that there are four possible two-stage models this time. 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. Click **Cancel** to avoid calculating MLE.

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



- 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 also cannot use the Model Selection window to fully compare the two-stage models until every local model in the test plan has a two-stage model calculated.

- To calculate the two-stage model for POLY2, click **Select** at the POLY2 node. Close the Model Selection window and click **Yes** to accept the best model. Click **Cancel** to avoid calculating MLE, then the two-stage model is calculated.

## Comparing Models

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

| Local Models | Local RMSE | Two-Stage RMSE | PRESS RMSE | Two-Stage T <sup>2</sup> | -log L    |
|--------------|------------|----------------|------------|--------------------------|-----------|
| PS22         | 0.23463    | 0.80481        |            | 3.7486                   | -560.2946 |
| PS32         | 0.26756    | 1.3404         | 1.75       | 2.5392                   | -616.8609 |
| POLY2        | 0.5        | 1.2431         | 1.4983     | 1.6794                   | -468.5096 |

'Untitled/Two-Stage/tq/PS22' is the best model for tq

- 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 “Model Selection Guide”. 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).

Notice that your first two-stage model (PS22) does not have a PRESS RMSE value. This is because it cannot be displayed for MLE models. You need non-MLE models to be able to use PRESS for direct comparison.

- 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 Now click **Select** to open Model Selection to compare all three two-stage models simultaneously. 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 the Model Selection tool to introduce the use of RMSE and PRESS to help you identify better models. The principle is the same at each level in the model tree: add new child models and use the Model Selection window to choose the best.

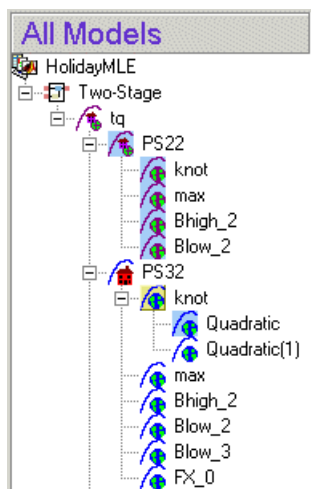
- 1 Select one of the response feature nodes under the PS32 node, such as knot.



- 2 Click **New** in the lower pane, **New Model** in the toolbar, or select **File > New Model**. The Model Setup dialog appears. Click **OK** without changing any settings, to create a copy of the parent model.
- 3 Return to the parent model and repeat to create a second child node of **knot**.

Two new global model child nodes now appear underneath **knot**, as shown. Both are labeled **Quadratic**, as they are currently copies of the parent model. You can create any number of child nodes to search for the best global model fit for each response feature in your tree. When you choose the best, it is copied to the parent node, in this case **knot**, including any outliers you decide to exclude.

A good technique for creating multiple models can be to leave the first child node unchanged, then you always have a copy of the original model for comparison.



- 4 Select one of the new **Quadratic** nodes, then select the menu item **Model > Set Up**.

The Global Model Setup dialog appears. Here you can change the type and settings of the model to see if you can find a better fit to the data with a different model type.

- 5 Use the drop-down menu to change the **Model class** to **Hybrid RBF** and click **OK**.

The new model fit is calculated, and the **Quadratic** node's name changes to **Linear-RBF**.

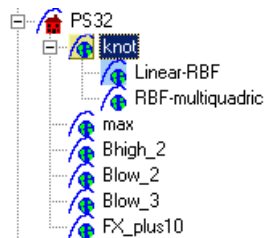
- 6 Select the remaining **Quadratic** node, then select **Model > Set Up**.

- 7 Use the drop-down menu to change the **Model class** to **Radial basis function**, and click **OK**. There are many other settings you can alter for both these model types, but for a quick exploration of the trends in the data it is worth trying the default model settings.

The new model fit is calculated and the **Quadratic** node's name changes to **RBF - multiquadric**.

- 8 To compare the two child node models, select the parent node **knot** and click **Select**. Whichever model you assign as best is copied to the **knot** node when you close the Model Selection window and click **OK**.

Notice that the child node model assigned as best is highlighted in blue, and the local node has changed from the two-stage icon back to the local model icon (a red house) as shown. This is because you have changed one of the response feature models, and so you need to recalculate the two-stage model using the new global model for this response feature. First you need to select best global models for every response feature.



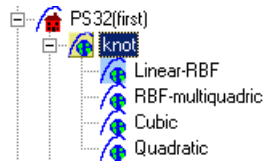
- 9 Add two more child nodes to the **knot** global model (select **knot**, then click **New**, and repeat).

Notice that now the new nodes are copies of **Linear-RBF**, because that model was selected as best.

- 10 Select the two new nodes in turn and change their model types. Try a cubic and quadratic polynomial model type as follows:
  - a Select the menu item **Model > Set Up**.
  - b Choose **Linear model** from the **Model class** drop-down menu and set the polynomial order for each factor to **3** to create a cubic model. Click **OK**

Repeat for the other model and set the polynomial order to **2** to create a quadratic.

- 11 To compare all four child node models, select the parent node **knot** and click **Select**. **Linear-RBF** still performs the best for **PRESS RMSE**. Whichever model you assign as best is copied to the **knot** node.



- 12 Select the **knot** model node, then select **Model > Make Template**. Browse to a suitable work directory and enter the name **Mytemplate**. Click **Save**.

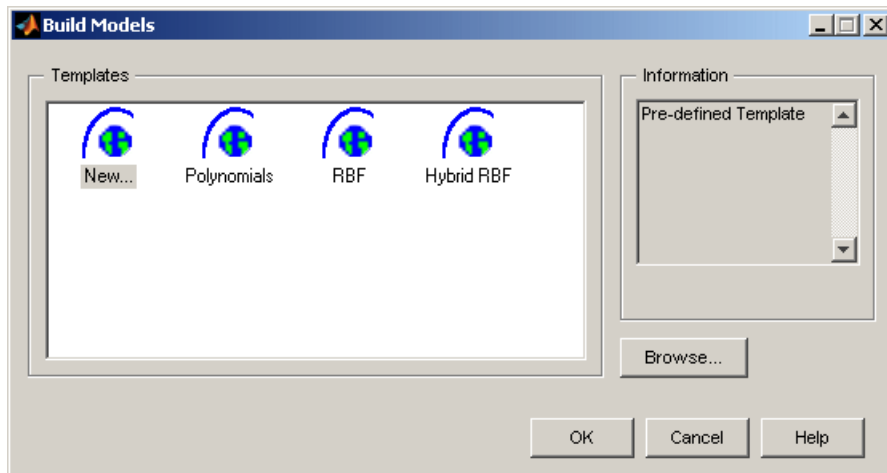
## Creating Multiple Models Using Build Models

The quickest way to create multiple different models to compare is to use the **Build Models** function. You can use this to select a template and build a selection of models as child nodes of the current node. The best model of this selection of child nodes is automatically selected (it will have a blue icon), based on the selection criteria you choose (such as **PRESS RMSE**, **RMSE**, **Box-Cox**, and so on).

- 1 Before calculating **MLE**, select a global model such as **max**.

You cannot reach the **Build Models** dialog from an **MLE** global model. Note that calculating **MLE** is not irreversible — to go back you can always go to **Model Selection** (from the local node) and select the **Univariate** model as best.

- 2 Click **Build Models** in the toolbar.



The Build Models dialog appears. Here you can choose a template for the type of models you want to build. There are predefined templates for polynomials, RBFs, hybrid RBFs, free knot splines (for single input models), or you can select a suitable parent node in the current project to use as a template.

You can also create templates of whatever models you choose by selecting the **New** template in the Build Models dialog box, or using the **Model > Make Template** menu item at a model node, as you did in the previous section. Your user-defined templates can then be found via the Build Models dialog. You can use the **Browse** button to find stored templates that are not in the default directory.

- 3 Click **Browse** and select the directory containing the template you created earlier, named **Mytemplate**. Click **OK**.

Your new template (called **Mytemplate**) now appears in the Build Models dialog along with the defaults. Note that you can set the default directory where the toolbox looks for templates (and models, data, and projects) using **File > Preferences**.

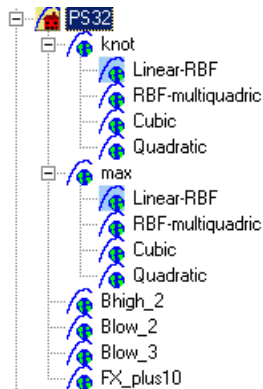
- 4 Select **Mytemplate**. Notice the four model types you saved in this template appear in the **Information** pane. Click **OK**.

The Model Selection dialog appears, where you can choose a criterion for automatically selecting the best model out of the child nodes.

- 5 Use the drop-down menu to choose **PRESS RMSE** as the selection criteria for the best model, and click **OK**.

Four child nodes appear: **Linear-RBF**, **RBF-multiquadric**, **Cubic**, and **Quadratic**. These are the model types you selected when you built the template and are now fitted to the data for **max**.

The most favorable child node model, based on **PRESS RMSE**, is selected as best (highlighted in blue) as shown in the following figure. This model is also copied to the parent node **max**, in the same way as if you had used the Model Selection window to assign a best model.



Try one of the default templates in the Build Models dialog box as follows:

- 1 Select another global model such as **Blow\_2**.
- 2 Click **Build Models** in the toolbar.
- 3 Select **RBF** and click **OK**. Click **Build** in the following Model Building Options dialog to build a selection of child nodes.

Similarly, you can use the Build Models dialog to automatically build a selection of polynomial or hybrid RBF models, or your own selection of model types, to search for the best fit.

- 4 In the Model Selection dialog select **PRESS RMSE** as the selection criteria for the best model and click **OK**.
- 5 Look at the statistics in the lower list pane to quickly compare all the different RBF kernel child models. If one model performs significantly better in terms of **PRESS RMSE** and **RMSE** you might choose not to click **Select** to compare all the child node models. However, it is usually useful to visually inspect the models to see how they compare.

- 6 When you have chosen a best model, it can be useful to select some (or all) of the rejected models in the bottom list pane and press **Delete**. 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.

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.

When you have chosen best global models for all your response features, you need to recalculate the two-stage model. Click **Select** at the local model (**PS32**) node to calculate the two-stage model.

# Design of Experiment

---

This section discusses the following topics:

- “Design of Experiment” on page 9-2
- “Set Up a Model and Create a Design” on page 9-5
- “Create Optimal Designs” on page 9-8
- “View Design Displays” on page 9-15
- “Use the Prediction Error Variance Viewer” on page 9-18
- “Create Classical Designs” on page 9-24
- “Use the Design Evaluation Tool” on page 9-30
- “Create Space-Filling Designs” on page 9-33
- “Apply Constraints” on page 9-35
- “Save Designs” on page 9-41

## Design of Experiment

**In this section...**

“Why Use Design of Experiment?” on page 9-2

“Design Styles” on page 9-3

“Get Started With The Design Editor” on page 9-3

### 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.

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 9-8.

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 a way as to maximize coverage of the factors' ranges as quickly as possible. See “Create Space-Filling Designs” on page 9-33.

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 9-24.

You can augment any design by optimally 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.

## Get Started With The Design Editor

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

- 1 To start the tutorial, you pick a model to design an experiment for, enter the Design Editor, and construct an optimal design. Once you create a design, you can use the

displays and tools to examine the properties of the design, save the design, and make changes.

See

- “Set Up a Model and Create a Design” on page 9-5
  - “Create Optimal Designs” on page 9-8
  - “View Design Displays” on page 9-15
  - “Use the Prediction Error Variance Viewer” on page 9-18
  - “Save Designs” on page 9-41
  - “Improving the Design” on page 9-21
- 2** Next you create a classical design, and use the Prediction Error Variance Viewer to compare it with the previous design. You can also use the Design Evaluation tool to view all details of any design; it is introduced in this example.

See

- “Create Classical Designs” on page 9-24
  - “Use the Design Evaluation Tool” on page 9-30
- 3** Lastly you construct a space-filling design and compare it with the others using the Prediction Error Variance Viewer. Then you construct and apply two different constraints to this design and view the results. Normally 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

- “Create Space-Filling Designs” on page 9-33
- “Apply Constraints” on page 9-35

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

## Set Up a Model and Create a Design

### In this section...

“Setting Up Model Inputs” on page 9-5

“Open the Design Editor” on page 9-5


“Creating a New Design” on page 9-6

### Setting 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 Quick Start tutorial: 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.

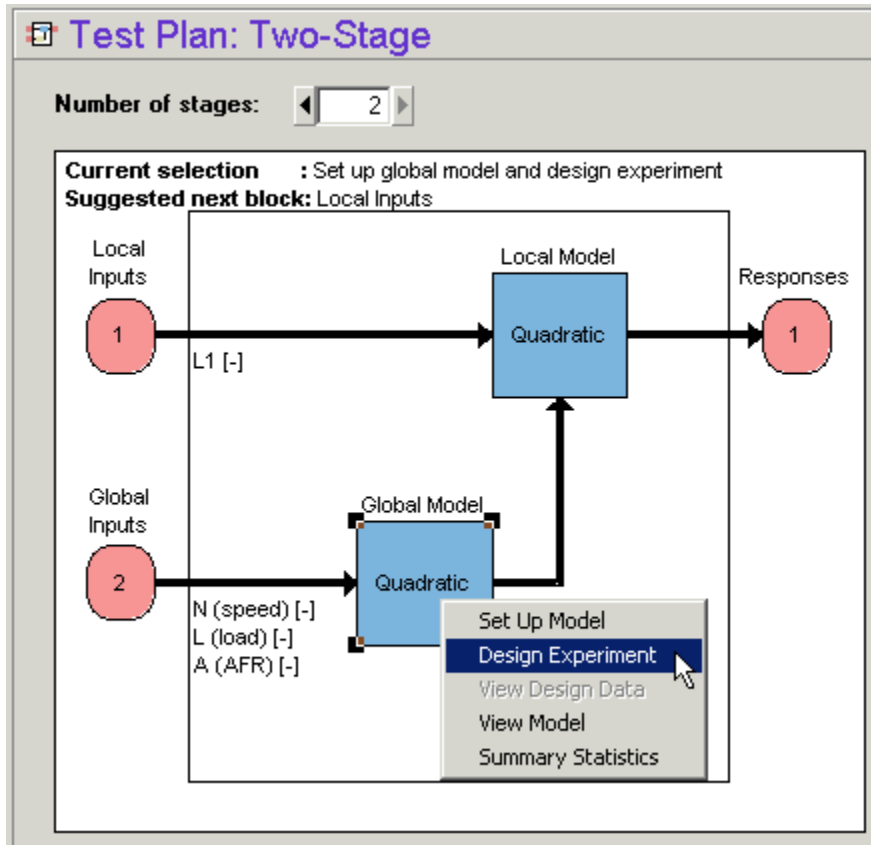
### Open the Design Editor

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


The Design Editor window appears.

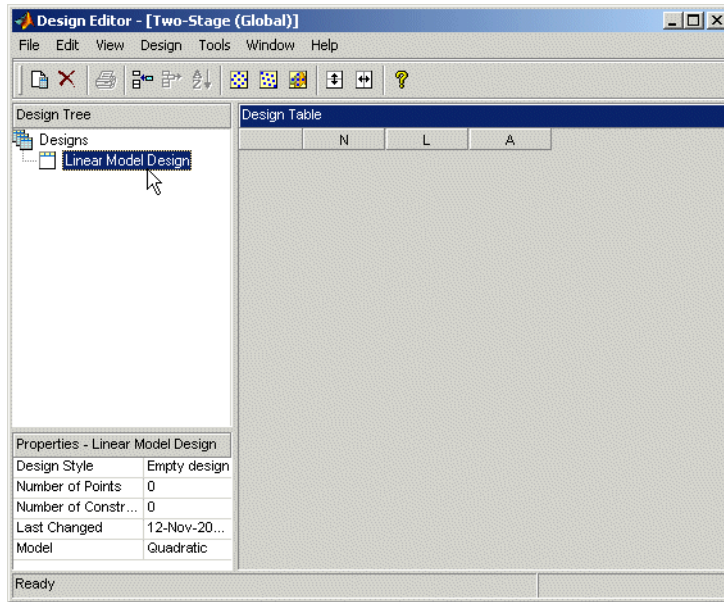
Alternatively, to open the Design Editor, you can also use either of the following methods:

- 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**.



## Creating a New Design

- 1 Click the  button in the toolbar or select **File > New**. A new node called Linear Model Design appears.



- 2 The new **Linear Model Design** node is automatically selected. An empty Design Table appears (see above) because you have not yet chosen a design. For this example you create an optimal design for the default global model, which is a quadratic.

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 **Optimal** (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 Optimal Designs” on page 9-8.

## Create Optimal Designs

### In this section...


“Introducing Optimal Designs” on page 9-8

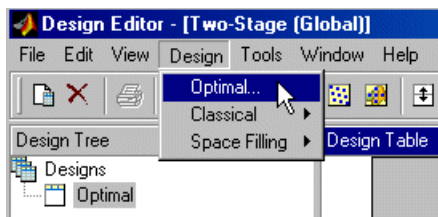
“Start Point Tab” on page 9-9

“Candidate Set Tab” on page 9-10

“Algorithm Tab” on page 9-13

### Introducing Optimal Designs

Choose 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.

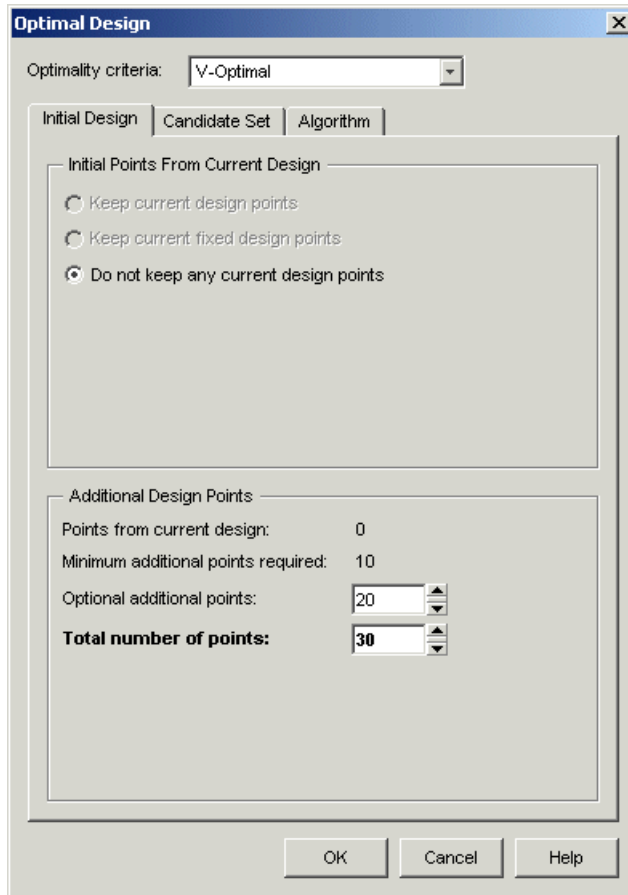
The Optimal Design dialog consists of several tabs that contain the settings for three main aspects of the design:

- 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

Optimality criteria: V-Optimal

Initial Design Candidate Set Algorithm

Generation algorithm: Grid

Allow replicated points in design

Options

View coded values

Equally spaced levels

Minimum N value: 0

Maximum N value: 100

Number of levels for N: 21

User-specified levels

0.5:100

Display

Display a maximum of 2500 points

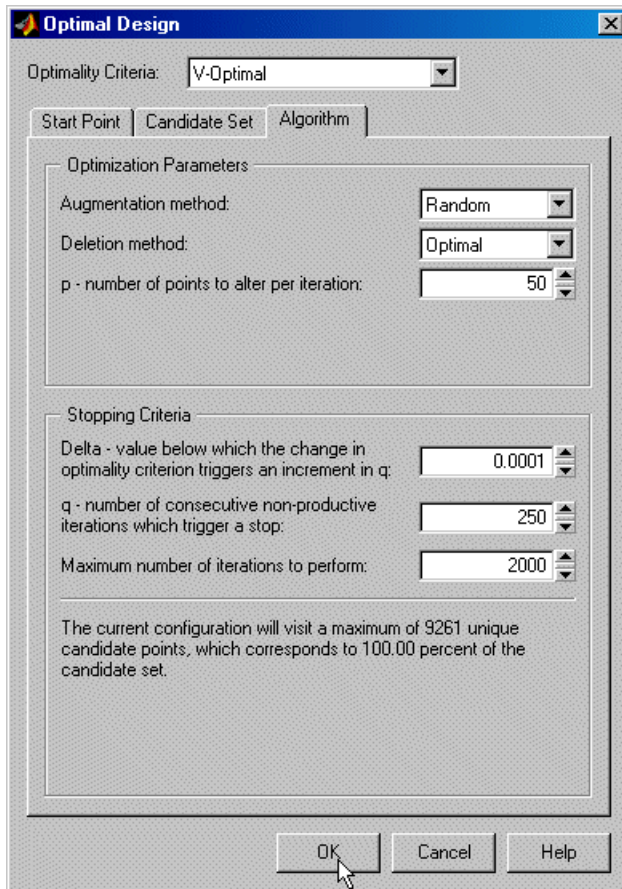
0 constraints will be applied to this candidate set.

OK Cancel Help

Open display windows with these buttons.

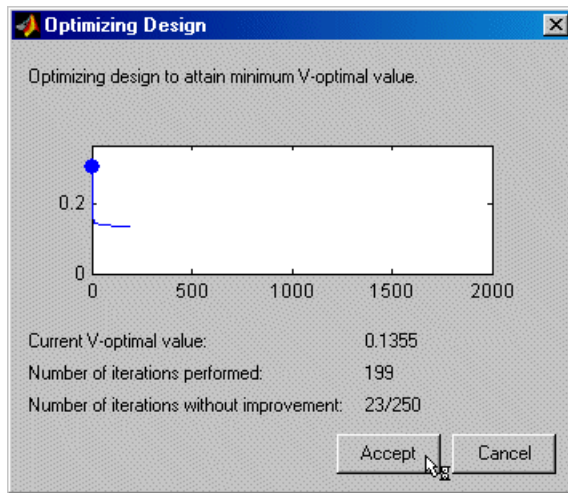
Change the number of levels of the selected variable here.

## Algorithm Tab



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

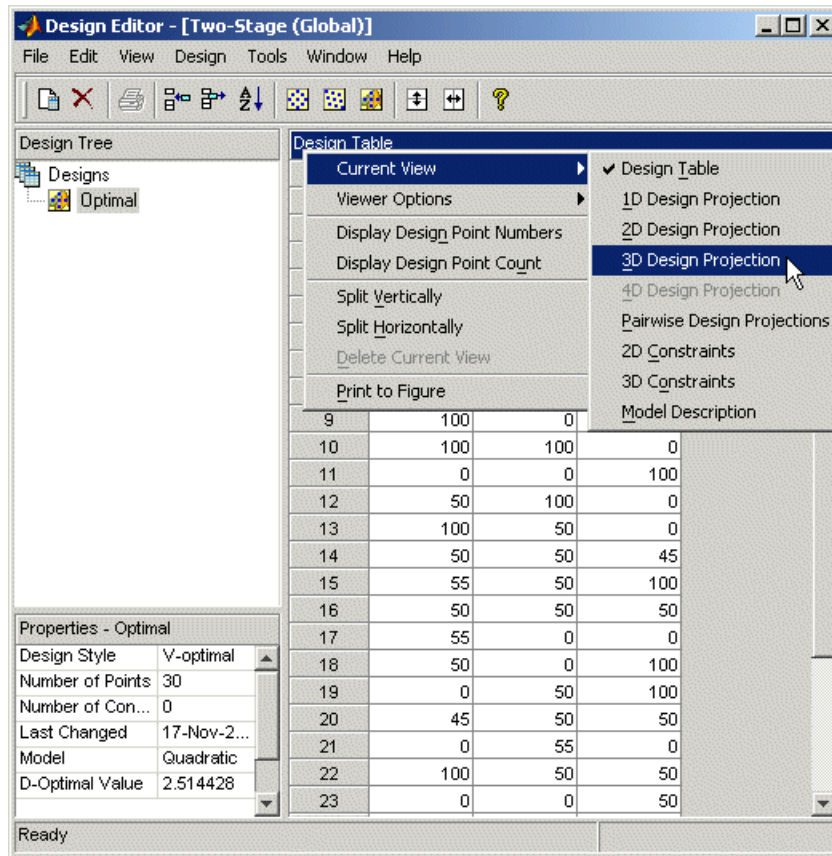
When you click the **OK** button on the Optimal Design dialog, the Optimizing Design dialog appears, containing a graph. This dialog shows the progress of the optimization and has two buttons: **Accept** and **Cancel**. **Accept** stops the optimization early and takes the current design from it. **Cancel** 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.

## View Design Displays

When you press the **Accept** button, you return to the Design Editor.



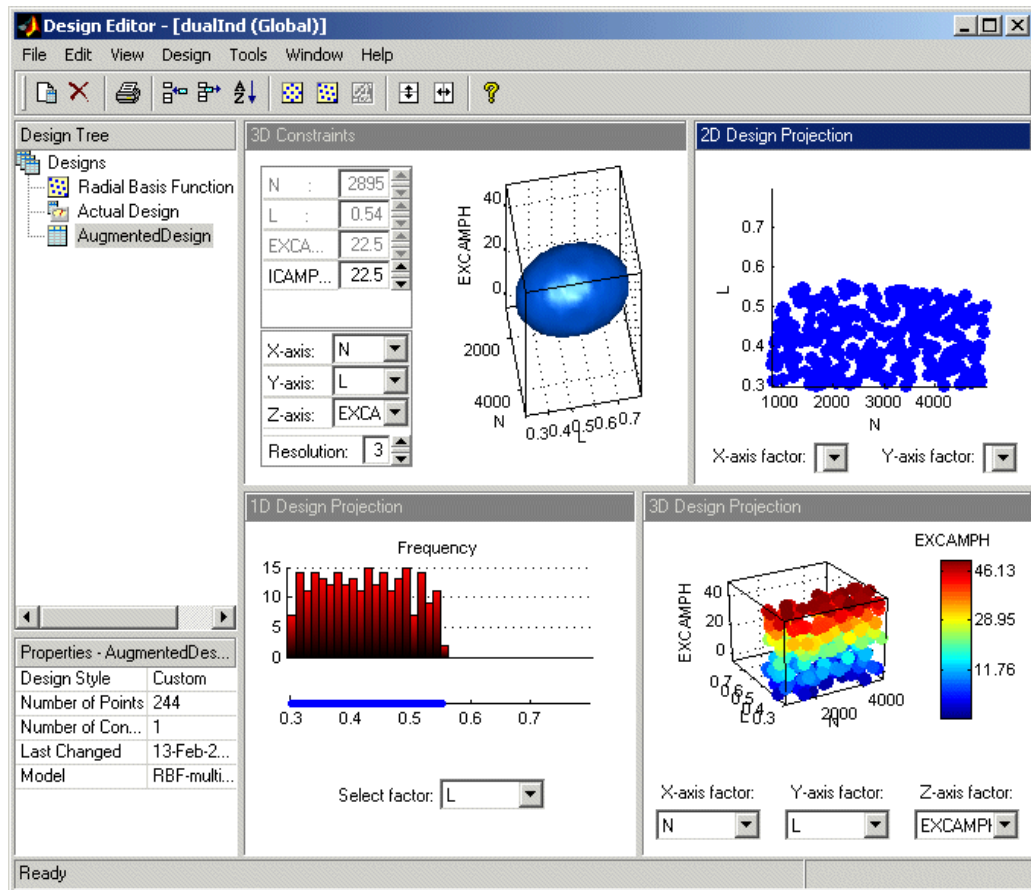
When you first see the main display area, it shows the default **Design Table** view of the design (see preceding example). There is a context menu, available by right-clicking on the title bar, in which you can change the view of the design to 1-D, 2-D, 3-D, 4-D, and pairwise design projections, 2-D 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”.

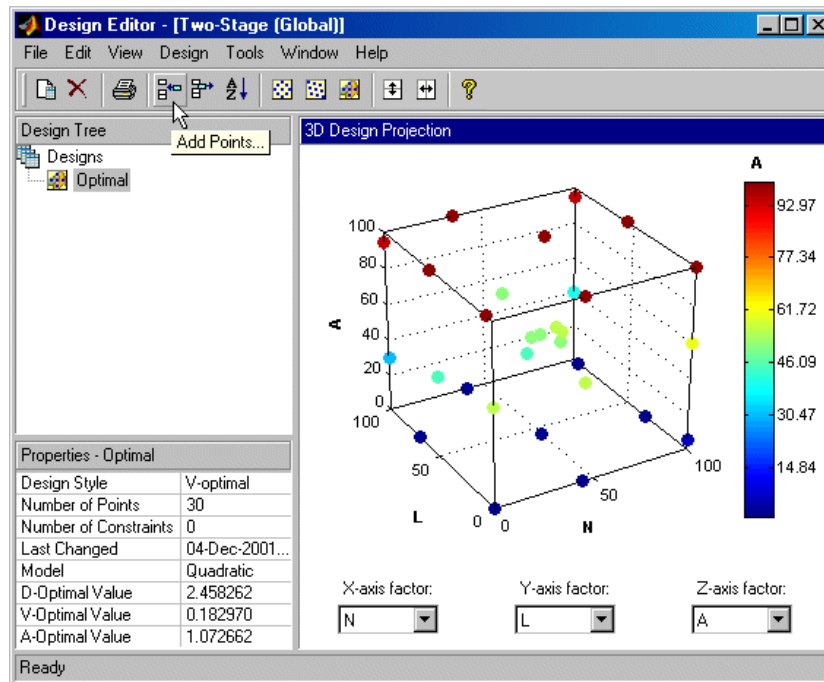
## Display Options

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 following example shows several views in use at once.



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.

## Use the Prediction Error Variance Viewer

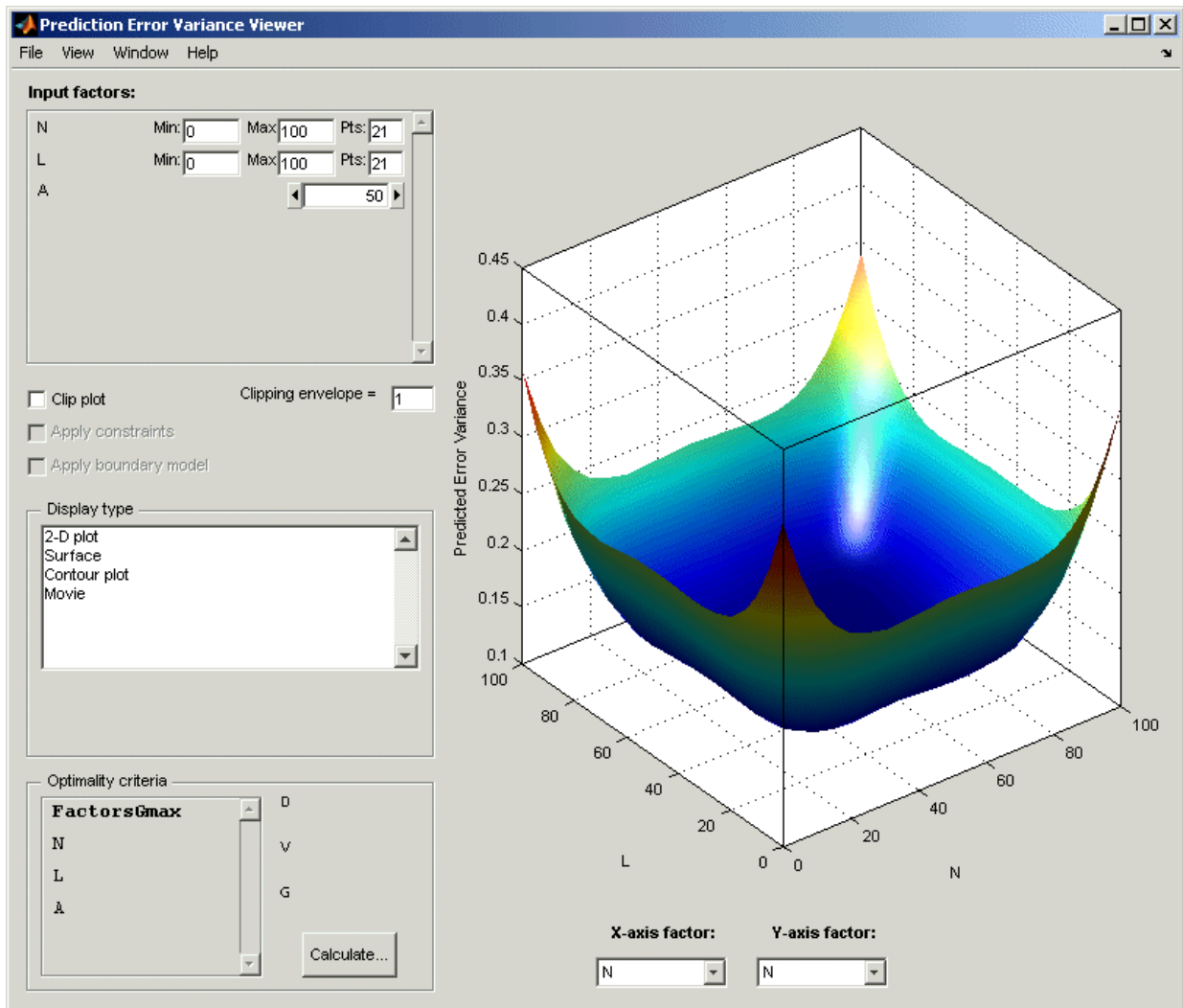
| In this section...  |
|---|
| “Introducing the Prediction Error Variance Viewer” on page 9-18 |
| “Improving the Design” on page 9-21                             |

### 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.

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).

