Model-Based Calibration Toolbox

For Use with MATLAB® and Simulink®

- Computation
- Visualization
- Programming
- Simulation

CAGE User's Guide

Version 2

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

Mail

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

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

Model-Based Calibration Toolbox CAGE User's Guide

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)
	February 2003	Online only	Revised for Version 2.0 (Release 13+)
	June 2004	Online only	Revised for Version 2.1 (Release 14)
	June 2004	Online only	Revised for Version 2.1.1 (Release 14+)

Contents

1 Getting Started

2 Tutorial: Feature Calibration

3 Tutorial: Tradeoff Calibration

4 Tutorial: Data Sets

5 Tutorial: Filling Tables from Data

6 Tutorial: Optimization and Automated Tradeoff

7 Using CAGE

8 Normalizers

9 Feature Calibrations

10 Tradeoff Calibrations

11 Optimization in CAGE

12 Data Sets

13 Calibration Manager

14 Surface Viewer

15 Manual Calibration and the History Display

16 CAGE Case Studies

Index

1

Getting Started

This section includes the following topics:

Starting the CAGE Browser

To start the application, type

```
cage
```

at the MATLAB® command prompt.

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.

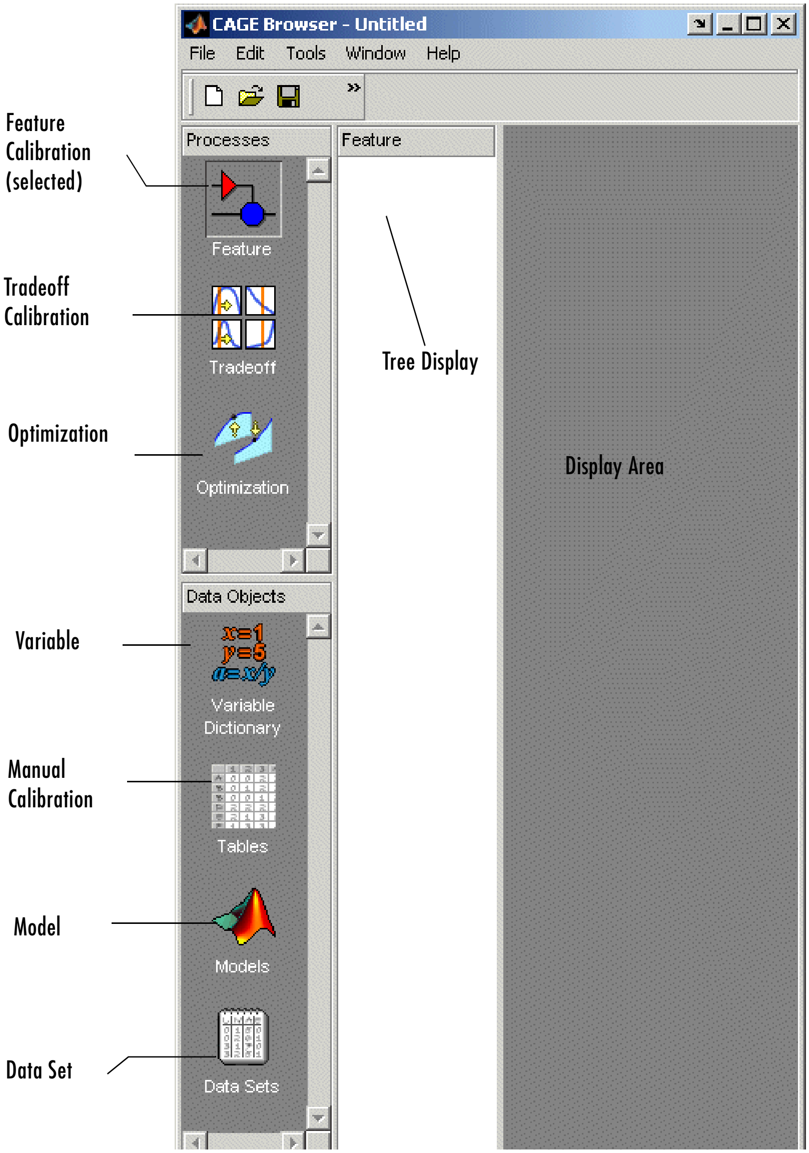
For example, after completing a calibration, you can import experimental data from a spreadsheet. You can use CAGE to compare your calibration to the data.

Navigating 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)

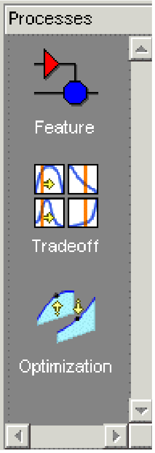
When you open CAGE, it looks like this.



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**.



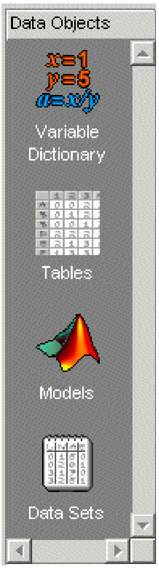
The **Processes** pane has three buttons:

- **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.
- **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 7-7.
- **Tables** enables you to see all the tables and normalizers in your session. You can calibrate tables manually here. See "Using the Tables View" on page 15-2.
- **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 7-14.
- **Data Sets** enables you to set up operating points for running optimizations, see the data produced by your objects and use that data to fill tables, 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.

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

2

Tutorial: Feature Calibration

This section includes the following topics:

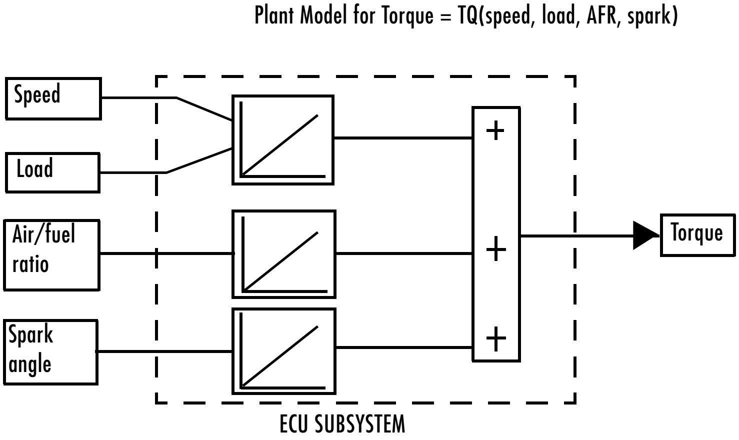
What Are Feature Calibrations?

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.

There are advantages to feature calibration compared with simply calibrating using experimental data. Data is noisy (that is, there is measurement error) and this can be smoothed by modeling; also models can make predictions for areas where you have no data. This means you can calibrate more accurately while reducing the time and effort required for gathering experimental data.

An example of an ECU subsystem control feature estimates the value of torque, depending on the four inputs: speed, load, air/fuel ratio (AFR), and spark angle.

A diagram of this ECU subsystem example follows.



In this tutorial example, there are three lookup tables:

- A speed-load table
- A modifier, or table, for AFR
- A modifier for spark angle

This tutorial takes you through the various steps required to set up this feature and then calibrate it using CAGE. You will use CAGE to fill the tables by comparing them with a torque engine model.

The model is a copy of the torque model built in the Model Browser's Quick Start tutorial using engine data. This illustrates how you can use the Model-Based Calibration Toolbox to map engine behavior and transfer this information to engine calibrations. You can construct a model using the Model Browser; then you can use CAGE to calibrate lookup tables by reference to the model.

Setting Up Calibrations

Start CAGE by typing

```
cage
```

at the MATLAB prompt.

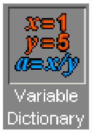
Note If you have a CAGE session open, select **File –> New –> Project**.

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,

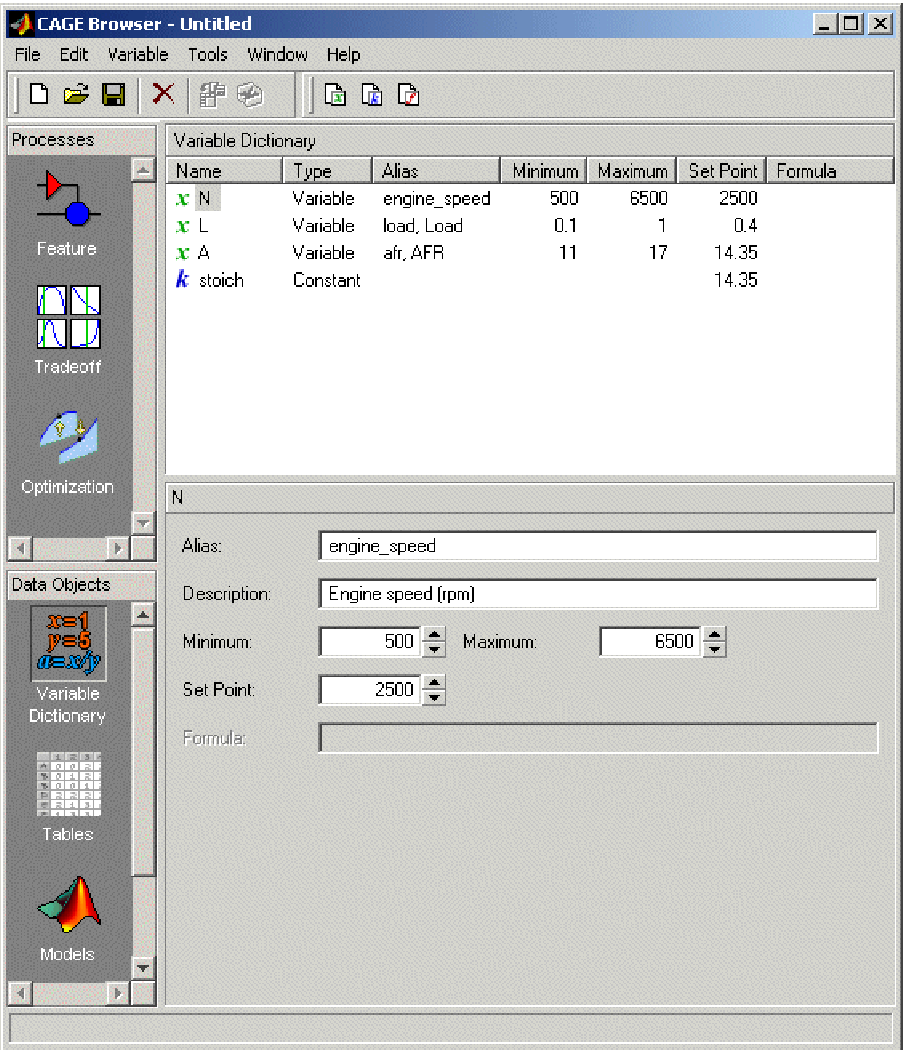
1 Select **File –> Import –> Variable Dictionary**.

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.



Variable Dictionary

	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	

N

Alias: engine_speed

Description: Engine speed (rpm)

Minimum: 500 Maximum: 6500

Set Point: 2500

Formula:

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** (or press **F2**) to rename the variable.

3 Enter SPK as the name.

4 Set the range of the variable by entering -5 as the **Minimum** and 50 as the **Maximum**.

The variable dictionary enables you to specify different names for the same variable, and also give descriptions of variables. For example, the variable spk might be referred to as S or spark in other models.

To ensure that CAGE recognizes an instance of S or spark as the same as spk, specify the aliases of SPK:

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 7-7.

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 opens a file browser.

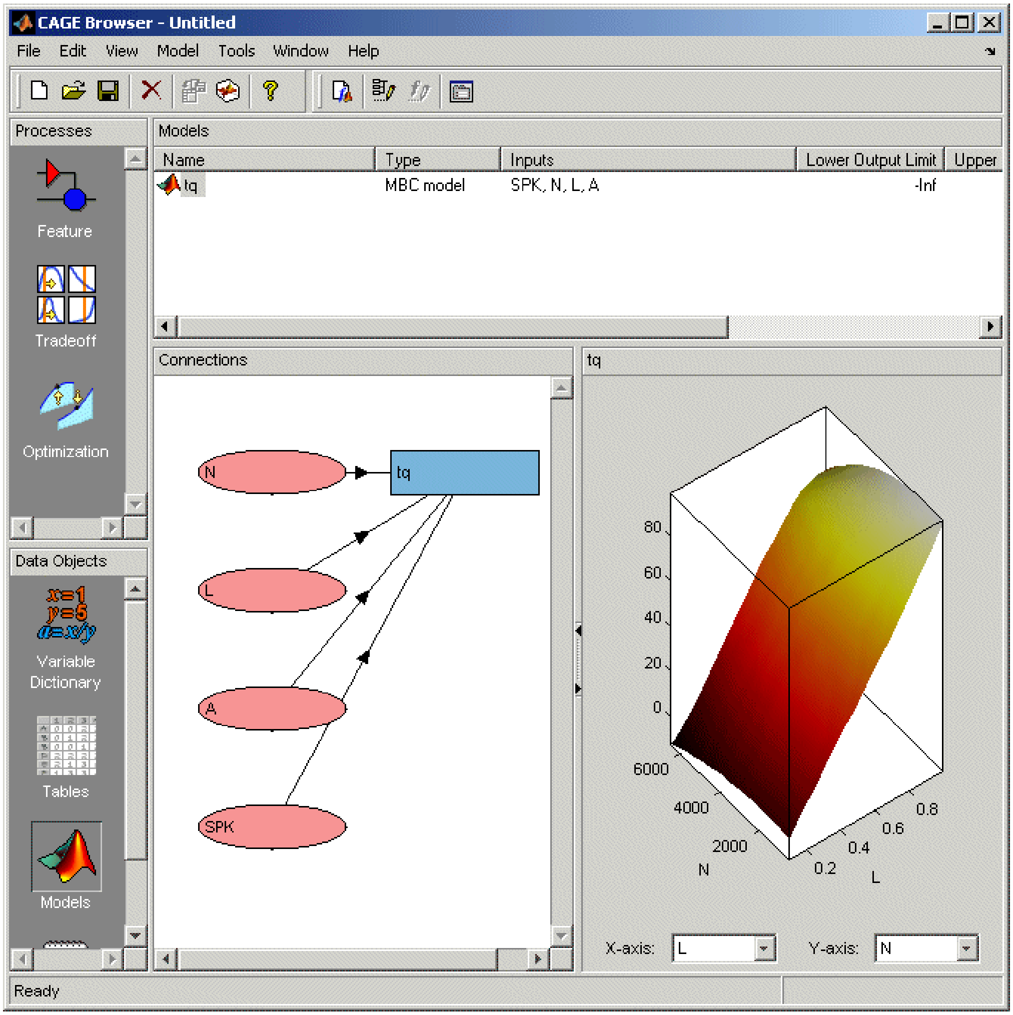
2 Browse to `matlab\toolbox\mbc\mbctraining`, select the `tutorial.exm` file (this is a copy of the torque model built in the Model Browser's Quick Start tutorial), and click **Open**. The **Model Import Wizard** appears.

3 There are two models stored in this file, `tq` and `knot`. Highlight `tq` and select the check box to **Automatically assign/create inputs**.

   CAGE automatically assigns variables in the variable dictionary to the model input factors or their aliases (as long as names are exact). If names are not exact you can select variables manually using the wizard.

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

When you complete the wizard, you return to the **Model**s view, as shown following.



Models

Name	Type	Inputs	Lower Output Limit	Upper
tq	MBC model	SPK, N, L, A	-Inf	

X-axis: L

Y-axis: N

For more information about models, see “Setting Up Your Models” on page 7-14.

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.

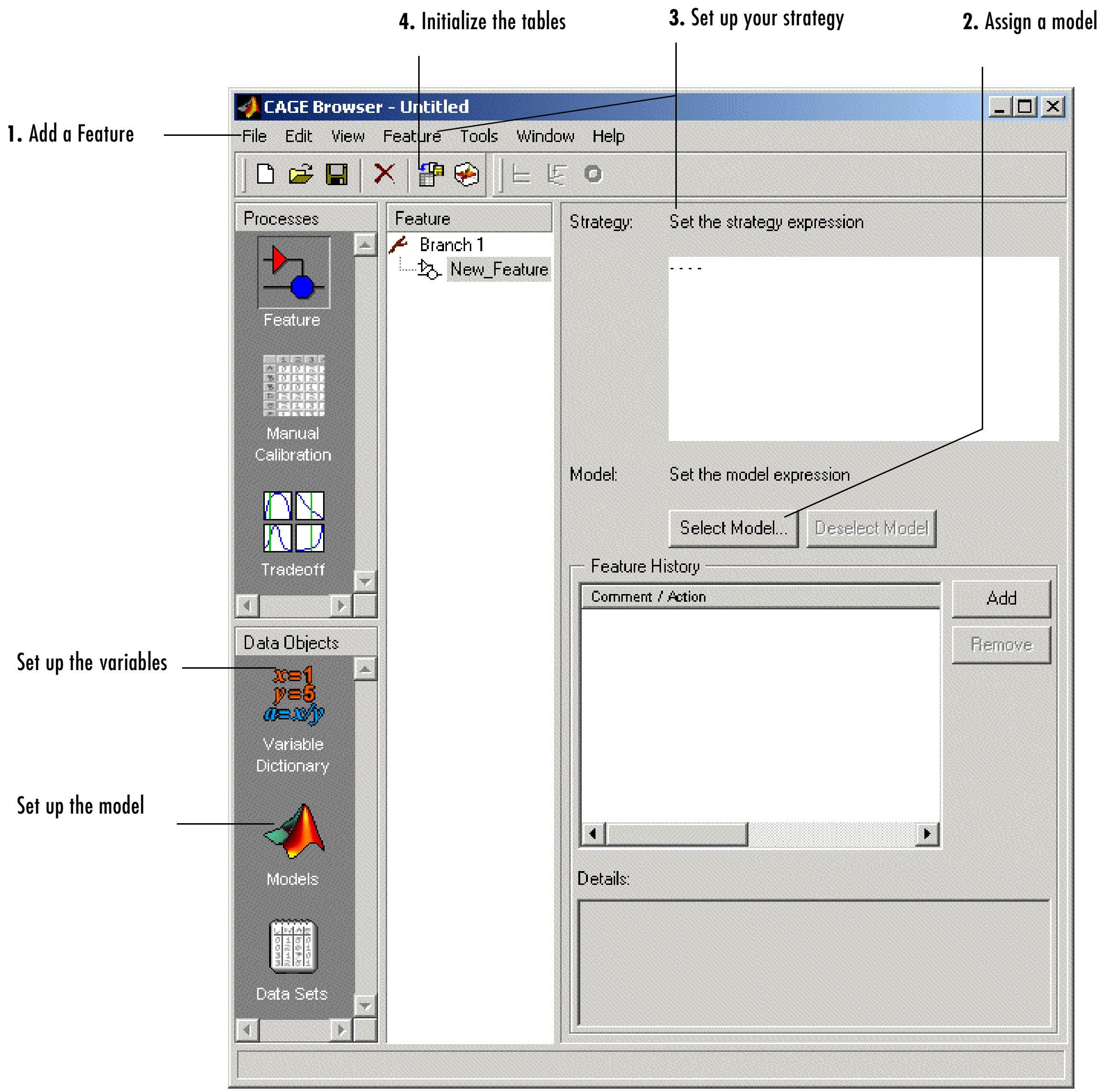
2. Click the **Select Model** button. This opens the **Select Model** dialog. Select `tq` (currently the only model in your project) and click **OK**.

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

3. Create a strategy. For instructions, 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. After you have created a strategy, the next step is to set up your tables. For more information, see the section, “Setting Up the Tables” on page 2-13.



Strategy: Set the strategy expression

- - - -

Model: Set the model expression

Setting Up the Strategy

The toolbox uses Simulink to graphically specify the strategy.

Importing a Strategy

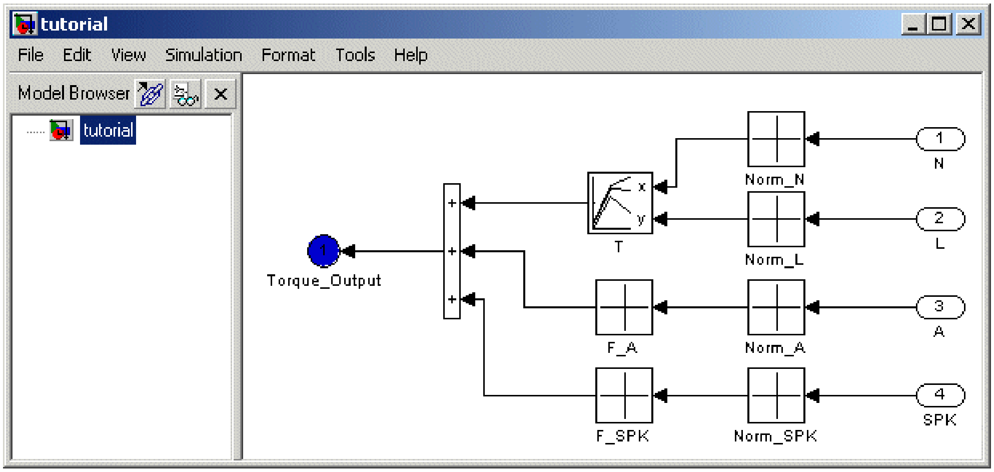
To import a strategy,

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 as follows:

```
New_Feature = T(Norm_N(N),Norm_L(L)) + F_A(Norm_A(A)) +
F_SPK(Norm_SPK(SPK))
```

5 Select **View –> Full Strategy Display** to turn off the full description and see this simplified expression:

```
New_Feature = T + F_A + F_SPK
```

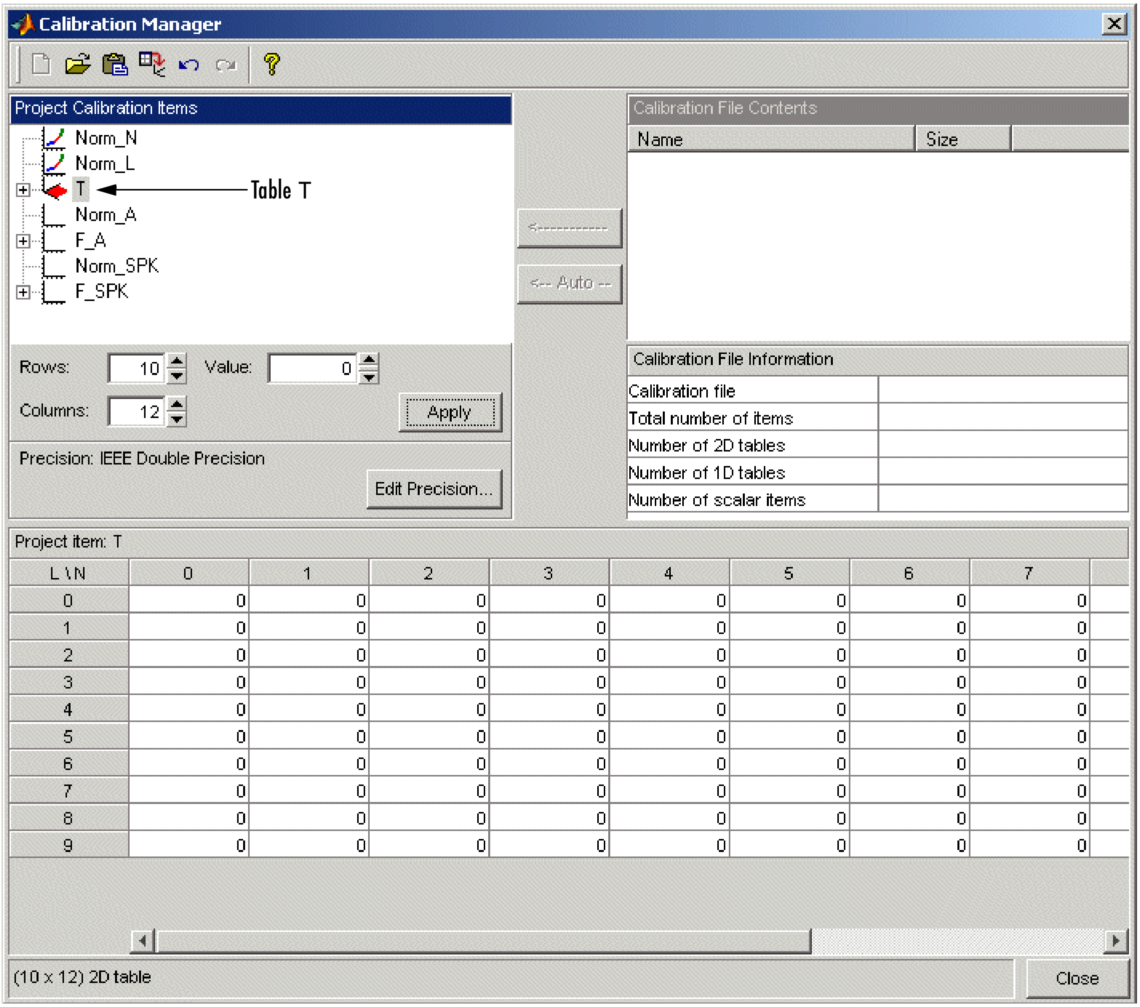
This shows the collection of tables that makes up the new feature — a torque table `T` (with normalizers in speed (`N`) and load (`L`)) combined with modifier tables depending on the values of air/fuel ratio and spark. You will fill these tables by using CAGE to compare them with the torque model.

For a more detailed description of the strategies, see "Setting Up Your Strategy" on page 9-6.

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.



Rows: 10 Value: 0

Columns: 12

Precision: IEEE Double Precision

Calibration File Information	
Calibration file	
Total number of items	
Number of 2D tables	
Number of 1D tables	
Number of scalar items	

Project item: T

L \ N	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0

(10 x 12) 2D table

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 Leave the value for each cell set to 0.

4 Click **Apply**. The pane changes 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 and press Enter.

  c Leave the value for each cell set to 0.

  d Click **Apply**.

6 Repeat step 5 for F_SPK.

Note The icons change as you initialize each table or function.

7 Click **Close** to leave the **Calibration Manager**.

After completing these steps, you can calibrate the lookup tables.

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:

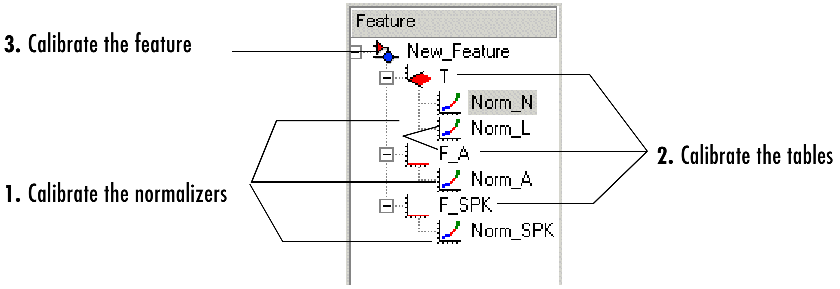
1 Calibrate the normalizers.

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.

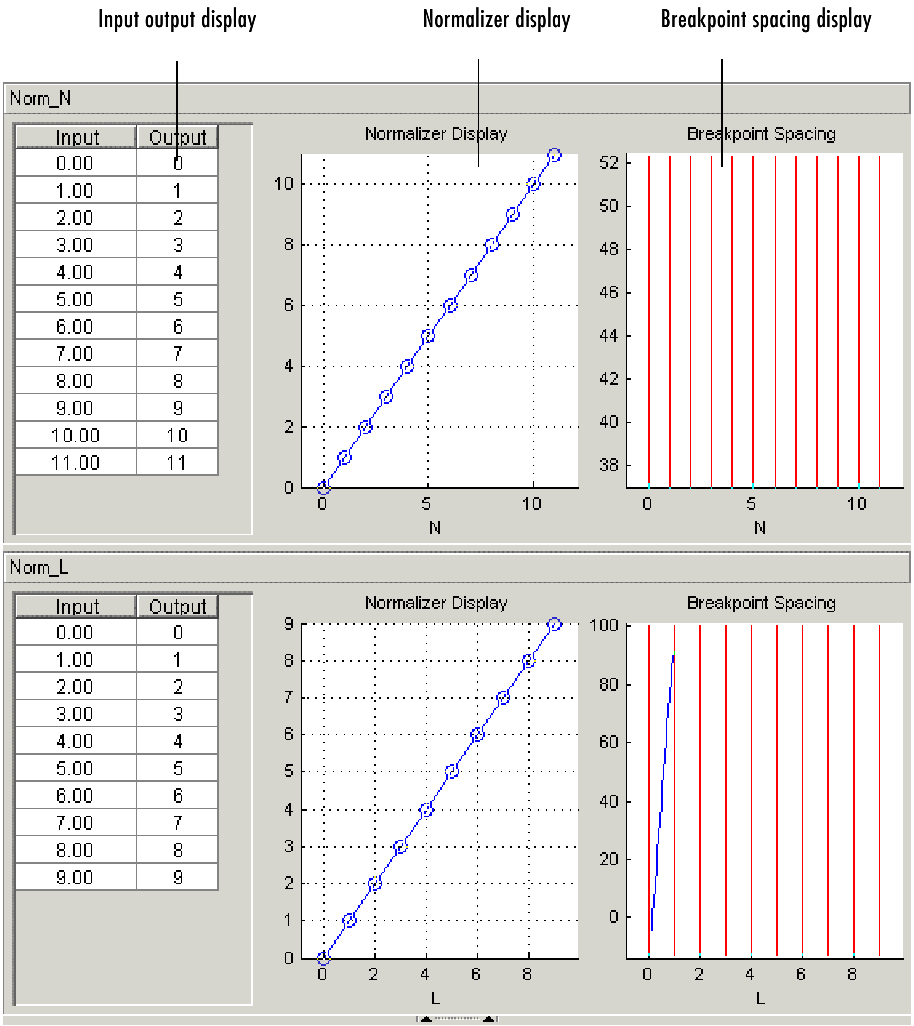


Each node in the display has a different view and different operations.

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.



Input output display

Normalizer display

Breakpoint spacing display

Norm_N

Input	Output
0.00	0
1.00	1
2.00	2
3.00	3
4.00	4
5.00	5
6.00	6
7.00	7
8.00	8
9.00	9
10.00	10
11.00	11

Norm_L

Input	Output
0.00	0
1.00	1
2.00	2
3.00	3
4.00	4
5.00	5
6.00	6
7.00	7
8.00	8
9.00	9

The **Normalizer** view has two panes, **Norm_N** and **Norm_L**.

In each pane, you see

- An input/output table
- A normalizer display
- A breakpoint spacing display

In both **Normalizer** panes, the **Input Output** table and the **Normalizer Display** show the position of the breakpoints.

The **Breakpoint Spacing** display shows a blue slice through the model with the break points overlaid as red lines.

For a more detailed description of the **Normalizer** view, see "Normalizer View" on page 8-14.

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.

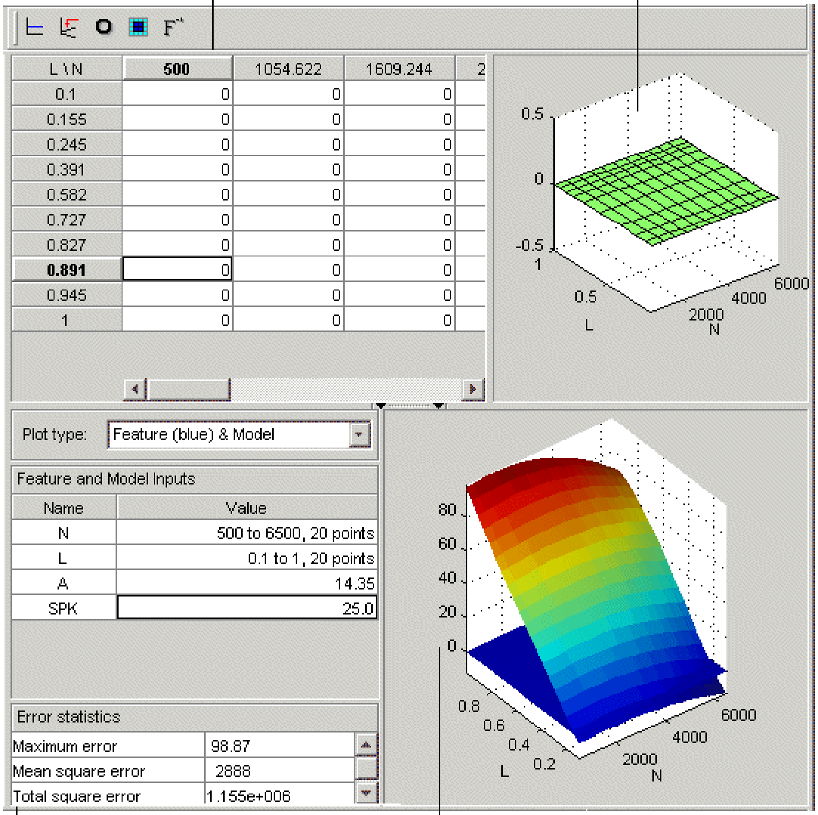
Calibrating the Tables

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.

Lookup table

Graph of table



L \ N	500	1054.622	1609.244	2
0.1	0	0	0	
0.155	0	0	0	
0.245	0	0	0	
0.391	0	0	0	
0.582	0	0	0	
0.727	0	0	0	
0.827	0	0	0	
0.891	0	0	0	
0.945	0	0	0	
1	0	0	0	

Plot type: Feature (blue) & Model

Feature and Model Inputs

Name	Value
N	500 to 6500, 20 points
L	0.1 to 1, 20 points
A	14.35
SPK	25.0

Error statistics

Maximum error	98.87
Mean square error	2888
Total square error	1.155e+006

Error between table and model

Comparison of results

This view has three panes: the table, the graph, and the comparison-of-results pane.

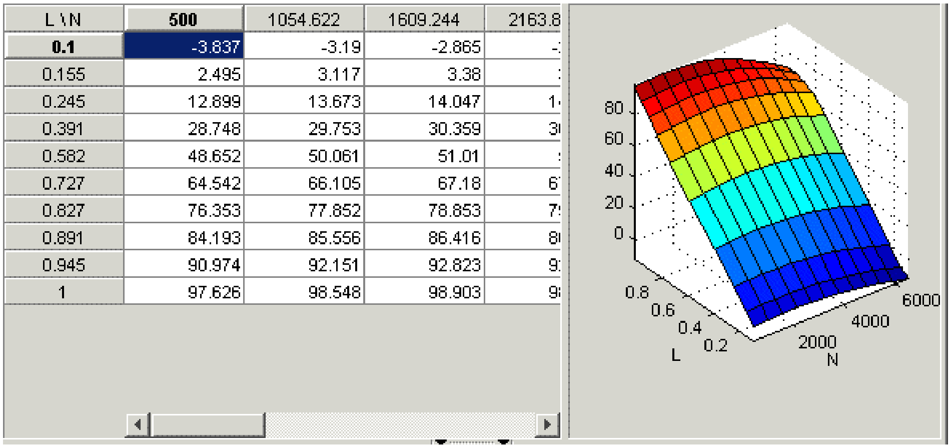
To fill the table with values of the model at the appropriate operating points,

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`.

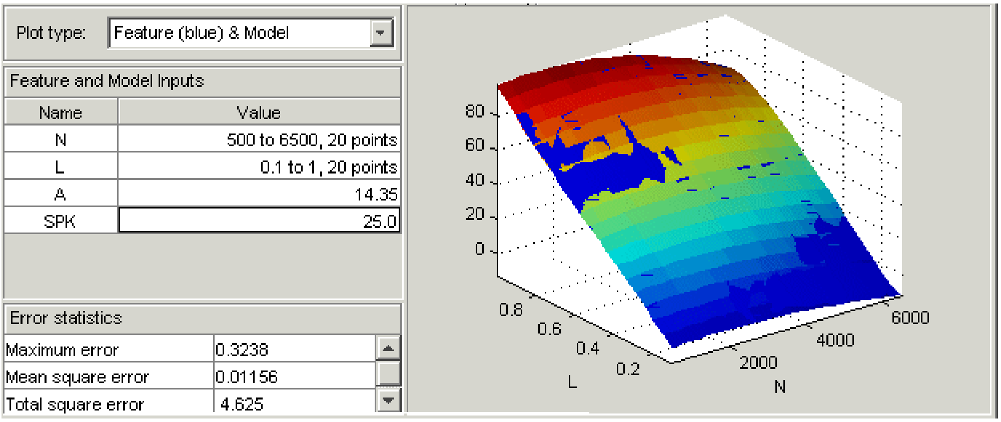
2 Click **OK**.

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



L \ N	500	1054.622	1609.244	2163.8
0.1	-3.837	-3.19	-2.865	-:
0.155	2.495	3.117	3.38	:
0.245	12.899	13.673	14.047	1
0.391	28.748	29.753	30.359	3
0.582	48.652	50.061	51.01	:
0.727	64.542	66.105	67.18	6
0.827	76.353	77.852	78.853	7
0.891	84.193	85.556	86.416	8
0.945	90.974	92.151	92.823	9
1	97.626	98.548	98.903	9

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



Plot type: Feature (blue) & Model

Feature and Model Inputs

Name	Value
N	500 to 6500, 20 points
L	0.1 to 1, 20 points
A	14.35
SPK	25.0

Error statistics

Maximum error	0.3238
Mean square error	0.01156
Total square error	4.625

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 9-12.

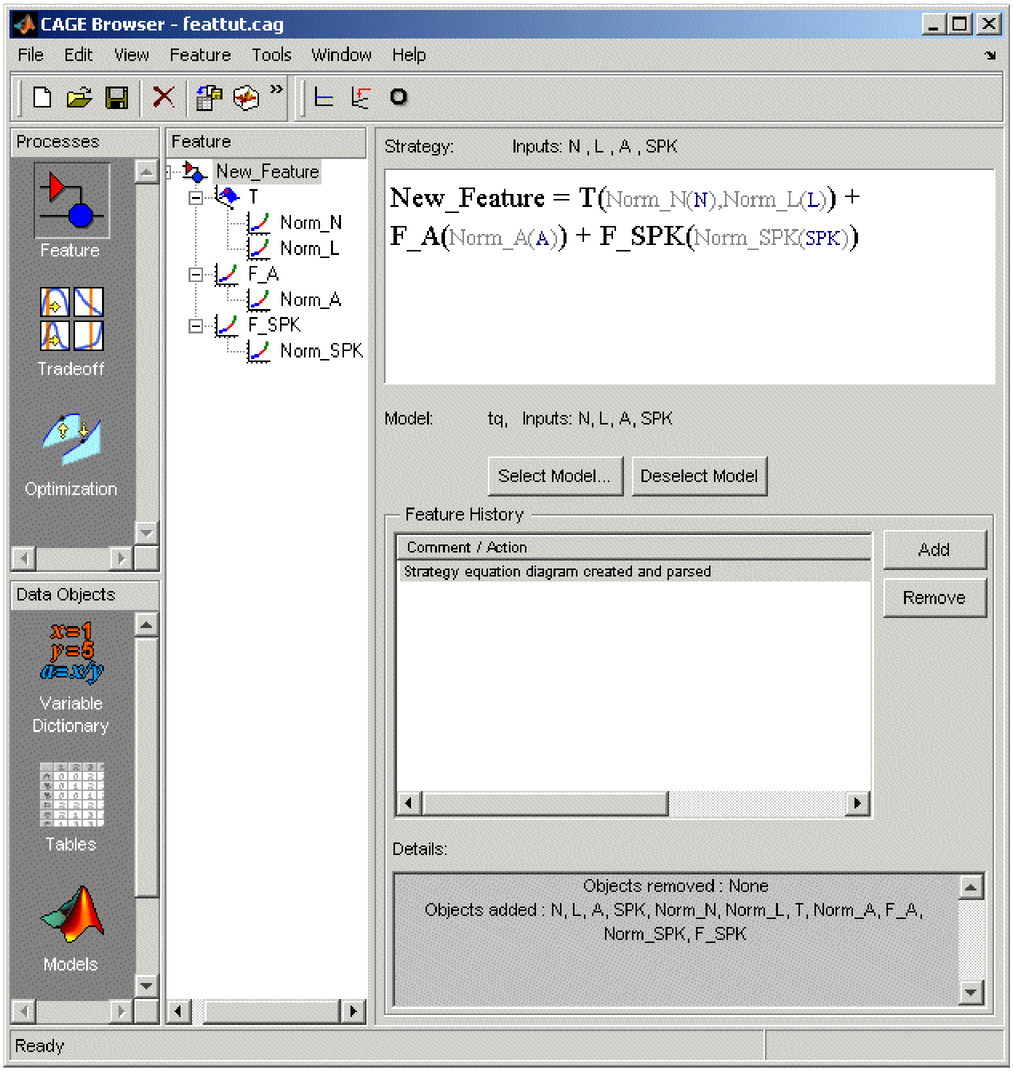
Now you must fill the tables F_A and F_SPK and their normalizers. The tables are modifiers for AFR and the spark angle respectively. These steps are described in the next section.

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.



Strategy: Inputs: N , L , A , SPK

New_Feature = T(Norm_N(N),Norm_L(L)) + F_A(Norm_A(A)) + F_SPK(Norm_SPK(SPK))

Model: tq, Inputs: N, L, A, SPK

Comment / Action
Strategy equation diagram created and parsed

Objects removed : None
Objects added : N, L, A, SPK, Norm_N, Norm_L, T, Norm_A, F_A, Norm_SPK, F_SPK

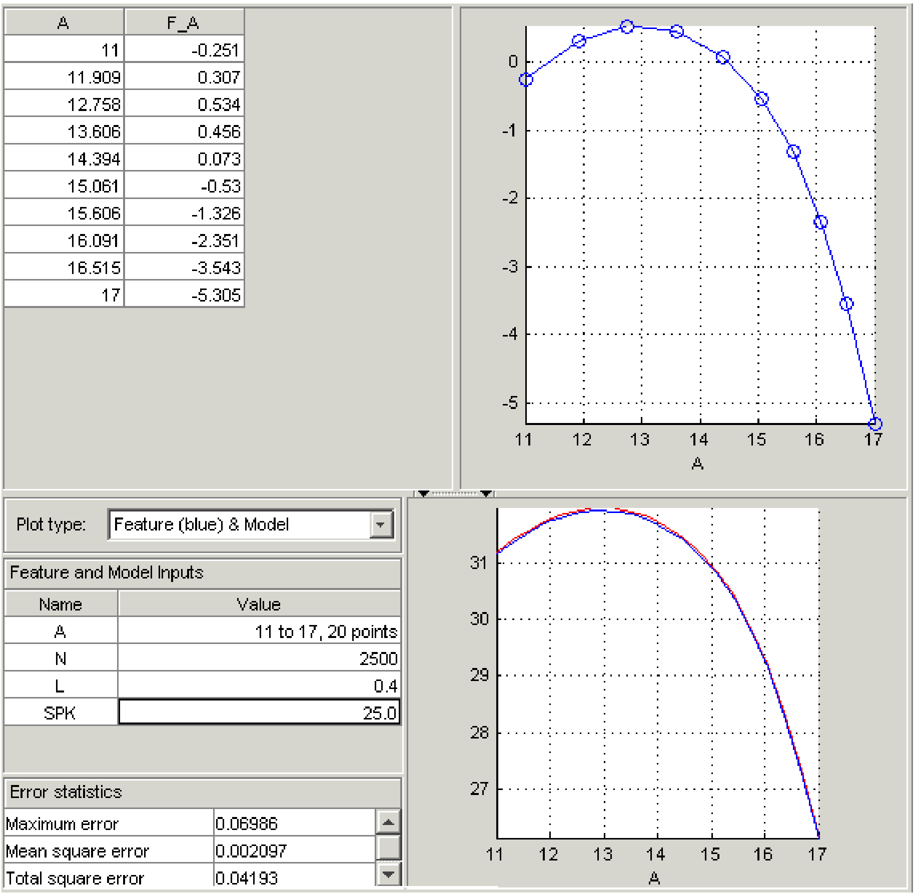
To calibrate all the tables and their normalizers,

1 Select **Feature –> Fill** (or use the **Fill Feature** toolbar button) 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.



A	F_A
11	-0.251
11.909	0.307
12.758	0.534
13.606	0.456
14.394	0.073
15.061	-0.53
15.606	-1.326
16.091	-2.351
16.515	-3.543
17	-5.305

Plot type: Feature (blue) & Model

Feature and Model Inputs

Name	Value
A	11 to 17, 20 points
N	2500
L	0.4
SPK	25.0

Error statistics

Maximum error	0.06986
Mean square error	0.002097
Total square error	0.04193

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 9-39.

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.

3

Tutorial: Tradeoff Calibration

This section includes the following topics:

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**.

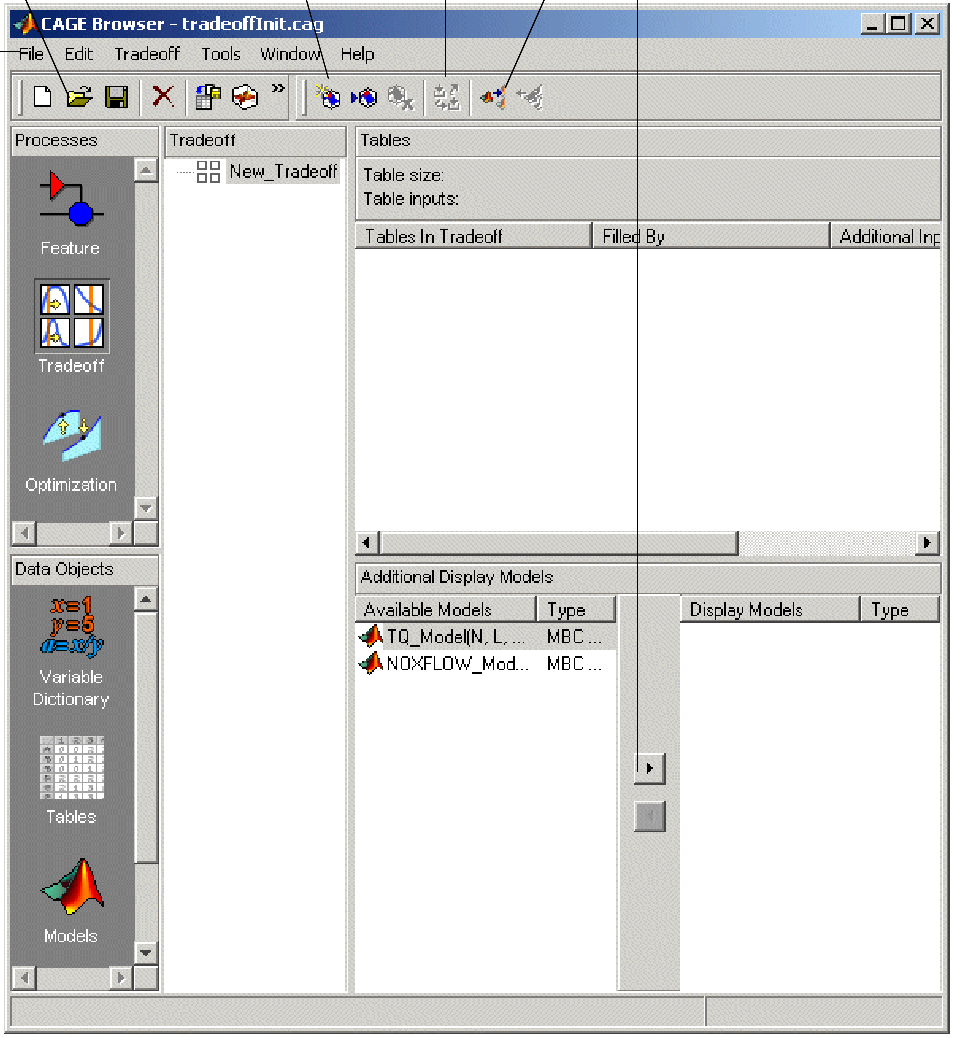
   The `tradeoffInit.cag` project contains two models and all the variables necessary for this tutorial. For information about how to set up models and variables, see “Using CAGE” on page 7-1.

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.



Available Models	Type
TQ_Model(N, L, ...	MBC ...
NOXFLOW_Mod...	MBC ...

1. Open the project file.

2. Add a new tradeoff.

3. Add new table.

4. Select item to fill table.

5. Display the models.

Adding Tables to a Tradeoff Calibration

The models of torque and NOX are in the current session. You must add the lookup table to calibrate.

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

- Exhaust gas recycling (EGR)
- Air/fuel ratio (AFR)
- Spark angle
- Speed
- Load

For this tutorial, you are interested in the spark angle over the range of speed and load.

To generate a lookup table for the spark angle,

1 Click (Add New Table) in the toolbar. This opens the **Table Setup** dialog.

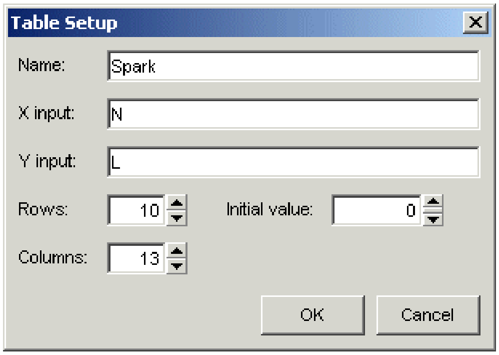


Table Setup

Name: Spark

X input: N

Y input: L

Rows: 10

Initial value: 0

Columns: 13

2 Enter `Spark` as the table **Name**.

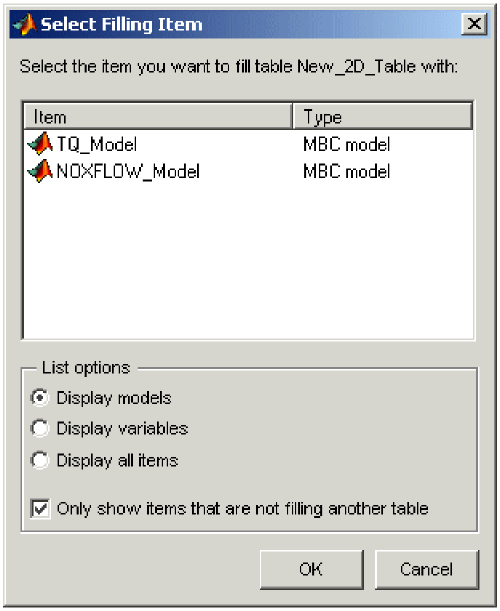
3 Check that `N` is the **X input** and `L` is the **Y input** (these are selected automatically as the first two variables in the current Variable Dictionary).

4 Enter `10` as the size of the load axis (**Rows**).

5 Enter `13` as the size of the speed axis (**Columns**).

6 Click **OK**.

7 Double-click the new Spark table under **Tables in Tradeoff**, or click to select the table and then click Change Filling Item ( ) in the toolbar. A dialog appears where you can select inputs to fill the table.



Select the item you want to fill table New_2D_Table with:

Item	Type
TQ_Model	MBC model
NOXFLOW_Model	MBC model

List options

● Display models

○ Display variables

○ Display all items

☑ Only show items that are not filling another table

8 Select **Display variables**, then select SPK to fill the table and click **OK**.

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

Displaying the Models

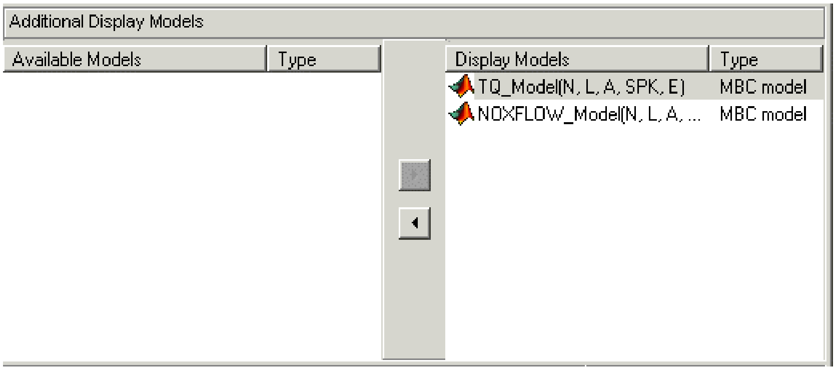
For this tutorial, you are comparing values of the torque and NOX models. Thus, you need to display these models.

To display both models,

- Click Display Model in the toolbar twice. This will move both available models into the Display list.

- Alternatively, click > twice to include both models in the current display. In this case you want to include all available models. You can click to select particular models in the list to display.

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



Display Models	Type
TQ_Model(N, L, A, SPK, E)	MBC model
NOXFLOW_Model(N, L, A, ...	MBC model

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. Check the normalizers.
2. Set values for the other variables, AFR and EGR.
3. Fill key operating points with values for spark angle.
4. Fill the table by extrapolation.

Once you have completed the calibration, you can export the calibration for use in the electronic control unit.

5. Export the calibration.

3. Trade off torque and NOX to find values of spark to fill key operating points in the table.

4. Fill the table by extrapolation.

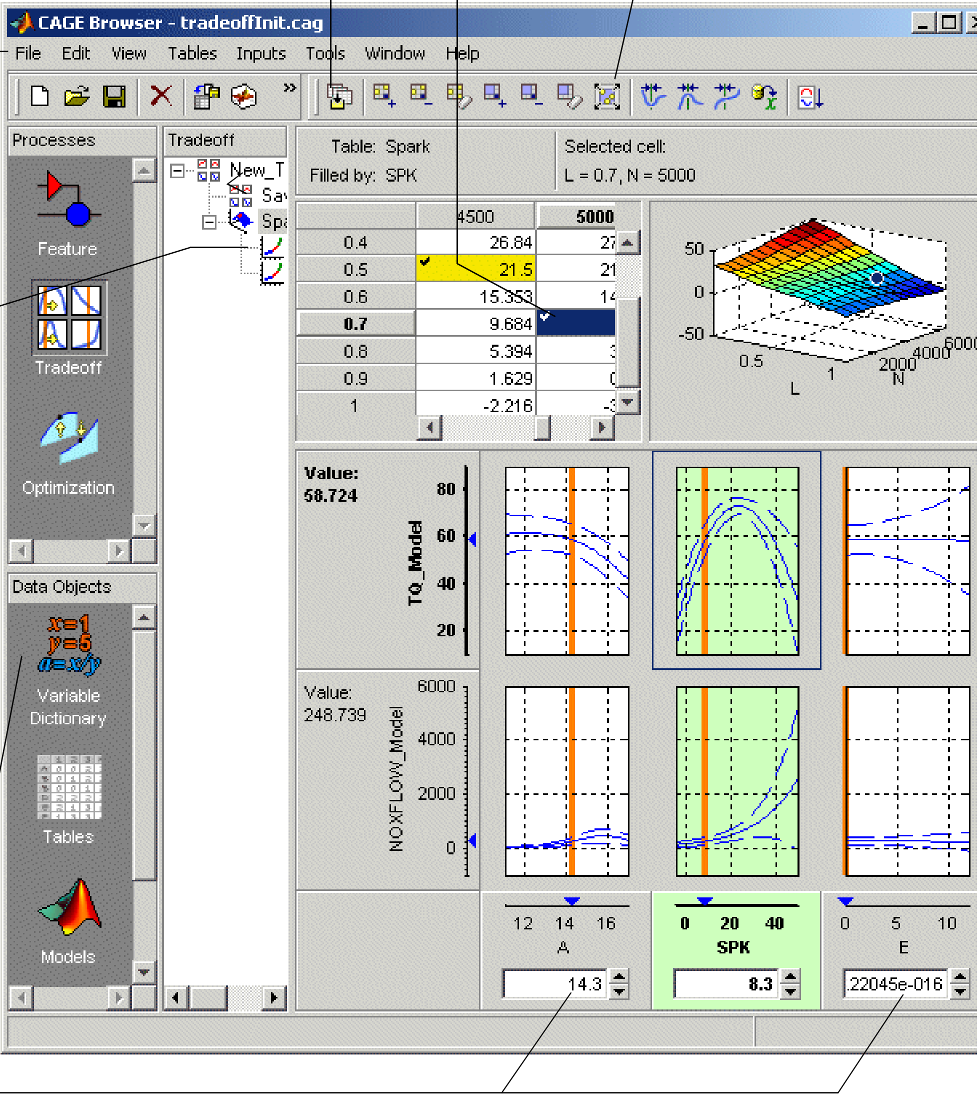


Table: Spark
Filled by: SPK

Selected cell:
L = 0.7, N = 5000

	4500	5000
0.4	26.84	27
0.5	21.5	21
0.6	15.353	14
0.7	9.684	
0.8	5.394	3
0.9	1.629	0
1	-2.216	-3

Value:
58.724

Value:
248.739

14.3

8.3

.22045e-016

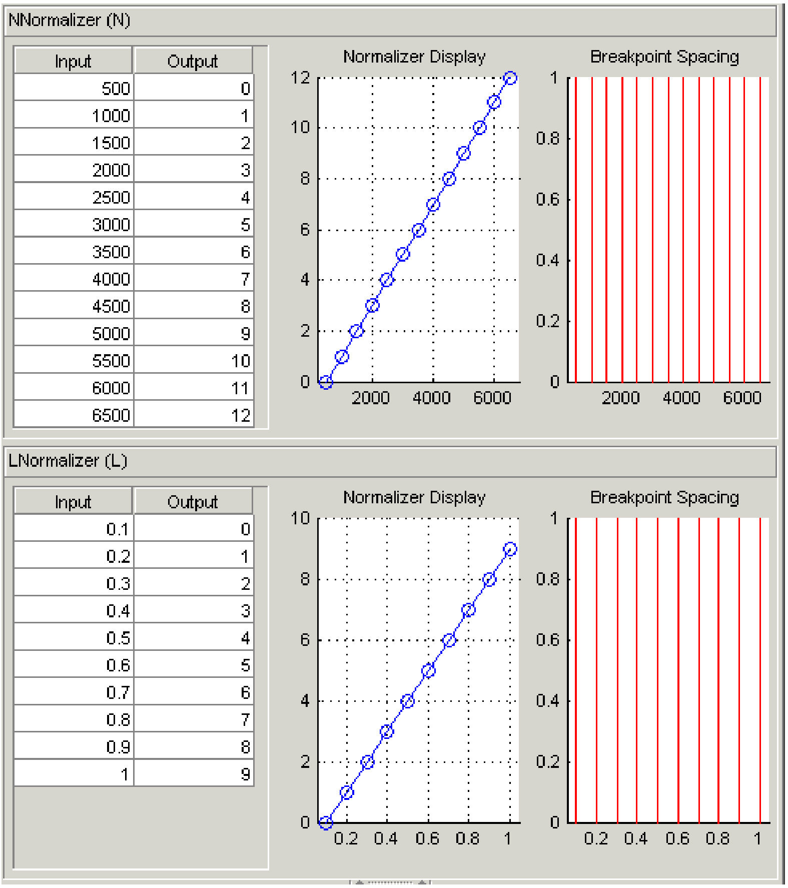
1. Check the normalizers.

2. Set values for the other variables, either individually for each operating point or in the Variable Dictionary.

Checking the Normalizers

A normalizer is the axis of the lookup table (which is the collection of breakpoints). The breakpoints of the normalizers are automatically spaced over the ranges of speed and load. These define the operating points that form the cells of the tradeoff table.

Expand the Tradeoff tree by clicking the plus sign in the display, so you can see the normalizers Speed and Load. Click to highlight either normalizer to see the normalizer view. A tradeoff calibration does not compare the model and the table directly, so you cannot space the breakpoints by reference to the model.



NNormalizer (N)

Input	Output
500	0
1000	1
1500	2
2000	3
2500	4
3000	5
3500	6
4000	7
4500	8
5000	9
5500	10
6000	11
6500	12

LNormalizer (L)

Input	Output
0.1	0
0.2	1
0.3	2
0.4	3
0.5	4
0.6	5
0.7	6
0.8	7
0.9	8
1	9

Setting Values for Other Variables

At each operating point, you must fill the values of the spark table. Both of the models depend on spark, AFR (labeled `A`, in the session), and EGR (labeled `E` in the session). You could set the values for AFR and EGR individually for each operating point in the table, but for simplicity you will set constant values for these model inputs.

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

1. Click **Variable Dictionary** in the **Data Objects** pane.

2. Click `A` and change the set point to `14.3`, the stoichiometric constant.

3. Click `E` and change the set point to `0`.

   You have set these values for every operating point in your tradeoff table. You can now fill the spark angle lookup table. The process is described next.

4. Click **Tradeoff** in the **Processes** pane to return to the tradeoff view.

5. Highlight the `Spark` node in the branch display.

6. In the lower pane, check that the value for `A` is `14.3`, and the value for `E` is `0`, as shown in the following example. You leave these values unchanged for each operating point.

For each operating point you change the values of spark to trade off the torque and NOX objectives; that is, you search for the best value of spark that gives acceptable torque within the emissions constraint. The following example illustrates the controls you use, and there are step-by-step instructions in the following section.

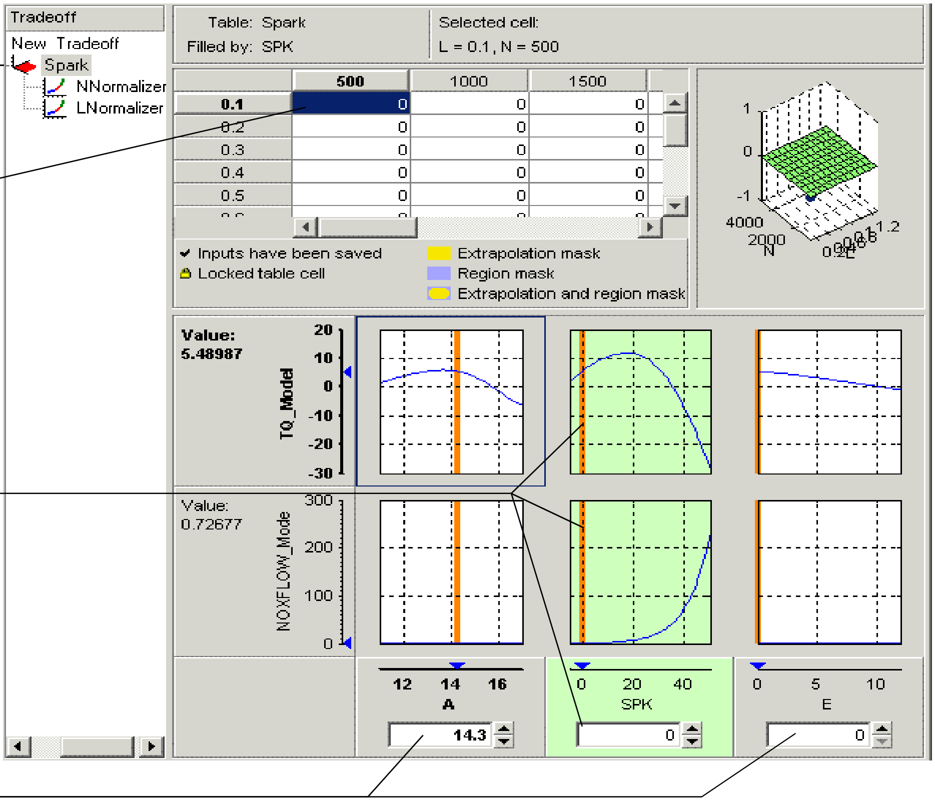


Table: Spark
Filled by: SPK

Selected cell:
L = 0.1, N = 500

	500	1000	1500
0.1	0	0	0
0.2	0	0	0
0.3	0	0	0
0.4	0	0	0
0.5	0	0	0

✓ Inputs have been saved
🔒 Locked table cell
🟨 Extrapolation mask
🟪 Region mask
🟨 Extrapolation and region mask

Value:
5.48987

Value:
0.72677

A: 14.3
SPK: 0
E: 0

1. Highlight the **Spark** node.

2. Select operating points in the **Spark** tradeoff table.

3. Change values of **SPK** to trade off torque and NOX emissions at each operating point.

4. Leave **A** and **E** at the set point values.

Filling Key Operating Points

You now fill the key operating points in the lookup table for spark angle.

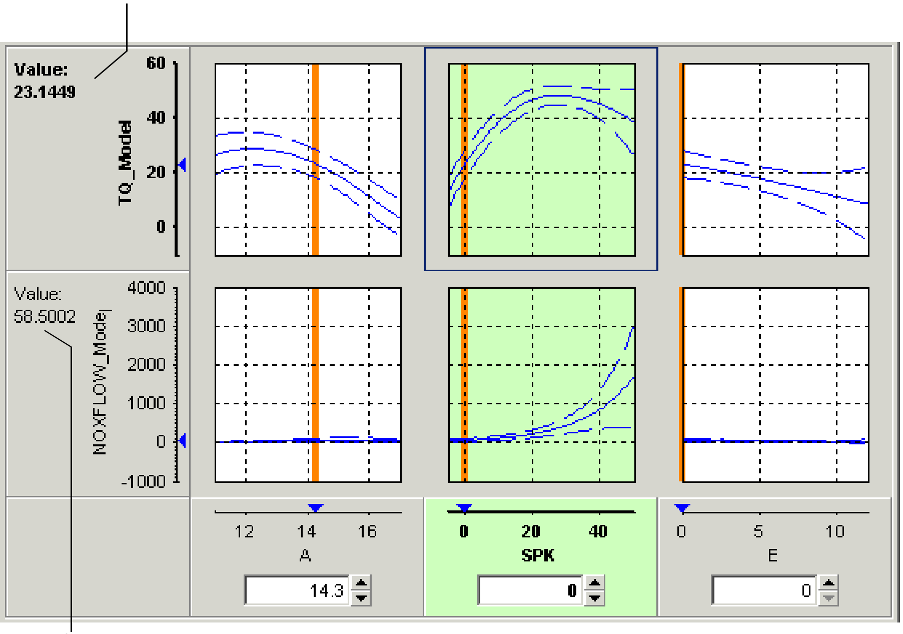
The upper pane displays the lookup table, and the lower pane displays the behavior of the torque and NOX emissions models with each variable.

The object is to maximize the torque and restrict NOX emissions to below 250 g/hr.

Determining the Value of Spark

At each operating point, the behavior of the model alters. The following display shows the behavior of the models over the range of the input variables at the operating point selected in the table, where speed (N) is 4500 and load (L) is 0.5. You can show confidence intervals by selecting **View –> Display Confidence Intervals**.

Value of the torque model



Value of the NOX model

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

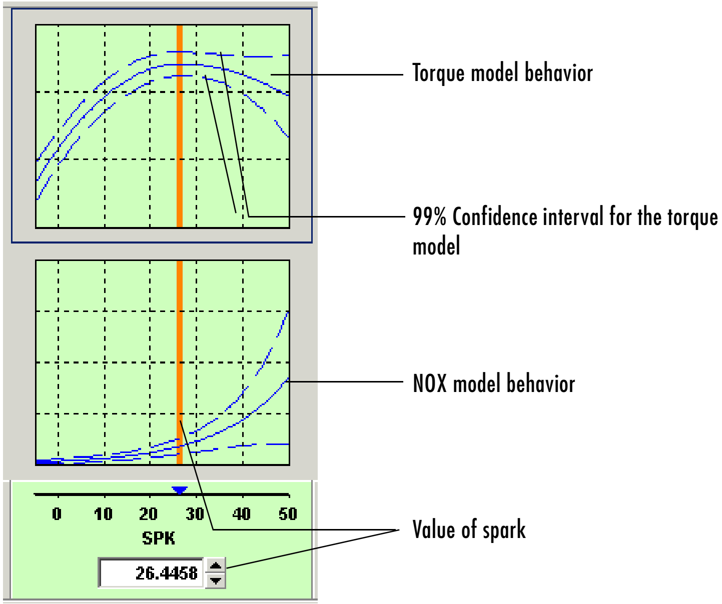
You are calibrating the Spark table, so the two spark (SPK) graphs are green, indicating that these graphs are directly linked to the currently selected lookup table.

