

Model-Based Calibration Toolbox

For Use with MATLAB®

Computation

Visualization

Programming

CAGE User's Guide

Version 2.0

The MathWorks

How to Contact The MathWorks:



www.mathworks.com	Web
comp.soft-sys.matlab	Newsgroup



support@mathworks.com	Technical support
suggest@mathworks.com	Product enhancement suggestions
bugs@mathworks.com	Bug reports
doc@mathworks.com	Documentation error reports
service@mathworks.com	Order status, license renewals, passcodes
info@mathworks.com	Sales, pricing, and general information



508-647-7000	Phone
--------------	-------



508-647-7001	Fax
--------------	-----



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

For contact information about worldwide offices, see the MathWorks Web site.

Model-Based Calibration Toolbox CAGE User's Guide

© COPYRIGHT 2001-2003 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 or for the federal government of the United States. By accepting delivery of the Program, the government hereby agrees that this software qualifies as "commercial" computer software within the meaning of FAR Part 12.212, DFARS Part 227.7202-1, DFARS Part 227.7202-3, DFARS Part 252.227-7013, and DFARS Part 252.227-7014. The terms and conditions of The MathWorks, Inc. Software License Agreement shall pertain to the government's use and disclosure of the Program and Documentation, and shall supersede any conflicting contractual terms or conditions. If this license fails to meet the government's minimum needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to MathWorks.

MATLAB, Simulink, Stateflow, Handle Graphics, and Real-Time Workshop are registered trademarks, and TargetBox is a trademark of The MathWorks, Inc.

Other product or brand names are trademarks or registered trademarks of their respective holders.

Printing History:	November 2000	Online Only	New for Version 1.0 (Release 12.1)
	July 2002	Online only	Revised for Version 1.1 (Release 13)
	September 2003	Online only	Revised for Version 2.0 (Release 13SP1)

Getting Started

1

About CAGE	1-2
Calibrating Lookup Tables Using CAGE	1-2
Comparing Calibrations to Data	1-3
Starting CAGE	1-4
Processes	1-6
Data Objects	1-7
How to Use This Manual	1-8
System Requirements	1-10
Hardware Requirements	1-10
Operating System Requirements	1-10
Required MathWorks Products	1-11
Optional MathWorks Products	1-11

Tutorial: Feature Calibration

2

What Are Feature Calibrations?	2-2
Setting Up Calibrations	2-3
Setting Up Variables	2-3
Setting Up Models	2-6
Creating a Feature Calibration	2-10
Setting Up the Strategy	2-12
Setting Up the Tables	2-13
Calibrating a Feature	2-16
Calibrating the Normalizers	2-16

Calibrating the Tables	2-19
Calibrating the Feature	2-21
Exporting Calibrations	2-25

Tutorial: Tradeoff Calibration

3

What Is a Tradeoff Calibration?	3-2
Creating a Tradeoff Calibration	3-3
Adding Tables to a Tradeoff Calibration	3-5
Displaying the Models	3-6
Performing the Tradeoff Calibration	3-7
Calibrating the Normalizers	3-9
Setting Values for Other Variables	3-11
Filling Key Operating Points	3-12
Filling the Table by Extrapolation	3-15
Exporting Calibrations	3-17

Tutorial: Data Sets

4

Setting Up the Data Set	4-2
Opening an Existing Calibration	4-4
Importing Experimental Data into a Data Set	4-4
Adding an Item to a Data Set	4-6
Comparing the Items in a Data Set	4-7
Viewing the Data Set as a Table	4-7
Viewing the Data Set as a Plot	4-8
Displaying the Error	4-9

Coloring the Display	4-11
Reassigning Variables	4-14

Tutorial: Filling Tables from Data

5

Setting Up a Table and Experimental Data	5-2
Adding Variables	5-3
Adding a New Table	5-3
Setting Up the Normalizers	5-5
Importing Experimental Data	5-7
 Filling the Table from the Experimental Data	 5-10
 Selecting Regions of the Data	 5-14
 Exporting the Calibration	 5-16

Tutorial: Optimization and Automated Tradeoff

6

Getting Started	6-2
Setting Up Your Optimization Session	6-3
 Single-Objective Optimization	 6-6
Using the Optimization Wizard	6-6
Setting Objectives, Constraints and Operating Point Sets	6-10
Running the Optimization	6-13
 Multiobjective Optimization	 6-17
Optimization Output Node	6-22
Selecting Best Solutions	6-28

Sum Optimization	6-32
Automated Tradeoff	6-35
Worked Example Optimization	6-38
Using the Worked Example Optimization	6-39
Creating an Optimization from Your Own Algorithm	6-46
Step 1: Examine the Algorithm	6-47
Step 2: Create a CAGE Optimization Function	6-50
Step 3: Define the Optimization Options	6-51
Step 4: Add the Algorithm to the Optimization Function	6-54
Step 5: Register Your Optimization Function with CAGE	6-58
Step 6: Verify Your New Optimization	6-59

Using CAGE

7

How to Use CAGE	7-2
CAGE Views and Processes	7-3
Setting Up Your Variable Items	7-6
Importing and Exporting a Variable Dictionary	7-8
Adding Variable Items	7-9
Using the Variable Menu	7-11
Using Aliases	7-12
Setting Up Your Models	7-13
Importing Models	7-15
Adding New Function Models	7-17
Renaming and Editing Models	7-18
Exporting Calibrations	7-21
Specifying Locations of Files	7-22

8

About Normalizers	8-2
Calibrating the Normalizers	8-3
Initializing Breakpoints	8-4
Filling Breakpoints	8-5
Optimizing Breakpoints	8-9
Normalizer View	8-13
Input/Output Display	8-15
Normalizer Display	8-16
Breakpoint Spacing Display and Deleting Breakpoints	8-17
Viewing the Comparison Pane	8-20

Feature Calibrations

9

Performing Feature Calibrations	9-2
Setting Up a Feature Calibration	9-4
Adding a Feature	9-4
Assigning a Model	9-5
Setting Up Your Strategy	9-5
Calibrating the Tables	9-11
Initializing Table Values	9-12
Filling Table Values	9-13
How CAGE Fills Tables	9-13
Optimizing Table Values	9-16
Filling the Table by Extrapolation	9-19
Inverting a Table	9-21
Inverting One-Dimensional Tables	9-23
Inverting Two-Dimensional Tables	9-26
Table View	9-29

Viewing a Table	9-31
Using the Graph of the Table	9-32
Comparing the Strategy and the Model	9-33
Calibrating the Feature Node	9-37
Initializing the Feature	9-37
Filling the Feature	9-39
Optimizing the Feature	9-41
Feature View	9-44
Feature Menu	9-45

Tradeoff Calibrations

10

Performing a Tradeoff Calibration	10-2
Setting Up a Tradeoff Calibration	10-5
Adding a Tradeoff	10-5
Adding Tables to a Tradeoff	10-6
Displaying Models in Tradeoff	10-7
Calibrating Tables in a Tradeoff Calibration	10-9
Setting Values of Other Variables	10-11
Determining a Value at a Specific Operating Point	10-13
Using Regions	10-15
Multimodel Tradeoffs	10-17
Adding a Multimodel Tradeoff	10-18
Calibrating Using a Multimodel Tradeoff	10-20

Using the Optimization View	11-2
Setting Up Optimizations	11-4
Optimization Wizard Step 1	11-4
Optimization Wizard Step 2	11-5
Optimization Wizard Step 3	11-8
Optimization Wizard Step 4	11-9
Optimization Wizard Step 5	11-10
Optimization Wizard Step 6	11-11
Running Optimizations	11-12
Optimization Node Toolbar	11-14
Optimization Parameters Dialog	11-15
Objective and Constraint Editors	11-21
Optimization Output Node	11-25
Optimization Output Node Toolbar	11-25
Solution View	11-25
Pareto View	11-27
Weighted Pareto View	11-28
Selected Solution View	11-31
Automated Tradeoff	11-33
Using Automated Tradeoff	11-33
What Are Appropriate Optimizations?	11-35
User-Defined Optimization	11-37
Overview of Optimization Function Structure	11-38
Checking User-Defined Optimizations into CAGE	11-40
About the Worked Example Optimization Algorithm	11-42
Optimization Template	11-45
List of Optimization Functions	11-47
Methods of cgoptimstore	11-47
Methods of cgoptimoptions	11-48

Alphabetical List of Functions 11-51

Data Sets

12

Data Sets Views 12-2

Setting Up Data Sets 12-4

- Importing Experimental Data 12-4
- Importing Data from a Table in Your Session 12-6
- Specifying the Factors Manually 12-7
- Creating a Factor from the Error Between Factors 12-10

Viewing Data in a Table 12-11

Plotting Outputs 12-13

Using Color to Display Information 12-16

- Coloring a Plot 12-16
- Restricting the Color 12-18

Linking Factors in a Data Set 12-20

Assigning Columns of Data 12-22

Manipulating Models in Data Set View 12-23

Filling Tables from Experimental Data 12-24

- Creating Rules 12-27

Calibration Manager

13

Setting Up Tables 13-2

Copying Table Data from Other Sources	13-5
Table Properties	13-6
Floating-Point Precision	13-6
Polynomial Ratio, Fixed Point	13-8
Lookup Table, Fixed Point	13-10

Surface Viewer

14

The Surface Viewer in CAGE	14-2
Viewing a Model or Strategy	14-3
Setting Variable Ranges	14-5
Displaying the Model or Feature	14-7
Displaying Errors	14-12
Feature Error Data	14-12
Prediction Error Data	14-12
Making Movies	14-14
Printing and Exporting the Display	14-16

Manual Calibration and the History Display

15

Using the Manual Calibration View	14-2
Adding and Deleting Tables	14-3
Adding Tables	14-3
Deleting Tables	14-4

Using the History Display	14-5
Resetting to Previous Versions	14-7
Comparing Versions	14-8

Index

Getting Started

This section includes the following topics:

About CAGE (p. 1-2)

Introducing the CAGE browser part of the Model-Based Calibration Toolbox. You can use CAGE to calibrate lookup tables using models and data. You can trade off competing objectives, and validate calibrations against data.

Starting CAGE (p. 1-4)

How to start CAGE and navigate between processes, tables, data, variables, and models.

How to Use This Manual (p. 1-8)

How to find information in this User's Guide, with links to tutorials and reference chapters for all CAGE functionality.

System Requirements (p. 1-10)

Hardware and operating system requirements, and required and related products from The MathWorks.

About 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 models from the Model Browser part of the Model-Based Calibration Toolbox. From these models, CAGE builds steady-state ECU calibrations.

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

Calibrating Lookup Tables Using CAGE

There are two different types of calibration that you can perform using CAGE:

- Feature calibration
- Tradeoff calibration

Feature Calibration

A feature calibration compares a model of an estimated signal with a lookup table (or algebraic collection of tables) that estimates the same signal in the ECU. CAGE finds the optimum calibration for the lookup table(s).

For example, a typical engine subsystem controls the spark angle to produce the peak torque; that is, the Maximum Brake Torque (MBT) spark. Using the Model Browser, you can build a statistically sound model of MBT spark, over a range of engine speeds and relative air charges, or loads. Use the feature calibration to fill a lookup table by comparing the table to the model.

Tradeoff Calibration

A tradeoff calibration fills lookup tables by comparing models of different engine characteristics at key operating points.

For example, there are several models of important engine characteristics, such as torque and nitrogen oxides (NOX) emissions. Both models depend on the spark angle. At a particular operating point, a slight reduction of torque can result in a dramatic reduction of NOX emissions. Thus, the calibrator uses

the value of the spark angle that gives this reduction in NOX emissions instead of the spark angle that generates maximum torque.

Comparing Calibrations to Data

You can compare your calibrations to experimental data for validation.

For example, after completing a calibration, you can import experimental data from a spreadsheet. You can use CAGE to compare your calibration to the data.

Starting CAGE

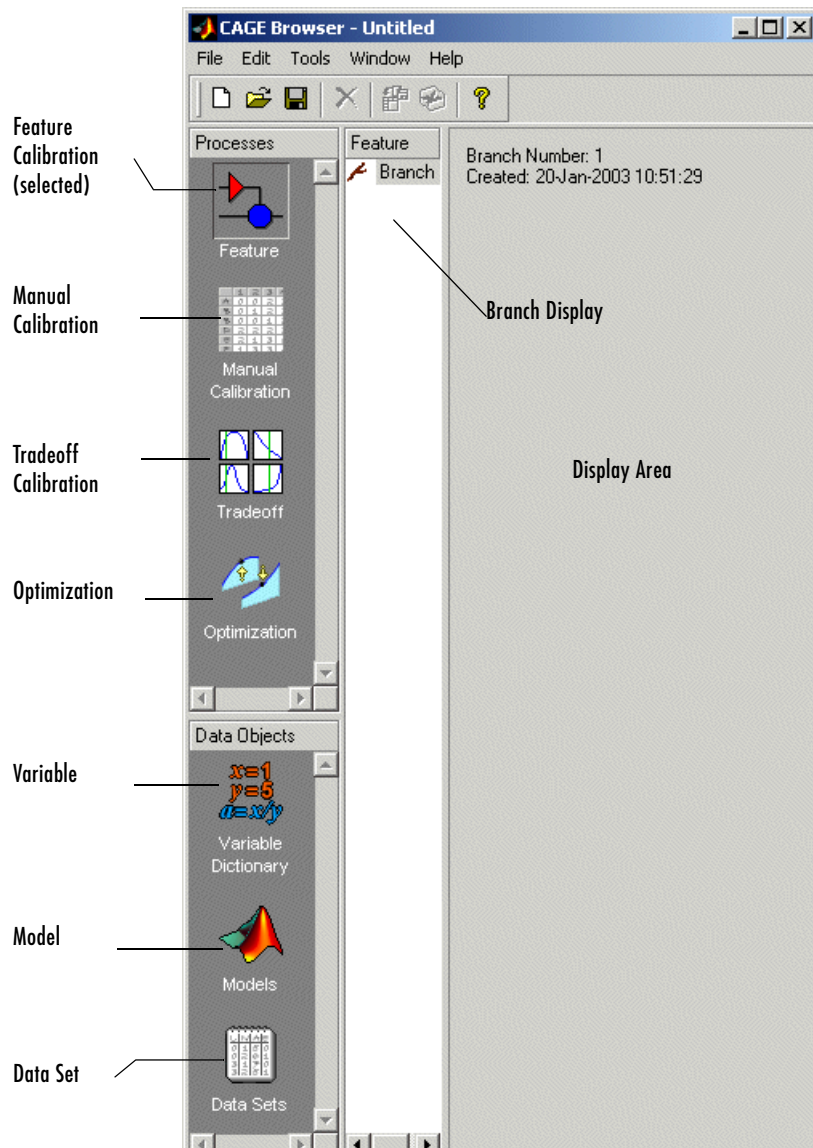
To start the application, type

```
cage
```

The view of CAGE depends on two things:

- The process or object type that you are viewing
- The item you highlight in the branch display (tree)

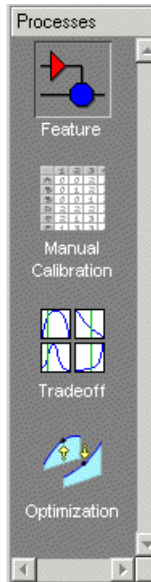
When you open CAGE, it looks like this.



CAGE includes a **Processes** pane and a **Data Objects** pane to help you identify the type of calibration you want to do and the data objects that you intend to use. You use the buttons in these panes to navigate.

Processes

The **Processes** pane enables you to select the type of calibration that you want to perform with CAGE. For example, if you want to calibrate lookup tables by comparing them to a model, select **Feature**.

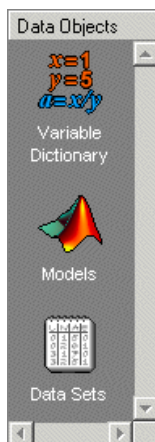


The **Processes** pane has four buttons:

- **Feature** shows the **Feature** view, with any tables or strategies that are associated with that feature.
For more information, see “Feature Calibrations” on page 9-1.
- **Manual Calibration** enables you to calibrate tables manually. It also acts as a store for all the tables and normalizers in your session.
- **Tradeoff** shows the **Tradeoff** view, with a list of the tables and models to display. For more information, see “Tradeoff Calibrations” on page 10-1.
- **Optimization** gives you access to the optimization functionality of CAGE. Here you can set up optimizations and automated tradeoffs. See “Optimization in CAGE” on page 11-1.

Data Objects

The **Data Objects** pane lets you identify the different objects that you need to perform the calibrations with CAGE. For example, if you want to fill lookup tables by comparing them to models, you need to include models. Select **Models** to view the models in your session.



The **Data Objects** pane has these buttons:

- **Variable Dictionary** stores all the variables, constants, and formulas in your session. Click **Variable Dictionary** to view and edit any variables in any part of your session.
For more information, see “Setting Up Your Variable Items” on page 7-6.
- **Models** stores all the models in your session. Click **Models** to show a graphical display of the models in your session.
For more information, see “Setting Up Your Models” on page 7-13.
- **Data Sets** enables you to see the data produced by your objects, and enables you to compare this with other data such as experimental data.
For more information, see “Data Sets” on page 12-1.

For a more detailed overview of CAGE functionality in different views and links to in-depth help on each topic, see “Using CAGE” on page 7-1.

How to Use This Manual

This manual is the CAGE User's Guide. See also the Model Browser User's Guide for information on the other main interface of the Model-Based Calibration Toolbox.

Learning CAGE

There are four tutorial chapters, with worked examples to guide you through using the tools in CAGE:

- “Tutorial: Feature Calibration” on page 2-1 describes how to set up and calibrate lookup tables by reference to a model.
- “Tutorial: Tradeoff Calibration” on page 3-1 describes how to calibrate lookup tables using tradeoff calibrations.
- “Tutorial: Data Sets” on page 4-1 describes how to validate calibrations using experimental data.
- “Tutorial: Filling Tables from Data” on page 5-1 describes how to fill lookup tables using experimental data.
- “Tutorial: Optimization and Automated Tradeoff” on page 6-1 describes how to set up and run optimizations, including automated tradeoff.

Using CAGE

- “Using CAGE” on page 7-1 describes how to set up CAGE sessions before performing calibrations and gives an overview of where in CAGE to find all the functionality for different processes.
- “Normalizers” on page 8-1 describes what normalizers are, and how to space breakpoints in a normalizer.
- “Feature Calibrations” on page 9-1 describes how to calibrate lookup tables by reference to models built using the model browser.
- “Tradeoff Calibrations” on page 10-1 describes how to calibrate lookup tables by adjusting one value to fulfill different objectives.
- “Optimization in CAGE” on page 11-1 describes how to use the optimization functions, including automated tradeoffs, and describes all the functions available for user-defined optimizations.
- “Data Sets” on page 12-1 describes how to use CAGE to compare calibrations to experimental data, and how to use experimental data to fill lookup tables.

- “Calibration Manager” on page 13-1 describes how to use the **Calibration Manager**.
- “Surface Viewer” on page 14-1 describes how to use the **Surface Viewer**.
- “Manual Calibration and the History Display” on page 15-1 describes how to use the **History** viewer, and how to add and delete tables and manually calibrate tables.

Training Material

The files for the tutorial chapters are all contained in the `matlab/toolbox/mbc/mbctraining` directory.

System Requirements

This section lists the following:

- Hardware requirements
- Operating system requirements
- Required MathWorks products
- Optional MathWorks products

Hardware Requirements

The Model-Based Calibration Toolbox has been tested on the following processors:

- Pentium, Pentium Pro, Pentium II, Pentium III, and Pentium IV
- AMD Athlon

Minimum memory:

- 256 MB

Minimum disk space:

- 450 MB for the software and the documentation

Operating System Requirements

The Model-Based Calibration Toolbox is a PC-Windows only product.

The product has been tested on

Microsoft Windows NT, 2000, and 98.

You can see the system requirements for MATLAB online at

<http://www.mathworks.com/products/system.shtml/Windows>

Required MathWorks Products

Model-Based Calibration requires the following other MathWorks products:

- Simulink®
- Optimization Toolbox
- Statistics Toolbox
- Extended Symbolic Toolbox

Optional MathWorks Products

The Model-Based Calibration Toolbox can use the following MathWorks product:

- Neural Networks Toolbox

Tutorial: Feature Calibration

This section includes the following topics:

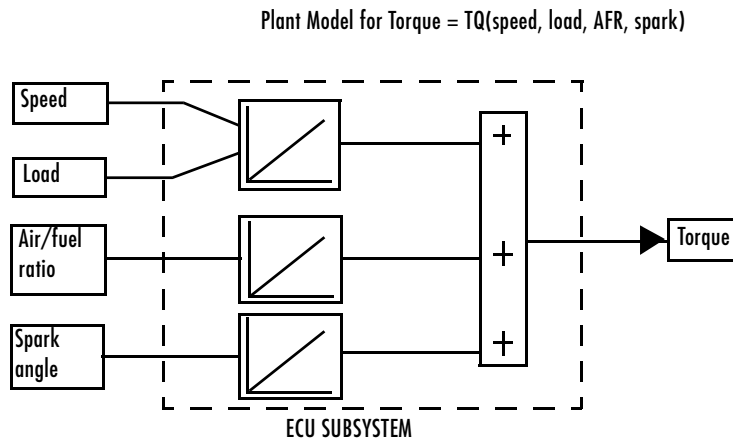
What Are Feature Calibrations? (p. 2-2)	Introduction to feature calibrations. These allow you to calibrate tables by comparing a strategy (or collection of tables) to a model.
Setting Up Calibrations (p. 2-3)	How to set up variables and models in order to calibrate tables.
Creating a Feature Calibration (p. 2-10)	Setting up a strategy (or collection of tables) and setting up tables to prepare for calibration.
Calibrating a Feature (p. 2-16)	How to calibrate normalizers, tables, and the whole feature (collection of tables) at once, by comparing a strategy and a model.
Exporting Calibrations (p. 2-25)	How to export calibrations to file.

What Are Feature Calibrations?

The feature calibration process within the Model-Based Calibration Toolbox 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.

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.



In this 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.

Setting Up Calibrations

Start CAGE by typing

```
cage
```

at the MATLAB prompt.

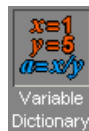
Note If you have a CAGE session open, select **File -> New -> Project**.

Before you can perform a calibration, you must set up the variable dictionary and models that you want to use.

Setting 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** displays all the variables, constants, and formulas in a session.

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,

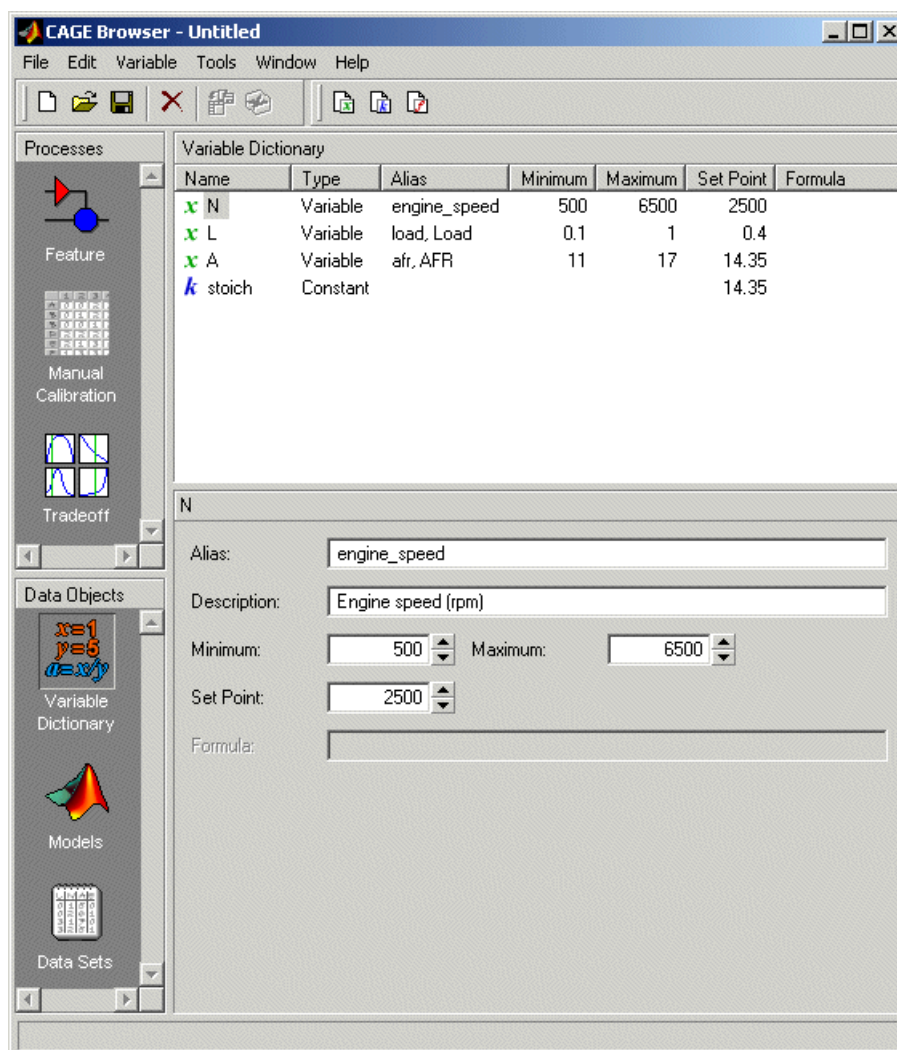
1 Select **File -> Import -> Variable Dictionary**.

- 2 Select the `tutorial.xml` file found in `matlab\toolbox\mbc\mbctraining` and click **Open**.

This imports a set of variables and a constant. In our example, the variable dictionary contains


- `N`, engine speed
- `L`, load
- `A`, AFR
- The stoichiometric constant, `stoich`

Your display should resemble the following.



Adding and Editing Variables and Constants

To add a variable for the spark angle,

- 1 Click  in the toolbar. This adds a new variable to your dictionary.

- 2 Select **Edit** → **Rename** 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:

- 5 Enter S, spark in the **Alias** edit box.
- 6 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.

- 7 Enter 25 in the **Set Point** edit box to specify the preferred value for spk.

For more information about the variables, see “Setting Up Your Variable Items” on page 7-6.

Setting Up Models

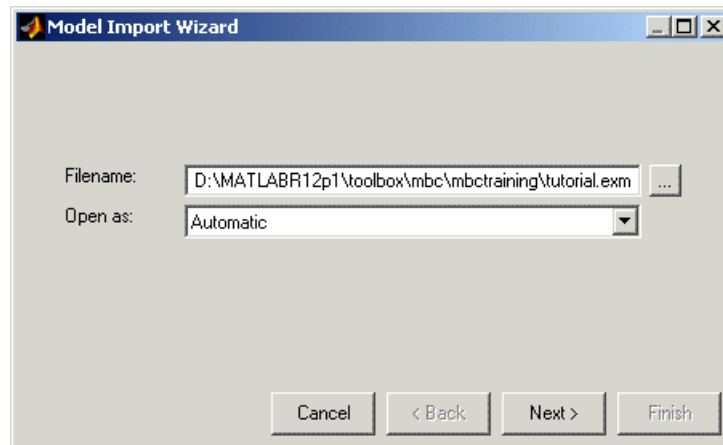
A model in the Model-Based Calibration Toolbox is a function of a set of variables. Typically, you construct a model using the Model Browser; then you can use CAGE to calibrate lookup tables by reference to the model.

The following example uses a model of how torque behaves with varying spark angle, air/fuel ratio, engine speed, and load.

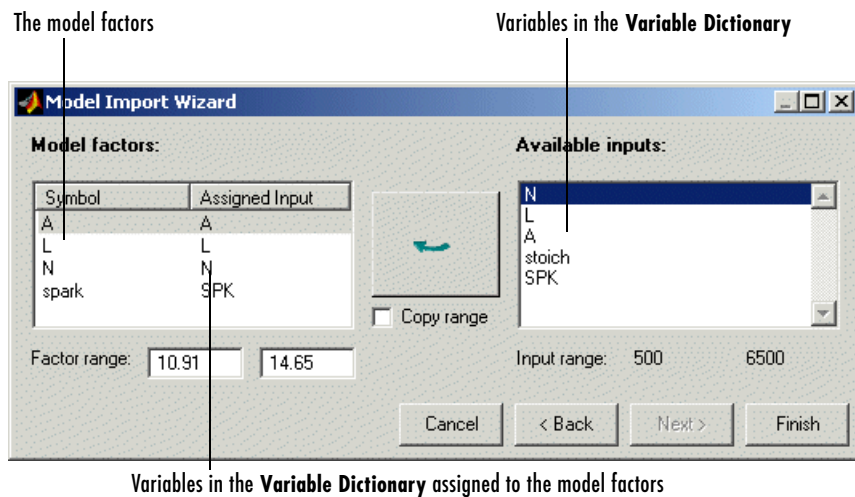
Importing a Model

To import a model built using the Model Browser,

- 1 Select **File** -> **Import** -> **Model**, which starts the **Model Import Wizard**.



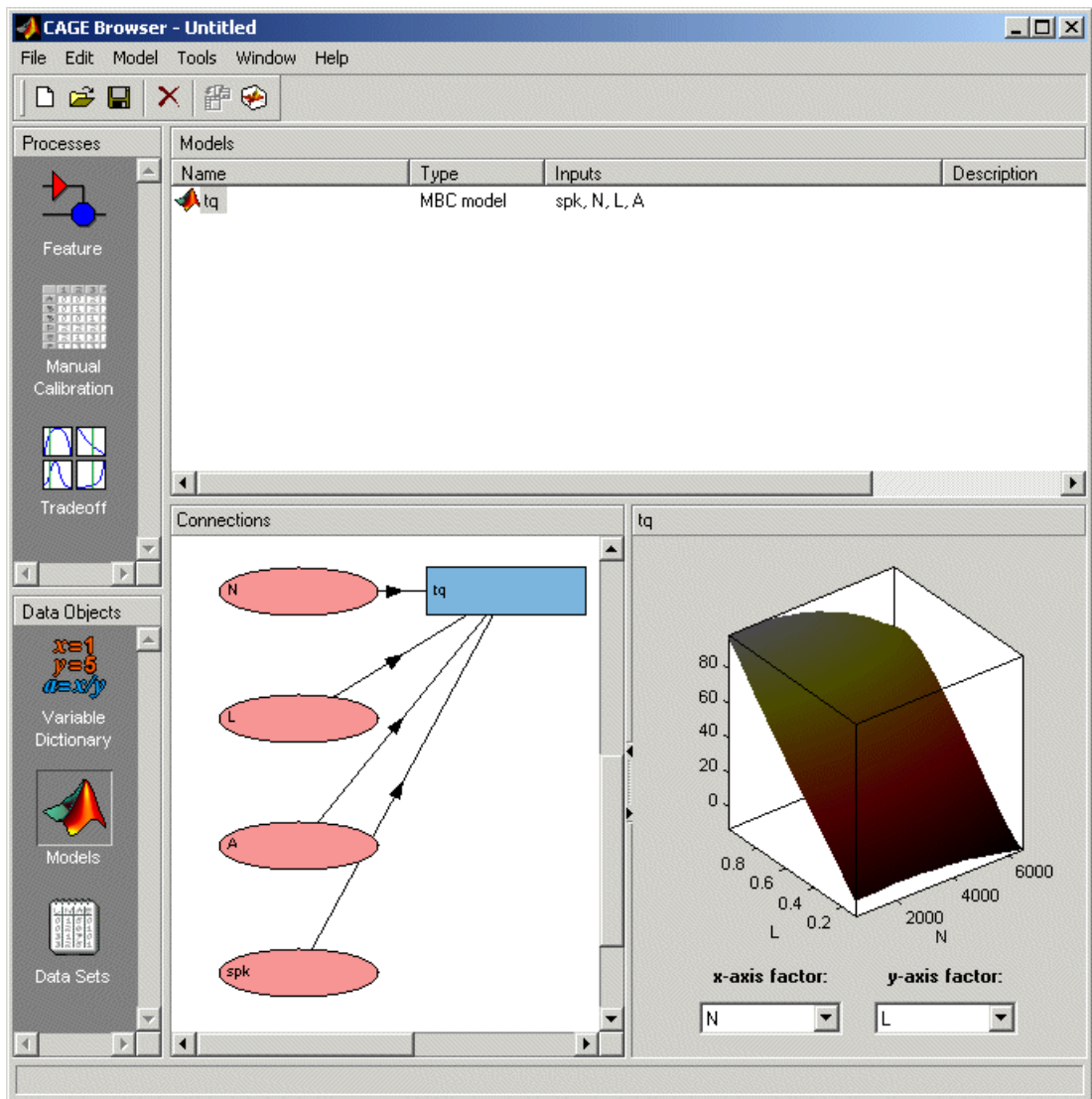
- 2 Click **...** to browse for the correct model file.
- 3 Select the `tutorial.exm` file, found in `matlab\toolbox\mbc\mbctraining` (this is a copy of the torque model built in the Model Browser's quick start tutorial), and click **Open**.
- 4 To accept the default setting, leave **Open as:** Automatic and click **Next**.
- 5 There are two models stored in this file, `tq` and `knot`. Highlight `tq` and click **Next** to show the following.



CAGE automatically assigns variables in the variable dictionary to the model input factors or their aliases (as long as names are exact).

6 Click **Finish** to complete the wizard.

When you complete the wizard, you return to the **Models** view.



For more information about models, see “Setting Up Your Models” on page 7-13.

Creating a Feature Calibration

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:

- 1 Select **File** -> **New** -> **Feature**.

This automatically displays the **Feature** pane and creates a new feature.

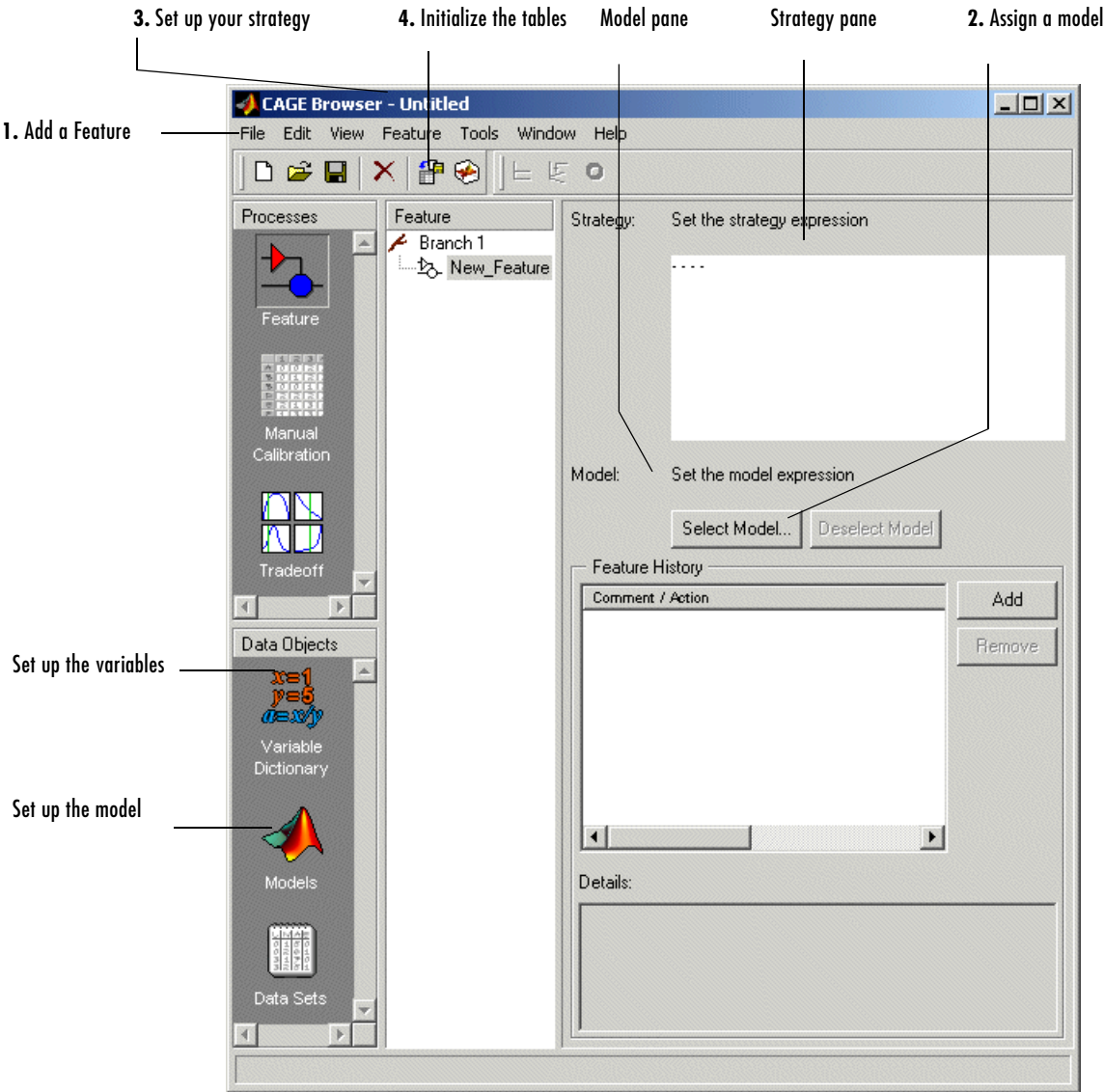
- 2 Click **Select Model**. This automatically selects tq, the torque model, for you (because there is currently only one model in your project).

You can see the model appear above the **Select Model** button.

- 3 Create a strategy. For more information, see the next section, “Setting Up the Strategy” on page 2-12.

A strategy is a collection of tables. The Model-Based Calibration Toolbox uses Simulink[®] to enable you to graphically specify the collection of tables for a feature.

- 4 Set up your tables. For more information, see the following section, “Setting Up the Tables” on page 2-13.



Setting Up the Strategy

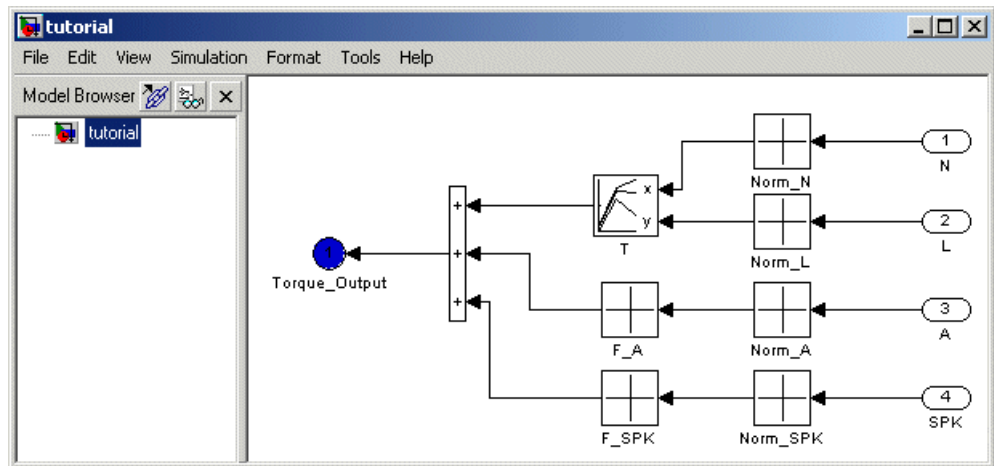
The toolbox uses Simulink to graphically specify the strategy.

Importing a Strategy

To import a strategy,

- 1 Select **File -> Import -> Strategy**.
- 2 Select the file called `tutorial.mdl`, found in `matlab\toolbox\mbc\mbctraining`, and click **Open**.
- 3 This opens the **Import Strategy** dialog box. To view the strategy, click **Manual**.

This opens the following Simulink window.



This shows how the strategy is built up.

- 4 Now double-click the blue circle labeled `Torque_Output`.


Note This shuts down the Simulink window and parses the new feature into the calibration browser.

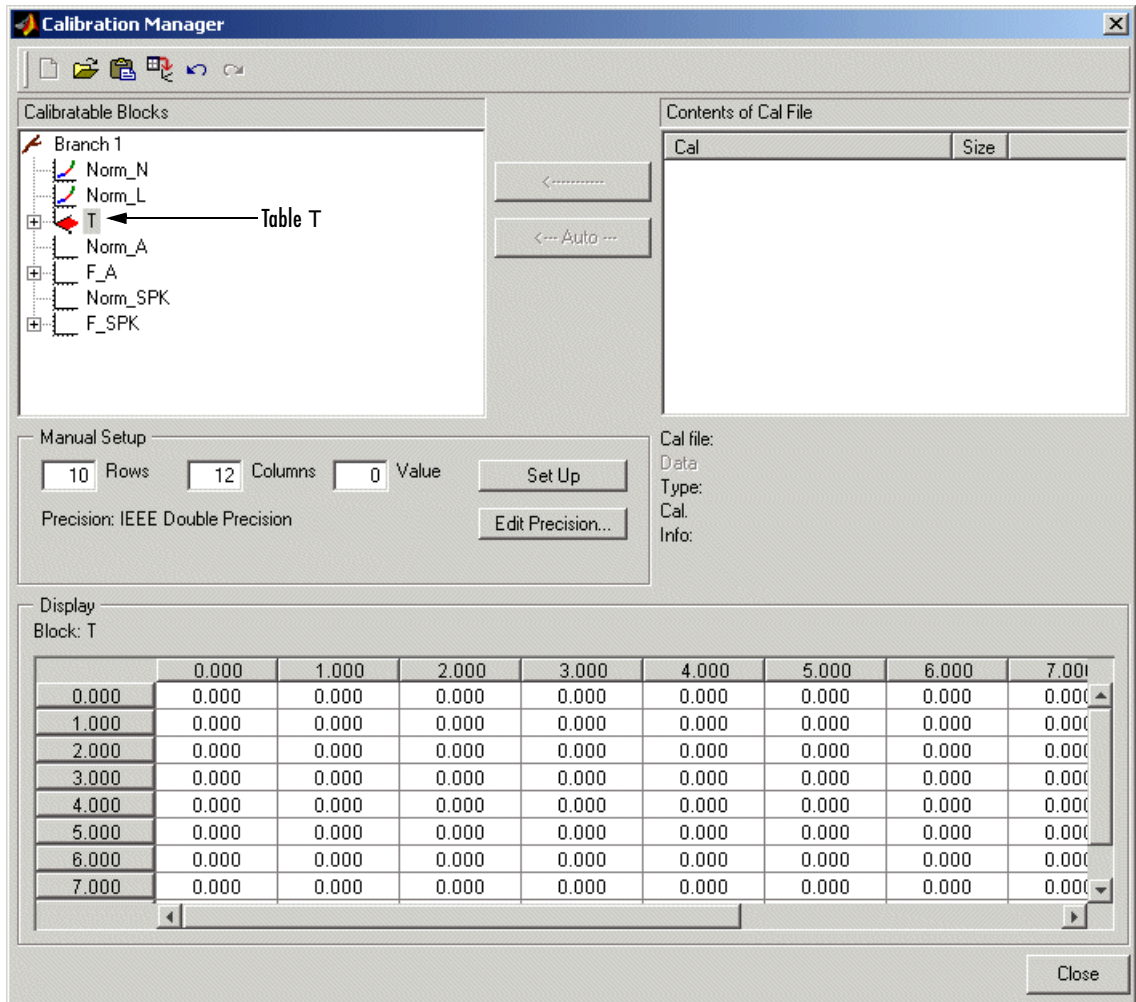
The `New_Feature` is the output of the algebraic equation of tables. You can see this parsed into the **Strategy** pane.

For a more detailed description of the strategies, see “Setting Up Your Strategy” on page 9-5.

Setting Up the Tables

Currently, the lookup tables have neither rows nor columns, so you must set up the tables.

Click  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 in each cell to be 0.

- 4** Click **Set Up** to change the **Display** pane to show the table is set up.
- 5** Follow the same procedure for the F_A table. In other words,
 - a** Highlight the F_A node.
 - b** Set the number of rows to be 10.
 - c** Set the value in each cell to be 0.
 - d** Click **Set Up**.
- 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.


Calibrating a Feature

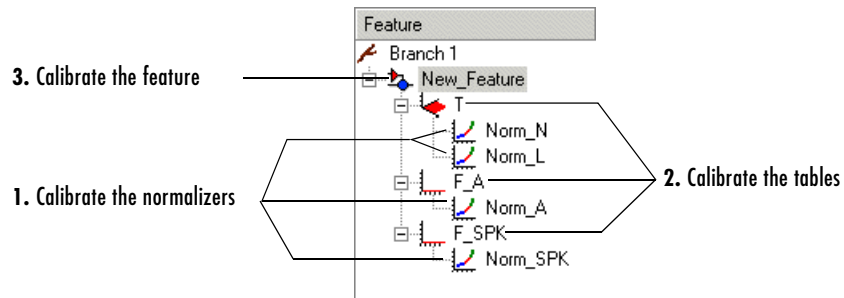
The feature contains both a strategy (which is a collection of tables) and a model. You can use CAGE to fill the lookup tables using the model as a reference.

These are the three steps to calibrate a feature:

- 1 Calibrate the normalizers.
- 2 Calibrate the tables.
- 3 Calibrate the feature as a whole.

These steps are described in the next sections.

Click the expand icon, , to expand the nodes and display all the tables and normalizers in the feature.

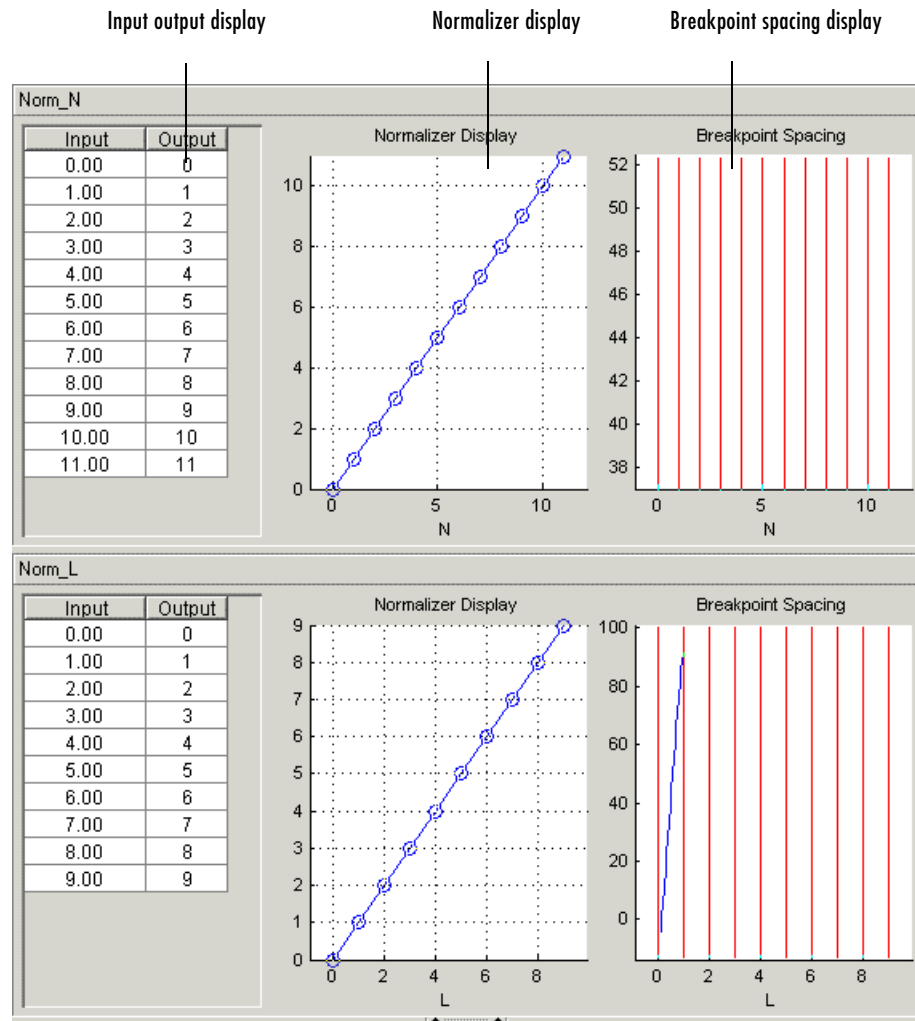


Each node in the display has a different view and different operations.

Calibrating 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 “Normalizer View” on page 8-13.

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  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  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.

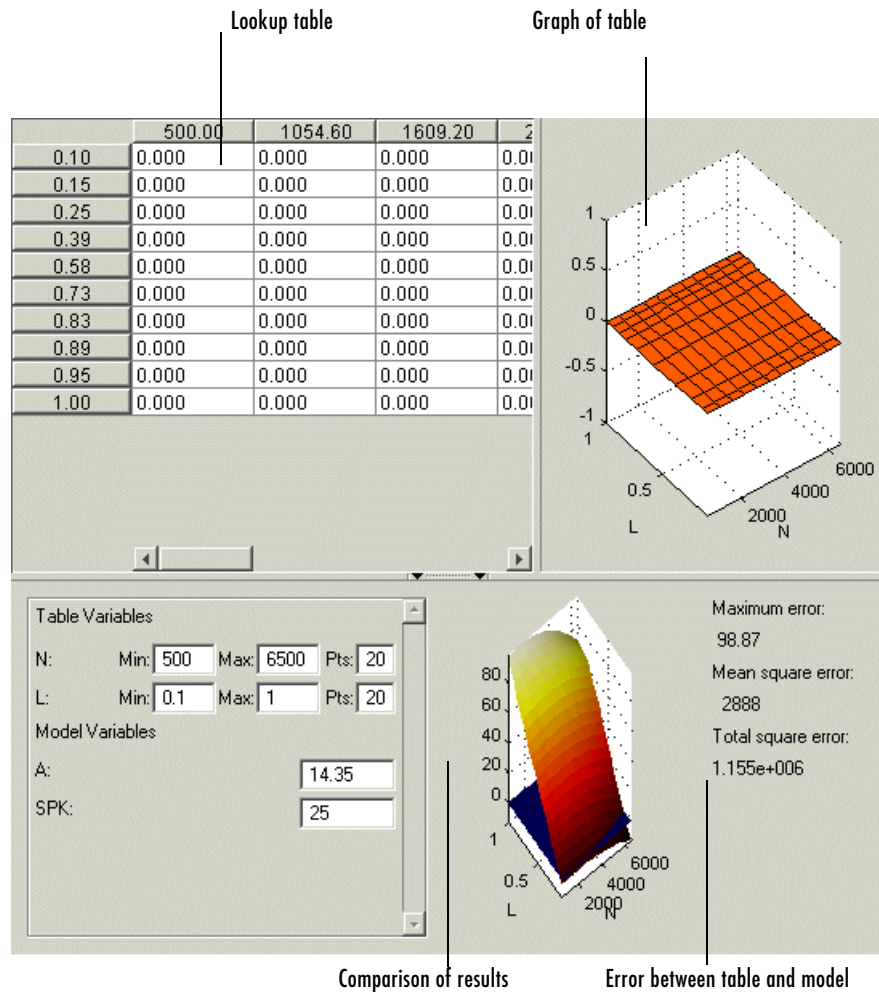
For more information about calibrating the normalizers, see “Normalizers” on page 8-1.

You can now calibrate the lookup tables; this is described in the next section.

Calibrating 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 three panes: the table, the graph, and the comparison-of-results pane.

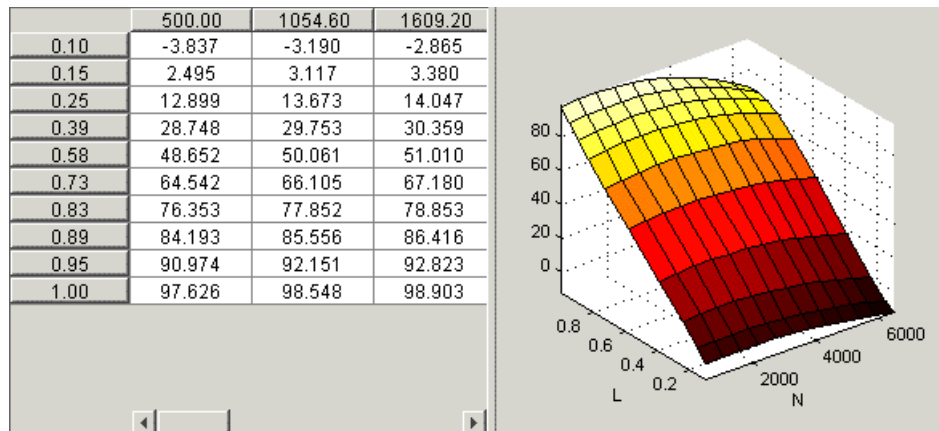
To fill the table with values of the model at the appropriate operating points,

1 Click  on the toolbar.

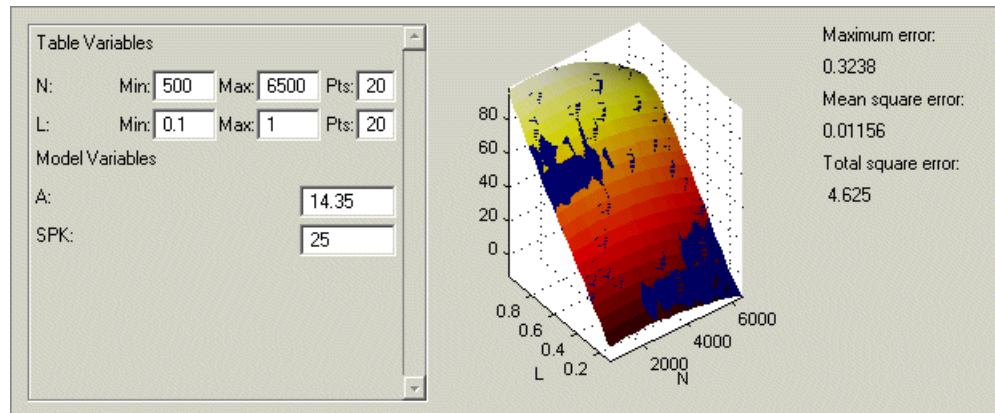
This opens a dialog box that suggests the set points of AFR and SPK as appropriate values for evaluating the model over the range of N and L.

2 Click **OK**.

The following view shows the table filled with values of the model.



The following comparison-of-results pane shows just how good a fit the strategy is to the model.



The model is represented by the multicolored surface and the strategy is the blue surface.

The table T is now filled with values of the model at these operating points.

For more information about the process of filling tables, see “Calibrating the Tables” on page 9-11.

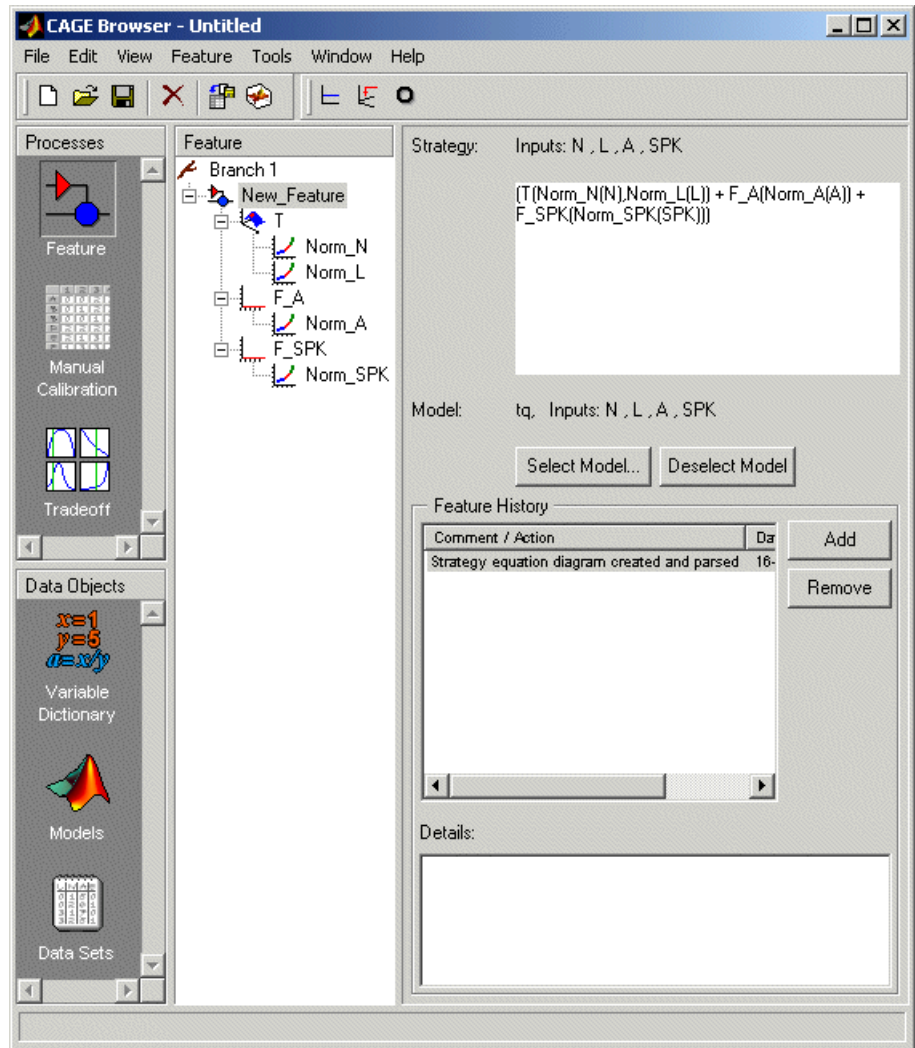
Now you must fill the tables F_A and F_SPK and their normalizers. The tables are modifiers for AFR and the spark angle respectively. These steps are described in the next section.

Calibrating the Feature

A feature is a strategy (which is a collection of tables) and a model. Currently the torque table, T, is filled with values of 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 New_Feature node.

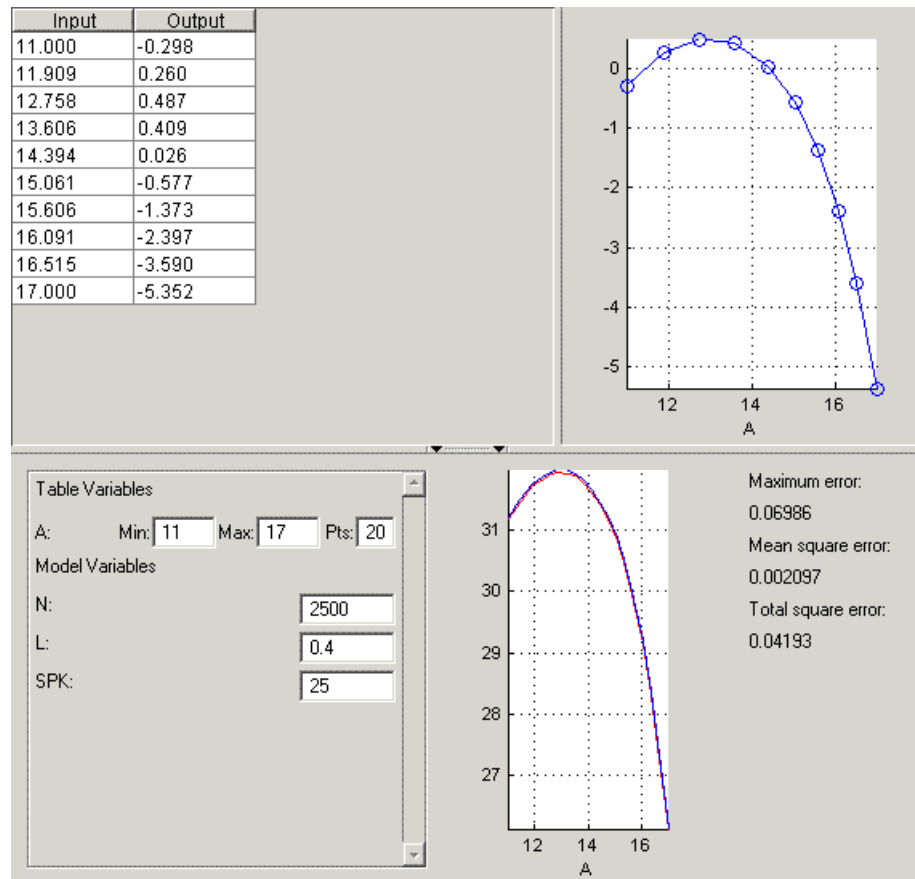


To calibrate all the tables and their normalizers,

- 1 Select **Feature** → **Fill** to open a dialog box.
- 2 Confirm the variable ranges and the table-filling order by clicking **OK** in the **Feature Filling Options** dialog box.

All three tables and normalizers are filled.


As the model and the feature are four-dimensional objects, it is difficult to fully view a comparison between the feature and the model. A meaningful comparison is shown in the lower half of the following figure (select the F_A node in the branch display). The equation $model = strategy$ is rearranged so that the table is compared to the model and the remainder of the strategy.



This display shows that the range of the normalizer for F_A is 11 to 17, the range of AFR. The lower pane shows a comparison between the red strategy and a slice through the model, over the range of AFR.

You can use CAGE to improve on these results. CAGE can run an optimization routine over the feature to minimize the total square error between the model and the feature.

To optimize the feature,

- 1 Select the `New_Feature` node.
- 2 Click . This opens a dialog box, suggesting ranges for the variables.
- 3 To confirm the default variable ranges and table-filling order, click **OK** in the dialog box.

This reduces the error between the feature and the model.

To view this reduction in error, select the `F_A` node in the branch display.

Notice that the mean square error between the model and the feature over this range of values is 0.001348, which is less than the 0.002097 previously obtained.

This completes the calibration of the torque feature.

For more information about calibrating features, see “Calibrating the Feature Node” on page 9-37.

You now need to export the calibration for the ECU.

Exporting Calibrations

To export your feature,

- 1** Select the `New_Feature` node in the branch display.
- 2** Select **File -> Export -> Calibration**.
- 3** Choose the type of file you want to save your calibrations as. You can choose from
 - **Comma Separated Value (.csv)**
 - **MAT-file (.mat)**
 - **M-file script**
- 4** For the purposes of this tutorial, select **Comma Separated Value (.csv)**.
- 5** Enter `tutorial.csv` as the file name and click **Save**.

This exports the successful calibration, ready for the ECU.

Note that 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.

Tutorial: Tradeoff Calibration

This section includes the following topics:

What Is a Tradeoff Calibration? (p. 3-2) Introducing tradeoff calibrations.

Creating a Tradeoff Calibration (p. 3-3) Adding tables and displaying models to set up a tradeoff calibration.

Performing the Tradeoff Calibration (p. 3-7) How to use tradeoff: calibrating normalizers, setting value for other variables, filling key operating points, and filling the table by extrapolation.

Exporting Calibrations (p. 3-17) How to export your calibration.

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.

Creating a Tradeoff Calibration

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 “Using CAGE” on page 7-1.

To create a tradeoff calibration,

- 2** 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 3-5.
- “Displaying the Models” on page 3-6 describes how you display the models of torque and NOX emissions.

The screenshot shows the CAGE Browser interface with the following components and annotations:

- 1. Open the project file.** Points to the 'Tradeoff' icon in the 'Processes' sidebar.
- 2. Add a new tradeoff.** Points to the 'New Tradeoff' button in the 'Tradeoff' pane.
- 3. Add tables.** Points to the 'Tables' pane, which currently displays 'No tables in trade-off'.
- 4. Display the models.** Points to the right-pointing arrow button in the 'Display Options' pane, used to move models from the 'Available models' list to the 'Current display' area.

The 'Available models' list in the 'Display Options' pane contains:

- TQ_Model(SPK, L, N, A, E)
- NOXFLOW_Model(SPK, L, ...)

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.

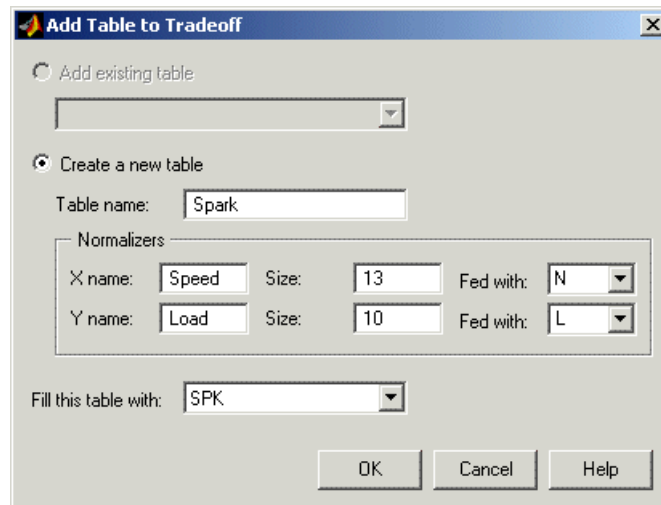
Both models have five inputs. The inputs for the torque and NOX models are

- 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 . This opens the **Add Table to Tradeoff** dialog.



- 2 Enter Spark as the Table Name.
- 3 Enter Speed as the **X Name** and Load as the **Y name**.

- 4 Enter 13 as the size of the speed axis.
- 5 Enter 10 as the size of the load axis.
- 6 Fill the Speed axis with N and the Load axis with L.
- 7 Fill the table with SPK (spark angle).
- 8 Click **OK**.

Before you can perform the calibration, you must display the models.

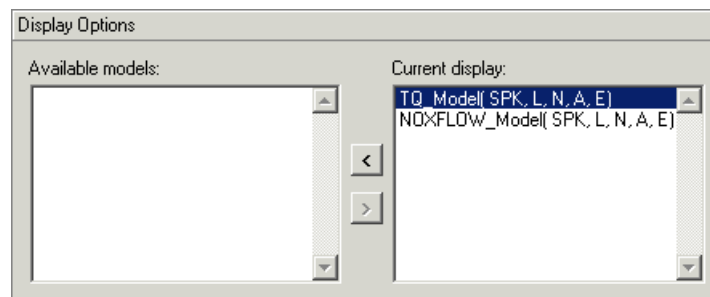
Displaying the Models

For this tutorial, you are comparing values of the torque and NOX models. Thus, you need to display these models.

To display both models,

- 1 Highlight `TQ_Model` and click **>** to include the model in the current display.
- 2 Repeat for `NOXFLOW_Model`.

The **Display Options** pane following shows both models selected for display.



You can now calibrate the tradeoff.

Performing the Tradeoff Calibration

You now fill the lookup table for spark angle by trading off gain in torque for reduction in NOX emissions.

The method that you use to fill the lookup table is

- Obtain the maximum possible torque.
- Restrict NOX to below 250 g/hr at any operating point.

To perform the tradeoff calibration, follow the instructions in the next four sections:

- 1** Calibrate the normalizers.
- 2** Set values for the other variables.
- 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.

3 Tutorial: Tradeoff Calibration

5. Export the calibration.

4. Fill the table by extrapolation.

3. Fill key operating points in the spark table.

1. Calibrate the normalizers.

2. Set values for the other variables.

The screenshot shows the CAGE Browser interface for tradeoff calibration. The main window title is "CAGE Browser - tradeoffInit.cag". The interface includes a menu bar (File, Edit, View, Table, Tools, Window, Help), a toolbar, and several panels:

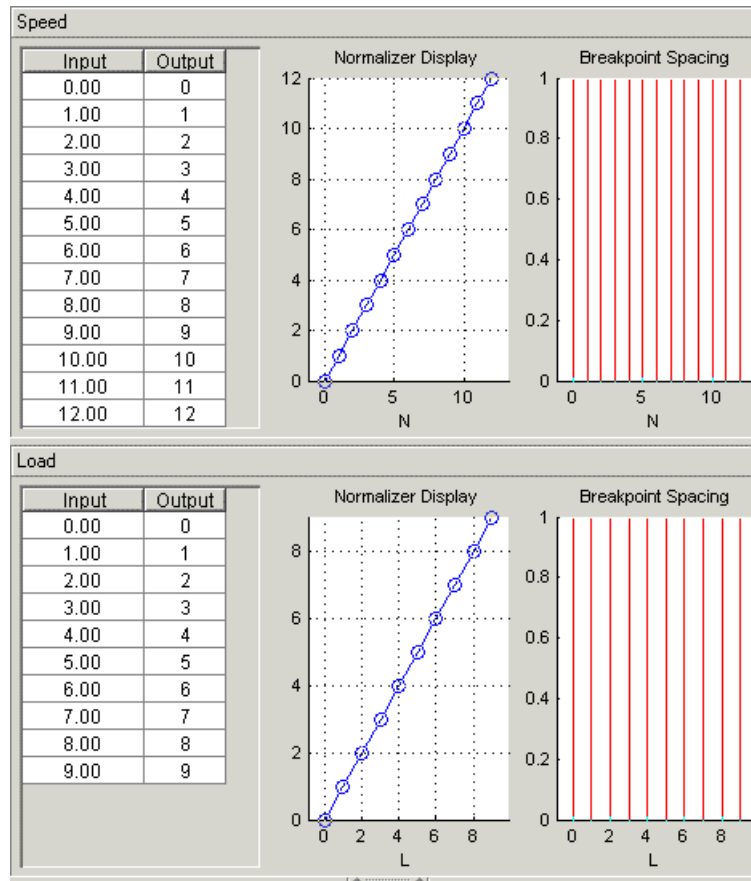
- Processes:** Shows a tree view with "Branch 1", "New_Tradeoff", "Spark", "Speed", and "Load".
- Tradeoff:** A table with columns for Spark, Speed, and Load. The Spark column is highlighted in blue, and the value 0.000 is selected. The table is titled "Spark: N = 500, L = 0.1. Cell filled with SPK.".
- 3D Plot:** A 3D surface plot showing the tradeoff surface. The axes are labeled 0.5, 1, 2000, 4000, and 6000.
- 2D Plots:** A grid of plots showing the relationship between variables. The top row shows "TQ_Model" vs "SPK", "A", and "E". The bottom row shows "NOXFLOW_Model" vs "SPK", "A", and "E". The plots are labeled with values 0, 14.3, and 6. The values 2.7096 and 0.16166 are displayed on the right side of the plots.

	500.000	10
0.100	0.000	()
0.200	0.000	()
0.300	0.000	()
0.400	0.000	()
0.500	0.000	()
0.600	0.000	()
0.700	0.000	()
0.800	0.000	()


Calibrating the Normalizers

A normalizer is the axis of the lookup table (which is the collection of breakpoints). This section describes how to space the breakpoints over the ranges of speed and load.

Expand the tree by clicking in the branch display so you can see the normalizers Speed and Load. Highlight either normalizer by clicking 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.

- Click  to space the breakpoints evenly over the ranges of speed and load.
- To accept the default ranges of the variables, click **OK** in the **Breakpoint Initialization Options** dialog box.

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). So you must set values for AFR and EGR for the models.

To set constant values of AFR and EGR for all operating points,

- 1 Highlight the Spark node in the branch display.
- 2 In the lower pane, check that the value for **A** is 14.3, the stoichiometric constant.
- 3 Enter 0 as the value for **E**.

The screenshot shows the Tradeoff software interface. On the left, a tree view under 'Tradeoff' shows 'Branch 1' expanded to 'New_Tradeoff', with 'Spark' highlighted. Three lines point from the instructions to the 'Spark' node, the 'A' plot, and the 'E' plot.

The main window displays a table with the following data:

	500.000	1000.000
0.100	0.000	0.000
0.200	0.000	0.000
0.300	0.000	0.000
0.400	0.000	0.000
0.500	0.000	0.000
0.600	0.000	0.000
0.700	0.000	0.000

Below the table is a 3D surface plot showing a green grid on a flat surface. The axes are labeled 0.5, 1, 2000, 4000, 6000 and -1, 0, 1.

The bottom section contains a grid of plots for 'TQ_Model' and 'NOXFLOW_Model' against 'SPK', 'A', and 'E'. The 'A' plot has a value of 14.3 and the 'E' plot has a value of 0. The 'TQ_Model' plots show values of 5.4899 and 0.72677.

You can now fill the spark angle lookup table. The process is described next.

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.