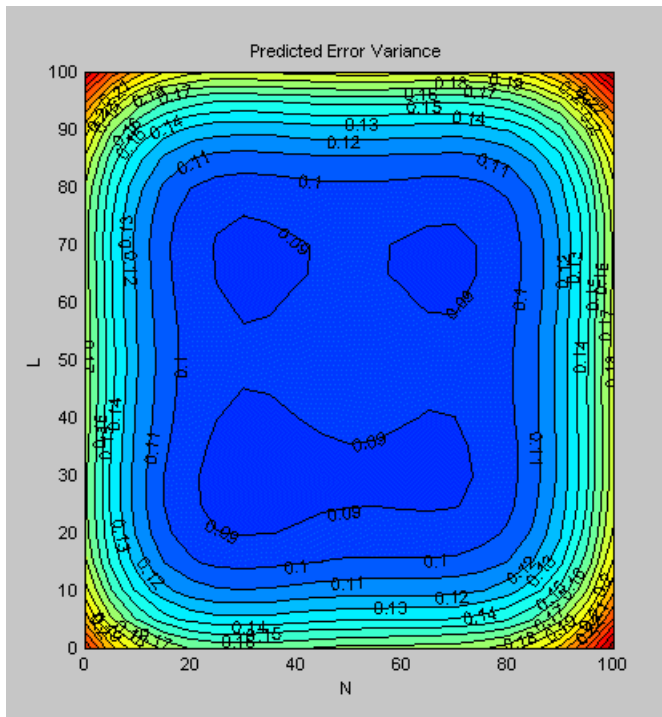
You can examine PEV for designs and models. The two are related in this way:

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.



## Improving the Design

You can further optimize the 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.

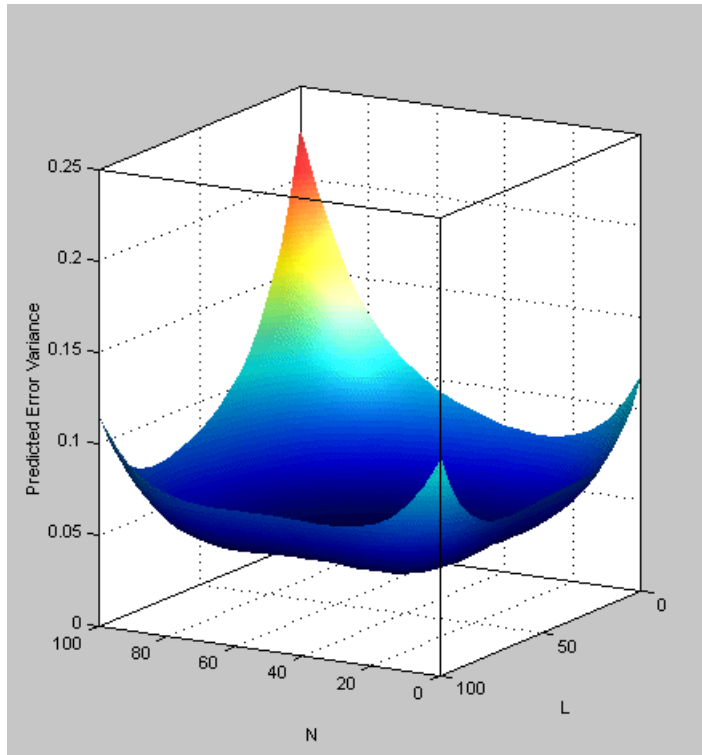
To strengthen the current optimal design:

- 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).



## Create Classical Designs

### In this section...

“Adding a Classical Design” on page 9-24


“Classical Design Browser” on page 9-26

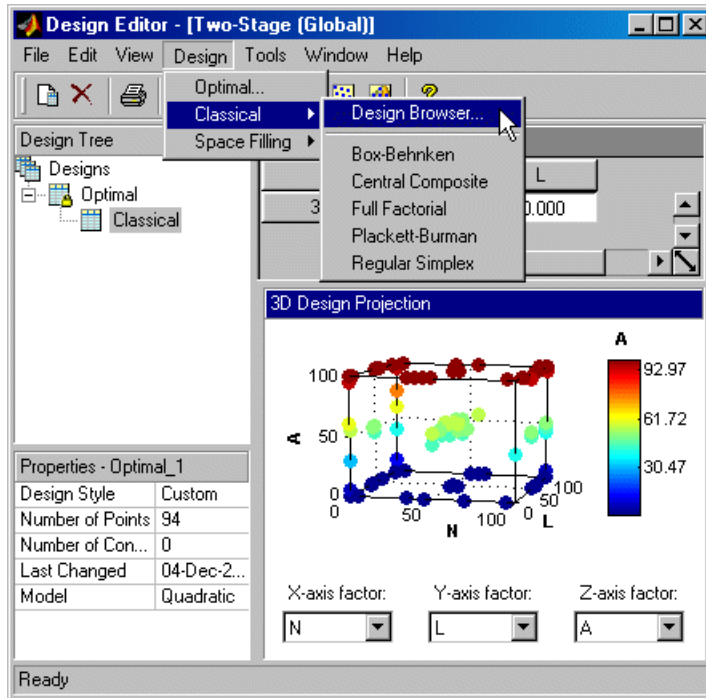
“Setting Up and Viewing a Classical Design” on page 9-27

### Adding a Classical Design

- 1 In the Design Editor window, select the Optimal design in the design tree by clicking.
- 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.

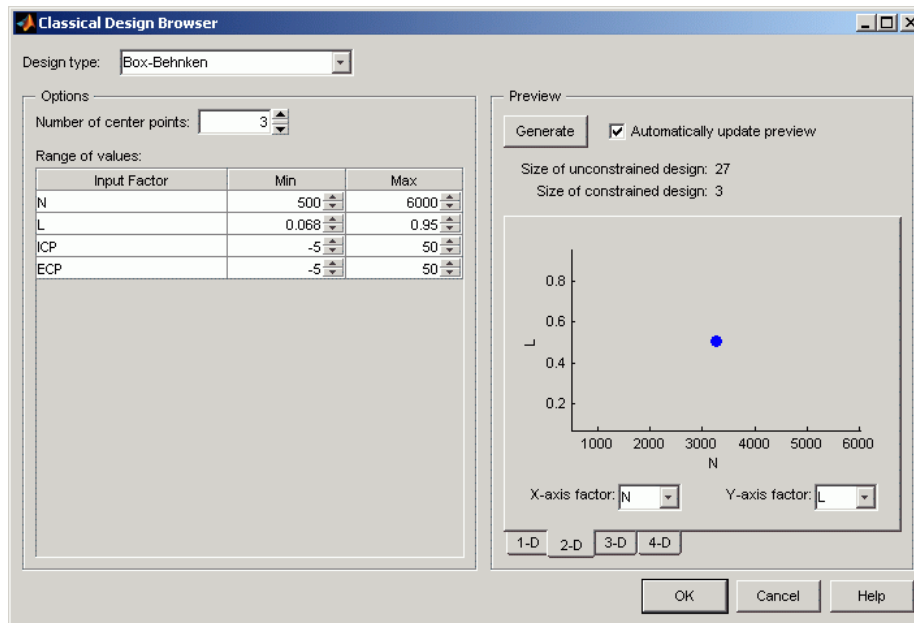
---

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**

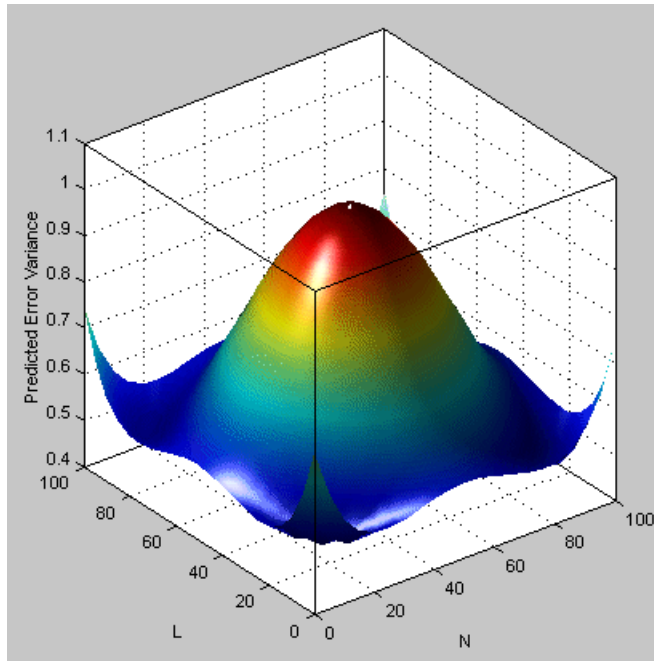
These are “screening” designs. They are two-level designs that are designed to allow you to work out which factors are contributing any effect to the model while using the minimum number of runs. For example, for a 30-factor problem this can be done with 32 runs.

- **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.



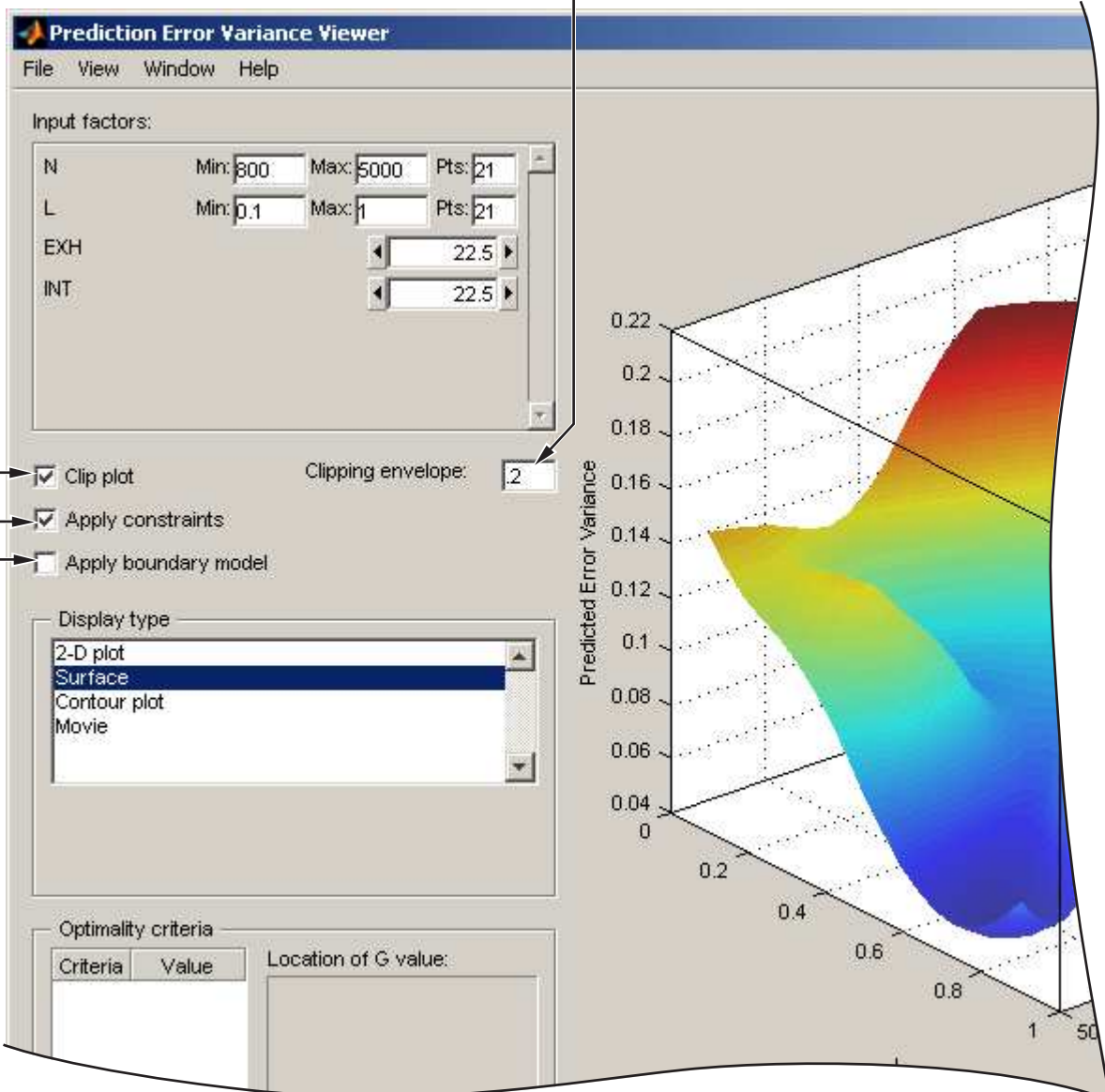
As you can see, this is not a realistic comparison, as this design has only 13 points (you can find this information in the bottom left of the main Design Editor display), whereas the previous optimal design had 100, but this is a good illustration of leverage. A single point in the center is very bad 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

## Use the Design Evaluation Tool

The Design Evaluation Tool is available for linear models only. See also “Global Model Class: Multiple Linear Models”.

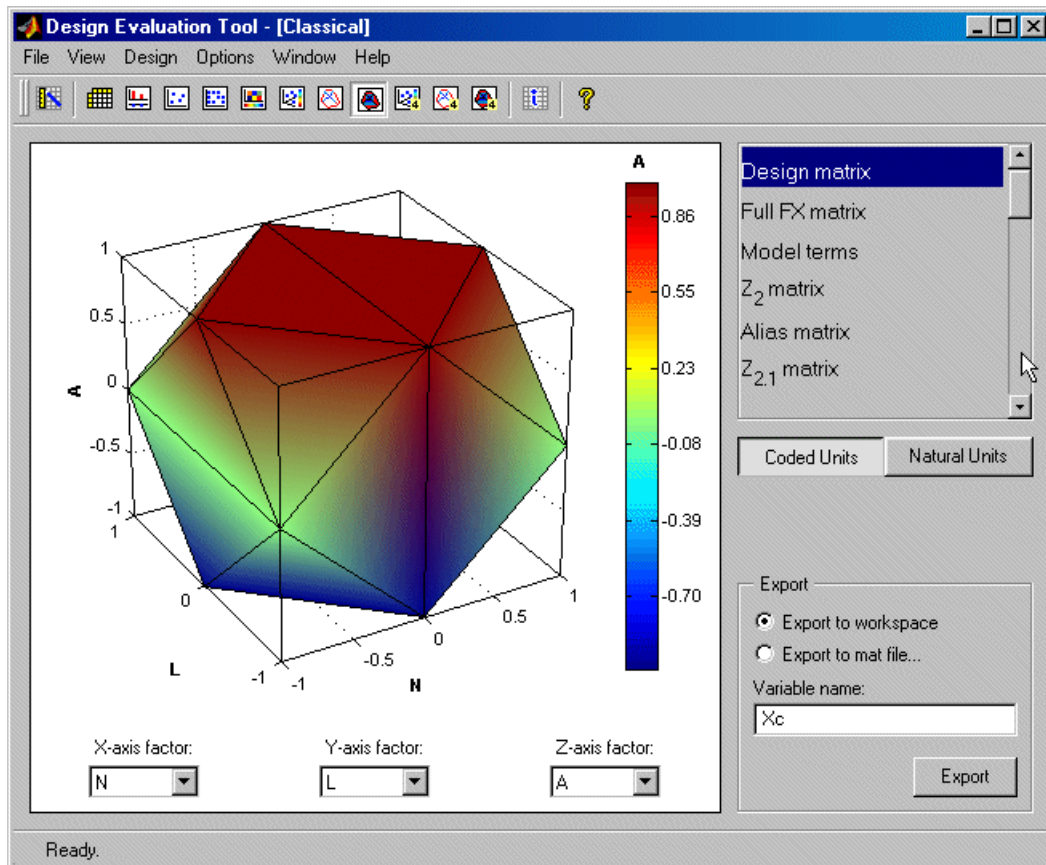
- 1 Return to the Design Editor and select **Tools > Evaluate Designs**.
- 2 Choose the **Box - Behnken** design and click **OK** in the Select Designs dialog.

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 9-30.

### 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.

## Create Space-Filling Designs

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 Add a new design by clicking the  button in the toolbar.

A new Classical child node appears in the tree. Select it by clicking. As before, the displays remain the same: the child node inherits all points from the parent design. Notice that in this case the parent node does *not* acquire a padlock to indicate it is locked — it is blue and therefore selected as the best design. Designs are locked when they are selected as best.

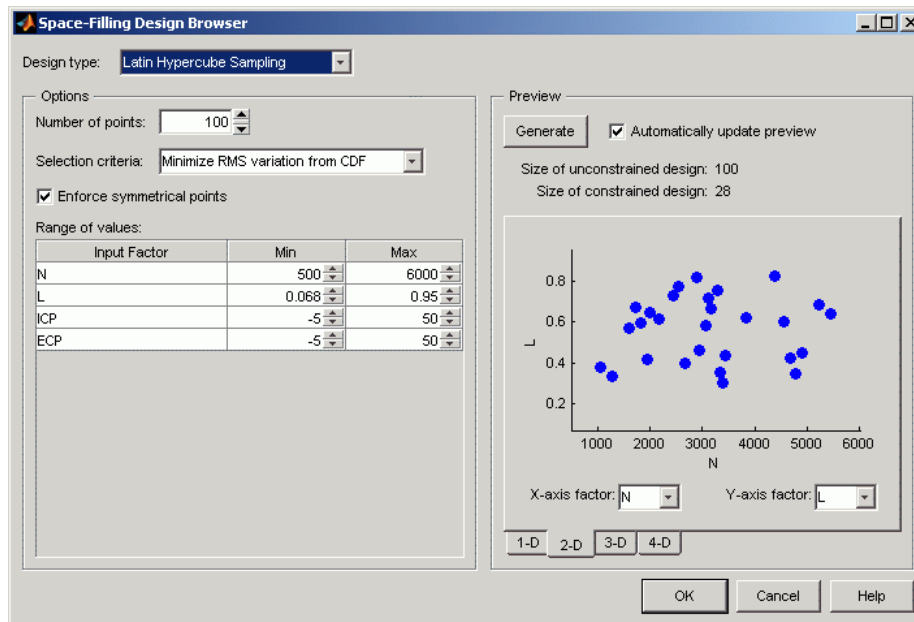
- 2 Rename the new node **Space Filling** (click again or press **F2**).
- 3 Select **Design > Space Filling > Design Browser**, or click the Space Filling Design button on the toolbar.
- 4 Click **OK** in the dialog to replace the current design points with a new design.

The Space Filling Design Browser appears.

---

**Note** As with the Classical Design Browser, you can select the three of design you can preview in the Space Filling Design Browser from the **Design > Space Filling** menu in situations when you already know the type of space-filling design you want.

---



- 1 Select Latin Hypercube Sampling from the **Design Style** drop-down menu.
- 2 Leave the default **Number of points**, and the default **Selection criteria**.
- 3 Observe the **Enforce Symmetrical Points** check box is selected by default. This creates a design in which every design point has a *mirror* design point on the opposite side of the center of the design volume and an equal distance away. Restricting the design in this way tends to produce better Latin Hypercubes.
- 4 Use the tabs under the display to view 2-D and 3-D previews.
- 5 Click **OK** to calculate the Latin Hypercube and return to the main Design Editor.
- 6 Use the Design Evaluation Tool and Prediction Error Variance Viewer to evaluate this design.



## 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).

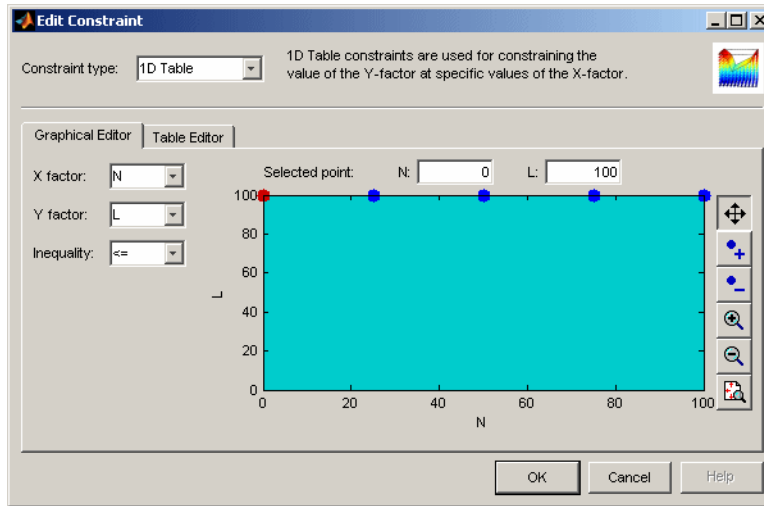
You would usually set up constraints *before* making designs. Applying constraints to classical and space-filling designs simply removes points outside the constraint. Constraining the candidate set for optimal designs ensures that design points are optimally chosen within the area of interest only.

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.

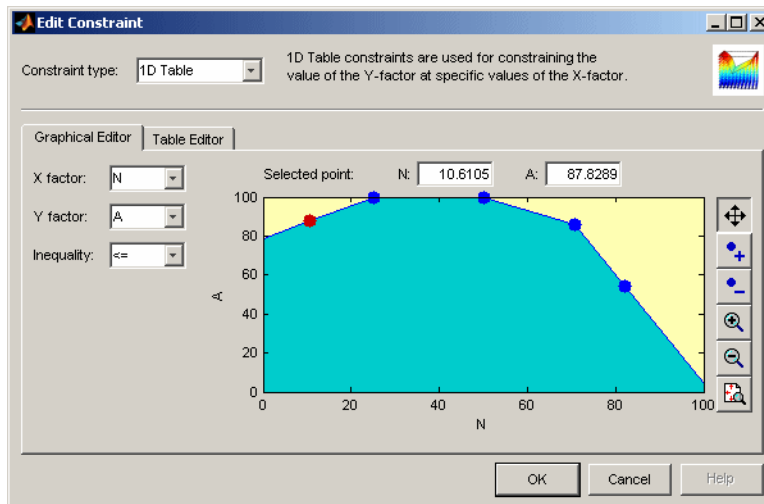
To add a constraint to your currently selected design:

- 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. The default **1D Table** is selected in the **Constraint Type** drop-down menu.



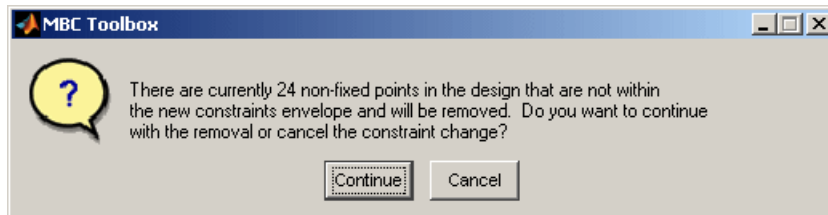
- 3 You can select the appropriate factors to use. For this example, choose speed (N) and air/fuel ratio (A) for the X and Y factors.
- 4 Move the large dots (click and drag them) to define a boundary. The Constraint Editor should look something like the following.



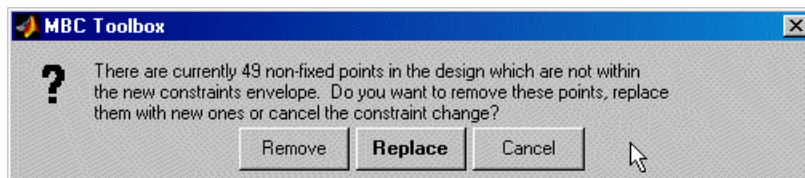
- 5 Click **OK**.

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.

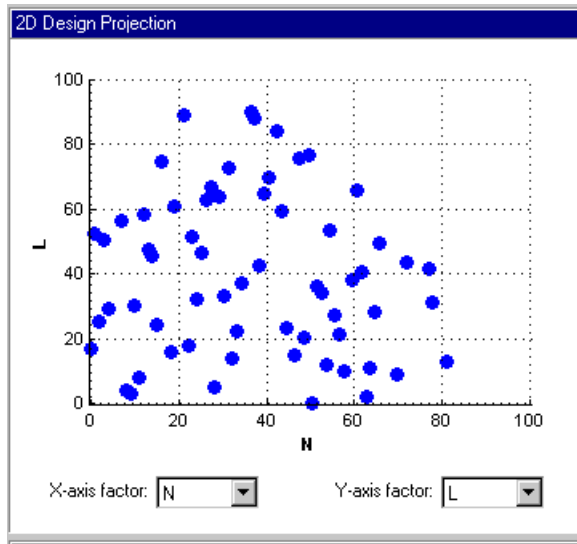
- You can click **Continue** to delete the points outside the constraint, or cancel the constraint. Note that fixed points are not deleted by this process.



- For *optimal* designs you see the following dialog, where you also have the option to replace the points with new ones chosen (optimally if possible) within the new candidate set.

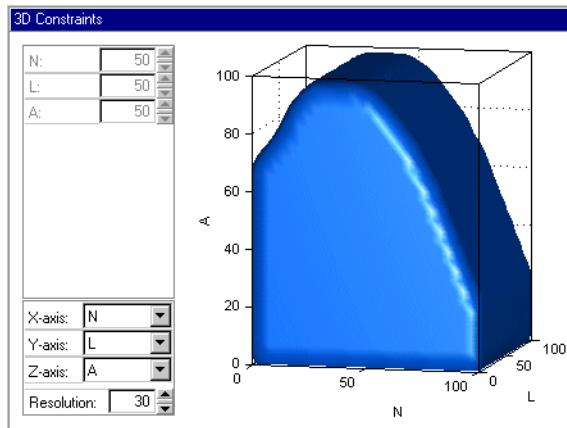


- 6 The default if you are constraining your space-filling design is to **Continue** and remove the points outside the new constraint area; choose this.

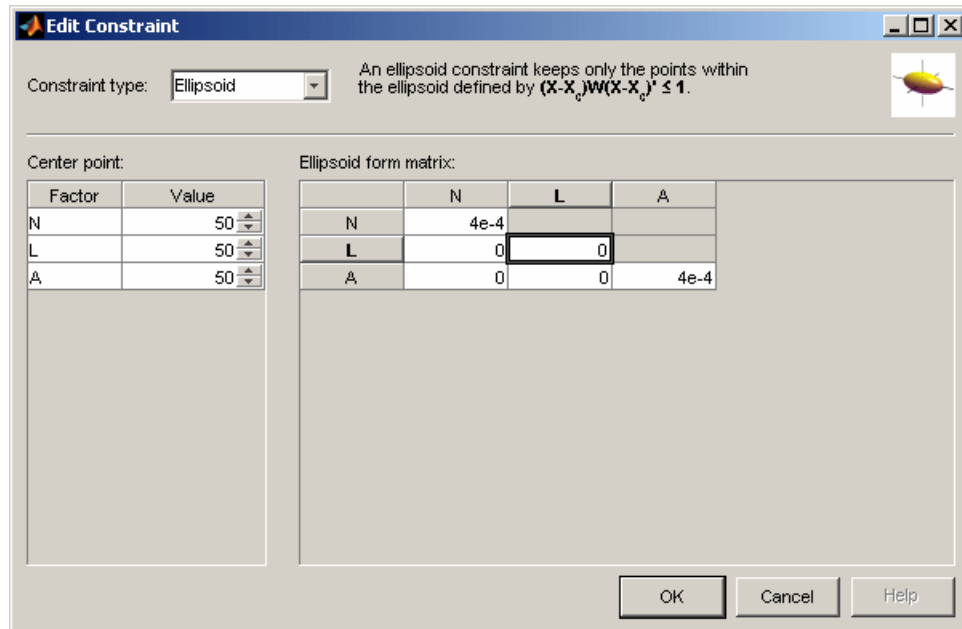


If you examine the 2-D projection of the hypercube, you will notice the effects of the new constraint on the shape of the design, as shown in the preceding example.

- 7 Right-click the display pane to reach the context menu, and select **Current View > 3D Constraints**.

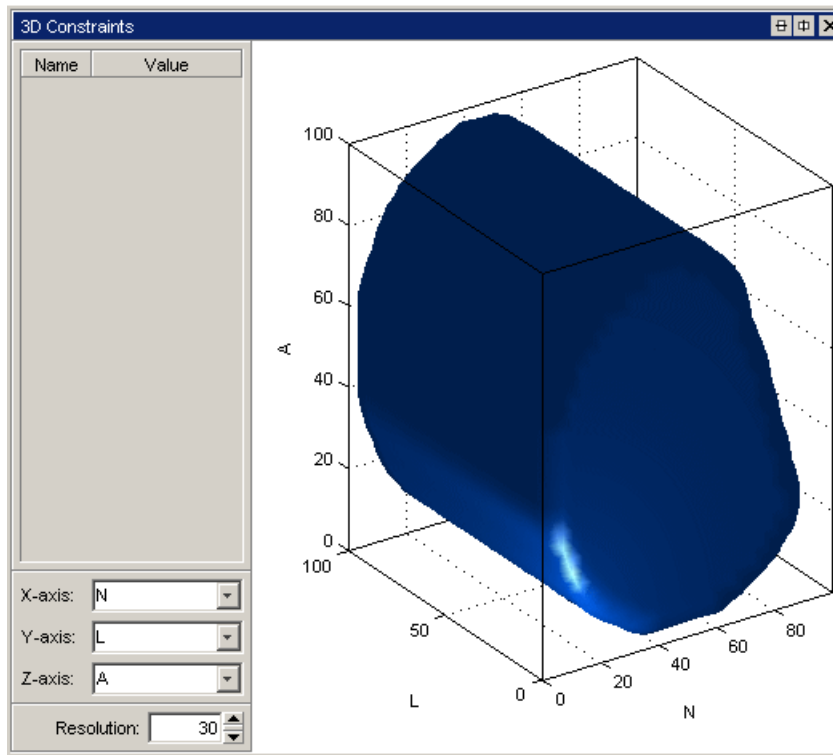


- 8 Return to the Constraint Editor, choose **Edit > Constraint**, and click **Add** in the Constraint Manager.
- 9 Add an ellipsoid constraint. Choose **Ellipsoid** from the drop-down menu of constraint types.



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 plot illustrated.



Both constraints are applied to this design.

## Save Designs

To save your design:

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

There are three **Export to** options:

- **Design Editor file** generates a Design Editor file (.mvd).
  - **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.
  - **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 Design Editor file.
  - 3 Choose the destination file by typing `Designtutorial.mvd` in the edit box.
  - 4 Click **OK** to save the file.





# Data Editor for Modeling

---

This section discusses the following topics:

- “Data Manipulation for Modeling” on page 10-2
- “Load Data” on page 10-3
- “View and Edit the Data” on page 10-6
- “Create New Variables and Filters” on page 10-12
- “Store Variables, Filters, and Plot Preferences” on page 10-17
- “Define Test Groupings” on page 10-19
- “Match Data to Experimental Designs” on page 10-23

## Data Manipulation 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 10-3
- “View and Edit the Data” on page 10-6
- “Create New Variables and Filters” on page 10-12
- “Store Variables, Filters, and Plot Preferences” on page 10-17
- “Define Test Groupings” on page 10-19
- “Match Data to Experimental Designs” on page 10-23

For comprehensive help on using data in the Model Browser, see “Data Manipulation for Modeling”.

## Load Data

### In this section...

“Entering the Data Editor” on page 10-3

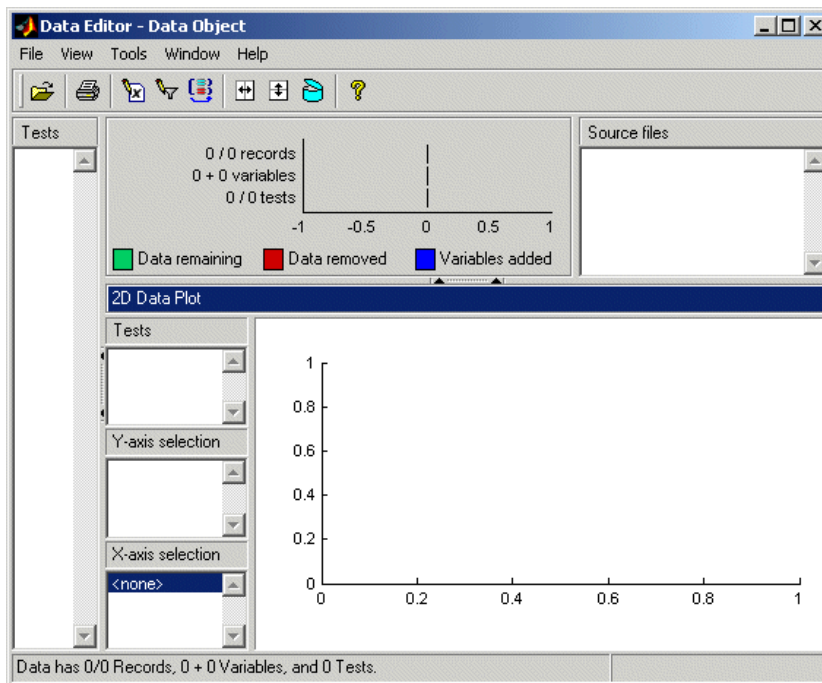
“Loading a Data File” on page 10-4

## Entering 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.

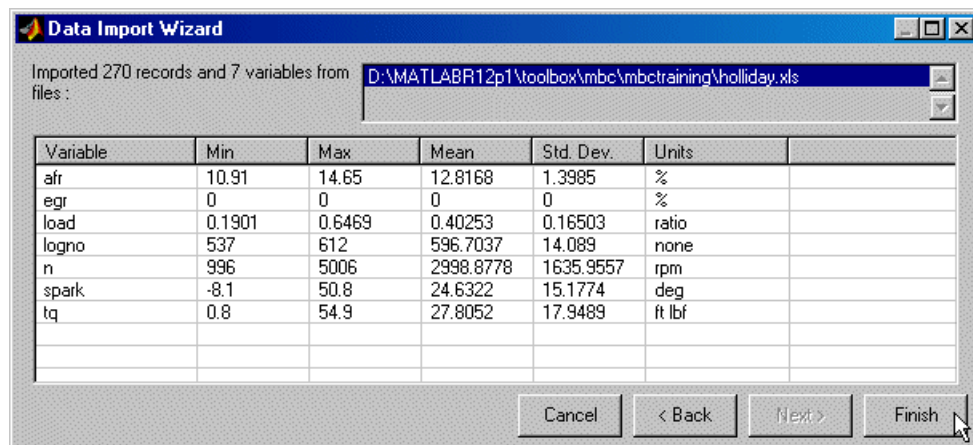
By default the new data object is called **Data Object**. Select **File > Rename Data** to enter a new name.