1 Select the operating point N = 4500 and L = 0.5 in the lookup table.

2 Now try to find the spark angle that gives the maximum torque and restricts NOX emissions to below 250 g/hr. You can change the value of spark by clicking and dragging the orange line on the SPK graphs, or by typing values into the SPK edit box. You can change the values of any of the other tradeoff variables in the same way, but as you have already set constant values for A and E you should not change these. Try different values of spark and look at the resulting values of the torque and NOX models.

3 Click to select the top **SPK - TQ_Model** graph (when selected the graph is outlined as shown in the following example).

4 Now click 'Find maximum of output' ( ) in the toolbar. This calculates the value of spark that gives the maximum value of torque. The following display shows the behavior of the two models when the spark angle is 26.4458, which gives maximum torque output.



At this operating point, the maximum torque that is generated is 48.136 when the spark angle is 26.4458. However, the value of NOX is 348.968, which is greater than the restriction of 250 g/hr. Clearly you have to look at another value of spark angle.

5 Click and drag the orange bar to change to a lower value of spark. Notice the change in the resulting values of the torque and NOX models.

6 Enter `21.5` as the value of **SPK** in the edit box at the top of the **SPK** column.

The value of the NOX emissions model is now `249.154`. This is within the restriction, and the value of torque is `47.2478`.

At this operating point, this value of `21.5` degrees is acceptable for the spark angle lookup table, so you want to apply this point to your table.

7 Press **Ctrl+T** or click (Apply table filling values) in the toolbar to apply that value to the spark table.

This automatically adds the selected value of spark to the table and turns this cell yellow. It is blue when selected, yellow if you click elsewhere. Look at the table legend to see what this means: yellow cells have been added to the extrapolation mask, and the tick mark indicates you saved this input value by applying it from the tradeoff. You can use the **View** menu to choose whether to display the legend.

8 Now repeat this process of finding acceptable values of spark at four more operating points listed in the table following. In each case,

- Select the cell in the spark table at the specified values of speed and load.
- Enter the value of spark given in the table (the spark angles listed all satisfy the requirements).
- Press **Ctrl+T** or click (Apply table filling values) in the toolbar to apply that value to the spark table.

Speed, N	**Load, L**	**Spark Angle, SPK**
2500	0.3	25.75
3000	0.8	10.7
5000	0.7	8.2
6000	0.2	41.3

After you enter these key operating points, you can fill the table by extrapolation. This is described in the next section.

Filling the Table by Extrapolation

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

When you apply the value of the spark angle to the lookup table, the selected cell is automatically added to the extrapolation mask. This is why the cell is colored yellow. The extrapolation mask is the set of cells that are used as the basis for filling the table by extrapolation.

Click ▨ 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.

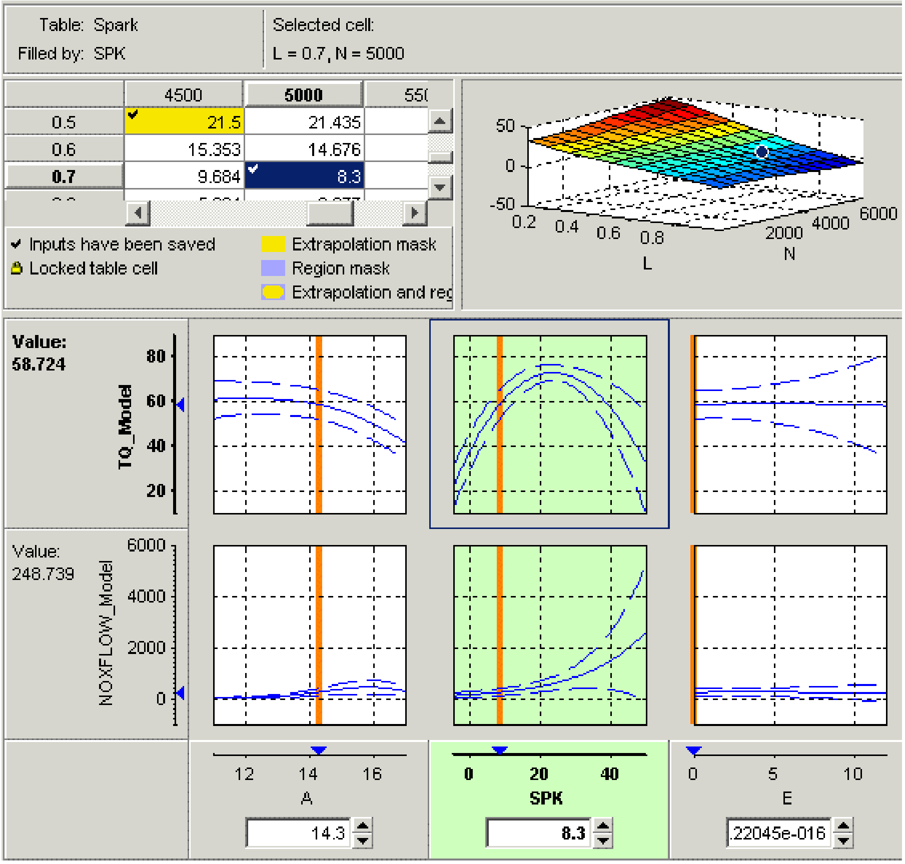


Table: Spark
Filled by: SPK

Selected cell:
L = 0.7, N = 5000

	4500	5000	55(
0.5	✓ 21.5	21.435	
0.6	15.353	14.676	
0.7	9.684	✓ 8.3	

✓ Inputs have been saved
🔒 Locked table cell
🟨 Extrapolation mask
🟦 Region mask
🟨 Extrapolation and reg

Value:
58.724

Value:
248.739

A: 14.3
SPK: 8.3
E: 22045e-016

Note Not all the points in the lookup table will necessarily fulfill the requirements of maximizing torque and restricting the NOX emissions.

You could use these techniques to further improve the calibration and trade off torque and NOX to find the best values for each cell in the spark table.

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

You can now export this calibration to file.

Exporting Calibrations

To export your table and its normalizers,

1 Select the `Spark` node in the branch display.

2 Select **File –> Export –> Calibration**.

3 Choose the file type you want for your calibration. You can choose from

- **Comma Separated Value (.csv)**
- **MAT-file (.mat)**
- **M-file**

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 an ECU.

You have now completed the tradeoff calibration tutorial.

4

Tutorial: Data Sets

This section includes the following topics:

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.

To set up the data set tutorial, you need to

1. Open an existing calibration.
2. Import the experimental data.
3. Add the `Torque` feature to the data set.

Your data set contains all the input factors and output factors required. As the imported data contains various operating points, this information is also included in the data set.

The next sections describe these processes in more detail.

Opening an Existing Calibration

For this tutorial, use the file `datasettut.cag`, found in the `matlab\toolbox\mbc\mbctraining` directory.

To open this file,

1 Select **File –> Open Project**.

2 In the file browser, select `datasettut.cag` and click **Open**.

This opens a file that contains a complete calibrated feature with its associated models and variables. This particular feature is a torque calibration, using a torque table (labeled `T1`) and modifiers for spark (labeled `T2`) and air/fuel ratio (labeled `T3`).

For information about completing a feature calibration, see “Feature 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**.

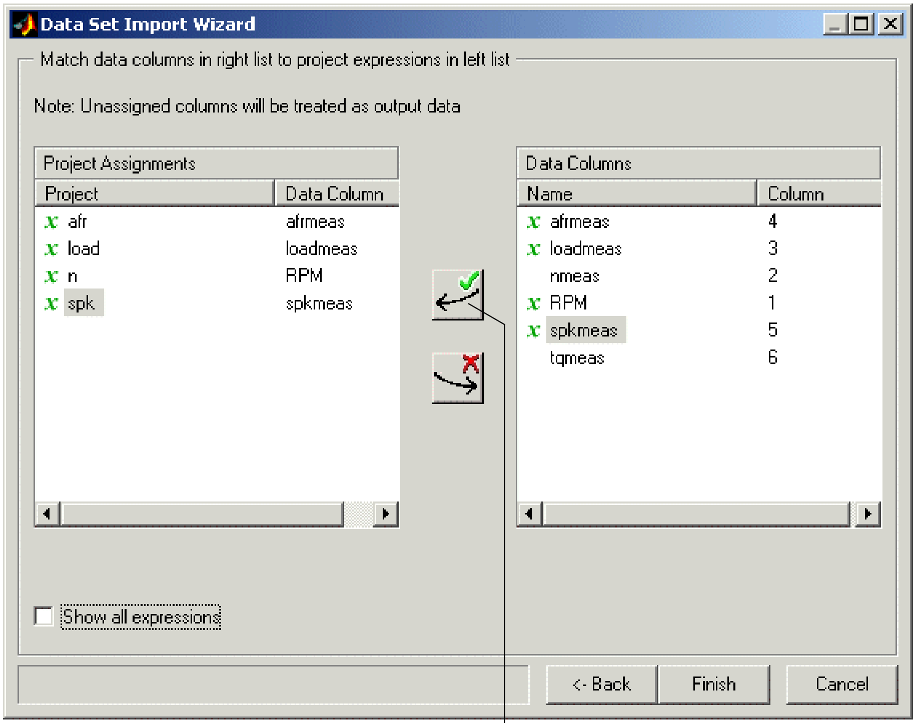
   This set of data includes six columns of data, the test cell settings for engine speed (`RPM`), and the measured values of torque (`tqmeas`), engine speed (`nmeas`), air/fuel ratio (`afrmeas`), spark angle (`spkmeas`), and load (`loadmeas`).

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.



Data Set Import Wizard

Match data columns in right list to project expressions in left list

Note: Unassigned columns will be treated as output data

Project Assignments

Project	Data Column
x afr	afrmeas
x load	loadmeas
x n	RPM
x spk	spkmeas

Data Columns

Name	Column
x afrmeas	4
x loadmeas	3
nmeas	2
x RPM	1
x spkmeas	5
tqmeas	6

□ Show all expressions

Assign button

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

Note If you need to reassign any inputs after closing this dialog you can click ▦ in the toolbar or select **Data –> Assign**.

Adding an Item to a Data Set

To add the `Torque` feature to the data set,

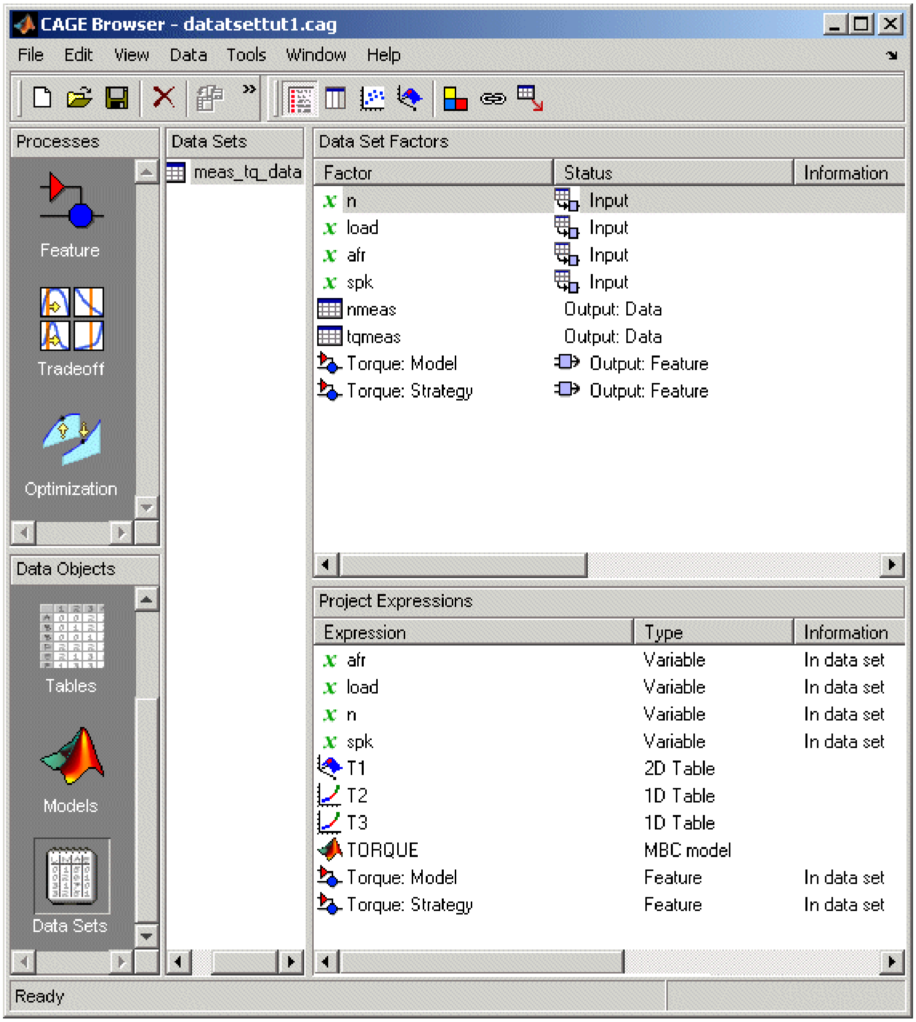
1 Highlight the `Torque` feature in the lower list of **Project Expressions**.

2 Select **Data –> Factors –> Add to Data Set**.

This adds two objects to the data set: `Torque: Model` and `Torque: Strategy`. These two objects make up the `Torque` feature.

- `Torque: Model` is the model used as a reference point to calibrate the feature.
- `Torque: Strategy` is the values of the feature at these operating points.

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



Factor	Status	Information
n	Input	
load	Input	
afr	Input	
spk	Input	
nmeas	Output: Data	
tqmeas	Output: Data	
Torque: Model	Output: Feature	
Torque: Strategy	Output: Feature	

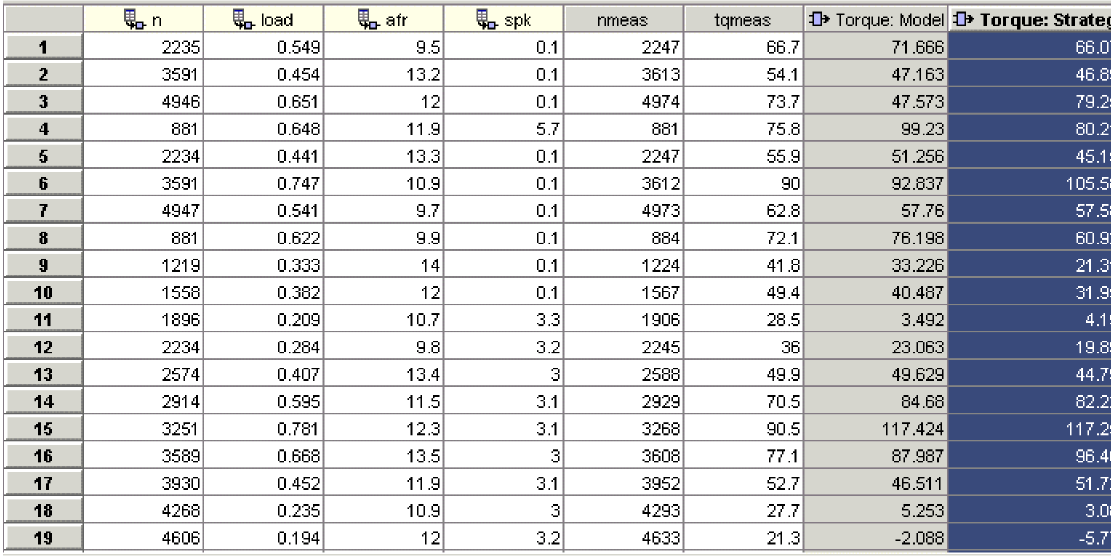
Expression	Type	Information
afr	Variable	In data set
load	Variable	In data set
n	Variable	In data set
spk	Variable	In data set
T1	2D Table	
T2	1D Table	
T3	1D Table	
TORQUE	MBC model	
Torque: Model	Feature	In data set
Torque: Strategy	Feature	In data set

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: Strateg
1	2235	0.549	9.5	0.1	2247	66.7	71.666	66.0
2	3591	0.454	13.2	0.1	3613	54.1	47.163	46.8
3	4946	0.651	12	0.1	4974	73.7	47.573	79.2
4	881	0.648	11.9	5.7	881	75.8	99.23	80.2
5	2234	0.441	13.3	0.1	2247	55.9	51.256	45.1
6	3591	0.747	10.9	0.1	3612	90	92.837	105.5
7	4947	0.541	9.7	0.1	4973	62.8	57.76	57.5
8	881	0.622	9.9	0.1	884	72.1	76.198	60.9
9	1219	0.333	14	0.1	1224	41.8	33.226	21.3
10	1558	0.382	12	0.1	1567	49.4	40.487	31.9
11	1896	0.209	10.7	3.3	1906	28.5	3.492	4.1
12	2234	0.284	9.8	3.2	2245	36	23.063	19.8
13	2574	0.407	13.4	3	2588	49.9	49.629	44.7
14	2914	0.595	11.5	3.1	2929	70.5	84.68	82.2
15	3251	0.781	12.3	3.1	3268	90.5	117.424	117.2
16	3589	0.668	13.5	3	3608	77.1	87.987	96.4
17	3930	0.452	11.9	3.1	3952	52.7	46.511	51.7
18	4268	0.235	10.9	3	4293	27.7	5.253	3.0
19	4606	0.194	12	3.2	4633	21.3	-2.088	-5.7

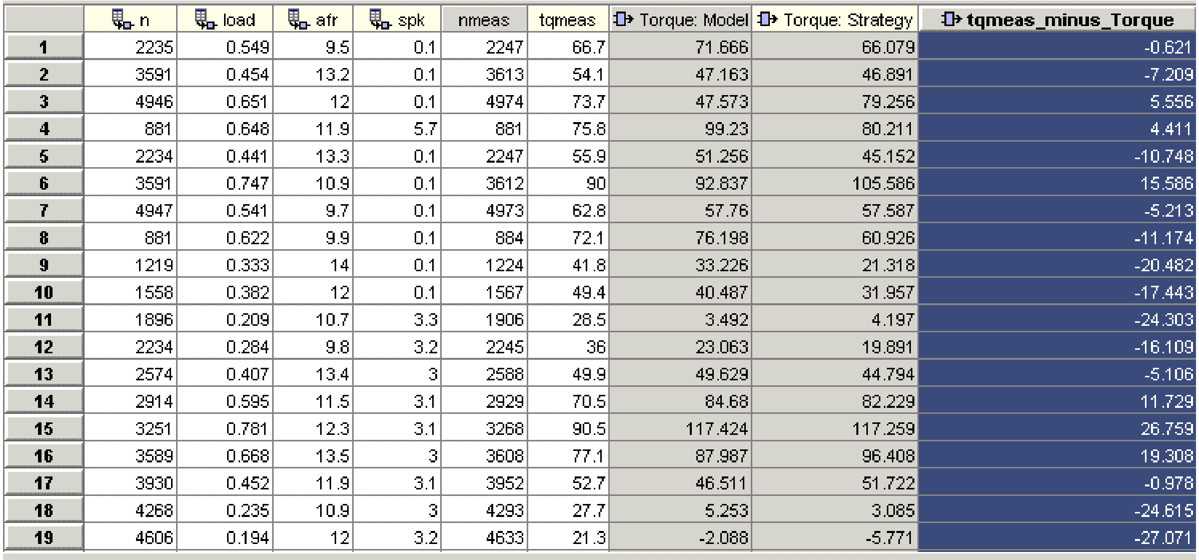
In the table, the input cells are white and the output cells are grey. Select the `Torque: Strategy` column header to see the view shown. The selected column turns blue and the column headers of the strategy's inputs (`n`, `load`, `afr` and `spk`) turn cream. Column headers are always highlighted in this way when they are associated with the currently selected column (such as model inputs, strategy inputs or linked columns).

In addition to viewing the columns, you can use data sets to create a column that shows the difference between two columns:

1 Select the `tqmeas` and `Torque: Strategy` columns by using **Ctrl+click**.

2 Select **Create Error** from the right-click menu on either column header.

This creates another column that is the difference between `tqmeas` and `Torque: Strategy`. Note that all the columns that are inputs to this new column have highlighted headers.



	n	load	afr	spk	nmeas	tqmeas	Torque: Model	Torque: Strategy	tqmeas_minus_Torque
1	2235	0.549	9.5	0.1	2247	66.7	71.666	66.079	-0.621
2	3591	0.454	13.2	0.1	3613	54.1	47.163	46.891	-7.209
3	4946	0.651	12	0.1	4974	73.7	47.573	79.256	5.556
4	881	0.648	11.9	5.7	881	75.8	99.23	80.211	4.411
5	2234	0.441	13.3	0.1	2247	55.9	51.256	45.152	-10.748
6	3591	0.747	10.9	0.1	3612	90	92.837	105.586	15.586
7	4947	0.541	9.7	0.1	4973	62.8	57.76	57.587	-5.213
8	881	0.622	9.9	0.1	884	72.1	76.198	60.926	-11.174
9	1219	0.333	14	0.1	1224	41.8	33.226	21.318	-20.482
10	1558	0.382	12	0.1	1567	49.4	40.487	31.957	-17.443
11	1896	0.209	10.7	3.3	1906	28.5	3.492	4.197	-24.303
12	2234	0.284	9.8	3.2	2245	36	23.063	19.891	-16.109
13	2574	0.407	13.4	3	2588	49.9	49.629	44.794	-5.106
14	2914	0.595	11.5	3.1	2929	70.5	84.68	82.229	11.729
15	3251	0.781	12.3	3.1	3268	90.5	117.424	117.259	26.759
16	3589	0.668	13.5	3	3608	77.1	87.987	96.408	19.308
17	3930	0.452	11.9	3.1	3952	52.7	46.511	51.722	-0.978
18	4268	0.235	10.9	3	4293	27.7	5.253	3.085	-24.615
19	4606	0.194	12	3.2	4633	21.3	-2.088	-5.771	-27.071

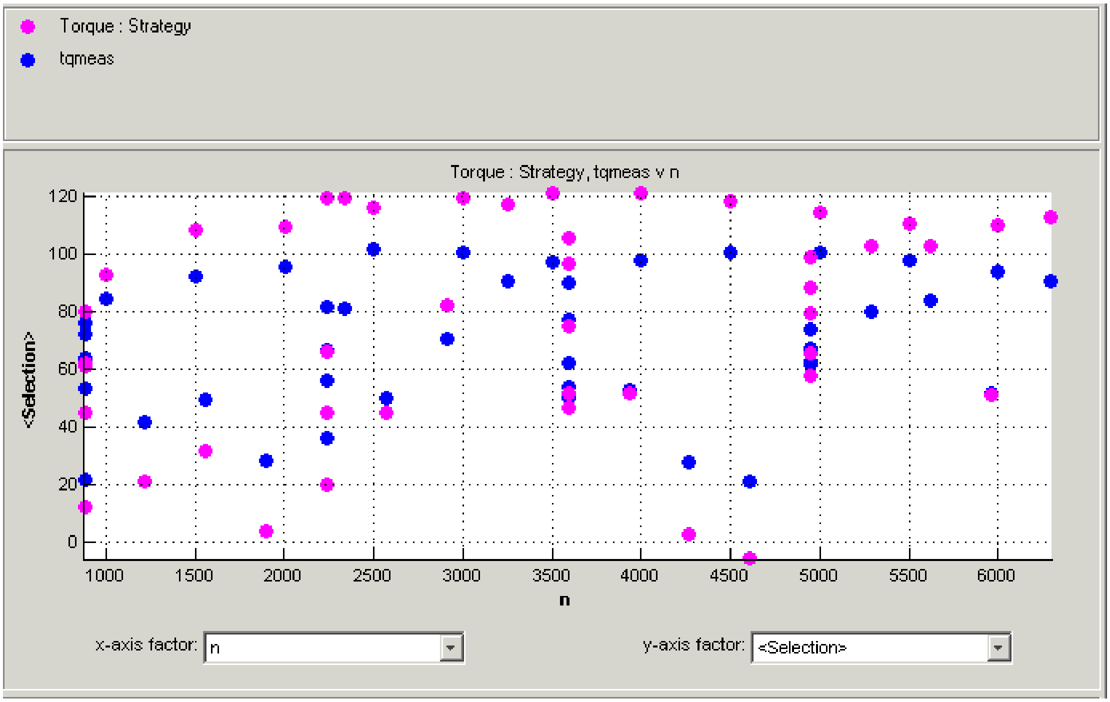
The error column is simply the difference between `tqmeas` and `Torque: Strategy`. This provides a simple way of comparing the feature and the measured data.

Viewing the Data Set as a Plot

1 Click or select **View –> Plot** to view the data set as a plot.

The lower pane lists all the output expressions in the data set and in the project.

2 Use **Ctrl+click** to select `tqmeas` and `Torque: Strategy` from the lower list.



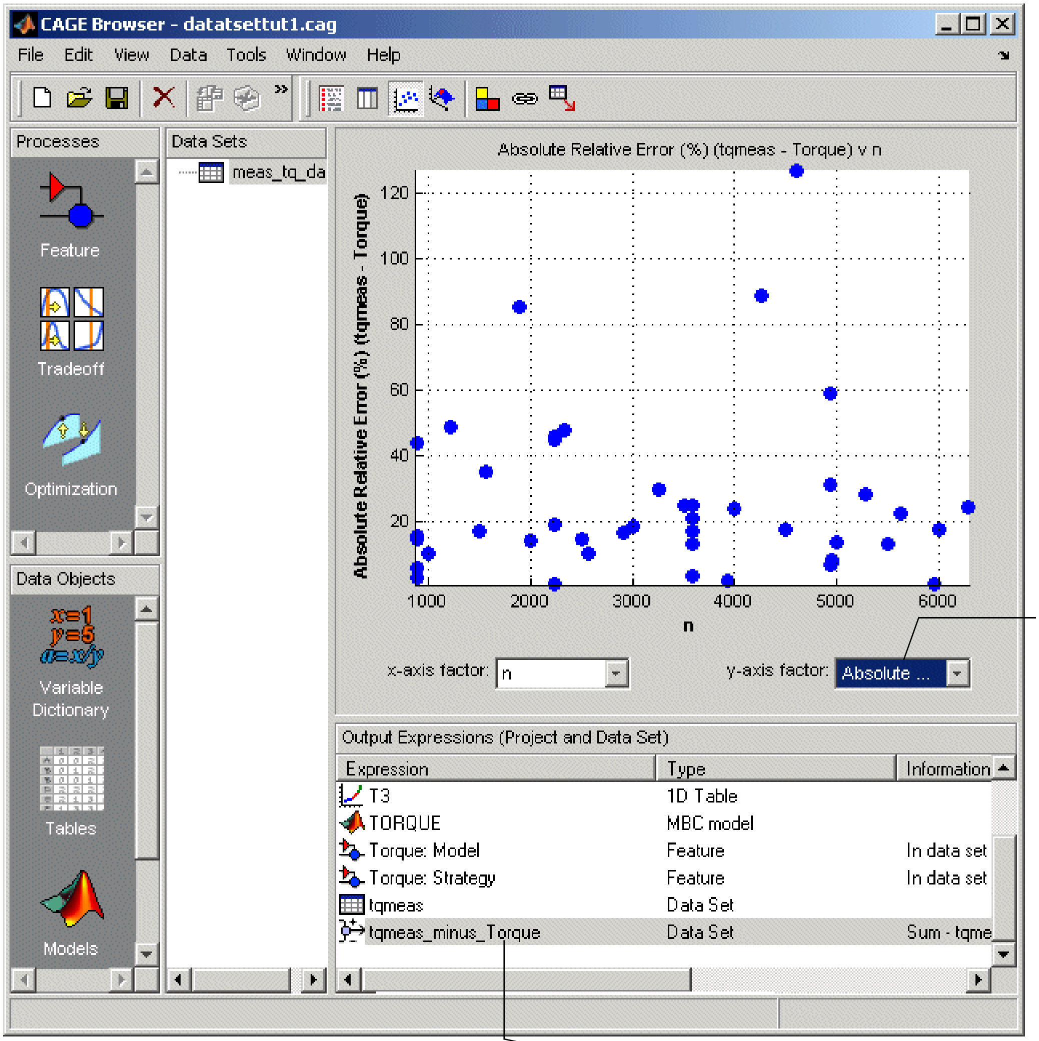
3 Change the ***x*-axis factor** to n from the drop-down menu.

This displays the calibrated values of torque from the feature, and the measured values of torque from the experimental data, against the test cell settings for engine speed.

Clearly there is some discrepancy between the two.

Displaying the Error

View the error between the calibrated and measured values of torque.


CAGE Browser - datasettut1.cag
File Edit View Data Tools Window Help
Processes
Feature
Tradeoff
Optimization
Data Objects
x=1
y=5
a=x/y
Variable Dictionary
Tables
Models
Data Sets
meas_tq_da
Absolute Relative Error (%) (tqmeas - Torque) v n
Absolute Relative Error (%) (tqmeas - Torque)
120
100
80
60
40
20
1000
2000
3000
4000
5000
6000
n
Output Expressions (Project and Data Set)

x-axis factor: n

y-axis factor: Absolute ...

Expression	Type	Information
T3	1D Table	
TORQUE	MBC model	
Torque: Model	Feature	In data set
Torque: Strategy	Feature	In data set
tqmeas	Data Set	
tqmeas_minus_Torque	Data Set	Sum - tqme

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

1. Select tqmeas_minus_Torque.

1 Select `tqmeas_minus_Torque` from the lower list (**Output Expressions**).

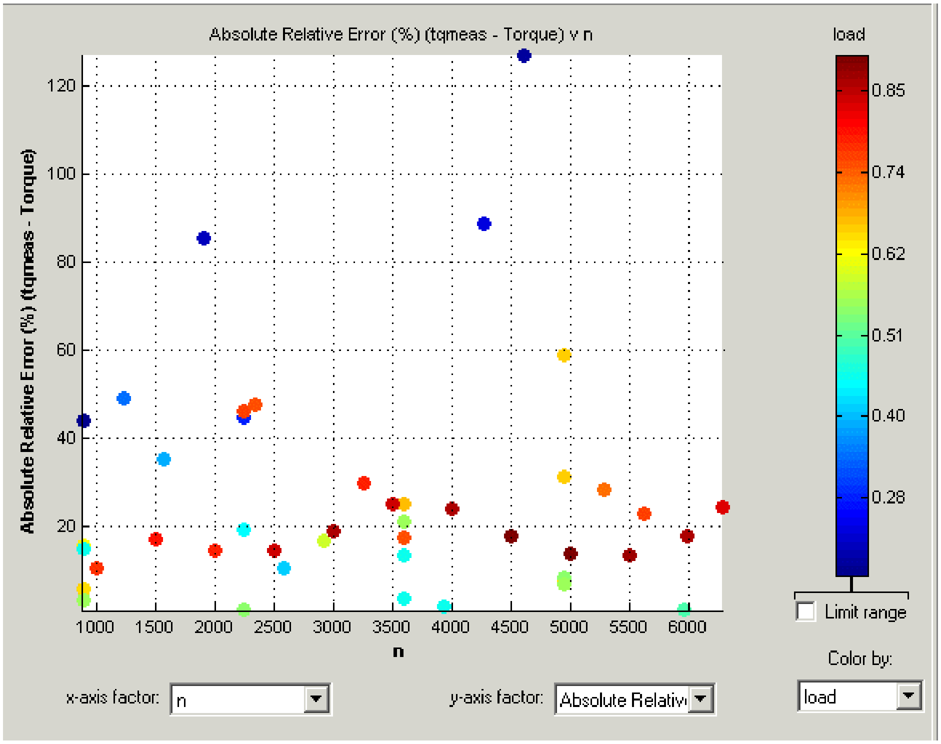
2 For the ***y*-axis factor**, select **Absolute Relative Error (tqmeas - Torque)** from the drop-down menu.

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`.


Absolute Relative Error (%) (tqmeas - Torque) v n
Absolute Relative Error (%) (tqmeas - Torque)
120
100
80
60
40
20
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500
6000
n
load
0.85
0.74
0.62
0.51
0.40
0.28

□ Limit range

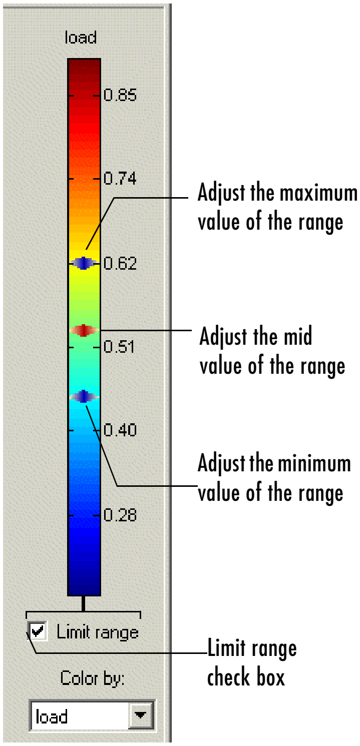
x-axis factor: n

y-axis factor: Absolute Relativ

Color by: 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

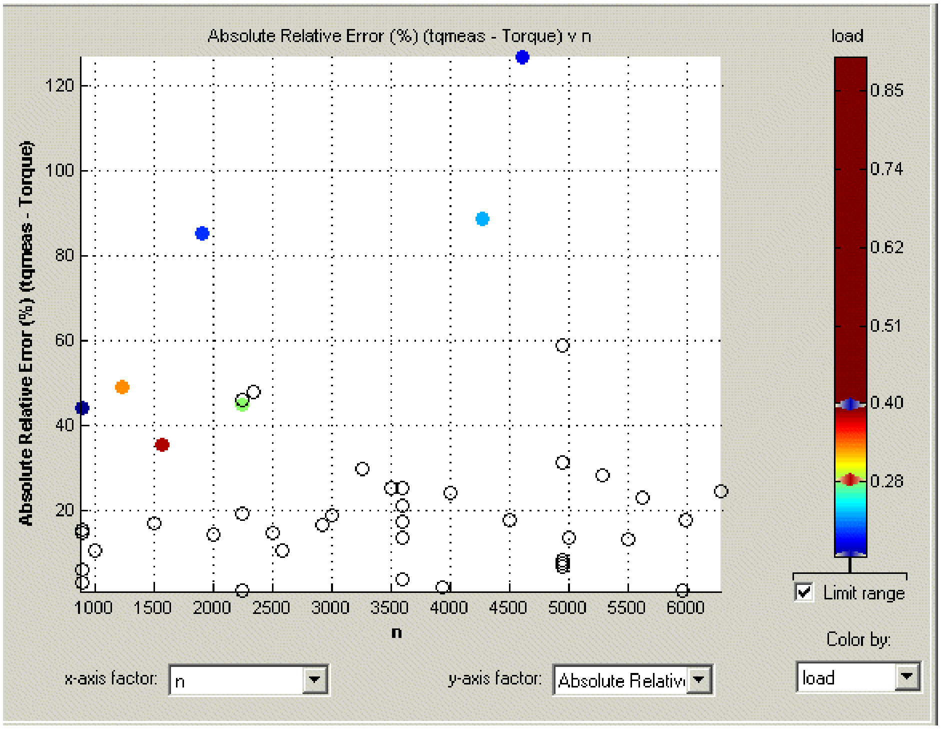

load
0.85
0.74
0.62
0.51
0.40
0.28
☑ Limit range
Color by:
load
Adjust the maximum value of the range
Adjust the mid value of the range
Adjust the minimum value of the range
Limit range check box

To view the colors in more detail, you can limit the range of the colors:

1 Select the **Limit range** box (or you could right-click the graph and select **Restrict Color to Limits**).

2 Set the minimum value of the color range to be as low as possible by dragging the minimum value down.

3 Set the maximum value of the color range to be around 0.4.


Absolute Relative Error (%) (tqmeas - Torque) v n
Absolute Relative Error (%) (tqmeas - Torque)
120
100
80
60
40
20
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500
6000
n
load
0.85
0.74
0.62
0.51
0.40
0.28

☑ Limit range

Color by: load

x-axis factor: n

y-axis factor: Absolute Relativ

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

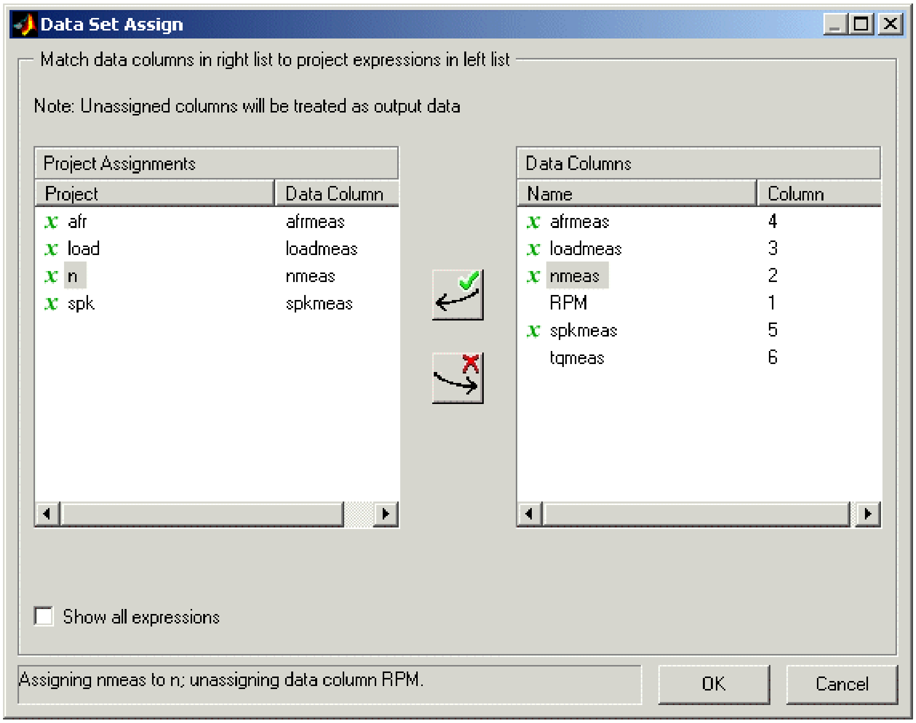
Instead of using the test cell settings for the engine speed (RPM), you might want to use the measured values of engine speed (nmeas). So you have to reassign the variable n to nmeas.

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.



Data Set Assign

Match data columns in right list to project expressions in left list

Note: Unassigned columns will be treated as output data

Project Assignments

Project	Data Column
x afr	afrmeas
x load	loadmeas
x n	nmeas
x spk	spkmeas

Data Columns

Name	Column
x afrmeas	4
x loadmeas	3
x nmeas	2
RPM	1
x spkmeas	5
tqmeas	6

□ Show all expressions

Assigning nmeas to n; unassigning data column RPM.

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.

5

Tutorial: Filling Tables from Data

This section includes the following topics:

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. You can also fill tables with the output of optimizations.

This tutorial takes you through the steps of calibrating a lookup table for torque, based on experimental data.

- This section describes the steps required to set up CAGE in order to calibrate a table by reference to a set of data.
- “Filling the Table from the Experimental Data” on page 5-9 describes the process of filling the lookup table.
- “Selecting Regions of the Data” on page 5-13 describes how you can select some of the data for inclusion when you fill the table.
- “Exporting the Calibration” on page 5-15 describes how to export your completed calibration.

1 Start CAGE by typing

```
cage
```

at the MATLAB prompt.

If you already have a CAGE session open, select **File –> New Project**.

First you will set up a blank table ready for filling using experimental data or optimization output.

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

1. Add the variables for speed and load by importing a variable dictionary.
2. Add a new table to your session.
3. Import your experimental data.

The next sections describe each of these processes in detail.

Adding Variables

Before you can add tables to your session, you must add variables to associate with the normalizers or axes.

To add a variable dictionary,

1. Select **File –> Import –> Variable Dictionary**.
2. Select `table_filling_tutorial.xml` from the `matlab\toolbox\mbc\mbctraining` directory.

This loads a variable dictionary into your session. The variable dictionary includes the following:

- `N`, the engine speed
- `L`, the relative air charge
- `A`, the air/fuel ratio (AFR)
- `stoich`, the stoichiometric constant

You can now add a table to your session.

Adding a New Table

You must add a table to fill.

To add a new table,

1 Select **File –> New –> 2D table**.

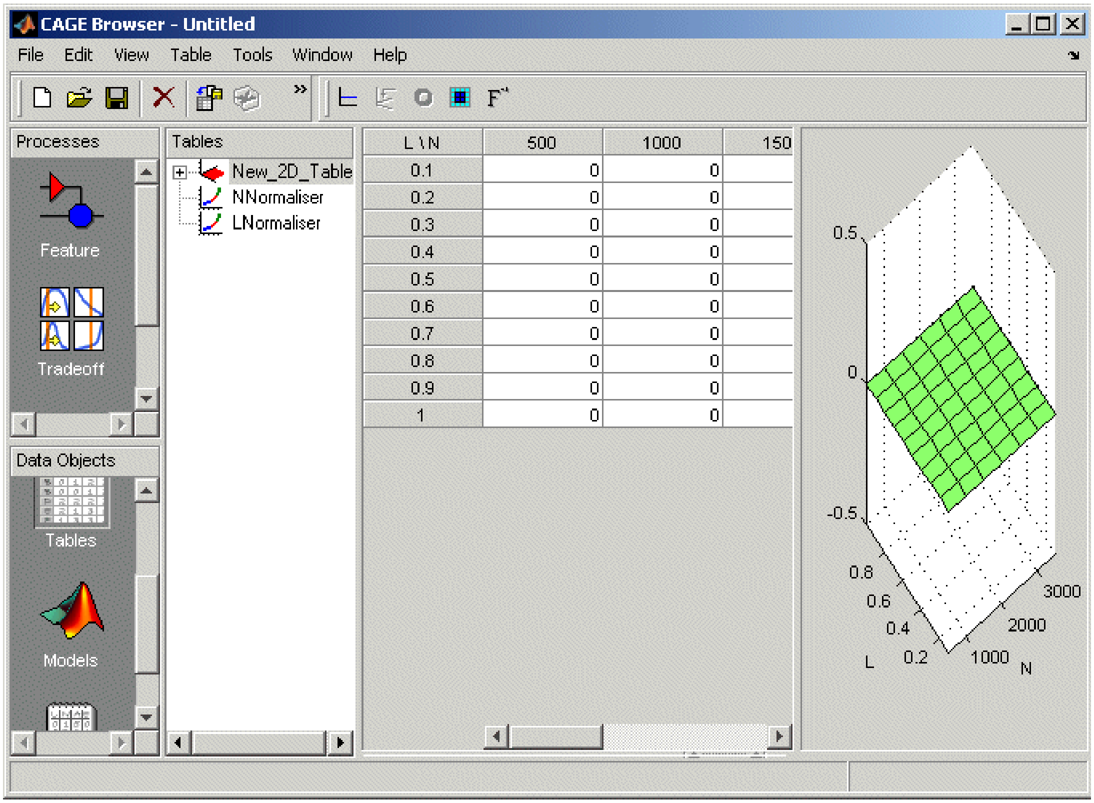
This opens a dialog box that asks you to specify the variable names for the normalizers. As you can see in the dialog controls, accepting the defaults will create a table with ten rows and ten columns with an initial value of `0` in each cell.

2 Change the number of columns to `7`.

3 Select `L` as the variable for normalizer **Y** and `N` as the variable for normalizer **X**, then click **OK**.

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

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

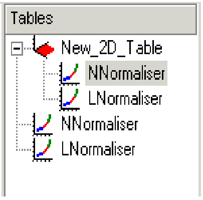


L \ N	500	1000	150
0.1	0	0	
0.2	0	0	
0.3	0	0	
0.4	0	0	
0.5	0	0	
0.6	0	0	
0.7	0	0	
0.8	0	0	
0.9	0	0	
1	0	0	

Inspecting the Values of the Normalizers

CAGE has automatically initialized the normalizers by spacing the breakpoints evenly across the range of values for the engine speed (N) and load (L). The variable ranges are found in the variable dictionary. Switch to the Normalizer view to inspect the normalizers.

4 Expand the table branch by clicking ⊞, and select `NNormalizer` as shown.



This displays the two normalizers for the table.

You have an empty table with breakpoints over the ranges of the engine speed and load, which you can fill with values based on experimental data.

Importing Experimental Data

To fill a table with values based on experimental data, you must add the data to your session. If you want to fill a table with the output of an optimization, the output appears automatically in the Data Sets view as a new data set called `Exported_Optimization_Data` when you select the Export to Data Set toolbar button. For this tutorial you need to import some experimental data.

CAGE uses the **Data Sets** view to store grids of data. Thus, you need to add a data set to your session as well.

Select **File –> New –> Data Set** to add a data set to your session. This changes the view to the **Data Set** view.

You can now import experimental data into the data set:

1 Select **File –> Import –> Data**.

2 In the file browser, select `meas_tq_data.csv` from the `matlab\toolbox\mbc\mbctraining` directory and click **Open**.

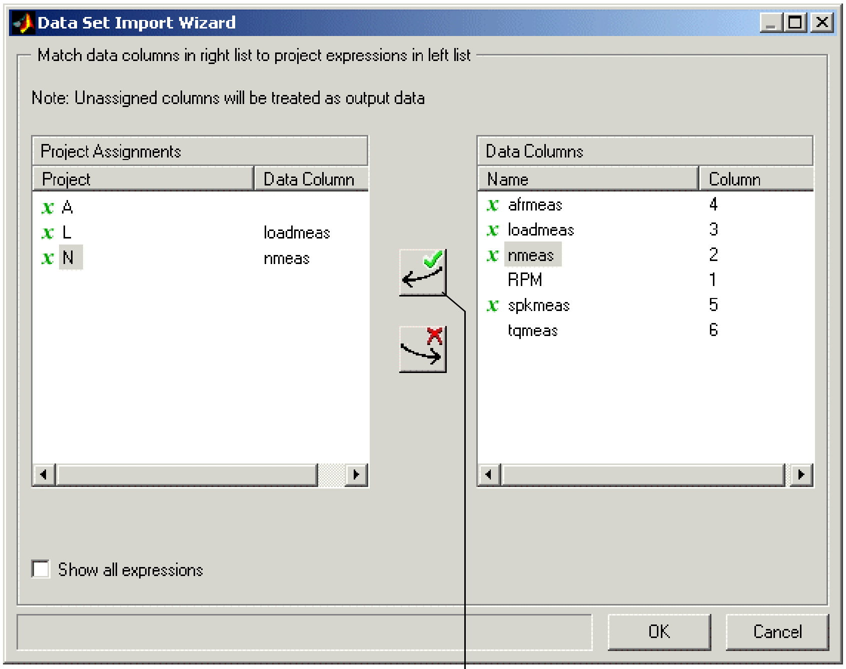
This set of data includes six columns of data: the test cell settings for engine speed (`RPM`), and the measured values of torque (`tqmeas`), engine speed (`nmeas`), air/fuel ratio (`afrmeas`), spark angle (`spkmeas`), and load (`loadmeas`).

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

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

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

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



Data Set Import Wizard

Match data columns in right list to project expressions in left list

Note: Unassigned columns will be treated as output data

Project Assignments

Project	Data Column
x A	
x L	loadmeas
x N	nmeas

Data Columns

Name	Column
x afrmeas	4
x loadmeas	3
x nmeas	2
RPM	1
x spkmeas	5
tqmeas	6

□ Show all expressions

OK Cancel

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,

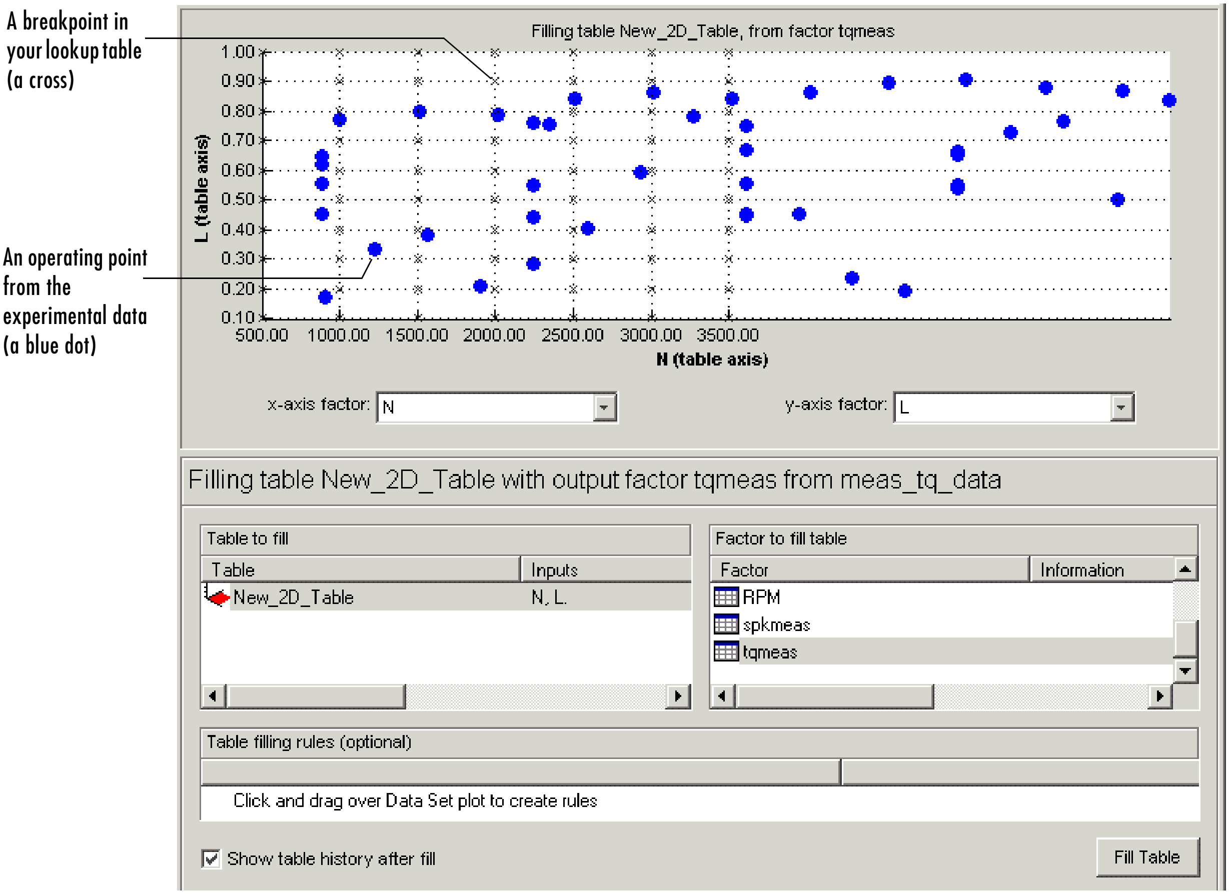
1. To view the **Table Filler** display, click (Fill Table From Data Set) in the toolbar in the **Data Sets** view; or select **View –> Table Filler**.

   You can use this display to specify the table you want to fill and the factor you want to use to fill it.

2. In the lower pane, select `New_2D_Table` from the **Table to fill** list.

3. Select `tqmeas` from the **Factor to fill table** list. This is the data that you want to use to fill the table.

4. Select `N` from the ***x*-axis factor** list and `L` from the ***y*-axis factor** list. Your session should be similar to the following display.



A breakpoint in your lookup table (a cross)

An operating point from the experimental data (a blue dot)

x-axis factor: N

y-axis factor: L

Filling table New_2D_Table with output factor tqmeas from meas_tq_data

Table to fill

Table	Inputs
New_2D_Table	N, L

Factor to fill table

Factor	Information
RPM	
spkmeas	
tqmeas	

Table filling rules (optional)

Click and drag over Data Set plot to create rules

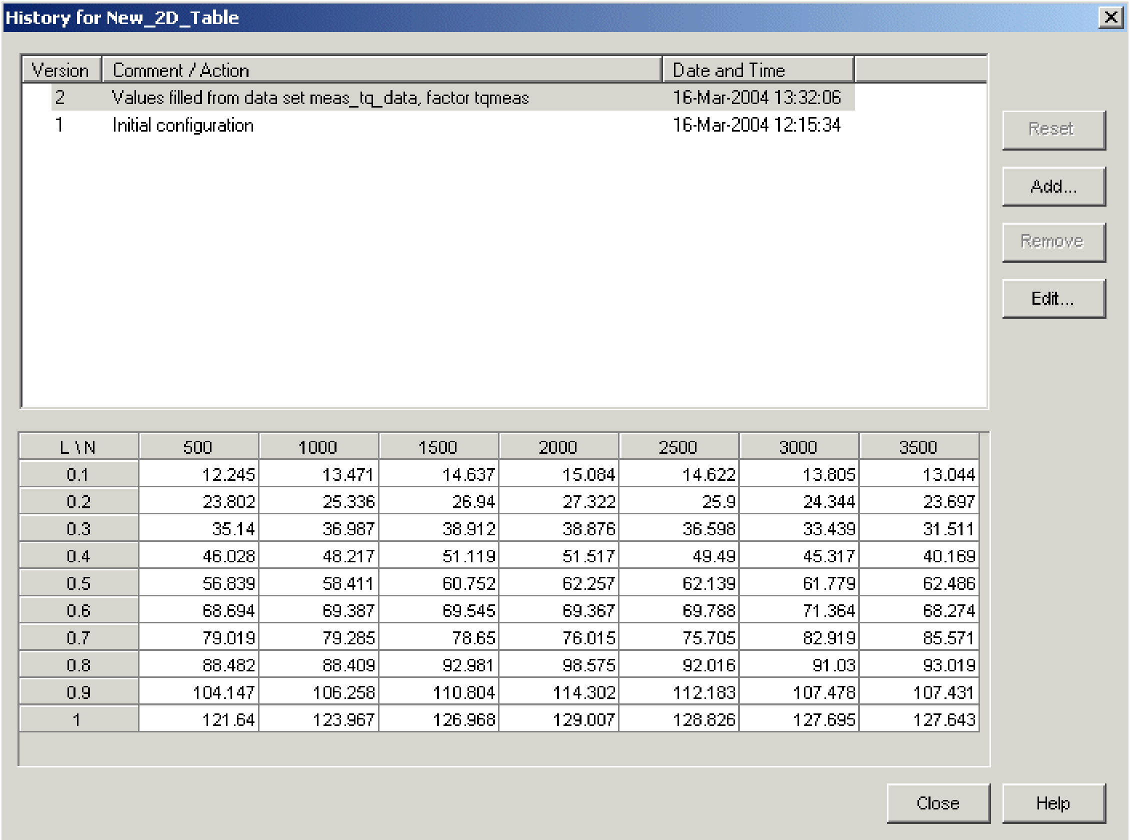
☑ Show table history after fill

The upper pane displays the breakpoints of your table as crosses and the operating points where there is data as blue dots. Data sets display the points in the experimental data, not the values at the breakpoints. You can inspect the spread of the data compared to the breakpoints of your table before you fill the table.

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

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

This opens the **History** dialog box, shown.



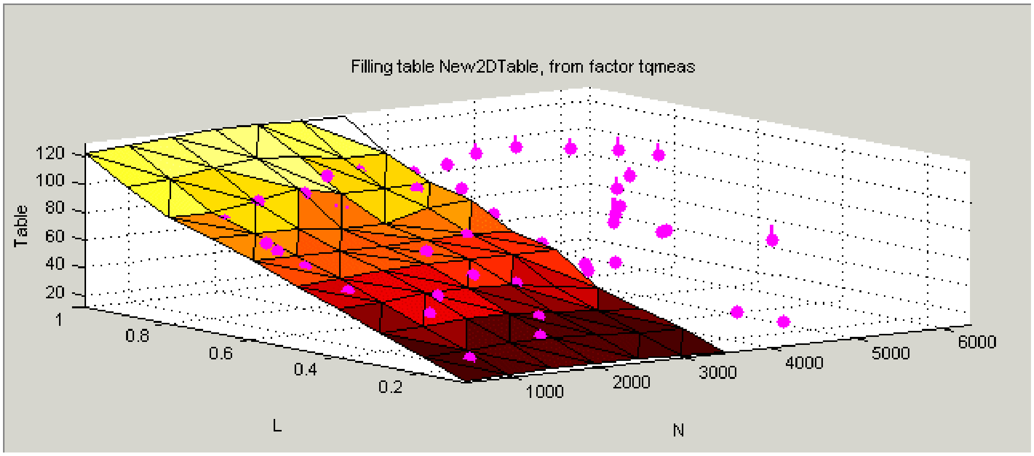
History for New_2D_Table

Version	Comment / Action	Date and Time
2	Values filled from data set meas_tq_data, factor tqmeas	16-Mar-2004 13:32:06
1	Initial configuration	16-Mar-2004 12:15:34

L \ N	500	1000	1500	2000	2500	3000	3500
0.1	12.245	13.471	14.637	15.084	14.622	13.805	13.044
0.2	23.802	25.336	26.94	27.322	25.9	24.344	23.697
0.3	35.14	36.987	38.912	38.876	36.598	33.439	31.511
0.4	46.028	48.217	51.119	51.517	49.49	45.317	40.169
0.5	56.839	58.411	60.752	62.257	62.139	61.779	62.486
0.6	68.694	69.387	69.545	69.367	69.788	71.364	68.274
0.7	79.019	79.285	78.65	76.015	75.705	82.919	85.571
0.8	88.482	88.409	92.981	98.575	92.016	91.03	93.019
0.9	104.147	106.258	110.804	114.302	112.183	107.478	107.431
1	121.64	123.967	126.968	129.007	128.826	127.695	127.643

7 Click **Close** to close the **History** dialog box and return to the **Table Filler** display.

8 To view the graph of your table, as shown, select **Data –> Plot –> Surface**.



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

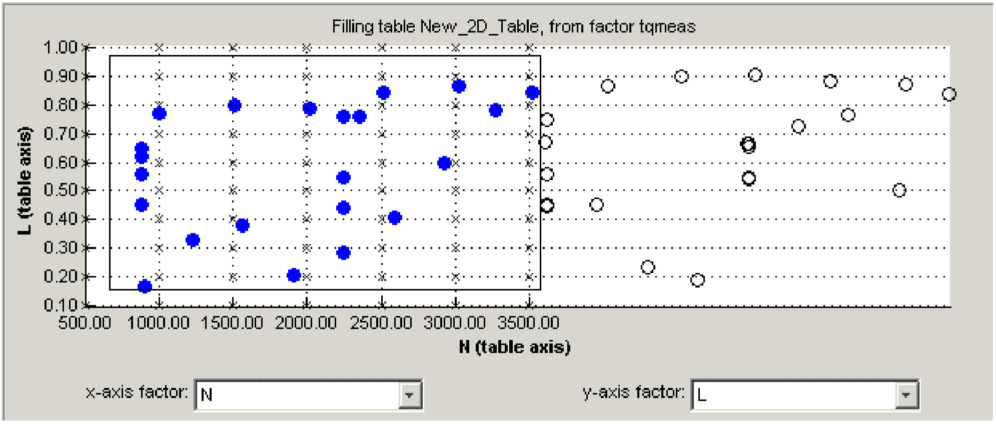
You can ignore points in the data set when you fill your lookup table.

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.

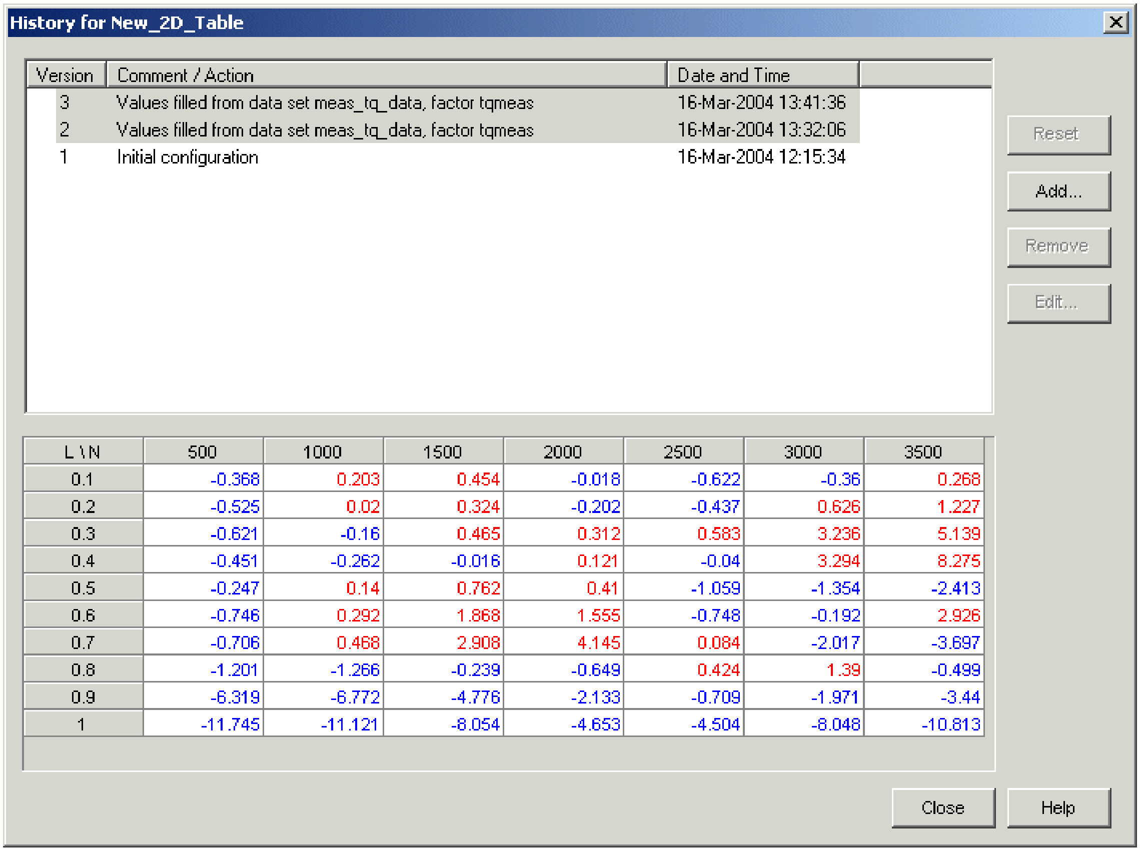


x-axis factor: N

y-axis factor: L

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.



History for New_2D_Table

Version	Comment / Action	Date and Time
3	Values filled from data set meas_tq_data, factor tqmeas	16-Mar-2004 13:41:36
2	Values filled from data set meas_tq_data, factor tqmeas	16-Mar-2004 13:32:06
1	Initial configuration	16-Mar-2004 12:15:34

L \ N	500	1000	1500	2000	2500	3000	3500
0.1	-0.368	0.203	0.454	-0.018	-0.622	-0.36	0.268
0.2	-0.525	0.02	0.324	-0.202	-0.437	0.626	1.227
0.3	-0.621	-0.16	0.465	0.312	0.583	3.236	5.139
0.4	-0.451	-0.262	-0.016	0.121	-0.04	3.294	8.275
0.5	-0.247	0.14	0.762	0.41	-1.059	-1.354	-2.413
0.6	-0.746	0.292	1.868	1.555	-0.748	-0.192	2.926
0.7	-0.706	0.468	2.908	4.145	0.084	-2.017	-3.697
0.8	-1.201	-1.266	-0.239	-0.649	0.424	1.39	-0.499
0.9	-6.319	-6.772	-4.776	-2.133	-0.709	-1.971	-3.44
1	-11.745	-11.121	-8.054	-4.653	-4.504	-8.048	-10.813

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.

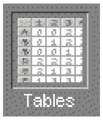
You have filled a lookup table with values taken from experimental data.

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 **Tables**, shown.



2 Highlight the `New_2D_Table`.

3 Select **File –> Export –> Calibration –> Selected Item**.

4 Choose the type of file you want to save your calibrations as. 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.

6

Tutorial: Optimization and Automated Tradeoff

This section includes the following topics:

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 6-19.
- A sum optimization to maximize torque while minimizing NOX, weighted to give more importance to idle speed. See “Sum Optimization” on page 6-33.
- 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 6-37.

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 Optimization button in the **Processes** pane.

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.

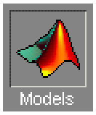
Setting Up Your Optimization Session

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.

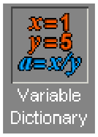
2 Select the **Models** view by clicking the button shown in the **Data Objects** pane.



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.

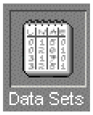
3 To view the items in the **Variable Dictionary,** click the button, shown, in the **Data Objects** pane.



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.

4 Select **File –> New –> Data Set**.

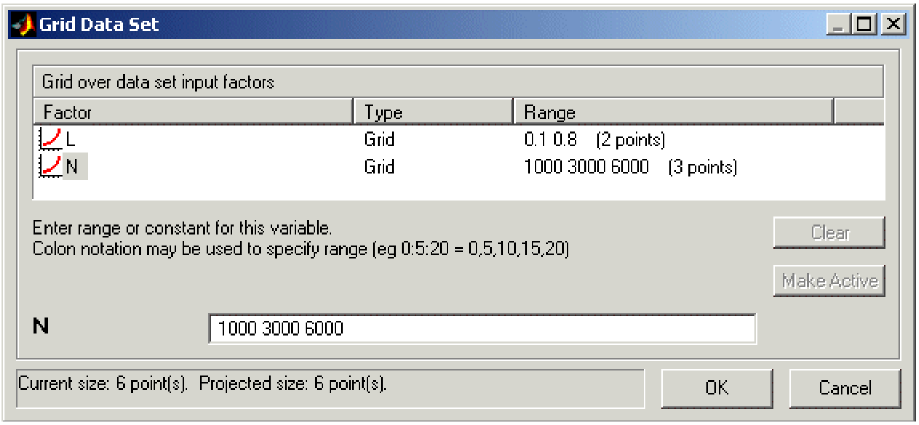
CAGE creates an empty data set. The **Data Sets** view appears. You can switch to this view to see your data sets at any time by clicking the **Data Sets** button, shown, in the **Data Objects** pane.



You need a data set to run your optimization at several points. You can use the data set to define a set of operating points for the optimization. Note that you do not have to have an operating point set; if you do not, the optimization will run at a single point of your choosing (the set points of variables is the default).

5 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**.

6 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.



Grid Data Set

Grid over data set input factors

Factor	Type	Range
L	Grid	0.1 0.8 (2 points)
N	Grid	1000 3000 6000 (3 points)

Enter range or constant for this variable.
Colon notation may be used to specify range (eg 0:5:20 = 0,5,10,15,20)

N: 1000 3000 6000

Current size: 6 point(s). Projected size: 6 point(s).