You maximize the torque and restrict NOX emissions to below 250 g/hr.

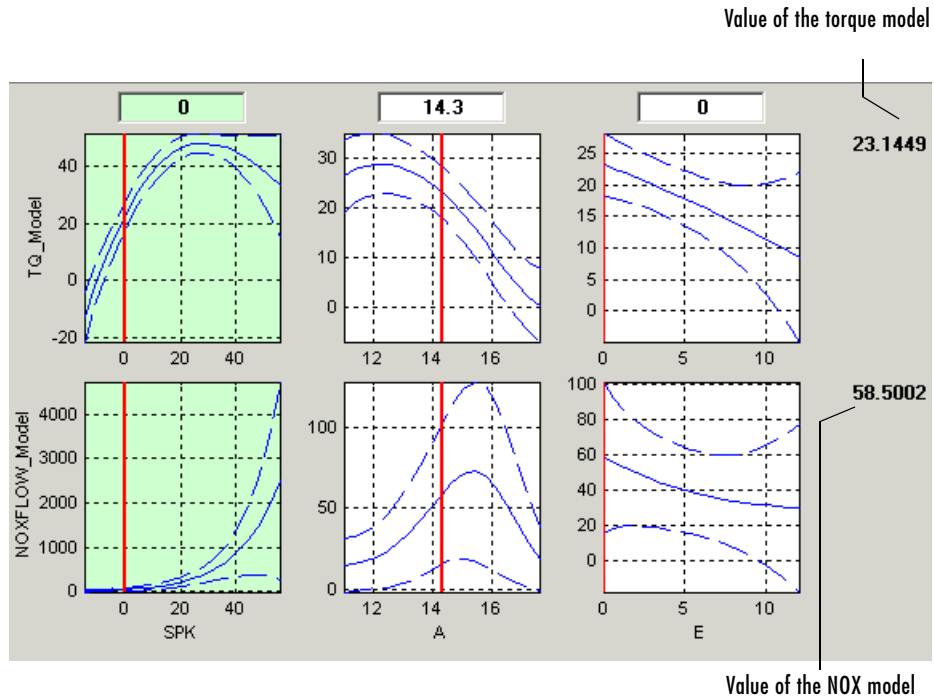
To fill an operating point,

- 1 Select the operating point $N = 4500$ and $L = 0.5$ in the lookup table.
- 2 Find the spark angle that gives the maximum torque and restricts NOX emissions to below 250 g/hr. There are detailed instructions on how to do this tradeoff following.

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.

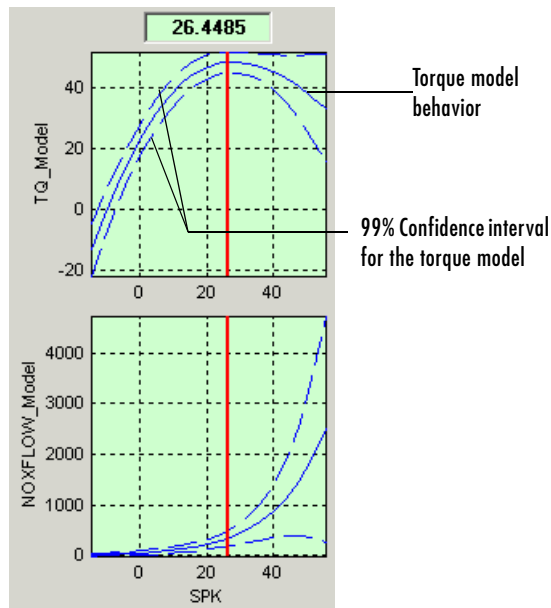
Graphs of the Models When $N = 4500$ and $L = 0.5$



The top three graphs show how the torque model varies with the spark angle, the AFR (labeled A), and the EGR (E), respectively. The lower panes show how the NOX emissions model varies with these variables, respectively.

Looking at the Spark table, the two spark (SPK) graphs are green, indicating that these graphs are directly linked to the lookup table.

You can change the value of spark by dragging the red line on the SPK graphs or typing values into the edit box. The following graph shows the behavior of the two models when the spark angle is 26.4458.




- 3** Select **Find Global Maximum** from the right-click menu of the **SPK - TQ_Model** graph. This calculates the value of spark that gives the maximum value of torque.

At this operating point, the maximum torque that is generated is 48.136 when the spark angle is 26.4485. However, the value of NOX is 347.7861, which is greater than the restriction of 250 g/hr. Clearly you have to look at another value of spark angle.


- 4** Enter 21.5 as the value of **SPK** in the edit box at the top of the **SPK** column.

The value of the NOX emissions model is now 249.1542. 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.

- 5 Press **Ctrl+T** or click  (Add Point) 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 pink when selected, yellow if you click elsewhere.

- 6 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  (Add Point) 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	8.3
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

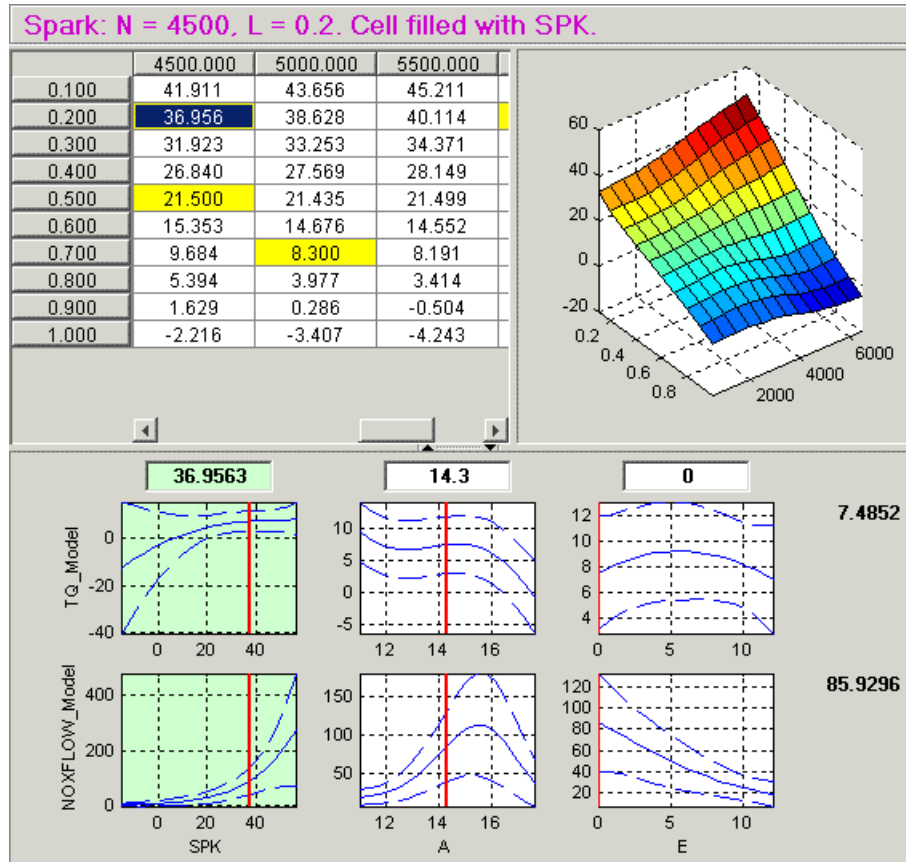
When you have calibrated several key operating points, you can produce a smooth extrapolation of these values across the whole table.

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  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.

The calibrator could now take these techniques to further improve the calibration. That is not the purpose of this tutorial.

For a more detailed description of tradeoff calibrations, see “Tradeoff Calibrations” on page 10-1.

You can now export this calibration for the electronic control unit.

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 type of file you want to save your calibrations as. You can choose from
 - **Comma Separated Value (.csv)**
 - **MAT-file (.mat)**
 - **M-file script**
- 4** For the purposes of this tutorial, select **Comma Separated Value (.csv)**.
- 5** Enter `tradeoff.csv` as the file name and click **Save**.

This exports the spark angle table and the normalizers, Speed and Load, ready for the ECU.

You have now completed the tradeoff calibration tutorial.

Tutorial: Data Sets

This section includes the following topics:

Setting Up the Data Set (p. 4-2)

You can use the **Data Sets** view in CAGE to compare features, tables, and models with experimental data. This tutorial takes you through the basic steps required to compare a completed feature calibration to a set of experimental data. This section covers how to set up a new data set, open an existing calibration, import experimental data into a data set and add data set items.

Comparing the Items in a Data Set
(p. 4-7)

How to use the views to investigate data sets, viewing as tables or plots, displaying errors and using color in the display.

Reassigning Variables (p. 4-14)

How to alter the data set by changing which variables are used for project expressions. You can also use data sets to fill lookup tables from experimental data. For information, see “Tutorial: Filling Tables from Data” on page 5-1.

Setting Up the Data Set

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.

Note If you have a CAGE session open, select **File -> New -> Project**.

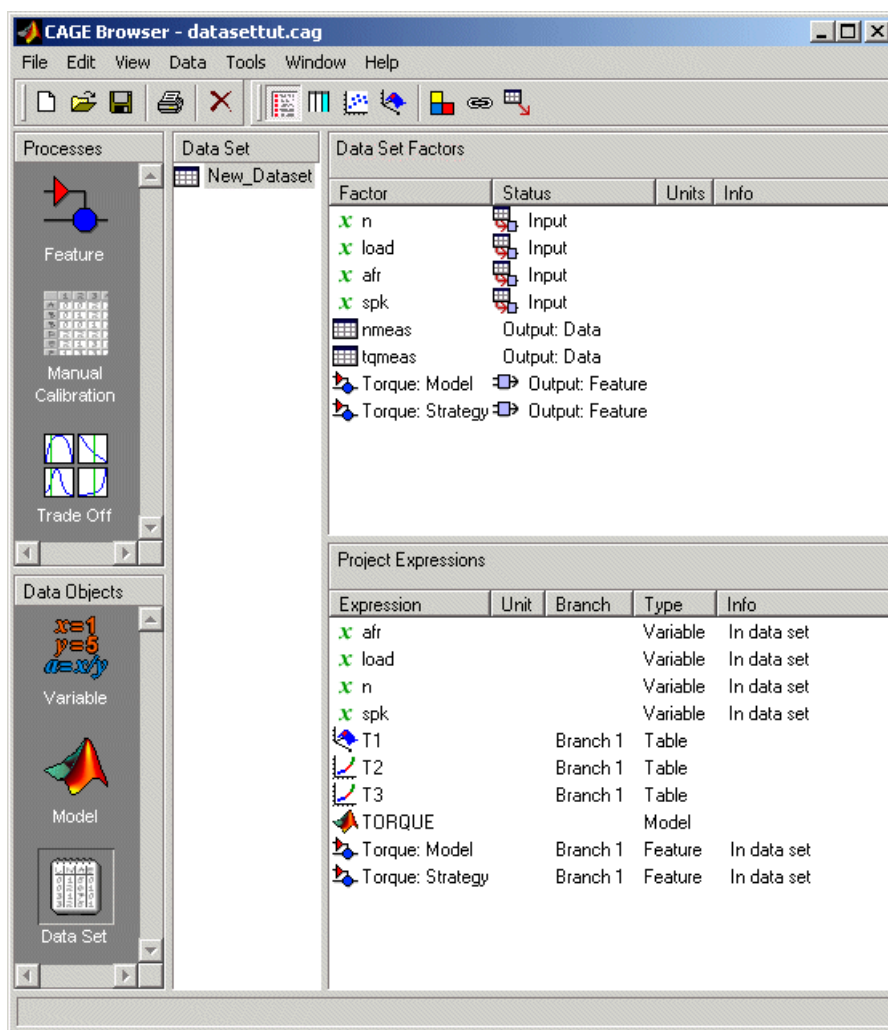
To set up the data set,

- 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.

When these steps are complete, the list of factors includes four input factors and four output factors, as shown.



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 Calibrations” on page 9-1.

- 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**.
- 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`).

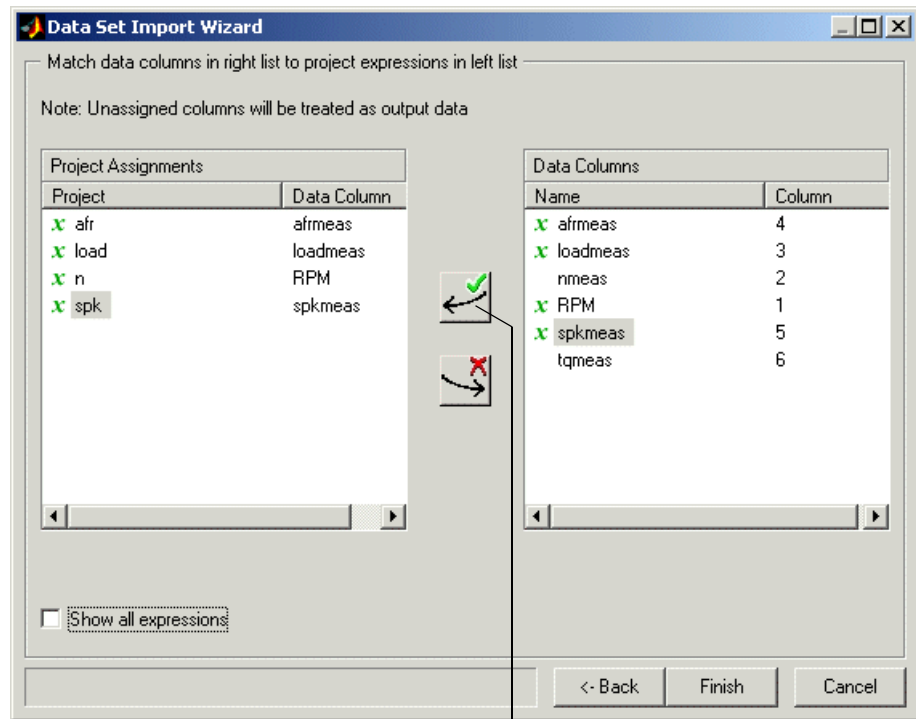
- 3 The **Data Set Import Wizard** asks which of the columns of data you would like to import. Click **Next** to import them all.

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  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.

Comparing the Items in a Data Set

By viewing the data set, you can compare experimental data with calibrations or models in your project.

Viewing the Data Set as a Table

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	2235.000	0.549	9.500	0.100	2247.000	66.700	71.666	66.079
2	3591.000	0.454	13.200	0.100	3613.000	54.100	47.163	46.891
3	4946.000	0.651	12.000	0.100	4974.000	73.700	47.573	79.256
4	881.000	0.648	11.900	5.700	881.000	75.800	99.230	80.211
5	2234.000	0.441	13.300	0.100	2247.000	55.900	51.256	45.152
6	3591.000	0.747	10.900	0.100	3612.000	90.000	92.837	105.586
7	4947.000	0.541	9.700	0.100	4973.000	62.800	57.760	57.587
8	881.000	0.622	9.900	0.100	884.000	72.100	76.198	60.926
9	1219.000	0.333	14.000	0.100	1224.000	41.800	33.226	21.318
10	1558.000	0.382	12.000	0.100	1567.000	49.400	40.487	31.957
11	1896.000	0.209	10.700	3.300	1906.000	28.500	3.492	4.197
12	2234.000	0.284	9.800	3.200	2245.000	36.000	23.063	19.891
13	2574.000	0.407	13.400	3.000	2588.000	49.900	49.629	44.794
14	2914.000	0.595	11.500	3.100	2929.000	70.500	84.680	82.229
15	3251.000	0.781	12.300	3.100	3268.000	90.500	117.424	117.259
16	3589.000	0.668	13.500	3.000	3608.000	77.100	87.987	96.408
17	3930.000	0.452	11.900	3.100	3952.000	52.700	46.511	51.722
18	4268.000	0.235	10.900	3.000	4293.000	27.700	5.253	3.085
19	4606.000	0.194	12.000	3.200	4633.000	21.300	-2.088	-5.771
20	4945.000	0.550	10.100	3.100	4972.000	61.600	55.754	65.743
21	5284.000	0.727	12.600	5.000	5310.000	79.900	78.438	102.468
22	5621.000	0.763	11.300	3.000	5649.000	83.900	75.659	102.957
23	5961.000	0.500	13.200	3.000	5995.000	51.700	35.034	51.304
24	882.000	0.557	12.100	4.400	887.000	64.100	82.455	62.160

In the table, the input cells are white and the output cells are blue.

In addition to viewing the difference between 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.

	n	l...	afr	spk	nmeas	tqmeas	Torque: Model	Torque: Strategy	tqmeas_minus_Torque
1	2...	...	9...	0.1...	2247.000	66.700	71.666	66.079	-0.621
2	3...	...	1...	0.1...	3613.000	54.100	47.163	46.891	-7.209
3	4...	...	1...	0.1...	4974.000	73.700	47.573	79.256	5.556
4	3...	...	1...	5.7...	881.000	75.800	99.230	80.211	4.411
5	2...	...	1...	0.1...	2247.000	55.900	51.256	45.152	-10.748
6	3...	...	1...	0.1...	3612.000	90.000	92.837	105.586	15.586
7	4...	...	9...	0.1...	4973.000	62.800	57.760	57.587	-5.213
8	3...	...	9...	0.1...	884.000	72.100	76.198	60.926	-11.174
9	1...	...	1...	0.1...	1224.000	41.800	33.226	21.318	-20.482
10	1...	...	1...	0.1...	1567.000	49.400	40.487	31.957	-17.443
11	1...	...	1...	3.3...	1906.000	28.500	3.492	4.197	-24.303
12	2...	...	9...	3.2...	2245.000	36.000	23.063	19.891	-16.109
13	2...	...	1...	3.0...	2588.000	49.900	49.629	44.794	-5.106
14	2...	...	1...	3.1...	2929.000	70.500	84.680	82.229	11.729
15	3...	...	1...	3.1...	3268.000	90.500	117.424	117.259	26.759
16	3...	...	1...	3.0...	3608.000	77.100	87.987	96.408	19.308
17	3...	...	1...	3.1...	3952.000	52.700	46.511	51.722	-0.978
18	4...	...	1...	3.0...	4293.000	27.700	5.253	3.085	-24.615
19	4...	...	1...	3.2...	4633.000	21.300	-2.088	-5.771	-27.071
20	4...	...	1...	3.1...	4972.000	61.600	55.754	65.743	4.143
21	5...	...	1...	5.0...	5310.000	79.900	78.438	102.468	22.568
22	5...	...	1...	3.0...	5649.000	83.900	75.659	102.957	19.057
23	5...	...	1...	3.0...	5995.000	51.700	35.034	51.304	-0.396

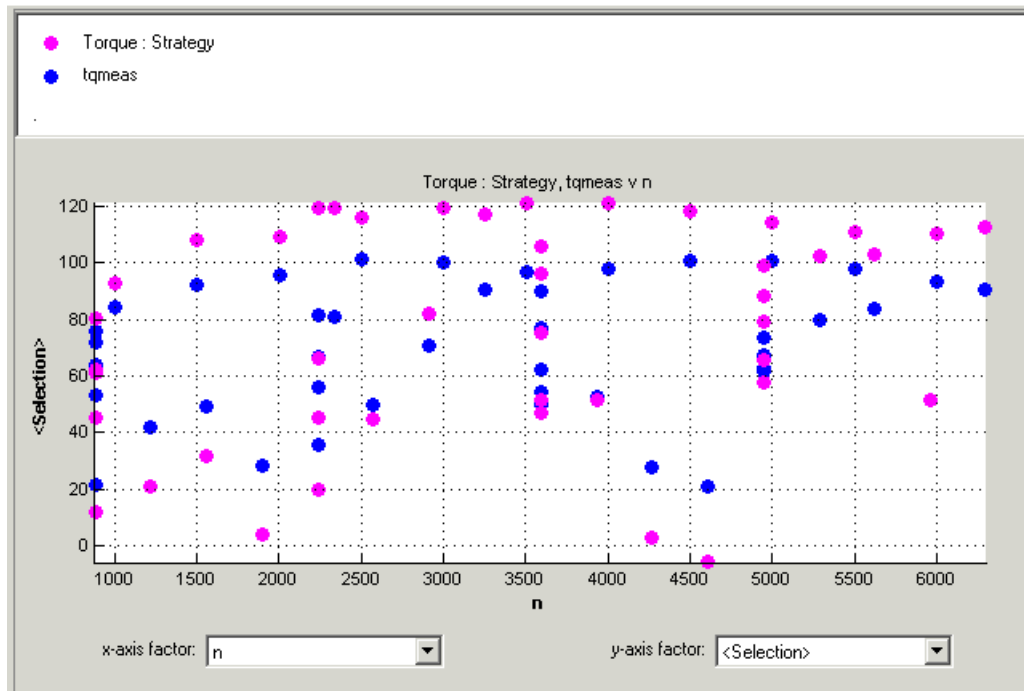
The error column is simply the difference between tqmeas and Torque: Strategy. This provides a simple way of comparing the feature and the measured data.

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.



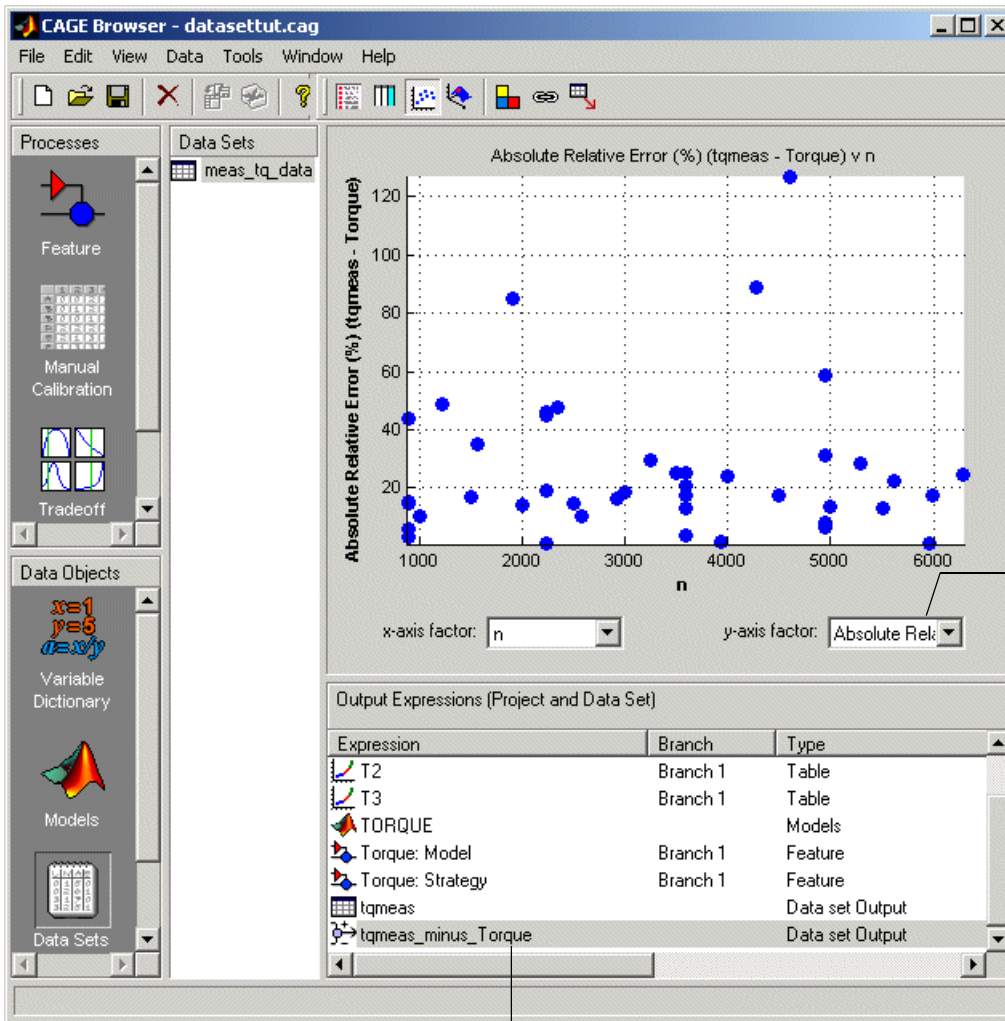
3 Change the **x-axis factor** to n from the drop-down menu.

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.



2. Select **Absolute Relative Error (tqmeas - Torque)**.

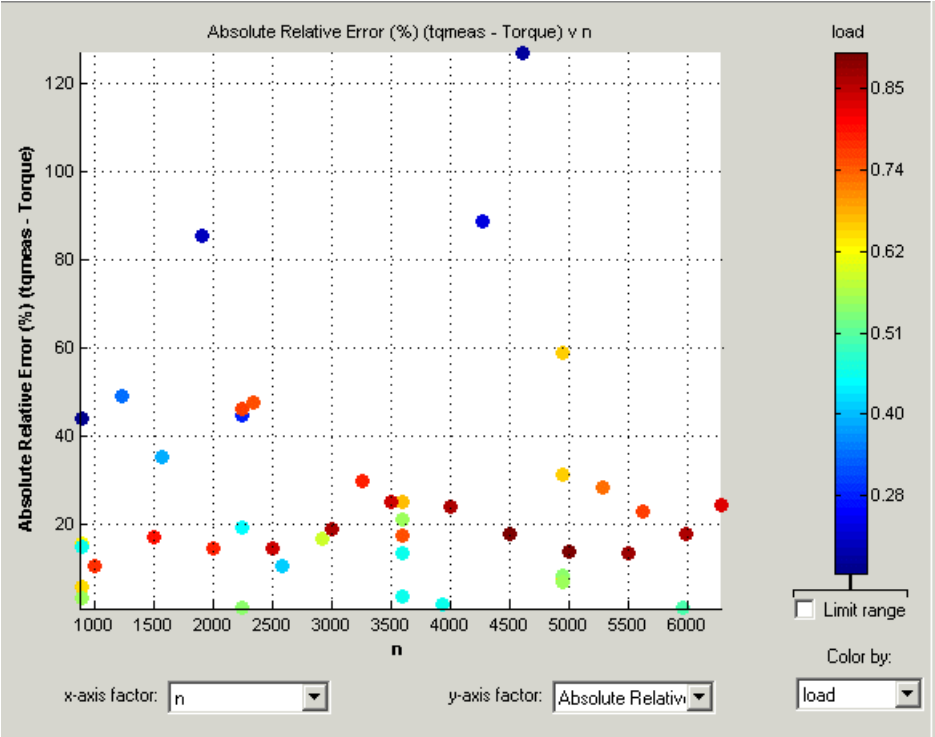
1. Select **tqmeas_minus_Torque**.

- 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.

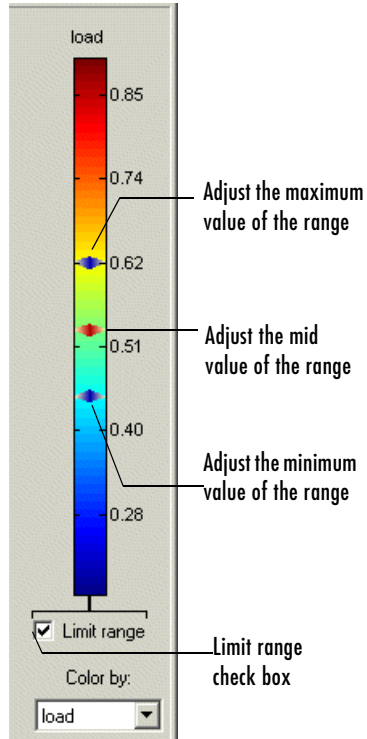
Coloring the Display

- 1 Select **Color by Value** from the right-click menu on the graph.
- 2 From the **color by** drop-down menu, select load.



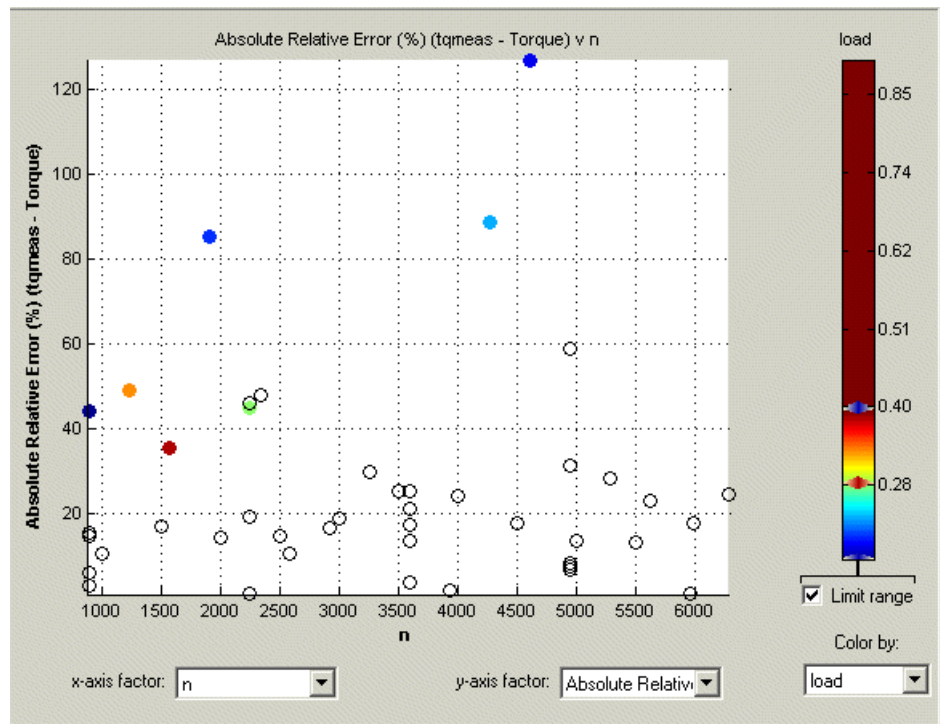
In this display, you can see that some of the low values of load display a high error.

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.
- 2 Right-click the graph and select **Restrict Color to Limits**.
- 3 Set the minimum value of the color range to be as low as possible by dragging the minimum value down.
- 4 Set the maximum value of the color range to be around 0.4.




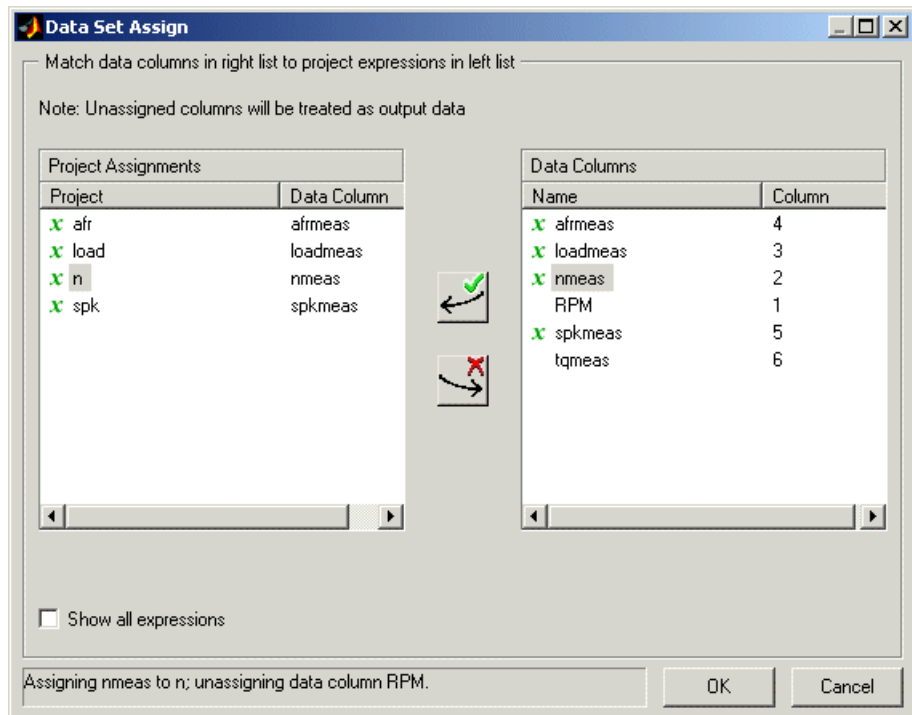
As the low values of load are causing large errors, it would be wise to reexamine the calibration, particularly at small values of load.

Reassigning Variables

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 complete functionality of data sets, see “Data Sets” on page 12-1.

You have now completed the data sets tutorial.

Tutorial: Filling Tables from Data

This section includes the following topics:

- | | |
|--|--|
| Setting Up a Table and Experimental Data (p. 5-2) | How to set up a new table and import experimental data. |
| Filling the Table from the Experimental Data (p. 5-10) | How to fill a table from experimental data. |
| Selecting Regions of the Data (p. 5-14) | How to use regions to include only the data you want to use. |
| Exporting the Calibration (p. 5-16) | How to export your calibration. |

Setting Up a Table and Experimental Data

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.

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 5-10 describes the process of filling the lookup table.
- “Selecting Regions of the Data” on page 5-14 describes how you can select some of the data for inclusion when you fill the table.
- “Exporting the Calibration” on page 5-16 describes how to export your completed calibration.

1 Start CAGE by typing

`cage`

at the MATLAB prompt.

2 Select **File** -> **New Project**.

This section describes how to set up a blank table ready for filling using experimental data.

The steps that you need to follow to set up the CAGE session are

- 1** Add the variables for speed and load.
- 2** Add a new table to your session.

- 3 Set up your normalizers.
- 4 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,

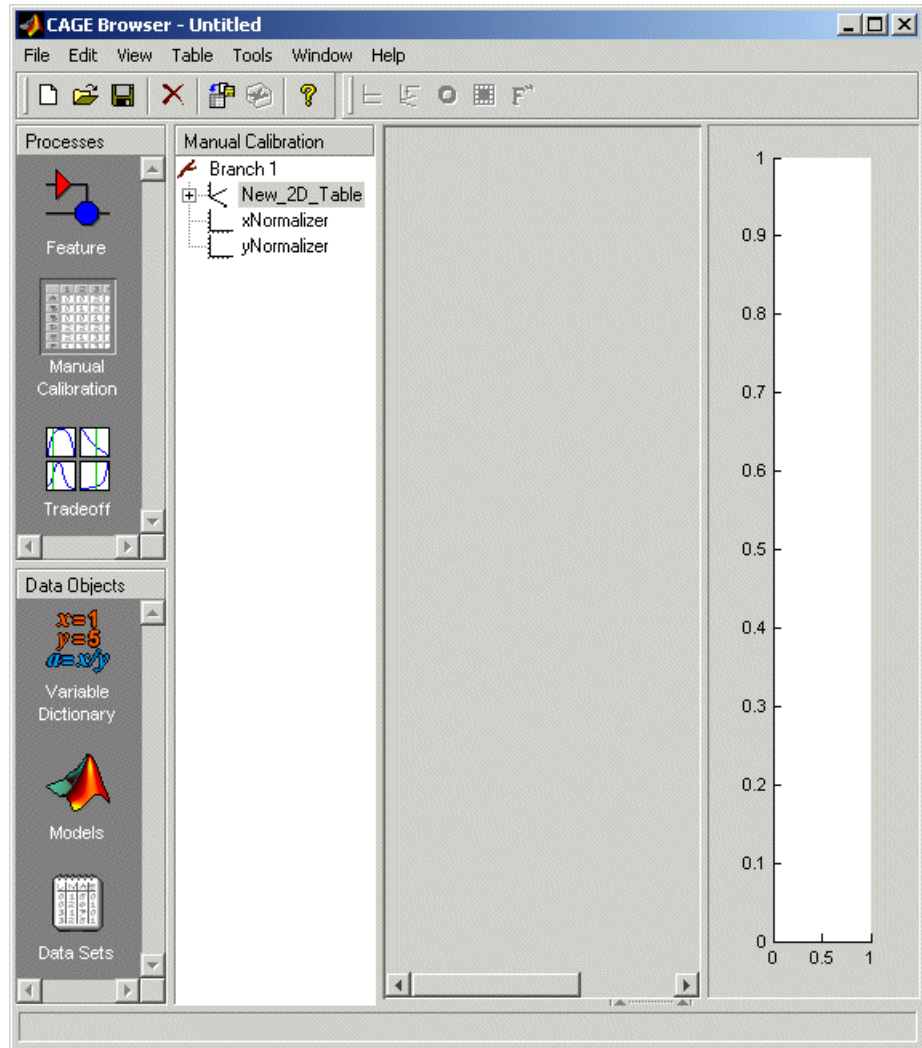
- 1 Select **File -> New -> 2D table**.

This opens a dialog box that asks you to specify the variable names for the normalizers.

- 2 To accept N as the variable for normalizer **X** and L as the variable for normalizer **Y**, click **OK**.

Note In CAGE, a 2-D table is defined as a table with two inputs.

CAGE takes you to the **Manual Calibration** view, where you can see the following.



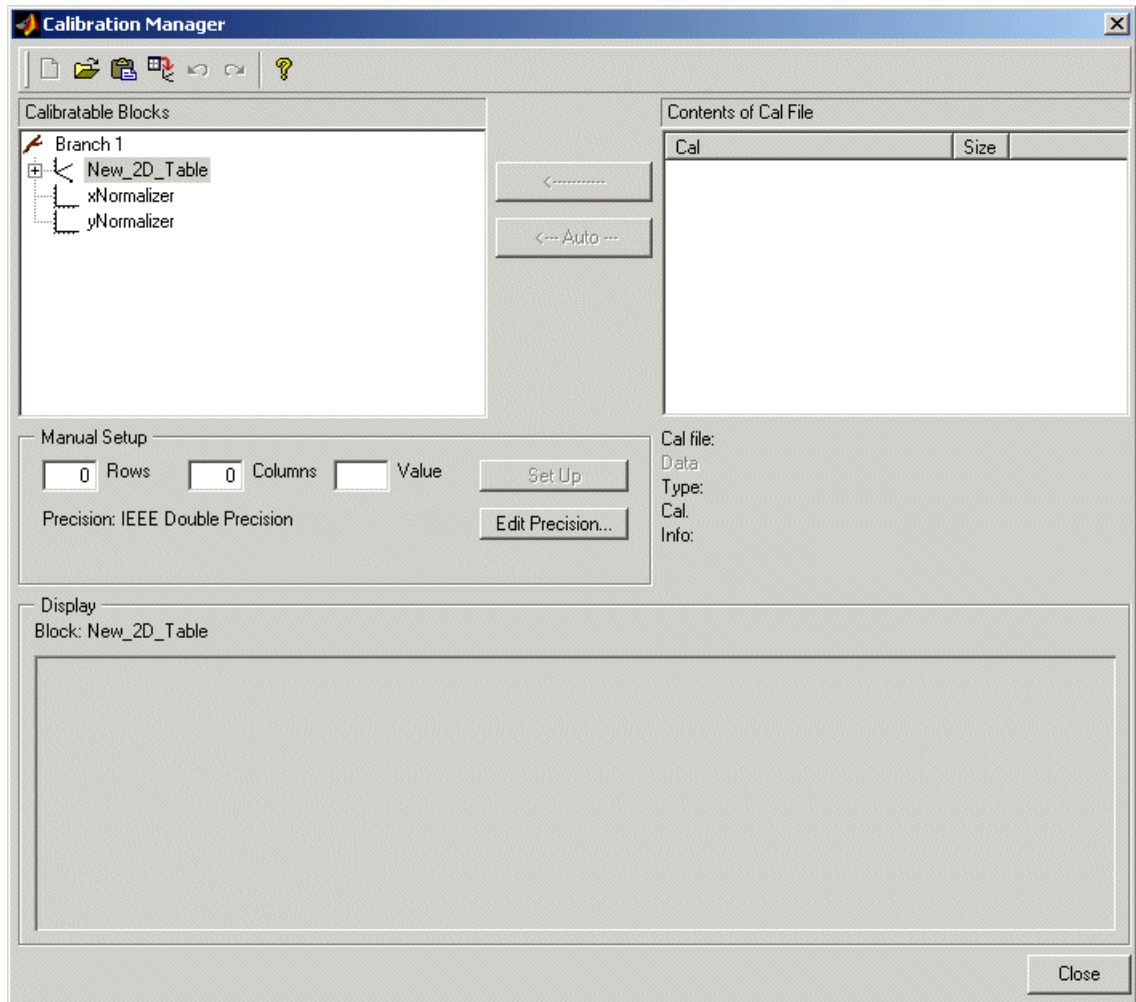
You must now set up and initialize the normalizers with suitable values for the engine speed and load.

Setting Up the Normalizers

Currently, the lookup table has neither rows nor columns, so you must set up the table.

To set up the table,

- 1 Open the **Calibration Manager** dialog box, shown, by clicking .




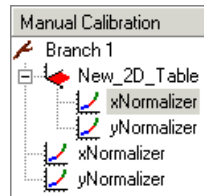
- 2 Highlight the **New_2D_Table** in the **Calibratable Blocks** display.
- 3 To determine the size of each normalizer, enter 10 as the number of rows and 7 as the number of columns.
- 4 Set the value in each cell to be 0.
- 5 Click **Set Up** to confirm your selection.

- 6 Click **Close** to close the **Calibration Manager** and return to the table view.


Setting the Values of the Normalizers

You can now set values of the normalizers so that they cover the range of the variables N and L:

- 1 Expand the table branch by clicking , and select xNormalizer as shown.



This displays the two normalizers for the table.

- 2 To space the breakpoints evenly over the range of the variables N and L, click .

This opens a dialog box that suggests the range of values for each normalizer.

- 3 To accept the suggested ranges of N and L, click **OK**. These suggested ranges are determined by the variable dictionary.

You now 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.

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.

This set of data includes six columns of data: the test cell settings for engine speed (RPM), and the measured values of torque (`tqmeas`), engine speed (`nmeas`), air/fuel ratio (`afrmeas`), spark angle (`spkmeas`), and load (`loadmeas`).

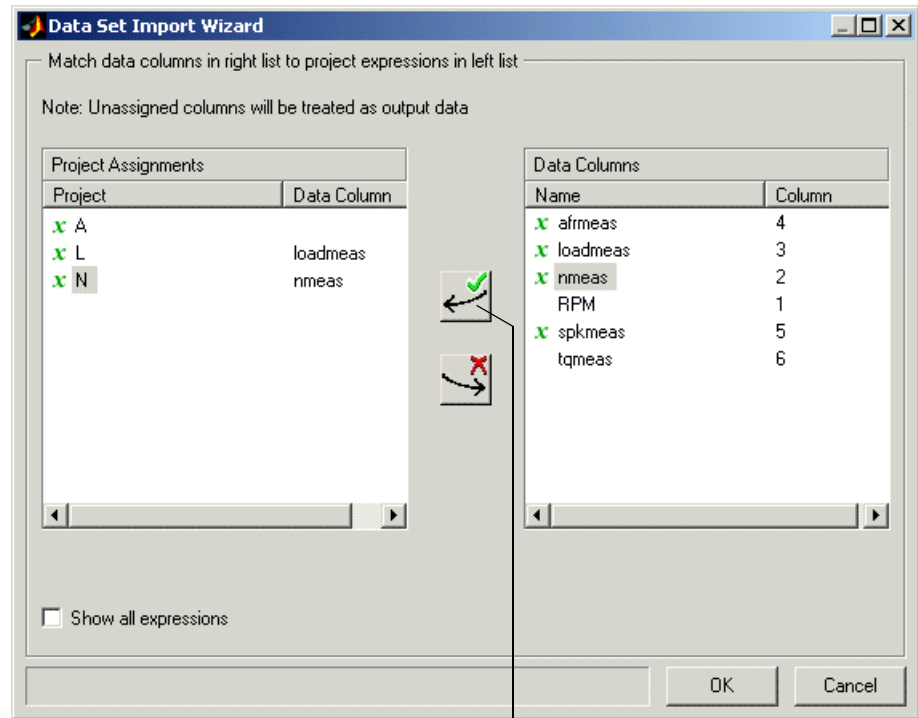
- 3 This opens the **Data Set Import Wizard**. The first screen asks which of the columns of data you want to import. Click **Next** to import them all.

The following screen asks you to associate variables in your project with data columns in the data.

- 4 Highlight `N` in the **Project Assignments** column and `nmeas` in the **Data Column**, then click the assign button, shown.



- 5 Repeat this to associate `L` with `loadmeas`. The dialog box should be the same as the following.



Assign button

6 Click **Finish** to close the dialog box.

You now have an empty table and some experimental data in your session. You are ready to fill the table with values based on this data.


Filling the Table from the Experimental Data

You have an empty table and the experimental data in your session. You can now fill the table with values based on your data.

The data that you have imported is a series of measured values of torque at a selection of different operating points. These operating points do not correspond to the values of the breakpoints that you have specified. The lookup table has a range of engine speed from 500 revolutions per minute (rpm) to 3500 rpm. The range of the experimental data is far greater.

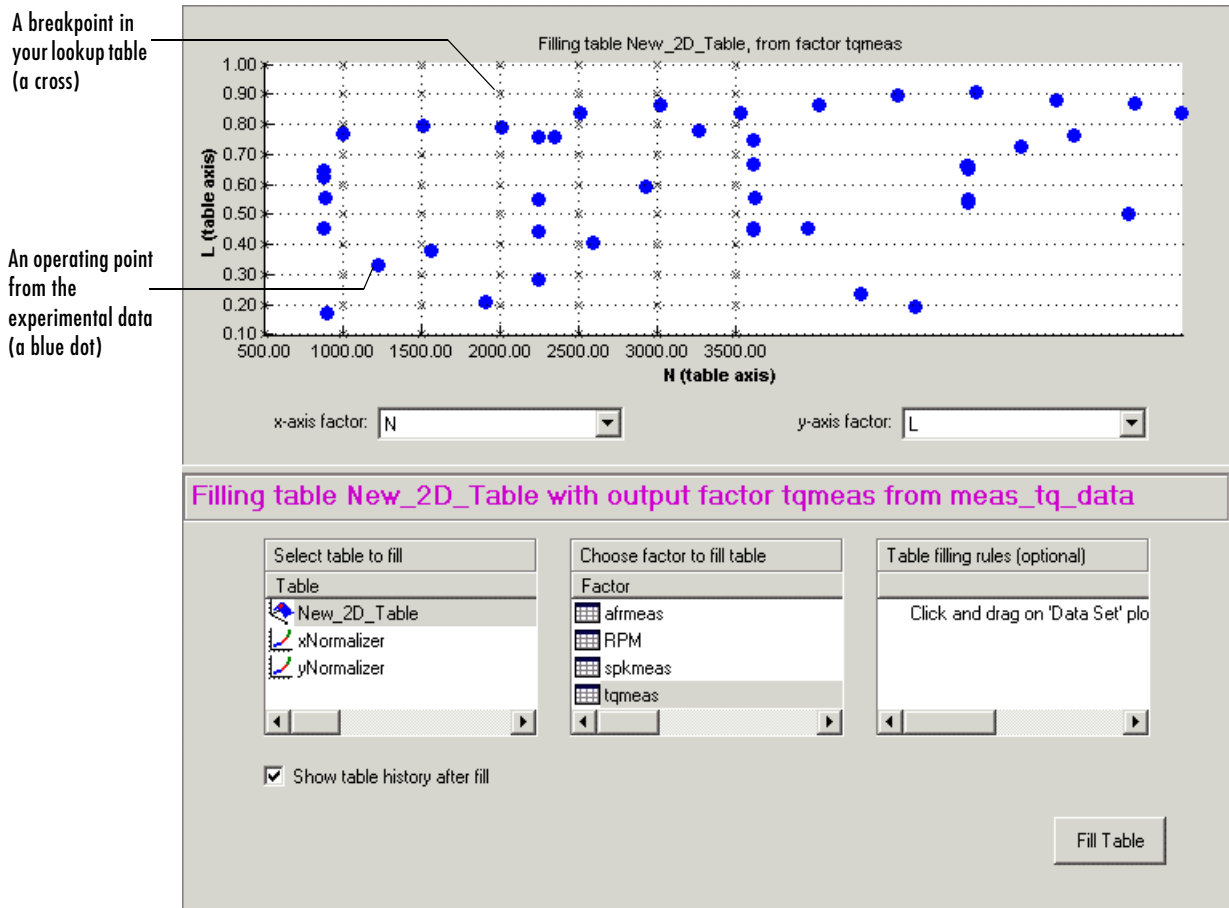
CAGE extrapolates the values of the experimental data over the range of your table. Then it fills the table by selecting the torque values of the extrapolation at your breakpoints.

To fill the table with values based on the experimental data,

- 1** To view the **Table Filler** display, click  in the toolbar in the **Data Sets** view.

This display asks you 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** list. This is the table that you want to fill.
- 3** Select `tqmeas` from the **Factor** 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.



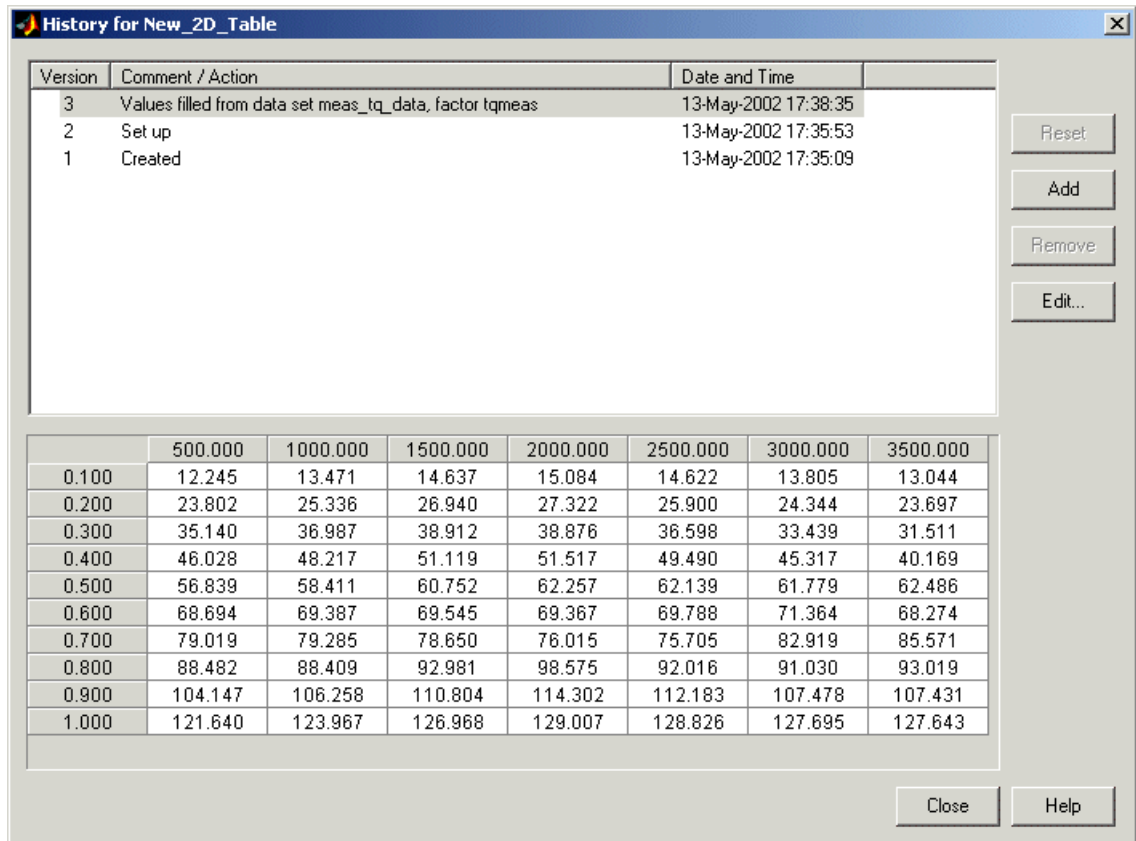
The upper pane displays the breakpoints of your table as crosses and the operating points where there is data as blue dots.

- To view the table after it is filled, ensure that the **Show table history after fill** box, at the bottom left, is selected.

Data sets display the points in the experimental data, not the values at the breakpoints.

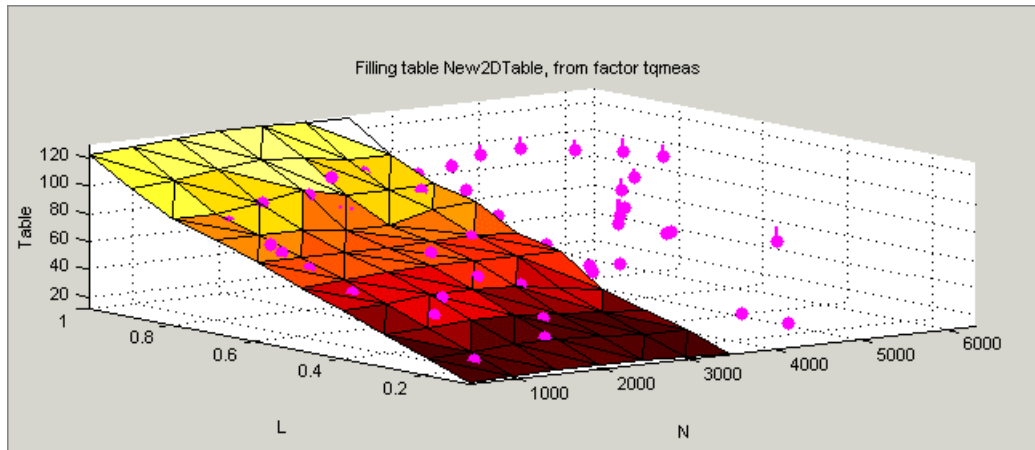
- 6 To fill the table with values of tqmeas extrapolated over the range of the normalizers, click **Fill Table**.

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.

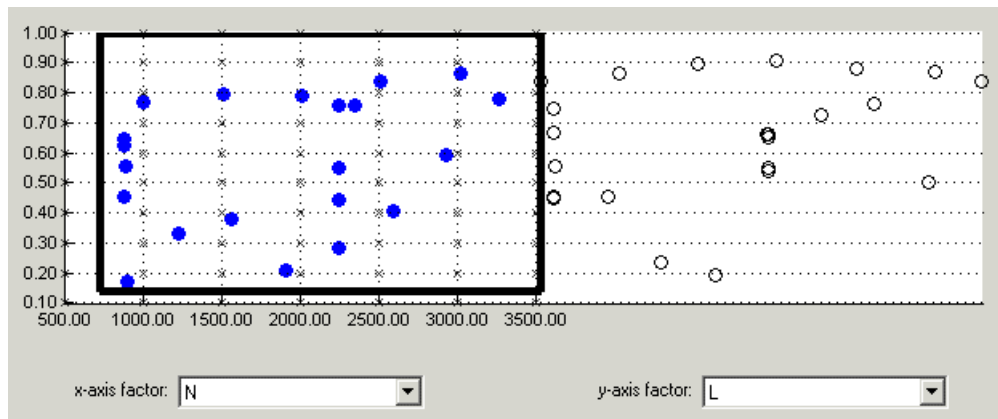
Selecting Regions of the Data

You can ignore points in the data set when you fill your lookup table.

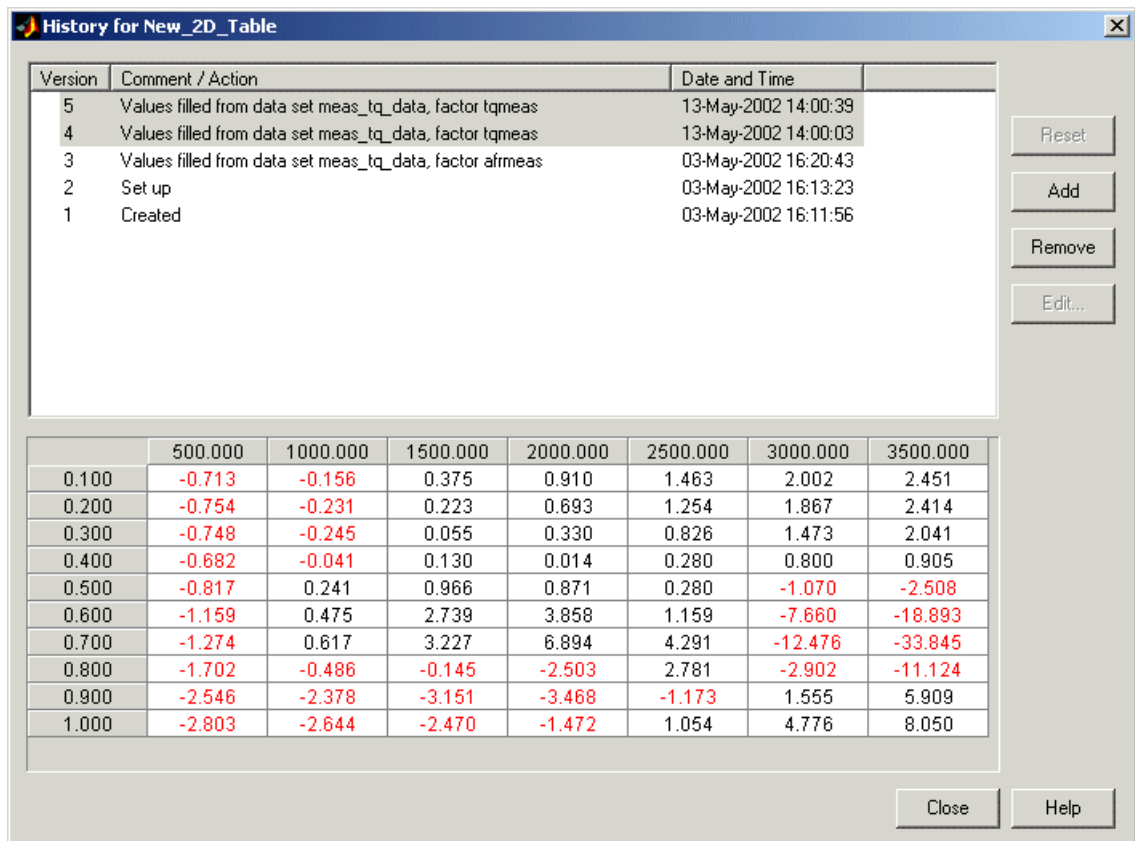
For example, in this tutorial the experimental data ranges over values that are not included in the lookup table. You want to ignore the values of engine speed that are greater than the range of the table.

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 3 and 4, 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

You can export the calibration for use in an electronic control unit (ECU).

To export the calibration,

- 1 To highlight the table that you want to export, you must first click **Manual Calibration**, shown.



- 2 Highlight the `New_2D_Table`.
- 3 Select **File -> Export -> Calibration**.
- 4 Choose the type of file you want to save your calibrations as. You can choose from
 - a **Comma Separated Value (.csv)**
 - b **MAT-file (.mat)**
 - c **M-file script**
- 5 For the purposes of this tutorial, select **Comma Separated Value (.csv)**.
- 6 Enter `table_filling_tutorial.csv` as the file name and click **Save**.

This exports the successful calibration, ready for the ECU.

You have now completed this tutorial.

Tutorial: Optimization and Automated Tradeoff

This section includes the following topics:

Getting Started (p. 6-2)	Introducing the tutorial optimizations and how to set up your session by importing models and setting up an operating point set.
Single-Objective Optimization (p. 6-6)	How to set up and run a simple single-objective optimization, examine and export your results, and use those results to fill a table.
Multiobjective Optimization (p. 6-17)	How to set up and run a multiobjective optimization, and use the output views to select the best solutions to export.
Sum Optimization (p. 6-32)	How to set up a sum optimization so you can assign weights to more significant operating points, for example to constrain emissions over a whole drive cycle.
Automated Tradeoff (p. 6-35)	How to use your optimizations for automated tradeoff.
Worked Example Optimization (p. 6-38)	How to use the Worked Example. We provide this to illustrate modifying the template to create your own user-defined optimizations.
Creating an Optimization from Your Own Algorithm (p. 6-46)	A detailed walk-through of the steps involved in incorporating your own algorithm into an optimization function for use in CAGE.

Getting Started

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. See “Single-Objective Optimization” on page 6-6.
- A multiobjective optimization to maximize torque and minimize NOX emissions. See “Multiobjective Optimization” on page 6-17.
- A sum optimization to maximize torque while minimizing NOX, weighted to give more importance to idle speed. See “Sum Optimization” on page 6-32.
- Using any of your optimizations to run an automated tradeoff. Once you have set up an optimization you can apply it to a tradeoff. See “Automated Tradeoff” on page 6-35.

The Optimization View

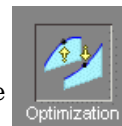
You can use the **Optimization** view to set up, run, view, and export optimizations. You must also set up optimizations here in order to use them for automated tradeoff.

Start the CAGE Browser part of the Model-Based Calibration Toolbox by typing

```
cage
```

at the MATLAB prompt.

To reach the **Optimization** view, click the



button in the **Processes**

Here you can set up and view optimizations. As with other CAGE processes, 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. When you first open the **Optimization** view both panes are blank until you create an optimization.

As for other CAGE processes, you must set up your session for an optimization. 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. In this case you also need to set up this set of points using the **Data Sets** view. The steps required are as follows:

- 1 Import a model or models.
- 2 Define an operating point set if required.
- 3 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.

Setting Up Your Optimization Session

Before you can set up the optimization, you must set up your session.

- 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 “Using CAGE” on page 7-1.

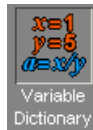
- 2 Select the **Models** view by clicking the button shown in the **Data Objects** pane.



Observe that the project you have opened already contains two models: TQ_Model and NOXFLOW_Model. In this tutorial you use these models to optimize torque values subject to emissions constraints.

The project already has the relevant variables defined, so you do not need to import a variable dictionary.

- 3 To view the items in the **Variable Dictionary**, click the button, shown, in the **Data Objects** pane.



The **Variable Dictionary** view appears, displaying the variables, constants, and formulas in the current project. Note that the variables have ranges and set points defined.

To run your optimization at several points, you need to create an operating point set to use in 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).

- 4 Click the **Data Sets** button, shown, in the **Data Objects** pane.




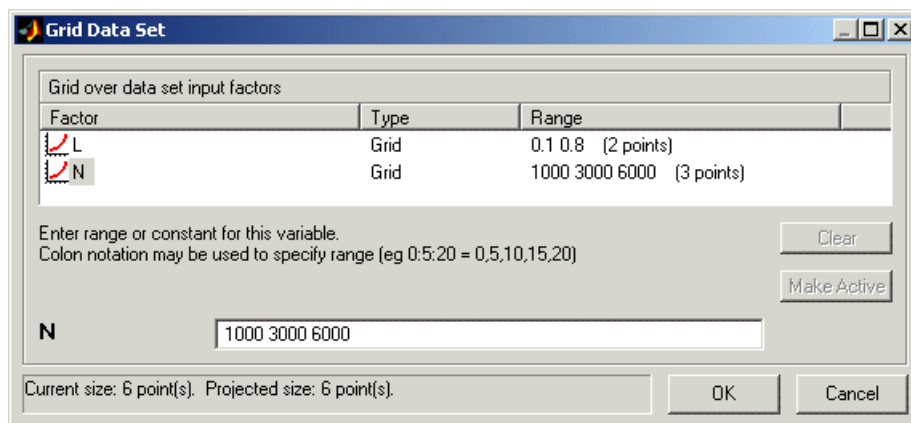
The **Data Sets** view appears.

5 Select **File** -> **New** -> **Data Set**.

CAGE creates an empty data set.

6 To add speed and load to your empty data set, press and hold the **Shift** key while you click **N** and **L** in the lower **Project Expressions** pane to select them both. Then right-click (**N** or **L**) and select **Add to Data Set**.

7 To create a grid of points for your operating point set, click **Build Grid** in the toolbar () or select **Data** -> **Build Grid**. The **Grid Data Set** dialog appears, as shown below.




8 Click **L** in the **Factor** column and enter two values in the edit box, as shown below, then press **Enter**:

0.1 0.8

9 Click **N** and enter three values in the edit box, as shown below, then press **Enter**:

1000 3000 6000

As you can see reported at the bottom of the dialog, this will give you an operating point set containing six points. Click **OK**.

You can click the **View Data** toolbar button () to see these six operating points displayed as a table. Your session is now ready to create a new optimization using this operating point set.

Single-Objective Optimization

The following sections describe these stages:

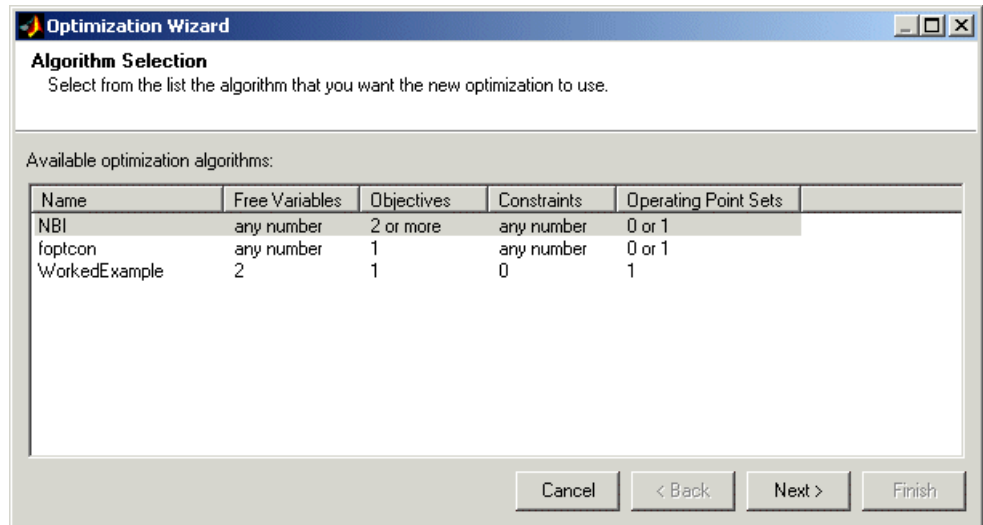
- 1** Using the **Optimization Wizard** to choose
 - Your optimization algorithm
 - How many objectives, constraints, and operating point sets to use
 - What free variables to use
- 2** Using the **Optimization** node to choose
 - A model for your objective
 - A model, type, and value for your constraint
 - An operating point set for your optimization
- 3** Running the optimization, examining the output, exporting to a data set, and using the output to fill a table

Using the Optimization Wizard

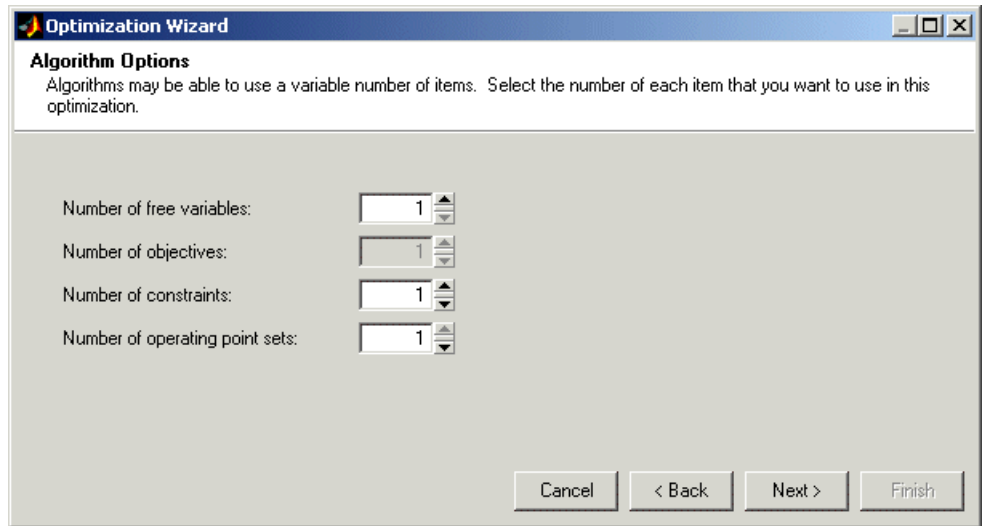
To create a new optimization,

- 1** Select **File -> New -> Optimization**.

This opens the **Optimization Wizard**.

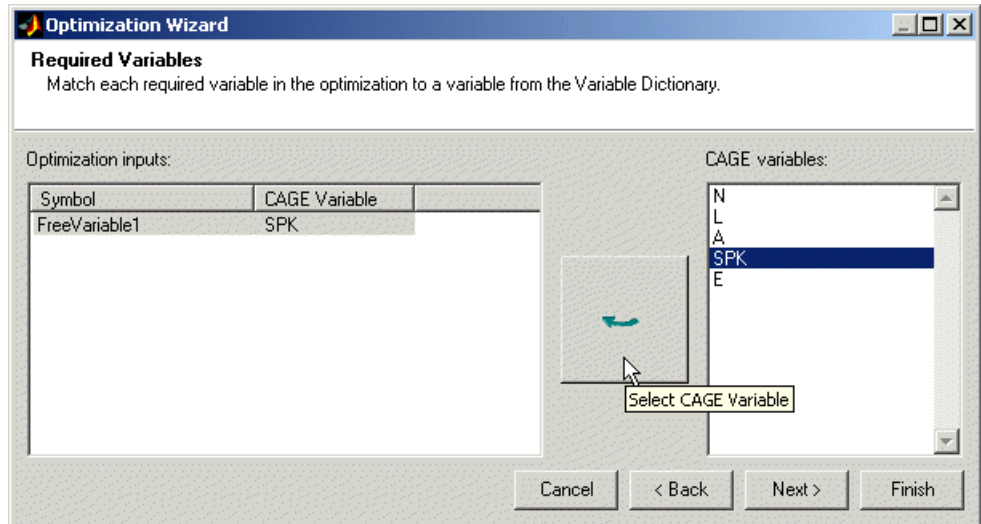


- 2 Click to select `foptcon` in the list. This is the optimization algorithm you will use for this example. Note that this algorithm specifies a single objective in the **Objectives** column. Click **Next**.
- 3 On the next screen, set the number of constraints and the number of operating point sets to 1, as shown.

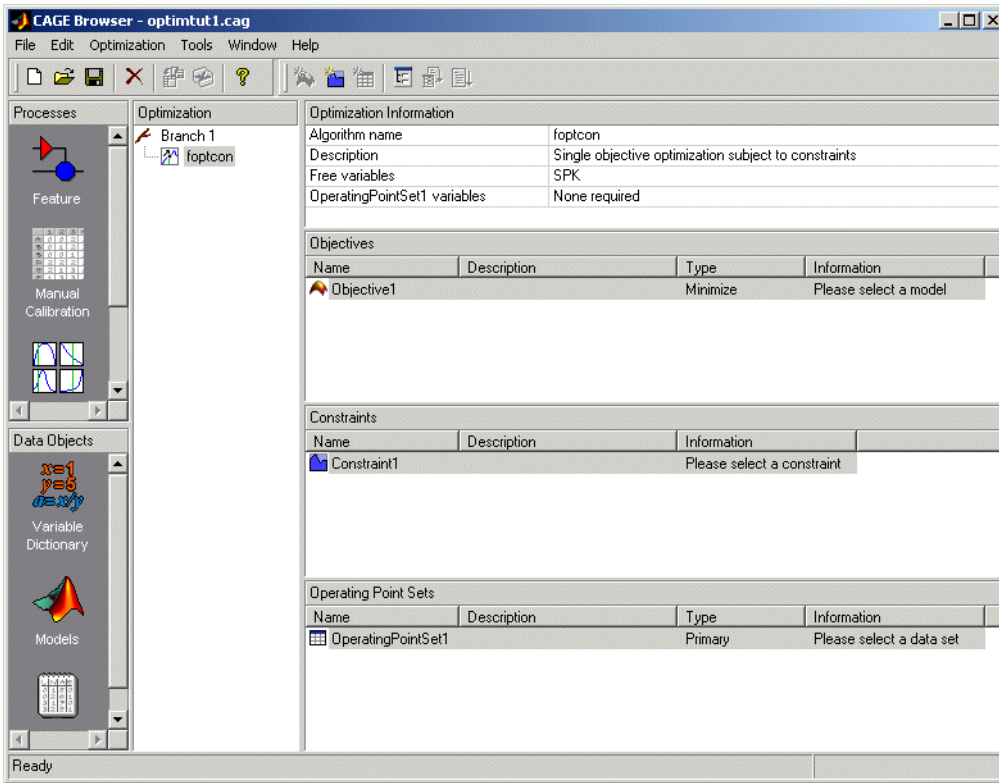


Leave the number of free variables at 1 (spark will be the free variable).
Leave the other controls and click **Next**.