## Loading a Data File

- 1 Click the Open File icon in the toolbar  to load data from a file.

The Data Import Wizard appears to select a file.

- 2 Use the Browse button to find and select the `Holiday.xls` data file in the `mbctraining` folder. Double-click to load the file. You can also enter the file pathname in the edit box. The pop-up menu contains the file types recognized by the Model Browser (Excel File, Delimited Text File, MATLAB Data File). Leave this at the default, Auto. This setting tries to determine what type of file is selected by looking at the file extension.
- 3 Click **Next**.



- 4 The Data Import Wizard displays a summary screen showing the total number of records and variables imported, and you can view each variable's range, mean, standard deviation, and units in the list box. You can double-click variables in the

list to edit names and units. Click **Finish** to accept the data. (If you have data loaded already, you cannot click **Finish** but must continue to the data merging functions.)

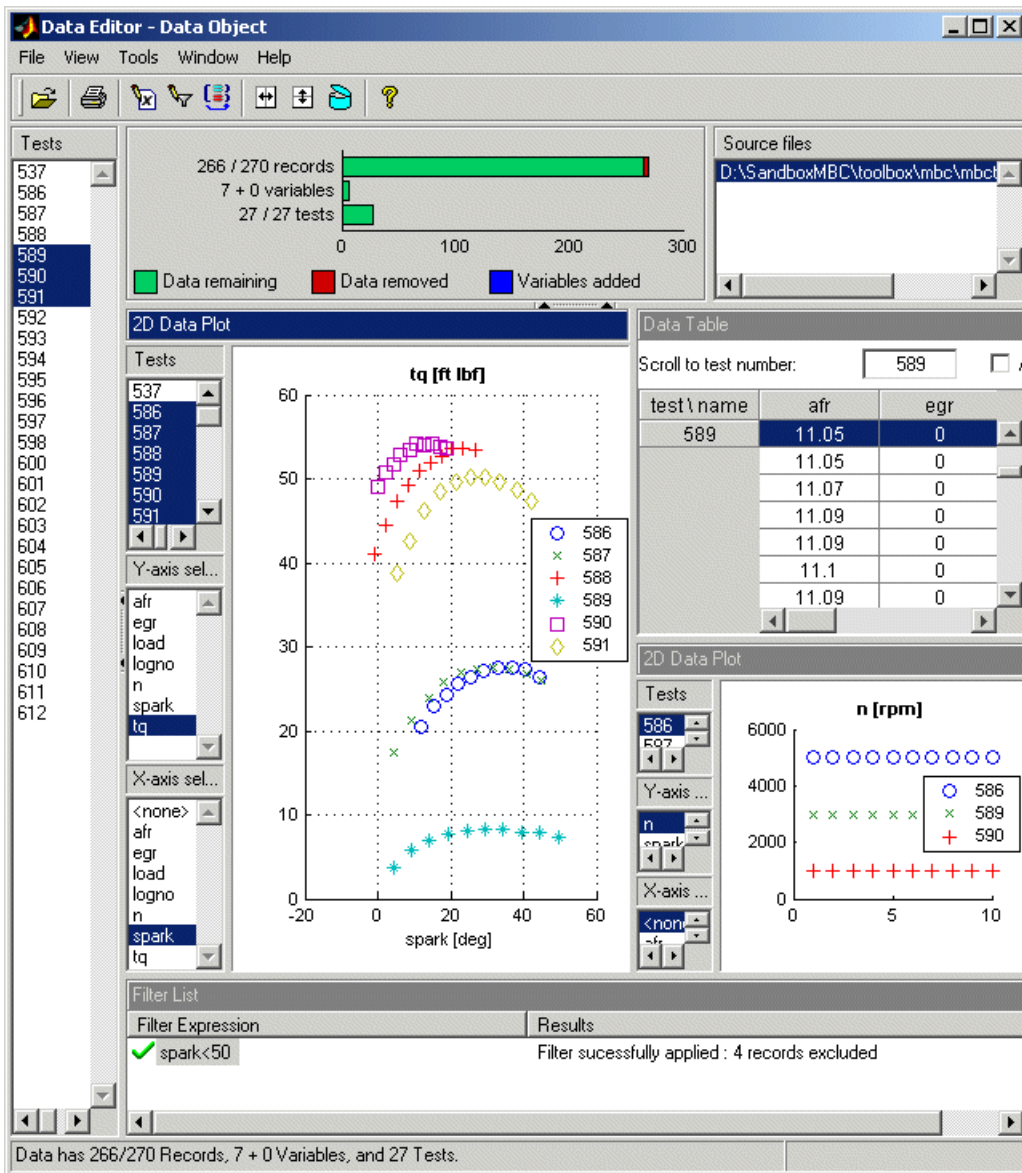
The Data Import Wizard disappears and the view returns to the Data Editor, which now contains the data you just loaded.

## View and Edit the Data

| In this section...                                   |
|--|
| “Viewing Data” on page 10-6                          |
| “Using Notes to Sort Data for Plotting” on page 10-8 |
| “Removing Outliers and Problem Tests” on page 10-9   |
| “Reordering and Editing Data” on page 10-10          |

### Viewing Data

As shown in the following figure you can split the views to display several plots at once, as in the Design Editor. You can 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, cluster plots, data tables, and list views of filters, variables, test filters, test notes and cluster information.



In the 2-D plot view, the list boxes on the left allow a combination of tests and variables to be plotted simultaneously. The example shown plots torque against spark for multiple

tests on the left, and speed for three selected tests on the right. You can multiple-select tests and y-axes to compare the data in the tests (hold down **Shift** or **Control**). 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 **Viewer Options > 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 Right-click the view and select **Split View > Test Note Definitions**.

The current view is divided into two.

- 6 Select **Tools > Test Notes > Add**.

The Test Note Editor appears.

- 7 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**.
- 8 Right-click the **Test Notes List** view and select **Split View > Notes View**.

The current view is split into two. You can sort records by notes in the new **Notes** view.

- 9 Click the column header of the new **Low torque** note in the **Notes** view. All the tests that satisfy the condition **mean(tq)<10** are sorted to the top of the list.
- 10 Now create some more views.
  - Right-click a view and select **Split View > Data Table**.
  - Right-click a view and select **Split View > 3D Data Plot**.



- 11 In the **Notes** view, 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 > Filters > Remove Outliers** (or use the keyboard shortcut **Ctrl+A**). Select **Tools > Filters > Restore Outliers** (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 **Split View > Test Filter Definitions**.
  - b Select **Tools > Test Filters > Add**.
  - c The Test Filter Editor appears. Enter `mean(tq)>10` to keep all tests where the mean torque is greater than 10, and click **OK**.

In the new **Test Filter List** view, you should see the new test filter 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 10-13.

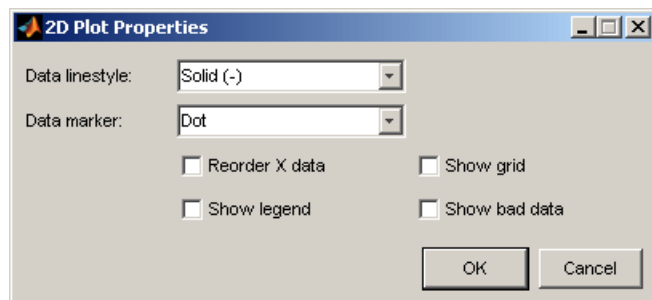
- 3 To view removed data in the table view, right-click and select **Viewer Options > Allow Editing**. Removed records are red. To view removed data in the 2-D and Multiple Data Plots, select **Viewer Options > Properties** and select the box **Show bad data**.

## Reordering and Editing Data

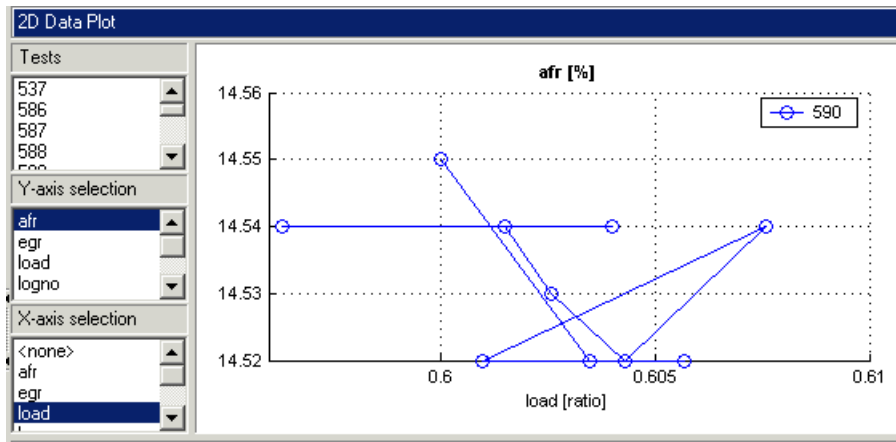
To change the display, right-click a 2-D plot and select **Viewer Options > Properties**. You can alter grid and plot settings including lines to join the data points.

**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 **Viewer Options > Properties** and choose **solid** from the **Data Linestyle** drop-down menu, as shown below. Click **OK**.

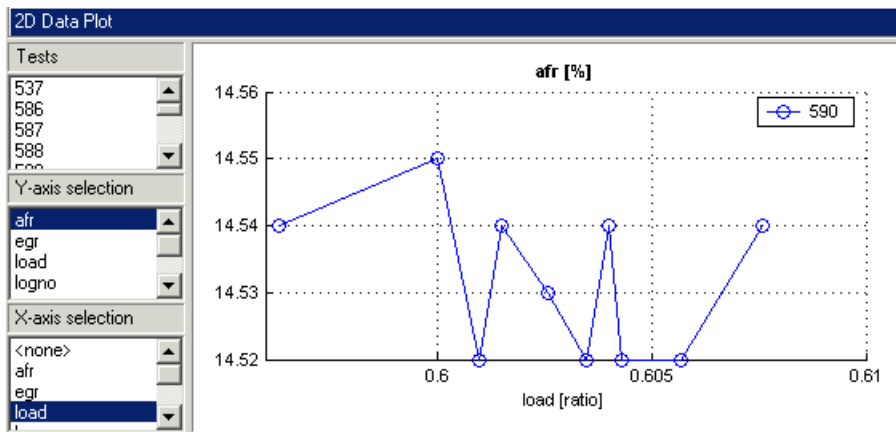


- 3 Choose **afr** for the *y*-axis.
- 4 Choose **Load** for the *x*-axis.
- 5 Select test 590. You must use the test controls contained within the 2-D plot. The **Tests** pane on the left applies to other views: tables and 3-D and multiple data plots.



- 6 Right-click and select **Viewer Options > 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 **Viewer Options > Allow Editing**, and then you can double-click cells to edit them.

## Create New Variables and Filters

### In this section...

“Adding New Variables” on page 10-12

“Applying a Filter” on page 10-13


“Sequence of Variables” on page 10-15

“Deleting and Editing Variables and Filters” on page 10-16

### Adding New Variables

You can add new variables to the data set.

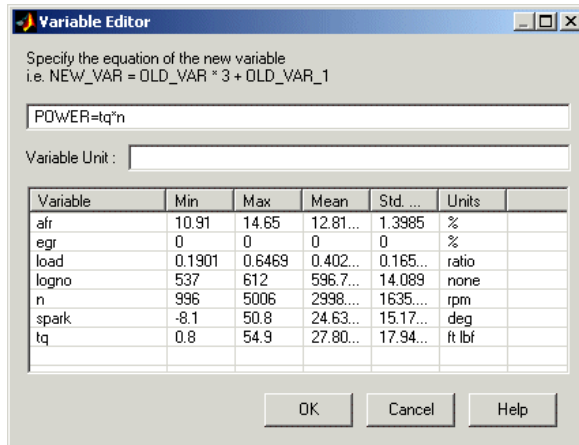
- 1 Select **Tools > Variables > Add**.

Alternatively, click the  toolbar button.

The Variable Editor appears.

You can define new variables in terms of existing variables. You define the new variable by writing an equation in the edit box at the top of the Variable Editor dialog.


- 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.



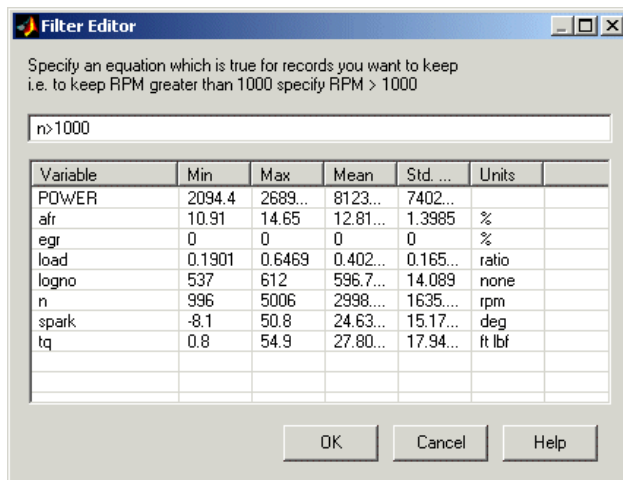
- 3 Click **OK** to add this variable to the current data set.
- 4 This new variable can be seen in the Data Editor by right-clicking in a view and selecting **Split Plot > Variable Definitions**. A new view appears containing a list of your user-defined variables. You can also now see **7 + 1 variables** next to the top information bars.

## 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.

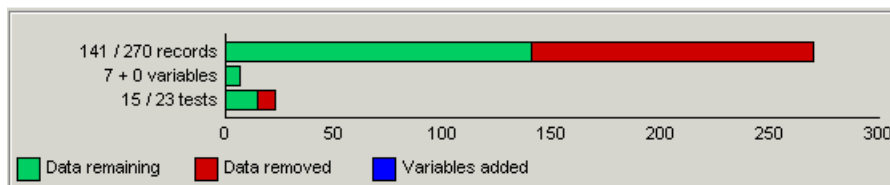
- 1 Choose **Tools > Filters > Add**, or click the  button in the Data Editor window.

The Filter Editor dialog appears.



You define the filter using logical operators on the existing variables.

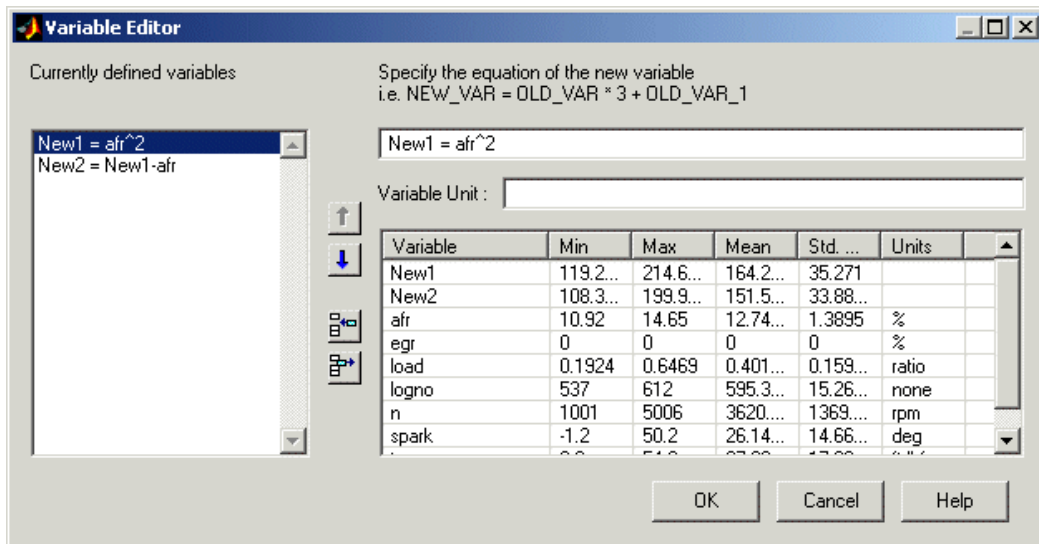
- 2 Keep all records with speed ( $n$ ) greater than 1000. Type  $n$  (or double-click on the variable  $n$ ), then type  $>1000$ .
- 3 Click **OK** to impose this filter on the current data set.
- 4 This new filter can be seen in the Data Editor by right-clicking in a view (try the **Variable List** view) and selecting **Split Plot > Filter List**. A new view appears containing a list of your user-defined filters and information on how many records are removed by the new filter. You can also now see 141/270 records next to the top information bars and a red section 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 arrow buttons.

Select **Tools > Variables > Edit** to open the Variable Editor.

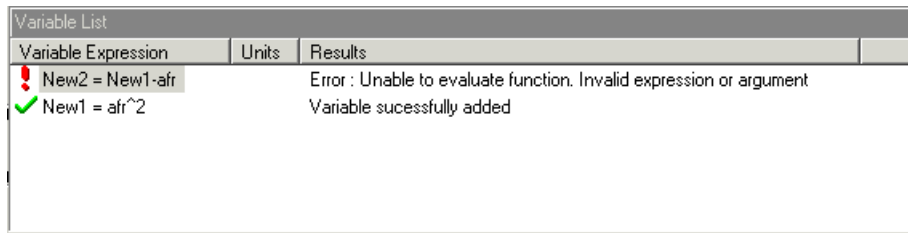


Example:

- 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 variable list view you see the following error message:



| Variable Expression | Units | Results   |
|---------------------|-------|---|
| New2 = New1-afr     |       | Error : Unable to evaluate function. Invalid expression or argument |
| New1 = afr^2        |       | Variable sucessfully added  |

- 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 a **Variable List** view 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 a Filter List view and using the **Delete** key.


You can also edit current user-defined variables and filters using the relevant menu items or toolbar buttons.

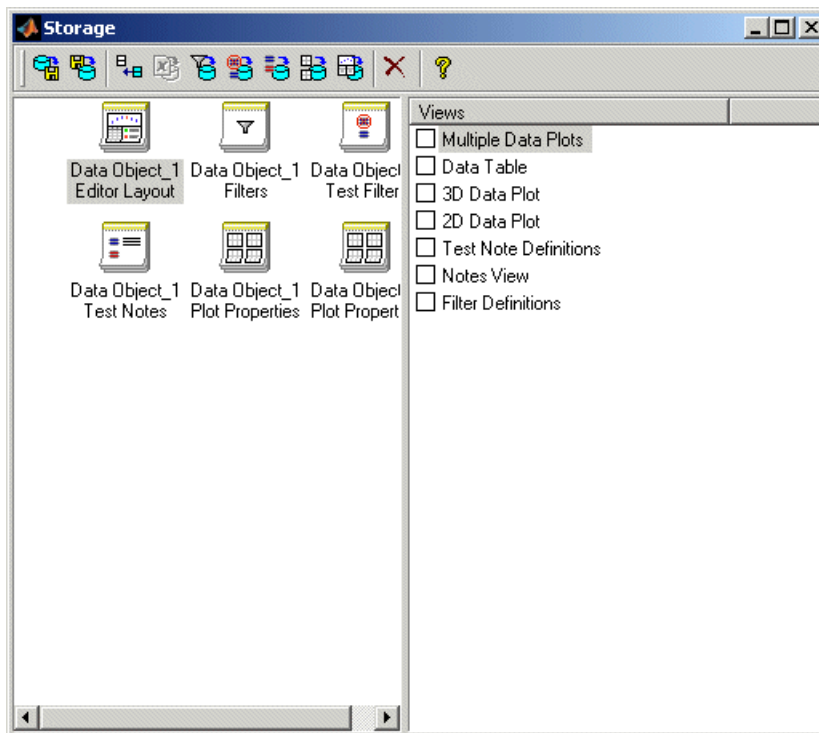


## Store Variables, Filters, and Plot Preferences

Storage allows you to store 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.

You can open the Storage window from the Data Editor window in either of these ways:

- Using the menu **Tools > Open Storage**
- Using the toolbar button 



The example above contains a variety of stored objects. The toolbar buttons Store Current Variables and Store Current Filters, Test Filters or Test Notes allow you to put all user-defined variables and filters from the current session into storage. They appear in the Storage window. All stored user-defined variables and filters appear here regardless of which project is open — once created and brought into storage, they remain

there. If you do not delete them, they are there indefinitely. You can also store view settings with the toolbar button Store Current Data Editor Layout.

The Data Editor retains memory of your plot type settings and when reopened will display the same types of views. You can also use Store Current Data Plots to save the details of your Multiple Data Plots, such as which factors to display, line style, grid, etc.

You can double-click any item in storage to append the object to the current views. For example if you double-click a Data Editor Layout object, the current views will be replaced by the saved views. Other objects add items to the current views.

You can select Export to File to send the stored objects to a file. You might do this to move the objects to a different user or machine. Select Import from File to bring such variables and filters into storage, and use Append Stored Object to add items from storage to your current project.


- 1** Use the controls to bring the variable POWER and the filter you just created into storage.
- 2** Close the Storage window.

For a detailed description of the functionality in Storage, see “Store Variables and Filters”.

## 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 > Change Test Groupings**
- Using the toolbar button 


When you enter the dialog, 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 as seen in the following example. 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.

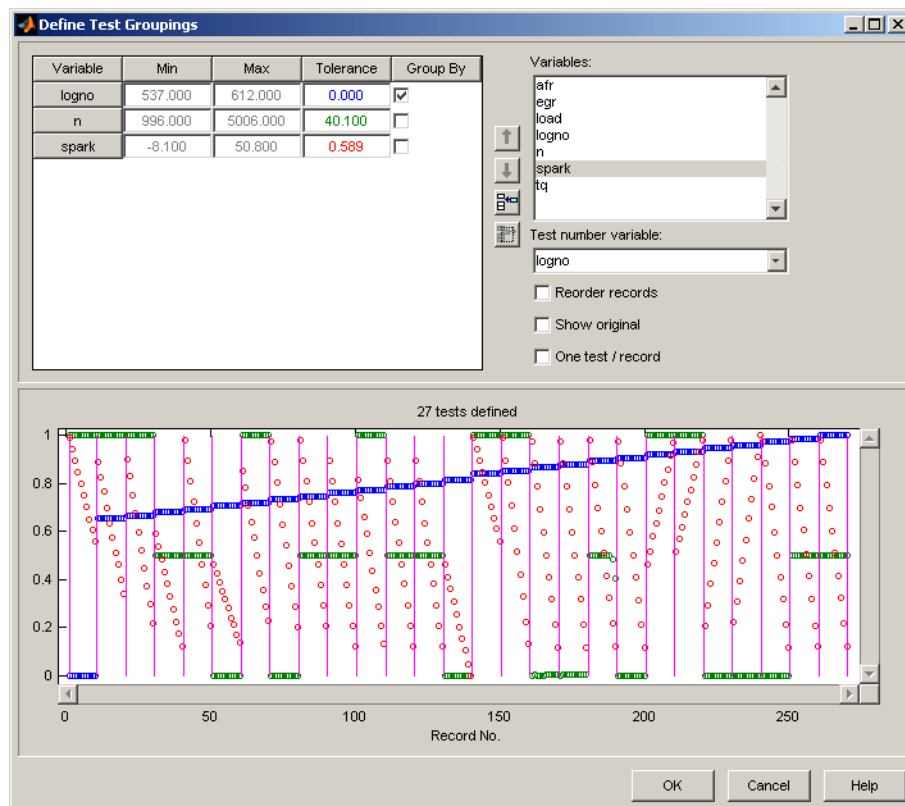
- 3 Clear the box **Group by** for `logno`. Notice that variables can be plotted without being used to define groups.
- 4 Add `load` to the list by selecting it on the right and clicking .
- 5 Change the `load` tolerance to 0.01 and watch the test grouping change in the plot.
- 6 Clear the **Group By** 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, as shown below. 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.



**Reorder records** allows records in the data set to be reordered before grouping. Otherwise, the groups are defined using the order of records in the original data object.

**Show original** displays the original test groupings if any were defined.

**One test/record** 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.

**Test number variable** contains a pop-up menu showing 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.

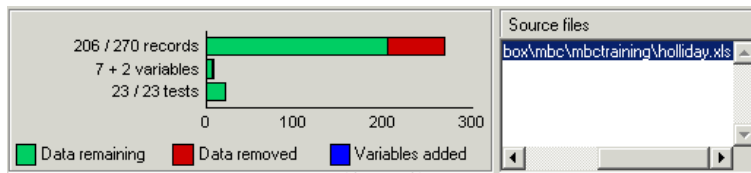
If you chose **none** from the **Test number variable** list, the tests would be numbered 1,2,3 and so on in the order in which the records appear in the data file. With **logno** chosen, you will see tests in the Data Editor listed as 586, 587 etc.

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 dismiss the dialog.

You return to the Data Editor window. At the top is a summary of this data set now that your new variable has been added and a new filter applied (example shown below).



- 11** 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 10-23

“Tolerances and Cluster Information” on page 10-26

“Understanding Clusters” on page 10-29

## Introducing Matching Data to Designs

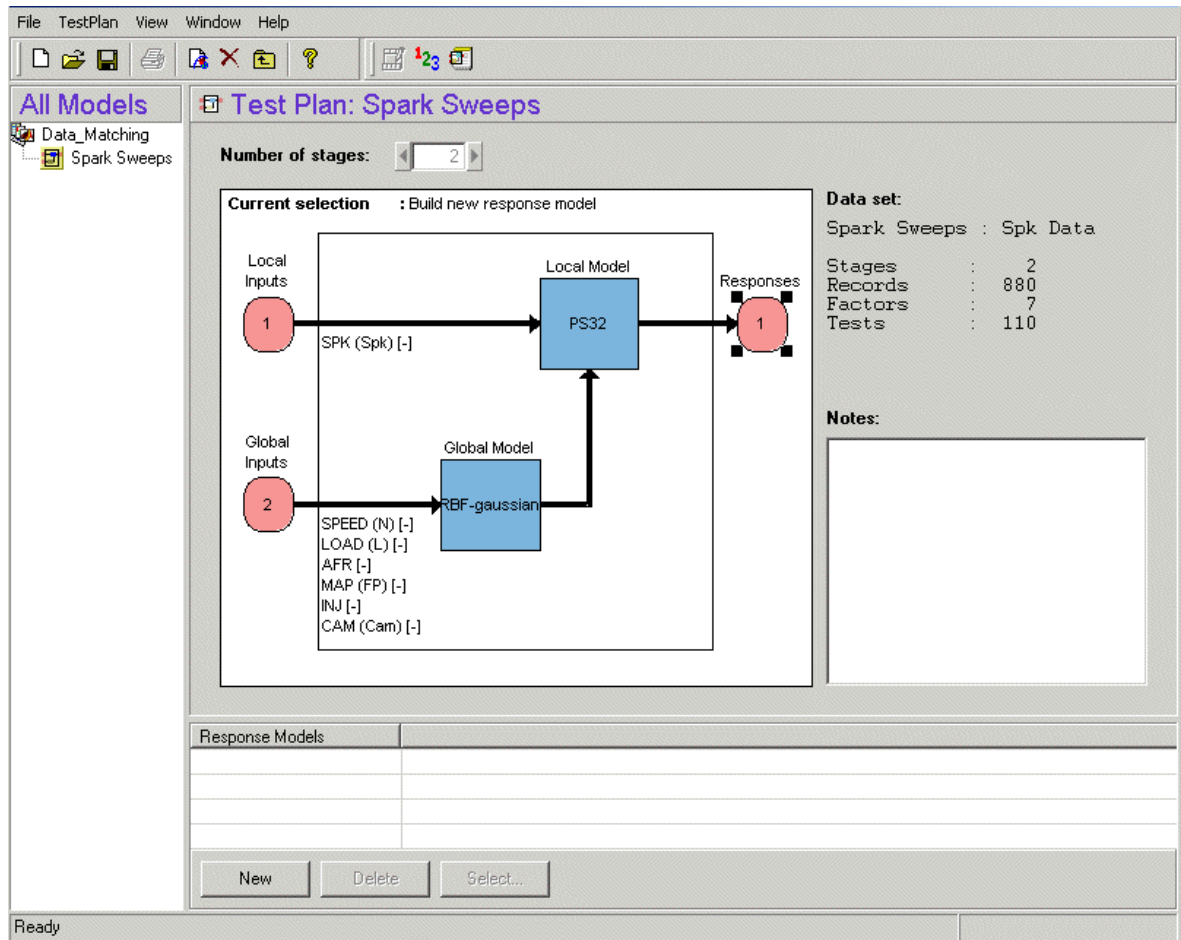
We provide an example project to illustrate the process of matching experimental data to designs.


Experimental data is unlikely to be identical to the desired design points. You can use the Cluster Plot 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, select **File > Open Project** and browse to the file `Data_Matching.mat` in the `mbctraining` directory.
- 2 Click the **Spark Sweeps** node in the model tree to change to the test plan view, as shown.

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.

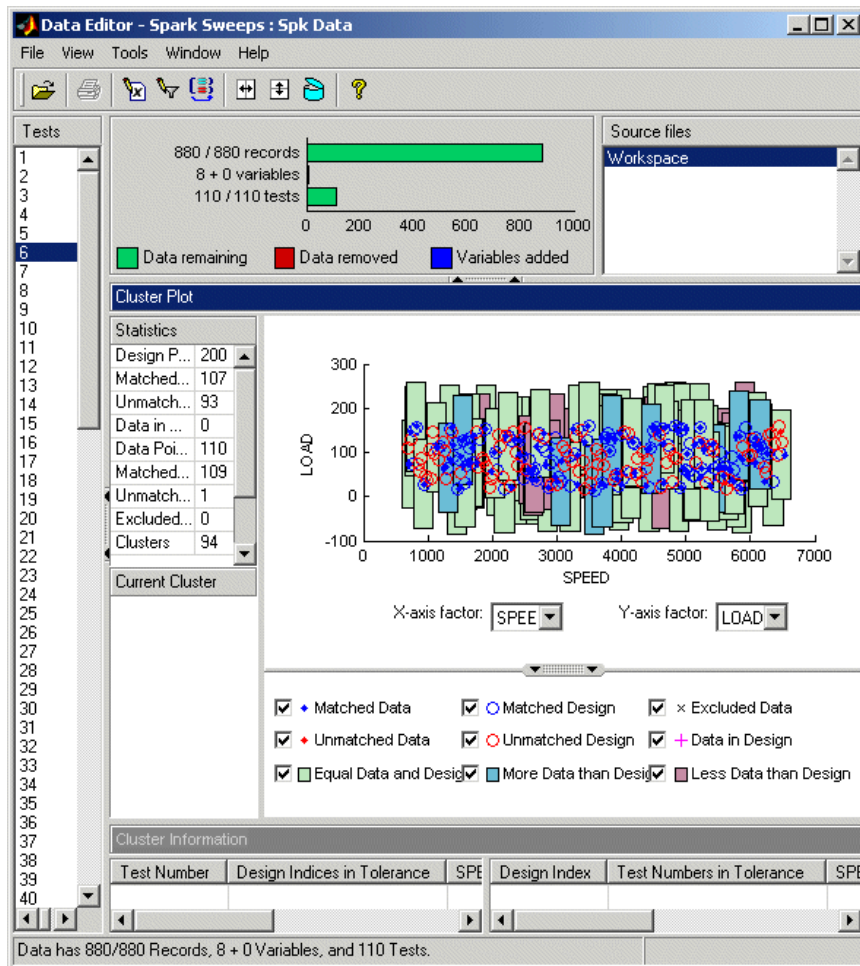


- 3 Click the Select Data button (  ) in the toolbar.

The Data Editor appears.

- 4 You need a **Cluster View** to examine design and data points. Right-click a view in the Data Editor and select **Current View > Cluster View**.

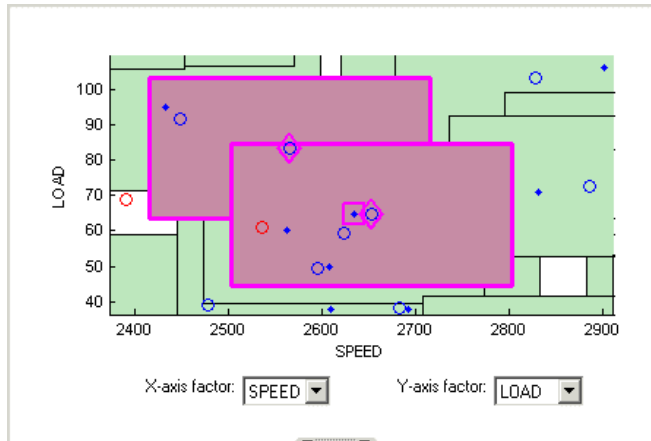




In the Cluster Plot you can see colored areas containing points. These are “clusters” where closely matching design and data points have been selected by the matching algorithm.