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

```
0.1 0.8
```

8 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** view to choose

- A model for your objective
- A model, type, and value for your constraint
- An operating point set for your optimization

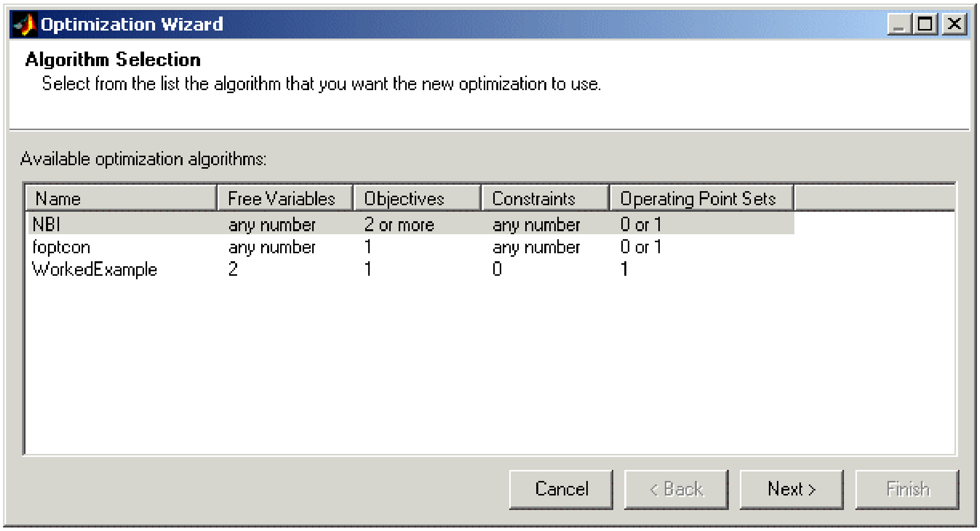
3 Running the optimization, examining the output, exporting to a data set, and using the output to fill a table

Using the Optimization Wizard

To create a new optimization,

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

This opens the **Optimization Wizard**.



Algorithm Selection

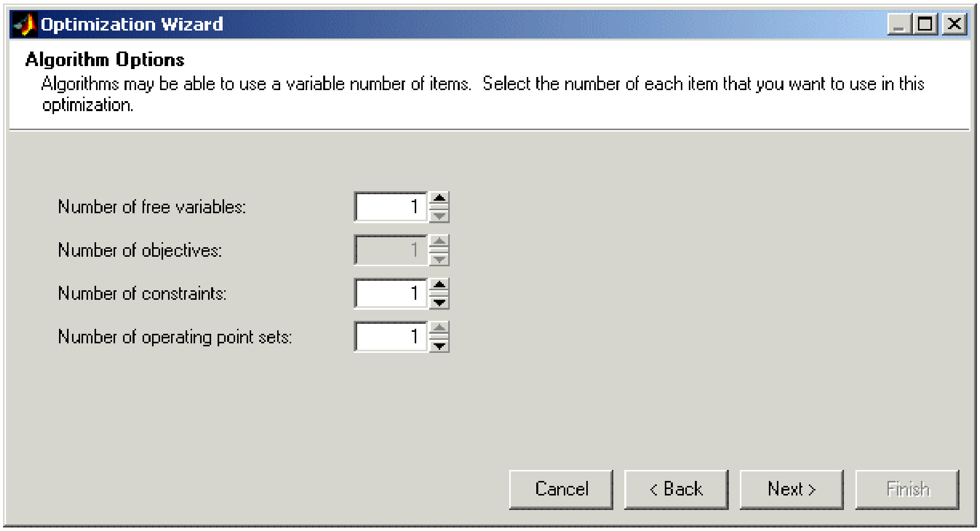
Select from the list the algorithm that you want the new optimization to use.

Available optimization algorithms:

Name	Free Variables	Objectives	Constraints	Operating Point Sets
NBI	any number	2 or more	any number	0 or 1
foptcon	any number	1	any number	0 or 1
WorkedExample	2	1	0	1

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.



Optimization Wizard

Algorithm Options

Algorithms may be able to use a variable number of items. Select the number of each item that you want to use in this optimization.

Number of free variables: 1

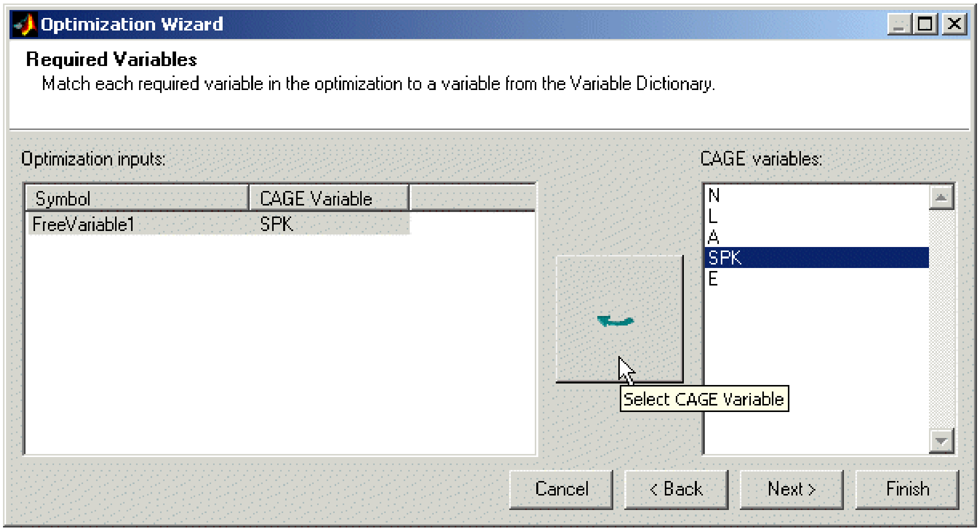
Number of objectives: 1

Number of constraints: 1

Number of operating point sets: 1

Leave the number of free variables at 1 (spark will be the free variable). 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**.



Optimization Wizard

Required Variables

Match each required variable in the optimization to a variable from the Variable Dictionary.

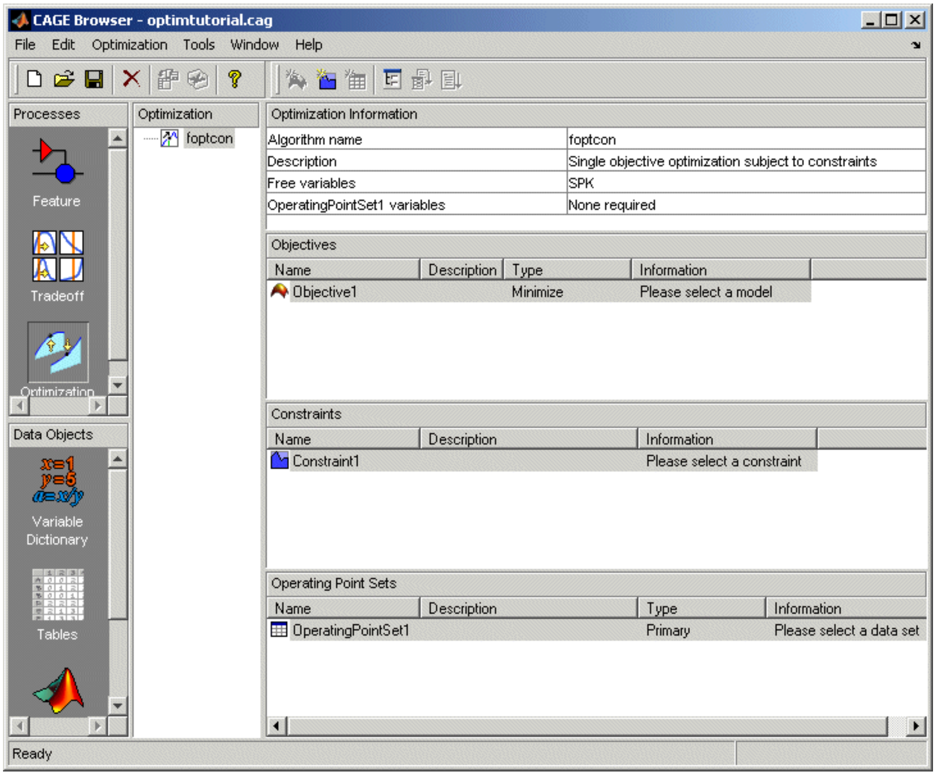
Optimization inputs:

Symbol	CAGE Variable
FreeVariable1	SPK

CAGE variables:

N
L
A
SPK
E

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.



Optimization Information

Algorithm name	foptcon
Description	Single objective optimization subject to constraints
Free variables	SPK
OperatingPointSet1 variables	None required

Objectives

Name	Description	Type	Information
Objective1		Minimize	Please select a model

Constraints

Name	Description	Information
Constraint1		Please select a constraint

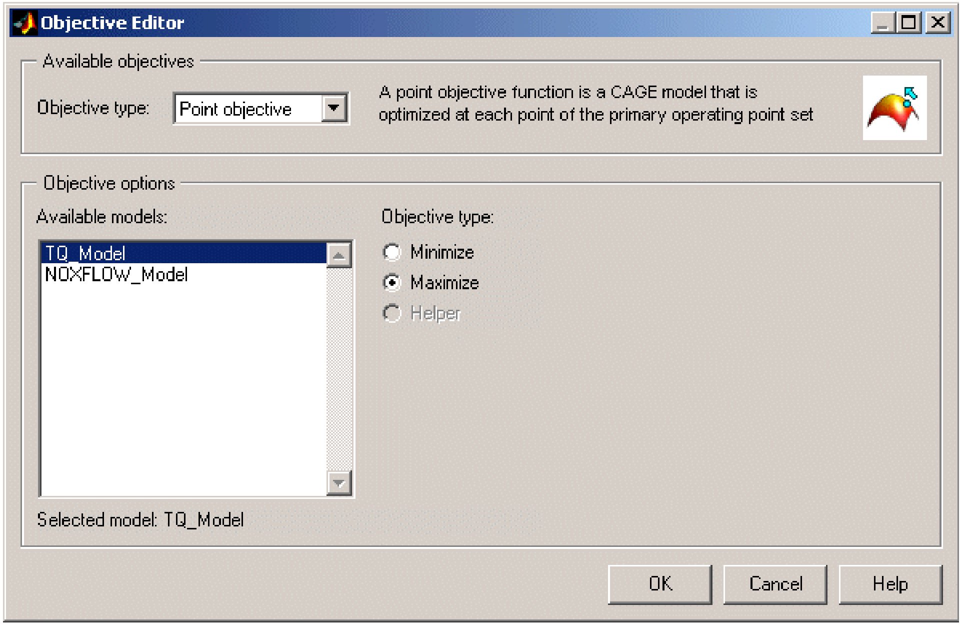
Operating Point Sets

Name	Description	Type	Information
OperatingPointSet1		Primary	Please select a data set

Setting Objectives, Constraints and Operating Point Sets

1 Double-click `Objective1` in the **Objectives** pane.

The **Objective Editor** appears.



Objective Editor

Available objectives

Objective type: Point objective

A point objective function is a CAGE model that is optimized at each point of the primary operating point set

Objective options

Available models:

TQ_Model
NOXFLOW_Model

Objective type:

○ Minimize
● Maximize
○ Helper

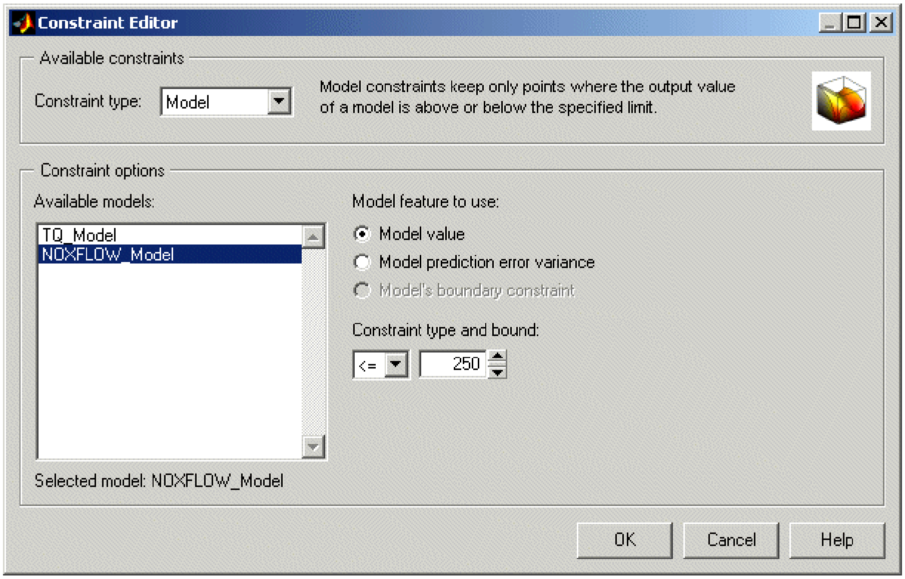
Selected model: TQ_Model

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 view. 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.



Constraint type: Model

Model constraints keep only points where the output value of a model is above or below the specified limit.

Available models:

TQ_Model
NOXFLOW_Model

Model feature to use:

● Model value
○ Model prediction error variance
○ Model's boundary constraint

Constraint type and bound: <= 250

Selected model: NOXFLOW_Model

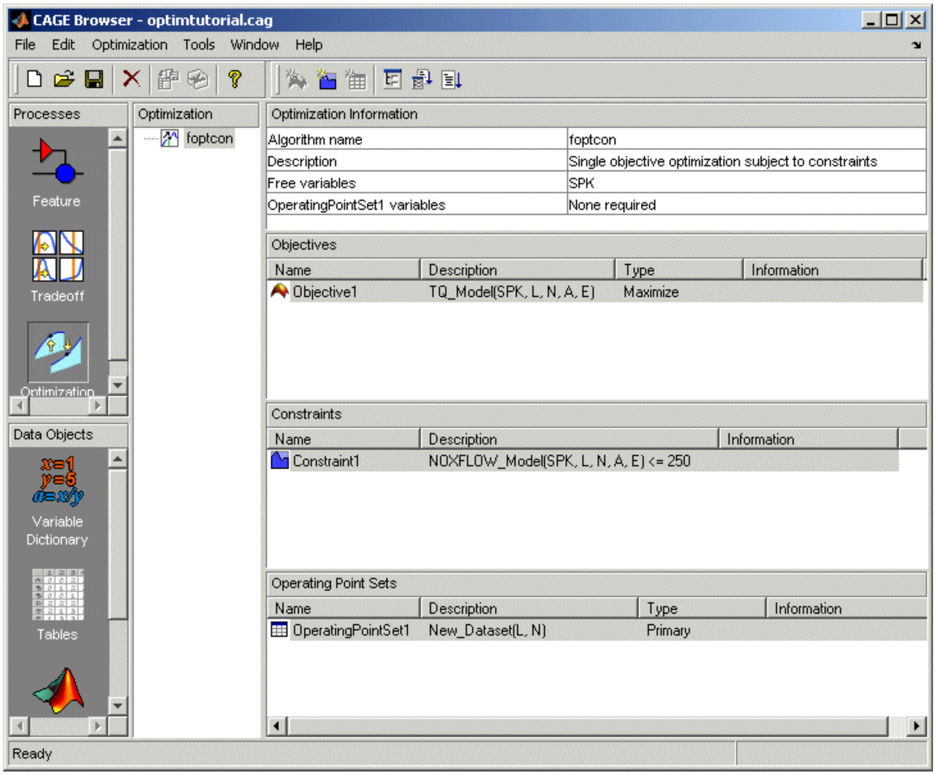
4 Ensure that `Model` is selected from the **Constraint Type** drop-down menu.

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 view. 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.



Optimization Information

Algorithm name	foptcon
Description	Single objective optimization subject to constraints
Free variables	SPK
OperatingPointSet1 variables	None required

Objectives

Name	Description	Type	Information
Objective1	TQ_Model(SPK, L, N, A, E)	Maximize	

Constraints

Name	Description	Information
Constraint1	NOXFLOW_Model(SPK, L, N, A, E) <= 250	

Operating Point Sets

Name	Description	Type	Information
OperatingPointSet1	New_Dataset(L, N)	Primary	

Running the Optimization

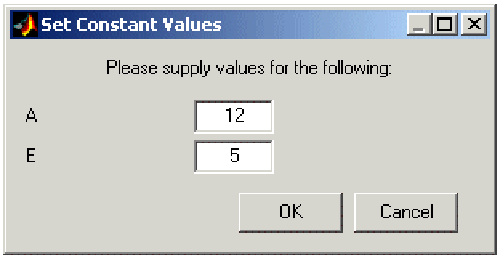
1 Click Run Optimization ( )in the toolbar.

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.

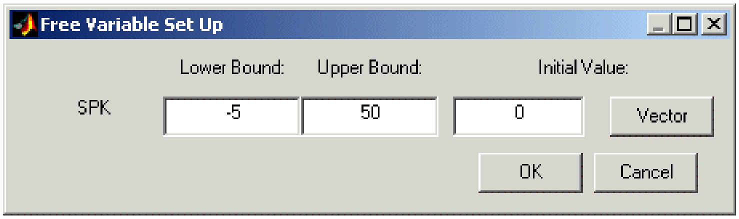
2 A dialog appears called **Set Constant Values**, asking you to supply values for the fixed variables in the model not specified by your operating point set (A and E). The defaults are taken from the set points for these variables in the **Variable Dictionary**. Click **OK** to accept these values.



Please supply values for the following:

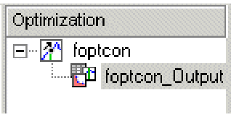
A 12
E 5

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.

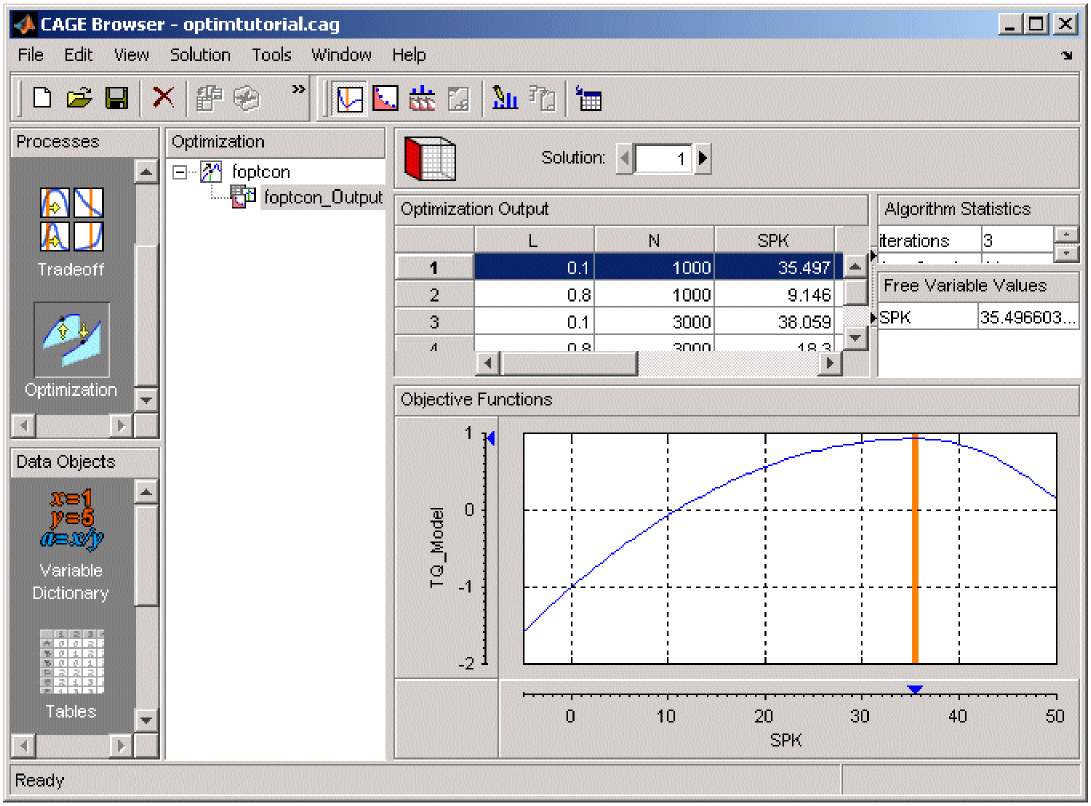


	Lower Bound:	Upper Bound:	Initial Value:
SPK	-5	50	0

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.



Solution: 1

Optimization Output

	L	N	SPK
1	0.1	1000	35.497
2	0.8	1000	9.146
3	0.1	3000	38.059
4	0.8	3000	18.3

Algorithm Statistics

iterations	3

Free Variable Values

SPK	35.496603...

Ready

Using Optimization Results to Fill Tables

As an example, to use these optimization results to fill a table, first create a new table as follows:

1 Build a `SPK` table in `N` and `L`. Select **File –> New–> 2-D Table**.

2 Leave `10` in the **Rows** and **Columns** edit boxes and `0` in the **Value** edit box.

3 Use the drop-down menus to select `L` and `N` for the Y and X inputs.

4 Click **OK**. Your CAGE browser shows the **Tables** view. CAGE has automatically initialized the normalizers to space breakpoints evenly over the ranges of `N` and `L`.

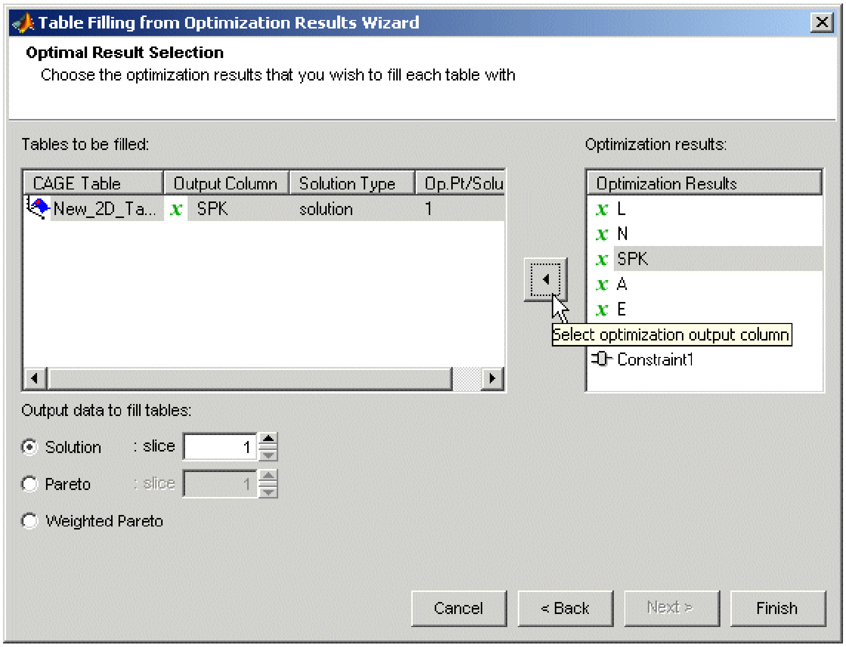
There are two methods for filling tables with optimization results.

1 From the `foptcon_Output` optimization output node, select **Solution –> Fill Tables**.

   The **Table Filling** wizard appears.

2 Select the `New_2D_Table` to fill and click **Add**. Click **Next**.

3 Select `SPK` from the list of optimization results and click the button to select, as shown. Notice that by default you will fill the table with solution `1` — in this single objective optimization there is only one solution for each operating point.



Optimal Result Selection

Choose the optimization results that you wish to fill each table with

Tables to be filled:

CAGE Table	Output Column	Solution Type	Op.Pt/Solu
New_2D_Ta...	SPK	solution	1

Optimization results:

Optimization Results
L
N
SPK
A
E
Constraint1

Select optimization output column

Output data to fill tables:

● Solution : slice 1

○ Pareto : slice 1

○ Weighted Pareto

4 Click **Finish**.

You will see a dialog reporting successful table filling. Switch to the **Tables** view to examine the new spark table.

The other method of filling tables with optimization output uses Data Sets.

1 From the `foptcon_Output` optimization output node, click Export to Data Set ( )in the toolbar (or select **Solution –> Export to Data Set**)

2 Go to the **Data Sets** view (click **Data Sets** in the **Data Objects** pane) to see that the table of optimization results is contained in the new data set `foptcon_sol1`. The new data set takes the name of the optimization (in this case `foptcon`) and the suffix `sol1` identifies that you exported solution one from the solution view. In this case there is only one solution, but for multiobjective optimizations there are more than one. This is covered in a later section of this tutorial, “Selecting Best Solutions” on page 6-29.

You can now use this data set (or any optimization results) to fill tables, as you can with any data set.

3. You can fill tables in the **Data Sets** view. Select the data set `foptcon_sol1` and if you are not already in the table view, click the View Data button in the toolbar.
4. 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.
5. Click (Fill Table From Data Set) in the toolbar.
6. Choose to fill the spark table with the `Copy_of_SPK` optimization output by selecting them in the two lists, then click **Fill Table**.

See also

- “Tutorial: Filling Tables from Data” on page 5-1 for more details on using data sets to fill tables.

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**.

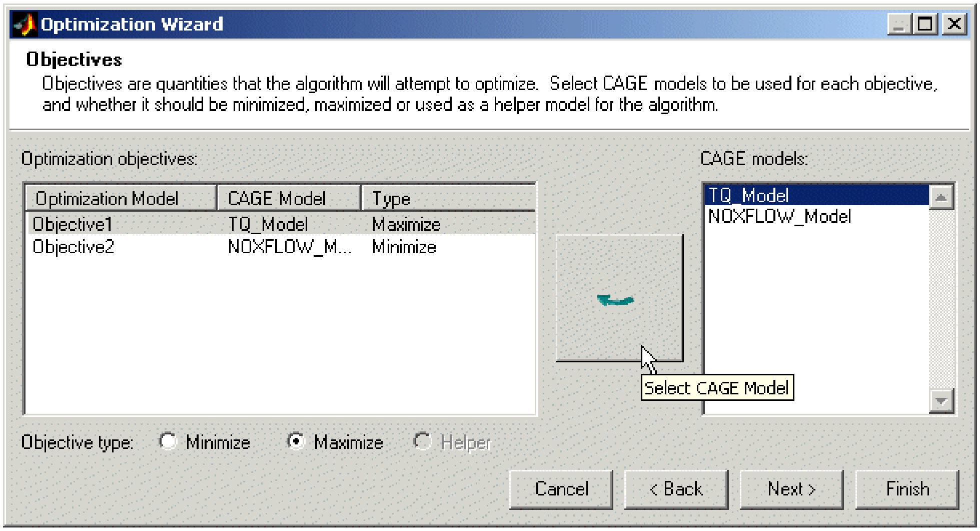
This opens the **Optimization Wizard**.

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.



Objectives

Objectives are quantities that the algorithm will attempt to optimize. Select CAGE models to be used for each objective, and whether it should be minimized, maximized or used as a helper model for the algorithm.

Optimization objectives:

Optimization Model	CAGE Model	Type
Objective1	TQ_Model	Maximize
Objective2	NOXFLOW_M...	Minimize

CAGE models:

TQ_Model
NOXFLOW_Model

Objective type: ○ Minimize ● Maximize ○ Helper a Select TQ_Model and click the button to match it up to Objective1, then select the **Maximize** radio button.

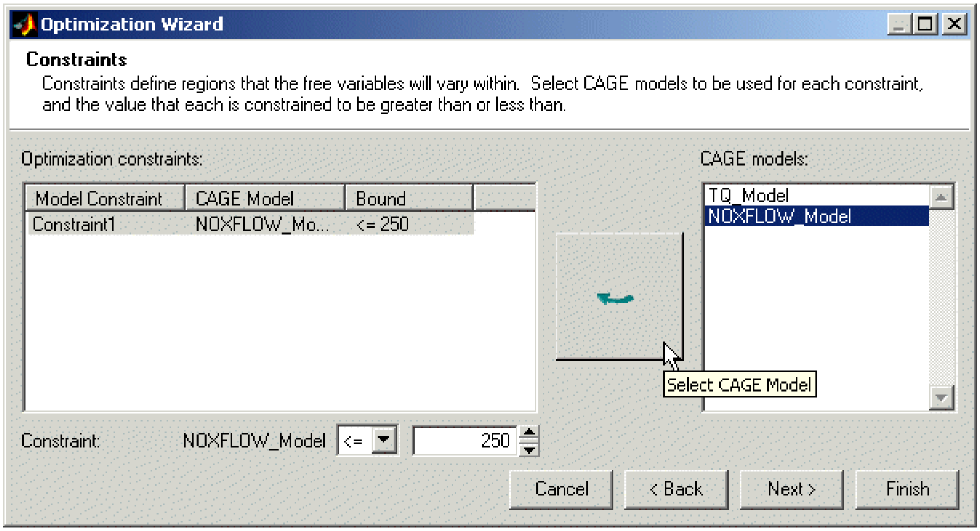
b Select NOXFLOW_Model and click the button to match it up to Objective2. Leave the **Minimize** radio button selected

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.

c Click **Next**.

6 On the next screen you can set up a constraint.



Optimization Wizard

Constraints

Constraints define regions that the free variables will vary within. Select CAGE models to be used for each constraint, and the value that each is constrained to be greater than or less than.

Optimization constraints:

Model Constraint	CAGE Model	Bound
Constraint1	NOXFLOW_Mo...	<= 250

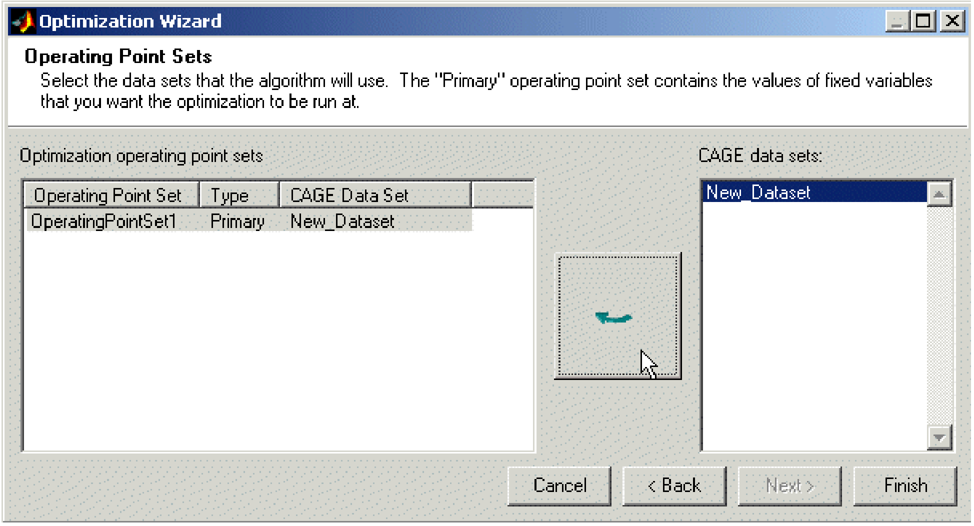
CAGE models:

TQ_Model
NOXFLOW_Model

Constraint: NOXFLOW_Model <= 250

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

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



Operating Point Sets

Select the data sets that the algorithm will use. The "Primary" operating point set contains the values of fixed variables that you want the optimization to be run at.

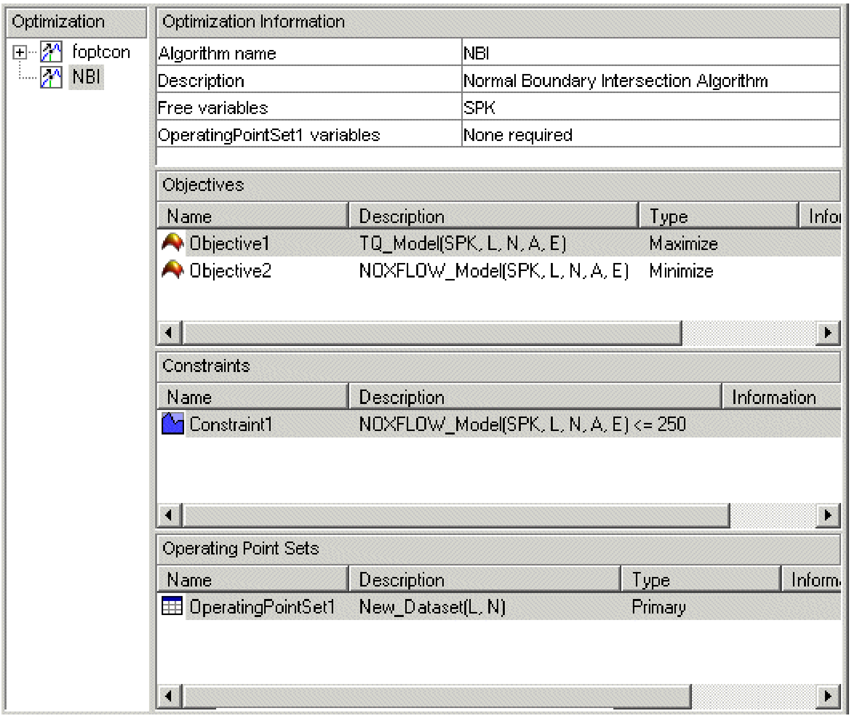
Optimization operating point sets

Operating Point Set	Type	CAGE Data Set
OperatingPointSet1	Primary	New_Dataset

CAGE data sets:

New_Dataset

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	Infor
Objective1	TQ_Model(SPK, L, N, A, E)	Maximize	
Objective2	NOXFLOW_Model(SPK, L, N, A, E)	Minimize	

Constraints

Name	Description	Information
Constraint1	NOXFLOW_Model(SPK, L, N, A, E) <= 250	

Operating Point Sets

Name	Description	Type	Inform
OperatingPointSet1	New_Dataset(L, N)	Primary	

8 Click Run Optimization ( )in the toolbar.

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

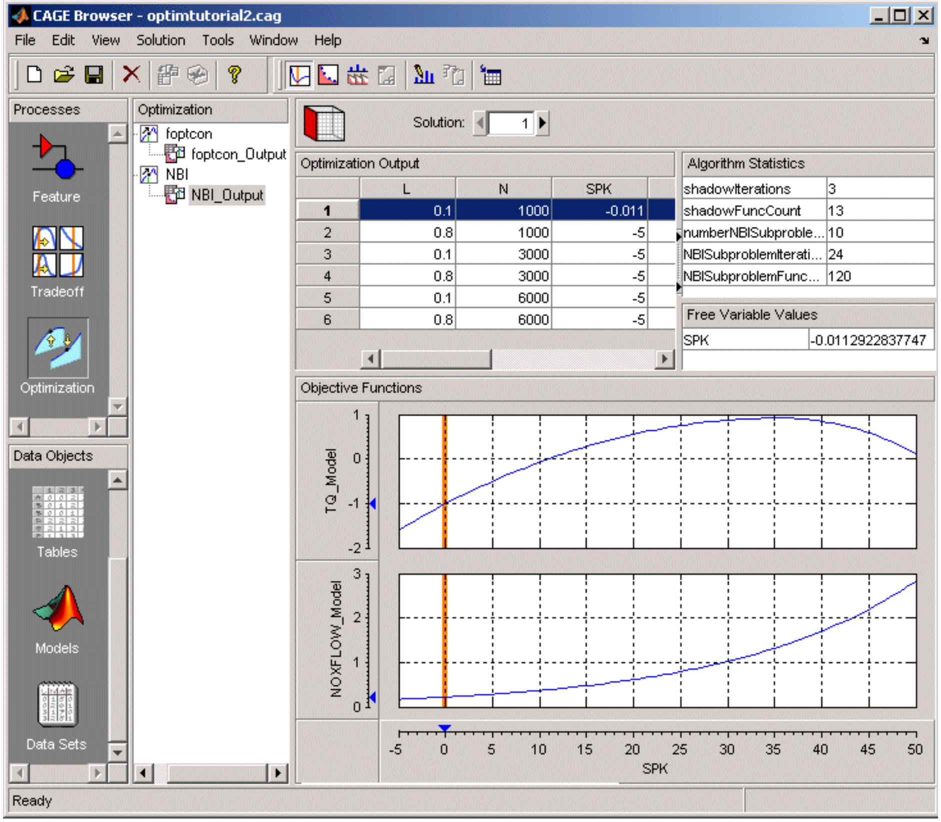
- SPK is the free variable, so the optimization will provide values for SPK as it searches for the optimum. A starting value and range are required for SPK.
- L and N are specified by the operating point set, but you need to give values for the other fixed variables, A and E.

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 View

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.



Optimization Output

	L	N	SPK
1	0.1	1000	-0.011
2	0.8	1000	-5
3	0.1	3000	-5
4	0.8	3000	-5
5	0.1	6000	-5
6	0.8	6000	-5

Algorithm Statistics

shadowIterations	3
shadowFuncCount	13
numberNBISubproble...	10
NBISubproblemIterati...	24
NBISubproblemFunc...	120

Free Variable Values

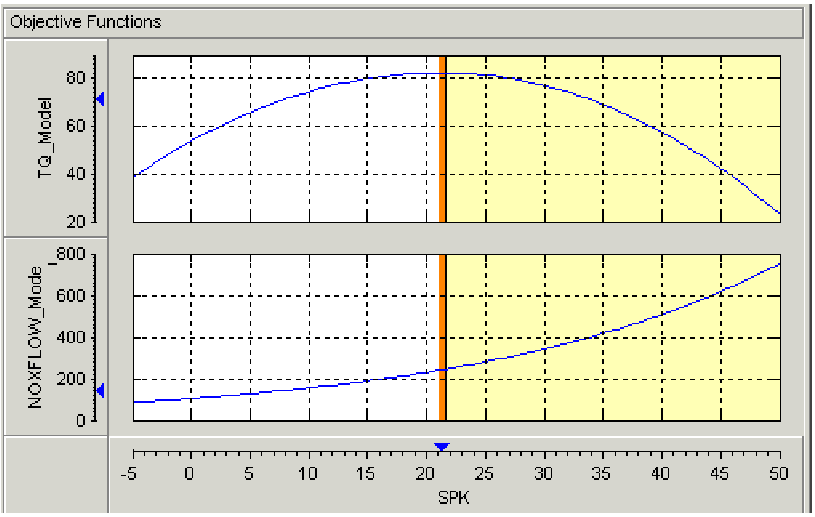
SPK	-0.0112922837747

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.

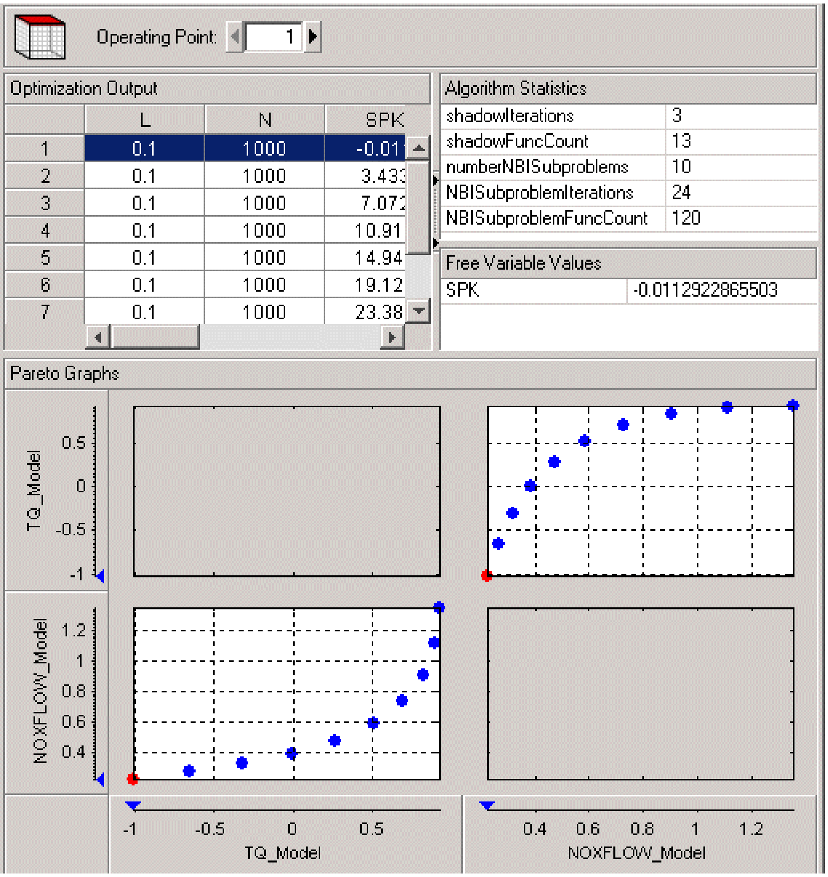
The graphs show the objective functions at the currently selected operating point (highlighted in the table), with the solution value shown in red.

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.



Operating Point: 1

Optimization Output

	L	N	SPK
1	0.1	1000	-0.01
2	0.1	1000	3.433
3	0.1	1000	7.072
4	0.1	1000	10.91
5	0.1	1000	14.94
6	0.1	1000	19.12
7	0.1	1000	23.38

Algorithm Statistics

shadowIterations	3
shadowFuncCount	13
numberNBISubproblems	10
NBISubproblemIterations	24
NBISubproblemFuncCount	120

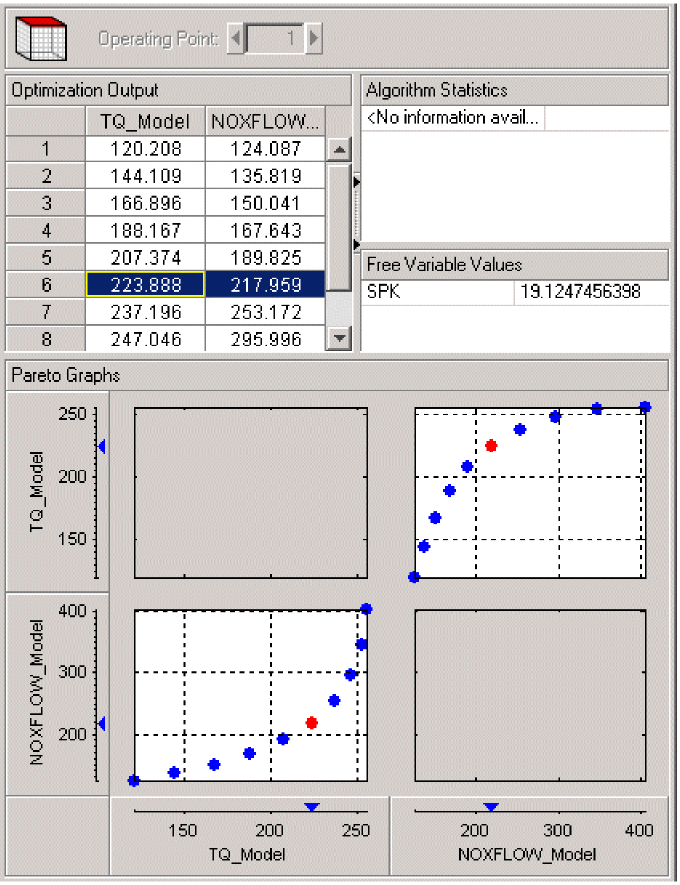
Free Variable Values

SPK	-0.0112922865503

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 6-29.

4 Click Weighted Pareto View ( ) in the toolbar.



Operating Point: 1

Optimization Output

	TQ_Model	NOXFLOW...
1	120.208	124.087
2	144.109	135.819
3	166.896	150.041
4	188.167	167.643
5	207.374	189.825
6	223.888	217.959
7	237.196	253.172
8	247.046	295.996

Algorithm Statistics

<No information avail...

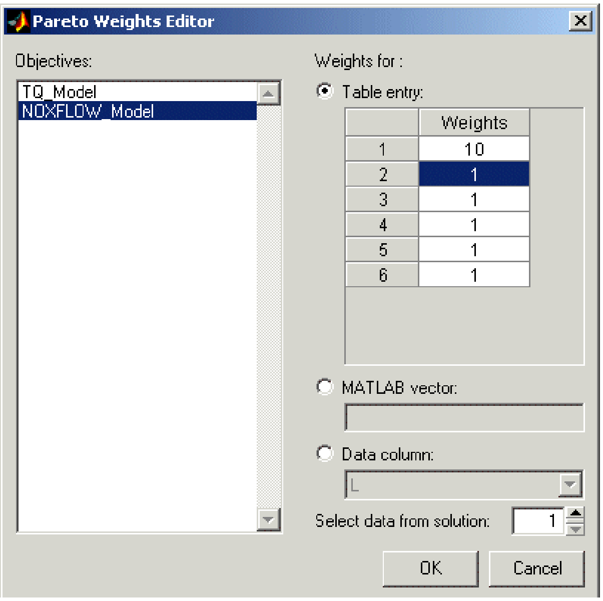
Free Variable Values

SPK	19.1247456398

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.



TQ_Model
NOXFLOW_Model

● Table entry:

	Weights
1	10
2	1
3	1
4	1
5	1
6	1

○ MATLAB vector:

○ Data column: L

Select data from solution: 1

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.

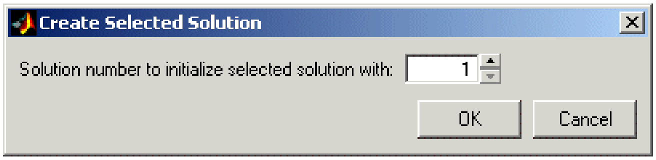
Selecting Best Solutions

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**.

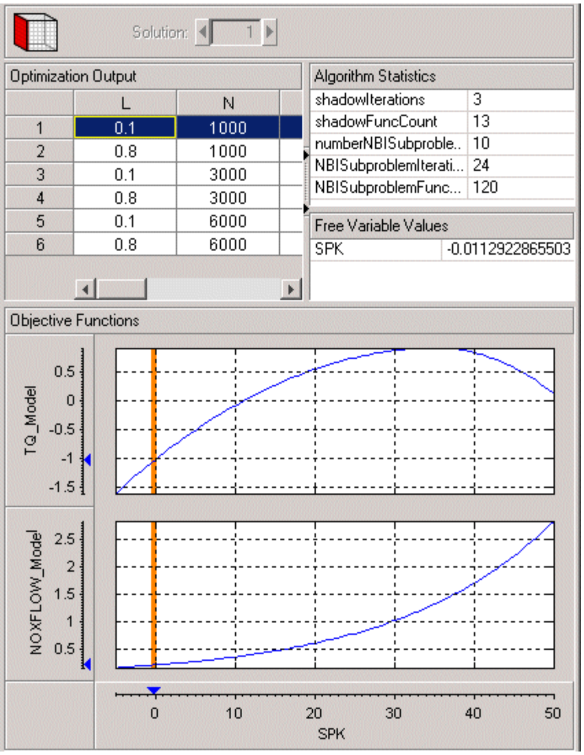
   A dialog called **Create Selected Solution** appears. Click **OK**.

   Solution number to initialize selected solution with: 1

   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.



Optimization Output

	L	N
1	0.1	1000
2	0.8	1000
3	0.1	3000
4	0.8	3000
5	0.1	6000
6	0.8	6000

Algorithm Statistics

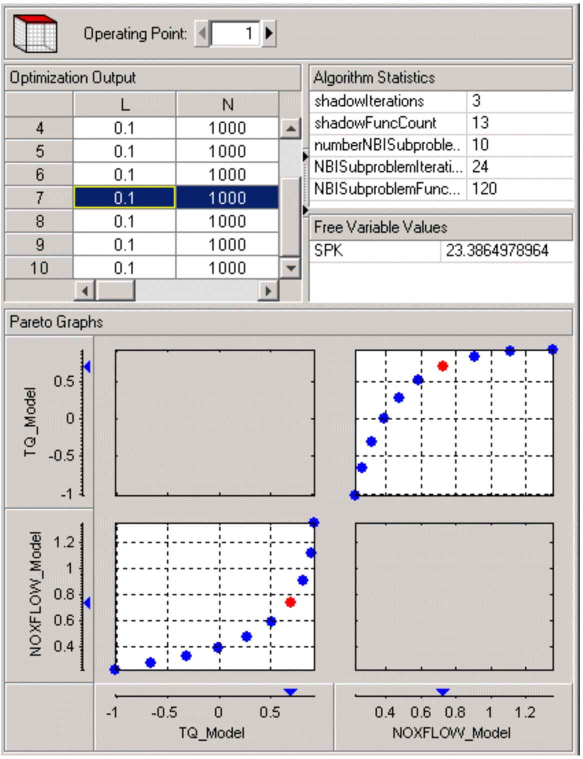
shadowIterations	3
shadowFuncCount	13
numberNBISubproble..	10
NBISubproblemIterati...	24
NBISubproblemFunc...	120

Free Variable Values

SPK	-0.0112922865503

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**.



Optimization Output

	L	N
4	0.1	1000
5	0.1	1000
6	0.1	1000
7	0.1	1000
8	0.1	1000
9	0.1	1000
10	0.1	1000

Algorithm Statistics

shadowIterations	3
shadowFuncCount	13
numberNBISubproble..	10
NBISubproblemIterati...	24
NBISubproblemFunc...	120

Free Variable Values

SPK	23.3864978964

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 **Data Objects** pane) to see that the table of optimization results is contained in a new data set, `NBI_selectedsol`. 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 must be sum objectives, and the optimization must contain a data set. You can use a mixture of point and model sum constraints, but if you include a sum constraint then all objectives *must* be sum objectives. 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** (or press **F2**). Edit the name to read `SUM_NBI`.

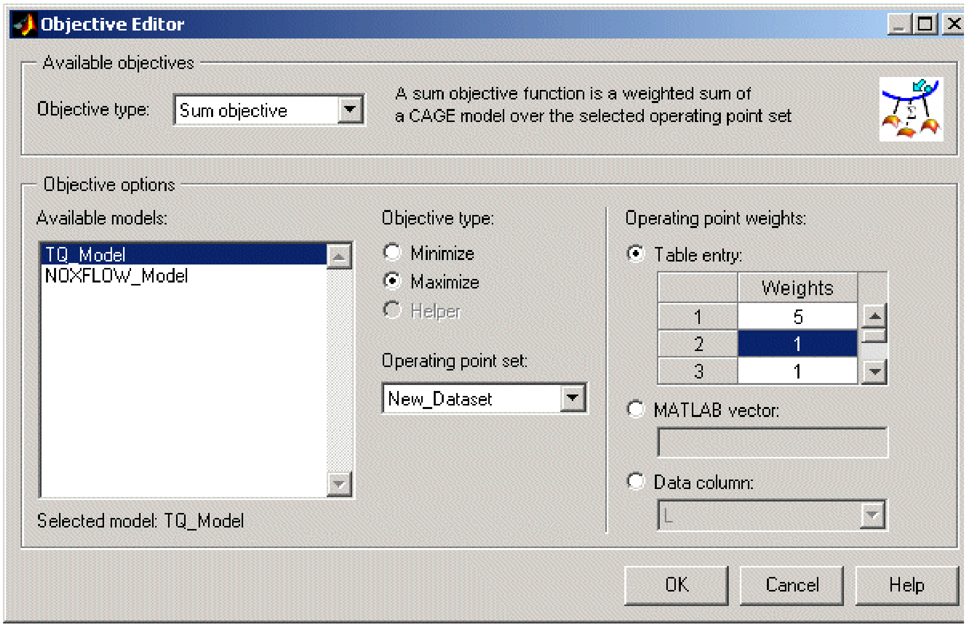
   You can only construct a sum optimization if your optimization contains an operating point set (as you cannot construct a sum if there is only one point).

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

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.



Available objectives

Objective type: Sum objective

A sum objective function is a weighted sum of a CAGE model over the selected operating point set

Objective options

Available models:
TQ_Model
NOXFLOW_Model

Selected model: TQ_Model

Objective type:
○ Minimize
● Maximize
○ Helper

Operating point set: New_Dataset

Operating point weights:
● Table entry:

	Weights
1	5
2	1
3	1

○ MATLAB vector:

○ Data column: L

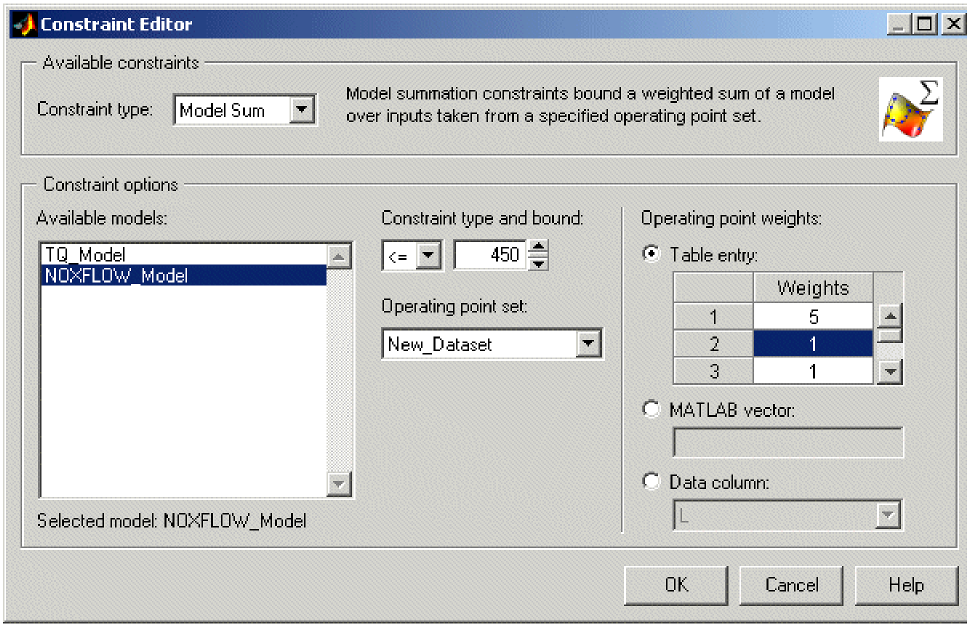
6 Select `TQ_Model` and **Maximize**. Torque has a strong correlation with fuel consumption so this sort of problem could be useful for a fuel consumption study.

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*operating point 1.

10 You can also edit your constraint to be a sum constraint. Note that you can use a mixture of point and sum constraints, but if you include a sum constraint then all objectives must be sum objectives or the optimization cannot be evaluated. Double-click `Constraint1` and the **Constraint Editor** appears.



Constraint Editor

Available constraints

Constraint type: Model Sum

Model summation constraints bound a weighted sum of a model over inputs taken from a specified operating point set.

Constraint options

Available models:

TQ_Model
NOXFLOW_Model

Selected model: NOXFLOW_Model

Constraint type and bound: <= 450

Operating point set: New_Dataset

Operating point weights:

● Table entry:

	Weights
1	5
2	1
3	1

○ MATLAB vector:

○ Data column: L

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 (Add New Table). This opens the **Table Setup** dialog.

3 Enter `Spark` as the `Table name`.

4 Select `L` as the **Y name** and `N` as the **X name**.

5 Click **Select** to open the **Select Filling Item** dialog.

   a Select the radio button to Display variables.

   b Click to select `SPK`.

   c Click **OK** to return to the **Table Steup** dialog.

6 Leave `10` as the size of the rows and columns (the speed and load axes), and 0 as the initial value.

7 Click **OK**.

   A new `Spark` table appears in the **Tradeoff** tree. CAGE has automatically spaced the normalizers evenly over the ranges of `N` and `L`.

8 Click to expand the `New_Tradeoff` tree and select the `Spark` table node to view the new table.

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

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

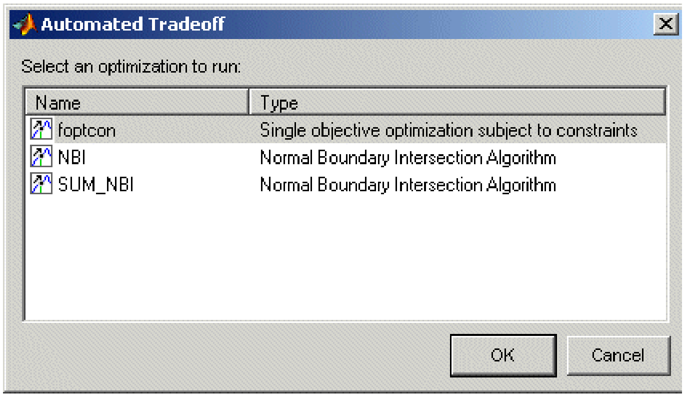
2 Click (or right-click and select **Define Region**) to define the region. The cells become light blue.

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

1 Click any cell in a region.

2 To apply optimization, select **Inputs –> Automated Tradeoff**.

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



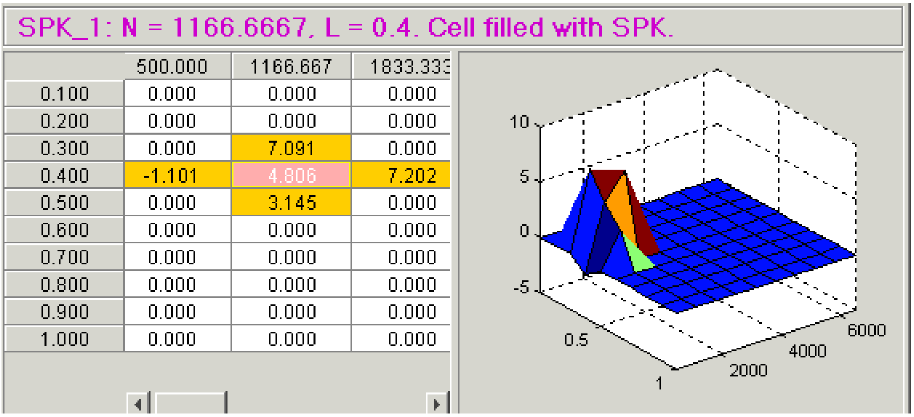
Automated Tradeoff

Select an optimization to run:

Name	Type
foptcon	Single objective optimization subject to constraints
NBI	Normal Boundary Intersection Algorithm
SUM_NBI	Normal Boundary Intersection Algorithm

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.



SPK_1: N = 1166.6667, L = 0.4. Cell filled with SPK.

	500.000	1166.667	1833.333
0.100	0.000	0.000	0.000
0.200	0.000	0.000	0.000
0.300	0.000	7.091	0.000
0.400	-1.101	4.806	7.202
0.500	0.000	3.145	0.000
0.600	0.000	0.000	0.000
0.700	0.000	0.000	0.000
0.800	0.000	0.000	0.000
0.900	0.000	0.000	0.000
1.000	0.000	0.000	0.000

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 11-47.

- 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.

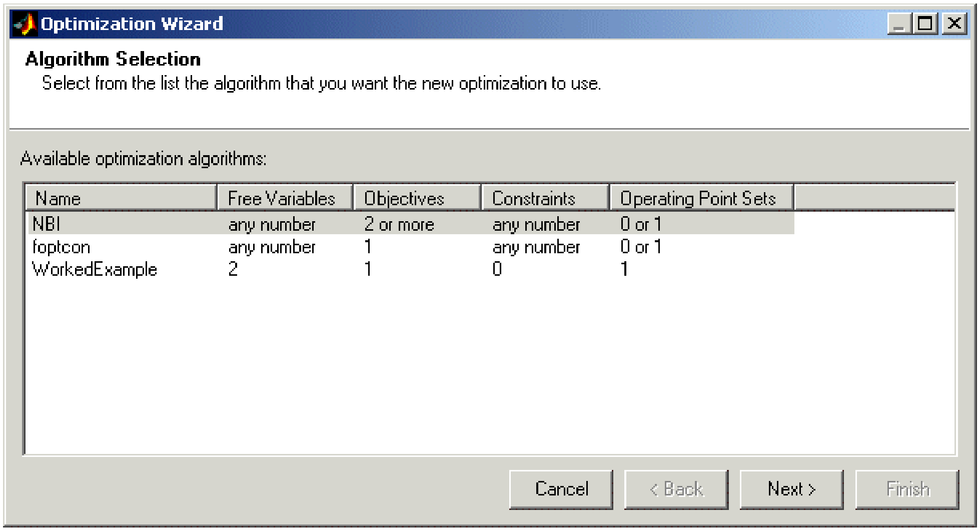
- The `tq` model was fitted to the Holliday engine data and exported from the Model Browser quick start tutorial (also used in the CAGE feature calibration tutorial). It can be found in `tutorial.exm` in the `mbctraining` directory. To view this model in your current session, click the **Models** button 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**.



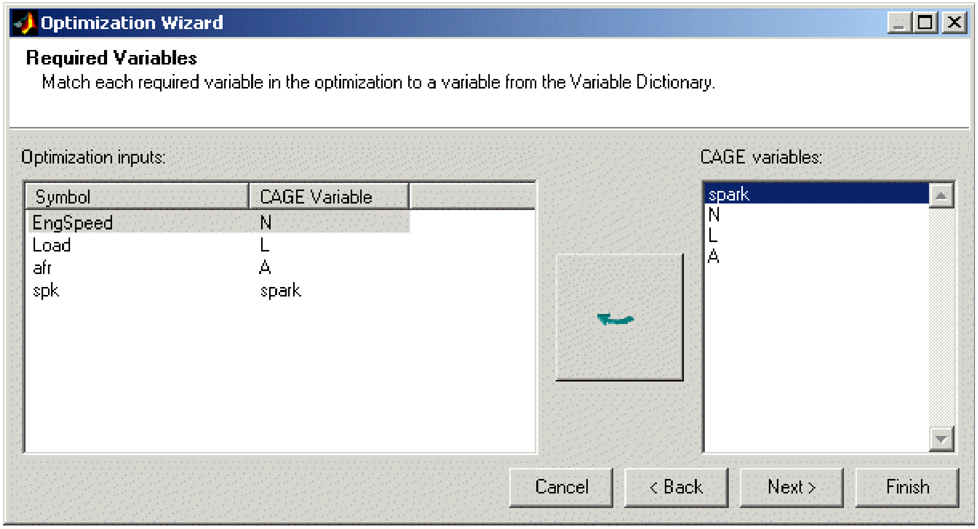
Algorithm Selection

Select from the list the algorithm that you want the new optimization to use.

Available optimization algorithms:

Name	Free Variables	Objectives	Constraints	Operating Point Sets
NBI	any number	2 or more	any number	0 or 1
foptcon	any number	1	any number	0 or 1
WorkedExample	2	1	0	1

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**.



Optimization Wizard

Required Variables

Match each required variable in the optimization to a variable from the Variable Dictionary.

Optimization inputs:

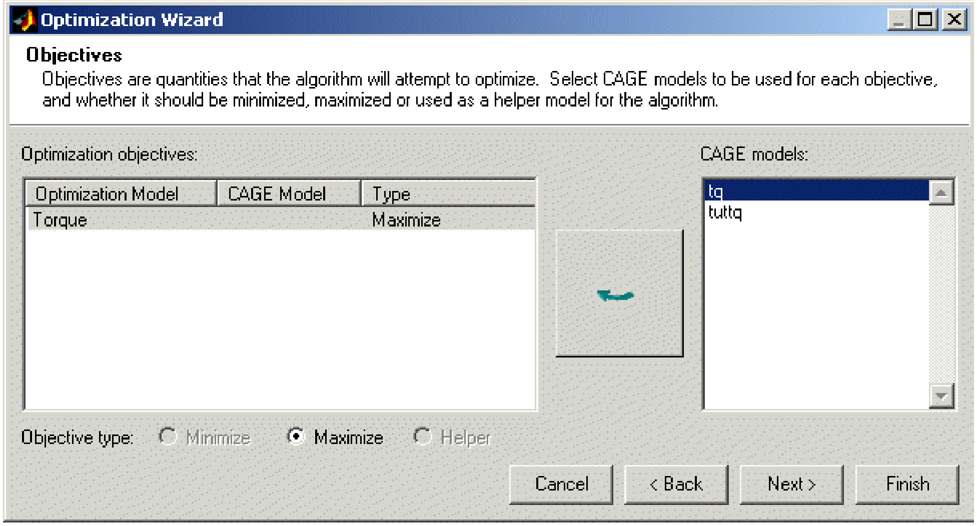
Symbol	CAGE Variable
EngSpeed	N
Load	L
afr	A
spk	spark

CAGE variables:

spark
N
L
A

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**.



Optimization Wizard

Objectives

Objectives are quantities that the algorithm will attempt to optimize. Select CAGE models to be used for each objective, and whether it should be minimized, maximized or used as a helper model for the algorithm.

Optimization objectives:

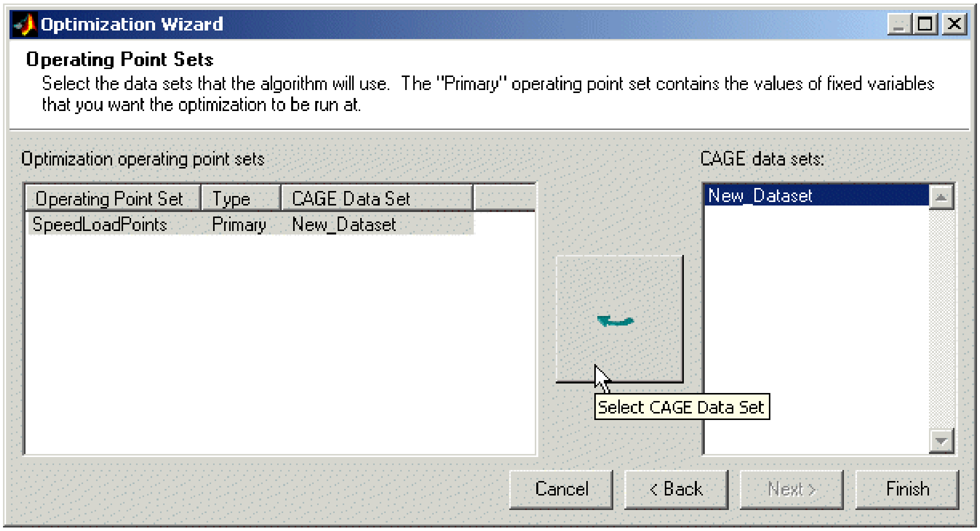
Optimization Model	CAGE Model	Type
Torque		Maximize

CAGE models:

tq
tuttq

Objective type: ○ Minimize ● Maximize ○ Helper

6 On the next screen of the wizard select the `New_Dataset`. Click the button to match it to the operating point set `SpeedLoadPoints`, then click **Finish**.



Optimization Wizard

Operating Point Sets

Select the data sets that the algorithm will use. The "Primary" operating point set contains the values of fixed variables that you want the optimization to be run at.