- 4 On the next screen, choose spark as your free variable for this optimization by clicking SPK in the list on the right, then click the button to match it up with FreeVariable1, as shown. Then click **Finish**.



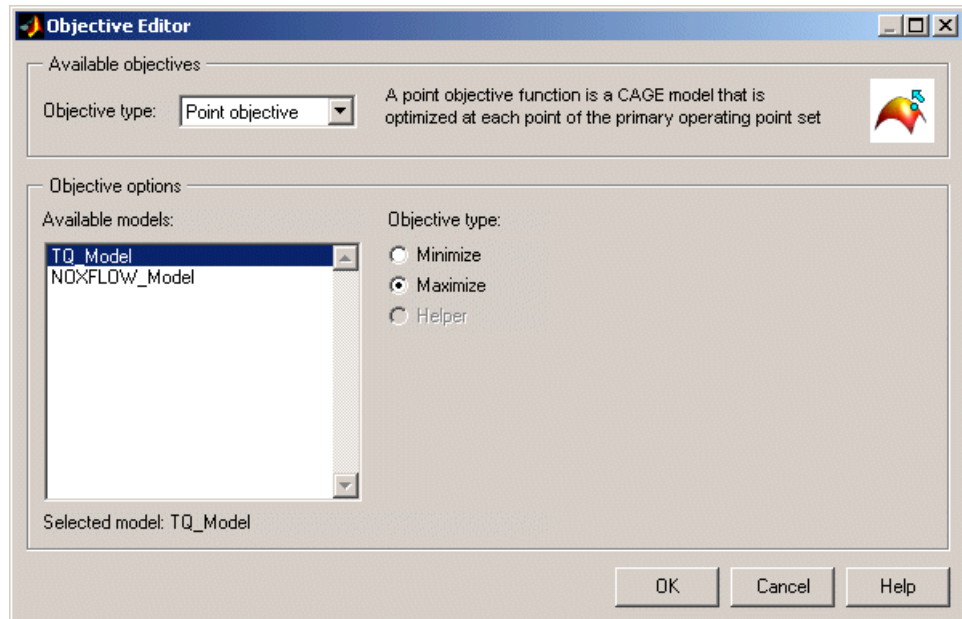
A new branch named foptcon appears in the Optimization tree. Your CAGE browser should look like the following example. In the **Optimization Information** pane you can see listed the algorithm name foptcon, free variable SPK, and the description Single objective optimization subject to constraints. In the three panes **Objectives**, **Constraints**, and **Operating Point Sets** you can see messages informing you that you need to specify a model for an objective, a constraint, and an operating point set.



Setting Objectives, Constraints and Operating Point Sets

- 1 Double-click Objective1 in the Objectives pane.

The **Objective Editor** appears.

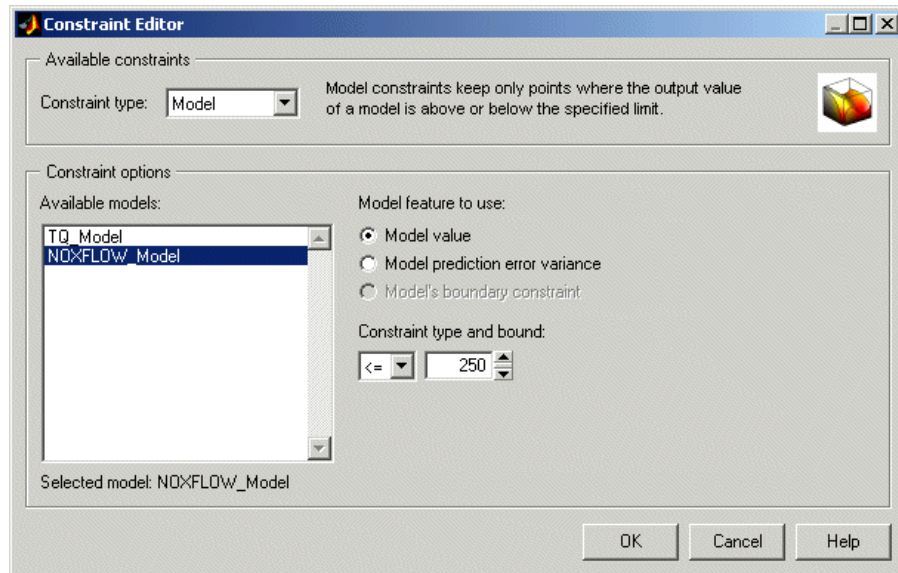


- 2 Click to select `TQ_Model` and select **Maximize** from the radio buttons on the right. Click **OK**.

You return to the CAGE Browser Optimization node. The **Description** `TQ_Model (SPK,L,N,A,E)` appears in the **Objectives** pane.


- 3 Double-click `Constraint1` in the **Constraints** pane.

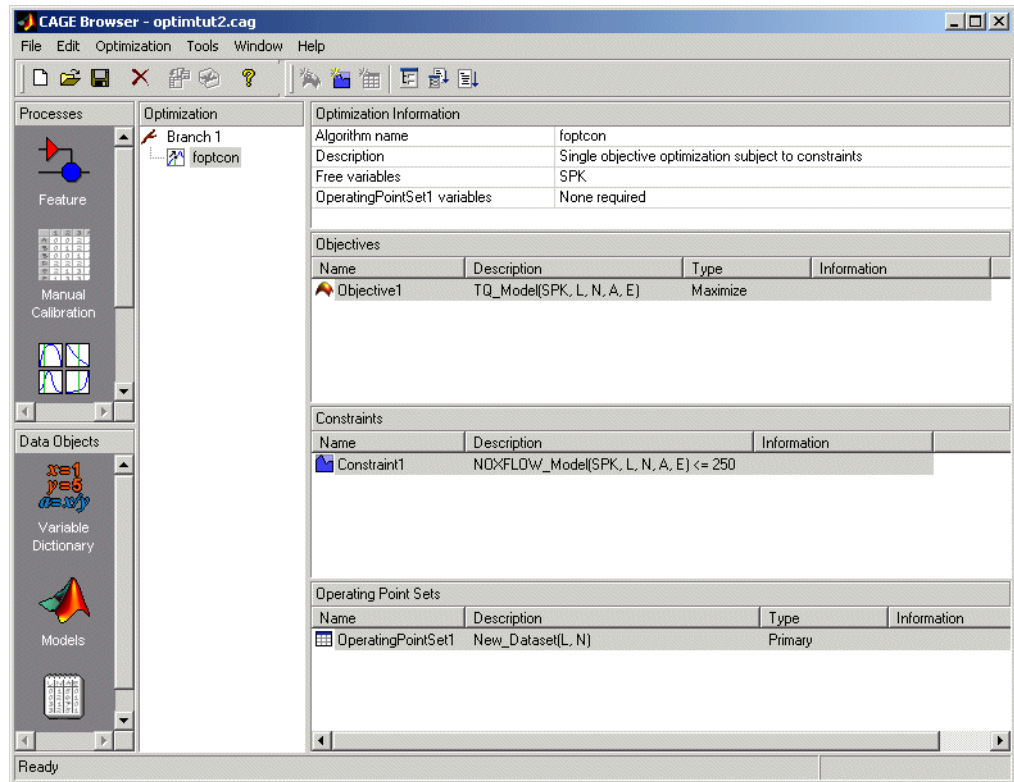
The **Constraints Editor** appears.




- 4 Ensure that Model1 is selected from the **Constraint Type** drop-down menu.
- 5 Select NOXFLOW_Model from the list, and enter 250 in the edit box as the maximum value for the constraint, as shown above. Click **OK**.

You return to the CAGE Browser Optimization node. Notice that the **Description** NOXFLOW_Model (SPK, L, N, A, E) ≤ 250 appears in the **Constraints** pane.

- 6 Double-click OperatingPointSet1. CAGE automatically selects New_Dataset (L,N) because it is the only one in your session that contains appropriate variables.
- 7 Your CAGE Browser should now look like the following example, with an objective, constraint, and operating point set. Note that the toolbar button Run Optimization () is now enabled, because your optimization setup has provided enough information to start an optimization.



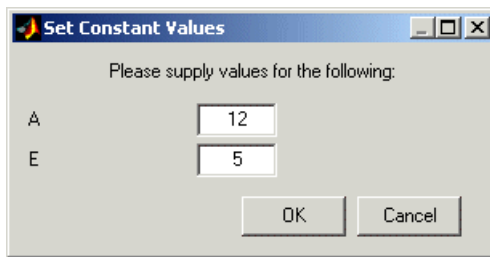
Running the Optimization

- 1 Click Run Optimization () in the toolbar.

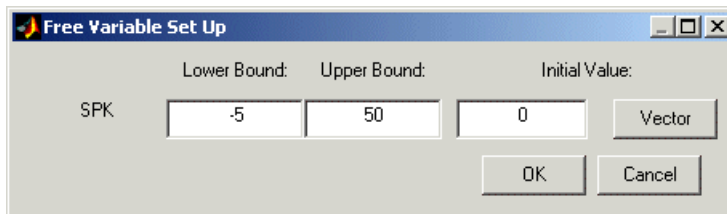
Running the optimization requires the selected models to be evaluated (many times over) and hence values are required for all the model input factors (L, N, A, E, and SPK). Before the optimization can proceed two dialogs will appear because some values are needed.

- Spark (SPK) has been designated a free variable, so the optimization will choose different values for SPK in trying to find the best. You do not need to state a value for SPK but you do need to choose a starting value of SPK and a range.

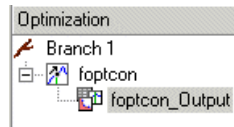
- Your operating point set defines values of L and N where you want the optimization to run. A and E are the remaining variables required by the optimization to evaluate the TQ and NOX models. You need to choose values for these variables that will be used at every operating point.
- 2 A dialog appears called **Set Constant Values**, asking you to supply values for the fixed variables in the model not specified by your operating point set (A and E). The defaults are taken from the set points for these variables in the **Variable Dictionary**. Click **OK** to accept these values.



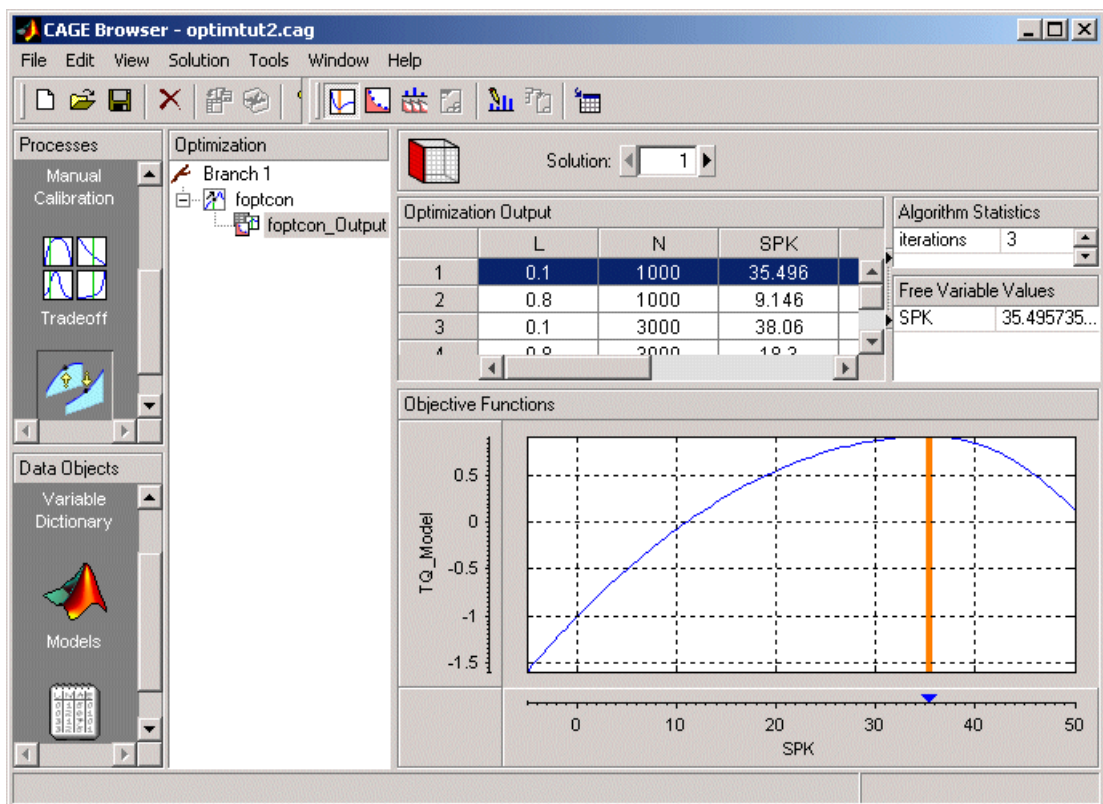
- 3 The **Free Variable Set Up** dialog appears, showing the bounds and initial value of your free variable (SPK). The defaults are taken from the range and set point in the **Variable Dictionary**. Click **OK** to accept these values.




When you click **OK**, the optimization runs, with progress messages as each point is evaluated. On completion of the optimization, a new node appears in the Optimization tree. Click the plus sign next to the foptcon node to expand the tree, then click the new node foptcon_Output to view the optimization results.



This single-objective optimization produces one best solution for each point in the operating point set. Click the cells of the table to view solutions at those operating points.



- 4 Click Export to Data Set () in the toolbar. Go to the **Data Sets** view (click **Data Sets** in the **Processes** pane) to see that the table of optimization results is contained in the new data set Exported_Optimization_Data. You

could use these (or any optimization results) to fill tables, as you can with any data set.

Using Optimization Results to Fill Tables

As an example, to use these optimization results to fill a table,

- 1** Build a SPK table in N and L.
 - a** Select **File -> New-> 2-D Table**.
 - b** Open the **Calibration Manager** to choose the number of rows and columns and set up the table by filling it with zeros. Enter 10 in the **Rows** and **Columns** edit boxes and 0 in the **Value** edit box and click **Set Up**, then click **Close**.
- 2** Your CAGE browser shows the **Manual Calibration** view. Select the **Normalizer** view by clicking one of the table normalizers (N or L) in the tree, then click the Initialize toolbar button to space breakpoints over the ranges of N and L. Click **OK** in the dialog and the breakpoints are spaced. Repeat for the other table normalizer.
- 3** Go to **Data Sets** to fill the new table.
 - a** Select `Exported_Optimization_Data` and click the View Data button in the toolbar to see the table view.
 - b** Right-click the SPK column header and select **Copy Current Values**. You must make a copy of the values of SPK because CAGE will not allow you to use model inputs to fill tables.
 - c** Click Fill Table From a Data Set in the toolbar. Choose to fill the spark table with the `Copy_of_SPK` optimization output by selecting them in the lists, then click **Fill Table**.

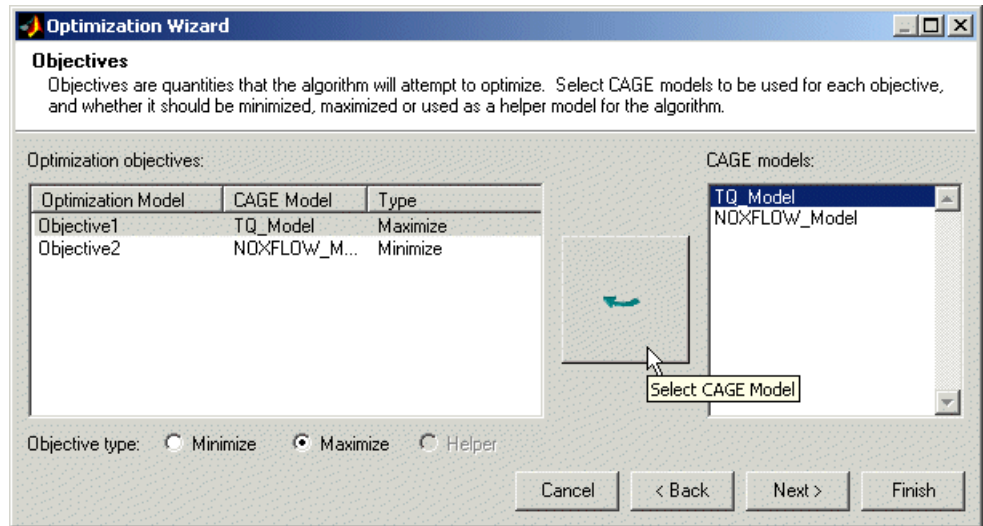
Multiobjective Optimization

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.

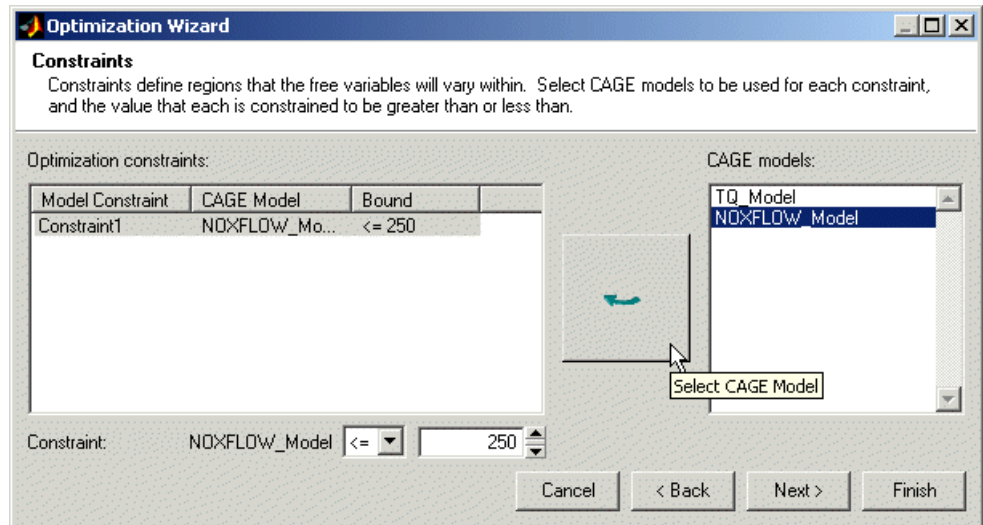
- 1 Select **File** -> **New** -> **Optimization**.

This opens the **Optimization Wizard**.

- 2 Click to select NBI in the list. This is the optimization algorithm you will use for this example. Note that this algorithm specifies two or more objectives in the **Objectives** column. Click **Next**.
- 3 On the next screen, set the number of constraints and the number of operating point sets to 1. Leave the number of free variables at 1 (this will be spark) and objectives at 2 (these will be the TQ and NOXFLOW models). Click **Next**.
- 4 On the next screen, select SPK as your free variable and click the button to match it up to FreeVariable1. Click **Next**.
- 5 Here you must select models for your objectives. Note that this stage was skipped for the first simple example (single-objective optimization) by clicking **Finish** on the previous screen.
 - You can set up objectives, constraints, and operating point sets in the **Optimization Wizard**. You can change any of these settings later.
 - You can also click **Finish** on any of these screens of the wizard and set up any or all of these later from the main **Optimization** view in the CAGE Browser, as in the first tutorial example.

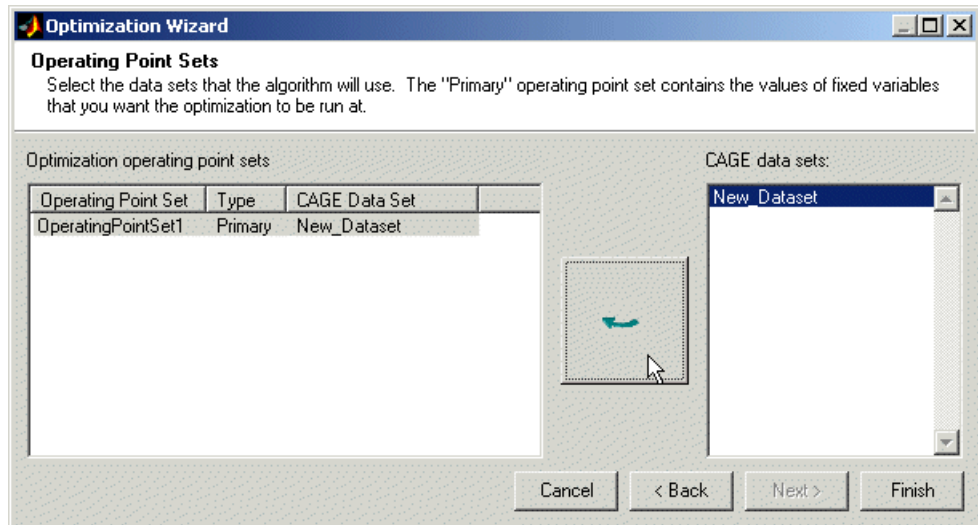


- 6 Select TQ_Model and click the button to match it up to Objective1, then select the **Maximize** radio button.
- 7 Select NOXFLOW_Model and click the button to match it up to Objective2. Leave the **Minimize** radio button selected, and click **Next**.
- 8 On the next screen you can set up a constraint.




Select the `NOXFLOW_Model` and click the button to match it to `Constraint1`. Enter 250 in the edit box and leave the operator at `<=` to constrain the `NOXFLOW_Model` to a maximum of 250. Click **Next**.

- 9 Select `New_Dataset` and click the button to match it up to `OperatingPointSet1`. Then click **Finish**.



A new node, NBI, appears in the Optimization tree. Your CAGE browser should look like the following example. Your optimization has objectives, an operating point set, and a constraint all set up and is ready to run.

Optimization Information			
Algorithm name	NBI		
Description	Normal Boundary Intersection Algorithm		
Free variables	SPK		
OperatingPointSet1 variables	None required		
Objectives			
Name	Description	Type	Information
Objective1	TQ_Model(SPK, L, N, A, E)	Maximize	
Objective2	NOXFLOW_Model(SPK, L, N, ...)	Minimize	
Constraints			
Name	Description	Information	
Constraint1	NOXFLOW_Model(SPK, L, N, A, E) <= 250		
Operating Point Sets			
Name	Description	Type	
OperatingPointSet1	New_Dataset(L, N)	Primary	

10 Click Run Optimization () in the toolbar.

As before, some values for the model input factors are required before the optimization can run. Recall that

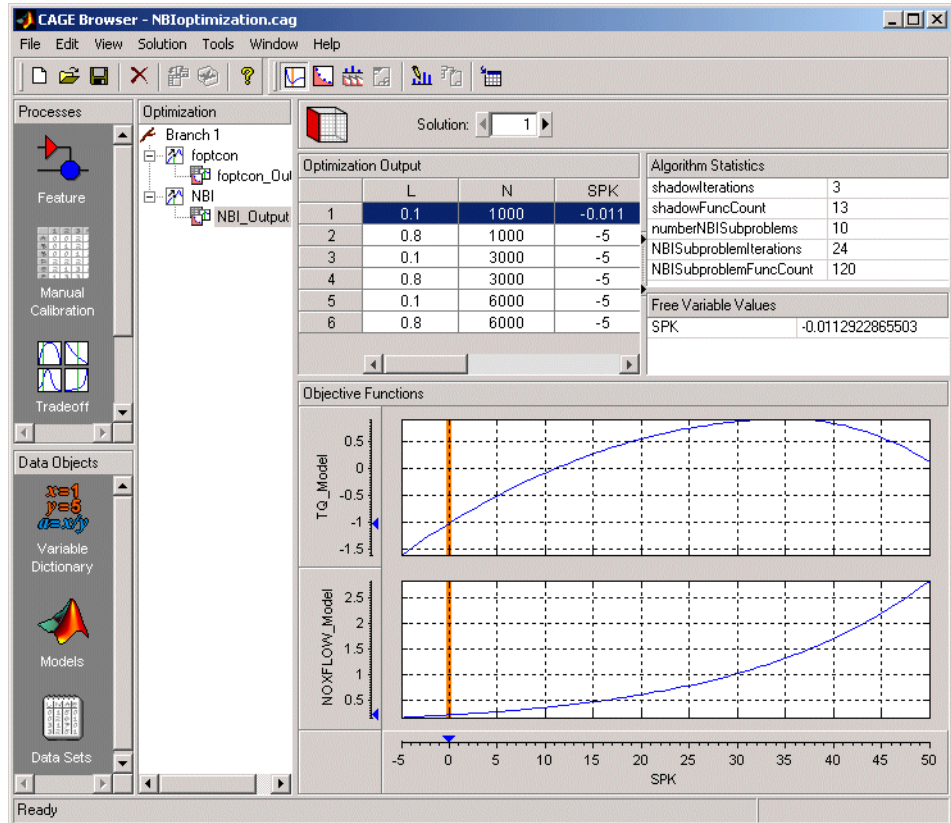
- SPK is the free variable, so the optimization will provide values for SPK as it searches for the optimum. A starting value and range are required for SPK.
- L and N are specified by the operating point set, but you need to give values for the other fixed variables, A and E.


A dialog appears called **Set Constant Values**. Click **OK** to accept the defaults, then click **OK** in the next dialog, **Free Variable Set Up**, to accept those default values.

The optimization runs, showing progress messages as each point is evaluated until the optimization is complete. A new node, NBI_Output, appears in the Optimization tree.

Optimization Output Node

- 1 Expand the NBI node in the Optimization tree by clicking the plus sign. Then click the NBI_Output node to view the optimization output.



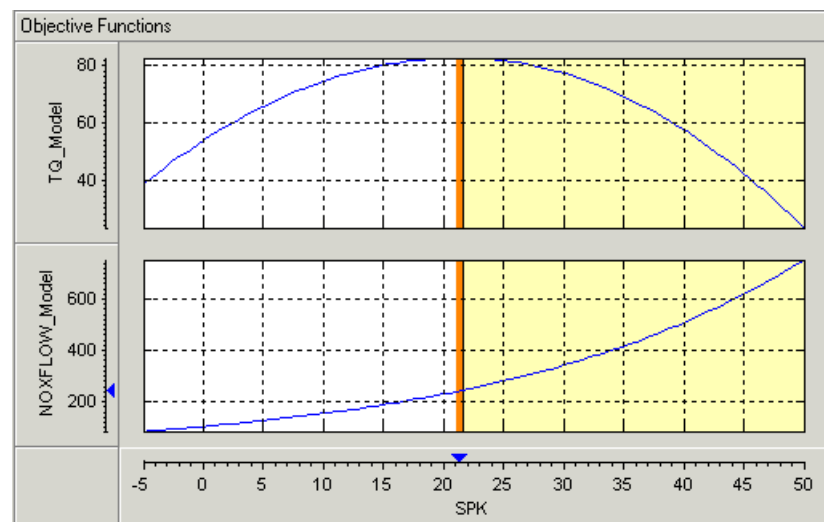
The toolbar buttons determine which view is displayed. The default is the Solution View (). The Solution View shows one solution at all operating points. That is, you can see a table of all operating points at once, and you can scroll through the solutions using the **Solution** buttons at the top. At 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.

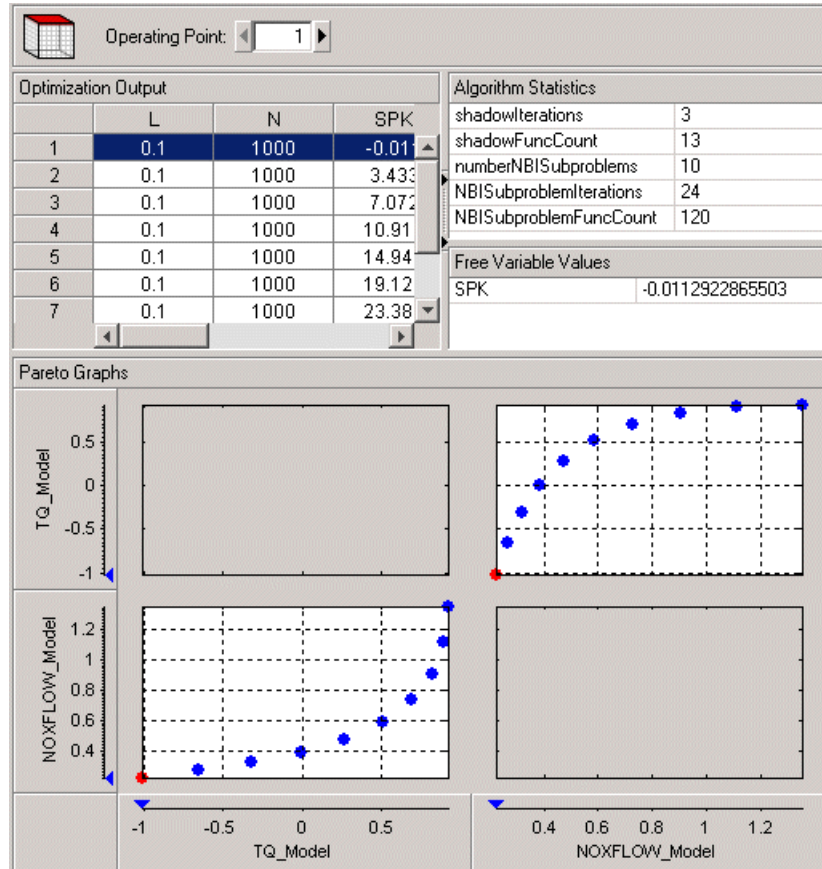
The graphs show the objective functions at the currently selected operating point (highlighted in the table), with the solution value shown in red.

Note that before you run an NBI optimization you can specify how many solutions you want the optimization to find, using the Set Up and Run Optimization toolbar button.

- 2 For an example point click in the table to select operating point 6, and enter 10 in the **Solution** edit box. Observe the effect of the constraint you applied in the objective function graphs, as shown in the example. Areas in yellow are excluded by constraints. Similarly, if you export boundary constraint models from the Model Browser, they appear in optimizations 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 more optimizations to find valid solutions. Changing your settings to make constraints less stringent and choosing more suitable starting values can help in these cases.



3 Click Pareto View () in the toolbar.

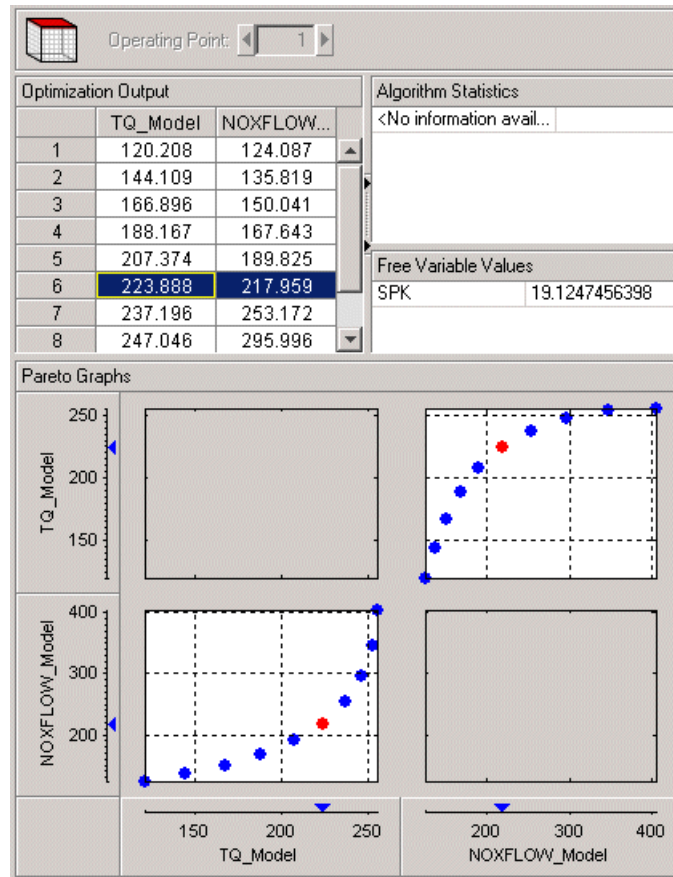


Here you can see all solutions found by the optimization at one operating point. The table shows all solutions at a single operating point. You can scroll through the operating points using the **Operating Point** buttons at the top. The graphs show all solutions for the selected operating point, with the currently selected solution in red. Click in the table to select different solutions.

Recall that the first example, a single-objective optimization, produced a single solution at each point, so you could view the **Pareto View** but it only


contained one solution at each operating point. The **Pareto View** 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 6-28.

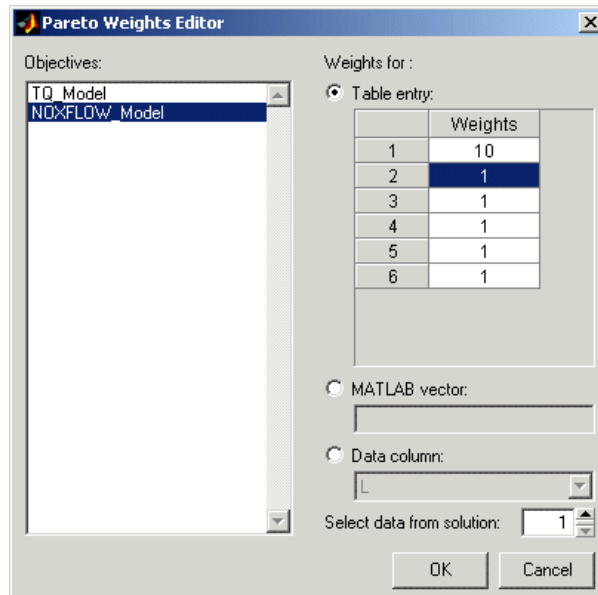
4 Click Weighted Pareto View () in the toolbar.



This 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 data column in your data set by selecting the other radio buttons. If you select **Data 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 View** 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.

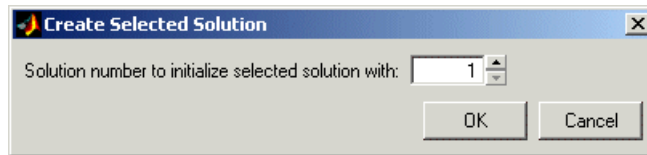
Selecting Best Solutions

In a multiobjective optimization, there is more than one possible optimal solution at each operating point. You can use the **Selected Solution View** to collect and export those solutions you have decided are optimal at each operating point.


Once you have enabled the **Selected Solution View**, you can use the plots in the **Pareto View** and **Solution View** to help you select best solutions for each operating point. These solutions are saved in the **Selected Solution View**. You can then export your chosen optimization output for each point from the **Selected Solution View**, in order to use your optimization output to fill tables.

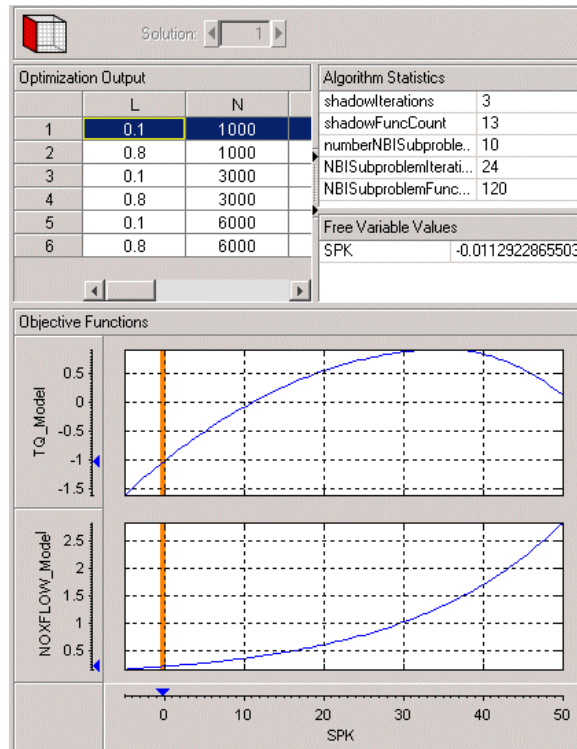
- 1 In order to choose a single solution at each operating point, you need to enable the **Selected Solution View**. Select **Solution** -> **Selected Solution** -> **Initialize**.

A dialog called **Create Selected Solution** appears. Click **OK**.




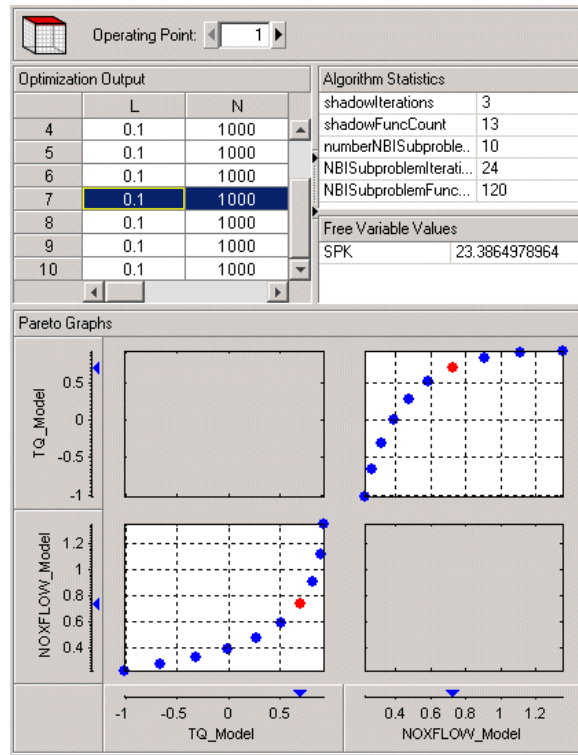
The default initializes the first solution for each operating point as the selected solution.


- 2 Click the Selected Solution View button () which is now enabled in the toolbar. Observe that the **Solution** number at the top is not editable, and is initially solution 1 for each operating point you click in the table. You must select the solutions you want using the **Pareto View** and **Solution View** to decide which solution is best for each point.



3 Return to the **Pareto View** and select a solution for operating point 1. An example is shown with solution 7 highlighted. To select this solution as best, do one of the following:

- Click Select Solution () in the toolbar.
- Select the menu item **Solution** -> **Selected Solution** -> **Select Current Solution**.



- 4 If you return to the **Selected Solution View** you can now see that solution 7 is now present for operating point 1, while all the other operating points remain at the initial solution 1. This 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 your optimization output to fill tables, you should repeat this process to select a suitable solution for all operating points. Then click **Export to Data Set** () in the toolbar. Go to the **Data Sets** view (click **Data Sets** in the **Processes** pane) to see that the table of optimization results is contained in a new data set, `Exported_Optimization_Data_1`. You could use

these optimization results to fill tables, as described in “Using Optimization Results to Fill Tables” on page 6-16.

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 View** before exporting the data. If you export from the **Pareto View**, the new data set contains all solutions at the single currently selected operating point set. If you export from the **Solution View** 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 View**, as there was no choice of solutions to be selected. See “Single-Objective Optimization” on page 6-6.

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. A sum optimization finds the optimal value of a weighted sum of each objective function. The weighted sum is taken over each point in the operating point set, and the weights can be edited.

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.

In order to create a sum optimization, all objectives and constraints must be sum objectives and model sum constraints, and the optimization must contain a data set. The following instructions describe these settings.

- 1 Select the NBI node in the Optimization tree.
- 2 Select **Edit** → **Duplicate NBI**.

A copy of the NBI node called NBI_1 appears in the tree. 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.

- 3 Select the node NBI_1 and select **Edit** → **Rename**. Edit the name to read SUM_NBI.

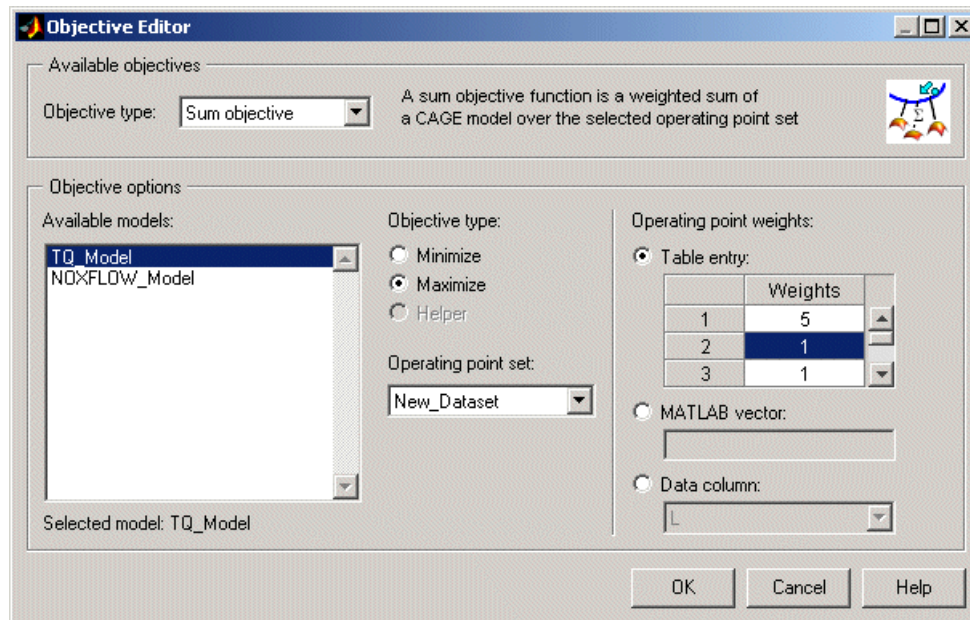
You can only construct a sum optimization if your optimization contains an operating point set (as you cannot construct a sum if there is only one point).

You must edit all the objectives and constraints in your existing optimization to be sum objectives and model sum constraints.

- 4 Double-click Objective1 (or right-click and select **Edit Objective**). The **Objective Editor** appears.

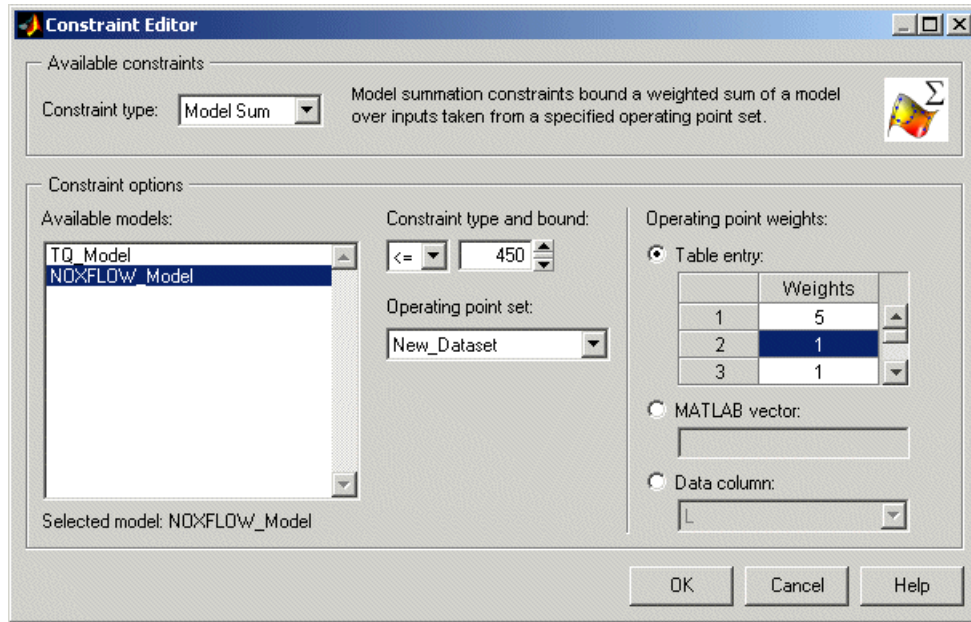
- 5 Select **Sum Objective** from the **Objective Type** drop-down menu.

Operating Point Weights controls appear. You need to set up your new objective.



- 6 Select **TQ_Model** and **Maximize**. Torque has a strong correlation with fuel consumption so this sort of problem could be useful for a fuel consumption study.
- 7 You can edit the weights to make certain operating points (for example idle engine speed) more important, giving more flexibility to other points. As in the example shown, edit the weight of the first operating point to read 5, and leave the other weights at 1. To calculate the weighted sum, for each solution the first operating point output value will be multiplied by 5, and the other operating point values by 1.
- 8 Click **OK** to finish editing the objective.
- 9 Repeat to edit **Objective2**. Set up a sum objective to minimize **NOXFLOW**, with a weighting of $5 \times \text{operating point 1}$.


- 10 You must also edit your constraint to be a sum constraint. Double-click Constraint1 and the **Constraint Editor** appears.



- 11 Select Model Sum from the **Constraint Type** drop-down menu, then ensure that NOXFLOW_Model is selected under **Available Models**.

- 12 Enter 450 in the constraint bound edit box, and give the first operating point a weighting of 5. Click **OK**.

You have modified your objectives and constraint for a sum optimization, which is ready to run.

- 13 Click Run Optimization () in the toolbar, and click **OK** in the next two dialogs to accept the default starting values.

- 14 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, select the output node to examine the results.


Automated Tradeoff

Once you have set up an optimization you can use automated tradeoff. You can select cells to fill by applying 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 “Creating a Tradeoff Calibration” on page 3-3).

- 1 Select **File -> New -> Tradeoff**.

This takes you to the **Tradeoff** view. You need to add tables to the tradeoff.

- 2 Click . This opens the **Add Table to Tradeoff** dialog.
- 3 Select the **Create a New Table** radio button.
- 4 Enter Spark as the Table name.
- 5 Enter Speed as the **X name** and Load as the **Y name**.
- 6 Enter 10 as the size of the speed and load axes.
- 7 Fill the Speed axis with N and the Load axis with L.
- 8 Fill the table with SPK (spark angle).
- 9 Click **OK**.

A new SPK table appears in the **Tradeoff** tree. Set up the spark table as follows:

- 1 Expand the SPK node and select the normalizer Speed.
- 2 Click Initialize in the toolbar to space the normalizers over the ranges of N and L.

You need to select the cells where you want to apply automated tradeoff. Create a region within the table:

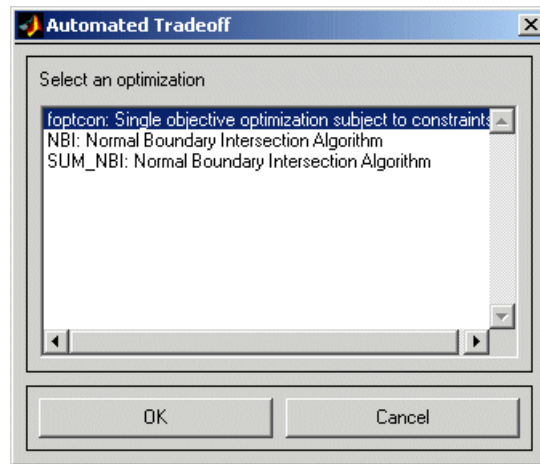
- 1 Highlight a rectangle of cells in the SPK table by clicking and dragging. Note that a large region can take a very long time to evaluate. Try four cells to start with.

- 2 Click  (or right-click and select **Define Region**) to define the region. The cells become blue.

To use automated tradeoff on the cells in a defined region,

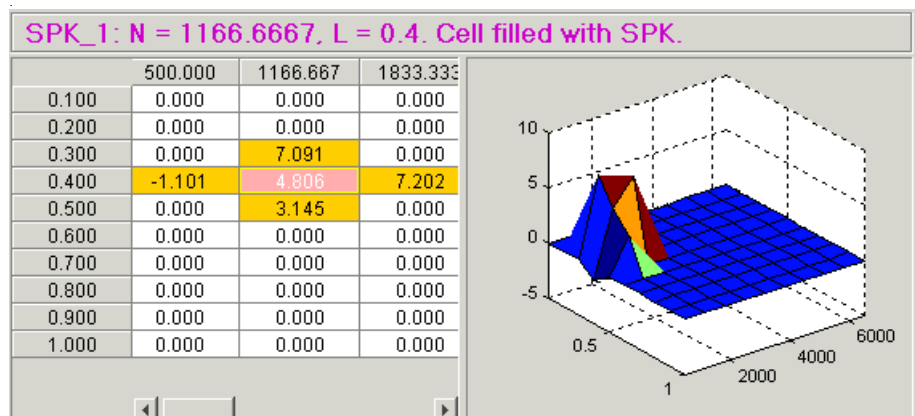
- 1 Click any cell in a region.
- 2 To apply optimization, select **Table -> Automated Tradeoff**.

The **Automated Tradeoff** dialog appears, showing a list of available optimizations in your session that are set up and ready to run.



- 3 Select your NBI multiobjective optimization to apply to the tradeoff, and click **OK** in the following two dialogs to accept the default starting values for the optimization.

The automated tradeoff optimization runs, showing progress messages as each point in the region is evaluated. 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.



Worked Example Optimization

There is a simple worked example provided to show you what you can do by modifying the template file to write your own optimizations. This example demonstrates a simple use of the CAGE optimization feature. The aim of this example is to obtain values of spark (SPK) and air/fuel ratio (AFR) that maximize torque at a given speed (N) and load (L). These values could then be used to fill calibration tables.

An example of a user-defined optimization algorithm is provided.

- To see a description of this algorithm, at the command line type
`help mbcweoptimizer`

`mbcweoptimizer` is an example of a user-specified optimization that solves the following problem:

Maximum TQ over (AFR, SPK) at a given (N, L) point.

The syntax for this example function, `mbcweoptimizer`, mimics that used in the Optimization Toolbox.

`[bestafr,bestspk]=mbcweoptimizer(TQ, speed, load)` finds an optimal `(bestafr,bestspk)` that gives a maximum TQ at the given speed and load.

- To evaluate this at the command line, type this example:
`[bestafr,bestspk]=mbcweoptimizer(@mbcTQ,1000,0.2)`

The optimization finds values of AFR and spark (the free variables) that give the maximum output from TQ at the values of speed and load (the fixed variables) that you specified, as shown below.

```
bestafr =  
12.9167  
  
bestspk =  
25
```

To use this optimization algorithm in CAGE, you need to include the function in a CAGE optimization function M-file. This worked example modifies the template provided to show you how to use your own user-defined algorithms.

You can find detailed information on all the available CAGE optimization interface functions in “Optimization Template” on page 11-45.

- To view the worked example M-file, at the command line, type
`edit mbcOSworkedexample`

The worked example optimization wraps `mbcweoptimizer` in a function that can be called by the CAGE optimization feature. When you run your optimization from CAGE, you can alter the search ranges of the free variables and the resolution of the search.

Using the Worked Example Optimization

In order to run any optimization, you first need to set up your CAGE session. You need the following:

- A model
- An operating point set (in the case of this worked example)

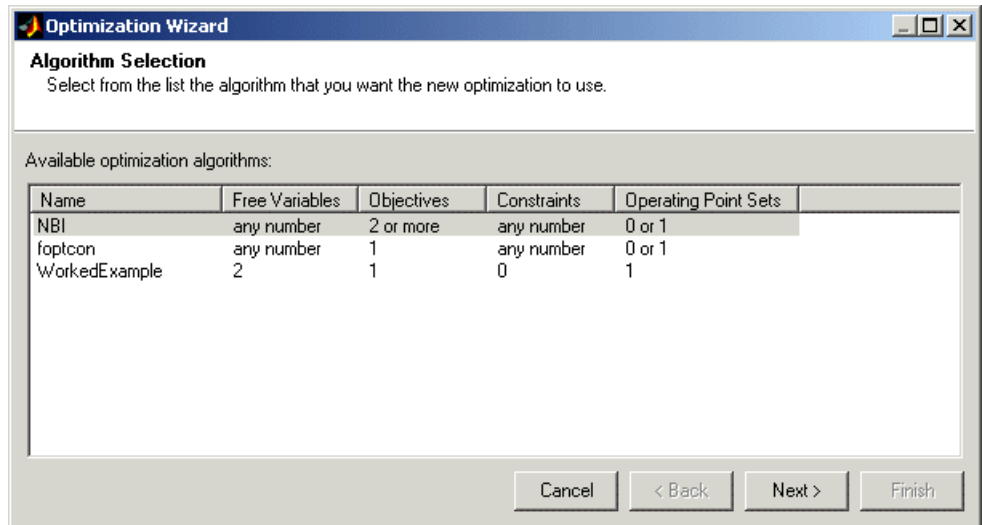
For this example, the CAGE session requires

- A torque model
- A variable dictionary defining required variable ranges and set points (N, L, AFR, and SPK)
- A data set defining the (N,L) points where you want to run the optimizer

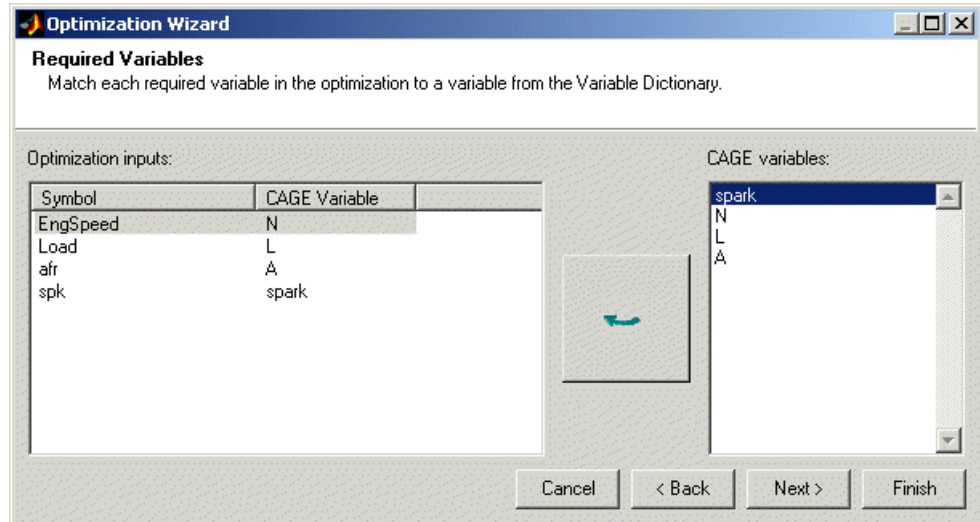
There is a preconfigured session provided that contains the model, variable dictionary, and data set.

- 1 Select **File** → **Open Project** and load the file `optimworkedexample.cag`. This should be in the `mbctraining` directory.
- The `tq` model was fitted to the Holliday engine data and exported from the Model Browser quick start tutorial (also used in the CAGE feature calibration tutorial). It can be found in `tutorial.exm` in the `mbctraining` directory. To view this model in your current session, click the **Models** button in the **Data Objects** pane.
 - You can look at the variables by clicking the **Variable Dictionary** button in the **Data Objects** pane.

- You can look at the operating point set by clicking **Data Sets** in the **Data Objects** pane.
- 2 Select **File -> New -> Optimization**.
The **Optimization Wizard** appears.
 - 3 Select **Worked Example**, and click **Next**.

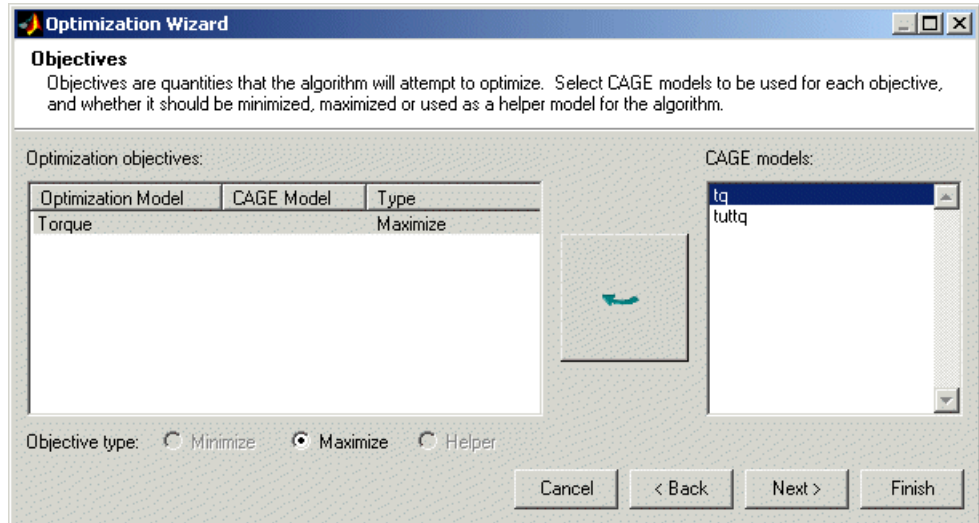


- 4 Associate each pair of inputs and variables, for example by clicking spark in the left and right lists, and then click the large **Select** button. Similarly associate Speed with N, Load with L, and afr with A. Click **Next**.

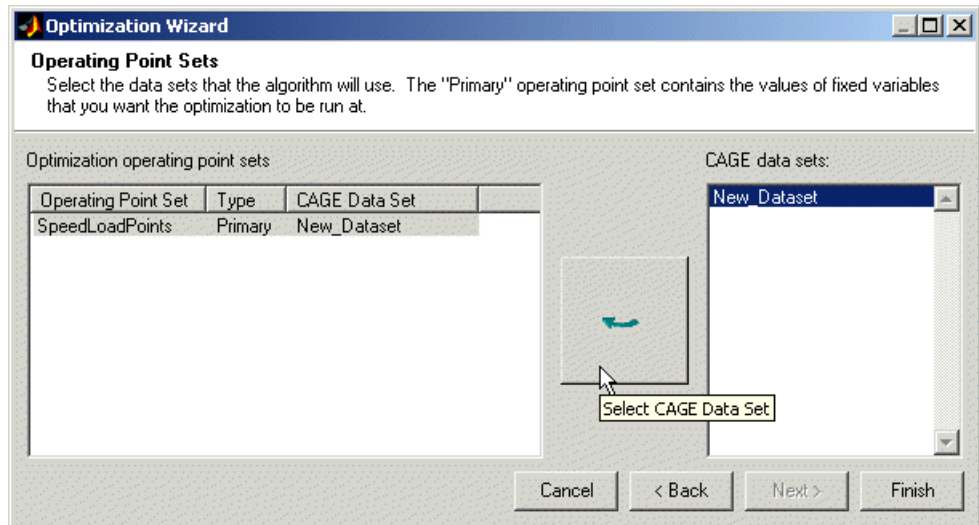


- 5 The next screen of the wizard automatically shows the tq model selected and **Maximize** chosen; these are specified in the function. Click the button to

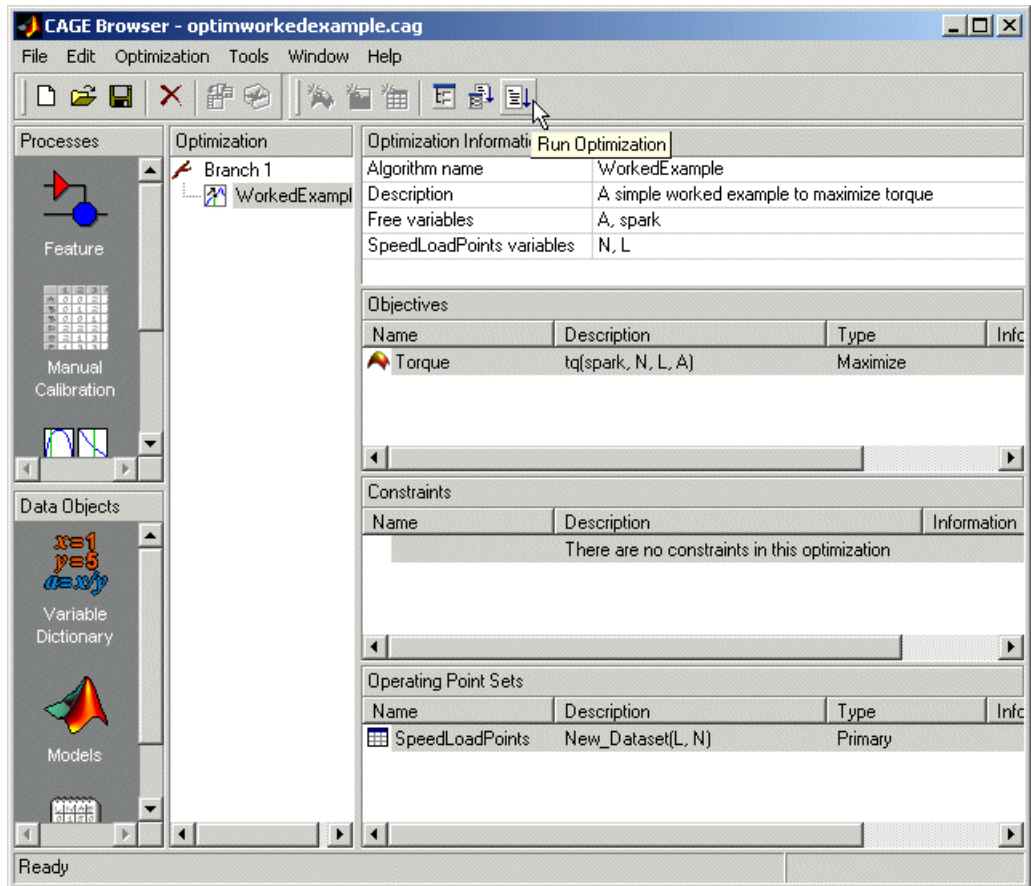
match the tq CAGE model with the Torque optimization model, then click **Next**.




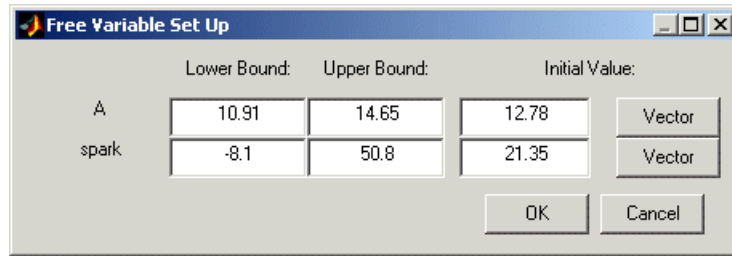
- 6 The next screen of the wizard automatically shows the `New_Dataset` selected (as it is the only one in the session). Click the button to match it to the operating point set `SpeedLoadPoints`, then click **Finish**.



This completes the optimization setup. CAGE switches to the **Optimization** view and the new WorkedExample node appears in the tree. The setup details appear in the right pane: the model to be maximized and the operating point set to use, as shown in the following example.



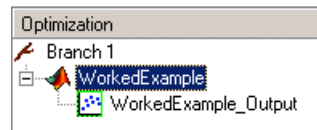
7 Click Run Optimization  in the toolbar.



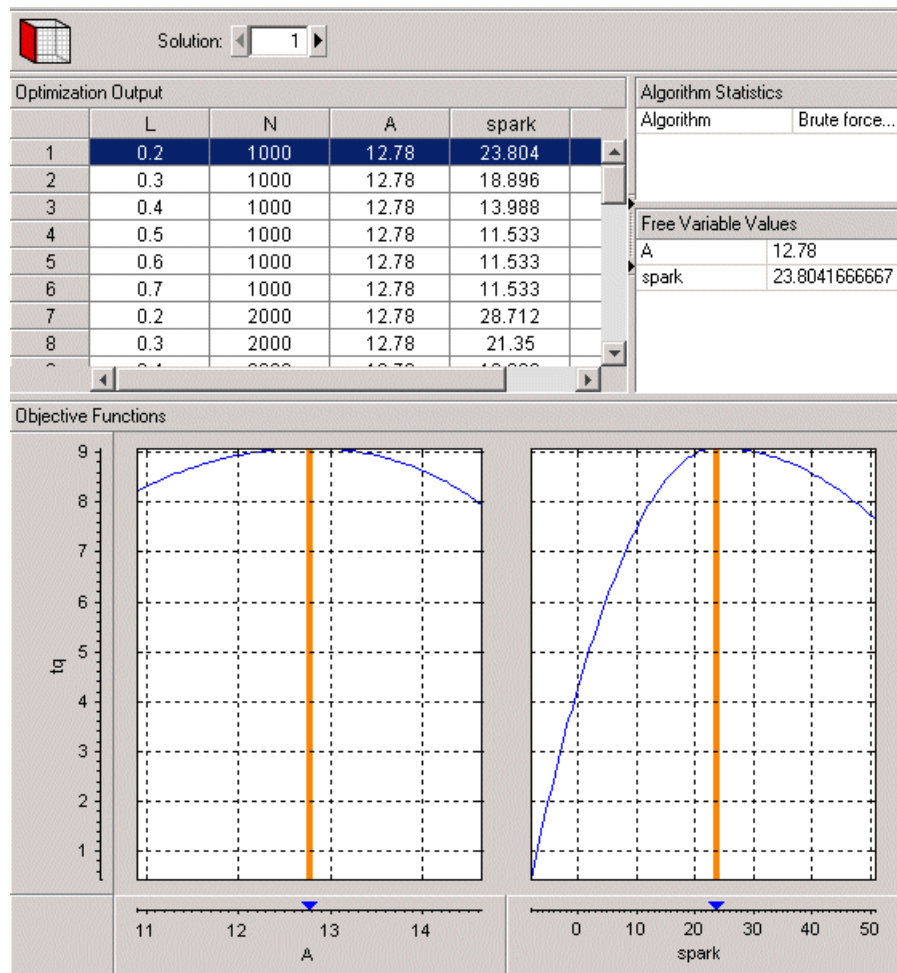
The **Free Variable Set Up** dialog box appears. These bounds and initial values are taken from the Variable Dictionary (where you also could change them if you wanted). Click **OK** to accept these values.

You will see the **Optimization** progress bar as the optimization runs.

- 8 When the progress bar disappears, click the new `WorkedExample_Output` node in the tree. First you must expand the `WorkedExample` node as shown below.



The output display should look like the following. The optimization has found the values of SPK and AFR that give the maximum model value of torque at each operating point specified. Select different operating points by clicking in the table: the model plots at the selected operating point are shown. There is only one solution per operating point, so you cannot scroll through the solutions.



For a discussion of the worked example code and how the external optimization algorithm is implemented in the CAGE optimization, see “About the Worked Example Optimization Algorithm” on page 11-42. For a detailed walk-through of incorporating an example user-defined optimization algorithm into a CAGE optimization function, see the next tutorial section, “Creating an Optimization from Your Own Algorithm” on page 6-46.

Creating an Optimization from Your Own Algorithm

The CAGE optimization feature allows you to use your own optimization algorithms as alternatives to the library routines `foptcon` and `NBI`.

Using an example, this tutorial illustrates how to take an existing optimization algorithm and implement it as an optimization function for use in CAGE optimization.

The problem to be solved is the worked example problem:

Maximize torque (TQ) over the free variables (SPK, AFR) over a specified set of (N, L) points. These points are defined in the data set `New_Dataset`, which can be found in the CAGE session `optimworkedexample.cag`.

The torque model to be used is that in `/mbctraining/Holliday.mat`.

Process Overview

- 1 Start with your own algorithm. We provide an example.
- 2 Create a CAGE optimization function.
- 3 Define the attributes of your optimization in the CAGE optimization function.
- 4 Add your algorithm to the CAGE optimization function.
- 5 Register your completed optimization function with CAGE.
- 6 Verify the optimization.

The steps of this tutorial lead you through a series of examples illustrating how to construct the code to incorporate your own algorithm into an optimization in CAGE.

Before you begin you must create a working directory.

- 1 Create a new folder (for example, `C:\Optimization_Work`). We recommend that you place this directory outside your MATLAB folders to avoid interfering with toolbox files.

- 2 Copy the following six files from the mbctraining directory into your new working folder:

```
curr tutoptim.m  
mbc0Stemplate.m  
mbc0Stutoptimfunc.m  
mbc0Stutoptimfunc_s1.m  
optimtut.mat  
optimtuteg.mat
```

- 3 Make sure your new working directory is on the MATLAB path.
 - a Select **File** -> **Set Path**.
 - b Click **Add Folder** and browse to your working directory.
 - c Click **OK**.
 - d Click **Save**.

Step 1: Examine the Algorithm

- 1 Open curr tutoptim.m from the mbctraining directory. curr tutoptim.m is an optimization algorithm that solves the worked example problem in the MATLAB workspace. You should see the following code in the MATLAB editor.

```

function [bestx, OUTPUT] = curr_tutoptim(x0, fixedvars)
%MBCOSTTUTOPTIM CAGE Optimization tutorial - Original Algorithm
%
% Copyright 2000-2003 The MathWorks, Inc. and Ford Global Technologies, Inc.
% $Revision: $    $Date: $

no_of_speed_load_points = size(fixedvars, 1);

% Optimize torque
waitH = WAITBAR(0, '', 'name', 'Tutorial Optimization');
for i = 1:no_of_speed_load_points
    [bestx(i, :), notused1, notused2, OUTPUT{i}] = fminunc(@i_evalObj, x0, [], fixedvars(i, :));
    wbarstr = ['Computing optimal settings for operating point ' num2str(i)];
    waitbar((i-1)/no_of_speed_load_points, waitH, wbarstr);
end

% End Optimization
waitbar(1, waitH, 'Optimization completed');
close(waitH);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function tq = i_evalObj(x0, fixedvars)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Evaluate the torque objective function

tq = trqfunc(x0(1), x0(2), fixedvars(1), fixedvars(2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = trqfunc(S, A, N, L)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%TRQFUNC Objective function (Torque) for Optimization example
%

% Load the torque model from MBC
load('optimtut.mat');

% Evaluate
y = evalModel(tq, [S, N, L, A]);
y = -y;

```

2 Verify that this algorithm solves the worked example problem by typing the following commands at the MATLAB prompt:

```

load optimtuteg.mat;
bestX = curr_tutoptim(x0, data)

```

The workspace output should resemble

```

BestX =
    23.768    12.78
    18.179    12.78

```

14.261	12.78
12.014	12.78
11.439	12.78
12.535	12.78
27.477	12.78
21.887	12.78
17.969	12.78
15.722	12.78
15.147	12.78
16.243	12.78
31.185	12.78
25.595	12.78
21.677	12.78
19.43	12.78
18.855	12.78
19.951	12.78
34.893	12.78
29.303	12.78
25.385	12.78
23.138	12.78
22.563	12.78
23.659	12.78
38.601	12.78
33.012	12.78
29.093	12.78
26.847	12.78
26.271	12.78
27.368	12.78
42.309	12.78
36.72	12.78
32.802	12.78
30.555	12.78
29.979	12.78
31.075	12.78

The matrix `bestX` contains the optimal SPK and AFR values that maximize the MBC model torque (exported from `Holiday.mat`) at the speed and load points defined in the matrix data.

`curr_tutoptim.m` is the example optimization algorithm that you want to transfer to the CAGE optimization feature.

This tutorial shows how to make `curr_tutoptim.m` available for use in the CAGE optimization feature.

Step 2: Create a CAGE Optimization Function

Any optimization algorithm you want to use in CAGE must be contained in an optimization function. A CAGE optimization function consists of two sections.

The first section defines the following attributes of the optimization:

- A name for the optimization
- A description of the optimization
- Number of objectives
- Labels for objective functions, so the user can match models in CAGE to the required algorithm objectives
- Number of constraints
- Labels for constraints, so the user can match models in CAGE to the required models in your algorithm constraints
- Number of operating point sets
- Labels for operating point sets, so the user can match data sets in CAGE to the required fixed variable data for your algorithm
- Any other parameters required by the optimization algorithm

The second section contains the optimization algorithm.

3 Open `mbcOSTemplate.m` from the `mbctraining` directory.

You should see the following M-file in the MATLAB editor.

```

function out = mbcOStemplate(action, in)
%MBCOSTEMPLATE CAGE Optimization template function
% OUT = MBCOSTEMPLATE(ACTION, IN) is a template function for use with
% CAGE Optimization. This function can be used to create user-defined
% optimization functions that can subsequently be used in CAGE.

% Copyright 2000-2003 The MathWorks, Inc. and Ford Global Technologies, Inc.

% $Revision: $    $Date: $

% Deal with the action inputs
if strcmp(action, 'options')

    options = in;

    %
    % Define optimization attributes here
    %

    out = options;

elseif strcmp(action, 'evaluate')

    optimstore = in;

    %
    % Put optimization algorithm here
    %

    out = optimstore;

else
    error('Incorrect action type specified');
end

```

Section 1

Section 2

`mbcOStemplate.m` is an empty CAGE optimization function. The two (currently empty) sections of the function are labeled above. Note that this M-file can be used as a template for any optimization function that you write.