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.



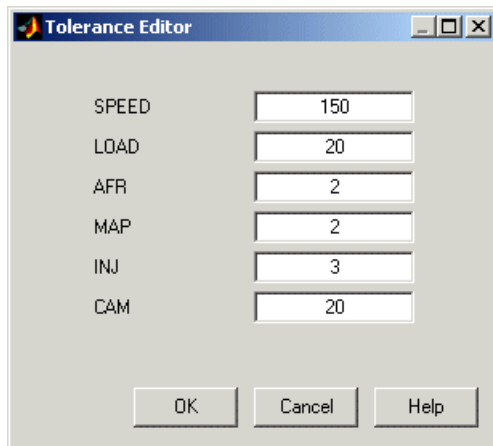
Notice the shape formed by overlapping clusters. The example shown outlined in pink is a single cluster formed where a data point lies within tolerance of two design points.

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 10-29.

## Tolerances and Cluster Information

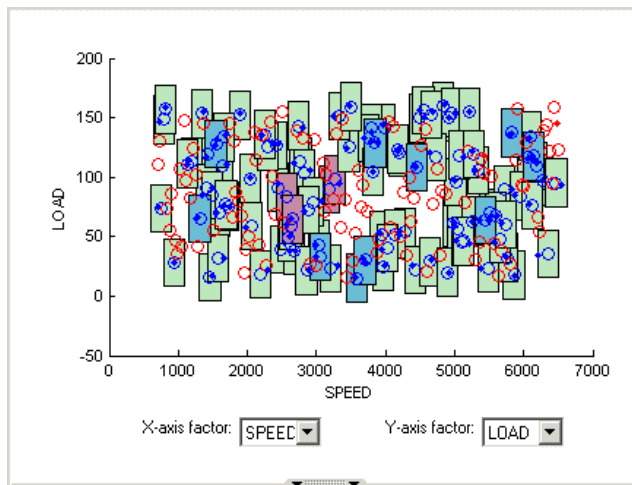
- 1 To edit tolerance values, select **Tools > Tolerances**.

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.

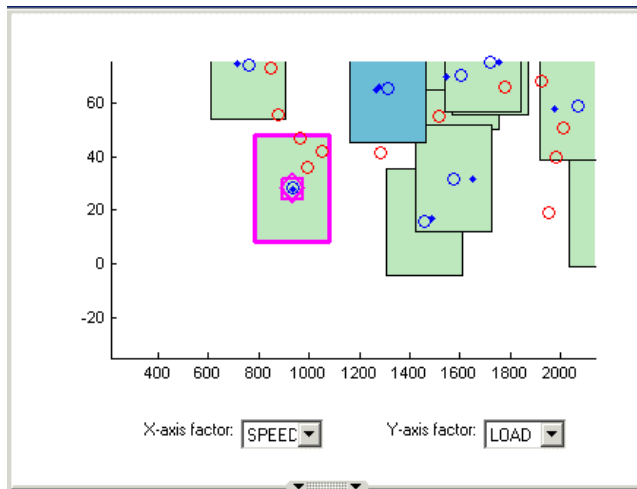


- 2 Click the **LOAD** edit box and enter 20, as shown. Click **OK**.

Notice the change in shape of the clusters in the **Cluster Plot** 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.



- 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.

You need to ensure you are displaying a **Cluster Information** list view to select or exclude points. The Data Editor retains memory of previous data views and if you had a cluster plot in your saved settings then this plot is used.

- If you do not already have a Cluster Information list view displayed, right-click the **Cluster Plot** view and select **Split Vertically**. A new view appears underneath the cluster plot. Right-click the new view and select **Current Plot > Cluster Information**.

| Cluster Information                 |                             |       |       |    |                                     |                           |       |       |    |  |
|-------------------------------------|-----------------------------|-------|-------|----|-------------------------------------|---------------------------|-------|-------|----|--|
| Test Number                         | Design Indices in Tolerance | SPEED | LOAD  | AI | Design Index                        | Test Numbers in Tolerance | SPEED | LOAD  | AI |  |
| <input type="checkbox"/>            | 22, 137                     | 2634  | 64.77 | 1  | <input checked="" type="checkbox"/> | 22                        | 2653  | 64.55 | 1  |  |
| <input checked="" type="checkbox"/> | 52, 137                     | 2562  | 60.23 | 1  | <input checked="" type="checkbox"/> | 52                        | 2624  | 59.45 | 1  |  |
|                                     |                             |       |       |    | <input checked="" type="checkbox"/> | 137                       | 2565  | 83.49 | 1  |  |

Notice that the **Cluster Information** list view 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 view 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 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.

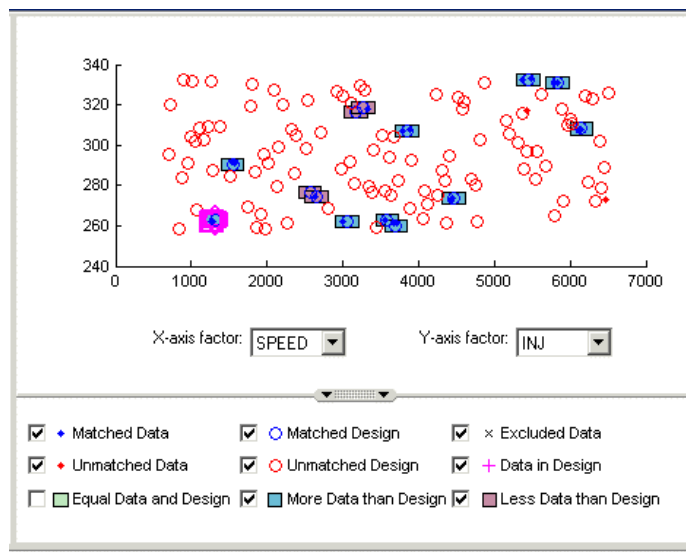
- Blue clusters have more data points than design points. All the data points will replace the design points in the **Actual Design**.
- Red clusters have more design points than data points. These data points will not be added to your design as the algorithm cannot choose which design points to replace, so you must manually make selections to deal with red clusters if you want to use these data points in your design. The example **Cluster Information** list view shows a selected red cluster with more design than data points.

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 **Cluster Plot** and select **Viewer Options > Select Unmatched Data**. Notice that the remaining unmatched data points appear in the **Cluster Information** list view. 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 **Cluster Plot** view. You control what is plotted using these check boxes.

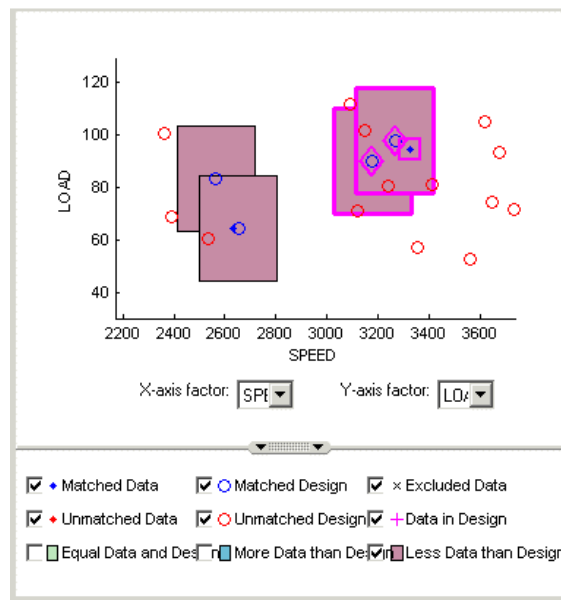


This removes the green clusters from view, as shown. These clusters are matched; you are more likely to be interested in unmatched points and clusters with uneven

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 **Cluster View** and select **Viewer Options** > **Select Unmatched Data** to display the remaining unmatched data points in the **Cluster Information** list view. 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 cluster plots 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”.



# Feature Calibration

---

This section includes the following topics:

## Feature Calibration

### In this section...

“What Are Feature Calibrations?” on page 11-2

“Start CAGE” on page 11-4

“Set Up Variables” on page 11-4

“Set Up Models” on page 11-7

“Set Up a New Feature” on page 11-9

“Set Up the Strategy” on page 11-10

“Set Up the Tables” on page 11-12

“Process For Feature Calibration” on page 11-14

“Calibrate the Normalizers” on page 11-15

“Calibrate the Tables” on page 11-19

“Calibrate the Feature” on page 11-25

“Export Calibrations” on page 11-30

## What Are Feature Calibrations?

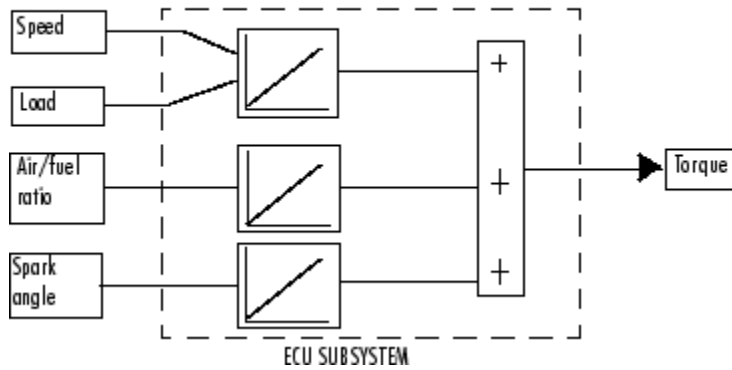
The feature calibration process within the Model-Based Calibration Toolbox product calibrates an estimator, or feature, for a control subsystem in an electronic control unit (ECU). These features are usually algebraic collections of one or more tables. You use the features to estimate signals in the 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 =  $TQ(\text{speed, load, AFR, spark})$



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**.

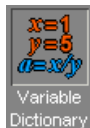
---

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

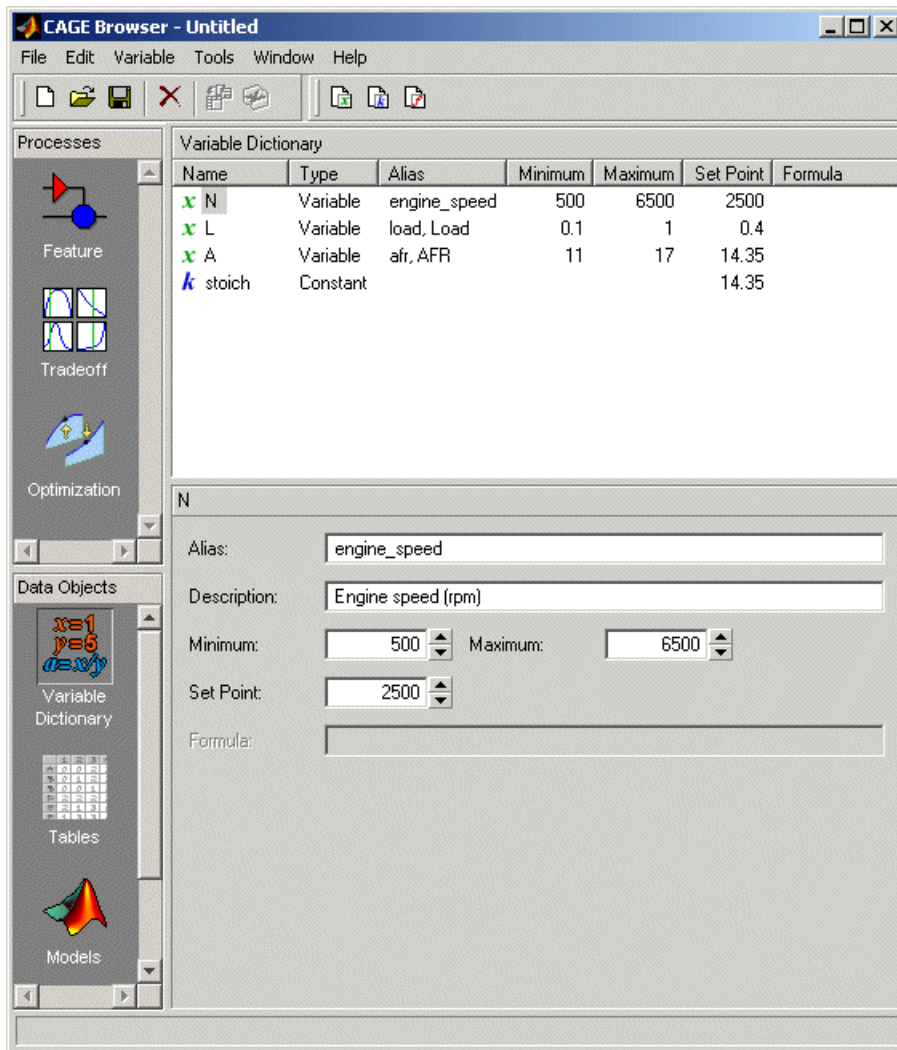
To import a variable dictionary,

- 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, `stoch`
- `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,

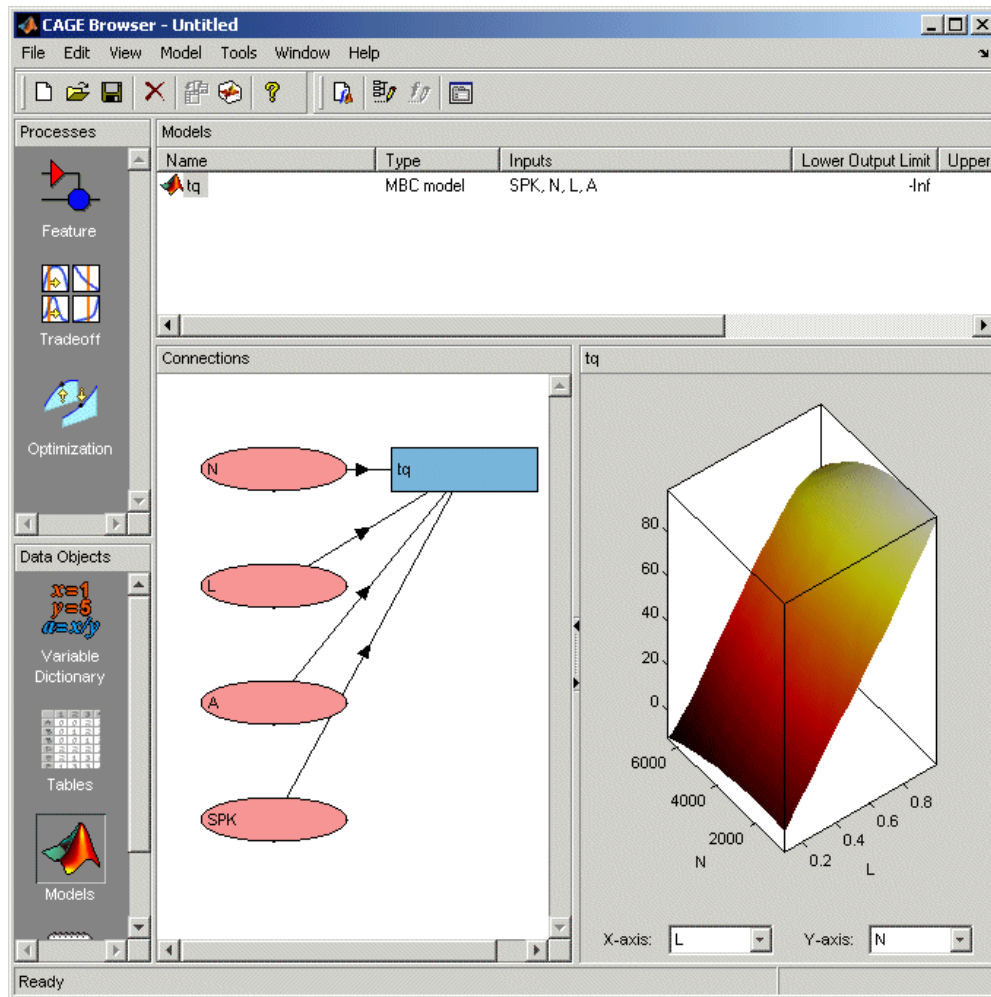
- 1 Select **File > Import > Model**, which opens a file browser.

- 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 11-10.

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 11-12.

## Set Up the Strategy

The toolbox uses Simulink to graphically specify the strategy.

### Importing a Strategy

To import a strategy,

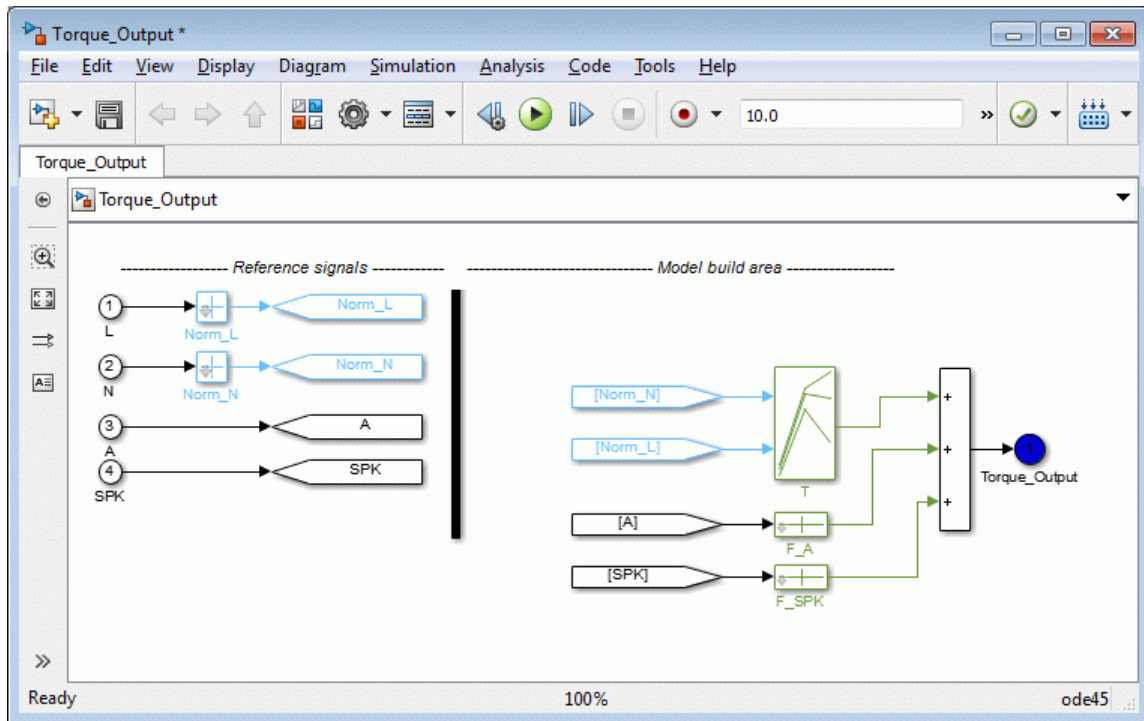
- 1 Select **File > Import > Strategy**.
- 2 Select the model file called `tutorial`, found in `matlab\toolbox\mbc\mbctraining`, and click **Open**.

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:

$$\text{Torque\_Output} = T(\text{Norm\_N}(N), \text{Norm\_L}(L)) + F\_A(A) + F\_SPK(\text{SPK})$$

- 5 Select **View > Full Strategy Display** to turn off the full description and see this simplified expression:

$$\text{Torque\_Output} = T + F\_A + F\_SPK$$

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.

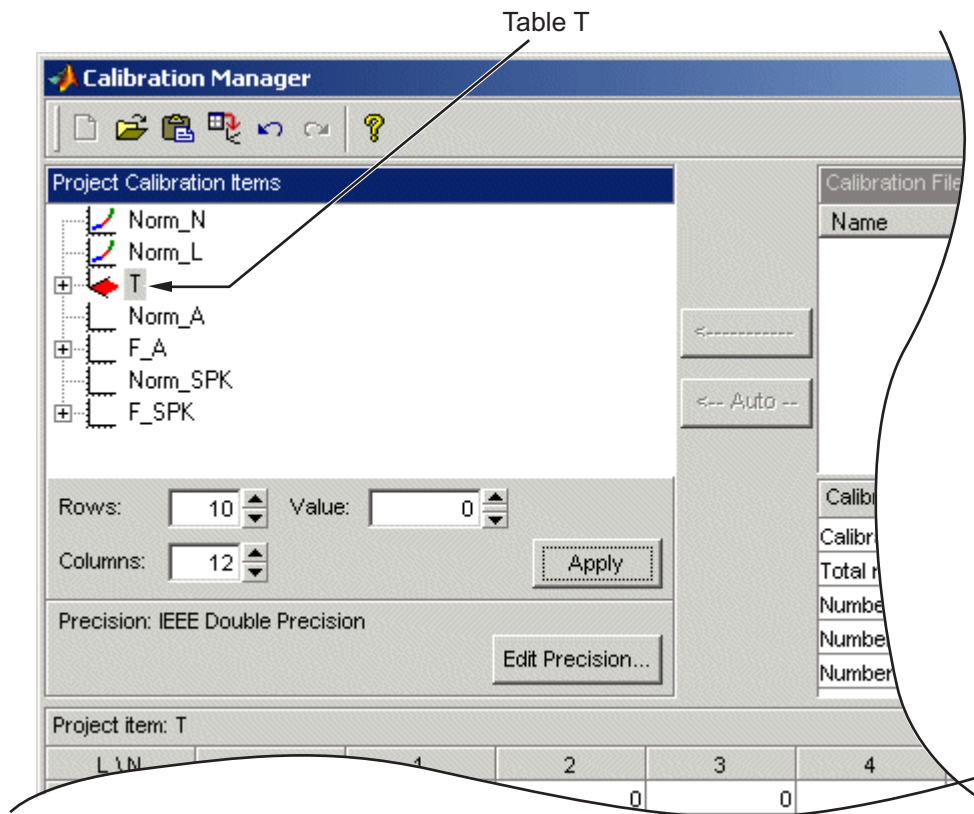
- 6 Click the plus next to `Torque_Output` to expand the Feature tree and observe the tables created by importing the strategy: `T`, `F_A`, and `F_SPK`. Expand each table node in turn to view the normalizers of each. You will define the sizes of the tables next.

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.



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.

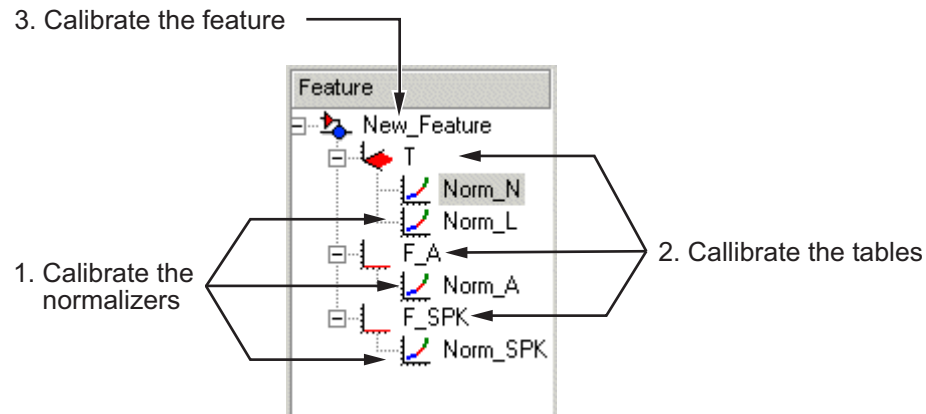
## Process For Feature Calibration

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.
- 2 Calibrate the tables.
- 3 Calibrate the feature as a whole.

Click the expand icon, , to expand the nodes and display all the tables and normalizers in the feature.



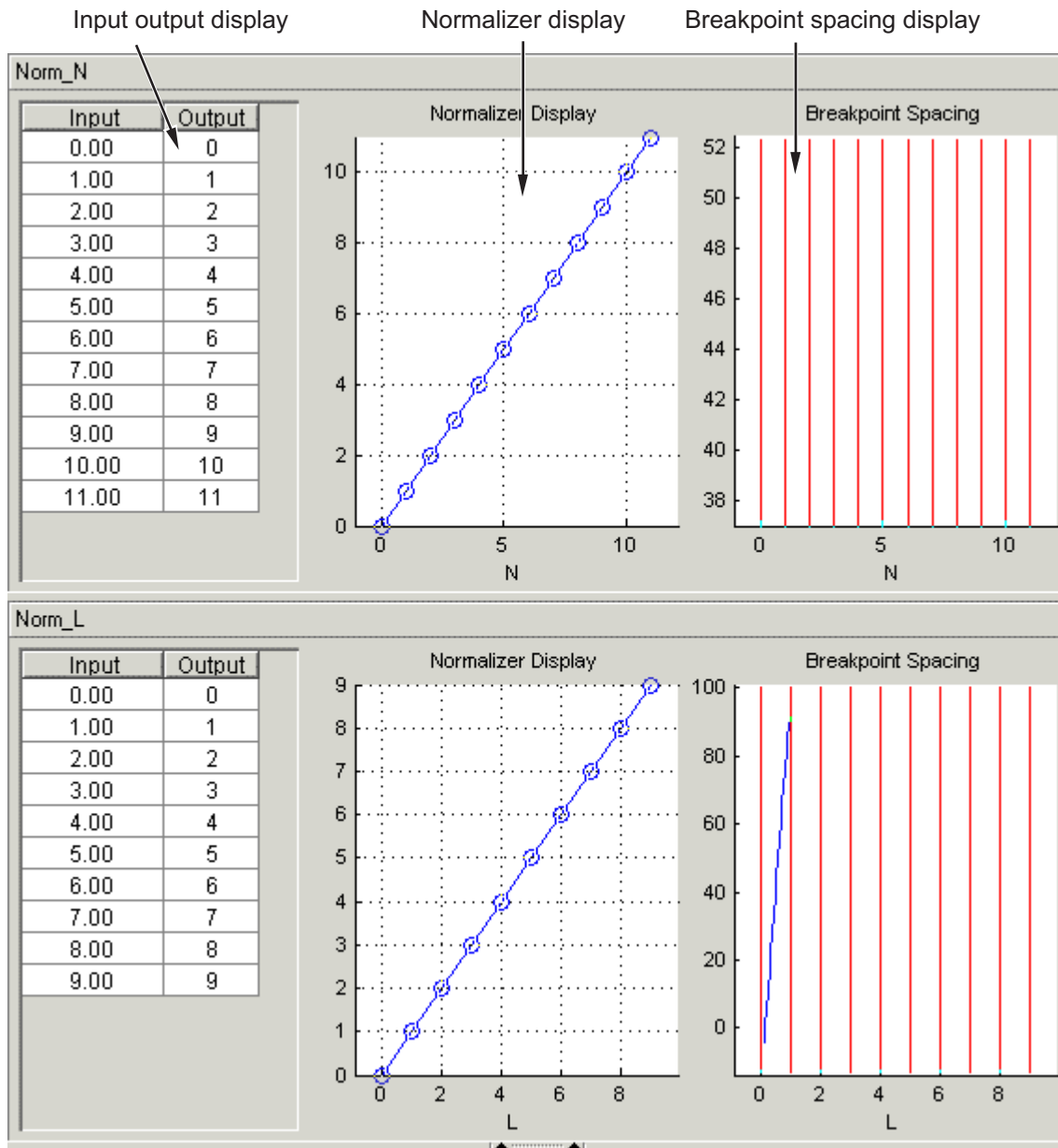


Each node in the display has a different view and different operations.

## **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.



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

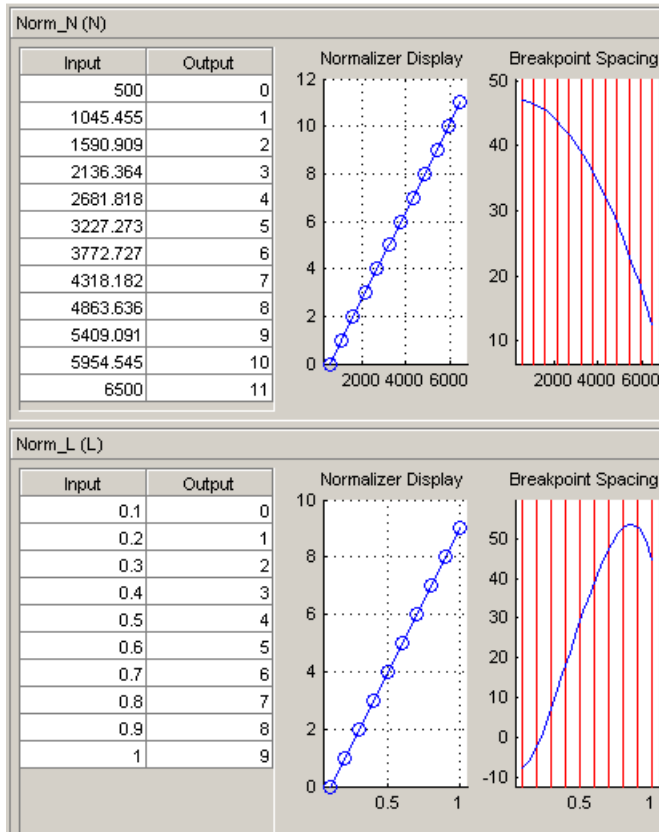
You now must space the breakpoints across the range of each variable. For example, Norm\_N takes values from 500 to 6500, the range of the engine speed.

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**.



A better fit between model and table can often be achieved by spacing the breakpoints nonlinearly.

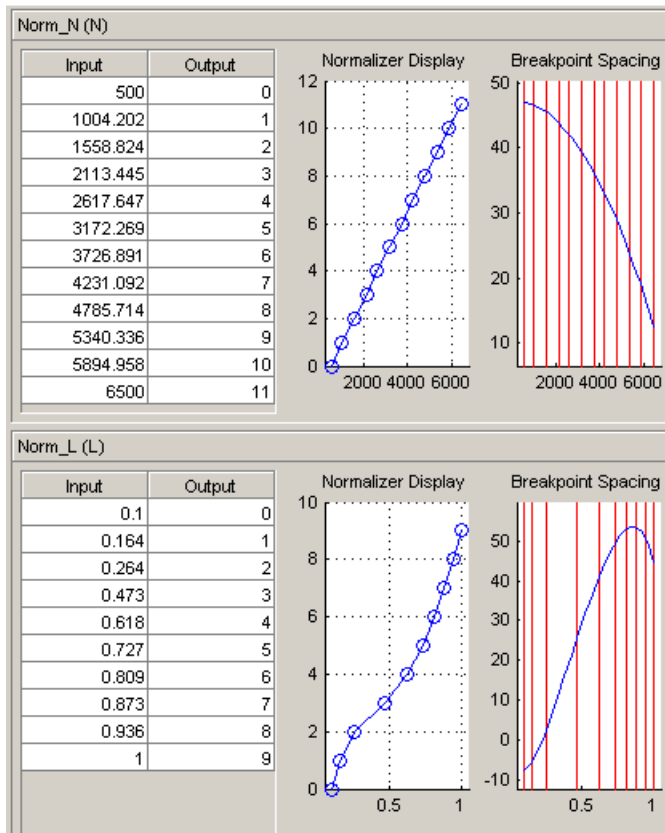
- 1 Click Fill  in the toolbar. Alternatively, select **Normalizer > Fill**.

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.

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.

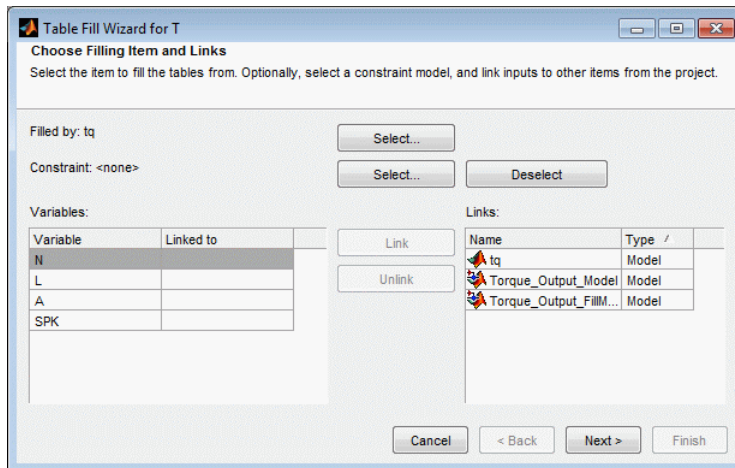
This view has two panes: the table and the graph of the table.

To fill the table with values of the model at the appropriate operating points,

- 1 Click Fill  on the toolbar.

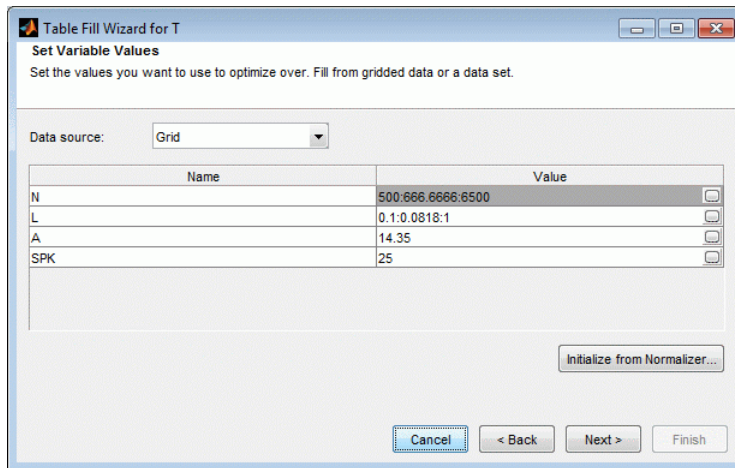
This opens the Feature Fill Wizard.