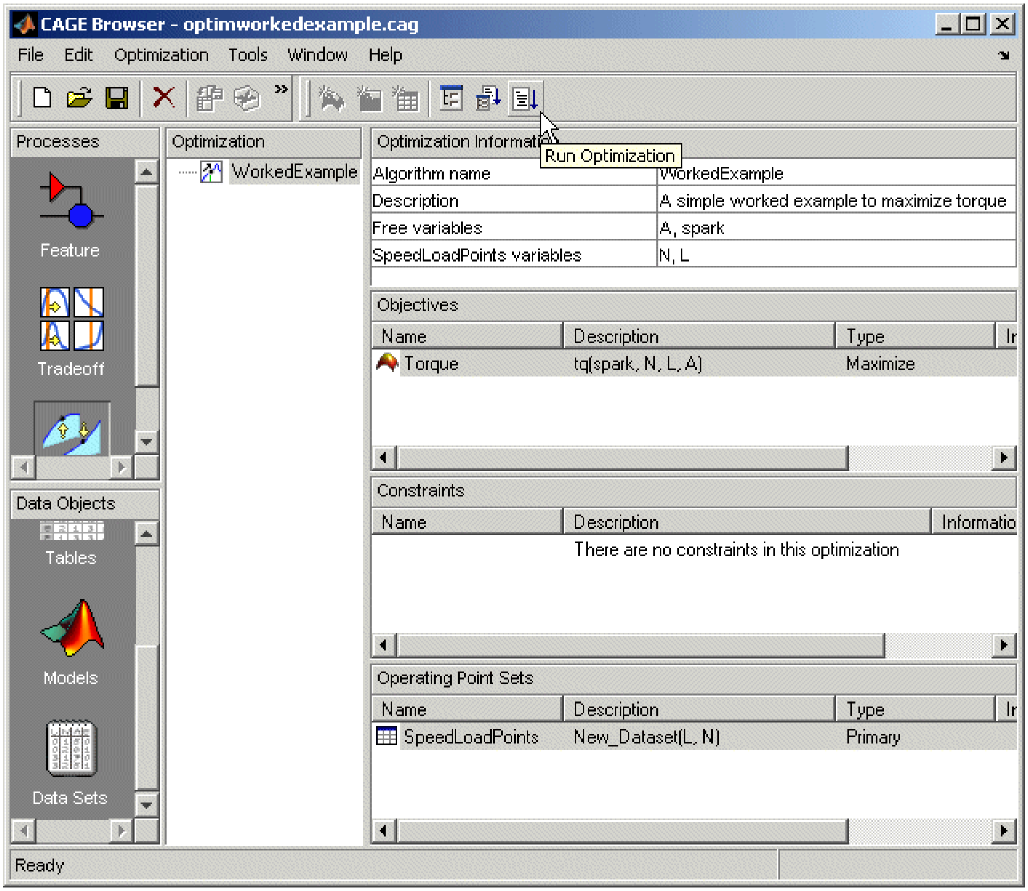
Optimization operating point sets

Operating Point Set	Type	CAGE Data Set
SpeedLoadPoints	Primary	New_Dataset

CAGE data sets:

New_Dataset

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.



Algorithm name	WorkedExample
Description	A simple worked example to maximize torque
Free variables	A, spark
SpeedLoadPoints variables	N, L

Objectives

Name	Description	Type	Ir
Torque	tq(spark, N, L, A)	Maximize	

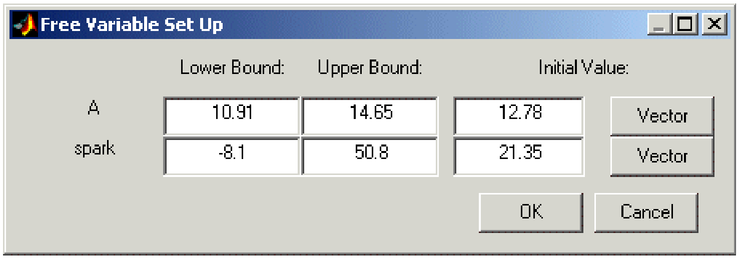
Constraints

Name	Description	Informatio
	There are no constraints in this optimization	

Operating Point Sets

Name	Description	Type	Ir
SpeedLoadPoints	New_Dataset(L, N)	Primary	

7 Click Run Optimization ▤ in the toolbar.

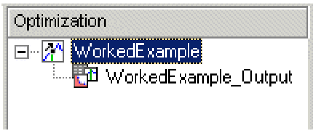


	Lower Bound:	Upper Bound:	Initial Value:
A	10.91	14.65	12.78
spark	-8.1	50.8	21.35

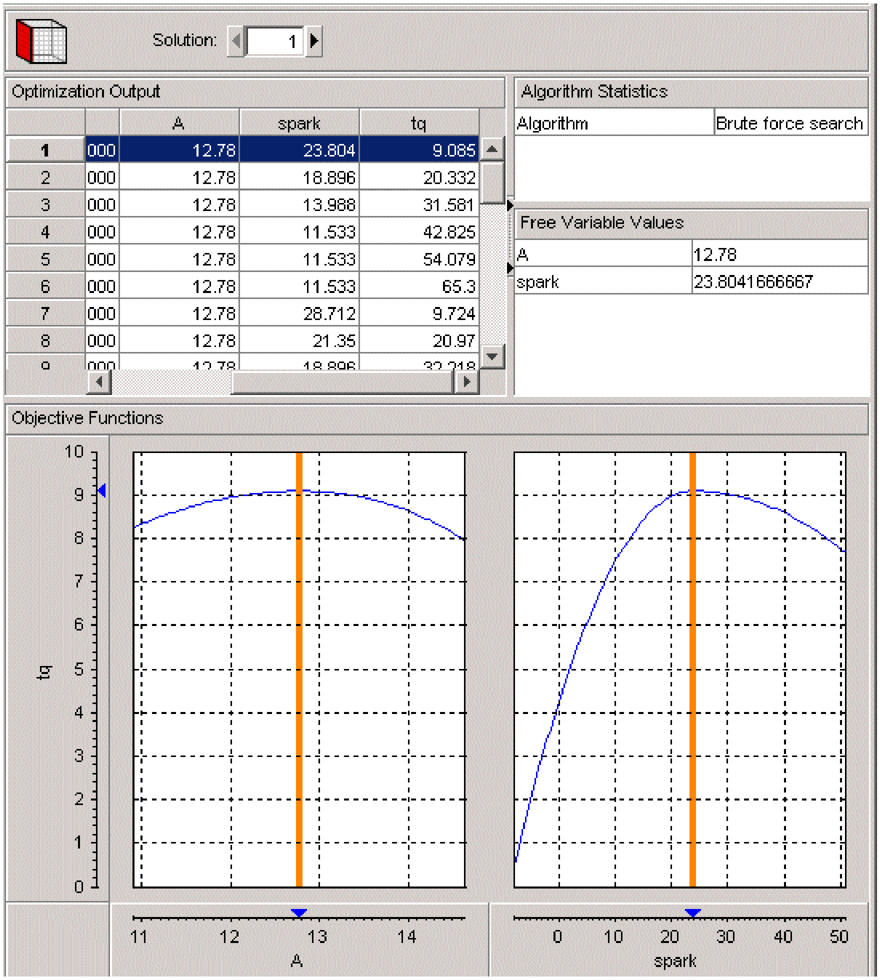
The **Free Variable Set Up** dialog box appears. These bounds and initial values are taken from the Variable Dictionary (where you also could change them if you wanted). Click **OK** to accept these values.

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.



Solution: 1

Optimization Output

		A	spark	tq
1	000	12.78	23.804	9.085
2	000	12.78	18.896	20.332
3	000	12.78	13.988	31.581
4	000	12.78	11.533	42.825
5	000	12.78	11.533	54.079
6	000	12.78	11.533	65.3
7	000	12.78	28.712	9.724
8	000	12.78	21.35	20.97
9	000	12.78	18.896	32.218

Algorithm Statistics

Algorithm	Brute force search

Free Variable Values

A	12.78
spark	23.8041666667

Objective Functions

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 11-44. 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 6-48.

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:

```
currtutoptim.m
mbcOStemplate.m
mbcOStutoptimfunc.m
mbcOStutoptimfunc_s1.m
optimtut.mat
optimtuteg.mat
```

3 Make sure your new working directory is on the MATLAB path.

   a Select **File –> Set Path**.

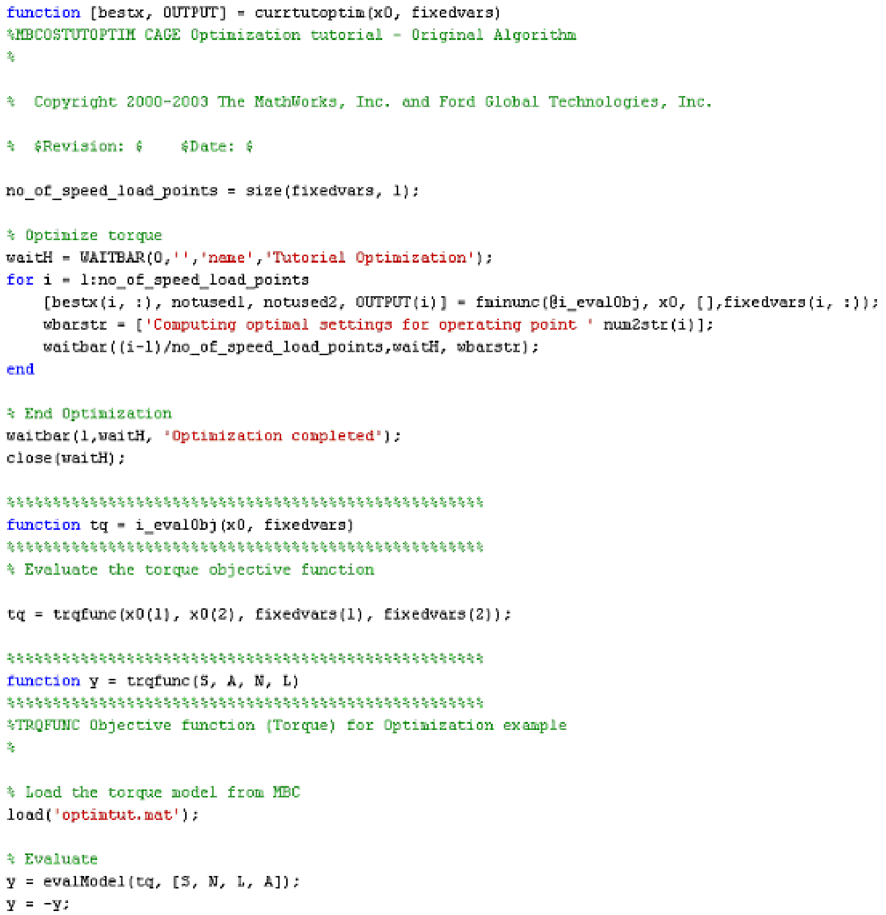
   b Click **Add Folder** and browse to your working directory.

   c Click **OK**.

   d Click **Save**.

Step 1: Examine the Algorithm

1 Open `currtutoptim.m` from the `mbctraining` directory. `currtutoptim.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] = currtutoptim(x0, fixedvars)
%MBCOSTUTOPTIM 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 = currtutoptim(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 `Holliday.mat`) at the speed and load points defined in the matrix data.

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

This tutorial shows how to make currtutoptim.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.

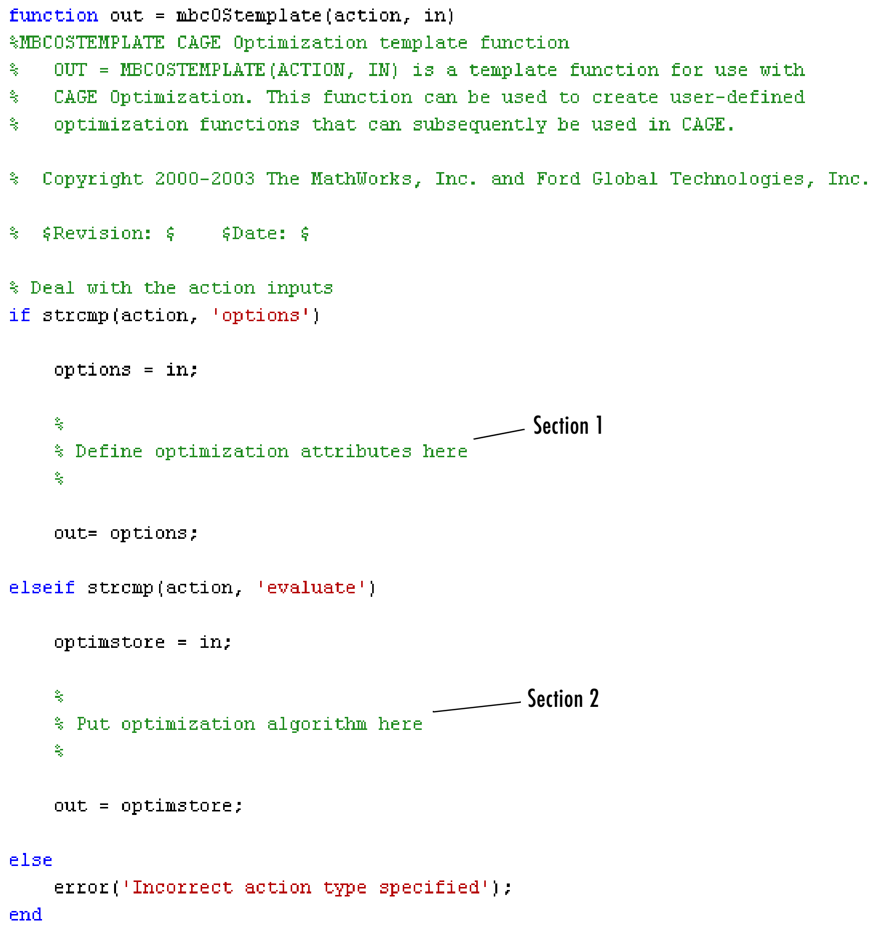
The first section defines the following attributes of the optimization:

- 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
```

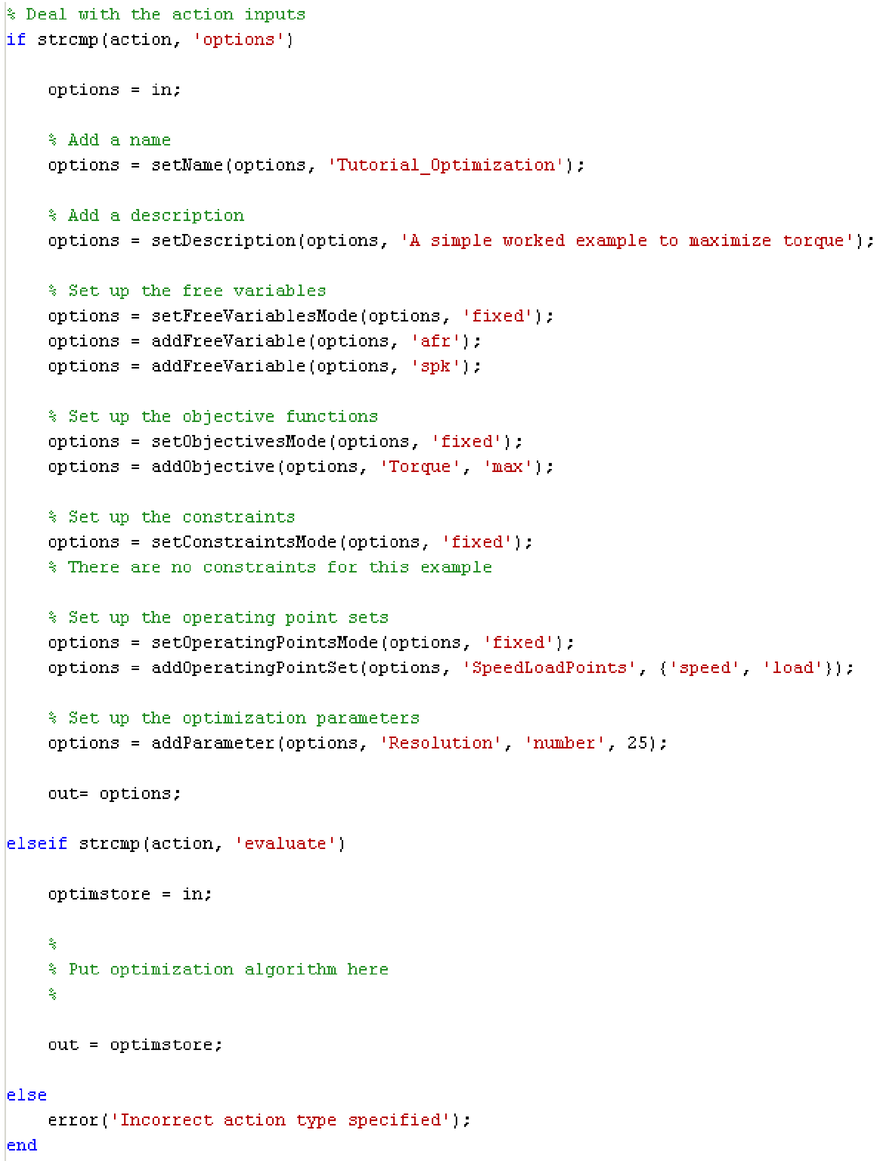
mbcOStemplate.m is an empty CAGE optimization function. The two (currently empty) sections of the function are labeled above. Note that this M-file can be used as a template for any optimization function that you write.

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 `cgoptimoptions` object, referenced by options in the code in `mbcOStutoptimfunc_s1.m`. See after the table for details of the `cgoptimoptions` object. The `cgoptimoptions` 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 11-49. The above code has used the `cgoptimoptions` 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	`Tutorial_Optimization`	`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`
		`setFreeVariablesMode`

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`
		`addFreeVariables`

Number of Objectives	Cannot be changed by the user in the GUI (the mode has been set to `'fixed'`)	`Set up the objective functions`
		`setObjectivesMode`

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` `addObjective`

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)
```

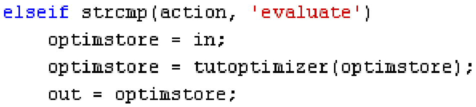
This is how your optimization function receives the `cgoptimoptions` object. Note that your optimization function *must* support this interface.

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 `mbcOStutoptimfunc.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 `mbcOStutoptimfunc.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, `currtutoptim`, 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_indexth 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**.

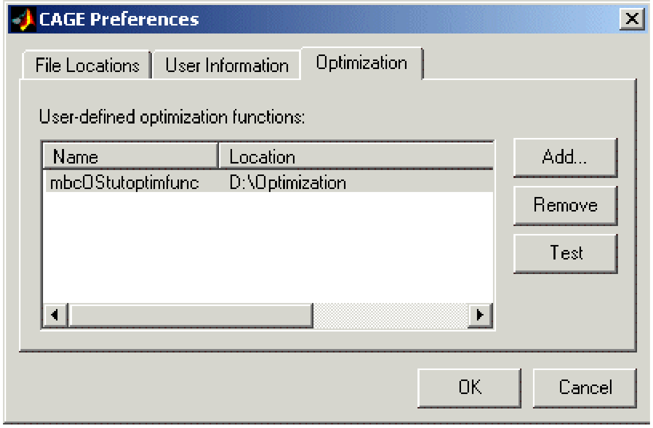
1 In CAGE, select **File –> Preferences**.

   The **CAGE Preferences** dialog appears.

2 Click the **Optimization** tab and click **Add** to browse to your M-file.

3 Navigate to the `mbcOStutoptimfunc.m` file (in the working directory you created) and click **Open**.

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



Name	Location
mbcOStutoptimfunc	D:\Optimization

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.

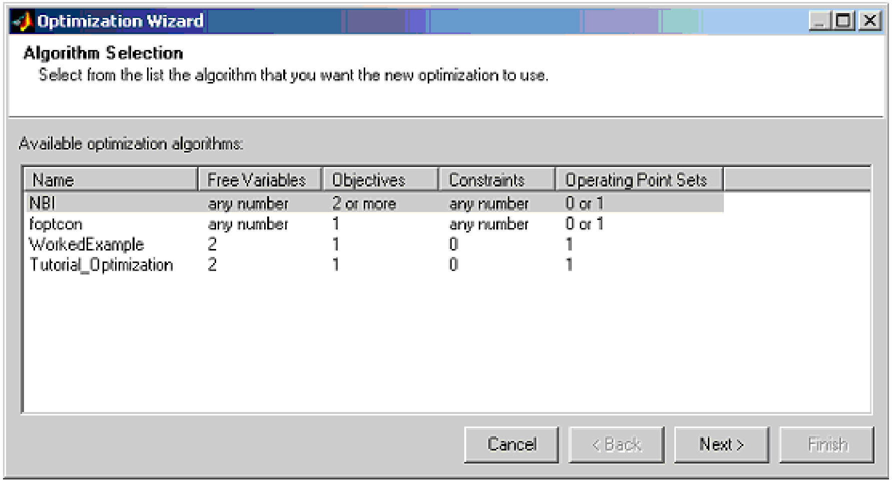
5 Click **OK** to leave the **CAGE Preferences** dialog. If the optimization function tested successfully, it is registered as an optimization function that can be used in CAGE, and appears in the **Optimization Wizard**.

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**.



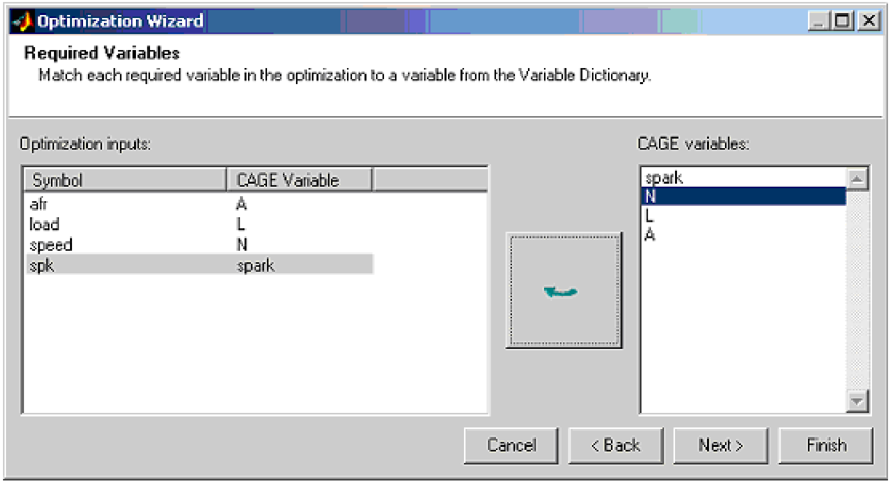
Algorithm Selection

Select from the list the algorithm that you want the new optimization to use.

Available optimization algorithms:

Name	Free Variables	Objectives	Constraints	Operating Point Sets
NBI	any number	2 or more	any number	0 or 1
foptcon	any number	1	any number	0 or 1
WorkedExample	2	1	0	1
Tutorial_Optimization	2	1	0	1

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



Required Variables

Match each required variable in the optimization to a variable from the Variable Dictionary.

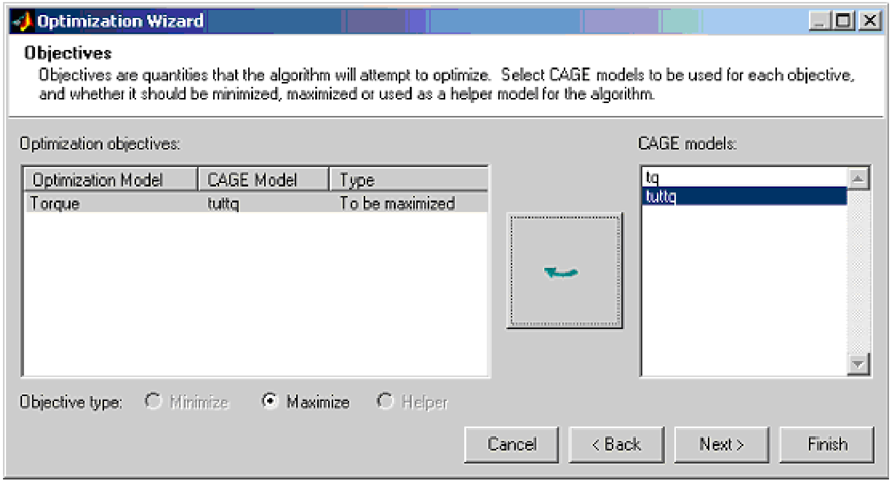
Optimization inputs:

Symbol	CAGE Variable
afr	A
load	L
speed	N
spk	spark

CAGE variables:

spark
N
L
A

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



Optimization Wizard

Objectives

Objectives are quantities that the algorithm will attempt to optimize. Select CAGE models to be used for each objective, and whether it should be minimized, maximized or used as a helper model for the algorithm.

Optimization objectives:

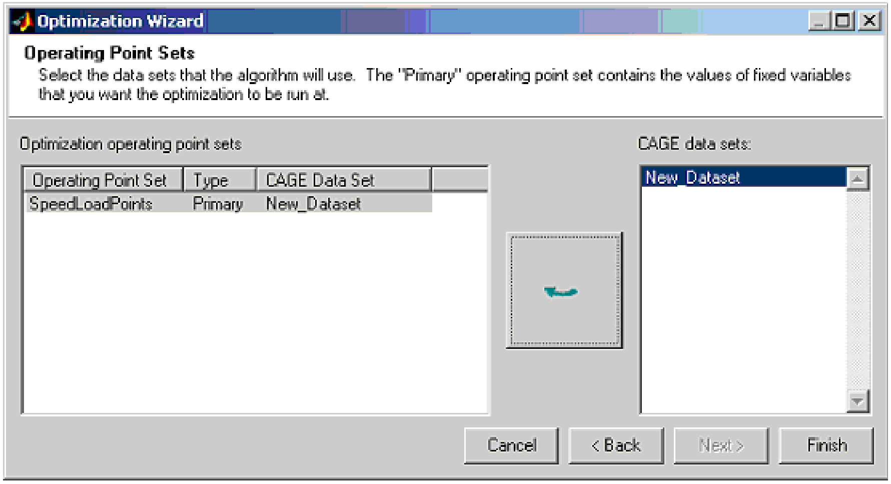
Optimization Model	CAGE Model	Type
Torque	tuttq	To be maximized

CAGE models:

- tq
- tuttq

Objective type: ○ Minimize ● Maximize ○ Helper

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



Optimization Wizard

Operating Point Sets

Select the data sets that the algorithm will use. The "Primary" operating point set contains the values of fixed variables that you want the optimization to be run at.

Optimization operating point sets

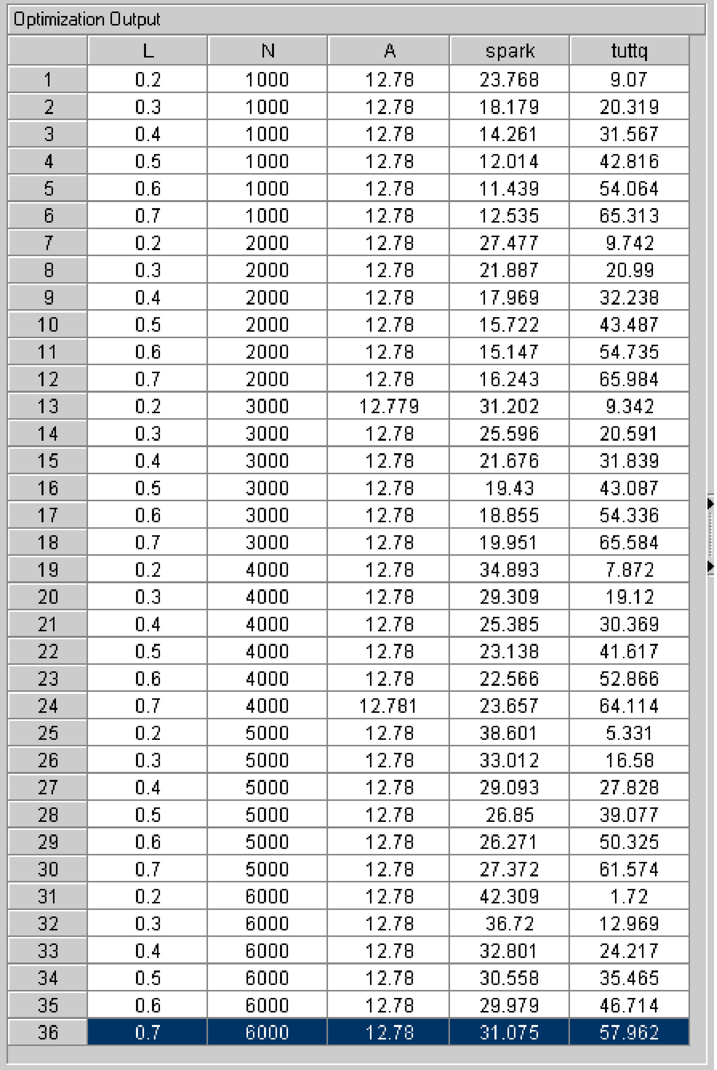
Operating Point Set	Type	CAGE Data Set
SpeedLoadPoints	Primary	New_Dataset

CAGE data sets:

- New_Dataset

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.



	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

7

Using CAGE

This section includes the following topics:

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 7-7
- “Setting Up Your Models” on page 7-14

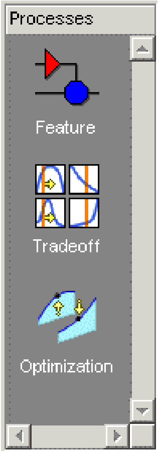
You can also use CAGE to calibrate tables directly from experimental data by interpolation, without using models. See “Tutorial: Filling Tables from Data” on page 5-1 for an example.

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 tree display

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 three buttons:

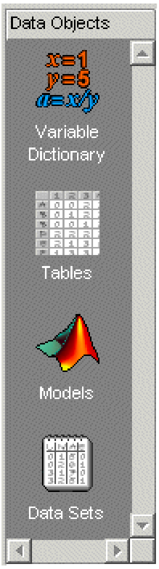
- **Feature** shows the **Feature** view, with the tables and strategies that are associated with that feature. See “Feature View” on page 9-47.

  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.

- **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 the **Feature** and **Tradeoff Process** views, and from the **Tables** view, but not **Optimization**. 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 four 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 7-7.
- **Tables** enables you to see all the tables and normalizers in your session. You can also calibrate tables manually here if you want. 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 15-6.
- **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 7-14 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.

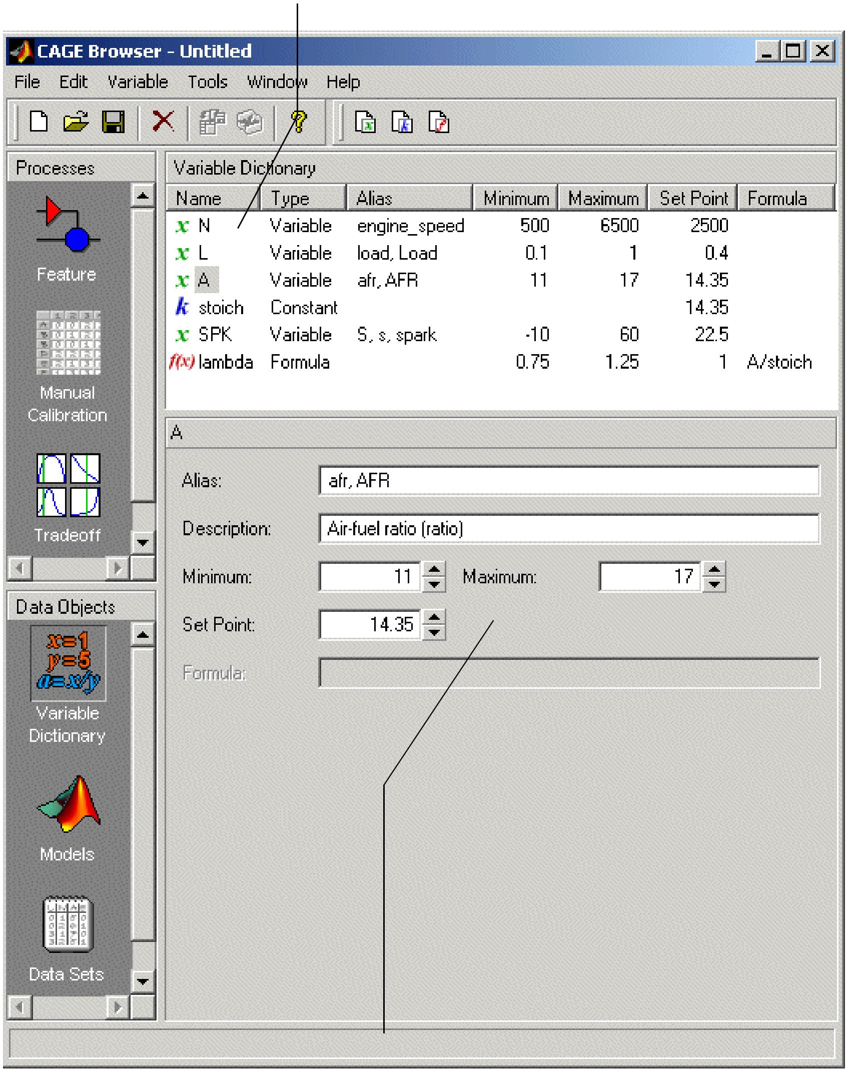
To view or edit the items in the **Variable Dictionary,** click the button, shown, in the **Data Objects** pane.

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 7-9
- "Adding Variable Items" on page 7-10
- "Using the Variable Menu" on page 7-12
- "Using Aliases" on page 7-13

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



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

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

Alias: afr, AFR

Description: Air-fuel ratio (ratio)

Minimum: 11 Maximum: 17

Set Point: 14.35

Formula:

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.

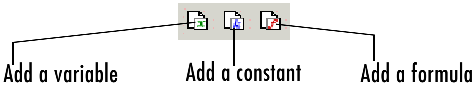
See Also

- “Setting Up Your Variable Items” on page 7-7
- “Adding Variable Items” on page 7-10

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. You can edit set points in the variable dictionary or the models view.

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

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

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, in this case `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 7-7
- “Adding Variable Items” on page 7-10

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 7-13.
- **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 7-7
- “Using Aliases” on page 7-13

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.

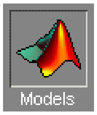
See Also

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

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 7-18
- “Renaming and Editing Models” on page 7-20

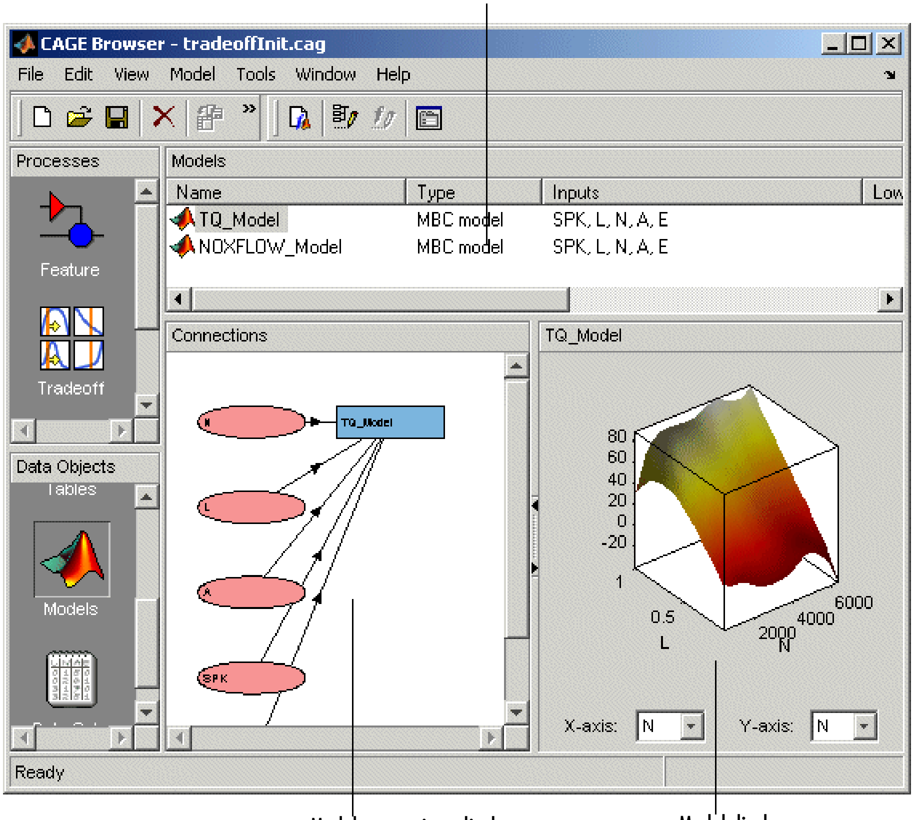
The **Models** view displays the following:

- 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. You can use the **View** menu to zoom in and out, zoom to fit, and reset.
- An image of the response surface of the selected model; you can select factors to display. Use the **View** menu to choose whether to display constraints and to edit input set points.

  View –> Edit Input Set Points opens a dialog where you can edit the set points of your model variables. This alters the model display and also any calculations involving the set points throughout CAGE. This is the same as altering the set points in the Variable Dictionary, see “Using Set Points in the Variable Dictionary” on page 7-10.

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

List of the current models



CAGE Browser - tradeoffInit.cag

Name	Type	Inputs	Low
TQ_Model	MBC model	SPK, L, N, A, E	
NOXFLOW_Model	MBC model	SPK, L, N, A, E	

X-axis: N

Y-axis: N

Ready

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.

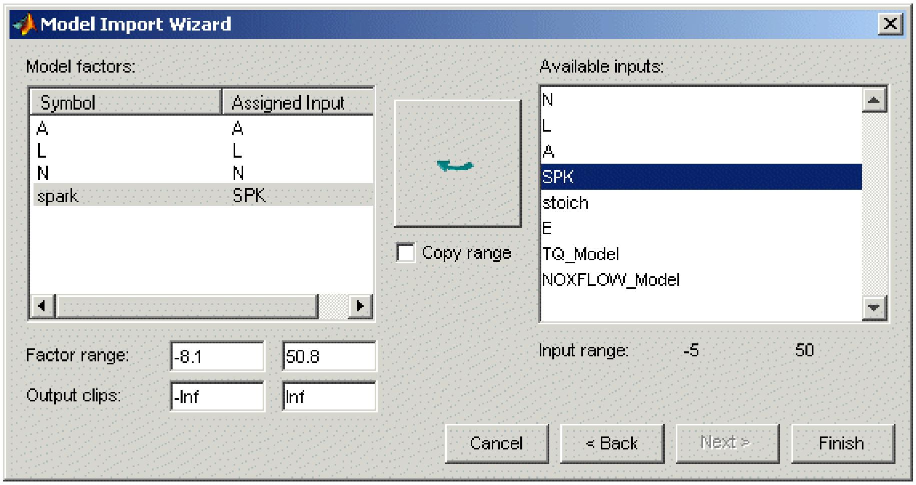
4 Click **Next >** to select the model file.

5 Select the models that you want to import by highlighting the models from the list, or click **Select All** if you want every model.

6 Click **Next >** to select the models, or select the check box **Automatically assign/create inputs**, then you can click **Finish**.

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,



Model factors:

Symbol	Assigned Input
A	A
L	L
N	N
spark	SPK

Available inputs:

N
L
A
SPK
stoich
E
TQ_Model
NOXFLOW_Model

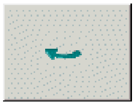
Factor range: -8.1 50.8

Output clips: -Inf Inf

Input range: -5 50

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 7 and 8 by selecting the **Automatically assign/create inputs** box at step 6.

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

See Also

- "Setting Up Your Models" on page 7-14
- "Adding New Function Models" on page 7-18

- “Renaming and Editing Models” on page 7-20

Adding New Function Models

A function model is a model that is expressed algebraically. The function can be any MATLAB function (including user-defined functions). The only restriction is that the function must be vectorized, that is, take in column vectors and return a column vector of the same size, as in this example:

```
function y = foo(x1, x2)
y = x1 .* x2;
```

Once you have a function like this, you can create a function model applying it to any models or variables in your session, like the following example.

```
foo(NOX, SPK)
```

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`.

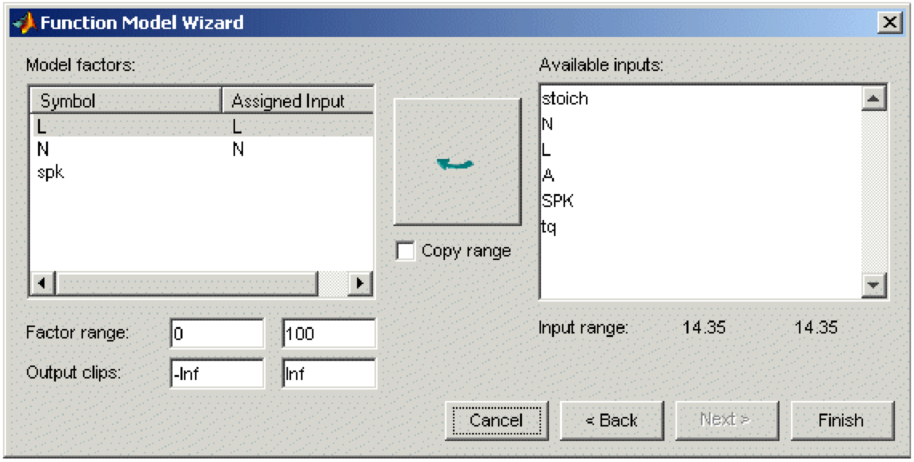
3. Press **Enter**. CAGE checks that the function is recognized; if so, you can click **Next** >. If the function is incorrectly entered, you cannot click **Next**.

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,



Model factors:

Symbol	Assigned Input
L	L
N	N
spk	

Available inputs:

stoich
N
L
A
SPK
tq

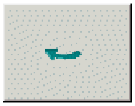
Factor range: 0 100

Output clips: -Inf Inf

Input range: 14.35 14.35

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 b 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 7-14
- “Importing Models” on page 7-15
- “Renaming and Editing Models” on page 7-20

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 Model Connections

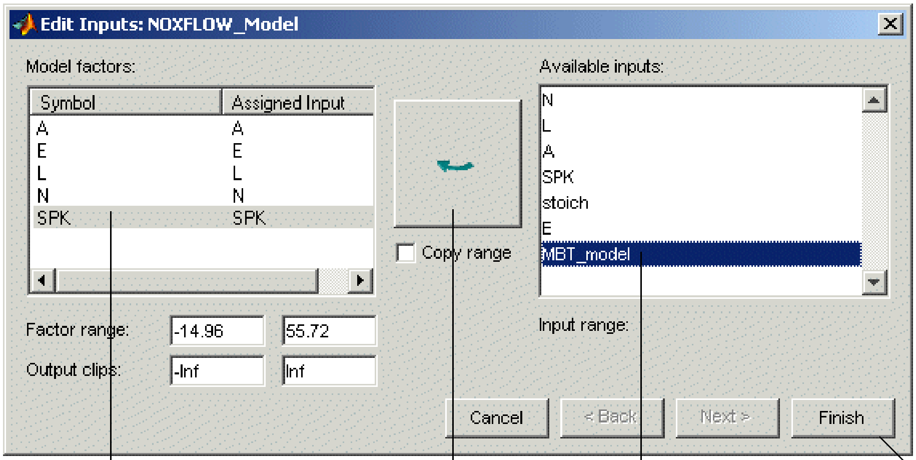
You can adjust a model so that variables, formulas, or other models are the factors of the model. For example, a model of torque depends on the spark angle. In place of the spark angle, you can have a model of the maximum brake torque (MBT).

To edit the connections of a model,

1 Highlight the model.

2 Select **Model –> Edit Inputs**.

This opens the **Edit Inputs** dialog box, shown.



Model factors:

Symbol	Assigned Input
A	A
E	E
L	L
N	N
SPK	SPK

□ Copy range

Available inputs:

N
L
A
SPK
stoich
E
MBT_model

Factor range: -14.96 55.72

Output clips: -Inf Inf

Input range:

Highlight the model factor that you want to change.

Click **Select Input**.

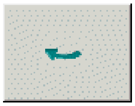
Highlight the new input.

Click **Finish**.

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 You can also edit the ranges. Highlight the model factor that you want to alter, and enter the new range in the **Factor range** boxes.

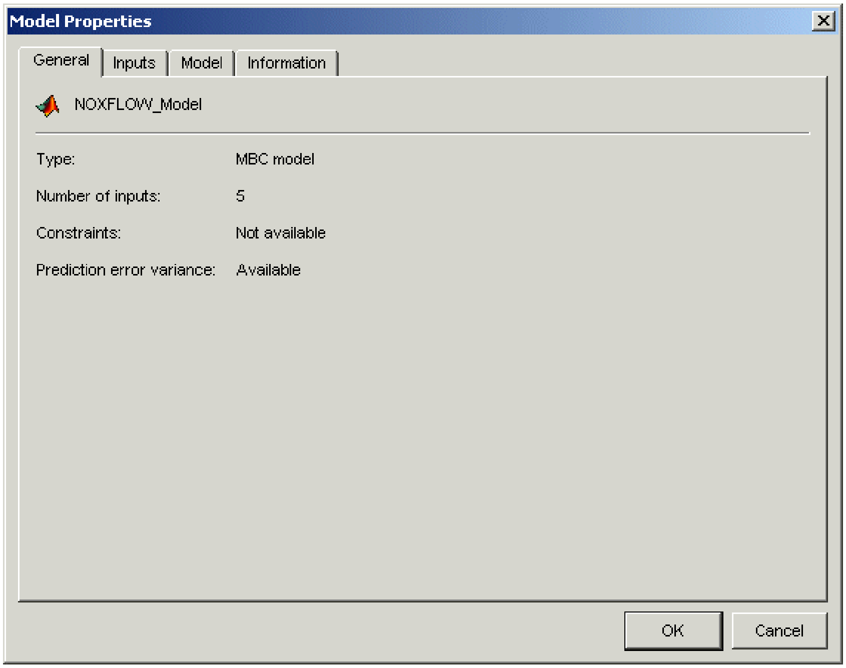
Note This does not change the range of the variable over the entire project; rather, it changes the range of the variable in the selected model. If you want to change the range of a variable in the entire session, change the range in the variable dictionary. For more information, see “Using the Variable Menu” on page 7-12.

7 To close the dialog box, click **Finish**.

Model Properties

Select **Model –> Properties** (or right-click) to view information about the selected model. This opens the **Model Properties** dialog where you can see the model type, definition, inputs, availability of PEV and constraints, creation date, user name, and toolbox version.

Model Properties: General



NOXFLOW_Model

Type: MBC model

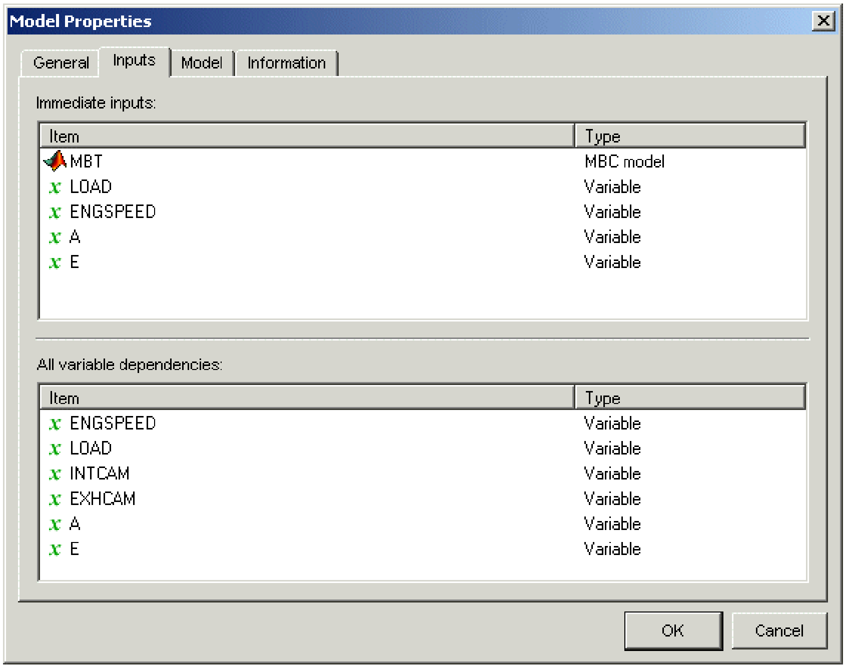
Number of inputs: 5

Constraints: Not available

Prediction error variance: Available

Here you can see the model type (such as MBC model or function model), the number of inputs, and the availability of constraints and Prediction Error.

Model Properties: Inputs



Immediate inputs:

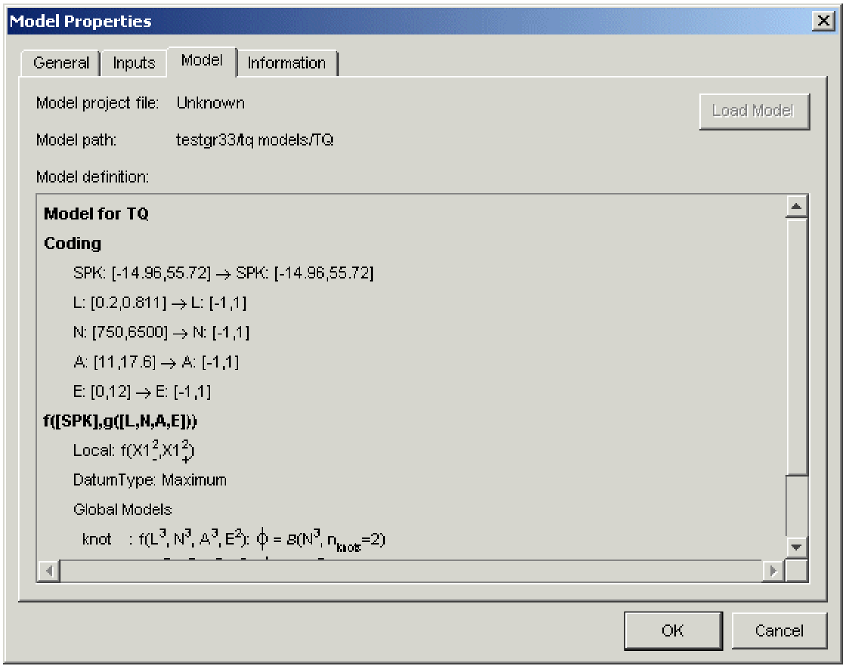
Item	Type
MBT	MBC model
LOAD	Variable
ENGSPEED	Variable
A	Variable
E	Variable

All variable dependencies:

Item	Type
ENGSPEED	Variable
LOAD	Variable
INTCAM	Variable
EXHCAM	Variable
A	Variable
E	Variable

Here you can view all the immediate inputs and variable dependencies of your model. For some models the two lists will be the same; in the example shown one of the inputs is another model (`MBT`) so the variable dependencies list also shows the variable inputs for that model. This information is shown graphically in the **Connections** pane.

Model Properties: Model



Model project file: Unknown

Model path: testgr33/tq models/TQ

Model definition:

Model for TQ

Coding

SPK: [-14.96,55.72] → SPK: [-14.96,55.72]

L: [0.2,0.811] → L: [-1,1]

N: [750,6500] → N: [-1,1]

A: [11,17.6] → A: [-1,1]

E: [0,12] → E: [-1,1]

f([SPK],g([L,N,A,E]))

Local: $f(X1^2_-, X1^2_+)$

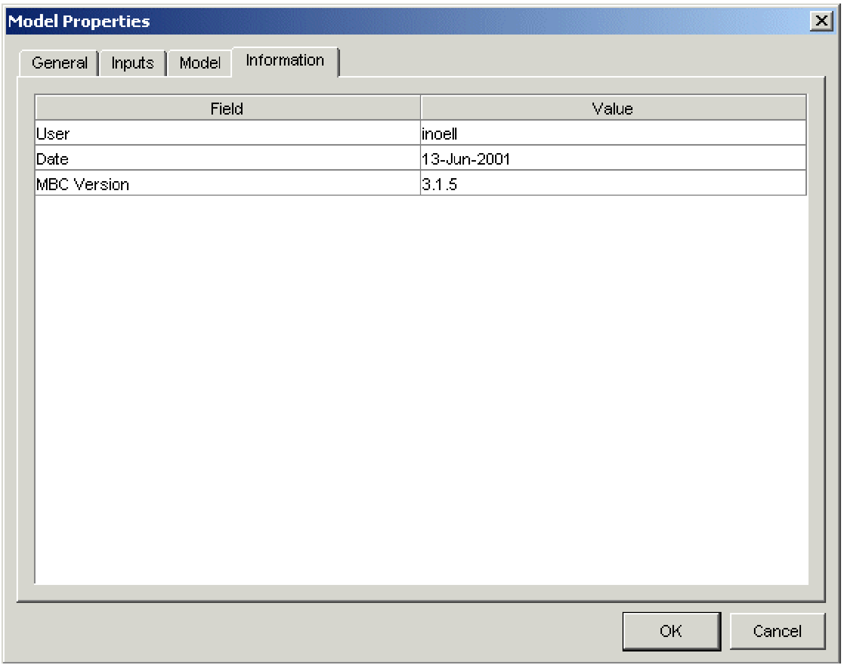
DatumType: Maximum

Global Models

knot : $f(L^3, N^3, A^3, E^2)$: $\phi = B(N^3, n_{knot}=2)$

Here you can view the model definition, the project file, and the model path. Function model definitions are shown here. For MBC models the model definition (showing the parameters and coefficients of the model formula) is the same information you would see in the Model Browser part of the toolbox when selecting **View –> Model Definition**.

Model Properties: Information



Field	Value
User	inoell
Date	13-Jun-2001
MBC Version	3.1.5

Here you can see the user name associated with the model, the date of creation and the version number of the Model-Based Calibration Toolbox used to create the model.

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**.

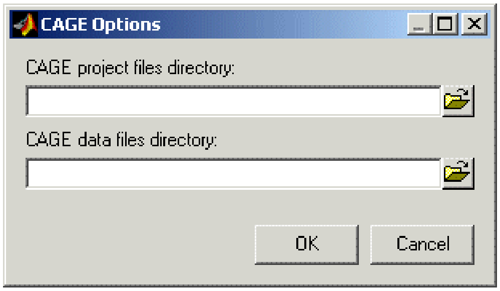
Project files have the file extension `.cag` and store entire CAGE sessions.

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**.

8

Normalizers

This section includes the following topics:

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.

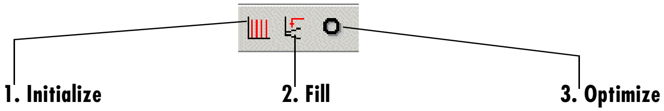
Calibrating the Normalizers

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 8-14.

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 8-10.

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.

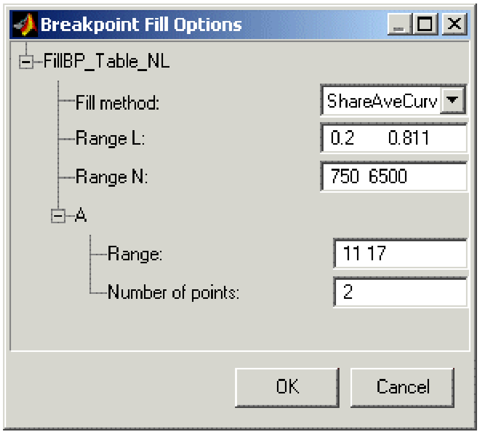
This option is only available when you are performing **Feature** calibrations.

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.



FillBP_Table_NL
Fill method: ShareAveCurv
Range L: 0.2 0.811
Range N: 750 6500
A
Range: 11 17
Number of points: 2

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 $MBT_{\mathrm{AV}}(N, L)$ 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 $MBT_{\mathrm{AV}}(N, L)$,

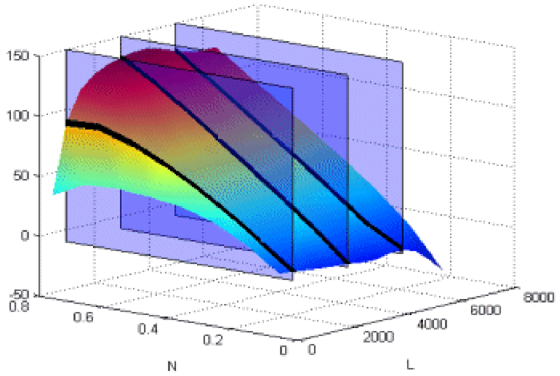
$$K = \sum_{i=1}^{\text{fine mesh}} (MBT_{AV}''(N, L))^{1/2}$$

as an approximation for

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

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 *L*, 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

$$\int_{750}^{N_i} |MBT_{AV}''(N, L)|^{1/2} dN = \frac{i-1}{j-1} \times K$$

Essentially this condition spaces out the breakpoints so that an equal amount of curvature (in an appropriate metric) occurs in each breakpoint interval. The breakpoint placement is optimal in the sense that the maximum error between the lookup table estimate and the model decreases with the optimal convergence rate of $O(N^{-2})$. This compares with an order of $O(N^{-1/2})$ for equally spaced breakpoints.

The theorem for determing the position of the unequally spaced breakpoints is from the field of Approximation Theory — page 46 of the following reference: de Boor, C., *A Practical Guide to Splines*, New York, Springer–Verlag, 1978.

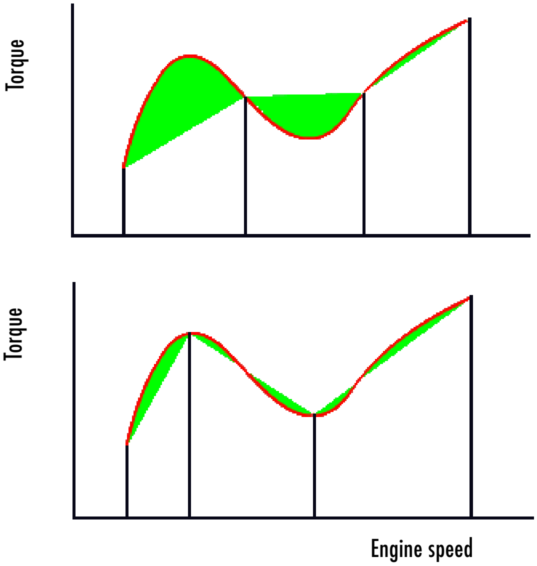
See Also

- "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 9-18.

See “Filling Methods” on page 8-6 for details on how the optimal breakpoint spacing is calculated.

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,

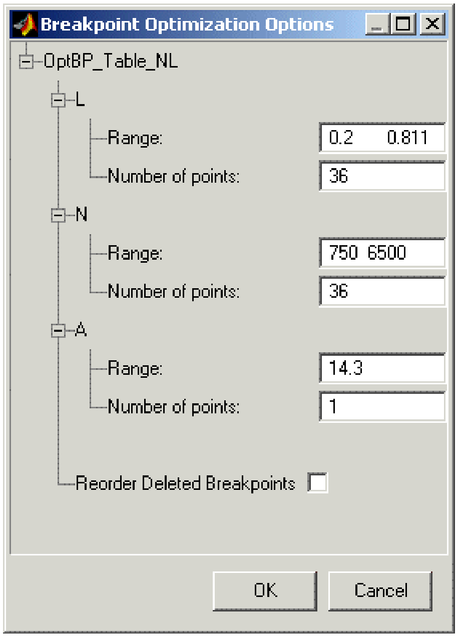
1 Ensure that the optimization routine works over reasonable values for the table by choosing one of these methods:

   a Select **Normalizer –> Initialize**.

   b Select **Normalizer –> Fill**.

2 Click O on the toolbar or select **Normalizer –> Optimize**.

   This opens the following dialog box.



Breakpoint Optimization Options

OptBP_Table_NL

L
Range: 0.2 0.811
Number of points: 36

N
Range: 750 6500
Number of points: 36

A
Range: 14.3
Number of points: 1

Reorder Deleted Breakpoints □

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 8-19.

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

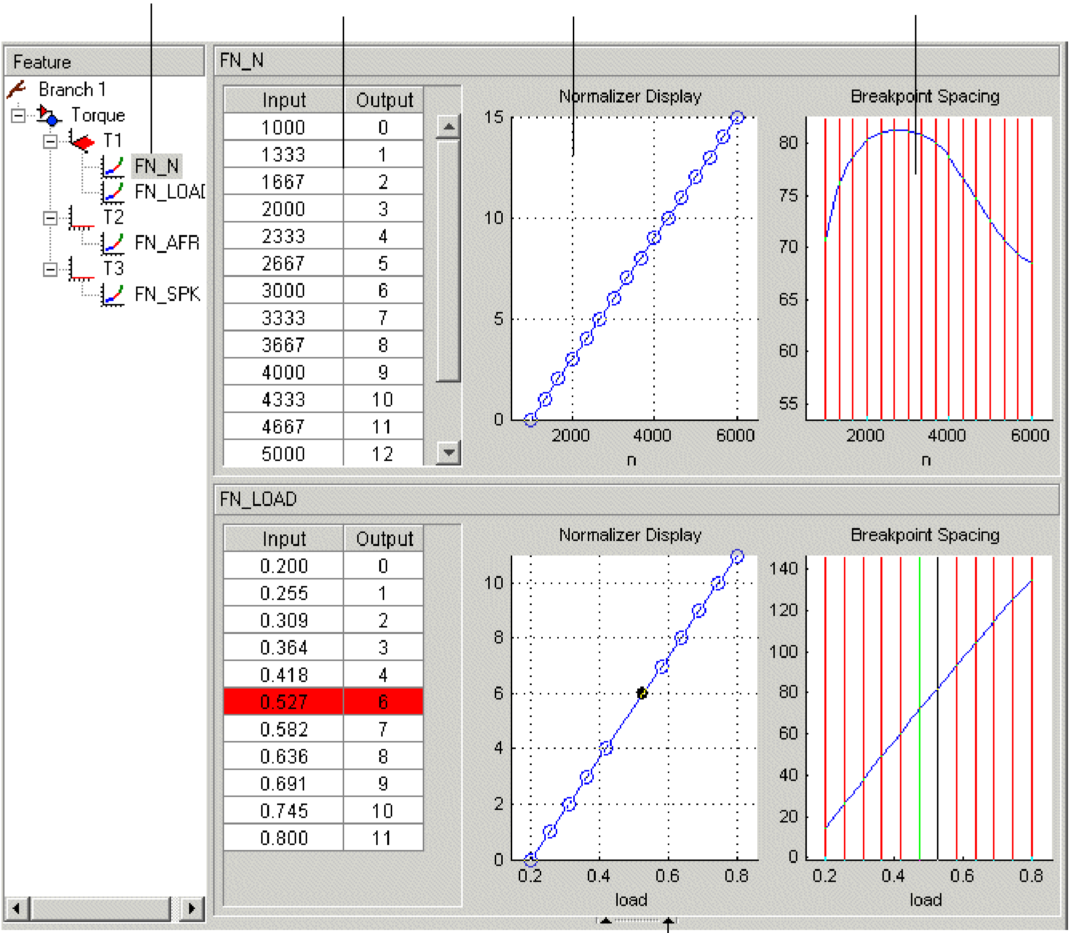
Normalizer View

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

4. To view the comparison pane

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

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 8-16.

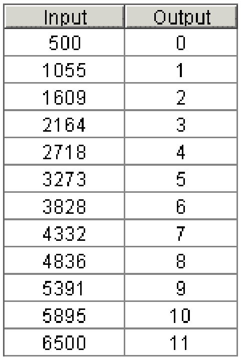
2. The **Normalizer Display** is a graphical representation of the **Input Output** display. For information, see “Normalizer Display” on page 8-17.

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 8-18.

4. The comparison pane. For information, see “Viewing the Comparison Pane” on page 8-21.

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 15-6.

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 breakpoint cells have padlock icons.

To unlock cells, follow the same procedure.

See Also

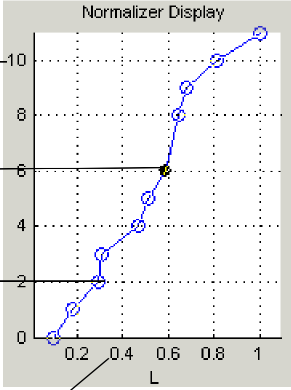
- "Normalizer View" on page 8-14

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



Values of the breakpoint labels.

A locked breakpoint

Breakpoint

Values of the breakpoints

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 8-14

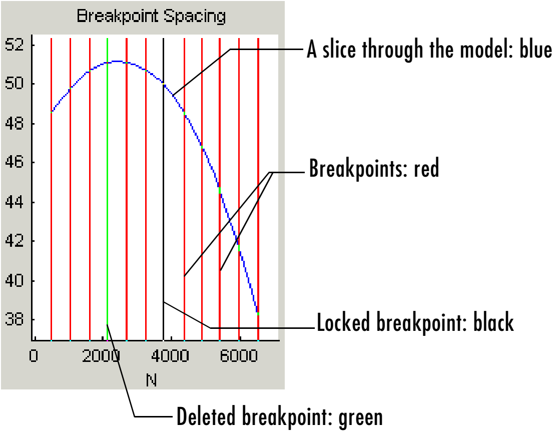
Breakpoint Spacing Display and Deleting Breakpoints

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.

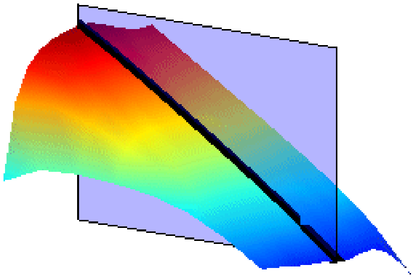
2 Select **Display –> Model Curvature**.

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

Multiple Slice View

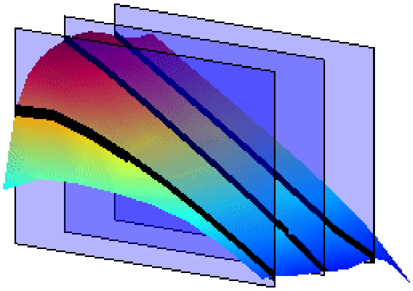
By default the **Breakpoint Spacing** display shows one slice through the model, shown.

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.

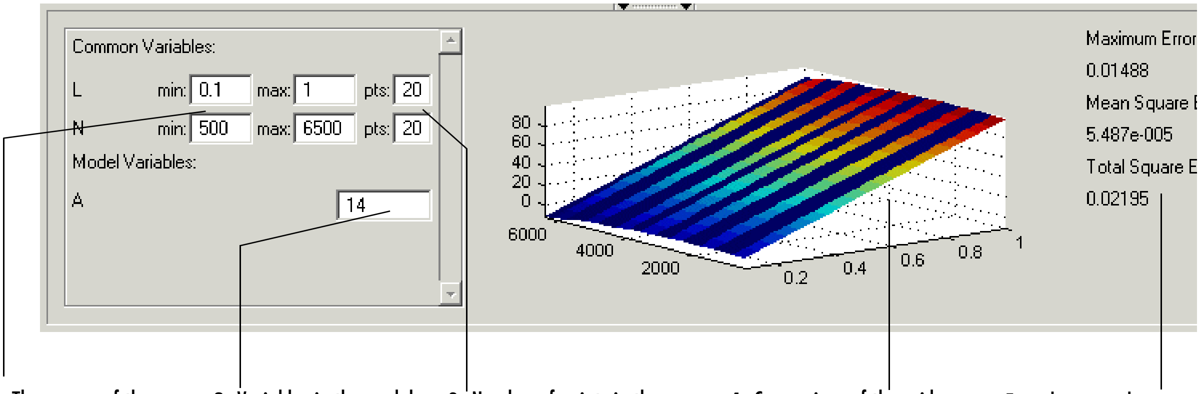
2 From the menu, select **Number of Lines** and choose the number of slices that you want to view from the list.

See Also

- "Normalizer View" on page 8-14

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 8-14.



Common Variables:

L min: 0.1 max: 1 pts: 20

N min: 500 max: 6500 pts: 20

Model Variables:

A 14

Maximum Error

0.01488

Mean Square E

5.487e-005

Total Square E

0.02195

1. The ranges of the variables common to the table and model

2. Variables in the model, not in the table

3. Number of points in the comparison display

4. Comparison of the grid and the model

5. Error between the table and the model

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 9-12.

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 7-10.