Step 3: Define the Optimization Options

The next step is to define the attributes of your optimization.

- 1 Open `mbcOStutoptimfunc_s1.m` from the `mbctraining` directory. In this M-file, you can see the optimization attributes that have been defined.

The following is a code fragment from this file:

```
% Deal with the action inputs
if strcmp(action, 'options')

    options = in;

    % Add a name
    options = setName(options, 'Tutorial_Optimization');

    % Add a description
    options = setDescription(options, 'A simple worked example to maximize torque');

    % Set up the free variables
    options = setFreeVariablesMode(options, 'fixed');
    options = addFreeVariable(options, 'afr');
    options = addFreeVariable(options, 'spk');

    % Set up the objective functions
    options = setObjectivesMode(options, 'fixed');
    options = addObjective(options, 'Torque', 'max');

    % Set up the constraints
    options = setConstraintsMode(options, 'fixed');
    % There are no constraints for this example

    % Set up the operating point sets
    options = setOperatingPointsMode(options, 'fixed');
    options = addOperatingPointSet(options, 'SpeedLoadPoints', {'speed', 'load'});

    % Set up the optimization parameters
    options = addParameter(options, 'Resolution', 'number', 25);

    out= options;
elseif strcmp(action, 'evaluate')

    optimstore = in;

    %
    % Put optimization algorithm here
    %

    out = optimstore;
else
    error('Incorrect action type specified');
end
```


The optimization attributes are passed to CAGE via the `cgoptoptions` object, referenced by options in the code in `mbc0Stutoptimfunc_s1.m`. See after the table for details of the `cgoptoptions` object. The `cgoptoptions` object has a set of functions that set the optimization attributes in CAGE. This is where you specify the name, description, free variables, objective functions, constraints, operating point sets, and optimization parameters for the optimization.

For detailed information on all the available functions, see “List of Optimization Functions” on page 11-47. The above code has used the `cgoptoptions` object (`options`) to set the optimization attributes as described in the following table.

Look through the code to locate the listed **Code Section Where Set** for each attribute to see how each of the optimization options is set up.

Attribute	Value	Code Section Where Set
Optimization Name	<code>Tutorial_Optimization</code>	Add a name
Description	A simple worked example to maximize torque	Add a description
Number of Free Variables	Cannot be changed by the user in the GUI (the mode has been set to 'fixed')	Set up the free variables <code>setFreeVariablesMode</code>
Required Free Variables	This function requires two free variables, labeled 'afr' and 'spk'. The user matches these free variable labels to CAGE variables in the Optimization Wizard .	Set up the free variables <code>addFreeVariables</code>
Number of Objectives	Cannot be changed by the user in the GUI (the mode has been set to 'fixed')	Set up the objective functions <code>setObjectivesMode</code>
Required Objective functions	This function requires one objective function, which will be labeled 'Torque' in the optimization feature. The user matches this 'Torque' label to a CAGE model.	Set up the objective functions <code>addObjective</code>

Attribute	Value	Code Section Where Set
Number of Constraints	Cannot be changed by the user in the GUI (the mode has been set to 'fixed')	Set up the constraints SetConstraintsMode
Required Constraints	As the mode is fixed and no constraint labels have been defined, this is an unconstrained optimization.	Set up the constraints %There are no constraints
Number of Operating Point Sets	Cannot be changed by the user in the GUI (the mode has been set to 'fixed')	Set up the operating point sets setOperatingPointsMode
Required Operating Point Sets	This function requires one operating point set, which will be labeled 'SpeedLoadPoints' in the CAGE optimization feature. This data set must have two columns, which will correspond to the fixed variables speed and load. The end user must match the 'SpeedLoadPoints' label to a CAGE data set.	Set up the operating point sets addOperatingPointSet
Optimization Parameters	This function can use an optional parameter, 'Resolution'. The default value of this parameter is 25.	Set up the optimization parameters addParameter

When one of your optimizations is created in the CAGE GUI, CAGE first calls your optimization function to define the attributes of the optimization. The function call from CAGE has the form

```
optionsobj = <your_optimization_function>('options', optionsobj)
```

This is how your optimization function receives the `cgoptoptions` object. Note that your optimization function *must* support this interface.

Step 4: Add the Algorithm to the Optimization Function

In this step you complete the optimization function by adding your algorithm. To do this, a few changes need to be made to the algorithm code, as data (for

example, free variable values, constants, and so on) will now be passed to and from CAGE rather than from the MATLAB workspace.

1 Open `mbc0Stutoptimfunc.m` from the `mbctraining` directory.

This M-file contains the completed optimization algorithm. The following is a code fragment from this file.

```
elseif strcmp(action, 'evaluate')
    optimstore = in;
    optimstore = tutoptimizer(optimstore);
    out = optimstore;
```

A single line has been added, namely

```
optimstore = tutoptimizer(optimstore)
```

This line calls the modified optimization algorithm. Note the syntax of the algorithm: it *must* take the form

```
Optimstore = <your_optimization_algorithm>(optimstore)
```

2 The subfunction `tutoptimizer` can be found at the bottom of the `mbc0Stutoptimfunc.m` file. Scroll down to view the modified algorithm.

`optimstore` is a `cgoptimstore` object. This is an interface object that allows you to get data from and set data in the CAGE optimization feature. You can now see how the `optimstore` object is used by comparing the modified optimization algorithm, `tutoptimizer`, with the original algorithm, `curr_tutoptim`, for each of the main actions of the algorithm.

Action 1. Determine the number of points (`no_of_speed_load_points`) the optimization is to be run at.

Original code:

```
no_of_speed_load_points = size(fixedvars, 1);
```

Modified code:

```
no_of_speed_load_points = getnumrowsindataset(optimstore,
'SpeedLoadPoints');
```

In the original algorithm, a matrix of (N , L) points had to be passed in to the function and `no_of_speed_load_points` is determined from that. In CAGE,

the data is not passed in to the algorithm in this manner, so we invoke a function on the `optimstore` object to return `no_of_speed_load_points`.

Action 2. Get the start conditions (`x0`) for the free variables.

Original code:

`x0` passed in as an input to the algorithm.

Modified code:

```
x0 = getInitFreeVal(optimstore);
```

In the original algorithm, `x0` is passed into the algorithm as an input. In CAGE, we invoke the `getInitFreeVal` function on the `optimstore` object.

Action 3. Perform the optimization.

Original code:

```
[bestx(i, :), notused1, notused2] = fminunc(@i_evalObj, x0,  
[], fixedvars(i, :));
```

which calls the following code to evaluate the cost function:

```
function tq = i_evalObj(x0, fixedvars)  
  
% Evaluate the torque objective function  
  
tq = trqfunc(x0(1), x0(2), fixedvars(1), fixedvars(2));  
  
function y = trqfunc(S, A, N, L)  
  
%TRQFUNC Objective function (Torque) for Optimization example  
  
% Load the torque model from MBC  
load('optimtut.mat');  
  
% Evaluate  
y = evalModel(tq, [S, N, L, A]);  
y = -y;
```

Modified code:

```
[bestx(i, :), notused1, notused2] = fminunc(@trqfunc_new, x0(i,  
:), [], optimstore, i);
```

which calls the following code to evaluate the cost function:

```
function y = trqfunc_new(x, optimstore, row_index)

%TRQFUNC_NEW Objective function (Torque) for Optimization example

y = evaluate(optimstore, x, {'Torque'}, 'SpeedLoadPoints',
row_index);
y = -y;
```

In performing the algorithm, the only difference between the original and modified code is how the objective function is evaluated. The original algorithm requires the objective function (a Model-Based Calibration Toolbox model for torque) to be loaded in and evaluated as required. In the modified algorithm the objective function (torque) is evaluated by invoking the evaluate function on the optimstore object. Note that the inputs to the torque model are passed in to the evaluate function as shown in the following table.

Original Input	Input to Evaluate Function
S	X(1)
A	X(2)
N, L	The (N, L) points are retrieved from the CAGE data set, which is referenced by the label 'SpeedLoadPoints'. The (N, L) point used is the row_index th point of that data set.

Action 4. Retrieve output data.

Original code:

Optimal free variable settings are returned to the workspace.

Modified code:

```
% Write output info to optimstore
% Set the best values calculated for the free variable(s) into the
output data set

optimstore = setfreevariables(optimstore, bestx);

% return OK = 1 if everything went OK
```

```
OK = 1;  
% Update error message  
errormessage = '';  
  
% OK, output, errormessage  
  
optimstore = setOutputInfo(optimstore, OK, errormessage,  
    struct([]));
```

In the modified algorithm, the results need to be sent back to the CAGE optimization feature and not the MATLAB workspace. To do this, optimization results are set in the `optimstore` object, which is then returned to CAGE. There are two functions you should invoke on the `optimstore` object to return optimization results to CAGE:

- `setFreeVariables` — Returns the optimal free variable values to CAGE
- `setOutputInfo` — Returns any diagnostic information on the algorithm to CAGE

Step 5: Register Your Optimization Function with CAGE

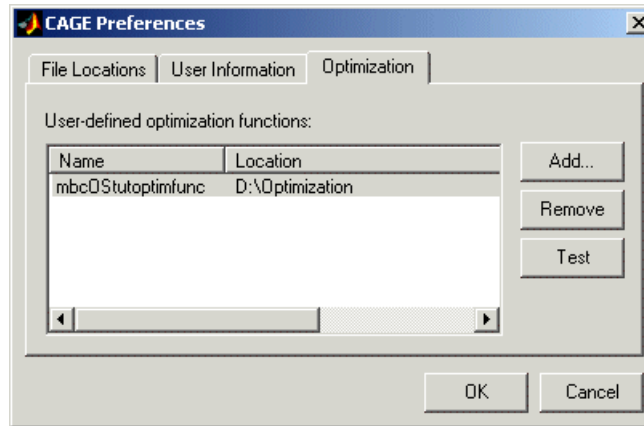
The worked example provided is preregistered so you can see it as an option in the **Optimization Wizard** when setting up a new optimization. You must register new functions before you can use them. When you have modified the template to create your own optimization function, as in this example, you must register it with the Model-Based Calibration Toolbox in order to use the function in CAGE. Once you have checked in your optimization function it appears in the **Optimization Wizard**.

- 1 In CAGE, select **File** → **Preferences**.

The **CAGE Preferences** dialog appears.

- 2 Click the **Optimization** tab and click **Add** to browse to your M-file.
- 3 Navigate to the `mbc0Stutoptimfunc.m` file (in the working directory you created) and click **Open**.

Clicking **Open** registers the optimization function with CAGE.



- 4** You can now test the function by clicking **Test**. This is a good check for any syntax errors in your optimization function. This is a very useful function when you use your own functions; if anything is incorrectly set up the test results will tell you where to start correcting your function.

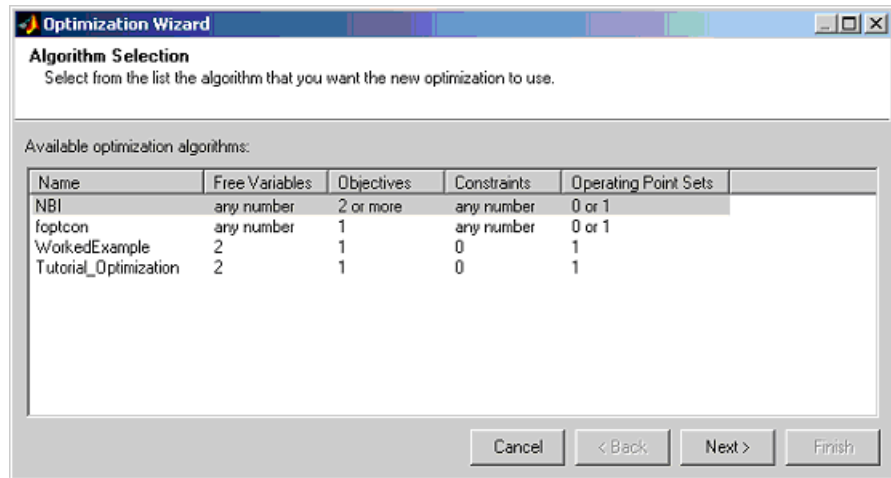
You could see an example of this by saving a copy of the worked example file and changing one of the variable names (such as `afr`) to a number. Try to check this altered function into CAGE, and the **Test** button will return an informative error specifying the line you have altered.

- 5** Click **OK** to leave the **CAGE Preferences** dialog. If the optimization function tested successfully, it is registered as an optimization function that can be used in CAGE, and appears in the **Optimization Wizard**.

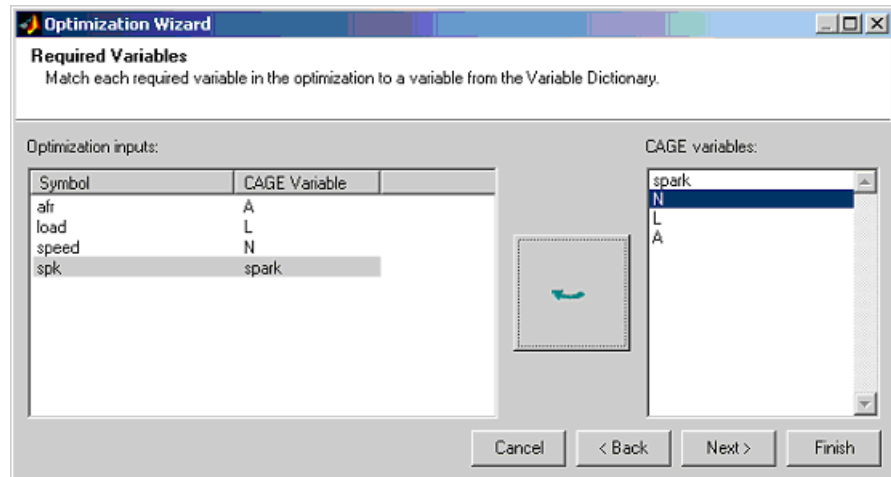
Step 6: Verify Your New Optimization

To verify the algorithm we set up a CAGE session to run the optimization that was performed in step 0. For this example, the CAGE session has already been set up. Follow the steps below to run the tutorial optimization in CAGE.

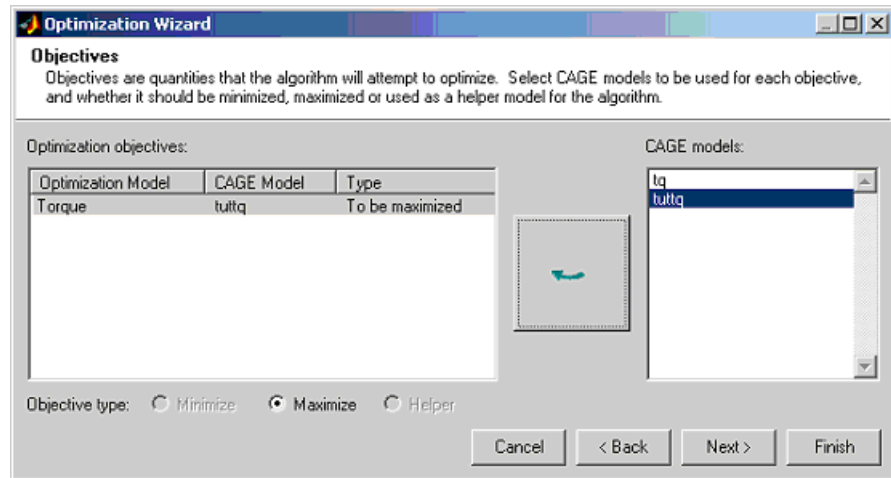
- 1** Select **File** -> **Open Project** and open the session `optimworkedexample.cag`.
- 2** Select **File** -> **New** -> **Optimization**.



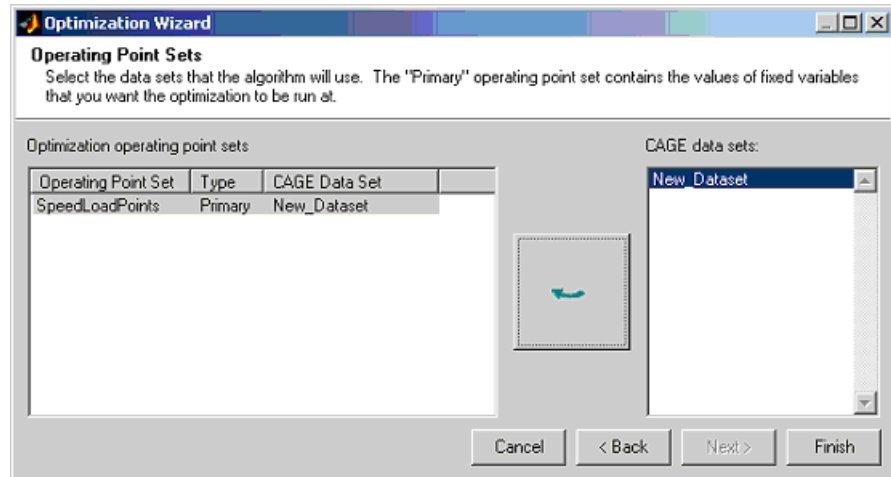
- 3 The newly registered optimization appears in the list of algorithm names. Select `Tutorial_Optimization` from the list. Click **Next**.



- 4 Match the variables as above. Click **Next**.



- 5 Match the Torque model to the tuttq CAGE model as above. Click **Next**.



- 6 Match the operating point set SpeedLoadPoints to New_Dataset as above. Click **Finish**.

- 7 Run the optimization and view the results. The output data matrix should resemble the following. Note that the optimal values for A and SPK are very similar to those from the original algorithm.

Optimization Output					
	L	N	A	spark	tuttq
1	0.2	1000	12.78	23.768	9.07
2	0.3	1000	12.78	18.179	20.319
3	0.4	1000	12.78	14.261	31.567
4	0.5	1000	12.78	12.014	42.816
5	0.6	1000	12.78	11.439	54.064
6	0.7	1000	12.78	12.535	65.313
7	0.2	2000	12.78	27.477	9.742
8	0.3	2000	12.78	21.887	20.99
9	0.4	2000	12.78	17.969	32.238
10	0.5	2000	12.78	15.722	43.487
11	0.6	2000	12.78	15.147	54.735
12	0.7	2000	12.78	16.243	65.984
13	0.2	3000	12.779	31.202	9.342
14	0.3	3000	12.78	25.596	20.591
15	0.4	3000	12.78	21.676	31.839
16	0.5	3000	12.78	19.43	43.087
17	0.6	3000	12.78	18.855	54.336
18	0.7	3000	12.78	19.951	65.584
19	0.2	4000	12.78	34.893	7.872
20	0.3	4000	12.78	29.309	19.12
21	0.4	4000	12.78	25.385	30.369
22	0.5	4000	12.78	23.138	41.617
23	0.6	4000	12.78	22.566	52.866
24	0.7	4000	12.781	23.657	64.114
25	0.2	5000	12.78	38.601	5.331
26	0.3	5000	12.78	33.012	16.58
27	0.4	5000	12.78	29.093	27.828
28	0.5	5000	12.78	26.85	39.077
29	0.6	5000	12.78	26.271	50.325
30	0.7	5000	12.78	27.372	61.574
31	0.2	6000	12.78	42.309	1.72
32	0.3	6000	12.78	36.72	12.969
33	0.4	6000	12.78	32.801	24.217
34	0.5	6000	12.78	30.558	35.465
35	0.6	6000	12.78	29.979	46.714
36	0.7	6000	12.78	31.075	57.962

Using CAGE

This section includes the following topics:

- | | |
|---|--|
| How to Use CAGE (p. 7-2) | This introduction is an overview of CAGE processes: where to find the functionality for different processes and how to set up variables and models before performing calibrations. |
| Setting Up Your Variable Items (p. 7-6) | Using the Variable Dictionary view. |
| Setting Up Your Models (p. 7-13) | Using models by importing, renaming, and editing models, and creating new function models. |
| Exporting Calibrations (p. 7-21) | How to export your calibrations. |
| Specifying Locations of Files (p. 7-22) | How to use file preferences in CAGE. |

How to Use CAGE

The following reference sections describe how to use CAGE to perform calibrations:

- This section is an overview of CAGE processes: where to find the functionality for different processes and how to set up variables and models before performing calibrations.
- “Normalizers” on page 8-1 describes what normalizers are, and how to space breakpoints in a normalizer.
- “Feature Calibrations” on page 9-1 describes how to calibrate lookup tables by reference to models built using the model browser.
- “Tradeoff Calibrations” on page 10-1 describes how to calibrate lookup tables by adjusting one value to fulfill different objectives.
- “Optimization in CAGE” on page 11-1 describes how to set up and run optimizations, including automated tradeoffs.
- “Data Sets” on page 12-1 describes how to use CAGE to compare calibrations to experimental data, and how to use experimental data to fill lookup tables.
- “Calibration Manager” on page 13-1 describes how to use the **Calibration Manager**.
- “Surface Viewer” on page 14-1 describes how to use the **Surface Viewer**.
- “Manual Calibration and the History Display” on page 15-1 describes how to add and delete tables and manually calibrate tables, and how to use the **History** viewer.

Overview of CAGE Processes

Before you can perform a calibration using CAGE, you need to set up the variables, constants, and the models you want to use. If you import a model, it has variables associated with it, in which case you might not have to import a variable dictionary.

The following sections describe how to set up variables and models before performing calibrations:

- “Setting Up Your Variable Items” on page 7-6
- “Setting Up Your Models” on page 7-13

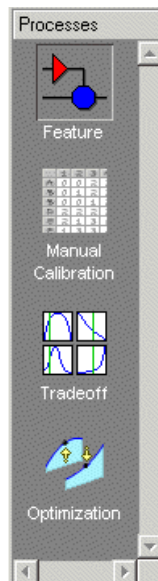
You can also use CAGE to calibrate tables directly from experimental data by interpolation, without using models. See “Tutorial: Filling Tables from Data” on page 5-1 for an example.

The view and functionality available in CAGE depend on two things:

- Which of the seven large buttons you select in the **Processes** and **Data Objects** panes
- The item you highlight in the branch display (tree)

See the next section, “CAGE Views and Processes” on page 7-3, for a summary of the functionality you can reach in each view and links to in-depth help for each process.

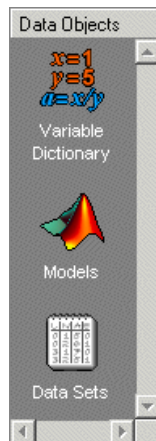
CAGE Views and Processes



The **Processes** pane has four buttons:

- **Feature** shows the **Feature** view, with the tables and strategies that are associated with that feature. See “Feature View” on page 9-44.
A feature is a strategy (or collection of tables) and a model used to calibrate those tables. In the **Feature** view, you can fill tables by comparing a strategy to a model. See “Feature Calibrations” on page 9-1. You can import existing strategies or construct new ones using Simulink from the feature view.
From the feature node in the branch display, you can access the **Surface Viewer** to examine the strategy or model or both. See “Surface Viewer” on page 14-1.
- **Manual Calibration** enables you to calibrate tables manually. It also acts as a store for all the tables and normalizers in your session. Here you can add and delete tables from the project. From any table display (here, or in other views) you can access the **History Display** to manage changes in your tables and normalizers. You can use the **History Display** to reverse changes. See “Using the History Display” on page 15-5.
- **Tradeoff** shows the **Tradeoff** view, with a list of the tables and models to display. Here you can see graphically the effects of manually altering variables to trade off different objectives (such as maximizing torque while minimizing emissions). At the tradeoff node, you can calibrate table values to achieve the best compromise between competing objectives. You can calibrate using single or multimodel tradeoffs. See “Tradeoff Calibrations” on page 10-1. You can also use the optimization functionality of CAGE to run automated tradeoffs, described in the Optimization section (see below).
- **Optimization** shows the **Optimization** view. From here you can set up and run optimizations, including automated tradeoffs. There are standard routines available and also templates provided so you can write your own optimization routines. You can find full instructions in “Optimization in CAGE” on page 11-1.

You can reach the **Calibration Manager** from all the **Process** views except **Optimization** (**Feature**, **Manual Calibration**, and **Tradeoff**). In the **Calibration Manager** you can set up the size of tables (manually or using existing calibration files) and edit the precision used for values (to match the kind of electronic control unit you are going to use). See “Calibration Manager” on page 13-1.



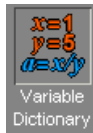
The **Data Objects** pane has three buttons:

- **Variable Dictionary** stores all the variables, constants, and formulas in your session. Here you can view, add, and edit any variables in any part of your session. See “Setting Up Your Variable Items” on page 7-6.
- **Models** stores all the models in your session. Here you can view a graphical display of these models, including a diagram of the model’s input structure. This is useful because a model can have other models as inputs. You can change the inputs here. For example, you can change your model’s input Spark to be connected to a model for Spark rather than to the variable Spark. You can also access the surface viewer here to examine models. See “Setting Up Your Models” on page 7-13 and “Surface Viewer” on page 14-1.
- **Data Sets** enables you to evaluate your models and features over a custom set of input values. Here you can create and edit a set of input values and view several models or features evaluated at these points. You can compare your tables and models with experimental data to validate your calibrations. You can also fill tables directly from experimental data by loading the experimental data as a new data set. See “Data Sets” on page 12-1.

Setting Up Your Variable Items

The **Variable Dictionary** is a store for all the variables, constants, and formulas in your session.

To view or edit the items in the **Variable Dictionary**, click the button, shown, in the **Data Objects** pane.



Selecting the **Variable Dictionary** view displays the variables, constants, and formulas in the current project.

This section describes the following:

- “Importing and Exporting a Variable Dictionary” on page 7-8
- “Adding Variable Items” on page 7-9
- “Using the Variable Menu” on page 7-11
- “Using Aliases” on page 7-12

Following is an example of the **Variable Dictionary** view.

List of all the constants, variables, and formulas in the project

The screenshot shows the CAGE Browser software interface. The main window is titled "CAGE Browser - Untitled" and has a menu bar with "File", "Edit", "Variable", "Tools", "Window", and "Help". Below the menu bar is a toolbar with various icons. The interface is divided into several sections:

- Processes:** A vertical sidebar on the left containing icons for "Feature", "Manual Calibration", and "Tradeoff".
- Data Objects:** A vertical sidebar on the left containing icons for "Variable Dictionary", "Models", and "Data Sets".
- Variable Dictionary:** A table listing project variables and constants. The table has columns for Name, Type, Alias, Minimum, Maximum, Set Point, and Formula.
- Edit Settings:** A panel on the right showing the configuration for the selected variable "afr, AFR". It includes input fields for Alias, Description, Minimum, Maximum, and Set Point, and a Formula field.

The Variable Dictionary table is as follows:

Name	Type	Alias	Minimum	Maximum	Set Point	Formula
x N	Variable	engine_speed	500	6500	2500	
x L	Variable	load, Load	0.1	1	0.4	
x A	Variable	afr, AFR	11	17	14.35	
k stoich	Constant				14.35	
x SPK	Variable	S, s, spark	-10	60	22.5	
f(x) lambda	Formula		0.75	1.25	1	A/stoich

The edit settings for the selected variable "afr, AFR" are:

- Alias: afr, AFR
- Description: Air-fuel ratio (ratio)
- Minimum: 11
- Maximum: 17
- Set Point: 14.35
- Formula: (empty)

Edit boxes to change the settings of the selected constant, variable, or formula

The upper pane shows a list of all the current variables, constants, and formulas. The lower pane displays edit boxes so you can specify the settings of the selected variable, constant, or formula.

Importing and Exporting a Variable Dictionary

A variable dictionary contains all the variable items for your calibrations. You can set up your variable dictionary once, and use it in many calibrations.

Importing a Variable Dictionary

To import a dictionary of variables from an .xml file,

- 1 Select **File -> Import -> Variable Dictionary**.
- 2 Select the correct dictionary.

Exporting a Variable Dictionary

After setting up a variable dictionary, you can save the dictionary for use in many different calibrations.

To export a dictionary of variables to an .xml file,

- 1 Select **File -> Export -> Variable Dictionary**.
- 2 Select a suitable name for the dictionary.

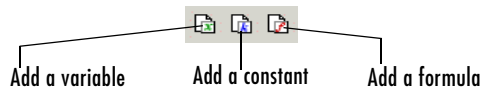
See Also

- “Setting Up Your Variable Items” on page 7-6
- “Adding Variable Items” on page 7-9

Adding Variable Items

To add variable items you can use the **Variable Dictionary** toolbar, shown, or you can select items from the **File -> New -> Variable Items** menu.

Variable Dictionary Toolbar



Adding a Variable

To add a variable,

- 1 Select **File -> New -> Variable Item -> Variable**.

A new variable is added to the variable dictionary.

- 2 Select **Edit -> Rename** to alter the name of the variable.

- 3 Specify the **Minimum** and **Maximum** values of the variable in the edit boxes in the lower pane.

- 4 Specify the value of the **Set Point** in the edit box.

Using Set Points in the Variable Dictionary. The set point of a variable is a point that is of particular interest in the range of the variable.

For example, for the air/fuel ratio variable, AFR, the range of values is typically 11 to 17. However, whenever only one value of AFR is required, it is preferable to choose 14.3, the stoichiometric constant, over any other value. So enter 14.3 as the **Set Point**.

CAGE uses the set point as the default value of the variable wherever one value from the variable range is required. For instance, CAGE uses the set point when evaluating a model over the range of a different variable.

For example, a simple model for torque depends on AFR, engine speed, and relative air charge. CAGE uses the set point of AFR when it calculates the values of the model over the ranges of the engine speed and relative air charge.

Adding a Constant

To add a constant,

- 1** Select **File** -> **New** -> **Variable Item** -> **Constant**.

A new constant is added to the variable dictionary.

- 2** Select **Edit** -> **Rename** to alter the name of the constant.

- 3** Specify the value of the constant in the **Set Point** edit box, in the lower pane.

Adding Formulas

You might want to add a formula to your session. For example, the formula

$$\lambda = \frac{\text{afr}}{\text{stoich}}$$

where afr is the air/fuel ratio and stoich is the stoichiometric constant.

To add a formula,

- 1** Select **File** -> **New** -> **Variable Item** -> **Formula**.

The **Add Formula** dialog box appears.

- 2** In the dialog, enter the right side of the formula, afr/stoich, and click **OK**.

A new formula is added to the variable dictionary.

- 3** Select **Edit** -> **Rename** to alter the name of the formula.

Note Formulas can only have one variable.

See Also

- “Setting Up Your Variable Items” on page 7-6
- “Adding Variable Items” on page 7-9

Using the Variable Menu

The **Variable** menu in the variable dictionary enables you to alter variable items.

Change item to

- **Alias**

Changes the selected item to be an alias of another item in the current project. For example, if you have two variables, `engine_speed` and `n`, you can change `n` to be an alias of `engine_speed`, with its maximum and minimum values. For more information, see the next section, “Using Aliases” on page 7-12.

- **Formula**

Changes a variable or constant into a formula. You have to define the right side of the formula, and use the edit boxes to change the ranges.

- **Constant**

Changes a variable or formula into a constant. The value of the constant is by default the midpoint of the variable’s range.

- **Variable**

Changes a constant or formula into a variable. The minimum value of the variable is, by default, the value of the constant, and its maximum is, by default, twice the minimum value.

See Also

- “Setting Up Your Variable Items” on page 7-6
- “Using Aliases” on page 7-12

Using Aliases

The variable dictionary enables you to use the same set of variables, constants, and formulas with many different models and calibrations.

Why Use Aliases?

It is possible that in one model the engine speed has been defined as N, and in another it has been defined as rpm. The alias function enables you to refer to the same quantity by a variety of different names.

Creating an Alias

For example, in a variable dictionary there are two variables:

- N, with a range of 500 to 6500
- rpm, with a range of 2500 to 3500

To set rpm to be an alias of N,

- 1 Highlight the variable rpm.
- 2 Select **Variable** → **Change item to** → **Alias**.
- 3 In the dialog, choose N from the list.

This eliminates the variable rpm from your variable dictionary, and every model and calibration that refers to rpm now refers to N instead.

Note If N is made an alias of rpm in the preceding example, the range of N is restricted to the range of rpm, 2500 to 3500.

See Also

- “Setting Up Your Variable Items” on page 7-6

Setting Up Your Models

CAGE generally calibrates lookup tables by reference to models. The **Models** view is a storage place for all the models in your session.

To view and edit the models in your session, select **Models** by clicking the button shown in the **Data Objects** pane.



This section describes the following:

- “Importing Models” on page 7-15
- “Adding New Function Models” on page 7-17
- “Renaming and Editing Models” on page 7-18

The **Models** view displays the following:

- A list of all the models in the current project.
- The model connections. That is, which constants, variables, and models are linked to the selected model.
- An image of the response surface of the selected model.

Following is an example of the **Models** display.

List of the current models

The screenshot shows the CAGE Browser software interface. At the top is a menu bar with 'File', 'Edit', 'Model', 'Tools', 'Window', and 'Help'. Below the menu bar is a toolbar with icons for file operations and help. The main interface is divided into several panels:

- Processes:** A vertical sidebar on the left containing icons for 'Feature', 'Manual Calibration', and 'Tradeoff'.
- Models:** A table listing the current models in the workspace.

Name	Type	Inputs	Description
tq	MBC model	spk, N, L, A	
MBT_Spark	MBC model	L, N, A, E	
Spark	Function model	MBT_Spark	min(30, MBT_Spark)
- Connections:** A diagram showing the flow of data between models. Four red ovals labeled 'N', 'L', 'A', and 'E' have arrows pointing to a blue rectangular box labeled 'MBT_Spark'. An arrow then points from 'MBT_Spark' to another blue rectangular box labeled 'Spark'.
- Spark:** A 3D surface plot showing a curved surface. The vertical axis (z-axis) ranges from 10 to 30. The horizontal axes are labeled 'L' and 'N'. The 'L' axis ranges from 0.2 to 0.8, and the 'N' axis ranges from 1000 to 3000. Below the plot are two dropdown menus: 'x-axis factor:' with 'N' selected, and 'y-axis factor:' with 'L' selected.

Model connections display

Model display


Importing Models

CAGE enables you to calibrate lookup tables by referring to models constructed in the Model Browser.

To import a Model Browser model,

1 Select **File -> Import -> Model**.

This opens the **Model Import Wizard**. The following steps take you through the three screens of the wizard.

2 Select the correct file by clicking  to browse for the correct file.

3 CAGE can only open Model Browser files.

- a** If the model is saved as an `.exm` file, select **Automatic** from the **Open As** drop-down menu.
- b** If the model is not saved as an `.exm` file, select **MBC Model** from the **Open As** drop-down menu. For example, the file extension might be accidentally changed.

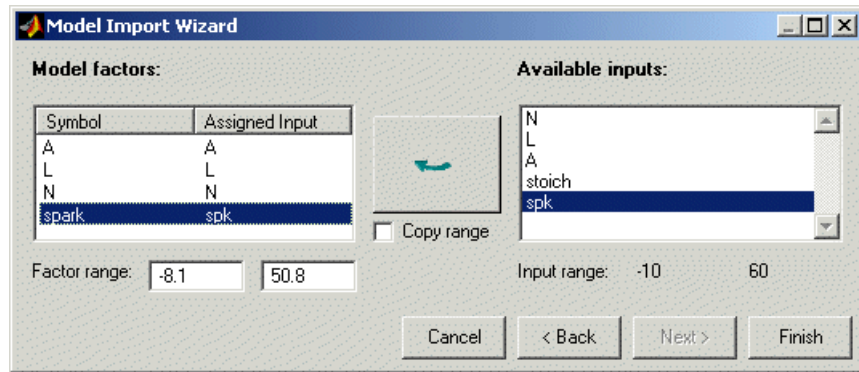
4 Click **Next >** to select the model file.

5 Select the models that you want to import by highlighting the models from the list.

6 Click **Next >** to select the models.

7 Associate the model factors with the available inputs in your session.

For example, to associate the model factor `spark` with the variable `spk` in your session,



- a Highlight a model factor, spark, in the list on the left and the corresponding variable, spk, in the list on the right.
- b Click the select input button, shown.



- c Repeat 7a and 7b for all the model factors.
- 8 Click **Finish** to close the wizard and return you to the **Models** view.

Note You can skip steps 6 and 7 by selecting the **Automatically assign/create inputs** box at step 5.

You can now see a display of the model surface and the model connections.

See Also

- “Setting Up Your Models” on page 7-13
- “Adding New Function Models” on page 7-17
- “Renaming and Editing Models” on page 7-18

Adding New Function Models

A function model is a model that is expressed algebraically.

For example, you might want to view the behavior of torque efficiency. So you create a function model of torque efficiency = torque/peak torque.

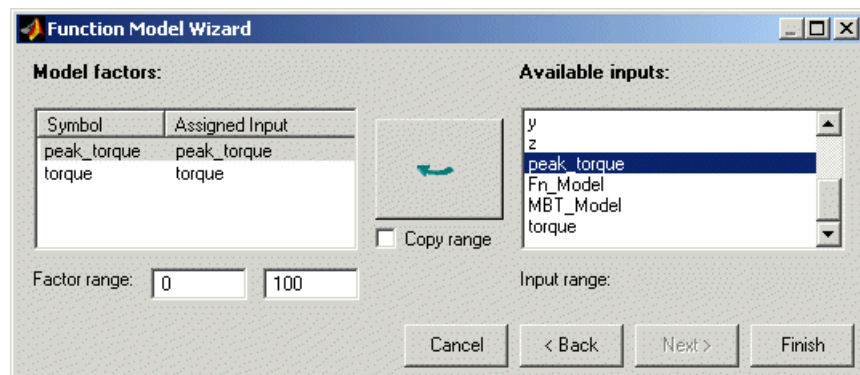
To add a function model to your session,

- 1 Select **File** -> **New** -> **Function Model**.

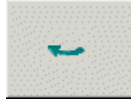
This opens the **Function Model Wizard**.

- 2 In the dialog box, enter the formula for your function model. For example, enter torque_efficiency=torque/peak_torque.
- 3 Press **Enter**. CAGE checks that the function is recognized; if so, you can click **Next** >. If the function is incorrectly entered, you cannot click **Next**.
- 4 Select the models that you want to import by highlighting the models from the list.
- 5 Click **Next** > to select the models.
- 6 Associate the model factors with the available inputs in your session.

For example, to associate the model factor peak_torque with the peak_torque model in your session,



- a Highlight a model factor, `peak_torque`, in the list on the left and the corresponding model, `peak_torque`, in the list on the right.
- b Click the select input button, shown.



- c Repeat 6a and 7b for all the model factors.
- 7 Click **Finish** to close the wizard and return you to the **Models** view.

Note You can skip steps 5 and 6 by selecting the **Automatically assign/create inputs** box at step 4.

You can now see a display of the model and its connections.

See Also

- “Setting Up Your Models” on page 7-13
- “Importing Models” on page 7-15
- “Renaming and Editing Models” on page 7-18

Renaming and Editing Models

Renaming Models

To rename a model,

- 1 Highlight the model that you want to rename.
- 2 Select **Edit** -> **Rename**.
- 3 Enter the new name for the model and press **Return**.

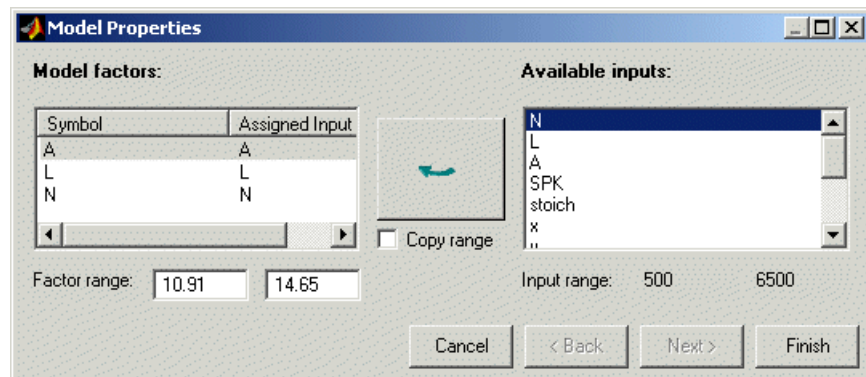
You can also rename the model by selecting a model and clicking the name.

Editing Ranges of Model Factors

To edit the ranges of the model factors in the currently selected model,

- 1 Highlight the model that you want to edit.
- 2 To change the ranges of the model factors, select **Model → Properties**.

This opens the **Model Properties** dialog box, shown.



- 3 Highlight the model factor that you want to alter.
- 4 Enter the new range in the **Factor range** boxes.

Note This does not change the range of the variable over the entire project; rather, it changes the range of the variable in the selected model. If you want to change the range of a variable in the entire session, change the range in the variable dictionary. For more information, see “Using the Variable Menu” on page 7-11.

Editing Model Connections

You can adjust a model so that variables, formulas, or other models are the factors of the model. For example, a model of torque depends on the spark angle. In place of the spark angle, you can have a model of the maximum brake torque (MBT).

To edit the connections of a model,

- 1 Highlight the model.
- 2 Select **Model -> Properties**.

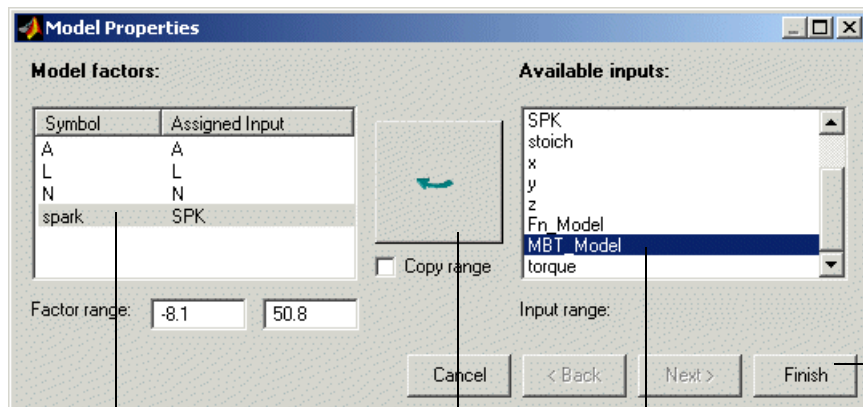
This opens the **Model Properties** dialog box.

- 3 Highlight the model factor that you want to adjust, in the list on the left.
- 4 Highlight the new input for that factor, in the list on the right.
- 5 Click the **Select Input** button, shown.



- 6 To close the dialog box, click **Finish**.

Example of Editing the Connections of the Model



3. Highlight the model factor that you want to change.

5. Click **Select Input**.

4. Highlight the new input.

6. Click **Finish**.

Exporting Calibrations

When you have filled some tables using any of the CAGE processes, you can export the tables.

- 1 Select **File -> Export -> Calibration**.
- 2 Choose the type of file you want to save your calibrations as. You can choose from
 - a **Comma Separated Value (.csv)**
 - b **MATLAB-file (.mat)**
 - c **M-file script**
- 3 Enter the filename and click **Save**.

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** or **Tradeoff** node outputs the whole feature or tradeoff (all tables and nodes).

Selecting a branch node outputs all the **Features** or **Tradeoffs** under the branch.

Specifying Locations of Files

You can specify preferred locations of project and data files, using **File -> Preferences**.

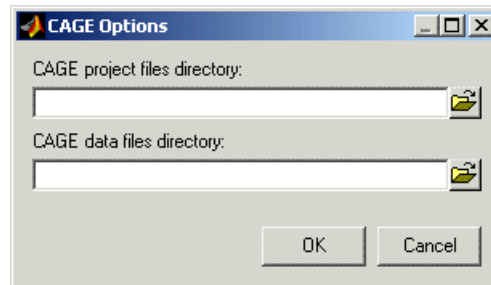
Project files have the file extension `.cag` and store entire CAGE sessions.


Data files are the files that form part of the CAGE session. For example, the following is a list of some of the data files used in CAGE:

- Simulink diagrams (`.mdl`)
- Experimental data (`.xls`, `.csv`, or `.mat`)
- Variable dictionaries (`.xml`)
- Models (`.exm`)

To specify preferred locations for project and data files,

- 1 Select **File -> Preferences**. This opens the dialog box shown.



- 2 Enter the directory where your CAGE project files are stored. Alternatively, click  to browse for the directory.
- 3 Enter or browse for the directory where your data files are stored.
- 4 Click **OK**.

Normalizers

This section includes the following topics:

About Normalizers (p. 8-2)

What are normalizers? A normalizer is the axis of your lookup table. It is the same as the collection of the breakpoints in your table.

Calibrating the Normalizers (p. 8-3)

This section describes how to calibrate the normalizers by spacing the breakpoints. This covers initializing, filling, and optimizing breakpoints.

Normalizer View (p. 8-13)

This section describes what you can see when you highlight a normalizer in the branch display: the input/output display, normalizer display, breakpoint spacing display, how to delete breakpoints, and how to use the comparison pane.

About Normalizers

CAGE distinguishes between the normalizers and the tables that they belong to.

Using models to calibrate lookup tables enables you to perform analysis of the models to determine where to place the breakpoints in a normalizer. This is a very powerful analytical process.

It is important to stress that in CAGE a lookup table can be either one-dimensional or two dimensional. One-dimensional tables are sometimes known as characteristic lines or functions. Two-dimensional tables are also known as characteristic maps or tables. This is important because normalizers are very similar to characteristic lines.

For example, a simple strategy to calibrate the behavior of torque in an engine might have a two-dimensional table in speed and relative air charge (a measure of the load). Additionally, this strategy might take into account the factors of air/fuel ratio (AFR) and spark angle. Each of these compensating factors is accounted for by the use of a simple characteristic line. In CAGE, these characteristic lines are one-dimensional tables. In the example strategy, there are the following tables and normalizers:

- One characteristic map: the torque table
- Six characteristic lines:
 - Two tables: one for AFR and one for spark angle
 - Four normalizer functions: speed, load, AFR, and spark angle

Notice also that a breakpoint is a point on the normalizer where you set values for the lookup table.

Thus, when you *calibrate a normalizer* you place the individual breakpoints over the range of the table's axis.

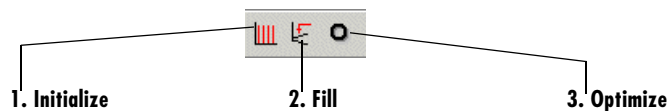
Calibrating the Normalizers

Select a normalizer in the branch display. This enables you to calibrate the normalizer, and it displays the **Normalizer** view.

For more information about the **Normalizer** view, see “Normalizer View” on page 8-13.

This section describes how you can use CAGE to space the breakpoints over the range of the normalizers.

Normalizer Toolbar



To space the breakpoints, either click the buttons on the toolbar or select from the following options on the **Normalizer** menu:

- **Initialize**

This spaces the breakpoints evenly along the normalizer. For more information, see “Initializing Breakpoints” on page 8-4.

- **Fill**

This spaces the breakpoints by reference to the model. For example, you can place more breakpoints where the model curvature is greatest. For more information, see “Filling Breakpoints” on page 8-5.

- **Optimize**

This moves the breakpoints to minimize the least square error over the range of the axis. For more information, see “Optimizing Breakpoints” on page 8-9.


The next sections describe each of these in detail.

Initializing Breakpoints

Initializing the breakpoints places the breakpoints at even intervals along the range of the variable defined for the normalizer.

For example, a torque table has two normalizers, engine speed and relative air charge, or load. You can evenly space the breakpoints of both normalizers over the range 500 rpm to 6500 rpm for speed and 0.1 to 1 for the relative air charge.

To space the breakpoints evenly,

- 1 Click  on the toolbar or select **Normalizer -> Initialize**.
- 2 In the dialog box, enter the range of values for the normalizer.

In the preceding example, for the speed normalizer, N, enter 500 6500, and for the load normalizer, L, enter 0.1 1.

- 3 Click **OK**.

Note If the selected table has two normalizers, both are evenly spaced automatically.

Filling Breakpoints

Filling breakpoints spaces the breakpoints in such a way as to place the breakpoints by reference to the model. For example, one method places the majority of the breakpoints where the curvature of the model is greatest.

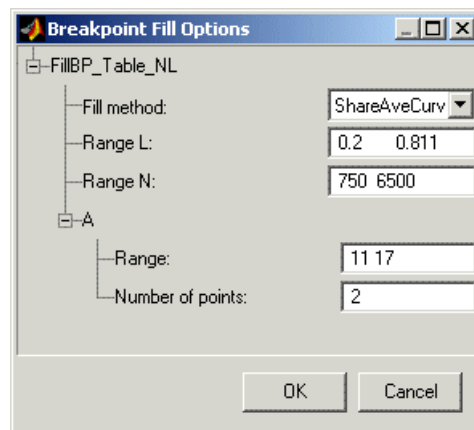
This option is only available when you are performing **Feature** calibrations.

For example, a model of the spark angle that produces the maximum brake torque (MBT) has the following inputs: engine speed N , relative air charge L , and air/fuel ratio A . You can space the breakpoints for engine speed and relative air charge over the range of these variables by referring to the model.

To space the breakpoints based on model curvature,

- 1 Click  or select **Normalizer -> Fill**.

The **Breakpoint Fill Options** dialog box opens.



- 2 Choose the appropriate method to space your breakpoints, from the drop-down menu in the dialog box.

For the preceding example, select **ShareAveCurv**. For more information about the methods for spacing the breakpoints, see “Filling Methods” on page 8-6.

3 Enter the ranges of the values for the normalizers.

In the preceding example, for **Range N**, enter 500 6500, and for **Range L**, enter 0.1 1.

4 Enter the ranges of the other model variables.

CAGE spaces the breakpoints by reference to the model. It does this at selected points of the other model variables. In the preceding example, enter 11 17 for the **Range of A** and enter 2 for the **Number of points**. This takes two slices through the model at $A = 11$ and $A = 17$. Each slice is a surface in N and L . That is, $MBT(N, L, 11)$ and $MBT(N, L, 17)$.

CAGE computes the average value of these two surfaces to give an average model $MBT_{AV}(N, L)$.

5 Click **OK**.

Note If any of the breakpoints is locked, each group of unlocked breakpoints is independently spaced according to the selected algorithm.

If you increase the number of slices through the model, you increase the computing time required to calculate where to place the breakpoints.

Filling Methods

This section describes in detail the methods for spacing the breakpoints of your normalizers in CAGE.

- For one-dimensional tables, the two fill methods are
 - ReduceError
 - ShareAveCurv
- For two-dimensional tables, the two fill methods are
 - ShareAveCurv
 - ShareCurvThenAve

ReduceError

Spacing breakpoints using ReduceError uses a greedy algorithm:

- 1 CAGE locks two breakpoints at the extremities of the range of values.
- 2 Then CAGE interpolates the function between these two breakpoints.
- 3 CAGE calculates the maximum error between the model and the interpolated function.
- 4 CAGE places a breakpoint where the error is maximum.
- 5 Steps 2, 3, and 4 are repeated.
- 6 The algorithm ends when CAGE locates all the breakpoints.

ShareAveCurv and ShareCurvThenAve

Consider calibrating the normalizers for speed, N , and relative air-charge, L , in the preceding MBT model.

In both cases, CAGE approximates the $MBT_{AV}(N, L)$ model, in this case using a fine mesh.

The breakpoints of each normalizer are calibrated in turn. In this example, these routines calibrate the normalizer in N first.

Spacing breakpoints using ShareAveCurv or ShareCurvThenAve calculates the curvature, K , of the model $MBT_{AV}(N, L)$,

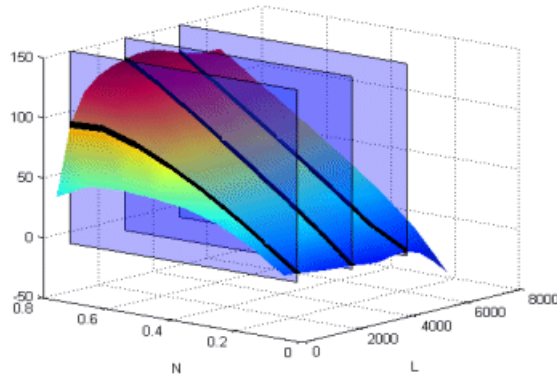
$$K = \sum_{i=1}^{\text{fine mesh}} (MBT_{AV}''(N, L))^{1/2}$$

as an approximation for

$$K = \int_{750}^{6000} |MBT_{AV}''(N, L)|^{1/2} dN$$

Both routines calculate the curvature for a number of slices of the model at various values of L . For example, the figure shown has a number of slices of a model at various values of L .

Model Slices at Various Values of L



Then

- ShareAveCurv averages the curvature over the range of N , then spaces the breakpoints by placing the i^{th} breakpoint according to the following rule.
- ShareCurvThenAve places the i^{th} breakpoint according to the rule, then finds the average position of each breakpoint.

Rule for Placing Breakpoints. If j breakpoints need to be placed, the i^{th} breakpoint, N_i , is placed where the average curvature so far is

$$\int_0^{N_i} |MBT_{AV}''(N, L)|^{1/2} dN = \frac{i-1}{j-1} \times K$$

Reference. de Boor, C., *A Practical Guide to Splines*, New York, Springer-Verlag, 1978.

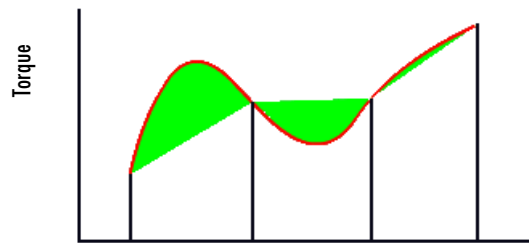
See Also

- “Calibrating the Normalizers” on page 8-3

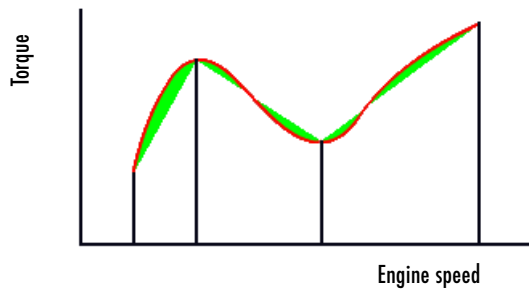
Optimizing Breakpoints

Optimizing breakpoints alters the position of the table normalizers so that the total square error between the model and the table is reduced.

This routine improves the fit between your strategy and your model. The following illustration shows how the optimization of breakpoint positions can reduce the difference between the model and the table. The breakpoints are moved to reduce the peak error between breakpoints. In CAGE this happens in two dimensions across a table.



The green shaded areas show the error between the interpolated table values and the model using the initial breakpoints.




Optimizing the position of the breakpoints can greatly reduce the error between the interpolated table values and the model.

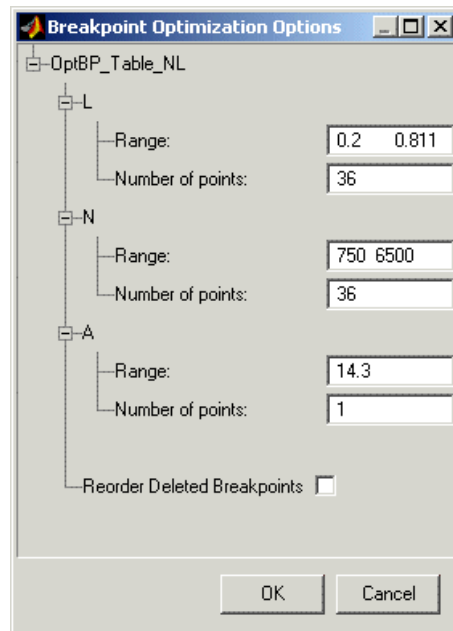
To see the difference between optimizing breakpoints and optimizing table values, compare with the illustration in “Optimizing Table Values” on page 9-16.

For an example of breakpoint optimization, say you have a model of the spark angle that produces the MBT (maximum brake torque). The model has the following inputs: engine speed, N , relative air charge, L , and air/fuel ratio, A . You can optimize the breakpoints for N and L over the ranges of these variables.

To optimize the breakpoints,

- 1 Ensure that the optimization routine works over reasonable values for the table by choosing one of these methods:
 - a Select **Normalizer** → **Initialize**.
 - b Select **Normalizer** → **Fill**.
- 2 Click  on the toolbar or select **Normalizer** → **Optimize**.

This opens the following dialog box.



3 Enter the ranges for the normalizers.

For the preceding example, enter 0.2 0.811 for the **Range of L**, and enter 750 6500 for **N**.

4 Enter the appropriate number of grid points for the optimization.

This defines a grid over which the optimization works. In the preceding example, the number of grid points is 36 for both L and N . This mesh is combined using cubic splines to approximate the model.

5 Enter ranges and numbers of points for the other model variables.

In the preceding example, the **Range of A** is 14.3 and the **Number of points** is 1.

6 Decide whether or not to reorder deleted breakpoints, by clicking the radio button.

If you choose to reorder deleted breakpoints, the optimization process might redistribute them between other nondeleted breakpoints (if they are more useful in a different position).

For information about deleting breakpoints, see “Deleting Breakpoints” on page 8-18.

7 Click **OK**.

CAGE calculates the table filled with the mesh at the current breakpoints. Then CAGE calculates the total square error between the table values and the mesh model.

The breakpoints are adjusted until this error is minimized, using nonlinear least squares optimization (`lsqnonlin`).

When optimizing the breakpoints, it is worth noting the following:

- The default range for the normalizer variable is the range of the variable.
- The default value for all other model variables is the set point of the variable.
- The default number of grid points is three times the number of breakpoints.

See Also

- Reference page for `lsqnonlin`
- “Calibrating the Normalizers” on page 8-3

Normalizer View

The normalizer node shows the **Normalizer** view, which displays

- One normalizer if the table selected is one-dimensional
- Both normalizers if the table is two-dimensional

The table in the following example is two-dimensional.

Normalizer View

Selected node 1. Input output display 2. Normalizer display 3. Breakpoint spacing display

FN_N

Input	Output
1000	0
1333	1
1667	2
2000	3
2333	4
2667	5
3000	6
3333	7
3667	8
4000	9
4333	10
4667	11
5000	12

FN_LOAD

Input	Output
0.200	0
0.255	1
0.309	2
0.364	3
0.418	4
0.527	6
0.582	7
0.636	8
0.691	9
0.745	10
0.800	11

4. To view the comparison pane

Note If the table has two normalizers, both are displayed, the normalizer for the table columns at the top, the normalizer for the table rows below. This is true whichever normalizer on the tree is highlighted.

The parts of the display are

- 1 The **Input Output** display shows the breakpoints of the normalizer. For information, see “Input/Output Display” on page 8-15.
- 2 The **Normalizer Display** is a graphical representation of the **Input Output** display. For information, see “Normalizer Display” on page 8-16.
- 3 The **Breakpoint Spacing** display shows a slice of the model over the range of the breakpoints. For information, see “Breakpoint Spacing Display and Deleting Breakpoints” on page 8-17.
- 4 The comparison pane. For information, see “Viewing the Comparison Pane” on page 8-20.

The following sections describe in detail each part of the **Normalizer** view.

Input/Output Display

Input	Output
500	0
1055	1
1609	2
2164	3
2718	4
3273	5
3828	6
4332	7
4836	8
5391	9
5895	10
6500	11

The table consists of the breakpoints of the normalizer function.

The table has inputs and outputs:

- The inputs are the values of the breakpoints.
- The outputs refer to the row/column indices of the attached table.

To change values of the normalizers using the **Input Output** display, double-click a cell in the **Input** column and change its value.

Viewing the History of a Normalizer

To view the history of the normalizer function,

- 1 Right-click the table.
- 2 Select **Show History** from the menu.

This opens the **History** dialog box. For a more detailed description of the **History** dialog box, see “Using the History Display” on page 15-5.

Locking and Unlocking Breakpoints in the Input/Output Display

Locking breakpoints ensures that the locked breakpoint does not alter its position. You might want to lock a breakpoint when you are satisfied that it has the correct value.

To lock a breakpoint in the **Input/Output** display,

- 1 Right-click the selected breakpoint.
- 2 Select **Lock/Unlock** from the menu.

Locked breakpoints have red cells.

To unlock cells, follow the same procedure.

See Also

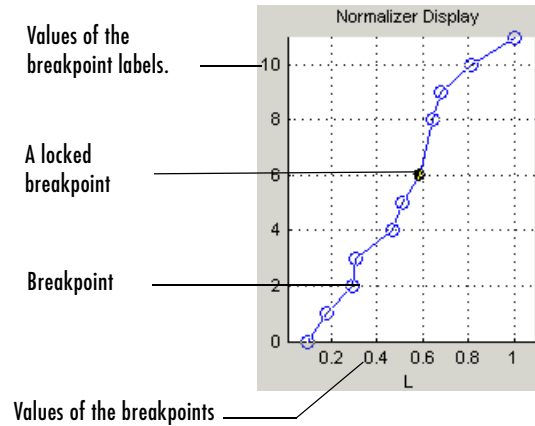
- “Normalizer View” on page 8-13

Normalizer Display

This displays the values of the breakpoints plotted against the marker numbers of the table (that is, the inputs against the outputs).

Click and drag the breakpoints to move them.

Example of the Normalizer Display



Locking and Unlocking Breakpoints in the Normalizer Display

To lock a breakpoint in the **Normalizer Display**, right-click the selected breakpoint and select **Lock Breakpoint**. You might want to lock a breakpoint when you are satisfied that it has the correct value.

Locked breakpoints are colored black.

See Also

- “Normalizer View” on page 8-13

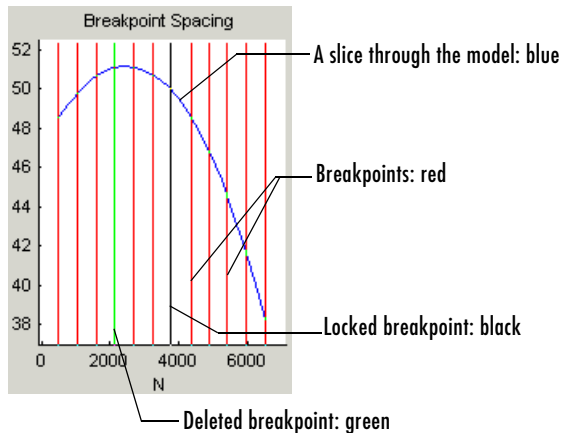
Breakpoint Spacing Display and Deleting Breakpoints

The **Breakpoint Spacing** display shows

- A slice through the model in blue
- The breakpoints in red

To move breakpoints, click and drag.

Example of the Breakpoint Spacing Display



Locking Breakpoints in the Breakpoint Spacing Display

You might want to lock a breakpoint when you are satisfied with its value.

To lock a breakpoint in the **Breakpoint Spacing** display, right-click a breakpoint and select **Lock Breakpoint** from the menu.

Locked breakpoints are colored black.

Deleting Breakpoints

Deleting breakpoints removes them from the normalizer table. There are still table values for the deleted breakpoints: CAGE determines the positions of the deleted breakpoints by spacing them linearly by interpolation between the nondeleted breakpoints.

Deleting breakpoints frees ECU memory. For example, a speed normalizer runs from 500 to 5500 rpm. Six breakpoints are spaced evenly over the range of speed, that is, at 500, 1500, 2500, 3500, 4500, and 5500 rpm. If you delete all the breakpoints except the endpoints, 500 and 5500 rpm, you reduce the amount stored in the ECU memory. The ECU calculates where to place the breakpoints by linearly spacing the breakpoints between the 500 rpm breakpoint and the 5500 rpm breakpoint.

To delete a breakpoint, right-click the breakpoint and select **Delete Breakpoint**.

Deleted breakpoints are green in the **Breakpoint Spacing** display.

Show the Model's Curvature

You might want to view the curvature of the model to manually move breakpoints to where the model's curvature is greatest.

To display the model slice as its second-order derivative, the curvature of the model,

1 Right-click the model in the **Breakpoint Spacing** display.

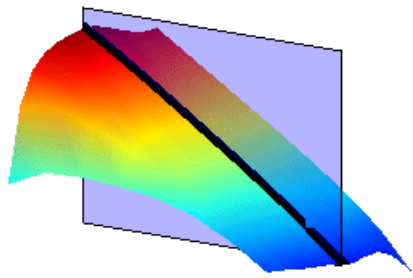
2 Select **Display -> Model Curvature**.

You can revert to displaying the model by selecting **Display -> Model** from the right-click menu.

Multiple Slice View

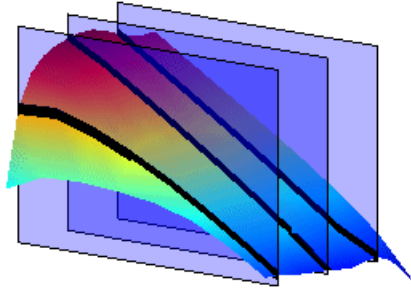
By default the **Breakpoint Spacing** display shows one slice through the model, shown.

Slice Through a Model Surface



Viewing many slices of the model gives a better impression of the curvature of the model. For example, see the following figure.

Many Slices Through a Model Surface




To view multiple slices through the model,

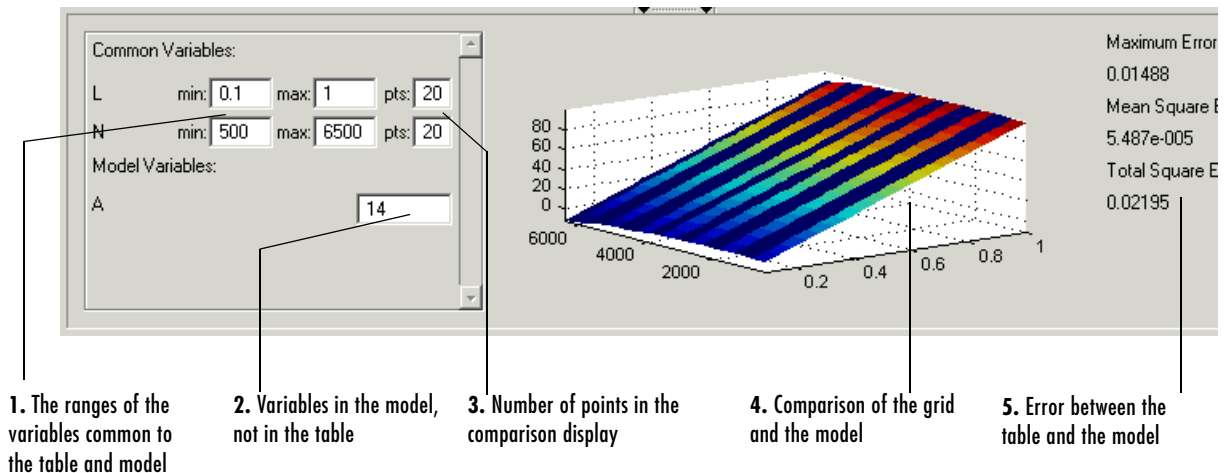
- 1 Right-click the model slice in the **Breakpoint Spacing** display.
- 2 From the menu, select **Number of Lines** and choose the number of slices that you want to view from the list.

See Also

- “Normalizer View” on page 8-13

Viewing the Comparison Pane

To view the comparison pane, select **View -> Comparison**. Alternatively, click , the “snapper point” at the bottom of the normalizer display panes. This is labeled in the diagram of the “Normalizer View” on page 8-13.



The comparison pane displays a comparison between the following:

- A full factorial grid filled using these breakpoints
- The model

Note This is not a comparison between the current table values and the model. To compare the current table values and the model, see “Calibrating the Tables” on page 9-11.

To make full use of the comparison pane,

- 1 Adjust the ranges of the variables that are common to the model and table.
- 2 Adjust the values selected for any variables in the model that are not in the selected table.

The default for this is the set point of the variable, as specified in the variable dictionary. For more information, see “Using Set Points in the Variable Dictionary” on page 7-9.

- 3 Check the number of points at which the display is calculated.

- 4 Check the comparison between the table and the model.

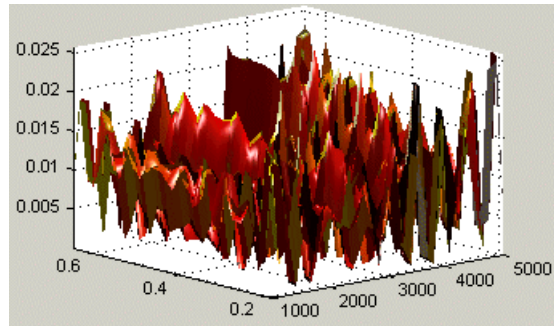
Right-click the comparison graph to view the error display.

- 5 Check some of the error statistics for the comparison, and use the comparison to locate where improvements can be made.

Error Display

The comparison pane can also be used to display the error between the model and the strategy.

Error Display in the Comparison Pane



To display the error,

- 1 Right-click the axes of the comparison display.
- 2 Select **Error** from the menu.

This changes the graph to display the error between the model and the strategy.

You can display the error data in one of the following ways:

- **Error**. This is the difference between the feature and the model.
- **Squared Error**. This is the error squared.
- **Absolute Error**. This is the absolute value of the error.
- **Relative Error (%)**. This is the error as a percentage of the value of the model.

- **Absolute Relative Error (%)**. This is the absolute value of the relative error.

To select one of these displays of the error data,

- 1 Right-click the display.
- 2 Select **Error Display** and select the appropriate display of the error from the context menu.

See Also

- “Normalizer View” on page 8-13
- “Comparing the Strategy and the Model” on page 9-33

This describes the comparison made when a table node is selected in the branch display.

Feature Calibrations

This section includes the following topics:

Performing Feature Calibrations (p. 9-2)	Introduction to feature calibrations and an overview of the processes involved.
Setting Up a Feature Calibration (p. 9-4)	How to add a new feature, assign a model, and set up your strategy and tables.
Calibrating the Tables (p. 9-11)	How to initialize, fill, optimize, and extrapolate your table values.
Table View (p. 9-29)	How to use the Table view.
Calibrating the Feature Node (p. 9-37)	How to calibrate a whole feature at once, rather than table by table.
Feature View (p. 9-44)	Functionality available in the Feature view.

Performing Feature Calibrations



A **Feature** calibration is the process of calibrating lookup tables and their normalizers by comparing a Simulink strategy to a model.

The Simulink strategy is an algebraic collection of lookup tables. It is used to estimate signals in the engine that cannot be measured and that are important for engine control.

CAGE calibrates an electronic control unit (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.

See “Tutorial: Feature Calibration” on page 2-1 for a tutorial showing how to perform a simple feature calibration.

The basic procedure for performing feature calibrations is as follows:

- 1 Set up the variables and constants. (See “Setting Up Your Variable Items” on page 7-6.)
- 2 Set up the model or models. (See “Setting Up Your Models” on page 7-13.)
- 3 Set up the feature calibration. (See “Setting Up a Feature Calibration” on page 9-4.)
- 4 Calibrate the normalizers. (See “Calibrating the Normalizers” on page 8-3.)
- 5 Calibrate the tables. (See “Calibrating the Tables” on page 9-11.)
- 6 Calibrate and view the entire feature. (See “Calibrating the Feature Node” on page 9-37.)