- 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.

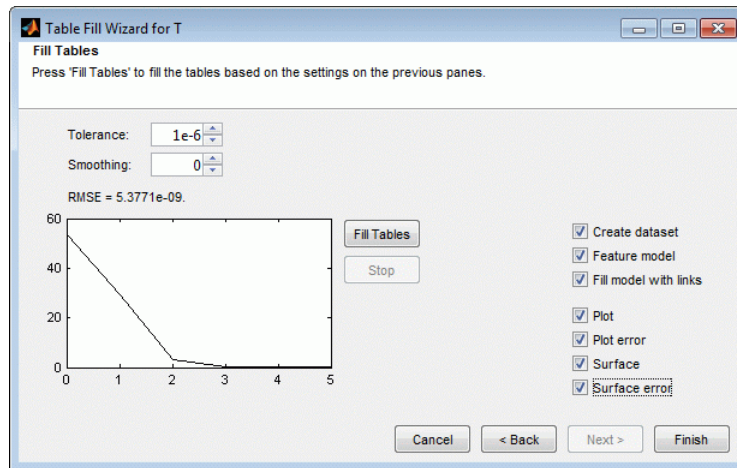


By default the table's normalizer breakpoints (here N and L) and the set points of the other variables (A and SPK) are selected. You can select different normalizers, and edit values to optimize over a range rather than at a single point. You can edit values directly in the **Value** edit box or click 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.



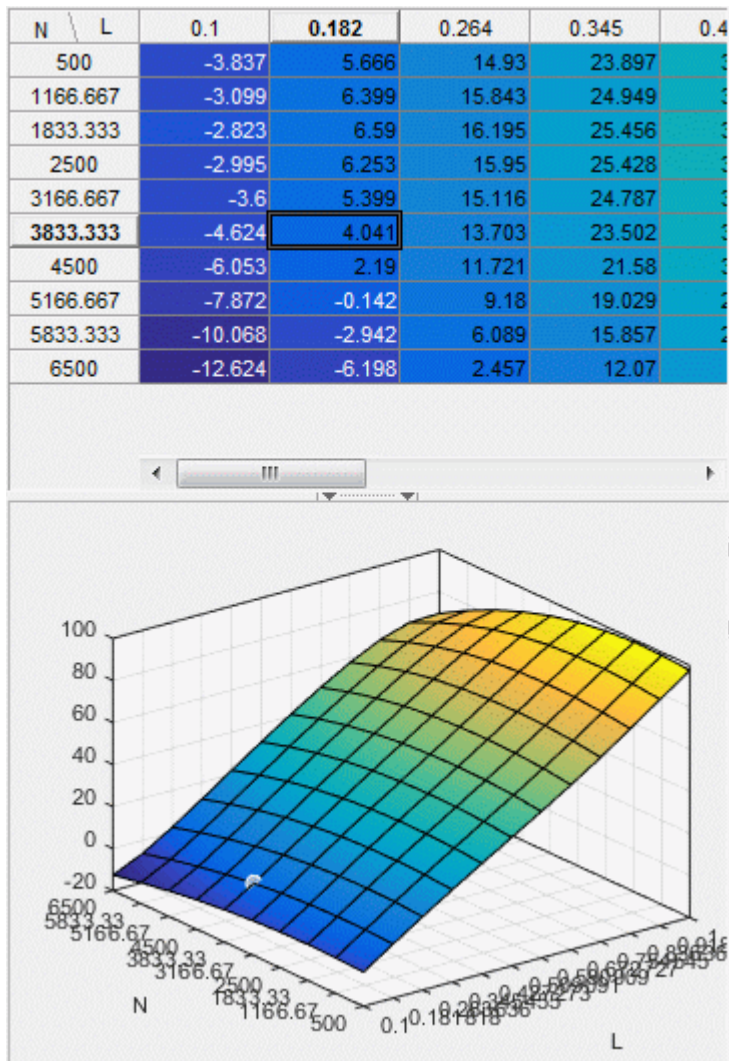


**6** Click **Finish**.

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.



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

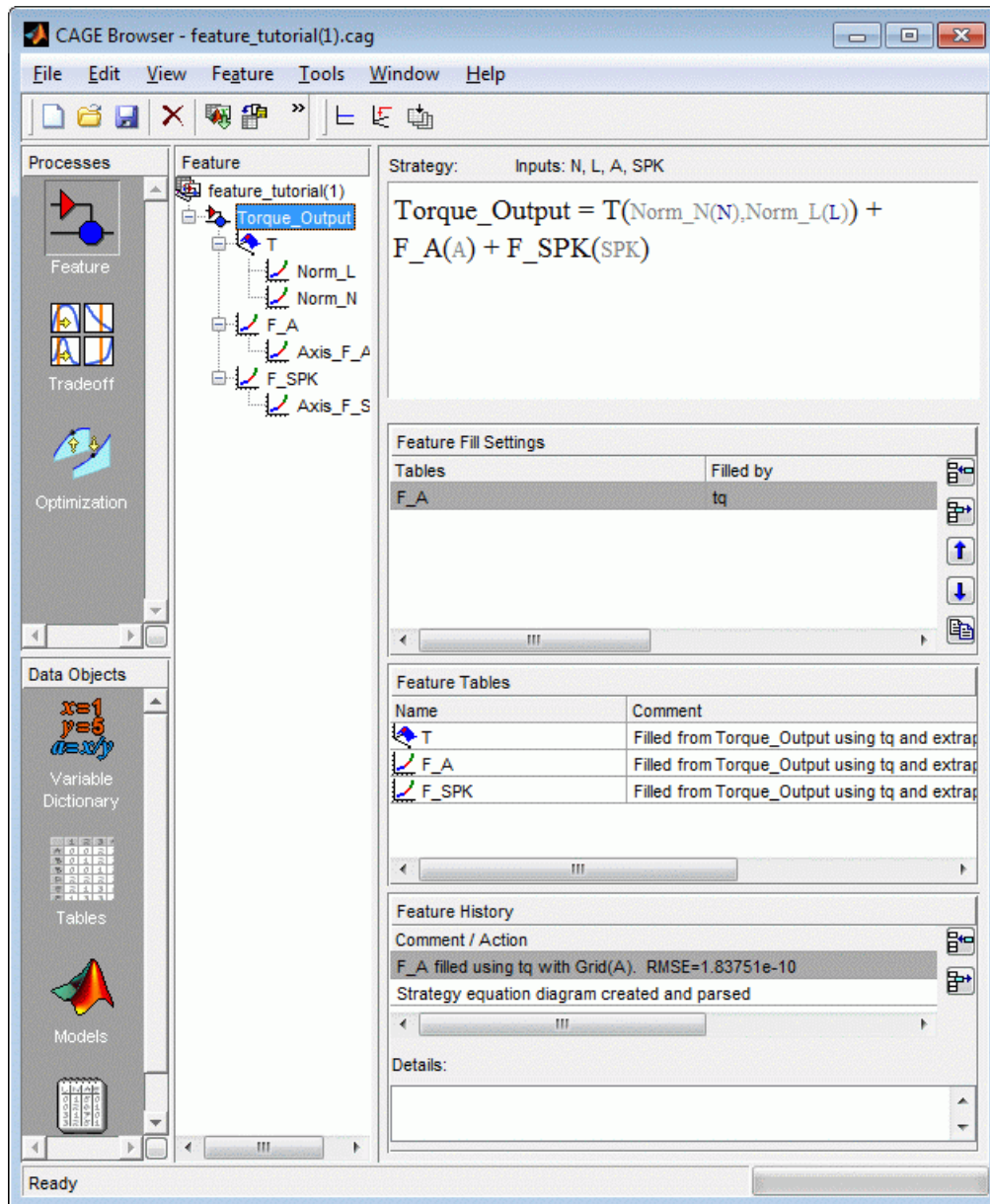
Now you must fill the tables F\_A and F\_SPK and their normalizers. The tables are modifiers for AFR and the spark angle respectively. These steps are described in the next section.

## **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.

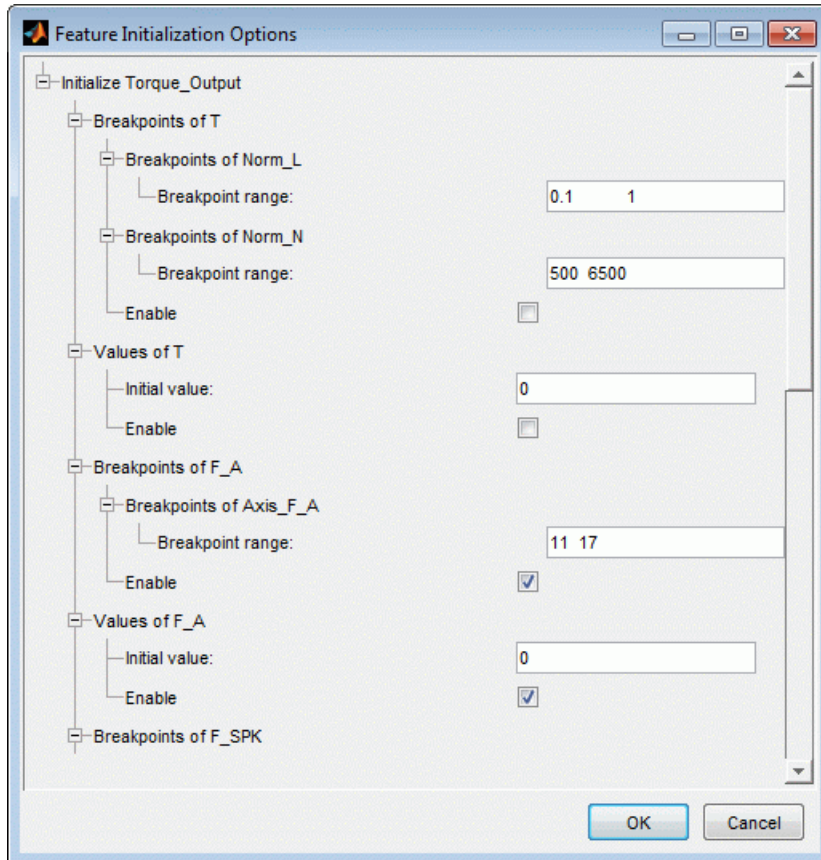
You could calibrate the normalizers and then the tables for F\_A and F\_SPK in turn. However, CAGE enables you to calibrate the entire feature in one procedure.

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.



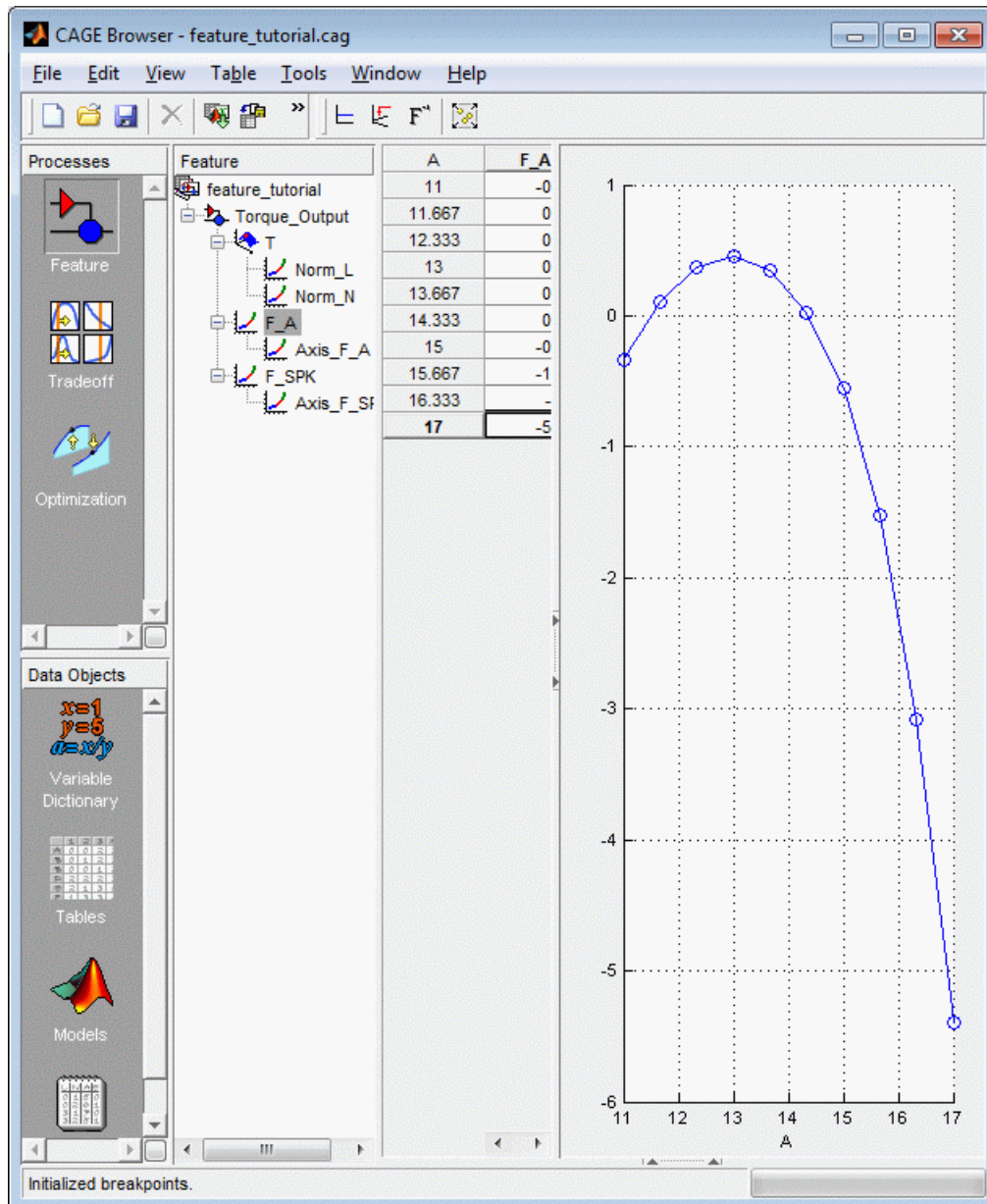
You have already optimized the breakpoints and table values for table T, so you only want to initialize the other tables F\_A and F\_SPK.

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 12-2     |
| “Setting Up a Tradeoff Calibration” on page 12-2   |
| “Performing the Tradeoff Calibration” on page 12-7 |

### 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 12-2
- “Adding Tables to a Tradeoff Calibration” on page 12-5
- “Displaying the Models” on page 12-6

#### 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 12-5.
- “Displaying the Models” on page 12-6 describes how you display the models of torque and NOX emissions.

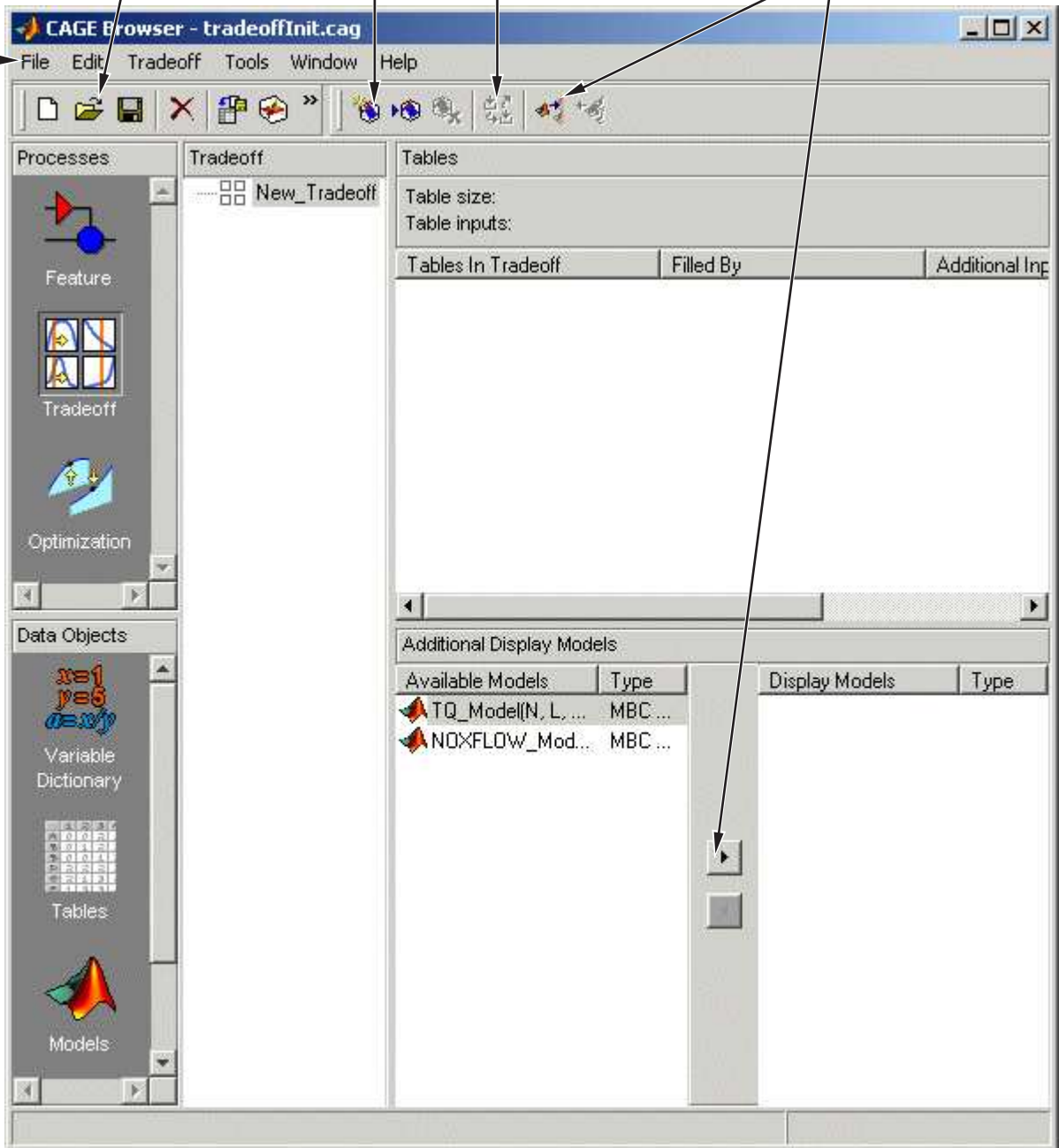
1. Open the project file

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.

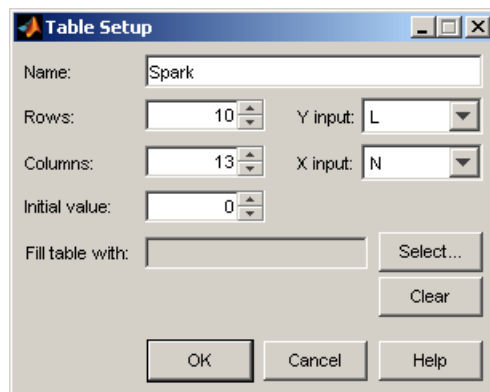
Both models have five inputs. The inputs for the torque and NOX models are

- Exhaust gas recycling (EGR)
- Air/fuel ratio (AFR)
- Spark angle
- Speed
- Load

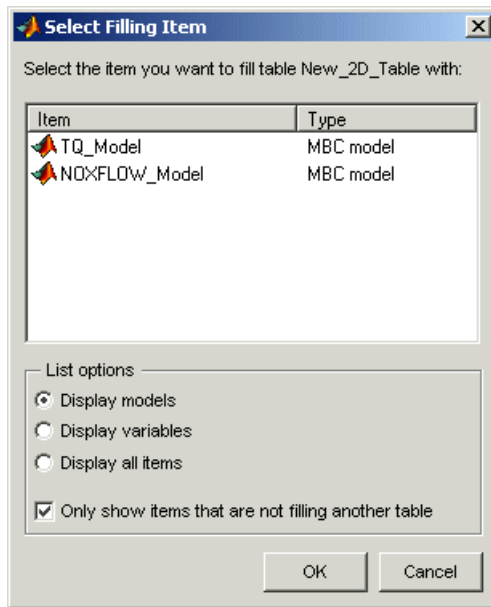
For this tutorial, you are interested in the spark angle over the range of speed and load.

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.



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.



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.

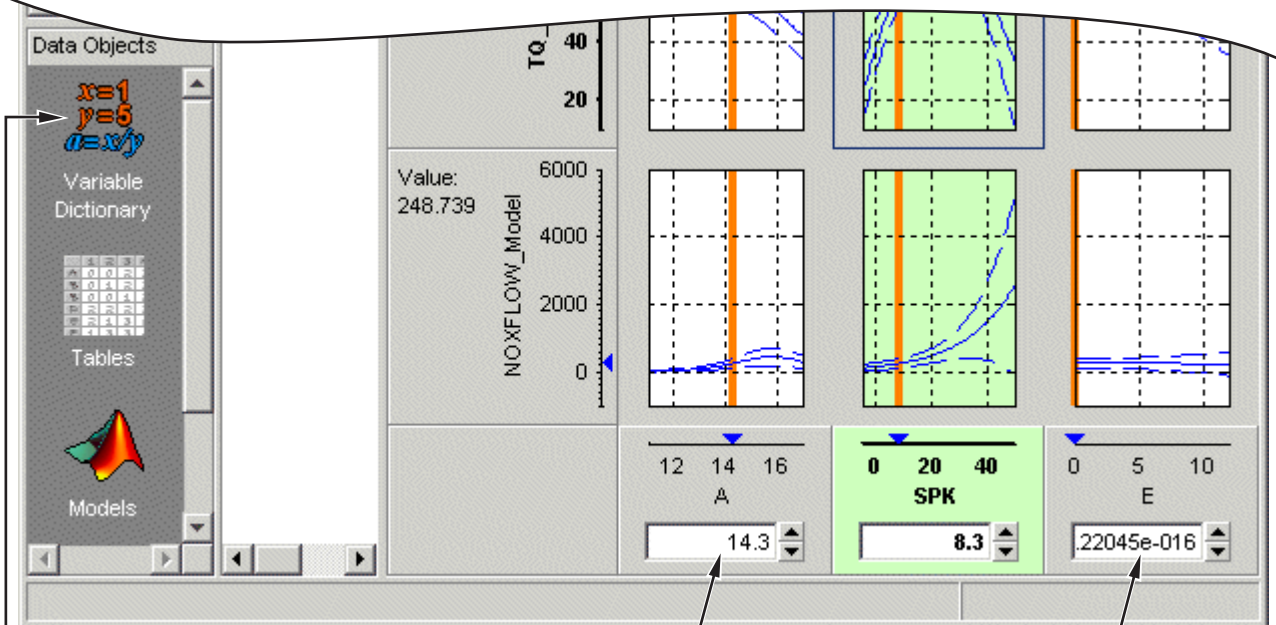
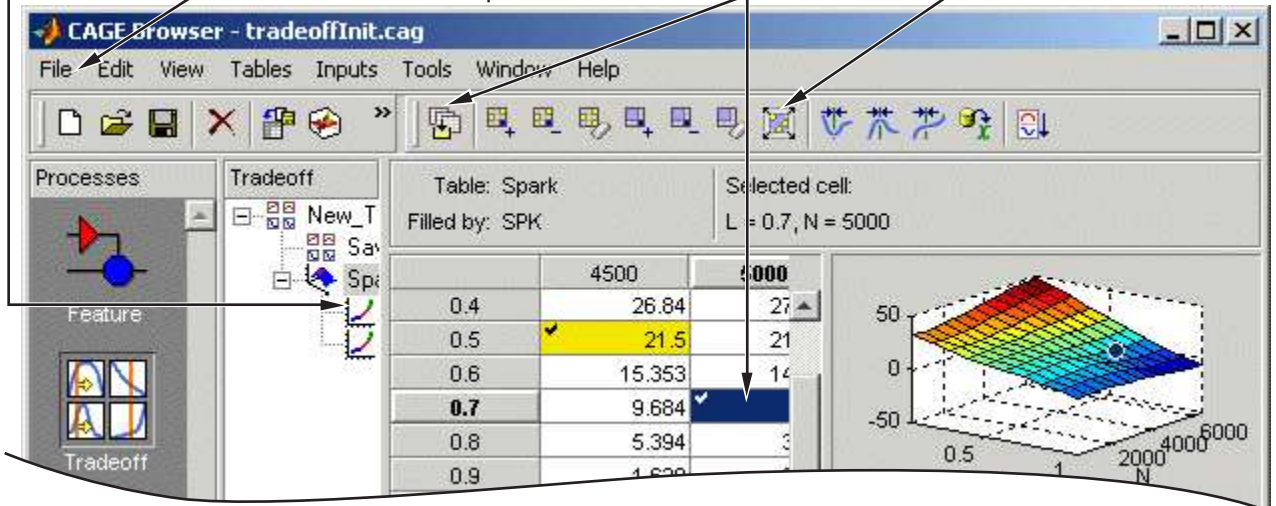
To perform the tradeoff calibration, follow the instructions in the next four sections:

- 1 Check the normalizers.
- 2 Set values for the other variables, AFR and EGR.
- 3 Fill key operating points with values for spark angle.
- 4 Fill the table by extrapolation.

Once you have completed the calibration, you can export the calibration for use in the electronic control unit.



1. Check the normalizers.
2. Set values for the other variables, either individually for each operating point or in the Variable Directory.
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.
5. Export the calibration.

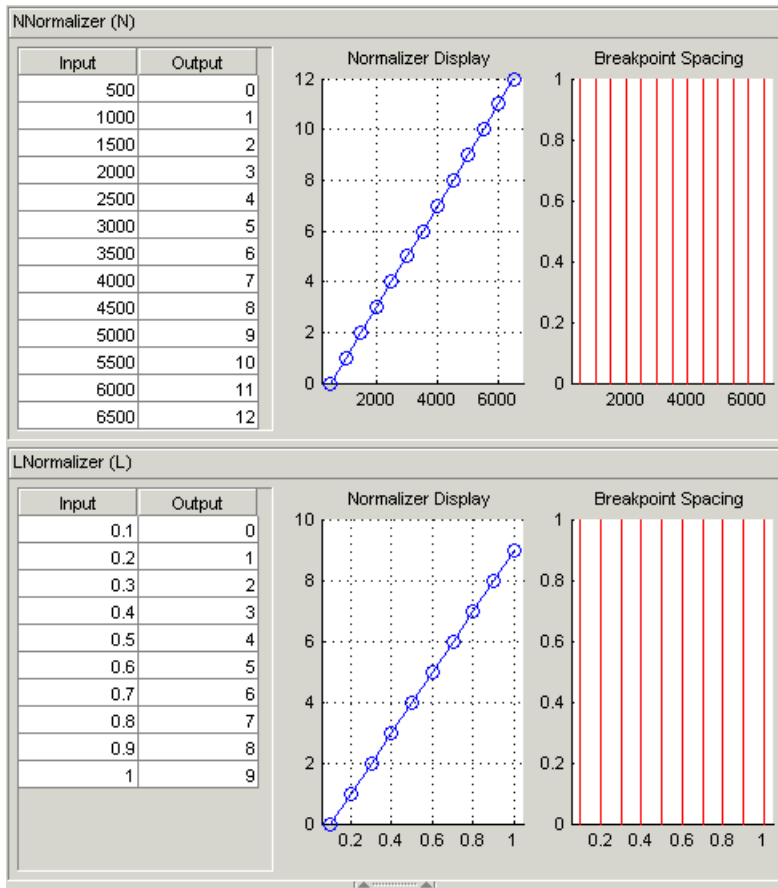


2. Set values for the other variables, either individually for each operating point or in the Variable Directory.

### 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.

Tradeoff

New Tradeoff

Spark

NNormalizer

LNNormalizer

Table: Spark

Filled by: SPK

Selected cell:  
L = 0.1, N = 500

|     | 500 | 1000 | 1500 |
|-----|-----|------|------|
| 0.1 | 0   | 0    | 0    |
| 0.2 | 0   | 0    | 0    |
| 0.3 | 0   | 0    | 0    |
| 0.4 | 0   | 0    | 0    |
| 0.5 | 0   | 0    | 0    |

Inputs have been saved  
 Locked table cell  
 Extrapolation mask  
 Region mask  
 Extrapolation and region mask

Value: 5.48987

TQ\_Model

Value: 0.72677

NOXFLOW\_Mode

12 14 16  
A

0 20 40  
SPK

0 5 10  
E

14.3

0

0

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.

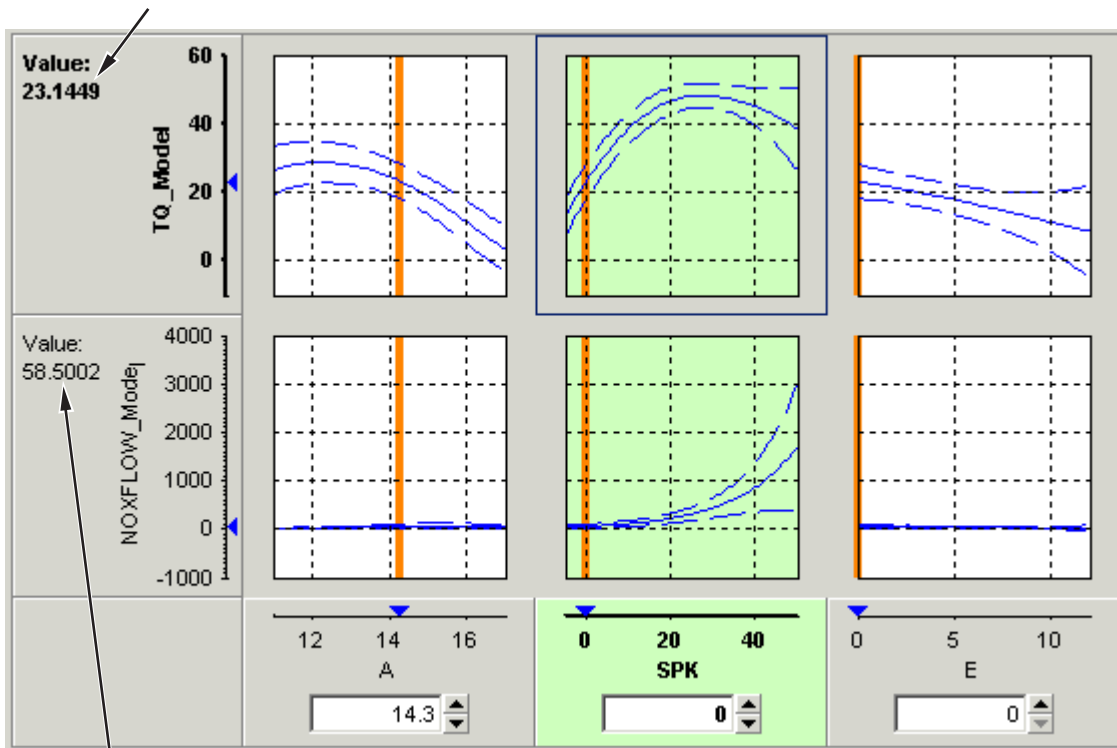
The upper pane displays the lookup table, and the lower pane displays the behavior of the torque and NOX emissions models with each variable.

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

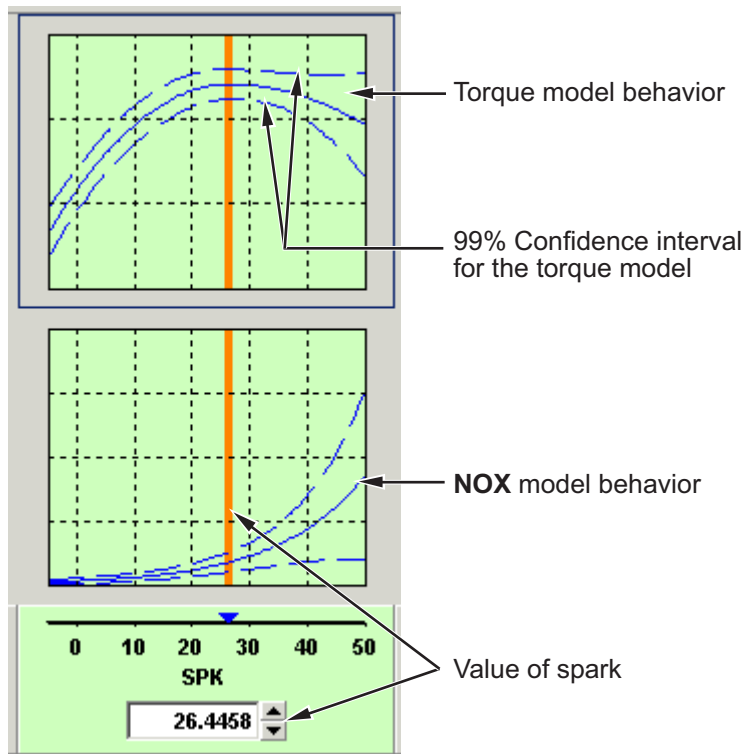


Value of the **NOX** model

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.




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  (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  (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.