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

4 Check the comparison between the table and the model.

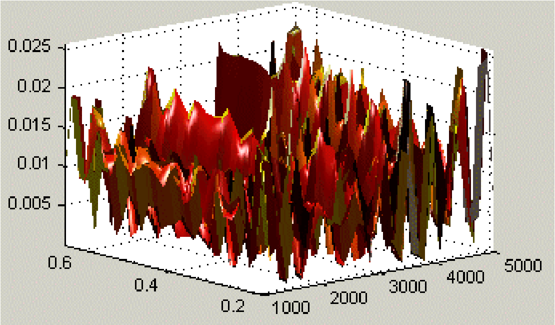
Right-click the comparison graph to view the error display.

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 8-14
- "Comparing the Strategy and the Model" on page 9-36

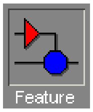
  This describes the comparison made when a table node is selected in the branch display.

9

Feature Calibrations

This section includes the following topics:

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 7-7.)

2 Set up the model or models. (See "Setting Up Your Models" on page 7-14.)

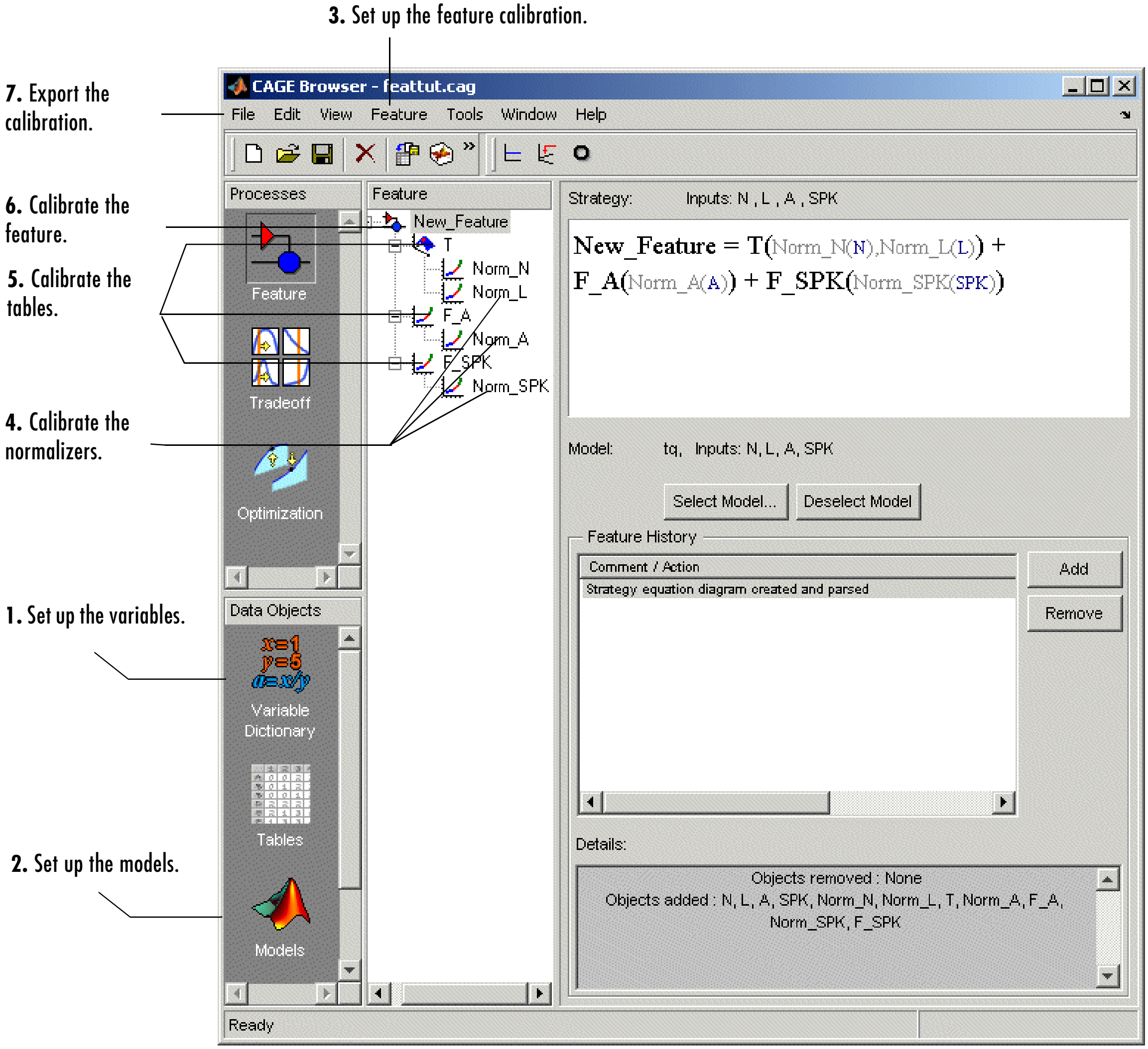
3 Set up the feature calibration. (See "Setting Up a Feature Calibration" on page 9-5.)

4 Calibrate the normalizers. (See "Calibrating the Normalizers" on page 8-3.)

5 Calibrate the tables. (See "Calibrating the Tables" on page 9-12.)

6 Calibrate and view the entire feature. (See “Calibrating the Feature Node” on page 9-39.)

7 Export the normalizers, tables, and features. (See “Exporting Calibrations” on page 7-27.)



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.

Strategy: Inputs: N , L , A , SPK

New_Feature = T(Norm_N(N),Norm_L(L)) + F_A(Norm_A(A)) + F_SPK(Norm_SPK(SPK))

Model: tq, Inputs: N, L, A, SPK

Comment / Action
Strategy equation diagram created and parsed

Objects removed : None
Objects added : N, L, A, SPK, Norm_N, Norm_L, T, Norm_A, F_A, Norm_SPK, F_SPK

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 9-5.

2. Assign a model. This is described in “Assigning a Model” on page 9-6.

3. Set up your strategy. This is described in “Setting Up Your Strategy” on page 9-6.

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. A dialog box appears.

3 Highlight the correct model to assign to your feature and click **OK**. You will see the model name and inputs appear above the **Select Model** button.

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 9-7
- “Constructing a Strategy” on page 9-8
- “Exporting Strategies” on page 9-11

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 outport.

4 Select the outport that you want to use.

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

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

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

Or

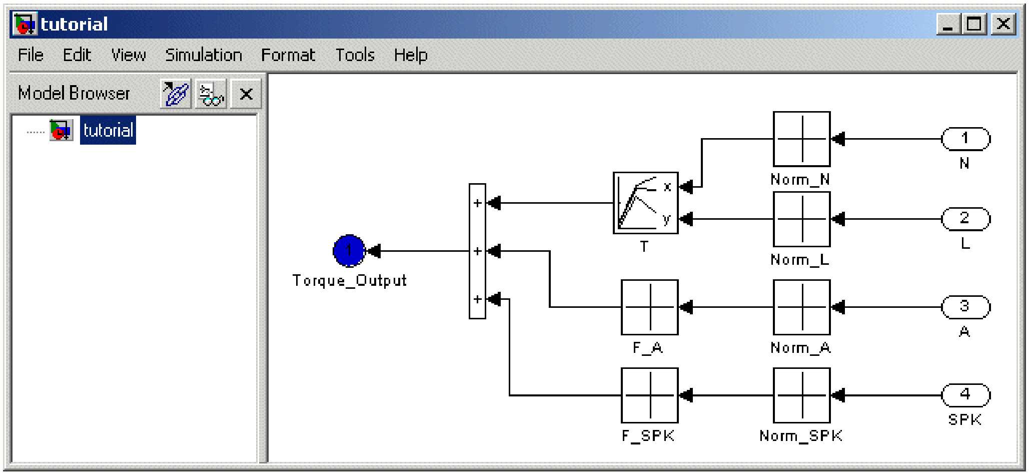
b Select **Manual** to edit the strategy. Double-click the blue outport circle to parse the strategy to your feature.

Note When you double-click the blue outport, 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. Select **View –> Full Strategy Display** to switch between the full description and the simplified expression. You can select and copy the strategy equation to the clipboard.

For information about using Simulink to amend strategies, see “Constructing a Strategy” on page 9-8.

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` outport 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.

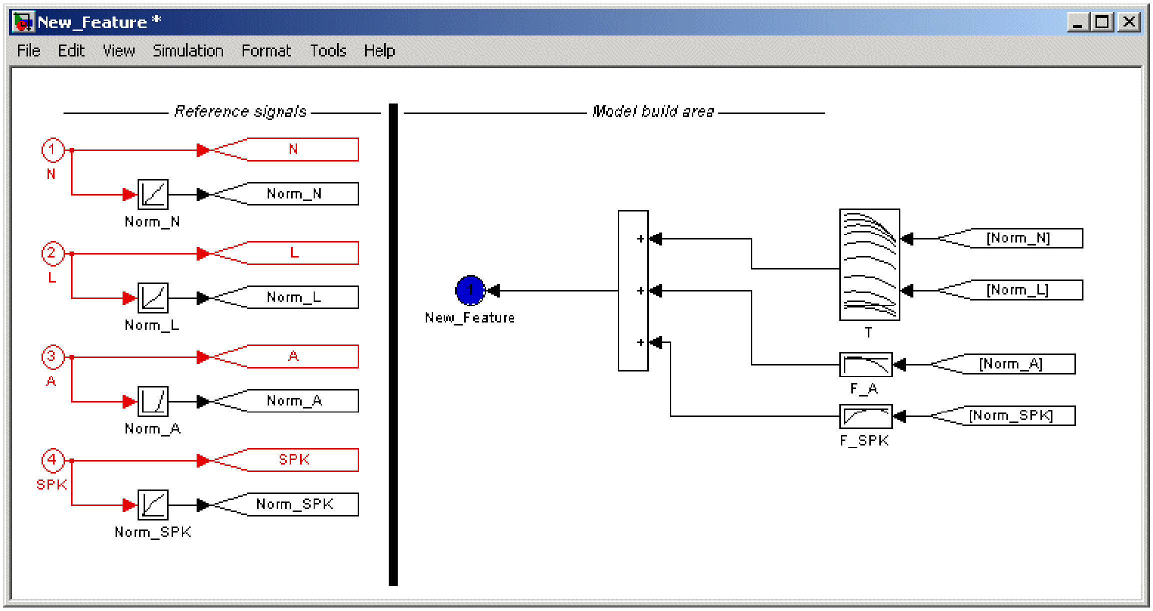
To construct a strategy using Simulink,

1 Highlight the correct feature by clicking the **Feature** node.

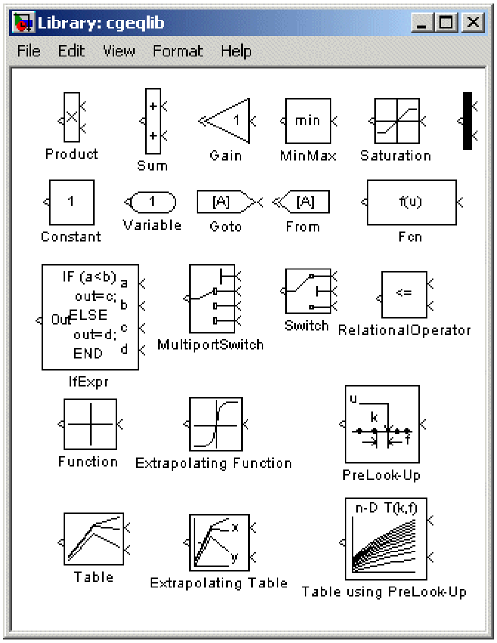
2 Select **Feature –> Graphical Strategy Editor** or press **Ctrl+E**.

Three Simulink windows open:

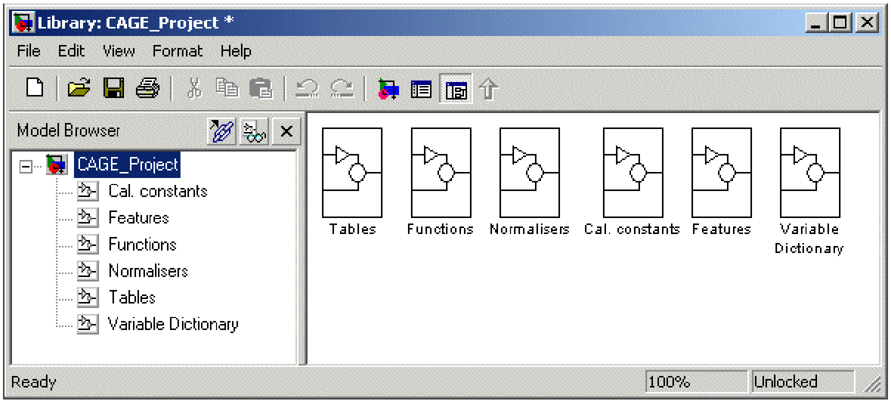
- The strategy window for editing your strategy, like the following example.



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



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



3 In the strategy window, build your strategy using the blocks available in the library windows.

4 Double-click the blue outport 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.

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 9-31.

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.

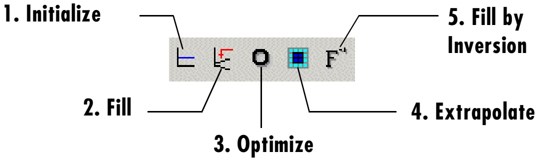
For example, a simple MBT feature has two tables:

- 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 9-13.

- **Fill**

  Fills the table values by reference to the model. For information, see “Filling Table Values” on page 9-14.
- **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 9-18.
- **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 9-21.
- **Fill by Inversion**

  Fills the table by creating an inversion of another table. For information, see “Inverting a Table” on page 9-23.

The next sections describe each of these toolbar options in detail. See the “Table Menu Options” on page 9-34 for other options.

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 9-15.

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**.

A dialog appears where you can choose the values of other input variables not specified by the normalizers. You need to specify the values of other inputs so CAGE can evaluate the model to fill the table. You can also just click **OK** to accept the default values. These defaults are taken from the set points for each variable, found in the Variable Dictionary.

For example, if your normalizers are in speed and load, and the model also requires an AFR input, you need to specify AFR. 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). See the following section “How CAGE Fills Tables” on page 9-15 for more detailed explanation.

To fill a table 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.

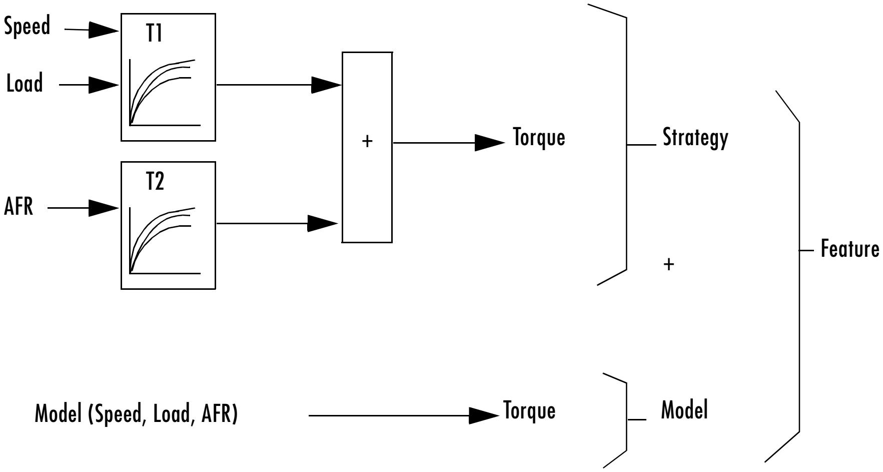
How CAGE Fills Tables

CAGE fills tables in a feature calibration by rearranging the equation model = 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.

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

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

$$Model \approx T1 + T2$$

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

$$T1 = Model - T2$$

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

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

$$T1 = Model$$

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):

$$T2(A) = Model(N,L,A) - T1(N,L)$$

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:

$$T2(12.5) = Model(3000, 0.4, 12.5) - T1(3000, 0.4)$$

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 $\mathrm{Model} \approx \mathrm{T1} \times \mathrm{T2}$ for T1 gives $\mathrm{T1} \approx \mathrm{Model}/\mathrm{T2}$, and you cannot divide by zero. This operation will fail.

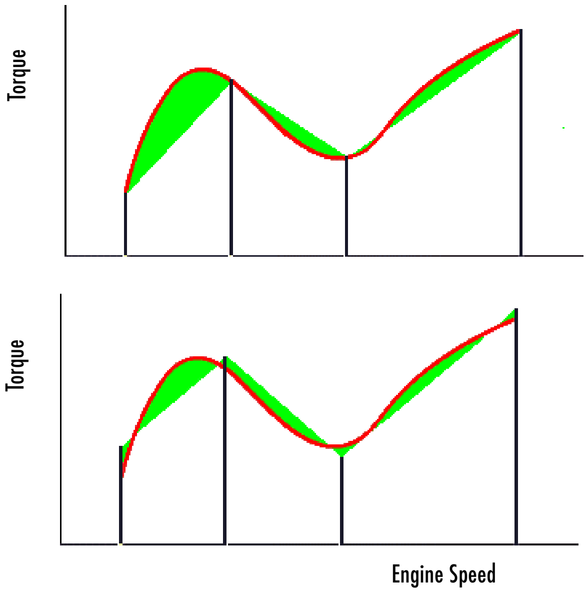
Solving $\mathrm{Model} \approx \mathrm{Strategy}$ algebraically for a table in the strategy is not always possible. In these cases you must use optimization.

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.



This shows the error when filling the table using model values.

This shows the reduced error after optimizing 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 8-10.

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.

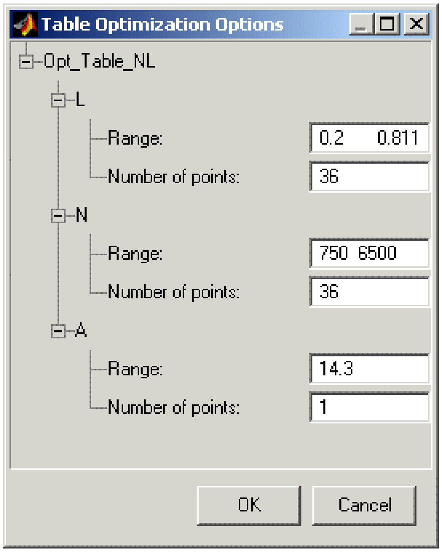


Table Optimization Options

Opt_Table_NL

L

Range: 0.2 0.811

Number of points: 36

N

Range: 750 6500

Number of points: 36

A

Range: 14.3

Number of points: 1

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 9-12

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 Mask** or **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 yellow and have a padlock icon.

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,

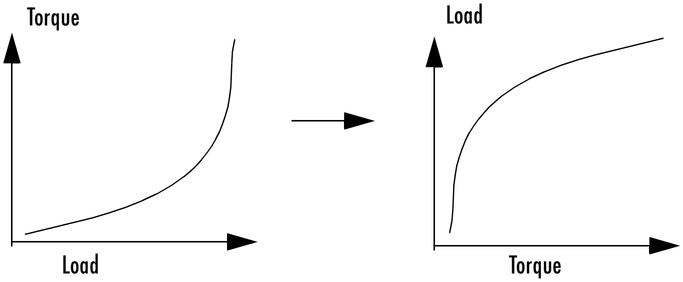
1 Select **Table –> Generate Mask From PE**.

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

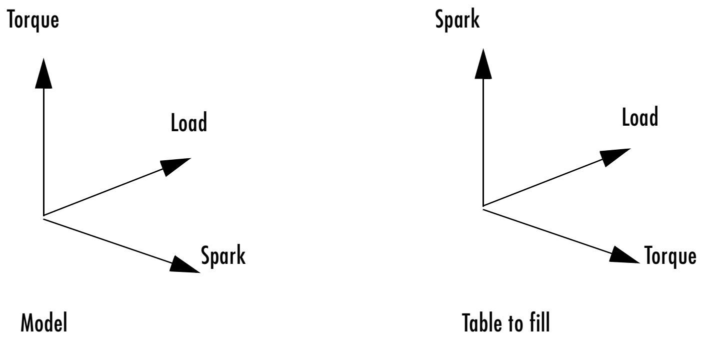
You can use CAGE to produce a table that is the inverse of another table. This involves swapping a table input with a table output, and you can invert 1-D or 2-D tables.



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.

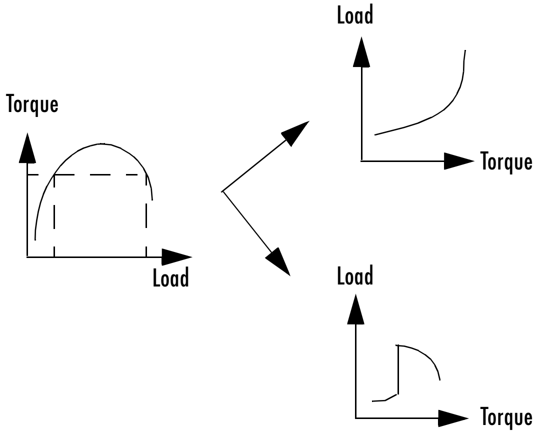
Normally you fill tables in CAGE by comparing with data or models. Ideally you want to fill using the correct strategy, but that might not be possible to find or measure. If you only have a forward strategy but want a backward one, you can fill using the forward strategy (tables or model) and then invert 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 F^{-1} 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).

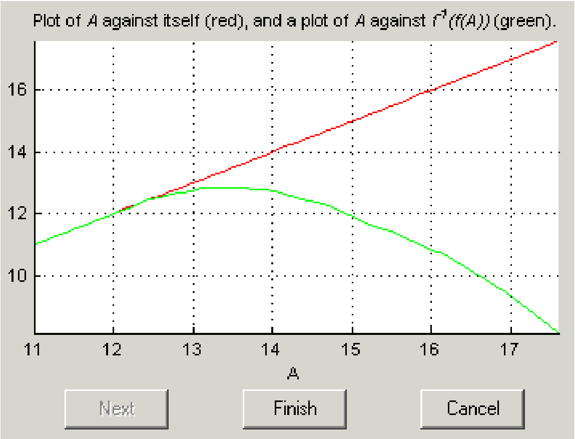
For example, the function $y = x^2$ is impossible to invert over the range -1 to 1. You can specify to invert the range from 0 to 1, sacrificing the inversion in the lower range, or the reverse. To select the range from 0 to 1, highlight **Maximum**.

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


Plot of A against itself (red), and a plot of A against f⁻¹(f(A)) (green).
16
14
12
10
11
12
13
14
15
16
17
A
Next
Finish
Cancel

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^{-1} 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**.

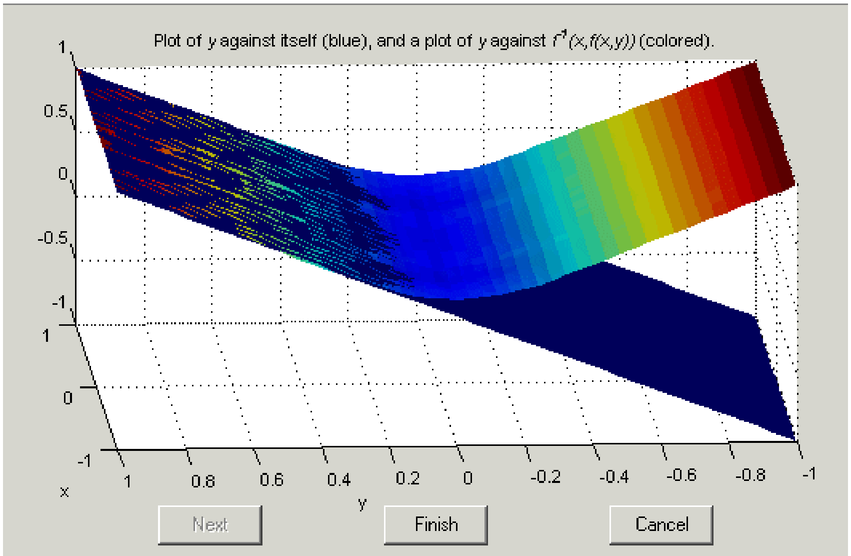
CAGE asks what to do if it encounters multiple values. The choices are

- **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 9-25).

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.


Plot of y against itself (blue), and a plot of y against f⁻¹(x,f(x,y)) (colored).
1
0.5
0
-0.5
-1
1
0
-1
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
y
Next
Finish
Cancel

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:

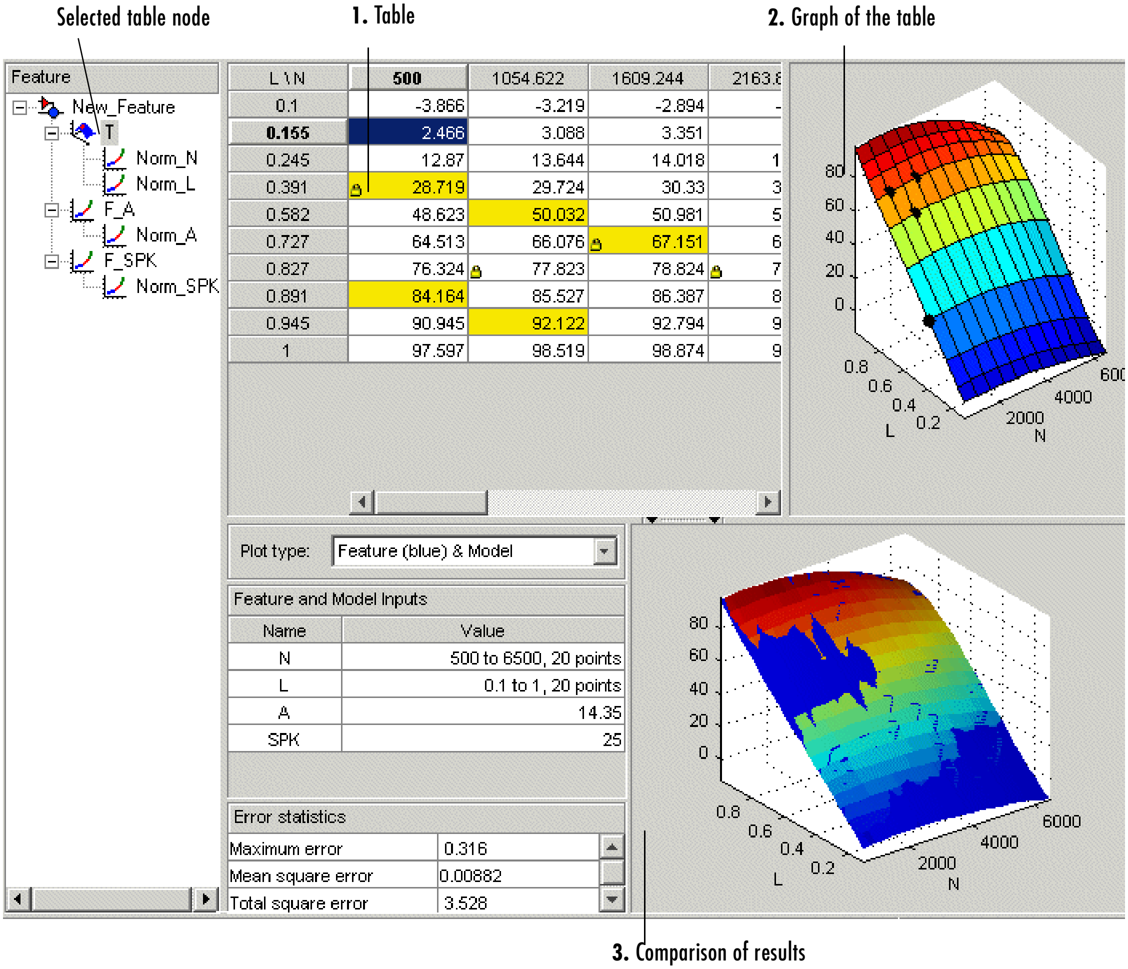
- A two-dimensional table with speed and relative air charge as its normalizers
- A one-dimensional table with AFR as its normalizer

For feature calibration (filling the tables by comparing a strategy and a model), see “Calibrating the Tables” on page 9-12.

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 the Tables view for manual calibration, you do not see the comparison pane because you are not comparing tables with a model.

Table Display in Feature Calibration



L \ N	500	1054.622	1609.244	2163.8
0.1	-3.866	-3.219	-2.894	-
0.155	2.466	3.088	3.351	
0.245	12.87	13.644	14.018	1
0.391	28.719	29.724	30.33	3
0.582	48.623	50.032	50.981	5
0.727	64.513	66.076	67.151	6
0.827	76.324	77.823	78.824	7
0.891	84.164	85.527	86.387	8
0.945	90.945	92.122	92.794	9
1	97.597	98.519	98.874	9

Plot type: Feature (blue) & Model

Feature and Model Inputs

Name	Value
N	500 to 6500, 20 points
L	0.1 to 1, 20 points
A	14.35
SPK	25

Error statistics

Maximum error	0.316
Mean square error	0.00882
Total square error	3.528

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 9-33.)

2 The graph of the table pane displays the table values graphically. (See “Using the Graph of the Table” on page 9-35.)

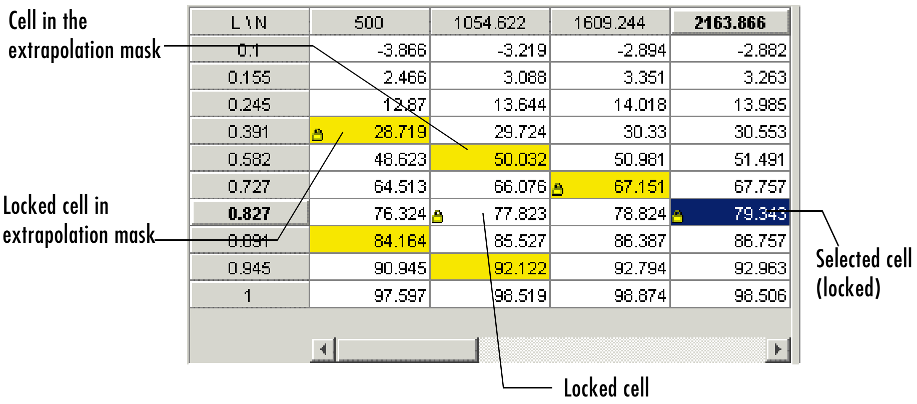
3 The comparison-of-results pane displays a comparison between the current output of the strategy and the feature model. (See “Comparing the Strategy and the Model” on page 9-36.)

Note You can view the **History** display by selecting **View –> History**. For information, see “Using the History Display” on page 15-6.

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.



L \ N	500	1054.622	1609.244	**2163.866**
0.1	-3.866	-3.219	-2.894	-2.882
0.155	2.466	3.088	3.351	3.263
0.245	12.87	13.644	14.018	13.985
0.391	🔒 28.719	29.724	30.33	30.553
0.582	48.623	50.032	50.981	51.491
0.727	64.513	66.076	🔒 67.151	67.757
0.827	76.324	🔒 77.823	78.824	🔒 79.343
0.891	84.164	85.527	86.387	86.757
0.945	90.945	92.122	92.794	92.963
1	97.597	98.519	98.874	98.506

Cell in the extrapolation mask

Locked cell in extrapolation mask

Selected cell (locked)

Locked cell

To edit a value in the table, double-click the cell. Selected cells are outlined (or blue with a padlock icon if they are locked).

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 from the menu. Locked cells have a padlock icon in the display. You can also lock an entire table using the **Table** menu.

Table Menu Options

All the toolbar button functions are also found in the table menu: **Initialize**, **Fill**, **Optimize**, **Extrapolate**, **Fill by Inversion**. For information on these see “Calibrating the Tables” on page 9-12

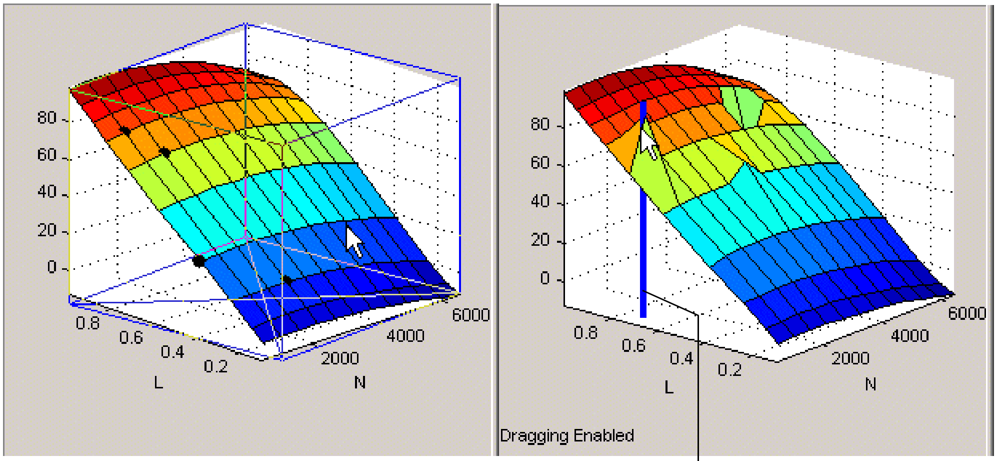
The **Table** menu contains the following other options

- **Adjust Cell Values**. This opens a dialog where you can specify an arithmetic operation to apply to either the whole table or only the cells currently selected. Arguments to operations can be numeric (plus 10) or percentages (minus 5%). You can set the selected cells to a value or to the mean. You can also apply user-defined functions. Also in the table context menu.
- **Extrapolation Mask** The following items are also in the table context menu:
  - **Add Selection** — Adds selected cells to the extrapolation mask.
  - **Remove Selection** — Removes selected cells from the extrapolation mask.
  - **Clear Mask** — This ensures that none of the cells are in the extrapolation mask.
  - **Generate From PE** — Generate extrapolation mask depending on the value of prediction error (PE). Only available for tables in feature calibration, as you must have a model to calculate PE. A dialog opens where you can specify the threshold value of PE below which you want to include cells in the mask. The dialog contains information about the range and mean of prediction error for the model to help you select a threshold.
  - **Generate From Boundary Model** — Generate extrapolation mask to include only cells within the boundary model. Only available for tables in feature calibration, as you must have a boundary model.
- **Extrapolate** — Extrapolates values from the cells in the extrapolation mask to fill the whole table. Also in the toolbar.
- **Table Cell Locks** The following items are also in the table context menu:
  - **Lock Selection** — Locks the selected cells and a padlock icon appears..
  - **Unlock Selection** — Unlocks the selected cells.
  - **Lock Entire Table** — Locks every cell in the current table.
  - **Clear All Locks** — Unlocks all cells in the table.

- **Convert to Model**. This option converts a table directly to a model.
- **Properties**. This opens the **Table Properties** dialog where you can set the precision type of the table data. You can also reach this from the Calibration Manager. See “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 in dragging and rotation mode.



Line indicates which value in the table you are editing.

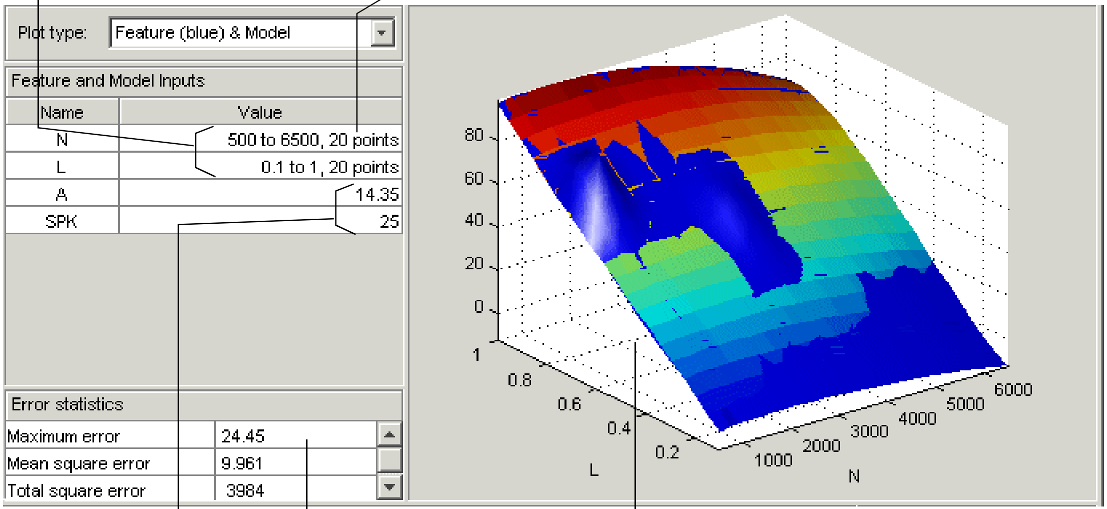
- In the default mode, you can rotate the graph of the table by clicking and dragging the axes.
- Select **View –> Edit Table Surface** to alter values in the table by clicking and dragging vertically any point. In this mode, when you click a point, a blue line indicates the selected point in the table. To return to table rotation mode without altering table values, select **View –> Rotate Table Surface**.

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.

The ranges of the common variables

Number of points in the comparison display



Plot type: Feature (blue) & Model

Feature and Model Inputs

Name	Value
N	500 to 6500, 20 points
L	0.1 to 1, 20 points
A	14.35
SPK	25

Error statistics

Maximum error	24.45
Mean square error	9.961
Total square error	3984

Variables in the model, not in the table

Error between the strategy and the model

Comparison of the strategy and the model

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. For each variable check the number of points at which the display is calculated. Double-click to edit any variable range or number of points.

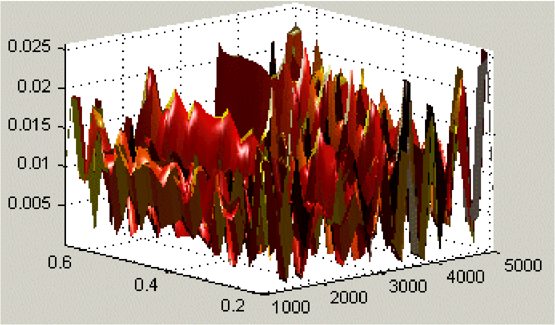
2 Check the values selected for any variables in the model that are not in the selected table. The default for this is the set point of the variable's range. Double-click to edit.

3 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.

4 Use the **Plot Type** drop-down menu to display the error statistics for the comparison.

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, select one of the **Error** options from the **Plot Type** drop-down 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 (Feature-Model)**. 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.

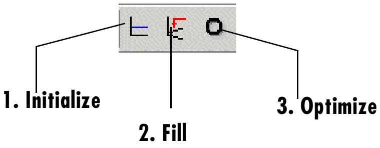
- **Relative Error (%)**. This is the error as a percentage of the value of the model.
- **Absolute Relative Error (%)**. This is the absolute value of the relative error.

Calibrating the Feature Node

Selecting a **Feature** node displays the **Feature** view. For more information about the **Feature** view, see “Feature View” on page 9-47.

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 9-39
2. “Filling the Feature” on page 9-41
3. “Optimizing the Feature” on page 9-44

Initializing the Feature

For example, a simple feature for maximum brake torque (MBT) consists of the following tables:

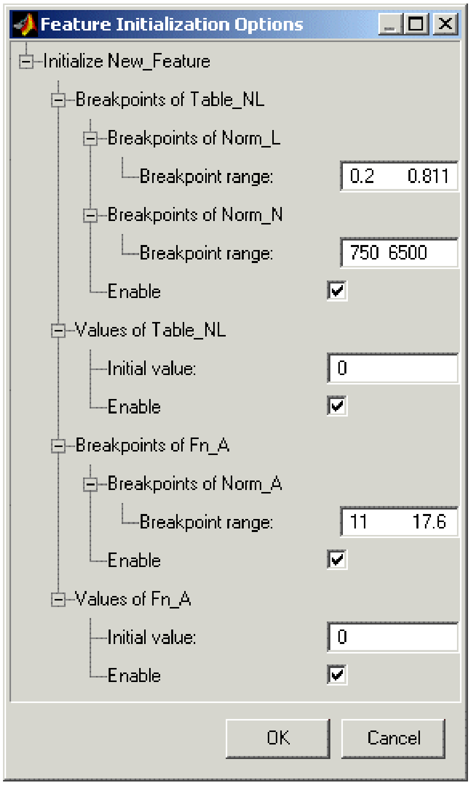
- A speed (*N*), load (*L*) table
- A table to account for the behavior of air/fuel ratio (*A*)

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.



Feature Initialization Options

- Initialize New_Feature
  - Breakpoints of Table_NL
    - Breakpoints of Norm_L
      - Breakpoint range: 0.2 0.811
    - Breakpoints of Norm_N
      - Breakpoint range: 750 6500
    - Enable ☑
  - Values of Table_NL
    - Initial value: 0
    - Enable ☑
  - Breakpoints of Fn_A
    - Breakpoints of Norm_A
      - Breakpoint range: 11 17.6
    - Enable ☑
  - Values of Fn_A
    - Initial value: 0
    - Enable ☑

2 Enter the ranges for the breakpoints in your normalizers. In the preceding example, these are the 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, the cell values are

- **Table_NL** has initial value 0.
- **Fn_A** has initial value 0.

4 Click **OK** to initialize the feature.

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

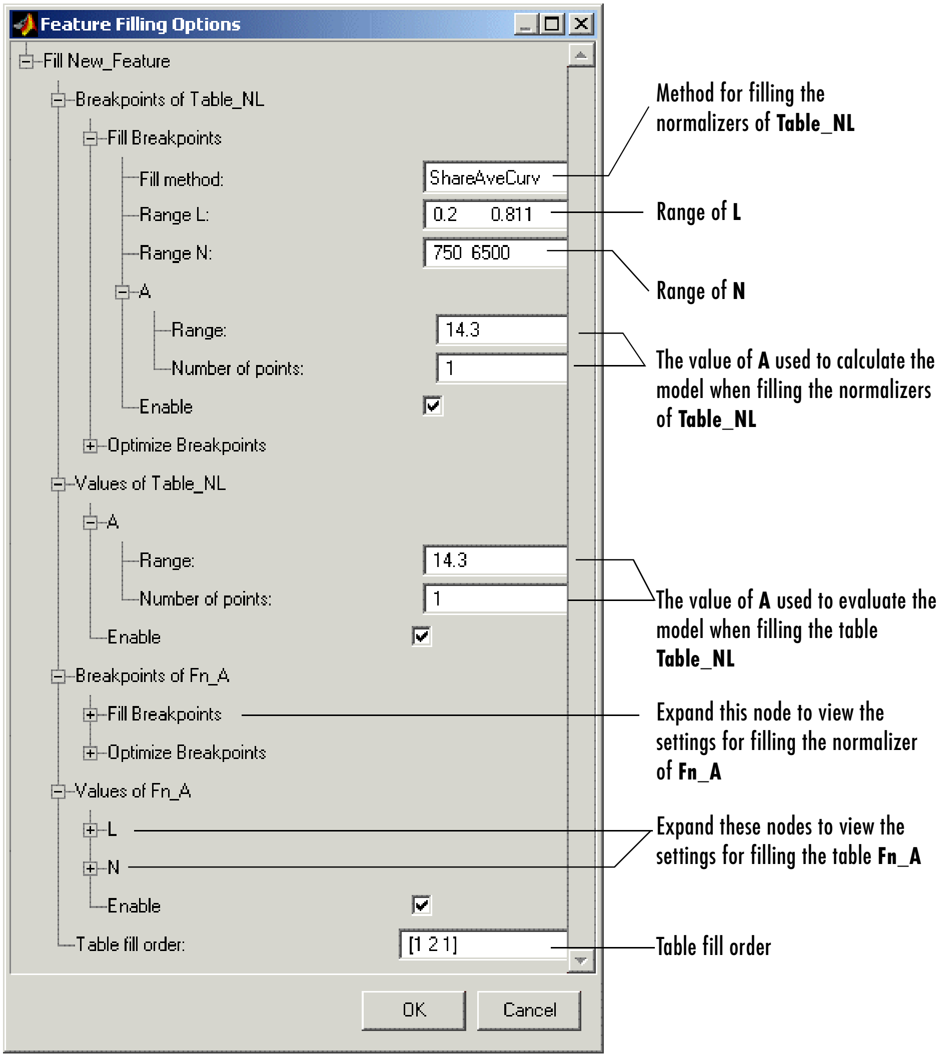
A very quick way to calibrate a feature is to fill it. This does two things:

- 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 9-14.

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.



- Fill New_Feature
  - Breakpoints of Table_NL
    - Fill Breakpoints
      - Fill method: ShareAveCurv
      - Range L: 0.2 0.811
      - Range N: 750 6500
      - A
        - Range: 14.3
        - Number of points: 1
      - Enable ☑
    - Optimize Breakpoints
  - Values of Table_NL
    - A
      - Range: 14.3
      - Number of points: 1
    - Enable ☑
  - Breakpoints of Fn_A
    - Fill Breakpoints
    - Optimize Breakpoints
  - Values of Fn_A
    - L
    - N
    - Enable ☑
  - Table fill order: [1 2 1]

Method for filling the normalizers of **Table_NL**

Range of **L**

Range of **N**

The value of **A** used to calculate the model when filling the normalizers of **Table_NL**

The value of **A** used to evaluate the model when filling the table **Table_NL**

Expand this node to view the settings for filling the normalizer of **Fn_A**

Expand these nodes to view the settings for filling the table **Fn_A**

Table fill order

2 Select the correct method for filling your normalizers. See “Filling Methods” on page 8-6.

3 Enter the ranges for the normalizers of the tables. The default is the range of the variable.

4 Enter the ranges of the other variables when filling the normalizers.

5 Enter the ranges of the other variables when filling the tables.

   As when filling individual tables, you can choose one value of each variable for the normalizer or table (as shown for A in the example above), or you can choose a range of values and fill using the average model value at each cell. The default is the set point of each variable. See “Filling Table Values” on page 9-14.

6 Enter the table fill order.

   In the example shown this is `[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** to fill the feature.

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.

See also “Optimizing the Feature” on page 9-44 to further improve the fit of the strategy to the model.

When you have completed a calibration, you can export your feature. For information, see “Exporting Calibrations” on page 7-27.

Optimizing the Feature

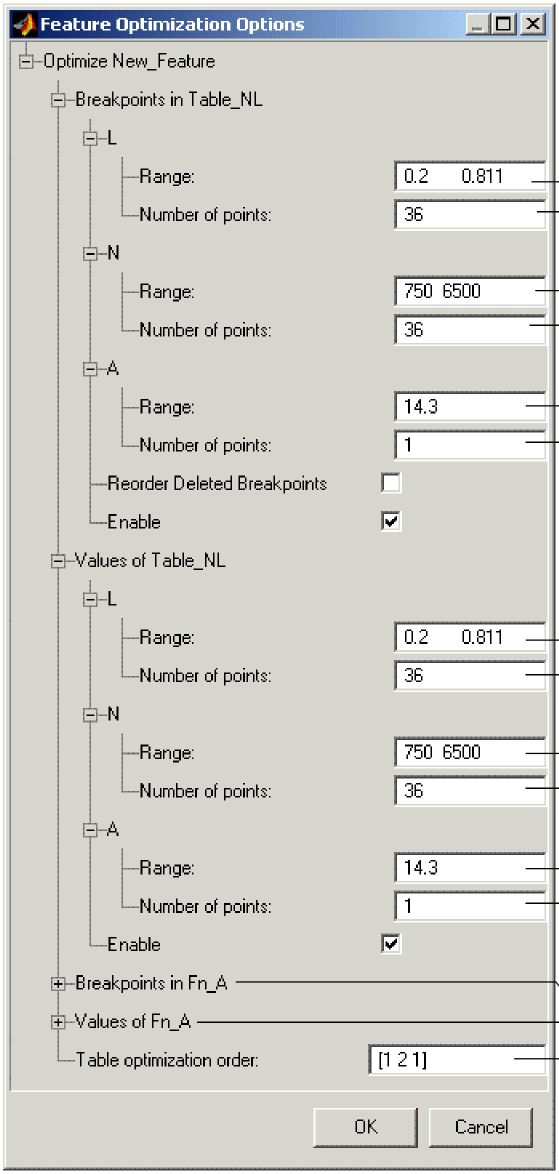
After filling the feature, you can improve the fit of the strategy to the model by optimizing the feature. The optimization routine does the following:

- First CAGE optimizes the breakpoints for the normalizers. (See “Optimizing Breakpoints” on page 8-10.)
- Next CAGE optimizes the values of the tables. (See “Optimizing Table Values” on page 9-18.)

This section gives the procedure for optimizing a feature. For further details of how optimization works, see the references given above. Note that to give the optimization a useful starting point, you should fill the feature before optimizing. Using **Fill** spaces the breakpoints of the normalizers by reference to the model, and places model values at the appropriate operating points into each cell of the tables. From this point, the optimization routine has the best chance of finding a good solution quickly; otherwise it can be very time consuming. See “Filling the Feature” on page 9-41.

To optimize a feature,

1 Click ○. This opens the **Feature Optimizing Options** dialog box, as shown.



Feature Optimization Options

- Optimize New_Feature
  - Breakpoints in Table_NL
    - L
      - Range: 0.2 0.811
      - Number of points: 36
    - N
      - Range: 750 6500
      - Number of points: 36
    - A
      - Range: 14.3
      - Number of points: 1
    - Reorder Deleted Breakpoints □
    - Enable ☑
  - Values of Table_NL
    - L
      - Range: 0.2 0.811
      - Number of points: 36
    - N
      - Range: 750 6500
      - Number of points: 36
    - A
      - Range: 14.3
      - Number of points: 1
    - Enable ☑
  - Breakpoints in Fn_A
  - Values of Fn_A
  - Table optimization order: [1 2 1]

Options for Optimizing Normalizers of Table_NL

Range of normalizer **L**

Number of grid points for normalizer **L**

Range of normalizer **N**

Number of grid points for normalizer **N**

The value of **A** used to evaluate the model when optimizing the normalizers of **Table_NL**

Options for Optimizing Table Values of Table_NL

Range and number of grid points for normalizer **L**, used for optimizing **Table_NL**

Range and number of grid points for normalizer **N**, used for optimizing **Table_NL**

The value of **A** used to evaluate the model when optimizing **Table_NL**

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 fill order

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.

As when optimizing individual tables, this information applies to optimizing a whole feature at once:

- 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 "Optimizing Table Values" on page 9-18 and "Optimizing Breakpoints" on page 8-10.

When you have completed a calibration, you can export your feature. For information, see "Exporting Calibrations" on page 7-27.

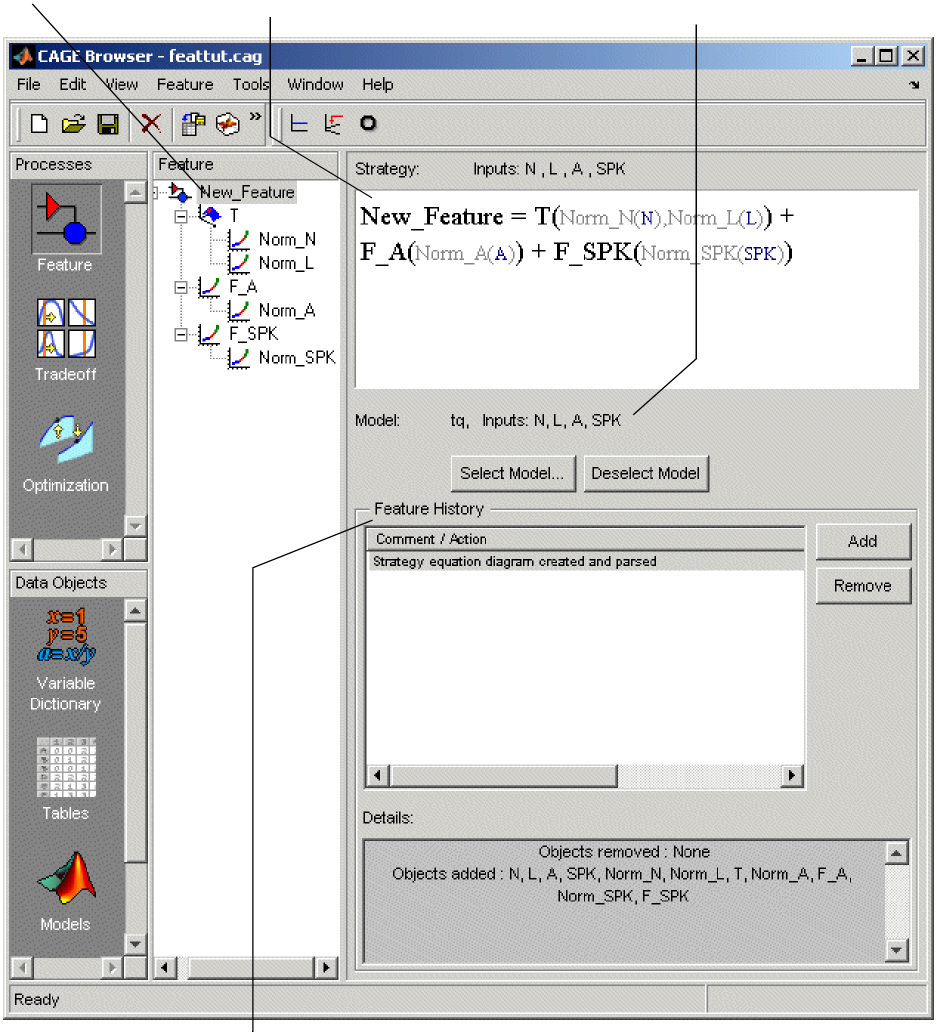
Feature View

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



Strategy: Inputs: N , L , A , SPK

New_Feature = T(Norm_N(N),Norm_L(L)) + F_A(Norm_A(A)) + F_SPK(Norm_SPK(SPK))

Model: tq, Inputs: N, L, A, SPK

Feature History

Comment / Action
Strategy equation diagram created and parsed

Details:

Objects removed : None
Objects added : N, L, A, SPK, Norm_N, Norm_L, T, Norm_A, F_A, Norm_SPK, F_SPK

Ready

3. Feature History pane

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 to Model**

  Takes the current feature and converts it to a model, which you can view by clicking the **Model** button.
- **Graphical Strategy Editor**

  Opens your current strategy for editing. For more information, see "Setting Up Your Strategy" on page 9-6.
- **Parse Strategy Diagram**

  Performs the same function as double-clicking the blue outport of your strategy diagram. For more information, see "Setting Up Your Strategy" on page 9-6.

  Enables you to view the feature and the model using the surface viewer. For information, see "Surface Viewer" on page 14-1.
- **Clear Current Strategy**

  Clears the current strategy from your feature.
- **Initialize**

  Initializes the feature; also in the toolbar. See "Initializing the Feature" on page 9-39 for details.

- **Fill**

  Fills the feature; also in the toolbar. See “Filling the Feature” on page 9-41 for details.

- **Optimize**

  Optimizes the feature; also in the toolbar. See “Optimizing the Feature” on page 9-44 for details.

10

Tradeoff Calibrations

This section includes the following topics:

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 7-7.)
2. Set up the model or models. (See “Setting Up Your Models” on page 7-14.)
3. Set up the tradeoff calibration. (See “Setting Up a Tradeoff Calibration” on page 10-5.)
4. Calibrate the tables. (See “Calibrating Tables in a Tradeoff Calibration” on page 10-11.)
5. Export the normalizers, tables, and tradeoffs. (See “Exporting Calibrations” on page 7-27.)

You can also use regions to enhance your calibration. (See “Using Regions” on page 10-22.)

See Also

- “Tutorial: Tradeoff Calibration” on page 3-1

  This is a tutorial giving an example of how to set up and complete a simple example tradeoff calibration.
- “Automated Tradeoff” on page 11-35 is a guide to using the optimization functionality in CAGE for tradeoffs.

3. Set up the tradeoff calibration.

5. Export the calibration.

4. Calibrate the tables.

1. Set up the variables.

2. Set up the models.

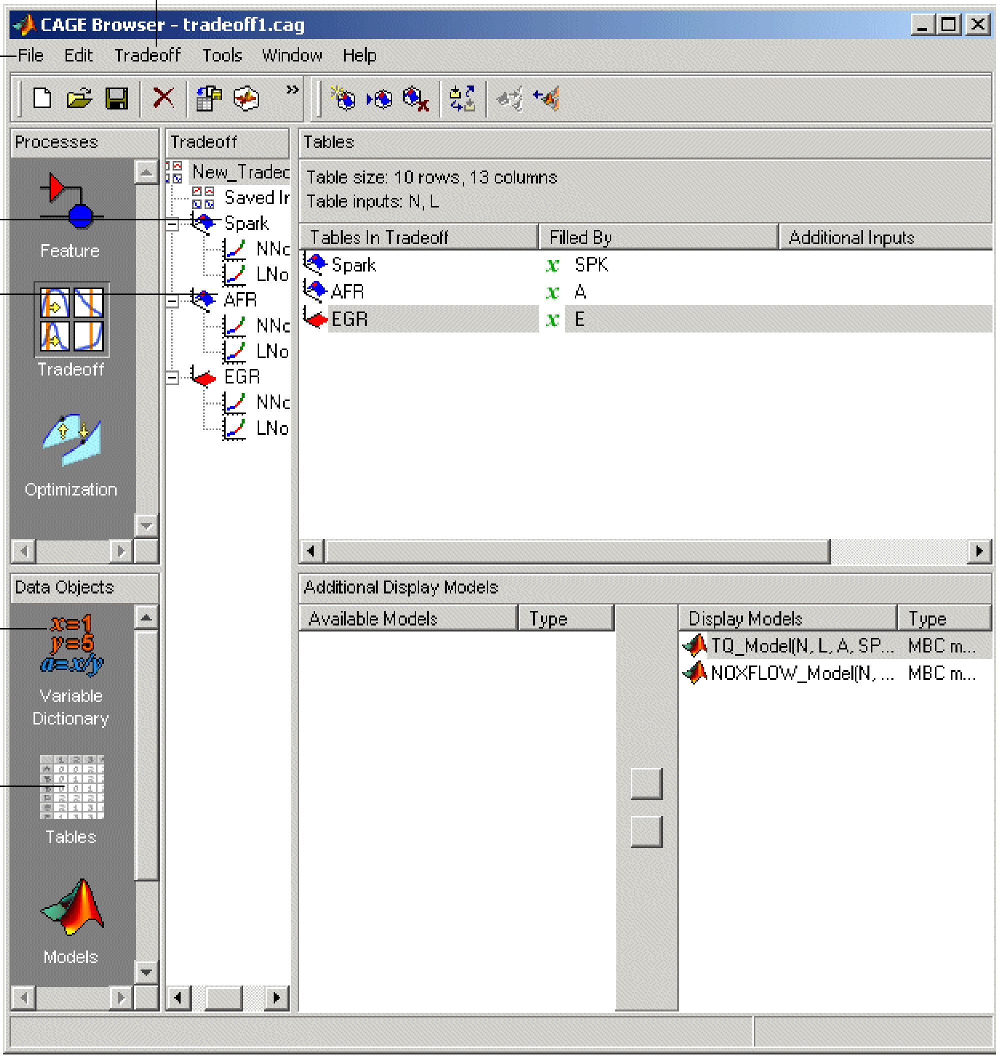


Table size: 10 rows, 13 columns
Table inputs: N, L

Tables In Tradeoff	Filled By	Additional Inputs
Spark	x SPK	
AFR	x A	
EGR	x E	

Available Models	Type

Display Models	Type
TQ_Model(N, L, A, SP...	MBC m...
NOXFLOW_Model(N, ...	MBC m...

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 10-9.

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

When you finish these steps, you are ready to calibrate the 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.

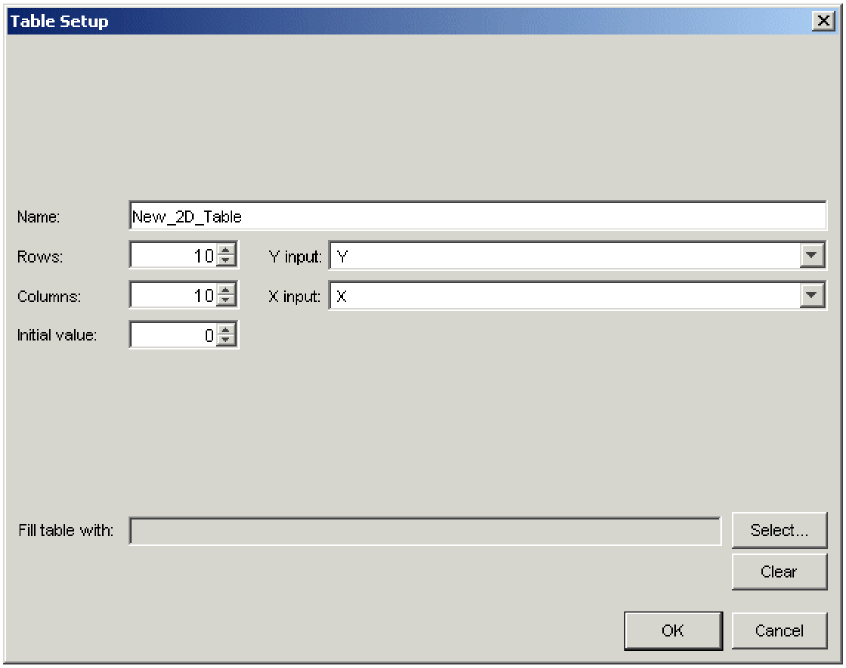
After you add a tradeoff you must add tables to your tradeoff.

Adding Tables to a Tradeoff

1 Add a table by selecting **Tradeoff –> Add New Table** or click in the toolbar. You can also add existing tables from your CAGE session; see “Adding Existing Tables” on page 10-9.

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.



Name: New_2D_Table

Rows: 10

Y input: Y

Columns: 10

X input: X

Initial value: 0

Fill table with:

2. Enter the name for the table.

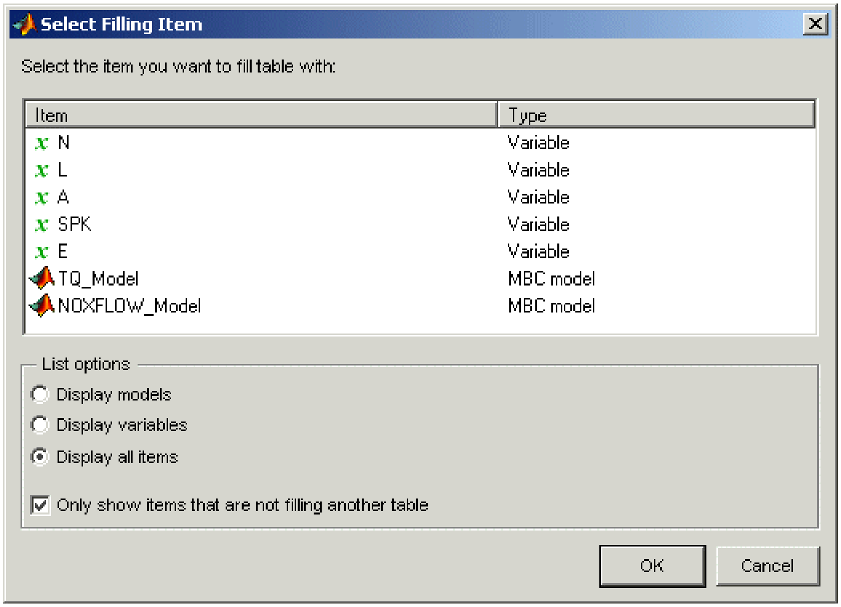
   If your tradeoff already contains one or more tables, when you add additional tables they must be the same size and have the same inputs (and therefore have the same normalizers). So if your tradeoff has existing tables, you can only enter the new table name and the initial value.

   For the first table in a tradeoff, you must set the normalizer inputs and sizes:

   a. Edit the names for the X and Y normalizer inputs (the first two variables in the current variable dictionary are automatically selected here).
   b. Enter sizes for each of the normalizers (Y input = rows, X input = columns)

3. Enter an initial value to fill the table cells, or leave this at zero.

4. Click **Select** to choose a filling item for a table. A dialog opens where you can select from the models and variables in your session.



Select Filling Item

Select the item you want to fill table with:

Item	Type
x N	Variable
x L	Variable
x A	Variable
x SPK	Variable
x E	Variable
TQ_Model	MBC model
NOXFLOW_Model	MBC model

List options

○ Display models

○ Display variables

● Display all items

☑ Only show items that are not filling another table

a Depending on what kind of input you want, click the radio buttons to display models or variables or both. You can choose to also show items that are filling another table by clearing the checkbox.

b Select the filling item for the table and click **OK**.

5 Click **OK** to dismiss the Tabe Setup dialog and create the new table.

CAGE adds a table node to the tradeoff tree. Note you can still change the input for the table as follows. Double-click the new table in the list under **Tables in Tradeoff**, or click to select the table (it is selected automatically if it is the only table in the tradeoff) and then click Change Filling Item ( ) in the toolbar. This is also in the **Tradeoff** menu.

The **Select Filling Item** dialog appears where you can select inputs to fill the table, as described above.

6 Repeat this procedure for each new table you want to add. Each additional table in the tradeoff must have the same normalizers as the first table, so you do not have to select normalizer inputs and sizes repeatedly. For each new table you only have to enter the name and initial value.

Adding Existing Tables

1 Add a table by selecting **Tradeoff –> Add Existing Tables** or click in the toolbar.

A dialog appears where you can select from a list of tables in the current session.

2 Select a table and click **OK**. It may be helpful to first select the check box to only show suitable tables that can be added to the tradeoff.

Displaying Models in Tradeoff

To display models when viewing tables in the tradeoff display,

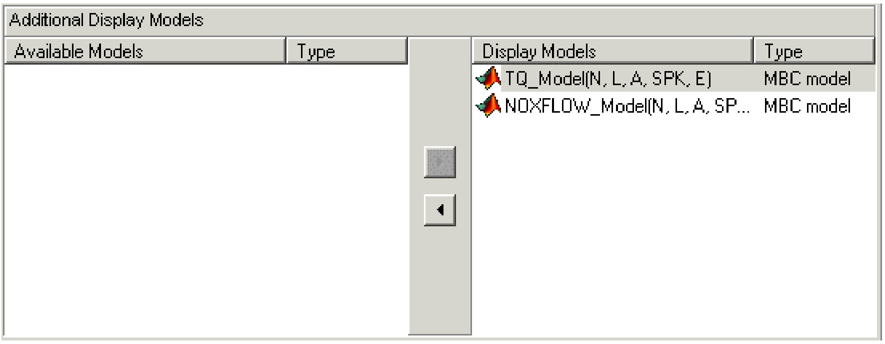
1 Highlight the tradeoff node in the tree.

2 From the **Available Models** list, select the one you want to display.

Models that are filling a table are automatically displayed.

3 Click Display Model in the toolbar or > in the **Display Models** pane to move the selected model into the **Current Display** pane. To quickly add all available models to the display list, click the display button repeatedly and each successive model will be added.

4 Repeat steps 2 and 3 to add all the models you want to the display list.



Display Models	Type
TQ_Model(N, L, A, SPK, E)	MBC model
NOXFLOW_Model(N, L, A, SP...	MBC model

Removing a Model

1 In the **Display Models** list, select the model that you want to remove.

2 Click in the toolbar, or < in the **Display Models** pane, 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 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 10-13.

4 Determine the value of the desired operating point.

For instructions, see “Determining a Value at a Specific Operating Point” on page 10-16.

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 in the toolbar.

For information, see “Filling the Table by Extrapolation” on page 9-21.