7 Export the normalizers, tables, and features. (See “Exporting Calibrations” on page 7-21.)

3. Set up the feature calibration.

7. Export the calibration.

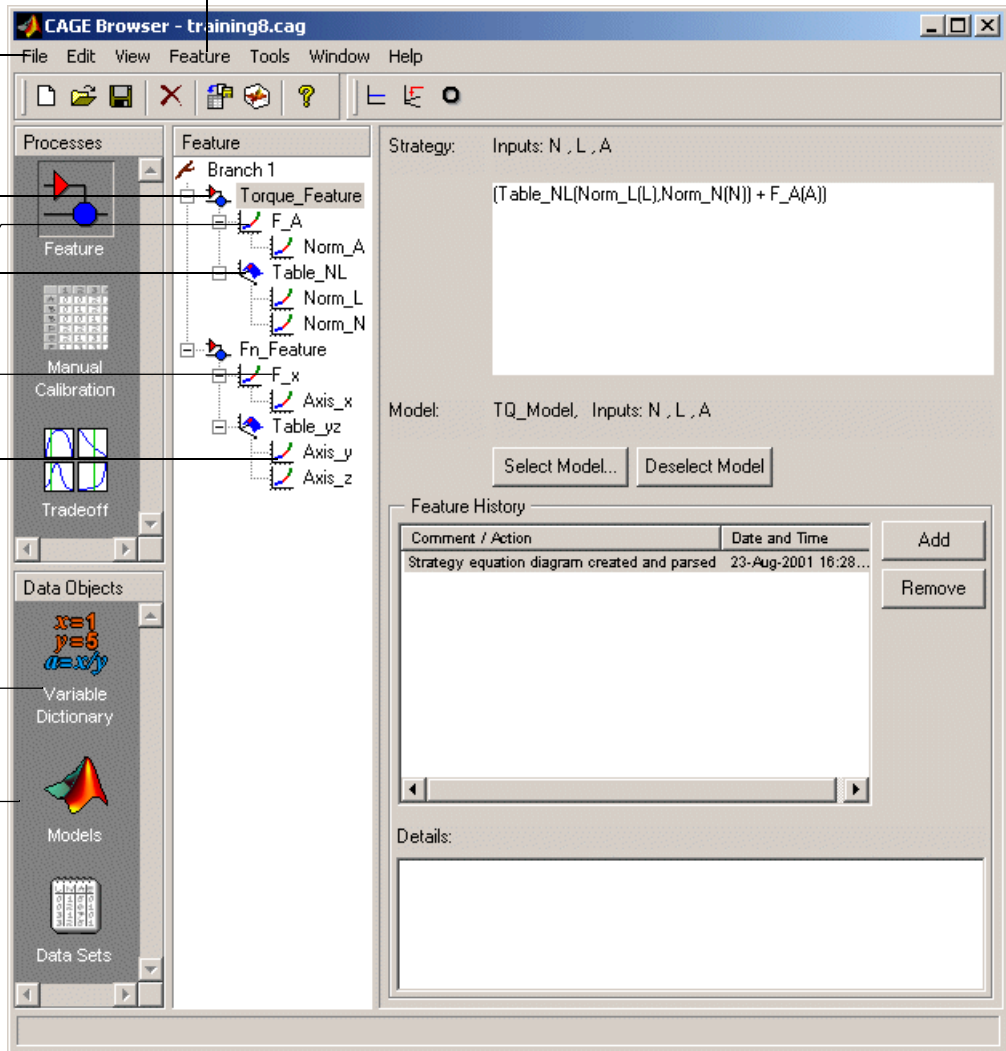
6. Calibrate the feature.

5. Calibrate the tables.

4. Calibrate the normalizers.

1. Set up the variables.

2. Set up the models.



The normalizers, tables, and features form a hierarchy of nodes, each with its own view and toolbar.

Setting Up a Feature Calibration

A feature calibration is the process of calibrating lookup tables and their normalizers by comparing a collection of lookup tables to a model. The collection of lookup tables is determined by a strategy.

A feature refers to the object that contains the model and the collection of lookup tables. For example, a simple feature for calibrating the lookup tables for the maximum brake torque (MBT) consists of

- A model of MBT
- A strategy that adds the two following tables:
 - A speed (N), load (L) table
 - A table to account for the behavior of the air/fuel ratio (A)

Having already set up your variable items and models, you can follow the procedure below to set up your feature calibration:

- 1** Add a feature. This is described in the next section, “Adding a Feature” on page 9-4.
- 2** Assign a model. This is described in “Assigning a Model” on page 9-5.
- 3** Set up your strategy. This is described in “Setting Up Your Strategy” on page 9-5.
- 4** Set up the tables. This is described later, in “Setting Up Tables” on page 13-2.



This section describes steps 1, 2, and 3 in turn.

When you have completed these four steps, you are ready to calibrate the normalizers, tables, and features.

Adding a Feature

A feature consists of a model and a collection of lookup tables, organized in a strategy.

To add a feature to your session, select **File -> New -> Feature**. This automatically switches you to the **Feature** view and adds an empty feature to your session.

An incomplete feature is a feature that does not contain both an assigned model and a strategy. If a feature is incomplete, it is displayed as  in the branch display. If a feature is complete, it is displayed as  in the branch display.

Assigning a Model

Having already added a feature and a model to your session, you can assign a model to your feature.

To assign a model to your feature,

- 1 Highlight the **Feature** node in the branch display.
- 2 Click **Select Model** to select the model you want to work with.

If there is only one model in your project, it is selected automatically.

If there is more than one model in your project, a dialog box appears. Highlight the correct model to assign to your feature and click **OK**.

Setting Up Your Strategy

A strategy is an algebraic collection of tables, and forms the structure of the feature.

For example, a simple strategy to calibrate a feature for MBT adds two tables:

- A table ranging over the variables speed and relative air charge
- A table to account for the behavior of the model as the AFR varies

To evaluate the feature side by side with the model, you need to have a strategy that takes some or all of the same variables as the model.

The strategy is expressed using Simulink diagrams.

You can either import a strategy or you can construct a strategy.

The following topics are described next:

- “Importing a Strategy” on page 9-6
- “Constructing a Strategy” on page 9-7
- “Exporting Strategies” on page 9-9
- “Finding Blocks in Simulink” on page 9-10

Importing a Strategy

To import a Simulink strategy,

- 1 Highlight the **Feature** in the branch display.
- 2 Select **File -> Import -> Strategy**.
- 3 Select the appropriate .mdl file. CAGE checks the strategy for more than one output.
- 4 Select the output that you want to use.

If there is more than one output to your strategy, a Simulink window opens. Double-click the correct blue output to parse (or import) the strategy to your feature.

If there is only one output to your strategy, a dialog box opens:

- a Select **Automatic** to parse the strategy without viewing it.

Or

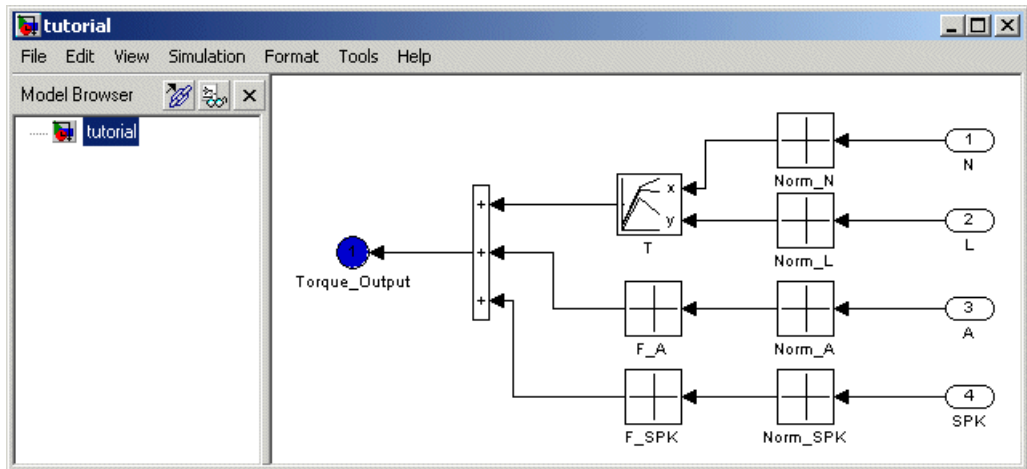
- b Select **Manual** to edit the strategy. Double-click the blue output circle to parse the strategy to your feature.

Note When you double-click the blue output, the Simulink windows shut and parse this strategy to your feature.

To view a representation of your strategy, select the **Feature** node. Your strategy is represented in the **Strategy** pane.

For information about using Simulink to amend strategies, see “Constructing a Strategy” on page 9-7.

Example. In the `matlab\toolbox\mbc\mbctraining` directory, there is a Simulink diagram called `tutorial.mdl`. If you import this and select **Manual** in the dialog box, you see the following diagram.



Double-click the Torque -Output outputport to parse the strategy into the **Strategy** pane.

Constructing a Strategy

For you to perform a feature calibration, the strategy and the model must have some variables in common.

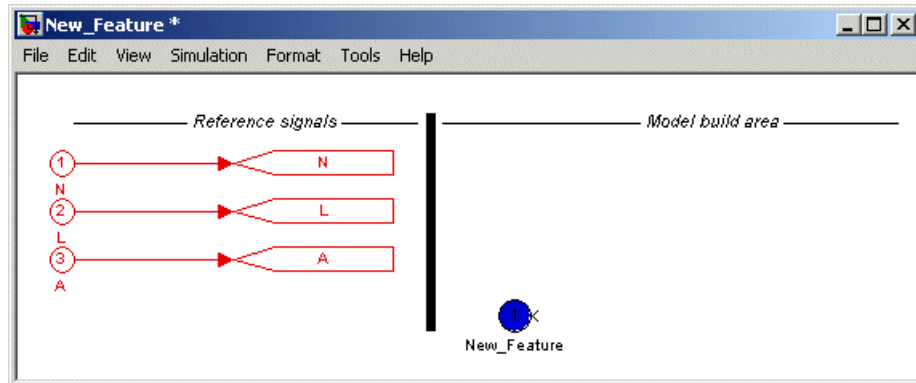
To construct a strategy using Simulink,

- 1 Highlight the correct feature by clicking the **Feature** node.
- 2 Select **Feature** -> **Graphical Strategy Editor** or press **Ctrl+E**.

Three Simulink windows open:

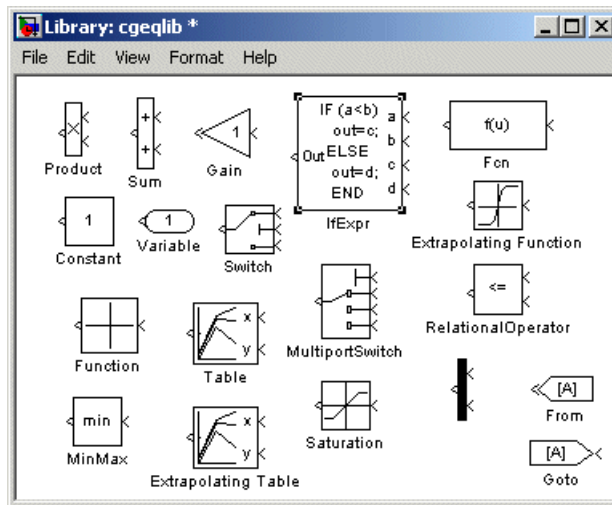
- The strategy window for editing your strategy

Example of a Strategy Window



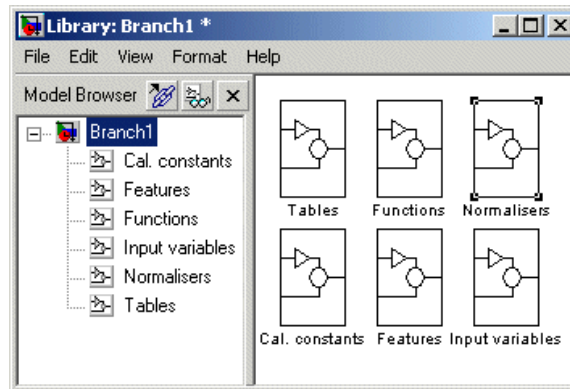
- A library window with all the blocks available for building a strategy

Library of All Available Blocks



- A library window with all the existing blocks in your session, organized in libraries

Library of All Existing Blocks



- 3 In the strategy window, build your strategy using the blocks available in the library windows.
- 4 Double-click the blue outputport circle to parse the strategy into the CAGE session.

Note This closes all three Simulink windows and parses your strategy into the feature.

For more information about using Simulink to build your strategy, see Simulink Help.

Exporting Strategies

Simulink strategies can be exported. For example, you might want to

- Include a strategy in a Simulink vehicle model
- Evaluate the strategy using Real-Time Workshop[®] to produce C code
- Evaluate the strategy using Simulink

To export a strategy from CAGE,

- 1 Highlight the **Feature** node that contains the strategy that you want to save.

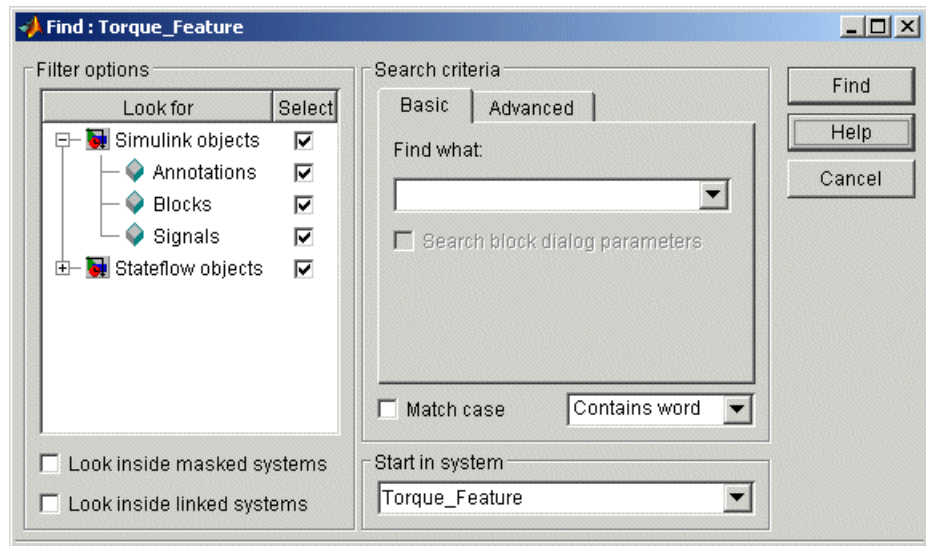
- 2 Select **File** -> **Export** -> **Strategy**.
- 3 Assign a name for your strategy.

The strategy is saved as a Simulink model (.mdl) file.

Finding Blocks in Simulink

To find blocks in Simulink, highlight a Simulink window and press **Ctrl+F** or select **Edit** -> **Find**.

This opens the **Find** dialog box, which helps find the blocks associated with your feature.



For more information about finding blocks, see Simulink Finder Help.

Calibrating the Tables

After you set up your session and your tables, you can calibrate your tables.

Highlight a table in the branch display to view the **Table** view. For more information about the **Table** view, see “Table View” on page 9-29.

In CAGE, a table is defined to be either a one-dimensional or a two-dimensional lookup table. One-dimensional tables are sometimes known as characteristic lines or functions. Two-dimensional tables are also known as characteristic maps or tables.

Each lookup table has either one or two axes associated with it. These axes are normalizers. See “Normalizers” on page 8-1 for more information.

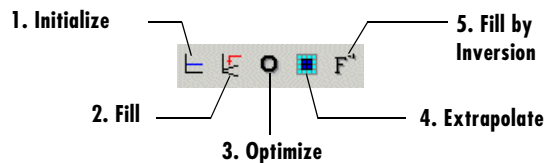
For example, a simple MBT feature has two tables:

- A two-dimensional table with speed and relative air charge as its normalizer inputs
- A one-dimensional table with AFR as its normalizer input

Before you can calibrate your tables, you must calibrate your normalizers. For information, see “Calibrating the Normalizers” on page 8-3.

This section describes how you can use CAGE to fill your lookup tables by reference to a model.

Table Node Toolbar



To fill the table values, either click the buttons in the toolbar or select from the following options in the **Table** menu:

- **Initialize**

Sets each cell in the lookup table to a specified value. For information, see “Initializing Table Values” on page 9-12.


- **Fill**
Fills the table values by reference to the model. For information, see “Filling Table Values” on page 9-13.
- **Optimize**
Fills the table values by minimizing the total square error between the table values and the model. For information, see “Optimizing Table Values” on page 9-16.
- **Extrapolate**
Fills the table values based on the cells specified in the extrapolation mask. You can choose values in cells that you trust to define the extrapolation mask and fill the rest of the table using only those cells for extrapolation. For information, see “Filling the Table by Extrapolation” on page 9-19.
- **Fill by Inversion**
Fills the table by creating an inversion of another table. For information, see “Inverting a Table” on page 9-21.

The next sections describe each of these in detail.

Initializing Table Values

Initializing table values sets the value of every cell in the selected table to a constant.

To initialize the values of the table,

- 1 Click  or select **Table -> Initialize**.
- 2 In the dialog box that appears, select the constant value that you want to insert into each cell.

When initializing tables, you should think about your strategy. Filling with zeros can cause a problem for some strategies using “modifier” tables. For example, your strategy might use several speed-load tables for different values of AFR, or you might use an AFR table as a “modifier” to add to a single speed-load table to adjust for the effects of different AFR levels on your torque output.

If your table is a modifier that is added to other tables, you should initially fill it with zeros; if it is a modifier that multiplies other tables, you should fill it

with 1s. If you do not, when CAGE tries to fill the table by rearranging the strategy equation (model = table * modifier table), there is a problem, as you cannot divide by zero. This operation will fail.

See “How CAGE Fills Tables” on page 9-13.

Filling Table Values

This tool fills the table with the values of the model at the operating points specified in your normalizers.

To fill the table values by reference to the model,

- Click  or select **Table -> Fill**.

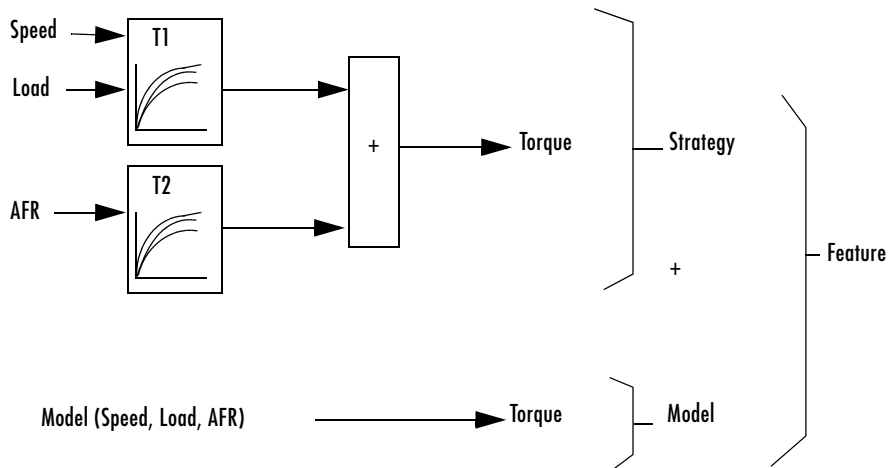
How CAGE Fills Tables

CAGE fills tables in a feature calibration by rearranging the equation $\text{model} = \text{strategy}$.

Example

A very simple example strategy for torque might consist of two tables:

- A speed-load (or relative air-charge) table filled with values of torque
- An air/fuel ratio (AFR) modifier table to account for the behavior of AFR



This example is a strategy with a base speed-load (N, L) table for torque and a modifier table to account for the behavior of AFR (A). Your strategy might use several speed-load tables for different values of AFR, or as in this case you might use an AFR table as a modifier to add to a single speed-load table to adjust for the effects of different AFR levels on your torque output.

$$T1(N,L) + T2(A) \approx \text{Model}(N, L, \text{AFR}) -$$

With the tables arranged in the following manner, this is the feature equation:

$$\text{Model} \approx T1 + T2$$

To fill the speed-load table, the equation is rearranged to

$$T1 = \text{Model} - T2$$

If the AFR modifier table (T2) is initialized with zeros, this becomes

$$T1 = \text{Model} - 0 \text{ or}$$

$$T1 = \text{Model}$$

Each cell in the table is therefore filled with the corresponding values of the model at the operating point specified by the breakpoints.

For example, to fill the T1 cell (Speed = 2500, Load = 0.5), CAGE evaluates the model at Speed = 2500, Load = 0.5, and uses the value of AFR that you choose in the dialog that appears.

You can choose one value of AFR for the whole table (for example, 14.3), or you can choose a range of values and fill using the average model value at each cell. For example, if you choose AFR = 11, 13, 15, the model is evaluated at all three values for each cell and the average model value is used. The default AFR value is the set point (which you can set in the variable dictionary).

Note To fill using model values averaged over a range of an input variable, you type the minimum and maximum (separated by a space) in the **Range** edit box, and the number of desired points in the **Points** edit box.

When the base table (T1) is filled, CAGE rearranges the equation again to fill the modifier table (T2):

$$T2(A) = \text{Model}(N,L,A) - T1(N,L)$$

For example, to fill the T2 cell at AFR = 12.5, you choose values of speed and load (such as 3000,0.4) and CAGE evaluates the following:

$$T2(12.5) = \text{Model}(3000, 0.4, 12.5) - T1(3000, 0.4)$$

As before, you can choose a range of values for speed and load and use the average to fill the table.

Note Be careful not to initialize modifier tables with 0 if they are multipliers in your strategy. In this case, solving $\text{Model} \approx T1 \times T2$ for T1 gives $T1 \approx \text{Model}/T2$, and you cannot divide by zero. This operation will fail.

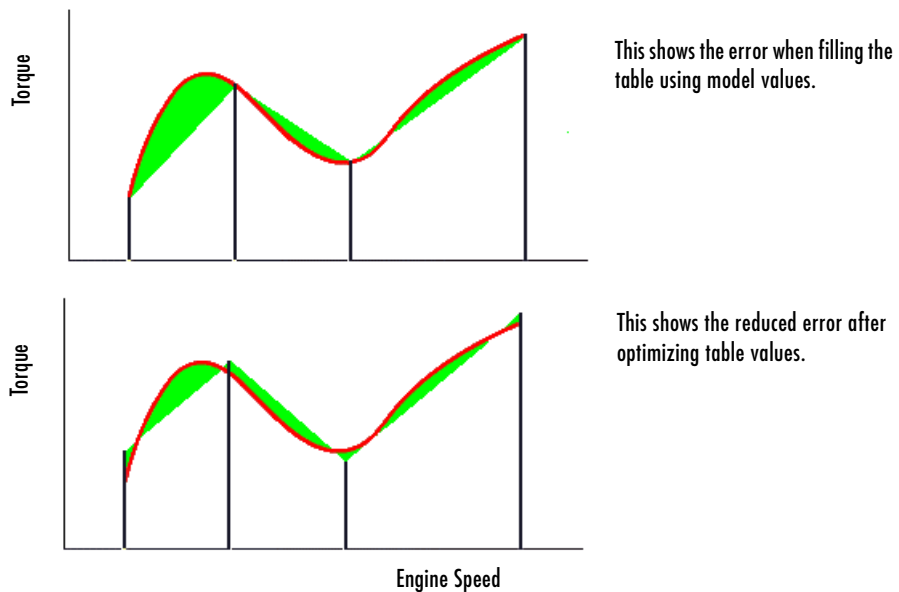
Solving $\text{Model} \approx \text{Strategy}$ algebraically for a table in the strategy is not always possible. In these cases you must use optimization.

Optimizing Table Values

Optimizing the table values minimizes the current total square error between the table values and the model.

This routine improves the fit between your strategy and your model. Using **Fill** places model values directly into your table, whereas the optimization process can shift those values up and down to give the least overall error between the interpolation between table values and the model surface. You should use **Fill** first to place model values into your table — this gives the optimization routine a good starting point.

This process is illustrated by the following example; the green shaded areas show the error between the mesh model (evaluated at the number of grid points you choose) and the table values.



To see the difference between optimizing table values and optimizing the positions of breakpoints, compare with the illustration in “Optimizing Breakpoints” on page 8-9.


For an example of optimizing table values, say you have a model of the spark angle that produces the maximum brake torque (MBT). The model has inputs engine speed N , relative air charge L , and air/fuel ratio A . The strategy that you are using to calibrate the MBT feature has an N - L table and a table to account for the variation of MBT over the range of A .

To optimize the table values,

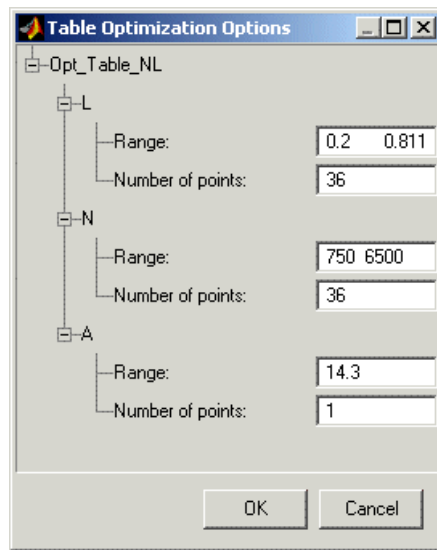
- 1 Ensure that the optimization routine works over a reasonable range of values by selecting **Table -> Fill**.

Optimization works best if you start with a sensible range of values. Using **Fill** places model values at the appropriate operating points into each cell of the table. From this point, the optimization routine has the best chance of finding a good solution quickly.

For example, a model for MBT might have a range of values from 20 degrees to 35 degrees. Running the optimization routine when most of the cell values are outside this range (say if your table is filled with zeros), is very time consuming.

- 2 Click  or select **Table -> Optimize**.

This opens the following dialog box.



3 Enter the ranges for the normalizers.

For the preceding example, enter 0.2 0.811 for the range of **L** and enter 750 6500 for **N**.

4 Enter the number of grid points for the optimization.

This defines a grid over which the optimization works. Above, the number of grid points is 36 for both L and N. This mesh is used to approximate the model. The default number of grid points is three times the number of breakpoints in the table.

5 Enter the ranges and numbers of points for the other model variables.

In the preceding example, the range of **A** is 14.3 and the **Number of points** is 1. The mesh approximates the value of the model at only one value of A.

6 Click **OK**.

CAGE evaluates the model over the number of grid points specified, then calculates the total square error between this mesh model and the table values.

CAGE adjusts the table values until this error is minimized, using `lsqnonlin`.

When optimizing the table values, it is worth noting the following:


- The default range for a normalizer variable is the range of the variable.
- The default value for all other model variables is the set point of the variable's range.
- The default number of grid points is three times the number of breakpoints.
- Increasing the number of grid points increases the quality of the approximation, but also the computation time.

See Also

- Reference page for `lsqnonlin`
- “Calibrating the Tables” on page 9-11

Filling the Table by Extrapolation

Filling a table by extrapolation fills the table with values based on the values already placed in the extrapolation mask. The extrapolation mask is described below.

To fill a table by extrapolating over a preselected mask, click  or select **Table** -> **Extrapolate**.

This extrapolation does one of the following:

- If the extrapolation mask has only one value, all the cell values change to the value of the cell in the mask.
- If the extrapolation mask has two or more collinear values, the cell values change to create a plane parallel to the line of values in the mask.
- If the extrapolation mask has three or more coplanar values, the cell values change to create that plane.
- If the extrapolation mask has four or more ordered cells (in a grid), the extrapolation routine fills the cells by a grid extrapolation.
- If the extrapolation mask has four or more unordered (scattered) cells, the extrapolation routine fills the cell values using a thin plate spline interpolant (a type of radial basis function).

Using the Extrapolation Mask

The extrapolation mask defines a set of cells that form the basis of any extrapolation.

For example, a speed-load (or relative air charge) table has values in the following ranges that you consider to be accurate:

- Speed 3000 to 5000 rpm
- Load 0.4 to 0.6

You can define an extrapolation mask to include all the cells in these ranges. You can then fill the rest of your table based on these values.

To add or remove a cell from the extrapolation mask,

- 1 Right-click the table.
- 2 Select **Add to/Remove from Mask** from the menu.

Cells included in the extrapolation mask are colored yellow.

Cells that are locked and in the extrapolation mask are colored green.

Generating the Extrapolation Mask from the Predicted Error

Predicted error (PE) is the standard deviation of the error between the model and the data used to create the model. You can automatically generate an extrapolation mask based on the predicted error.

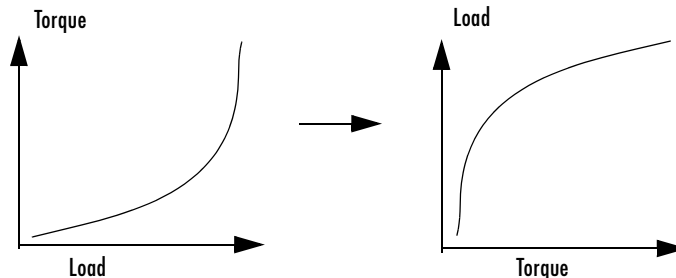
To generate a mask automatically,

- 1 Right-click the **Menu** button of the table and select **Generate Extrapolation Mask from Predicted Error**.
- 2 In the dialog box, set the PE threshold. Click **OK**.

The cells in the table where the predicted error is within the threshold now form the extrapolation mask, and thus are colored yellow.

Inverting a Table

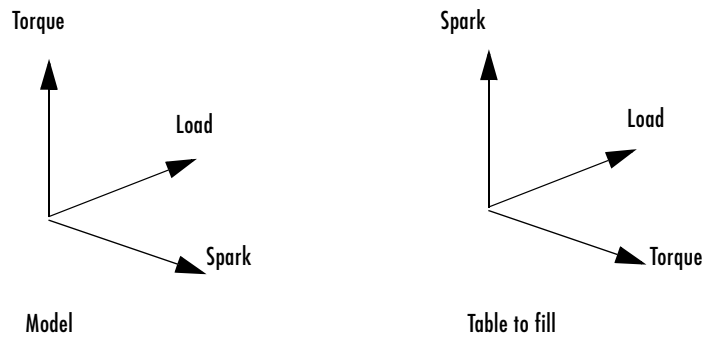
You can use CAGE to produce a table that is the inverse of another table. This involves swapping a table input with a table output, and you can invert 1-D or 2-D tables.



Inverting a table allows you to link a *forward strategy* to a *backward strategy*; that is, swapping inputs and outputs. This process is desirable when you have a “forward” strategy, for example predicting torque as a function of speed and load, and you want to reverse this relationship in a “backward strategy” to find out what value of load would give a particular torque at a certain speed.

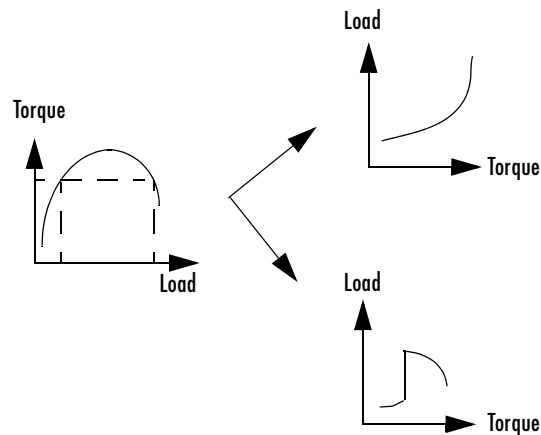
Normally you fill tables in CAGE by comparing with data or models. Ideally you want to fill using the correct strategy, but that might not be possible to find or measure. If you only have a forward strategy but want a backward one, you can fill using the forward strategy (tables or model) and then invert the table.

For example, in order to fill a table normally from a model, you need the model response to be the table output, and the model inputs to be a function of the table inputs (or it should be possible to derive the input — for example, air mass from manifold pressure). If the available model is “inverted” (the model response is a table input and the table output is a model input) and you cannot change the model, you can invert the table in CAGE.



In the diagram of a table shown, the x - and y -axes represent the normalizers (which you want to be spark and load) and the z -axis is the output at each breakpoint (torque). To fill this table correctly from the model is a two-step process. First you need to fill a table that has the same input and output as the model, and then fill a second table by inversion.

For the inversion to be deterministic and accurate, the table to be inverted must be monotonic; that is, always increasing or decreasing. This requirement is explained by the following one-dimensional example. Every point on the y -axis must correspond to a unique point on the x -axis. The same problem applies also to two-dimensional tables: for any given output in the first table there must be a unique input condition; that is, every point on the z -axis should correspond to a unique point in the x - y plane. Some table inversions have multiple values and so do not meet this requirement, just as the square root function can take either positive or negative values. You can use the inversion wizard in CAGE to handle this problem; you can control the inversion process and determine what to do in these cases.



The preceding example illustrates a table with multiple values. There are two solutions for a single value of torque. CAGE has a table inversion wizard that can help overcome this problem. You can specify whether you want to use the upper or lower values for filling certain parts of the table; this allows you to successfully invert a multiple-valued function. See the inversion instructions for 1-D and 2-D tables in the next sections.

The process of inverting a one-dimensional table is different from the process of inverting a two-dimensional table.

Inverting One-Dimensional Tables

To invert a one-dimensional table,

- 1 Ensure that your session contains two tables:
 - a The first table from your forward strategy, filled
 - b The second table from your backward strategy, which you want to fill
- 2 Highlight the second table.
- 3 Click **F³** or select **Table -> Inversion**.
The lower pane now acts as a wizard.
- 4 In the lower pane, highlight the table that you want to invert.

- 5 Click **Next**. The next page asks what CAGE should do if it encounters multiple values. The options are
- **Maximum** selects the uppermost range if a given number has two possible inverses (like selecting the positive square root of a number).
 - **Minimum** selects the lower of the two if a given number has two possible inverses (like selecting the negative square root of a number).
 - **Intermediate** selects the middle range if a given number has more than two possible inverses.
 - **Automatic** selects the range that produces the least error (see below; the last page of the wizard plots the error metric).

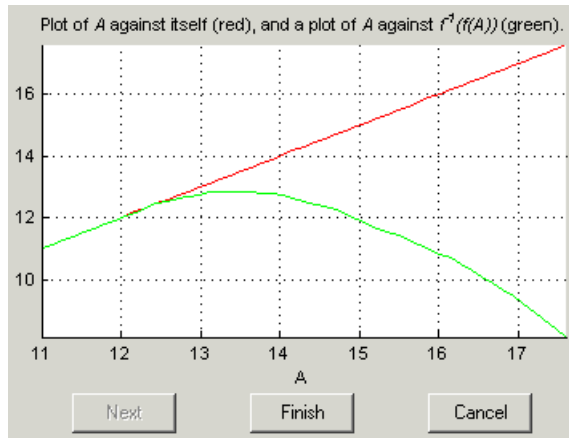
For example, the function $y = x^2$ is impossible to invert over the range -1 to 1. You can specify to invert the range from 0 to 1, sacrificing the inversion in the lower range, or the reverse. To select the range from 0 to 1, highlight **Maximum**.

The display shows a comparison between the table (green) and the function $x = f^{-1}(f(x))$.

- 6 Highlight the part of the table to invert, then click **Next**.

The last page of the wizard has a comparison plot that shows how successful the inversion has been. If your forward function is $y = f(x)$, and your inverse function is $x = g(y)$, then, combining these, in an ideal world, you should have $x = g(f(x))$. The plot then displays a red line showing x against x and a green line showing x against $g(f(x))$. The closeness of these two lines indicates how good the inversion has been: a perfect inverse would show the lines exactly on top of each other. In the following example, the lines are together and then diverge; this plot can show you which part of your table has not successfully inverted and where you should try a different routine.

Inverting a One-Dimensional Table



Note The automatic inversion routine tries to minimize the total distance between these lines. This can sometimes lead to unexpected results. For example, given the function $f(x) = x^2$ between -1 and 1 , if you select either positive or negative square root as the inverse, this induces a large error in the combined inverse. If you choose $g(y) = \sqrt{y}$, then $g(f(-1)) = 1$, an error of 2 . To minimize this, the automatic routine might choose to send everything to zero and accept a medium error over the whole range rather than a large error over half the range. The more knowledge you have of the form of the “forward” table, the more you can make an informed choice about which routine to select.

- 7** Click **Finish** to accept the inversion or **Cancel** to ignore the result and return to the original table.

Inverting Two-Dimensional Tables

To invert a two-dimensional table,

- 1 Ensure that your session contains two tables:
 - a The first table from your forward strategy, filled
 - b The second table from your backward strategy, which you want to fill
- 2 Highlight the second table.
- 3 Click **F⁺** or select **Table -> Inversion**.

The lower pane now acts as a wizard.

- 4 In the lower pane, highlight the table that you want to invert.
- 5 Click **Next**.
- 6 Identify the corresponding signals.

The forward table and backward table share a common input. This page of the wizard lists all possible combinations of inputs into the forward and backward tables and asks you to highlight the combination that gives the two common inputs. To illustrate this, if the forward table gives torque in terms of the variables engine speed and load, whereas you want the backward table to give load in terms of RPM and Tq, then the list would read

- RPM and engine speed
- RPM and load
- Tq and engine speed
- Tq and load

In this case, you would select the first option.

- 7 Highlight the part of the table to invert, then click **Next**.

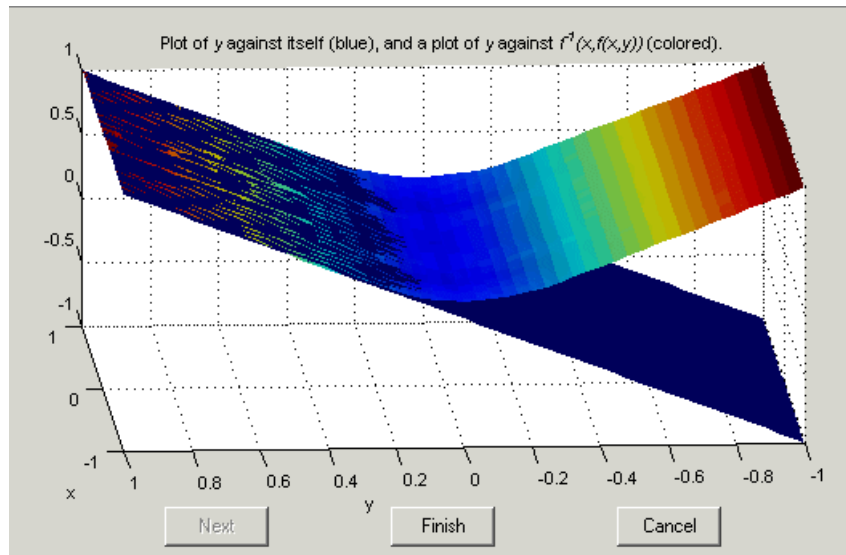
CAGE asks what to do if it encounters multiple values. The choices are

- **Maximum** selects the uppermost range (like choosing a positive square root of a number).

- **Minimum** selects the lower value if there are two choices (like choosing a negative square root of a number).
- **Intermediate** selects the middle range when there are more than two choices.
- **Automatic** selects the range that produces the least error. CAGE tries to choose values to put in the inverse table that minimize an error metric similar to the error metric for 1-D tables (see “Inverting One-Dimensional Tables” on page 9-23).

8 Choose one of these options and click **Next**.

The last page of the wizard has a comparison plot that shows how successful the inversion has been. If the forward function is $z = f(x,y)$, and the inverse function is $x = g(y,z)$, then, combining these, in an ideal world you should have $x = g(y, f(x,y))$. The plot then displays a plane showing x plotted against x and y , and a colored surface showing $g(y, f(x,y))$ plotted against x and y . The closeness of these two planes indicates how good the inversion is. Following is an example. In this case, the forward table is a quadratic ($z = y^2$); the backward table is inverted using the positive square root of z (maximum range). As you can see, this leads to large errors at negative values of y , but good inversion for positive values of y .



- 9 Click **Finish** to accept the result or **Cancel** to ignore the result and return to the original table.

Table View

When you select a table in the branch tree (in feature or manual calibration), you see the **Table** view. In CAGE, a table is defined to be either a one-dimensional or a two-dimensional lookup table. One-dimensional tables are sometimes known as characteristic lines or functions. Two-dimensional tables are also known as characteristic maps or tables. CAGE regards them both as similar objects.

Each lookup table has either one or two axes associated with it. These axes are normalizers.

For example, a simple MBT feature has two tables:

- A two-dimensional table with speed and relative air charge as its normalizers
- A one-dimensional table with AFR as its normalizer

For feature calibration (filling the tables by comparing a strategy and a model), see “Calibrating the Tables” on page 9-11.

For an example of manual calibration (filling tables using experimental data), see “Tutorial: Filling Tables from Data” on page 5-1.

The example following is a feature view. In manual calibration, you do not see the comparison pane because you are not comparing tables with a model.

Table Display in Feature Calibration

Selected node

1. Table

2. Graph of the table

Feature	0.20	0.24	0.29	0
750.00	25.714	22.628	19.712	17.
1156.60	34.226	30.692	27.289	24.
1621.20	39.013	35.408	31.888	28.
2143.90	39.327	35.995	32.706	29.
3131.30	35.611	32.957	30.279	27.
3654.00	37.104	34.525	31.876	29.
4351.00	44.358	41.579	38.645	35.
4931.80	45.449	42.952	40.255	37.
5861.10	40.024	38.680	37.129	35.
6500.00	37.393	36.996	36.423	35.

3. Comparison of results

Table variables

L: Min: 0.2 Max: 0.811 Pts: 20

N: Min: 750 Max: 6500 Pts: 20

Model variables

A: 14.3

E: 6

Maximum error: 0.7656

Mean square error: 0.05351

Total square error: 21.4

The parts of the display are numbered and labeled as follows:

- 1 The table pane displays the breakpoints of the normalizer and the values of the table. (See “Viewing a Table” on page 9-31.)
- 2 The graph of the table pane displays the table values graphically. (See “Using the Graph of the Table” on page 9-32.)

- 3** The comparison-of-results pane displays a comparison between the current output of the strategy and the feature model. (See “Comparing the Strategy and the Model” on page 9-33.)

Note You can view the **History** display by selecting **View -> History**. For information, see “Using the History Display” on page 15-5.

This section describes each of these parts in detail.

Viewing a Table

The table displays the values of your lookup table and displays the breakpoints of the normalizers. For example, the following table shows a lookup table with speed and relative air charge (load) as its normalizers.

The screenshot shows a table with 13 rows and 4 columns. The first column contains values from 500.00 to 6500.00. The second and third columns contain values from -3.201 to -11.961 and 8.078 to -0.679 respectively. The fourth column contains values from 19.191 to 10.433. Annotations point to various features: a 'Menu button' at the top left, a 'Cell in the extrapolation mask' (yellow cell at row 4, col 4), a 'Locked cell in extrapolation mask' (green cell at row 6, col 2), a 'Selected cell' (blue cell at row 8, col 4), and a 'Locked cell' (red cell at row 10, col 3). A scroll bar is visible at the bottom of the table.

	0.10	0.20	0.30
500.00	-3.201	8.078	19.191
999.35	-2.449	8.758	19.907
1501.90	-2.023	9.218	20.351
2011.70	-1.824	9.403	20.542
2499.80	-1.895	9.336	20.473
3000.20	-2.241	8.991	20.128
3500.70	-2.827	8.403	19.541
4000.20	-3.703	7.528	18.666
4501.50	-4.819	6.412	17.550
5000.10	-6.204	5.024	16.163
5499.50	-7.862	3.380	14.512
5999.30	-9.745	1.461	12.611
6500.00	-11.961	-0.679	10.433

To edit a value in the table, double-click the cell. Selected cells are light blue (or pink if they are locked or in the extrapolation mask).

Locking and Unlocking Cell Values

When you are satisfied with a region of the table, you might want to lock the cell values in that region, to ensure that those values do not change.

To lock or unlock a cell value, right-click the cell and select **Lock/Unlock** from the menu.

Locked cells are colored red in the display (they turn pink when selected).

Menu Button Options

Right-clicking the menu button in the top left corner of the table gives you the following options:

- **Lock/unlock entire table.** This toggles between the table's being locked or unlocked.
- **Generate Extrapolation mask from predicted error.** See "Generating the Extrapolation Mask from the Predicted Error" on page 9-20.
- **Clear extrapolation mask.** This ensures that none of the cells are in the extrapolation mask.

Properties

The table properties enable you to specify the precision type of the table data.

You can choose from

- Floating-point precision
- Polynomial ratio, fixed-point precision
- Lookup table, fixed-point precision

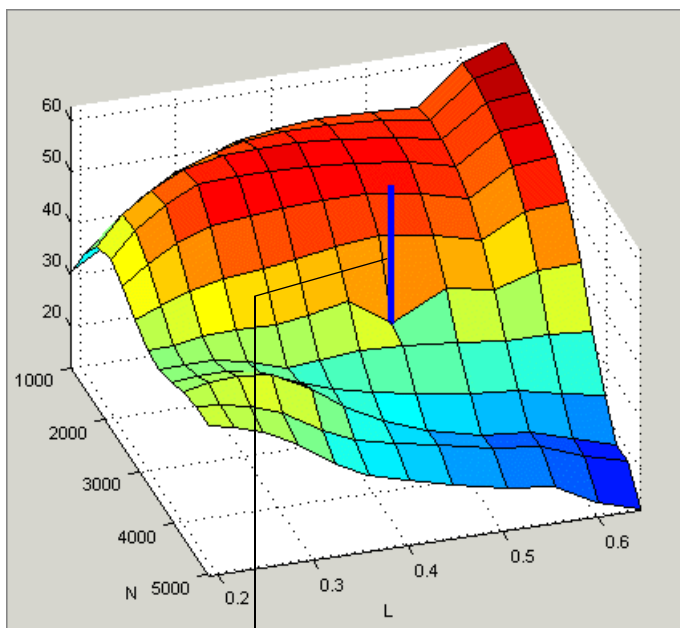
To display the properties of the table, select **Table -> Properties**.

This opens the **Table Properties** dialog box.

Table properties are discussed in detail in "Table Properties" on page 13-6.

Using the Graph of the Table

The table view displays both the table values and a graph of the table. This gives a useful display of the table's behavior. Shown is an example of a graph.

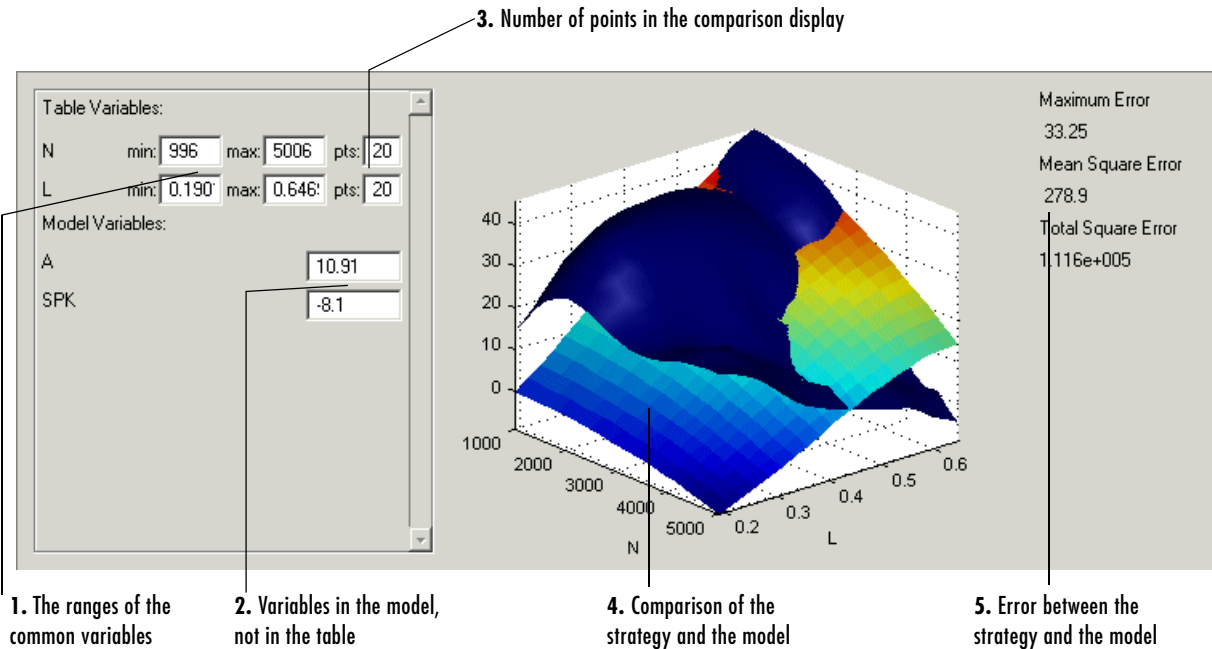


Line indicates which value in the table you are editing.

- You can rotate the graph of the table by clicking and dragging the axes.
- You can alter values in the table by clicking and dragging vertically any point.
- When you click a point, a blue line indicates the selected point in the table.

Comparing the Strategy and the Model

When you calibrate a strategy, or collection of tables, by reference to a model, it is useful to compare the strategy and the model. The following pane illustrates a comparison.



Note This is a comparison between the current strategy values and the model, unlike the comparison pane from the normalizer node, which compares the model and a full factorial grid filled using the breakpoints.

To make full use of the comparison-of-results pane,

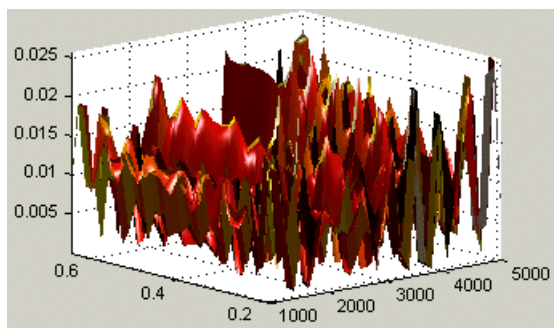
- 1 Check the ranges of the variables that are common to the model and table.
- 2 Check the values selected for any variables in the model that are not in the selected table. The default for this is the set point of the variable's range.
- 3 Check the number of points at which the display is calculated.
- 4 Check the comparison between the table and the model. You can rotate this comparison by clicking and dragging, so that you can view all parts of the comparison easily.

5 Check some of the error statistics for the comparison.

You can also view the error over the range of the feature.

Error Display

The comparison-of-results pane can also be used to display the error between the model and the strategy.



To display the error,

- 1 Right-click the axes of the comparison display.
- 2 Select **Error** from the menu.

This changes the graph to display the error between the model and the strategy.

You can display the error data in one of the following ways:

- **Error**. This is the difference between the feature and the model.
- **Squared Error**. This is the error squared.
- **Absolute Error**. This is the absolute value of the error.
- **Relative Error (%)**. This is the error as a percentage of the value of the model.
- **Absolute Relative Error (%)**. This is the absolute value of the relative error.

To select one of these displays of the error data,

- 1 Right-click the display.

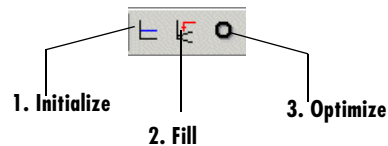
- 2** Select **Error Display** and select the appropriate display of the error from the context menu.

Calibrating the Feature Node

Selecting a **Feature** node displays the **Feature** view. For more information about the **Feature** view, see “Feature View” on page 9-44.

The **Feature** view enables you to calibrate the entire feature. Calibrating the feature means that you fill the breakpoints in the normalizer, and the table values, by referring to a model.

Feature Node Toolbar



To calibrate the feature, either click the buttons on the toolbar or select from the following options on the **Feature** menu described in these sections:

- 1 “Initializing the Feature” on page 9-37
- 2 “Filling the Feature” on page 9-39
- 3 “Optimizing the Feature” on page 9-41

Initializing the Feature

For example, a simple feature for maximum brake torque (MBT) consists of the following tables:

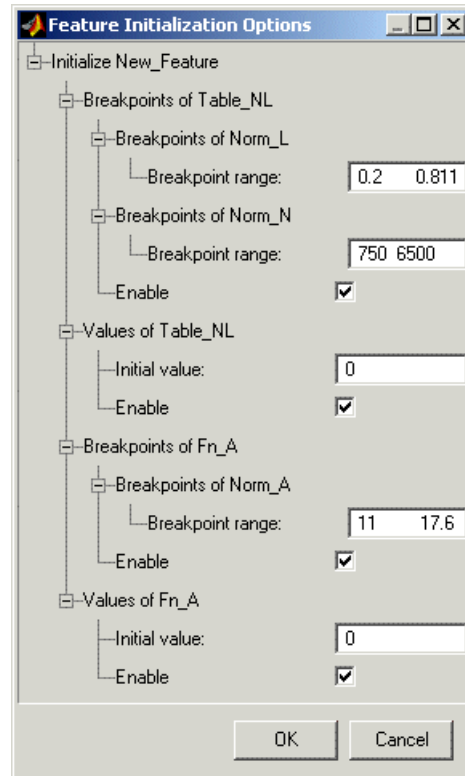
- A speed (N), load (L) table
- A table to account for the behavior of air/fuel ratio (A)

Initializing this feature sets the values of the normalizers for speed, load, and AFR over the range of each variable and put specified values into each cell of the two tables.

A table that is already initialized provides a useful starting point for a more detailed calibration.

To initialize the feature,

- 1 Click . This opens the **Feature Initialization Options** dialog box, as shown.



- 2 Enter the ranges for the breakpoints in your normalizers. In the preceding example, enter the following breakpoint ranges:
 - **L** has range 0.2 0.811.
 - **N** has range 750 6500.
 - **A** has range 11 17.6.
- 3 Enter the initial table value for each cell in each table. Above, enter the cell values as
 - **Table_NL** has initial value 0.
 - **Fn_A** has initial value 0.

4 Click **OK**.

Note The default values in this dialog box are taken from the variable dictionary. If you clear any **Enable** box, the associated table or normalizer is left unchanged.

Filling the Feature

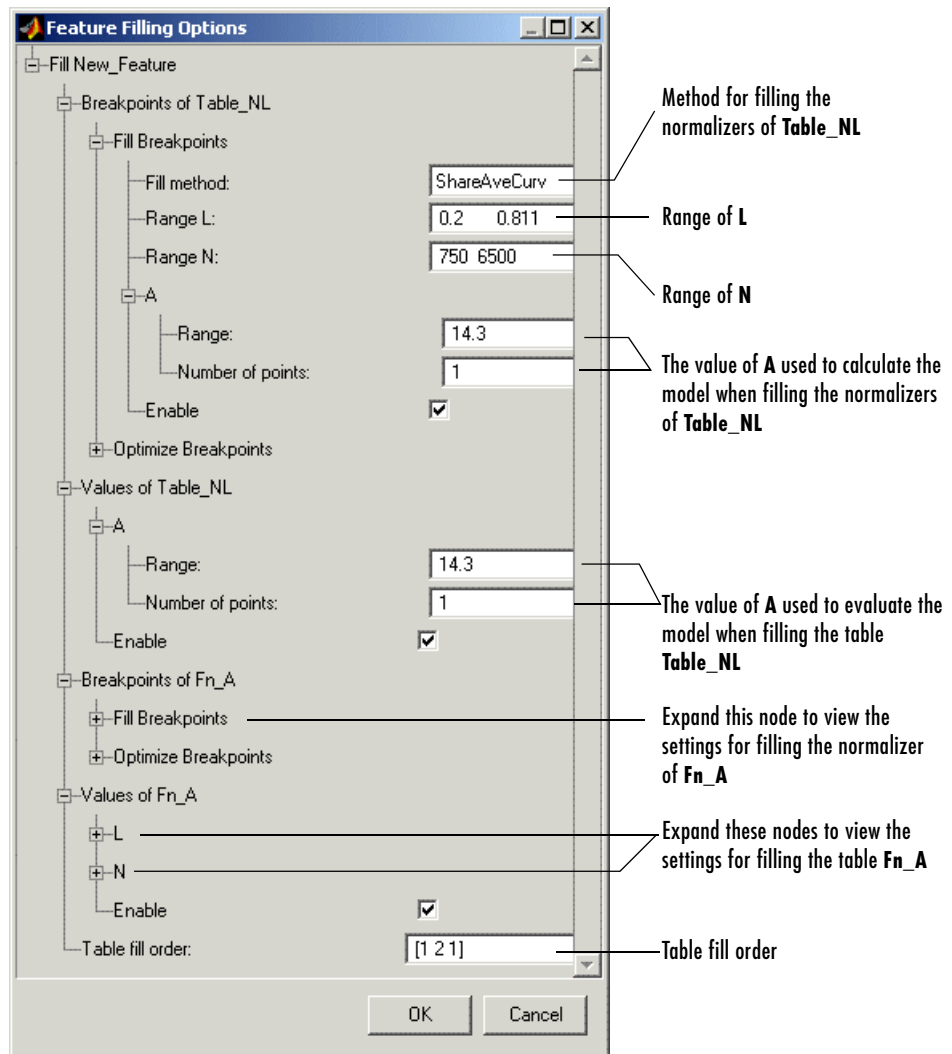
A very quick way to calibrate a feature is to fill it. This does two things:

- CAGE spaces the breakpoints of the normalizers by reference to the model. For example, the breakpoints can be spaced to place most breakpoints where the curvature of the model is greatest. This process is described in detail in “Filling Breakpoints” on page 8-5.
- Then CAGE fills the tables by reference to the model. This process is described in detail in “Filling Table Values” on page 9-13.

This section describes the procedure to fill a feature. For a detailed description about the filling processes, see the sections listed above.

To fill a feature,

- 1 Click . This opens the **Feature Filling Options** dialog box, shown.



- 2 Select the correct method for filling your normalizers.
- 3 Enter the ranges for the normalizers of the tables.
- 4 Enter the ranges of the other variables when filling the normalizers.

5 Enter the ranges of the other variables when filling the tables.

6 Enter the table fill order.

In this example, enter [1 2 1]. This fills the normalizers and then fills the table values of **Table_NL**. Then CAGE fills the normalizers, and then the table values of table **Fn_A**. After which CAGE then fills the normalizers and then the table values for **Table_NL** again.

7 Click **OK**.

You can iterate this process using the table fill order edit box, as in the example. This further improves the fit of the strategy and the model.

Note Feature fill gives you the choice of optimizing each normalizer before filling the table values. To do this, expand the **Optimize Breakpoints** node and select the **Enable** box. If you clear any **Enable** box, that table is not filled.

When you have completed a calibration, you can export your feature. For information, see “Exporting Calibrations” on page 7-21.

Optimizing the Feature

After filling the feature, you can improve the fit of the strategy to the model by optimizing the feature. The optimization routine does the following:

- First CAGE optimizes the breakpoints for the normalizers. (See “Optimizing Breakpoints” on page 8-9.)
- Next CAGE optimizes the values of the tables. (See “Optimizing Table Values” on page 9-16.)

This section gives the procedure for optimizing a feature. For further details of how optimization works, see the references given above.

To optimize a feature,

- 1 Click . This opens the **Feature Optimizing Options** dialog box, as shown.

The screenshot shows the 'Feature Optimization Options' dialog box with a tree view on the left and various input fields on the right. The tree view is expanded to show settings for 'Breakpoints in Table_NL', 'Values of Table_NL', and 'Breakpoints in Fn_A'. Annotations with arrows point to specific input fields and checkboxes, providing a detailed description of each setting.

Options for Optimizing Normalizers of Table_NL

- Range of normalizer L: 0.2 0.811
- Number of grid points for normalizer L: 36
- Range of normalizer N: 750 6500
- Number of grid points for normalizer N: 36
- The value of A used to evaluate the model when optimizing the normalizers of Table_NL: 14.3
- Number of points: 1
- Reorder Deleted Breakpoints:
- Enable:

Options for Optimizing Table Values of Table_NL

- Range and number of grid points for normalizer L, used for optimizing Table_NL: 0.2 0.811
- Number of points: 36
- Range and number of grid points for normalizer N, used for optimizing Table_NL: 750 6500
- Number of points: 36
- The value of A used to evaluate the model when optimizing Table_NL: 14.3
- Number of points: 1
- Enable:

Options for Optimizing the Normalizers and Table Fn_A

- Expand these nodes to view the settings for optimizing the normalizers and table Fn_A.
- Table optimization order: [1 2 1]
- Table fill order

Buttons: OK, Cancel

2 Enter the ranges of the normalizers for the normalizer optimization.

- 3 Enter the numbers of grid points for the normalizers.
- 4 Enter the values for the other variables.
- 5 Enter the ranges of the normalizers for the table optimization.
- 6 Enter the numbers of grid points for the normalizers.
- 7 Enter the values of the other variables.
- 8 Enter the table fill order.

In this example, enter [1 2 1]. This optimizes the normalizers and then the table values of **Table_NL**. Next CAGE optimizes the normalizers, and then the table values of table **Fn_A**. After which CAGE optimizes the normalizers and then the table values for **Table_NL** again.

- 9 Click **OK**.

Note You can iterate this process using the table fill order edit box, as in the example. This further improves the fit of the strategy and the model.

When you have completed a calibration, you can export your feature. For information, see “Exporting Calibrations” on page 7-21.

Feature View

As you select a **Feature** node you see the **Feature** view, shown. This section describes the **Feature** view and the **Feature** menu options.

Selected feature 1. The strategy for the selected feature 2. The model associated with the selected feature

The screenshot displays the Feature View interface. On the left is a tree view under 'Feature' containing 'Branch 1', 'New_Feature', 'T', 'Norm_N', 'Norm_L', 'F_A', and 'F_SPK'. The main area is divided into three sections:

- 1. The strategy for the selected feature:** Shows 'Strategy: Inputs: N , L , A , SPK' and the equation $(T(\text{Norm}_N(N), \text{Norm}_L(L)) + F_A(\text{Norm}_A(A)) + F_{SPK}(\text{Norm}_{SPK}(SPK)))$.
- 2. The model associated with the selected feature:** Shows 'Model: tq, Inputs: N , L , A , SPK' and buttons for 'Select Model...' and 'Deselect Model'.
- 3. Feature History pane:** A scrollable list with 'Comment / Action' and 'Strategy equation diagram created and parsed', accompanied by 'Add' and 'Remove' buttons.

Details:

The parts of the **Feature** view include

- 1 The strategy for the selected feature. This is the algebraic collection of the tables that you are using to calibrate the selected feature.
- 2 The model associated with the selected feature.
- 3 The **Feature History** pane, which displays the history of the feature.

Feature Menu

The **Feature** menu has the following options:

- **Select Model**
Use this to select the correct model for your feature.
- **Deselect Model**
Use this to clear the current model from your feature.
- **Convert Feature to a Model**
Takes the current feature and converts it to a model, which you can view by clicking the **Model** button.
- **Clear Current Strategy**
Clears the current strategy from your feature.
- **Graphical Strategy Editor**
Opens your current strategy for editing. For more information, see “Setting Up Your Strategy” on page 9-5.
- **Parse Strategy Diagram**
Performs the same function as double-clicking the blue output of your strategy diagram. For more information, see “Setting Up Your Strategy” on page 9-5.
Enables you to view the feature and the model using the surface viewer. For information, see “Surface Viewer” on page 14-1.
- **Initialize**
Initializes the feature; also in the toolbar. See “Initializing the Feature” on page 9-37 for details.

- **Fill**
Fills the feature; also in the toolbar. See “Filling the Feature” on page 9-39 for details.
- **Optimize**
Optimizes the feature; also in the toolbar. See “Optimizing the Feature” on page 9-41 for details.

Tradeoff Calibrations

This section includes the following topics:

Performing a Tradeoff Calibration (p. 10-2)	An overview of the steps required for tradeoff calibration.
Setting Up a Tradeoff Calibration (p. 10-5)	How to set up a new tradeoff, add tables, and display models.
Calibrating Tables in a Tradeoff Calibration (p. 10-9)	How to calibrate tables in a tradeoff calibration; setting values for other variables and determining suitable values at specific operating points.
Using Regions (p. 10-15)	How to use regions to fill specific parts of your table by extrapolation.
Multimodel Tradeoffs (p. 10-17)	How to set up and use multimodel tradeoffs.

Performing a Tradeoff Calibration



A tradeoff calibration is the process of calibrating lookup tables by adjusting the control variables to result in table values that achieve some desired aim.

For example, you might want to set the spark angle and the air/fuel ratio (AFR) to achieve the following objectives:

- Maximize torque
- Restrict CO emissions

The data in the tradeoff is presented in such a way as to aid the calibrator in making the correct choices. For example, sometimes the model is such that only a slight reduction in torque results in a dramatic reduction in CO emissions.

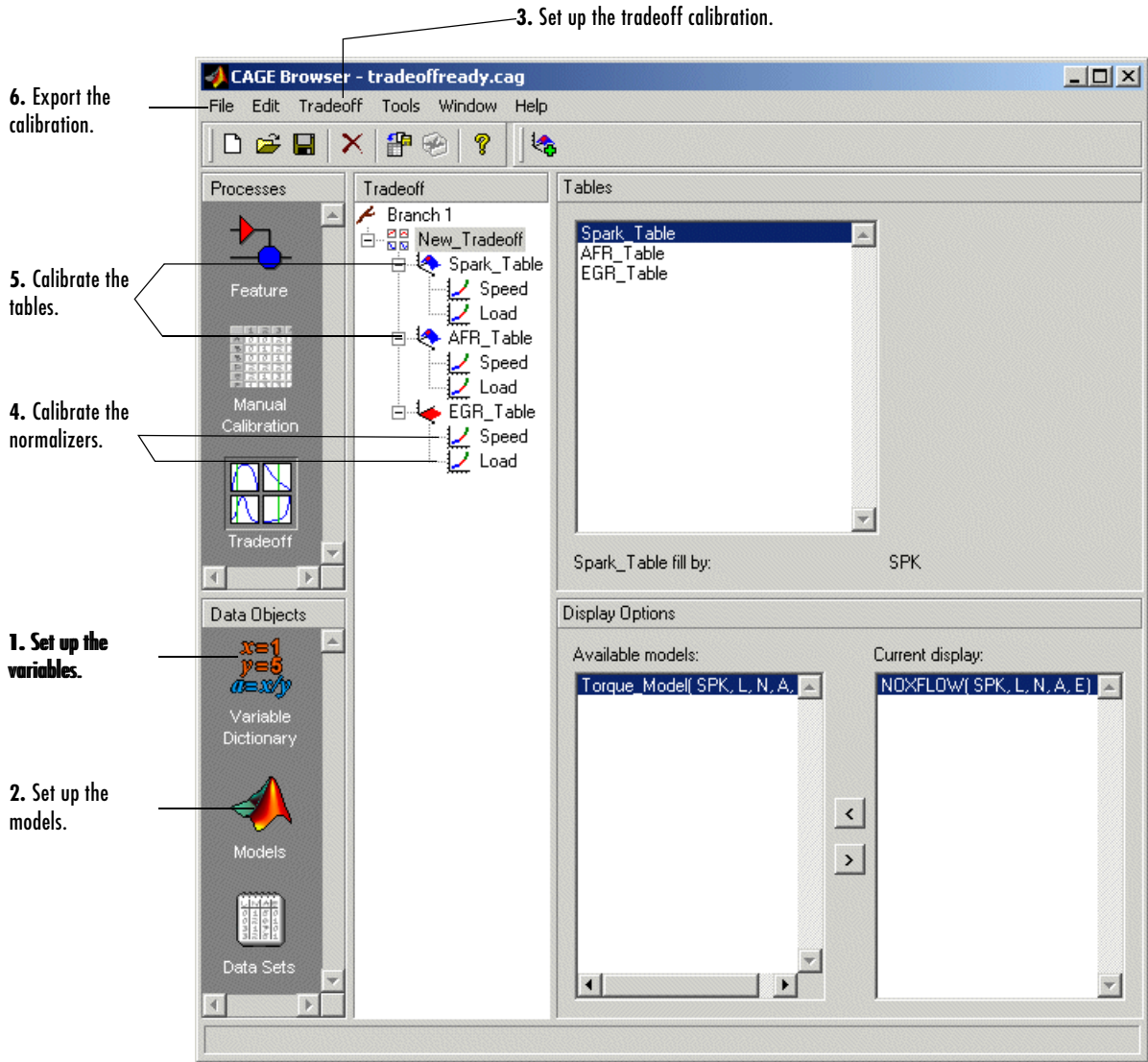
The basic procedure for performing tradeoff calibrations is as follows:

- 1** Set up the variables and constants. (See “Setting Up Your Variable Items” on page 7-6.)
- 2** Set up the model or models. (See “Setting Up Your Models” on page 7-13.)
- 3** Set up the tradeoff calibration. (See “Setting Up a Tradeoff Calibration” on page 10-5.)
- 4** Calibrate the normalizers. (See “Calibrating the Normalizers” on page 8-3.)
- 5** Calibrate the tables. (See “Calibrating Tables in a Tradeoff Calibration” on page 10-9.)
- 6** Export the normalizers, tables, and tradeoffs. (See “Exporting Calibrations” on page 7-21.)

You can also use regions to enhance your calibration. (See “Using Regions” on page 10-15.)

See Also

- “Tutorial: Tradeoff Calibration” on page 3-1
This is a tutorial giving an example of how to set up and complete a tradeoff calibration.
- “Automated Tradeoff” on page 11-33 is a guide to using the optimization functionality in CAGE for tradeoffs.



The normalizers, tables, and tradeoff form a hierarchy of nodes, each with its own view and toolbar.

Setting Up 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 instead of the values that produce the maximum value of torque.

A tradeoff also refers to the object that contains the models and tables. Thus, a simple tradeoff can involve balancing the torque output while restricting NOX emissions.

After you set up your variable items and models, you can follow the procedure below to set up your tradeoff calibration:



- 1 Add a tradeoff. This is described in the next section, “Adding a Tradeoff” on page 10-5.
- 2 Add tables to the tradeoff. This is described in “Adding Tables to a Tradeoff” on page 10-6.
- 3 Display the models. This is described in “Displaying Models in Tradeoff” on page 10-7.

This section describes steps 1, 2, and 3 in turn.

When you finish these steps, you are ready to calibrate the normalizers and tables.

Adding a Tradeoff

To add a tradeoff to your session, select **File -> New -> Tradeoff**. This automatically switches you to the **Tradeoff** view and adds an empty tradeoff to your session.

An incomplete tradeoff is a tradeoff that does not contain any tables. If a tradeoff is incomplete, it is displayed as  in the branch display. If a tradeoff is complete, it is displayed as  in the branch display.

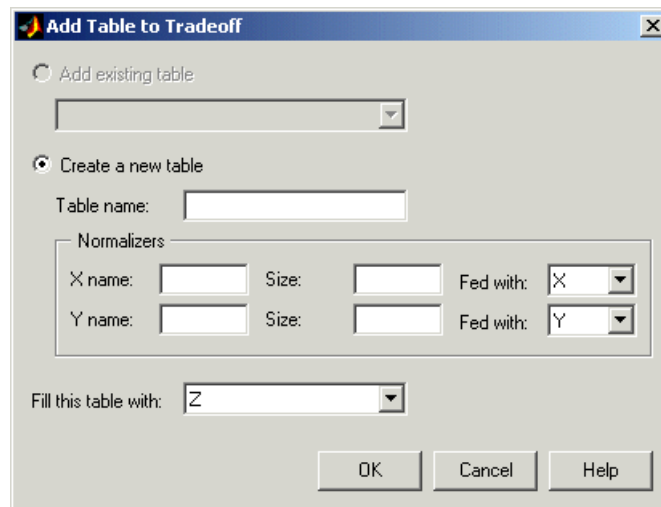
After you add a tradeoff you must add tables to your tradeoff.

Adding Tables to a Tradeoff

1 Add a table by selecting **Tradeoff** -> **Add Table**.

Note that you must select the top tradeoff branch in the tree display to use the **Tradeoff** menu. This is automatically selected if your tradeoff has no tables yet (it is the only branch). You must also add at least three variables (in the variable dictionary) to your project before you can add a table, because CAGE needs a variable to fill the table and two more variables to define each of the two normalizers.

A dialog box opens.



2 Either select a current table from your CAGE session to calibrate, using the top list box, or create a new table. You can add existing tables in the session;

you can choose them from the drop-down list as long as they are not empty (that is, they must be initialized tables that have values and defined sizes).

To create a new table,

- a Enter the name for the table.
- b Enter names for each of the normalizers.
- c Enter sizes for each of the normalizers.
- d Select the input for each of the normalizers.
- e Select the variable or model you want to fill the table with.

3 Click **OK**.

CAGE adds a table node to the tree.