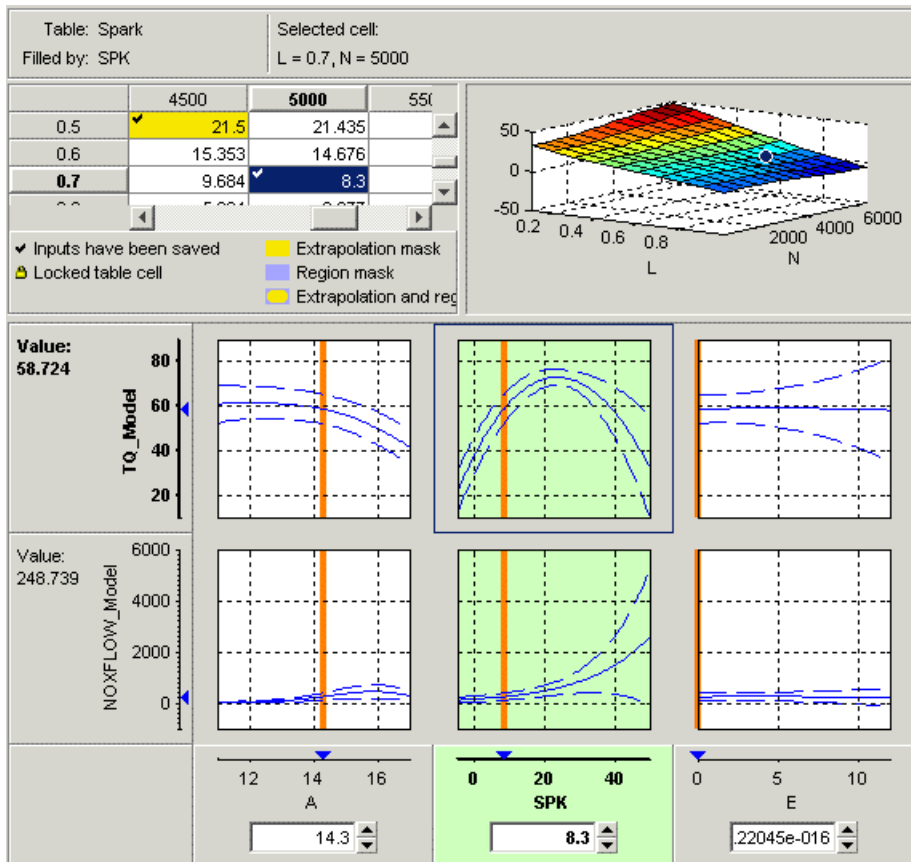
When you apply the value of the spark angle to the lookup table, the selected cell is automatically added to the extrapolation mask. This is why the cell is colored yellow. The extrapolation mask is the set of cells that are used as the basis for filling the table by extrapolation.

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

### In this section...

- “Setting Up the Data Set” on page 13-2
- “Comparing the Items in a Data Set” on page 13-7
- “Reassigning Variables” on page 13-14

### Setting Up the Data Set

- “Tutorial Overview” on page 13-2
- “Opening an Existing Calibration” on page 13-3
- “Importing Experimental Data into a Data Set” on page 13-3
- “Adding an Item to a Data Set” on page 13-6

### 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.
- 2 Import the experimental data.

- 3 Add 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`).

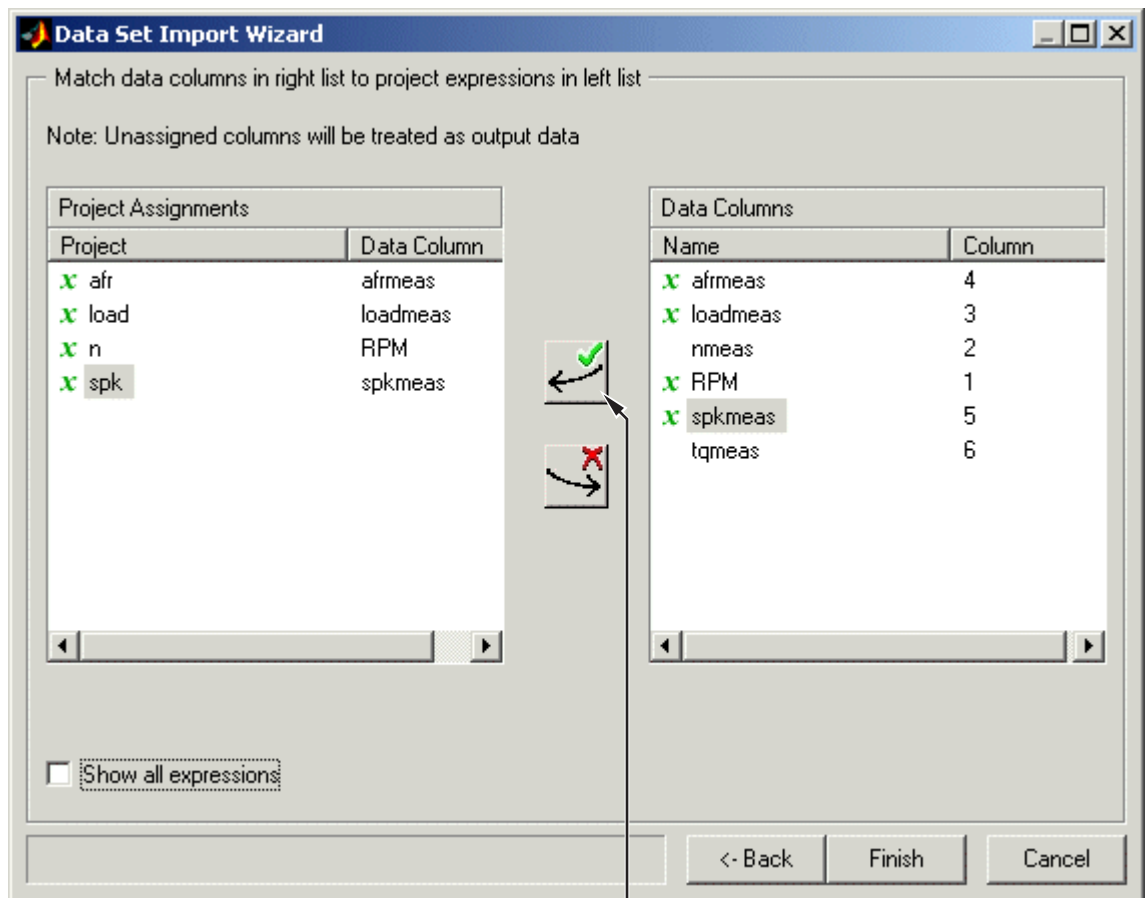
- 3 The Data Set Import Wizard asks which of the columns of data you would like to import. Click **Next** to import them all.

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  in the toolbar or select **Data > Assign**.

---

### **Adding an Item to a Data Set**

To add the Torque feature to the data set,

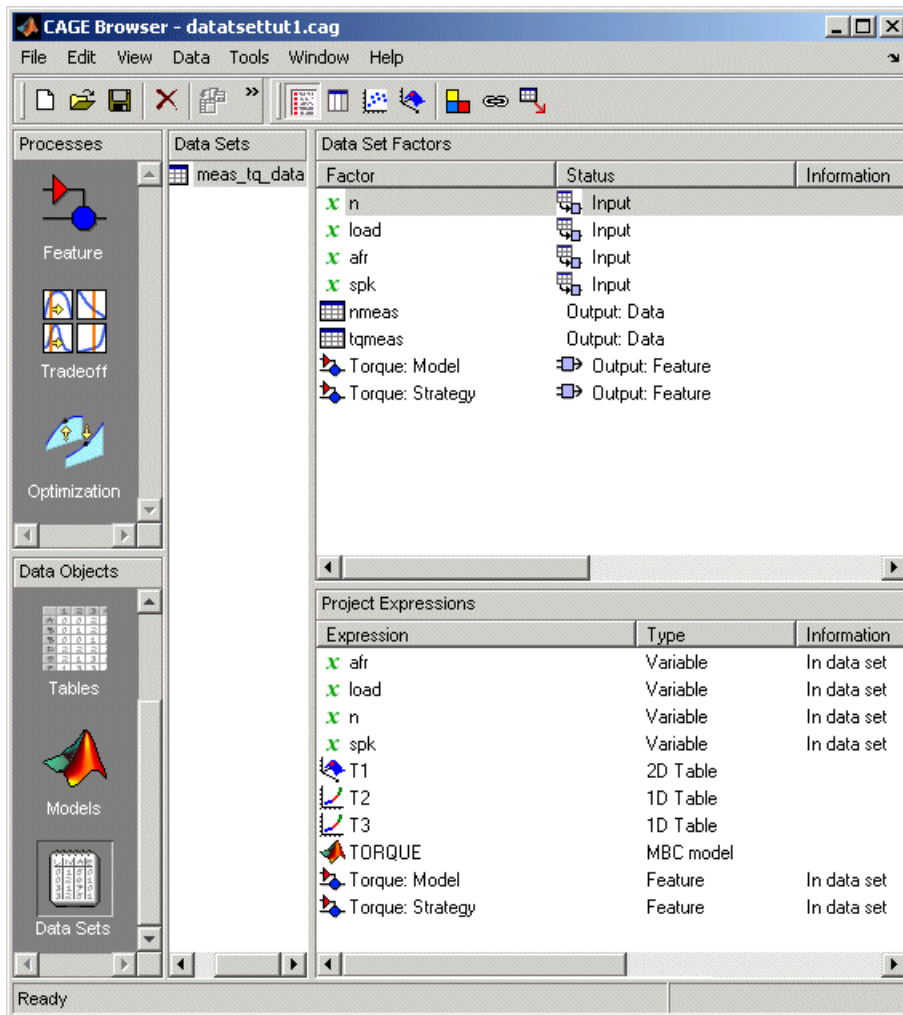
- 1** Highlight the Torque feature in the lower list of **Project Expressions**.
- 2** Select **Data > Factors > Add to Data Set**.

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

- “Viewing the Data Set as a Table” on page 13-8
- “Viewing the Data Set as a Plot” on page 13-9
- “Displaying the Error” on page 13-10
- “Coloring the Display” on page 13-12

## 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   |

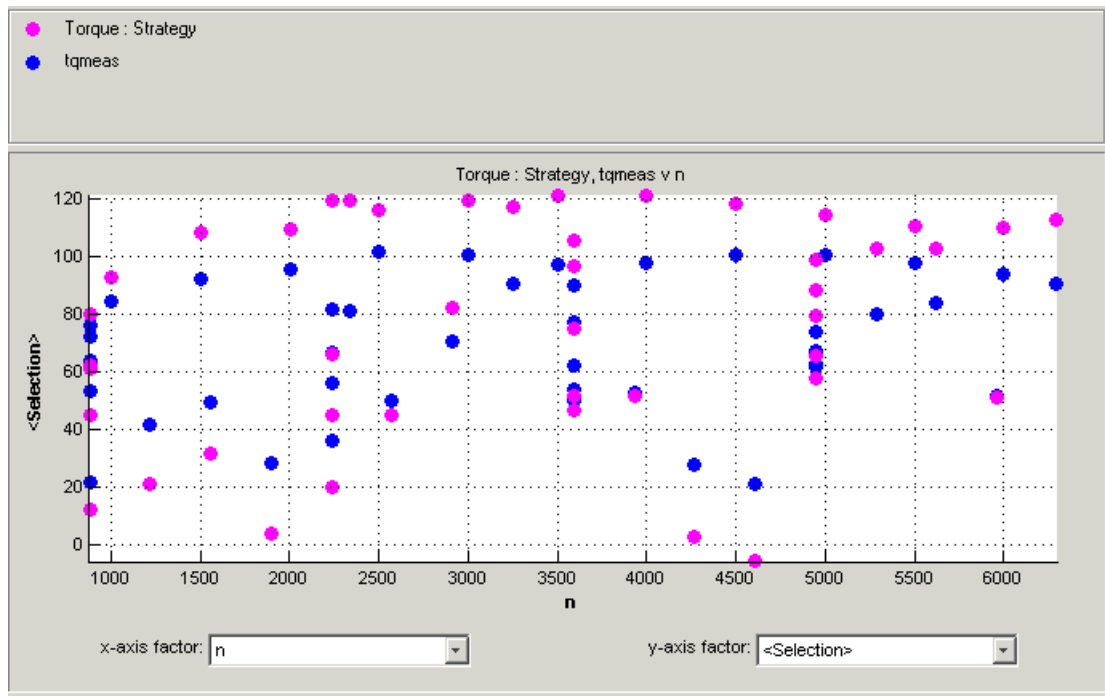
The error column is simply the difference between `tqmeas` and `Torque: Strategy`. This provides a simple way of comparing the feature and the measured data.

### 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.



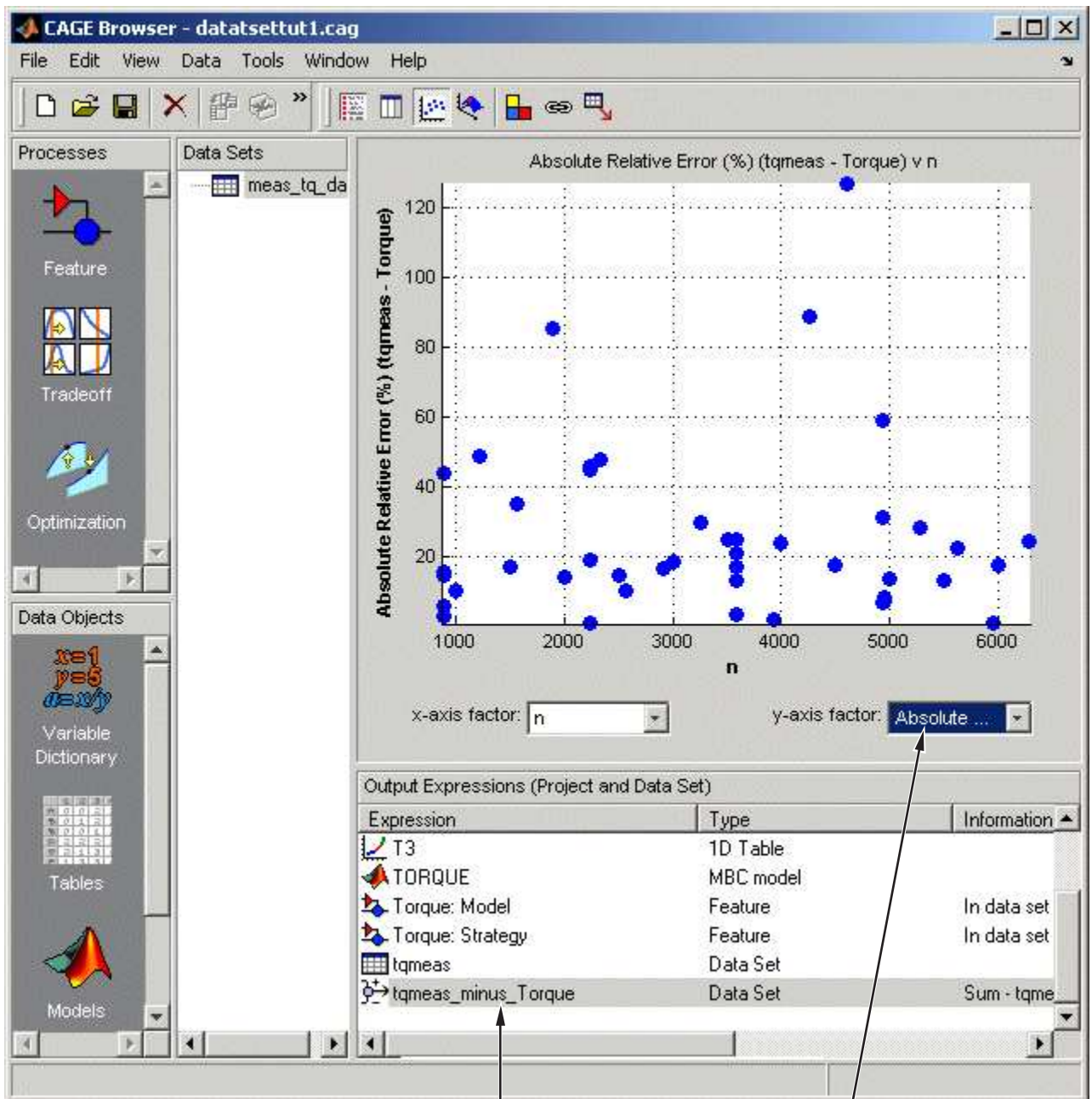
### 3 Change the **x-axis factor** to n from the drop-down menu.

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.



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.

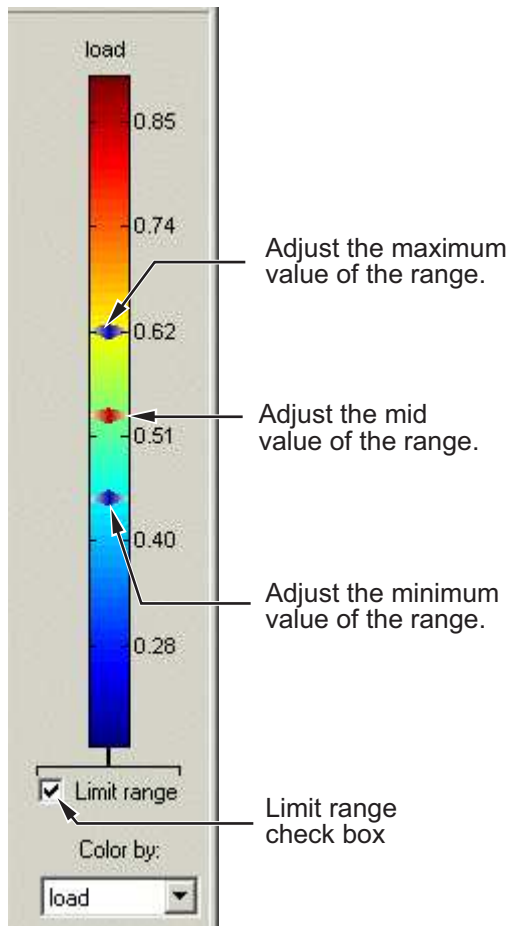
As you can see, there seems to be no particular correlation between engine speed and the error in the calibration.

### **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

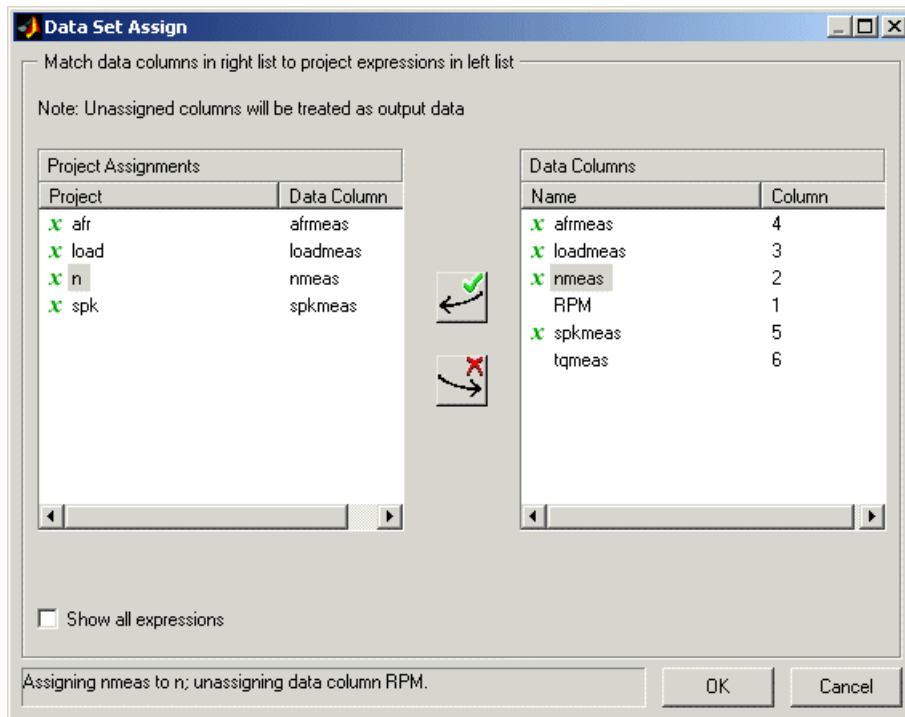
You can alter the data set by changing which variables are used for project expressions.

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

|   |
|---|
| <b>In this section...</b>                                   |
| “Setting Up a Table and Experimental Data” on page 14-2     |
| “Filling the Table from the Experimental Data” on page 14-8 |
| “Selecting Regions of the Data” on page 14-11               |
| “Exporting the Calibration” on page 14-13                   |

### Setting Up a Table and Experimental Data

- “Tutorial Overview” on page 14-2
- “Adding Variables” on page 14-3
- “Adding a New Table” on page 14-4
- “Importing Experimental Data” on page 14-6

#### 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 14-8 describes the process of filling the lookup table.
- “Selecting Regions of the Data” on page 14-11 describes how you can select some of the data for inclusion when you fill the table.
- “Exporting the Calibration” on page 14-13 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 for speed and load by importing a variable dictionary.
- 2 Add a new table to your session.
- 3 Import your experimental data.

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.

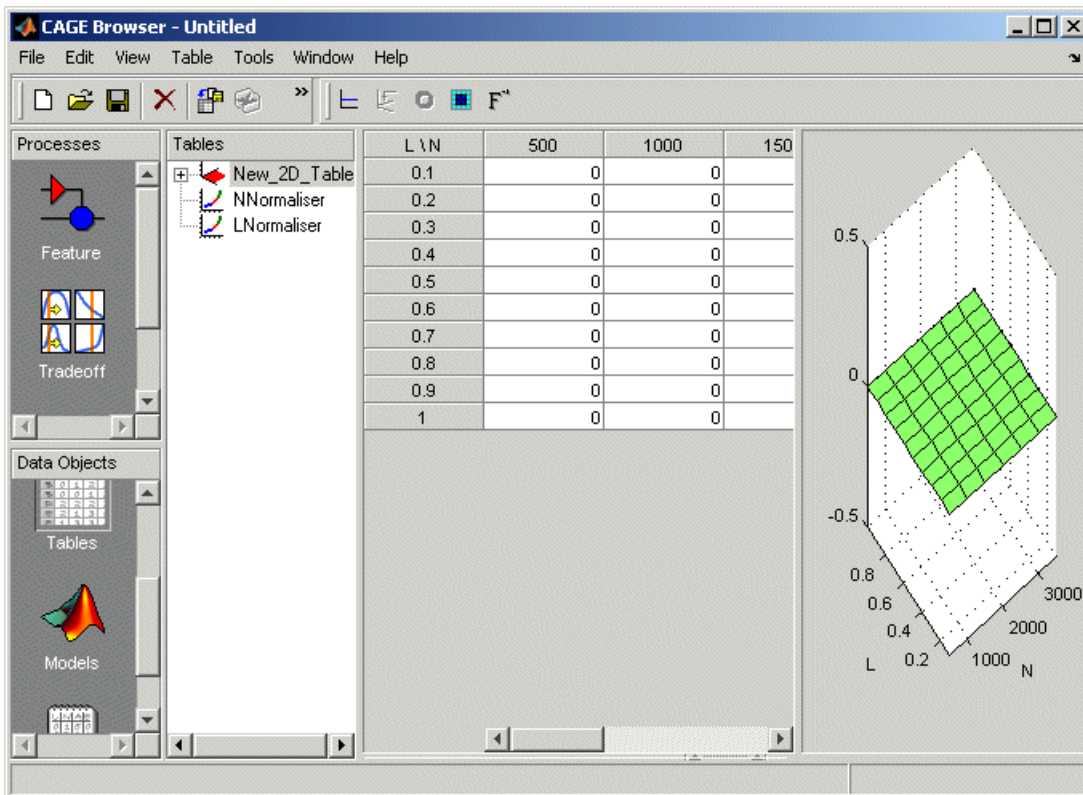
- 2 Change the number of columns to 7.
- 3 Select L as the variable for normalizer **Y** and N as the variable for normalizer **X**, then click **OK**.

---

**Note** In CAGE, a 2-D table is defined as a table with two inputs.


---

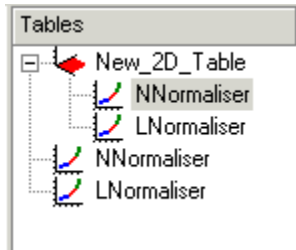
CAGE takes you to the **Tables** view, where you can see the following.



### 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 (`afmeas`), 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.

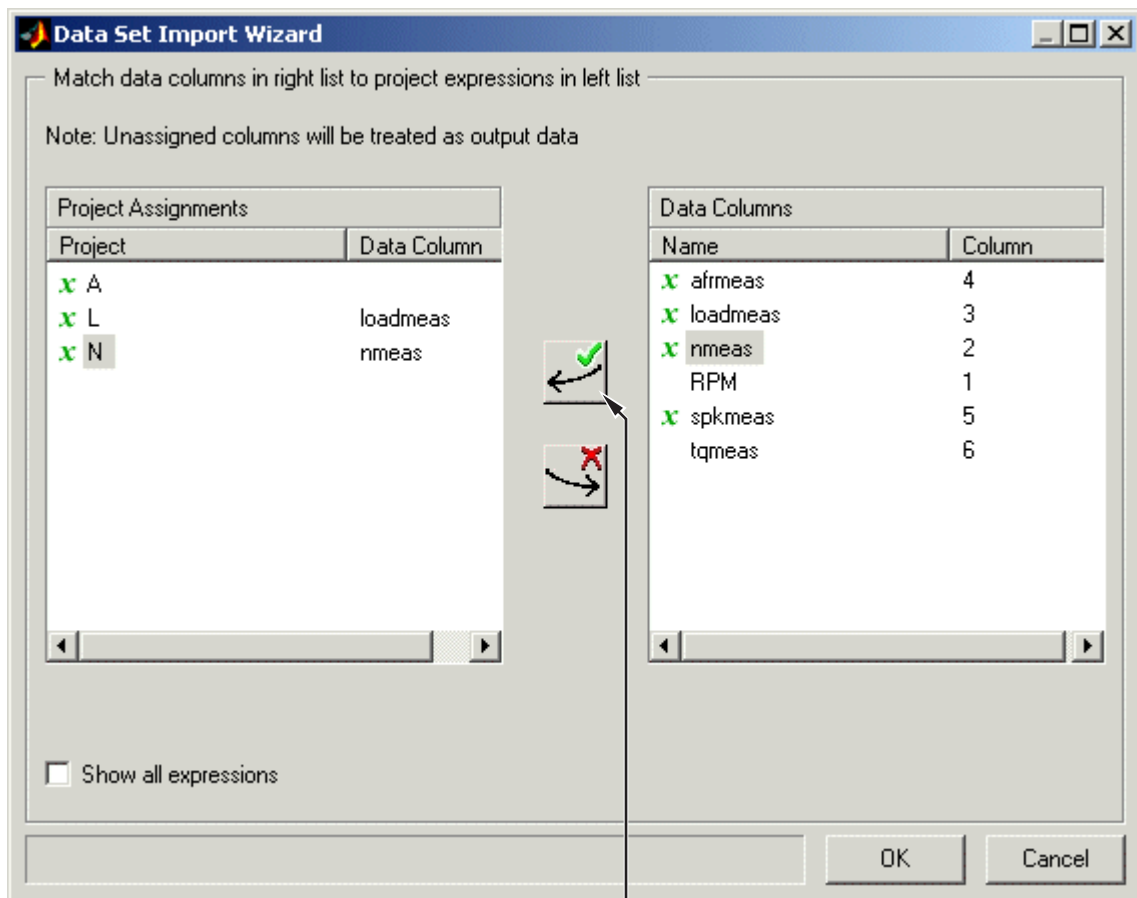
The following screen asks you to associate variables in your project with data columns in the data.



- Highlight N in the **Project Assignments** column and nmeas in the **Data Column**, then click the assign button, shown.



- Repeat this to associate L with loadmeas. The dialog box should be the same as the following.



Assign button

- 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  (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

L (table axis)

N (table axis)

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

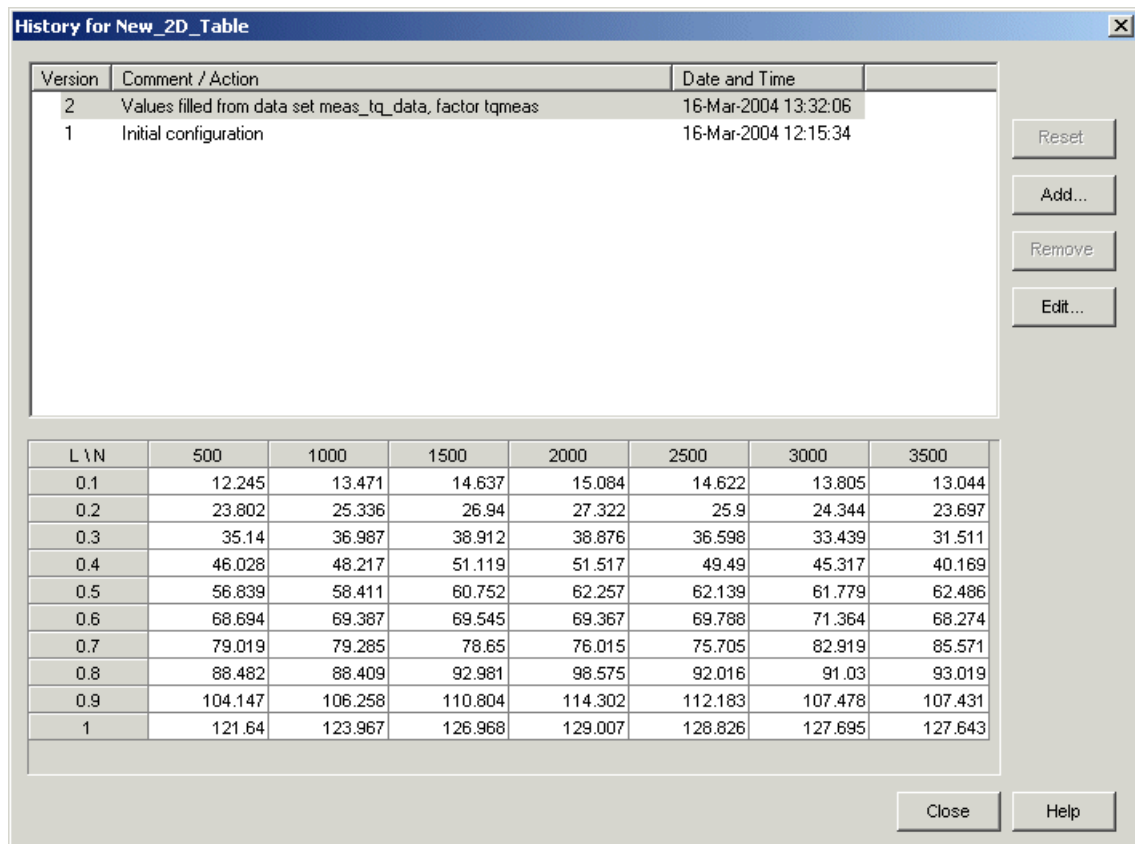
Show table history after fill

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.

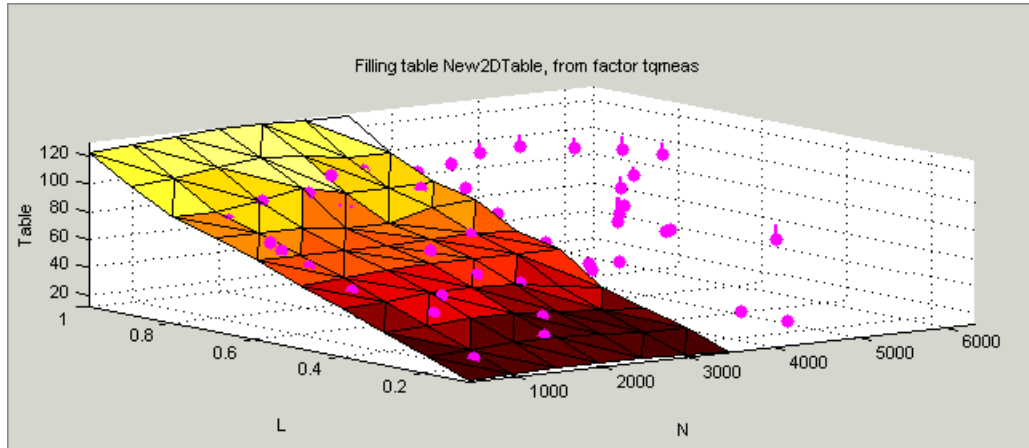
- 5 To view the table after it is filled, ensure that the **Show table history after fill** box, at the bottom left, is selected.
- 6 To fill the table with values of `tqmeas` extrapolated over the range of the normalizers, click **Fill Table**.

This opens the History dialog box, shown.



- 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**.



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

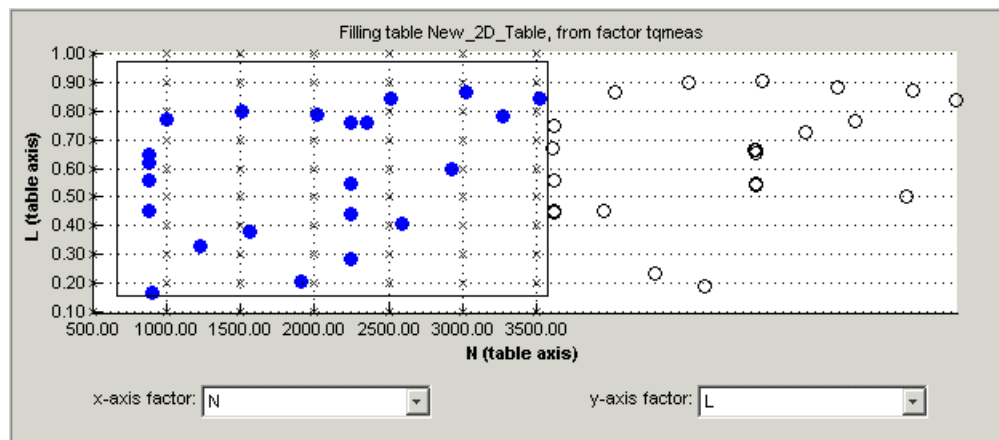
You can ignore points in the data set when you fill your lookup table.