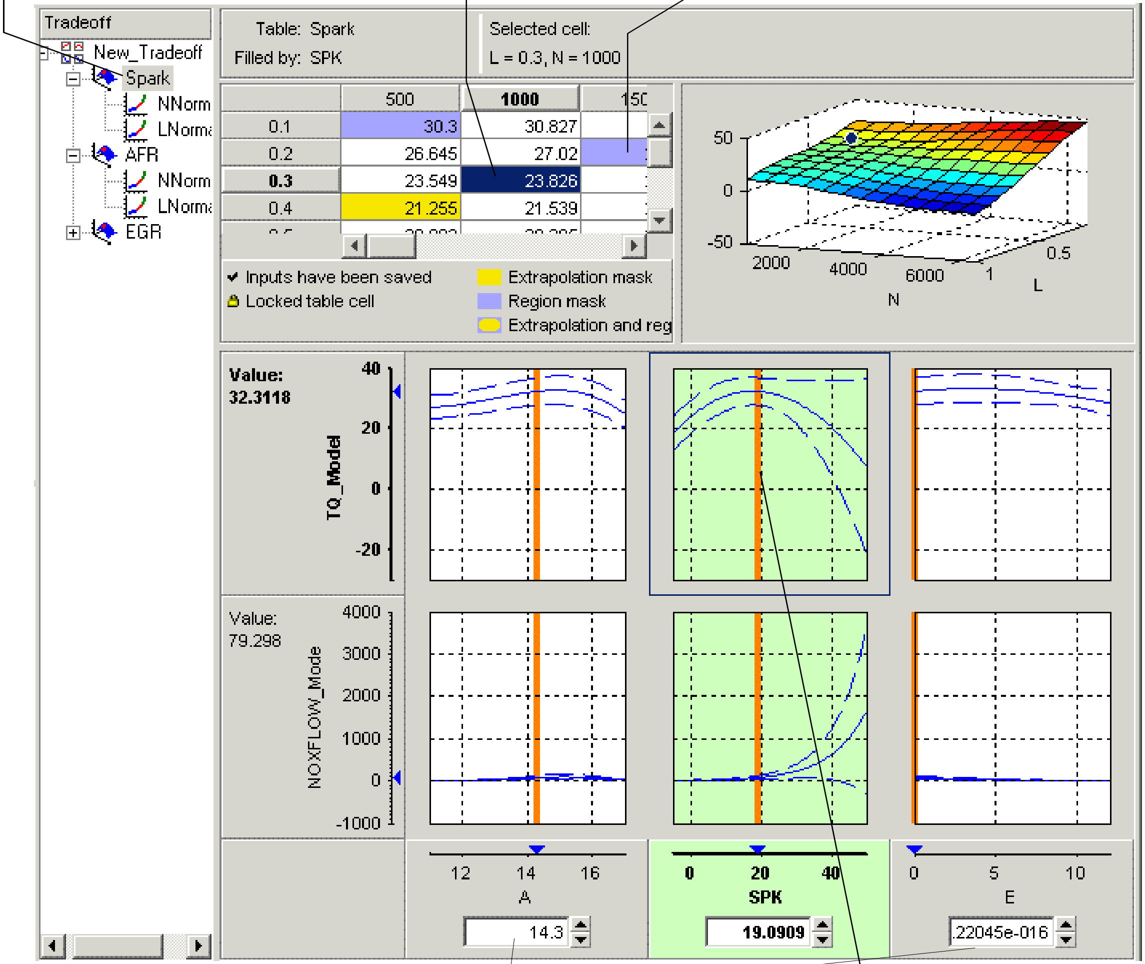
After you complete all these steps you can export your calibration. For information, see “Exporting Calibrations” on page 7-27.

Table View in a Tradeoff Calibration

1. Select the table.

2. Select the operating point in the table that you want to calibrate.

5. Repeat this process over a number of operating points in the table, then fill the table by extrapolation.



Tradeoff

New_Tradeoff
- Spark
  - NNorm
  - LNorma
- AFR
  - NNorm
  - LNorma
- EGR

Table: Spark
Filled by: SPK

Selected cell:
L = 0.3, N = 1000

	500	1000	150
0.1	30.3	30.827	
0.2	26.645	27.02	
0.3	23.549	23.826	
0.4	21.255	21.539	

✓ Inputs have been saved
🔒 Locked table cell
🟨 Extrapolation mask
🟪 Region mask
🟡 Extrapolation and reg

Value:
32.3118

Value:
79.298

A 14.3

SPK 19.0909

E .22045e-016

3. Set the values for other input variables.

4. Determine a suitable value for the point.

Notice that the graphs colored green indicate how the highlighted table will be filled:

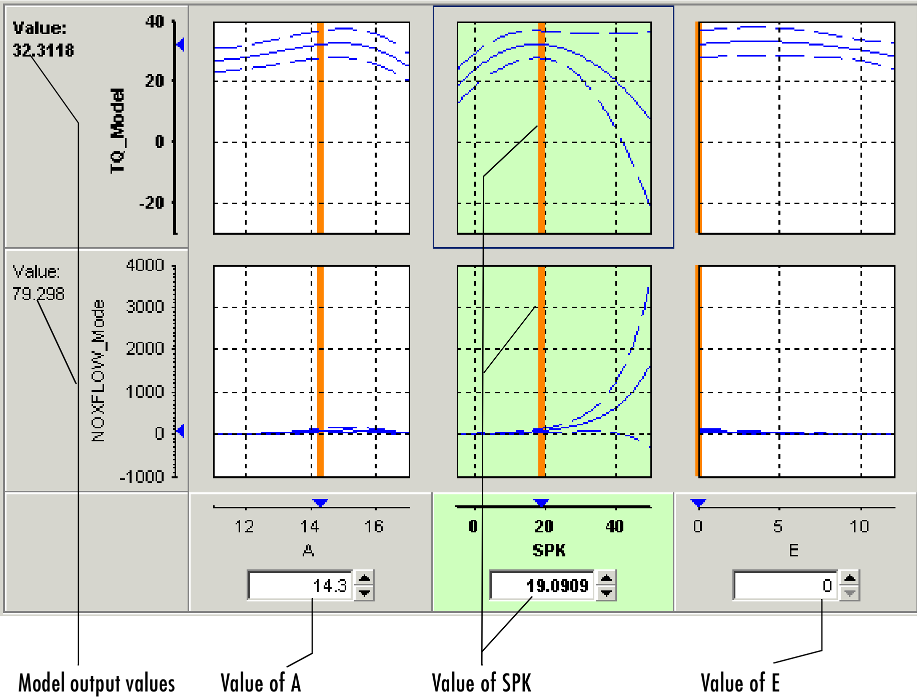
- If a row of graphs is highlighted, the table is being filled by the indicated model evaluation (the value shown at the left of the row).
- If the column of graphs is green, the table is being filled by the indicated input variable (shown in the edit box below the column).

The next sections describe the following in detail:

- “Setting Values of Other Variables” on page 10-13
- “Determining a Value at a Specific Operating Point” on page 10-16

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.



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 or drag the red bars 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, for example:

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`.

The default values are the set points of variables, which you can edit in the Variable Dictionary.

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,

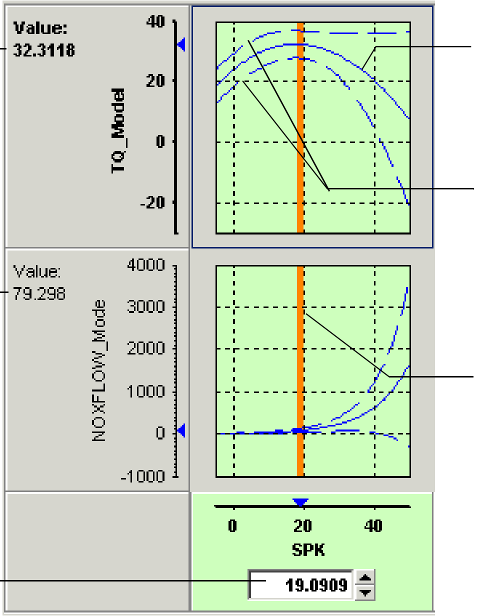
1. Highlight the table in the branch display.
2. Select one operating point in the table.
3. Enter the desired value of the cell.
4. Right-click and select **Extrapolation Mask –> Add Selection**.

   This adds the cell to the extrapolation mask.

5. 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



Value: 32.3118

Value: 79.298

19.0909

Value of the TQ_ Model

Value of the NOXFLOW_Model

Edit box displaying the value of SPK

Behavior of TQ_Model

99% confidence limits for TQ_Model

Value of spark

Performing a tradeoff calibration necessarily involves the comparison of two or more models. For example, in this case, the tradeoff allows a calibrator to check that a value of spark that gives peak torque also gives an acceptable value for the NOX flow model.

1 To select a value of an input, do one of the following:

- Drag the red line.
- Right-click a graph and select **Find** the minimum, maximum, or turning point of the model as appropriate (also in the toolbar and **Inputs** menu).
- Click the edit box under the graph as shown above and enter the required value.

2. Once you are satisfied with the value of your variable at this operating point, you apply this value to the table by doing one of the following:
   - Press **Ctrl+T**.
   - Click (Apply Table Filling Values) in the toolbar.
   - Select **Tables –> Apply Fill to Table**.

Right-Click Menu

Right-clicking a graph enables you to

- Find minimum of model output with respect to the input variable
- Find maximum of model output with respect to the input variable
- Find turning point of model with respect to the input variable

  These first three options are also in the **Inputs** menu.
- Reset graph zooms (also in the **View** menu)

There are also toolbar buttons to find the minimum, maximum and turning point of the selected model graph.

Using Zoom Controls on the Graphs

To zoom in on a particular region, shift-click or click with both mouse buttons simultaneously and drag to define the region as a rectangle.

To zoom out to the original graph, double-click the selected graph, or use the right-click **Reset Graph Zooms** option (also in the **View** menu).

Note Zooming on one graph adjusts other graphs to the same scale.

Tradeoff Table Menus

View Menu

Selecting the **View** menu offers you the following options:

- **Table History**

  This opens the **History** display. For information, see "Using the History Display" on page 15-6.

- **Configure Hidden Items**

  This opens a dialog box that allows you to show or hide models and input variables. Select or clear the check boxes to display or hide items. This is particularly useful if you are trading off a large number of models or models that have a large number of factors.
- **Display Confidence Intervals**

  When you select this, the graphs display the 99% confidence limits for the models.
- **Display Common Y Limits**

  Select this to toggle a common *y*-axis on and off for all the graphs. You can also press **CTRL**+Y as a shortcut to turn common Y limits on and off.
- **Display Constraints**

  Select this to toggle constraint displays on and off. Regions outside constraints are shown in yellow on the graphs, as elsewhere in the toolbox.
- **Graph Size**

  Select from the following options for number and size of graphs:

  - **Display All Graphs**
  - **Small**
  - **Medium**
  - **Large**
- **Large Graph Headers**

  Select this to toggle graph header size. The smaller size can be useful when you need to display many models at once.
- **Reset Graph Zooms**

  Use this to reset if you have zoomed in on areas of interest on the graphs. Zoom in by shift-clicking (or clicking both buttons) and dragging. You can also reset the zooms by double-clicking, or by using the right-click context menu on the graphs.

- **Display Table Legend**

  Select this to toggle the table legend display on and off. You might want more display space for table cells once you know what the legend means. The table legend tells you how to interpret the table display:

  - Cells with a tick contain saved values that you have applied from the tradeoff graphs (using the **Apply Fill To Table** toolbar or menu option).
  - Yellow cells are in the extrapolation mask.
  - Blue cells are in a region mask.
  - Yellow and blue cells with rounded corners are both in a region and the extrapolation mask.
  - Cells with a padlock icon are locked.

Tables Menu

- **Apply Fill to Table**

  Select this option to apply the values from the tradeoff graphs to the selected table cell. This option is also in the toolbar, and you can use the keyboard shortcut **CTRL**+T.

  Note that the corresponding cell in all tables is filled with the appropriate input, not just the cell in the currently displayed table. For example if you have graphs for spark and EGR inputs, selecting **Apply Fill to Table** fills the spark table cell with the spark value in the graphs, and the EGR table cell with the EGR value.

- **Extrapolation Mask** — Also available in the toolbar and the context menu (by right-clicking a table cell). Use these options to add and remove cells from the mask for filling tables by extrapolation. Note that cells filled by applying values from the tradeoff graphs (using the **Apply Fill To Table** toolbar and menu option) are automatically added to the extrapolation mask.

  - **Add Selection**
  - **Remove Selection**
  - **Clear Mask**

- **Extrapolation Regions** — Also available in the toolbar and the context menu (by right-clicking a table cell). Use these options to add and remove cells from regions. A region is an area that defines locally where to extrapolate before globally extrapolating over the entire table. Use regions to define high-priority areas for use when filling tables by extrapolation. See "Using Regions" on page 10-22.

- **Add Selection**
- **Remove Selection**
- **Clear Regions**

- **Extrapolate** — This option (also in the toolbar) fills the table by extrapolation using regions (to define locally where to extrapolate before globally extrapolating) and the cells defined in the extrapolation mask.
- **Extrapolate (Ignore Regions)** — This option fills the table by extrapolation only using cells in the extrapolation mask.
- **Table Cell Locks** — Also available in the context menu by right-clicking a table cell. Use these options to lock or unlock cells; locked cells are not changed by extrapolating.
  - **Lock Selection**
  - **Unlock Selection**
  - **Lock Entire Table**
  - **Clear All Locks**

Inputs Menu

- **Reset to Last Saved Values** — This option resets all the graph input values to the last saved value. Also in the toolbar.
- **Set to Table Value** — This option sets the appropriate input value on the graphs to the value in the table.

The following three options are only enabled if a graph is selected (click to select, and a blue frame appears around the selected graph). They are also available in the right-click context menu on the graphs.

- **Find Minimum of** *model* **vs** *input factor*
- **Find Maximum of** *model* **vs** *input factor*
- **Find Turning Point of** *model* **vs** *input factor*

  where *model* and *input factor* are the model and input factor displayed in the currently selected graph, for example, `TQ_model` vs `Spark`.
- **Automated Tradeoff** — Use this option once you have set up an optimization, to apply that optimization to the selected region of your tradeoff table. See "Automated Tradeoff" on page 11-35 for information.

Tools Menu

- **Calibration Manager** — opens the Calibration Manager. See “Calibration Manager” on page 13-1.
- **Surface Viewer** — Opens the Surface Viewer. See “Surface Viewer” on page 14-1.

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.

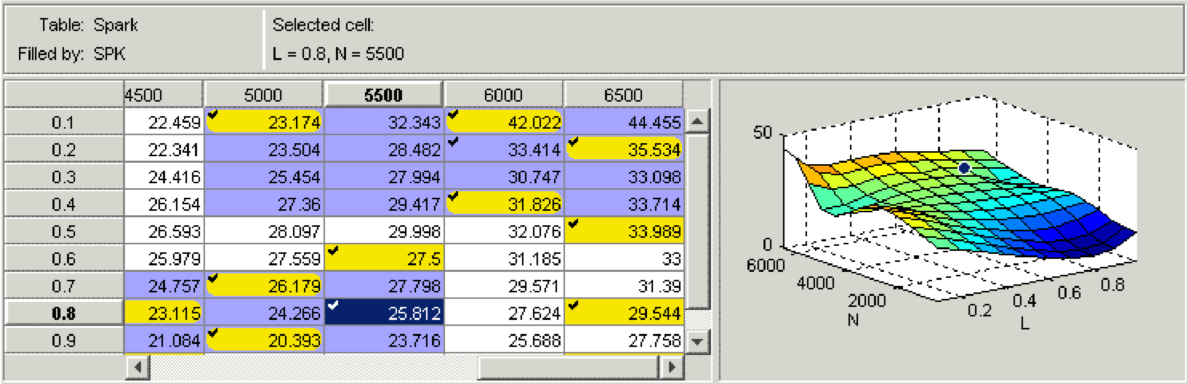


Table: Spark
Filled by: SPK

Selected cell:
L = 0.8, N = 5500

	4500	5000	5500	6000	6500
0.1	22.459	✓ 23.174	32.343	✓ 42.022	44.455
0.2	22.341	23.504	28.482	✓ 33.414	✓ 35.534
0.3	24.416	25.454	27.994	30.747	33.098
0.4	26.154	27.36	29.417	✓ 31.826	33.714
0.5	26.593	28.097	29.998	32.076	✓ 33.989
0.6	25.979	27.559	✓ 27.5	31.185	33
0.7	24.757	✓ 26.179	27.798	29.571	31.39
0.8	23.115	24.266	✓ 25.812	27.624	✓ 29.544
0.9	21.084	✓ 20.393	23.716	25.688	27.758

Cells are colored

- Yellow if they form part of the extrapolation mask

- Blue if they are part of a region
- Yellow and blue with rounded corners if they are part of the extrapolation mask and part of a region

Defining a Region

1 Click and drag to highlight the rectangle of cells in your table.

2 To define the region, click in the toolbar, or right-click and select **Extrapolation Regions –> Add Selection**, or select the menu option **Tables –> Extrapolation Regions –> Add Selection**.

The cells in the region are colored blue.

Clearing a Region

1 Highlight the rectangle of cells in your table.

2 To clear the region, click in the toolbar, or right-click and select **Extrapolation Regions –> Remove Selection**, or select the menu option **Tables –> Extrapolation Regions –> Remove Selection**.

You can clear all regions at once by selecting **Clear Regions** from the **Extrapolation Regions** submenu.

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 10-25.)

2 Calibrate the tables. (See “Calibrating Using a Multimodel Tradeoff” on page 10-27.)

3 Export your calibration. (See “Exporting Calibrations” on page 7-27.)

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 have model icons 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 so you cannot use models to tradeoff. You can edit these cells and 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 10-16. 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 9-21.

Adding a Multimodel Tradeoff

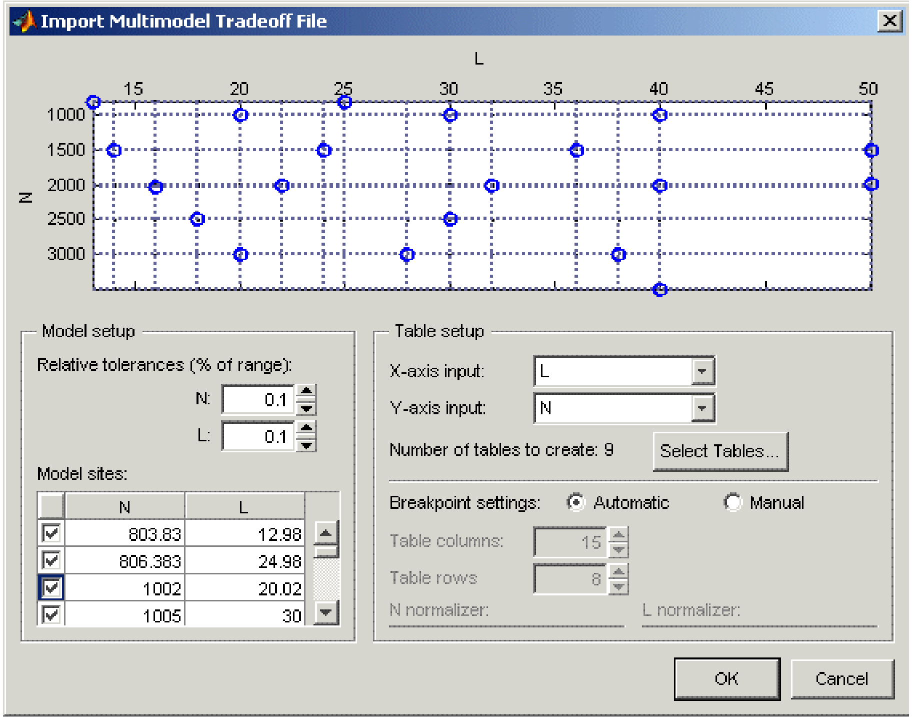
To add a multimodel tradeoff to your session,

1 Select **File –> New –> Tradeoff**. CAGE switches to the tradeoff view and creates a new empty tradeoff.

2 Select the new tradeoff in the tree, then select **File –> Import –> 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). See "Multimodel Tradeoffs" on page 10-24.

3 Select the correct file to import and click **Open**. This opens a dialog box.



Model setup

Relative tolerances (% of range):

N: 0.1

L: 0.1

Model sites:

☐	N	L
☑	803.83	12.98
☑	806.383	24.98
☑	1002	20.02
☑	1005	30

Table setup

X-axis input: L

Y-axis input: N

Number of tables to create: 9

Breakpoint settings: ● Automatic ○ Manual

Table columns: 15

Table rows: 8

N normalizer:

L normalizer:

4. In the left **Model sites** list, you can clear the check boxes for any models at operating points that you do not want to import.

   Notice that the operating points are displayed graphically at the top. If an operating point is deselected, it is displayed as gray here, rather than blue.

   CAGE will create tables for all the models and input variables, with breakpoints at all the model operating points. You can choose not to create all the tables; click **Select Tables** to choose from the list which tables you want.

5. Choose the normalizers (axes) of the tables by using the `X-` and `Y-axis input` drop-down menus.

6. You can adjust the number of breakpoints in the following ways:

   - Leave the **Automatic** breakpoint settings radio button selected and edit the relative tolerances around the model sites. Use the tolerance edit boxes in the model setup pane. You can observe the effects of altering the tolerances on the number of breakpoint dotted lines drawn on the top graphic. Initially each model site has a breakpoint. If operating points are close together, you can increase the tolerances to decrease the number of breakpoints.

     For example, if several close points may all have been intended to run at exactly the same point, you might want to adjust the tolerances until those model points (displayed as blue dots) only have one breakpoint line. The number of rows and columns that will be created is displayed in the edit boxes on the right.
   - Alternatively you can select the **Manual** breakpoint settings radio button and enter the number of rows and columns in the edit boxes, and you can directly edit the values of the breakpoints.

7. Click **OK**.

When you click **OK**, CAGE creates all the tables for the multimodel tradeoff, with breakpoints at the values you have selected.

Note When you calibrate the tables, you can only use models to tradeoff at the operating points defined for the models. These cells have model icons in the table. You can edit other cells, but they have no models to tradeoff associated with them.

You can now calibrate your tables. See the next section, "Calibrating Using a Multimodel Tradeoff" on page 10-27.

Calibrating Using a Multimodel Tradeoff

Each editable operating point in your tables has a model icon in the cell, like this example cell.

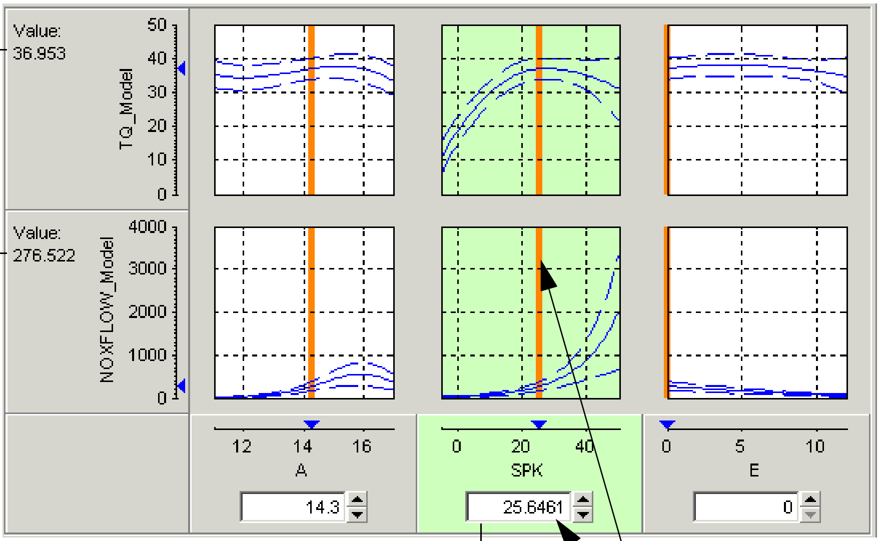
4602.722

These cells have 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 10-16.

1 Change input values by dragging the red lines on the graphs or by typing directly into the edit boxes above the graphs. Use the context menu, toolbar or **Inputs** menu to find the maximum, minimum, or turning point of a model if appropriate.

2 Look at the model evaluation values (to the left of each row of graphs) and the input variable values (in the edit boxes below 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 left) fills the table at that operating point; if a column is green, the input variable value (in the edit box below) 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 Apply Table Filling Values button ( ).



Value: 36.953

Value: 276.522

A: 14.3

SPK: 25.6461

E: 0

Value of the TQ model

Value of the NOXFLOW model

This column is green, so this value of SPK is entered in the table when you select **Apply Fill to Table**.

Value of spark

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 **Tables –> Apply Fill to Table**, 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 9-21.

You can then export your calibration; see “Exporting Calibrations” on page 7-27. An example multimodel tradeoff is shown following.

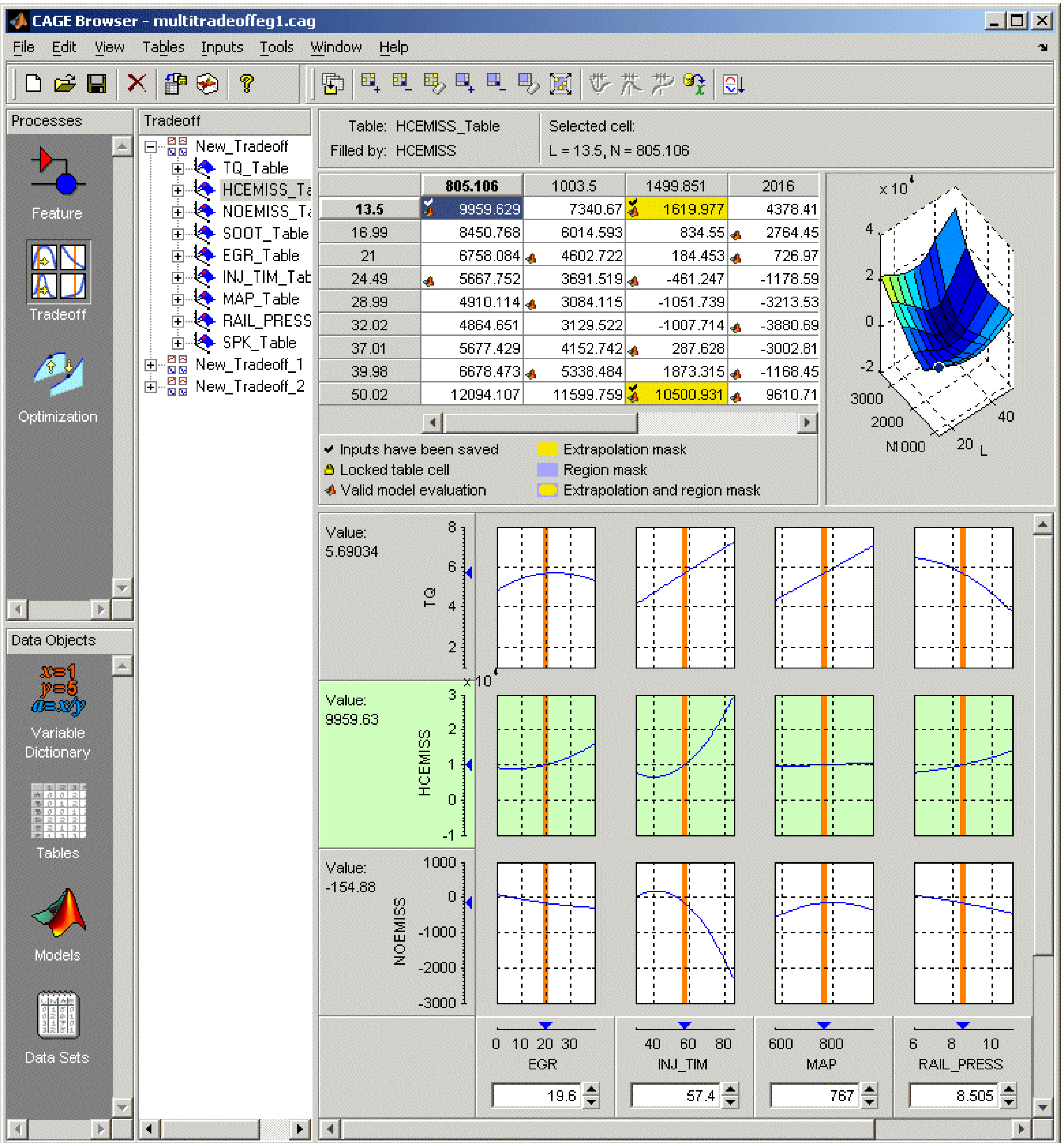


Table: HCEMISS_Table

Filled by: HCEMISS

Selected cell:

L = 13.5, N = 805.106

	805.106	1003.5	1499.851	2016
13.5	9959.629	7340.67	1619.977	4378.41
16.99	8450.768	6014.593	834.55	2764.45
21	6758.084	4602.722	184.453	726.97
24.49	5667.752	3691.519	-461.247	-1178.59
28.99	4910.114	3084.115	-1051.739	-3213.53
32.02	4864.651	3129.522	-1007.714	-3880.69
37.01	5677.429	4152.742	287.628	-3002.81
39.98	6678.473	5338.484	1873.315	-1168.45
50.02	12094.107	11599.759	10500.931	9610.71

✓ Inputs have been saved

🔒 Locked table cell

▲ Valid model evaluation

🟨 Extrapolation mask

🟪 Region mask

🟨 Extrapolation and region mask

Value:
5.69034

Value:
9959.63

Value:
-154.88

EGR: 19.6

INJ_TIM: 57.4

MAP: 767

RAIL_PRESS: 8.505

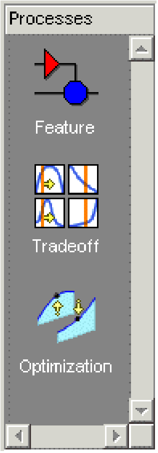
11

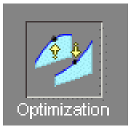
Optimization in CAGE

This section includes the following topics:

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 View" on page 11-26

Once you have set up an optimization you can apply it to a region in a tradeoff table. See "Automated Tradeoff" on page 11-35

You can define your own optimization functions for use in CAGE. See "User-Defined Optimization" on page 11-39

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

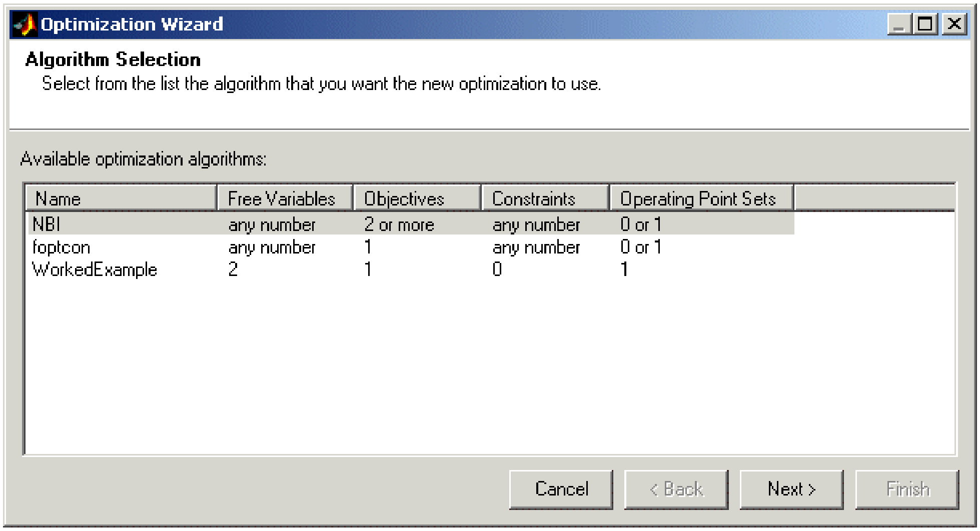
Normally you run an optimization on a session you already have open. For an example session you could open the `tradeoffInit.cag` file in the `mbctraining` folder.

- 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.



Algorithm Selection

Select from the list the algorithm that you want the new optimization to use.

Available optimization algorithms:

Name	Free Variables	Objectives	Constraints	Operating Point Sets
NBI	any number	2 or more	any number	0 or 1
foptcon	any number	1	any number	0 or 1
WorkedExample	2	1	0	1

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.

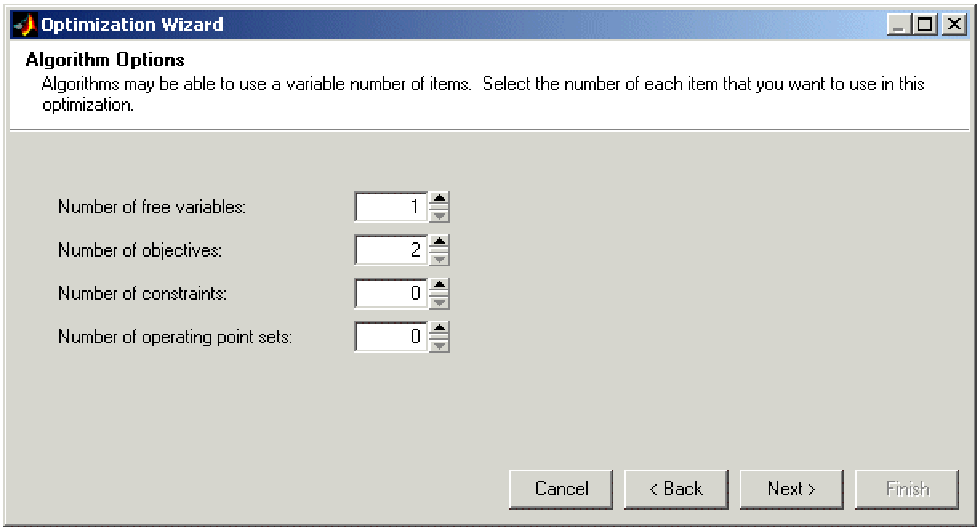
In many cases these standard routines are sufficient to allow you to solve your optimization problem. Sometimes, however, you might need to write a customized optimization algorithm; to do this you can use the supplied template to modify for your needs. See “User-Defined Optimization” on page 11-39 for information. The Worked Example option is designed to show you how to use the modified template; see “Worked Example Optimization” on page 6-40 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:



Optimization Wizard

Algorithm Options

Algorithms may be able to use a variable number of items. Select the number of each item that you want to use in this optimization.

Number of free variables: 1

Number of objectives: 2

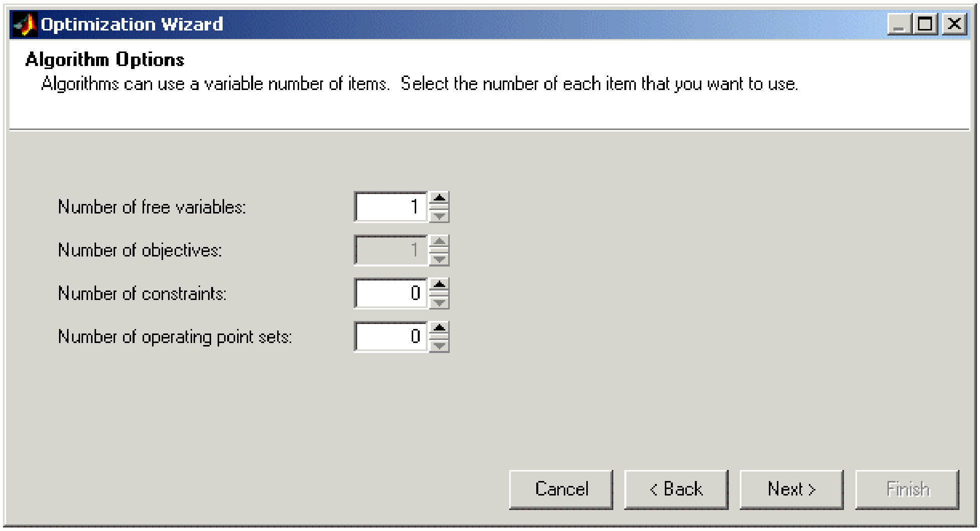
Number of constraints: 0

Number of operating point sets: 0

`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:



Optimization Wizard

Algorithm Options

Algorithms can use a variable number of items. Select the number of each item that you want to use.

Number of free variables: 1

Number of objectives: 1

Number of constraints: 0

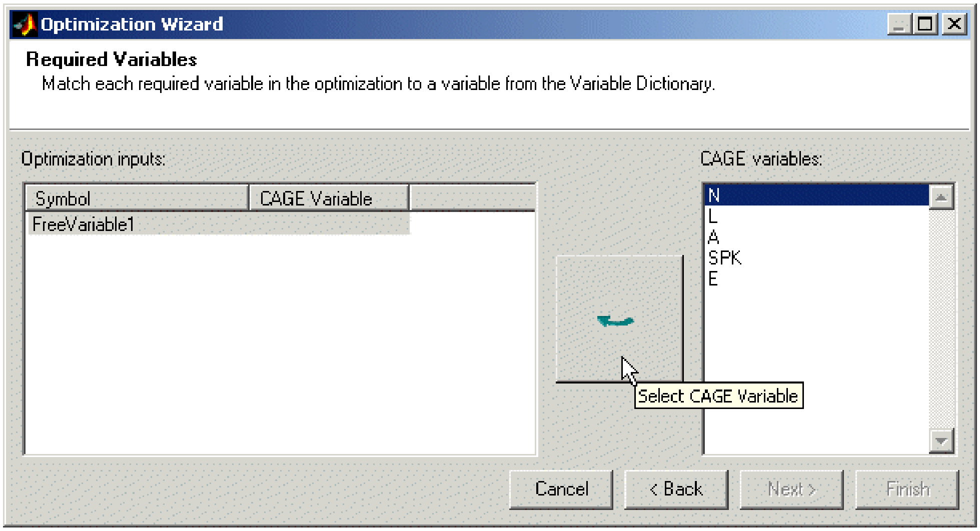
Number of operating point sets: 0

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.



Optimization Wizard

Required Variables

Match each required variable in the optimization to a variable from the Variable Dictionary.

Optimization inputs:

Symbol	CAGE Variable
FreeVariable1	

CAGE variables:

N
L
A
SPK
E

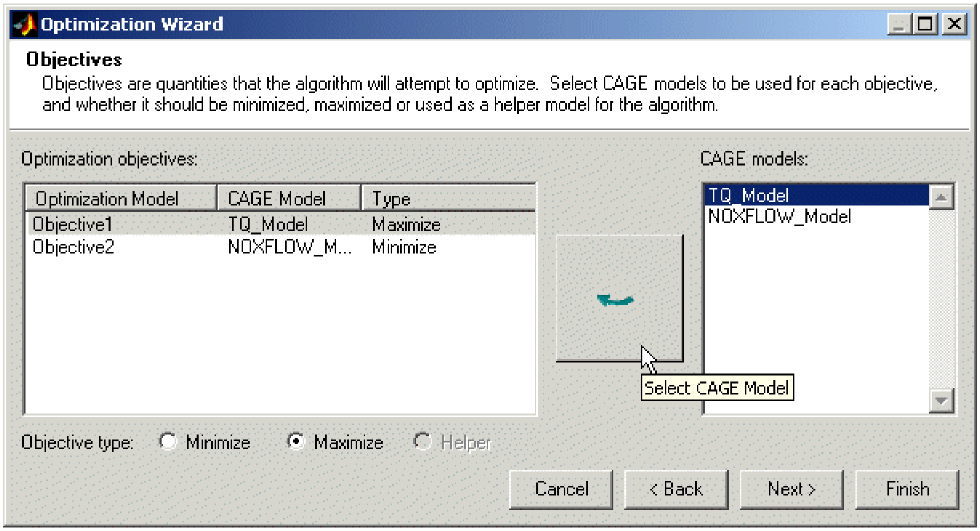
Use this screen to associate the variables from your CAGE session with the free variable(s) you want to use in the optimization. Select the correct pair in the right and left lists by clicking, then click the large button as indicated in the figure.

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 view in CAGE. See “Objective Editor” on page 11-22.



Objectives

Objectives are quantities that the algorithm will attempt to optimize. Select CAGE models to be used for each objective, and whether it should be minimized, maximized or used as a helper model for the algorithm.

Optimization objectives:

Optimization Model	CAGE Model	Type
Objective1	TQ_Model	Maximize
Objective2	NOXFLOW_M...	Minimize

CAGE models:

- TQ_Model
- NOXFLOW_Model

Objective type: ○ Minimize ● Maximize ○ Helper

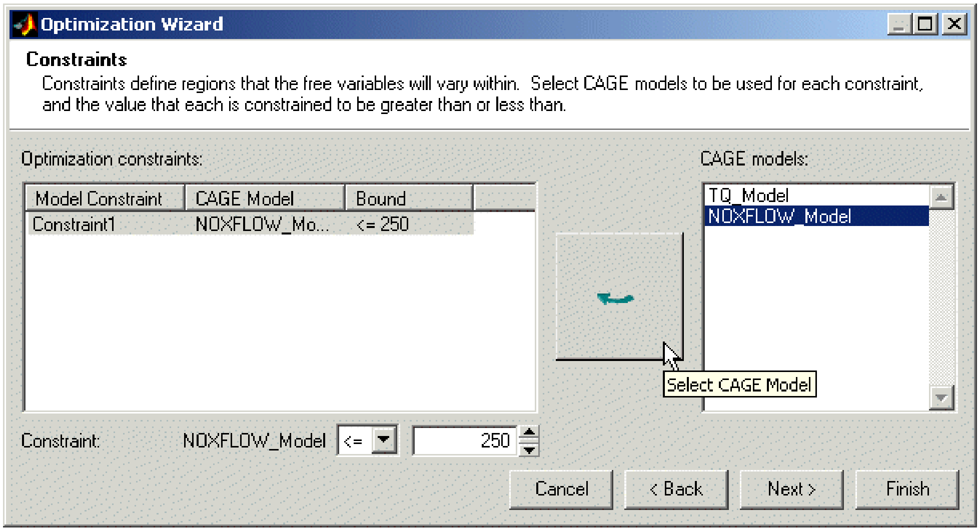
Here you can select which models from your session you want to use for the optimization, and whether you want to maximize or minimize the model output. The `foptcon` algorithm is for single objectives, so you can only maximize or minimize one model. The `NBI` algorithm can evaluate multiple objectives. 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. Note you can only set up point objectives in the wizard, but you can also set up sum objectives in the main **Optimization** view. See “Objective and Constraint Editors” on page 11-21.

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 in the Optimization view in CAGE. You can also add other types of constraints in the Optimization view. See “Constraint Editor” on page 11-24.



Constraints

Constraints define regions that the free variables will vary within. Select CAGE models to be used for each constraint, and the value that each is constrained to be greater than or less than.

Optimization constraints:

Model Constraint	CAGE Model	Bound
Constraint1	NOXFLOW_Mo...	<= 250

CAGE models:

- TQ_Model
- NOXFLOW_Model

Constraint: NOXFLOW_Model <= 250

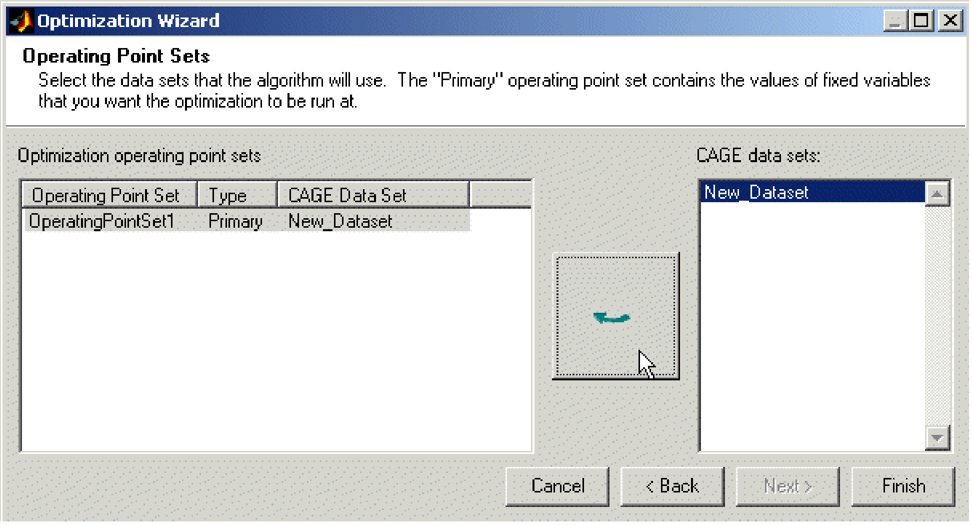
Select a model for each constraint by selecting a CAGE model and a model constraint and clicking the button to match them up.

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_Model <= 250`.

- Click **Next** to proceed to setting up operating point sets.
- Click **Finish** to complete the **Optimization Wizard** and return to the **Optimization** view. Note you can only set up point constraints in the wizard, but you can also set up sum constraints in the main **Optimization** view. See “Objective and Constraint Editors” on page 11-21.

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 in the Optimization view in CAGE.



Optimization Wizard

Operating Point Sets

Select the data sets that the algorithm will use. The "Primary" operating point set contains the values of fixed variables that you want the optimization to be run at.

Optimization operating point sets

Operating Point Set	Type	CAGE Data Set
OperatingPointSet1	Primary	New_Dataset

CAGE data sets:

New_Dataset

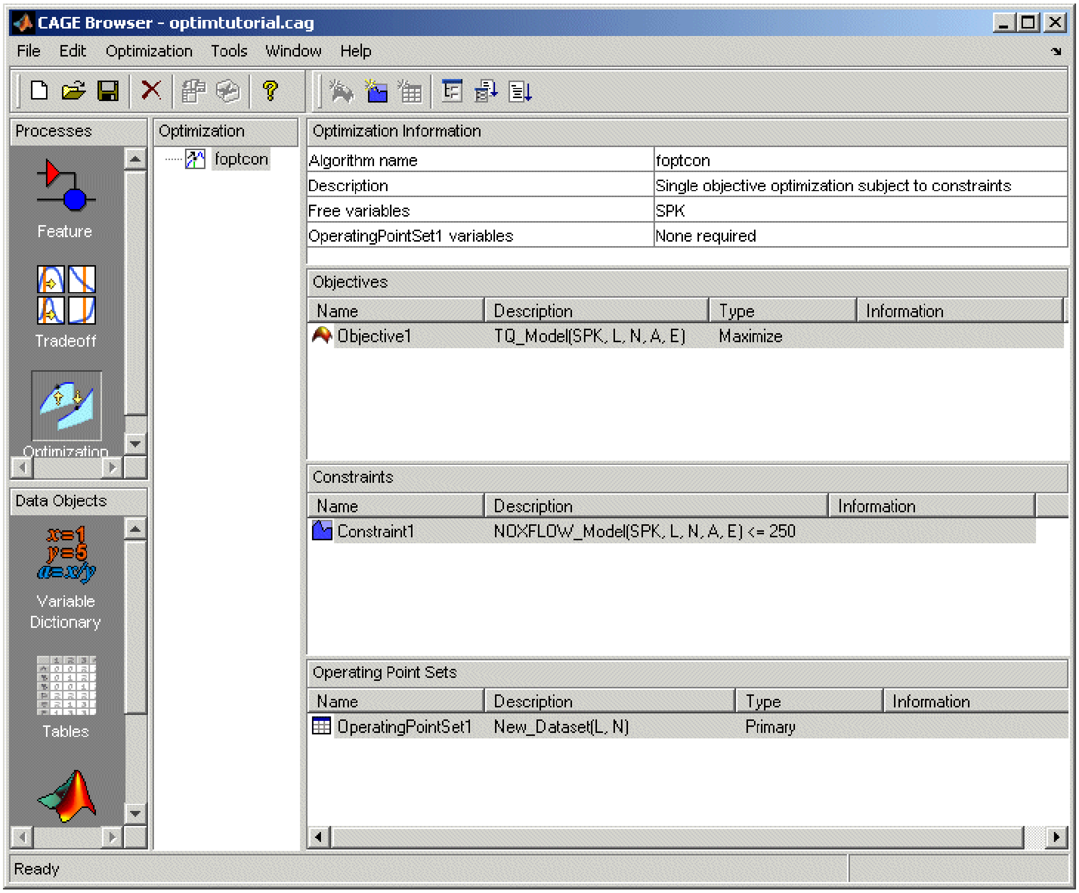
Select a CAGE data set and click the button to match it up with `OperatingPointSet1`. The `Primary` (first) operating point set contains the set of points defined by the fixed variables where you want the optimization to run.

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 view in CAGE.

Running Optimizations

When you click **Finish** to complete the **Optimization Wizard**, you return to the Optimization view 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:



Optimization Information

Algorithm name	foptcon
Description	Single objective optimization subject to constraints
Free variables	SPK
OperatingPointSet1 variables	None required

Objectives

Name	Description	Type	Information
Objective1	TQ_Model(SPK, L, N, A, E)	Maximize	

Constraints

Name	Description	Information
Constraint1	NOXFLOW_Model(SPK, L, N, A, E) <= 250	

Operating Point Sets

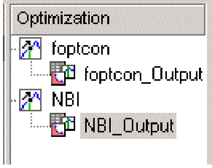
Name	Description	Type	Information
OperatingPointSet1	New_Dataset(L, N)	Primary	

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**

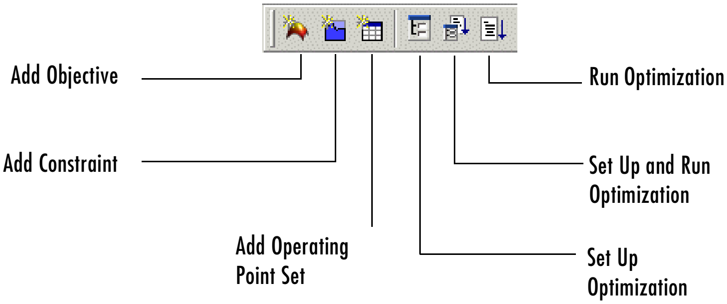
and Run Optimization if you want to change settings such as number of required solutions and tolerances for termination.

- 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 View" on page 11-26.



Optimization View 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.

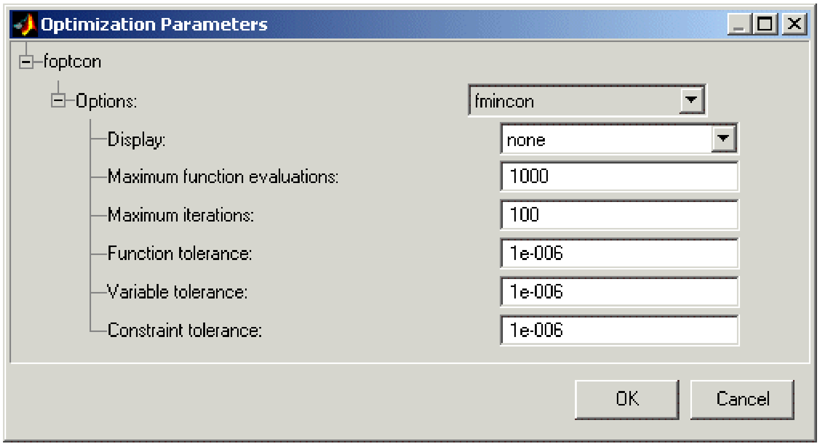
Optimization Parameters Dialog

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.



foptcon
Options: fmincon
Display: none
Maximum function evaluations: 1000
Maximum iterations: 100
Function tolerance: 1e-006
Variable tolerance: 1e-006
Constraint tolerance: 1e-006

- **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.

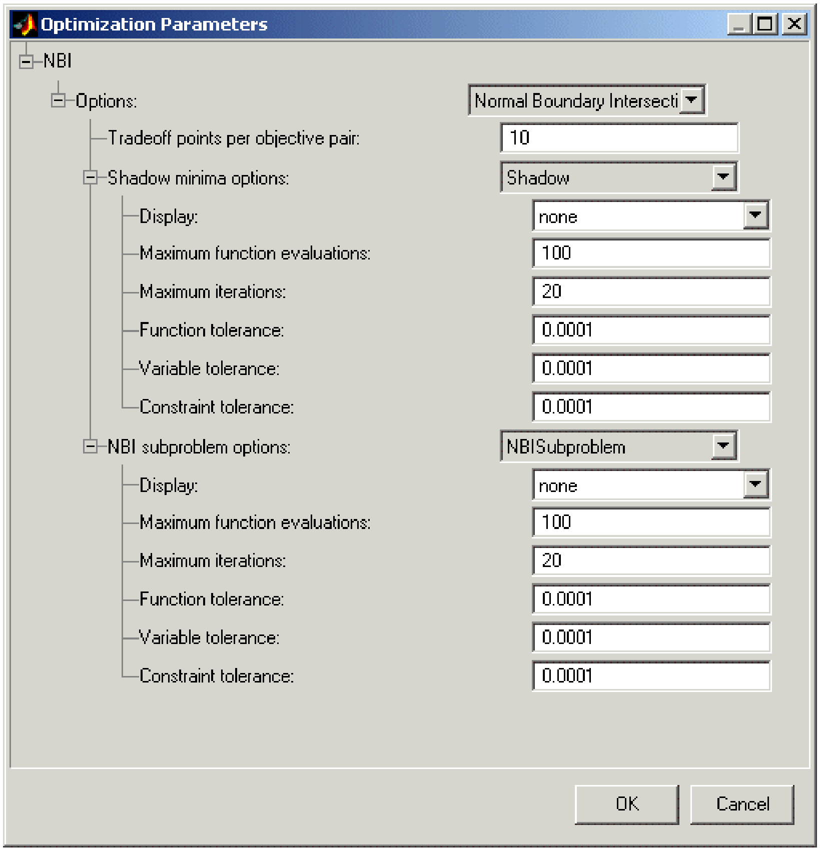
  Termination tolerance on the function value
- **Variable tolerance** — Choose a positive scalar.

  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.



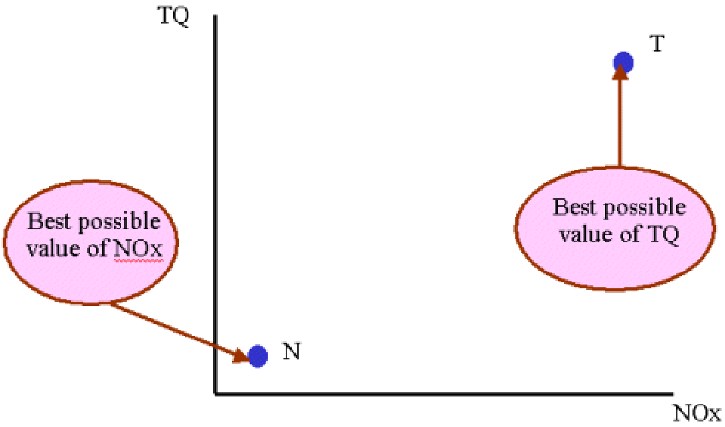
- NBI
  - Options: Normal Boundary Intersecti
    - Tradeoff points per objective pair: 10
    - Shadow minima options: Shadow
      - Display: none
      - Maximum function evaluations: 100
      - Maximum iterations: 20
      - Function tolerance: 0.0001
      - Variable tolerance: 0.0001
      - Constraint tolerance: 0.0001
    - NBI subproblem options: NBISubproblem
      - Display: none
      - Maximum function evaluations: 100
      - Maximum iterations: 20
      - Function tolerance: 0.0001
      - Variable tolerance: 0.0001
      - Constraint tolerance: 0.0001

Background on the NBI (Normal Boundary Intersection Algorithm)

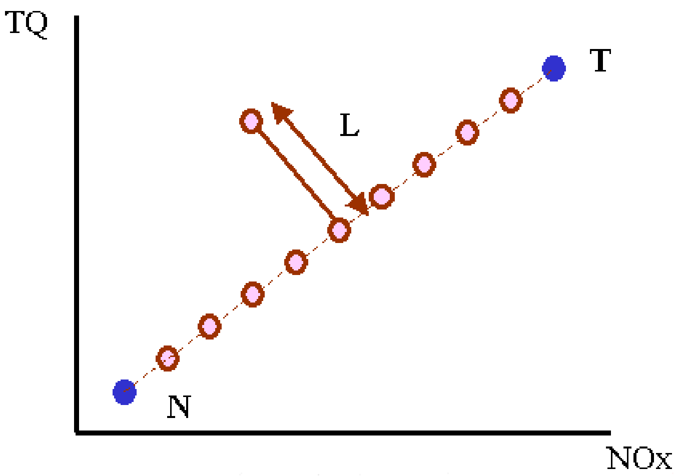
To understand the options for the NBI algorithm, some limited understanding of the algorithm is required. For more information on the NBI algorithm, see the NBI home page at the following URL:

`http://www.caam.rice.edu/~indra/NBIhomepage.html`

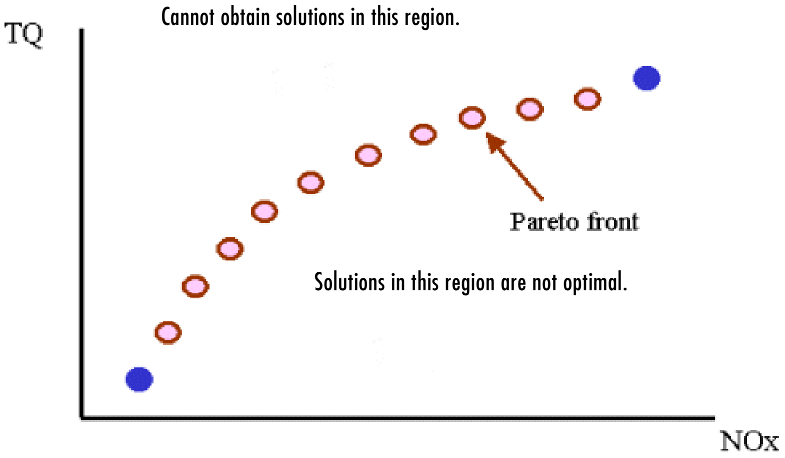
The NBI algorithm is performed in two steps. The first step is to find the global extrema of each objective individually. This is called the shadow minima problem, and is a single-objective problem for each objective function. The MATLAB routine `fmincon` is used to find these extrema. Once these extrema 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 `Npts` 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 `Npts` 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** (`Np`)

The number of tradeoff solutions between your objectives that you want to find, `Npts`, is determined by the following formula

$$Npts = \binom{n + Np - 2}{Np - 1}$$

where

- `Np` 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),

$$Npts = Np$$

- 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 6-33.

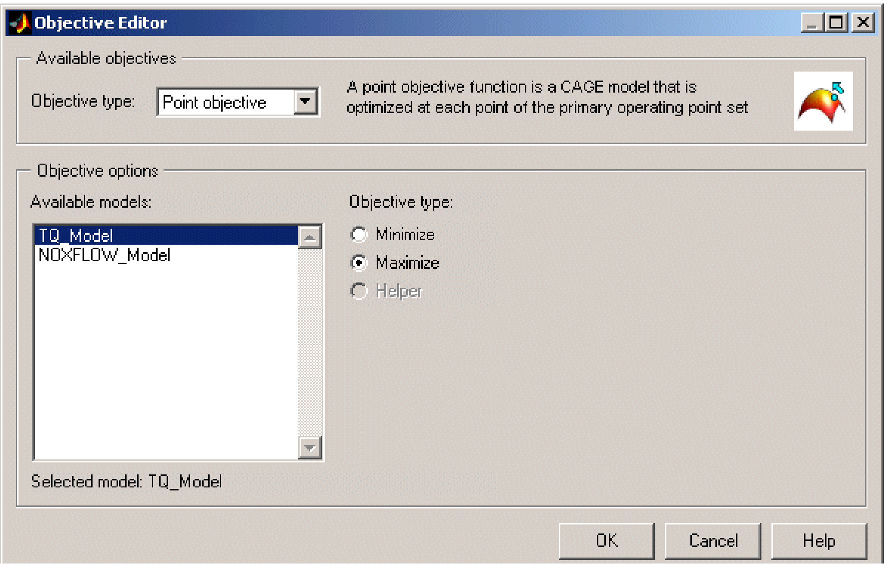
Note If you have a weighted sum constraint, you *must* use sum objectives.

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 have a mixture of point and sum constraints — but if you add a sum constraint you must also make all your objectives sum objectives or CAGE will not be able to solve the optimization problem. CAGE does not allow point objectives with sum constraints as this could lead to a trade off between each point objective to meet any one of the sum constraints.

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



Objective type: Point objective

A point objective function is a CAGE model that is optimized at each point of the primary operating point set

Available models:

TQ_Model
NOXFLOW_Model

Objective type:

○ Minimize
● Maximize
○ Helper

Selected model: TQ_Model

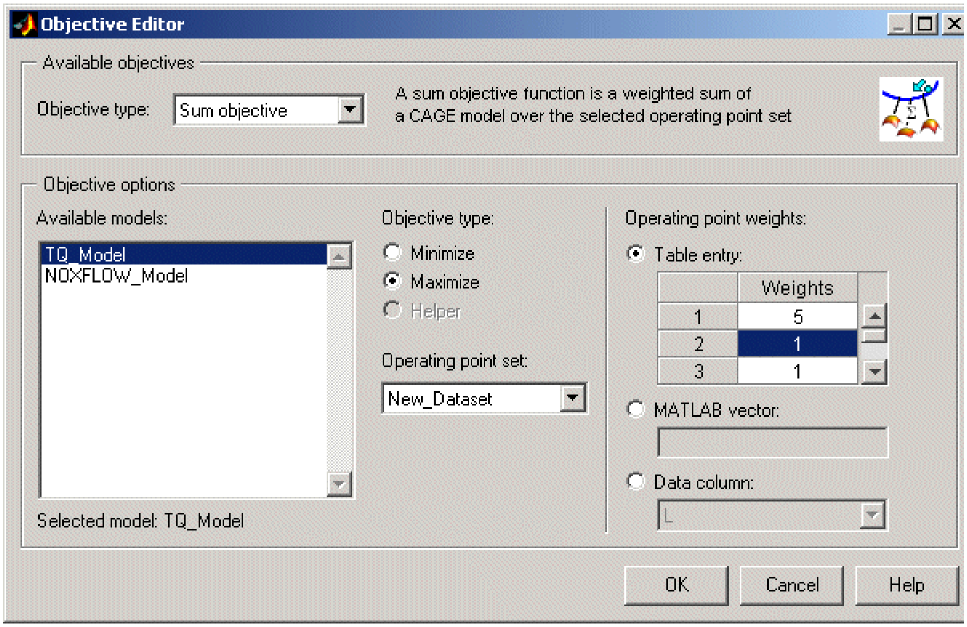
You can select `Point objective` or `Sum objective` from the **Objective Type** drop-down menu. Use sum objectives only for weighted sum optimizations; otherwise, use point objectives.

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.



Objective Editor

Available objectives

Objective type: Sum objective

A sum objective function is a weighted sum of
a CAGE model over the selected operating point set

Objective options

Available models:

TQ_Model
NOXFLOW_Model

Selected model: TQ_Model

Objective type:

○ Minimize
● Maximize
○ Helper

Operating point set: New_Dataset

Operating point weights:

● Table entry:

	Weights
1	5
2	1
3	1

○ MATLAB vector:

○ Data column: L

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:

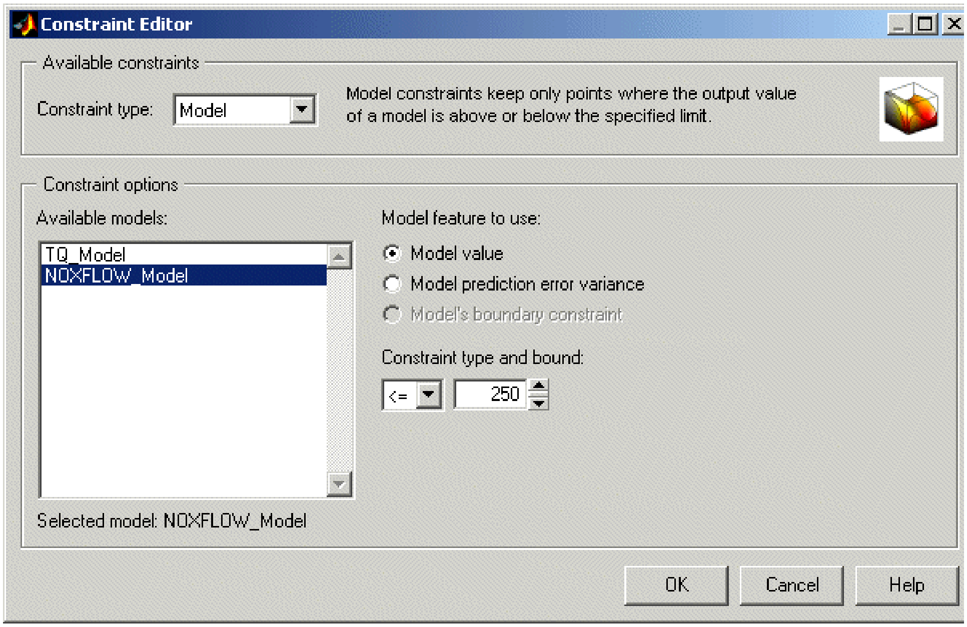
- Edit weights directly using the table entry radio button choice.
- Specify a MATLAB vector.
- Specify a data column to supply the weights.

This is the same process as selecting weights for the **Weighted Pareto View**. See “Weighted Pareto View” on page 11-30. For a tutorial example of a sum optimization, see “Sum Optimization” on page 6-33 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. You can have a mixture of point and sum constraints — but if you add a sum constraint you must also make all your objectives sum objectives or the optimization problem cannot be evaluated.

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.



Constraint Editor

Available constraints

Constraint type: Model

Model constraints keep only points where the output value of a model is above or below the specified limit.

Constraint options

Available models:

TQ_Model
NOXFLOW_Model

Model feature to use:

● Model value
○ Model prediction error variance
○ Model's boundary constraint

Constraint type and bound:

<= 250

Selected model: NOXFLOW_Model

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.

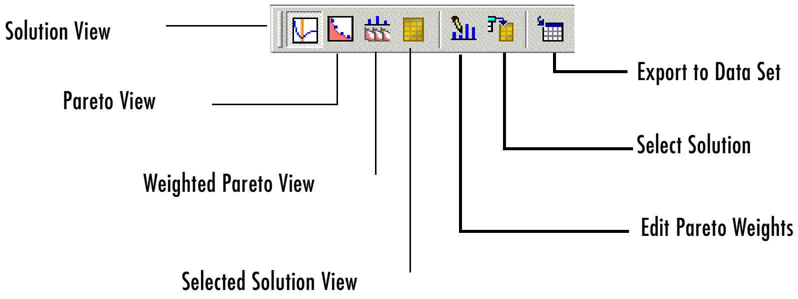
Model. Select a model. You can use the radio buttons to choose **Model value** or **Model prediction error variance** (PEV) to define your constraint, then enter a value in the edit box. Select the operator to define whether the optimization output should be constrained to be greater than or less than 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 11-22 and the tutorial "Sum Optimization" on page 6-33 for more information.

Optimization Output View

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 view determine what is displayed. The first view (shown by default) is the **Solution** view.

Optimization Output View Toolbar



Export to Data Set — Exports the table visible in the current view only to a new data set (the name of the data set is taken from the name of the optimization node). Also in the **Solution** menu. Use this if you want to fill tables with your selected solutions in multiobjective optimization. See "Using Optimization Results to Fill Tables" on page 6-16.