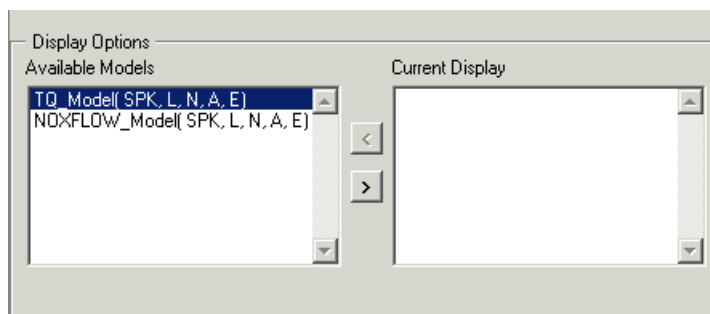
4 Repeat this procedure for each new table you want to add.

Note Each additional table must have the same normalizers as the first table, so you do not have to perform steps **b**, **c**, and **d** repeatedly.


Displaying Models in Tradeoff

To display models when viewing tables in the tradeoff display,

- 1 Highlight the tradeoff node.
- 2 From the **Available Models**, select the one you want to display.
Models that are referenced by tables are automatically displayed.
- 3 Click **>** to move the selected model into the **Current Display** pane.
- 4 Repeat steps 2 and 3 until you have displayed all the models that you want to work with.



Deselecting a Model.

- 1 In the **Current Display**, select the model that you want to remove.
- 2 Click  to move the selected model into the **Available Models** pane.
- 3 Repeat until you have cleared all the appropriate models.

Once you have displayed all the models that you want to work with, you are ready to calibrate your normalizers and tables.

Calibrating Tables in a Tradeoff Calibration

Selecting a table node in the branch tree display enables you to view the models that you have displayed and calibrate that table.


To calibrate the tables,

- 1 Select the table that you want to calibrate.
- 2 Highlight one operating point from the table.
- 3 Set the values for other input variables.


For information, see “Setting Values of Other Variables” on page 10-11.

- 4 Determine the value of the desired operating point.

For information, see “Determining a Value at a Specific Operating Point” on page 10-13.

- 5 Click  to apply this value to the lookup table.

This automatically adds the point to the extrapolation mask.

- 6 Repeat steps 2, 3, 4, and 5 at various operating points.
- 7 Extrapolate to fill the table by clicking .

For information, see “Filling the Table by Extrapolation” on page 9-19.

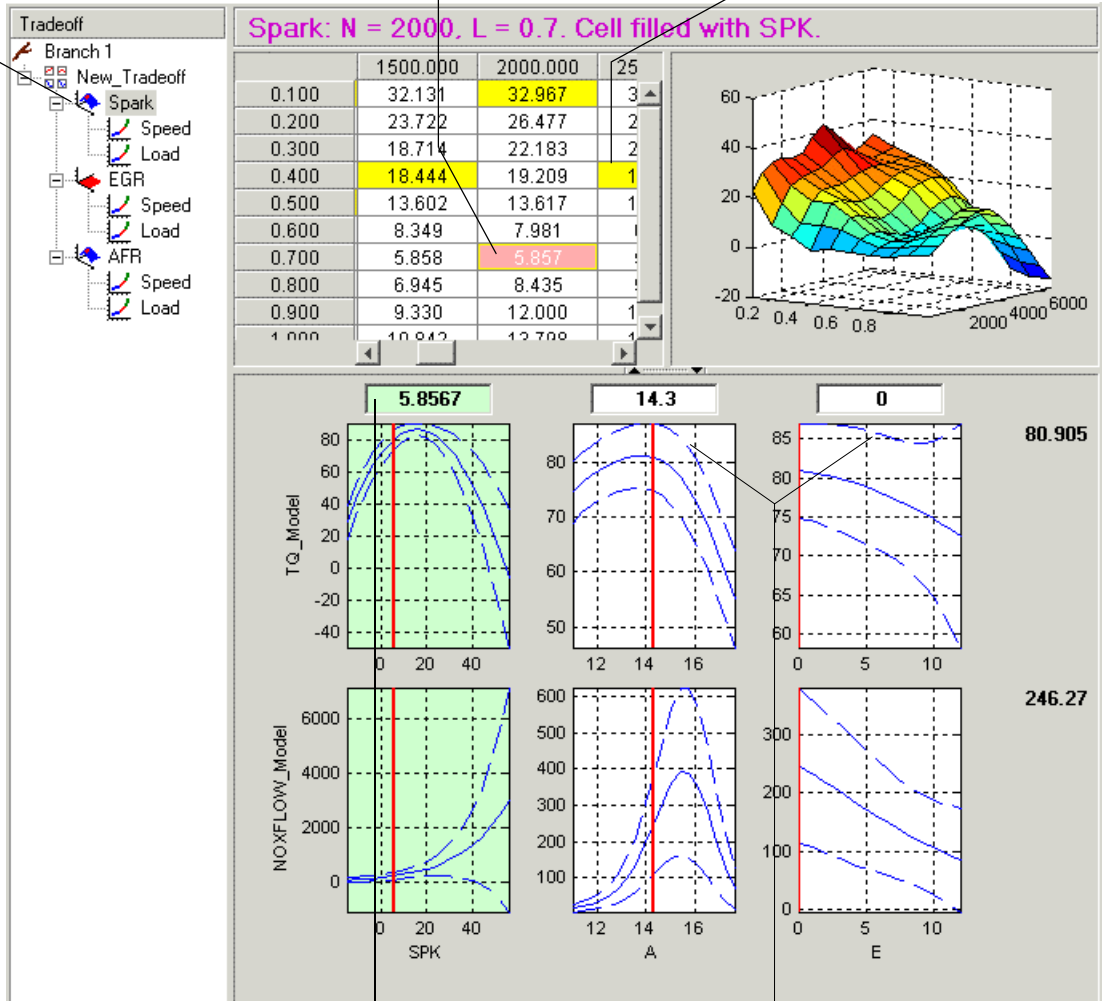
After you complete all these steps you can export your calibration. For information, see “Exporting Calibrations” on page 7-21.

Table View in a Tradeoff Calibration

1. Select the table.

2. Select the operating point in the table that you want to calibrate.

6. Repeat this process over a number of operating points in the table.



4. Determine a suitable value for the point.

3. Set the values for other input variables.

Notice that the graphs colored green indicate how the highlighted table will be filled:

- If a row of graphs is highlighted, the table is being filled by the indicated model evaluation (the value shown at the right of the row).
- If the column of graphs is green, the table is being filled by the indicated input variable (shown in the edit box above the column).

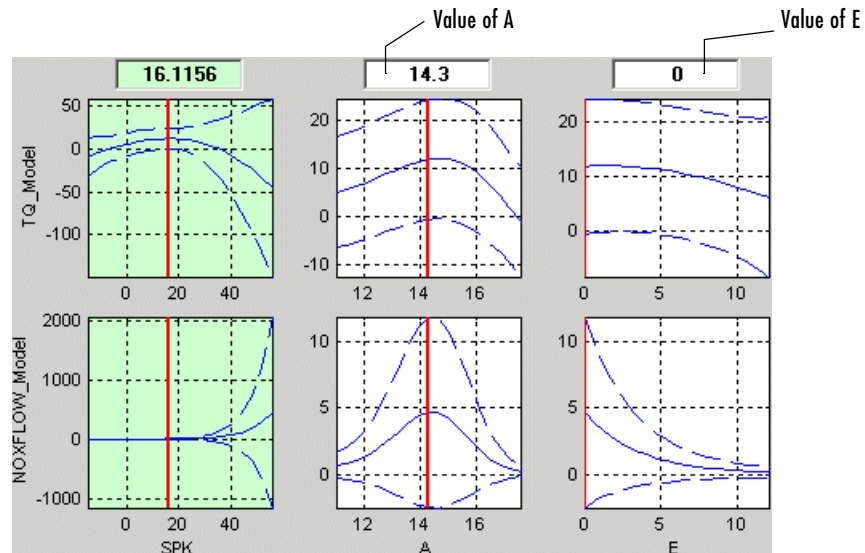
The next sections describe the following in detail:

- “Setting Values of Other Variables” on page 10-11
- “Determining a Value at a Specific Operating Point” on page 10-13

Setting Values of Other Variables

Typically the models that you use to perform a tradeoff calibration have many inputs. When calibrating a table of just one input, you need to set values for the other inputs.

Graphs in Table View



Setting Values for Individual Operating Points

To set values for inputs at individual operating points,

- 1 Highlight the operating point in the lookup table.
- 2 Use the edit boxes to specify the values of the other variables.

In the preceding example, the spark table is selected (the SPK graph is colored green). You have to specify the values of AFR (A) and EGR (E) to be used:

- 1 Select the spark table from the branch menu.
- 2 Click in the edit box for A and set its value to 14.3.
- 3 Click in the edit box for E and set its value to 0.


Setting Values for All Operating Points

For example, if you are using a tradeoff to calibrate a table for spark angle, you might want to set the initial values for tables of air/fuel ratio (AFR) and exhaust gas recycling (EGR).

To set constant values for all the operating points of one table,

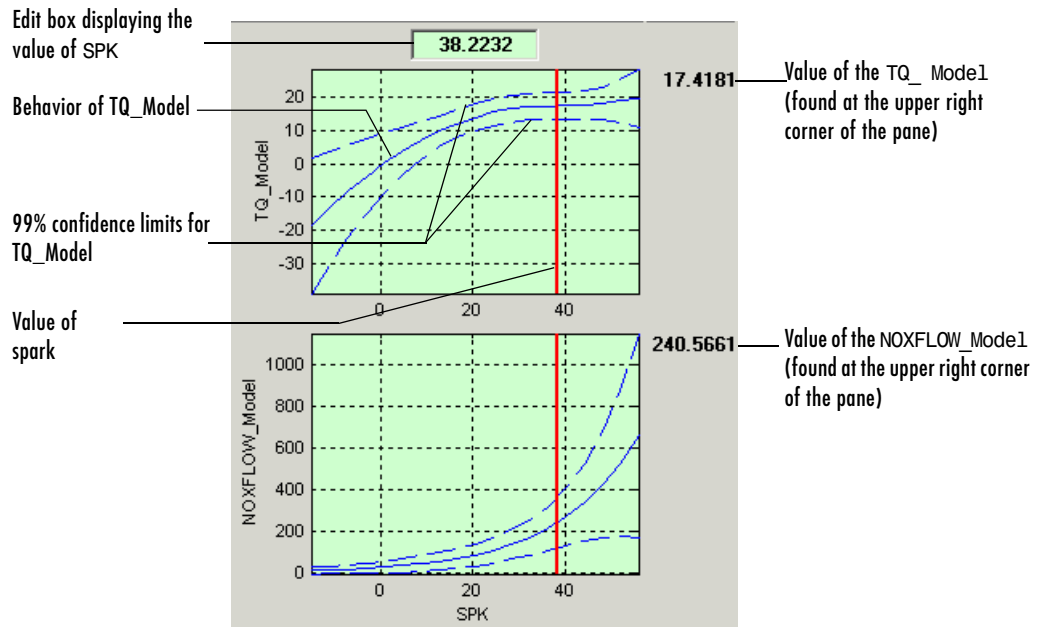
- 1 Highlight the table in the branch display.
- 2 Select one operating point in the table.
- 3 Enter the value of the cell.

This automatically adds this cell to the extrapolation mask.

- 4 Click  to extrapolate over the entire table.

This fills the table with the value of the one cell.

Determining a Value at a Specific Operating Point




Performing a tradeoff calibration necessarily involves the comparison of two or more models.

For example, in this case, the calibrator uses his or her judgment to select a value of spark that gives acceptable values for both the torque and the NOX flow models.

To select a value of spark, do one of the following:

- Click the edit box as shown above and enter the required value.
- Drag the red line.

Once you determine the value of your variable at this operating point, you apply this value to the table by pressing **Ctrl+T**, by selecting **Table -> Apply Point**, or by clicking  (Add Point) in the toolbar.

Right-Click Menu

Right-clicking a graph enables you to

- Find Nearest Turning Point
- Find Global Maximum
- Find Global Minimum

Using Zoom Controls on the Graphs

To zoom in on a particular region, click with both mouse buttons simultaneously and define the region as a rectangle.

To zoom out to the original graph, double-click the selected graph.

Note Zooming on one graph adjusts other graphs to the same scale.

View Options

Selecting the **View** menu offers you the following options:

- **History**
This opens the **History** display. For information, see “Using the History Display” on page 15-5.
- **Show/Hide Factors**
This opens a dialog box that allows you to show or hide factors. This is particularly useful if you are trading off models that have a large number of factors.
- **Show/Hide Models**
This opens a dialog box that allows you to show or hide models. This is particularly useful if you are trading off a large number of models.
- **Show Predicted Error**
When you select this, the graphs display the 99% confidence limits for the models.
- **Same Y Limits On Graphs**
When you select this, all the graphs share a common *y*-axis.


Using Regions

A region is an area that defines locally where to extrapolate before globally extrapolating over the entire table.

For example, consider filling a large table that has twenty breakpoints for each normalizer by extrapolation. Two problems arise:

- To have meaningful results, you need to set values at a large number of operating points.
- To set values at a large number of operating points takes a long time.

To overcome this problem, you can

- 1 Define regions within the lookup table.
- 2 In each region, set the values of some operating points.
- 3 Click  to fill the table by extrapolation.

Each region is filled by extrapolation in turn. Then the rest of the table is filled by extrapolation. The advantage of using regions is that you can have more meaningful results by setting values for a smaller number of operating points.

Tradeoff Table


	500.000	1000.000	1500.000	2000.000	2500.000
0.100	22.028	29.094	32.197	32.967	32.383
0.200	10.040	18.056	24.116	28.781	27.388
0.300	-0.000	10.517	19.067	22.467	22.763
0.400	0.640	10.256	18.444	19.268	17.475
0.500	1.731	8.761	13.162	13.511	11.641
0.600	0.891	5.396	7.895	7.963	6.423
0.700	0.027	3.368	5.342	5.857	5.548
0.800	0.128	3.337	5.809	7.606	8.958
0.900	0.964	4.447	7.609	10.517	13.283
1.000	1.929	5.630	9.156	12.488	15.546

Cells are colored

- Yellow if they form part of the extrapolation mask
- Blue if they are part of a region


- Gold if they are part of the extrapolation mask and part of a region

Defining a Region

- 1 Highlight the rectangle of cells in your table.
- 2 Click  to define the region.

The cells in the region are colored blue.

Clearing a Region

- 1 Highlight the rectangle of cells in your table.
- 2 Click  to clear the region.

Multimodel Tradeoffs

There are two types of tradeoff that you can add to your session, a tradeoff of independent models, as described earlier (see “Performing a Tradeoff Calibration” on page 10-2), or a tradeoff of interconnected models (a multimodel tradeoff).


A multimodel tradeoff is a specially built collection of models from the Model Browser.

You can build a series of models so that each operating point has a model associated with it. In the Model Browser, you can export models for a multimodel tradeoff from the test plan node. The models must be two-stage and must have exactly two global inputs.

The procedure for calibrating by using a multimodel tradeoff follows:

- 1** Add the multimodel tradeoff. (See the following section, “Adding a Multimodel Tradeoff” on page 10-18.)
- 2** Calibrate the tables. (See “Calibrating Using a Multimodel Tradeoff” on page 10-20.)
- 3** Export your calibration. (See “Exporting Calibrations” on page 7-21.)

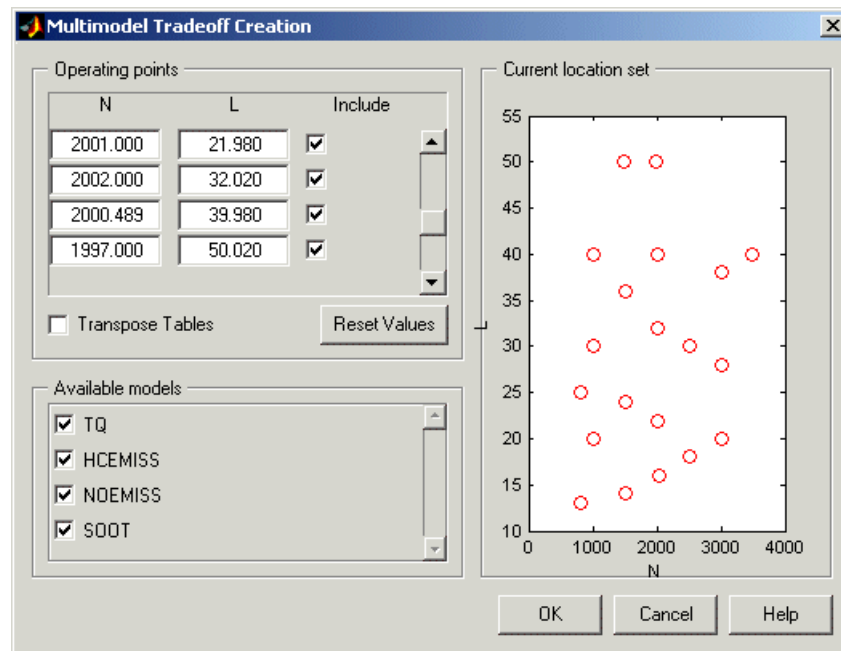
When you calibrate the tables in a multimodel tradeoff, you can only adjust a value in the tables if there is a model defined at this operating point.

The multimodel is only defined for certain cells in the tradeoff tables. These are the operating points that were modeled using the Model Browser part of the toolbox. These cells are colored purple in the table. At each of these operating points, you can use the model to trade off, and by doing this you can adjust the value in the table. The multimodel is not defined for all other cells in the table and you cannot edit these cells (but they can be filled by extrapolation). You trade off values at each of the model operating points in exactly the same way as when using independent models, as described in “Determining a Value at a Specific Operating Point” on page 10-13. When you have determined table values at each of the model operating points, you can fill the whole table by extrapolation by clicking . See “Filling the Table by Extrapolation” on page 9-19.

Adding a Multimodel Tradeoff

To add a multimodel tradeoff to your session,

- 1 Select **File** -> **New** -> **Multimodel Tradeoff**. The file must have been exported from the MBC Model Browser using the **Tradeoff** button (only enabled for two-stage models with exactly two global inputs).
- 2 Select the correct file to import and click **Open**. This opens a dialog box.



- 3 In the **Operating points** pane, you can clear the check boxes for any operating points that you do not want to import.

Notice that the operating points are displayed graphically in the **Current location set** pane. If an operating point is deselected, it is displayed as gray here, rather than red.

CAGE creates tables for all the models and input variables, with breakpoints at all the operating points.

- 4** You can adjust any of the operating points to reduce the number of breakpoints.

For example, in the session pictured, there are operating points at values of 2001, 2002, 2000.489, and 1997. This results in breakpoints in the table at each of these four values. However, all four operating values are very close to 2000 and might all have been intended to run at exactly 2000. You can choose to adjust all these to 2000 by typing in the edit boxes. The table then has a single breakpoint at 2000 instead of the four closely spaced breakpoints. You can click **Reset Values** to return to the original operating points.

- 5** In the **Available models** pane, clear any model that you do not want to import.

For example, you might want to perform a tradeoff of soot (SOOT) and torque (TQ) in the preceding example. Clear the check boxes for HCEMISS and NOEMISS.

- 6** Click **OK**.

When you import the multimodel tradeoff, you import the tables and normalizers, so you do not have to calibrate the normalizers.


Note When you calibrate the tables, you can only adjust the values of the tables at the operating points defined for the models. These are colored purple in the table.

You can now calibrate your tables. See the next section, “Calibrating Using a Multimodel Tradeoff” on page 10-20.

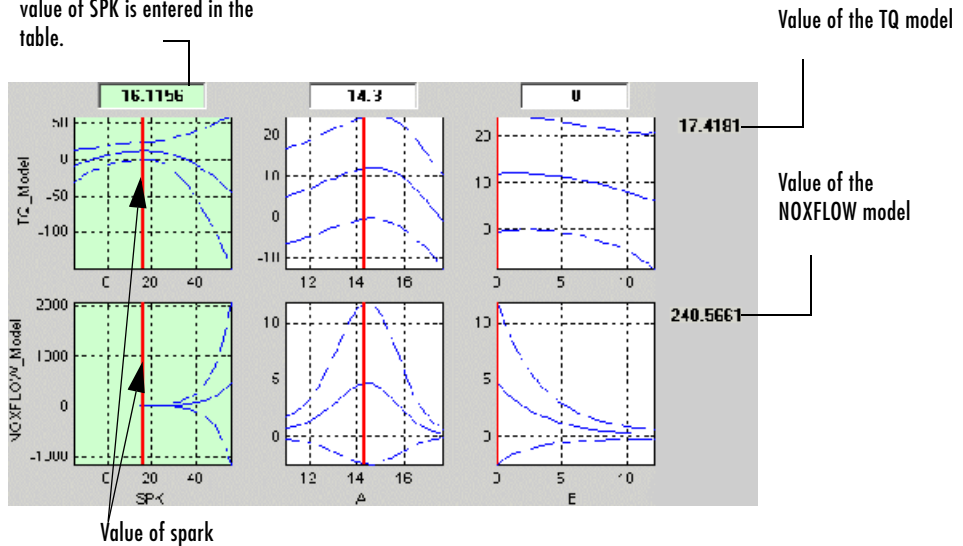
Calibrating Using a Multimodel Tradeoff



Each purple (editable) operating point in your tables has a model defined at that point. You use the display of these models to help you trade off values at these points to fulfill your aims in exactly the same way as when using independent models in “ordinary” tradeoff mode, as described in “Determining a Value at a Specific Operating Point” on page 10-13.

- 1 Change input values by dragging the red lines on the graphs or by typing directly into the edit boxes above the graphs.
- 2 Look at the model evaluation values (to the right of each row of graphs) and the input variable values (in the edit boxes above the graphs) to see if they meet your requirements.

Remember that the green highlighted graphs indicate how the selected table is filled: if a row is green, the model evaluation value (to the right) fills the table at that operating point; if a column is green, the input variable value (in the edit box above) fills the table. See the example following; the SPK column of graphs is green, so the value of SPK in the edit box is entered in the table when you click the Add Point button ().

This column is green, so this value of SPK is entered in the table.



- 3** When you are satisfied with the tradeoff given by the value of your variable at this operating point, you apply this value to the table by pressing **Ctrl+T**, selecting **Table -> Apply Point**, or clicking  in the toolbar.
- 4** When you have determined table values at each of the model operating points, you can fill the whole table by extrapolation by clicking . See “Filling the Table by Extrapolation” on page 9-19.

You can then export your calibration; see “Exporting Calibrations” on page 7-21.

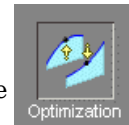
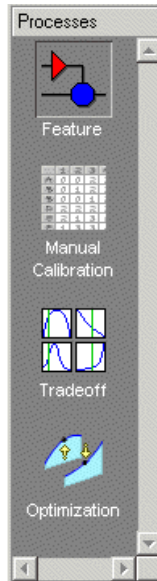
Optimization in CAGE

This section includes the following topics:

- | | |
|---|--|
| Using the Optimization View (p. 11-2) | An introduction to setting up your session for optimizations. |
| Setting Up Optimizations (p. 11-4) | Creating and running optimizations. You use the Optimization Wizard to choose your algorithm, algorithm options, and free variables. You can set up objectives, constraints, and operating point sets either in the wizard or from the main Optimization view, where you can then run optimizations. |
| Optimization Output Node (p. 11-25) | Using the optimization output views to investigate your results and select and export your preferred solutions. |
| Automated Tradeoff (p. 11-33) | How to apply an optimization to a tradeoff table. |
| User-Defined Optimization (p. 11-37) | An overview of the process of customizing the optimization template to use your own optimization routines in CAGE. |
| Optimization Template (p. 11-45) | The structure of the template for implementing your own custom optimization functions. |
| List of Optimization Functions (p. 11-47) | Detailed information on all the methods available for writing your own optimization functions. |

Using the Optimization View

Optimization functionality is one of the CAGE processes. The **Optimization** button can be found in the left-hand **Processes** pane.



To reach the **Optimization** view, click the  button.

Here you can set up and view optimizations. As with other CAGE processes, the left **Optimization** pane shows a tree hierarchy of your optimizations, and the right panes display details of the optimization selected in the tree. When you first open the **Optimization** view both panes are blank until you create an optimization.

As for other CAGE processes, you must set up your session for an optimization. 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. In this case you also need to set up this set of points using the Data Sets view. The steps required are

- 1** Import a model or models.
- 2** Define an operating point set if required.
- 3** Set up a new optimization.

Optimization functionality in CAGE is described in the following sections:

- “Setting Up Optimizations” on page 11-4
- “Optimization Output Node” on page 11-25

Once you have set up an optimization you can apply it to a region in a tradeoff table. See “Automated Tradeoff” on page 11-33

You can define your own optimization functions for use in CAGE. See “User-Defined Optimization” on page 11-37

There is also a tutorial to guide you through the optimization functionality. See “Tutorial: Optimization and Automated Tradeoff” on page 6-1.

Setting Up Optimizations

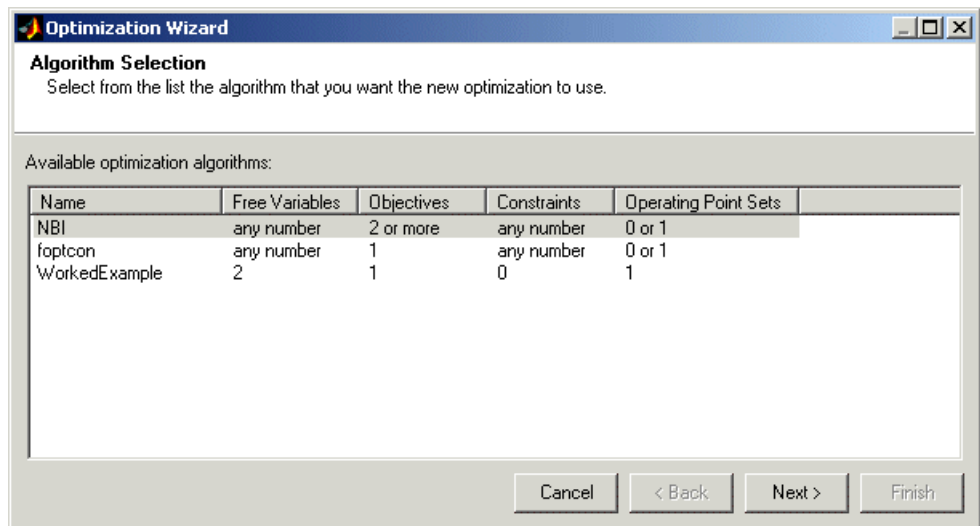
Normally you run an optimization on a session you already have open. For an example session you could open the `tradeoffInit.cag` file in the `mbctraining` folder.

- To create a new optimization, select **File -> New -> Optimization**.

This takes you to the **Optimization Wizard**, which leads you through the steps of choosing the optimization to run, specifying the number of variables to optimize over (unless this is predefined by the function), and linking the variables referenced in the optimization to CAGE variables.

Optimization Wizard Step 1

First you must choose your algorithm. The first screen of the **Optimization Wizard** is shown below.



The first two algorithm choices in the list are standard routines you can use for constrained single and multiobjective optimization.

- `foptcon` is a single-objective optimization subject to constraints. This function uses the MATLAB `fmincon` algorithm from the Optimization Toolbox.

- NBI stands for Normal Boundary Intersection algorithm, which is multiobjective and can also be subject to constraints.

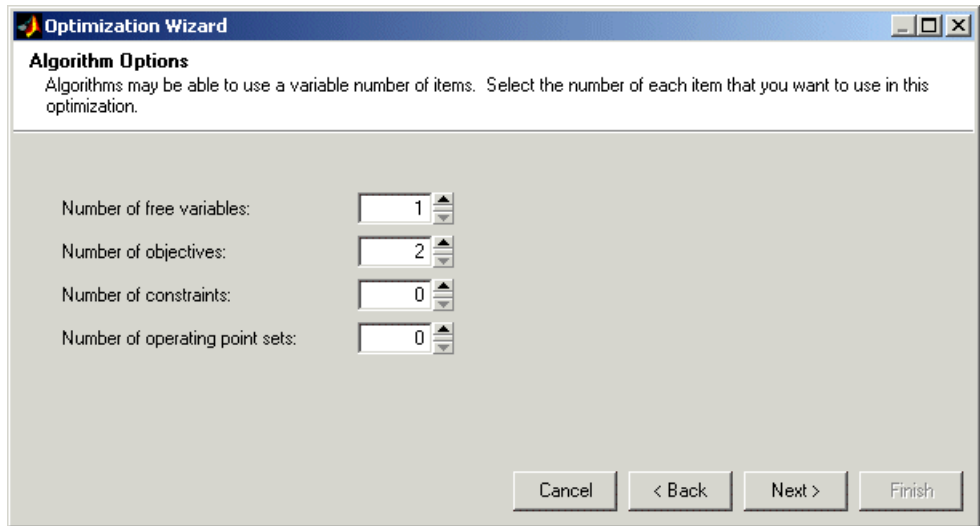
In many cases these standard routines are sufficient to allow you to solve your optimization problem. Sometimes, however, you might need to write a customized optimization algorithm; to do this you can use the supplied template to modify for your needs. See “User-Defined Optimization” on page 11-37 for information. The Worked Example option is designed to show you how to use the modified template; see “Worked Example Optimization” on page 6-38 in the optimization tutorial. Any optimization functions that you have checked into CAGE appear in this list.

Note If you choose a user-defined optimization function at step 1, all choices in subsequent steps depend on the settings defined by that function. When writing user-defined optimizations you can choose to set predetermined algorithm options or allow the user to make selections on any subsequent screen of the **Optimization Wizard**.

Optimization Wizard Step 2

Here you select algorithm options for numbers of free variables, objectives, constraints, and operating point sets.

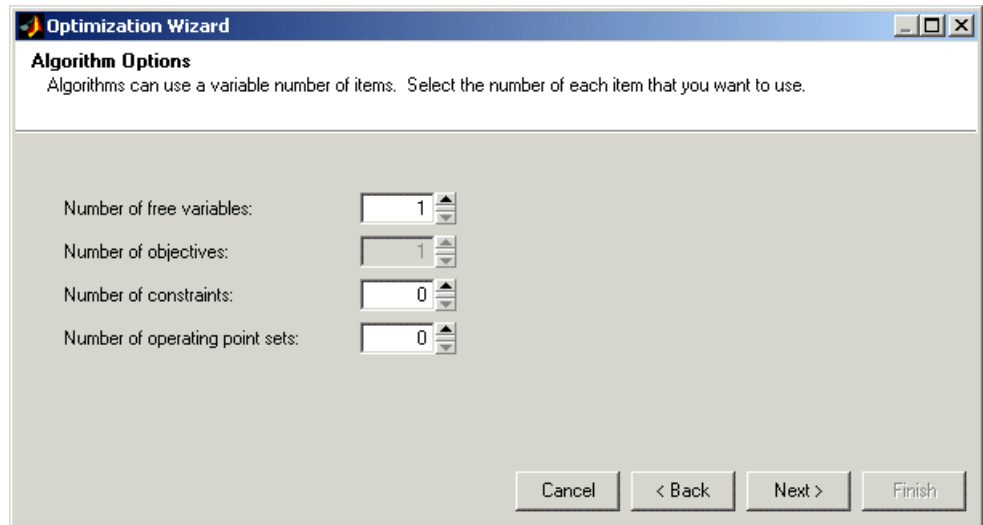
- If in step 1 you choose the first algorithm, NBI, and click **Next**, you see this:



NBI must have a minimum of two objectives, and you can choose as many free variables and constraints as you like. You can add constraints later if required. The operating point set defines the *fixed* variables. You can choose none or 1, and you can also add these later. An operating point set defines the points at which you want to run the optimization. The optimization tries to find the best values of the *free* variables.

If you leave this set to 0 (no operating point set), the optimization will run at a single point. This point is defined by the input values you set when you run the optimization. The defaults for these values are the set points of each variable, which you can set in the **Variable Dictionary** view.

- If in step 1 you select the foptcon algorithm and click **Next**, you get the following choices:

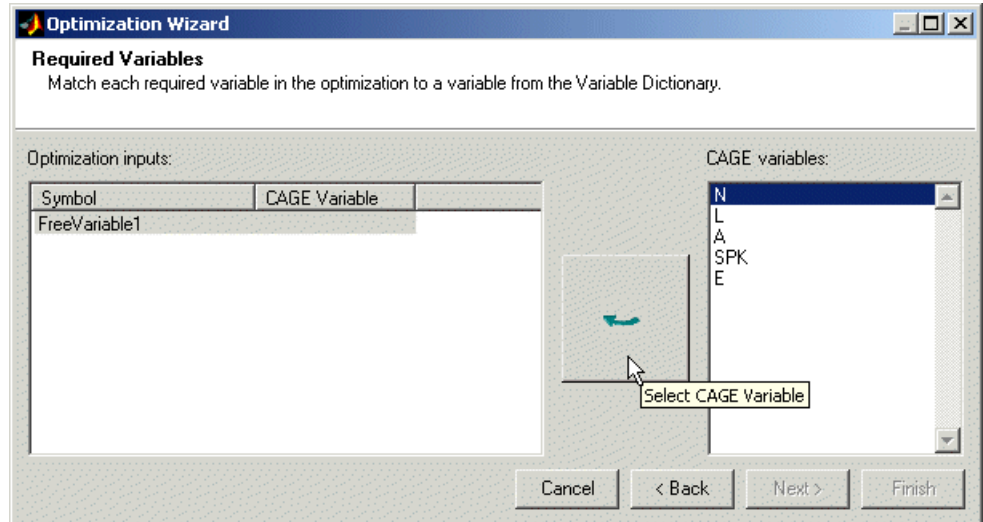


The foptcon algorithm can only have a single objective, so this control is not enabled. Choose the number of free variables and constraints you require. You can choose 0 or 1 sets of operating points.

Click **Next** to proceed to setting up free variables.

Optimization Wizard Step 3

You must select variables to link with the free variables used in your optimization.



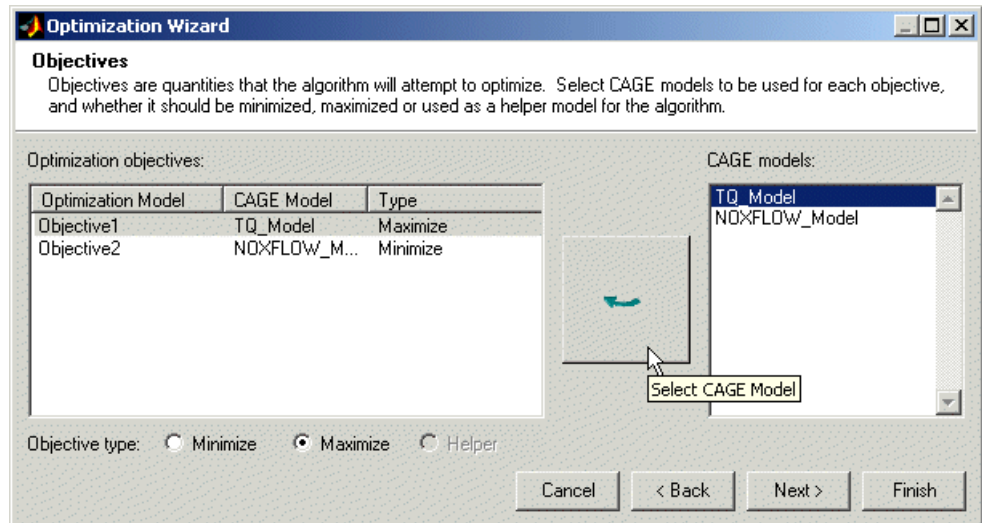
Use this screen to associate the variables from your CAGE session with the free variable(s) you want to use in the optimization. Select the correct pair in the right and left lists by clicking, then click the large button as indicated in the figure.

Once you have assigned your free variables here you can either click **Next** or **Finish**. This also applies to all later steps in the **Optimization Wizard**.

- If you click **Next** you proceed to further screens of the **Optimization Wizard** where you can set up objectives, constraints, and operating point sets.
- If you click **Finish** you return to the **Optimization** view in CAGE. You can set up your objectives, constraints, and operating point sets from the **Optimization** view instead of using the **Optimization Wizard**. You cannot run your optimization until objectives (and constraints and operating point sets if required) have been set up.

Optimization Wizard Step 4

You can set up your objectives here or you can set them up at the Optimization node in CAGE. See “Objective Editor” on page 11-21.



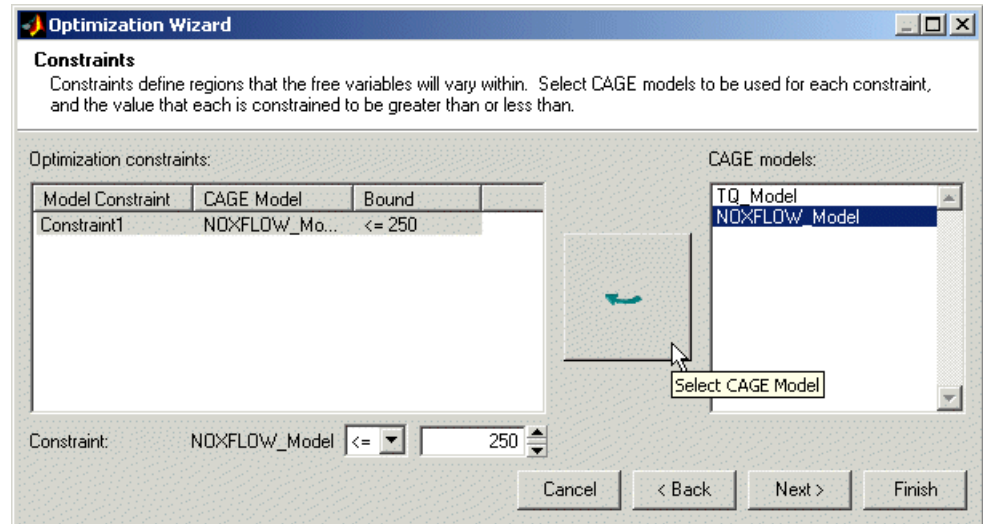
Here you can select which models from your session you want to use for the optimization, and whether you want to maximize or minimize the model output. The foptcon algorithm is for single objectives, so you can only maximize or minimize one model. The NBI algorithm can evaluate multiple objectives. For example, you might want to maximize torque while minimizing NOX emissions. Remember you can also define constraints later, for example, using emissions requirements.

You can also include ‘helper’ models in your user-defined optimizations, so you can view other useful information to help you make optimization decisions (this is not enabled for NBI or foptcon).

- Click **Next** to proceed to setting up constraints and operating point sets.
- Click **Finish** to complete the **Optimization Wizard** and return to the **Optimization** view.

Optimization Wizard Step 5

You can use models to define constraint regions that confine free variables. If you want to use constraints you can select them here, or add them at the Optimization node in CAGE. You can also add other types of constraints at the Optimization node. See “Constraint Editor” on page 11-23.



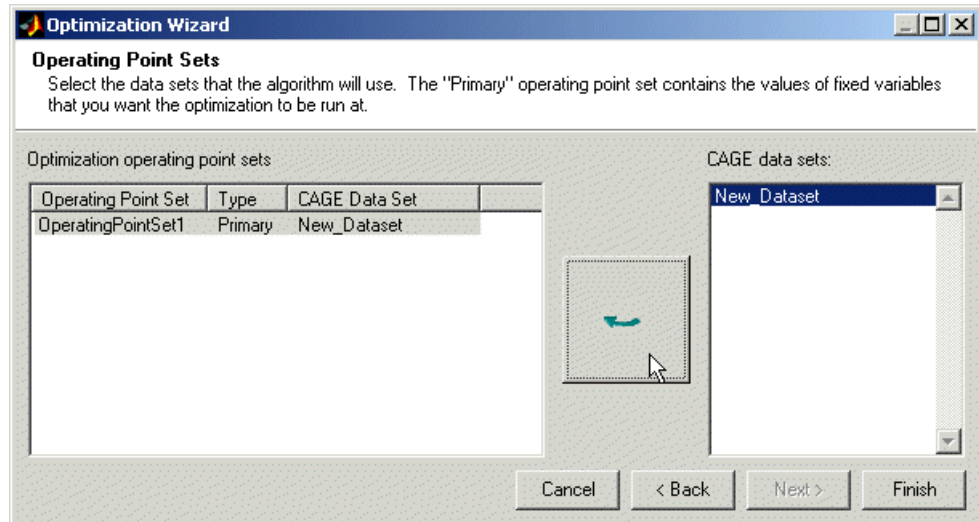
Select a model for each constraint by selecting a CAGE model and a model constraint and clicking the button to match them up.

For each constraint enter a value in the edit box. Select the operator to define whether the optimization output should be constrained to be greater than or less than the value. The example shown is NOXFLOW_Model1 <= 250.

- Click **Next** to proceed to setting up operating point sets.
- Click **Finish** to complete the **Optimization Wizard** and return to the **Optimization** view.

Optimization Wizard Step 6

If you want to use an operating point set you need to select a data set from your session to be used in the optimization. If you do not use an operating point set the optimization will run at a single point of your choosing. You can set up operating point sets here or at the Optimization node in CAGE.



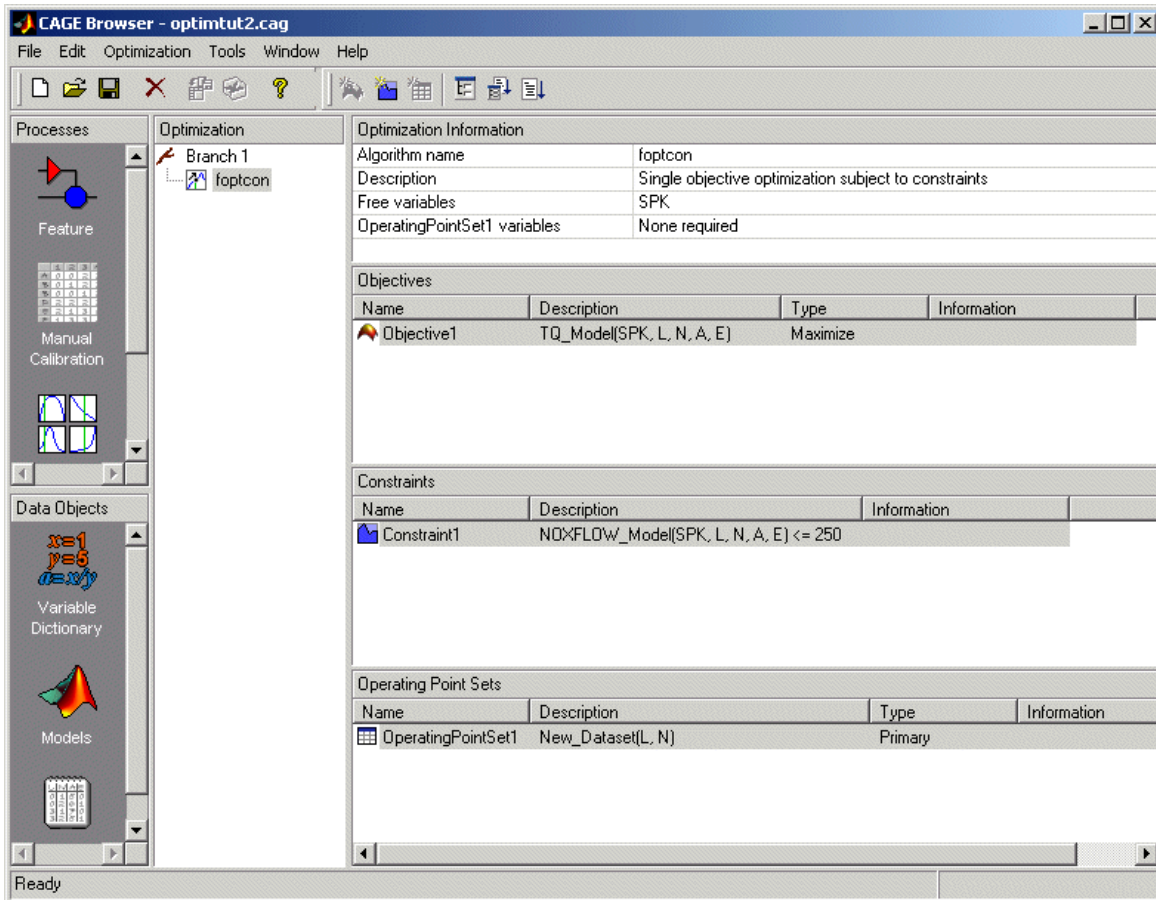
Select a CAGE data set and click the button to match it up with OperatingPointSet1. The Primary (first) operating point set contains the set of points defined by the fixed variables where you want the optimization to run.

User-defined optimizations allow you to add subsequent operating point sets. These can be used as “helper” data sets. You can use these to evaluate models over a different set of operating points during an optimization run. As an example, you could run an optimization at the points (N1, L1), (N2, L2), but an important quantity to monitor and possibly act upon is, say, temperature at points (N3, L3), (N4, L4). You can monitor this through the use of helper data sets to help you select optimization results. Helper data sets are not allowed for foptcon and NBI optimizations.

Click **Finish** to return to the Optimization node in CAGE.

Running Optimizations

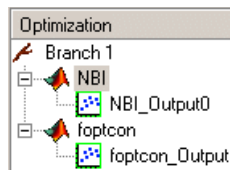
When you click **Finish** to complete the **Optimization Wizard**, you return to the Optimization node in CAGE. Your new optimization appears as a new node in the tree pane on the left, and the setup details appear on the right. An example follows:



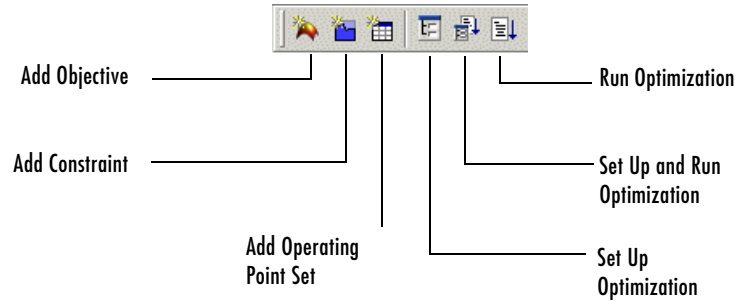
If your optimization is ready to run you can click Run Optimization in the toolbar to proceed. If you need to set up any objectives or constraints **Run** will not be enabled. If your optimization is ready to run you can also click **Set Up and Run Optimization** if you want to change settings such as number of required solutions and tolerances for termination.

- If you click **Set Up and Run Optimization**, you can change settings in the “Optimization Parameters Dialog”, then when you click **OK** the optimization process begins.
 - If the models to be evaluated require fixed variable inputs that are not specified by your operating point set, you see the **Set Constant Values** dialog. Here you are requested to set values for these fixed variable model inputs. These values will be used at every operating point. The defaults are taken from the set points that you can define in the **Variable Dictionary** view. Click **OK** when you are finished with these.
 - The **Free Variable Set Up** dialog appears, where you can set bounds and starting points for your free variables. These defaults are taken from the ranges and set points in the Variable Dictionary. Click **OK** when you are satisfied with these and the optimization runs.
- If you click **Run Optimization** instead, you do not see the optimization settings, but go straight to the **Set Constant Values** dialog and then the **Free Variable Set Up** dialog. Click **OK** in these dialogs and the optimization runs.

You will see a progress bar as the optimization proceeds. When it is finished, a new Output node appears under your Optimization node in the tree. Expand the Optimization node and click the Output node to see the displays. An example tree is shown. See “Optimization Output Node” on page 11-25.



Optimization Node Toolbar



- **Add Objective** — Adds an objective to your optimization (if enabled; remember foptcon can only have a single objective). You must double-click the new objective to open the **Objective Editor** and select a model and whether to maximize or minimize.
- **Add Constraint** — Adds a constraint to your optimization. You must double-click the new constraint (in the list of constraints) to open the **Constraint Editor** and set up the constraint.
- **Add Operating Point Set** — Adds an operating point set to your optimization (if enabled; NBI and foptcon can only have a single operating point set). You must double-click the new operating point set to select a data set to use.
- **Set Up Optimization** — Opens the **Optimization Parameters** dialog where you can change optimization settings such as tolerances and number of solutions. When you close the dialog the settings are saved but the optimization does not run.
- **Set Up and Run Optimization** — Opens the **Optimization Parameters** dialog where you can change optimization settings such as tolerances and number of solutions. When you close the dialog the optimization requests starting values and then runs.
- **Run Optimization** — Starts the optimization. CAGE requests starting values and then runs the optimization. See “Running Optimizations” above.

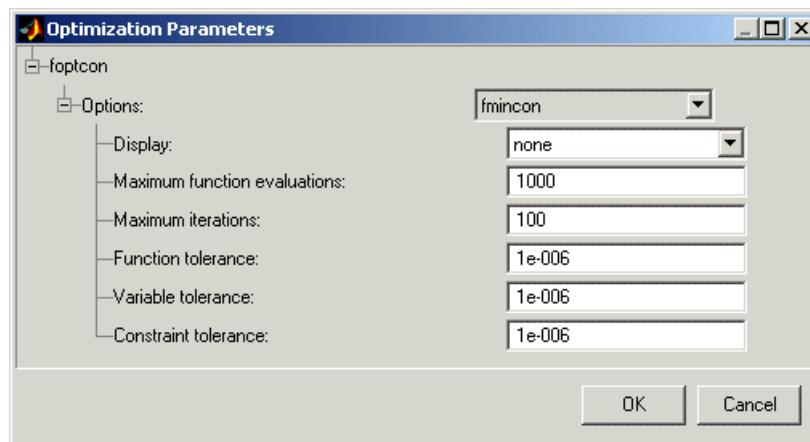
Optimization Parameters Dialog

The settings in this dialog are algorithm specific.

Following is an example showing the `foptcon` options. For the NBI options, see the next section, “NBI Optimization Parameters” on page 11-16.

`foptcon` Optimization Parameters

The `foptcon` optimization algorithm in CAGE uses the MATLAB `fmincon` algorithm from the Optimization Toolbox. `foptcon` wraps up the `fmincon` function so that you can use the function for maximizing as well as minimizing. For more information, see the `fmincon` reference page in the Optimization Toolbox documentation.

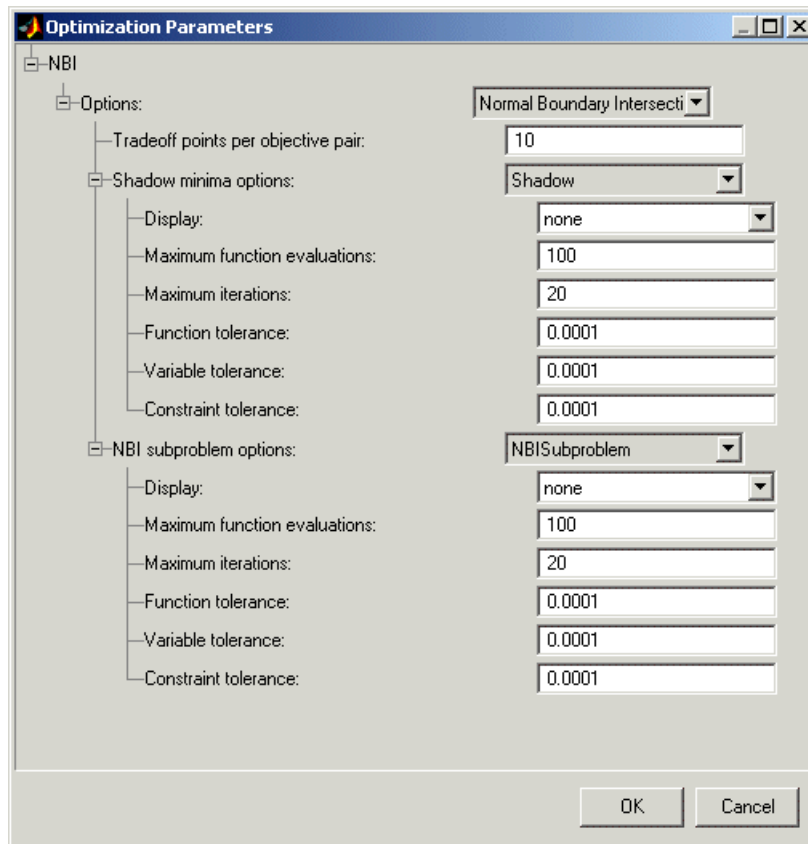


- **Display** — choose `none`, `iter`, or `final`. This setting determines the level of diagnostic information displayed in the MATLAB workspace.
 - `none` — No information is displayed.
 - `iter` — Displays statistical information every iteration.
 - `final` — Displays statistical information at the end of the optimization.
- **Maximum function evaluations** — Choose a positive integer.
Maximum number of function evaluations allowed
- **Maximum iterations** — Choose a positive integer.
Maximum number of iterations allowed

- **Function tolerance** — Choose a positive scalar.
Termination tolerance on the function value
- **Variable tolerance** — Choose a positive scalar.
Termination tolerance on the free variables
- **Constraint tolerance** — Choose a positive scalar.
Termination tolerance on the constraint violation

NBI Optimization Parameters

The example following shows the NBI options in the **Optimization Parameters** dialog.

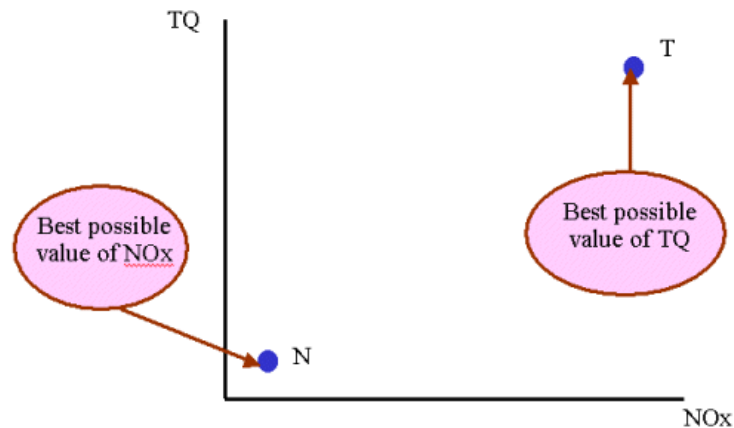


Background on the NBI (Normal Boundary Intersection Algorithm)

To understand the options for the NBI algorithm, some limited understanding of the algorithm is required. For more information on the NBI algorithm, see the NBI home page at the following URL:

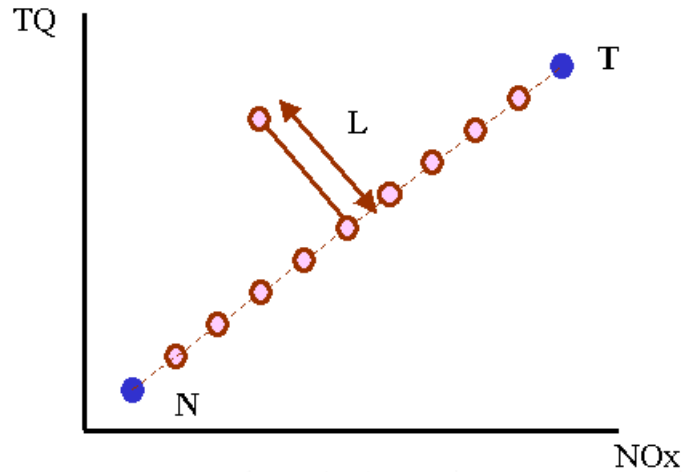
<http://www.caam.rice.edu/~indra/NBIhomepage.html>

The NBI algorithm is performed in two steps. The first step is to find the global extrema of each objective individually. This is called the shadow minima problem, and is a single-objective problem for each objective function. The MATLAB routine `fmincon` is used to find these extrema. Once these extrema are found, they can be plotted against each other. For example, consider an NBI optimization that simultaneously maximizes TQ and minimizes NOx emissions. A plot of the extrema against each other might resemble the following.

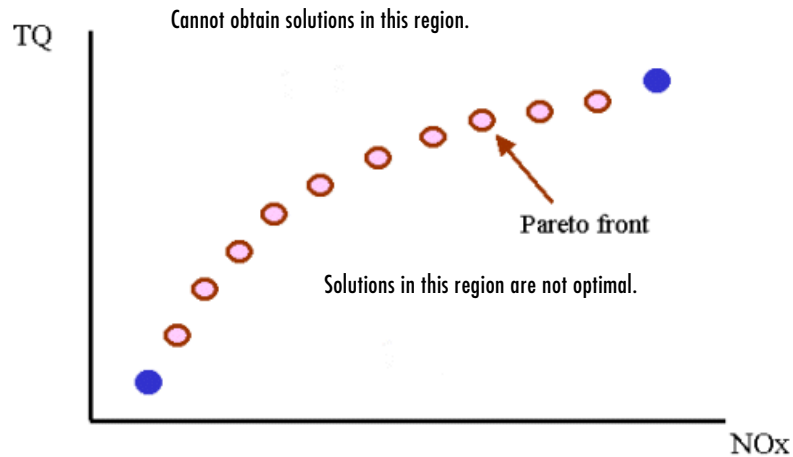


The second step is to find the "best" set of tradeoff solutions between your objectives. To do this, the NBI algorithm spaces N_{pts} start points in the $(n-1)$ hypersurface, S , that connects the shadow extrema. In the above example, S is the straight line that connects the points N and T . For each of the N_{pts} points on S , the algorithm tries to maximize the distance along the normal away from this surface (this distance is labeled L in the following figure). This is called the NBI subproblem. For each of the points, the NBI subproblem is a single-objective

problem and the algorithm uses the MATLAB `fmincon` routine to solve it. This is illustrated below for the TQ-NOX example.



The figure above shows spacing of the points between the extrema along the $(n-1)$ surface. The algorithm tries to maximize the distance L along the normal away from the surface. The following figure shows the final solution found by the NBI algorithm.



NBI Options

- **Tradeoff points per objective pair** (N_p)

The number of tradeoff solutions between your objectives that you want to find, N_{pts} , is determined by the following formula

$$N_{pts} = \binom{n + N_p - 2}{N_p - 1}$$

where

- N_p is the number of points per objective pair.
- n is the number of objective functions.

Note the following:

- For problems with two objectives ($n = 2$),

$$N_{pts} = N_p$$

- For problems with three objectives ($n = 3$),

$$Npts = \frac{Np(Np + 1)}{2}$$

- **Shadow Minima Options and NBI Subproblem Options**

As the NBI algorithm uses the MATLAB `fmincon` algorithm to solve the shadow minima problem and the NBI subproblems, the options here are identical to those for the `foptcon` library function. For more information on these options, see the previous section, “foptcon Optimization Parameters” on page 11-15.

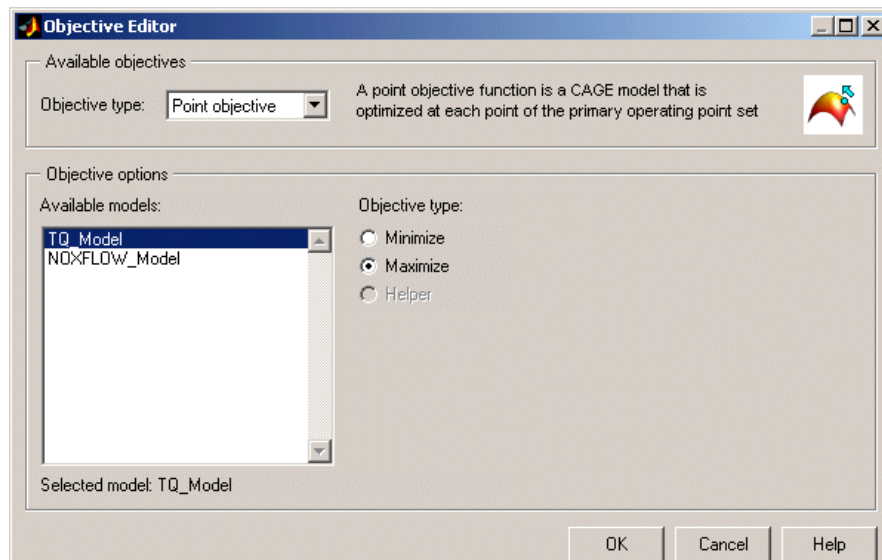
Objective and Constraint Editors

You can set up objectives and constraints from the main CAGE **Optimization** view, as well as within the **Optimization Wizard**. You can double-click objectives and constraints in the **Objectives** or **Constraints** panes to open the editors. You can also right-click and select **Edit**. You can also add and delete objectives and constraints by using the right-click menu.

You can run two types of optimizations, point optimizations and sum optimizations. Point optimizations look for the optimal values of each objective function at each point of an operating point set. A sum optimization finds the optimal value of a weighted sum of each objective function. The weighted sum is taken over each point in the operating point set, and the weights can be edited. For an example see the tutorial section “Sum Optimization” on page 6-32.

You need to use the **Objective Editor** and **Constraint Editor** to set up sum objectives and model sum constraints. You must do this to run weighted sum optimizations. You cannot set these up from the **Optimization Wizard**. You can also set up linear and ellipsoid constraints in the **Constraint Editor**, as for designs in the Model Browser part of the Model-Based Calibration Toolbox.

Objective Editor



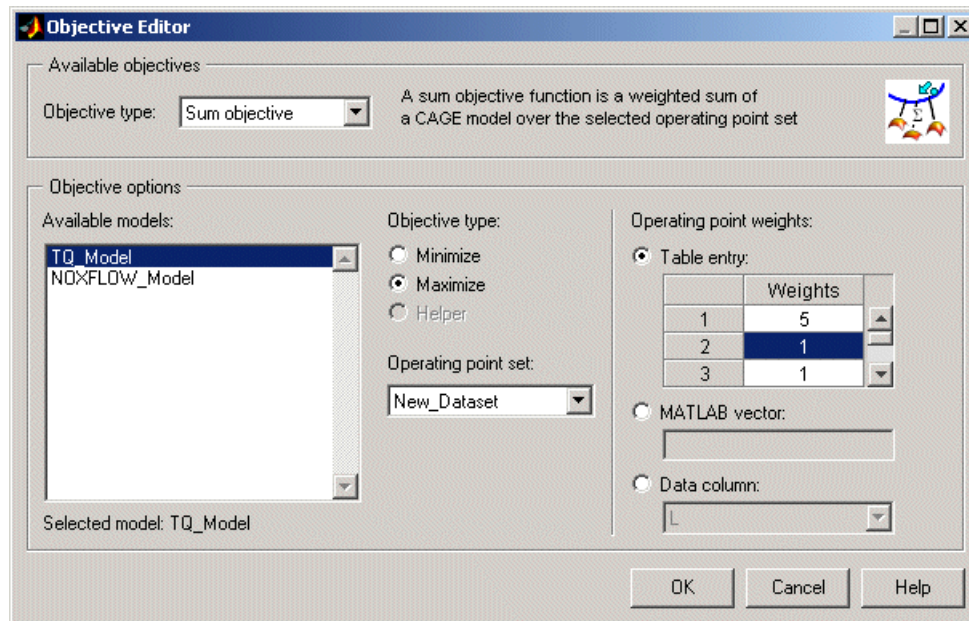
You can select Point objective or Sum objective from the **Objective Type** drop-down menu. Use sum objectives only for weighted sum optimizations; otherwise, use point objectives.

Point Objectives. The preceding example shows the point objective controls. Select which models from your session you want to use for the optimization, and whether you want to maximize or minimize the model output. The foptcon algorithm is for single objectives, so you can only maximize or minimize one model. The NBI algorithm can evaluate multiple objectives. For example, you might want to maximize torque while minimizing NOX emissions.

You can also include 'helper' models in your user-defined optimizations, so you can view other useful information to help you make optimization decisions (this is not enabled for NBI or foptcon).

These are the same options you can choose in the **Optimization Wizard**. See “Optimization Wizard Step 4” on page 11-9.

Sum Objectives. For weighted sum optimizations you must make all objectives sum objectives. See the following example.



As for point objectives, select which models from your session you want to use for the optimization, and whether you want to maximize or minimize the model output. You can select an operating point set.

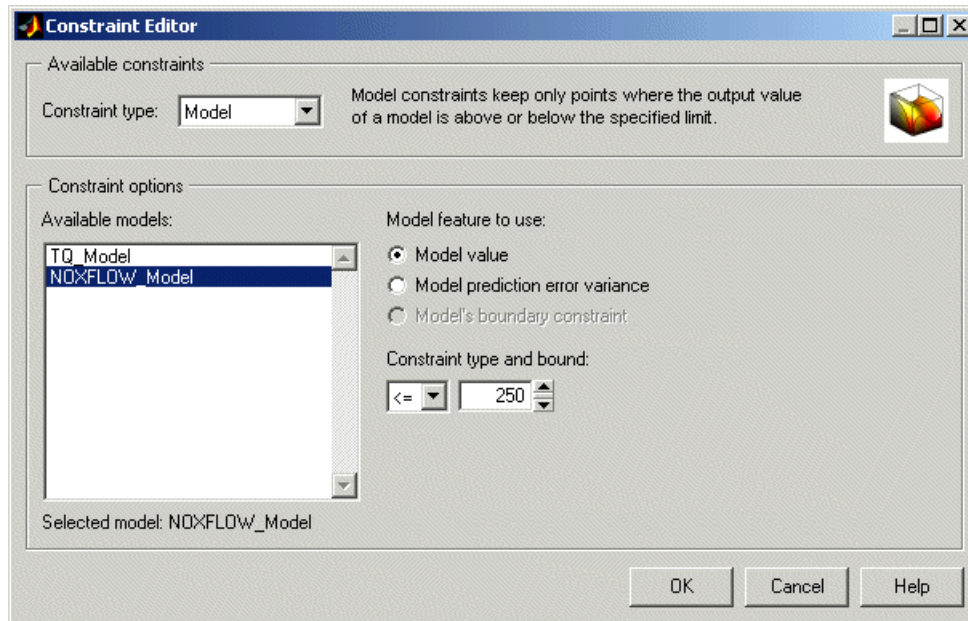
You can edit the weights to make certain operating points (for example, idle engine speed) more important, giving more flexibility to solutions for other points. The same weights are applied to each solution to calculate the weighted sums. You can select from these radio button choices:

- Edit weights directly using the table entry radio button choice.
- Specify a MATLAB vector.
- Specify a data column to supply the weights.

This is the same process as selecting weights for the **Weighted Pareto View**. See “Weighted Pareto View” on page 11-28. For a tutorial example of a sum optimization, see “Sum Optimization” on page 6-32 for more information.

Constraint Editor

Here you can set up Linear, Ellipsoid, Model, and Model Sum constraints. Select from the **Constraint type** drop-down menu. The model constraint settings are shown below. These are the same settings as seen in the **Optimization Wizard**. See “Optimization Wizard Step 5” on page 11-10.



For linear and ellipsoid constraints, see the section on Constraint Types in the Designs chapter of the Model Browser documentation. These are the same constraints you can apply to designs in the Model Browser part of the Model-Based Calibration Toolbox.

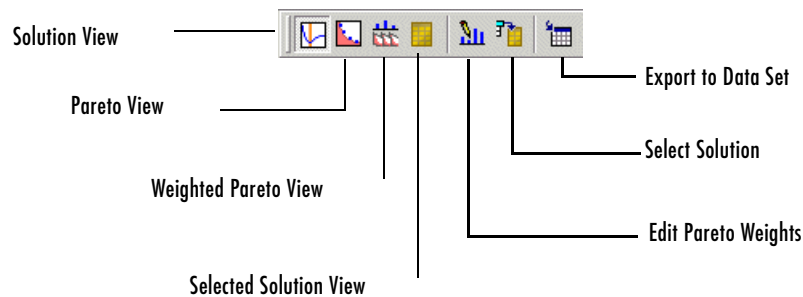
Model. Select a model. You can use the radio buttons to choose **Model value** or **Model prediction error variance** (PEV) to define your constraint, then enter a value in the edit box. Select the operator to define whether the optimization output should be constrained to be greater than or less than the value. If you imported an associated boundary constraint model from the Model Browser part of the toolbox, then you can select **Model's boundary constraint**. This constrains your optimization within the defined boundary model.

Model Sum. Use these for weighted sum optimizations. Choose a model value and operator as for a model constraint. You can also select an operating point set and apply weights as for a sum objective. See “Objective Editor” on page 11-21 and the tutorial “Sum Optimization” on page 6-32 for more information.

Optimization Output Node

When you have run an optimization an Output node appears in the optimization tree. When you select an Output node the **Optimization Output** views appear. The toolbar buttons at the output node determine what is displayed. The first view (shown by default) is the **Solution** view.

Optimization Output Node Toolbar



Export to Data Set — Exports the table visible in the current view only to a new data set called `Exported_Optimization_Data`.

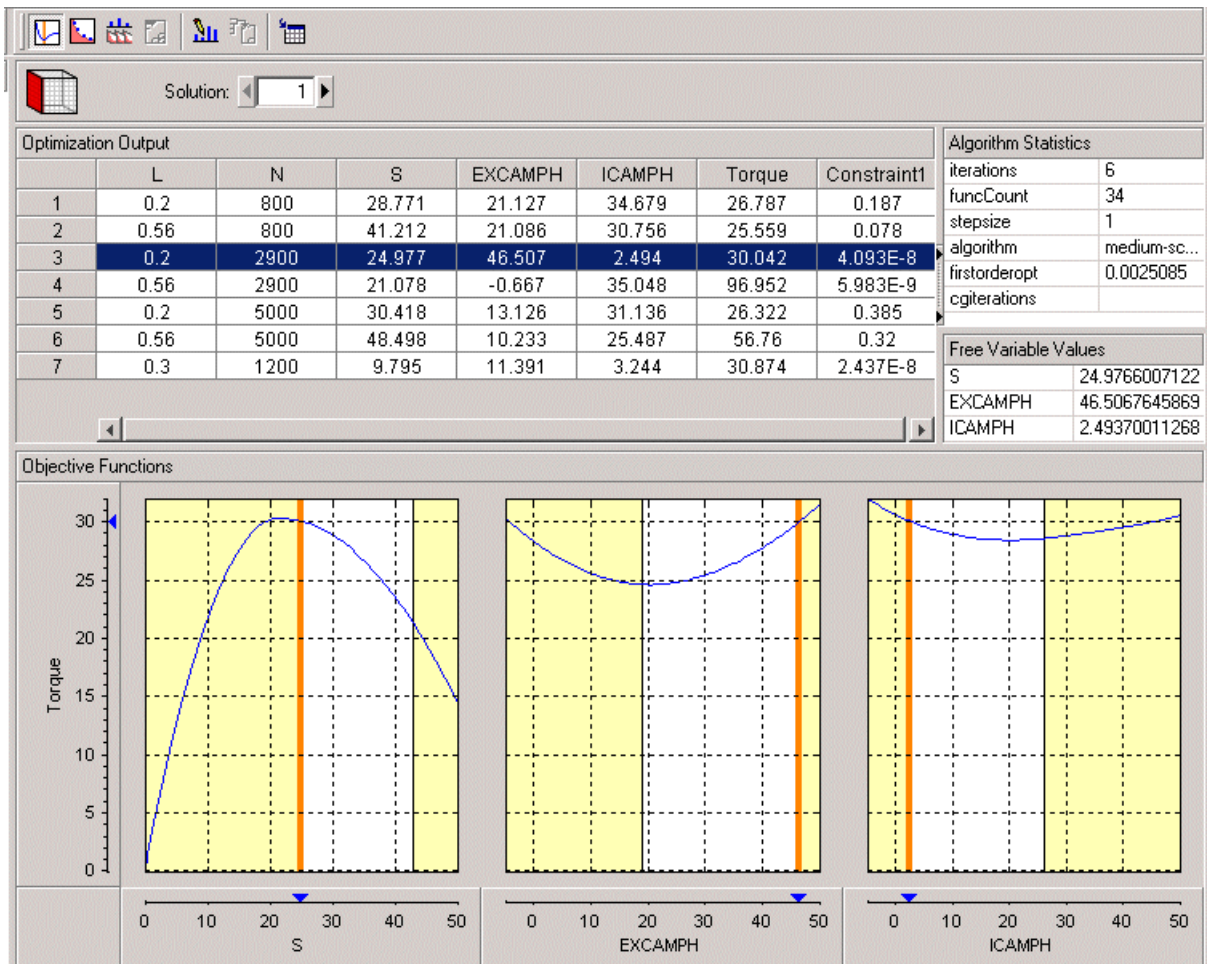
See

- “Solution View” on page 11-25
- “Pareto View” on page 11-27
- “Weighted Pareto View” on page 11-28
- “Selected Solution View” on page 11-31

Solution View

The **Solution** view shows one solution at all operating points.