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.

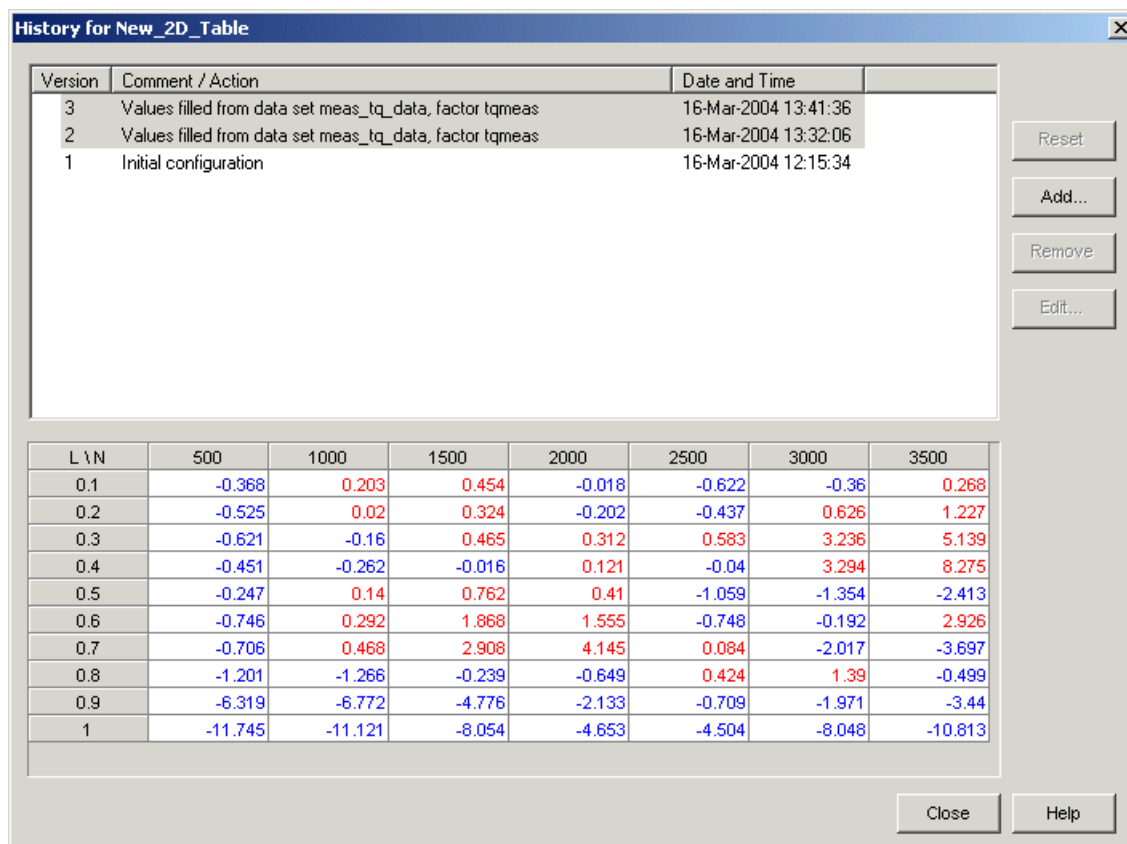
To ignore points in the data set,

- 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.



- 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.



- 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” on page 15-2
- “Single-Objective Optimization” on page 15-5
- “Multiobjective Optimization” on page 15-18
- “Sum Optimization” on page 15-30
- “Automated Tradeoff” on page 15-35

## Optimization and Automated Tradeoff

### In this section...

“Import Models to Optimize” on page 15-2

“Optimization Example Problems” on page 15-4

### 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 “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**.

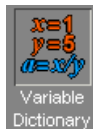
The `tradeoffInit.cag` project contains two models and all the variables necessary for this tutorial. For more information about how to set up models and variables, see “Calibration Setup” in the CAGE documentation.

- 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.

## Optimization Example Problems

In this tutorial you will use optimization to find solutions to the following problems:

- A single-objective optimization to find maximum values of torque, subject to a constraint to keep NOX emissions below a specified level. You will export the output and use it to fill a table.
- A multiobjective optimization to maximize torque and minimize NOX emissions.
- A sum optimization to maximize torque while minimizing NOX, weighted to give more importance to idle speed.
- Using any of your optimizations to run an automated tradeoff. Once you have set up an optimization you can apply it to a tradeoff.

To work through these examples, follow the steps in the these sections in order:

- 1 “Single-Objective Optimization” on page 15-5
- 2 “Multiobjective Optimization” on page 15-18
- 3 “Sum Optimization” on page 15-30
- 4 “Automated Tradeoff” on page 15-35

---

**Note:** Start with “Single-Objective Optimization” on page 15-5.

---

# Single-Objective Optimization

**In this section...**

“Process Overview” on page 15-5

“Using the Create Optimization from Model Wizard” on page 15-6

“Setting Constraints and Operating Points” on page 15-9

“Running the Optimization” on page 15-12

“Using Optimization Results to Fill Tables” on page 15-14

“Using a Custom Fill Routine to Fill Tables” on page 15-16

## Process Overview

The following sections describe these stages:

- 1** Using the Create Optimization from Model wizard to choose
  - A model for your objective
  - The optimization settings:
    - Which algorithm to use
    - Maximize or minimize
    - Point or sum objective
    - Operating points for your optimization
    - What free variables to use
- 2** Using the Optimization view to choose
  - A model, type, and value for your constraint
- 3** Running the optimization, examining the output, exporting to a data set, and using the output to fill a table

---

**Note:** To follow these steps, you must first load models. See “Import Models to Optimize” on page 15-2.

---

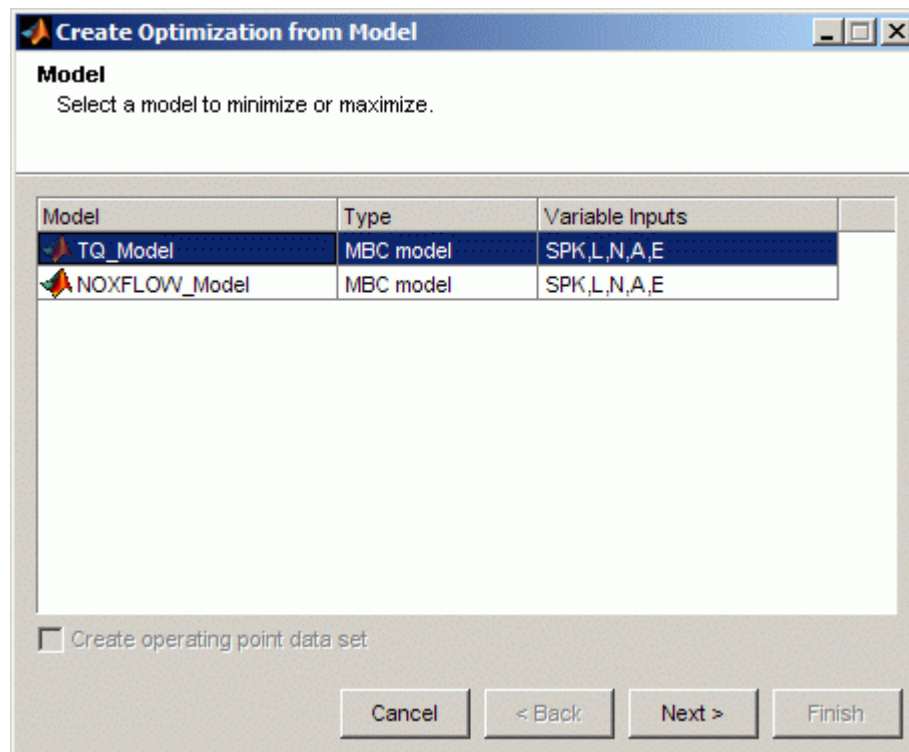
## Using the Create Optimization from Model Wizard

To create your optimization,

- 1 Select **Tools > Create Optimization From Model** (or use the toolbar button).

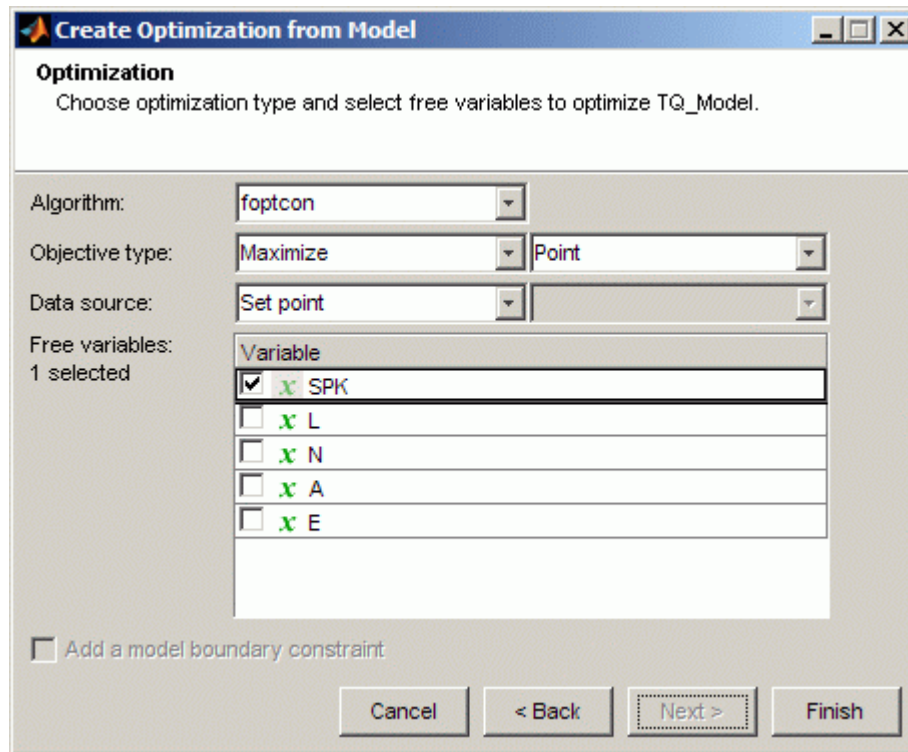
The **Create Optimization From Model** Wizard appears. If you are viewing a model, then the wizard automatically selects the current model.

- 2 Select TQ\_Model as the model you want to optimize.



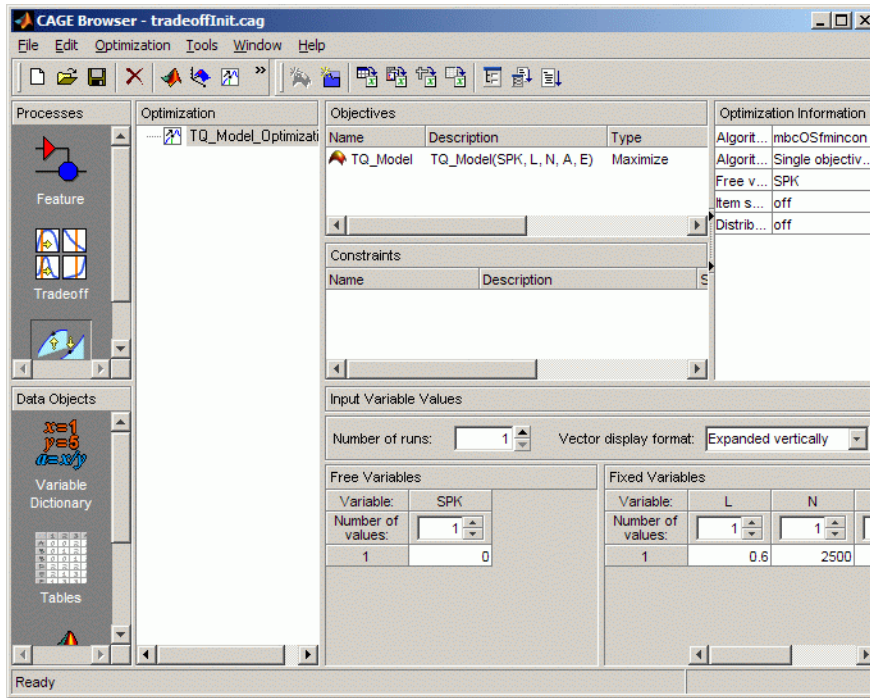
Click **Next**.

- 3 On this page of the wizard you select the optimization settings.
  - a Select **Maximize** for the **Objective type**.
  - b Clear the check boxes for all the **Free variables** except SPK.



Leave the other settings at the defaults, and click **Finish** to create the optimization.


You see the CAGE Browser Optimization view. A new branch named **TQ\_Optimization** appears in the Optimization tree. View the new optimization. Your CAGE browser should look like the following example.



In the **Objectives** pane you can see the **Description** TQ\_Model (SPK, L, N, A, E) and the **Type** is Maximize.

In the **Optimization Information** pane you can see listed the default algorithm name mbcOSfmincon, free variable SPK, and if you hover the mouse you can read the full description Single objective optimization subject to constraints.

The **Constraints** pane is empty as you have not yet added any constraints.

Note that the toolbar button Run Optimization (  ) is enabled, because your optimization setup has provided enough information to start an optimization.

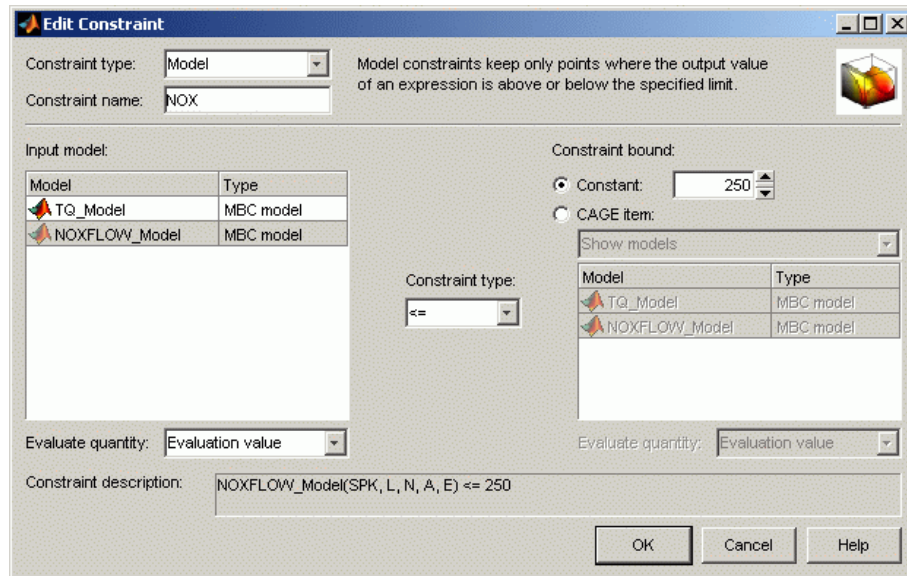
Your optimization is ready to run, but would run at a single point (the set points of the variables is the default). To finish setting up your optimization you need to specify a constraint and edit the points where you want the optimization to run.



## Setting Constraints and Operating Points

- 1 Right-click the **Constraints** pane and select **Add Constraint**.

The Edit Constraint dialog appears.



- a Leave the **Constraint type** drop-down menu at the default, **Model**.
- b Edit the **Constraint name** to **NOX**.
- c Select **NOXFLOW\_Model** from the **Input model** list.
- d Make sure the inequality is **<=**, and enter **250** in the **Constant** edit box as the maximum value for the constraint, as shown above.
- e Press **Enter**.

You return to the CAGE Browser Optimization view. Make sure the **Description** **NOXFLOW\_Model(SPK, L, N, A, E) <= 250** appears in the **Constraints** pane.

- 2 You can use the **Optimization Point Set** panes to define a set of operating points for the optimization. Note that you do not have to have an operating point set; if you

do not, the optimization will run at a single point of your choosing (the set points of variables is the default).

You can use the Create Optimization from Model wizard to choose the points where you want to run the optimization, or you can set up points later in the Optimization view. In both cases you can choose to use points from a suitable data set or table grid if they exist in your project.

Running the optimization requires the selected models to be evaluated (many times over) and hence values are required for all the model input factors (L, N, A, E, and SPK). The defaults of the fixed variables (L, N, A, E) are their set points, as shown in the **Fixed Variables** pane. You have chosen SPK as a free variable, so the optimization will choose different values for SPK in trying to find the best. The default initial value for a free variable is the set point, as shown in the **Free Variables** pane.

To define the set of operating points for the optimization,

- a In the **Optimization Point Set** pane, increase the **Number of runs** to 6. Notice 6 rows appear in both fixed and free variables panes, all containing the default set point values of each variable.
- b Enter, or copy and paste, these values into the N column of the **Fixed Variables** pane. (Select all N rows before pasting):

| N    |
|------|
| 1000 |
| 1000 |
| 3000 |
| 3000 |
| 6000 |
| 6000 |

- c Enter, or copy and paste, these values into the L column of the **Fixed Variables** pane:

| L   |
|-----|
| 0.1 |
| 0.8 |

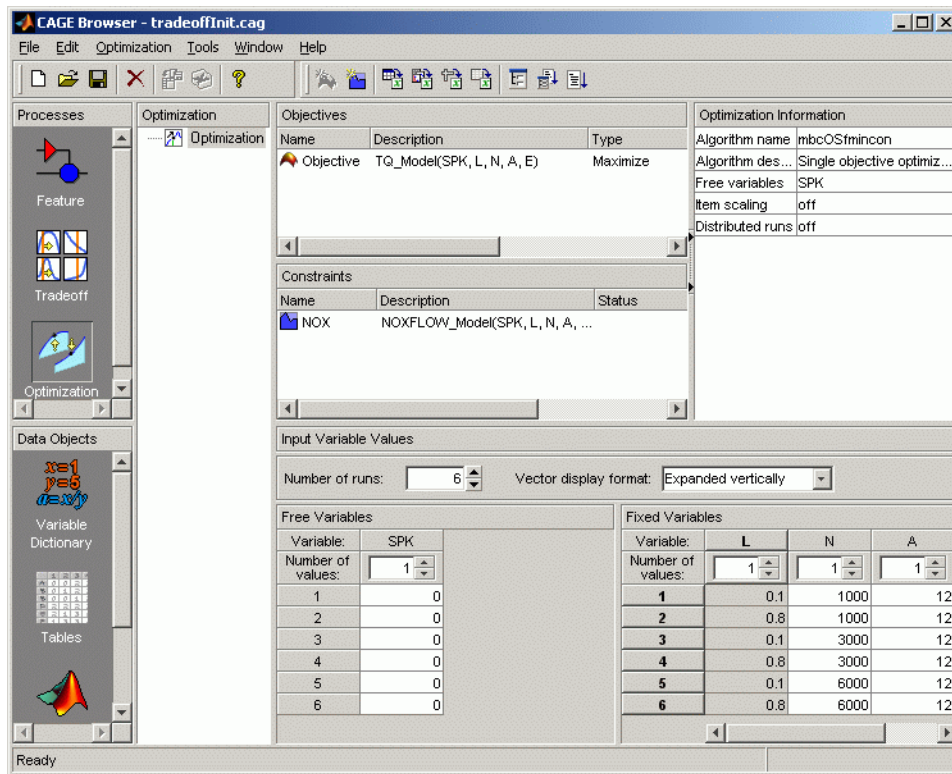
| L   |
|-----|
| 0.1 |
| 0.8 |
| 0.1 |
| 0.8 |

The **Fixed Variables** pane should look as shown.


| Variable:         | L   | N    | A  | E |
|-------------------|-----|------|----|---|
| Number of values: | 1   | 1    | 1  | 1 |
| 1                 | 0.1 | 1000 | 12 | 5 |
| 2                 | 0.8 | 1000 | 12 | 5 |
| 3                 | 0.1 | 3000 | 12 | 5 |
| 4                 | 0.8 | 3000 | 12 | 5 |
| 5                 | 0.1 | 6000 | 12 | 5 |
| 6                 | 0.8 | 6000 | 12 | 5 |

Leave the other fixed variables and the free variable values at the defaults. If you wished to restrict the range of the free variables, you could select **Optimization > Edit Free Variable Ranges**. The default is the range of the variable as defined in the Variable Dictionary. For this example, leave the default.

- 3 Your CAGE Browser should now look like the following example, with an objective, constraint, and set of operating points. The optimization is ready to run.



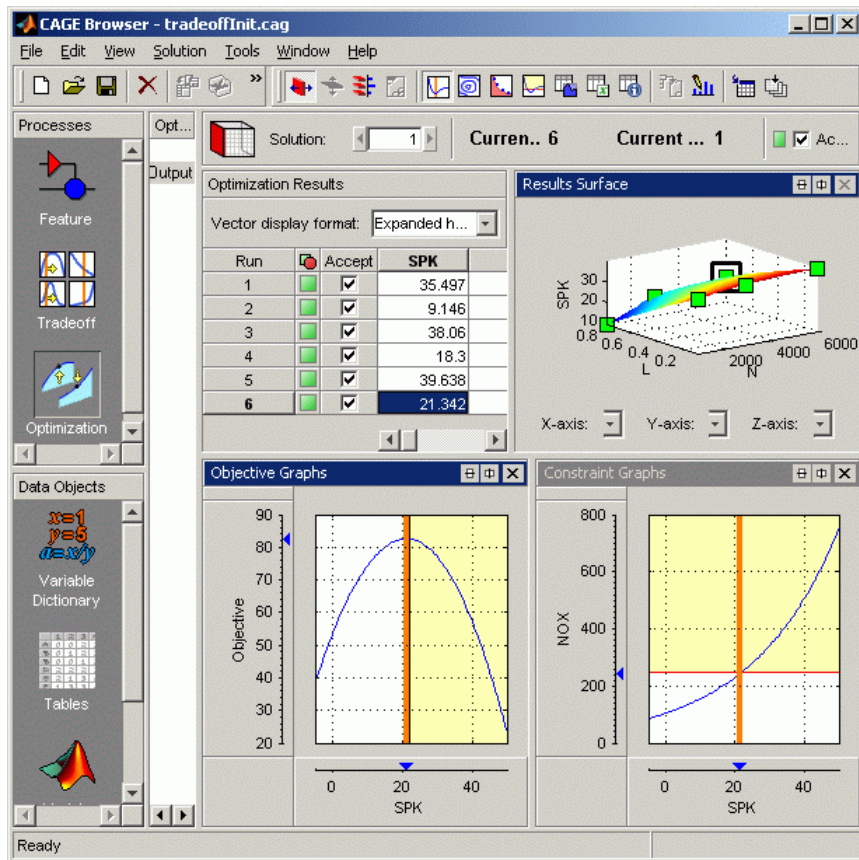
## Running the Optimization

- 1 Click Run Optimization (  ) in the toolbar.

The optimization runs, showing progress messages as each point is evaluated until the optimization is complete. On completion of the optimization, a new node appears in the Optimization tree.

- 2 The view switches to the new node TQ\_Model\_Optimization\_Output where you can view the optimization results.
- 3 The optimization output view retains a memory of previous layout. If you have not used these views before, try the buttons and right-click context menus in the view title bars to add **Constraint Graphs** to examine your results.

- 4 This single-objective optimization produces one best solution for each point in the operating point set. To view solutions at particular operating points, either:
  - Click the cells of the table, or
  - Click the points in the **Results Surface** or **Results Contour** plot.
- 5 Select **Run 6** and examine the results.



For more information, see “Analyzing Point Optimization Output” in the CAGE User's Guide documentation.

## Using Optimization Results to Fill Tables

As an example, to use these optimization results to fill a table, first create a new table as follows:

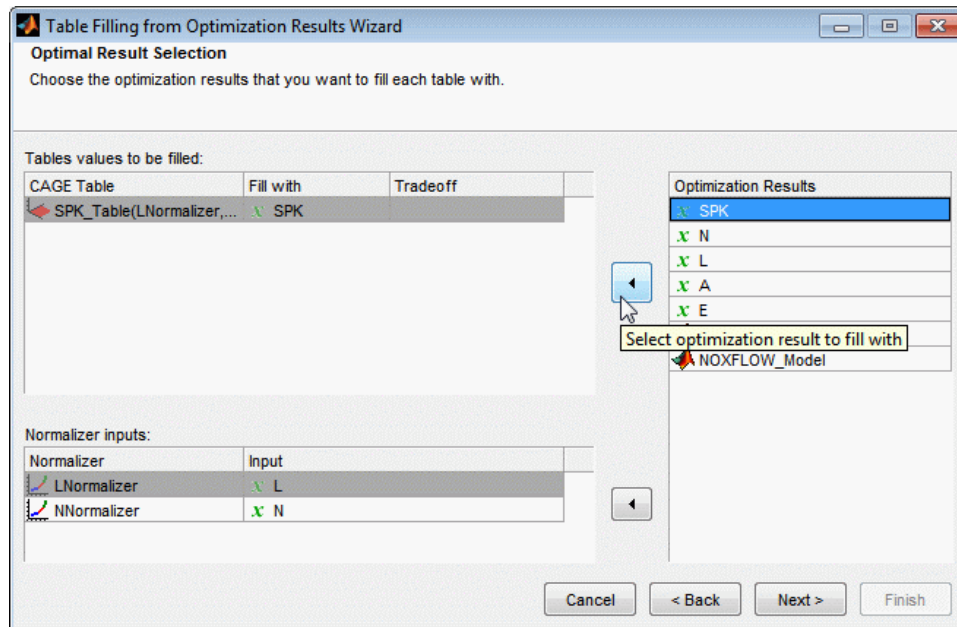
- 1 Build a SPK table in N and L. Select **File > New > 2-D Table**.
- 2 Leave 10 in the **Rows** and **Columns** edit boxes and 0 in the **Initial Value** edit box.
- 3 Use the drop-down menus to select L and N for the Y and X inputs.
- 4 Rename the table to SPK\_Table.
- 5 Click **OK**. Your CAGE browser switches to the **Tables** view. CAGE has automatically initialized the normalizers to space breakpoints evenly over the ranges of N and L.

There are two methods for filling tables with optimization results.

- 1 Click the **Optimization** button in the **Processes** pane to return to the Optimization view
- 2 Click the plus to expand the TQ\_Model\_Optimization node, and select the TQ\_Model\_Optimization\_Output node.
- 3 Select **Solution > Fill Tables** (or the toolbar button **Fill tables using optimal settings**).

The Table Filling wizard appears.

- 4 Select the SPK\_Table table and click the button to add it to the list of tables to be filled. Click **Next**.
- 5 Select the SPK\_Table table, and SPK in the list of optimization results, and click the button to select it to fill the table, as shown.




- 6 Click **Next**, then **Finish**.

You see a dialog reporting successful table filling.

- 7 Leave the **View selected item** check box selected to switch to the **Tables** view, then click **Close**. Examine the new spark table.

The other method of filling tables with optimization output uses Data Sets.

- 1 From the **Optimization\_Output** optimization output node, click **Export to Data Set** (  ) in the toolbar (or select **Solution > Export to Data Set**). Click **OK** in the **Export to Data Set** dialog box to accept the defaults.
- 2 CAGE displays the **Data Sets** view. Click **View Data** in the toolbar to see the table of optimization results contained in the new data set **New\_Dataset**.

You can now use this data set (or any optimization results) to fill tables, as you can with any data set.

- 3 Click  (Fill Table From Data Set) in the toolbar.

- 4 Choose to fill the spark table with the SPK optimization output by selecting them in the two lists:
  - a Select `SPK_Table` in the **Table to fill** list.
  - b Select the SPK optimization output in the **Factor to fill table** list.
  - c Click the button **Fill Table** at the bottom right.
- 5 The Table History dialog box appears to show you that CAGE filled the table from the data set. Click **Close** to dismiss the dialog box.
- 6 To see the filled table surface and the optimization output spark values on the same plot, right-click the display and select **Surface**. Recall you already filled the spark table from the optimization results with the table filling wizard, so the table should look unchanged.

See also “Fill Tables from Data” on page 14-2 for more details on using data sets to fill tables.

In the next section you will use a custom fill routine to fill the table.

## Using a Custom Fill Routine to Fill Tables

It can be useful to create your own custom fill function to fill tables from the results of an optimization. Some example situations are:

- You have your own smoothing strategy for certain regions of your look-up tables
- Implementation of an alternative method to the two fill methods supplied
- You want to produce some customized output

You can use a custom fill routine to fill the `SPK_Table` table from the optimization results.

- 1 Create a custom fill function. For this example, you can use the supplied example, `griddataTableFill.m`, which can be found in the `mbctraining` directory. Copy `griddataTableFill.m` to a directory away from the MATLAB root directory, and make sure this directory is on the MATLAB path (or change the current directory to the location where you copied the file).
- 2 At the optimization output node, select **Solution > Fill Tables**.
- 3 The wizard retains a memory so the `SPK_Table` table is already selected to be filled. Click **Next**.



- 4 Similarly, **SPK** is already selected from the list of optimization results to fill the table. Click **Next**.
- 5 Select **Custom** from the **Fill Method** drop down menu. Use the file selector, or enter the name of the fill function you wish to use to fill your tables. In this case, select or enter `griddataTableFill`, and press **Enter**. Note that this function must be on the MATLAB path.
- 6 Click **Finish** to fill the `SPK_Table` table.
- 7 You see a dialog reporting successful table filling. Click **Close**.

In the next section you will add a multiobjective optimization to this project. For next steps, see “Multiobjective Optimization” on page 15-18.

## Multiobjective Optimization

| In this section...   |
|--|
| “Setting Up and Running the Multiobjective Optimization” on page 15-18 |
| “Optimization Output View” on page 15-23                               |
| “Selecting Best Solutions” on page 15-28                               |

---

**Note:** To follow these steps, you must first complete the previous steps in the tutorial to load models and create an optimization. See “Optimization and Automated Tradeoff” on page 15-2.

---

### Setting Up and Running the Multiobjective Optimization

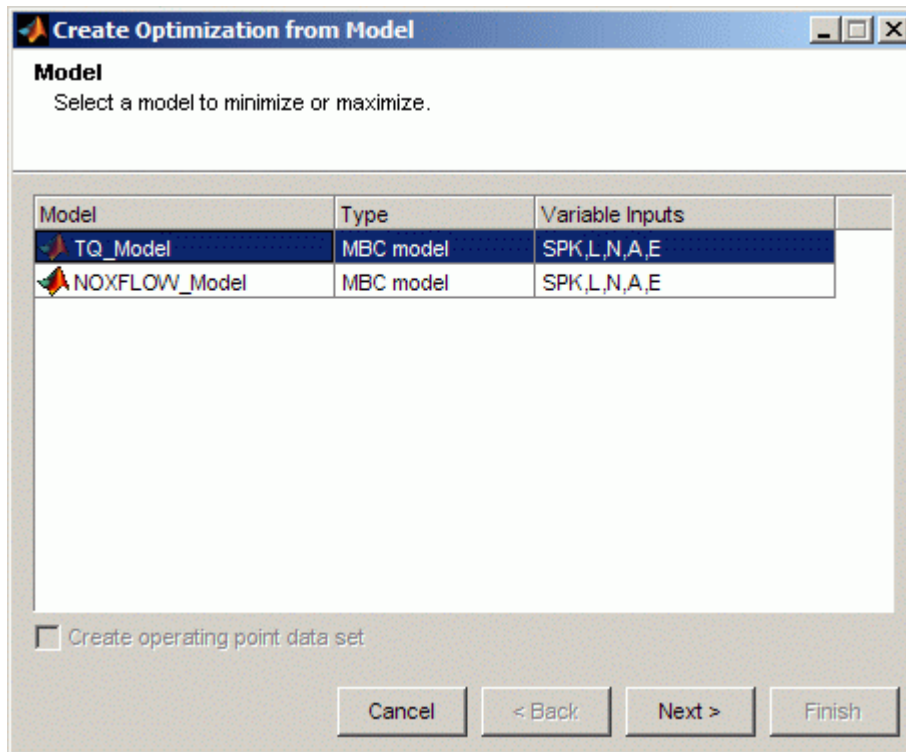
In this optimization you will construct a search for values of spark that maximize values of torque while minimizing values of NOX at a series of (L, N, A, E) points.

To create your optimization,

- 1 Select **Tools > Create Optimization From Model** (or use the toolbar button).

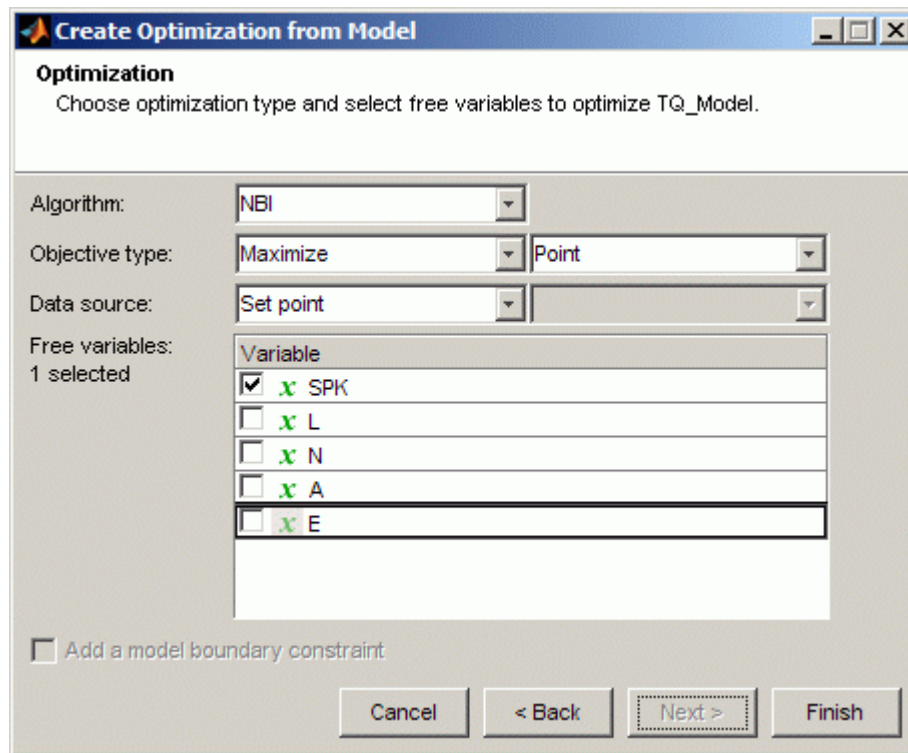
The **Create Optimization From Model** Wizard appears.