See

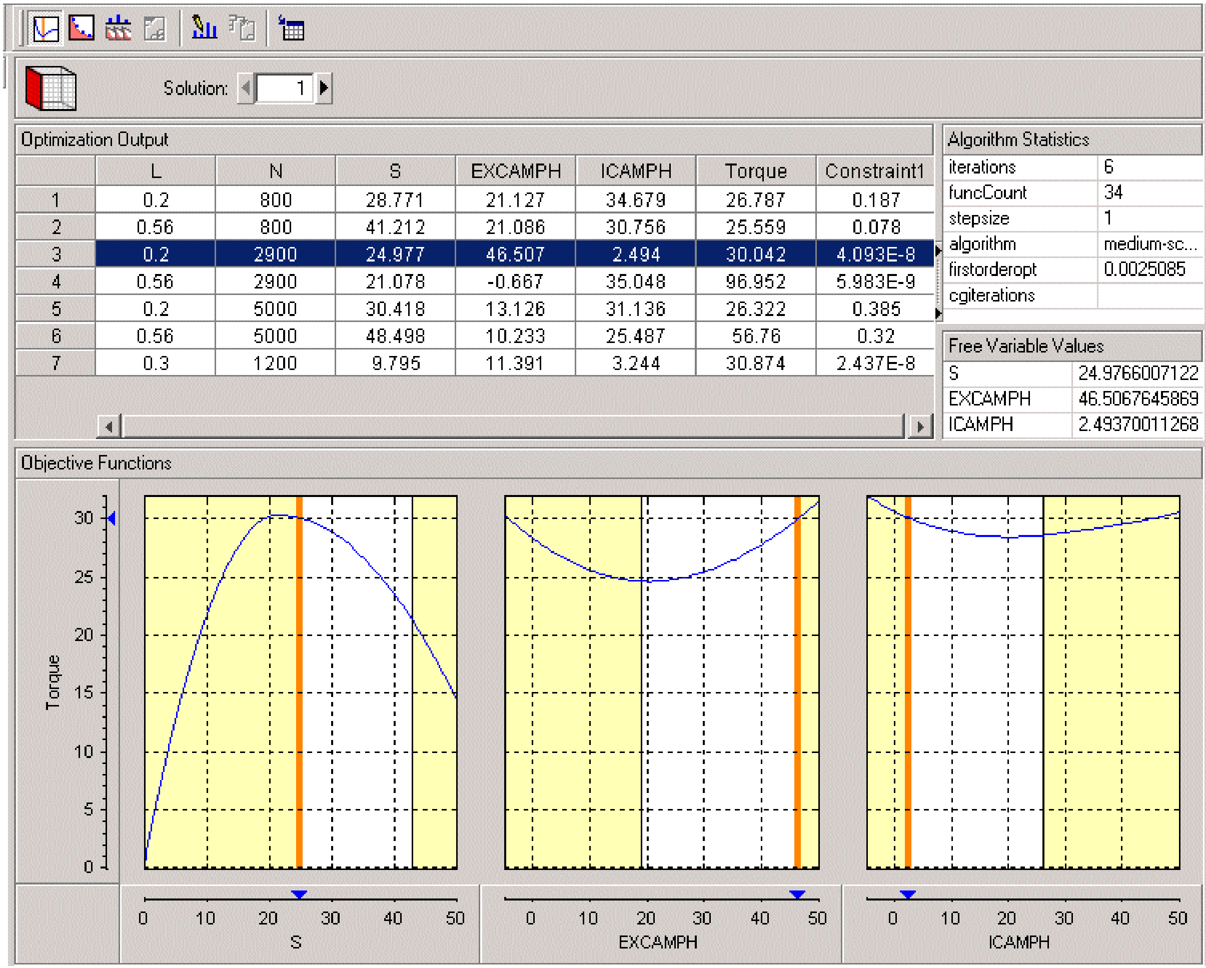
- "Solution View" on page 11-26
- "Pareto View" on page 11-28
- "Weighted Pareto View" on page 11-30
- "Selected Solution View" on page 11-32

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.



Solution: 1

Optimization Output

	L	N	S	EXCAMPH	ICAMPH	Torque	Constraint1
1	0.2	800	28.771	21.127	34.679	26.787	0.187
2	0.56	800	41.212	21.086	30.756	25.559	0.078
3	0.2	2900	24.977	46.507	2.494	30.042	4.093E-8
4	0.56	2900	21.078	-0.667	35.048	96.952	5.983E-9
5	0.2	5000	30.418	13.126	31.136	26.322	0.385
6	0.56	5000	48.498	10.233	25.487	56.76	0.32
7	0.3	1200	9.795	11.391	3.244	30.874	2.437E-8

Algorithm Statistics

iterations	6
funcCount	34
stepsize	1
algorithm	medium-sc...
firstorderopt	0.0025085
cgiterations	

Free Variable Values

S	24.9766007122
EXCAMPH	46.5067645869
ICAMPH	2.49370011268

Objective Functions

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.

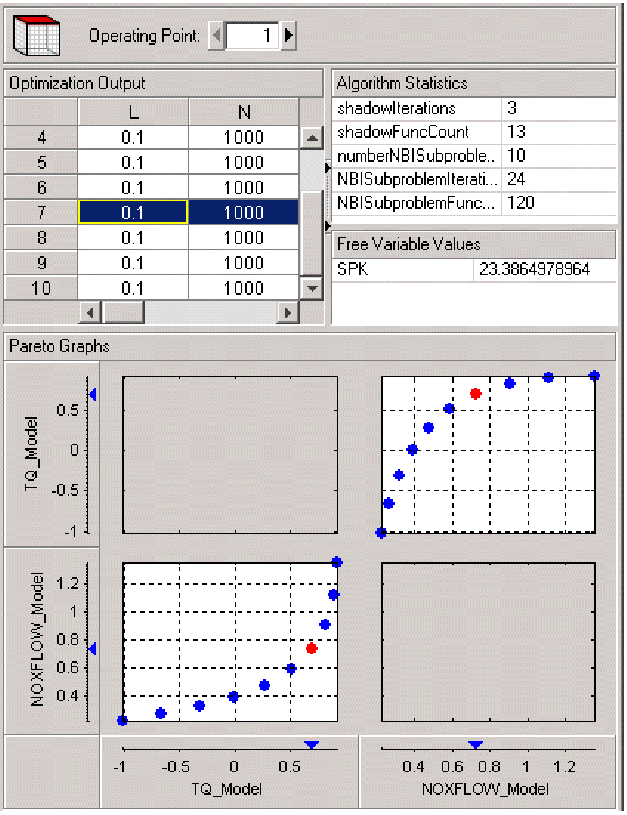
The table shows the selected solution at all operating points. Note that if you export the output to a data set (using the toolbar button) it is the current table that is exported. The graphs show the objective functions at the selected operating point, with the solution value in red.

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 11-32.

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.



Operating Point: 1

Optimization Output

	L	N
4	0.1	1000
5	0.1	1000
6	0.1	1000
7	0.1	1000
8	0.1	1000
9	0.1	1000
10	0.1	1000

Algorithm Statistics

shadowIterations	3
shadowFuncCount	13
numberNBISubproble...	10
NBISubproblemIterati...	24
NBISubproblemFunc...	120

Free Variable Values

SPK	23.3864978964

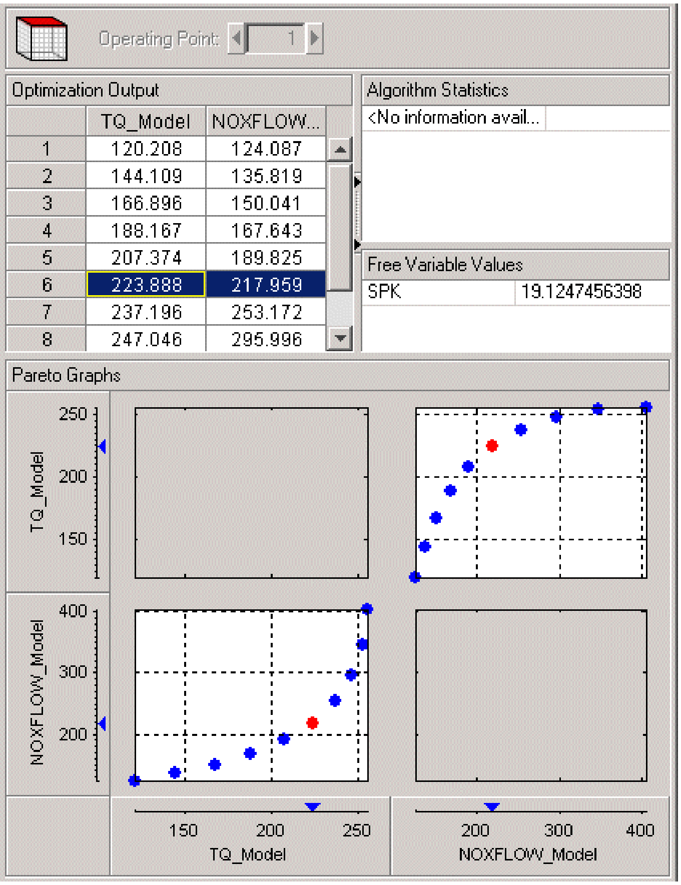
Pareto Graphs

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 11-32. 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_Model` 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.



Optimization Output

	TQ_Model	NOXFLOW...
1	120.208	124.087
2	144.109	135.819
3	166.896	150.041
4	188.167	167.643
5	207.374	189.825
6	223.888	217.959
7	237.196	253.172
8	247.046	295.996

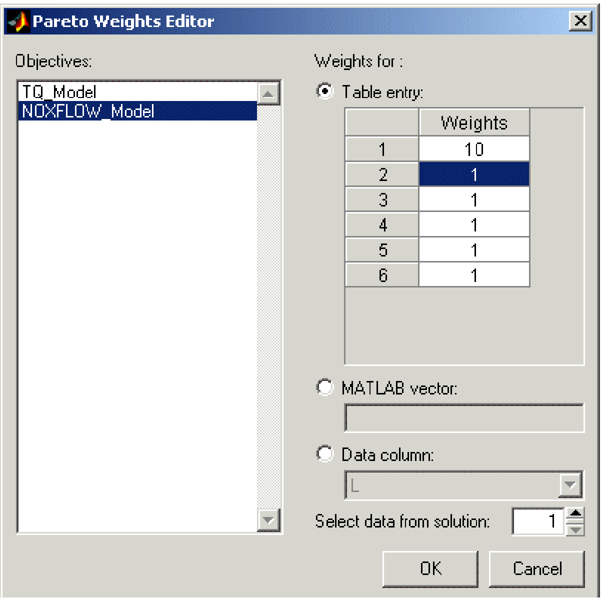
Algorithm Statistics

<No information avail...

Free Variable Values

SPK	19.1247456398

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.



Pareto Weights Editor

Objectives:

- TQ_Model
- NOXFLOW_Model

Weights for :

● Table entry:

	Weights
1	10
2	1
3	1
4	1
5	1
6	1

○ MATLAB vector:

○ Data column: L

Select data from solution: 1

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.

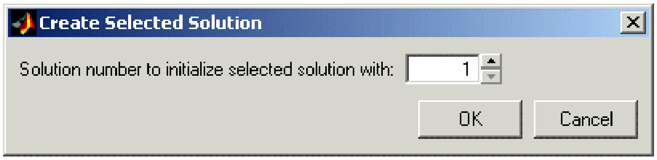
Selected Solution View

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**.

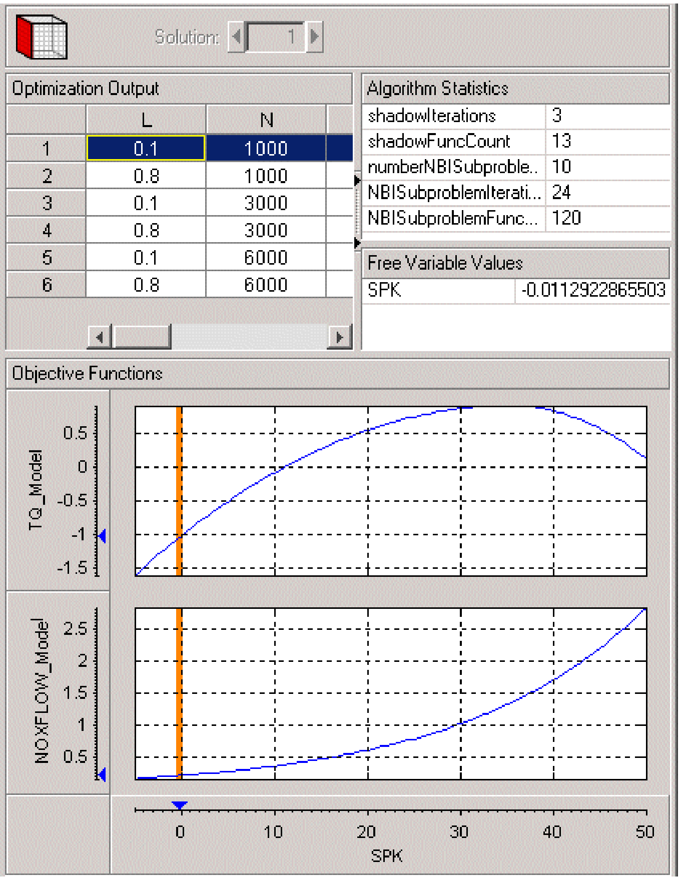
2 A dialog called **Create Selected Solution** appears. The default 1 initializes the first solution for each operating point as the selected solution. You can edit the solution number here if you want. For example if you select 4, solution number 4 is initialized as the best solution for every operating point. When you click **OK**, the toolbar buttons for the **Selected Solutions** view and **Select Solution** are enabled.


Create Selected Solution

Solution number to initialize selected solution with: 1

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.



Solution: 1

Optimization Output

	L	N
1	0.1	1000
2	0.8	1000
3	0.1	3000
4	0.8	3000
5	0.1	6000
6	0.8	6000

Algorithm Statistics

shadowIterations	3
shadowFuncCount	13
numberNBISubproble..	10
NBISubproblemIterati...	24
NBISubproblemFunc...	120

Free Variable Values

SPK	-0.0112922865503

Objective Functions

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 in the Optimization Output view 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 **Extrapolation Regions –> Add Selection** or use the toolbar button (to add selection to extrapolation regions).

3 In the tradeoff view, click a cell in a region.

4 To apply optimization: select **Inputs –> 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.

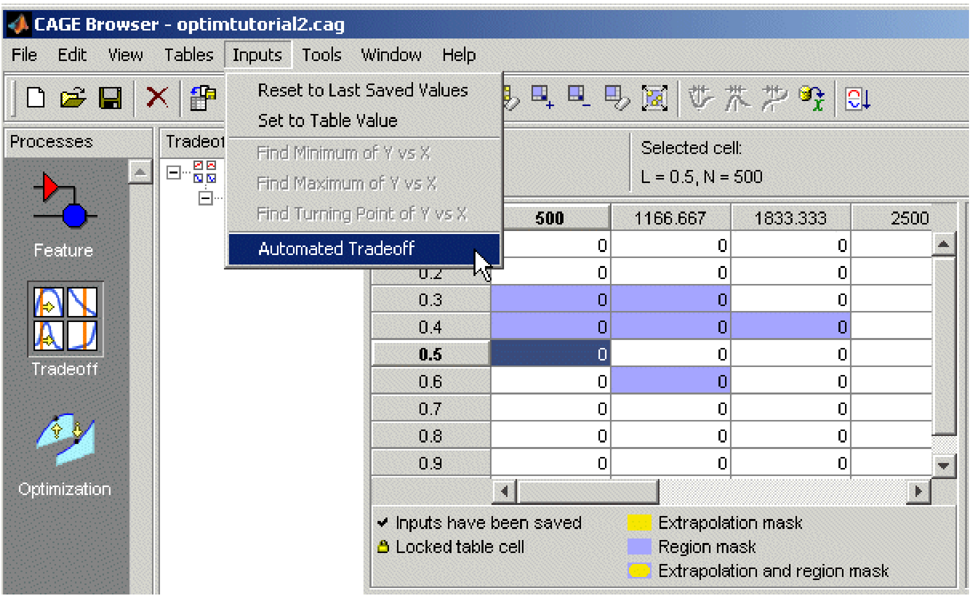
The cell/region is made into a data set of operating points that is linked to be the primary operating point set in the optimization. If the optimization object does not already have a primary operating set defined, then a new one is created. Note that any existing operating point set is reset to the previous state at the end of the automated tradeoff.

5 The optimization is run as if you were clicking **Run** from the Optimization view. 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 in the tree). As for any other optimization you can use the Optimization Output views to investigate the output. See "Optimization Output View" on page 11-26.



Selected cell:
L = 0.5, N = 500

	500	1166.667	1833.333	2500
	0	0	0	
0.2	0	0	0	
0.3	0	0	0	
0.4	0	0	0	
0.5	0	0	0	
0.6	0	0	0	
0.7	0	0	0	
0.8	0	0	0	
0.9	0	0	0	

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 11-35) 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 11-40 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 6-40 in the optimization tutorial.
- "Optimization Template" on page 11-47 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 6-40 describes the process of using the worked example.
- "About the Worked Example Optimization Algorithm" on page 11-44 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 6-48 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 11-47.

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.

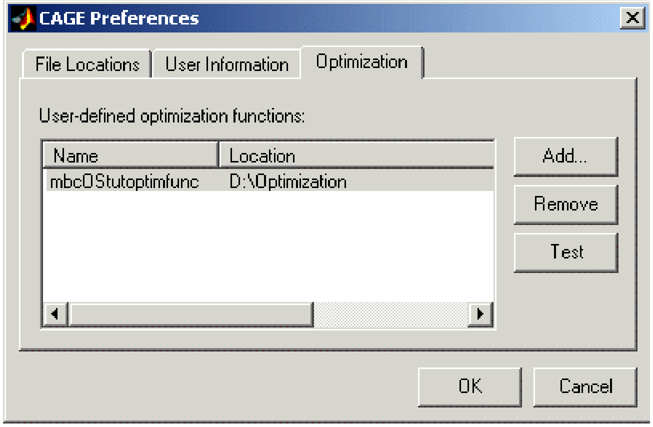
Checking User-Defined Optimizations into CAGE

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 11-5.

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.



CAGE Preferences

User-defined optimization functions:

Name	Location
mbcOStutoptimfunc	D:\Optimization

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.

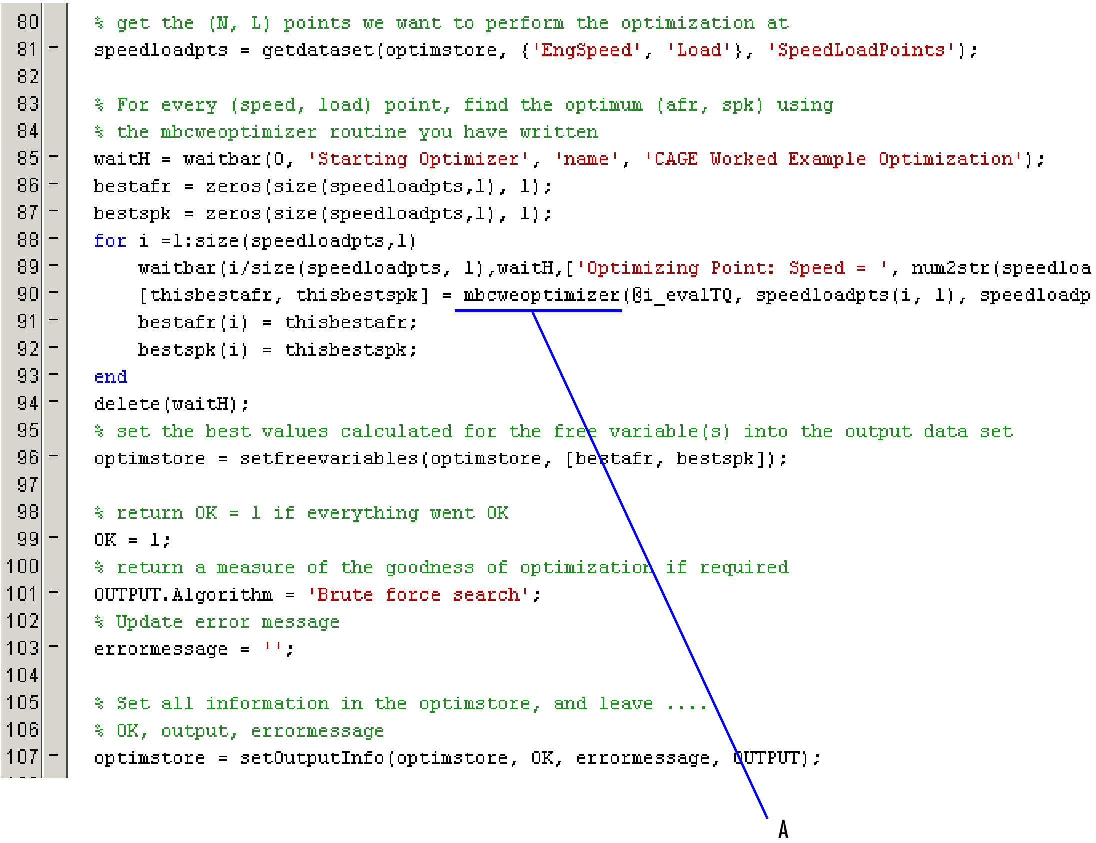
The Structure of the Worked Example

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.



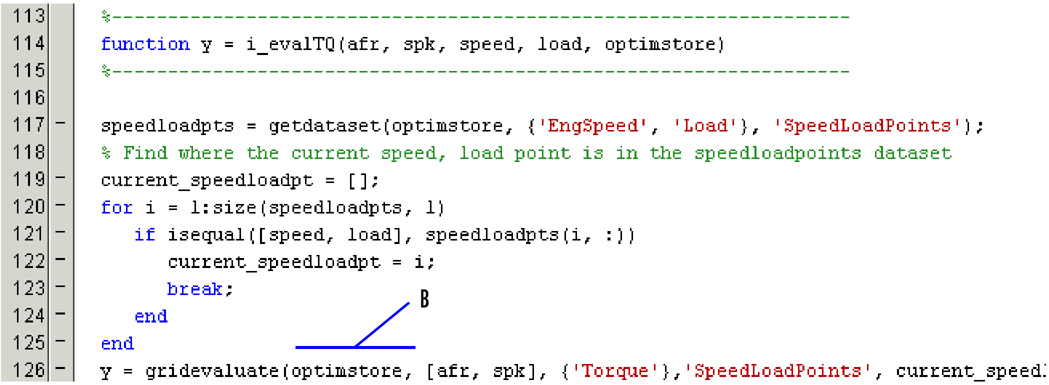
```
% get the (N, L) points we want to perform the optimization at
speedloadpts = getdataset(optimstore, {'EngSpeed', 'Load'}, 'SpeedLoadPoints');

% For every (speed, load) point, find the optimum (afr, spk) using
% the mbcweoptimizer routine you have written
waitH = waitbar(0, 'Starting Optimizer', 'name', 'CAGE Worked Example Optimization');
bestafr = zeros(size(speedloadpts,1), 1);
bestspk = zeros(size(speedloadpts,1), 1);
for i =1:size(speedloadpts,1)
    waitbar(i/size(speedloadpts, 1),waitH,['Optimizing Point: Speed = ', num2str(speedloa
    [thisbestafr, thisbestspk] = mbcweoptimizer(@i_evalTQ, speedloadpts(i, 1), speedloadp
    bestafr(i) = thisbestafr;
    bestspk(i) = thisbestspk;
end
delete(waitH);
% set the best values calculated for the free variable(s) into the output data set
optimstore = setfreevariables(optimstore, [bestafr, bestspk]);

% return OK = 1 if everything went OK
OK = 1;
% return a measure of the goodness of optimization if required
OUTPUT.Algorithm = 'Brute force search';
% Update error message
errormessage = '';

% Set all information in the optimstore, and leave ....
% OK, output, errormessage
optimstore = setOutputInfo(optimstore, OK, errormessage, OUTPUT);
```

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.



```
%------------------------------------------------------------------
function y = i_evalTQ(afr, spk, speed, load, optimstore)
%------------------------------------------------------------------

speedloadpts = getdataset(optimstore, {'EngSpeed', 'Load'}, 'SpeedLoadPoints');
% Find where the current speed, load point is in the speedloadpoints dataset
current_speedloadpt = [];
for i = 1:size(speedloadpts, 1)
   if isequal([speed, load], speedloadpts(i, :))
      current_speedloadpt = i;
      break;
   end
end
y = gridevaluate(optimstore, [afr, spk], {'Torque'},'SpeedLoadPoints', current_speedl
```

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 6-48. 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 11-50 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 11-49.

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	`evaluate`
Grid evaluation of optimization objectives and constraints	`gridEvaluate`
Evaluate prediction error variance (PEV)	`pevEvaluate`
Grid evaluation of prediction error variance (PEV)	`gridPevEvaluate`
Get optimization properties	`get`
Get values from optimization operating point sets	`getDataset`
Get the initial free values for the optimization	`getInitFreeVal`
Get the number of rows in an optimization operating point set	`getNumRowsInDataset`
Get output information for the optimization	`getOutputInfo`
Get optimization parameter	`getParam`
Set the optimal values of the free variables	`setFreeVariables`
Set output information for the optimization	`setOutputInfo`

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	`getEnabled`
Return the optimization free variable labels	`getFreeVariables`
Return the current usage of free variables	`getFreeVariablesMode`
Get the current name label for the optimization function	`getName`
Return information about the optimization objectives	`getObjectives`
Return the current usage of objective functions	`getObjectivesMode`
Return information about the optimization operating point sets	`getOperatingPointSets`
Return the current usage of operating point sets	`getOperatingPointsMode`
Return information about the optimization parameters	`getParameters`
Set how the optimization constraints are to be used	`setConstraintsMode`
Provide a description for the optimization function	`setDescription`
Set the enabled status for this optimization function	`setEnabled`
Set how the optimization free variables are used	`setFreeVariablesMode`
Provide a name label for an optimization function	`setName`

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 11-49
- “Methods of cgoptimoptions” on page 11-50

Alphabetical List of Functions

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 11-56, pevEvaluate on page 11-59

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 : `O1`, `O2`

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

```
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 11-55

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 11-60

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 11-59

Purpose

Get optimization properties

Syntax

```
V = get(optimstore, 'PropertyName')
```

Description

Returns the value of the specified property in the optimization. The properties are as follows:

`NumFreeVariables`: 'Number of free variables in this optimization'

`NumDataSets`: 'Number of data sets in this optimization'

`NumObjectiveFuncs`:'Number of objective functions in this optimization'

`NumConstraints`: 'Number of constraints in this optimization'

`A`: 'Matrix for linear constraints A*x <= b'

`b`: 'Vector for linear constraints A*x <= b'

`NonLinearConstraints`: 'Labels for the non-linear constraints'

`ObjectiveSums`: 'Labels for the objective sums'

`ConstraintSums`: 'Labels for the constraint sums'

`LB`: 'Lower bounds for free variables'

`UB`: 'Upper bounds for free variables’

`ObjectiveFuncTypes`: 'Character array of objective function types'

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.

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 11-73).

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 11-73

Purpose

Get the initial free values for the optimization

Syntax

```
XO = 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. `XO` 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 11-61, `setFreeVariablesMode` on page 11-91

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 11-68, "Worked Example Optimization" on page 6-40

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 11-74, “Worked Example Optimization” on page 6-40

setFreeVariables

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 6-40 for an example of creating an `output` structure.

See Also

`getOutputInfo` on page 11-65

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 >= `bound`

Similarly, if `boundtype` = `'lessthan'`, the model constraint takes the form

CAGE model <= `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 11-75, `addLinearConstraint` on page 11-70, `setConstraintsMode` on page 11-88

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) + ... + A(n)X(n) <= b
```

where `X(i)` is the i^{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 11-76, `addModelConstraint` on page 11-69, `setConstraintsMode` on page 11-88

addFreeVariable

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 11-91, `getFreeVariablesMode` on page 11-81, `getFreeVariables` on page 11-80

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 11-83, `setObjectivesMode` on page 11-93, `getObjectivesMode` on page 11-84

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 >= 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 11-85, `setOperatingPointsMode` on page 11-94, `getOperatingPointsMode` on page 11-86

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:

typestr	Value
`'number'`	Scalar, real number
`'list'`	Cell array of strings, one for each list member
`'boolean'`	True or false

See Also

`getParameters` on page 11-87, `getParam` on page 11-66

Purpose Get model constraint placeholder information

Syntax

```
out = getmodelconstraints (options)
```

Description Returns a structure array of information regarding the model constraints in the optimization. The structure has three fields: `label`, `boundtype`, and `bound`. See the help for `addModelConstraint` for more information on these fields.

See Also `addModelConstraint` on page 11-69, `setConstraintsMode` on page 11-88

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 11-70, `setConstraintsMode` on page 11-88

getConstraintsMode

Purpose Return the current usage of constraints

Syntax

```
mode = getConstraintsMode(options)
```

Description Returns a string describing how the optimization makes constraints available to the user. `mode` will be one of ‘`any` or ‘`fixed` .

See Also `setConstraintsMode` on page 11-88

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 11-89

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 11-79

getFreeVariables

Purpose

Return the optimization free variable labels

Syntax

```
labels=getFreeVariables(options)
```

Description

Returns the current placeholder labels for the free variables in the optimization. The labels are returned in a (`1-by-NFreeVar`) cell array, `labels`, where `NFreeVar` is the number of free variables that have been added to the optimization.

See Also

`addFreeVariable` on page 11-71, `setFreeVariablesMode` on page 11-91, `getFreeVariablesMode` on page 11-81

Purpose

Return the current usage of free variables

Syntax

```
mode= getFreeVariablesMode(options)
```

Description

Returns a string describing how the optimization makes free variables available to the user. `mode` is set to `any` or `fixed`.

See Also

`setFreeVariablesMode` on page 11-91

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 11-92

Purpose

Return information about the optimization objectives

Syntax

```
objinfo=getObjectives(options)
```

Description

Returns a structure array of information regarding the optimization objective functions. `objinfo(i).label` contains the label for the `i`th objective. A string defining the type of the `i`th objective (`max`, `min`, `min/max`, or `helper`) is stored in `objinfo(i).type`.

See Also

`addObjective` on page 11-72, `setObjectivesMode` on page 11-93, `getObjectivesMode` on page 11-84

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 11-93

Purpose

Return information about the optimization operating point sets

Syntax

```
getOperatingPointSets(options)
```

Description

Returns a structure array of information regarding the optimization operating point sets. The structure has two fields, `label` and `vars`. See the help for `addOperatingPointSet` for more information on these fields.

See Also

`addOperatingPointSet` on page 11-73, `setOperatingPointsMode` on page 11-94, `getOperatingPointsMode` on page 11-86

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` wil be one of ‘`default` , ‘`fixed` , or ‘`any` .

See Also `setOperatingPointsMode` on page 11-94

Purpose Return information about the optimization parameters

Syntax `getParameters(options)`

Description Returns a structure array containing information about the parameters that are defined for the optimization. Parameter information is returned in a structure with fields `label`, `typestr`, and `value`. See the help for `addParameter` for more information on these fields.

See Also `addParameter` on page 11-74

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 11-77, `addModelConstraint` on page 11-69, `addLinearConstraint` on page 11-70

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 11-78

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 `getEnabled` on page 11-79

Purpose

Set how the optimization free variables are used

Syntax