The following example shows a **Solution** view display. The yellow areas show a boundary constraint model exported from the Model Browser part of the Model-Based Calibration Toolbox. All constraint regions in optimization displays (as in the rest of the toolbox) are shown in yellow.




The **Solution** view shows one solution at all operating points in the set; you can scroll through the solutions using the arrows or edit box at the top. Click in the table to make the graphs display the objective functions at the selected operating point.

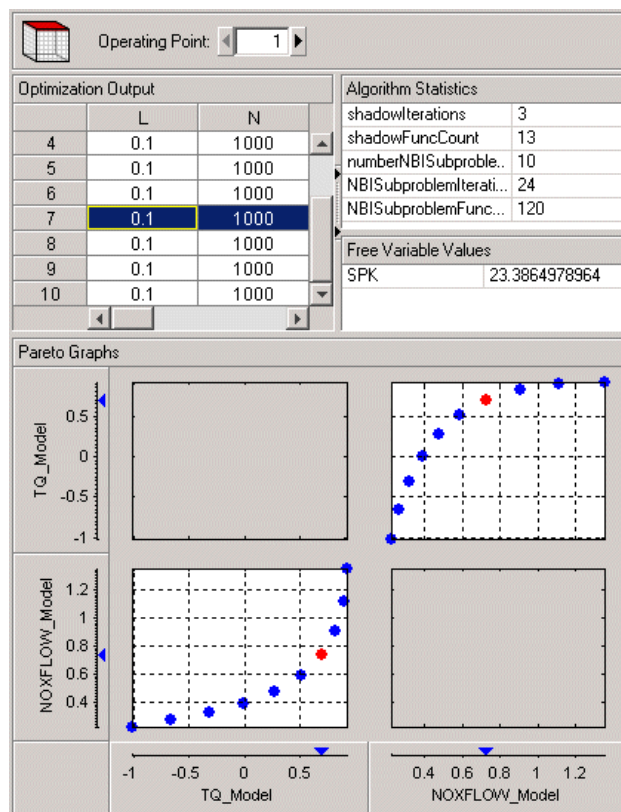
The table shows the selected solution at all operating points. Note that if you export the output to a data set (using the toolbar button) it is the current table that is exported. The graphs show the objective functions at the selected operating point, with the solution value in red.


Note that before you run an NBI optimization you can specify how many solutions you want the optimization to find, using the Set Up and Run Optimization toolbar button. For single-objective optimizations there is only one solution per operating point, so the **Solution** view is the only useful view.

For information on selecting best solutions at each operating point for subsequent export to a data set, see “Selected Solution View” on page 11-31.


Pareto View

The **Pareto** view (click ) shows all solutions at one operating point in the set; you can scroll through the operating points using the arrows or edit box at the top. Pareto plots show the available solutions with the current selection highlighted in red. An example is shown.

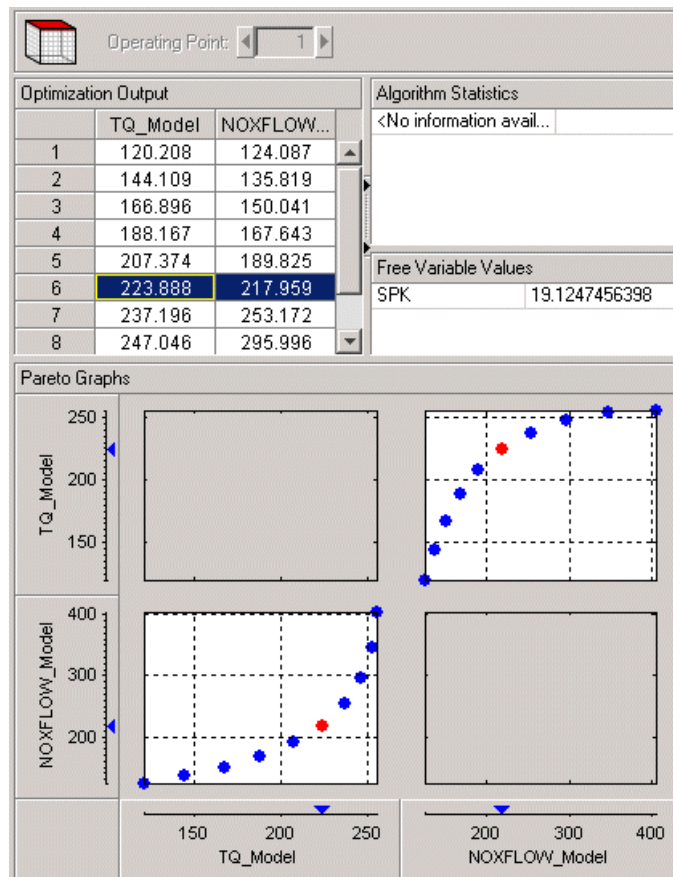



Use the plots to select the best solution for the operating point. In the example above there is a tradeoff to be made between torque and emissions. When you have decided which solution you want to use for the currently selected operating point you can select it as best by clicking Select Solution () in the toolbar. You cannot select solutions until you enable the **Selected Solutions** view. See “Selected Solution View” on page 11-31. You can also select best solutions in the **Solution** view.

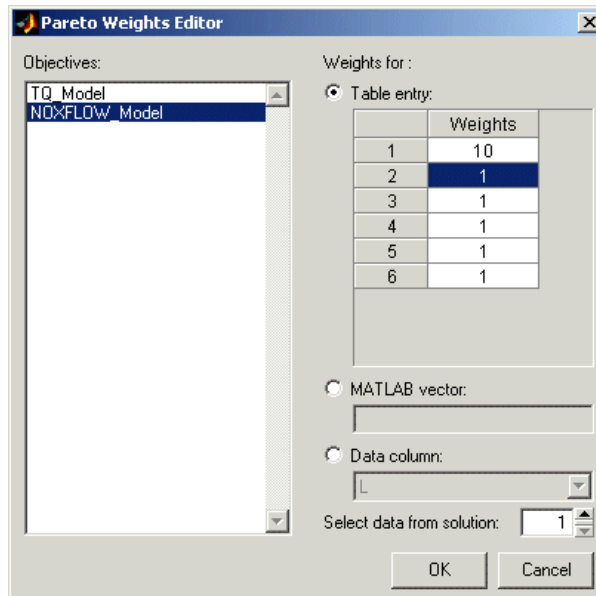
Weighted Pareto View

The **Weighted Pareto** view (click ) shows a weighted sum Pareto solution. This is a weighted sum of the output values at all operating points for each solution.

In the following example, the value in the NOXFLOW_Mode1 column in the first row shows the sum of the solution 1 values of NOX across all operating points. The second row shows the sum of solution 2 NOX values across all 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.



You can change the weights; for example, if you need a weighted sum of emissions over a drive cycle, you might want to give a higher weight to the value at idle speed. You can alter weights by clicking Edit Pareto Weights () in the toolbar. The **Pareto Weights Editor** appears.



Here you can select models to sum, 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 data column in your data set by selecting the other radio buttons. If you select **Data 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.

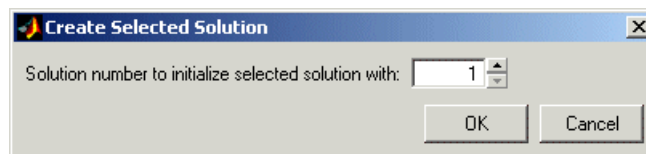
Note Weights applied in the **Weighted Pareto View** do not alter the results of your optimization as seen in other views. You can use the weighted sums to investigate your results only. You need to perform a sum optimization if you want to optimize using weighted operating points.


Selected Solution View

In a multiobjective optimization, there is more than one possible optimal solution at each operating point. You can use the **Selected Solutions** view to collect and export those solutions you have decided are optimal at each operating point.

Once you have enabled the **Selected Solution** view, you can use the plots in the **Pareto** view and **Solution** view to help you select best solutions for each operating point. These solutions are saved in the **Selected Solutions** view. You can then export your chosen optimization output for each point from the **Selected Solutions** view, in order to use your optimization output to fill tables.

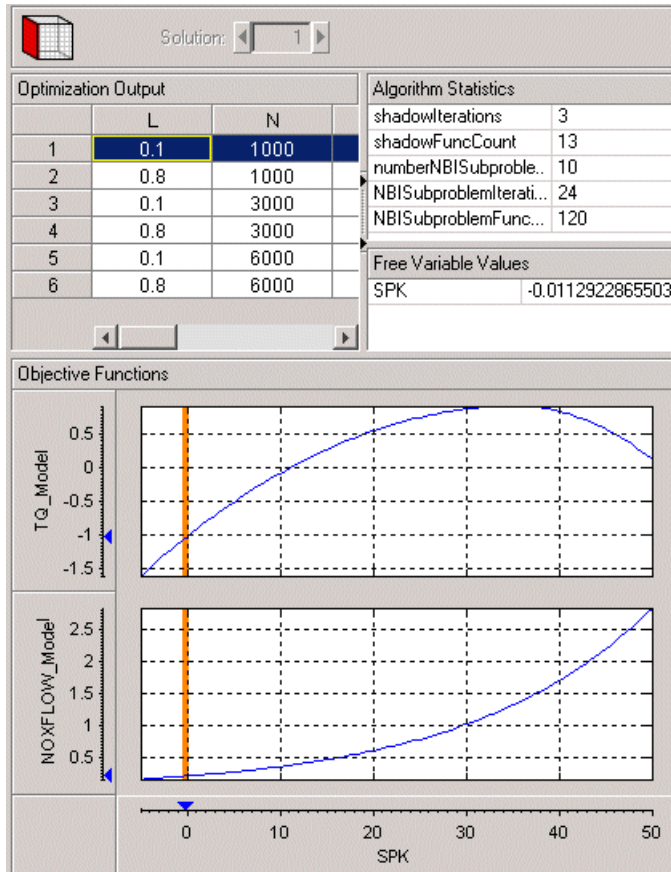
- 1 You cannot select best solutions until you have enabled the **Selected Solutions** view. Do this by selecting **Solution** -> **Selected Solution** -> **Initialize**.
- 2 A dialog called **Create Selected Solution** appears. The default 1 initializes the first solution for each operating point as the selected solution. You can edit the solution number here if you want. For example if you select 4, solution number 4 is initialized as the best solution for every operating point. When you click **OK**, the toolbar buttons for the **Selected Solutions** view and **Select Solution** are enabled.



Once you have enabled the **Selected Solutions** view, you can use the plots in the **Pareto** view and **Solution** view to help you select best solutions for each operating point. Click **Select Solution** () in the toolbar to select the current solution as best. These solutions are saved in the **Selected Solutions** view. This 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 for the second, and so on. You can then export your chosen optimization output for each point from the **Selected Solutions** view.

An example of the **Selected Solutions** view is shown. It looks similar to the **Solution** view, except the solution controls at the top are not enabled. You

cannot change solution number here. The solution chosen as best (in the **Pareto** or **Solution** views) for the currently selected operating point is shown in the grayed out edit box.



Automated Tradeoff

You can use automated tradeoff to run an optimization routine and fill your tradeoff tables. Once you have set up an optimization you can run an automated tradeoff. As with any other tradeoff you need at least one table. You can apply an optimization to a cell or region of a tradeoff table and the tradeoff values found are used to fill the selected cells. You can then fill the entire table by extrapolation. You can examine the optimization results in more detail at the Optimization Output node in the usual way.

There is an example automated tradeoff in the tutorial chapter, “Tutorial: Optimization and Automated Tradeoff” on page 6-1.

Using Automated Tradeoff

- 1 You need a CAGE session with some models and a tradeoff containing some tables.
 - See “Tradeoff Calibrations” on page 10-1 for instructions on setting up a tradeoff. You could use the tradeoff tutorial to generate a suitable example session.

You also need to set up an optimization before you can run an automated tradeoff. Free and fixed variables, objectives, and constraints must be set up.

- For an example work through the step-by-step tutorial to set up some optimizations and then apply them to a tradeoff table. See “Tutorial: Optimization and Automated Tradeoff” on page 6-1.
- 2 Go to the tradeoff table you want to automate. You must select some table cells to apply the optimization to. Select individual cells, or click and drag to select a rectangle of cells. The selected cells do not have to be adjacent. Note that if you define a large region with many cells it can take a long time to calculate a multiobjective optimization for each cell. Try a small region (say up to six cells) to begin with. Right-click selected cells and select **Define Region** or use the toolbar button.
 - 3 In the tradeoff view, click a cell in a region.
 - 4 To apply optimization: select **Table -> Automated Tradeoff**.

- The toolbox checks to see if the selected cell is part of a defined region. If not, an error dialog informs you that you can only perform automated tradeoff on cells in a defined region.
- If part of a region, then a dialog appears that allows an appropriate (defined below) optimization to be selected from the current project.

Note You must set up an optimization to run before you can perform an automated tradeoff. You do this in the **Optimization** view. See also “Setting Up Optimizations” on page 11-4.

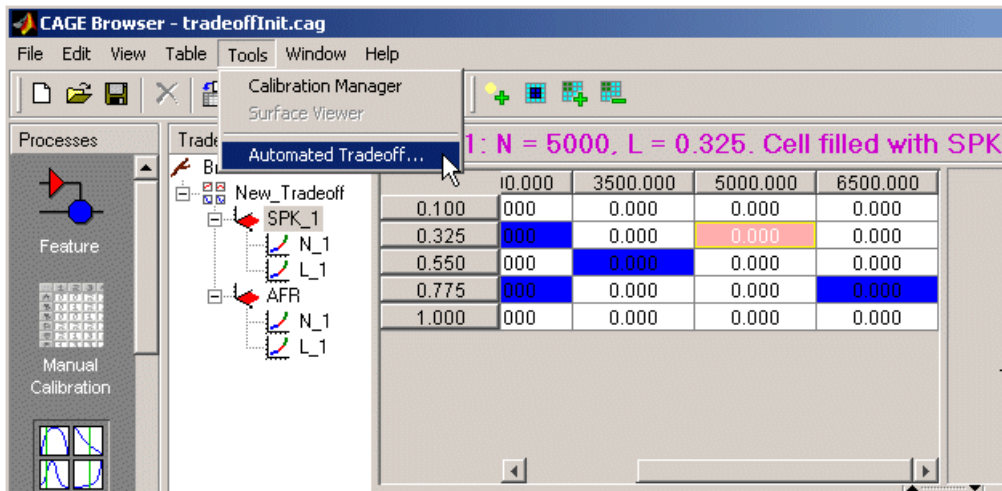
The cell/region is made into a data set of operating points that is linked to be the primary operating point set in the optimization. If the optimization object does not already have a primary operating set defined, then a new one is created. Note that any existing operating point set is reset to the previous state at the end of the automated tradeoff.

- 5 The optimization is run as if you were clicking **Run** from the Optimization node. See “Running Optimizations” on page 11-12. The dialogs for the free variable ranges/initial values and any other variable fixed values not specified by the operating point set appear and take the values from the data dictionary as usual. Click **OK** when you are satisfied with the values in these, and the optimization runs.

Results are placed in the tradeoff object, that is, values for the tables involving the free variables or values for the tables for constraint or objective models. If the routine applied gives more than one solution, for example, an NBI optimization, then only the values from the first solution are placed in the tradeoff tables. Every cell in the defined region is filled.

- 6 The cells of the region become part of the extrapolation mask (as if apply point has been applied); so if you want you can then click Extrapolate in the toolbar to fill the rest of the table from your optimized automated tradeoff.

The output from the optimization appears in the usual way (as an Output node under the optimization object). As for any other optimization you can use the Optimization Output node views to investigate the output. See “Optimization Output Node” on page 11-25.



What Are Appropriate Optimizations?

The list of all optimizations in the project is filtered. To be eligible for selection,

- The optimization must be ready to run (toolbar button enabled). The exception to this is when the primary operating point set has not been selected yet; it is selected in step 4 (see “Using Automated Tradeoff” on page 11-33) when you choose to apply an optimization to a region in a tradeoff table. The set of cells in the region you have selected becomes the operating point set for the optimization.
- The variables in the axes of the tradeoff tables must not be free variables in the optimization. For example, if one of the axes is speed, then speed cannot be a free variable.
- If the primary operating point set specifies the variables that must appear in it, then these must be a subset of the variables in the axes of the tradeoff tables. For example, if the primary operating point set requires variables Speed and Load, then these must be the axes variables in the tradeoff table.

Multimodel Tradeoff

For a multimodel tradeoff, things work slightly differently. The multimodel is only defined for certain cells in the tradeoff tables. These are the operating points that were modeled using the Model Browser part of the toolbox. Such cells are colored purple, and you should select these for running the automated tradeoff. You can select any region, but the optimization can only find values for the operating points defined by the multimodel.

User-Defined Optimization

User-defined optimizations are described in the following sections:

- “Implementing Your Optimization Algorithm in CAGE” on page 11-38 is an overview of how to customize the optimization template to use your optimization routines in CAGE.

There is a step-by-step guide to using the worked example provided to help you understand how to modify the template file to use your own optimization functions. See “Worked Example Optimization” on page 6-38 in the optimization tutorial.

- “Optimization Template” on page 11-45 describes the details of the available subroutines for customizing each section of the template file.

Introduction to Writing User-Defined Optimizations

In many cases the standard routines supplied for constrained single and multiobjective optimization (foptcon and NBI) are sufficient to allow you to solve your optimization problem. Sometimes, however, you need to write a customized optimization algorithm. This can be useful in many situations, for example,

- For an expert to capture an optimization process to solve a particular problem, for example, determination of optimal spark angle and exhaust gas recirculation rate on a port-fuel injection engine
- To implement an alternative optimization algorithm to those supplied
- To implement a complex constraint or objective that is only possible through writing M-code
- To produce custom output graphics

User-defined optimization functions in CAGE allow advanced users to write their own optimization routines that can access current CAGE data. In order to access the user function from CAGE, you must register the M-file with CAGE and place it on the MATLAB path. It is crucial that this function conform to the template specified. The following sections describe this process.

Implementing Your Optimization Algorithm in CAGE

At some point a CAGE optimization function calls on an algorithm to optimize the objective functions over the free variables. You can implement the algorithm in the CAGE optimization function as an external M-file. You use the template file as a basis for your optimization function. The best way to understand how to alter the template file to implement your own optimization algorithms is to compare it with the worked example, as described in the optimization tutorial.

- “Worked Example Optimization” on page 6-38 describes the process of using the worked example.
- “About the Worked Example Optimization Algorithm” on page 11-42 examines the coding involved in implementing an external optimizer in a CAGE optimization M-file.
- The optimization tutorial section “Creating an Optimization from Your Own Algorithm” on page 6-46 describes in detail the steps necessary to use an example optimization algorithm in CAGE.

Overview of Optimization Function Structure

The optimization function M-files have two sections. To compare these sections in the worked example with the template file on which it is based:

- 1 Locate and open the file `mbcOStemplate` in the `mbctraining` directory
- 2 Type the following at the command line to open the example:

```
edit mbcOSworkedexample
```

The two sections are the Options section and Evaluate section.

- The Options function section contains all the settings that define your optimization.
Here you set up all the free and fixed variables, objectives, constraints, operating point sets, and optimization parameters. CAGE interacts with the `cgoptimoptions` object, where all these settings are stored.
- The Evaluate function section contains the optimization routine.
You place your optimization routine under this section, interacting with CAGE (obtaining inputs and sending outputs) via the `cgoptimstore` object.

Any subfunctions called by your optimization routine should also be placed at the bottom of this section.

There is a detailed discussion of all the functions used to set up these sections of your optimization function in “Optimization Template” on page 11-45.

Note Be careful not to overwrite the worked example and template files when you are trying them out — save them under a new name when you make changes.

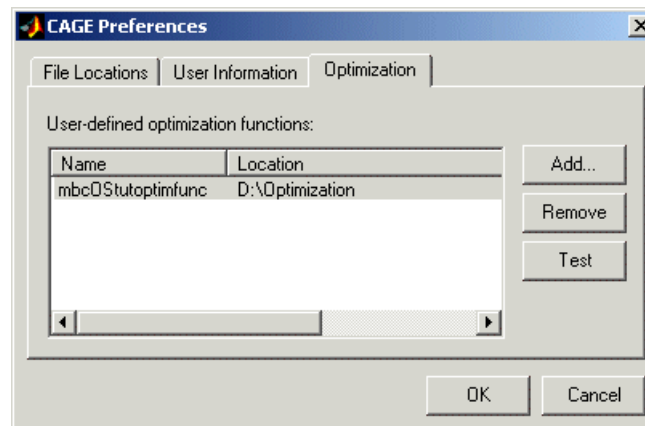
There is a step-by-step guide to using the worked example optimization function in the worked example section of the optimization tutorial. See “Tutorial: Optimization and Automated Tradeoff” on page 6-1.

Checking User-Defined Optimizations into CAGE

When you have modified the template to create your own optimization function, you must check it into the Model-Based Calibration Toolbox in order to use the function in CAGE. Once you have checked in your optimization function it appears in the **Optimization Wizard**. See “Optimization Wizard Step 1” on page 11-4.

To check a user-defined optimization into CAGE,

- 1 Select **File** → **Preferences**.
- 2 Click the **Optimization** tab and click **Add...** to browse to your M-file. Select the file and click **Open**. This registers the optimization function with CAGE. You need to do this when you customize your own optimizations.



- 3 You can click **Test** to check that the optimization function is correctly set up. This is a very useful function when you use your own functions; if anything is incorrectly set up the test results tell you where to start correcting your function.

You can see an example of this by saving a copy of the worked example file and changing one of the variable names (such as `afr`) to a number. Try to check this altered function into CAGE and the **Test** button will return an informative error specifying the line you have altered.

- 4 Click **OK** to dismiss the **CAGE Preferences** dialog and return to the CAGE browser.

Registered optimizations appear in the **Optimization Wizard** when you set up a new optimization.

About the Worked Example Optimization Algorithm

`mbcweoptimizer` is an example of a user-specified optimization that solves the following problem:

max TQ over (AFR, SPK) at a given (N, L) point.

`[bestafr,bestspk]=mbcweoptimizer(TQ, speed, load)` finds a maximum `(bestafr,bestspk)` to the function TQ.

TQ must be a function (or a function handle) that depends on AFR, SPK, SPEED, and LOAD only. The function must depend on the variables in that order. This routine does no variable matching.

- `[bestafr,bestspk]=mbcweoptimizer(TQ, speed, load, afrrng, spkrng)` finds a maximum `(bestafr,bestspk)` to the function TQ.
afrrng and spkrng are 1-by-2 row vectors containing search ranges for those variables.

- `[bestafr,bestspk]=mbcweoptimizer(TQ, speed, load, afrrng, spkrng, res)` finds a maximum `(bestafr,bestspk)` to the function TQ.
This optimization is performed over a res-by-res grid of (AFR, SPK) values. If res is not specified, the default grid resolution is 100.

- `[bestafr,bestspk]=mbcweoptimizer(TQ, speed, load, afrrng, spkrng, res, optimstore)` finds a maximum `(bestafr,bestspk)` to the function TQ within a CAGE optimization function.

`optimstore` is passed to this function when it is called from the Evaluate section subroutine of your optimization function. In this case, TQ must be a function handle that takes the inputs AFR, SPK, SPEED, LOAD, and OPTIMSTORE, in that order.

The Structure of the Worked Example

The best way to understand how to implement an external optimizer in a CAGE optimization function is to study the details of the example.

- To view the whole worked example M-file, at the command line, type
`edit mbcOSworkedexample`

The following code section is taken from the Evaluate section of the worked example file as an example.

```

80 % get the (N, L) points we want to perform the optimization at
81 speedloadpts = getdataset(optimstore, {'EngSpeed', 'Load'}, 'SpeedLoadPoints');
82
83 % For every (speed, load) point, find the optimum (afr, spk) using
84 % the mbcweoptimizer routine you have written
85 waitH = waitbar(0, 'Starting Optimizer', 'name', 'CAGE Worked Example Optimization');
86 bestafr = zeros(size(speedloadpts,1), 1);
87 bestspk = zeros(size(speedloadpts,1), 1);
88 for i =1:size(speedloadpts,1)
89     waitbar(i/size(speedloadpts, 1),waitH,['Optimizing Point: Speed = ', num2str(speedloa
90     [thisbestafr, thisbestspk] = mbcweoptimizer(@i_evalTQ, speedloadpts(i, 1), speedloadp
91     bestafr(i) = thisbestafr;
92     bestspk(i) = thisbestspk;
93 end
94 delete(waitH);
95 % set the best values calculated for the free variable(s) into the output data set
96 optimstore = setfreevariables(optimstore, [bestafr, bestspk]);
97
98 % return OK = 1 if everything went OK
99 OK = 1;
100 % return a measure of the goodness of optimization if required
101 OUTPUT.Algorithm = 'Brute force search';
102 % Update error message
103 errormessage = '';
104
105 % Set all information in the optimstore, and leave ....
106 % OK, output, errormessage
107 optimstore = setOutputInfo(optimstore, OK, errormessage, OUTPUT);

```

A

The line of code labeled A above calls the worked example optimization algorithm external to the optimization function. As with functions in the Optimization Toolbox, the first argument to the call to the optimizer is a function handle that evaluates the objectives at a given input point. We recommend you place the function pointed at by the function handle in the optimization file. If you do not place them in the same file you must make sure the evaluate function M-file is on the MATLAB path. As an example, the optimization evaluation function in the worked example optimization is shown in the code fragment following.

```

113 - %-----
114 - function y = i_evalTQ(afr, spk, speed, load, optimstore)
115 - %-----
116 -
117 - speedloadpts = getdataset(optimstore, {'EngSpeed', 'Load'}, 'SpeedLoadPoints');
118 - % Find where the current speed, load point is in the speedloadpoints dataset
119 - current_speedloadpt = [];
120 - for i = 1:size(speedloadpts, 1)
121 -     if isequal([speed, load], speedloadpts(i, :))
122 -         current_speedloadpt = i;
123 -         break;
124 -     end
125 - end
126 - y = gridevaluate(optimstore, [afr, spk], {'Torque'}, 'SpeedLoadPoints', current_speedloadpt);

```

The first four inputs to this function are the torque (in this case) model inputs. The final input is the `optimstore` object, where information about the optimization is stored. To evaluate objectives, there are two possible functions from the `optimstore` object that can be used — `evaluate` or `gridevaluate`. In the above example, the line of code referenced by B evaluates the torque model in the worked example at the `(afr, spk, speed, load)` input points.

The two subfunctions presented above are an example of how to implement an external optimizer in a CAGE optimization M-file.

See also the tutorial section “Creating an Optimization from Your Own Algorithm” on page 6-46. This example describes in detail the steps involved in incorporating an example algorithm into a CAGE optimization M-file.

Optimization Template

To view the template M-file, at the command line, locate the mbctraining directory and open the file

```
mbcOStemplate.m
```

There are two sections:

- The Options section. Here you can set up these seven attributes:

- Name
- Description
- Free variables
- Objective functions
- Constraints
- Operating point sets
- Optimization parameters

See “Methods of cgoptimoptions” on page 11-48 for information about setting up the options section.

- The Evaluate section. Place your optimization routine in here. CAGE calls this section when the **Run** button is clicked.

Your optimization interacts with CAGE through the cgoptimstore object and must conform to the following syntax:

```
optimstore = <Your_Optimization> (optimstore)
```

where <Your_Optimization> is the name of your optimization function.

Any subfunctions called by your optimization routine should also be placed at the bottom of the Evaluate section.

In order to access the information in the cgoptimstore, a number of functions are offered. These are described in the next section, “List of Optimization Functions” on page 11-47.

If you leave the `cgoptimoptions` function unchanged, your optimization function must be able to support the default options. That is, your optimization will have:

- One objective
- Any number of constraints (selected by the user in CAGE)
- Either no operating point set or one operating point set (selected by the user in CAGE)

List of Optimization Functions

Methods of `cgoptimstore`

Type `help cgoptimstore /method_name` to see online help (where *method_name* is the name of a function, such as 'getDataset').

These methods are available:

Methods of `cgptimstore`

Task	Command
Evaluate optimization objectives and constraints	<code>evaluate</code>
Grid evaluation of optimization objectives and constraints	<code>gridEvaluate</code>
Evaluate prediction error variance (PEV)	<code>pevEvaluate</code>
Grid evaluation of prediction error variance (PEV)	<code>gridPevEvaluate</code>
Get optimization properties	<code>get</code>
Get values from optimization operating point sets	<code>getDataset</code>
Get the initial free values for the optimization	<code>getInitFreeVal</code>
Get the number of rows in an optimization operating point set	<code>getNumRowsInDataset</code>
Get output information for the optimization	<code>getOutputInfo</code>
Get optimization parameter	<code>getParam</code>
Set the optimal values of the free variables	<code>setFreeVariables</code>
Set output information for the optimization	<code>setOutputInfo</code>

Methods of cgoptimoptions

You use these functions to set up all your optimization settings in the Options section of the file. You can set up any or all of these seven attributes:

- Name
- Description
- Free variables
- Objective functions
- Constraints
- Operating point sets
- Optimization parameters

The following methods are available:

Methods of cgoptimoptions

Task	Command
Add a model constraint to the optimization	addModelConstraint
Add a linear constraint to the optimization	addLinearConstraint
Add a free variable to the optimization	addFreeVariable
Add an objective to the optimization	addObjective
Add an operating point set to the optimization	addOperatingPointSet
Add a parameter to the optimization	addParameter
Get model constraint placeholder information	getModelConstraints
Get linear constraint placeholder information	getLinearConstraints
Return the current usage of constraints	getConstraintsMode
Get the current description for the optimization function	getDescription

Task	Command
Get the current enabled status for the optimization	<code>getEnabled</code>
Return the optimization free variable labels	<code>getFreeVariables</code>
Return the current usage of free variables	<code>getFreeVariablesMode</code>
Get the current name label for the optimization function	<code>getName</code>
Return information about the optimization objectives	<code>getObjectives</code>
Return the current usage of objective functions	<code>getObjectivesMode</code>
Return information about the optimization operating point sets	<code>getOperatingPointSets</code>
Return the current usage of operating point sets	<code>getOperatingPointsMode</code>
Return information about the optimization parameters	<code>getParameters</code>
Set how the optimization constraints are to be used	<code>setConstraintsMode</code>
Provide a description for the optimization function	<code>setDescription</code>
Set the enabled status for this optimization function	<code>setEnabled</code>
Set how the optimization free variables are used	<code>setFreeVariablesMode</code>
Provide a name label for an optimization function	<code>setName</code>

Task	Command
Set how the optimization objective functions are used	setObjectivesMode
Set how the optimization operating point sets are used	setOperatingPointsMode

Optimization Function Reference

Following are the reference pages for all the functions you can use in user-defined optimizations. They are divided into two sections. See the tables above for an overview of all the available functions:

- “Methods of cgoptimstore” on page 11-47
- “Methods of cgoptimoptions” on page 11-48

Alphabetical List of Functions

evaluate	11-53
gridEvaluate	11-54
pevEvaluate	11-57
gridPevEvaluate	11-58
get	11-59
getDataset	11-60
getInitFreeVal	11-61
getNumRowsInDataset	11-62
getOutputInfo	11-63
getParam	11-64
setFreeVariables	11-65
setOutputInfo	11-66
addModelConstraint	11-67
addLinearConstraint	11-68
addFreeVariable	11-69
addObjective	11-70
addOperatingPointSet	11-71
addParameter	11-72
getModelConstraints	11-73
getLinearConstraints	11-74
getConstraintsMode	11-75
getDescription	11-76
getEnabled	11-77
getFreeVariables	11-78
getFreeVariablesMode	11-79
getName	11-80
getObjectives	11-81
getObjectivesMode	11-82
getOperatingPointSets	11-83
getOperatingPointsMode	11-84
getParameters	11-85
setConstraintsMode	11-86
setDescription	11-87
setEnabled	11-88
setFreeVariablesMode	11-89

setName	11-90
setObjectivesMode	11-91
setOperatingPointsMode	11-92

Purpose Evaluate optimization objectives and constraints

Syntax `Y = evaluate(optimstore, X)`

Description Evaluates all the optimization objectives and constraints at the free variable values *X*. *X* must be a (NPoints-by-NFreeVar) matrix where NPoints is the number of points in the Primary data set and NFreeVar is the number of free variables in the optimization. The operating points used for evaluation are those in the Primary data set.

Examples The following command evaluates the objectives and constraints specified in the cell array of strings *itemnames*, at the free variable values *X*. The values of the objectives and constraints are returned in *Y*, which is of size (NPoints-by-NItems) where NItems is the number of objectives and constraints listed in *itemnames*.

```
Y = evaluate(optimstore, X, itemnames)
```

The following command evaluates the specified objectives and constraints at the operating points in the data set specified by the string *datasetname*.

```
Y = evaluate(optimstore, X, itemnames, datasetname)
```

The following command evaluates the specified objectives and constraints at the points of *datasetname* given by *rowind*. *X* must be a (NRows-by-NFreeVar) matrix where NRows is the length of *rowind*. *rowind* must be a list of integer indices in the range [1 NumRowsInDataset]. *Y* is a (Nrows-by-NItems) matrix.

```
Y = evaluate(optimstore, X, itemnames, datasetname, rowind)
```

See Also `gridEvaluate` on page 11-54, `pevEvaluate` on page 11-57

gridEvaluate

Purpose Grid evaluation of optimization objectives and constraints

Syntax `Y = gridEvaluate(optimstore, X)`

Description Evaluates all the objectives and constraints at all combinations of the points in the Primary (first) data set, P, with X. The return matrix, Y, is of size `size(X, 1)` by `(nobj+ncon)` by `npts`, where `nobj` is the number of objectives, `ncon` is the number of constraints, and `npts` is the number of rows in P. Further, `Y(I, J, K)` is the value of the `J`th objective/constraint at `X(I, :)` and `P(K, :)`.

Examples Objectives : 01, 02

Constraints : C1, C2

Primary data set:

A	B
4	5
1	3

Free variables:

X

X1	X2	X3
2	4	8
1	9	3
6	2	7

In this case the following command

```
Y = gridEvaluate(optimstore, X)
```

evaluates objectives and constraints at the following points:

A	B	X1	X2	X3
4	5	2	4	8
4	5	1	9	3
4	5	6	2	7
1	3	2	4	8
1	3	1	9	3
1	3	6	2	7

Y is a 3-by-4-by-2 matrix where

$Y(:, 1, 1)$ = Values of 01 at A = 4, B = 5

$Y(:, 2, 1)$ = Values of 02 at A = 4, B = 5

$Y(:, 3, 1)$ = Values of C1 at A = 4, B = 5

$Y(:, 4, 1)$ = Values of C2 at A = 4, B = 5

$Y(:, 1, 2)$ = Values of 01 at A = 1, B = 3

$Y(:, 2, 2)$ = Values of 02 at A = 1, B = 3

$Y(:, 3, 2)$ = Values of C1 at A = 1, B = 3

$Y(:, 4, 2)$ = Values of C2 at A = 1, B = 3

gridEvaluate

```
Y = gridEvaluate(optimstore, X, objconname)
```

evaluates the objectives/constraints specified in the cell array of strings, `objconname`, as described above. Note that the return matrix is of size `size(X, 1)` by `length(objconname)` by `npts`.

```
Y = gridEvaluate(optimstore, X, objconname, datasetname)
```

evaluates the objectives/constraints as described above. The evaluation is performed at the operating points in the data set specified by the string `datasetname`. Note that the return matrix is of size `size(X, 1)` by `length(objconname)` by `npts`, where `npts` is the number of points in the data set specified by `datasetname`.

```
Y = gridEvaluate(optimstore, X, objconname, datasetname, rowind)
```

evaluates the specified objectives/constraints at the points of `datasetname` given by `rowind` as described above. `Y` is a `length(rowind)` by `length(objconname)` by `npts` matrix.

See Also

`evaluate` on page 11-53

Purpose Evaluate prediction error variance (PEV)

Syntax `Y = pevEvaluate(optimstore, X)`

Description Evaluates PEV for optimization objectives/constraints at the free variable values *X*. *X* must be a *npts* by *nfreevar* matrix where *npts* is the number of points in the Primary data set and *nfreevar* is the number of free variables. Note that the operating points used for evaluation are those in the Primary data set. For any objectives/constraints that do not support PEV, NaN is returned.

Examples `Y = pevEvaluate(optimstore, X, objconname)`

evaluates PEV for objectives/constraints specified in the cell array of strings, *objconname*, at the free variable values *X*. The values of the objectives/constraints are returned in *Y*, which is of size *npts* by `length(objconname)`.

`Y = pevevaluate(optimstore, X, objconname, datasetname)`

evaluates PEV for the objectives/constraints at the operating points in the data set specified by the string *datasetname*.

`Y = pevevaluate(optimstore, X, objconname, datasetname, rowind)`

evaluates PEV for the specified objectives/constraints at the points of *datasetname* given by *rowind*. *X* must be a `length(rowind)` by *nfreevar* matrix. *Y* is a `length(rowind)` by `length(objconname)` matrix.

See Also `gridPevEvaluate` on page 11-58

gridPevEvaluate

Purpose Grid evaluation of prediction error variance (PEV)

Syntax `Y = gridPevEvaluate(optimstore, X)`

Description Evaluates PEV for the objectives and constraints at all combinations of the points in the Primary data set, P, with X. The return matrix, Y, is of size `size(X, 1)` by `(nobj+ncon)` by `npts`, where `nobj` is the number of objectives, `ncon` is the number of constraints, and `npts` is the number of rows in P. Further, `Y(I, J, K)` is the value of the Jth objective/constraint at `X(I, :)` and `P(K, :)`.

Examples `Y = gridPevEvaluate(optimstore, X, objconname)`

evaluates PEV for the objectives/constraints specified in the cell array of strings, `objconname`, as described above. Note that the return matrix is of size `size(X, 1)` by `length(objconname)` by `npts`.

`Y = gridPevEvaluate(optimstore, X, objconname, datasetname)`

evaluates PEV for the objectives/constraints as described above. The evaluation is performed at the operating points in the data set specified by the string `datasetname`. Note that the return matrix is of size `size(X, 1)` by `length(objconname)` by `npts`, where `npts` is the number of points in the data set specified by `datasetname`.

`Y = gridPevEvaluate(optimstore, X, objconname, datasetname, rowind)`

evaluates PEV for the specified objectives/constraints at the points of `datasetname` given by `rowind` as described above. Y is a `length(rowind)` by `length(objconname)` by `npts` matrix.

See Also `pevEvaluate` on page 11-57

Purpose Get optimization properties

Syntax `V = get(optimstore, 'PropertyName')`

Description Returns the value of the specified property in the optimization.

Examples `get(optimstore)`

displays all property names and a description of each property for the `optimstore` object.

`S = get(optimstore)`

returns a structure where each field name is the name of a property of `optimstore` and each field contains the description of that property.

getDataset

Purpose Get values from optimization operating point sets

Syntax `V = getDataset(optimstore, factornames)`

Description Returns data from the Primary operating point set. This function can only be used to return data columns that are required to be in the Primary operating point set. `factornames` is a cell array of labels specifying which columns of data are to be returned from the Primary data set. `factornames` must only include those strings used to label the required columns in the Primary data set. These labels must be created in the Options section of the user-defined script (these labels will have been set via the `addoperatingpointset` command; see “`addOperatingPointSet`” on page 11-71).

Examples `V = getDataset(optimstore, factornames, datasetname)`

returns data from the operating point set labeled `datasetname` in the optimization. `V` is a `npts` by `length(factornames)` matrix, where `npts` is the number of rows in the operating point set labeled `datasetname`.

`V = getDataset(optimstore, {'speed', 'afr'}, 'myDS')`

returns a `npts` by 2 matrix, `V`. `npts` is the number of rows in the operating point set labeled `myDS`, `V(:, 1)` is the data for the variable labeled `speed`, and `V(:, 2)` is the data for the variable labeled `afr`.

See Also `addOperatingPointSet` on page 11-71

Purpose Get the initial free values for the optimization

Syntax `X0 = getInitFreeVal(optimstore)`

Description Returns the initial values of the free variables used in the optimization. These values are set by the user in the **Free Variable Set Up** dialog when the optimization is run. X0 is a (NPoints-by-NFreeVar) matrix where NPoints is the number of rows in the Primary data set and NFreeVar is the number of free variables in the optimization.

See Also get on page 11-59, setFreeVariablesMode on page 11-89

getNumRowsInDataset

Purpose Get the number of rows in an optimization operating point set

Syntax `npts = getNumrowsInDataset(optimstore)`

Description Returns the number of rows in the Primary operating point set.

Examples `npts = getNumrowsInDataset(optimstore, datasetname)`
returns the number of rows in the operating point set labeled datasetname.

Purpose Get output information for the optimization

Syntax [ok, errmsg] = getOutputInfo (optimstore)

Description Returns diagnostic output information from optimstore. OK denotes the success (OK = 1) or failure (OK = 0) of the current optimization run. If the optimization is unsuccessful, an error message is returned in errmsg.

Examples [ok, errmsg, output] = getoutputinfo (optimstore)

returns in addition a structure of algorithm-specific information in output. For output to be nonempty, the user must create it in his algorithm. See the worked example and tutorial for more information on how to create output structures.

See Also setOutputInfo on page 11-66, “Worked Example Optimization” on page 6-38

getParam

Purpose Get optimization parameter

Syntax `V = getParam(optimstore, 'Parameter_name')`

Description Returns the value of the specified parameter in the optimization. These optimization parameters must be set up in the Options section of the user-defined script.

See Also `addParameter` on page 11-72, “Worked Example Optimization” on page 6-38

Purpose Set the optimal values of the free variables

Syntax `OUT = setFreeVariables(optimstore, results)`

Description Sets the optimal values of the free variables, as returned by the optimization, into the `optimstore`. `results` is a `npts` by `nfreevar` matrix containing the optimal values of the free variables. `npts` is the number of rows in the Primary operating point set and `nfreevar` is the number of free variables.

Note This function *must* be called at the end of the optimization for the optimal values to be stored.

setOutputInfo

Purpose Set output information for the optimization

Syntax `optimstore = setOutputInfo (optimstore, ok, errmsg, output)`

Description Sets output information for the optimization in `optimstore`. The following information is set:

- `ok`: (0/1), indicating whether the optimization has completed without error
- `errmsg`: Error message string if `ok = 0`, or an empty string if `ok = 1`
- `output`: Structure of algorithm statistics for the optimization

See the worked example “Worked Example Optimization” on page 6-38 for an example of creating an output structure.

See Also `getOutputInfo` on page 11-63

- Purpose** Add a model constraint to the optimization
- Syntax** `options=addModelConstraint(options, label, boundtype, bound)`
- Description** Adds a placeholder for a model constraint to the optimization. The string `label` is used to refer to the constraint in CAGE.
- `boundtype` can be set either to the string 'greaterthan' or 'lessthan'.
- `bound` must be a scalar real.
- If `boundtype = 'greaterthan'`, the model constraint takes the following form:
CAGE model \geq bound
- Similarly, if `boundtype = 'lessthan'`, the model constraint takes the form
CAGE model \leq bound
- Examples** An optimization requires a constraint where a user-defined function must be less than 500. The following code line adds a placeholder for this constraint that is labeled 'mycon':
- ```
opt = addModelConstraint(opt, 'mycon', 'lessthan', 500);
```
- See Also** `getModelConstraints` on page 11-73, `addLinearConstraint` on page 11-68, `setConstraintsMode` on page 11-86

# addLinearConstraint

---

**Purpose** Add a linear constraint to the optimization

**Syntax** `options = addLinearConstraint(options, label, A, B)`

**Description** Adds a placeholder for a linear constraint to the optimization. The string `label` is used to refer to the constraint in the CAGE GUI. Linear constraints can be written in the form

$$A(1)X(1) + A(2)X(2) + \dots + A(n)X(n) \leq b$$

where  $X(i)$  is the  $i^{\text{th}}$  free variable,  $A$  is a vector of coefficients, and  $b$  is a scalar bound.

**Examples**

```
% Add SPK and EGR variables to an optimization
opt = addFreeVariable(opt, 'SPK');
opt = addFreeVariable(opt, 'EGR');
% Add a linear constraint such that 3*SPK - 2*EGR <= 30
opt = addLinearConstraint(opt, 'newCon', [3 -2], 30);
```

**See Also** `getLinearConstraints` on page 11-74, `addModelConstraint` on page 11-67, `setConstraintsMode` on page 11-86



- Purpose** Add a free variable to the optimization
- Syntax** `options = addfreeVariable (options, label)`
- Description** Adds a placeholder for a free variable to the optimization. The string label is used to refer to the variable in CAGE.
- See Also** `setFreeVariablesMode` on page 11-89, `getFreeVariablesMode` on page 11-79, `getFreeVariables` on page 11-78

# addObjective

---

**Purpose** Add an objective to the optimization

**Syntax** `options = addObjective(options, label, typestr)`

**Description** Adds a placeholder for an objective function to the optimization. The string `label` is used to refer to the constraint in CAGE.

`typestr` can take one of four values, 'max', 'min', 'min/max', or 'helper'.

**Examples** `opt = addObjective(opt, 'newObj', 'max')`

Adds an objective function labeled `newObj` to the optimization and indicates that it is to be maximized.

`opt = addObjective(opt, 'newObj', 'min/max')`

Adds an objective function labeled `newObj` to the optimization and indicates that the user should be allowed to choose whether it is minimized or maximized from CAGE.

`opt = addObjective(opt, 'newObj2', 'helper')`

Adds an objective function labeled `newObj2` to the optimization. The string 'helper' indicates that the function is used as part of the determination of the cost function but is not directly minimized or maximized.

**See Also** `getObjectives` on page 11-81, `setObjectivesMode` on page 11-91, `getObjectivesMode` on page 11-82

**Purpose** Add an operating point set to the optimization

**Syntax** `options = addOperatingPointSet(options, label, vars)`

**Description** Adds a placeholder for an additional operating point set to the optimization. The string `label` is used to refer to the constraint in CAGE. `vars` is a (1-by-N) cell array of strings where  $N \geq 1$ . Each element of `vars` is a label for a CAGE variable that must appear in the operating point set that the user chooses.

**See Also** `getOperatingPointSets` on page 11-83, `setOperatingPointsMode` on page 11-92, `getOperatingPointsMode` on page 11-84

# addParameter

---

**Purpose** Add a parameter to the optimization

**Syntax** `options = addParameter(options, label, typestr, value)`

**Description** Adds a parameter to the optimization. The string `label` is used to refer to the parameter. The string `typestr` takes one of 'number', 'list', or 'boolean'. A default value for the parameter must be supplied in `value`. The form of `value` must be one of the following:

| <b>typestr</b> | <b>Value</b>                                    |
|----------------|-------------------------------------------------|
| 'number'       | Scalar, real number                             |
| 'list'         | Cell array of strings, one for each list member |
| 'boolean'      | True or false                                   |

**See Also** `getParameters` on page 11-85, `getParam` on page 11-64

|                    |                                                                                                                                                                                                                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Get model constraint placeholder information                                                                                                                                                                                                                                                |
| <b>Syntax</b>      | <code>out = getModelConstraints (options)</code>                                                                                                                                                                                                                                            |
| <b>Description</b> | Returns a structure array of information regarding the model constraints in the optimization. The structure has three fields: <code>label</code> , <code>boundtype</code> , and <code>bound</code> . See the help for <code>addModelConstraint</code> for more information on these fields. |
| <b>See Also</b>    | <code>addModelConstraint</code> on page 11-67, <code>setConstraintsMode</code> on page 11-86                                                                                                                                                                                                |

# getLinearConstraints

---

**Purpose** Get linear constraint placeholder information

**Syntax** `out = getLinearConstraints(options)`

**Description** Returns a structure array of information regarding the linear constraints in the optimization. The structure has three fields: `label`, `A`, and `b`. See the help for `addLinearConstraint` for more information on these fields.

**See Also** `addLinearConstraint` on page 11-68, `setConstraintsMode` on page 11-86

|                    |                                                                                                                                 |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Return the current usage of constraints                                                                                         |
| <b>Syntax</b>      | <code>mode = getConstraintsMode(options)</code>                                                                                 |
| <b>Description</b> | Returns a string describing how the optimization makes constraints available to the user. mode will be one of 'any' or 'fixed'. |
| <b>See Also</b>    | <code>setConstraintsMode</code> on page 11-86                                                                                   |

# getDescription

---

**Purpose** Get the current description for the optimization function

**Syntax** `desc = getDescription(options)`

**Description** Returns the description, desc, of the user-defined optimization function.

**See Also** `setDescription` on page 11-87



**Purpose** Get the current enabled status for the optimization

**Syntax** `en=getEnabled(options)`

**Description** Returns whether this user-defined optimization is available to be run. `en` is set to true or false. When an optimization is disabled, the user can still register it with CAGE but is not allowed to create new optimizations using it.

**See Also** `getEnabled` on page 11-77

# getFreeVariables

---

|                    |                                                                                                                                                                                                                                                                         |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Return the optimization free variable labels                                                                                                                                                                                                                            |
| <b>Syntax</b>      | <code>labels=getFreeVariables(options)</code>                                                                                                                                                                                                                           |
| <b>Description</b> | Returns the current placeholder labels for the free variables in the optimization. The labels are returned in a (1-by-NFreeVar) cell array, <code>labels</code> , where <code>NFreeVar</code> is the number of free variables that have been added to the optimization. |
| <b>See Also</b>    | <code>addFreeVariable</code> on page 11-69, <code>setFreeVariablesMode</code> on page 11-89, <code>getFreeVariablesMode</code> on page 11-79                                                                                                                            |

|                    |                                                                                                                           |
|--------------------|---------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Return the current usage of free variables                                                                                |
| <b>Syntax</b>      | <code>mode= getFreeVariablesMode(options)</code>                                                                          |
| <b>Description</b> | Returns a string describing how the optimization makes free variables available to the user. mode is set to any or fixed. |
| <b>See Also</b>    | <code>setFreeVariablesMode</code> on page 11-89                                                                           |

# getName

---

**Purpose** Get the current name label for the optimization function

**Syntax** name=getName(options)

**Description** Returns the current name label, name, for the user-defined optimization function.

**See Also** setName on page 11-90

|                    |                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Return information about the optimization objectives                                                                                                                                                                                                                                                                                                                         |
| <b>Syntax</b>      | <code>objinfo=getObjectives(options)</code>                                                                                                                                                                                                                                                                                                                                  |
| <b>Description</b> | Returns a structure array of information regarding the optimization objective functions. <code>objinfo(i).label</code> contains the label for the $i^{\text{th}}$ objective. A string defining the type of the $i^{\text{th}}$ objective ( <code>max</code> , <code>min</code> , <code>min/max</code> , or <code>helper</code> ) is stored in <code>objinfo(i).type</code> . |
| <b>See Also</b>    | <code>addObjective</code> on page 11-70, <code>setObjectivesMode</code> on page 11-91, <code>getObjectivesMode</code> on page 11-82                                                                                                                                                                                                                                          |

# getObjectivesMode

---

**Purpose** Return the current usage of objective functions

**Syntax** `mode = getObjectivesMode(options)`

**Description** Returns a string describing how the optimization makes objectives available to the user. `mode` will be one of 'multiple', 'any', or 'fixed'.

**See Also** `setObjectivesMode` on page 11-91

|                    |                                                                                                                                                                                                                                                             |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Return information about the optimization operating point sets                                                                                                                                                                                              |
| <b>Syntax</b>      | <code>getOperatingPointSets(options)</code>                                                                                                                                                                                                                 |
| <b>Description</b> | Returns a structure array of information regarding the optimization operating point sets. The structure has two fields, <code>label</code> and <code>vars</code> . See the help for <code>addOperatingPointSet</code> for more information on these fields. |
| <b>See Also</b>    | <code>addOperatingPointSet</code> on page 11-71, <code>setOperatingPointsMode</code> on page 11-92, <code>getOperatingPointsMode</code> on page 11-84                                                                                                       |

# getOperatingPointsMode

---

**Purpose** Return the current usage of operating point sets

**Syntax** `mode=getOperatingPointsMode(options)`

**Description** Returns a string describing how the optimization makes operating point sets available to the user. mode will be one of 'default', 'fixed', or 'any'.

**See Also** `setOperatingPointsMode` on page 11-92



|                    |                                                                                                                                                                                                                                                                                                                                 |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Return information about the optimization parameters                                                                                                                                                                                                                                                                            |
| <b>Syntax</b>      | <code>getParameters(options)</code>                                                                                                                                                                                                                                                                                             |
| <b>Description</b> | Returns a structure array containing information about the parameters that are defined for the optimization. Parameter information is returned in a structure with fields <code>label</code> , <code>typestr</code> , and <code>value</code> . See the help for <code>addParameter</code> for more information on these fields. |
| <b>See Also</b>    | <code>addParameter</code> on page 11-72                                                                                                                                                                                                                                                                                         |

# setConstraintsMode

---

**Purpose** Set how the optimization constraints are to be used

**Syntax** `options=setConstraintsMode(options, modestr)`

**Description** Sets the mode that governs how the user can set up constraints for the optimization in CAGE.

When `modestr = any`, the user can add any number of constraints.

When `modestr = fixed`, the user can only edit the constraints that are added by the user-defined optimization function.

**See Also** `getConstraintsMode` on page 11-75, `addModelConstraint` on page 11-67, `addLinearConstraint` on page 11-68

- Purpose** Provide a description for the optimization function
- Syntax** `options=setDescription(options, desc)`
- Description** Sets the description for the optimization object to be the string desc.
- See Also** `getDescription` on page 11-76

# setEnabled

---

**Purpose** Set the enabled status for this optimization function

**Syntax** `options = setEnabled(options, status)`

**Description** Sets the optimization function enabled status. `status` must be `true` or `false`. When an optimization is disabled, you can still register it with CAGE but are not allowed to create new optimizations using it.

**See Also** `setEnabled` on page 11-77

|                    |                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Set how the optimization free variables are used                                                                                                                                                                                                                                                                                                                                               |
| <b>Syntax</b>      | <code>options = setFreeVariablesMode(options, modestr)</code>                                                                                                                                                                                                                                                                                                                                  |
| <b>Description</b> | <p>Sets the mode that governs how the user is allowed to set up free variables for the optimization in the CAGE GUI.</p> <p>When <code>modestr = 'any'</code>, the user is allowed to add any number of free variables.</p> <p>When <code>modestr = 'fixed'</code>, the user is only allowed to use the number of free variables that are added by the user-defined optimization function.</p> |
| <b>See Also</b>    | <code>getFreeVariablesMode</code> on page 11-79, <code>addFreeVariable</code> on page 11-69                                                                                                                                                                                                                                                                                                    |

# setName

---

**Purpose** Provide a name label for an optimization function

**Syntax** `options = setName(options, name)`

**Description** Sets the name label for the optimization object to be the string name.

**See Also** `getName` on page 11-80

**Purpose** Set how the optimization objective functions are used

**Syntax** `options = setObjectivesMode(options, modestr)`

**Description** Sets the mode that governs whether the user is allowed to set up objectives for the optimization in the CAGE GUI. When `modestr = 'any'`, the user is allowed to add any number of objectives. When `modestr = 'fixed'`, the user is only allowed to edit the objectives that are added by the user-defined optimization function. When `modestr = 'multiple'`, the user is only allowed to run the optimization if he or she has defined two or more objectives.

**See Also** `getObjectivesMode` on page 11-82, `addObjective` on page 11-70

# setOperatingPointsMode

---

**Purpose** Set how the optimization operating point sets are used

**Syntax** `options = setObjectivesMode(options, modestr)`

**Description** Sets the mode that governs how the user is allowed to set up operating point sets for the optimization in CAGE.

When `modestr = 'any'`, the user is allowed to add any number of operating point sets.

When `modestr = 'default'`, the user is allowed to optionally define a single operating point set to run the optimization over.

When `modestr = 'fixed'`, the number of operating point sets required can be fixed by the optimization function and the user is not allowed to add or remove any using the CAGE GUI.

**See Also** `getOperatingPointsMode` on page 11-84, `addOperatingPointSet` on page 11-71



# Data Sets

---

This section includes the following topics:

|                                                  |                                                                                                                                                                        |
|--------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Data Sets Views (p. 12-2)                        | How to use the <b>Data Sets</b> views.                                                                                                                                 |
| Setting Up Data Sets (p. 12-4)                   | How to set up data sets by importing experimental data, importing data from tables, specifying factors manually, and creating a factor from the error between factors. |
| Viewing Data in a Table (p. 12-11)               | How to use the table view.                                                                                                                                             |
| Plotting Outputs (p. 12-13)                      | How to use the plot view.                                                                                                                                              |
| Using Color to Display Information (p. 12-16)    | How to use color plots and restrict the color to display factor information.                                                                                           |
| Linking Factors in a Data Set (p. 12-20)         | How to link factors.                                                                                                                                                   |
| Assigning Columns of Data (p. 12-22)             | How to assign columns of data to input factors, for example, in order to compare experimental data with tables or models.                                              |
| Manipulating Models in Data Set View (p. 12-23)  | How to change models from input to output factors.                                                                                                                     |
| Filling Tables from Experimental Data (p. 12-24) | How to fill tables from data, including creating rules.                                                                                                                |

## Data Sets Views



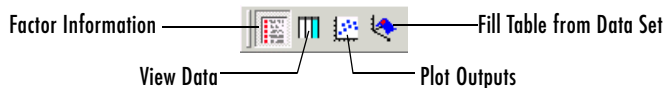
The **Data Set** view has two main functions:

- Validating calibrations with experimental data
- Filling tables by reference to a set of experimental data

For worked examples about data sets, see

- “Tutorial: Data Sets” on page 4-1  
This shows the process of validating a calibration.
- “Tutorial: Filling Tables from Data” on page 5-1  
This tutorial shows the process of filling a table from experimental data.

**Data Sets** consists of four views. These views display different aspects of the data set. Each view is accessible from the **View** menu or by clicking the appropriate button on the toolbar.



- **Factor Information**  
List of all available project expressions, which can be added to the data set for display and evaluation.
- **View Data**  
Displays the data in a table. Individual entries can be altered. Columns of data can be assigned to CAGE expressions.
- **Plot Outputs**  
Displays models and features evaluated at the data points (of the data set).
- **Fill Table from Data Set**  
This mode allows you to fill tables by reference to experimental data.

CAGE Browser - datasettut.cag

File Edit View Data Tools Window Help

Processes

Feature

Manual Calibration

Trade Off

Data Objects

Variable

Model

Data Set

Data Set

New\_Dataset


Data Set Factors

| Factor           | Status          | Units | Info |
|------------------|-----------------|-------|------|
| x n              | Input           |       |      |
| x load           | Input           |       |      |
| x afr            | Input           |       |      |
| x spk            | Input           |       |      |
| nmeas            | Output: Data    |       |      |
| mqmeas           | Output: Data    |       |      |
| Torque: Model    | Output: Feature |       |      |
| Torque: Strategy | Output: Feature |       |      |

Project Expressions

| Expression       | Unit | Branch   | Type     | Info        |
|------------------|------|----------|----------|-------------|
| x afr            |      |          | Variable | In data set |
| x load           |      |          | Variable | In data set |
| x n              |      |          | Variable | In data set |
| x spk            |      |          | Variable | In data set |
| T1               |      | Branch 1 | Table    |             |
| T2               |      | Branch 1 | Table    |             |
| T3               |      | Branch 1 | Table    |             |
| TORQUE           |      |          | Model    |             |
| Torque: Model    |      | Branch 1 | Feature  | In data set |
| Torque: Strategy |      | Branch 1 | Feature  | In data set |

## Setting Up Data Sets

The **Data Sets** view displays the strategies, tables, and models, etc., by default as a list of factors in the **Factor Information** view. You can also display the same factors as columns in a grid, with all factors displayed as columns in the list, by selecting the View Data toolbar button (  ). The data set works over a grid of values, which is not necessarily the same as the normalizers of any included tables in the data set.

You have to set the input factors and their values to define the grid in the data set. You can do this in one of three ways:

- Import experimental data. (See “Importing Experimental Data” on page 12-4.)
- Import the values from a table in your CAGE session. (See “Importing Data from a Table in Your Session” on page 12-6.)
- Specify the factors and their values manually. (See “Specifying the Factors Manually” on page 12-7.)

The next sections describe each of these in detail.

### Importing Experimental Data

You can import experimental data to a data set, either to validate a calibration or to use it as the basis for a calibration.

You can import data that is stored in the following formats:

- Microsoft Excel spreadsheets
- Comma-separated value files
- MAT-files

**Importing from Excel or Comma-Separated Value.** When you import data from either a Microsoft Excel spreadsheet or from a comma-separated value file, you must ensure that the data is organized in the following manner:

- The first column can either be row markers (text) or entries (numbers).
- The first row can either be column headers (text) or entries (numbers).
- All the other row and column entries must be numbers.

**Importing from MAT-files.** When you import from a MAT-file, you must ensure that the file contains numbers only, that is, a double array.

To import experimental data,

- 1 Select **File -> Import -> Data**.
- 2 In the file browser, select the correct file to import.

This opens the **Data Set Import Wizard**.

- 3 Discard any columns of data you do not want to import by selecting the column and clicking the button shown.



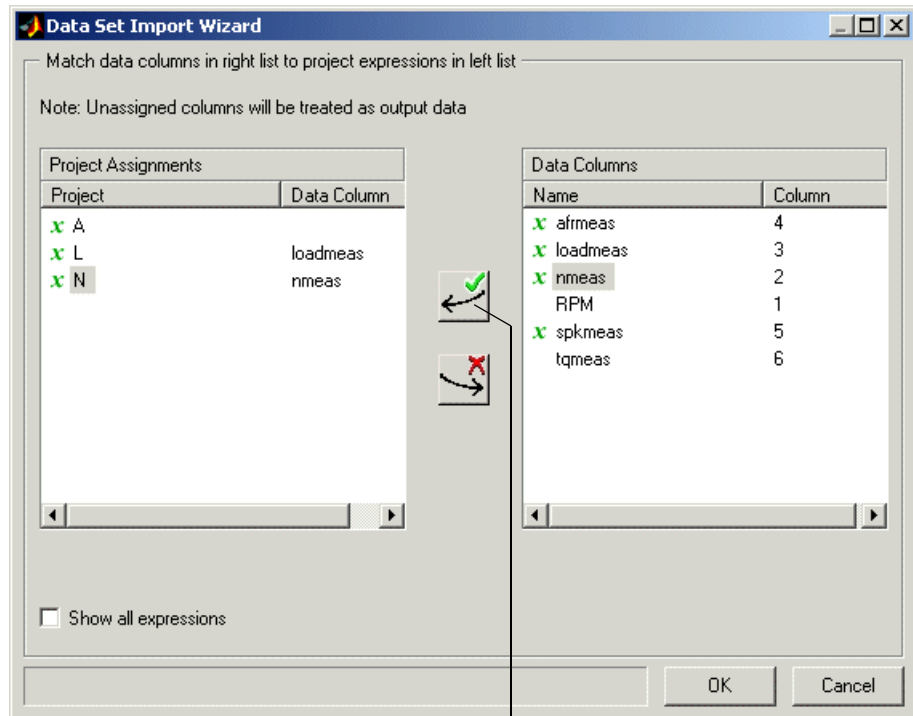
- 4 Click **Next**.

The following screen asks you to associate variables in your project with data columns in the data.

- 5 Highlight the variable in the **Project Assignments** column and the corresponding data column in the **Data Column**, then click the assign button, shown.



- 6 Repeat step 5 until you are satisfied that you have associated all the variables and data columns. Any unassigned data columns are treated as output factors.



Assign button

7 Click **Finish** to close the dialog box.

This imports your data into the data set. When you have imported your data, you can view your data set.

## Importing Data from a Table in Your Session

To import data from a table,

1 Select **Data** -> **Import** -> **Import from Table**.

If your data set is not empty, a dialog box asks whether you want to **Fill** the data set from the table or **Overwrite** the data set from the table. Select **Fill** to use the table values to fill the factors in your data set. Select **Overwrite**

to disregard all factors in your data set and fill the data set with the input and output factors from the table. A dialog box opens.

- 2 Select the correct table from your session to import and click **OK**.

When you have imported your data, you are ready to view the data set.

## Specifying the Factors Manually


- 1 Select the **Data Set** view by clicking the large **Data Sets** button in the **Data Objects** pane.
- 2 Add a data set to the project by selecting **File -> New -> Data Set**.
- 3 Select the factors. (See “Selecting the Factors” on page 12-7.)
- 4 Build the grid. (See “Manually Setting Values of the Input Variables” on page 12-9.)

Once you have completed these steps you can view the data set.

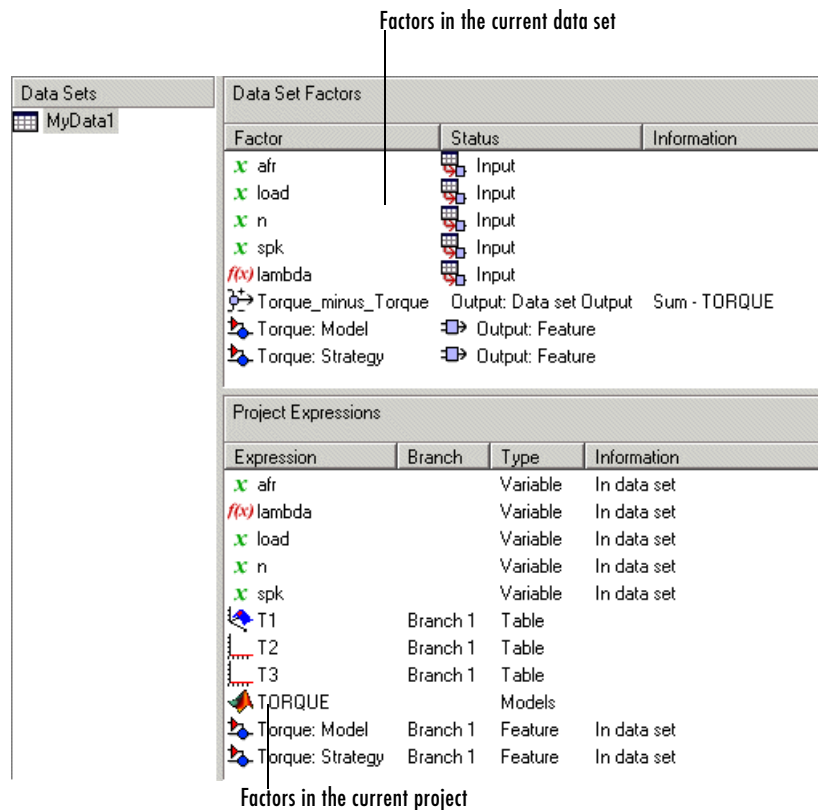
This section describes

- “Selecting the Factors” on page 12-7
- “Manually Setting Values of the Input Variables” on page 12-9

### Selecting the Factors

Clicking the Factors View button in the toolbar (  ). This displays two list boxes.

- The upper list shows all factors within the data set. You can sort factors by clicking the column headings.
- The lower list shows CAGE project expressions.



You can use this view to add factors to or remove factors from the data set.

To add a factor to a data set,

- 1 Select the factor or factors that you want to add to the data set from the list in the lower **Project Expressions** pane.

To make multiple selections, use the standard **Shift+click** or **Ctrl+click**.

- 2 Select **Data -> Factors -> Add to Data Set**. Alternatively, you can right-click the factor and select **Add to Data Set** from the context menu.



To remove a factor from a data set,


- 1 Select the factor or factors that you want to remove from the data set.
- 2 Select **Remove from Data Set** from the right-click menu.

---

**Note** Links between the two lists are always preserved, so clicking load in the upper list also selects load in the lower list. In other words, you can copy or remove from either list and the relevant results appear in both. Multiply selecting in one list therefore deselects everything in the other list.

---

### Manually Setting Values of the Input Variables

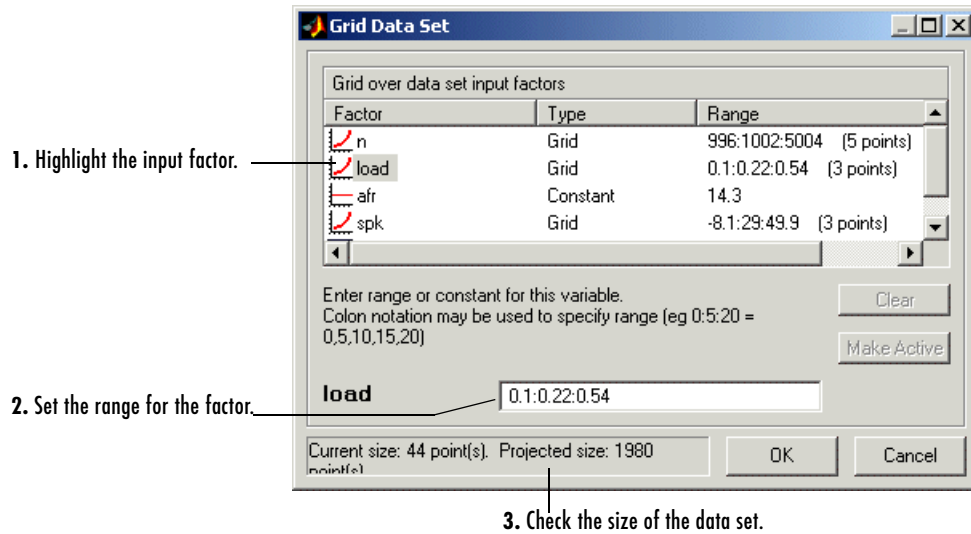
Clicking the Build Grid toolbar button (  ) or selecting **Data -> Build Grid** enables you to set the values of the input variables for the data set.

To build a full factorial grid,

- 1 Select **Data -> Build Grid**.
- 2 Select the factor that you want to define a grid for.
- 3 Set the grid for the factor.

To set a grid of 5, 10, 15, 20, 25, 30, input the following: 5:5:30, where the first number is the minimum, the second is the step size, and the last number is the maximum value.

- 4 Check the size of the data set in the pane. The current size reported at the bottom of the dialog is the size if you click **Cancel** to leave the data set unchanged. The projected size is created if you click **OK**. In the following example, the projected size of 45 you can see is obtained by multiplying the number of points for each factor with a grid (in this case,  $3 * 5 * 3$ ).
- 5 Select the next factor that you want to define a grid for.
- 6 When you have set the grids for all the factors, click **OK**.




## Creating a Factor from the Error Between Factors


To create a factor that is the difference between two other factors,

- 1 Highlight the two factors, using **Ctrl+click** or **Shift+click**.
- 2 Select **Create Error** from the right-click menu on either column head.

This creates a new factor that is the difference between the two other factors.

## Viewing Data in a Table

Click the View Data button (  ) in the toolbar or select **View -> Data** to display the data in tabular form and a list of the current items in the project.