- 2 Select TQ\_Model1 as the first model you want to optimize.



Click **Next**.

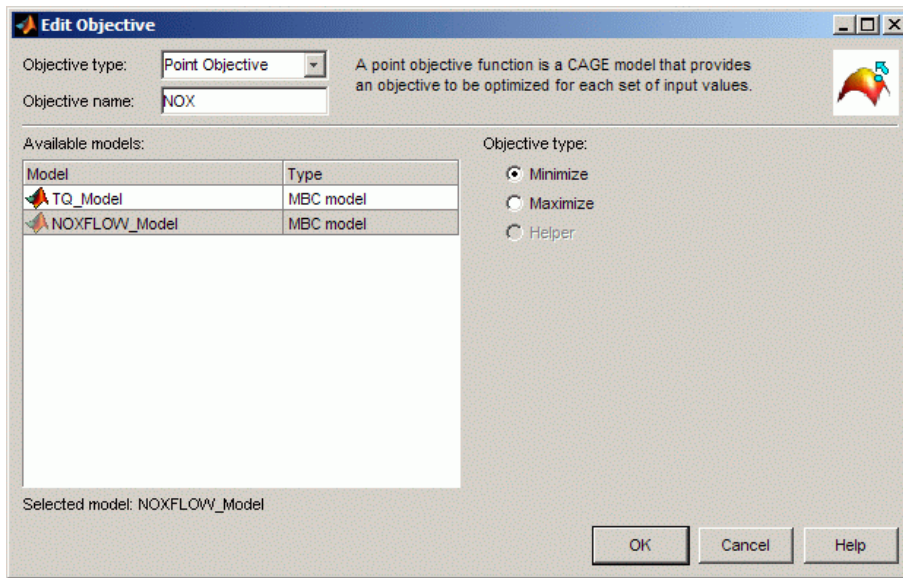
- 3** On this page of the wizard you select the optimization settings.
  - a** Select **NBI** for the **Algorithm**. You must use the NBI algorithm to solve multiobjective optimizations.
  - b** Select **Maximize** for the **Objective type**.
  - c** Clear the check boxes for all the **Free variables** except **SPK**.
  - d** Click **Finish** to create the optimization.



You see the CAGE Browser Optimization view. A new branch named `TQ_Model_Optimization_1` appears in the Optimization tree.

You need to set up your second objective. In the Objectives pane you see a status message informing you that you need to specify a model for the second objective.

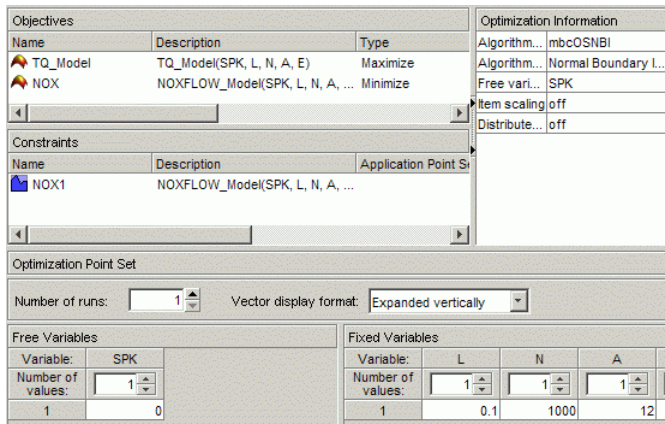
- 1 Double-click `Objective2` to edit the objective.
- 2 Edit the Objective name to `NOX`, and specify `NOXFLOW_Model1` and **Minimize**. Click **OK**.



Import the NOX constraint from your previous optimization.

- 1 Right-click in the Constraints pane and select **Import Constraints**.
- 2 In the Import Constraints dialog box, select the NOX constraint from your previous optimization and click **OK**.


Your CAGE browser should look like the following example.



Your optimization has objectives and a constraint set up and is ready to run. However unless you edit the fixed variable values it will run at a single point, the set point of the variables.

- 1 In the **Optimization Point Set** pane, increase the **Number of runs** to **6**. Notice 6 rows appear in both fixed and free variable values panes, all containing the default set point values of each variable.
- 2 Select **Optimization > Import From Output**. The Import From Output dialog box appears.
  - a Select the previous single objective optimization node, **TQ\_Model\_Optimization\_Output**, in the top list to import values from this optimization.
  - b Clear the check boxes for **SPK**, **A**, and **E**, to leave only **N** and **L** selected for import, and click **OK**.
- 3 View these values in the **N** and **L** columns in the **Fixed Variables** pane.


| Number of runs: <input type="text" value="6"/> |                                | Vector display format: <input type="text" value="Expanded vertically"/> |                                |                                |                                |                                |
|--|--------------------------------|---|--------------------------------|--------------------------------|--------------------------------|--------------------------------|
| Free Variables                                 |                                | Fixed Variables   |                                |                                |                                |                                |
| Variable:                                      | SPK                            | Variable:   | L                              | N                              | A                              | E                              |
| Number of values:                              | <input type="text" value="1"/> | Number of values:   | <input type="text" value="1"/> | <input type="text" value="1"/> | <input type="text" value="1"/> | <input type="text" value="1"/> |
| 1  | 0                              | 1   | 0.1                            | 1000                           | 12                             | 5                              |
| 2  | 0                              | 2   | 0.8                            | 1000                           | 12                             | 5                              |
| 3  | 0                              | 3   | 0.1                            | 3000                           | 12                             | 5                              |
| 4  | 0                              | 4   | 0.8                            | 3000                           | 12                             | 5                              |
| 5  | 0                              | 5   | 0.1                            | 6000                           | 12                             | 5                              |
| 6  | 0                              | 6   | 0.8                            | 6000                           | 12                             | 5                              |

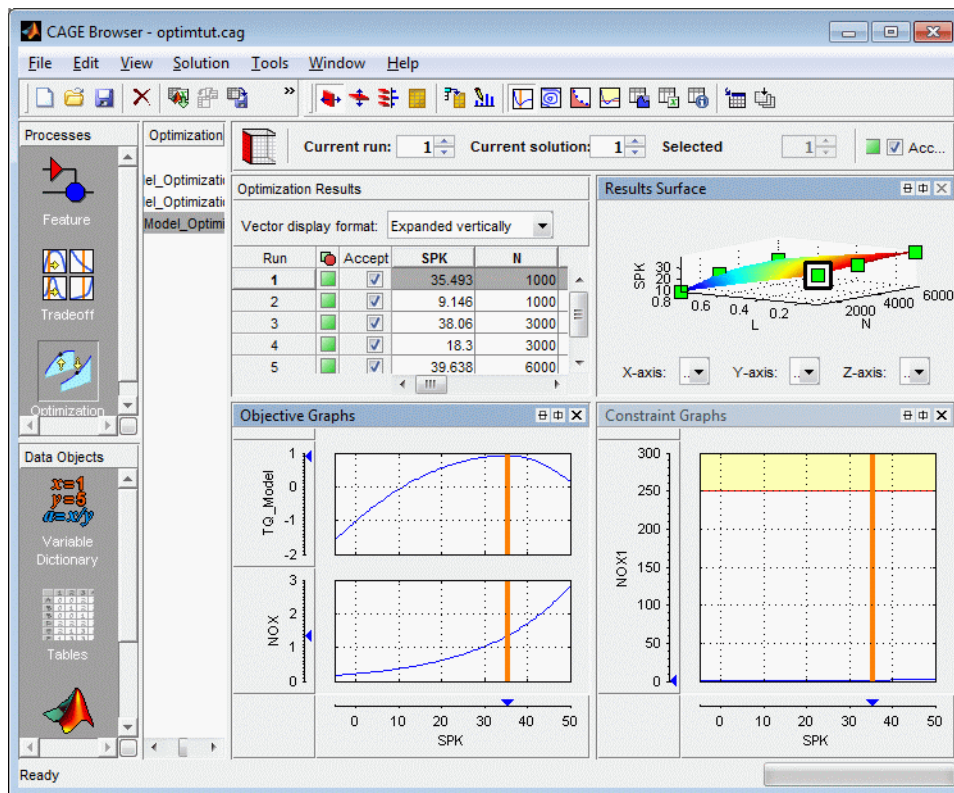
- 4 Click Run Optimization (  ) in the toolbar.

The optimization runs, showing progress messages as each point is evaluated until the optimization is complete. A new node, **TQ\_Model\_Optimization\_1\_Output**, appears under **TQ\_Model\_Optimization\_1** in the Optimization tree.

## Optimization Output View

The view switches to the `TQ_Model_Optimization_1_Output` node in the Optimization tree where you can examine the optimization output.

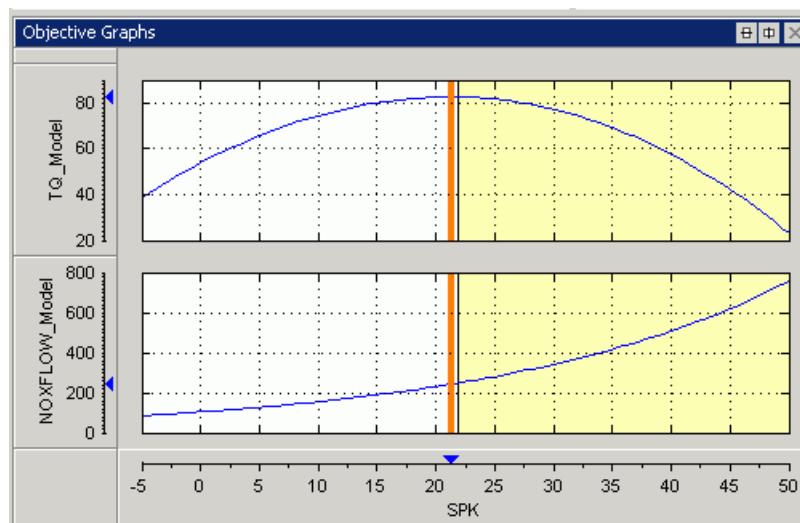
The toolbar buttons determine which view is displayed. The default is the Solution Slice () . The Solution Slice shows one solution at all operating points. That is, you can see a table (and surface plot) of all operating points at once, and you can scroll through the solutions using the **Current solution** controls at the top. At the start all 6 operating points show solution 1. Change solution to 2, and you see the second solution for all 6 operating points, and so on. As this is a multiobjective optimization, there are several solutions for each operating point.



The graphs show the objective functions at the currently selected operating point (highlighted in the table and Results Surface view), with the solution value shown in red.

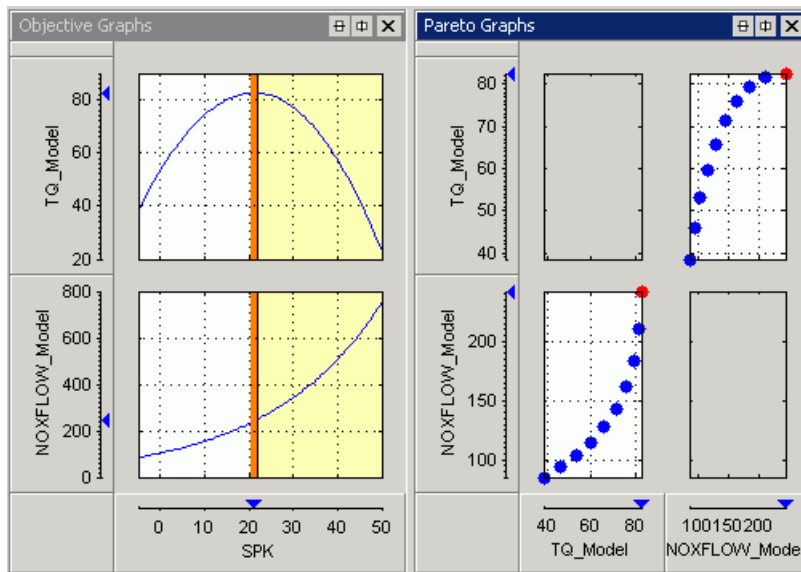
Note that before you run an optimization you can specify how many solutions you want the optimization to find, using the Set Up and Run Optimization toolbar button.

- 1 For an example point, click in the table to select operating point 6, and enter **10** in the **Current solution** edit box. Observe the constraint you applied in the objective function graphs, as shown in the example. Areas in yellow are excluded by constraints. Similarly, if you use a boundary constraint model exported from the Model Browser as a constraint, areas outside the boundary appear in optimization graphs as yellow areas. Note that for some problems the optimization might fail to find a value within the constraints (depending on the constraints and starting values) in which case you might need to run the optimization again to find valid solutions. Choosing more suitable starting values and changing your settings to make constraints less stringent can help in these cases. See “Analyzing Point Optimization Output” in the CAGE User's Guide documentation.




- 2 Right-click the Objective Graphs view and select **Split View > Pareto Graphs**.





The view splits to show both objective and pareto graphs. You can right-click and select **Graph Size** to adjust how many plots can be displayed. In the Pareto Graphs view you can see all solutions found by the optimization at the selected operating point (the selected solution is highlighted in red). Try clicking different points in the pareto graph to see the different solutions in the objective graphs.

- 3 Click Pareto Slice (  ) in the toolbar. This changes the table to display all solutions at a single operating point. You can scroll through the operating points using the **Run** buttons at the top. The pareto graphs always show the currently selected solution in red. Click in the table or the graph to select different solutions.


Recall that the first example, a single-objective optimization, produced a single solution at each point, so you could not view the Pareto Slice. The Pareto Slice is useful to show you the set of optimal tradeoff solutions when using multiobjective optimizations, as in this case. You can use these plots to help you select the best solution for each operating point. As you can see, this example trades off NOX emissions for torque, so it is a judgment call to choose the best depending on your priorities. You will select best solutions in a later section, “Selecting Best Solutions” on page 15-28.

- 4 Click Weighted Pareto Slice (  ) in the toolbar.

Run:  Current run: <none> Current solution: 5


Optimization Results

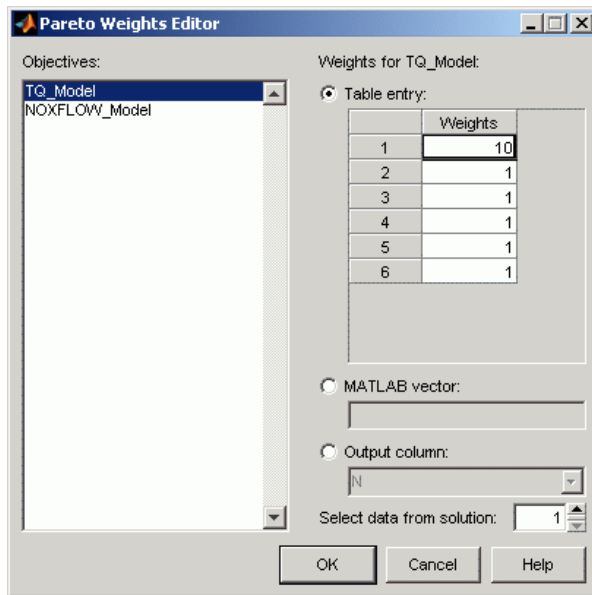
Vector display format: Expanded vertically

| Solution |  Accept | TQ_Model | NOXFLO... |
|----------|--|----------|-----------|
| 1        | <input checked="" type="checkbox"/>  | 119.626  | 124.037   |
| 2        | <input checked="" type="checkbox"/>  | 143.645  | 135.766   |
| 3        | <input checked="" type="checkbox"/>  | 166.544  | 149.986   |
| 4        | <input checked="" type="checkbox"/>  | 187.918  | 167.587   |
| 5        | <input checked="" type="checkbox"/>  | 207.213  | 189.77    |
| 6        | <input checked="" type="checkbox"/>  | 223.797  | 217.909   |
| 7        | <input checked="" type="checkbox"/>  | 237.154  | 253.132   |
| 8        | <input checked="" type="checkbox"/>  | 247.031  | 295.968   |
| 9        | <input checked="" type="checkbox"/>  | 253.294  | 346.526   |
| 10       | <input checked="" type="checkbox"/>  | 255.571  | 404.93    |

This table view displays a weighted sum objective output across all operating points for each solution.

The value in the NOXFLOW\_Model column in the first row shows the weighted sum of the solution 1 values of NOX across all 6 operating points. The second row shows the weighted sum of solution 2 NOX values across all 6 operating points, and so on. This can be useful, for example, for evaluating total emissions across a drive cycle. The default weights are unity (1) for each operating point.

- 5 You can alter these weights by clicking Edit Pareto Weights (  ) in the toolbar. The Pareto Weights Editor appears.



Here you can select models, and select weights for any operating point, by clicking and editing, as shown in the example above. The same weights are applied to each solution to calculate the weighted sums. Click **OK** to apply new weights, and the weighted sums are recalculated.

You can also specify weights with a MATLAB vector or any column in your optimization output by selecting the other radio buttons. If you select **Output column** you can also specify which solution; for example you could choose to use the values of spark from solution 5 at each operating point as weights. Click **Table Entry** again, and you can then view and edit these new values.

---

**Note** Weights applied in the Weighted Pareto Slice do not alter the results of your optimization as seen in other views. You can use the weighted sums to investigate your results only. You need to perform a sum optimization if you want to optimize using weighted operating points.

---

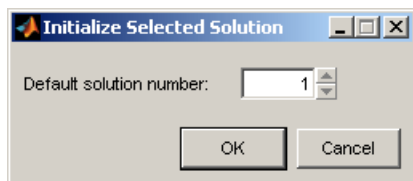
## Selecting Best Solutions

In a multiobjective optimization, there is more than one possible optimal solution at each operating point. You can export all solutions, or you can select a solution for each point. You can use the Selected Solution Slice to collect and export those solutions you have decided are optimal at each operating point.


Once you have enabled the Selected Solution Slice, you can use the plots in the Pareto Slice and Solution Slice to help you select best solutions for each operating point. These solutions are saved in the Selected Solution Slice. You can then export your chosen optimization output for each point from the Selected Solution Slice, or use your chosen optimization output to fill tables.

- 1 In order to choose a single solution at each operating point, you need to enable the Selected Solution Slice. Select **Solution > Selected Solution > Initialize**.

A dialog called Initialize Selected Solution appears. Click **OK**.




The default initializes the first solution for each operating point as the selected solution.


- 2 Click the Selected Solution Slice button (  ) which is now enabled in the toolbar. Observe that the **Current solution** number at the top is not editable, and is initially solution 1 for each operating point you click in the table. You must decide which solution is best for each point. You can select solutions in the **Selected solution** edit box, either here in the Selected Solution Slice or in the Pareto Slice or Solution Slice views.
- 3 In the Selected Solution Slice view, choose a solution for run 6. Enter **6** in the **Current run** edit box, and use the **Selected solution** controls to click through the available solutions and view each solution in the objective and pareto graphs. For example, to select solution 7 as best for the current run, enter **7** in the **Selected solution** edit box.

Leave your chosen solution selected and the table remembers your selection for each run.

If you want to select best solutions from the Pareto Slice or Solution Slice views, either:

- Click Select Solution (  ) in the toolbar.
  - Select the menu item **Solution > Selected Solution > Select Current Solution**.
- 4 The Selected Solution Slice view collects all your selected solutions together in one place. For example, you might want to select solution 7 for the first operating point, and solution 6 best for the second, and so on.
  - 5 In order to use one solution per point from your optimization output to fill tables, you should repeat this process to select a suitable solution for all operating points. Then use the Table Filling From Optimization Results Wizard (**Solution > Fill Tables**) as before — the table is filled with your selected solution for each run.

Alternatively you could fill from a data set containing the selected solutions. To

do this, click Export to Data Set (  ) and click **OK** in the dialog box. Go to the **Data Sets** view (click **Data Sets** in the **Data Objects** pane) to see that the table of optimization results is contained in a new data set. You could use these optimization results to fill tables. Both these table-filling methods are described in “Using Optimization Results to Fill Tables” on page 15-14.

Note that the table in the current view is exported to the data set. If you want to export your selected best solutions for each operating point, make sure you display the Selected Solution Slice before exporting the data. If you export from the Pareto Slice, the new data set contains all solutions at the single currently selected operating point set. If you export from the Solution Slice the new data set will contain the current solution at all operating points.

Recall that the previous example was a single-objective optimization and therefore only had one solution per operating point. In that case the optimization results could be exported directly from the Solution Slice, as there was no choice of solutions to be selected. See “Single-Objective Optimization” on page 15-5.

In the next tutorial section you will duplicate this NBI optimization example and alter it to create a sum optimization. For next steps, see “Sum Optimization” on page 15-30.

## Sum Optimization

### What Is a Sum Optimization?

In this exercise you will use a copy of the NBI optimization from the last example to create a sum optimization.

Up to this point, you have found the optimal values of each objective function at each point of an operating point set individually. A sum optimization finds the optimal value of a weighted sum of each objective function over all free variables simultaneously. The weighted sum is taken over each operating point in the run, and the weights can be edited.

A sum optimization problem without constraints, for  $M$  operating points, is the same as  $M$  individual optimizations; the minimum of the sum is the same as the sum of the minima.

To illustrate this, consider the following example:

Say the objective function is  $f(a,b)$ . Consider  $a$  to be the free variable and  $b$  to be the fixed variable. To set up a sum optimization, at different values of  $b$ , we want to find  $[a_1, a_2, a_3 \dots a_M]$  which minimizes:

$$w_1*f(a_1,b_1) + w_2*f(a_2,b_2) + w_3*f(a_3,b_3) + \dots + w_M*f(a_M,b_M), [1]$$

where  $w_1, w_2$  etc. are the weights.

There are two ways of viewing this problem. It can be viewed as a big optimization problem in an  $M$ -dimensional space for the vector  $[a_1, a_2, a_3, \dots a_M]$  as shown in [1]. Alternatively, as each element of the sum depends on its own subset of the free variables, the problem can be written as  $M$  separate optimization problems, as in [2]:

$$\min w_i*f(a_i,b_i) \text{ for } i=1:M, [2]$$

Once the  $M$  individual problems are solved, the weighted sum can be constructed to get the answer.

When there are sum constraints present, it is not true that a  $M$ -point sum optimization problem is equivalent to solving  $M$  individual optimizations. In this case, all points must

be evaluated together to find the optimal solution that meets the sum constraints across all of the points.

In a sum optimization, the objectives are typically sum objectives. You can use a mixture of point and model sum constraints in a sum optimization. The following instructions describe these settings.

## Procedure

---

**Note:** To follow these steps, you must first complete the previous steps in the tutorial to load models and create optimizations. See “Optimization and Automated Tradeoff” on page 15-2.

---

- 1 Right-click the `TQ_Model_Optimization_1` node in the Optimization tree and select **Duplicate TQ\_Model\_Optimization\_1**.

A copy of the `TQ_Model_Optimization_1` node called `TQ_Model_Optimization_2` appears in the tree. You do not need an existing NBI optimization to create a sum optimization. This example is used for convenience and also to illustrate copying optimizations. This feature can be useful when you are trying different settings to improve your optimizations while keeping previous attempts for comparison.

You use this copy to create a sum optimization. This means that instead of performing the optimization at each point individually, the optimization takes the sum of solutions at all points into consideration. You can apply different weights to operating points, allowing more flexibility for some parts of the optimization.

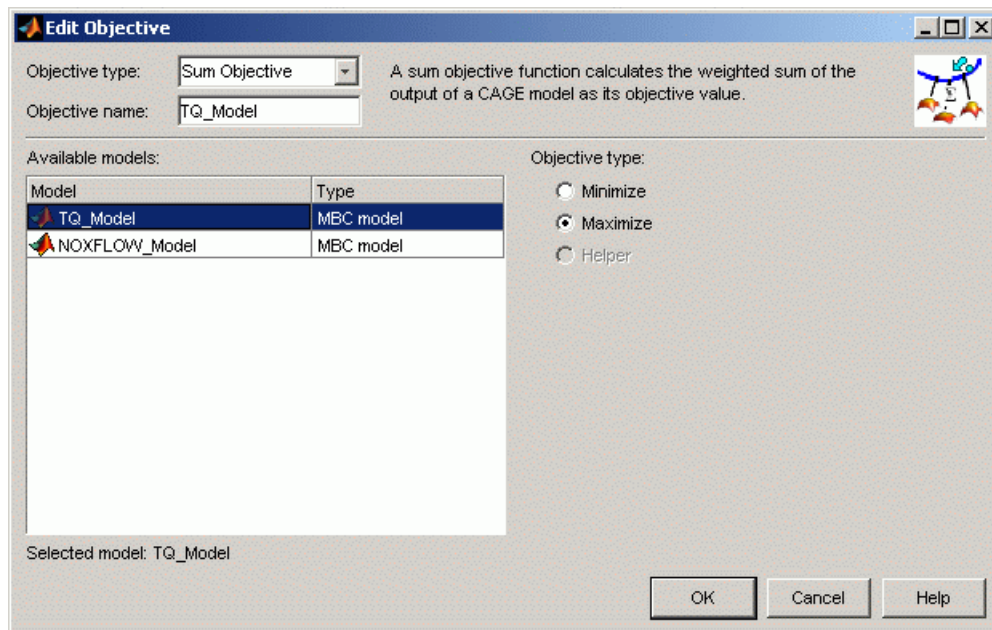
- 2 Select the node `TQ_Model_Optimization_2` and select **Edit > Rename** (or press **F2**). Edit the name to read `SUM_NBI`.

You will edit all the objectives in your existing optimization to be sum objectives.

- 3 Double-click `TQ_Model` (or right-click and select **Edit Objective**). The Edit Objective dialog appears.

To set up your new objective,

- a Select **Sum Objective** from the **Objective Type** drop-down menu.



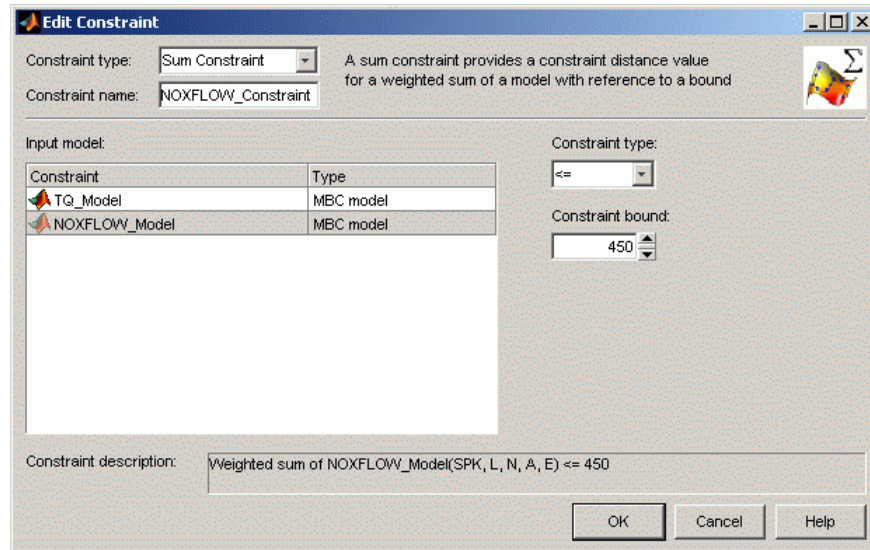
- b Select **TQ\_Model** and make sure **Maximize** is selected. Torque has a strong correlation with fuel consumption so this sort of problem could be useful for a fuel consumption study.
  - c Click **OK** to finish editing the objective.
- 4 Repeat to edit **NOXFLOW\_Model**. Set up a sum objective to minimize NOXFLOW.
- 5 You can also edit your constraint to be a sum constraint. You can use a mixture of point and sum constraints.

To set up your sum constraint,

- a Double-click the constraint **NOX1** and the Constraint Editor appears.
- b Select **Sum Constraint** from the **Constraint Type** drop-down menu, then select **NOXFLOW\_Model** under **Input Model**.
- c Edit the **Constraint name** to **NOXFLOW\_Constraint** to help you later distinguish it from the **NOXFLOW\_Model** in the Optimization view fixed variables pane.




- d Make sure the inequality is  $\leq$ , and enter 450 in the constraint bound edit box as shown.



- e Press **Enter** to dismiss the dialog box and return to the Optimization view.
- 6 You want to perform a sum optimization across a series of operating points. Select **Optimization > Convert to Single Run**. Look at the change in the **Variable Values** panes. Instead of 6 separate runs you now have one run containing 6 operating points. **Number of runs** is now 1, and **Number of Values** is 6 for all variables (you use these controls to set the number of operating points within each run). Each solution will be a sum across all the points in the run.
- 7 Look in the **Fixed Variables** pane for the weights columns, last columns on the right. Enter 5 in the first row (run 1, point 1) for **TQ\_Model\_weights**, **NOX\_weights**, and **NOXFLOW\_Constraint\_weights**, to weight the first operating point by five, as shown.

| Fixed Variables   |                  |   |     |      |    |                       |                           |   |
|-------------------|------------------|---|-----|------|----|-----------------------|---------------------------|---|
| Variable:         | TQ_Model_weights | L | N   | A    | E  | NOXFLOW_Model_weights | NOXFLOW_Constraint_wei... |   |
| Number of values: | 6                | 6 | 6   | 6    | 6  | 6                     | 6                         |   |
| 1                 | (1)              | 5 | 0.1 | 1000 | 12 | 5                     | 5                         | 5 |
|                   | (2)              | 1 | 0.8 | 1000 | 12 | 5                     | 1                         | 1 |
|                   | (3)              | 1 | 0.1 | 3000 | 12 | 5                     | 1                         | 1 |
|                   | (4)              | 1 | 0.8 | 3000 | 12 | 5                     | 1                         | 1 |
|                   | (5)              | 1 | 0.1 | 6000 | 12 | 5                     | 1                         | 1 |
|                   | (6)              | 1 | 0.8 | 6000 | 12 | 5                     | 1                         | 1 |

- 8 You have modified your objectives and constraint for a sum optimization, which is ready to run. Click Run Optimization (  ) in the toolbar.
- 9 There is a wait notice as the optimization runs. There are no progress messages as points are evaluated because sum optimizations do not evaluate points individually.

When the optimization is complete, examine the results at the output node. Select the Pareto Slice table. Look at the objective and constraint results for the ten solutions (number of solutions is specified in the Optimization Parameters dialog). In the Pareto Slice table, negative constraint values are within the constraint. Look at the constraint summary view, where the left and right information corresponds to the constraint inequality; the left value is the distance from the constraint.

For more information see “Interpreting Sum Optimization Output” in the CAGE User’s Guide documentation.

For next steps, see “Automated Tradeoff” on page 15-35.

## Automated Tradeoff

Once you have set up an optimization you can fill tables in a tradeoff using automated tradeoff. You can select cells and fill them from the results of an optimization. The cells you select in the tradeoff table define the operating point set for the optimization.

Set up a tradeoff as follows (also described in “Setting Up a Tradeoff Calibration” on page 12-2).


---

**Note:** To follow these steps, you must first complete the previous steps in the tutorial to load models and create an optimization. See “Optimization and Automated Tradeoff” on page 15-2.

---

- 1 Select **File > New > Tradeoff**.

This takes you to the **Tradeoff** view. You need to add tables to the tradeoff.

- 2 Click  (Add New Table). This opens the Table Setup dialog.
- 3 Enter **Spark** as the table **Name**.
- 4 Select **L** as the **Y input** and **N** as the **X input**.
- 5 Click **Select** to open the Select Filling Item dialog.

- a Select the radio button to **Display variables**.

- b Click to select **SPK**.

- c Click **OK** to return to the Table Setup dialog.


- 6 Leave **10** as the size of the rows and columns (the speed and load axes), and **0** as the initial value, and click **OK**.

A new **Spark** table appears in the **Tradeoff** tree. CAGE has automatically spaced the normalizers evenly over the ranges of **N** and **L**.

- 7 Click to expand the **New\_Tradeoff** tree and select the **Spark** table node to view the new table.

You need to select the cells where you want to apply automated tradeoff. Create a region within the table:

- 1 Highlight a rectangle of cells in the **SPK** table by clicking and dragging. Note that a large region can take a very long time to evaluate. Try four cells to start with.

- 2 Click  (or right-click and select **Extrapolation Regions > Add Selection**) to define the region. The cells become light blue.

To use automated tradeoff on the cells in the defined region,

- 1 Select **Inputs > Automated Tradeoff** or click the toolbar button .

The Automated Tradeoff dialog appears, showing a list of available optimizations in your session that are set up and ready to run.

- 2 Select your **TQ\_Model\_Optimization\_1** multiobjective optimization (of **Type: Normal Boundary Intersection Algorithm**) to apply to the tradeoff and click **OK**.
- 3 Click **OK** in the following dialog to optimize only the table cells that are in the region, rather than all cells.

The automated tradeoff optimization runs. The results appear in the selected cells in the table, as shown in the example. You could optimize several regions and then use these results to extrapolate across the whole table.