```
options = setFreeVariablesMode(options, modestr)
```

Description

Sets the mode that governs how the user is allowed to set up free variables for the optimization in the CAGE GUI.

When `modestr = 'any'`, the user is allowed to add any number of free variables.

When `modestr = 'fixed'`, the user is only allowed to use the number of free variables that are added by the user-defined optimization function.

See Also

`getFreeVariablesMode` on page 11-81, `addFreeVariable` on page 11-71

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 11-82

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 11-84, `addObjective` on page 11-72

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 11-86, `addOperatingPointSet` on page 11-73

12

Data Sets

This section includes the following topics:

Data Sets Views

The **Data Set** view has these main functions:

- Validating calibrations with experimental data
- Filling tables by reference to a set of experimental data
- Constructing operating point sets for running optimizations
- Investigating optimization results and using them to fill tables

For worked examples about data sets, see

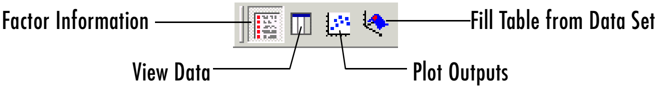
- "Tutorial: Data Sets" on page 4-1

  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.
- "Tutorial: Optimization and Automated Tradeoff" on page 6-1

  This tutorial includes using the output from an optimization to fill a table.

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**

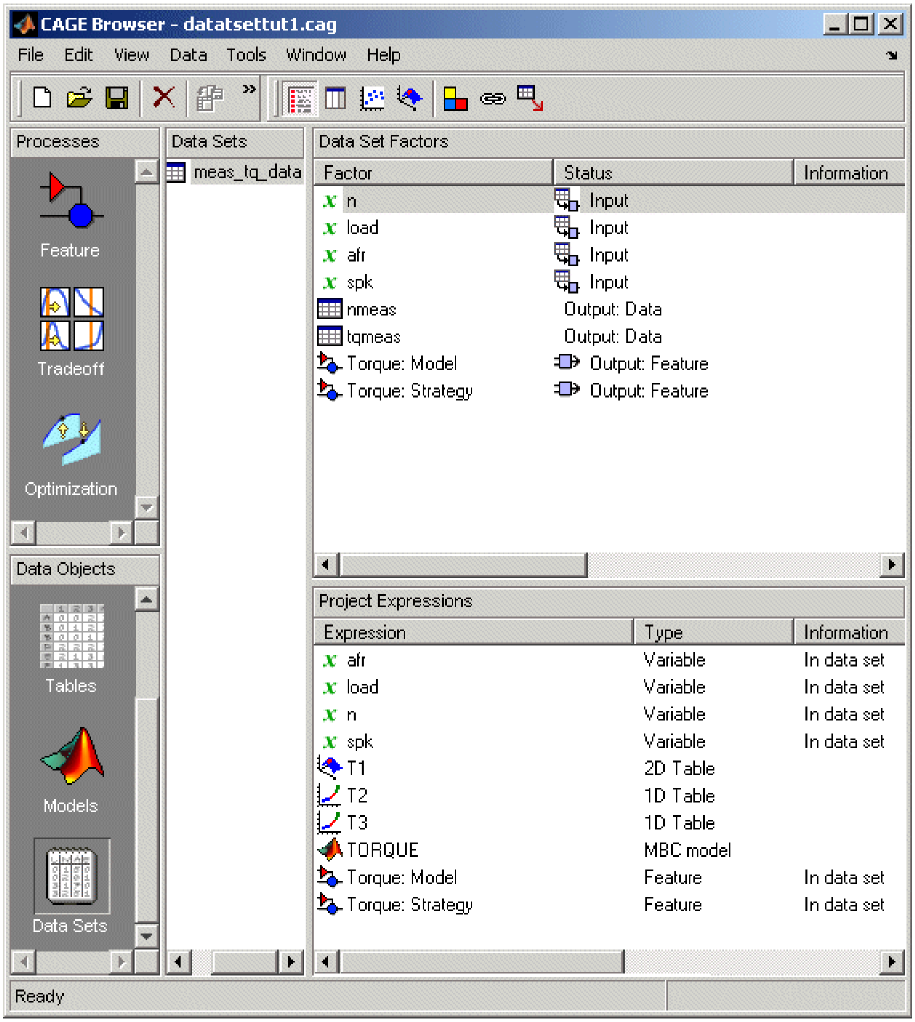
  List of all available project expressions, which can be added to the data set for display and evaluation.

- **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.



Factor	Status	Information
n	Input	
load	Input	
afr	Input	
spk	Input	
nmeas	Output: Data	
tqmeas	Output: Data	
Torque: Model	Output: Feature	
Torque: Strategy	Output: Feature	

Expression	Type	Information
afr	Variable	In data set
load	Variable	In data set
n	Variable	In data set
spk	Variable	In data set
T1	2D Table	
T2	1D Table	
T3	1D Table	
TORQUE	MBC model	
Torque: Model	Feature	In data set
Torque: Strategy	Feature	In data set

Setting Up Data Sets

The **Data Sets** view displays the strategies, tables, and models, etc., as a list of factors in the default **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 12-5.)
- Import the values from a table in your CAGE session. (See “Importing Data from a Table in Your Session” on page 12-7.)
- Specify the factors and their values manually. (See “Specifying the Factors Manually” on page 12-8.)

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.

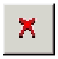
To import experimental data,

1 Select **File –> Import –> Data**.

2 In the file browser, select the correct file to import.

   This opens the **Data Set Import Wizard**.

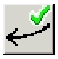
3 Discard any columns of data you do not want to import by selecting the column and clicking the button shown.



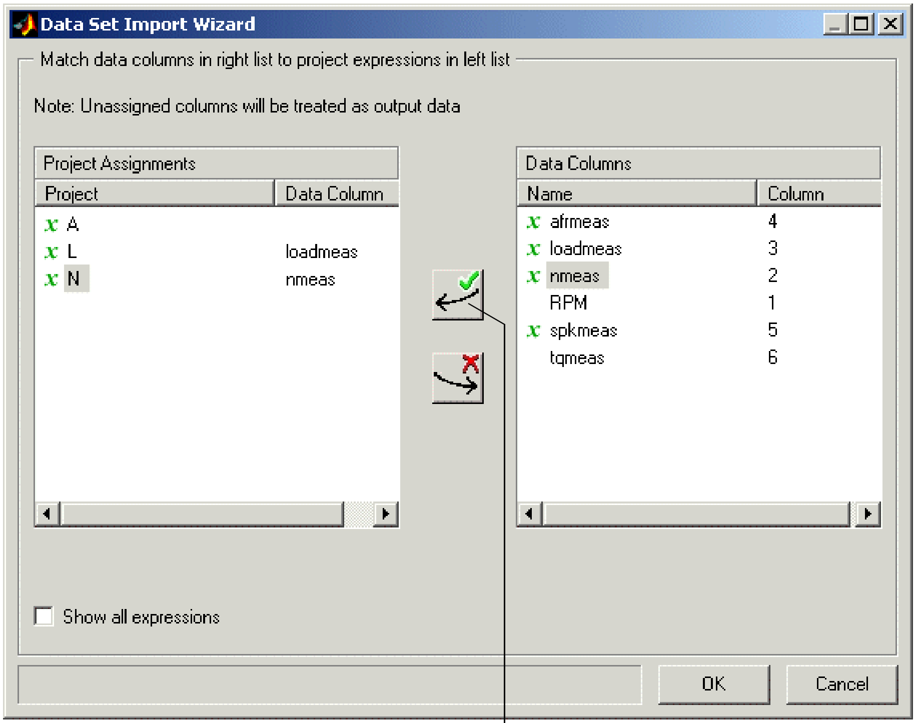
4 Click **Next**.

   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.



Match data columns in right list to project expressions in left list

Note: Unassigned columns will be treated as output data

Project Assignments

Project	Data Column
x A	
x L	loadmeas
x N	nmeas

Data Columns

Name	Column
x afrmeas	4
x loadmeas	3
x nmeas	2
RPM	1
x spkmeas	5
tqmeas	6

□ Show all expressions

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 12-8.)

4 Build the grid. (See “Manually Setting Values of the Input Variables” on page 12-10.)

Once you have completed these steps you can view the data set.

This section describes

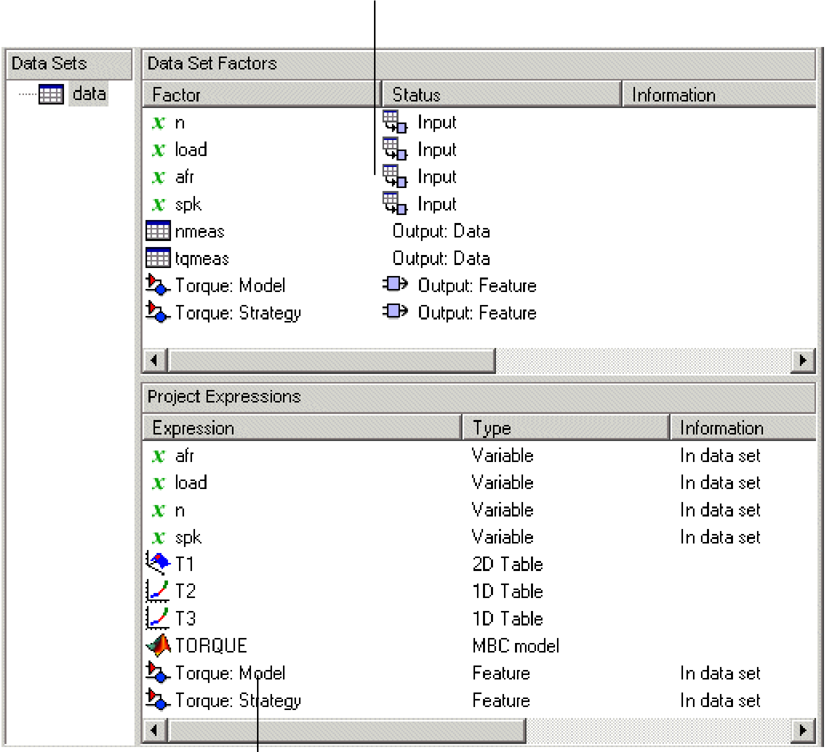
- “Selecting the Factors” on page 12-8
- “Manually Setting Values of the Input Variables” on page 12-10

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.

Factors in the current data set



Data Sets

- data

Data Set Factors

Factor	Status	Information
n	Input	
load	Input	
afr	Input	
spk	Input	
nmeas	Output: Data	
tqmeas	Output: Data	
Torque: Model	Output: Feature	
Torque: Strategy	Output: Feature	

Project Expressions

Expression	Type	Information
afr	Variable	In data set
load	Variable	In data set
n	Variable	In data set
spk	Variable	In data set
T1	2D Table	
T2	1D Table	
T3	1D Table	
TORQUE	MBC model	
Torque: Model	Feature	In data set
Torque: Strategy	Feature	In data set

Factors in the current project

You can use this view to add factors to or remove factors from the data set.

To add a factor to a data set,

- Right-click a factor and select **Add to Data Set** from the context menu.
- Alternatively, select the factor or factors that you want to add to the data set from the list in the lower **Project Expressions** pane, then select **Data –> Factors –> Add to Data Set**.

  To make multiple selections, use the standard **Shift+click** or **Ctrl+click**.

To remove a factor from a data set,

1 Select the factor or factors that you want to remove from the data set.

2 Right-click and select **Remove from Data Set**, or select the menu item **Data –> Factors –> Remove From Data Set**.

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.

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.

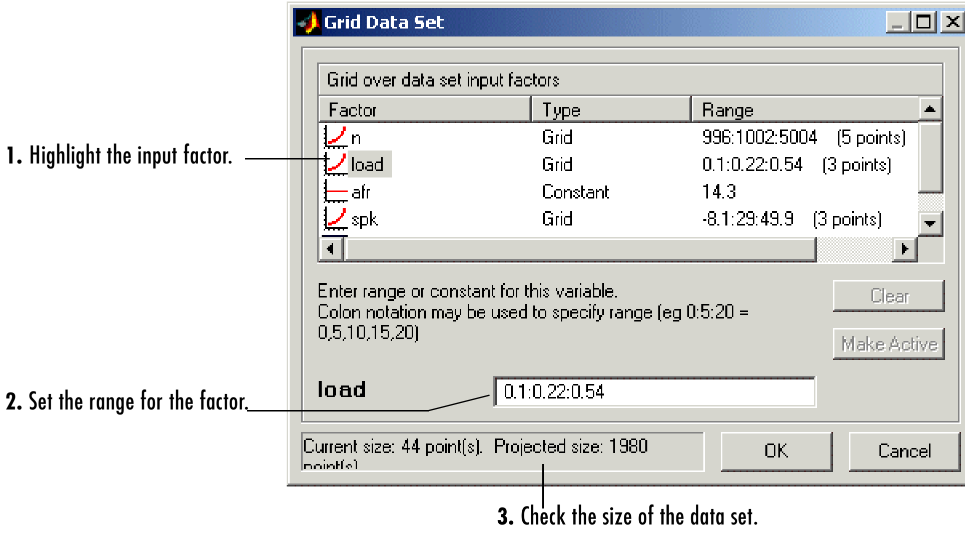
3 Set the grid for the factor.

   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**.



Grid over data set input factors

Factor	Type	Range
n	Grid	996:1002:5004 (5 points)
load	Grid	0.1:0.22:0.54 (3 points)
afr	Constant	14.3
spk	Grid	-8.1:29:49.9 (3 points)

Enter range or constant for this variable.
Colon notation may be used to specify range (eg 0:5:20 = 0,5,10,15,20)

load 0.1:0.22:0.54

Current size: 44 point(s). Projected size: 1980 point(s)

1. Highlight the input factor.

2. Set the range for the factor.

3. Check the size of the data set.

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**.

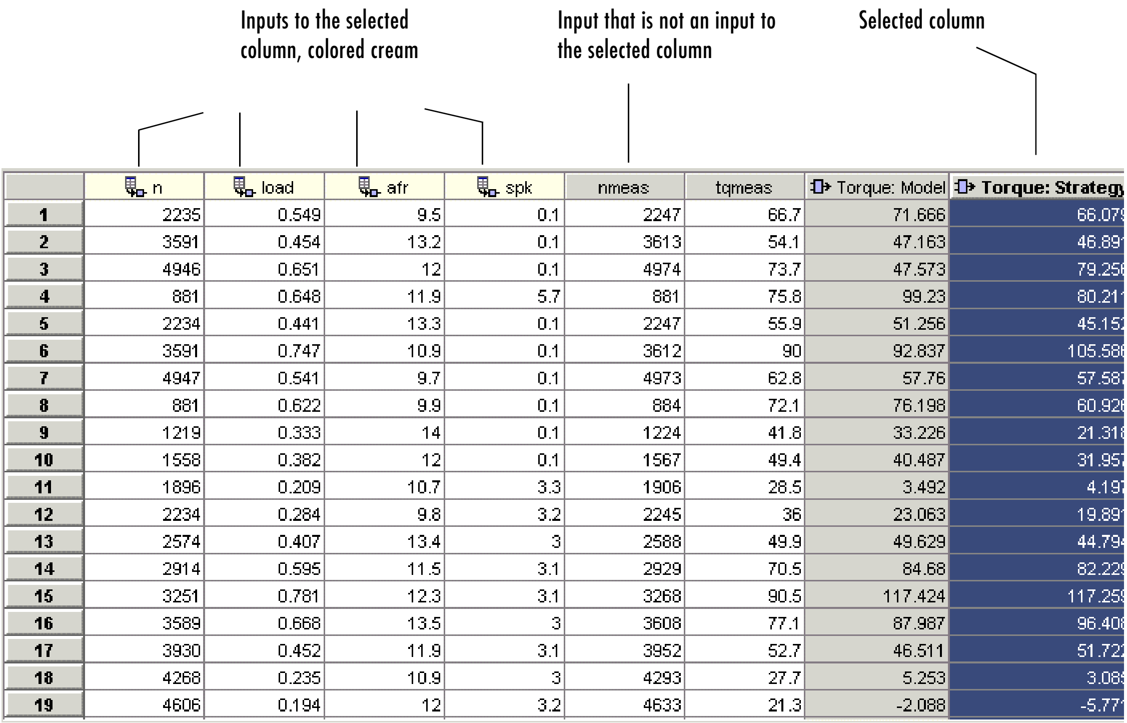
2 Select **Create Error** from the right-click menu on either column head.

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 12-5.



Inputs to the selected column, colored cream

Input that is not an input to the selected column

Selected column

	n	load	afr	spk	nmeas	tqmeas	Torque: Model	Torque: Strategy
1	2235	0.549	9.5	0.1	2247	66.7	71.666	66.079
2	3591	0.454	13.2	0.1	3613	54.1	47.163	46.891
3	4946	0.651	12	0.1	4974	73.7	47.573	79.258
4	881	0.648	11.9	5.7	881	75.8	99.23	80.211
5	2234	0.441	13.3	0.1	2247	55.9	51.256	45.152
6	3591	0.747	10.9	0.1	3612	90	92.837	105.588
7	4947	0.541	9.7	0.1	4973	62.8	57.76	57.587
8	881	0.622	9.9	0.1	884	72.1	76.198	60.928
9	1219	0.333	14	0.1	1224	41.8	33.226	21.318
10	1558	0.382	12	0.1	1567	49.4	40.487	31.957
11	1896	0.209	10.7	3.3	1906	28.5	3.492	4.197
12	2234	0.284	9.8	3.2	2245	36	23.063	19.891
13	2574	0.407	13.4	3	2588	49.9	49.629	44.794
14	2914	0.595	11.5	3.1	2929	70.5	84.68	82.229
15	3251	0.781	12.3	3.1	3268	90.5	117.424	117.259
16	3589	0.668	13.5	3	3608	77.1	87.987	96.408
17	3930	0.452	11.9	3.1	3952	52.7	46.511	51.722
18	4268	0.235	10.9	3	4293	27.7	5.253	3.089
19	4606	0.194	12	3.2	4633	21.3	-2.088	-5.771

Columns are color coded by factor type:

- Input factors are white.
- Output factors are gray.

Selecting an output column highlights the input columns associated with it by turning the header cells cream.

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

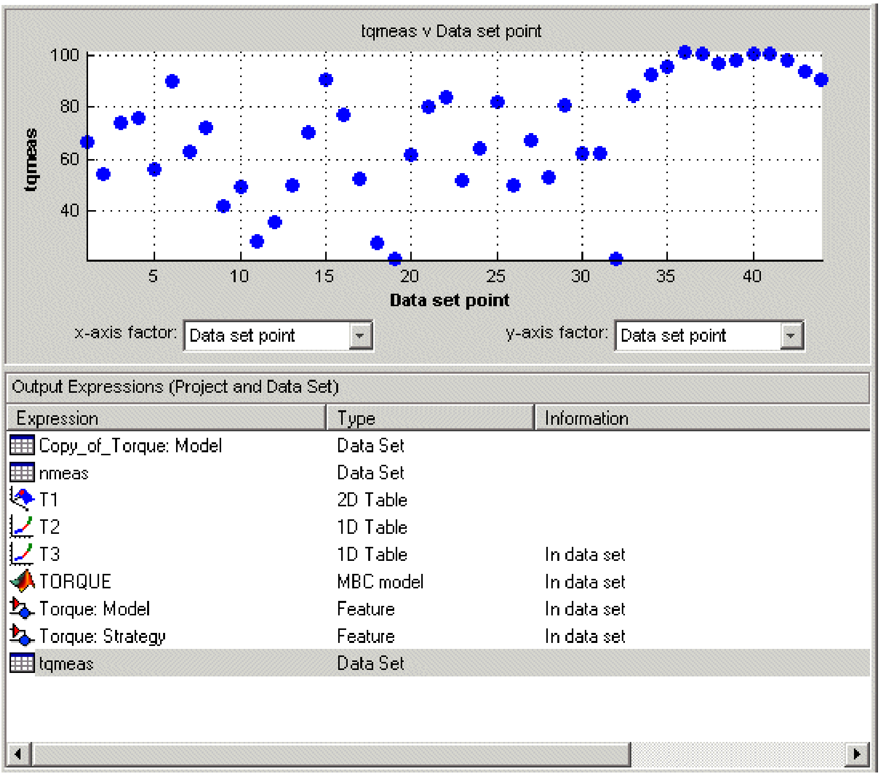
Use this to plot the outputs of your data sets.

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.



x-axis factor: Data set point

y-axis factor: Data set point

Output Expressions (Project and Data Set)

Expression	Type	Information
Copy_of_Torque: Model	Data Set	
nmeas	Data Set	
T1	2D Table	
T2	1D Table	
T3	1D Table	In data set
TORQUE	MBC model	In data set
Torque: Model	Feature	In data set
Torque: Strategy	Feature	In data set
tqmeas	Data Set	

3 Use the pop-up menus below the plot to change the factors displayed.

To zoom in on an area of interest,

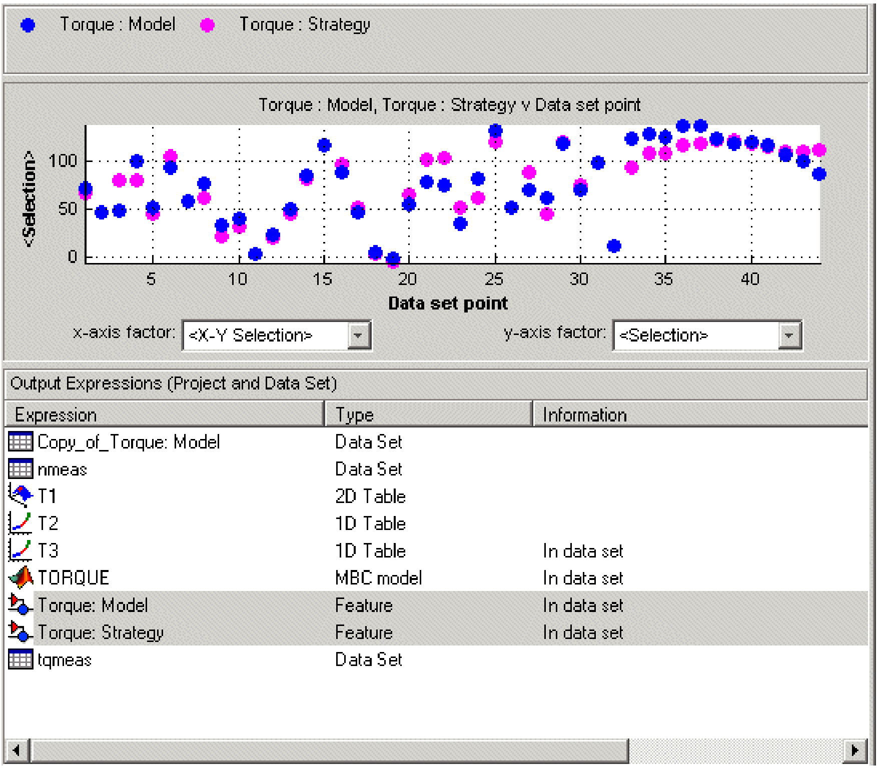
- Press both mouse buttons simultaneously and drag a rectangle; double-click the graph to return to full size.

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



x-axis factor: <X-Y Selection>

y-axis factor: <Selection>

Output Expressions (Project and Data Set)

Expression	Type	Information
Copy_of_Torque: Model	Data Set	
nmeas	Data Set	
T1	2D Table	
T2	1D Table	
T3	1D Table	In data set
TORQUE	MBC model	In data set
Torque: Model	Feature	In data set
Torque: Strategy	Feature	In data set
tqmeas	Data Set	

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

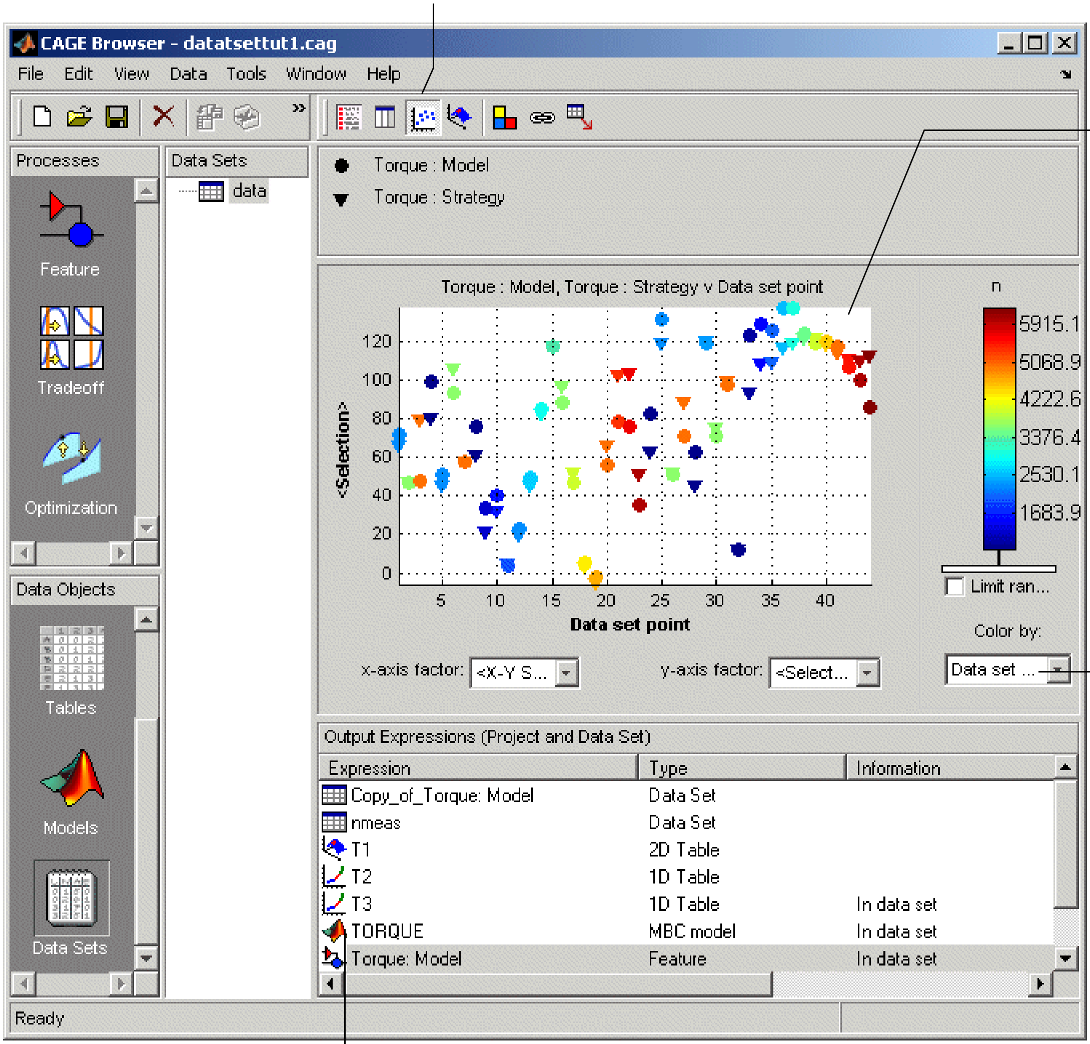
1 Select **View –> Plot** or click .

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.



Output Expressions (Project and Data Set)

Expression	Type	Information
Copy_of_Torque: Model	Data Set	
nmeas	Data Set	
T1	2D Table	
T2	1D Table	
T3	1D Table	In data set
TORQUE	MBC model	In data set
Torque: Model	Feature	In data set

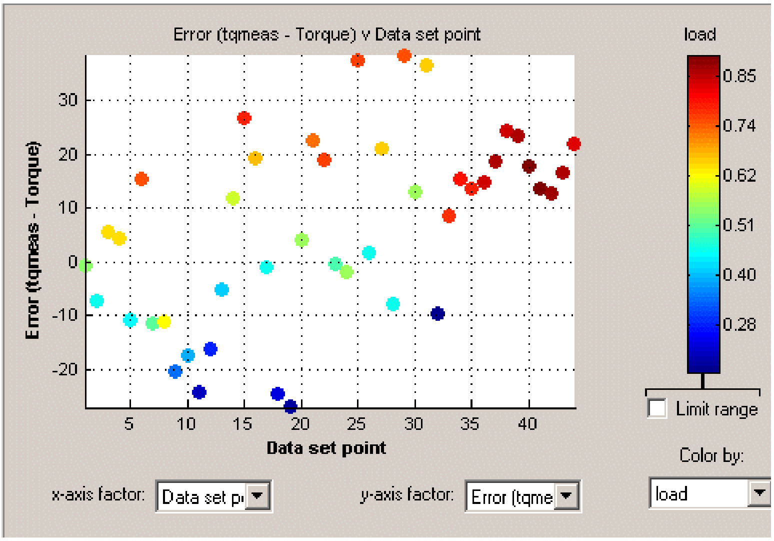
3. Select **Color by Value** from the right-click menu.

4. Select the correct variable.

2. Select the expression.

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.



□ Limit range

Color by: load

x-axis factor: Data set p

y-axis factor: Error (tqme

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, or right-click the plot and select **Limit Color Range**.

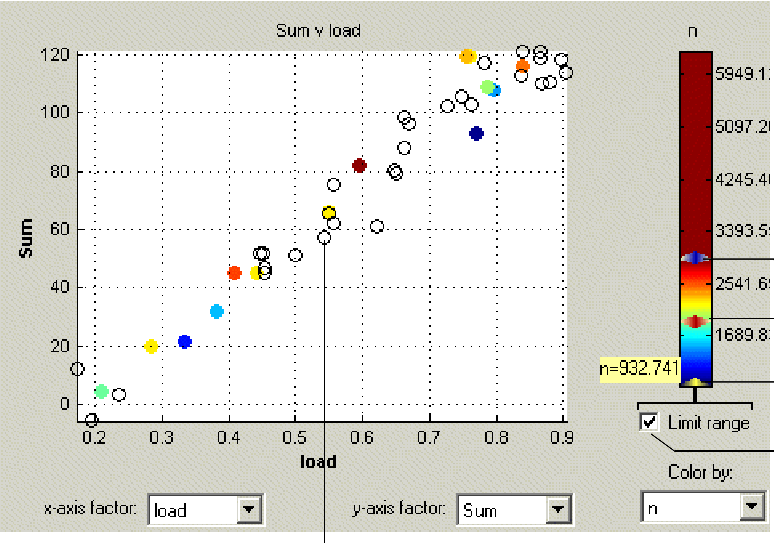
   Three limit markers appear in the color bar. The colors in the color bar are compressed within the limit markers. This increases the range of colors over the range you are interested in (between the limits), making it easier to see the distribution of points.

2 Adjust the maximum, midpoint, and minimum of the range by dragging the limit markers 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 **Exclude** to remove all points outside the limits from the display.
- Select **Color Outside Limits** to display all points in color, including those outside the limits. Points outside the limits are still colored, but only dark red or dark blue, depending on which end of the range they are.
- Select **No Color Outside Limits** to display the points as in the example shown. Points outside the limits are plotted as empty circles.



☑ Limit range

Color by: n

x-axis factor: load

y-axis factor: Sum

2. Adjust the range.

1. Select the **Limit range** box.

A point outside the range

Linking Factors in a Data Set

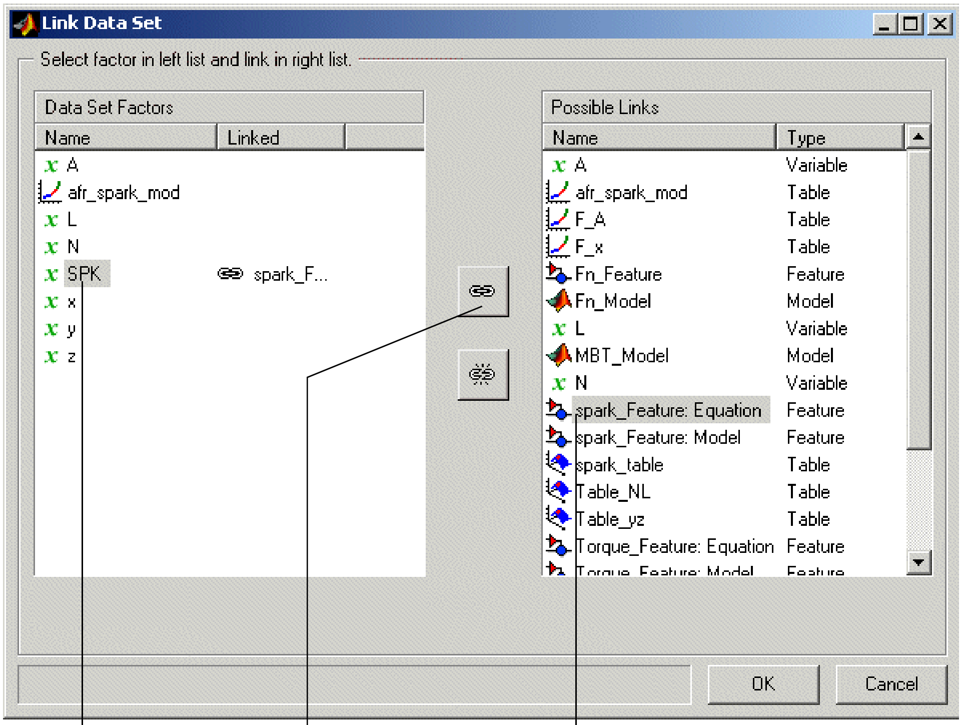
A factor can be linked to another. The factor then takes on the values of that other factor, overwriting the original values.

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.



Select factor in left list and link in right list.

Data Set Factors

Name	Linked
A	
afr_spark_mod	
L	
N	
SPK	spark_F...
x	
y	
z	

Possible Links

Name	Type
A	Variable
afr_spark_mod	Table
F_A	Table
F_x	Table
Fn_Feature	Feature
Fn_Model	Model
L	Variable
MBT_Model	Model
N	Variable
spark_Feature: Equation	Feature
spark_Feature: Model	Feature
spark_table	Table
Table_NL	Table
Table_yz	Table
Torque_Feature: Equation	Feature
Torque_Feature: Model	Feature

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 12-5

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 (or any data set, such as optimization output).

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,

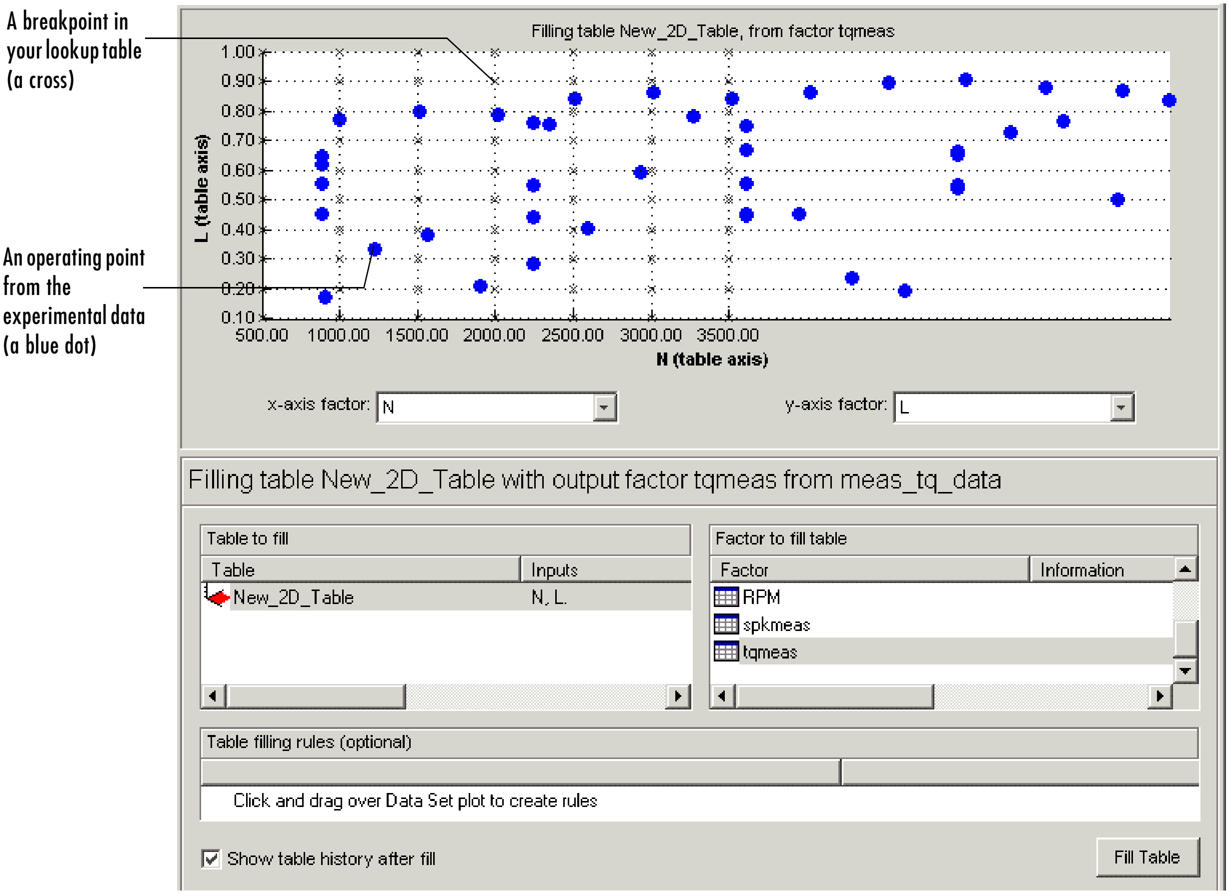
1. To view the **Table Filler** display, click (Fill Table From Data Set) in the toolbar; or select **View –> Table Filler**.

   You can use this display to specify the table you want to fill and the factor you want to use to fill it.

2. In the lower pane, select the table from the **Table to fill** list. This is the table that you want to fill.

3. Select the experimental data from the **Factor to fill table** list. This is the data that you want to use to fill the table.

   For example, see the following display.



A breakpoint in your lookup table (a cross)

An operating point from the experimental data (a blue dot)

x-axis factor: N

y-axis factor: L

Filling table New_2D_Table with output factor tqmeas from meas_tq_data

Table to fill

Table	Inputs
New_2D_Table	N, L

Factor to fill table

Factor	Information
RPM	
spkmeas	
tqmeas	

Table filling rules (optional)

Click and drag over Data Set plot to create rules

☑ Show table history after fill

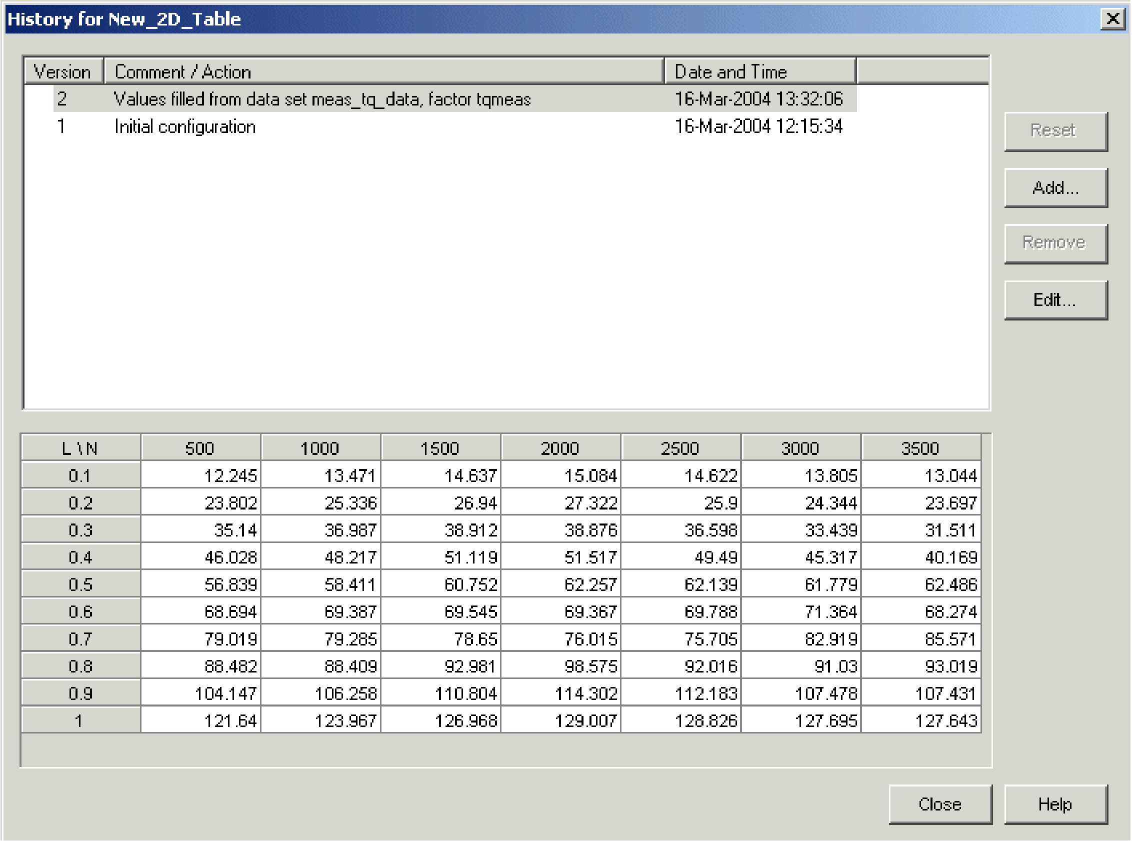
Fill Table

The upper pane displays the breakpoints of your table as crosses and the operating points where there is data as blue dots. Data sets display the points in the experimental data, not the values at the breakpoints. You can inspect the spread of the data compared to the breakpoints of your table before you fill the table.

4 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.

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.

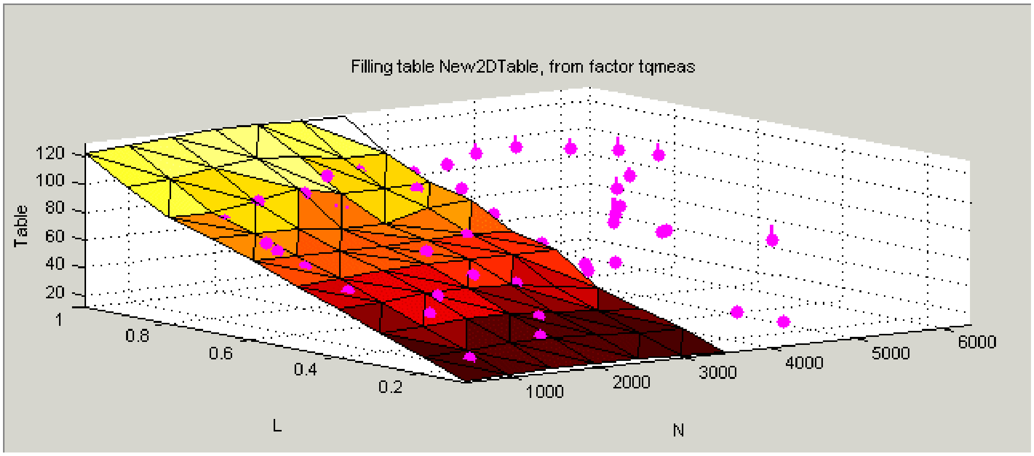


Version	Comment / Action	Date and Time
2	Values filled from data set meas_tq_data, factor tqmeas	16-Mar-2004 13:32:06
1	Initial configuration	16-Mar-2004 12:15:34

L \ N	500	1000	1500	2000	2500	3000	3500
0.1	12.245	13.471	14.637	15.084	14.622	13.805	13.044
0.2	23.802	25.336	26.94	27.322	25.9	24.344	23.697
0.3	35.14	36.987	38.912	38.876	36.598	33.439	31.511
0.4	46.028	48.217	51.119	51.517	49.49	45.317	40.169
0.5	56.839	58.411	60.752	62.257	62.139	61.779	62.486
0.6	68.694	69.387	69.545	69.367	69.788	71.364	68.274
0.7	79.019	79.285	78.65	76.015	75.705	82.919	85.571
0.8	88.482	88.409	92.981	98.575	92.016	91.03	93.019
0.9	104.147	106.258	110.804	114.302	112.183	107.478	107.431
1	121.64	123.967	126.968	129.007	128.826	127.695	127.643

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.

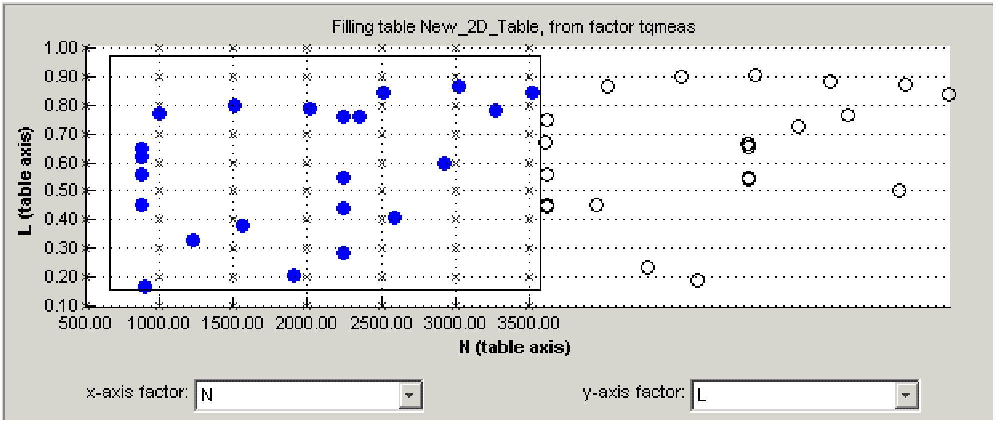
For example, you might want to fill a lookup table that has a range of operating points that is smaller than the range of the experimental data.

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.



x-axis factor: N

y-axis factor: L

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

Select **Data –> Table Fill** to reach 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.

13

Calibration Manager

This section includes the following topics:

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.

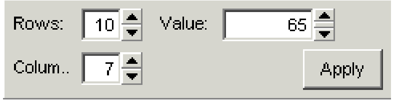
Note that you can enter the required inputs, number of rows and columns and an initial value for table cells when you add a new table using the **File –> New** menu items. See "Adding and Deleting Tables" on page 15-3. You can use the Calibration Manager to change the sizes, values and precision of tables.

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 on the left and select initial values for each cell in the table.

3 Click **Apply**.



Rows: 10 Value: 65

Colum.. 7

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 9-15 and "Initializing Table Values" on page 9-13.

---

4 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 that empty data is filtered out and any empty variables will not appear.

Note `tutorialcal.mat` is an example calibration file in the `mbctraining` folder.

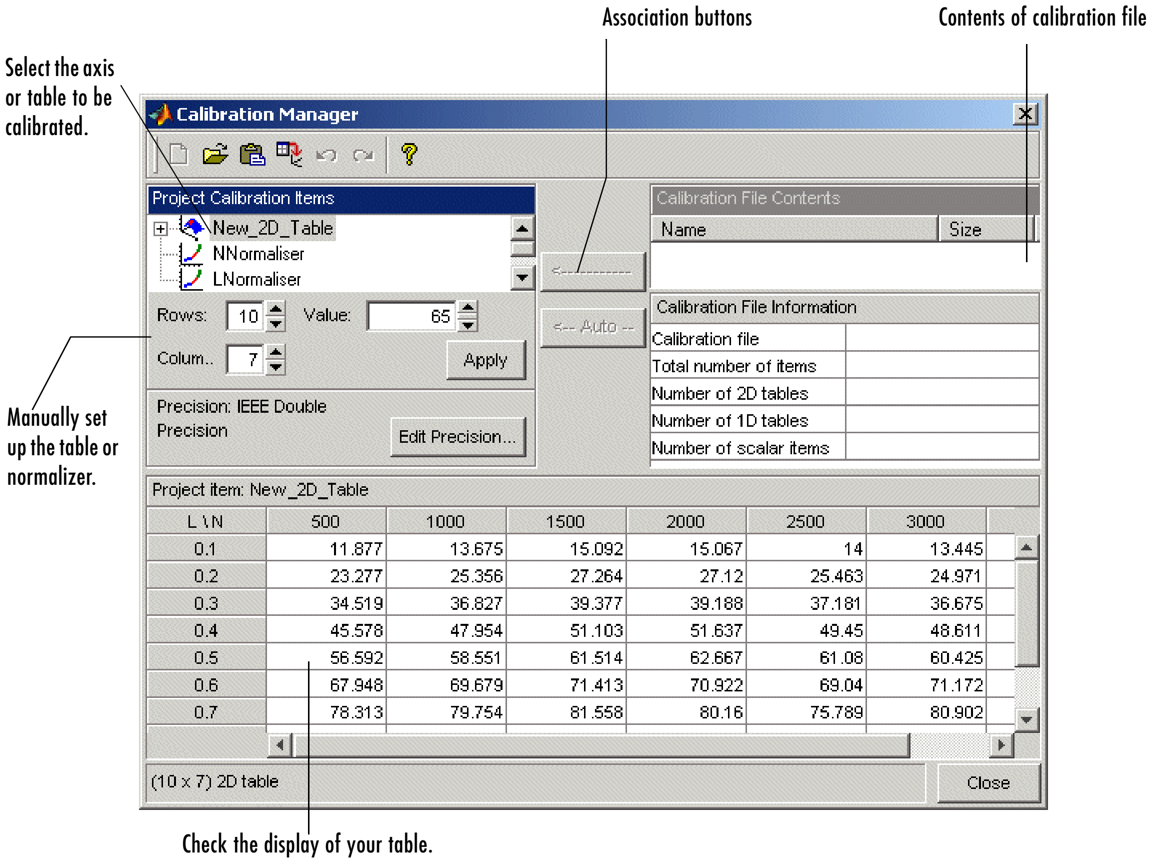
---

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 <--- Auto ---.

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.

Check the display of your table.

Rows: 10 Value: 65

Colum.. 7

Precision: IEEE Double Precision

Calibration File Information	
Calibration file	
Total number of items	
Number of 2D tables	
Number of 1D tables	
Number of scalar items	

Project item: New_2D_Table

L \ N	500	1000	1500	2000	2500	3000
0.1	11.877	13.675	15.092	15.067	14	13.445
0.2	23.277	25.356	27.264	27.12	25.463	24.971
0.3	34.519	36.827	39.377	39.188	37.181	36.675
0.4	45.578	47.954	51.103	51.637	49.45	48.611
0.5	56.592	58.551	61.514	62.667	61.08	60.425
0.6	67.948	69.679	71.413	70.922	69.04	71.172
0.7	78.313	79.754	81.558	80.16	75.789	80.902

(10 x 7) 2D 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/**.

Finding Items in the Calibration File

In a large calibration file, you might want to search for an item by name. To search the **Calibration File Contents** pane,

1 Click the **Calibration File Contents** 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 calibration (in the Tables or Feature views), 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.

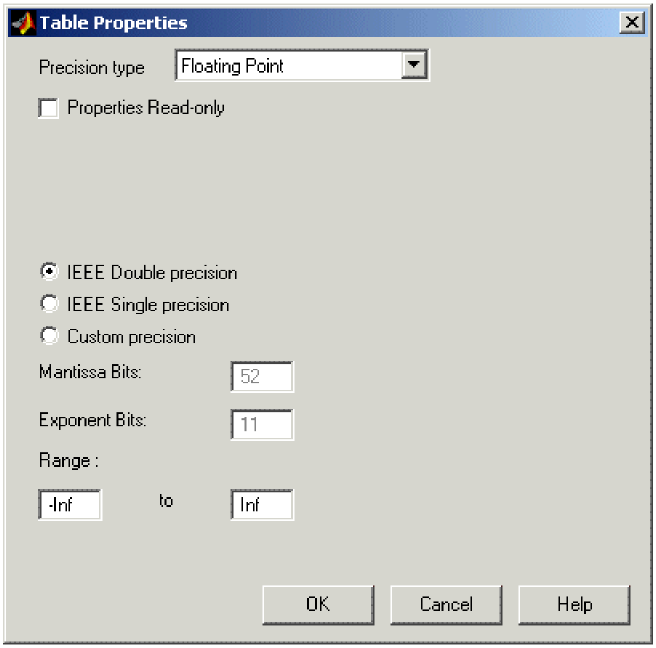


Table Properties

Precision type: Floating Point

□ Properties Read-only

● IEEE Double precision

○ IEEE Single precision

○ Custom precision

Mantissa Bits: 52

Exponent Bits: 11

Range : -Inf to Inf

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®, 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)

$$y = \frac{50x + 0}{0 + 255}$$

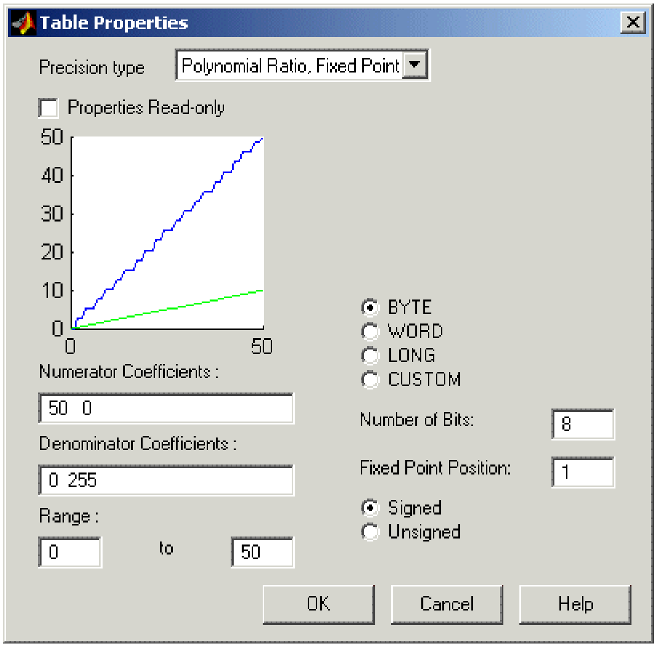


Table Properties

Precision type: Polynomial Ratio, Fixed Point

□ Properties Read-only

Numerator Coefficients : 50 0

Denominator Coefficients : 0 255

Range : 0 to 50

● BYTE
○ WORD
○ LONG
○ CUSTOM

Number of Bits: 8

Fixed Point Position: 1

● Signed
○ Unsigned

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

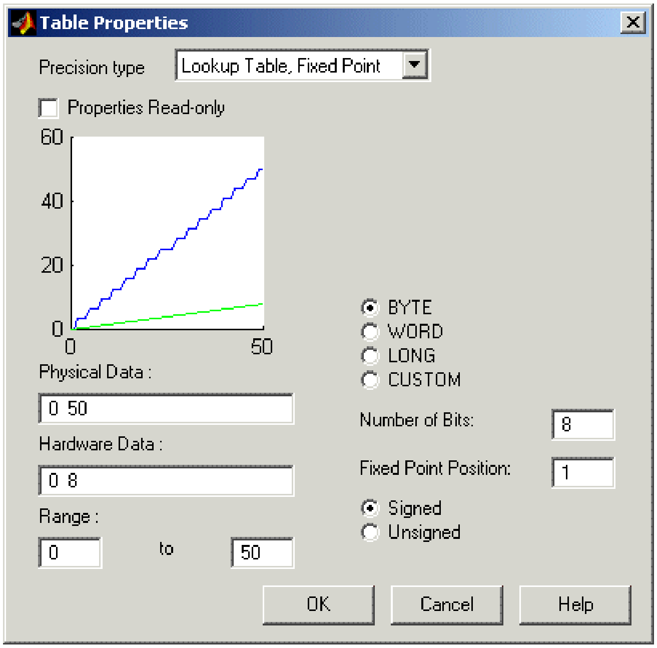


Table Properties

Precision type: Lookup Table, Fixed Point

□ Properties Read-only

Physical Data : 0 50

Hardware Data : 0 8

Range : 0 to 50

● BYTE
○ WORD
○ LONG
○ CUSTOM

Number of Bits: 8

Fixed Point Position: 1

● Signed
○ Unsigned

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.

14

Surface Viewer

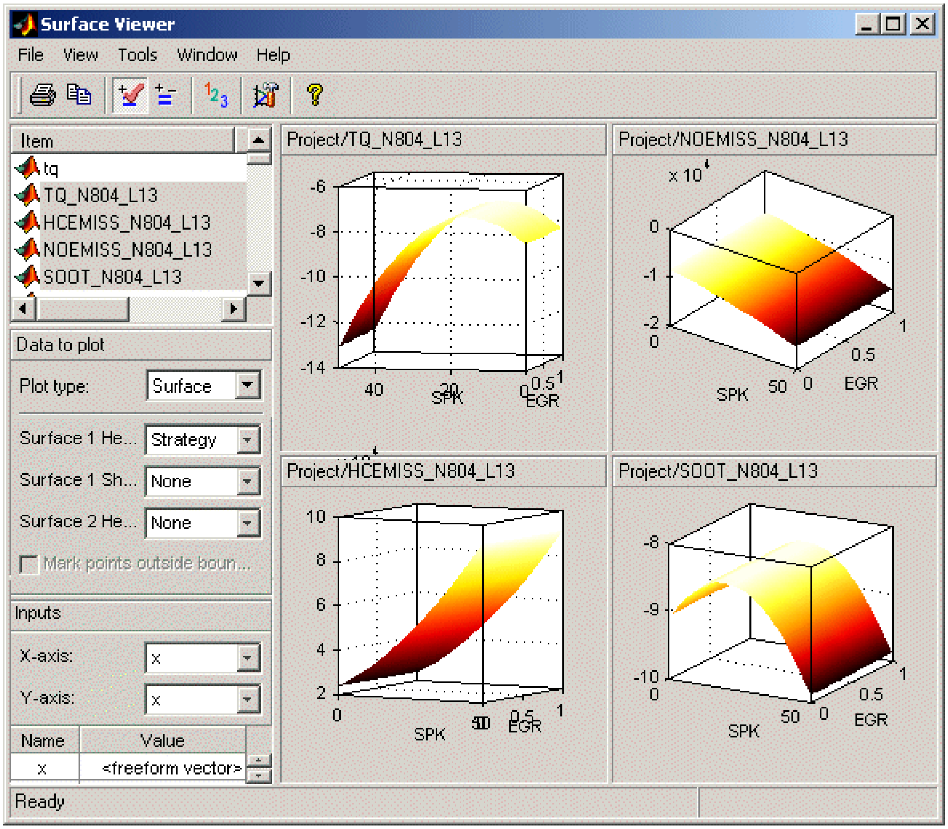
This section includes the following topics:

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, tradeoffs or the feature node of a feature calibration.

Following is an example of the **Surface Viewer** displays.



Plot type: Surface

Surface 1 He... Strategy

Surface 1 Sh... None

Surface 2 He... None

□ Mark points outside boun...

X-axis: x

Y-axis: x

Name	Value
x	<freeform vector>

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.

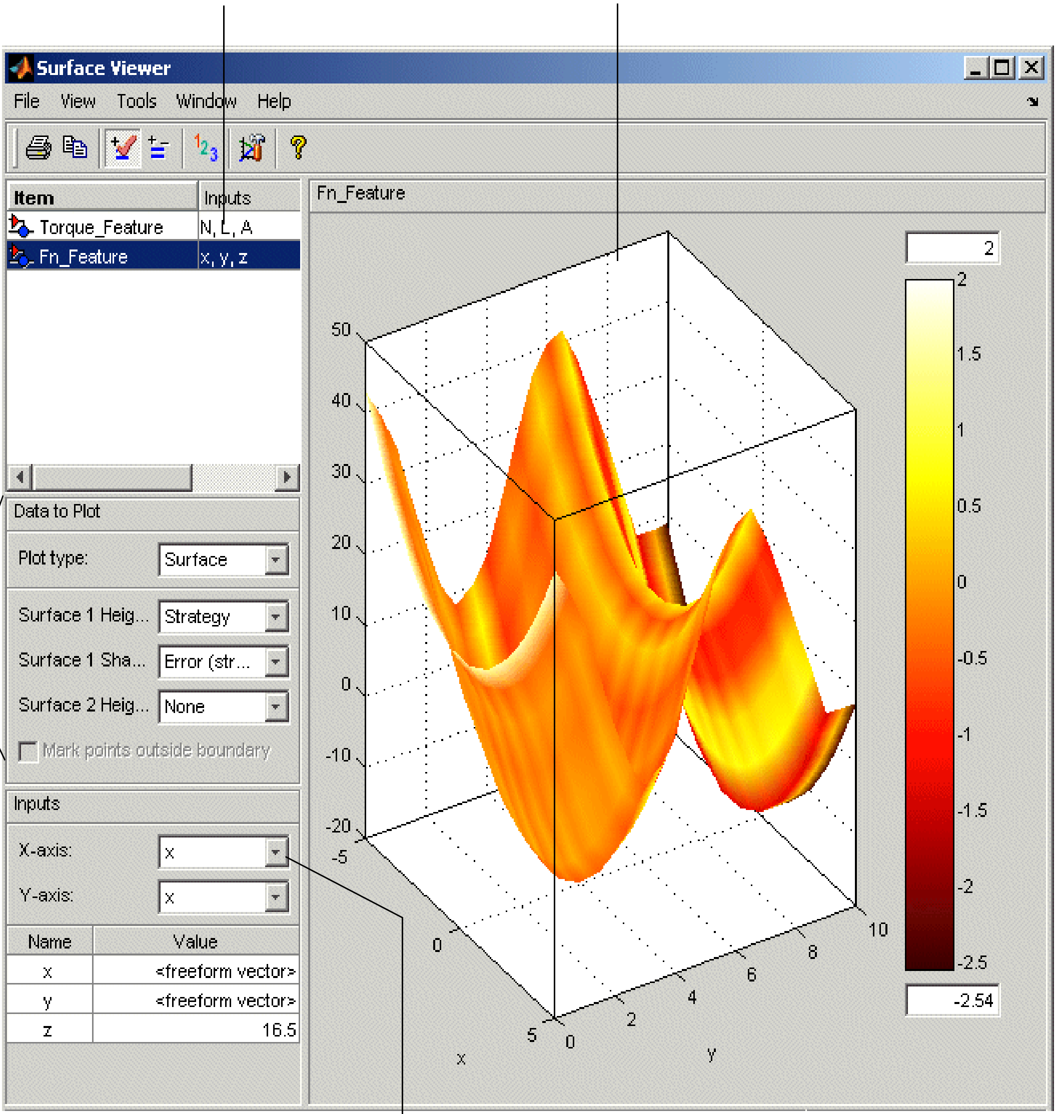
   For example, as you view a feature, you can view either the strategy, the model associated with that feature, the error between the model and the strategy, or the prediction error if the model was imported from the Model Browser. You can also use one of these factors to shade the surface formed by one 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.

Models or features in the project and their inputs

The model in the feature, shaded by the error (in this case)



Item	Inputs
Torque_Feature	N, L, A
Fn_Feature	x, y, z

Data to Plot

Plot type: Surface

Surface 1 Heig... Strategy

Surface 1 Sha... Error (str...

Surface 2 Heig... None

□ Mark points outside boundary

Inputs

X-axis: x

Y-axis: x

Name	Value
x	<freeform vector>
y	<freeform vector>
z	16.5

3. Plot controls

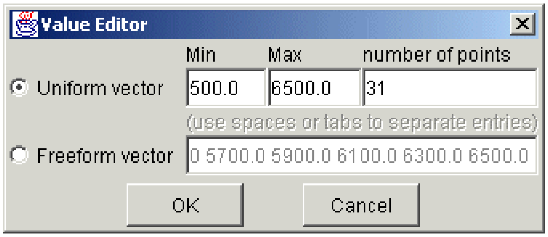
Variable ranges

Axes controls

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.



	Min	Max	number of points
● Uniform vector	500.0	6500.0	31

(use spaces or tabs to separate entries)

○ Freeform vector: 0 5700.0 5900.0 6100.0 6300.0 6500.0

Here you can indicate the points to include in the display. You can specify

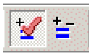
- The minimum and maximum values and the number of points across that range by choosing **Uniform Vector** and typing in the edit boxes **Min**, **Max**, and **Number of points**.
- Each discrete point at which you want to evaluate the model (or feature), by choosing **Freeform vector**, and then typing 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.

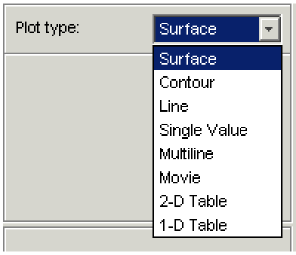
When you alter the variables, you can select whether you want the display to update automatically or not. You can toggle the automatic update on and off by selecting **Tools –> Auto-Evaluate**. When you want to update the display, select **Tools –> Evaluate Now**. Both of these options have equivalent toolbar buttons:



Displaying the Model or Feature

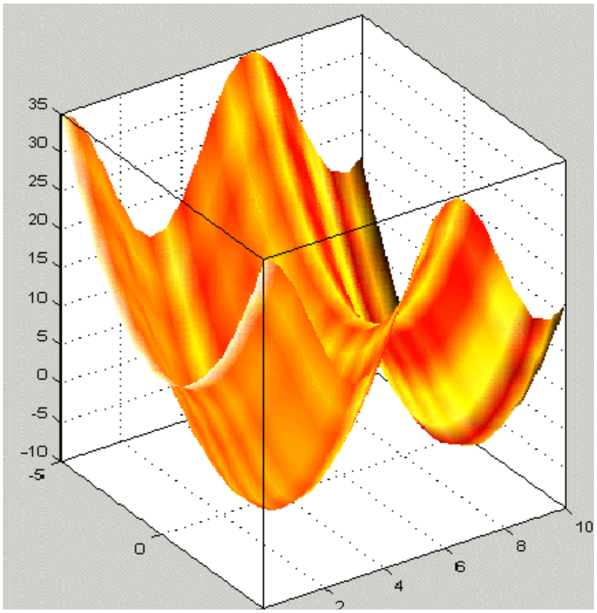
There are several aspects of the display to consider before making the movie.

- 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.



Plot type: Surface

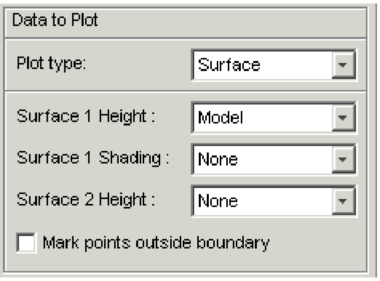
- Use the options in this menu to display the model or feature in the following ways:
  - Surface



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)

When viewing models and tradeoff models there are no strategy options. You can choose these options from the drop-down menus for **Surface 1 Height**, **Surface 1 Shading**, and **Surface 2 Height**, as illustrated below.



Data to Plot

Plot type: Surface

Surface 1 Height : Model

Surface 1 Shading : None

Surface 2 Height : None

□ Mark points outside boundary

You can view any of these options alone as a primary surface (by leaving the last 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.

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 above and below the color bar. You can see an example like this in “Viewing a Model or Strategy” on page 14-3.

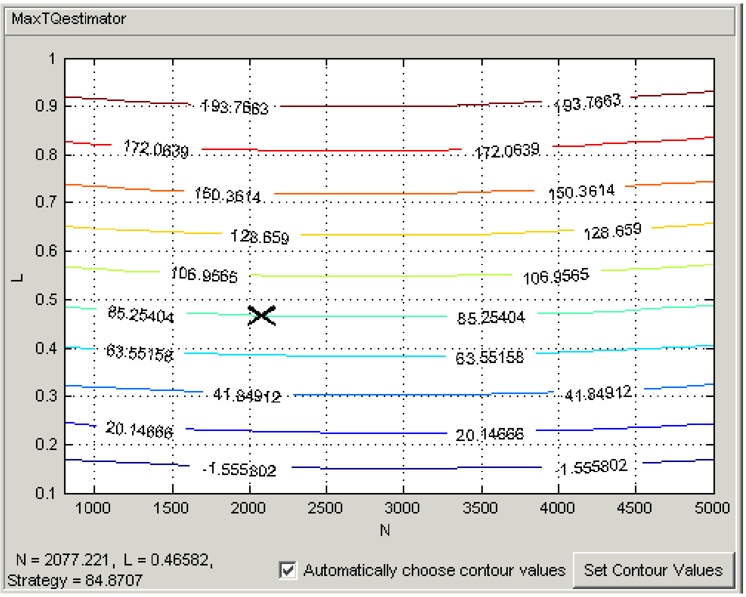
You can add a second surface to display any two of the options simultaneously, for example, your model and your strategy.

If you have a boundary model, you can view your models with the boundary marked by selecting the check box.

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

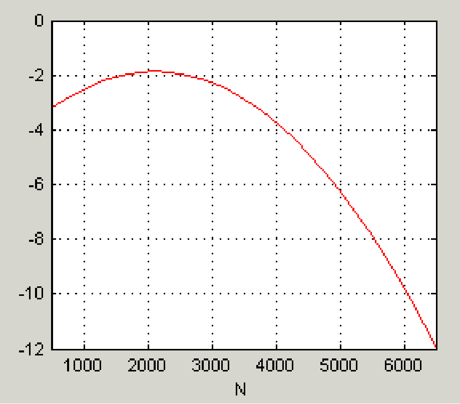


N = 2077.221, L = 0.46582,
Strategy = 84.8707

☑ Automatically choose contour values

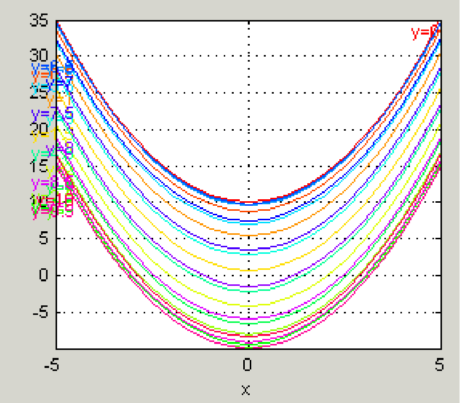
You can click on the plot to display the values at a point (plotted with an X), and you can specify where you want contours by clicking **Set Contour Values**. Use the check box to return to automatic contour value selection.

- A line plot - you can display up to three different lines (strategy, model, prediction error and error between the model and strategy). You can clip to a boundary if available.



- A single value

  This displays the value of the model, strategy, prediction error or error at the point you have specified in the variable value boxes.
- Multiline plot. You can clip to a boundary if available.



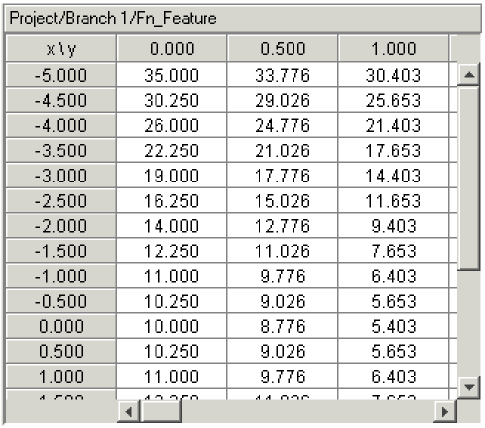
- Movie

  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. You can use the check box to mark boundaries if available.

- 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

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 **Surface 1 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.

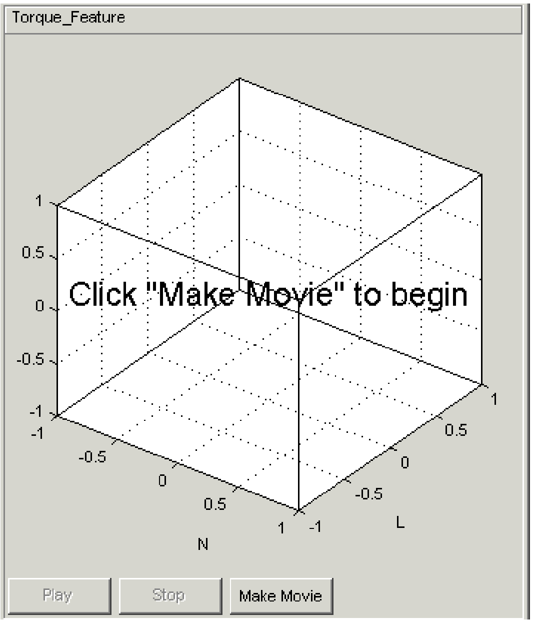
Prediction Error Variance (PEV) is a very useful way to investigate the predictive capability of your model. It gives a measure of the precision of a model’s predictions. PEV can also be examined in the Model Browser, both in the **Prediction Error Variance Viewer** and to shade surfaces in the **Model 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.

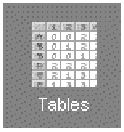
15

Manual Calibration and the History Display

This section includes the following topics:

Using the Tables View

Select this view by clicking the **Tables** button. It opens automatically if you add a table using the **File –> New** menu.



The **Tables** 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.

The next sections cover:

- “Adding and Deleting Tables” on page 15-3
- “Using the History Display” on page 15-6

See also

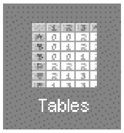
- “Table View” on page 9-31 for information on using the table view functionality once you have added tables to your project
- “Tutorial: Filling Tables from Data” on page 5-1

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 **Tables** view. CAGE automatically switches to this view if you add a table using the **File –> New** menu items.



The **Tables** 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.

   Adding new tables automatically switches you to the **Tables** view.

3 In the **Table Setup** dialog you can enter the table name, number of rows and columns and initial value, and select the input variable (or variables) from the drop-down menus.

4 Click **OK** to add the new table. CAGE automatically initializes the normalizers of the table by spacing the breakpoints evenly over the ranges of the selected input variables.

5 You can also select **Tools –> Calibration Manager** to change the size of a table. For information, see “Setting Up Tables” on page 13-2.

You can rename tables by first selecting the table, then

- Press **F2**, or
- Select **Edit –> Rename**.

You can manually calibrate by entering values in any table. You can also fill tables using experimental data or optimization output by going to the **Data Sets** view; see “Tutorial: Filling Tables from Data” on page 5-1.

Deleting Tables

To delete a table or a normalizer from a session,

1. Select **Tables** view.
2. Highlight the required table or normalizer.
3. Click ✕; or press **Delete**; or select **Edit –> Delete** *table_name* (‘*table_name*’ is the currently selected table).

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.

Duplicating Tables

To copy a table or a normalizer from a session,

1. Select **Tables** view.
2. Highlight the required table or normalizer.
3. Select **Edit –> Duplicate** *table_name* (‘*table_name*’ is the currently selected table).

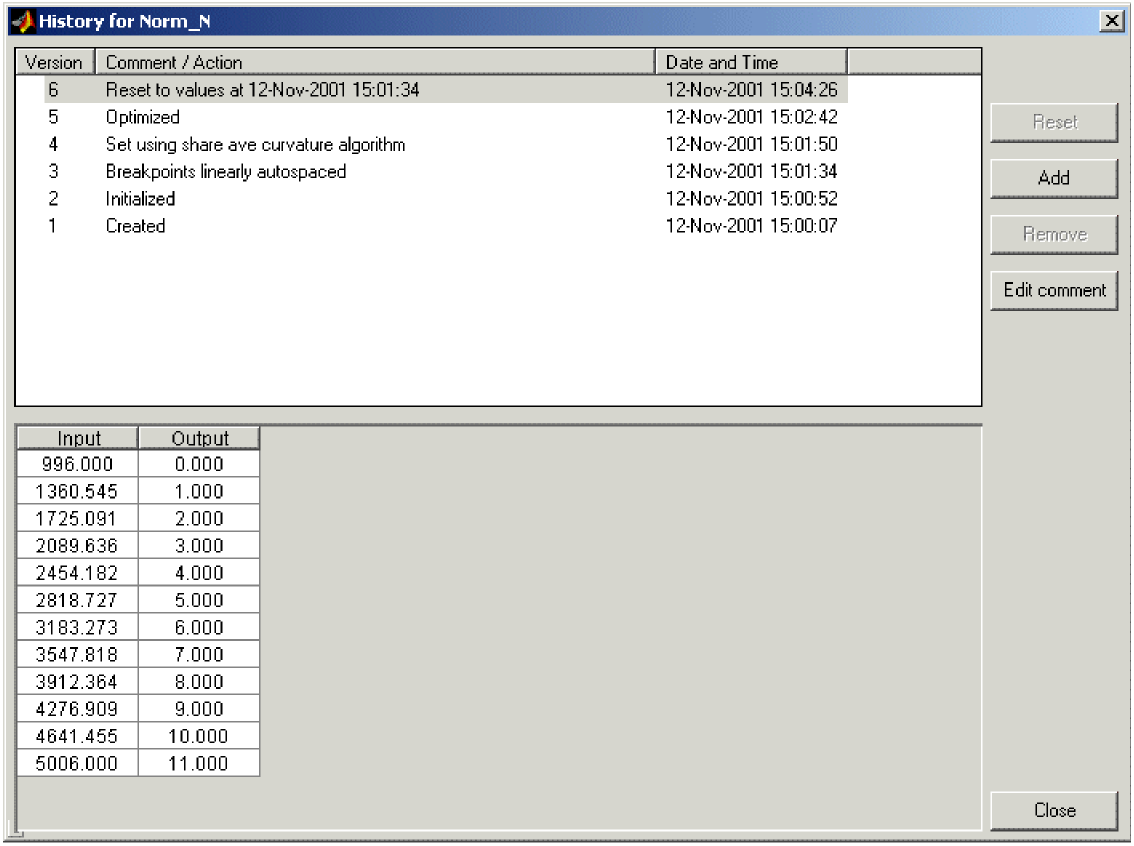
Using the History Display

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 15-8.)
- Compare different versions of tables and normalizers (See “Comparing Versions” on page 15-9.)

You can view the **History** display of a table or normalizer by selecting **View –> History**.



History for Norm_N

Version	Comment / Action	Date and Time
6	Reset to values at 12-Nov-2001 15:01:34	12-Nov-2001 15:04:26
5	Optimized	12-Nov-2001 15:02:42
4	Set using share ave curvature algorithm	12-Nov-2001 15:01:50
3	Breakpoints linearly autospaced	12-Nov-2001 15:01:34
2	Initialized	12-Nov-2001 15:00:52
1	Created	12-Nov-2001 15:00:07

Input	Output
996.000	0.000
1360.545	1.000
1725.091	2.000
2089.636	3.000
2454.182	4.000
2818.727	5.000
3183.273	6.000
3547.818	7.000
3912.364	8.000
4276.909	9.000
4641.455	10.000
5006.000	11.000

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, select **View –> History** to open the **History Display**.

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**.

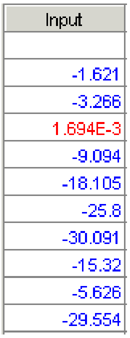
3 In the dialog box, edit the comment.

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 higher value in the later version.
- Cells that have blue entries have a lower value in the earlier version.



Input
-1.621
-3.266
1.694E-3
-9.094
-18.105
-25.8
-30.091
-15.32
-5.626
-29.554

CAGE Case Studies

This section includes the following topics:

Gasoline Example Study

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

Problem Definition

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

- Intake Cam Phase
- Exhaust Cam Phase

  as a function of Load and RPM, subject to the following constraint
- Constrain Exhaust Temperature <=1200C (to protect the catalyst)

Benefits of Automated Calibration

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

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

- Control problems
  - Injection timing and duration
  - EGR valve

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

Producing Optimized Calibrations

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

1. Load models of engine responses, decide on optimization strategy and define additional models — “Importing Models into CAGE” on page 16-4.
2. Set up tables — “Setting Up Calibration Tables to Fill” on page 16-5.
3. Set up an operating point set for the optimization — “Defining an Operating Point Set” on page 16-6.
4. Define optimization objective and constraints, and run the optimization — “Set Up and Run the Optimization” on page 16-7.
5. Fill tables from optimization results — “Filling Tables with Optimization Results” on page 16-9.
6. Use models and optimized tables to fill a torque estimator table — “Torque Estimator Problem” on page 16-10.

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

Importing Models into CAGE

1 Start CAGE by typing cage at the MATLAB command line.

2 Select **File –> Import –> Model**.

3 Locate the model file you exported from the Model Browser. We provide an example, Gasoline_models.exm. Locate the file in the mbctraining directory and click **Open**.

4 The Model Import Wizard appears, where you can select from a list of models in the file. Select the BTQ, EXTEMP, and knot models by **Shift**+clicking in the list.

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

5 Select the check box to **Automatically assign/create inputs** and click **Finish** to import the models.

   CAGE switches to the Models view, where you should see the BTQ, EXTEMP, and knot models in the list. The selected model is displayed in the other panes.

6 Select knot and press **F2** to rename the model (or right-click, or use the Edit menu). Change the name to MBT.

   The objective is to produce optimized tables for spark and cam timings, subject to an exhaust temperature constraint (to limit the temperature to a safe region for the catalyst). You want the optimization to search for the timings that give the best torque and minimum fuel consumption, subject to the constraint. You could fix the spark timing to be MBT spark (the spark angle that produces maximum brake torque) by using the MBT spark model as the spark input to the other models in the optimization. With spark fixed at MBT, you can set up an optimization that allows the two valve timings to vary, exploring the space via a gradient descent algorithm to locate the optimal timings. Remember this is a simplified problem and running an engine at MBT is knock-limited and so is not possible at all operating points. For this example you will not use the MBT model as you will use spark as a free variable in the optimization. You will return to the MBT model later.

Setting Up Calibration Tables to Fill

1 Set up a new 2-D table. Select **File –> New –> 2D Table.**

2 Name the table SPK.

3 Select L and N from the **Y** and **X Input** drop-down menus.

4 Leave the number of rows and columns at 10.

5 Leave 0 for the initial value.

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

CAGE switches to the Tables view, where you can see the new table and its normalizers in the Tables tree on the left. CAGE has automatically initialized the normalizers by spacing the breakpoints evenly across the range of the input variables N (speed) and L (load). Note that the normalizers appear as calibratable items in their own right, and as descendants (child nodes) of their tables.

Once you have created a table you can duplicate it to create more tables that share the same breakpoints.

7 Right-click SPK in the tree and select **Duplicate SPK**.

8 Click to select the new table, press **F2**, and rename the table INTCAM.

9 Right-click SPK in the tree and select **Duplicate SPK**.

10 Click to select the new table, press **F2**, and rename the table EXHCAM.

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

Defining an Operating Point Set

You need to define the set of points where you want the optimization to run. You use a data set to define these operating points.

1 Select **File –> New –> Data Set**.

2 Right-click on the following variables in the Project Expressions list and select **Add to Data Set** for each variable:

   a `N`

   b `L`

3 Click **Build Grid** in the toolbar. The **Grid Data** dialog appears.

4 Select `N`. To achieve 15 points, enter `zeros(1,15)` in the edit box and press **Enter**.

5 Click **OK** to accept this grid of 15 points. Click **View Data** in the toolbar to see the list of operating points in the data set. Edit the points as in the following table.

N	L
1000	0.3
2000	0.3
3000	0.3
4000	0.3
5000	0.3
4000	0.5
1000	0.5
1500	0.5
2000	0.5
3000	0.5
4000	0.5

5000	0.5
2500	0.6
3500	0.6
4500	0.7

Set Up and Run the Optimization

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

The objective is to maximize torque; therefore, this is a single objective optimization problem with a constraint.

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

1 Select **File –> New –> Optimization**. The **Optimization Wizard** appears.

2 Select `foptcon` and click **Next**.

3 Increase

   a The number of free variables to 3

   b The number of constraints to 1

   c The number of operating point sets to 1

   Click **Next**.

4 Select `S`, `EXH` and `INT` in turn for the free variables, click the button to select them, and click **Next**.

5 Select `BTQ` from the list of models on the right and click to select it for the objective. Select the **Maximize** radio button, and click **Next**.

6 Select `EXTEMP` from the list of models on the right and click to select it for the constraint. Enter 1290 in the edit box and press **Enter**. Ensure that the

expression reads `EXTEMP <= 1290` (to constrain the optimization to a safe operating temperature for the catalyst). Click **Next**.

7 Select the data set (the only one in the session) and click the button to select it for the optimization operating point set. Click **Finish**.

CAGE switches to the Optimization view. Look at the information displayed — here you can examine and edit the objective, constraint, and operating point set. Your optimization is ready to run.

8 Click **Run Optimization** in the toolbar.

The **Free Variable Set Up** dialog appears, where you can select starting values and ranges for the free variables (in this case, the spark and cam timings). This allows you to pick sensible starting conditions for the free variables. Sensible starting conditions can help avoid the solution from an optimization being trapped in a local minimum. The defaults are taken from the ranges and set points in the variable dictionary.

9 Edit the **Upper Bound** of the spark variable to `40`, to limit the range of this free variable, and click **OK**.

The optimization runs. You will see progress messages as each operating point is optimized.

10 When the optimization is complete, a new child node appears in the Optimization tree under the `foptcon` node. Click the plus sign next to `foptcon` to expand the tree, then select `foptcon_Output` to view the results.

11 Look through the solutions at different operating points by clicking cells in the output table. Regions that do not meet the constraint are yellow in the graphs. Notice that for higher speed points the optimization has much more limited space in which to find a solution within the constraint. This is expected at high speed and load; this data was all taken at stoich (where the air/fuel ratio is at 14.3, the stoichiometric constant) and this region requires increased richness of the air/fuel mixture to bring the exhaust temperature down to safer levels.

Because this is a single-objective optimization, there is only one solution at each point. With multiobjective optimization, you would have pareto plots available to help you select the best solution for each operating point.

Filling Tables with Optimization Results

1 Select **Solution –> Fill Tables**.

The **Table Filling Wizard** appears.

2 Click **Add** three times to add the SPK, INTCAM and EXHCAM_MBT tables to the filling list. Click **Next**.

3 Select S from the right list of optimization results, select the SPK table on the left, and click the button to associate the two. Repeat for INT and INTCAM; and EXH and EXHCAM. There is only one solution to fill the tables with, so you can click **Finish**.

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

4 Before you switch to the **Tables** view to view the filled tables, from the optimization output node click Export to Data Set in the toolbar. The table of optimization results is exported to a data set which you willl use later.

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

6 For this control problem, you might want a table of spark timings as specified by the MBT model to compare with your optimization results. First create a spark table:

a Switch to the **Tables** view.

b Right-click INTCAM and duplicate the table.

c Rename the new table SPK_MBT.

7 Switch to the **Data Sets** view. You want to use the MBT model to fill the new table, and you can do this by adding this model to a data set. You can use your data set containing the optimization results — this evaluates the model at the operating points specified, and when you fill the table the values are automatically extrapolated to fill the whole table.

8 Select the data set (named foptcon_sol1 after the optimization) and follow these steps.

9 Right-click MBT in the project expressions list and select **Add to Data Set**.

10 Click **Fill Table From Data Set** in the toolbar to switch to the table filling view.

11 Select the SPK_MBT table in the left list, and MBT in the right list. Look at the plots to see where the model evaluation will fill the table. Click **Fill Table**. The values are extrapolated across the whole spark table.

12 Switch to the **Tables** view to examine the newly filled spark table. Compare with the previous spark table. You will fill and optimize tables from models in more detail in the next section.

Torque Estimator Problem

The cam timings optimization results solved a control problem. You can also use CAGE to calibrate estimator problems. Here you will use a BTQ_MBT model to produce a torque estimator table.

1 You can convert tables directly to models in CAGE. In the **Tables** view, select the EXHCAM and select **Table –> Convert to Model**.

2 Repeat for the INTCAM table.

3 Click **Models** to go to the models view. You can set up function models to fill the torque estimator table at optimum cam settings.

4 To avoid confusion rename the new feature models (converted from tables) to EXHCAM_Model, INTCAM_Model.

5 Now you will create a torque model with the MBT model as the spark input. To do this.

   a Right-click BTQ and select **Duplicate BTQ** to make a copy of this model.

   b Select the copy, BTQ_1, and select **Model –> Edit Inputs**.

   c In the dialog that appears, select MBT in the right list of available inputs, and S in the left model factors list, and click the button to associate them. Click **Finish** to close the dialog.

   d Now look at BTQ_1 in the Connections diagram. Select **View –> Connections Graph –> Zoom To Fit** if you cannot see all of the diagram.

The MBT model replaces the spark input, so this model output is now torque at MBT.

e Select BTQ_1 and press **F2** to rename the model to BTQ_MBT.

6 Duplicate the MBT and BTQ_MBT models (right-click each to do this).

7 Rename the new models MBT_optcam and BTQ_optcam.

8 Now click MBT_optcam, and select **Model –> Edit Inputs**.

9 Select EXH in the model factors, and the EXHCAM_Model in the available inputs list, and click the button to associate them.

10 Repeat for INT in the model factors and INTCAM_Model.

11 Click **Finish** to close the dialog.

12 Now click BTQ_optcams, and select **Model –> Edit Inputs**.

a Select S in the model factors, and the MBT_optcam model in the available inputs list, and click the button to associate them.

b Repeat for EXH and EXHCAM_Model.

c Repeat for INT and INTCAM_Model.

d Click **Finish** to close the dialog.

Observe the changes in the Connections diagram. Select **View –> Connections Graph –> Zoom To Fit** if you cannot see all of the diagram.

Now you have a torque model with optimal cam timing inputs. You will use this to fill a torque estimator.

13 Select **File –> New –> Feature**. CAGE switches to the Feature view. Rename the feature MaxTQestimator.

14 First you need to select the model to fill the table.

a Click **Select Model**.

b Select BTQ_optcam and click **OK**. The model appears in the **Model** pane with its inputs N and L.

15 You need to create or import a strategy. To start with, you create the simplest strategy, a single maximum torque table you will fill from the BTQ_MBT model. Select **Feature –> Graphical Strategy Editor** or press **Ctrl**+**E**.

Three Simulink windows open: two libraries (with all the blocks available for building a strategy, and all the blocks in your CAGE session) and a new strategy window.

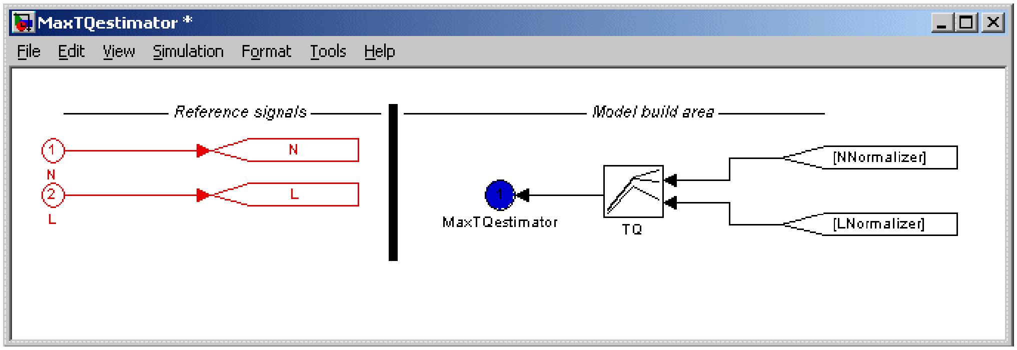
16 Select a Table block from the block library and drag it into your strategy window.

- **a** Rename the table TQ.
- **b** Click to select the TQ table, then hold **Ctrl** and click the blue MaxTQestimator outport to connect them.

17 You need to provide inputs for the table. To make the table inputs speed and load, you must add speed and load normalizers to your strategy and connect them to the table.

- **a** Double-click the Normalizers library block in the CAGE project library.
- **b** Drag the NNormalizer and LNormalizer blocks into your strategy.
- **c** Click to select NNormalizer, then hold **Ctrl** and click the TQ table to connect them. Repeat for LNormalizer.

Your strategy should look like this example.



18 Double-click the blue MaxTQestimator outport to parse the strategy into CAGE. You should see the following in the **Strategy** pane in CAGE:

```
MaxTQestimator = TQ(NNormalizer(N),LNormalizer(L))
```

19 Click to expand the MaxTQestimator feature. You can see that the TQ table with its speed and load normalizers has been added to the feature.

20 Select the TQ table in the feature tree.

You must specify the size of this table before you can go any further. You use the **Calibration Manager** to do this.

21 Click **Calibration Manager** in the toolbar (or use the **Tools** menu).

22 Make sure the TQ table is selected, then enter 10 in the rows and columns edit boxes and click **Apply**. The table is set up with the default 0 initial value in each cell. You have also specified the size of the N and L normalizers by setting up the table size. Click **Close** to dismiss the **Calibration Manager**.

You return to the table view in CAGE. Now you can see plotted in the comparison pane both the flat blue table (initialized with zeros) and the sloping model. The breakpoints of the normalizers are already spaced across the ranges of speed and load, because these normalizers were already in the project.

23 To fill the table from the model,

a Make sure the TQ table is selected.

b Click **Fill Table** in the toolbar, and the table is filled.

24 Observe the change in the comparison pane plot: the filled table is now very close to the model at this point.

In this case, because this is a simple strategy containing only one table, you can also fill a torque estimator table directly from the BTQ results in your optimization output data set. However the feature-filling technique results in a model for torque at optimal cam timings across the whole operating space, instead of only at the optimization output data set operating points.

For guidance, look at the example finished project, Gasoline_optimization.cag.

Diesel Example Study

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

Problem Definition

Produce tables in speed and torque for

Best injection timing	`soi`
Best fuel quantity	`basefuelmass`
Best fuel pressure	`fuelpress`
Best VTG	`grackmea`
Best EGR	`egrlft`

Minimize brake specific fuel consumption, subject to constraints on

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

Benefits of Automated Calibration

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

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

- Example Control Applications

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

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

Producing Optimized Calibrations

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

1. Load models of engine responses — “Importing Models of Engine Responses into CAGE” on page 16-16.
2. Define additional models and variables required by optimization strategy — “Defining Additional Variables and Models for the Optimization Strategy” on page 16-16.
3. Set up tables — “Setting Up Calibration Tables to Fill” on page 16-19.
4. Set up data set — “Define an Operating Point Set” on page 16-19.

5 Define optimization objective and constraints, and then run the optimization — "Set Up and Run the Optimization" on page 16-21.

6 Fill tables from optimization results — "Filling Tables with Optimization Results" on page 16-23.

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

Importing Models of Engine Responses into CAGE

Models are exported from the Model Browser as `.exm` files

1 Start CAGE by typing `cage` at the MATLAB command line.

2 Select **File –> Import –> Model**.

3 Locate the model file you exported from the Model Browser. We provide an example, `Diesel_models.exm`. Locate the file in the `mbctraining` directory and click **Open**.

   The **Model Import Wizard** appears, where you can select from a list of models in the file.

4 Click **Select All**.

5 Select the check box to **Automatically assign/create inputs** and click **Finish** to import the models.

CAGE switches to the Models view, where you should see the models in the list. The selected model is displayed in the other panes. Look at the models to verify the model inputs.

Defining Additional Variables and Models for the Optimization Strategy

Looking at the problem definition, you need to define some additional variables and models for this optimization. You want to minimize BSFC, but this is not a model output. You need to fill tables against speed and torque, but torque is not an input for the models. The solution to this is to create function models. You can then use a torque constraint and the BSFC function model in the optimization. To implement the torque constraint, you need to define a new variable for desired torque, tq_desired, and a new function model for torque error, tq_err, that is constrained to be 0.

Follow these steps:

1 Select **File –> New –> Variable Item –> Variable**. The Variable Dictionary view appears.

   a Rename the new variable tq_desired.

   b Set the range of this variable to be minimum 0 and maximum 1500

   c Edit the set point to 600 Nm.

2 Select **File –> New –> Function Model**.

   a Enter `tq_err = tq - tq_desired` and click **Next**.

   b Select **Automatically assign/create inputs** and click **Finish**.

   The Models view appears.

3 Select **File –> New –> Function Model**.

   a Enter `bsfc = 5400 / pi * basefuelmass / tq` and click **Next**.

   Assuming: base fuel mass [mg/inj], Tq [Nm], 6 cylinders, 4 stroke

   b Select **Automatically assign/create inputs** and click **Finish**.

Constraints are also required on the input space. Such constraints can be captured in the form of boundary models from the Model Browser, or can be implemented directly in CAGE. In this example, you implement the constraints directly by constructing additional function models as follows:

1 For each new function model, select **File –> New –> Function Model**. Construct the following models:

   a fuelpress_min as a function of rpm:

   ```
   fuelpress_min = 90+(measrpm-1600)/600*30
   ```

   b fuelpress_max as a function of rpm

   ```
   fuelpress_max = 110+(measrpm-1600)/600*50
   ```

   c basefuelmass_max as a function of rpm

```
basefuelmass_max = 200+(measrpm-1600)/600*-25
```

d `soi_min` and `soi_max` as a function of rpm

```
soi_min = -3+(measrpm-1600)/600*-6
soi_max = 3+(measrpm-1600)/600*-6
```

e `grackmea_min` and `grackmea_max` as a function of rpm

```
grackmea_min = 0.2+(measrpm-1600)/600*0.2
grackmea_max = 0.6+(measrpm-1600)/600*0.3
```

2 Now create some more function models that use these minimum and maximum models. You use these to constrain the optimization.

a `basefuelmass_over = basefuelmass - basefuelmass_max`

b `soi_over = soi - soi_max`

c `soi_under = soi_min - soi`

d `grackmea_over = grackmea - grackmea_max`

e `grackmea_under = grackmea_min - grackmea`

f `fuelpress_over = fuelpress - fuelpress_max`

g `fuelpress_under = fuelpress_min - fuelpress`

Constraints are also required on the output models. In optimization routines, it is good to avoid having any constraints whose order of magnitude is significantly larger than the other constraints. You can avoid this problem by rescaling such constraints.

For each new function model, select **File –> New –> Function Model**. Construct the following models:

3 `pkpress_over = (pkpress - 18e6)/18e6`

4 `vtgrpm_over = (vtgrpm - 128000)/128000`

This optimization also requires a constraint on the air/fuel ratio (AFR). However, as you could not model AFR directly you need to create a function model that relates equivalence ratio to air/fuel ratio. Then you can constrain AFR. To do this, construct the following function models:

5 `afr = 14.46/eqrexh`

6 `afr_under = afr_max - afr`

This function model creates a variable `afr_max`, which you can use to specify a different AFR maximum at each speed/torque point.

Setting Up Calibration Tables to Fill

1 Set up a new 2-D table. Select **File –> New –> 2D Table**.

2 Name the table `soi_best`.

3 Select `measrpm` and `tq_desired` from the **Y Input** and **X Input** drop-down menus.

4 Enter 11 rows and 11 columns.

5 Enter 0 for the initial value.

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

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

7 Once you create a table you can duplicate it to create more tables that share the same breakpoints. Click `soi_best` in the tree, then select **Edit –> Duplicate soi_best**. Repeat until you have four new tables, and rename them as follows:

a `basefuelmass_best`

b `fuelpress_best`

c `grackmea_best`

d `egrlft_best`

Define an Operating Point Set

You need to define the set of points where you want the optimization to run. You use a data set to define these operating points.

1 Select **File –> New –> Data Set**. CAGE switches to the **Data Sets** view.

2 Right-click and select **Add to Data Set** for these three variables in the **Project Expressions** list:

   a `measrpm`

   b `tq_desired`

   c `afr_max`

   This allows you to specify the speed and torque values you want in the data set, and also you can specify the maximum AFR at each N, TQ point.

3 Click **Build Grid** in the toolbar. The **Grid Data** dialog appears.

4 Select `measrpm`. To achieve 7 points, enter `zeros (1,7)` in the edit box.

5 Click **OK** to create a 7 point data set.

6 Click **View Data** in the toolbar to see the list of operating points in the data set.

7 Now you can edit the points in the table, to the values shown below. Double click on a cell to edit the value.

measrpm	`tq_desired`	`afr_max`
2200	1263	25.5
2200	947	27.75
2200	632	30.0
2200	126	0
1600	1550	22
1600	1163	22.5
1600	775	23

You are now ready to set up and run the optimization. You have set up tables to fill with the optimization results and defined a set of operating points and additional variables and function models. Click the **Optimization** button in the **Processes** pane to return to the **Optimization** view.

Set Up and Run the Optimization

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

The objective is to minimize BSFC; therefore, this is a single-objective optimization problem. All other considerations are constraints.

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

1 Select **File –> New –> Optimization**. The **Optimization Wizard** appears.

2 Select `foptcon` and click **Next**.

3 Here you set up your options as follows:

   a Increase the number of constraints to 12

   b Increase the number of free variables to 5

   c Increase the number of data sets to 1, and click **Next**.

4 Select `soi`, `basefuelmass`, `fuelpress`, `grackmea` and `egrlft` for the free variables and click the button to select them. Click **Next**.

5 Select `bsfc` from the list of models on the right and click the button to select this model for the objective.Click **Next**.

6 You will set up the constraints in the main optimization view, so just click **Next** on this screen.

7 Select the data set (the only one in the session) and click the button to select it for the optimization operating point set. Click **Finish**.

The **Optimization** view appears. You have not yet set up your constraints, so you can add them here.

You need to set up two types of constraint: equality and inequality. Inequality constraints are defined in the form “model, inequality, constant.”

For example, `NO` < 200

Equality constraints are defined in the form "model = constant."

For example, `tq_err` = 0

Equality constraints can be achieved using two inequality constraints, one <=, the other >=. This case study problem has CAGE model constraints on the following quantities (which you will set up below):

- `tq_err`
- `vtgrpm`
- `pkpress`
- `afr`
- `soi`
- `grackmea`
- `fuelpress`
- `basefuelmass`

Set up these constraints as follows:

1 Double-click to edit the first constraint. The **Constraint Editor** appears.

2 These are all `Model` constraint types (select `Model` from the drop-down menu).

3 Select the model `tq_err`, select the inequality <=, and enter `0` in the edit box for constraint bound. Click **OK** to return to the Optimization view.

4 Similarly set up `tq_err >= 0`.

5 Repeat these steps to create

   a `soi_over <=0`

   b `soi_under <=0`

   c `grackmea_over <=0`

   d `grackmea_under <=0`

   e `fuelpress_over <=0`

   f `fuelpress_under <=0`

   g `basefuelmass_over <=0`

h `pkpress_over <=0`

i `afr_under <=0`

j `vtgrpm_over <=0`

6 Now you can run the optimization. You have set up objectives, constraints and an operating point set. Click **Run Optimization** in the toolbar.

The **Free Variable Set Up** dialog appears, where you can select starting values and ranges for the free variables. This allows you to pick sensible starting conditions for the free variables. Sensible starting conditions can help avoid the solution from an optimization being trapped in a local minimum. The defaults are taken from the ranges and set points in the variable dictionary.

Click **OK** to accept the default starting values for the free variables, and the optimization runs. You will see progress messages as each operating point is optimized.

7 When the optimization is complete, a new child node appears in the Optimization tree under the `foptcon` node. Click the plus sign next to `foptcon` to expand the tree, then select `foptcon_Output` to view the results.

8 Look through the solutions at different operating points by clicking cells in the output table. Compare with the gasoline example, where regions that do not meet the constraint are yellow in the graphs — in this case the equality constraints cause all the graphs to be yellow, because there is only a solution at a single point.

Filling Tables with Optimization Results

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

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

The **Table Filling Wizard** appears.

2 Click **Add** repeatedly to add all your tables to the filling list:

- `soi_best`
- `basefuelmass_best`
- `fuelpress_best`

- grackmea_best
- egrlft_best

3 Click **Next**.

4 Match up the following pairs of an output from the right list of optimization results with a table on the left, and in each case click the button to associate the two:

   - soi with soi_best
   - basefuelmass with basefuelmass_best
   - fuelpress with fuelpress_best
   - grackmea with grackmea_best
   - egrlft with egrlft_best

5 There is only one solution to fill the tables with, so you can click **Finish**.

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

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

Compare with the gasoline example; see “Filling Tables with Optimization Results” on page 16-9.

Look at the example finished project, Diesel_optimization.cag.

Index