Note that this view is only enabled if you have a grid of points at which to evaluate and display the models and variables. This grid is not necessarily derived from the normalizers of any tables included in the data set. You can set the grid by importing experimental or table data, or by using the Build Grid toolbar button (  ). See “Setting Up Data Sets” on page 12-4.

|    | N        | L     | A      | SPK   | EGR   | nmeas    | tgmeas | New_Feature: Model | New_Feature: Strategy |
|----|----------|-------|--------|-------|-------|----------|--------|--------------------|-----------------------|
| 1  | 2235.100 | 0.549 | 9.545  | 0.098 | 0.000 | 2247.200 | 66.662 | 36.326             | 35.477                |
| 2  | 3591.000 | 0.454 | 13.182 | 0.069 | 0.000 | 3613.200 | 54.114 | 26.381             | 24.362                |
| 3  | 4946.100 | 0.651 | 12.046 | 0.083 | 0.000 | 4973.500 | 73.700 | 34.406             | 42.730                |
| 4  | 880.800  | 0.648 | 11.934 | 5.719 | 0.000 | 880.570  | 75.769 | 58.109             | 50.743                |
| 5  | 2234.300 | 0.441 | 13.278 | 0.079 | 0.000 | 2246.900 | 55.882 | 30.038             | 24.112                |
| 6  | 3590.700 | 0.747 | 10.888 | 0.064 | 0.000 | 3611.800 | 89.983 | 45.420             | 56.687                |
| 7  | 4946.500 | 0.541 | 9.715  | 0.119 | 0.000 | 4972.700 | 62.753 | 20.032             | 29.869                |
| 8  | 880.800  | 0.622 | 9.904  | 0.057 | 0.000 | 884.230  | 72.084 | 47.888             | 41.787                |
| 9  | 1218.700 | 0.333 | 14.017 | 0.100 | 0.000 | 1224.400 | 41.775 | 17.776             | 11.358                |
| 10 | 1558.000 | 0.382 | 11.955 | 0.069 | 0.000 | 1567.200 | 49.384 | 25.404             | 17.214                |
| 11 | 1895.600 | 0.209 | 10.680 | 3.288 | 0.000 | 1905.700 | 28.509 | 5.919              | 0.296                 |
| 12 | 2233.800 | 0.284 | 9.805  | 3.159 | 0.000 | 2245.300 | 36.020 | 12.344             | 8.835                 |
| 13 | 2574.400 | 0.407 | 13.423 | 3.024 | 0.000 | 2588.000 | 49.883 | 26.942             | 23.297                |
| 14 | 2913.900 | 0.595 | 11.506 | 3.090 | 0.000 | 2929.400 | 70.537 | 46.414             | 43.871                |
| 15 | 3258.000 | 0.784 | 12.227 | 3.077 | 0.000 | 3267.600 | 89.545 | 58.160             | 54.820                |

Columns are color coded by factor type:

- Input factors are colored white.
- Output factors are colored light blue.

Selecting an output column highlights the input columns associated with it by turning the header cells yellow.

Standard editing facilities are available.

Double-click an input cell to edit the value.

Cut and paste using the desktop clipboard. Cells, columns, and rows can be copied directly to and from other applications (for example, Excel).

---


**Note** You can only edit input values, not output values.

---

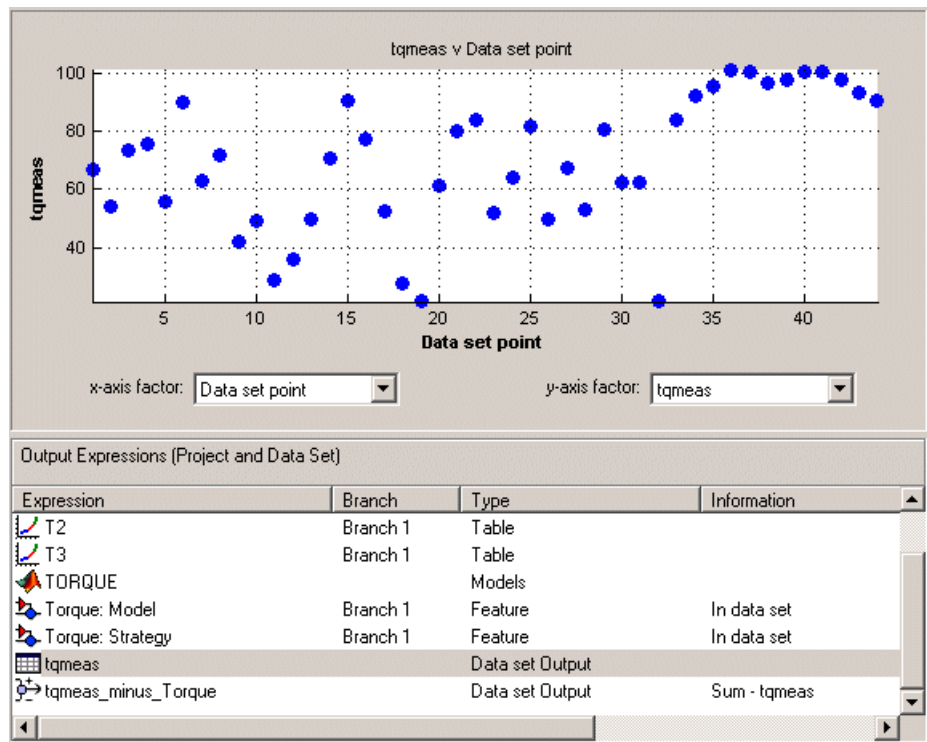
## Plotting Outputs

Use this to plot the outputs of your data sets.

To view a plot,

- 1 Select **View** -> **Plot** or click the  toolbar button.
- 2 Select an expression from the list to view.

A plot of the selected output factor appears in the top pane.



- 3 Use the pop-up menus below the plot to change the factors displayed.

To zoom in on an area of interest,

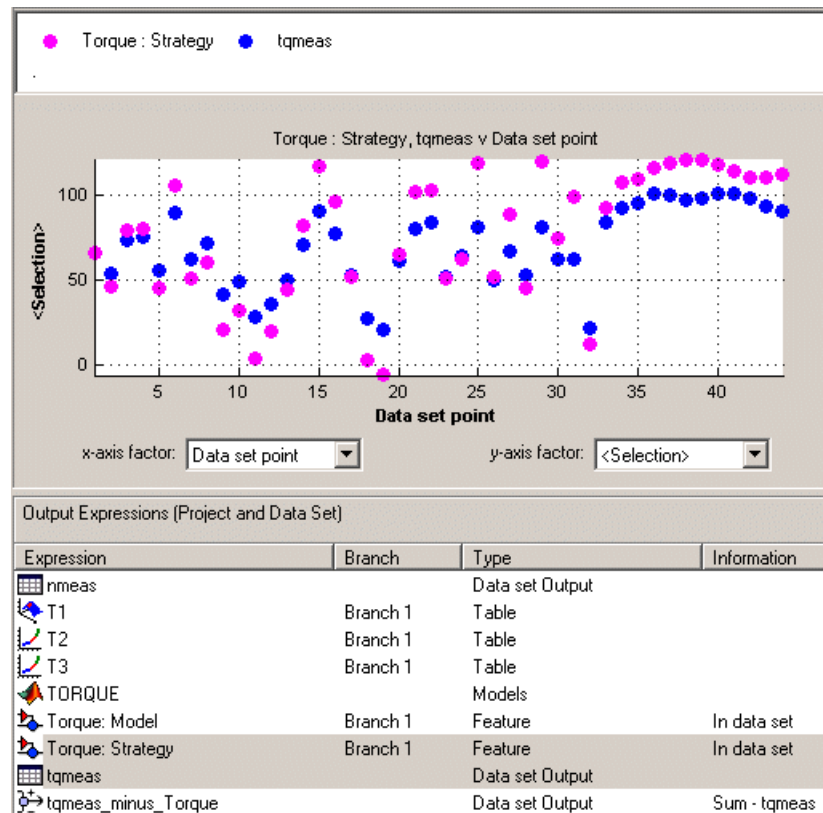
- Press both mouse buttons simultaneously and drag a rectangle; double-click the graph to return to full size.

## Plotting Multiple Selections

You can plot a multiple selection by using standard **Ctrl+click** and **Shift+click** operations.

A legend at the top of the screen displays the key to the graph.

## Multiple Plot Outputs




When exactly two items are displayed, further plot options are available:

- Plot the first item against the second item (**X-Y Selection**).
- Display the error using one of the following options:
  - Error
  - Absolute error
  - Relative error (%)
  - Absolute relative error (%)

## Using Color to Display Information

You can use the plot view to display more information by coloring the plots.

### Coloring a Plot

- 1 Select **View** -> **Plot** or click .
- 2 Highlight the correct expression in the **Output Expressions (Project and Data Set)** pane.
- 3 Select **Color by Value** from the right-click menu of the plot.
- 4 Select from the pop-up menu the variable you want to use to color the plot.



1. Click Plot Outputs.

3. Select Color by Value from the right-click menu.

4. Select the correct variable.

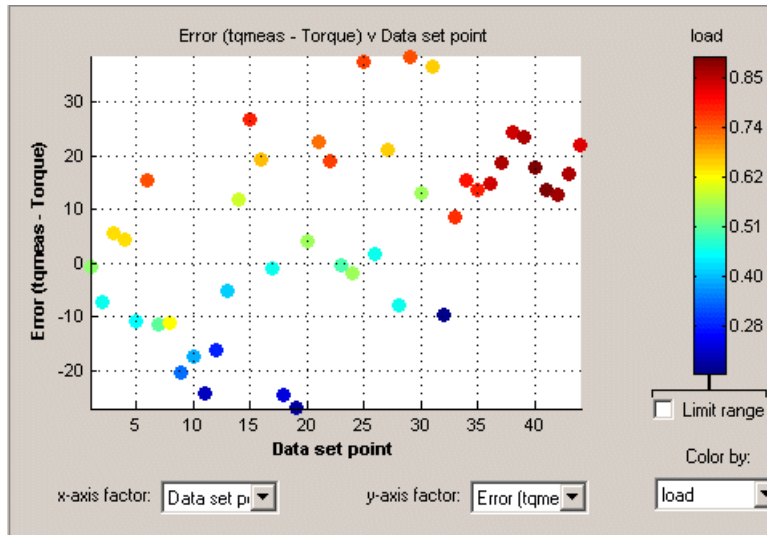
2. Select the expression.

| Expression       | Unit | Branch   | Type            | Info        |
|------------------|------|----------|-----------------|-------------|
| RPM              |      |          | Data set Output |             |
| T1               |      | Branch 1 | Table           |             |
| T2               |      | Branch 1 | Table           |             |
| T3               |      | Branch 1 | Table           |             |
| TORQUE           |      |          | Model           |             |
| Torque: Model    |      | Branch 1 | Feature         | In data set |
| Torque: Strategy |      | Branch 1 | Feature         | In data set |
| torqueas         |      |          | Data set Output |             |

In the following figure, you can see

- A plot of the Sum vs Data Set Point (this is the strategy from a torque feature calibration).

- The points are colored by load.
- For this example it can be seen that, in general, the higher the load, the higher the value of torque.



## Restricting the Color

You might be interested in only part of the display; for example, you might only be interested in points with a low engine speed. The various display options enable you to color only the points that you are interested in.

To restrict the color,

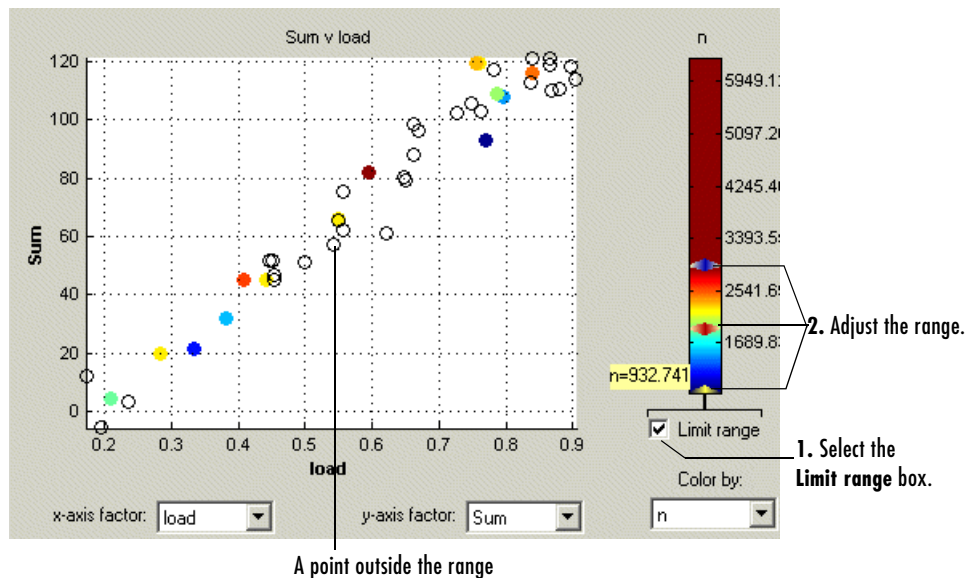
- 1 Select the **Limit range** box.
- 2 Adjust the maximum, midpoint, and minimum of the range by dragging the icons on the color bar.
- 3 Examine the data points and those that are outside the range.

Use the right-click menu to alter the view of the points outside the range:

- Select **Restrict Color to Limits** to compress the colors in the color bar within the limit markers, as in the example shown. Points outside the limits are still

colored, but only dark red or dark blue, depending on which end of the range they are. This increases the range of colors over the range you are interested in (between the limits), making it easier to see the distribution of points.

- Select **Exclude** to remove all points outside the limits from the display.
- Select **No Color Outside Limits** to display the points as in the example shown. Points outside the limits are plotted as empty circles.
- Select **Color Outside Limits** to display all points in color, including those outside the limits.



## Linking Factors in a Data Set

A factor can be linked to another. The factor then takes on the values of that other factor, overwriting the original values.

For example, you might want to link a variable spark with a model for maximum brake torque (MBT) to evaluate a torque model.


To link two factors,

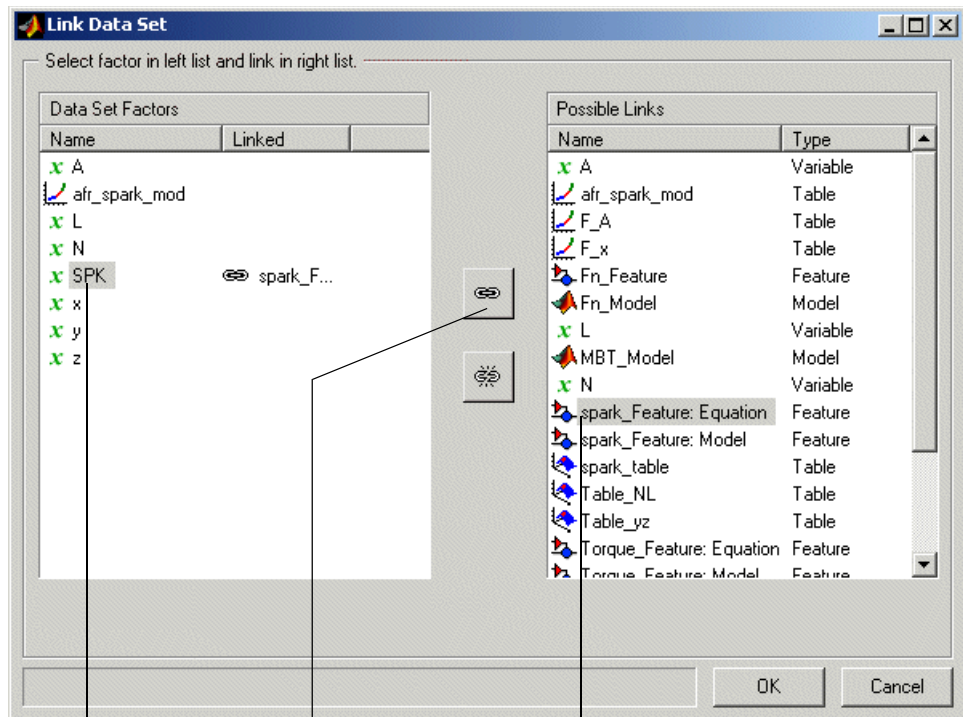
**1** Select **Data** -> **Links**. This opens a dialog box.

**2** Select the data set factor that you want to overwrite.

CAGE generates a list of factors that you could possibly link to the selected factor. (For example, you cannot link to a factor that depends on the selected factor.)

**3** Select the factor that you want to link the selected factor with.

**4** Click  to link the two factors.




2. Select the factor that you want to overwrite.

4. Click here to link the factors.

3. Select the factor that you want to link it with.

CAGE then overwrites the data set factor with the link.

To break a link and return to normal evaluation, click .

Once all the links have been created or broken as you want, click **OK** to exit the dialog.

### See Also


- “Setting Up Data Sets” on page 12-4

## Assigning Columns of Data


To analyze imported data, you need to assign columns of data to input factors in the CAGE data set.

Data can be imported into a data set from outside CAGE, for example, from an engine test cell. In many cases, this data contains a set of input points (or operating points) and the values of important measurable variables at those points. To compare data like this with models (and/or tables) in a CAGE data set, you have to assign columns of the data to the corresponding input factors in the data set.

To assign data,

- 1 Select **Data -> Assign**.
- 2 In the dialog box, highlight the column that you want to assign and the variable that you want to assign it to.
- 3 Click  to assign.

To unassign data,

- 1 Select **Data -> Assign**.
- 2 In the dialog box, highlight the variable that you want to unassign.
- 3 Click  to unassign.

---

**Note** Assigning data to a CAGE expression overwrites that expression in the data set. This does not affect the expression in the other parts of the CAGE project.

---

## Manipulating Models in Data Set View

A model in a data set can be treated as either an input or an output. This is particularly useful when a model is used as an input to another model and you want to view specific values of the input model. For example, linking a model of MBT Spark to a Spark model allows the evaluation of a TQ model at MBT.

To change a model to an input,

- 1 Highlight the desired model in either the factor view or the table view.
- 2 Select **Treat as Input** from the right-click menu.

To revert a model to an output,


- 1 Highlight the desired model in either the factor view or the table view.
- 2 Select **Treat as Output** from the right-click menu.

## Filling Tables from Experimental Data

Any table in the project whose axes (normalizers) exist as factors in the data set can be filled from imported experimental data.

CAGE extrapolates the values of the experimental data over the range of your table. Then it fills the table by selecting the values of the extrapolation at your breakpoints.

To fill the table with values based on the experimental data,

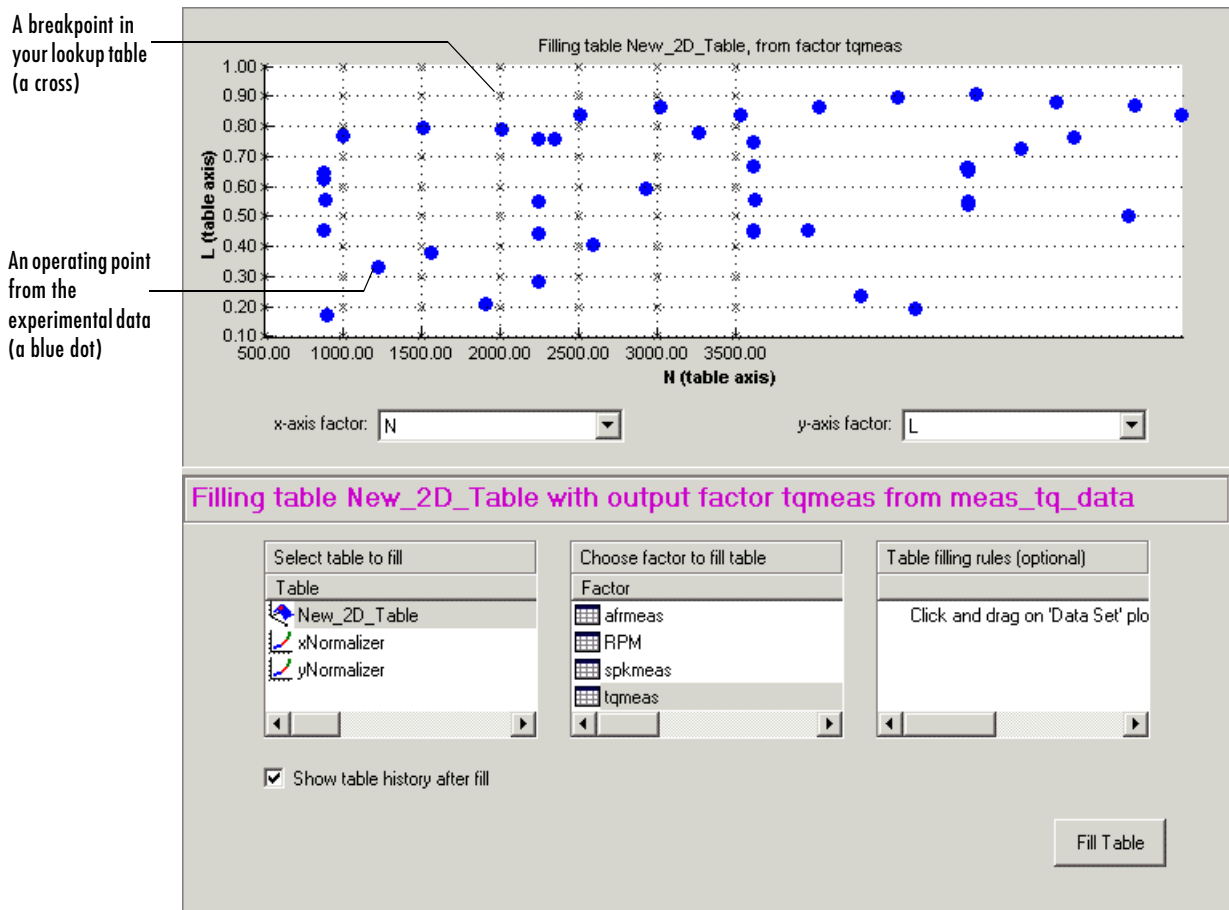
- 1 To view the **Table Filler** display, click  in the toolbar.

This display asks you to specify the table you want to fill and the factor you want to use to fill it.

- 2 In the lower pane, select the table from the **Table** list. This is the table that you want to fill.
- 3 Select the experimental data from the **Factor** list. This is the data that you want to use to fill the table.

For example, see the following display.





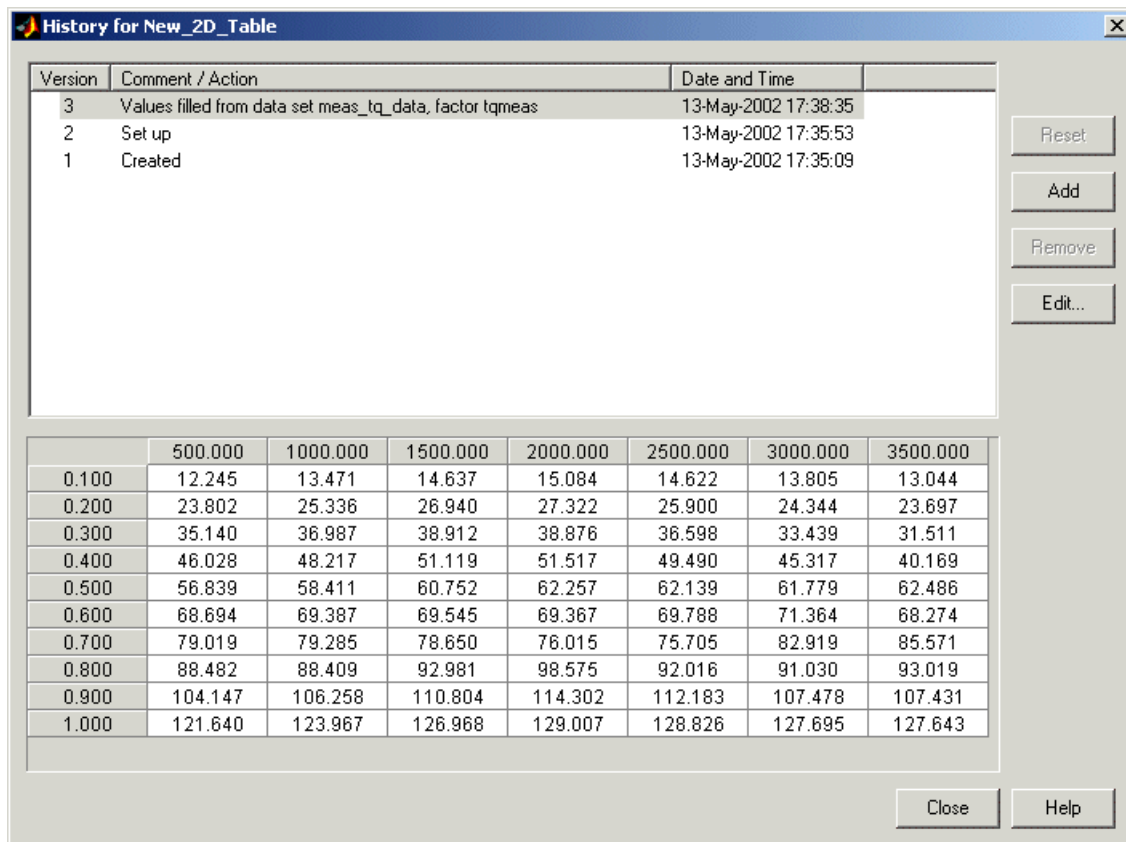
The upper pane displays the breakpoints of your table as crosses, and the operating points where there is data as blue dots.

- To view the table after it is filled, make sure the **Show table history after fill** box at the bottom left is selected. This is selected by default.

Data sets always display the points in the experimental data, not the values at the breakpoints.

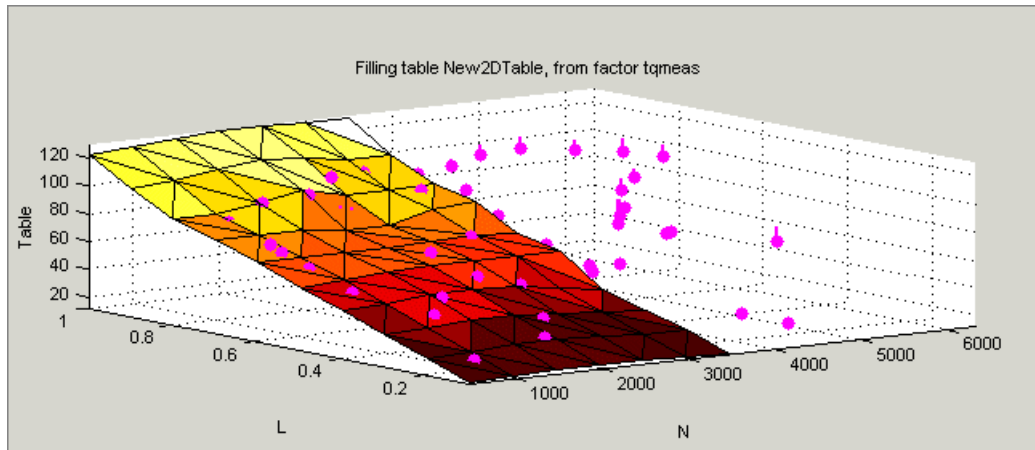
5 To fill the table, click **Fill Table**.

If the **Show table history after fill** box is selected, the **History** dialog box opens, similar to the one shown.



6 Click **Close** to close the **History** dialog box and return you to the **Table Filler** display.

7 To view the graph of your table, select **Data -> Plot -> Surface**.



This display shows the table filled with the experimental points overlaid as purple dots.

## Creating Rules

You can ignore points in the data set when you fill your lookup table.

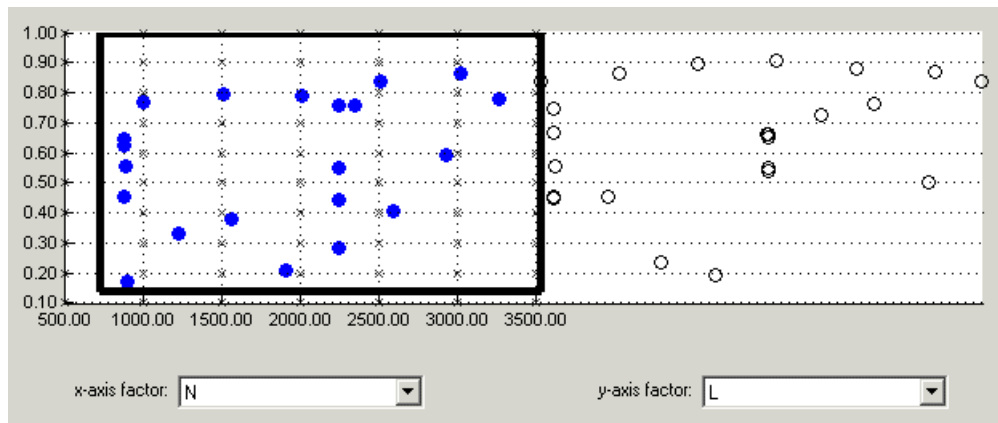
By defining a region to include or exclude such points, you create a rule for the table filling.

For example, you might want to fill a lookup table that has a range of operating points that is smaller than the range of the experimental data.

To ignore points in the data set,

- 1 Select **Data** → **Plot** → **Data Set**. This displays 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. For example, see the following display.

This region defines a rule in the **Table filling rules** pane.



**3** To fill the table based on an extrapolation over these data points only, click **Fill Table**.

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 can now review your data set using the options in the **View** and **Plot** panes of **Data Sets**.

You can add any number of rules to follow when filling tables. For example, you might be aware that a particular test run included in the chosen area is not good data. You can click and drag to enclose any chosen point, then right-click that rule (in the **Table filling rules** pane) and select **Exclude Points**. You can set any number of rules to make sure you fill the table by using just the points you are interested in.

### Right-Click Options

Right-clicking the **Table filling rules** pane gives you the following options:

- **Enable Rule:** Apply the rule to the data.
- **Disable Rule:** Do not apply the rule, but also do not delete it.
- **Exclude Points:** Do not include these points in table filling.
- **Include Points:** Include points in table filling.
- **Promote Rule:** Change order of rules.
- **Demote Rule:** Change order of rules.

- **Clear Rule:** Delete this rule.

You can use these options to enable an iterative process. You can fine-tune the selection of data points: try different selections of data to fill your tables, check the results, then reuse the same rules for the same or different tables.



# Calibration Manager

---

This section includes the following topics:


Setting Up Tables (p. 13-2)

The **Calibration Manager** dialog box enables you to manage the sizes, values, and precision of all items that can be calibrated. You can set these properties manually or from a calibration file. This section describes how to use the Calibration Manager to set up tables and copy table data from other sources.

Table Properties (p. 13-6)

How to change table properties to specific precision (floating-point, polynomial ratio fixed point, or lookup table fixed point) to suit your ECU.

## Setting Up Tables

To set up tables in CAGE, you first open the **Calibration Manager** dialog box. Do this by selecting **Tools** → **Calibration Manager** or by clicking  in the toolbar.

You can either set up your tables manually or from a calibration file. You can also copy table data from other sources.

### Setting Up Tables Manually

- 1 Select the normalizer or table to set up from the list on the left.
- 2 Enter the number of rows and columns in the edit boxes in the **Manual Setup** pane and select initial values for each cell in the table.



Rows  Columns  Value

Precision: IEEE Double Precision

Buttons: Set Up, Edit Precision...

---

**Note** When initializing tables for a feature calibration (comparing a model to a strategy) you should think about your strategy. CAGE cannot fill those tables if you try to divide by zero. Modifier tables should be initialized with a value of 1 for all cells if they are multipliers, and a value of 0 if they are to be added to other tables. See “How CAGE Fills Tables” on page 9-13 and “Initializing Table Values” on page 9-12.

---

- 3 Check the display of your table, then click **Close**.

### Setting Up Tables Using an Existing Calibration File

- 1 Open the file by clicking .

This opens the **Import Calibration** dialog box.

- 2 Select the type of file you want to open (M- or MAT-file) or select **Automatic**. Click **OK** to open the file browser.



- 3 Browse to the calibration file, select it, and click **Open**.

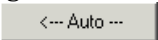
---

**Note** tutorialcal.mat is an example calibration file in the mbctraining folder.

---

- 4 Highlight both the table in the **Contents of Calibration File** pane and the table in the **Calibratable Blocks** pane that you want to associate with it.

- 5 Associate these two files by clicking .

To associate all the items listed in the **Calibratable Blocks** pane with items having the same names listed in the **Contents of Calibration File** pane, click .

- 6 Check the display of your table, then click **Close**.

Association buttons

Contents of calibration file

Select the axis or table to be calibrated.

Manually set up the table or normalizer.

Contents of Calibration File

| Name        | Size                                                         |
|-------------|--------------------------------------------------------------|
| Axis_x      | 10                                                           |
| Table_x     | 10                                                           |
| Axis_z      | 10                                                           |
| Axis_y      | 10                                                           |
| Table_yz    | 10 x 10                                                      |
| Norm_L      | 10                                                           |
| Norm_A      | 10                                                           |
| Cal file:   |                                                              |
| Data file:  |                                                              |
| Type:       |                                                              |
| Cal. level: |                                                              |
| Info:       | 10 calibration items, comprising:<br>2 Tables<br>8 Functions |

Block: SPK

| L \ N | 500    | 1000   | 1500   | 2000   | 2500   | 3000   | 3500   | 4000   |
|-------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0.1   | 25     | 25     | 25     | 25     | 25     | 25     | 25.001 | 25.001 |
| 0.2   | 25     | 25     | 25     | 25     | 25.91  | 26.82  | 27.73  | 28.64  |
| 0.3   | 25     | 25     | 25     | 25     | 26.82  | 28.64  | 30.46  | 32.28  |
| 0.4   | 25     | 25     | 25     | 24.999 | 27.729 | 30.459 | 33.189 | 35.919 |
| 0.5   | 25     | 25     | 24.999 | 24.999 | 28.639 | 32.279 | 35.919 | 39.558 |
| 0.6   | 27.375 | 27.515 | 28.264 | 29.014 | 32.499 | 35.984 | 39.469 | 42.954 |
| 0.7   | 29.749 | 30.03  | 31.529 | 33.028 | 36.358 | 39.689 | 43.019 | 46.349 |
| 0.8   | 32.124 | 32.545 | 34.794 | 37.042 | 40.218 | 43.393 | 46.569 | 49.744 |
| 0.9   | 34.499 | 35.06  | 38.058 | 41.057 | 44.078 | 47.098 | 50.119 | 53.14  |
| 1     | 36.874 | 37.575 | 41.323 | 45.071 | 47.937 | 50.803 | 53.669 | 56.535 |

Close

Check the display of your table.

---

**Note** You can add additional file formats to configure CAGE to work with your processes.

---

Contact The MathWorks for details about adding file formats at [www.mathworks.com/products/mbc/](http://www.mathworks.com/products/mbc/).


### Finding Items in the Calibration File


In a large calibration file, you might want to search for an item by name. To search the **Contents of Calibration File** pane,

- 1 Click the **Contents of Calibration File** list.
- 2 Type the first few letters of the item that you are searching for.


The cursor moves to the letters specified as you type.

### Copying Table Data from Other Sources

You can paste table values from Excel, for example, by copying the array in Excel and clicking **Paste** :

- 1 Open the desired Excel file and copy the array that you want to import.
- 2 In the **Calibration Manager** dialog box, click **Paste** .

You can also set up a table from a text file:

- 1 Click  in the toolbar.
- 2 Select the desired file, then click **Open**.

---

**Note** If the size of the table is different from the file that you are copying, CAGE changes the size of the table in the session.

---

## Table Properties

Table properties allow you to edit the precision of selected tables and normalizers according to the way tables are implemented in the electronic control unit (ECU). The ECU designer chooses the type of precision for each element to make best use of available memory or processor power.

To edit the precision of a table or normalizer,

- 1 Select **Edit Precision** in the **Calibration Manager** dialog box.

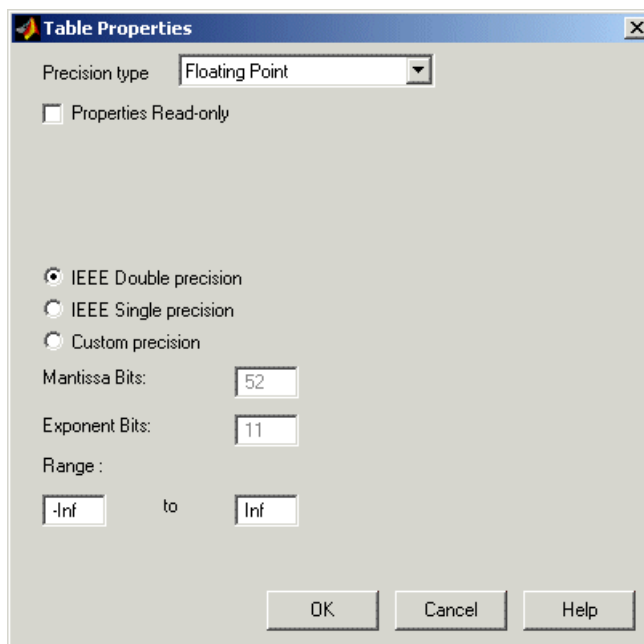
Alternatively, if you highlight a table in a feature calibration, display the table properties by selecting **Table -> Properties**.

- 2 Decide whether you want the precision to be writable, then either select or clear the **Properties Read-only** check box.
- 3 Decide the type of precision you require for the table:
  - **Floating Point** (See “Floating-Point Precision” on page 13-6.)
  - **Polynomial Ratio, Fixed Point** (See “Polynomial Ratio, Fixed Point” on page 13-8.)
  - **Lookup Table, Fixed Point** (See “Lookup Table, Fixed Point” on page 13-10.)

The following sections describe these types of precision in detail.

### Floating-Point Precision

The advantage of using floating-point precision is the large range of numbers that you can use, but that makes the computation harder.



There are three types of floating-point precision that you can choose from:

- **IEEE double precision (64 bit)**
- **IEEE single precision (32 bit)**
- **Custom precision**

If you choose **Custom precision**, you must specify the following:

- Number of mantissa bits
- Number of exponent bits
- Range of values restricting the values in the table

When you are done, click **OK**.

### See Also

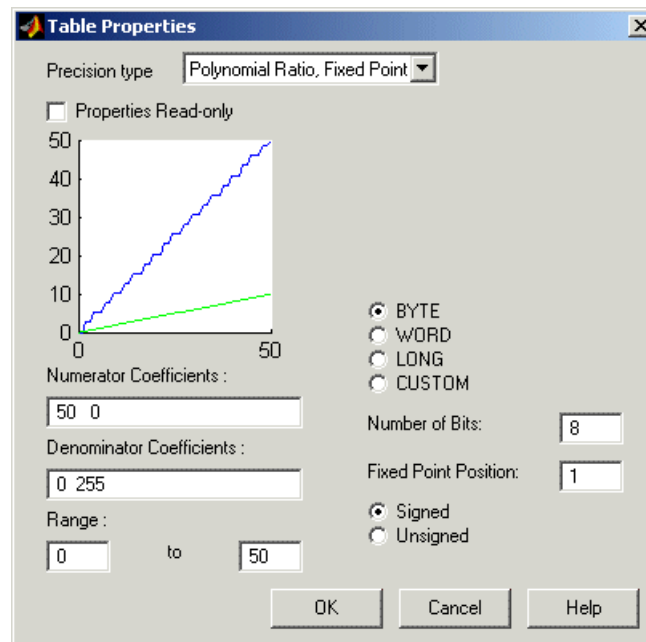
- For more information on IEEE double precision in MATLAB<sup>®</sup>, see Moler, C., "Floating points," *The MathWorks Company Newsletter*, 1996.

## Polynomial Ratio, Fixed Point

The advantage of using fixed-point precision is the reduction in computation needed for such numbers. However, it restricts the numbers available to the user.

For example, the polynomial ratio is of the form (see the ratio shown)

$$y = \frac{50x + 0}{0 + 255}$$



To edit the polynomial ratio,

- 1 Select the **Numerator Coefficients** edit box and enter the coefficients. In the preceding example, enter 50 0.

The number of coefficients determines the order of the polynomial, and the coefficients are ordered from greatest to least.

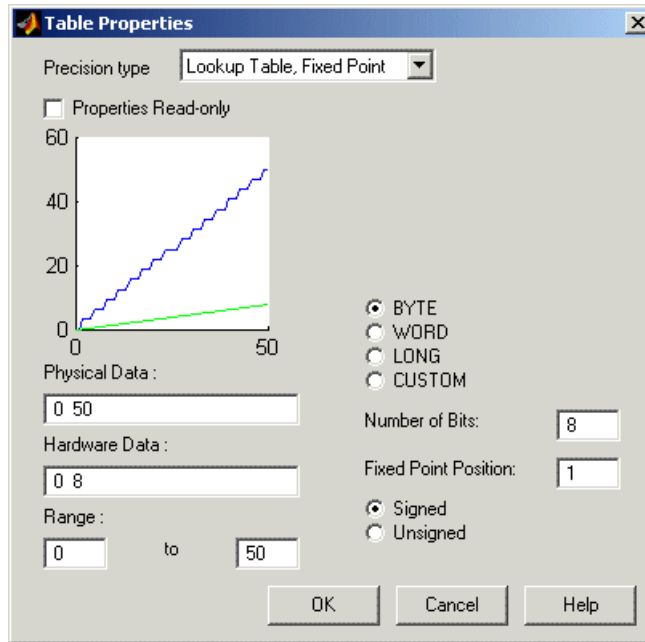
- 2** Select the **Denominator Coefficients** edit box and enter the coefficients. In the preceding example, enter 0 255.
- 3** Determine the range of values that you want to have in the table. In the preceding example, enter 0 50.

To edit the size of the precision, choose from

- **Byte (8 bits)**
- **Word (16 bits)**
- **Long (32 bits)**
- **Custom**

Next, determine whether you want the numbers to be signed (negative and positive) or unsigned (nonnegative).

## Lookup Table, Fixed Point



The advantage of using fixed-point precision is the reduction in computation needed for such numbers. However, it restricts the numbers available to the user.

For example, consider using a lookup table for the physical quantity *spark advance for maximum brake torque (MBT spark)*. Typically, the range of values of MBT spark is 0 to 50 degrees. This is the physical data. The ECU can only store bytes of information and you want to restrict the hardware store to a range of 0 to 8, with at most one decimal place stored.

To adjust the fixed-point precision of the lookup table,

- 1 Select the **Physical Data** edit box and enter the range of the physical data. In the preceding example, enter 0 50.
- 2 Select the **Hardware Data** and enter the range to store. In the preceding example, enter 0 8.



- 3** Determine the range of values that you want to have in the table. In the preceding example, enter 0 50.

To edit the size of the precision, choose from

- **Byte (8 bits)**
- **Word (16 bits)**
- **Long (32 bits)**
- **Custom**

In the preceding example, the hardware is restricted to 8 bytes and to one decimal place.

- 4** Choose whether you want the numbers to be signed (negative and positive) or unsigned (nonnegative) by clicking the radio buttons.



# Surface Viewer

---

This section includes the following topics:

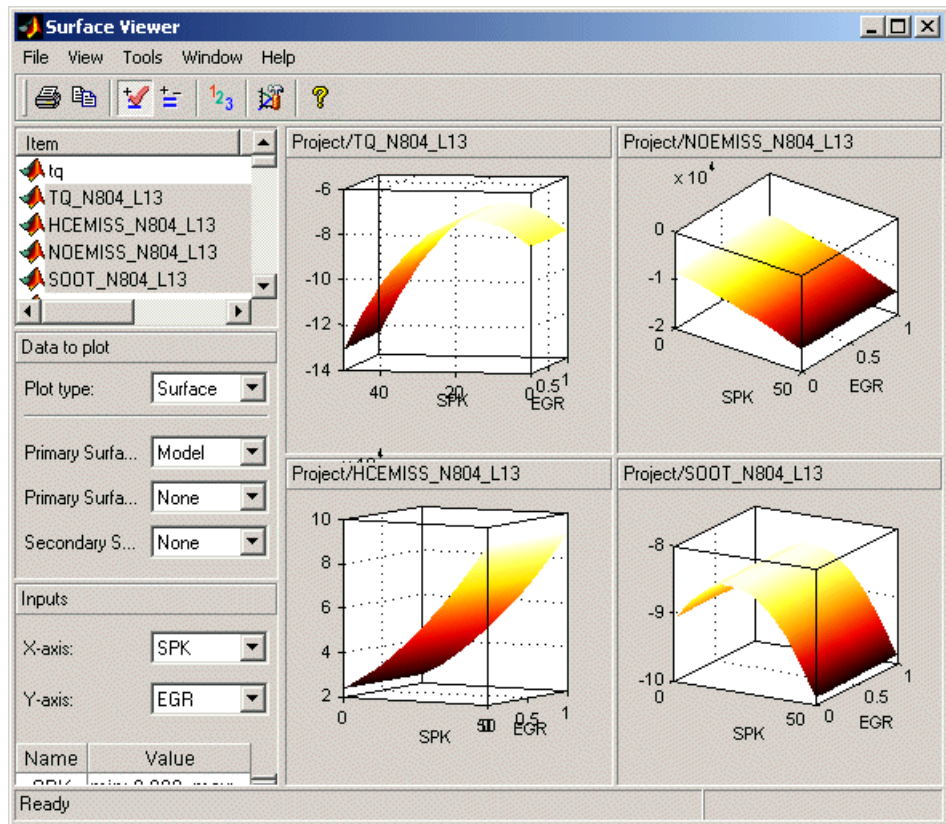
|                                               |                                                                                                                                |
|-----------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| The Surface Viewer in CAGE (p. 14-2)          | Introduction to the <b>Surface Viewer</b> .                                                                                    |
| Viewing a Model or Strategy (p. 14-3)         | How to view models or strategies.                                                                                              |
| Setting Variable Ranges (p. 14-5)             | How to set ranges for display.                                                                                                 |
| Displaying the Model or Feature (p. 14-7)     | This section describes the display options available: surface, contour, single line, single value, multiline, movie, or table. |
| Displaying Errors (p. 14-12)                  | How to display errors: predicted error of the model and the error between a model and a strategy (feature error).              |
| Making Movies (p. 14-14)                      | How to make a movie that allows you to see an evaluation over two variables at successive values of a third variable.          |
| Printing and Exporting the Display (p. 14-16) | How to print and export displays.                                                                                              |

## The Surface Viewer in CAGE

The **Surface Viewer** enables you to view the model or the feature as it varies over the ranges of its variables. You can automatically step through values of a variable, to make a movie of the behavior of the feature or model. You can view the model or feature using a variety of plot types.

**Note** The **Surface Viewer** is only available when you are viewing models or the feature node of a feature calibration.

Following is an example of the **Surface Viewer** displays.



## Viewing a Model or Strategy

To access the surface viewer, select **Tools -> Surface Viewer** or click  on the toolbar.

These are the main steps to view the model or feature using the **Surface Viewer** dialog box:

- 1 The model or feature selected when you open the **Surface Viewer** is displayed in the plot. If you have more than one model or feature, select what to display from the top **Items** list.

You can multiply select up to 4 items at once using **Ctrl+click** (the plot view on the right divides into a maximum of 4 plots). All the settings below the **Items** list apply to all plots. If one of the features selected in the **Items** list does not contain the appropriate input variables you select to plot, there will be no plot for that item.

- 2 Select the ranges for the variables. (See “Setting Variable Ranges” on page 14-5.)
- 3 Display the model or feature in the correct format. (See “Displaying the Model or Feature” on page 14-7.). You can view surfaces, contour plots, single and multilines, movies, tables, and single values.

For example, as you view a feature, you can view either the strategy, the model associated with that feature, the error between the model and the strategy, or the prediction error if the model was imported from the Model Browser. You can also use one of these factors to shade the surface formed by one of the other factors, and you can select any two factors to display simultaneously as two surfaces.

- You can make a movie. (See “Making Movies” on page 14-14). This enables you to view the model or feature as it steps through several values of a variable. For example, if you want to view a feature calibrated for maximum brake torque (MBT) as it varies over exhaust gas recycling (EGR), you can make a movie of the feature.
- You can also print or export the display. (See “Printing and Exporting the Display” on page 14-16.)

The following sections describe these steps in more detail.

Features in the project and their inputs

The model in the feature, shaded by the error (in this case)

3. Plot controls

Variable ranges

Axes controls

Surface Viewer

File View Tools Window Help

Project/Branch 1/Fn\_Feature

| Item           | Inputs  |
|----------------|---------|
| Torque_Feature | N, L, A |
| Fn_Feature     | x, y, z |

Data to plot

Plot type: Surface

Primary Surface: Model

Primary Surface... Error (strateg)

Secondary Surf... None

Inputs

X-axis: x

Y-axis: y

| Name | Value                   |
|------|-------------------------|
| x    | min: -5.000, max:5.0... |
| y    | min: 0.000, max:10.0... |
| z    | 10.000                  |

Ready

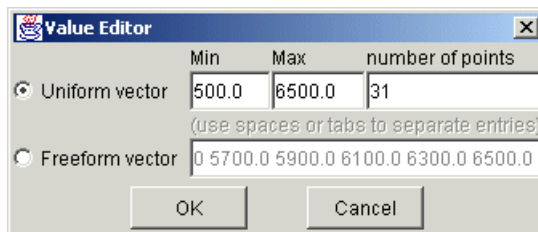
max 1.00

min -1.6

## Setting Variable Ranges

The **Surface Viewer** does not work over continuous ranges, only at discrete points. You must specify, for the model or feature, the discrete points you want to include in the display. You can display models or features over a range of points. To edit the displayed values of a variable, double-click in the value box for the appropriate variable.

- Variables not being used for the axes plotted have a single value for that plot; to edit the displayed value for these variables you can type directly into the edit box after double-clicking.
- For variables specified by the axes drop-down menus, the value box displays the range over which that variable is plotted and the number of points plotted across that range. To edit both the range and the number of points, double-click the value box. The **Value Editor** opens.



Here you can indicate the points to include in the display. You can specify

- The minimum and maximum values and the number of points across that range by choosing **Uniform Vector** and typing in the edit boxes **Min**, **Max**, and **Number of points**.
- Each discrete point at which you want to evaluate the model (or feature), by choosing **Freeform vector**, and then typing the required values.

For example, if you want to display the variable  $x$  at 0, 1, 7, 30, and 50, enter the following in the **Freeform vector** edit box, separated by tabs or spaces:

```
0 1 7 30 50
```

Click **OK** to apply your changes to the plot.

When you alter the variables, you can select whether you want the display to update automatically or not. You can toggle the automatic update on and off by

selecting **Tools -> Auto-Evaluate**. When you want to update the display, select **Tools -> Evaluate Now**. Both of these options have equivalent toolbar buttons:

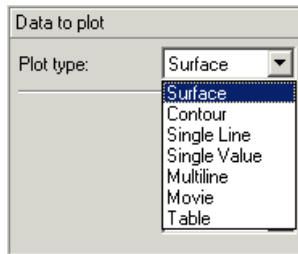




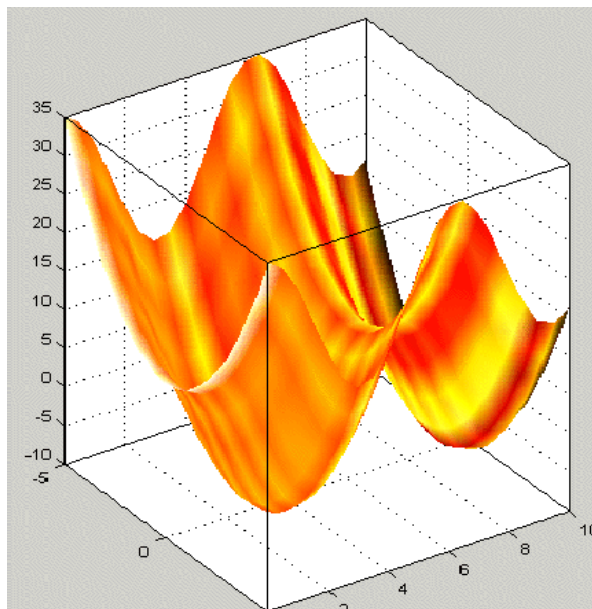
## Displaying the Model or Feature

There are several aspects of the display to consider before making the movie.

- You can rotate the surface plots by left-clicking and dragging.
- The **Plot Type** drop-down menu gives the options on how to display the model or feature, as shown below.



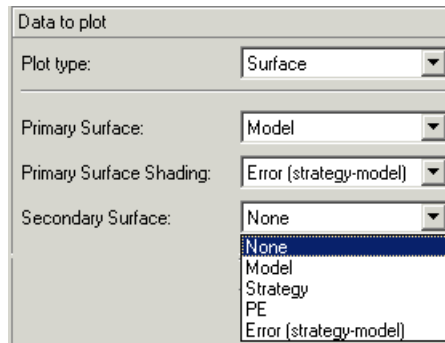
- Use the options in this menu to display the model or feature in the following ways:
  - Surface



Note that if you are using the surface viewer to view a feature, you can choose the following options to display:

- Model
- Strategy
- Prediction Error
- Error (between the model and the strategy)

You can choose these options from the drop-down menus for **Primary Surface**, **Primary Surface Shading**, and **Secondary Surface**, as illustrated below.



You can view any of these options alone as a primary surface (by leaving the other two options set to **None**). You can add a second option to shade the primary surface, for example to color your model surface with the error between the model and the strategy, to highlight problem areas. This is the setup in the example above.

When you choose to shade a primary surface, a color bar appears to the right of the plot to show you the scale. You can change the maximum and minimum values of the shading factor by typing in the edit boxes under the color bar. You can see an example like this in “Viewing a Model or Strategy” on page 14-3.

You can add a secondary surface to display any two of the options simultaneously, for example, your model and your strategy.

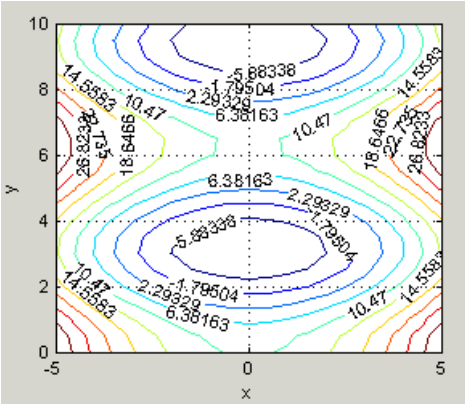
---

**Note** For information on the two different error displays available using the surface view, see the next section, “Displaying Errors” on page 14-12.

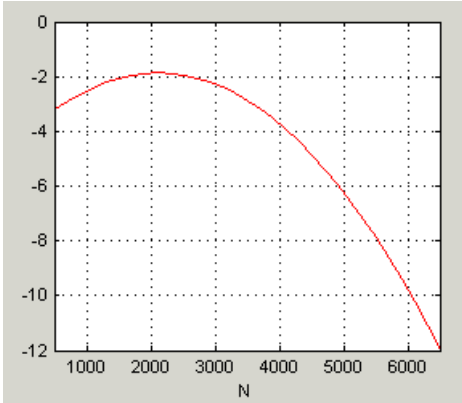
---

The following plot options are also available:

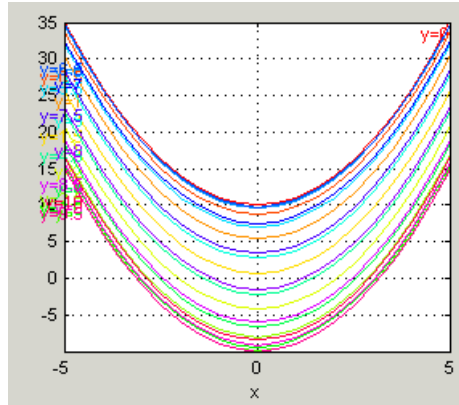
- Contour lines



- A single line



- A single value  
This displays the value of the model at the point you have specified in the variable value boxes.
- Multiline plot



- Movie  
The movie option allows you to see an evaluation over two variables at successive values of a third variable. For example, a model of torque might have speed (N), load (L), and air/fuel ratio (A) as inputs. The movie option allows you to view how the torque model behaves over the ranges of speed and load for successive values of air/fuel ratio. See “Making Movies” on page 14-14.
- A table  
Choose two variables to be the axes of your table and set the range and number of points in the same way as for all the plots. Set single values for any other variables. For more information, see “Setting Variable Ranges” on page 14-5.  
In the table view you can select **View -> Statistics**, or click the equivalent toolbar button. This opens a dialog box with a list of the summary statistics (mean, standard deviation, maximum, or minimum) for the current table output. You can also view the statistics of your currently selected model, strategy, or error from the other plots.

| Project/Branch 1/Fn_Feature |        |        |        |
|-----------------------------|--------|--------|--------|
| x\y                         | 0.000  | 0.500  | 1.000  |
| -5.000                      | 35.000 | 33.776 | 30.403 |
| -4.500                      | 30.250 | 29.026 | 25.653 |
| -4.000                      | 26.000 | 24.776 | 21.403 |
| -3.500                      | 22.250 | 21.026 | 17.653 |
| -3.000                      | 19.000 | 17.776 | 14.403 |
| -2.500                      | 16.250 | 15.026 | 11.653 |
| -2.000                      | 14.000 | 12.776 | 9.403  |
| -1.500                      | 12.250 | 11.026 | 7.653  |
| -1.000                      | 11.000 | 9.776  | 6.403  |
| -0.500                      | 10.250 | 9.026  | 5.653  |
| 0.000                       | 10.000 | 8.776  | 5.403  |
| 0.500                       | 10.250 | 9.026  | 5.653  |
| 1.000                       | 11.000 | 9.776  | 6.403  |
| 1.500                       | 12.250 | 11.026 | 7.653  |

## Displaying Errors

There are two different error displays available in the surface display options for primary and secondary surfaces and surface shading:

- Error between the model and the strategy (See “Feature Error Data” following.)
- Predicted error of the model (See “Prediction Error Data” on page 14-12.)

### Feature Error Data

When you are viewing a feature, this displays the error between the strategy and the model.

To display the error, select **Error (strategy-model)** from the drop-down menu for primary or secondary surface. You can also choose to shade your primary surface with the error by using the **Primary Surface Shading** menu.

To view the error statistics, select **View -> Statistics**. This opens a dialog box with a list of the summary statistics for the error between model or feature.

### Prediction Error Data

If the model is imported from the Model Browser, it is possible to display the *prediction error* (PE) data.

Prediction Error Variance (PEV) is a very useful way to investigate the predictive capability of your model. It gives a measure of the precision of a model’s predictions. PEV can also be examined in the Model Browser, both in the **Prediction Error Variance Viewer** and to shade surfaces in the **Model Selection** and **Model Evaluation** views. Here you can examine the PEV of designs and models. When you export the model to CAGE you can see this data in the **Surface Viewer** in the Prediction Error option. See the Model Browser GUI Reference and Technical Documents for details about the calculation of Prediction Error.

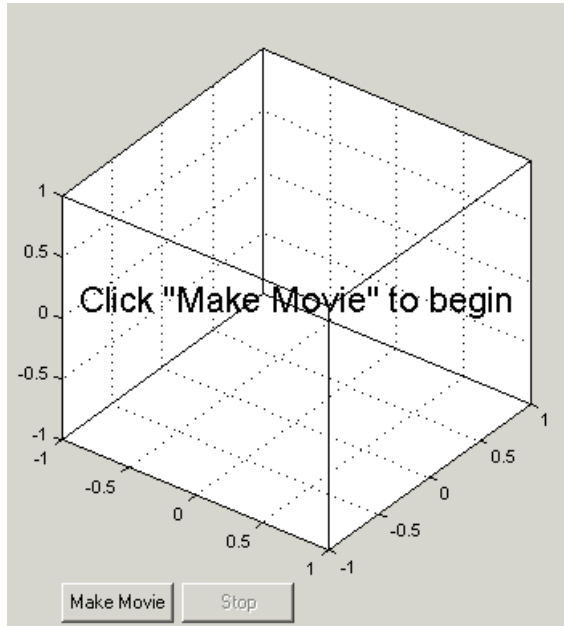
### Viewing the Predicted Error

Select Prediction Error from the drop-down display menus for primary or secondary surfaces. You can also choose Prediction Error to shade your primary surface. As with all other plots, you can view the statistics for the Prediction Error displayed by selecting **View -> Statistics**. The mean,

standard deviation, and so on are calculated over the range specified in the variable value boxes.

## Making Movies

Choose **Movie** from the **Plot Type** drop-down menu in the **Data to Plot** pane.



The movie option allows you to see an evaluation over two variables at successive values of a third variable. For example, a model of torque might have speed (N), load (L), and air/fuel ratio (A) as inputs.

The movie option allows you to view how the torque model behaves over the ranges of speed and load for successive values of air/fuel ratio.

- 1 You must choose three variables from the **X**, **Y**, and **Time** drop-down menus, to indicate which variable you want to display on which axis.

You can view the model surface plotted across the range of two variables, and define the third variable as “time” to see the model surface change across the third variable’s range.

- 2 Define the variable ranges as usual using the **Value** boxes for the inputs. See “Setting Variable Ranges” on page 14-5.



- 3** Click **Make Movie**. You can click **Stop** if you change your mind, for example, if you have set a very large number of points in the **Time** variable and the movie making is taking too long.
- 4** Once the movie is made, you can replay the movie by clicking **Replay**. Note that the **Stop** button does not function during the playback mode.

## Printing and Exporting the Display

To print the display, select **File -> Print**. Selecting **File -> Copy to Clipboard** (or using the equivalent toolbar button) copies the plot image to the clipboard. This is useful if you want to place plot images into other applications.

You can also export the display data to a comma-separated variable file.

To export the display, select **File -> Export to CSV**. The currently selected option is exported. The primary input to the first plot is exported (this is the top left if you have multiple plots). The output is the values at the grid of points specified by the current ranges and input values. The inputs for shading and secondary surfaces are not exported.

Note that you cannot print table plots or copy them to the clipboard: if you want to transfer them it is more useful to export them to CSV files and then load them into Excel.

# Manual Calibration and the History Display

---

This section includes the following topics:

|                                             |                                                                         |
|---------------------------------------------|-------------------------------------------------------------------------|
| Using the Manual Calibration View (p. 15-2) | An overview of the functionality in the <b>Manual Calibration</b> view. |
| Adding and Deleting Tables (p. 15-3)        | How to add and remove tables.                                           |
| Using the History Display (p. 15-5)         | Comparing and reverting to previous versions.                           |

## Using the Manual Calibration View

Select this view by clicking the **Manual Calibration** button. It opens automatically if you add a table using the **File -> New** menu.



The **Manual Calibration** view lists all the tables and normalizers in the current CAGE session.

Here you can add or delete tables and normalizers, and you can calibrate them manually. Once you have added new tables here you can also fill them using experimental data by going to the **Data Sets** view.

You can use the **History** display from here (and from any other table or normalizer view in CAGE) to view and manage changes in your tables. You can use the **History** display to reverse changes and revert to previous versions of your tables.

See also

- “Tutorial: Filling Tables from Data” on page 5-1

## Adding and Deleting Tables

When you are calibrating a collection of tables using either **Feature** or **Tradeoff** calibrations, you cannot easily add or delete tables without affecting the entire calibration.

You might want to add a table (for example, to fill by reference to experimental data). Or you might want to delete a table (for example, after adjusting a strategy for a feature calibration).

To add or delete tables, you can first select the **Manual Calibration** view. CAGE automatically switches to this view if you add a table using the **File -> New** menu items.



The **Manual Calibration** view lists all the tables and normalizers in the current CAGE session.

### Adding Tables

To add a table to a session,

- 1 Decide whether you want to add a one- or a two-dimensional table.

For example if you want to add a modifier table to account for the variation in exhaust gas recirculation, add a one-dimensional table (which has one input). If, however, you want to add a table with speed and load as its normalizer inputs, then add a two-dimensional table.

- 2 Select **File -> New -> 1D Table** or **File -> New -> 2D Table** as appropriate.


This automatically switches you to the **Manual Calibration** view.

- 3 Select **Tools -> Calibration Manager** to determine the size of your new table. For information, see “Setting Up Tables” on page 13-2.

You can manually calibrate by entering values in any table.

## Deleting Tables

To delete a table or a normalizer from a session,

- 1 Select **Manual Calibration** view.
- 2 Highlight the required table or normalizer.
- 3 Click .

---

**Note** When deleting items, you must delete from the highest level down. For example, you cannot delete a table that is part of a feature; you must delete the feature first.

---

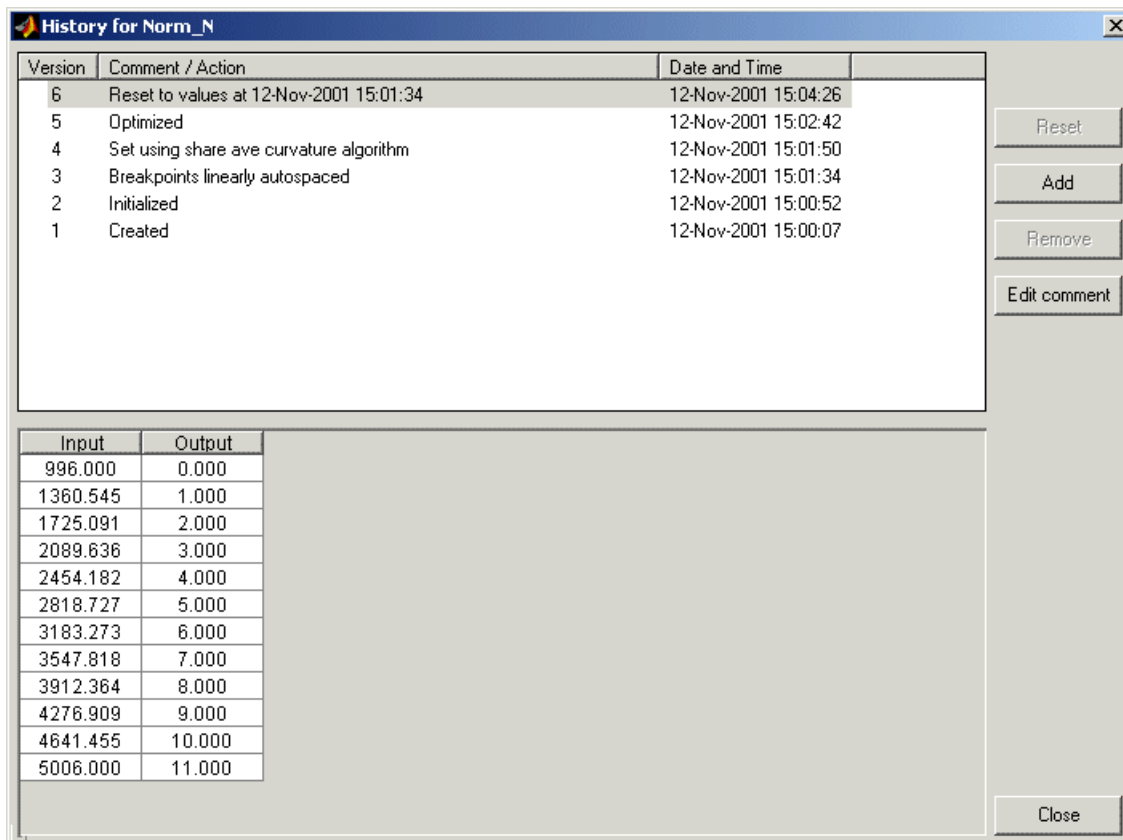
## Using the History Display

The **History** display enables you to view the history of any table or normalizer in a CAGE session.

The **History** display lets you

- Revert to previous versions of tables and normalizers (See “Resetting to Previous Versions” on page 15-7.)
- Compare different versions of tables and normalizers (See “Comparing Versions” on page 15-8.)

You can view the **History** display of a table or normalizer by selecting **View – > History**.



The upper pane of the **History** display lists all the versions of the highlighted object.

The lower pane displays the normalizer or table of the highlighted version.



## Resetting to Previous Versions

To reset the normalizer or table to a previous version,

- 1 Highlight the previous version that you want to revert to.
- 2 Click **Reset**.

---

**Note** Tables are independent of normalizers, so if you reset a table to a previous version you must also reset the normalizers to that version (if they have changed).

---

To remove previous versions of the object or comments,

- 1 Highlight the version that you want to remove.
- 2 Click **Remove**.

## Adding and Editing Comments About Versions

To add comments,

- 1 Click **Add**.
- 2 In the dialog box enter your comment.
- 3 Click **OK**.

To edit comments,

- 1 Select the comment that you want to edit.
- 2 Click **Edit comment**.
- 3 In the dialog box, edit the comment.
- 4 Click **OK**.

## Comparing Versions

To compare two different versions of a normalizer or table, highlight the two versions using **Ctrl+click**. Note the following:

- The lower pane shows the difference between the later and the earlier versions.
- Cells that have no entries have no difference.
- Cells that have red entries have a lower value in the later version.
- Cells that have black entries have a higher value in the earlier version.

### Comparison Table in the History Display

|       | 1468.000 | 1772.000 | 2153.000 | 2616.000 | 3117.000 | 3549.000 | 3963.000 |
|-------|----------|----------|----------|----------|----------|----------|----------|
| 0.200 |          |          |          |          |          |          |          |
| 0.259 |          |          |          |          |          |          |          |
| 0.323 |          | -15.347  |          |          |          |          |          |
| 0.392 |          |          |          |          |          | -15.947  |          |
| 0.462 |          |          | -13.149  | -9.119   |          |          |          |
| 0.530 |          | 13.703   |          |          |          | 9.764    |          |
| 0.594 |          |          |          | 19.264   |          |          |          |
| 0.646 |          |          |          |          |          |          |          |
| 0.694 |          |          |          |          |          |          |          |
| 0.731 |          |          |          |          |          |          |          |
| 0.766 |          |          |          |          |          |          |          |
| 0.800 |          |          |          |          |          |          |          |

## A

aliases 7-12

## B

breakpoints

deleting 8-18

filling 8-5

initializing 8-4

locking in Breakpoint Spacing display 8-18

locking in Input/Output display 8-16

locking in Normalizer display 8-17

optimizing 8-9

## C

calibration files

finding items 13-5

calibrations

exporting 7-21

constants

adding 7-10

editing 7-11

## D

data sets

importing data from a table 12-6

importing experimental data 12-4

plotting multiple outputs 12-14

plotting outputs 12-13

setting up manually 12-9

viewing as tables 12-11

## E

error

displaying for normalizers 8-22

displaying for tables 9-35

displaying in the surface viewer 14-12

extrapolation mask

generating automatically 9-20

using 9-20

## F

factors

assigning to data 12-22

creating error between two factors 12-10

linking 12-20

features

calibrating 9-2, 9-37

exporting 7-21

filling 9-37

initializing 9-37

optimizing 9-37

setting up 9-4

formulas

adding 7-10

editing 7-11

## H

History display

using 14-5

## L

lookup tables

calibrating in a feature calibration 9-11

filling by extrapolation 9-19

filling using a model 9-13

initializing 9-12

inverting 9-21

## M

### models

- adding 7-17
- assigning to features 9-5
- displaying curvature 8-19
- displaying in tradeoff calibrations 10-7
- displaying in tradeoff tutorial 3-6
- displaying multiple slices 8-19
- editing 7-18
- editing connections 7-19
- importing 7-15
- setting up 7-13

## N

### normalizers

- calibrating 8-3
- comparison between model and feature 8-20
- copying breakpoint values 13-5
- exporting 7-21
- filling 8-5
- initializing 8-4
- optimizing 8-9

## O

### optimization

- output view 11-25
- setup 11-4
- toolbar 11-14
- user-defined 11-37
- view 11-2

## P

### precision

- changing the precision of a table 13-6
- fixed point, lookup table 13-10
- fixed point, polynomial ratio 13-8
- floating point 13-6

predicted error

- displaying in the surface viewer 14-12

## R

ReduceError fill method 8-6

## S

set points 7-9

ShareAveCurv fill method 8-7

ShareCurvThenAve fill method 8-7

strategies

- constructing 9-7
- exporting 9-9
- importing 9-6
- setting up 9-5

sum optimization 6-32

surface viewer

- editing ranges of variables 14-5
- movie controllers 14-14

system requirements 1-10

## T

### tables

- adding to a tradeoff 10-6
- calibrating in a feature calibration 9-11
- calibrating in a tradeoff calibration 10-9
- comparing to a feature 9-33
- copying cell values 13-5

- exporting 7-21
- filling by extrapolation 9-19
- filling using a model 9-13
- graph of table 9-32
- initializing 9-12
- inverting 9-21
- locking cell values 9-31
- optimizing 9-16
- setting up 13-2

tradeoffs

- adding 10-5
- automatic 11-33
- calibrating 10-2
- exporting 7-21
- setting up 10-5

## V

- variable dictionaries
  - exporting 7-8
  - importing 7-8
- variable items
  - adding 7-9
- variables
  - adding 7-9
  - alias 7-12
  - editing 7-11
- version control
  - comparing versions 14-8

